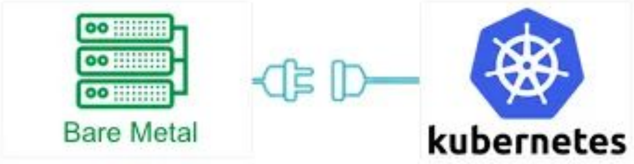
Optimizing Infrastructure: Transitioning to Bare Metal Kubernetes

Jing Yan
Jul 13, 2024

Agenda

- Background
  - Evolution of infrastructure
  - Motivation towards bare metal k8s
- Bare metal k8s
  - Benefits
  - Challenges
  - Key Technologies
    - Metal³ and How Metal³ manages (bare metal) machines
    - **Cluster API (CAPI) and How CAPI works**
    - Demo time: CAPI
- Summary
- References



Background: evolution of infrastructure

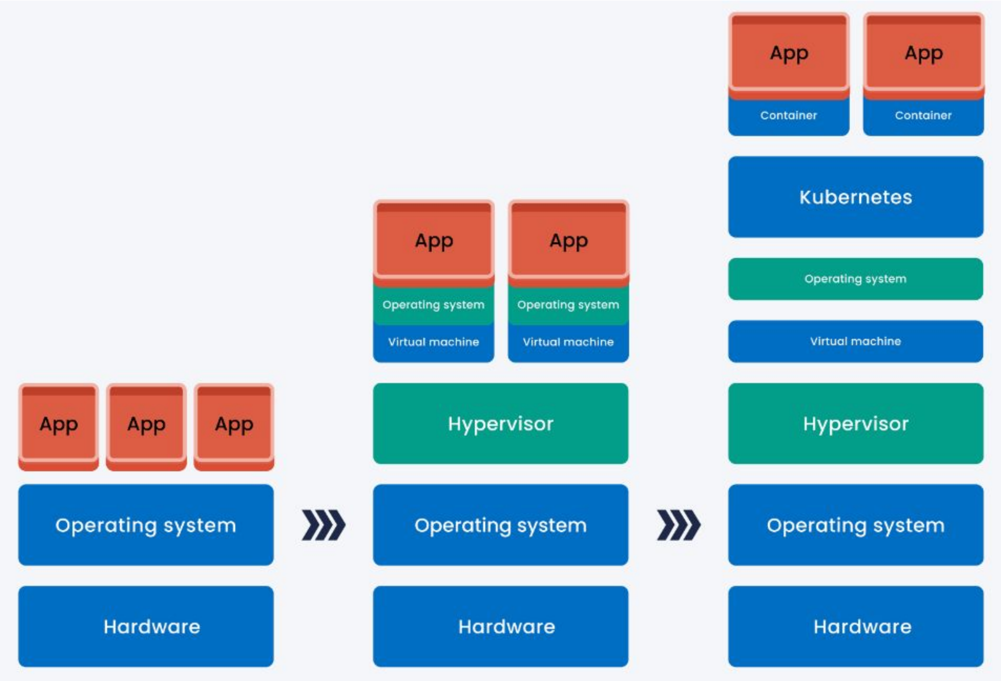


Figure: Evolution of Infrastructure [1]

Background: new needs sensitive to hypervisor overhead

Machine Learning as a Service (MLaaS)

- Training and Inference Tasks
- Real-Time Analytics

High-Performance Computing (HPC)

- Scientific Simulations
- Financial Modeling

Big Data Analytics

- Data Processing Pipelines
- Streaming Analytics

Real-Time Applications

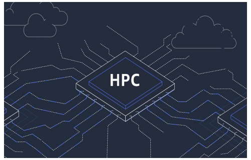
- Online Gaming
- Augmented Reality (AR) and Virtual Reality (VR)

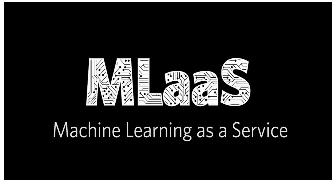
Database as a Service (DBaas)

- High-Transaction Databases
- In-Memory Databases

Edge Computing

- IoT Devices
- Autonomous Vehicles





Bare metal k8s

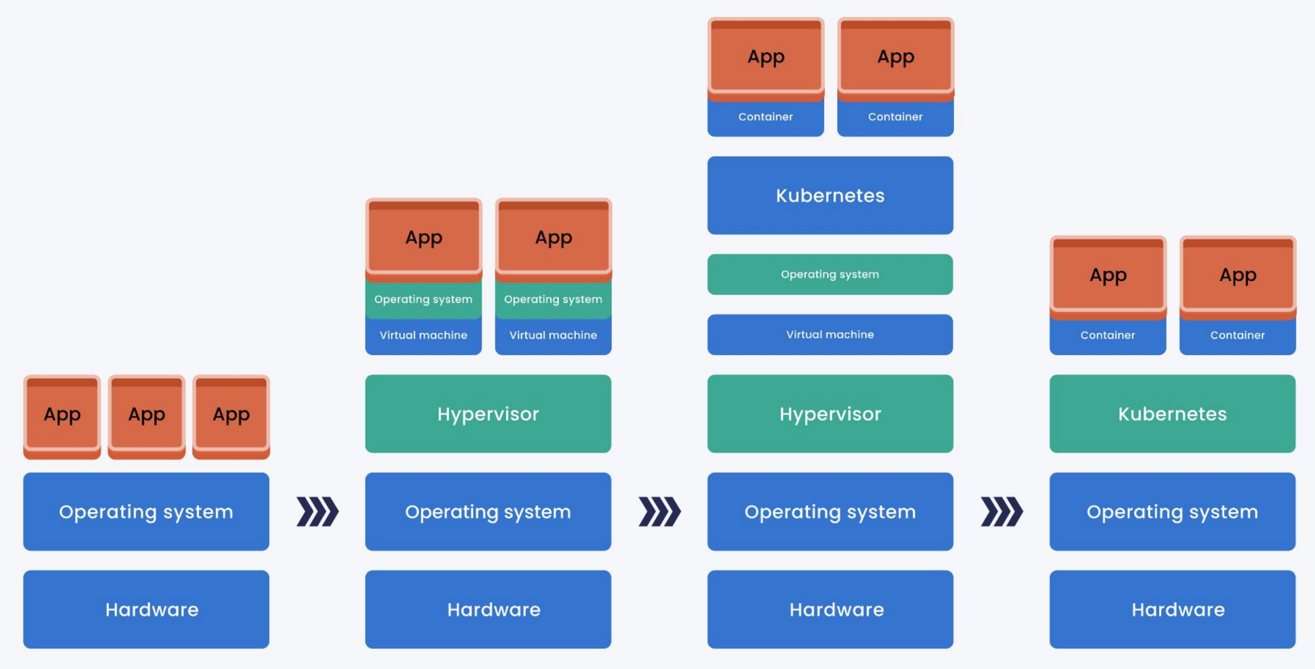


Figure: Evolution of Infrastructure [1]

Bare metal k8s - benefits

Deploying and Running k8s on top of Bare Metal Machines instead of Virtual Machines means one layer less to manage: the hypervisor.

Eliminating the hypervisor with bare metal servers can offer:

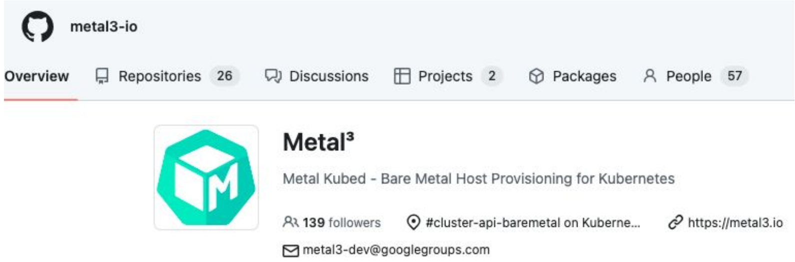
- Increased performance, particularly for specialized workloads like big data and AI.
- Better resource utilization, by as much as 20% [1].
- Savings, both in software licensing and operational efficiencies.
- Reduced risk through fewer technical and commercial dependencies.
- Better visibility into the health of your hardware.
- …

Bare metal k8s - challenges

- How to provision a large number of bare metal machines
  - Solution: **Metal^3** - bare metal server lifecycle management
- How to provision, upgrade, and operate multiple k8s clusters
  - Solution: **Cluster API (CAPI)** - cluster lifecycle management
- Configuration (Networking) Complexity
  - Compare with the relative simplicity of managing VM images, configuring **networking**, **storage**, and other resources in a bare metal Kubernetes environment provides more flexibility but can be more complex.

Metal^3 [2][3]

Metal^3 (pronounced “metal cubed”) is an open-source project that provides a set of tools for managing bare-metal infrastructure using Kubernetes [2].



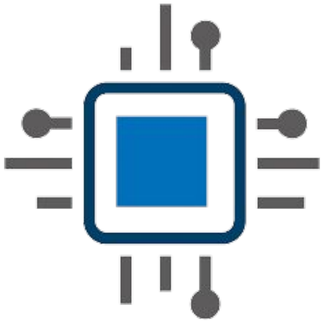
Metal^3

Metal Kubed - Bare Metal Host Provisioning for Kubernetes

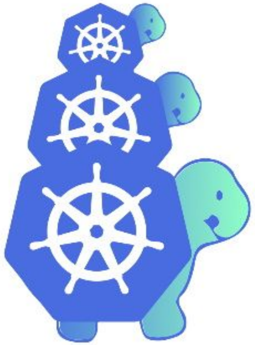
139 followers #cluster-api-baremetal on Kuberne... https://metal3.io

metal3-dev@googlegroups.com

Ironic

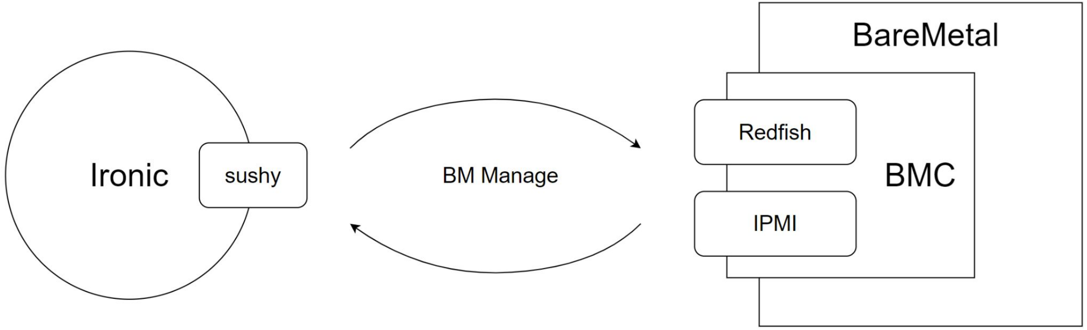


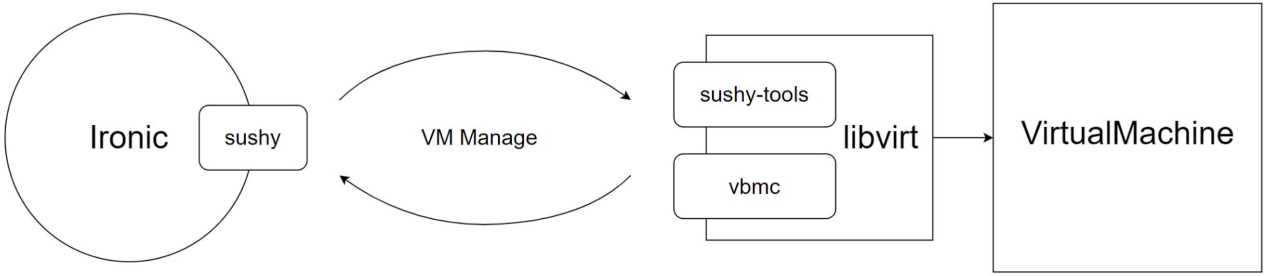
Bare Metal Operator

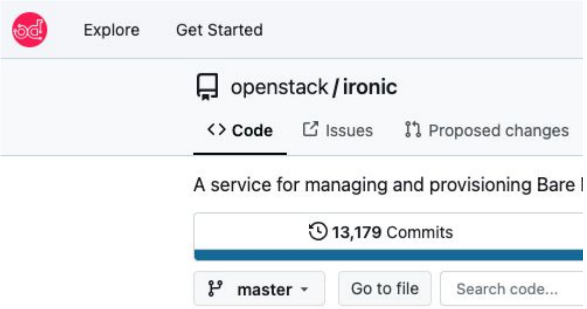


Cluster API Provider Metal^3

How Metal³ manages (bare metal) machines







openstack / ironic

A service for managing and provisioning Bare I

13,179 Commits

IRONIC

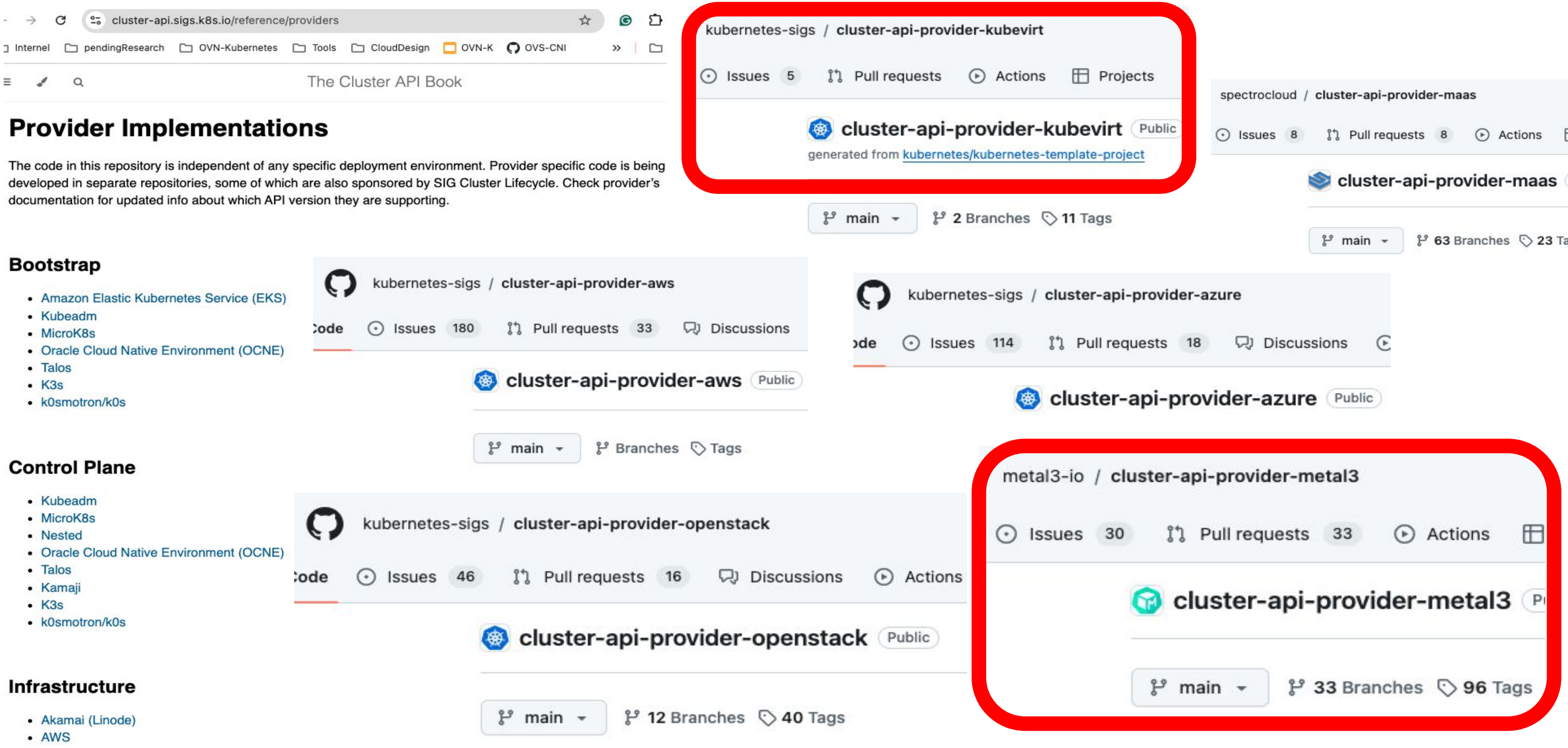
an OpenStack Community Project

CAPI [4]

CAPI is a tool for programmatically configuring and deploying k8s clusters on a variety of different infrastructures.

- Declarative k8s management
  - CAPI enables you to describe and manage the lifecycle of clusters using YAML definitions for resources such as Cluster and Machine (**Like defining Pods and Services in Kubernetes using YAML files**).
- **Platform Diversity**
  - CAPI supports multiple infrastructure providers, including bare metal servers, public clouds, and private clouds.

CAPI: platform diversity



Provider Implementations

The code in this repository is independent of any specific deployment environment. Provider specific code is being developed in separate repositories, some of which are also sponsored by SIG Cluster Lifecycle. Check provider's documentation for updated info about which API version they are supporting.

Bootstrap

- Amazon Elastic Kubernetes Service (EKS)
- Kubeadm
- MicroK8s
- Oracle Cloud Native Environment (OCNE)
- Talos
- K3s
- k0smotron/k0s

Control Plane

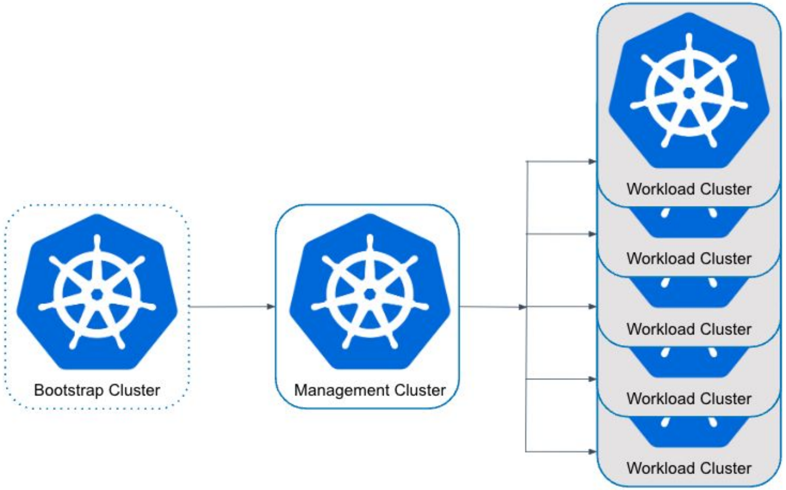
- Kubeadm
- MicroK8s
- Nested
- Oracle Cloud Native Environment (OCNE)
- Talos
- Kamaji
- K3s
- k0smotron/k0s

Infrastructure

- Akamai (Linode)
- AWS

kubernetes-sigs / cluster-api-provider-kubevirt
Issues 5
Pull requests
Actions
Projects
cluster-api-provider-kubevirt
Public
generated from kubernetes/kubernetes-template-project
main
2 Branches
11 Tags
spectrocloud / cluster-api-provider-maas
Issues 8
Pull requests 8
Actions
Proj
cluster-api-provider-maas
Public
main
63 Branches
23 Tags
kubernetes-sigs / cluster-api-provider-aws
Code
Issues 180
Pull requests 33
Discussions
cluster-api-provider-aws
Public
main
Branches
Tags
kubernetes-sigs / cluster-api-provider-azure
ode
Issues 114
Pull requests 18
Discussions
cluster-api-provider-azure
Public
metal3-io / cluster-api-provider-metal3
Issues 30
Pull requests 33
Actions
cluster-api-provider-metal3
P
main
33 Branches
96 Tags
kubernetes-sigs / cluster-api-provider-openstack
Code
Issues 46
Pull requests 16
Discussions
Actions
cluster-api-provider-openstack
Public
main
12 Branches
40 Tags

CAPI: deployment (1)



Deploying CAPI involves two k8s clusters: one is a temporary cluster called the **bootstrap cluster** that will be discarded later, which creates a second cluster that becomes the permanent CAPI **management cluster**.

Management cluster: responsible for creating and overseeing other clusters through Cluster API. They are your agents for infrastructure management, focusing entirely on provisioning, monitoring, and managing other clusters.

Workload cluster(s): handle application workloads for users, running microservices, and handling requests.

CAPI: deployment (2)

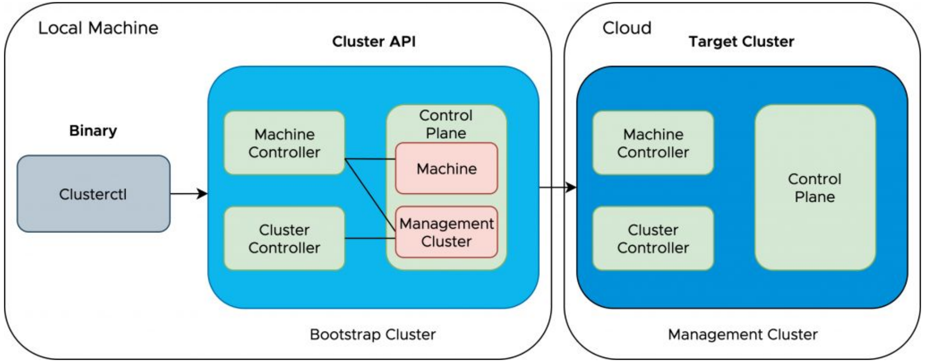


Figure: Deployment of Management Cluster [5]

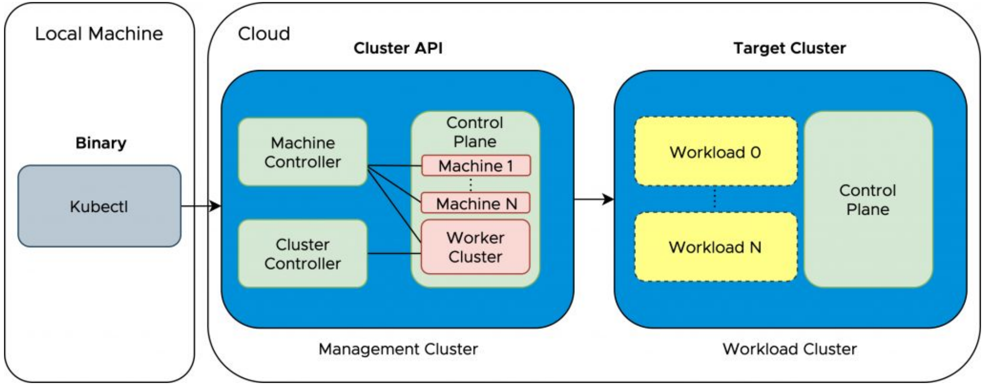
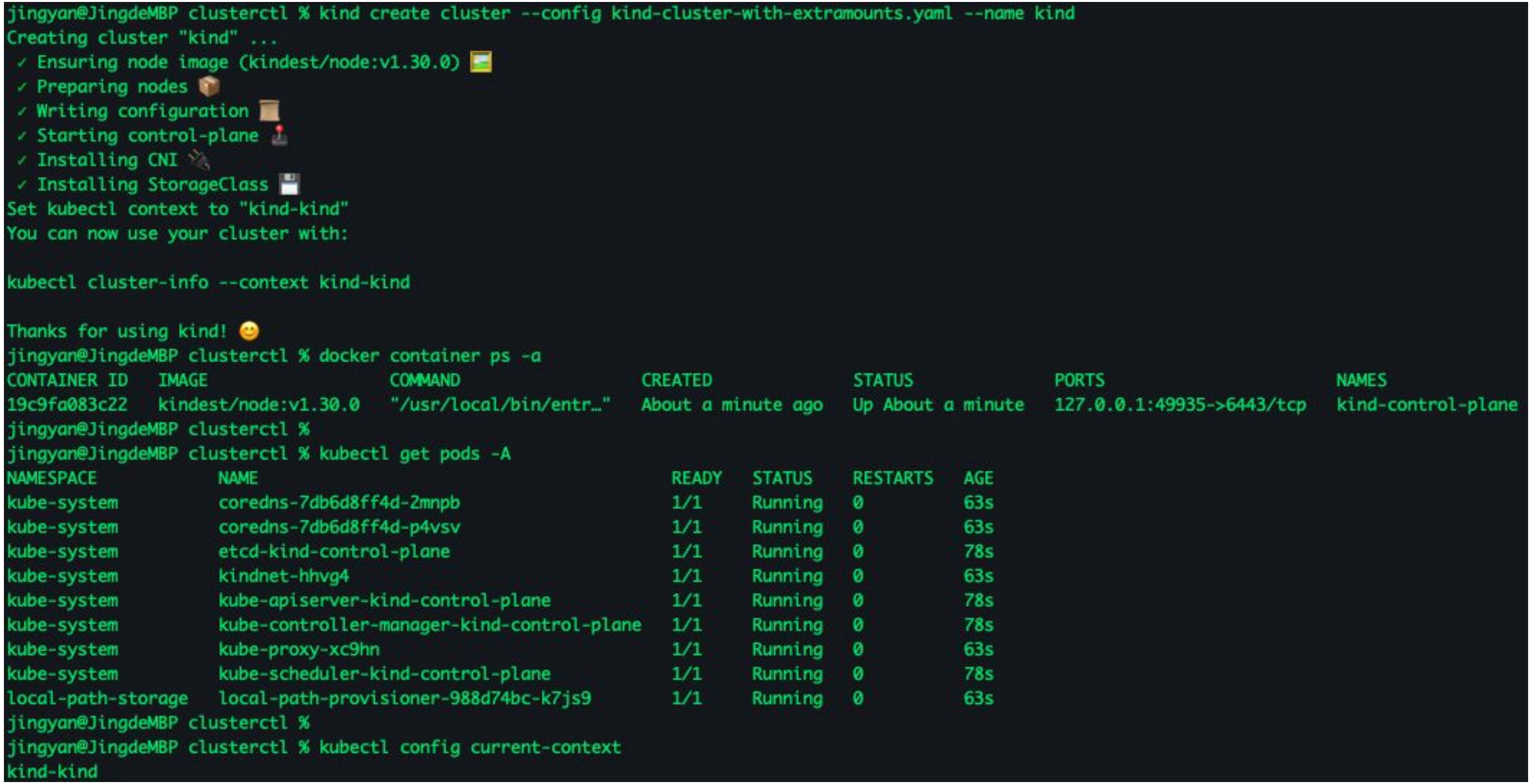


Figure: Deployment of Workload Cluster [5]

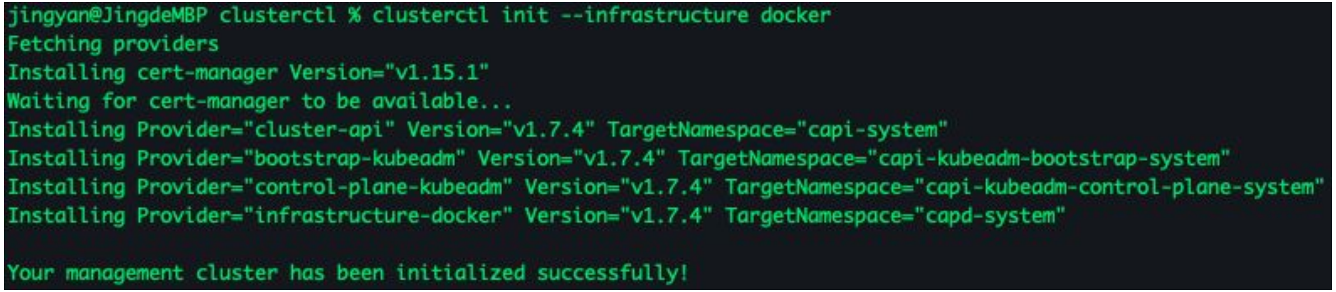
- Light blue: a Kubernetes in Docker (KinD) cluster.
- Dark blue: a cluster deployed to an infrastructure provider.
- Green: a controller or set of controllers.
- Red: CustomResourceDefinitions (CRDs).
- Yellow: user workloads.

CAPI: set up temporary bootstrap k8s cluster

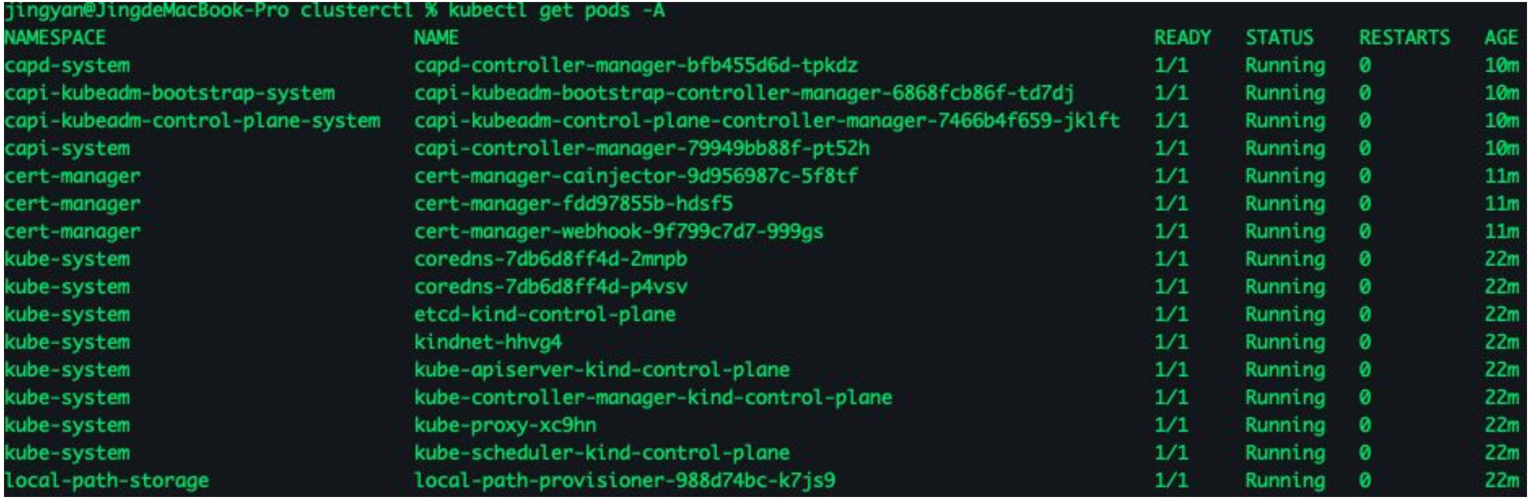


```
jingyan@JingdeMBP clusterctl % kind create cluster --config kind-cluster-with-extramounts.yaml --name kind
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.30.0) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Thanks for using kind! 😊
jingyan@JingdeMBP clusterctl % docker container ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS                       NAMES
19c9fa083c22   kindest/node:v1.30.0   "/usr/local/bin/entr…"   About a minute ago   Up About a minute   127.0.0.1:49935->6443/tcp   kind-control-plane
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl get pods -A
NAMESPACE            NAME                                         READY   STATUS    RESTARTS   AGE
kube-system          coredns-7db6d8ff4d-2mnpb                     1/1     Running   0          63s
kube-system          coredns-7db6d8ff4d-p4vsv                     1/1     Running   0          63s
kube-system          etcd-kind-control-plane                      1/1     Running   0          78s
kube-system          kindnet-hhvg4                                1/1     Running   0          63s
kube-system          kube-apiserver-kind-control-plane            1/1     Running   0          78s
kube-system          kube-controller-manager-kind-control-plane   1/1     Running   0          78s
kube-system          kube-proxy-xc9hn                             1/1     Running   0          63s
kube-system          kube-scheduler-kind-control-plane            1/1     Running   0          78s
local-path-storage   local-path-provisioner-988d74bc-k7js9        1/1     Running   0          63s
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl config current-context
kind-kind
```

CAPI: set up management k8s cluster

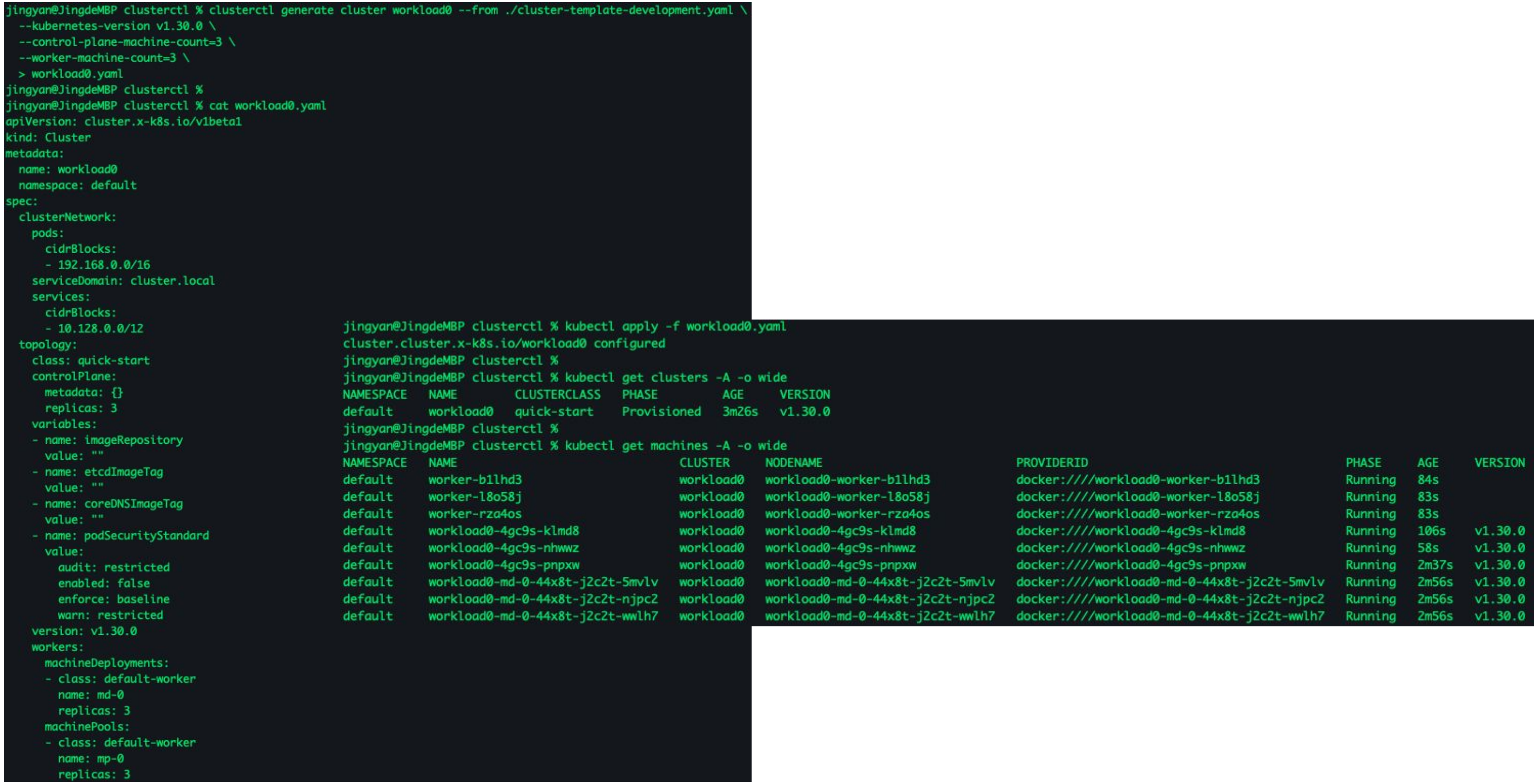


```
jingyan@JingdeMBP clusterctl % clusterctl init --infrastructure docker
Fetching providers
Installing cert-manager Version="v1.15.1"
Waiting for cert-manager to be available...
Installing Provider="cluster-api" Version="v1.7.4" TargetNamespace="capi-system"
Installing Provider="bootstrap-kubeadm" Version="v1.7.4" TargetNamespace="capi-kubeadm-bootstrap-system"
Installing Provider="control-plane-kubeadm" Version="v1.7.4" TargetNamespace="capi-kubeadm-control-plane-system"
Installing Provider="infrastructure-docker" Version="v1.7.4" TargetNamespace="capd-system"

Your management cluster has been initialized successfully!
```



```
jingyan@JingdeMacBook-Pro clusterctl % kubectl get pods -A
NAMESPACE                           NAME                                                             READY   STATUS    RESTARTS   AGE
capd-system                         capd-controller-manager-bfb455d6d-tpkdz                          1/1     Running   0          10m
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager-6868fcb86f-td7dj       1/1     Running   0          10m
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-manager-7466b4f659-jklft   1/1     Running   0          10m
capi-system                         capi-controller-manager-79949bb88f-pt52h                         1/1     Running   0          10m
cert-manager                        cert-manager-cainjector-9d956987c-5f8tf                          1/1     Running   0          11m
cert-manager                        cert-manager-fdd97855b-hdsf5                                     1/1     Running   0          11m
cert-manager                        cert-manager-webhook-9f799c7d7-999gs                             1/1     Running   0          11m
kube-system                         coredns-7db6d8ff4d-2mnpb                                         1/1     Running   0          22m
kube-system                         coredns-7db6d8ff4d-p4vsv                                         1/1     Running   0          22m
kube-system                         etcd-kind-control-plane                                          1/1     Running   0          22m
kube-system                         kindnet-hhvg4                                                    1/1     Running   0          22m
kube-system                         kube-apiserver-kind-control-plane                                1/1     Running   0          22m
kube-system                         kube-controller-manager-kind-control-plane                       1/1     Running   0          22m
kube-system                         kube-proxy-xc9hn                                                 1/1     Running   0          22m
kube-system                         kube-scheduler-kind-control-plane                                1/1     Running   0          22m
local-path-storage                  local-path-provisioner-988d74bc-k7js9                            1/1     Running   0          22m
```

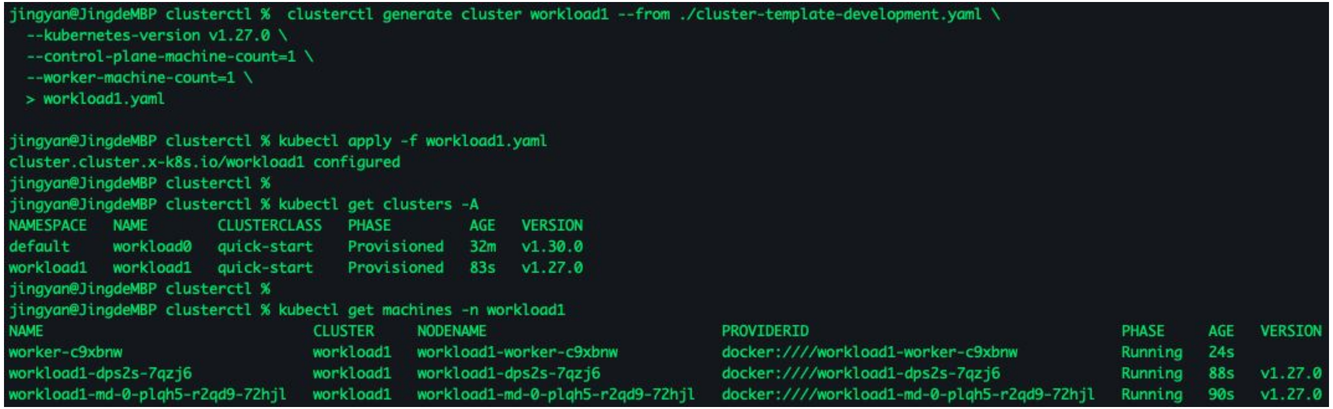
CAPI: set up workload k8s clusters (1)



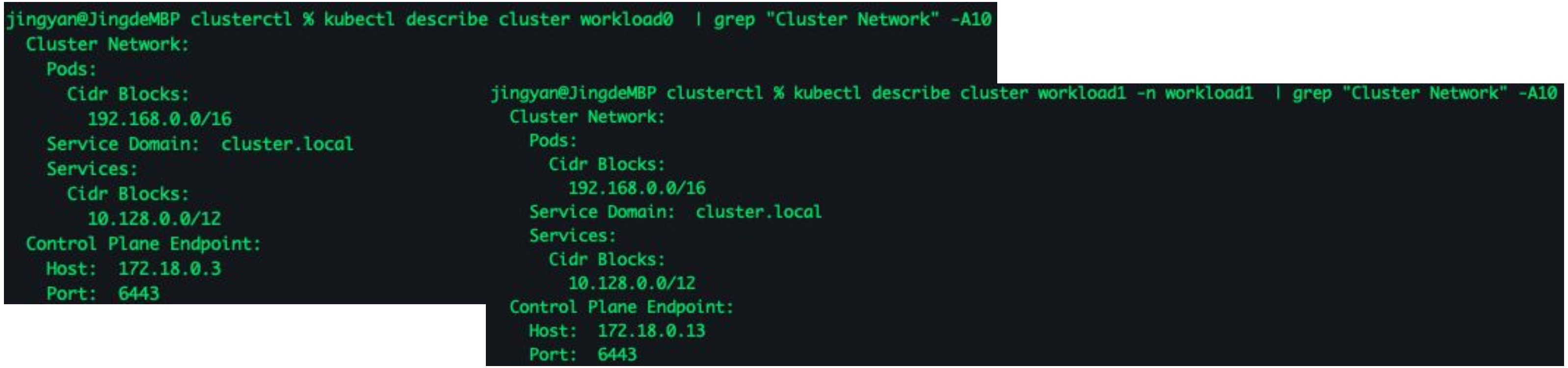
```
jingyan@JingdeMBP clusterctl % clusterctl generate cluster workload0 --from ./cluster-template-development.yaml \
  --kubernetes-version v1.30.0 \
  --control-plane-machine-count=3 \
  --worker-machine-count=3 \
  > workload0.yaml
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % cat workload0.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: workload0
  namespace: default
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.168.0.0/16
    serviceDomain: cluster.local
    services:
      cidrBlocks:
      - 10.128.0.0/12
  topology:
    class: quick-start
    controlPlane:
      metadata: {}
      replicas: 3
    variables:
    - name: imageRepository
      value: ""
    - name: etcdImageTag
      value: ""
    - name: coreDNSImageTag
      value: ""
    - name: podSecurityStandard
      value:
        audit: restricted
        enabled: false
        enforce: baseline
        warn: restricted
    version: v1.30.0
    workers:
      machineDeployments:
      - class: default-worker
        name: md-0
        replicas: 3
      machinePools:
      - class: default-worker
        name: mp-0
        replicas: 3
```

```
jingyan@JingdeMBP clusterctl % kubectl apply -f workload0.yaml
cluster.cluster.x-k8s.io/workload0 configured
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl get clusters -A -o wide
NAMESPACE   NAME        CLUSTERCLASS   PHASE          AGE     VERSION
default     workload0   quick-start    Provisioned    3m26s   v1.30.0
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl get machines -A -o wide
NAMESPACE   NAME                                CLUSTER     NODENAME                            PROVIDERID                                     PHASE     AGE     VERSION
default     worker-b1lhd3                       workload0   workload0-worker-b1lhd3             docker:////workload0-worker-b1lhd3             Running   84s
default     worker-l8o58j                       workload0   workload0-worker-l8o58j             docker:////workload0-worker-l8o58j             Running   83s
default     worker-rza4os                       workload0   workload0-worker-rza4os             docker:////workload0-worker-rza4os             Running   83s
default     workload0-4gc9s-klmd8               workload0   workload0-4gc9s-klmd8               docker:////workload0-4gc9s-klmd8               Running   106s    v1.30.0
default     workload0-4gc9s-nhwwz               workload0   workload0-4gc9s-nhwwz               docker:////workload0-4gc9s-nhwwz               Running   58s     v1.30.0
default     workload0-4gc9s-pnpxw               workload0   workload0-4gc9s-pnpxw               docker:////workload0-4gc9s-pnpxw               Running   2m37s   v1.30.0
default     workload0-md-0-44x8t-j2c2t-5mvlv    workload0   workload0-md-0-44x8t-j2c2t-5mvlv    docker:////workload0-md-0-44x8t-j2c2t-5mvlv    Running   2m56s   v1.30.0
default     workload0-md-0-44x8t-j2c2t-njpc2    workload0   workload0-md-0-44x8t-j2c2t-njpc2    docker:////workload0-md-0-44x8t-j2c2t-njpc2    Running   2m56s   v1.30.0
default     workload0-md-0-44x8t-j2c2t-wwlh7    workload0   workload0-md-0-44x8t-j2c2t-wwlh7    docker:////workload0-md-0-44x8t-j2c2t-wwlh7    Running   2m56s   v1.30.0
```

CAPI: set up workload k8s clusters (2)



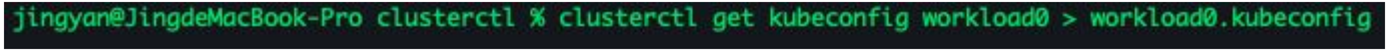
```
jingyan@JingdeMBP clusterctl %  clusterctl generate cluster workload1 --from ./cluster-template-development.yaml \
  --kubernetes-version v1.27.0 \
  --control-plane-machine-count=1 \
  --worker-machine-count=1 \
  > workload1.yaml

jingyan@JingdeMBP clusterctl % kubectl apply -f workload1.yaml
cluster.cluster.x-k8s.io/workload1 configured
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl get clusters -A
NAMESPACE   NAME        CLUSTERCLASS   PHASE         AGE   VERSION
default     workload0   quick-start    Provisioned   32m   v1.30.0
workload1   workload1   quick-start    Provisioned   83s   v1.27.0
jingyan@JingdeMBP clusterctl %
jingyan@JingdeMBP clusterctl % kubectl get machines -n workload1
NAME                              CLUSTER     NODENAME                          PROVIDERID                                    PHASE     AGE   VERSION
worker-c9xbnw                     workload1   workload1-worker-c9xbnw           docker:////workload1-worker-c9xbnw            Running   24s
workload1-dps2s-7qzj6             workload1   workload1-dps2s-7qzj6             docker:////workload1-dps2s-7qzj6              Running   88s   v1.27.0
workload1-md-0-plqh5-r2qd9-72hjl  workload1   workload1-md-0-plqh5-r2qd9-72hjl  docker:////workload1-md-0-plqh5-r2qd9-72hjl   Running   90s   v1.27.0
```



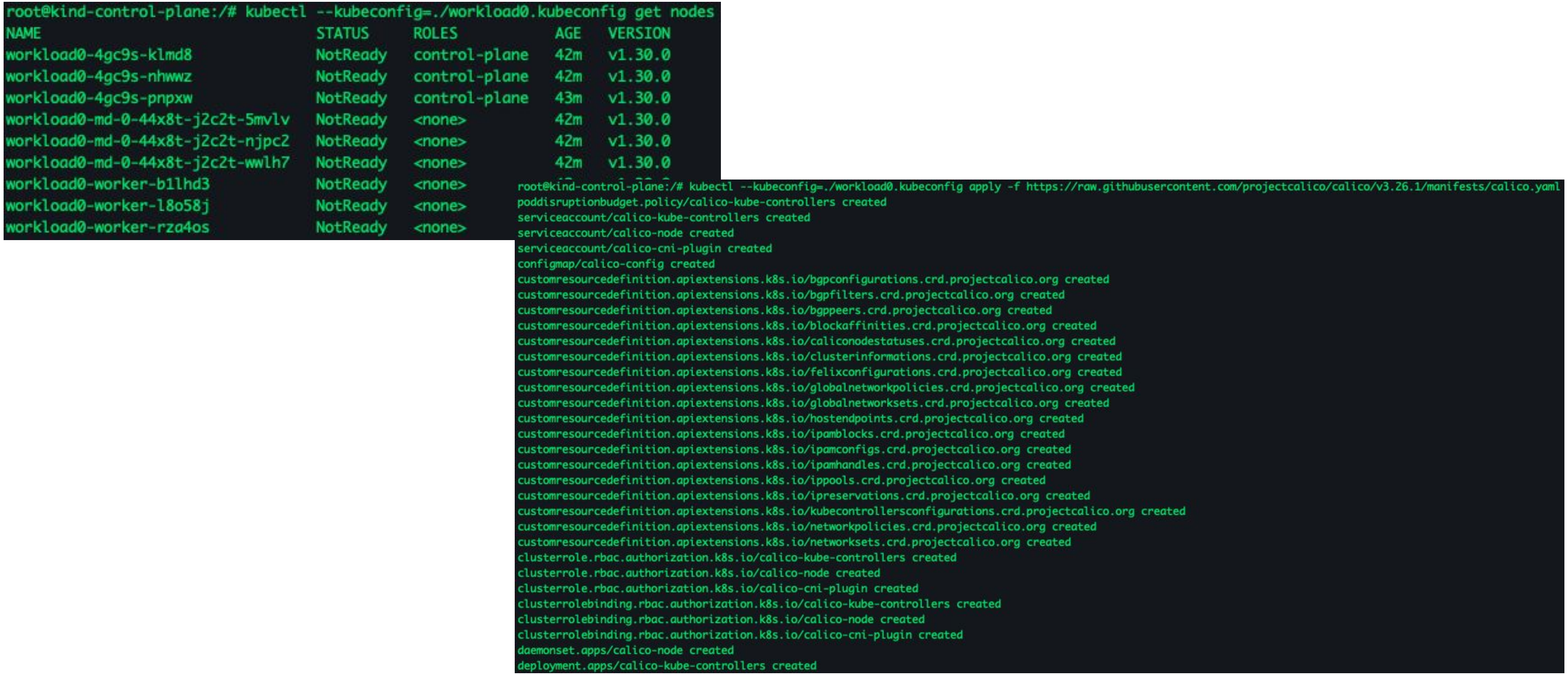
```
jingyan@JingdeMBP clusterctl % kubectl describe cluster workload0  | grep "Cluster Network" -A10
  Cluster Network:
    Pods:
      Cidr Blocks:
        192.168.0.0/16
    Service Domain:  cluster.local
    Services:
      Cidr Blocks:
        10.128.0.0/12
  Control Plane Endpoint:
    Host:  172.18.0.3
    Port:  6443
```

```
jingyan@JingdeMBP clusterctl % kubectl describe cluster workload1 -n workload1  | grep "Cluster Network" -A10
  Cluster Network:
    Pods:
      Cidr Blocks:
        192.168.0.0/16
    Service Domain:  cluster.local
    Services:
      Cidr Blocks:
        10.128.0.0/12
  Control Plane Endpoint:
    Host:  172.18.0.13
    Port:  6443
```

CAPI: deploy cni for workload0 (1)



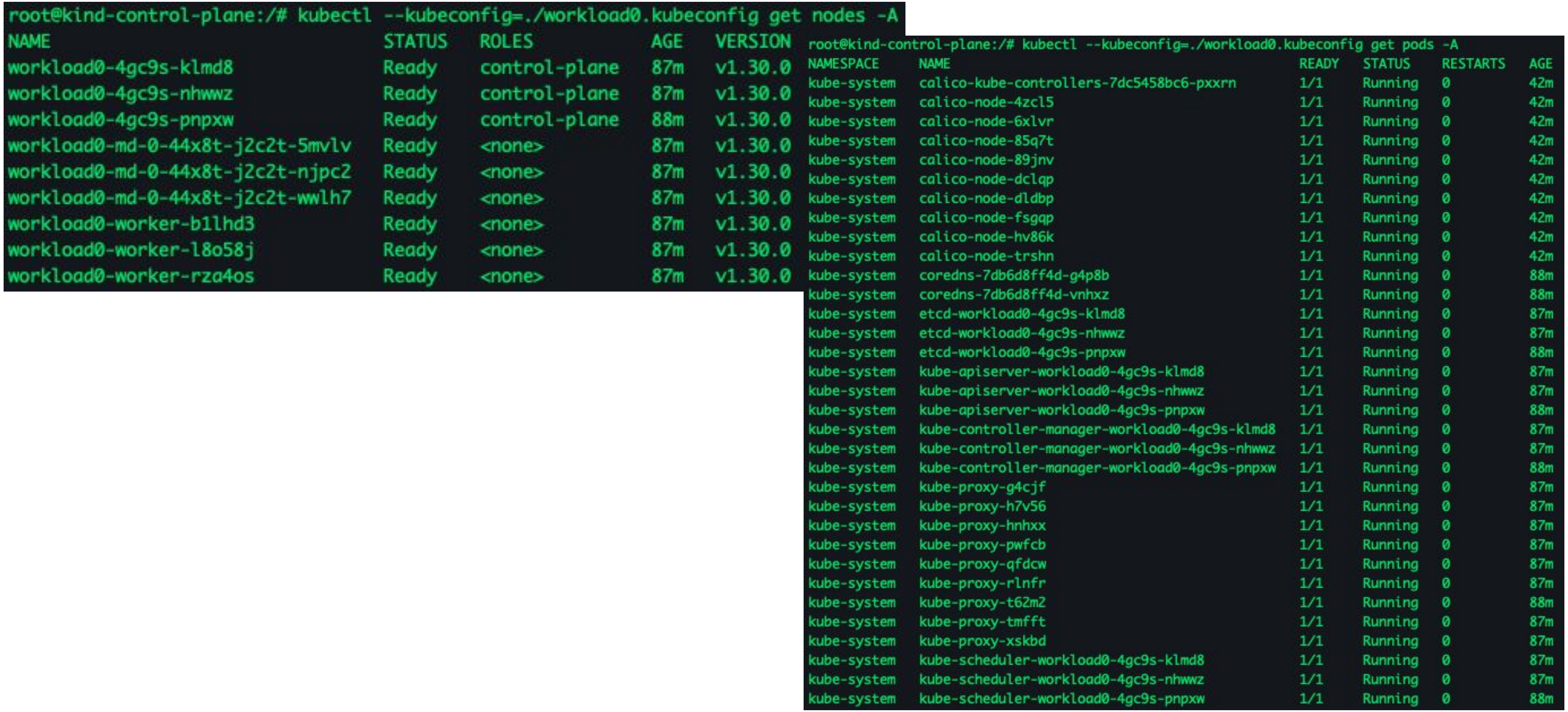
```
jingyan@JingdeMacBook-Pro clusterctl % clusterctl get kubeconfig workload0 > workload0.kubeconfig
```



```
root@kind-control-plane:/# kubectl --kubeconfig=./workload0.kubeconfig get nodes
NAME                              STATUS     ROLES           AGE   VERSION
workload0-4gc9s-klmd8             NotReady   control-plane   42m   v1.30.0
workload0-4gc9s-nhwwz             NotReady   control-plane   42m   v1.30.0
workload0-4gc9s-pnpxw             NotReady   control-plane   43m   v1.30.0
workload0-md-0-44x8t-j2c2t-5mvlv  NotReady   <none>          42m   v1.30.0
workload0-md-0-44x8t-j2c2t-njpc2  NotReady   <none>          42m   v1.30.0
workload0-md-0-44x8t-j2c2t-wwlh7  NotReady   <none>          42m   v1.30.0
workload0-worker-b1lhd3           NotReady   <none>
workload0-worker-l8o58j           NotReady   <none>
workload0-worker-rza4os           NotReady   <none>
```

```
root@kind-control-plane:/# kubectl --kubeconfig=./workload0.kubeconfig apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

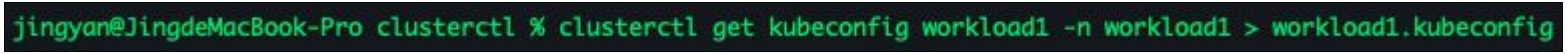
CAPI: deploy cni for workload0 (2)



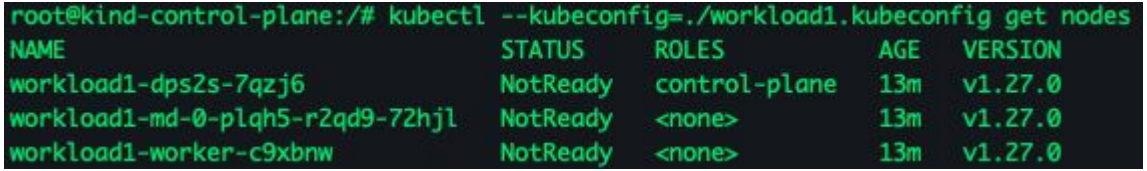
```
root@kind-control-plane:/# kubectl --kubeconfig=./workload0.kubeconfig get nodes -A
NAME                              STATUS   ROLES           AGE   VERSION
workload0-4gc9s-klmd8             Ready    control-plane   87m   v1.30.0
workload0-4gc9s-nhwwz             Ready    control-plane   87m   v1.30.0
workload0-4gc9s-pnpxw             Ready    control-plane   88m   v1.30.0
workload0-md-0-44x8t-j2c2t-5mvlv  Ready    <none>          87m   v1.30.0
workload0-md-0-44x8t-j2c2t-njpc2  Ready    <none>          87m   v1.30.0
workload0-md-0-44x8t-j2c2t-wwlh7  Ready    <none>          87m   v1.30.0
workload0-worker-b1lhd3           Ready    <none>          87m   v1.30.0
workload0-worker-l8o58j           Ready    <none>          87m   v1.30.0
workload0-worker-rza4os           Ready    <none>          87m   v1.30.0
```

```
root@kind-control-plane:/# kubectl --kubeconfig=./workload0.kubeconfig get pods -A
NAMESPACE     NAME                                              READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-7dc5458bc6-pxxrn          1/1     Running   0          42m
kube-system   calico-node-4zcl5                                 1/1     Running   0          42m
kube-system   calico-node-6xlvr                                 1/1     Running   0          42m
kube-system   calico-node-85q7t                                 1/1     Running   0          42m
kube-system   calico-node-89jnv                                 1/1     Running   0          42m
kube-system   calico-node-dclqp                                 1/1     Running   0          42m
kube-system   calico-node-dldbp                                 1/1     Running   0          42m
kube-system   calico-node-fsgqp                                 1/1     Running   0          42m
kube-system   calico-node-hv86k                                 1/1     Running   0          42m
kube-system   calico-node-trshn                                 1/1     Running   0          42m
kube-system   coredns-7db6d8ff4d-g4p8b                          1/1     Running   0          88m
kube-system   coredns-7db6d8ff4d-vnhxz                          1/1     Running   0          88m
kube-system   etcd-workload0-4gc9s-klmd8                        1/1     Running   0          87m
kube-system   etcd-workload0-4gc9s-nhwwz                        1/1     Running   0          87m
kube-system   etcd-workload0-4gc9s-pnpxw                        1/1     Running   0          88m
kube-system   kube-apiserver-workload0-4gc9s-klmd8              1/1     Running   0          87m
kube-system   kube-apiserver-workload0-4gc9s-nhwwz              1/1     Running   0          87m
kube-system   kube-apiserver-workload0-4gc9s-pnpxw              1/1     Running   0          88m
kube-system   kube-controller-manager-workload0-4gc9s-klmd8     1/1     Running   0          87m
kube-system   kube-controller-manager-workload0-4gc9s-nhwwz     1/1     Running   0          87m
kube-system   kube-controller-manager-workload0-4gc9s-pnpxw     1/1     Running   0          88m
kube-system   kube-proxy-g4cjf                                  1/1     Running   0          87m
kube-system   kube-proxy-h7v56                                  1/1     Running   0          87m
kube-system   kube-proxy-hnhxx                                  1/1     Running   0          87m
kube-system   kube-proxy-pwfcb                                  1/1     Running   0          87m
kube-system   kube-proxy-qfdcw                                  1/1     Running   0          87m
kube-system   kube-proxy-rlnfr                                  1/1     Running   0          87m
kube-system   kube-proxy-t62m2                                  1/1     Running   0          88m
kube-system   kube-proxy-tmfft                                  1/1     Running   0          87m
kube-system   kube-proxy-xskbd                                  1/1     Running   0          87m
kube-system   kube-scheduler-workload0-4gc9s-klmd8              1/1     Running   0          87m
kube-system   kube-scheduler-workload0-4gc9s-nhwwz              1/1     Running   0          87m
kube-system   kube-scheduler-workload0-4gc9s-pnpxw              1/1     Running   0          88m
```

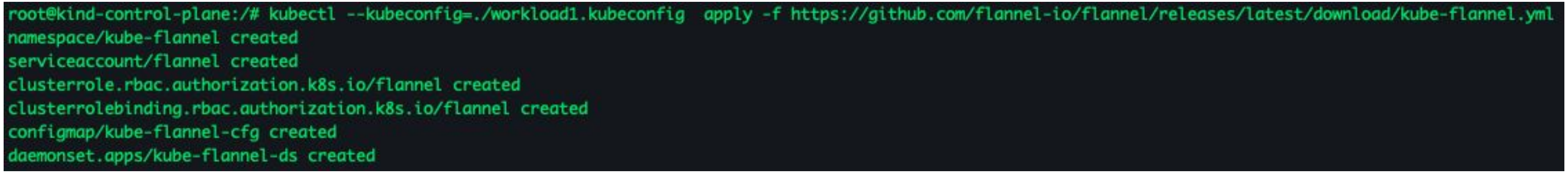
CAPI: deploy cni for workload1



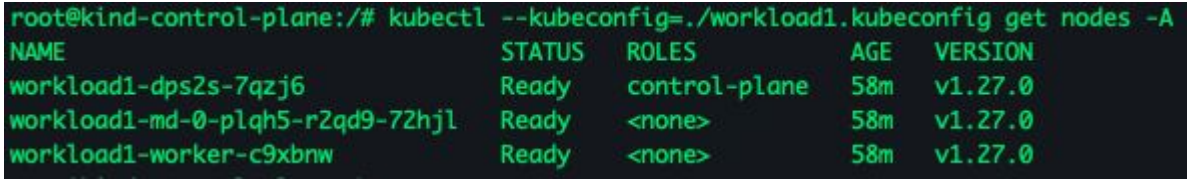
```
jingyan@JingdeMacBook-Pro clusterctl % clusterctl get kubeconfig workload1 -n workload1 > workload1.kubeconfig
```



```
root@kind-control-plane:/# kubectl --kubeconfig=./workload1.kubeconfig get nodes
NAME                               STATUS     ROLES           AGE   VERSION
workload1-dps2s-7qzj6              NotReady   control-plane   13m   v1.27.0
workload1-md-0-plqh5-r2qd9-72hjl   NotReady   <none>          13m   v1.27.0
workload1-worker-c9xbnw            NotReady   <none>          13m   v1.27.0
```

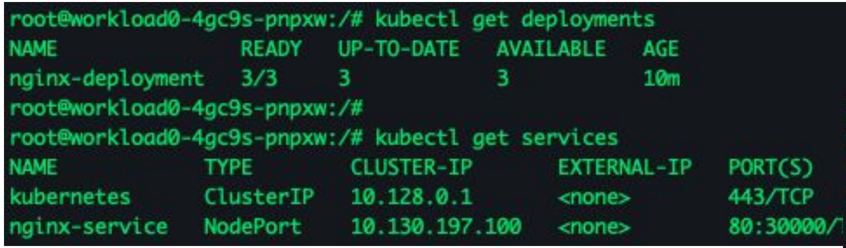


```
root@kind-control-plane:/# kubectl --kubeconfig=./workload1.kubeconfig  apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

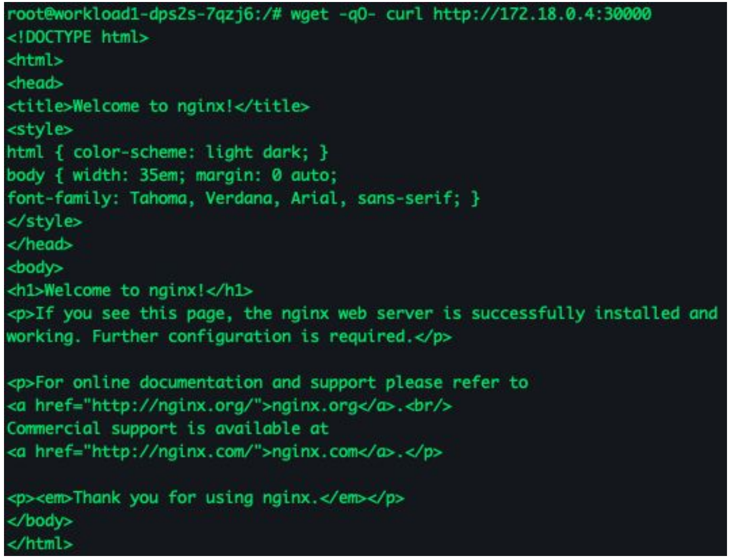


```
root@kind-control-plane:/# kubectl --kubeconfig=./workload1.kubeconfig get nodes -A
NAME                               STATUS   ROLES           AGE   VERSION
workload1-dps2s-7qzj6              Ready    control-plane   58m   v1.27.0
workload1-md-0-plqh5-r2qd9-72hjl   Ready    <none>          58m   v1.27.0
workload1-worker-c9xbnw            Ready    <none>          58m   v1.27.0
```

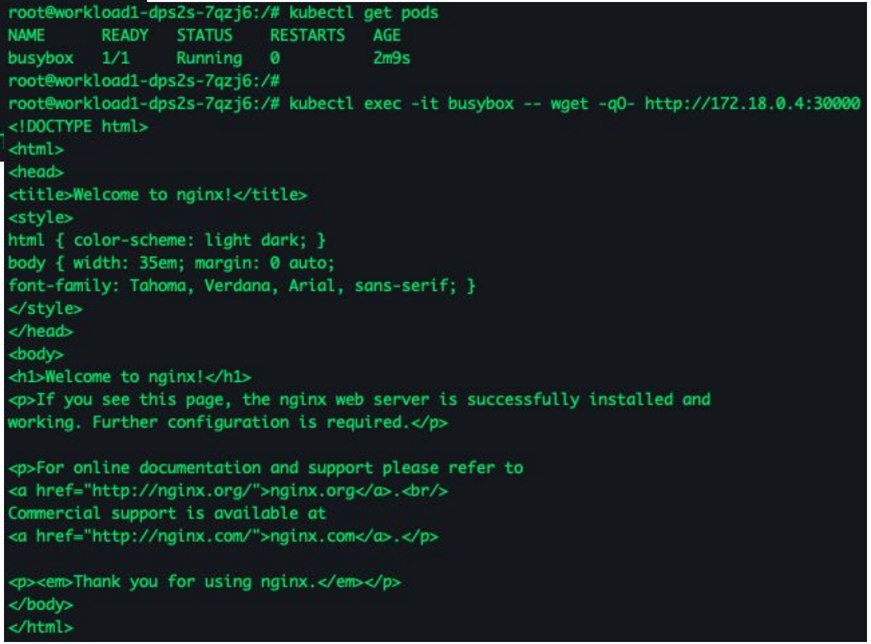
CAPI: deploy service and access service



```
root@workload0-4gc9s-pnpxw:/# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           10m
root@workload0-4gc9s-pnpxw:/#
root@workload0-4gc9s-pnpxw:/# kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
kubernetes      ClusterIP   10.128.0.1       <none>        443/TCP
nginx-service   NodePort    10.130.197.100   <none>        80:30000/
```

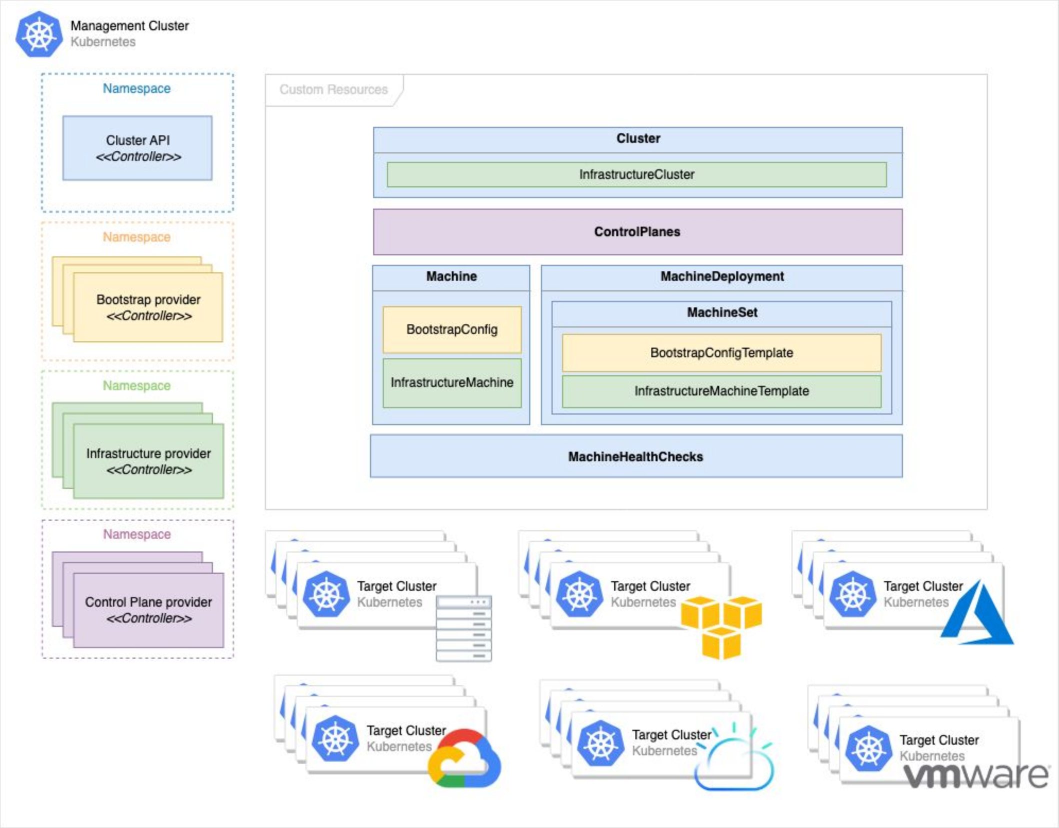


```
root@workload1-dps2s-7qzj6:/# wget -qO- curl http://172.18.0.4:30000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```



```
root@workload1-dps2s-7qzj6:/# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
busybox   1/1     Running   0          2m9s
root@workload1-dps2s-7qzj6:/#
root@workload1-dps2s-7qzj6:/# kubectl exec -it busybox -- wget -qO- http://172.18.0.4:30000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

How CAPI works [4]



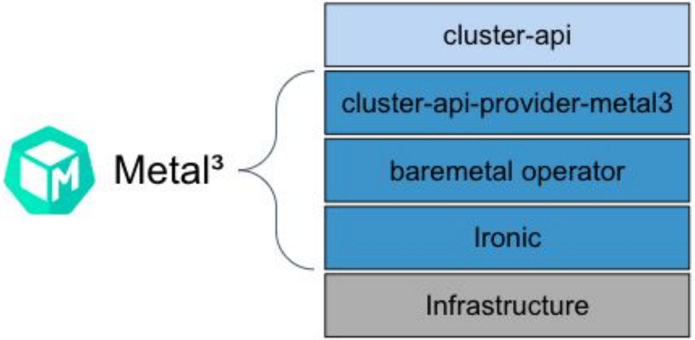
Cluster API Controllers: responsible for managing the lifecycle of the Kubernetes clusters created using the Cluster API. These controllers are responsible for provisioning, scaling, and deleting the clusters, and ensuring that the Workload clusters are in the desired state.

Bootstrap provider: responsible for bootstrapping (installing and configuring) the Kubernetes control plane components on a newly created cluster.

Infrastructure provider: responsible for the provisioning of infrastructure resources required by the Cluster or by Machines.

Custom Resources: Kubernetes resources used by the Cluster API to create, manage, and delete clusters. These resources include **Cluster, Machine, MachineSet, and MachineDeployment**.

Summary



Metal³

cluster-api

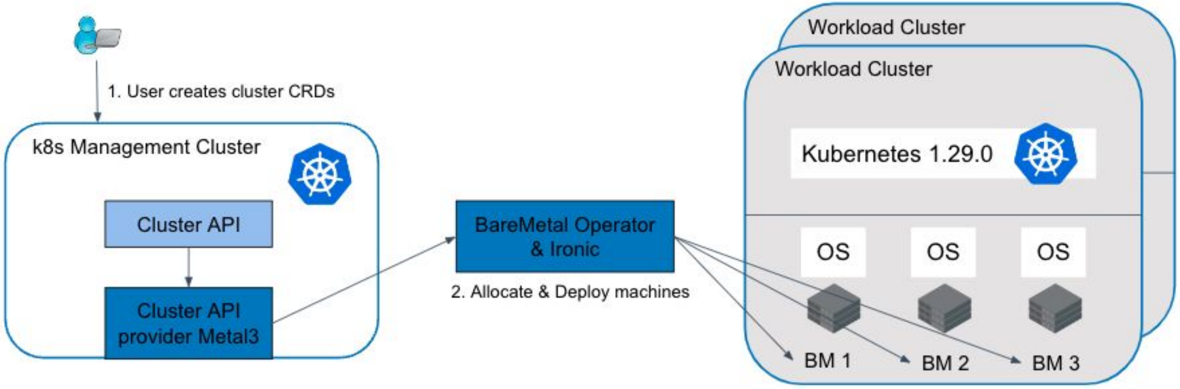
cluster-api-provider-metal3

baremetal operator

Ironic

Infrastructure

cluster-api: https://github.com/kubernetes-sigs/cluster-api
cluster-api-provider-metal3: https://github.com/metal3-io/cluster-api-provider-metal3
baremetal-operator: https://github.com/metal3-io/baremetal-operator/tree/main
ironic: https://github.com/metal3-io/ironic-image



1. User creates cluster CRDs

k8s Management Cluster

Cluster API

Cluster API provider Metal3

BareMetal Operator & Ironic

2. Allocate & Deploy machines

Workload Cluster

Workload Cluster

Kubernetes 1.29.0

OS

OS

OS

BM 1

BM 2

BM 3

3. PXE boots, installs OS and k8s clusters

Figure: Workflow of creating k8s CRDs on bare metal [1]

References

[1] https://www.spectrocloud.com/blog/introducing-bare-metal-kubernetes-what-you-need-to-know
[2] https://metal3.io/
[3] https://github.com/metal3-io
[4] https://cluster-api.sigs.k8s.io/introduction
[5] https://tanzu.vmware.com/content/blog/pattern-recognition-how-cluster-api-reveals-the-core-of-kubernetes