CDK/Ceph installation via MaaS and Juju

Final components layout:

	Bare Metal	**Container #1**	**Container #2**	**Container #3**	**Container #4**	**Container #5**	**VM #1**
Node #1	Ceph OSD	Ceph mon	k8s master/ flannel	etcd	flannel	easyrsa	k8s worker/ flannel
Node #2	Ceph OSD	Ceph mon	k8s master/ flannel	etcd	flannel	RADOSGW	k8s worker/ flannel
Node #3	Ceph OSD	Ceph mon	k8s master/ flannel	etcd	flannel	kubeapi load balancer	k8s worker/ flannel
	Bare Metal	VM #1					
Infra #1	MAAS Rack + Region	Juju Controller					

Pre-requisites:

- Have a VM created on the infra node with a pre-defined name (let it be “juju-controller”)

For that:
Adjust netplan config for having juju-controller VM to be connected to the bridge:

```
network:
    ethernets:
        enp129s0f0:
            addresses: []
            dhcp4: true
        enp129s0f1:
            addresses: []
            dhcp4: true
        enp1s0f0:
            dhcp4: no
            addresses: [172.17.1.6/16]
            gateway4: 172.17.0.1
            nameservers:
              addresses: [ 8.8.8.8,8.8.4.4 ]
        enp1s0f1:
            match:
              macaddress: 0c:c4:7a:f7:ef:71
        enp1s0f2:
            addresses: []
            dhcp4: true
            optional: true
        enp1s0f3:
            addresses: []
            dhcp4: true
            optional: true
    bridges:
        br0:
            dhcp4: no
            addresses: [192.168.1.1/24]
            interfaces:
              - enp1s0f1
    version: 2
```

After that run

```
sudo netplan try
#in case network is functioning properly - accept changes
```

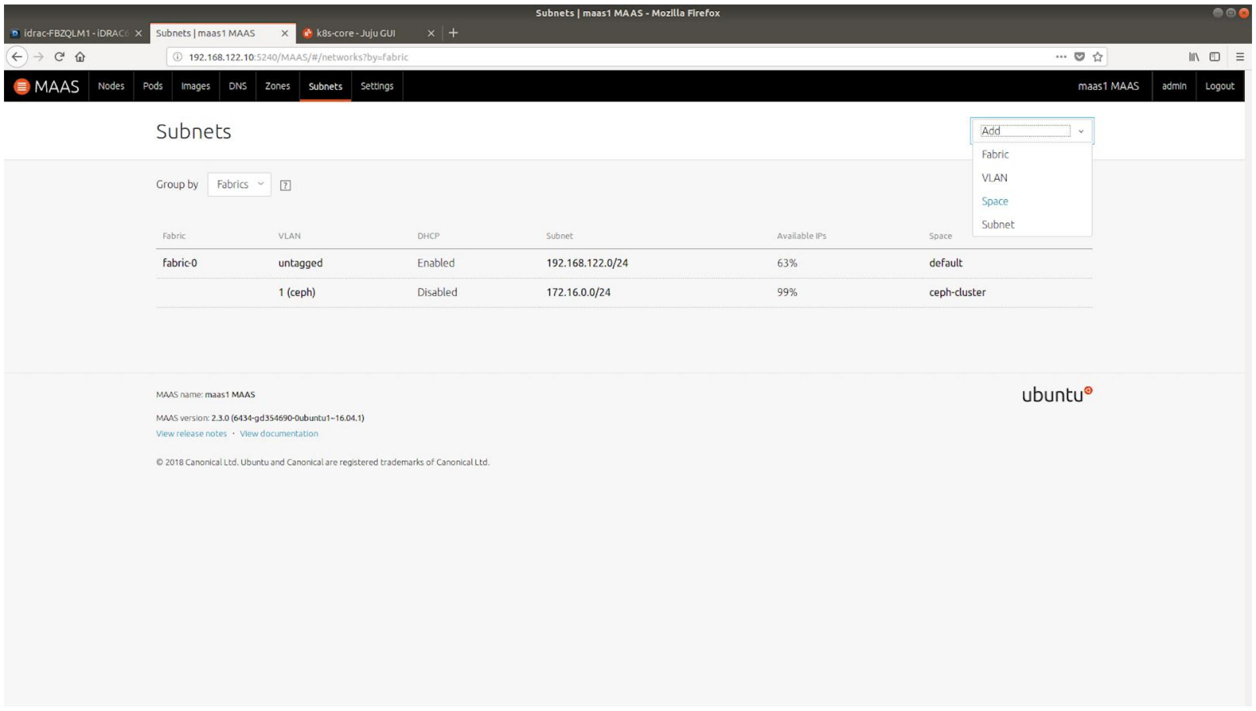
Set up masquerade routing for the nodes:

```
smadmin@sm-maas-kb:~$ cat iptables
#!/bin/bash
iptables -t nat -A POSTROUTING -o enp1s0f0 -j MASQUERADE
iptables -A FORWARD -i enp1s0f0 -o br0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i enp1s0f0 -o br0 -j ACCEPT
smadmin@sm-maas-kb:~$ chmod +x iptables
smadmin@sm-maas-kb:~$ sudo ./iptables
```

Create a VM using virt-install:

```
virt-install --name=juju-controller --disk size=50,sparse=no,pool=images --virt-type
kvm --graphics spice --vcpus=2 --ram=4096 --pxe --network bridge=br0
--os-type=linux --os-variant=ubuntu16.04
```

- MaaS should have 2 network spaces defined, one for general purposes **“default”** and another one for Ceph replication **“ceph-cluster”**. This can be done via MaaS UI, section “Subnets”:



Subnets

Fabric	VLAN	DHCP	Subnet	Available IPs	Space
fabric-0	untagged	Enabled	192.168.122.0/24	63%	default
	1 (ceph)	Disabled	172.16.0.0/24	99%	ceph-cluster

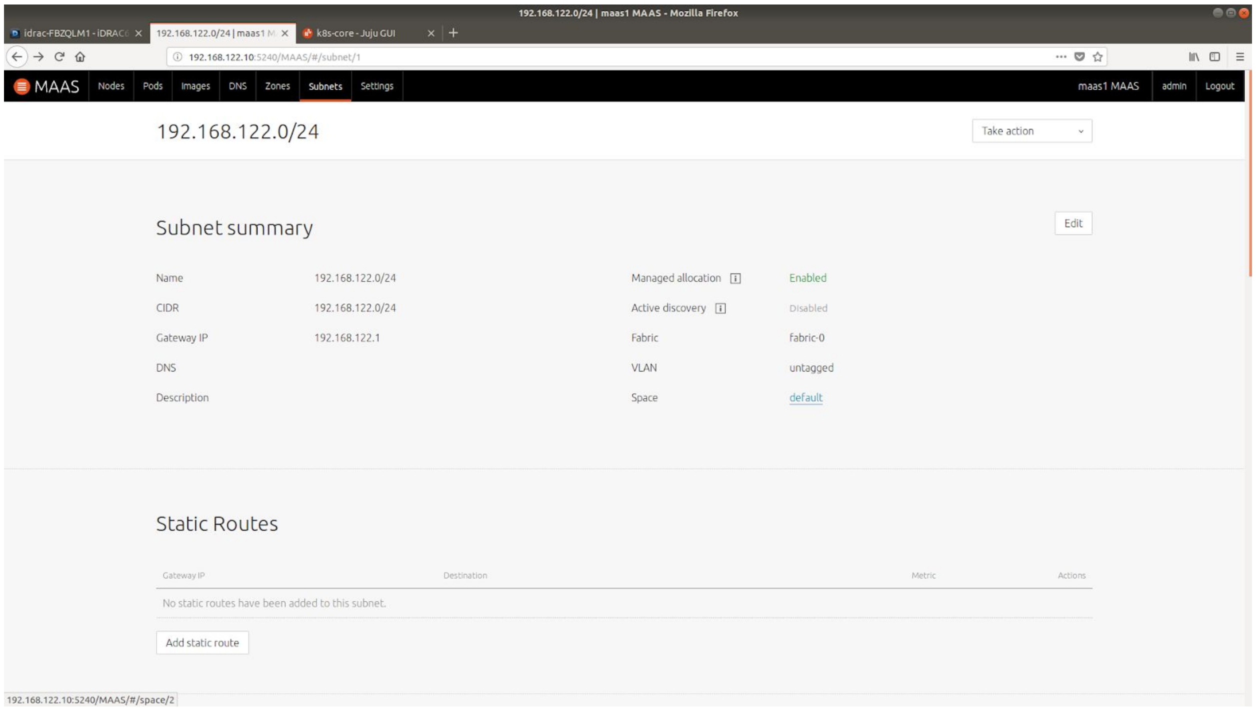
MAAS name: maas1 MAAS

MAAS version: 2.3.0 (6434-gd354690-0ubuntu1~16.04.1)

View release notes · View documentation

© 2018 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd.

After that each of 2 subnets should be edited to belong to its own space.


192.168.122.0/24 | maas1 MAAS - Mozilla Firefox
idrac-FBZQLM1 - iDRAC
192.168.122.0/24 | maas1 M
k8s-core - Juju GUI
192.168.122.10:5240/MAAS/#/subnet/1
MAAS
Nodes
Pods
Images
DNS
Zones
Subnets
Settings
maas1 MAAS
admin
Logout
Take action
Edit
Add static route
192.168.122.10:5240/MAAS/#/space/2

192.168.122.0/24

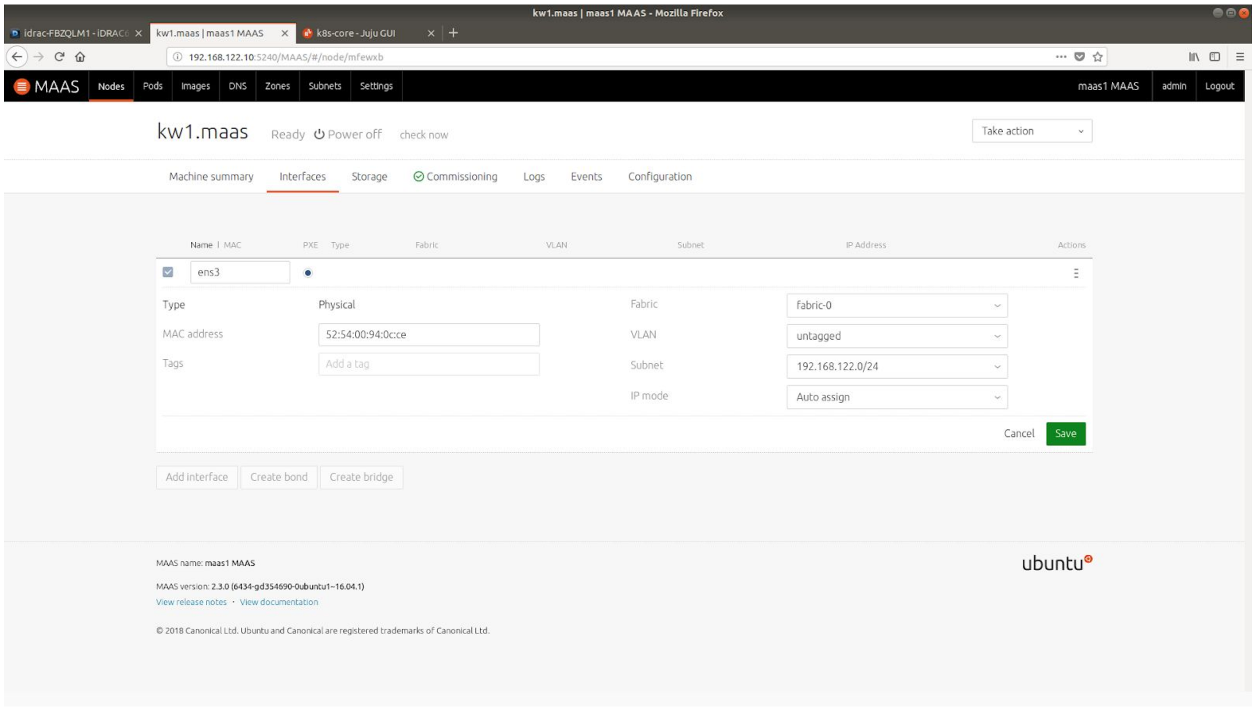
Subnet summary

Name	192.168.122.0/24	Managed allocation	Enabled
CIDR	192.168.122.0/24	Active discovery	Disabled
Gateway IP	192.168.122.1	Fabric	fabric-0
DNS		VLAN	untagged
Description		Space	default

Static Routes

Gateway IP	Destination	Metric	Actions
No static routes have been added to this subnet.			

- All cloud nodes should have their interfaces configured - assigned to the proper VLAN and subnet, IP address in "Auto-assigned" mode.



kw1.maas Ready Power off check now

Name: ens3

Type	Physical
MAC address	52:54:00:94:0c:ce
Tags	Add a tag
Fabric	fabric-0
VLAN	untagged
Subnet	192.168.122.0/24
IP mode	Auto assign

MAAS name: maas1 MAAS

MAAS version: 2.3.0 (6434-gd354690-0ubuntu1~16.04.1)

© 2018 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd.

All cloud nodes should be in “Ready” state.

<u>!NOTE Do not use LVM for the OS disks, otherwise nodes boot will never happen.</u>

Further actions should happen on the separate machine/container you will treat as your juju client (Ubuntu 18.04)

```
sudo snap install juju --classic
```

Get oauth1 credential from MaaS (API key, in UI under “admin”, MaaS keys)

```
ubuntu@ubuntu:~$ juju add-cloud
Cloud Types
  maas
  manual
  openstack
  oracle
  vsphere

Select cloud type: maas

Enter a name for your maas cloud: cloud1

Enter the API endpoint url: http://192.168.122.10:5240/MAAS

Cloud "cloud1" successfully added
You may bootstrap with 'juju bootstrap cloud1'

ubuntu@ubuntu:~$ juju add-credential cloud1
```

```
Enter credential name: cloud1-maas-creds

Using auth-type "oauth1".

Enter maas-oauth:

Credential "cloud1-maas-creds" added locally for cloud "cloud1".


ubuntu@ubuntu:~$ juju bootstrap cloud1 controller1 --to=juju-controller
Creating Juju controller "controller1" on cloud1
Looking for packaged Juju agent version 2.4.2 for amd64
Launching controller instance(s) on cloud1...
 - wh4wse (arch=amd64 mem=4G cores=2)
Installing Juju agent on bootstrap instance
Fetching Juju GUI 2.13.2
Waiting for address
Attempting to connect to 192.168.122.12:22
Connected to 192.168.122.12
Running machine configuration script...
Bootstrap agent now started
Contacting Juju controller at 192.168.122.12 to verify accessibility...
Bootstrap complete, "controller1" controller now available
Controller machines are in the "controller" model
Initial model "default" added

ubuntu@ubuntu:~$ juju add-model cdk-ceph
Uploading credential 'cloud1/admin/cloud1-maas-creds' to controller
Added 'cdk-ceph' model with credential 'cloud1-maas-creds' for user 'admin'

ubuntu@ubuntu:~$ juju deploy cdk-ceph-small.yaml

….
ubuntu@ubuntu:~$ juju status
Model     Controller   Cloud/Region  Version  SLA          Timestamp
k8s-core  controller1  cloud1        2.4.2    unsupported  21:57:04Z

App                Version  Status   Scale  Charm              Store       Rev  OS
Notes
easyrsa                     waiting    1/2  easyrsa            jujucharms   68
ubuntu
etcd               3.2.9    blocked      1  etcd               jujucharms  126
ubuntu
flannel            0.10.0   blocked      2  flannel            jujucharms   81
ubuntu
kubernetes-master  1.11.2   active       1  kubernetes-master  jujucharms  144
ubuntu  exposed
kubernetes-worker  1.11.2   waiting      1  kubernetes-worker  jujucharms  163
ubuntu  exposed

Unit                 Workload     Agent       Machine  Public address  Ports
Message
easyrsa/0*           maintenance  executing   0/lxd/0  192.168.122.15
(install) installing charm software
easyrsa/1            waiting      allocating  1/lxd/0
waiting for machine
etcd/0*              blocked      idle        0        192.168.122.13
Missing relation to certificate authority.
kubernetes-master/0*  active      idle        0        192.168.122.13
Kubernetes master running.
```

```
  flannel/0*          blocked      idle                 192.168.122.13
Waiting for etcd relation.
kubernetes-worker/0*  waiting      idle        1        192.168.122.14
Waiting for cluster DNS.
  flannel/1           maintenance  idle                 192.168.122.14
Unpacking flannel resource.

Machine  State    DNS             Inst id              Series  AZ       Message
0        started  192.168.122.13  etke74               bionic  default  Deployed
0/lxd/0  started  192.168.122.15  juju-3f7cb5-0-lxd-0  bionic  default  Container
started
1        started  192.168.122.14  yf7y6p               bionic  default  Deployed
1/lxd/0  pending                  juju-3f7cb5-1-lxd-0  bionic  default  Container
started
```

Once the deployment is done there should be k8s cluster created on top of 4 machines.

```
ubuntu@ubuntu:~$ mkdir -p ~/.kube
ubuntu@ubuntu:~$ juju scp kubernetes-master/0:config ~/.kube/config
ubuntu@ubuntu:~$ sudo snap install kubectl --classic
[sudo] password for ubuntu:
kubectl 1.11.2 from Canonical✓ installed
ubuntu@ubuntu:~$ kubectl cluster-info
Kubernetes master is running at https://192.168.122.13:6443
Heapster is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/prox
y
kubernetes-dashboard is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/https:kubernetes-
dashboard:/proxy
Metrics-server is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/https:metrics-ser
ver:/proxy
Grafana is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/monitoring-grafan
a/proxy
InfluxDB is running at
https://192.168.122.13:6443/api/v1/namespaces/kube-system/services/monitoring-influx
db:http/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

If it is necessary to extend the cluster (add physical worker node):

```
juju add-unit ceph-osd
juju add-unit kubernetes-worker --to=kvm:<new_machine_id>
```

Accessing Kubernetes cluster

1. Install kubectl on your machine (for Ubuntu - "`sudo snap install kubectl --classic`")
   Create directories:
   ```
   mkdir -p ~/.kube
   ```
2. Copy kube config from MaaS server to your machine:
   ```
   scp -r smadmin@64.169.30.89:~/.kube/* /home/agrebennikov/.kube/
   ```
3. Adjust the address of the k8s API server:
   ```
   sed -i 's/192.168.2.100:443/127.0.0.1:8443/' ~/.kube/config
   ```
4. Establish a tunnel to the MaaS server with port forwarding:
   ```
   ssh -L 8443:192.168.2.100:443 smadmin@64.169.30.89
   ```
   Make sure nobody else is occupying the same port 8443 at the moment, and if so - use another dynamic port.
5. Start kube proxy on the local machine:
   ```
   kubectl proxy
   ```
6. In the browser access the cluster via URL:
   ```
   http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-d
   ashboard:/proxy/
   ```

Appendix 1

CDK+Ceph bundle

Marked in red is unique per cluster and needs to be adjusted.

```
series: bionic
machines:
  '0':
    series: bionic
  '1':
    series: bionic
  '2':
    series: bionic
applications:
  keepalived:
    charm: cs:~containers/keepalived-4
    annotations:
      gui-x: '450'
      gui-y: '750'
    options:
      virtual_ip: 192.168.1.100/24
  flannel:
    charm: cs:~containers/flannel
    annotations:
      gui-x: '450'
      gui-y: '750'
  flannel-worker:
    charm: cs:~containers/flannel
    annotations:
      gui-x: '450'
      gui-y: '750'
```

```
    options:
      iface: enp0s2
  easyrsa:
    charm: cs:~containers/easyrsa
    num_units: 1
    annotations:
      gui-x: '450'
      gui-y: '550'
    to:
    - lxd:ceph-osd
    bindings:
      "": default
  ceph-osd:
    charm: cs:ceph-osd
    num_units: 3
    options:
      osd-devices: /dev/nvme0n1 /dev/nvme1n1
    bindings:
      "": default
      cluster: ceph-storage
    annotations:
      gui-x: '300'
      gui-y: '300'
    to:
    - 0
    - 1
    - 2
  ceph-radosgw:
    annotations:
      gui-x: '1000'
      gui-y: '250'
    charm: cs:ceph-radosgw
    num_units: 1
    bindings:
      "": default
    to:
    - lxd:ceph-osd
  kubernetes-worker:
    charm: cs:~containers/kubernetes-worker
    constraints: cores=24 mem=393216 root-disk=2560G
    num_units: 3
    expose: true
    annotations:
      gui-x: '100'
      gui-y: '850'
    bindings:
      "": default
    to:
    - 'kvm:0'
    - 'kvm:1'
    - 'kvm:2'
  kubernetes-master:
    charm: cs:~containers/kubernetes-master
    num_units: 3
    annotations:
      gui-x: '800'
      gui-y: '850'
    bindings:
      "": default
    to:
```

```
    - lxd:0
    - lxd:1
    - lxd:2
  ceph-mon:
    charm: 'cs:ceph-mon'
    num_units: 3
    annotations:
      gui-x: '600'
      gui-y: '300'
    bindings:
      "": default
    to:
     - lxd:0
     - lxd:1
     - lxd:2
  etcd:
    charm: cs:~containers/etcd
    num_units: 3
    annotations:
      gui-x: '800'
      gui-y: '550'
    bindings:
      "": default
    to:
    - lxd:ceph-osd
  kubeapi-load-balancer:
    charm: cs:~containers/kubeapi-load-balancer
    num_units: 1
    expose: true
    annotations:
      gui-x: '450'
      gui-y: '250'
    bindings:
      "": default
    options:
      extra_sans: 192.168.1.100
    to:
    - lxd:ceph-osd
relations:
- - 'ceph-mon:osd'
  - 'ceph-osd:mon'
- - 'kubernetes-master:kube-api-endpoint'
  - 'kubeapi-load-balancer:apiserver'
- - 'kubernetes-master:loadbalancer'
  - 'kubeapi-load-balancer:loadbalancer'
- - 'kubernetes-master:kube-control'
  - 'kubernetes-worker:kube-control'
- - 'kubernetes-master:certificates'
  - 'easyrsa:client'
- - 'etcd:certificates'
  - 'easyrsa:client'
- - 'kubernetes-master:etcd'
  - 'etcd:db'
- - 'kubernetes-worker:certificates'
  - 'easyrsa:client'
- - 'kubernetes-worker:kube-api-endpoint'
  - 'keepalived:website'
- - 'kubernetes-master:loadbalancer'
  - 'keepalived:loadbalancer'
- - 'kubeapi-load-balancer:website'
```

```
  - 'keepalived:lb-sink'
- - 'kubeapi-load-balancer:juju-info'
  - 'keepalived:juju-info'
- - 'kubeapi-load-balancer:certificates'
  - 'easyrsa:client'
- - 'flannel:etcd'
  - 'etcd:db'
- - 'flannel-worker:etcd'
  - 'etcd:db'
- - 'flannel:cni'
  - 'kubernetes-master:cni'
- - 'flannel-worker:cni'
  - 'kubernetes-worker:cni'
- - 'kubernetes-master:ceph-storage'
  - 'ceph-mon:admin'
- - 'ceph-mon:radosgw'
  - 'ceph-radosgw:mon'
```

Marked in red is unique per cluster and needs to be adjusted.

Issues

to activate Pod Security Policy:

ADMISSION_CONTROLLERS="$(juju ssh kubernetes-master/0 sudo snap get kube-apiserver admission-control)"
juju config kubernetes-master api-extra-args=admission-control=PodSecurityPolicy,$ADMISSION_CONTROLLERS

and then you need to create policies on top of that, as described in https://kubernetes.io/docs/concepts/policy/pod-security-policy/, to allow pods to be created