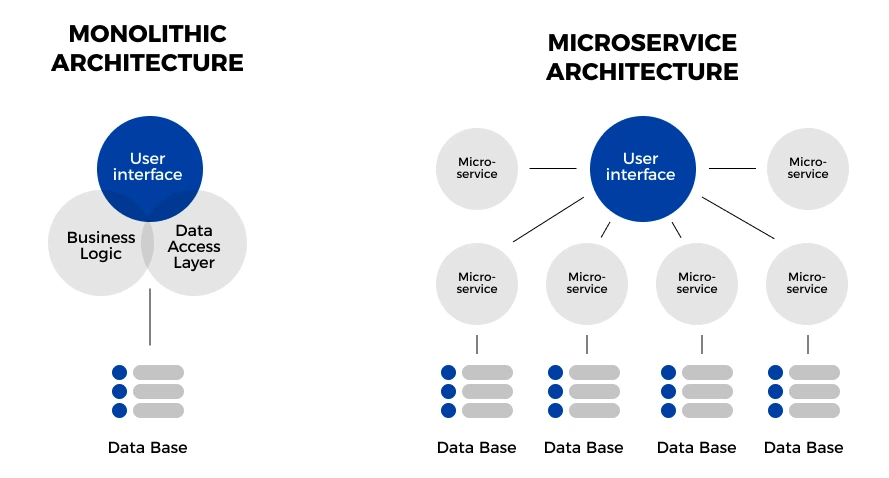
**Microservices:**

It is a finely tuned piece of software that performs a single or small collection of tasks. They are highly maintainable and testable, which means they are quick to deploy and develop. They're loosely coupled meaning they don't have any major dependencies on other parts of a system’s code base. They are independently deployable. New versions can be deployed to the stack without interfering with other components.

* Highly maintainable and testable
* Loosely coupled
* Independently deployable

**Monolith Code Base:**

* All sources into a single code base
* Tightly coupled (Classes can be dependent on each other throughout different parts of the code)
* Single point of failure (If one piece of the application crashes, it has the potential to crash the entire application rather than just that individual component.)
* Difficult to deployments and maintenance (It's difficult to manage the appointments and maintenance as well. Chances are you will have considerably more downtime for deployments with this model of code)
* Difficult to scale.



**Kubernetes Overview:**

* Open source container ***orchestration tool*** that is designed to automate, deploy, scale, and operate containerized applications
* Originally designed by Google, now CNCF (Cloud Native Computing Foundation) is maintaining it.
* Organizations adopting Kubernetes increased their velocity by having the ability to release faster and recover faster with Kubernetes self-healing mechanisms.
* It is a distributed system, so we can handle our applications that is running on Physical, virtual, or in the cloud
* Kubernetes often abbreviated as “K8s”

**Kubernetes Highlights:**

* Automated deployment rollout and rollback
* Seamless horizontal scaling
* Secret management
* Service discovery and load balancing
* Linux and windows container support
* Simple log collection
* Stateful application support
* Persistent volume management
* CPU and memory quotas
* Batch job processing
* Role-based access control

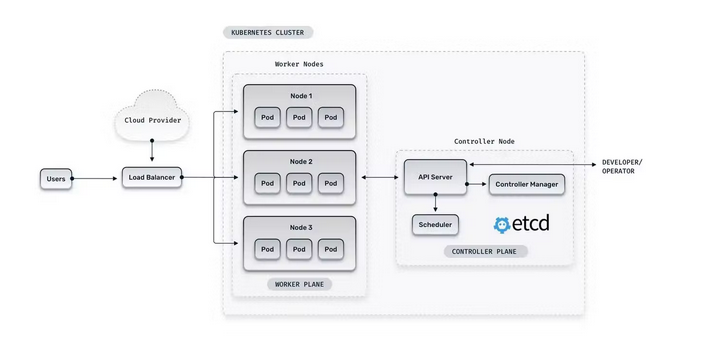
There are many other tools available in the market,

DC/OS (Distributed Cloud Operating System)

Amazon ECS (Elastic Container Service)

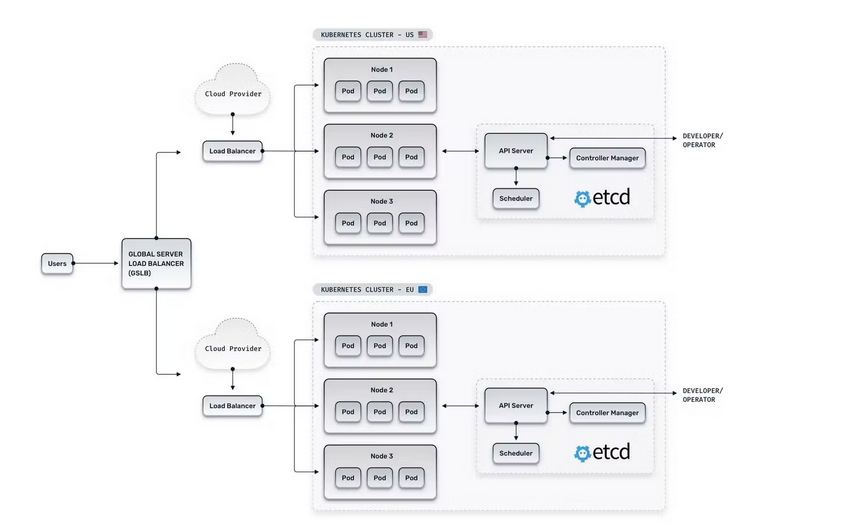
Docker swarm

**Kubernetes Clusters:**

**Single Node Cluster:** ***You should go for a single cluster if:***

* You prefer cost saving over high availability and fault tolerance
* You are short on skilled DevOps resources to manage multi-cluster
* There is no compliance or regulatory requirement to use separate infrastructure for a particular client

**Multi-Node Cluster:**



***You should go for multi-cluster if:***

* Your application requires a high degree of high availability, and you want zero downtime for your application
* Your team has enough technical expertise to manage complex setup and maintenance of multi-cluster
* You have compliance or regulatory need to place your clusters in a specific geographical region

**Kubernetes Deployments:**

1. **Single–Node Kubernetes Clusters:**

Docker - Docker for Mac and Docker for Windows, both include support for running Kubernetes on the local machine in a single-node configuration. Just make sure Kubernetes is enabled in the settings. This is the easiest way to get started if you already have Docker installed.

(Link: docker.com/products/docker-desktop)

Minikube – Supports Linux in addition to Mac and Windows

(Link: github.com/kubernetes/minikube)

Kubeadm – Linux can use kubeadm to set up the single-node cluster

(Link: kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/)

1. **Multi-Node Kubernetes Clusters:**

Fully managed: Amazon EKS (Elastic Kubernetes Service), AKS (Azure Kubernetes Service), GKE (Google Kubernetes Engine)

Full Control: Kubespray, KOPS, Kubeadm.

**Kubernetes Architecture:**

* **Cluster** refers to all of the machines collectively as the entire running system
* **Nodes** are the machines in the cluster (A group of Nodes form a Cluster)

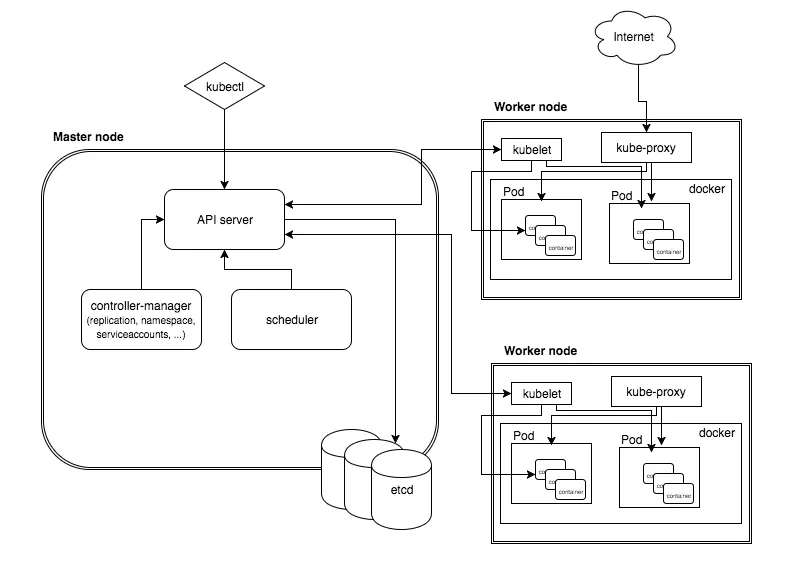
Nodes are categorized as workers or masters

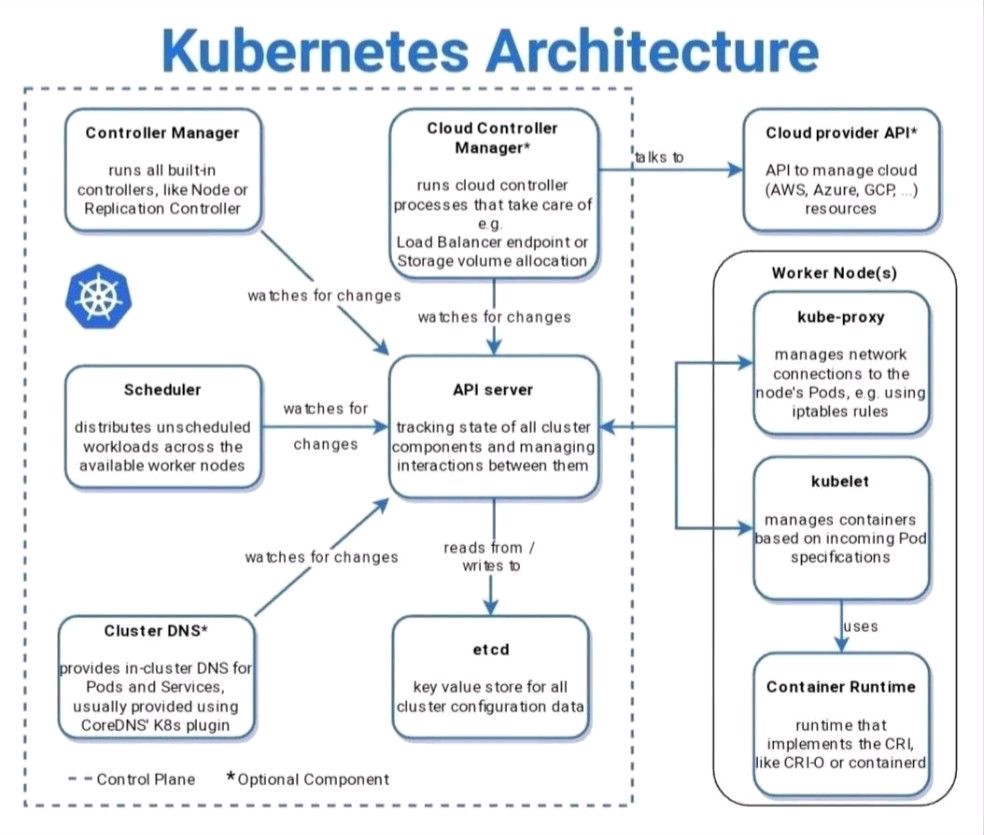
***Worker nodes*** include software to run containers managed by the Kubernetes control plane

***Master nodes*** run the control plane

Control Plane is a set of APIs and software that Kubernetes users interact with.

* **Pods** is an encapsulations of one or more containers, these are considered to be the smallest building block in Kubernetes. More complex and useful abstractions are built on top of Pods.
* **Services** define networking rules for exposing groups of Pods one to another or exposing Pods to the internet.
* **Deployments** used to manage the configuration and changes to running Pods as well as horizontal scaling. (In other words, this control rollout and rollback of Pods)





**Kubernetes Installation using Minikube:**

**Minikube System Requirements:**

* Minimum 2 CPUs or more
* Minimum 2GB of free memory
* Minimum 20GB of free disk space
* Internet connection
* Containers or virtual machine managers, such as Docker, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMware Fusion/Workstation

**Steps:**

**Create Ubuntu machine (22.04) – t2.medium – open all ports**

Update the system packages (*sudo apt update*)

**Install Kubectl on Ubuntu:**

* Download kubectl binary with curl on Ubuntu using below command

*curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl*

* Make the kubectl binary executable

*chmod +x ./kubectl*

* Move kubectl to /usr/local/bin/kubectl directory

*sudo mv ./kubectl /usr/local/bin/kubectl*

* To check kubectl version on Ubuntu

*kubectl version*

* Output:

Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.4", GitCommit:"b695d79d4f967c403a96986f1750a35eb75e75f1", GitTreeState:"clean", BuildDate:"2021-11-17T15:48:33Z", GoVersion:"go1.16.10", Compiler:"gc", Platform:"linux/amd64"}

**Install Docker on Ubuntu:**

* Install below packages before installing docker, you can use [official docker site](https://docs.docker.com/engine/install/ubuntu/)

*sudo apt-get install ca-certificates curl gnupg lsb-release*

* Add Docker official GPG Key

*sudo mkdir -p /etc/apt/keyrings*

*curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg*

* Setup Docker repository using below command

*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*

* Update the package to take effect

*sudo apt-get update*

* Install Docker on Ubuntu 22.04 LTS using below command

*sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin*

* To check docker service status on Ubuntu

*sudo systemctl status docker*

* Create group named docker

*sudo groupadd docker*

(groupadd: group 'docker' already exists = leave as it is), if already the group was not available then it will be created.

* Configure to Run docker without sudo permission

*sudo usermod -aG docker $USER && newgrp docker*

* To enable docker service at system startup

*sudo systemctl enable docker*

* To check status of docker service

*sudo systemctl status docker*

* To start/stop docker service

*sudo systemctl start/stop docker*

(Use only when it’s necessary)

**Install cri-dockerd on Ubuntu:**

* Clone the below git repo

*git clone https://github.com/Mirantis/cri-dockerd.git*

* To install, on a Linux system that uses systemd, and already has Docker Engine installed

*wget https://storage.googleapis.com/golang/getgo/installer\_linux*

*chmod +x ./installer\_linux*

*./installer\_linux*

*source ~/.bash\_profile*

*cd cri-dockerd*

*mkdir bin*

*go build -o bin/cri-dockerd*

**Note: Please wait above command takes some time to complete.**

*mkdir -p /usr/local/bin*

*sudo install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd*

*sudo cp -a packaging/systemd/\* /etc/systemd/system*

*sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service*

*sudo systemctl daemon-reload*

*sudo systemctl enable cri-docker.service*

*sudo systemctl enable --now cri-docker.socket*

(Change the directory back to Ubuntu user: *cd,* if you are trying the pervious steps as root user)

**Install conntrack package on Ubuntu:**

* Install conntrack using apt for minikube on Ubuntu 22.04 LTS

*sudo apt-get install -y conntrack*

**Install crictl package on Ubuntu:**

* Install crictl: CLI for kubelet CRI on Ubuntu using below commands follow [official crictl GitHub page](https://github.com/kubernetes-sigs/cri-tools#install-crictl)

*VERSION="v1.24.2"*

*wget https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz*

*sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin*

*rm -f crictl-$VERSION-linux-amd64.tar.gz*

**Download and Install Minikube on Ubuntu:**

* Download and Install Minikube on Ubuntu 22.04 LTS using below commands, To download latest minikube setup refer [minikube official download page](https://minikube.sigs.k8s.io/docs/start/)

*curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64*

* Make the minikube binary executable

*chmod +x minikube*

* Move minikube to /usr/local/bin/kubectl directory

*sudo mv minikube /usr/local/bin/*

**OR**

* You can use below command to download latest minikube

*curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64*

* Install Minikube on Ubuntu 22.04 LTS AWS EC2

*sudo install minikube-linux-amd64 /usr/local/bin/minikube*

* To check minikube version on ubuntu

*minikube version*

**Output:**

minikube version: v1.24.0  
commit: 76b94fb3c4e8ac5062daf70d60cf03ddcc0a741b

* Start the minikube Kubernetes cluster on Ubuntu with calico network plugin

*minikube start --network-plugin=cni --cni=calico*

**OR**

*minikube start --network-plugin=cni --cni=calico --wait=false*

**If there are Error, then follow these steps:**

minikube v1.29.0 on Ubuntu 22.04 (xen/amd64)

Using the none driver based on user configuration

X Exiting due to GUEST\_MISSING\_CONNTRACK: Sorry, Kubernetes 1.22.3 requires conntrack to be installed in root's path

**Solution:**

*sudo apt-get install -y conntrack*

* Now start the minikube on Ubuntu

*minikube start --network-plugin=cni --cni=calico*

**Output:**

minikube start --network-plugin=cni --cni=calico

\* minikube v1.29.0 on Ubuntu 22.04 (xen/amd64)

\* Automatically selected the docker driver. Other choices: none, ssh

! With --network-plugin=cni, you will need to provide your own CNI. See --cni flag as a user-friendly alternative

\* Using Docker driver with root privileges

\* Starting control plane node minikube in cluster minikube

\* Pulling base image ...

\* Downloading Kubernetes v1.26.1 preload ...

> preloaded-images-k8s-v18-v1...: 397.05 MiB / 397.05 MiB 100.00% 98.55 M

> gcr.io/k8s-minikube/kicbase...: 407.18 MiB / 407.19 MiB 100.00% 17.17 M

\* Creating docker container (CPUs=2, Memory=2200MB) ...

\* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...

- Generating certificates and keys ...

- Booting up control plane ...

- Configuring RBAC rules ...

\* Configuring Calico (Container Networking Interface) ...

- Using image gcr.io/k8s-minikube/storage-provisioner:v5

\* Verifying Kubernetes components...

\* Enabled addons: storage-provisioner, default-storageclass

\* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

* To Check the status of Minikube

*minikube status*

**Output:**

ubuntu@ip-172-31-36-57:~$ minikube status

minikube

type: Control Plane

host: Running

kubelet: Running

apiserver: Running

kubeconfig: Configured

* To check Minikube cluster information

*kubectl cluster-info*

**Output:**

Kubernetes control plane is running at https://192.168.49.2:8443

CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kubesystem/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

* To check minikube cluster status

*kubectl get nodes*

* To view minikube cluster events

*kubectl get events*

* To view the kubectl configuration

*kubectl config view*

**Output:**

apiVersion: v1

clusters:

- cluster:

certificate-authority: /home/ubuntu/.minikube/ca.crt

extensions:

- extension:

last-update: Sun, 19 Feb 2023 09:16:02 UTC

provider: minikube.sigs.k8s.io

version: v1.29.0

name: cluster\_info

server: https://192.168.49.2:8443

name: minikube

contexts:

- context:

cluster: minikube

extensions:

- extension:

last-update: Sun, 19 Feb 2023 09:16:02 UTC

provider: minikube.sigs.k8s.io

version: v1.29.0

name: context\_info

namespace: default

user: minikube

name: minikube

current-context: minikube

kind: Config

preferences: {}

users:

- name: minikube

user:

client-certificate: /home/ubuntu/.minikube/profiles/minikube/client.crt

client-key: /home/ubuntu/.minikube/profiles/minikube/client.key

* Lets create first container on Minikube Cluster using kubectl

*kubectl run hello-minikube --image=gcr.io/google\_containers/echoserver:1.4 --port=8080*

* To check pods on minikube using kubectl command

*kubectl get pods*

**Output:**

kubectl get pods

NAME READY STATUS RESTARTS AGE

hello-minikube 1/1 Running 0 37s

**Deploy an app on Minikube Cluster:**

* Lets create deployment on Minikube cluster

*kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4*

* To check deployment on minikube cluster

*kubectl get deployment*

**Output:**

kubectl get deployment

NAME READY UP-TO-DATE AVAILABLE AGE

hello-node 1/1 1 1 31s

* Expose the deployment using service on minikube cluster

*kubectl expose deployment hello-node --type=NodePort --port=8080*

* To check service on minikube cluster using kubectl

*kubectl get svc*

**Output:**

kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

hello-node NodePort 10.111.170.87 <none> 8080:32548/TCP 26s

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 11m

* we can access it within the cluster via $(minikube ip):$NODE\_PORT
* To check minikube internal IP

*minikube ip*

*curl -v 192.168.49.2:32548*

* Access the app on minikube cluster using cluster IP

*curl -v 192.168.49.2:32548*

**Output:**

\* Trying 13.234.67.176:32548...

\* TCP\_NODELAY set

\* Connected to 13.234.67.176 (13.234.67.176) port 32548 (#0)

> GET / HTTP/1.1

> Host: 13.234.67.176:32548

> User-Agent: curl/7.68.0

> Accept: \*/\*

>

\* Mark bundle as not supporting multiuse

< HTTP/1.1 200 OK

< Server: nginx/1.10.0

< Date: Thu, 25 Nov 2021 14:59:53 GMT

< Content-Type: text/plain

< Transfer-Encoding: chunked

< Connection: keep-alive

<

CLIENT VALUES:

client\_address=172.17.0.1

command=GET

real path=/

query=nil

request\_version=1.1

request\_uri=http://13.234.67.176:8080/

SERVER VALUES:

server\_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:

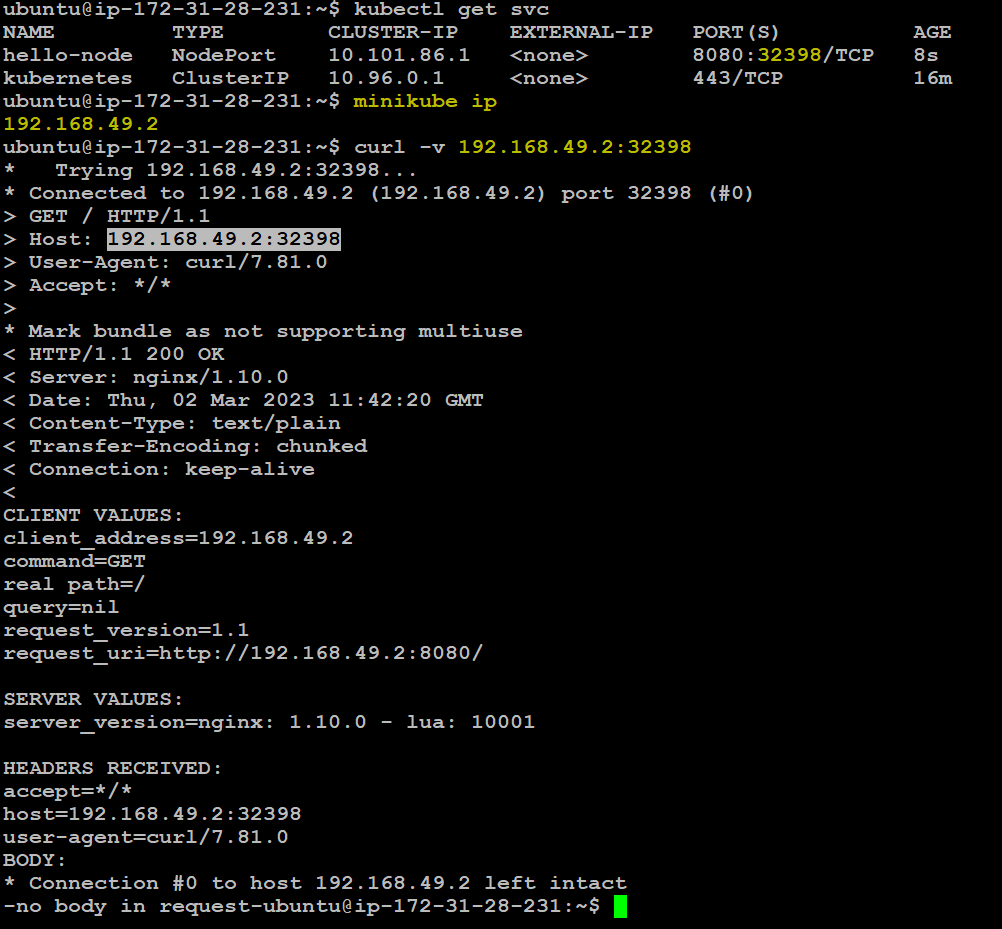
accept=\*/\*

host=13.234.67.176:32548

user-agent=curl/7.68.0

BODY:

\* Connection #0 to host 13.234.67.176 left intact

****

* To delete service on minikube cluster using kubectl

*kubectl delete service hello-node*

* To delete deployment on minikube cluster using kubectl

*kubectl delete deployment hello-node*

* To stop minikube cluster

*minikube stop*

**Output:**

\* Stopping node "minikube”...

\* 1 node stopped.

* To delete minikube cluster

*minikube delete*

**Output:**

\* Uninstalling Kubernetes v1.22.3 using kubeadm...

\* Deleting "minikube" in none...

\* Removed all traces of the "minikube" cluster.

Use this link: https://www.fosstechnix.com/how-to-install-minikube-on-ubuntu-22-04-lts/

Deploying Containerized Applications to Kubernetes:

Pods:

* What’s in a Pod Declaration
* Container image
* Container ports
* Container restart policy
* Resource limits