

---

# Deploying Cloud Foundry on Private Cloud Documentation

*Release 1*

**EniWARE**

**Jan 16, 2019**



<b>1</b>	<b>1. Install MAAS</b>	<b>3</b>
1.1	1.1. Requirements	3
1.2	1.2. Installation	4
1.3	1.3. On-boarding	4
1.4	1.4. Connectivity and images	5
1.5	1.5. SSH key	5
1.6	1.6. Networking	6
	1.6.1 Extending a reserved dynamic IP range:	6
	1.6.2 Enabling DHCP:	7
1.7	1.7. Images	8
1.8	1.8. Network services	8
1.9	1.9. Adding nodes	9
1.10	1.10. Commission nodes	11
1.11	1.11. Next steps	13
<b>2</b>	<b>2. Install Juju</b>	<b>15</b>
2.1	2.1. Package installation	15
2.2	2.2. Client configuration	16
2.3	2.3. Testing the environment	17
2.4	2.4. Opening the Juju GUI	18
2.5	2.5. Next steps	18
<b>3</b>	<b>3. Install OpenStack</b>	<b>21</b>
3.1	3.1. Juju controller deployment	21
3.2	3.2. OpenStack deployment	22
3.3	3.3. OpenStack testing	27
3.4	3.4. Next steps	28
<b>4</b>	<b>4. Configure OpenStack</b>	<b>29</b>
4.1	4.1. Installing clients for different OpenStack operations	29
4.2	4.2. Environment variables	30
4.3	4.3. Define an external network	31
	4.3.1 Define an external network using web UI:	31
	4.3.2 Define an external network using CLI:	32
4.4	4.4. Cloud images	34
4.5	4.5. Working with flavors	34
	4.5.1 Working with flavors using web UI:	35

4.5.2	Working with flavors using CLI: . . . . .	37
4.6	4.6. Working with domains, projects and users . . . . .	38
4.6.1	Working with domains and projects using web UI: . . . . .	38
4.6.2	Working with domains and projects using CLI: . . . . .	40
4.7	4.7. View and manage quotas . . . . .	41
4.7.1	View and manage quotas using web UI: . . . . .	41
4.7.2	View and manage quotas using CLI: . . . . .	42
4.8	4.8. Next steps . . . . .	44
<b>5</b>	<b>5. Deploying CloudFoundry with BOSH Director on OpenStack</b>	<b>45</b>
5.1	5.1. Prerequisites . . . . .	45
5.2	5.2. CF-OpenStack-Validator installation . . . . .	46
5.2.1	5.2.1. Prerequisites for CF-OpenStack-Validator . . . . .	46
5.2.2	5.2.2. Installation of CF-OpenStack-Validator . . . . .	46
5.2.3	5.2.3. Additional configurations . . . . .	47
5.3	5.3. Validate the OpenStack configuration . . . . .	47
5.4	5.4. Prepare OpenStack environment for BOSH and Cloud Foundry via Terraform . . . . .	48
5.4.1	5.4.1. Install Terraform module . . . . .	48
5.4.2	5.4.2. OpenStack environment for BOSH . . . . .	48
5.4.2.1	Setup an OpenStack project to install BOSH: . . . . .	48
5.4.2.2	Terraform tempalte file configuration for BOSH: . . . . .	49
5.4.3	5.4.3. OpenStack environment for Cloud Foundry . . . . .	50
5.4.3.1	Setup an OpenStack project to install Cloud Foundry: . . . . .	50
5.4.3.2	Terraform tempalte file configuration for Cloud Foundry: . . . . .	50
5.5	5.5. Install BOSH . . . . .	51
5.6	5.6. Cloud Config . . . . .	51
5.7	5.7. Deploy Cloud Foundry . . . . .	52
<b>6</b>	<b>6. Install BOSH</b>	<b>53</b>
6.1	6.1. Getting Started . . . . .	53
6.2	6.2. Installing the BOSH CLI . . . . .	53
6.2.1	Using the binary directly . . . . .	54
6.2.2	Using Homebrew on macOS . . . . .	54
6.3	6.3. Additional Dependencies . . . . .	54
6.3.1	Ubuntu Trusty . . . . .	54
6.4	6.4. Quick Start . . . . .	55
6.4.1	Prerequisites . . . . .	55
6.4.2	Install . . . . .	55
6.4.3	Deploy . . . . .	56
6.4.4	Clean up . . . . .	56
6.5	6.5. Initialize New Environment on OpenStack . . . . .	56
<b>7</b>	<b>Indices and tables</b>	<b>57</b>



# EniWARE

**Version** 1

**Language** en

**Description**

**EniWARE** Deploying Cloud Foundry on Private Cloud

**Author** EniWARE

**Rendered** Jan 16, 2019

**Contents:**



---

## 1. Install MAAS

---

**MAAS** (Metal As A Service) - provides the management of physical servers like virtual machines in the cloud. MAAS can work at any scale, from a test deployment using a handful of machines to thousands of machines deployed across multiple regions.

The typical MAAS environment includes as a framework for deployment the following:

- A Region controller interacts with and controls the wider environment for a region.
- One or more Rack controllers manage locally grouped hardware, usually within a data centre rack.
- Multiple Nodes are individual machines managed by the Rack controller, and ultimately, the Region controller.
- Complex Networking topologies can be modelled and implemented by MAAS, from a single fabric to multiple zones and many overlapping spaces.

---

**Note:** See [Concepts and Terms](#) in the MAAS documentation for clarification on the terminology used within MAAS.

---

### 1.1 1.1. Requirements

The minimum requirements for the machines that run MAAS vary widely depending on local implementation and usage. The minimum requirements for the machines that run MAAS are considered in the [MAAS documentation](#).

The hardware that will be used for the purpose of this documentation is based on the following specifications:

- 1 x MAAS Rack with Region controller: 8GB RAM, 2 CPUs, 2 NIC (one for IPMI and one for the network), 40GB storage

Your hardware could differ considerably from the above and both MAAS and Juju will easily adapt. The Juju node could operate perfectly adequately with half the RAM (this would need to be defined as a bootstrap constraint) and adding more nodes will obviously improve performance.

---

**Note:** It will be used the web UI whenever possible. However it can also be used [CLI](#) and the [API](#).

---

### 1.2 1.2. Installation

First, you need to have fresh install of [Ubuntu Server 18.04 LTS](#) on the machine that will be hosting both the MAAS Rack and Region Controllers.

In our case, as a hosting machine is used VM machine created in [ESXi 6.5 \(VMware ESXi\)](#). You can use the ESXi client [vSphere Client](#).

The configuration of the network is depends on your own infrastructure (see the [Ubuntu Server Network Configuration](#) documentation for further details on modifying your network configuration).

For the purposes of this documentation, the IP address configured for the MAAS machine hosted on Ubuntu is set to be 192.168.40.16.

To update the package database and install MAAS, issue the following commands:

```
sudo apt update
sudo apt install maas
```

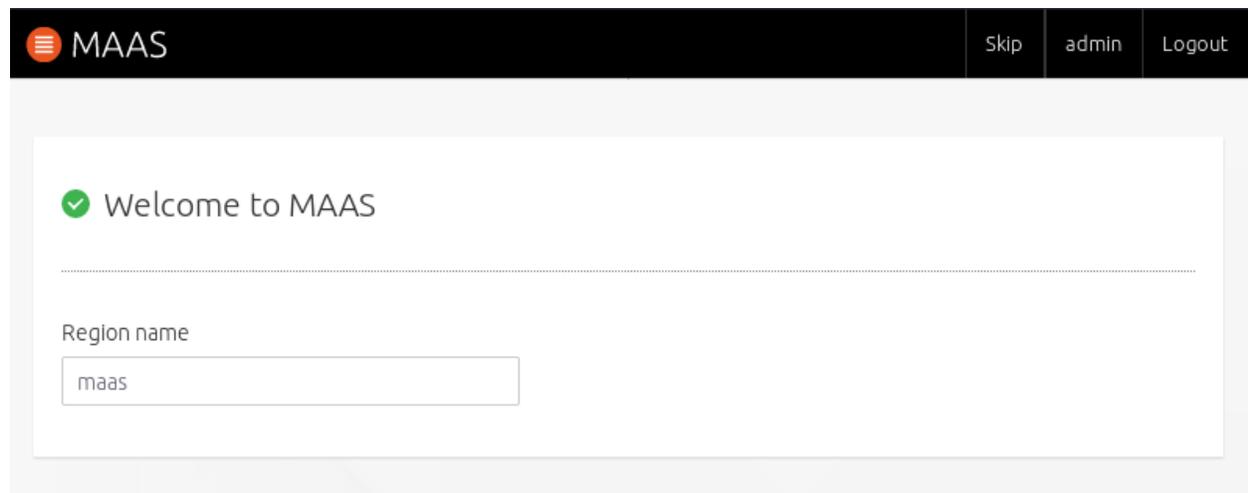
Before MAAS can be configured an administrator account need to be created:

```
sudo maas createadmin
```

An username, password and email address should be filled in. After that you need to specify if you want to import an SSH key. MAAS uses the public SSH key of a user to manage and secure access to deployed nodes. If you want to skip this, press `Enter`. In the next step you can do this from the web UI.

### 1.3 1.3. On-boarding

MAAS is now running without being configured. You can check this by pointing your browser to <http://<your.maas.ip>:5240/MAAS/>. Now you sign in with the login credentials, and the web interface will launch the ‘Welcome to MAAS’ on-boarding page:



## 1.4 1.4. Connectivity and images

There are two steps left necessary for MAAS to get up and running. Unless you have specific requirements, most of these options can be left at their default values:

- **Connectivity:** important services that default to being outside of your network. These include package archives and the DNS forwarder.
- **Ubuntu:** for deployed nodes, MAAS needs to import the versions and image architectures. Specify 18.04 LTS as well as 16.04 LTS to add additional image.

✓
Ubuntu

---

Select images and architecture to be imported and kept in sync daily. Images will be available for deploying to machines managed by MAAS.

Choose source

maas.io
 Custom

### Images

18.04 LTS

18.10

16.04 LTS

17.10

14.04 LTS

17.04

12.04 LTS

16.10

### Architectures

amd64

i386

arm64

ppc64el

armhf

s390x

Release	Architecture	Size	Status	Actions
<input checked="" type="radio"/> 18.04 LTS	amd64	315.4 MB	Synced	
<input type="radio"/> 16.04 LTS	amd64	-	Selected for download	

Update selection

## 1.5 1.5. SSH key

You have several options for importing your public SSH key(s). One is to import the key from Launchpad or Github by entering your user ID for these services. Another option is to add a local public key file, usually `HOME/ssh/id_rsa.pub` by selecting **Upload** button and placing the content in the box that appears. Click **Import** to complete the setting.

Adding SSH keys completes this initial MAAS configuration. Press **Go** to the dashboard to move to the MAAS dashboard and the device discovery process

You can generate a local SSH public/private key pair from the Linux account you are using for managing MAAS. When asked for a passphrase, leave it blank.

MAAS 10.0

✓ SSH keys for eniware

Add multiple keys from Launchpad and Github or enter them manually.

### Keys

Source	ID	Number of keys
Upload	ssh-rsa AAAAB3NzaC1yc2EAAAADAQ...	

Source  ID

```
ssh-keygen -t rsa
```

This completes the initial setup of MAAS. Press **Go** button to the dashboard to go to the device discovery process.

## 1.6. Networking

By default, MAAS will monitor local network traffic and report any devices it discovers on the **Network discovery** page of the web UI. This page is basic and is the first one to load after finishing installation.

### Network discovery

2 items discovered

Discovery enabled

ⓘ DHCP is not enabled on any VLAN. This will prevent machines from being able to PXE boot, unless an external DHCP server is being used.

[Dismiss](#)

Name	MAC address	IP	Rack	Last seen	Actions
_gateway	64:d1:54:d0:00:13 Unknown	192.168.40.1	MAAS	Tue, 16 Oct. 2018 19:24:56	⌵
unknown	00:25:90:80:35:c0 Nest Labs Inc.	192.168.40.187	MAAS	Tue, 16 Oct. 2018 19:29:42	⌵

Before taking the configuration further, you need to tell MAAS about your network and how y'd like connections to be configured.

### 1.6.1 Extending a reserved dynamic IP range:

**Note:** If you do not have DHCP reserved ranges in your network, you can skip to the step *Enabling DHCP*.

If DHCP reserved ranges are defined in your network, you have to set the appropriate settings described below.

These options are managed from the **Subnets** page of the web UI. The subnets page defaults to listing connections by fabric and MAAS creates one fabric per physical NIC on the MAAS server. Once you are set up a machine with a single NIC, a single fabric will be listed linked to the external subnet.

You should select the **untagged** VLAN the subnet to the right of **fabric-0** and add in the **Reserved ranges** field the reserved portions of the subnet to the dynamic IP range:

### Reserved ranges

Start IP Address	End IP Address	Owner	Type	Comment	Actions
192.168.40.1	192.168.40.42	eniware	Reserved		
192.168.40.60	192.168.40.100	eniware	Reserved		
192.168.40.122	192.168.40.122	eniware	Reserved		
192.168.40.154	192.168.40.184	eniware	Reserved		
192.168.40.191	192.168.40.254	MAAS	Dynamic	Dynamic	

Furthermore, since DHCP is enabled on a VLAN basis and a VLAN can contain multiple subnets, it is possible to add a portion from those subnets as well. Just select the subnet under the ‘Subnets’ page and reserve a dynamic range.

### 1.6.2 Enabling DHCP:

You can add DHCP by selecting **untagged** VLAN the subnet to the right of **fabric-0**.

The page that appears will be labelled something similar to **Default VLAN in fabric-0**. From here, click the **Take action** button at the top right and select **Provide DHCP**.

If you do not have reserved ranges of IP addresses, a new pane will appear that allows you to specify the start and end IP addresses for the DHCP range. Select **Provide DHCP** to accept the default values. The VLAN summary should now show DHCP as **Enabled**.

If you have reserved ranges of IP addresses, a new pane will appear that shows us the current **Rack controller**. Select **Provide DHCP** to accept the settings and the VLAN summary should now show DHCP as **Enabled**.

### Default VLAN in fabric-0

Provide DHCP

Rack controller

## 1.7 1.7. Images

You have already downloaded the images you need as part of the on-boarding process, but it's worth checking that both the images you requested are available. To do this, select the **Images** page from the top menu of the web UI.

The **Images** page allows you to download new images, use a custom source for images, and check on the status of any images currently downloaded. These appear at the bottom, and both 18.04 LTS and 16.04 LTS should be listed with a status of **Synced**.

Release	Architecture	Size	Status	Actions
<input checked="" type="radio"/> 18.04 LTS	amd64	310.1 MB	Synced	
<input checked="" type="radio"/> 16.04 LTS	amd64	495.7 MB	Synced	

## 1.8 1.8. Network services

Before *adding new nodes*, it is necessary to configure the network services. From the **Settings** menu select **Network services**.

**Warning:** In the **Proxy** field for **HTTP proxy used by MAAS to download images** is selected **MAAS Built-in** by default. It is necessary to select **Do not use a proxy**.

### Proxy

HTTP proxy used by MAAS to download images, and by provisioned machines for APT packages.

- Don't use a proxy
- MAAS Built-in
- External
- Peer

In the **DNS** field, it is necessary to set **Upstream DNS used to resolve domains not managed by this MAAS**. In our case, we assign DNS address 8 . 8 . 8 . 8 (which is [Google Public DNS IP addresses](#)).

## DNS

Upstream DNS used to resolve domains not managed by this MAAS (space-separated IP addresses)

Only used when MAAS is running its own DNS server. This value is used as the value of 'forwarders' in the DNS server config.

Enable DNSSEC validation of upstream zones

Only used when MAAS is running its own DNS server. This value is used as the value of 'dnssec\_validation' in the DNS server config.

[Save](#)

## 1.9 1.9. Adding nodes

MAAS is now ready to accept new nodes. To do this, first ensure your four cloud nodes and single Juju node are set to boot from a PXE image. Now simply power them on. MAAS will add these new nodes automatically by taking the following steps:

- Detect each new node on the network
- Probe and log each node's hardware (using an ephemeral boot image)
- Add each node to the **Machines** page with a status of **New**

While it is not the most appropriate way, at this stage it is advisable to include each node individually in order to trace each one strictly.

In order for MAAS to fully manage a node it must be able to power cycle it. This is done via a communication channel with the **BMC** card of the node's underlying system. A newly added node is therefore incomplete until its power type has been configured.

---

**Note:** See the [MAAS documentation](#) for more information on power types, including a [table](#) showing a feature comparison for the supported BMC drivers.

---

To configure a node's power type, begin by clicking on the node from the **Machines** page of the web UI followed by its **Configuration** tab. Scroll down for **Power configuration**. If the power type is undefined the following will be displayed:

⊗ Error: This node does not have a power type set and MAAS will be unable to control it. Update the power information below.

Choose a type in the dropdown menu that corresponds to the node's underlying machine's BMC card.

### Power configuration

Power type	American Power Con <sup>n</sup> ▾
IP for APC PDU	Select your power type
APC PDU node outlet number (1-16)	American Power Conversion (APC) PDU
Power ON outlet delay (seconds)	Christmann RECS Box Power Driver
	Cisco UCS Manager
	Digital Loggers, Inc. PDU
	Facebook's Wedge
	HP Moonshot - iLO Chassis Manager
	HP Moonshot - iLO4 (IPMI)
	IBM Hardware Management Console (HMC)
	IPMI
	Intel AMT
	Manual
	Microsoft OCS - Chassis Manager
	OpenStack Nova
	Rack Scale Design
	SeaMicro 15000
	Sentry Switch CDU
	VMware
	Virsh (virtual systems)

Cancel Save changes

Use the drop-down **Power type** menu to open the configuration options for your node's specific power configuration and enter any further details that the configuration may require. When you make the necessary changes, click **Save changes**. The machine can now be turned off from the **Take option** menu in the top right.

By default, the machine gets a random name. It is recommended that the name of each new machine be edited in accordance with its intended purpose. This can be done by selecting the corresponding machine from the **Machines** page. A **Machine summary** field opens where in the upper left corner we have to click and change the name of the selected machine and save the change with the **Save** button:

loved-tahr . maas ▾ Cancel Save

**Warning:** If you add a node (machine) and then remove it without deleting the disks, you will not be able to add this node again! To add the node manually, please see the [official documentation](#) or follow the steps outlined above in [this section](#).

To add the node, you need the following information about your machine: the MAC address of your IPMI interface and the MAC address of your PXE interface. After entering the information, you have to restart the processes *MAAS controller* and *Region controller* using the following commands:

```
sudo service maas-rackd restart
sudo service maas-regiond restart
```

## 1.10 1.10. Commission nodes

Once a node is added to MAAS (see [Adding nodes](#)) the next logical step is to *commission* it.

To commission, the underlying machine needs to be configured to netboot (this should already have been done during the enlistment stage). Such a machine will undergo the following process:

1. DHCP server is contacted
2. kernel and initrd are received over TFTP
3. machine boots
4. initrd mounts a Squashfs image ephemerally over HTTP
5. cloud-init runs commissioning scripts
6. machine shuts down

The commissioning scripts will talk to the region API server to ensure that everything is in order and that eventual deployment will succeed.

The image used is, by default, the latest Ubuntu LTS release and should not require changing. However, it can be configured in the **Settings** page of the web UI by selecting the **General** tab and scrolling down to the **Commissioning** section.

To commission, on the **Machines** page, select a node and choose **Commission** under the **Take action** drop-down menu.



You have the option of selecting some extra parameters (checkboxes) and performing hardware tests. These options include:

- **Allow SSH access and prevent machine powering off:** Machines are normally powered off after commissioning. This option keeps the machine on and enables SSH so you can access the machine.
- **Retain network configuration:** When enabled, preserves any custom network settings previously configured for the machine. See [Networking](#) for more information.

- **Retain storage configuration:** When enabled, preserves any storage settings previously configured for the machine. See [Storage](#) for more details.
- **Update firmware:** Runs scripts tagged with `update_firmware`. See [Testing scripts](#) for more details.
- **Configure HBA:** Runs scripts tagged with `configure_hba`. As above, see [Testing scripts](#) for further details.

The screenshot shows the MAAS machine configuration page for 's56.maas'. At the top left, the machine name 's56.maas' is displayed next to a 'New' label. On the top right, there is a 'Commission' dropdown menu. Below this, there are four checkboxes for configuration options: 'Allow SSH access and prevent machine from powering off', 'Retain network configuration', 'Retain storage configuration', 'Update firmware', and 'Configure HBA'. A 'Hardware tests' section contains a text input field with 'smartctl-validate' and a close button (X). Below the input field is a 'Select scripts' button. At the bottom right, there are two buttons: 'Cancel' and a green 'Commission machine' button.

Click the **Hardware tests** field to reveal a drop-down list of tests to add and run during commissioning. See [Hardware testing](#) for more information on hardware testing scripts.

From the **Hardware tests** field, we deactivate `smartctl-validate`, which will speed up work as SMART health for all drivers in parallel will not be validated.

This is a close-up of the 'Hardware tests' field from the previous screenshot. It shows a text input field containing 'smartctl-validate' with a close button (X) to its right. Below the input field is a 'Select scripts' button. At the bottom right, there are two buttons: 'Cancel' and a green 'Commission machine' button.

Finalise the directive by hitting **Commission machine**. While a node is commissioning its status will change to *Commissioning*. During this time the node's network topology will be discovered. This will prompt one of the node's network interfaces to be connected to the fabric, VLAN, and subnet combination that will allow it to be configured. By default, a static IP address will be assigned out of the reserved IP range for the subnet. That is, an IP assignment mode of **Auto assign** will be used.

After a few minutes, successfully commissioned nodes will change their status to **Ready**. The CPU cores, RAM, number of drives and storage fields should now correctly reflect the hardware on each node.

**Tags** are normally used to identify nodes with specific hardware, such as GPUs for GPU-accelerated CUDA processing. This allows Juju to target these capabilities when deploying applications that may use them. But they can also be used for organisational and management purposes. This is how you are going to use them, by adding a **compute** tag to the four cloud nodes and a **juju** tag to the node that will act as the Juju controller.

Tags are added from the **Machine summary** section of the same individual node page we used to rename a node. Click **Edit** on the **Tags** section. A tag is added by activated **Edit** function in **Machine configuration** field and entering a

name for the tag in the empty field and clicking **Save changes**.

## Machine configuration Edit

Architecture amd64/generic v

Minimum Kernel No minimum kernel v

Zone default v

Resource pool default v

Tags juju X  
Add a tag

Cancel
Save changes

A common picture of the state of the nodes that have already been added to the MAAS. You can see the names, tags, and hardware information on each node:

Node name	Tag(s)	CPU(s)	RAM	Drives	Storage
os-compute01.maas	compute	2	6.0	3	85.9
os-compute02.maas	compute	2	6.0	3	85.9
os-compute03.maas	compute	2	6.0	3	85.9
os-compute04.maas	compute	2	6.0	3	85.9
os-juju01.maas	juju	2	4.0	1	42.9

## 1.11 1.11. Next steps

The next step is to *deploy the Juju controller* onto its own node. From there, you will be using Juju and MAAS together to deploy OpenStack into the four remaining cloud nodes.



---

## 2. Install Juju

---

Juju is an open source application modelling tool that allows you to deploy, configure, scale and operate your software on public and private clouds.

In the *previous step*, we installed, deployed and configured MAAS to use as a foundation for Juju to deploy a fully fledged OpenStack cloud.

We are now going to install and configure the following two core components of Juju to use our MAAS deployment:

- The *controller* is the management node for a cloud environment. We will be using the MAAS node we tagged with **juju** to host the Juju controller.
- The *client* is used by the operator to talk to one or more controllers, managing one or more different cloud environments. As long as it can access the controller, almost any machine and operating system can run the Juju client.

### 2.1 2.1. Package installation

We're going to start by installing the **Juju client** on a VM machine created in [ESXI 6.5](#) with the following parameters:

- 1 x Juju node: 4GB RAM, 2 CPUs, 1 NIC, and 40 GB Storage
- installed [Ubuntu Server 18.04 LTS](#) with network access to the MAAS deployment.

For other installation options, see [Getting started with Juju](#).

---

**Important:** When you install the operation system do not forget to install SSH agent ([Open SSH for Ubuntu Server 18.04 LTS](#))!

---

To install Juju, enter the following in the terminal:

```
sudo add-apt-repository -yu ppa:juju/stable
sudo apt install juju
```

**Note:** If you have problems to clone juju repository use the following command:

```
sudo apt install software-properties-common
```

For the purposes of this documentation, the IP address configured for the Juju client hosted on Ubuntu is set to be 192.168.40.17. After the installation is complete you can change the IP address (if it necessary).

Go to `/etc/netplan/` directory and edit the file `01-netcfg.yaml` using the following command:

```
sudo nano /etc/netplan/01-netcfg.yaml
```

To stop DHCP use the **networkd** daemon to configure your network interface:

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    ens160:
      dhcp4: no
      addresses: [192.168.40.17/24]
      gateway4: 192.168.40.1
      nameservers:
        addresses: [8.8.8.8,8.8.4.4]
```

Save and apply your changes by running the command below:

```
sudo netplan apply
```

## 2.2.2. Client configuration

The Juju client needs two pieces of information before it can control our MAAS deployment.

1. A cloud definition for the MAAS deployment. This definition will include where MAAS can be found and how Juju can authenticate itself with it.
2. A separate credentials definition that's used when accessing MAAS. This links the authentication details to the cloud definition.

To create the cloud definition, type `juju add-cloud mymaas` to add a cloud called **mymaas**. This will produce output similar to the following:

```
Cloud Types
  maas
  manual
  openstack
  vsphere

Select cloud type:
```

Enter `maas` as the cloud type and you will be asked for the API endpoint URL. This URL is the same as the URL used to access the MAAS web UI in the previous step: **`http://<your.maas.ip>:5240/MAAS/`**.

With the endpoint added, Juju will inform you that **mymaas** was successfully added. The next step is to add credentials. This is initiated by typing `juju add-credential mymaas`. Enter `admin` when asked for a credential name. Juju will output the following:

```
Enter credential name: admin
Using auth-type "oauth1".
Enter maas-oauth:
```

The **oauth1** credential value is the MAAS API key for the **admin** user. To retrieve this, login to the MAAS web UI and click on the **admin** username near the top right. This will show the user preferences page. The top field will hold your MAAS keys:



Copy and paste this key into the terminal and press return. You will be informed that credentials have been added for cloud **mymaas**. You can check the cloud definition has been added with the `juju clouds` command, and you can list credentials with the `juju credentials` command.

## 2.3 2.3. Testing the environment

The Juju client now has everything it needs to instruct MAAS to deploy a Juju controller.

But before we move on to deploying OpenStack, it's worth checking that everything is working first. To do this, we'll simply ask Juju to create a new controller for our cloud:

```
juju bootstrap --constraints tags=juju mymaas maas-controller
```

The constraint in the above command will ask MAAS to use any nodes tagged with `juju` to host the controller for the Juju client. We tagged this node within MAAS in the *previous step*.

The output to a successful bootstrap will look similar to the following:

```
Creating Juju controller "maas-controller" on mymaas
Looking for packaged Juju agent version 2.4-alpha1 for amd64
Launching controller instance(s) on mymaas...
- 7cm8tm (arch=amd64 mem=48G cores=24)
Fetching Juju GUI 2.14.0
Waiting for address
Attempting to connect to 192.168.40.185:22
Bootstrap agent now started
Contacting Juju controller at 192.168.40.185 to verify accessibility...
```

(continues on next page)

(continued from previous page)

```
Bootstrap complete, "maas-controller" controller now available.  
Controller machines are in the "controller" model.  
Initial model "default" added.
```

If you're monitoring the nodes view of the MAAS web UI, you will notice that the node we tagged with **juju** starts deploying Ubuntu 18.04 LTS automatically, which will be used to host the Juju controller.

## 2.4 2.4. Opening the Juju GUI

Juju has a [graphical user interface \(GUI\)](#) available to help with the tasks of managing and monitoring your Juju environment. The GUI is a JavaScript and HTML web application that is encapsulated in its own charm. Once installed, the GUI will talk with Juju over a websocket to provide a real-time interface with the applications installed, the units that comprise them, and the machines available. Additionally, the GUI can talk with the charm store in order to search, browse, and deploy charms to your environment.

To view the URL and login credentials for Juju GUI, use the following command:

```
juju gui
```

The **username** and **password** will be displayed for log in Juju which will be something like this:

```
GUI 2.14.0 for model "admin/default" is enabled at:  
https://192.168.40.52:17070/gui/u/admin/default  
Your login credential is:  
username: admin  
password: 1e4e614eee21b2e1355671300927ca52
```

You have to open the GUI IP address in your browser and to copy and enter the **username** and **password** into the GUI:

---

**Tip:** If you don't want to copy and paste the URL manually, typing `juju gui --browser` will open the link in your default browser automatically.

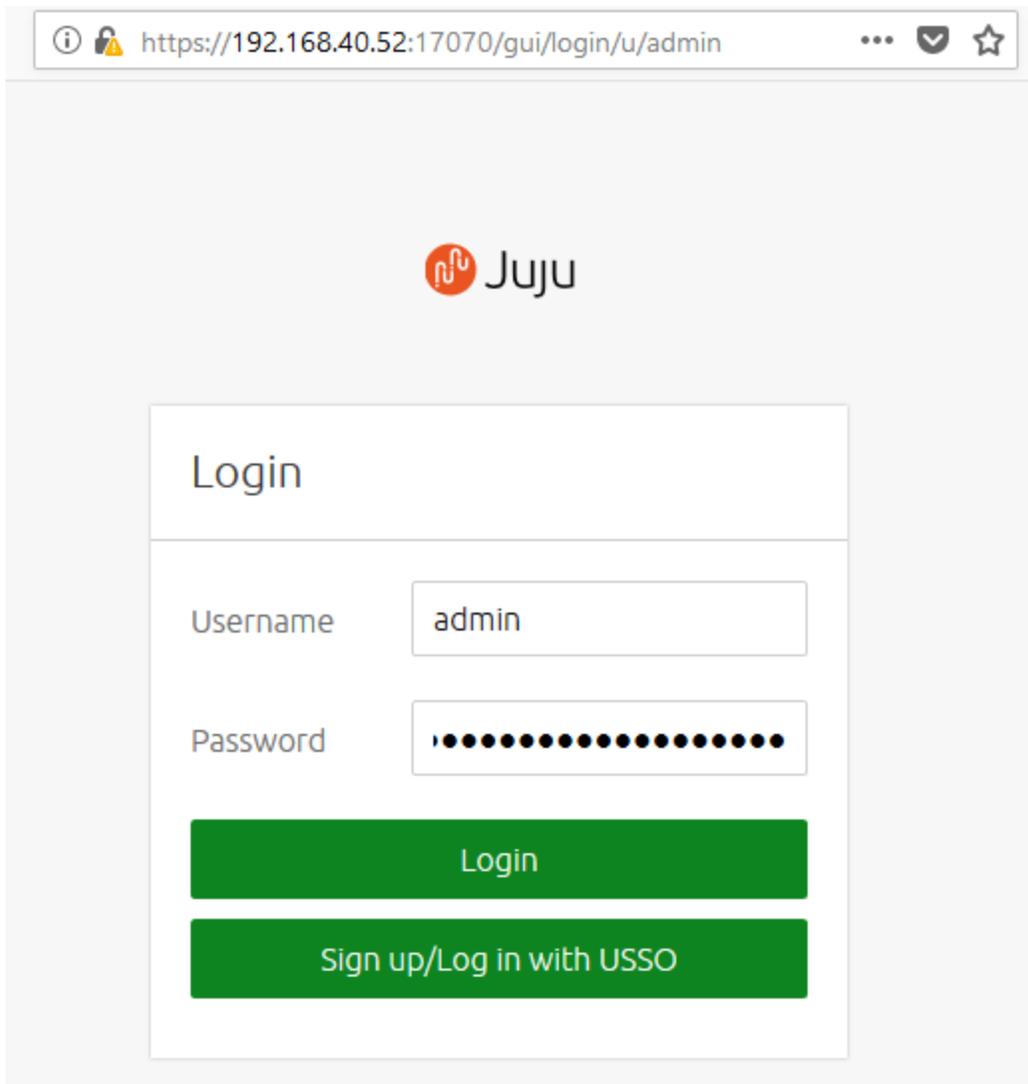
---

**Note:** If you'd rather not have your login credentials displayed in the output of `juju gui`, they can be suppressed by adding the `--hide-credential` argument.

---

## 2.5 2.5. Next steps

The *next step* will be to use Juju to deploy and link the various components required by OpenStack.





---

## 3. Install OpenStack

---

Prerequisites of an OpenStack open cloud platform deployment are as follows:

1. *Installed and configured MAAS.*
2. *Successfully deployed Juju controller.*

In general, there are two options for installing OpenStack:

1. Separate installation and configuration of individual OpenStack components and applications. This allows you control capabilities and a better understanding of the OpenStack deployment processes - you can keep track of exactly what MAAS and JuJu are doing.
2. The second option is to use ready-made **bundle**. A bundle is an encapsulation of a working deployment, including all configuration, resources and references. That allows you to deploy OpenStack with a single command or share that deployment.

---

**Important:** The installation of the OpenScan applications on the **EniWARE platform** is based on the second option - using **bundle** (see section 3.2. *Deploy OpenStack*).

---

---

**Note:** You can find more information at [Deploying OpenStack as a bundle](#) to learn about deploying as a bundle.

---

### 3.1 3.1. Juju controller deployment

In *section 2.3. Testing the environment*, we've demonstrated how you can deploy a new JuJu controller called **maas-controller** in order to test the environment (MAAS and Juju configuration).

- **JuJu controller operation status:**

To **check the operating state of the created JuJu controller**, use the command `juju status`. With the Juju controller running, the output will look similar to the following:

Model	Controller	Cloud/Region	Version	SLA	Timestamp
default	maas-controller	mymaas	2.4.4	unsupported	15:04:54+03:00

In case you need to remove the controller (called **maas-controller** in this case), use the following command:

```
juju kill-controller maas-controller
```

**Important:** In addition to the above command, the machine on which the **Juju controller** is located must be deleted from MAAS. To add the machine again, you can follow the instructions in section “1.9. Adding nodes” and do not forget to commission it as described in section “1.10. Commission nodes” (use tag *juju!*).

You can redeploy this JuJu controller with the following command:

```
juju bootstrap --constraints tags=juju mymaas maas-controller
```

During the bootstrap process, Juju will create a **model** called **default**, as shown in the output from `juju status` command above. **Models** act as containers for applications.

- **Create a new model:**

The next step is to **create a new model** called **test** that will be used for the purposes of **OpenStack deployment** exclusively, making the entire deployment easier to manage and maintain. To create a model called **test** (and switch to it), type the following command:

```
juju add-model test
```

After you add these model you can *log in to the Juju GUI*. To view the URL and login credentials for Juju GUI, use the following command:

```
juju gui
```

This will produce output similar to the following:

```
GUI 2.14.0 for model "admin/test" is enabled at:
  https://192.168.40.53:17070/gui/u/admin/test
Your login credential is:
  username: admin
  password: 67d4c5dbbb2c56990c3fdaab1d5a355c
```

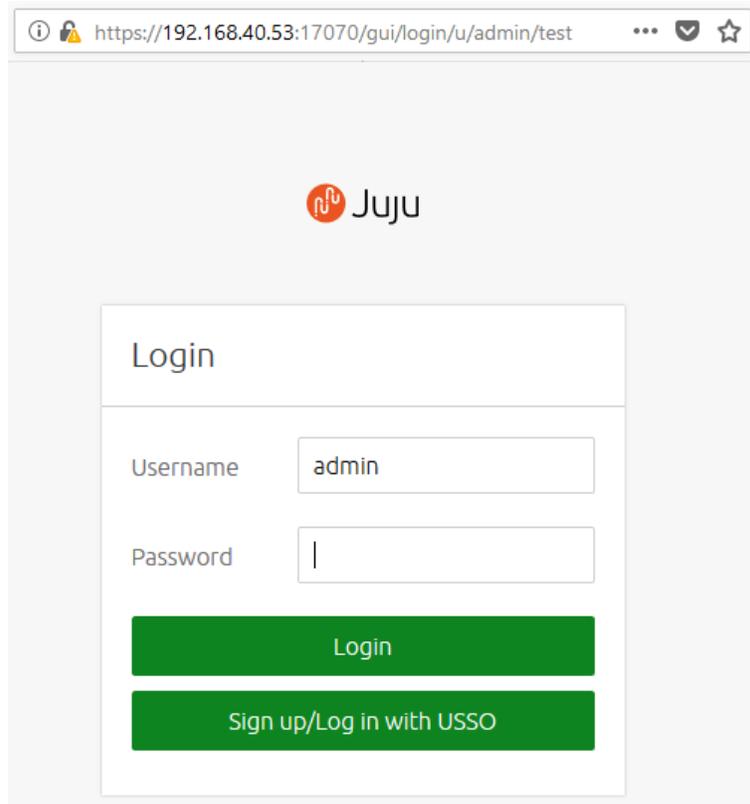
Open your browser at the specified IP address and enter the given login credentials:

## 3.2 3.2. OpenStack deployment

We are now going to step through adding the OpenStack components to the new model. The applications will be installed from the [eniware-org/openstack-bundles](https://github.com/eniware-org/openstack-bundles) repository. We'll be providing the configuration for the charms as a **yaml** file which we include as we deploy it.

After you Clone the repository to your Juju machine, go to folder `stable/openstack-base`. The configuration is held in the file called **bundle.yaml**. Deployment requires no further configuration than running the following command:

```
juju deploy bundle.yaml
```



**Warning:** Do not use autocomplete with **Tab** button.

To get the status of the deployment, run `juju status`. For constant updates, combine it with the `watch` command:

```
watch juju status
```

This will produce output similar to the following:

Model	Controller	Cloud/Region	Version	SLA	Timestamp		
test	maas-controller	mymaas	2.4.4	unsupported	16:23:02+03:00		
App	Store	Rev	OS	Version	Status	Scale	Charm
ceph-mon	jujucharms	26	ubuntu	13.2.1+dfsg1	waiting	2/3	ceph-mon
ceph-osd	jujucharms	269	ubuntu	13.2.1+dfsg1	blocked	3	ceph-osd
ceph-radosgw	jujucharms	259	ubuntu		maintenance	1	ceph-radosgw
cinder	jujucharms	273	ubuntu		waiting	0/1	cinder
cinder-ceph	jujucharms	234	ubuntu		waiting	0	cinder-ceph
glance	jujucharms	268	ubuntu		waiting	0/1	glance
keystone	jujucharms	283	ubuntu		maintenance	1	keystone

(continues on next page)

(continued from previous page)

mysql	5.7.20-29.24	active	1	percona-cluster	└	
↪jujucharms 269	ubuntu					
neutron-api		maintenance	1	neutron-api	└	
↪jujucharms 262	ubuntu					
neutron-gateway	13.0.1	waiting	1	neutron-gateway	└	
↪jujucharms 253	ubuntu					
neutron-openvswitch	13.0.1	waiting	3	neutron-openvswitch	└	
↪jujucharms 251	ubuntu					
nova-cloud-controller		waiting	0/1	nova-cloud-controller	└	
↪jujucharms 311	ubuntu					
nova-compute	18.0.1	waiting	3	nova-compute	└	
↪jujucharms 287	ubuntu					
ntp	4.2.8p10+dfsg	maintenance	4	ntp	└	
↪jujucharms 27	ubuntu					
openstack-dashboard		maintenance	1	openstack-dashboard	└	
↪jujucharms 266	ubuntu					
rabbitmq-server	3.6.10	active	1	rabbitmq-server	└	
↪jujucharms 78	ubuntu					
Unit	Workload	Agent	Machine	Public address	Ports	└
↪Message						
ceph-mon/0	maintenance	executing	1/lxd/0	192.168.40.110		└
↪(install) installing charm software						
ceph-mon/1	waiting	allocating	2/lxd/0			└
↪waiting for machine						
ceph-mon/2*	maintenance	executing	3/lxd/0	192.168.40.105		└
↪(install) installing charm software						
ceph-osd/0*	waiting	idle	1	192.168.40.58		└
↪Incomplete relation: monitor						
ceph-osd/1	blocked	idle	2	192.168.40.59		└
↪Missing relation: monitor						
ceph-osd/2	waiting	idle	3	192.168.40.101		└
↪Incomplete relation: monitor						
ceph-radosgw/0*	maintenance	executing	0/lxd/0	192.168.40.103		└
↪(install) Installing radosgw packages						
cinder/0	waiting	allocating	1/lxd/1			└
↪waiting for machine						
glance/0	waiting	allocating	2/lxd/1			└
↪waiting for machine						
keystone/0*	maintenance	executing	3/lxd/1	192.168.40.109		└
↪(install) installing charm software						
mysql/0*	active	idle	0/lxd/1	192.168.40.102	3306/tcp	└
↪Unit is ready						
neutron-api/0*	maintenance	executing	1/lxd/2	192.168.40.108		└
↪(install) installing charm software						
neutron-gateway/0*	waiting	idle	0	192.168.40.57		└
↪Incomplete relations: network-service, messaging						
ntp/0*	active	idle		192.168.40.57	123/udp	└
↪Ready						
nova-cloud-controller/0	waiting	allocating	2/lxd/2			└
↪waiting for machine						
nova-compute/0*	waiting	idle	1	192.168.40.58		└
↪Incomplete relations: image, messaging, storage-backend						
neutron-openvswitch/0*	waiting	idle		192.168.40.58		└
↪Incomplete relations: messaging						
ntp/1	active	idle		192.168.40.58	123/udp	└
↪Ready						

(continues on next page)

(continued from previous page)

```

nova-compute/1          waiting      executing    2          192.168.40.59
↳Incomplete relations: messaging, storage-backend, image
  neutron-openvswitch/2 maintenance  executing    192.168.40.59
↳(install) Installing apt packages
  ntp/3                 maintenance  executing    192.168.40.59
↳(install) installing charm software
nova-compute/2          waiting      executing    3          192.168.40.101
↳Incomplete relations: messaging, image, storage-backend
  neutron-openvswitch/1 maintenance  executing    192.168.40.101
↳(install) Installing apt packages
  ntp/2                 maintenance  executing    192.168.40.101
↳(install) installing charm software
openstack-dashboard/0* maintenance  executing    3/lxd/2    192.168.40.106
↳(install) installing charm software
rabbitmq-server/0*     active       executing    0/lxd/2    192.168.40.104
↳(config-changed) Enabling queue mirroring

Machine  State   DNS           Inst id           Series  AZ       Message
0        started 192.168.40.57 skyhk8            bionic  default  Deployed
0/lxd/0  started 192.168.40.103 juju-4052d2-0-lxd-0 bionic  default  Container_
↳started
0/lxd/1  started 192.168.40.102 juju-4052d2-0-lxd-1 bionic  default  Container_
↳started
0/lxd/2  started 192.168.40.104 juju-4052d2-0-lxd-2 bionic  default  Container_
↳started
1        started 192.168.40.58 t678hy           bionic  default  Deployed
1/lxd/0  started 192.168.40.110 juju-4052d2-1-lxd-0 bionic  default  Container_
↳started
1/lxd/1  pending                juju-4052d2-1-lxd-1 bionic  default  Container_
↳started
1/lxd/2  started 192.168.40.108 juju-4052d2-1-lxd-2 bionic  default  Container_
↳started
2        started 192.168.40.59 dsktgg           bionic  default  Deployed

```

The deployed **bundle.yaml** file includes the following applications:

<ul style="list-style-type: none"> <li>• <b>Openstack Dashboard</b> - it provides a Django based web interface for use by both administrators and users of an OpenStack Cloud. It allows you to manage Nova, Glance, Cinder and Neutron resources within the cloud.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Keystone</b> - this charm provides Keystone, the OpenStack identity service. Its target platform is (ideally) Ubuntu LTS + OpenStack.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Glance</b> - The Glance project provides an image registration and discovery service and an image delivery service. These services are used in conjunction by <b>Nova</b> to deliver images from object stores, such as OpenStack's Swift service, to Nova's compute nodes.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>MySQL</b> - Percona XtraDB Cluster is a high availability and high scalability solution for MySQL clustering. Percona XtraDB Cluster integrates Percona Server with the Galera library of MySQL high availability solutions in a single product package which enables you to create a cost-effective MySQL cluster. This charm deploys Percona XtraDB Cluster onto Ubuntu.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Cinder</b> - Cinder is the block storage service for the OpenStack. This charm provides the Cinder volume service for OpenStack. It is intended to be used alongside the other OpenStack components. Cinder is made up of 3 separate services: an API service, a scheduler and a volume service. This charm allows them to be deployed in different combination, depending on user preference and requirements.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Cinder Ceph</b> - This charm provides a Ceph storage backend for <b>Cinder</b> charm. This allows multiple Ceph storage clusters to be associated with a single Cinder deployment, potentially alongside other storage backends from other vendors.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>RabbitMQ</b> - RabbitMQ is an implementation of AMQP, the emerging standard for high performance enterprise messaging. The RabbitMQ server is a robust and scalable implementation of an AMQP broker. This charm deploys RabbitMQ server and provides AMQP connectivity to clients.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Nova Compute</b> - this charm is a cloud computing fabric controller which provides the OpenStack compute service. This charm provides the Nova Compute hypervisor service and should be deployed directly to physical servers. Its target platform is Ubuntu (preferably LTS) + OpenStack.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Ceph OSD</b> - Ceph is a distributed storage and network file system designed to provide excellent performance, reliability, and scalability. This charm deploys additional Ceph OSD storage service units and should be used in conjunction with the <b>Ceph-mon</b> charm to scale out the amount of storage available in a Ceph cluster.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Ceph Mon</b> - This charm deploys a Ceph monitor cluster.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Ceph Radosgw</b> - This charm provides the RADOS HTTP gateway supporting S3 and Swift protocols for object storage.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>Neutron API</b> - Neutron is a virtual network service for OpenStack. Neutron provides an API to dynamically request and configure virtual networks. These networks connect "interfaces" from other OpenStack services (e.g., virtual NICs from Nova VMs). The Neutron API supports extensions to provide advanced network capabilities (e.g., QoS, ACLs, network monitoring, etc.). This principle charm provides the OpenStack Neutron API service which was previously provided by the <b>Nova-cloud-controller</b> charm. When this charm is related to the Nova-cloud-controller charm the Nova-cloud controller charm will shutdown its api service, de-register it from Keystone and inform the compute nodes of the new Neutron url.</li> </ul>

<ul style="list-style-type: none"> <li>• <b>Nova Cloud Controller</b> - OpenStack Compute, codenamed Nova, is a cloud computing fabric controller. This charm provides the cloud controller service for OpenStack Nova and includes <b>nova-scheduler</b>, <b>nova-api</b> and <b>nova-conductor</b> services.</li> </ul>
---

**Note:** Remember, you can check on the status of a deployment using the `juju status` command. To see the status of a single charm of application, append the charm name. For example, for a Ceph OSD charm:

```
juju status ceph-osd
```

### 3.3.3. OpenStack testing

After everything has deployed and the output of `juju status` settles, you can check to make sure OpenStack is working by logging into the Horizon Dashboard.

The quickest way to get the IP address for the Dashboard is with the following command:

```
juju status --format=yaml openstack-dashboard | grep public-address | awk '{print $2}'
```

he following commands may alternatively be used:

- to get the IP address for the OpenStack Dashboard:

```
juju status | grep dashboard
```

- to get the IP address for the OpenStack Keystone node for authentication.:

```
juju status | grep keystone
```

The OpenStack Dashboard URL will be **http://<IP ADDRESS>/horizon**. When you enter this into your browser you will need a login domain, username and a password. The **admin login domain** is **admin\_domain**. To login with *user admin* you will need a *password* that can be called with the following command:

```
juju run --unit keystone/0 leader-get admin_passwd
```

If everything works, you will see something similar to the following:

The screenshot shows the OpenStack Horizon dashboard interface. The top navigation bar includes the Ubuntu logo, the current user 'admin' in the 'admin\_domain', and a user profile icon. The left sidebar contains a navigation menu with options: Project, Admin, Identity, Domains, Projects (selected), Users, Groups, Roles, and Application Credentials. The main content area is titled 'Projects' and includes a search filter for 'Project Name', a 'Filter' button, and buttons for '+ Create Project' and 'Delete Projects'. Below the filters, it states 'Displaying 1 item' and shows a table with the following data:

<input type="checkbox"/>	Name	Description	Project ID	Domain Name	Enabled	Actions
<input type="checkbox"/>	admin	Created by Juju	b6502a607037460498a3b8b91d8f139a	admin_domain	Yes	<a href="#">Manage Members</a>

Below the table, it again states 'Displaying 1 item'.

## 3.4 3.4. Next steps

With this final step you've successfully deployed a working OpenStack environment using both Juju and MAAS. The next step is to *configure OpenStack* for use within a production environment.

---

## 4. Configure OpenStack

---

In previous sections we've deployed OpenStack using both Juju and MAAS. The next step is to configure OpenStack for use within a production environment.

The steps to be followed are:

- Installing a client for OpenStack Nova API (the **novaclient** module)
- Installing a client for OpenStack Identity API (the **keystoneclient** modules)
- Installing a client for OpenStack Images API (the **glanceclient** module)
- Installing a client for OpenStack Networking API (the **neutronclient** module)
- Installing a command-line client for OpenStack (the **OpenStackClient** - OSC)
- Setting up the *environment variables*
- Creating the necessary *flavors*
- Adding a *domain, project and user*
- Managing *quotas*
- *External network* access and Ubuntu *cloud image* deployment

Some of the procedures can be made either from the web UI (Horizon) or from the command line interface (CLI).

### 4.1 4.1. Installing clients for different OpenStack operations

To install the clients for OpenStack Nova API, OpenStack Identity API, OpenStack Images API, OpenStack Networking API and OpenStackClient, execute the following commands:

```
sudo add-apt-repository cloud-archive:rocky -y
sudo apt-get update
sudo apt-get install python-novaclient python-keystoneclient python-glanceclient_
↪python-neutronclient python-openstackclient -y
```

## 4.2 4.2. Environment variables

When accessing OpenStack from the command line, specific environment variables need to be set:

- OS\_AUTH\_URL
- OS\_USER\_DOMAIN\_NAME
- OS\_USERNAME
- OS\_PROJECT\_DOMAIN\_NAME
- OS\_PROJECT\_NAME
- OS\_PROJECT\_NAME
- OS\_REGION\_NAME
- OS\_IDENTITY\_API\_VERSION
- OS\_AUTH\_VERSION

The OS\_AUTH\_URL is the address of the OpenStack Keystone node for authentication. To retrieve this IP address by Juju use the following command (or as shown [here](#)):

```
juju status --format=yaml keystone/0 | grep public-address | awk '{print $2}'
```

The specific environment variables are included in a file called **openrc**. It is located in `openstack_bundles/stable/openstack-base` or you can [download](#) it. It can easily be sourced (made active).

The **openrc** file contains the following:

```
_OS_PARAMS=$(env | awk 'BEGIN {FS="="} /^OS_/ {print $1;}' | paste -sd ' ')
for param in $_OS_PARAMS; do
    if [ "$param" = "OS_AUTH_PROTOCOL" ]; then continue; fi
    if [ "$param" = "OS_CACERT" ]; then continue; fi
    unset $param
done
unset _OS_PARAMS

_keystone_unit=$(juju status keystone --format yaml | \
    awk '/units:$/ {getline; gsub(/:$/, ""); print $1}')
_keystone_ip=$(juju run --unit ${_keystone_unit} 'unit-get private-address')
_password=$(juju run --unit ${_keystone_unit} 'leader-get admin_passwd')

export OS_AUTH_URL=${OS_AUTH_PROTOCOL:-http}://${_keystone_ip}:5000/v3
export OS_USERNAME=admin
export OS_PASSWORD=${_password}
export OS_USER_DOMAIN_NAME=admin_domain
export OS_PROJECT_DOMAIN_NAME=admin_domain
export OS_PROJECT_NAME=admin
export OS_REGION_NAME=RegionOne
export OS_IDENTITY_API_VERSION=3
# Swift needs this:
export OS_AUTH_VERSION=3
# Gnocchi needs this
export OS_AUTH_TYPE=password
```

The environment variables can be enabled/sourced with the following command:

```
source openstack-bundles/stable/openstack-base/openrc
```

After the **openrc** is created, you can use OpenStack’s Horizon web UI to download the file, which automatically adjusts the environment variables. You should be logged with the given *username*. Right click on the *user* dropdown menu in the the upper right corner and select **openrc -v3** from the list.

**Note:** If the **openrc** file is manually edited, it is important that all variables are correctly entered.

You can check the variables have been set correctly by seeing if your OpenStack endpoints are visible with the `openstack endpoint list` command. The output will look something like this:

```

+-----+-----+-----+-----+
| ID                | Region  | Service Name | Service Type |
+-----+-----+-----+-----+
| 060d704e582b4f9cb432e9ecbf3f679e | RegionOne | cinderv2     | volumev2     |
| 269fe0ad800741c8b229a0b305d3ee23 | RegionOne | neutron      | network      |
| 3ee5114e04bb45d99f512216f15f9454 | RegionOne | swift        | object-store  |
| 68bc78eb83a94ac48e5b79893d0d8870 | RegionOne | nova         | compute      |
| 59c83d8484d54b358f3e4f75a21dda01 | RegionOne | s3           | s3           |
| bebd70c3f4e84d439aa05600b539095e | RegionOne | keystone     | identity     |
| 1eb95d4141c6416c8e0d9d7a2eed534f | RegionOne | glance       | image        |
| 8bd7f4472ced40b39a5b0ecce29df3a0 | RegionOne | cinder       | volume       |
+-----+-----+-----+-----+

```

If the endpoints aren’t visible, it’s likely your environment variables aren’t configured correctly.

**Hint:** As with both MAAS and Juju, most OpenStack operations can be accomplished using either the command line or a web UI.

### 4.3 4.3. Define an external network

To allow OpenStack network access, it is necessary to enter external network settings.

You should be logged as an *user admin* in the OpenStack Dashboard Horizon. To do this, you need to know the following:

- the IP address for OpenStack Dashboard
- the user credential (*domain, user name and password*)

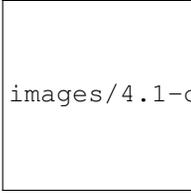
#### 4.3.1 Define an external network using web UI:

Using the commands shown in section “3.3. OpenStack testing” log in to the Dashboard with the following:

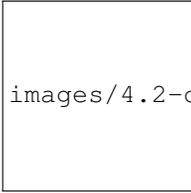
- Dashboard IP address: **192.168.40.145**
- Domain: **admin\_domain**
- User Name: **admin**
- Password: **your\_password**

First step is to define a network called **ext\_net**. It will use a subnet within the *range of addresses* reserved in MAAS.

From the panel on the left, click on **Admin** and choose section **Network**, subsection **Networks**. Then press the button **+ Create Network**:



images/4.1-cfconfig\_horizon.png



images/4.2-cfconfig\_net\_create.png

After opening the **Create network** window, you should enter the following settings:

- Name: **ext\_net**
- Project: **admin**
- Network type: **flat**
- Physical network: **physnet1**
- Marked checkboxes **Enable Admin State**, **Shared**, **External Network** and **Create Subnet**



images/4.3-cfconfig\_net\_settings.png

The second step is to create a subnet for the network using the various addresses from our MAAS and Juju configuration:

- Subnet Name: **ext\_net\_subnet**
- Network address (the network address where OpenStack is deployed): **192.168.40.0/24**
- IP Version: **IPv4**
- Gateway IP: **192.168.40.1**

In the **Subnet details** tab it is important do unmark the **Enable DHCP** checkbox. An **Allocation Pools** should be defined (in format: *start\_IP\_address, end\_IP\_address*) as well as **DNS Name Servers** (on the first line: *the IP address of the MAAS server*, which in this case is **192.168.40.16** - see section “[1.2. Installation](#)”), on the second line: **the DNS used to resolve domains not managed by MAAS** which in this case is **8.8.8.8** - see section “[1.8. Network services](#)”):

### 4.3.2 Define an external network using CLI:

To define a network called **ext\_net** type the following command:

```
openstack network create ext_net --share --external
```

The output from this command will show the various fields and values for the chosen configuration option. To show the new network ID alongside its name type the command `openstack network list`:

images/4.4-cfconfig\_net\_subnet.png

images/4.5-cfconfig\_net\_subdetails.png

ID	Name	Subnets
fc171d22-d1b0-467d-b6fa-109dfb77787b	ext_net	

To create a subnet for the network using the various addresses from our MAAS and Juju configuration type the following command:

```
openstack subnet create ext_net_subnet --allocation-pool \
start=192.168.40.191,end=192.168.40.254 --subnet-range 192.168.40.0/24 \
--no-dhcp --gateway 192.168.40.1 --dns-nameserver 192.168.40.16 \
--dns-nameserver 8.8.8.8 --network ext_net
```

The output from the previous command provides a comprehensive overview of the new subnet's configuration:

Field	Value
allocation_pools	192.168.40.191-192.168.40.254
cidr	192.168.40.0/24
created_at	2019-01-04T13:43:48
description	
dns_nameservers	192.168.40.16, 8.8.8.8
enable_dhcp	<b>False</b>
gateway_ip	192.168.40.1
host_routes	
id	563ecd06-bbc3-4c98-b93e
ip_version	4
ipv6_address_mode	<b>None</b>
ipv6_ra_mode	<b>None</b>
name	ext_net_subnet
network_id	fc171d22-d1b0-467d-b6fa-109dfb77787b
project_id	4068710688184af997c1907137d67c76
revision_number	<b>None</b>
segment_id	<b>None</b>
service_types	<b>None</b>
subnetpool_id	<b>None</b>
updated_at	2019-01-04T13:43:48
use_default_subnet_pool	<b>None</b>

**Note:** OpenStack has [deprecated](#) the use of the **neutron** command for network configuration, migrating most of its functionality into the Python OpenStack client. Version 2.4.0 or later of this client is needed for the `subnet create` command.

---

### 4.4 4.4. Cloud images

You need to download an **Ubuntu image** locally in order to be able to dd it to a **Glance**. Canonical's Ubuntu cloud images can be found here:

<https://cloud-images.ubuntu.com>

You could use `wget` to download the image of **Ubuntu 18.04 LTS (Bionic)**:

```
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
```

To add this image to Glance use the following command:

```
openstack image create --public --min-disk 3 --container-format bare \
--disk-format qcow2 --property architecture=x86_64 \
--property hw_disk_bus=virtio --property hw_vif_model=virtio \
--file bionic-server-cloudimg-amd64.img \
"bionic x86_64"
```

Typing `openstack image list` you can make sure the image was successfully imported:

```
+-----+-----+-----+
| ID                | Name                | Status |
+-----+-----+-----+
| d4244007-5864-4a2d-9cfd-f008ade72df4 | bionic x86_64      | active |
+-----+-----+-----+
```

The **Compute > Images** page of **OpenStack's Horizon web UI** lists many more details about imported images. In particular, note their size as this will limit the minimum root storage size of any OpenStack flavours used to deploy them.

### 4.5 4.5. Working with flavors

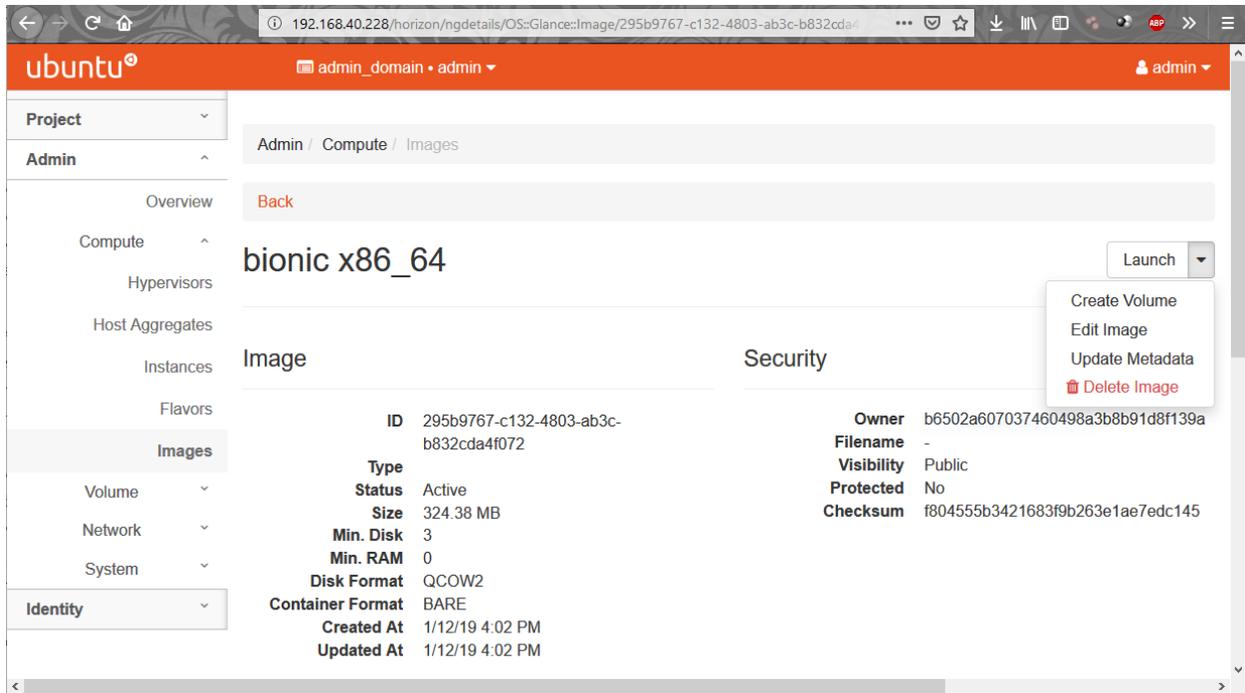
The **flavors** define the compute, memory, and storage capacity of nova computing instances. A **flavor** is an available hardware configuration for a server. It defines the size of a virtual server that can be launched.

---

**Hint:** For information on the flavors and flavor extra specs, refer to [Flavors](#).

---

The following flavors should be created:



Name	CPUs	RAM (MiB)	Root (GiB)	Disk	Ephemeral Disk (GiB)
minimal	1	3840	3		10
small	2	7680	3		14
small-50GB-ephemeral-disk	2	7680	3		50
small-highmem	4	31232	3		10
small-highmem-100GB-ephemeral-disk	4	31232	3		100
m1.xlarge	8	16384	160		0

#### 4.5.1 Working with flavors using web UI:

From the panel on the left, click on **Admin** and choose section **Compute**, subsection **Flavors**. Then press the button **+ Create Flavor**.

After opening the **Create Flavor** window, you should enter the following settings (for **m1.xlarge** flavor, for example):

- Name: **m1.xlarge**
- ID: *auto*
- VCPUs: **8**
- RAM (MB): **16384**
- Root Disk (GB): **160**
- Ephemeral Disk (GB): **0**
- Swap Disk (MB): **0**
- RX/TX Factor: **1**

In the **Flavor Access** tab select the project where the created flavor will be used:

### Create Flavor ✕

Flavor Information \* Flavor Access

**Name** \*

**ID** ?

**VCPUs** \*

**RAM (MB)** \*

**Root Disk (GB)** \*

**Ephemeral Disk (GB)**

**Swap Disk (MB)**

**RX/TX Factor**

Flavors define the sizes for RAM, disk, number of cores, and other resources and can be selected when users deploy instances.

**Note:** If no projects are selected, then the flavor will be available in all projects.

Click the **Create Flavor** button to save changes.

#### 4.5.2 Working with flavors using CLI:

Admin users can use the `openstack flavor` command to [create, customize and manage flavor](#).

To create a flavor using an `openstack flavor create` command, you should specify the following parameters:

- flavour name
- ID
- RAM size
- disk size
- the number of vCPUs for the flavor

For the purpose of OpenStack configuration and CloudFoundry deployment, you need to create flavors with the following names and configuration:

```
openstack flavor create --vcpus 1 --ram 3840 --disk 3 --ephemeral 10 minimal
openstack flavor create --vcpus 2 --ram 7680 --disk 3 --ephemeral 14 small
openstack flavor create --vcpus 2 --ram 7680 --disk 3 --ephemeral 50 small-50GB-
↪ephemeral-disk
openstack flavor create --vcpus 4 --ram 31232 --disk 3 --ephemeral 10 small-highmem
openstack flavor create --vcpus 4 --ram 31232 --disk 3 --ephemeral 100 small-highmem-
↪100GB-ephemeral-disk
openstack flavor create --vcpus 8 --ram 16384 --disk 160 --ephemeral 0 ml.xlarge
```

To list the created flavors and show the ID and name, the amount of memory, the amount of disk space for the root partition and for the ephemeral partition, the swap, and the number of virtual CPUs for each flavor, type the command:

```
openstack flavor list
```

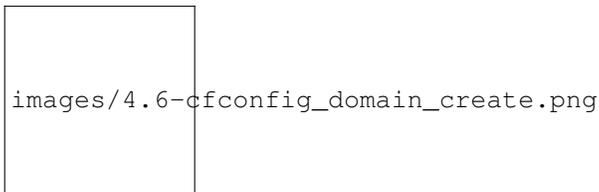
## 4.6 4.6. Working with domains, projects and users

The following is vital part of OpenStack operations:

- **Domains** - abstract resources; a domain is a collection of users and projects that exist within the OpenStack environment.
- **Projects** - organizational units in the cloud to which you can assign users (a project is a group of zero or more users).
- **Users** - members of one or more projects.
- **Roles** - define which actions users can perform. You assign roles to user-project pairs.

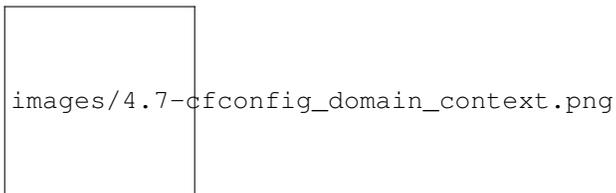
### 4.6.1 Working with domains and projects using web UI:

To create a **domain** using Dashboard, click on **Identity** from the panel on the left and choose section **Domains**. Then press the button + **Create Domain**:



You need to create domain with name **cf\_domain**.

After the **cf\_domain** is created you need to locate it in the table with domains and press the corresponding button **Set Domain Context** from the **Actions** column. In this way, all subsequent operations will be executed in the context of this domain.



To create a **Project** in the context of **cf\_domain** domain click on **Identity** from the panel on the left and choose section **Projects**. Then press the button + **Create Project** and enter the name **cloudfoundry** for this new project:



To create a **User** with a *role member* of **cloudfoundry project**, click on **Identity** from the panel on the left and choose section **Users**. Then press the button + **Create User** and enter the name **eniware** for the **User Name**:

## Create User



### Domain ID

### Domain Name

### User Name \*

### Description

### Email

### Password \*

### Confirm Password \*

### Primary Project

### Role

  Enabled

### Description:

Create a new user and set related properties including the Primary Project and Role.

You should specify a **password** *your\_password* for this user.

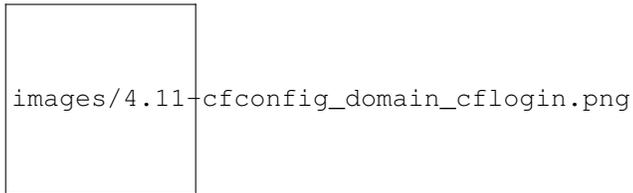
After the **project** and **user** are created, you should go back into **Identity / Domains** section and press the button **Clear Domain Context** to complete the execution of procedures in the context of **cf\_domain**:



The final step is to log out user **admin\_domain** from the Dashboard.

Now you can log in to Dashboard with the created domain **cf\_domain**:

- Domain: **cf\_domain**
- User: **eniware**
- Password: *your\_password*



### 4.6.2 Working with domains and projects using CLI:

To create a single *domain* with a single *project* and single *user* for a new deployment, start with the *domain*:

```
openstack domain create cf_domain
```

To add a *project* **cloudfoundry** to the *domain* **cf\_domain**:

```
openstack project create cloudfoundry --domain cf_domain
```

To add a *user* **eniware** with a *role* **member** and assign that user to the *project* **cloudfoundry** within **cf\_domain**:

```
openstack user create eniware --domain cf_domain --project cloudfoundry --password_  
↪your_password  
openstack role add --project cloudfoundry --project-domain cf_domain --user eniware --  
↪user-domain cf_domain Member
```

The output to the previous command will be similar to the following:

```
+-----+-----+  
| Field          | Value                                     |  
+-----+-----+  
| default_project_id | 914e59223944433dbf12417ac4cd4031 |  
| domain_id        | 7993528e51344814be2fd53f1f8f82f9 |  
| enabled          | True                                  |  
| id               | e980be28b20b4a2190c41ae478942ab1 |  
| name             | cf_domain                               |  
+-----+-----+
```

(continues on next page)

(continued from previous page)

```

| options          | | {}          |
| password_expires_at | None        |
+-----+-----+

```

Every subsequent action will now be performed by **eniware** user within the new **cf\_project** project.

## 4.7 4.7. View and manage quotas

To prevent system capacities from being exhausted without notification, you can set up **quotas**. Quotas are operational limits. The *Compute* and *Block Storage service* quotas are described [here](#).

### 4.7.1 View and manage quotas using web UI:

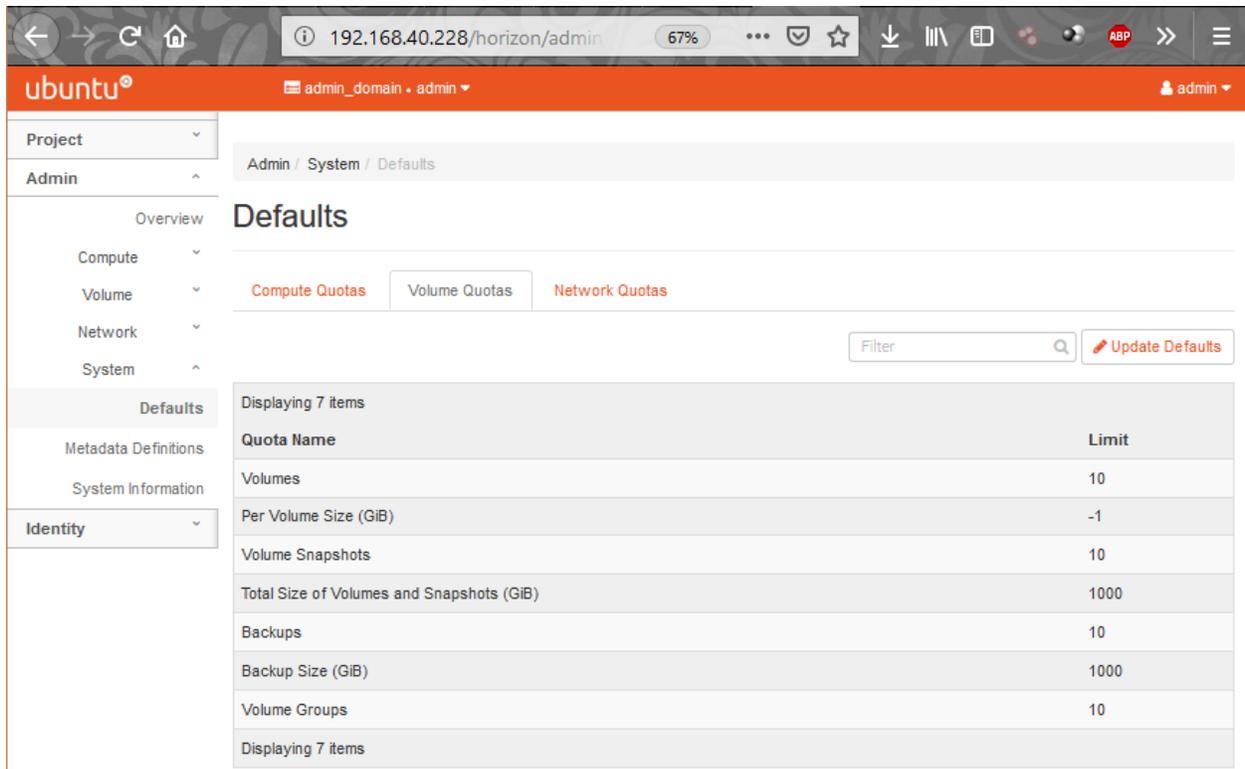
Log in to the Dashboard and select the **admin project** from the drop-down list. On the **Admin** tab, open the **System** tab and click the **Defaults** category. The default quota values are displayed:

- Compute Quotas:

The screenshot shows the OpenStack Horizon web interface. The browser address bar indicates the URL is 192.168.40.228/horizon/admin. The page title is 'Admin / System / Defaults'. The left sidebar shows the navigation menu with 'System' expanded and 'Defaults' selected. The main content area displays 'Compute Quotas' with a table of default values.

Quota Name	Limit
VCPUs	20
Injected File Content Bytes	10240
Length of Injected File Path	255
Injected Files	5
Instances	10
Key Pairs	100
Metadata Items	128
RAM (MB)	51200
Server Group Members	10
Server Groups	10

- Volume Quotas:
- Network Quotas:



To update project quotas click the **Update Defaults** button. In the **Update Default Quotas** window, you can edit the default quota values. Click the **Update Defaults** button to save changes.

**Note:** Network quotas can not be edited in this way because they depend on the *network settings* that are configured

## 4.7.2 View and manage quotas using CLI:

The dashboard does not show all possible project quotas. To view and update the quotas for a service, you can use OpenStackClient CLI.

To list all default quotas for all projects use the following command:

```
openstack quota show --default
```

To list the currently set quota values for a **cloudfoundry** project use the following command:

```
openstack quota show cloudfoundry
```

To update quota values for a given existing project:

```
openstack quota set --QUOTA_NAME QUOTA_VALUE PROJECT_OR_CLASS
```

To update quotas for **cloudfoundry** project use the following commands:

The screenshot shows the OpenStack Horizon admin interface. The browser address bar displays '192.168.40.228/horizon/admin'. The page title is 'Admin / System / Defaults'. The left sidebar contains a navigation menu with categories: Project, Admin, Overview, Compute, Volume, Network, System, Defaults (selected), Metadata Definitions, System Information, and Identity. The main content area is titled 'Defaults' and has three tabs: 'Compute Quotas', 'Volume Quotas', and 'Network Quotas'. A search filter box is present. Below the tabs, a table displays 18 items with columns for 'Quota Name' and 'Limit'.

Quota Name	Limit
Networks	10
Subnets	10
Subnet Pool	-1
Ports	50
Routers	10
Floating IPs	50
RBAC Policies	10
Security Groups	10
Security Group Rules	100
Firewall Rule	100
Firewall Policy	10
Firewall	10
L7Policy	-1
Member	-1

```
openstack quota set --instances 100 --cores 96 --ram 153600 --key-pairs 100_
↳cloudfoundry
openstack quota set --volumes 100 --per-volume-gigabytes 500 --gigabytes 4096_
↳cloudfoundry
openstack quota set --secgroup-rules 100 --secgroups 100 --networks 500 --subnets_
↳1000 --ports 2000 --routers 1000 --vips 100 --subnetpools 500 cloudfoundry
```

- The first command will update the the OpenStack *Compute service* quota **instances** - number of instances or amount of CPU that a for **cloudfoundry** *project* can use.
- The second one will update the OpenStack *Block Storage service* quotas - **volumes** allowed for the project.
- The third command will update the the OpenStack *Compute service* quotas - **security group rules** allowed for the project.

### 4.8 4.8. Next steps

You have now successfully deployed and configured OpenStack, taking full advantage of both Juju and MAAS. The next step is to *deploy CloudFoundry with BOSH Director on OpenStack*.

---

## 5. Deploying CloudFoundry with BOSH Director on OpenStack

---

---

**Todo:** Draft: to be deleted after the documentation is ready:

1. Configure OpenStack Domain, Project, User, Network for the deployment - Done in Section “4. *Configure OpenStack*”
  2. Validate the OpenStack configuration using: <https://github.com/eniware-org/cf-openstack-validator> :
    - cf-openstack-validator installation done in section 5.2. *CF-OpenStack-Validator installation*
  3. Setup the OpenStack projects for the BOSH and CloudFoundry installation using TerraForm modules from here: <https://github.com/eniware-org/bosh-openstack-environment-templates>
  4. Install BOSH: <https://bosh.io/docs/init-openstack/#deploy>
    5. Prepare and upload `cloud-config.yml` to BOSH to finalize the cloud configuration.
    6. Deploy CloudFoundry.
- 

In *previous section* we’ve configured OpenStack for use within a production environment. Various types of *clients* were installed for different OpenStack operations. We have set *environment variables*, *external network*, *flavors*, *domain*, *project and user*.

In this section we’ll create an environment that consists of a BOSH Director and Cloudfoudry deployment that it orchestrates.

### 5.1 5.1. Prerequisites

To be able to proceed, you need to the following:

- *Working OpenStack environment* using both Juju and MAAS.
- A *user* able to create/delete resource in this environment.
- *Flavors* with properly configured names and settings.

## 5.2 5.2. CF-OpenStack-Validator installation

CF OpenStack Validator is an extension that verifies whether the OpenStack installation is ready to run BOSH and install Cloud Foundry. The intended place to run the validator is a VM within your OpenStack.

### 5.2.1 5.2.1. Prerequisites for CF-OpenStack-Validator

OpenStack configuration requirements are as follows:

- *Keystone* v.2/v.3 Juju charm installed.
- Created *OpenStack project*.
- Created *user* with access to the previously created *project* (ideally you don't want to run as admin).
- Created network - connect the network with a router to your external network.
- Allocated a floating IP.
- Allowed ssh access in the *default* security group - create a key pair by executing:

```
$ ssh-keygen -t rsa -b 4096 -N "" -f cf-validator.rsa_id
```

Upload the generated public key to OpenStack as **cf-validator**.

- A public *image* available in **Glance**.

The validator runs on Linux. Please ensure that the following packages are installed on your system:

- *ruby* 2.4.x or newer
- *make*
- *gcc*
- *zlib1g-dev*
- *libssl-dev*
- *ssh*

### 5.2.2 5.2.2. Installation of CF-OpenStack-Validator

To clone the CF-OpenStack-Validator repository:

```
git clone https://github.com/eniware-org/cf-openstack-validator
```

Navigate to the **cf-openstack-validator** folder:

```
cd cf-openstack-validator
```

Copy the *generated private key* into the **cf-openstack-validator** folder.

Copy *validator.template.yml* to *validator.yml* and replace occurrences of **<replace-me>** with appropriate values (see *prerequisites*):

- If using Keystone v.3, ensure there are values for *domain* and *project*.
- If using Keystone v.2, remove *domain* and *project*, and ensure there is a value for *tenant*. Also use the Keystone v.2 URL as *auth\_url*.

```
$ cp validator.template.yml validator.yml
```

Download a **stemcell** from [OpenStack stemcells bosh.io](https://bosh.io/d/stemcells/bosh-openstack-kvm-ubuntu-trusty-go_agent):

```
$ wget --content-disposition https://bosh.io/d/stemcells/bosh-openstack-kvm-ubuntu-  
↪trusty-go_agent
```

Install the following dependencies:

```
$ gem install bundler  
$ bundle install
```

### 5.2.3 5.2.3. Additional configurations

- **CPI:**

Validator downloads **CPI** release from the URL specified in the validator configuration. You can override this by specifying the `--cpi-release` command line option with the path to a CPI release tarball.

If you already have a CPI compiled, you can specify the path to the executable in the environment variable `OPENSTACK_CPI_BIN`. This is used when no CPI release is specified on the command line. It overrides the setting in the validator configuration file.

- **Extensions:**

You can extend the validator with custom tests. For a detailed description and examples, please have a look at the [extension documentation](#).

The [eniware-org repository](#) already contains some extensions. Each extension has its own documentation which can be found in the corresponding extension folder.

To learn about available cf-validator options run the following command:

```
$ ./validate --help
```

You can find more additional OpenStack related configuration options for possible solutions [here](#).

## 5.3 5.3. Validate the OpenStack configuration

Before deploying Cloud Foundry, make sure to successfully run the [CF-OpenStack-Validator](#) against your project:

- Make sure you have the *required flavors* on OpenStack by enabling the [flavors extension](#) with the `flavors.yml` file in this directory. Flavor names need to match those specified in the cloud config.
- If you plan using the [Swift ops file](#) to enable Swift as blobstore for the Cloud Controller, you should also run the [Swift extension](#).

To **start the validation process** type the following command:

```
$ ./validate --stemcell bosh-stemcell-<xxx>-openstack-kvm-ubuntu-trusty-go_agent.tgz -  
↪-config validator.yml
```

## 5.4 5.4. Prepare OpenStack environment for BOSH and Cloud Foundry via Terraform

You can use a **Terraform environment template** to configure your OpenStack project automatically. You will need to create a `terraform.tfvars` file with information about the environment.

---

**Important:** The terraform scripts will output the OpenStack resource information required for the BOSH manifest. Make sure to treat the created `terraform.tfstate` files with care.

---

---

**Hint:** Instead of using Terraform, you can prepare an OpenStack environment **manually** as described [here](#).

---

### 5.4.1 5.4.1. Install Terraform module

Make sure you have updated package database and installed unzip package:

```
sudo apt-get update
sudo apt-get install -y git unzip
```

To install the **Terraform** module:

```
git clone https://github.com/eniware-org/bosh-openstack-environment-templates.git
wget https://releases.hashicorp.com/terraform/0.11.11/terraform_0.11.11_linux_amd64.
↪zip
unzip terraform_0.11.11_linux_amd64.zip
chmod +x terraform
sudo mv terraform /usr/local/bin/
```

### 5.4.2 5.4.2. OpenStack environment for BOSH

#### 5.4.2.1 Setup an OpenStack project to install BOSH:

To setup an OpenStack project to install BOSH please use the following [Terraform module](#). Adapt `terraform.tfvars.template` to your needs.

1. Create a working folder:

```
mkdir tmp
```

2. Copy the *template file*:

```
cp bosh-openstack-environment-templates/bosh-init-tf/terraform.tfvars.
↪template tmp/terraform.tfvars
```

3. Generate a key pair executing the following script:

```
sh bosh-openstack-environment-templates/bosh-init-cf/generate_ssh_keypair.sh
```

4. Move the generated key pair to your working folder `tmp`:

```
mv bosh.* tmp/
```

5. Navigate to the working folder `tmp`:

```
cd tmp
```

6. Configure the *Terraform environment template* `terraform.tfvars`.

7. Run the following commands:

```
terraform init ../bosh-openstack-environment-templates/bosh-init-tf/
terraform apply ../bosh-openstack-environment-templates/bosh-init-tf/
```

8. Save the `terraform.tfvars` and `terraform.tfstate` files for **bosh-init-tf**:

```
mv terraform.tfvars bosh_terraform.tfvars
mv terraform.tfstate bosh_terraform.tfstate
```

#### 5.4.2.2 Terraform template file configuration for BOSH:

The content of the terraform template file `terraform.tfvars` for BOSH is as follows:

```
auth_url = "<auth_url>"
domain_name = "<domain_name>"
user_name = "<ostack_user>"
password = "<ostack_pw>"
tenant_name = "<ostack_tenant_name>"
region_name = "<region_name>"
availability_zone = "<availability_zone>"

ext_net_name = "<ext_net_name>"
ext_net_id = "<ext_net_id>"

# in case your OpenStack needs custom nameservers
# dns_nameservers = 8.8.8.8

# Disambiguate keypairs with this suffix
# keypair_suffix = "<keypair_suffix>"

# Disambiguate security groups with this suffix
# security_group_suffix = "<security_group_suffix>"

# in case of self signed certificate select one of the following options
# cacert_file = "<path-to-certificate>"
# insecure = "true"
```

To edit the `terraform.tfvars` for BOSH using the described in this documentation scenario:

```
nano terraform.tfvars
```

Enter the following settings:

```
auth_url="http://192.168.40.228:5000/v3"
domain_name="cf_domain"
user_name="eniware"
password= <your_password>
```

(continues on next page)

(continued from previous page)

```
tenant_name="cloudfoundry"  
region_name="RegionOne"  
availability_zone="nova"  
ext_net_name="ext_net"  
ext_net_id="db178716-7d8a-444b-854a-685feb5bf7ea"
```

- `auth_url` is the *URL of the Keystone service*, which is `http://192.168.40.228:5000/v3` in our case (it can be retrieved by using `juju status | grep keystone/0` command).
- The *created domain `cf_domain`*, with *project `cloudfoundry`* and *user `eniware`* are set in the template in the `domain_name`, `user_name`, `password`, and `tenant_name` fields.
- `region_name` can be retrieved when editing the Neutron config file or from [here](#).
- The *defined external network* is set in `ext_net_name` field.
- The `ext_name_id` identifier can be retrieved from the OpenStack web UI (go to *Project > Network > Networks*, click on `ext_net` and go to **Overview** tab) or by using the *command* `openstack network list`.

### 5.4.3 OpenStack environment for Cloud Foundry

#### 5.4.3.1 Setup an OpenStack project to install Cloud Foundry:

To setup the project to install Cloud Foundry please use the following [Terraform module](#). Adapt `terraform.tfvars.template` to your needs. Variable `bosh_router_id` is output of the previous BOSH terraform module.

1. Copy the *template file* `terraform.tfvars` file for **cf-deployment-tf**:

```
cp ../bosh-openstack-environment-templates/cf-deployment-tf/terraform.tfvars.  
→template ./terraform.tfvars
```

2. Configure the *Terraform environment template* `terraform.tfvars` for **cf-deployment-tf**:
3. Run the following commands:

```
terraform init ../bosh-openstack-environment-templates/cf-deployment-tf/  
terraform apply ../bosh-openstack-environment-templates/cf-deployment-tf/
```

#### 5.4.3.2 Terraform template file configuration for Cloud Foundry:

The content of the terraform template file `terraform.tfvars` for Cloud Foundry is as follows:

```
auth_url = "<auth-url>"  
domain_name = "<domain>"  
user_name = "<user>"  
password = "<password>"  
project_name = "<project-name>"  
region_name = "<region-name>"  
availability_zones = ["<az-1>", "<az-2>", "<az-3>"]  
ext_net_name = "<external-network-name>"  
  
# the OpenStack router id which can be used to access the BOSH network  
bosh_router_id = "<bosh-router-id>"  
  
# in case Openstack has its own DNS servers
```

(continues on next page)

(continued from previous page)

```
dns_nameservers = ["<dns-server-1>","<dns-server-2>"]

# does BOSH use a local blobstore? Set to 'false', if your BOSH Director uses e.g. S3
↳to store its blobs
use_local_blobstore = "<true or false>" #default is true

# enable TCP routing setup
use_tcp_router = "<true or false>" #default is true
num_tcp_ports = <number> #default is 100, needs to be > 0

# in case of self signed certificate select one of the following options
# cacert_file = "<path-to-certificate>"
# insecure = "true"
```

To edit the `terraform.tfvars` for Cloud Foundry using the described in this documentation scenario:

```
nano terraform.tfvars
```

Enter the following settings:

```
auth_url="http://192.168.40.228:5000/v3"
domain_name="cf_domain"
user_name="eniware"
password= <your_password>
tenant_name="cloudfoundry"
region_name="RegionOne"
availability_zones = ["nova", "nova", "nova"]
bosh_router_id = ""
dns_nameservers = ["8.8.8.8"]
use_local_blobstore = "true"
use_tcp_router = "true"
num_tcp_ports = 100
```

- `auth_url`, `domain_name`, `user_name`, `password`, `tenant_name`, and `region_name` are the same as in `terraform.tfvars` *template file for BOSH*.
- `bosh_router_id` can be retrieved from the output of the previous *terraform script* for BOSH.

## 5.5 5.5. Install BOSH

To install the BOSH director please follow the instructions on section *6. Install BOSH* of this documentation.

For additional information you can visit [bosh.io](http://bosh.io).

Make sure the BOSH director is accessible through the BOSH cli, by following the instructions on [bosh.io](http://bosh.io). Use this mechanism in all BOSH cli examples in this documentation.

## 5.6 5.6. Cloud Config

After the BOSH director has been installed, you can prepare and upload a cloud config based on the `cloud-config.yml` file.

Take the variables and outputs from the Terraform run of `cf-deployment-tf` to finalize the cloud config.

Use the following command to upload the cloud config.

```
bosh update-cloud-config \  
-v availability_zone1="<az-1>" \  
-v availability_zone2="<az-2>" \  
-v availability_zone3="<az-3>" \  
-v network_id1="<cf-network-id-1>" \  
-v network_id2="<cf-network-id-2>" \  
-v network_id3="<cf-network-id-3>" \  
cf-deployment/iaas-support/openstack/cloud-config.yml
```

## 5.7 5.7. Deploy Cloud Foundry

To deploy Cloud Foundry run the following command filling in the necessary variables. `system_domain` is the user facing domain name of your Cloud Foundry installation.

```
bosh -d cf deploy cf-deployment/cf-deployment.yml \  
-o cf-deployment/operations/use-compiled-releases.yml \  
-o cf-deployment/operations/openstack.yml \  
-v system_domain="<system-domain>"
```

### With Swift as Blobstore

- Create four containers in Swift, which are used to store the artifacts for buildpacks, app-packages, droplets, and additional resources, respectively. The container names need to be passed in as variables in the below command snippet
- Set a [Temporary URL Key](#) for your Swift account

Add the following lines to the deploy cmd:

```
-o cf-deployment/operations/use-swift-blobstore.yml \  
-v auth_url="<auth-url>" \  
-v openstack_project="<project-name>" \  
-v openstack_domain="<domain>" \  
-v openstack_username="<user>" \  
-v openstack_password="<password>" \  
-v openstack_temp_url_key="<temp-url-key>" \  
-v app_package_directory_key="<app-package-directory-key>" \  
-v buildpack_directory_key="<buildpack-directory-key>" \  
-v droplet_directory_key="<droplet-directory-key>" \  
-v resource_directory_key="<resource-directory-key>"
```



**BOSH** is a project that unifies release engineering, deployment, and lifecycle management of small and large-scale cloud software. BOSH can provision and deploy software over hundreds of VMs. It also performs monitoring, failure recovery, and software updates with zero-to-minimal downtime.

While BOSH was developed to deploy Cloud Foundry PaaS, it can also be used to deploy almost any other software (Hadoop, for instance). BOSH is particularly well-suited for large distributed systems. In addition, BOSH supports multiple Infrastructure as a Service (IaaS) providers like VMware vSphere, Google Cloud Platform, Amazon Web Services EC2, Microsoft Azure, and OpenStack. There is a Cloud Provider Interface (CPI) that enables users to extend BOSH to support additional IaaS providers such as Apache CloudStack and VirtualBox.

### 6.1 6.1. Getting Started

The bosh **CLI** is the command line tool used for interacting with all things BOSH. Release binaries are available on [GitHub](#). See [Installation](#) for more details on how to download and install.

### 6.2 6.2. Installing the BOSH CLI

Choose your preferred installation method below to get the latest version of bosh.

### 6.2.1 Using the binary directly

To install the BOSH binary directly:

1. Navigate to the [BOSH CLI GitHub release page](#) and choose the correct download for your operating system.
2. Make the bosh binary executable and move the binary to your **PATH**:

```
$ chmod +x ./bosh
$ sudo mv ./bosh /usr/local/bin/bosh
```

3. You should now be able to use bosh. Verify by querying the CLI for its version:

```
$ bosh -v
version 5.3.1-8366c6fd-2018-09-25T18:25:51Z

Succeeded
```

### 6.2.2 Using Homebrew on macOS

If you are on macOS with [Homebrew](#), you can install using the [Cloud Foundry tap](#).

1. Use **brew** to install **bosh-cli**:

```
$ brew install cloudfoundry/tap/bosh-cli
```

2. You should now be able to use bosh. Verify by querying the CLI for its version:

```
$ bosh -v
version 5.3.1-8366c6fd-2018-09-25T18:25:51Z

Succeeded
```

---

**Note:** We currently do not publish BOSH CLI via apt or yum repositories.

---

## 6.3 6.3. Additional Dependencies

When you are using bosh to bootstrap BOSH or other standalone VMs, you will need a few extra dependencies installed on your local system.

---

**Note:** If you will not be using create-env and delete-env commands, you can skip this section.

---

### 6.3.1 Ubuntu Trusty

If you are running on Ubuntu Trusty, ensure the following packages are installed on your system:

```
$ sudo apt-get install -y build-essential zlibc zlibg-dev ruby ruby-dev openssl_
↪ libxslt-dev libxml2-dev libssl-dev libyaml-dev libsqlite3-dev sqlite3
```

(continues on next page)

(continued from previous page)

```
wget http://archive.ubuntu.com/ubuntu/pool/main/r/readline6/libreadline6_6.3-8ubuntu2_
↳amd64.deb
wget http://archive.ubuntu.com/ubuntu/pool/main/r/readline6/libreadline6-dev_6.3-
↳8ubuntu2_amd64.deb
sudo dpkg -i libreadline6_6.3-8ubuntu2_amd64.deb
sudo dpkg -i libreadline6-dev_6.3-8ubuntu2_amd64.deb
```

## 6.4 6.4. Quick Start

The easiest ways to get started with BOSH is by running on your local workstation with [VirtualBox](#). If you are interested in bringing up a director in another environment, like [Google Cloud Platform](#), choose your IaaS from the navigation for more detailed instructions.

### 6.4.1 Prerequisites

Before trying to deploy the Director, make sure you have satisfied the following requirements:

1. For best performance, ensure you have at least 8GB RAM and 50GB of free disk space.
2. Install the bosh *CLI* and its *additional dependencies*.
3. Install VirtualBox.

### 6.4.2 Install

First, create a workspace for our virtualbox environment. This directory will keep some state and configuration files that we will need.

```
$ mkdir -p ~/bosh-env/virtualbox
$ cd ~/bosh-env/virtualbox
```

Next, we'll use [bosh-deployment](#), the recommended installation method, to bootstrap our director.

```
$ git clone https://github.com/cloudfoundry/bosh-deployment.git
```

Now, we can run the `virtualbox/create-env.sh` script to create our test director and configure the environment with some defaults.

```
$ ./bosh-deployment/virtualbox/create-env.sh
```

During the bootstrap process, you will see a few stages:

- Creating BOSH Director - dependencies are downloaded, the VM is created, and BOSH is installed, configured, and started.
- Adding Network Routes - a route to the virtual network is added to ensure you will be able to connect to BOSH-managed VMs.
- Generating `.envrc` - a settings file is generated so you can easily connect to the environment later.
- Configuring Environment Alias - an alias is added for the bosh command so you can reference the environment as `vbox`.
- Updating Cloud Config - default settings are applied to the Director so you easily deploy software later.

After a few moments, BOSH should be started. To verify, first load your connection settings, and then run your first bosh command where you should see similar output.

```
$ source .envrc
$ bosh -e vbox env
Using environment '192.168.50.6' as client 'admin'

Name          bosh-lite
UUID          7ce65259-471a-424b-88cb-9d3cee85db2c
Version       265.2.0 (00000000)
CPI           warden_cpi
User          admin
```

Congratulations - BOSH is running! Now you're ready to deploy.

---

**Note:** *Troubleshooting* If you run into any trouble, please continue to the [VirtualBox Troubleshooting](#) section.

---

### 6.4.3 Deploy

Run through quick steps below or follow [deploy workflow](#) that goes through the same steps but with more explanation.

1. Update cloud config

```
$ bosh -e vbox update-cloud-config bosh-deployment/warden/cloud-config.yml
```

2. Upload stemcell

```
$ bosh -e vbox upload-stemcell https://bosh.io/d/stemcells/bosh-warden-
↳boshlite-ubuntu-trusty-go_agent?v=3468.17 \ --sha1_
↳1dad6d85d6e132810439daba7ca05694cec208ab
```

3. Deploy example deployment

```
$ bosh -e vbox -d zookeeper deploy <(wget -O- https://raw.githubusercontent.com/cppforlife/zookeeper-release/master/manifests/zookeeper.yml)
```

4. Run Zookeeper smoke tests

```
$ bosh -e vbox -d zookeeper run-errand smoke-tests
```

### 6.4.4 Clean up

The test director can be deleted using the `virtualbox/delete-env.sh` script.

```
$ ./bosh-deployment/virtualbox/delete-env.sh
```

## 6.5 6.5. Initialize New Environment on OpenStack

TODO

# CHAPTER 7

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`