Kubernetes集群管理的轻松之道（下）

RANCHER

Rancher Labs • 2018-11

大纲

- 如何部署应用
- 认证方式
- 审计日志

如何部署应用

- Kubernetes应用相关的几个基本概念
  - Pod
  - Replicaset
  - Service
  - Configmap
  - Ingresses
  - 使用Rancher管理应用

Pod

- Kubenetes可以部署的最小单元
- 包含一个或多个可以容器
- 一个pod中的容器会调度到一起
- 每个pod有唯一的ip
- pod内的容器可以通过localhost沟通

Resplicaset

- 确保Pod以你指定的副本数运行，即如果有容器异常退出，会自动创建新的 Pod 来替代；而异常多出来的容器也会自动回收。

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        ports:
        - containerPort: 80
```

Services

- Service可以看作是一组提供相同服务的Pod对外的统一入口。
- 定义了可以访问一组pod的DNS入口
- 有不同类型：NodePort，ClusterIP，Loadbalancer

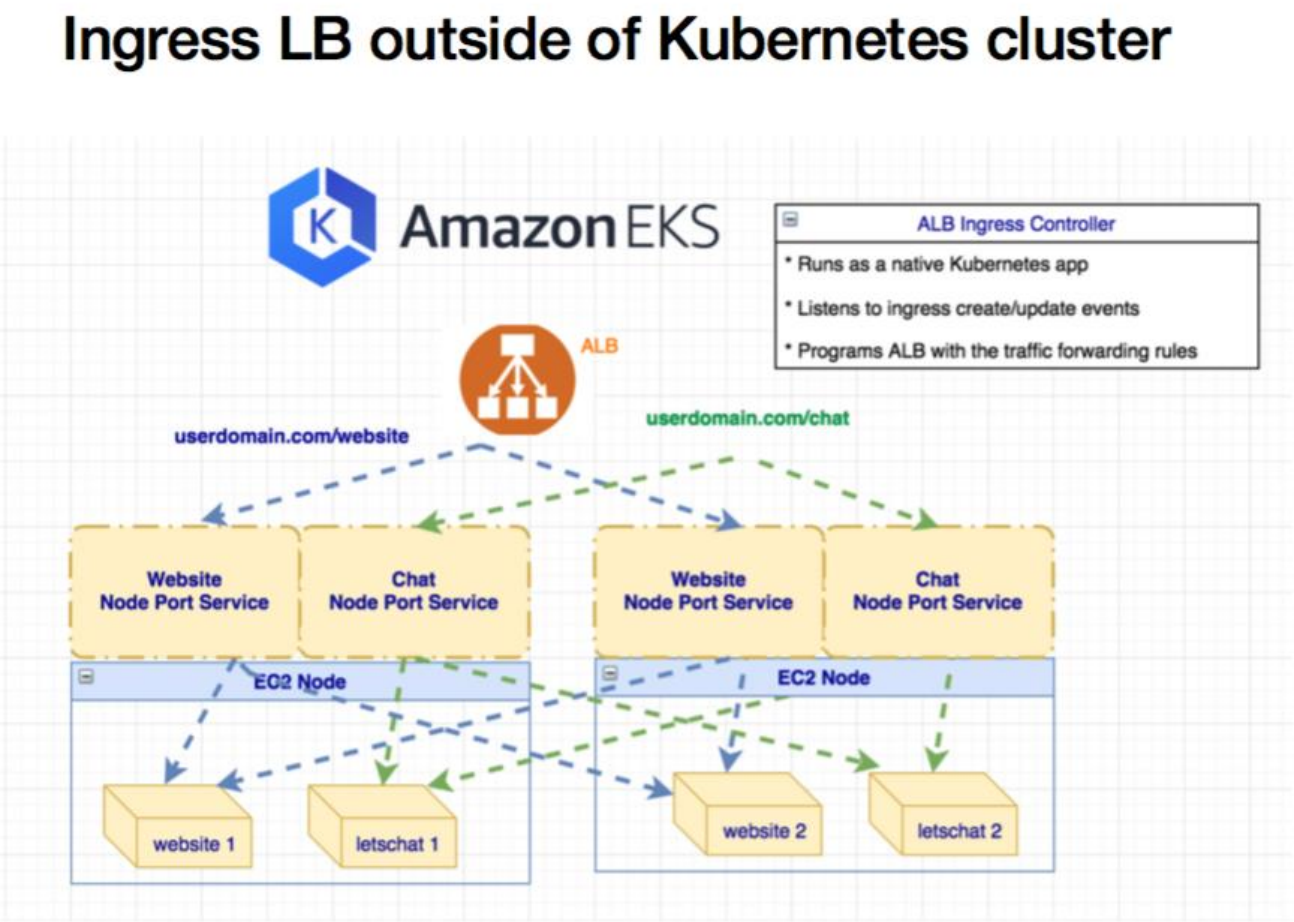
Configmap

- 保存配置或配置文件
- 在pod中可以作为环境变量或本地文件访问
- 解耦了pod spec的状态定义和应用的配置

Ingresses

- 定义Cluster外的流量怎么路由到Cluster中
- 用于暴露kubernetes service
- 可以通过host，path等属性进行路由
- 通常通过负载均衡实现，例如Nginx，HAProxy，AWS ELB 等

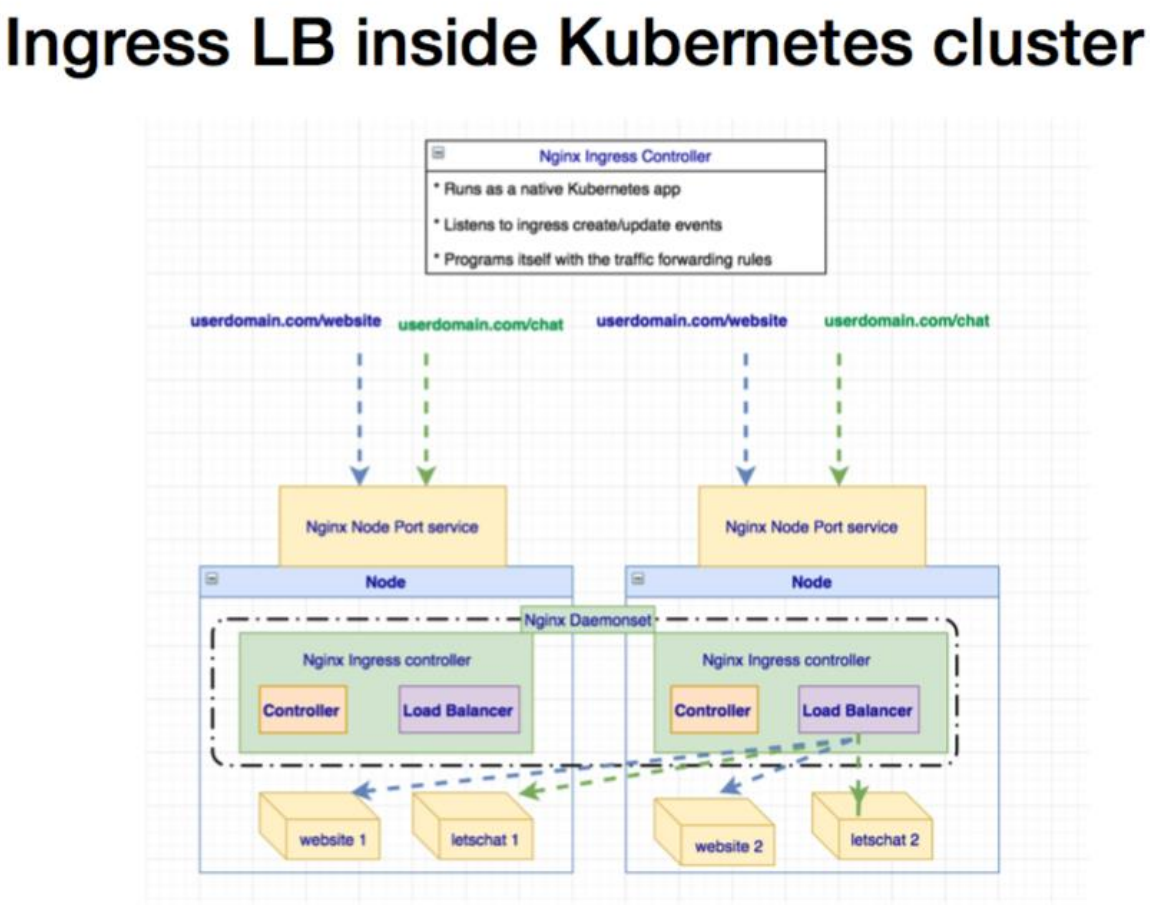
Ingress controllers for LBs deployed outside of Kubernetes cluster



Ingress controllers for LBs deployed outside of Kubernetes cluster

- Amazon ALB
  - https://github.com/kubernetes-sigs/aws-alb-ingress-controller
- Google GLBC
  - https://github.com/kubernetes/ingress-gce
- F5
  - https://github.com/F5Networks/k8s-bigip-ctlr
- Netscaler
  - https://github.com/citrix/kube-ingress-citrix-netscaler
- Openstack Octavia
  - https://github.com/kubernetes/cloud- provider-openstack/blob/master/docs/using-octavia-ingress- controller.md

Ingress controllers for LBs deployed inside of Kubernetes cluster



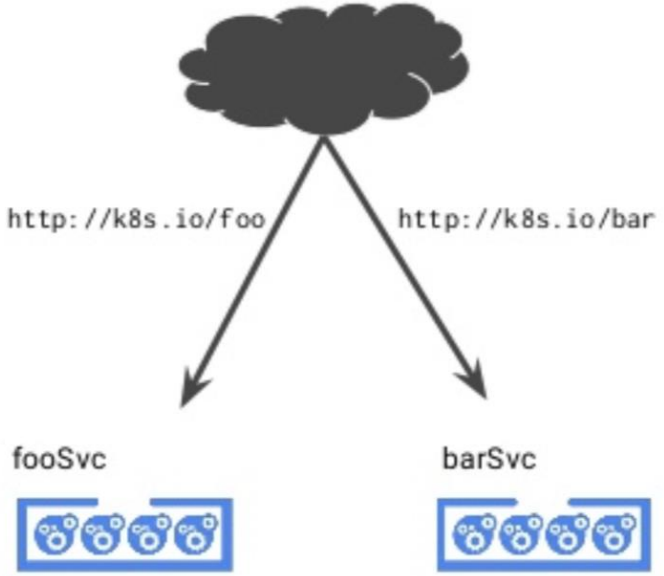
Ingress controllers for LBs deployed inside Kubernetes cluster

- Nginx
  - https://github.com/kubernetes/ingress-nginx
- Haproxy
  - https://github.com/jcmoraisjr/haproxy-ingress
- Traefik
  - https://github.com/containous/traefik/blob/master/docs/us er-guide/kubernetes.md
- Contour
  - https://github.com/heptio/contour

Ingresses

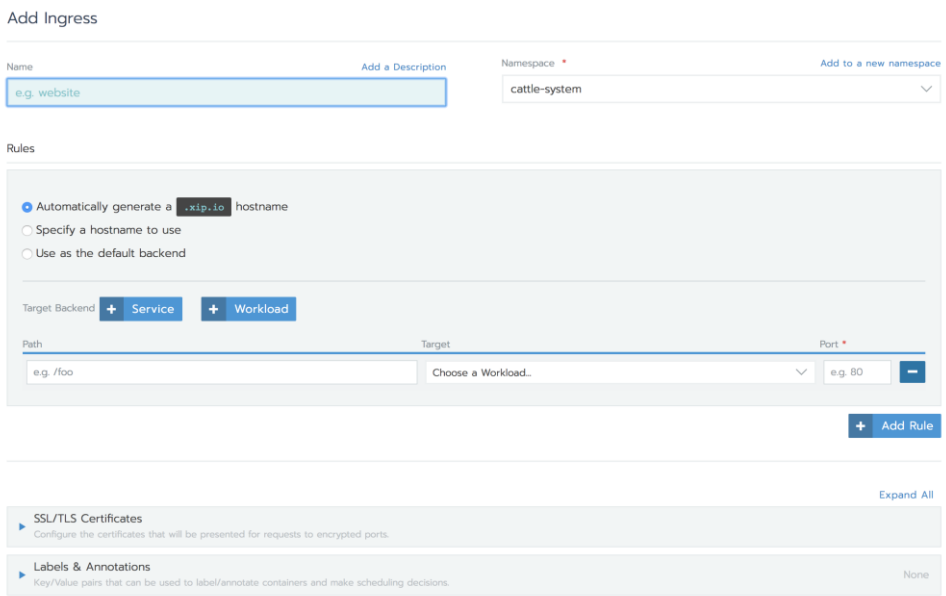
Ingress API

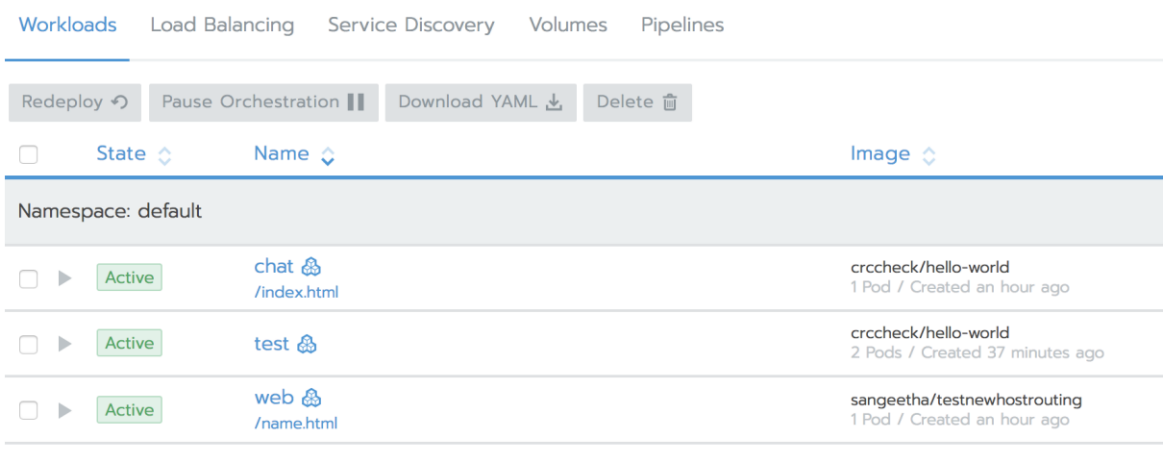
```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - path: /foo
        backend:
          serviceName: fooSvc
          servicePort: 80
      - path: /bar
        backend:
          serviceName: barSvc
          servicePort: 80
```



What Rancher do?

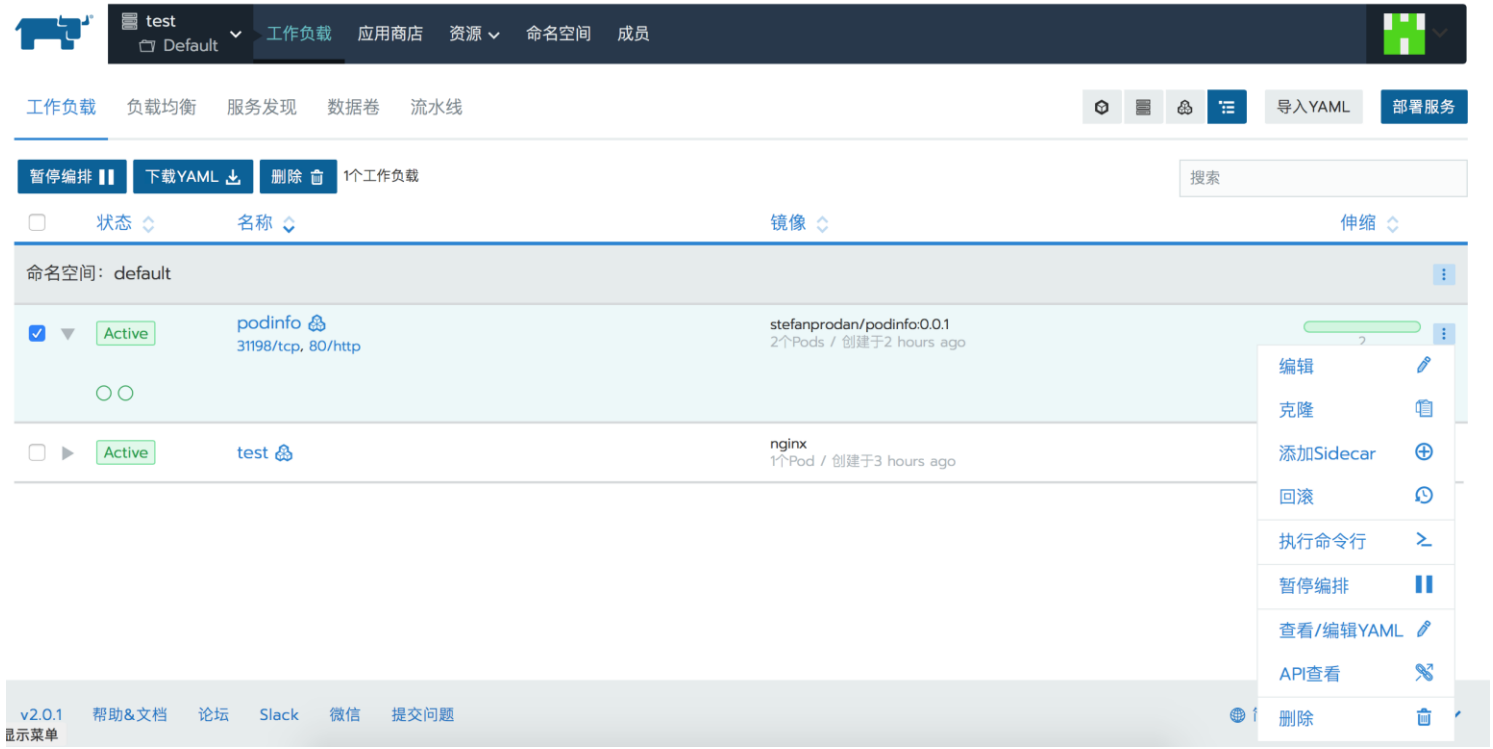
- 可以指定ingress对应的服务为workload或者service
- 支持在ingress中自动配置xip.io为hostname
- 支持的workload中根据开放的端口类型自动配置loadbalance
- 在workload中可以获取到配置的ingress，从而可以直接访问





应用生命周期管理

- Rancher容器云平台提供简单易用的图形化界面，用户可以通过图形界面便捷的进行容器应用部署。应用部署完成后，可以进行容器的全生命周期管理和各种操作，包括启动、暂停、删除、克隆容器等。



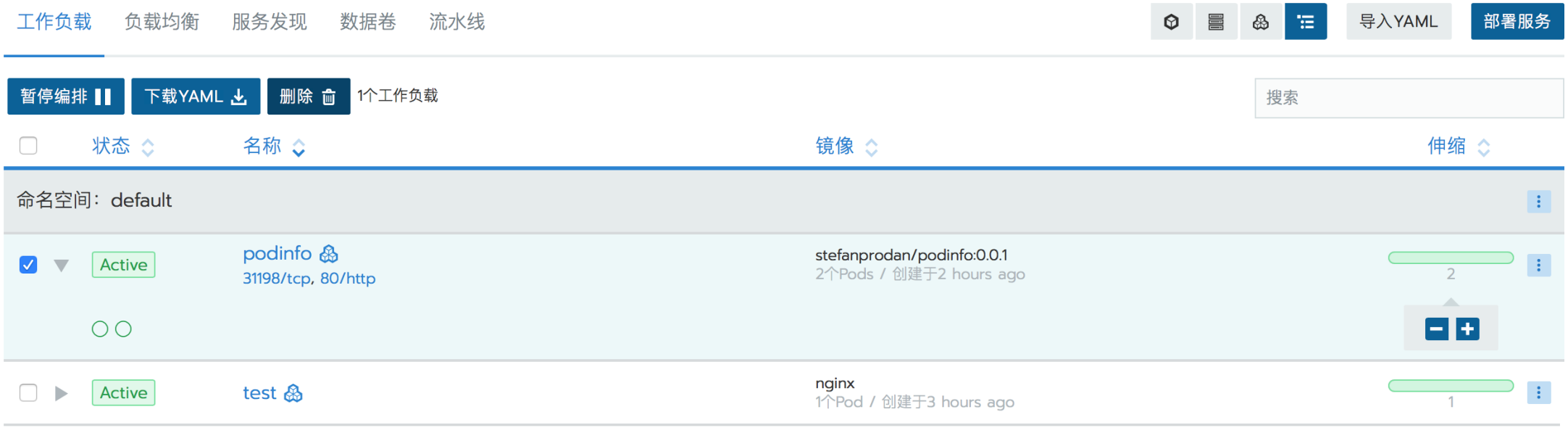
应用部署方式

1. Deployment
   - Deployment's pod is ephemeral
   - Easy to scale up and down
   - No desired context
2. StatefulSet
   - More like pets than cattle
   - Mostly using for persistent storage
   - Roll out in orders and implement rolling update
3. DaemonSet
4. CronJob/Job

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

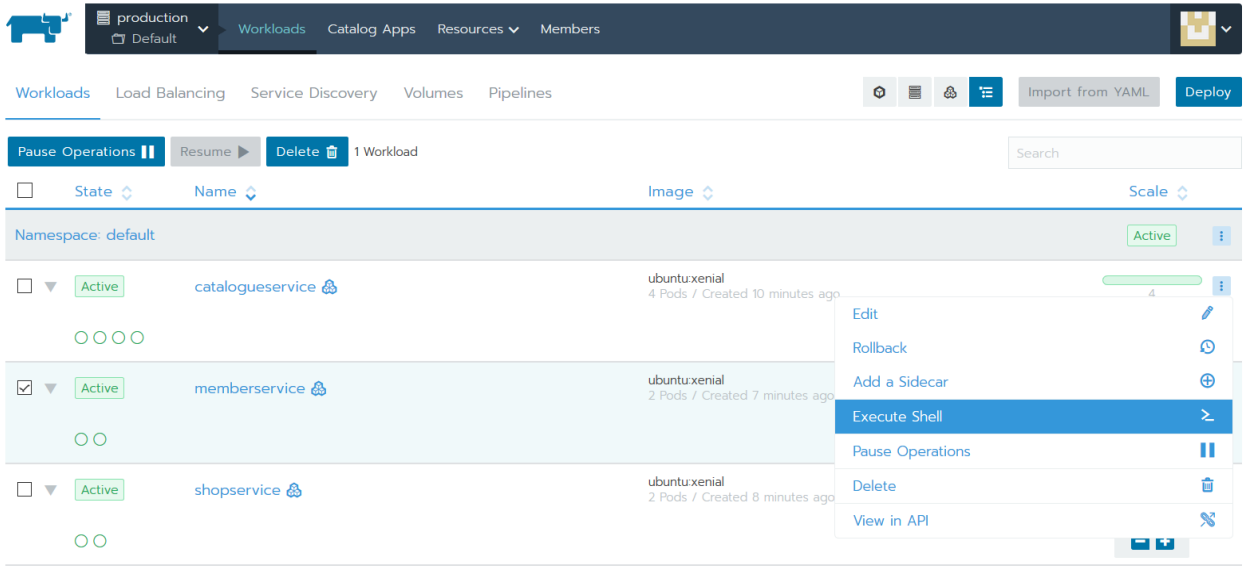
应用状态管理

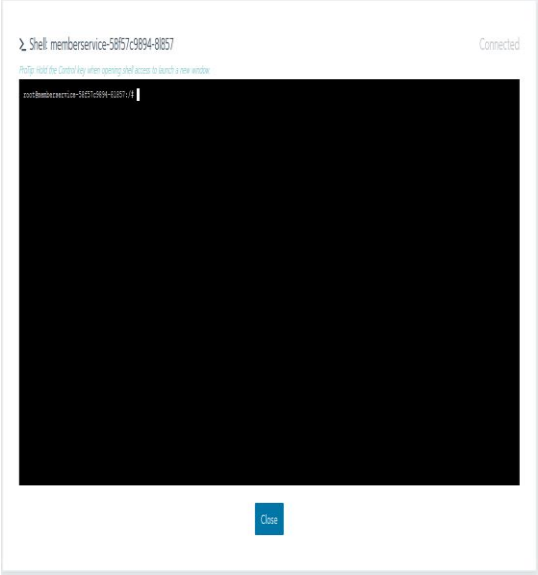
- 通过图形界面，用户可以便捷的查看容器的信息和运行状态，包括健康状态、容器名称、IP地址、主机、镜像和所在节点等信息。



容器控制台

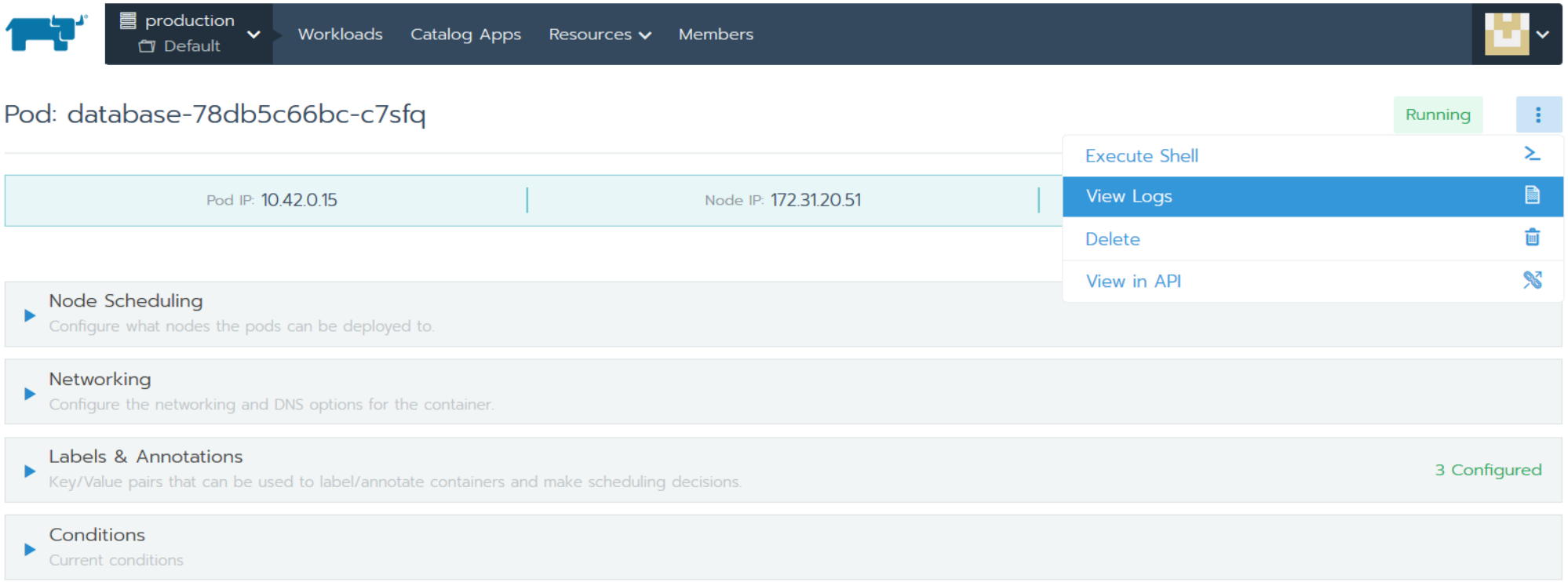
- Rancher容器云平台支持从图形界面直接访问容器Shell，从管理界面可以进入容器Shell支持命令，方便后台控制人员对容器进行相关操作。Rancher容器云平台支持从图形界面直接访问容器Shell，从管理界面可以进入容器Shell支持命令，方便后台控制人员对容器进行相关操作。





应用日志查看

- Rancher容器云平台支持从图形界面查看容器日志，日志支持实施刷新，方便管理和运维人员实时查看应用系统日志。



容器调度管理

- Rancher容器云平台支持多种调度策略，具体如下。
- 指定Node节点调度，容器云平台有三种方式指定 Pod 只运行在指定的 Node 节点上，包括：

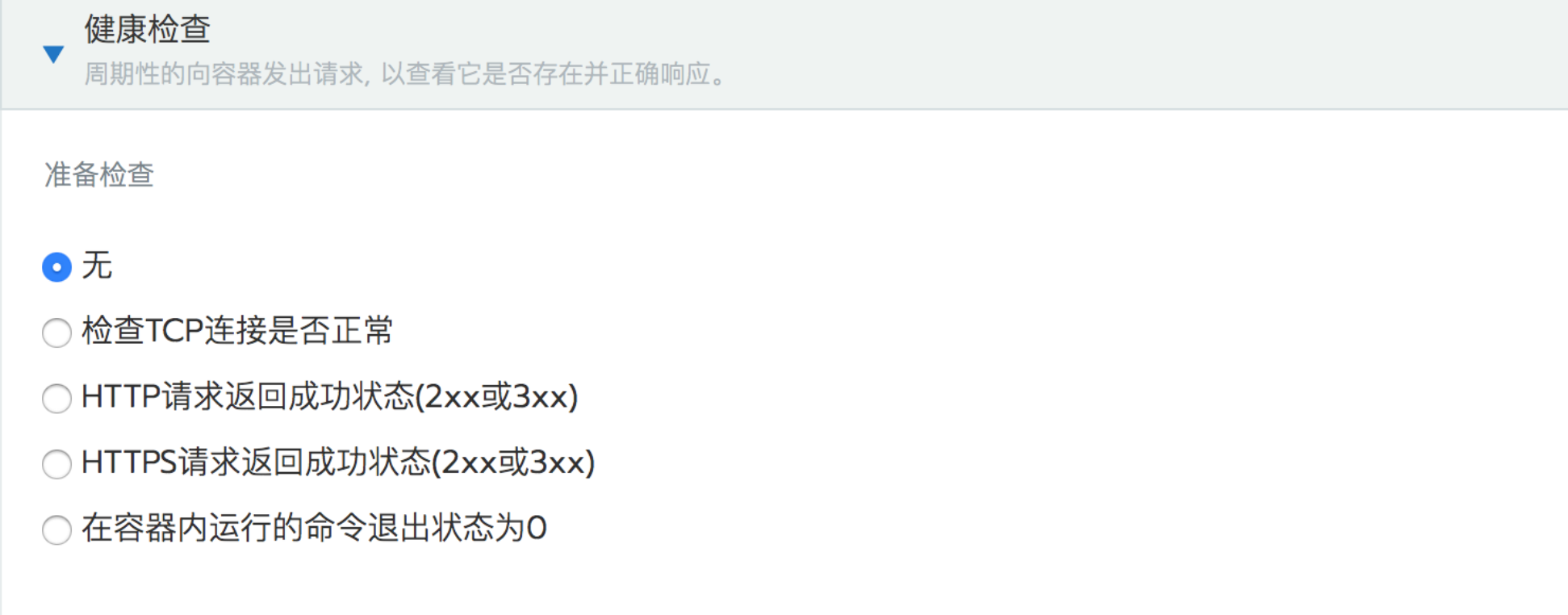
1. NodeSelector 只调度到匹配指定 label 的 Node 上

2. NodeAffinity 功能更丰富的 Node 选择器，比如支持集合操作

3. PodAffinity 调度到满足条件的 Pod 所在的 Node 上

应用健康检查

- Rancher容器云平台支持对部署的容器Pod进行健康检查并跟进检查结果进行相应操作，以确保应用处于健康运行状态。

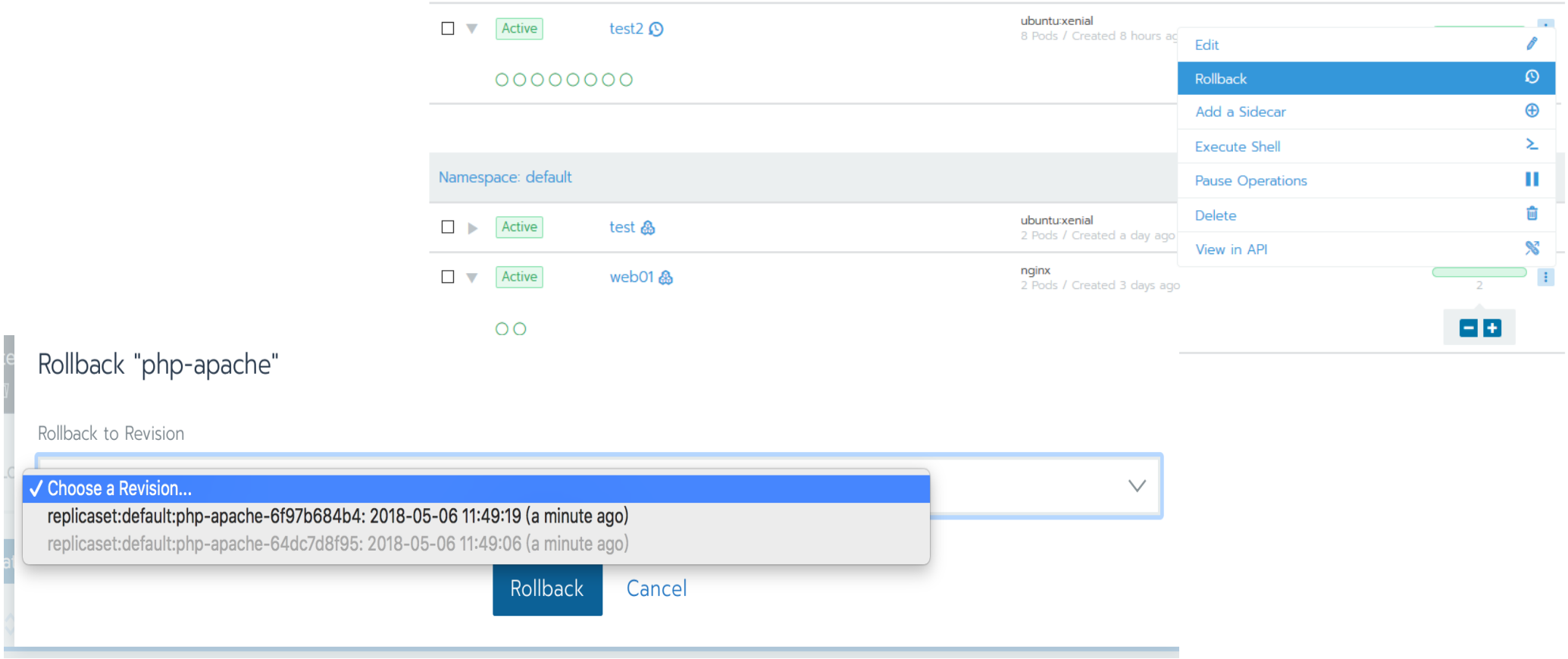


应用升级策略

- Rancher容器云平台支持多种升级策略
  - 滚动升级：启动新的，然后停止旧的
  - 滚动升级：停止旧的pod，然后开始新的pod
  - 杀掉所有旧的pod，然后启动新的pod

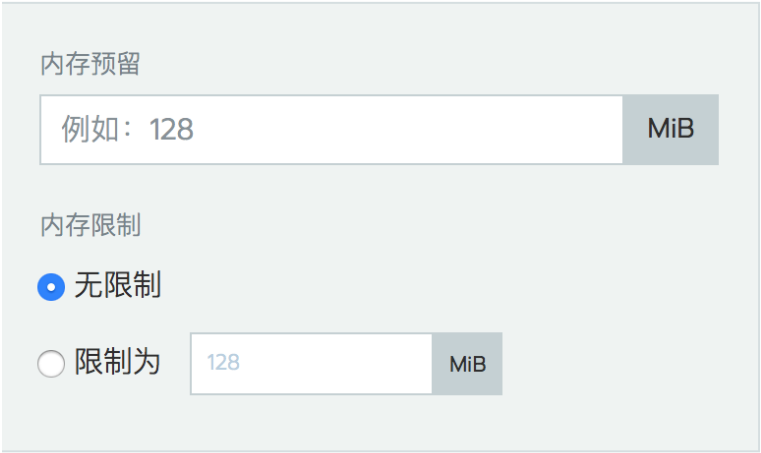
应用灰度发布

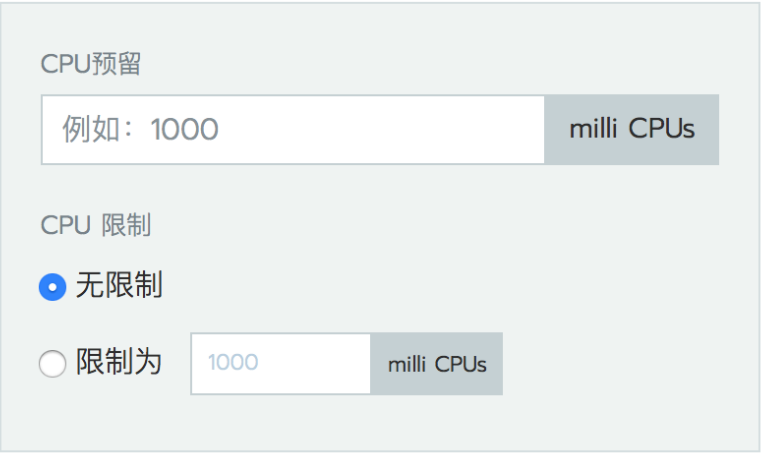
- 容器云支持灰度升级，升级过程中服务不中断。灰度升级阶段，可以点击升级成功或回滚到原来版本。

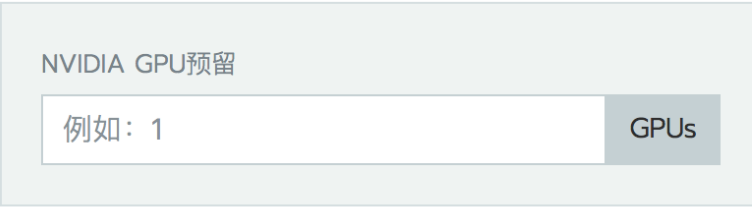


应用资源管理

- Rancher容器云平台支持对应用的资源管理，主要分为应用调度最小资源需求和应用运行过程中最大资源限制，限制的资源主要分为CPU、memory、GPU

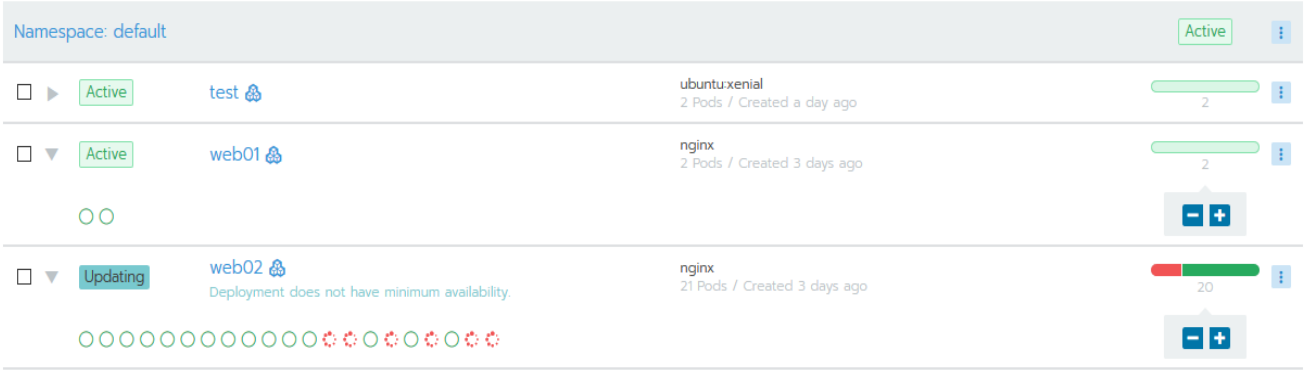


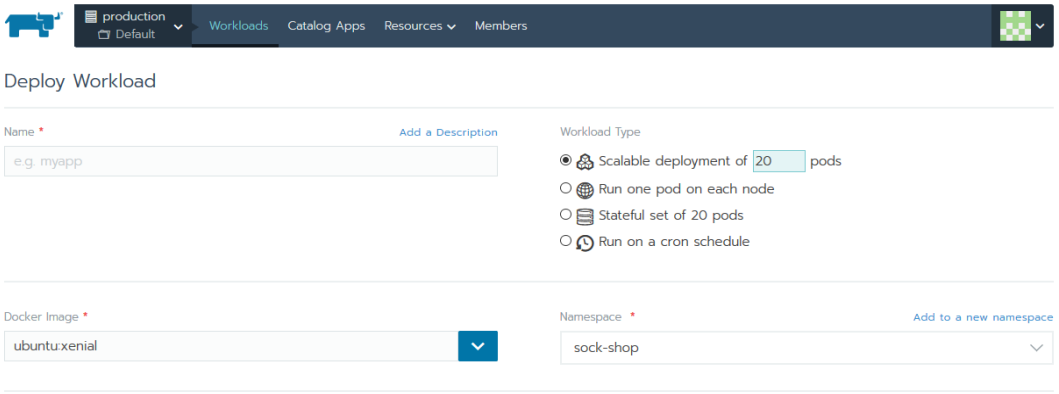




应用的高可靠

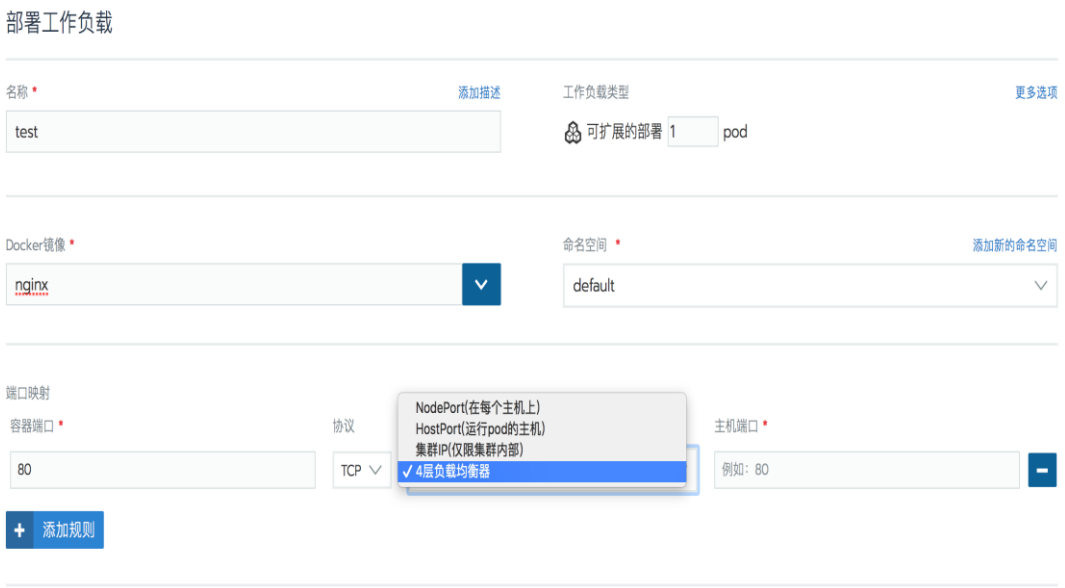
- 容器云部署应用时可以设定服务容器的需要保证的有效数量。当系统主机故障时，云平台的控制器可保证有效的的容器数量，在健康主机上重新创建容器。可以保证应用服务维持指定的容器数量，以保证服务正常。





Rancher2.0服务暴露方式

1. NodePort 在集群内每台宿主机上暴露
2. HostPort 在运行pod的节点上
3. ClusterIP 集群内部
4. LoadBlance 调用云厂商负载均衡器



部署一个应用

- Demo

Authentication and Authorization in k8s & Rancher

1. Kubernetes RBAC Authorization
2. Rancher2.0 多集群RBAC管理
3. Rancher 2.0 的角色管理

Kubernetes RBAC – role based authorization

1. 通过RBAC API控制 – Role, ClusterRole, RoleBinding, ClusterRoleBinding
2. Role
    1. 一系列的授权定义
    2. 通过添加更多的规则来定义操作更多资源的权限
3. Role Binding
    1. Binding User, Group, Service and Service Account

Kubernetes Role & ClusterRole

1. Role: 在namespace内生效
2. ClusterRole: 在集群内生效，不分Namespace
    1. Cluster Resources (e.g. Nodes)
    2. None-standard type of endpoint (e.g. “/health”)
    3. Resource like pod deployed in all namespaces

Role

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

```yaml
rules:
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["my-config"]
  verbs: ["get"]
```

Cluster Role

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

```yaml
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["get", "list", "watch"]
```

```yaml
rules:
- nonResourceURLs: ["/healthz", "/healthz/*"]
  verbs: ["get", "post"]
```

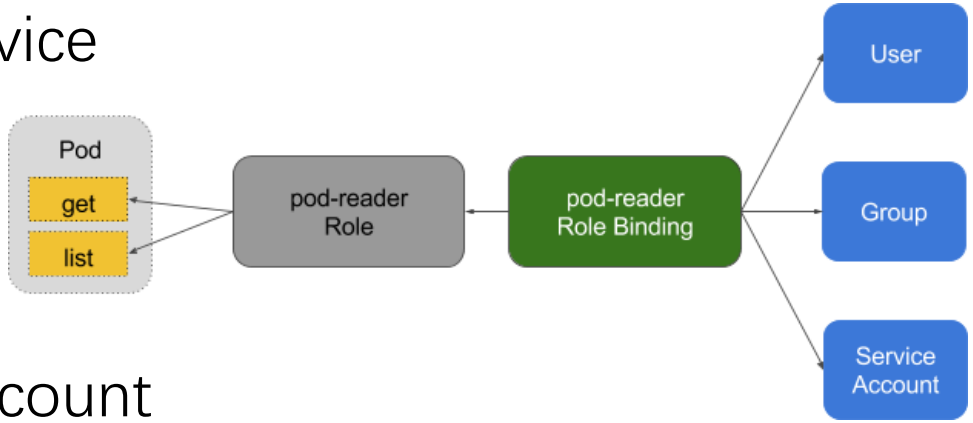
Role/ClusterRole Binding

```yaml
# This role binding allows "jane" to read pods in the "default" namespace.
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
```

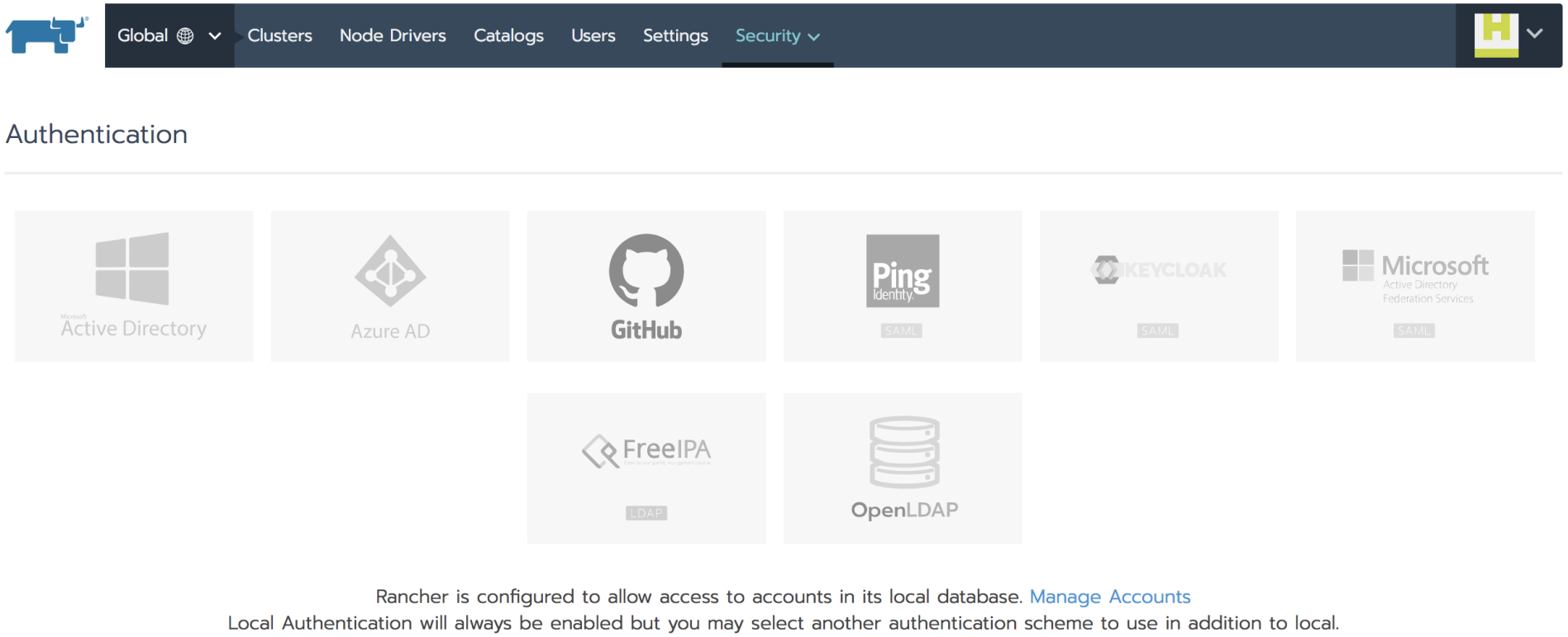
```yaml
# This cluster role binding allows anyone in the "manager" group to read secrets in any namespa
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Kubernetes Users

1. Normal Users
    1. Managed by other independent auth service
    2. Kubernetes didn't contains APIs for it
2. Service Accounts
    1. Managed by Kubernetes API's Service Account
    2. Bind to the namespace



Rancher 2.0 支持的认证方式

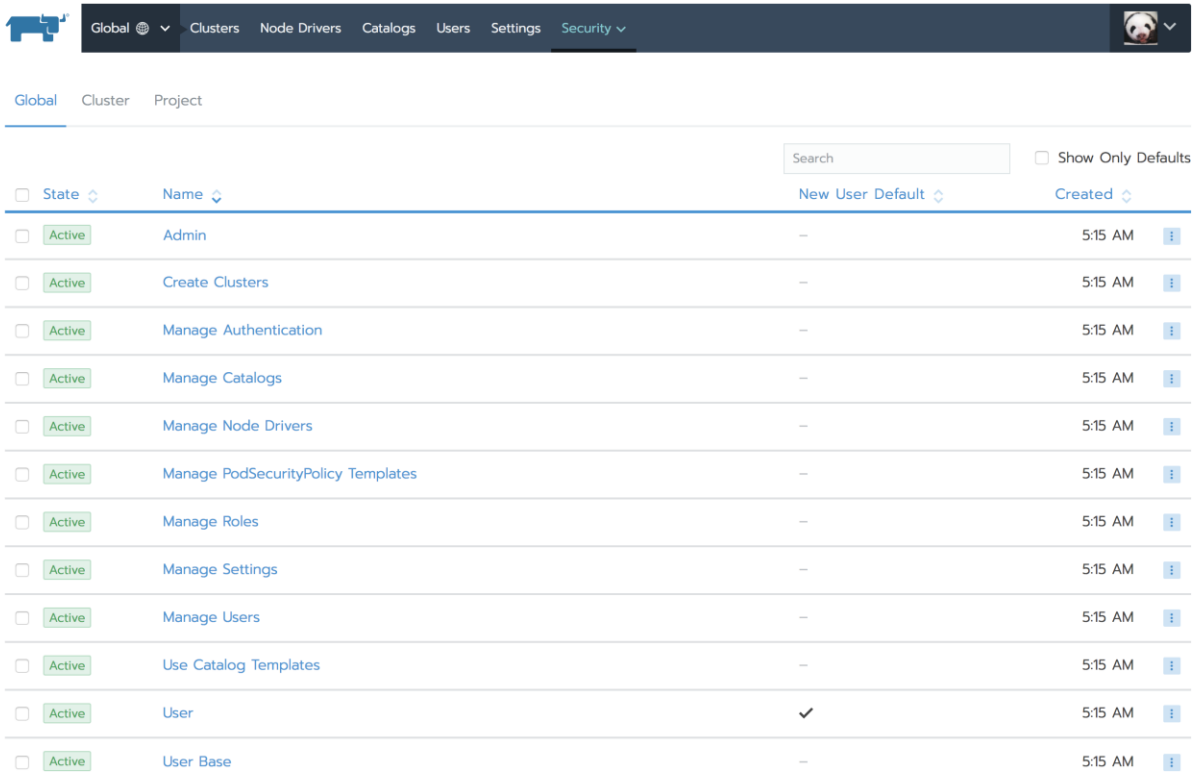


Rancher 2.0 RBAC

1. 在多集群管理中使用统一的认证系统
2. 使用Kubernetes RBAC
3. 通过Role+RoleBinding实现
4. 3个层级
    1. Global
    2. Cluster
    3. Project

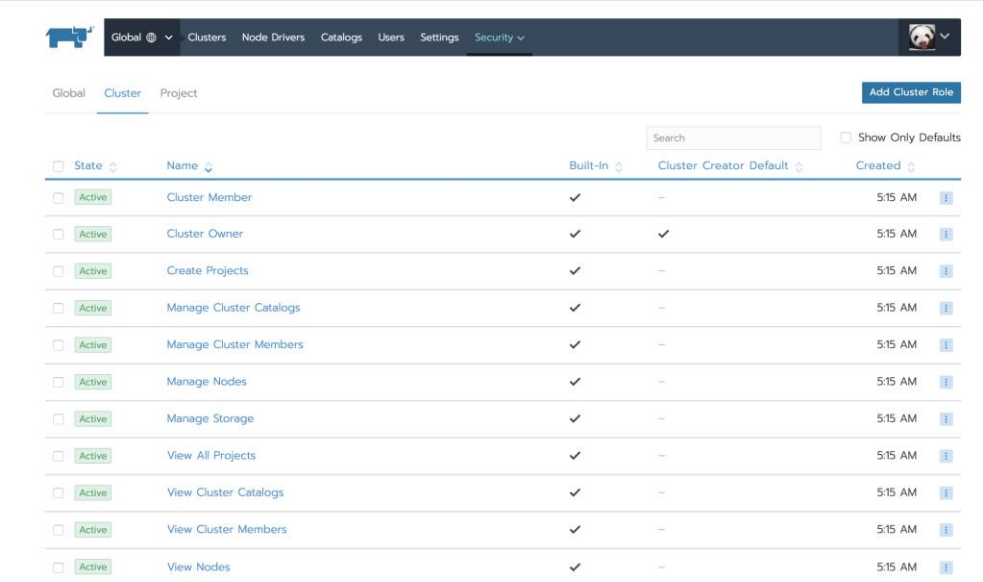
Global Level Authorization

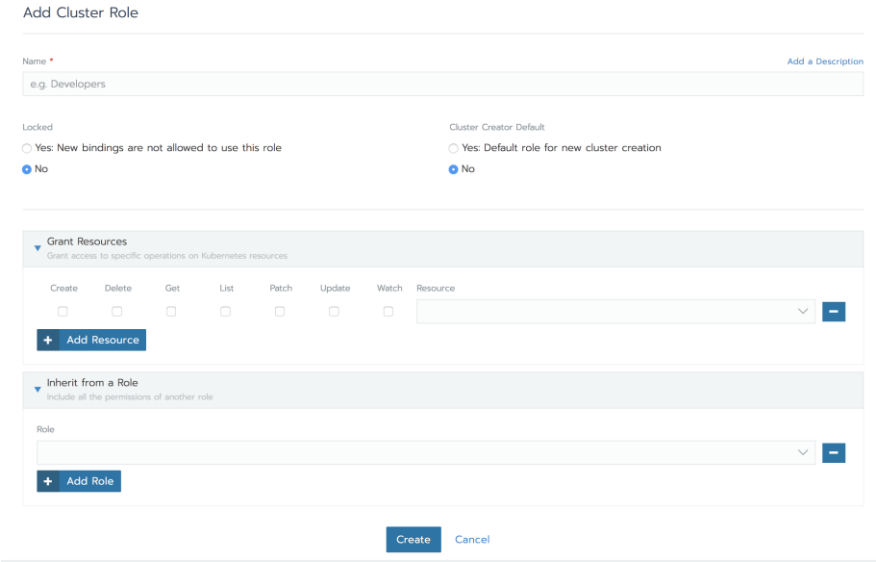
- 针对像Users, Authentication, etc.这样的全局资源
- 包含预定义的全局权限



Cluster Level Authentication

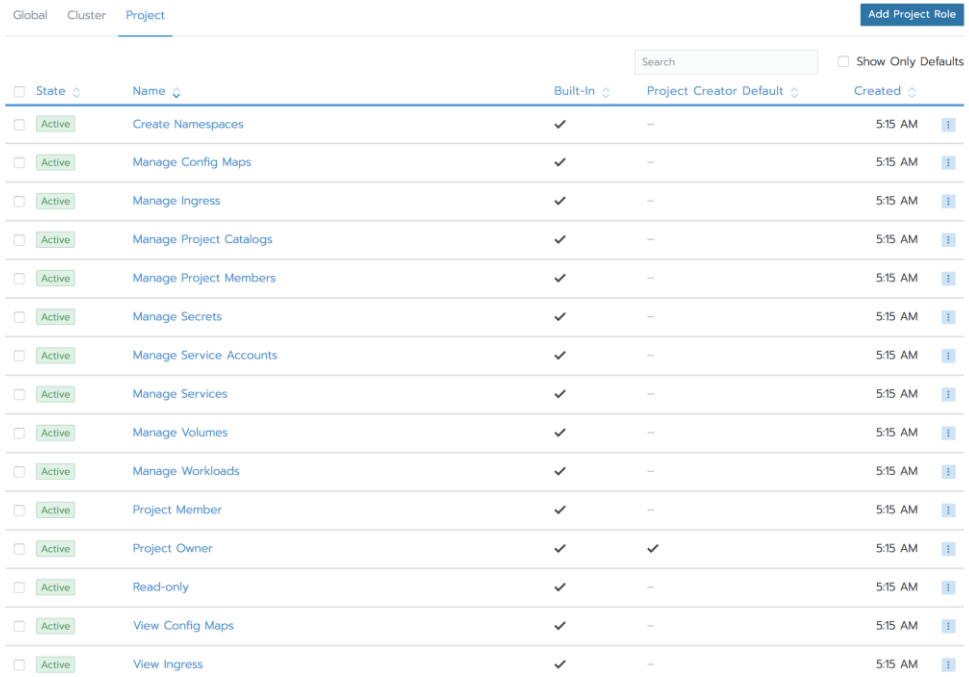
- 控制Cluster级别的资源
- 在实现上Cluster级别的权限控制通过ClusterRoleBinding来暴露求权限

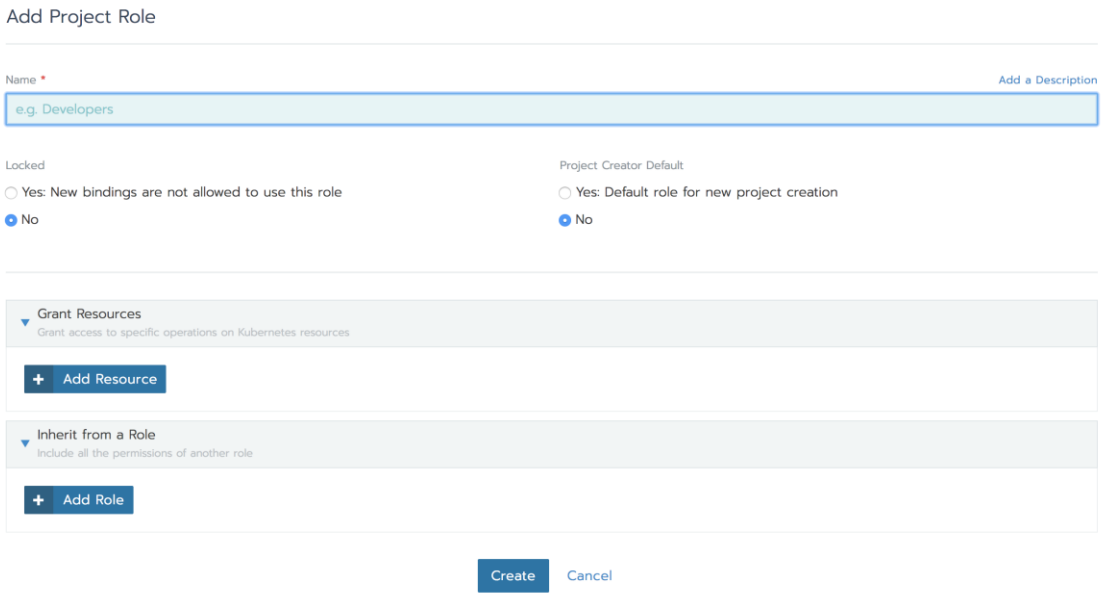




Project Level Authentication

- 控制Project级别的资源，比如application.
- Project级别通过Rolebinding来实现





Kubeconfig file with RBAC configurated

```yaml
users:
- name: "user-slzjw"
  user:
    token: "kubeconfig-user-slzjw:br2qkpvnbwk2mzljz5tc5bd8p7b9qj7r4tv879qtncjvvmf5rsw8z4"

contexts:
- name: "satomic"
  context:
    user: "user-slzjw"
    cluster: "satomic"

current-context: "satomic"
```

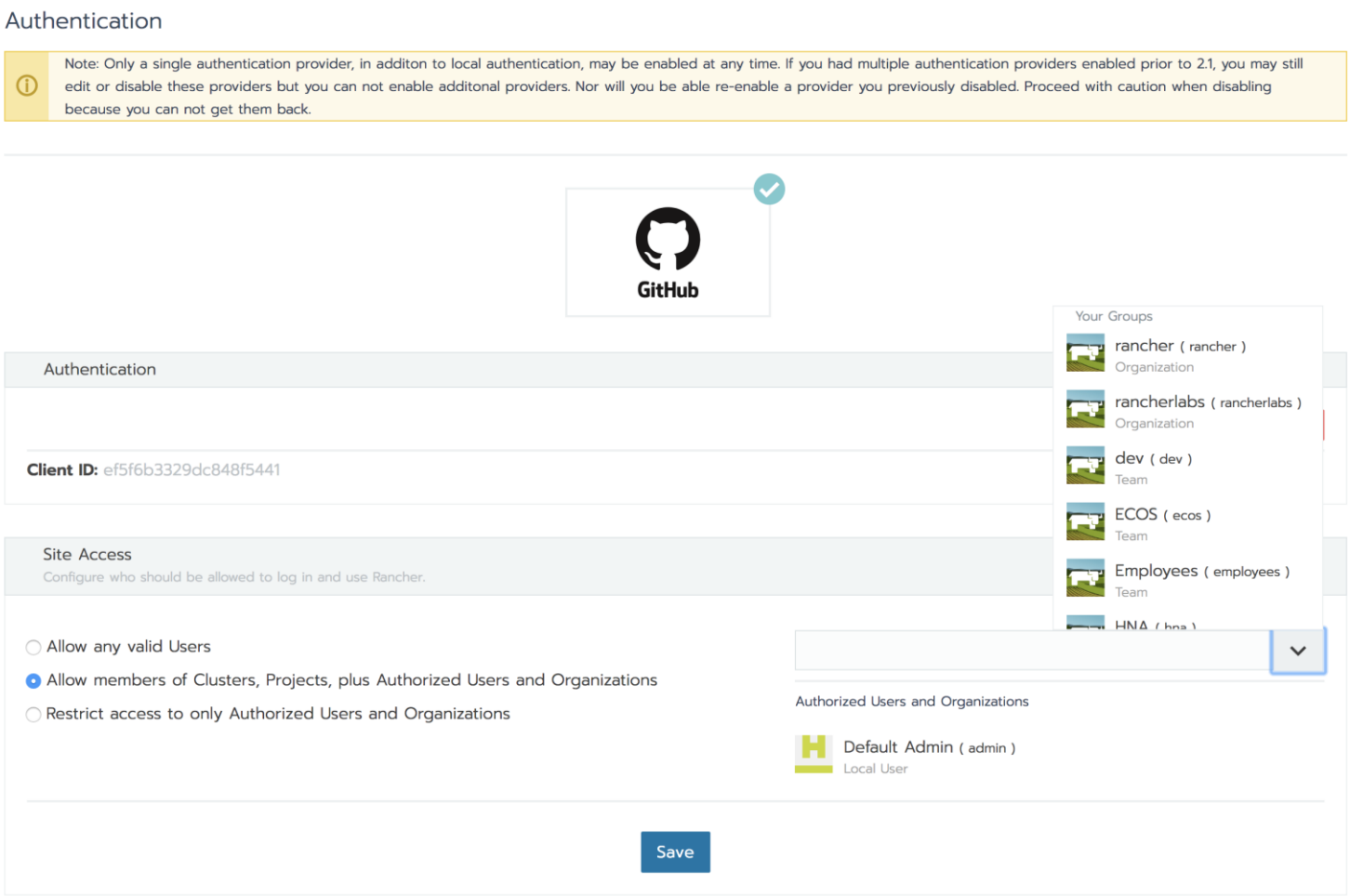

```
root@jenkins:~# kubectl get pods --all-namespaces
NAMESPACE          NAME                                      READY   STATUS    RESTARTS   AGE
cattle-alerting    alertmanager-649c5fc4d7-h7h9q             2/2     Running   4          4d
cattle-system      cattle-cluster-agent-55cb6f845-gd56v      1/1     Running   1          24d
cattle-system      cattle-node-agent-q9dl8                   1/1     Running   6          22d
cattle-system      cattle-node-agent-vgkmv                   1/1     Running   2          24d
default            nginx-668f465c85-jf9kg                    1/1     Running   0          1d
ingress-nginx      default-http-backend-564b9b6c5b-zs9rq     1/1     Running   6          24d
ingress-nginx      nginx-ingress-controller-dvhpv            1/1     Running   1          24d
```

```yaml
users:
- name: "u-p8kth"
  user:
    token: "kubeconfig-u-p8kth:hhqfvx5qh9n2kjbpz56ml89j2nxvm7mjdx7bmsm2qk74shxmqcdchh"

contexts:
- name: "satomic"
  context:
    user: "u-p8kth"
    cluster: "satomic"

current-context: "satomic"
```


```
root@jenkins:~# kubectl get service --kubeconfig kubeconfig-test.yaml -n test-ns
NAME      TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
nginx     NodePort   10.43.223.196   <none>        80:30808/TCP   5h
root@jenkins:~# kubectl get pods --kubeconfig kubeconfig-test.yaml
Error from server (Forbidden): pods is forbidden: User "u-p8kth" cannot list pods in the namespace "default"
root@jenkins:~# kubectl get service --kubeconfig kubeconfig-test.yaml
Error from server (Forbidden): services is forbidden: User "u-p8kth" cannot list services in the namespace "default"
```

如何启动Github登陆并配置权限

- Demo



Audit Log in Kubenetes && Rancher

- 开启Kubernete的Audit Log
- 开启Rancher的Audit Log

Audit Log in Kubernetes

- https://rancher.com/docs/rke/v0.1.x/en/example-yamls/

Audit Log in Rancher

- https://rancher.com/docs/rancher/v2.x/en/admin-settings/api-audit-log/
- configure
- AUDIT_LOG_PATH - Log path for Rancher Server API. Default path is /var/log/auditlog/rancher-api-audit.log
- AUDIT_LOG_MAXAGE - Defined the maximum number of days to retain old audit log files, default is 10 days.
- AUDIT_LOG_MAXBACKUP - Defines the maximum number of audit log files to retain, default is 10
- AUDIT_LOG_MAXSIZE - Defines the maximum size in megabytes of the audit log file before it gets rotated, default size is 100M
-
- AUDIT_LEVEL - 0 - disable audit log, 1 - log event metadata, 2 - log event metadata and request body, 3 - log event metadata, request body and response body
-

- Run Sample
- docker run -v /root/var/log/auditlog:/var/log/auditlog -e AUDIT_LEVEL=1 -e AUDIT_LOG_PATH=/var/log/auditlog/rancher-api-audit.log -e CATTLE_AGENT_IMAGE=rancher/rancher-agent:master -p 8080:80 -p 8443:443 micheliac/rancher:dev

Thank you