Cloud & Automation Class

Lab 1 · Minikube (I)

Mouna Rouini · Meriem Cherif · Iskander Soltane · Adam Lahbib

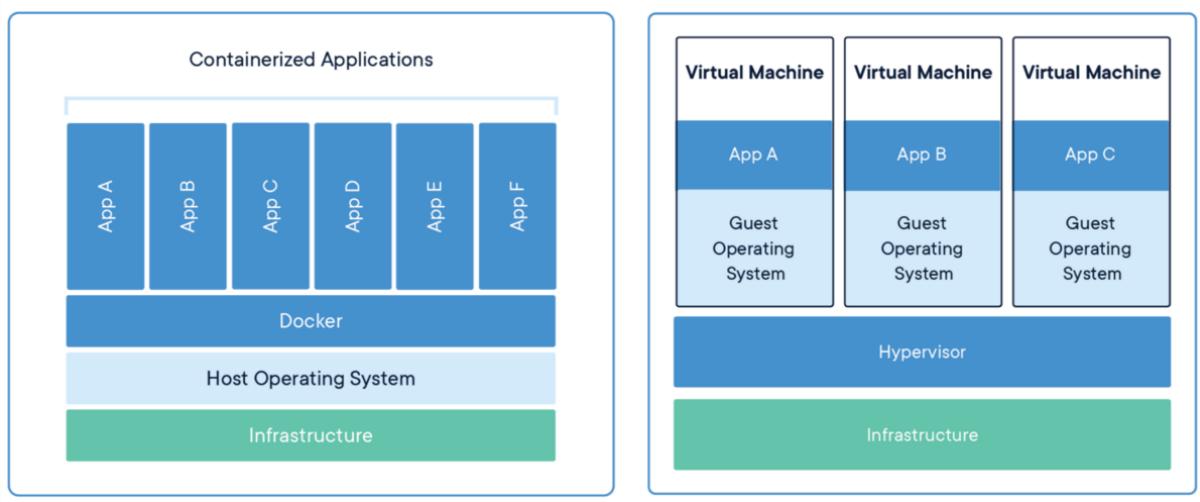
Feb. 22, 2023 @ INSAT

Contents

1 Introduction

1.1 Docker

Docker is a platform that allows developers to build, package, and distribute applications as containers. Containers are a lightweight and portable way to package and run software, isolating applications from the underlying system and other applications running on the same system. Docker provides tools to create and manage containers, making it easier to deploy applications across different environments and infrastructures.



1.1.1 Installing Docker Engine, containerd, and Docker Compose

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
↪  /etc/apt/keyrings/docker.gpg
```

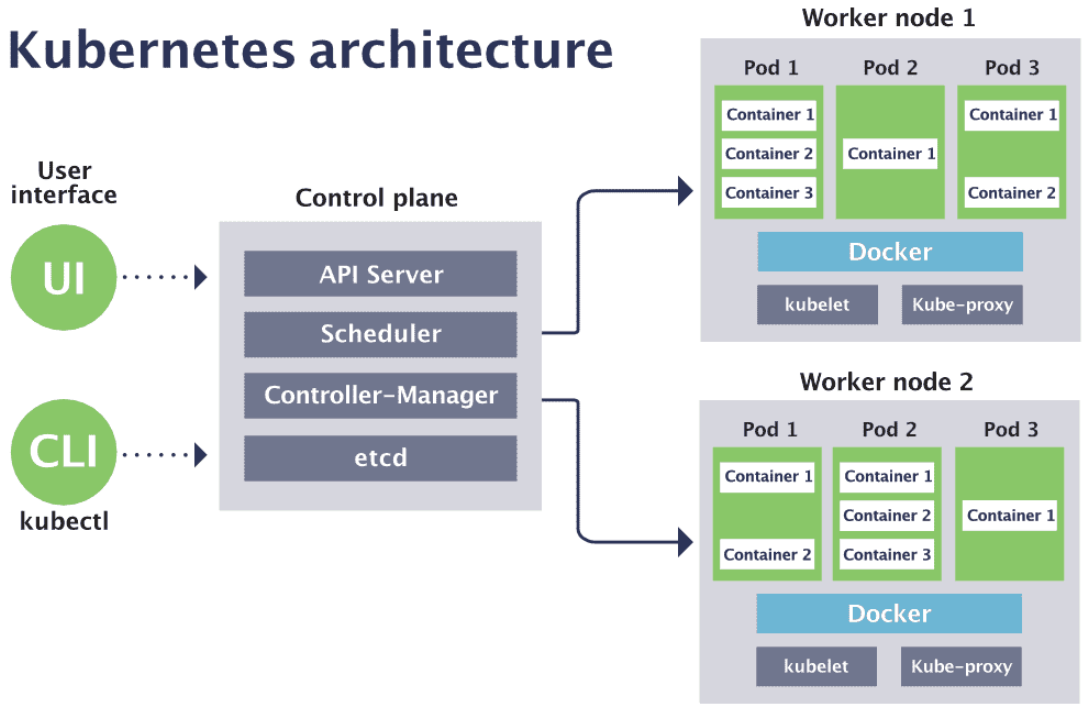
```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
  ↪  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
↪  docker-compose-plugin
```

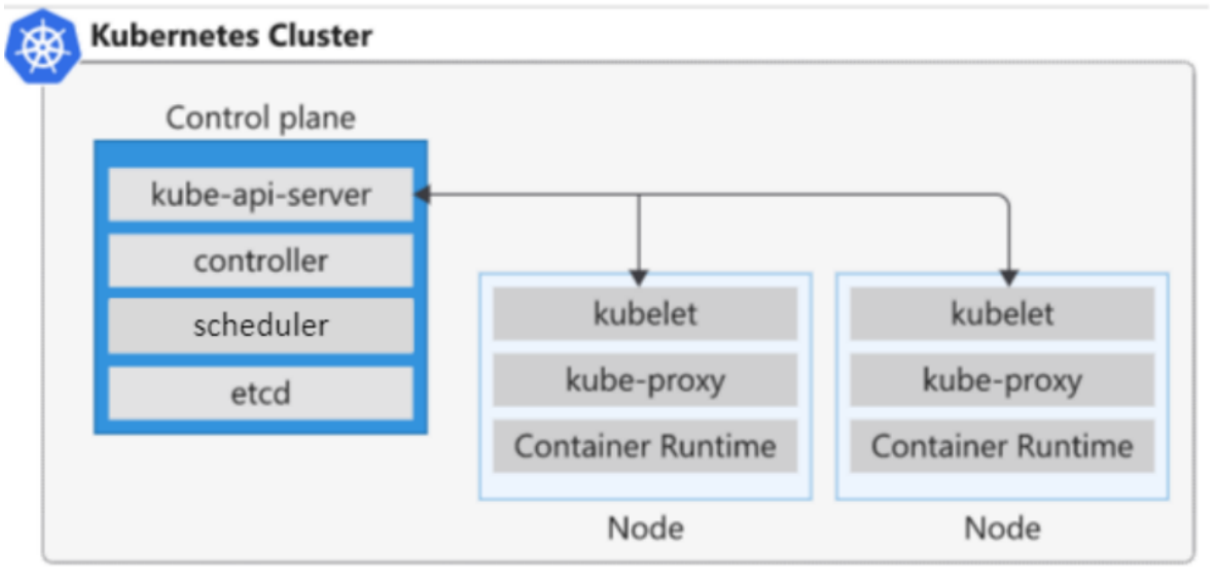
1.2 Kubernetes

Kubernetes, on the other hand, is a container orchestration platform. It helps manage and deploy containers across a cluster of nodes, providing features like automated scaling, load balancing, and self-healing. Kubernetes allows developers to define their application requirements in a declarative way, and it automatically handles the scheduling and management of containerized applications.



1.2.1 Kubernetes Cluster

A Kubernetes cluster consists of one or more master nodes and one or more worker nodes. The master nodes are responsible for managing the cluster, while the worker nodes are responsible for running the applications.



1.2.1.1 Control Plane

1.2.1.1.1 API Server The API server is the front-end for the Kubernetes control plane. It exposes the Kubernetes API, which is used to interact with the cluster. The API server is the only Kubernetes component that talks directly to the master nodes.

1.2.1.1.2 Scheduler The scheduler is responsible for scheduling pods to worker nodes. It watches for newly created pods with no assigned node, and selects a node for them to run on. The scheduler takes into account factors like resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

1.2.1.1.3 Controller Manager The controller manager is a daemon that embeds the core control loops shipped with Kubernetes. It watches the state of the cluster, and uses the Kubernetes API to make changes (create, update, delete) as needed to move the current state towards the desired state. The controller manager runs multiple control loops, including the node controller, replication controller, endpoints controller, service account and token controllers, and the garbage collector.

1.2.1.1.4 etcd etcd is a distributed key-value store that provides a reliable way to store data across a cluster of machines. It's used by Kubernetes as a backing store for all cluster data.

1.2.1.2 Worker Nodes

1.2.1.2.1 Kubelet The kubelet is an agent that runs on each worker node in the cluster. It makes sure that containers are running in a pod. More specifically, it ensures that the containers described in the pod's PodSpec are running and healthy. If a container fails, the kubelet restarts it. If the pod is deleted, the kubelet deletes all of its containers and restarts the pod.

1.2.1.2.2 Container Runtime The container runtime is the software that is responsible for running containers. Kubernetes supports several container runtimes: Docker, rkt, and containerd.

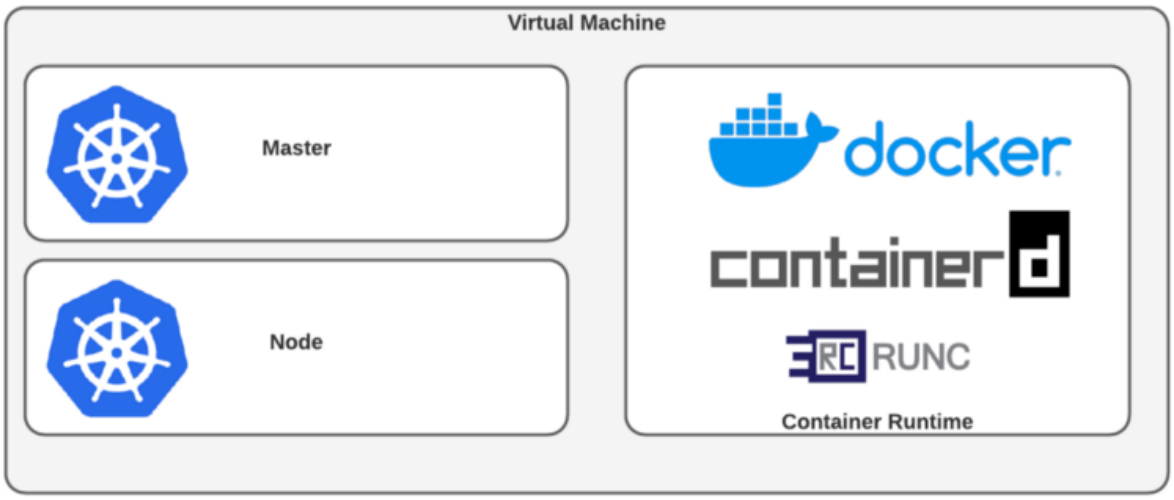
1.2.1.2.3 Kube-proxy The kube-proxy is a network proxy that runs on each worker node in the cluster. It maintains network rules on the nodes and implements connection forwarding across a cluster.

For more, feel free to check the theoretical document alongside this lab.

1.3 MiniKube

Minikube is a tool that enables you to run a single-node Kubernetes cluster on your local machine. It's designed to make it easy for developers to experiment with Kubernetes without having to set up a full-scale production environment. With Minikube, you can quickly spin up a Kubernetes cluster, test your applications, and experiment with different configurations.

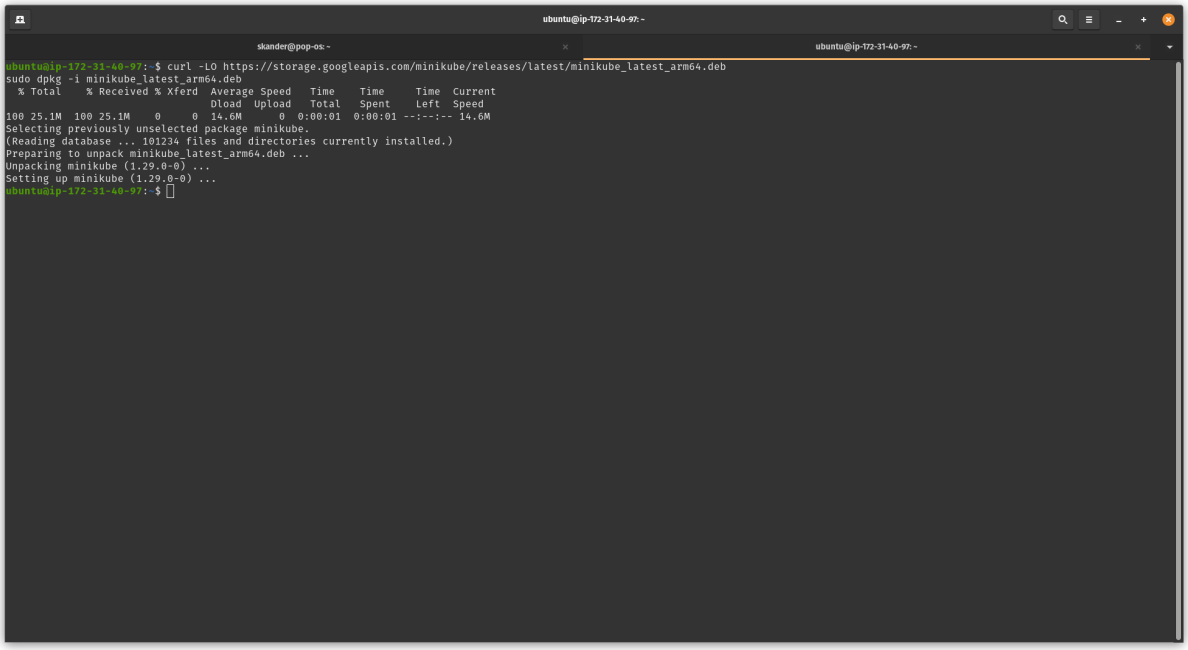
Minikube works by running a lightweight virtual machine on your local machine, which runs a single-node Kubernetes cluster. You can use the kubectl command-line tool to interact with the cluster, just like you would with a larger Kubernetes deployment. Minikube also provides features like automatic node restarts, load balancing, and persistent storage, making it easier to develop and test your applications.



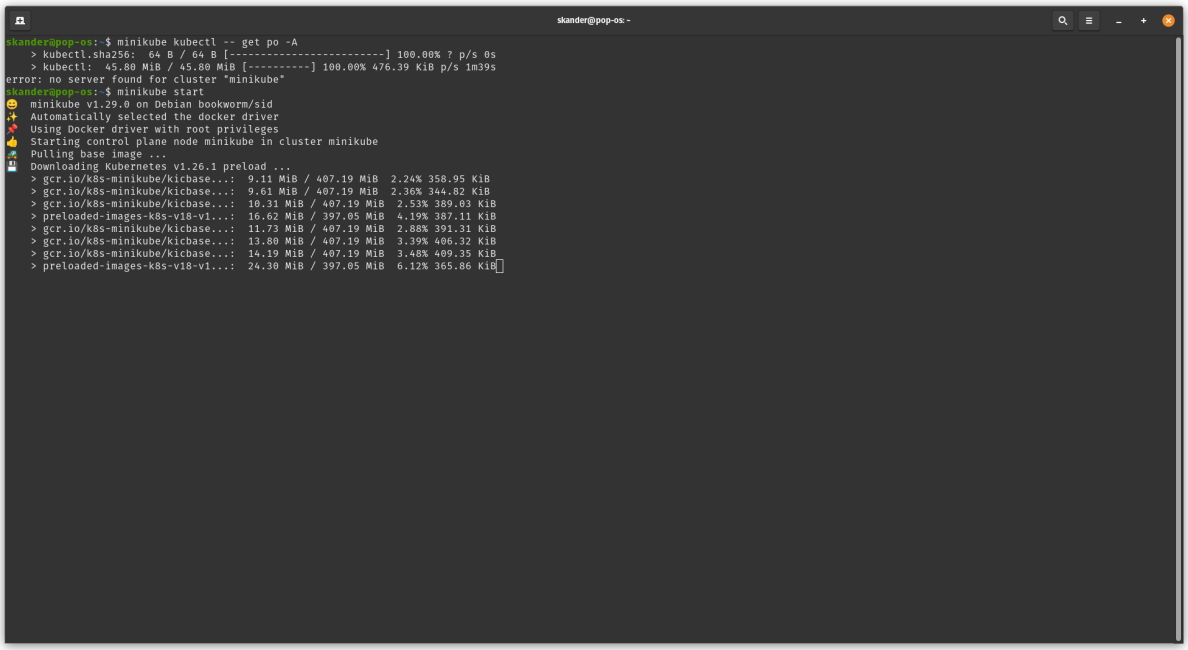
1.3.1 Installation

To install the latest minikube stable release on x86-64 Linux using binary download, run the following commands:

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



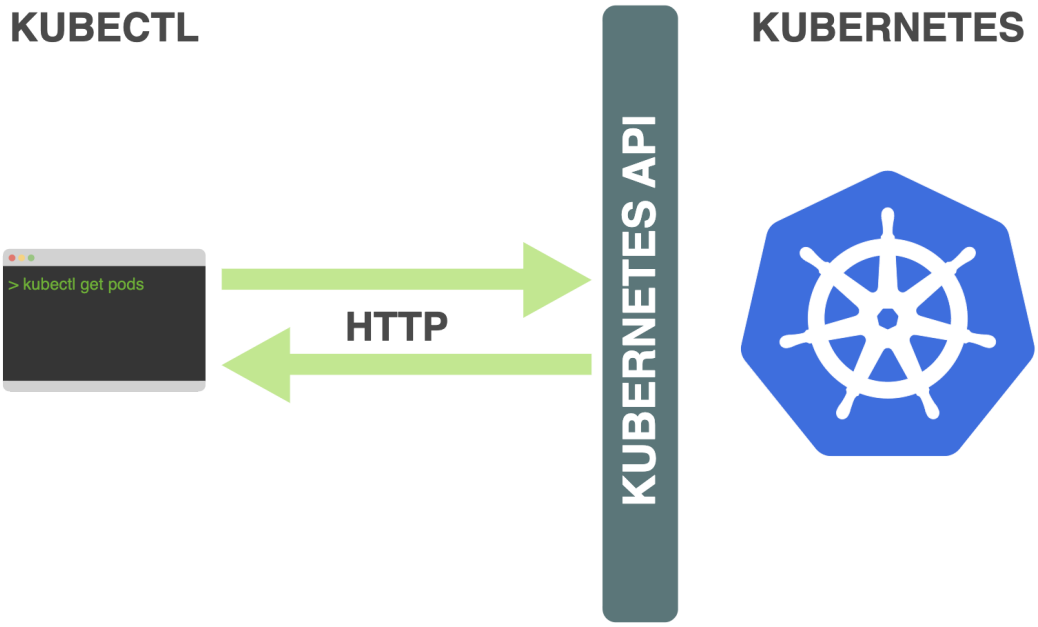
We wait for the installation to finish, and then we can check the version of minikube:



1.4 Kubectl (“kube-cuttle”, Canonical hate that)

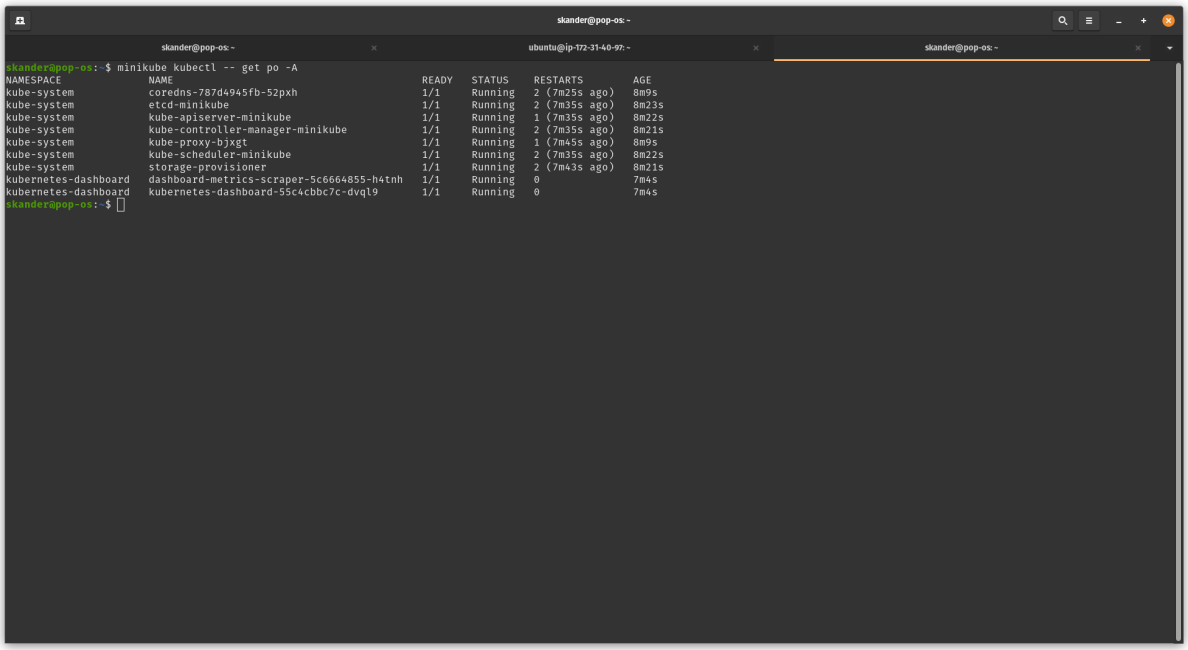
Kubectl is the command-line tool used to interact with Kubernetes clusters. It allows you to deploy, inspect, and manage applications running on a Kubernetes cluster. With kubectl, you can create and

update deployments, services, and other Kubernetes resources, as well as check the status of pods, nodes, and clusters.

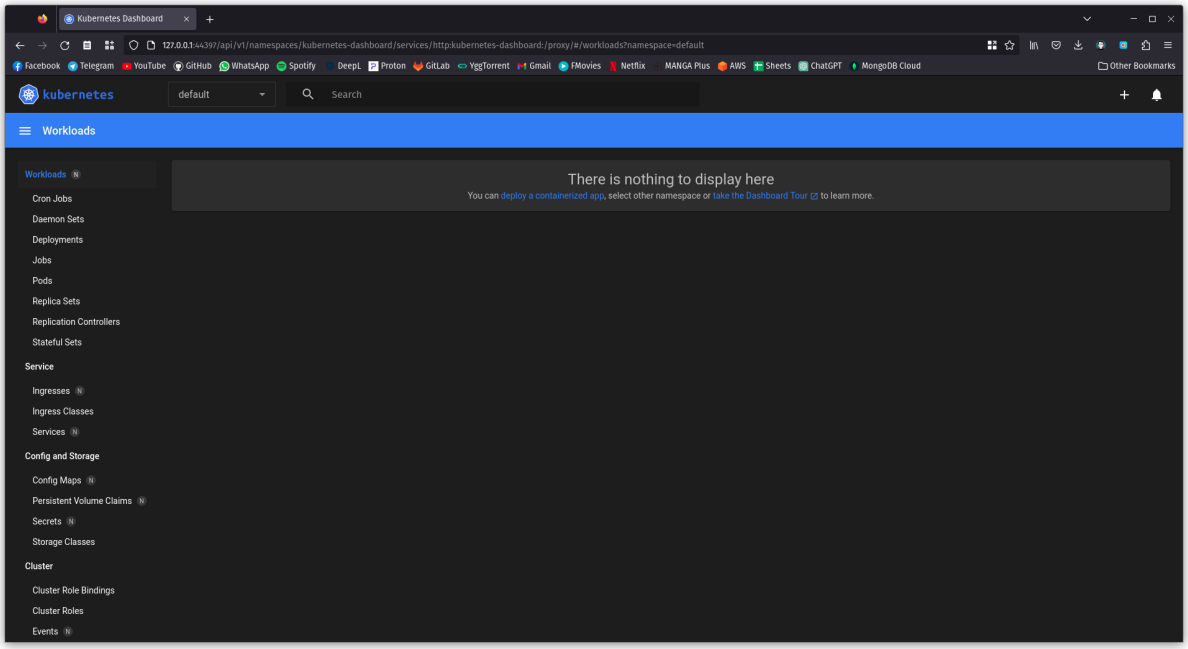


1.4.1 Installation

We will use the minikube's kubectl version, so we don't have to install it separately. We did an alias to make it easier to use.



A look at the minikube dashboard using `minikube dashboard`:



We still have nothing running…

1.4.2 Commands

`kubectl [command] [type] [name] [flags]`

- command - The operation you want to perform on one or more resources, for example create, get, describe, delete, apply.
- type - The resource type. The following resource types are supported: pods (po), services (svc), deployments (deploy), replicasets (rs), replicationcontrollers (rc), nodes (no), events (ev), configmaps (cm), and secrets.
- name - The name of the resource. Names are case-sensitive. If the name is omitted, details for all resources are displayed, for example `kubectl get pods`.
- flags - Optional flags. For example, you can use the -o flag to specify the output format.

An example of a command using kubectl is:

```
kubectl get pods hello-world -o yaml
```

This command will make a call to the Kubernetes API and out the configuration for a pod named hello-world in YAML format.

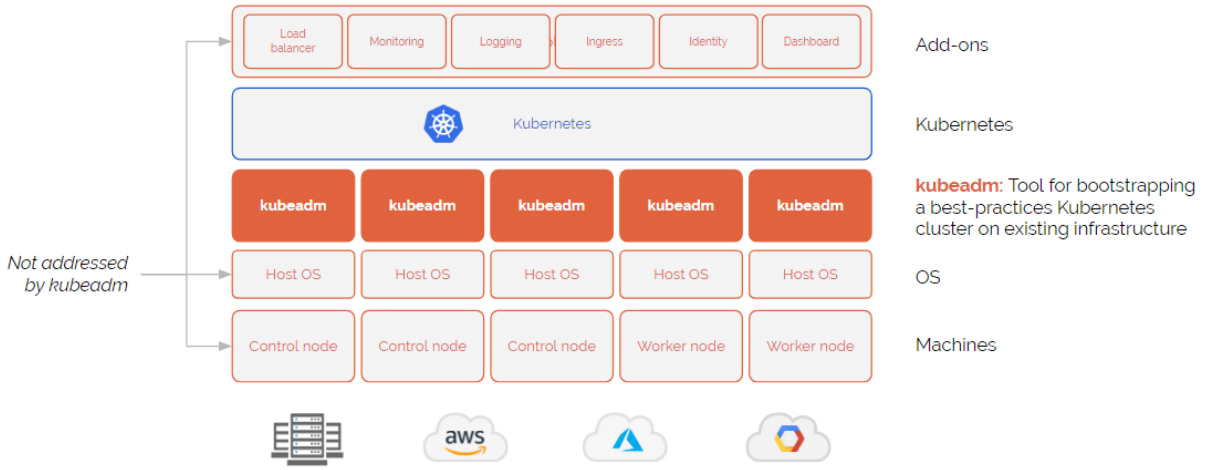
A way to tell Kubernetes to apply a configuration file is to use the apply command:

```
kubectl apply -f hello-world.yaml
```

For more, feel free to check the theoretical document alongside this lab. we explained pods, deployments, replicasets, and statefulsets

1.5 Kubeadm

Kubeadm is a tool used to set up and manage Kubernetes clusters. It automates many of the steps required to create a new cluster, including creating the control plane and joining worker nodes to the cluster. With kubeadm, you can easily set up a new Kubernetes cluster from scratch, or upgrade an existing cluster to a new version of Kubernetes.



1.5.1 Commands

`kubeadm [command] [flags]`

- command - The operation you want to perform on the cluster, for example init, join, or upgrade.
- flags - Optional flags. For example, you can use the –config flag to specify a configuration file.

1.5.2 Application

Let's create a Kubernetes cluster using kubeadm. First, we need to install kubeadm on our machine. You can install it using the following command:

1.5.2.1 Install kubeadm

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a
↪  /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

1.5.2.2 Initialize the cluster

First, create a control plane node by running the following command on the node you want to use as the control plane:

```
sudo kubeadm init --pod-network-cidr=<desired-pod-network-cidr>
```

The –pod-network-cidr flag is used to specify the CIDR range used by the pod network. This flag is required for the cluster to work properly. You can use any CIDR range that is not already in use on your network.

This command will initialize the control plane and generate a kubeadm join command that you'll need to use to join worker nodes to the cluster.

1.5.2.3 Configure kubectl

After the control plane node is initialized, you'll need to configure kubectl to connect to the cluster. Run the following commands on the control plane node:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

1.5.2.4 Install a pod network

Kubernetes uses a pod network to allow pods to communicate with each other. You must install a pod network add-on master-side so that your pods can communicate across nodes. One popular option is Calico.

```
kubectl apply -f https://docs.projectcalico.org/v3.19/manifests/calico.yaml
```

1.5.2.5 Join worker nodes

To join a worker node to the cluster, run the kubeadm join command that was generated when you initialized the control plane node. This command must be run as root.

```
sudo kubeadm join <control-plane-host>:<control-plane-port> --token <token>
↪  --discovery-token-ca-cert-hash <hash>
```

Replace and with the hostname and port of the control plane node, and replace and with the values generated by the kubeadm init command in step 1.

1.5.2.6 Verify the cluster

To verify that the cluster is working, run the following command on the control plane node:

```
kubectl get nodes
```

2 Application Deployment

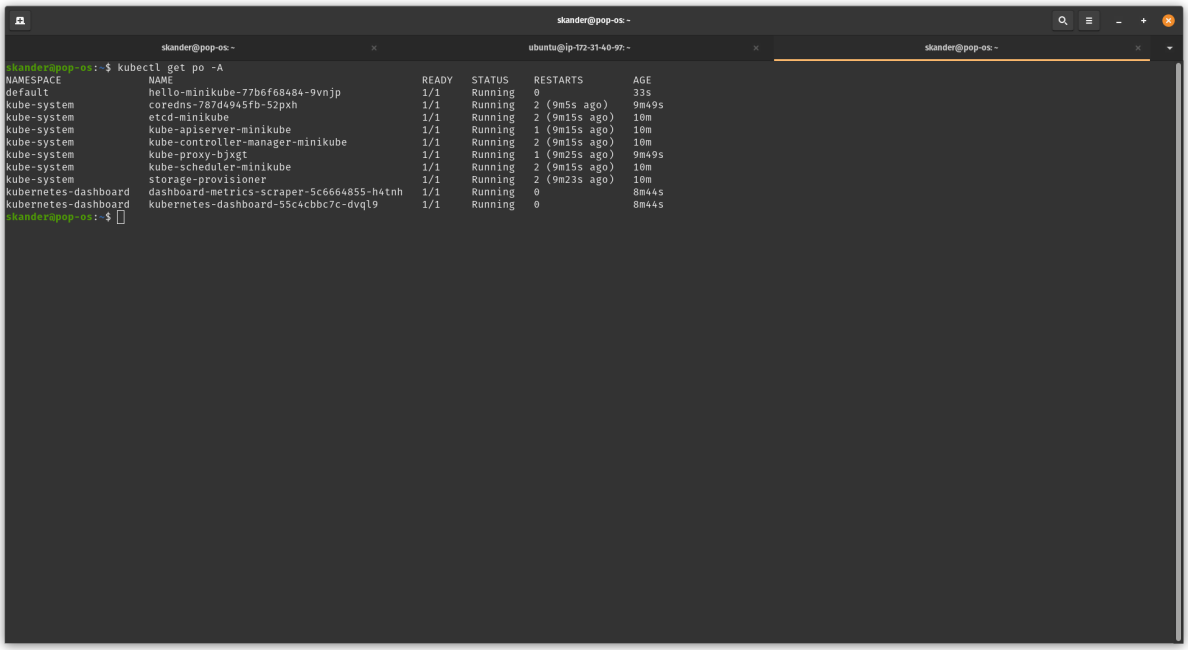
2.1 Hello-MiniKube

Let's continue where we left off, we had the usual stuff in our cluster, let's add an application.

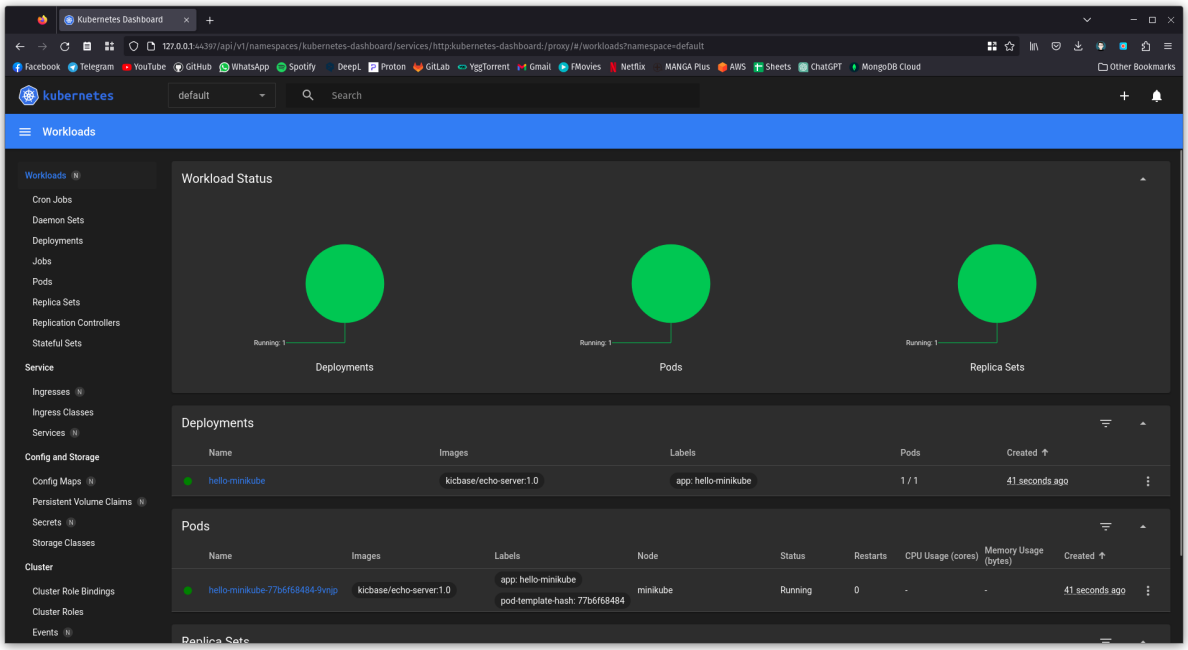
```
kubectl create deployment hello-minikube --image=kicbase/echo-server:1.0
```

This command will create a deployment named hello-minikube with a single replica. The deployment will create a pod that runs the echo-server container. The echo-server container will listen on port 8080 and echo back anything you send to it. You can verify that the pod is running by running the following command:

```
kubectl get pods
```



- Minikube Dashboard:



2.2 Exposing the application

To access the echo-server from outside the Kubernetes cluster, you must expose the pod as a Kubernetes Service. A Kubernetes Service is an abstraction that defines a logical set of pods and a policy by which to access them.

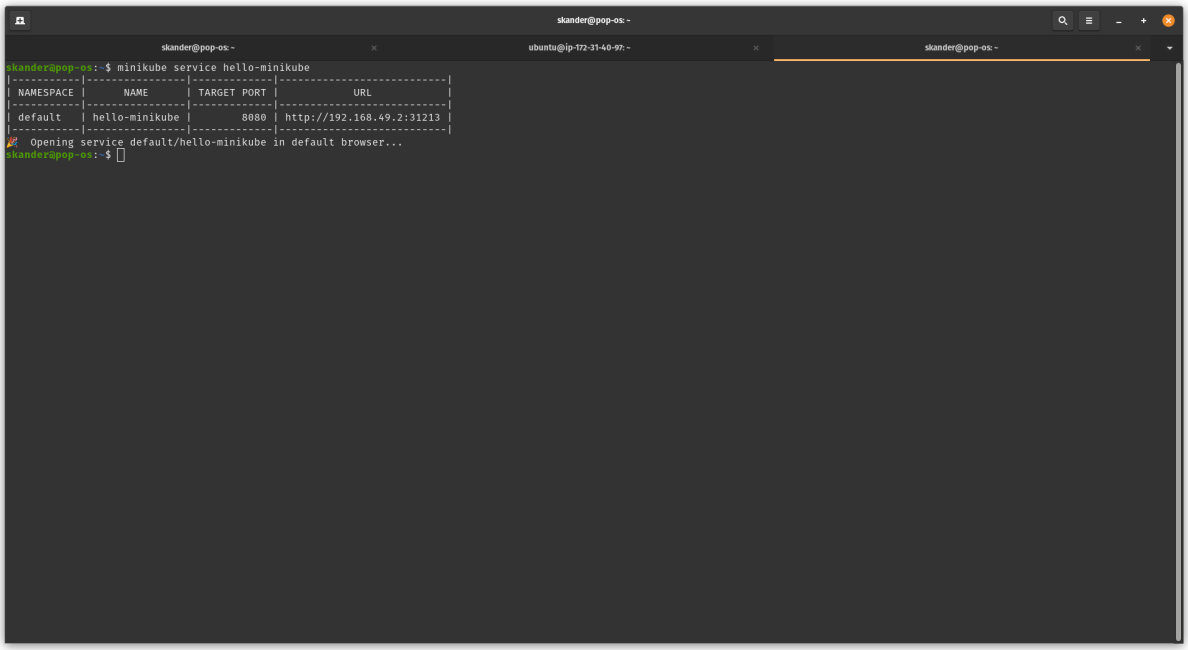
```
kubectl expose deployment hello-minikube --type=NodePort --port=8080
```

This command will create a service named hello-minikube that exposes the deployment on port 8080 using the NodePort service type. The NodePort service type exposes the service on each node's IP at a static port (the NodePort). To determine the port, run the following command:

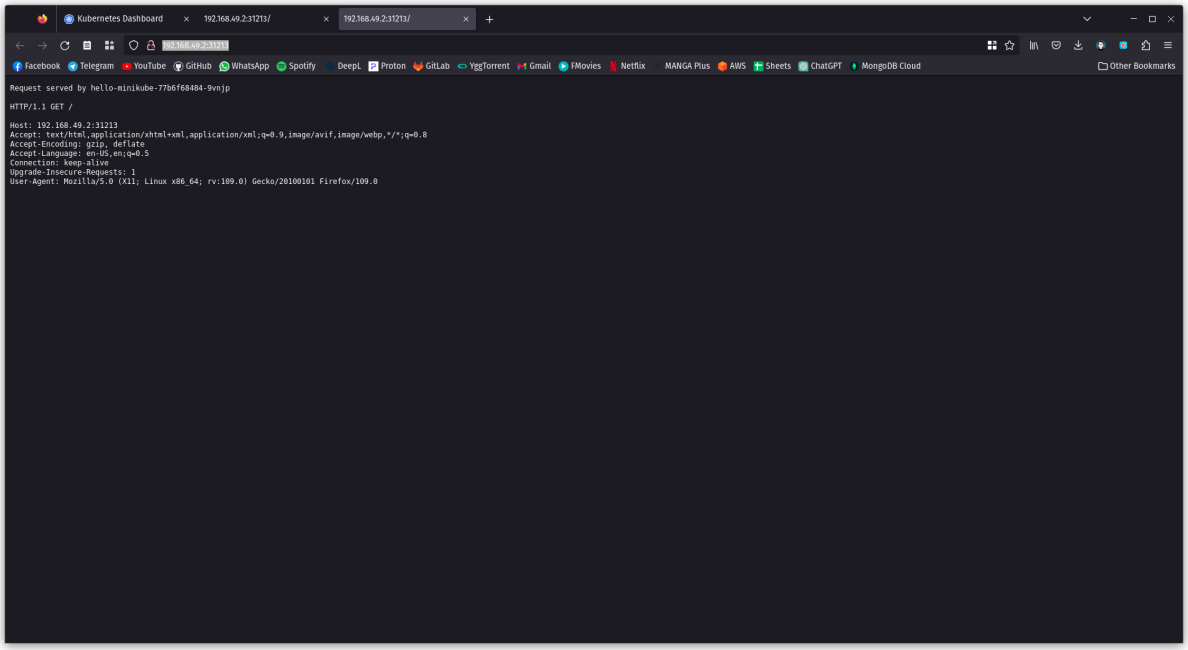
```
kubectl get services hello-minikube
```

To open the echo-server in your browser, run the following command:

```
minikube service hello-minikube --url
```



- From Firefox:



3 Upgrading a cluster in Minikube

To upgrade a cluster in Minikube, you must first upgrade the Minikube binary. Then, you can upgrade the Kubernetes version used by Minikube.

3.1 Upgrade the Minikube binary

To upgrade the Minikube binary, run the following command:

```
minikube update-check
```

This command will check if a newer version of Minikube is available. If a newer version is available, you will be prompted to upgrade. To upgrade, run the following command:

```
minikube update-context
```

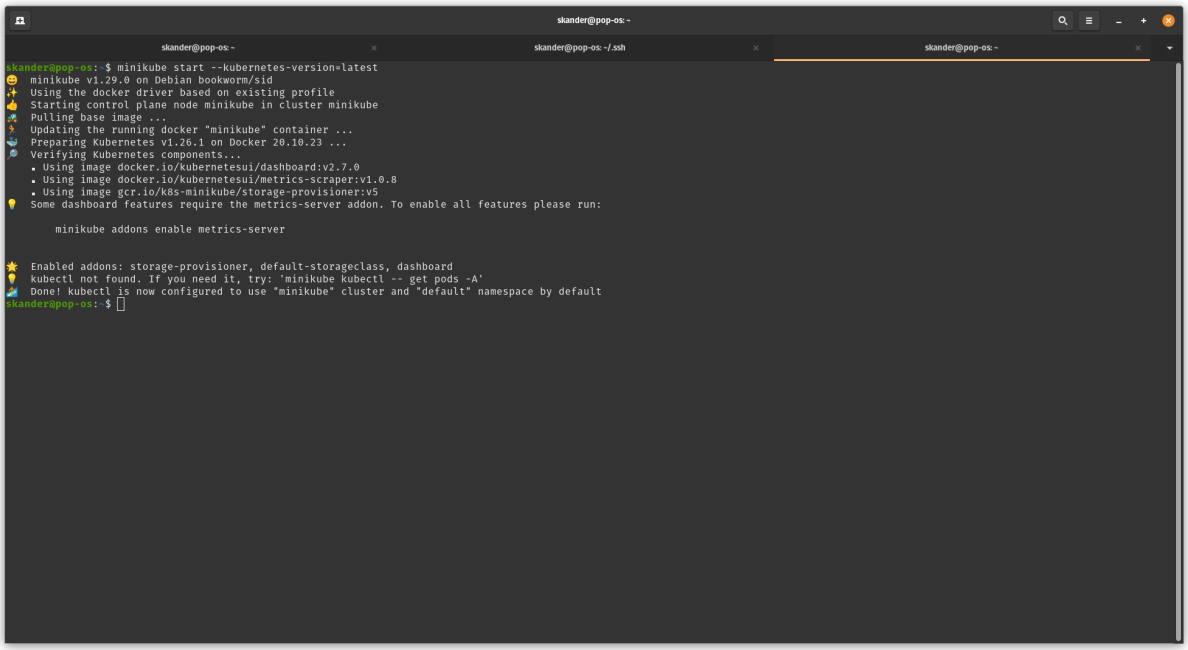
3.2 Upgrade the Kubernetes version

To upgrade the Kubernetes version used by Minikube, run the following command:

```
minikube start --kubernetes-version=<desired-version>
```

Replace with the version you want to use. For example, to use Kubernetes version 1.18.0, run the following command:

```
minikube start --kubernetes-version=latest
```


skander@pop-os: ~
skander@pop-os: ~
skander@pop-os: ~/.ssh
skander@pop-os: ~
skander@pop-os:~$ minikube start --kubernetes-version=latest
minikube v1.29.0 on Debian bookworm/sid
Using the docker driver based on existing profile
Starting control plane node minikube in cluster minikube
Pulling base image ...
Updating the running docker "minikube" container ...
Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
Verifying Kubernetes components...
Using image docker.io/kubernetesui/dashboard:v2.7.0
Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
Using image gcr.io/k8s-minikube/storage-provisioner:v5
Some dashboard features require the metrics-server addon. To enable all features please run:
minikube addons enable metrics-server
Enabled addons: storage-provisioner, default-storageclass, dashboard
kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
skander@pop-os:~$

4 Starting a second local cluster

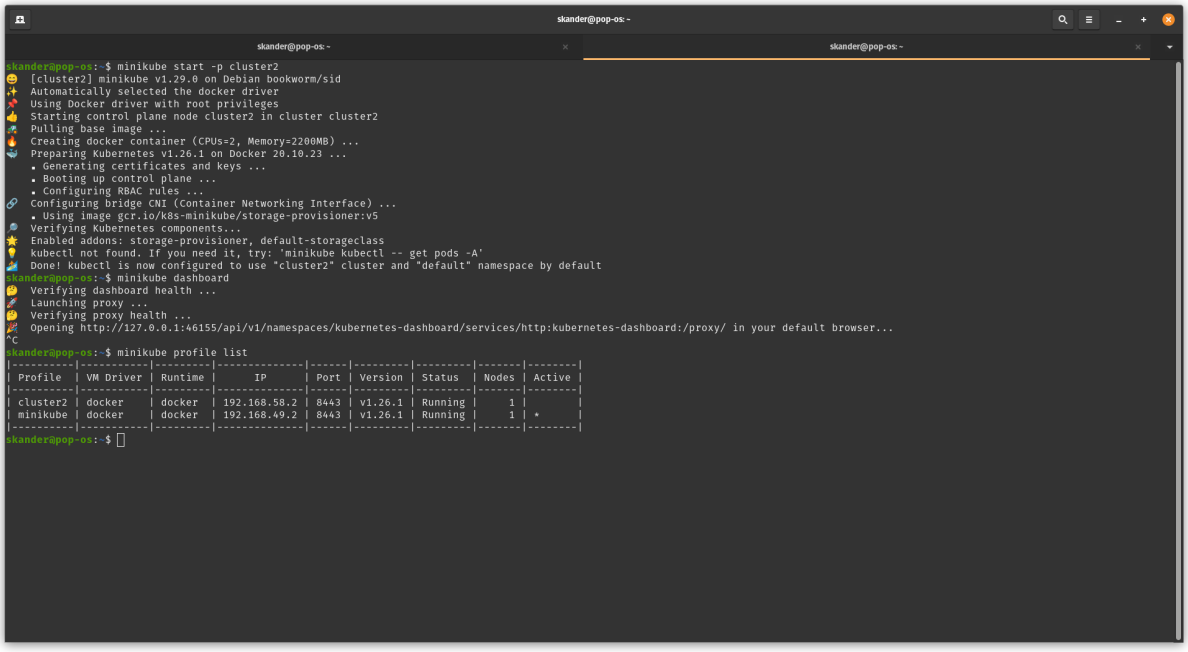
4.1 MiniKube Profiles

To start a second local cluster, run the following command:

```
minikube start --profile=<cluster-name>
```

To view the list of clusters, run the following command:

```
minikube profile list
```



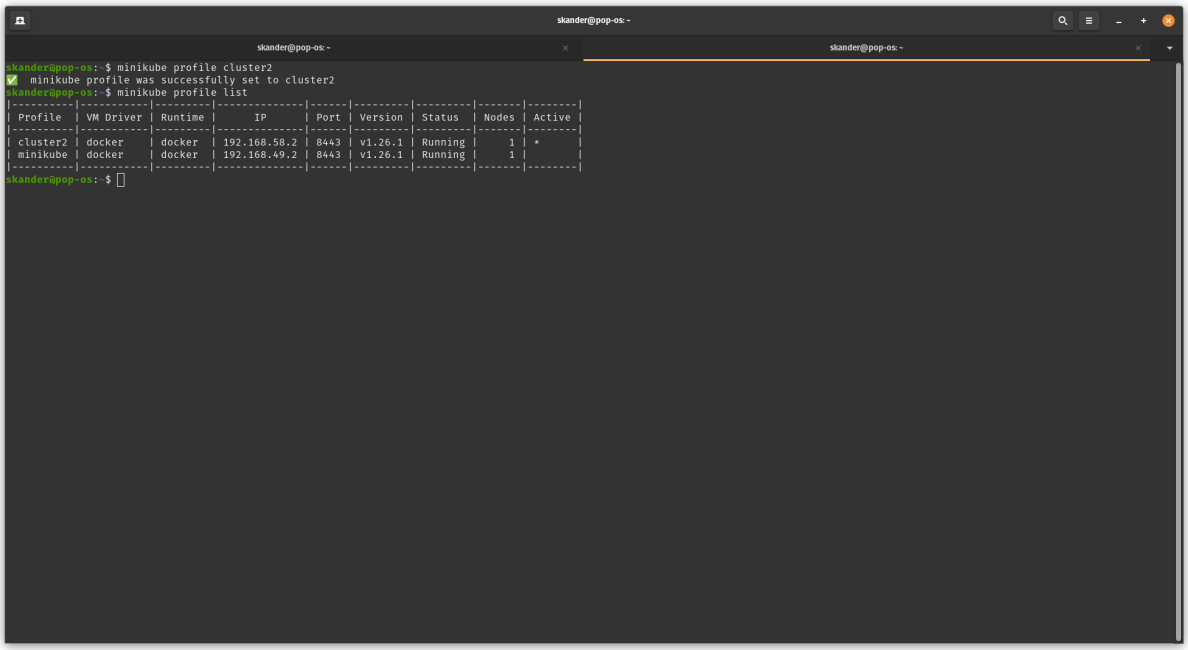
4.2 Switching between clusters

To switch between clusters, run the following command:

```
minikube profile <cluster-name>
```

To view the current cluster, run the following command:

```
minikube profile
```

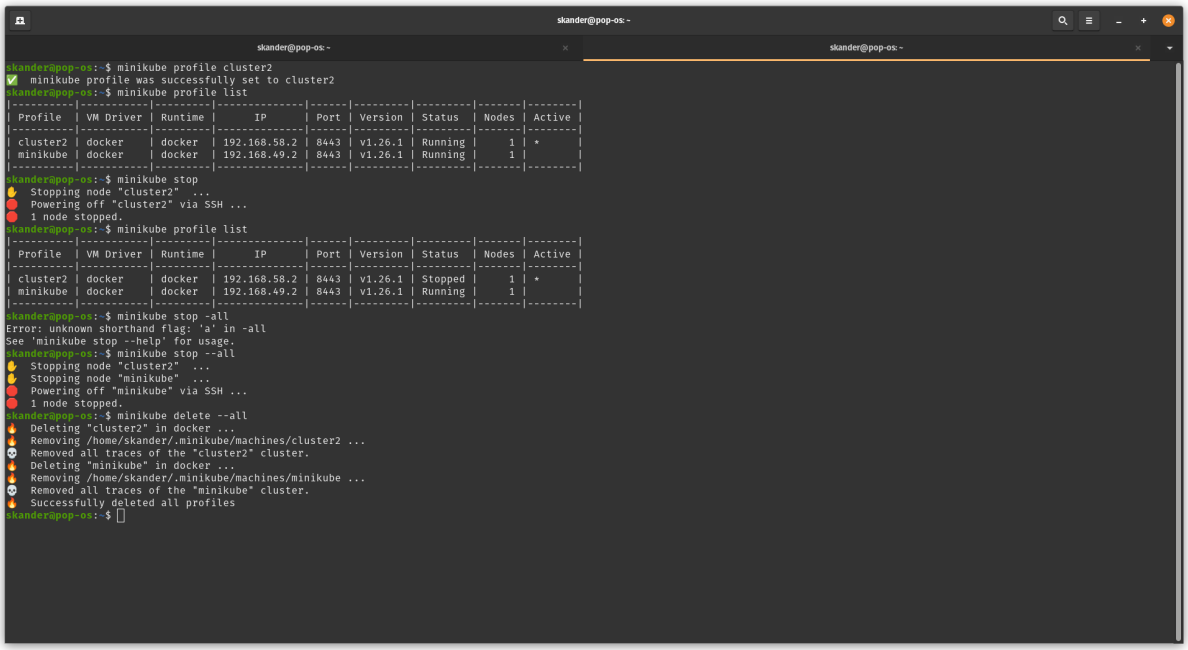


4.3 Deleting a cluster

To delete a cluster, run the following command:

```
minikube delete --profile=<cluster-name>
```

The `--all` flag can be used to delete all clusters.


skander@pop-os: ~
skander@pop-os: ~
skander@pop-os: ~
skander@pop-os:~$ minikube profile cluster2
✅ minikube profile was successfully set to cluster2
skander@pop-os:~$ minikube profile list
|---------|-----------|---------|--------------|------|---------|---------|-------|--------|
Profile	VM Driver	Runtime	IP	Port	Version	Status	Nodes	Active
cluster2	docker	docker	192.168.58.2	8443	v1.26.1	Running	1	*
minikube	docker	docker	192.168.49.2	8443	v1.26.1	Running	1	
skander@pop-os:~$ minikube stop
✋ Stopping node "cluster2" ...
🛑 Powering off "cluster2" via SSH ...
🛑 1 node stopped.
skander@pop-os:~$ minikube profile list
|---------|-----------|---------|--------------|------|---------|---------|-------|--------|
Profile	VM Driver	Runtime	IP	Port	Version	Status	Nodes	Active
cluster2	docker	docker	192.168.58.2	8443	v1.26.1	Stopped	1	*
minikube	docker	docker	192.168.49.2	8443	v1.26.1	Running	1	
skander@pop-os:~$ minikube stop -all
Error: unknown shorthand flag: 'a' in -all
See 'minikube stop --help' for usage.
skander@pop-os:~$ minikube stop --all
✋ Stopping node "cluster2" ...
✋ Stopping node "minikube" ...
🛑 Powering off "minikube" via SSH ...
🛑 1 node stopped.
skander@pop-os:~$ minikube delete --all
🔥 Deleting "cluster2" in docker ...
🔥 Removing /home/skander/.minikube/machines/cluster2 ...
💀 Removed all traces of the "cluster2" cluster.
🔥 Deleting "minikube" in docker ...
🔥 Removing /home/skander/.minikube/machines/minikube ...
💀 Removed all traces of the "minikube" cluster.
🔥 Successfully deleted all profiles
skander@pop-os:~$

5 Conclusion

5.1 Summary

Docker allows you to package and run applications in containers, while Kubernetes helps manage and deploy those containers across multiple nodes in a cluster. Together, they provide a powerful platform for building, deploying, and scaling modern applications.

kubectl is used to interact with Kubernetes clusters, while kubeadm is used to set up and manage Kubernetes clusters. Together, they provide a powerful set of tools for deploying and managing applications on a Kubernetes cluster.

5.2 Final Words

When speaking of testing and development, Minikube is a great tool to have in your toolbox as it is easy to use. However Kubernetes in Docker or KIND is a faster and more realistic way to test your applications.

Kind creates a multi-node Kubernetes cluster using Docker containers, which more closely resembles a real-world Kubernetes deployment than the single-node cluster created by Minikube.

Lab by Mr. Saifeddine Souissi