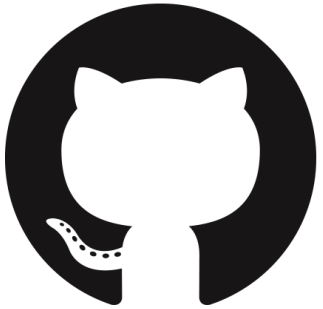
Kubernetes Workshop

Basic Concepts

Jan Harrie

Source: https://upload.wikimedia.org/wikipedia/commons/b/bd/The_new_container_ship_MOL_Tribute_on_the_Elbe_with_destination_port_of_Hamburg_-_2017.jpg

Before we get started



- Technical material available on Github:
  https://github.com/NodyHub/k8s-ws
  `git clone https://github.com/NodyHub/k8s-ws.git`

- Make sure that you have an Minikube up and running
  Else, use Vagrant and spin up a VM in Virtual Box with
  https://github.com/NodyHub/k8s-ws/tree/master/00_prep/minikube

- Make sure that you have `kubectl` installed and configured

#whoami - Jan

- Freelancer – Security Consultant
- Former Security Analyst/Consultant/Pentester
- M.Sc. IT-Security TU Darmstadt

- Interests:
  - Orchestration Solution
  - Cloud
  - Gardening

#whoareyou?

- Expectation
- Background?
  - Dev
  - Ops
  - Networker
- Experience Docker/Kubernetes/PaaS

Source: https://sayingimages.com/who-are-you-meme/

Re-Cap

Container

Source: https://libreshot.com/wp-content/uploads/2018/11/transportation-containers.jpg

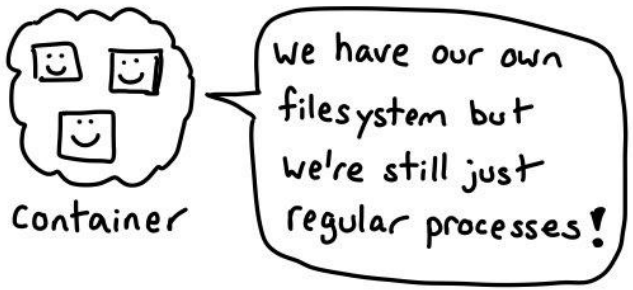
History

- Chroot circa **1982**
- FreeBSD Jails circa 2000
- Solaris Zones circa 2004
- Meosys - MetaClusters with Checkpoint/Restore 2004-05
- Linux OpenVZ circa 2005 (not in mainstream Linux)
- AIX WPARs circa 2007
- LXC circa 2008
- Systemd-nspawn circa 2010-2013
- Docker circa **2013**

Julia Evans
@b0rk

what's a container?

a Linux container is a group of processes



Linux containers are isolated from other processes

they can have their own:

- users
- network namespace
- filesystem
- process IDs
- memory / CPU limits

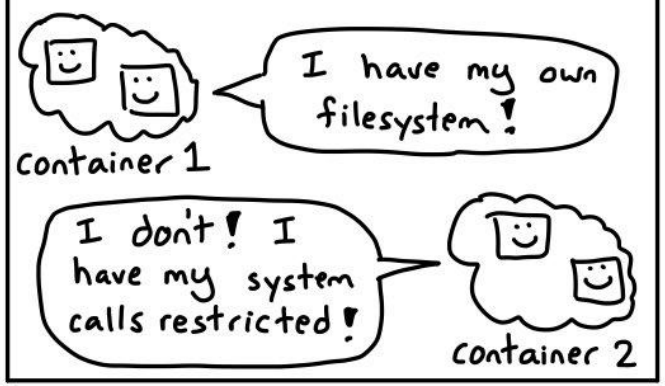
Kernel features that isolate Linux containers

cgroups
namespaces
capabilities
seccomp-bpf

there are many ways to run Linux containers

runc
systemd-nspawn
LXC
Docker ← Docker uses runc under the hood
your own homegrown bash script

and containers can be set up in different ways



extra confusion:

"container" sometimes means "lightweight VM"

Fargate and kata Containers are actually VMs and not Linux containers (they don't share a kernel with other containers)

Source: https://twitter.com/b0rk/status/1225445956734390273

Build a container:

00_prep/k8s-ws/

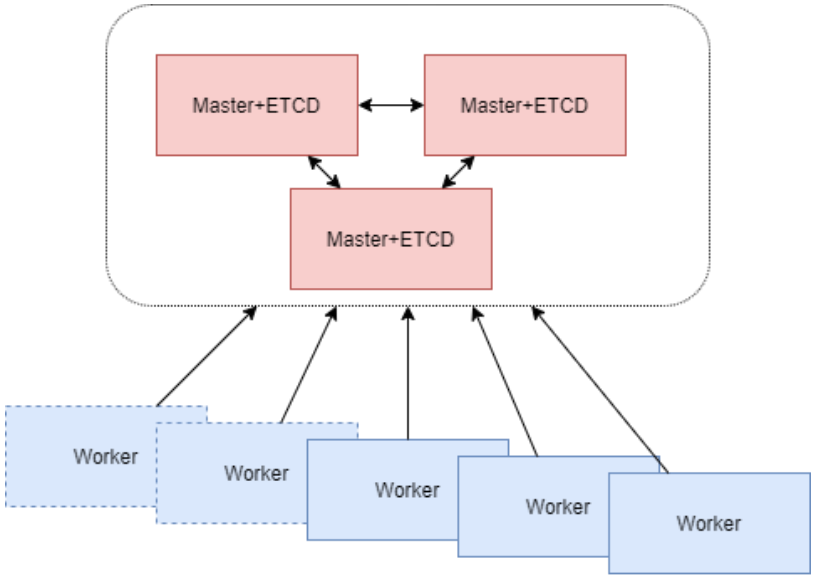
Exercise

Kubernetes

Architecture

What is Kubernetes?

- Widespread Container Orchestration solution
- Similar concepts to Docker Swarm
  - Uses RAFT for master node
  - Secrets
  - Services / Deployments
- Former Borg[1] (originally developed by Google)
- Uses etcd KV-Store



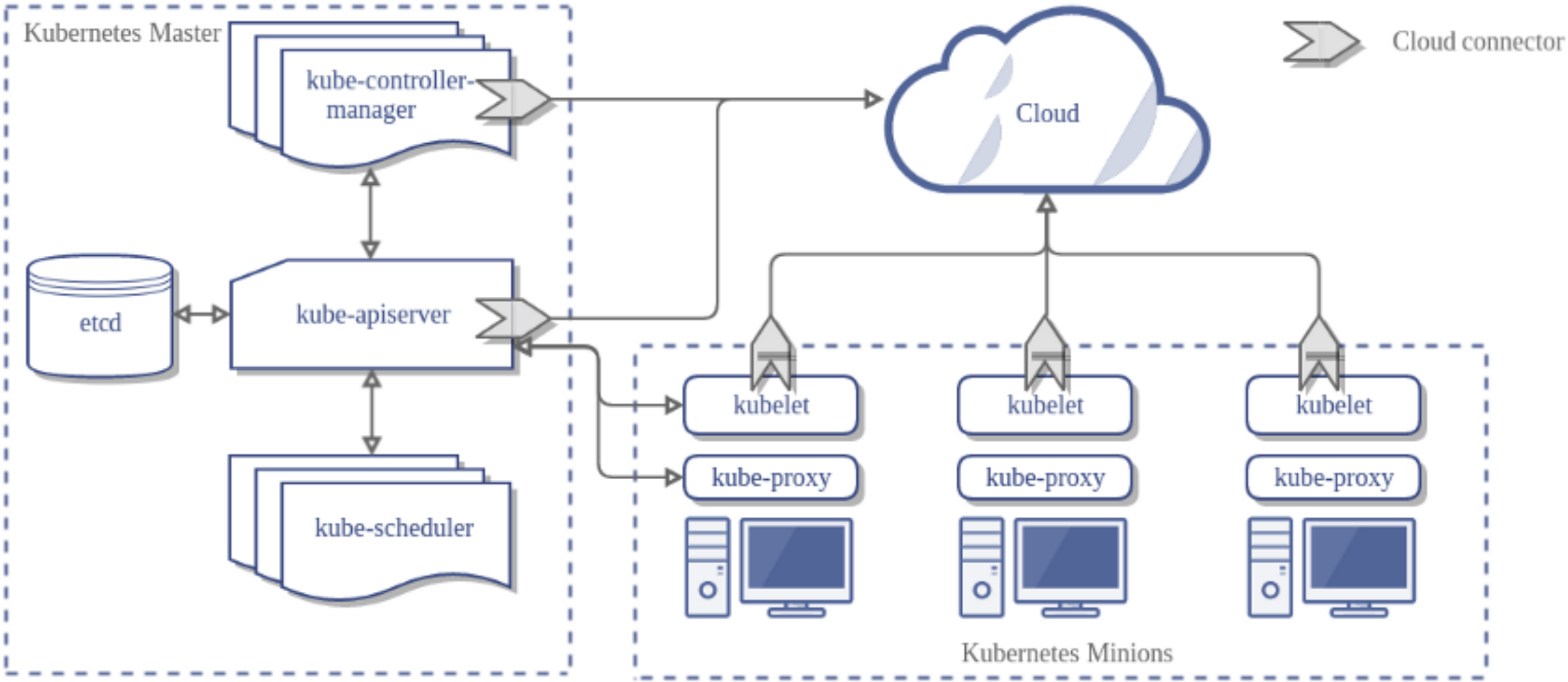
[1]: https://research.google.com/pubs/pub43438.html

What is Kubernetes?

- Manager nodes handle cluster management tasks
  - Cluster state maintenance
  - Scheduling of services
  - API endpoints
- Worker nodes provide executes the wotkload

Kubernetes Clusters

- Builds on the Raft Consensus Algorithm
- Manager nodes keep the cluster state consistent in etcd
- In case of failure:
  - majority of nodes needs to agree on values
  - (N-1)/2 failures tolerated, otherwise no more requests are processed



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

Kubernetes CLI (kubectl)

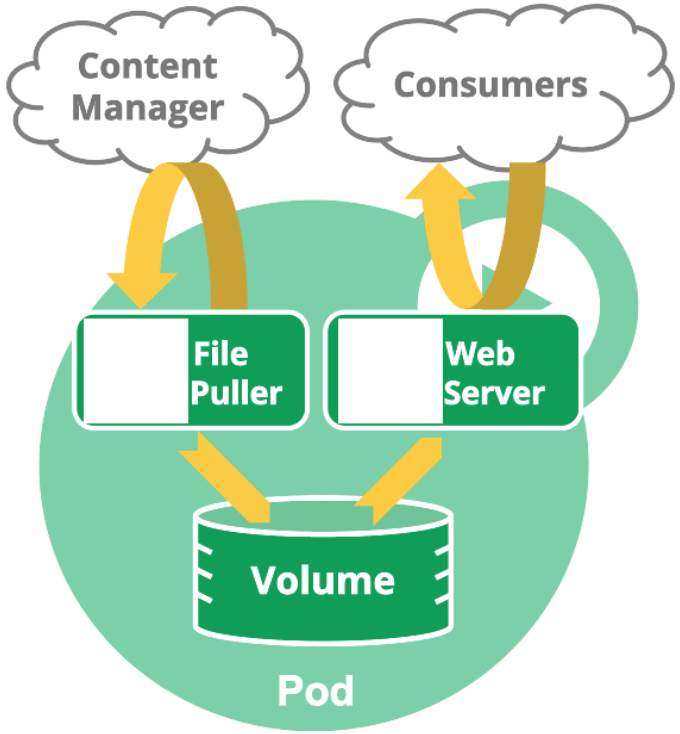
- kubectl get
  - Get a resource
  - i.e. kubectl get nodes
- kubectl describe
  - Describe a resource
- kubectl apply
  - Apply a resource file
- kubectl create / kubectl run / kubectl expose
  - Create / manage services / deployments

Kubernets

Pod

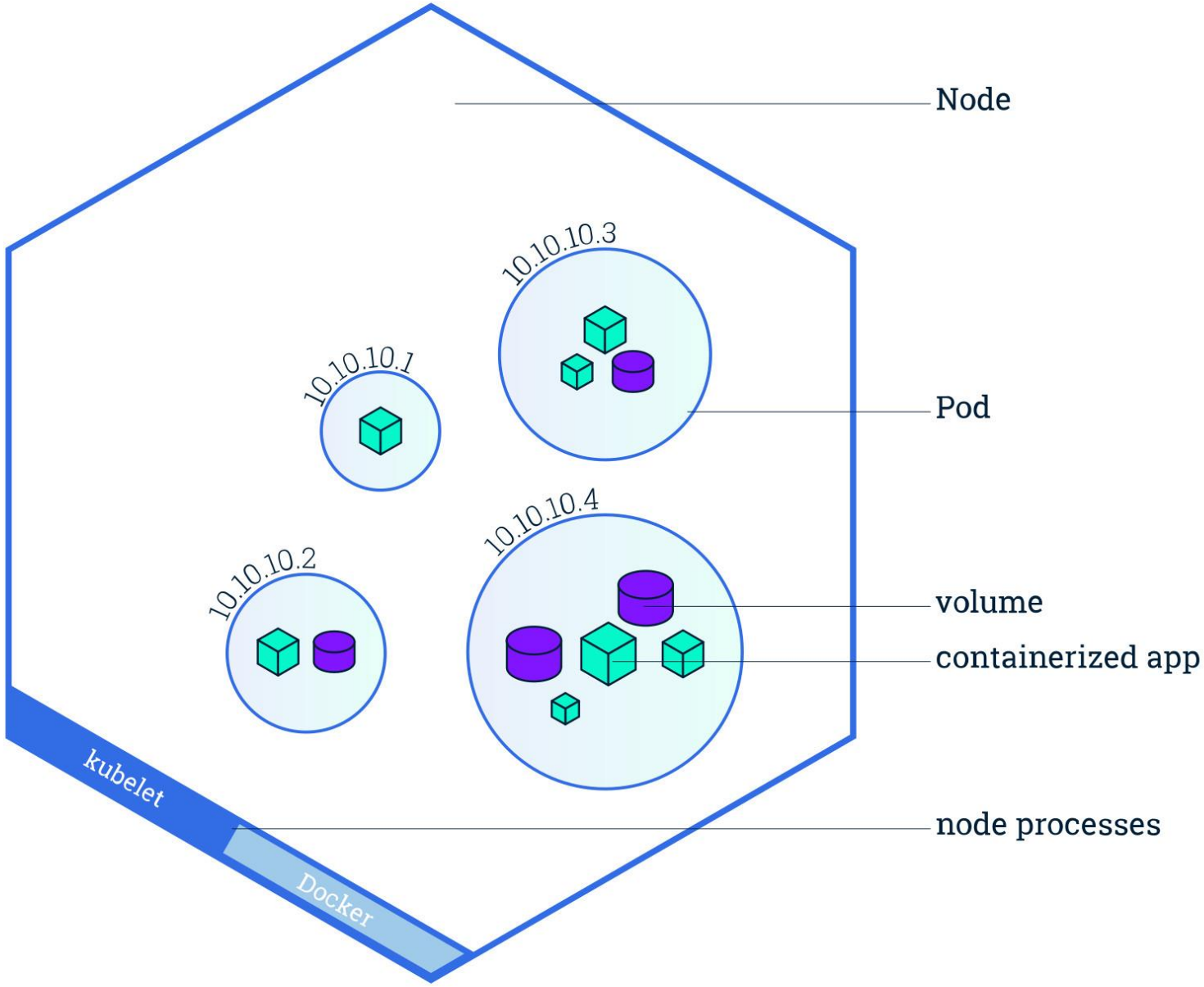
Pod

- Smallest unit in the Kubernetes ecosystem
- A pod is a group (>=1) of containers
- Are co-scheduled and share resources
- Shared IP/Ports
- C-to-C communication: IPC
- Best practice: 1 Container per Pod
- Health checks for containers



Source: https://kubernetes.io/docs/concepts/workloads/pods/

Pods



Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore-intro/

Start a pod

01_getting-started/00-pod/

Exercise

Pod Lifecycle (1/2)

- Stati:
  - Pending: Created, but not scheduled yet
  - Running: Scheduled
  - Succeeded: All containers terminated successfully
  - Failed: At least one container terminated with an error
  - Unknown: API server was not able to query state
  - CrashLoopBackoff: Container failed to start, trying it again
  - ImagePullBackOff: Image cannot be pulled

Pod Lifecycle (2/2)

- Container probes
  - livenessProbe
  - readinessProbe
  - Implemented by
    - ExecAction
    - TCPSocketAction
    - HTTPGetAction
- On stop
  - PreStop hook called
  - Containers are killed

Kubernets

Deployment

Workloads

- Pod
- ReplicaSet (former ReplicationController)
- Deployment
- StatefulSet
- DaemonSet
- (Cron-)Job (Batch-style workload)

ReplicaSet

- Successor of ReplicationController
- Ensures that no. of pod replicas are running
- Mostly used in combination with Deployments

Deployment

- ReplicaSet with control
- Regularly check status of pods
- Update & rollback possible
  - Rolling updates
  - Blue-Green deployments

DaemonSet

- Bypasses the default scheduler
- Run a pod on every node
- Used to i.e. bootstrap
  - Network plugins make use of it

Job

- Creates one or more pods
- Make sure a specified number of them terminates successfully => Job done
- i.e. simple job which has to run once
  - Job will start new pod, if first pod fails or is deleted (i.e. due to node hardware failure)
- Multiple pods possible
- Cronjob: timed Job

Create a Deployment

01-deployment

Exercise

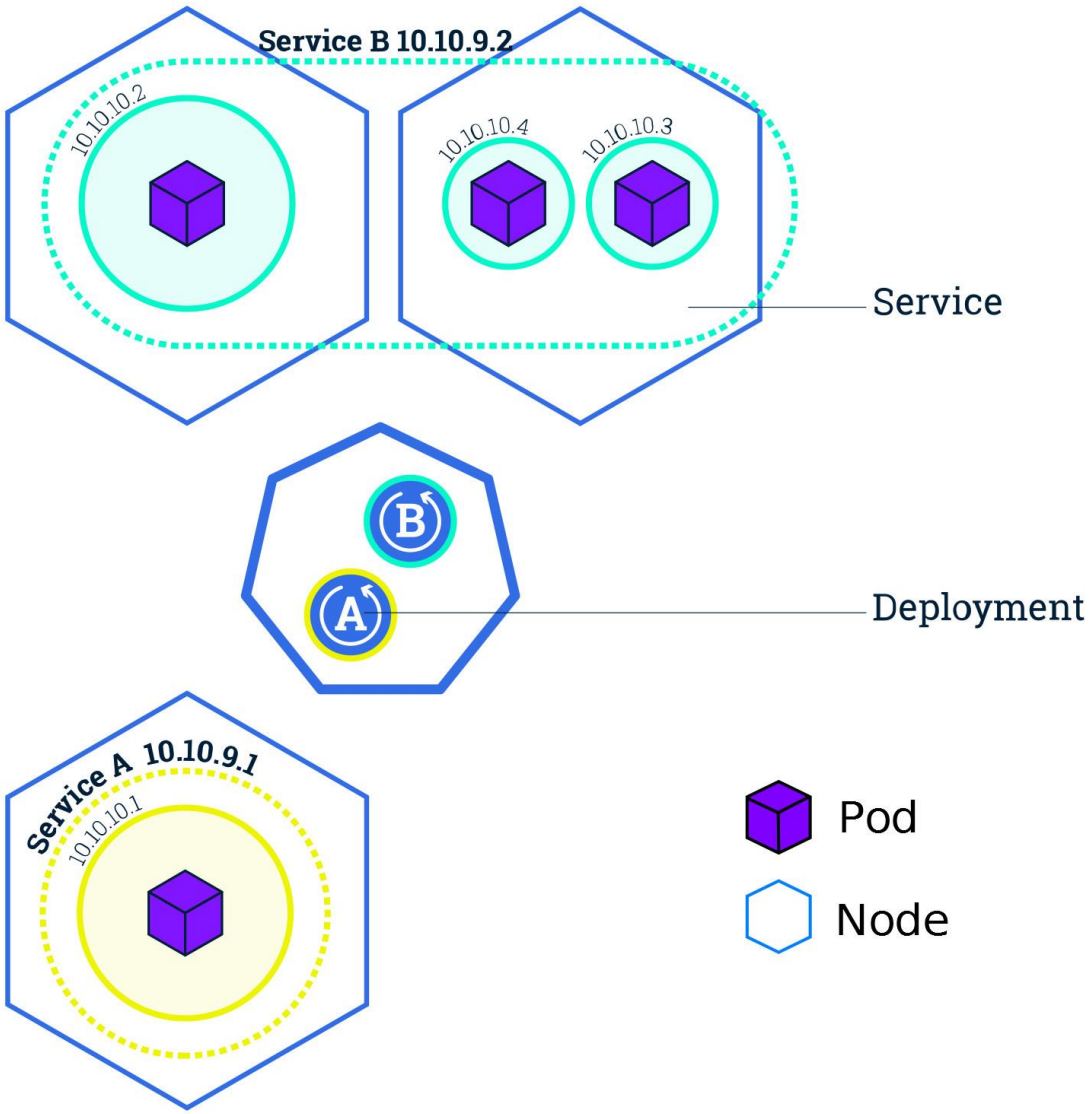
Kubernets

Service

Services

- Problems
  - Pods are mortal
  - Multiple Pods for one service exist
  - No access from outside
- A Service defines a set of Pods and a policy (i.e. ports)
  - Including a VIP
- Service types
  - ClusterIP (only internal, default)
  - NodePort (binds port on each node)
  - LoadBalancer (use a cloud provided load balancer)
  - ExternalName (CNAME entry)

Services



Source: Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose-intro/

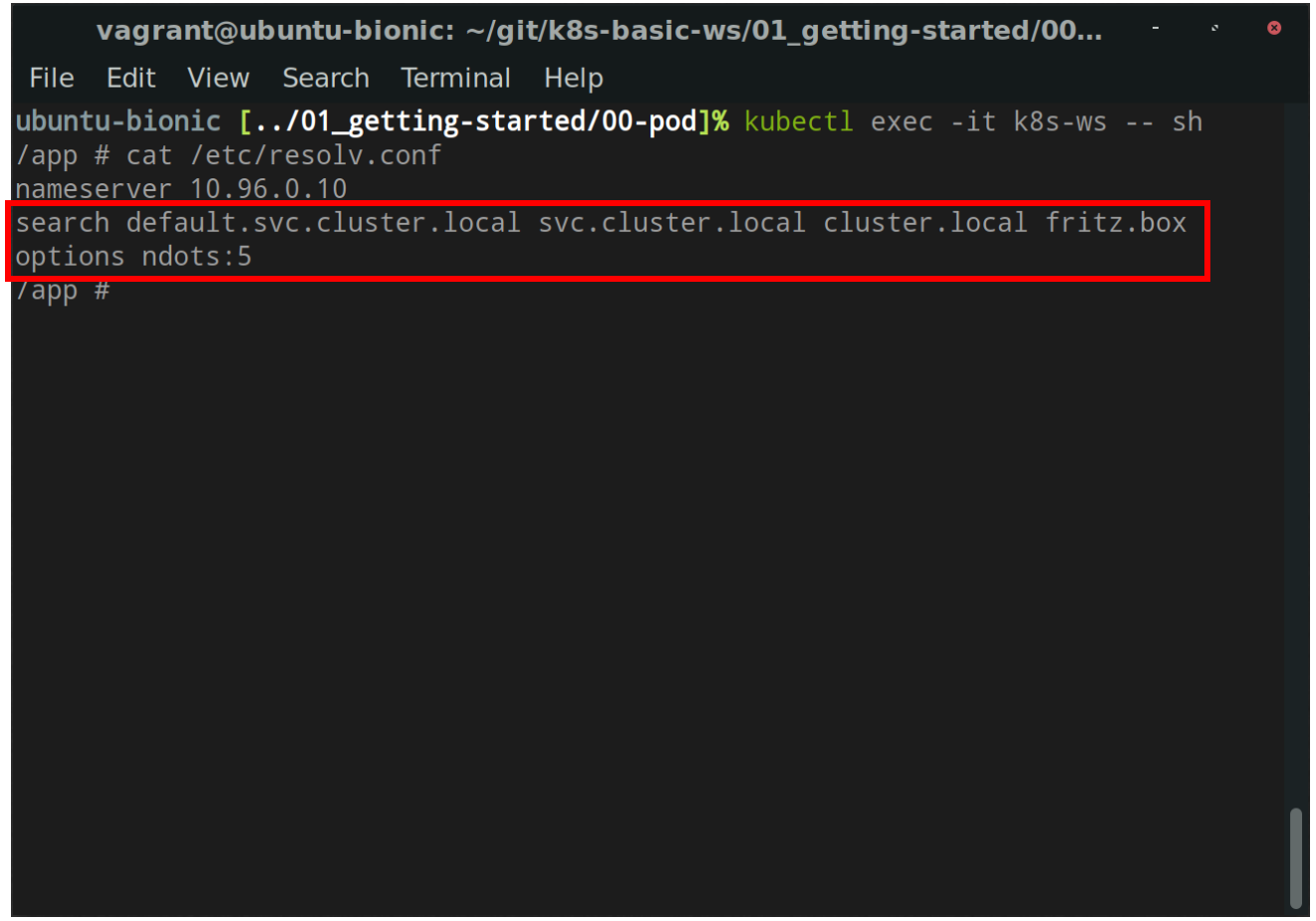
Service Discovery (1/3)

- Service Discovery is build up on DNS
- CoreDNS is used as backend
- Every service receives a unique DNS name
  - i.e. service foo in namespace bar
    - Pod in same namespace: foo
    - Pod in other namespace: foo.bar
  - Can also be enabled for pods
    - 1-2-3-4.default.pod.cluster.local

Service Discovery (2/3)

- A Record
  - <service>.<ns>.svc.<zone>.
  - kubernetes.default.svc.cluster.local.
- SRV Record (service record)
  - _<port>._<proto>.<service>.<ns>.svc.<zone>.
  - _https._tcp.kubernetes.default.svc.cluster.local.
- PTR Record (reverse DNS)
  - <a>.<b>.<c>.<d>.in-addr.arpa.
  - 1.0.3.10.in-addr.arpa.

Service Discovery (3/3)



Create a Service

02-service

Exercise

Kubernets

Secrets & Configmaps

ConfigMaps

- Pass information to workloads (configuration)
- Decoupling of configuration from images
- Consume by ENV or mounted volumes

Secrets

- Similar to ConfigMaps

- Pass information to workloads (secrets)
- Decoupling of secrets from images
- Consume by ENV or mounted volumes

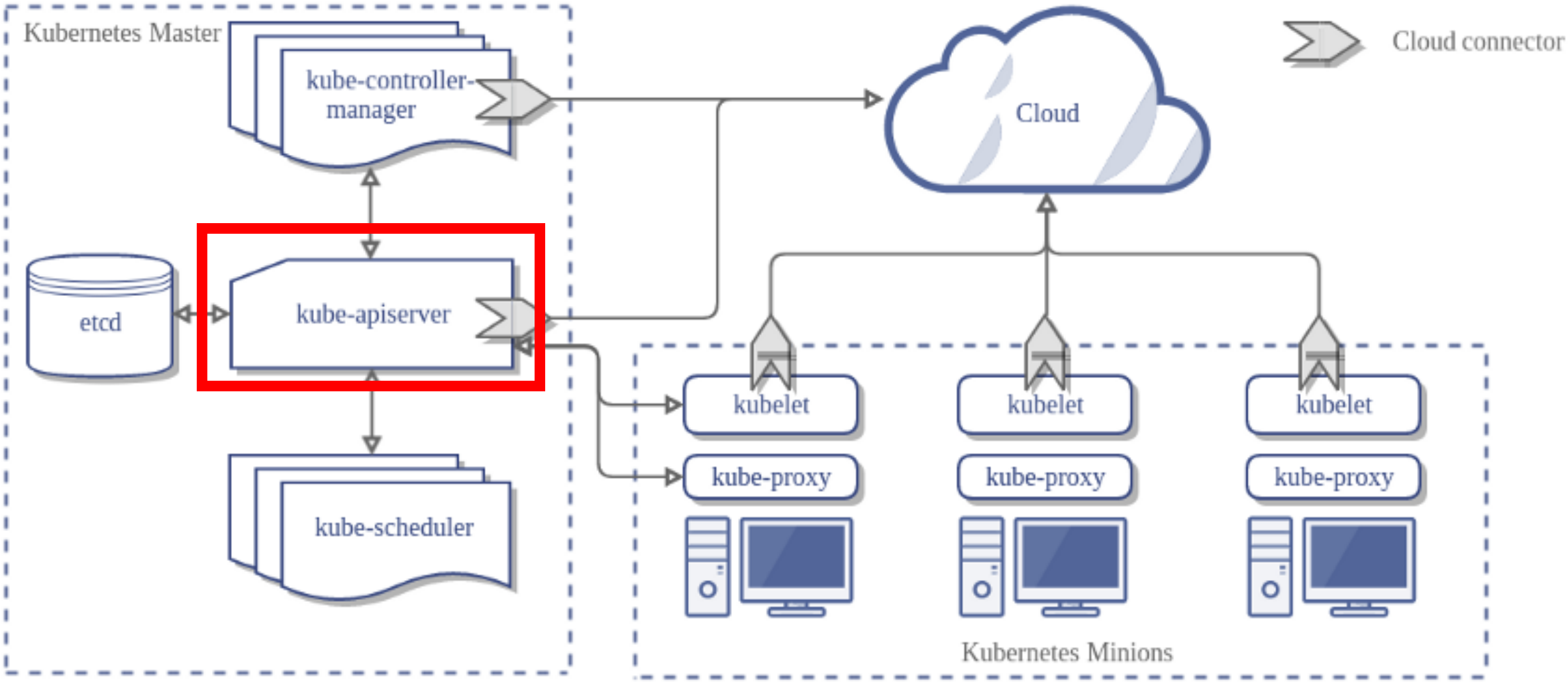
Secrets & Configmaps

03-secrets-configmaps

Exercise

Kubernets

Master Components, etcd & Yaml Files



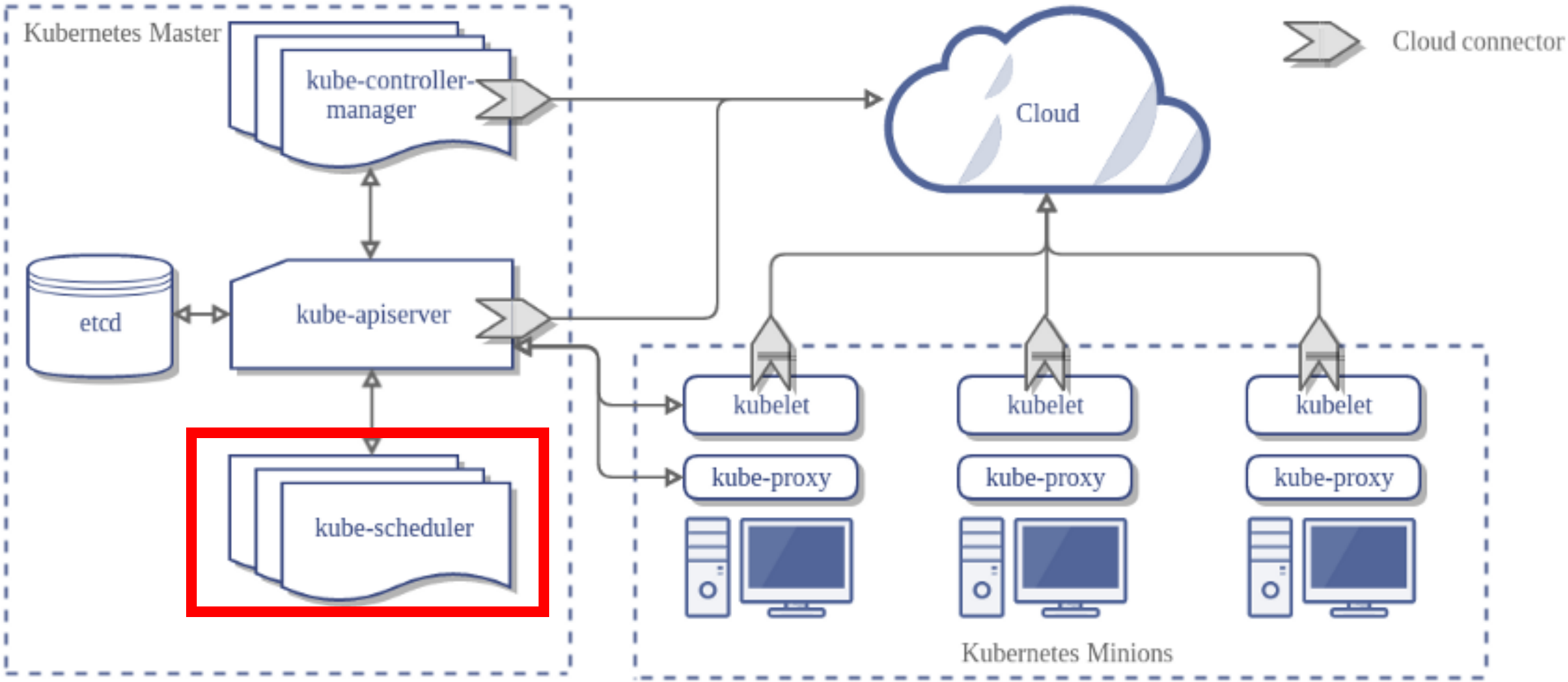
Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

Kubernetes API

- REST API / CRUD
- HTTP API with JSON
- Versioned
  - /api/v1
  - /apis/extensions/v1beta1
- Base path
  - /apis/batch/v1/namespaces/….
  - /apis/extensions/…
- Core API (historically not /apis/core/v1)
  - /api/v1
- Custom Resource Definitions (CRD) possible

Kubernetes API (Objects)

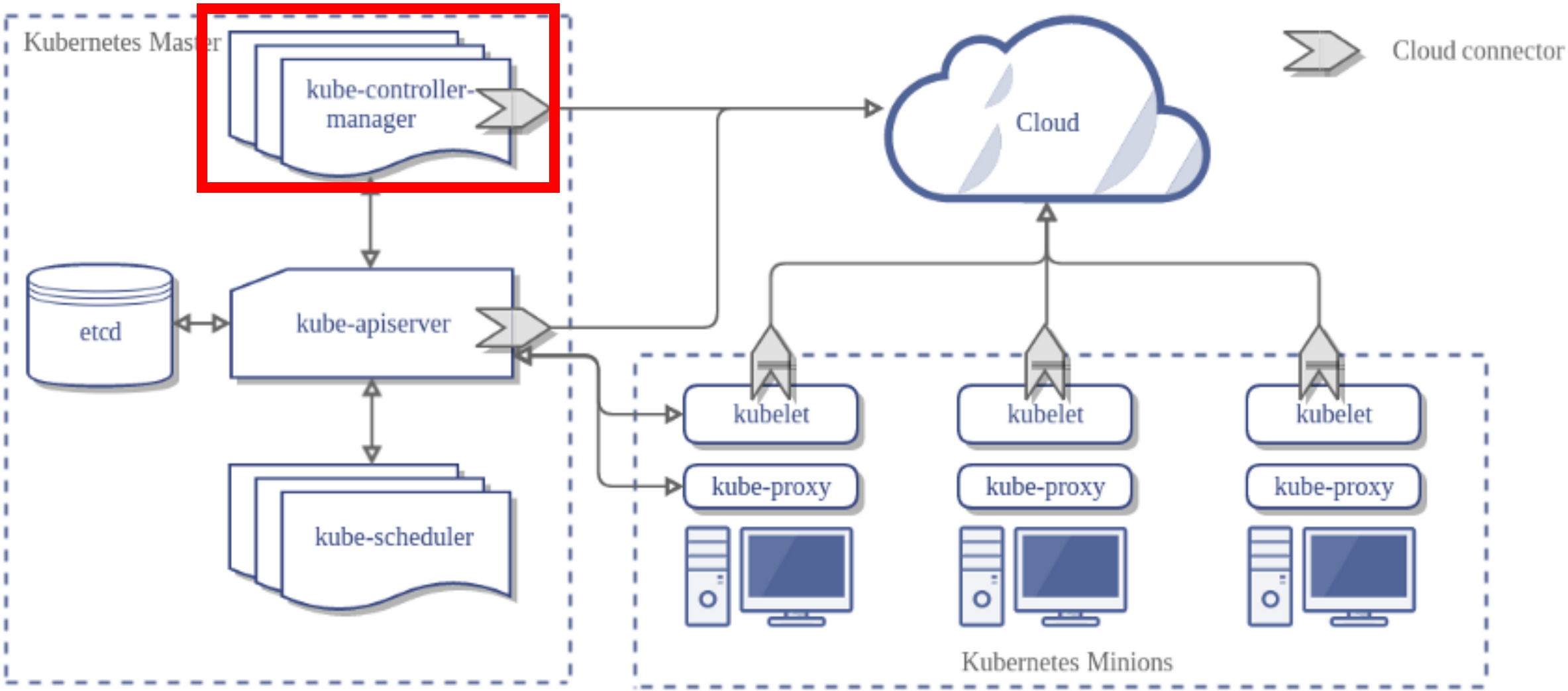
- apiVersion
- kind (Object type)
  - Persistent entity (Deployment, Service, Pod, …)
  - List (collection of resources)
  - Special purpose (i.e. bindings, status)
- metadata (labels, name)
- spec (desired state of the object)
  - “Kind”-specific data



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

kube-scheduler

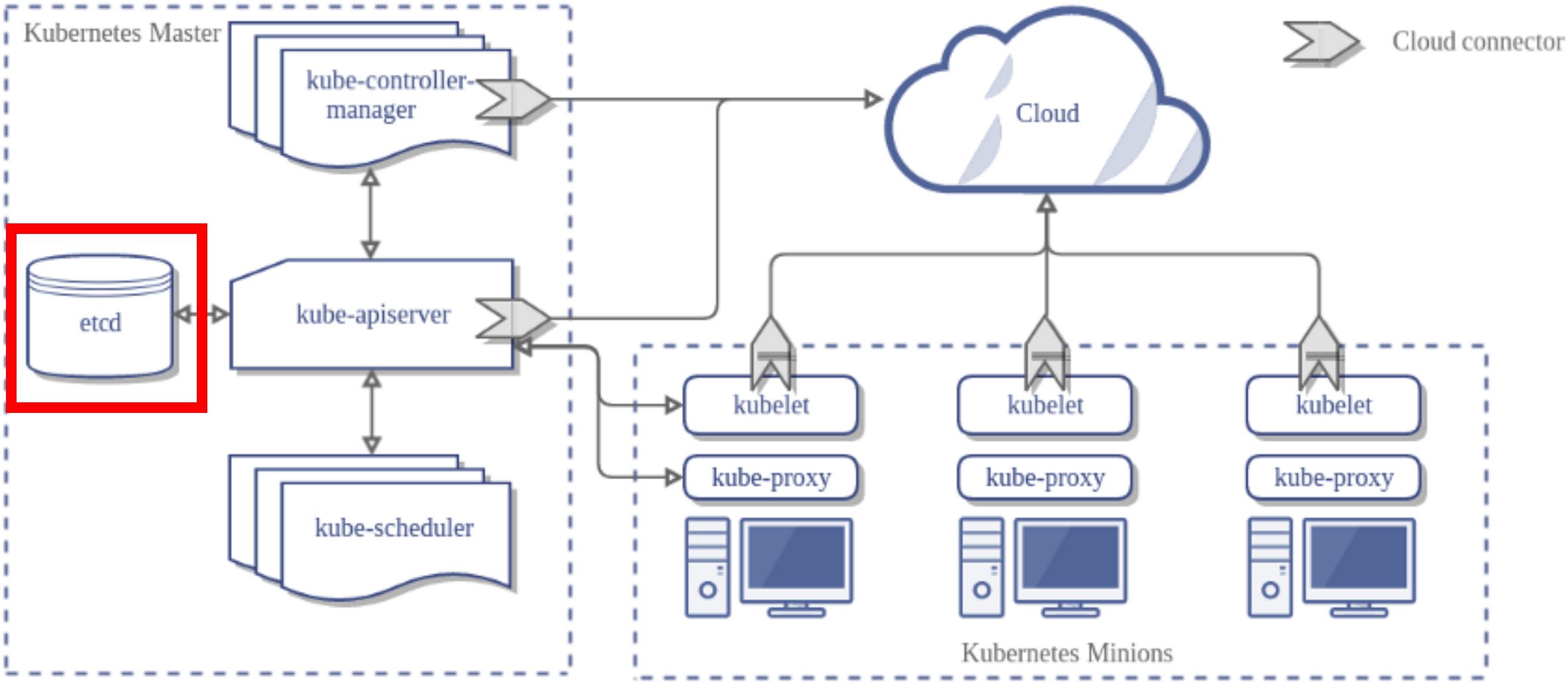
- Binds unscheduled pods to nodes
- Pluggable: multiple (custom) schedulers possible



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

Controller Manager

- Single binary on master which contains controllers
  - Endpoint Controller (manages Endpoints)
  - Node Controller (manages nodes)
  - Replication Controller (maintain no. of pods)
  - Service Account and Token Controller
  - Cloud Controller (cloud integration)



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

etcd

- Distributed persistence layer (key-value storage)
- Stores the cluster state
- Distribution / Replication
  - Leader elections
  - Fault tolerance
  - Distributed locks
- v2 vs v3
- Authn / Authz (disabled by default)
- *etcdctl* to control etcd

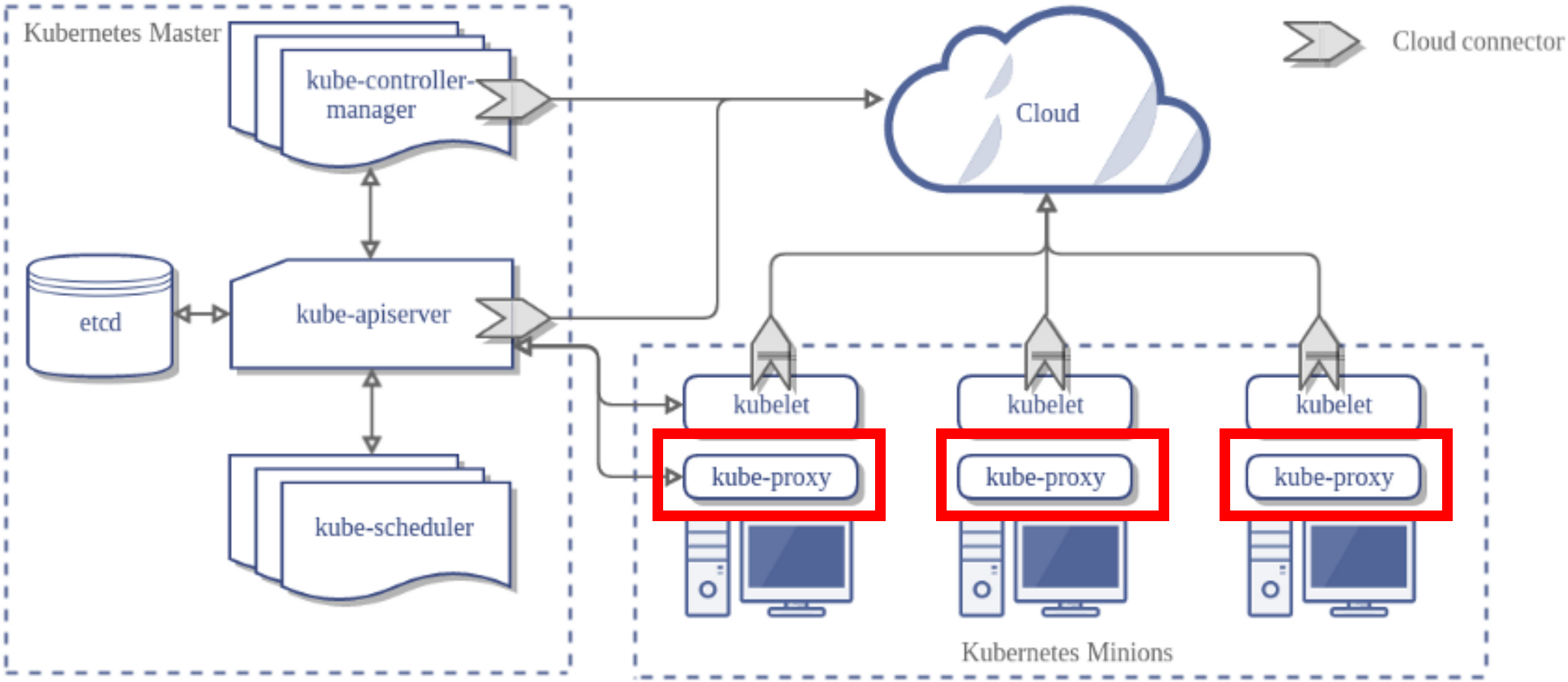
Yaml Files

04-yaml

Exercise

Kubernets

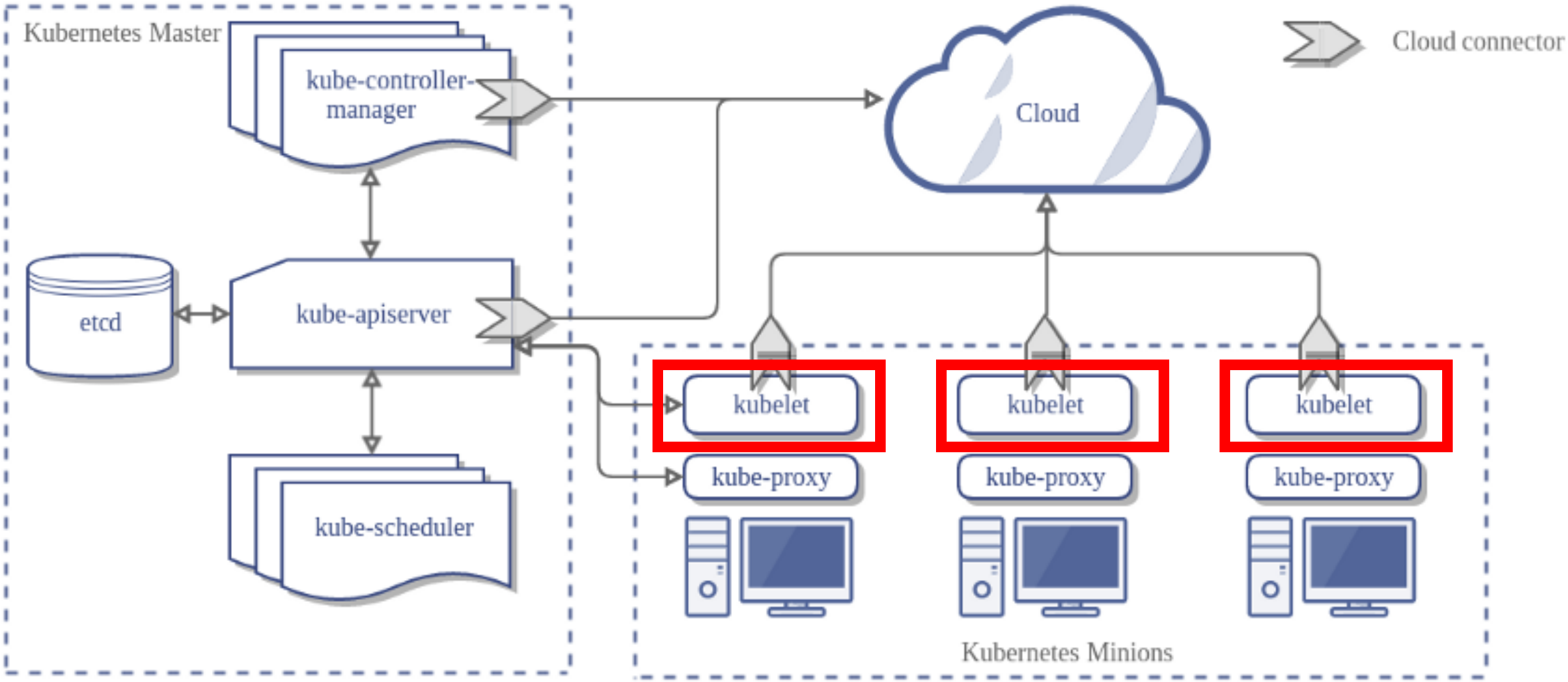
Node Components



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

kube-proxy

- Route packages to pods
- Load balancer for pods
- Maintain network rules (iptables)
- “Services” for the nodes



Source: https://kubernetes.io/de/docs/concepts/architecture/cloud-controller/

kubelet

- Manages pods, containers, volumes, secrets
- Agent which runs on every computing node (nodes + sometimes master)
- Authentication for kubelet (port 10250)
  - X509 client certificate
  - WebHook
    - Calls *TokenReview* API

Manage the environment

05-env

Exercise

Kubernets

Labels & Selectors

Labels and Selectors

- Labels are key / value pairs that are attached to objects (i.e. Pods)
  - i.e. environment (dev, testing, prod), release (stable, dev), tier (frontend, backend, cache)
- Selectors are used to query objects based on labels
  - Restart all pods with label production
  - Required for Service, Ingress, LoadBalancer, NetworkPolicies

Manage the environment

06-link-configmap

Exercise

Manage the environment

07-link-secret

Exercise

Kubernets

Ingress

Ingress

- L4-L7 decisions
  - Defines rules
    - Redirects traffic to Services
- “Service load balancer”
- SSL termination support
- Managed by Ingress Controller
  - Part of *kube-controller-manager* binary
  - Supports GCE and nginx controllers

Manage the environment

08-ingress

Exercise

Kubernets

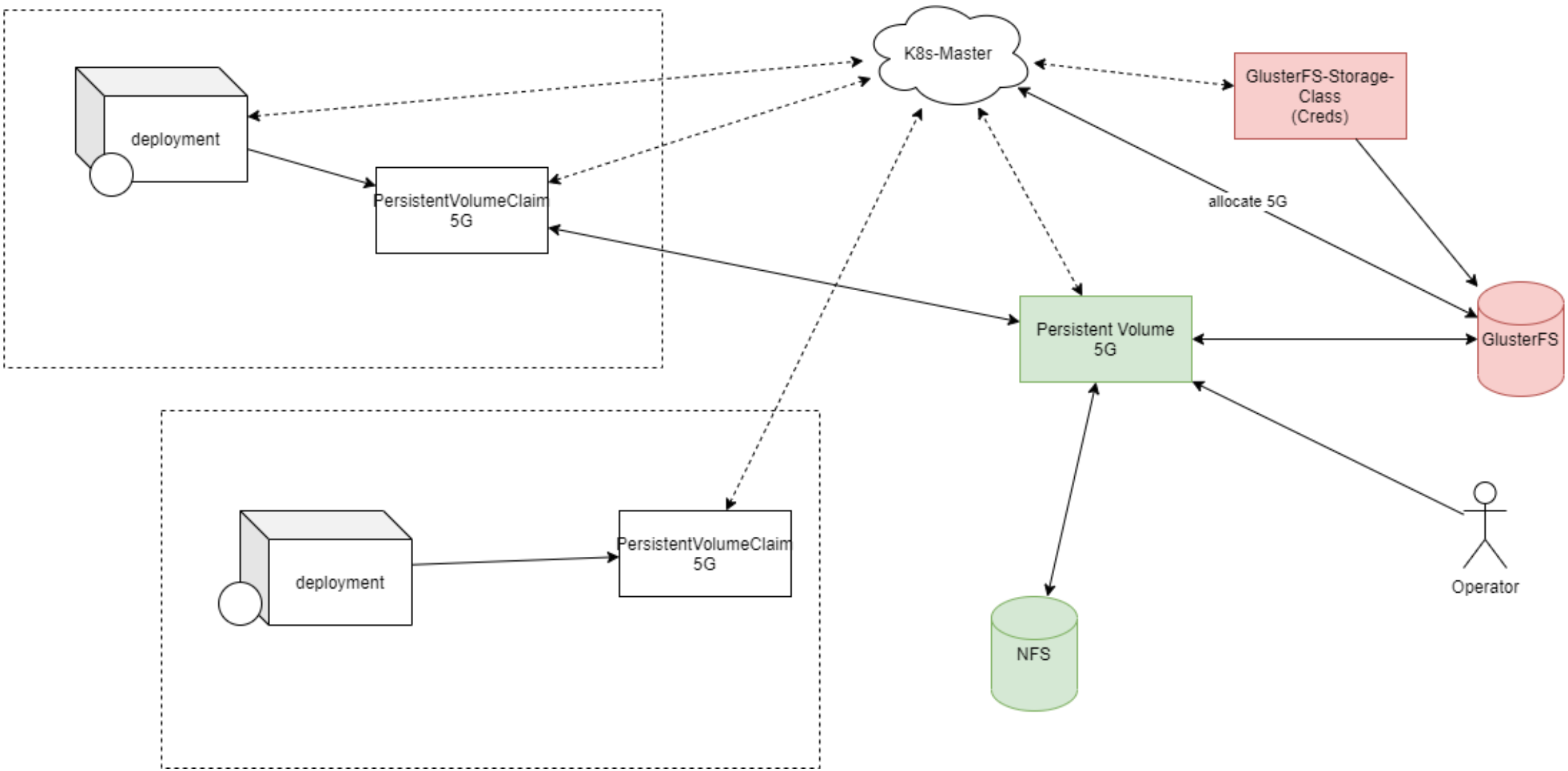
Persistence

Volumes

- Per default data is ephemeral
  - Data lifetime == Pod lifetime
  - i.e. used for caching
- Persistence
  - Data lifetime != Pod lifetime
  - Concept of Volumes
  - Core: just a directory with specified backend
  - Types: emptyDir, hostPath, nfs, iscsi, glusterfs

PersistentVolumes API

- Abstraction of details how storage is provided and consumed
- PersistentVolume (PV)
  - Storage provisioned by administrator
  - Labeled
- PersistentVolumeClaim (PVC)
  - Request for storage by user
  - Request specific class, size and access mode
- StorageClasses (i.e. SSD, HDD)
  - provisioner (i.e. AWS Block storage, iSCSI, NFS)
  - parameter (i.e. type, zone, …)
  - reclaimPolicy (i.e. Delete or Retain)



Manage the environment

09-pvc

Exercise

Thanks for your Attention

- Any Further Question?

Jan Harrie

Mail: jharrie@harrie-consulting.de
Twitter: @NodyTweet
Blog: https://blog.nody.cc/
GitHub: https://github.com/NodyHub
Dockerhub: https://hub.docker.com/u/nodyd