edquest

Day 8 Agenda

Orchestration with Kubernetes

- Recap
- Quiz for the previous Day
- K8s architecture: pods, deployments
- Deploying and managing model containers
- Minikube or local cluster deployment
- Workshop
- Quiz

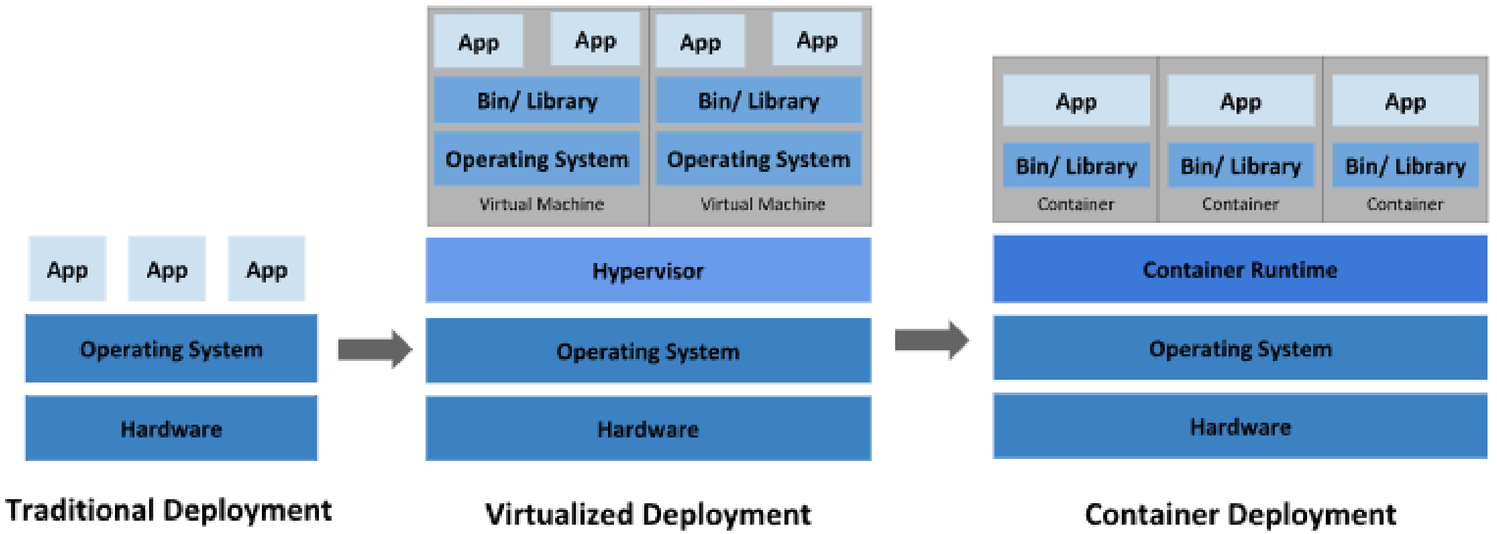
Kubernetes

edquest

Historical context for Kubernetes

edquest

Let's take a look at why Kubernetes is so useful by going back in time.



edquest

K8s architecture: pods, deployments

What is Kubernetes?

edquest

What is Kubernetes

Kubernetes (K8s) is an open-source container orchestration platform for automating deployment, scaling, and management of containerized applications.

edquest

Why you need Kubernetes and what it can do

- Containers are a great way to package and run applications, but managing them in production—ensuring uptime, scaling, and recovery—can be challenging.
- Kubernetes comes to the rescue! Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.

Kubernetes provides you with

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

edquest

edquest

Why Use Kubernetes?

- **Container Management** : Manage containers across multiple hosts.
- **Automation** : Automate rollouts, rollbacks, and scaling.
- **Cloud Native** : Works on-prem, in public cloud, or hybrid environments.
- **Self-Healing :** Auto-replaces failed containers, reschedules pods.
- **Scalability :** Horizontal scaling with ease.

Core Components Of Kubernetes

- **Pod :** Smallest deployable unit (group of containers).
- **Node :** Worker machine in the cluster.
- **Cluster :** Set of nodes running containerized applications.
- **Deployment :** Manages updates & replicas of pods.
- **Service :** Exposes a set of pods as a network service.

edquest

Control Plane Components

edquest

kube-apiserver

- The core component server that exposes the Kubernetes HTTP API.

etcd

- Consistent and highly-available key value store for all API server data.

kube-scheduler

- Looks for Pods not yet bound to a node, and assigns each Pod to a suitable node.

kube-controller-manager

- Runs controllers to implement Kubernetes API behavior.

cloud-controller-manager (optional)

- Integrates with underlying cloud provider(s)

Node Components

Run on every node, maintaining running pods and providing the Kubernetes runtime environment:

kubelet

- Ensures that Pods are running, including their containers.

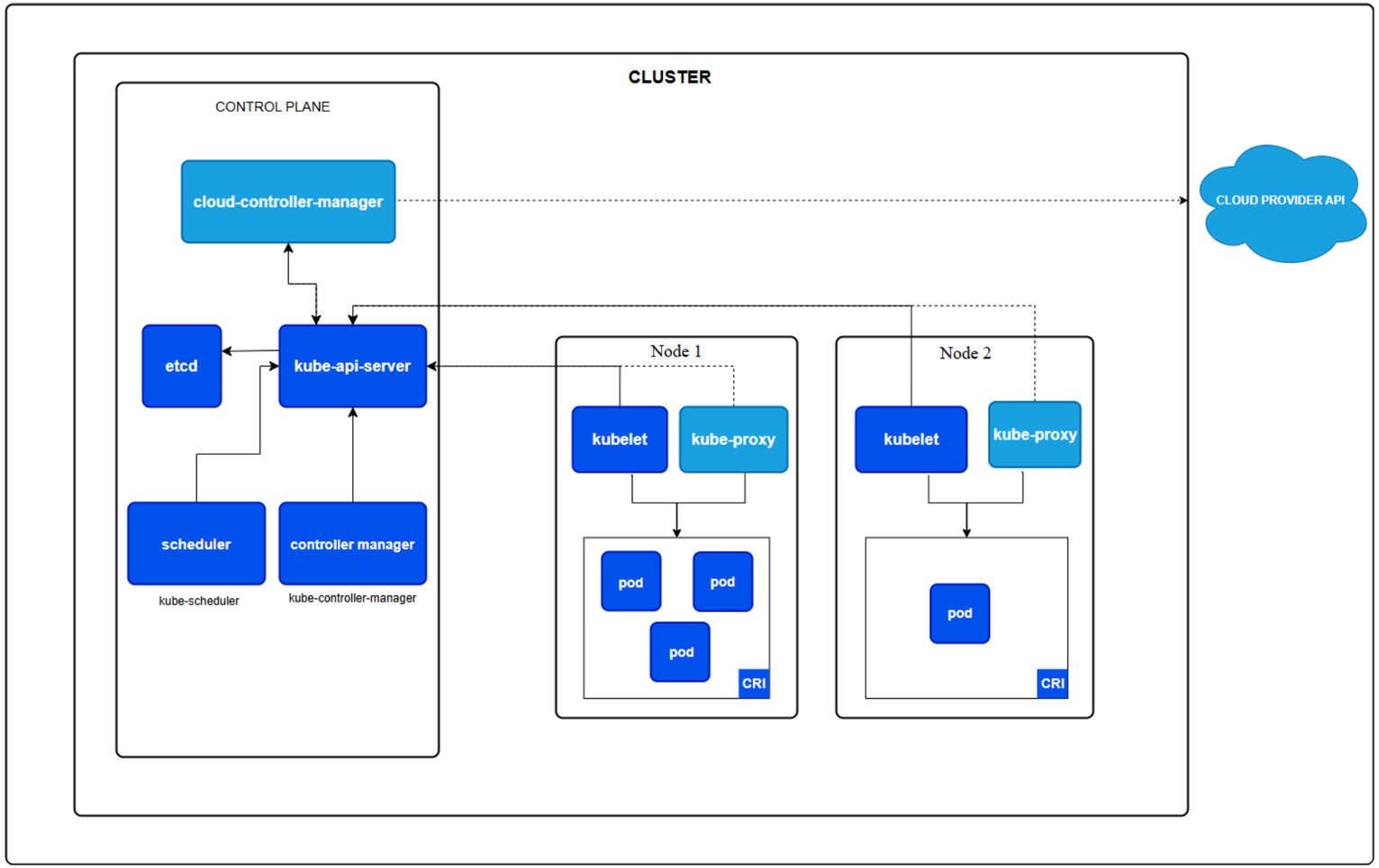
kube-proxy (optional)

- Maintains network rules on nodes to implement Services.

Container runtime

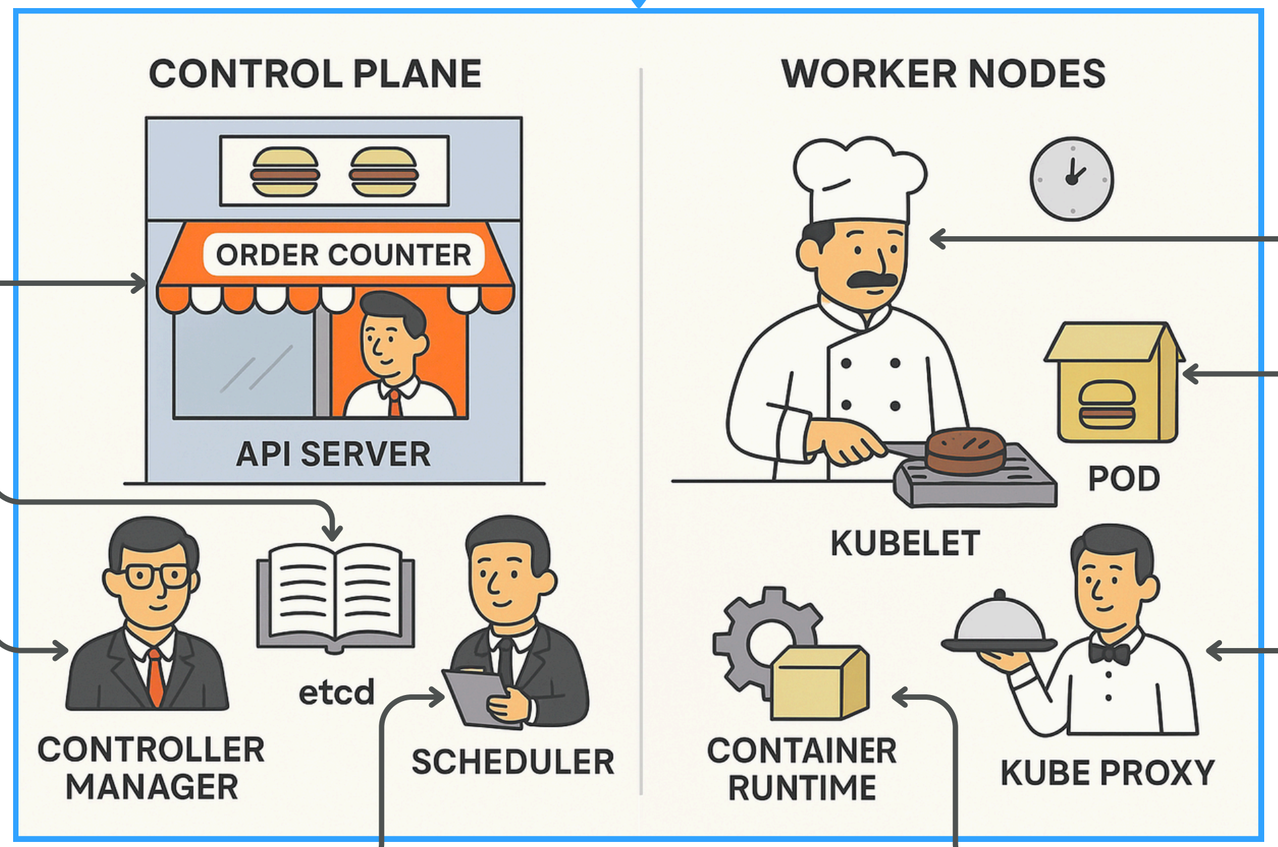
- Software responsible for running containers. Read Container Runtimes to learn more.

Kubernetes Architecture



A BURGER RESTAURANT: K8S CLUSTER

edquest



The Order Counter
(API Server)

Recipe Book + Whiteboard
(etcd)

Operations Supervisor
(Controller Manager)

Task Assigning Manager
(Scheduler)

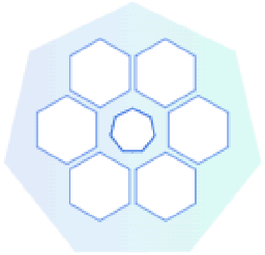
The Head Chef
(Kubelet)

Burger Orders
(Pods)

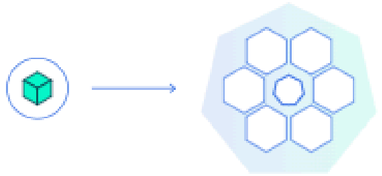
The Waiter
(Kube Proxy)

Cooking Tools
(Container Runtime)

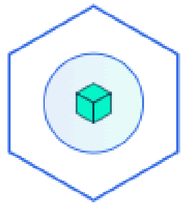
Kubernetes Basic Modules



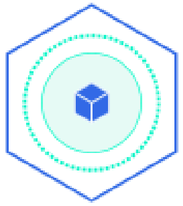
1. Create a Kubernetes cluster



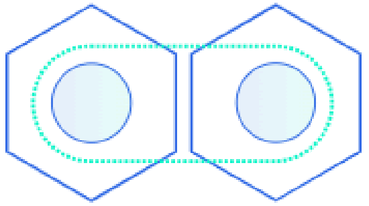
2. Deploy an app



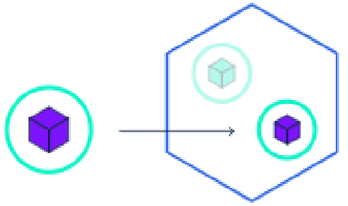
3. Explore your app



4. Expose your app publicly



5. Scale up your app



6. Update your app

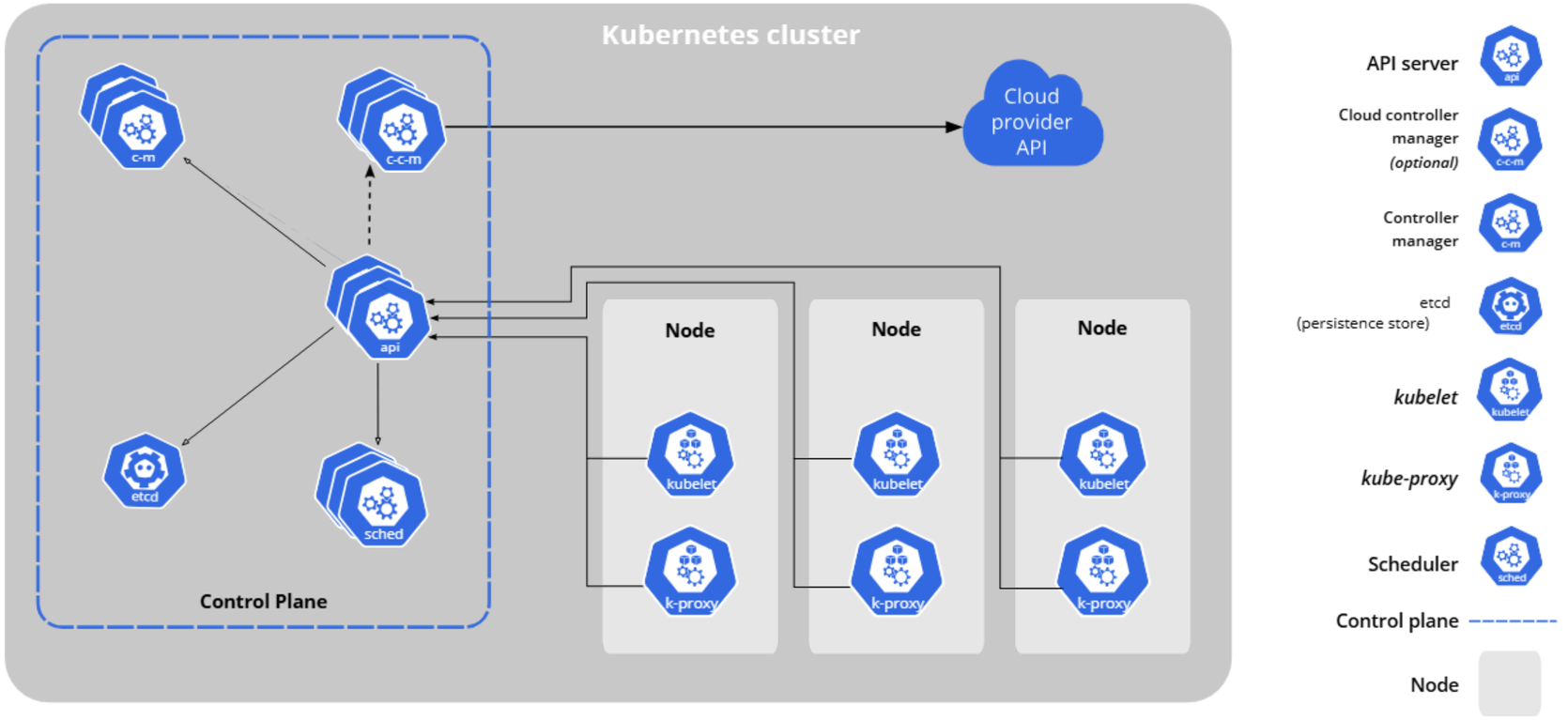
Using Minikube to Create a Cluster

edquest

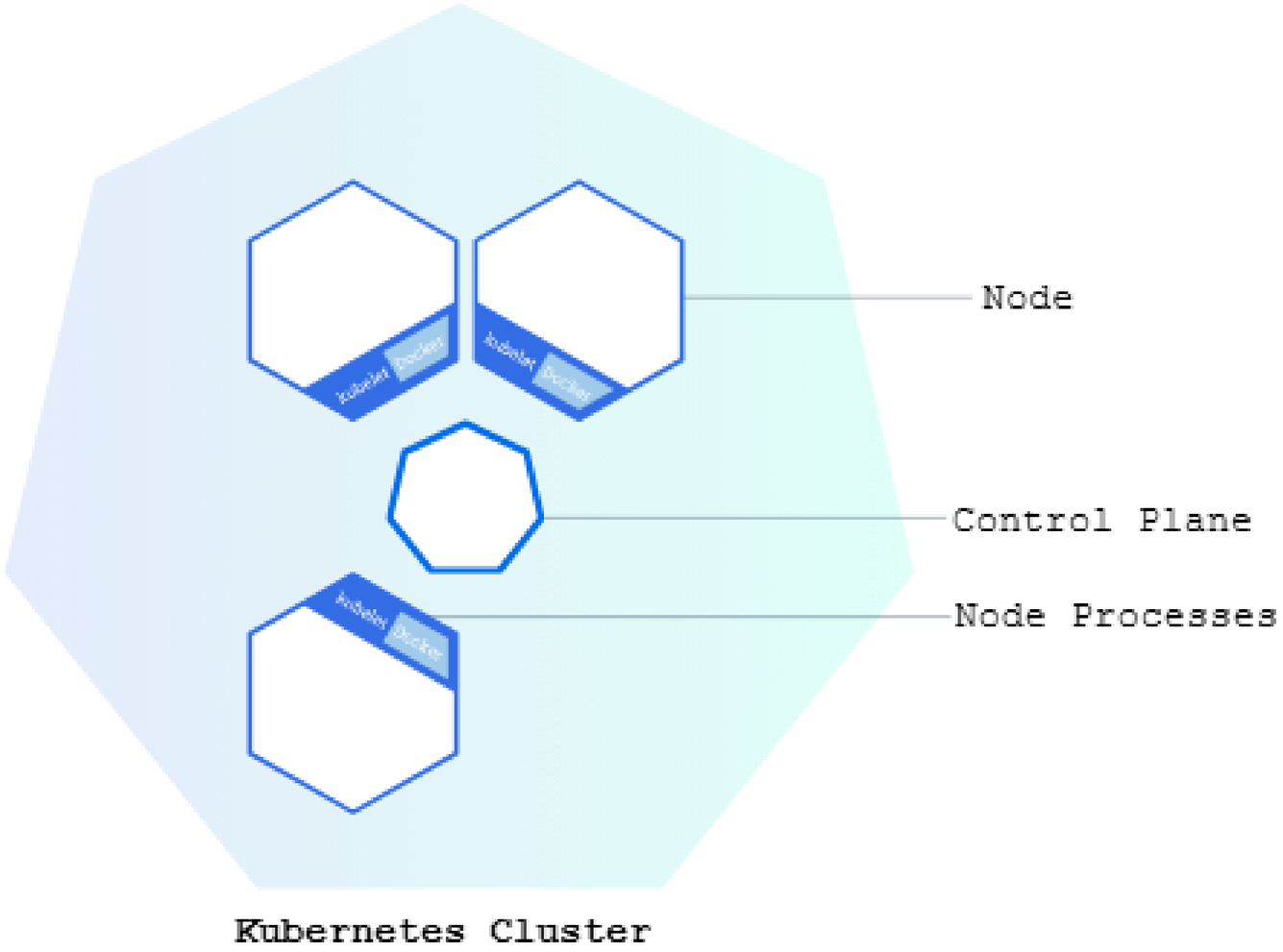
Kubernetes Cluster: Kubernetes is a production-grade, open-source platform that orchestrates the placement (scheduling) and execution of application containers within and across computer clusters..

- Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit

Kubernetes Cluster



Using Minikube to Create a Cluster



Initialize minikube K8s Cluster

Step-by-Step Installation of minikube

1. Install Chocolatey (if not already installed)
2. Install kubectl and Minikube
3. Start Minikube with Docker driver

`NOTE: complete installation step is in README.md`

edquest

Service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: ice-cream-service
spec:
  type: NodePort
  selector:
    app: ice-cream-api
  ports:
    - protocol: TCP
      port: 5001          # Exposes this port inside the cluster
      targetPort: 5001    # Matches containerPort in the deployment
      nodePort: 32001     # Exposes this port outside (optional; choose 30000-32767)
```

Deploying ML models on Kubernetes

edquest

What We'll Do

In this session, we’ll:

- Package our ML model into a Docker image
- Create Kubernetes YAML files for deployment
- Expose the model using a Kubernetes Service
- Test it via an endpoint

Deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ice-cream-api
  labels:
    app: ice-cream-api
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ice-cream-api
  template:
    metadata:
      labels:
        app: ice-cream-api
    spec:
      containers:
        - name: ice-cream-api
          image: edquest/icecream_api:latest
          command: ["python"]
          args:
            - app.py
          ports:
            - containerPort: 5001
          imagePullPolicy: IfNotPresent
```

edquest