2. Bare Metal Provisioning with MaaS

MCP does not provision bare metal servers directly. Instead Mirantis recommends using MaaS ("Metal as a Service"), a standalone open source tool for provisioning operating system and setting up bare metal nodes.

In this lab, we will install MaaS, configure it, and use it to provision operating system on multiple servers in parallel.

Chapter Details	
Chapter Goal	Install and configure MaaS and use it to provision bare metal nodes
Chapter Sections	*2.1. Introduction to MaaS* *2.2. Explore the Environment* *2.3. Install and Configure MaaS* *2.4. Provision the Nodes* *2.5. Customize Provisioning*

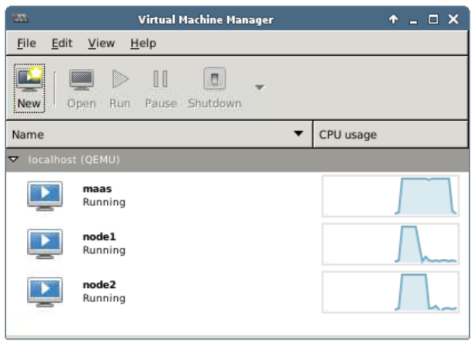
2.1. Introduction to MaaS

Metal as a Service (MaaS) is an open source tool for provisioning bare metal servers. MaaS is developed by Canonical and released under GNU Affero General Public License (AGPL) v3.

In MCP, we use MaaS to install Ubuntu on cluster nodes.

2.2. Explore the Environment

Let's start from exploring your lab environment. On your host, start `Virtual Machine Manager`. You should see several VMs that we have created for you:



The VMs `node1` and `node2` do not have operating system yet (we will provision it later) and they are configured to boot via PXE.

The dedicated `maas` VM has operating system (Ubuntu 16.04) installed. We will use this VM to install MaaS and provision other nodes. You can use the following credentials to log in to the `maas` VM:

Host name	IP address	User	Password

Host name	IP address	User	Password
maas	172.16.1.2	ubuntu	stack

2.3. Install and Configure MaaS

2.3.1. Install MaaS

Step 1 Log in to the `maas` node using `ubuntu` for the user name and `stack` for the password:

```
stack@lab:~$ ssh ubuntu@172.16.1.2
ubuntu@172.16.1.2's password: stack
```

Step 2 Add an additional apt repository for MaaS:

```
ubuntu@maas:~$ sudo add-apt-repository ppa:maas/stable
Press [ENTER] to continue
```

Step 3 Update the apt package index:

```
ubuntu@maas:~$ sudo apt-get -q update
```

Step 4 Install `maas` with the specified version:

```
ubuntu@maas:~$ sudo apt-get install -y maas
```

Step 5 Install `libvirt-bin` package:

```
ubuntu@maas:~$ sudo apt-get install -y libvirt-bin
```

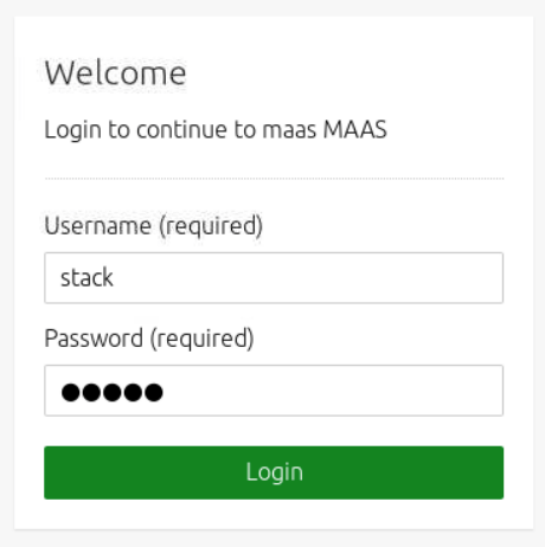
The `libvirt-bin` package is required for MaaS to enable power management for the VMs we are going to provision.

2.3.2. Configure MaaS

Step 1 To create an administrative account for MaaS on `maas` node execute the following command:

```
ubuntu@maas:~$ sudo maas createadmin \
--username=stack \
--password=stack \
--email='anyemail@anydomain.com'
```

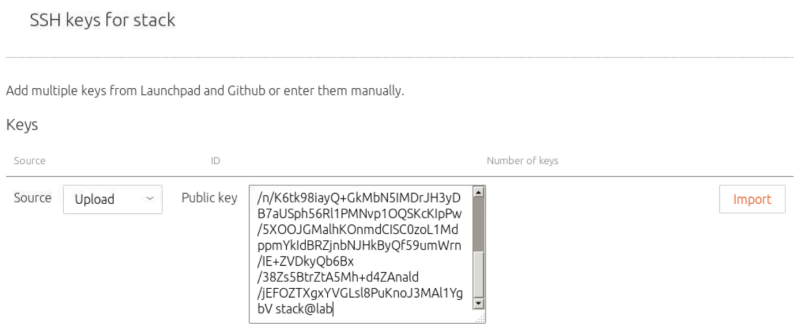
Step 2 On your host, launch Firefox and go to the address http://172.16.1.2:5240. Use `stack`/`stack` as username and password:



Step 3 Enter `8.8.8.8` in the `DNS forwarder`:

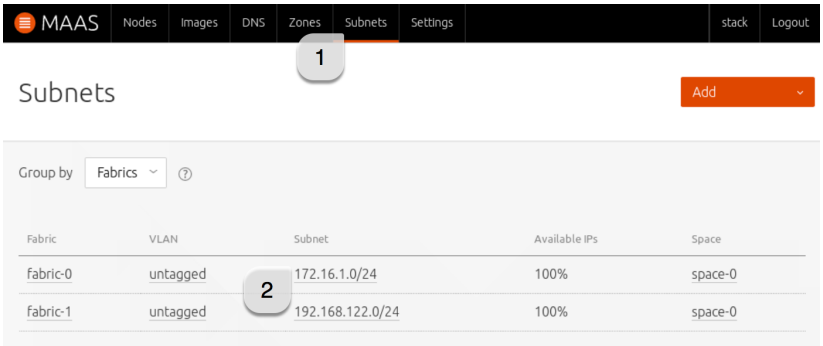
Then press `Continue` in the bottom of the page.

Step 4 On the next page, choose `Upload` as a source for SSH key, copy and paste the public key from the `~/.ssh/id_rsa.pub` file, which is located on your host:

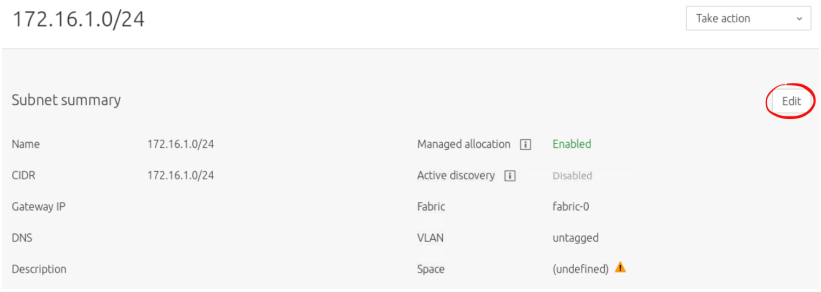


Then press `Import` and `Go to dashboad` in the bottom of the page.

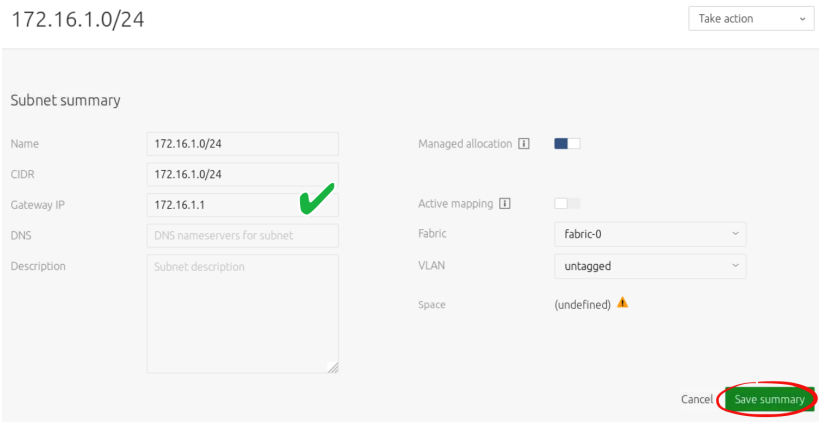
Step 5 On the MaaS dashboard, choose `Subnets`, then choose the `172.16.1.0/24` subnet:



Step 6 On the subnet page click `Edit`:

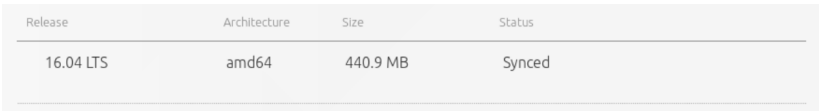


Step 7 Enter `172.16.1.1` for `Gateway IP` then click `Save Summary`:



> **Important:**
>
> In this lab, we use the existing DHCP server to assign IP addresses dynamically. Do not enable built-in DHCP server in MaaS to prevent the address namespace from being corrupted.

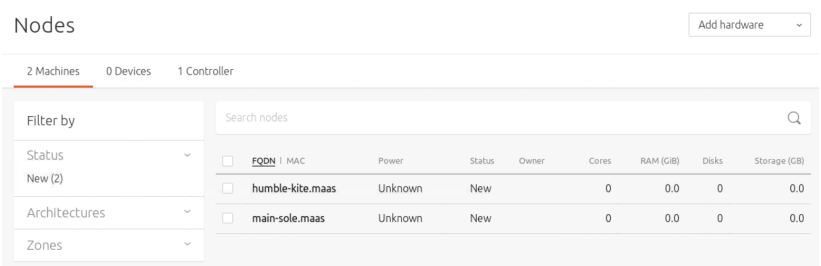
Step 8 Wait while MaaS is downloading the Ubuntu image. On the MaaS dashboard, choose `Images` and wait for `Synced` in the status column:



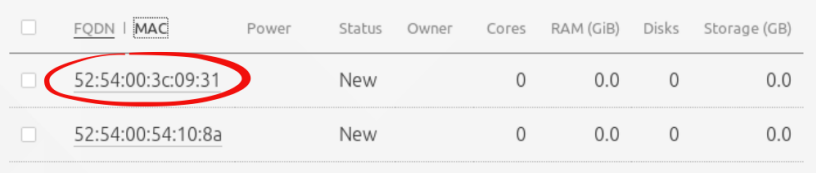
2.4. Provision the Nodes

Step 1 Reboot `node1` and `node2` (you can use `Force Reset` from the `Shutdown` drop down menu in Virtual Machine Manager). These VMs will be discovered by MaaS via PXE.

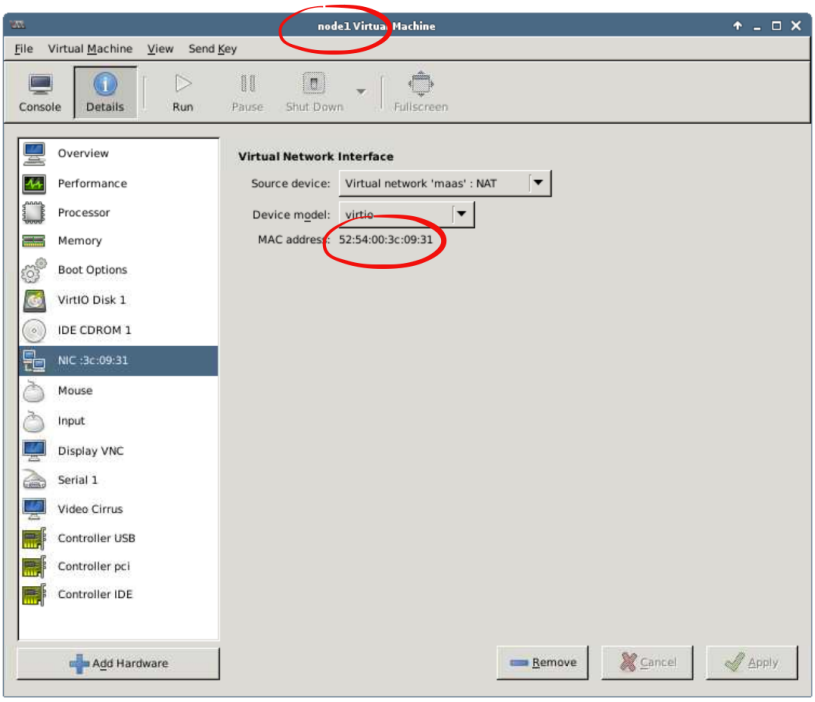
Step 2 On the MaaS dashboard, choose `Nodes` and wait while MaaS is discovering the nodes. Finally, you should see two nodes in the `New` status:



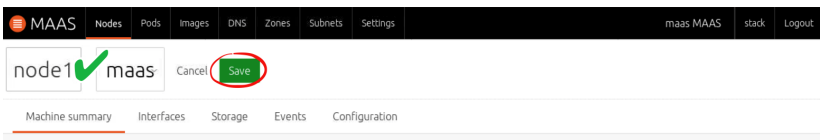
Step 3 Note that MaaS has assigned randomly generated names to the nodes. Press `MAC` in the table header to show MAC addresses for the nodes:



You can match the node's MAC address in MaaS and in Virtual Machine Manager (double click on VM, then choose `Details`, `NIC`) to set a proper name for the node:



Step 4 Edit each discovered node name by clicking on the existing name and typing in a new desired name (`node1` and `node2`):



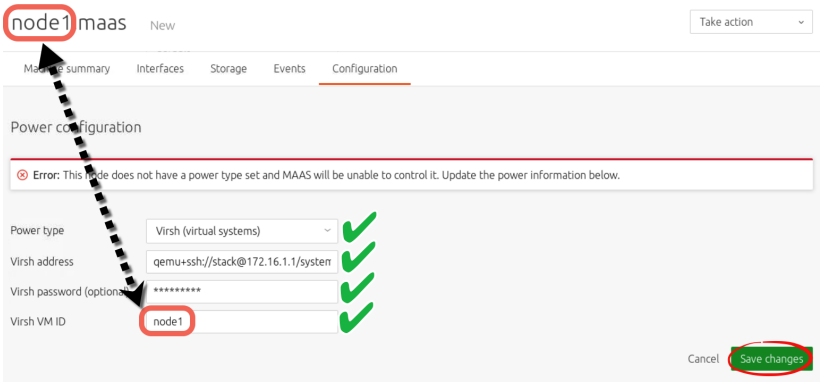
Click `Save` after editing the name.

Now we will edit each node again and set power management options for each node. This will allow to turn on and off nodes automatically. For physical servers, you need to specify vendor-specific parameters. Since we use virtual machines for the nodes, we should tell MaaS how to control the power state of the VMs running on the host.

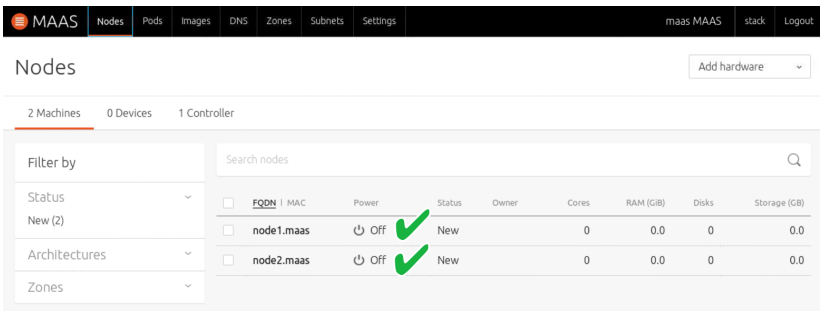
Step 5 Under the `Configuration` tab of each node, scroll down and select `Virsh (virtual systems)` for the `Power type` drop-down menu. Additional fields will appear below the power configuration. Input the following parameters:

- **Virsh address**: `qemu+ssh://stack@172.16.1.1/system`
- **Virsh password (optional)**: <Host password provided by instructor>
- **Virsh VM ID**: <Selected host name>

Click `Save changes` when finished. Make sure to apply this step for both of your nodes.

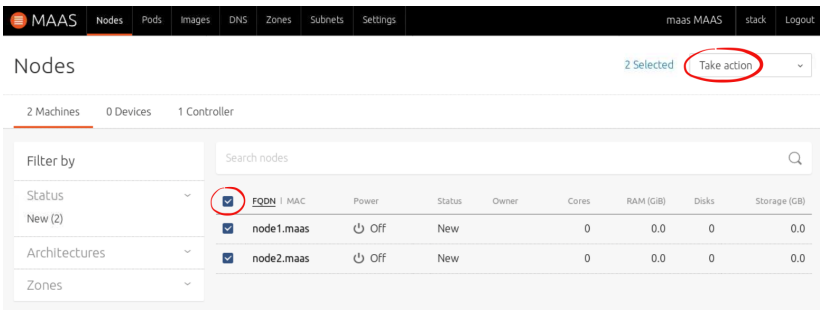


Check that MaaS detected the current power state of the nodes (double check node's power management parameters if it did not).

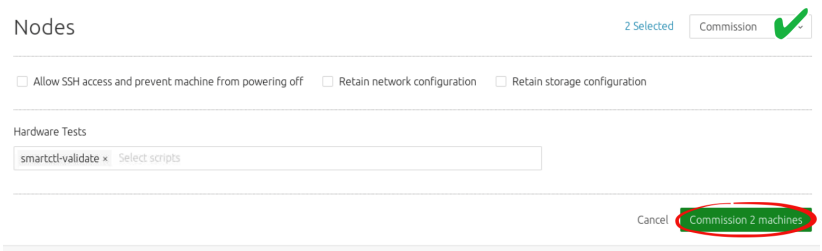


2.4.1. Commission the Nodes

Step 1 In the `Nodes` tab in MaaS UI, select all the nodes then click `Take action`:



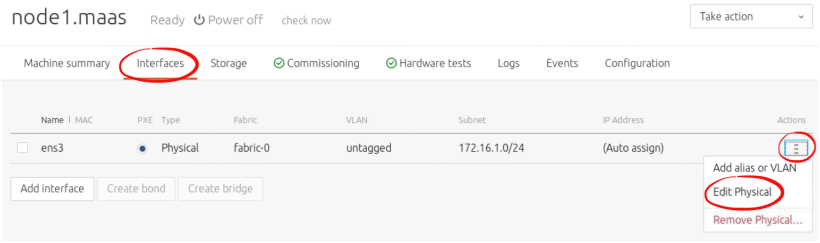
Step 2 From the `Take action` drop-down menu, click `Commission`. Leave all fields as-is and click `Commission 2 machines`:



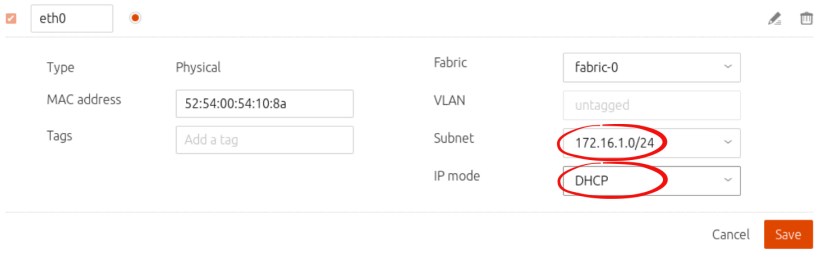
Wait while MaaS is commissioning the nodes. You should see the nodes `Power` status change to On and `Ready` as the status.

2.4.2. Provision the Nodes via UI

Step 1 Edit each node again. In the `Interfaces` section, edit the existing interface, for example `ens3` (can have a different name in your case):



Step 2 Choose `172.16.1.0/24` for `Subnet` and `DHCP` for `IP mode`, then press `Save`:



Ensure that you have done this for both of your existing nodes.

Step 3 Select both nodes, choose `Take action`, `Deploy`, then press `Deploy 2 machines`. Wait while MaaS is deploying the node. Finally, you should see the nodes in the `Deployed` status.

Step 4 Check that you can log in to the provisioned nodes via ssh using the provided ssh key. Our external DHCP server has assigned the following IP addresses to the nodes:

Host name	IP address	User	Password
node1	172.16.1.101	ubuntu	N/A
node2	172.16.1.102	ubuntu	N/A

Check that you can access `node1` and `node2`:

```
stack@lab:~$ ssh ubuntu@172.16.1.101
...
ubuntu@node1:~$ exit

stack@lab:~$ ssh ubuntu@172.16.1.102
...
ubuntu@node2:~$ exit
```

> **Notes:**
>
> When MaaS is providing DHCP, you can get the assigned IP addresses directly in the MaaS UI.

2.4.3. Release the Nodes

Before going further, let's release our nodes. In the MaaS UI, select both nodes, choose `Take action`, `Release`, then press `Release 2 machines`. The nodes should be in the `Ready` status.

2.4.4. Provision the Nodes via CLI

Step 1 To enable MaaS CLI log in to the `maas` node using `ubuntu` for the user name and `stack` for the password:

```
stack@lab:~$ ssh ubuntu@172.16.1.2
ubuntu@172.16.1.2's password: stack
ubuntu@maas:~$
```

Step 2 Save MaaS API key in the `maas_api_key`:

```
ubuntu@maas:~$ sudo maas-region apikey --username=stack > maas_api_key
```

Step 3 Log in to MaaS using the provided API key:

```
ubuntu@maas:~$ maas login stack http://localhost:5240/MAAS/api/2.0 - < maas_api_key

You are now logged in to the MAAS server at
http://localhost:5240/MAAS/api/2.0/ with the profile name 'stack'.

For help with the available commands, try:

  maas stack --help
```

Step 4 Verify that MaaS CLI works:

```
ubuntu@maas:~$ maas stack machines read
Success.
Machine-readable output follows:
...
```

Step 5 On the `maas` node, use the following commands to list the system IDs for the allocated nodes:

```
ubuntu@maas:~$ for n in {node1,node2} ; do
maas stack machines read hostname=$n | grep system_id -m 1 | cut -d '"' -f 4
done

n7wnrs
amncxh
```

Note that you will probably have different system IDs in your lab.

Step 6 To allocate (acquire) the nodes execute the following commands (use the actual machine IDs from the previous output):

```
ubuntu@maas:~$ maas stack machines allocate system_id=n7wnrs
...
ubuntu@maas:~$ maas stack machines allocate system_id=amncxh
...
```

You may check that the nodes have the `Allocated` status in the MaaS UI.

Step 7 To deploy (provision) the nodes execute the following commands (use the actual machine IDs):

```
ubuntu@maas:~$ maas stack machine deploy n7wnrs
...
ubuntu@maas:~$ maas stack machine deploy amncxh
...
```

You may check that the nodes have the `Deploying` status in the MaaS UI.

2.5. Customize Provisioning

Step 1 Before going further, let's release our nodes again. In the MaaS UI, select both nodes, choose `Take action`, `Release`, then press `Release 2 machines`. The nodes should be in the `Ready` status.

Step 2 On the `maas` node copy the existing file `curtin_userdata` to `curtin_userdata_amd64_generic_xenial_node1` in the directory `/etc/maas/preseeds`:

```
ubuntu@maas:~$ cd /etc/maas/preseeds
ubuntu@maas:/etc/maas/preseeds$ sudo cp curtin_userdata \
curtin_userdata_amd64_generic_xenial_node1
```

Step 3 As root, edit the file `curtin_userdata_amd64_generic_xenial_node1` in the directory `/etc/maas/preseeds`:

```
# You need to use sudo to edit the file
ubuntu@maas:/etc/maas/preseeds$ sudo vim curtin_userdata_amd64_generic_xenial_node1
```

Find the `late_commands:` key, and add the following line (starting with `touch`) after it:

```
...
late_commands:
  touch: ["curtin", "in-target", "--", "touch", "/root/this_is_node1"]
...
```

Step 4 Provision both nodes via UI or CLI.

Step 5 On the host, check that the file `/root/this_is_node1` exists on `node1` and does not exist on `node2` (you may need to remove old ssh key fingerprints in order to log in to the nodes):

```
stack@lab:~$ ssh-keygen -f "/home/stack/.ssh/known_hosts" -R 172.16.1.101
...

stack@lab:~$ ssh-keygen -f "/home/stack/.ssh/known_hosts" -R 172.16.1.102
...

stack@lab:~$ ssh ubuntu@172.16.1.101 'sudo ls /root' | grep node
this_is_node1

stack@lab:~$ ssh ubuntu@172.16.1.102 'sudo ls /root' | grep node
```

Checkpoint:

- Install and configure MaaS
- Provision bare metal nodes with MaaS (UI and CLI)
- Use a custom Curtin file to customize OS deployment