# Introduction to Kubernetes

Part 1

# Course Goal

By the end of the course, you should know:

- What is Kubernetes, how access it and install local sandbox
- What is Kubernetes architecture, principals, core objects
- How to deploy an application to Kubernetes
- How to troubleshoot Kubernetes applications
- Where to look for further information

# Course Plan

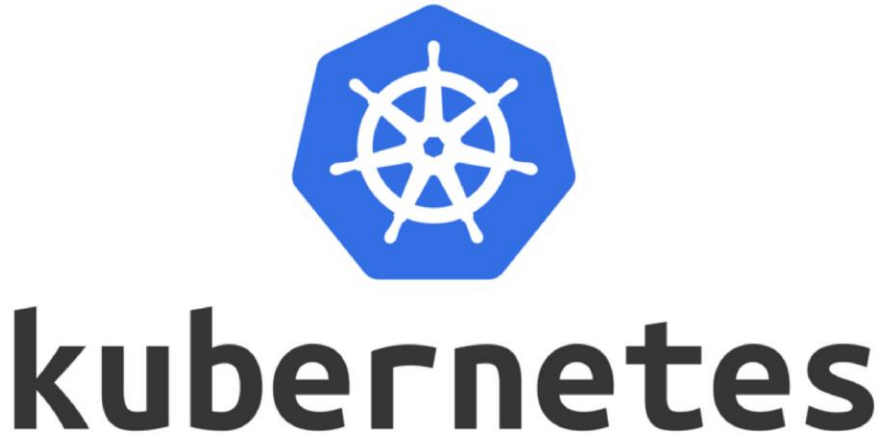
1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

# Course Plan

1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

# What is Kubernetes

- Open-source platform to orchestrate containerized applications
- Developed in Google based on Borg
- Donated to Cloud Native Computing Foundation (CNCF)



# What is Kubernetes

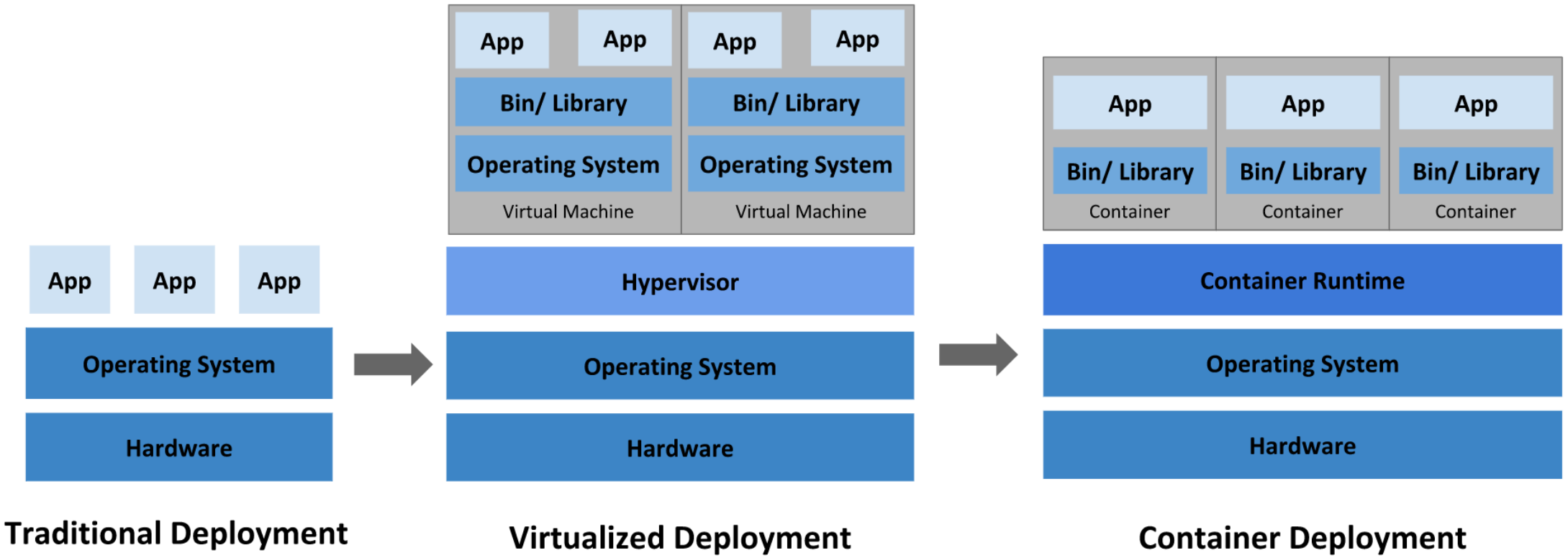
Kubernetes **glues compute and storage** resources into a single pile and gives an **API to schedule workloads** on them.

It also gives abstractions to deal with service-discovery, rollouts, self-healing, RBAC, scheduling customization, policies, and leaves extensibility points.

# Course Plan

1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

# What is Container



# What is Container

Container – 1..N processes that are isolated from the rest of the system

# What is Container

Container is based on Linux features:

- namespaces (what can see)
- cgroups (what can use)

# What is Container

Use-cases:

- Application (production and local)
- Working or Build environment
- Instead of installing applications/tools

# What is Container

Containers are distributed as Images

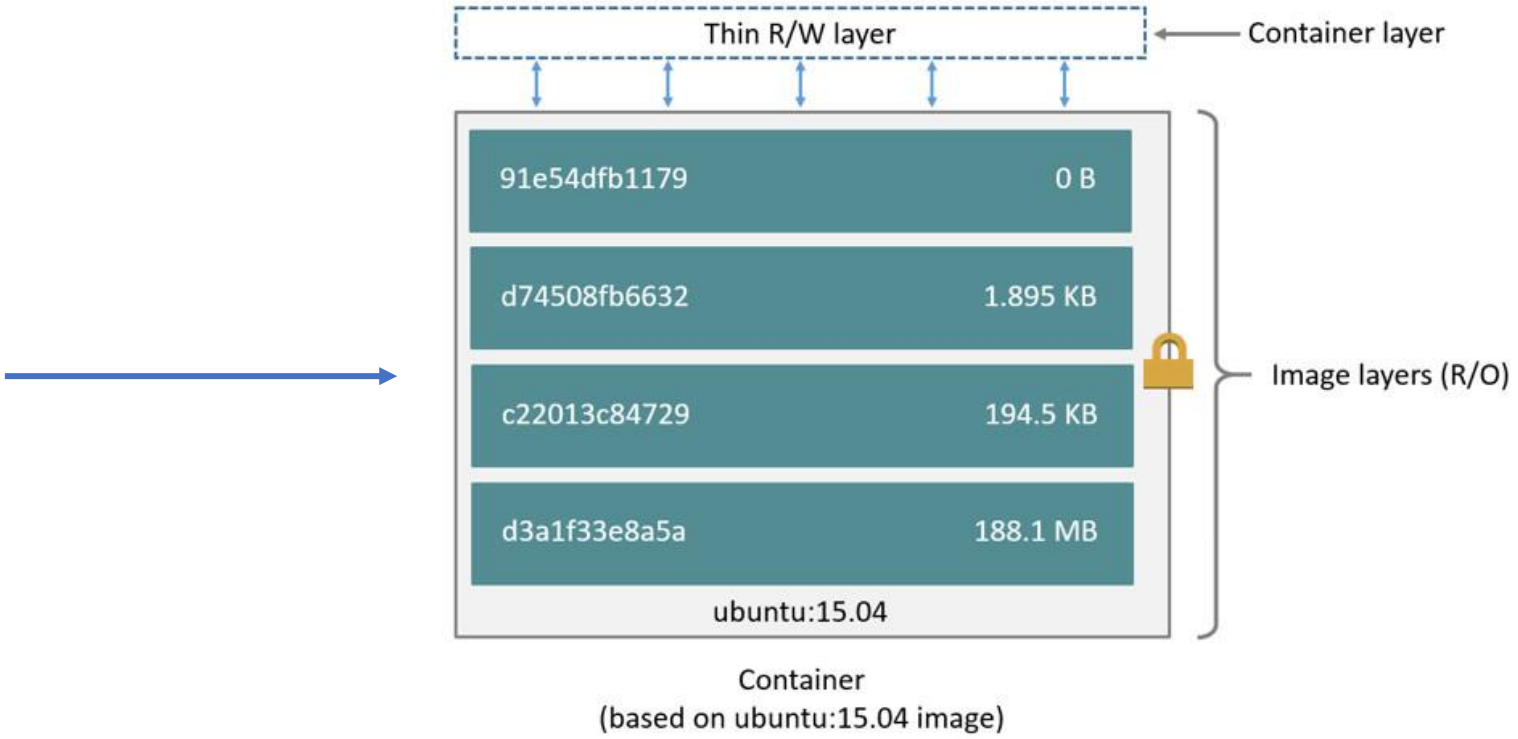
# What is Container

De facto standard for container image creation is Dockerfile

```
1  FROM alpine:3.12.3
2
3  LABEL org.opencontainers.image.title="Dockerfile Sample"
4  # https://github.com/opencontainers/image-spec/blob/master/annotations.md#pre-defined-annotation-keys
5
6  RUN mkdir /usr/share/sample_app
7  WORKDIR /usr/share/sample_app
8  # COPY from_path to_container_path
9
10 ENV HELLO_ENV_VAR="Hello from container"
11
12 EXPOSE 8080
13 USER 10001
14
15 ENTRYPOINT ["sh", "-c", "echo $HELLO_ENV_VAR in $PWD"]
```

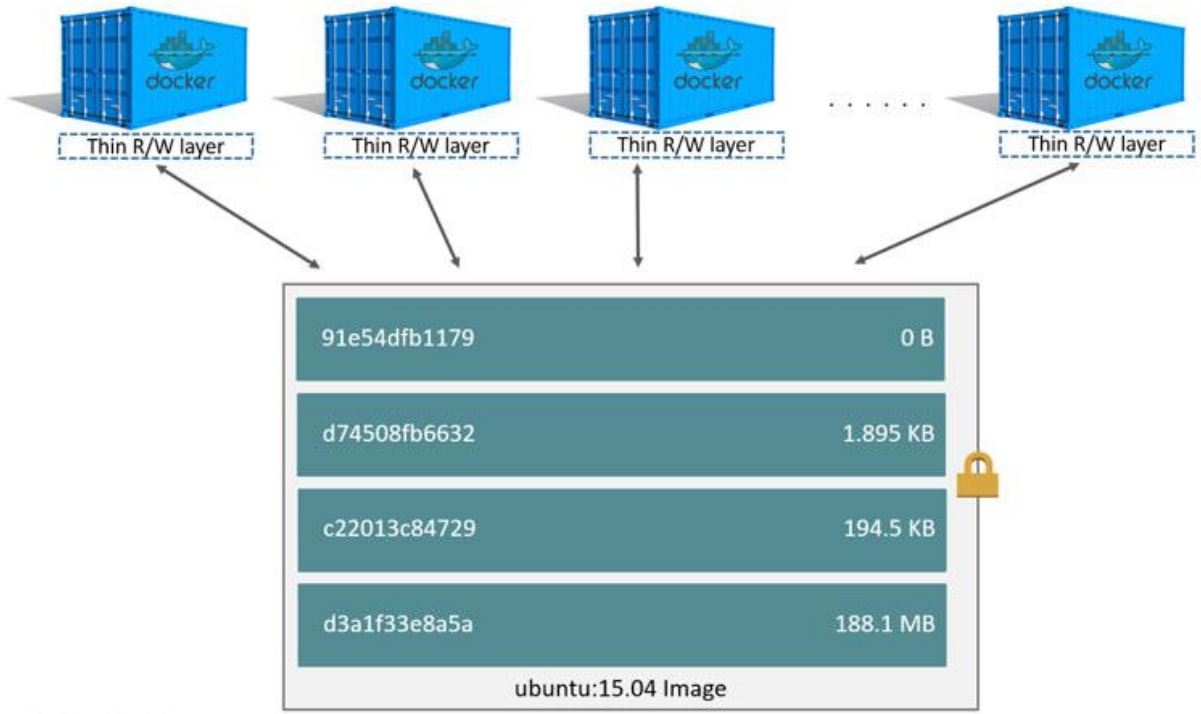
# What is Container

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```



# What is Container

container is an instance of an image



# What is Container

- Container is disposable
- When container is gone – created in runtime data fades away
- If data should survive – use volume

# What is Container

Container Image:

- Immutable
- Repeatable and portable
- Layered
- Uses union filesystem with copy-on-write (CoW) strategy

# What is Container

Image name format: `{registry:optional}/{image_name}:{tag}`

- docker.elastic.co/kibana/kibana-oss:7.2.0
- k8s.gcr.io/nginx-slim:0.8
- redis:6.0.10-alpine3.12
- ubuntu:latest

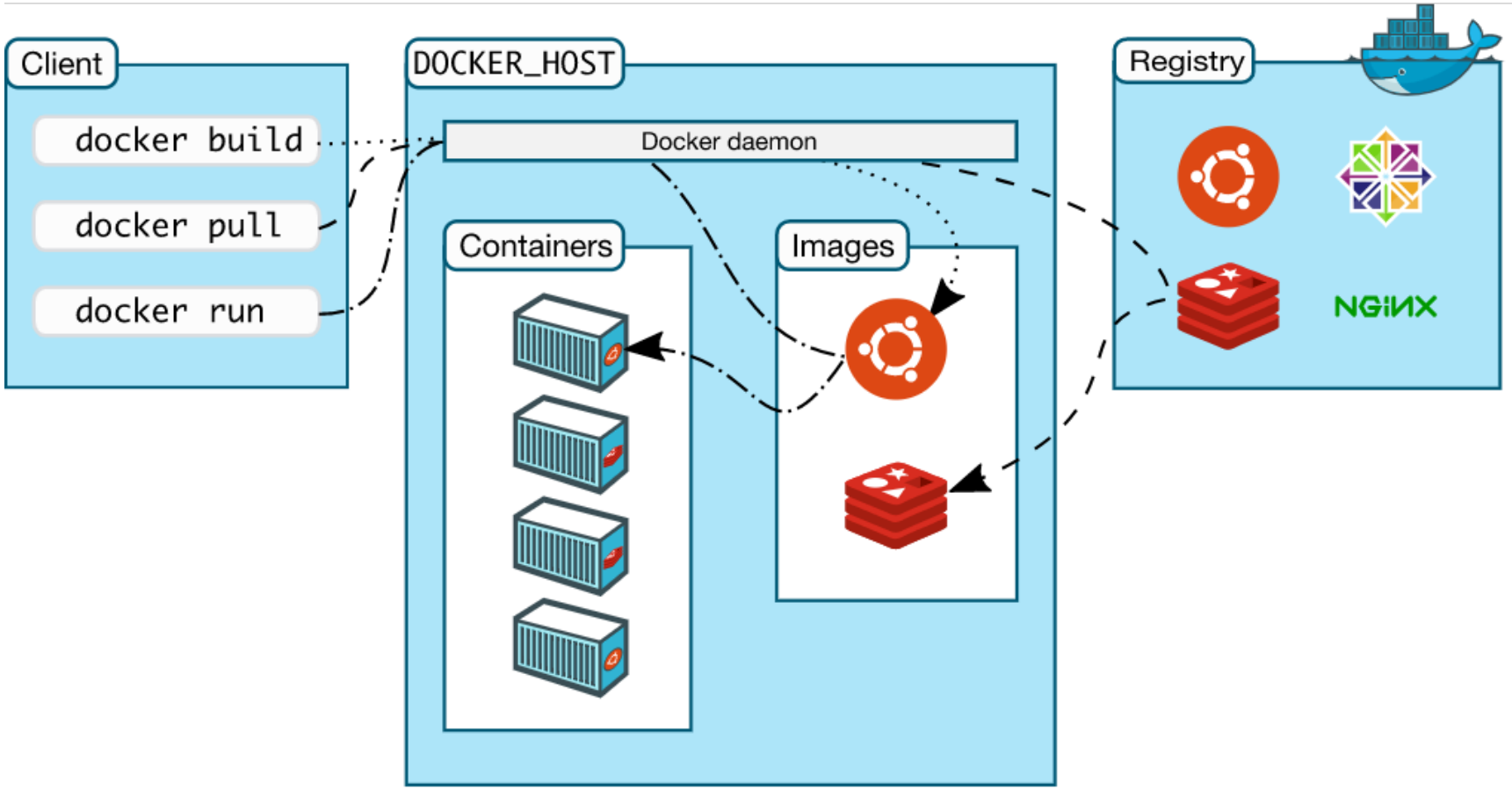
# What is Container

Container Registry – service, that stores and distributes container images

# What is Container

Container Runtime – software which runs and manages containers

# What is Container



# What is Container

- `docker build -t image:latest .` -> creates image
- `docker push image:latest` -> pushes image to the registry
- `docker pull image:latest` -> pulls image from the registry
- `docker run image:latest` -> creates container from image in a runtime

# What is Container

Demo

# What is Container

Docker alternatives

- buildah + podman + skopeo

Runtimes:

- cri-o (based on crun)
- Containerd (based on runc)

# What is Container

Tips & Tricks

- tags are mutable
- do not use latest tag
- use linters/scanners to catch issues at build time
- registries could have limits

# Containers additional materials

Alternative containers introduction:

- VMware course: video lectures
- Beginner-friendly introduction to containers article (what containers are)
- Docker-handbook article (how to use docker)

# Containers additional materials

Deep-dives:

- How Unix Works: article
- Docker Layers article
- A deep dive into Linux namespaces: part-1 - part-4
- How containers work video
- Anatomy of Container video
- Best-practices for writing dockerfiles: article

# Course Plan

1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

# Kubernetes 10000-foot view

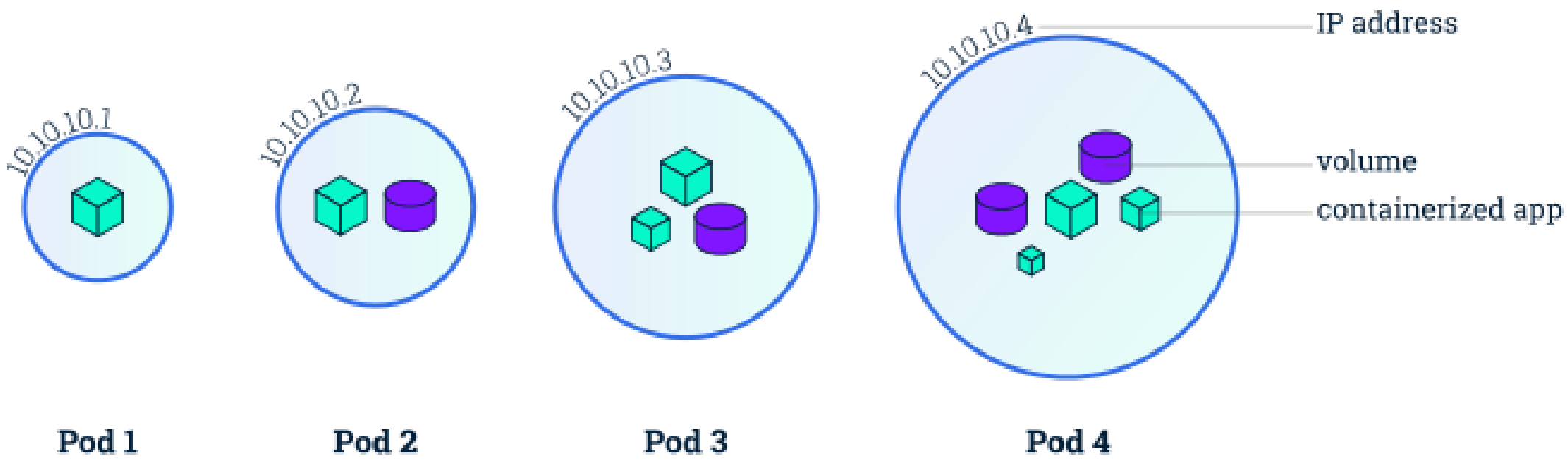
Kubernetes is a distributed system, which has

- Control-plane
- Data-plane

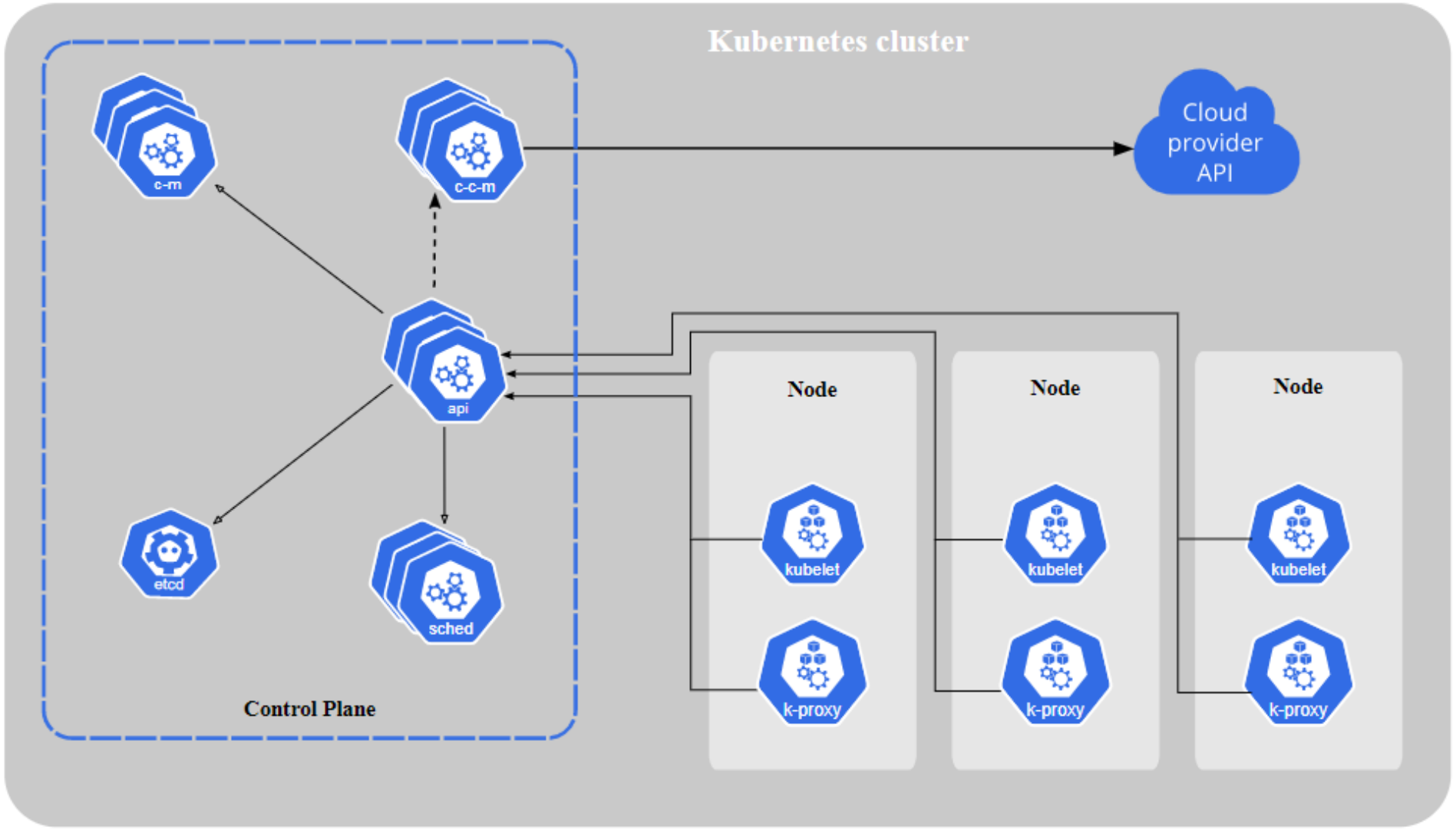
# Kubernetes 10000-foot view

Pod – is a core object of Kubernetes

It wraps 1..N containers and 0..M volumes into a single unit



# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

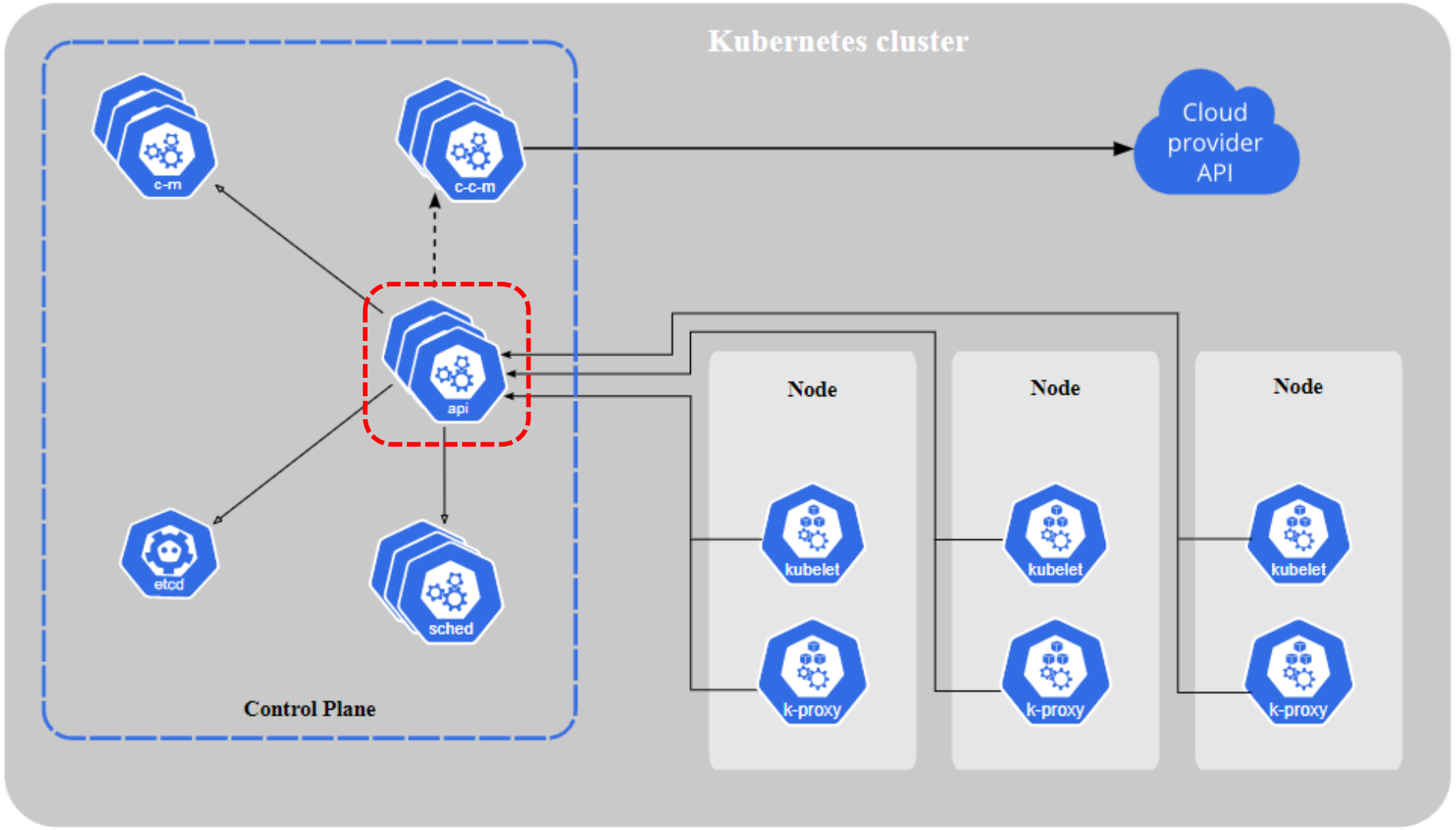
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

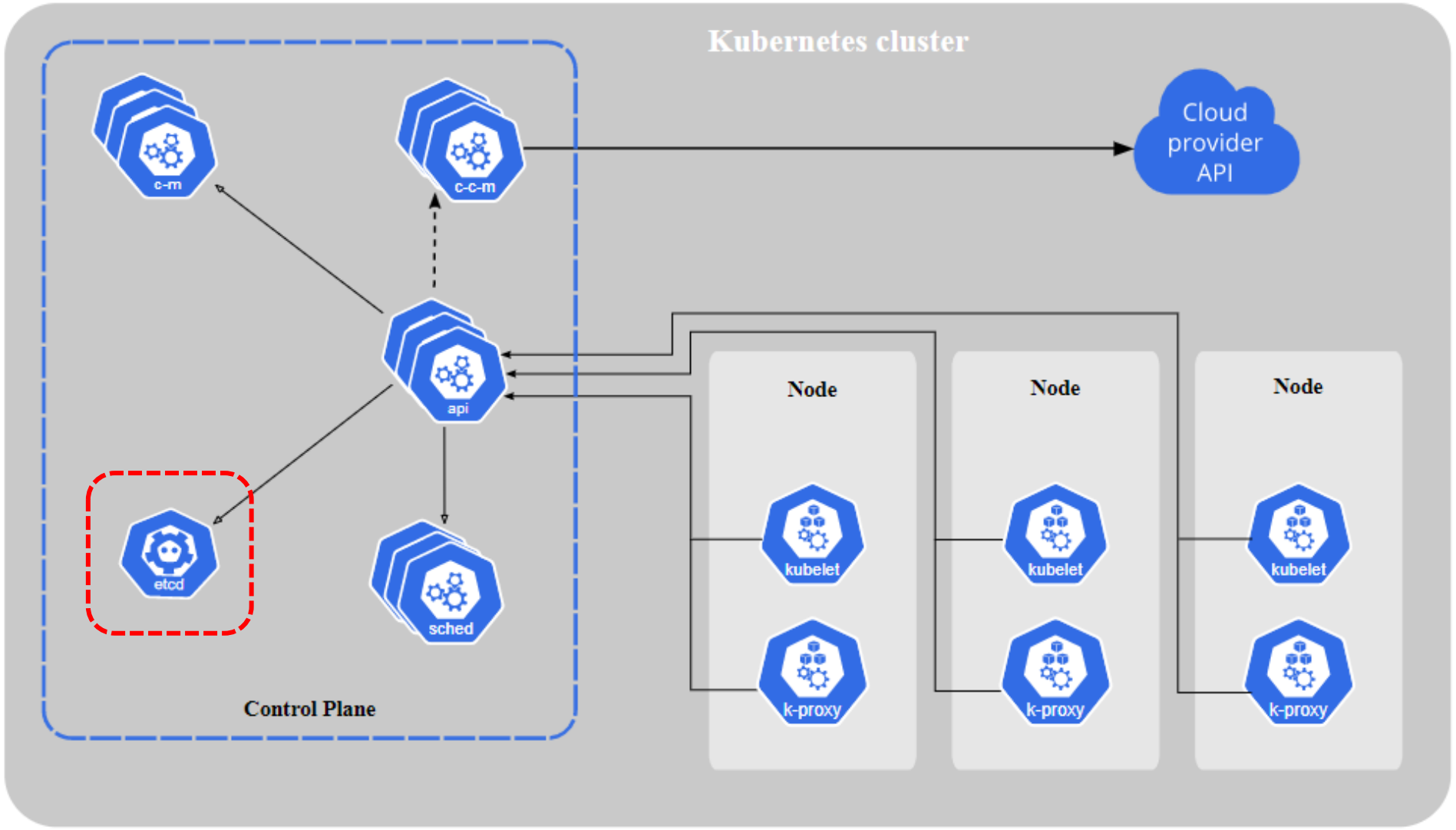
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

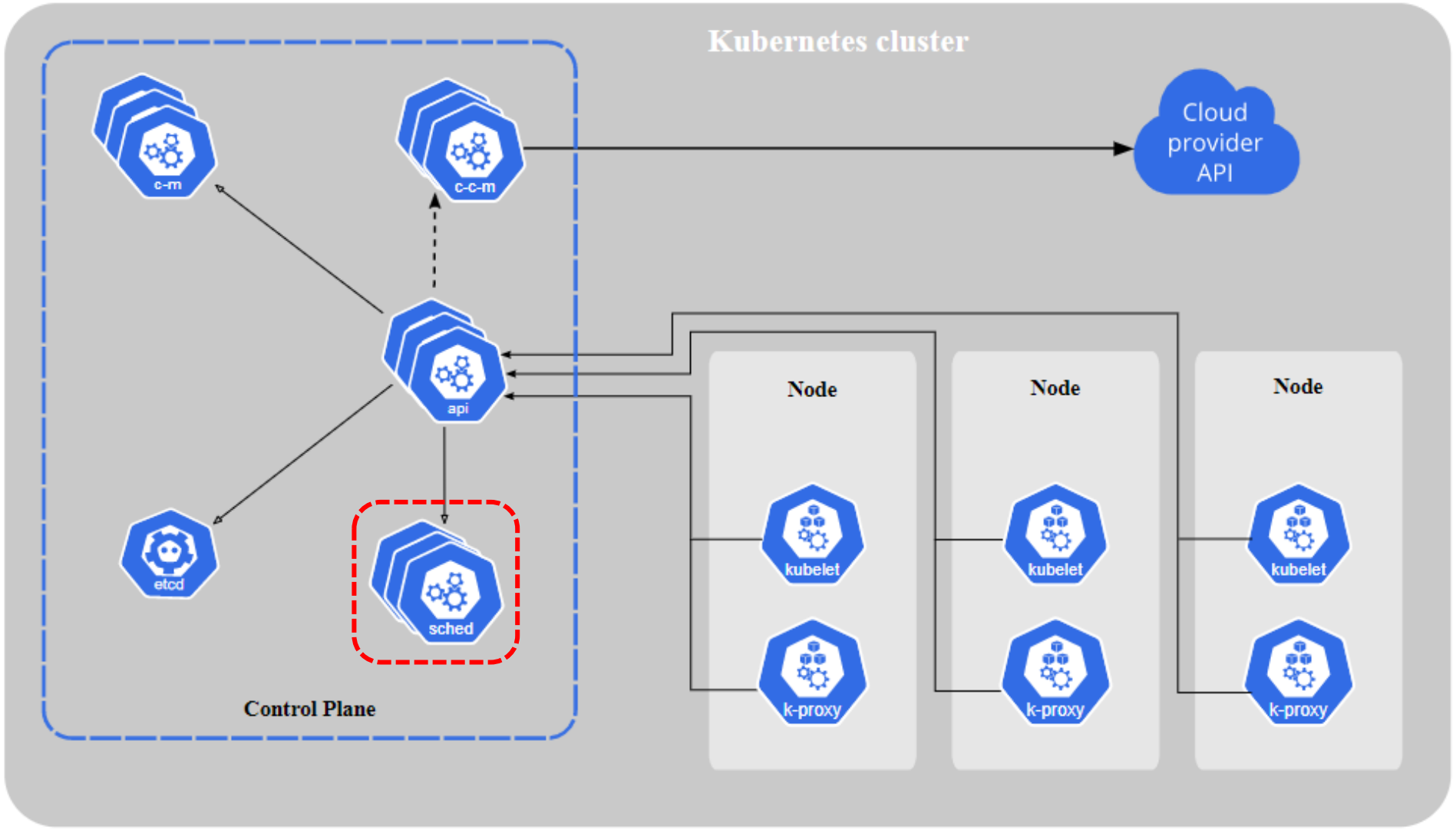
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

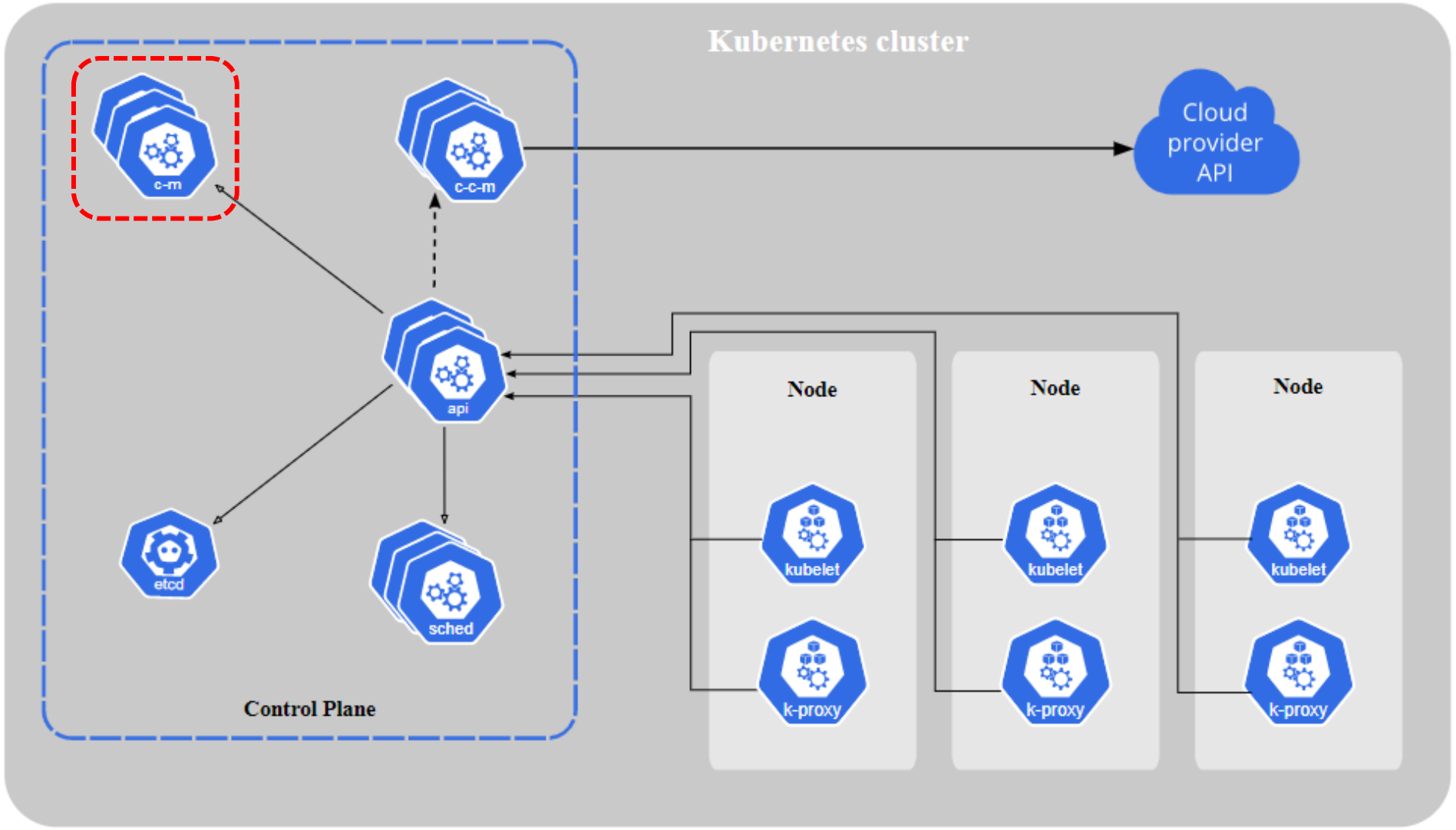
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

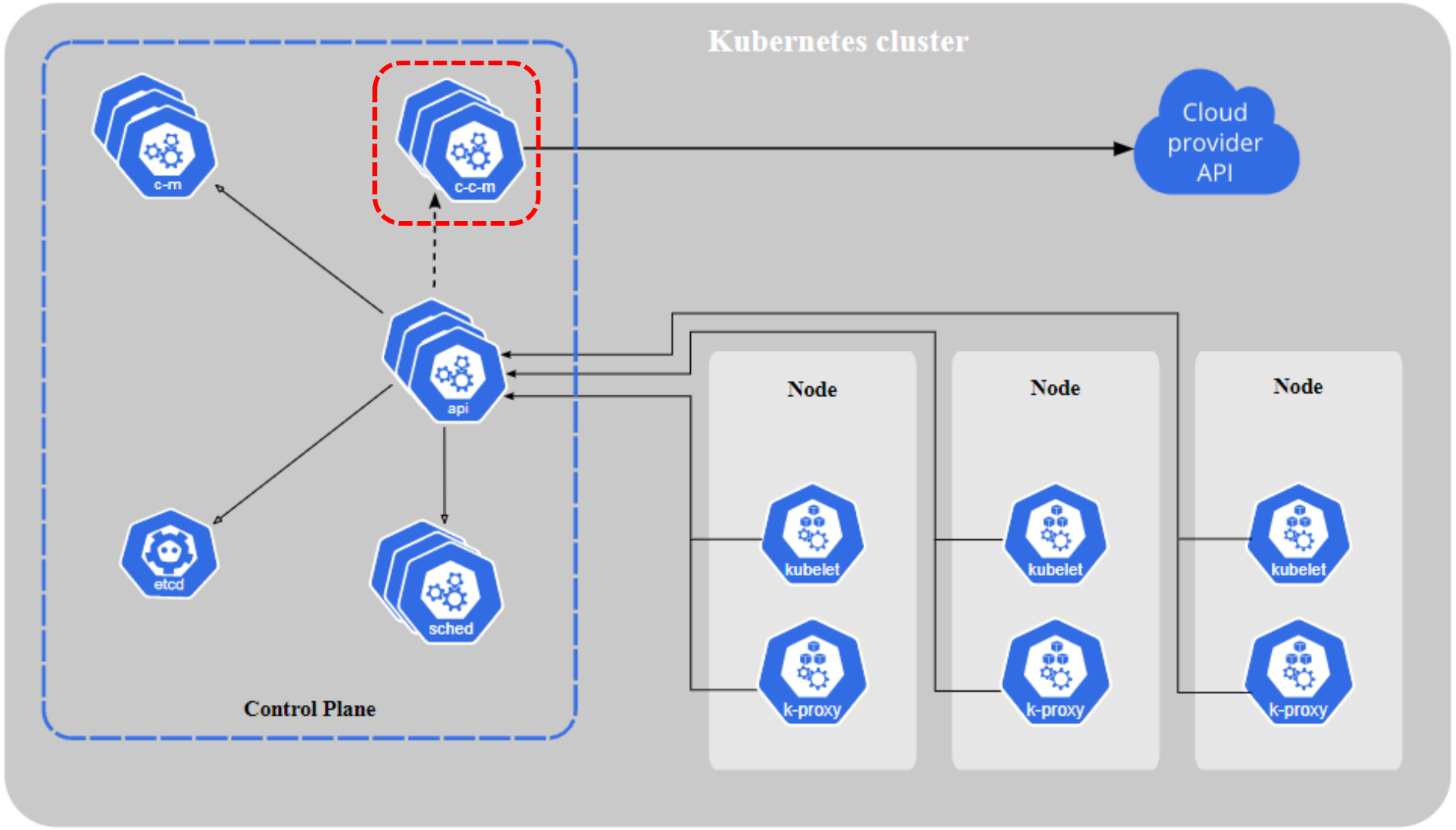
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

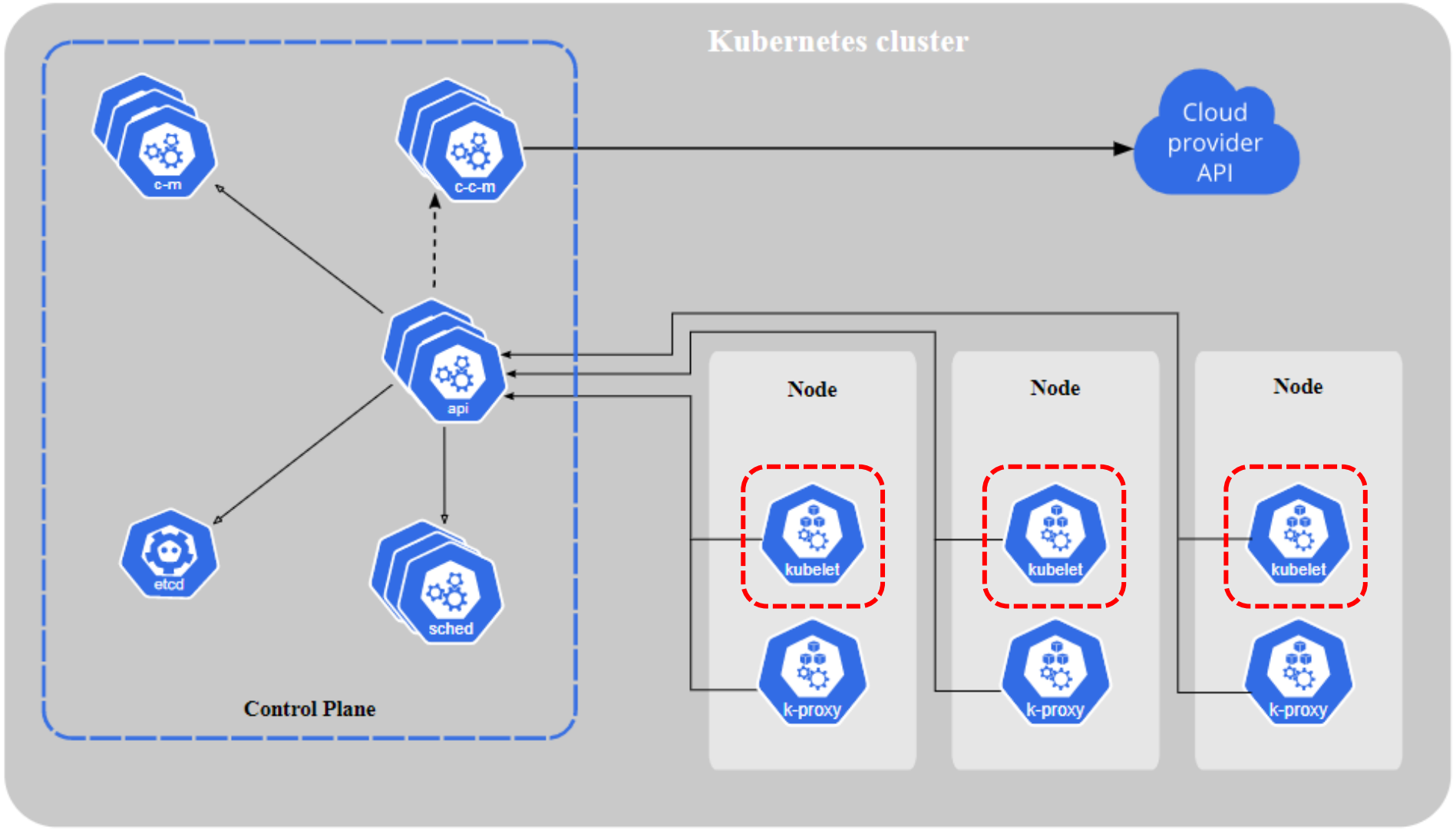
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

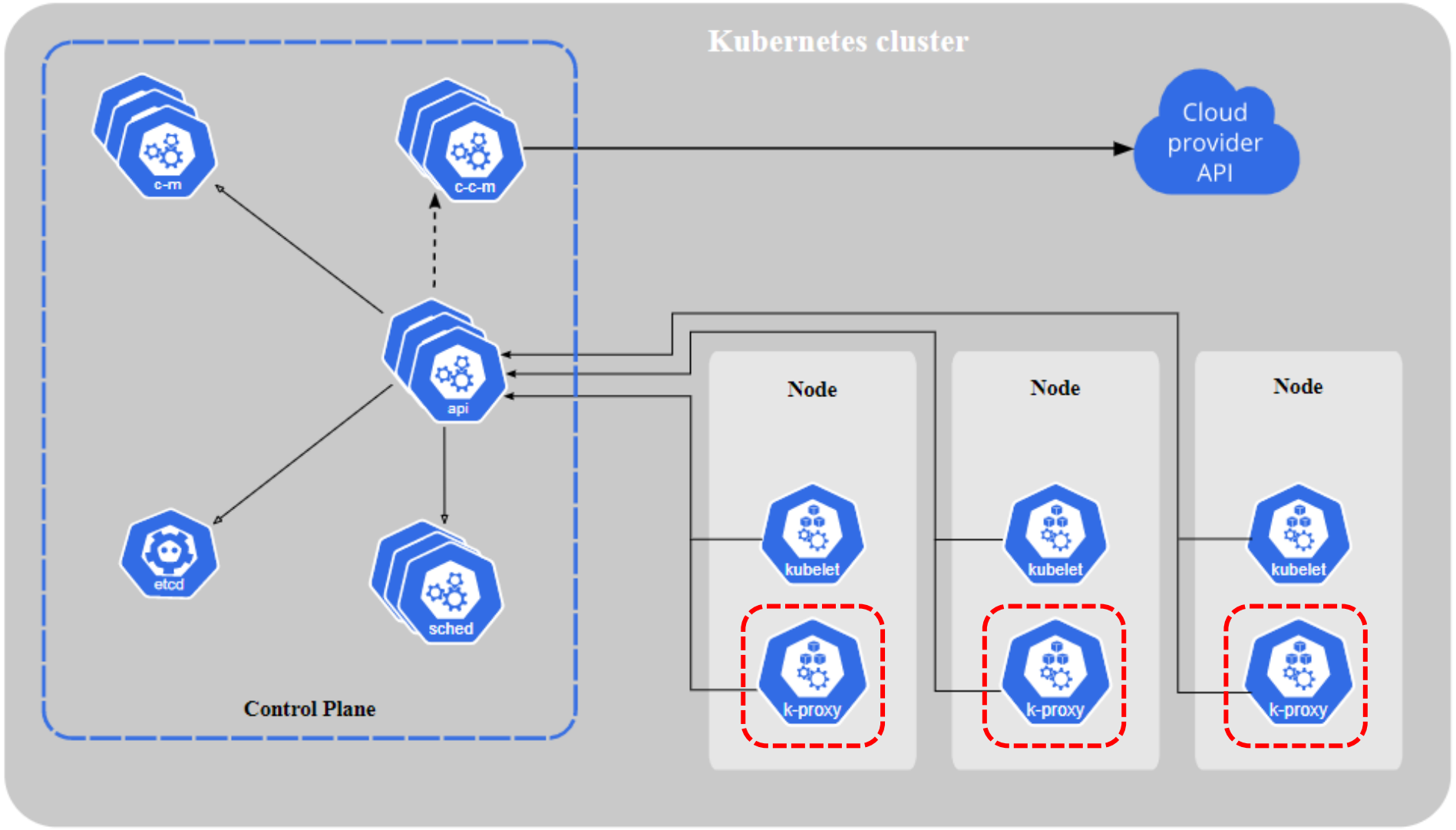
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

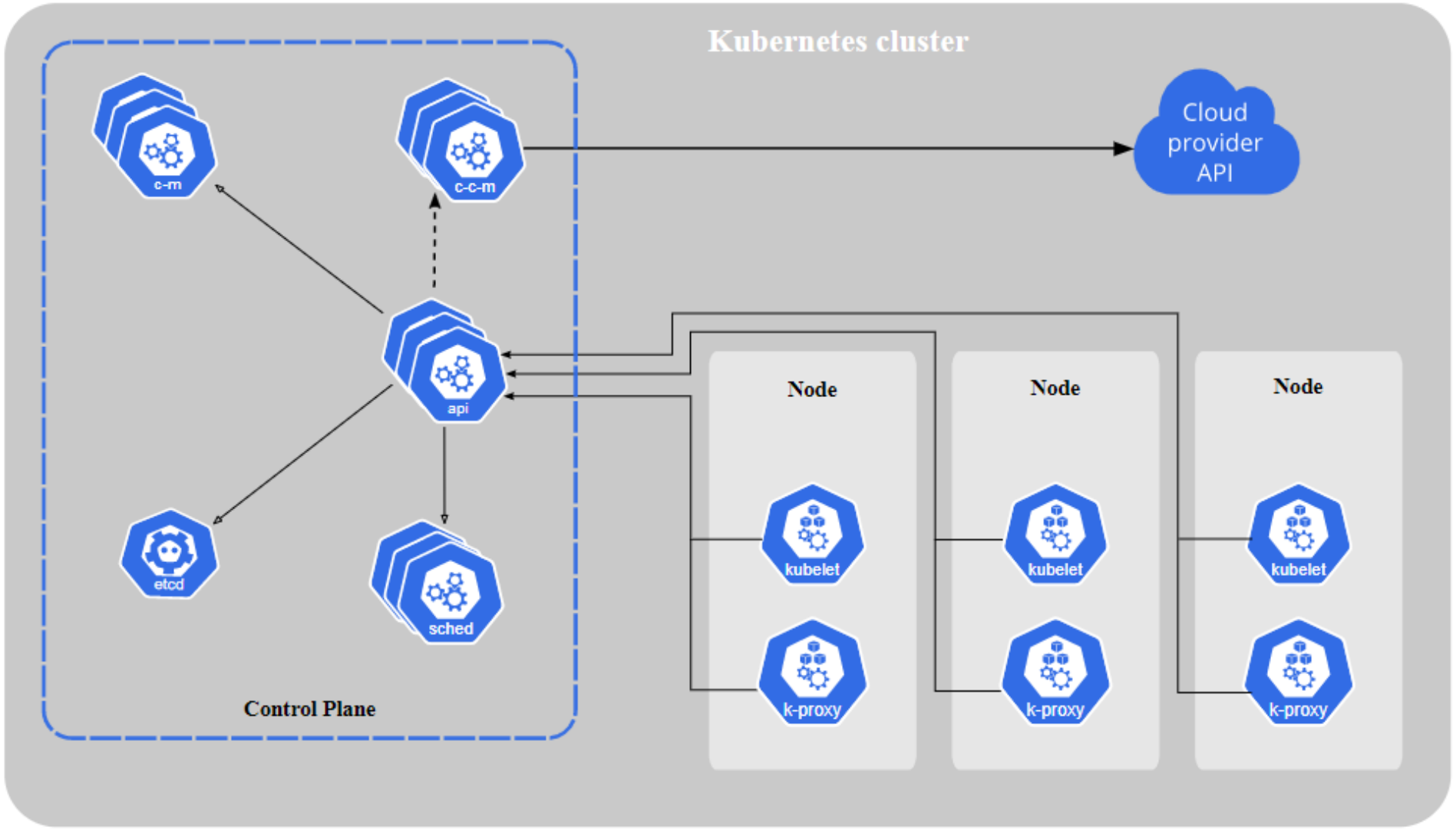
*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

*kube-proxy*

Scheduler

Control plane

Node

# Kubernetes 10000-foot view

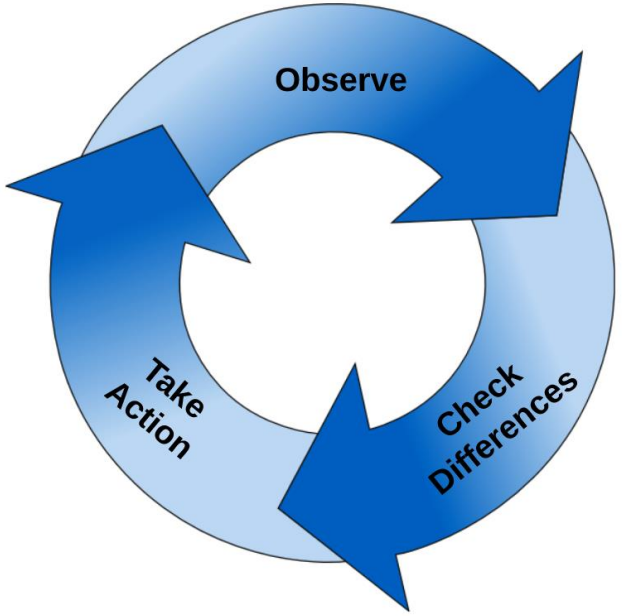
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 500m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 100Mi
```

Kubernetes is declarative: it stores desired state

# Kubernetes 10000-foot view

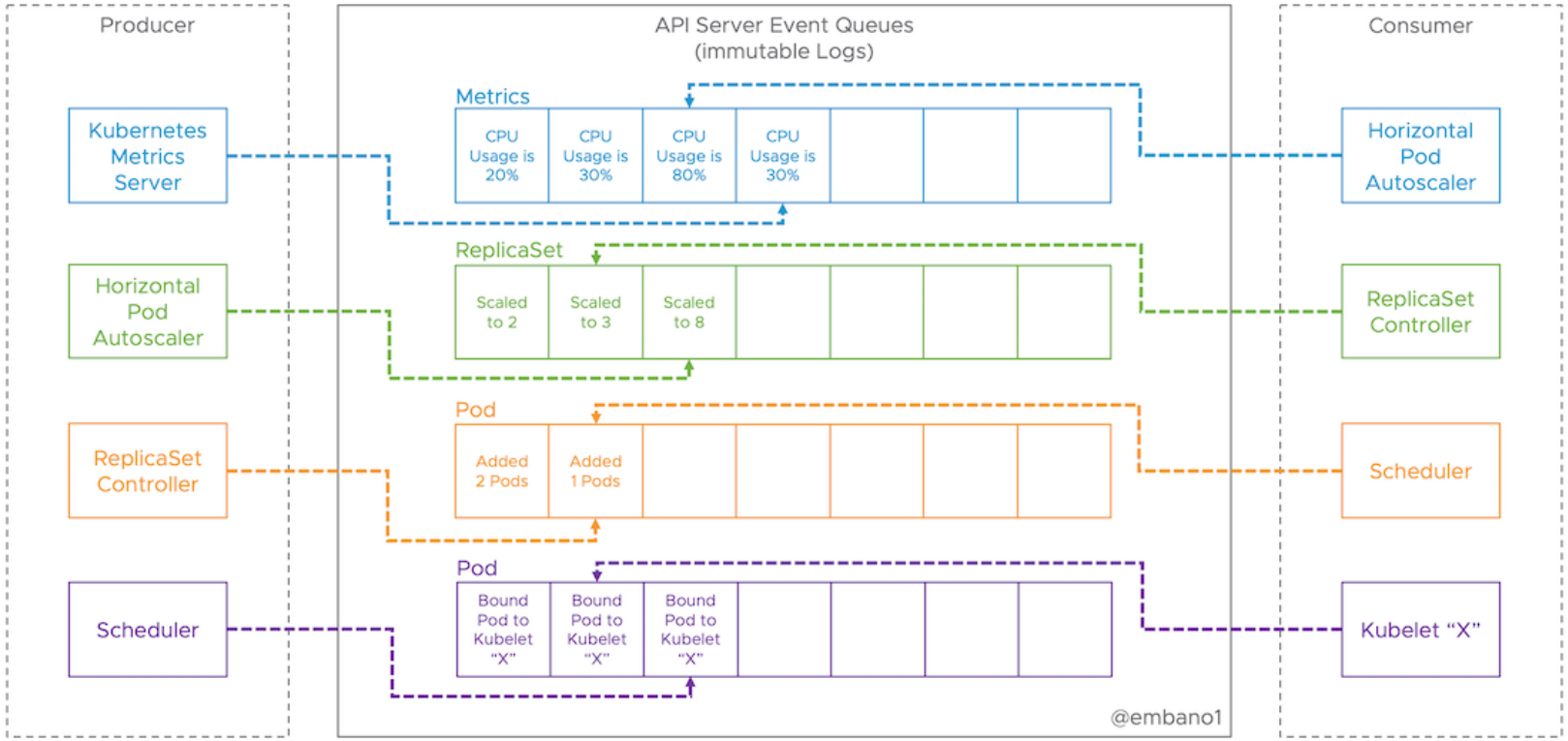
- Kubernetes controllers try to shape real objects to desired state
- Kubernetes heart is control loops

```
for {
  desired := getDesiredState()
  current := getCurrentState()
  makeChanges(desired, current)
}
```



# Kubernetes 10000-foot view

- There is no central orchestrator
- Each k8s component reacts on events
- All events are distributed via API service



# Kubernetes 10000-foot view

Labels and Annotations:

- Labels are key/value pairs; used to filter/search (both humans and apps)
  - "release" : "stable"
  - "environment" : "prod"
  - "team" : "data-science"
- Labels are not unique
- Via a label selector, the client/user can identify a set of objects
- Annotations – metadata, used by tools/libraries

# Kubernetes 10000-foot view

Namespace:

- Namespace – virtual cluster inside a physical one
- Kubernetes Objects could be namespaced or not
  - *kubectl api-resources --namespaced=true*
- Namespace enforces scope, policies, limits, roles/permissions
- Namespaces use-cases:
  - per environment
  - per team
  - per application

# Kubernetes 10000-foot view

To interact with k8s:

- API
- kubectl
  - Config
  - Commands/parameters: `kubectl {command} {object_type} {object_name} {params}`
  - Extendible via plugins (you can use plugin manager krew)
- k9s, lens, octant

# Kubernetes 10000-foot view

Run k8s:

- in cloud
- Locally
  - kind
  - minikube
  - kubeadm

# Kubernetes 10000-foot view

Demo

# Kubernetes introduction: additional resources

Alternative Kubernetes introduction:

- VMware course: videos
- Official docs: What is Kubernetes and Kubernetes Components

# Kubernetes introduction: additional resources

Deep dives:

- Events, the DNA of Kubernetes: article
- Deep-dive into api-server: part-1 (handle request), part-2 (store object), and part-3a (CRD)
- Kubernetes plugins and getting started with them article
- Official documentation: Working with Kubernetes Objects and Architecture