
KubeCon
CloudNativeCon
North America 2019

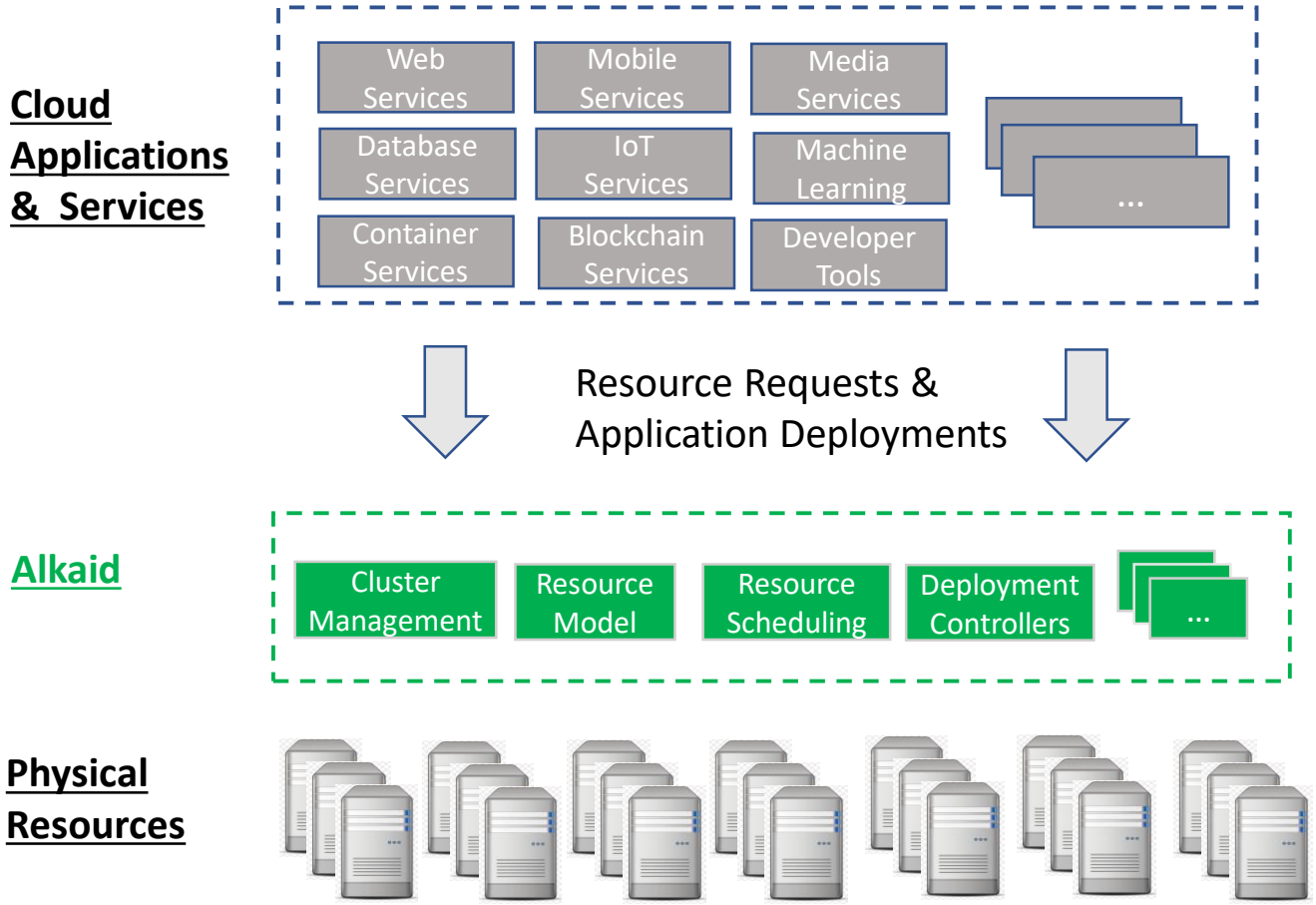
Alkaid Compute

Cloud Lab, Futurewei Technologies

Agenda

- Alkaid Overview
- Key Features
  - Multi Tenancy
  - Large Scalability
  - Unified VM/Container Stack

Alkaid Overview



Hard Multi-tenancy

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

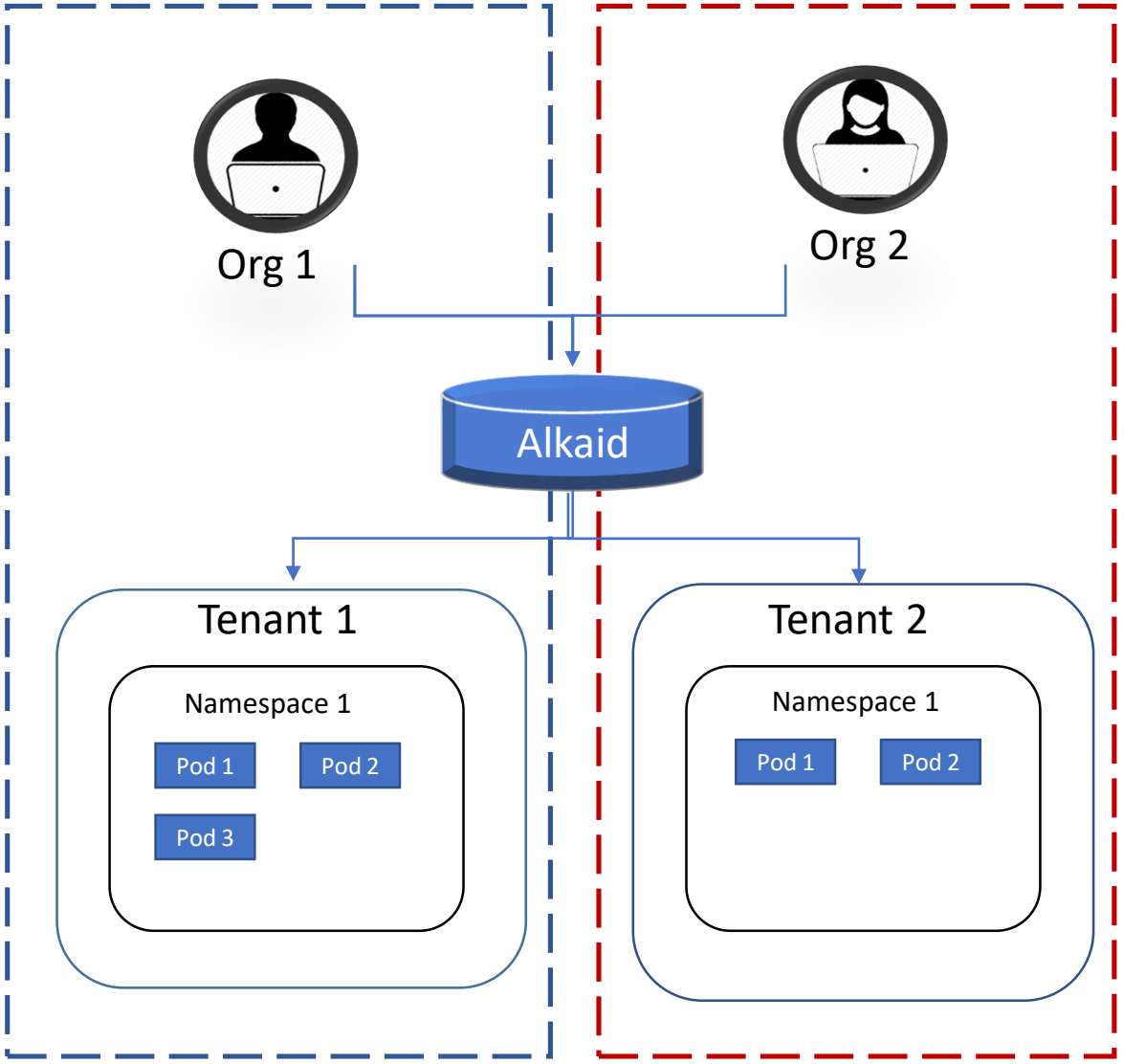
Cloud Scale

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

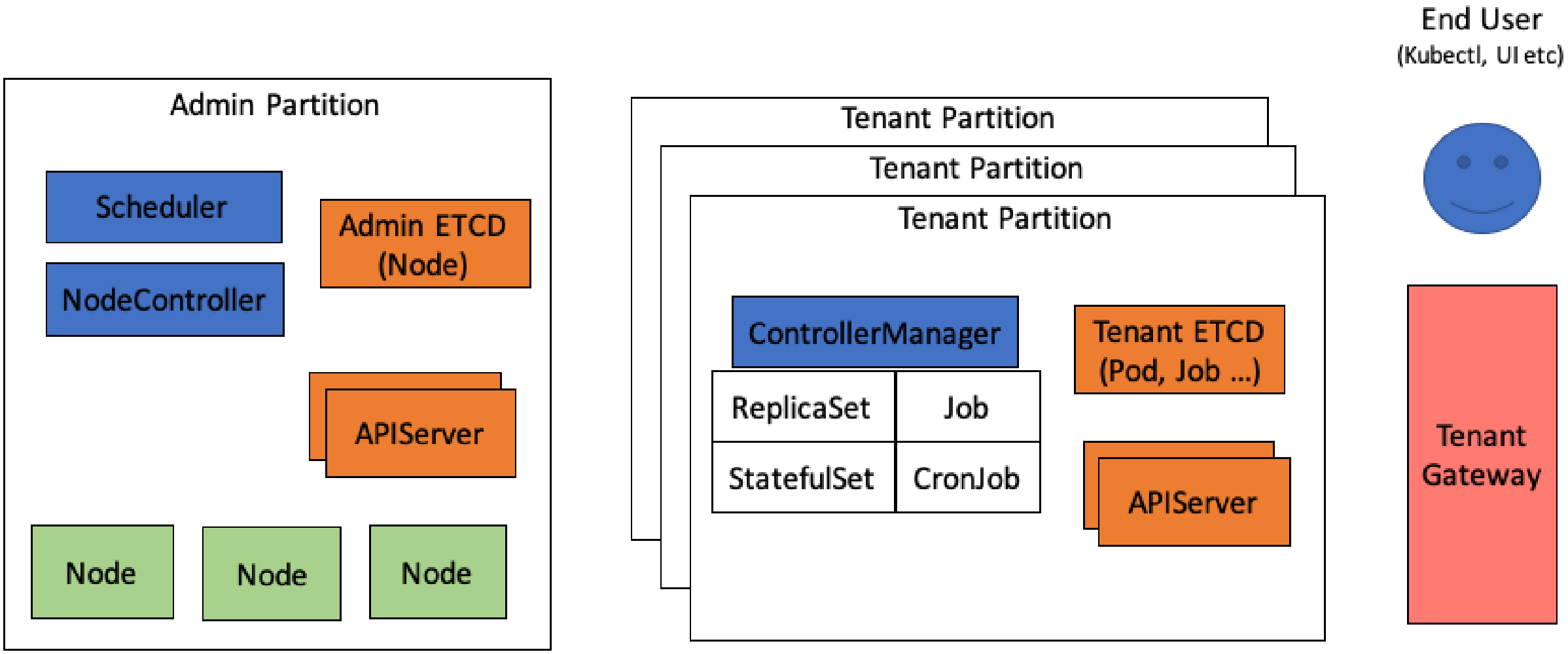
- Enable organizations/departments to safely share one infrastructure, without deploying/operating multiple clusters.
- Support per-tenant resource view, access control, quota, etc.
- Assume no trust among tenants; ready for strict scenarios like public cloud.

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.

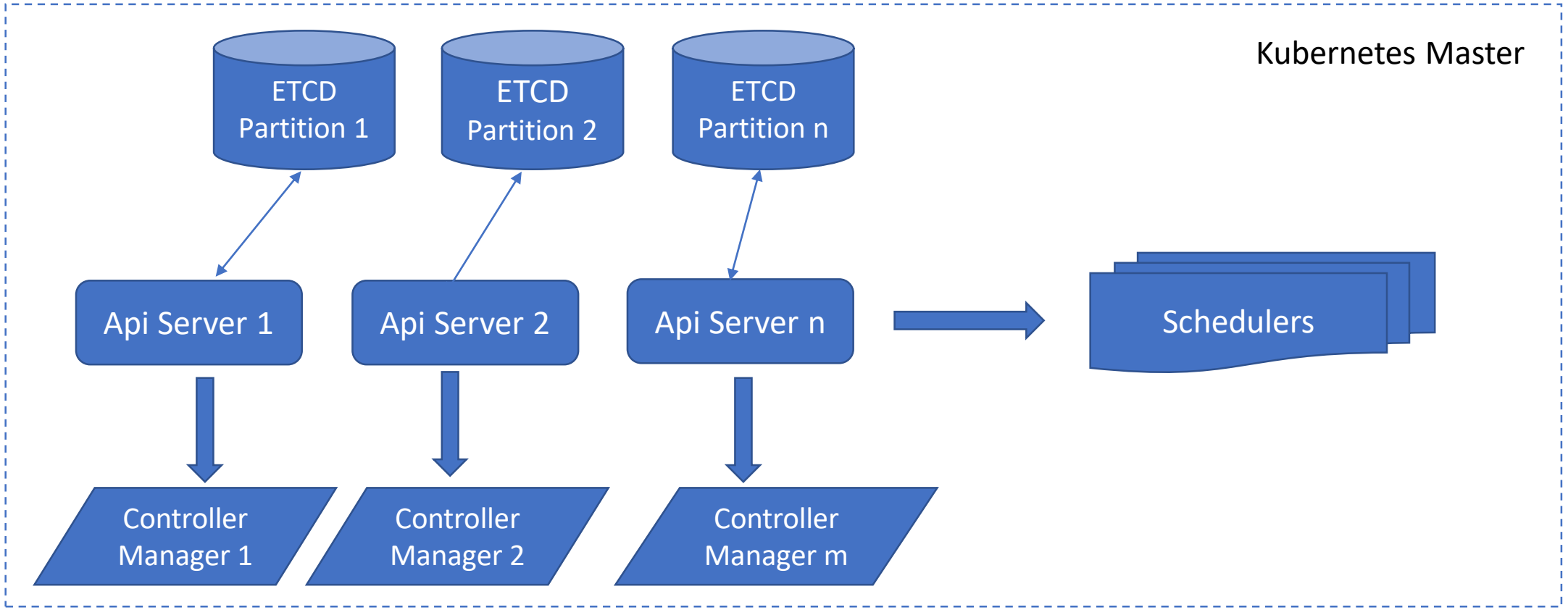
Demo: Multi-Tenancy

Scalability Architecture Option 1



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

Scalability Architecture Option 2

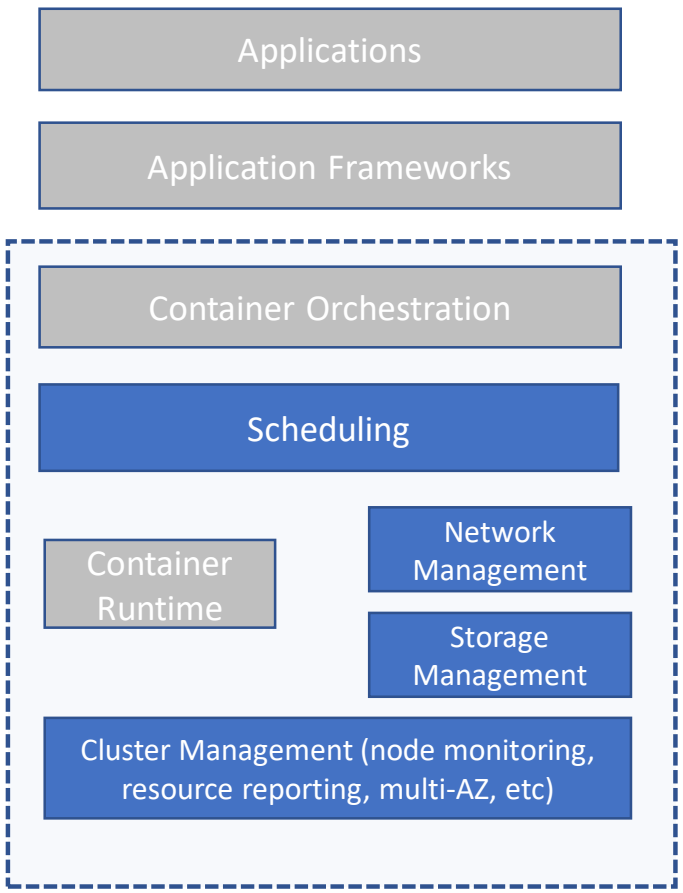


- Allow list/watch by range of field value (demo?)
- Support multiple controller instances with the same type (demo?)
  - Add hash key to metadata
  - Split workloads by hash key and assign to a single controller instance based on hash key range

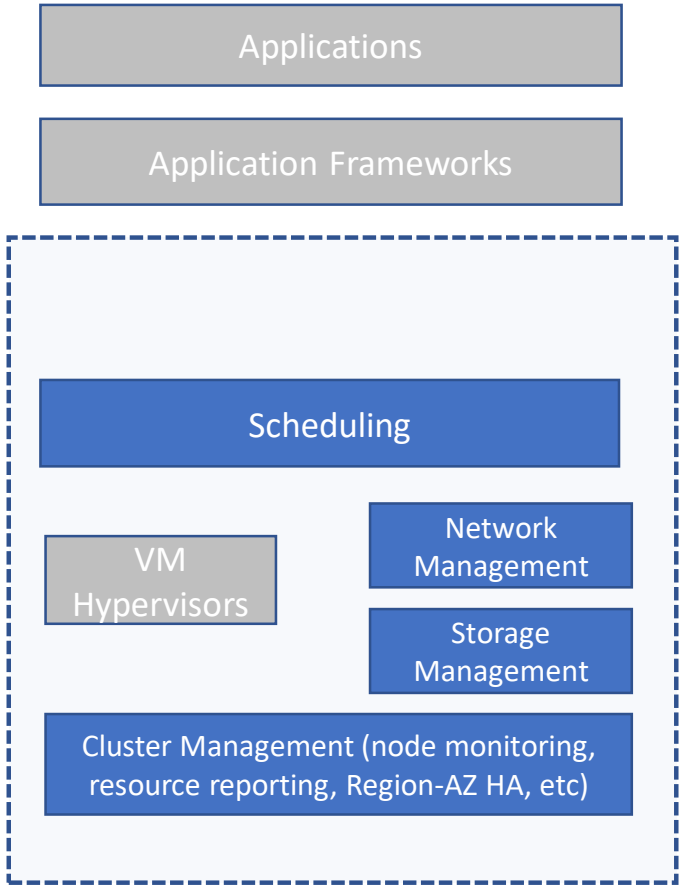
Scalability Performance Test Output

- ReplicaSet controller (POC)
  - Latency
  - Throughput

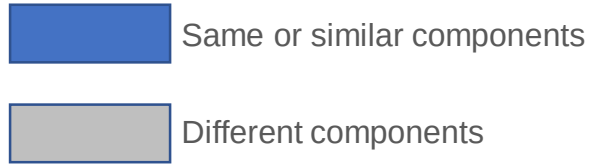
Two Stacks in Today's Data Center



Container stack such as Kubernetes

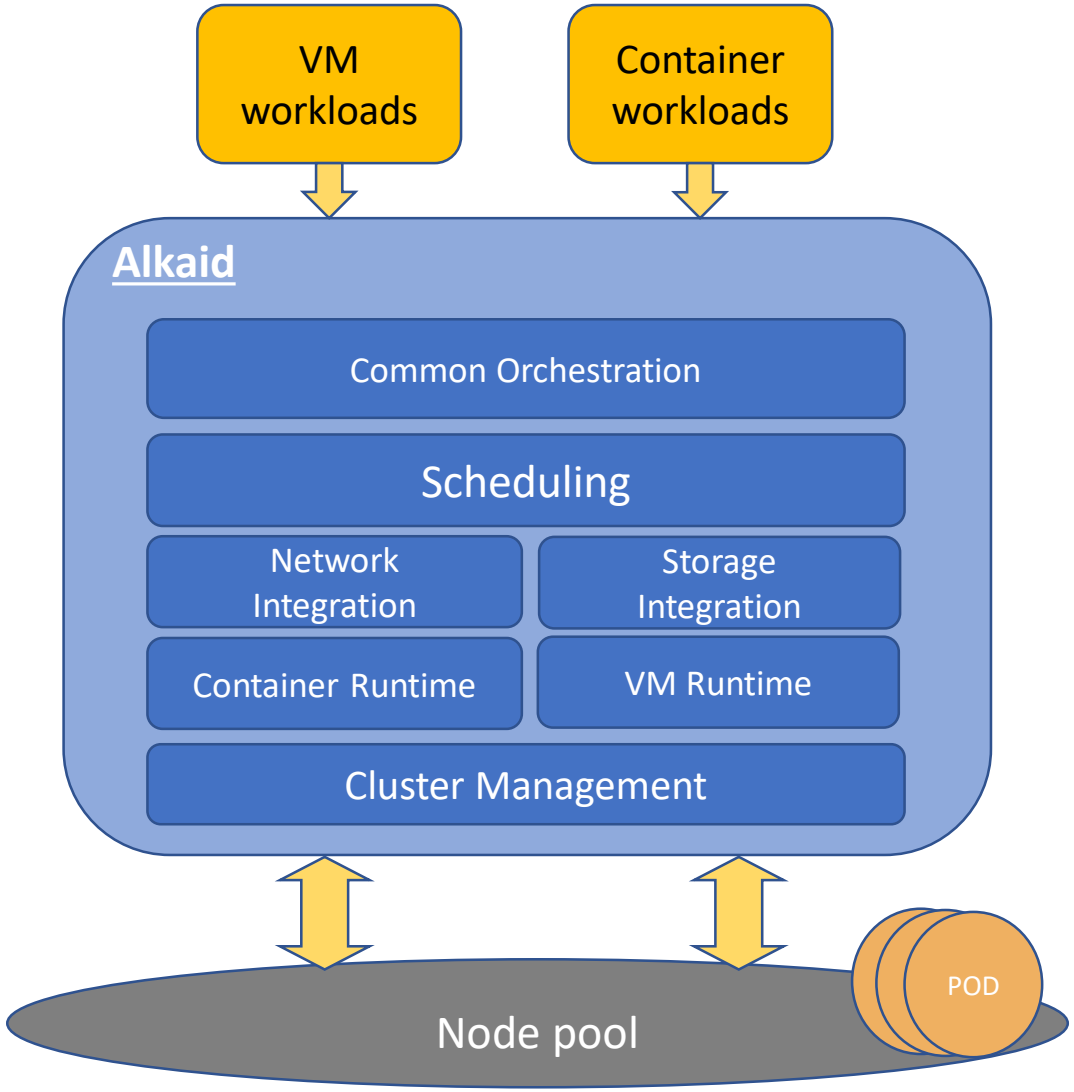


VM stack such as OpenStack Nova



- 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 Alkaid



Different Options to Support VM in K8S

Addon-based Approach	Native Approach
VM defined as CRD	**Single API object hierarchy**
Additional controllers and agents	**No additional components**
Additional tools	**Single tool chain**
No changes to Kubernetes	**Fundamental changes inside K8s**
Other offerings	**Alkaid**

Native VM Support in Alkaid


```yaml
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"
```

```yaml
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

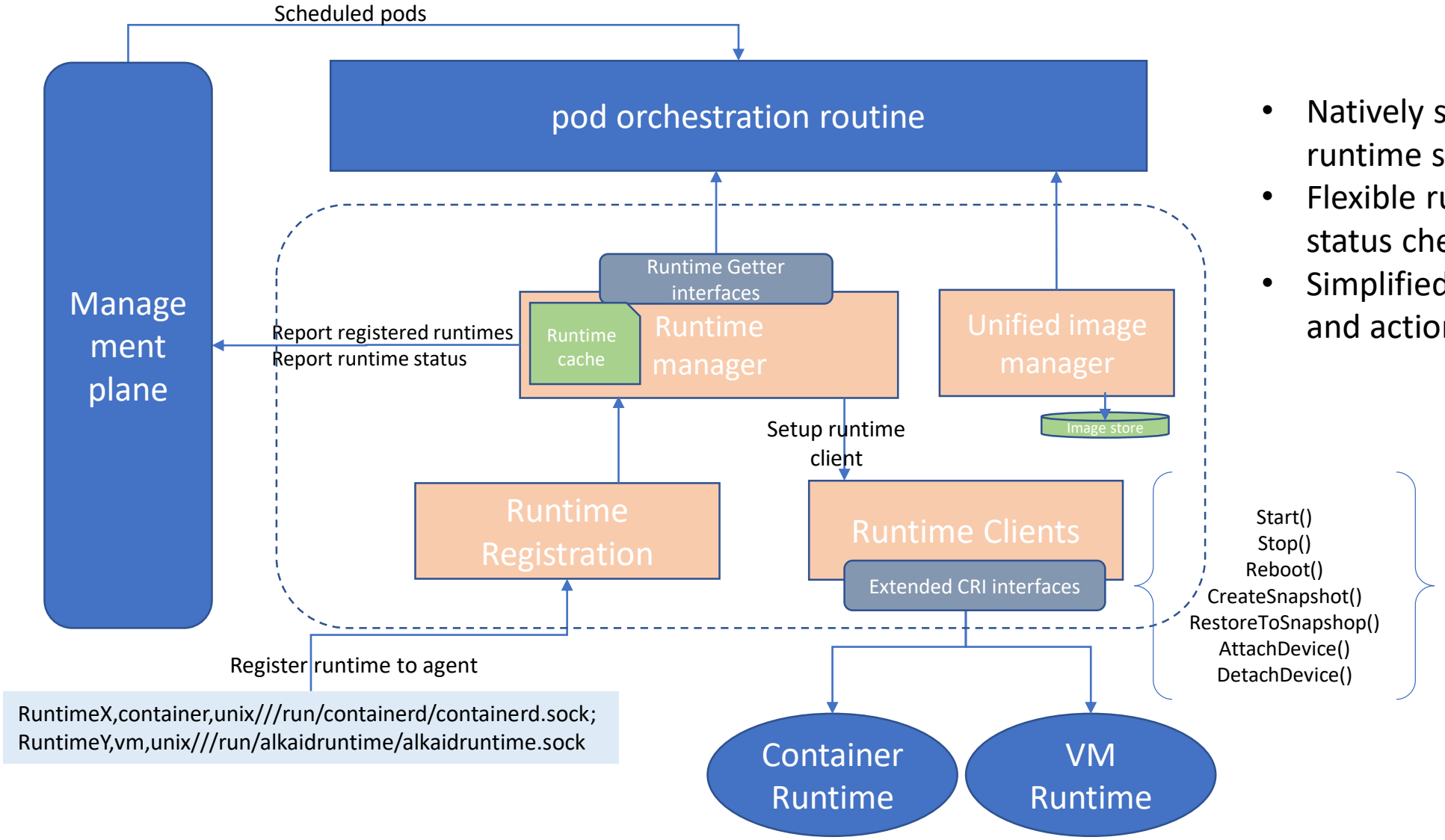
Controllers

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

Agent

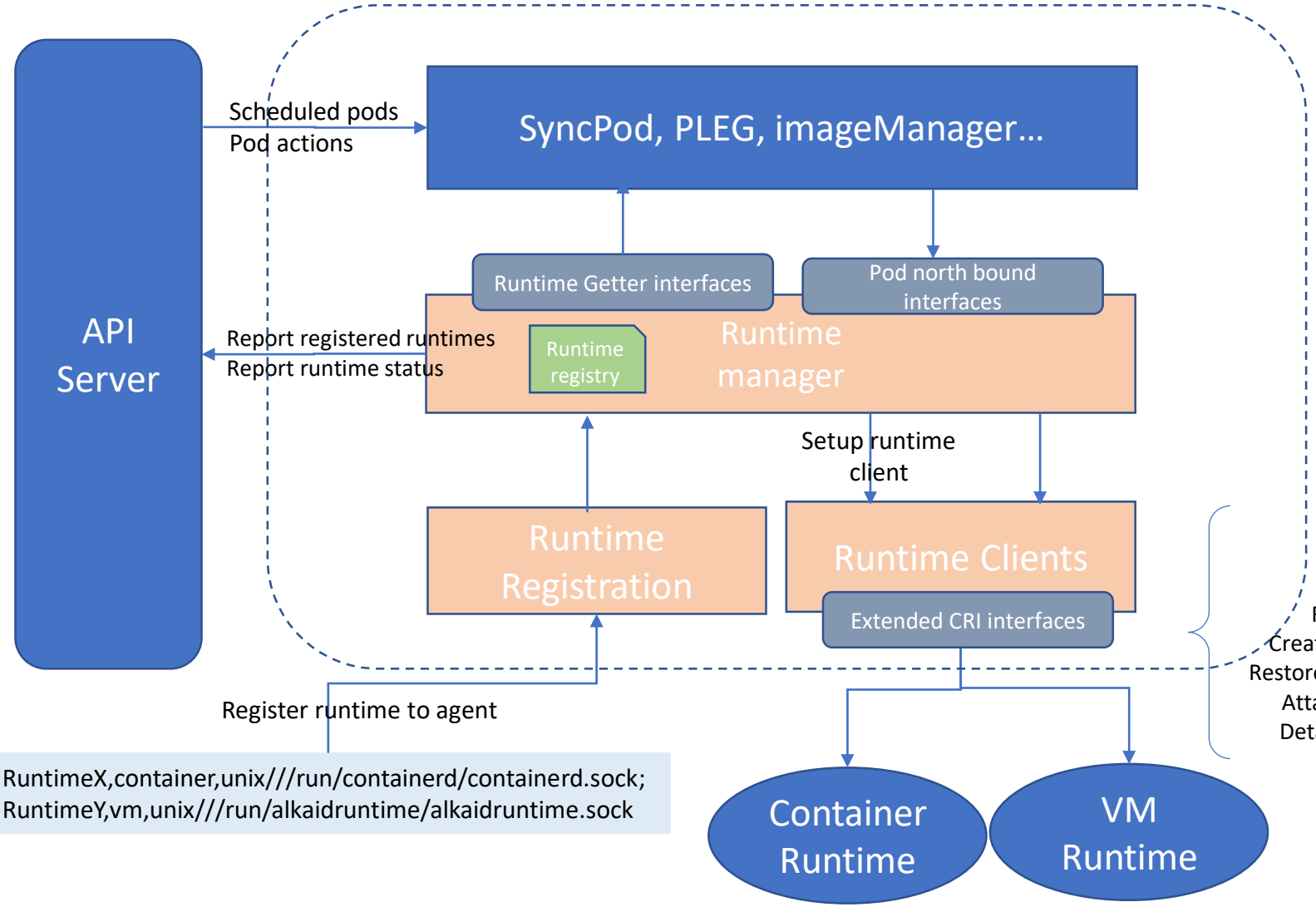
- Handle the VM object in sync loop
- Support multiple CRI endpoints for containers and VMs
- Extend CRI to add methods for VM
- A VM CRI runtime server

VM Pod: Built-in Multi Runtime



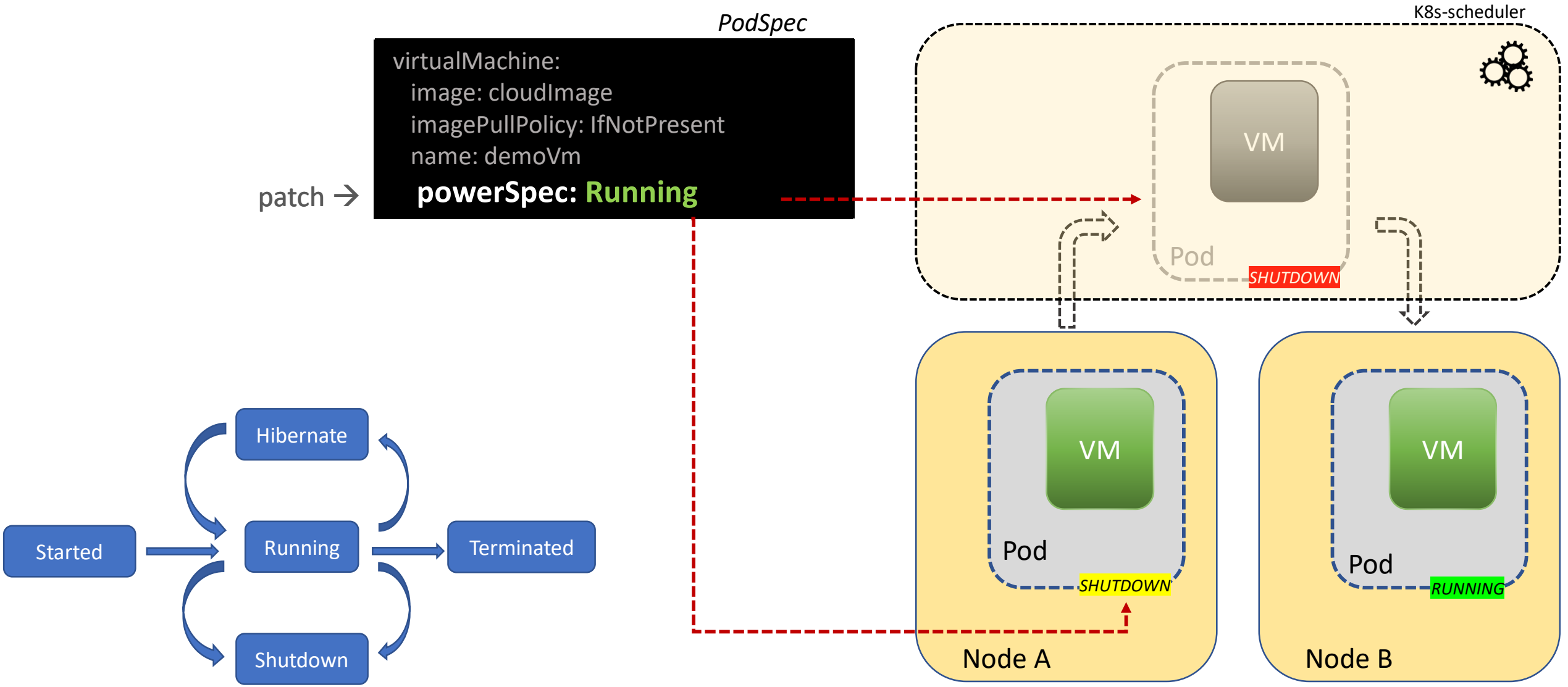
- Natively supports VM and container runtime services
- Flexible runtime registration and status check/reports
- Simplified workload orchestration and actions flows

VM Pod: Multi Runtime

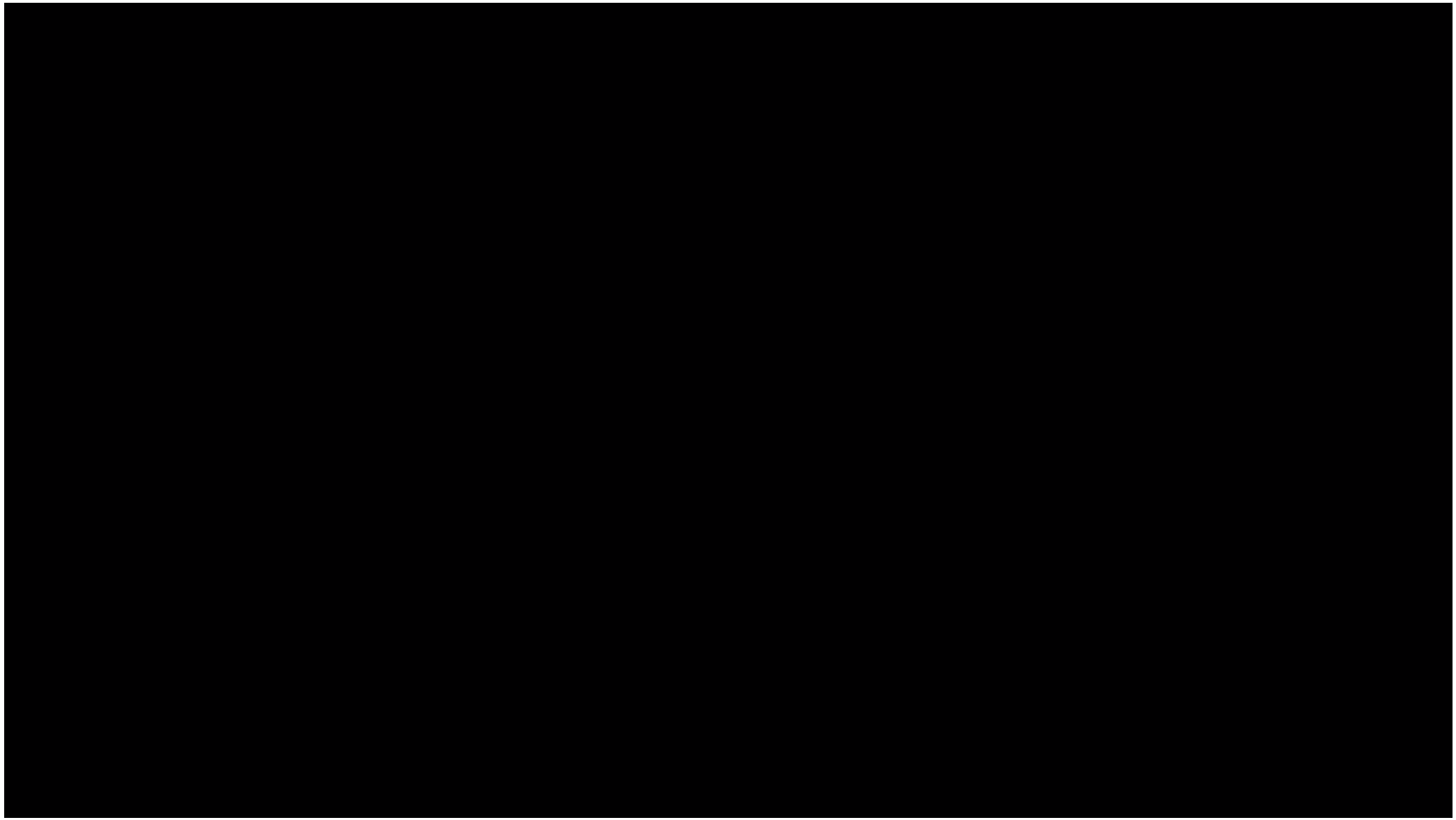


- Flexible runtime registration and status check/reports
- Supports both VM and Container runtime services
- North bound API abstracts runtime functionalities and implementation details
- Provides foundation for future works such as unified image manager

VM Pod: State Management



Demo: Start and Stop VM

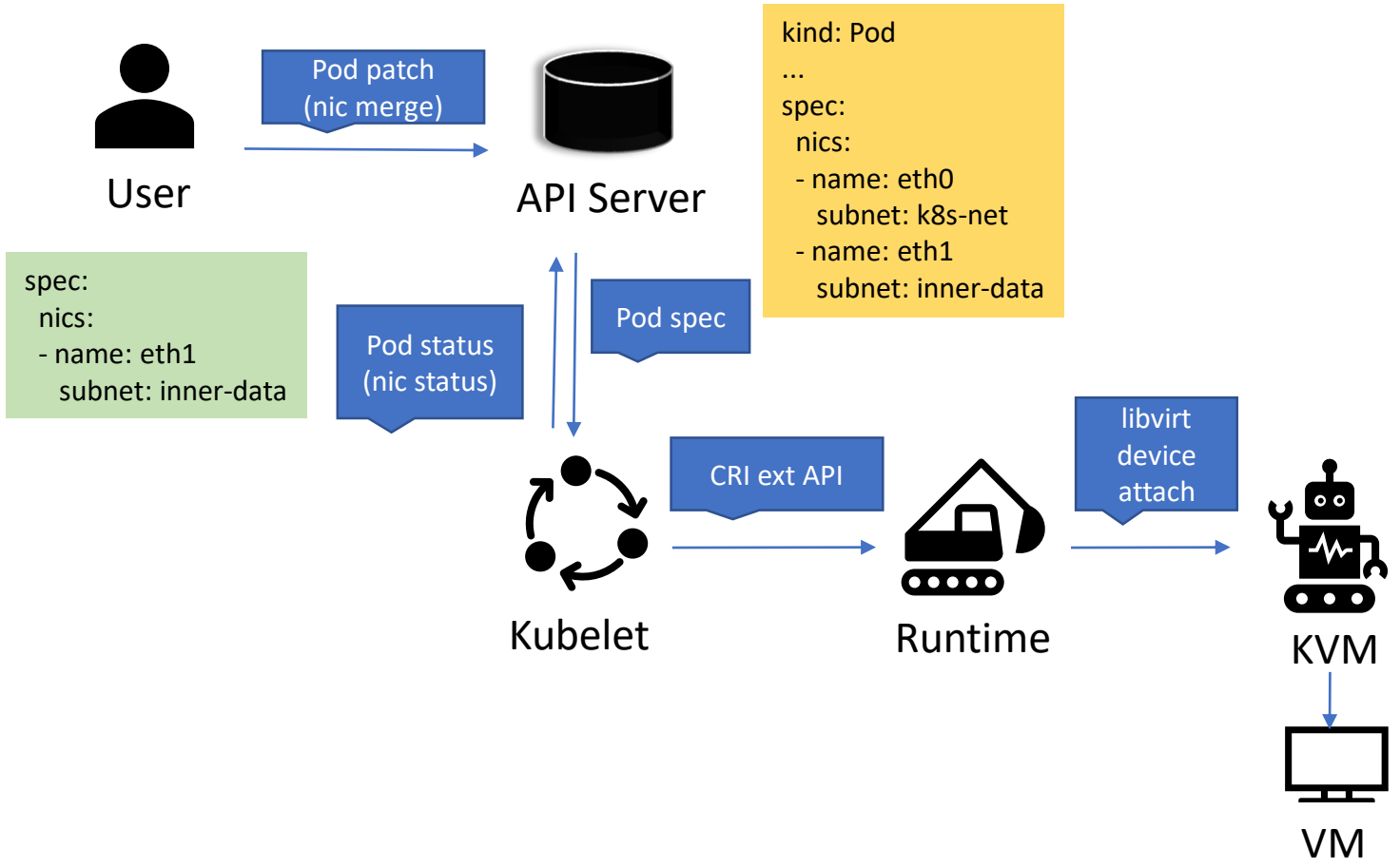


VM Pod: Configuration Management

- 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

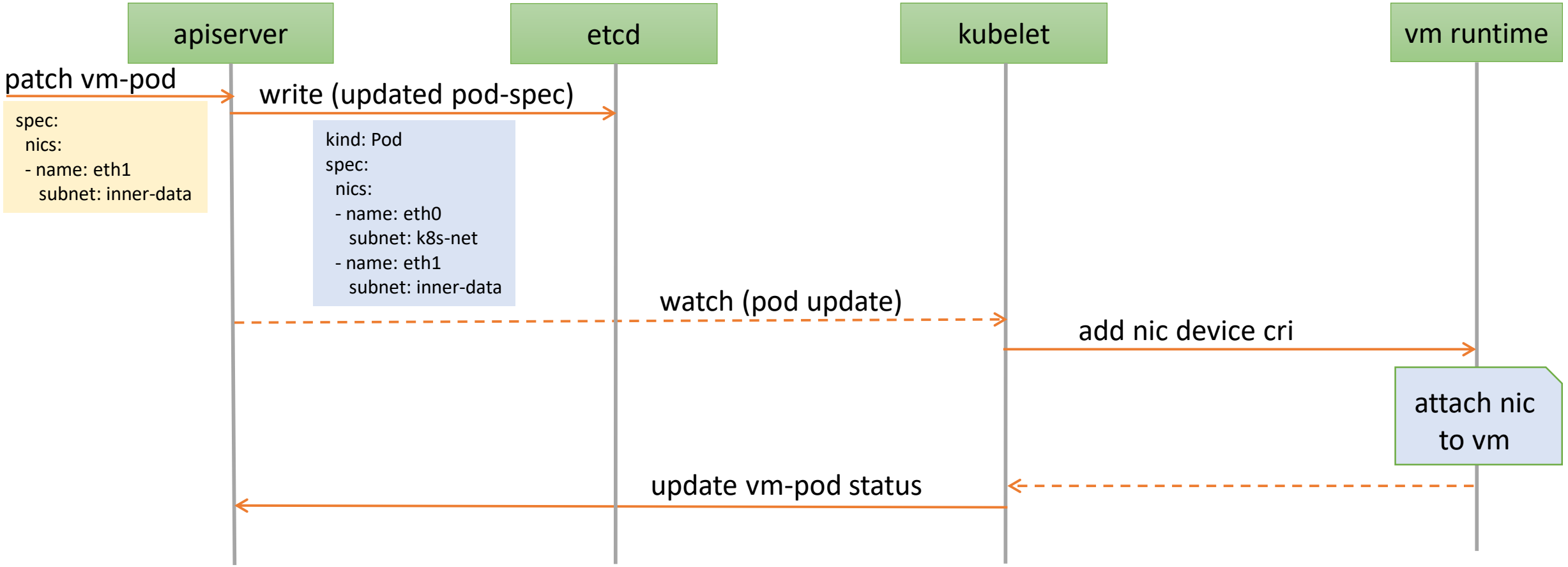
Example: NIC Hot Plug Message Flow



VM Pod Configuration Management

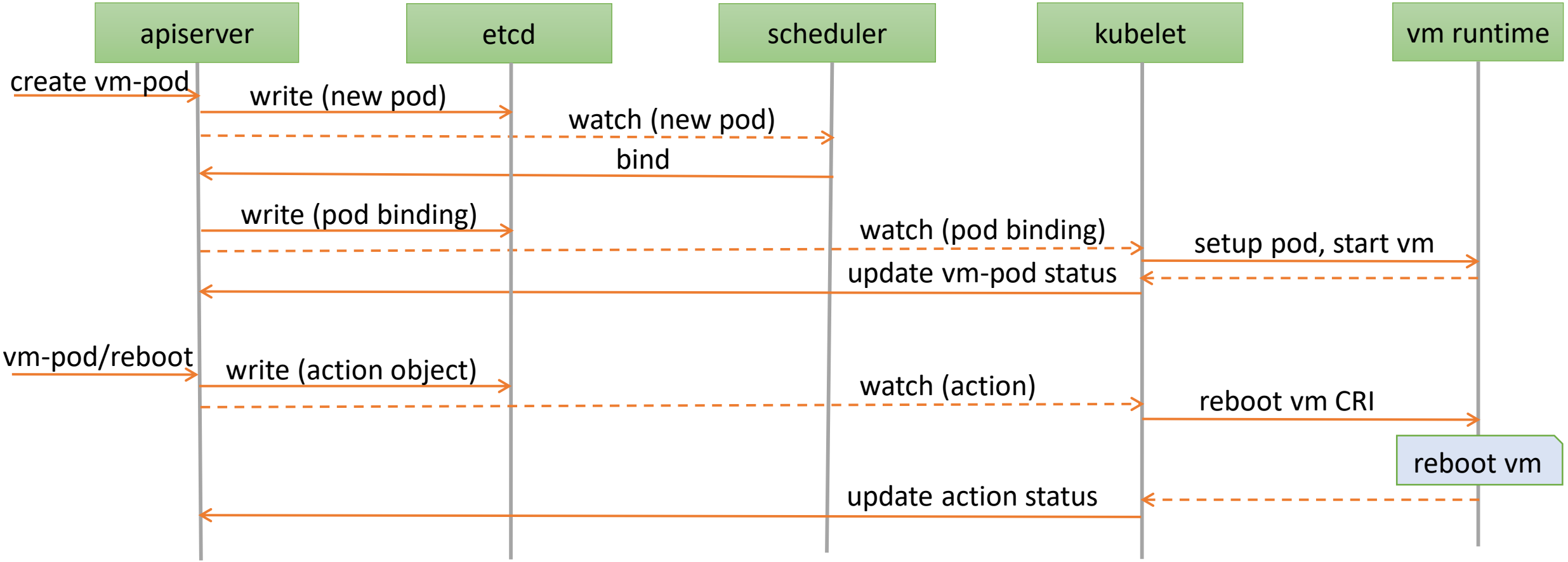
- K8s user changes the desired VM resources by PATCHing Pod Spec of a running VM Pod
- System reconciles to ensure actual resources eventually match desired resources
- Supports updating VM CPU/memory resources, and NIC/storage hot-plug

Example: NIC Hot Plug Workflow



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#
```

The End