

当前位置： [首页](#) > [开发杂谈](#) >

前言

前几篇文章探讨了如何透过 Rancher 操作与管理 Kubernetes 丛集，不论是直接抓取 Kubeconfig 或是使用网页上的 web terminal 来操作，此外也探讨了 Rancher 整合的应用程序，特别是最重要的 Monitoring 该如何使用。

有了上述的概念後，使用者已经可以顺利的操作 Kubernetes 来部署各种服务。

不过 Rancher 想做的事情可没有这麽简单，Rancher 希望能够强化 Kubernetes 让其更佳适合给多位使用者共同使用了。

对於这种多租户的概念，Kubernetes 提供了 namespace 的机制来达到资源隔离，不过 namespace 普遍上被认为是个轻量级的隔离技术，毕竟 namespace 主要是逻辑层面的隔离，底层的运算与网路资源基本上还是共用的。

就算是 RKE 也没有办法完全颠覆 namespace 让其变成真正的隔离技术，毕竟 CPU/Memory/Network 等相关资源因为 Container 的关系本来就很难切割，要达到如 VM 般真正隔离还不是这麽容易。

不过 Rancher 还是有别的方向可以去发展与强化，就是如何让 Kubernetes 变得更适合一个团队使用，如果该团队内有数个不同的专案，这些专案要如何共同的使用一套 Kubernetes 丛集同时又可以有一个清楚且清晰的管理方式。

Project

Rancher 提出了一个名为 project 的概念，Project 是基於 Kubernetes Namespace 的实作的抽象管理概念，每个 Project 可以包含多个 namespace，同时 project 也会与 Rancher 内部的使用者权限机制整合。

从架构层面来看

1. Rancher 管理了多套 Kubernetes 丛集
2. Kubernetes 丛集管理多个 Project
3. Project 拥有多个 namespace.

如同前面探讨的, Kubernetes 原生提供的 namespace 机制是个轻量级虚拟化概念, 所有 kubernetes 内的机制也都是以 namespace 为基础去设计的, 这意味者如果你今天要透过 RBAC 设定权限等操作你都需要针对 namespace 去仔细设计。但是 Rancher 认为一个产品专案可能不会只使用一个 namespace, 而是会使用多个 namespace 来区隔不同的应用程序。

这种情况下你就必须要针对每个 namespace 一个一个的去重复设定, 从结果来说一样可以达到效果, 但是操作起来就是不停重复相同的动作。

透过 Rancher Project 的整合, 丛集管理者可以达到

1. 整合使用者群组权限, 一口气让特定群组/使用者的人针对多个 namespace 去设定 RBAC
2. 针对 Project 为单位去进行资源控管, 一口气设定多个 namespace 内 CPU/Memory 的使用量
3. 套用 Pod Security Policy 到多个 namespace 中

因此实际上管理 Kubernetes 丛集就变成有两种方式

1. 完全忽略 Rancher 提供的 Project 功能, 直接就如同其他 Kubernetes 版本一样去操作
2. 使用 Rancher 所设计的 Project 来管理

Rancher 会开发 Project 势必有其好处, 但是要不要使用就是另外一回事, 因为这个技术与概念是只有 Rancher 才有的, 如果今天团队同时拥有多套不同的 Kubernetes 丛集, 有些用 Rancher 管理, 有些没有。

这种情况下也许不要使用 Rancher 工具而是采用尽可能原生统一的工具来管理会更好, 因为可以避免团队中使用的工具有太多的客制化行为, 造成开发与维护都不容易。相反的使用所有 Kubernetes 发行版本都有的工具与管理方式反而有机会降低工具的复杂性。


所以到底要不要使用这类型的工具反而是见仁见智, 请依据每个团队需求去思考。

接下来就来看一下到底如何使用 Rancher Project 概念。

操作

Project 因为是用来简化同时操作多个 namespace 的一种概念，因此管理上会跟 namespace 放在一起。

Rancher 画面上方的 Projects/Namespaces 就是用来管理这类型概念的，点选进去会看到类似下图的版面。

ithome-devClusterNodesStorageProjects/NamespacesMembersToolsCluster Explorer

Projects/Namespaces

MoveDownload YAMLDel

StateNamespace NameCreated

Namespace: All

沒有被任何 Project 管理的 namespace

Add Namespace

<input type="checkbox"/>	Active	cattle-dashboards	Yesterday at 9:31 PM	
<input type="checkbox"/>	Active	cattle-gatekeeper-system	Last Thursday at 9:06 PM	
<input type="checkbox"/>	Active	cattle-monitoring-system	Yesterday at 9:29 PM	
<input type="checkbox"/>	Active	cis-operator-system	Last Thursday at 9:04 PM	
<input type="checkbox"/>	Active	longhorn-system	Last Thursday at 9:06 PM	

Project: Default

Default project created for the cluster

Add Namespace

<input type="checkbox"/>	Active	default	Last Wednesday at 12:35 AM	
--------------------------	--------	---------	----------------------------	--

Project: System

System project created for the cluster

Add Namespace

<input type="checkbox"/>	Active	cattle-system	Last Wednesday at 12:37 AM	
<input type="checkbox"/>	Active	fleet-system	Last Wednesday at 12:37 AM	
<input type="checkbox"/>	Active	ingress-nginx	Last Wednesday at 12:37 AM	

因为 Project 包含多个 namespace，所以版面中都是以 Project 为主，列出该 Project 底下有哪些 namespace，Rancher 内的任何 Kubernetes 丛集预设都会有两个 Projects，System 与 Default System 内会放置任何跟 Rancher 以及 Kubernetes 有关的 namespace，譬如 cattle-system, fleet-system, kube-system, kube-public 等

Default 是预设的 Project，预设对应到 default 这个 namespace。

任何不是透过 Rancher 创立的 namespace 都不会加入到任何已知的 project 底下，因此图片中最上方可以看到一堆 namespace，而这些 namespace 都不属于任何一个 project。

因此要使用 project 的话就需要把这些 namespace 给搬移到对应的 project 底下。

图中右上方有一个按钮可以创立新的 Project，点下去可以看到如下画面

Project Name *

Add a Description


myApplication

▼ Members 1

Configure who has access to the resources in this project and what permissions they have

Name

Role

 宏璋 邱 ([redacted]@onmicrosoft.com)
User

Project Owner

+ Add Member

▼ Resource Quotas 2

Configure how much of the resources the project can consume

+ Add Quota

▼ Container Default Resource Limit 3

Configure how much of the resources the container can consume by default

CPU Limit

e.g. 1000

milli CPUs

Memory Limit

e.g. 128

MiB

CPU Reservation

e.g. 1000

milli CPUs

Memory Reservation

e.g. 128

MiB

▼ Labels & Annotations 4

Configure labels and annotations for the project.

None

Labels

+ Add Label

Annotations

+ Add Annotation

创立一个 Project 有四个资讯需要输入，分别是

1. 使用者权限

2. Project 资源控管

3. Container 资源控管

4. Labels/Annotation.

使用者权限可以控制属于什么样的使用者/群组可以对这个 Project 有什么样的操作。

Project 与 Container 的资源控管之后会有一篇来介绍

创立完 Project 之后就可以回到最外层的介面，将已经存在的 namespace 给挂到 project 底下

Move 2 namespaces:

longhorn-system
cis-operator-system

To project:

☒ None
☐ Default
☐ System
☐ myApplication




Move Cancel


State	Namespace Name	Created
<input type="checkbox"/> Active	cattle-dashboards	Yesterday at 9:31 PM
<input type="checkbox"/> Active	cattle-gatekeeper-system	Last Thursday at 9:06 PM
<input type="checkbox"/> Active	cattle-monitoring-system	Yesterday at 9:29 PM
<input checked="" type="checkbox"/> Active	cis-operator-system	Last Thursday at 9:04 PM
<input checked="" type="checkbox"/> Active	longhorn-system	Last Thursday at 9:06 PM

譬如上述范例就将 cis-operator-system, longhorn-system 这两个 namespace 给分配到刚刚创立的 project 底下。

▼ Members


Configure who has access to the resources in this project and what permissions they have

Name	Role	
 Default Admin (admin) Local User	Project Owner	
<input type="text" value="qa"/>	<input type="text" value="Owner"/>	

 Add Member

之後重新进入到该 Project 去编辑，尝试将 QA 使用群组加入到该 Project 底下，让其变成 Project Owner，代表拥有完整权限。

ithome-dev ▼ Cluster Nodes Storage ▼ Projects/Namespaces Members Tools ▼

Cluster Explorer 

Global

Search...

Clusters

Projects in ithome-dev

ithome-dev

Active

Default

Active

local

Active

myApplication

Active

rke-it

Active

System

Active

rke-qa

Active

Search

Created ↕

Add Namespace

Yesterday at 9:31 PM

创造完毕後，就可以透过 UI 切换到不同的 project，如上图所示，可以看到 ithome-dev 丛集底下有三个 Project，其中有两个是预设的，一个是刚刚前述创立的。

rancher.hwchiu.com/p/c-z8j6q:p-p6xrd/ns

clusterID/projectID

ithome-dev
myApplication

Resources Apps Namespaces Members Tools

Cluster Explorer

Namespaces

Add Namespace

Move Download YAML Delete

Search

State Namespace Created

Not in a project

<input type="checkbox"/>	Active	cattle-dashboards	Yesterday at 9:31 PM	
<input type="checkbox"/>	Active	cattle-gatekeeper-system	Last Thursday at 9:06 PM	
<input type="checkbox"/>	Active	cattle-monitoring-system	Yesterday at 9:29 PM	

In this project

<input type="checkbox"/>	Active	cis-operator-system	Last Thursday at 9:04 PM	
<input type="checkbox"/>	Active	longhorn-system	Last Thursday at 9:06 PM	

切换到该 Project 之後，观察当前的 URL 可以观察到两个有趣的ID，c-xxxx/p-xxxxx 会分别对应到 clusterID 以及 project ID，因此之後只要看到任何 ID 是 c-xxx 开头的，基本上都是 Rancher 所创立的，跟 Cluster 有关，而 p-xxxx 开头的则是跟 Project 有关，每个 Project 都势必属於某个 Cluster。

有了 ProjectID 之後，仿造之前透过 kubectl 去观察使用者权限的方式，这次继续观察前述加入的 QA 群组会有什麼变化。

```
$ kc get clusterrolebinding -o json | jq '.items[] | select(.subjects[0].name == "azuread_group://ec55ce9e-dbd4-427c-905c-d8063b19f150").roleRef'
[
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "p-vblmb-namespaces-readonly"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "create-ns"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "project-owner-promoted"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "read-only-promoted"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "cluster-member"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "p-p6xrd-namespaces-edit"
  },
  {
    "apiGroup": "rbac.authorization.k8s.io",
    "kind": "ClusterRole",
    "name": "p-q46q5-namespaces-readonly"
  }
]
```

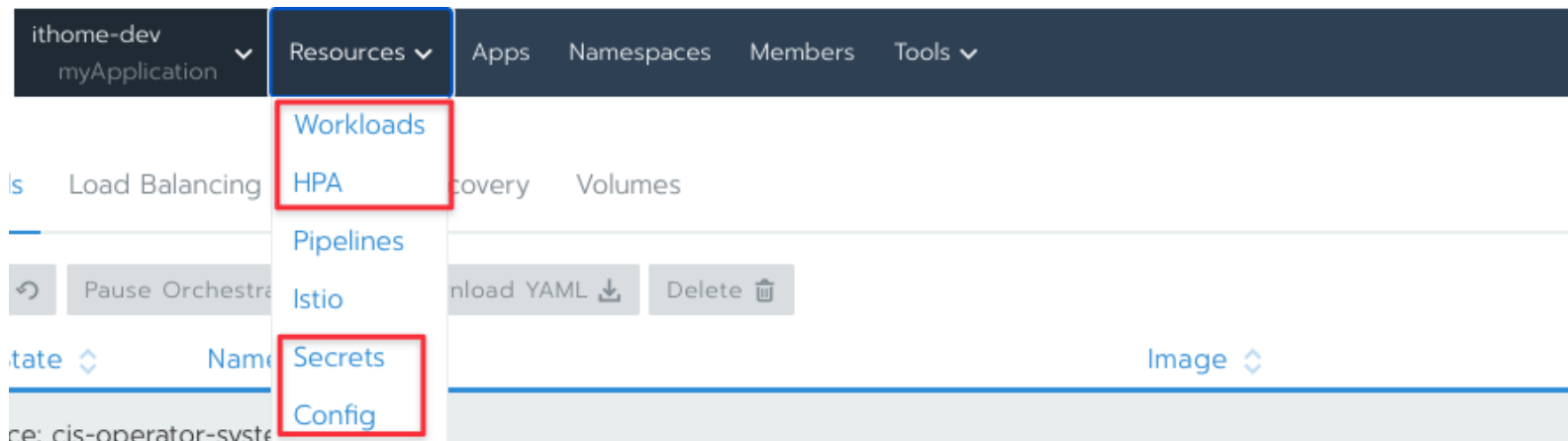
从上述指令中可以看到 QA 群组对应到一个新的 ClusterRole，叫做 p-p6xrd-namespaces-edit，其中 p-p6xrd 就是对应到前述创立的 project，而 edit 代表则是拥有 owner 般的权限，能够去编辑任何资源。

接者更详细的去看一下该 ClusterRole 的内容

```
$ kc get ClusterRole p-p6xrd-namespaces-edit -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    authz.cluster.auth.io/project-namespaces: p-p6xrd-namespaces-edit
  creationTimestamp: "2021-08-15T01:33:23Z"
  labels:
    cattle.io/creator: norman
  name: p-p6xrd-namespaces-edit
  resourceVersion: "1326442"
  uid: 6a1a0dc0-2872-49c7-9d4d-2d116cae5425
rules:
- apiGroups:
  - ""
  resourceNames:
  - cis-operator-system
  - longhorn-system
  resources:
  - namespaces
  verbs:
  - '*'
```

可以看到该 ClusterRole 针对设定的两个 namespace 都给予了 "*" 的动词权限，基本上就是让该使用者能够如管理者般去使用这两个 namespace。

除了上述的权限外，当切换到 Project 的页面时，就可以看到从 Rancher 中去看到该 Project 底下 namespaces 内的相关 Kubernetes 物件资源，譬如下图



Workloads 就是最基本的运算单元，譬如 Pod, Deployment, Job, DaemonSet..等

而 Config(ConfigMap), Secrets 可以看到整个丛集内的相关资源，此外 secret 透过网页的可以直接看到透过 base64 解密後的结果。

注: Pipelines 请忽略他， v2.5 之後 Rancher 会主推使用 GitOps 的方式来部署，因此过往 pipeline 的方式这边就不介绍了。

以 workloads 为范例，点进去後可以更详细的去看当前系统中有哪些 workloads。

The screenshot shows the Rancher UI interface. At the top, there's a navigation bar with 'ithome-dev myApplication' and tabs for 'Resources', 'Apps', 'Namespaces', 'Members', and 'Tools'. A 'Cluster Explorer' button is on the right. Below the navigation bar, there are tabs for 'Workloads', 'Load Balancing', 'Service Discovery', and 'Volumes'. A toolbar contains 'Redeploy', 'Pause Orchestration', 'Download YAML', and 'Delete' buttons. A search bar is on the right. The main content area shows a list of workloads. The 'test' workload in the 'default' namespace is selected, and its context menu is open, showing options like 'Clone', 'Redeploy', 'Add a Sidecar', 'Rollback', 'Execute Shell', 'Pause Orchestration', 'View/Edit YAML', 'View in API', and 'Delete'. The menu is highlighted with a red box.

State	Name	Image	Scale
Namespace: cis-operator-system			
Active	cis-operator	rancher/cis-operator:v10.4 1 Pod / Created 2 days ago / Pod Restarts: 0	1
Namespace: default			
Active	test	nginx 3 Pods / Created 7 minutes ago / Pod Restarts: 0	3
Namespace: longhorn-system			
Active	csi-attacher	rancher/mirrored-longhornio-csi-attacher:v2.2.1-lh1 3 Pods / Created 2 days ago / Pod Restarts: 3	
Active	csi-provisioner	rancher/mirrored-longhornio-csi-provisioner:v16.0-lh1 3 Pods / Created 2 days ago / Pod Restarts: 2	
Active	csi-resizer	rancher/mirrored-longhornio-csi-resizer:v0.5.1-lh1 3 Pods / Created 2 days ago / Pod Restarts: 4	
Active	csi-snapshotter	rancher/mirrored-longhornio-csi-snapshotter:v2.11-lh1 3 Pods / Created 2 days ago / Pod Restarts: 3	
Active	engine-image-ci-817782hc	rancher/mirrored-longhornio-longhorn-engine:v1.11	

每个 workloads 旁边都有一个选项可以打开，打开後会看到如上的选择，这边就有很多功能可以使用，譬如

1. Add a Sidecar, 可以帮忙加入一个 sidecar containr 进去
2. Rollback 到之前版本
3. Redeploy 重新部署
4. 取得该 Pod 的 shell (如果该 workloads 底下有多种 pod, 则不建议这边使用这个功能)

基本上这些功能都可以透过 kubectl 来达到，网页只是把 kubectl 要用的指令给简化，让他更轻松操作。
上述的范例 test 是使用 deployment 去部署的，点进去该 deployment 可以看到更详细 pods 的资讯，如下

Workload: test Active

Namespace: default

Image: nginx

Workload Type: Deployment

Endpoints: n/a

Config Scale: 3
Ready Scale: 3

Created: 7:01 PM
Pod Restarts: 0

[Expand All](#)

▼ Pods

Pods in this workload

Download YAML

Delete 1 Item

<input type="checkbox"/>	State	Name	Image	Node	
<input checked="" type="checkbox"/>	Running	test-7cbffb8484-q25zk	nginx 10.42.0.25 / Created 8 minutes ago / Restarts: 0	worker1 168.62.10.187 / 192.168.0.4	<div>Execute Shell View Logs View/Edit YAML View in API Delete</div>
<input type="checkbox"/>	Running	test-7cbffb8484-jfbp6	nginx 10.42.0.24 / Created 8 minutes ago / Restarts: 0	worker1 168.62.10.187 / 192.168.0.4	
<input type="checkbox"/>	Running	test-7cbffb8484-c28j5	nginx 10.42.0.26 / Created 8 minutes ago / Restarts: 0	worker1 168.62.10.187 / 192.168.0.4	

► Events

Events of current Deployment

► Environment Variables

Environment Variables that were added at creation.

► Ports

Mappings of container listening ports to host ports on public IP addresses

► Node Scheduling

Configure what nodes the pods can be deployed to.

► Health Check

Periodically make a request to the container to see if it is alive and responding correctly.

► Scaling/Upgrade Policy

Configure how pods are replaced when performing an upgrade.

该画面中就可以看到每个 Pod 的资讯，包含 Pod 的名称，部署到哪个节点，同时也可以透过 UI 去执行该 Pod 的 shell 或是观看相关 log。

以上就介绍了关于 Project 的基本概念。Project 是 Rancher 内的最基本单位，因此要透过 Rancher 的 UI 去管理丛集内的各种部署资源则必须要先准备好相关的 Project，并且设定好每个 Project 对应的 namespace 以及使用者权限。

当然 Rancher Project 不是一个一定要使用的功能，因为也是有团队单纯只是依赖 Rancher 去部署 Kubernetes 丛集，而继续使用本来的方式来管理与部署 Kubernetes 丛集，毕竟现在有很多种不同的专案可以提供 Kubernetes 内的资源状况。一切都还是以团队需求为主。