

Kubernetes (K8s)

»Wer zum Geier sind Sie?« –

»Ich bin der, der Ihnen sagt wo es langgeht.«

»Get Shorty«, USA 1995

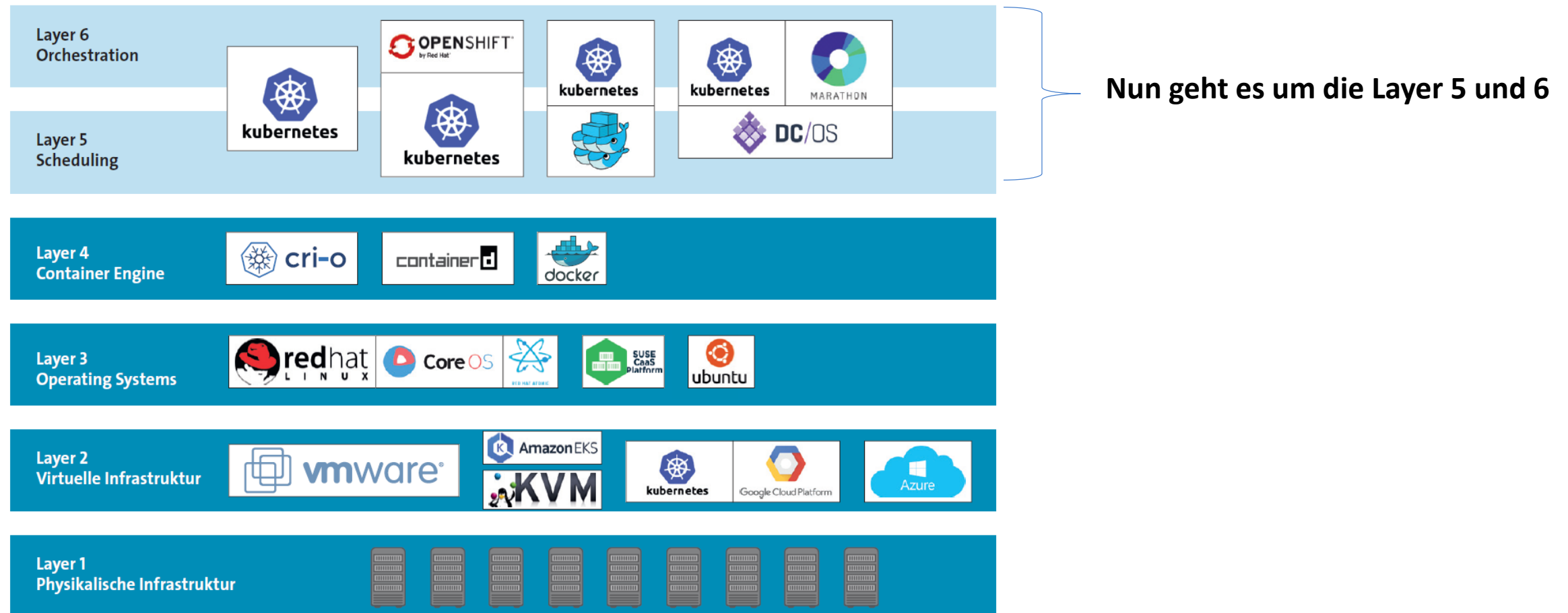
Lernziele

- ★ Sie haben einen ersten Einblick in Kubernetes

Zeitlicher Ablauf

- ★ Die (vereinfachten) »Layer« der Container-Welt, Layer 5 und 6
- ★ Kubernetes (K8s) im Überblick und Merkmale
- ★ K8s-Komponenten
- ★ K8s-Dienste auf den Master und Worker Nodes
- ★ Networking in Kubernetes
- ★ K8s Master- und Worker-Setup
- ★ Übung
- ★ Reflexion
- ★ Lernzielkontrolle

Die (vereinfachten) »Layer« der Container-Welt



Kubernetes (K8s) im Überblick



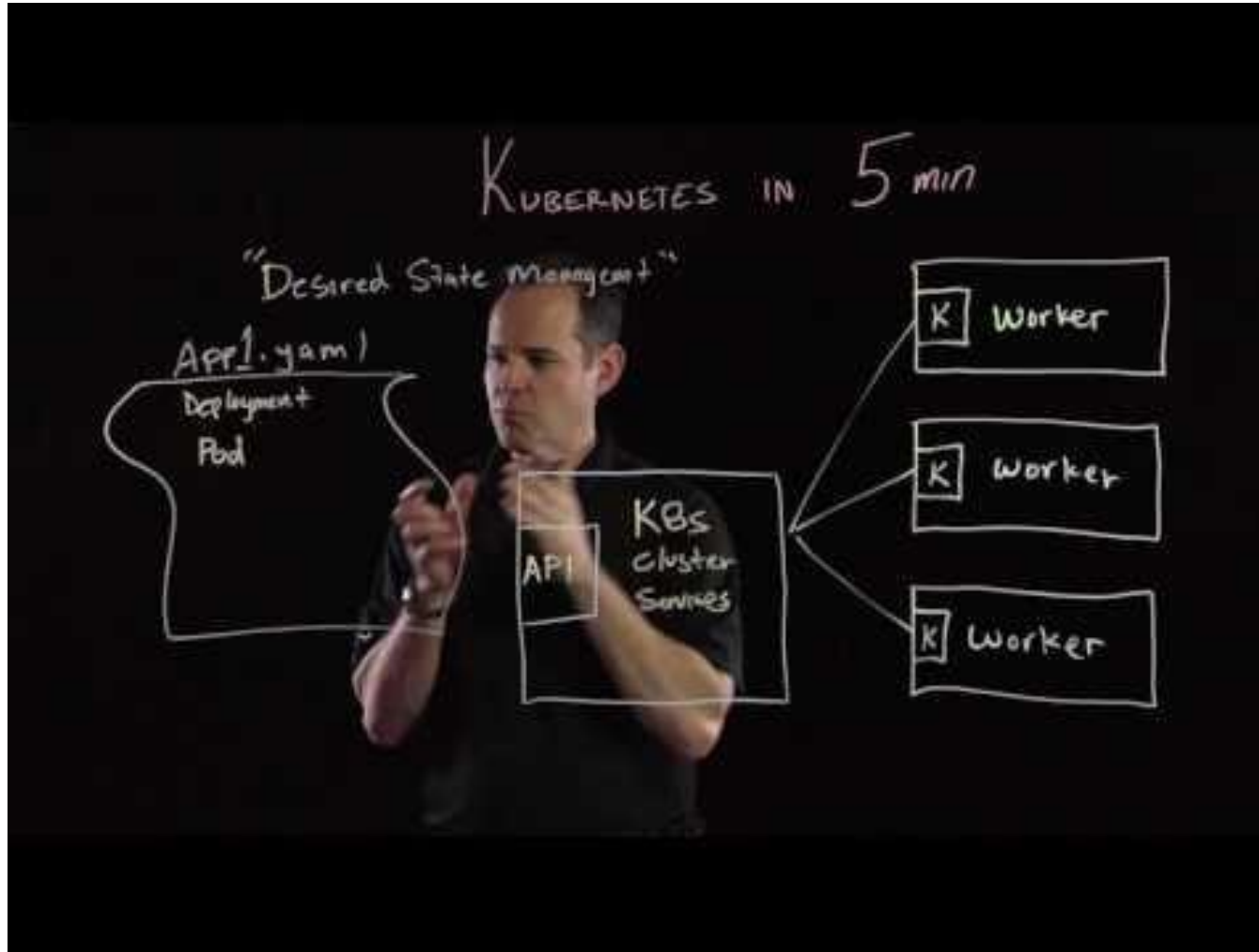
- ★ Das im Juli 2014 gestartete (griechisch: Steuermann) stellt die derzeit **populärste** Container-Cluster-/Orchestrierungs-Lösung dar.
- ★ Kubernetes ist mittlerweile bei der **Cloud Native Computing Foundation (CNCF)** gehostet.
- ★ Kubernetes' Hauptaufgabe ist die **Verwaltung und Orchestrierung der Container** innerhalb eines Clusters, der üblicherweise aus mindestens einem Kubernetes Master und multiplen Worker Nodes besteht.
- ★ Kubernetes wurde von Google ursprünglich als Orchestrierungs-Framework unter der Code-Bezeichnung Borg initiiert. **Ziel** war es, Docker- oder Rocket-(CoreOS)-**Container** im **grossem Umfang** zu **deployen**, zu **administrieren** und transparent zu **orchestrieren**.
- ★ Da Google Kubernetes bzw. seine Google-interne Implementierung davon selbst einsetzt, ist relativ sicher, dass K8s **kein kurzlebiges Projekt** sein wird.
- ★ **K8s arbeitet primär mit Docker-Containern.**

Kubernetes version	Release month	End-of-life-month
v1.6.x	March 2017	December 2017
v1.7.x	June 2017	March 2018
v1.8.x	September 2017	June 2018
v1.9.x	December 2017	September 2018
v1.10.x	March 2018	December 2018
v1.11.x	June 2018	March 2019

Kubernetes (K8s) Merkmale

- ★ Immutable (Unveränderlich) statt Mutable.
- ★ Deklarative statt Imperative (Ausführen von Anweisungen) Konfiguration.
- ★ Selbstheilende Systeme - Neustart bei Absturz.
- ★ Entkoppelte APIs – LoadBalancer / Ingress ([Reverse Proxy](#)).
- ★ Skalieren der Services durch Änderung der Deklaration.
- ★ Anwendungsorientiertes statt Technik (z.B. Route 53 bei AWS) Denken.
- ★ Abstraktion der Infrastruktur statt in Rechnern Denken.

Kubernetes: in 5 Minuten



Kubernetes unter Windows



! Hinweis

Ein vor kurzem aktualisierter Linux-Computer ist erforderlich, um nachvollziehen; Kubernetes master-Ressourcen wie Kube-Dns, Kube-Scheduler und Kube-Apiserver nicht zu Windows noch portiert wurden.

💡 Tipp

Die Linux-Anweisungen sind für **Ubuntu 16.04** zugeschnitten. Andere Linux-Distributionen, die zum Ausführen von Kubernetes-Zertifizierung sollten auch entsprechende Befehle anbieten, die Sie ersetzen können. Sie können auch erfolgreich mit Windows ausgeführt werden.

K8s-Komponenten

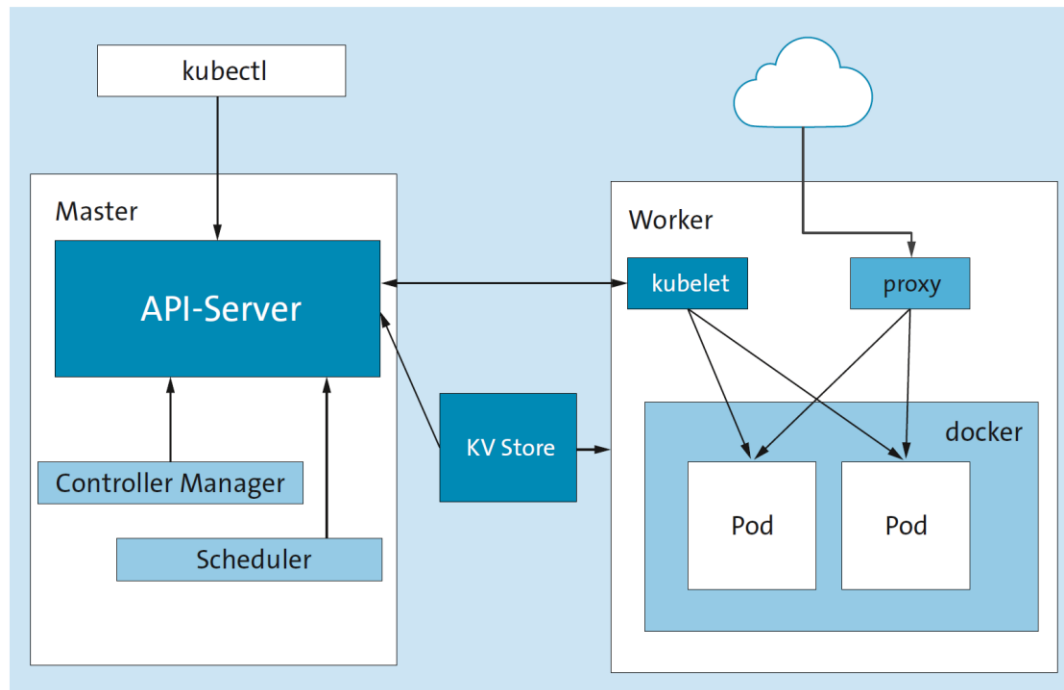


Abbildung 13.1 K8s Master und Worker aus der Vogelperspektive

★ Master

- der zentrale Koordinator in einem K8s Cluster, über den der Cluster gesteuert wird.

★ Worker

- Sie sind die Arbeits-Nodes des K8s. Auf ihnen laufen die Container bzw. im K8s-Kontext: Pods

★ Konfigurationsdateien und Keys:

- `/etc/kubernetes/`

K8s-Dienste auf den Master Nodes

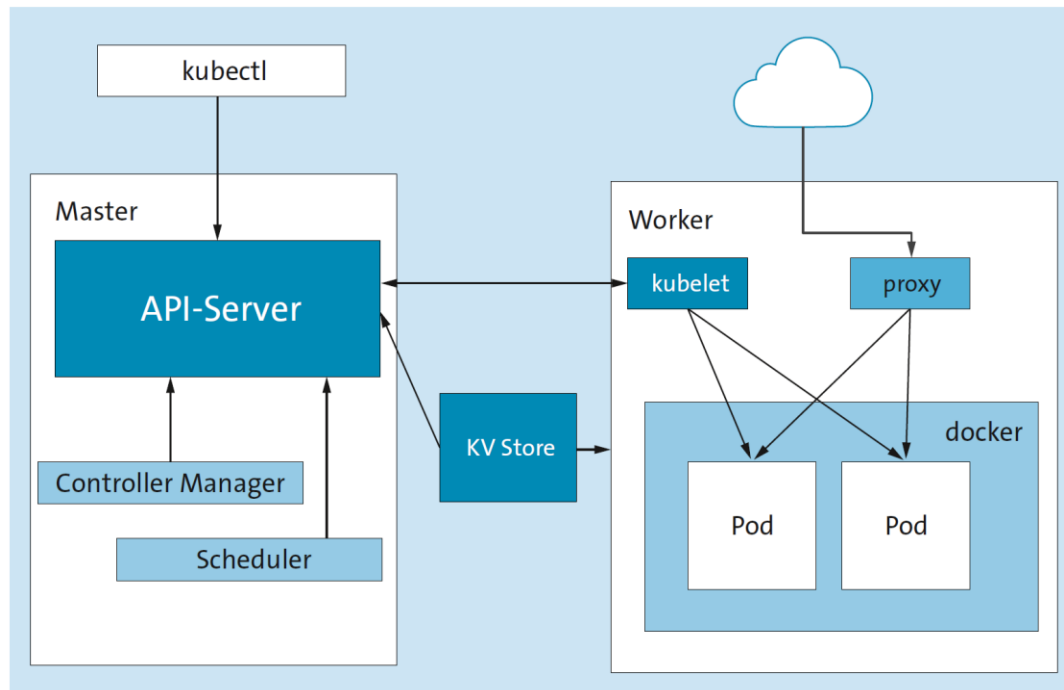


Abbildung 13.1 K8s Master und Worker aus der Vogelperspektive

★ API-Server

- zentrale administrative Service-Komponente.
- Managen aller Ressourcen des K8s Clusters.

★ Controller Manager (Replication Controller)

- Er ist u.a. verantwortlich für die gewünschte Anzahl von Replication Controller- / ReplicaSets-Objekten im gesamten Cluster.

★ Scheduler

- Der Scheduler-Prozess weist den Worker Nodes ihre Workloads, also die Pods/Container-Instanzen zu. Der Scheduler Prozess (**kube-scheduler**) kennt die Auslastung aller Worker Nodes und kann die Deployments so immer bestmöglich platzieren.

★ Etcd

- Key/Value (KV)-Store

K8s-Dienste auf den Workern

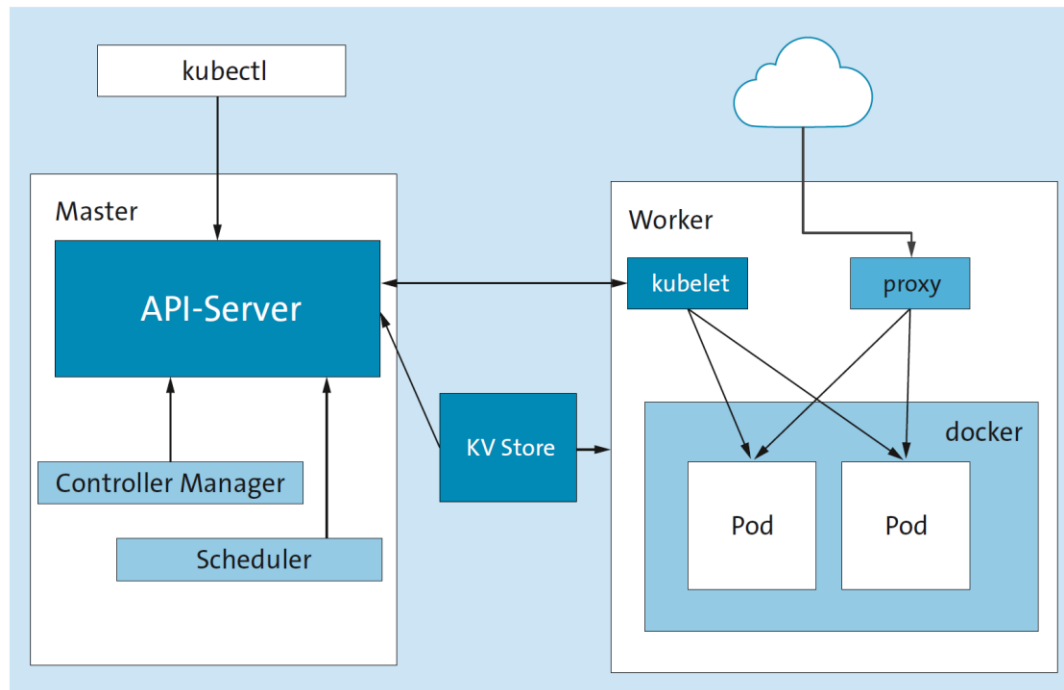


Abbildung 13.1 K8s Master und Worker aus der Vogelperspektive

★ Docker

- Die eigentliche Container-Plattform, mit der K8s arbeitet.

★ Kubelet (Cloud siehe

<https://kubernetes.io/docs/concepts/architecture/cloud-controller/>)

- Der kubelet-Service verbindet die Worker Nodes mit dem Kubernetes Master Node.

★ Kube-Proxy

- Der K8s-Netzwerk-Proxy – er arbeitet auf jedem Worker Node und kümmert sich um die eingehenden Requests und ihr Routing.
- Er kann als primitiver Loadbalancer fungieren

Networking in Kubernetes (1)

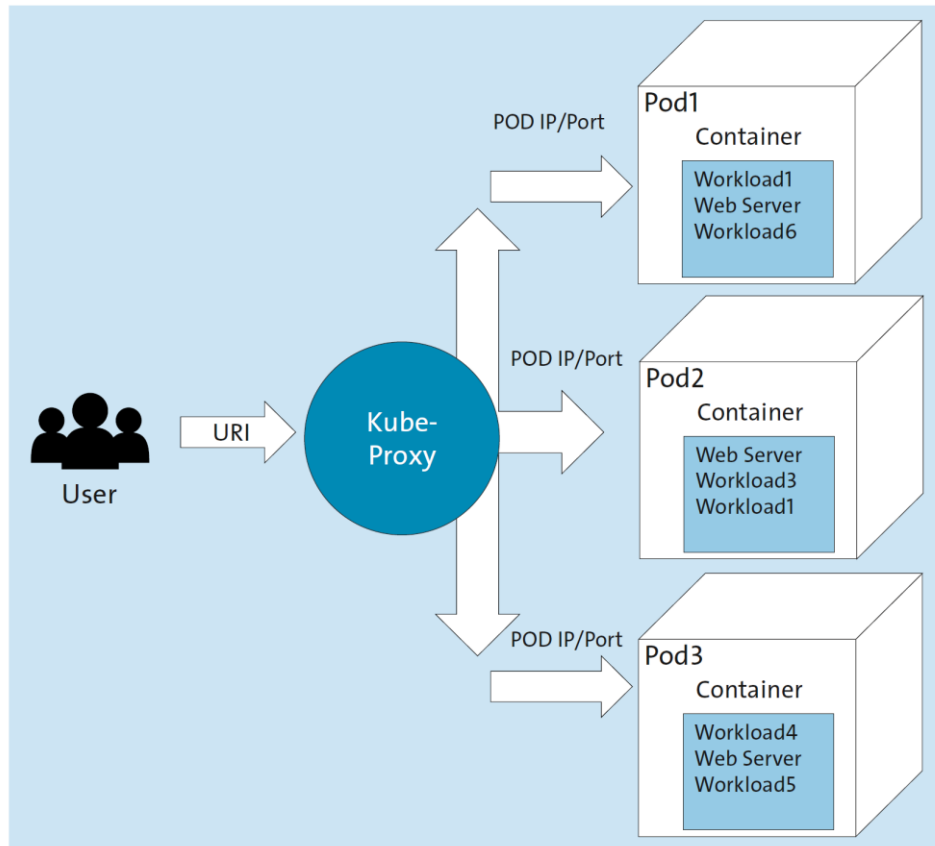


Abbildung 13.2 Arbeitsweise des Kube-Proxy exemplarisch auf einem K8s Worker Node

- ★ **Kubernetes** verwendet im Unterschied zu Docker eine **flache Netzwerkstruktur**:
 - Jeder Container kann mit jedem anderen ohne NAT kommunizieren.
 - Alle Kubernetes Nodes können mit allen Containern (und in die andere Richtung) ohne NAT kommunizieren.
 - Die IP, die ein Container von sich selbst sieht, ist auch die, die jeder andere Node oder Container im Netz von ihm sieht.
- ★ **Die Container können netzwerktechnisch eher wie VMs betrachtet und behandelt werden.**

Networking in Kubernetes (2)

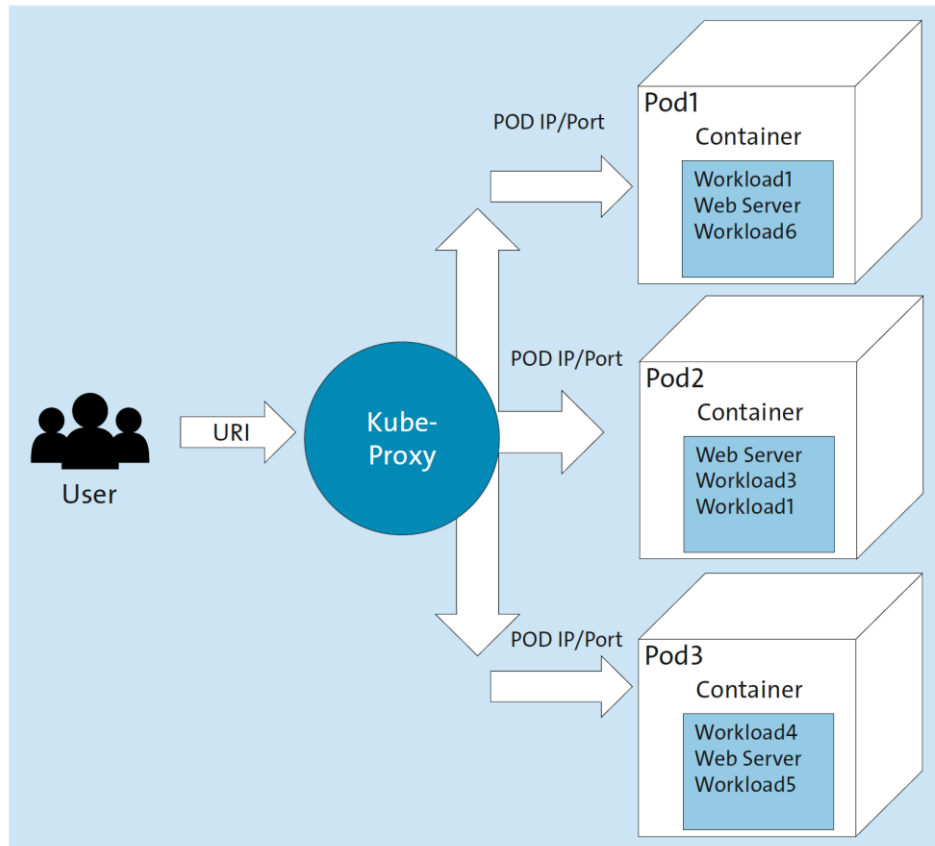


Abbildung 13.2 Arbeitsweise des Kube-Proxy exemplarisch auf einem K8s Worker Node

- ★ Grundsätzlich kann man die Kommunikation in K8s-Netzen in fünf (Buch vier) verschiedene Bereiche unterteilen:
 - **Container-zu-Container-Kommunikation:** Container innerhalb eines Pods teilen sich die gleiche IP.
 - **Pod-zu-Pod-Kommunikation:** Jeder Pod eines K8s Clusters erhält eine IP.
 - **Pod-zu-Service-Kommunikation:** ein Service stellt ein DNS-Name/Port zur Verfügung.
 - **Pod-zu-Ingress-Kommunikation:** ein Ingress stellt ein URL zur Verfügung.
 - **Externe-zu-interner-Kommunikation:** via externe Loadbalancer, Services, Ingress.

Networking in Kubernetes (3)

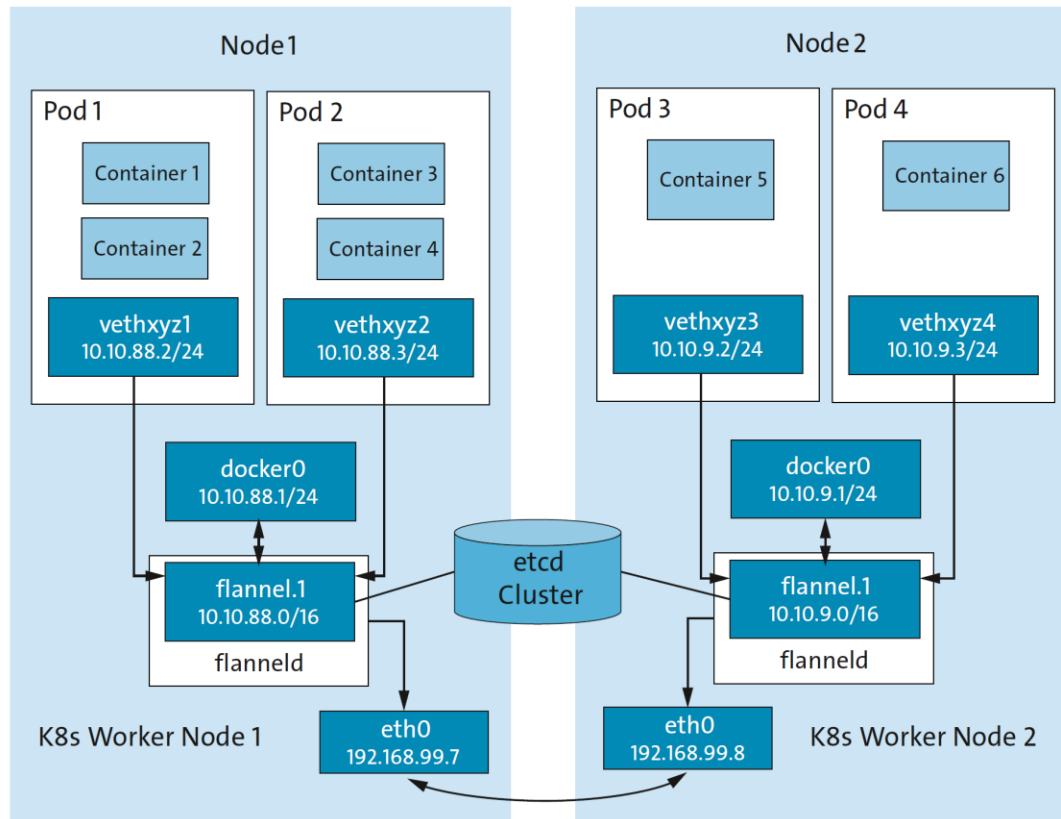
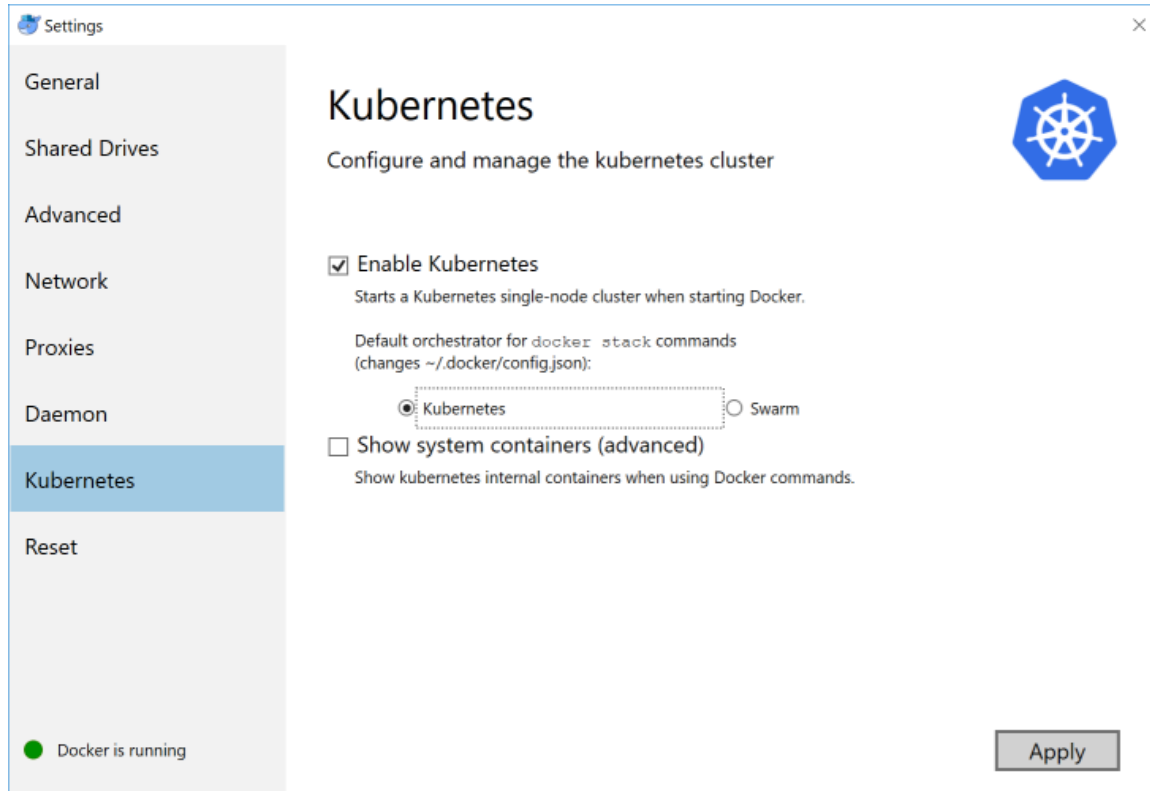


Abbildung 13.4 Schematisches Zusammenspiel der Netzwerkkomponenten

- ★ Was K8s zunächst fehlt, ist ein **Overlay-Netzwerk** welches die **Pods Node-übergreifend für die Interkommunikation** nutzen können.
- ★ Das Overlay-Netzwerk wird zwischen jedem K8s Node eingerichtet, sodass sich die Container/Pods auf verschiedenen Nodes direkt sehen bzw. miteinander kommunizieren können.
- ★ **Flannel** ist solch ein Overlay-Netzwerk, stammt aus der CoreOS-Schmiede, und fungiert als speziell für K8s designtes Netzwerk-Overlay.
- ★ Neben Flannel existieren weitere Plugins welche das [CNI - the Container Network Interface](#) implementieren.
- ★ Siehe auch: [Kubernetes Network Policy](#), [Installing a pod network add-on](#)

K8s Master-Setup

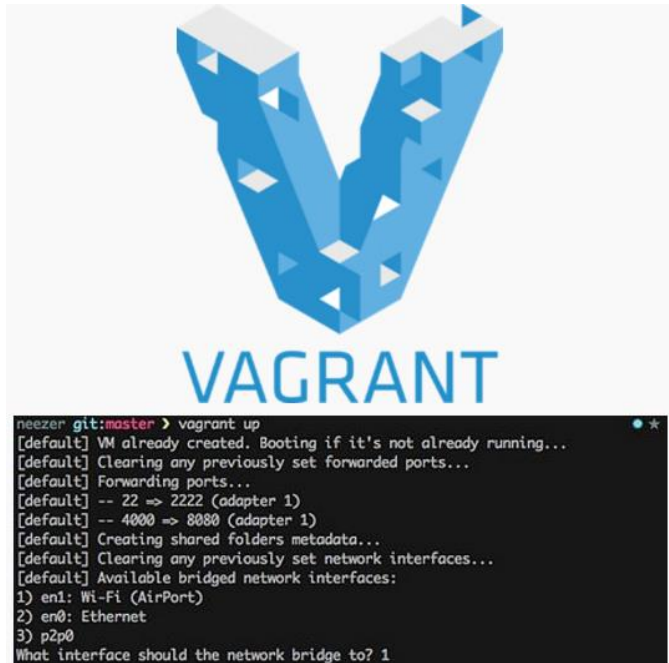


Setup Assistent von «Docker for Windows Edge»

<https://github.com/mc-b/lernkube/tree/master/docker4windows>

- ★ Minikube: zum schnell K8s Luft Schnuppern. Nur Single Node.
- ★ «**Docker for Windows/Mac**»: für Entwickler. Nur Single Node.
- ★ kubeadm: Single Node bis HA Cluster
- ★ Vagrant und kubeadm: bevorzugte Methode des Kursleiters.
- ★ Weitere Hochverfügbare (HA) Cluster siehe Buch Skalierbare Container-Infrastrukturen, Kapitel 13.9 – 13.12
- ★ **Hinweis: ab Version 1.5/1.6 basieren die K8s Dienste auf Pod's.**

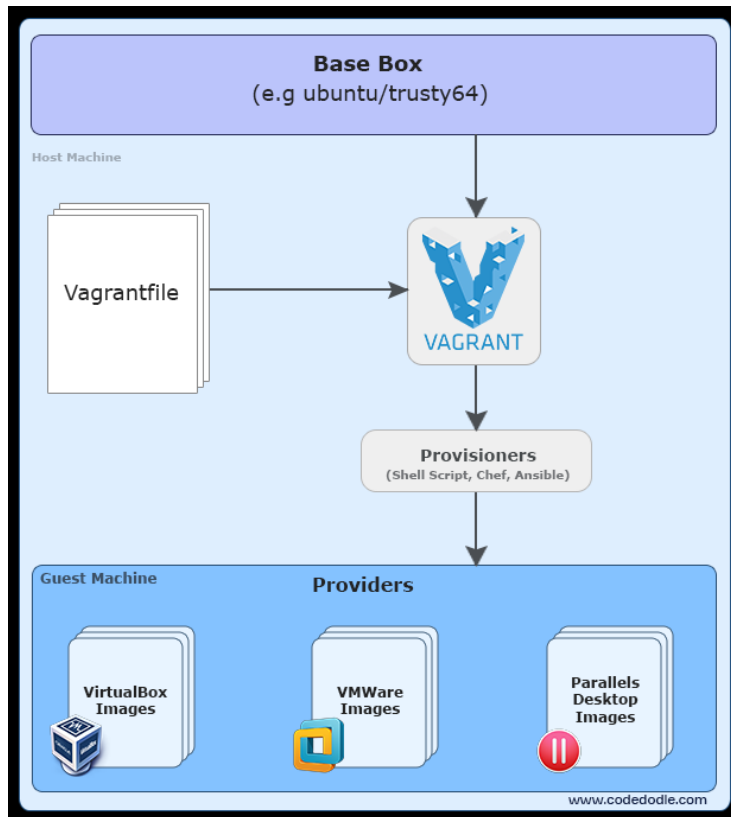
Vagrant



```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest:80,
  config.vm.synced_folder ".", "/var/www/html"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "512"
  end
  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get -y install apache2
  SHELL
end
```

- Vagrant automatisiert die Erstellung von Virtuellen Maschinen anhand von Definitionsdateien.
- Links: Start VM. Rechts: Vagrant Definitionsdatei

Vagrant: Funktionsweise und Konzepte

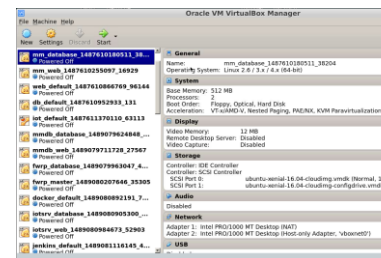


<https://app.vagrantup.com/boxes/search>

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest:80,
  config.vm.synced_folder ".", "/var/www/html"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "512"
  end
  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get -y install apache2
  SHELL
end
```

vagrant box add ...
vagrant box list

vagrant init ...
vagrant up



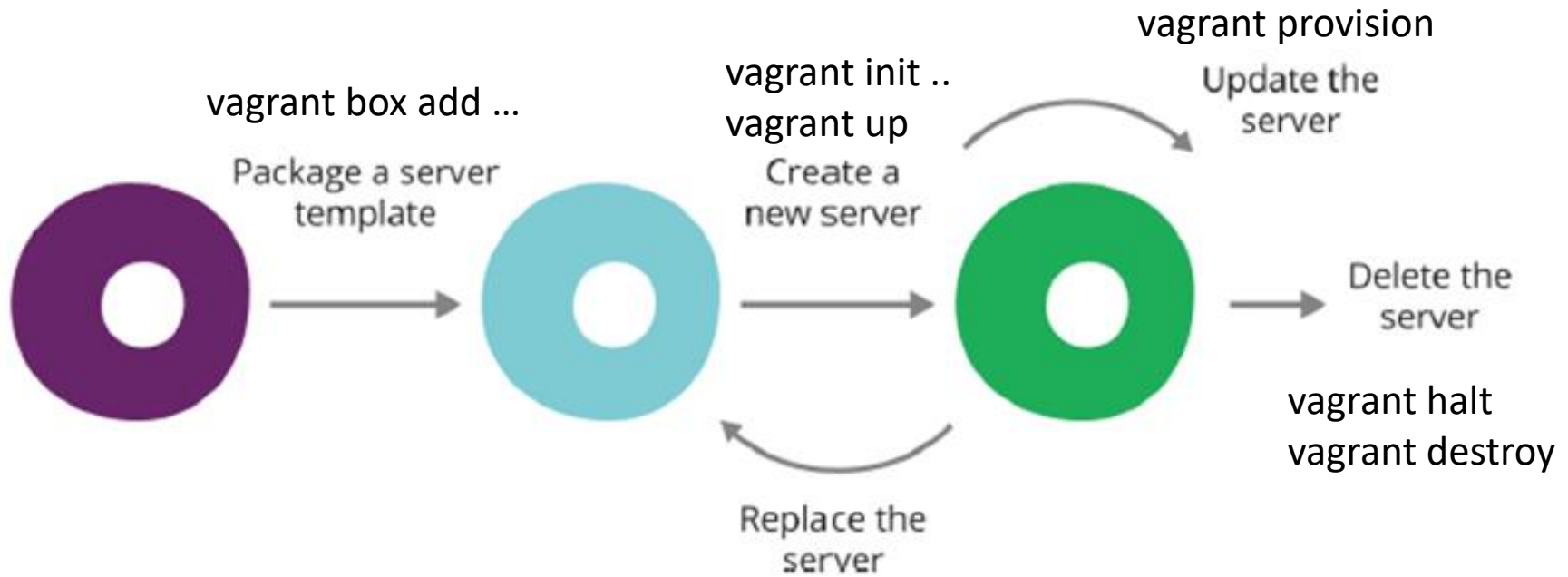
vagrant ssh
\$
\$ exit
vagrant halt

Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "forwarded_port", guest:80,
  config.vm.synced_folder ".", "/var/www/html"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "512"
  end
  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get -y install apache2
  SHELL
end
```

- config.vm.box
 - Welche Box als Grundlage für VM verwendet werden soll
- config.vm.network
 - Weiterleitung Port an Host
- config.vm.synced_folder
 - Zugriff auf Ordner Host
- config.vm.provider
 - Informationen für Hypervisor
- config.vm.provision
 - Installationsbefehle, z.B. als Shellscripts
-

Vagrant: Workflow



K8s Master-Setup - Vagrant

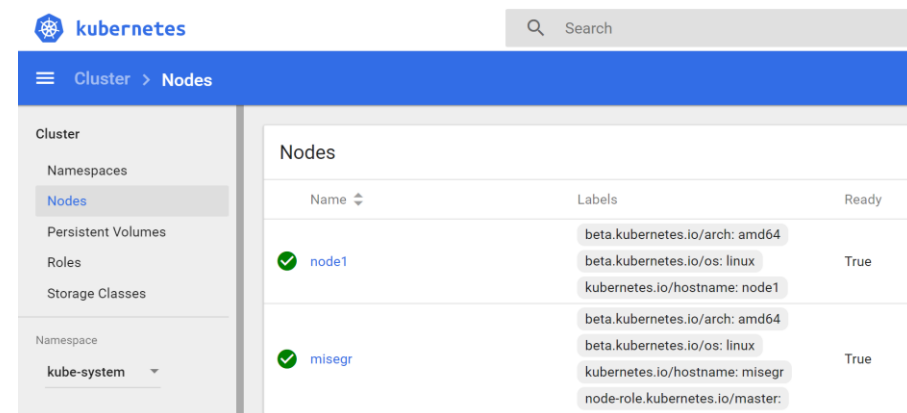
```
Vagrant.configure(2) do |config|  
  
  config.vm.box = "ubuntu/xenial64"  
  
  config.vm.provision "docker" do |d|  
  end  
  
  apt install -y kubelet kubeadm  
  
  kubeadm init  
  
  kubectl apply -f kube-flannel.yaml  
  
  kubectl apply -f xxx.yaml  
  
end
```

- ★ Die einzelnen Installationsschritte stehen im Vagrantfile welche vom CLI vagrant ausgewertet werden.
- ★ Die einzelnen Schritte für Master und Worker sind wie folgt:
 - Starten der Ubuntu Linux Box
 - Installation Docker und ggf. Anpassung Server Zertifikat für Remote Zugriff.
 - Installation K8s Komponenten
- ★ Nur auf dem Master
 - kubeadm Initialisieren
 - Internes Pods Netzwerk starten
 - Weitere Services wie ingress, Dashboard etc. Starten

K8s Worker Setup und Join

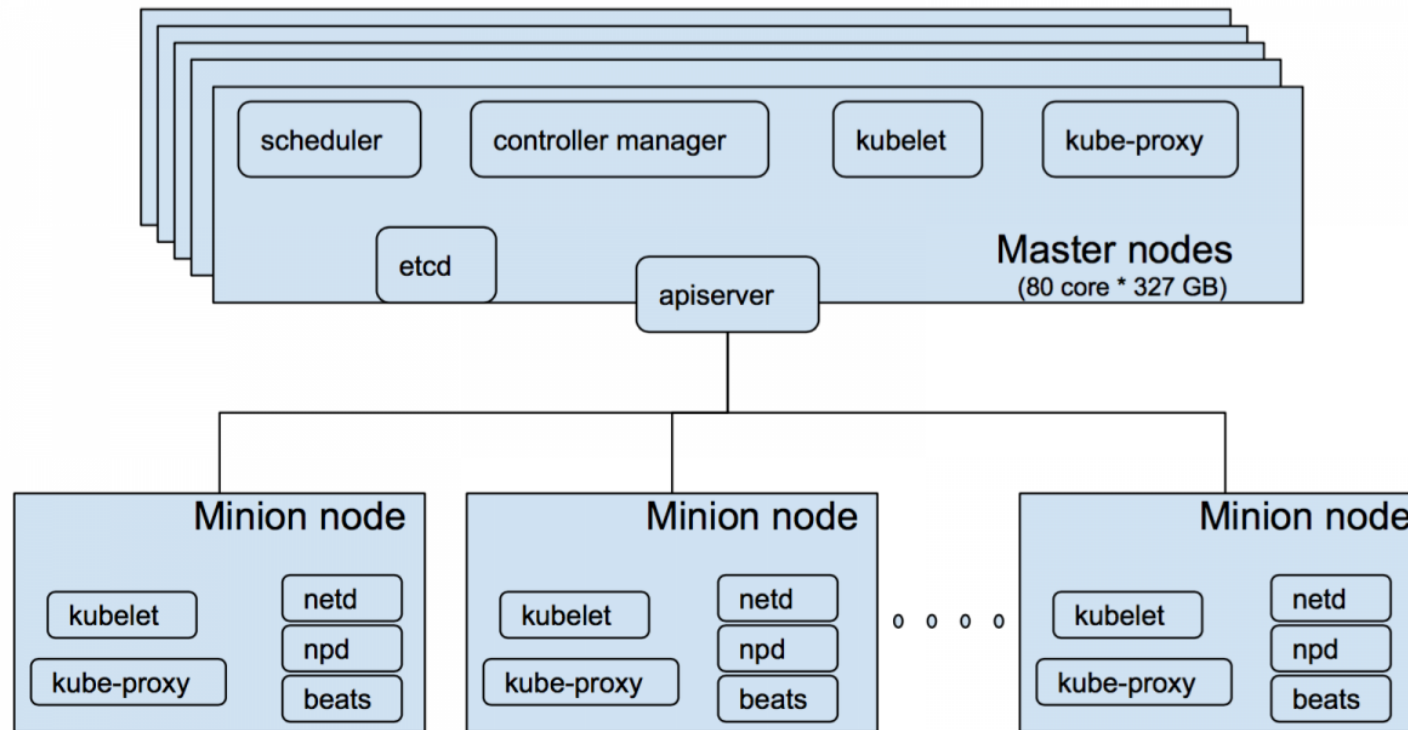
- ★ VM mit Docker aufsetzen und Pakete kubelet und kubeadm installieren (siehe vorherige Folie).
- ★ Join Token auf dem K8s Master ausgeben:
 - `sudo kubeadm token create --print-join-command`
- ★ In K8S Worker wechseln und Node joinen, mittels Ausgabe von oben:
 - ★ `kubeadm join --token <token> 192.168.137.100:6443 --discovery-token-ca-cert-hash <hash>`

- ★ Das Ergebnis ist im K8s [Dashboard](#) ersichtlich:



Name	Labels	Ready
✓ node1	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: node1	True
✓ misegr	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: misegr node-role.kubernetes.io/master:	True

eBay: 5000-Knoten-Skalierbarkeit für den tess.IO-Cluster



- ★ Kubernetes behauptet das 5.000 Nodes in einem Cluster möglich sind.
- ★ Der eBay Tess.IO-Cluster bestätigt, nach der Optimierung / Korrektur, diese Behauptung.
- ★ Quelle: <https://tech.ebayinc.com/engineering/scalability-tuning-on-tess-io-cluster/>

Federated/Geografisch verteilte K8s Cluster

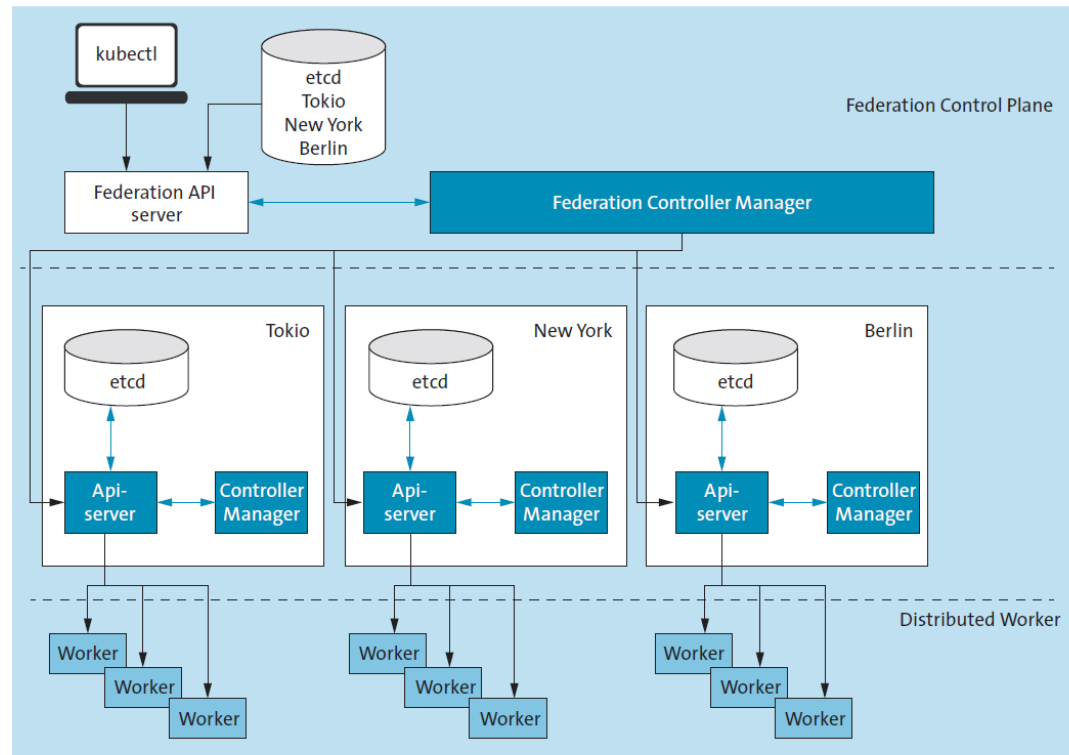


Abbildung 17.1 Federated K8s Cluster

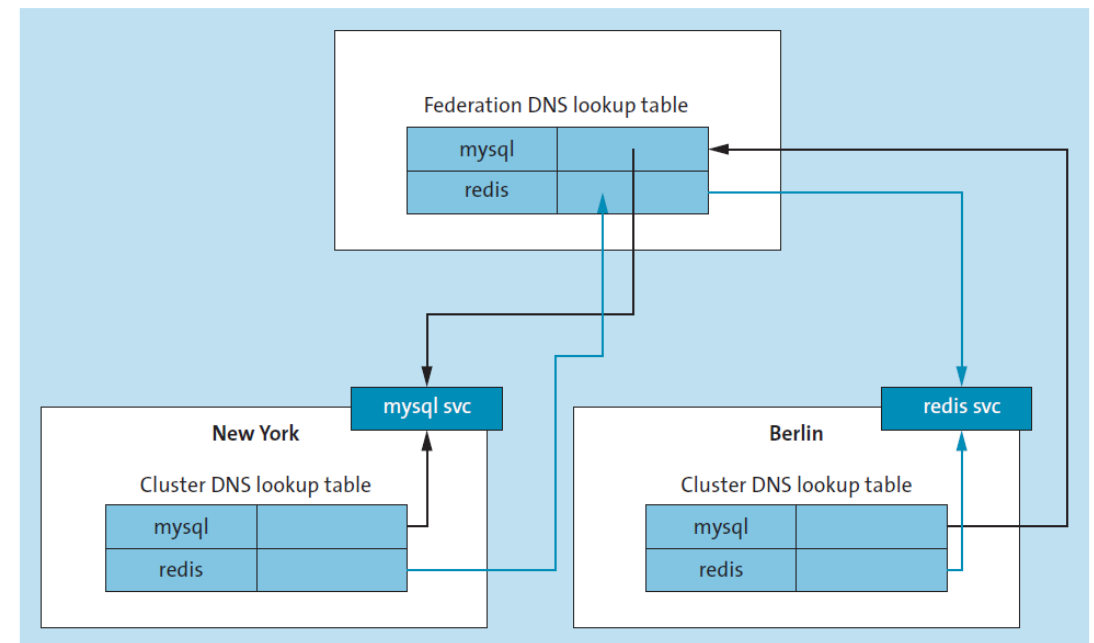
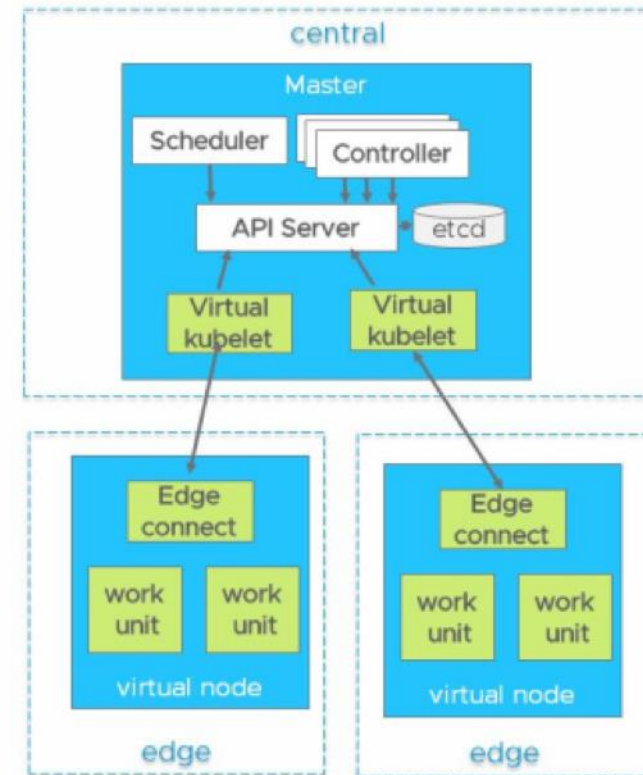
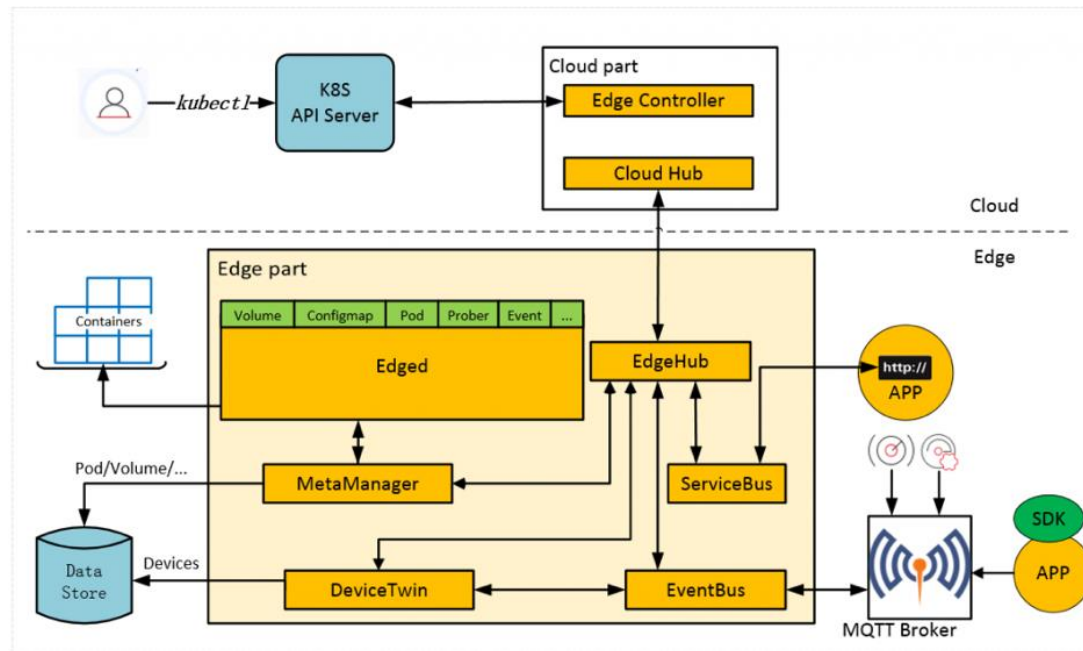


Abbildung 17.2 DNS mit Federated Setup

<https://kubernetes.io/docs/tasks/federation/federation-service-discovery/>

KubeEdge





Übung: Setup Master und Join K8s Worker Node

```
# Cluster Umgebung mit einem Master  
# und einer Worker Node
```

```
master:  
  count: 1  
  cpus: 2  
  memory: 8192
```

```
worker:  
  count: 1  
  cpus: 2  
  memory: 8192
```

```
k8s:  
  version: 1.15-1.00  
use_dhcp: false
```

- ★ Die `config.yaml` Datei dient zum Konfigurieren der Anzahl Master und Worker Nodes.
- ★ Änderungen in dieser Datei wirken sich auf die Anzahl zu Installierenden VM aus.
- ★ Startet die CLI Umgebung, im Verzeichnis `lernkube` mittels `kubeps.bat`
- ★ Löscht die aktuelle Docker/Kubernetes VM
 - `vagrant destroy -f`
- ★ Editiert die Datei `config.yaml` und erhöht die Anzahl Worker Nodes auf 1.
- ★ Startet die Erstellung von Worker und Master Nodes
 - `vagrant up`
- ★ Der Join Worker und Master findet automatisch statt.
- ★ Überprüft das Resultat im K8s [Dashboard](#) oder im Weave Scope.



Reflexion

- ★ Das im Juli 2014 gestartete (griechisch: Steuermann) stellt die derzeit **populärste** Container-Cluster-/Orchestrierungs-Lösung dar.
- ★ Von Single Node über Cluster, zu HA-Cluster zu Federated K8s werden praktisch alle Setup's unterstützt.
- ★ Es gilt heute, auch Dank der [CNCF](#), als **Industrie Standard**.

? Lernzielkontrolle

- ★ Sie haben einen ersten Einblick in Kubernetes