

Alpaquita Linux

Setting up a Kubernetes cluster



Alpaquita Linux
Revision 1.0
October 2026

be//soft

Copyright © BellSoft Corporation 2018-2026.

BellSoft software contains open source software. Additional information about third party code is available at https://bell-sw.com/third_party_licenses. You can also get more information on how to get a copy of source code by contacting info@bell-sw.com.

THIS INFORMATION MAY CHANGE WITHOUT NOTICE. TO THE EXTENT PERMITTED BY APPLICABLE LAW, BELLSOFT PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL BELLSOFT BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF BELLSOFT IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in this document is governed by the applicable license agreement, which is not modified in any way by the terms of this notice.

Alpaquita, Liberica and BellSoft are trademarks or registered trademarks of BellSoft Corporation. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Java and OpenJDK are trademarks or registered trademarks of Oracle and/or its affiliates. Other trademarks are the property of their respective owners and are used only for identification purposes.

Contents

1. Overview	5
2. Prerequisites	6
3. Initial setup on each node	7
Kernel configuration	7
Mount propagation	7
Installing packages	8
Flannel	8
Cilium	9
4. Initializing the control plane node	10
Flannel	10
Calico	10
Cilium	11
5. Adding the working node to the cluster	13

6. Verifying the cluster	14
--------------------------	----

7. See also	15
-------------	----

1. Overview

This document describes how to set up a Kubernetes cluster on Alpaquita Linux with one control plane and one worker nodes using one of the following Container Network Interface (CNI) plugins:

- Flannel
- Calico
- Cilium

2. Prerequisites

Set up two Alpaquita Linux hosts: one for a control-plane node and another for a worker node. Minimum hardware requirements for each node:

- 2 CPUs
- 4 GB of memory
- 10 GB of disk space

Also, make sure that swap memory is disabled on both hosts since by default Kubernetes fails to start if swap memory is detected on a node.

3. Initial setup on each node

This part describes steps to perform on both control plane and worker nodes.

Kernel configuration

For proper networking functionality of the cluster, the `br_netfilter` kernel module must be loaded as follows:

```
# modprobe br_netfilter
```

Enable the following kernel parameters:

```
# sysctl net.ipv4.ip_forward=1
# sysctl net.bridge.bridge-nf-call-iptables=1
```

To make these changes effective on each boot, do the following:

```
# echo "br_netfilter" > /etc/modules-load.d/k8s.conf
# echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
# echo "net.bridge.bridge-nf-call-iptables=1" >> /etc/sysctl.conf
```

Mount propagation

Calico and Cilium CNI plugins require `/sys` and `/run` mount points to have a recursively shared mount propagation type:

```
# mount --make-rshared /sys
# mount --make-rshared /run
```

Add these commands to a script for a `local` service to run at each boot:

```
# cat >> /etc/local.d/k8s.start <<EOF
#!/bin/sh

mount --make-rshared /sys
mount --make-rshared /run
EOF
# chmod +x /etc/local.d/k8s.start
```

Enable the local service:

```
# rc-update add local
```

You can check if the filesystems are mount properly using a `findmnt` utility as follows:

```
# findmnt -o TARGET,PROPAGATION | grep shared
└-/sys                                shared
└└-/sys/kernel/security               shared
└└-/sys/kernel/debug                 shared
└└-/sys/kernel/tracing               shared
└└-/sys/fs/pstore                    shared
└└-/sys/fs/bpf                      shared
└-/run                              shared
```

Installing packages

Kubernetes needs a container runtime where pods can be run. Use the `containerd` for this purpose on Alpaquita Linux.

Run the following commands to install `containerd` and to enable and start a `containerd` service:

```
# apk add containerd
# rc-update add containerd
# rc-service containerd start
```

After that install all basic kubernetes packages and enable a `kubelet` service:

```
# apk add cni-plugins kubelet kubeadm kubectrl
# rc-update add kubelet
```

Note that Calico and Cilium CNI plugins search for standard networking CNI plugins in a `/opt/cni/bin` directory but on Alpaquita Linux they are installed into a `/usr/libexec/cni/` directory. This is why you need to create a symbolic link to it:

```
# mkdir -p /opt/cni/
# ln -s /usr/libexec/cni /opt/cni/bin
```

Flannel

If you want to use Flannel for networking, install the `cni-plugin-flannel` package:

```
# apk add cni-plugin-flannel
```

Cilium

If you want to use Cilium for networking, install the `cilium-cli` package:

```
# apk add cilium-cli
```

4. Initializing the control plane node

Run the following command on the control plane node:

```
# kubeadm init --pod-network-cidr=10.244.0.0/16 --node-name=$(hostname)
```

10.244.0.0/16 is a default Flannel pod network CIDR (a range of IP addresses for the pod network,) but it can be also used for other CNI plugins.

After that set up a `kubectl` configuration for the `root` user:

```
# mkdir -p /root/.kube  
# ln -s /etc/kubernetes/admin.conf /root/.kube/config
```

Flannel

To install the Flannel CNI plugin, run:

```
# kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml
```

Then wait till Flannel pods become ready:

```
# kubectl -n kube-flannel wait --for=condition=ready pod --selector=app=flannel
```

For more information about Flannel installation, see the [Flannel project on GitHub](https://github.com/flannel-io/flannel).

Calico

To install the Calico CNI plugin, first download its manifest file:

```
# wget  
https://raw.githubusercontent.com/projectcalico/calico/v3.31.7/manifests/calico  
.yaml
```

In the current example, we are using version v3.31.7, but you can try the most recent stable one.

Then adjust the pod network CIDR in the manifest file so it matches the one we used in the `kubeadm init` command:

```
# sed -i -e '/CALICO_IPV4POOL_CIDR/s;# ;;' -e '/192.168.0.0\16/{s/#  
//;s#192.168.0.0/16#10.244.0.0/16#}' calico.yaml
```

After that apply the manifest file:

```
# kubectl apply -f calico.yaml
```

Then wait till Calico pods become ready:

```
# kubectl -n kube-system wait --for=condition=ready pod --selector=k8s  
-app=calico-node
```

For more information about Calico installation, see [Install Calico networking and network policy for on-premises deployments](#).

Cilium

Cilium can operate in two device modes:

- `veth`
- `netkit`

`veth` is the default device mode whereas `netkit` is a newer one that is considered to be more performant.

For more information about `netkit`, see [Cilium netkit: The Final Frontier in Container Networking Performance](#).

To install the Cilium CNI plugin with the `veth` device mode, run:

```
# cilium install --set ipam.mode=kubernetes
```

To install the Cilium CNI plugin with the `netkit` device mode, run:

```
# cilium install --set ipam.mode=kubernetes \  
  --set routingMode=native \  
  --set bpf.datapathMode=netkit \  
  --set bpf.masquerade=true \  
  --set kubeProxyReplacement=true \  
  --set ipv4NativeRoutingCIDR=10.244.0.0/16 \  
  --set autoDirectNodeRoutes=true
```

After that wait till Cilium pods become ready:

```
# cilium status --wait
```

For more information about Cilium installation, see the [Cilium Quick Installation document](#).

5. Adding the working node to the cluster

On the control plane node, get a join command:

```
# kubeadm token create --print-join-command
```

After that run the join command on the worker node.

Then wait till the worker node becomes ready by running this command on the control plane node:

```
# kubectl wait --for=condition=ready node <worker_node_hostname>
```

Where <worker_node_hostname> is a host name of the worker node.

6. Verifying the cluster

You can check the status of all nodes and pods by running the following commands on the control plane node:

```
# kubectl get nodes -o wide
# kubectl get pods -A -o wide
```

Then you can try running your own pod in the cluster, for example:

```
# kubectl run alpaquita --image=bellsoft/alpaquita-linux-base -it
All commands and output from this session will be recorded in container logs,
including credentials and sensitive information passed through the command
prompt.
If you don't see a command prompt, try pressing enter.
/ # cat /etc/os-release
NAME="BellSoft Alpaquita Linux"
ID=alpaquita
ID_LIKE=alpine
VERSION="Stream"
VERSION_ID=stream
PRETTY_NAME="BellSoft Alpaquita Linux Stream (musl)"
HOME_URL="https://bell-sw.com/"
BUG_REPORT_URL="https://bell-sw.com/support/"
LIBC_TYPE=musl
```

7. See also

See the [Alpine Linux: K8s in 10 Minutes](#) wiki article for more information.



Alpaquita Linux

Setting up a Kubernetes cluster

be//soft