White Paper
CPublic

Deploy K3s on SUSE Linux Enterprise Micro and Cisco UCS C220, C240, and C240 SD

Contents

Purpose of this document

This document provides a high-level procedure for deploying the K3s lightweight Kubernetes distribution on Cisco UCS® C220, C240, and C240 SD Rack Servers in space-constrained locations. The focus will be on areas where the deployment deviates from default installations. Everything that is not specified in this document can be configured based on the default settings for your local environment.

Introduction

During the past few years, organizations have been participating in a radical transformation of the way that modern applications are built, deployed, and operated. Monolithic applications are being broken down into microservices and serverless functions to ease development exponentially, facilitate lifecycle management, increase the speed at which new features are deployed, and improve the availability of services offered.

More and more mission-critical workloads have become containerized. According to various Gartner and IDC estimates, between 35 and 50 percent of an enterprise's application sprawl is now containerized—and not just the application front ends or the dashboards, but mission-critical workloads such as revenue-generating data analytics pipelines, middleware, and core business logic.

Not only are workloads and applications changing, but the locations at which data is generated, accessed, and partially processed are changing from the data center to a highly distributed world. Hybrid cloud, edge, the Internet of Things (IoT), and similar technologies are becoming the default for more and more companies, and IT departments must find ways to deploy, manage, and support containerized workloads at nearly every place: in the data center, at the shop floor, in vehicles, and in the public cloud.

This document provides a sample configuration for deploying a container platform on a single server that provides all the capabilities of the data center while fitting into a shortened network rack at the shop floor or edge location: the Cisco UCS C240 SD Rack Server. For the operating system, the solution uses SUSE Enterprise Linux (SLE) Micro: an optimized container option based on the proven enterprise-class Linux distribution. The lightweight Kubernetes service K3s, which is optimized to run on a single server, eliminates the need to install multiple servers.

About Cisco Unified Computing System

The solution uses a Cisco UCS C240 M5SX Rack Server with solid-state disks (SSDs) and hard-disk drives (HDDs). The configuration can be used with any Cisco UCS C-Series Rack Server.

Cisco UCS C240 M5 Rack Server overview

The Cisco UCS C240 M5 Rack Server is an enterprise-class server in a 2-rack-unit (2RU) form factor. It is designed to deliver exceptional performance, expandability, and efficiency for storage and I/O-intensive infrastructure workloads. These workloads include big data analytics, virtualization, and graphics-intensive and bare-metal applications.

The Cisco UCS C240 M5 server provides:

- Support for a 2RU 2-socket server using Intel® Xeon® Scalable processors
- Support for 2666-MHz DDR4 DIMMs and 128-GB DIMMs
- Increased storage density with 24 front-pluggable 2.5-inch small-form-factor (SFF) drive bays, or 12 front-pluggable 3.5-inch large-form-factor (LFF) drive bays and 2 rear 2.5-inch SFF drive bays

- Non-Volatile Memory Express (NVMe) PCI Express (PCIe) SSD support (for up to 2 drives on the standard chassis SKU or up to 10 drives on the NVMe-optimized SKU)
- Cisco® 12-Gbps SAS RAID modular controller and Cisco 12-Gbps SAS host bus adapter (HBA) controller
- 2 Flexible Flash (FlexFlash) Secure Digital (SD) card slots or 2 modular M.2 SATA slots
- 10-Gbps embedded Intel x550 10GBASE-T LAN-on-motherboard (LOM) port
- 1 modular LOM (mLOM) slot
- 6 PCIe Generation 3 (Gen 3) slots
- Up to 2 hot-pluggable redundant power supplies

The Cisco UCS C240 M5 server can be deployed as a standalone device or as part of a managed Cisco Unified Computing System™ (Cisco UCS) environment. Cisco UCS unifies computing, networking, management, virtualization, and storage access into a single integrated architecture that can enable end-to-end server visibility, management, and control in both bare-metal and virtualized environments. With a Cisco UCS managed deployment, the Cisco UCS C240 M5 takes advantage of our standards-based unified computing innovations to significantly reduce customers' total cost of ownership (TCO) and increase business agility.

About SUSE Linux Enterprise Micro

SUSE Linux Enterprise, or SLE, Micro is an ultra-reliable, lightweight operating system purpose-built for containerized and virtualized workloads. It uses the enterprise-hardened security and compliance components of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.

About K3s lightweight Kubernetes

K3s is packaged as a single binary about 50 MB in size. Bundled in that single binary is everything needed to run Kubernetes anywhere, including low-powered IoT and edge-based devices. The binary includes:

- The container runtime
- Any essential host utilities, such as iptables, socat and, du

The only OS dependencies are the Linux kernel itself and proper dev, proc, and sysfs mounts (these are included automatically in all modern Linux distributions).

K3s bundles these Kubernetes components:

- kube-apiserver
- kube-controller-manager
- kube-scheduler
- kubelet
- kube-proxy

Solution overview

The Cisco Integrated Management Console (IMC) is the basic option for managing Cisco UCS C-Series Rack Servers over the network and installing the operating system using the virtual console. SUSE Rancher is used to manage the Kubernetes installations in the data center, in branch offices, at the edge, and in the public cloud (Figure 1).

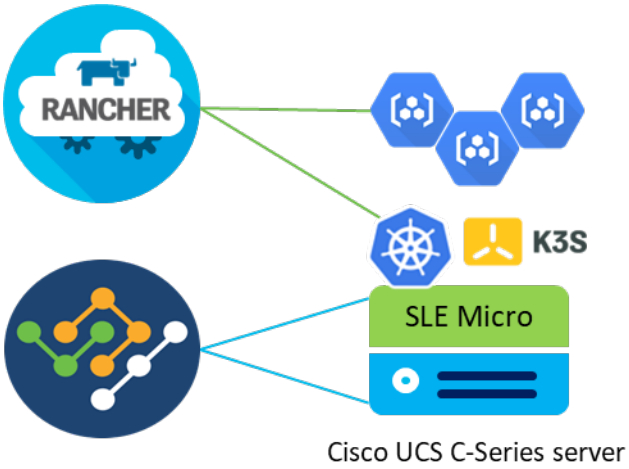


Figure 1.
Solution overview with Cisco IMC and SUSE Rancher

Prerequisites

The following items need to be preconfigured before you begin the setup and configuration of a K3s system on a Cisco UCS C240 SD server:

- Linux host with kubectl client binary installed and access to the Internet to download required software packages
- One Cisco UCS C240 SD racked and cabled
- Domain Host Configuration Protocol (DHCP) server to provide an IP address to the Cisco IMC
- Monitor, keyboard, and mouse for initial IMC configuration

Configure Cisco UCS C240 SD through the Cisco IMC

Use the procedure described in this section to prepare the Cisco UCS C-Series server for the SLE Micro installation. The main focus here is the configuration of the storage and the network.

The configuration steps presented here are for Cisco UCS servers that are not connected to a pair of Cisco UCS fabric interconnects (Cisco UCS managed mode) or to the Cisco Intersight™ platform. All configuration steps are performed on the local Cisco IMC.

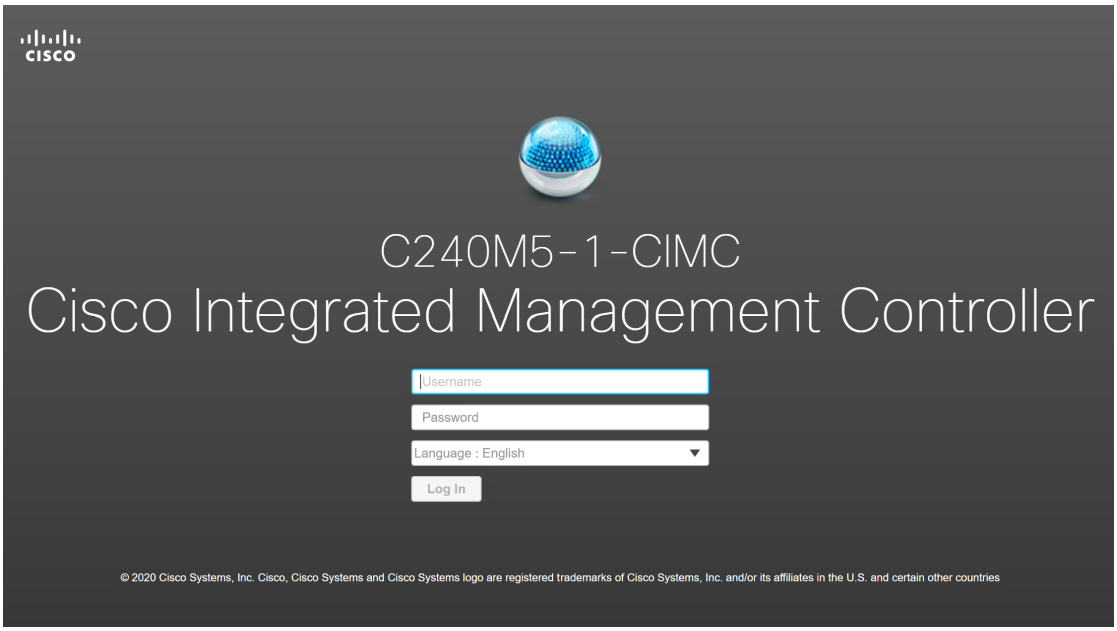
Perform initial setup

Hardware installation details and the initial server setup process are documented in the used server's installation documentation. For Cisco the UCS C240 SD M5, the document can be found here: https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/c/hw/c240sdm5/install/c240sdm5/C240M5_chapter_01.html.

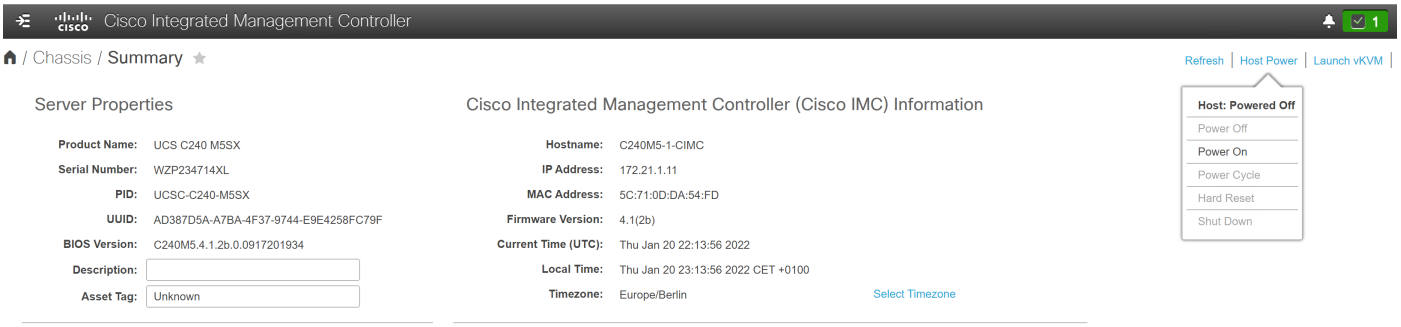
Configure the storage and network

Follow these steps to configure the storage and network for SLE Micro and K3s:

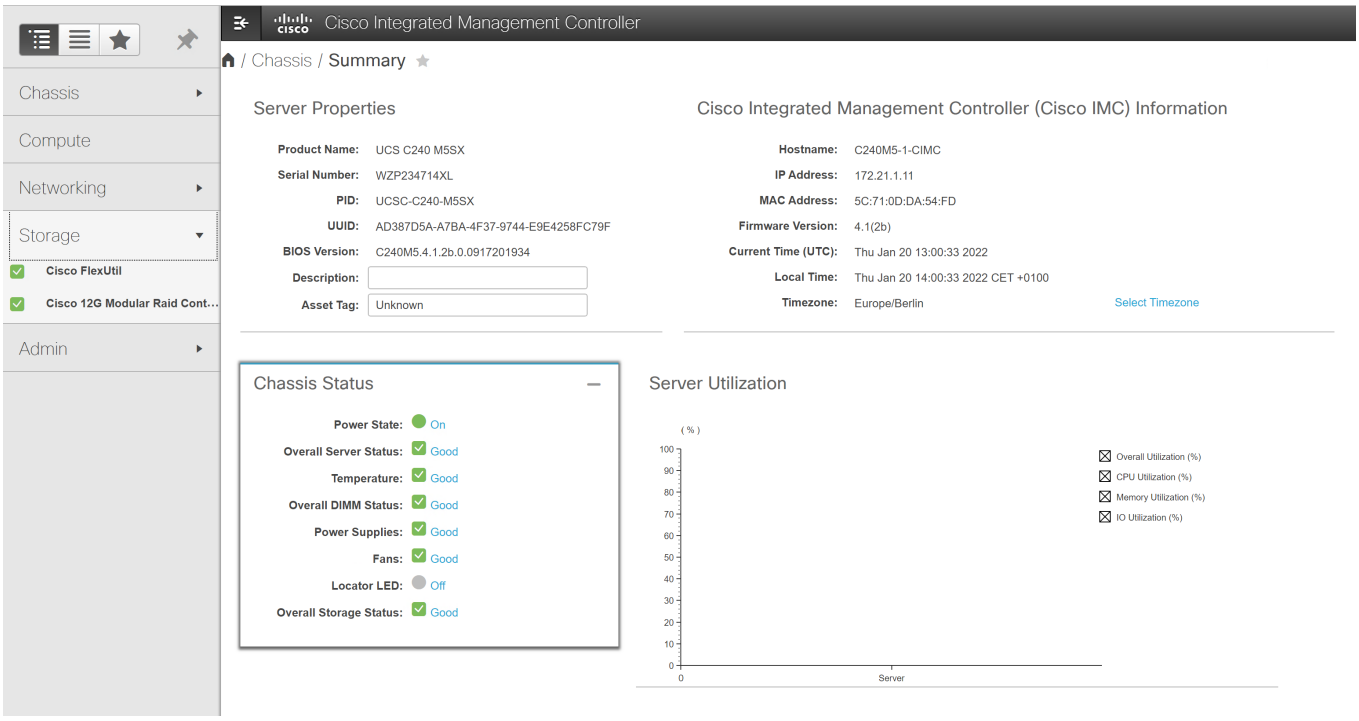
1. Open the IMC page in a web browser.



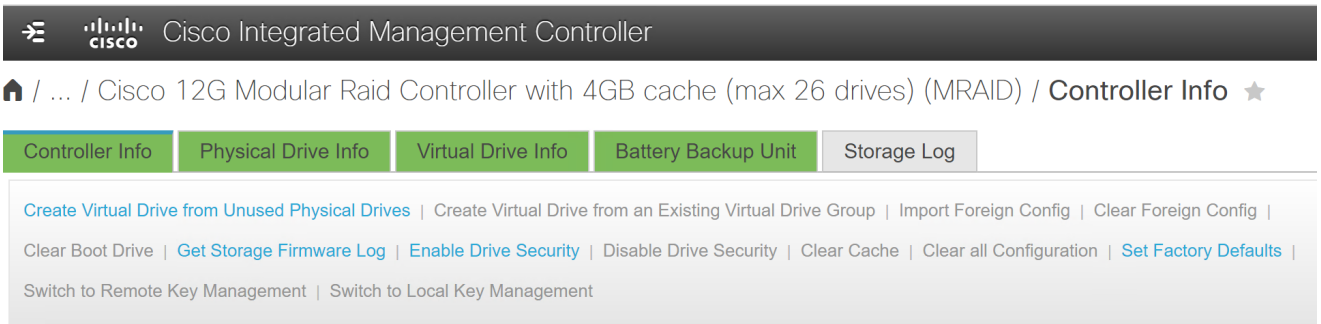
2. Enter the username and password and click Log In.
3. Some of the configuration steps require the server to be powered on. At the top right side of the window, click Host Power > Power On and then click OK in the pop-up window. Wait 30 seconds before proceeding to the next step.



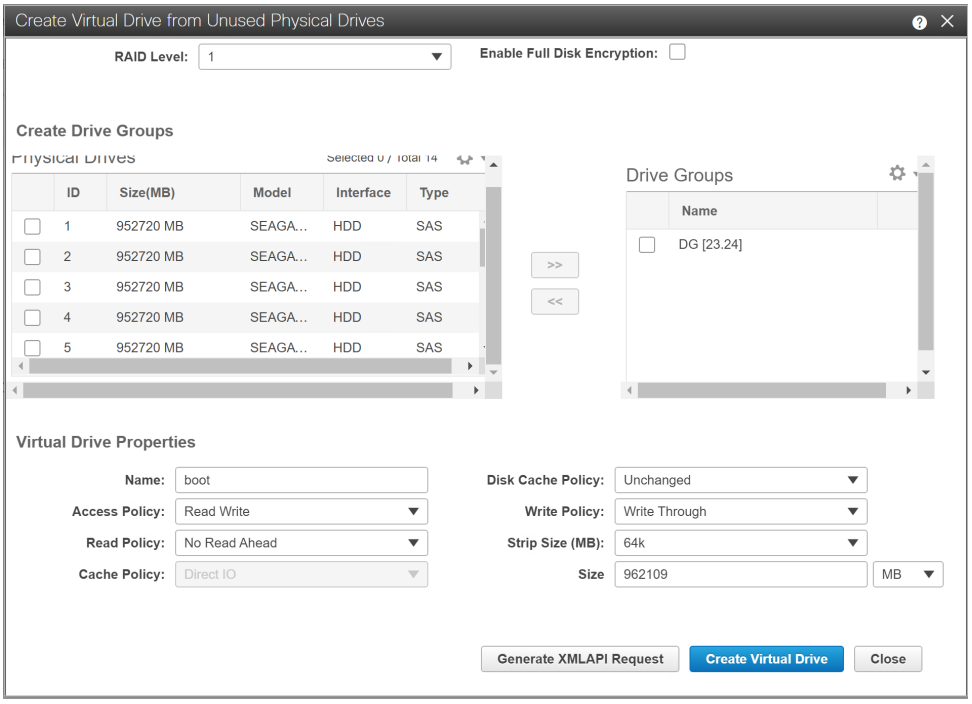
4. In the left menu, click Storage > Cisco 12G Modular Raid Controller.



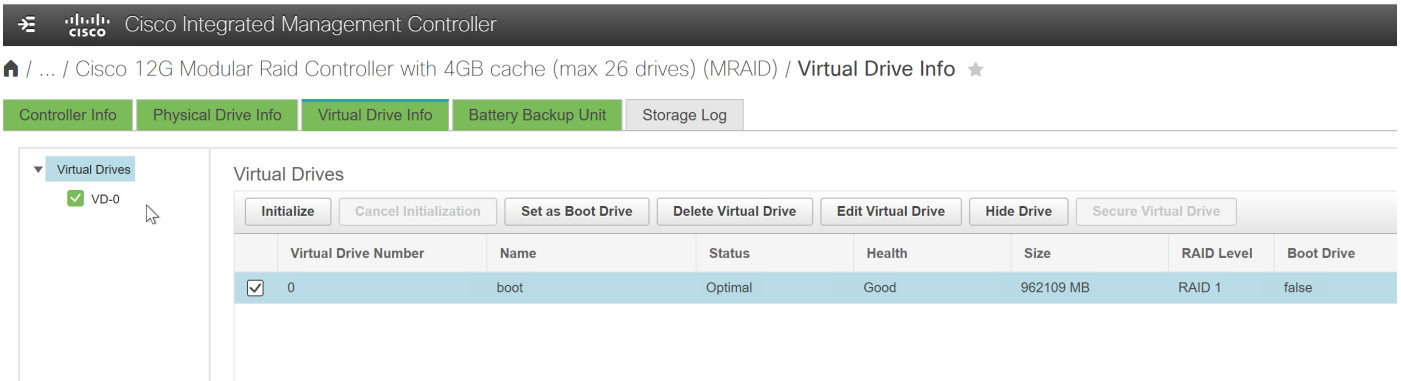
5. Click Create Virtual Drive from Unused Physical Drives.



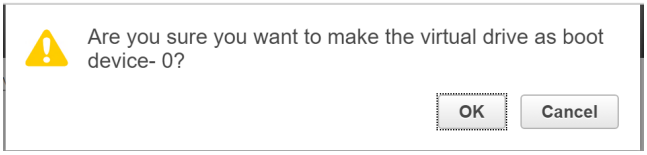
6. Select RAID Level 1, pick two SSD drives for the operating system, and click the >> button.
7. Under Virtual Drive Properties, enter **boot** as the name and then click Create Virtual Drive.



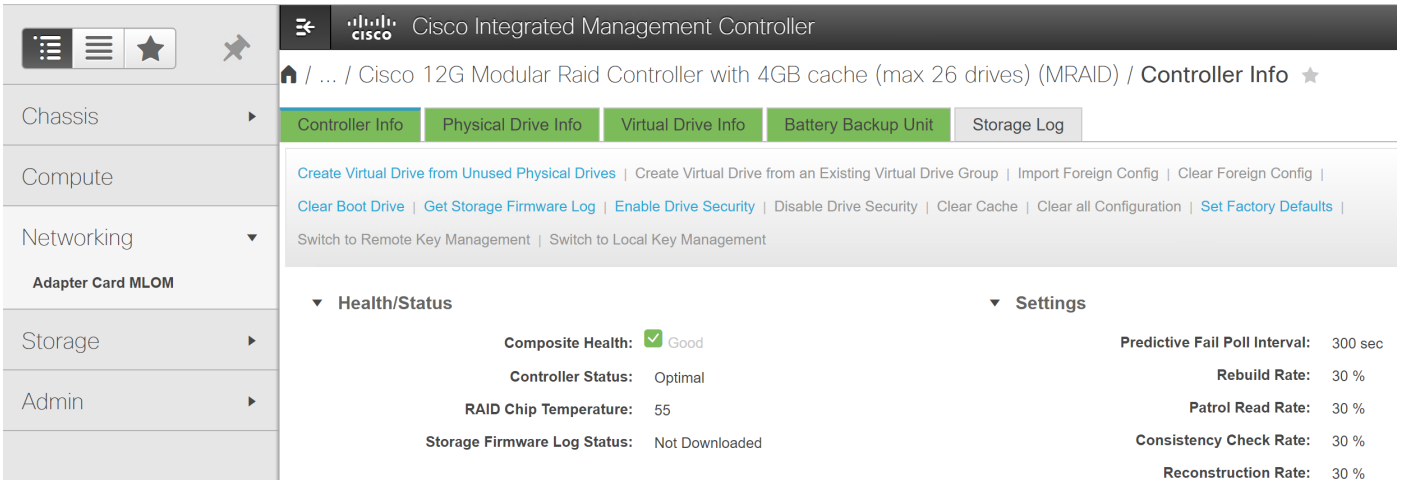
8. Click Virtual Drive Info, select the boot virtual drive in the list, and click Set as Boot Drive.



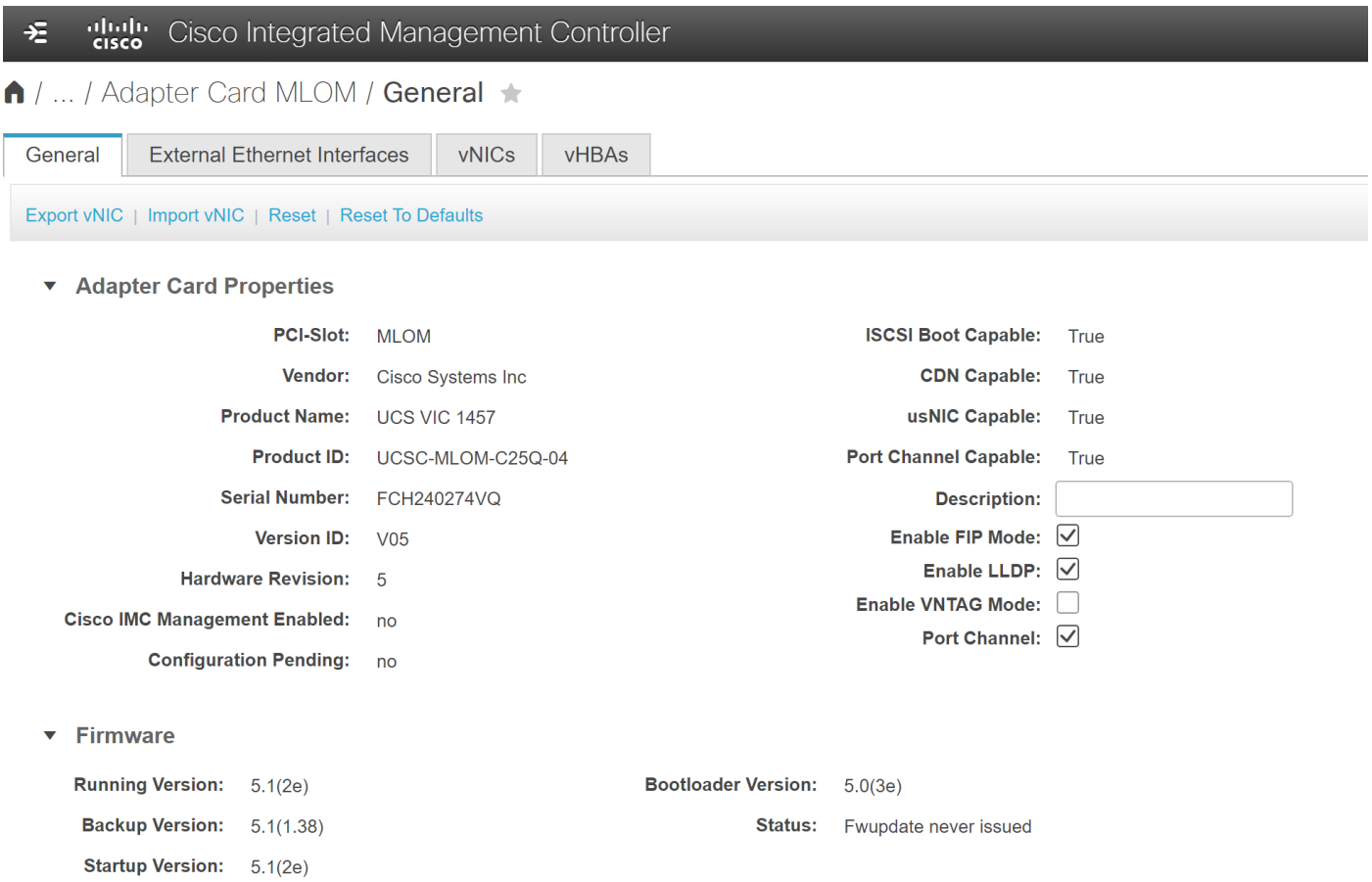
9. In the pop-up window, click OK to confirm the selection.



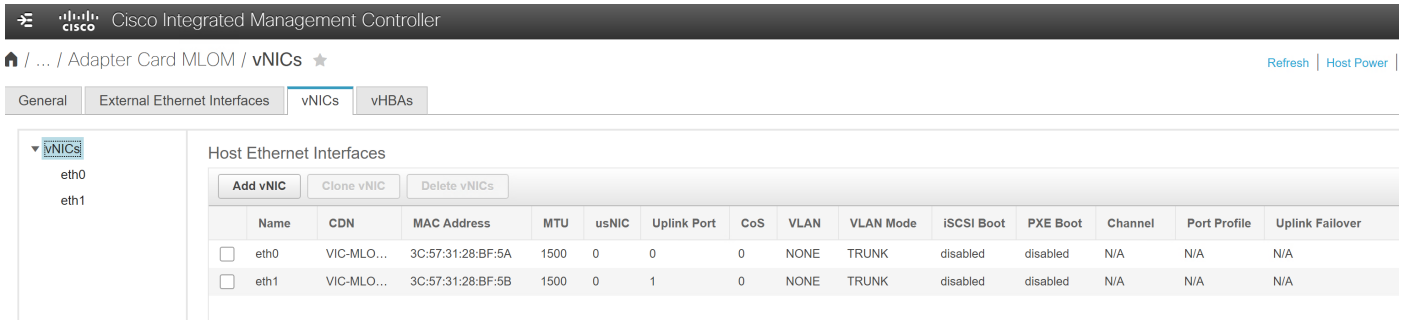
10. In the left menu, click Networking > Adapter Card MLOM.



11. Click the vNICs tab.

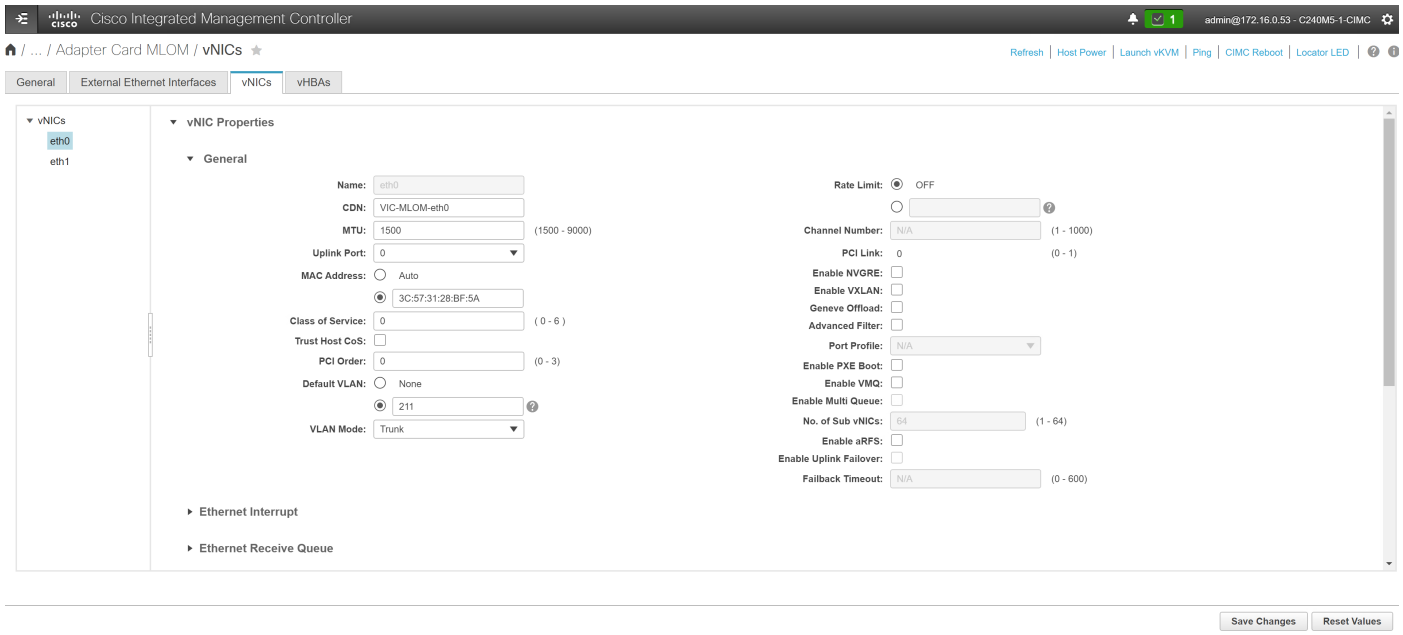


12. The factory default configuration comes with two virtual network interface cards (vNICs) defined: one assigned to port 0, and one assigned to port 1. Both vNICs are configured to allow any kind of traffic, with or without a VLAN tag. VLAN IDs must be managed at the operating system level.

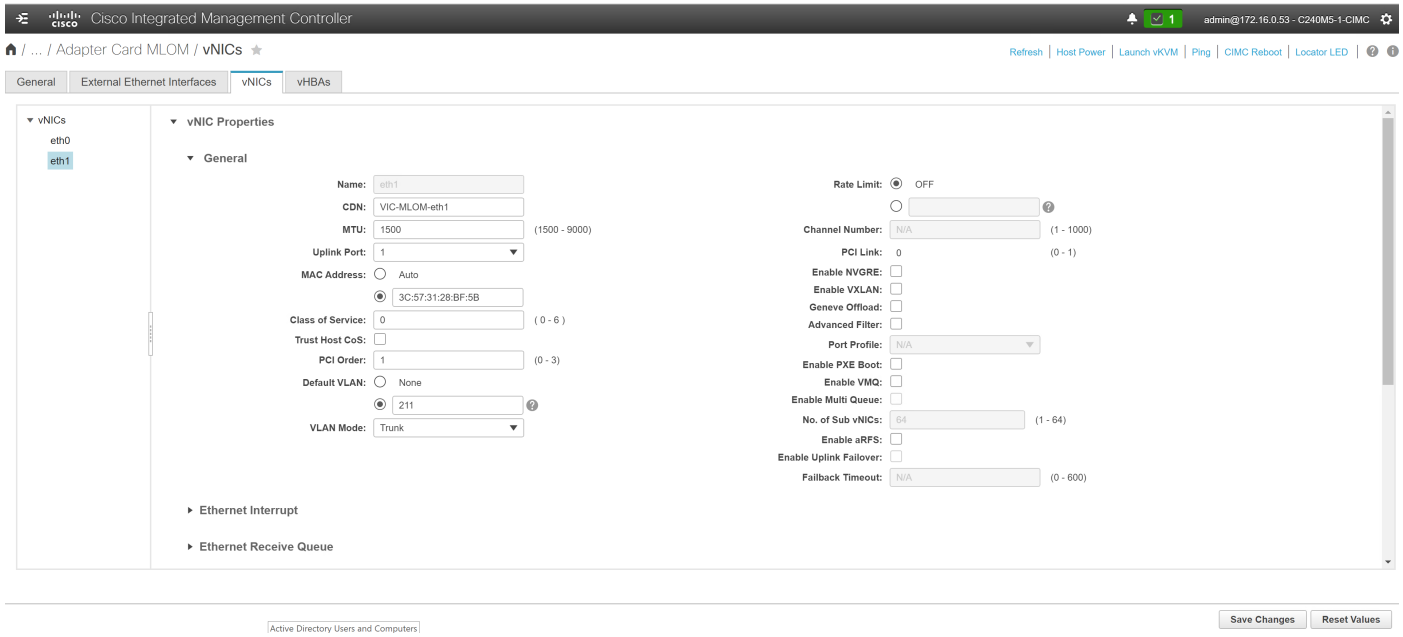


A useful feature of the Cisco virtual interface card (VIC) is the capability to define multiple virtual network adapters to be presented to the operating system, with each configured for specific uses. For example, you can configure administration traffic with a maximum transmission unit (MTU) of 1500 to be compatible with all communication partners, and you can configure the network for storage traffic with MTU 9000 for the best throughput. This sample configuration uses this approach, creating two vNICs for administration traffic, two vNICs for default user traffic, and two vNICs for data traffic to the storage location. For high availability, the two network devices per traffic type will be combined in a bond on the operating system layer.

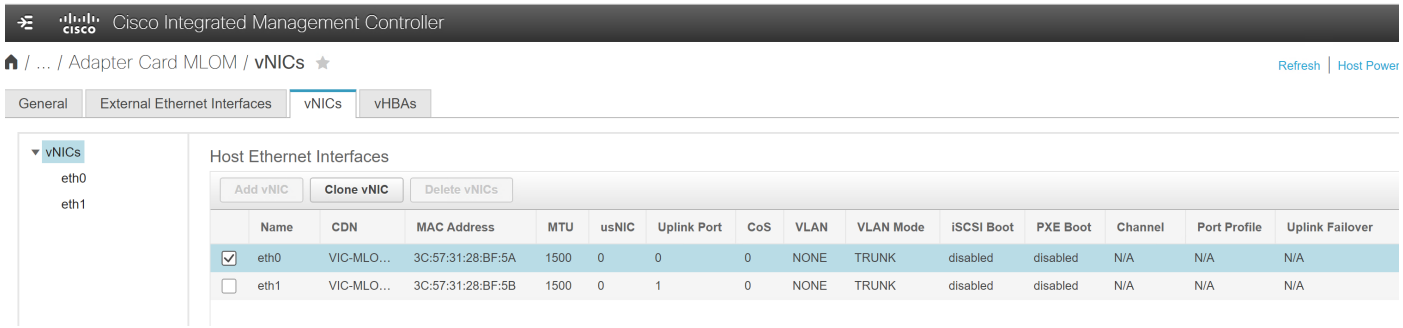
13. Click the first vNIC in the list on the left side and change the default VLAN to 211. Click Save Changes.



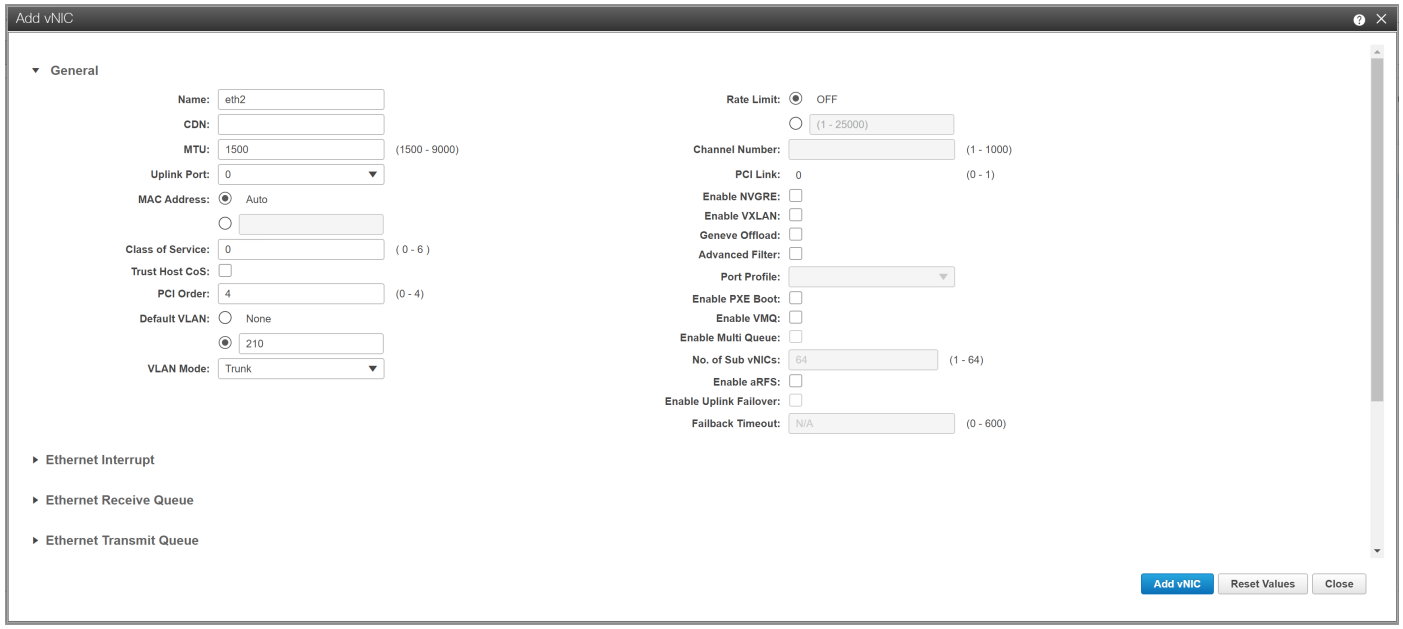
14. Click the second vNIC in the list and change the default VLAN to 211 as well. Click Save Changes.



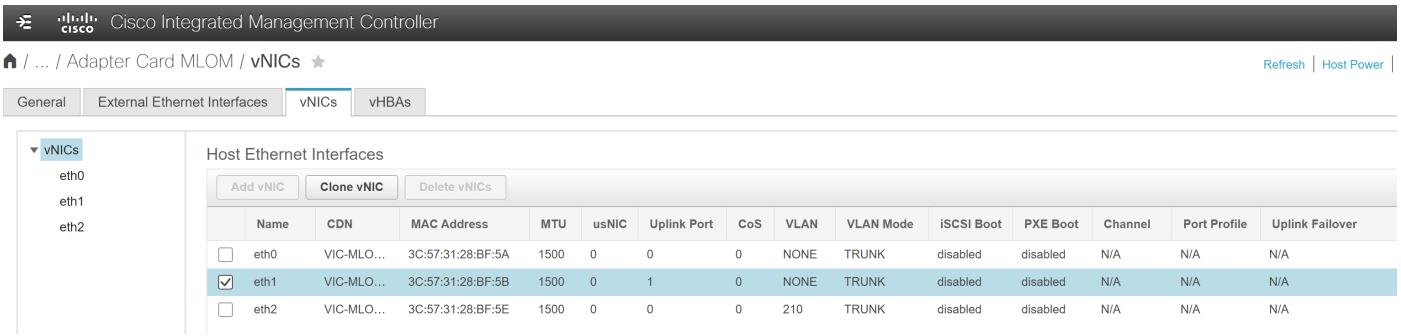
15. Click vNICs in the left menu and select the first vNIC in the table in the right pane. Click Clone vNIC.



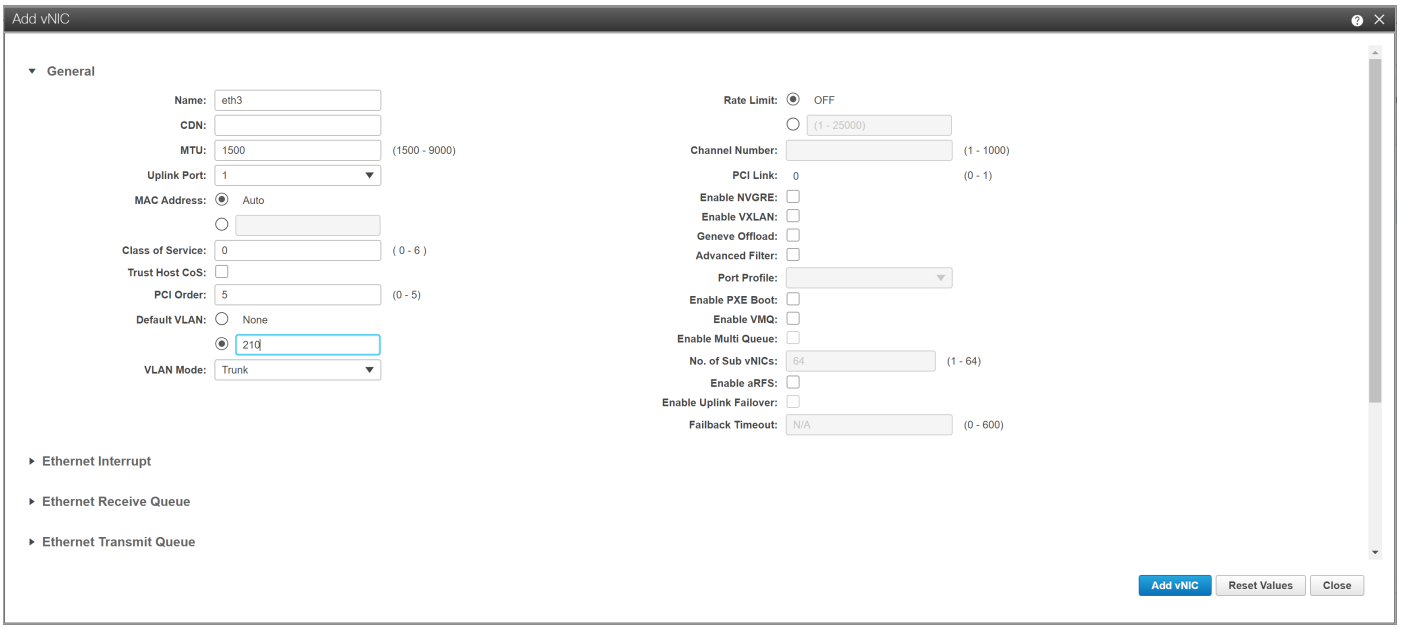
16. In the pop-up window, enter a name. Here, we used **eth2**. Change the default VLAN to 210. Click Add vNIC. Then confirm the creation of the vNIC by clicking OK in the next pop-up window.



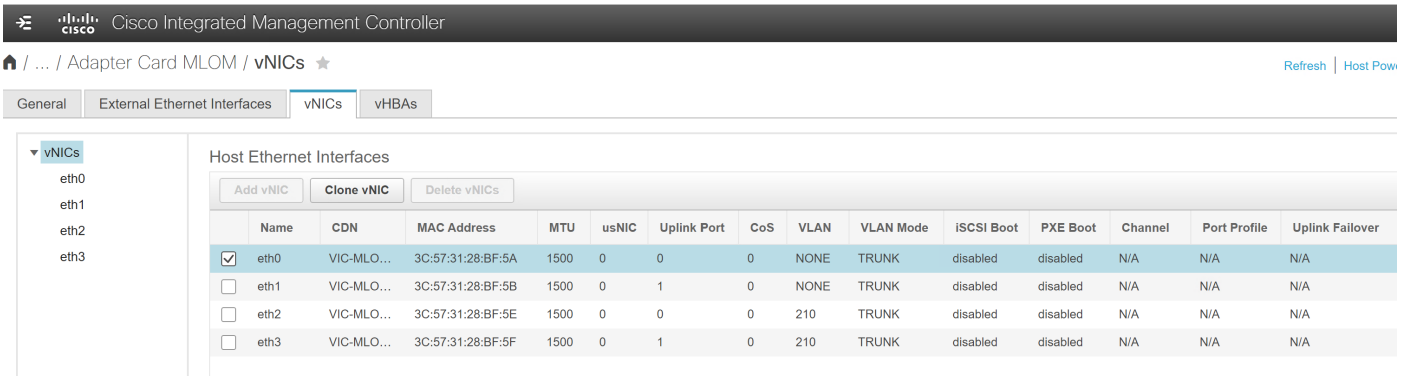
17. Select the second vNIC in the table and click Clone vNIC.



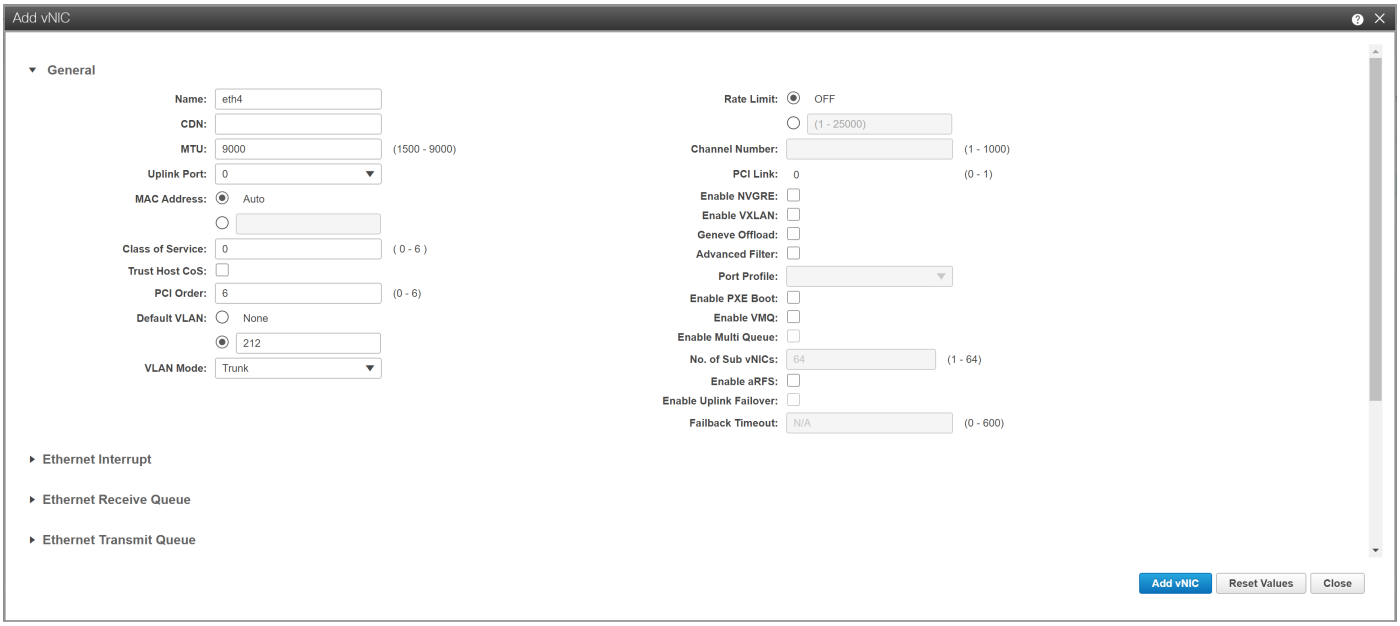
18. In the pop-up window, enter a name. Here, we used **eth3**. Change the default VLAN to 210. Click Add vNIC. Then confirm the creation of the vNIC by clicking OK in the next pop-up window.



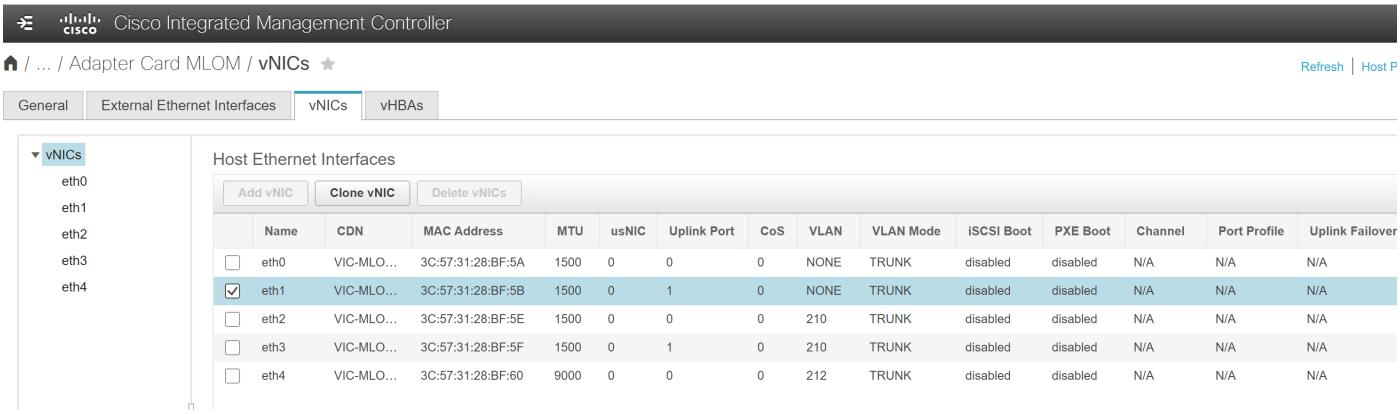
19. Select the first vNIC in the table in the right pane. Click Clone vNIC.



20. In the pop-up window, enter a name. Here, we used **eth4**. Change the MTU to 9000 and change the default VLAN to 212. Click Add vNIC. Then confirm the creation of the vNIC by clicking OK in the next pop-up window.

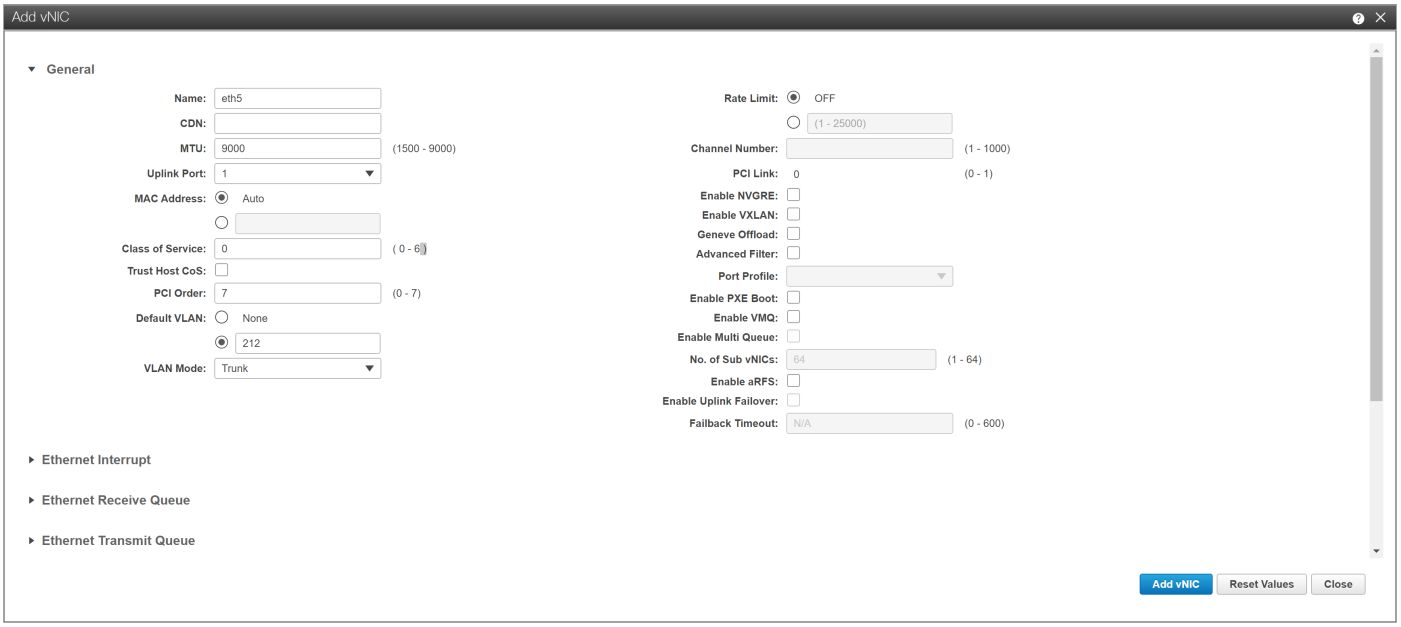


21. Select the second vNIC in the table and click Clone vNIC.



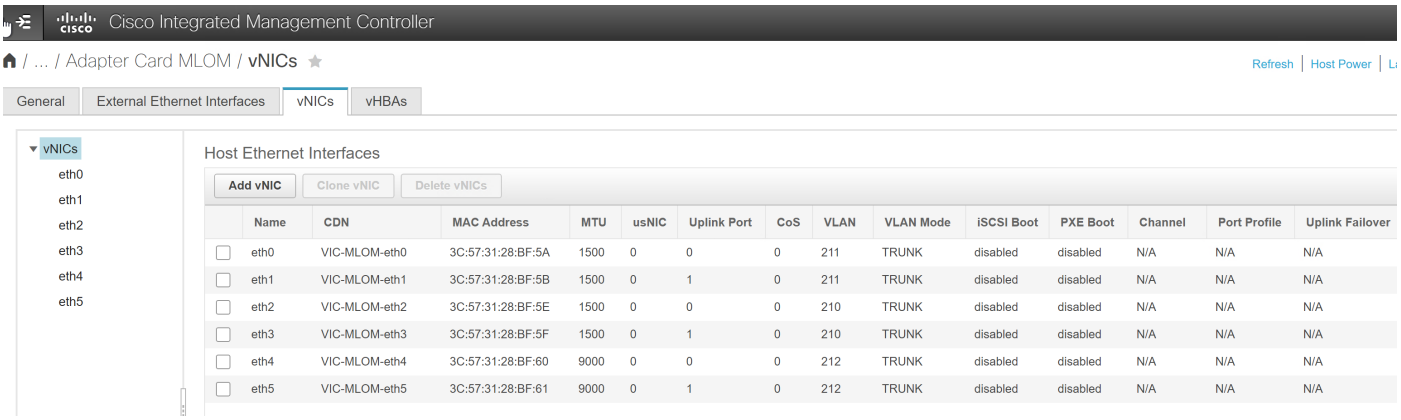
Name	CDN	MAC Address	MTU	usNIC	Uplink Port	CoS	VLAN	VLAN Mode	iSCSI Boot	PXE Boot	Channel	Port Profile	Uplink Failover
eth0	VIC-MLO…	3C:57:31:28:BF:5A	1500	0	0	0	NONE	TRUNK	disabled	disabled	N/A	N/A	N/A
eth1	VIC-MLO…	3C:57:31:28:BF:5B	1500	0	1	0	NONE	TRUNK	disabled	disabled	N/A	N/A	N/A
eth2	VIC-MLO…	3C:57:31:28:BF:5E	1500	0	0	0	210	TRUNK	disabled	disabled	N/A	N/A	N/A
eth3	VIC-MLO…	3C:57:31:28:BF:5F	1500	0	1	0	210	TRUNK	disabled	disabled	N/A	N/A	N/A
eth4	VIC-MLO…	3C:57:31:28:BF:60	9000	0	0	0	212	TRUNK	disabled	disabled	N/A	N/A	N/A

22. In the pop-up window, enter a name. Here, we used **eth5**. Change the MTU to 9000 and change the default VLAN to 212. Click Add vNIC. Then confirm the creation of the vNIC by clicking OK in the next pop-up window.



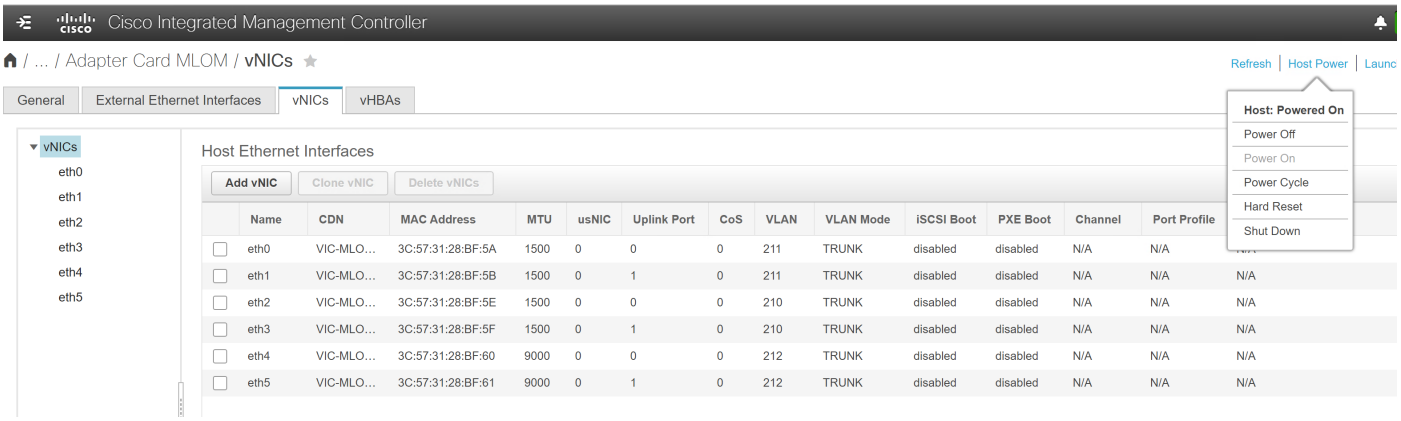
23. The vNICs tab shows the final list of vNICs. For every VLAN ID, there are two vNICs: one on uplink port 0, and one on uplink port 1.

    You will use this list later to validate the bond configuration at the operating system layer.

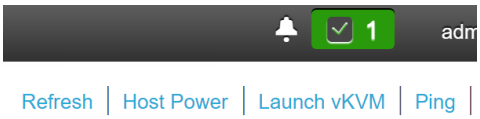


Name	CDN	MAC Address	MTU	usNIC	Uplink Port	CoS	VLAN	VLAN Mode	iSCSI Boot	PXE Boot	Channel	Port Profile	Uplink Failover
eth0	VIC-MLOM-eth0	3C:57:31:28:BF:5A	1500	0	0	0	211	TRUNK	disabled	disabled	N/A	N/A	N/A
eth1	VIC-MLOM-eth1	3C:57:31:28:BF:5B	1500	0	1	0	211	TRUNK	disabled	disabled	N/A	N/A	N/A
eth2	VIC-MLOM-eth2	3C:57:31:28:BF:5E	1500	0	0	0	210	TRUNK	disabled	disabled	N/A	N/A	N/A
eth3	VIC-MLOM-eth3	3C:57:31:28:BF:5F	1500	0	1	0	210	TRUNK	disabled	disabled	N/A	N/A	N/A
eth4	VIC-MLOM-eth4	3C:57:31:28:BF:60	9000	0	0	0	212	TRUNK	disabled	disabled	N/A	N/A	N/A
eth5	VIC-MLOM-eth5	3C:57:31:28:BF:61	9000	0	1	0	212	TRUNK	disabled	disabled	N/A	N/A	N/A

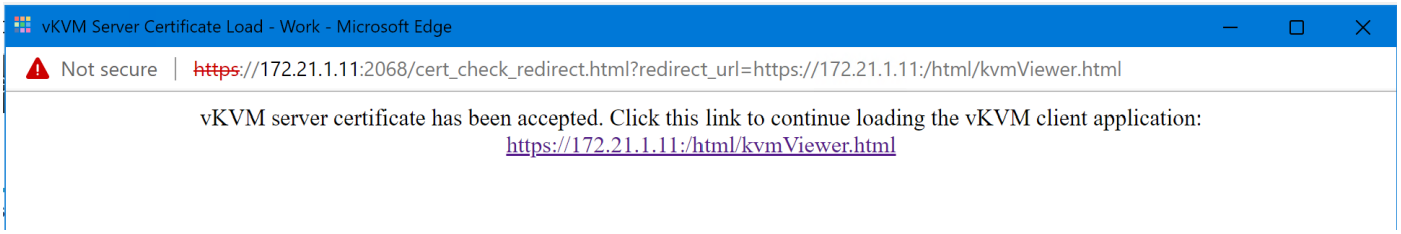
24. The new settings become active with the next power cycle of the server. At the top right side of the window, click Host Power > Power Off. In the pop-up window, click OK.



25. Click Launch vKVM.



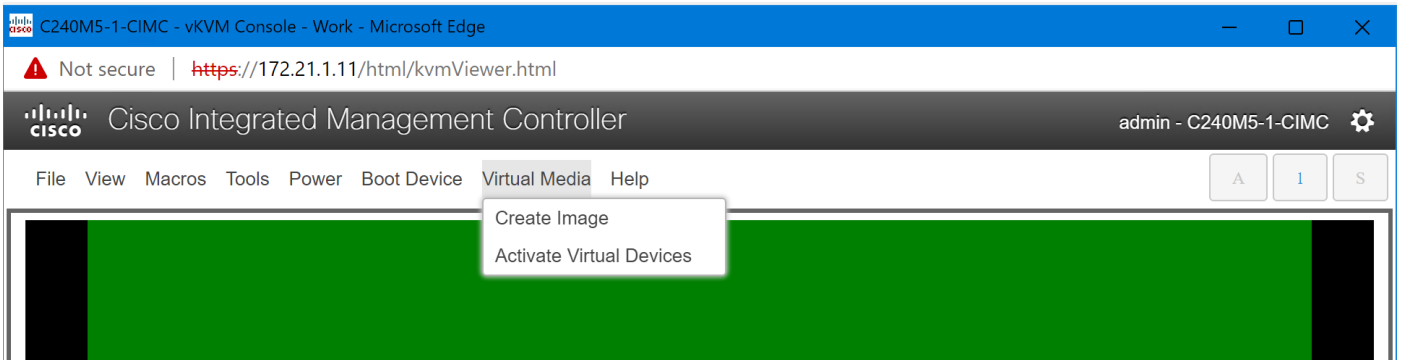
26. In the new window, take the necessary steps to continue with an untrusted certificate.



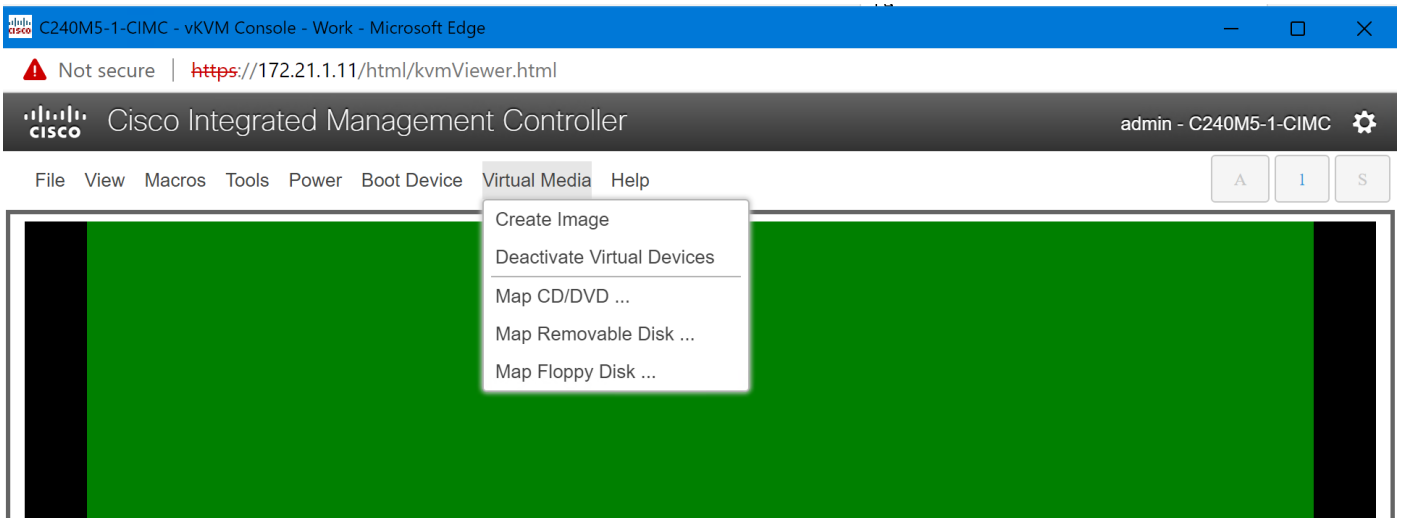
Install SLE Micro

Follow the steps here to install the SLE Micro operating system on the prepared server.

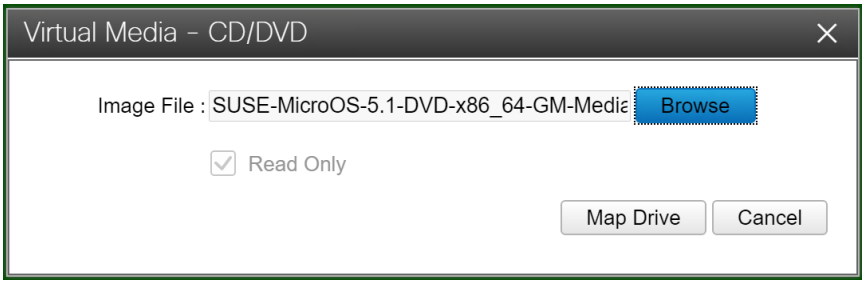
1. In the virtual keyboard, video, and mouse (vKVM) window, click Virtual Media > Activate Virtual Devices.



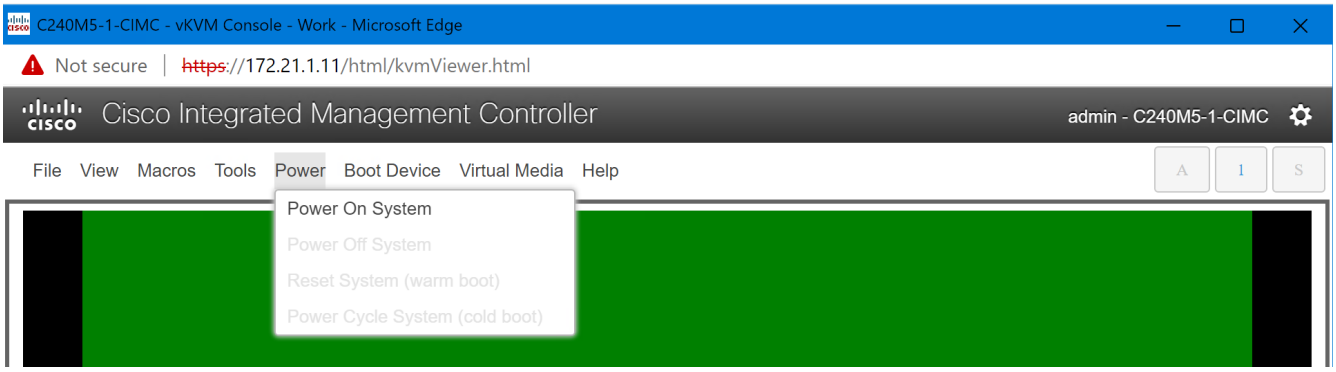
2. Again click Virtual Media and now click Map CD/DVD.



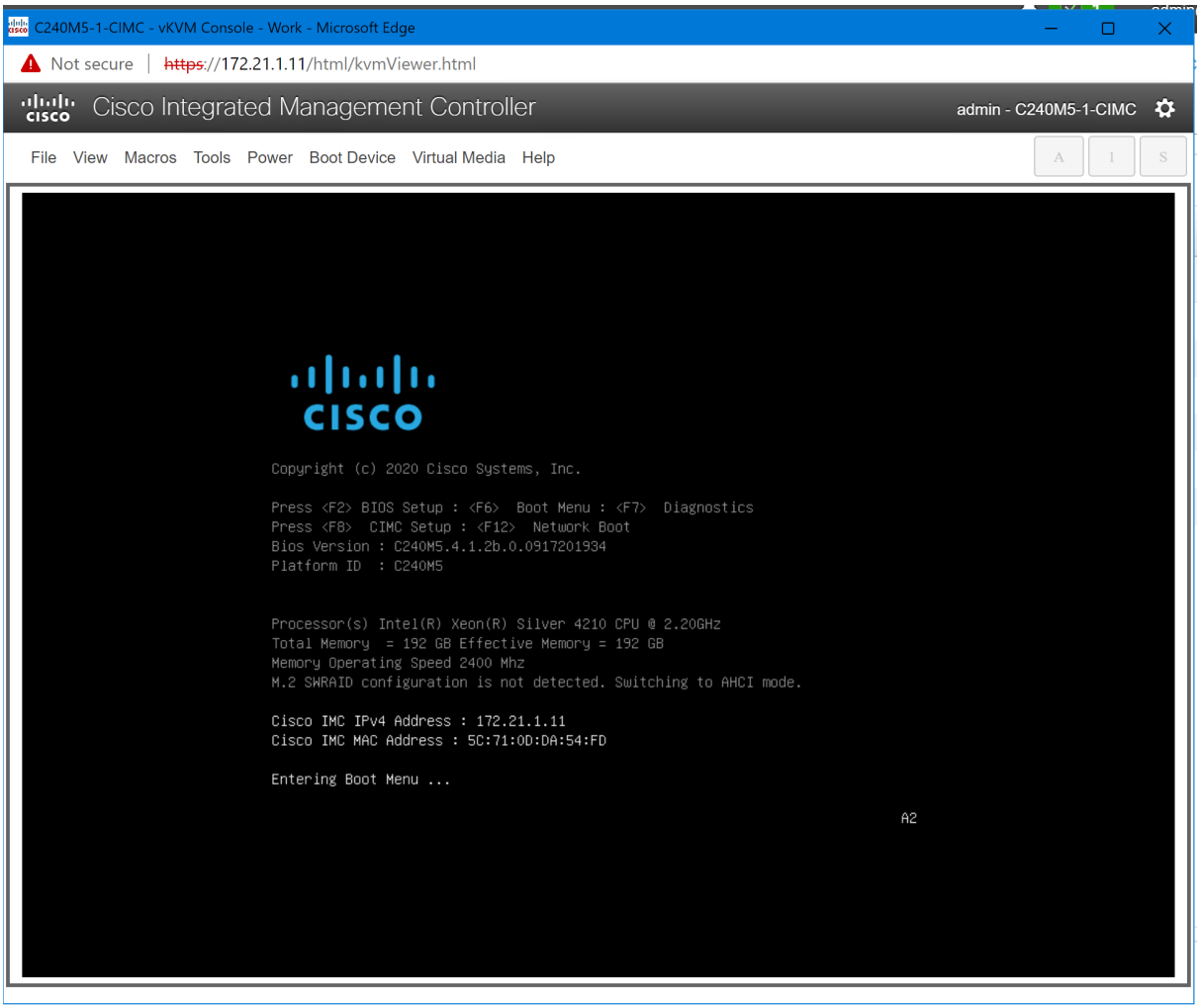
3. Click Browse, select the SLE Micro Media ISO image, and click Map Drive.



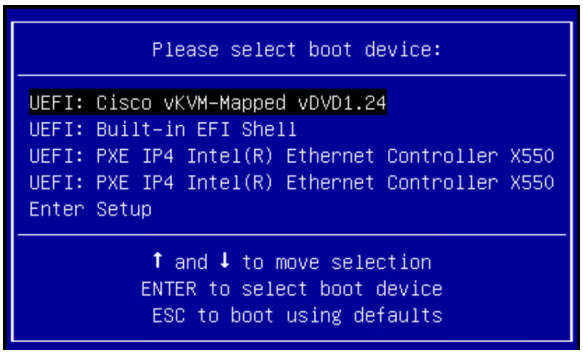
4. Click Power and choose Power On System. In the pop-up window, click OK.



5. As soon the selection menu appears, press F6 to enter the Boot Menu.

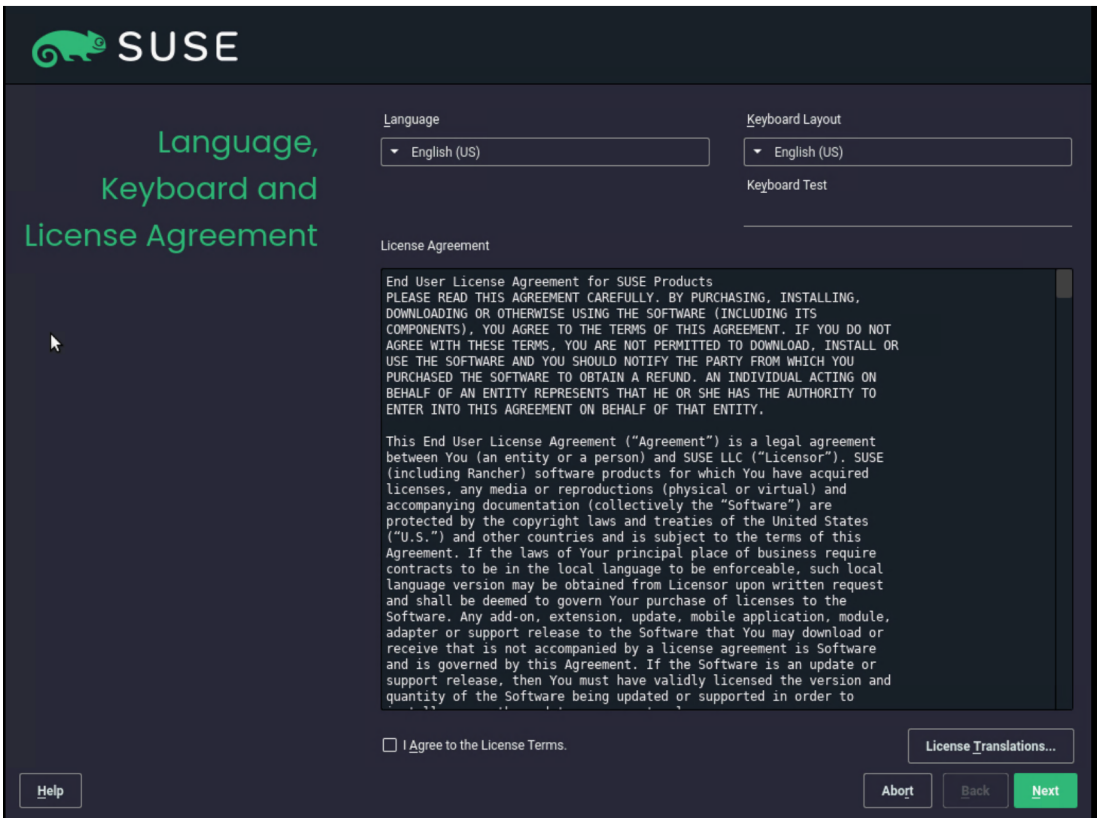


6. Select UEFI: Cisco vKVM-Mapped vDVD and press Enter.

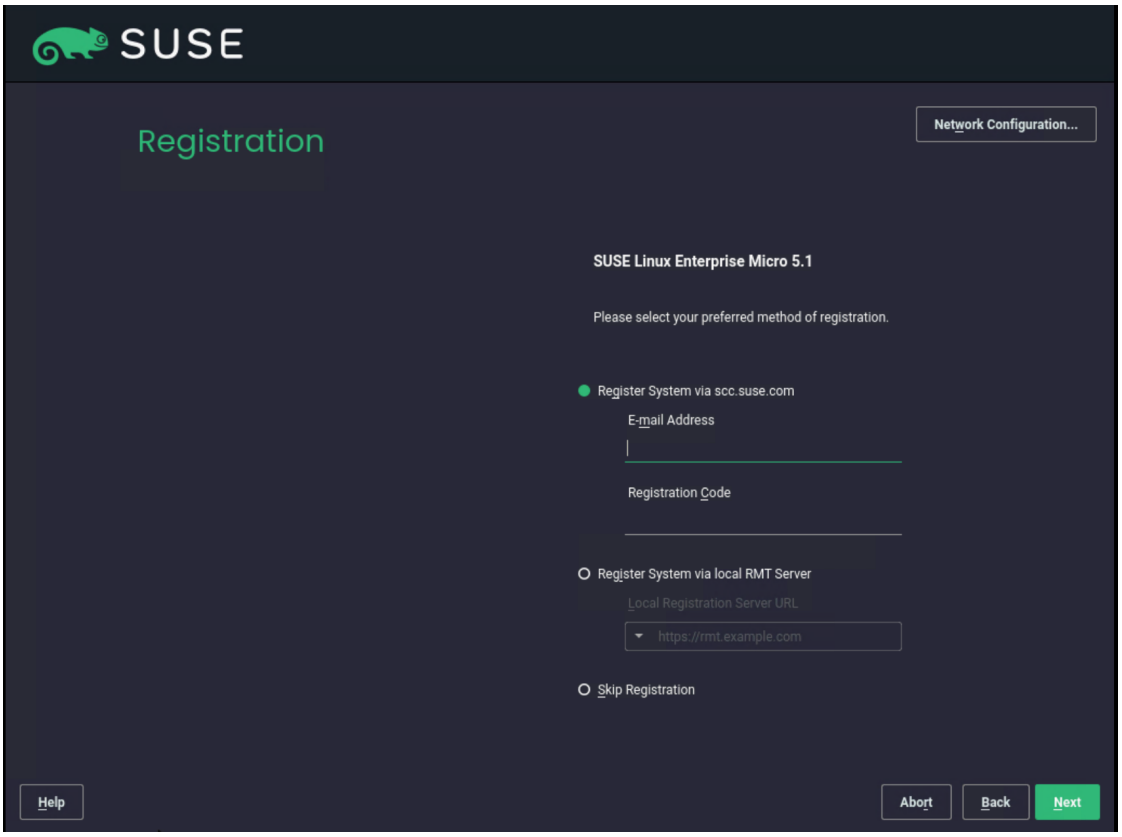


The SUSE installation process will start automatically.

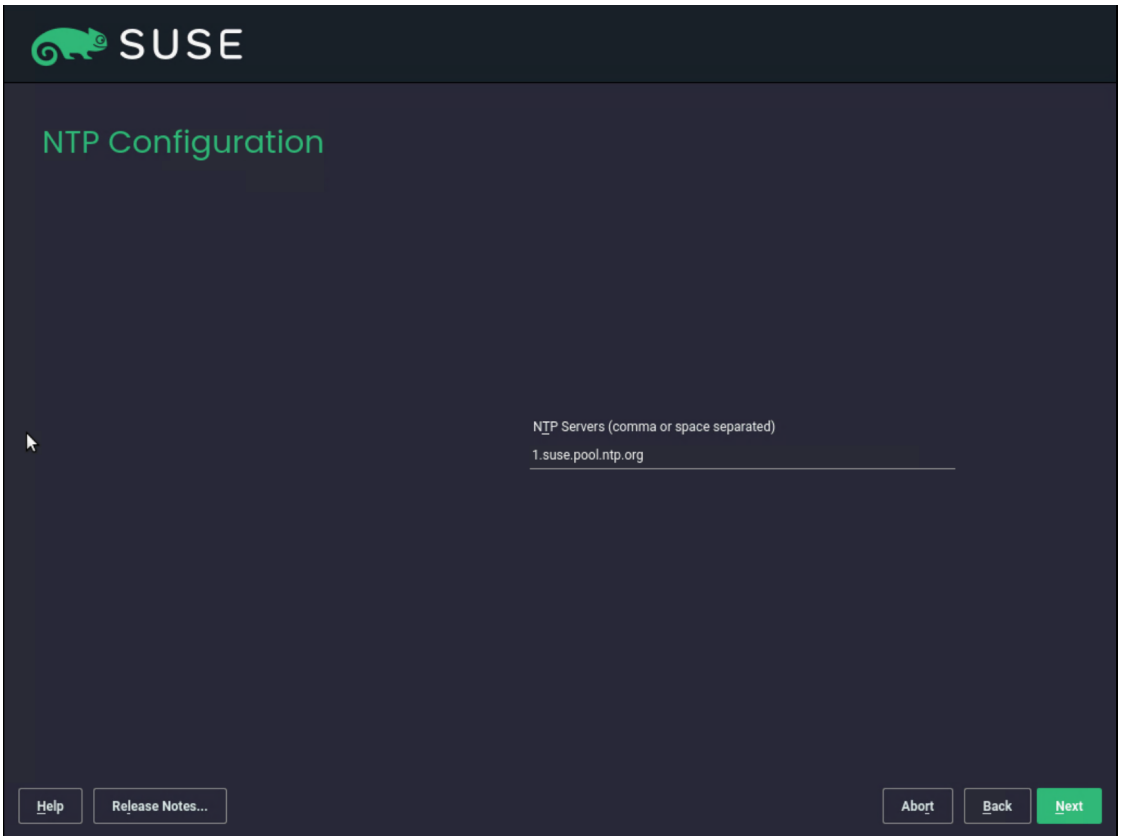
7. Select your language and keyboard layout, agree to the license terms, and click Next.



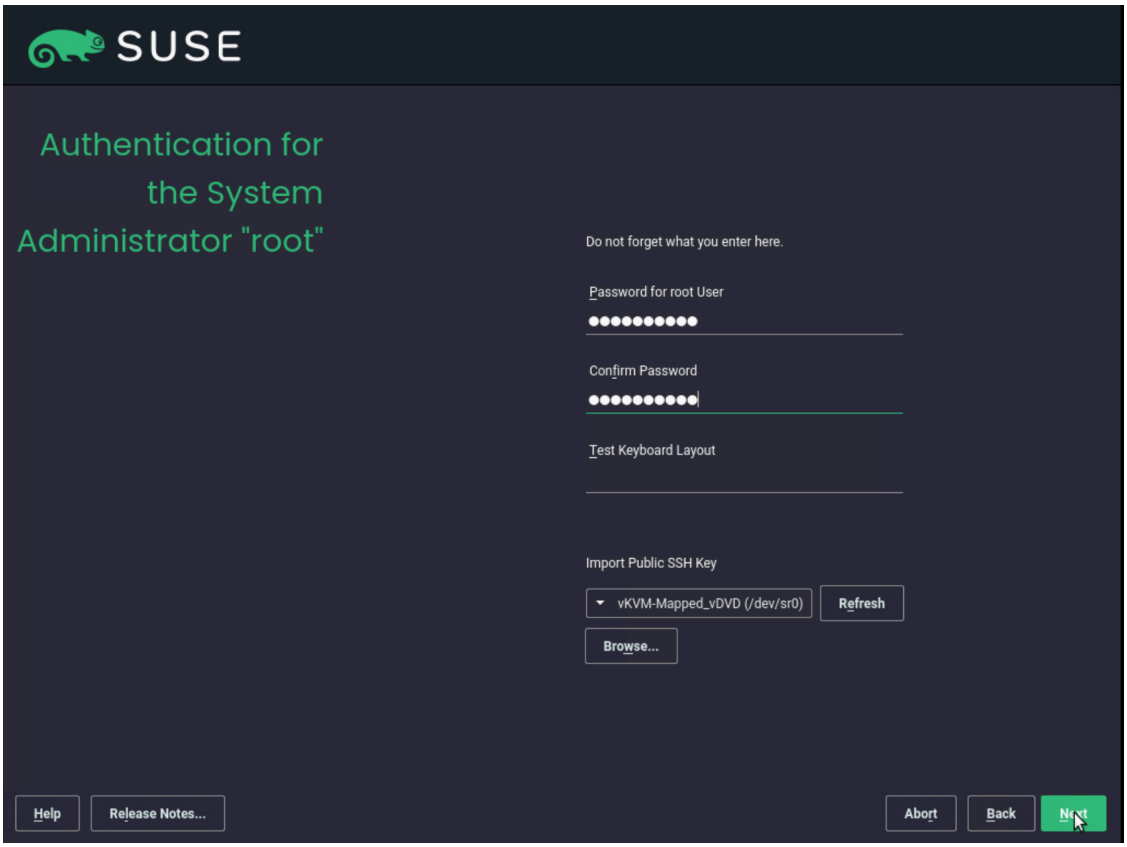
8. Enter your information to register this installation or select Skip Registration and click Next.



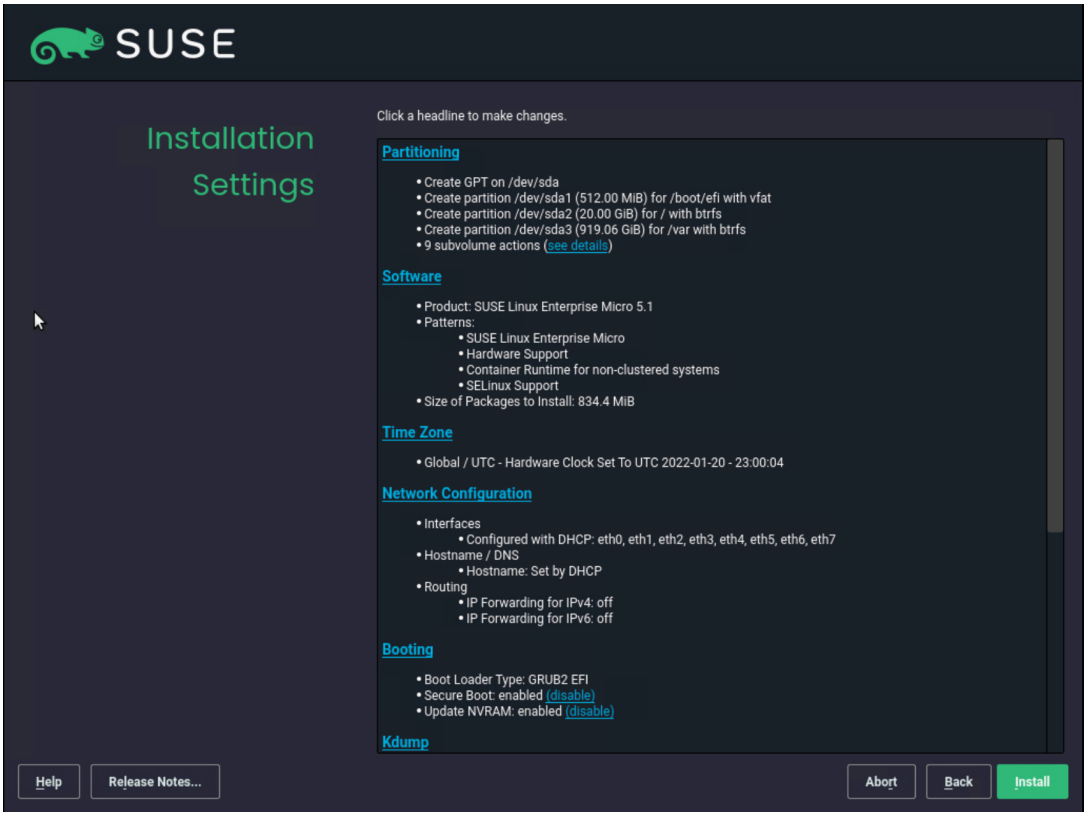
9. Enter the list of Network Time Protocol (NTP) servers and click Next.



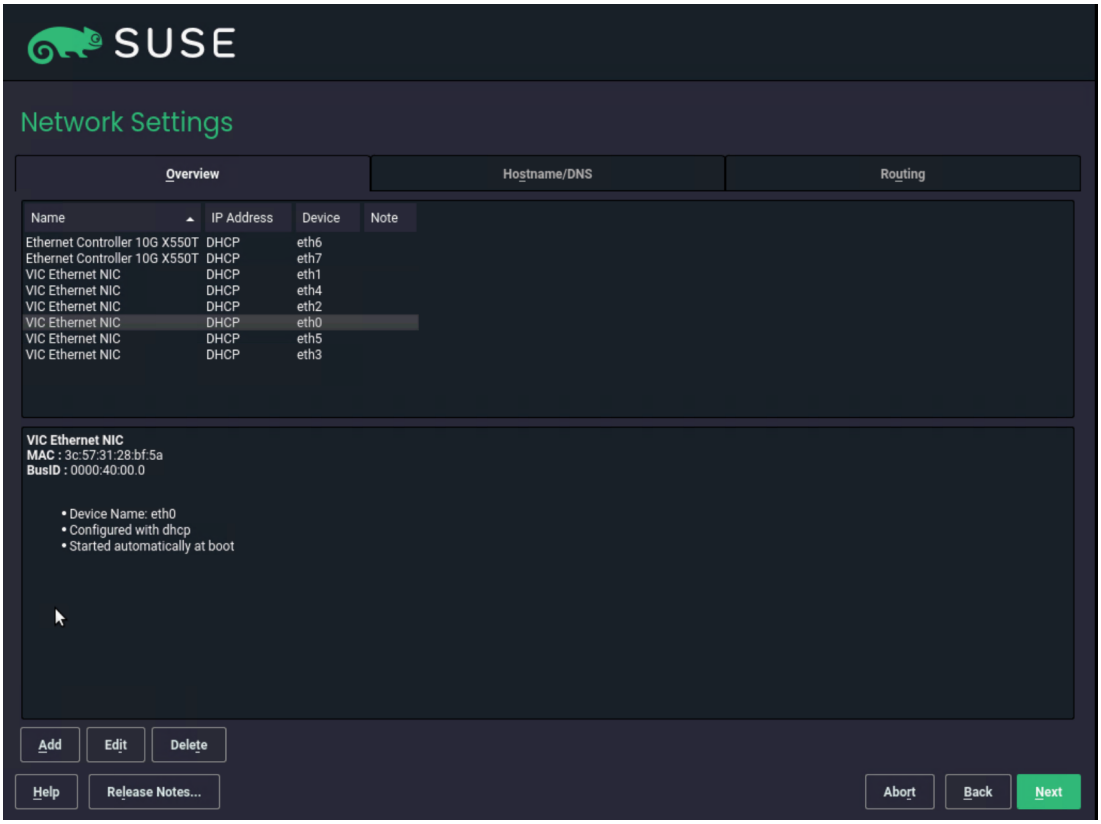
10. Enter and confirm the password for the user root. Then click Next.



11. The partitioning suggested by SUSE is already optimized for container deployments, and no changes are required. Configure the time zone as required for this setup. Click Network Configuration.

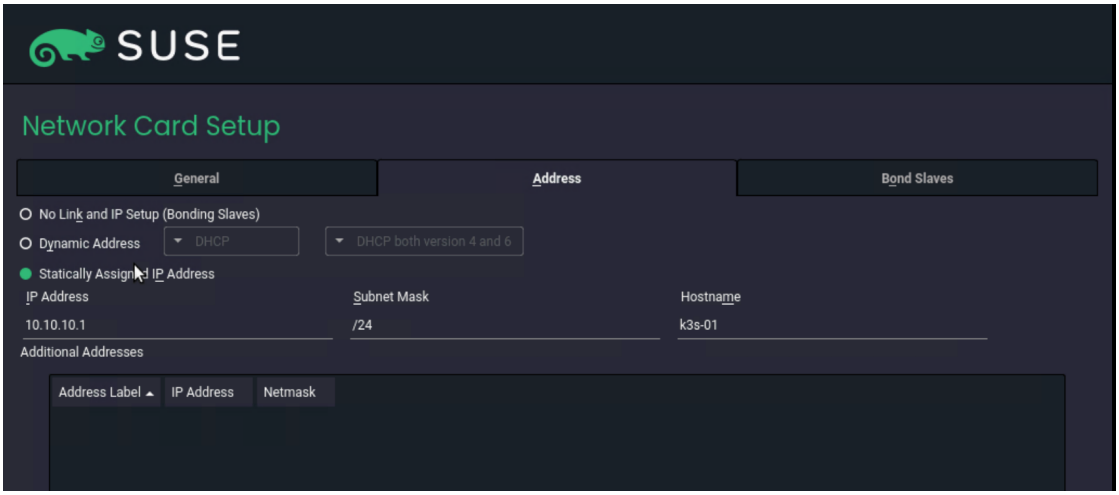


12. Click the various devices in the network view and compare the names and MAC addresses with the vNIC list from the IMC. Click Add.

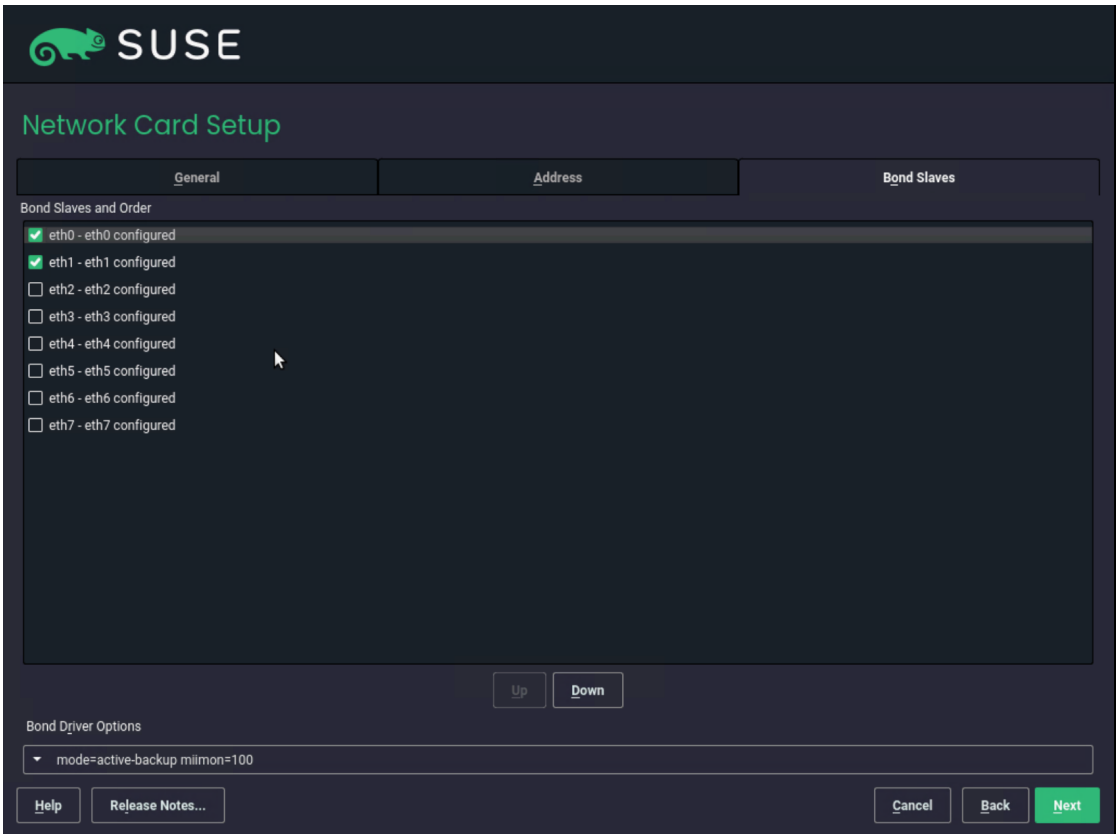


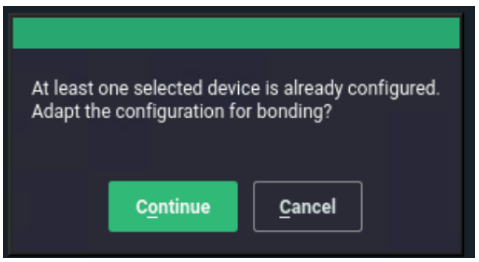
13. We want to create bonding devices for high availability. Select Bonding and click Next.

14. Enter the IP address, the netmask for the administration traffic network, and a hostname. Click Bond Slaves.

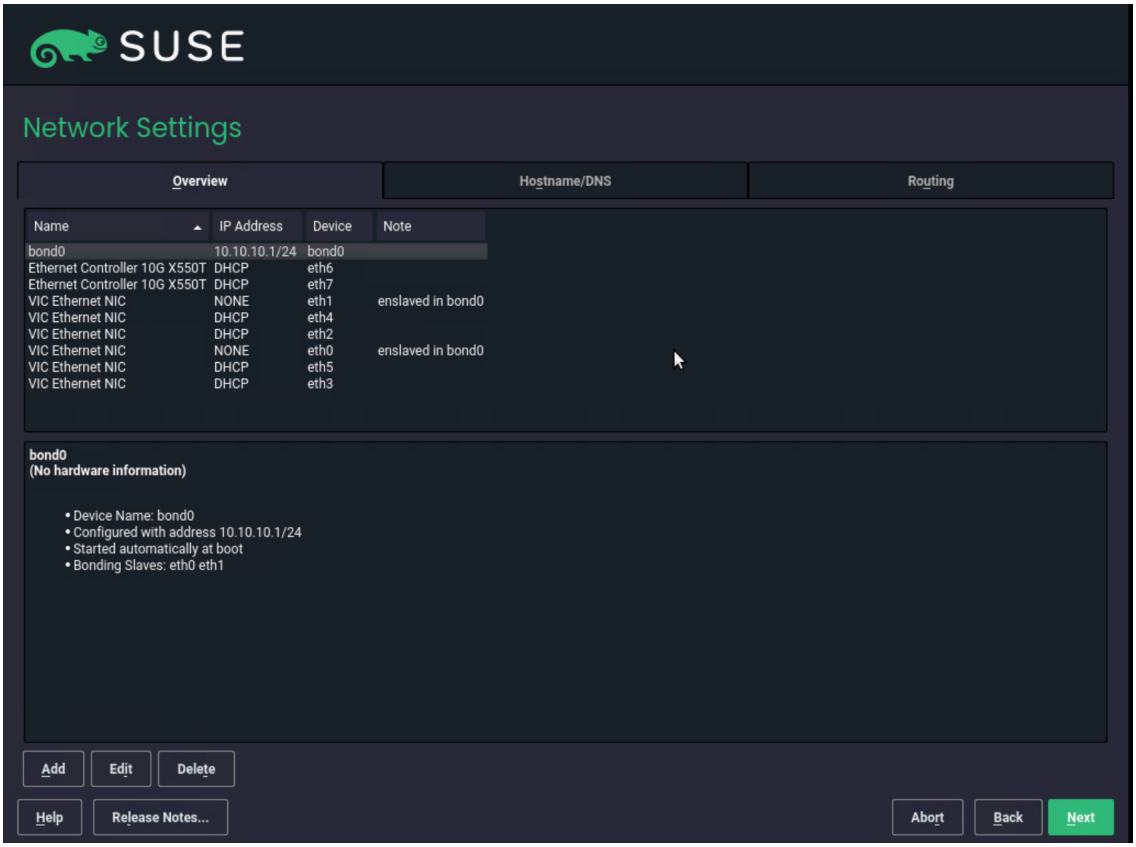


15. Select the two interfaces created for administration traffic (eth0 and eth1) and use active-backup as the mode. Click Next. In the pop-up window, click Continue.

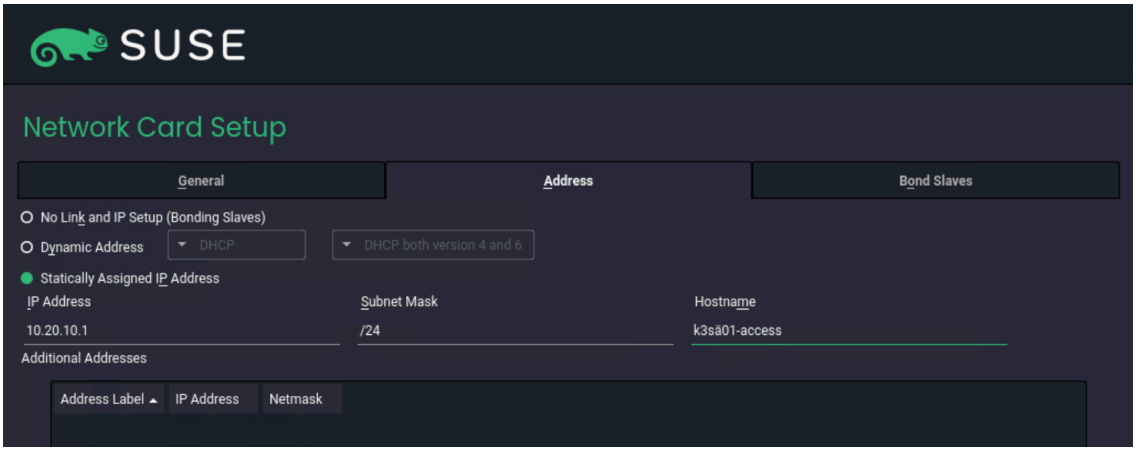




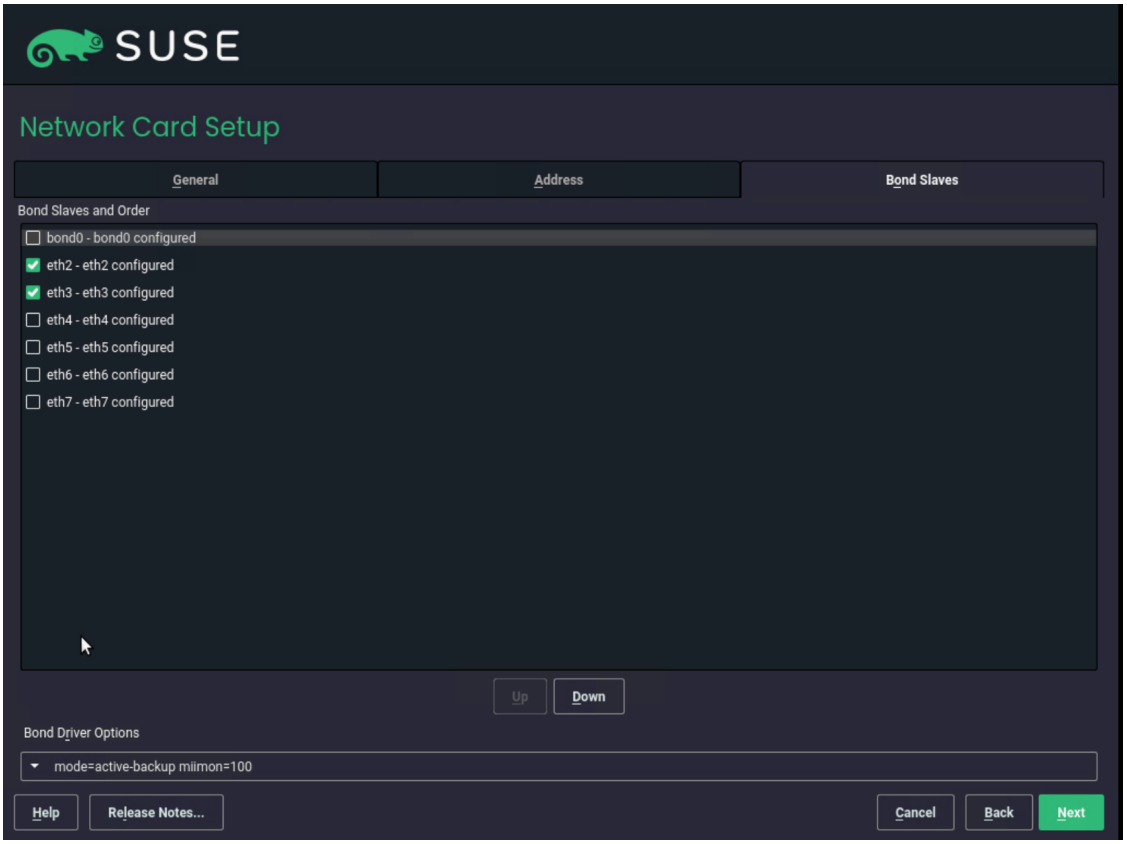
16. Click Add to create the bonding device for the access traffic. Select Bonding on the next screen and click Next.



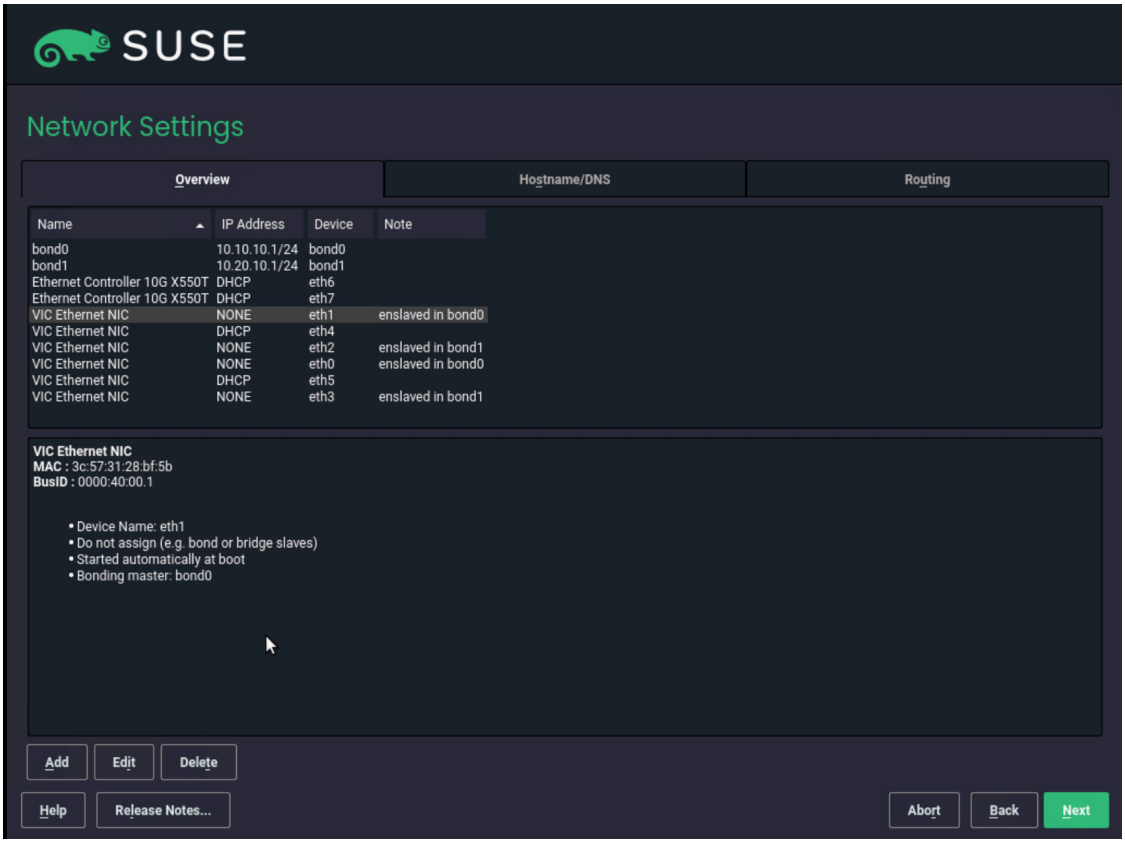
17. Enter the IP address and netmask for the access traffic connection and a hostname. Click Bond Slaves.



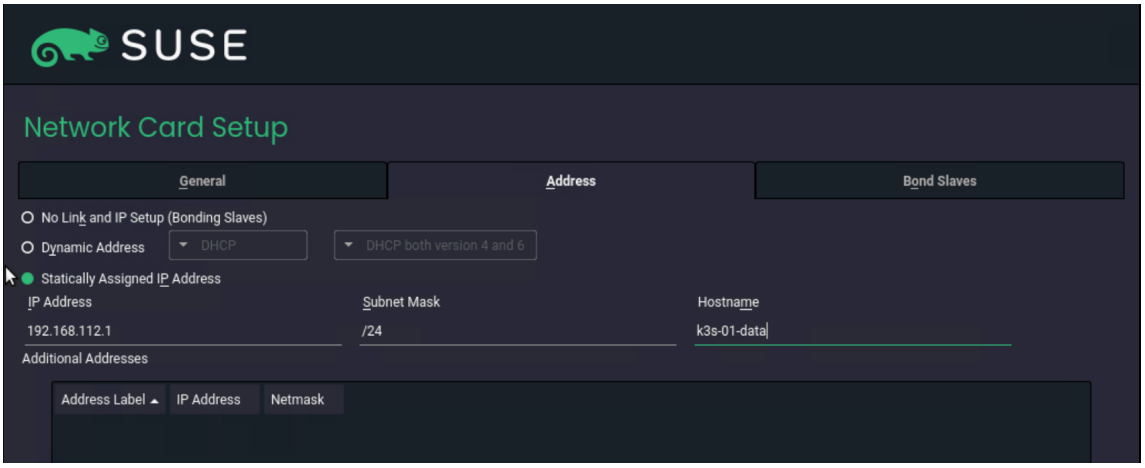
18. Select the two interfaces created for access traffic (eth2 and eth3) and use active-backup as the mode. Click Next. In the pop-up window, click Continue.



19. Back on the Network Settings screen, click Add to create the bonding for storage traffic. Select Bonding on the next screen and click Next.

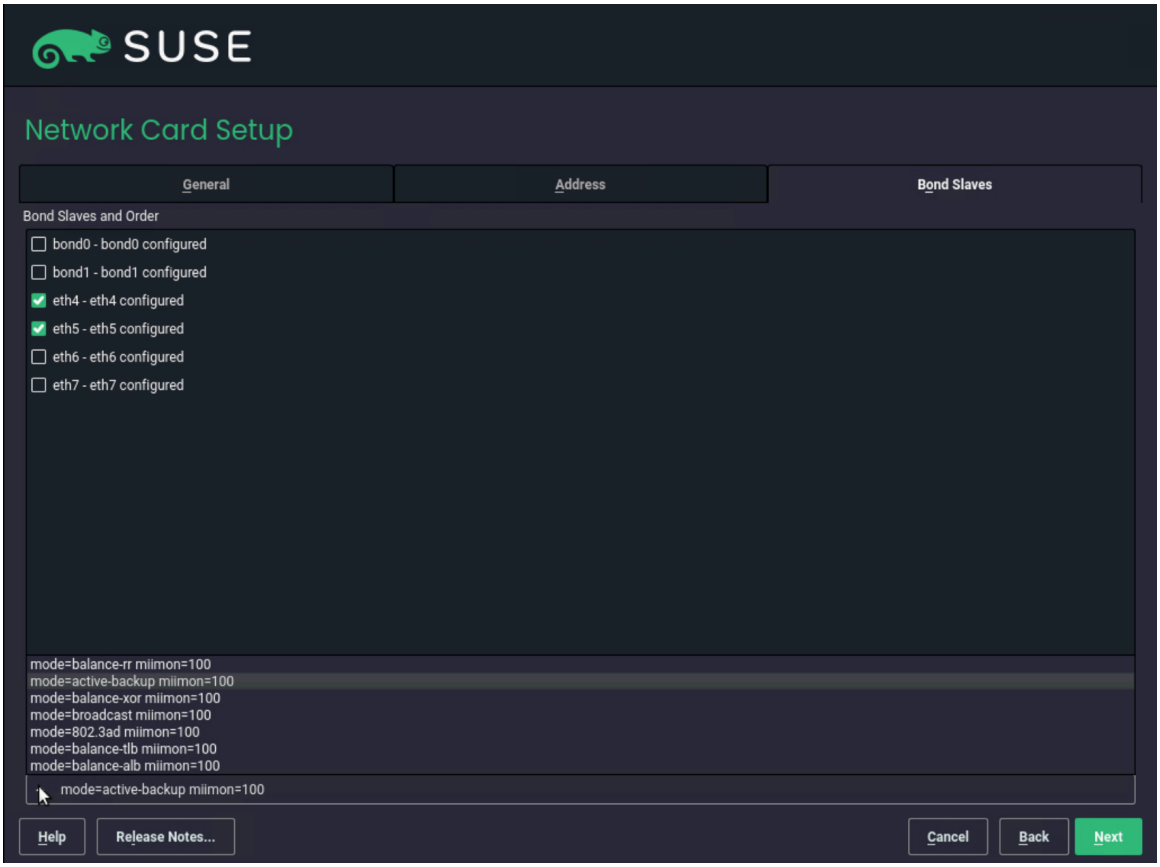


20. Enter the IP address and netmask for the storage traffic connection and a hostname. Click Bond Slaves.

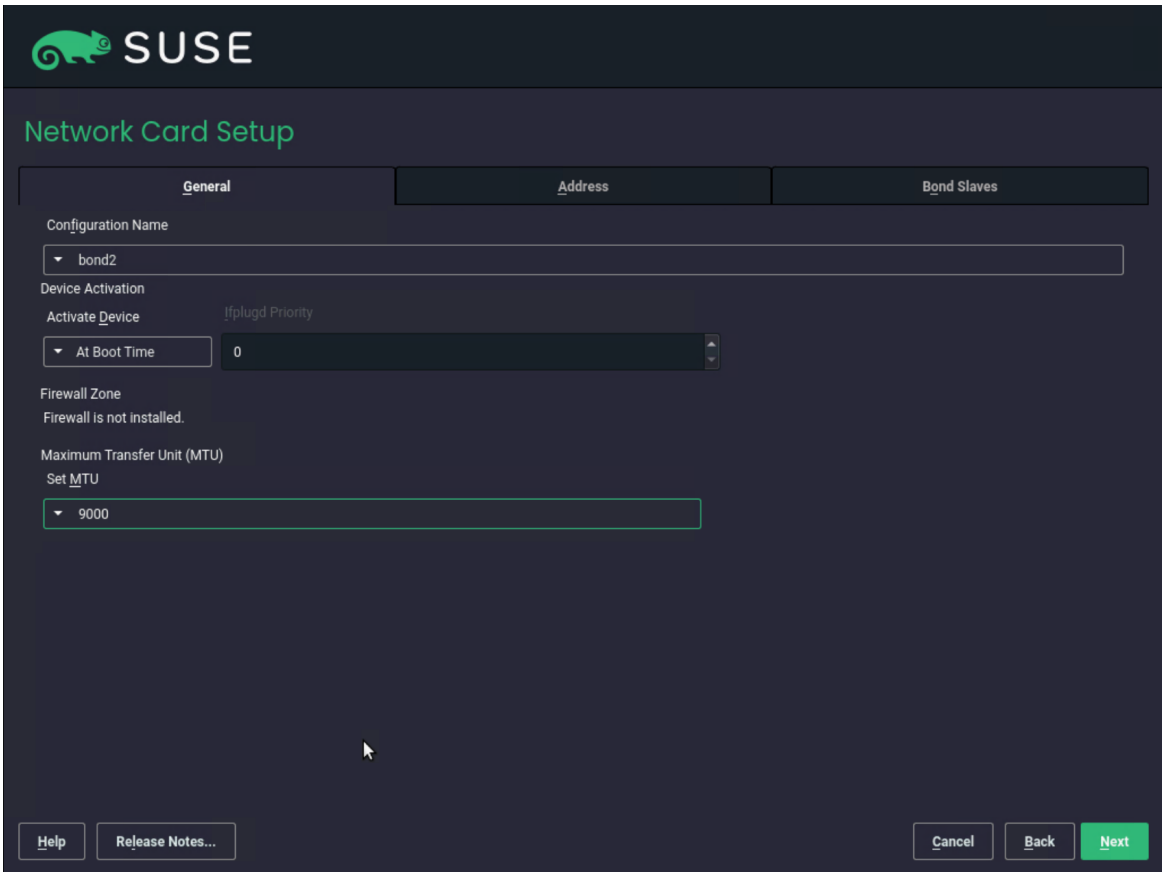


21. Select the two interfaces created for storage traffic (eth4 and eth5). Check with your networking and storage teams to determine whether an active-active bonding option for storage access is possible. An active-active option will increase the maximum throughput between this server and the storage system. In the absence of a clear answer from the network team, use active-backup as the mode.

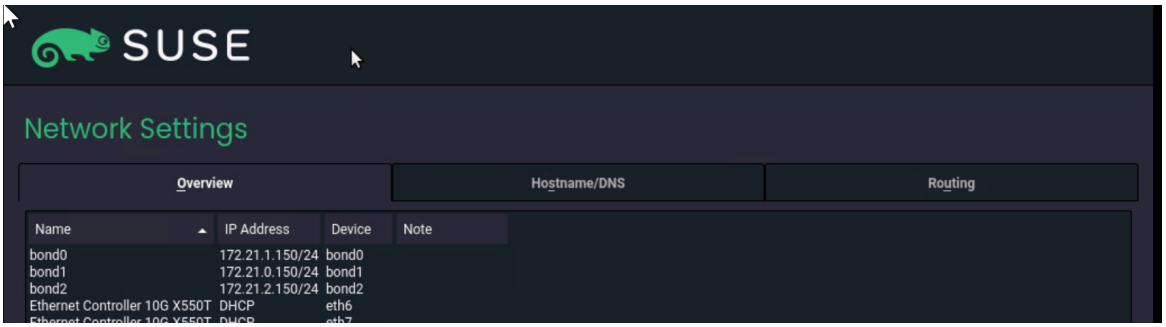
22. Click General and in the pop-up window click Continue.



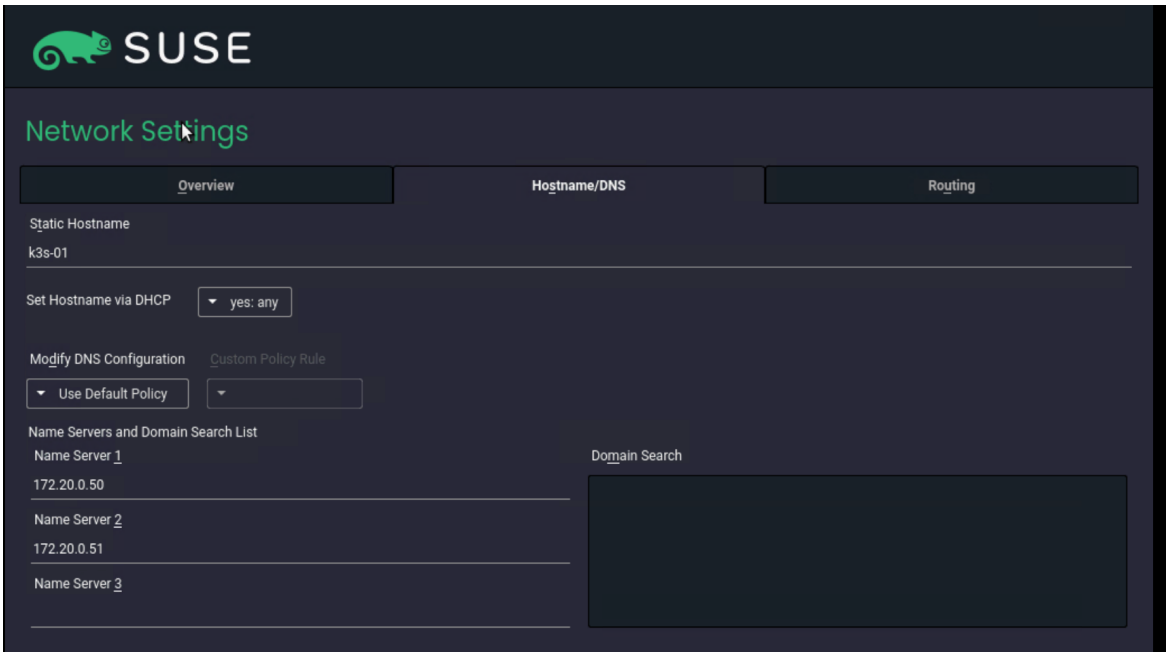
23. Enter **9000** in the field under Set MTU and click Next.



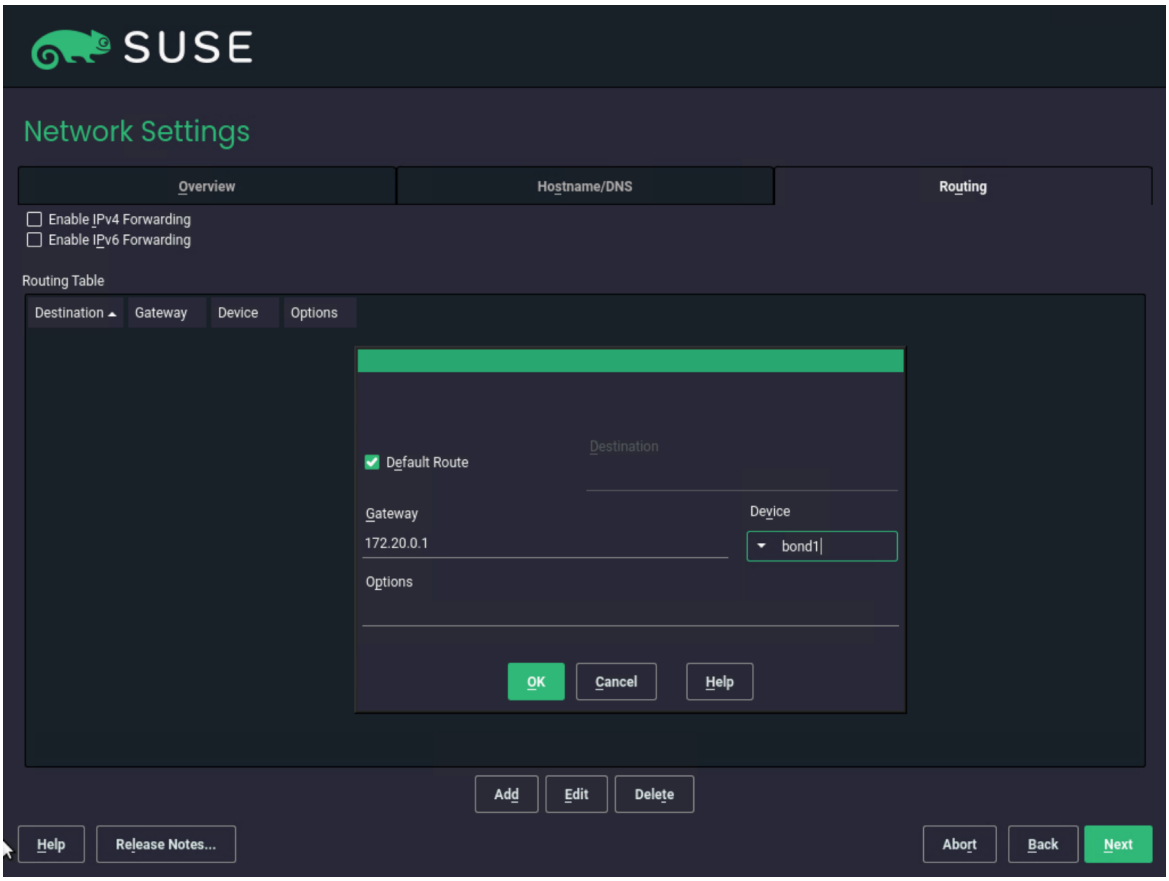
24. Click Hostname / DNS.



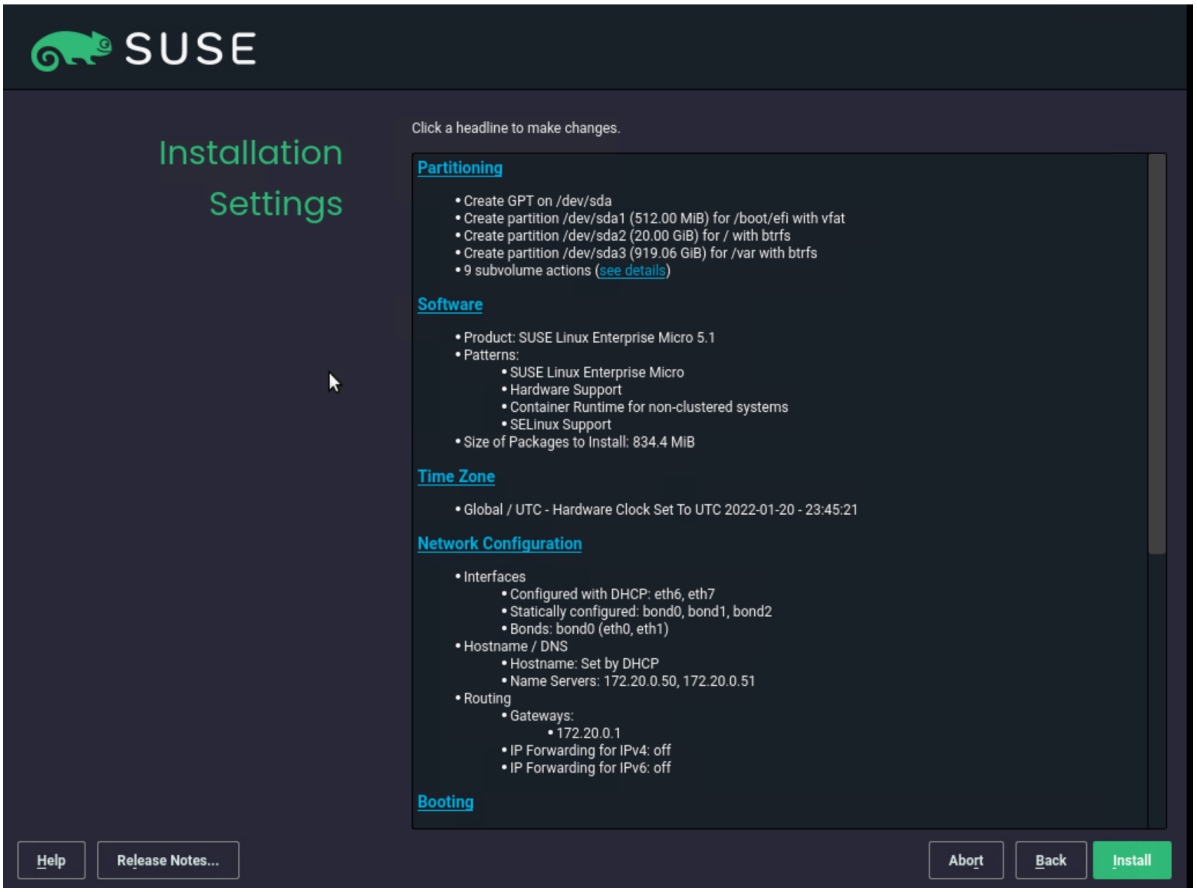
25. Enter the static hostname for this system and the IP address for at least one name server. Click Routing.



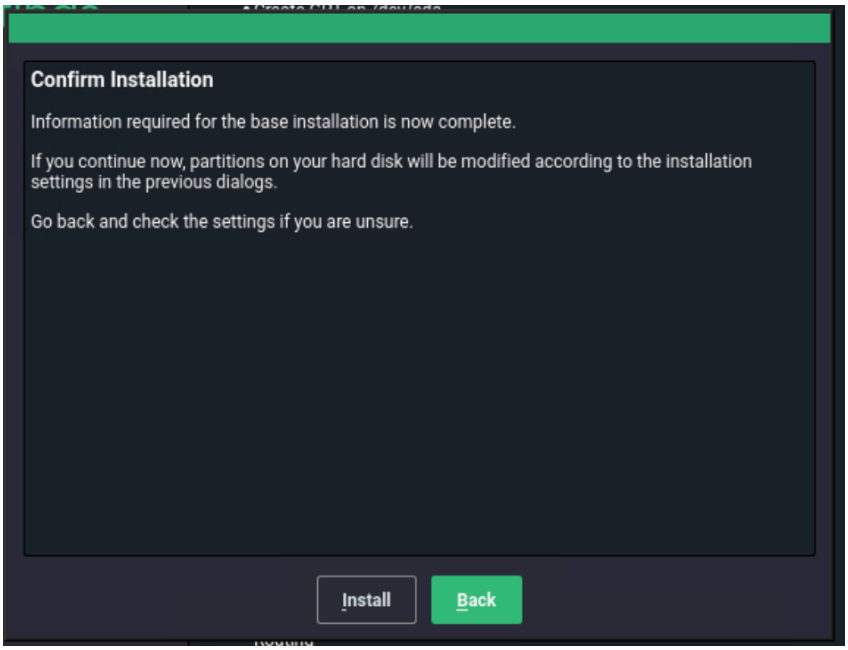
26. Click Add and in the pop-up window enter at least the default route for your network. Click OK. Click Next.



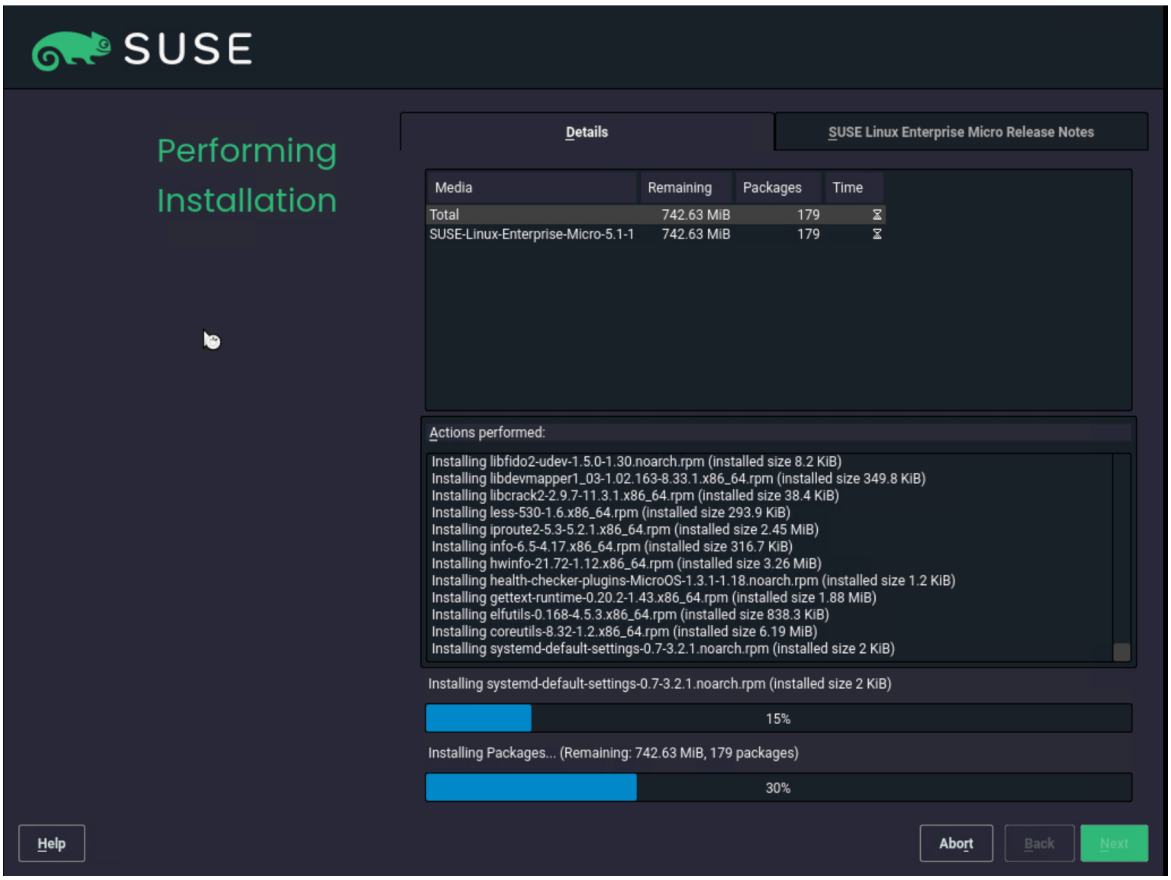
27. Check all the information and then click Install.



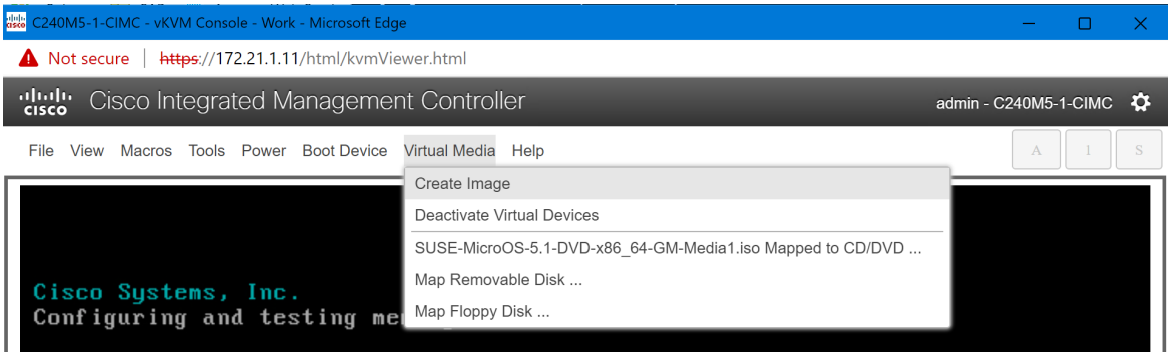
28. Click Install.



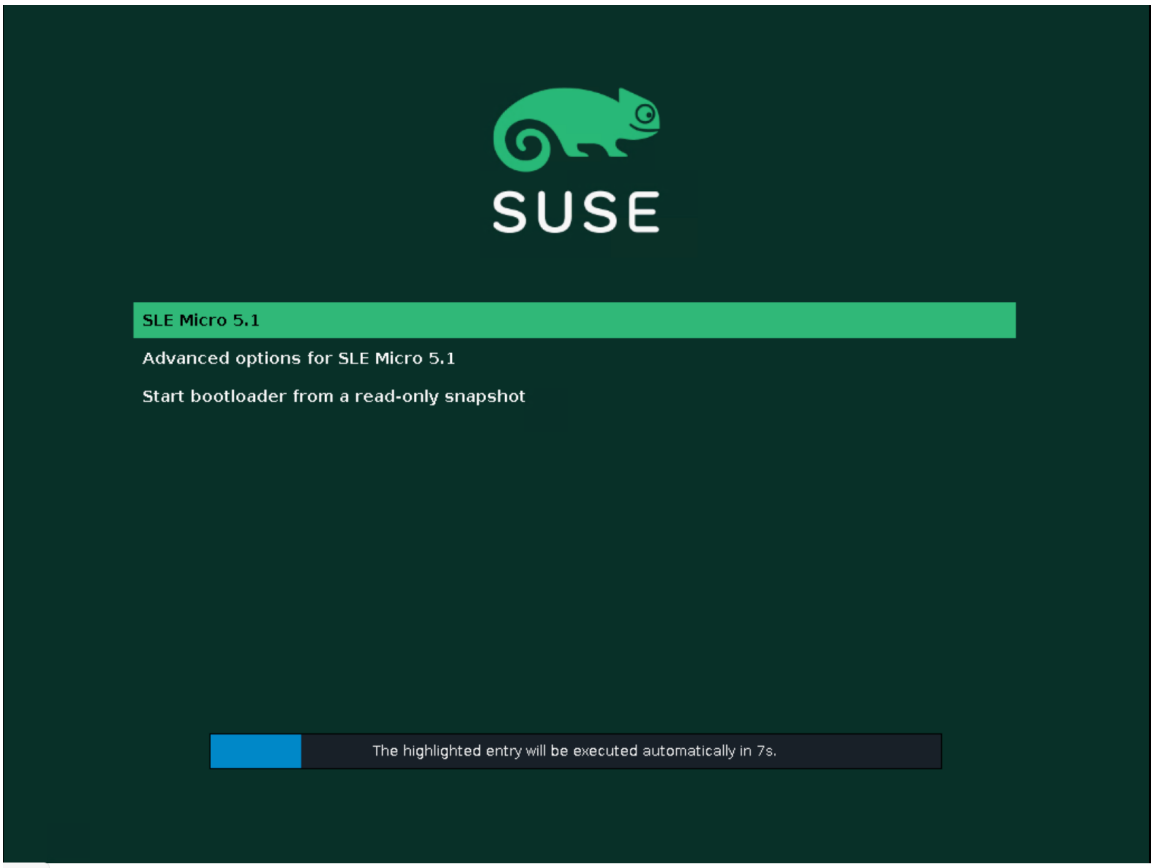
The installation process proceeds.



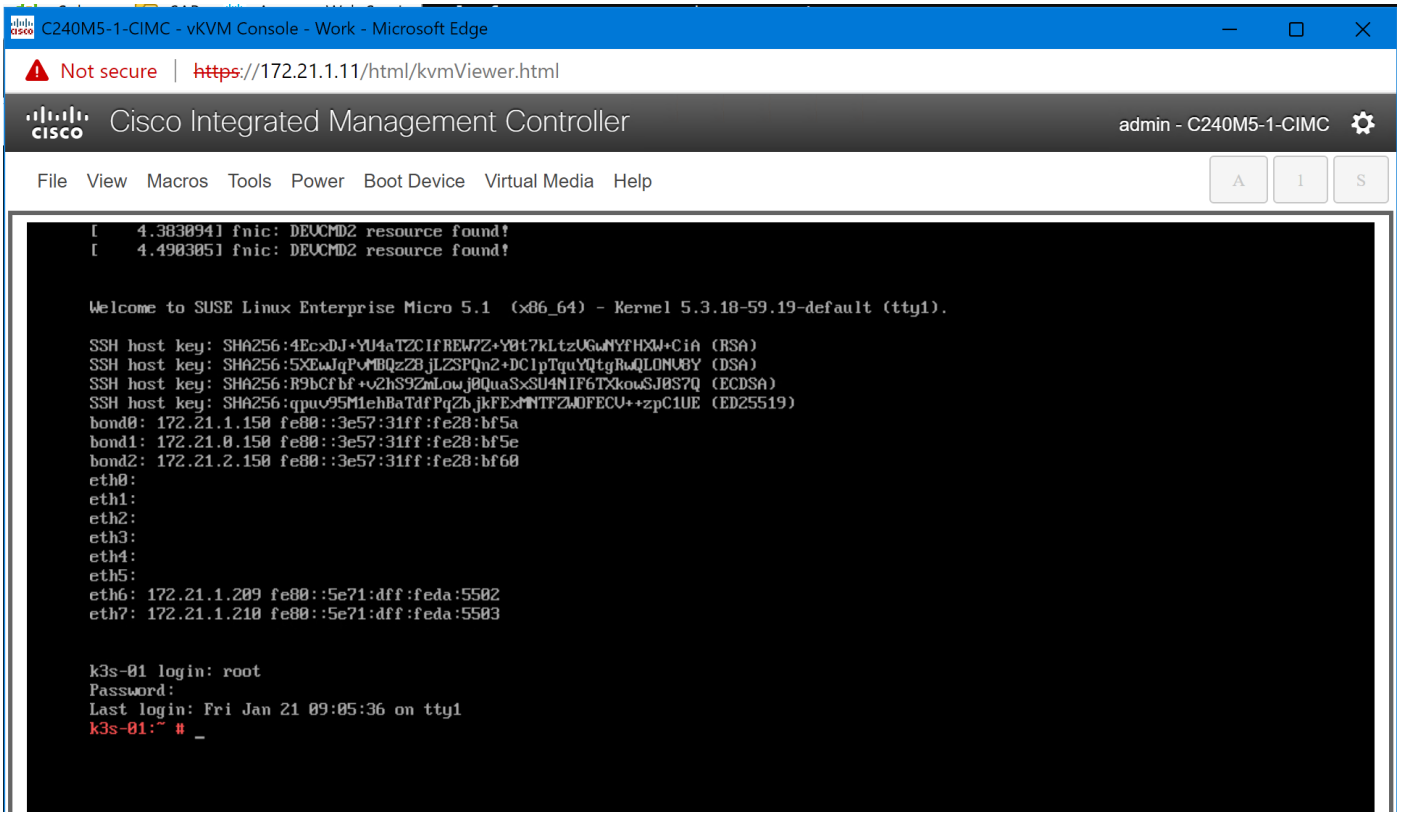
29. You must “eject” the CD/DVD as soon the installation process has finished and the reboot is initiated. Click Virtual Media > “SUSE-MicorOS-5.1-DVD-x86_64-GM-Media1.iso Mapped to CD/DVD” and confirm the ejection by clicking OK in the pop-up window.



After the installation is complete, the system will reboot automatically.



30. Log on to the system as the user root and using the password provided during the installation process.



31. Run the following commands to check the network configuration:

```
k3s-01:~ # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond0 state UP
group default qlen 1000
    link/ether 3c:57:31:28:bf:5a brd ff:ff:ff:ff:ff:ff
    altname enp64s0f0
.
.
.
10: bond2: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group
default qlen 1000
    link/ether 3c:57:31:28:bf:60 brd ff:ff:ff:ff:ff:ff
    inet 172.21.2.150/24 brd 172.21.2.255 scope global bond2
       valid_lft forever preferred_lft forever
    inet6 fe80::3e57:31ff:fe28:bf60/64 scope link
       valid_lft forever preferred_lft forever
11: bond1: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 3c:57:31:28:bf:5e brd ff:ff:ff:ff:ff:ff
    inet 172.21.0.150/24 brd 172.21.0.255 scope global bond1
       valid_lft forever preferred_lft forever
    inet6 fe80::3e57:31ff:fe28:bf5e/64 scope link
       valid_lft forever preferred_lft forever
12: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 3c:57:31:28:bf:5a brd ff:ff:ff:ff:ff:ff
    inet 172.21.1.150/24 brd 172.21.1.255 scope global bond0
       valid_lft forever preferred_lft forever
    inet6 fe80::3e57:31ff:fe28:bf5a/64 scope link
       valid_lft forever preferred_lft forever
k3s-01:~ #


k3s-01:~ # ip route
default via 172.21.1.1 dev eth6 proto dhcp
172.21.0.0/24 dev bond1 proto kernel scope link src 172.21.0.150
172.21.1.0/24 dev eth6 proto kernel scope link src 172.21.1.209
```

```
172.21.1.0/24 dev eth7 proto kernel scope link src 172.21.1.210
172.21.1.0/24 dev bond0 proto kernel scope link src 172.21.1.150
172.21.2.0/24 dev bond2 proto kernel scope link src 172.21.2.150
k3s-01:~ #

k3s-01:~ # cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: eth0
MII Status: up
Speed: 25000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 3c:57:31:28:bf:5a
Slave queue ID: 0

Slave Interface: eth1
MII Status: up
Speed: 25000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 3c:57:31:28:bf:5b
Slave queue ID: 0
k3s-01:~ #

k3s-01:~ # ping wdf02-4-pdc.wdf02-4-dmz.local. -c 3
PING wdf02-4-pdc.wdf02-4-dmz.local (172.20.0.50) 56(84) bytes of data.
64 bytes from wdf02-4-pdc.wdf02-4-dmz.local (172.20.0.50): icmp_seq=1 ttl=126 time=0.254 ms
64 bytes from wdf02-4-pdc.wdf02-4-dmz.local (172.20.0.50): icmp_seq=2 ttl=126 time=0.259 ms
64 bytes from wdf02-4-pdc.wdf02-4-dmz.local (172.20.0.50): icmp_seq=3 ttl=126 time=0.379 ms
```

```
--- wdf02-4-pdc.wdf02-4-dmz.local ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.254/0.297/0.379/0.059 ms
k3s-01:~ #

k3s-01:~ # ping www.google.de. -c 3
PING www.google.de (142.250.179.131) 56(84) bytes of data.
64 bytes from ams17s10-in-f3.1e100.net (142.250.179.131): icmp_seq=1 ttl=115 time=16.9 ms
64 bytes from ams17s10-in-f3.1e100.net (142.250.179.131): icmp_seq=2 ttl=115 time=16.9 ms
64 bytes from ams17s10-in-f3.1e100.net (142.250.179.131): icmp_seq=3 ttl=115 time=16.9 ms

--- www.google.de ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 16.915/16.937/16.963/0.019 ms
k3s-01:~ #
```

Install K3s

This section presents the installation procedure for the K3s software as described in [Rancher Docs: K3s - Lightweight Kubernetes](#).

1. Use the curl command to download the K3s software package and install

```
k3s-01:~ # curl -sfL https://get.k3s.io | sh -
[INFO]  Finding release for channel stable
[INFO]  Using v1.22.5+k3s1 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.22.5+k3s1/sha256sum-amd64.txt
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.22.5+k3s1/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
transactional-update 3.5.6 started
Options: --no-selfupdate -d run zypper --gpg-auto-import-keys install -y k3s-selinux
Separate /var detected.
2022-01-21 09:27:10 tukit 3.5.6 started
2022-01-21 09:27:10 Options: --discard -c1 open
2022-01-21 09:27:10 Using snapshot 1 as base for new snapshot 3.
2022-01-21 09:27:10 No previous snapshot to sync with - skipping
ID: 3
2022-01-21 09:27:10 Transaction completed.
2022-01-21 09:27:10 tukit 3.5.6 started
2022-01-21 09:27:10 Options: --discard call 3 zypper --gpg-auto-import-keys install -y k3s-selinux
2022-01-21 09:27:11 Executing `zypper --gpg-auto-import-keys install -y k3s-selinux`:
```

```
Building repository 'Rancher K3s Common (stable)' cache
.............................................................[done]
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  k3s-selinux

1 new package to install.
Overall download size: 20.0 KiB. Already cached: 0 B. After the operation, additional 85.1
KiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package k3s-selinux-0.5-1.SLE.noarch        (1/1),  20.0 KiB ( 85.1 KiB unpacked)
Retrieving: k3s-selinux-0.5-1.sle.noarch.rpm
.............................................................[done (713 B/s)]
k3s-selinux-0.5-1.sle.noarch.rpm:
    Header V4 RSA/SHA1 Signature, key ID e257814a: NOKEY
    V4 RSA/SHA1 Signature, key ID e257814a: NOKEY

Looking for gpg key ID E257814A in cache /var/cache/zypp/pubkeys.
Looking for gpg key ID E257814A in repository Rancher K3s Common (stable).
  gpgkey=https://rpm.rancher.io/public.key
Retrieving:
public.key...............................................................[done]

Automatically importing the following key:

  Repository:       Rancher K3s Common (stable)
  Key Fingerprint:  C8CF F216 4551 26E9 B9C9 18BE 925E A29A E257 814A
  Key Name:         Rancher (CI) <ci@rancher.com>
  Key Algorithm:    RSA 3072
  Key Created:      Tue Mar 10 22:43:06 2020
  Key Expires:      (does not expire)
  Subkey:           AA7E9EC8FE21FDCF 2020-03-10 [does not expire]
  Rpm Name:         gpg-pubkey-e257814a-5e6817fa

Note: A GPG pubkey is clearly identified by it's fingerprint. Do not rely the keys name. If
you are not sure whether the presented key is authentic, ask the repository provider or
check his web site. Many provider maintain a web page showing the fingerprints of the GPG
keys they are using.

Checking for file
conflicts:...................................................................[done]
```

```
(1/1) Installing: k3s-selinux-0.5-1.sle.noarch
.....................................................................[done]
Executing %posttrans
scripts.....................................................................[done]
2022-01-21 09:27:21 Application returned with exit status 0.
2022-01-21 09:27:22 Transaction completed.
2022-01-21 09:27:22 tukit 3.5.6 started
2022-01-21 09:27:22 Options: --discard close 3
2022-01-21 09:27:22 New default snapshot is #3 (/.snapshots/3/snapshot).
2022-01-21 09:27:22 Transaction completed.

Please reboot your machine to activate the changes and avoid data loss.
New default snapshot is #3 (/.snapshots/3/snapshot).
transactional-update finished
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Creating /usr/local/bin/crictl symlink to k3s
[INFO]  Creating /usr/local/bin/ctr symlink to k3s
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s.service
[INFO]  systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service →
/etc/systemd/system/k3s.service.
k3s-01:~ #
```

2. Use the **systemctl** command to start the K3s server and check the status.

```
k3s-01:~ # systemctl start k3s
k3s-01:~ # systemctl status k3s
● k3s.service - Lightweight Kubernetes
     Loaded: loaded (/etc/systemd/system/k3s.service; enabled; vendor preset: disabled)
     Active: active (running) since Fri 2022-01-21 09:37:50 UTC; 7min ago
       Docs: https://k3s.io
    Process: 2583 ExecStartPre=/bin/sh -xc ! /usr/bin/systemctl is-enabled --quiet nm-cloud-
setup.service (code=exited, s>
    Process: 2596 ExecStartPre=/sbin/modprobe br_netfilter (code=exited, status=0/SUCCESS)
    Process: 2610 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 2611 (k3s-server)
      Tasks: 225
…
k3s-01:~ #
```

3. Get basic information from the installed K3s cluster.

```
k3s-01:~ # kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:6443
CoreDNS is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
k3s-01:~ #
k3s-01:~ # kubectl get nodes -o wide
NAME     STATUS   ROLES                  AGE   VERSION        INTERNAL-IP    EXTERNAL-IP
OS-IMAGE                          KERNEL-VERSION         CONTAINER-RUNTIME
k3s-01   Ready    control-plane,master   15m   v1.22.5+k3s1   172.21.1.209   <none>
SUSE Linux Enterprise Micro 5.1   5.3.18-59.19-default   containerd://1.5.8-k3s1
k3s-01:~ #
k3s-01:~ # kubectl get all -A
NAMESPACE     NAME                                          READY  STATUS     RESTARTS    AGE
kube-system   pod/local-path-provisioner-64ffb68fd-7qs4m   1/1    Running    1 (11m ago) 16m
kube-system   pod/metrics-server-9cf544f65-nxrd2           1/1    Running    0           16m
kube-system   pod/helm-install-traefik-crd--1-gln5p        0/1    Completed  0           16m
kube-system   pod/helm-install-traefik--1-sf5dz            0/1    Completed  1           16m
kube-system   pod/svclb-traefik-24sf4                      2/2    Running    0           11m
kube-system   pod/coredns-85cb69466-vwbkc                  1/1    Running    1 (11m ago) 16m
kube-system   pod/traefik-786ff64748-x4cz5                 1/1    Running    0           11m

NAMESPACE     NAME                      TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)
AGE
default       service/kubernetes        ClusterIP      10.43.0.1      <none>         443/TCP
16m
kube-system   service/kube-dns          ClusterIP      10.43.0.10     <none>
53/UDP,53/TCP,9153/TCP       16m
kube-system   service/metrics-server    ClusterIP      10.43.136.93   <none>         443/TCP
16m
kube-system   service/traefik           LoadBalancer   10.43.32.86    172.21.1.209
80:32380/TCP,443:32713/TCP   11m

NAMESPACE     NAME                           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
NODE SELECTOR   AGE
kube-system   daemonset.apps/svclb-traefik   1         1         1       1            1
<none>          11m

NAMESPACE     NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/local-path-provisioner   1/1     1            1           16m
kube-system   deployment.apps/coredns                  1/1     1            1           16m
```

```
kube-system   deployment.apps/metrics-server          1/1     1            1           16m
kube-system   deployment.apps/traefik                 1/1     1            1           11m

NAMESPACE     NAME                                            DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/local-path-provisioner-64ffb68fd   1         1         1       16m
kube-system   replicaset.apps/coredns-85cb69466                 1         1         1       16m
kube-system   replicaset.apps/metrics-server-9cf544f65          1         1         1       16m
kube-system   replicaset.apps/traefik-786ff64748                1         1         1       11m

NAMESPACE     NAME                                COMPLETIONS   DURATION   AGE
kube-system   job.batch/helm-install-traefik-crd  1/1           5m16s      16m
kube-system   job.batch/helm-install-traefik      1/1           5m17s      16m
k3s-01:~ #
```

The system is now installed and is ready for more specific configurations dependent on local requirements.

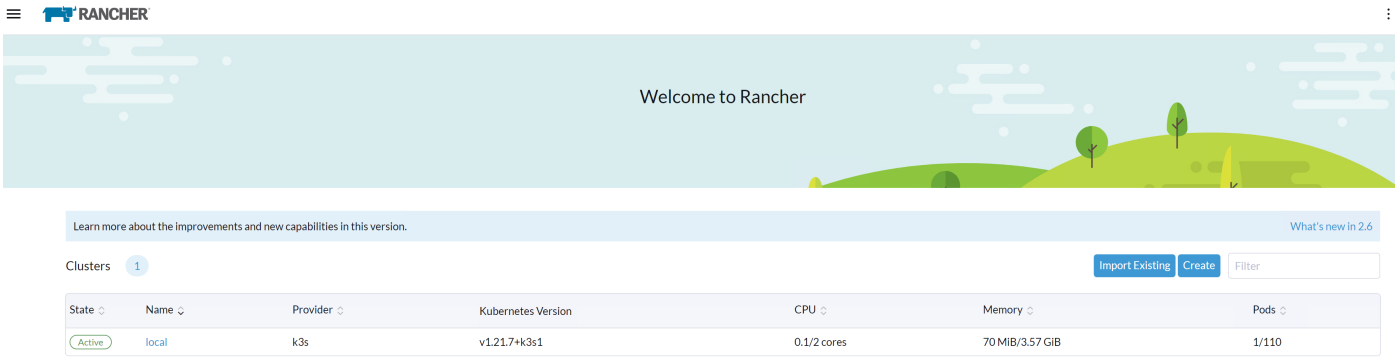
K3s integration into the workload management tool

Many options are available to manage a Kubernetes landscape with multiple clusters, with different workloads, and at different locations. We tested two options: integration into SUSE Rancher Kubernetes Operations Platform and integration into the Rafay Kubernetes Operations Platform.

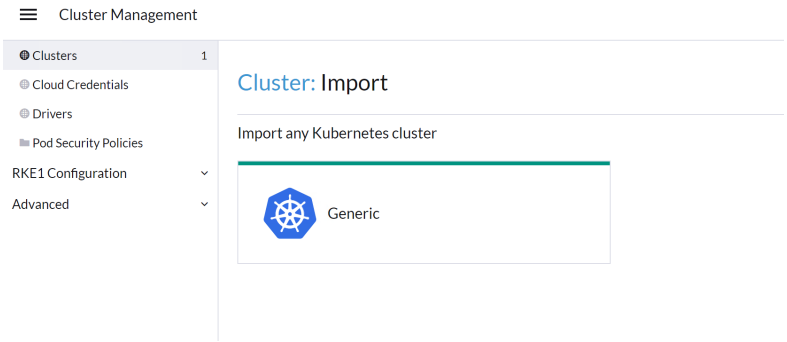
Integrate into Rancher Kubernetes Operations Platform

The obvious option for managing landscapes with SLE Micro and K3s components is SUSE Rancher. This section shows how to integrate a K3s system into the Rancher Kubernetes Operations Platform.

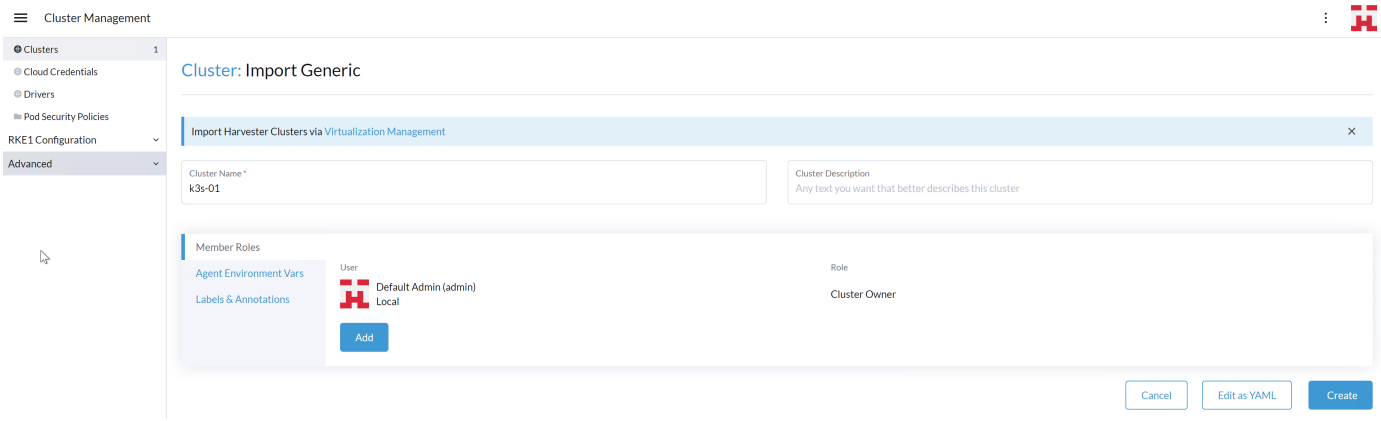
1. In the Rancher console, navigate to the list of clusters and click Import Existing.



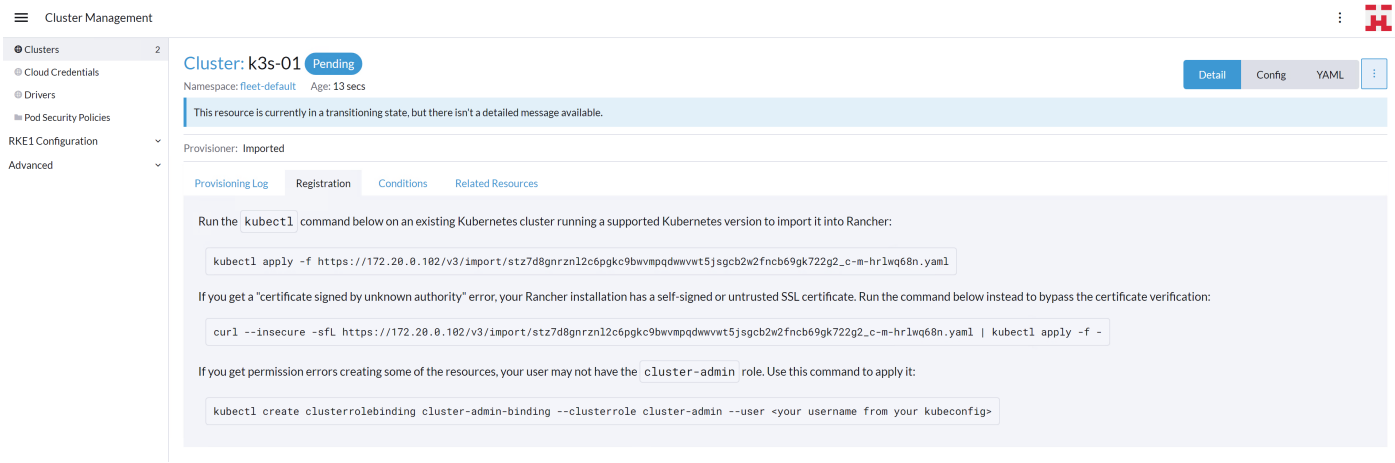
2. Click Generic.



3. Enter a cluster name and click Create.



4. Follow the steps shown on the next screen and click Done.

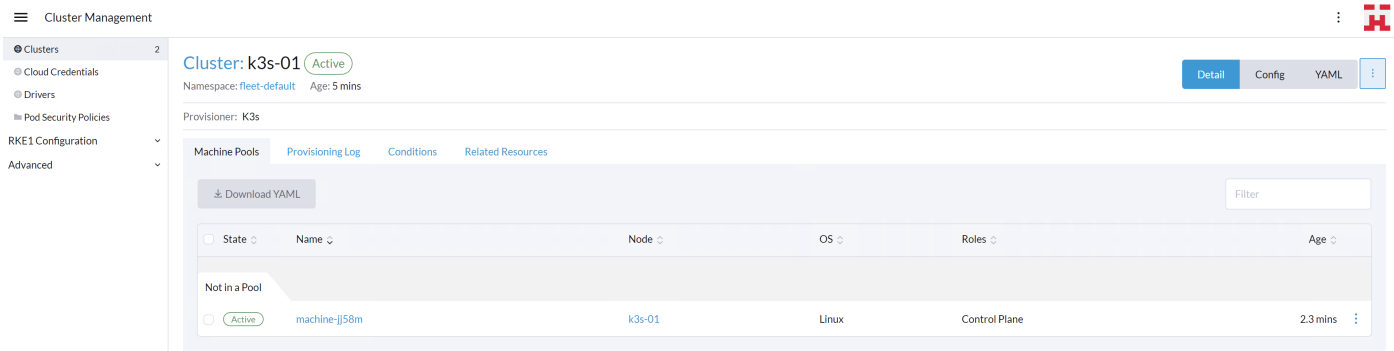


5. Log on to the installed k3s system and run the listed commands from the preceding screen.

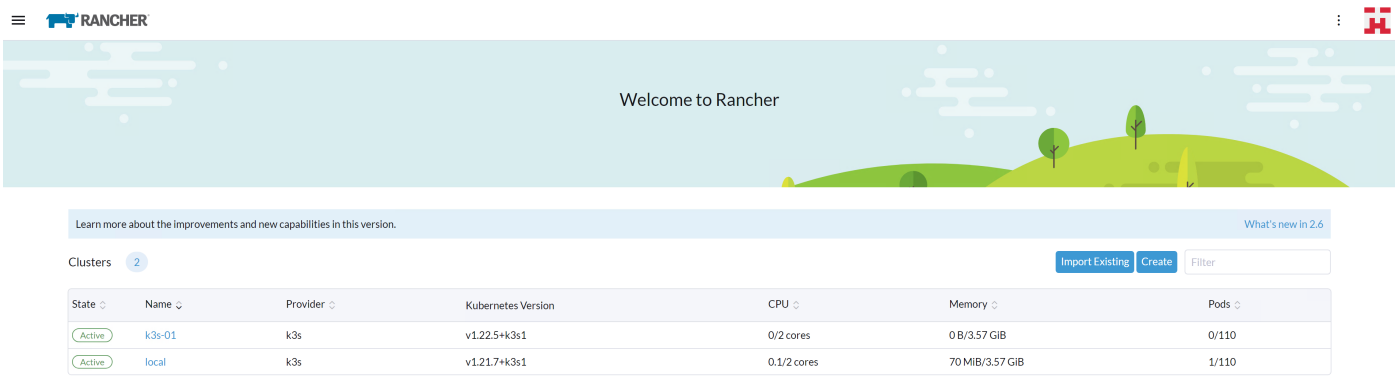
```
k3s-01:~ # kubectl create clusterrolebinding cluster-admin-binding \
>   --clusterrole cluster-admin \
>   --user root
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-binding created
k3s-01:~ #
k3s-01:~ # curl --insecure -sfL
https://172.20.0.102/v3/import/stz7d8gnrznl2c6pgkc9bwvmpqdwwvwt5jsgcb2w2fncb69gk722g2_c-m-
hrlwq68n.yaml | kubectl apply -f -
```

```
clusterrole.rbac.authorization.k8s.io/proxy-clusterrole-kubeapiserver created
clusterrolebinding.rbac.authorization.k8s.io/proxy-role-binding-kubernetes-master created
namespace/cattle-system created
serviceaccount/cattle created
clusterrolebinding.rbac.authorization.k8s.io/cattle-admin-binding created
secret/cattle-credentials-fad2056 created
clusterrole.rbac.authorization.k8s.io/cattle-admin created
Warning:
spec.template.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.node
SelectorTerms[0].matchExpressions[0].key: beta.kubernetes.io/os is deprecated since v1.14;
use "kubernetes.io/os" instead
deployment.apps/cattle-cluster-agent created
service/cattle-cluster-agent created
k3s-01:~ #
k3s-01:~ #
k3s-01:~ # kubectl get all -n cattle-system
NAME                                        READY   STATUS    RESTARTS   AGE
pod/cattle-cluster-agent-56d66975fc-t56mz   1/1     Running   0          60s

NAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/cattle-cluster-agent  ClusterIP   10.43.118.86   <none>        80/TCP,443/TCP   3m20s

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/cattle-cluster-agent   1/1     1            1           3m20s

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/cattle-cluster-agent-56d66975fc   1         1         1       60s
replicaset.apps/cattle-cluster-agent-857c647888   0         0         0       3m20s
k3s-01:~ #
```

6. Return to the Rancher user interface. The cluster is now shown as Active.



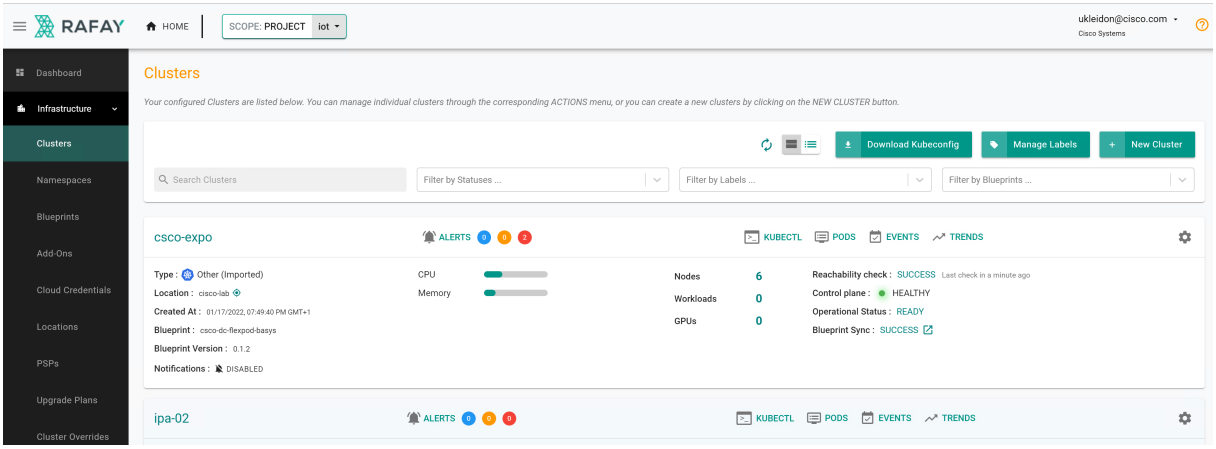
7. View the home screen. The high-level information of the cluster is shown in the home screen of Rancher.



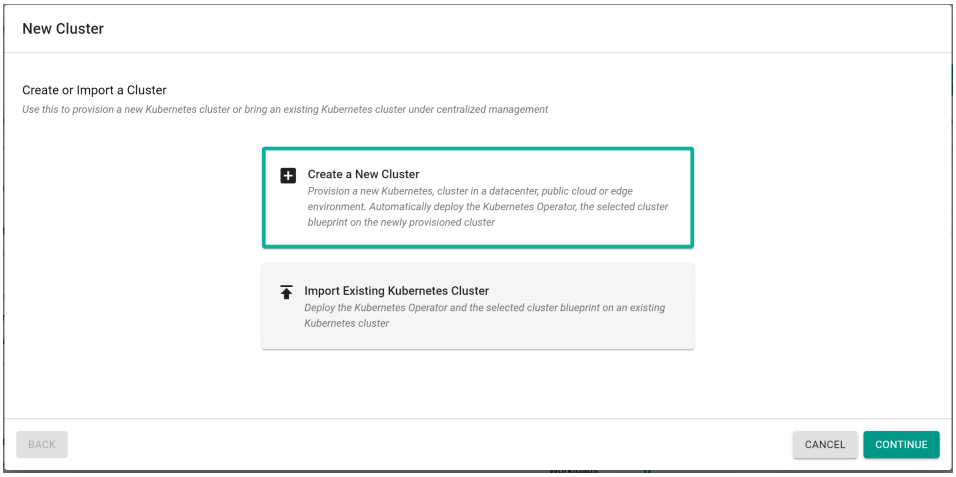
Integrate into Rafay Kubernetes Operations Platform

To demonstrate the manageability of an SLE Micro and K3s system with another tool, this section shows integration into the Rafay Kubernetes management console.

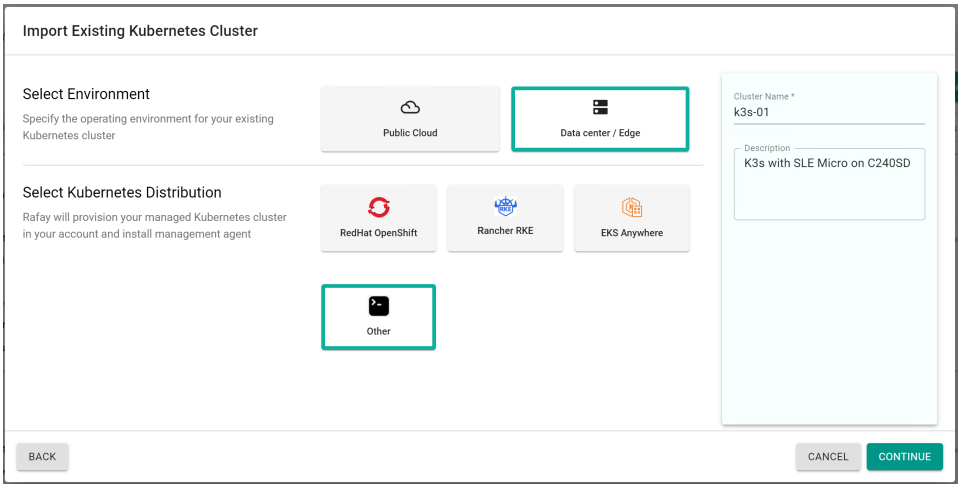
1. In the Rafay console, navigate to the list of clusters in the project of choice and click New Cluster.



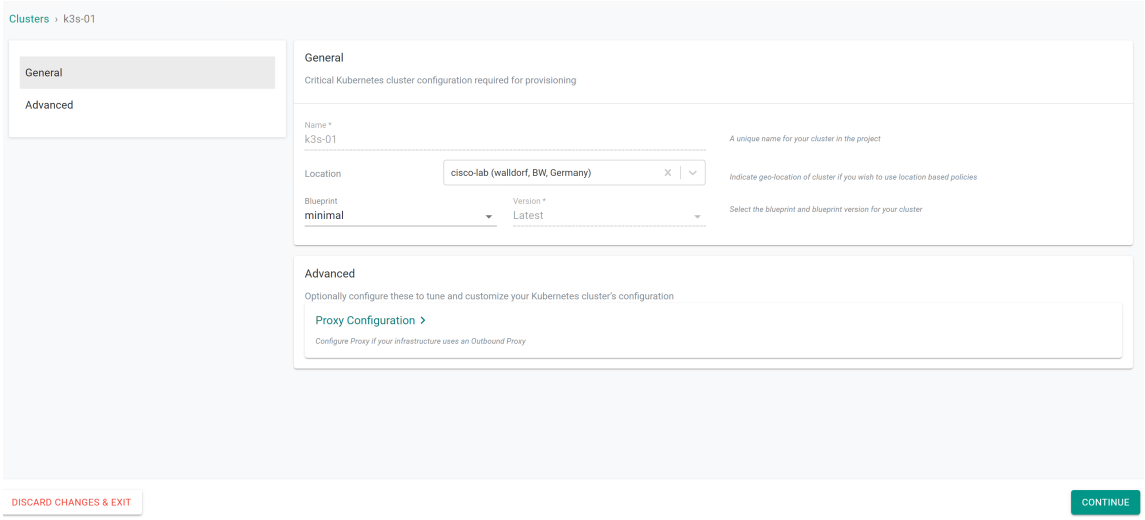
2. Click Import Existing Kubernetes Cluster and click Continue.



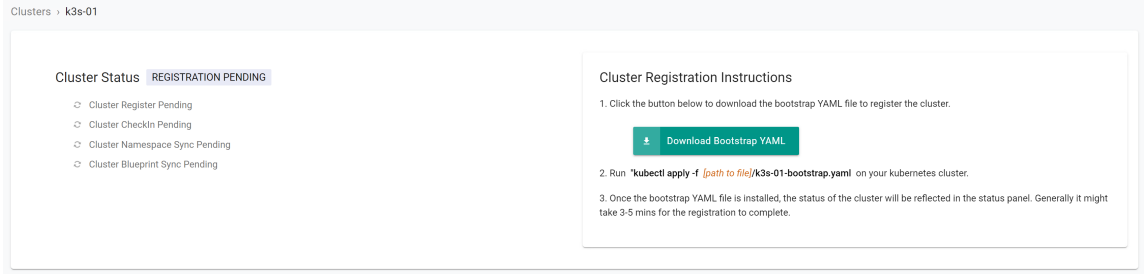
3. Click Data center / Edge and then click Other. Enter a name for the new cluster and a description if wanted. Click Continue.



4. Select the location and deployment blueprint for this setup. If this is the first time a K3s cluster will be integrated into the Rafay system, the best practice is to start with the blueprint minimal or default. Those are the basic blueprints from Rafay to make the system work (minimal) or to add components such as monitoring and reporting (default). Then click Continue.



5. Download the Bootstrap YAML file to the K3s system.

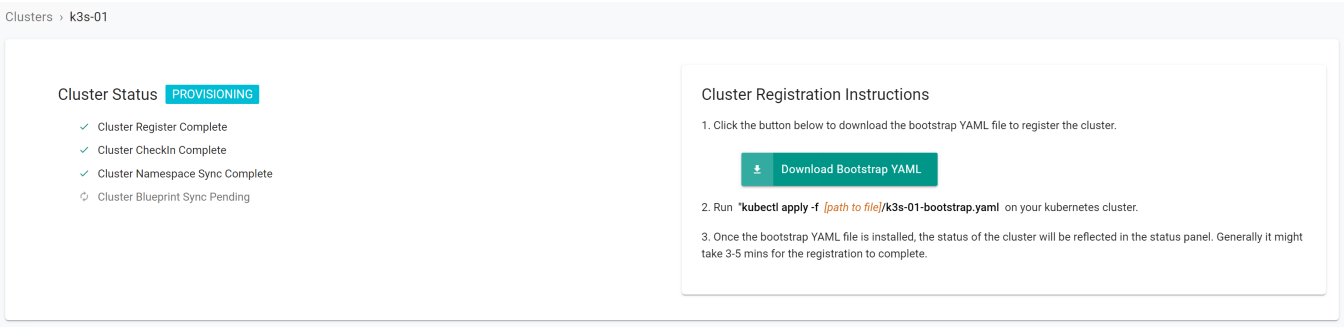


6. Log on to the K3s system and apply the bootstrap file.

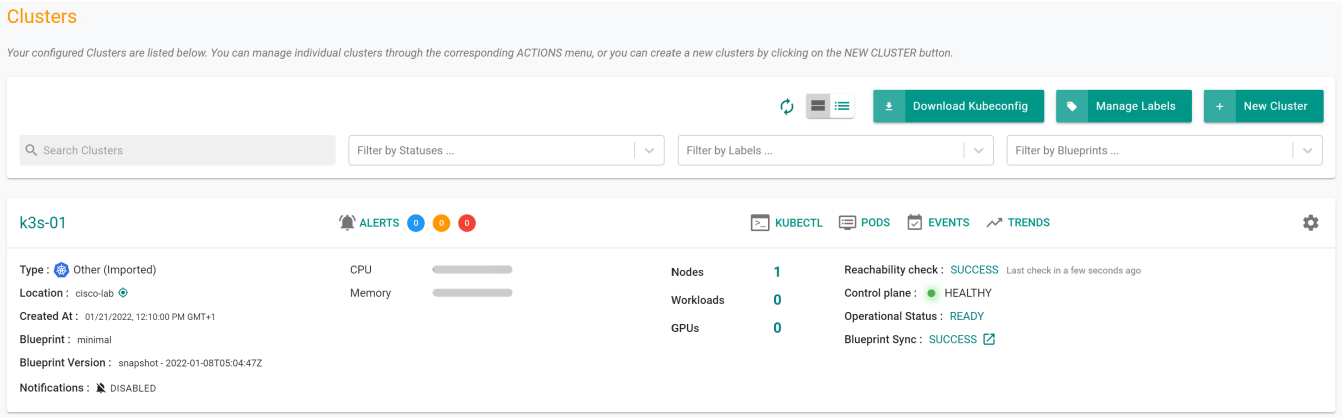
```
k3s-01:~ # ls -l /tmp/k3s-01-bootstrap.yaml
-rwxr-xr-x 1 root root 13801 Jan 21 11:16 /tmp/k3s-01-bootstrap.yaml
k3s-01:~ #
```

```
k3s-01:~ # kubectl apply -f /tmp/k3s-01-bootstrap.yaml
namespace/rafay-system created
serviceaccount/system-sa created
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/rafay-privileged-psp created
clusterrole.rbac.authorization.k8s.io/rafay:manager created
clusterrolebinding.rbac.authorization.k8s.io/rafay:rafay-system:manager-rolebinding created
clusterrole.rbac.authorization.k8s.io/rafay:proxy-role created
clusterrolebinding.rbac.authorization.k8s.io/rafay:rafay-system:proxy-rolebinding created
priorityclass.scheduling.k8s.io/rafay-cluster-critical created
role.rbac.authorization.k8s.io/rafay:leader-election-role created
rolebinding.rbac.authorization.k8s.io/rafay:leader-election-rolebinding created
customresourcedefinition.apiextensions.k8s.io/namespaces.cluster.rafay.dev created
customresourcedefinition.apiextensions.k8s.io/tasklets.cluster.rafay.dev created
customresourcedefinition.apiextensions.k8s.io/tasks.cluster.rafay.dev created
service/controller-manager-metrics-service-v3 created
deployment.apps/controller-manager-v3 created
configmap/connector-config-v3 created
configmap/proxy-config-v3 created
deployment.apps/rafay-connector-v3 created
service/rafay-drift-v3 created
validatingwebhookconfiguration.admissionregistration.k8s.io/rafay-drift-validate-v3 created
k3s-01:~ #
```

The process is shown in the Rafay console.



7. After the deployment is finished, the cluster is shown in the list with basic information about the status.



8. On the K3s system, a new namespace rafay-system is created to enable communication between the Rafay Kubernetes Operations Platform and the local K3s system.

```
k3s-01:~ # kubectl get all -A
NAMESPACE       NAME                                        READY STATUS      RESTARTS       AGE
kube-system     pod/local-path-provisioner-64ffb68fd-7qs4m  1/1   Running     1 (110m ago)   115m
kube-system     pod/metrics-server-9cf544f65-nxrd2          1/1   Running     0              115m
kube-system     pod/helm-install-traefik-crd--1-gln5p       0/1   Completed   0              115m
kube-system     pod/helm-install-traefik--1-sf5dz           0/1   Completed   1              115m
kube-system     pod/svclb-traefik-24sf4                     2/2   Running     0              110m
kube-system     pod/coredns-85cb69466-vwbkc                 1/1   Running     1 (110m ago)   115m
kube-system     pod/traefik-786ff64748-x4cz5                1/1   Running     0              110m
rafay-system    pod/edge-client-8c7748dfb-sk4l6             1/1   Running     0              8m21s
rafay-system    pod/relay-agent-78d645bc89-9w6qw            1/1   Running     0              8m20s
rafay-system    pod/controller-manager-v3-6bb696cc8b-5bsch  1/1   Running     0              6m13s
rafay-system    pod/rafay-connector-v3-6c8dcf8cf9-9m84r     1/1   Running     1 (5m33s ago)  6m14s

NAMESPACE       NAME                                          TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)                      AGE
default         service/kubernetes                            ClusterIP      10.43.0.1
<none>          443/TCP                      115m
kube-system     service/kube-dns                              ClusterIP      10.43.0.10
<none>          53/UDP,53/TCP,9153/TCP       115m
kube-system     service/metrics-server                        ClusterIP      10.43.136.93
<none>          443/TCP                      115m
kube-system     service/traefik                               LoadBalancer   10.43.32.86
172.21.1.209    80:32380/TCP,443:32713/TCP   110m
rafay-system    service/controller-manager-metrics-service-v3 ClusterIP      10.43.9.227
<none>          8443/TCP                     10m
rafay-system    service/rafay-drift-v3                        ClusterIP      10.43.2.198
<none>          8081/TCP                     10m
```

```
NAMESPACE     NAME                          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
NODE SELECTOR   AGE
kube-system   daemonset.apps/svclb-traefik   1         1         1       1            1
<none>          110m

NAMESPACE      NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
kube-system    deployment.apps/local-path-provisioner   1/1     1            1           115m
kube-system    deployment.apps/coredns                  1/1     1            1           115m
kube-system    deployment.apps/metrics-server           1/1     1            1           115m
kube-system    deployment.apps/traefik                  1/1     1            1           110m
rafay-system   deployment.apps/edge-client              1/1     1            1           8m22s
rafay-system   deployment.apps/relay-agent              1/1     1            1           8m21s
rafay-system   deployment.apps/controller-manager-v3    1/1     1            1           10m
rafay-system   deployment.apps/rafay-connector-v3       1/1     1            1           10m

NAMESPACE      NAME                                                DESIRED   CURRENT  READY AGE
kube-system    replicaset.apps/local-path-provisioner-64ffb68fd   1         1        1     115m
kube-system    replicaset.apps/coredns-85cb69466                  1         1        1     115m
kube-system    replicaset.apps/metrics-server-9cf544f65           1         1        1     115m
kube-system    replicaset.apps/traefik-786ff64748                 1         1        1     110m
rafay-system   replicaset.apps/edge-client-8c7748dfb              1         1        1     8m22s
rafay-system   replicaset.apps/relay-agent-78d645bc89             1         1        1
8m21s
rafay-system   replicaset.apps/rafay-connector-v3-88ff764c5       0         0        0     10m
rafay-system   replicaset.apps/controller-manager-v3-6bb696cc8b   1         1        1     6m14s
rafay-system   replicaset.apps/controller-manager-v3-7785d7b9d4   0         0        0     10m
rafay-system   replicaset.apps/rafay-connector-v3-6c8dcf8cf9      1         1        1     6m15s

NAMESPACE     NAME                                 COMPLETIONS   DURATION   AGE
kube-system   job.batch/helm-install-traefik-crd   1/1           5m16s      115m
kube-system   job.batch/helm-install-traefik       1/1           5m17s      115m
k3s-01:~ #
```

Conclusion

The combination of SUSE Linux Enterprise Micro, the lightweight Kubernetes system K3s, and the Cisco UCS C220, C240, and C240 SD servers can run modern cloud-native applications developed for Kubernetes in a single server deployed in a short-depth network cabinet.

For more information

For additional information, see the following resources:

- https://suse.com/products/micro
- https://k3s.io/
- https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/c/hw/c240sdm5/install/c240sdm5.html
- https://rafay.co/

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA 222265.1 01/23