As always let’s start with the prerequisites😅,

 Open VMware settings for the VM.

 Increase RAM to **at least 4 GB**.

Will update our local repositories first so that we always have the latest version of the packages we want to install

sudo yum -y update

Now, you must install a virtualization application since the single node cluster will be executed within a virtual machine. So let’s use KVM Hypervisor, you can go with any other hypervisor if you feel like it.

There are some dependencies that we first need to meet to install KVM, we will run the below commands to meet them

sudo yum -y install epel-release

Now that the dependencies are taken care of let’s focus on getting our KVM up & running, below command should do it for you

sudo yum -y install libvirt qemu-kvm virt-install virt-top libguestfs-tools bridge-utils

Additional Check through chatpgt

 Check if KVM is supported:

egrep -c '(vmx|svm)' /proc/cpuinfo

If the result is 0, then **your VM doesn't support virtualization acceleration**.

 Or, install and configure libvirt/kvm properly **if supported**:

sudo yum install -y qemu-kvm libvirt virt-install bridge-utils

sudo systemctl enable --now libvirtd

Awesome now you have your Virtualization tool installed, lets start it up and enable it

sudo systemctl start libvirtd

sudo systemctl enable libvirtd

et's check its status, if you receive an active running status you are good to go ahead with it

sudo systemctl status libvirtd

Do you have it active and enabled?… great let's add a new user for libvrtd group

sudo usermod -a -G libvirt $(whoami)

We now need to do some changes to the configuration files of our virtualization service

sudo vim /etc/libvirt/libvirtd.conf  
# We need to set two values here in this file as  
unix\_sock\_group = "libvirt"  
unix\_sock\_rw\_perms = "0770"

Save the file using the following keys in exact order Esc, shift colon & finally wq and enter.  
Done?

Additional Steps

Add NOPASSWD entry to /etc/sudoers for podman:

 Add this line: visudo

**yourusername** ALL=(ALL) NOPASSWD: /usr/bin/docker

 Or tell Minikube to use Podman/docker **in rootless mode**:

minikube config set rootless true

**Docker: Permission Denied**

**Error:**

permission denied while trying to connect to the Docker daemon socket

**Fix:**  
Add your user to the docker group:

sudo usermod -aG docker $USER

newgrp docker

Then, **logout and log back in**, or restart your terminal session.

**Verify:**

docker version

Awesome, final step to get our Virtualization tool running is restarting it so it reloads with the newly added configuration

sudo systemctl restart libvirtd.service

We have now successfully set up our Virtualization tool let’s get our Minikube to work now, shall we?

Let’s download⏬ the latest Minikube binary using the below wget command

wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

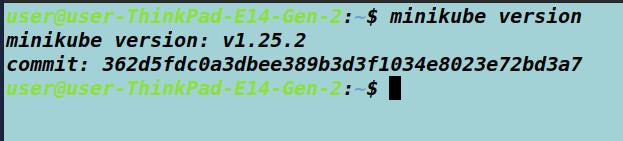
et's bestow the proper permission👮‍♂️ on the downloaded binary using chmod, and move it to /usr/bin

sudo chmod +x minikube-linux-amd64 && sudo mv minikube-linux-amd64 /usr/bin/minikube

With this step, you should be able to check Minikube version in the terminal with the command

minikube version

And it should return an output similar to this



Now we need to install one last binary to get everything working “kubectl”, a CLI-based tool to reach Kubernetes, Let's download⏬ its binary first

### Install kubectl using native package management

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

curl -LO "https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl.sha256"

1. Validate the kubectl binary against the checksum file:
2. echo "**$(**cat kubectl.sha256**)** kubectl" | sha256sum --check

If valid, the output is:

kubectl: OK

If the check fails, sha256 exits with nonzero status and prints output similar to:

kubectl: FAILED

sha256sum: WARNING: 1 computed checksum did NOT match

#### Note:

Download the same version of the binary and checksum.

1. Install kubectl
2. sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

#### Note:

If you do not have root access on the target system, you can still install kubectl to the ~/.local/bin directory:

chmod +x kubectl

mkdir -p ~/.local/bin

mv ./kubectl ~/.local/bin/kubectl

*# and then append (or prepend) ~/.local/bin to $PATH*

1. Test to ensure the version you installed is up-to-date:
2. kubectl version --client

Or use this for detailed view of version:

kubectl version --client --output=yaml

Amazing, let's fire up🔥 our tiny single-node Minikube now to check a few things

minikube start --driver=docker

minikube start

# you can run below commands to start, stop & restart minikube

minikube stop  
minikube restart

always delete minikube

minikube delete

### ****Disable Rootless Mode in Minikube****

If you're OK running Docker in **normal rootful mode** (which is most common), just **disable rootless mode in Minikube**:

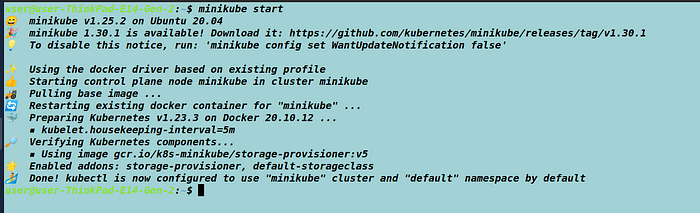
minikube config unset rootless

Then try starting again:

minikube delete

minikube start --driver=docker

You will receive a similar output in a Happy scenario,



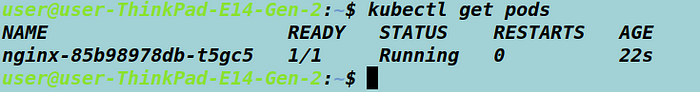
Finally, let's run a few Kubernetes commands and deploy an nginx pod on the server using the below commands

kubectl create deployment nginx --image=nginx

Now let's check if we have the pod running properly

kubectl get pods

you will see a pod created for nginx in the default namespace

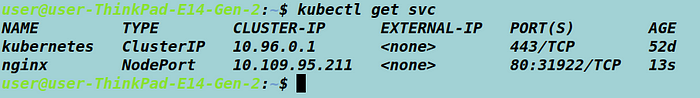


Cool, now let's expose the nginx on our localhost with a service

kubectl create service nodeport nginx --tcp=80:80

Awesome you will now have a service created with service type nodePort

kubectl get svc



Let's see if we can find our nginx UI on the browser, So similar to docker 80 is an internal port here for nginx application hosted inside the pod, but for our local intranet, it will be the 31922 port. Let's access our application on Control Planes IP. To get your control planes IP run the below command

kubectl get cluster-info

It should output similar to this

https://miro.medium.com/v2/resize:fit:700/1*6rNuULL93g_f3IV1X_Rn1A.png

Awesome let's copy our cluster plane IP and paste it into the browser remember to replace the HTTPS with HTTP since we have it running locally and add our node port replacing the port 8443 example

http://192.168.49.2:31922/

You should get your nginx UI on the address