作者：李晓辉

联系方式：

1. 微信：Lxh_Chat

2. 邮箱：939958092@qq.com

Namespace 权限管理的复杂度高

Kubernetes 的 Namespace本来是想把集群里的资源分分组，逻辑上隔离一下。但用起来，真心有点小麻烦呢。

权限管理太复杂

权限这块，用 RBAC 配置，得手动一点点弄，稍不留神就容易出错，管理起来也挺费劲的。

资源配额控制不够智能

虽然可以用 ResourceQuota 来限制 Namespace 的资源，但没有好工具来监控和自动调整。要是管理多个 Namespace 的资源配额，那可真是够烦的。

缺少项目级别的管理功能

Namespace 主要就是资源隔离，但没法实现环境（开发、测试、生产）的自动化和版本控制，功能上有点捉襟见肘。

多租户管理不行

在多租户环境下，Namespace 没法给每个租户提供完全隔离的环境，共享集群资源很容易出安全和性能问题。

OpenShift Project 的优势

不过，OpenShift 的 Project 可就厉害了！它也是基于 Kubernetes Namespace，但加了不少新功能，把上面这些问题都解决了。

权限管理更简单

OpenShift 有个很直观的权限管理界面，管理员通过内置的角色和绑定机制，能轻松搞定用户和组的访问权限，再也不用手动折腾了。

资源配额管理自动化

OpenShift Project 可以让管理员为每个项目设置资源配额，而且有工具能监控和调整这些配额，资源分配更合理，管理起来也轻松多了。

项目级别的管理功能

OpenShift 项目能把应用、构建、部署、服务和配置资源都组织在一起，提供了一个更高级别的管理视图。这样就能更好地管理不同环境（开发、测试、生产）了。

多租户支持很强大

OpenShift 提供了强大的多租户管理功能，每个项目都能当一个独立的租户环境，资源、网络和安全都能完全隔离。这样一来，不同团队和应用就能在一个更安全的环境下运行啦。

项目概述

OpenShift 弄了点新东西，让命名空间更安全，用起来也更爽。它在 API 服务器里加了个 `Project` 资源类型。你要是想看看项目列表，API 服务器就会把命名空间列出来，但只会显示你有权看的那些，还会把这些命名空间包装成项目的样子返回给你。

更厉害的是，它还搞了个 `ProjectRequest` 资源类型。你提交创建项目请求的时候，OpenShift API 服务器就会根据模板来创建命名空间。这样一来，集群管理员就能通过模板来定制命名空间的创建了。比如，管理员可以确保新命名空间有特定的权限、资源配额或者限值范围。

这些功能简直就是命名空间的自助服务管理神器。集群管理员可以让用户自己创建命名空间，但又不用给用户修改命名空间元数据的权限。管理员还能定制命名空间的创建，确保它们符合组织的要求。

怎么规划项目模板?

你可以把任何命名空间资源加到项目模板里。比如，你可以加这些：

角色和角色绑定

把角色和角色绑定加到模板里，这样新项目里就能有特定的权限了。默认模板会给请求项目的用户 `admin` 角色，你可以保留这个，也可以改改，比如把 `admin` 角色给一组用户。你还可以加更精细的权限，比如对特定资源类型的权限。

资源配额和限值范围

把资源配额加到项目模板里，这样所有新项目都会有资源限制了。要是你加了资源配额，创建工作负载的时候就得明确声明资源限值。你还可以考虑加限值范围，这样创建工作负载的时候就能省点事。

就算所有命名空间都有配额，用户还是可以创建项目来往集群里加工作负载。要是碰到这种情况，可以考虑给集群加个集群资源配额。

网络政策

把网络政策加到模板里，这样就能按照组织的要求来实施网络隔离了。

创建项目模板

`oc adm create-bootstrap-project-template` 这个命令会输出一个模板，你可以用它来搞自己的项目模板。

这个模板跟 OpenShift 里默认的项目创建行为一模一样。它会自动加一个角色绑定，把新命名空间上的 `admin` 集群角色直接给请求项目的用户。这样一来，用户在创建项目的时候，就能直接变成管理员了，很方便对吧！

而且，这个模板还很灵活。你要是想加点别的东西，比如资源配额或者更细粒度的权限，都可以根据自己的需求来调整。这样一来，项目创建的时候就能直接把管理员权限和资源配额这些都搞定，再也不用一个个手动配置了，省心多了！

```
[student@workstation ~]$ oc adm create-bootstrap-project-template -o yaml > template.yml
[student@workstation ~]$ cat template.yml
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  creationTimestamp: null
  name: project-request
objects:
- apiVersion: project.openshift.io/v1
  kind: Project
  metadata:
    annotations:
      openshift.io/description: ${PROJECT_DESCRIPTION}
      openshift.io/display-name: ${PROJECT_DISPLAYNAME}
      openshift.io/requester: ${PROJECT_REQUESTING_USER}
    creationTimestamp: null
    name: ${PROJECT_NAME}
  spec: {}
  status: {}
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: null
    name: admin
    namespace: ${PROJECT_NAME}
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: admin
  subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: ${PROJECT_ADMIN_USER}
parameters:
- name: PROJECT_NAME
- name: PROJECT_DISPLAYNAME
- name: PROJECT_DESCRIPTION
- name: PROJECT_ADMIN_USER
- name: PROJECT_REQUESTING_USER
```

我们没有看到配额信息，只有rbac，我们来手工添加配额等信息，手工写配额信息会出错，所以先创建好之后，转成yaml，并复制到项目模板

```
[student@workstation ~]$ oc get limitranges lixiaohui-limit -o yaml
apiVersion: v1
kind: LimitRange
metadata:
  creationTimestamp: "2024-12-21T11:18:15Z"
  name: lixiaohui-limit
  namespace: default
  resourceVersion: "99495"
  uid: b275749e-eb52-47c2-a7ea-5fe5acd81157
spec:
  limits:
  - default:
      cpu: 500m
      memory: 512Mi
    defaultRequest:
      cpu: 250m
      memory: 256Mi
    max:
      cpu: "1"
      memory: 1Gi
    min:
      cpu: 125m
      memory: 128Mi
    type: Container
```

手工添加到参数上面就行

```yaml
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  creationTimestamp: null
  name: project-request
objects:
- apiVersion: project.openshift.io/v1
  kind: Project
  metadata:
    annotations:
      openshift.io/description: ${PROJECT_DESCRIPTION}
      openshift.io/display-name: ${PROJECT_DISPLAYNAME}
      openshift.io/requester: ${PROJECT_REQUESTING_USER}
    creationTimestamp: null
    name: ${PROJECT_NAME}
  spec: {}
  status: {}
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: null
    name: admin
    namespace: ${PROJECT_NAME}
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: admin
  subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: ${PROJECT_ADMIN_USER}
- apiVersion: v1
  kind: LimitRange
  metadata:
    name: lixiaohui-limit
    namespace: ${PROJECT_NAME} # 这里手工写了变量
  spec:
    limits:
    - default:
        cpu: 500m
        memory: 512Mi
      defaultRequest:
        cpu: 250m
        memory: 256Mi
      max:
        cpu: "1"
        memory: 1Gi
```

```
      min:
        cpu: 125m
        memory: 128Mi
      type: Container
parameters:
- name: PROJECT_NAME
- name: PROJECT_DISPLAYNAME
- name: PROJECT_DESCRIPTION
- name: PROJECT_ADMIN_USER
- name: PROJECT_REQUESTING_USER
```

改到满意之后，用下面的方法来完成创建

```
[student@workstation ~]$ oc create -f template.yml -n openshift-config
template.template.openshift.io/project-request created

[student@workstation ~]$ oc get template -n openshift-config
NAME              DESCRIPTION   PARAMETERS    OBJECTS
project-request                 5 (5 blank)   3
```

我们来更新一下集群资源，让集群在创建新的project的时候，用我们的模板

```
[student@workstation ~]$ oc edit projects.config.openshift.io cluster
...
spec: # 下面是添加的内容
  projectRequestTemplate:
    name: project-request
```

这个一旦更新，我们的apiserver会重新生成，可能没那么快，慢慢等待

```
[student@workstation ~]$ oc get co
openshift-apiserver                        4.14.0    False       False         False      3
```

等上面的更新完成后，我们创建一个project，然后进去创建一个deployment，看看是否会自带我们的限额

```
[student@workstation ~]$ oc new-project hello
```

创建一个deployment

```
cat > deployment.yml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
EOF
```

发现已经自带了我们的限额参数

```
[student@workstation ~]$ oc create -f deployment.yml
[student@workstation ~]$ oc get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6645d8bb58-4jbct   1/1     Running   0          8s
[student@workstation ~]$ oc describe pod nginx-deployment-6645d8bb58-4jbct
...
    Limits:
      cpu:     500m
      memory:  512Mi
    Requests:
      cpu:        250m
      memory:     256Mi
```

恢复默认模板

如果需要恢复默认的模板，就清除spec下面的参数

```
[student@workstation ~]$ oc edit projects.config.openshift.io cluster
...
spec: {}
```

如果需要，你可以重新创建project、deployment，验证pod不再自带限额信息

管理自行置备权限

具有 `self-provisioner` 集群角色的用户可以创建项目。默认情况下， self-provisioner 角色绑定到所有经过身份验证的用户。

```
[student@workstation ~]$ oc describe clusterrolebinding.rbac self-provisioners
Name:         self-provisioners
Labels:       <none>
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  self-provisioner
Subjects:
  Kind   Name                         Namespace
  ----   ----                         ---------
  Group  system:authenticated:oauth
```

如果需要删除经过身份验证的用户创建project的权限，那就删除这个集群角色绑定或者edit一下，清空subjects下面的内容就行，或者你换成别的组或用户，仅授权而已

```
[student@workstation ~]$ oc delete clusterrolebindings self-provisioners
clusterrolebinding.rbac.authorization.k8s.io "self-provisioners" deleted
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

You don't have any projects. Contact your system administrator to request a project.
```

我们发现，他已经需要联系管理员才能创建project了

不过我们要注意`Annotations:  rbac.authorization.kubernetes.io/autoupdate: true`这个参数，这个参数确保我们修改的参数不会影响集群工作，也就是说，一旦执行下面的命令，这个clusterrolebind就会重新恢复，也就是说，仅删除到 API 服务器重启那一刻

```
[student@workstation ~]$ oc rollout restart deployment -n openshift-apiserver
```

如果你真想永久删除，你得这么做

先禁用自动更新，然后再删除绑定

```
oc annotate clusterrolebinding/self-provisioners --overwrite rbac.authorization.kubernetes.
```

```
[student@workstation ~]$ oc delete clusterrolebindings self-provisioners
```

建议熟悉一下前面RBAC的章节，配合这里的内容，实现将角色授予或回收权限。

本文档在线版本：https://www.linuxcenter.cn