LinuxTechi
LINUX TUTORIALS & GUIDE

How To Install Kubernetes On Rocky Linux 9 | AlmaLinux 9

By Pradeep Kumar / Last Updated: November 26, 2023 / 5 Minutes Of Reading

In this blog post, we will explain how to install Kubernetes cluster on Rocky Linux 9 or AlmaLinux 9 with Kubeadm utility.

Kubernetes, often referred to as K8s, is an open-source container orchestration platform. With its robust capabilities for automating deployment, scaling, and managing containerized applications, Kubernetes has become the go-to solution for DevOps teams worldwide.

Table of Contents

Prerequisites

- A fresh Installation of Rocky Linux 9 or AlmaLinux 9
- Sudo user with admin rights
- Minimum of 2 GB RAM, 2 vCPUs and 20 GB Disk Space
- A reliable Internet Connection

Lab Setup

We have used three Virtual machines with following specification.

- K8s-master01 – 192.168.1.190
- K8s-worker01 – 192.168.1.191
- K8s-worker02 – 192.168.1.192
- Sysops as sudo user on each node

Without any further delay, lets deep dive into Kubernetes installation steps.

Step 1: Set Hostname And Update Hosts File

Login or ssh each machine and run hostnamectl commands to set their respective hostname.

```
$ sudo hostnamectl set-hostname "k8s-master01" && exec bash
$ sudo hostnamectl set-hostname "k8s-worker01" && exec bash
$ sudo hostnamectl set-hostname "k8s-worker02" && exec bash
```

Add the following entries in /etc/hosts file on each node.

```
192.168.1.190   k8s-master01
192.168.1.191   k8s-worker01
192.168.1.192   k8s-worker02
```

Step 2: Disable Swap Space On Each Node

For kubelet to work smoothly, we must disable swap space on all the nodes. Run beneath command,

```
$ sudo swapoff -a
```

```
$ sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

Step 3: Adjust SELinux And Firewall Rules For Kubernetes

Set SELinux mode as permissive on all the nodes using following commands,

```
$ sudo setenforce 0
$ sudo sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=permissive/g' /etc/sysconfig/selinux
```

On the master node, allow following ports in the firewall.

```
$ sudo firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252,10257,10259,179}/tcp
$ sudo firewall-cmd --permanent --add-port=4789/udp
$ sudo firewall-cmd --reload
```

On the Worker Nodes, allow beneath ports in the firewall,

```
$ sudo firewall-cmd --permanent --add-port={179,10250,30000-32767}/tcp
$ sudo firewall-cmd --permanent --add-port=4789/udp
$ sudo firewall-cmd --reload
```

Step 4: Add Kernel Modules And Parameters

For kuberetes cluster, we must add the overlay and br_netfilter kernel modules on all the nodes.

Create a file and add following content to it,

```
$ sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
```

In order to load above modules, run

```
$ sudo modprobe overlay
$ sudo modprobe br_netfilter
```

Next, add the following kernel parameters, create a file and with following content,

```
$ sudo vi /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

Save & close the file.

Now add these parameters by running below command

```
$ sudo sysctl --system
```

Step 5: Install Conatinerd Runtime

Kubernetes requires a container runtime, and one of the most popular choices is containerd. But It is not available in the default package repositories of Rocky Linux or AlmaLinux, so add the following docker repo on all the nodes.

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

Now, run following dnf command to install containerd on all the nodes.

```
$ sudo dnf install containerd.io -y
```

```
[sysops@k8s-master01 ~]$ sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ sudo dnf install containerd.io -y
Docker CE Stable - x86_64                                                                                3.0 kB/s |  31 kB     00:10
Dependencies resolved.
================================================================================================================================================
 Package                          Architecture             Version                                Repository                           Size
================================================================================================================================================
Installing:
 containerd.io                    x86_64                   1.6.22-3.1.el9                         docker-ce-stable                     33 M
Installing dependencies:
 container-selinux                noarch                   3:2.205.0-1.el9_2                      appstream                            50 k

Transaction Summary
================================================================================================================================================
Install  2 Packages
```

Configure containerd so that it will use systemdcgroup, execute the following commands on each node.

```
$ containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
$ sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
```

Restart and enable containerd service using beneath commands,

```
$ sudo systemctl restart containerd
$ sudo systemctl enable containerd
```

Verify conatinerd service status, run

```
$ sudo systemctl status containerd
```

```
[sysops@k8s-master01 ~]$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: disabled)
     Active: active (running) since Sat 2023-09-16 15:19:54 IST; 1min 1s ago
       Docs: https://containerd.io
   Main PID: 2422 (containerd)
      Tasks: 8
     Memory: 14.7M
        CPU: 410ms
     CGroup: /system.slice/containerd.service
             └─2422 /usr/bin/containerd

Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.665571946+05:30" level=info msg=serving... address=/run/containerd/container>
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.665668769+05:30" level=info msg=serving... address=/run/containerd/container>
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.666646433+05:30" level=info msg="Start subscribing containerd event"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.666901360+05:30" level=info msg="Start recovering state"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.666997490+05:30" level=info msg="Start event monitor"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.667021811+05:30" level=info msg="Start snapshots syncer"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.667033541+05:30" level=info msg="Start cni network conf syncer for default"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.667041932+05:30" level=info msg="Start streaming server"
Sep 16 15:19:54 k8s-master01 containerd[2422]: time="2023-09-16T15:19:54.669232779+05:30" level=info msg="containerd successfully booted in 0.057829s"
Sep 16 15:19:54 k8s-master01 systemd[1]: Started containerd container runtime.
```

Step 6: Install Kubernetes Tools

Kubernetes tools like Kubeadm, kubectl and kubelet are not available in the default package repositories of Rocky Linux 9 or AlmaLinux 9. So, to install these tools, add the following repository on all the nodes.

```
$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

```
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[sysops@k8s-master01 ~]$
```

Note: At time of writing this post, Kubernetes 1.28 version was available, that's why I have mentioned v1.28 while adding the repo.

Next, install Kubernetes tools by running following dnf command,

```
$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

```
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:10:47 ago on Sat 16 Sep 2023 03:28:10 PM IST.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version              Repository      Size
================================================================================
Installing:
 kubeadm                 x86_64        1.28.2-150500.1.1    kubernetes      9.9 M
 kubectl                 x86_64        1.28.2-150500.1.1    kubernetes       10 M
 kubelet                 x86_64        1.28.2-150500.1.1    kubernetes       19 M
Installing dependencies:
 conntrack-tools         x86_64        1.4.7-2.el9          appstream       221 k
 cri-tools               x86_64        1.28.0-150500.1.1    kubernetes      8.1 M
 kubernetes-cni          x86_64        1.2.0-150500.2.1     kubernetes      6.2 M
 libnetfilter_cthelper   x86_64        1.0.0-22.el9         appstream        23 k
 libnetfilter_cttimeout  x86_64        1.0.0-19.el9         appstream        23 k
 libnetfilter_queue      x86_64        1.0.5-1.el9          appstream        28 k
 socat                   x86_64        1.7.4.1-5.el9        appstream       300 k

Transaction Summary
================================================================================
Install  10 Packages
```

After installing Kubernetes tools, start the kubelet service on each node.

```
$ sudo systemctl enable --now kubelet
```

Step 7: Install Kubernetes Cluster On Rocky Linux 9 / Alma Linux 9

Now, we are all set to install Kubernetes cluster. Run beneath Kubeadm command to initialize the Kubernetes cluster from the master node.

```
$ sudo kubeadm init --control-plane-endpoint=k8s-master01
```

Once above command is executed successfully, we will get following output,

```
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

  kubeadm join k8s-master01:6443 --token 69s57o.3muk7ey0j0zknw69 \
        --discovery-token-ca-cert-hash sha256:8000dff8e803e2bf687f3dae80b4bc1376e5bd770e7a752a3c9fa314de6449fe \
        --control-plane

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join k8s-master01:6443 --token 69s57o.3muk7ey0j0zknw69 \
        --discovery-token-ca-cert-hash sha256:8000dff8e803e2bf687f3dae80b4bc1376e5bd770e7a752a3c9fa314de6449fe
[sysops@k8s-master01 ~]$
```

From the output above make a note of the command which will be executed on the worker nodes to join the Kubernetes cluster.

To start interacting with Kubernetes cluster, run the following commands on the master node.

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Next, join the worker nodes to the cluster, run following Kubeadm command from the worker nodes.

```
$ kubeadm join k8s-master01:6443 --token 69s57o.3muk7ey0j0zknw69 \
  --discovery-token-ca-cert-hash sha256:8000dff8e803e2bf687f3dae80b4bc1376e5bd770e7a752a3c9fa314de6449fe
```

Output from Worker01

```
[sysops@k8s-worker01 ~]$
[sysops@k8s-worker01 ~]$ sudo kubeadm join k8s-master01:6443 --token 69s57o.3muk7ey0j0zknw69 \
        --discovery-token-ca-cert-hash sha256:8000dff8e803e2bf687f3dae80b4bc1376e5bd770e7a752a3c9fa314de6449fe
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[sysops@k8s-worker01 ~]$
```

Output from Worker02

```
[sysops@k8s-worker02 ~]$
[sysops@k8s-worker02 ~]$ sudo kubeadm join k8s-master01:6443 --token 69s57o.3muk7ey0j0zknw69 \
        --discovery-token-ca-cert-hash sha256:8000dff8e803e2bf687f3dae80b4bc1376e5bd770e7a752a3c9fa314de6449fe
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[sysops@k8s-worker02 ~]$
```

Now, head back to master node and run kubectl command to verify the nodes status.

```
$ kubectl get nodes
```

```
[sysops@k8s-master01 ~]$ kubectl get nodes
NAME           STATUS     ROLES           AGE     VERSION
k8s-master01   NotReady   control-plane   16m     v1.28.2
k8s-worker01   NotReady   <none>          2m51s   v1.28.2
k8s-worker02   NotReady   <none>          106s    v1.28.2
[sysops@k8s-master01 ~]$
```

Output above shows that nodes is “NoteRead”, so to make the nodes status “Ready”, install Calico network addon or plugin in the next step.

Step 8: Install Calico Network Addon

Calico network addon is required on Kubernetes cluster to enable communication between pods, to make DNS service function with the cluster and to make the nodes status as Ready.

In order to install calico CNI (Container Network Interface) addon, run following kubectl commands from the master node only.

```
$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
```

```
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
[sysops@k8s-master01 ~]$
```

Verify calico pods status,

```
$ kubectl get pods -n kube-system
```

```
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl get pods -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
calico-kube-controllers-7ddc4f45bc-5bxmt  1/1     Running   0          3m17s
calico-node-86q6k                         1/1     Running   0          3m17s
calico-node-gd7ng                         1/1     Running   0          3m17s
calico-node-wpmlq                         1/1     Running   0          3m17s
coredns-5dd5756b68-tmhzr                  1/1     Running   0          33m
coredns-5dd5756b68-ztwck                  1/1     Running   0          33m
etcd-k8s-master01                         1/1     Running   0          33m
kube-apiserver-k8s-master01               1/1     Running   0          33m
kube-controller-manager-k8s-master01      1/1     Running   0          33m
kube-proxy-62tcn                          1/1     Running   0          33m
kube-proxy-djlc5                          1/1     Running   0          19m
kube-proxy-rnlfn                          1/1     Running   0          18m
kube-scheduler-k8s-master01               1/1     Running   0          33m
[sysops@k8s-master01 ~]$
```

Next, verify the nodes status, this time nodes status should be in Ready State.

```
$ kubectl get nodes
```

```
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
k8s-master01   Ready    control-plane   38m   v1.28.2
k8s-worker01   Ready    <none>          24m   v1.28.2
k8s-worker02   Ready    <none>          23m   v1.28.2
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$
```

Perfect, output above confirms nodes are in Ready state and can handle workload. Let's test our Kubernetes installation the next step.

Step 9: Test Kubernetes Cluster Installation

To test Kubernetes cluster installation, let's try to deploy nginx based application using deployment. Run following kubectl commands,

```
$ kubectl create deployment web-app01 --image nginx --replicas 2
$ kubectl expose deployment web-app01 --type NodePort --port 80
$ kubectl get deployment web-app01
$ kubectl get pods
$ kubectl get svc web-app01
```

```
[sysops@k8s-master01 ~]$ kubectl create deployment web-app01 --image nginx --replicas 2
deployment.apps/web-app01 created
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl expose deployment web-app01 --type NodePort --port 80
service/web-app01 exposed
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl get deployment web-app01
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
web-app01   2/2     2            2           66s
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
web-app01-7b6d698dc6-68ndv   1/1     Running   0          72s
web-app01-7b6d698dc6-tp9fk   1/1     Running   0          72s
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$ kubectl get svc web-app01
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
web-app01   NodePort   10.110.68.31   <none>        80:31121/TCP   41s
[sysops@k8s-master01 ~]$
[sysops@k8s-master01 ~]$
```

Try to access the application using nodeport "31121", run following curl command,

```
$ curl k8s-worker01:31121
```

Great, above confirms that we can access our application web page. This also confirms that our Kubernetes cluster has been installed successfully.

That's all from this post, we believe that you have found informative and useful. If you have any queries and feedback, please do post it in below comments section.

Also Read: How to Install Kubernetes Dashboard Using Helm

About The Author

Pradeep Kumar

I am a Cloud Consultant with over 15 years of experience in Linux, Kubernetes, cloud technologies (AWS, Azure, OpenStack), automation (Ansible, Terraform), and DevOps. I hold certifications like RHCA, CKA, CKAD, CKS, AWS, and Azure.

← Previous Post Next Post →

1 Thought On "How To Install Kubernetes On Rocky Linux 9 | AlmaLinux 9"

RAVI
OCTOBER 1, 2024 AT 4:21 PM

Thanks Pradeep for the straight forward guide. It worked fine.

Reply

Leave A Comment

Your email address will not be published. Required fields are marked *

Type here..

Name* Email* Website

Post Comment »

Search...

Join Our Newsletter!

*Name**

*Email Address**

Subscribe

Recent Posts

How to Install Elementary OS 8 Step-by-Step
How to Install KVM on Debian 12 Step-by-Step
How to Dual Boot Windows 11 And Ubuntu 24.04
How to Install Git on RHEL 9 | Rocky Linux 9
How to Install KVM on Ubuntu 24.04 Step-by-Step
How to Install Python 3.12 On Ubuntu 24.04
How to Install Elastic (ELK) Stack on Ubuntu 24.04
How to Setup SFTP Server On Ubuntu 24.04
How to Setup SFTP Server on Debian 12
How to Install PHP 8.3 on Ubuntu 24.04 (Simple Guide)
How to Add New Kubernetes Worker Node Step-by-Step
How to Install Python 3 on RHEL 9 (Red Hat Enterprise Linux)