TMF – Technologie- und Methodenplattform
für die vernetzte medizinische Forschung e.V.

Dockerbank 2

Container-basiertes Deployment von biomedizinischen IT-Lösungen

Fortgeschrittene Werkzeuge

Matthias Löbe, Sebastian Stäubert

Institut für Medizinische Informatik, Statistik und Epidemiologie, Universität Leipzig

Praktische Übungen bzw. Teil 0
Installation der Docker-Plattform (Wiederholung)

Inhalt

- Installation unter Windows
- Installation unter Linux
- (Verwendung der Docker TMF VM)

Installation unter Windows[1]

a) Docker for Windows (Native Installation)

Voraussetzungen: Windows 10 Enterprise/Pro/Edu 64Bit, Hyper-V!

- Anleitung unter https://docs.docker.com/docker-for-windows/
- Docker steht anschließend nativ auf der Kommandozeile zur Verfügung

b) Docker Toolbox

Voraussetzungen: Windows 7 64Bit oder neuer, Virtualisierung aktiviert

- Anleitung unter https://docs.docker.com/toolbox/toolbox_install_windows/ bzw. 2016-03-02-Docker-Kurzanleitung.pdf[2]
  - Docker Client
  - Docker Toolbox management tool and ISO
  - **Oracle VM VirtualBox**
  - Git MSYS-git UNIX tools
- Docker läuft anschließend in einer VM
- Docker Toolbox terminal

*https://docs.docker.com/engine/installation/windows/

Installation unter Linux*

Voraussetzung: 64Bit, Kernel 3.10, Ubuntu ab v12

1. Paketinfos aktualisieren:
   ```
   $ sudo apt-get update
   $ sudo apt-get install apt-transport-https ca-certificates
   ```
2. GPG key hinzufügen:
   ```
   $ sudo apt-key adv
      --keyserver hkp://p80.pool.sks-keyservers.net:80
      --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
   ```
3. Paketquelle hinzufügen:
   ```
   $ echo „deb https://apt.dockerproject.org/repo ubuntu-xenial
      main“ > /etc/apt/sources.list.d/docker.list
   ```
4. Paketinfos aktualisieren und docker installieren:
   ```
   $ sudo apt-get update
   $ sudo apt-get install docker-engine
   ```
5. Docker Daemon starten:
   ```
   $sudo service docker start
   ```

* https://docs.docker.com/engine/installation/linux/ubuntulinux/
https://docs.docker.com/engine/installation/linux/debian/

Praktische Übungen bzw. Teil 1
Docker Swarm und Docker Machine

Inhalt

- Docker Netzwerke (CLI)
- Docker Netzwerke (Compose)
- Docker Machine
- Docker Swarm

Docker Netzwerke (CLI)

Szenario:

- Datenintegrationszentrum der Nationalen Initiative Medizininformatik
  - Eine hypothetische App "MedicalKnowledgeArchive" (**mkArchive**).
- Ziel: Aufbau eines Docker-Netzwerks **mkarchive-network**
  - Besteht aus zwei Containern: einem modifizierten Tomcat-Image und nutzt eine externe Datenbank

```
# Legt ein virtuelles Docker-Netz an
$ docker network create mkarchive-network

# Lädt ggfs. die Images vom Docker-Hub und startet die Container (im Hintergrund -d)
$ docker run --net=mkarchive-network --name=mkarchive-db -d tmf-ev/mkarchive-db:v1

$ docker run --net=mkarchive-network --name=mkarchive-web -d -p 80:8080 tmf-ev/mkarchive-web:v1
```

Docker Netzwerke (CLI)

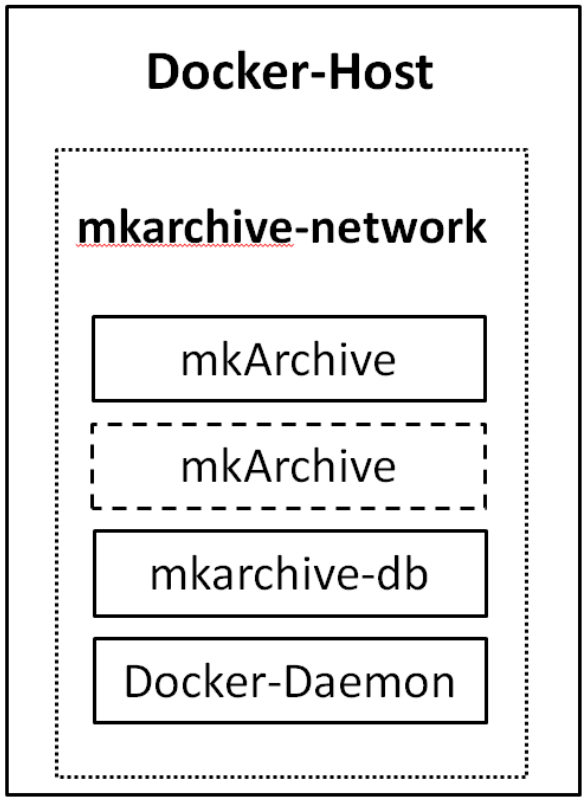
- Jeder Container bekommt eine eigene private IP-Adresse (172.20.0.2, 172.20.0.3)

```
# Gibt die IP aus, für welche Port 80 "freigeschaltet" wurde
$ docker inspect mkarchive-web | grep IPAddress

# Netzwerkkonfiguration (Ausschnitt)
# Docker-Netz im Bridge-Modus (nur ein Host)
$ ifconfig
br-b06060d3d2fb Link encap:Ethernet  HWaddr 02:42:27:bb:39:1a
          inet addr:172.20.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:27ff:febb:391a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12284 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12288 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:708341 (708.3 KB)  TX bytes:1767110 (1.7 MB)
```

Docker Netzwerke (Compose)

Compose-File (YAML):

```yaml
version: "1"
services:
  mkArchive:
    image: tmfev/mkarchive-web:v1
    ports:
     - "8080"
    depends_on:
     - tmfev/mkarchive-db
  mkarchive-db:
    image: tmfev/mkarchive-db:v1

networks:
  default:
    external:
      name: mkarchive-network
```



Docker Netzwerke (Compose)

```
# Anwendung starten
$ docker-compose up -d

# Gibt die IP-Adresse und Port aus, auf welchen die Anwendung erreichbar ist
# Host-Port wird jetzt dynamisch vergeben (wegen möglicher Skalierung!)
$ docker-compose port mkArchive 8080
```

- Netzwerktechnisch identisch zu Docker-Netzwerk (logisch, wird ja referenziert)
  - Aber auch Verteilung über mehrere Hosts möglich (Overlay-Netzwerk mit Key-Value-Store)
  - Neustart von Containern im Fehlerfall einstellbar

```
# Aber nun ist Lastverteilung möglich (2 x Web, 1 x DB) wegen hoher Zugriffszahlen!
# Neuer Container mit neuer IP, aber mit (einziger) DB verbunden
$ docker-compose scale mkArchive=2

# Gemeinsames Logfile aller Container (fortlaufend aktualisiert)
$ docker-compose logs -f
```

Docker Netzwerke (Machine)

- Installiert Docker auf entfernten Hosts (über SSH)
  - Compose arbeitet nur auf einem Host
- Docker-Machine unterstützt Virtualisierungslösungen wie VMWare oder Hyper-V, aber auch Cloud-Anbieter
  - Docker-Machine muss wie Docker-Compose als Binary installiert werden
  - Kann aber nicht ohne Weiteres innerhalb einer VM benutzt werden, da dort Virtualisierungsflags fehlen (VT-X/AMD-v)

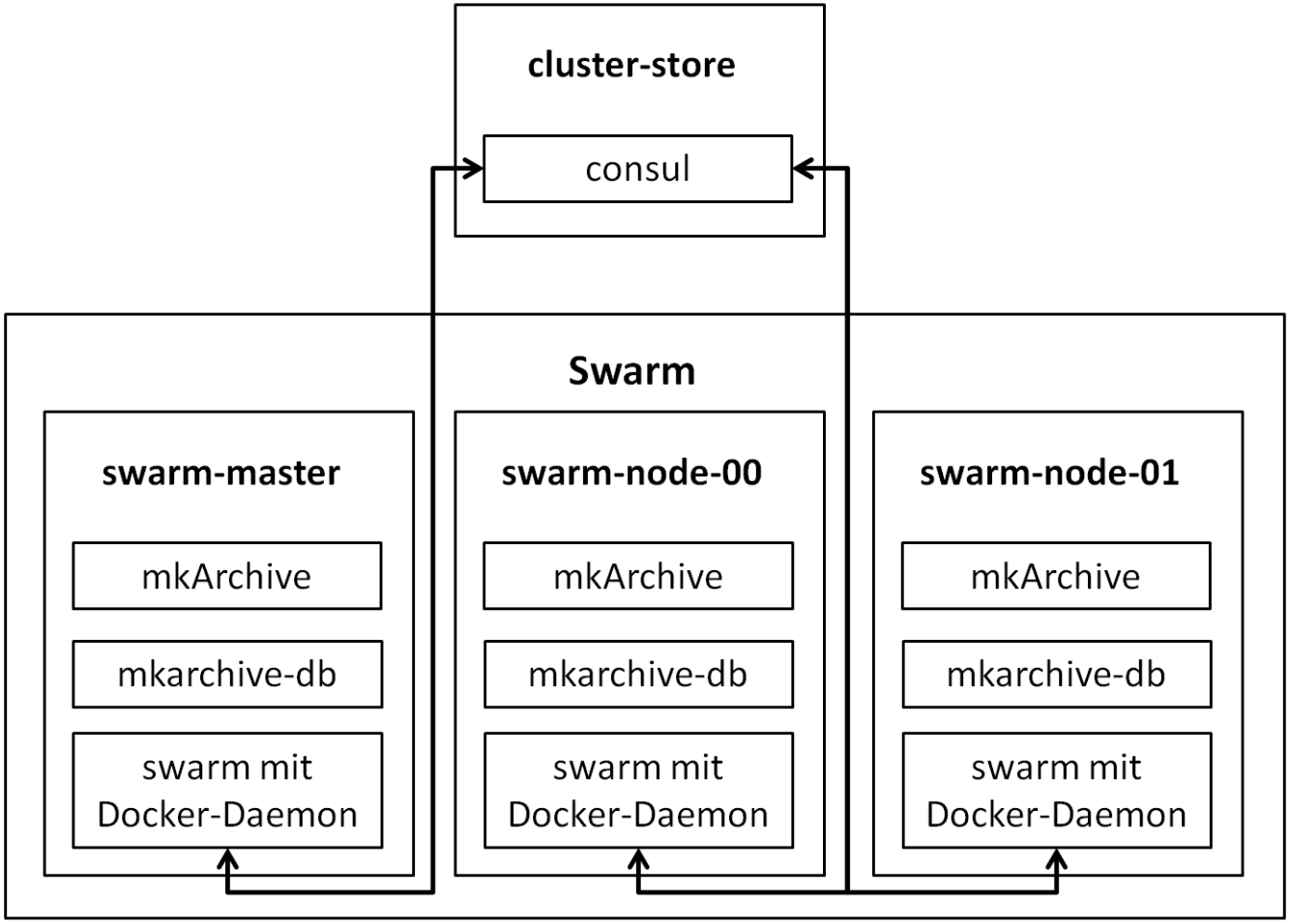
```
# VirtualBox installieren
$ apt-get install virtualbox

# Zentralen Cluster-Store erzeugen
$ docker-machine create -d virtualbox cluster-store

# Consul ist ein Key-Value-Store zur Verwaltung der Cluster-Metadaten (und ein Docker-Image)
$ docker ${DOCKER_OPTS} run -d -p 8500:8500 -h „consul“ consul ...
```

Docker Netzwerke (Swarm)

- Clustering von Docker-Hosts
  - Docker-Swarm ist auch nur ein Image mit dem Docker-Daemon
  - Jeder Clusterknoten (swarm-node) führt einen swarm-Container aus
- Genau ein Swarm-Master ist die zentrale Kommunikationsinstanz
  - Swarm verteilt die Container dann über die Knoten

```
# Konfigurationseinträge zum Finden des Cluster-Stores
$ DOCKER_OPTS=-d virtualbox --swarm --swarm-discovery=consul://$(docker-
machine ip cluster-store):8500 --engine-opt=cluster-advertise=eth1:0 ...

# Drei weitere Hosts (Nodes) starten
$ docker-machine create ${DOCKER_OPTS} --swarm-master swarm-master
$ docker-machine create ${DOCKER_OPTS} swarm-node-00
$ docker-machine create ${DOCKER_OPTS} swarm-node-01

# Umgebungsvariablen setzen, sodass alle Befehle auf dem Swarm-Master ausgeführt werden
$ eval $(docker-machine env --swarm swarm-master)

# Start der App, Swarm managed die Aktivierung der Container nach vorgegebener Strategie
$ docker-compose up -d
```

Architekturbilder Docker-Swarm



Praktische Übungen 2
Installation Kubernetes (minikube)

Inhalt

- Einführung kubernetes, Architektur, Begriffe
- Installation unter Windows
- Installation unter Linux
- (Verwendung der Docker TMF VM → geht nicht → Erklären)

Kubernetes: Einführung und Begriffe

Kubernetes („Ku-ber-ne-ties“ oder auch k8s):

- Ist ein Cluster-Manager für Docker-Container (Swarm++)
- Verteilt, skaliert und überwacht Container host-übergreifend
- Wird seit 2015 von Google entwickelt (Open Source)
- Kubernetes bringt eigene Konzepte und Konstrukte mit

Pods:

- Enthalten einen oder mehrere Container (z.B. für Dateiaustausch über Volumes)
- Verteilen, skalieren und überwachen Container host-übergreifend
- Wird immer auf einem Host ausgeführt (dedizierte IP-Adresse, alle Ports mögl.)

Labels:

- Schlagwörter, die an einen Pod gehängt werden („v1“, „dev“, „tomcat“, „backend“)
- Mit Filtern auf Labels lassen sich später Aktionen auf Gruppen von Pods durchführen

Kubernetes: Einführung und Begriffe

Controller:

- Laufen auf jedem Cluster-Knoten
- Übernehmen Steuerungsaufgaben im Cluster
- Replication Controller: Startet die gewünschte Zahl von Pods je Knoten

Services:

- Laufen auf jedem Cluster-Knoten
- Steuern Gruppen von Pods über vergebene Labels (auf allen Knoten)
- (Externer) Zugriff auf Dienste (Pods/Container) erfolgt über Services
- *Endpoints*: Kombination aus IP-Adresse und Port
- Werden von Kubernetes in einem internen DNS-Server verwaltet

Node (früher Minion):

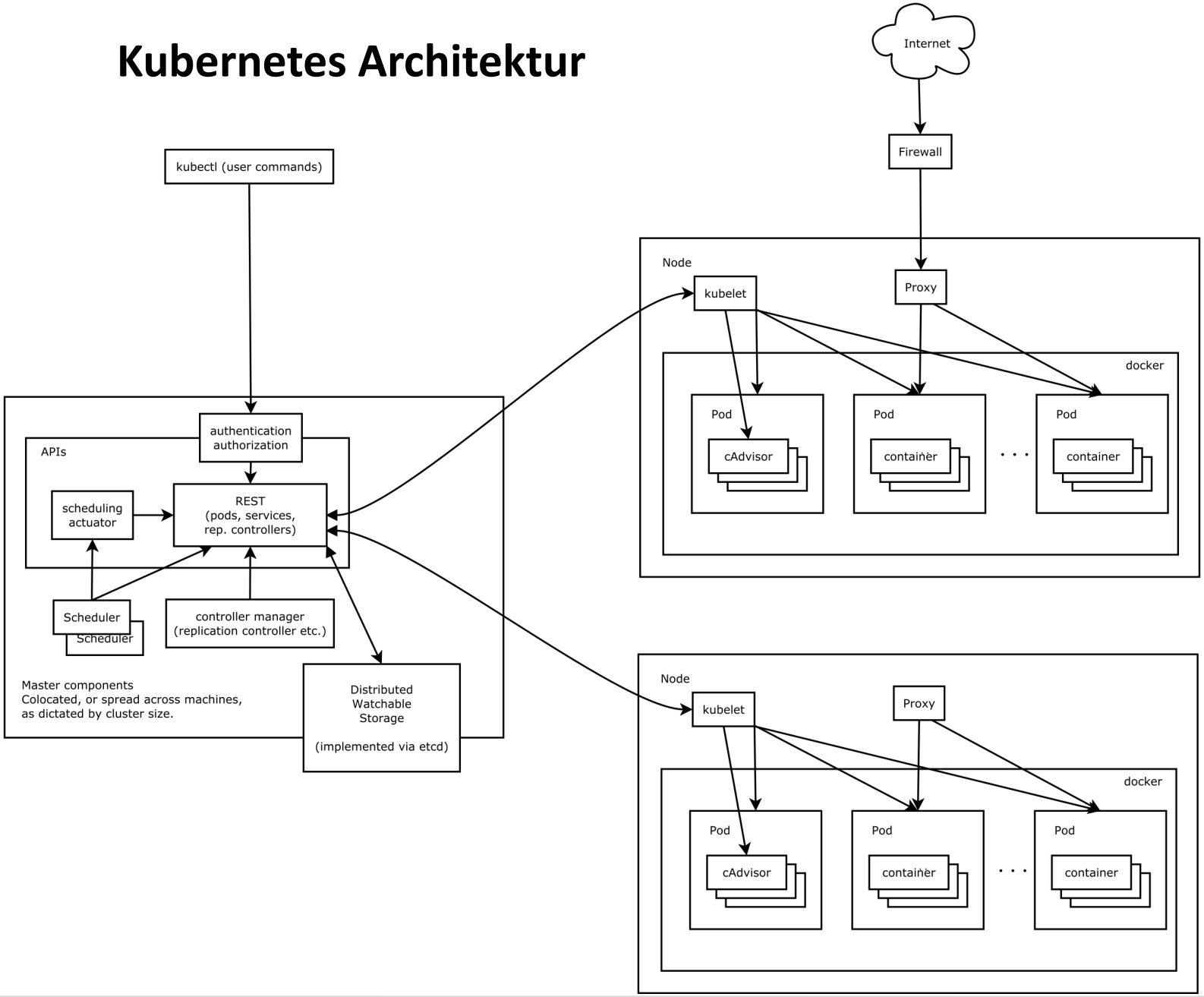
- Cluster-Knoten
- *Kubelet*: Kubernetes-Client auf jedem Knoten
- *cAdvisor*: Statistiken zur Ressourcennutzung

Kubernetes Einführung, Architektur, Begriffe

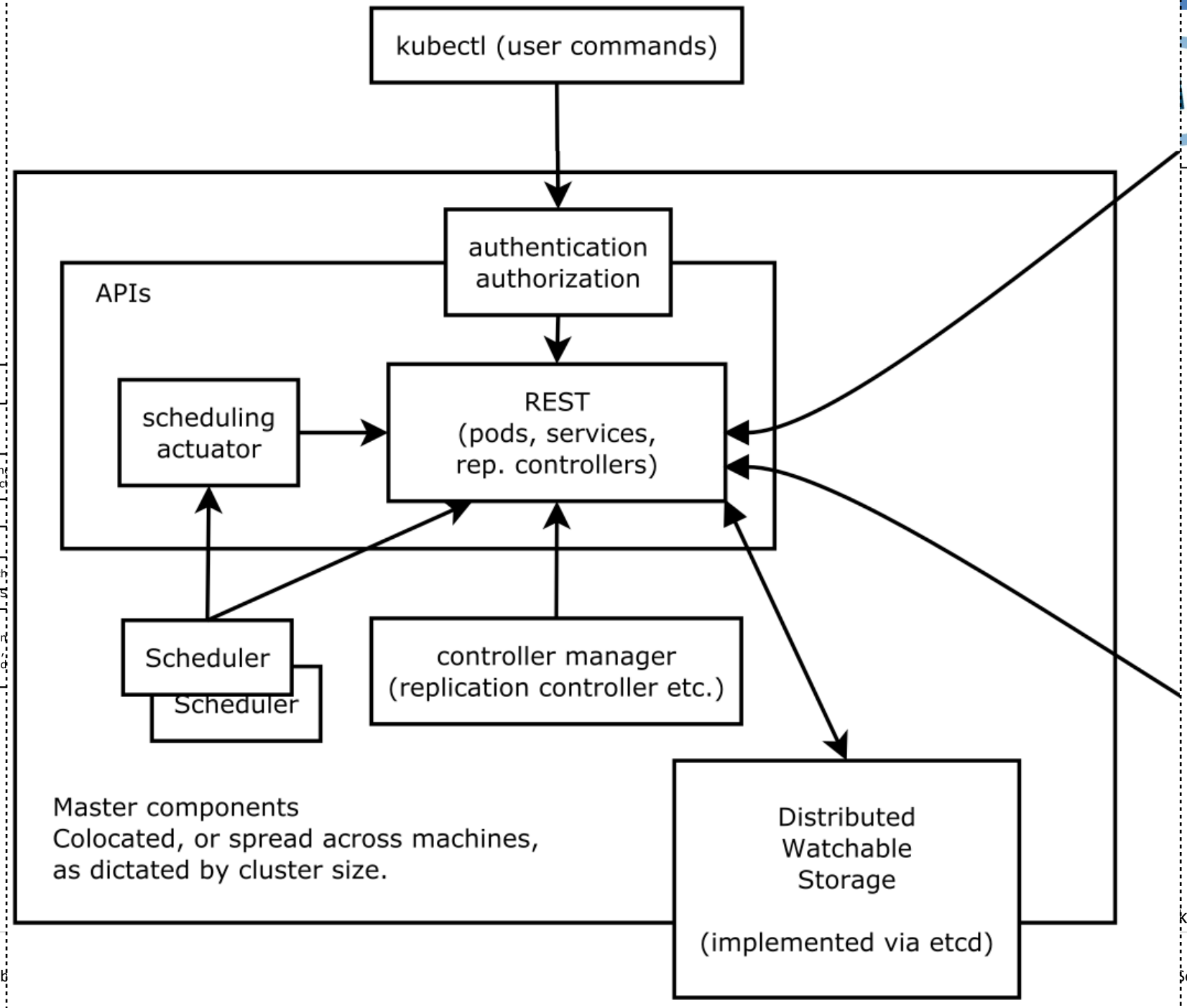
Weitere:

- Scheduler:
- Controller Manager
- REST-API
- Namespace
- Persistant Volume
- Workload
- Deployment
- Replica Sets, (Daemon Set)
- (Pet Sets), (Jobs)
- (Ingress)
- (Persistant Volume Claims)
- Config
- (Secrets), (Config Maps)

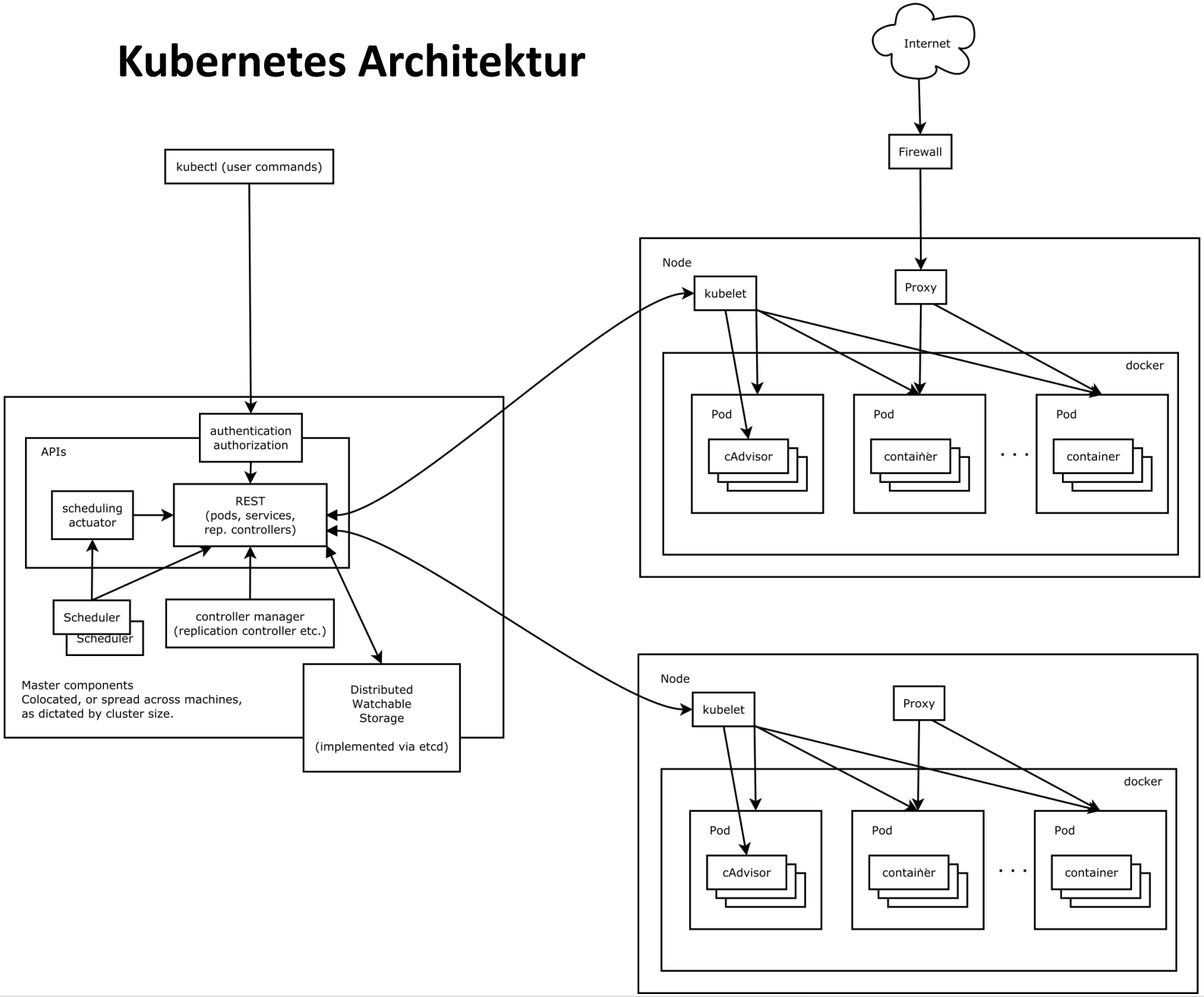
Kubernetes Architektur



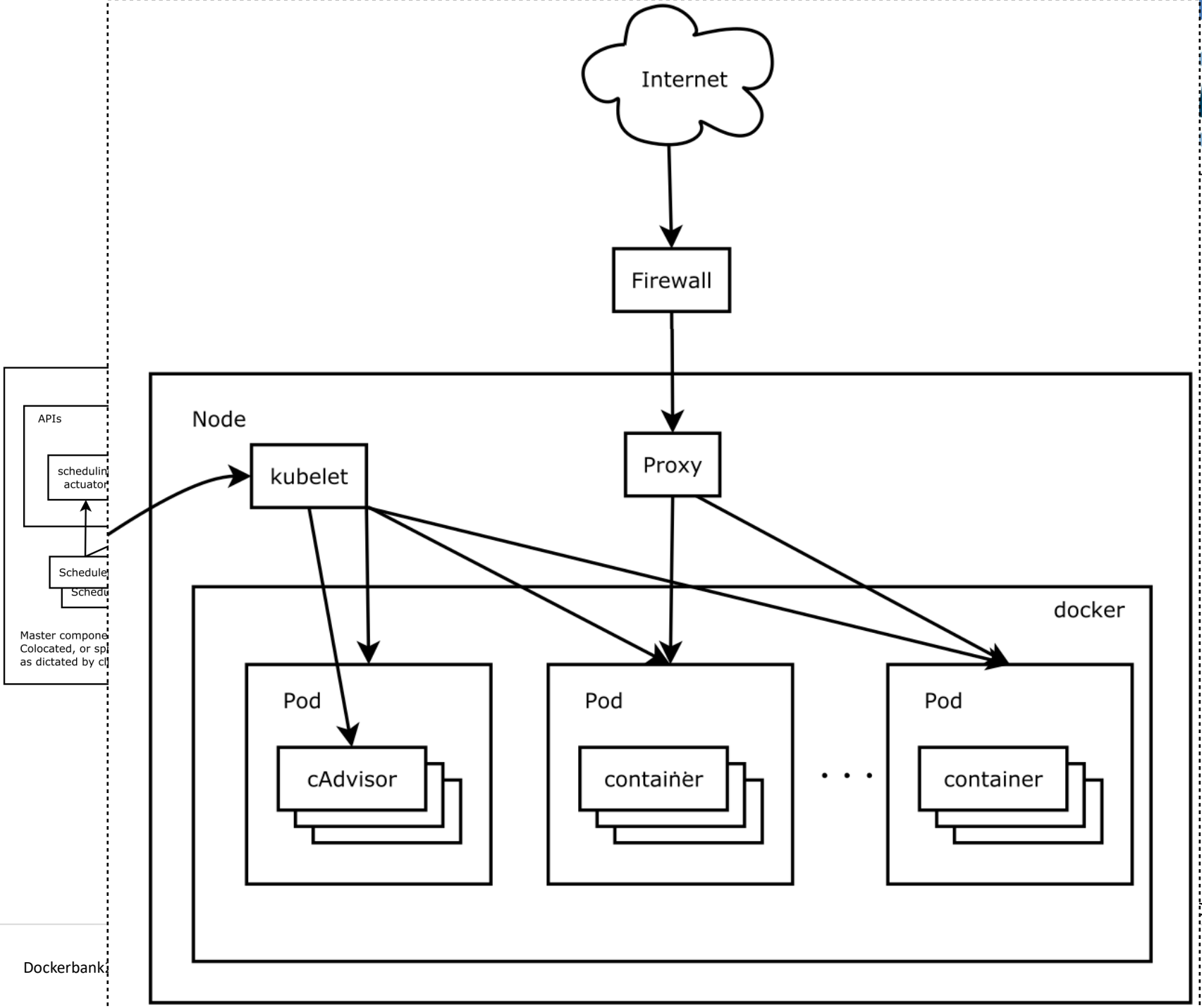
Quelle: Docker Projekt


kubectl (user commands)
authentication
authorization
APIs
scheduling
actuator
REST
(pods, services,
rep. controllers)
Scheduler
Scheduler
controller manager
(replication controller etc.)
Master components
Colocated, or spread across machines,
as dictated by cluster size.
Distributed
Watchable
Storage
(implemented via etcd)

Kubernetes Architektur



Quelle: Docker Projekt


Internet
Firewall
APIs
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
Pod
container

Installation Kubernetes unter Windows

1. Kubernetes Commandline Client (kubectl.exe)
   - Anleitung unter [1] bzw. [2]
   - Download kubectl-windows-amd64.exe [3]
   - `kubectl-windows-amd64.exe` → `kubectl.exe` und nach c:\ verschieben
2. Kubernetes Cluster (minikube)
   - Anleitung unter [4]
   - Download minikube-windows-amd64.exe [5]
   - `minikube-windows-amd64.exe` → `minikube.exe` und nach c:\ verschieben

Hinweis: `kubectl.exe` und `minikube.exe` müssen über die Umgebungsvariable PATH erreichbar sein, z.B. nach c:\ bewegen

[1] http://kubernetes.io/docs/user-guide/kubectl/
[2] http://kubernetes.io/docs/getting-started-guides/kubectl/
[3] https://storage.cloud.google.com/kubernetes-release/release/v1.4.6/bin/windows/amd64/kubectl.exe
[4] https://github.com/kubernetes/minikube/releases
[5] https://storage.googleapis.com/minikube/releases/v0.13.0/minikube-windows-amd64.exe

Installation Kubernetes unter Linux

(0.) Virtualbox installieren: `$ sudo apt-get install virtualbox`

1. Kubernetes Commandline Client (kubectl)
   - Anleitung unter [1] bzw. [2]
   - Download und Install ([3] kubernetes komplett, [4] kubectl (Linux, amd64))

```
$ wget [4] && chmod +x kubectl
$ sudo mv kubectl /usr/local/bin/
```

2. Kubernetes Cluster (minikube)
   - Download und Anleitung unter [5], [6]

```
$ curl -Lo minikube [6]  && chmod +x minikube
$ sudo mv minikube /usr/local/bin/
```

Hinweis: `kubectl` und `minikube` müssen über die Umgebungsvariable PATH erreichbar sein

[1] http://kubernetes.io/docs/user-guide/kubectl/
[2] http://kubernetes.io/docs/getting-started-guides/kubectl/
[3] https://github.com/kubernetes/kubernetes/releases
[4] https://storage.googleapis.com/kubernetes-release/release/v1.4.4/bin/linux/amd64/kubectl
[5] https://github.com/kubernetes/minikube/releases
[6] https://storage.googleapis.com/minikube/releases/v0.13.0/minikube-linux-amd64

Praktische Übungen 3
Einrichtung Kubernetes (minikube)

Inhalt

- Kubernetes Versionen abfragen
- Kubernetes einrichten und starten
- Status abfragen
- IP-Adresse abfragen
- Dashboard im Browser aufrufen

Kubernetes Versionen abfragen

Voraussetzung: Admin-Rechte

1. Powershell bzw. Terminal starten:
   Windows: `Windows Powershell „Als Adminsitrator ausführen“`
   Linux: `$ sudo -i`
2. Version abfragen:
   `$ minikube version`
3. Kommandos ansehen:
   `$ minikube --help`
4. Unterstützte Versionen von Kubernetes (Server) anzeigen:
   `$ minikube get-k8s-versions`
5. Verzeichnis %HOMEPATH%\.minikube untersuchen
   Windows: `$ gci env:user*`
   Linux: `$ ls %HOMEPATH%\.minikube`
6. Kopieren Sie <USB>\kubernetes\.minikube* nach %HOMEPATH%\.minikube\

Kubernetes Cluster einrichten und starten

Voraussetzung: Admin-Rechte, Virtualbox, TMF Docker VM nicht aktiv

1. Parameter für das „start“ Kommando anzeigen:
   ```
   $ minikube start --help
   ```
2. Kubernetes Cluster starten
   ```
   $ minikube start --kubernetes-version="v1.4.3"
                        --vm-driver="virtualbox"
                        --show-libmachine-logs
                        --alsologtostderr
   ```
3. Schauen Sie sich die Ausgabe an. Achten Sie auf folgende Meldungen:
   a) `Starting local Kubernetes cluster...`
   b) `Creating Machine...`
   c) `Docker is up and running!`
   d) `Kubectl is now configured to use the cluster.`

- Kubernetes (minikube) läuft anschließend in einer VM
- Zugriff erfolgt über kubectl

Status des Kubernetes Cluster prüfen

Voraussetzung: vorherige Schritte waren erfolgreich

1. Fragen Sie den Status des Cluster ab:
   ```
   $ minikube status
   ```
2. Fragen Sie Cluster Informationen über kubectl ab:
   ```
   $ kubectl cluster-info
   ```
3. Fragen Sie die Server und Client Version ab:
   ```
   $ kubectl version
   ```

(IP-)Adressen anzeigen und Dashboard aufrufen

Voraussetzung: vorherige Schritte waren erfolgreich

1. Fragen Sie die IP-Adresse des Cluster ab:
   ```
   $ minikube ip
   ```
2. Rufen Sie das Kubernetes Dashboard auf:
   ```
   $ minikube dashboard
   ```
3. Fragen Sie die URL des Dashboards ab:
   ```
   $ minikube dashboard --url=true
   ```
4. Schauen Sie sich die Nodes im Dashboard an:
   Kubernetes → Admin → Nodes
5. Vergleichen Sie diese Angaben mit der Ausgabe des folgenden Befehls:
   ```
   $ kubectl get nodes
   ```
6. Schauen Sie sich weitere Details über die Ausgabe des folgenden Befehls an:
   ```
   $ kubectl describe nodes
   ```

Praktische Übungen 4
Container anlegen, starten und verwalten

Inhalt

- Anwendungscontainer erzeugen (Commandline Interface)
  - Deployment erzeugen und Status prüfen
  - Replica set
  - Pods und Details
  - Logs
  - Services
- Anwendungscontainer erzeugen (Dashboard)
  - Deployment erzeugen und Status prüfen
  - Replica set
  - Pods und Details
  - Logs
  - Services

Anwendungscontainer erzeugen (CLI)

1. Erzeugen Sie ein Deployment auf Basis eines Docker Images:
   ```
   $ kubectl run nginx-hello-tmf
       --image="tmfev/nginx-hello-tmf"
       --port=80
   ```
2. Im Dashboard des Status des Deployments prüfen – Wie viele Pods laufen?:
   ```
   $ kubectl get deployments
   ```
3. Wenn das Deployments fertig ist, ist ein Replica Set verfügbar:
   ```
   $ kubectl get rs
   $ kubectl describe rs nginx-hello-tmf-<rs-hash>
   ```
4. Schauen Sie sich die Details zu den laufenden Pods an:
   ```
   $ kubectl get pods
   $ kubectl describe pod nginx-hello-tmf-<rs-hash>-<p-hash>
   ```
5. Schauen Sie sich die Logs des laufenden Pods an:
   ```
   $ kubectl logs [-f] nginx-hello-tmf-<rs-hash>-<p-hash>
   ```

Anwendungscontainer erzeugen (CLI)

7. Service anlegen:
   ```
   $ kubectl expose deployment nginx-hello-tmf --type=NodePort
   ```
8. Service anzeigen:
   ```
   $ kubectl get services
   $ kubectl describe service nginx-hello-tmf
   ```
9. Service URL anzeigen:
   ```
   $ minikube service --url=true nginx-hello-tmf
   ```
10. Rufen Sie die angezeigte URL im Browser auf.
11. Schauen Sie sich erneut die Logs (5.) an:
    ```
    $ kubectl logs [-f] nginx-hello-tmf-<rs-hash>-<p-hash>
    ```

Anwendungscontainer erzeugen (Dashboard)

1. Erzeugen Sie ein Deployment auf Basis eines Docker Images:
   Dashboard → Deployments → Create:
   `App name:` nginx-hello-tmf
   `Container image:` tmfev/nginx-hello-tmf
   `Service:` External; `Port:` 80; `Target port:` 80

2. Im Dashboard des Status des Deployments prüfen – Wie viele Pods laufen?:
   Dashboard → Deployments → nginx-hello-tmf
3. Wenn das Deployments fertig ist, ist ein Replica Set verfügbar:
   Dashboard → Replicat Sets → nginx-hello-tmf-<rs-hash>
4. Schauen Sie sich die Details zu den laufenden Pods an:
   Replicat Sets → nginx-hello-tmf-<rs-hash> → Pods
   → nginx-hello-tmf-<rs-hash>-<p-hash>
   Pods → nginx-hello-tmf-<rs-hash>-<p-hash>
5. Schauen Sie sich die Logs des laufenden Pods an:
   → View logs

Anwendungscontainer erzeugen (Dashboard)

6. Schauen Sie sich die Services an:
   Dashboard → Services
7. Schauen Sie sich die Details des Service nginx-hello-tmf an:
   Dashboard → Services → nginx-hello-tmf
8. Von welchem „Type“ ist dieser Service?
9. Setzen Sie den „Type“ von „LoadBalancer“ auf „NodePort“:
   Edit → Such-Feld: „Type“
   → „LoadBalancer“ durch „NodePort“ ersetzen
   → Update
10. Schauen Sie sich unter Services nun das Icon des Services nginx-hello-tmf an – was fällt Ihnen auf?
11. Gehen Sie zurück zu 7. und rufen Sie die Cluster-IP (siehe Dashboard URL) mit dem Port des zweiten „Internal endpoints“ im Browser auf.

Praktische Übungen 5
Skalierung und Rückbau

Inhalt

- Skalierung
- stop & delete

Skalierung, Stop, Delete (CLI)

1. Skalieren Sie das Deployment nginx-hello-tmf um 2 weitere Pods:
   ```
   $ kubectl scale --replicas=3 deployments/nginx-hello-tmf
   ```
2. Mit kubectl den Status der Erzeugung prüfen:
   ```
   $ kubectl get deployment
   $ kubectl get rs
   $ kubectl get pods
   ```
3. Logs der Pods ansehen:
   ```
   $ kubectl logs [-f] nginx-hello-tmf-<rs-hash>-[*]
   ```
4. Last erzeugen und Logs dabei prüfen:
   ```
   Docker-VM starten und folgende Kommandos aufrufen:
   $ apt-get install apache2-utils
   $ ab -n 1000 <service-URL>
   ```
5. Deployment und Service stoppen/löschen (stop=delete):
   ```
   $ kubectl delete deployment/nginx-hello-tmf
   $ kubectl delete service/nginx-hello-tmf
   ```
6. Cluster stoppen:
   ```
   $ minikube stop
   ```

Skalierung, Stop, Delete (Dashboard)

1. Führen Sie die Aufgaben der letzten Folie im Dashboard aus. :-)

Ausblick

1. Rolling Updates inkl. Parallelbetrieb
2. Betrieb von Stateful Applications (z.B. mysql)
3. Verwaltung von (Persistant) Volumes
4. Pets Sets...
5. Daemon Sets…
6. Multi Node
7. Grafische Auswertung (Überblick, Problemanalyse, uvm.)
8. Addons
9. Uvm.

Vielen Dank für Ihre Aufmerksamkeit!

matthias.loebe@imise.uni-leipzig.de
sebastian.staeubert@imise.uni-leipzig.de

Referenzen

1. http://kubernetes.io/ (Kubernetes Hompage, Doku, Howtos, Blog, usw.)
2. http://kubernetes.io/docs/
3. http://kubernetes.io/docs/user-guide/
4. http://kubernetes.io/docs/tutorials/
5. http://kubernetes.io/docs/tasks/
6. http://kubernetes.io/docs/reference/
7. http://kubernetes.io/docs/samples/
8. http://kubernetes.io/docs/tutorials/kubernetes-basics/cluster-intro/
9. http://kubernetes.io/docs/tutorials/stateless-application/run-stateless-application-deployment/
10. http://kubernetes.io/docs/tutorials/stateful-application/run-stateful-application/
11. http://kubernetes.io/docs/user-guide/docker-cli-to-kubectl
12. http://kubernetes.io/docs/getting-started-guides/kubectl/ (u.a. Befehlsvervollständigung)
13. http://kubernetes.io/docs/getting-started-guides/minikube/ ()
14. https://github.com/kubernetes (kubernetes Repositories)
15. https://github.com/kubernetes/kubernetes/releases (Download kubernetes)
16. https://github.com/kubernetes/minikube/releases (Download minikube)
17. https://github.com/kubernetes/minikube/blob/v0.13.0/README.md (Doku minikube-Release spezifisch)
18. https://github.com/docker/toolbox/releases (Download Docker Toolbox, inkl. Docker, VirtualBox usw.)
19. https://rominirani.com/tutorial-getting-started-with-kubernetes-on-your-windows-laptop-with-minikube-3269b54a226#.q52u2h57j
20. https://www.windowspro.de/script/umgebungsvariablen-setzen-anzeigen-loeschen-powershell (Powershell)
21. https://hub.docker.com/_/nginx/ (Docker-Hub nginx – Basis für das nginx-hello-tmf Image)