Apphosting med Kubernetes

En jämförelse mellan 2 lösningar för apputveckling i Kubernetes

Examensarbete

InformationsTeknik

2022

Tobias Holm

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsanalys
Identifikationsnummer:	
Författare:	Tobias Holm
Arbetets namn:	Apphosting genom Kubernetes
Handledare (Arcada):	Fredrik Welander
Uppdragsgivare:	

Sammandrag:
Syftet med arbetet är att utbilda och upplysa vad kubernetes kluster är, hur strukturen fungerar och jämföra populära alternativ och diskutera för- och nackdelar i processen. Att hosta applikationer och program på Kubernetes blir allt vanligare i dagens läge och rekommenderas för framtida företag att använda sig av. Kubernetes kluster och swarmar har under senaste år blivit ett alternativ som flera nya företag samt äldre företag har sett som den nya lösningen för deras applikationer. Men att testa och jämföra olika alternativ tar tid och extrapengar som firmor inte kan undvara. Därför ska detta arbete fungera som en guide för andra att ta beslut gällande ibruktagning av kubernetes. Arbetet kommer att innehålla en utvidgad guide för drifttagning av applikationer i de båda lösningarna. Arbetet kommer att ta upp kubernetes, docker och applikationers driftsättning. Jag kommer att gå igenom för och nackdelarna med varje delmoment och vilka typer av applikationer som kan passa bättre med vilka lösningar.

Nyckelord:	Kubernetes, docker, Appvärd, applösning, openshift, kubeadm
Sidantal:	25
Språk:	svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information analysis
Identification number:	
Author:	Tobias Holm
Title:	Apphosting with Kubernetes
Supervisor (Arcada):	Fredrik Welander
Commissioned by:	

Abstract:	

The purpose of this thesis is to educate and enlighten the public what Kubernetes and clusters are, how the structure is set up and to compare popular methods of setting it up, discussing pros and cons of these methods. Hosting applications through Kubernetes clusters are becoming more common and the new solution for both new companies looking to get their app on the market and current companies to migrate their app hosting to.

Kubernetes clusters and swarms have become a more relevant solution for companies to host their applications. But to test which method is best suited for them takes time and effort that companies do not have extra time to put towards. That's where this thesis comes in to enlighten readers which method might suit them the best.

This thesis covers an extended guide of commissioning applications with both methods, the thesis includes Kubernetes, docker and commission of applications. The thesis covers pros and cons of every step on both methods and what type of application that works better with which method.

Keywords:	Kubernetes, docker, Application hosting, application solution, Openshift, KubeADM
Number of pages:	25
Language:	Swedish
Date of acceptance:	

INNEHÅLL

Figurer

1 INLEDNING

1.1 Kubernetes Historia

Kubernetes är ett projekt med öppen-källkod först lanserat 2014 under namnet ”project 7”. Projektet byggde på det gamla ”borg” systemet från google och utvecklingsteamet bestod av gamla Google-utvecklare. Kubernetes officiella första version 1.0 släpptes 2015. (ferenc harmoni, 2022)

Kubernetes skapades originellt av Google-utvecklare. Kubernetes nämndes originellt till projekt 7 internt i Google, detta refererade till de 7 pinnarna på logon, men nämndes sedan till kubernetes inför utsläppet av applikationen. Originellt skrevs kubernetes systemet i

c++ men skrevs om i Golang som det körs med nuförtiden. Kubernetes 1.0 släpptes ut på internet 21 juli 2015 och under 2016 kom funktioner såsom helm pakethanteraren och intern alarmeringssystem. (ferenc harmoni, 2022)

Kubernetes är ett verktyg beskrivet som ”Dirigerade Containeriserade kluster av värdar för hosting av användarens applikation” För att bryta ner detta koncept så pratar man om ett verktyg för automatisering, uppskalning och hantering av applikationer i containers. Kubernetes kluster används för att hantera applikationer av mindre och större skala och följa upp förbruknings mängden på applikationen och kan enkelt användas med t.ex grafana, graphite och rancher för olika sorters övervakning med enkla integrationer för ett alarmsystem på dessa kluster till din epost och slack. (What is Kubernetes, 2022)

Kubernetes fungerar endast som ett verktyg för hantering av dina applikationer på nätet, själva hosten kan vara helt vad användaren vill, de flesta molnplattformer stöder kubernetes applikationer, populära alternativ såsom Azure, AWS, google cloud services, rancher UI m.fl.

Kubernetes kluster bygger och skalar upp användarens applikationer baserat på mängden CPU-kärnor, minne eller specifierade parametrar som användaren förbestämt. Parametrar som datalagring och kontakt mellan andra datasystem sköts med t.ex Kubernetes egna API. (What is Kubernetes, 2022)

1.2 KubeADM

KubeADM är utvecklat av kubernetes som ett lätthanterligt verktyg för att guida användare genom uppsättningen av deras applikation och sätter upp ett MVP-kluster med möjlighet att lätt installera tillägg som monitorering, kubernetes administrationssida, och molntjänster.

KubeADM kommer som ett paket som låter användaren ha ett fungerande MVP-program i ett kluster inom minuter. En simpel användare kan enkelt köra igång ett Kubernetes kluster i KubeADM med säkerhets checkar från KubeADM som granskar slutprodukten före den får grönt ljus att köras igång.

KubeADM är ett väldigt lätt verktyg för att hosta applikationer i kluster och kan därför enkelt användas på svagare hårdvara såsom en Raspberry PI. Till skillnad från ett standard Kubernetes-paket så kommer KubeADM med en mindre mängd funktioner och expansionspaket som ett switch-system för att konfigurera trafiken mellan olika servrar och webbsidor, vad KubeADM ger är ett lätthanterat, portabelt kluster system för användarens applikationer eller webbplats och fungerar som stomme för integration med olika verktyg som kubespray. KubeADM rekommenderas dock ej som lösning för produktionskluster tack vare dess avsaknad av uppskalning/autoskalning

1.3 OpenShift

Openshift är ett verktyg utvecklat av RedHat för hantering och producering av kubernetes applikationer och containers. OpenShifts fokus ligger på att ge företag en långtidslösning för värd av deras applikation eller webbsida. Systemet är byggt för att fungera lika enkelt oberoende om det körs helt i molnet, helt lokalt eller hybrid. Användar-gränssnittet fungerar likadant och det finns enkla metoder att installera tillägg såsom övervakning, containers, och länka till olika molntjänster. (simplilearn, 2022)

Openshift till skillnad från standard Kubernetes ligger i själva upplägget, Openshift är delvis byggt på docker. Openshift kommer med inbyggd säkerhet med IP-adresser och brandväggsregler, ett övervakningssystem, en centraliserad firma policy-hanterare och lätta integrationer med Kubernetes.

Openshift har utanför sitt egna gränssnitt även kommandon för avancerade uppsättningar och kan skapa enkla template frameworks med GO, node.js, java, PHP eller python.

Några av de största skillnaderna mellan Kubernetes och Openshift bygger på Openshifts utvecklare RedHat. Skall användare köra Openshift istället för Kubernetes måste man

låsa sig till RHELAH, eller Red Hat Enterprise Linux Atomic Host. Med andra ord RedHat’s egna Linux distro.

Openshift kommer med ett säkerhetspaket för att snabbare komma igång med kluster, men paketet kommer med strängare regler gällande Kubernetes upplägget. T.ex. Containrarna I poddarna krävde en skilld användare istället för root som körde kommandot I containrarna. Openshift kommer dock inte med inbyggd autentiserings funktion utan utvecklarna måste själva skapa tokens för autentisering.
Kubernetes uppdateringsstöd är även bättre med stöd för uppdateringar I delar när nya versioner släpps medan Openshift kräver uppdatering av hela paketet ifall användare vill ha en nyare version.
Kubernetes kommer inte med en standard nätverksbrandvägg eller switch-system medan Openshift kommer med Open-vSwitch, ett öppet-källkod program för OpenShift som kommer med en konfigurerad brandvägg med moderna nätverksstandarder. (OpenvSwitch, 2016)

1.4 Syfte

Syftet med detta arbete går ut på att informera läsaren vad skillnaden mellan KubeADM och OpenShift är och varför dessa bör tas i beaktande vid implementation av framtida webbservrar eller applikationer. Detta arbete ämnar sig både för fritidsintresse samt företagsledningar att läsa för att få en djup inblick i kubernetes-miljön och en klar bild över skillnaderna mellan KubeADM och Openshift för att enklare kunna ta beslut om vilken produkt som passar deras ändamål. Detta arbete kommer att behandla vad kubernetes är, vad man kan göra med kubernetes och hur man kör igång en enkel webbserver i båda programmen.

2 FUNKTIONALITET

Kubernetes fungerar som ett system för att publicera din applikation eller webbsida modernt och med utmärkt långvarigt stöd. Tack vare användningen av microtjänster inom klustren sköt Kubernetes snabbt i höjden för övervakningssystem för företags it-applikationer och hemsidor för att enklare kunna hantera alameringsregler och få larm till e-post, slack eller rakt till telefon.
En enskilld nod i klustret består av flera olika delar, ”Kubelet” är ett extraverktyg som kan styra instanserna av varje enskilld nod, håller koll på deras hälsa, tillstånd och startar om dem i deras rätta stadie ifall noden eller podden inte gör det effektivt. ”Kube-proxy” är en inbyggd proxy som styr alla jobb och balanserar förbrukningen mellan noderna baserat på IP-adress och portnummer. ”Containern”, dvs. en docker image med parametrar från en dockerfil som tillsammans kan bygga upp valfritt programbas för olika kodspråk. Containrar håller basnivån, själva applikationen, importerade bibliotek och deras behov. Allting styrs av kubernetes kontrollern som finns inbyggd i varje kubernetes service som körs på nätet. Kontrollern är masternoden i systemet som kör och styr arbetsfördelningen och kommunikationen mellan noderna och nätet. System som kräver en högre drifttid kan även ha flera Masternoder som delar på arbetet. (Brendan Burns, 2018)

Kontrollern består av vissa olika delar, ”etcd” sköter den tillfälliga datan och vad som händer. Etcd kan visa aktuell info gällande klusterfördelning, arbeten, mängden resurser som används och informationen därifrån kan enkelt skrapas från olika övervakningsprogram som grafana.
API servern är Kubernetes egna inbyggda API som sköter kommunikation till och från olika tjänster som REST och andra noder eller kluster. API server baserar sig på etcd’s data.
”Schedulern” eller schemaläggaren, söker hela tiden efter noder som har mindre eller inga jobb och delegerar aktiva jobben som körs till den noden.
Inne i kontrollern finns en mindre kontroller som loopar igenom alla noder och deras stadier och arbetar med API’n för att skapa, uppdatera och ta bort resurser och småjobb som körs. Det finns ett antal av olika kontrollers, såsom replikationskontrollern, som styr

replikation och upp- eller nerskalning av noderna. Det finns också daemon-noden, som kan individuellt köra enstaka poddar på olika maskiner som är kopplade till klustret.

Ett Kubernetes ekosystem består av kluster som sköter de olika applikationerna och kommunikationen till internet och andra kluster, varje kluster består av master och arbetsnoder. Masternoderna sköter infrastrukturen och styr arbetsfördelningen mellan arbetsnoderna. Arbetsnoderna sköter alla arbetsuppgifter som kommer in på rad varje sekund, medan masternoderna fördelar arbetet så ingen nod skall bli överbelastad så utför noderna alla uppgifter som dom får. (Kubernetes Docs, 2022)
Sist i arbetspyramiden kommer då poddarna, som kommunicerar och samarbetar med varandra mellan noderna för att få arbetet gjort så användarens kluster kan funktionera enligt deras egna konfigurationer.

2.1.1 Nätverksprotkoll

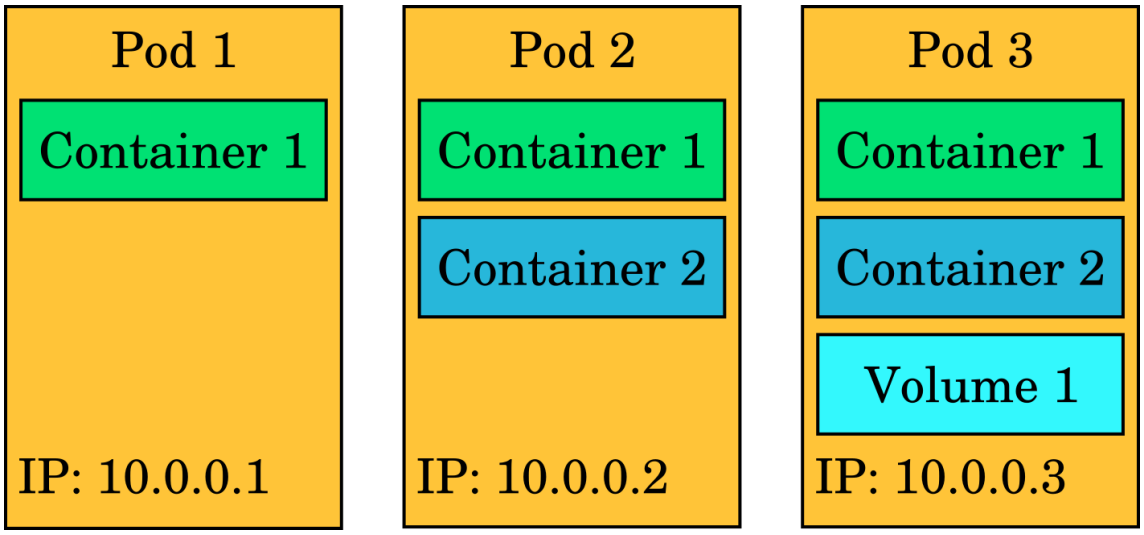
KubeADM tack vare sin lätta konfiguration har ett inbyggt skript som bygger ut IP-banor och mappar ut DNS för klusternätverket. Dock finns det möjlighet att mata in egna parametrar vid konfigurationen för en helt skillt inställd konfiguration. Ett populärt alternativ för att konfigurera nätverk i en KubeADM miljö är Calico Nätverksinstickningsmodul. Calico bygger på kubernetes miljön och har samma typ av YAML konfigurationsparametrar för uppsättningen äv nätverket dock krävs en omstart för att applicera ändringar efter att man körd igång yaml filen. (Dimuthu De Silva, 2019)

Openshift har en mer företagsinriktad tankegång på deras uppbyggnad av nätverket i klustren, konfigurationen kräver lite mer arbete för att det skall fungera.
IP-adresser och intervaller måste själv ställas in samt i fallet av en webapplikation med både front- och backend så måste backend’en byggas om först med den nya IP-adressen för att applicera ändringarna rätt. RedHat har dock en automatisk DNS-räknare liknande till KubeADM’s system.

2.1.2 Image och Poddar

Utanför detta Kubernetes ekosystem finns själva byggfilerna som kallas ”image”. Image kommer från dockerfile som kubernetes miljön bygger på. Alla parametrar och konfigurationer för en Kubernetes basmiljö matas in i en dockerfil som sedan bygger en ”image”. Dessa ”images” kan laddas upp till olika register t.ex DockerHub eller Azure container registry.

Poddar är små ”sceduling units” som sköter det direkta arbetet i noderna. Poddarna består av olika docker containers. Poddarna kan samarbeta med varandra även om de existerar i olika noder. Varje nod har en egen IP-adress inne i klustret vilket ger den rätten att enkelt dela arbeten mellan noderna utan risk för konflikt. Poddarna kan prata med noderna och varandra utan problem inom klustren och klustren pratar med varandra genom klusterspecifika IP-adresser och portar. Men om en container i en podd vill prata med en annan container i en annan podd så används poddens egna IP-adress. För att ekosystemet skall användas på större skala så rekommenderar Kubernetes att använda ’ephmeral’ system istället för att använda hårdkodade IP-adresser utan specifierade IP-adresser och portar som kan användas. (CSC Docs, 2022)



Figur 1 Ett exempel på hur ett poddsystem kan se ut med kluster och fästa skivor. (CSC Docs)

2.1.2.1 KubeADM

Inne i en ”image” sitter all binär data som styr klustret och alla integrationers moduler. Dessa ”images” byggs ihop och publiceras till ett register före användaren skapar kontakten till applikationer från de olika poddarna.

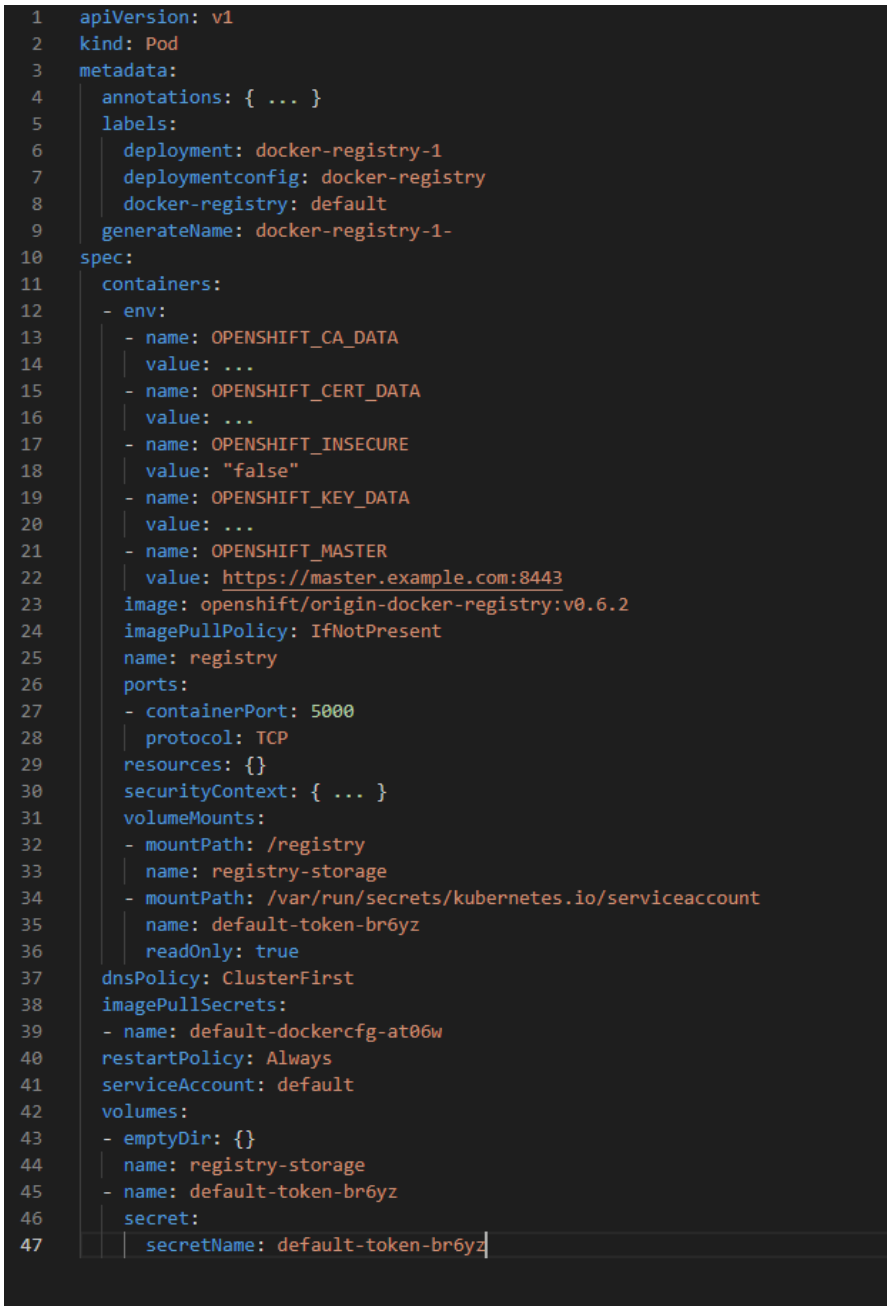
Poddarna sköter arbetsuppgifterna. Alltså när det kommer ett anrop från ett annat kluster eller service så ges uppgiften till en av poddarna som har mindre arbetsuppgifter på kö.

```yaml
---
apiVersion: v1
kind: Pod
spec:
  containers:
  - name: webserver
    image: registry.access.redhat.com/rhscl/nginx-112-rhel7
    ports:
    - containerPort: 8080
      protocol: TCP
    volumeMounts:
    - name: website-content-volume
      mountPath: /usr/share/nginx/html
    volumes:
    - name: website-content-volume
      persistentVolumeClaim:
        claimName: web-content-pvc
```

Figur 2 Ett exempel på en web-server pod. (csc docs)

2.1.2.2 OpenShift

Openshift Poddar och images är mera utbyggda för större webservrar och långtidsprojekt och har därför en mer utförligare uppläggning för hur poddarna och klustren konfigureras från installation. Systemet gällande hur poddarna kommunicerar och arbetsuppgifter tilldelas fungerar dock på samma sätt som i KubeADM.



Figur 3: *En kubernetes podd i openshift som konfigurerats för långtidskörning.*

Dessa poddar kan bli namngivna med olika ”tags” för att enklare gruppera poddarna enligt vilken del av applikationen dom skall arbeta med. I bilden ovan ser vi ett exempel på detta med Labels taggen som ger 2 parametrar för uppladdning av imagen i poddarna till ett register samt genererar ett namn vilket ges som prefix till varje ’image’-namn som skapas.

Vidare skapar vi 4 olika containers för att hålla olika typer av data såsom certifikaten, Kommunikationen mellan poddarna och nätet samt säker och osäker data. Podden får ett span av portar för att arbeta och sätta up IP-adresser för containrarna med TCP protokoll. Sen appliceras flera förvaringspunkter med tillhörande sökvägar för att hitta filen eller mappen som datan skall skrivas till. Slutligen sätts DNS och omstartnings parametrarna och hela förvaringsdriven appliceras till podden. (”Architecture Core concepts, Containers and images”, 2022)

2.1.3 Produkten

2.1.3.1 KubeADM

KubeADM’s produktide siktar in sig på att ge användaren ett enkelt sätt att ha en enkel bas som kan köras på lättare hårdvara såsom en Raspberry eller en billig hyrd server.

2.1.3.2 OpenShift

Openshift från RedHat annonserar sig själva som den bästa lösningen för större företag som söker en mer permanent lösning för hosting av deras applikation. Openshift kräver lite mera uppsättning och konfiguration men erbjuder en stabil kraftfull bas men lätta kopplingar till integrationer och större uppskalning.

2.1.4 Tidsestimat för MVP

2.1.4.1 KubeADM

Estimerat 30-45 minuter för en fullständing basinstallation utan större problem. KubeADM kräver att all installation görs manuellt som initialisering av master och arbetsnoderna, spanet av IP-adresser och portar. Inkörning av valfritt baspaket till själva klustret räknas inte med eftersom dessa kan variera kraftigt beroende på vad som installeras till klustret. Klustret kan enkelt utvidgas med flera noder och poddar efteråt samt med olika integrationer

2.1.4.2 OpenShift

Estimerat 15-20 minuter, OpenShift's basinstallation kommer med ett färdigkonfigurerat paket som kan köras in direkt i ett tomt kluster var det finns ett större utbud på färdiga miljöer att köra igång och börja konfigurera applikationen eller webbservern som skall köras igång i klustret. OpenShift har enkla integrationsmöjligheter med andra RedHat produkter och servicer men kan också kopplas till samma integrationer som KubeADM, dock inte lika smidigt.

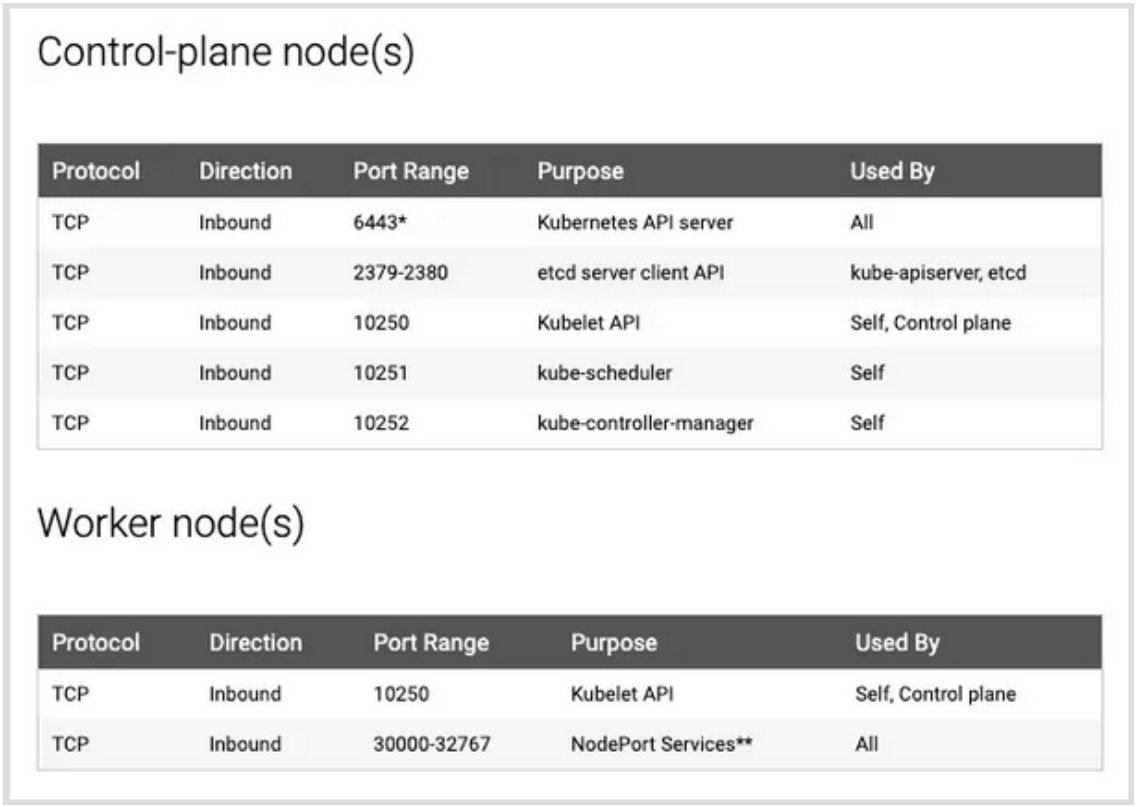
3 METODEN

3.1 Syfte

En enkel webbserver kommer att sättas upp i varsin miljö, en med KubeADM och en med OpenShift, Båda miljöerna kommer att bygga på en Ubuntu 20.04 basinstallation.
Arbetet kommer att jämföra de olika skeden och kommentera på skillnaderna i produkterna. Resultatet kommer att innehålla jämförelser såsom svårighetsgrad, tidskrav, konfigurations möjligheter och utvidgningsmöjligheter till olika integrationer och servicer.

3.2 KubeADM

Det första steget för att få ett fungerande KubeADM instans är att stänga av 'swap' på alla noder och köra in ett script i /etc/fstab för att hålla swap modulen avstängt vid möjlig omstart av servern. Kubernetes är inte programmerat att använda datan som existerar i swap filerna och därför finns en risk för dataläckor. (outcoldman 2017)



Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

Figur 4: Port konfigurationen för detta KubeADM kluster.

Sedan installeras containerd för enklare hantering av containrarna tillsammans med docker och docker's CLI. Vi konfigurerar vår docker daemon att köra hela tiden med att skriva dens konfiguration till 'systemd'. Sen startas docker daemon'en.

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

Figur 5: Ett exempel på hur konfigurationen kan se ut

Först initieras masternoden med IP-adress, ett span med IP-adresser för arbetsnoderna, namnge nodgruppen eller ändra filtret av felmeddelanden som kan skippas över under initieringen.

Initieringen körs som root så efteråt bör konfigurationsfilen kopieras till en vanlig användares hemmamapp. Vi kan verifiera noderna med ett kubectl kommando som är ett av kubernetes integrationerna som kubeadm bygger på.

```
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                 READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-88z28             0/1     Pending   0          9m54s
coredns-558bd4d5db-c9gr4             0/1     Pending   0          9m54s
etcd-master-node                     1/1     Running   0          10m
kube-apiserver-master-node           1/1     Running   0          10m
kube-controller-manager-master-node  1/1     Running   0          10m
kube-proxy-lh5bv                     1/1     Running   0          9m54s
kube-scheduler-master-node           1/1     Running   0          10m
vagrant@master-node:~$
```

Figur 6 Ett exempel på hur ett poddsystem kan se ut.

KubeADM kommer inte med ett färdigkonfigurerat nätverksinstickningsmodul men en stark rekommendation för en lösning är ”Calico network plugin”. Det är lätt att sätta upp och integreras enkelt och effektivt med kubeADM. Calico kommer med en färdig YAML fil som man applicerar till KubeADM med ett kubectl kommando. Efter appliceringen bör man se 2 nya poddar i klustret med att kontrollera nätverksmiljön.
Efter detta kan masternoden kopplas fast i arbetsnoderna

```
vagrant@master-node:~$ kubectl get nodes
NAME           STATUS     ROLES                  AGE   VERSION
master-node    Ready      control-plane,master   55m   v1.21.0
worker-node01  NotReady   <none>                 6s    v1.21.0
```

Figur 7: Exempel på hur nodupplägget ser ut.

Samma kommando kan användas flera gånger för att koppla ihop flera arbetsnoder till masternoden.
Efter detta kan en ”metrics-server” installeras för att kontrollera nodernas arbetsuppgifter, hälsa och annat. KubeADM kommer inte med ett färdig

kommande, men ett liknande kommando till den som kopplade noderna ihop kan användas. Metrics systemet körs in från en github repo med ’**kubectm apply**’ kommandot.

Efter någon minut är systemet installerat och poddarnas process- och minnesförbrukning kan monitoreras genom **top** kommandot.

```
vagrant@master-node:/vagrant$ kubectl top nodes
NAME            CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master-node     161m         8%     1578Mi          41%
worker-node01   53m          5%     739Mi           39%
```

Figur 8: Ett exempel på nodförbrukningen på ett system med 2 CPU cores och 4gb minne.

I detta skede är KubeADM konfigurerat med en master och en arbetsnod, ett nätverkssystem och en övervakningstjänst. I detta skede är det dags att installera applikationen eller webservern som ska köras på systemet. För detta exempel körs det ett nginx basapplikation för vidare konfiguration av en webbserver.

```
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
EOF
```

Figur 9 En enkel nginx basapplikation

Efter kommandot för igångkörningen av nginx applikationen surfar man in på nätadressen för klustret för att verifiera att allting fungerar.

3.3 OpenShift

Mycket likt till KubeADM's installation så bör docker installeras och startas igång med 'systemctl'. Openshifts installation stänger swap färdigt vid installation så man behöver inte adressera det problemet skillt.
Sedan installeras OpenShift paketet i sin helhet från OpenShift's github repo, Filerna packas upp och kubectl konfigurationen flyttas till systemets egna mapp under /usr/local/bin/. Till skillnad från KubeADM så kommer OpenShifts paketinstallation med kubectl och med deras beroende subsystem färdigkonfigurerade så man behöver endast kopiera filerna till systemet som klustret ska köra på installationen.
Verifiera installationen med 'oc version' kommandot för att verifiera att allting installerats rätt och vilka versioner.

```
oc v3.11.0+0cbc58b
kubernetes v1.11.0+d4cacc0
features: Basic-Auth GSSAPI Kerberos SPNEGO
```

Figur 10: Ett exempelutskrift av oc version kommandot.

OpenShift, som tidigare nämnt är konfigurerat och specialiseras för större företagsmiljöer som söker långtidslösningar och har behov av större kluster med hög tillgänglighetsfaktor. Denna version av OpenShift som installeras för detta test kommer från en del av RedHats demoversion för testning. Därför skall man konfigurera om docker daemonen i denna openshift miljö för att godkänna icke-certifierade docker register med { "insecure-registries" : ["172.30.0.0/16"] }, starta sedan om docker servicen med systemctl
I detta skede har OpenShift installerat ett kluster med 1 masternod innehållande 2 CPU cores och 4gb minne och en arbetsnod med 1 CPU core och 2gb minne.

Klustret kan köras igång med ’oc cluster up –IP’. När alla moduler och instickningsmoduler laddats in kan man logga in till web-gränssnittet med standard adminkontot system:admin

```
In project default on server https://your-server-ip:8443

svc/docker-registry - 172.30.1.1:5000
  dc/docker-registry deploys docker.io/openshift/origin-docker-registry:v3.11
    deployment #1 deployed 2 minutes ago - 1 pod

svc/kubernetes - 172.30.0.1:443 -> 8443

svc/router - 172.30.94.157 ports 80, 443, 1936
  dc/router deploys docker.io/openshift/origin-haproxy-router:v3.11
    deployment #1 deployed 2 minutes ago - 1 pod

View details with 'oc describe /' or list everything with 'oc get all'.
```

Figur 11: En exempelkörning på openshifts projekt status

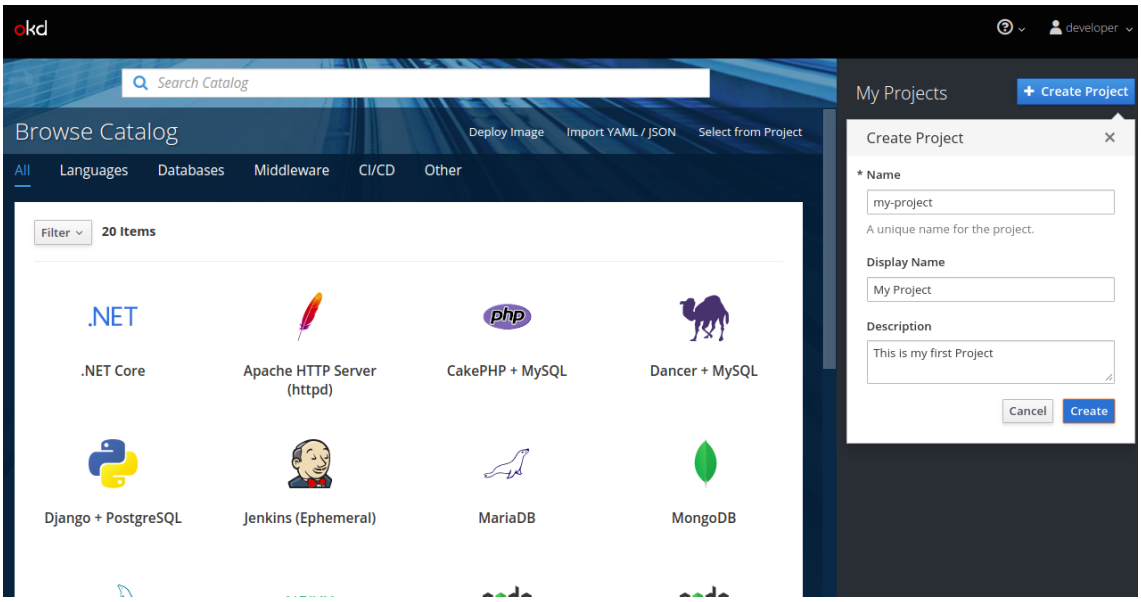
OpenShift, som tagits upp tidigare i detta arbete, kommer med en färdigkonfigurerad grund som låter företagsledningar och nya kunder enkelt och effektivt integrera sina produkter i klustermiljön med minimal nertid för användare. OpenShift kräver därför mycket mindre arbete för att få igång ett likadant kluster som KubeADM eller Kubernetes basen. Dock har OpenShift tillgång till ett liknande nodsystem som KubeADM.

```
Logged into "https://your-server-ip:8443" as "system:admin" using existing credentials.

You have access to the following projects and can switch between them with 'oc project ':

    default
    kube-dns
    kube-proxy
    kube-public
    kube-system
  * myproject
    openshift
    openshift-apiserver
    openshift-controller-manager
    openshift-core-operators
    openshift-infra
    openshift-node
    openshift-service-cert-signer
    openshift-web-console

Using project "myproject".
```

Figur 12: Ett exempel på OpenShift’s nodmiljö

Genom OpenShift's webb-gränssnitt skapas sedan basapplikationen som används, var användaren sedan kan köra in sitt egna projekt eller bygga vidare på basen man valt att använda för sitt projekt.

För att skapa ett eget projekt i OpenShift måste man först komma åt webb-gränssnittet med oc login och ge ett användarnamn samt lösenord som skall användas och sedan skapa ett nytt projekt med **"oc new-project dev --display-name="Project - Dev" --description="My Project".**
Sedan kommer man åt sitt projekt genom en webbläsare och att surfa till **serverip:port/console,** logga in med samma användare och lösenord som till projektet. I detta skede kommer en sådan bild emot efter att man klickat på "create projekt" uppe i höger kant:



Figur 13: Bas för OpenShift projekt.

Hoppa sedan tillbaka till konsolen och skapa projektet med 'oc project <namn>' och verifiera projektet med 'oc status'. Som exempel används ett simpelt image exempel från OpenShift's Docker Hub register. Vi laddar in och taggar imagen med:

```
oc tag --source=docker openshift/deployment-example:v2 deployment-example:latest
```

Taggen används för att enklare köra igång och stänga ner projektet utan att varje gång använda det långa genererade namnet från Docker Hub. Namnet på taggen blir i bilden ovan "deployment example" Nu kan 'oc new-app deplyment-examples'

köras, så skapas appen och knuffas in i klustret. För en full vy av information gällande det nybyggda klustret och applikationen körs ’oc get svc’ som returnerar all relevant data från klustret i simpelt format.

```
Name:              deployment-example
Namespace:         my-project
Labels:            app=deployment-example
Annotations:       openshift.io/generated-by=OpenShiftNewApp
Selector:          app=deployment-example,deploymentconfig=deployment-example
Type:              ClusterIP
IP:                172.30.87.146
Port:              8080-tcp  8080/TCP
TargetPort:        8080/TCP
Endpoints:         172.17.0.10:8080
Session Affinity:  None
Events:
```

Figur 14: En exempelkörning för klusterinformation.

Se applikationen i webbläsaren från https://deployment-example-projekt.IP.dy.fi

4 RESULTAT

4.1 KubeADM eller OpenShift

Baserat på resultaten tidigare i detta arbete kan man se skillnanderna i användningsområden mellan KubeADM och OpenShift. Medan KubeADM främst fungerar som en lärande miljö för användaren var man med relativt få resurser kan skapa ett baskluster för skolprojekt eller mindre uppbyggda kluster medan OpenShift fokuserar sig på företagsbilden av klusteruppbyggnaden.

Frågan gällande vilken produkt som bättre passar läsarens egna behov kommer inte som ett rakt svar från detta arbete utan läsaren själv bör ta beslut baserat på tidigare punkter i arbetet. KubeADM är ett väldigt lättdrivet system som kan användas för på t.o.m en liten Raspberry pi för att dra en första webbserver eller skolprojekt men kan fortfarande fullt utnyttja den enkla uppskalningsfunktionaliteten i Kubernetes för att hantera plötslig ökning av trafik utan att ta ner hela sidan för ens några minuter utan tilläggande av processkraft eller minne kan sättas in manuellt och poddarna startar om sig i ordning för att uppdatera alla utan att klustret någonsin är nere. Dock kan det lättdrivda systemet snabbt stöta emot problem med en större mängd trafik som en webbsida som plötsligt får en större boost av uppmärksamhet

och trafik. Speciellt på en Raspberry PI eller en mindre hemmaserver kan långtidsanvändare stöta på dessa problem med KubeADM.
OpenShift kommer med ett enklare installationsmedium i fråga av en färdigkonfigurerat medium som man kör in på sitt valda system och genom webb-gränssnittet kan man enkelt lägga upp flera olika applikationer och webbservrar helt enligt behov, OpenShift är mera marknadsfört och drivet för större företag i långtidsdrivna miljöer och oftast på kraftfullare servrar än 8 cpu cores. OpenShifts uppskalningsparametrar är optimerade i en större skala så att även vid 16 CPU cores och 128gb minne så skall klustren sömlöst kunna uppgradera sig själva för en webbapplikation som 100 personer använder konstant utan att någon rapporterar några större problem eller långsammare service. Openshift som producerats av RedHat fungerar och integreras bäst med andra RedHat produkter och större mätinstrument såsom Grafana och RancherUI. KubeADM kan integreras enkelt med alla möjliga öppna integrationsmoduler dock brukar endast mindre moduler behövas för dom enkla webbservrar och program som körs på KubeADM.

4.2 Slutdiskussion

Installationsprocesserna för båda programmen gick relativt enkelt med egen erfarenhet men för en nybörjare utan docker eller klusterkunnande kan en första installation av KubeADM visa sig vara ett större projekt att ta sig an. OpenShift fungerar enklare för större projekt var användaren redan tagit sig an Kubernetes och förstår sig på klustermiljön. Installationen För OpenShift må verka enklare men att sedan köra igång systemet för en längre tid och arbeta ut problem som uppstår kan visa sig vara en stor utmaning för den hårt integrerade miljön i OpenShift.

5 KÄLLOR / REFERENCES

“Architecture Core concepts, Containers and Images”, 2020. Tillgängligt: https://docs.okd.io/3.11/architecture/core_concepts/containers_and_images.html#architecture-core-concepts-containers-and-images Hämtad 1.6.2022

CSC docs, 2021, tillgängligt: https://docs.csc.fi/cloud/rahti/concepts/ Hämtad 1.6 2022

Kubernetes dokumentation, 2022. Tillgängligt: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/ Hämtad 1.6.2022

OpenvSwitch, 2016 tillgängligt: https://www.openvswitch.org/ Hämtad 1.6.2022

” Running a Kubernetes cluster with calico, Dimuthu De Silva, 2019. Tillgängligt: https://medium.com/platformer-blog/running-a-kubernetes-cluster-on-ubuntu-with-calico-9e372fb9175e Hämtad 1.6.2022

Simplilearn, 2021, Tillgängligt: https://docs.okd.io/3.11/architecture/core_concepts/containers_and_images.html#architecture-core-concepts-containers-and-images Hämtad 1.6.2022

Swap module ,outcoldman 2017 tillgängligt: https://github.com/kubernetes/kubernetes/issues/53533 hämtad 10.5 2022

“The history of Kubernetes” Ferenc Hamori, 2022. Tillgängligt:

https://blog.risingstack.com/the-history-of-kubernetes/ Hämtad 14.2 2022

” The History of Kubernetes and the Community behind it”, Brendan Burns 2018. Tillgängligt https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes-the-community-behind-it/ Hämtad 1.6.2022

”What is Kubernetes, 2022. Tillgängligt https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ Hämtad 1.6.2022