



PROJECT REPORT

Research and deploy OpenNebula

Subject: Calculation and distribution system

Grade: NT533.O22.MMCL

Instructors : Bui Thanh Binh

STUDENTS GROUP:

First and last name	Student ID
Le Hoang Khanh	21522205
Nguyen Dinh Khoa	21522227
Le Xuan Linh	21522286

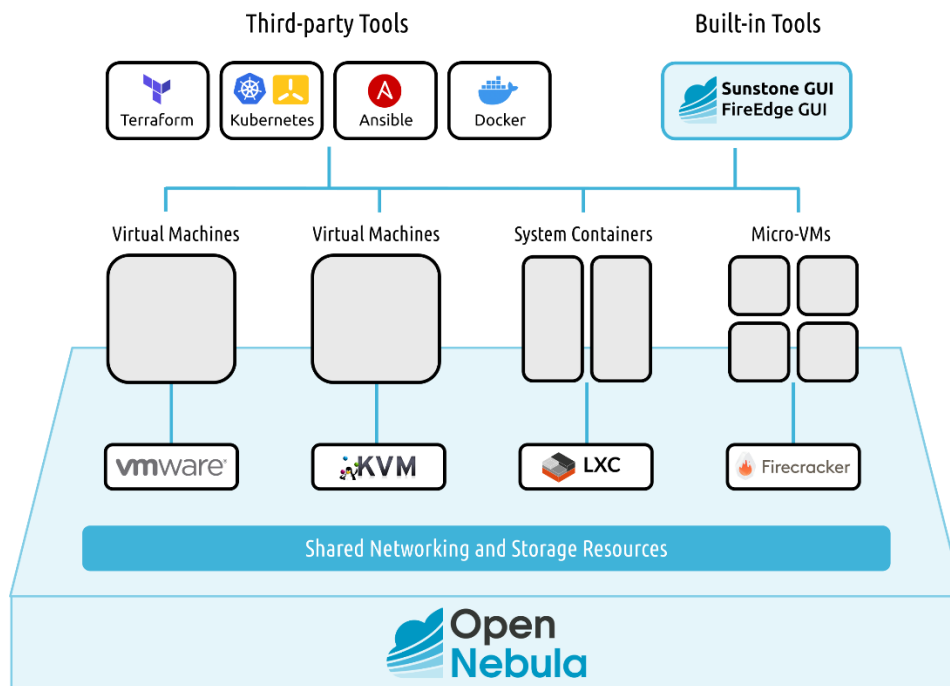
TABLE OF CONTENTS

A.	DETAILED REPORT	3
1.	Learn about OpenNebula.....	3
a.	Introducing OpenNebula	3
b.	OpenNebula Model for Cloud Users	4
c.	Architecture of OpenNebula.	6
d.	OpenNebula components	6
2.	Compare with another IssA Platfrom	9
3.	System architecture to be implemented.	11
4.	Deploying Front-end Node.	12
5.	Deploying KVM Node.....	15
6.	Deploy FireCracker Node.....	15
7.	Set up Passwordless SSH	17
8.	Network Configuration	19
a.	Configure Bridge Interface on KVM Node.	19
b.	Configure Bridge Interface on FireCracker Node.	20
9.	Add Host to Frontend.....	21
10.	Create Virtual Network.....	22
B.	DEMO.....	24
1.	Virtual Machine.....	24
2.	Docker Image	26
3.	Create VM with Terraform.....	28
C.	FUTURE DEVELOPMENT ORIENTATION	31
D.	REFERENCES.....	32

A. DETAILED REPORT

1. Learn about OpenNebula

a. Introducing OpenNebula



OpenNebula is an open-source cloud computing platform, developed by IMDEA Software Institute in Spain. Combining virtualization and container technology with multi-tenancy, automatically and flexibly deliver on-demand applications and services across private, hybrid and edge environments .

OpenNebula provides a unified, feature-rich, and flexible platform that centrally manages IT infrastructure and applications, prevents vendor lock-in, and reduces complexity, resource consumption and operating costs. OpenNebula supports deployments with a variety of virtualization types, including KVM, LXC, VMware, and Firecracker.

OpenNebula can manage:

- **Any Application:** Combine containerized applications from Kubernetes and Docker Hub ecosystems with Virtual Machine workloads in a common shared environment to offer the best of both worlds: mature virtualization technology and orchestration of application containers.
- **Any Infrastructure:** Unlock the power of a true hybrid, edge and multi-cloud platform by combining expanding your private cloud with infrastructure resources from third-party public cloud and bare-metal providers such as AWS and Packet (Equinix Metal).
- **Any Virtualization:** Integrate multiple types of virtualization technologies to meet your workload needs, including VMware and KVM virtual machines for

fully virtualized clouds, LXC system containers for container clouds, and Firecracker microVMs for serverless deployments.

- **Any Time:** Automatically add and remove new resources in order to meet peaks in demand, or to implement fault-tolerant strategies or latency requirements.

b. OpenNebula Model for Cloud Users

OpenNebula is designed to be flexible to help you adapt it to your actual needs. Below are some of the basic use cases and application models that OpenNebula supports.

Virtualized Applications:

- OpenNebula orchestrates Virtual Machines, but depending on the kind of workload you can use different types of hypervisors. OpenNebula can be deployed on top of your VMware vCentre infrastructure, but it can also manage KVM-based workloads as well as LXC system containers and lightweight Firecracker microVMs (especially convenient, for instance, to run application containers).
- OpenNebula provides multi-tenancy by design, offering different types of interfaces for users depending on their roles within your organization or the level of expertise or functionality required.
- OpenNebula can manage both single VMs and complex multi-tier services composed of several VMs that require sophisticated elasticity rules and dynamic adaptability.
- VM-based applications are created from images and templates that are available from the OpenNebula Public Marketplace but can also be created by the users themselves and shared by the cloud administrator using a private corporate marketplace.
- This model enables the quick instantiation of applications and complex services, including for instance the deployment of Kubernetes clusters at the edge.

Containerized Applications:

OpenNebula offers a new, native approach for running containerized applications and workflows by directly using the official Docker images available from the Docker Hub and running them as LXC system containers or as lightweight Firecracker microVMs, a method that provides an extra level of efficiency and security. This solution combines all the benefits of containers with the security, orchestration and multi-tenant features of a solid Cloud Management Platform but without adding extra layers of management, thus reducing the complexity and costs—compared with Kubernetes or OpenShift.

Cloud Access Model and Roles:

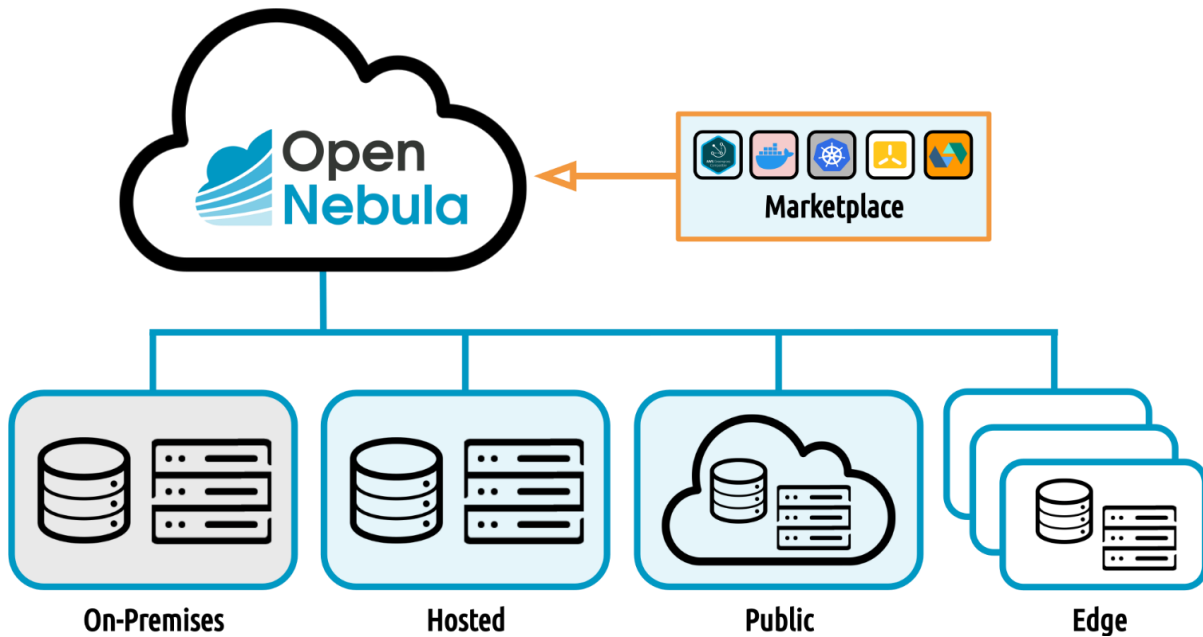
OpenNebula offers a flexible and powerful cloud provisioning model based on Virtual Data Centers (VDCs) that enables an integrated, comprehensive framework to dynamically provision the infrastructure resources in large multi-datacenter and multi-cloud environments to different customers, business units or groups. For example, the following are common enterprise use cases in large cloud computing deployments:

- **On-premise Private Clouds** serving multiple Projects, Departments, Units or Organizations: On-premise private clouds in large organizations require powerful and flexible mechanisms to manage the access privileges to the virtual and physical infrastructure and to dynamically allocate the available resources.
- **Cloud Providers** offering Virtual Private Cloud Computing: Cloud providers offering their customers a fully-configurable and isolated environment where they have full control and capacity to administer its users and resources. This combines a public cloud with the control usually seen in a personal private cloud system.

c. Architecture of OpenNebula.

The standard Cloud OpenNebula architecture includes:

- The **Cloud Management Cluster** with the Front-end node(s), and
- The **Cloud Infrastructure**, made of one or several workload **Clusters** with the hypervisor nodes and the storage system, which can be located at multiple geographical locations, all interconnected with multiple networks for internal storage and node management, and for private and public guest (VM or container) communication.



d. OpenNebula components

OpenNebula has been designed to be easily adapted to any infrastructure and easily extended with new components. The result is a modular system that can implement a variety of cloud architectures and can interface with multiple data center services.

Main components of the OpenNebula system:

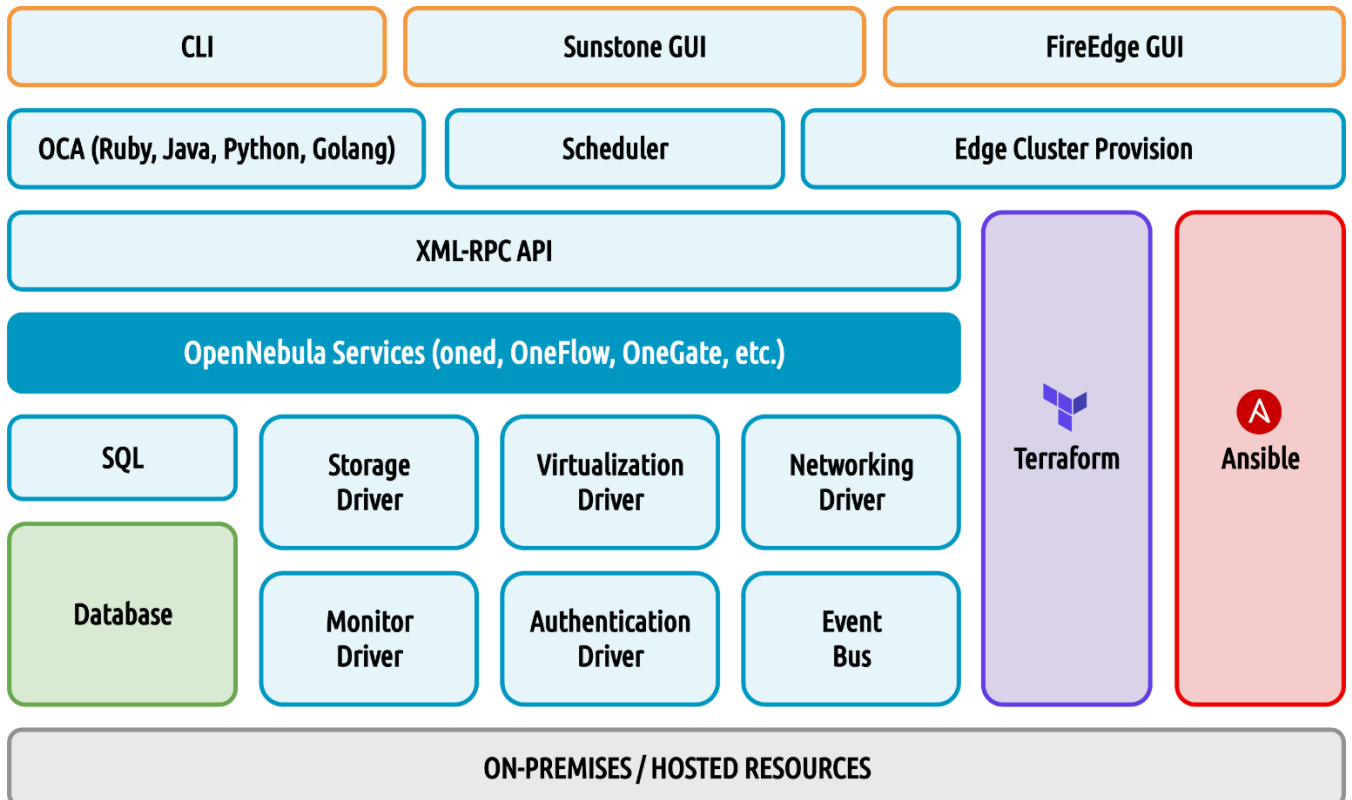
- **OpenNebula Daemon (*oned*):** The OpenNebula Daemon is the core service of the cloud management platform. It manages the cluster nodes, virtual networks and storages, groups, users and their virtual machines, and provides the XML-RPC API to other services and end-users.
- **Database:** OpenNebula persists the state of the cloud into the selected SQL database. This is a key component that should be monitored and tuned for the best performance by cloud administrators following the best practices of the particular database product.
- **Scheduler:** The OpenNebula Scheduler is responsible for the planning of the pending Virtual Machines on available hypervisor Nodes. It's a dedicated

daemon installed alongside the OpenNebula Daemon, but can be deployed independently on a different machine.

- **Edge Cluster Provision:** This component creates fully functional OpenNebula Clusters on public cloud or edge providers. The Provision module integrates Edge Clusters into your OpenNebula cloud by utilizing these three core technologies: Terraform, Ansible and the OpenNebula Services.
- **Monitoring:** The monitoring subsystem is represented by a dedicated daemon running as part of the OpenNebula Daemon. It gathers information relevant to the Hosts and the Virtual Machines, e.g. Host status, basic performance indicators, Virtual Machine status, and capacity consumption.
- **OneFlow:** The OneFlow orchestrates multi-VM services as a whole, defining dependencies and auto-scaling policies for the application components, interacts with the OpenNebula Daemon to manage the Virtual Machines (starts, stops), and can be controlled via the Sunstone GUI or over CLI. It's a dedicated daemon installed by default as part of the Single Front-end Installation, but can be deployed independently on a different machine.
- **OneGate:** The OneGate server allows Virtual Machines to pull and push information from/to OpenNebula, so users and administrators can use it to gather metrics, detect problems in their applications, and trigger OneFlow elasticity rules from inside the VMs. It can be used with all hypervisor Host types (KVM, LXC, Firecracker, and vCenter) if the guest operating system has preinstalled the OpenNebula contextualization package. It's a dedicated daemon installed by default as part of the Single Front-end Installation, but can be deployed independently on a different machine.
- **OneGate/Proxy:** The OneGate/Proxy service is a simple TCP proxy solution, that can be used to improve security of the OneGate's endpoint. Users can enable it on hypervisor Nodes, then it should become much easier to protect OneGate's traffic with a VPN solution, or at least, the requirement of exposing OneGate on a public IP in certain environments is no longer present.

OpenNebula system interfaces:

- **Sunstone:** OpenNebula comes with a Graphical User Interface (WebUI) intended for both end users and administrators to easily manage all OpenNebula resources and perform typical operations. It's a dedicated daemon installed by default as part of the Single Front-end Installation, but can be deployed independently on a different machine.
- **FireEdge:** The FireEdge server provides a next-generation Graphical User Interface (WebUI) for the provisioning of remote OpenNebula Clusters (leveraging the new OneProvision tool) as well as additional functionality to Sunstone.
- **CLI:** OpenNebula provides a significant set of commands to interact with the system and its different components via terminal.
- **XML-RPC API:** This is the primary interface for OpenNebula, through which you can control and manage any OpenNebula resource, including VMs, Virtual Networks, Images, Users, Hosts, and Clusters.
- **OpenNebula Cloud API:** The OCA provides a simplified and convenient way to interface with the OpenNebula core XML-RPC API, including support for Ruby, Java, Golang, and Python.
- **OpenNebula OneFlow API:** This is a RESTful service to create, control and monitor services composed of interconnected Virtual Machines with deployment dependencies between them.



2. Compare with another IssA Platform

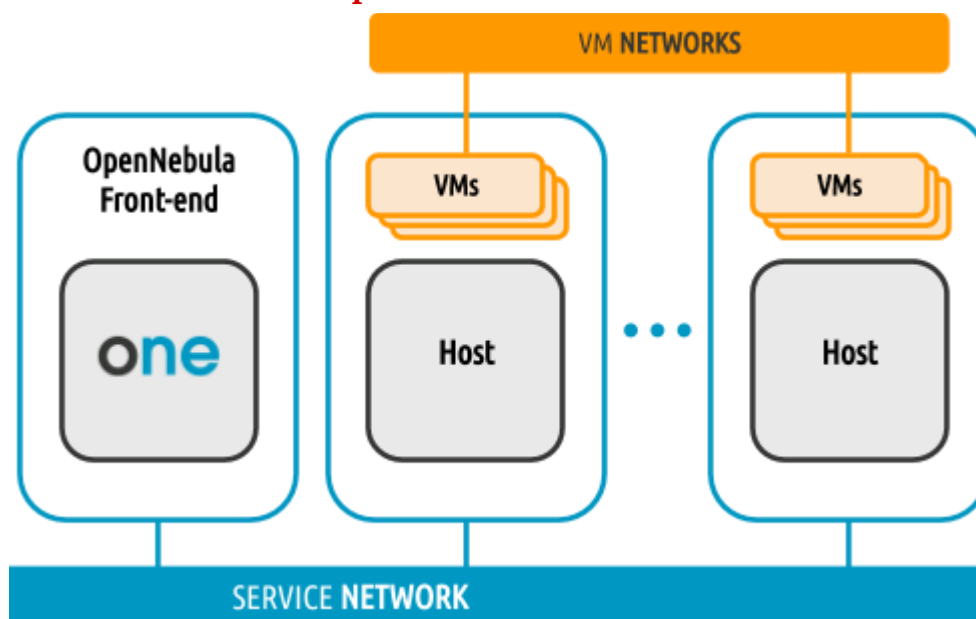
Feature	OpenNebula Pro	Apache CloudStack	OpenStack
Architecture	Modular: Flexible, component-based design for easier customization & scaling	Monolithic: All components tightly integrated, can be complex for large environments	Modular, but known for complex setup and configuration
Ease of Use	Simpler setup and administration, user-friendly web interface	Moderate learning curve, CLI and API access	Steep learning curve, requires significant expertise for deployment & management
Scalability	Scales horizontally, suitable for small to medium-sized clouds	Scales horizontally, designed for large-scale production environments	Highly scalable, ideal for large and complex cloud infrastructures
Container Orchestration	Native Kubernetes and Docker support	Requires additional integration with Kubernetes	Kubernetes integration through Magnum project
Virtualization	Supports a wide range of hypervisors (KVM, vSphere, Xen, etc.) & container runtimes	Primarily focuses on KVM	Supports various hypervisors (KVM, Hyper-V, VMware, etc.)
Networking	Software-defined networking (SDN) with Open vSwitch	Built-in virtual networking capabilities	Complex networking with Neutron, requires additional configuration
Cost	Open-source core, Pro version with enterprise features and support	Open-source, community-driven	Open-source, but enterprise support and distributions available from vendors

Hybrid Cloud	Strong focus on hybrid and multi-cloud environments	Limited support for multi-cloud	Extensive multi-cloud capabilities with various projects and integrations
Edge Computing	Lightweight, suitable for edge deployments	Less optimized for resource-constrained edge environments	Can be adapted for edge, but may require significant customization

OpenNebula Pro Advantages:

- **Modularity:** The flexible design makes it easier to tailor OpenNebula Pro to specific use cases and add/remove components as needed.
- **Kubernetes/Docker Integration:** Streamlined support for containerized workloads makes it well-suited for modern cloud-native applications.
- **Hybrid Cloud Focus:** Built-in features and integrations for managing hybrid cloud environments.
- **Simpler Administration:** Compared to OpenStack, OpenNebula Pro is generally easier to set up, manage, and troubleshoot.

3. System architecture to be implemented.



The system includes 3 virtual machines:

- 01 virtual machine as OpenNebula Front-end node.
- The remaining 02 virtual machines will be Hypervison Node including 2 Nodes using virtualization technologies KVM and Firecracker respectively.

Most Nodes will use Ubuntu 22.04.

Nodes use the same network.

Table of IP addresses of Nodes:

Node	IP address
Frontend	192.168.122.233
KVM	192.168.122.123
FireCracker	192.168.122.150

4. Deploying Front-end Node.

Access to Front-end Node.

- **Step 1:** Turn off SELINUX

SELinux may block some operations launched by the OpenNebula Front-end, resulting in the specific operation failing.

```
echo "SELINUX=disabled" >> /etc/selinux/config
```

- **Step 2:** Install the necessary packages for the Open Nebula Front-End installation

```
sudo apt update  
sudo apt -y install gnupg wget apt-transport-https
```

- **Step 3:** Log in to User root.

```
sudo su - root
```

- **Step 4:** Add OpenNebula Repository for installation

Add Repository GPG Key:

```
wget -q -O- https://downloads.opennebula.io/repo/repo2.key | gpg --  
dearmor --yes --output /etc/apt/keyrings/opennebula.gpg
```

Add Repository to the system

```
echo "deb [signed-by=/etc/apt/keyrings/opennebula.gpg]  
https://downloads.opennebula.io/repo/6.8/Ubuntu/20.04 stable opennebula"  
> /etc/apt/sources.list.d /opennebula.list
```

- **Step 5:** Front-End Installation.

Install all OpenNebula Front-end components by executing the following commands:

```
apt-get update  
apt-get -y install opennebula opennebula-sunstone opennebula-fireedge  
opennebula-gate opennebula-flow opennebula-provision
```

- **Step 6:** Install Docker.

Install Docker then add user **oneadmin** to the **docker group**.

```
usermod -a -G docker oneadmin
```

- **Step 7:** Configure OpenNebula

Log in to `oneadmin` with the command:

```
sudo -u oneadmin /bin/sh
```

Create file `/var/lib/one/.one/one_auth` with initial password in format `oneadmin:<password>`

```
echo 'oneadmin:changeme123' > /var/lib/one/.one/one_auth
```

This will set OpenNebula's admin password to **changeme123**, but will only set OpenNebula's password the first time it boots. If you want to change the password later, you must use the **oneuser passwd** command.

- **Step 8:** Configure FireEdge.

Edit the file `/etc/one/sunstone-server.conf` and change the parameter of **:public_fireedge_endpoint** to the access address to the Front-end node. For example here is 192.168.122.233 (Can be domain if Front-node is public).

```
:public_fireedge_endpoint: http://192.168.122.233:2616
```

- **Step 9:** Launch OpenNebula Services

Launch all OpenNebula services with the following command:

```
systemctl start opennebula opennebula-sunstone opennebula-fireedge  
opennebula-gate opennebula-flow
```

To automatically start these services on node startup, it is necessary to enable them with the following command:

```
systemctl enable opennebula opennebula-sunstone opennebula-fireedge  
opennebula-gate opennebula-flow
```

- **Step 10:** Verify the installation was successful

Run the following command as system user `oneadmin` and find similar results :

```
oneuser show
```

```
frontend@frontend:~$ sudo su - oneadmin
[sudo] password for frontend:
oneadmin@frontend:~$ oneuser show
USER 0 INFORMATION
-----
ID          : 0
NAME        : oneadmin
GROUP       : oneadmin
PASSWORD    : 704b89b07bfab51d67605065377ba5be28e29611ad94283c605bae715d6c0aa5
AUTH_DRIVER : core
ENABLED     : Yes

TOKENS

USER TEMPLATE
SSH_PUBLIC_KEY="ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBJIbLDWXT1tVlvYvI9SXfEUWUBqDzQcZt5gh3x9psrc"
TOKEN_PASSWORD="72170528df7464f7bfeecafa9db4ec571b581c2d28ac7996a35d51511eb6636"

VMS USAGE & QUOTAS

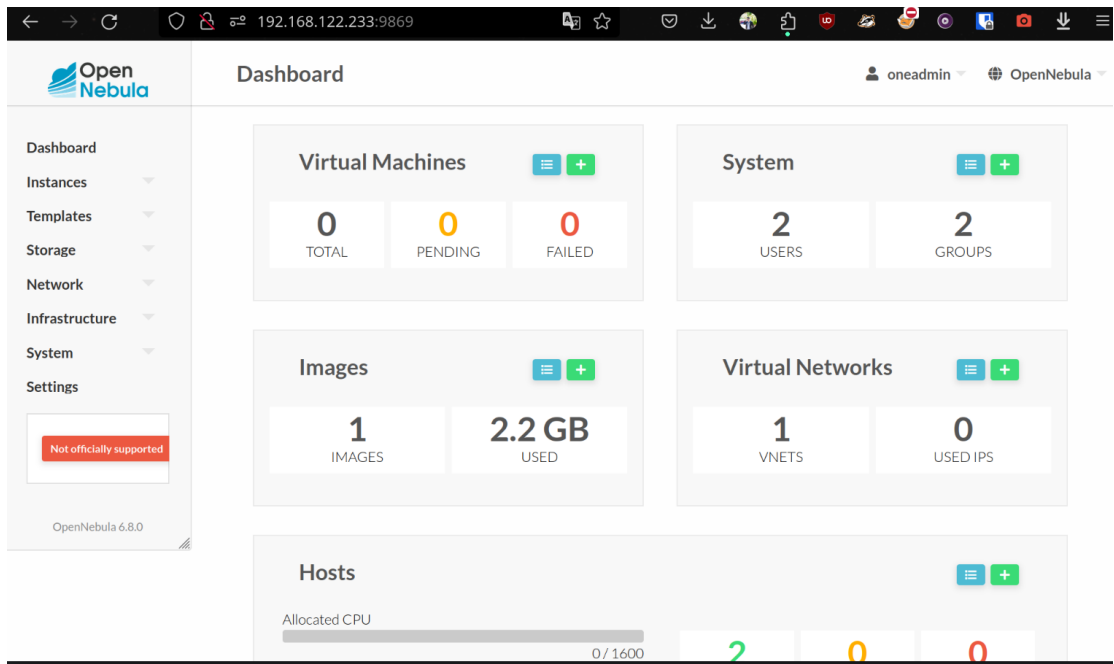
VMS USAGE & QUOTAS - RUNNING

DATASTORE USAGE & QUOTAS

NETWORK USAGE & QUOTAS

IMAGE USAGE & QUOTAS
```

Now you can try to log in through the Sunstone GUI using the url: <http://192.168.122.231:9869> (this is the address of the Front-end node). The login page will appear. The default username will be oneadmin and the password was set in the steps above.



5. Deploying KVM Node.

KVM helps run multiple Virtual Machines with Linux or Windows operating system images. Each VM will own its own virtualization hardware - including network cards, drives, graphics cards, etc...

Access KVM Node.

- **Step 1:** Add OpenNebula Repo, similar to Step 2,3,4 of Front-end deployment.
- **Step 2:** Install OpenNebula KVM Node on Ubuntu.

```
apt-get update  
apt-get -y install opennebula-node-kvm  
systemctl restart libvirtd
```

- **Step 3:** Change the password of user **oneadmin** .

It is necessary to set a password for the **oneadmin** user to set up SSH Passwordless in the following steps. To set a password, execute the following command.

```
sudo passwd oneadmin
```

6. Deploy FireCracker Node.

Firecracker uses Linux kernel-based virtual machine (KVM - Kernel-based Virtual Machine) technology to create and manage microVMs (tiny virtual machines). Firecracker is designed to be minimalist, eliminating unnecessary devices and functions to minimize the amount of memory used, as well as the attack surface of each microVM.

Access FireCracker Node.

- **Step 1:** Add OpenNebula Repo, similar to Step 2,3,4 of Front-end deployment.
- **Step 2:** Install OpenNebula FireCracker Node on Ubuntu.

```
apt-get update  
apt-get -y install opennebula-node-firecracker
```

- **Step 3:** Change the password of user **oneadmin** .

It is necessary to set a password for the **oneadmin** user to set up SSH Passwordless in the following steps. To set a password, execute the following command.

```
sudo passwd oneadmin
```

- **Step 4:** Network Connection for Firecracker MicroVM

Firecracker works with all OpenNebula network drivers. However, because it does not directly manage virtual tap devices (used in microVM network connections), OpenNebula will take care of managing and connecting these devices to the corresponding bridge .

To enable networking for microVM, the following commands need to be executed in the Frontend Node:

```
cp /var/lib/one/remotes/vnm/hooks/pre/firecracker  
/var/lib/one/remotes/vnm/Bridge/pre.d/firecracker  
cp /var/lib/one/remotes/vnm/hooks/clean/firecracker  
/var/lib/one/remotes/vnm/Bridge/clean.d/firecracker  
onehost sync -f
```

- **Step 5:** Turn off cgroup v2.

By default Ubuntu will be cgroup v2 however FireCracker has some problems with cgroup v2. So this feature is turned off.

Edit the file /etc/default/grub.

Adjust the parameter of GRUB_CMDLINE_LINUX_DEFAULT to the following.

```
GRUB_CMDLINE_LINUX_DEFAULT="systemd.unified_cgroup_hierarchy=0  
systemd.legacy_systemd_cgroup_controller"
```

Then run the Grub reconfigure command and reboot.

```
grub-mkconfig -o /boot/grub/grub.cfg
```


7. Set up Passwordless SSH

SSH connections: OpenNebula Front-end establishes SSH connections to:

- Itself (Front-end to Front-end)
- Hypervisor nodes
- Other Hypervisor nodes (during VM migration)
- Back to Front-end (to copy data)

Key Management:

The OpenNebula server automatically generates an SSH key pair for the user "oneadmin" (if it does not exist).

The public key needs to be distributed to the /var/lib/one/.ssh/authorized_keys directory on all Hypervisor nodes to establish a passwordless connection.

From version 5.12, OpenNebula uses a separate SSH authentication service on the Front-end through the SSH authentication agent service. This service manages **oneadmin's private key** and supports authorization of connections to the hypervisor node for necessary tasks. And there is no need to distribute private keys from Front-end to hypervisor node anymore.

Access the Front-end node.

- **Step 1:** Populate Host SSH Keys

Set up a list of SSH public keys of nodes (also called known_hosts) so that communicating parties can identify each other, increasing security.

Switch user permissions to **oneadmin** :

```
su - oneadmin
```

Create the **known_hosts** file with the **ssh-keyscan** command, passing in the front-end hostname and ip of the Hypervisor Node:

```
ssh-keyscan frontend 192.168.122.150 192.168.122.123 >>  
/var/lib/one/.ssh/known_hosts
```

Of which 2 IP addresses are the addresses of Hypervisor Node.

- **Step 2:** Distributed Authentication Configuration

Enables SSH connections between Front-end and nodes without entering a password, increasing convenience and security.

Copy oneadmin's public key to each node:

```
ssh-copy-id -i /var/lib/one/.ssh/id_rsa.pub 192.168.122.123  
ssh-copy-id -i /var/lib/one/.ssh/id_rsa.pub 192.168.122.150
```

Copy the known_hosts file to each node:

```
scp -p /var/lib/one/.ssh/known_hosts 192.168.122.123:/var/lib/one/.ssh/  
scp -p /var/lib/one/.ssh/known_hosts 192.168.122.150:/var/lib/one/.ssh/
```

8. Network Configuration .

OpenNebula Front-end needs network connection to Hosts to manage, monitor, and transmit images. Creating a Bridge Interface is the simplest way to do the above. There are also other ways 802.1Q VLAN, VXLAN, Open vSwitch and Open vSwitch on VXLAN.

a. Configure Bridge Interface on KVM Node.

Bridge Interface is created and configured through systemd-networkd.

- **Step 1:** Create Bridge Interface.

Create a configuration file at /etc/systemd/network/10-br0.netdev

```
[NetDev]
Name=br0
Kind=bridge
```

- **Step 2:** Add Ethernet Interface to Bridge Interface.

Create configuration file at /etc/systemd/network/10-br0-en.network

```
[Match]
Name=en*

[Network]
Bridge=br0
```

- **Step 3:** Bridge Network configuration.

Create configuration file at /etc/systemd/network/10-br0.network.

```
[Match]
Name=br0

[Network]
DNS=192.168.122.1
Address=192.168.122.123/24
Gateway=192.168.122.1
```

Where Address= is the IP address of the Ethernet Interface. Gateway is the Default Gateway of the Ethernet Interface.

- **Step 4:** Restart the systemd-networkd service.

```
sudo systemctl restart systemd-networkd
```

After performing the above steps, interface **br0** and Ethernet interface have been added to the Bridge Interface.

```

node1@node1:~$ ip a
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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UP group default qlen 1000
    link/ether 52:54:00:03:7b:dc brd ff:ff:ff:ff:ff:ff
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ea:f3:69:19:ef:90 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.123/24 brd 192.168.122.255 scope global br0
        valid_lft forever preferred_lft forever
    inet6 fe80::e8f3:69ff:fe19:ef90/64 scope link
        valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:44:aa:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.123.1/24 brd 192.168.123.255 scope global virbr0
        valid_lft forever preferred_lft forever
node1@node1:~$ bridge link
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state forwarding priority 32 cost 100

```

b. Configure Bridge Interface on FireCracker Node.

The steps are similar to those on KVM Node. After execution, the results will be as follows.

```

node2@node2:~$ ip a
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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UP group default qlen 1000
    link/ether 52:54:00:55:38:1a brd ff:ff:ff:ff:ff:ff
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether fa:da:8d:b8:32:37 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.150/24 brd 192.168.122.255 scope global br0
        valid_lft forever preferred_lft forever
    inet6 fe80::f8da:8dff:feb8:3237/64 scope link
        valid_lft forever preferred_lft forever
node2@node2:~$ bridge link
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state forwarding priority 32 cost 100

```

9. Add Host to Frontend.

Set the hostname for Hypervisor Nodes instead of using IP by adding the following lines to the Frontend Node's /etc/hosts file.

```
192.168.122.123 kvm
192.168.122.150 firecracker
```

Add Hypervisor Node to Host, with following command.

```
onehost create kvm -i kvm -v kvm
onehost create firecracker -i kvm -v kvm
```

After successfully adding, Status will change to ON.

```
frontend@frontend:~$ sudo onehost list
```

ID	NAME	CLUSTER	TVM	ALLOCATED_CPU	ALLOCATED_MEM	STAT
2	kvm	default	1	100 / 800 (12%)	768M / 1.8G (41%)	on
1	firecracker	default	1	100 / 800 (12%)	768M / 1.8G (41%)	on

10. Create Virtual Network.

Name the Virtual Network

The screenshot shows the 'General' tab of the OpenNebula Virtual Network creation wizard. At the top, there are buttons for navigation: a back arrow, 'Reset', and 'Create'. Below these are tabs for 'General', 'Conf', 'Addresses', 'Security', 'QoS', and 'Context'. The 'General' tab is active. It contains a 'Name' field with the text 'network', a 'Cluster' dropdown menu set to '0: default', and a 'Description' text area.

Bridge is the name of the Bridge Interface created in the above step. Network mode is set to Bridged.

The screenshot shows the 'Conf' tab of the OpenNebula Virtual Network configuration wizard. It features the same navigation tabs as the previous screen. The 'Conf' tab is active. It contains a 'Bridge' field with the text 'br0', a 'Network mode' dropdown menu set to 'Bridged', and a 'Physical device' field. Below the 'Network mode' dropdown, there is a note: 'Bridged, virtual machine traffic is directly bridged. The Linux bridge is created in the nodes as needed. No traffic filtering is made.'

First IPv4 Address is the first IP address of the Virtual Network. Size is the size of Virtual Network.

The screenshot shows the 'Addresses' tab of the OpenNebula Virtual Network configuration wizard. It features the same navigation tabs. The 'Addresses' tab is active. On the left, there is a sidebar with a button labeled 'AR' and a '+' icon. The main area contains radio buttons for 'IPv4', 'IPv4/6', 'IPv6', and 'Ethernet', with 'IPv4' selected. Below these are fields for 'First IPv4 address' (containing '192.168.122.10'), 'First MAC address' (empty), and 'Size' (a dropdown menu set to '10'). At the bottom, there is a link for 'Advanced Options'.

In the Context section, this will be the network configuration of VMs created on OpenNebula when using this Virtual Network.

General

Conf

Addresses

Security

QoS

Context

Network address

192.168.122.0

Network mask

255.255.255.0

Gateway

192.168.122.1

IPv6 Gateway

DNS

192.168.122.1

MTU of the Guest interfaces

Method

static (Based on context) ▼

IPv6 Method

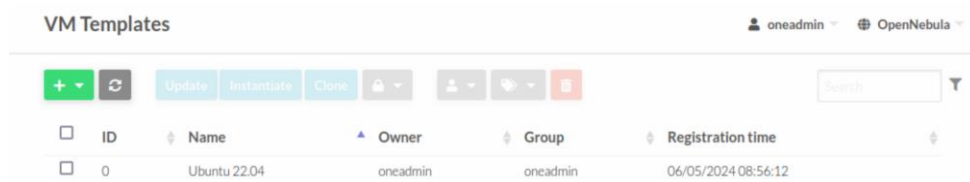
none (Use default) ▼

Custom attributes

B. DEMO

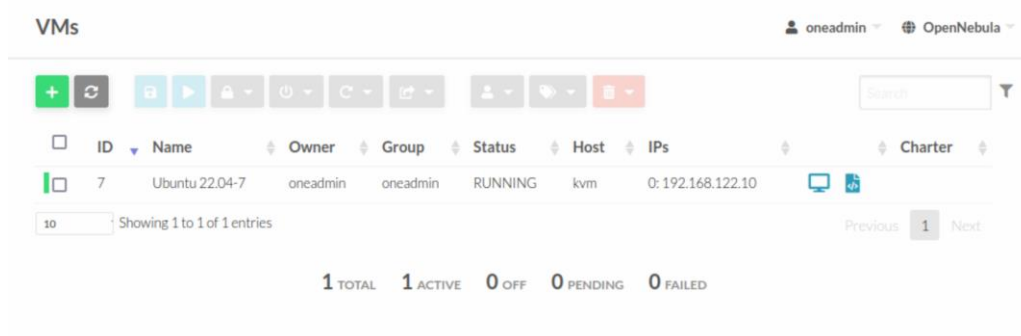
1. Virtual Machine.

Ubuntu Templates are imported from Marketplace.



ID	Name	Owner	Group	Registration time
0	Ubuntu 22.04	onecadmin	onecadmin	06/05/2024 08:56:12

Created virtual machine successfully.

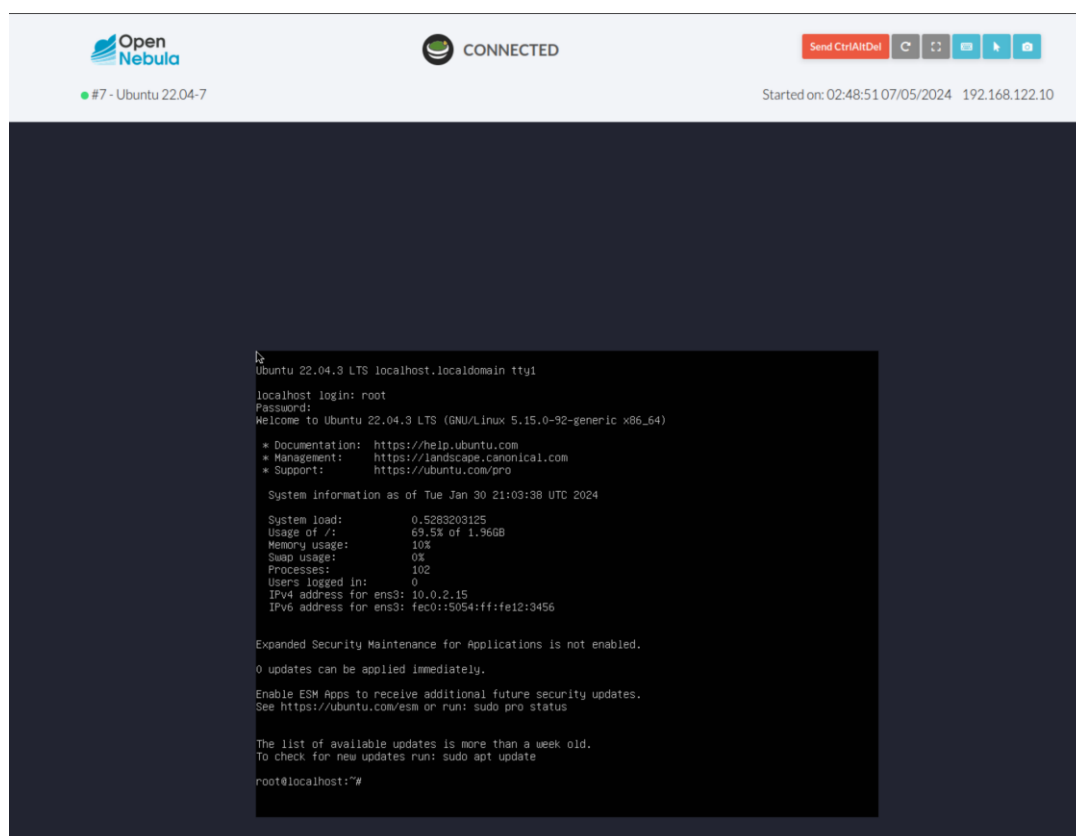


ID	Name	Owner	Group	Status	Host	IPs	Charter
7	Ubuntu 22.04-7	onecadmin	onecadmin	RUNNING	kvm	0: 192.168.122.10	

Showing 1 to 1 of 1 entries

1 TOTAL 1 ACTIVE 0 OFF 0 PENDING 0 FAILED

Can VNC to the newly created VM.



```
Open Nebula
#7 - Ubuntu 22.04-7
Started on: 02:48:51 07/05/2024 192.168.122.10

Ubuntu 22.04.3 LTS localhost.localdomain tty1
localhost login: root
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-92-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Tue Jan 30 21:03:38 UTC 2024
System load:      0.5283203125
Usage of /:        69.5% of 1.96GB
Memory usage:     10%
Swap usage:       0%
Processes:        102
Users logged in:   0
IPv4 address for ens3: 10.0.2.15
IPv6 address for ens3: fec01:5054:ff:fe12:3456

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

root@localhost:~#
```


It is also possible to ssh to the newly created VM via private key.

```
> ssh root@192.168.122.10
The authenticity of host '192.168.122.10 (192.168.122.10)' can't be established.
ED25519 key fingerprint is SHA256:Co9TSWi7ZHo9l0CJjq1uer105PQdiR+T6pa7SJTPWrY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.122.10' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/tuilakhanh/.ssh/id_ed25519':
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-92-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Mon May  6 19:51:07 UTC 2024

System load:  0.2197265625      Processes:           93
Usage of /:   58.6% of 1.96GB   Users logged in:    1
Memory usage: 23%              IPv4 address for eth0: 192.168.122.10
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

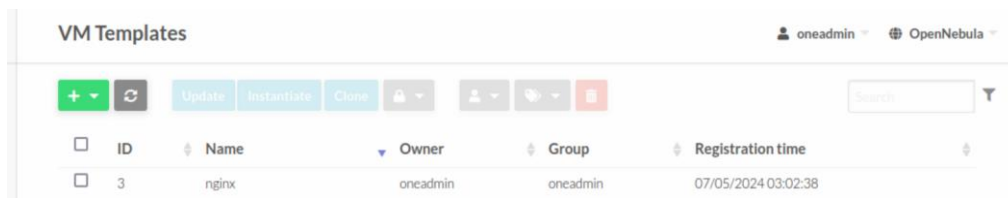
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon May  6 19:49:29 2024
root@localhost:~#
```

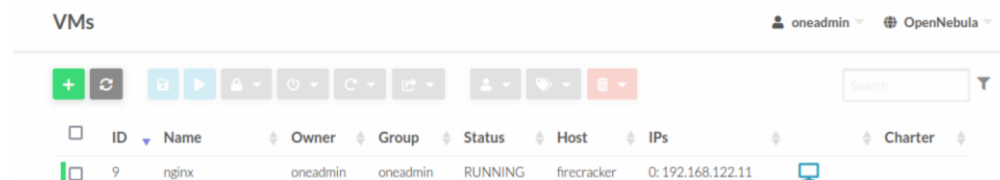
2. Docker Image

Nginx docker image is obtained from DockerHub via MarketPlace.



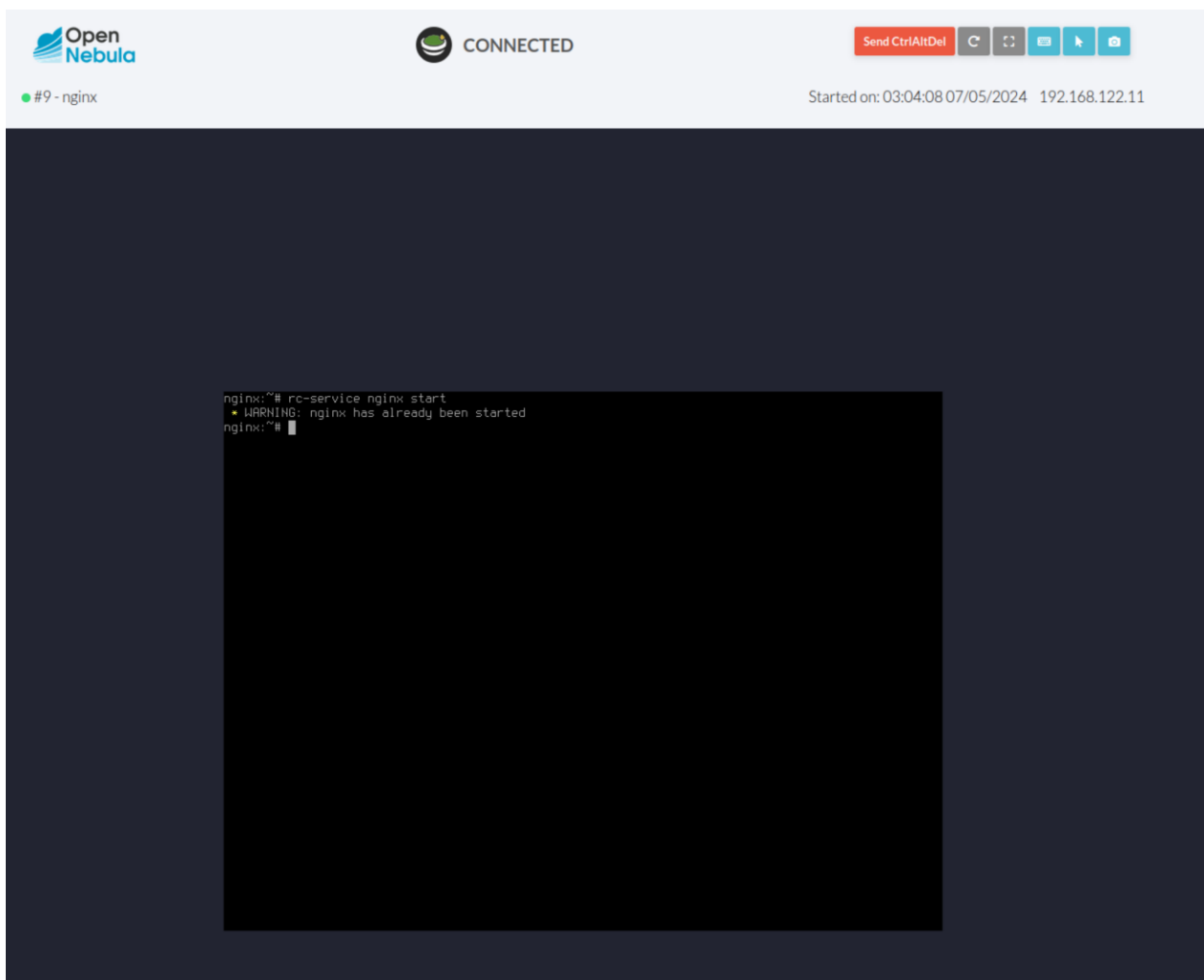
ID	Name	Owner	Group	Registration time
3	nginx	oneadmin	oneadmin	07/05/2024 03:02:38

Create VMs from the previous image.

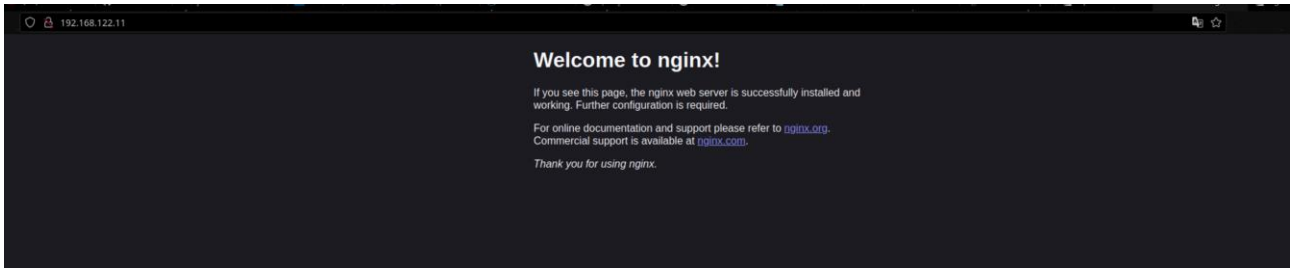


ID	Name	Owner	Group	Status	Host	IPs	Charter
9	nginx	oneadmin	oneadmin	RUNNING	firecracker	0: 192.168.122.11	

Can VNC to the newly created VM.



You can see nginx is up and running and can be accessed.



3. Create VM with Terraform

main.tf

```
terraform {
  required_providers {
    opennebula = {
      source = "OpenNebula/opennebula"
      version = "~> 1.4"
    }
  }
}

provider "opennebula" {
  endpoint      = "http://192.168.100.188:2633/RPC2"
  username      = var.opennebula_username
  password      = var.opennebula_token
  insecure      = true
}
```

variables.tf

```
variable "opennebula_username" {
  default = "oneadmin"
}

variable "opennebula_group" {
  default = "oneadmin"
}

variable "opennebula_token" {
  default = "07092003"
}

variable "opennebula_network_id" {
  default = "0"
}

variable "opennebula_template_id" {
  default = "0" # Ubuntu 22.04
}

variable "opennebula_image_id" {
  default = "0" # Ubuntu 22.04
}
```

host.tf

```
resource "opennebula_host" "node0" {  
  name      = "kvm"  
  type      = "kvm"  
}
```

vm.tf

```
resource "opennebula_virtual_machine" "node0" {  
  
  template_id = var.opennebula_template_id  
  
  name = "ubuntu-22-04"  
  
  cpu      = 0.5  
  memory   = 1024  
  group    = var.opennebula_group  
  
  nic {  
    model      = "virtio"  
    network_id = var.opennebula_network_id  
  }  
  
  disk {  
    image_id = var.opennebula_image_id  
    target   = "vda"  
    size     = 3096  
    driver   = "qcow2"  
  }  
}
```

Run Terraform with the following command.

```
terraform init  
terraform apply
```

Can see that apply successful.

```
Plan: 1 to add, 0 to change, 2 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

opennebula_host.node1: Destroying... [id=4]
opennebula_virtual_machine.node0: Destroying... [id=22]
opennebula_host.node1: Still destroying... [id=4, 10s elapsed]
opennebula_virtual_machine.node0: Still destroying... [id=22, 10s elapsed]
opennebula_host.node1: Destruction complete after 10s
opennebula_virtual_machine.node0: Destruction complete after 15s
opennebula_virtual_machine.node0: Creating...
opennebula_virtual_machine.node0: Still creating... [10s elapsed]
opennebula_virtual_machine.node0: Creation complete after 10s [id=23]

Apply complete! Resources: 1 added, 0 changed, 2 destroyed.
```

Check WebUI, can see that VM created successful.

VMs

oneadmin OpenNebula

Search

ID	Name	Owner	Group	Status	Host	IPs	Charter
23	ubuntu-22-04	oneadmin	oneadmin	SHUTDOWN	kvm	0: 192.168.100.10	

Showing 1 to 1 of 1 entries

Previous 1 Next

1 TOTAL 1 ACTIVE 0 OFF 0 PENDING 0 FAILED

C. FUTURE DEVELOPMENT ORIENTATION .

- Test system deployment on physical server.
- Use NAS or separate storage system as DataStore.
- Use MYSQL, MongoDB... instead of SQLite.
- Deploying Ansible automates the system build process.
- Deploy focuses more on system availability and the ability to scale larger later.
- Deploy Public/Private network infrastructure.

D. REFERENCES

<https://docs.opennebula.io/>

<https://wiki.archlinux.org/>

<https://help.ubuntu.com/community/man>