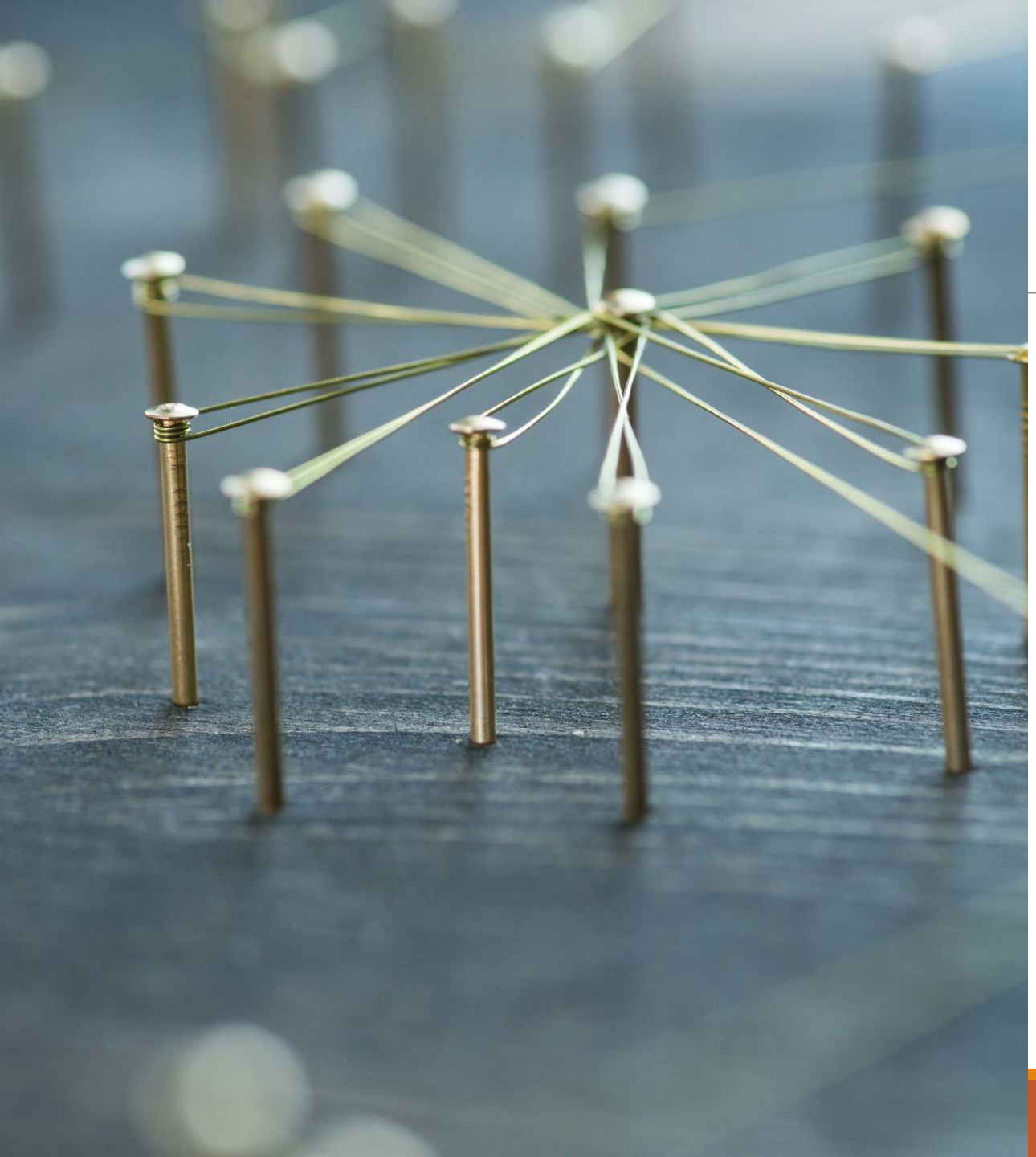
Kubernetes clusters

Kubernetes terminology

Nodes	• Individual VM running containerized applications
Pools	• Groups of nodes with identical configurations
Pods	• Single instance of an application • It's possible for a pod to contain multiple containers within the same node
Deployments	• One or more identical pods managed by Kubernetes
Manifests	• YAML file describing a deployment



Cluster components

Nodes

Control Plane components

Node components

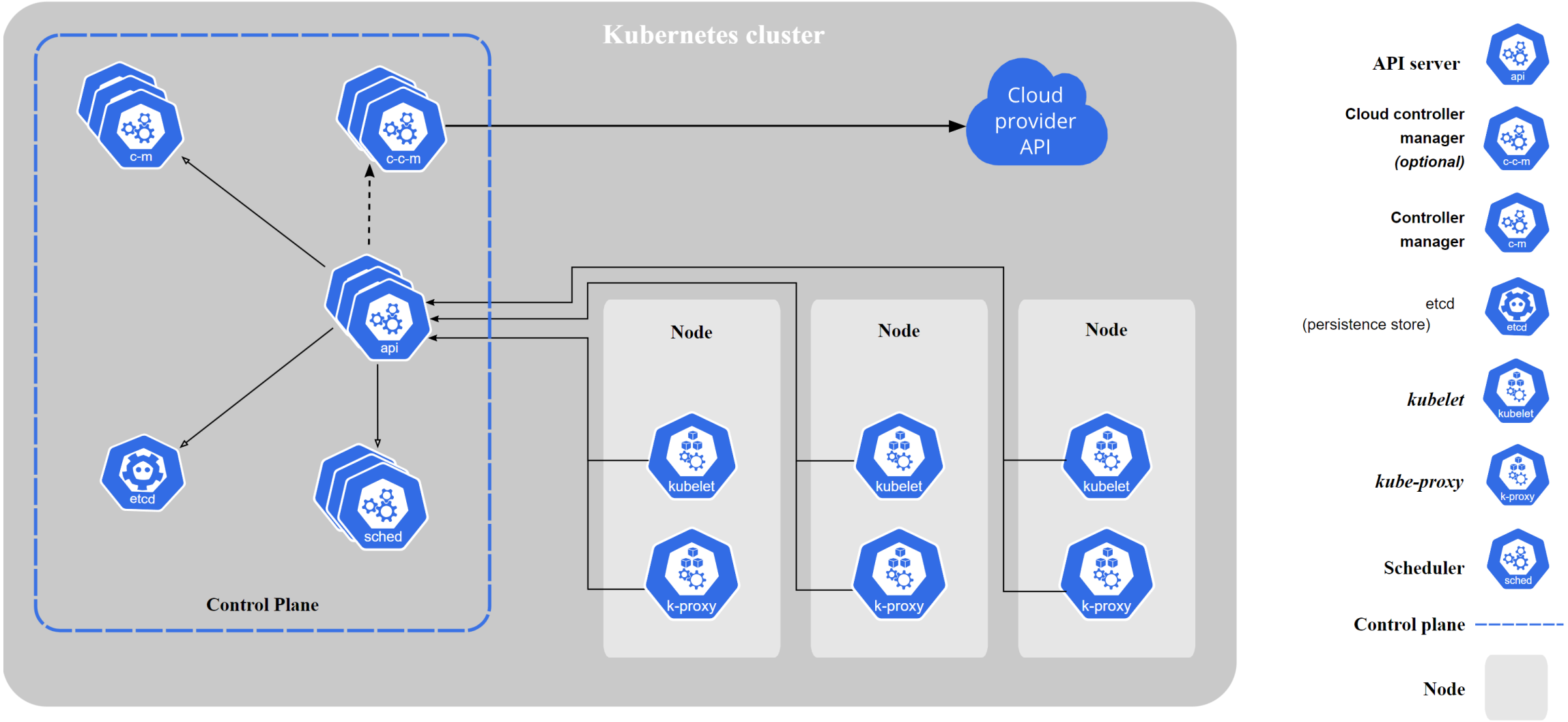
Types of Nodes in a Kubernetes Cluster

Master nodes

- Manage the cluster
- You must have an odd number, to form a consensus group
- In production, you need at least 3 (for failover)

Worker nodes

- Do the work for your orchestrated application
- In production, have as many as necessary to share the work (e.g. 1, 10, 100, 1000, etc.)


Kubernetes cluster
c-m
c-c-m
Cloud provider API
api
etcd
sched
Control Plane
Node
kubelet
k-proxy
Node
kubelet
k-proxy
Node
kubelet
k-proxy
API server
Cloud controller manager (optional)
Controller manager
etcd (persistence store)
kubelet
kube-proxy
Scheduler
Control plane
Node

Control Plane Components

kube-apiserver

- Front-end of the control plane

Etcd

- HA key-value store

kube-scheduler

- Looks for newly created pods and schedules them on nodes

controller-manager

- manages controller processes
  - Node controller
  - Job controller
  - ServiceAccount controller
  - EndpointSlice controller (links Services and Pods)

Node Components

Container runtime

- Ability to run containers
- Can be: Docker, containerd, CRI-O, ..,
  or another implementation of Kubernetes Container Runtime Interface

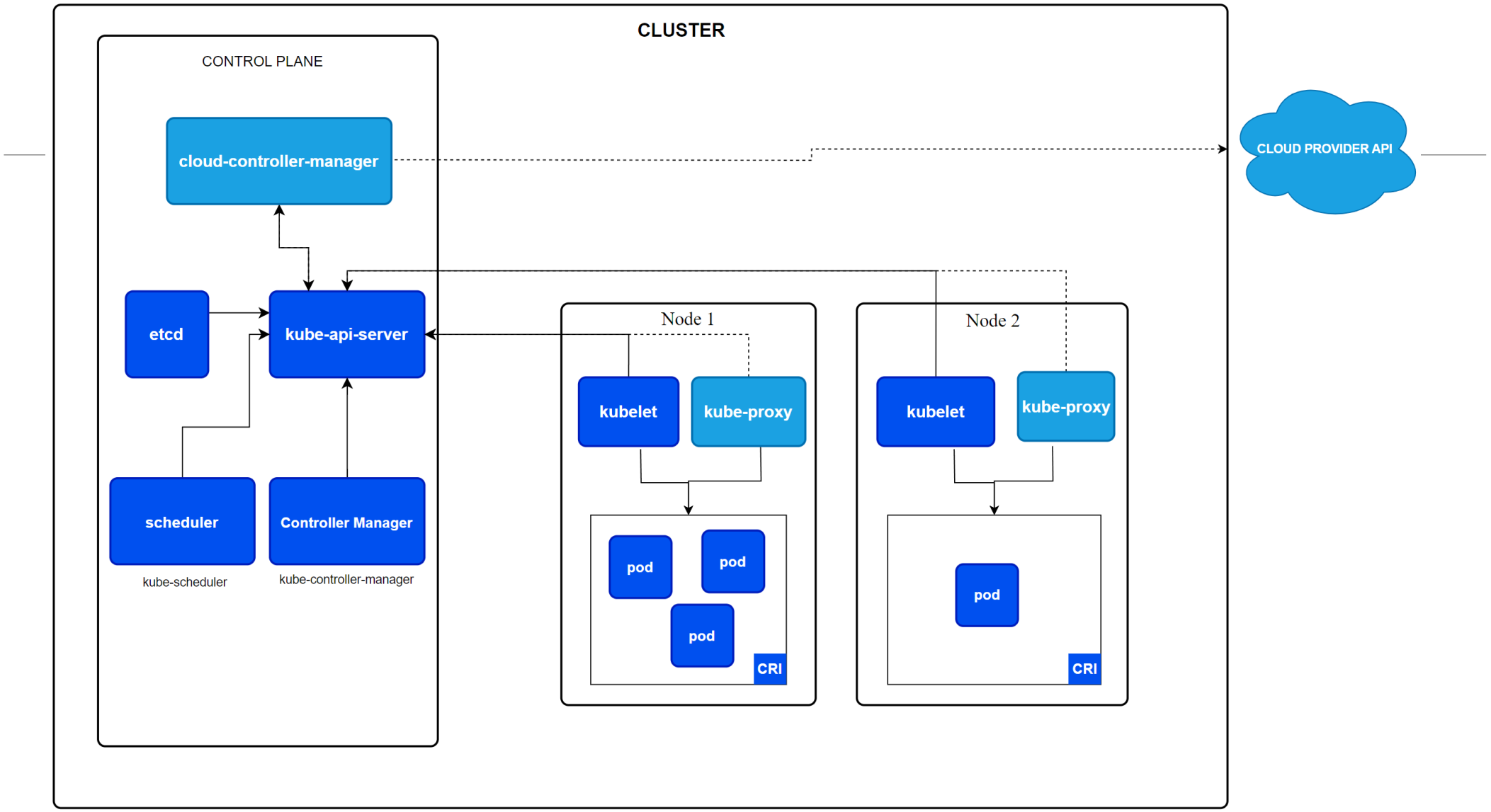
Kubelet

- Agent runs on all nodes
- Make sure Pods are running
  (and healthy)

Kube-proxy

- Network proxy on all nodes
- Maintains rules

And more …
Addons, DNS, Resource Monitoring, Logging, … to name a few


CLUSTER
CONTROL PLANE
cloud-controller-manager
etcd
kube-api-server
scheduler
kube-scheduler
Controller Manager
kube-controller-manager
Node 1
kubelet
kube-proxy
pod
pod
pod
CRI
Node 2
kubelet
kube-proxy
pod
CRI
CLOUD PROVIDER API

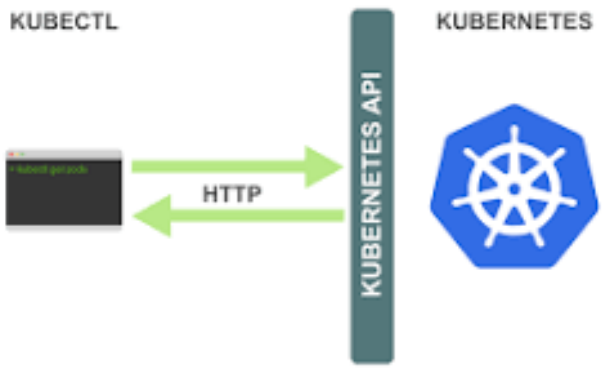
Kubectl

Kubectl is a command line tool used to run commands against Kubernetes clusters

It does this by authenticating with the Master Node of your cluster and making API calls to do a variety of management actions.

You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs

For a complete list of features: https://kubernetes.io/docs/reference/kubectl/



Kubectl

```
kubectl [command] [TYPE] [NAME] [flags]
```

Command: create, get, describe, delete

Type: resource type. Pods, services, configmaps, namespaces

Name: name of the resource (case sensitive)

Inspecting your cluster

kubectl cluster-info

```
PS C:\Users\RonaldHarmsen> kubectl cluster-info
Kubernetes master is running at https://172.17.191.19:16443
Heapster is running at https://172.17.191.19:16443/api/v1/namespaces/kube-system/services/heapster/proxy
CoreDNS is running at https://172.17.191.19:16443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Grafana is running at https://172.17.191.19:16443/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
InfluxDB is running at https://172.17.191.19:16443/api/v1/namespaces/kube-system/services/monitoring-influxdb:http/proxy


To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
PS C:\Users\RonaldHarmsen> kubectl get componentstatuses
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health":"true"}
```

ed v19+)

Inspecting your cluster

```
kubectl get nodes
```

```
PS C:\Users\RonaldHarmsen> kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
microk8s-vm   Ready    <none>   7d3h  v1.15.3
```

```
kubectl version
```

```
PS C:\Users\RonaldHarmsen> kubectl version
Client Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.6", GitCommit:"96fac5cd13a5dc064f7d9f4f23030a6aefa
ce6cc", GitTreeState:"clean", BuildDate:"2019-08-19T11:13:49Z", GoVersion:"go1.12.9", Compiler:"gc", Platform:"windows/a
md64"}
Server Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.3", GitCommit:"2d3c76f9091b6bec110a5e63777c332469e
0cba2", GitTreeState:"clean", BuildDate:"2019-08-19T11:05:50Z", GoVersion:"go1.12.9", Compiler:"gc", Platform:"linux/amd
64"}
```

Getting details

kubectl describe node *microk8s-vm*

```
PS C:\Users\RonaldHarmsen> kubectl describe node microk8s-vm
Name:               microk8s-vm
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=microk8s-vm
                    kubernetes.io/os=linux
                    microk8s.io/cluster=true
Annotations:        node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Sun, 08 Sep 2019 10:59:58 +0200
Taints:             <none>
Unschedulable:      false
```

+ a lot more information about resources(consumption)

```
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason
    Message
  ----             ------  -----------------                 ------------------                ------
    -------
  MemoryPressure   False   Sun, 15 Sep 2019 14:17:45 +0200   Sun, 15 Sep 2019 13:10:41 +0200   KubeletHasSufficientMemor
y   kubelet has sufficient memory available
  DiskPressure     False   Sun, 15 Sep 2019 14:17:45 +0200   Sun, 15 Sep 2019 13:10:41 +0200   KubeletHasNoDiskPressure
    kubelet has no disk pressure
  PIDPressure      False   Sun, 15 Sep 2019 14:17:45 +0200   Sun, 15 Sep 2019 13:10:41 +0200   KubeletHasSufficientPID
    kubelet has sufficient PID available
  Ready            True    Sun, 15 Sep 2019 14:17:45 +0200   Sun, 15 Sep 2019 13:10:41 +0200   KubeletReady
    kubelet is posting ready status. AppArmor enabled
```

```
Addresses:
  InternalIP:  172.17.191.19
  Hostname:    microk8s-vm
Capacity:
 cpu:                1
 ephemeral-storage:  40470732Ki
 hugepages-1Gi:      0
 hugepages-2Mi:      0
 memory:             6803112Ki
 pods:               110
Allocatable:
 cpu:                1
 ephemeral-storage:  39422156Ki
 hugepages-1Gi:      0
 hugepages-2Mi:      0
 memory:             6700712Ki
 pods:               110
```

```
System Info:
 Machine ID:                 c7d437b97f664e21918c8acfa20e607a
 System UUID:                25A6285A-E4A0-CD44-93F5-940F683E131E
 Boot ID:                    70db53da-9f78-457b-a5de-f1a70282a22e
 Kernel Version:             4.15.0-62-generic
 OS Image:                   Ubuntu 18.04.3 LTS
 Operating System:           linux
 Architecture:               amd64
 Container Runtime Version:  containerd://1.2.5
 Kubelet Version:            v1.15.3
 Kube-Proxy Version:         v1.15.3
Non-terminated Pods:         (9 in total)
```

```
  Namespace                   Name                                        CPU Requests  CPU Limits  Memory Request
s  Memory Limits  AGE
  ---------                   ----                                        ------------  ----------  --------------
-  -------------  ---
  container-registry          registry-6c99589dc-25zmc                    0 (0%)        0 (0%)      0 (0%)           0 (0%)         16h
  default                     default-http-backend-5d5ff5d4f5-qcnrm       10m (1%)      10m (1%)    20Mi (0%)        20Mi (0%)      17h
  default                     nginx-ingress-microk8s-controller-99ftc     0 (0%)        0 (0%)      0 (0%)           0 (0%)         17h
  default                     webapp-6cdccfc747-8n7cq                     0 (0%)        0 (0%)      0 (0%)           0 (0%)         16h
  kube-system                 coredns-f7867546d-v4bjs                     100m (10%)    0 (0%)      70Mi (1%)        170Mi (2%)     3d15h
  kube-system                 heapster-v1.5.2-844b564688-mcjgd            288m (28%)    288m (28%)  596176Ki (8%)    596176Ki (8%)  3d15h
  kube-system                 hostpath-provisioner-65cfd8595b-hjqqk       0 (0%)        0 (0%)      0 (0%)           0 (0%)         16h
  kube-system                 kubernetes-dashboard-7d75c474bb-x4fgd       0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d15h
  kube-system                 monitoring-influxdb-grafana-v4-6b6954958c-mlv7t  200m (20%)    200m (20%)  600Mi (9%)       600Mi (9%)     3d15h
```

```
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests         Limits
  --------           --------         ------
  cpu                598m (59%)       498m (49%)
  memory             1302736Ki (19%)  1405136Ki (20%)
  ephemeral-storage  0 (0%)           0 (0%)
Events:              <none>
```

Kubernetes Proxy

```
$ kubectl get daemonSets --namespace=kube-system kube-proxy
NAME          DESIRED   CURRENT   READY     NODE-SELECTOR   AGE
kube-proxy    4         4         4         <none>          45d
```

Kubernetes DNS

```
$ kubectl get deployments --namespace=kube-system kube-dns
NAME        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kube-dns    1         1         1            1           45d
```

```
$ kubectl get services --namespace=kube-system kube-dns
NAME        CLUSTER-IP    EXTERNAL-IP   PORT(S)         AGE
kube-dns    10.96.0.10    <none>        53/UDP,53/TCP   45d
```

Kubernetes UI | Dashboard

```
$ kubectl get deployments --namespace=kube-system kubernetes-dashboard
NAME                   DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-dashboard   1         1         1            1           45d
```

```
$ kubectl get services --namespace=kube-system kubernetes-dashboard
NAME                   CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes-dashboard   10.99.104.174   <nodes>       80:32551/TCP   45d
```

Provisioning a Kubernetes Cluster

It's possible to provision a Kubernetes cluster from scratch

- You provide your own physical servers or VMs
- This is hard!

A better option is to use a tool to set up a single-node Kubernetes cluster, such as:

- Minikube
- Docker for Desktop support for Kubernetes
- Or leverage something like Microk8s or K3s as production ready lighweight options

An alternative option (in production) is to leverage Kubernetes support from cloud providers, such as:

- Azure Kubernetes Service - AKS
- Amazon Elastic Container Service for Kubernetes - Amazon EKS
- Google Kubernetes Engine – GKE
- Digital Ocean hosted Kubernetes
- …

Setting up your development environment

Development Environment options

Kubernetes with Docker for Desktop (Windows/Mac)

MiniKube

MicroK8s.io (complete single node system)

K3S

…

Docker for Desktop

Out-of-the-box containerization

Contains Docker engine

Toolkit to build, run and publish containerized applications

Option to enable Kubernetes

Easiest starting point

https://www.docker.com/products/docker-desktop/

Minikube

Supports the latest Kubernetes release (+6 previous minor versions)

Cross-platform (Linux, macOS, Windows)

Deploy as a VM, a container, or on bare-metal

Multiple container runtimes (CRI-O, containerd, docker)

Advanced features such as LoadBalancer, filesystem mounts, FeatureGates, and network policy

Addons for easily installed Kubernetes applications

Supports common CI environments

https://minikube.sigs.k8s.io/docs/start/

Microk8s

The simplest way to get K8s anywhere

Zero-ops, pure-upstream, HA Kubernetes from developer workstations to production.

Minimal, CNCF-certified distribution

Maintained by Canonical (also Ubuntu)

Addons for easily installed Kubernetes applications

https://microk8s.io/

MicroK8s on desktop

You need a Linux kernel, so VM is the option here

Simple setup:

Get multipass:
https://github.com/CanonicalLtd/multipass/releases/

```
multipass launch --name microk8s-vm --mem 4G --disk 40G

multipass exec microk8s-vm -- sudo snap install microk8s --
classic

multipass exec microk8s-vm -- sudo iptables -P FORWARD ACCEPT
```

MicroK8s - Connect to cluster in VM

```
multipass exec microk8s-vm -- /snap/bin/microk8s.config >
kubeconfig


kubectl --kubeconfig=kubeconfig get all --all-namespaces


//set the currentconfig to the file just generated:

$env:KUBECONFIG=("kubeconfig")
KUBECONFIG=kubeconfig
```

Alternative (stdout flush not working)

```
multipass shell microk8s /snap/bin/microk8s.config > kubeconfig

exit


multipass transfer microk8s-vm:/home/multipass/kubeconfig
kubeconfig
```

Working with multiple environments

You will probably have multiple Kubernetes environments, i.e.

- Docker Desktop with Kubernetes
- MicroK8s / minikube for development
- Azure Kubernetes Service
- Google Kubernetes Engine
- etc.

Multiple credentials and environment settings are needed

Kubectl has support for that

Connecting to cloud clusters

This will combine the kubeconfig directly into ./kube/config:

```
// Get credentials for Azure Kubernetes Service
az aks get-credentials --resource-group mygroup --name mycluster

// Get credentials for Google Kubernetes Engine
gcloud container clusters get-credentials mycluster

// Get credentials for Digital Ocean Kubernetes Cluster
doctl kubernetes cluster kubeconfig save mycluster
```

Switching configured contexts

```
kubectl config get-contexts
```

```
PS C:\Users\RonaldHarmsen> kubectl config get-contexts
CURRENT   NAME                 CLUSTER            AUTHINFO         NAMESPACE
          docker-desktop       docker-desktop     docker-desktop
          docker-for-desktop   docker-desktop     docker-desktop
*         microk8s             microk8s-cluster   admin
```

```
kubectl config current-context
```

```
kubectl config use-context docker-for-desktop
```

Combine

```
# Set multiple config files. cd ~ first
KUBECONFIG=.kube/config:kubeconfig.file

$env:KUBECONFIG=(".kube\config;kubeconfig")

# Get configuration files combined into one
kubectl config view –flatten > combinedconfig
```

Common commands

Namespaces:

```
--namespace=…
```

```
kubectl config set-context my-context --namespace=mystuff
```

```
kubectl config use-context my-context
```

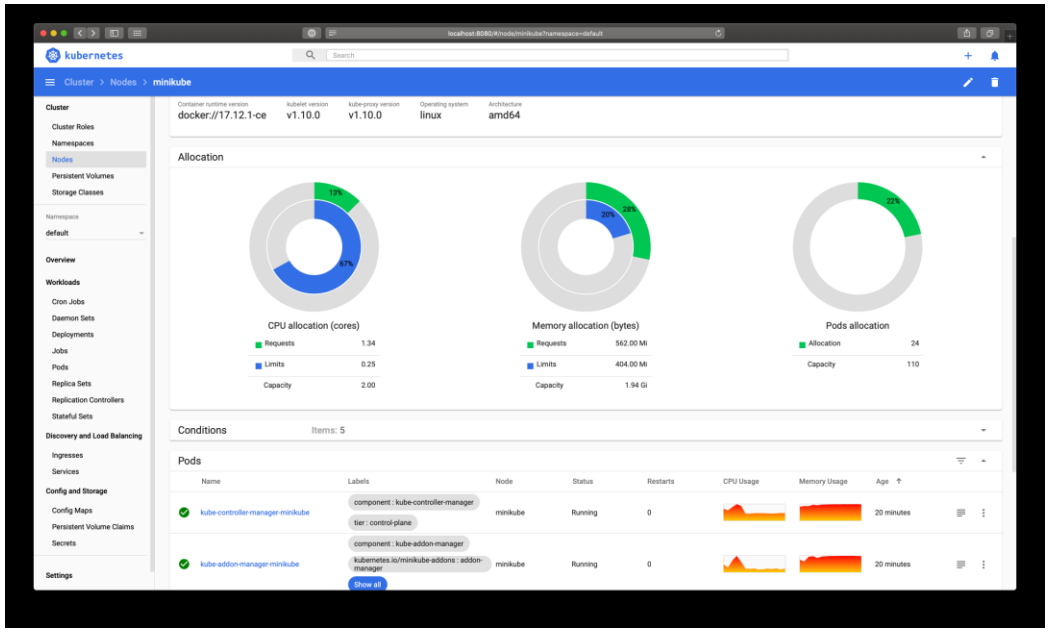
```
kubectl config get-contexts
```

Dashboards

Kubernetes Dashboard

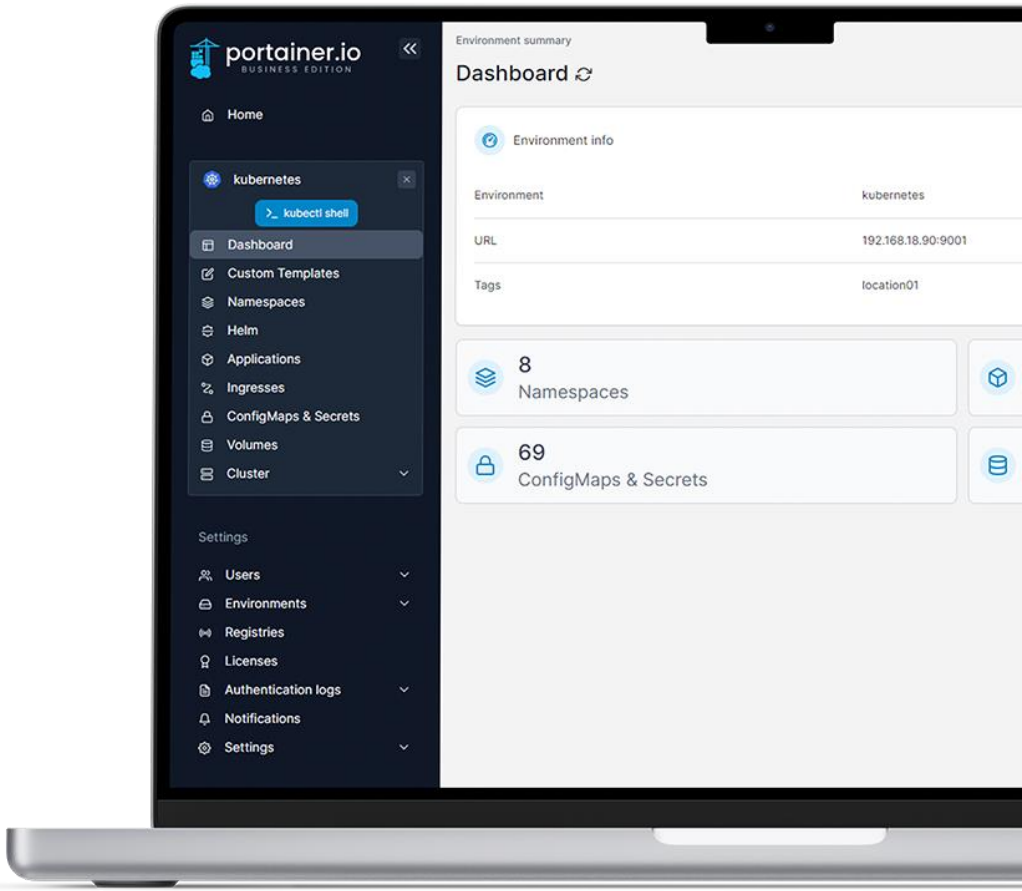
Kubernetes Dashboard is a general purpose, web-based UI for Kubernetes clusters.

It allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself.



Portainer

Graphical Kubernets management



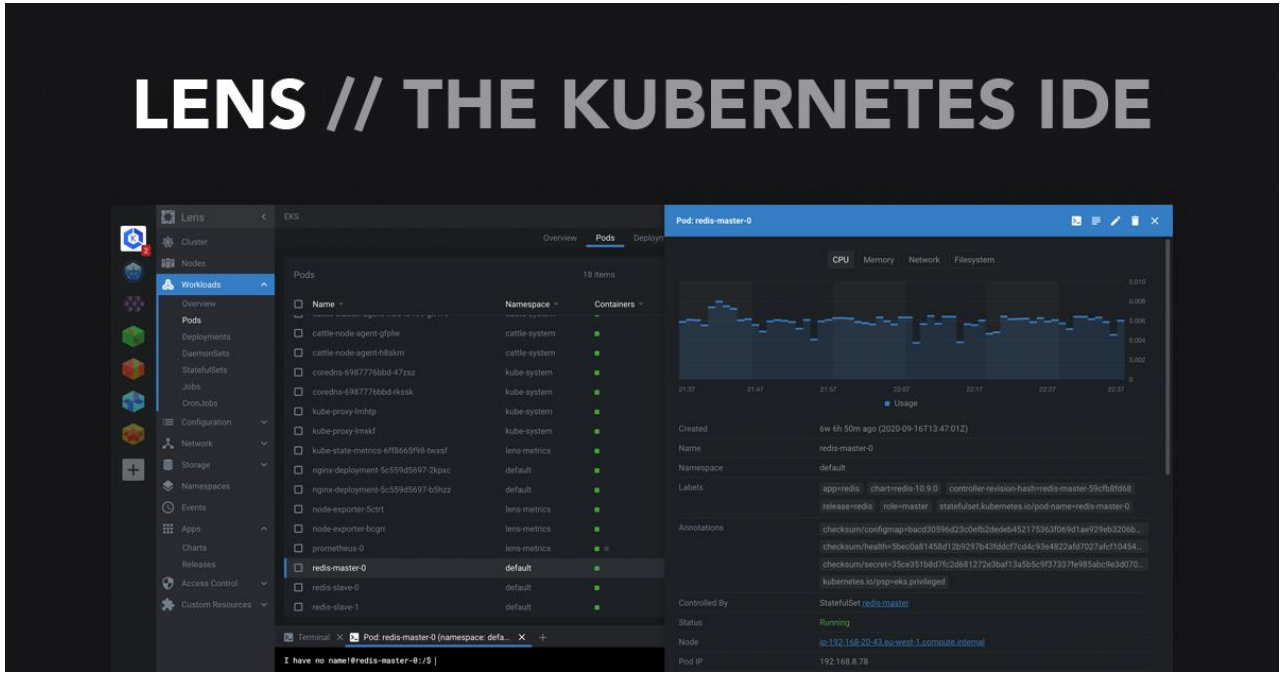
https://portainer.io

Lens

GUI for managing Kubernetes clusters.

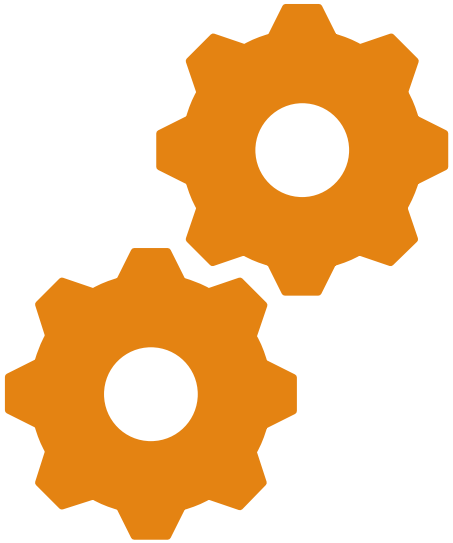
Allows to connect / manage multiple clusters simultaneously

Commercial application
Free for personal / development use option



https://k8slens.dev/

Wrap up



Kubernetes can be setup in different ways:

Standalone (hard)

Prepackaged

Managed by service provider (cloud)

https://github.com/ronaldharmsen/k8s-labs

Appendix: connecting to Microk8s dashboard

MicroK8s Dashboard

Setup dns and dashboard

```
multipass exec microk8s-vm -- sudo /snap/bin/microk8s.enable dns
dashboard
```

Run proxy to get access

```
/snap/bin/microk8s.kubectl proxy --address='0.0.0.0' --accept-
hosts='.*'
Starting to serve on [::]:8001
```

MicroK8s Dashboard

```
microk8s.kubectl edit deployment/kubernetes-dashboard --namespace=kube-system


spec:
      containers:
      - args:
        - --auto-generate-certificates
        - --enable-skip-login
        image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
```

http://localhost:8001/api/v1/namespaces/kube-system/services/**monitoring-grafana**/proxy/?orgId=1

http://localhost:8001/api/v1/namespaces/kube-system/services/**https:kubernetes-dashboard:/**proxy/#!/login

multipass exec microk8s-vm -- sudo /snap/bin/microk8s.kubectl expose deployment.apps/monitoring-influxdb-grafana-v4 -n kube-system --type=NodePort

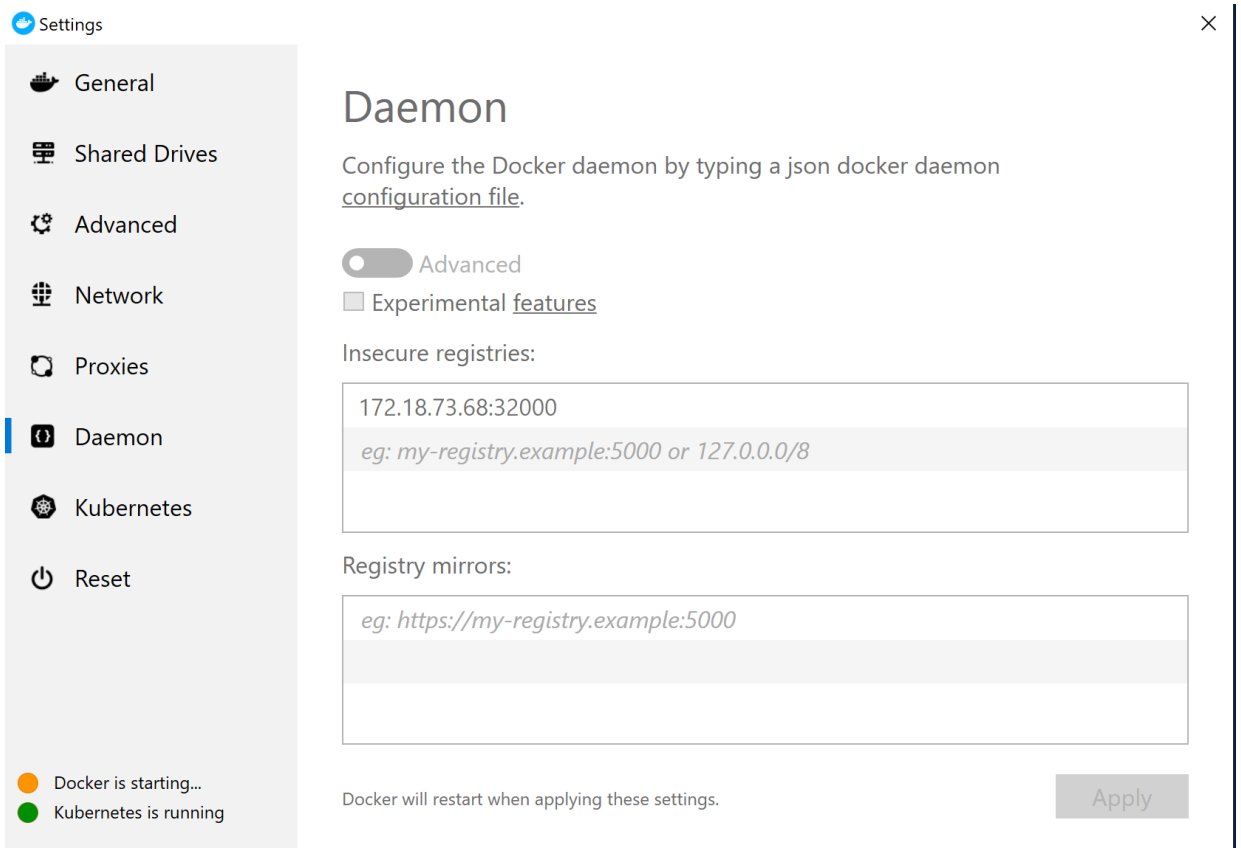
/snap/bin/microk8s.kubectl get services -n kube-system

```
microk8s.kubectl expose -n kube-system
deployment.apps/kubernetes-dashboard --type NodePort --name ds-np
```

microk8s.enable registry

```
multipass@microk8s-vm:~$ microk8s.enable registry
Enabling the private registry
Enabling default storage class
deployment.extensions/hostpath-provisioner created
storageclass.storage.k8s.io/microk8s-hostpath created
serviceaccount/microk8s-hostpath created
clusterrole.rbac.authorization.k8s.io/microk8s-hostpath created
clusterrolebinding.rbac.authorization.k8s.io/microk8s-hostpath created
Storage will be available soon
Applying registry manifest
namespace/container-registry created
persistentvolumeclaim/registry-claim created
deployment.extensions/registry created
service/registry created
The registry is enabled
```

Register VM IP & Port 32000 in Docker Desktop



```
docker tag k8s-labs/simple-webapp 172.18.73.68:32000/simple-webapp

docker push 172.18.73.68:32000/simple-webapp
```