Kubernetes Clusters as a Service

Gardener

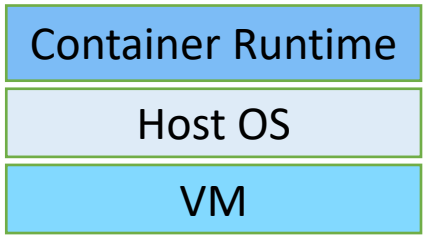
The Kubernetes Botanist

Matthias Sohr
Adel Zaalouk
SAP

From Containers to Kubernetes

From Containers to Kubernetes

Container



From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

From Containers to Kubernetes

Container

Container Runtime

Host OS

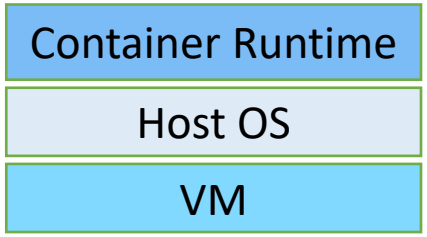
VM

Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

From Containers to Kubernetes

Container



Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

From Containers to Kubernetes

Container Scheduler

Container

Container Runtime

Host OS

VM

Benefits

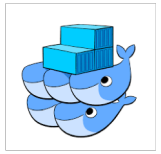
Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

From Containers to Kubernetes

Container Scheduler



Container

Container Runtime

Host OS

VM

Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

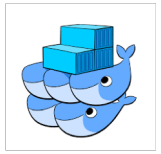
Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

Container Scheduler



From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

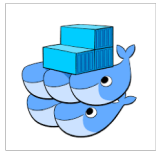
Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

Container Scheduler



From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

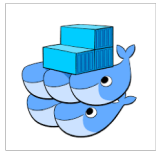
Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

Container Scheduler



Kubernetes

Orchestration of cluster of containers across multiple hosts

- Automatic placements, networking, deployments, scaling, roll-out/-back, A/B testing

From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

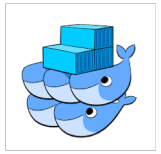
Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

Container Scheduler



Kubernetes

Orchestration of cluster of containers across multiple hosts

- Automatic placements, networking, deployments, scaling, roll-out/-back, A/B testing

Declarative – not procedural

- Declare target state, reconcile to desired state
- Self-healing

From Containers to Kubernetes

Container

Container Runtime

Host OS

VM

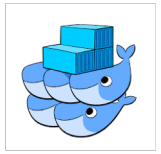
Benefits

Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing

Challenges

Networking
Deployments
Service Discovery
Auto Scaling
Persisting Data
Logging, Monitoring
Access Control

Container Scheduler



Kubernetes

Orchestration of cluster of containers across multiple hosts

- Automatic placements, networking, deployments, scaling, roll-out/-back, A/B testing

Declarative – not procedural

- Declare target state, reconcile to desired state
- Self-healing

Workload Portability

- Abstract from cloud provider specifics
- Multiple container runtimes

What does Kubernetes not cover ?

- **Install and manage many clusters**
- Across **Multi-Cloud**
  - Public Cloud Providers
  - Private Cloud

What does Kubernetes not cover ?

- **Install and manage many clusters**
- Across **Multi-Cloud**
  - Public Cloud Providers
  - Private Cloud
- **Zero Ops**
  - Minimal TCO
  - Manage Nodes
  - Manage Control Planes
  - Day 2 Operations

What does Kubernetes not cover ?

- **Install and manage many clusters**
- Across **Multi-Cloud**
  - Public Cloud Providers
  - Private Cloud
- **Zero Ops**
  - Minimal TCO
  - Manage Nodes
  - Manage Control Planes
  - Day 2 Operations

Gardener

WHAT do we want to achieve with the Gardener?

WHAT do we want to achieve with the Gardener?

Provide and establish solution for Kubernetes Clusters as a Service

WHAT do we want to achieve with the Gardener?

Provide and establish solution for Kubernetes Clusters as a Service

- ✓ Central Provisioning

WHAT do we want to achieve with the Gardener?

Provide and establish solution for Kubernetes Clusters as a Service

- ✓ Central Provisioning
- Engage with Open Source community, foster adoption, become CNCF project

WHAT do we want to achieve with the Gardener?

Provide and establish solution for Kubernetes Clusters as a Service

- ✓ Central Provisioning
- ◌ Engage with Open Source community, foster adoption, become CNCF project
- ✓ Large scale organisations need hundreds or thousands of clusters

WHAT do we want to achieve with the Gardener?

Homogenously on Hyper-Scale Providers and for the Private Cloud

WHAT do we want to achieve with the Gardener?

Homogenously on Hyper-Scale Providers and for the Private Cloud

- ✓ Full Control of Kubernetes,
  Homogeneous Across All Installations

WHAT do we want to achieve with the Gardener?

Homogenously on Hyper-Scale Providers and for the Private Cloud

- ✓ Full Control of Kubernetes,
  Homogeneous Across All Installations

- ✓ AWS, Azure, GCP, Alibaba and Others

WHAT do we want to achieve with the Gardener?

Homogenously on Hyper-Scale Providers and for the Private Cloud

- ✓ Full Control of Kubernetes,
  Homogeneous Across All Installations
- ✓ AWS, Azure, GCP, Alibaba and Others
- Private DCs for Data Privacy:
- ✓ OpenStack
- ◌ and eventually Bare Metal

WHAT do we want to achieve with the Gardener?

with Minimal TCO and Full Day-2 Operations Support

WHAT do we want to achieve with the Gardener?

with Minimal TCO and Full Day-2 Operations Support

- ✓ Full Automation, Backup & Recovery,
  High Resilience and Robustness, Self-Healing,
  Auto-Scaling, …

WHAT do we want to achieve with the Gardener?

with Minimal TCO and Full Day-2 Operations Support

- ✓ Full Automation, Backup & Recovery,
  High Resilience and Robustness, Self-Healing,
  Auto-Scaling, …
- ✓ Rollout Bug Fixes, Security Patches,
  Updates of Kubernetes, OS, Infrastructure,
  Certificate Management,
  …

Gardener Mission

Provide and establish solution for Kubernetes Clusters as a Service

Homogenously on Hyper-Scale Providers and for the Private Cloud

with Minimal TCO and Full Day-2 Operations Support

Primary Gardener Architecture Principle

Primary Gardener Architecture Principle

Following the definition of Kubernetes…

Primary Gardener Architecture Principle

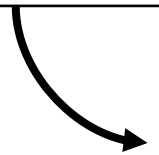
Following the definition of Kubernetes…

> Kubernetes is a system for automating **deployment, scaling, and management** of containerized software

Primary Gardener Architecture Principle

Following the definition of Kubernetes…

> Kubernetes is a system for automating **deployment, scaling, and management** of containerized software



…we do the following:

Primary Gardener Architecture Principle

Following the definition of Kubernetes…

Kubernetes is a system for automating **deployment, scaling, and management** of containerized software

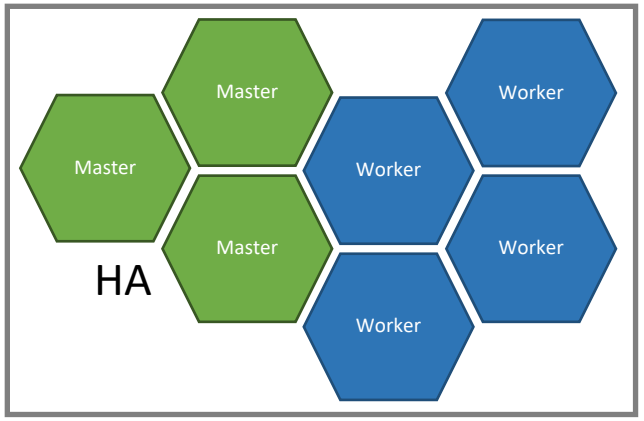
…we do the following:

We use **Kubernetes to deploy, host and operate Kubernetes**

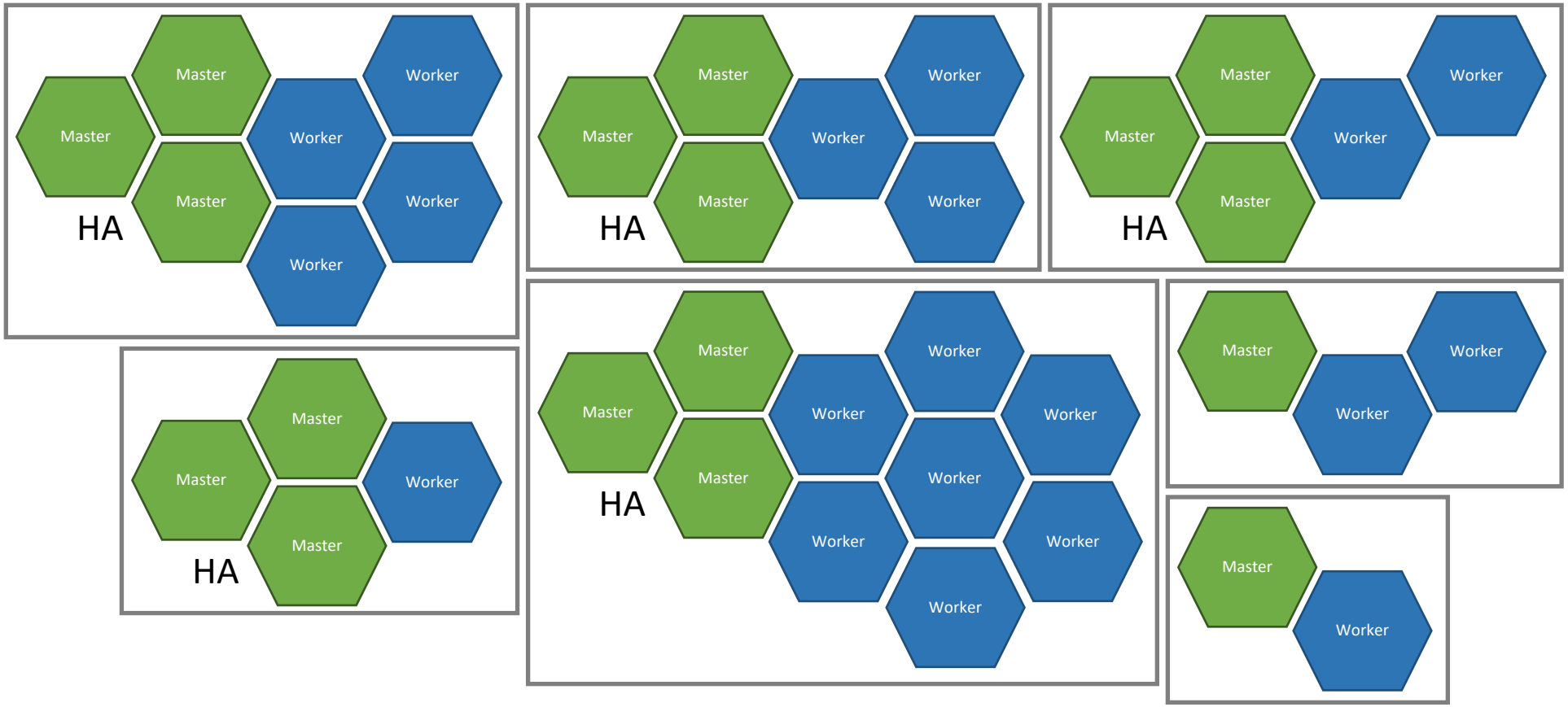
Control planes are **“seeded”** into already existing clusters

Common Kubernetes Cluster Setup

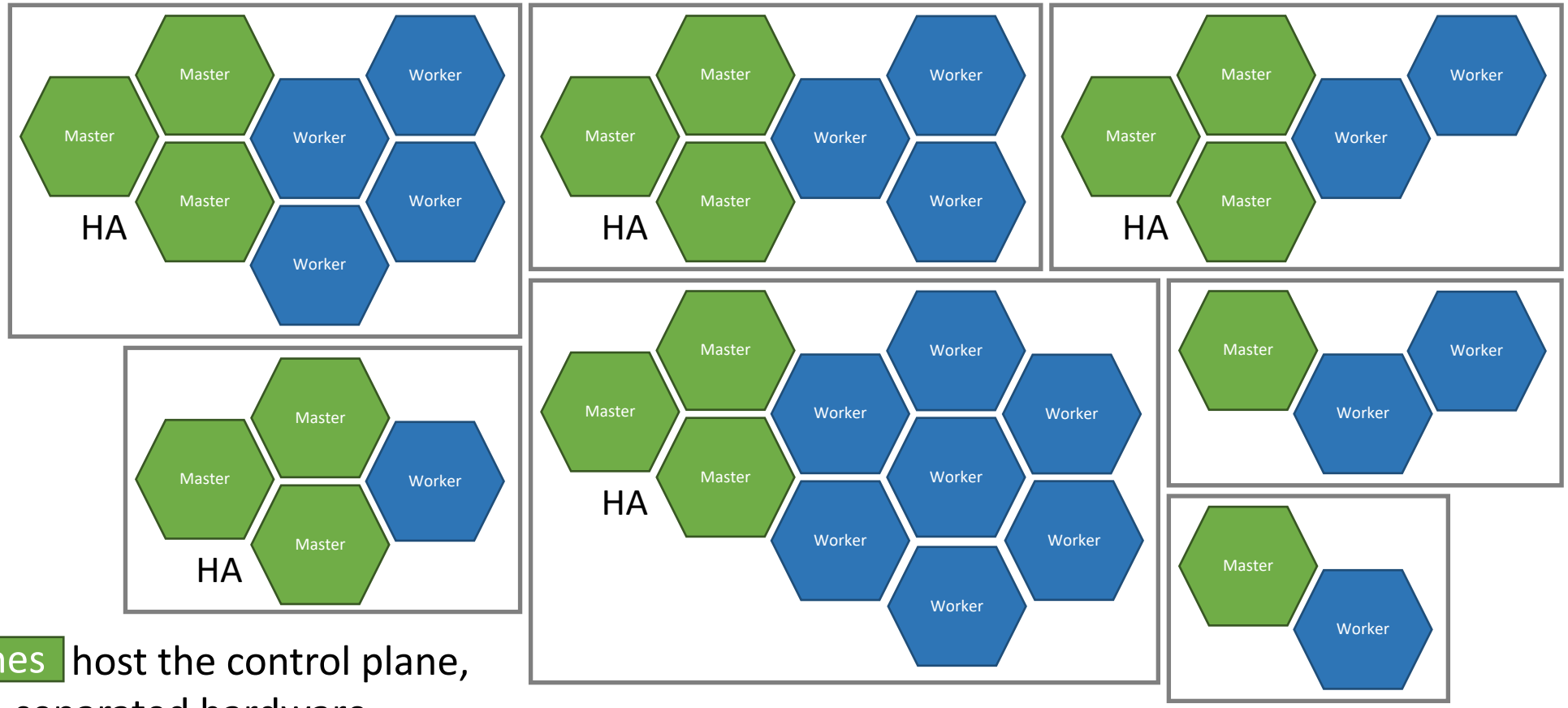
Common Kubernetes Cluster Setup



Common Kubernetes Cluster Setup



Common Kubernetes Cluster Setup

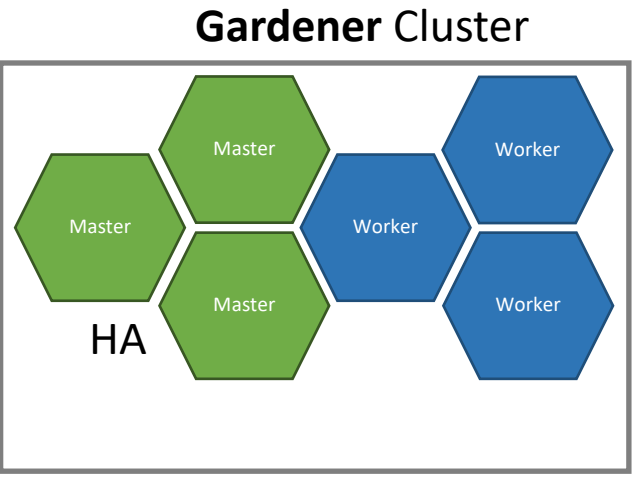


The green machines host the control plane, often in HA and on separated hardware (usually underutilized or, worse, overutilized)

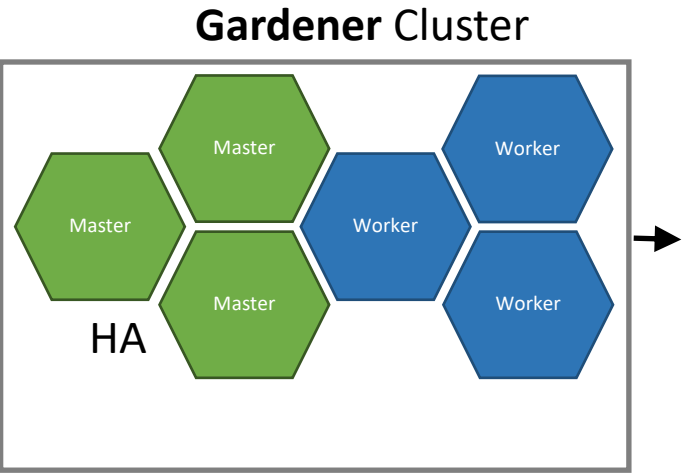
The blue machines host the actual workload and are managed by Kubernetes (usually pretty well utilized)

Gardener Kubernetes Cluster Setup

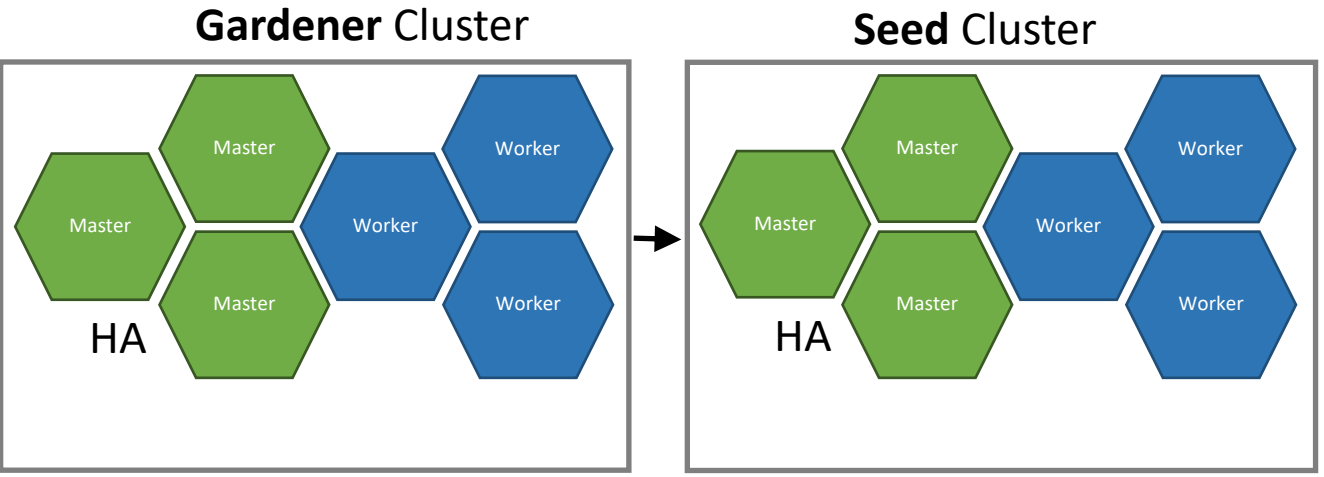
Gardener Kubernetes Cluster Setup



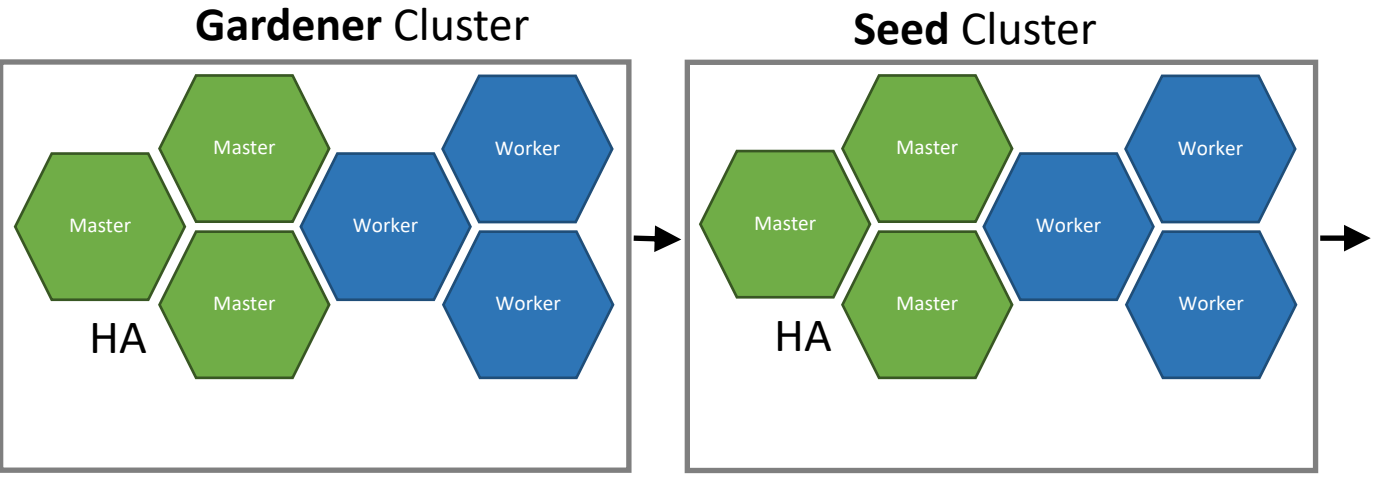
Gardener Kubernetes Cluster Setup



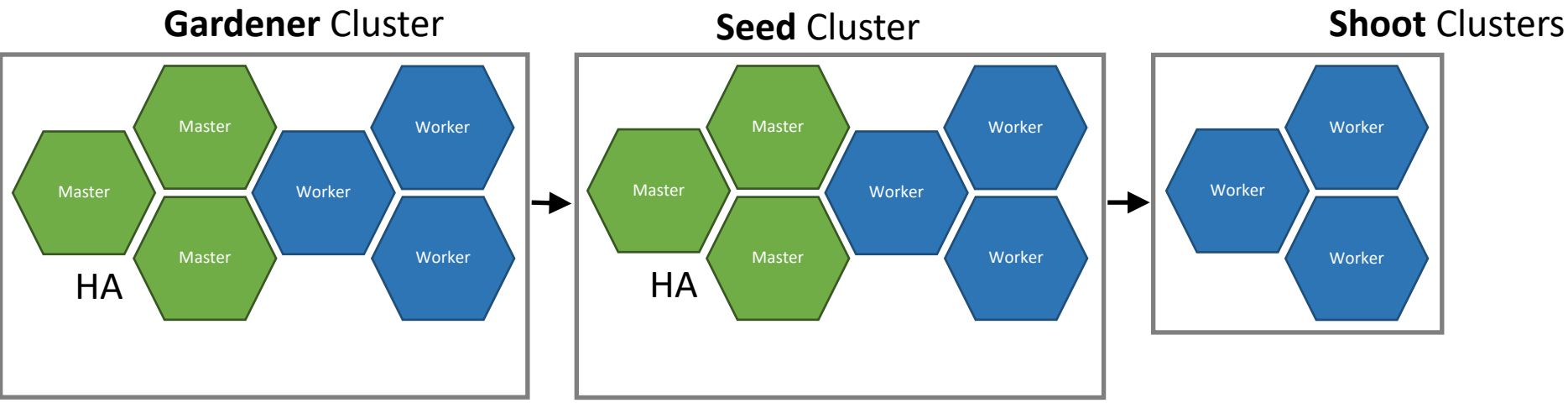
Gardener Kubernetes Cluster Setup



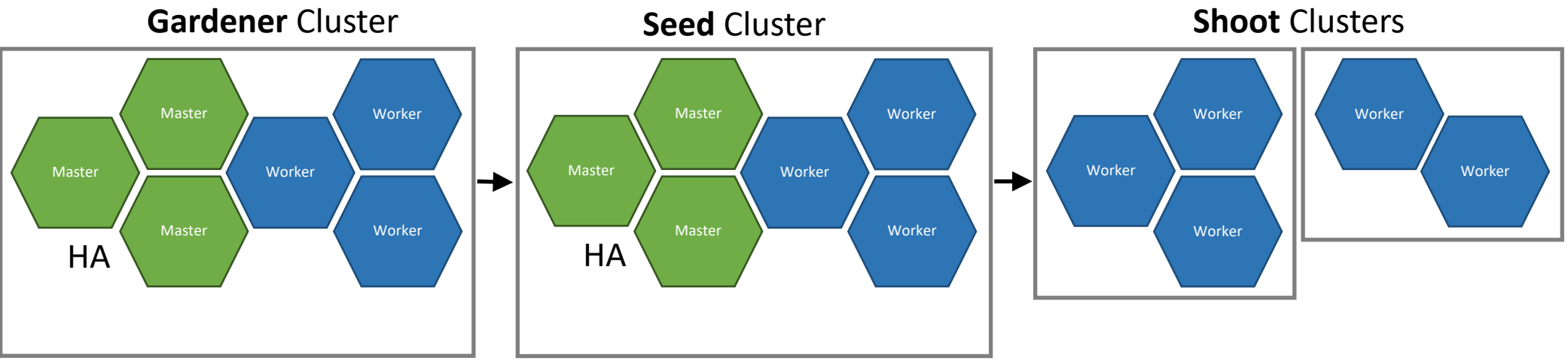
Gardener Kubernetes Cluster Setup



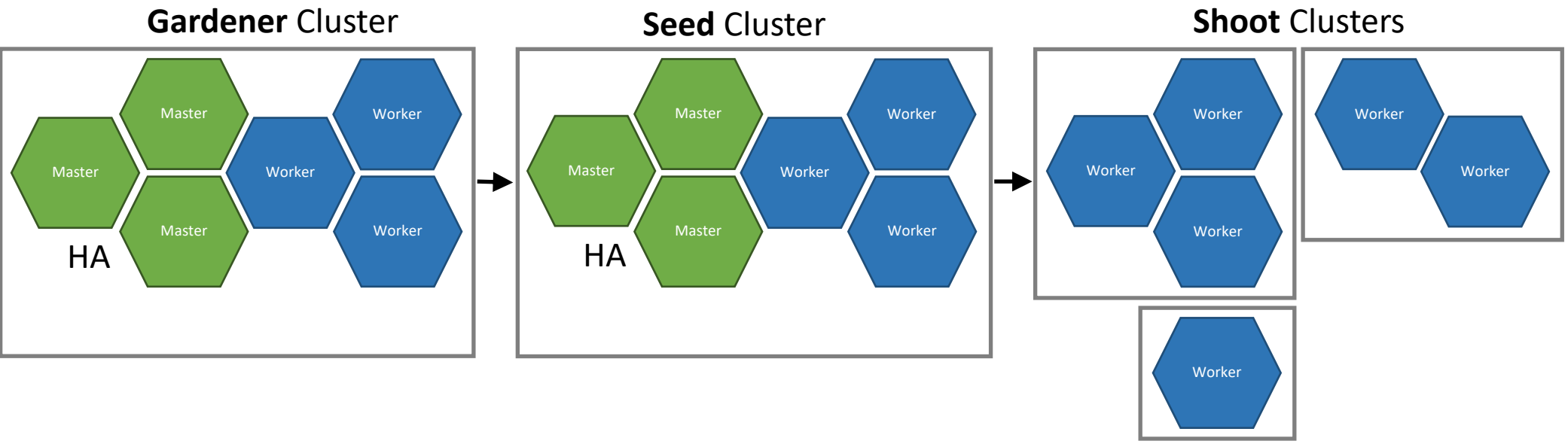
Gardener Kubernetes Cluster Setup



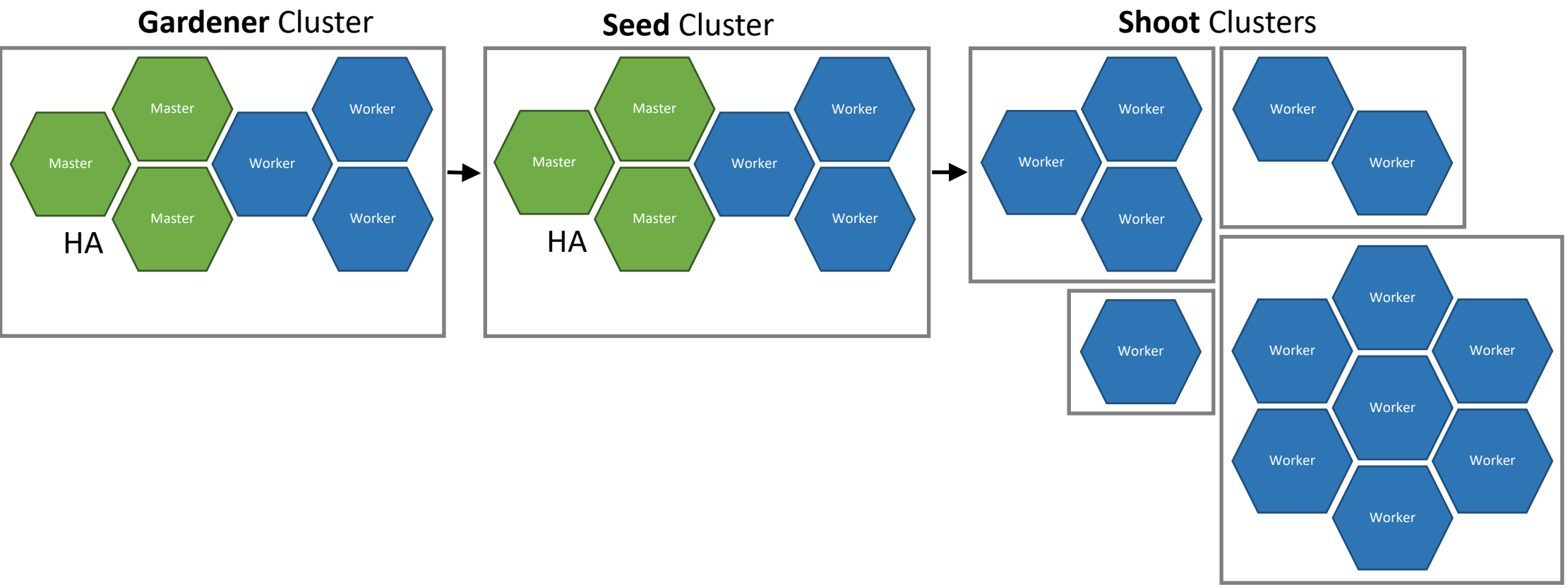
Gardener Kubernetes Cluster Setup



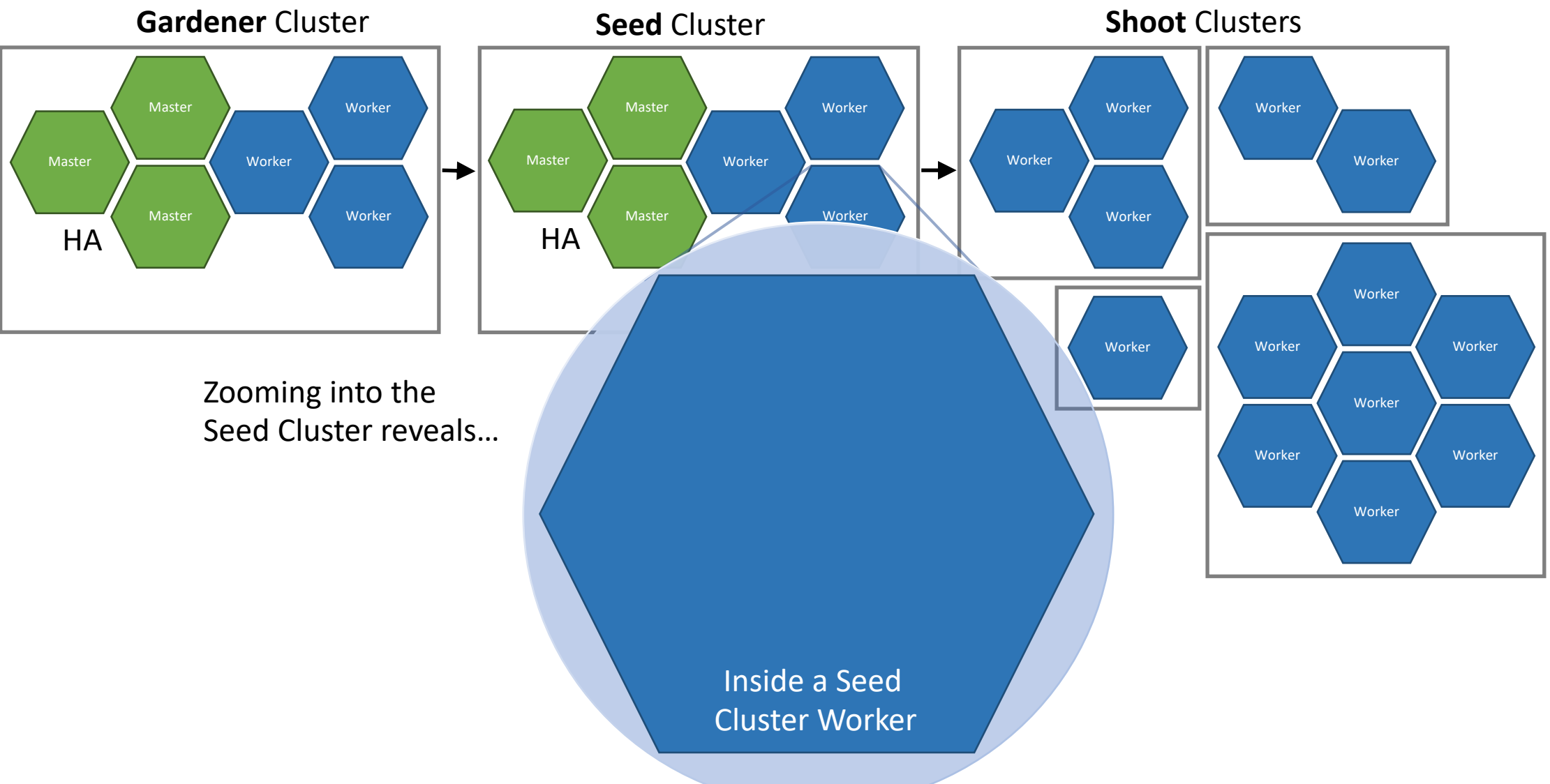
Gardener Kubernetes Cluster Setup



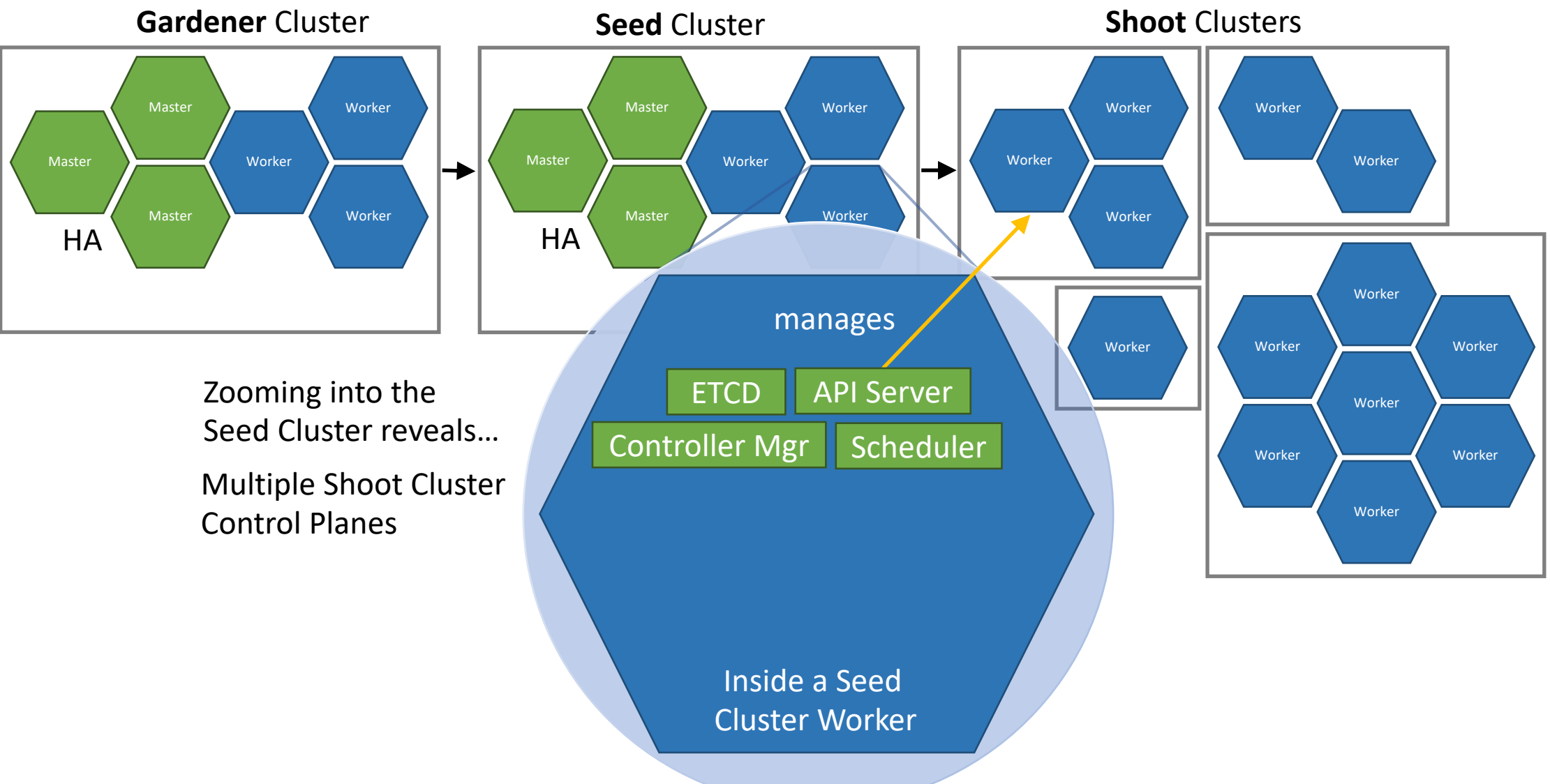
Gardener Kubernetes Cluster Setup



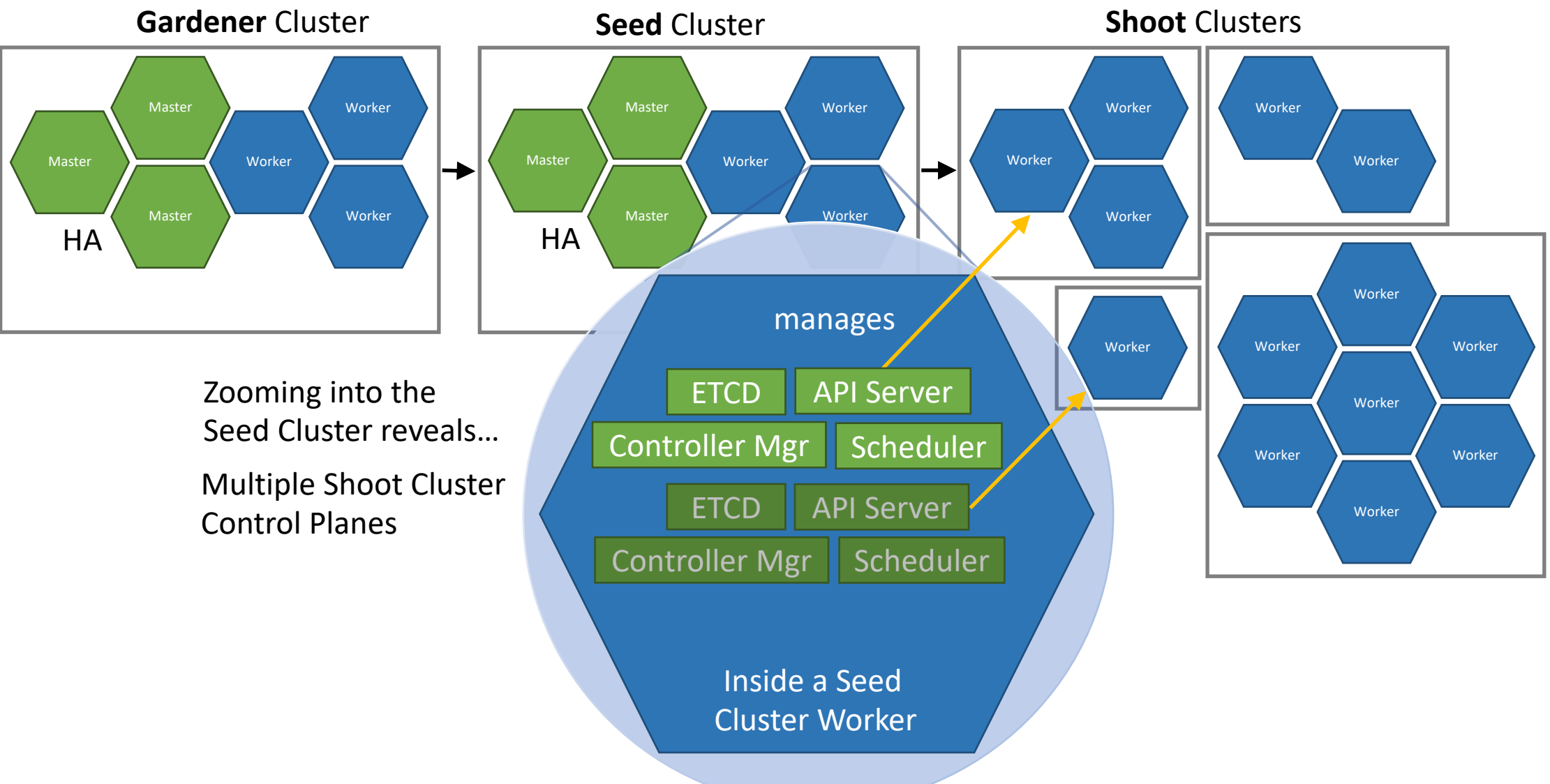
Gardener Kubernetes Cluster Setup



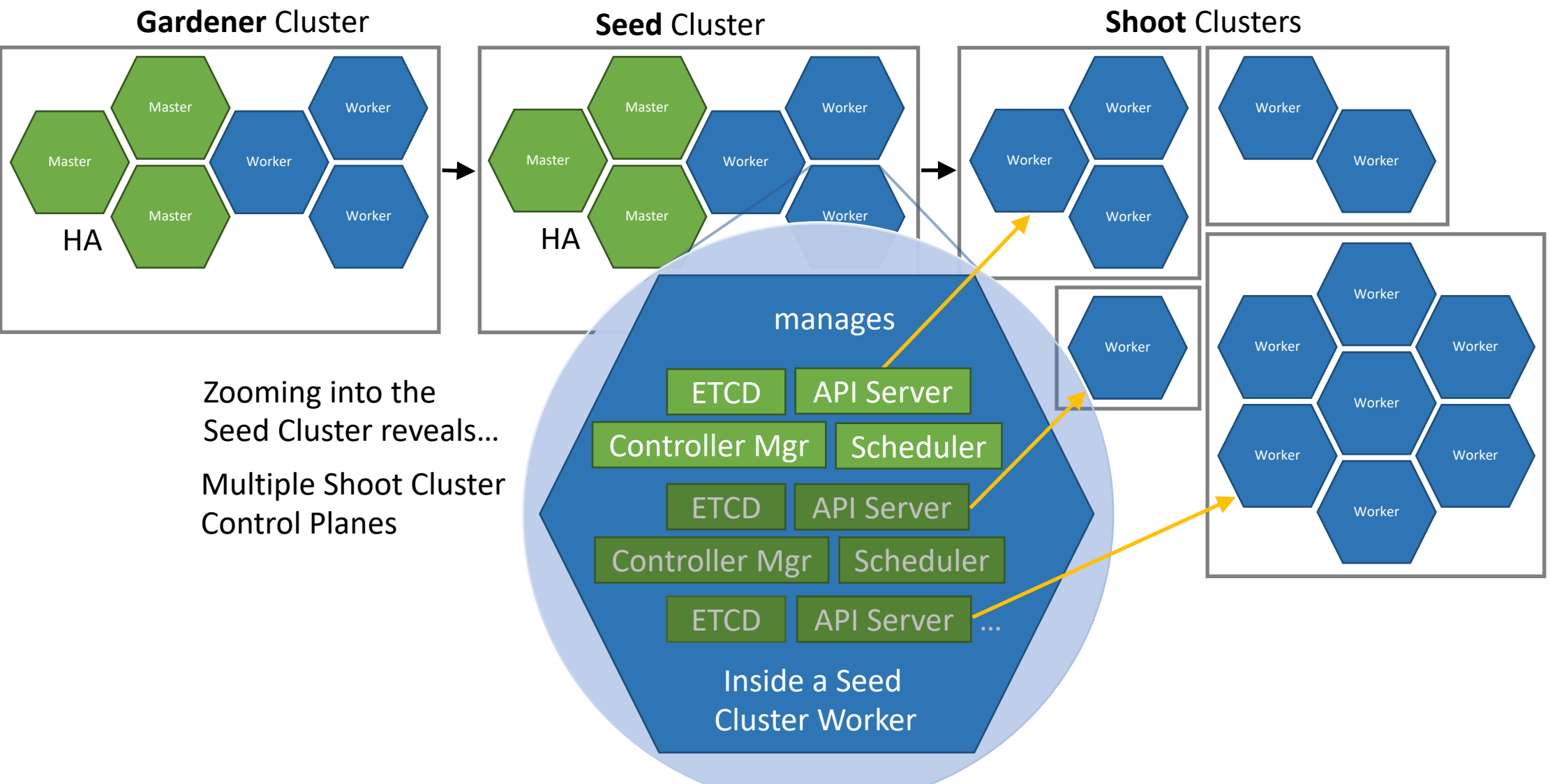
Gardener Kubernetes Cluster Setup



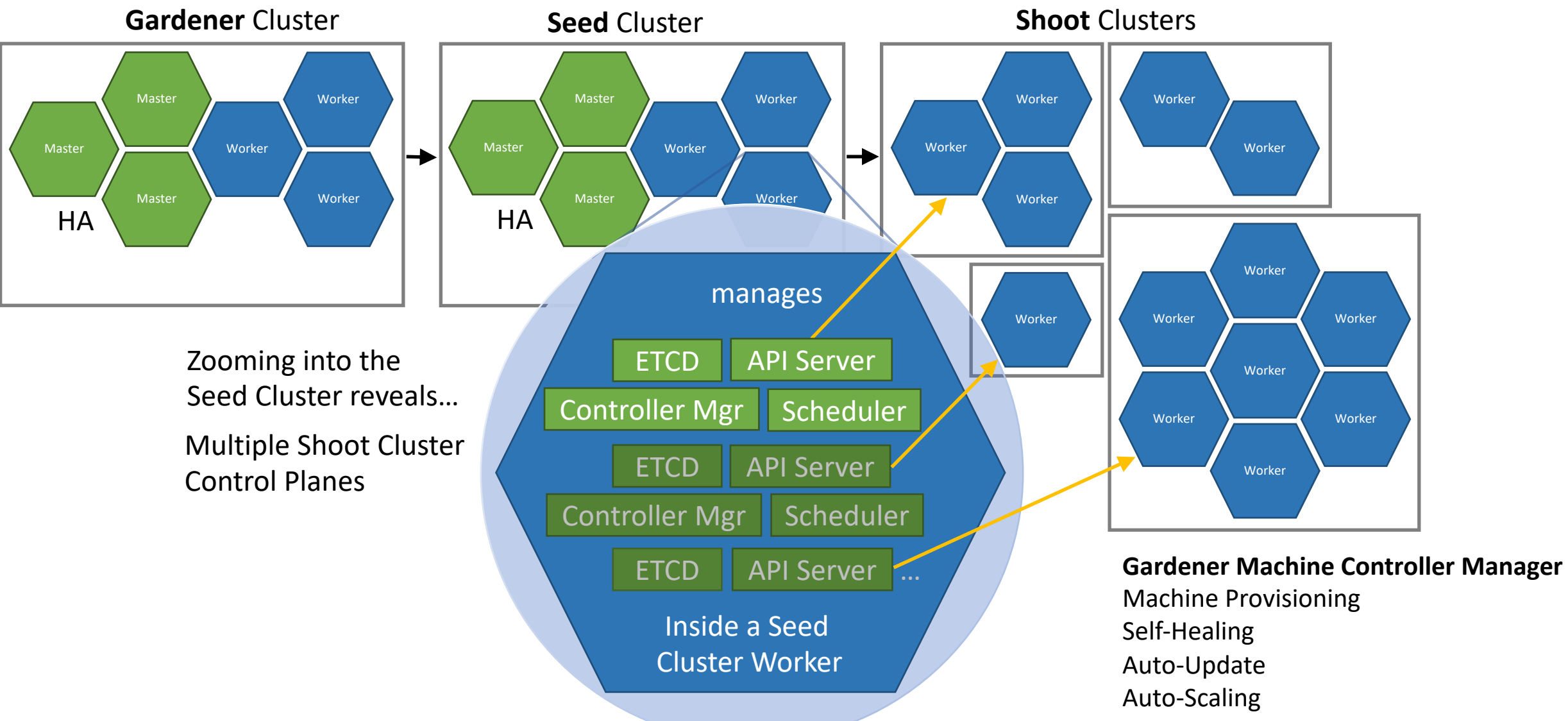
Gardener Kubernetes Cluster Setup



Gardener Kubernetes Cluster Setup



Gardener Kubernetes Cluster Setup



Zooming into the Seed Cluster reveals…

Multiple Shoot Cluster Control Planes

Gardener Machine Controller Manager

Machine Provisioning

Self-Healing

Auto-Update

Auto-Scaling

Primary Gardener Design Principle

Primary Gardener Design Principle

Do not reinvent the wheel …

"Let **Kubernetes** drive the design of the **Gardener**."

Lingua Franca – Gardener Cluster Resource

Lingua Franca – Gardener Cluster Resource

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

```
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

Set Kubernetes Version →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

Set Kubernetes Version →

Tweak Kubernetes Control Plane →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

Set Kubernetes Version →

Tweak Kubernetes Control Plane →

Define When and What to Update →

```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

Set Kubernetes Version →

Tweak Kubernetes Control Plane →

Define When and What to Update →

Gardener Reported Status →

```
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

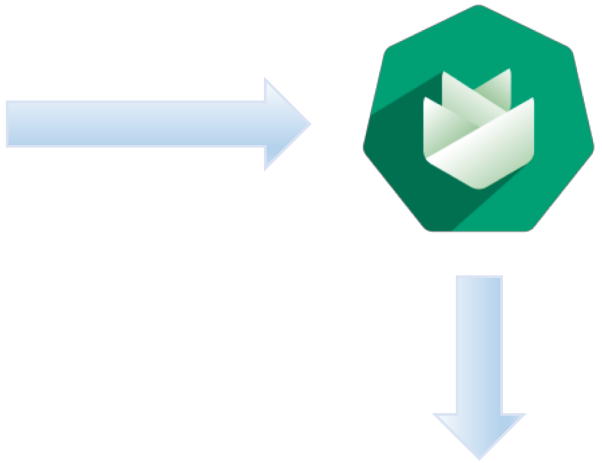
Set Kubernetes Version →

Tweak Kubernetes Control Plane →

Define When and What to Update →

Gardener Reported Status →

```
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```



Lingua Franca – Gardener Cluster Resource

Native Kubernetes Resource →

Gardener or Self-Managed DNS →

Define Your Infrastructure Needs →

Specify Worker Pools →

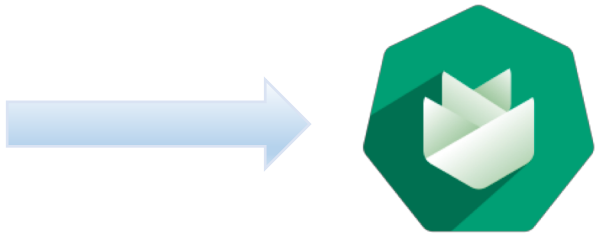
Set Kubernetes Version →

Tweak Kubernetes Control Plane →

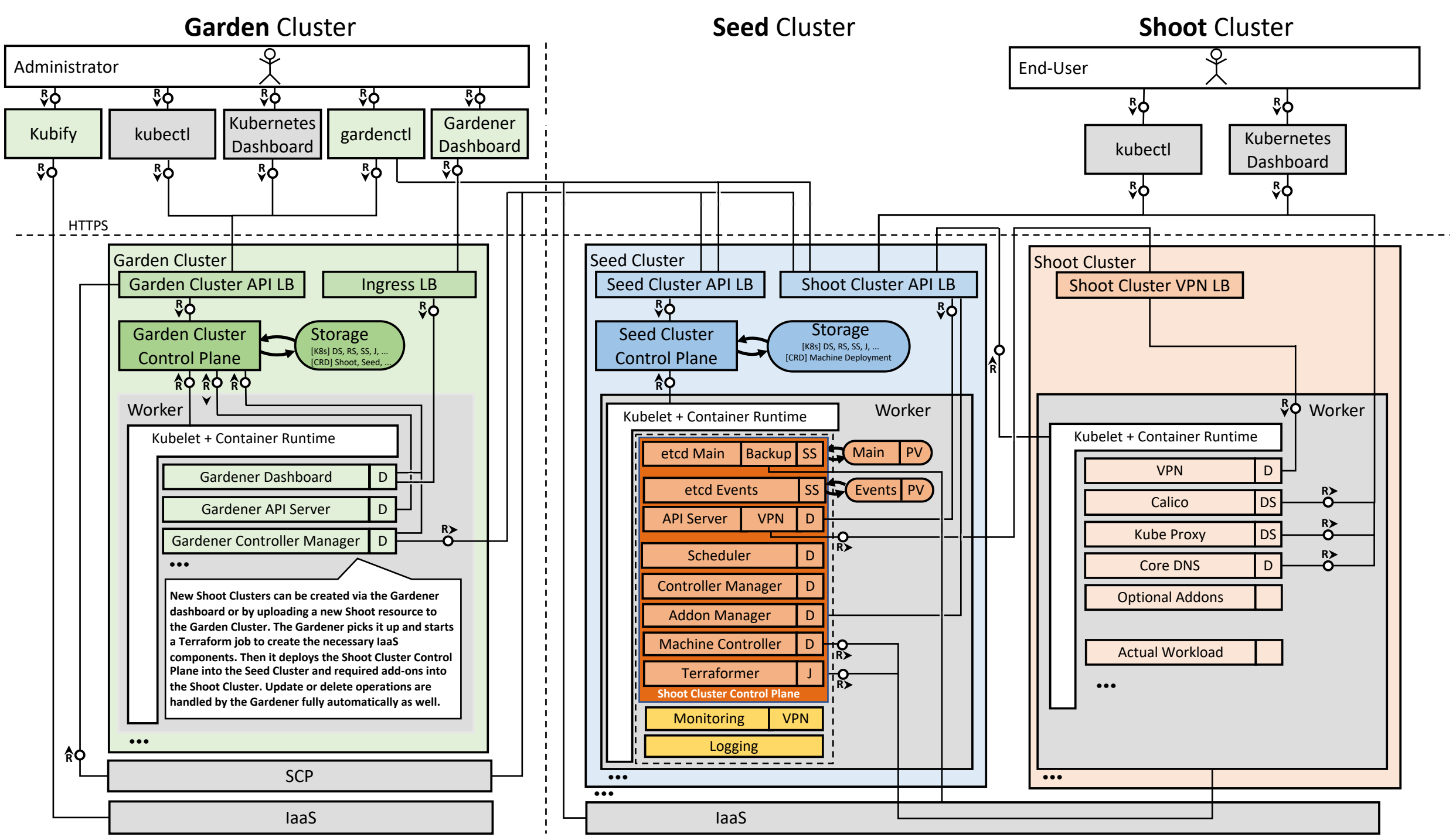
Define When and What to Update →

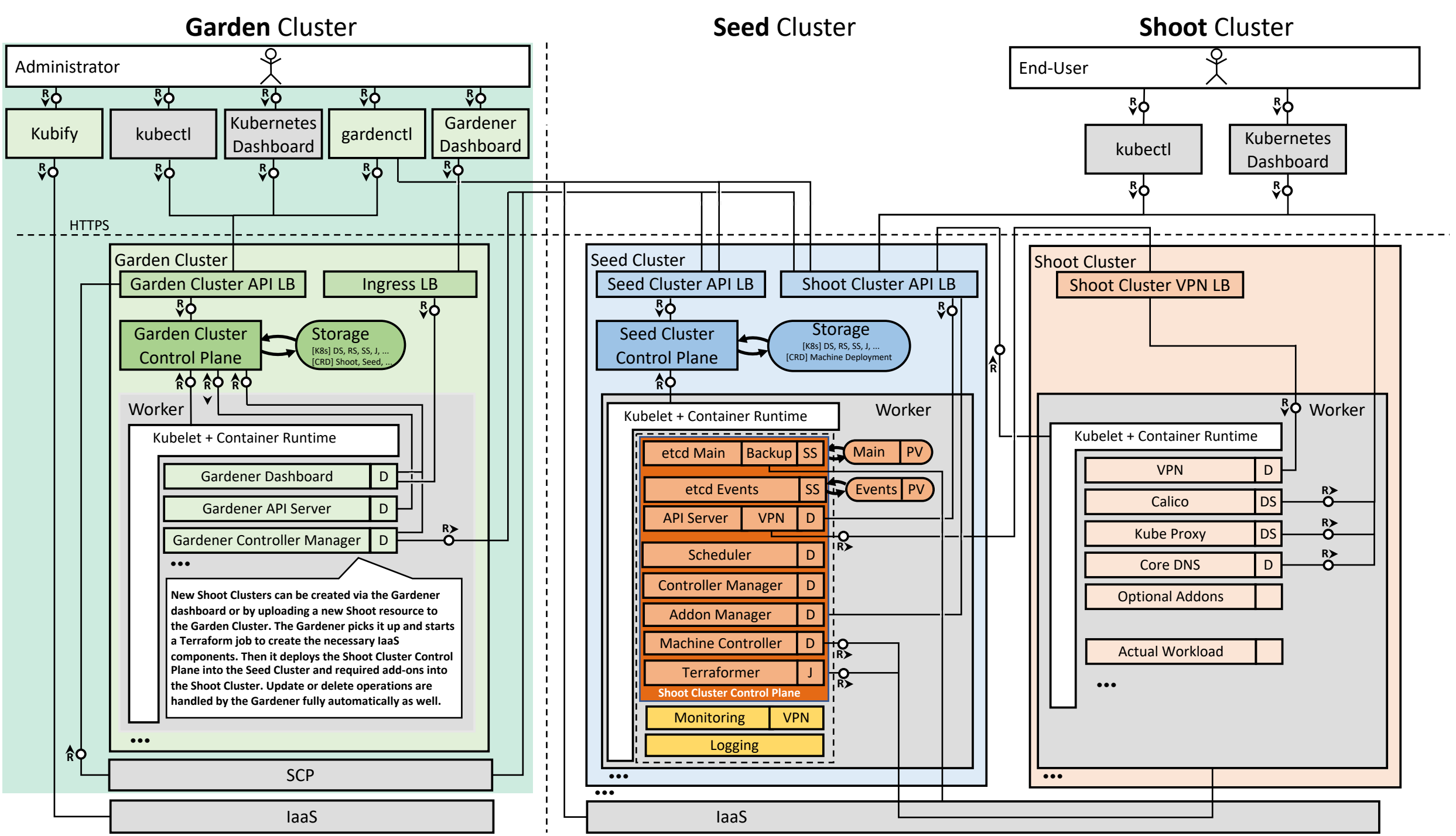
Gardener Reported Status →

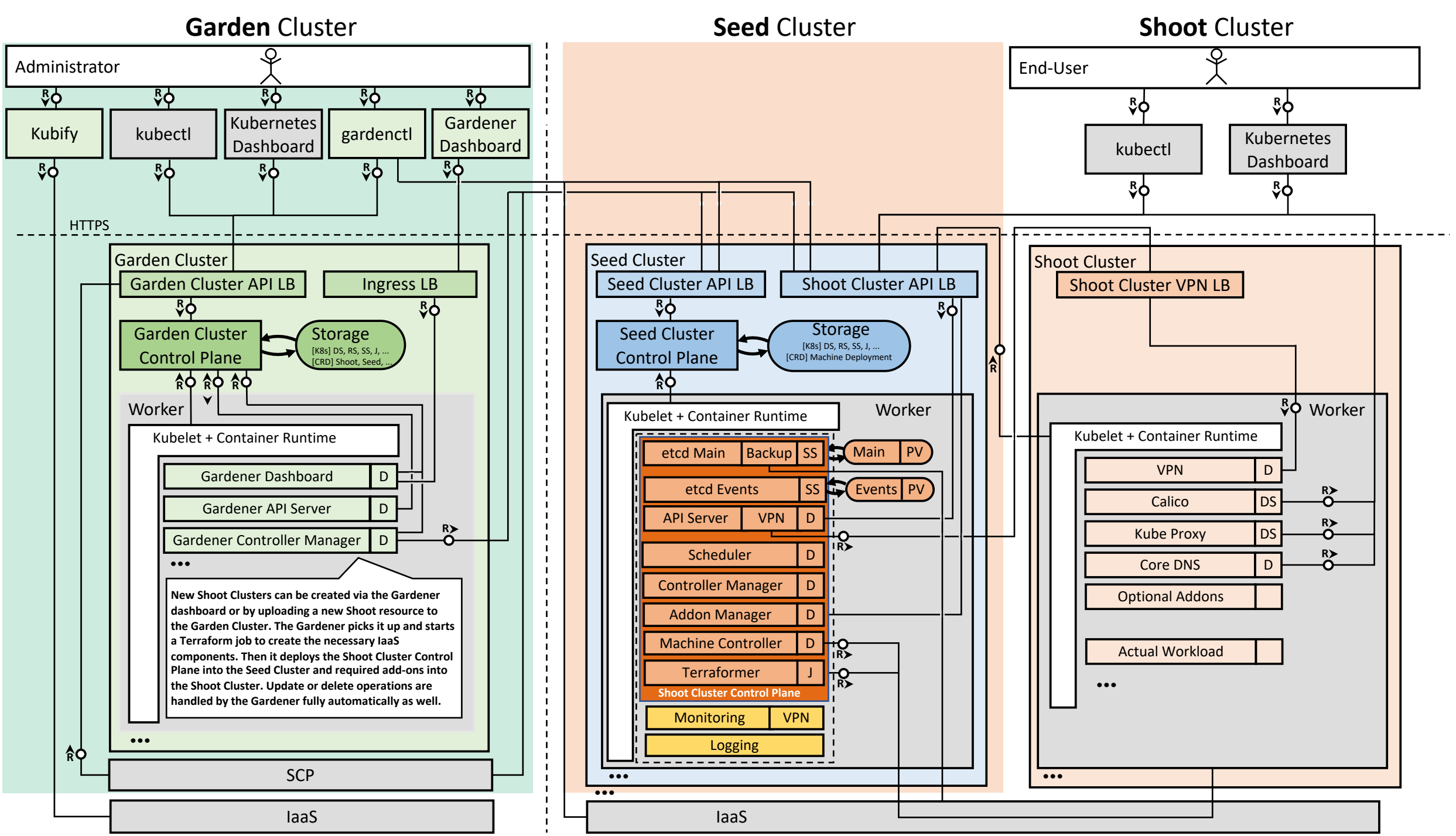
```yaml
apiVersion: garden.sapcloud.io/v1
kind: Shoot
metadata:
  name: my-cluster
  namespace: garden-project
spec:
  dns:
    provider: aws-route53
    domain: cluster.ondemand.com
  cloud:
    aws:
      networks:
        vpc:
          cidr: 10.250.0.0/16
      workers:
      - name: cpu-worker
        machineType: m4.xlarge
        autoScalerMin: 5
        autoScalerMax: 20
  kubernetes:
    version: 1.11.2
    kubeAPIServer:
      featureGates: ...
      runtimeConfig: ...
      admissionPlugins: ...
    kubeControllerManager:
      featureGates: ...
    kubeScheduler:
      featureGates: ...
    kubelet:
      featureGates: ...
  maintenance:
    timeWindow:
      begin: 220000+0000
      end: 230000+0000
    autoUpdate:
      kubernetesVersion: true
status:
  ...
```

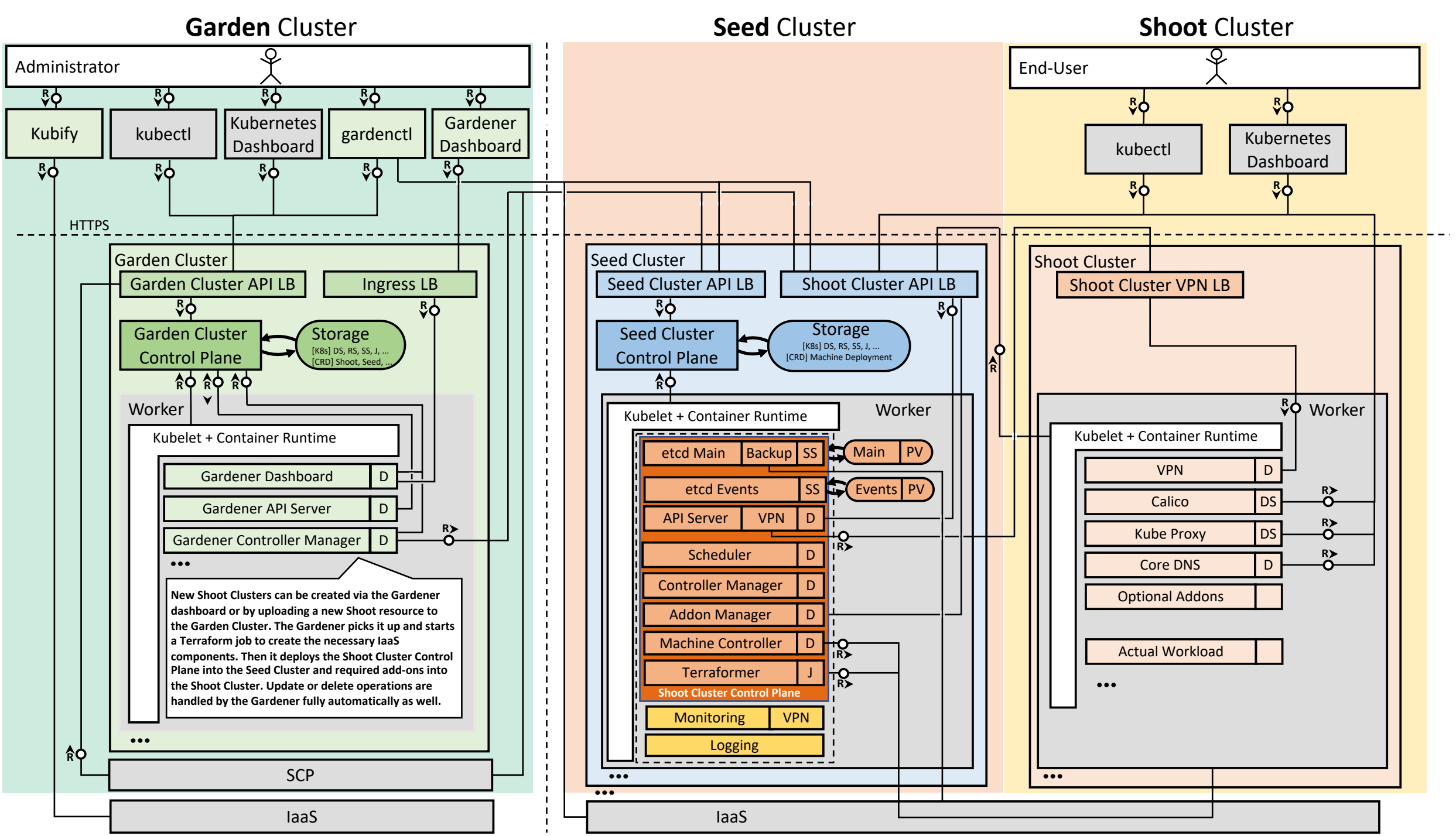


Avoid Vendor Lock-In


Garden Cluster
Seed Cluster
Shoot Cluster
Administrator
Kubify
kubectl
Kubernetes Dashboard
gardenctl
Gardener Dashboard
End-User
kubectl
Kubernetes Dashboard
HTTPS
Garden Cluster
Garden Cluster API LB
Ingress LB
Garden Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Shoot, Seed, ...
Worker
Kubelet + Container Runtime
Gardener Dashboard
D
Gardener API Server
D
Gardener Controller Manager
D
New Shoot Clusters can be created via the Gardener dashboard or by uploading a new Shoot resource to the Garden Cluster. The Gardener picks it up and starts a Terraform job to create the necessary IaaS components. Then it deploys the Shoot Cluster Control Plane into the Seed Cluster and required add-ons into the Shoot Cluster. Update or delete operations are handled by the Gardener fully automatically as well.
SCP
IaaS
Seed Cluster
Seed Cluster API LB
Shoot Cluster API LB
Seed Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Machine Deployment
Kubelet + Container Runtime
Worker
etcd Main
Backup
SS
Main
PV
etcd Events
SS
Events
PV
API Server
VPN
D
Scheduler
D
Controller Manager
D
Addon Manager
D
Machine Controller
D
Terraformer
J
Shoot Cluster Control Plane
Monitoring
VPN
Logging
IaaS
Shoot Cluster
Shoot Cluster VPN LB
Worker
Kubelet + Container Runtime
VPN
D
Calico
DS
Kube Proxy
DS
Core DNS
D
Optional Addons
Actual Workload


Garden Cluster
Seed Cluster
Shoot Cluster
Administrator
Kubify
kubectl
Kubernetes Dashboard
gardenctl
Gardener Dashboard
End-User
kubectl
Kubernetes Dashboard
HTTPS
Garden Cluster
Garden Cluster API LB
Ingress LB
Garden Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Shoot, Seed, ...
Worker
Kubelet + Container Runtime
Gardener Dashboard
D
Gardener API Server
D
Gardener Controller Manager
D
New Shoot Clusters can be created via the Gardener dashboard or by uploading a new Shoot resource to the Garden Cluster. The Gardener picks it up and starts a Terraform job to create the necessary IaaS components. Then it deploys the Shoot Cluster Control Plane into the Seed Cluster and required add-ons into the Shoot Cluster. Update or delete operations are handled by the Gardener fully automatically as well.
SCP
IaaS
Seed Cluster
Seed Cluster API LB
Shoot Cluster API LB
Seed Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Machine Deployment
Kubelet + Container Runtime
Worker
etcd Main
Backup
SS
Main
PV
etcd Events
SS
Events
PV
API Server
VPN
D
Scheduler
D
Controller Manager
D
Addon Manager
D
Machine Controller
D
Terraformer
J
Shoot Cluster Control Plane
Monitoring
VPN
Logging
IaaS
Shoot Cluster
Shoot Cluster VPN LB
Worker
Kubelet + Container Runtime
VPN
D
Calico
DS
Kube Proxy
DS
Core DNS
D
Optional Addons
Actual Workload


Garden Cluster
Seed Cluster
Shoot Cluster
Administrator
Kubify
kubectl
Kubernetes Dashboard
gardenctl
Gardener Dashboard
End-User
kubectl
Kubernetes Dashboard
HTTPS
Garden Cluster
Garden Cluster API LB
Ingress LB
Garden Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Shoot, Seed, ...
Worker
Kubelet + Container Runtime
Gardener Dashboard
D
Gardener API Server
D
Gardener Controller Manager
D
New Shoot Clusters can be created via the Gardener dashboard or by uploading a new Shoot resource to the Garden Cluster. The Gardener picks it up and starts a Terraform job to create the necessary IaaS components. Then it deploys the Shoot Cluster Control Plane into the Seed Cluster and required add-ons into the Shoot Cluster. Update or delete operations are handled by the Gardener fully automatically as well.
SCP
IaaS
Seed Cluster
Seed Cluster API LB
Shoot Cluster API LB
Seed Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Machine Deployment
Kubelet + Container Runtime
Worker
etcd Main
Backup
SS
Main
PV
etcd Events
SS
Events
PV
API Server
VPN
D
Scheduler
D
Controller Manager
D
Addon Manager
D
Machine Controller
D
Terraformer
J
Shoot Cluster Control Plane
Monitoring
VPN
Logging
IaaS
Shoot Cluster
Shoot Cluster VPN LB
Worker
Kubelet + Container Runtime
VPN
D
Calico
DS
Kube Proxy
DS
Core DNS
D
Optional Addons
Actual Workload


Garden Cluster
Seed Cluster
Shoot Cluster
Administrator
Kubify
kubectl
Kubernetes Dashboard
gardenctl
Gardener Dashboard
End-User
kubectl
Kubernetes Dashboard
HTTPS
Garden Cluster
Garden Cluster API LB
Ingress LB
Garden Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Shoot, Seed, ...
Worker
Kubelet + Container Runtime
Gardener Dashboard
D
Gardener API Server
D
Gardener Controller Manager
D
New Shoot Clusters can be created via the Gardener dashboard or by uploading a new Shoot resource to the Garden Cluster. The Gardener picks it up and starts a Terraform job to create the necessary IaaS components. Then it deploys the Shoot Cluster Control Plane into the Seed Cluster and required add-ons into the Shoot Cluster. Update or delete operations are handled by the Gardener fully automatically as well.
SCP
IaaS
Seed Cluster
Seed Cluster API LB
Shoot Cluster API LB
Seed Cluster Control Plane
Storage
[K8s] DS, RS, SS, J, ...
[CRD] Machine Deployment
Worker
Kubelet + Container Runtime
etcd Main
Backup
SS
Main
PV
etcd Events
SS
Events
PV
API Server
VPN
D
Scheduler
D
Controller Manager
D
Addon Manager
D
Machine Controller
D
Terraformer
J
Shoot Cluster Control Plane
Monitoring
VPN
Logging
IaaS
Shoot Cluster
Shoot Cluster VPN LB
Worker
Kubelet + Container Runtime
VPN
D
Calico
DS
Kube Proxy
DS
Core DNS
D
Optional Addons
Actual Workload

Following the Design Principle Gardener uses…

Kubernetes as deployment underlay

Following the Design Principle Gardener uses…

Kubernetes as deployment underlay

Following the Design Principle Gardener uses…

Kubernetes as deployment underlay

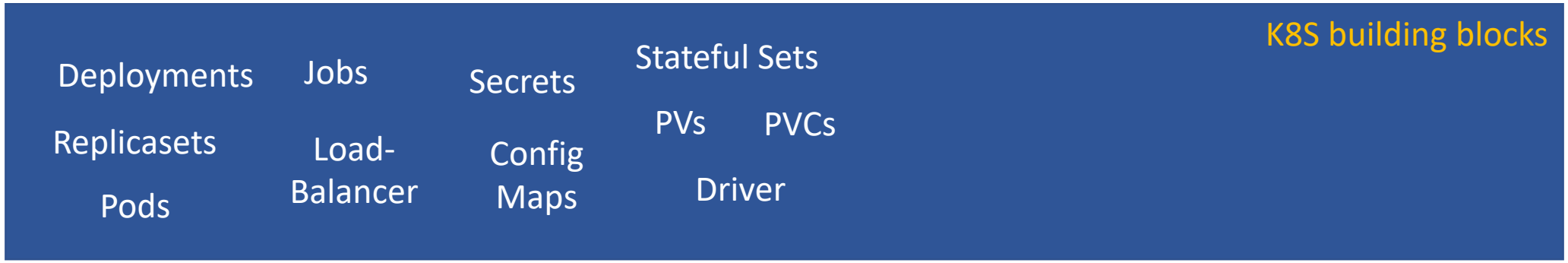
Following the Design Principle Gardener uses…

Kubernetes as deployment underlay

Following the Design Principle Gardener uses…

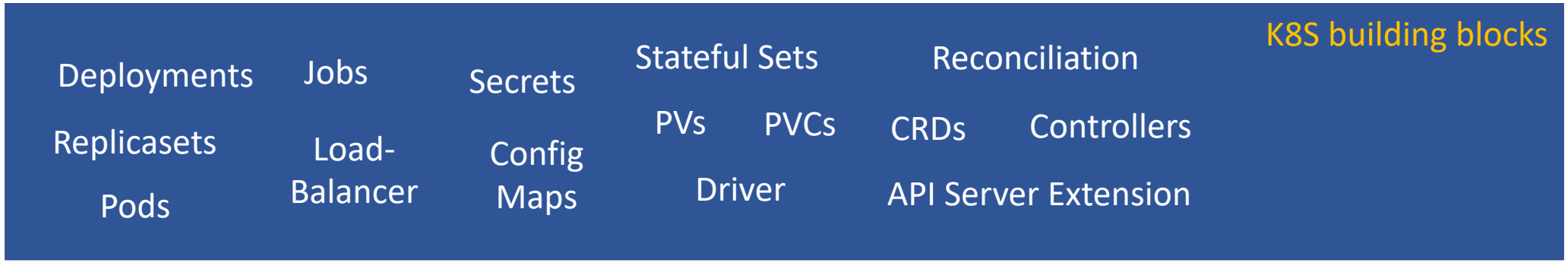
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



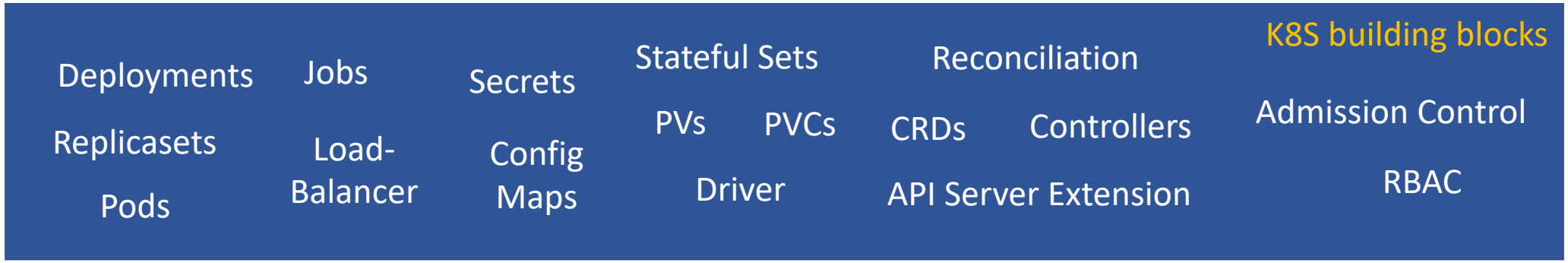
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



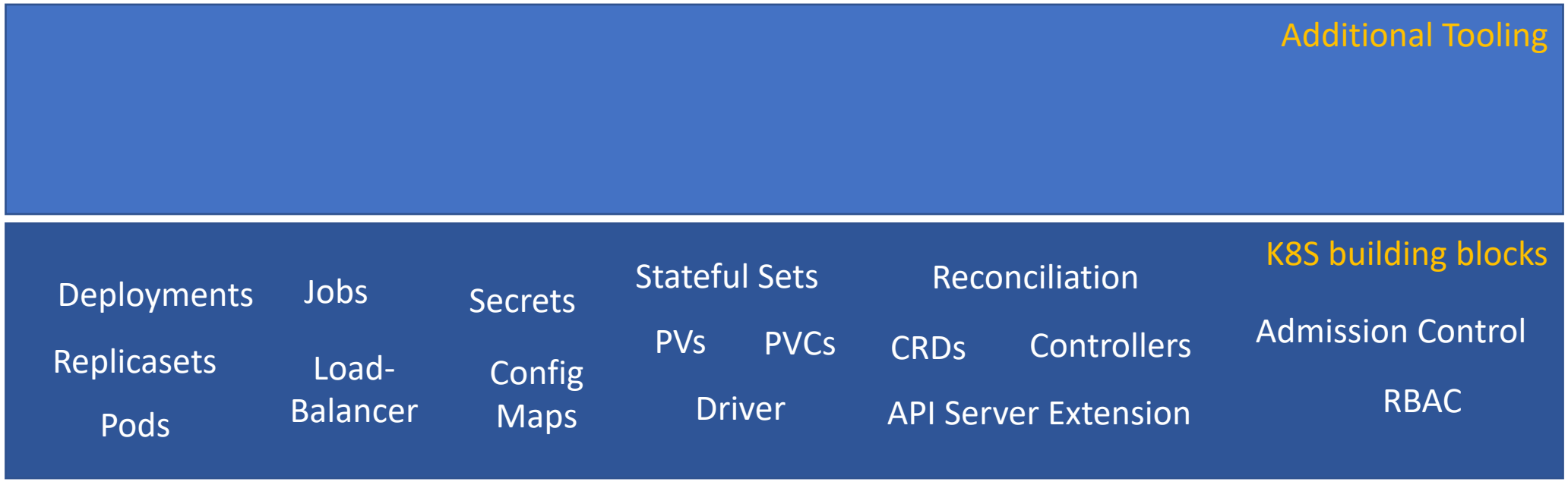
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



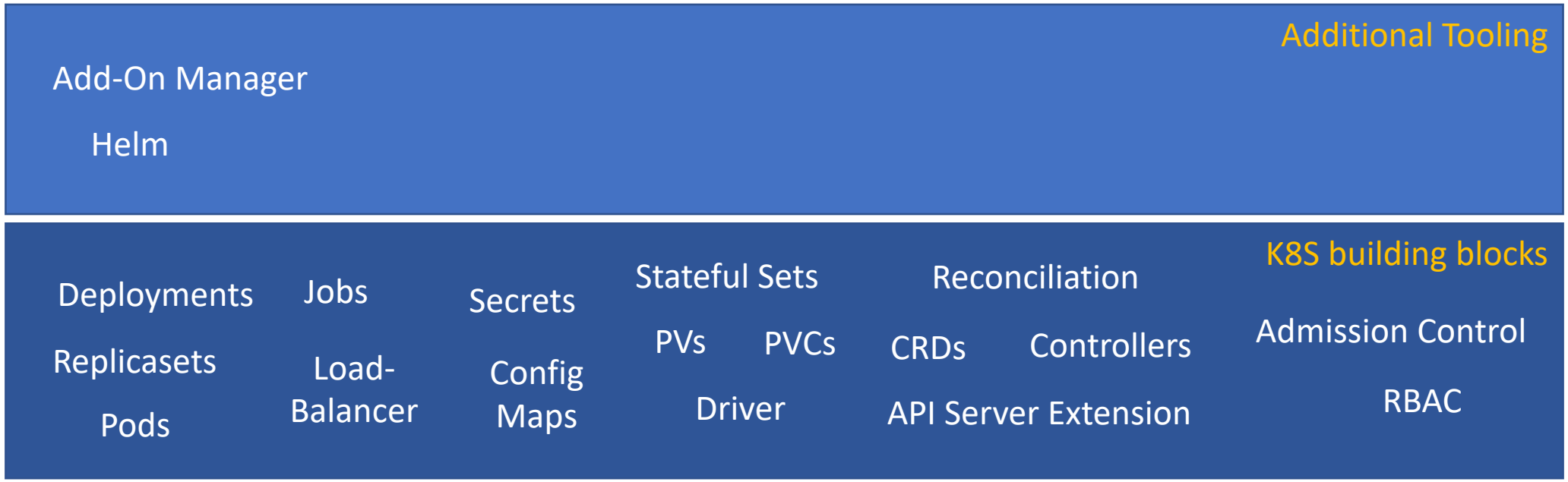
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



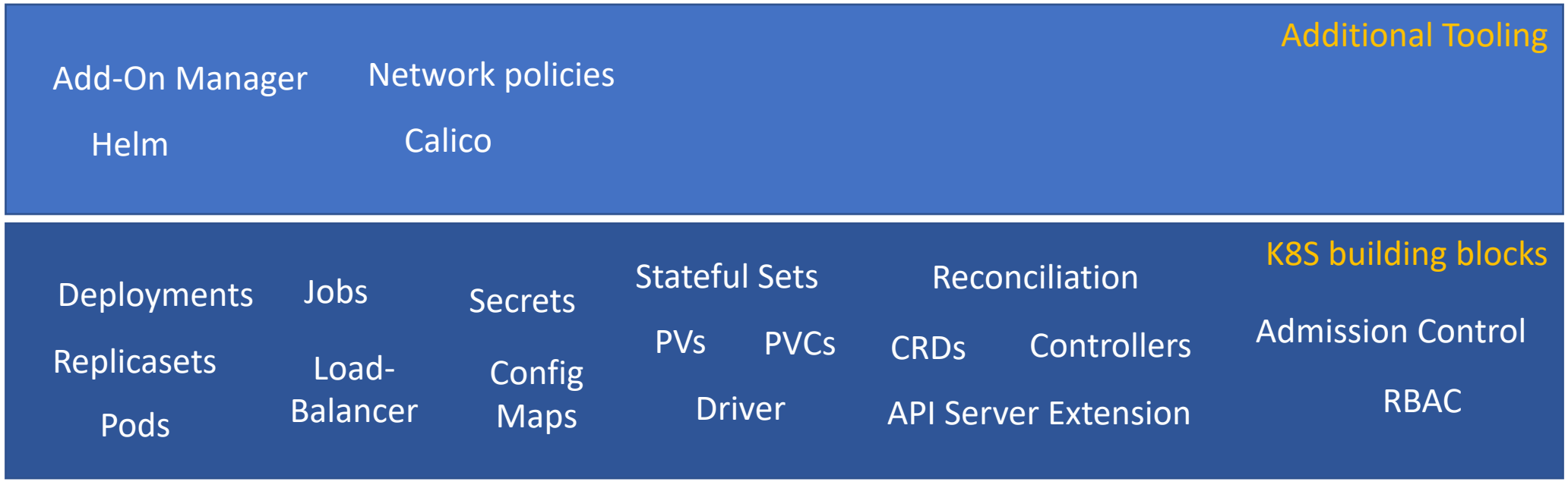
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



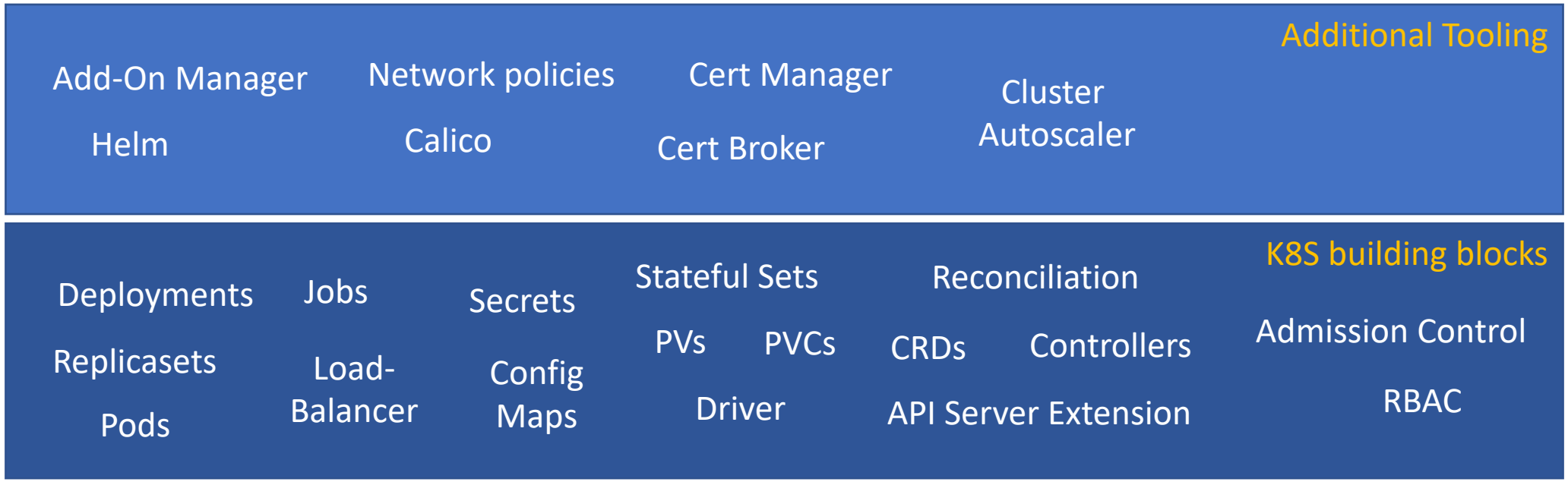
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



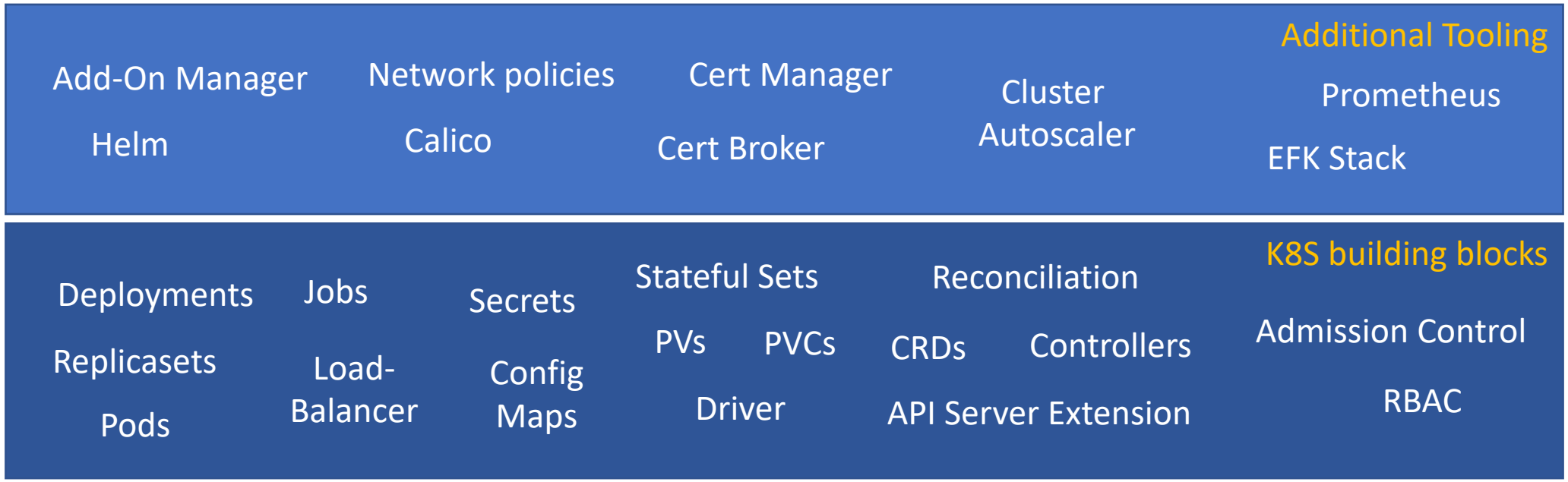
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



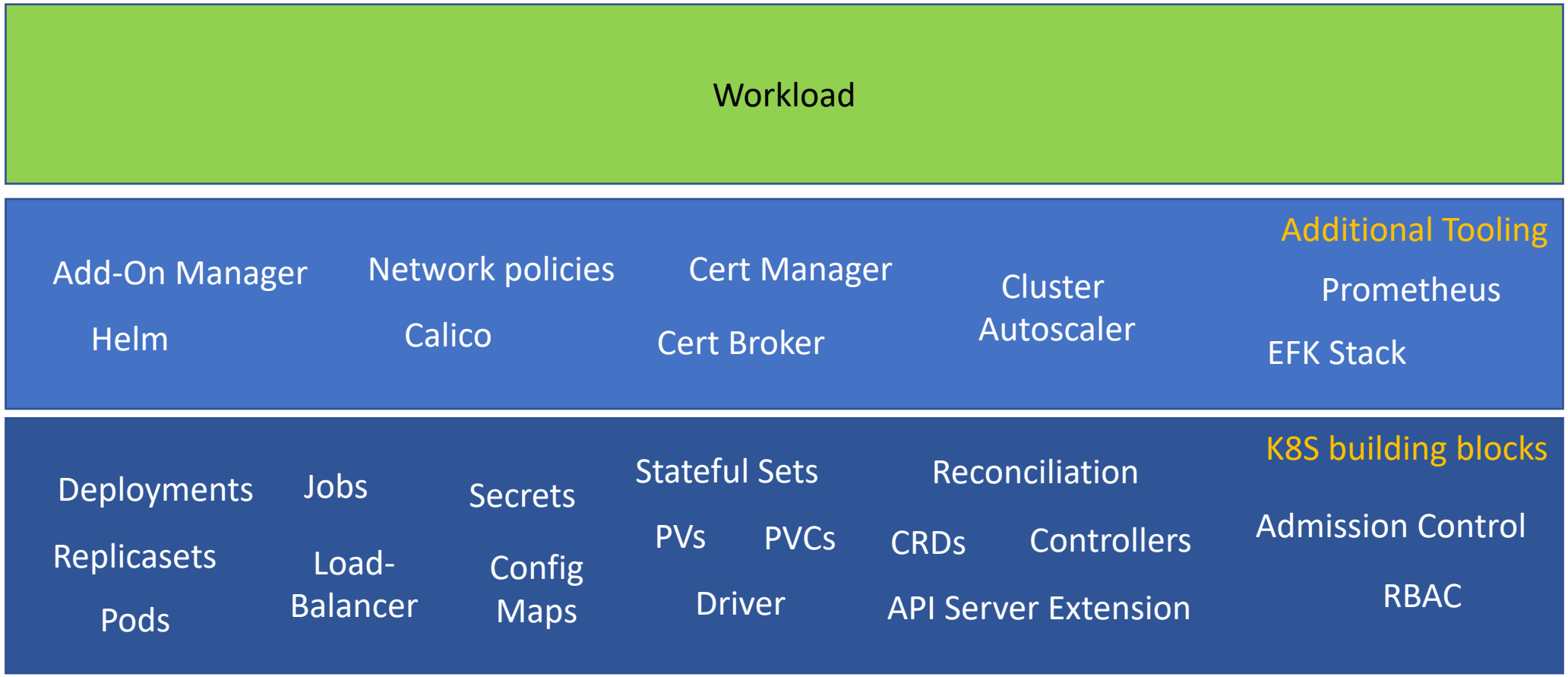
Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



Kubernetes as deployment underlay

Following the Design Principle Gardener uses…



Kubernetes as deployment underlay

Where are all these clusters coming from?

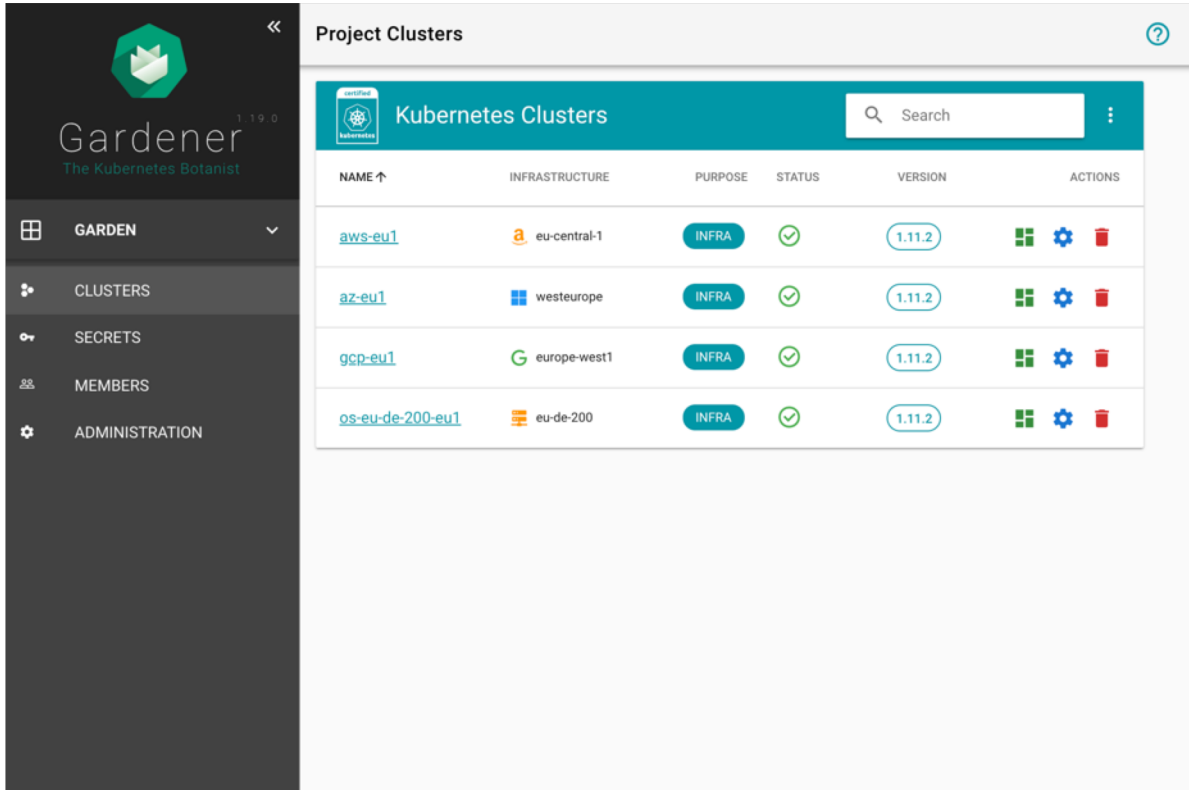
Garden clusters are installed on a bootstrap cluster

- in GKE, EKS, AKS
- set up using Gardener's **Kubify**
- DR setup with the **Gardener** Ring (planned)

Seed clusters are created as shoot clusters by the **Gardener**

Shoot clusters are created by their **seed cluster**
which is managed by the **Gardener**

Gardener Demo



Gardener 1.19.0

The Kubernetes Botanist

- GARDEN
- CLUSTERS
- SECRETS
- MEMBERS
- ADMINISTRATION

Project Clusters

Kubernetes Clusters

Search

NAME	INFRASTRUCTURE	PURPOSE	STATUS	VERSION	ACTIONS
aws-eu1	eu-central-1	INFRA	✓	1.11.2	
az-eu1	westeurope	INFRA	✓	1.11.2	
gcp-eu1	europe-west1	INFRA	✓	1.11.2	
os-eu-de-200-eu1	eu-de-200	INFRA	✓	1.11.2	

Gardener Community Installer

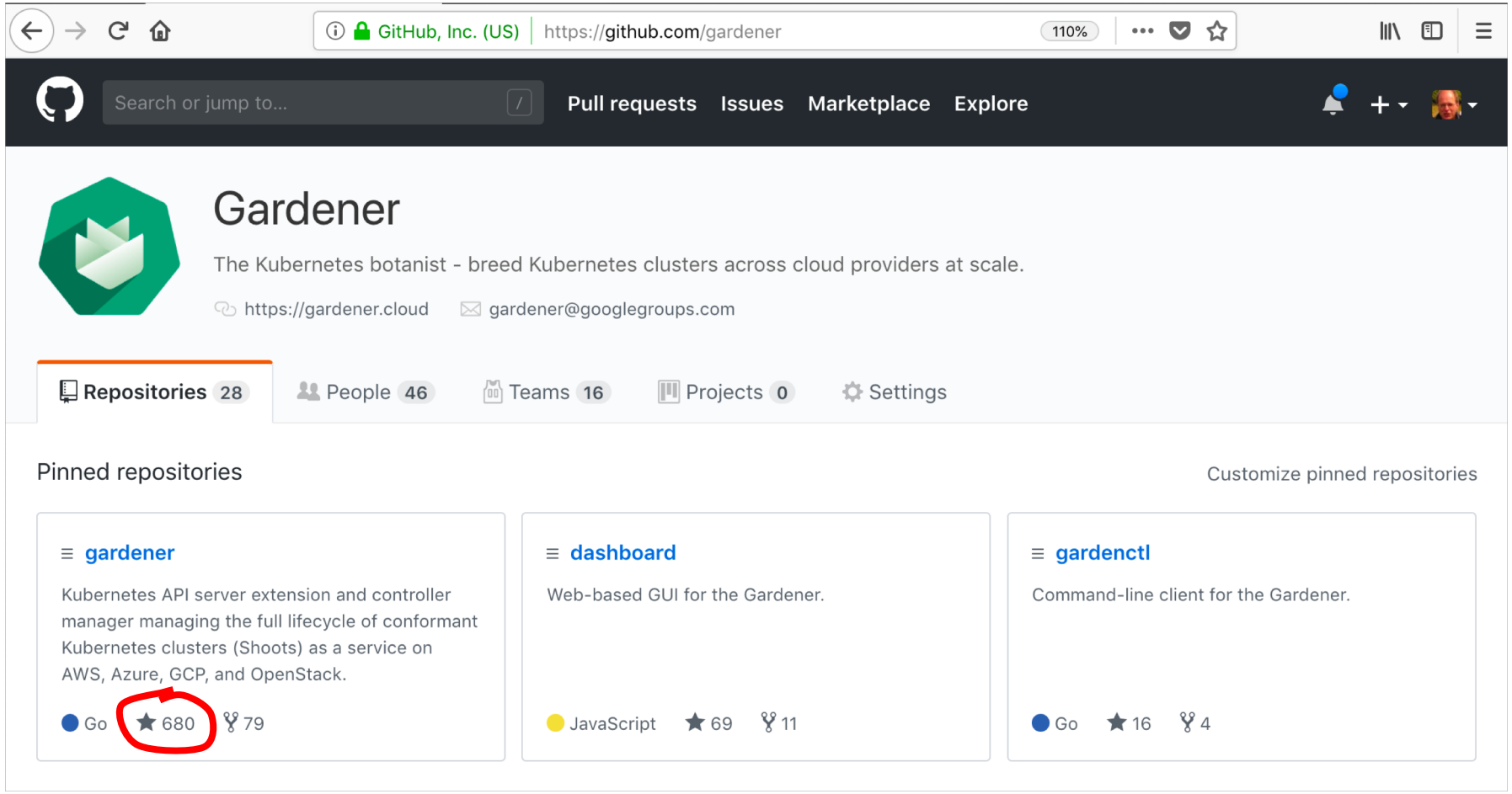
Setting up a Gardener landscape is not trivial,
so we have a community installer:

https://github.com/gardener/landscape-setup

- Many shortcuts to make it simple (Gardener and Seed in a single cluster)
- Do not use productively!
- You can use it as a starter for a productive setup
- Different cluster and different cloud provider accounts recommended

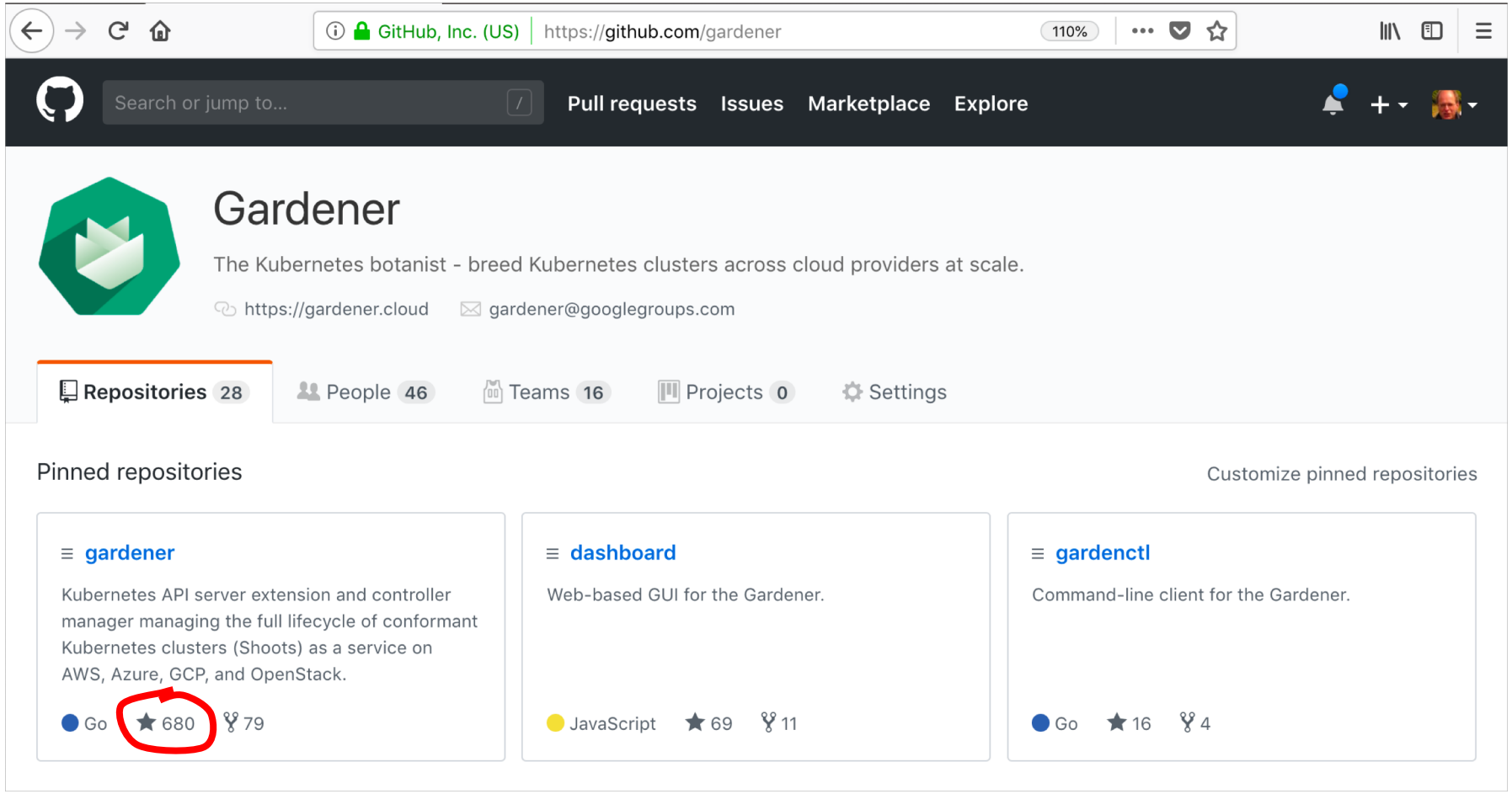
Gardener is Open Source

Apache-2.0



Gardener is Open Source

Apache-2.0

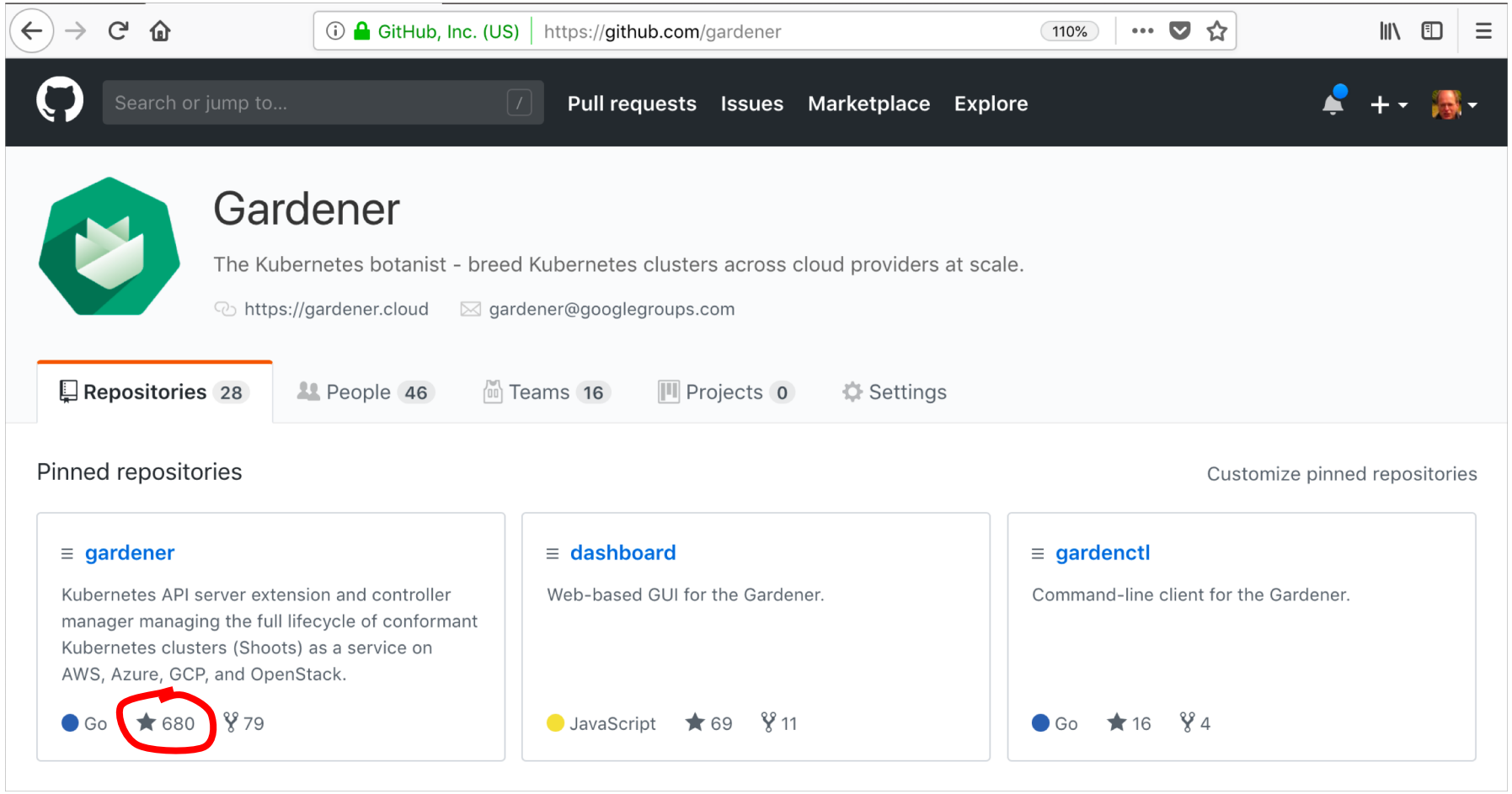


Long-Term Goal

Become CNCF Project

Gardener is Open Source

Apache-2.0



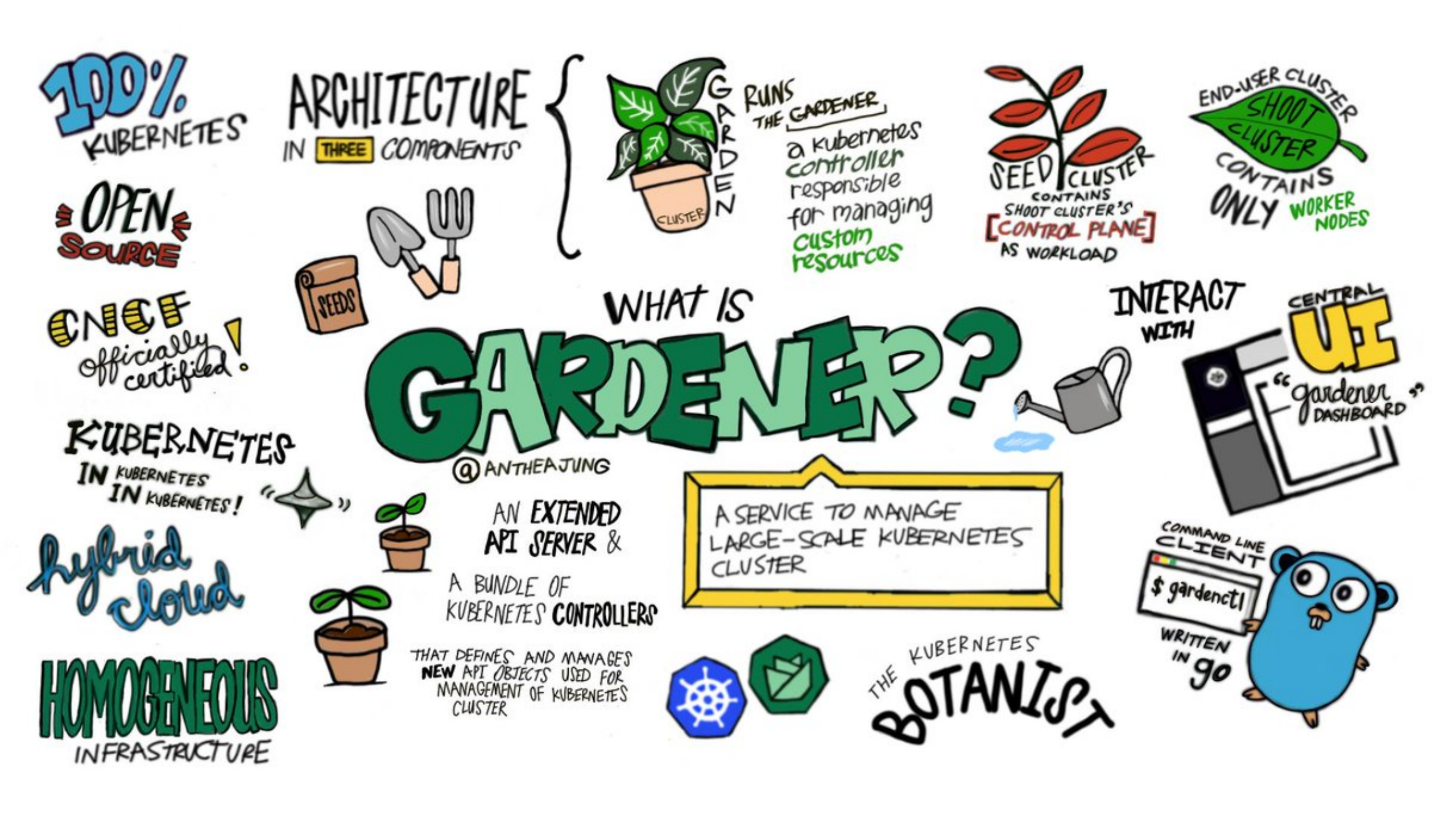
Long-Term Goal

Become CNCF Project

[Gardener Blog]
[CNCF Presentation]
[Kubernetes Podcast]
[Hacker News]
[Reddit]

Thank You!

GitHub	https://github.com/gardener
Home Page	https://gardener.cloud
Wiki	https://github.com/gardener/documentation/wiki
Mailing List	https://groups.google.com/forum/?fromgroups#!forum/gardener
Slack Channel	https://kubernetes.slack.com/messages/gardener
Community Installer	https://github.com/gardener/landscape-setup


100% KUBERNETES
OPEN SOURCE
CNCF officially certified!
KUBERNETES IN KUBERNETES IN KUBERNETES!
hybrid cloud
HOMOGENEOUS INFRASTRUCTURE
ARCHITECTURE IN THREE COMPONENTS
SEEDS
GARDEN
CLUSTER
RUNS THE GARDENER
a kubernetes controller responsible for managing custom resources
SEED CLUSTER
CONTAINS SHOOT CLUSTER'S [CONTROL PLANE] AS WORKLOAD
END-USER CLUSTER
SHOOT CLUSTER
CONTAINS ONLY WORKER NODES
WHAT IS
GARDENER?
@ANTHEAJUNG
AN EXTENDED API SERVER &
A BUNDLE OF KUBERNETES CONTROLLERS
THAT DEFINES AND MANAGES NEW API OBJECTS USED FOR MANAGEMENT OF KUBERNETES CLUSTER
A SERVICE TO MANAGE LARGE-SCALE KUBERNETES CLUSTER
THE KUBERNETES BOTANIST
INTERACT WITH
CENTRAL UI
"gardener DASHBOARD"
COMMAND LINE CLIENT
$ gardenctl
WRITTEN IN go

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

-1 0 +1

eclipsecon Europe 2018 OSGi Community Event 2018

Kubernetes Machine Controller Manager

Problem

- **Node provisioning and de-provisioning** is out of scope of current Kubernetes
- In the beginning we used terraform scripts ➦ unmanageable
- No mechanism
  - to smoothly scale clusters
  - upgrade cluster nodes for all providers

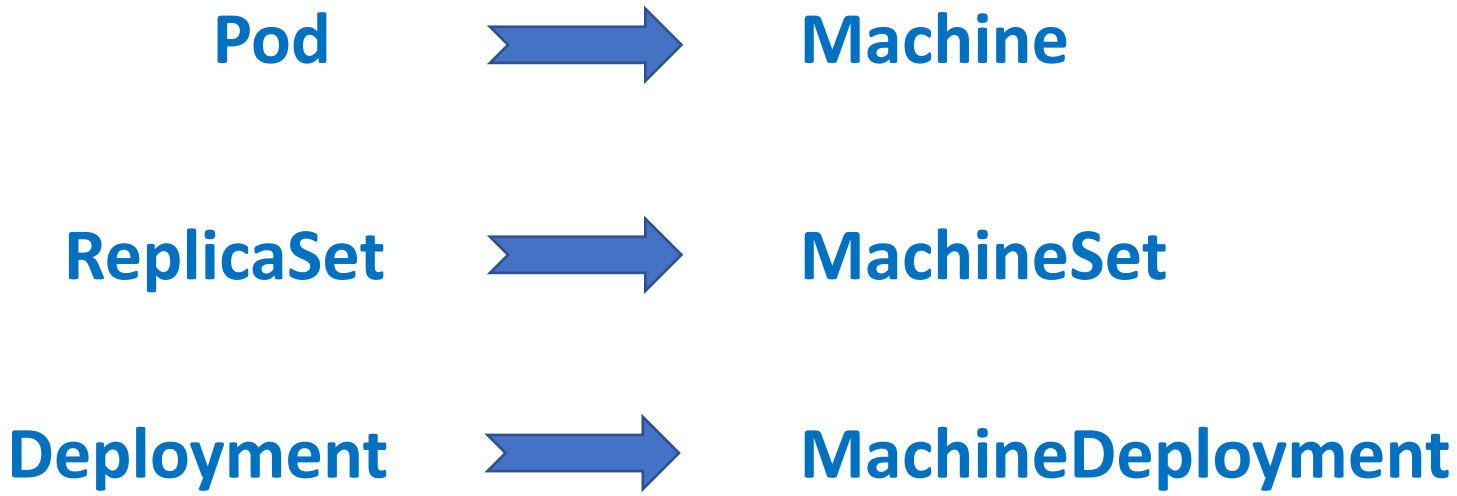
Machine Controller Manager

- **Node custom resources** to manage nodes via k8s API
- **Plugins** enable support for different **cloud providers**
- Enables cluster **auto-scaling** and **upgrade** of cluster nodes

MCM Model

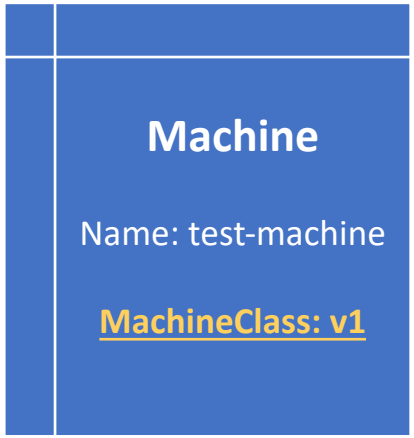
Model for Kubernetes deployments works great

So why not use it for machines?

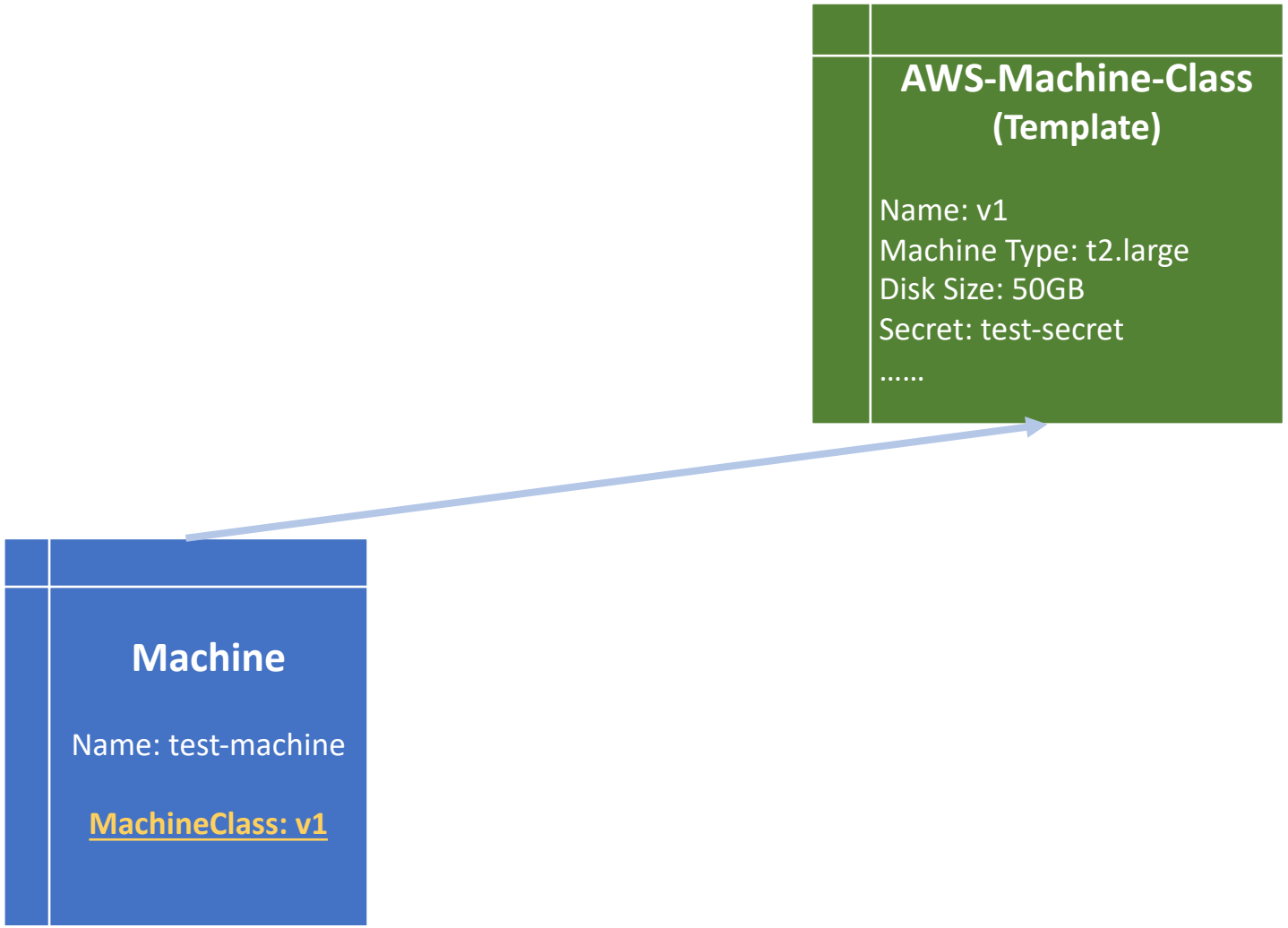


MCM Custom Resources

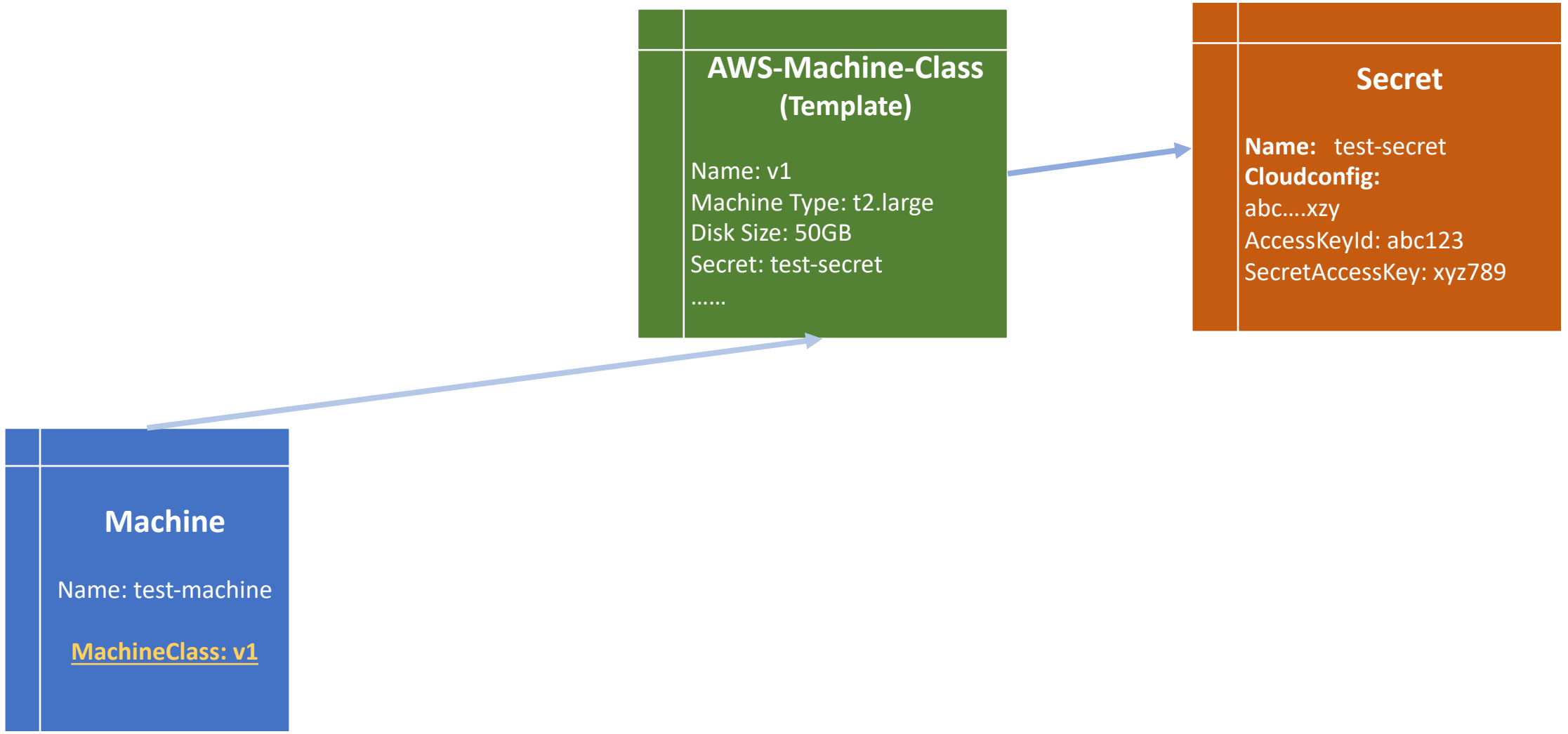
MCM Custom Resources



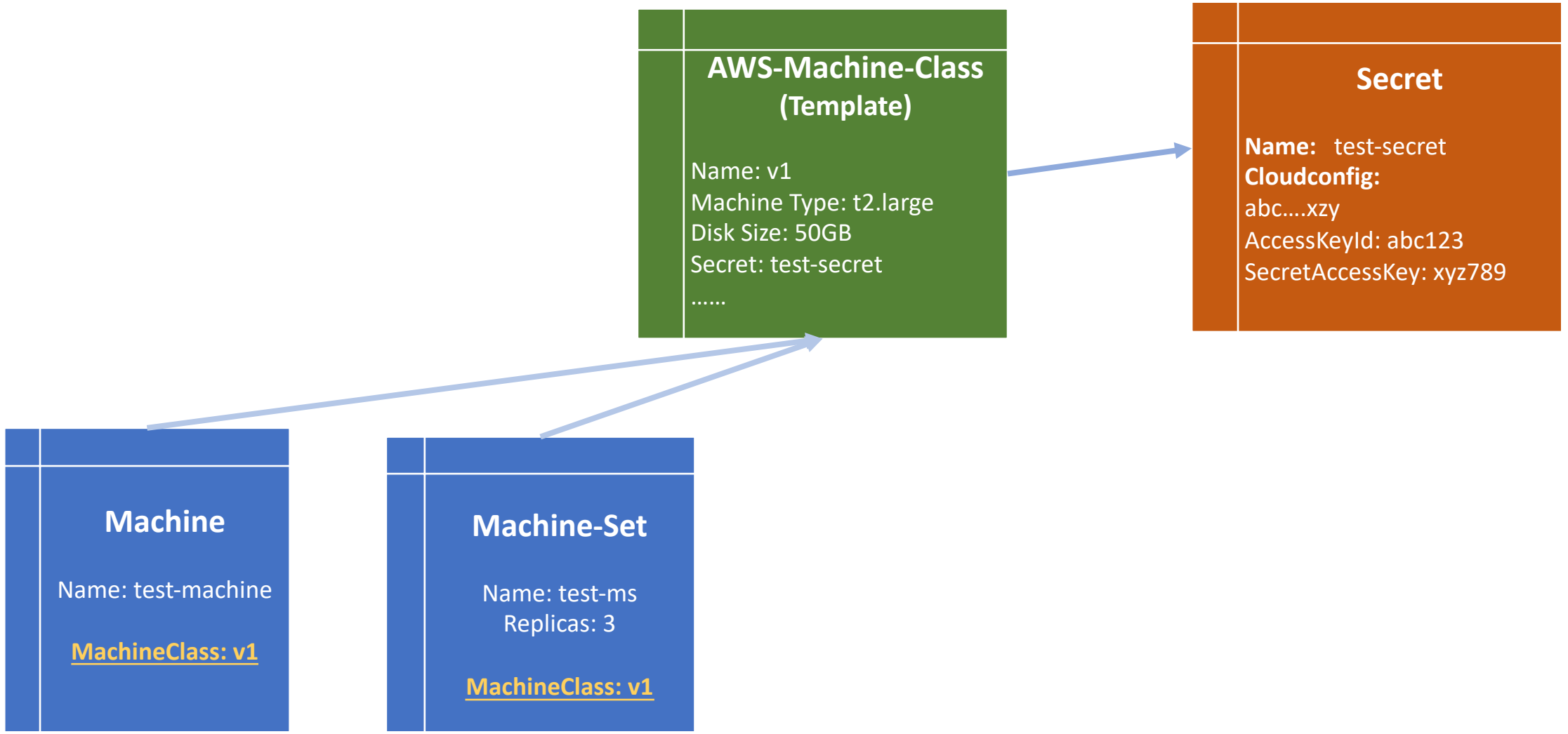
MCM Custom Resources



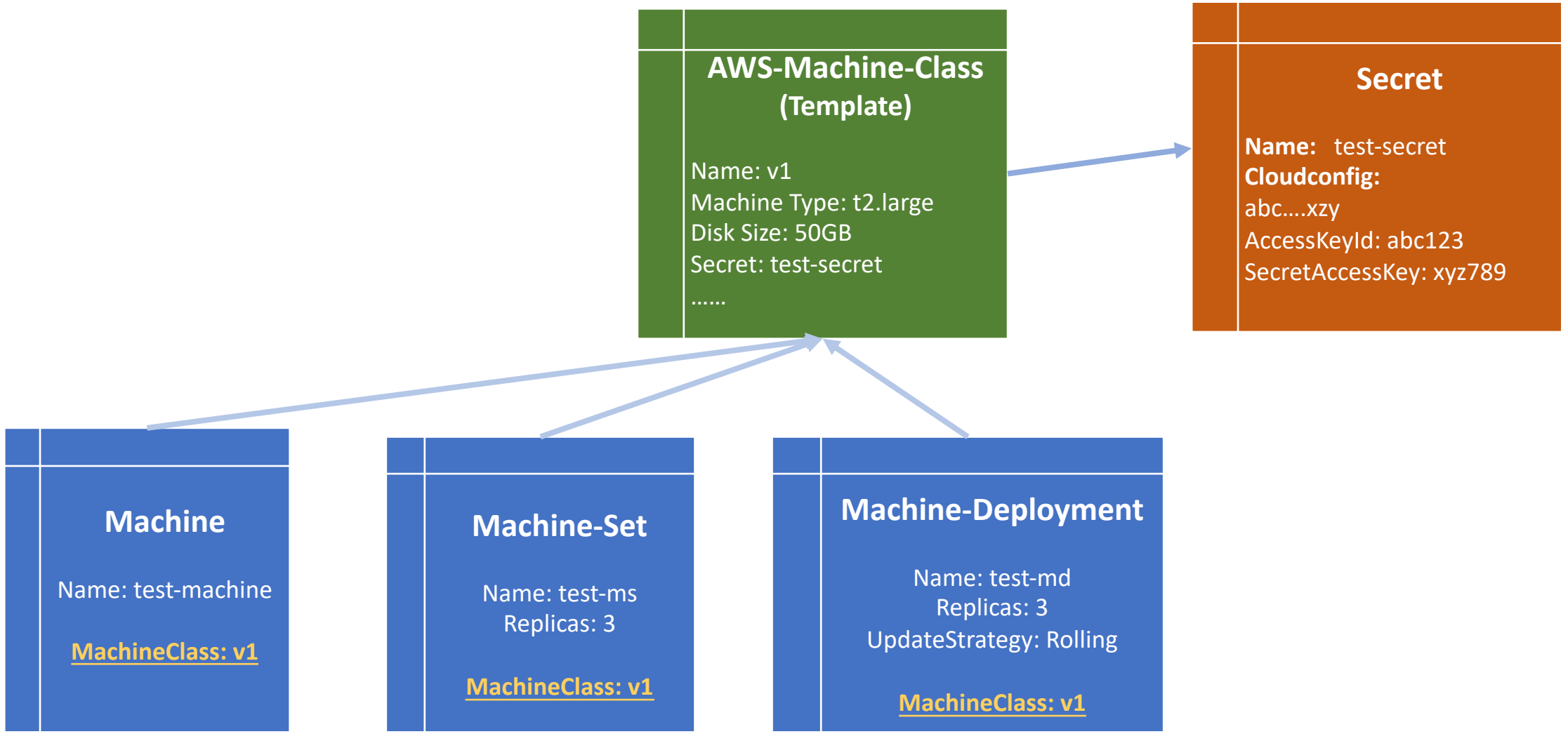
MCM Custom Resources



MCM Custom Resources

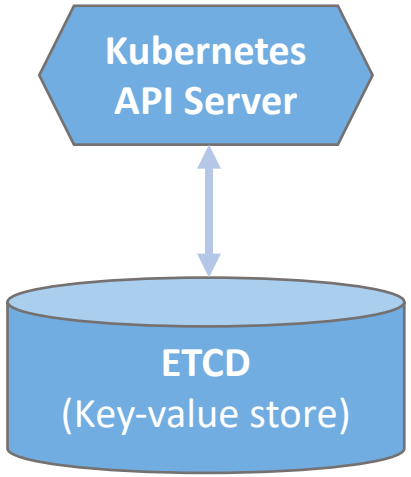


MCM Custom Resources

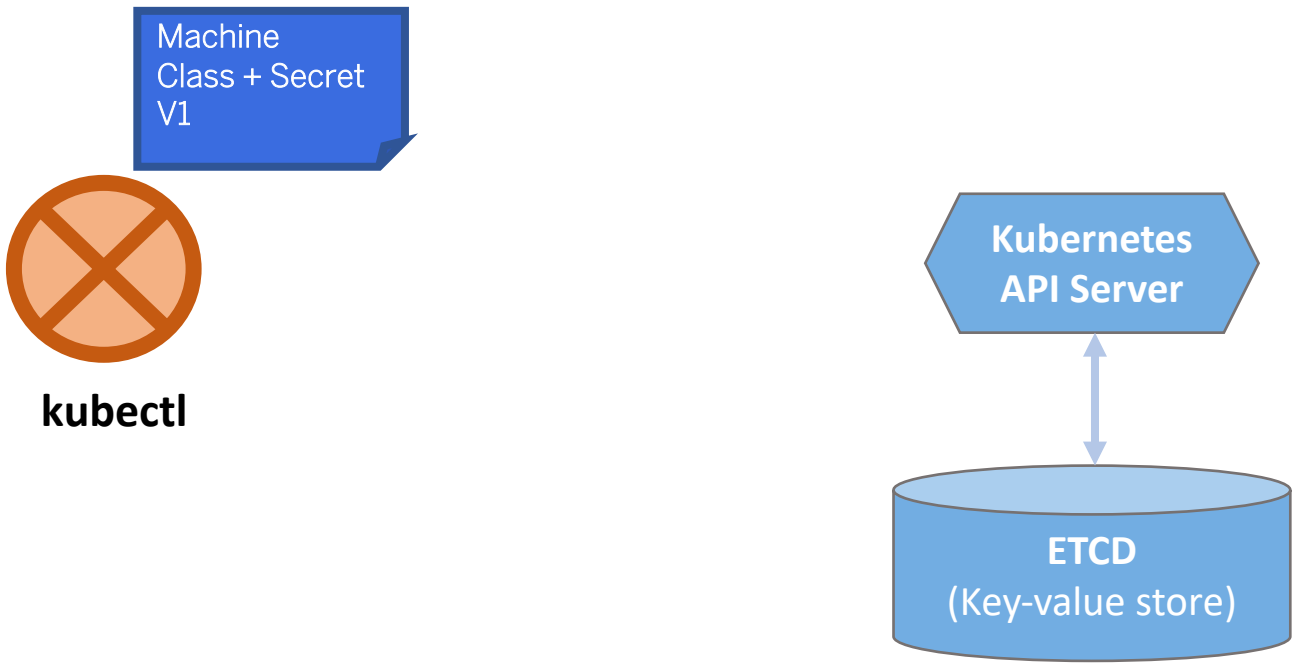


Working of MCM

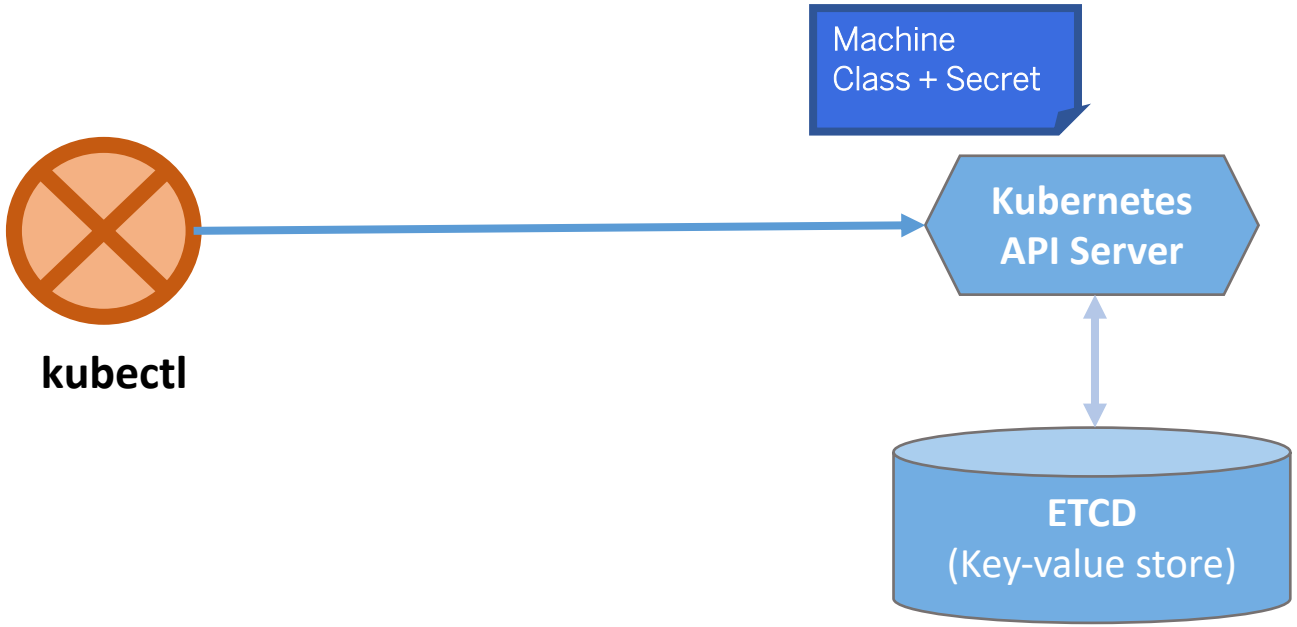
kubectl



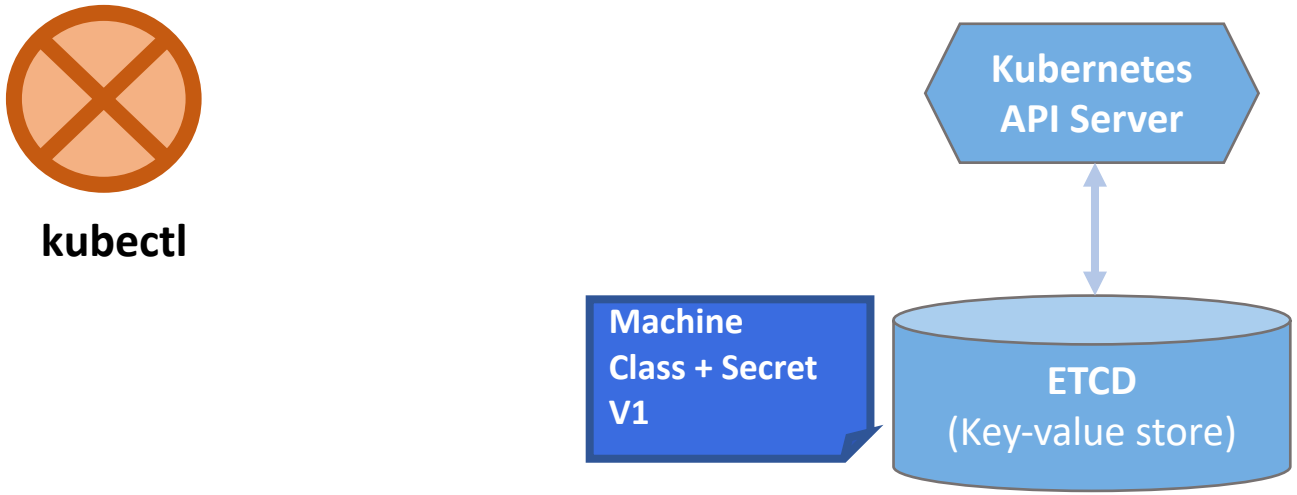
Working of MCM



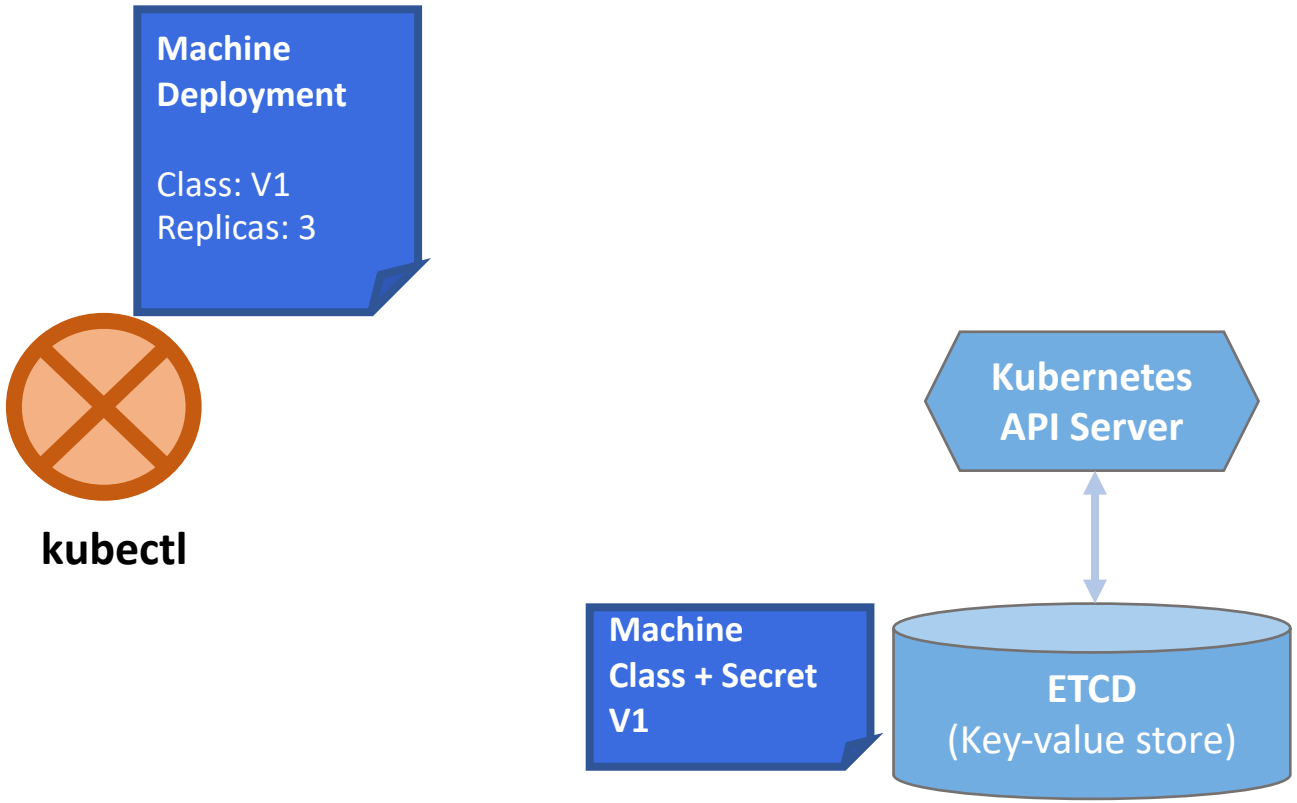
Working of MCM



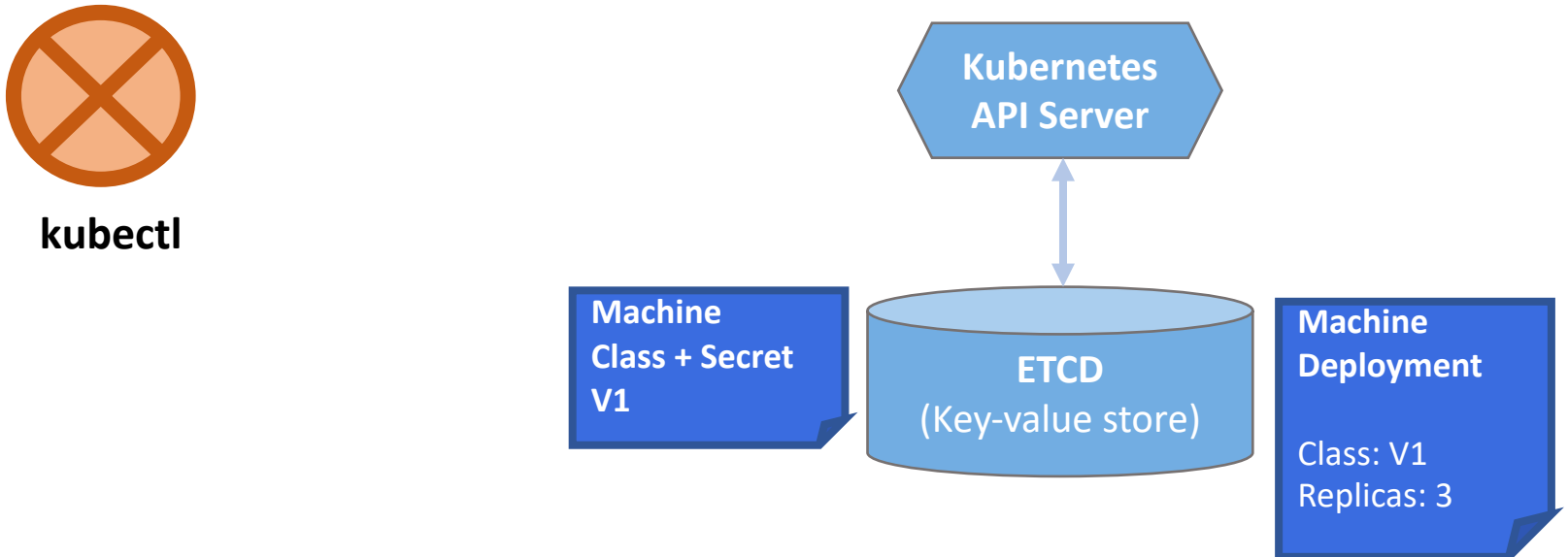
Working of MCM



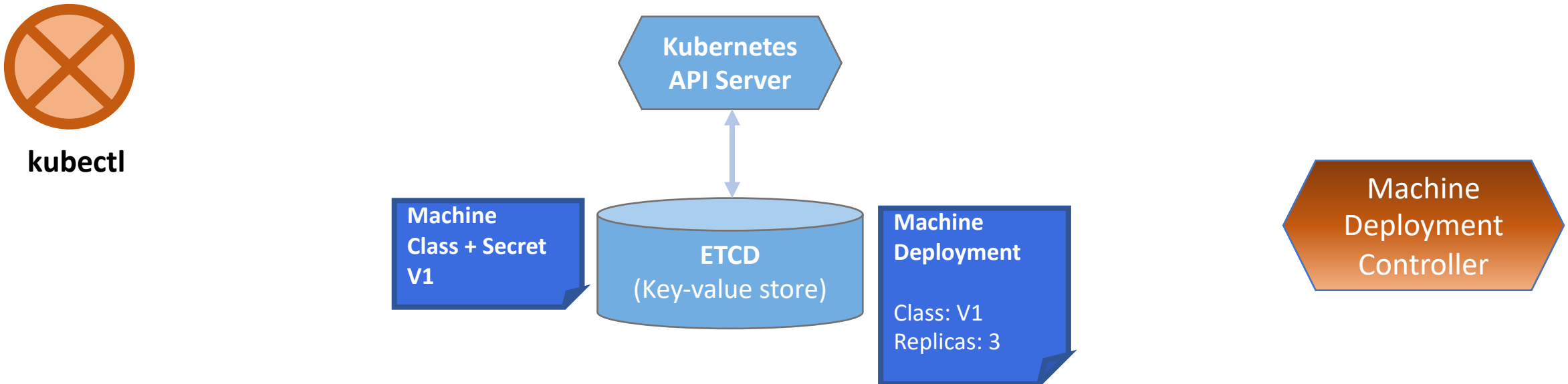
Working of MCM



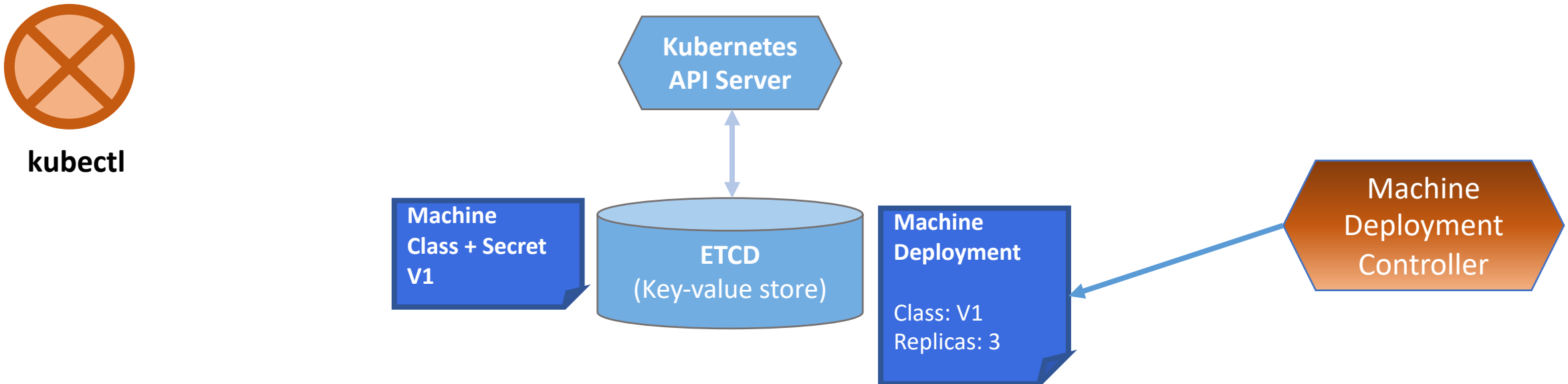
Working of MCM



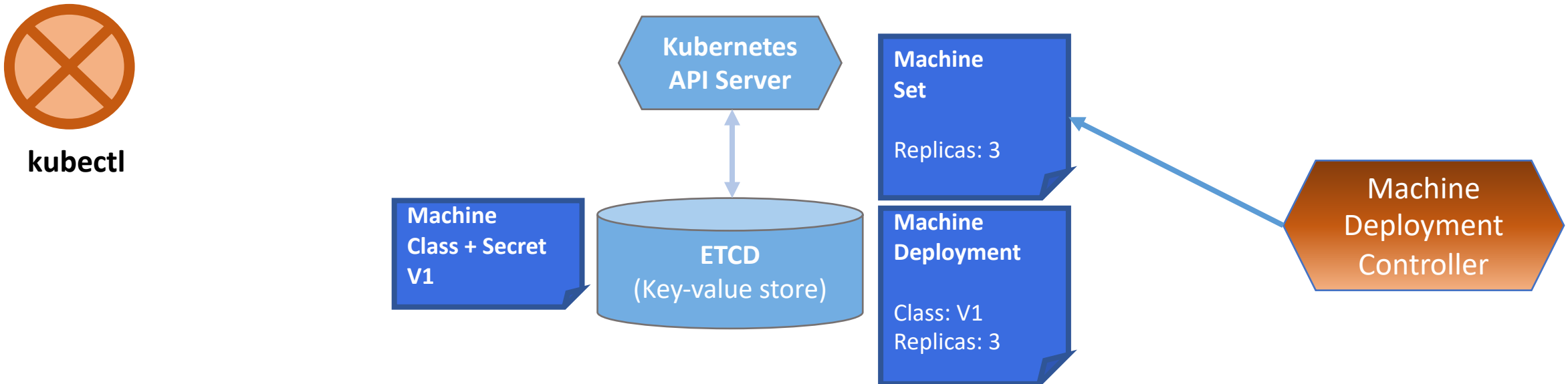
Working of MCM



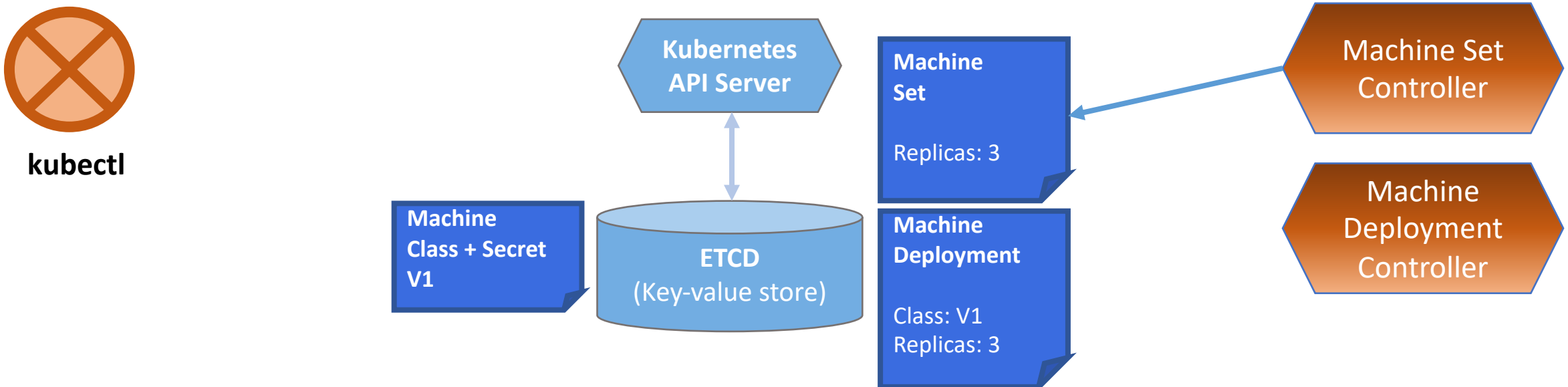
Working of MCM



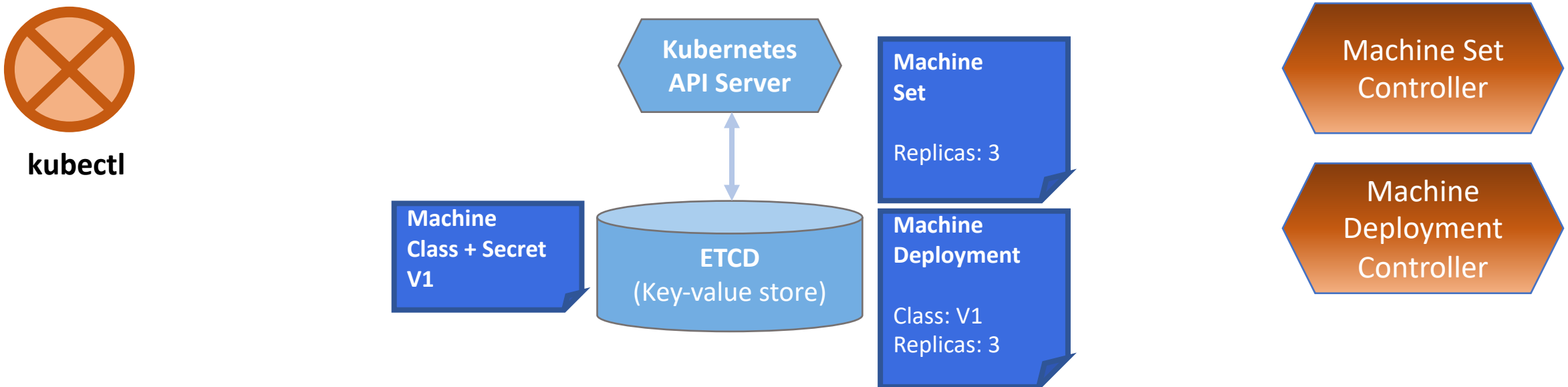
Working of MCM



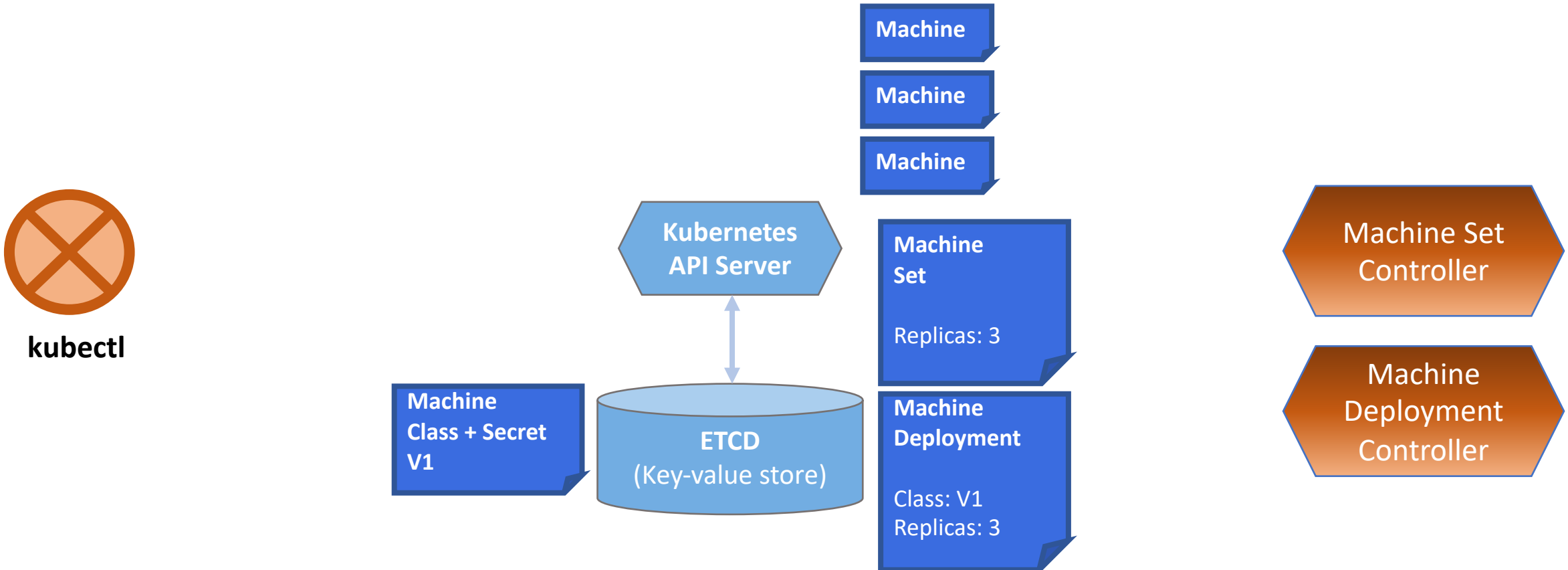
Working of MCM



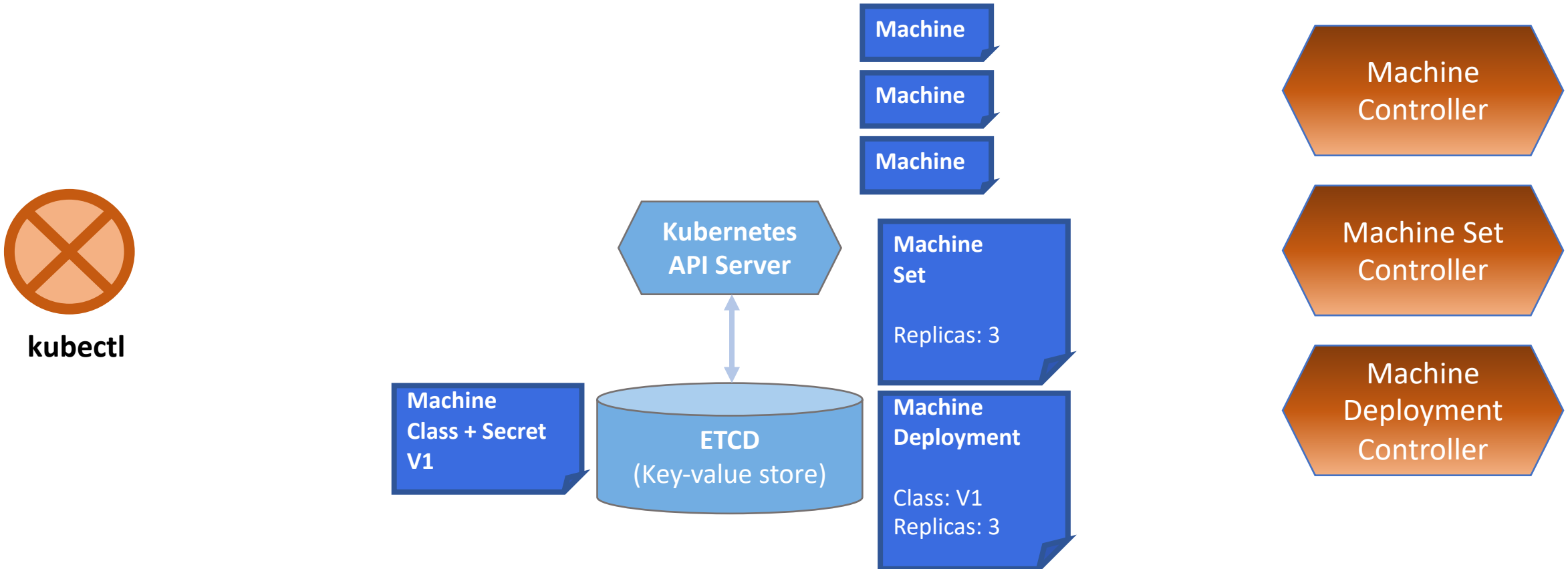
Working of MCM



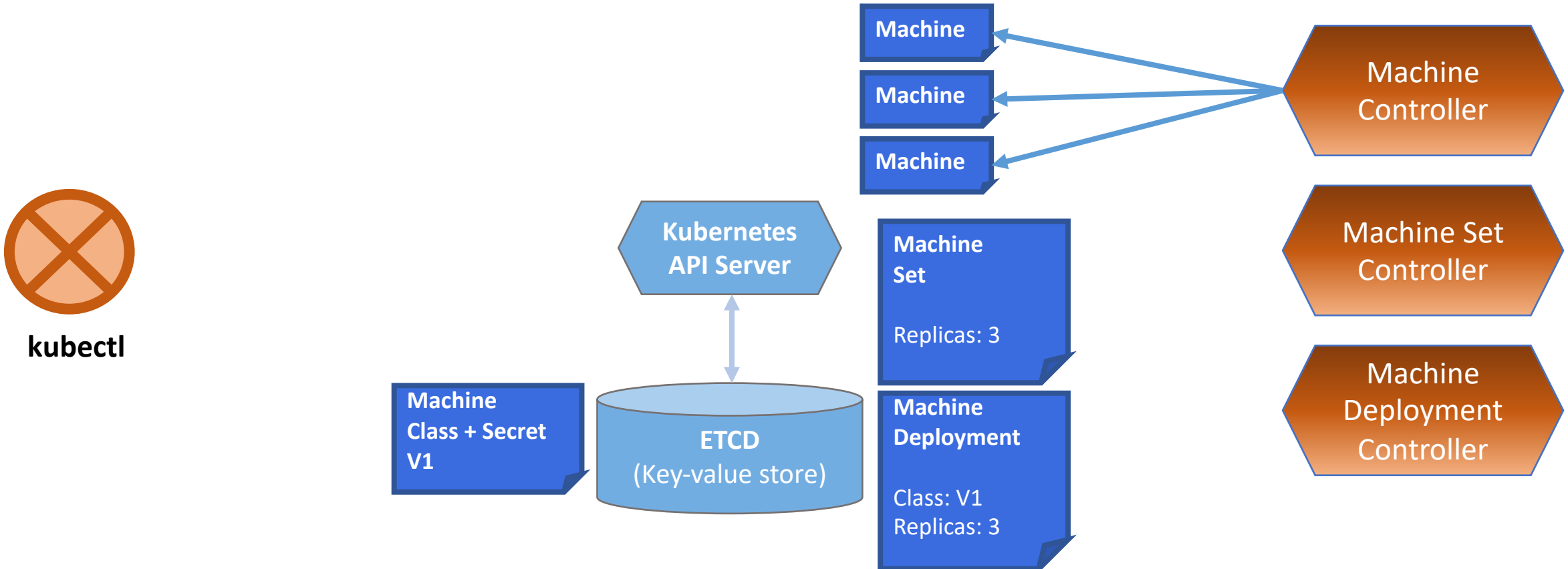
Working of MCM



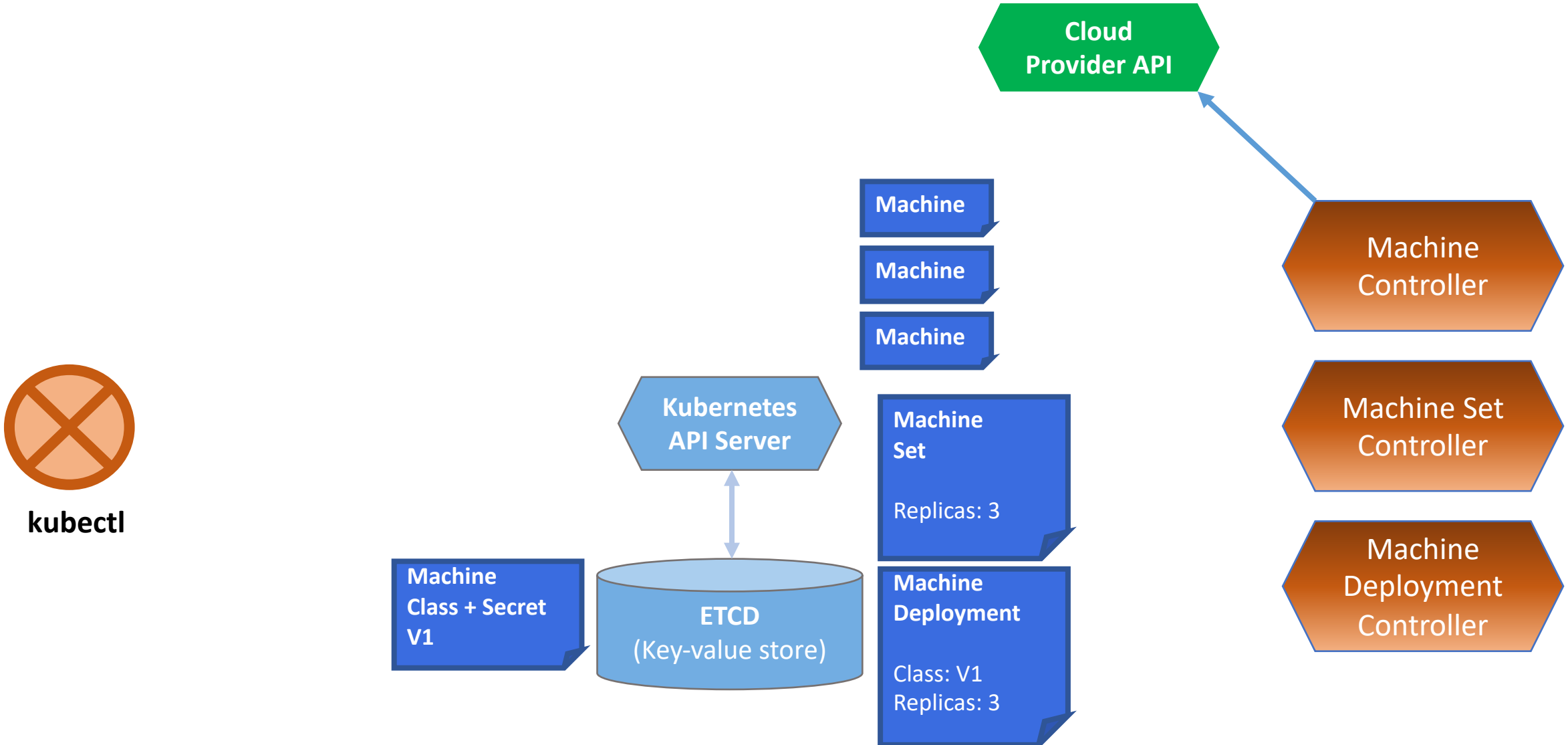
Working of MCM



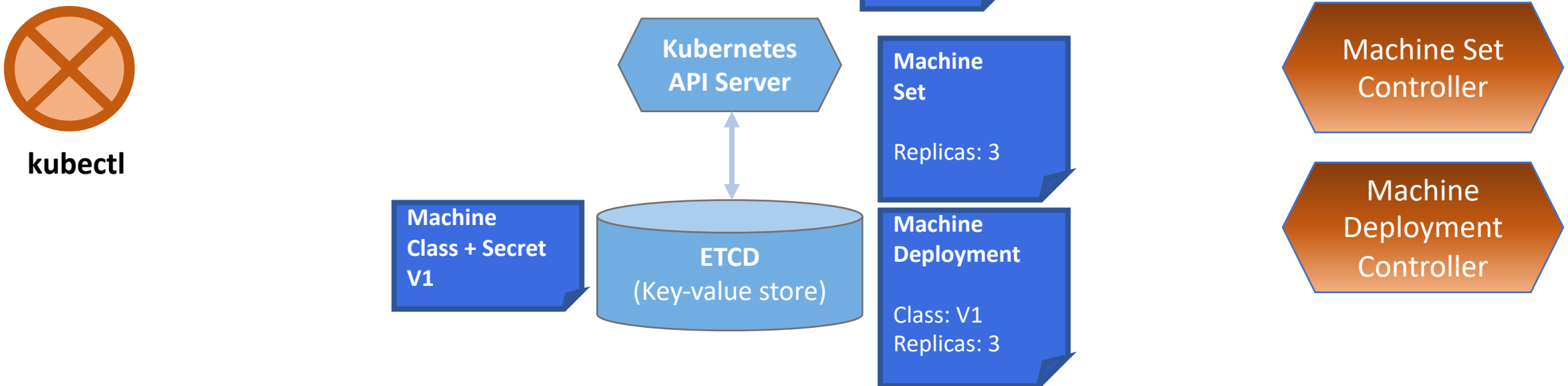
Working of MCM



Working of MCM

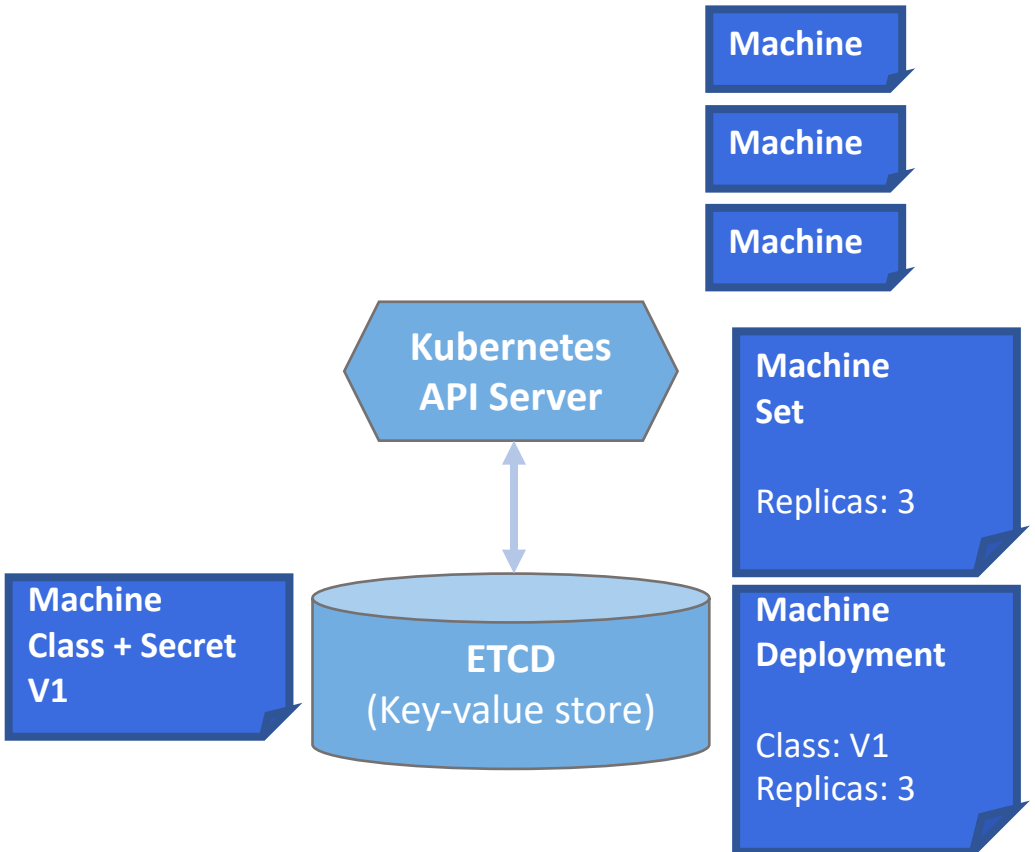


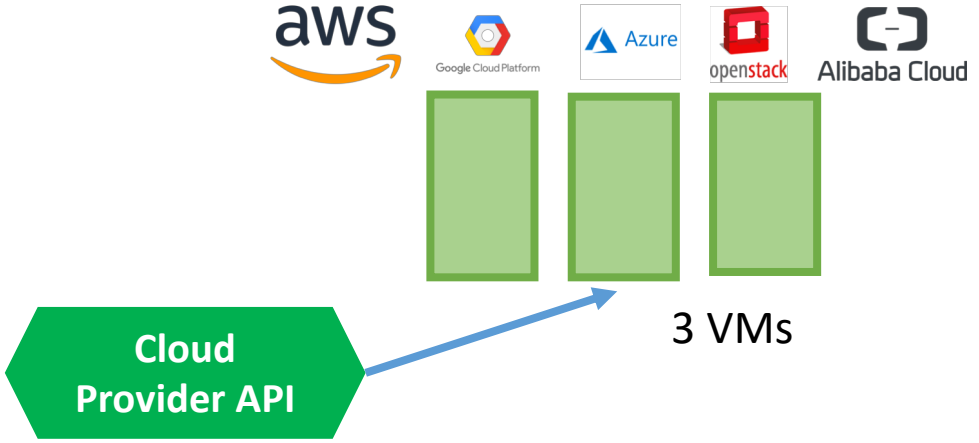
Working of MCM

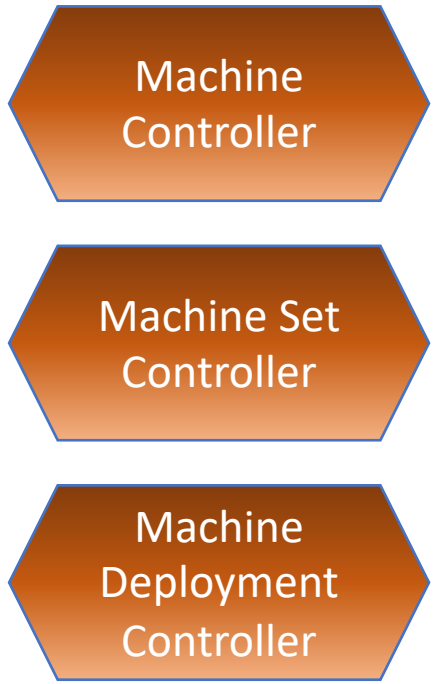


Working of MCM

kubectl

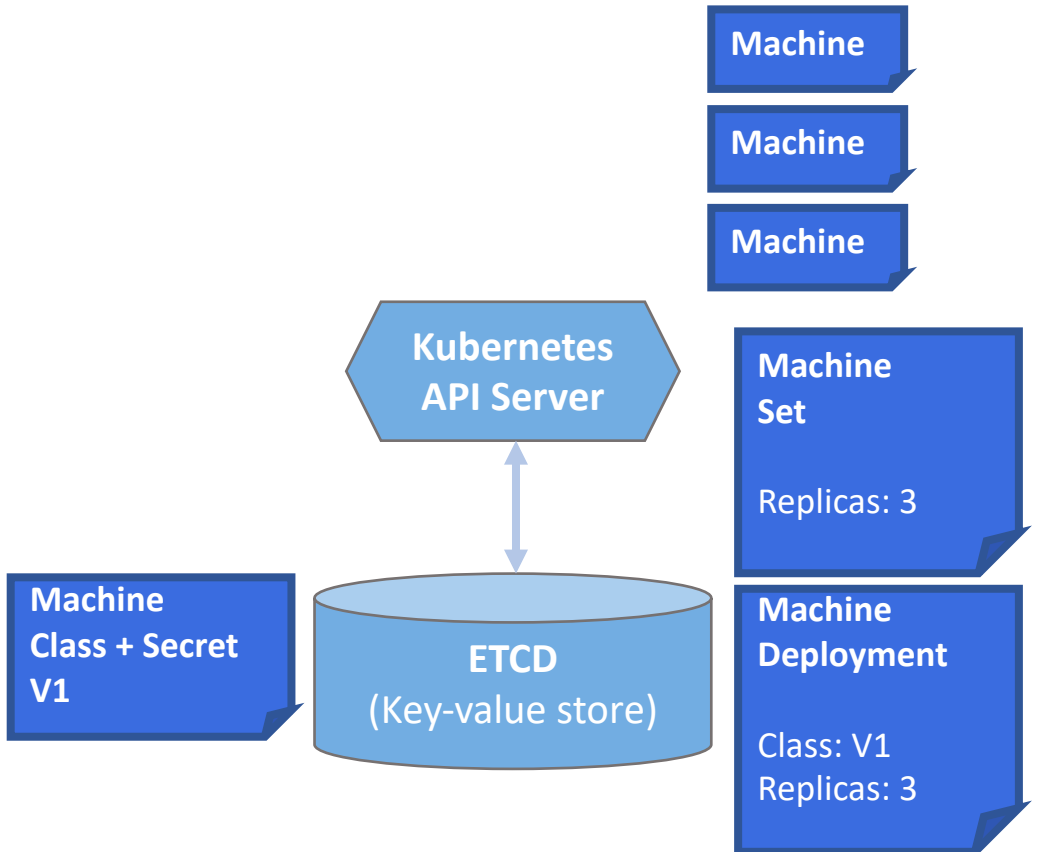






Working of MCM

kubectl

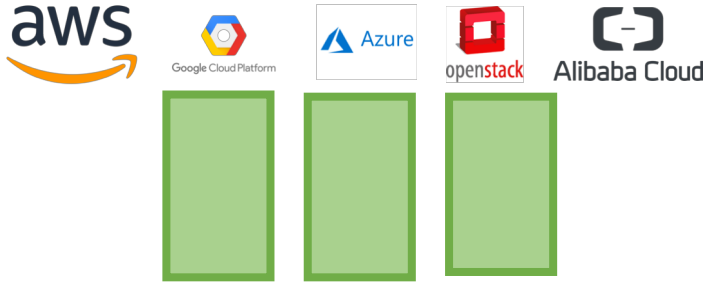


Machine Controller

Machine Set Controller

Machine Deployment Controller

Working of MCM



Kubernetes Controller Manager

Cloud Provider API

Node

Node

Node

Machine

Machine

Machine

Kubernetes API Server

Machine Set

Replicas: 3

Machine Class + Secret V1

ETCD (Key-value store)

Machine Deployment

Class: V1
Replicas: 3

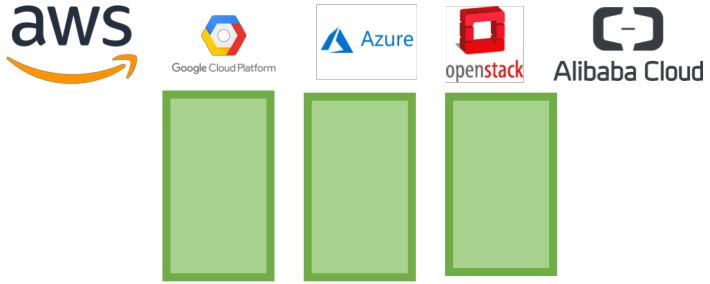
Machine Controller

Machine Set Controller

Machine Deployment Controller

kubectl

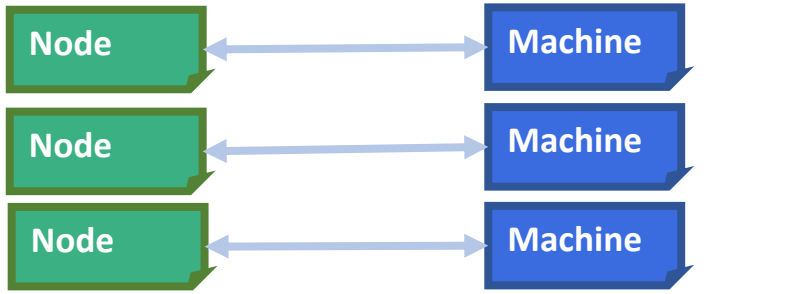
Working of MCM



Kubernetes Controller Manager

Cloud Provider API

Node objects help in monitoring the machine status – Health



Machine Controller

Machine Set Controller

Machine Deployment Controller

kubectl

Kubernetes API Server

Machine Set

Replicas: 3

Machine Class + Secret V1

ETCD (Key-value store)

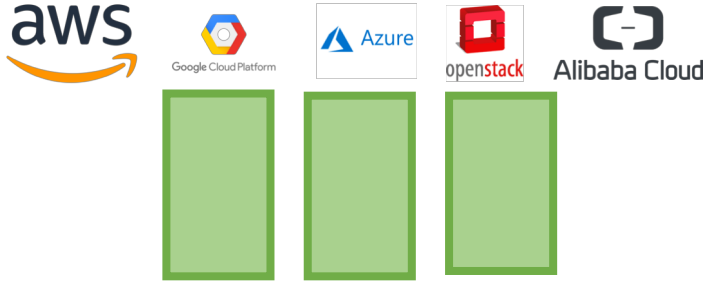
Machine Deployment

Class: V1
Replicas: 3

Working of MCM

kubectl

Kubernetes Controller Manager



3 VMs

Cloud Provider API

Node objects help in monitoring the machine status – Health

Node → **Machine**

Node → **Machine**

Node → **Machine**

Kubernetes API Server

Machine Class + Secret V1

ETCD (Key-value store)

Machine Set

Replicas: 3

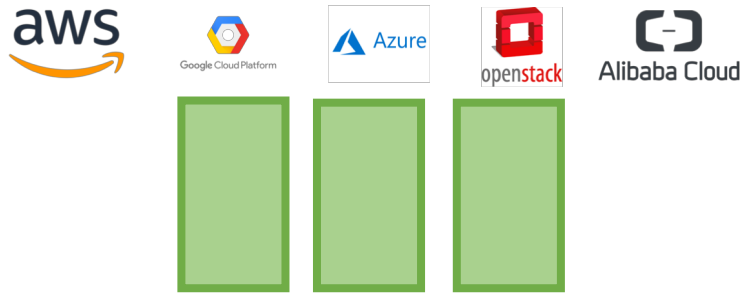
Machine Deployment

Class: V1
Replicas: 3

Machine Controller Manager

- Machine Controller
- Machine Set Controller
- Machine Deployment Controller

Autoscaling



Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

ETCD (Key-value store)

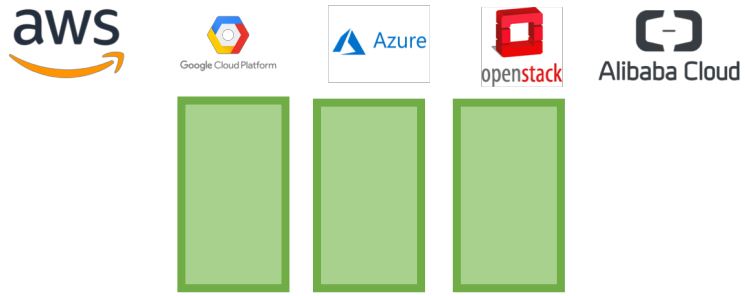
Machine Deployment

Class: V1

Replicas: 3

kubectl

Autoscaling

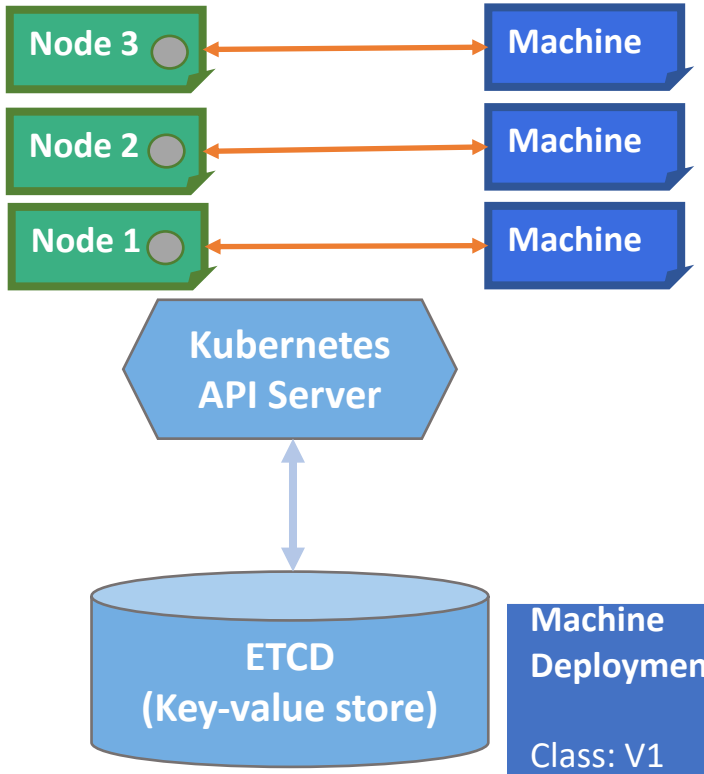


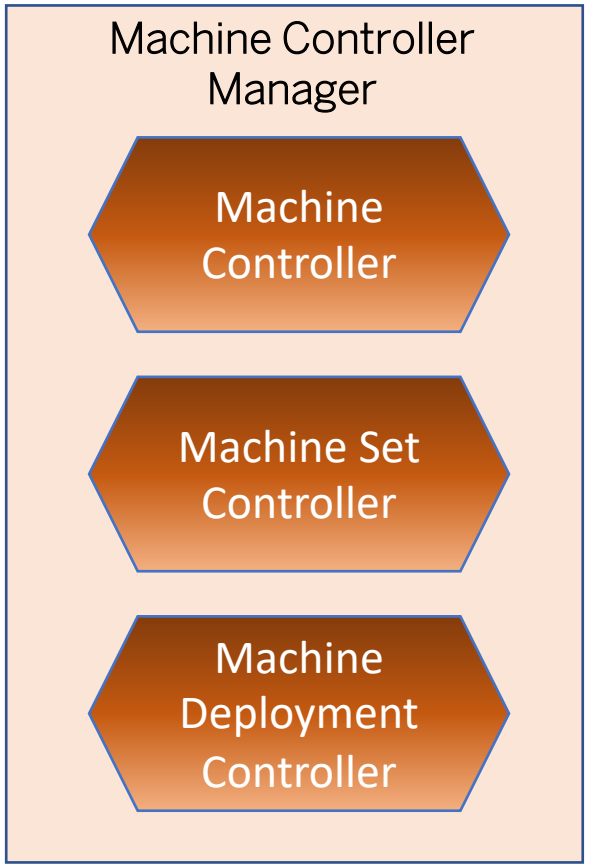
Kubernetes Controller Manager

Now assume that all the nodes resources are nearly consumed and a new pod is created

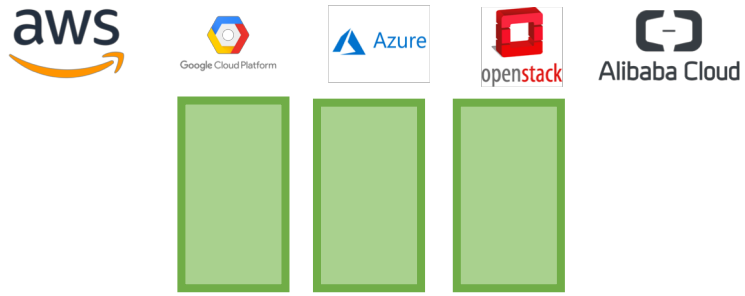
Cloud Provider API

kubectl





Autoscaling



Kubernetes Controller Manager

Cloud Provider API

Pod

Image: Nginx
Node: -

kubectl

Node 3 — Machine

Node 2 — Machine

Node 1 — Machine

Kubernetes API Server

ETCD (Key-value store)

Machine Deployment

Class: V1
Replicas: 3

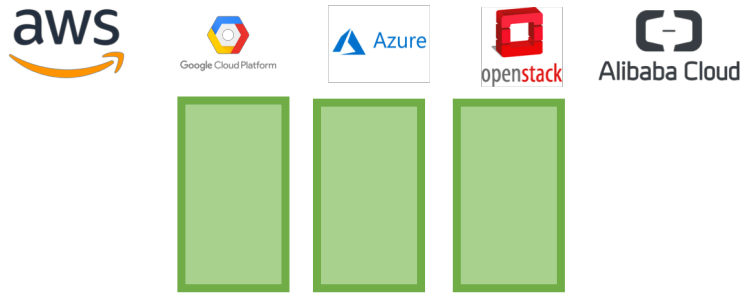
Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Autoscaling



Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

ETCD (Key-value store)

kubectl

Pod

Image: Nginx

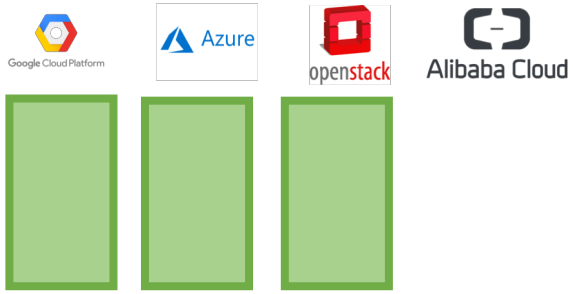
Node: Unschedulable

Machine Deployment

Class: V1

Replicas: 3

Autoscaling



Kubernetes Controller Manager

Forked Cluster Autoscaler

Cloud Provider API

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

kubectl

ETCD (Key-value store)

Pod

Image: Nginx

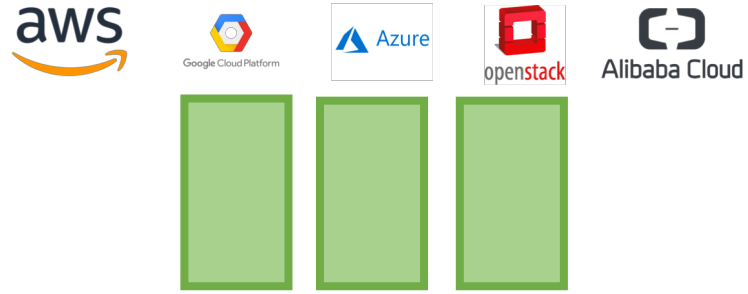
Node: Unschedulable

Machine Deployment

Class: V1

Replicas: 3

Autoscaling



Kubernetes Controller Manager

Forked Cluster Autoscaler

Cloud Provider API

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

kubectl

Pod

Image: Nginx

Node: Unschedulable

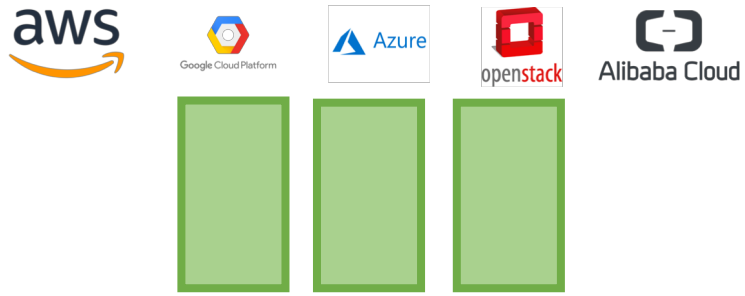
ETCD (Key-value store)

Machine Deployment

Class: V1

Replicas: 3

Autoscaling



Kubernetes Controller Manager

Forked Cluster Autoscaler

Cloud Provider API

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

kubectl

Pod

Image: Nginx

Node: Unschedulable

ETCD (Key-value store)

Machine Deployment

Class: v1

Replicas: 3

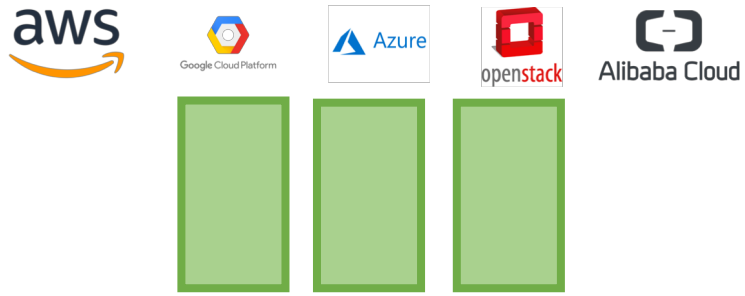
Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Autoscaling



Kubernetes Controller Manager

Forked Cluster Autoscaler

Cloud Provider API

Node 3

Node 2

Node 1

Machine

Machine

Machine

Kubernetes API Server

kubectl

Pod

Image: Nginx

Node: Unschedulable

ETCD (Key-value store)

Machine Deployment

Class: V1

Replicas: 4

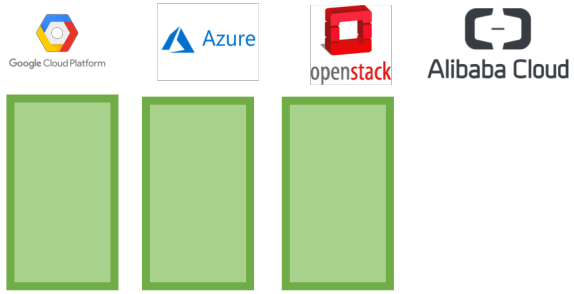
Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Autoscaling



Kubernetes Controller Manager

Forked Cluster Autoscaler

Cloud Provider API

Machine

Node 3

Machine

Node 2

Machine

Node 1

Machine

Kubernetes API Server

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

kubectl

Pod

Image: Nginx

Node: Unschedulable

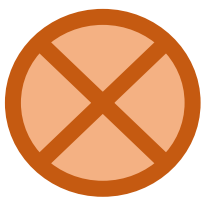
ETCD (Key-value store)

Machine Deployment

Class: V1

Replicas: 4

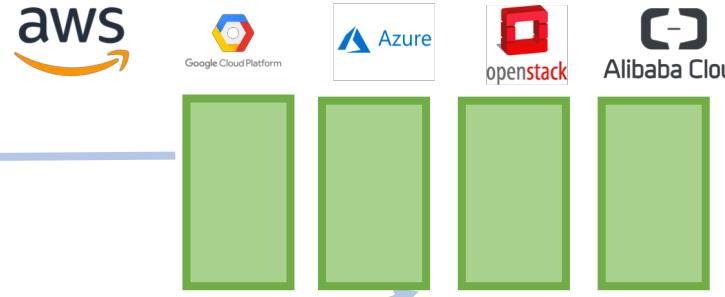
Autoscaling



kubectl

Forked Cluster Autoscaler

Kubernetes Controller Manager



Cloud Provider API

Node 4 — Machine

Node 3 — Machine

Node 2 — Machine

Node 1 — Machine

Kubernetes API Server

ETCD (Key-value store)

Pod

Image: Nginx

Node: Node4

Machine Deployment

Class: V1

Replicas: 4

Machine Controller Manager

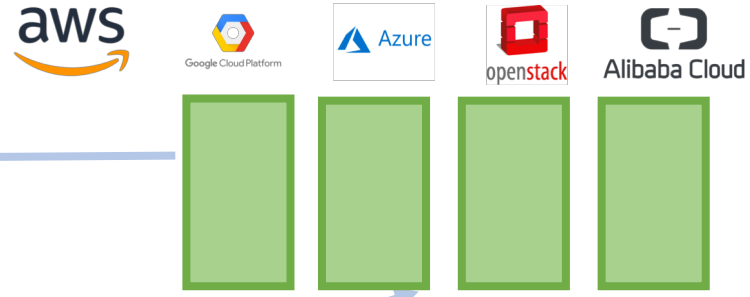
Machine Controller

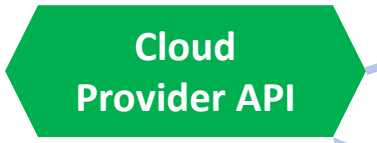
Machine Set Controller

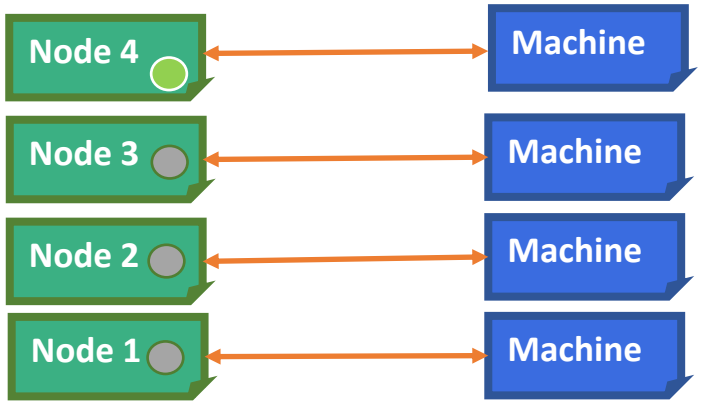
Machine Deployment Controller

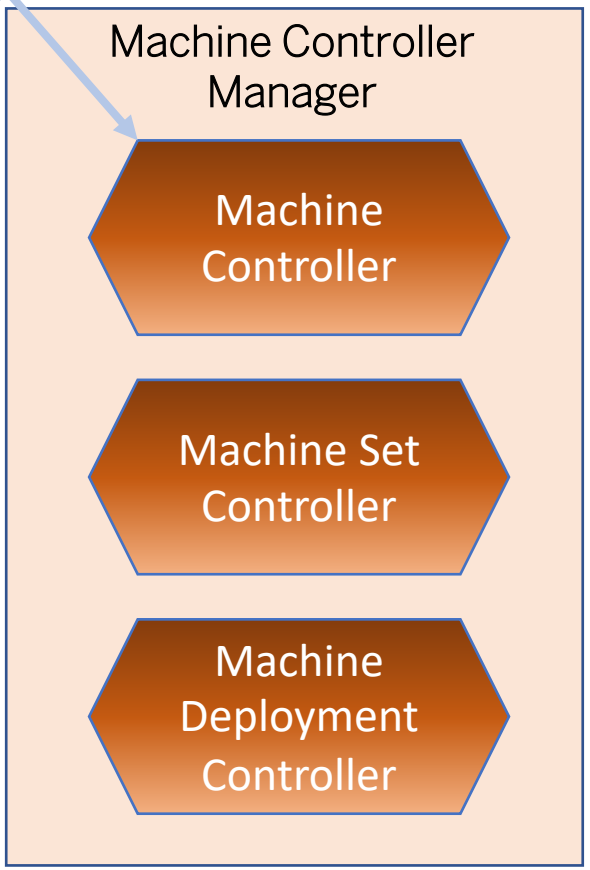
Autoscaling

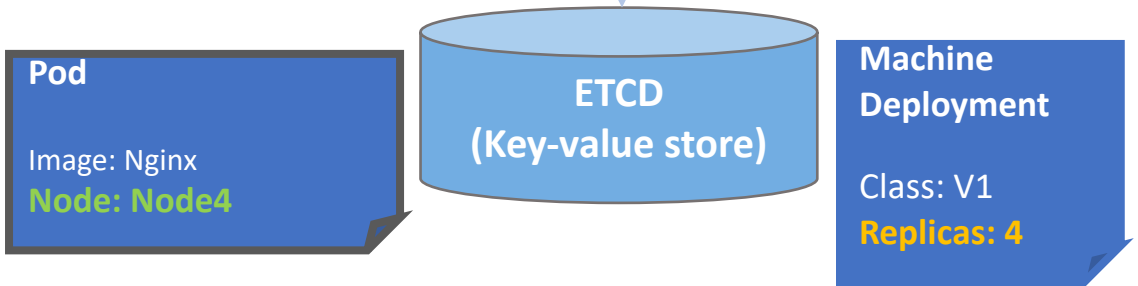
kubectl



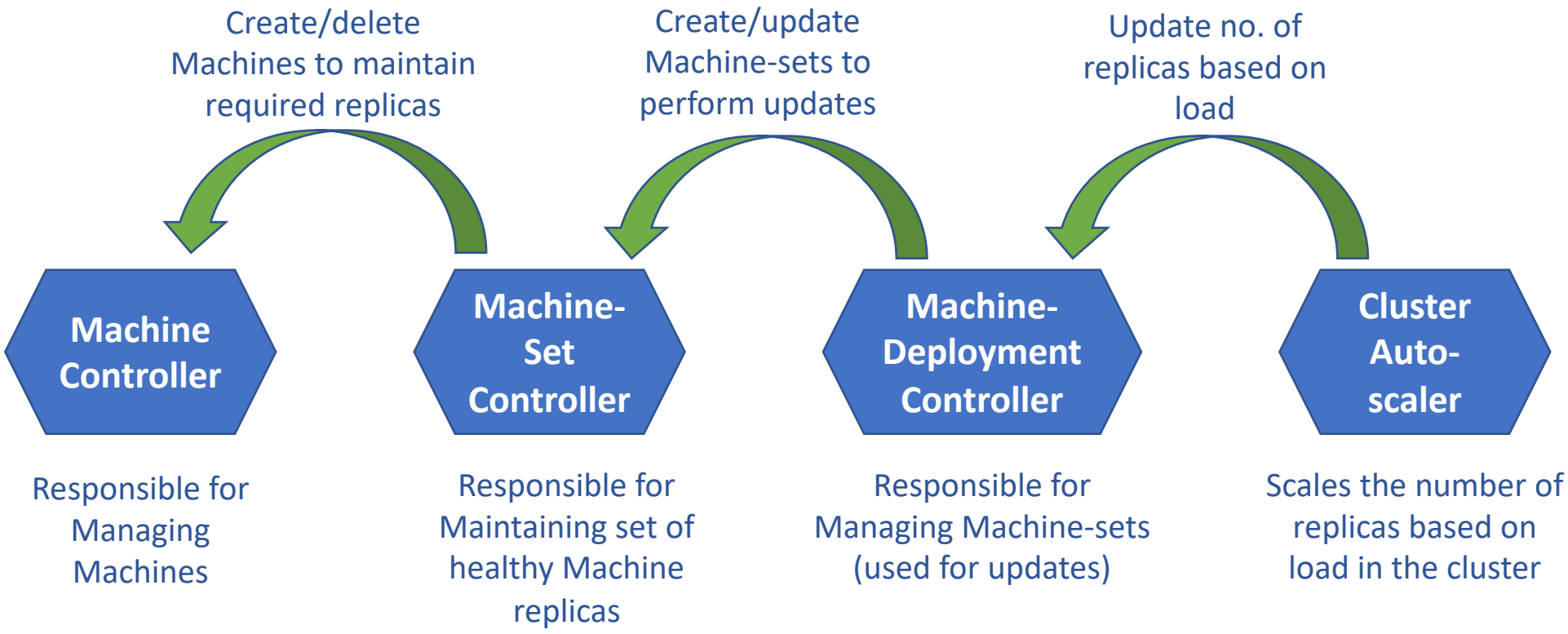








Machine Controller Manager - Components



Parent-child relationship: Adoption of orphaned children

Controllers cooperate, rather than racing with each other !

Machine Controller Manager