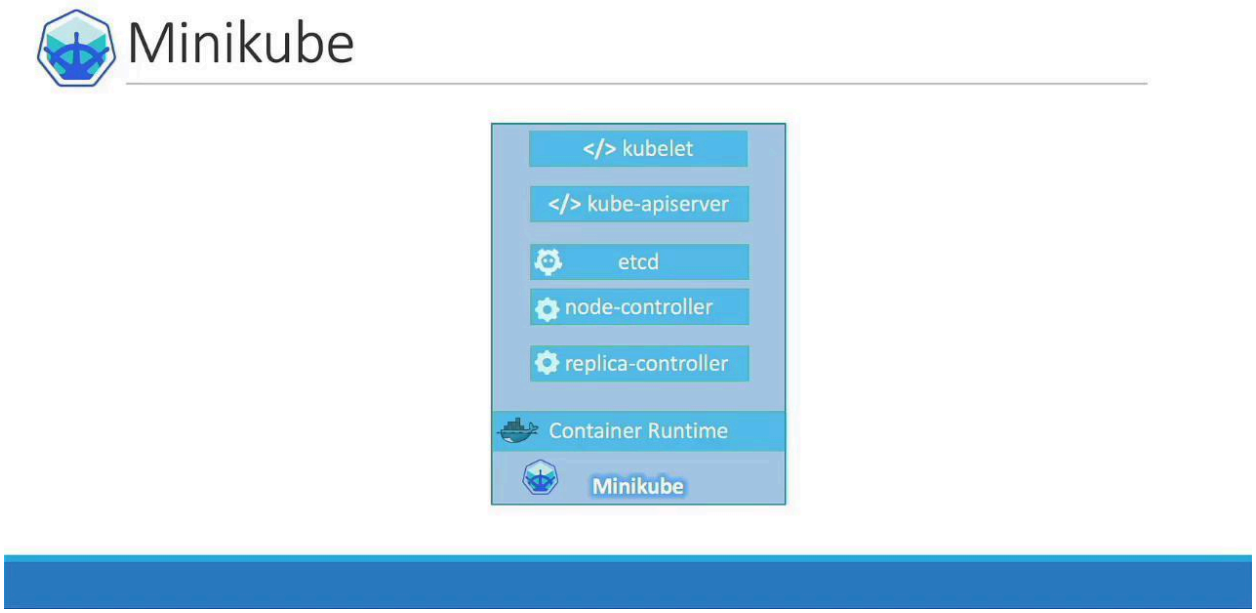
INSTALLING MINIKUBE AND KUBECTL IN WSL(UBUNTU)

Minikube is the easiest way to launch a local Kubernetes cluster.



How Minikube Works

Minikube packages the complete Kubernetes bundle into an ISO image that is automatically downloaded and deployed using its command-line utility. It integrates seamlessly with various virtualization platforms, such as:

- Oracle VirtualBox
- VMware Fusion
- Hyper-V (for Windows users)
- KVM (for Linux users)

To interact with your Kubernetes cluster, you also need to install the kubectl command-line tool. Here’s a quick summary of the requirements to get started with Minikube:

Requirement	Description
Hypervisor	A virtualization tool such as VirtualBox, Hyper-V, or KVM
kubectl	The official Kubernetes command-line tool
Minikube executable	The utility that automates ISO download and cluster deployment

Step 1: Install Dependencies in Ubuntu (WSL)

Open your Ubuntu WSL terminal:

```
sudo apt update -y

sudo apt install -y curl wget apt-transport-https
ca-certificates gnupg lsb-release conntrack
```

🐳 **Step 2: Install Docker (Optional but Recommended)**

Minikube can use Docker as a driver:

```
sudo apt install -y docker.io

sudo usermod -aG docker $USER

newgrp docker
```

Verify Docker:

```
docker version
```

📦 **Step 3: Install kubectl (Kubernetes CLI)**

LINK: https://gist.github.com/cmendible/ee6119ee202becd743888435e830b987

```
curl -LO
https://storage.googleapis.com/kubernetes-release/release/v1.16.0/bin/linux/amd6
4/kubectl
```

```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl


windowsUser=$1


mkdir -p ~/.kube
ln -sf "/mnt/c/users/$windowsUser/.kube/config" ~/.kube/config   —(try without this line once)


kubectl version
```

Step 4: Install Minikube

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

** Verify: `minikube version`

Step 5: Start Minikube (Using Docker driver)

```
minikube start --driver=docker
```

Got the below error:

```
X  Exiting due to GUEST_START: failed to start node: Failed kubeconfig update: writing kubeconfig: Error writing file /home/pallu/.kube/config: open /home/pallu/.kube/c
onfig: no such file or directory

    ⚠   If the above advice does not help, please let us know:
    ⚠   https://github.com/kubernetes/minikube/issues/new/choose

    Please run `minikube logs --file=logs.txt` and attach logs.txt to the GitHub issue.
```

Solution;

rm ~/.kube/config

mkdir -p ~/.kube

minikube start --driver=docker

Verify After Restart

Once it starts:

```
kubectl get nodes
```

```
pallu@DESKTOP-6UG7085:~$ rm ~/.kube/config
pallu@DESKTOP-6UG7085:~$ mkdir -p ~/.kube
pallu@DESKTOP-6UG7085:~$ minikube start --driver=docker
😄   minikube v1.36.0 on Ubuntu 24.04 (amd64)
✨  Using the docker driver based on existing profile

❗  The requested memory allocation of 1908MiB does not leave room for system overhead (total system  memory: 1908MiB). You may face stability issues.
💡   Suggestion: Start minikube with less memory allocated: 'minikube start --memory=1908mb'

📌   Starting "minikube" primary control-plane node in "minikube" cluster
🚜   Pulling base image v0.0.47 ...
🔄   Updating the running docker "minikube" container ...
🐳   Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
🔎   Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟   Enabled addons: storage-provisioner, default-storageclass

❗  /usr/local/bin/kubectl is version 1.16.0, which may have incompatibilities with Kubernetes 1.33.1.
    ▪ Want kubectl v1.33.1? Try 'minikube kubectl -- get pods -A'
🏄   Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
pallu@DESKTOP-6UG7085:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   3m36s   v1.33.1
pallu@DESKTOP-6UG7085:~$
```

minikube status

```
pallu@DESKTOP-6UG7085:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Deploying a Sample Application

With your cluster up and running, deploy a sample application to verify that the environment is fully operational.

1. Create a Deployment

Deploy a sample echoserver application with the following command:

```
kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10
```

You should see a confirmation message:

```
deployment.apps/hello-minikube created
```

2. Verify the Deployment

Check the deployment status with

```
kubectl get deployments
```

```
pallu@DESKTOP-6UG7085:~$ kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10
deployment.apps/hello-minikube created
pallu@DESKTOP-6UG7085:~$ kubectl get deployents
error: the server doesn't have a resource type "deployents"
pallu@DESKTOP-6UG7085:~$ kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   0/1     1            0           18s
pallu@DESKTOP-6UG7085:~$ kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           23s
pallu@DESKTOP-6UG7085:~$
```

3. Expose the Deployment as a Service

Expose the deployment on port 8080 using a NodePort service:

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

```
pallu@DESKTOP-6UG7085:~$ kubectl expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed
pallu@DESKTOP-6UG7085:~$ minikube service hello-minikube --url
http://127.0.0.1:35363
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

To obtain the URL of the exposed service, run:

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

Open the URL in your browser to view details of the application. Although the interface may be basic, this confirms your cluster's functionality.

URL: http://127.0.0.1:35363/

To make terminal Interactive:

✅ Step 1:

Get your pod name:

```
kubectl get pods
```

✅ Step 2: Forward pod port to your local machine

Replace `<your-pod-name>` below with the actual pod name from Step 1:

```
kubectl port-forward pod/<your-pod-name> 8080:8080
```

Using a terminal multiplexer (like tmux or screen)

If you prefer terminal sessions you can detach and reattach to, install tmux:

sudo apt install tmux

tmux

kubectl port-forward pod/hello-minikube-74878c8fcc-rmm9b 8080:8080

Then press **`Ctrl+b`**, then **`d`** to detach.

```
pallu@DESKTOP-6UG7085:~$ sudo apt install tmux
[sudo] password for pallu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tmux is already the newest version (3.4-1ubuntu0.1).
tmux set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 115 not upgraded.
pallu@DESKTOP-6UG7085:~$ tmux
[detached (from session 0)]
```

Reattach anytime with:

```
tmux attach
```

** Once forwarding is active, open your browser on Windows (or WSL browser if you have one) and go to:

http://localhost:8080