SIG Cluster Lifecycle

Overview and projects

Martin Vladev - SAP
Rostislav M. Georgiev - VMware

SIGs?

“Special Interest Groups (SIGs) are persistent open groups that focus on a part of the project.”

--- Kubernetes Community README

kubernetes

SIGs?

API Machinery
Architecture
Apps
Auth
Autoscaling
OpenStack
Scheduling
UI

Azure
Big Data
CLI
Cluster Lifecycle
Contributor Experience
PM
Service Catalog
VMware

Docs
IBM Cloud
Cloud Provider
Network
Cluster Ops
Release
Storage
Windows

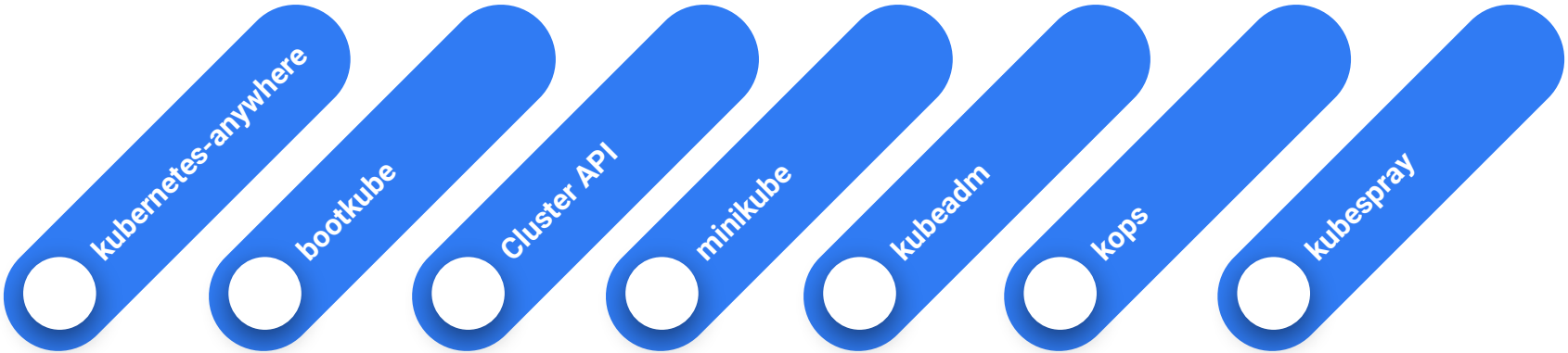
GCP
Instrumentation
Multicluster
Node
AWS
Scalability
Testing

SIG Cluster Lifecycle

*“**SIG Cluster Lifecycle**’s objective is to simplify creation, configuration, upgrade, downgrade, and teardown of Kubernetes clusters and their components.”*

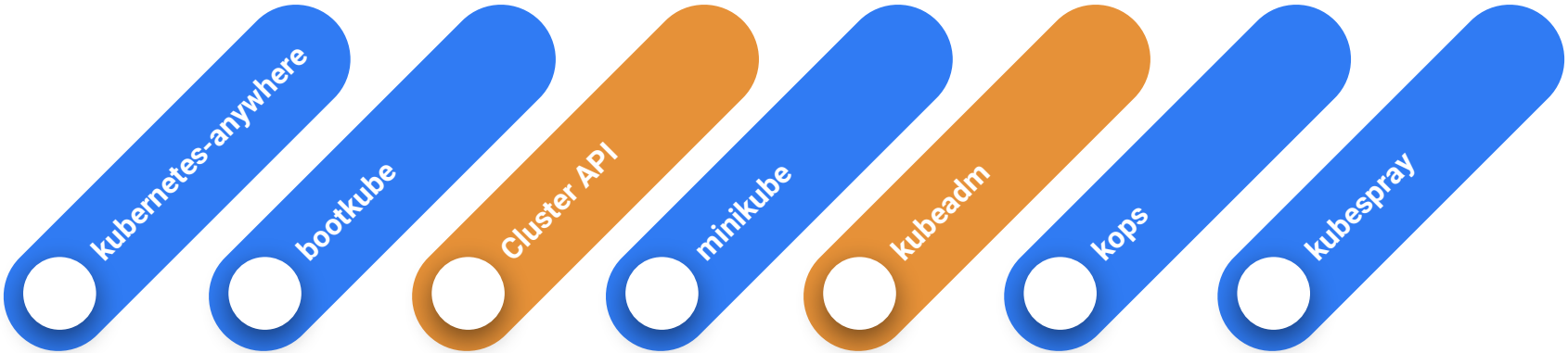
--- SIG Cluster Lifecycle Charter

SIG Cluster Lifecycle



kubernetes

SIG Cluster Lifecycle



kubernetes

KUBEADM

kubernetes

Kubeadm Design Goals

- Different deployment options
- Public & private clouds, VMs & bare metal
- Different Linux distros
- Different CRIs
- Flexible network options

Kubeadm Design Goals

- Upgrades
- Easy workflow
  - Provision one machine at a time
  - End users can use it
  - Automated deployment tools

Prerequisites

Machine 1

- Operating System

Machine 2

- Operating System

Machine 3

- Operating System

kubernetes

Prerequisites

Machine 1

- Container Runtime
- Operating System

Machine 2

- Container Runtime
- Operating System

Machine 3

- Container Runtime
- Operating System

kubernetes

Prerequisites

Machine 1

- Kubelet + kubeadm
- Container Runtime
- Operating System

Machine 2

- Kubelet + kubeadm
- Container Runtime
- Operating System

Machine 3

- Kubelet + kubeadm
- Container Runtime
- Operating System

Prerequisites

Control Plane

- Kubelet + kubeadm
- Container Runtime
- Operating System

Worker 1

- Kubelet + kubeadm
- Container Runtime
- Operating System

Worker 2

- Kubelet + kubeadm
- Container Runtime
- Operating System

kubernetes

kubeadm init

```
rosti@ubuntu:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] using Kubernetes version: v1.12.1
[preflight] running pre-flight checks
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
```

```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.217.129:6443 --token g7qg8j.vj4blgjdbsagr9x0 --discovery-token-ca-cert-hash sha256:ddf6a76a90edbd47ddbeb3fea4a06180d1f0b26eea9f91db42f207baadc2b43f
```

kubernetes

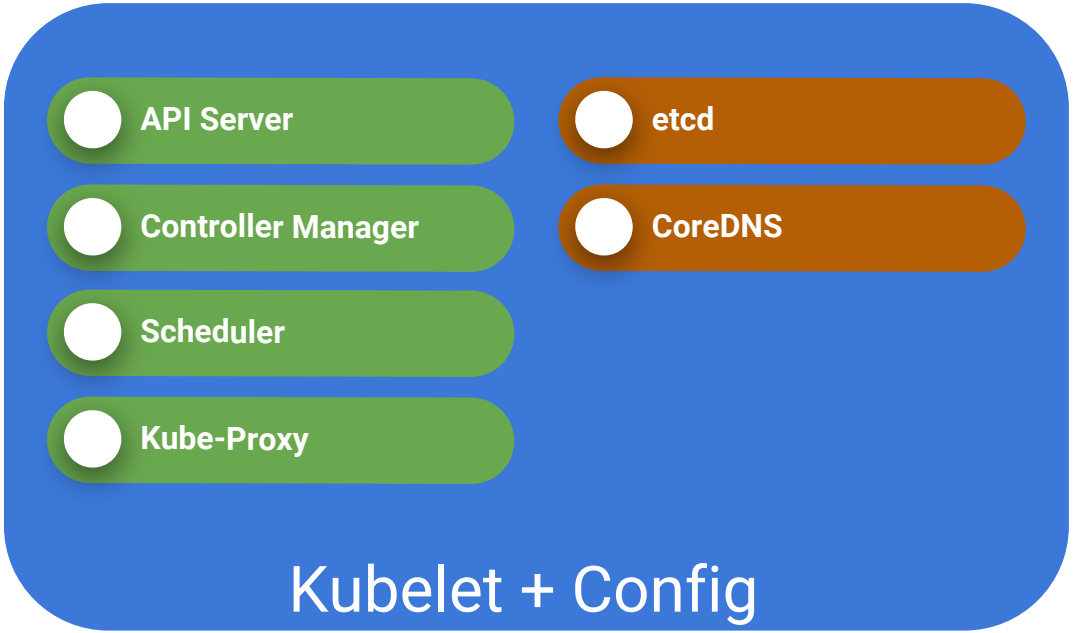
Kubeadm init

```
rosti@ubuntu:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                READY   STATUS              RESTARTS   AGE
kube-system   coredns-576cbf47c7-glxj9            0/1     ContainerCreating   0          72s
kube-system   coredns-576cbf47c7-zxp7r            0/1     ContainerCreating   0          72s
kube-system   etcd-ubuntu                         1/1     Running             0          9s
kube-system   kube-apiserver-ubuntu               1/1     Running             0          24s
kube-system   kube-controller-manager-ubuntu      1/1     Running             0          37s
kube-system   kube-proxy-b796d                    1/1     Running             0          72s
kube-system   kube-scheduler-ubuntu               1/1     Running             0          16s
```

kubeadm init

Control Plane

- PODs
- Kubelet + kubeadm
- Container Runtime
- Operating System



kubernetes

kubeadm init

```
rosti@ubuntu:~$ kubectl apply -f https://raw.githubusercontent.com/co
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
```

kubernetes

kubeadm init

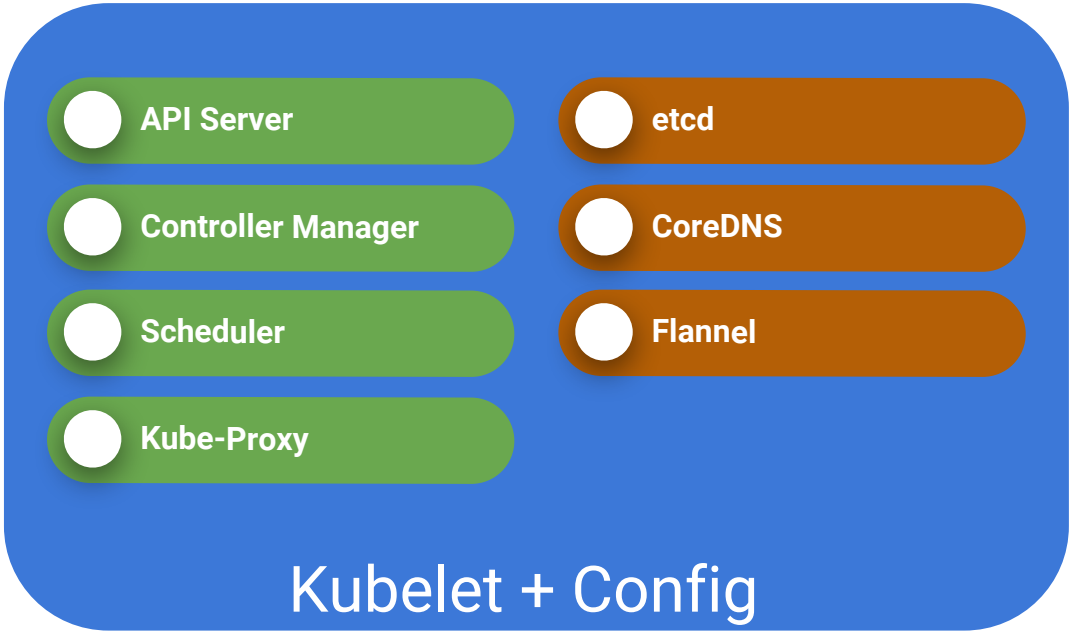
```
rosti@ubuntu:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
kube-system   coredns-576cbf47c7-glxj9            1/1     Running   0          4m20s
kube-system   coredns-576cbf47c7-zxp7r            1/1     Running   0          4m20s
kube-system   etcd-ubuntu                         1/1     Running   0          3m17s
kube-system   kube-apiserver-ubuntu               1/1     Running   0          3m32s
kube-system   kube-controller-manager-ubuntu      1/1     Running   0          3m45s
kube-system   kube-flannel-ds-amd64-c6kjt         1/1     Running   0          2m56s
kube-system   kube-proxy-b796d                    1/1     Running   0          4m20s
kube-system   kube-scheduler-ubuntu               1/1     Running   0          3m24s
```

kubernetes

kubeadm init

Control Plane

- PODs
- Kubelet + kubeadm
- Container Runtime
- Operating System



kubernetes

kubeadm join

```
rosti@worker1:~$ sudo kubeadm join 192.168.217.129:6443 --token g7qg8j.vj4blgjdbsagr9x0 --discovery-token-ca-cert-hash sha256:ddf6a76a90edbd47ddbeb3fea4a06180d1f0b26eea9f91db42f207baadc2b43
f
[preflight] running pre-flight checks
[discovery] Trying to connect to API Server "192.168.217.129:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.217.129:6443"
[discovery] Requesting info from "https://192.168.217.129:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "192.168.217.129:6443"
[discovery] Successfully established connection with API Server "192.168.217.129:6443"
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.12" ConfigMap in the kube-system namespace
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "worker1" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.

rosti@worker1:~$
```

kubernetes

kubeadm join

```
rosti@ubuntu:~$ kubectl get nodes
NAME       STATUS    ROLES     AGE    VERSION
ubuntu     Ready     master    16m    v1.12.1
worker1    Ready     <none>    72s    v1.12.1
```

```
rosti@ubuntu:~$ kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE   IP                NODE      NOMINATED NODE
kube-system   coredns-576cbf47c7-glxj9            1/1     Running   0          46m   10.244.0.9        ubuntu    <none>
kube-system   coredns-576cbf47c7-zxp7r            1/1     Running   0          46m   10.244.0.10       ubuntu    <none>
kube-system   etcd-ubuntu                         1/1     Running   0          33m   192.168.217.129   ubuntu    <none>
kube-system   kube-apiserver-ubuntu               1/1     Running   0          33m   192.168.217.129   ubuntu    <none>
kube-system   kube-controller-manager-ubuntu      1/1     Running   0          33m   192.168.217.129   ubuntu    <none>
kube-system   kube-flannel-ds-amd64-6zzsd         1/1     Running   0          31m   192.168.217.130   worker1   <none>
kube-system   kube-flannel-ds-amd64-c6kjt         1/1     Running   0          44m   192.168.217.129   ubuntu    <none>
kube-system   kube-proxy-4kmgk                    1/1     Running   0          31m   192.168.217.130   worker1   <none>
kube-system   kube-proxy-b796d                    1/1     Running   0          46m   192.168.217.129   ubuntu    <none>
kube-system   kube-scheduler-ubuntu               1/1     Running   0          33m   192.168.217.129   ubuntu    <none>
```

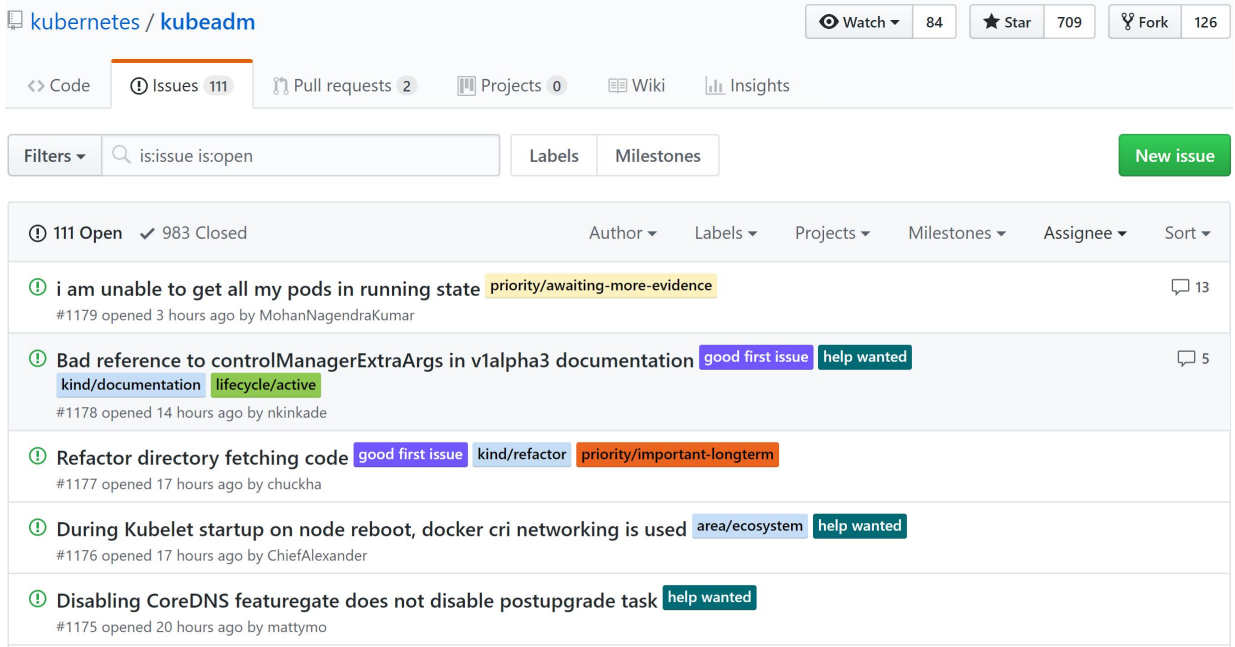
kubeadm

- The Future
  - Going GA in 1.13
  - Used by Cluster API
- Feedback
  - Have you considered kubeadm?
  - What is missing?
  - What is done wrong?
  - Can we do things better?

kubeadm

- Join the kubeadm office hours meetings
  - Wednesdays, 7PM BG
  - Meeting notes and agenda
- Ping us on Slack
  - `#sig-cluster-lifecycle`
  - `#kubeadm`

kubeadm



kubernetes

Cluster API

kubernetes

Overview

- Declarative, Kubernetes-style APIs to cluster creation, configuration, and management
- Extensible and environment-agnostic API
- Ease and standardize Node management and operation
  - Cluster autoscaler
  - Node Problem detector

https://sigs.k8s.io/cluster-api

kubernetes

API Resources 1/2

- Cluster
  - Describes a single cluster
  - Pod, Service and Domain
  - Provider-specific configuration

- Machine
  - single Node
  - Kubelet and Control plane configuration
  - Provider-specific configuration

kubernetes

API Resources 2/2

- MachineSet
  - Creates multiple Machine replicas
  - Same behavior as ReplicaSet

- MachineDeployment
  - Manages multiple ReplicaSet
  - Same behavior as Deployment
  - Rolling updates of Machines

kubernetes

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Cluster
metadata:
 name: test1-ckf8e
spec:
 clusterNetwork:
   services:
     cidrBlocks: ["10.96.0.0/12"]
   pods:
     cidrBlocks: ["192.168.0.0/16"]
   serviceDomain: cluster.local
 providerConfig:
   value:
     apiVersion: gceproviderconfig/v1alpha1
     kind: GCEClusterProviderConfig
     project: some-gcp-project
```

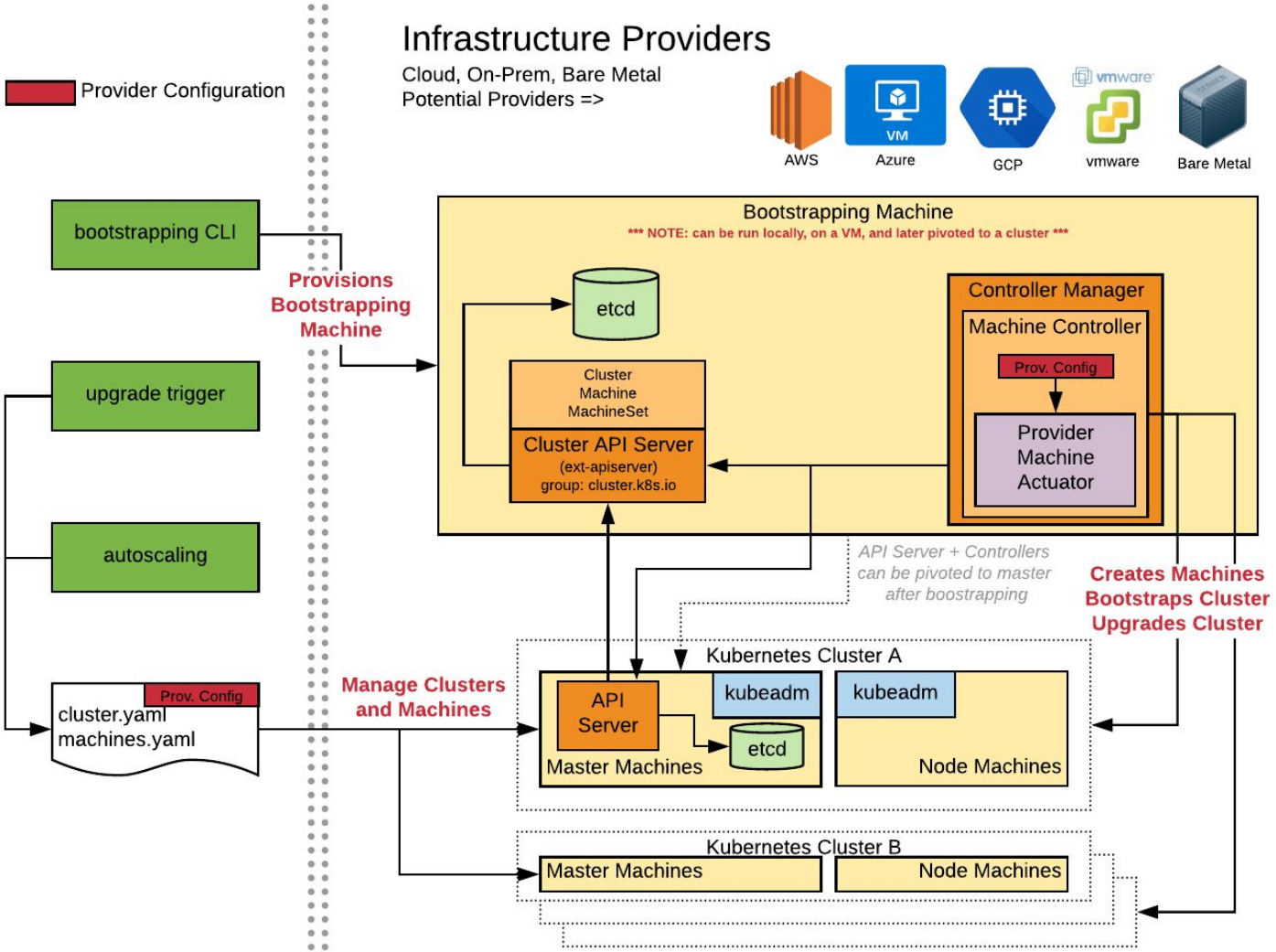
```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Cluster
metadata:
 name: test1-ckf8e
spec:
 clusterNetwork:
   services:
     cidrBlocks: ["10.96.0.0/12"]
   pods:
     cidrBlocks: ["192.168.0.0/16"]
   serviceDomain: cluster.local
 providerConfig:
   value:
     apiVersion: gceproviderconfig/v1alpha1
     kind: GCEClusterProviderConfig
     project: some-gcp-project
```

Provider-specific config

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Machine
metadata:
 generateName: gce-master-
 labels:
   set: master
spec:
 providerConfig:
   value:
     apiVersion: gceproviderconfig/v1alpha1
     kind: GCEMachineProviderConfig
     roles:
     - Master
     zone: europe-west2-a
     machineType: n1-standard-2
     os: ubuntu-1604-lts
     disks:
     - initializeParams:
         diskSizeGb: 30
         diskType: pd-standard
 versions:
   kubelet: 1.12.0
   controlPlane: 1.12.0
```

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Machine
metadata:
 generateName: gce-master-
 labels:
   set: master
spec:
 providerConfig:
   value:
     apiVersion: gceproviderconfig/v1alpha1
     kind: GCEMachineProviderConfig
     roles:
     - Master
     zone: europe-west2-a
     machineType: n1-standard-2
     os: ubuntu-1604-lts
     disks:
     - initializeParams:
         diskSizeGb: 30
         diskType: pd-standard
 versions:
   kubelet: 1.12.0
   controlPlane: 1.12.0
```

Provider-specific config


Infrastructure Providers
Cloud, On-Prem, Bare Metal
Potential Providers =>
AWS
Azure
GCP
vmware
Bare Metal
Provider Configuration
bootstrapping CLI
upgrade trigger
autoscaling
Prov. Config
cluster.yaml
machines.yaml
Provisions Bootstrapping Machine
Manage Clusters and Machines
Bootstrapping Machine
*** NOTE: can be run locally, on a VM, and later pivoted to a cluster ***
etcd
Cluster
Machine
MachineSet
Cluster API Server
(ext-apiserver)
group: cluster.k8s.io
Controller Manager
Machine Controller
Prov. Config
Provider
Machine
Actuator
API Server + Controllers can be pivoted to master after boostrapping
Creates Machines
Bootstraps Cluster
Upgrades Cluster
Kubernetes Cluster A
API Server
kubeadm
etcd
Master Machines
kubeadm
Node Machines
Kubernetes Cluster B
Master Machines
Node Machines

Thank you!

kubernetes