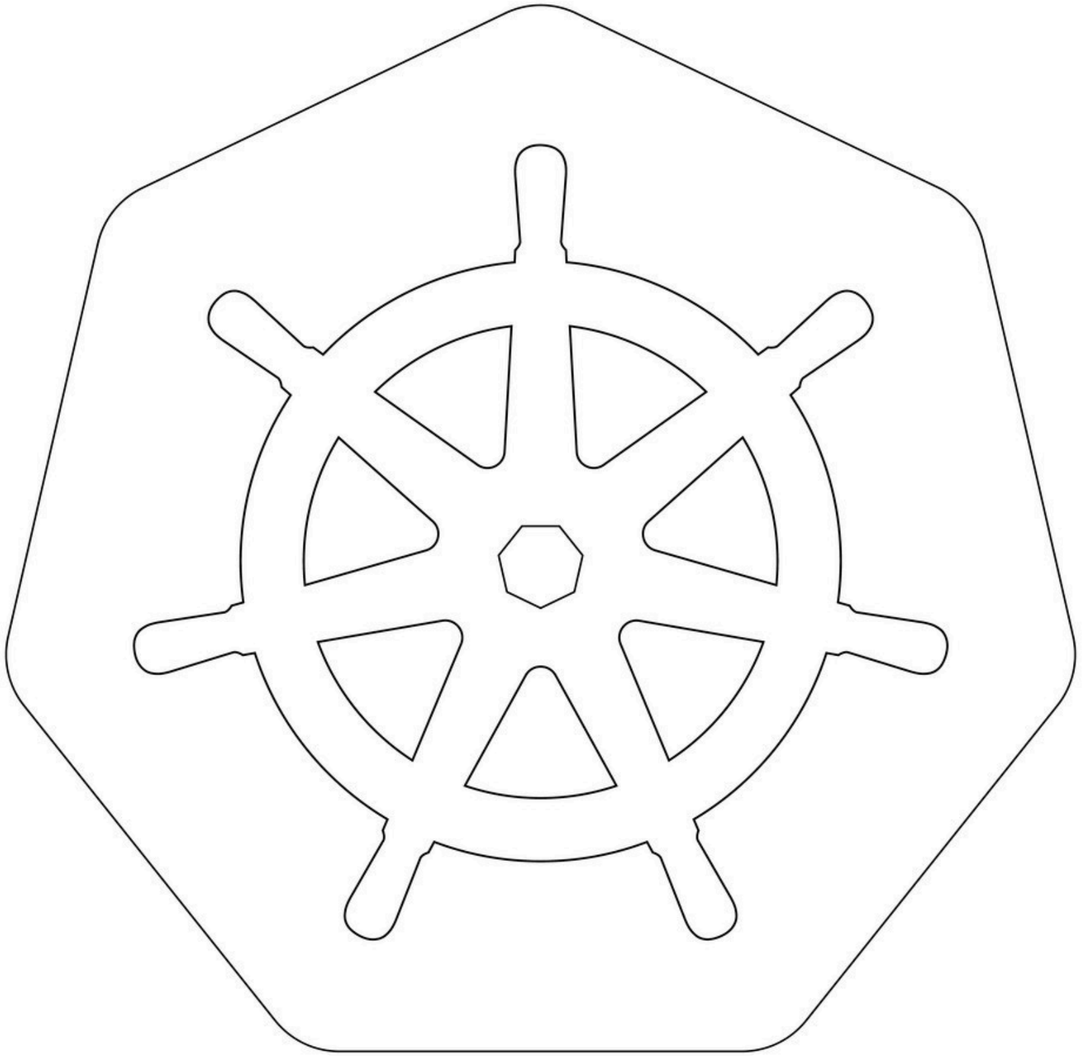
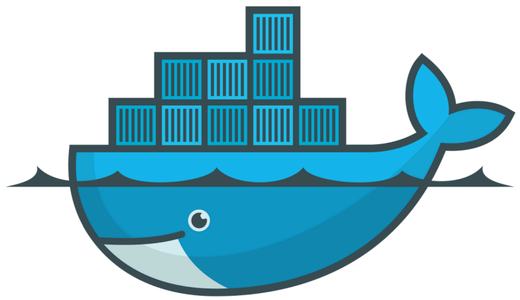


kubernetes

KUBERNETES:

KUBERNETES

K8S



WHY KUBERNETES

Earlier We used Docker Swarm as a container orchestration tool that we used to manage multiple containerized applications on our environments.

FEATURES	DOCKER SWARM	KUBERNETES
Setup	Easy	Complex
Auto Scaling	No Auto Scaling	Auto Scaling
Community	Good Community	Greater community for users like documentation, support and resources
GUI	No GUI	GUI

KUBERNETES:

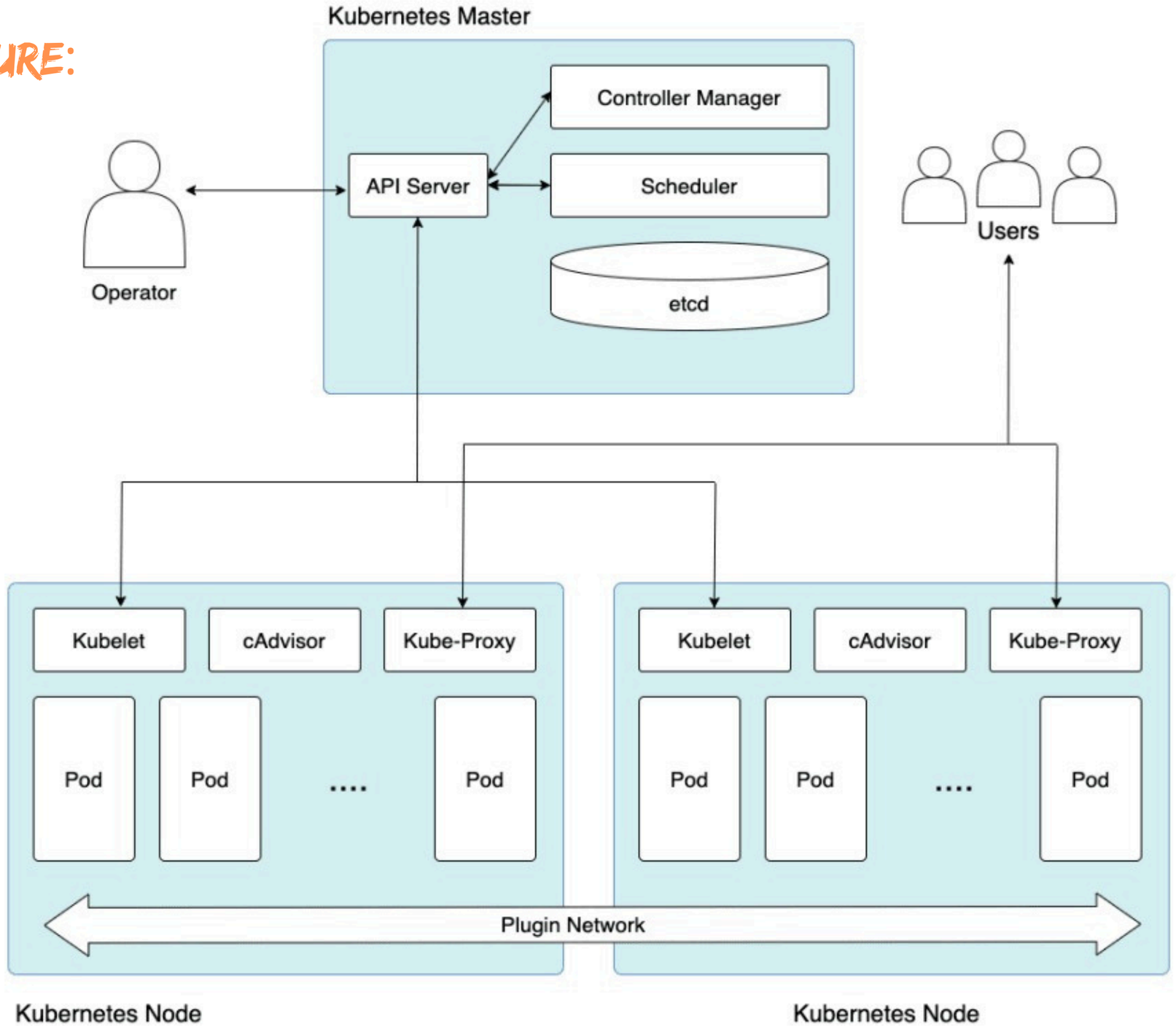
IT is an open-source container orchestration platform.
It is used to automates many of the manual processes like deploying, managing, and scaling containerized applications.
Kubernetes was developed by GOOGLE using GO Language.
Google donated K8's to CNCF in 2014.
1st version was released in 2015.

WHY KUBERNETES:

Containers are a good and easy way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. In docker we used docker swarm for this. but any how docker has drawbacks!

so we moved to KUBERNETES.

AECHITECTURE:



HOW IT WORKS:

get pods

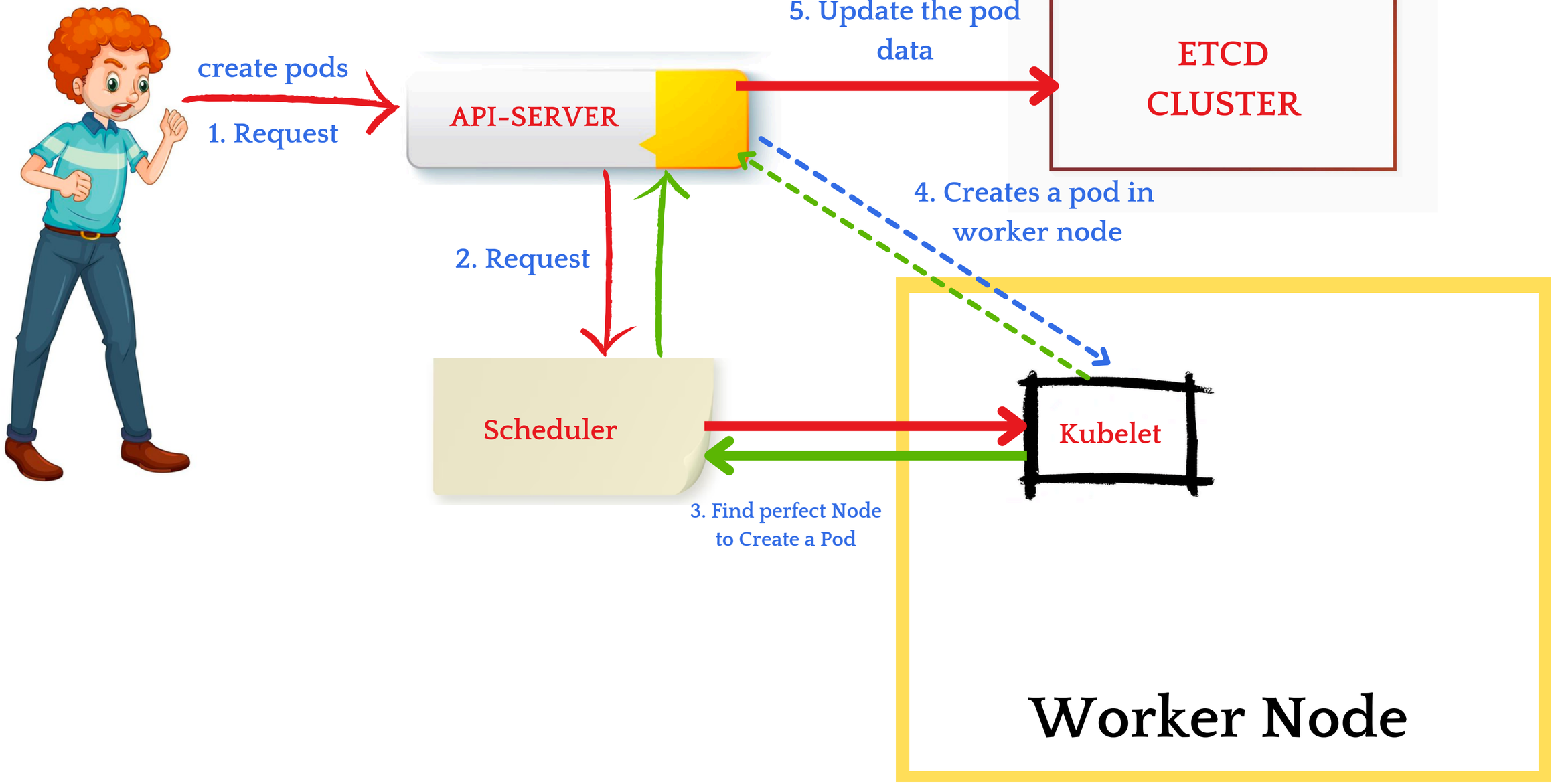
API-SERVER

ETCD
CLUSTER

Authenticate user → Validate Request → Retrieve Data

1 2 3

HOW POD CREATES:

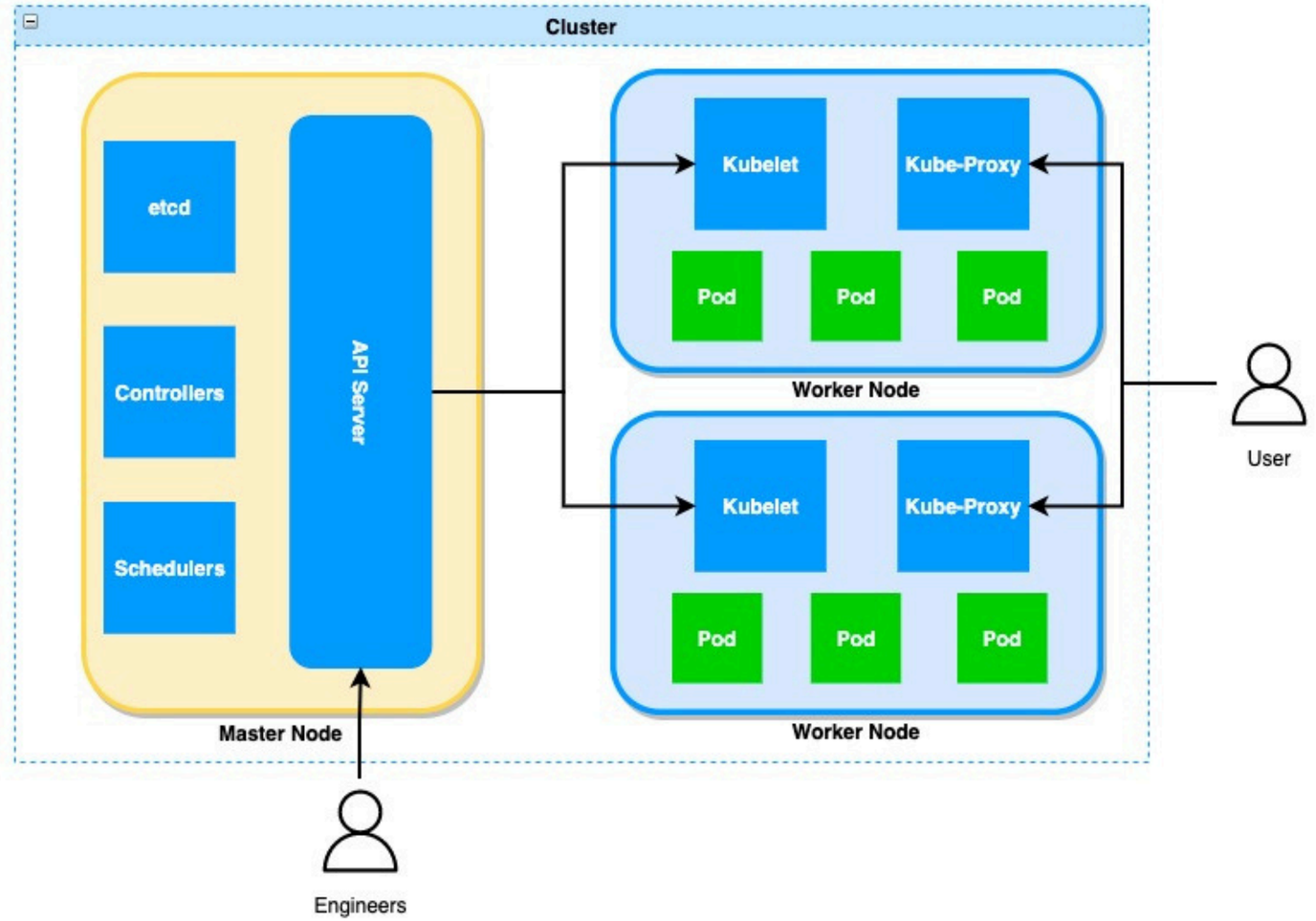


CLUSTER:

- It is a group of serversIt will have both manager and worker nodes.
- Master Node is used to assign tasks to Worker Nodes.
- Worker node will perform the task.
- we have 4 components in Master Node

  1.API Server
  2.ETCD
  3. Controllers-manager
  4. Schedulers
- we have 4 components in Worker Node.

  1. Kubelet
  2. Kube-Proxy
  3. Pod
  4. Container



API SERVER:

- It is used to accept the request from the user and store the request in ETCD.
- It is the only component that interacts with the ETCD directly

ETCD:

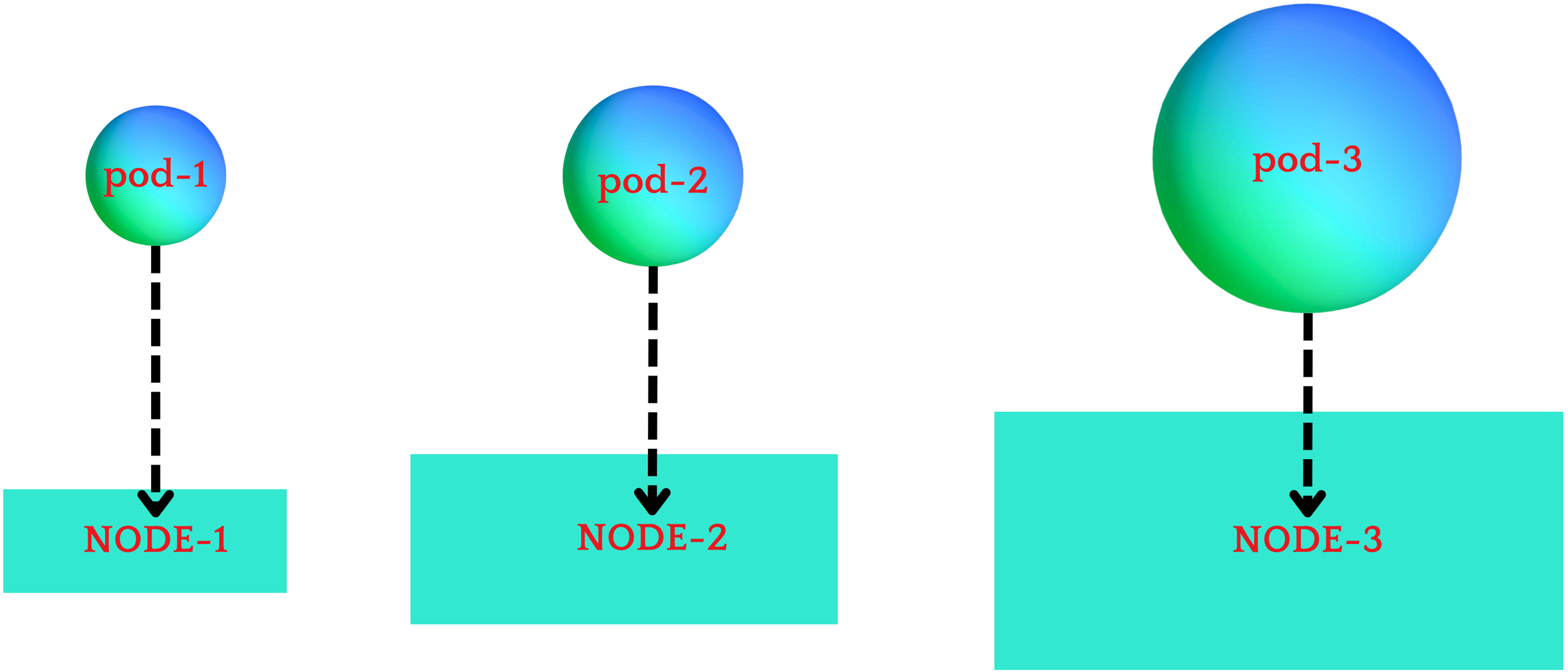
- It is like a database to our k8's
- it is used to store the requests of our cluster like pods, nodes, configs, secrets, roles etc..

SCHEDULER:

- It is used to search pending tasks which are present in ETCD.
- If any pending task found in ETCD, it will schedule in worker node.
- It will decide in which worker node our task should gets executed.
- It will decide by communication with the kubelet in worker node
- Scheduler always monitor API-Server
- The scheduler is a crucial component responsible for assigning pods to nodes
- This will ensure that pods are scheduled on suitable nodes that meet their resource requirements and constraints

HOW SCHEDULER DECIDES

The scheduler is going to decide the put the right pod in right node based on Size, Memory, CPU etc..



Just assume that we have 1 pod and 4 nodes just like the below image

Node-1	Node-2	Node-3	Node-4
2 CPU's	4 CPU's	10 CPU's	14 CPU's

Lets see How its decides

It Eliminates Node-1 & Node-2 because Node contains less CPU that pod

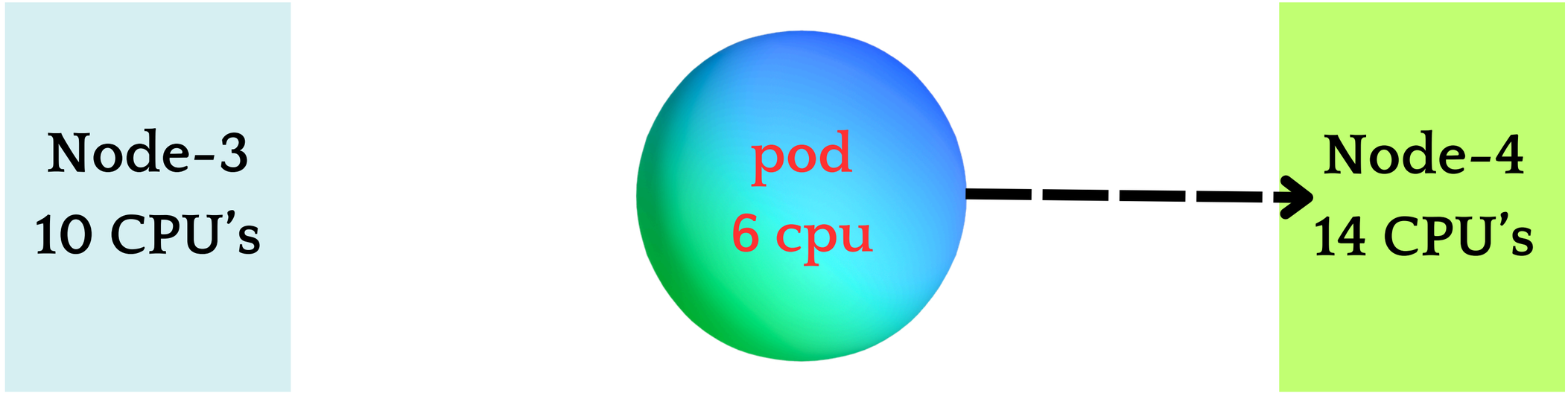
~~Node-1~~
~~2 CPU's~~

~~Node-2~~
~~4 CPU's~~

Node-3
10 CPU's

Node-4
14 CPU's

In the rest of these 2 nodes, scheduler decides based on CPU
Ex: If scheduler places the pod on Node-3 then Node-3 contains only 2 CPU's only
If scheduler places the pod on Node-4 then Node-4 contains only 4 CPU's.
So scheduler selects Node-4 to place a pod.



Not only CPU it selects based on different parameters like

- Resource Requirement limits
- Taints & Tolerations
- Node selector

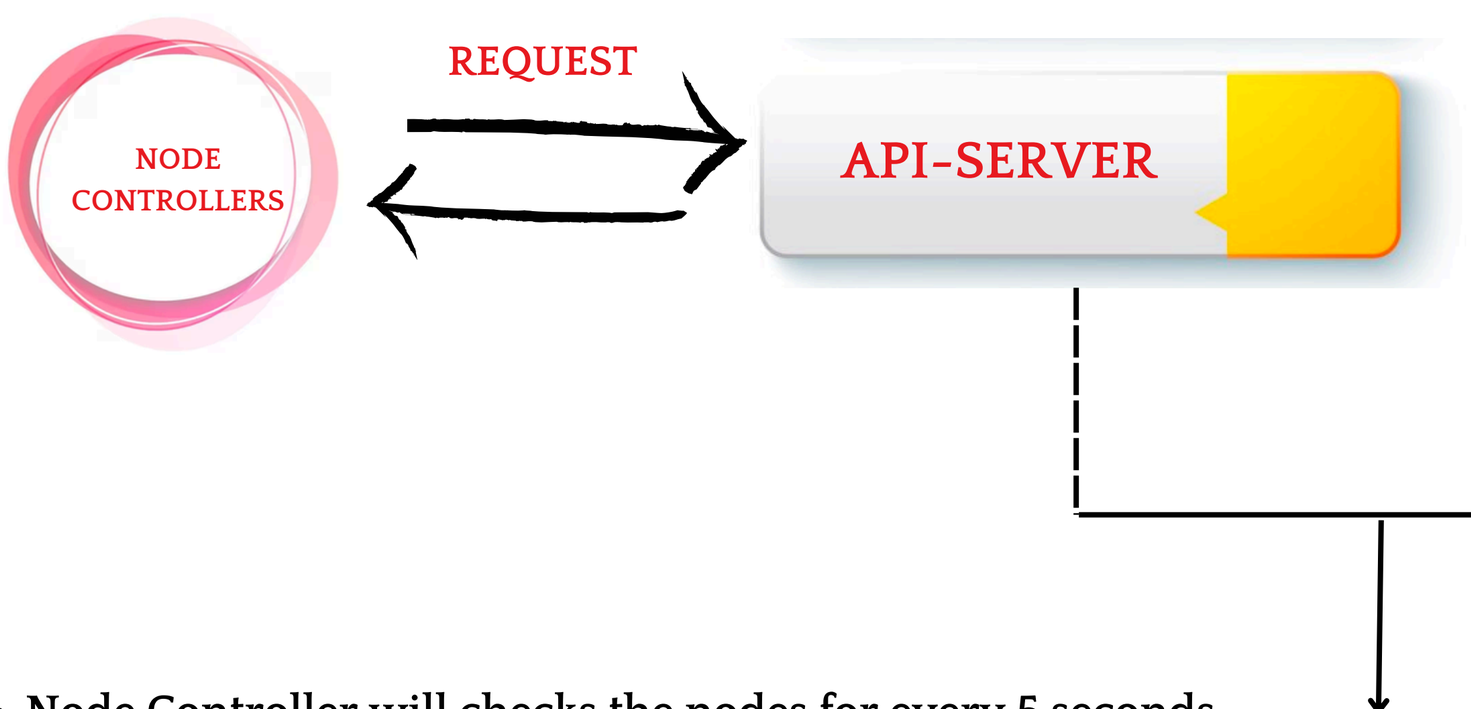
- Scheduler will do the following taks
    a. Pod Creation
    b. Pod scheduling
    c. Selecting Node
    d. Pod Running
    e. Kubelet Action

CONTROLLERS:

The controller Manager collects the data/information from the API Server of the Kubernetes cluster like the desired state of the cluster and then decides what to do by sending the instructions to the API Server.

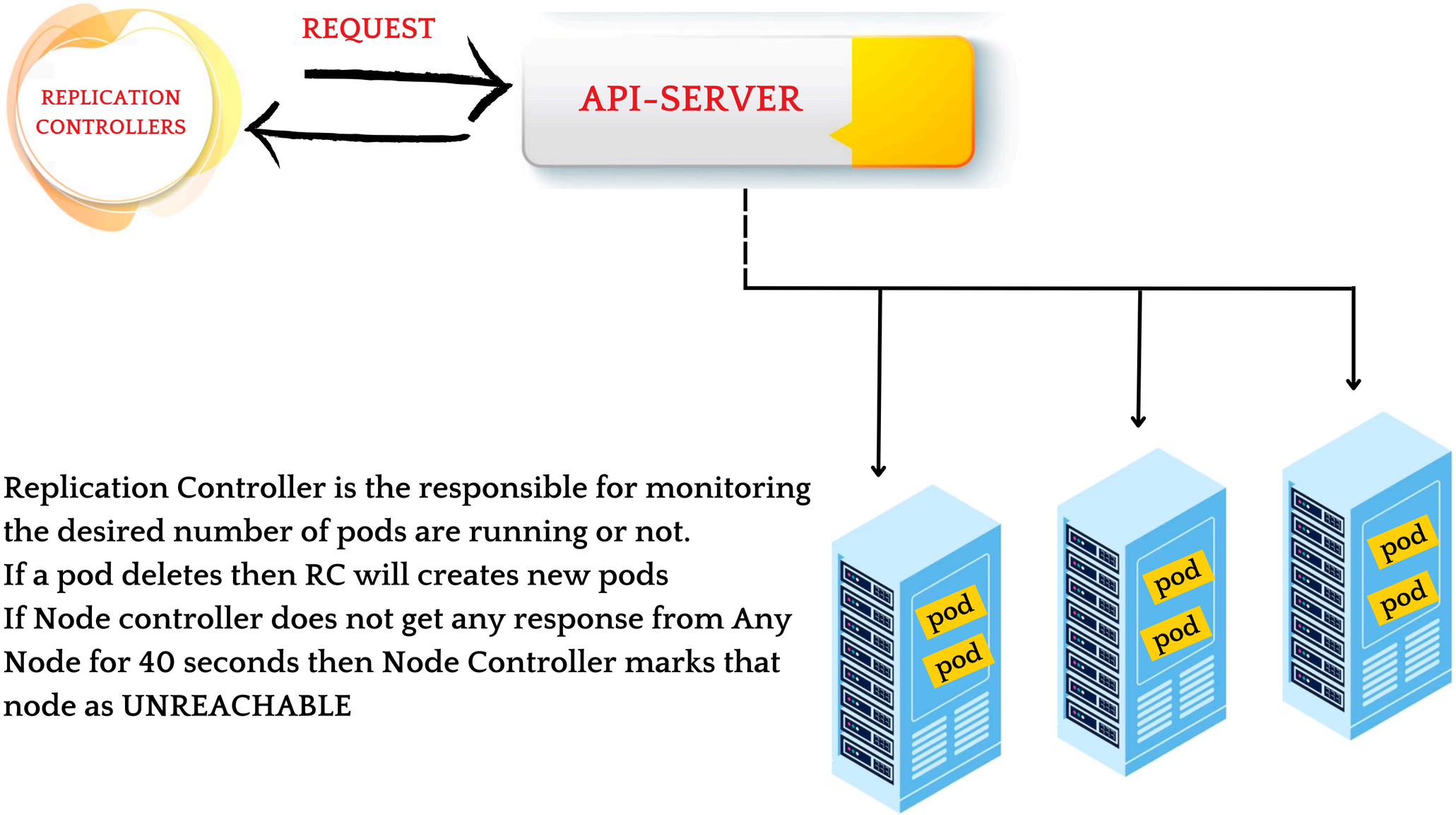
- Controller will do the following taks
    a. Replication Controller
    b. Node Controller
    c. Service Controller

HOW NODE CONTROLLER WORKS:



- Node Controller will checks the nodes for every 5 seconds through API-Server.
- It works like a heartbeat for our body.
- If Node controller does not get any response from Any Node for 40 seconds then Node Controller marks that node as UNREACHABLE

HOW REPLICATION CONTROLLER WORKS:



- Replication Controller is the responsible for monitoring the desired number of pods are running or not.
- If a pod deletes then RC will creates new pods
- If Node controller does not get any response from Any Node for 40 seconds then Node Controller marks that node as UNREACHABLE

KUBELET:

- kubelet is the primary component of the Worker Node which manages the Pods and regularly checks whether the pod is running or not.
- If pods are not working properly, then kubelet creates a new pod and replaces it with the previous one because the failed pod can’t be restarted hence, the IP of the pod might be changed

KUBE-PROXY

- kube-proxy contains all the network configuration of the entire cluster such as pod IP, Services, Deployments etc..
- It takes care of the load balancing and routing which comes under networking configuration.

POD: A group of one or more containers.

CONTAINER:

- It is a virtual machine which does not have any OS.
- it is used to run the applications in worker nodes.

KUBERNETES CLUSTER SETUP:

There are multiple ways to setup kubernetes cluster.

1.SELF MANAGER K8'S CLUSTER
a.mini kube (single node cluster)
b.kubeadm(multi node cluster)
c. KOPS

2. CLOUD MANAGED K8'S CLUSTER
a. AWS EKS
b.AZURE AKS
c.GCP GKS
d.IBM IKE

MINIKUBE:

It is a tool used to setup single node cluster on K8's.
It contains API Servers, ETCD database and container runtime
It helps you to containerized applications.
It is used for development, testing, and experimentation purposes on local.
Here Master and worker runs on same machine
It is a platform Independent.
By default it will create one node only.
Installing Minikube is simple compared to other tools.

NOTE: But we dont implement this in real-time

MINIKUBE SETUP:

REQUIREMENTS:

- 2 CPUs or more
- 2GB of free memory
- 20GB of free disk space
- Internet connection
- Container or virtual machine manager, such as: Docker.

UPDATE SERVER:

1 apt update -y

2 apt upgrade -y

INSTALL DOCKER:

3 sudo apt install curl wget apt-transport-https -y

4 sudo curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

INSTALL MINIKUBE:

5 sudo curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

6 sudo mv minikube-linux-amd64 /usr/local/bin/minikube

7 sudo chmod +x /usr/local/bin/minikube

8 sudo minikube version

INSTALL KUBECTL:

9 sudo curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

10 sudo curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

11 sudo echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

12 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

13 sudo kubectl version --client

14 sudo kubectl version --client --output=yaml

15 sudo minikube start --driver=docker --force

KUBECTL:

- kubectl is the CLI which is used to interact with a Kubernetes cluster.
- We can create, manage pods, services, deployments, and other resources
- We can also monitoring, troubleshooting, scaling and updating the pods.
- To perform these tasks it communicates with the Kubernetes API server.
- It has many options and commands, to work on.
- The configuration of kubectl is in the $HOME/.kube directory.
- The latest version is 1.28

SYNTAX:

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

kubectl api-resources : to list all api resources

Lets start working on kubernetes!

Note:

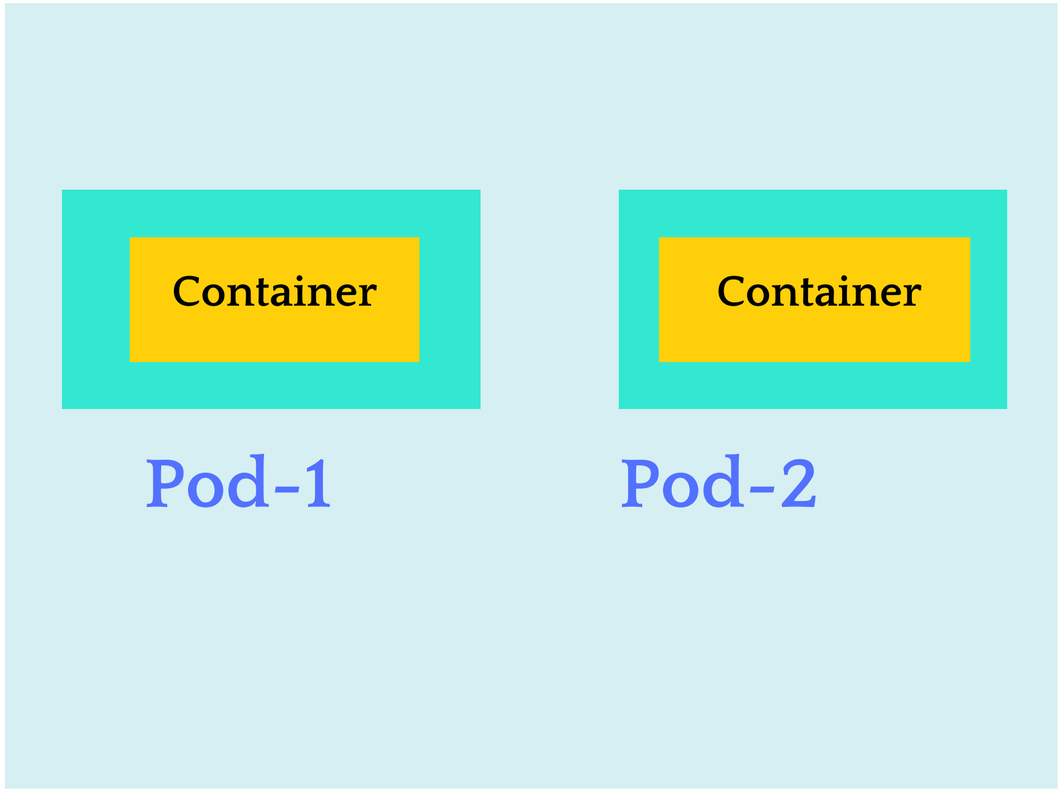
Kubernetes will not deploy the containers directly on worker nodes.
Kubernetes has a object called POD which contains containers.

Lets learn about PODS

POD:

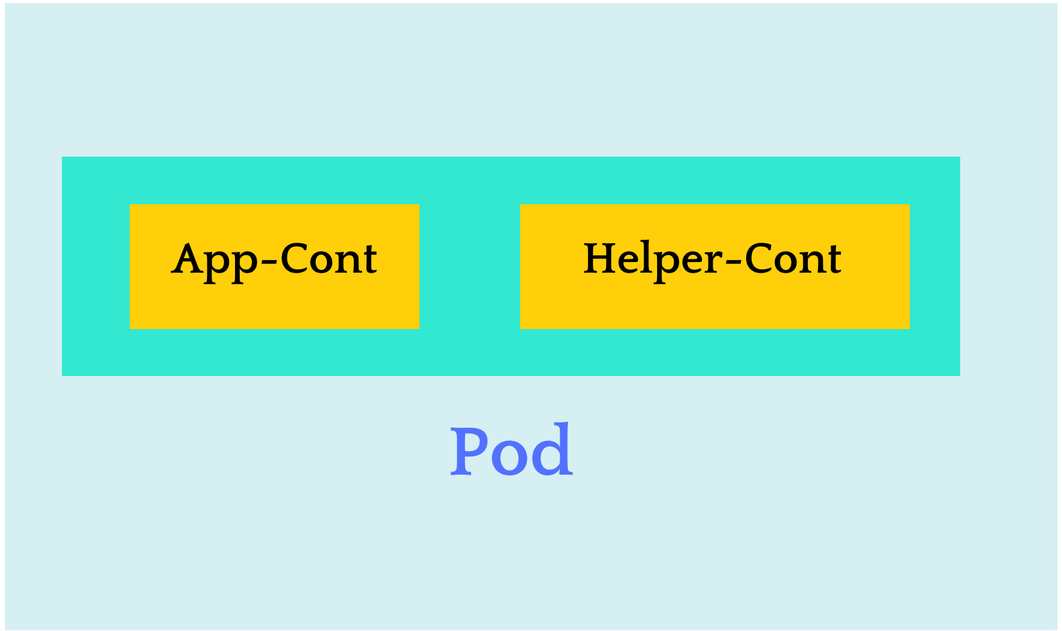
- It is a smallest object that we can create in K8's.
- It is a group of containers.
- Pod acts like a single instance for our application.
- Pods are ephemeral (short living objects)
- Mostly we can use single container inside a pod but if we required, we can create multiple containers inside a same pod.
- when we create a pod, containers inside pods can share the same network namespace, and can share the same storage volumes .
- While creating pod, we must specify the image, along with any necessary configuration and resource limits.
- K8's cannot communicate with containers, they can communicate with only pods.
- We can create this pod in two ways,
    - 1. Imperative(command)
    - 2. Declarative (Manifest file)

SINGLE CONTAINER POD



Basically, each pods needs only one container. But sometimes we need 2 containers in one pod.

MULTI CONTAINER POD



Helper containers are additional containers included in a pod to perform some tasks that support the primary container which are running in the pod.

How Helper container works:

- A helper container is used to collect the logs from the primary container.
- Helper containers can be used to run monitoring and metrics collection agents from the main application container.
- Helper containers can assist in syncing files or data between containers in a pod. For example, a pod might have a main application container and a helper container responsible for syncing configuration files or other shared resources.
- Security-related tasks, such as handling secrets, encryption, or authentication, can be done by helper container.

POD CREATION:

IMPERATIVE:

The imperative way uses kubectl command to create pod.
This method is useful for quickly creating and modifying the pods.
SYNTAX: kubectl run pod_name --image=image_name
COMMAND: *kubectl run pod-1 --image=nginx*

kubectl : command line tool run : action
pod-1 : name of pod
nginx : name of image

KUBECTL:

DECLARATIVE:

The Declarativ tive way we need to create a Manifest file in YAML Extension.
This file contains the desired state of a Pod.
It takes care of creating, updating, or deleting the resources.
This manifest file need to follow the yaml indentation.
YAML file consist of KEY-VALUE Pair.
Here we use create or apply command to execute the Manifest file.

SYNTAX: kubectl create/apply -f file_name

CREATE: if you are creating the object for first time we use create only.
APPLY: if we change any thing on files and changes need to apply the resources.

MANIFEST FILE:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

apiVersion: This is the version of API from Kubernetes which is used to create objects.

kind: defines the object that we are creating.

metadata: data about the object like names, labels etc ...

spec: It is the specifications of the object

COMMANDS:

- To get all the pods: **kubetctl get pods (or) kubectl get pod (or) kubectl get po**
- To delete a pod: **kubectl delete pod pod_name**
- To get IP of a pod: **kubectl get po pod_name -o wide**
- To get IP of all pods: **kubectl get po -o wide**
- To get all details of a pod: **kubectl describe pod podname**
- To get all details of all pods: **kubectl describe po**
- To get the pod details in YAML format: **kubectl get pod pod-1 -o yaml**
- To get the pod details in JSON format: **kubectl get pod pod-1 -o json**
- To enter into a pod: **kubectl exec -it pod_name -c cont_name bash**
- To get the logging info of our pod: **kubectl logs pod_name**
- To get the logs of containers inside the pod: **kubectl logs pod_name -c cont-name**

KUBERNETES METRICS SERVER:

- It is used to monitor the Kubernetes components like Pods, Nodes & Deployments etc...
- The Kubernetes Metrics Server measures CPU and memory usage across the Kubernetes cluster like Pods & Nodes.

INSTALLATION:

DOWNLOAD THE FILE: kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
SEE THE DEPLOYMENT: kubectl -n kube-system get deploy metrics-server
EDIT THE DEPLOYMENT : kubectl -n kube-system edit deploy metrics-server
Add these commands under the container args
- /metrics-server
- --kubelet-insecure-tls
- --kubelet-preferred-address-types=InternalIP

```
      k8s-app: metrics-server
  spec:
    containers:
    - args:
      - --cert-dir=/tmp
      - --secure-port=4443
      - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      - --kubelet-use-node-status-port
      - --metric-resolution=15s
      - /metrics-server
      - --kubelet-insecure-tls
      - --kubelet-preferred-address-types=InternalIP
```

MONITOR:

- Check the Node metrics: kubectl top nodes
- Now create some pods and check the pod metrics : kubectl top pod

Labels, Selectors, and Node Selectors:

Labels:

- Labels are used to organize Kubernetes Objects such as Pods, nodes, etc.
- You can add multiple labels over the Kubernetes Objects.
- Labels are defined in key-value pairs.
- Labels are similar to tags in AWS or Azure where you give a name to filter the resources quickly.
- You can add labels like environment, department or anything else according to you.

Label-Selectors:

- Once the labels are attached to the Kubernetes objects those objects will be filtered out with the help of labels-Selectors known as Selectors.

- The API currently supports two types of label-selectors equality-based and set-based. Label selectors can be made of multiple selectors that are comma-separated.

Node Selector:

Node selector means selecting the nodes. Node selector is used to choose the node to apply a particular command.

This is done by Labels where in the manifest file, we mentioned the node label name. While running the manifest file, master nodes find the node that has the same label and create the pod on that container.

Make sure that the node must have the label. If the node doesn’t have any label then, the manifest file will jump to the next node.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    env: testing
    department: DevOps
spec:
  containers:
    - name: containers1
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo This is our Pod; sleep 5 ; done"]
```

To see the list of pods with labels: kubectl get pods --show-labels

Now, I want to list those pods that have the label env=testing.

kubectl get pods -l env=testing

kubectl get pods -l department!=DevOps

As we have discussed earlier, there are two types of label-selectors equality and set-based.

This is the example of equality based where we used equalsTo(=).

Now, Suppose I forgot to add the label through the declarative(via manifest) method. So, I can add labels after creating the pods as well which is known as the imperative(via command) method.

kubectl label pods Pod_name Location=India

As we discussed the type of label-selector, let's see the example of a set-based label-selector where we use in, notin, and exists.

Lets list all those pods that have an env label with value for either testing or development.

kubectl get pods -l 'env in (testing, development)'

we are trying to list all those pods that don't have the India or US value for the Location key in the Label.

kubectl get pods -l 'Location not in (India, US)'

We can also delete the pods as per the label.
Let's deleted all those pods that don't have the Chinda value for the location key in the Label.

kubectl delete pod -l Location!=China

WE DEPLOYED THE POD

LETS ACCESS OUR APPLICATION