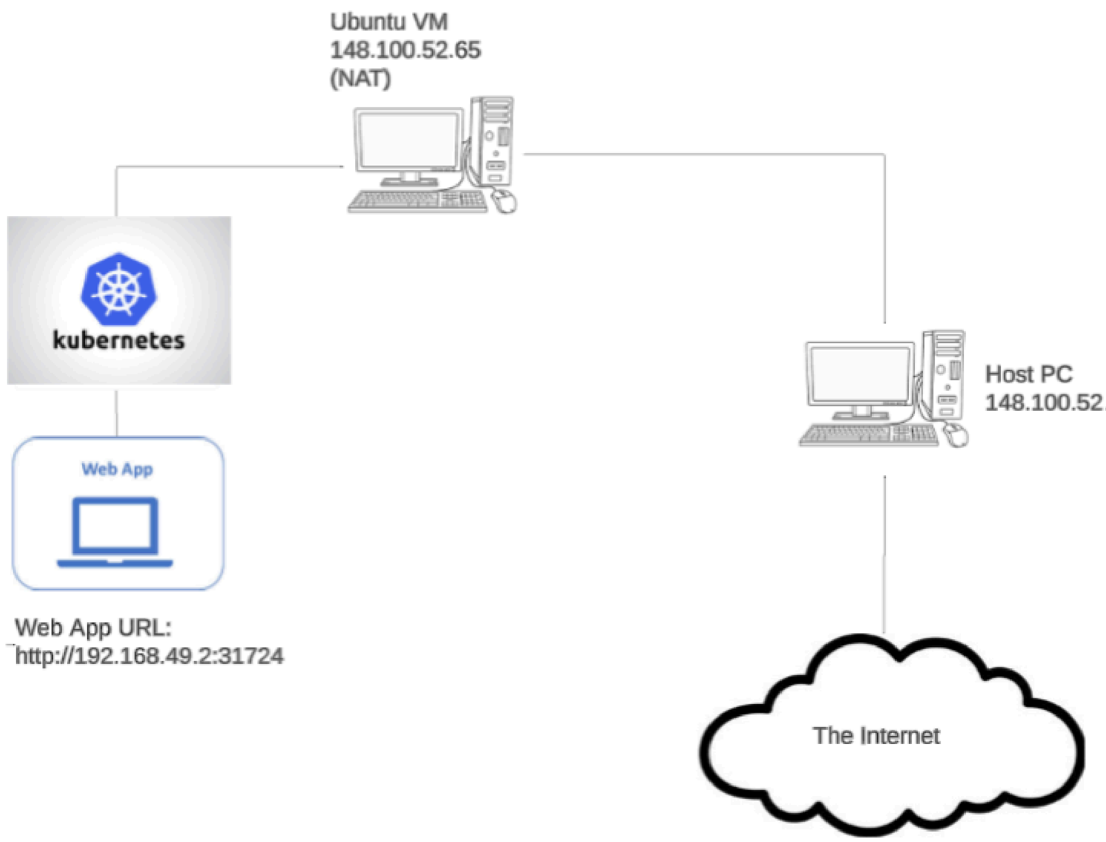
Eric Guzman
Professor Cannistra

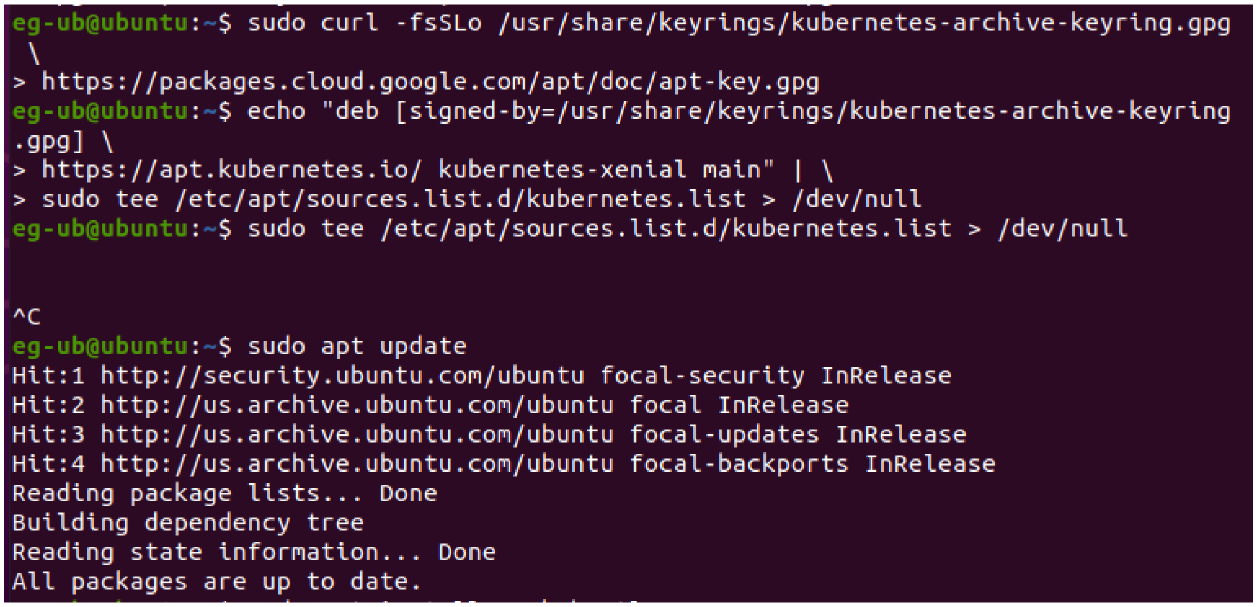
5/8/2025
Network Virtualization

Final Project

I chose option 2: Kubernetes Implementation.

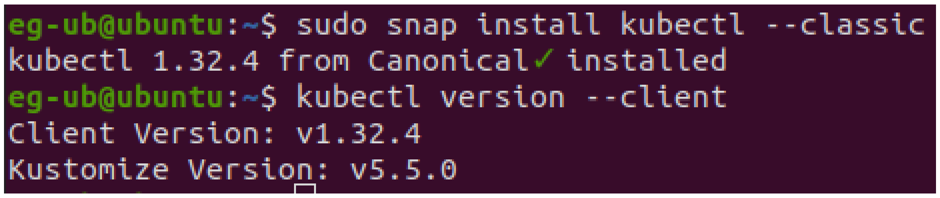
Topology:





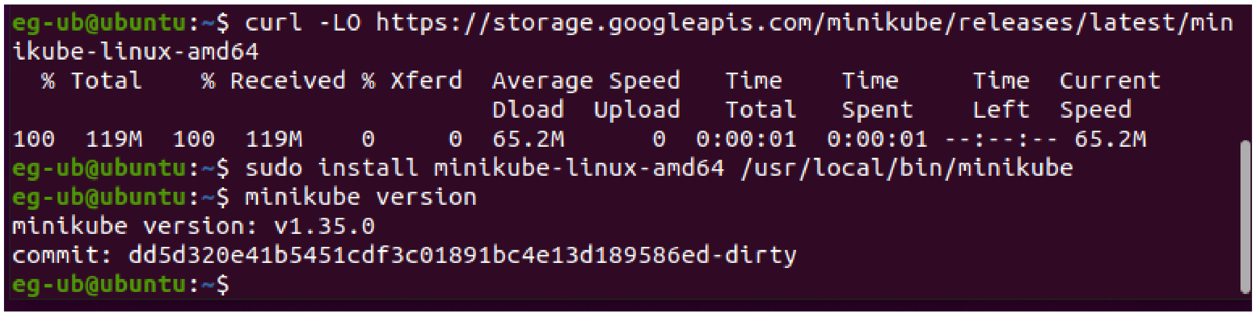
```
eg-ub@ubuntu:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
 \
> https://packages.cloud.google.com/apt/doc/apt-key.gpg
eg-ub@ubuntu:~$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring
.gpg] \
> https://apt.kubernetes.io/ kubernetes-xenial main" | \
> sudo tee /etc/apt/sources.list.d/kubernetes.list > /dev/null
eg-ub@ubuntu:~$ sudo tee /etc/apt/sources.list.d/kubernetes.list > /dev/null


^C
eg-ub@ubuntu:~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
```

This screenshot shows the addition of the Kubernetes APT repository and its GPG signing key to the system.



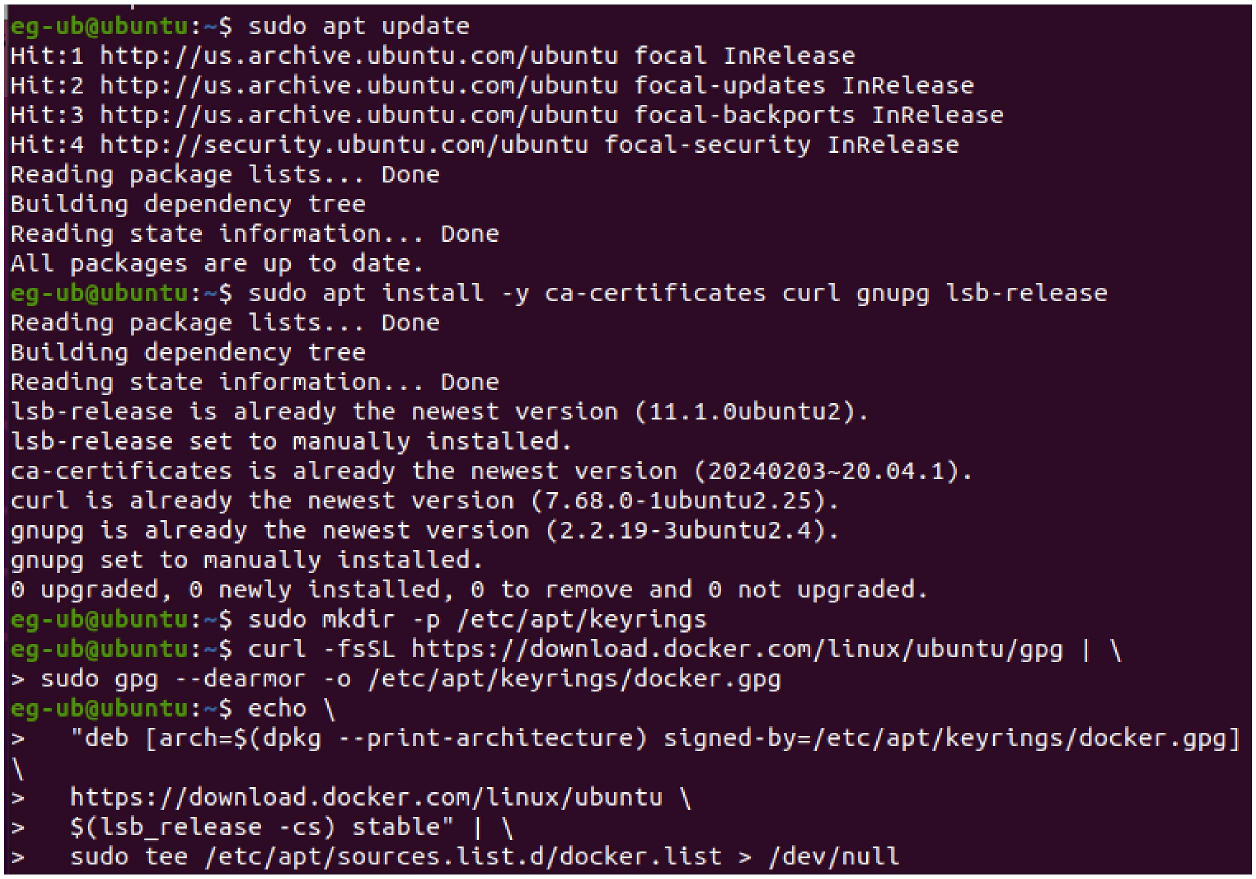
```
eg-ub@ubuntu:~$ sudo snap install kubectl --classic
kubectl 1.32.4 from Canonical✓ installed
eg-ub@ubuntu:~$ kubectl version --client
Client Version: v1.32.4
Kustomize Version: v5.5.0
```

This screenshot shows the successful installation and verification of kubectl using Snap package manager.



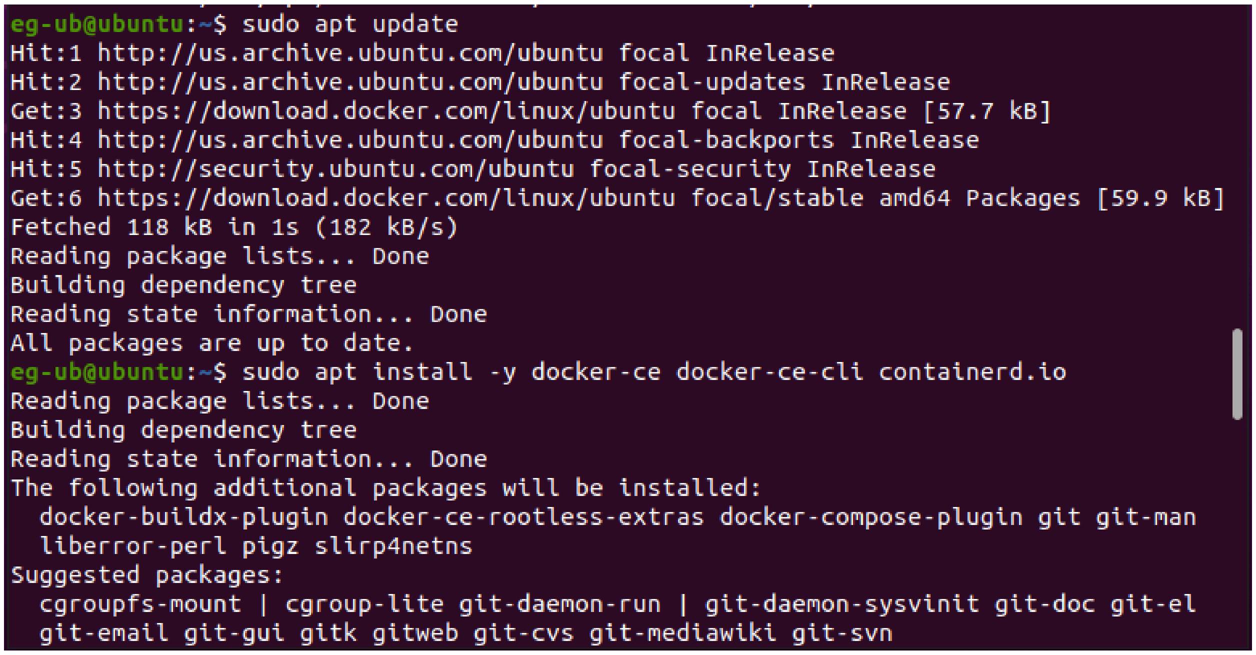
```
eg-ub@ubuntu:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/min
ikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0  65.2M      0  0:00:01  0:00:01 --:--:-- 65.2M
eg-ub@ubuntu:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
eg-ub@ubuntu:~$ minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
eg-ub@ubuntu:~$
```

This screenshot shows the installation of Minikube, and showing that version 1.35.0 was installed.



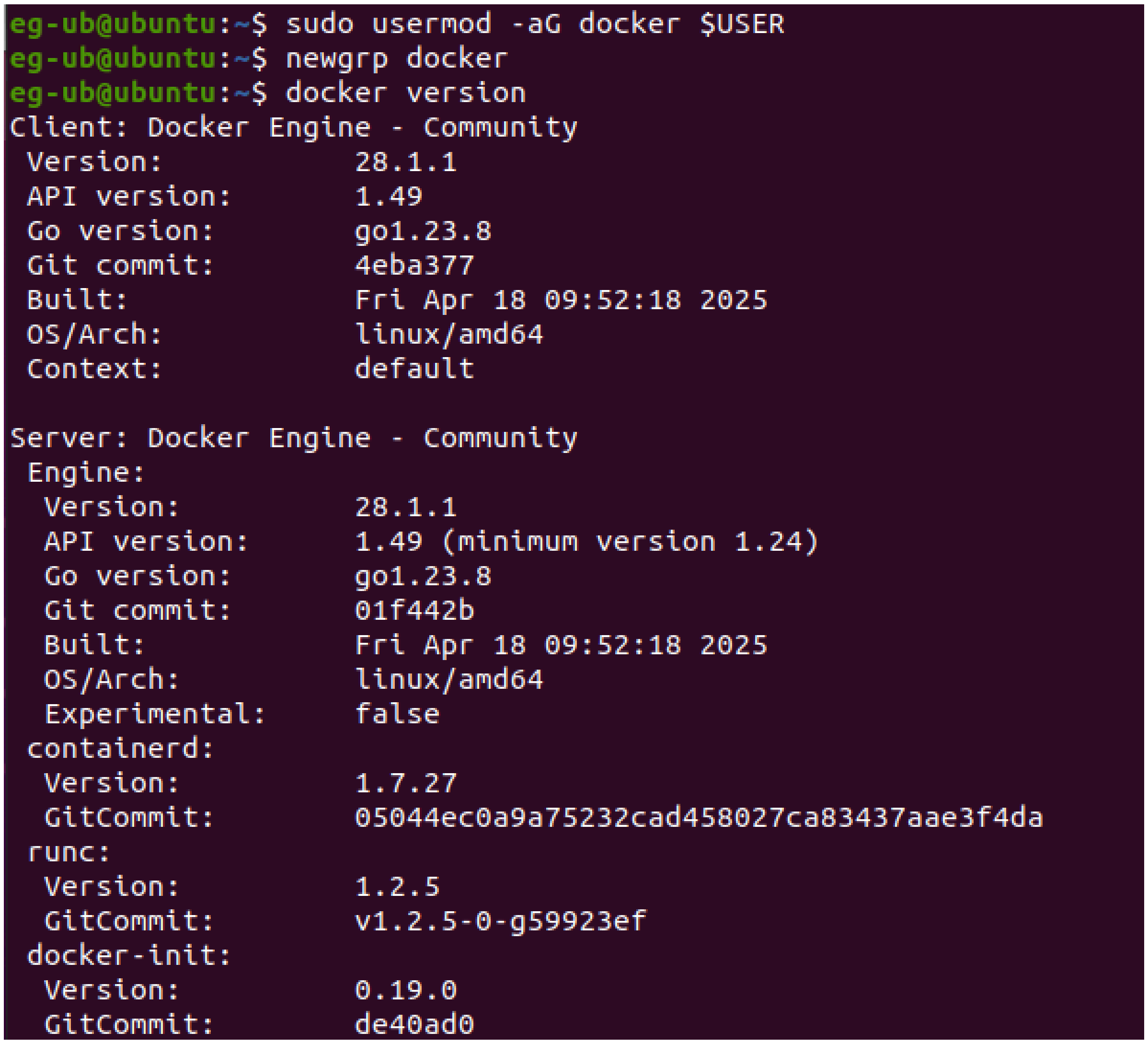
```
eg-ub@ubuntu:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
eg-ub@ubuntu:~$ sudo apt install -y ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu2).
lsb-release set to manually installed.
ca-certificates is already the newest version (20240203~20.04.1).
curl is already the newest version (7.68.0-1ubuntu2.25).
gnupg is already the newest version (2.2.19-3ubuntu2.4).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
eg-ub@ubuntu:~$ sudo mkdir -p /etc/apt/keyrings
eg-ub@ubuntu:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
> sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
eg-ub@ubuntu:~$ 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
```

This screenshot shows preparations to install Docker. I am using docker here to serve as the container runtime backend for Minikube. This lets the Kubernetes cluster run directly inside Docker containers which improves performance and simplifies the local setup.



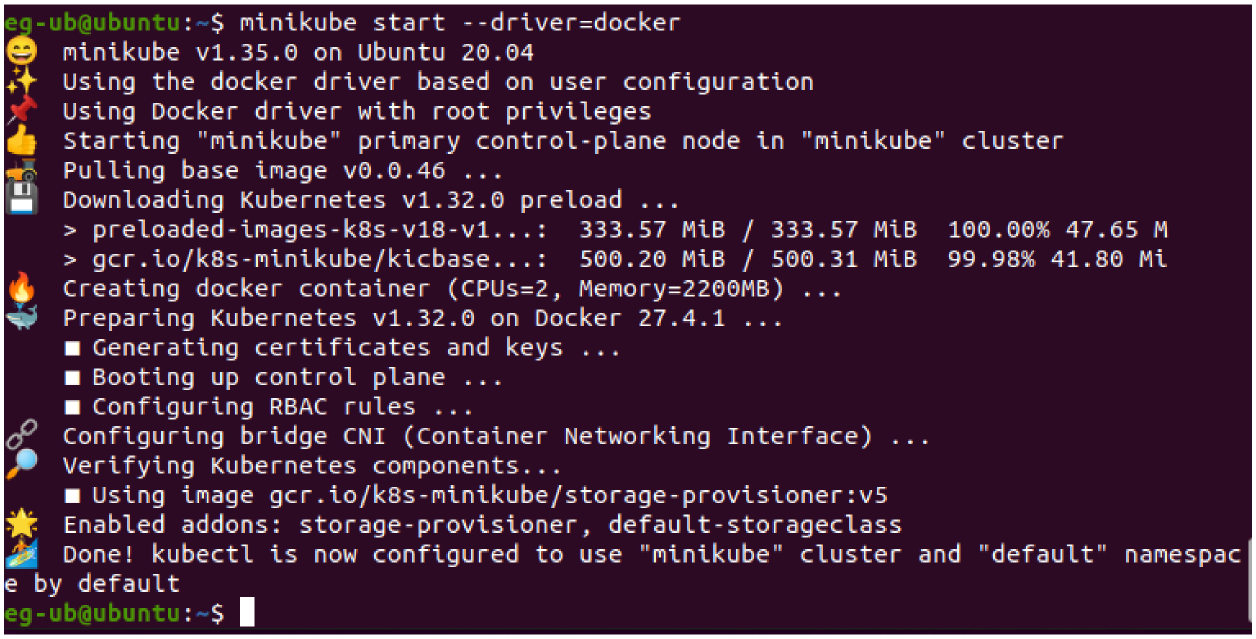
```
eg-ub@ubuntu:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Get:3 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu focal-security InRelease
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [59.9 kB]
Fetched 118 kB in 1s (182 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
eg-ub@ubuntu:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin git git-man
  liberror-perl pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-el
  git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
```

This screenshot shows the final install of Docker and its required components.



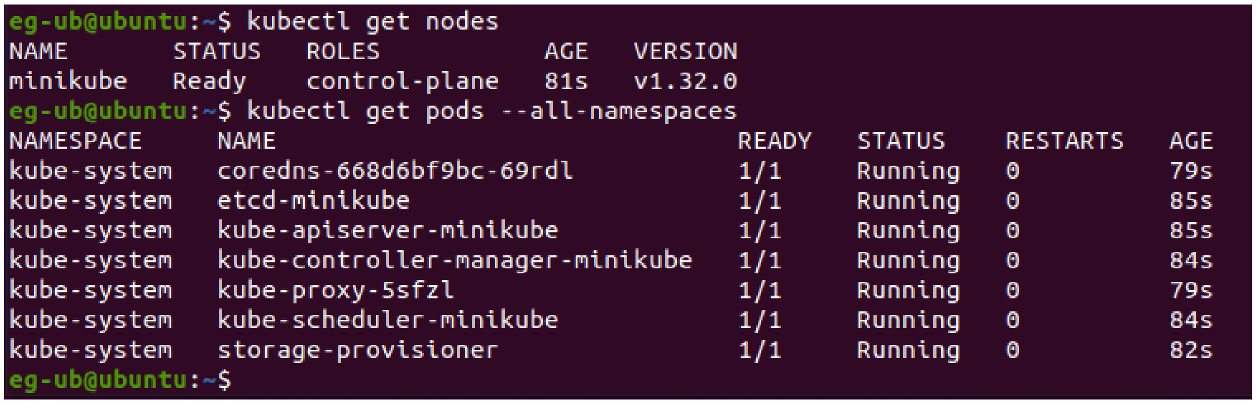
```
eg-ub@ubuntu:~$ sudo usermod -aG docker $USER
eg-ub@ubuntu:~$ newgrp docker
eg-ub@ubuntu:~$ docker version
Client: Docker Engine - Community
 Version:           28.1.1
 API version:       1.49
 Go version:        go1.23.8
 Git commit:        4eba377
 Built:             Fri Apr 18 09:52:18 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          28.1.1
  API version:      1.49 (minimum version 1.24)
  Go version:       go1.23.8
  Git commit:       01f442b
  Built:            Fri Apr 18 09:52:18 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
 runc:
  Version:          1.2.5
  GitCommit:        v1.2.5-0-g59923ef
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

This screenshot shows Docker correctly working after installation. We can see Docker version 28.1.1 is being used and the user was added to the Docker group.



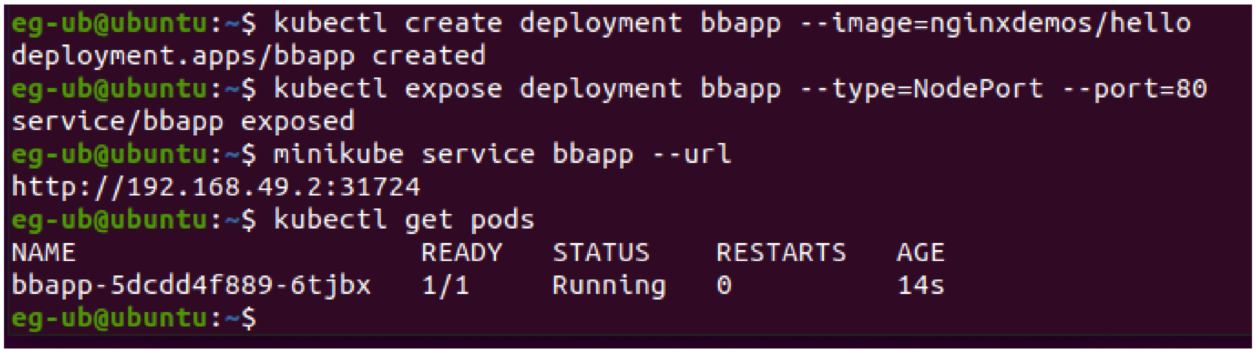
```
eg-ub@ubuntu:~$ minikube start --driver=docker
😄  minikube v1.35.0 on Ubuntu 20.04
✨  Using the docker driver based on user configuration
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
💾  Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  333.57 MiB / 333.57 MiB  100.00% 47.65 M
    > gcr.io/k8s-minikube/kicbase...:  500.20 MiB / 500.31 MiB  99.98% 41.80 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    ■ Generating certificates and keys ...
    ■ Booting up control plane ...
    ■ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespac
e by default
eg-ub@ubuntu:~$ 
```

This screenshot shows Minikube successfully starting a local Kubernetes cluster while using the Docker driver. This output confirms that kubectl is set up and ready to manage the cluster.



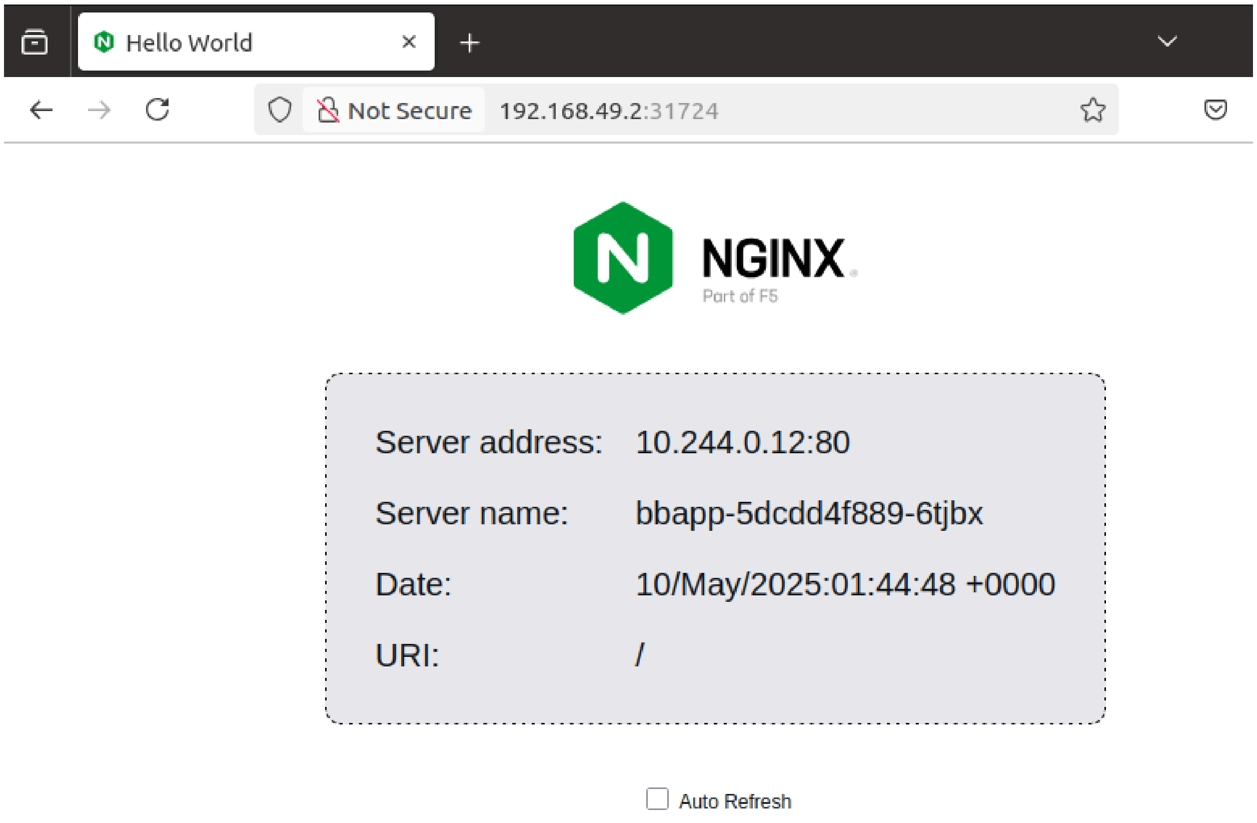
```
eg-ub@ubuntu:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   81s   v1.32.0
eg-ub@ubuntu:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-668d6bf9bc-69rdl           1/1     Running   0          79s
kube-system   etcd-minikube                      1/1     Running   0          85s
kube-system   kube-apiserver-minikube            1/1     Running   0          85s
kube-system   kube-controller-manager-minikube   1/1     Running   0          84s
kube-system   kube-proxy-5sfzl                   1/1     Running   0          79s
kube-system   kube-scheduler-minikube            1/1     Running   0          84s
kube-system   storage-provisioner                1/1     Running   0          82s
eg-ub@ubuntu:~$
```

This screenshot shows verification of the running Kubernetes cluster. The kubectl get nodes command shows that the Minikube node is ready while kubectl get pods --all-namespaces shows all of the essential system pods that are running without errors.



```
eg-ub@ubuntu:~$ kubectl create deployment bbapp --image=nginxdemos/hello
deployment.apps/bbapp created
eg-ub@ubuntu:~$ kubectl expose deployment bbapp --type=NodePort --port=80
service/bbapp exposed
eg-ub@ubuntu:~$ minikube service bbapp --url
http://192.168.49.2:31724
eg-ub@ubuntu:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
bbapp-5dcdd4f889-6tjbx   1/1     Running   0          14s
eg-ub@ubuntu:~$
```

This screenshot shows the creation and deployment of my sample application using Kubernetes. I launched a nginxdemo container then exposed it on port 80.



Server address:	10.244.0.12:80
Server name:	bbapp-5dcdd4f889-6tjbx
Date:	10/May/2025:01:44:48 +0000
URI:	/

This screenshot shows my web application successfully running in a browser after being deployed with Kubernetes.

The implementation of Kubernetes gives many benefits for managing containerized applications. It allows for the automation of deploying, scaling and recovery which greatly reduces any manual effort needed and increases reliability. Kubernetes can automatically restart failed containers and make sure they are constantly available. It can also efficiently utilize resources by scheduling workloads depending on any available system resources. It makes load balancing and service discovery very simple, making it very easy to expose applications. These features make Kubernetes a very ideal platform to scale applications and manage them efficiently.