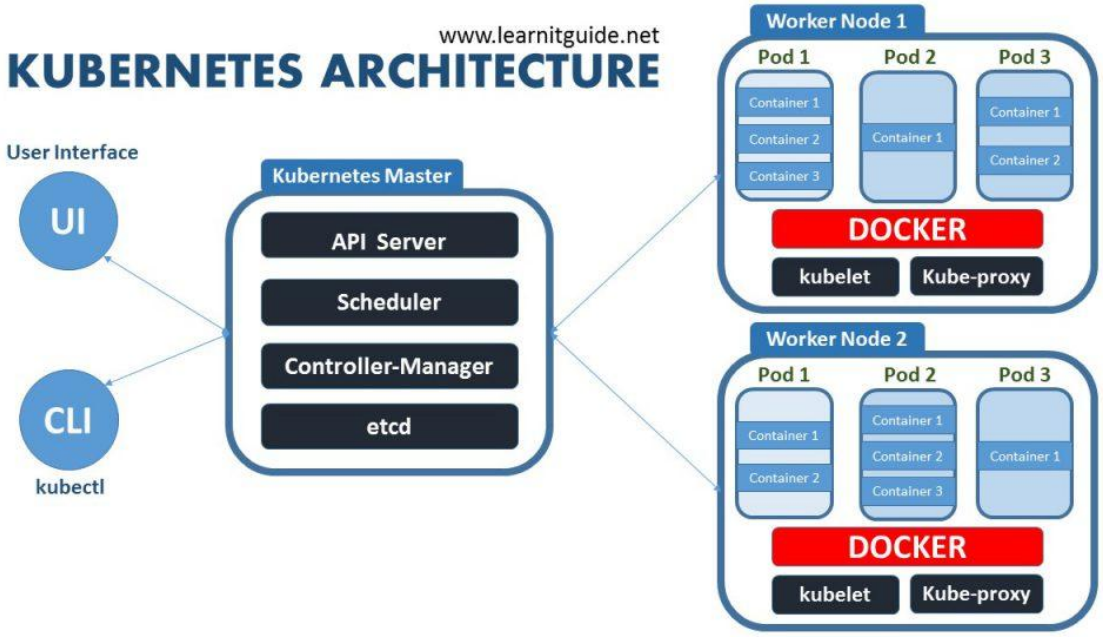
What is Kubernetes?

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers. It simplifies the developer's task of managing containerised applications. It solves many problems teams face during the management of containerised applications. Some of these challenges are as below:



Challenges of Containerized Applications

Managing containerized applications, whether using Docker containers or some other container runtime, comes with its own set of challenges, such as:

Scalability: As the number of containers grows, it becomes challenging to scale them effectively.

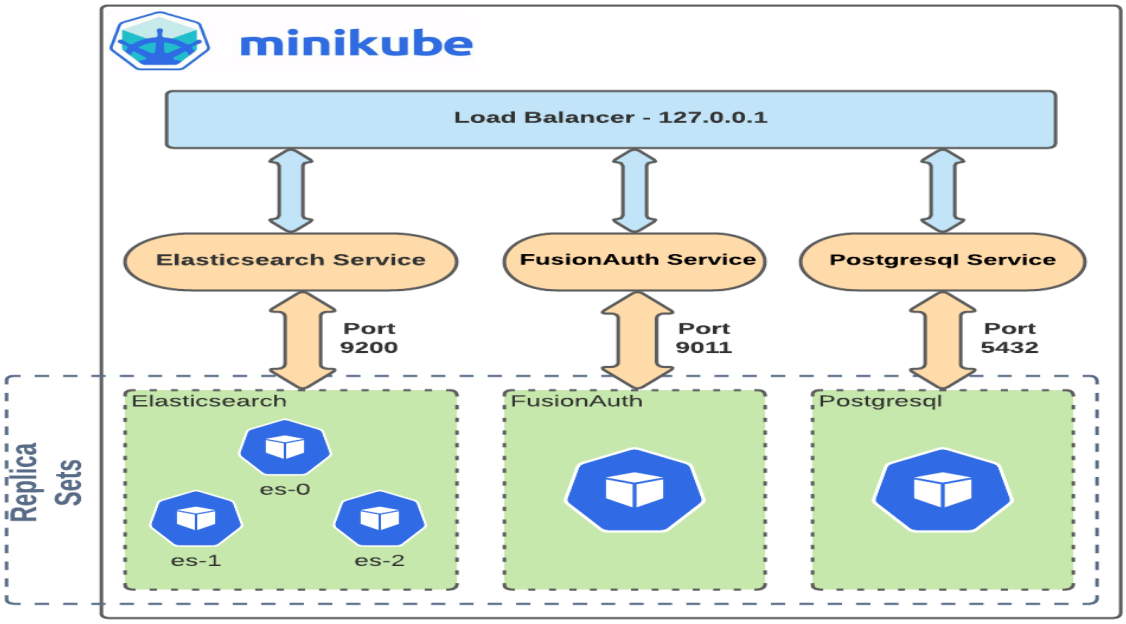
Complexity: Managing numerous containers, each with its role in a larger application, adds complexity.

Management: Keeping track of and maintaining these containers, ensuring they are updated and running smoothly, requires significant effort.

Setup local K8s cluster with minikube

What is Minikube?

Minikube is a lightweight Kubernetes implementation that lets you create a single-node Kubernetes cluster on your local machine. It's ideal for learning, development, and testing, without the need to set up a full-fledged production-level Kubernetes cluster.



Install Minikube on Linux:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux
sudo install minikube-linux-amd64 /usr/local/bin/minikube

# check version
minikube version
```

Start Minikube:

```
minikube start

minikube status
```

Deploying a Sample Application:

```
kubectl create deployment nginx --image=nginx

# Check the status of the deployment:
kubectl get deployments


#Expose the deployment as a service:
kubectl expose deployment nginx --type=NodePort --port=80


# Check the services to find the port Minikube has assigned:

kubectl get services


# To access your NGINX server, you need to use the Minikube IP:
minikube service nginx --url
```

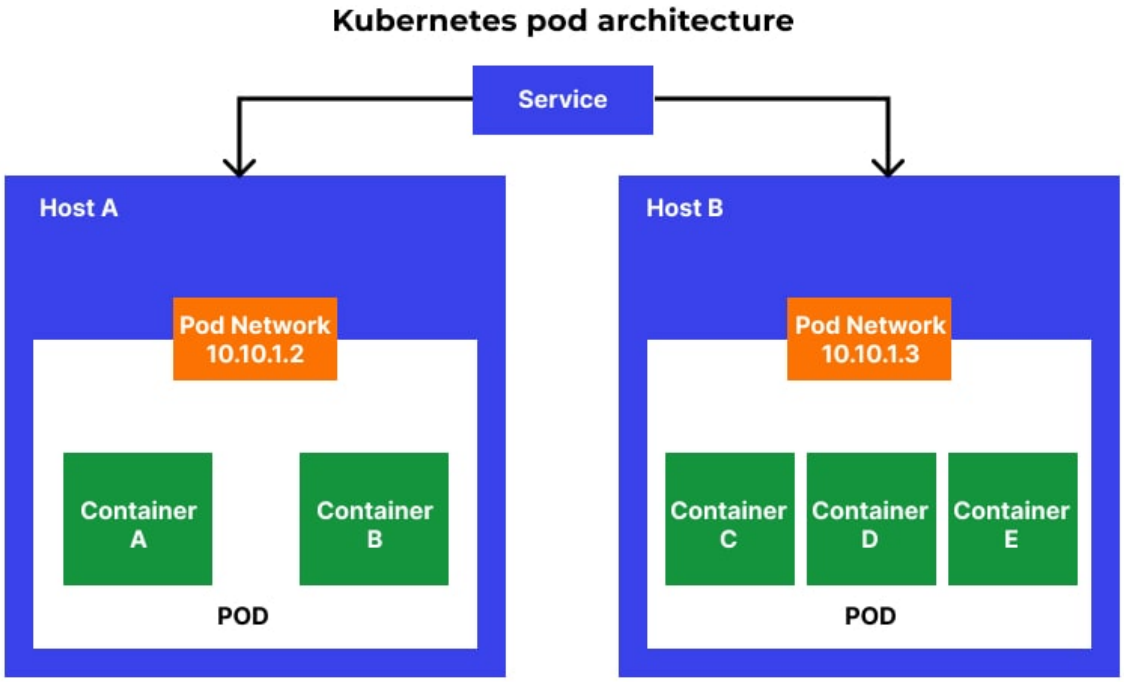
```
kubectl get nodes

kubectl get pods

minikube stop

minikube delete
```

Kubernetes Pods



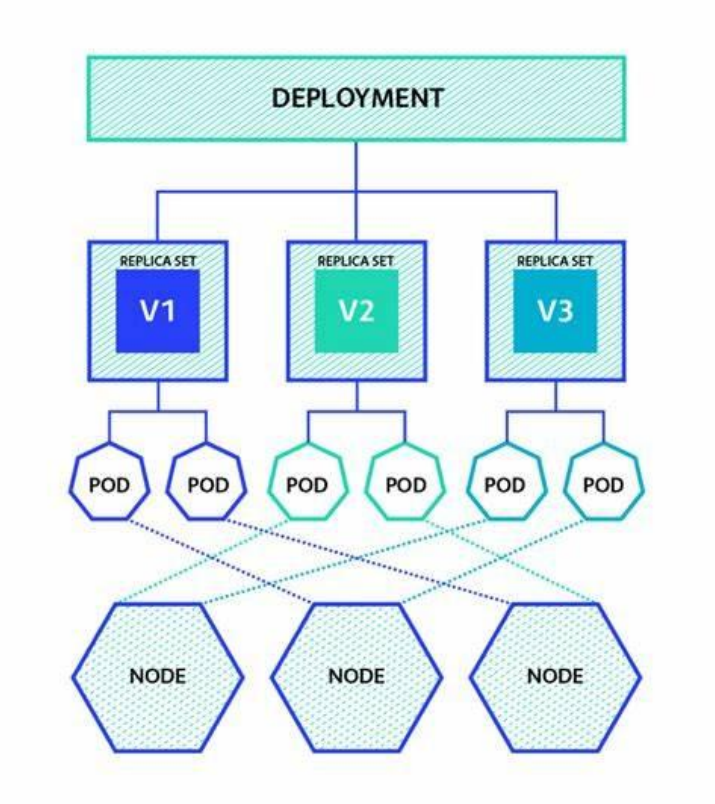
A pod is the smallest deployable unit in Kubernetes. It is a logical host for one or more containers and represents a single instance of a running process in a cluster. A pod can have one or more containers, and these containers share the same network namespace,

IP address, and hostname. Containers in a pod can communicate with each other using local hostnames and ports.

To create a pod in Kubernetes, you need to define a pod specification, which includes the container image, container port, and any other configuration required by the container. Here is an example of a simple pod specification:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx
    ports:
    - containerPort: 80
```

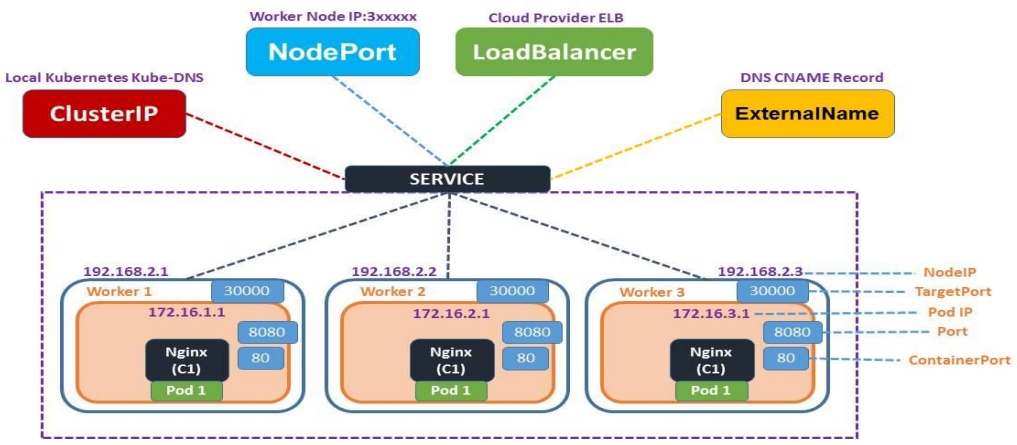
Kubernetes Deployments:



A deployment is a higher-level abstraction that manages a set of pods. It enables you to define the desired state of your application and automatically manages the creation, scaling, and updating of pods to match that state. A deployment includes a pod template that defines the pod specification used to create the pods.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx
        ports:
        - containerPort: 80
```

Kubernetes Services:



A service is an abstraction that enables network access to a set of pods. It provides a stable IP address and DNS name for a set of pods and enables load balancing across them. A service can be used to expose a deployment or a set of pods to the outside world.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
  selector:
    app: my-app
```