Resource and Service Orchestration for deploying OpenStack Cloud using MAAS and Juju

[1], Mtshali Mxolisi, [1] Lawrence Mboweni, [1] Hlabishi Kobo

[1]Council for Scientific and Industrial Research (CSIR), PO Box 395, Pretoria 0001, South Africa

Mmtshali1@csir.co.za

Abstract—**there is a huge demand for cloud computing platforms as they are a key enabler of digitalization. OpenStack is one of the prevalent cloud computing platforms in the industry. It is an open-source project which is gaining a lot of traction known for handing the core cloud-computing services of networking, storage, identity, and image services. There are several other open-source cloud technologies which are also available for private cloud implementations. However, we chose OpenStack due to its popularity, flexibility, and stability. The current manually based installation of OpenStack is very complex, tedious, and time-consuming, and therefore not ideal for a fast-paced environment such as high demand data intensity ecosystems, which are at the core of the current digital transformation. The primary objective for a cloud administrator is to commission and deploy OpenStack as fast as possible. Thus, in this paper, we used MAAS (Metal as a Service) and Juju to orchestrate the deployment of OpenStack cloud services. In the process of deploying OpenStack, the MAAS Ubuntu server cluster is first deployed on a private network, where machine Cores, RAM, Storage, and Disk are dynamically added using pre-defined power types. In addition, the Juju environment is set up, and Juju bootstrapping for the assignment of OpenStack services is completed. Step by step, five machines enlist on MAAS and display their resources until they are successfully deployed, with each machine listing the number of Cores, RAM, Storage, Disk, and so on. As a result, the OpenStack services are assigned to deployed machines, allowing for easy monitoring and control of resource and service orchestration.**

***Keywords*— Insert key words**

I. INTRODUCTION

Cloud computing is a concept for providing on-demand network access to a shared pool of computing resources such as processing, storage, and bandwidth that can be instantly supplied, configured, and released with minimal effort and service provider contact. The primary goal of cloud computing is to provide services and resources in a more timely and cost-effective manner. On-demand self-service, extensive network access, resource pooling, and quick elasticity are all characteristics of the cloud computing model. Fig. 1 shows three cloud-based service models including Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [1].

In Fig. 1, IaaS is concerned with providing consumers with hardware resources (processing, storage, network and bandwidth) as a service over the internet. PaaS is when providers host development tools on their infrastructure and allow customers to use Application Programming Interfaces (APIs) to access these tools over the internet. The operating system, storage, and applications deployed are all at the control of the customer, but not the underlying cloud infrastructure. SaaS is when PaaS is abstracted with software. A cloud platform is required to supply and implement cloud-based infrastructure services.

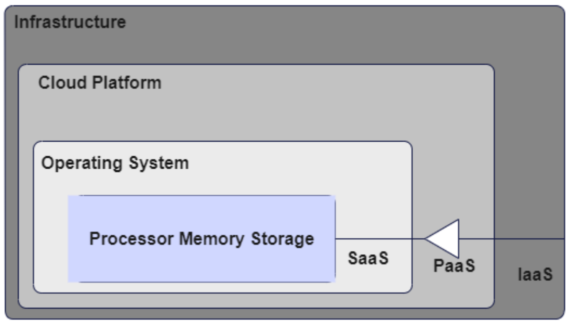


Fig. 1: IaaS Architecture in the Cloud.

For the cloud to be accessible to consumers it needs to be deployed and made available. Cloud computing deployments can be private, public, and hybrid [3], [4]. In private cloud deployment, the infrastructure resides and can only be utilised within a single organisation. In public clouds, infrastructure is located on the grounds of the service provider and is available for use by clients from any organisation. Hybrid clouds are made up of two or more different cloud infrastructures public and private. Once deployments are completed, computing resources such as processing cores, random access memory (RAM), storage, disk, and so on are orchestrated in a cluster and dynamically allocated to nodes based on their resource. To avoid IP addresses conflict when allocating resources, assign unique IP addresses within the same IP subnet. These resources are enlisted in the cluster and sent on to the network [5], [6].

The resource orchestrating mechanism for large cloud providers like Amazon, Google and Microsoft Azure is equipped with a cost-effective optimization strategy that reduces costs while increasing utilization. It also keeps track of underutilized computational resources in a local cache. They have their own strategies for resource distribution and tariff costs. However, they are not open source. On the other hand, there are several open-source cloud platforms available such as OpenStack, Apache Mesos, Eucalyptus, Open-Nebula, and CloudStack that can be adopted [7].We chose OpenStack for our research work since it is the most adopted open source platform by various organisations. For resource orchestration, OpenStack Autopilot, Fuel OpenSatck, Ansible OpenStack are some of the generic automated OpenStack tools that had been adopted to move away from manual complex deployment of OpenStack [8]–[10]. However, for this research work

OpenStack with Metal as a Service (MAAS) and Juju is adopted. Major reasons are, it is stable in deployment, allows multi-node, can be used in production and it forms the basis towards network slicing in the 5G ecosystem with its service orchestration capability. This paper's contribution is to show how resources are orchestrated using MAAS server cluster using the OpenStack tools MAAS and Juju, and how these resources are assigned to different machines so that specific OpenStack services can be provided.

The remainder of the paper is structured as follows: Section 2 provides background information on resource orchestration. The third section introduces MAAS (Metal as a Service), Juju, and OpenStack services. Section 4 shows the MAAS Ubuntu-server cluster's orchestration setup and approach, demonstrating the orchestration of computing resources and the allocation of OpenStack services to MAAS Ubuntu-server workstations. The results are discussed in Section 5. Finally, Section 6 brings the paper to a close.

II. RESOURCE ORCHESTRATION

The concept of cloud orchestration includes complex activities such as resource selection, deployment, monitoring, and run-time control are covered by cloud resource orchestration. The main purpose of orchestration is to ensure comprehensive and smooth application delivery by achieving both cloud application owners' and cloud resource providers' Quality of Service (QoS) goals. Because of the size at which resources have grown and the proliferation of heterogeneous cloud providers delivering resources at various levels of the cloud stack, resource orchestration is regarded as a difficult task [11]–[15]. In the context of this paper the resource orchestration is produced over a private network where computer resources are moved to a private cloud deployment. This is accomplished by having a subnet of private IP addresses. When a system starts up, it automatically recognizes other machines' IP addresses and gathers their computer resources, MAC addresses are listed with a unique machine name. The machines for which the resources belong to are first identified, and then they are commissioned and deployed. As many machines as possible can be added in the cluster.

III. OPENSTACK DEPLOYMENT TOOLS

A. *Metal As A Service*

Metal as a Service (MAAS) is a set of software developed by Canonical to instantly commission the bare-metal servers into manageable components for services such as OpenStack. Region controller, Rack/Cluster controller(s), and Target nodes (physical servers) are the main components of a typical MAAS setup. A region controller is linked to one or more cluster controllers, each of which oversees contacting the region controller and notifying it of its existence [16]. The default cluster controller is the region controller as shown in Fig. 2.

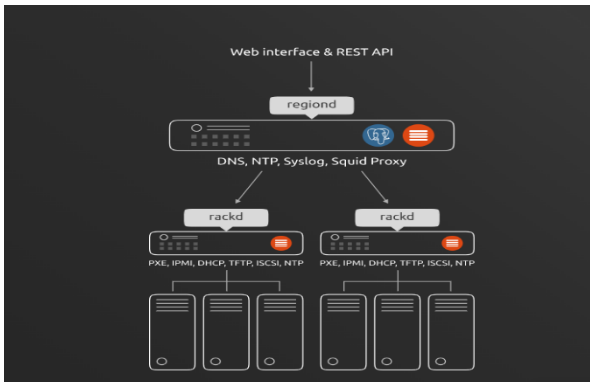

Web interface & REST API
regiond
DNS, NTP, Syslog, Squid Proxy
rackd
rackd
PXE, IPMI, DHCP, TFTP, ISCSI, NTP
PXE, IPMI, DHCP, TFTP, ISCSI, NTP

Fig. 2: MAAS Architecture.

Protocols such as DHCP, IPMI, PXE, TFTP, and other local services are provided by Rack Controller as outlined (rackd) in Fig. 2. For performance, they cache large items like operating system install images at the rack level, but they don't keep any exclusive state aside from credentials to communicate with the controller. MAAS makes it simple to deploy many devices at once and treat them as a group rather than as individuals. MAAS has a user-friendly online interface as well as a command-line interface. The actual machine can be installed, deployed, updated, recycled, and maintained using a simple web interface. Provisioning a server in MAAS entails three steps: enlistment, commissioning, and deployment. After registering with the MAAS server using PXE-based discovery, new systems are added to the pool of servers. MAAS has power over the nodes once they've been discovered. Preparing a newly discovered node for service is part of commissioning it. It entails running a script and acquiring information such as cores, memory, storage, and network interface cards (NICs), as well as performing storage partitioning and mounting partitions. Each MAAS-managed machine ("node") goes through a lifecycle, from enlistment to commissioning, where firmware and other hardware-specific elements are configured then allocation to a user and deployment, and finally release back to the pool or retirement. The lifecycle follows the state diagram shown in Fig. 3.

Fig. 3: State diagram for node lifecycle.

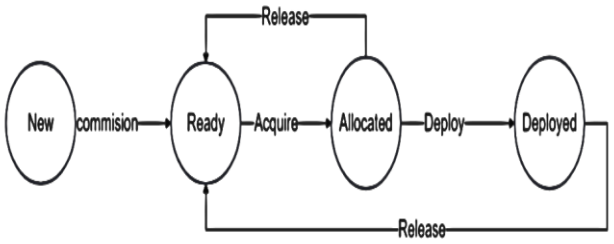


Fig. 3: State diagram for node lifecycle.

In Fig. 3 , a node often moves through the states of NEW, READY, ALLOCATED, and DEPLOYED, as well as BROKEN, RETIRED, and MISSING. In the event of a requirement, any node can be destroyed and redeployed following the same procedure.

B. *Juju*

Juju is a piece of software that controls your software. It enables you to take command of your entire application, infrastructure, and environment. It makes it simple and quick to manage, configure, deploy, maintain, and scale the cloud services. Juju's services are also available for use on public clouds, private clouds, and physical servers. OpenStack, as well as cloud applications can use Juju charms, a yaml-based configuration file. On Ubuntu versions, Juju facilitates the installation of OpenStack components. The Juju bootstrap node is one of the most important components in Juju's deployment and control of other services. Juju requires the end user to write a yaml configuration file for each OpenStack service component. These configuration files are then used to install various OpenStack services [17]. Fig. 4 shows the operation of Juju. During bootstrap, a server with the specified constraints is provisioned so that MAAS can interoperate with Juju using an installed Juju agent.

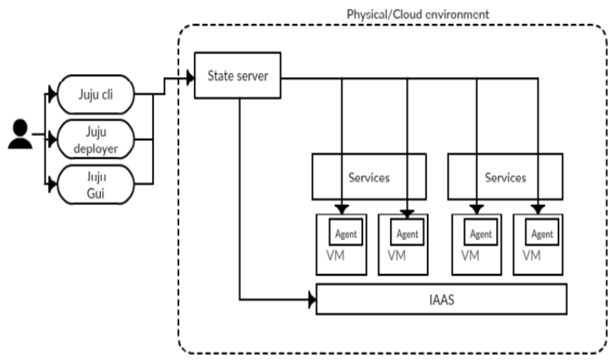


Fig. 4: Juju Operation

C. *OpenStack Components*

OpenStack is a cloud operating system that manages and provisions huge pools of compute, storage, and networking resources across a datacentre via APIs and common authentication mechanisms. A dashboard is also offered, which allows administrators to maintain control while empowering users to provision resources via a web interface. Additional components, in addition to typical infrastructure-as-a-service capability, include orchestration, fault management, and service management, among other services, to ensure that user applications are always available. OpenStack is a cloud operating system that manages and provisions huge pools of compute, storage, and networking resources across a datacentre via APIs and common authentication mechanisms. A dashboard is also offered, which allows administrators to maintain control while empowering users to provision resources via a web interface. Additional components, in addition to typical infrastructure-as-a-service capability, include orchestration, fault management, and service management, among other services, to ensure that user applications are always available [18]. The key services critical for OpenStack deployment are as follows:

- **Keystone** – OpenStack Identity service (Keystone) provides Authorization and Authentication for users and manage service catalogs.
- **Glance** - OpenStack Image Storage Service (Glance) stores and manages virtual machine images in different format.
- **Neutron** – OpenStack Network Service (Neutron) enables network connectivity to interface devices. It enables users to create and attach interface to networks.
- **Cinder** – OpenStack Block Storage Service (Cinder) provides block storage to guest virtual machines for expanded storage and better performance.
- **Swift** - OpenStack Object Storage Service (Swift) provides a cost-effective, scalable storage platform used for backup.
- **Ceilometer** – OpenStack Metering/Monitoring Service (Ceilometer) is used for billing, benchmarking and for gathering statistics

IV. RESOURCE ORCHESTRATION APPROACH AND SETUP

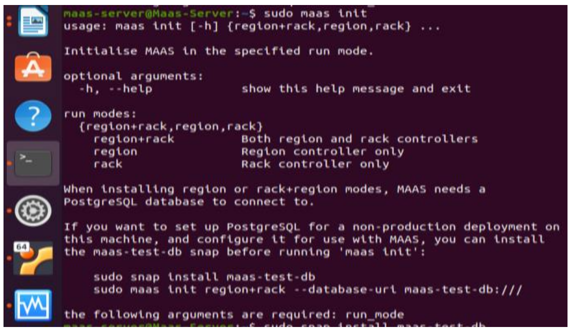
The resource orchestration configuration and setup consist of three physical machines. The MAAS machine consists of an Intel-Core i7-4790K CPU@ 4.00GHz x 8 consisting of 23GiB RAM, 500GiB Hard-disk. The second machine is the Juju Controller machine which is an Intel-Core i7-4790 CPU@ 3.6 GHz x 8 consisting of 16GiB and 500.1 GB Hard-disk. The last machine which is hosting 4 nodes inside the VirtualBox is an Intel-Core i7-8700 CPU@3.20GHz x 12 consisting of 39GiB RAM and 500 GiB hard disk. All the nodes must be assigned a private IP address from the MAAS subnet which has DHCP, and DNS management enabled. There musts be a certain range of private floating IPs reserved for allocation within the subnet. The software versions used for installation are as follows:

- Ubuntu 20.04 LTS (Focal) was used as an OS for the MAAS server, Juju controller, Juju client, and all cloud nodes including containers.
- MAAS 3.0.0
- Juju 2.9.15
- OpenStack Xena

A MAAS machine installation

To install MAAS and requirements of post-installations tasks with the goal to achieve a network topology that is both simple yet scalable, we have presented the following command line interfaces (CLI) commands from Fig 5 to Fig 6.

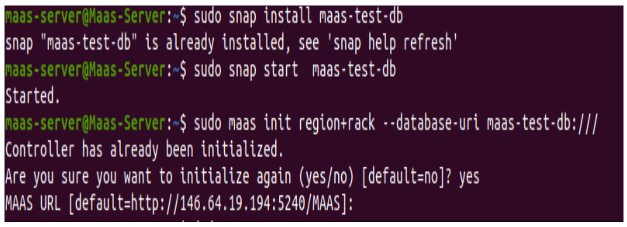
```
$ sudo apt update
$ sudo apt upgrade
$ sudo snap install maas
$ sudo maas init
```



```
maas-server@Maas-Server:~$ sudo maas init
usage: maas init [-h] {region+rack,region,rack} ...

Initialise MAAS in the specified run mode.

optional arguments:
  -h, --help            show this help message and exit

run modes:
  {region+rack,region,rack}
    region+rack         Both region and rack controllers
    region              Region controller only
    rack                Rack controller only

When installing region or rack+region modes, MAAS needs a
PostgreSQL database to connect to.

If you want to set up PostgreSQL for a non-production deployment on
this machine, and configure it for use with MAAS, you can install
the maas-test-db snap before running 'maas init':

    sudo snap install maas-test-db
    sudo maas init region+rack --database-uri maas-test-db:///

the following arguments are required: run_mode
```

Fig. 7: MAAS init cli.

```
$ sudo snap install maas-test-db
$ sudo snap start maas-test-db
$ sudo maas init region+rack --database-uri maas-test-db:///
```

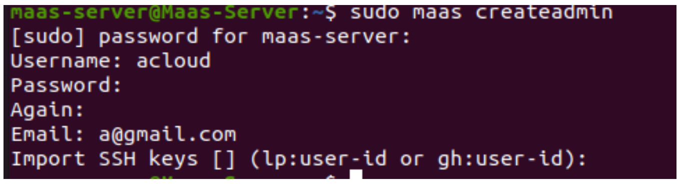


```
maas-server@Maas-Server:~$ sudo snap install maas-test-db
snap "maas-test-db" is already installed, see 'snap help refresh'
maas-server@Maas-Server:~$ sudo snap start  maas-test-db
Started.
maas-server@Maas-Server:~$ sudo maas init region+rack --database-uri maas-test-db:///
Controller has already been initialized.
Are you sure you want to initialize again (yes/no) [default=no]? yes
MAAS URL [default=http://146.64.19.194:5240/MAAS]:
```

Fig. 8: Identified MAAS IP

The IP address given by default on Fig. 8 is for the MAAS system and can be accepted as is or can assign your preferred IP address. This ensures that the cluster controller agent is pointed at the MAAS master controller's right address. The network interface on the cluster controller is then automatically recognized by MAAS. After that, we change the cluster interface and choose between DHCP and DNS. Finally, subnet mask, broadcast IP, router IP, and dynamic range of IP are configured so that IP is assigned dynamically to machines during PXE boot.

```
$ sudo maas createadmin
```



```
maas-server@Maas-Server:~$ sudo maas createadmin
[sudo] password for maas-server:
Username: acloud
Password:
Again:
Email: a@gmail.com
Import SSH keys [] (lp:user-id or gh:user-id):
```

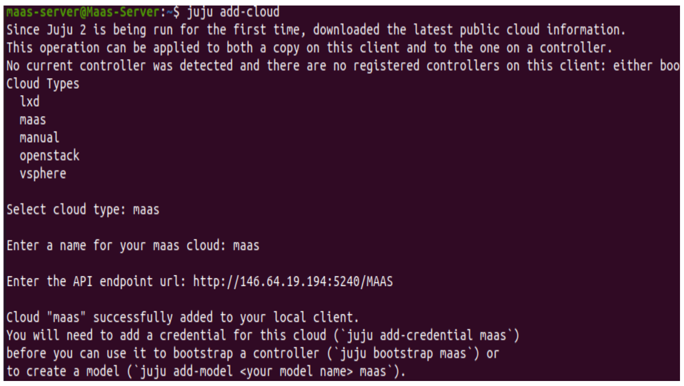
Fig. 9: Creating user admin credentials

As illustrated in Fig. 9, we can log in to the MAAS server using the username and password. Import boot images for Ubuntu 20.04 LTS after logging in to the MAAS server. A cluster controller has boot images by default, whereas a region controller does not. As a result, no machines will be provided until the boot images have been imported into the region controller. MAAS cluster controller should be installed. Once boot images have been imported, the machines can now boot from PXE image which is Network booting, often known as Netboot. Netboot is a technology that allows a server to be provisioned from another server on the network rather than a local hard disk. The enlisting process follows the state diagram from Fig. 3.

B. Juju Machine installation

Once the base environment set up MAAS cluster is up and running. Juju is implemented as a management solution for that environment. Juju goal is to create a controller that oversees the Juju managed clouds. The cli commands for Juju install are as follows:

```
$ sudo apt update
$ sudo snap install Juju --classic
$ Juju add-cloud
$ Juju add-credential maas
```

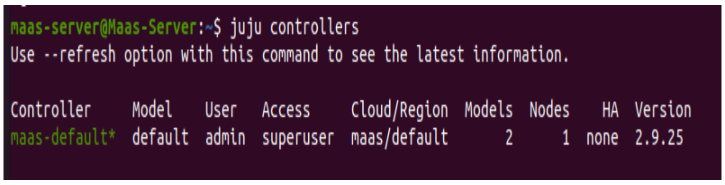


```
maas-server@Maas-Server:~$ juju add-cloud
Since Juju 2 is being run for the first time, downloaded the latest public cloud information.
This operation can be applied to both a copy on this client and to the one on a controller.
No current controller was detected and there are no registered controllers on this client: either boo
Cloud Types
  lxd
  maas
  manual
  openstack
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: maas

Enter the API endpoint url: http://146.64.19.194:5240/MAAS

Cloud "maas" successfully added to your local client.
You will need to add a credential for this cloud (`juju add-credential maas`)
before you can use it to bootstrap a controller (`juju bootstrap maas`) or
to create a model (`juju add-model <your model name> maas`).
```

Fig. 10: Juju add-cloud environment.

At this phase the Juju environment is configured by adding the MAAS server IP address and MAAS API key. The MAAS API key can be accessed by the admin users' profile in the web user interface (UI). The following cli commands creates the Juju controller using the machines tags to identify the targeted machine. On the MAAS UI the process on node being deployed will be shown and once it's done the 'Juju controllers' command will display the controller details as shown in Fig. 11.

```
$ Juju bootstrap maas –constraints="tags=Juju"
$ Juju controllers
```

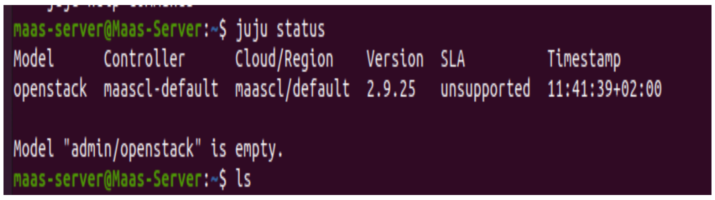


```
maas-server@Maas-Server:~$ juju controllers
Use --refresh option with this command to see the latest information.

Controller     Model    User   Access     Cloud/Region  Models  Nodes    HA  Version
maas-default*  default  admin  superuser  maas/default       2      1  none  2.9.25
```

Fig. 11: list controllers available.

In order to deploy OpenStack, a separate model must be created for organizational purposes. The model can be created with the following cli command. Once the model is created the 'Juju status' command can be used to see the summary of the environmental aspects.

```
$ sudo add-model --config default-series=focal OpenStack
$ Juju status
```



```
maas-server@Maas-Server:~$ juju status
Model      Controller     Cloud/Region   Version  SLA          Timestamp
openstack  maascl-default  maascl/default  2.9.25   unsupported  11:41:39+02:00

Model "admin/openstack" is empty.
maas-server@Maas-Server:~$ ls
```

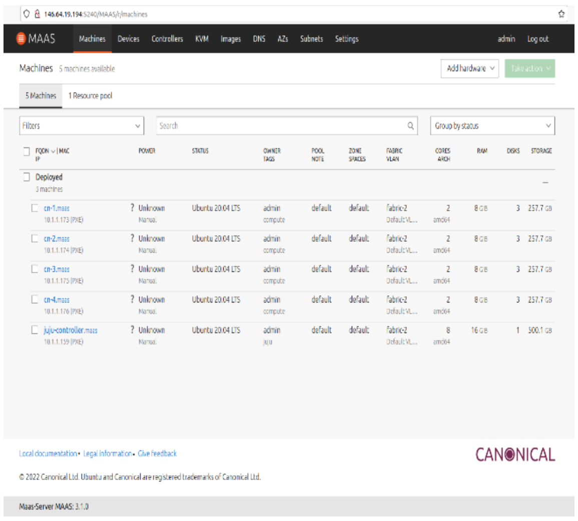
Fig. 12: Models available.

C. ADDING OPENSTACK SERVICES TO NODES

To distribute the load of services, OpenStack deploys several complicated services such as compute, controller, keystone, neutron, ceilometers, dashboard, glance, and others on multiple machines to provide a functionality to the cloud. The services can be deployed Bare Metal, LXD and KVM. To thus far we have installed MAAS, Juju, and created a Juju controller and model. The following section shows the results of installing OpenStack using individual Juju charms

V. Results and Discussions

The resource orchestration, which allocated the number of CPUs, RAM, storage, and disk on machines or nodes is displayed and the assignment of OpenStack services to machines is done on enrolled and empowered machines in the cluster. When you boot from PXE, all the machines are in a fresh state, and when you choose the machine in empowering mode, they all arrive in commissioning and then ready states, with each machine's Cores, Disk, RAM, and Storage shown separately. All the machines shown in Fig. 13 are on the deployed state. Fig. 13 shows 4 machines and 1 Juju controller on the deployed state. The 4 machines are host OpenStack services and host containers that containerized some of the OpenStack services


146.64.19.194:5240/MAAS/r/machines

Machines 5 machines available

FQDN | MAC IP	POWER	STATUS	OWNER TAGS	POOL NOTE	ZONE SPACES	FABRIC VLAN	CORES ARCH	RAM	DISKS	STORAGE
Deployed 5 machines										
cn-1.maas 10.1.1.173 (PXE)	? Unknown Manual	Ubuntu 20.04 LTS	admin compute	default	default	fabric-2 Default VL...	2 amd64	8 GB	3	257.7 GB
cn-2.maas 10.1.1.174 (PXE)	? Unknown Manual	Ubuntu 20.04 LTS	admin compute	default	default	fabric-2 Default VL...	2 amd64	8 GB	3	257.7 GB
cn-3.maas 10.1.1.175 (PXE)	? Unknown Manual	Ubuntu 20.04 LTS	admin compute	default	default	fabric-2 Default VL...	2 amd64	8 GB	3	257.7 GB
cn-4.maas 10.1.1.176 (PXE)	? Unknown Manual	Ubuntu 20.04 LTS	admin compute	default	default	fabric-2 Default VL...	2 amd64	8 GB	3	257.7 GB
juju-controller.maas 10.1.1.159 (PXE)	? Unknown Manual	Ubuntu 20.04 LTS	admin juju	default	default	fabric-2 Default VL...	8 amd64	16 GB	1	500.1 GB

Local documentation • Legal information • Give feedback

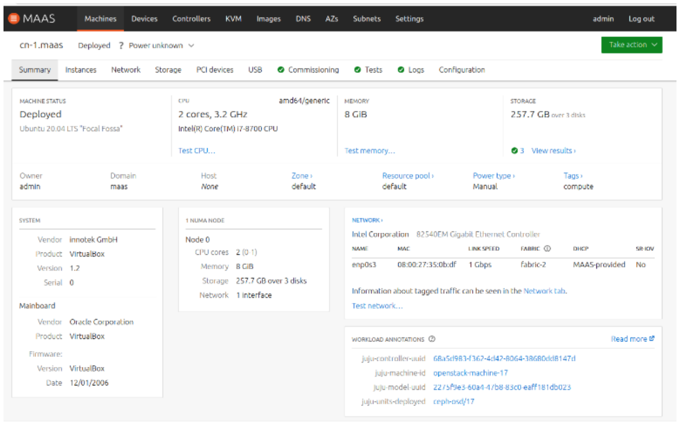
CANONICAL

© 2022 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd.

Maas-Server MAAS: 3.1.0

Fig. 13: Deployed Machines.

Fig. 14 shows how the information about the machine state, CPU, Memory, Storage, network, system, mainboard and workload annotations, the Juju controller-id, Juju models-uuid, and machine where ceph-osd is deployed.



cn-1.maas Deployed ? Power unknown

MACHINE STATUS
Deployed
Ubuntu 20.04 LTS "Focal Fossa"

CPU amd64/generic
2 cores, 3.2 GHz
Intel(R) Core(TM) i7-8700 CPU
Test CPU...

MEMORY
8 GiB
Test memory...

STORAGE
257.7 GB over 3 disks
3 View results ›

Owner	Domain	Host	Zone ›	Resource pool ›	Power type ›	Tags ›
admin	maas	None	default	default	Manual	compute

SYSTEM

Vendor	innotek GmbH
Product	VirtualBox
Version	1.2
Serial	0

Mainboard

Vendor	Oracle Corporation
Product	VirtualBox
Firmware:	
Version	VirtualBox
Date	12/01/2006

1 NUMA NODE

Node 0

CPU cores	2 (0-1)
Memory	8 GiB
Storage	257.7 GB over 3 disks
Network	1 interface

NETWORK ›

Intel Corporation 82540EM Gigabit Ethernet Controller

NAME	MAC	LINK SPEED	FABRIC	DHCP	SR-IOV
enp0s3	08:00:27:35:0b:df	1 Gbps	fabric-2	MAAS-provided	No

Information about tagged traffic can be seen in the Network tab.

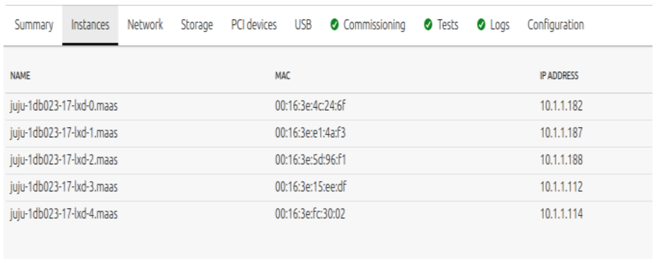
Test network...

WORKLOAD ANNOTATIONS Read more

juju-controller-uuid	68a5d983-f362-4d42-8064-38680dd8147d
juju-machine-id	openstack-machine-17
juju-model-uuid	2275f9e3-60a4-47b8-83c0-eaff181db023
juju-units-deployed	ceph-osd/17

Fig. 14: Summary of Machines Information.

Fig. 15 shows how each node has LXD instances hosted on it and exposes the IP address it belongs to.



NAME	MAC	IP ADDRESS
juju-1db023-17-lxd-0.maas	00:16:3e:4c:24:6f	10.1.1.182
juju-1db023-17-lxd-1.maas	00:16:3e:e1:4a:f3	10.1.1.187
juju-1db023-17-lxd-2.maas	00:16:3e:5d:96:f1	10.1.1.188
juju-1db023-17-lxd-3.maas	00:16:3e:15:ee:df	10.1.1.112
juju-1db023-17-lxd-4.maas	00:16:3e:fc:30:02	10.1.1.114

Fig. 15: Containers host on machines.

Once all the OpenStack service assignment is completed, the following commands in Fig. 16 will give the IP address and password where you can access the OpenStack dashboard.

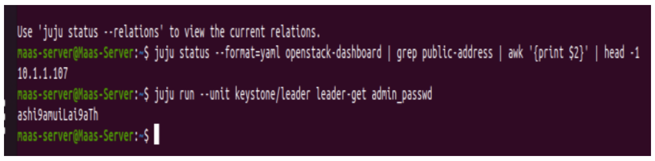

```
Use 'juju status --relations' to view the current relations.
maas-server@Maas-Server:~$ juju status --format=yaml openstack-dashboard | grep public-address | awk '{print $2}' | head -1
10.1.1.107
maas-server@Maas-Server:~$ juju run --unit keystone/leader leader-get admin_passwd
ashi9amuiLai9aTh
maas-server@Maas-Server:~$
```

Fig. 16: Horizon logging credentials.

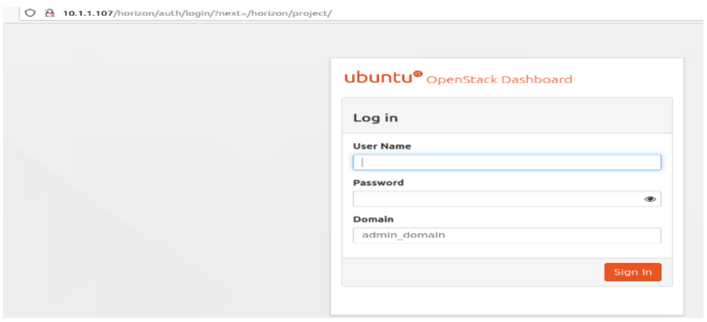
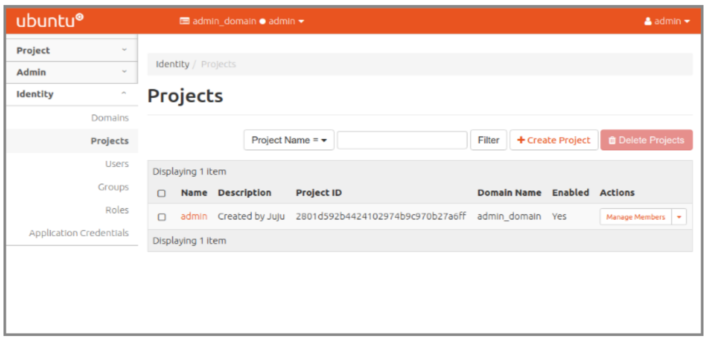

10.1.1.107/horizon/auth/login/?next=/horizon/project/

Fig. 17: OpenStack Horizon.

Once logged in to horizon you should see something like what is shown in Fig. 18.



Projects

Displaying 1 item

	Name	Description	Project ID	Domain Name	Enabled	Actions
☐	admin	Created by Juju	2801d592b4424102974b9c970b27a6ff	admin_domain	Yes	Manage Members

Displaying 1 item

Fig. 18: OpenStack Dashboard.

Conclusion

In this paper we described how to orchestrate resources and services using MAAS and Juju. This is basically an automated deployment of resource focusing on empowering the machines via PXE boot. The MAAS server cluster has successfully allocated the processing resources of 5 machines, including CPUs, RAM, disk, and storage. Furthermore, we explained how to use Juju bootstrapping to deploy the machines so that they become the cloud's owner. Furthermore, we discussed how the different OpenStack services are deployed and assigned task either on host machines or LXD containers. While explaining we also presented screenshots on how we successfully deployed a functional OpenStack using MAAS and Juju.

In future we plan to use Juju orchestrator to support network slicing leveraging an ecosystem of 5G services. The slicing must build on top of the Juju VNFM [19].

ACKNOWLEDGMENT

The authors would like to thank the Council for Scientific and Industrial Research (CSIR) Meraka Institute for support and resourceful environment.

REFERENCES

[1] S. S. Pol and S. V Gumaste, "Private Cloud: By Means of Different Open Source Softwares," no. August, 2015.

[2] G. Raj, C. Kapoor, and D. Singh Dr., "Comparative cloud deployment and service orchestration process using juju charms," Int. J. Eng. Technol., vol. 5, no. 2, pp. 1412–1418, 2013.

[3] M. Mtshali, S. Dlamini, M. Adigun, and P. Mudali, "K-Means Based on Resource Clustering for Smart Farming Problem in Fog Computing," IEEE AFRICON Conf., vol. 2019-Septe, 2019.

[4] M. Mtshali, S. Dlamini, M. O. Adigun, and P. Mudali, "Fog computing as an enabler to the Next Generation Industrial Development."

[5] A. Awasthip and R. Guptap, "Comparison of OpenStack Installers," IJISET-International J. Innov. Sci. Eng. Technol., vol. 2, no. 9, pp. 744–748, 2015.

[6] S. Islam, "Pooling of Computing Resources in Private Cloud Deployment," vol. 4, no. 11, pp. 92–98, 2017.

[7] "Top Open Source Cloud Platforms and Solutions | ComputingForGeeks." [Online]. Available: https://computingforgeeks.com/top-open-source-cloud-platforms-and-solutions/. [Accessed: 31-Mar-2022].

[8] "Top 5 Open Source Tools to Manage OpenStack Server." [Online]. Available: https://geekflare.com/openstack-opensource-tools/. [Accessed: 31-Mar-2022].

[9] "6 OpenStack Deployment tools that are awesome for your project." [Online]. Available: https://www.opcito.com/blogs/6-openstack-deployment-tools-that-are-awesome-for-your-project-and-why. [Accessed: 31-Mar-2022].

[10] "Open Source Cloud Computing Platform Software - OpenStack." [Online]. Available: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. [Accessed: 31-Mar-2022].

[11] O. Tomarchio, D. Calcaterra, and G. Di Modica, "Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks," J. Cloud Comput., vol. 9, no. 1, pp. 1–24, Dec. 2020.

[12] "Resource Orchestration Service: Simplify Computing Resources O&M - Alibaba Cloud." [Online]. Available: https://www.alibabacloud.com/product/ros. [Accessed: 31-Mar-2022].

[13] T. Yeh and S. Yu, "Realizing dynamic resource orchestration on cloud systems in the cloud-to-edge continuum," J. Parallel Distrib. Comput., vol. 160, pp. 100–109, Feb. 2022.

[14] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud Resource Orchestration Programming: Overview, Issues, and Directions," IEEE Internet Comput., vol. 19, no. 5, pp. 46–56, Sep. 2015.

[15] "What is Cloud Orchestration? | Answer from SUSE Defines." [Online]. Available: https://www.suse.com/suse-defines/definition/cloud-orchestration/. [Accessed: 31-Mar-2022].

[16] "MAAS | How it works." [Online]. Available: https://maas.io/how-it-works. [Accessed: 31-Mar-2022].

[17] "Juju | Operator lifecycle manager for K8s and traditional workloads." [Online]. Available: https://juju.is/#what-is-juju. [Accessed: 31-Mar-2022].

[18] E. Engineering, C. Town, E. Engineering, and C. Town, "v er e To w n ve rs ity e To w," 2019.

[19] K. Katsalis, N. Nikaein, and A. Huang, "JOX: An event-driven orchestrator for 5G network slicing," IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018, pp. 1–9, 2018.

Mxolisi Mtshali holds a master's in computer science from the University of Zululand. He is an Engineer in the Advance Networking and Architecture research group at the CSIR Meraka Institute in Pretoria. His research interests are IoT, Edge Computing SDN/NFV, MEC, 5G, and Network Slicing.