Kubernetes Introduction

What is Kubernetes?

According to their own website, Kubernetes is an

> **open-source system for automating deployment, scaling, and management of containerized applications.**

This means that Kubernetes is a collection of tools / services and concepts that help you **deploy containerized applications** - typically across multiple hosts (i.e. multiple remote machines).

Kubernetes **simplifies** the deployment and configuration of complex containerized applications and it helps with topics like scaling and load balancing.

Services like AWS ECS also help with that but of course you **have to follow the AWS-specific rules**, syntax and logic / concepts for that. To a certain extent, you are "locked in" - if you want to switch to a different provider / host, you have to "translate" all your deployment configs etc.

With Kubernetes, you can set up a configuration (following the Kubernetes rules and concepts) which will **work on any host** that supports Kubernetes - no matter if it's a cloud provider or your own, Kubernetes-configured data center.

How does Kubernetes work?

We'll dive into the specific and concret examples later (in the next section), but generally, you can **run a couple of commands against a Kubernetes Cluster** (a network of machines which are configured to support Kubernetes) to then deploy and start Containers.

Typically, you will write down **Kubernetes configuration files which describe your target state** - with a couple of Kubernetes commands, you can then bring that state to life on your cluster.

Here's an example configuration file (*you don't need to understand it yet - we'll dive into that later*):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: users-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: users
  template:
    metadata:
      labels:
```

```
        app: nginx
    spec:
      containers:
        - name: users
          image: my-repo/users-application
```

Why Kubernetes?

If your projects and hence your deployments become **more complex** (multiple Containers, scaling, load balancing), you probably don't want to run and monitor all your containers on your remote machines manually.

You might not want to do this for various reasons - also see the "Deployment" section, earlier in the course.

Kubernetes makes setting up and configuring complex deployments easy. It will also automatically monitor your containers and restart them if they go down. It makes scaling and load balancing easy (as it's built-in).

Managing data in **volumes** across multiple machines is also easy to set up. And much more.

For all these reasons, you might want to use Kubernetes.

As mentioned above already, you **could also use any other managed Container service** (e.g. AWS ECS) but you would then be kind of "**locked in**". With Kubernetes, you indeed use a open-source "tool" and you can use your Kubernetes configuration on any machine and any provider which supports Kubernetes.

What Kubernetes is NOT

It's important to understand that Kubernetes is **NOT** one of the following things:

- a cloud provider
- a cloud provider service (though cloud provider might offer Kubernetes-specific services)
- a tool or service that manages infrastructure - Kubernetes will NOT create and launch any machines or do anything like that (managed Kubernetes services by cloud providers might do that)
- a single tool or software - Kubernetes is a collection of concepts and tools (see below)

Core Concepts

For visuals, check out the **slides which you also find attached to the course sections** (always on the last lecture of each module).

Kubernetes is a collection of concepts and tools.

Specifically, a **Kubernetes Cluster** is required to run your Containers on. A Kubernetes Cluster is simply a **network of machines**.

These machines are called "**Nodes**" in the Kubernetes world - and there are two kinds of Nodes:

- The **Master Node**: Hosts the "**Control Plane**" - i.e. it's the control center which manages your deployed resources
- **Worker Nodes**: Machines where the actual Containers are running on

The Master Node hosts a couple of "tools" / processes:

- An **API Server**: Responsible for communicating with the Worker Nodes (e.g. to launch a new Container)
- A **Scheduler**: Responsible for managing the Containers, e.g. determine on which Node to launch a new Container
- A couple of other things, also see the official docs

On Worker Nodes, we got the following running "tools" / processes:

- **kubelet service**: The counterpart for the Master Node API Server, communicates with the Control Plane
- **Container runtime (e.g. Docker)**: Used for actually running and controlling the Containers
- **kube-proxy service**: Responsible for Container network (and Cluster) communication and access

If you create your own Kubernetes Cluster from scratch, you need to **create all these machines and then install the Kubernetes software** on those machines - of course you also need to manage permissions etc.

Unless you have tons of Linux and server administration experience, you might not want to do that - thankfully, there are tools and managed services which simplify the creation of a Kubernetes Cluster (learn more in the "Kubernetes Deployment" section).

Once the Cluster is up and running, Kubernetes will create, run, stop and manage Containers for you.

It does that in so-called "**Pods**" - which are the smalles unit in the Kubernetes world.

A **Pod** simply contains **one or more Container** (typically one though) and any configuration as well as volumes required by the Container(s).

With Kubernetes, you don't manage Containers but rather Pods which then manage the Containers - more on that in the next course section.

Further Resources

Official website & docs: https://kubernetes.io/docs/home/