KubeCon | CloudNativeCon

North America 2019

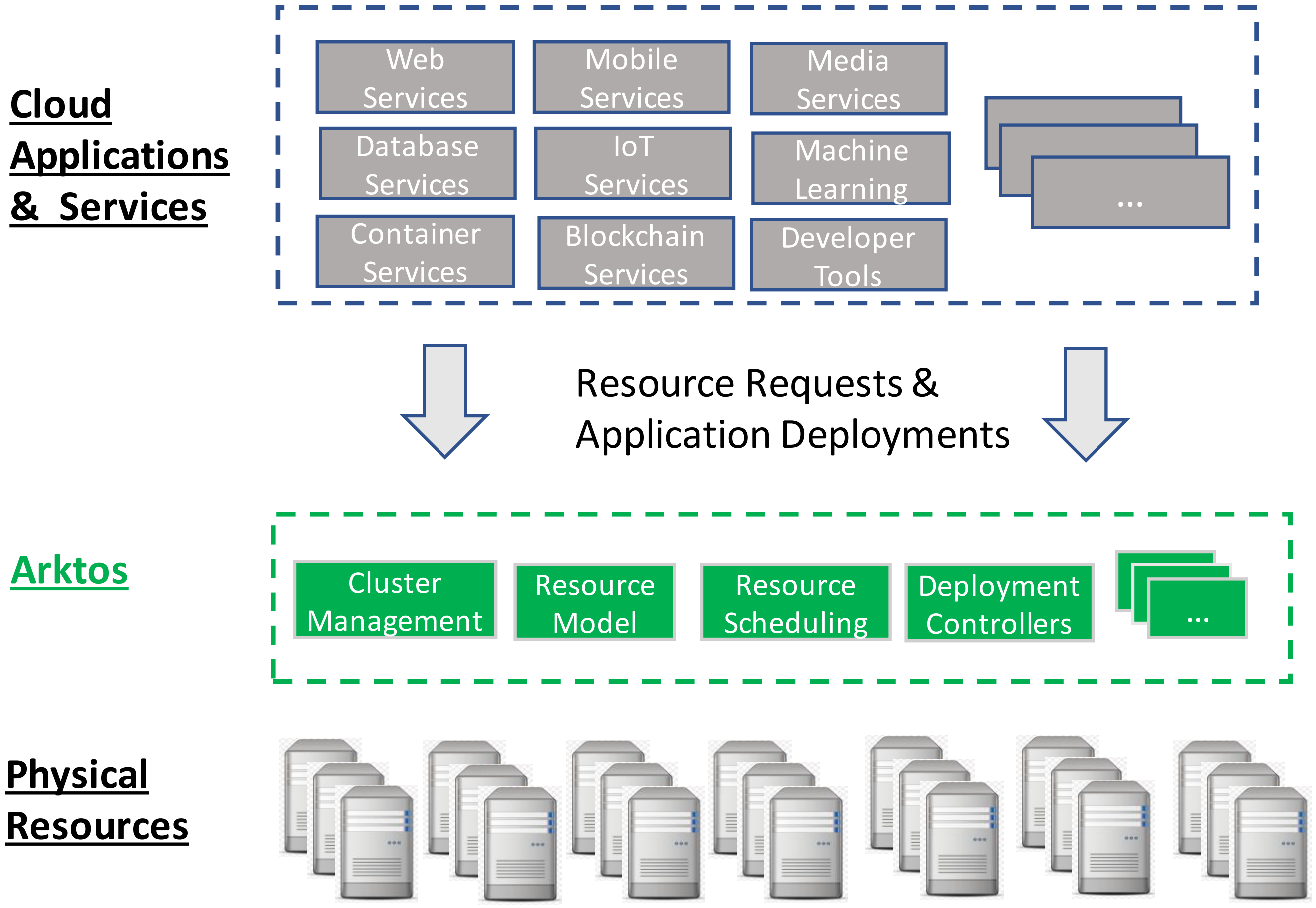
Project Arktos

Cloud Lab, Futurewei Technologies

Agenda

- Arktos Overview
- Key Features
  - Multi Tenancy
  - Large Scalability
  - Unified VM/Container Stack
- Future Plan

Arktos Overview



Hard Multi-tenancy

Built-in hard multi-tenancy model, providing a strong isolation among tenant resources.

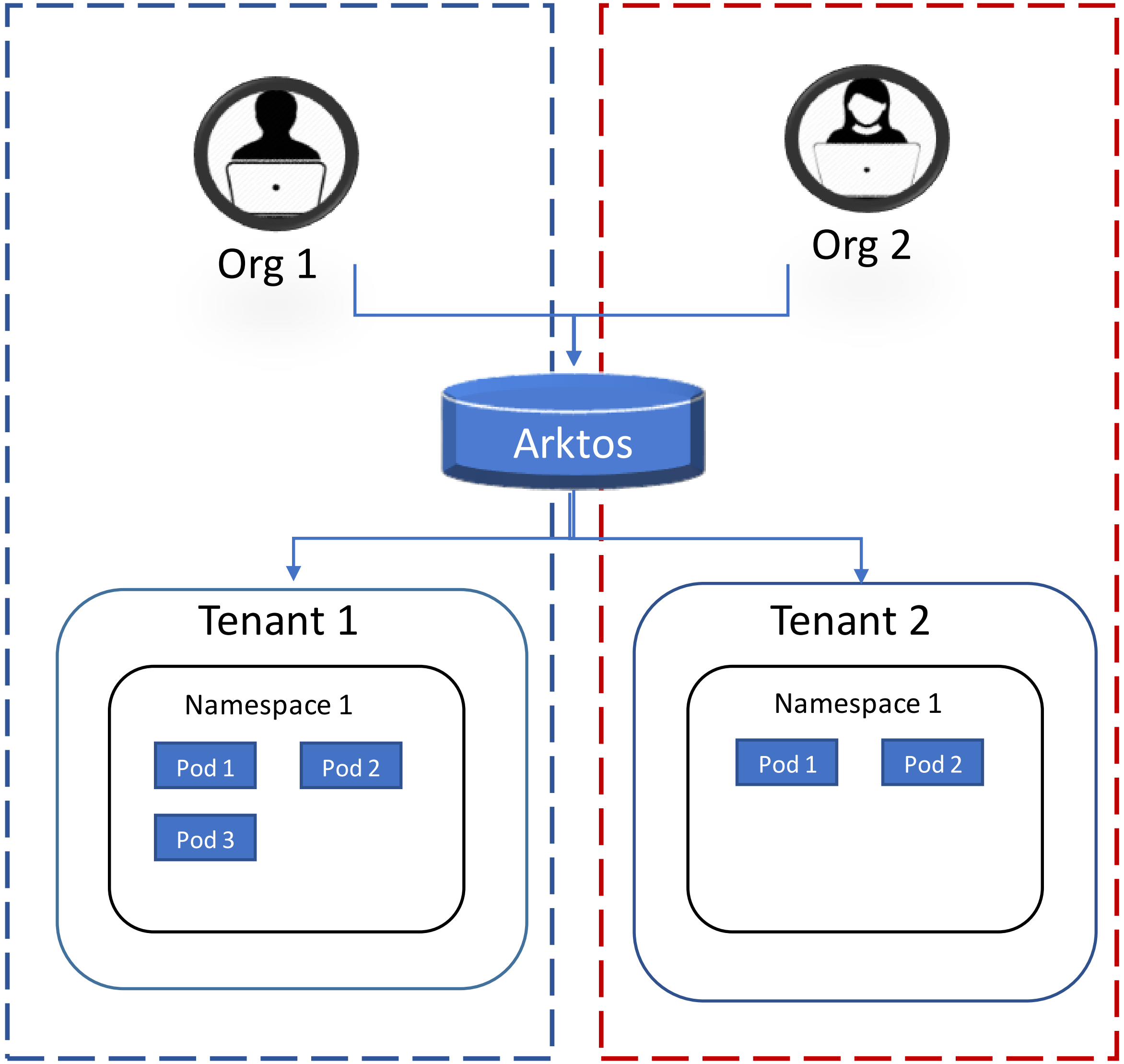
Cloud Scale

Designed to support 100K nodes per cluster. Partitioned and replicated storage, scheduler and controllers.

Unified Stack

One single unified stack for containers, VMs and bare metals, including API models, scheduling, runtime, etc.

Multi-Tenancy



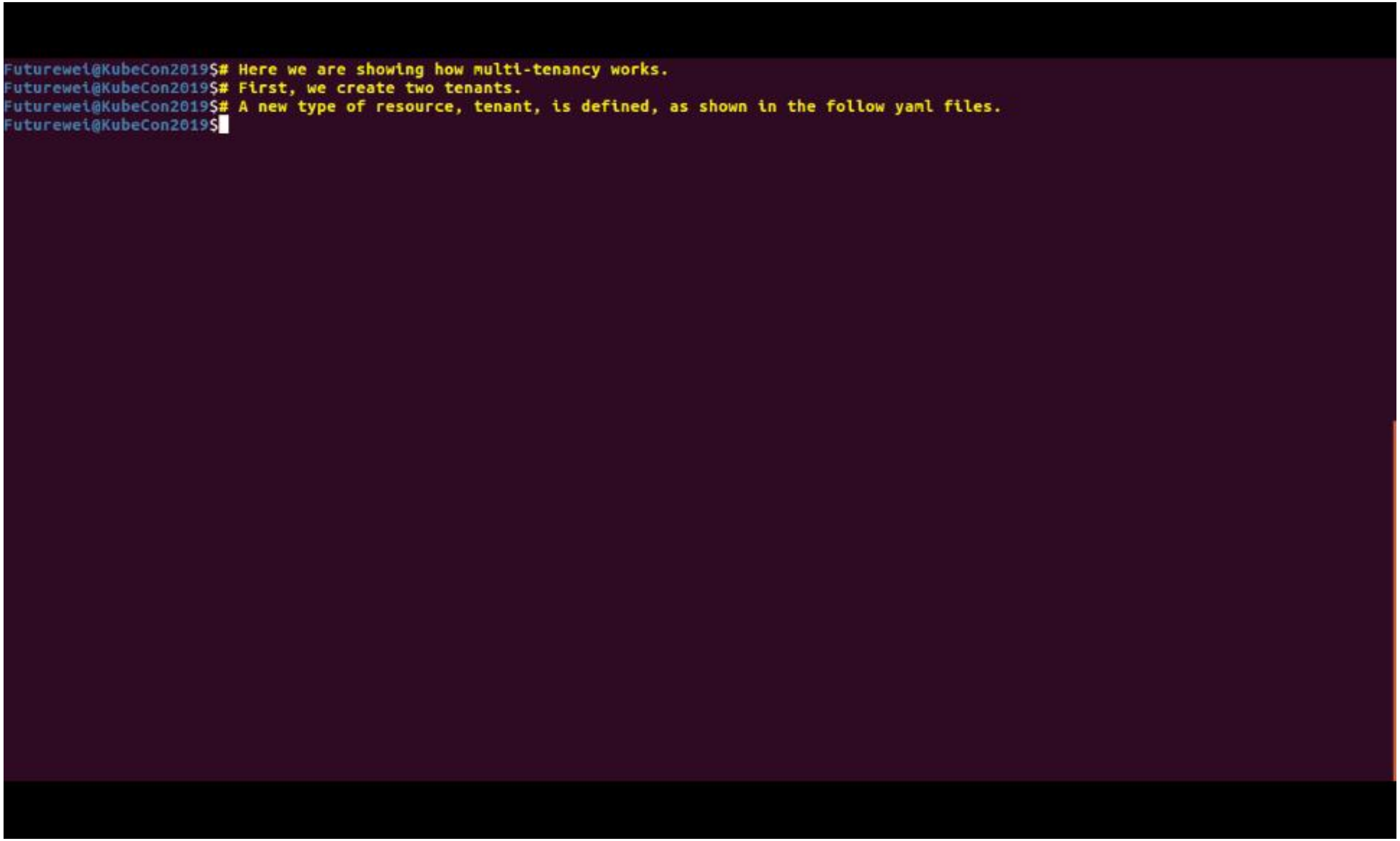
Hard Multi-Tenancy

- Enable organizations/departments to safely share one infrastructure, without deploying/operating multiple clusters.
- Support per-tenant resource view, access control, quota, etc.
- Assume no trust among tenants; ready for strict scenarios like public cloud.

Key Changes:

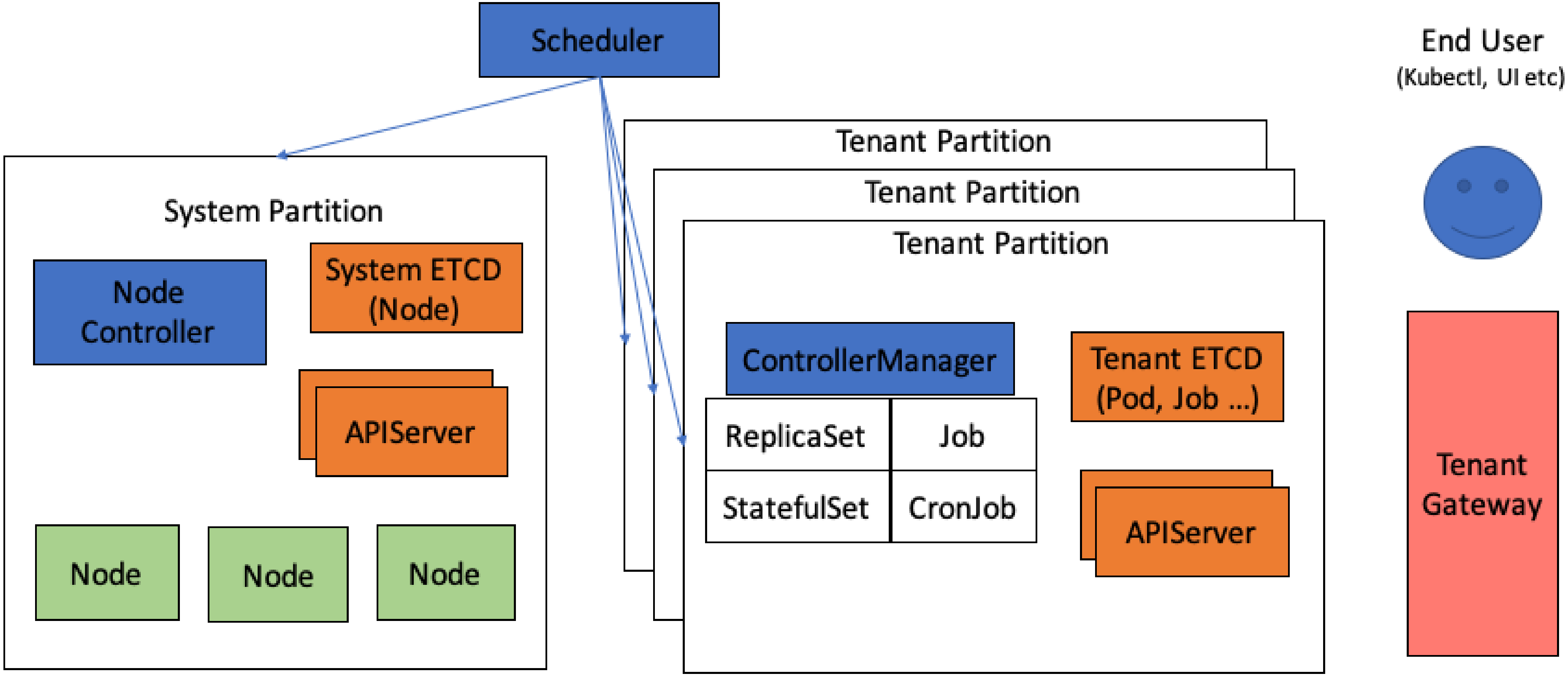
- A new API object: ***tenant***
- All API objects have a new field ***Tenant*** in its ***ObjectMeta*** section
- A new resource URL scheme: ***tenants/{tenant}/namespaces/{namespace}/{objectTypes}/{objectName}***
- Tenant-aware Client-Go library, scheduler, controllers, agent and CLI tools.

Demo: Multi-Tenancy



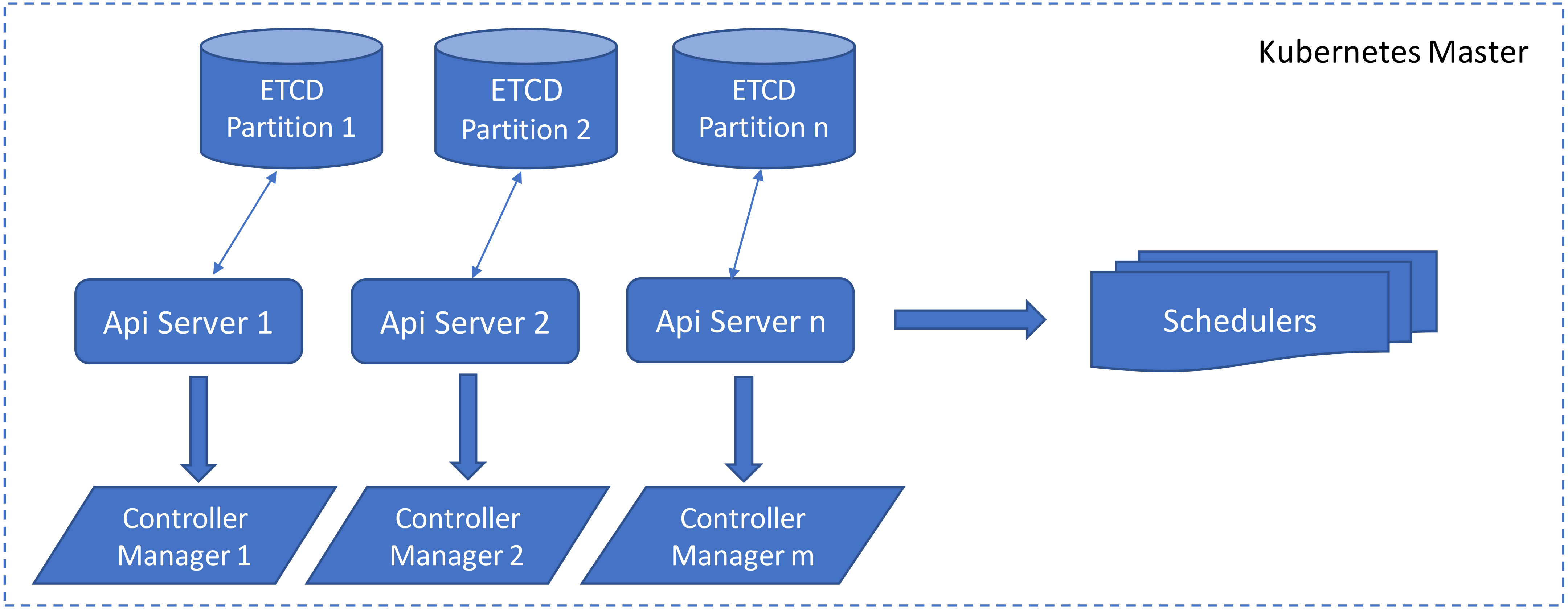
```
Futurewei@KubeCon2019$# Here we are showing how multi-tenancy works.
Futurewei@KubeCon2019$# First, we create two tenants.
Futurewei@KubeCon2019$# A new type of resource, tenant, is defined, as shown in the follow yaml files.
Futurewei@KubeCon2019$
```

Scalability Architecture



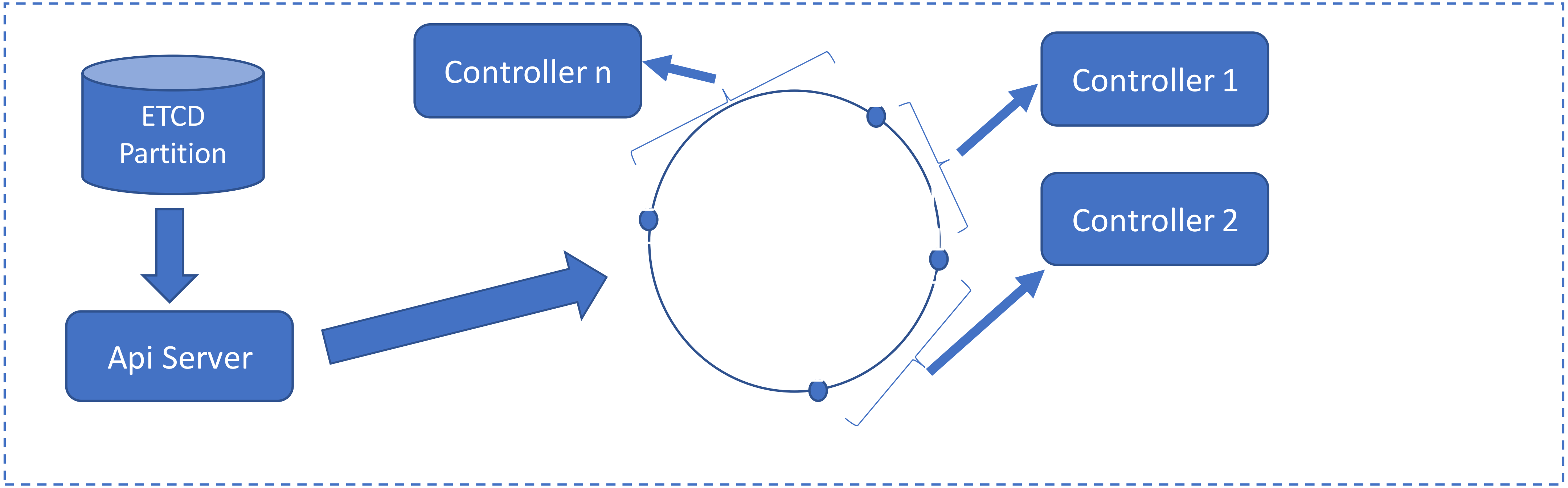
- Shard tenant data
- Scheduler has global view of all nodes in cluster

Scalability Architecture



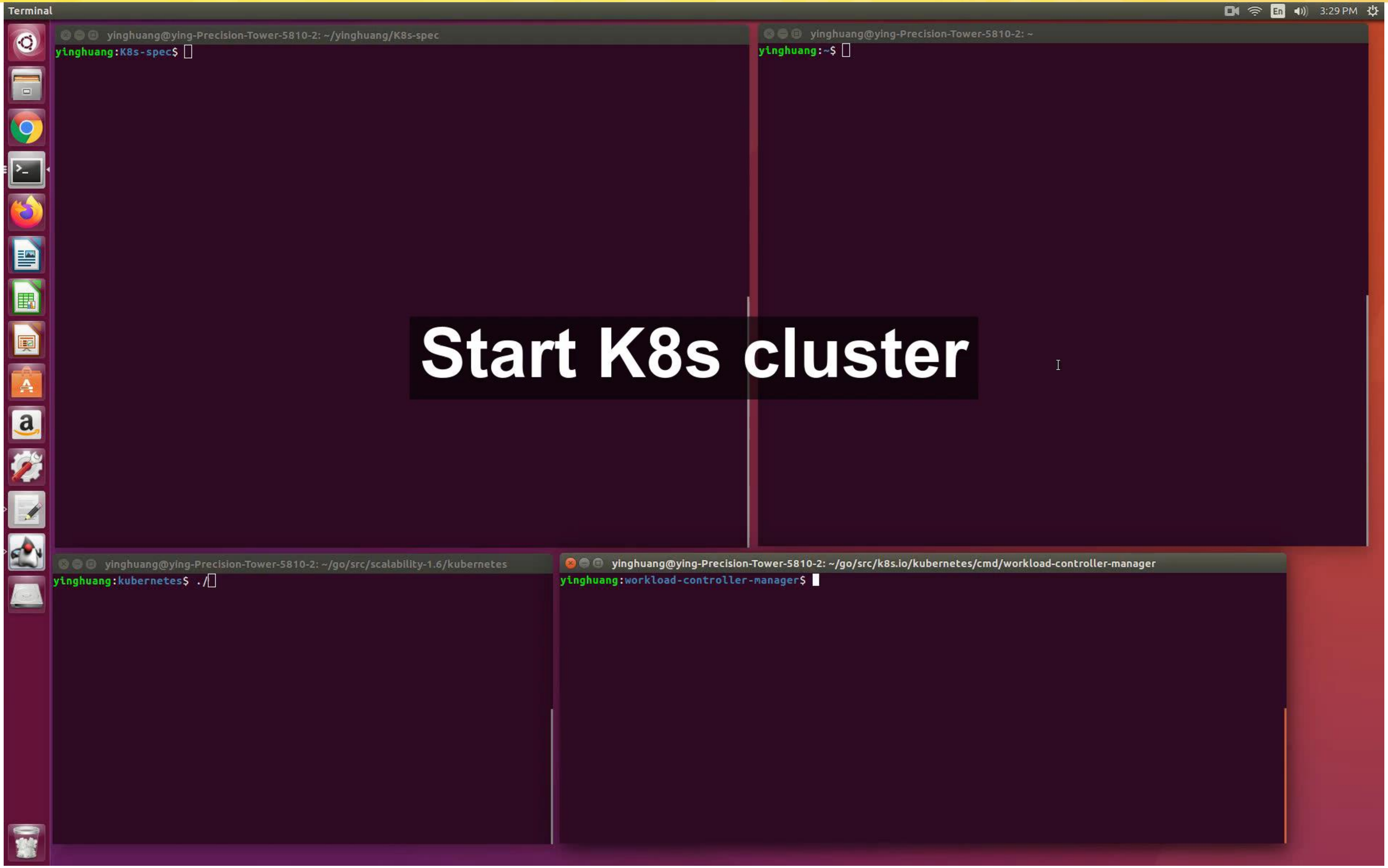
- One ETCD cluster gets partitioned (based on tenant and namespace)
- One API Server list-watch one partition to reduce cache footprint
- Any API server can handle write requests to any partition
- Any API server can handle non-list-watch read requests to any partition

Scalable Controllers



- List/watch by range of field value
- Multiple controller instances
  - Multiple controller managers works in active-active mode

Demo: Scalable Controllers



Terminal
3:29 PM
yinghuang@ying-Precision-Tower-5810-2: ~/yinghuang/K8s-spec
yinghuang@ying-Precision-Tower-5810-2: ~
yinghuang@ying-Precision-Tower-5810-2: ~/go/src/scalability-1.6/kubernetes
yinghuang@ying-Precision-Tower-5810-2: ~/go/src/k8s.io/kubernetes/cmd/workload-controller-manager

```
yinghuang:K8s-spec$
```

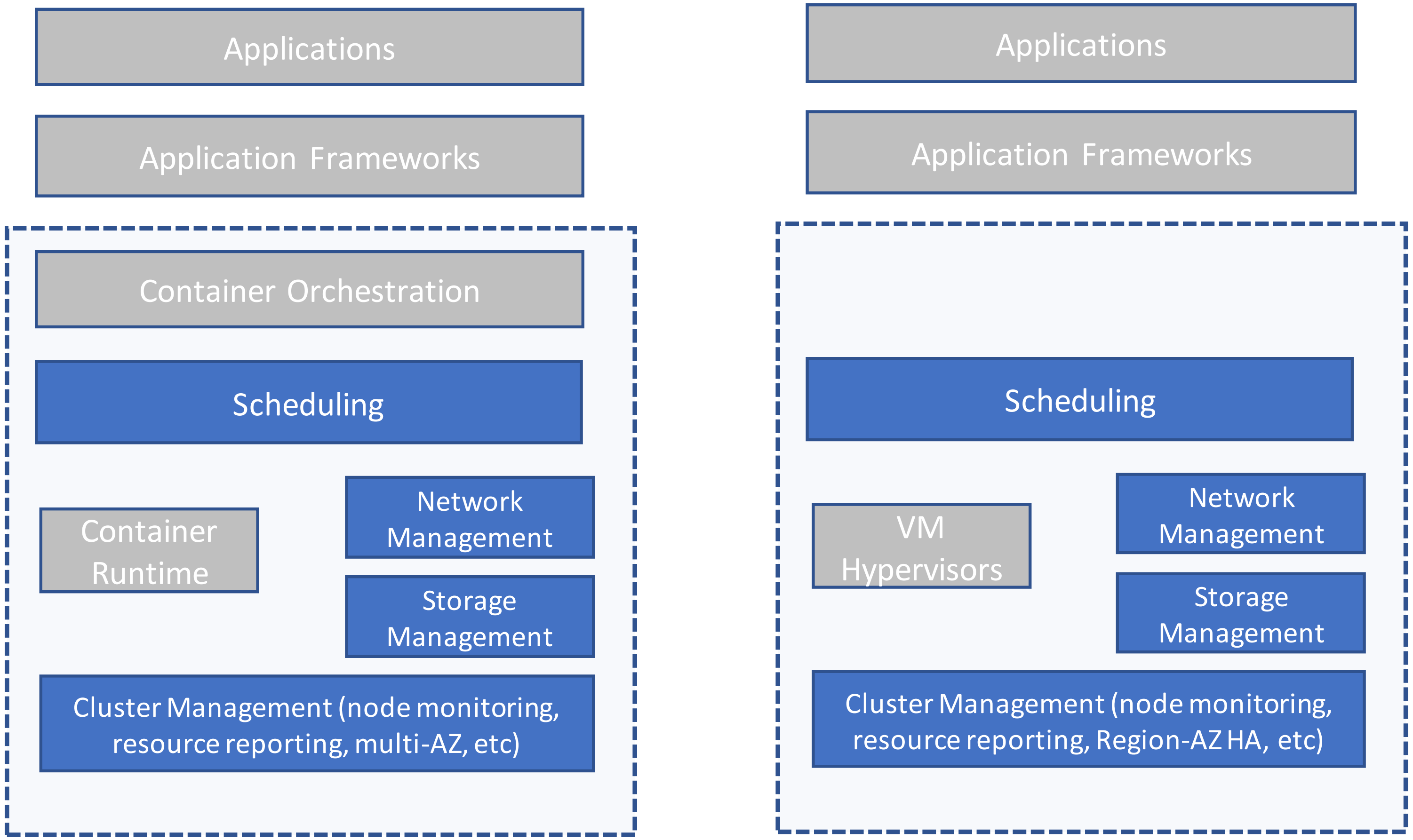
```
yinghuang:~$
```

```
yinghuang:kubernetes$ ./
```

```
yinghuang:workload-controller-manager$
```

Start K8s cluster

Two Stacks in Today's Data Center



Container stack such as Kubernetes

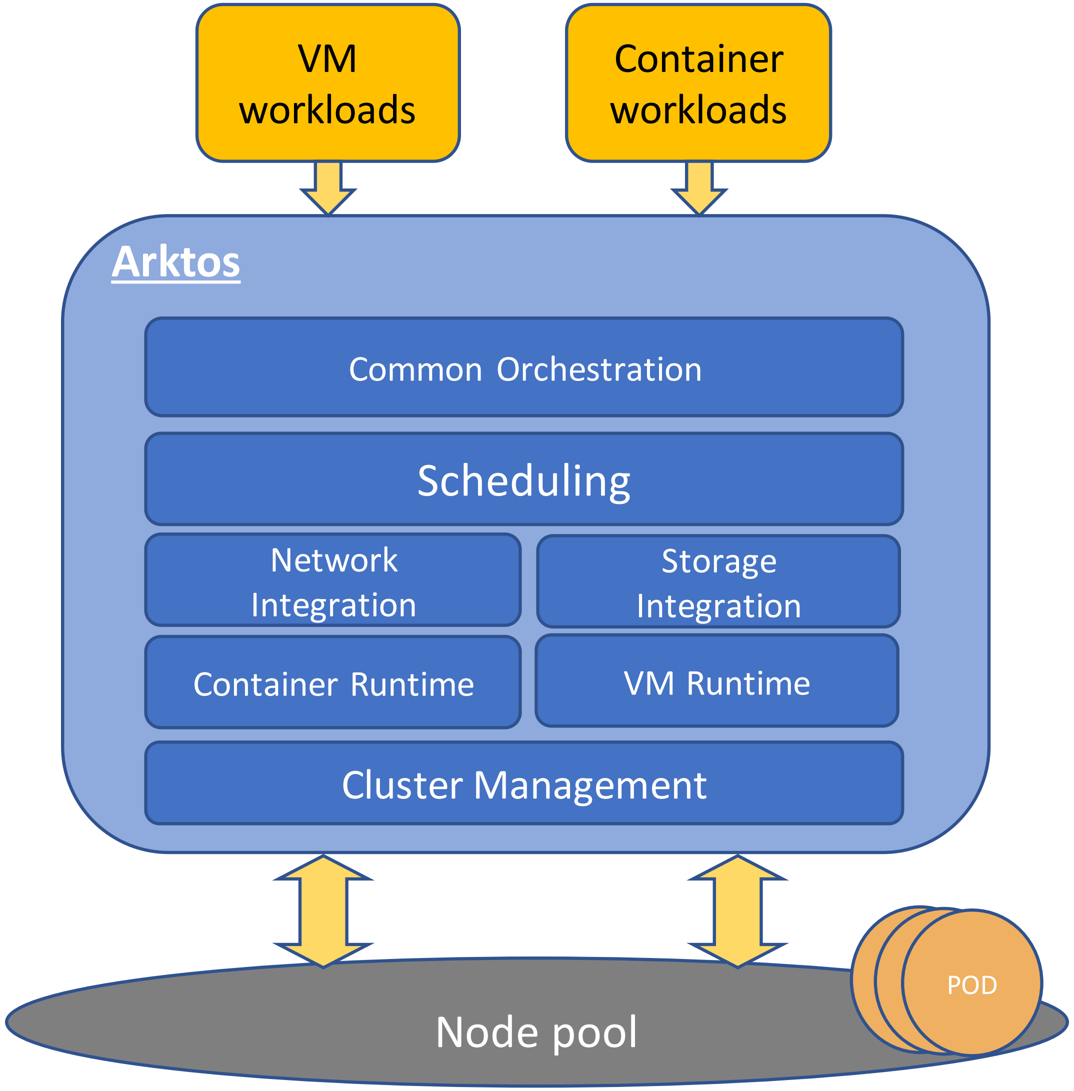
VM stack such as OpenStack Nova

🟦 Similar components

▨ Different components

- Having two separate stacks brings difficulty to development, operation and resource planning.
- It also hurts resource utilization by having separate resource pools.

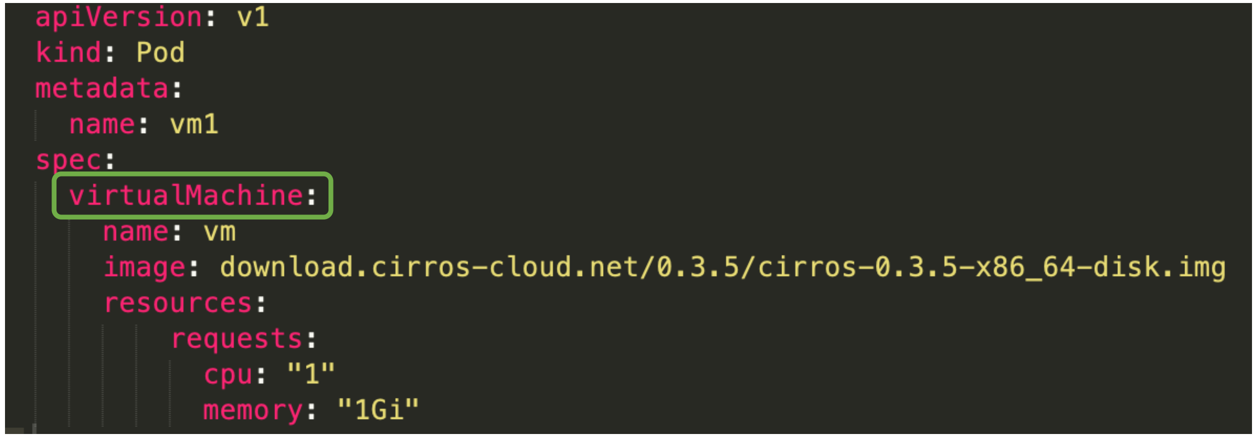
One Converged Stack with Arktos



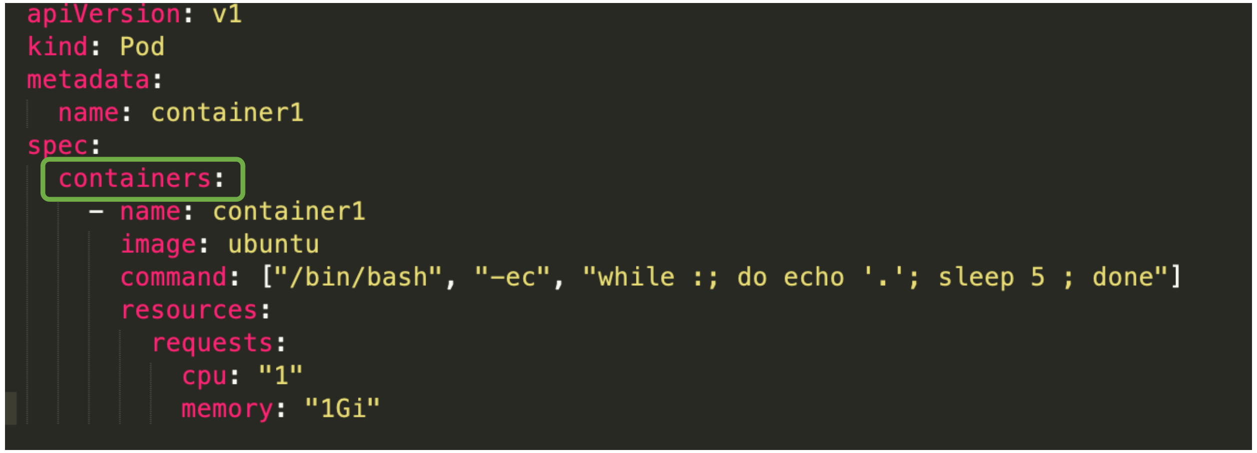
Different Options to Support VM in K8S

Addon-based Approach	Native Approach
Separate VM API objects	**Single API object hierarchy**
Additional operators and agents	**No additional components**
Additional tools	**Single tool chain**
No changes to Kubernetes	Fundamental changes inside K8s
Other offerings	**Arktos**

Native VM Support in Arktos



```
apiVersion: v1
kind: Pod
metadata:
  name: vm1
spec:
  virtualMachine:
    name: vm
    image: download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
    resources:
        requests:
          cpu: "1"
          memory: "1Gi"
```



```
apiVersion: v1
kind: Pod
metadata:
  name: container1
spec:
  containers:
    - name: container1
      image: ubuntu
      command: ["/bin/bash", "-ec", "while :; do echo '.'; sleep 5 ; done"]
      resources:
        requests:
          cpu: "1"
          memory: "1Gi"
```

APIs

- A pod contains one VM, or one or more containers
- Action object to support VM life-cycle

Scheduler

- Unified scheduling by a common representation of VM and container resources

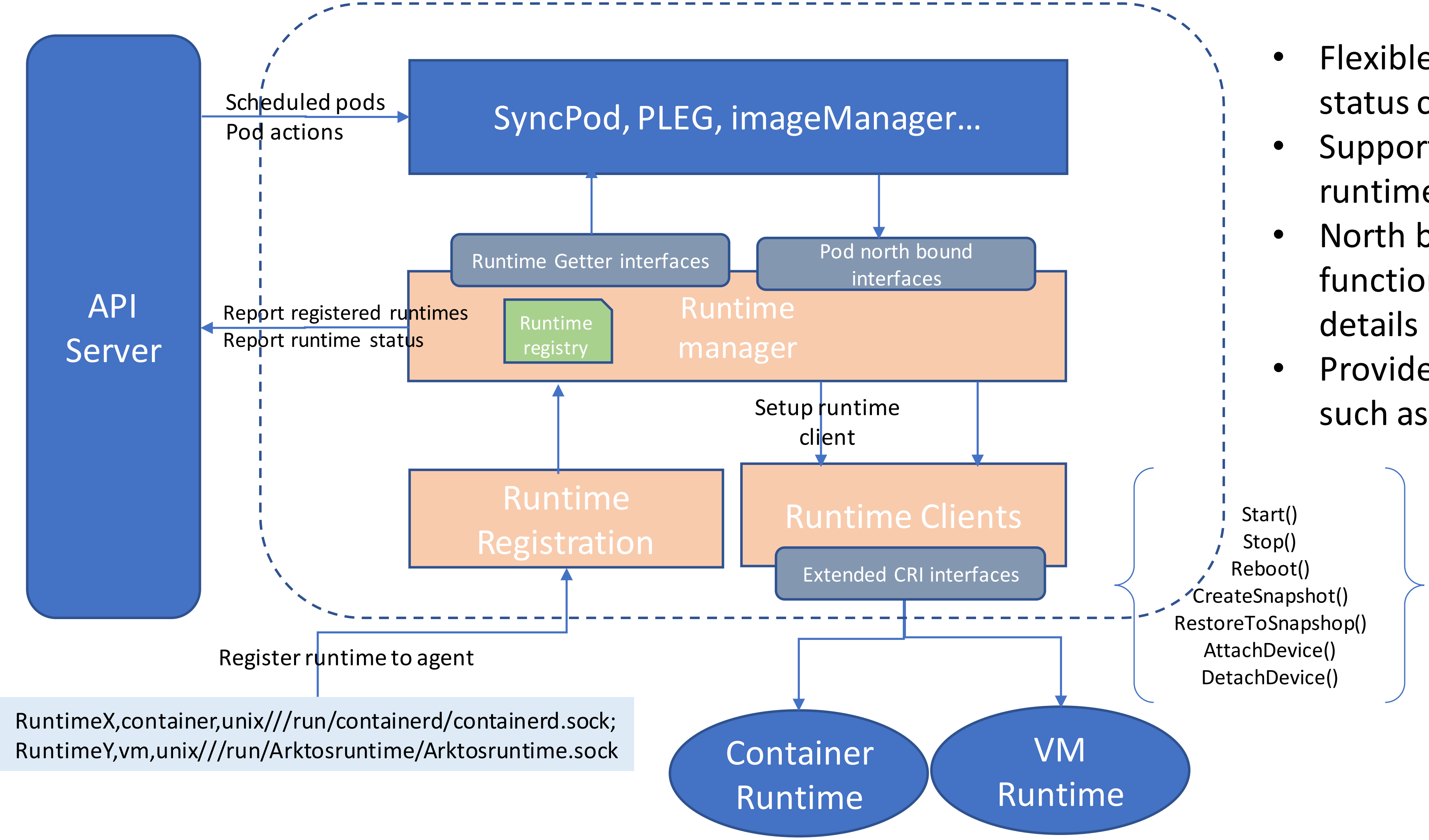
Controllers

- Reuse existing controllers like job controllers, RS controllers, etc

Agent

- Handle the VM object in sync loop
- Support multiple CRI endpoints for containers and VMs
- Extend CRI to add methods for VM
- A VM CRI runtime server

VM Pod: Multi Runtime



```
RuntimeX,container,unix///run/containerd/containerd.sock;
RuntimeY,vm,unix///run/Arktosruntime/Arktosruntime.sock
```

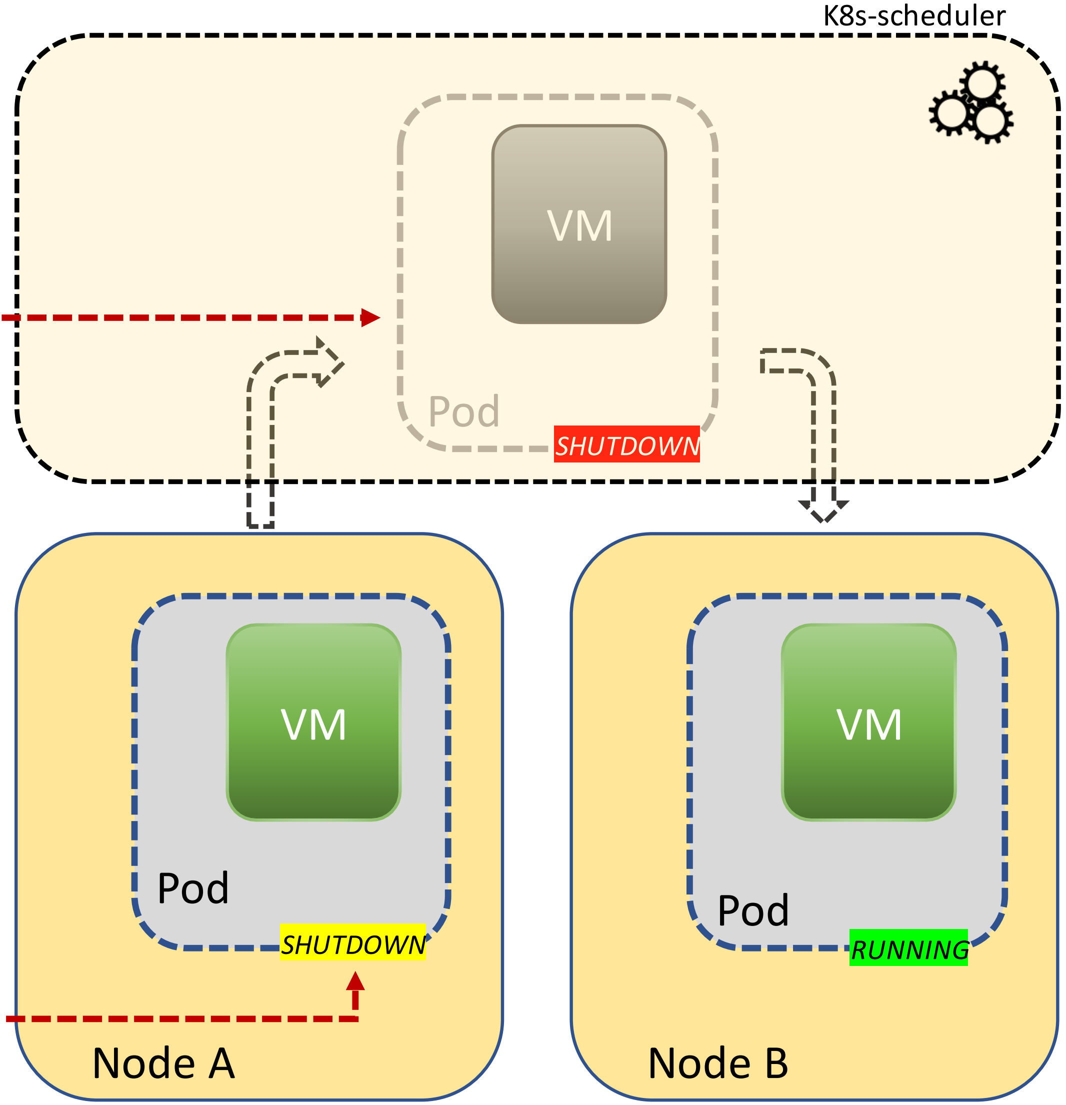
- Flexible runtime registration and status check/reports
- Supports both VM and Container runtime services
- North bound API abstracts runtime functionalities and implementation details
- Provides foundation for future works such as unified image manager

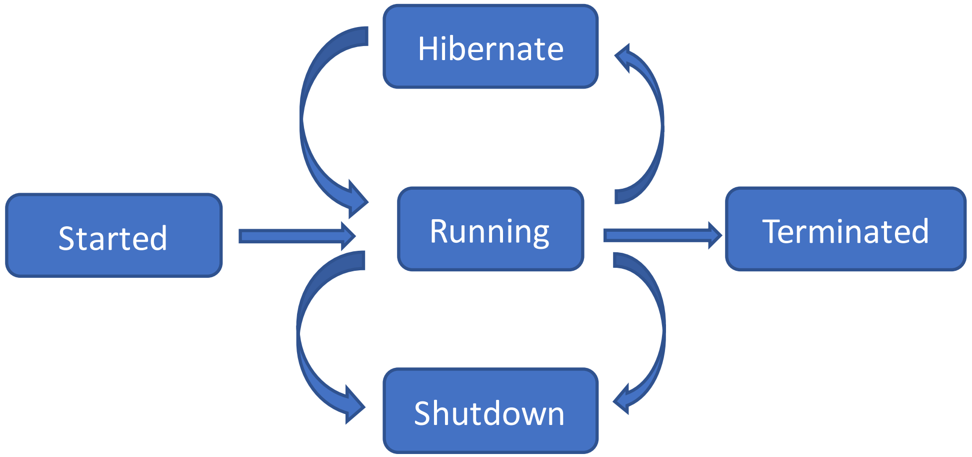
VM Pod: State Management

PodSpec

patch →

```
virtualMachine:
  image: cloudImage
  imagePullPolicy: IfNotPresent
  name: demoVm
   powerSpec: Running
```





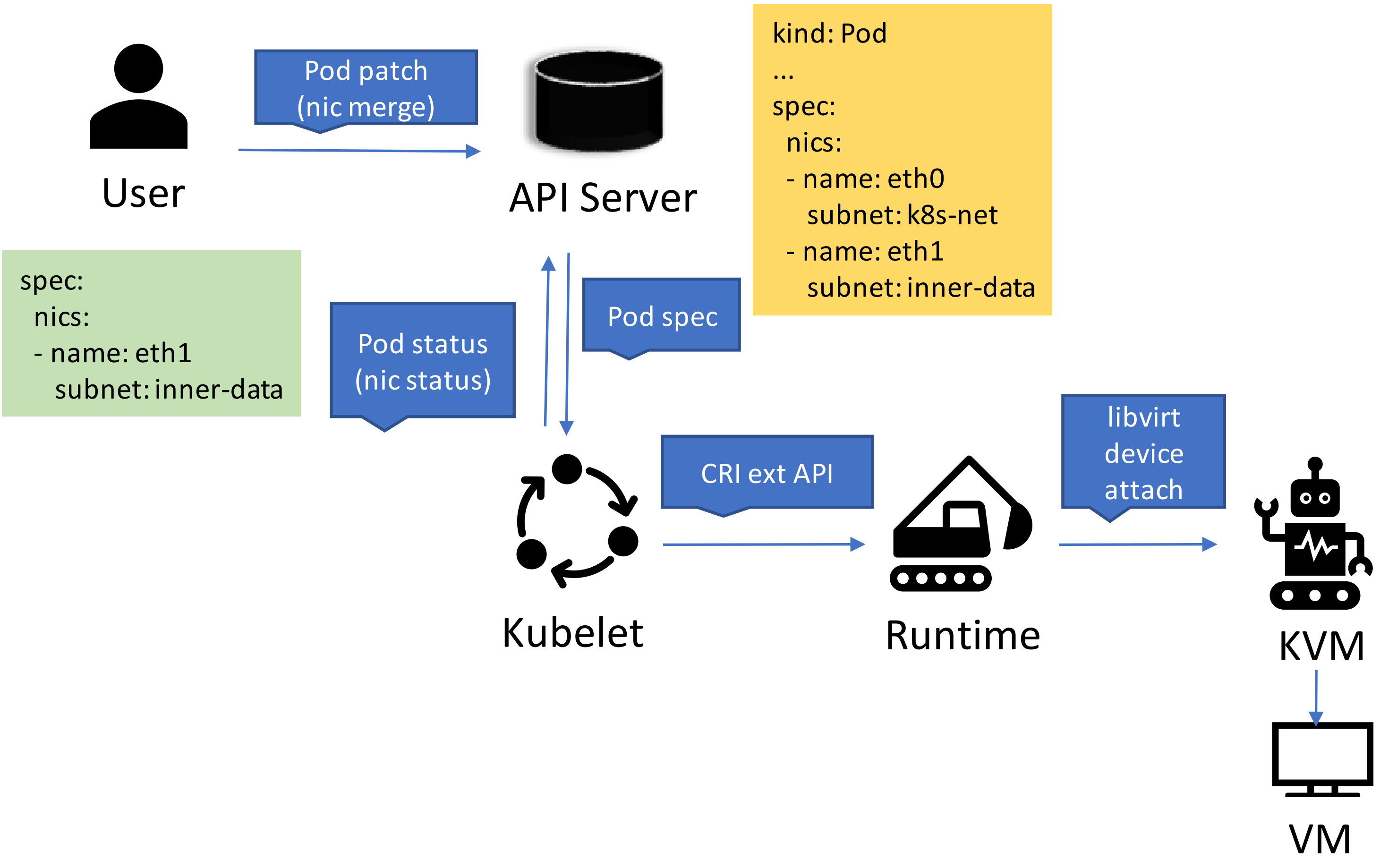
Demo: Start and Stop VM

VM Pod: Configuration Management

- user changes the desired VM resources in pod spec;
- system reconciles to ensure actual resources eventually in line with the desired;
- system reports the actual resources as part of pod status.

VM resources	Op to support
CPU number	Update
memory	Update
network interface	Hot plug
disk storage	Hot plug

Example: NIC Hot Plug Message Flow



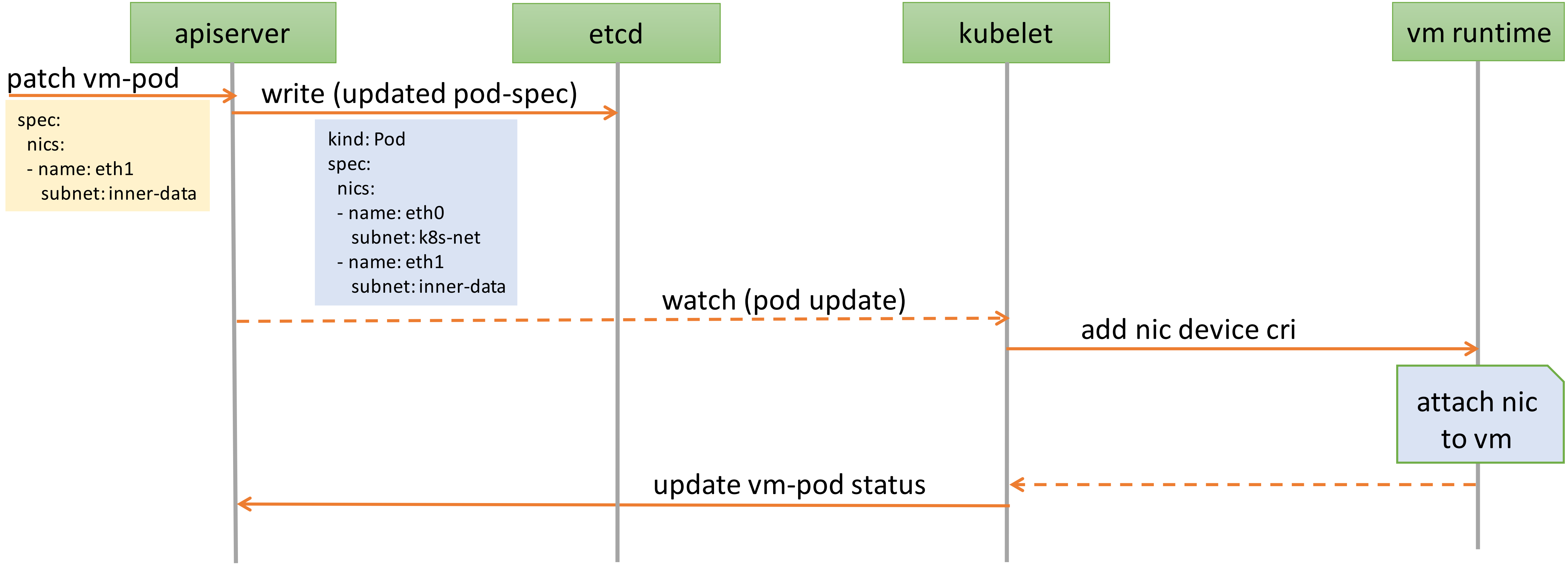
```
kind: Pod
...
spec:
  nics:
  - name: eth0
    subnet: k8s-net
  - name: eth1
    subnet: inner-data
```

```
spec:
  nics:
  - name: eth1
    subnet: inner-data
```

VM Pod Configuration Management

- K8s user changes the desired VM resources by PATCHing Pod Spec of a running VM Pod
- System reconciles to ensure actual resources eventually match desired resources
- Supports updating VM CPU/memory resources, and NIC/storage hot-plug

Example: NIC Hot Plug Workflow

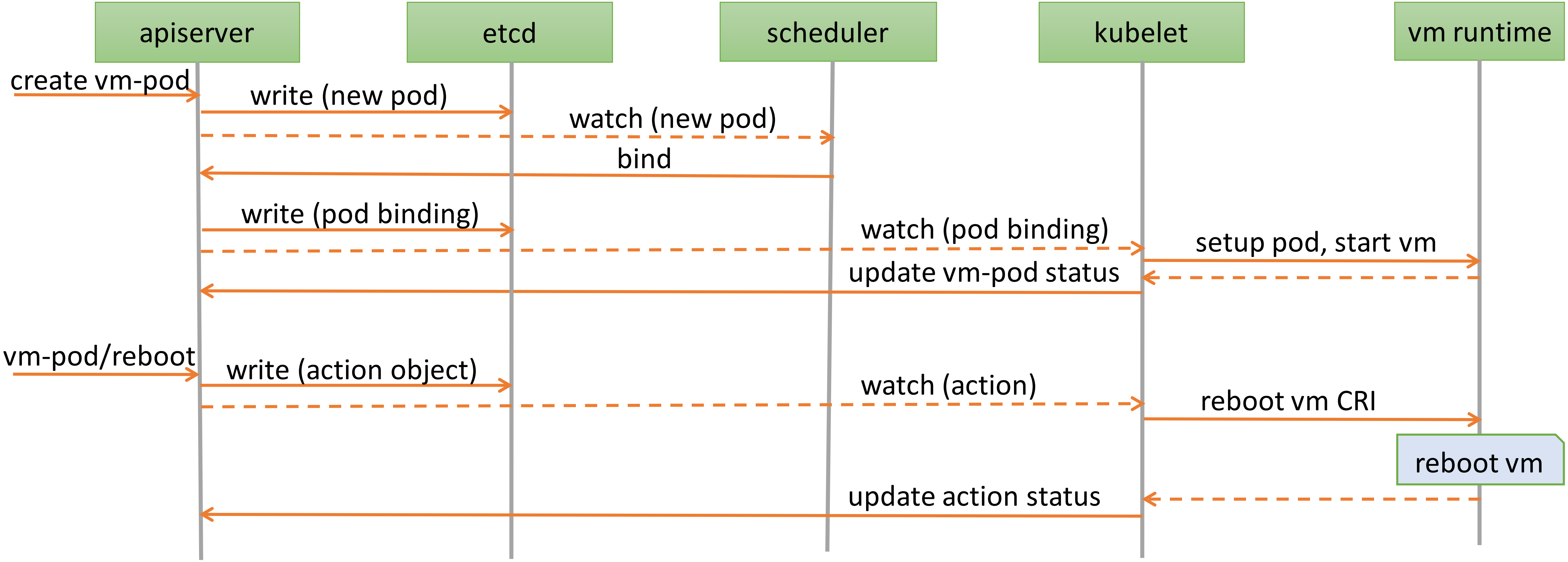


```
spec:
  nics:
  - name: eth1
     subnet: inner-data
```

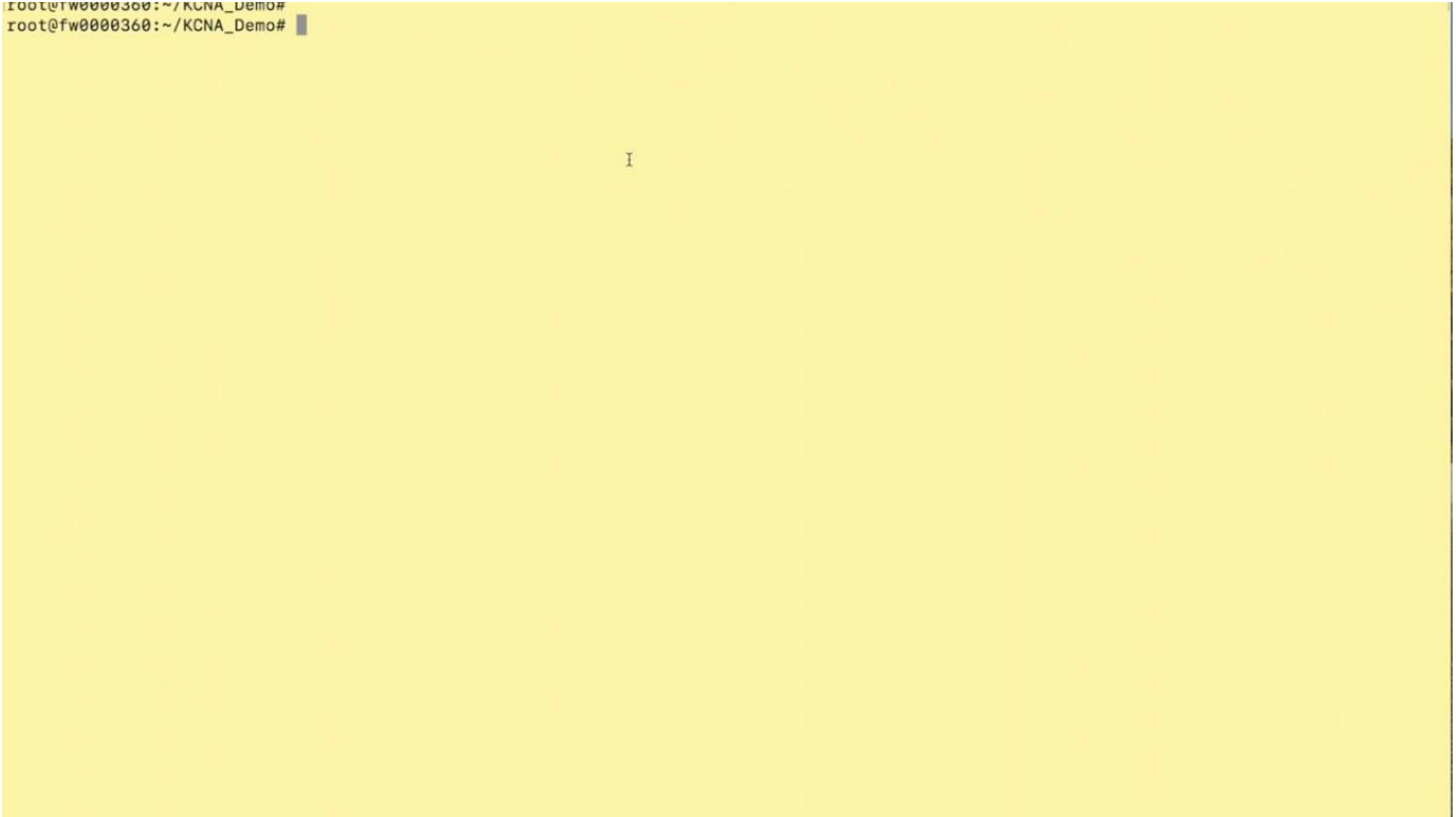
```
kind: Pod
spec:
  nics:
  - name: eth0
     subnet: k8s-net
  - name: eth1
     subnet: inner-data
```

VM Pod: Action Framework

- Allows user to perform operations on VM Pod
  - Examples: Reboot a VM, Take a VM snapshot, …
- User specifies desired Action by POSTing to pods/action subresource
- Agent responsible for Action watches for actions, implements it, and updates status

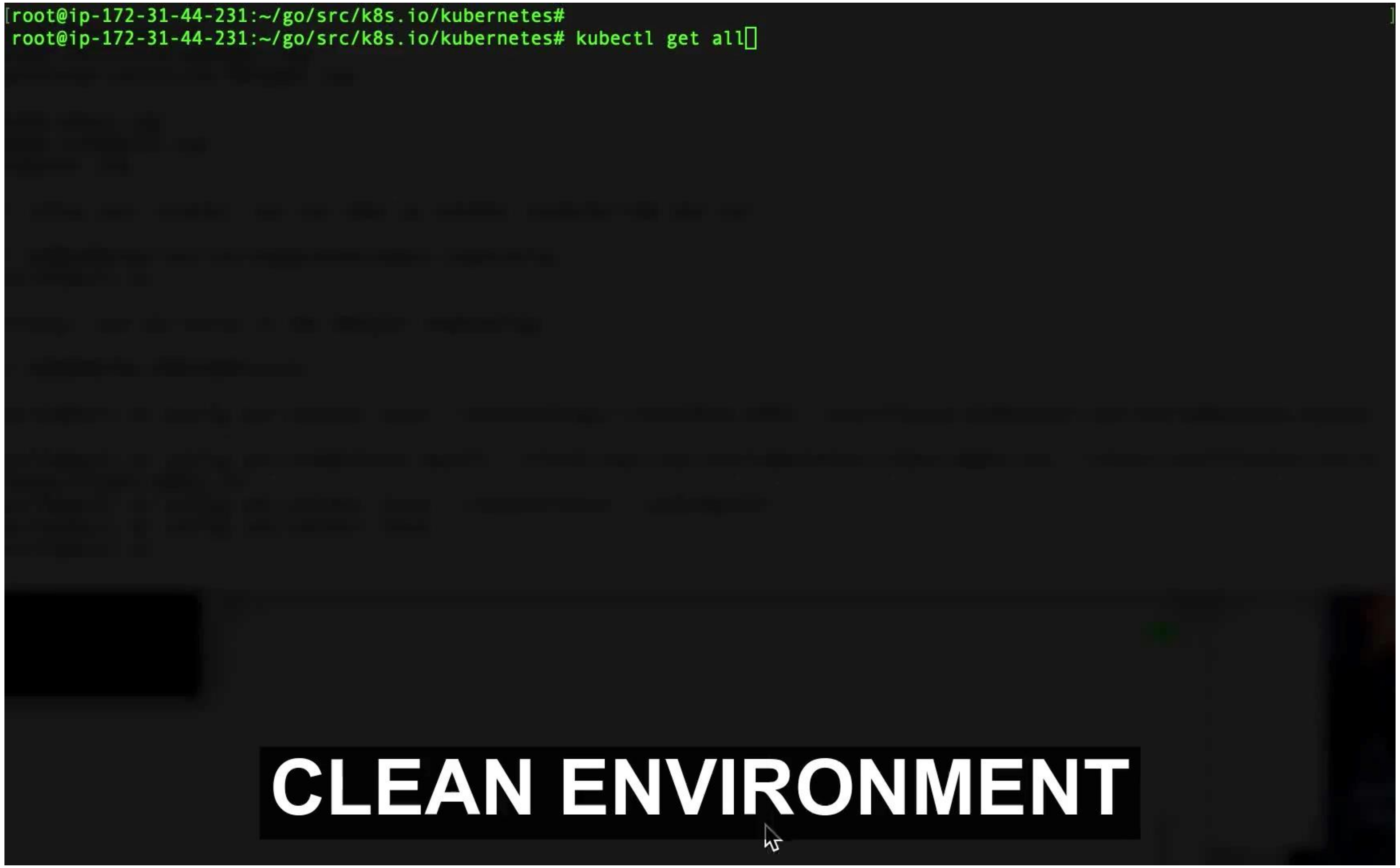


Demo: Snapshot and Restore a VM



```
root@fw0000360:~/KCNA_Demo#
root@fw0000360:~/KCNA_Demo# 
```

Demo: VM ReplicaSet Support



```
root@ip-172-31-44-231:~/go/src/k8s.io/kubernetes#
root@ip-172-31-44-231:~/go/src/k8s.io/kubernetes# kubectl get all
```

CLEAN ENVIRONMENT

- Support VM ReplicaSet
- Sample VM replicaset yaml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: demo
  labels:
    app: demoapp
    tier: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      virtualMachine:
        keyPairName: "foobar"
        name: vm
        image: "download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img"
        imagePullPolicy: IfNotPresent
        resources:
          limits:
            cpu: "1"
            memory: "200Mi"
          requests:
            cpu: "0.1"
            memory: "200Mi"
```

Thank you.