Kubernetes Clusters as a Service

Gardener

The Kubernetes Botanist
From Containers to Kubernetes
From Containers to Kubernetes

Container

<table>
<thead>
<tr>
<th>Container Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host OS</td>
</tr>
<tr>
<td>VM</td>
</tr>
</tbody>
</table>
From Containers to Kubernetes

Container

- Container Runtime
- Host OS
- VM
From Containers to Kubernetes

Benefits
Isolation
Immutable infrastructure
Portability
Faster deployments
Versioning
Ease of sharing
From Containers to Kubernetes

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

**Benefits**
- Isolation
- Immutable infrastructure
- Portability
- Faster deployments
- Versioning
- Ease of sharing

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

**Diagram Labels:**
- Container
- Container Runtime
- Host OS
- VM
- Container Scheduler
From Containers to Kubernetes

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

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

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

Container
- Container Runtime
- Host OS
- VM
From Containers to Kubernetes

Benefits
- Isolation
- Immutable infrastructure
- Portability
- Faster deployments
- Versioning
- Ease of sharing

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

Kubernetes
Orchestration of cluster of containers across multiple hosts
- Automatic placements, networking, deployments, scaling, roll-out/-back, A/B testing
From Containers to Kubernetes

Benefits
- Isolation
- Immutable infrastructure
- Portability
- Faster deployments
- Versioning
- Ease of sharing

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

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

Container Scheduler
From Containers to Kubernetes

**Benefits**
- Isolation
- Immutable infrastructure
- Portability
- Faster deployments
- Versioning
- Ease of sharing

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

**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
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 Cluster**

**Seed Cluster**

- **Worker**
- **Master**
- **HA**
Gardener Kubernetes Cluster Setup

Gardener Cluster

Seed Cluster
Gardener Kubernetes Cluster Setup

Gardener Cluster

Seed Cluster

Shoot Clusters
**Gardener** Kubernetes Cluster Setup

**Gardener** Cluster

**Seed** Cluster

**Shoot** Clusters

HA

Master

Worker

Master

Worker

Master

Worker

Master

Worker

Worker

Worker

Worker

Worker
Gardener Kubernetes Cluster Setup

- **Gardener Cluster**
  - Master
  - Worker
  - HA

- **Seed Cluster**
  - Master
  - Worker
  - HA

- **Shoot Clusters**
  - Worker
  - Worker
  - Worker
  - Worker
  - Worker
  - Worker
  - Worker
  - Worker

Gardener Kubernetes Cluster Setup

Zooming into the Seed Cluster reveals...

Inside a Seed Cluster Worker
Gardener Kubernetes Cluster Setup

Zooming into the Seed Cluster reveals...
Multiple Shoot Cluster Control Planes
Gardener Kubernetes Cluster Setup

Zooming into the Seed Cluster reveals...

Multiple Shoot Cluster Control Planes
Zooming into the Seed Cluster reveals...

Multiple Shoot Cluster Control Planes
Gardener Kubernetes Cluster Setup

Gardener Cluster

Seed Cluster

Shoot Clusters

Zooming into the Seed Cluster reveals...

Multiple Shoot Cluster Control Planes

Inside a Seed Cluster Worker

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:
  ...
```
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:
...
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:
```

**Gardener or Self-Managed DNS**

Define Your Infrastructure Needs
Gardener or Self-Managed DNS

Define Your Infrastructure Needs

Specify Worker Pools

Native Kubernetes Resource

---

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:
  ...
Native Kubernetes Resource

Gardener or Self-Managed DNS

Define Your Infrastructure Needs

Specify Worker Pools

Set Kubernetes Version

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

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:
    ...
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 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
**Lingua Franca – Gardener Cluster Resource**

- **Native Kubernetes Resource**
  - `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`:
    - ...

- **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**
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

Avoid Vendor Lock-In

Native Kubernetes Resource

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:
  ...
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.
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.
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.
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.
Following the Design Principle **Gardener** uses...

Kubernetes as deployment underlay
Following the Design Principle **Gardener** uses...

Kubernetes as deployment underlay

---

**K8S building blocks**

- Deployments
- Replicasets
- Pods
Following the Design Principle **Gardener** uses...

Kubernetes as deployment underlay

K8S building blocks
Following the Design Principle **Gardener** uses...

Kubernetes as deployment underlay

<table>
<thead>
<tr>
<th>Deployments</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicasets</td>
<td>Load-Balancer</td>
</tr>
<tr>
<td>Pods</td>
<td></td>
</tr>
</tbody>
</table>
Following the Design Principle **Gardener** uses...

- Deployments
- Replicasets
- Pods
- Jobs
- Load-Balancer
- Secrets
- Config Maps

**K8S building blocks**

Kubernetes as deployment underlay
Following the Design Principle **Gardener** uses...

<table>
<thead>
<tr>
<th>K8S building blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployments</td>
</tr>
<tr>
<td>Replicasets</td>
</tr>
<tr>
<td>Pods</td>
</tr>
<tr>
<td>Jobs</td>
</tr>
<tr>
<td>Load-Balancer</td>
</tr>
<tr>
<td>Secrets</td>
</tr>
<tr>
<td>Config Maps</td>
</tr>
<tr>
<td>Stateful Sets</td>
</tr>
<tr>
<td>PVs</td>
</tr>
<tr>
<td>PVCs</td>
</tr>
<tr>
<td>Driver</td>
</tr>
</tbody>
</table>

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...

<table>
<thead>
<tr>
<th>K8S building blocks</th>
<th>Additional Tooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployments</td>
<td>Add-On Manager</td>
</tr>
<tr>
<td>Replicasets</td>
<td>Helm</td>
</tr>
<tr>
<td>Pods</td>
<td>Network policies</td>
</tr>
<tr>
<td>Jobs</td>
<td>Calico</td>
</tr>
<tr>
<td>Load-Balancer</td>
<td></td>
</tr>
<tr>
<td>Secrets</td>
<td></td>
</tr>
<tr>
<td>Config Maps</td>
<td></td>
</tr>
<tr>
<td>Stateful Sets</td>
<td></td>
</tr>
<tr>
<td>PVs</td>
<td></td>
</tr>
<tr>
<td>PVCs</td>
<td></td>
</tr>
<tr>
<td>CRDs</td>
<td></td>
</tr>
<tr>
<td>Controllers</td>
<td></td>
</tr>
<tr>
<td>Reconciliation</td>
<td></td>
</tr>
<tr>
<td>API Server Extension</td>
<td></td>
</tr>
<tr>
<td>RBAC</td>
<td></td>
</tr>
</tbody>
</table>

Kubernetes as deployment underlay
Following the Design Principle **Gardener** uses...

Kubernetes as deployment underlay
Following the Design Principle **Gardener** uses...

### K8S building blocks
- Deployments
- Replicsets
- Pods
- Jobs
- LoadBalancer
- Secrets
- ConfigMaps
- StatefulSets
- PVs
- PVCs
- Driver
- CRDs
- Controllers
- Reconciliation
- API Server Extension
- Admission Control
- RBAC

### Additional Tooling
- Add-On Manager
- Helm
- Network policies
- Calico
- Cert Manager
- Cert Broker
- Cluster Autoscaler
- Prometheus
- EFK Stack

---

Kubernetes as deployment underlay
Following the Design Principle **Gardener** uses...

<table>
<thead>
<tr>
<th>Workload</th>
<th>Additional Tooling</th>
<th>K8S building blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add-On Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Helm</td>
<td>Cert Manager</td>
<td>CRDs</td>
</tr>
<tr>
<td>Network policies</td>
<td>Cert Broker</td>
<td>Controllers</td>
</tr>
<tr>
<td>Calico</td>
<td>Reconciliation</td>
<td>API Server Extension</td>
</tr>
<tr>
<td>Pod</td>
<td></td>
<td>RBAC</td>
</tr>
<tr>
<td>Deployments</td>
<td>Stateful Sets</td>
<td></td>
</tr>
<tr>
<td>Replicasets</td>
<td>PVs</td>
<td></td>
</tr>
<tr>
<td>Pods</td>
<td>PVCs</td>
<td></td>
</tr>
<tr>
<td>Jobs</td>
<td>Secrets</td>
<td></td>
</tr>
<tr>
<td>Load-Balancer</td>
<td>Config Maps</td>
<td></td>
</tr>
<tr>
<td>Secrets</td>
<td>Stateful Sets</td>
<td></td>
</tr>
<tr>
<td>Config Maps</td>
<td>PVs</td>
<td></td>
</tr>
<tr>
<td>Jobs</td>
<td>Driver</td>
<td></td>
</tr>
<tr>
<td>Load-Balancer</td>
<td>PVCs</td>
<td></td>
</tr>
<tr>
<td>Secrets</td>
<td>CRDs</td>
<td></td>
</tr>
<tr>
<td>Config Maps</td>
<td>Controllers</td>
<td></td>
</tr>
<tr>
<td>Jobs</td>
<td>Reconciliation</td>
<td></td>
</tr>
<tr>
<td>Load-Balancer</td>
<td></td>
<td>Admission Control</td>
</tr>
<tr>
<td>Secrets</td>
<td></td>
<td>RBAC</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>NAME</th>
<th>INFRASTRUCTURE</th>
<th>PURPOSE</th>
<th>STATUS</th>
<th>VERSION</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-eu1</td>
<td>eu-central-1</td>
<td>INFRA</td>
<td>✔️</td>
<td>1.11.2</td>
<td></td>
</tr>
<tr>
<td>az-eu1</td>
<td>westeurope</td>
<td>INFRA</td>
<td>✔️</td>
<td>1.11.2</td>
<td></td>
</tr>
<tr>
<td>gcp-eu1</td>
<td>europe-west1</td>
<td>INFRA</td>
<td>✔️</td>
<td>1.11.2</td>
<td></td>
</tr>
<tr>
<td>os-eu-de-200-eu1</td>
<td>eu-de-200</td>
<td>INFRA</td>
<td>✔️</td>
<td>1.11.2</td>
<td></td>
</tr>
</tbody>
</table>
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
Gardener is Open Source

Long-Term Goal
Become CNCF Project
Gardener is Open Source

Long-Term Goal
Become CNCF Project
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
Gardener

What is Gardener?

A service to manage large-scale Kubernetes cluster

A bundle of Kubernetes controllers that defines and manages new API objects used for management of Kubernetes cluster

End-user clusterShoot cluster contains only worker nodes

Seed cluster contains shoot cluster’s control plane as workload

Gardener runs the gardener, a Kubernetes controller responsible for managing custom resources

CNCF officially certified!

Hybrid cloud homogeneous infrastructure

Open source

100% Kubernetes

In three components

100% Kubernetes in Kubernetes

100% Kubernetes in Kubernetes in Kubernetes!
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1  0  +1
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?

Pod ➔ Machine
ReplicaSet ➔ MachineSet
Deployment ➔ MachineDeployment
MCM Custom Resources
MCM Custom Resources

<table>
<thead>
<tr>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: test-machine</td>
</tr>
<tr>
<td>MachineClass: v1</td>
</tr>
</tbody>
</table>
Machine
Name: test-machine

Machine Class: v1

AWS-Machine-Class (Template)
Name: v1
Machine Type: t2.large
Disk Size: 50GB
Secret: test-secret
......

MCM Custom Resources
Machine
Name: test-machine
MachineClass: v1

AWS-Machine-Class (Template)
Name: v1
Machine Type: t2.large
Disk Size: 50GB
Secret: test-secret

Secret
Name: test-secret
Cloudconfig:
abc....xzy
AccessKeyId: abc123
SecretAccesskey: xyz789
MCM Custom Resources

Machine
Name: test-machine
MachineClass: v1

Machine-Set
Name: test-ms
Replicas: 3
MachineClass: v1

AWS-Machine-Class (Template)
Name: v1
Machine Type: t2.large
Disk Size: 50GB
Secret: test-secret
......

Secret
Name: test-secret
Cloudconfig:
abc....xzy
AccessKeyld: abc123
SecretAccesskey: xyz789
MCM Custom Resources

**Machine**
- Name: test-machine
- **MachineClass**: v1

**Machine-Set**
- Name: test-ms
- Replicas: 3
- **MachineClass**: v1

**Machine-Deployment**
- Name: test-md
- Replicas: 3
- UpdateStrategy: Rolling
- **MachineClass**: v1

**AWS-Machine-Class (Template)**
- Name: v1
- Machine Type: t2.large
- Disk Size: 50GB
- Secret: test-secret
- ...

**Secret**
- Name: test-secret
- Cloudconfig: abc...xzy
- AccessKeyId: abc123
- SecretAccessKey: xyz789

---

**AWS Machine Class (Template)**

- Name: v1
- Machine Type: t2.large
- Disk Size: 50GB
- Secret: test-secret
- ...

---

**Secret**

- Name: test-secret
- Cloudconfig: abc...xzy
- AccessKeyId: abc123
- SecretAccessKey: xyz789
Working of MCM

Kubernetes API Server

ETCD (Key-value store)

kubectl
Working of MCM

- Machine Class + Secret V1
- kubectl
- Kubernetes API Server
- ETCD (Key-value store)
Working of MCM

- Machine Class + Secret
- Kubernetes API Server
- ETCD (Key-value store)
- kubectl
Working of MCM

- Kubernetes API Server
- ETCD (Key-value store)
- kubectl
- Machine Class + Secret V1
Working of MCM

Kubernetes API Server

ETCD (Key-value store)

Machine Deployment
Class: V1
Replicas: 3

kubectl

Machine Class + Secret V1
Working of MCM

- **kubectl**
  - Machine Class + Secret V1
  - ETCD (Key-value store)
  - Kubernetes API Server
  - Machine Deployment
    - Class: V1
    - Replicas: 3
Working of MCM

ETCD (Key-value store)

Kubernetes API Server

Machine Deployment
Class: V1
Replicas: 3

Machine Class + Secret V1

kubectl
Working of MCM

- Kubernetes API Server
- ETCD (Key-value store)
- Machine Deployment
  - Class: V1
  - Replicas: 3
- Machine Class + Secret V1
- kubectl
Working of MCM

Kubectl

ETCD (Key-value store)

Machine Class + Secret V1

Kubernetes API Server

Machine Set
Replicas: 3

Machine Deployment
Class: V1
Replicas: 3

Machine Deployment Controller
Working of MCM

- kubectl
- Machine Class + Secret V1
- ETCD (Key-value store)
- Kubernetes API Server
- Machine Set
  - Replicas: 3
- Machine Deployment
  - Class: V1
  - Replicas: 3
- Machine Set Controller
- Machine Deployment Controller
Working of MCM

- Kubernetes API Server
- ETCD (Key-value store)
- Machine Class + Secret V1
- Machine Deployment
  - Class: V1
  - Replicas: 3
- Machine Set
  - Replicas: 3
- Machine Set Controller
- Machine Deployment Controller

`kubectl`
Working of MCM

- **ETCD** (Key-value store)
- **Kubernetes API Server**
- **kubectl**
- **Machine Deployment Controller**
- **Machine Set Controller**
- **Machine Deployment**
  - Class: V1
  - Replicas: 3
- **Machine Set**
  - Replicas: 3
- **Machine Class + Secret V1**
Working of MCM

- **ETCD** (Key-value store)
- **Kubernetes API Server**
- **kubectl**
- **Machine Deployment Controller**
- **Machine Controller**
- **Machine Set Controller**

- **Machine Class + Secret V1**
- **Machine Set**
  - Replicas: 3
- **Machine Deployment**
  - Class: V1
  - Replicas: 3
Working of MCM

**kubectl**

- **ETCD (Key-value store)**
  - **Machine Deployment**
    - Class: V1
    - Replicas: 3
  - **Machine Set**
    - Replicas: 3
  - **Machine**
    - From Kubernetes API Server

- **Machine Controller**
  - From **Machine**
  - From **Machine Set Controller**

- **Machine Class + Secret V1**
Working of MCM

- **ETCD** (Key-value store)
- **Kubernetes API Server**
- **Machine Deployment Controller**
  - Machine: Class: V1, Replicas: 3
  - Machine Set: Replicas: 3
- **Machine Controller**
- **Cloud Provider API**
- **Machine Class + Secret V1**
- **kubectl**
Working of MCM

- **ETCD** (Key-value store)
- Kubernetes API Server
- Machine
- Machine
- Machine
- Machine Set
- Replicas: 3
- Machine Deployment
- Class: V1
  - Replicas: 3
- Machine Class + Secret V1
- Cloud Provider API
- 3 VMs
- Machine Controller
- Machine Set Controller
- Machine Deployment Controller
- `kubectl`

```
```
Working of MCM

kubectl

ETCD (Key-value store)

Machine Class + Secret V1

Machine Deployment
Class: V1
Replicas: 3

Machine Set
Replicas: 3

Machine Controller

Machine Deployment Controller

Cloud Provider API

3 VMs

Kubernetes API Server

Machine

Machine Set Controller
Working of MCM

Kubernetes Controller Manager

Cloud Provider API

3 VMs

kubectl

Machine

Machine

Machine

Machine Controller

Machine Set Controller

Machine Deployment Controller

Kubernetes API Server

ETCD (Key-value store)

Machine Class + Secret V1

Machine Set

Replicas: 3

Machine Deployment

Class: V1

Replicas: 3

Replicas: 3

3 VMs
Working of MCM

- Kubernetes Controller Manager
- ETCD (Key-value store)
- Kubernetes API Server
- Machine Class + Secret V1
- Machine
- Machine Deployment
- Machine Set
- Replicas: 3
- Node
- 3 VMs
- kubectl

Cloud Provider API

Machine Controller

Machine Set Controller

Machine Deployment Controller
Working of MCM

Node objects help in monitoring the machine status – Health

ETCD (Key-value store)

- Machine Class + Secret V1
- Machine Deployment
  - Class: V1
  - Replicas: 3

Cloud Provider API

- Kubernetes Controller Manager
- Machine Controller
- Machine Set Controller
- Machine Deployment Controller

3 VMs
Working of MCM

Node objects help in monitoring the machine status – Health

- Node
- Node
- Node

Kubernetes API Server

- Machine
- Machine
- Machine

Machine Class + Secret V1

ETCD (Key-value store)

- Machine Set
  - Replicas: 3

- Machine Deployment
  - Class: V1
  - Replicas: 3

3 VMs

kubectl
Autoscaling

Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager
  - Machine Controller
  - Machine Set Controller
  - Machine Deployment Controller

ETCD (Key-value store)

Machine Deployment
  - Class: V1
  - Replicas: 3

Kubernetes API Server

Node 1 → Machine
Node 2 → Machine
Node 3 → Machine

kubectl
Autoscaling

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

Pod
Image: Nginx
Node: -

kubectl

Pod
Image: Nginx
Node: -

Machine
Deployment
Controller

Machine
Set
Controller

Machine
Controller
Manager

Cloud
Provider API

ETCD
(Key-value store)

Replicas: 3

Class: V1

Kubernetes
Controller
Manager

Kubernetes
API Server

Node 1

Node 2

Node 3
Autoscaling

Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager

Machine Controller

Machine Set Controller

Machine Deployment Controller

Node 1

Node 2

Node 3

Machine

Machine

Machine

Kubernetes API Server

ETCD (Key-value store)

Pod

Image: Nginx

Node: Unschedulable

Machine Deployment

Class: V1

Replicas: 3

kubectl
Autoscaling

Forked Cluster Autoscaler

kubectl

Machine Deployment
Class: V1
Replicas: 3

Node 1
Node 2
Node 3

Machine
Machine
Machine

Forked Cluster Autoscaler

Kubernetes API Server

ETCD (Key-value store)

Image: Nginx
Node: Unschedulable
Autoscaling

Forked Cluster Autoscaler

kubectl

Machine Deployment
Class: V1
Replicas: 3

Node 1
Node 2
Node 3

Machine
Machine
Machine

Kubernetes API Server

ETCD (Key-value store)

Pod
Image: Nginx
Node: Unschedulable

Machine Controller Manager

Machine Controller
Machine Set Controller
Machine Deployment Controller

Cloud Provider API

Kubernetes Controller Manager

Autoscaler

ETCD (Key-value store)
Autoscaling

Forked Cluster Autoscaler

kubectl

Pod
- Image: Nginx
- Node: Unschedulable

Machine Deployment
- Class: V1
- Replicas: 3

ETCD (Key-value store)

Kubernetes Controller Manager

Cloud Provider API

AWS

GCP

Azure

OpenStack

Alibaba Cloud

Node 1

Node 2

Node 3
Autoscaling

Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager
  - Machine Controller
  - Machine Set Controller
  - Machine Deployment Controller

Forked Cluster Autoscaler

Node 1
Node 2
Node 3

Kubernetes API Server

Machine

ETCD (Key-value store)

Pod
Image: Nginx
Node: Unschedulable

Machine Deployment
Class: V1
Replicas: 4

kubectl
Autoscaling

Kubernetes Controller Manager

Cloud Provider API

Machine Controller Manager

Machine Set Controller

Machine Deployment Controller

Node 1
Node 2
Node 3
Node 4

Machine
Machine
Machine
Machine

Forked Cluster Autoscaler

Pod
Image: Nginx
Node: Node4

ETCD (Key-value store)

Kubernetes API Server

Machine Deployment
Class: V1
Replicas: 4

kubectl
Autoscaling

Forked Cluster Autoscaler

kubectl

Pod
Image: Nginx
Node: Node4

ETCD (Key-value store)

Machine Deployment
Class: V1
Replicas: 4

Cloud Provider API

Machine Controller Manager
Machine Controller
Machine Set Controller
Machine Deployment Controller

Node 1
Node 2
Node 3
Node 4

Machine
Machine
Machine
Machine
Machine Controller Manager - Components

- **Machine Controller**: Responsible for Managing Machines
- **Machine-Set Controller**: Responsible for Maintaining set of healthy Machine replicas
- **Machine-Deployment Controller**: Responsible for Managing Machine-sets (used for updates)
- **Cluster Auto-scaler**: Scales the number of replicas based on load in the cluster

Create/delete Machines to maintain required replicas
Create/update Machine-sets to perform updates
Update no. of replicas based on load

Parent-child relationship: Adoption of orphaned children
Controllers cooperate, rather than racing with each other!