Rancher Training

Session 3

RANCHER LABS

Arvin • 赵雨川

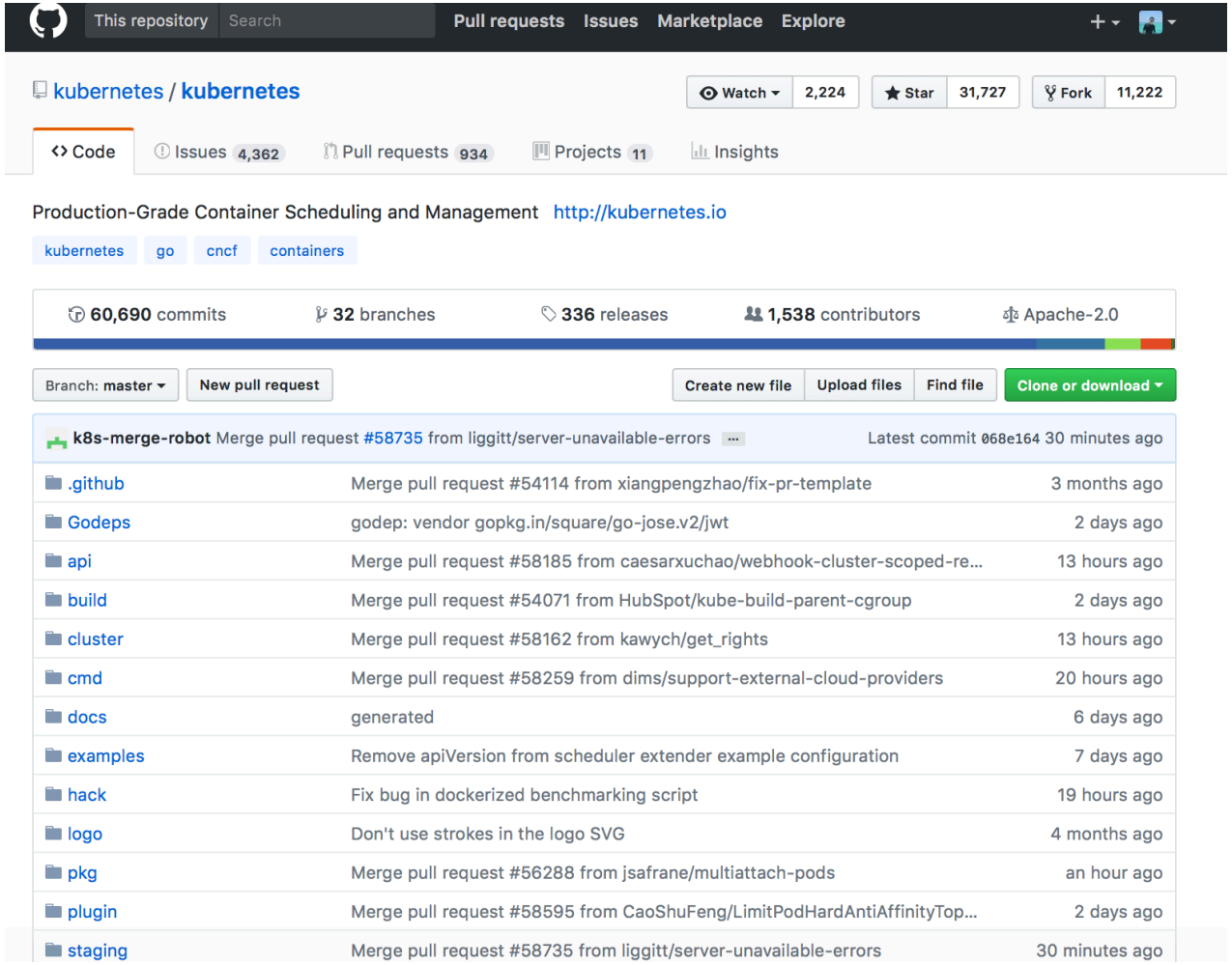
Morning Agenda

- Kubernetes
  - Kubernetes architecture
  - Deployment
  - How to access the cluster
  - Kubernetes in Rancher
  - Helm
  - Kubernetes Master HA
- Kubernetes deployment on Rancher
- RKE
- RancherOS（Zhibo）
- Exercise 3
- Exercise 4

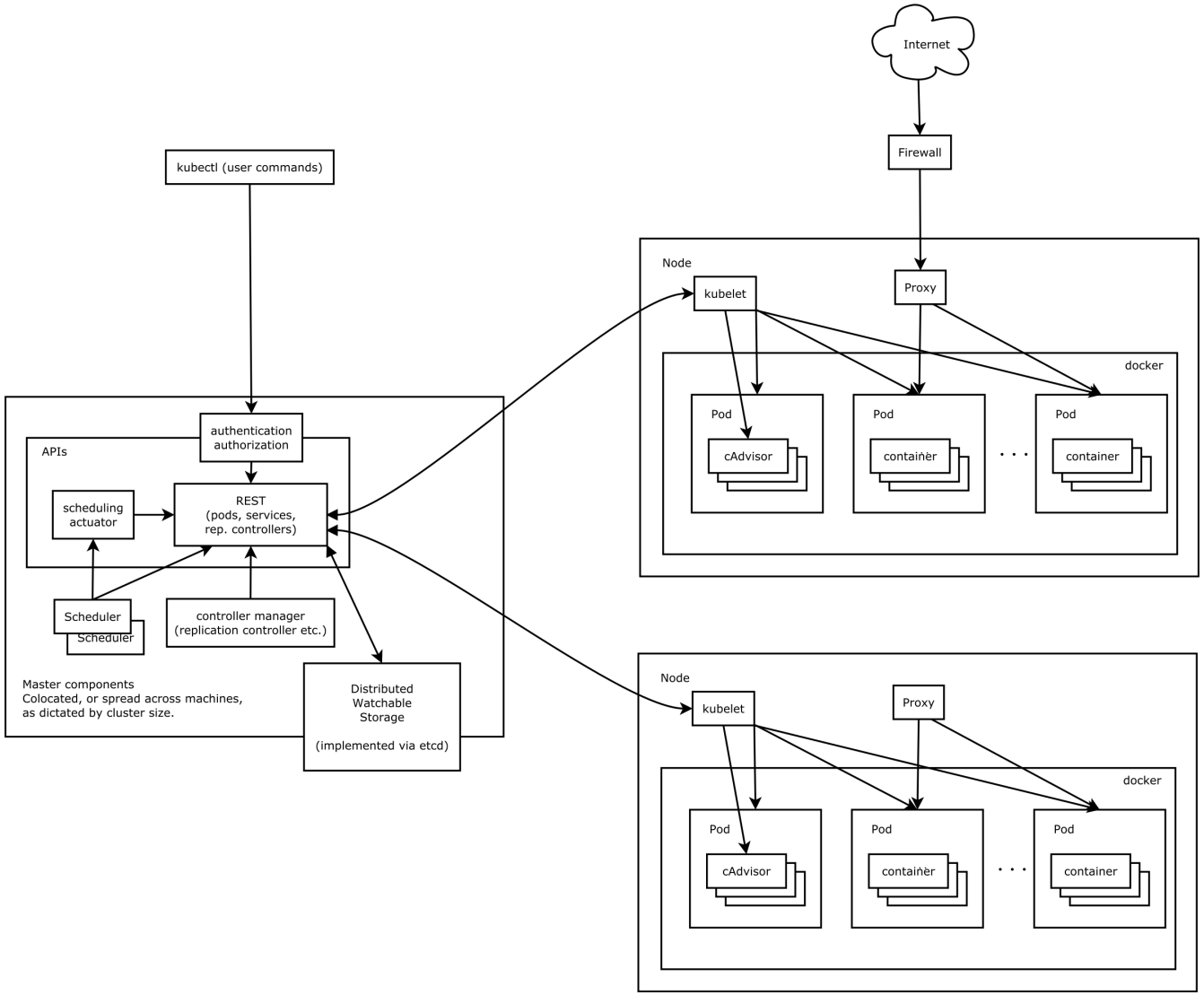
Kubernetes

- Kubernetes Introduction
- Kubernetes in Rancher

Brief Intro of Kubernetes

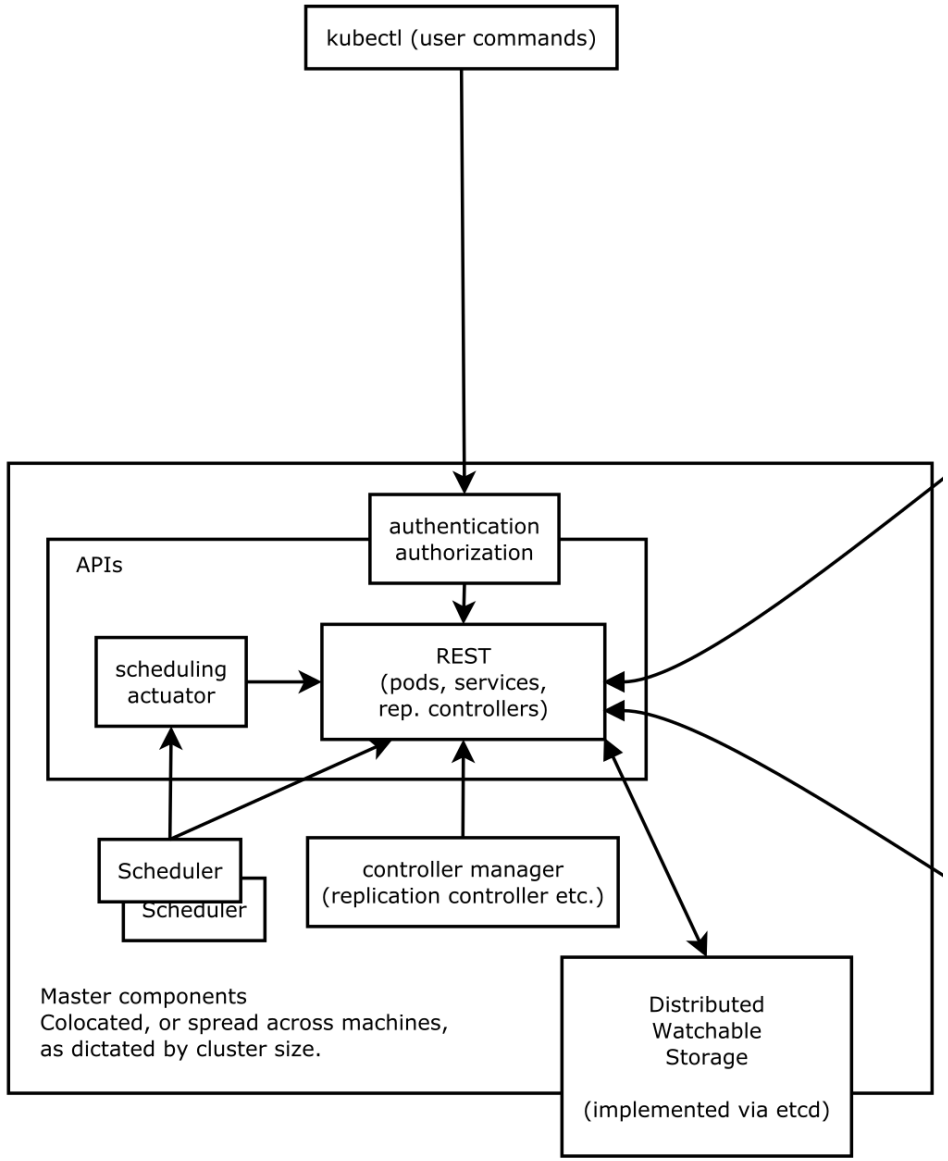


Kubernetes Architecture



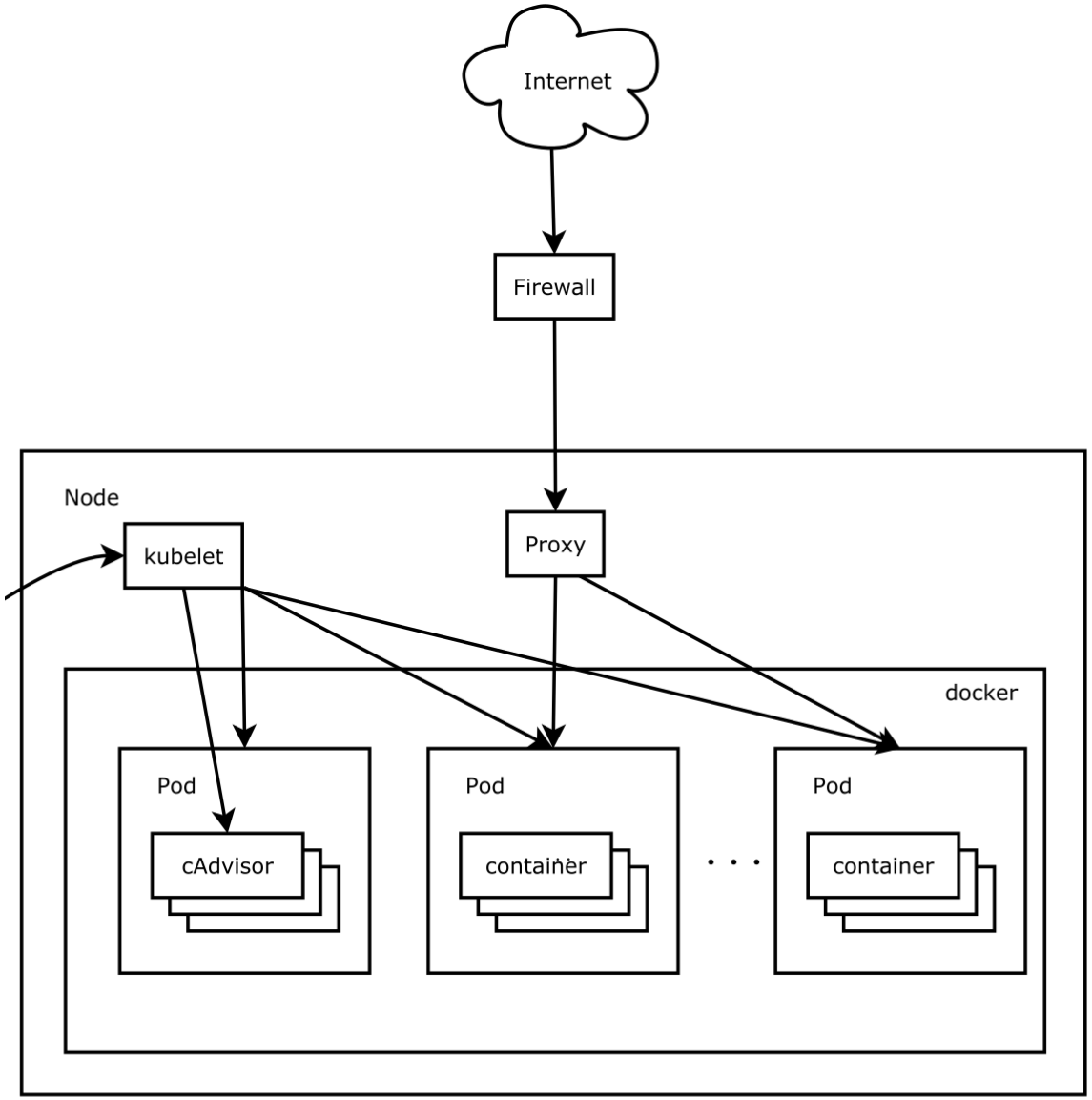
Components - Kubernetes Control Plane

- etcd
- API Server
- Scheduler
- Controller Manager



Components - Node

- kubelet
- kube-proxy



Key Concepts

- namespace: A namespace is like a prefix to the name of a resource. Namespaces help different projects, environments (e.g. dev and production), teams, or customers share the same cluster. It does this by preventing name collisions. Namespaces are a way to divide cluster resources between multiple uses (via resource quota).

```
kind: "Namespace"
apiVersion: "v1"
metadata:
  name: "development"
  labels:
    name: "development"
```

```
$ kubectl get namespaces
```

Key Concepts

- pods: Pods are a collocated group of application containers with shared volumes. The applications in the pod all use the same network namespace, IP address, and port space.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx-server
    image: nginx
    ports:
    - containerPort: 80
```

```
$ kubectl get pod
$ kubectl describe pod nginx
```

Key Concepts

- Replication controllers: manage the lifecycle of pods. They insure a specified number of specific pods are running at any given time. They do this by creating or deleting pods as required.

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: my-nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

```
$ kubectl get rc

$ kubectl scale --replicas=3 rc my-nginx
```

Key Concepts

- Replica Sets: Replica Set is the next-generation Replication Controller. Difference between a Replica Set and a Replication Controller right now is the selector support. Replica Set supports the new set-based selector requirements as described in the labels user guide whereas a Replication Controller only supports equality-based selector requirements.

Equality-based requirement

```
environment = production
tier != frontend
```

Set-based requirement

```
environment in (production, qa)
tier notin (frontend, backend)
partition
!partition
```

Key Concepts

- Services: provide a single stable name and address for a set of pods. They act as basic load balancers.
- Service information will be added as environment variable into pods created after service creation
- The Service's selector will be evaluated continuously and the results will be POSTed to an Endpoints

http://kubernetes.io/docs/user-guide/services/

```
apiVersion: v1
kind: Service
metadata:
  name: nginxsvc
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
```

```
$ kubectl get rc

$ kubectl scale --replicas=3 rc my-nginx
```

Key Concepts

- Labels: are used to organize and select groups of objects based on key-value pairs.

http://kubernetes.io/docs/user-guide/services/

```
apiVersion: v1
kind: Service
metadata:
  name: nginxsvc
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
```

```
$ kubectl get rc

$ kubectl scale --replicas=3 rc my-nginx
```

Key Concepts

•Service Type:

- ClusterIP: use a cluster-internal IP only - this is the default and is discussed above. Choosing this value means that you want this service to be reachable only from inside of the cluster.
- NodePort: on top of having a cluster-internal IP, expose the service on a port on each node of the cluster (the same port on each node). You'll be able to contact the service on any <NodeIP>:NodePort address.
- LoadBalancer: on top of having a cluster-internal IP and exposing service on a NodePort also, ask the cloud provider for a load balancer which forwards to the Service exposed as a <NodeIP>:NodePort for each Node.

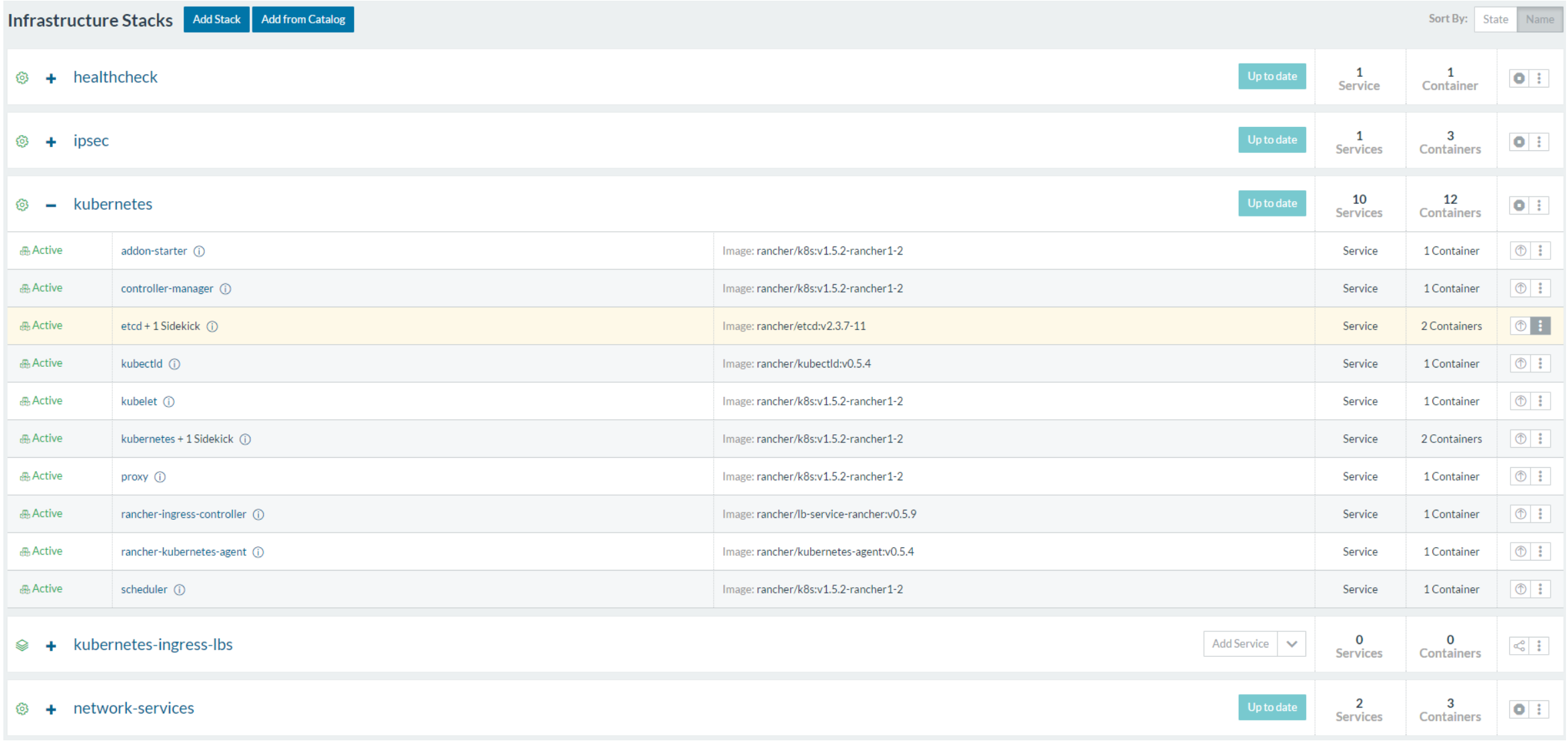
Key Concepts

- Volumes: A volume is a directory, possibly with some data in it, accessible to a container as part of its filesystem. Volumes are used used, for example, to store stateful app data.
- PersistentVolume (PV): is a piece of networked storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes, but have a lifecycle independent of any individual pod that uses the PV. This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

Key Concepts

- DaemonSet: A Daemon Set ensures that all (or some) nodes run a copy of a pod. As nodes are added to the cluster, pods are added to them. As nodes are removed from the cluster, those pods are garbage collected. Deleting a Daemon Set will clean up the pods it created.

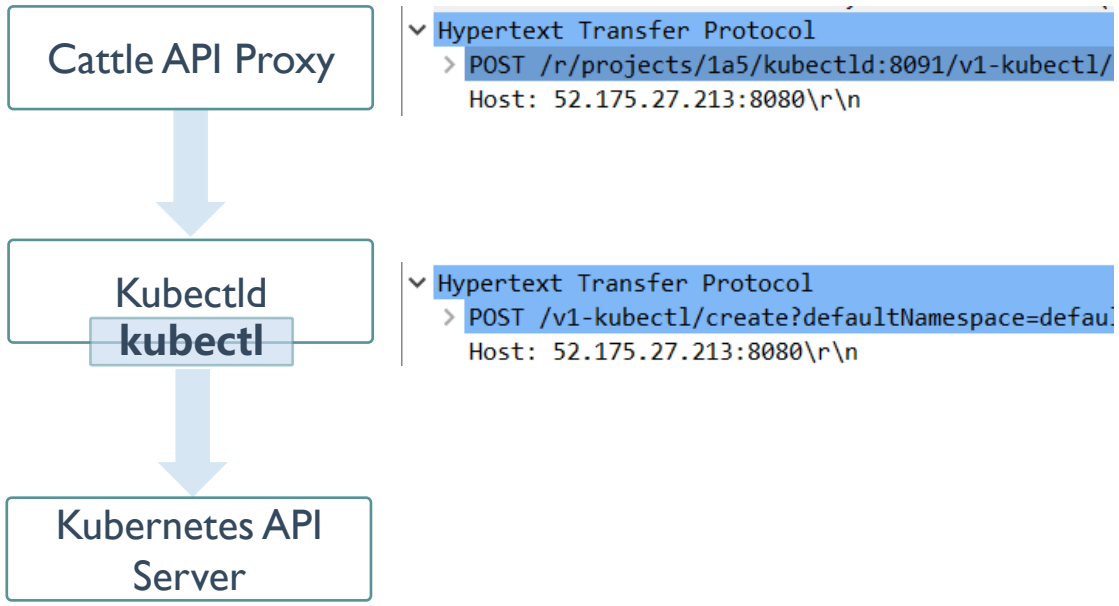
Kubernetes in Rancher

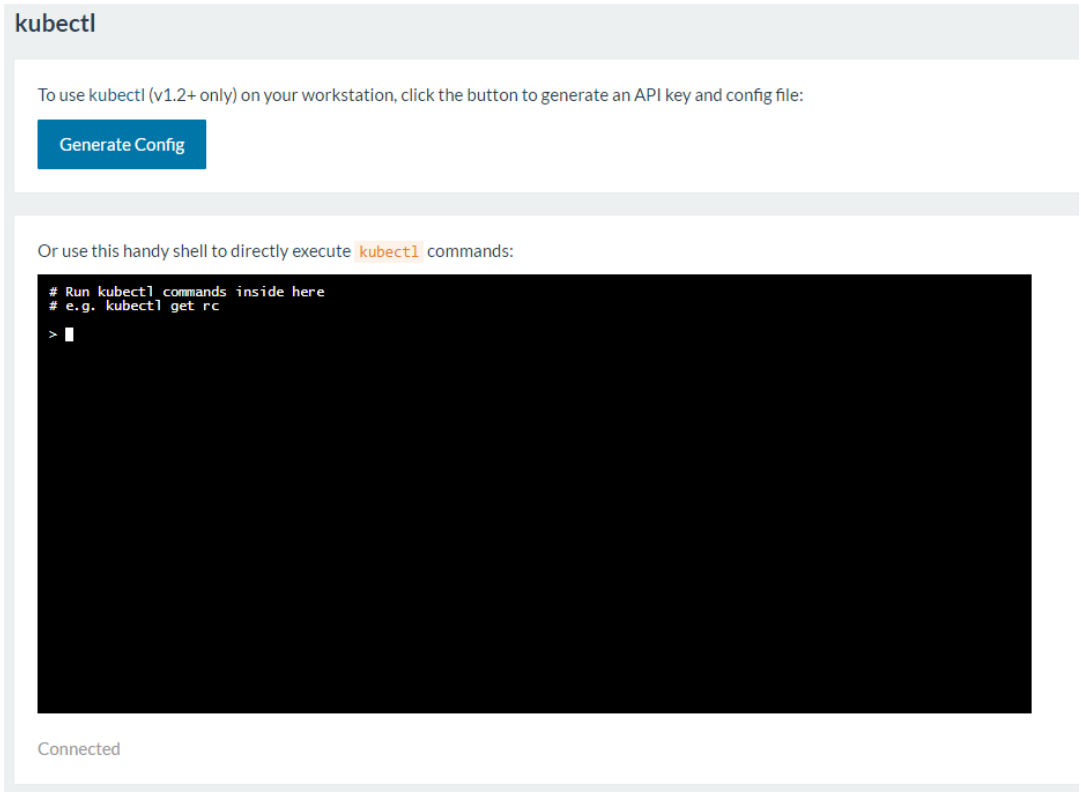


Kubectld

- kubectld –server=http://master –listen=:8091

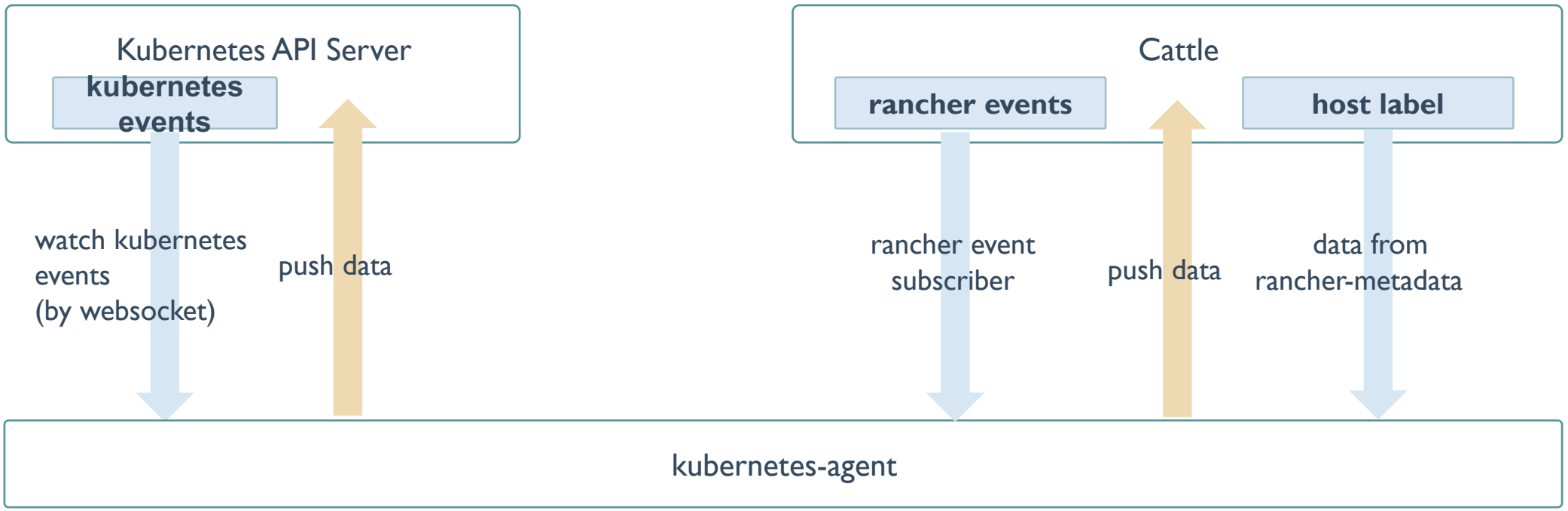
An embarrassingly simple microservice to expose `kubectl` `create/apply/get` logic





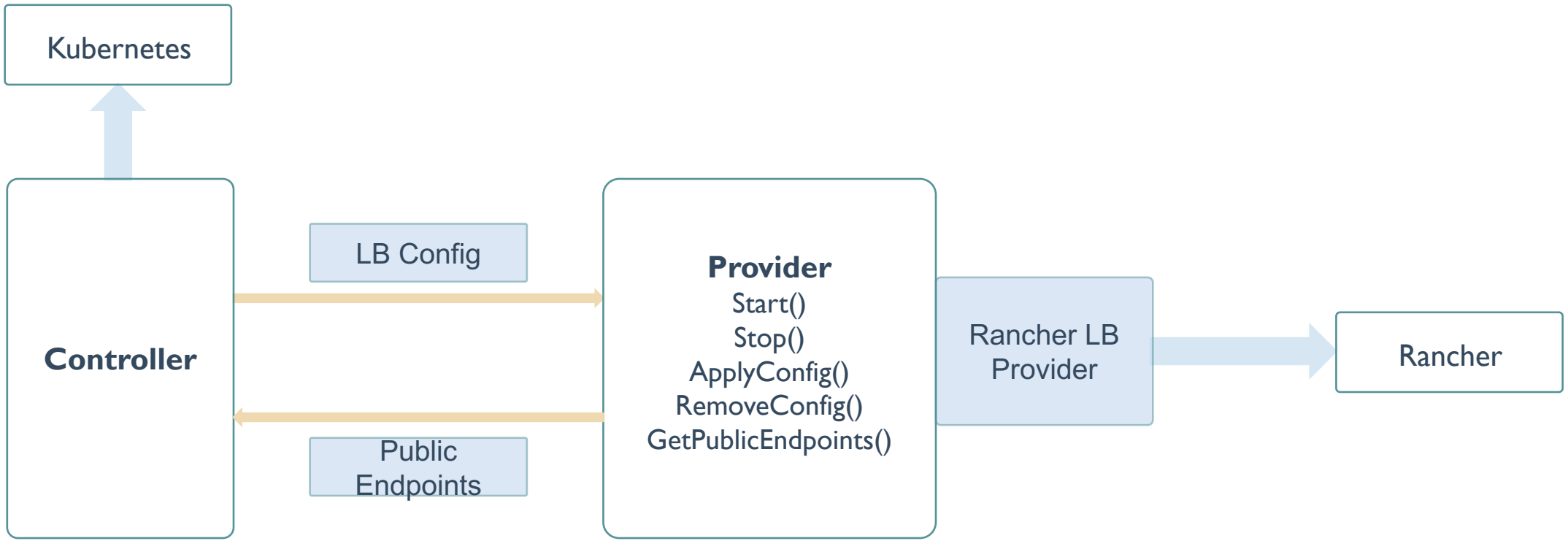
Kubernetes Agent

- Agent responsible for handling communications between Rancher and Kubernetes
-

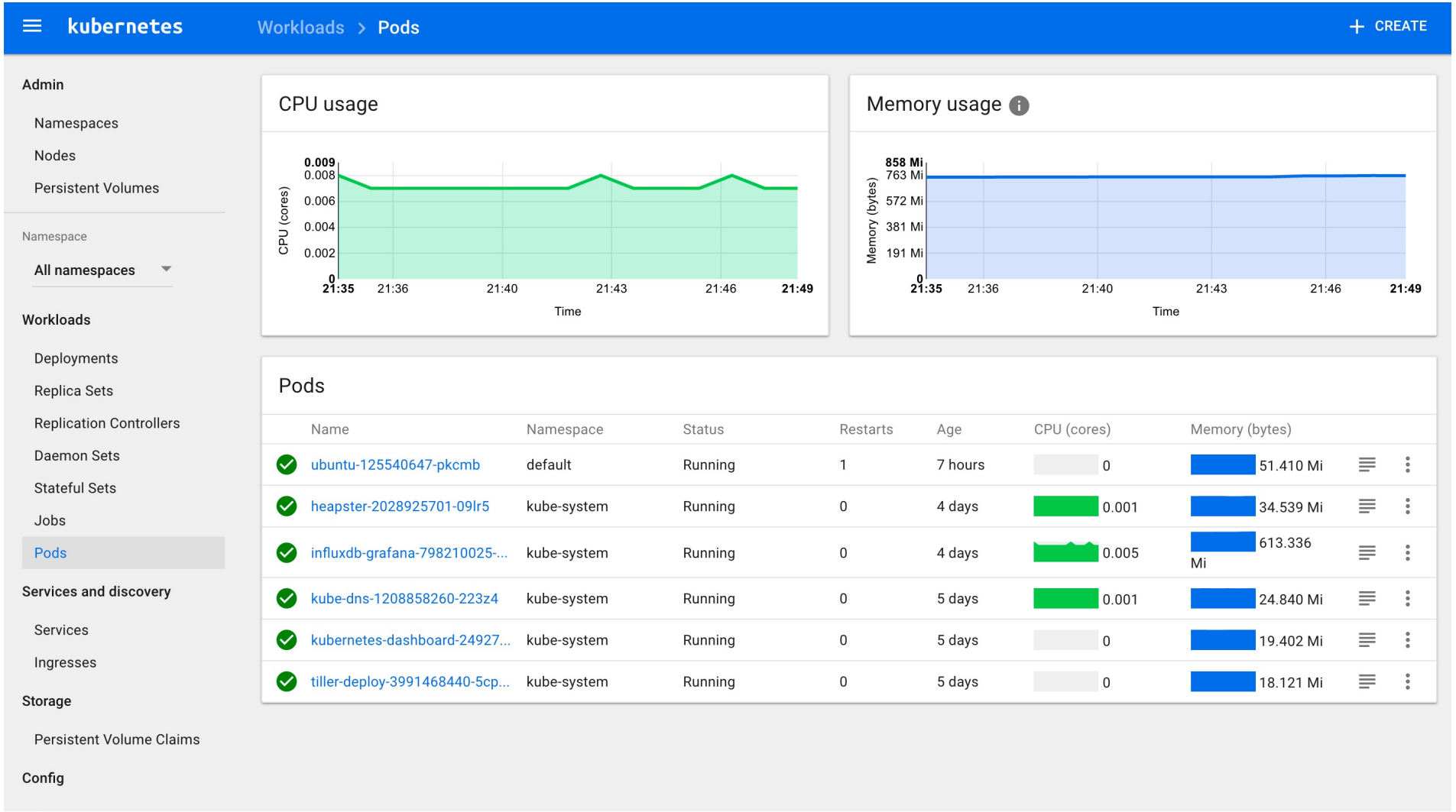


LB Controller

- L7 Load Balancer service managing load balancer provider configured via load balancer controller. Pluggable model allows different controller and provider implementation.



Using the Kubernetes Dashboard and Helm



Kubernetes Dashboard

- Web based Kubernetes control UI
- Deploy applications
- Provides overview of various Kubernetes resources
- Provides a log viewer for easy debugging

-

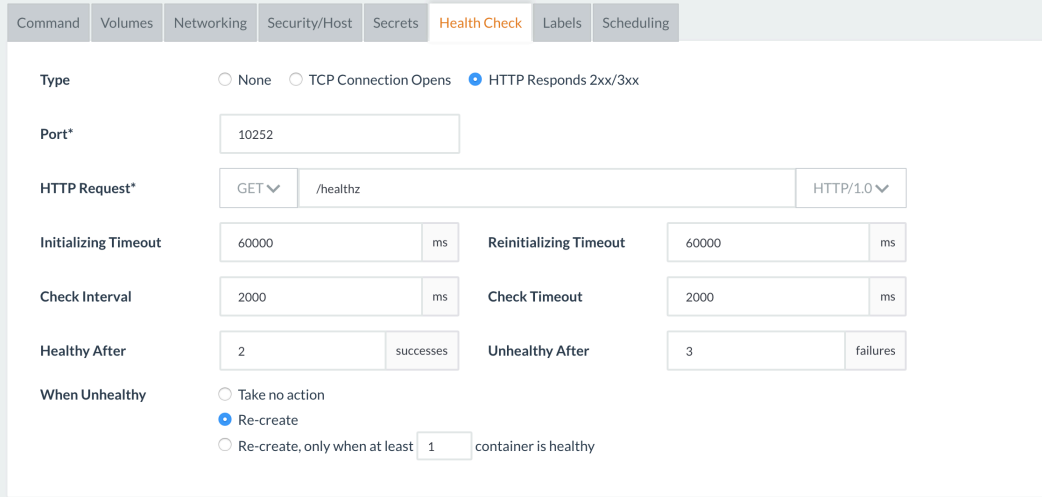
Kubernetes Helm

- Package manager for Kubernetes
- Supports private repositories
- Search for packages
- Configure and Install packages
- Delete packages

Kubernetes Master HA

rancher-compose.yml

```
version: '2'
services:
  controller-manager:
    scale: 1
    start_on_create: true
    health_check:
      healthy_threshold: 2
      response_timeout: 2000
      port: 10252
      unhealthy_threshold: 3
      initializing_timeout: 60000
      interval: 2000
      strategy: recreate
      request_line: GET /healthz HTTP/1.0
      reinitializing_timeout: 60000
  kubernetes:
```



Kubernetes deployment on Rancher

1. Modify Kubernetes Catalog

   Private registry:registry.cn-shenzhen.aliyuncs.com
   Image namespace for ADD-Ones:rancher_cn
   Image namespace for kubernetes-helm:rancher_cn
   Pod Infra Container Image:rancher_cn/pause-amd64:3.0

2. Demo

Rancher Kubernetes Engine (RKE)

2.0

COMPLETE CONTAINER MANAGEMENT PLATFORM

Application Management
User Interface • App Catalog • CI/CD • Monitoring • Logging

Kubernetes Management
Provisioning • Upgrades • RBAC • Policy • Security • Capacity • Cost

Rancher Kubernetes Engine (RKE)
AWS, vSphere, Bare metal

GKE

AKS

Rancher Kubernetes Engine (RKE)

- Simple,fast,Work anywhere
- Install/maintian/upgrade from a single cluster.yml file
- Easliy embeddable into third-party apps
- Support multiple network plugins

RKE Demo/Exercise

1. Use you custom image
2. Build a simple K8s Cluster, adding/removeing nodes
3. Build a HA master K8s Cluster

https://github.com/rancher/rke
http://rancher.com/an-introduction-to-rke
http://www.itdks.com/liveevent/detail/8343

RancherOS(zhibo)

Exercise 3

- Deploy a Kubernetes environment
- Try kubectl

Deploy a Kubernetes Environment

1. Create a new environment with K8s env template
2. Add host
3. Try to open Dashboard UI after system stack
4. Try Kubectl

Exercise 4

- Try RKE

Try RKE

1. Use you custom image
2. Build a simple K8s Cluster, adding/removeing nodes
3. Build a HA master K8s Cluster

http://www.itdks.com/liveevent/detail/8343