English communication

By Mathieu Perochon

2nd lesson : Kubernetes

What is Kubernetes ?

Please, take a few minutes to read this blog post :

https://medium.com/platformer-blog/benefits-of-kubernetes-e6d5de39bc48#:~:text=In%20Kubernetes%2C%20applications%20can%20be,as%20less%20machines%20are%20used.

Architecture

⇒ **Nodes :** The machines on which a cluster is running can either be Masters or Nodes. The naming makes sense. The Master is the control panel of the whole cluster. All commands we will run will be run on the Master instance. It will then decide which Node, or worker machine, in the cluster will take the workload.

⇒ **Network :** In order to enable the communication between the various containers in the cluster we need a network to provide IP addresses to them. Luckily, Kubernetes has a wide variety to choose from, and thankfully they work like magic. Here's a more detailed explanation. In all essence, such a network enables the pods in a cluster to talk to each other.

Architecture

⇒ **Interaction :** every Node has a Kubelet, which is an agent for managing the Node and communicating with the Kubernetes Master. All of this is glued together with the Kubernetes API which the Master exposes. We then use the Kubernetes API to directly interact with the cluster using a CLI tool called kubectl. More about this further a bit down.

Apart from the API, what enables Kubernetes to work properly is a globally available configuration store called etcd. It's a distributed key-value store that can be distributed across multiple nodes. Why is etcd so important? It stores configuration data for all of the nodes in the cluster so each and every part of it knows how to configure itself.

Architecture

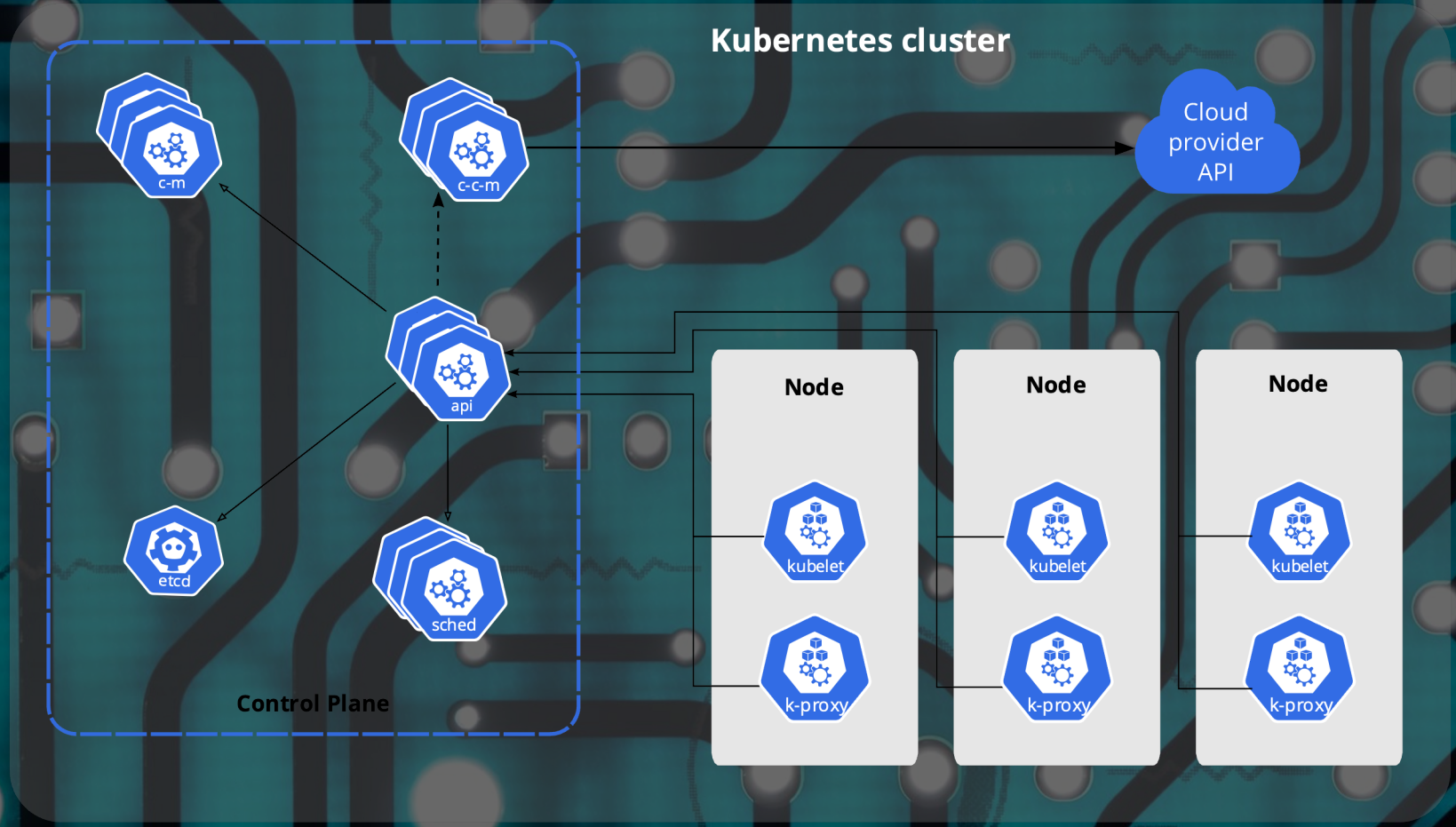
⇒ **Pod :** a group of one or more containers (such as Docker containers), with shared storage/network, and a specification for how to run the containers. Even if the pod has several containers, they will all be reachable in the network through a single IP address.

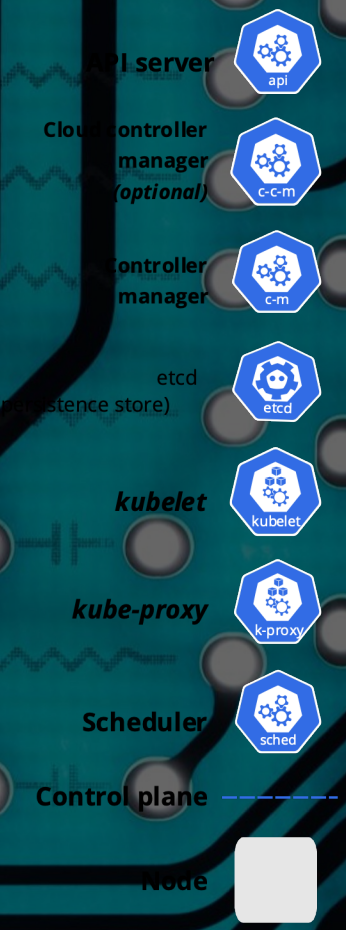
⇒ **Service :** an abstraction which defines a logical set of Pods and a policy by which to access them. Pods have a life cycle.

⇒ **ReplicaSet :** give Pods a label and control their replication.

⇒ **Deployment :** describes the desired state and makes sure to change the actual state to the desired state if needed. A deployment manages Pods and ReplicaSets.

architecture





Documentation

⇒ https://docs.docker.com/desktop/kubernetes/

⇒ https://kubernetes.io/docs/home/

⇒ https://kubernetes.io/docs/reference/kubectl/overview/

Let’s practice now !

Mission 1 : First step with Kubernetes

Steps :

1 – Install Docker Desktop for Windows

2 – Enable Kubernetes : https://docs.docker.com/desktop/kubernetes/

3 – Deploy the kubernetes Dashboard : https://collabnix.com/kubernetes-dashboard-on-docker-desktop-for-windows-2-0-0-3-in-2-minutes/

4– Access to the Dashboard

Practice now !

Mission 2 : Deploy your first application

Steps :

1 – update the A2SR repository (git pull...)

2 – Go to the mission2 directory and deploy the pod.yaml file on your kubernetes cluster.

3 – Check if the pod is up and running

4 - Check the logs of the pod, you should be able to see something like this :

64 bytes from 8.8.8.8: seq=0 ttl=37 time=21.393 ms

5 – Delete the pod

Practice now !

Mission 3 : Deploy your multi-container application

GOAL : In the last class, you created a docker compose file to deploy an application and its database. Now you have to do the same thing, but on Kubernetes.

Steps :

1- Create all the files required for deployment

2- Run the application on your cluster

3- Publish the code on your github repository

4- Write a report and send it with the github link to mperochon@gmail.com