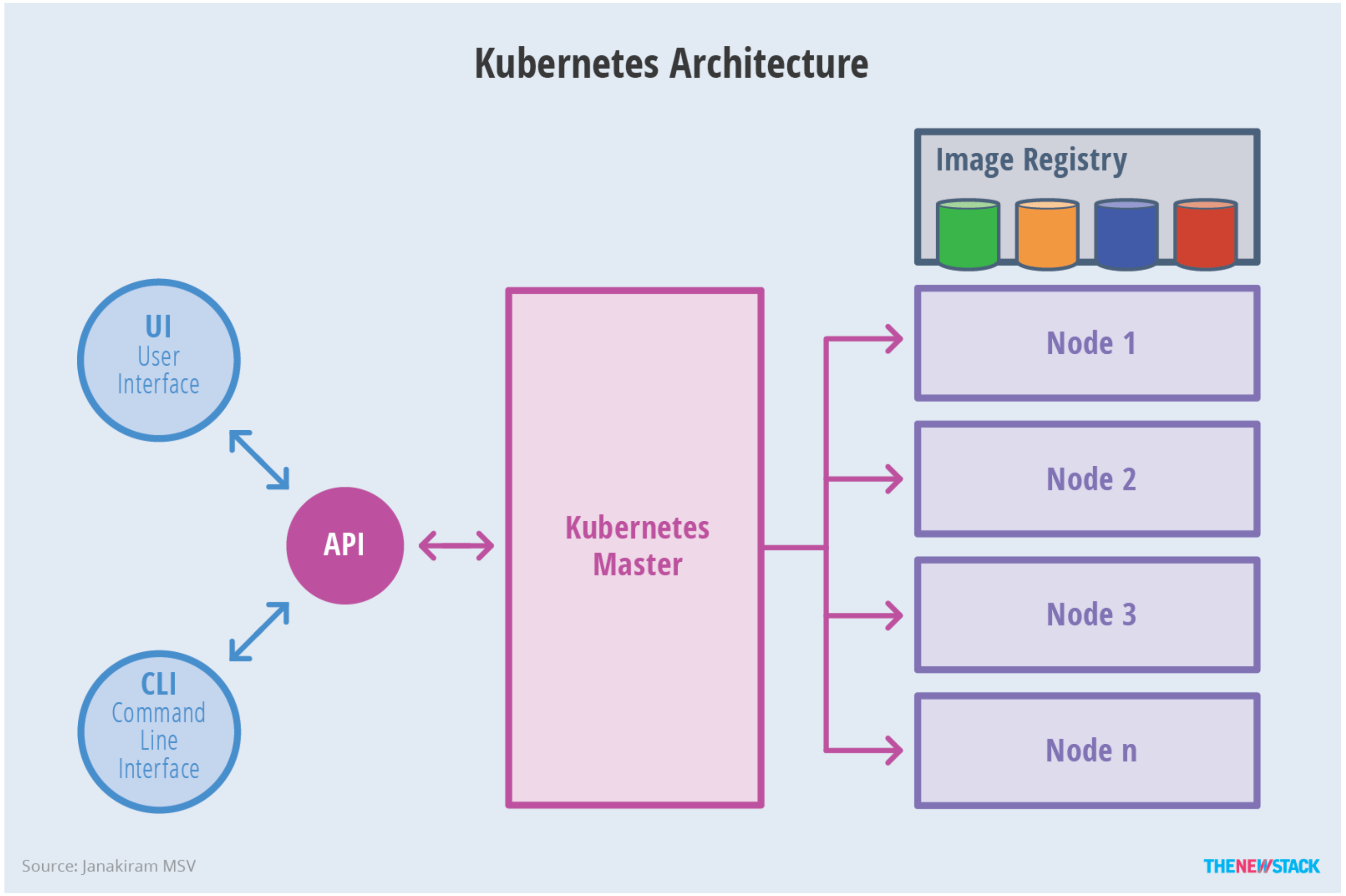
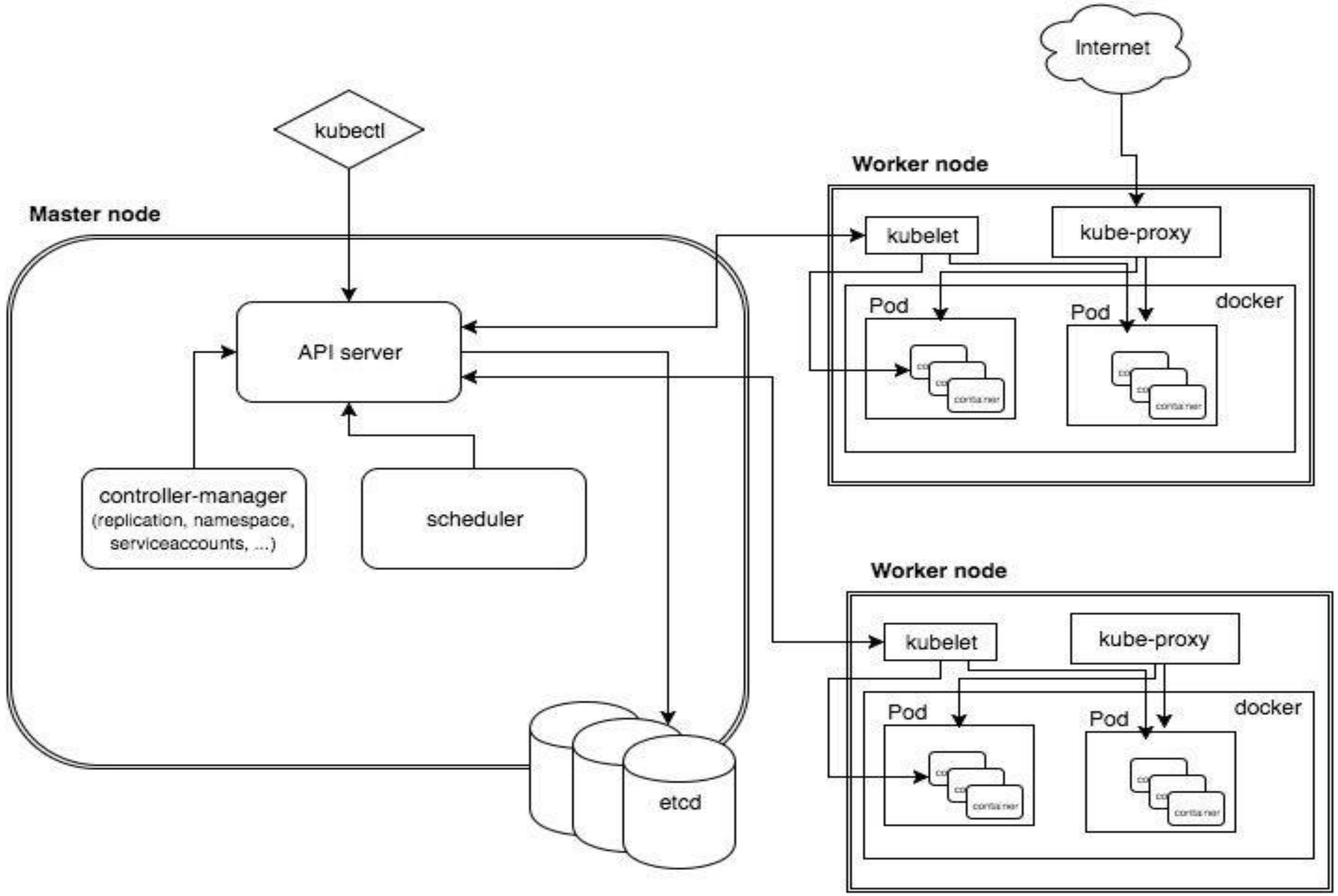
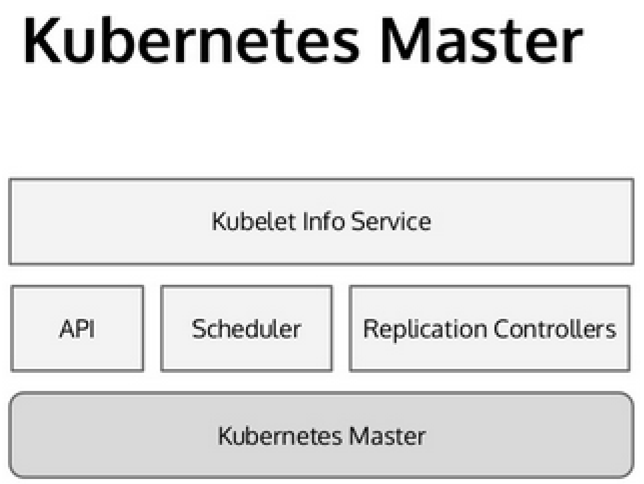
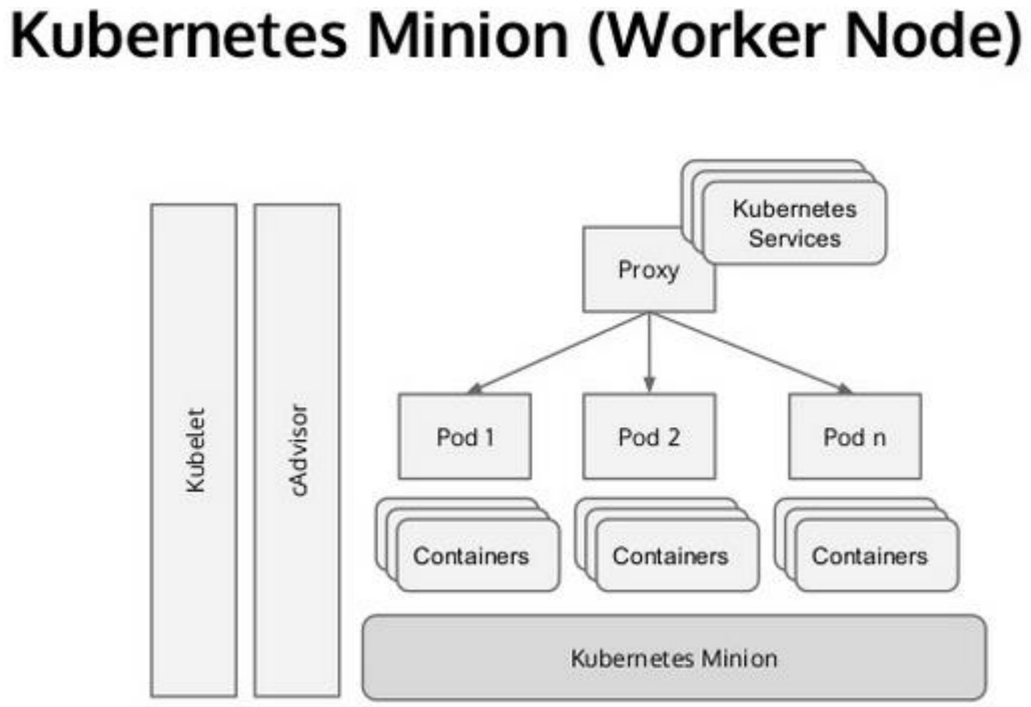
Kubernetes Architecture



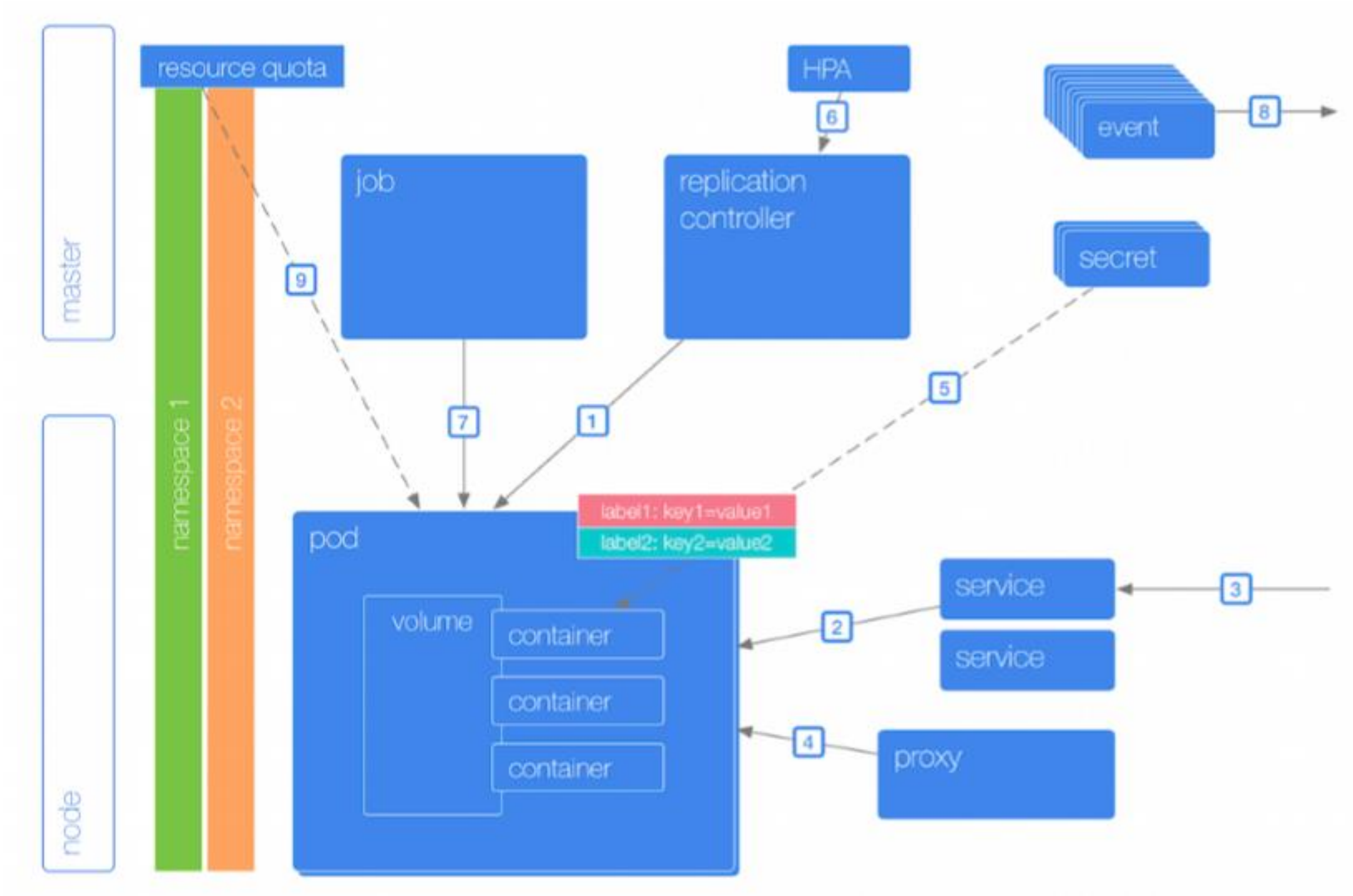

Internet
kubectl
Worker node
Master node
kubelet
kube-proxy
docker
Pod
Pod
API server
container
container
controller-manager
(replication, namespace, serviceaccounts, ...)
scheduler
Worker node
kubelet
kube-proxy
docker
Pod
Pod
container
container
etcd

Kubernetes architecture diagram





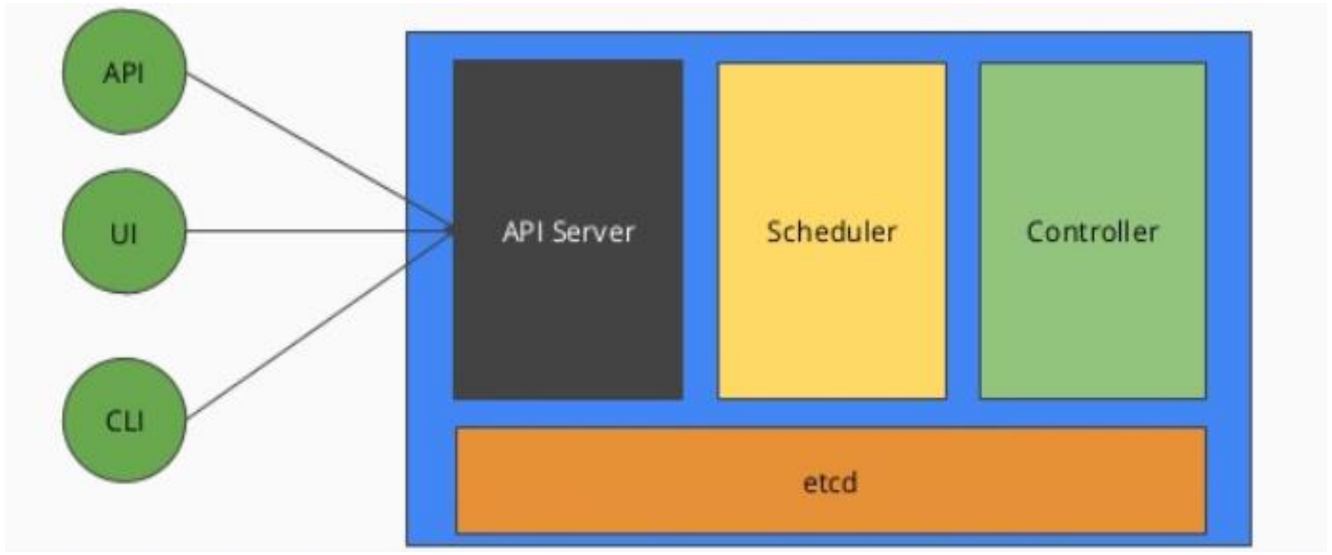
Kubernetes architecture diagram



Description of the components

The Master Node

- The master node is usually setup for High Availabilty (HA) to achieve high performance and failover purposes to reduce downtime.
- **kubectl** is used to interact with the cluster. Part from using the cluster through the Kubernetes dashboard or through API, *kubectl* is the main utility for interacting with Kubernetes from the command line.
- **API Server**: provides RESTful Kubernetes API to manage cluster configuration, backed by the etcd datastore.
- **etcd:** is a key-value store designed for strong consistency and high-availability. Kubernetes uses etcd to reliably store master state and configuration. The various master components 'watch' this data and act accordingly – for example, starting a new container to maintain a desired number of replicas.
- **Scheduler**: places unscheduled pods on nodes according to rules (e.g. labels). At this stage, the scheduler is simple but it is, like most components in Kubernetes, pluggable.
- **Controller Manager**: manages all cluster-level functions, including creating/updating endpoints, nodes discovery, management and monitoring, and management of pods.
- **Authentication/Authorization:** Kubernetes API endpoints are secured with TLS (Transport Layer Security) ensuring that every user is authenticated through the most secure mechanism available.
- **Controller and Replication Controller (RC) :** Control/Coordinate all nodes for maintain server as configure on cluster system
- **Secret**: Encrypt confidential data o HPA (Horizon Pods Auto scaling): Scale pods with CPU criteria (major)
- **Event**: Keep log and event on cluster
- **NameSpace**: Limit resource quota via define namespace (cpu, memory, pods etc)



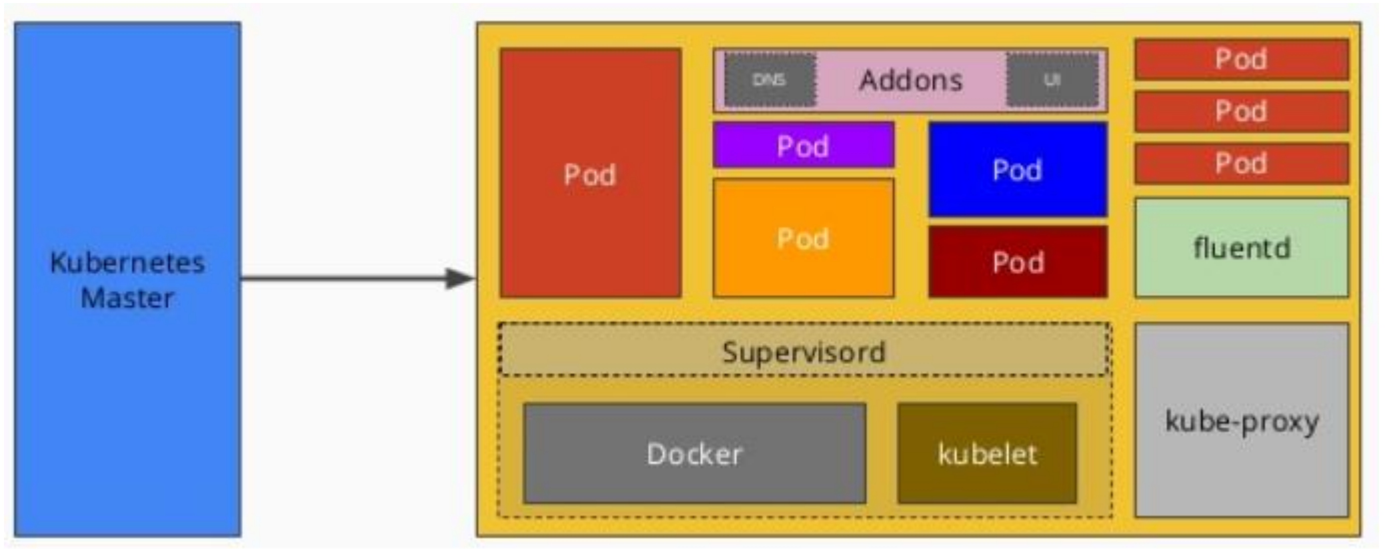
The Worker Nodes

The number of worker nodes depends on the size of the cluster. It can be one but its not uncommon to have Kubernetes cluster with over 2000 nodes. the size depends on your budget and use case. each worker node runs the following services.

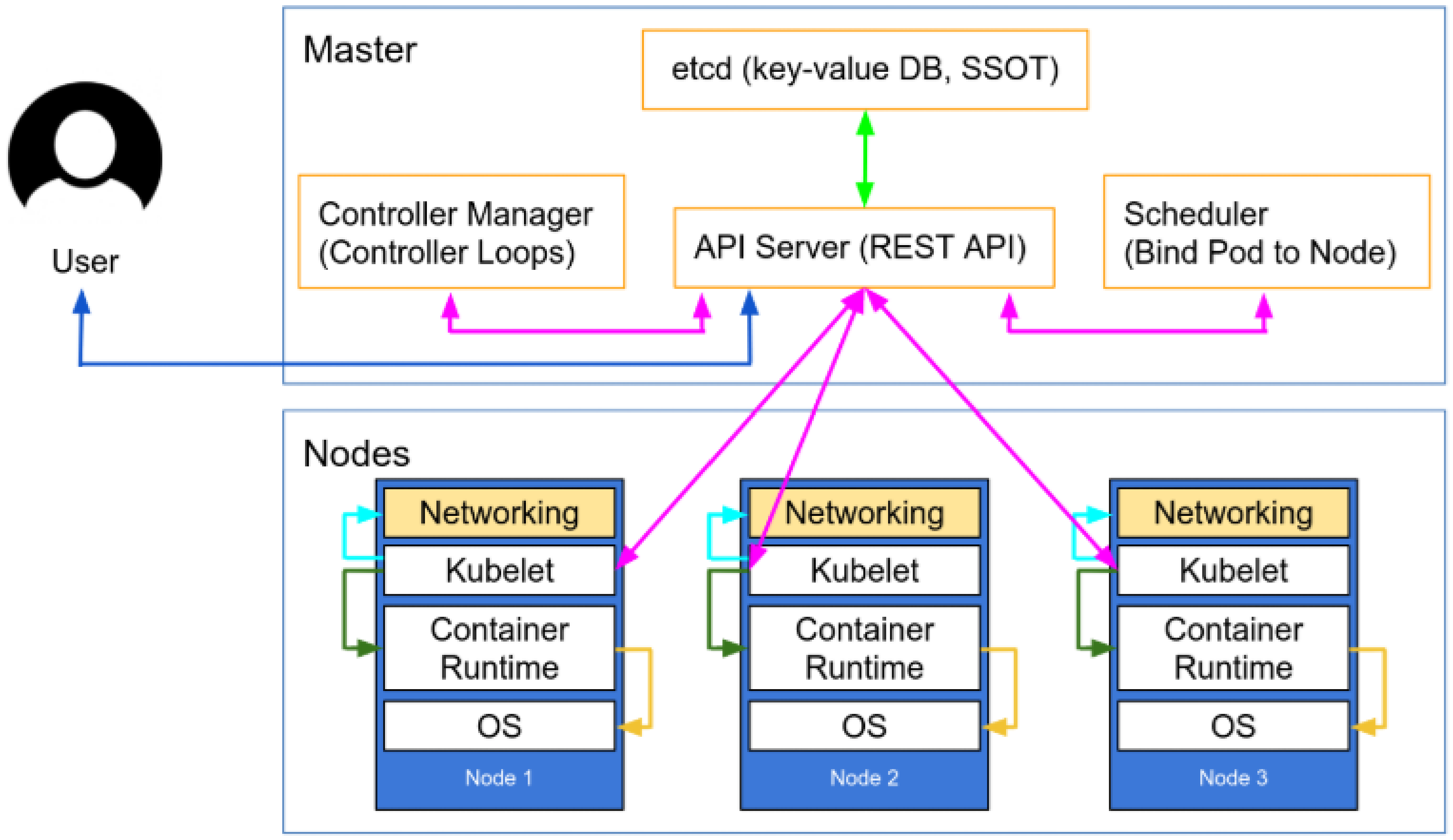
Kubelet: Is a service that receives orders from the **master** and a container runtime (e.g Docker) which it interfaces to manage container instances.

Kube-proxy: The proxy (**kube-proxy**) provides simple network proxy and load balancing capability. kube-proxy enables services to be exposed with a stable network address and name.

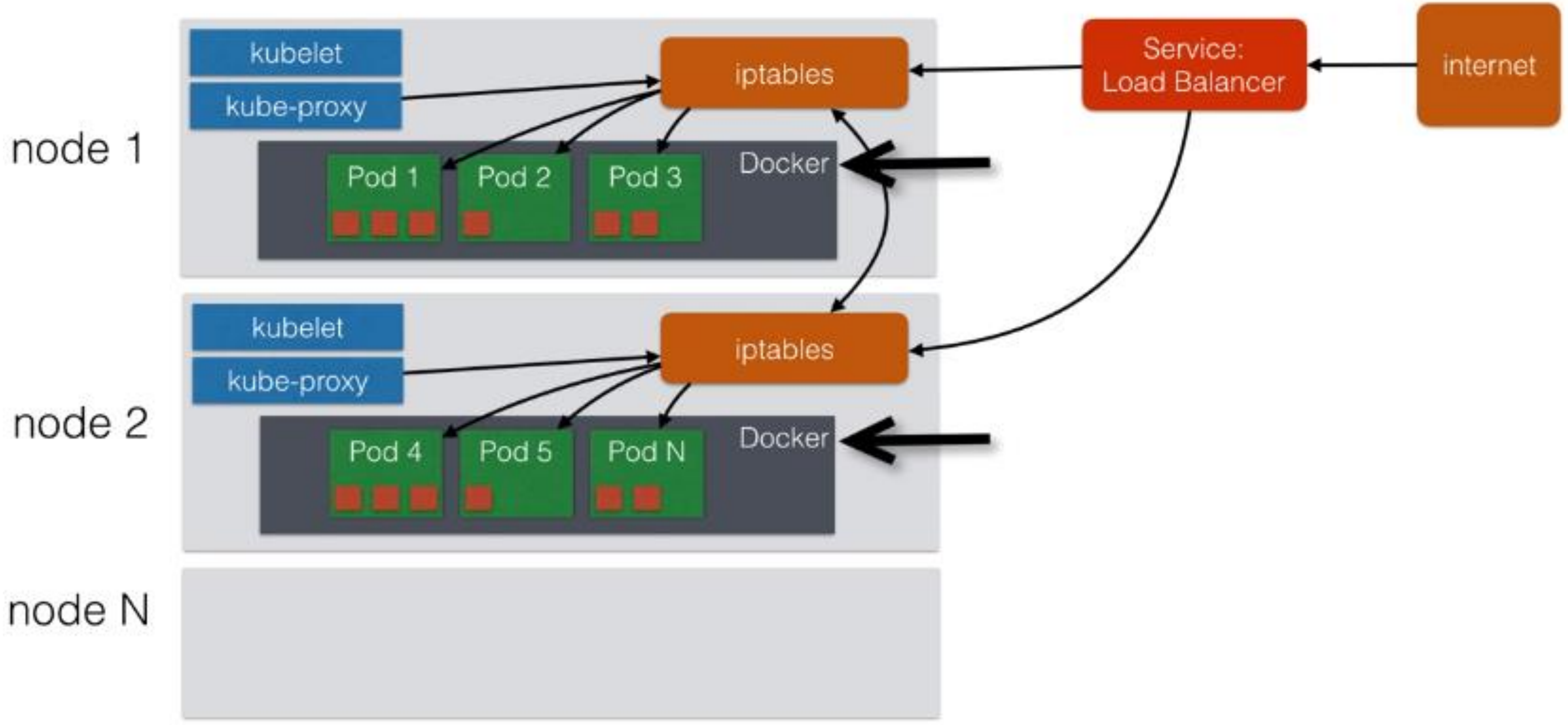
Docker: Docker is the container engine responsible for creating the pods on each slave node. To run the containers it has to download the images from a docker registry.



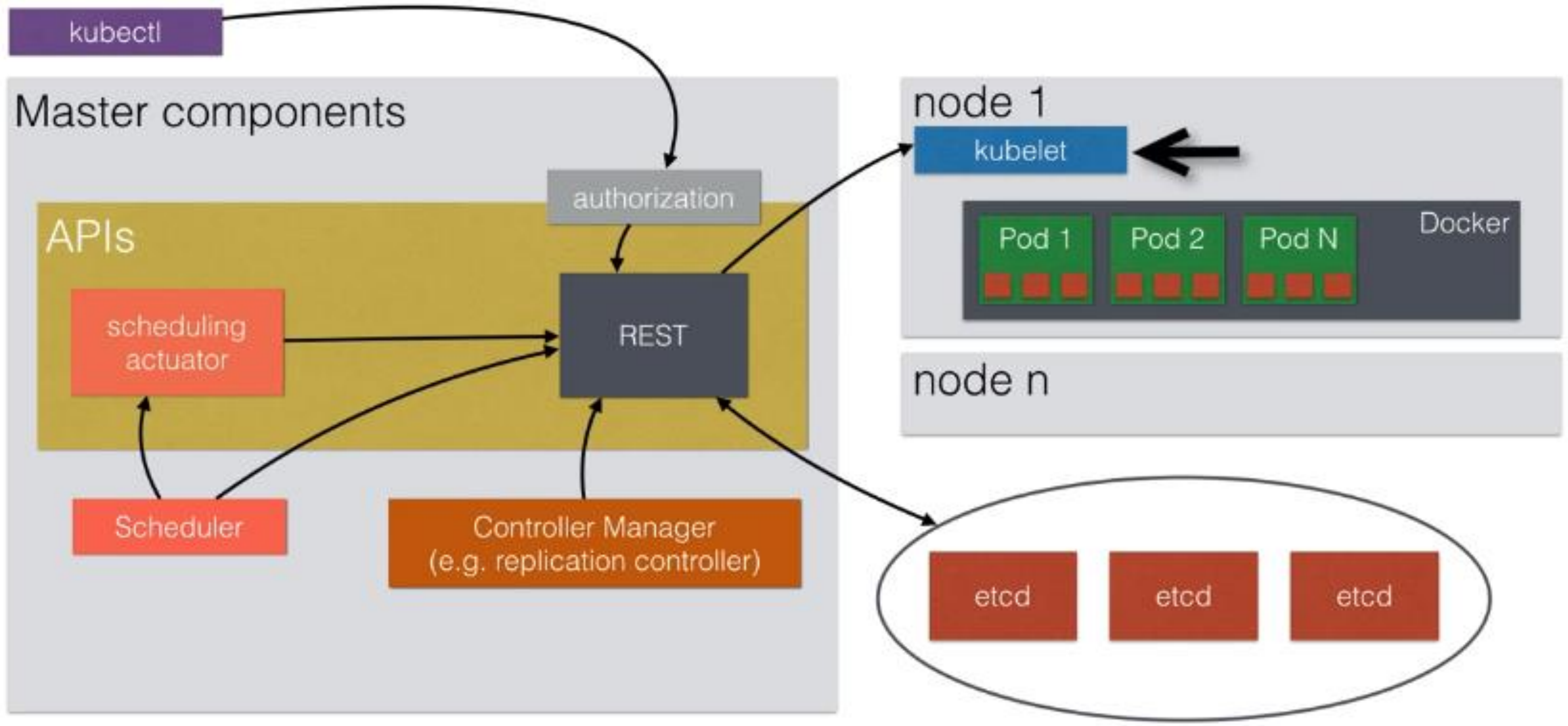
Kubernetes architecture diagram:


Master
etcd (key-value DB, SSOT)
Controller Manager (Controller Loops)
API Server (REST API)
Scheduler (Bind Pod to Node)
User
Nodes
Networking
Kubelet
Container Runtime
OS
Node 1
Node 2
Node 3

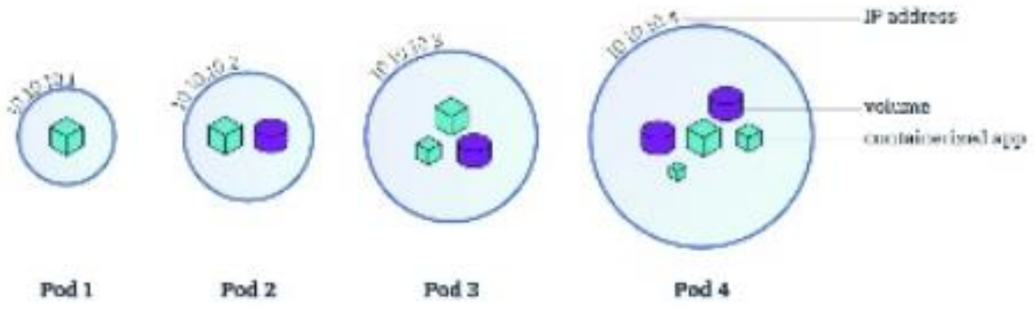
Architecture overview

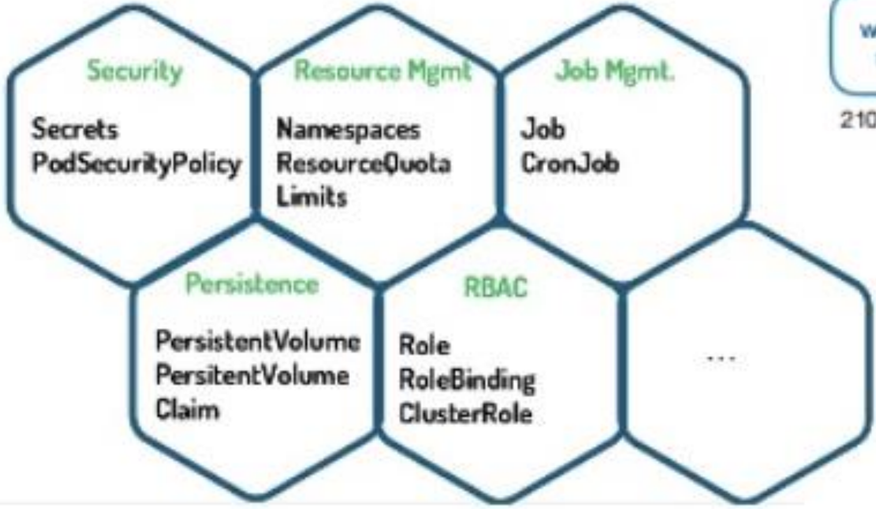


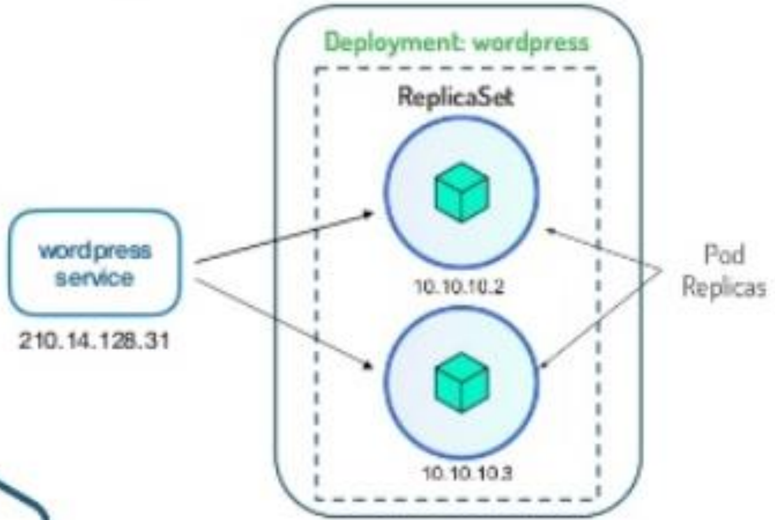
Architecture Overview



Kubernetes Objects


IP address
volume
containerized app
Pod 1
Pod 2
Pod 3
Pod 4


Security
Secrets
PodSecurityPolicy
Resource Mgmt
Namespaces
ResourceQuota
Limits
Job Mgmt.
Job
CronJob
Persistence
PersistentVolume
PersitentVolume
Claim
RBAC
Role
RoleBinding
ClusterRole
...


Deployment: wordpress
ReplicaSet
wordpress service
210.14.128.31
10.10.10.2
10.10.10.3
Pod Replicas