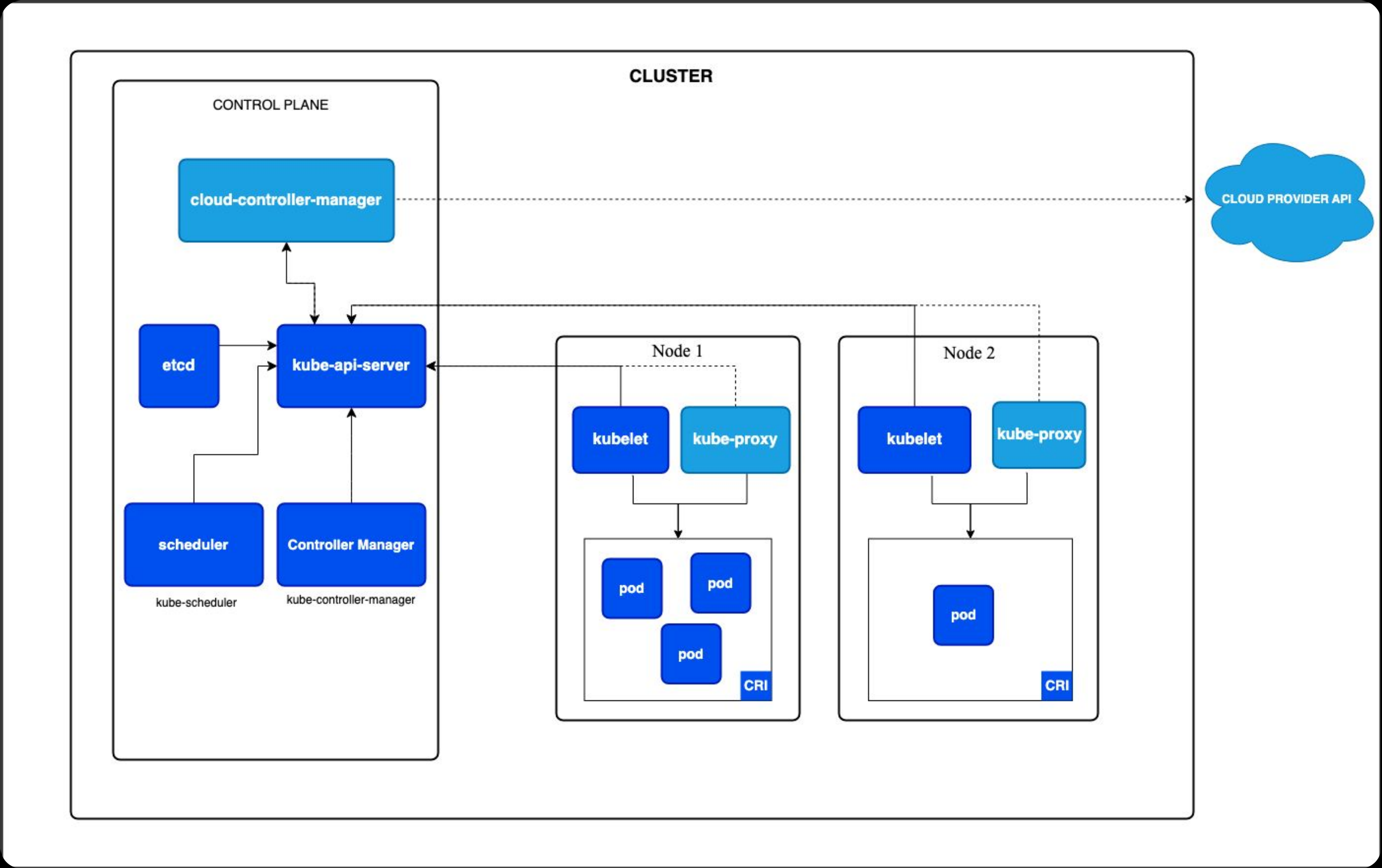
Kubernetes 101

b’nerd GmbH

hello@bnerd.com
bnerd.com/de

Sartoriusstraße 22, 20257 Hamburg



https://kubernetes.io/docs/concepts/architecture/

About Nodes

Nodes are basically your servers and can be **virtual** or **physical machines**.

We differentiate between Master and Worker Nodes.

Master vs. Worker Nodes

Master Node

monitors and controls the processes in the cluster, e.g. scheduling and distributing the workloads to the individual nodes

uses a number of components grouped together under the name **control plane**

Worker Nodes

host your application containers, grouped as pods

kubelet is the primary "node agent" communicating with the control plane

kube-proxy is a network proxy and ensures connectivity between services and pods

Control Plane components

The **kube-apiserver** (API Server) is used to interact with the Kubernetes API and your clusters. It’s stateless and scales horizontally.

etcd is a consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

Control Plane components

The **kube-controller-manager** (Controller Manager) is responsible for other controller managers that continuously monitor the state of the Kubernetes cluster by making requests against the API. If the actual state of the cluster differs from the desired state, the responsible controller takes appropriate measures.

The **kube-scheduler** (Scheduler) assigns new Pods onto the Nodes in your cluster. It establishes which nodes can fulfill the Pod’s requirements, then selects the most optimal placement to maximize performance and reliability.

Relevant Kubernetes resources

A **Pod** is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. The containers within a pod can communicate with each other via localhost.

A **Service** makes each pod within the cluster accessible to the other resources and also acts as a Load Balancer.

A **Deployment** describes in a declarative manner (mainly with yaml-files) the desired state of the system.

Ingress manages external access to the services in a cluster via HTTP or HTTPS.

Relevant Kubernetes resources

ReplicaSets ensure that a defined number of pods are executed in the cluster at all times. If a pod is removed or fails, the ReplicaSet automatically starts new instances until the desired cluster state is reached.

A **ConfigMap** is used to store non-confidential data in key-value pairs. Pods can consume these than as environment variables, command-line arguments, or as configuration files in a volume.

A **Secret** can be used for sensitive data such as a password, a token, or a key.

Jobs & Cronjobs are responsible for starting a pod with a specific task and then terminating it when the task has been completed. (Cronjobs execute these tasks at fixed times or regularly for an indefinite period of time.

kubectl

= Kubernetes command-line tool that allows you to run commands against your clusters, e.g.

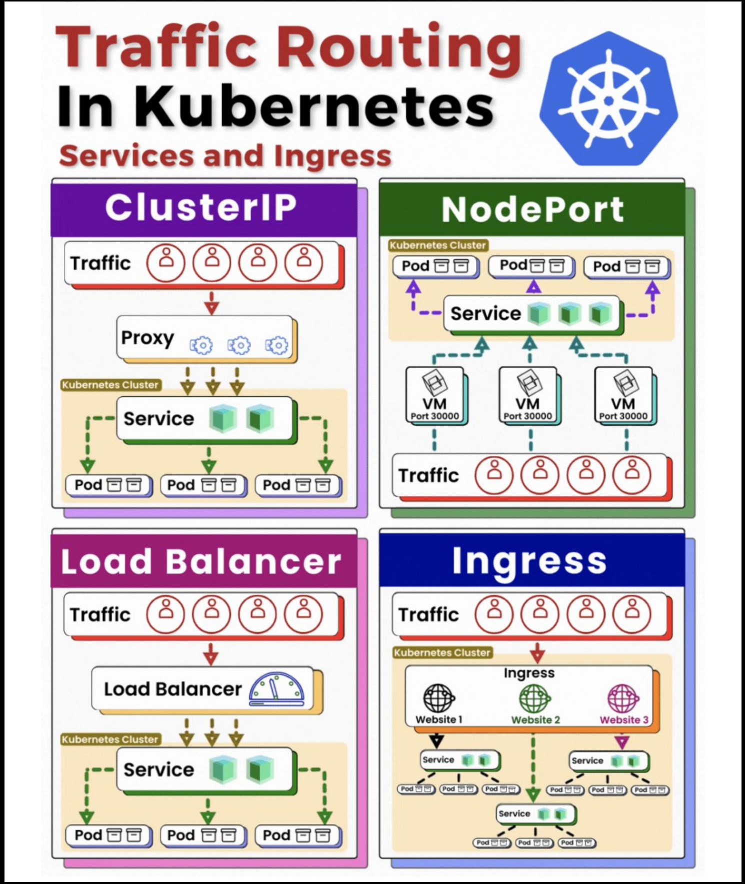
$$\left.\begin{array}{l}\text{kubectl create}\\ \text{kubectl get}\\ \text{kubectl describe}\\ \text{kubectl delete}\end{array}\right\}$$ can be used for all Kubernetes resources such as Pods, Nodes, Deployments, Services, ReplicaSets, ...

kubectl scale

kubectl rollout

kubectl apply -f

Exposing applications



https://www.linkedin.com/posts/brijpandeyji_kubernetes-services-and-ingress-demystified-activity-7168205635682033664-kLCe/