One Cluster Multiple Providers

Date: Jul 27th, 2024
Author: Jing Yan

This document aims to provide a step-by-step hands-on tutorial for setting up Kubernetes clusters via Cluster API with different providers, namely cluster-api-provider-metal3 and cluster-api-provider-kubevirt, so that Kubernetes clusters could be set up on both bare metal machines and KubeVirt VMs as what I have shared before in the following two tech sharings.

Bare Metal Kubernetes: https://drive.weixin.qq.com/s?k=ABAAngdnAAozJ5C4pz
Metal³: https://drive.weixin.qq.com/s?k=ABAAngdnAAoD1cTeyN

And, there are some other components mentioned in this doc, feel free to browse for more detailed information.

Kind: https://kind.sigs.k8s.io/
Cluster API: https://cluster-api.sigs.k8s.io/; https://github.com/kubernetes-sigs/cluster-api
Calico: https://docs.tigera.io/calico/latest/about/; https://github.com/projectcalico/calico
Cert Manager: https://cert-manager.io/; https://github.com/cert-manager/cert-manager
Virtualization on Ubuntu: https://ubuntu.com/server/docs/virtualisation-with-qemu
Metal³: https://book.metal3.io/introduction; https://github.com/metal3-io
Bare Metal Operator: https://book.metal3.io/bmo/introduction.html; https://github.com/metal3-io/baremetal-operator
MetalLB: https://metallb.universe.tf/; https://github.com/metallb/metallb
KubeVirt: https://kubevirt-manager.io/; https://github.com/kubevirt
cluster-api-provider-kubevirt: https://github.com/kubernetes-sigs/cluster-api-provider-kubevirt
cluster-api-provider-metal3: https://github.com/metal3-io/cluster-api-provider-metal3

Table of Contents

0. Environment description

In the following context, everything is deployed on this “X1 Carbon ThinkPad” laptop which has ubuntu-22.04 installed. If you would like to set the whole thing up on a distributed system, feel free to adjust the deployments.

```
# uname -a
Linux kubevirt-ThinkPad-X1-Carbon-Gen-11 6.5.0-1027-oem #28-Ubuntu SMP
PREEMPT_DYNAMIC Thu Jul 25 13:32:46 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

Docker will be necessary for the whole deployment, so just install it. Meanwhile, get your own “aliyun registry mirrors” and modify the docker configuration file accordingly IF YOU DO NOT HAVE A VPN && YOUR LOCATION IS CHINESE MAINLAND (which will save you ass thousands of times!!!).

```
# apt-get update
# apt-get install apt-transport-https ca-certificates curl
software-properties-common lrzsz -y
# curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key add -
# add-apt-repository "deb [arch=amd64]
https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
# apt-get update
# apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin -y
# systemctl enable --now docker
# mkdir -p /etc/docker
# sudo tee /etc/docker/daemon.json <<-'EOF'
{
 "registry-mirrors": ["https://6pra95pl.mirror.aliyuncs.com"]
}
EOF
# systemctl daemon-reload
# systemctl restart docker
# systemctl status docker
```

You will need “kubectl” to access the Kubernetes clusters later, so install it as follows.

```
# apt-get update && apt-get install -y apt-transport-https
# curl -fsSL
https://mirrors.aliyun.com/kubernetes-new/core/stable/v1.30/deb/Release.key | gpg
--dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
# echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://mirrors.aliyun.com/kubernetes-new/core/stable/v1.30/deb/ /" | tee
/etc/apt/sources.list.d/kubernetes.list
# apt-get update
# apt-get install -y kubectl
# kubectl version
```

You will need qemu-kvm-based virtual machines (VMs) to simulate bare metal machines, so the virtualization components are also necessary, as well as enabling Virtualization in BIOS.

```
# apt install cpu-checker -y
# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
# apt -y install bridge-utils cpu-checker libvirt-clients libvirt-daemon qemu
qemu-kvm virt-manager
# systemctl enable libvirtd --now
```

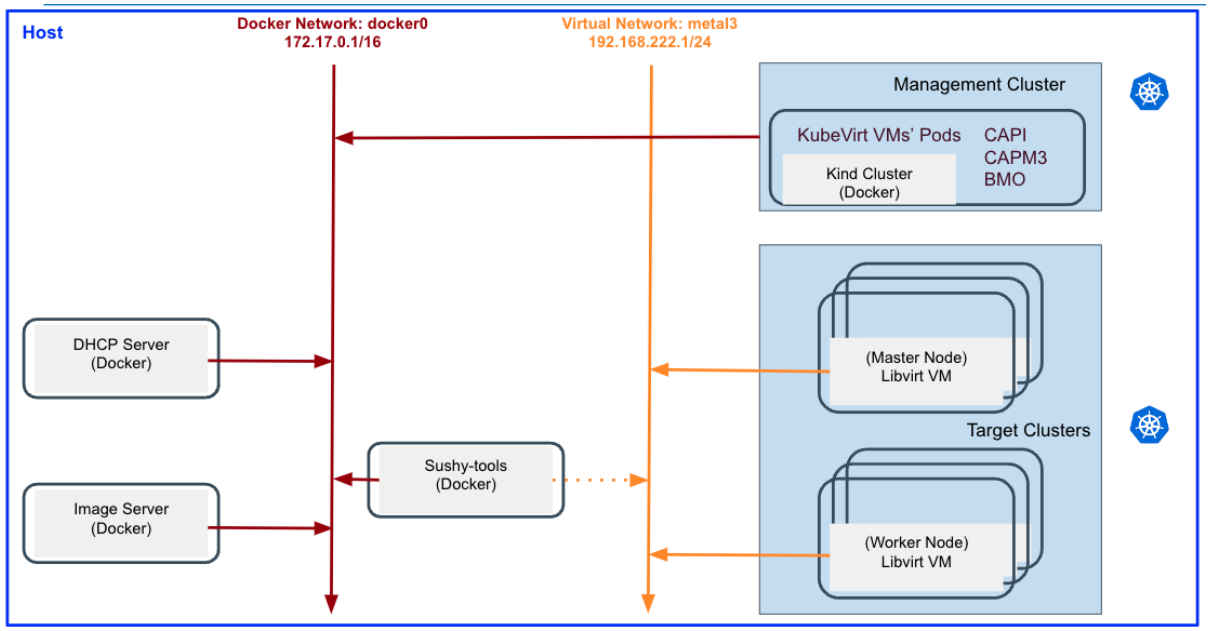
To manage the lifecycle of a Cluster API management cluster, the “clusterctl” CLI tool is needed as well.

```
# curl -L
https://github.com/kubernetes-sigs/cluster-api/releases/download/v1.8.1/clusterctl-
linux-amd64 -o clusterctl
# install -o root -g root -m 0755 clusterctl /usr/local/bin/clusterctl
# clusterctl version
```

What is more, you will need create and update the flat-files used to store usernames and password for basic authentication of Ironic, so install the following package.

```
# apt install apache2-utils
```

To have a better understanding of the environment, here is the figure.



Now, you are 100% ready to go, and it will be a plus if you are in a GOOD MOOD cause there will be, well, some unexpected and bitchy troubles most probably.

1. Set up the bootstrap Kubernetes cluster via kind

NOTE: This bootstrap Kubernetes cluster will serve as the management Kubernetes cluster later.
Pls, refer to Installation to install “kind” and choose whichever way you like, in this doc, “kind” is installed via “go install”. After the installation, the “kind” command should be available.

```
# Download the go package from https://go.dev/doc/install

# tar -C /usr/local -xzf go1.22.5.linux-amd64.tar.gz
# echo "export GOPATH=/root/go" >> ~/.bashrc
# echo "export GOROOT=/usr/local/go" >> ~/.bashrc
# echo "export GO111MODULE=on" >> ~/.bashrc
# echo "export GOPROXY=https://goproxy.cn,direct" >> ~/.bashrc
# echo "export PATH=$PATH:/usr/local/go/bin:/root/go/bin" >> ~/.bashrc
```

```
# source ~/.bashrc
# go install sigs.k8s.io/kind@latest
# kind --version
kind version 0.24.0
```

Then follow the following commands to create the Kubernetes cluster with the specific configuration. At the end, you should be able to see a Kubernetes cluster is up, and it is running inside a docker container (which is really cool!!!).

```
# cat kind.yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
 # Open ports for Ironic
 extraPortMappings:
 # Ironic httpd
 - containerPort: 6180
   hostPort: 6180
   listenAddress: "0.0.0.0"
   protocol: TCP
 # Ironic API
 - containerPort: 6385
   hostPort: 6385
   listenAddress: "0.0.0.0"
   protocol: TCP
 # Inspector API
 - containerPort: 5050
   hostPort: 5050
   listenAddress: "0.0.0.0"
   protocol: TCP
 extraMounts:
  - containerPath: /var/lib/kubelet/config.json
    hostPath: /etc/docker/daemon.json

# kind create cluster --config kind.yaml
Creating cluster "kind" ...
✓ Ensuring node image (kindest/node:v1.31.0) 🖼
✓ Preparing nodes 📦
✓ Writing configuration 📜
✓ Starting control-plane 🕹
✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Have a nice day! 👋

# kubectl cluster-info --context kind-kind
Kubernetes control plane is running at https://127.0.0.1:42937
CoreDNS is running at
https://127.0.0.1:42937/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

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

# docker container ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS              PORTS
NAMES
f7d23f56943c   kindest/node:v1.31.0   "/usr/local/bin/entr…"   About a minute ago
Up About a minute   0.0.0.0:5050->5050/tcp, 0.0.0.0:6180->6180/tcp,
0.0.0.0:6385->6385/tcp, 127.0.0.1:42937->6443/tcp   kind-control-plane
```

You could simply check the pods in the Kubernetes cluster with “kubectl” commands (“coredns” and “local-path-storage” are not ready because there is no network plugin deployed yet).

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -A
NAMESPACE            NAME                                         READY   STATUS    RESTARTS   AGE
cert-manager         cert-manager-7fbbc65b49-khr9p                0/1     Pending   0          14s
cert-manager         cert-manager-cainjector-6664fc84f6-gkwmg     0/1     Pending   0          14s
cert-manager         cert-manager-webhook-59598898fd-sxf9w        0/1     Pending   0          14s
kube-system          coredns-6f6b679f8f-9rf8t                     0/1     Pending   0          70m
kube-system          coredns-6f6b679f8f-fg7fg                     0/1     Pending   0          70m
kube-system          etcd-kind-control-plane                      1/1     Running   0          70m
kube-system          kube-apiserver-kind-control-plane            1/1     Running   0          70m
kube-system          kube-controller-manager-kind-control-plane   1/1     Running   0          70m
kube-system          kube-proxy-5kqms                             1/1     Running   0          70m
kube-system          kube-scheduler-kind-control-plane            1/1     Running   0          70m
local-path-storage   local-path-provisioner-57c5987fd4-hbcpd      0/1     Pending   0          70m
```

2. Deploy Calico CNI

Pls, refer to installation to install “Calico CNI” and choose whichever way you like, in this doc, “Calico CNI” is installed by simply using the Calico manifest to create the required resources. After the installation, you should be able to see Calico-related deployments and pods are up. Once the network plugin is ready, “coredns” and “local-path-storage” should be ready too.

```
# wget
https://raw.githubusercontent.com/projectcalico/calico/master/manifests/calico.yaml
# kubectl apply -f calico.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -A
NAMESPACE            NAME                                         READY   STATUS    RESTARTS   AGE
kube-system          calico-kube-controllers-7fbd86d5c5-4drcg     1/1     Running   0          10m
kube-system          calico-node-zc4rb                            1/1     Running   0          9m45s
kube-system          coredns-6f6b679f8f-9rf8t                     1/1     Running   0          83m
kube-system          coredns-6f6b679f8f-fg7fg                     1/1     Running   0          83m
kube-system          etcd-kind-control-plane                      1/1     Running   0          83m
kube-system          kube-apiserver-kind-control-plane            1/1     Running   0          83m
kube-system          kube-controller-manager-kind-control-plane   1/1     Running   0          83m
kube-system          kube-proxy-5kqms                             1/1     Running   0          83m
kube-system          kube-scheduler-kind-control-plane            1/1     Running   0          83m
local-path-storage   local-path-provisioner-57c5987fd4-hbcpd      1/1     Running   0          83m
```

3. Deploy Cert Manager

```
# wget
https://github.com/cert-manager/cert-manager/releases/download/v1.15.3/cert-manager
.yaml
# kubectl apply -f cert-manager.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -n cert-manager
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-7fbbc65b49-vjvpb              1/1     Running   0          16s
cert-manager-cainjector-6664fc84f6-lpcxh   1/1     Running   0          16s
cert-manager-webhook-59598898fd-cftb8      1/1     Running   0          16s
```

4. Start VMs

NOTE: As I do not have any bare metal machines yet, so I prepared the VMs to simulate bare metal machines.
First, prepare a virtual network for the VMs.

```
# cat baremetal.xml
<network>
 <name>baremetal</name>
 <forward mode='nat'>
   <nat>
     <port start='1024' end='65535'/>
   </nat>
 </forward>
 <bridge name='metal3'/>
 <ip address='192.168.222.1' netmask='255.255.255.0'>
 </ip>
</network>

# virsh net-define baremetal.xml
# virsh net-start baremetal
# virsh net-autostart baremetal
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# virsh net-list
 Name        State    Autostart   Persistent
--------------------------------------------
 baremetal   active   yes         yes
 default     active   yes         yes
```

Use “virt-install” to start two VMs (At this moment, VMs do not have any OS running inside).

```
# virt-install \
 --connect qemu:///system \
 --name bmh-vm-01 \
 --description "Virtualized BareMetalHost" \
 --osinfo=ubuntu-lts-latest \
 --ram=8192 \
 --vcpus=2 \
 --disk size=25 \
 --graphics=none \
 --console pty \
 --serial pty \
 --pxe \
 --network network=baremetal,mac="00:60:2f:31:81:01" \
 --noautoconsole

virt-install \
 --connect qemu:///system \
 --name bmh-vm-02 \
 --description "Virtualized BareMetalHost" \
 --osinfo=ubuntu-lts-latest \
 --ram=8192 \
```

```
--vcpus=2 \
--disk size=25 \
--graphics=none \
--console pty \
--serial pty \
--pxe \
--network network=baremetal,mac="00:60:2f:31:81:02" \
--noautoconsole
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# virsh list --all
 Id   Name        State
---------------------------
 13   bmh-vm-01   running
 14   bmh-vm-02   running
```

5. Set up sushy-tools to manage VMs

Metal3 relies on BMC to manage the bare metal machines, so we need something similar for the VMs. This comes in the form of sushy-tools.

```
# cat sushy-tools.conf
# Listen on 192.168.222.1:8000
SUSHY_EMULATOR_LISTEN_IP = u'192.168.222.1'
SUSHY_EMULATOR_LISTEN_PORT = 8000
# The libvirt URI to use. This option enables libvirt driver.
SUSHY_EMULATOR_LIBVIRT_URI = u'qemu:///system'

# docker run --name sushy-tools --rm --network host -d \
 -v /var/run/libvirt:/var/run/libvirt \
 -v "$(pwd)/sushy-tools.conf:/etc/sushy/sushy-emulator.conf" \
 -e SUSHY_EMULATOR_CONFIG=/etc/sushy/sushy-emulator.conf \
 quay.io/metal3-io/sushy-tools:latest sushy-emulator
```

Via the Redfish REST API, you should be able to use GET method to check the VMs information and use POST method to power on/off the VMs.

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# virsh domuuid bmh-vm-01
341c614e-46dd-4c34-b6cc-5312c15cf29c

root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# virsh domuuid bmh-vm-02
5778cdde-10dd-4dcc-b58d-c551d32c6d5f

root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# curl -s http://192.168.222.1:8000/redfish/v1/Systems | jq '.Members'
[
  {
    "@odata.id": "/redfish/v1/Systems/5778cdde-10dd-4dcc-b58d-c551d32c6d5f"
  },
  {
    "@odata.id": "/redfish/v1/Systems/341c614e-46dd-4c34-b6cc-5312c15cf29c"
  }
]
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# curl -s http://192.168.222.1:8000/redfish/v1/Systems/341c614e-46dd-4c34-b6cc-5312c15cf29c | jq '.Actions'
{
  "#ComputerSystem.Reset": {
    "target": "/redfish/v1/Systems/341c614e-46dd-4c34-b6cc-5312c15cf29c/Actions/ComputerSystem.Reset",
    "ResetType@Redfish.AllowableValues": [
      "On",
      "ForceOff",
      "GracefulShutdown",
      "GracefulRestart",
      "ForceRestart",
      "Nmi",
      "ForceOn"
    ]
  }
}
```

6. Set up DHCP Server for VMs

In reality, PXE is usually used when provisioning bare metal machines, a DHCP server is needed to assign the bare metal machines proper IPs.

```
# cat dnsmasq.env
DHCP_HOSTS=00:60:2f:31:81:01;00:60:2f:31:81:02
DHCP_IGNORE=tag:!known
# IP of the host from VM perspective
PROVISIONING_IP=192.168.222.1
GATEWAY_IP=192.168.222.1
DHCP_RANGE=192.168.222.100,192.168.222.149

# docker run --name dnsmasq --rm -d --net=host --privileged --user 997:994 \
 --env-file dnsmasq.env --entrypoint /bin/rundnsmasq \
 quay.io/metal3-io/ironic
```

7. Set up Image Server for VMs

In reality, when provisioning bare metal machines, iso files will be pulled to install the OS via PXE for the bare metal machines. Thus, an Image server is needed.

```
# mkdir disk-images
# cd disk-images
# wget https://cloud-images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img
# wget https://cloud-images.ubuntu.com/jammy/current/SHA256SUMS
# sha256sum --ignore-missing -c SHA256SUMS
# wget https://cloud.centos.org/centos/9-stream/x86_64/images/CentOS-Stream-GenericCloud-9-latest.x86_64.qcow2
# wget https://cloud.centos.org/centos/9-stream/x86_64/images/CentOS-Stream-GenericCloud-9-latest.x86_64.qcow2.SHA256SUM
# sha256sum -c CentOS-Stream-GenericCloud-9-latest.x86_64.qcow2.SHA256SUM
# wget https://artifactory.nordix.org/artifactory/metal3/images/k8s_v1.29.0/CENTOS_9_NODE_IMAGE_K8S_v1.29.0.qcow2
# sha256sum CENTOS_9_NODE_IMAGE_K8S_v1.29.0.qcow2
# cd ..
# docker run --name image-server --rm -d -p 80:8080 \
 -v "$(pwd)/disk-images:/usr/share/nginx/html" nginxinc/nginx-unprivileged
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# curl --head http://127.0.0.1:80/CENTOS_9_NODE_IM
AGE_K8S_v1.29.0.qcow2
HTTP/1.1 200 OK
Server: nginx/1.27.1
Date: Mon, 19 Aug 2024 08:43:40 GMT
Content-Type: application/octet-stream
Content-Length: 2270668288
Last-Modified: Wed, 10 Jan 2024 12:51:34 GMT
Connection: keep-alive
ETag: "659e92d6-8757a600"
Accept-Ranges: bytes
```

8. Deploy Ironic for Metal³

```
# tree ironic/
ironic/
├── ironic-auth-config
├── ironic_bmo.env
├── ironic-htpasswd
├── ironic-inspector-auth-config
├── ironic-inspector-htpasswd
├── ironic-patch.yaml
└── kustomization.yaml

# cat ironic-auth-config
[ironic]
auth_type=http_basic
username=IRONIC_USERNAME
password=IRONIC_PASSWORD

# cat ironic-inspector-auth-config
[inspector]
auth_type=http_basic
username=INSPECTOR_USERNAME
password=INSPECTOR_PASSWORD

# cat ironic-htpasswd
IRONIC_HTPASSWD="<output of `htpasswd -n -b -B IRONIC_USERNAME IRONIC_PASSWORD`>"

# cat ironic-inspector-htpasswd
INSPECTOR_HTPASSWD="<output of `htpasswd -n -b -B INSPECTOR_USERNAME
INSPECTOR_PASSWORD`>"

# cat ironic_bmo.env
HTTP_PORT=6180
PROVISIONING_INTERFACE=eth0
CACHEURL=http://192.168.222.1/images
IRONIC_KERNEL_PARAMS=console=ttyS0

# cat ironic-patch.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: ironic
spec:
 template:
   spec:
     containers:
     - name: ironic-dnsmasq
       $patch: delete

# cat kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
```

```
kind: Kustomization
namespace: baremetal-operator-system
resources:
- ../baremetal-operator-main/config/namespace
- ../baremetal-operator-main/ironic-deployment/base
# The kustomize components configure basic-auth and TLS
components:
- ../baremetal-operator-main/ironic-deployment/components/basic-auth
- ../baremetal-operator-main/ironic-deployment/components/tls
images:
- name: quay.io/metal3-io/ironic
 newTag: latest
# Create a ConfigMap from ironic_bmo.env and call it ironic-bmo-configmap.
# This ConfigMap will be used to set environment variables for the containers.
configMapGenerator:
- envs:
 - ironic_bmo.env
 name: ironic-bmo-configmap
 behavior: create

patches:
# Patch for removing dnsmasq
- path: ironic-patch.yaml
# The TLS component adds certificates but it cannot know the exact IPs of our
environment.
# Here we patch the certificates to have the correct IPs.
# - 172.18.0.2: kind cluster node IP. This is what Ironic will see attached to the
interface
#   and use to communicate with Inspector.
- patch: |-
   - op: replace
     path: /spec/ipAddresses/0
     value: 192.168.222.1
   - op: add
     path: /spec/ipAddresses/-
     value: 172.18.0.2
 target:
   kind: Certificate
   name: ironic-cert|ironic-inspector-cert
# The CA certificate should not have any IP address so we remove it.
- patch: |-
   - op: remove
     path: /spec/ipAddresses
 target:
   kind: Certificate
   name: ironic-cacert
# Create secrets from the authentication configuration.
# These will be mounted or used for environment variables.
# See the basic-auth component for more details on how they are used.
secretGenerator:
- name: ironic-htpasswd
 behavior: create
 envs:
 - ironic-htpasswd
- name: ironic-inspector-htpasswd
 behavior: create
 envs:
 - ironic-inspector-htpasswd
- name: ironic-auth-config
 files:
 - auth-config=ironic-auth-config
- name: ironic-inspector-auth-config
 files:
 - auth-config=ironic-inspector-auth-config

# kubectl create -k ironic --dry-run=client -o yaml
# kubectl apply -k ironic
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get deployment ironic -n baremetal-operator-system
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
ironic   1/1     1            1           81m
```

9. Deploy Bare Metal Operator for Metal³

```
# tree bmo/
bmo/
├── ironic.env
├── ironic-inspector-password
├── ironic-inspector-username
├── ironic-password
├── ironic-username
└── kustomization.yaml

# cat ironic-username
IRONIC_USERNAME

# cat ironic-password
IRONIC_PASSWORD

# cat ironic-inspector-username
INSPECTOR_USERNAME

# cat ironic-inspector-password
INSPECTOR_PASSWORD

# cat ironic.env
DEPLOY_KERNEL_URL=http://192.168.222.1:6180/images/ironic-python-agent.kernel
DEPLOY_RAMDISK_URL=http://192.168.222.1:6180/images/ironic-python-agent.initramfs
IRONIC_ENDPOINT=https://192.168.222.1:6385/v1/

# cat kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
namespace: baremetal-operator-system
resources:
- ../baremetal-operator-main/config/overlays/basic-auth_tls
images:
- name: quay.io/metal3-io/baremetal-operator
 newTag: latest
# Create a ConfigMap from ironic.env and name it ironic.
configMapGenerator:
- name: ironic
 behavior: create
 envs:
 - ironic.env

# We cannot use suffix hashes since the kustomizations we build on
# cannot be aware of what suffixes we add.
generatorOptions:
 disableNameSuffixHash: true
# Create secrets with the credentials for accessing Ironic.
secretGenerator:
- name: ironic-credentials
 files:
 - username=ironic-username
 - password=ironic-password
- name: ironic-inspector-credentials
 files:
 - username=ironic-inspector-username
 - password=ironic-inspector-password
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get deployment baremetal-operator-controller-manager
-n baremetal-operator-system
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
baremetal-operator-controller-manager   1/1     1            1           19m
```

10. Create BareMetalHosts

```
# cat bml-vm-01.yaml
apiVersion: v1
kind: Secret
metadata:
 name: bml-01
type: Opaque
stringData:
 username: replaceme
 password: replaceme
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
 name: bml-vm-01
spec:
 online: true
 bootMACAddress: 00:60:2f:31:81:01
 bootMode: UEFI # use 'legacy' for Scenario 2
 hardwareProfile: libvirt
 bmc:
   address:
redfish-virtualmedia+http://192.168.222.1:8000/redfish/v1/Systems/341c614e-46dd-4c3
4-b6cc-5312c15cf29c
   credentialsName: bml-01

# cat bml-vm-02.yaml
apiVersion: v1
kind: Secret
metadata:
 name: bml-02
type: Opaque
stringData:
 username: replaceme
 password: replaceme
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
 name: bml-vm-02
spec:
 online: true
 bootMACAddress: 00:60:2f:31:81:02
 bootMode: UEFI # use 'legacy' for Scenario 2
 hardwareProfile: libvirt
 bmc:
   address:
redfish-virtualmedia+http://192.168.222.1:8000/redfish/v1/Systems/5778cdde-10dd-4dc
c-b58d-c551d32c6d5f
   credentialsName: bml-02

# kubectl apply -f bml-vm-01.yaml
# kubectl apply -f bml-vm-02.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get bmh -A
NAMESPACE   NAME        STATE       CONSUMER   ONLINE   ERROR   AGE
default     bml-vm-01   available              true             9m52s
default     bml-vm-02   available              true             8m33s
```

11. Install MetalLB

KubeVirt is a cloud native virtualization solution. The VMs we're going to create and use for the workload cluster's nodes, are actually running within pods in the management cluster. In order to communicate with the workload cluster's API server, we'll need to expose it. The easiest way to expose the workload cluster's API server (a pod within a node running in a VM that is itself running within a pod in the management cluster, that is running inside a Docker container), is to use a LoadBalancer service, namely MetalLB in this doc.

```
# wget
https://raw.githubusercontent.com/metallb/metallb/v0.14.8/config/manifests/metallb-
native.yaml
# kubectl apply -f metallb-native.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get deployments -n metallb-system
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
controller   1/1     1            1           8m15s
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~#
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -n metallb-system
NAME                          READY   STATUS    RESTARTS   AGE
controller-8694df9d9b-jgr57   1/1     Running   0          8m29s
speaker-7cjcf                 1/1     Running   0          8m29s
```

Create the IPAddressPool and the L2Advertisement custom resources.

```
# cat capi-ip-pool.yaml
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
 name: capi-ip-pool
 namespace: metallb-system
spec:
 addresses:
 - 172.18.255.200-172.18.255.250
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
 name: empty
 namespace: metallb-system

# kubectl apply -f capi-ip-pool.yaml
```

12. Deploy KubeVirt

```
# wget
https://github.com/kubevirt/kubevirt/releases/download/v1.3.0/kubevirt-operator.yam
l
# wget
https://github.com/kubevirt/kubevirt/releases/download/v1.3.0/kubevirt-cr.yaml
# kubectl apply -f kubevirt-operator.yaml
```

```
# kubectl apply -f kubevirt-cr.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -n kubevirt
NAME                               READY   STATUS    RESTARTS   AGE
virt-api-79f4646554-h7mhp          1/1     Running   0          11m
virt-controller-7cbbdbbf8f-pjmcr   1/1     Running   0          10m
virt-controller-7cbbdbbf8f-v6jtj   1/1     Running   0          10m
virt-handler-z2dwv                 1/1     Running   0          10m
virt-operator-84d89fd9f6-cj9zl     1/1     Running   0          13m
virt-operator-84d89fd9f6-rbjd6     1/1     Running   0          13m
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~#
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get deployments -n kubevirt
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
virt-api          1/1     1            1           12m
virt-controller   2/2     2            2           10m
virt-operator     2/2     2            2           13m
```

13. Initialize the management cluster

In this chapter, steps to initialize the management cluster with the KubeVirt Provider and Metal³ Provider will be displayed.

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# clusterctl init --infrastructure kubevirt --infrastructure metal3

Fetching providers
Skipping installing cert-manager as it is already installed
Installing provider="cluster-api" version="v1.8.1" targetNamespace="capi-system"
Installing provider="bootstrap-kubeadm" version="v1.8.1" targetNamespace="capi-kubeadm-bootstrap-system"
Installing provider="control-plane-kubeadm" version="v1.8.1" targetNamespace="capi-kubeadm-control-plane-system"
Installing provider="infrastructure-kubevirt" version="v0.1.9" targetNamespace="capk-system"
Installing provider="infrastructure-metal3" version="v1.7.1" targetNamespace="capm3-system"

Your management cluster has been initialized successfully!

You can now create your first workload cluster by running the following:

  clusterctl generate cluster [name] --kubernetes-version [version] | kubectl apply -f -
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pod -n capi-system
NAME                                       READY   STATUS    RESTARTS   AGE
capi-controller-manager-6cb7846fdf-672ms   1/1     Running   0          56m
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~#
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pod -n capm3-system
NAME                                         READY   STATUS    RESTARTS   AGE
capm3-controller-manager-864b9cddc6-4t5l9    1/1     Running   0          56m
ipam-controller-manager-64f4697b6b-6pnrp     1/1     Running   0          56m
```

After all of these, you should be able to get a management Kubernetes cluster with the following pods/deployments/services.

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -A
NAMESPACE                           NAME                                                       READY   STATUS    RESTARTS   AGE
baremetal-operator-system           baremetal-operator-controller-manager-8d59474bd-ndmg7      2/2     Running   0          39h
baremetal-operator-system           ironic-8569ccdcfb-mdkpw                                    3/3     Running   0          40h
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager-7989ff8dd5-fs7jr 1/1     Running   0          38h
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-manager-5df44cc55f-cml8s 1/1 Running   0          38h
capi-system                         capi-controller-manager-6cb7846fdf-672ms                   1/1     Running   0          38h
capk-system                         capk-controller-manager-777f8c496b-9shzg                   1/1     Running   0          38h
capm3-system                        capm3-controller-manager-864b9cddc6-4t5l9                  1/1     Running   0          38h
capm3-system                        ipam-controller-manager-64f4697b6b-6pnrp                   1/1     Running   0          38h
cert-manager                        cert-manager-7fbbc65b49-vjvpb                              1/1     Running   0          45h
cert-manager                        cert-manager-cainjector-6664fc84f6-lpcxh                   1/1     Running   0          45h
cert-manager                        cert-manager-webhook-59598898fd-cftb8                      1/1     Running   0          45h
default                             virt-launcher-damn-kubevirt-control-plane-zmdq9-nvnsw      3/3     Running   0          19h
default                             virt-launcher-damn-kubevirt-md-0-6j2sz-b2drk-nlzg7         3/3     Running   0          19h
default                             virt-launcher-mixed-quickstart-control-plane-rgz2b-8zrjj   3/3     Running   0          19h
kube-system                         calico-kube-controllers-7fbd86d5c5-4drcg                   1/1     Running   0          45h
kube-system                         calico-node-zc4rb                                          1/1     Running   0          45h
kube-system                         coredns-6f6b679f8f-9rf8t                                   1/1     Running   0          47h
kube-system                         coredns-6f6b679f8f-fg7fg                                   1/1     Running   0          47h
kube-system                         etcd-kind-control-plane                                    1/1     Running   0          47h
kube-system                         kube-apiserver-kind-control-plane                          1/1     Running   0          47h
kube-system                         kube-controller-manager-kind-control-plane                 1/1     Running   0          47h
kube-system                         kube-proxy-5kqms                                           1/1     Running   0          47h
kube-system                         kube-scheduler-kind-control-plane                          1/1     Running   0          47h
kubevirt                            virt-api-79f4646554-h7mhp                                  1/1     Running   0          39h
kubevirt                            virt-controller-7cbbdbbf8f-pjmcr                           1/1     Running   0          38h
kubevirt                            virt-controller-7cbbdbbf8f-v6jtj                           1/1     Running   0          38h
kubevirt                            virt-handler-z2dwv                                         1/1     Running   0          38h
kubevirt                            virt-operator-84d89fd9f6-cj9zl                             1/1     Running   0          39h
kubevirt                            virt-operator-84d89fd9f6-rbjd6                             1/1     Running   0          39h
local-path-storage                  local-path-provisioner-57c5987fd4-hbcpd                    1/1     Running   0          47h
metallb-system                      controller-8694df9d9b-jgr57                                1/1     Running   0          39h
metallb-system                      speaker-7cjcf                                              1/1     Running   0          39h
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get deployments -A
NAMESPACE                           NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
baremetal-operator-system           baremetal-operator-controller-manager      1/1     1            1           39h
baremetal-operator-system           ironic                                     1/1     1            1           41h
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager  1/1     1            1           38h
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-manager 1/1  1            1           38h
capi-system                         capi-controller-manager                    1/1     1            1           38h
capk-system                         capk-controller-manager                    1/1     1            1           38h
capm3-system                        capm3-controller-manager                   1/1     1            1           38h
capm3-system                        ipam-controller-manager                    1/1     1            1           38h
cert-manager                        cert-manager                               1/1     1            1           45h
cert-manager                        cert-manager-cainjector                    1/1     1            1           45h
cert-manager                        cert-manager-webhook                       1/1     1            1           45h
kube-system                         calico-kube-controllers                    1/1     1            1           45h
kube-system                         coredns                                    2/2     2            2           47h
kubevirt                            virt-api                                   1/1     1            1           39h
kubevirt                            virt-controller                            2/2     2            2           38h
kubevirt                            virt-operator                              2/2     2            2           39h
local-path-storage                  local-path-provisioner                     1/1     1            1           47h
metallb-system                      controller                                 1/1     1            1           39h
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get services -A
NAMESPACE                           NAME                                              TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)                  AGE
baremetal-operator-system           baremetal-operator-controller-manager-metrics-service ClusterIP  10.96.197.130   <none>         8443/TCP                 39h
baremetal-operator-system           baremetal-operator-webhook-service                ClusterIP      10.96.62.32     <none>         443/TCP                  39h
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-webhook-service            ClusterIP      10.96.47.228    <none>         443/TCP                  38h
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-webhook-service        ClusterIP      10.96.156.3     <none>         443/TCP                  38h
capi-system                         capi-webhook-service                              ClusterIP      10.96.35.19     <none>         443/TCP                  38h
capk-system                         capk-webhook-service                              ClusterIP      10.96.102.66    <none>         443/TCP                  38h
capm3-system                        capm3-webhook-service                             ClusterIP      10.96.32.140    <none>         443/TCP                  38h
capm3-system                        ipam-webhook-service                              ClusterIP      10.96.102.170   <none>         443/TCP                  38h
cert-manager                        cert-manager                                      ClusterIP      10.96.138.108   <none>         9402/TCP                 45h
cert-manager                        cert-manager-webhook                              ClusterIP      10.96.116.20    <none>         443/TCP                  45h
default                             damn-kubevirt-lb                                  LoadBalancer   10.96.46.29     172.18.0.200   6443:30739/TCP           19h
default                             kubernetes                                        ClusterIP      10.96.0.1       <none>         443/TCP                  47h
kube-system                         kube-dns                                          ClusterIP      10.96.0.10      <none>         53/UDP,53/TCP,9153/TCP   47h
kubevirt                            kubevirt-operator-webhook                         ClusterIP      10.96.227.49    <none>         443/TCP                  39h
kubevirt                            kubevirt-prometheus-metrics                       ClusterIP      None            <none>         443/TCP                  39h
kubevirt                            virt-api                                          ClusterIP      10.96.220.50    <none>         443/TCP                  39h
kubevirt                            virt-exportproxy                                  ClusterIP      10.96.205.118   <none>         443/TCP                  39h
metallb-system                      metallb-webhook-service                           ClusterIP      10.96.15.38     <none>         443/TCP                  39h
```

14. Create the target cluster on KubeVirt VMs only

```
# export CAPK_GUEST_K8S_VERSION="v1.30.1"
# export CRI_PATH="/var/run/containerd/containerd.sock"
# export NODE_VM_IMAGE_TEMPLATE="quay.io/capk/ubuntu-2204-container-disk:v1.30.1"
```

```
# clusterctl generate cluster damn-kubevirt \
 --infrastructure="kubevirt" \
 --flavor lb \
 --kubernetes-version ${CAPK_GUEST_K8S_VERSION} \
 --control-plane-machine-count=1 \
 --worker-machine-count=1 \
 > damn-kubevirt.yaml
 # kubectl apply -f damn-kubevirt.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get cluster
NAME            CLUSTERCLASS   PHASE         AGE     VERSION
damn-kubevirt                  Provisioned   3m35s
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get kubevirtmachines
NAME                                AGE     READY
damn-kubevirt-control-plane-zmdq9   3m36s   true
damn-kubevirt-md-0-6j2sz-b2drk      3m36s   true
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get pods -A | grep damn
default                     virt-launcher-damn-kubevirt-control-plane-zmdq9-nvnsw          3/3     Running   0          12m
default                     virt-launcher-damn-kubevirt-md-0-6j2sz-b2drk-nlzg7             3/3     Running   0          11m
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get machines -A
NAMESPACE   NAME                                CLUSTER         NODENAME
  PROVIDERID                                    PHASE     AGE     VERSION
default     damn-kubevirt-control-plane-zmdq9   damn-kubevirt   damn-kubevirt-control-plane-zmdq9
  kubevirt://damn-kubevirt-control-plane-zmdq9  Running   3m41s   v1.30.1
default     damn-kubevirt-md-0-6j2sz-b2drk      damn-kubevirt   damn-kubevirt-md-0-6j2sz-b2drk
  kubevirt://damn-kubevirt-md-0-6j2sz-b2drk     Running   3m41s   v1.30.1
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# clusterctl describe cluster damn-kubevirt
NAME                                                          READY  SEVERITY  REASON                           SINCE  MESSAGE
Cluster/damn-kubevirt                                         True                                              10m
├─ClusterInfrastructure - KubevirtCluster/damn-kubevirt       True                                              11m
├─ControlPlane - KubeadmControlPlane/damn-kubevirt-control-plane  True                                          10m
│ └─Machine/damn-kubevirt-control-plane-zmdq9                 True                                              10m
└─Workers
  └─MachineDeployment/damn-kubevirt-md-0                      False  Warning   WaitingForAvailableMachines      11m    Minimum a
    └─Machine/damn-kubevirt-md-0-6j2sz-b2drk                  True                                              10m
```

15. Create the target cluster on Bare Metal Machines only (**problematic**)

```
# export
IMAGE_CHECKSUM="7b2fbe69b2f2446d151b3e198b7fb020a4f17b9ab237c1b59b843e2783218b66"
# export IMAGE_CHECKSUM_TYPE="sha256"
# export IMAGE_FORMAT="qcow2"
# export IMAGE_URL="http://192.168.222.1/UBUNTU_22.04_NODE_IMAGE_K8S_v1.30.0.qcow2"
# export KUBERNETES_VERSION="v1.30.0"
# export POD_CIDR='["192.168.10.0/24"]'
# export CTLPLANE_KUBEADM_EXTRA_CONFIG=""
# export WORKERS_KUBEADM_EXTRA_CONFIG=""
# export CLUSTER_APIENDPOINT_HOST="192.168.222.100"
# export CLUSTER_APIENDPOINT_PORT="6443"
# clusterctl generate cluster stupid-metal3 --control-plane-machine-count 1
--worker-machine-count 1 --infrastructure="metal3" > fuck-metal3.yaml
# kubectl apply -f fuck-metal3.yaml
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get cluster fuck-metal3
NAME          CLUSTERCLASS   PHASE         AGE    VERSION
fuck-metal3                  Provisioned   141m
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~#
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl get bmh -A
NAMESPACE   NAME        STATE         CONSUMER            ONLINE   ERROR   AGE
default     bml-vm-01   provisioned   fuck-metal3-dtkg2   true             144m
```

Indeed, there is an issue left without solving. As we can see from the pictures, the cluster is created and the “bmh” resource is consumed by the cluster successfully. What is more, the corresponding VM for the “bmh” resource is installed with OS via PXE and obtained the IP address. From the laptop, “ping” this “bmh” resource also succeeded.

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# kubectl describe bmh bml-vm-01
Name:         bml-vm-01
Namespace:    default
Labels:       cluster.x-k8s.io/cluster-name=fuck-metal3
Annotations:  <none>
API Version:  metal3.io/v1alpha1
Kind:         BareMetalHost
Metadata:
 Creation Timestamp:  2024-08-21T03:27:31Z
 Finalizers:
   baremetalhost.metal3.io
 Generation:  4
 Owner References:
   API Version:     infrastructure.cluster.x-k8s.io/v1beta1
   Controller:      true
   Kind:            Metal3Machine
   Name:            fuck-metal3-dtkg2
   UID:             90a82087-b5b4-4e08-8ddb-40c05152b303
 Resource Version:  714436
 UID:               f2990f89-ba55-4092-92fd-e57f36362cb3
Spec:
 Architecture:              x86_64
 Automated Cleaning Mode:  metadata
 Bmc:
   Address:
redfish-virtualmedia+http://192.168.222.1:8000/redfish/v1/Systems/341c614e-46dd-4c3
4-b6cc-5312c15cf29c
   Credentials Name:  bml-01
 Boot MAC Address:    00:60:2f:31:81:01
 Boot Mode:           UEFI
 Consumer Ref:
   API Version:     infrastructure.cluster.x-k8s.io/v1beta1
   Kind:            Metal3Machine
   Name:            fuck-metal3-dtkg2
   Namespace:       default
 Hardware Profile:  libvirt
 Image:
   Checksum:       7b2fbe69b2f2446d151b3e198b7fb020a4f17b9ab237c1b59b843e2783218b66
   Checksum Type:  sha256
   Format:         qcow2
   URL:            http://192.168.222.1/UBUNTU_22.04_NODE_IMAGE_K8S_v1.30.0.qcow2
 Online:           true
 User Data:
   Name:       fuck-metal3-dtkg2
   Namespace:  default
Status:
 Error Count:    0
 Error Message:
 Good Credentials:
   Credentials:
     Name:               bml-01
```

```
    Namespace:           default
  Credentials Version:  714034
Hardware:
  Cpu:
    Arch:   x86_64
    Count:  2
    Flags:
      3dnowprefetch
      abm
      adx
      aes
      apic
      arat
      arch_capabilities
      avx
      avx2
      avx_vnni
      bmi1
      bmi2
      clflush
      clflushopt
      clwb
      cmov
      constant_tsc
      cpuid
      cpuid_fault
      cx16
      cx8
      de
      ept
      ept_ad
      erms
      f16c
      flexpriority
      fma
      fpu
      fsgsbase
      fsrm
      fxsr
      gfni
      hypervisor
      ibpb
      ibrs
      ibrs_enhanced
      invpcid
      lahf_lm
      lm
      mca
      mce
      md_clear
      mmx
      movbe
      movdir64b
      movdiri
      msr
      mtrr
      nopl
      nx
      ospke
      pae
      pat
      pclmulqdq
      pdpe1gb
      pge
      pku
      pni
      popcnt
      pse
```

```
      pse36
      rdpid
      rdrand
      rdseed
      rdtscp
      rep_good
      sep
      serialize
      sha_ni
      smap
      smep
      ss
      ssbd
      sse
      sse2
      sse4_1
      sse4_2
      ssse3
      stibp
      syscall
      tpr_shadow
      tsc
      tsc_adjust
      tsc_deadline_timer
      tsc_known_freq
      umip
      vaes
      vme
      vmx
      vnmi
      vpclmulqdq
      vpid
      waitpkg
      x2apic
      xgetbv1
      xsave
      xsavec
      xsaveopt
      xsaves
      xtopology
    Model:  13th Gen Intel(R) Core(TM) i7-1360P
  Firmware:
    Bios:
      Date:     02/06/2015
      Vendor:   EFI Development Kit II / OVMF
      Version:  0.0.0
  Hostname:     localhost.localdomain
  Nics:
    Ip:           192.168.222.100
    Mac:          00:60:2f:31:81:01
    Model:        0x1af4 0x0001
    Name:         enp1s0
    Ip:           fe80::2f24:badb:8d3:796%enp1s0
    Mac:          00:60:2f:31:81:01
    Model:        0x1af4 0x0001
    Name:         enp1s0
  Ram Mebibytes:  8192
  Storage:
    Alternate Names:
      /dev/vda
      /dev/disk/by-path/pci-0000:04:00.0
    Name:         /dev/disk/by-path/pci-0000:04:00.0
    Rotational:   true
    Size Bytes:   26843545600
    Type:         HDD
    Vendor:       0x1af4
  System Vendor:
    Manufacturer:  QEMU
```

```
    Product Name:  Standard PC (Q35 + ICH9, 2009)
 Hardware Profile:  libvirt
 Last Updated:      2024-08-21T03:32:09Z
 Operation History:
   Deprovision:
     End:    <nil>
     Start:  <nil>
   Inspect:
     End:    2024-08-21T03:29:36Z
     Start:  2024-08-21T03:27:41Z
   Provision:
     End:    2024-08-21T03:32:09Z
     Start:  2024-08-21T03:30:49Z
   Register:
     End:              2024-08-21T03:30:49Z
     Start:            2024-08-21T03:30:49Z
 Operational Status:  OK
 Powered On:          true
 Provisioning:
   ID:         7af54a37-bc29-4e76-bd34-c279d79e1ba1
   Boot Mode:  UEFI
   Image:
     Checksum:
7b2fbe69b2f2446d151b3e198b7fb020a4f17b9ab237c1b59b843e2783218b66
     Checksum Type:  sha256
     Format:         qcow2
     URL:            http://192.168.222.1/UBUNTU_22.04_NODE_IMAGE_K8S_v1.30.0.qcow2
   Root Device Hints:
     Device Name:  /dev/vda
   State:          provisioned
 Tried Credentials:
   Credentials:
     Name:               bml-01
     Namespace:          default
   Credentials Version:  714034
Events:                   <none>
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# ping 192.168.222.100 -c1
PING 192.168.222.100 (192.168.222.100) 56(84) bytes of data.
64 bytes from 192.168.222.100: icmp_seq=1 ttl=64 time=0.204 ms

--- 192.168.222.100 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.204/0.204/0.204/0.000 ms
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~#
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# virsh console bmh-vm-01
Connected to domain 'bmh-vm-01'
Escape character is ^] (Ctrl + ])

bml-vm-01 login:
```

However, the “capm3-controller-manager-864b9cddc6-4t5l9” raised the error as follows, which is the root cause that the k8s on VMs (simulating the bare metal machines) failed to be up.

```
# kubectl describe machine fuck-metal3-dtkg2
...
Status:
 Bootstrap Ready:  true
 Conditions:
   Last Transition Time:  2024-08-21T03:32:12Z
```

```
   Message:               1 of 2 completed
   Reason:                SettingProviderIDOnNodeFailed
   Severity:              Error
   Status:                False
   Type:                  Ready
   Last Transition Time:  2024-08-21T03:30:48Z
   Status:                True
   Type:                  BootstrapReady
   Last Transition Time:  2024-08-21T03:32:12Z
   Message:               error retrieving node, requeuing. Object will be requeued
after 30s
   Reason:                SettingProviderIDOnNodeFailed
   Severity:              Error
   Status:                False
   Type:                  InfrastructureReady
   Last Transition Time:  2024-08-21T03:30:48Z
   Reason:                WaitingForNodeRef
   Severity:              Info
   Status:                False
   Type:                  NodeHealthy
 Last Updated:            2024-08-21T03:30:48Z
 Observed Generation:     2
 Phase:                   Provisioning
Events:                   <none>
```

```
# kubectl logs capm3-controller-manager-864b9cddc6-4t5l9 -n capm3-system
...
I0821 05:59:34.777394       1 metal3machine_manager.go:684] "msg"="Updating
machine" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
I0821 05:59:34.781713       1 metal3machine_manager.go:1599] "msg"="Metal3data is
ready" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
I0821 05:59:34.781748       1 metal3machine_manager.go:1120] "msg"="Deleting
nodeReuseLabelName from host, if any" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
I0821 05:59:34.782435       1 metal3machine_manager.go:729] "msg"="Finished
updating machine" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
E0821 05:59:34.795936       1 metal3machine_manager.go:1812] "msg"="error while
retrieving nodes" "error"="Get \"https://192.168.222.100:6443/api/v1/nodes\": dial
tcp 192.168.222.100:6443: connect: connection refused" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
I0821 05:59:34.795962       1 metal3machine_manager.go:1315] "msg"="error
retrieving node, requeuing" "cluster"="fuck-metal3"
"logger"="controllers.Metal3Machine.Metal3Machine-controller"
"machine"="fuck-metal3-dtkg2" "metal3-cluster"="fuck-metal3"
"metal3-machine"={"Namespace":"default","Name":"fuck-metal3-dtkg2"}
E0821 05:59:34.795984       1 metal3machine_controller.go:274] "msg"="Failed to set
the target node providerID" "error"="error retrieving node, requeuing. Object will
be requeued after 30s" "logger"="controllers.Metal3Machine" "providerID"=""
I0821 05:59:50.719091       1 metal3labelsync_controller.go:147] "msg"="Could not
find Node Ref on Machine object, will retry"
"logger"="controllers.Metal3LabelSync.metal3-label-sync-controller"
"metal3-label-sync"={"Namespace":"default","Name":"bml-vm-01"}
```

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# clusterctl describe cluster fuck-metal3
NAME                                                READY  SEVERITY  REASON                                                  SINCE  MESSAGE
Cluster/fuck-metal3                                 False  Error     SettingProviderIDOnNodeFailed @ Machine/fuck-metal3-dtkg2  150m   0 of 1 completed
├─ClusterInfrastructure - Metal3Cluster/fuck-metal3  True                                                                    152m
└─ControlPlane - KubeadmControlPlane/fuck-metal3    False  Error     SettingProviderIDOnNodeFailed @ Machine/fuck-metal3-dtkg2  150m   0 of 1 completed
  └─Machine/fuck-metal3-dtkg2                       False  Error     SettingProviderIDOnNodeFailed                              151m   1 of 2 completed
```

16. Create the target cluster on Bare Metal Machines and KubeVirt VMs (**problematic**)

Well, as we are still using VMs to simulate the bare metal machines here, so there is no doubt we will hit the issue in the last chapter. But I will provide a way here anyway about how to start a mixed cluster.

```
root@kubevirt-ThinkPad-X1-Carbon-Gen-11:~# clusterctl describe cluster damn-kubevirt
NAME                                                          READY  SEVERITY  REASON
Cluster/damn-kubevirt                                         True
├─ClusterInfrastructure - KubevirtCluster/damn-kubevirt       True
├─ControlPlane - KubeadmControlPlane/damn-kubevirt-control-plane  True
│ └─Machine/damn-kubevirt-control-plane-zmdq9                 True
└─Workers
  └─MachineDeployment/damn-kubevirt-md-0                      False  Warning   Waiting
    └─Machine/damn-kubevirt-md-0-6j2sz-b2drk                  True
```

So follow a similar topology, use “KubeVirt VM” as the control plane and “Metal3 Machines” as the workers, the yaml file should be in the following format.

```
# cat final.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
 name: mixed-quickstart
 namespace: default
spec:
 clusterNetwork:
   pods:
     cidrBlocks:
     - 192.168.0.0/16
   services:
     cidrBlocks:
     - 10.128.0.0/16
 controlPlaneRef:
   apiVersion: controlplane.cluster.x-k8s.io/v1beta1
   kind: KubeadmControlPlane
   name: mixed-quickstart-control-plane
   namespace: default
 infrastructureRef:
   apiVersion: infrastructure.cluster.x-k8s.io/v1alpha1
   kind: KubevirtCluster
   name: mixed-quickstart
   namespace: default
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3Cluster
metadata:
 name: mixed-quickstart
 namespace: default
spec:
 controlPlaneEndpoint:
```

```
   host: 192.168.0.101
   port: 6443
 noCloudProvider: true
---
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineDeployment
metadata:
 labels:
   cluster.x-k8s.io/cluster-name: mixed-quickstart
   nodepool: nodepool-0
 name: test1
 namespace: default
spec:
 clusterName: mixed-quickstart
 replicas: 1
 selector:
   matchLabels:
     cluster.x-k8s.io/cluster-name: mixed-quickstart
     nodepool: nodepool-0
 template:
   metadata:
     labels:
       cluster.x-k8s.io/cluster-name: mixed-quickstart
       nodepool: nodepool-0
   spec:
     bootstrap:
       configRef:
         apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
         kind: KubeadmConfigTemplate
         name: test1-workers
     clusterName: mixed-quickstart
     infrastructureRef:
       apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
       kind: Metal3MachineTemplate
       name: test1-workers
     nodeDrainTimeout: 0s
     version: v1.30.0
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3MachineTemplate
metadata:
 name: test1-workers
 namespace: default
spec:
 template:
   spec:
     dataTemplate:
       name: test1-workers-template
     image:
       checksum: 7b2fbe69b2f2446d151b3e198b7fb020a4f17b9ab237c1b59b843e2783218b66
       checksumType: sha256
       format: qcow2
       url: http://192.168.222.1/UBUNTU_22.04_NODE_IMAGE_K8S_v1.30.0.qcow2
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3DataTemplate
metadata:
 name: test1-workers-template
 namespace: default
spec:
 clusterName: mixed-quickstart
---
apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
kind: KubeadmConfigTemplate
metadata:
 name: test1-workers
 namespace: default
spec:
```

```
 template:
   spec:
     joinConfiguration:
       nodeRegistration:
         kubeletExtraArgs: {}
---
apiVersion: controlplane.cluster.x-k8s.io/v1beta1
kind: KubeadmControlPlane
metadata:
 name: mixed-quickstart-control-plane
 namespace: default
spec:
 kubeadmConfigSpec:
   clusterConfiguration:
     networking:
       dnsDomain: mixed-quickstart.default.local
       podSubnet: 192.168.0.0/16
       serviceSubnet: 10.128.0.0/16
   initConfiguration:
     nodeRegistration:
       criSocket: /var/run/containerd/containerd.sock
   joinConfiguration:
     nodeRegistration:
       criSocket: /var/run/containerd/containerd.sock
 machineTemplate:
   infrastructureRef:
     apiVersion: infrastructure.cluster.x-k8s.io/v1alpha1
     kind: KubevirtMachineTemplate
     name: mixed-quickstart-control-plane
     namespace: default
 replicas: 1
 version: v1.30.0
---
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha1
kind: KubevirtCluster
metadata:
 name: mixed-quickstart
 namespace: default
spec:
 controlPlaneEndpoint:
   host: 192.168.0.101
   port: 6443
 controlPlaneServiceTemplate:
   spec:
     type: LoadBalancer
---
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha1
kind: KubevirtMachineTemplate
metadata:
 name: mixed-quickstart-control-plane
 namespace: default
spec:
 template:
   spec:
     virtualMachineBootstrapCheck:
       checkStrategy: ssh
     virtualMachineTemplate:
       metadata:
         namespace: default
       spec:
         runStrategy: Always
         template:
           spec:
             domain:
               cpu:
                 cores: 2
               devices:
                 disks:
```

```
      - disk:
          bus: virtio
        name: containervolume
      networkInterfaceMultiqueue: true
    memory:
      guest: 4Gi
  evictionStrategy: External
  volumes:
  - containerDisk:
      image: quay.io/capk/ubuntu-2204-container-disk:v1.30.1
    name: containervolume
```

Conclusion

This is not a perfect tutorial yet as we do not have real bare metal machines to finish all the setup, however, in the first phase, we aim to R&D the KubeVirt VM-related scenarios, which has been proven to be feasible in this doc. Once I get the bare metal machines, I will update this doc as soon as possible.