What is the point of Kubernetes?

Kubernetes is defined as a container orchestration platform.

Containers are amazing: lightweight, portable, self-contained! However, there are a few questions, for example:

- *What happens if your container dies?*
- *What happens if the machine running your container fails?*
- *How do you enable networking between containers? Between containers on different hosts?*

To sum up, containers are not fault tolerant, and networking can easily become an entangled mess, especially with bigger deployments.
What is the point of Kubernetes?

Kubernetes is defined as a container orchestration platform.

The orchestration manages the entire lifecycle of individual containers, spinning up and shutting resources when needed. If a container shuts down unexpectedly, the orchestration will solve the problem by launching another container in its place.

It gives a dynamic route for applications to communicate with each other even as underlying containers are created, replicated and destroyed. Even when they run on different physical/virtual nodes.

Lastly, the orchestration selects a working node for each container automatically, while trying to balance the resource consumption and requirements of each individual container.
Design principles

Declarative - We define a desired state of our system and Kubernetes automation does it’s best to ensure that actual state of the system matches these desires.

Distributed - Kubernetes is a cluster of machines, with control plane and worker nodes. Kubernetes groups all the machines in it’s cluster into one unified management interface, so operators do not have to worry about particular machines.

Decoupled - Containerized applications naturally lead to the microservice architecture, and Kubernetes supports the idea of scalable and independently updatable resources.
Basic objects in K8s

Kubernetes objects are defined in YAML or JSON files, referred to as *manifests*.

**Pod** is the fundamental building block in K8s, comprised of one or more containers, a shared networking layer and shared file system volumes.
Deployments is an object that encompasses a collection of pods, defined by a template and a replica count (can be horizontally scaled). The cluster will try to have the pods available at all times.
Basic objects in K8s

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: serverpy-deploy
  namespace: testv0
spec:
  replicas: 1     <- How many Pods should be running?
  selector:
    matchLabels:
      app: server-py  <- What Pods belong to this Deployment?
  template:
    metadata:
      labels:
        app: server-py    <- A label identifying the Pod
    spec:
      containers:
        - name: docker-lab
          image: docker-lab:2.0.0
          imagePullPolicy: Never
          ports:
            - containerPort: 5000
```
Basic objects in K8s

Each Pod in K8s has a unique IP, similar to Docker.

A service provides you with a stable endpoint which directs traffic to the desired Pods even as the exact Pods are undergoing changes due to updates, scaling and failures.

```yaml
apiVersion: v1
class: Service
metadata:
  name: example
  namespace: testv0
spec:
  ports:
  - port: 80
    targetPort: 5000
    name: tcp
  selector:
    app: server-py

[root@b123123 centos]# kubectl get service -n testv0
NAME       TYPE     CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
example    ClusterIP  10.43.89.196  <none>        80/TCP         78m
```
Basic objects in K8s
Basic objects in K8s

While a Service allows us to expose application behind a stable endpoint, it is only available to internal cluster traffic, or via a host port.

**Ingress** allows to expose our application to external traffic, while being aware of all the HTTP layer benefits, like logging, certificates, domain names etc.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: default-backend-ingress
  namespace: testv0
  annotations:
    kubernetes.io/ingress.class: "traefik"

spec:
  defaultBackend:
    service:
      name: serverpy-deploy
      port:
        number: 80
```
Basic objects in K8s
K8s control plane

Kubernetes API server governs how workloads are executed, monitored and maintained on the cluster.

It runs on the 6443/tcp port of each control plane node.

You can directly interact with the API server with either the kubectl command line tool, or other tools like Lens
This lab

We will get an introduction into Kubernetes:

- Understand the Basic components and Principles
- Understand the principles of Deployments and compare them with Pods
- Deploy a pod and deployment
- Scale a deployment
- Deploy Service and Ingress to expose your deployments