MAAS sPOC Lab

MAAS (Metal as a Service) is a cloud platform for managing bare metal servers and virtual machines.

It creates a single point of control for scalable automation, reconfiguration, and reliability, of networks, machines and OS images.

MAAS is designed to be cost-efficient, easy for the end-user to operate, highly-available and easy to scale, and to offer the end-user a rich set of tools for quick use or advanced integrations.

The primary goals of our MAAS Deployment are:

- Managing the lifecycle of physical hardware via traditional PXE network booting with modernized/improved useability. Users may configure systems via GUI, CLI, API, Libraries and config-management modules (i.e.: Ansible), webhooks etc. Additionally, a full suite of metrics and logging for the MAAS environment itself and deployed machines is available.
- Enabling ZTP (Zero-Touch Provisioning) / LTP (Low-Touch Provisioning) for critical systems or platforms that may take 30 minutes or more to deploy and configure via traditional methods.

> As noted, our focus shall be bare-metal management and ZTP/LTP However, I want to quickly mention that MAAS also has functionality to build and manage the lifecycle of KVM virtual machines or LXC containers. This functionality is enabled when provisioning a bare-metal server and enabling one of the virtualization options below.
>
> - Libvirt
> The Libvirt option configures the bare-metal server with the KVM and Qemu virtualization packages as well as the 'virsh' management tools.
>
> - LXD (Pronounced Lex-Dee)
> Not to be confused with LXC, This option deploys the Canonical LXD environment on the selected host. LXD is a modern, secure and powerful system container and virtual machine manager. Itprovides a unified experience for running and managing full Linux systems via LXC containers or QEMU/KVM virtual machines. Please note that further information on the hypervisor / virtualization management features of MAAS are beyond the scope of both this project and documentation. Please see the official [MAAS Docs](https://maas.io/docs) if you wish to learn more.

Official Documentation

Please see the official documentation for detailed and up-to-date information on all aspects of MAAS https://maas.io/docs

Deployment Requirements

For the purposes of this PoC, the simplest form of the MAAS standard deployment (A single All-In-0ne host) will be deployed.

The following diagram shows the key services and request workflow for a basic distributed test environment. Our deployment will be simplified to a single node, however this diagram provides *a much better visual representation.*

[Deployment Diagram]

MAAS Single-Node Requirements

For a single-host test setup assuming the latest two Ubuntu LTS releases:

COMPONENT	MEMORY (MB)	CPU (GHZ)	DISK (GB)
Region controller (minus PostgreSQL)	512	0.5	5
PostgreSQL	512	0.5	5
Rack controller	512	0.5	5
Ubuntu Server	512	0.5	5

Total: 2 GB RAM, 2 GHz CPU, 20 GB disk.

Production Base Requirements

The following information is the minimum recommendation from Canonical for a large-scale, continuous client handling deployment.

COMPONENT	Quantity	MEMORY (MB)	CPU (GHZ)	DISK (GB)
Region controller	3	2048	2.0	5
PostgreSQL Node	3	2048	2.0	20
Rack controller	n	2048	2.0	20
Ubuntu Server	1	512	0.5	5

Totals: 4.5 GB RAM, 4.5 GHz CPU, and 45 GB disk per host for region controllers, and slightly less for rack controllers.

- The Region controllers in a basic prod deployment will run Pacemaker and Corrosync. They will act as proxy hosts for the MAAS UI, Squid Proxy and Postgresql database.

Additional Notes

- These specs are MAAS-specific and don’t cover nodes for additional services
- IPMI-based BMC controllers are recommended for power management.
- Factors affecting these numbers:
    - Client activity
    - Service distribution
    - Use of high availability/load balancing.
    - Number and type of stored images

Don’t forget, a local image mirror could significantly increase disk requirements. Also, rack controllers have a 1000-machine cap per subnet. For larger networks, add

more controllers.

PHYSICAL HARDWARE INVENTORY

Several bare-metal hosts have been made available for testing MAAS functionality

- Available Inventory

MAAS CONFIGURATION AS CODE, DEVELOPMENT AND IMAGE-CREATION TOOLS

MAAS-Ansible

An ansible project with Inventory,Vars,Playbooks,Roles,Libraries and Documentation for deploying a custom configured MAAS environment has been included in the repo.

The project includes everything needed to deploy:

- All-In-One Node
- Multiple-RegionD Hosts + HAProxy + EtcD, (n) RackD Hosts, HA PostgreSQL Cluster with Pacemaker
- Optional Observability Stack - enables /metrics endpoints for all MaaS services, Grafana-Agent Metrics and Log collection, Prometheus metrics storage, Loki Log Storage, Grafana Visualization

Devcontainer / Codespace code environment

A devcontainer has been created specifically for this environment, allowing anyone to get up and running almost immediately.

If you have Docker-Desktop and check out the repository locally or into a docker-volume, a container with PyEnv (which will set up the correct python version and activate the environment automatically, then install all python dependencies, linters, formatters, ansible utilities, helper scripts, and other configurations. '

You can also open the repository in a Codespace, which will build the identical container

MAAS-Packer

The MAAS-Packer repository for custom MAAS-compatible image builds has also been included in the repository.

remotely for you.

MAAS DEPLOYMENTS IN THE LAB

MAAS Proof-Of-Concept

PoC UI

- VM Hostname: `maas-poc-reg01`

- VM IP: `44.10.4.101/24`
- Dashboard: `http://44.10.4.101:5240/MAAS/`

PoC Metrics

Metrics for the MAAS environment as well as log collection for the environment (all services + syslog) and syslog for all deployed systems has been made available.

- Grafana: `http://44.10.4.101:3000`

PoC VM Host Network

- Host IP: `44.10.4.101/24`
- VLAN Name: `VLAN1175-KMA1-RackN_PXE_Net`
- VLAN ID: `1175`
- Subnet: `44.10.4.0/24`
- Gateway: `44.10.4.1`
- Broadcast: `44.10.4.255`
- Upstream DNS: `10.240.64.124`
- Proxy URL: `http://proxy4.spoc.charterlab.com:8080`

Cisco C220M6 Provisioning Network

- VLAN ID: `77`
- Subnet: `172.22.31.144/28`
- Gateway: `172.22.31.145`
- Broadcast: `172.22.31.159`
- Dynamic Range (MAAS): `172.22.31.150-158`
- MAAS Relay: `Fabric0 Untagged -> Fabric-77 Untagged`

Cisco C220M6 Switch Pair

- lfs81s1 : `10.240.72.198`
- lfs82s1 : `10.240.72.199`

Cisco C220M6 DHCP Helper

- 44.10.4.101

PoC vCenter URL

- https://ctecco01-tnsdcvcsa01.cloud.spoc.charterlab.com/ui/app/vm;nav=v/urn:vmomi:VirtualMachine:vm-2045452:e78e7661-7e60-4d6c-9a12-86ebfeaf067e/summary

PoC HTTP Console

- https://ctecco01-tnsdcvcsa01.cloud.spoc.charterlab.com/ui/webconsole.html?vmId=vm-2045452&vmName=maas-poc-aio01&numMksConnections=1&serverGuid=e78e7661-

7e60-4d6c-9a12-86ebfeaf067e&locale=en-US

MAAS [SANDBOX] Environment

[SANDBOX] UI

- VM Hostname: `maas-sandbox-aio01`
- VM IP: `44.10.4.200/24`
- Dashboard: `http://44.10.4.200:5240/MAAS/`

[SANDBOX] Metrics

Metrics for the MAAS environment as well as log collection for the environment (all services + syslog) and syslog for all deployed systems has been made available.

- Grafana: `http://44.10.4.200:3000`

[SANDBOX] VM Host Network

- Host IP: `44.10.4.200/24`
- VLAN Name: `VLAN1175-KMA1-RackN_PXE_Net`
- VLAN ID: `1175`
- Subnet: `44.10.4.0/24`
- Gateway: `44.10.4.1`
- Broadcast: `44.10.4.255`
- Upstream DNS: `10.240.64.124`
- Proxy URL: `http://proxy4.spoc.charterlab.com:8080`

Dell R7525 Provisioning Network

- VLAN ID: `77`
- Subnet: `172.22.34.16/28`
- Gateway: `172.22.34.17`
- Broadcast: `172.22.34.31`
- Dynamic Range (MAAS): `172.22.34.20-30`
- MAAS Relay: `Fabric0 Untagged -> Fabric-77 Untagged`

Dell R7525 Switch Pair

- lfs11s3 : `10.240.72.173`
- lfs12s3 : `10.240.72.174`

Dell R7525 DHCP Helper

- 44.10.4.200

[SANDBOX] vSphere URL

- https://ctecco01-tnsdcvcsa01.cloud.spoc.charterlab.com/ui/app/vm;nav=v/urn:vmomi:VirtualMachine:vm-2042902:e78e7661-7e60-4d6c-9a12-86ebfeaf067e/summary

[SANDBOX] HTTP Console

- https://ctecco01-tnsdcvcsa01.cloud.spoc.charterlab.com/ui/webconsole.html?vmId=vm-2045452&vmName=maas-poc-aio01&numMksConnections=0&serverGuid=e78e7661-7e60-4d6c-9a12-86ebfeaf067e&locale=en-US

Packer Image Build Host

This is a custom-built host that has all dependencies installed for creating custom MAAS images via Packer.

NOTE: This host was not created in VMWare, but rather via MAAS using one of the PoC Cisco C220-M6 Devices.

- The host was provisoned with Ubuntu 22.04 with the 'libvirt' option selected prior to deployment.
- This resulted in a host capable of running KVM virtual machines which are required by the Packer build process.

This host is accessible via SSH only at: ssh://ubuntu:ubuntu@172.22.31.150
The Packer-MAAS Git Repository (which is included as a submodule in this repository for convenience) has been added to the MaaS-Packer server at **/opt/packer-maas**

[MAAS-Packer] VM Host Access

- Host IP: `172.22.31.150`
- SSH User: `ubuntu`
- SSH Password: `ubuntu`

Packer-MAAS Documentation

Official documentation for Packer+MAAS can be found at Customizing Images

Documentation for custom images should be created any time a new image type or build process is configured.

- Harmonic Documentation: packer-maas/harmonic/README.md

Packer Image Build Requirements:

- QEMU/KVM/Libvirt libraries and utilities
- Utilities to manipulate image sqauashfs filessystem
- Latest Packer release
- Templates for creating MAAS images via Packer

- Misc (See documentation)

Packer-MAAS Repository

The maas-packer repository (complete with build-templates for most operating systems) is located at `/opt/packer-maas`

Image Creation Example

- See README files in each OS subdirectory.

Uploading Custom Image to MAAS

Depending on the image source (ISO, tar.gz), the process for importing a custom image may vary. However the basic command will look like this:

```
maas admin boot-resources create name='custom/harmonic-cos'
title='Harmonic cOS' architecture='amd64/generic' filetype='ddraw'
base_image='ubuntu/jammy' content@='APOLLO_PLATFORM-release-3.21.3.0-
7+auto15.iso' size='366399488'
sha256='ce8518f4a424742bb4274374f72108060c13c256598f693be87c5d41027e4932'
```

This command contains 2 arguments that will be unique to any image upload.

- size (of image in bytes)
- sha256 (sha256 sum of image)

The following script (modified slightly to accommodate different file titles) is executed at the end of the `harmonic os installer image` build process. It will print the actual command/args required to import the created image file.

This script has been modified to take a single argument, which should be an ISO file with a name format similar to: `scgp-rocky-9.2-23q2-charter.iso`

```
#!/bin/bash

##########################################################################
# gencmd.sh
#
# Generate the complete command that will copy a given image into MAAS.
# Usage: $0 <image file>
##########################################################################

image="$1"
name="$(sed -r 's/^(.*[0-9]\.[0-9]).*/\1/' < <(echo $image))"
title="$(sed -r 's/^.+((:?\b[a-z]+)-[0-9]\.[0-9]).*$/\1/;s/-/
/;s/\b(.)/\u\1/g' < <(echo $image))"
sha256="$(sha256sum ${image} | cut -d ' ' -f1)"
bytesize="$(stat -c'%s' ${image})"
```

```
cat <<EOF

maas admin boot-resources create \
name="${name}" title="${title}" \
architecture="amd64/generic" filetype="ddraw" content@="${image}" \
sha256="${sha256}" bytesize="${bytesize}"

EOF

read -rp "Press [Enter/Return] to run this command : "

maas admin boot-resources create name="${name}" title="${title}"
architecture="amd64/generic" filetype="ddraw" content@="${image}"
sha256="${sha256}" bytesize="${bytesize}"
```

Running the above script against the named ISO will print the following:

```
❯ ./gencmd.sh scgp-rocky-9.2-23q2-charter.iso

maas admin boot-resources create name="scgp-rocky-9.2" title="Rocky 9.2"
architecture="amd64/generic" filetype="ddraw" content@="scgp-rocky-9.2-
23q2-charter.iso"
sha256="4349275b0c17adce75a4dd62efe39ba3a947db79c48ac1b8b69c84da16501969"
bytesize="2056093696"

Press [Enter/Return] to run this command :
```

Deploying Custom Image

WORK-IN-PROGRESS

All-in-One Deployment Post-Installation Steps

- Enable SSH auth keys: sed -i -E 's/^#(PubKeyAuthentication.+$)/\1/' /etc/ssh/sshd_config systemctl restart sshd
- Apt sources: cat /etc/apt/sources.list.curtin.old >> /etc/apt/sources.list
- HTTP/S Proxies for apt and env: curl -f http://http.spoc.charterlab.com/software/bootstrap/proxy.sh >> /etc/environment curl -f http://http.spoc.charterlab.com/software/bootstrap/apt.conf >> /etc/apt/apt.conf.d/98proxy
- Resize Partitions:

```
parted --align opt --script /dev/sda resizepart 4 100%
parted /dev/sda print
```

```
pvscan
pvresize /dev/sda4
lvresize -L 5GiB /dev/vg_root/lv_tmp
lvresize -L 10GiB /dev/vg_root/lv_opt
lvresize -L 10GiB /dev/vg_root/lv_varlog
lvresize -L 40GiB /dev/vg_root/lv_root
lvresize -l +100%FREE /dev/vg_root/lv_var
resize2fs /dev/vg_root/lv_opt
resize2fs /dev/vg_root/lv_var
resize2fs /dev/vg_root/lv_varlog
resize2fs /dev/vg_root/lv_tmp
resize2fs /dev/vg_root/lv_root
```

- Add default route to Netplan:

```
# This file is generated from information provided by the datasource.
Changes
# to it will not persist across an instance reboot.  To disable cloud-
init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        eth0:
            dhcp6: no
            dhcp4: no
            addresses:
              - "44.10.4.200/24"
            routes:
              - to: default
                via: 44.10.4.1
                metric: 100
                on-link: true
            nameservers:
                addresses:
                  - 10.240.64.124

    vlans:
        vlan77:
            id: 77
            dhcp6: no
            dhcp4: no
            link: eth0
            routes:
              - to: default
                via: 44.10.4.200
                metric: 200
```

```
netplan generate
netplan apply

apt-get update
systemctl restart systemd-logind
systemctl restart systemd-resolved
```

Custom Images

Image Layout

A MAAS Image Contains:

- A bootloader: which boots the computer to the point that an operating system can be loaded. MAAS currently uses one of three types of bootloaders: open firmware, PXE, and UEFI.
- A bootable kernel (aka: vmlinuz)
- An initial ramdisk (aka: initrd)
- A SquashFS filesystem.

Default Source

Canonical provides a source stream for Ubuntu and CentOS(7-8) Cloud-Images. It is configured by default in MAAS

- http://images.maas.io/ephemeral-v3/candidate

MAAS Image URL Example

- https://images.maas.io/ephemeral-v3/stable/streams/v1/com.ubuntu.maas:stable:1:bootloader-download.sjson

Additional Reading / Documentation

MAAS Specific Glossary and Useful Terms

- Secure UEFI Boot
- PXE Boot instances
- Customize slightly
- Image cache
- Provisioning Scripts
- DHCP Snippets
- VLANs
- Subnets
- Customized configuration
- Power management / Integration

Fabrics

A fabric connects VLANs. If you understand a VLAN, you know that they permit network connections only between specific switch ports or specifically identified ports ("tagged" ports). Consequently, it would be impossible for two VLANs to communicate with each other. A fabric makes these VLAN-to-VLAN connections possible. Take me on a quick, deep dive on fabrics We can illustrate a network fabric more easily by rewinding the term to one of its earliest uses: the early phone system. In a telephone switchboard, subscriber lines (customer phone numbers) ran in a grid pattern in the back of the switchboard, but they didn't touch each other until the operator inserted the plugs of a patch cable to join them. With some "plugboards" (what a switchboard was actually called), an operator could conference multiple lines by adding more patch cords. These patch cords essentially acted like a VLAN, allowing only the subscribers whose lines were "patched in" to join the conversation. But the switchboard only covered one exchange, that is, one three-digit phone number prefix. If a subscriber wanted to conference someone from another exchange, there had to be patch from one exchange to another. This was handled by a long-distance operator. Each exchange had a more robust outgoing line, called a "trunk line," that connected exchanges in some central place. The long-distance operators could bridge trunks in a specific way, involving a local operator in each of the "bridged" exchanges. By now, you're probably starting to recognise a lot of network terms, which is completely appropriate. Almost all modern networking technology originated in the telephone system. Now imagine that you want to conference in six people, two in each of three distant exchanges. Each exchange operator had to patch two numbers and a trunk line. The long-distance operator had to patch three trunks in a specific way that prevented the conversation from going out to all numbers attached to the trunk. The details of the method aren't particularly relevant here, but it usually involved a pair of "bridge clips" that connected non-adjacent wire-crossings, with an insulated portion that laid across wires that weren't meant to be connected. It looked a lot like a little bridge when properly placed. Think of each of the local exchange conferences as a VLAN; the long-distance operator's patch cables created what was called a "fabric." Our use of fabric is exactly the same idea: some number of private "conversations" (connections) connected to each other so that specific people in each "group" can all talk to each other. You could describe a fabric as a VLAN namespace. It's a switch or a combination of switches that use trunking to provide access to specific VLANs. MAAS creates a default fabric ('fabric-0') for each detected subnet during installation. The following conceptual diagram shows two fabrics in the same data centre or region, each using distinct VLAN ranges and their associated subnets:

Availability Zones

Availability zones in MAAS serve as logical partitions that let you group and isolate resources. Think of them as a way to organise your physical and virtual machines for optimal efficiency and fault tolerance.

Zones

A physical zone, or just zone, is an organisational unit that contains nodes where each node is in one, and only one, zone. Later, while in production, a node can be taken (allocated) from a specific zone (or not from a specific zone). Since zones, by nature, are custom-designed (except for the 'default' zone), they provide more flexibility than a similar feature offered by a public cloud service (ex: availability zones). Some prime examples of zone usage include fault-tolerance, service performance, and power management. A newly installed MAAS comes with a default zone which contains all nodes unless you create a new zone. You can therefore safely ignore the entire concept if you’re not interested in leveraging zones. You cannot remove the ‘default’ zone or change its name.

Spaces

A space is a logical grouping of subnets that can communicate with one another. Spaces can be arranged to group subnets according to various parameters. One of the most common examples is a DMZ space, which might group subnets presenting a web interface to the public Internet. Behind this DMZ would be specific applications that aren’t allowed to interact directly with the user, but instead must interact with a Web UI in the DMZ space. MAAS does not create a default space during installation. Spaces facilitate machine allocation for Juju^. See Juju network spaces^ for more details.

Regions

A region is an organisational unit one level above a zone. It contains all information about all machines running in any possible zones. In particular, the PostgreSQL database runs at this level and maintains state for all these machines.

Controllers

There are two types of controllers: a region controller and a rack controller. The region controller deals with operator requests while one or more rack controllers provide the high-bandwidth services to multiple server racks, as typically found in a data centre.

Region

A region controller consists of the following components:

- REST API server (TCP port 5240)
- PostgreSQL database
- DNS
- caching HTTP proxy
- web UI

Think of a region controller can as being responsible for a data centre, or a single region. Multiple fabrics are used by MAAS to accommodate subdivisions within a single region, such as multiple floors in a data centre.

Rack

A rack controller provides the following services:

- DHCP
- TFTP
- HTTP (for images)
- power management

A rack controller is attached to each “fabric”. As the name implies, a typical setup is to have a rack controller in each data centre server rack. The rack controller will cache large items for performance, such as operating system install images, but maintains no independent state other than the credentials required to talk to the region controller. Both the region controller and the rack controller can be scaled-out as well as made highly available.

MAAS Network Configuration Exports


```
[
    {
        "name": "44.10.4.0/24",
        "description": "Default Subnet",
        "vlan": {
            "vid": 0,
            "mtu": 1500,
            "dhcp_on": true,
            "external_dhcp": null,
            "relay_vlan": null,
            "primary_rack": "mrw8bs",
            "secondary_rack": null,
            "fabric_id": 0,
            "id": 5001,
            "fabric": "fabric-vmware",
            "name": "untagged",
            "space": "POC",
            "resource_uri": "/MAAS/api/2.0/vlans/5001/"
        },
        "cidr": "44.10.4.0/24",
        "rdns_mode": 2,
        "gateway_ip": "44.10.4.1",
        "dns_servers": [
            "44.10.4.200"
        ],
        "allow_dns": true,
        "allow_proxy": true,
        "active_discovery": false,
        "managed": true,
        "disabled_boot_architectures": [],
        "id": 1,
        "space": "POC",
        "resource_uri": "/MAAS/api/2.0/subnets/1/"
    },
    {
        "name": "r7525",
        "description": "",
        "vlan": {
            "vid": 77,
            "mtu": 1500,
            "dhcp_on": false,
            "external_dhcp": null,
            "relay_vlan": {
                "vid": 0,
```

```
                "mtu": 1500,
                "dhcp_on": true,
                "external_dhcp": null,
                "relay_vlan": null,
                "primary_rack": "mrw8bs",
                "secondary_rack": null,
                "fabric_id": 0,
                "id": 5001,
                "fabric": "fabric-vmware",
                "name": "untagged",
                "space": "POC",
                "resource_uri": "/MAAS/api/2.0/vlans/5001/"
            },
            "primary_rack": null,
            "secondary_rack": null,
            "fabric_id": 35,
            "id": 5055,
            "fabric": "fabric-r7525",
            "name": "untagged",
            "space": "POC",
            "resource_uri": "/MAAS/api/2.0/vlans/5055/"
        },
        "cidr": "172.22.34.16/28",
        "rdns_mode": 2,
        "gateway_ip": "172.22.34.17",
        "dns_servers": [
            "44.10.4.200"
        ],
        "allow_dns": true,
        "allow_proxy": true,
        "active_discovery": false,
        "managed": true,
        "disabled_boot_architectures": [],
        "id": 42,
        "space": "POC",
        "resource_uri": "/MAAS/api/2.0/subnets/42/"
    }
]
```



```
[
    {
        "class_type": null,
        "id": 0,
        "name": "fabric-vmware",
        "vlans": [
            {
                "vid": 0,
                "mtu": 1500,
                "dhcp_on": true,
                "external_dhcp": null,
                "relay_vlan": null,
```

```
                "fabric": "fabric-vmware",
                "id": 5001,
                "primary_rack": "mrw8bs",
                "name": "untagged",
                "fabric_id": 0,
                "space": "POC",
                "secondary_rack": null,
                "resource_uri": "/MAAS/api/2.0/vlans/5001/"
            }
        ],
        "resource_uri": "/MAAS/api/2.0/fabrics/0/"
    },
    {
        "class_type": null,
        "id": 35,
        "name": "fabric-r7525",
        "vlans": [
            {
                "vid": 77,
                "mtu": 1500,
                "dhcp_on": false,
                "external_dhcp": null,
                "relay_vlan": {
                    "vid": 0,
                    "mtu": 1500,
                    "dhcp_on": true,
                    "external_dhcp": null,
                    "relay_vlan": null,
                    "fabric": "fabric-vmware",
                    "id": 5001,
                    "primary_rack": "mrw8bs",
                    "name": "untagged",
                    "fabric_id": 0,
                    "space": "POC",
                    "secondary_rack": null,
                    "resource_uri": "/MAAS/api/2.0/vlans/5001/"
                },
                "fabric": "fabric-r7525",
                "id": 5055,
                "primary_rack": null,
                "name": "untagged",
                "fabric_id": 35,
                "space": "POC",
                "secondary_rack": null,
                "resource_uri": "/MAAS/api/2.0/vlans/5055/"
            }
        ],
        "resource_uri": "/MAAS/api/2.0/fabrics/35/"
    }
]
```

```
[
    {
        "vid": 77,
        "mtu": 1500,
        "dhcp_on": false,
        "external_dhcp": null,
        "relay_vlan": {
            "vid": 0,
            "mtu": 1500,
            "dhcp_on": true,
            "external_dhcp": null,
            "relay_vlan": null,
            "primary_rack": "mrw8bs",
            "secondary_rack": null,
            "fabric_id": 0,
            "id": 5001,
            "fabric": "fabric-vmware",
            "name": "untagged",
            "space": "POC",
            "resource_uri": "/MAAS/api/2.0/vlans/5001/"
        },
        "primary_rack": null,
        "secondary_rack": null,
        "fabric_id": 35,
        "id": 5055,
        "fabric": "fabric-r7525",
        "name": "untagged",
        "space": "POC",
        "resource_uri": "/MAAS/api/2.0/vlans/5055/"
    }
]
```

```
[
    {
        "vid": 0,
        "mtu": 1500,
        "dhcp_on": true,
        "external_dhcp": null,
        "relay_vlan": null,
        "fabric": "fabric-vmware",
        "id": 5001,
        "primary_rack": "mrw8bs",
        "name": "untagged",
        "fabric_id": 0,
        "space": "POC",
        "secondary_rack": null,
        "resource_uri": "/MAAS/api/2.0/vlans/5001/"
    }
]
```