更新日：2019/12/1

本手順は MAAS/JUJU を使用し、Contrail(1912)、Kubernetes をインストールする手順となる。
※Contrail Insight は含まれない

MAAS にて 1 台の KVM を管理(Pod 機能)し、VirtualMachine の Deploy を実施
JUJU にて MAAS で Deploy された VirtualMachine に OS Install 及び Contrail Controller, OpenStack Controller/Compute, K8S Master/Worker の Install を実施

最新 Version の Install は本手順を参考に以下を参照下さい
https://www.juniper.net/documentation/en_US/contrail20/information-products/pathway-pages/contrail-install-and-upgrade-guide.html

本手順では Nested K8S 環境を作成する(K8S on OpenStack ではない)。

Contrail Version 毎に Supported SW Version は異なっているため、以下の通りインストールする必要がある
https://www.juniper.net/documentation/en_US/release-independent/contrail/topics/reference/contrail-supported-platforms.pdf

環境情報：

Network:

Mgmt / Data - 192.168.0.0/24

KVM: 192.168.0.1

VM:

MAAS Controller/JUJU Core : 192.168.0.2
JUJU Controller : dhcp
Contrail Controller : dhcp
K8S Master : dhcp
K8S Worker1 : dhcp

※Contrail Controller, K8S Master/Worker は MAAS/JUJU により Deploy されるため事前準備不要

Step1: KVM 事前準備

※以降全て"ubuntu" user で作業するものとする

Step1.1 KVM の Install 及び Nested Mode 有効化

```
sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system
bridge-utils virt-manager
sudo echo 'options kvm_intel nested=1' >> /etc/modprobe.d/qemu-system-
x86.conf
sudo cat /sys/module/kvm_intel/parameters/nested
```

Step1.2 Management/Data Net 作成

net-maas.xml

```
<network>
 <name>maas</name>
 <forward mode='route'/>
 <bridge name='br-mng-y' stp='off' delay='0' />
 <ip address='192.168.0.1' netmask='255.255.255.0'>
 </ip>
</network>
```

※MAAS は"maas"という名前の libvirt net を pxe boot 用に使用するため、名前変更不可

```
sudo virsh net-create net-maas.xml
```

Step1.3 Management と Data を分けたい場合は追加で Data 用 Net を作成(使用方法は後述)
net-data1.xml

```
<network>
 <name>data1</name>
 <forward mode='route'/>
 <bridge name='br-data1' stp='off' delay='0' />
 <ip address='192.168.1.1' netmask='255.255.255.0'>
 </ip>
</network>
```

```
sudo virsh net-create net-data1.xml
```

Step1.4 UFW 設定
/etc/default/ufw

```
DEFAULT_INPUT_POLICY="ACCEPT"
DEFAULT_FORWARD_POLICY="ACCEPT"
```

/etc/ufw/sysctl.conf

```
net.ipv4.ip_forward=1
```

Step2: MAAS Controller / JUJU Core 用 OS Install

```
sudo qemu-img create -f qcow2 /home/images/maas.qcow2 80G

sudo virt-install --name maas --disk path=/home/images/maas.qcow2,format=qcow2 --vcpus 1 --
ram=4096 --network bridge=br-mng-y,model=virtio --location /home/ubuntu/ubuntu-18.04.2-server-
amd64.iso --virt-type kvm --cpu host --os-variant ubuntu18.04 --serial pty --
console pty,target_type=virtio --graphics vnc,listen=0.0.0.0 --noautoconsole

sudo virt-install --name test --disk path=/home/images/test.qcow2,format=qcow2 --vcpus 1 --
ram=4096 --network bridge=br-mng-y,model=virtio --virt-type kvm --cpu host --os-variant ubuntu18.04 -
-serial pty --console pty,target_type=virtio --graphics vnc,listen=0.0.0.0 --noautoconsole
```

Ubuntu 18.04 をインストール

Step3: MAAS Controller Install

```
sudo timedatectl set-timezone Asia/Tokyo
sudo apt-get update
sudo apt-get upgrade
sudo reboot

sudo apt-get install ntp
sudo systemctl disable ufw
sudo systemctl stop ufw
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:maas/stable -y
sudo apt update
sudo apt install maas -y
```

Step3.1 MAAS 管理用 admin user の作成

```
sudo maas createadmin --username=admin --email=admin@localhost.local
```

Step3.2 KVM 管理用 maas user の ssh key の作成と KVM への Import

```
sudo chsh -s /bin/bash maas
sudo su - maas
ssh-keygen -f ~/.ssh/id\_rsa -N ''
logout
```

"sudo cat ~maas/.ssh/id_rsa.pub"で表示される maas user の PublicKey を MAAS からの管理対象である KVM Host の/home/ubuntu/.ssh/authorized_keys に追加

KVM にて以下を実施し、Password なしで list が表示できることを確認

```
sudo -H -u maas bash -c 'virsh -c qemu+ssh://ubuntu@192.168.0.1/system list --all'
```

Step3.3 作業用 ubuntu user の ssh key 作成

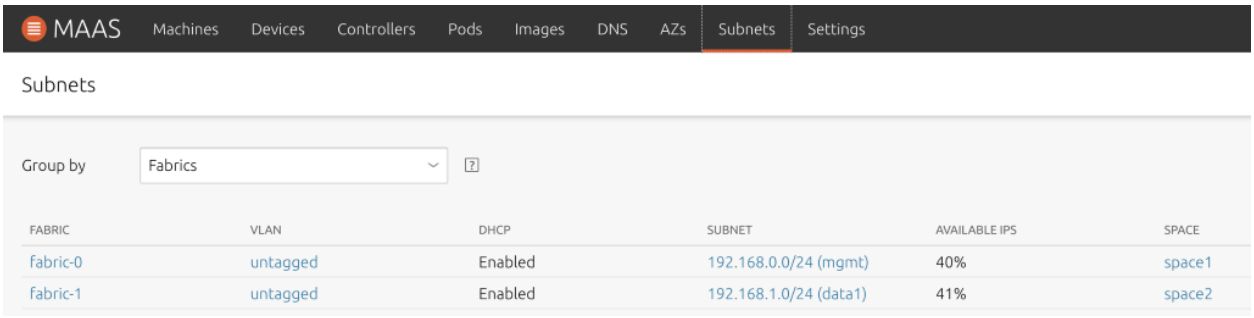
```
ssh-keygen
cat .ssh/id_rsa.pub
```

Step3.4 MAAS GUI Login

MAAS GUI(http://192.168.0.2:5240/MAAS)にアクセスし、Step3.3 で作成した id_rsa.pub を Import
※CLI Login は"maas login admin http://192.168.0.2:5240/MAAS"

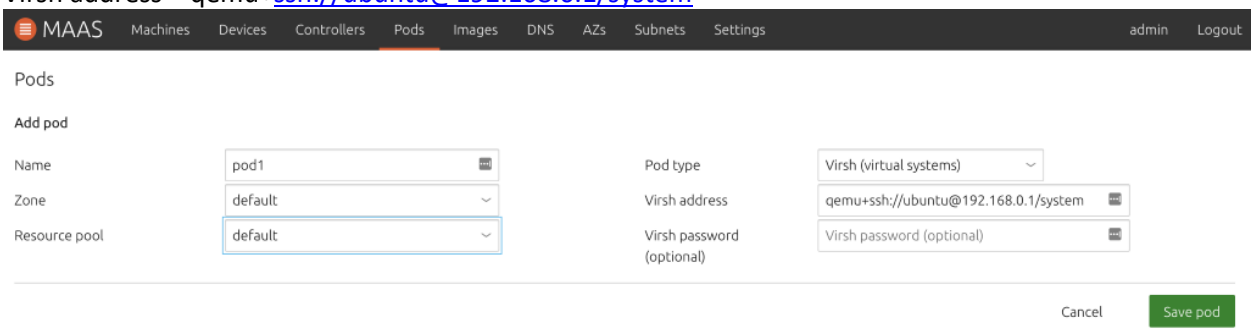
Step3.5 Subnet 作成

“Subnet” Tab の VLAN にて mgmt,data1 interface の DHCP を有効化
新規 space “space1”,“space2”を作成し、space1 に mgmt, space2 に data1 を設定



Step3.6 Pod 作成

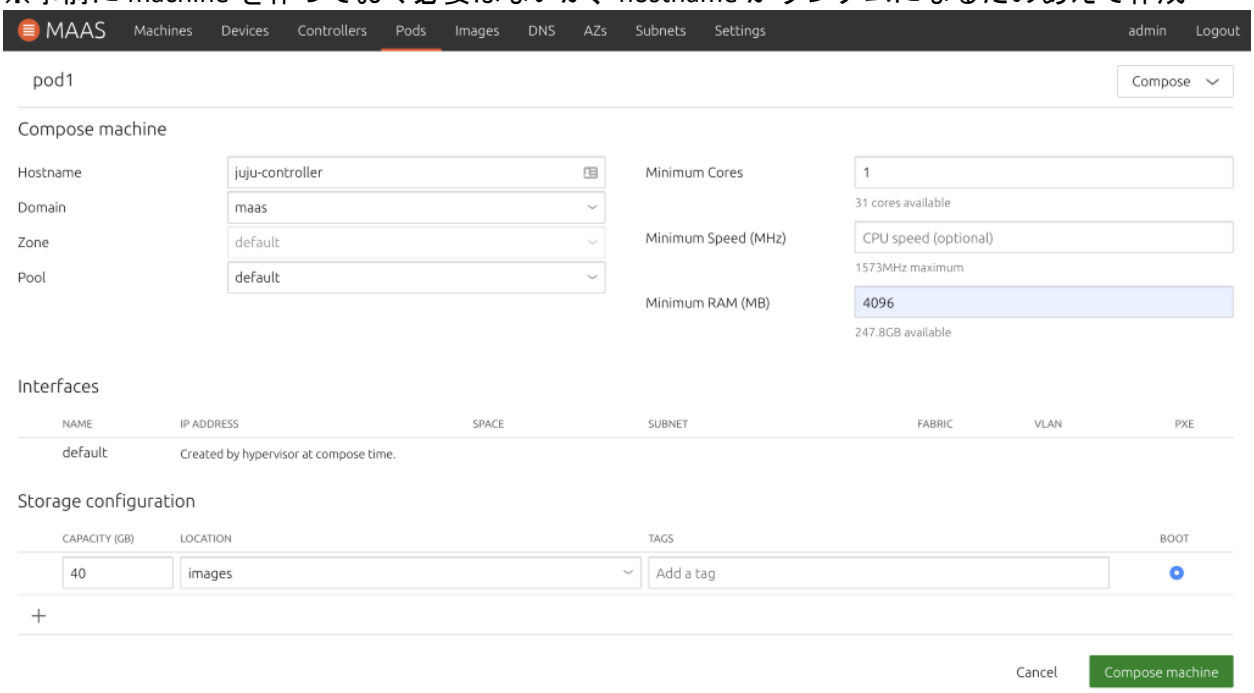
“Pod” Tab にて以下の設定で新規 Pod を作成
Pod Type = virsh
Virsh address = qemu+ssh://ubuntu@192.168.0.1/system



MAAS では Pod で指定した KVM に VM を Deploy する(後述)

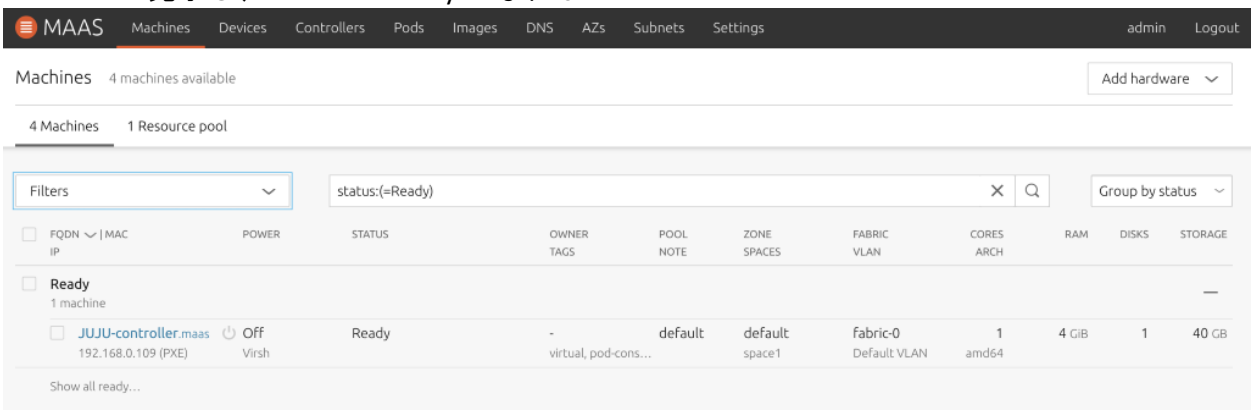
Step3.7 JUJU Controller 用 VM 作成

Pods->TakeAction->Compose にて JUJU Controller 用の VM を作成
※事前に machine を作っておく必要はないが、hostname がランダムになるためあえて作成



※以下は CLI での作成方法
maas admin pod compose 1 hostname=juju-controller cores=1 memory=4096 storage=40

PXE Boot が完了し、Status が Ready になれば OK



Step4 JUJU Core のインストール及び設定

MAAS Controller / JUJU Core の VM にて以下を実施

Step4.1 JUJU Core Install

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

Step4.2 JUJU で管理する Cloud を設定

JUJU では Cloud(aws, gcp, maas, etc)を指定し、各 Application を deploy 可能
今回は MAAS を利用しているため、maas 用の cloud を作成

juju add-cloud

```
Select cloud type: maas
Enter a name for your maas cloud: cloud1
Enter the API endpoint url: http://192.168.0.2:5240/MAAS
```

juju add-credential cloud1

```
Enter credential name: cloud1-creds
Enter maas-oauth: “MAAS GUI から Admin->MAAS Keys をコピペ"
```

Step4.2 JUJU Controller(BootStrap)をインストール

```
juju bootstrap --bootstrap-series=bionic cloud1 juju-controller
juju controller-config features="[multi-cloud]”
```

上記により MAAS で準備した juju-controller VM に Ubuntu bionic(18.04)と JUJU Controller がインストールされる

JUJU Controller の status 確認

```
ubuntu@maas:~$ juju status
Model    Controller      Cloud/Region    Version  SLA         Timestamp
default  juju-controller  cloud1/default  2.7.0    unsupported  14:55:46+09:00
```

Step4.3 JUJU GUI Access

以下にて GUI access 方法を確認

```
ubuntu@maas:~$ juju gui --no-browser
GUI 2.15.0 for model "admin/default" is enabled at:
  https://192.168.0.164:17070/gui/u/admin/default
Your login credential is:
  username: admin
  password: ca2ddc4fe5b3e837884f4f28d16f77b4
```

Step5 JUJU を使用し、K8S, Contrail をインストール

Step5.1 Contrail をインストールするための Charm を DownLoad

MAAS Controller / JUJU Core の VM にて以下を実施

```
git clone https://github.com/tungstenfabric/tf-charms
```

Step5.2 Multi NIC 設定

SingleNIC で問題ない場合は本 Step は Skip

MAAS 2.6.1 は Multi Space(Multi Interface)の Machine をデプロイできない Bug があり、juju bundle 内の Multi space が利用できない

※MultiNIC の Machine をデプロイするには以下を実施

---

maas admin pod compose 後に MAAS GUI から Inteface を追加

その後、KVM Host にて Virsh edit で Inteface を追加

juju add-machine --constraints spaces=space1,space2 を実施

Step5.3 各 VM を MAAS から Deploy

MAAS の VM 名はランダムなので、事前に Machine を用意する

※VM 名がランダムで OK の場合、本 Step は SKIP

```
maas admin pod compose 1 hostname=contrail1 cores=4 memory=16384 storage=200
maas admin tags create name=contrail1
maas admin tag update-nodes contrail1 add=[compose 時の system id]

maas admin pod compose 1 hostname=k8s-master cores=2 memory=8192 storage=40
maas admin tags create name=k8s-master
maas admin tag update-nodes k8s-master add=[compose 時の system id]

maas admin pod compose 1 hostname=k8s-worker1 cores=2 memory=8192 storage=40
```

```
maas admin tags create name=k8s-worker1
maas admin tag update-nodes k8s-worker1 add=[compose 時の system id]
```

また、Multi NIC の場合
Step5.4 の Charm Bundle の contrail-controller 箇所に xmpp 通信用の control-network を IP 指定する必要があるため、maas で machine 作成後に IP をチェックしておく

Step5.4 K8S, Contrail をインストールする JUJU Charm Bundle を設定

MAAS Controller / JUJU Core の VM にて以下のファイルを作成

k8s-contrail-bungle.yaml

```
machines:
 #openstack controller
 "1":
   series: bionic
   constraints: cores=2 mem=8G root-disk=40G
   #constraints: cores=2 mem=8G root-disk=40G tags=os-ctl                                  <- step5.3 実施時
 # k8s master
 "2":
   series: bionic
   constraints: cores=2 mem=8G root-disk=40G
   #constraints: cores=2 mem=8G root-disk=40G tags=k8s-master                              <- step5.2 実施時
   #constraints: cores=2 mem=8G root-disk=40G spaces=space1,space2 tags=k8s-master         <- step5.2,5.3 実施時
 # k8s worker
 "3":
   series: bionic
   constraints: cores=2 mem=8G root-disk=40G
   #constraints: cores=2 mem=8G root-disk=40G tags=k8s-worker1                             <- step5.2 実施時
   #constraints: cores=2 mem=8G root-disk=40G spaces=space1,space2 tags=k8s-worker1        <- step5.2,5.3 実施時


#common
series: bionic
services:
 ntp:
   charm: cs:ntp
   num_units: 0
   options:
     source: 210.173.160.27

#contrail
 contrail-agent:
   series: bionic
```

```
  charm: /home/ubuntu/tf-charms/contrail-agent
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    physical-interface: "ens5"
    vhost-gateway: "192.168.1.1"
    image-tag: "1912.32"
  num_units: 0
 contrail-analytics:
  series: bionic
  expose: true
  charm: /home/ubuntu/tf-charms/contrail-analytics
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    image-tag: "1912.32"
  num_units: 1
  to: [ "1" ]
 contrail-analyticsdb:
  series: bionic
  charm: /home/ubuntu/tf-charms/contrail-analyticsdb
  num_units: 1
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    image-tag: "1912.32"
    cassandra-minimum-diskgb: "4"
    cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
  to: [ "1" ]
 contrail-controller:
  series: bionic
  expose: true
  charm: /home/ubuntu/tf-charms/contrail-controller
  num_units: 1
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    image-tag: "1912.32"
#     control-network: "192.168.0.13/24"
#     data-network: "192.168.1.13/24”
    auth-mode: no-auth
    cassandra-minimum-diskgb: "4"
    cassandra-jvm-extra-opts: "-Xms1g -Xmx2g"
```

<- Data(xmpp)用 Interface と GatewayIP

<- multi nic の場合指定が必要

```
  to: [ "1" ]


# contrail-kubernetes
 contrail-kubernetes-master:
  charm: cs:~juniper-os-software/contrail-kubernetes-master
  series: bionic
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    image-tag: "1912.32"
    service_subnets: "10.96.0.0/12"
 contrail-kubernetes-node:
  charm: cs:~juniper-os-software/contrail-kubernetes-node
  series: bionic
  options:
    docker-registry: hub.juniper.net/contrail
    docker-user: xxxx
    docker-password: xxxx
    image-tag: "1912.32"


# kubernetes
 easyrsa:
  series: "bionic"
  charm: cs:~containers/easyrsa
  num_units: 1
  to:
  - "2"
 etcd:
  series: "bionic"
  charm: cs:~containers/etcd
  num_units: 1
  options:
    channel: 3.2/stable
  to:
  - "2"
 kubernetes-master:
  series: "bionic"
  charm: cs:~containers/kubernetes-master-696
  num_units: 1
  options:
    channel: 1.14/stable
    service-cidr: "10.96.0.0/12"
    enable-dashboard-addons: false
    enable-metrics: false
```

```
    dns-provider: "none"
    docker_runtime: "custom"
    docker_runtime_repo: "deb
[arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
    docker_runtime_key_url: "https://download.docker.com/linux/ubuntu/gpg"
    docker_runtime_package: "docker-ce"
  to:
  - "2"
 kubernetes-worker:
  series: "bionic"
  charm: cs:~containers/kubernetes-worker-550
  num_units: 1
  options:
    channel: 1.14/stable
    ingress: false
    docker_runtime: "custom"
    docker_runtime_repo: "deb
[arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
    docker_runtime_key_url: "https://download.docker.com/linux/ubuntu/gpg"
    docker_runtime_package: "docker-ce"
  to:
  - "3"


relations:
#contrail
- [ "contrail-analytics", "contrail-analyticsdb" ]
- [ "contrail-controller", "contrail-analytics" ]
- [ "contrail-controller", "contrail-analyticsdb" ]
- [ "contrail-agent", "contrail-controller"]
- [ "contrail-controller", "ntp" ]

# kubernetes
- [ kubernetes-master:kube-api-endpoint, kubernetes-worker:kube-api-
endpoint ]
- [ kubernetes-master:kube-control, kubernetes-worker:kube-control ]
- [ kubernetes-master:certificates, easyrsa:client ]
- [ kubernetes-master:etcd, etcd:db ]
- [ kubernetes-worker:certificates, easyrsa:client ]
- [ etcd:certificates, easyrsa:client ]
- [ kubernetes-master, ntp ]
- [ kubernetes-worker, ntp ]


#contrail kubernetes
- [ contrail-kubernetes-node:cni, kubernetes-master:cni ]
- [ contrail-kubernetes-node:cni, kubernetes-worker:cni ]
```

- [contrail-kubernetes-master:kube-api-endpoint, kubernetes-master:kube-api-endpoint] - [contrail-kubernetes-master:contrail-kubernetes-config, contrail-kubernetes-node:contrail-kubernetes-config] - [contrail-kubernetes-master:contrail-controller, contrail-controller:contrail-controller] - [contrail-agent:juju-info, kubernetes-worker:juju-info] - [contrail-agent:juju-info, kubernetes-master:juju-info]	

Step5.5 K8S, Contrail をインストール

```
juju add-model contrail1
juju switch contrail1
juju models
juju deploy ./k8s-contrail-bundle.yaml
juju status
```

JUJU では各 Application/Machine の集合を Model として定義可能(Model を分けておけば後でまとめて削除可能)
上記では contrail1 という model 内に openstack, K8S, contrail をインストールする

全て active になるまで 1.5h ほど

※以下のように contrail-analytics health check に失敗するため、bundle deploy 中に contrail 用の machine に login し、/etc/hosts を以下のように書き換える必要あり
alarm-gen is not ready. Reason: (Redis-UVE:AggregateRedis[None] connection down)

/etc/hosts in contrail-controller

```
192.168.0.xxx contrail1 contrail1.maas
#127.0.1.1 contrail1 contrail1.maas
```

Security Privilege を使用できるように設定

```
juju config kubernetes-master allow-privileged=true
```

※環境を削除する場合は以下
juju destroy-model contrail1

Step6.1. GUI アクセス

Contrail https://192.168.0.xx:8143 admin_domain/admin/contrail123

Step7. OpenStack Compute, K8S Worker 追加

K8S Worker を追加する場合は以下を実施

```
juju add-machine --constraints "mem=4G cores=2 root-disk=40G" --series=bionic
juju add-unit kubernetes-worker --to xxx
```