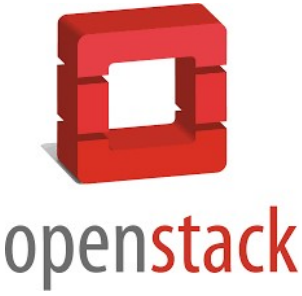
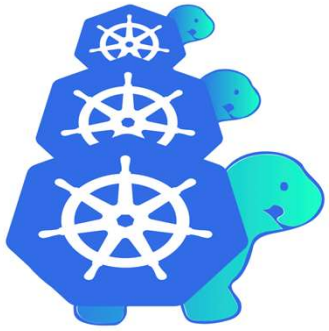
Bootstrapping a Kubernetes Cluster on Openstack with Cluster API





서한배
hanbae.seo@samsung.com
jazzsir@gmail.com

Kubernetes Cluster: Installation?

- **Infrastructure**
  - **Networking**
  - **Firewall Rules**
  - **Servers**
  - **Load balancer**
- Kubernetes components
- Add-ons
  - Ingress
  - Registry
  - Conformance
  - Logging / Monitoring tools

Cluster API?

- Declarative lifecycle management of Kubernetes clusters
- Non-goal
  - To manage a single cluster spanning multiple infrastructure providers.
- v1alpha2 release September 2019 (AWS, vSphere)
- Divided management functions into 2 parts.
  - Basic Cluster API
    - Framework used for all providers and maintained independently.
    - Controllers which reconcile resources, **`clusterctl`** CLI implementation.
  - Cluster API Provider (AWS, vSphere, GCP, Openstack, Azure, IBMcloud)
    - Implements operation details of different infrastructure
    - Cluster API abstracts overall management functions of a cluster

Cluster API Basics (v1alpha1)

- `clusterctl`
  - Generic CLI tool for the project
  - Each cloud provider forks
- Deploys CRD objects to some kubernetes cluster
  - Cluster{}
  - Machine{}
  - MachineSet{}, MachineDeployment{}
- Deploys controller that reads cluster api CRD objects
  - Cluster API controller: controller of cluster management functions for related resources
  - Provider controller: controller of provider resources for cluster and machine provisions

Controller?

- Takes care of routine tasks to ensure the desired state of the cluster matches the observed state.

- Examples
  - ReplicationController maintains a correct number of pods running in the cluster.
  - Node Controller looks up the state of servers and responds when servers go down.

Prerequisites

- Install kubectl
- Install minikube or use existing Kubernetes cluster / Kind
- Build `clusterctl` command
- Prepare clouds.yaml

```
clouds:
  openstack:
    auth:
      auth_url: http://10.8.8.200:35357/v3
      username: "kubernetes"
      password: "clusterapi"
      project_id: 58b57f6935b5457fb54ab011a18c6101
      domain_name: "Default"
      user_domain_name: "Default"
    region_name: "RegionOne"
    interface: "internal"
    identity_api_version: 3
```

Openstack Provider Versions

- Versions

- Compatible versions

	OpenStack Provider v1alpha1 (ea309e7f)
Cluster API	v1alpha1 (v0.1)
Kubernetes	1.12.4+
OpenStack	Pike, Queens, Rocky, **Stein**

Creating a cluster

- Using Kind

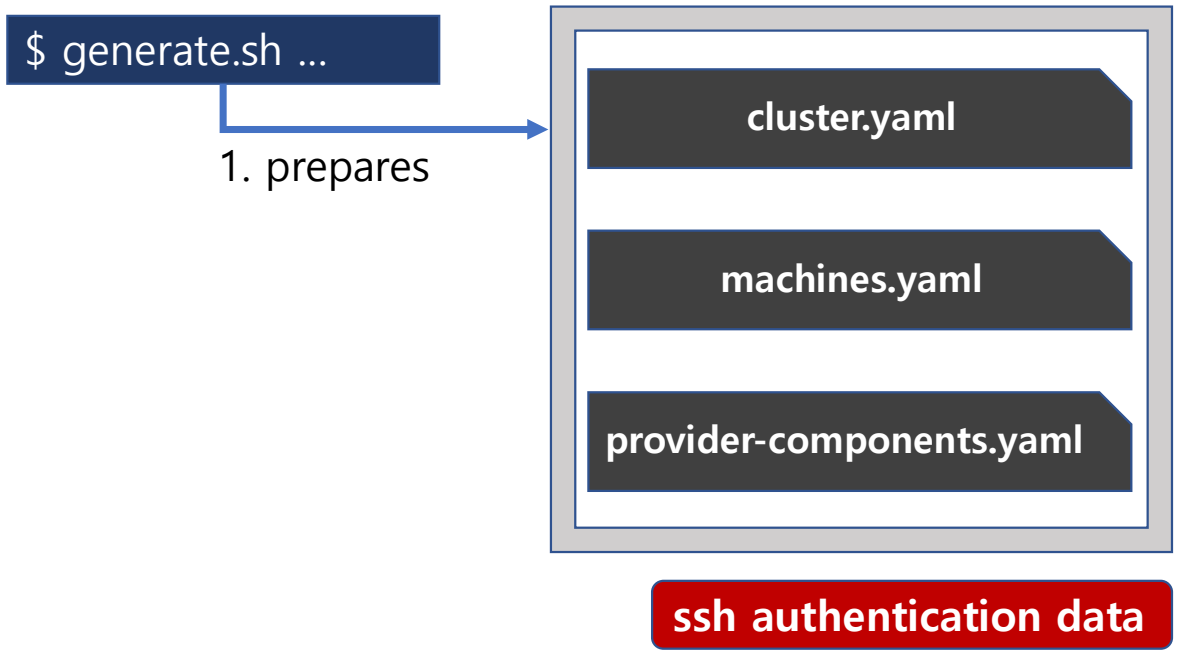
```
clusterctl create cluster ₩
        --bootstrap-type kind --bootstrap-cluster-cleanup=false ₩
        --provider openstack ₩
        -c examples/out/cluster.yaml ₩
        -m examples/out/machines.yaml ₩
        -p examples/out/provider-components.yaml ₩
```

- Using an existing Kubernetes cluster

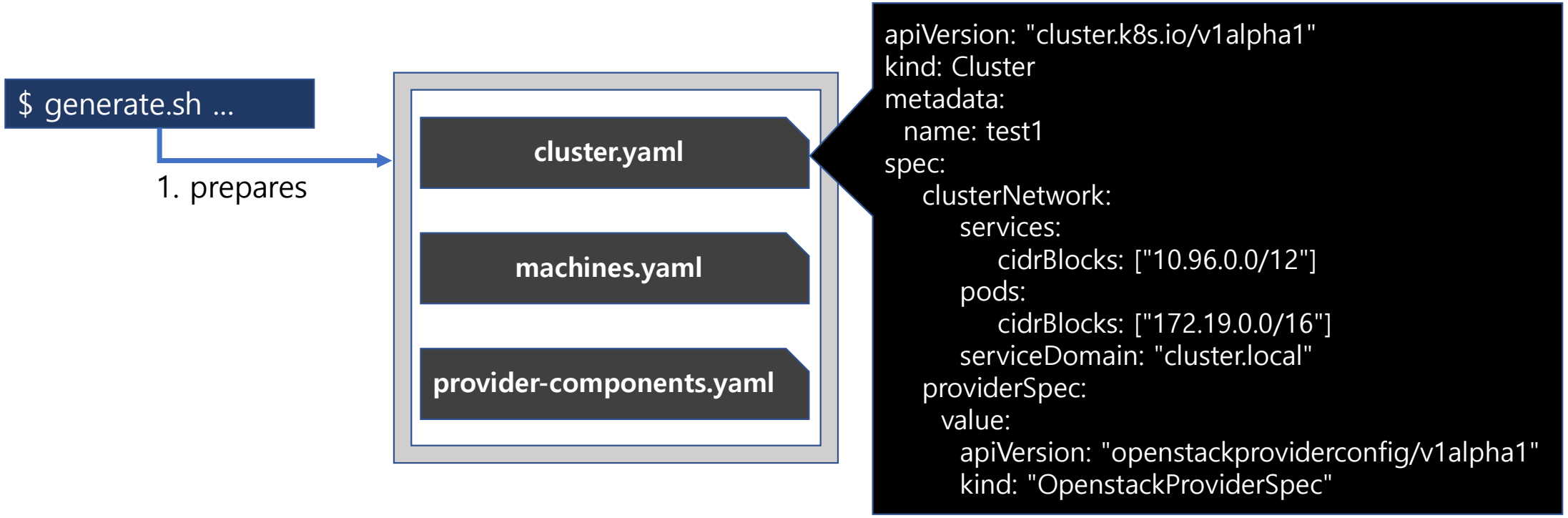
```
clusterctl create cluster ₩
        --bootstrap-cluster-kubeconfig ~/.kube/config ₩
        --provider openstack ₩
        -c examples/out/cluster.yaml ₩
        -m examples/out/machines.yaml ₩
        -p examples/out/provider-components.yaml
```

*** Start a Demo**

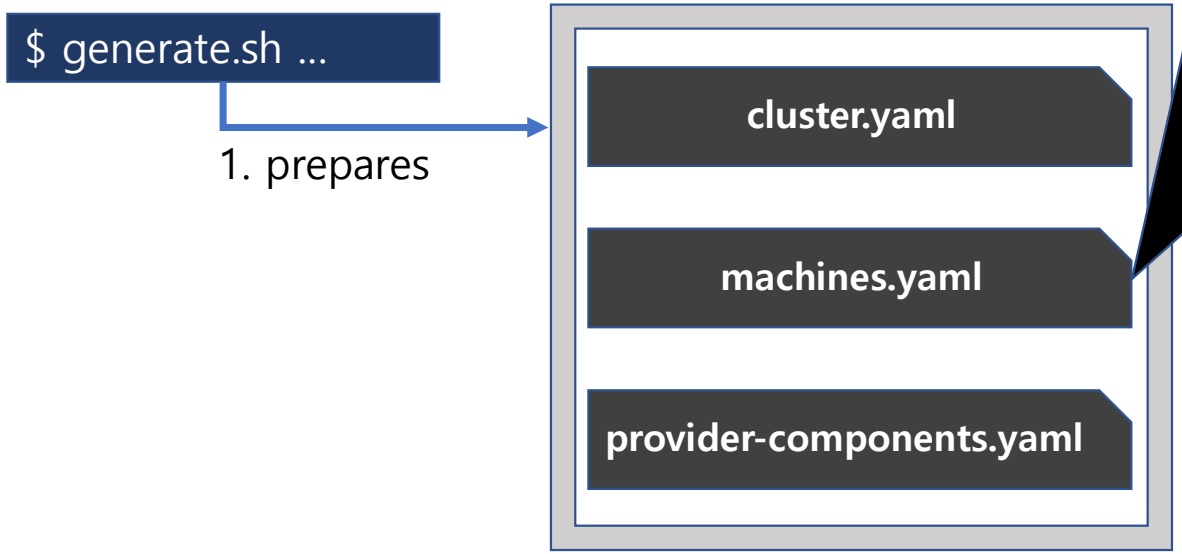
How it works in detail?



How it works in detail?

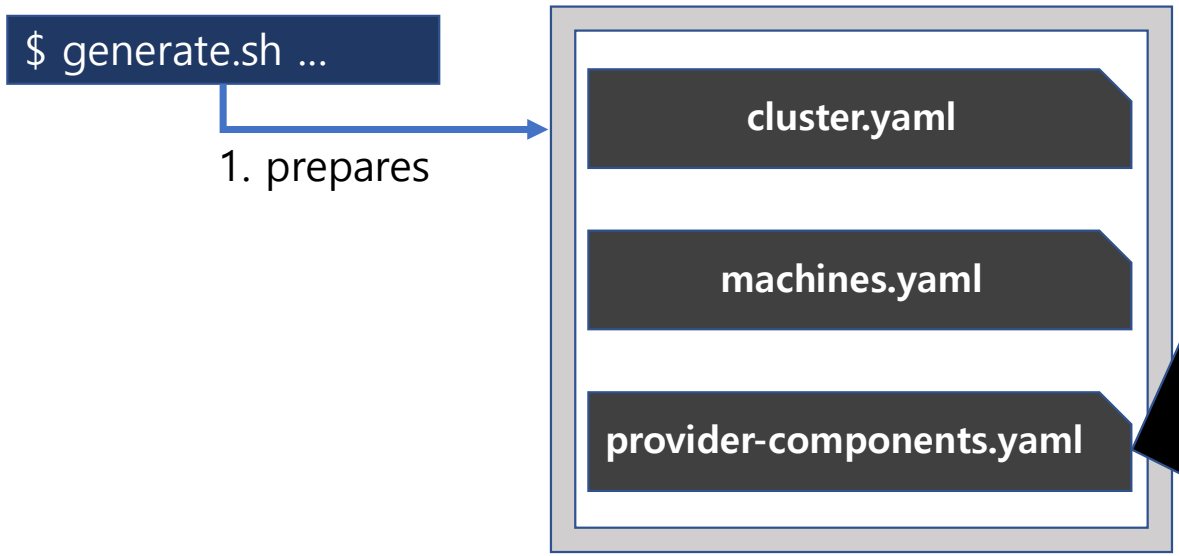


How it works in detail?



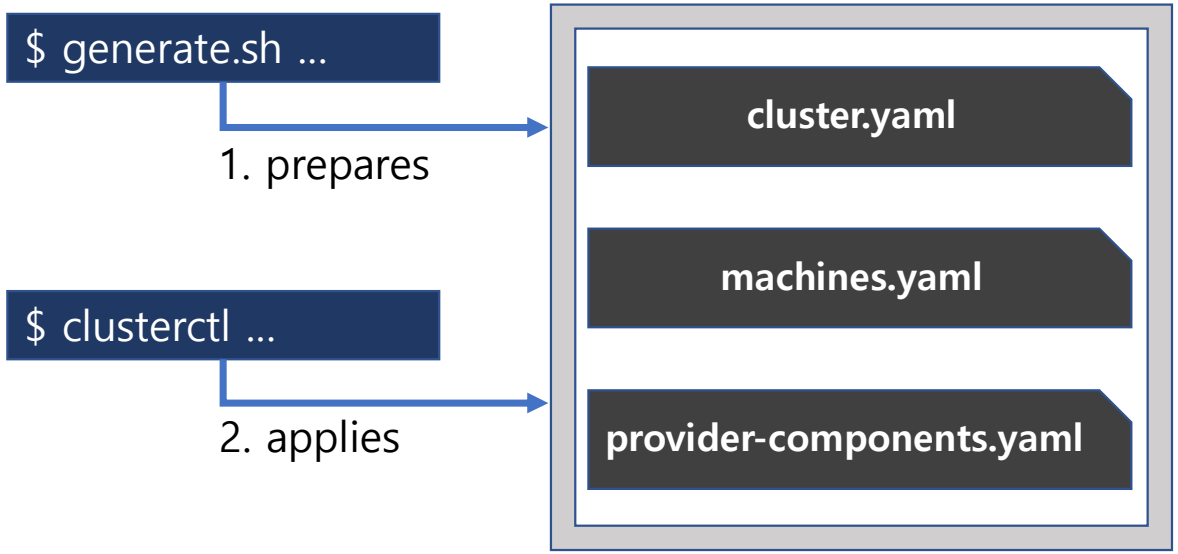

```
- apiVersion: "cluster.k8s.io/v1alpha1"
  kind: Machine
  ....
  spec:
    providerSpec:
      value:
        kind: "OpenstackProviderSpec"
        image: <Image Name>
        sshUserName: <SSH Username>
        networks:
        - uuid: <Kubernetes Network ID>
        floatingIP: <Available Floating IP>
        .....
    versions:
      kubelet: 1.12.3
      controlPlane: 1.12.3
- apiVersion: "cluster.k8s.io/v1alpha1"
  kind: Machine
  .....
    versions:
      kubelet: 1.12.3
- apiVersion: "cluster.k8s.io/v1alpha1"
  kind: Machine
  .....
    versions:
       kubelet: 1.12.3
```

How it works in detail?

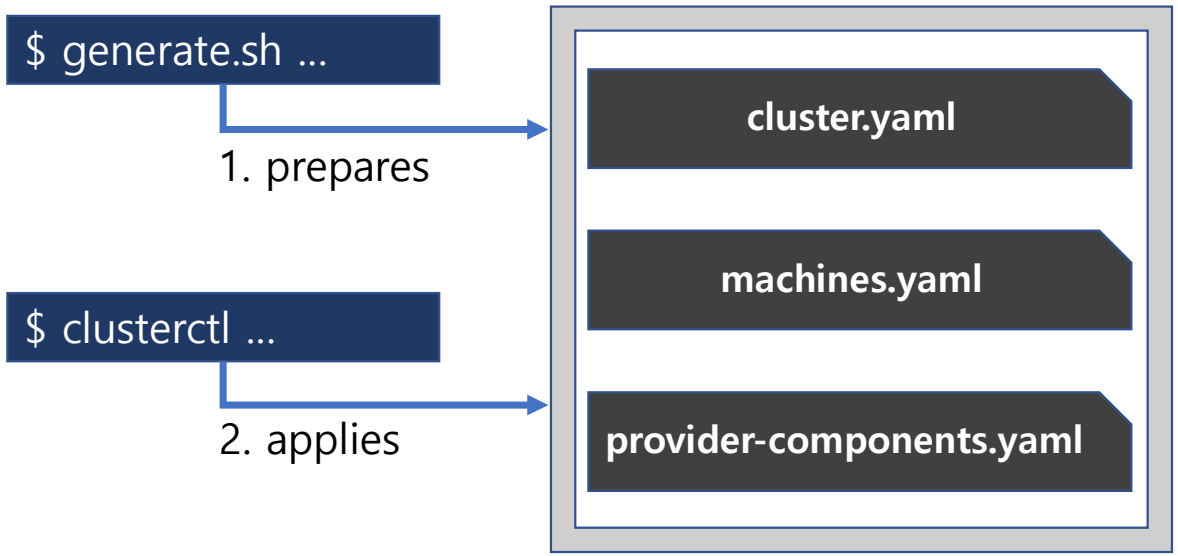



```
kind: CustomResourceDefinition
  name: clusters.cluster.k8s.io
  name: machines.cluster.k8s.io
  name: machinesets.cluster.k8s.io
  name: machineclasses.cluster.k8s.io
  name: machinedeployments.cluster.k8s.io
  name: openstackclusterproviderspecs....k8s.io
  name: openstackclusterproviderstatuses....k8s.io
  name: openstackproviderspecs....k8s.io
  ....

kind: StatefulSet
  name: controller-manager

kind: Deployment
  name: clusterapi-controllers

kind: Secret
  name: cloud-config
  name: master-user-data
  name: worker-user-data
```

How it works in detail?

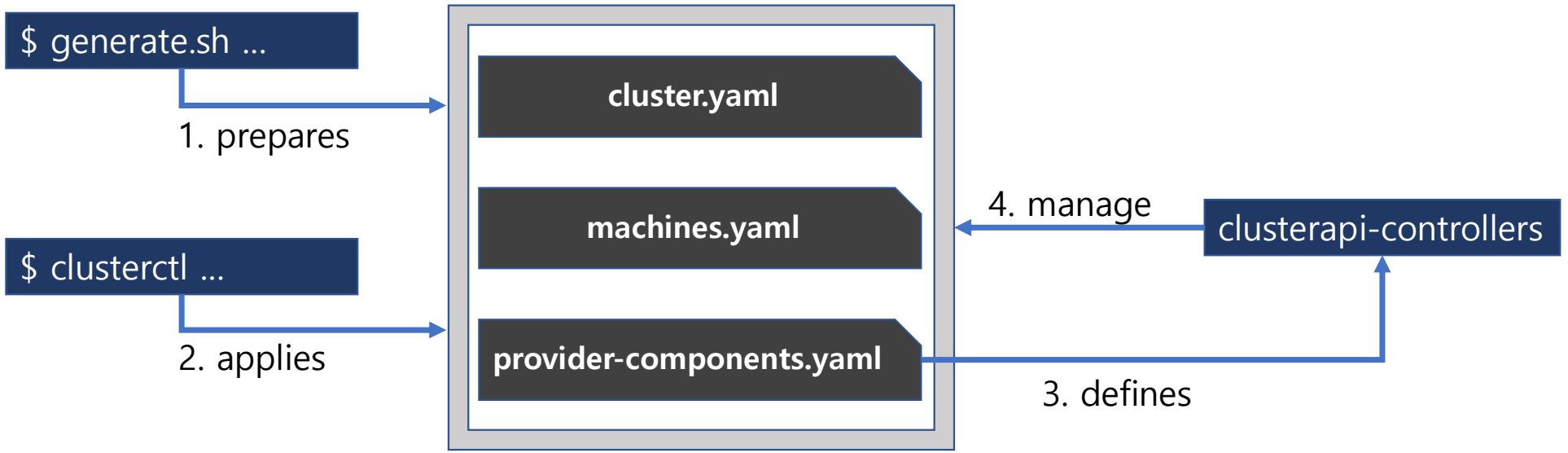


How it works in detail?

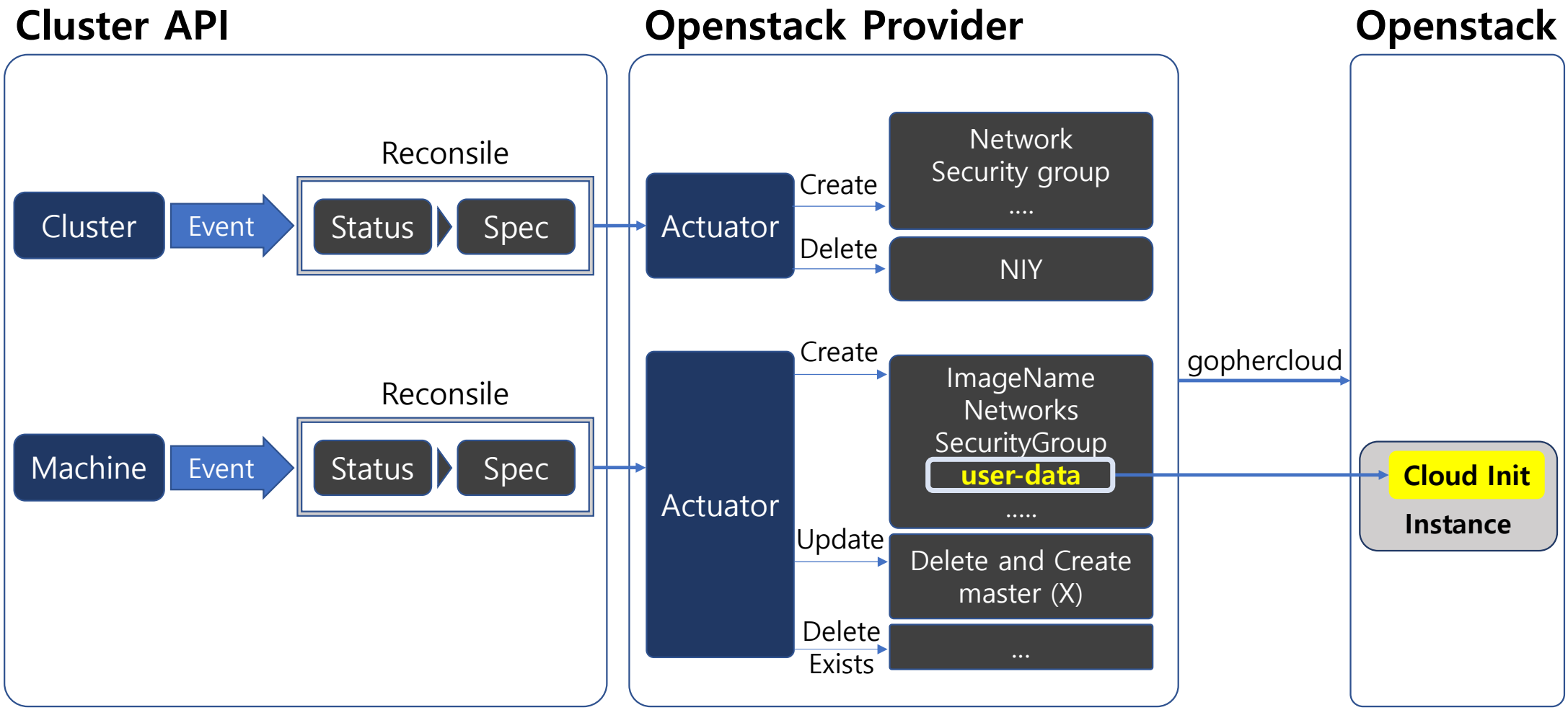


1. Get a control plane machine
2. If Kind, create a bootstrap cluster
3. Apply provider-components.yaml, cluster.yaml and machine.yaml to bootstrap cluster (**deploy clusterapi-controllers)**
4. Wait until a target cluster client is created.
5. Apply add-ons to the target cluster.
6. Apply cluster API components to target cluster except for worker machines. (pivoting)
7. Apply worker machines to target cluster

How it works in detail?



Cluster API controllers

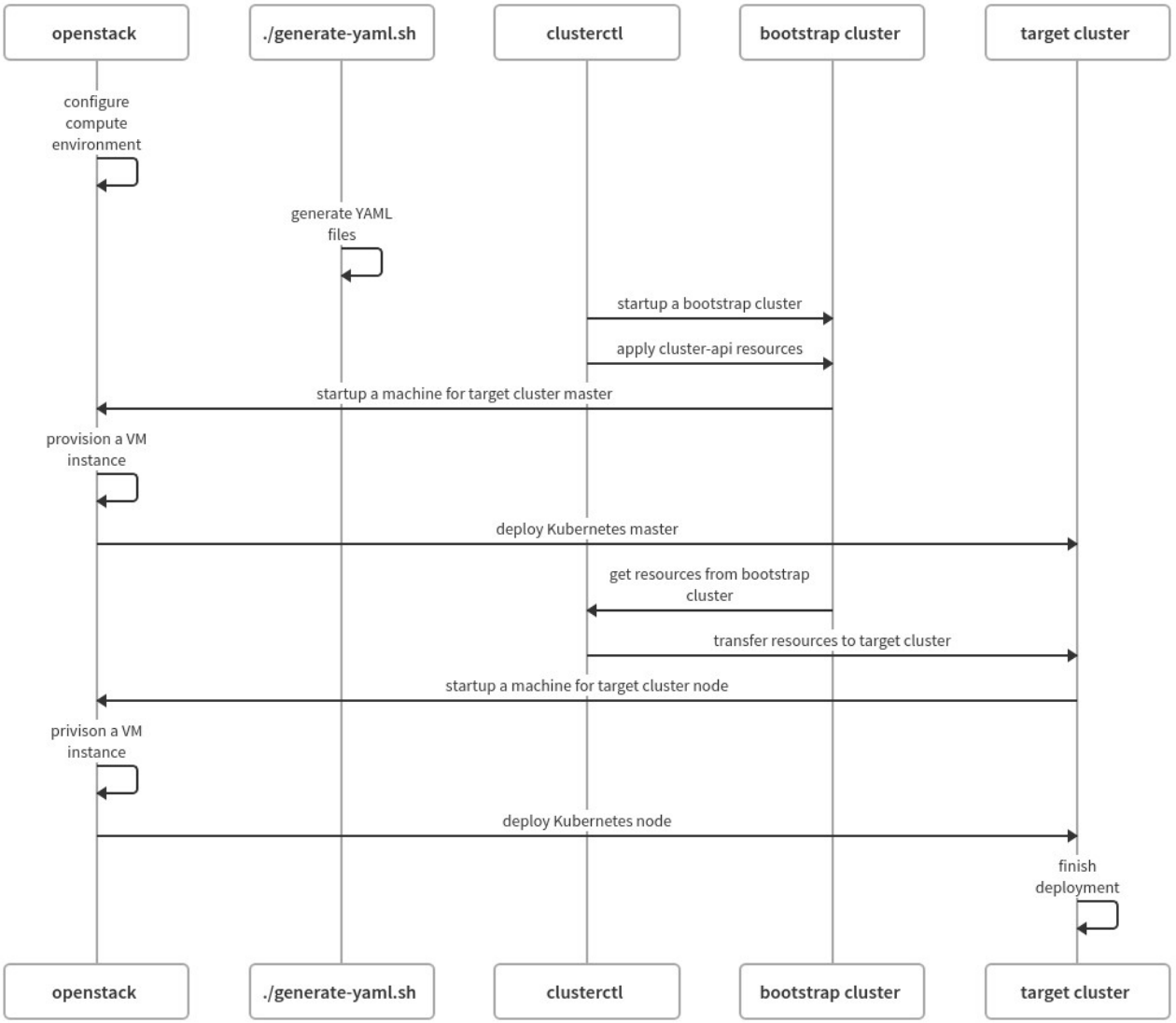


Demo

Customizing

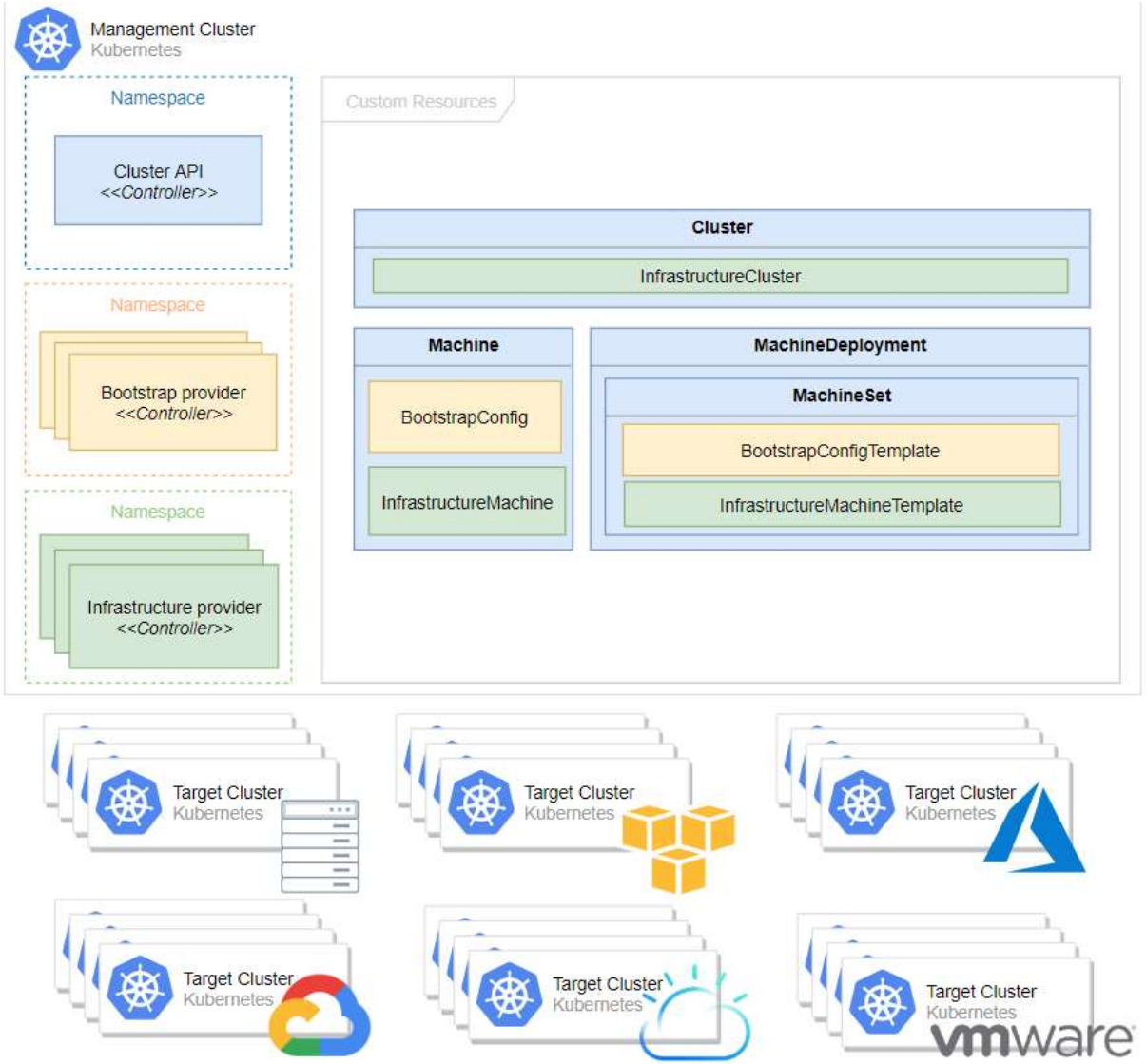
- Provider-components
  - Package versions
  - Kubeadm config
    - InitConfiguration, ClusterConfiguration, KubeletConfiguration, JoinConfiguration
  - CNI plugin
  - Port 6443
- Provider controllers
  - Apply a specific subnet
  - Port 6443
- Deploy a registry for provider controllers

Copy of Cluster API for Openstack



v1alpha2

- Managers
  - Core (Cluster API)
  - **Bootstrap (kubeadm) for user-data**
  - Infrastructure (aws, vsphere, etc)
- Independent controllers
  - Actuators are not used



Deploying Openstack

- Tool: Kolla-ansible
- Release: Stein
- OS: CentOS 7.6
- Multinodes deployment
  - Controller node.
    - [Notebook] Intel I5 Quad core / 16G
  - Compute/Network node.
    - [Notebook] Intel I7 Quad core / 16G

Thank you

Terms

- Target cluster
  - The declared cluster we intend to create and manage
- Bootstrap/Management cluster
  - The cluster that manages the target cluster
  - Possibly the same cluster
- cluserctl
  - Community CLI tool that favors a provider implementation for creating and managing a cluster
- Provider implementation
  - An implementation of the API specific to a cloud (Openstack, VMware, AWS, etc)