

Minikube is a free and open-source tool that allows us to set up a single node Kubernetes cluster locally on our system. Minikube is only used for learning Kubernetes and allows developers to build their test environment locally on their system.

In this post, we will show you how to install Minikube on Rocky Linux 9 step by step. To begin its installation, make sure following system requirements are met.

- Minimal Install Rocky Linux 9
- 2 GB free RAM or more
- 2 CPUs or more
- 25 GB free disk space
- Local user with sudo admin rights
- Internet connectivity
- Docker or virtual machine manager like VirtualBox, KVM, and VMware etc.

Without any further delay, let's jump into the actual steps,

1) Enable Docker Repository

Open the terminal and run following dnf command,

```
$ sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
```

Run following dnf command to verify whether docker registry is enabled or not.

```
$ sudo dnf repolist
```

```
[linuxbuzz@rocky-linux9 ~]$ sudo dnf repolist
repo id                                repo name
appstream                              Rocky Linux 9 - AppStream
baseos                                 Rocky Linux 9 - BaseOS
docker-ce-stable                       Docker CE Stable - x86_64
extras                                 Rocky Linux 9 - Extras
[linuxbuzz@rocky-linux9 ~]$
```

2) Install Docker CE (Community Edition)

Install docker ce along with all dependencies using beneath command,

```
$ sudo dnf install docker-ce docker-ce-cli containerd.io -y
```

```
[linuxbuzz@rocky-linux9 ~]$ sudo dnf install docker-ce docker-ce-cli containerd.io -y
[sudo] password for linuxbuzz:
Docker CE Stable - x86_64
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
containerd.io	x86_64	1.6.18-3.1.el9	docker-ce-stable	32 M
docker-ce	x86_64	3:23.0.1-1.el9	docker-ce-stable	23 M
docker-ce-cli	x86_64	1:23.0.1-1.el9	docker-ce-stable	7.0 M
Installing dependencies:				
container-selinux	noarch	3:2.189.0-1.el9	appstream	47 k
docker-ce-rootless-extras	x86_64	23.0.1-1.el9	docker-ce-stable	3.8 M
fuse-common	x86_64	3.10.2-5.el9.0.1	baseos	8.1 k
fuse-overlayfs	x86_64	1.9-1.el9	appstream	71 k
fuse3	x86_64	3.10.2-5.el9.0.1	appstream	53 k
fuse3-libs	x86_64	3.10.2-5.el9.0.1	appstream	91 k
libslirp	x86_64	4.4.0-7.el9	appstream	68 k
slirp4netns	x86_64	1.2.0-2.el9	appstream	46 k
Installing weak dependencies:				
docker-buildx-plugin	x86_64	0.10.2-1.el9	docker-ce-stable	12 M
docker-compose-plugin	x86_64	2.16.0-1.el9	docker-ce-stable	11 M
docker-scan-plugin	x86_64	0.23.0-3.el9	docker-ce-stable	3.8 M
Transaction Summary				
Install 14 Packages				
Total download size: 93 M				
Installed size: 362 M				

Add your local user to docker group so that it can run docker commands without sudo,

```
$ sudo usermod -aG docker $USER
```

```
$ newgrp docker
```

Start the docker service, run

```
$ sudo systemctl start docker
```

```
$ sudo systemctl enable docker
```

To verify the docker installation, run following docker container,

```
$ docker run hello-world
```

Output,

```
[linuxbuzz@rocky-linux9 ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:6e8b6f026e0b9c419ea0fd02d3905dd0952ad1feea67543f525c73a0a790fefb
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

[linuxbuzz@rocky-linux9 ~]$
```

Output above confirms that docker has been installed successfully.

3) Download and Install Minikube and kubectl binary

To download and install minikube binary, run following commands,

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

Install kubectl binary, run beneath commands,

```
$ curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
$ sudo cp kubectl /usr/local/bin/ && sudo chmod +x /usr/local/bin/kubectl
```

4) Start the minikube cluster

To start the minikube cluster with docker as driver, run

```
$ minikube start --driver docker
```

Output

```
[linuxbuzz@rocky-linux9 ~]$ minikube start --driver docker
* minikube v1.29.0 on Rocky 9.1
* Using the docker driver based on user configuration

X The requested memory allocation of 1962MiB does not leave room for system overhead (total system memory: 1962MiB).
* Suggestion: Start minikube with less memory allocated: 'minikube start --memory=1962mb'

* 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% 14.34 M
  > gcr.io/k8s-minikube/kicbase... : 407.19 MiB / 407.19 MiB 100.00% 6.77 Mi
* Creating docker container (CPUs=2, Memory=1962MB) ...
* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (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
[linuxbuzz@rocky-linux9 ~]$
```

Perfect, above output confirms that the minikube cluster has been started successfully. Let's try to interact with cluster and deploy sample application,

Execute the following set of commands to view minikube status, Kubernetes cluster and node info,

```
$ minikube status
```

```
$ kubectl cluster-info
```

```
$ kubectl get nodes
```

```
[linuxbuzz@rocky-linux9 ~]$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

[linuxbuzz@rocky-linux9 ~]$ kubectl cluster-info
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/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[linuxbuzz@rocky-linux9 ~]$
[linuxbuzz@rocky-linux9 ~]$ kubectl get nodes
NAME             STATUS    ROLES    AGE   VERSION
minikube         Ready    control-plane   5m4s   v1.26.1
[linuxbuzz@rocky-linux9 ~]$
[linuxbuzz@rocky-linux9 ~]$
```

5) Test Kubernetes by Deploying Sample Nginx Application

To deploy sample nginx based application, run following kubectl commands,

```
$ kubectl create deployment nginx-demo --image=nginx
```

```
$ kubectl expose deployment nginx-demo --type NodePort --port=80
```

Verify pods and service status, run

```
$ kubectl get pods,svc
```

```
[linuxbuzz@rocky-linux9 ~]$ kubectl get pods,svc
NAME                                READY    STATUS    RESTARTS   AGE
pod/nginx-demo-98c8d48ff-2p7lh      1/1      Running   0           71s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
service/kubernetes                  ClusterIP     10.96.0.1     <none>        443/TCP          14m
service/nginx-demo                  NodePort      10.103.219.229 <none>        80:31707/TCP     32s
[linuxbuzz@rocky-linux9 ~]$
```

Try accessing the application using following command,

```
$ minikube service nginx-demo --url
```

```
http://192.168.49.2:31707
```

```
$ curl http://192.168.49.2:31707
```

```
[linuxbuzz@rocky-linux9 ~]$ minikube service nginx-demo --url http://192.168.49.2:31707
[linuxbuzz@rocky-linux9 ~]$ curl http://192.168.49.2:31707
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[linuxbuzz@rocky-linux9 ~]$
```

6) Managing Minikube Addons

Using minikube addons, we can enable additional functionality to our cluster like Kubernetes dashboard and nginx ingress controller. To view all available addons, run

```
$ minikube addons list
```

```
[linuxbuzz@rocky-linux9 ~]$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
auto-pause	minikube	disabled	Google
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	disabled	Kubernetes
default-storageclass	minikube	enabled ✓	Kubernetes
efk	minikube	disabled	3rd party (Elastic)
freshpod	minikube	disabled	Google
gcp-auth	minikube	disabled	Google
gvisor	minikube	disabled	Google
headlamp	minikube	disabled	3rd party (kinvolk.io)
helm-tiller	minikube	disabled	3rd party (Helm)
inaccel	minikube	disabled	3rd party (InAccel [info@inaccel.com])
ingress	minikube	disabled	Kubernetes
ingress-dns	minikube	disabled	Google
istio	minikube	disabled	3rd party (Istio)
istio-provisioner	minikube	disabled	3rd party (Istio)
kong	minikube	disabled	3rd party (Kong HQ)
kubevirt	minikube	disabled	3rd party (KubeVirt)
logviewer	minikube	disabled	3rd party (unknown)
metallb	minikube	disabled	3rd party (MetalLB)
metrics-server	minikube	disabled	Kubernetes
nvidia-driver-installer	minikube	disabled	Google
nvidia-gpu-device-plugin	minikube	disabled	3rd party (Nvidia)
olm	minikube	disabled	3rd party (Operator Framework)
pod-security-policy	minikube	disabled	3rd party (unknown)
portainer	minikube	disabled	3rd party (Portainer.io)
registry	minikube	disabled	Google
registry-aliases	minikube	disabled	3rd party (unknown)
registry-creds	minikube	disabled	3rd party (UPMC Enterprises)
storage-provisioner	minikube	enabled ✓	Google
storage-provisioner-gluster	minikube	disabled	3rd party (Gluster)
volumesnapshots	minikube	disabled	Kubernetes

To enable addons like dashboard and ingress controller, run following minikube commands,

```
$ minikube addons enable dashboard
```

```
$ minikube addons enable ingress
```

Output,

```
[linuxbuzz@rocky-linux9 ~]$ minikube addons enable dashboard
* dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
- Using image docker.io/kubernetesui/dashboard:v2.7.0
- Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

* The 'dashboard' addon is enabled
[linuxbuzz@rocky-linux9 ~]$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
- Using image registry.k8s.io/ingress-nginx/controller:v1.5.1
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20220916-gd32f8c343
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20220916-gd32f8c343
* Verifying ingress addon...
* The 'ingress' addon is enabled
[linuxbuzz@rocky-linux9 ~]$
```

7) Managing Minikube Cluster

Stop and start minikube cluster, run

```
$ minikube stop
$ minikube start
```

Run following command to delete minikube cluster,

```
$ minikube delete
```

Set the custom resources to minikube

```
$ minikube config set memory 4096
$ minikube stop && minikube start
```