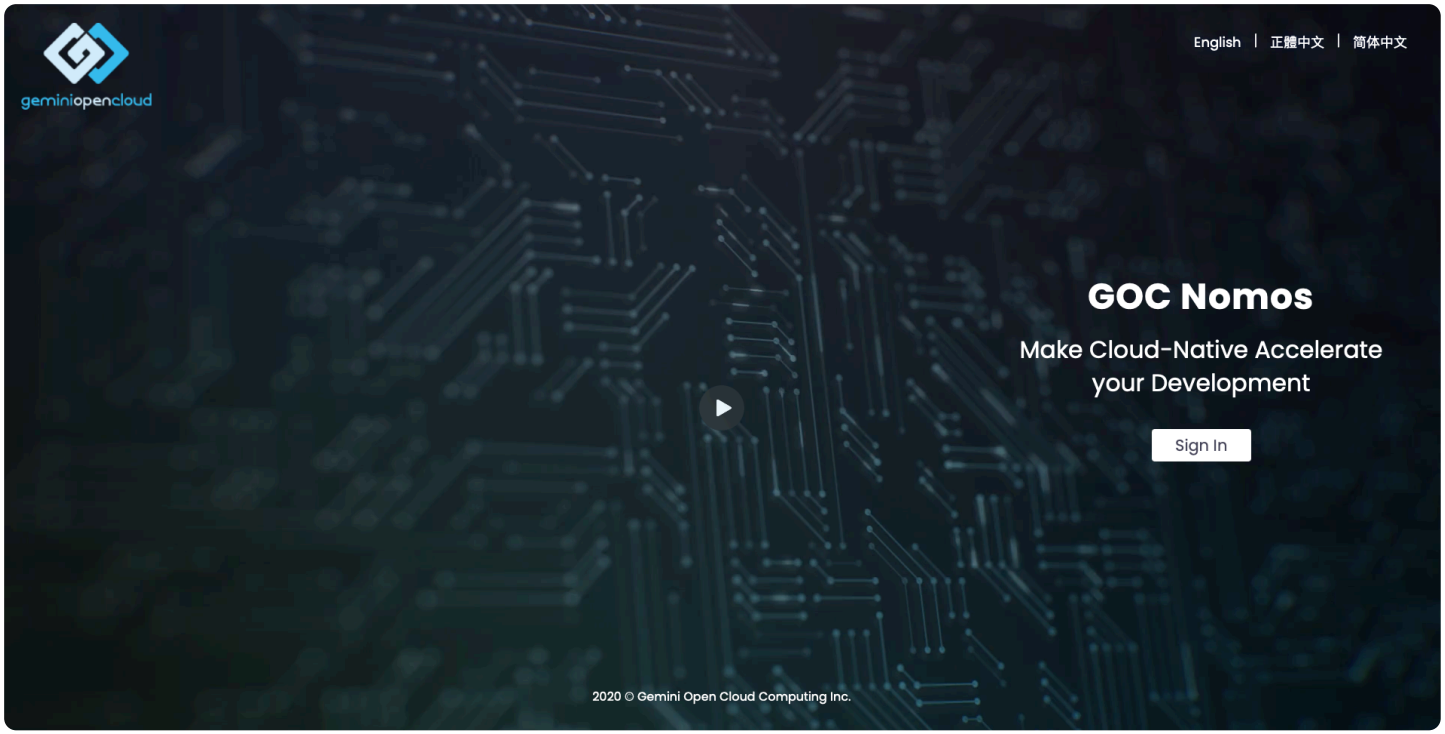
GOC Nomos 使用者操作手冊

User Account

在Nomos的User中，分別有Project Admin和Project User兩種不同的Role。每個Account可以加入多個Projects，並在不同Projects內被指派為不同的Role。在往後的章節，若不同的Role針對特定功能有不同的權限，將會特別說明。

Login Page



從瀏覽器進入Nomos的Web Portal，首先會看到Nomos的登入頁面，點選畫面右邊的Sign In後，就會出現登入表單。

1. 登入說明

在Login Page，使用者需要輸入Account/Email、Password並選擇Login Type。Login Type可選擇Local或LDAP，說明如下：

Local：表示使用者是透過Nomos建立的，其認證單位為Nomos。
LDAP：表示使用者是透過Nomos所整合之LDAP所建立，其認證單位為LDAP。

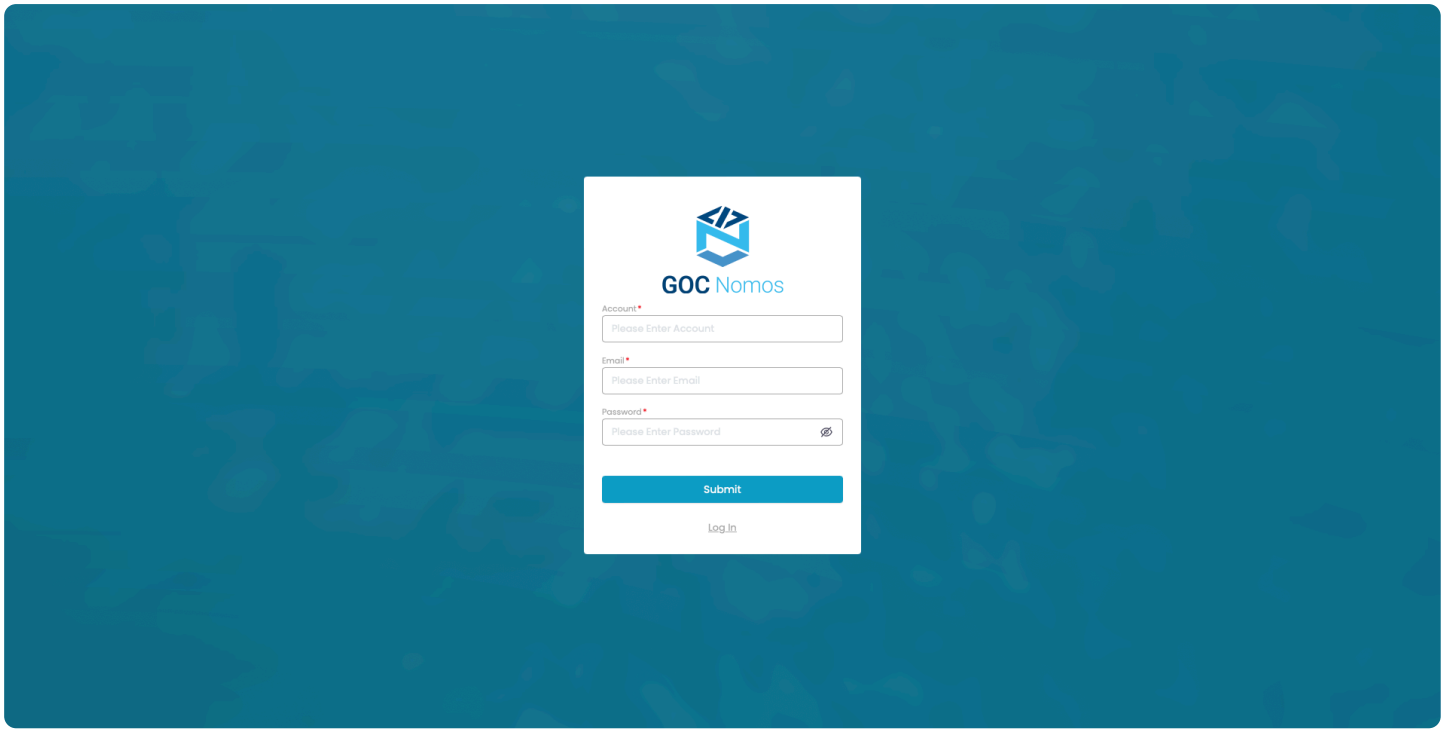
2. 預設使用帳號

Nomos預設的User Account為default，密碼為password，使用的Login Type為 Local。建議使用者在初次登入之後修改密碼，以避免資安問題。

此Account已預設為Project名稱為default的Project Admin。

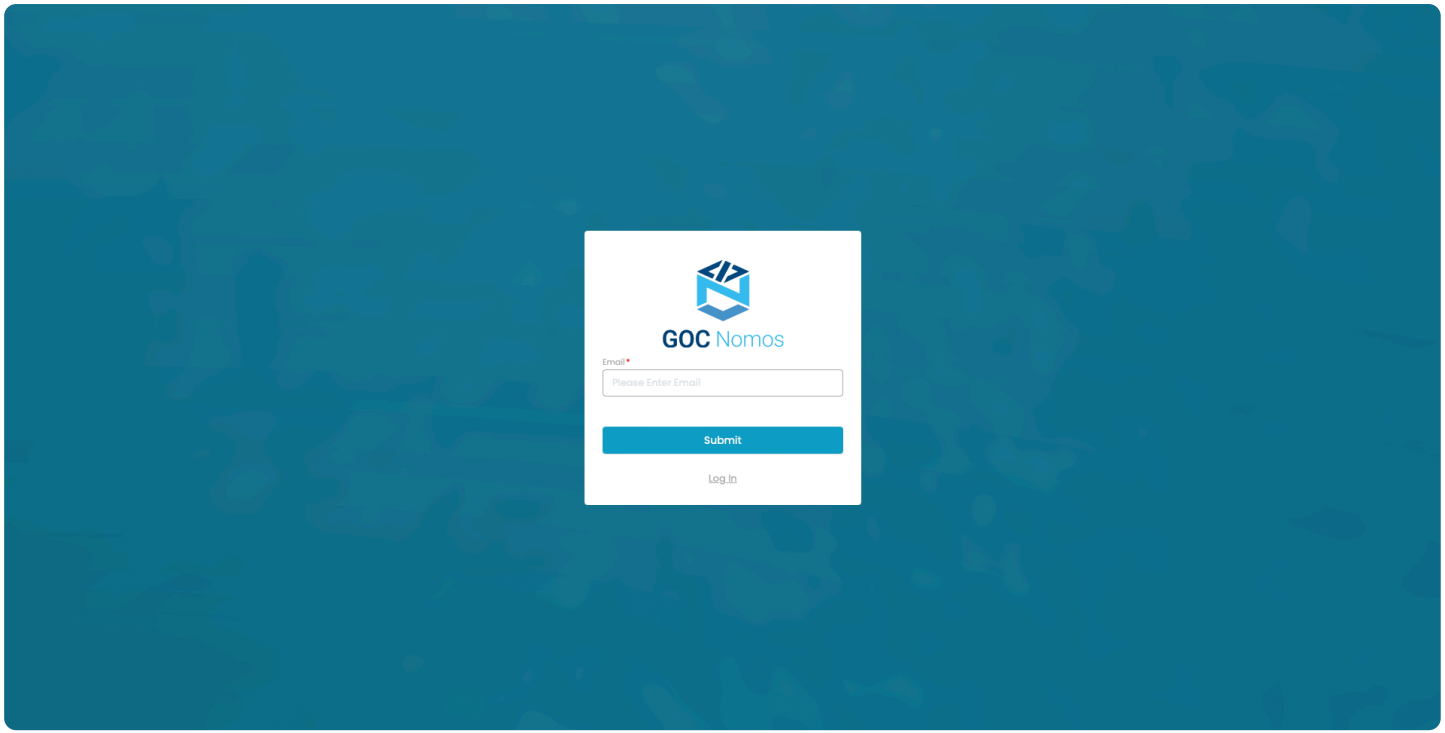
3. 帳號註冊

初次使用Nomos的使用者，可透過登入畫面下方的Register來註冊帳號。點選 Register之後，輸入Account、Email與Password並送出後，等待System Admin的核可，即可以註冊之帳號登入。



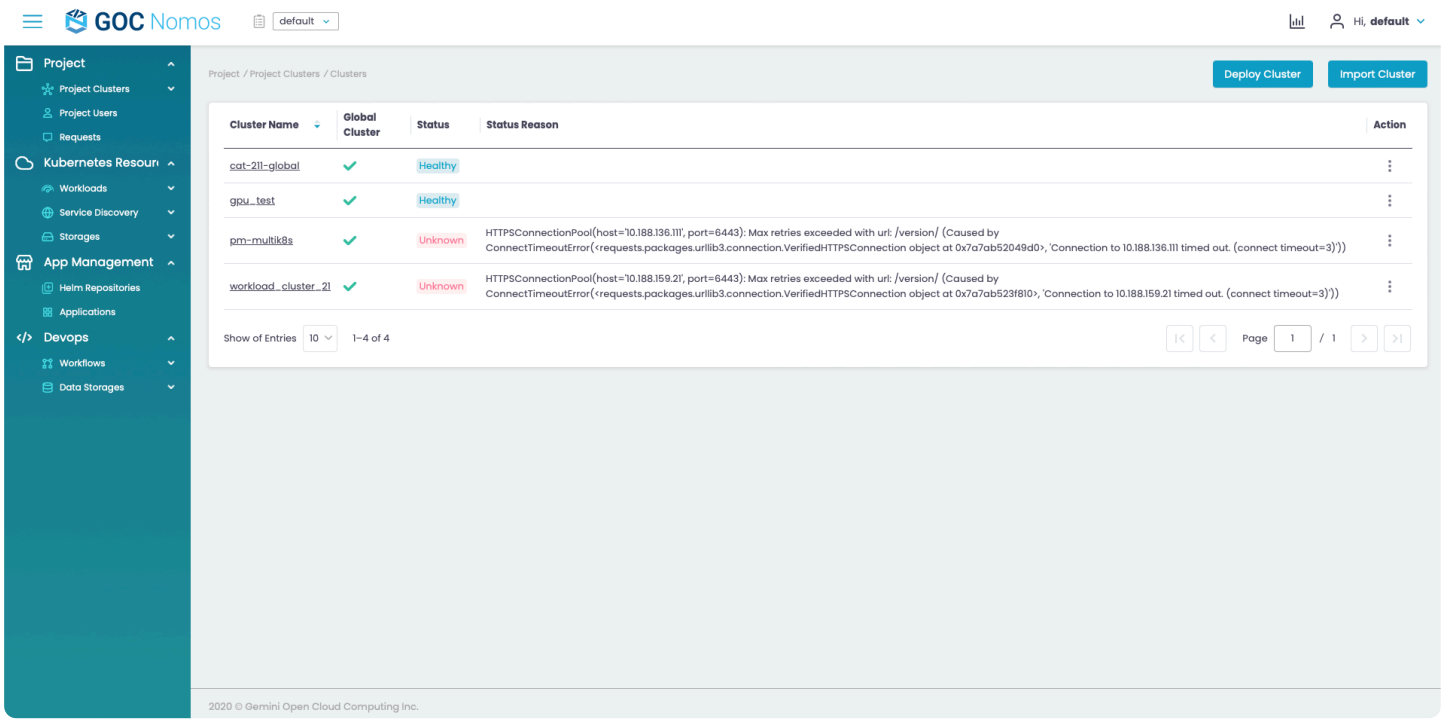
4. 忘記密碼

若使用者已經註冊過帳號，但是忘記了密碼，則可以透過表單下方點選Forget Password。點選Forget Password之後，輸入Email並送出後，系統會自動產生隨機的密碼並以Email通知使用者，使用者即可使用新的密碼登入。



Landing Page

Landing Page為使用者成功登入後第一個看到的頁面，Nomos所有的功能皆展示在此畫面中。主要可以分成三個區塊，以下分別說明。



1. 左側選單

左側選單為Nomos可使用的主要功能，包括：

- Projects
- Kubernetes Resources
- App Managemnet
- Devops

我們會在後面的章節做詳細的說明。

2. 上方選單

上方選單主要三個功能，包括：

- Project下拉選單
- User下拉選單
- Grafana Dashboard

其中Project下拉選單可讓使用者切換不同的Projects。User下拉選單與Grafana Dashboard的部分，我們會在後面的章節做詳細的說明。

3. 主畫面

當使用者透過左側選單或上方選單點選任意功能時，主畫面即呈現該功能所對應的畫面資訊，使用者可進一步於主畫面操作Nomos的各項功能。

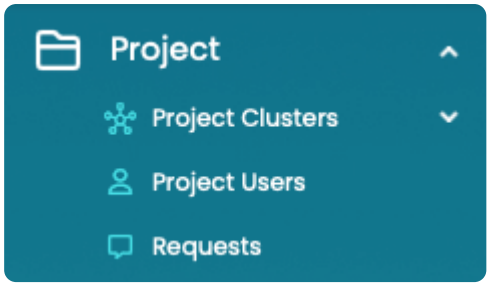
Projects

> Projects下的功能只有Project Admin可以使用。

在Projects中可以管理此Projects中重要的資源，因此，Projects下面的所有功能，皆只有Project Admin可以操作。包含：

1. Project Clusters
2. Project Users
3. Requests

我們會在後面的章節做詳細的說明。

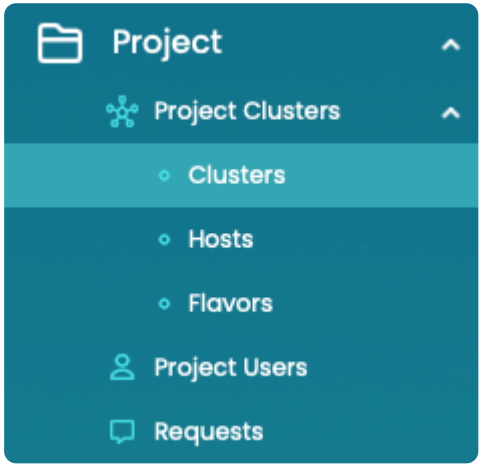


Project Clusters

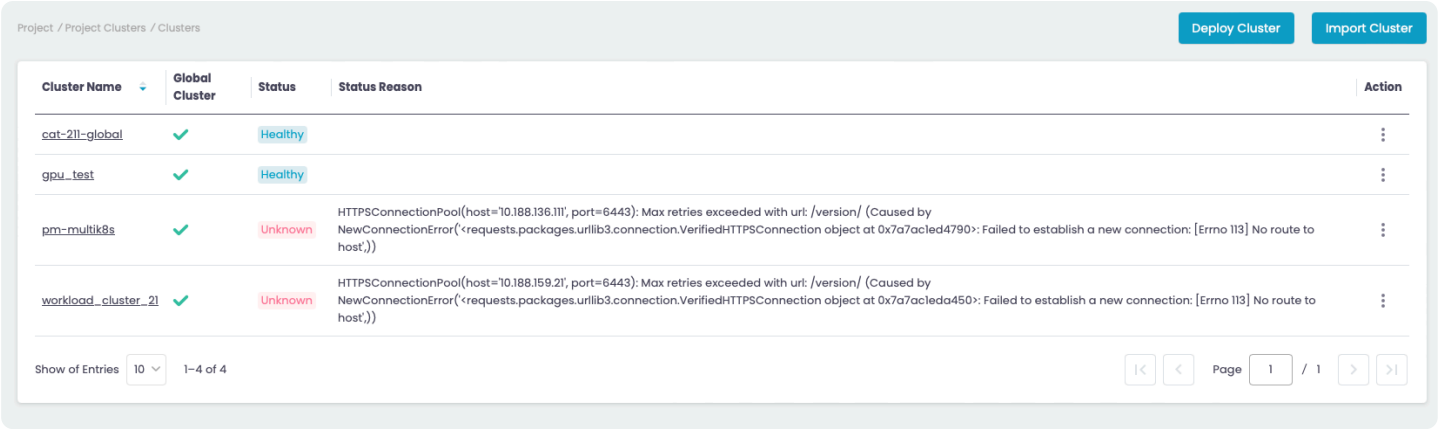
Project Clusters主要用來管理Kubernetes Clusters。在功能上又可以細分為：

1. Clusters
2. Hosts
3. Flavors

我們會在後面的章節做詳細的說明。

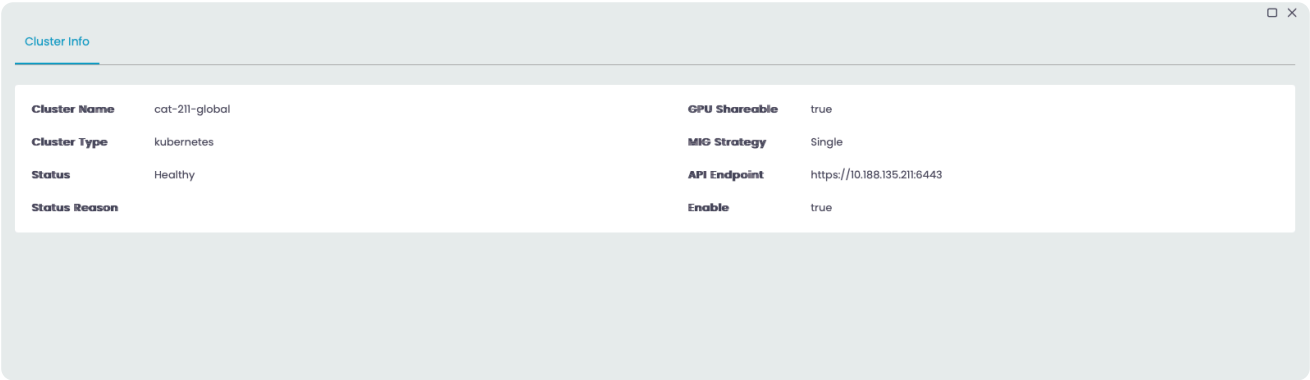


Clusters



Clusters頁面列出所有當下Project所能使用的Kubernetets，每座Kubernetes Cluster包含數個資訊，說明如下：

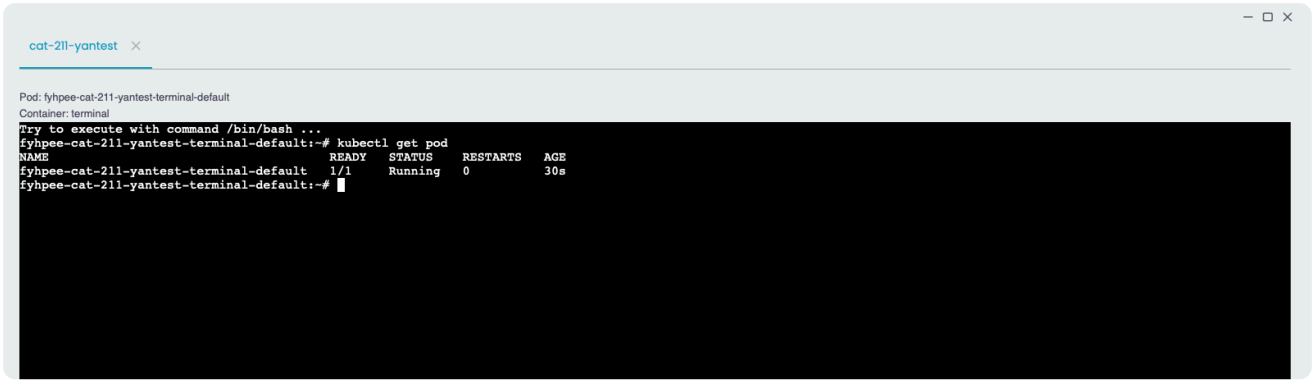
- Cluster Name：此Kubernetes Cluster的名稱。若點選名稱則可看到該Cluster的細節。除了在列表上可看到的基本資訊外，還會呈現包含以下的資訊：
  - GPU Shareable：該Cluster上的GPU是否支援GPU Partition功能。
  - MIG Strategy：該Cluster上的GPU的MIG Strategy。
  - API Endpoint：該Cluster的API Endpoint。
  - Enable：該Cluster是否啟用。



- Global Cluster：此Kubernetes Cluster是否為Global Cluster。若為Global Cluster，表示其為System Admin所建立，因此整個系統的使用者都可以使用此Kubernetes Cluster。反正，

則表示其為Project Admin所建立，即為Project Cluster。

- Status：此Kubernetes目前的狀態，正常狀態為Healthy。
- Status Reason：當此Kubernetes為異常狀態時，顯示異常狀態的細節。
- Action：該Kubernetes可執行的動作，包含：
  - Kubectl Shell：可直接於瀏覽器開啟終端機視窗，對指定的Kubernetes Cluster以Kubectl指令列操作。若使用者開啟的是Global Cluster，Nomos會自動根據所選擇的Project配對一個Namespace，使用者只能在該Namespace下進行操作並受到權限的管控。若使用者開啟的是Project Cluster，則使用者將沒有任何權限上的限制。



  - Download Kubeconfig：可下載指定的Kubernetes Cluster的Kubeconfig。此操作同樣會根據所選擇的Kubernetes是否為Global Cluster，而給予不同權限的Kubeconfig。
  - Delete：此為指定的Kubernetes為Project Cluster才會出現的選項。使用者可將此Kubernetes Cluster從Nomos中刪除，此動作並不會對該Kubernetes Cluster進行任何異動，而只是將其從Nomos的管理範圍中移除。

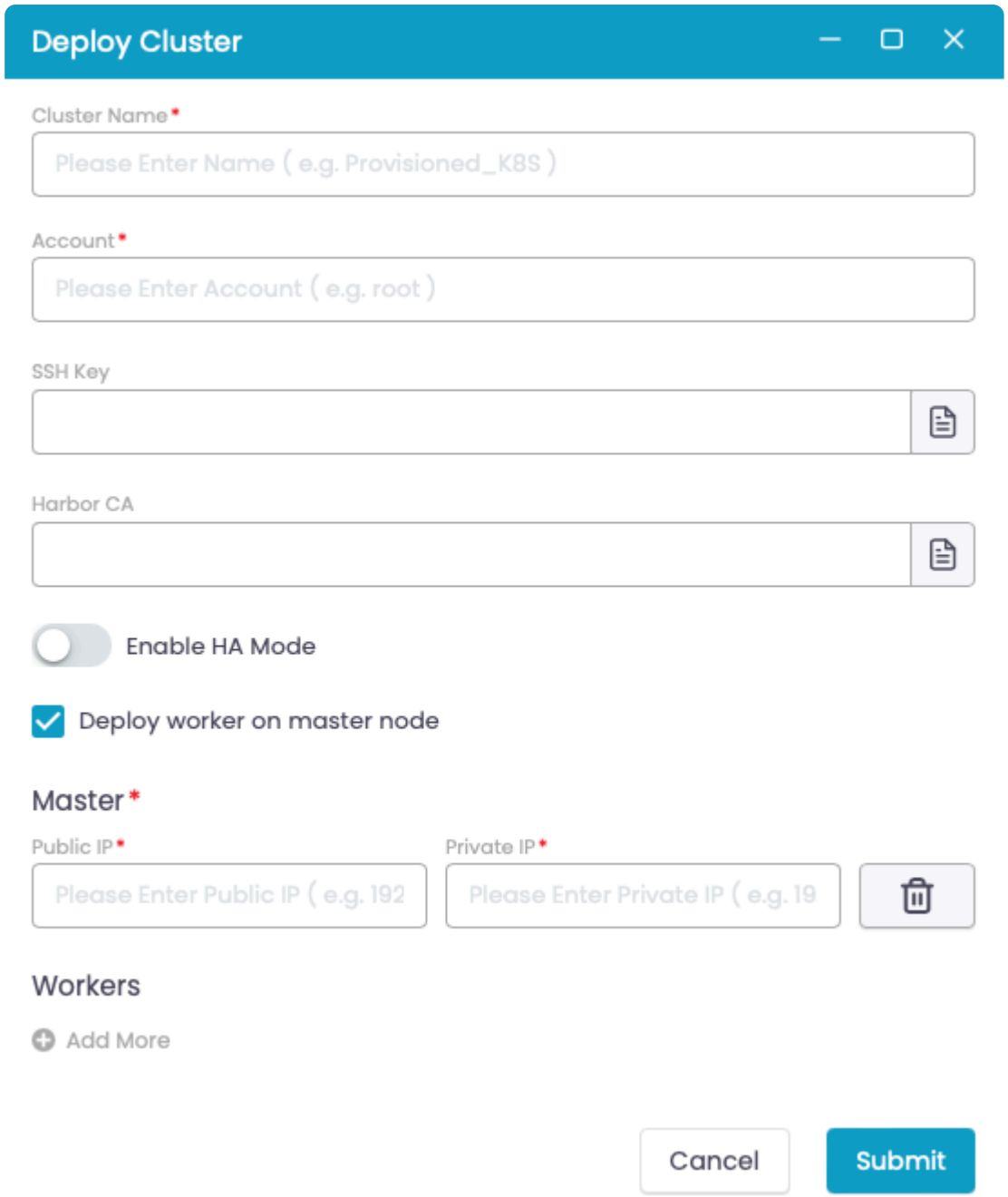
在Clusters頁面的右上角有兩個功能，分別是Deploy Cluster以及Import Cluster，說明如下：

1. Deploy Cluster

Deploy Cluster的功能是可以在指定的機器上部署Kubernetes Cluster，目前Nomos Cluster所支援部署的Kubernetes版本為v1.25.6。此外，指定的機器需符合幾個條件：

- 機器需處於Nomos服務可以存取到的網路環境，且機器的網路環境需可以對外。

- 機器上需事先安裝好指定的作業系統：Ubuntu 22.04。



點選Deploy Cluster後，即會出現建立表單，每個欄位的說明如下：

- Cluster Name：欲部署之Kubernetes Cluster名稱。
- Account：欲部署之機器的登入帳號。
- SSH Key：欲部署之機器的登入SSH Key。
- Harbor CA：此環境中使用的Harbor CA。
- Enable HA Mode：是否要將Kubernetes Cluster部署成HA模式。
- VIP：若選擇HA Mode，則可以指定VIP的資訊。此VIP需可將流量導入Kubernetes的 Master 中。
- Deploy worker on master node：是否要將worker部署到master node上，若不勾選，則master node將只負責Kubernetes Cluster的系統運作工作負載，反正，master node將負責部分應用服務的工作負載。若勾選此選項，則Workers欄位則可以選填。
- Master：欲部署為Master之機器的IP資訊。若機器在公有雲上，則可分別填Public IP與Private IP，反之，Public IP與Private IP皆填相同參數即可。在HA Mode時，需提供三台機器以供Master部署使用。
- Workers：欲部署為Worker之機器的IP資訊。若機器在公有雲上，則可分別填Public IP與Private IP，反之，Public IP與Private IP皆填相同參數即可。

當填寫完畢後，點擊Submit按鈕，即可開始部署Kubernetes Cluster，而部署的相關資訊，則會出現在Clusters頁面中。

2. Import Cluster

Import Cluster的功能是將現有的Kubernetes Cluster匯入Nomos Cluster中，以供Nomos進行後續的管理。匯入的Kubernetes Cluster需處於Nomos可以存取到的網路環境。

Import Cluster

1 Basic Info *

Cluster Name *

Please Enter Cluster Name

Server *

Please Enter Server (e.g. https://123.123.123.123:6443)

Vendor *

Linode

Auth Type *

Token

CA Certificate *

Please Enter CA Certificate

Token *

Please Enter Token

Endpoint Type *

cluster

Prometheus Monitor

Please Enter Monitor (e.g. http://prometheus_ip:port)

Back

Next

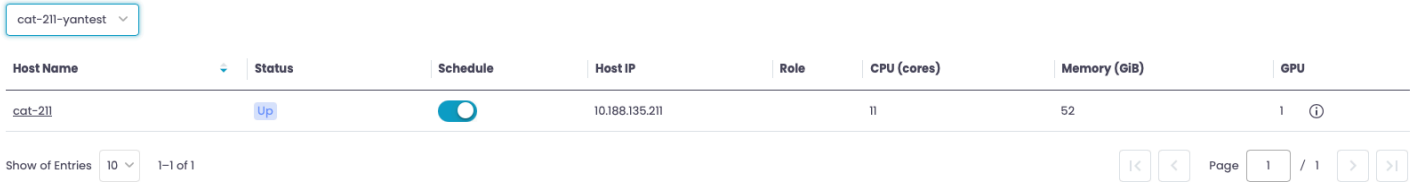
點選Import Cluster後，即會出現建立表單，每個欄位的說明如下：

- Cluster Name：欲匯入之Kubernetes Cluster名稱。
- Server：欲匯入之Kubernetes Cluster API Endpoint資訊。
- Vendor：目前分為Linode/GCP/Private三種。根據所選擇的為公有雲或私有雲，會影響License的計算方式。
- Auth Type：認證的機制。共分為Client Certificate/Token/Basic三種類型。每種類型需提供不同的認證資訊，列舉如下：
  - Client Certificate
    - CA Certificate
    - Client Certificate
    - Client Key
  - Token
    - CA Certificate
    - Token
  - Basic
    - CA Certificate
    - UserName
    - Password
- Endpoint Type：Endpoint Type分為cluster/node兩種類型。cluster表示當建立NodePort的Serivce時，顯示的IP為cluster IP，而選擇node時，則會顯示node IP。
- Prometheus Monitor：若欲匯入之Kubernetes Cluster有安裝Promethues來監控Kubernetes的狀態，可提供其API Endpoint資訊，Nomos將自動整合其監控資訊。

當填寫完畢後，點擊Next可預覽填寫結果，再點擊Submit即可開始匯入Kubernetes Cluster，匯入成功的Kubernetes Cluster則會出現在Clusters頁面中。

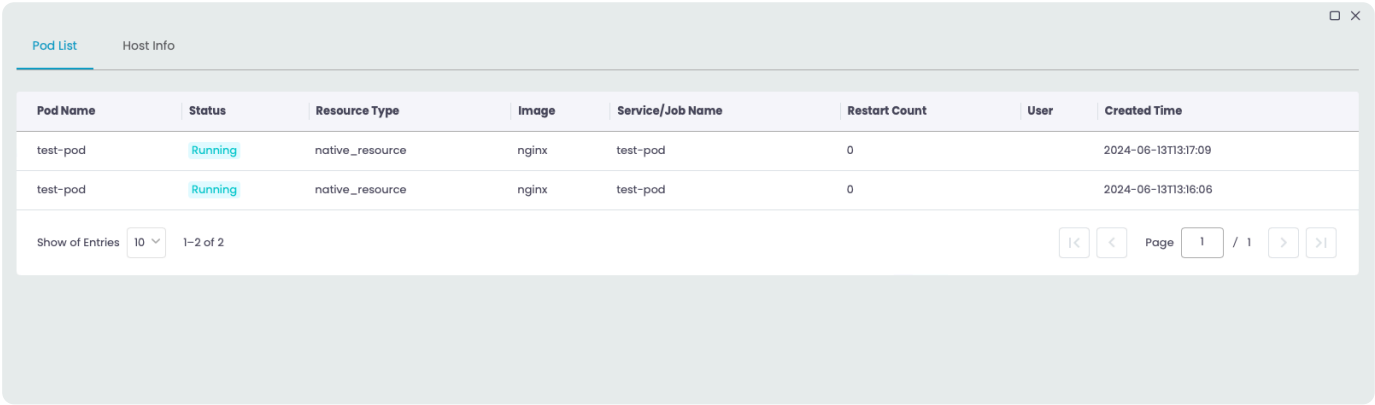
Hosts

Hosts頁面可根據所選擇的Kubernetes Cluster，顯示其對應的節點資訊。



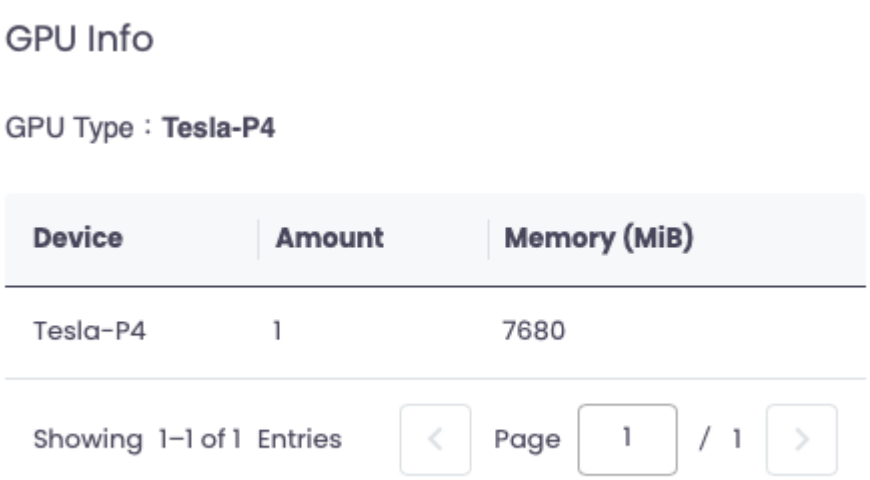
如圖所示，左上角的下拉選單，可以選擇欲查看的Kubernetes Cluster，因為Global Cluster的管理者為System Admin，所以選單中只會出現Project Cluster的選項。待選定Kubernetes Cluster後，畫面即會顯示該Cluster相關的節點資訊，包含：

- Host Name：該節點的Hostname。若直接點選特定Host，則會顯示該Host的細節，包含Pod List與Host Info。其中Pod List會顯示實際上在該Host上執行的Pod的相關資訊，而Host Info則是呈現該Host的基本資訊。



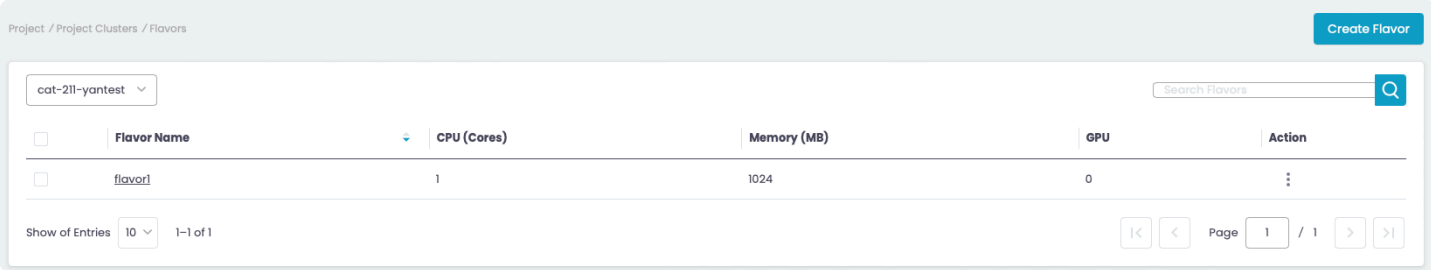
- Status：該節點目前的狀態為Up或Down。
- Schedule：該節點是否可以被派遣工作負載，可根據需求啟用或停用。

- Host IP：該節點的IP。
- Role：該節點在Kubernetes中的角色。
- CPU(cores)：該節點的CPU數量。
- Memory(GiB)：該節點的記憶體大小。
- GPU：該節點的GPU資訊，若將滑鼠移動到旁邊的提示符號，還會顯示GPU的細部資訊，包含GPU的型號、數量、和記憶體大小。



Flavors

Flavors為自定義的規格，可以設定CPU/Memory/GPU等規格，並在建立Devops中Workflow Templates時使用，用來指定所建立的運算資源的規格，後續章節將有詳細的說明。Flavors頁面可根據所選擇的Kubernetes Cluster，顯示其對應的Flavor資訊。

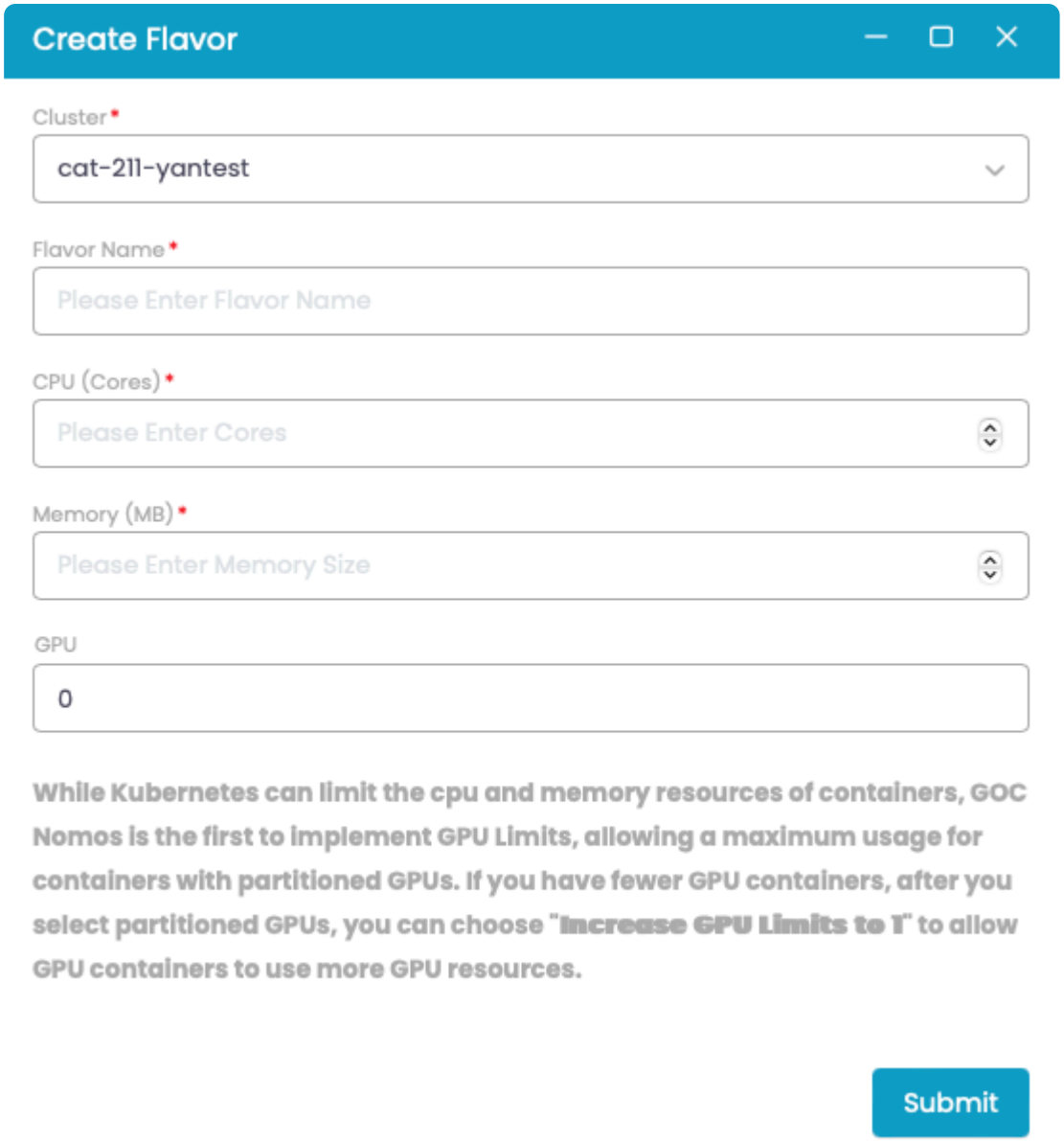


如圖所示，左上角的下拉選單，可以選擇欲查看的Kubernetes Cluster，因為Global Cluster的管理者為System Admin，所以選單中只會出現Project Cluster的選項。待選定Kubernetes Cluster後，畫面即會顯示該Cluster相關的Flavor資訊，包含：

- Flavor Name：該Flavor的名稱。
- CPU(Cores)：該Flavor的CPU數量。
- Memory(MB)：該Flavor的記憶體大小。
- GPU：該Flavor的GPU數量。
- Action：該Flavor可執行的動作。目前只有Delete這個選項，即刪除該Flavor。

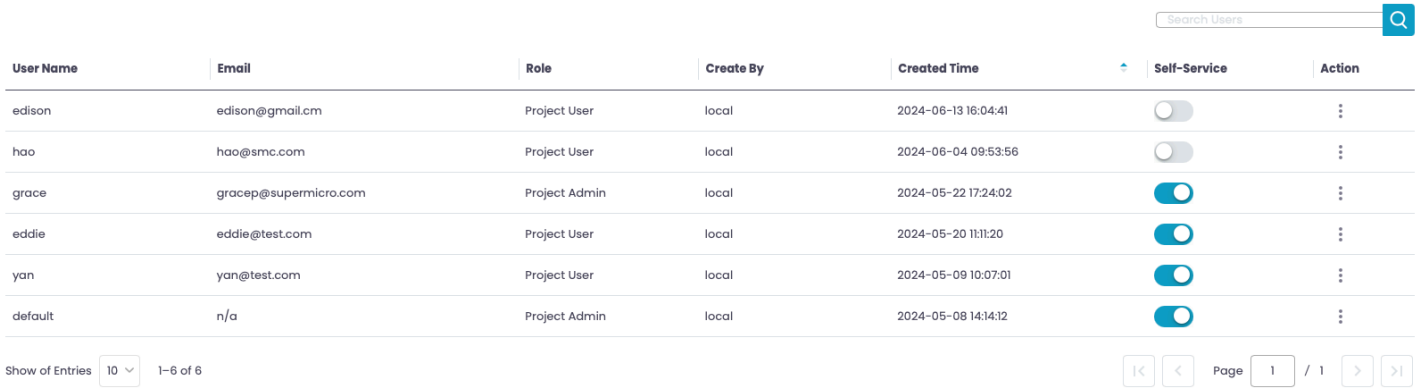
在Clusters頁面的右上角有Create Flavor的按鈕，點擊後即可出現Flavor的建立表單，每個欄位的說明如下：

- Cluster：選擇此Flavor要建立在哪個Kubernetes Cluster中，只會出現Project Cluster的選項。
- Flavor Name：該Flavor的名稱。
- CPU(Cores)：該Flavor的CPU數量。
- Memory(MB)：該Flavor的記憶體大小。
- GPU：該Flavor的GPU數量。



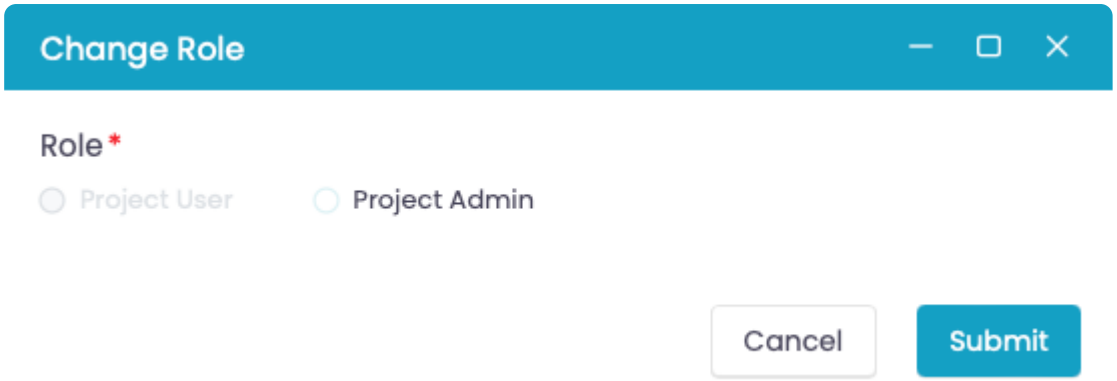
當填寫完畢後，點擊Submit按鈕，即可建立Flavor，相關資訊將會出現在Flavors頁面中。

Project Users



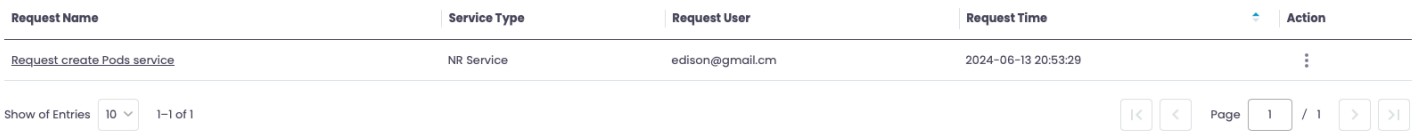
Project Users頁面用來管理此Project中的User，列出所有當下Project的User，每個User包含數個資訊，說明如下：

- User Name：該使用者的名稱。
- Email：該使用者的Email。
- Role：該使用者在此Project下的Role，可為Project Admin或Project User。
- Create By：該使用者的認證方式，可為local或LDAP。
- Crated Time：該使用者的建立時間。
- Self-Serivce：該使用者是否可自行建立資源。若使用者的身份為Project Admin，則必定可以自行建立資源，若使用者的身份為Project User，則可由任一Project Admin決定該使用者是否可以自行建立資源，若關閉Self-Service，則該使用所建立的資源都須經過Project Admin的同意後才可建立。Project Admin如何核准Project User的資源，將於後續章節進行說明。
- Action：該User可執行的動作。目前只有Change Role這個選項，即改變該User的Role為Project Admin或Project User。



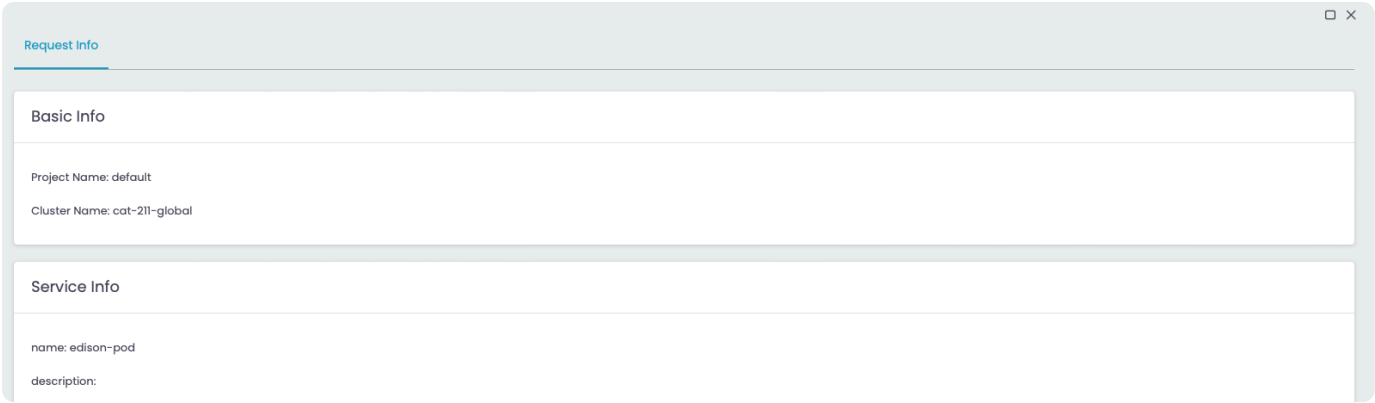
Requests

若使用者為Project User且其Self-Service並未啟用，則該使用者所建立的資源皆需經過Project Admin的同意才能建立。Project Admin可在Requests這個頁面來核准各Project User所發出的資源請求。

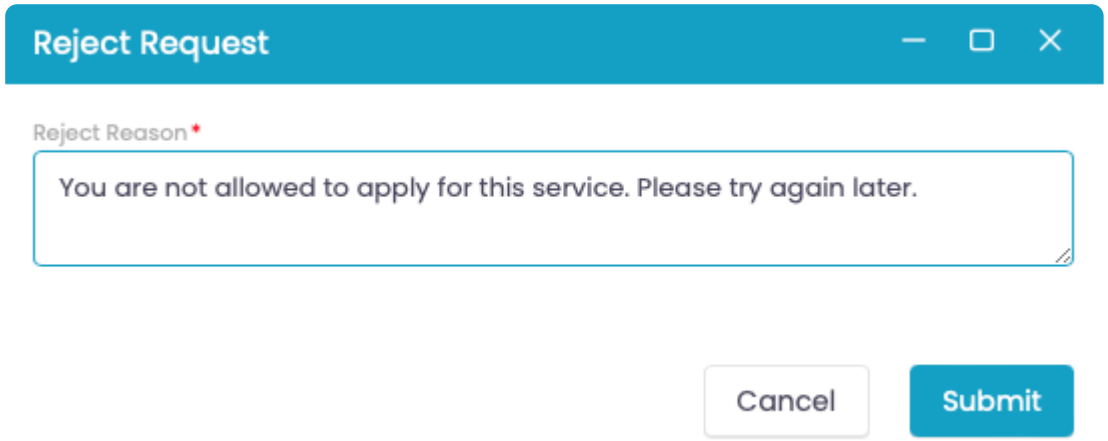


在Requests頁面，Project Admin可以瀏覽所有的Requsts，每個Reqeust包含數個資訊，說明如下：

- Request Name：該Request的名稱。若直接點選特定Request，則會顯示該Request的細節，包含Basic Info中的基本資訊，以及Service Info中，使用者欲建立的資源的詳細規格。



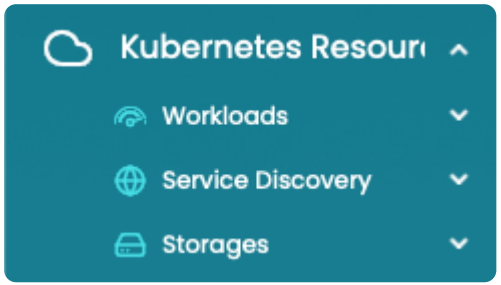
- Service Type：該Request的類型。
- Request User：提出該Request的User。
- Request Time：該Request建立的時間。
- Action：該Request可執行的動作，包含：
  - Approve：系統就會開始建立Request中的資源，反之則會將該Request拒絕
  - Reject：拒絕該Request。可留下拒絕的原因給提出Request的使用者。



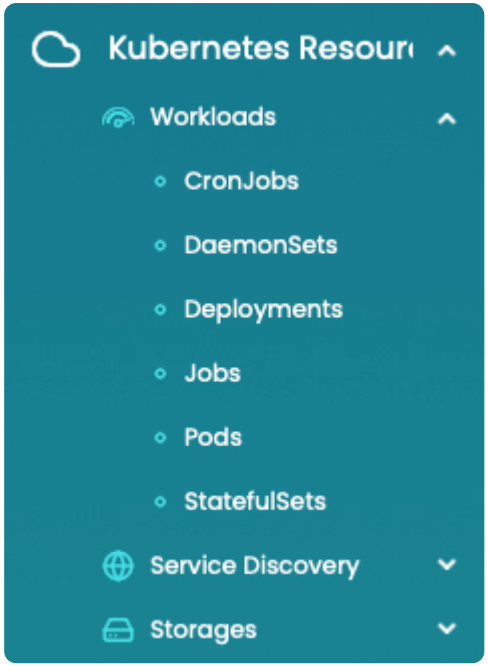
Kubernetes Resources

> Kubernetes Resources下的資源，可供Project Admin與Project User使用，但是有幾項差別。
>
> 1. Project Admin可以查看、更新和刪除所有這個Project下被建立的資源。Project User只能查看、更新和刪除由自己所建立的資源。
> 2. 某些特定的資源，只有Project Admin有權限建立。

Kubernetes Resources讓使用者可透過UI的方式管理各種常見的Kubernetes中的資源類型，依照資源的特性，我們將資源分為Workloads、Service Discovery以及Storages三大類，以下章節將進行詳細的說明。



Workloads

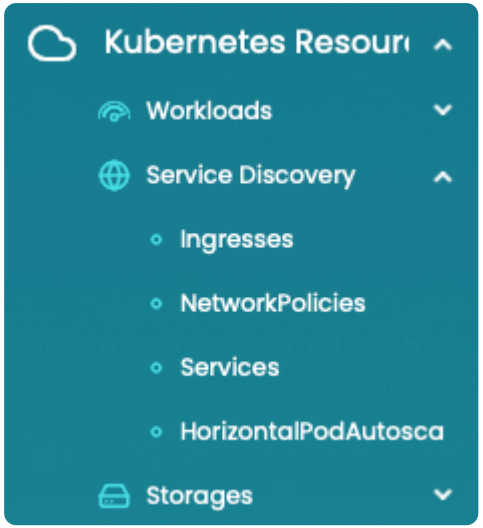


Workloads類型的資源為運算資源，即實際上會分配到CPU、Memory與GPU的資源類型。
Kubernetes上常見的Workloads類型資源可分為以下幾種：

- CronJobs：會定時執行的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/)。
- DaemonSets：在所有節點上同時運行的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/)。
- Deployments：適用應用服務運行的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)。
- Jobs：執行完畢就會自動結束的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/controllers/job/)。
- Pods：最基本的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/pods/)。

- StatefulSets：狀態固定的Workload，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/)。

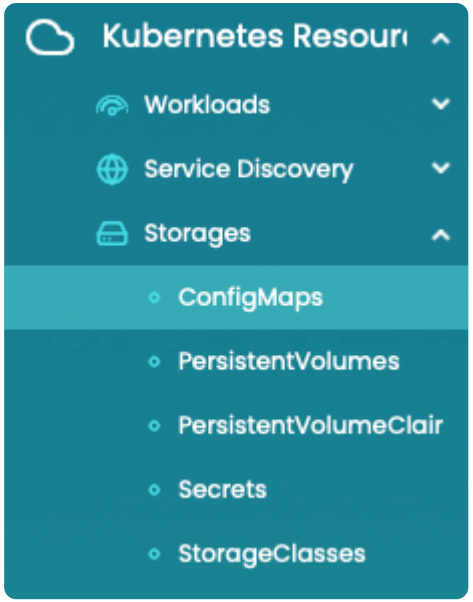
Service Discovery



Service Discovery類型的資源通常跟網路相關，Kubernetes上常見的Service Discovery類型資源可分為以下幾種：

- Ingresses：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/services-networking/ingress/)。
- NetworkPolicies：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/services-networking/network-policies/)。
- Service：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/services-networking/service/)。
- HorizontalPodAutoscalers：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/)。

Storages

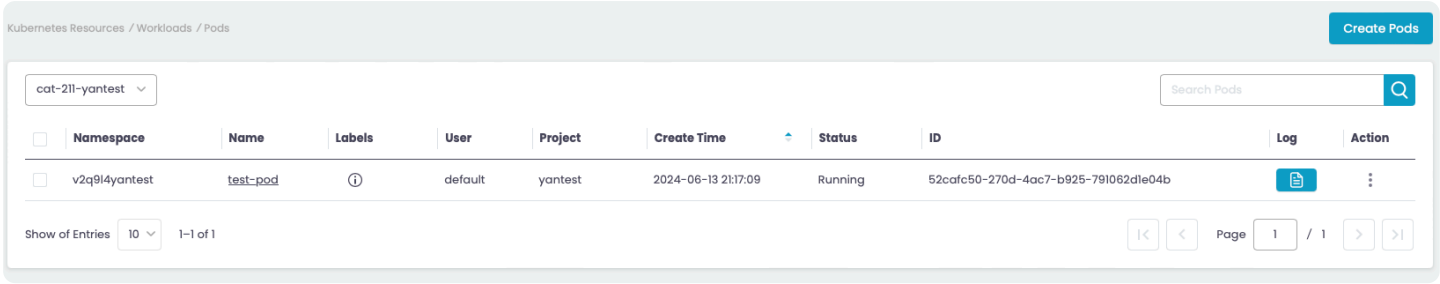


Storages類型的資源顧名思義跟儲存相關，Kubernetes上常見的Storages類型資源可分為以下幾種：

- ConfigMaps：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/configuration/configmap/)。
- PersistentVolumes：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/storage/persistent-volumes/)。
- PersistentVolumeClaims：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/storage/persistent-volumes/#persistentvolumeclaims)。
- Secrets：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/configuration/secret/)。
- StorageClasses：，詳情請見Kubernetes官網 (https://kubernetes.io/docs/concepts/storage/storage-classes/)。Storage Class是只有Project Admin能夠建立的資源。

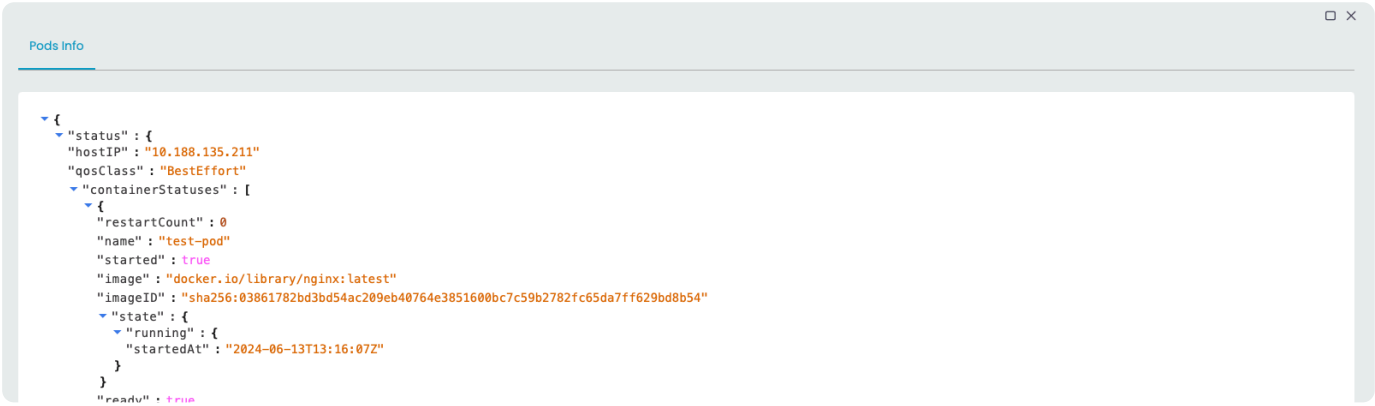
資源管理

由於Kubernetes Resources下的資源管理方式大同小異，以下以Pod為範例，說明如何於Nomos上管理Kubernetes資源。

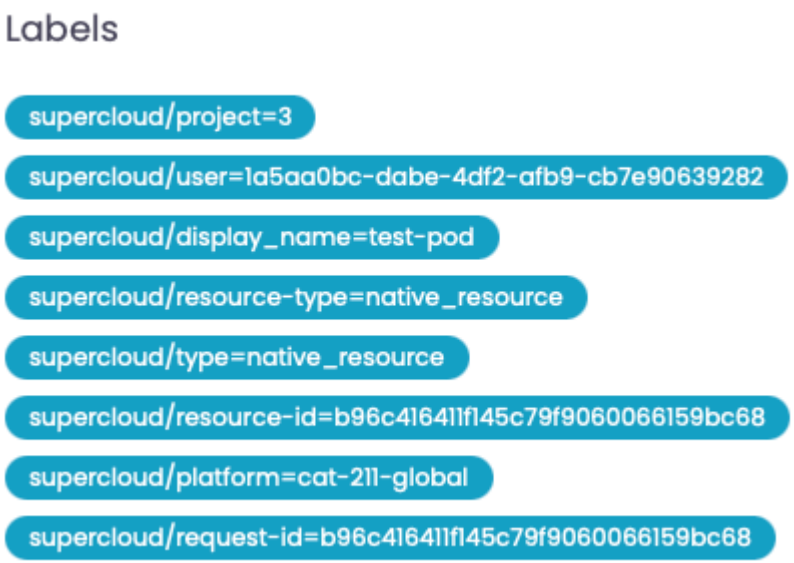


如圖所示，左上角的下拉選單，可以選擇欲建立資源的Kubernetes Cluster，包含Global Cluster與Project Cluster。待選定Kubernetes Cluster後，畫面即會顯示該Cluster下目前已經建立的Pod資訊，包含：

- Namespace：實際上在Kubernetes上運行Pod的Namespace。此Namespace與Nomos的Project為一對一的對應關係。
- Name：該Pod的名稱。若點選名稱則可看到該Pod的細節，包含其於Kubernetes上詳細的規格資訊。

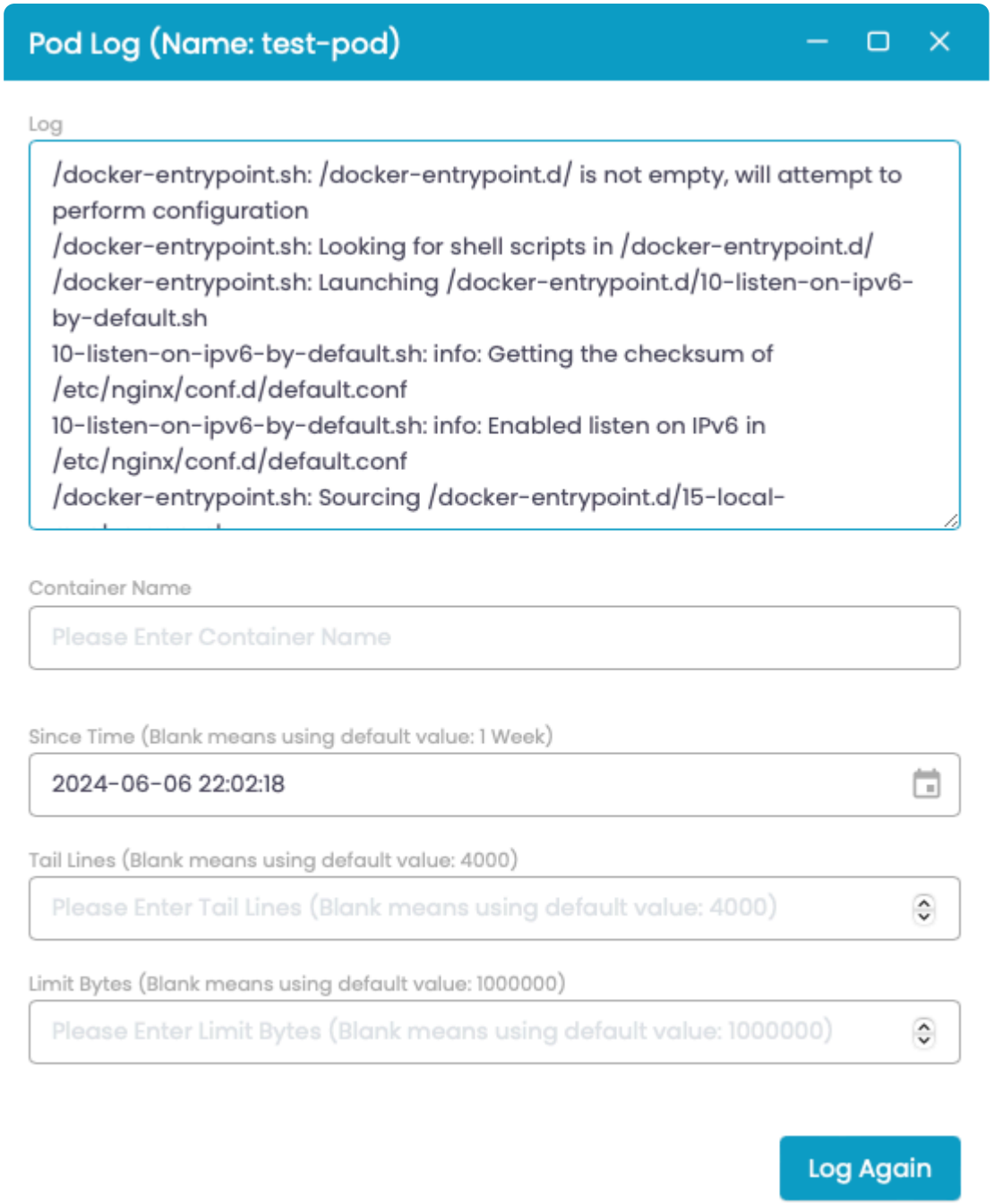


- Labels：該Pod所標註的Label，將滑鼠移動到提示符號可看到細節。

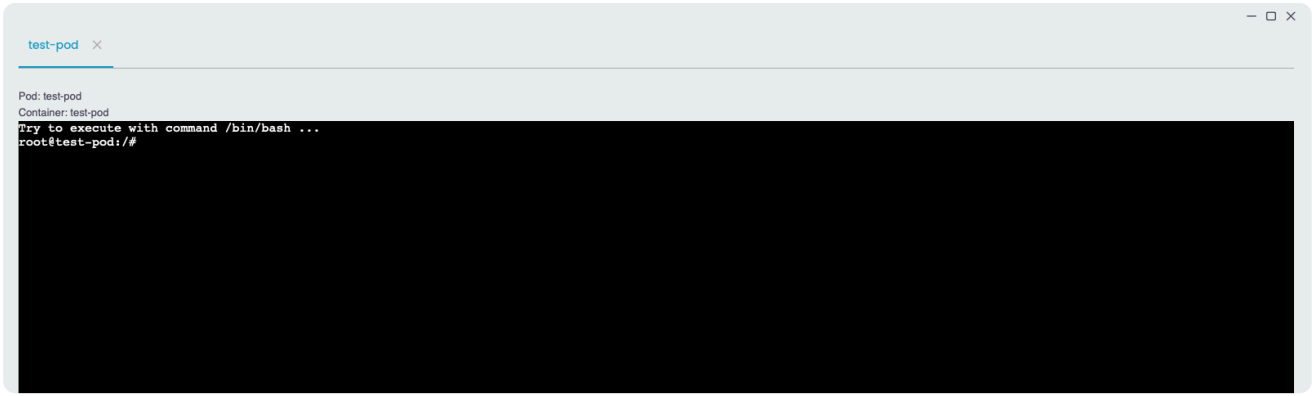


- User：建立該Pod的使用者。
- Project：該Pod於Nomos上所屬的Project，表示該Pod為使用者於此Project中建立的。
- Create Time：該Pod建立的時間。
- Status：該Pod目前的狀態。
- ID：該Pod於Kubernetes上的ID。
- Log：Log是只有Workloads這種類型的資源才擁有的屬性，點選Log圖示，可出現該Pod運行所產生的Log，亦可透過Container Name/Since Time/Tail Lines/Limit Bytes來過濾Log中的

資訊。

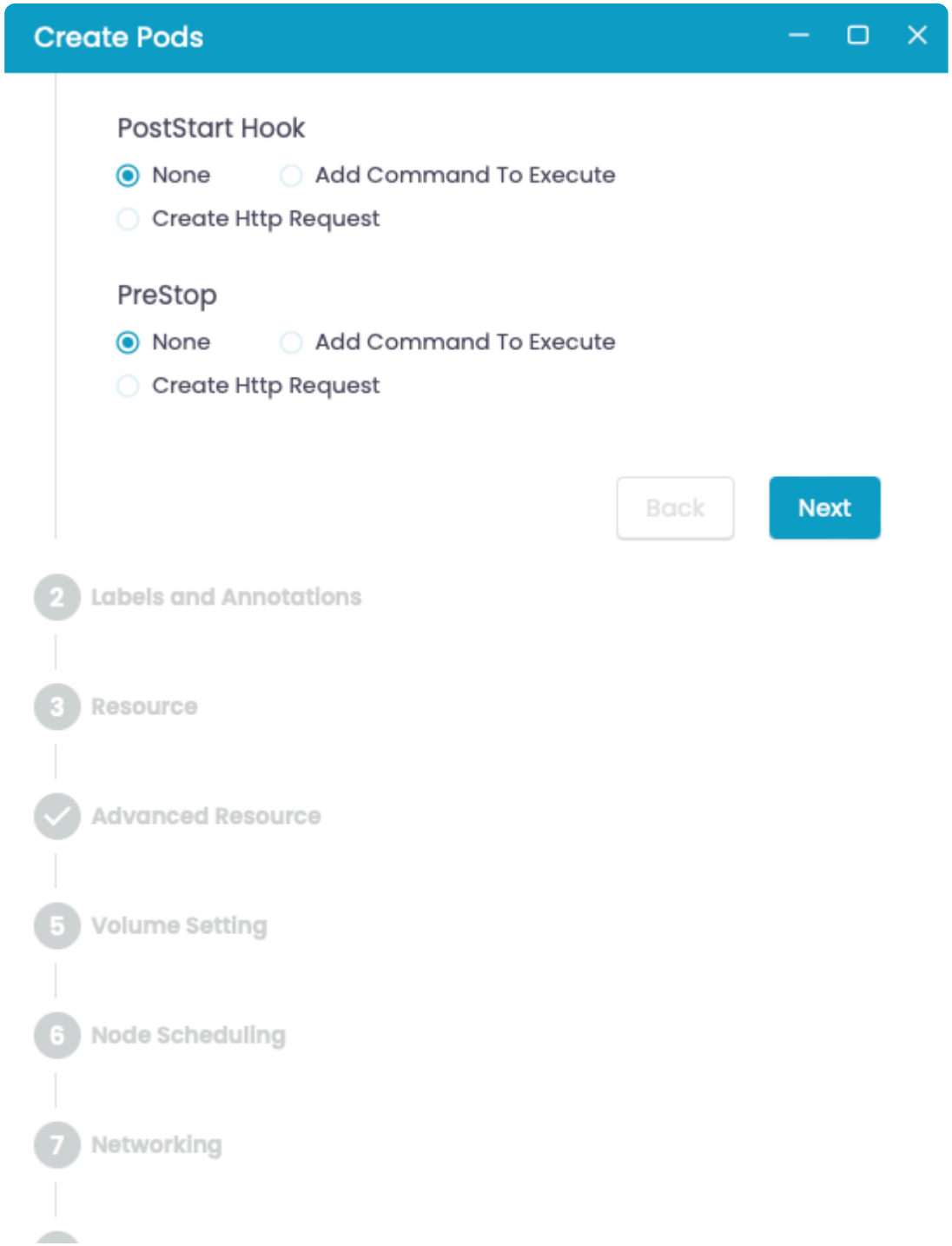

Pod Log (Name: test-pod)
Log
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-
Container Name
Please Enter Container Name
Since Time (Blank means using default value: 1 Week)
2024-06-06 22:02:18
Tail Lines (Blank means using default value: 4000)
Please Enter Tail Lines (Blank means using default value: 4000)
Limit Bytes (Blank means using default value: 1000000)
Please Enter Limit Bytes (Blank means using default value: 1000000)
Log Again

- Action：該Pod可執行的動作，包含：
  - Kubeshell：可直接於瀏覽器開啟終端機視窗，對指定的Pod以指令列的形式進行操作。



  - Update：更新該Pod的規格。點選Update後，會跳出更新表單，其操作邏輯跟建立表單類似，我們將在接下來的章節介紹。更新表單與建立表單最大的差異在於，更新表單只能針對某些特定欄位進行更新，因此相關於建立表單，能夠填寫的欄位較少。
  - Delete：即刪除該Pod。

在Pods頁面的右上角有Create Pods的按鈕，點擊後即可出現Pod的建立表單，使用者可根據自身需求設定必要的參數。由於Kubernetes上的每種資源都有很複雜的參數可與建立時進行設定，因此每種資源可設定參數的細節請參考各資源的官網資訊。Nomos會將各種參數分成數個不同的階段供使用者自行設定，若使用者有不需要設定的階段，可直接點選階段的名稱，來跳到欲修改設定的階段。以下圖的範例來說，若使用不需要設定2至8階段的參數，可直接點選Health Check，即可直接跳到第9個階段。當所有參數都依需求設定完畢後，使用者即可點選Submit按鈕來建立資源。


Create Pods
PostStart Hook
None
Add Command To Execute
Create Http Request
PreStop
None
Add Command To Execute
Create Http Request
Back
Next
2 Labels and Annotations
3 Resource
Advanced Resource
5 Volume Setting
6 Node Scheduling
7 Networking

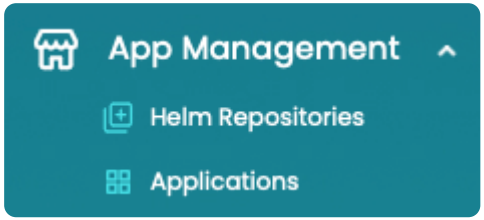
8 Security Context

9 Health Check

10 Overview

App Management

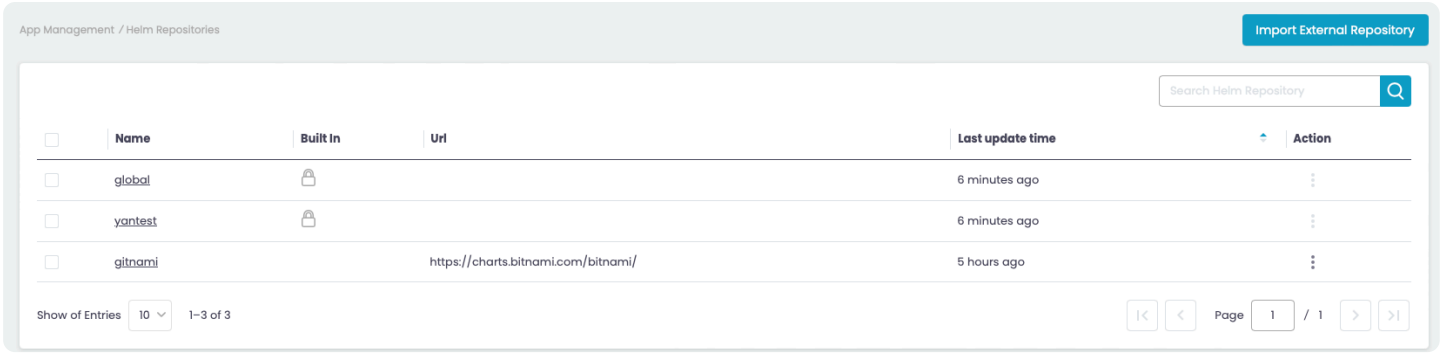
App Management整合Helm的功能，提供使用者於Nomos上管理各個Helm Repository。此外，使用者亦可透過各Helm Repository中的Chart建立Application，並於Nomos上統一管理。



Helm Repositories

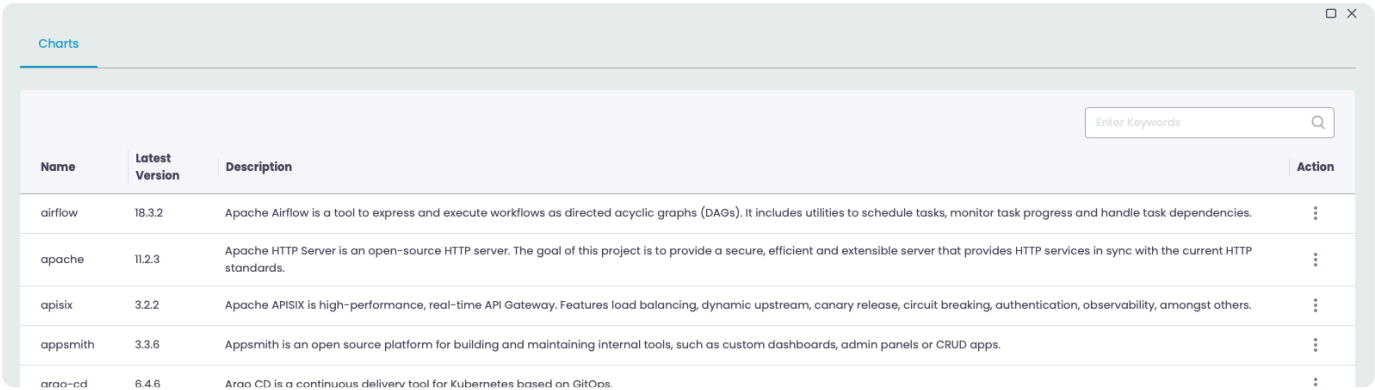
> Project User只能讀取Helm Repository的狀態，而Project Admin則可以新增、更新、刪除不同的Helm Repositories。

Helm Repositories可管理Project下的Helm Repository，每個Project預設可看到兩個Repository，分別是global與<project_name>。global為Nomos系統內建的Repository，存放一些由Nomos所提供的Chart。<project_name>為每個Project獨立的Repository，預設為空，可由使用者自行上傳編寫的Chart。

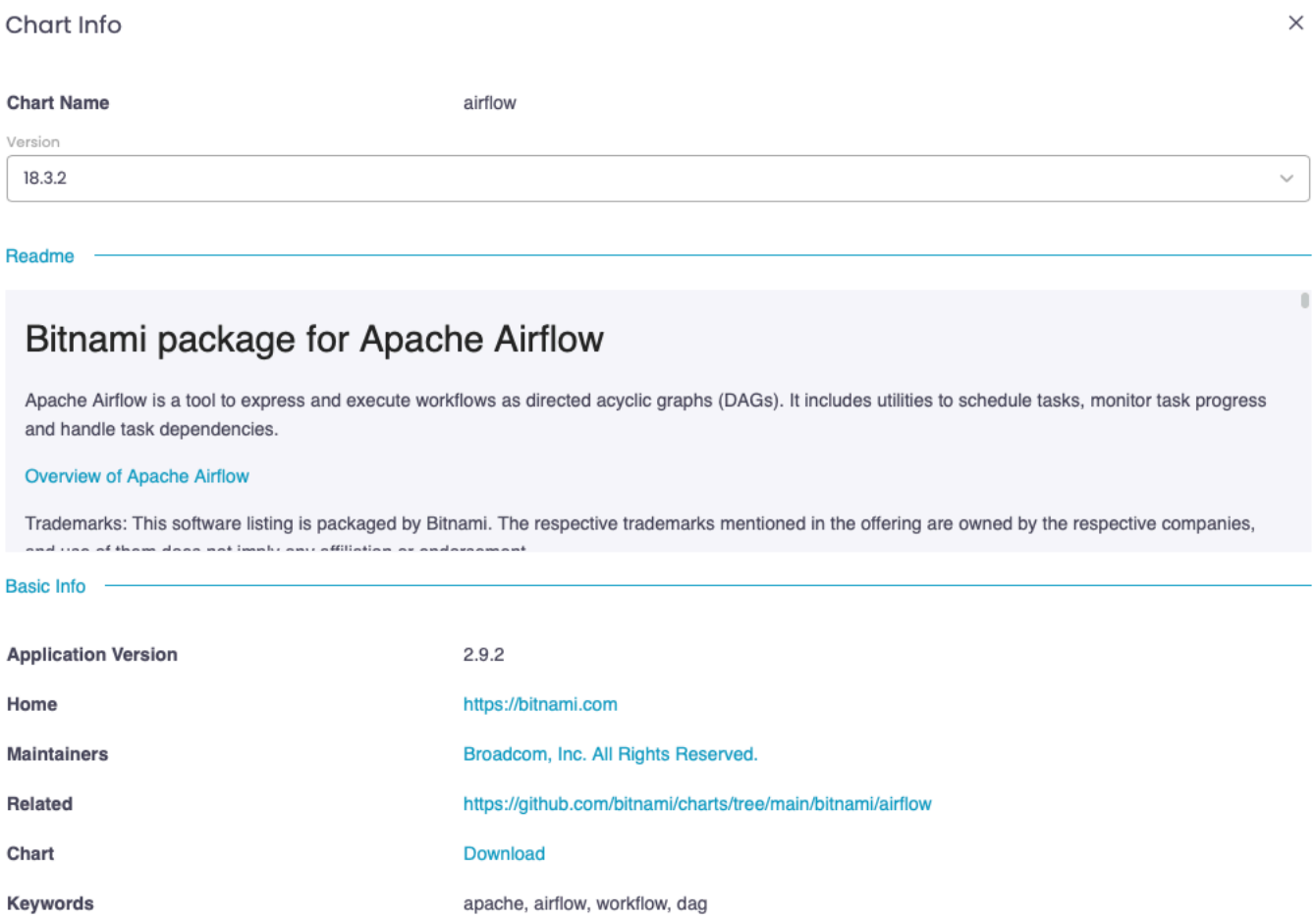


如圖所示，於Helm Repositories頁面中，使用者可以看到所有此Project下的Helm Repository，每個Helm Repository包含數個資訊，說明如下：

- Name：此Repository的名稱。點選名稱可以看到此Repository所包含的Chart。

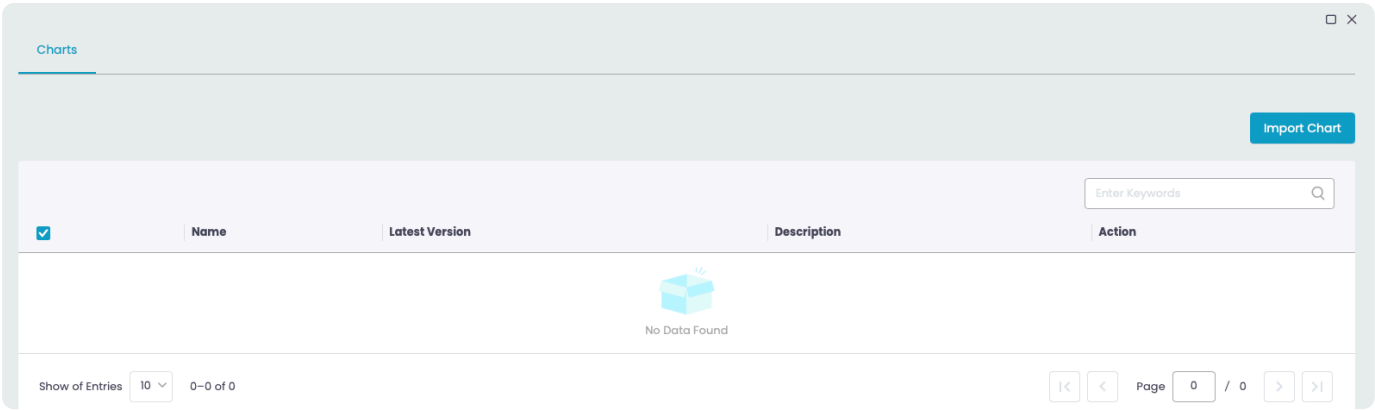


若點選Action中的View Info，則可以查看特定Chart的細節。包含Chart的版本，Readme以及基本資訊。



若點選的是預設的<project_name> Repository，則使用者還可以透過右上角的Import Chart

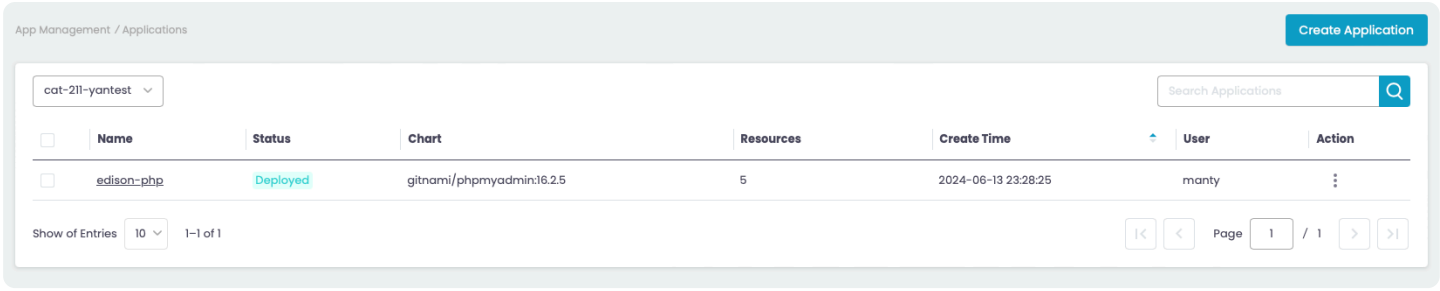
來上傳自行編寫的Chart。



- Built In：是否為Built In Repository，若為Built In Repository，則會有鎖的符號。Built In Repository是自動建立的，且無法刪除，每個Project都可看到Global與<project_name>兩個Built In Repository。
- Url：若是透過使用者自行匯入的Helm Repository，則會有URL的資訊。
- Last Update Time：Nomos會自動同步各個Helm Repository，確保所有資料都是最新的。
- Action：該Helm Repository可執行的動作，目前只有Delete，即刪除Helm Repository。且目前只能刪除非Built In的Helm Repository。

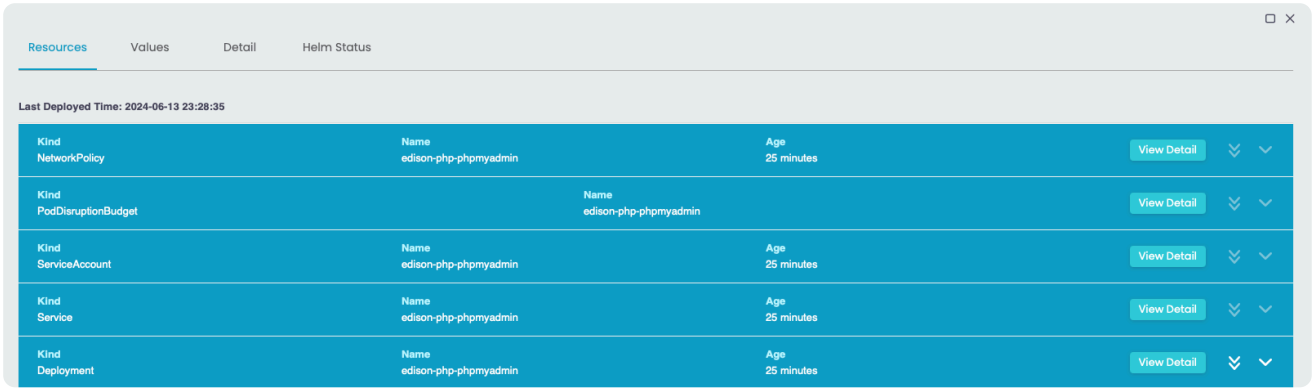
Applications

> Applications可供Project Admin與Project User使用，但Project Admin可以查看、更新和刪除所有於此Project下被建立的資源。Project User只能查看、更新和刪除由自己所建立的資源。

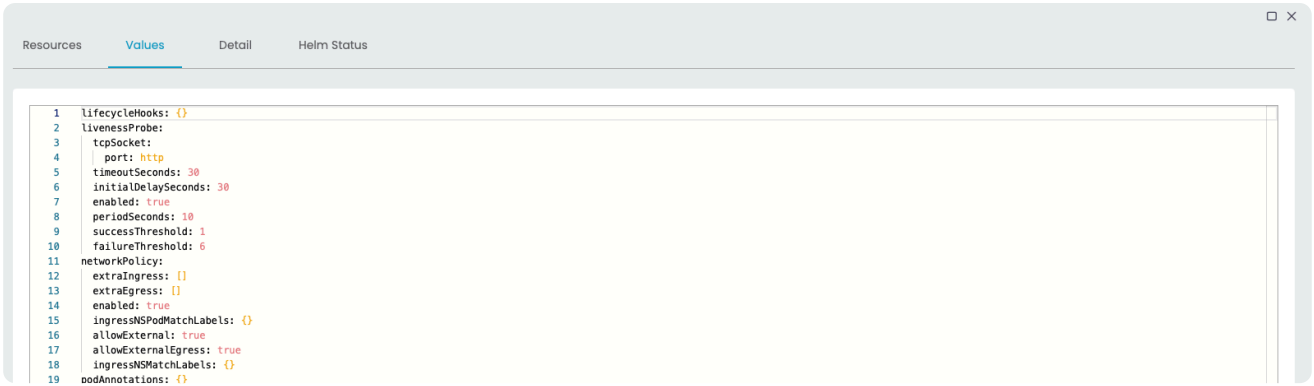


於Applications頁面中，可管理所有透過Chart建立的Application。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Application，每個Application包含數個資訊，說明如下：

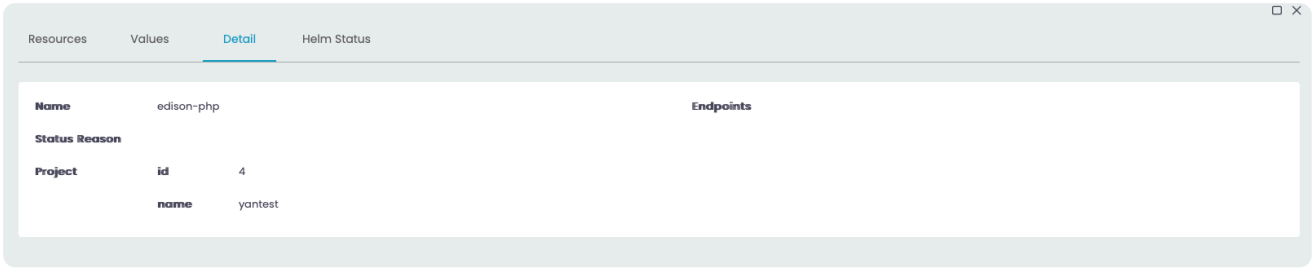
- Name：該Application的名稱。點選名稱可查看該Application的細部資料，包含
  - Resources：該Application內所有資源的狀態。每個資源於Kubernetes Resources的頁面中亦可查看對應的資訊。



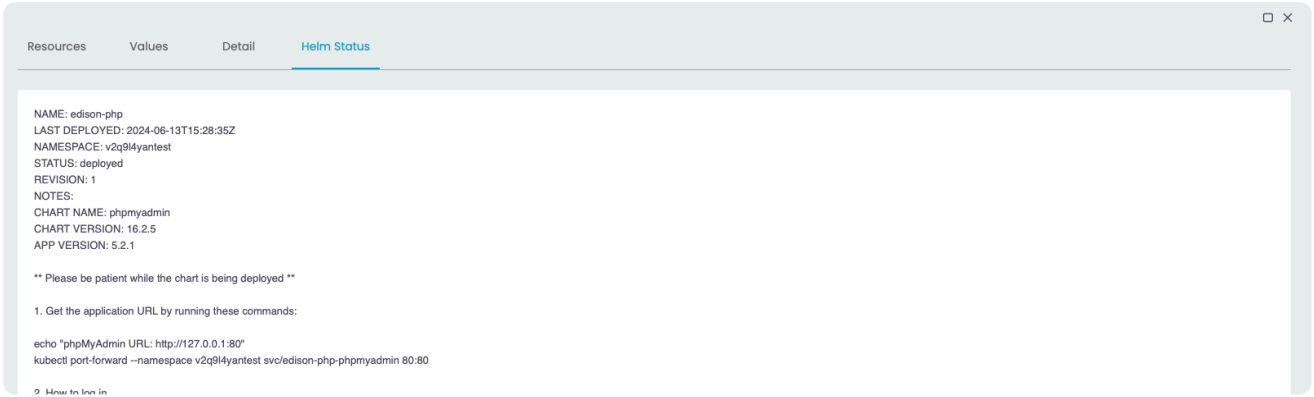
- Values：當初建立Application時所設定的Values Yaml。



- Detail：該Application的一些細部資訊。



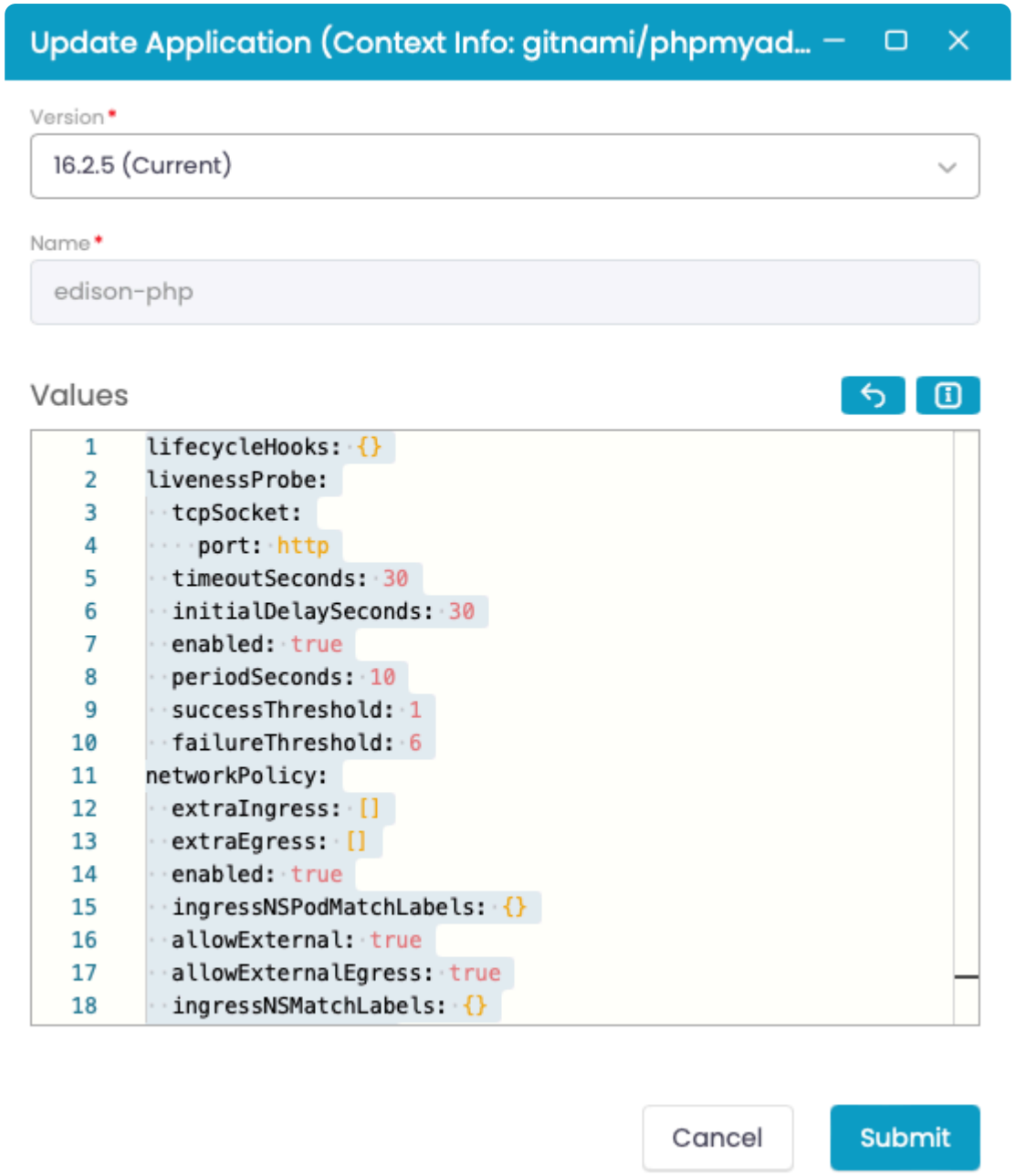
- Helm Status：該Application於Helm上的一些資訊。



- Status：該Application的狀態。
- Chart：該Application所使用的Chart。
- Resources：該Application所包含的Kubernetes Resource數量。

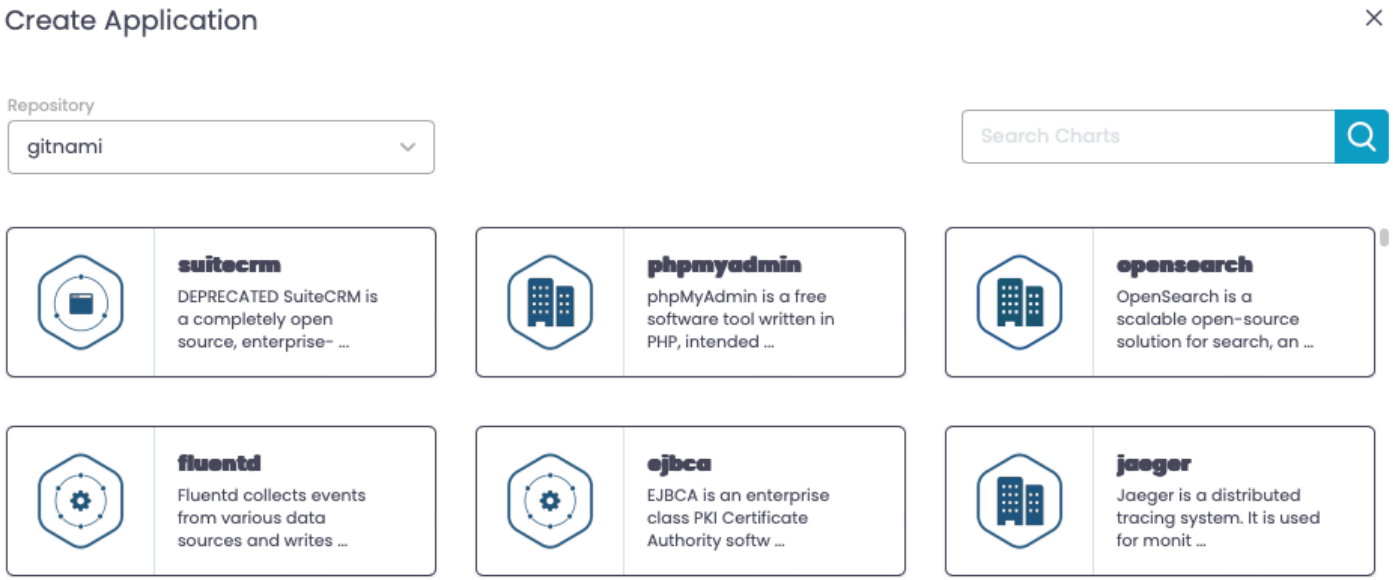
- Create Time：該Application建立的時機。
- User：建立該Application的User。
- Action：該Application可執行的動作，包含：
    - Update：更新該Application。點擊Update後系統即會跳出更新表單，更新表單類似於建立表單，建立表單在稍後的章節會有詳細的介紹。更新表單中的Application版本預設會選擇當初建立時的版本，使用者可於同一個版本中，調整Values內的參數，若想恢復為預設參數，則可點選Values右邊的reset按鈕。使用者亦可選擇更新到特定的版本，當使用者選擇了不同的版本後，Values中對應的Yaml也會跟著自動更新成對應版本的內容。一但使用

者修改完必要參數，即可點擊Submit按鈕進行更新。



- Delete：即刪除該Application。

點選畫面右上角的Create Application，即可從指定的Helm Repository中挑選適合的Chart來建立Application。



如圖所示，在點擊了Create Application後，畫面會跳出選擇表單，使用者先於左上角下拉選單中選擇適合的Repository，Nomos就會列出該Repository內中所有的Chart，接著使用者就可點選適合的Chart進行下一步。

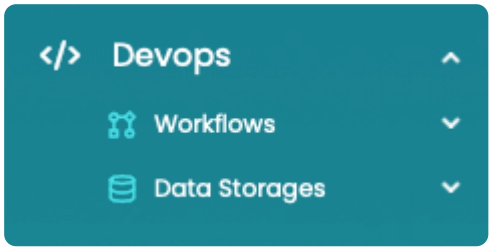
在點選適合的Chart後，系統會跳出建立表單，主要包含：

- Version：使用者可選擇想要安裝的Application版本。
- Name：使用者可決定安裝出來的Application名稱。
- Values：使用者編輯Values內的YAML檔，客製化自己的Application參數。在Values旁邊有個提示服務，點選後會顯示Values中各參數的定義，以供使用者參考。

當一旦都設定完畢後，即可點選右下角Submit按鈕來建立Application。

Devops

Devops提供使用者自行定義CICD的流程，並提供介面整合Gitlab、SonarQube、Zap等常見的CICD工具。透過Devops，使用者可藉由定時或Gitlab事件來觸發CICD流程，以達到自動化的建置、測試、部署軟體應用之情境，並可設定通知名單，將CICD執行結果透過Email方式通知對應的使用者。

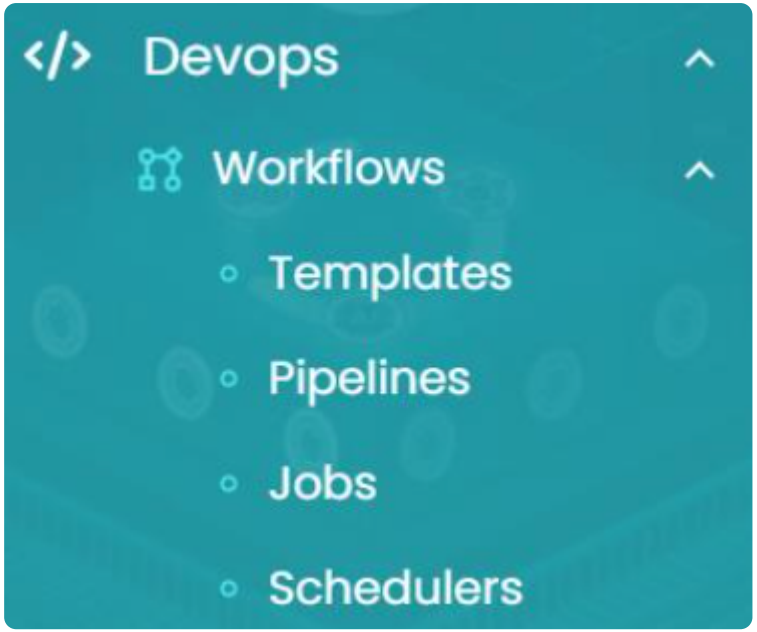


Workflows

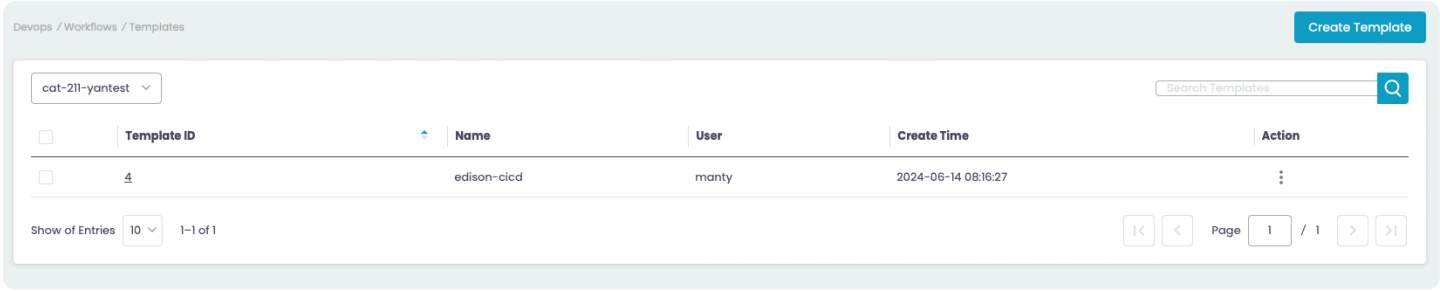
> Workflows下的各種功能可供Project Admin與Project User使用，但Project Admin可以查看、更新和刪除所有於此Project下被建立的資源。Project User只能查看、更新和刪除由自己所建立的資源。

使用者可於Workflow中建立、修改、刪除自行定義的CICD流程，亦可查看每次CICD的執行結果。Workflow基本上由Template、Pipeline與Job組成，Template即為一個使用者自行定義的CICD流程，而Pipeline則為以某個Template實際執行的結果，由於一個Template是由許多工作所組成，因

此在每個Pipeline中，會由許多Job組成，每個Job即為代表一個Template中特定工作的執行結果。接下來的章節將詳述各個功能的操作介面。



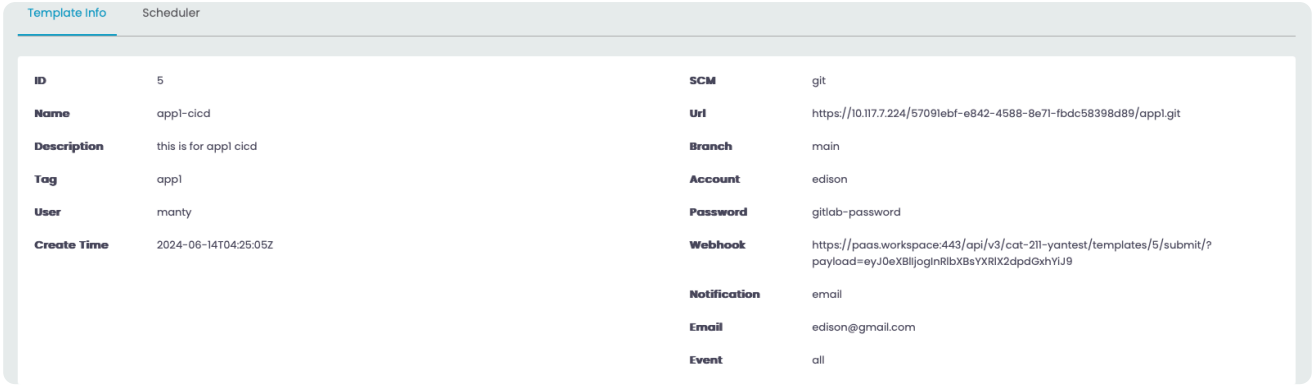
Templates



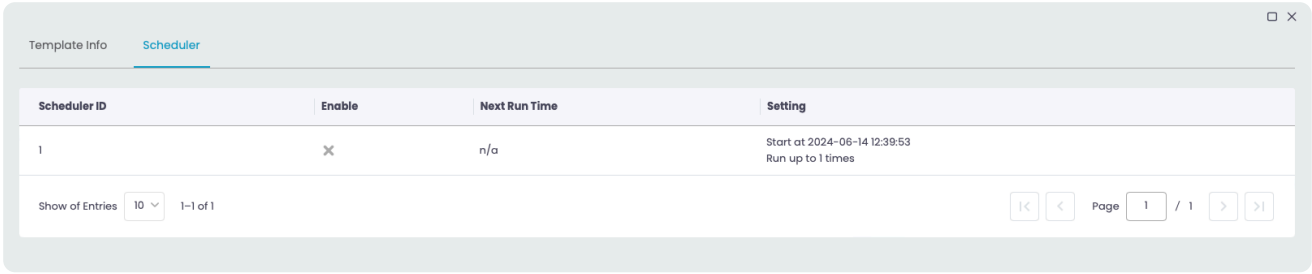
於Templates頁面中，可管理使用者所建立的Templates。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Template，每個Template包含數個資訊，說明如下：

- Template ID：該Template的ID。點擊ID可查看Template的細部資訊，包含：

- Template Info：主要包含Template的Basic Setting與Advanced Setting的相關設定。特別注意在Template Info裡有Webhook的URL資訊，此資訊需要設定至Gitlab上才可完成Webhook的設定，詳情將於後續章節進行說明。



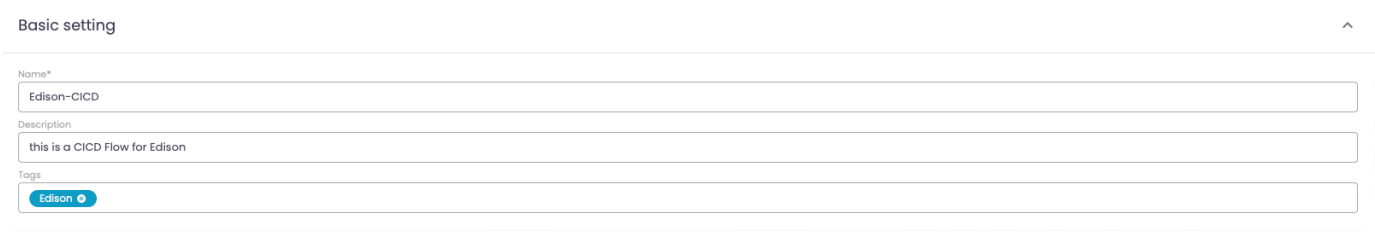
- Scheduler：關聯至此Template上的Scheduler資訊，包含Scheduler ID、Enable、Next Run Time和Setting。相關細節將於後續章節進行說明。



- Name：該Template的名稱。
- User：建立該Template的使用者。
- Create Time：該Template的建立時間。
- Action：該Template可執行的動作，包含：
  - Run Pipeline Immediately：直接執行Template，不修改環境變數。
  - Run Pipeline：執行Template，並在執行前提供介面修改環境變數。
  - Update：更新Template內容。
  - Duplicate：複製Template。
  - Delete：刪除該Template。當刪除Templaet時，也會一併刪除其所屬之Pipeline與Job。

在Templates頁面的右上角有Create Template的按鈕，點擊後即可出現Template的建立表單，使用者可根據自身需求設定必要的參數。Template建立表單主要分為三個部分，Basic Setting、Template Setting、Advanced Setting，以下分別就各部分進行說明。

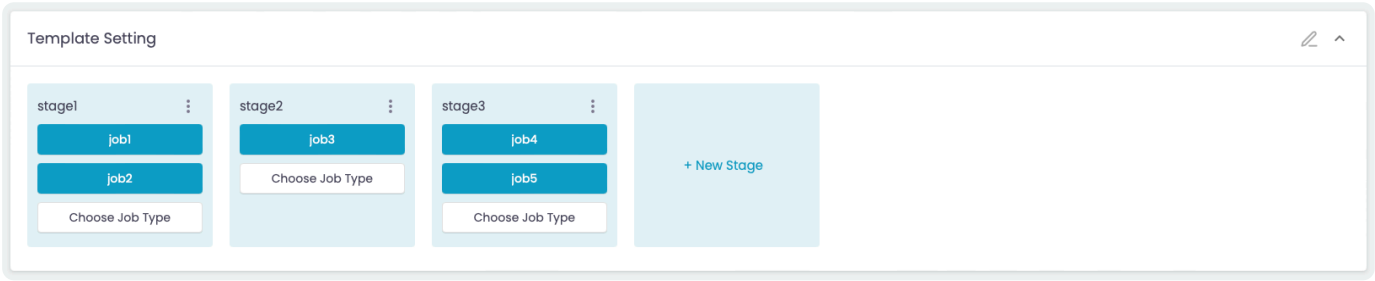
- Basic Setting



Basic Setting中可設定此Template的基本資訊，包含：

  - Name：此Template之名稱。
  - Description：此Template的額外描述。
  - Tags：此Template之標籤。使用者可輸入多個標籤，設定標籤的目的主要是為了在後續的流程中查看特定標籤的執行結果。

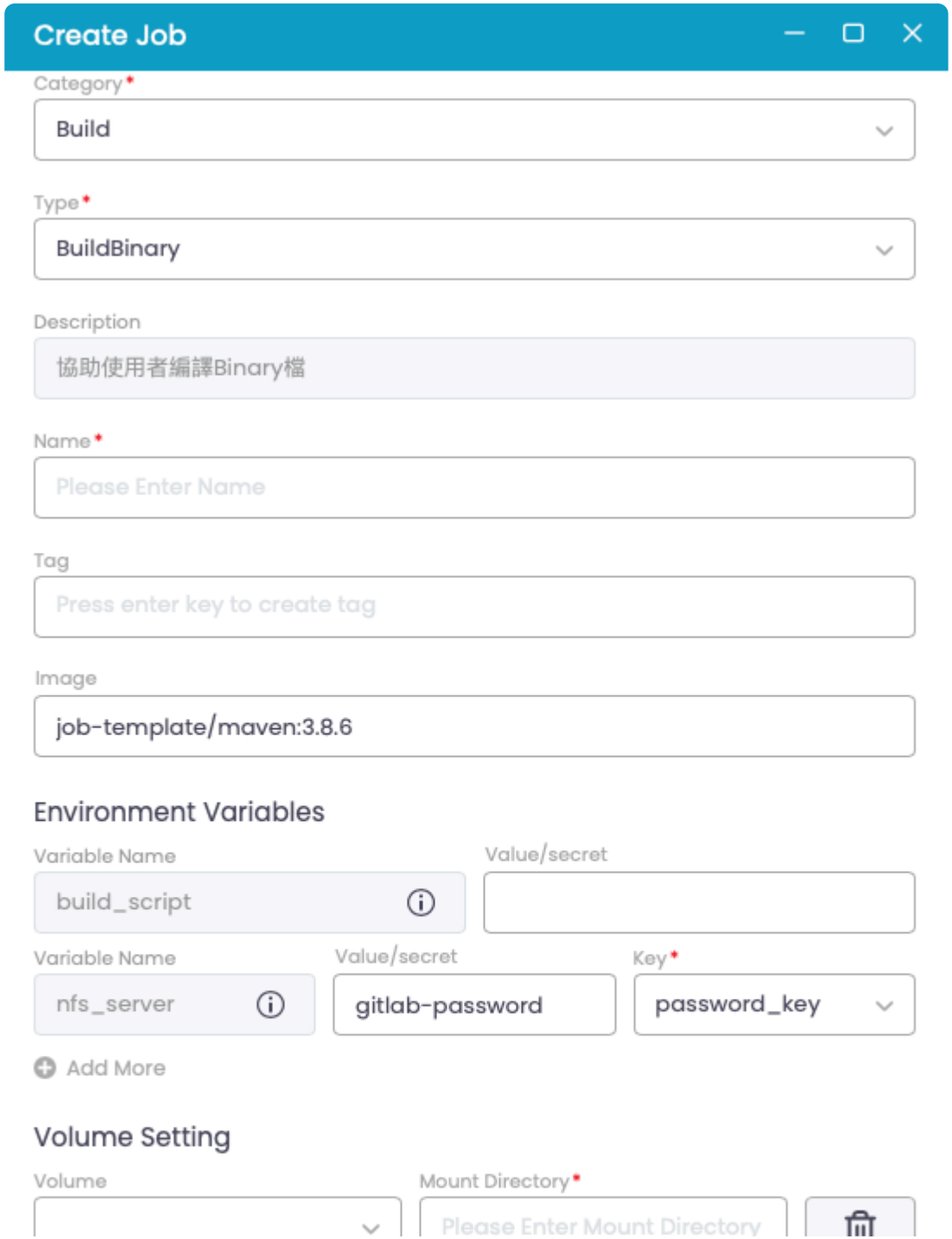
- Template Setting

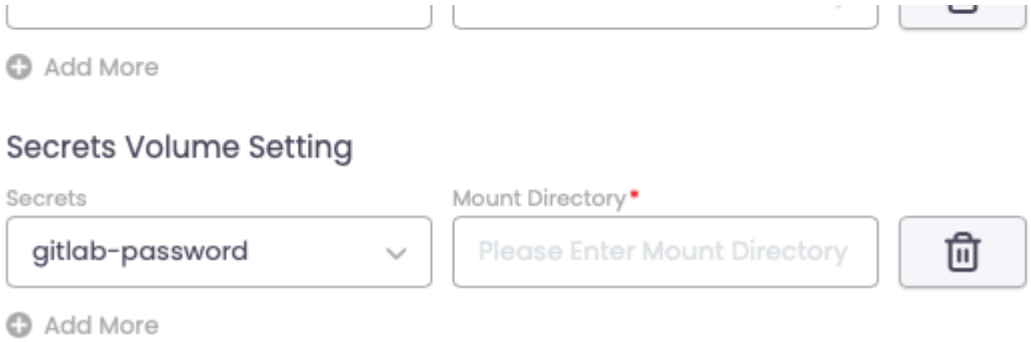


如圖所示，使用者可將整個流程分成數個Stage，每個Stage中可包含1到多個Job。使用者可點選畫面上的+ New Stage來新增Stage，亦可點選Stage中的Choose Job Type來新增特定Stage中的Job。當Template在執行時，Nomos會由左至右依需執行每個Stage，當某個Stage執行時，會同時執行該Stage中所有的Job。若某個Stage中的任一Job執行發生問題，則整個流程就會停在該Stage，等待使用者確認原因並進行修復。

當使用者點選Choose Job Type時，畫面會跳出兩種Job Type供使用者進行選擇，分別為Built-in Job與Customized Job。

- Built-in Job：Nomos內建的Job類型，提供CICD相關的工具可供整合。當點選Built-in Job後，系統會跳出建立表單，主要包含：


Create Job
Category*
Build
Type*
BuildBinary
Description
協助使用者編譯Binary檔
Name*
Please Enter Name
Tag
Press enter key to create tag
Image
job-template/maven:3.8.6
Environment Variables
Variable Name
Value/secret
build_script
Variable Name
Value/secret
Key*
nfs_server
gitlab-password
password_key
Add More
Volume Setting
Volume
Mount Directory*
Please Enter Mount Directory



- Category：指Job的種類，包含Build、Deployment、Unit Test等CICD中常見的種類，使用者可根據需求自行選擇。
- Type：在選擇Category後，即可選擇此Category下所屬的Type。例如當選擇Deployment時，就有Ansible與KubectI兩種Type可供使用者選擇。
- Description：當選擇Type後，Description即顯示相關之描述。
- Name：此Job之名稱。
- Tag：此Job之標籤。使用者可輸入多個標籤，設定標籤的目的主要是為了在後續的流程中查看特定標籤的執行結果。
- Image：此Job所可以使用的Image。有些Job有內建的Image可供選擇，有些Job則需要使用者指定自己上傳至Harbor的Image。
- Environment Variable：此Job所需輸入的環境變數，根據使用者所選擇的Type，會有不同的環境變數需要使用者輸入。每個環境變數的欄位有三個，若變數旁有垃圾桶的符號，表示此變數為選填欄位：
  - Variable Name：此變數的名稱，由系統決定。在變數的名稱旁邊有提示符號，點擊提示符號可看到關於此變數的說明。
  - Value/Secret：針對環境變數，可選擇兩種方式輸入。如果要直接輸入值，可在輸入值後按Enter確定，如果要以Data Secret的方式輸入，則選擇對應的Secret。
  - Key：若選擇透過Data Secret的方式輸入，則需指定所選Secret中的Key。

- Volume Setting：如有需要，例如想讀取特定的資料，或將Job執行結果儲存起來，使用者可將特定Volume掛載到此Job的執行環境中。使用者必須先於Data Volumes中建立對應的Volume，詳細的做法將在稍後的章節說明。欲掛載Volume，點選Add more即可增加Volume輸入欄位，亦可點選Volume輸入欄位旁的垃圾桶符號來刪除Volume輸入欄位。每個Volume包含兩個欄位：
  - Volume：以下拉式選單，選擇使用者欲掛載的Data Volume。
  - Mount Directory：指定使用者欲將Volume掛載到Job執行環境中的哪個路徑之下。
- Secrets Volume Setting：如果需要，例如做為密碼、憑證或參數檔使用，使用者亦可將Secret當作Volume掛載到此Job的執行環境中。使用者必須先於Data Secret中建立對應的Secret，詳細的做法將在稍後的章節說明。欲掛載Secret Volume，點選Add more即可增加Secret Volume輸入欄位，亦可點選Secret Volume輸入欄位旁的垃圾桶符號來刪除Volume輸入欄位。每個Secret Volume包含兩個欄位：
  - Secret：以下拉式選單，選擇使用者欲掛載的Data Secret Volume。
  - Mount Directory：指定使用者欲將Secret Volume掛載到Job執行環境中的哪個路徑之下。

當所有資料都填寫完畢後，即可點擊下方的Submit按鈕建立Job。

- Customized Job：若Built-in Job無法達到使用者的需求，使用者可選擇透過Customized Job的方式來客製化自己的Job。當點選Customized Job後，系統會跳出建立表單，建立表單主要分成三個部分：

- Basic Info：填寫Job最基本的資訊，包含：

1 Basic Info *

Job Name *

Please Enter Job Name

Image Pull Secret

No Secret

Image *

Please Enter Image

Flavor *

flavor1

1 Cores + 1024 MB memory

Command Type

Shell Script Command & Argument

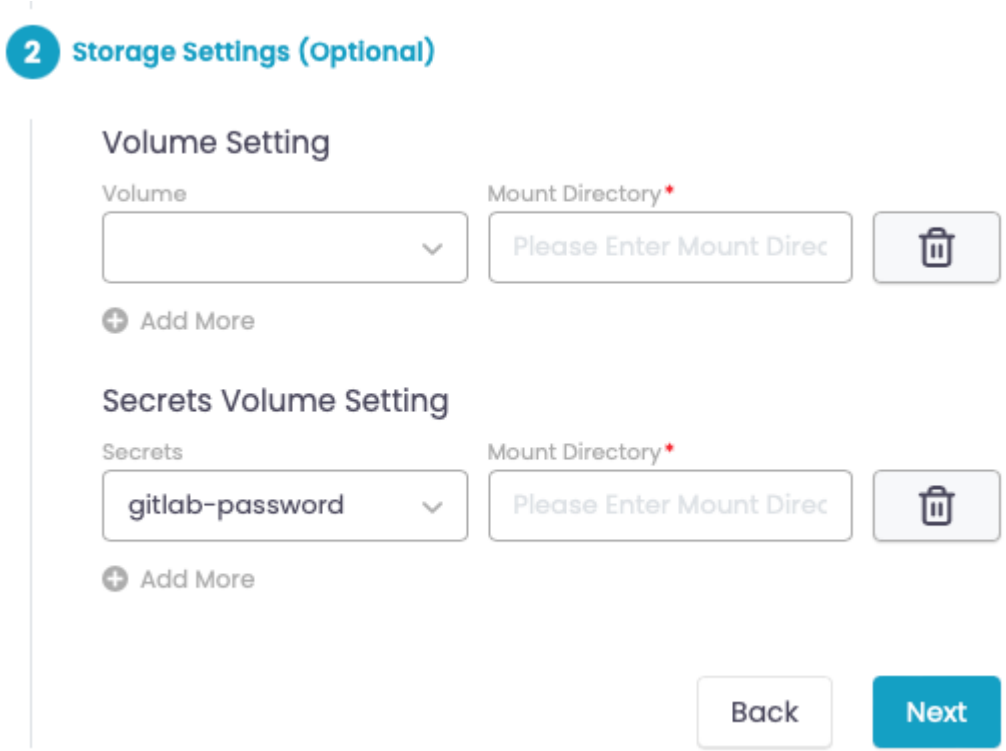
Command *

Please Enter Command
(e.g.
while true
do echo hello
sleep 10
done
)

Back Next

- Job Name：此Job的名稱。
- Image Pull Secret：若Job欲使用的Image有於Harbor上有設定權限，可事先於Data Secret建立對應的Secret，在執行Job時才可順利從Harbor上Pull Image至本地端。
- Image：此Job欲使用的Image，其路徑應來自Habor。
- Flavor：此Job欲使用的Flavor。
- Command Type：使用者可自行定義此Job要執行的腳本，目前有兩種方式可供定義，分別是Shell Script與Command & Argument
  - Shell Script：最基本的Shell Script，使用者可隨需求自行編寫需執行的指令。
  - Command & Argument：以Kubernetes的command與args語法進行定義，詳細用法可參考Kubernetes官網 (https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/)

- Storage Setting：若需要掛載Volume，則可在Storage Setting中進行設定，相關選項包含：

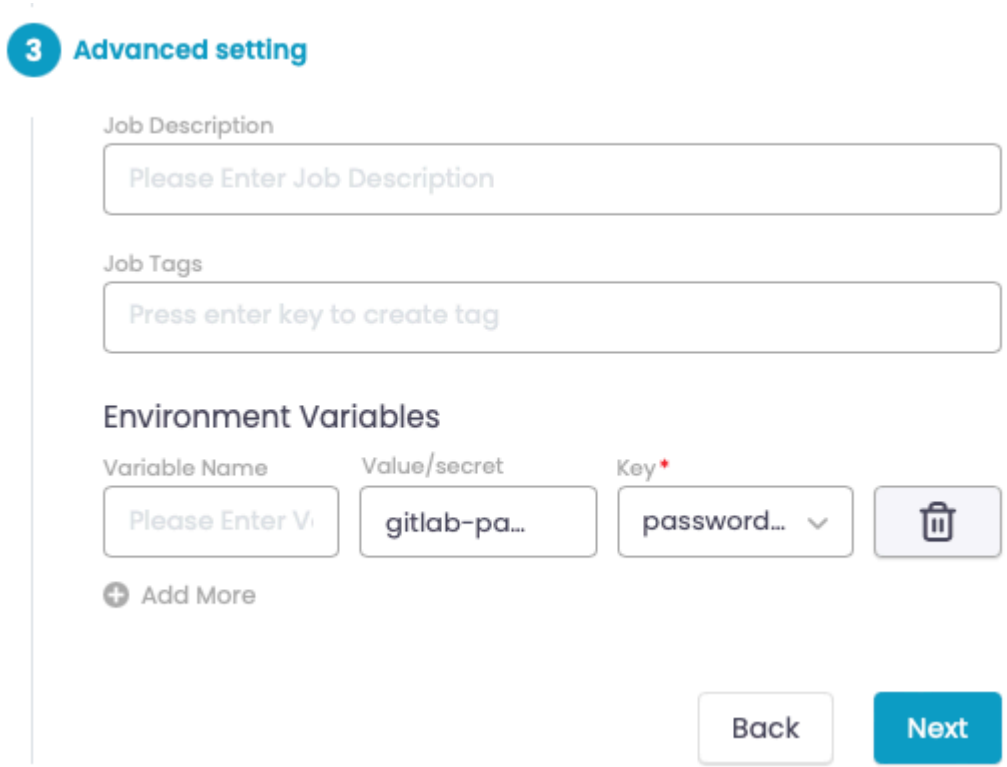


- Volume Setting：如有需要，例如想讀取特定的資料，或將Job執行結果儲存起來，使用者可將特定Volume掛載到此Job的執行環境中。使用者必須先於Data Volumes中建立對應的Volume，詳細的做法將在稍後的章節說明。欲掛載Volume，點選Add more即可增加Volume輸入欄位，亦可點選Volume輸入欄位旁的垃圾桶符號來刪除Volume輸入欄位。每個Volume包含兩個欄位：
  - Volume：以下拉式選單，選擇使用者欲掛載的Data Volume。
  - Mount Directory：指定使用者欲將Volume掛載到Job執行環境中的哪個路徑之下。
- Secrets Volume Setting：如果需要，例如做為密碼、憑證或參數檔使用，使用者亦可將Secret當作Volume掛載到此Job的執行環境中。使用者必須先於Data Secret中建立對應的Secret，詳細的做法將在稍後的章節說明。欲掛載Secret Volume，點選Add more即可增加Secret Volume輸入欄位，亦可點選Secret

Volume輸入欄位旁的垃圾桶符號來刪除Volume輸入欄位。每個Secret Volume包含兩個欄位：

- Secret：以下拉式選單，選擇使用者欲掛載的Data Secret Volume。
- Mount Directory：指定使用者欲將Secret Volume掛載到Job執行環境中的哪個路徑之下。

- Advanced Setting：其他進階設定可在Advanced Setting中設定，包含：



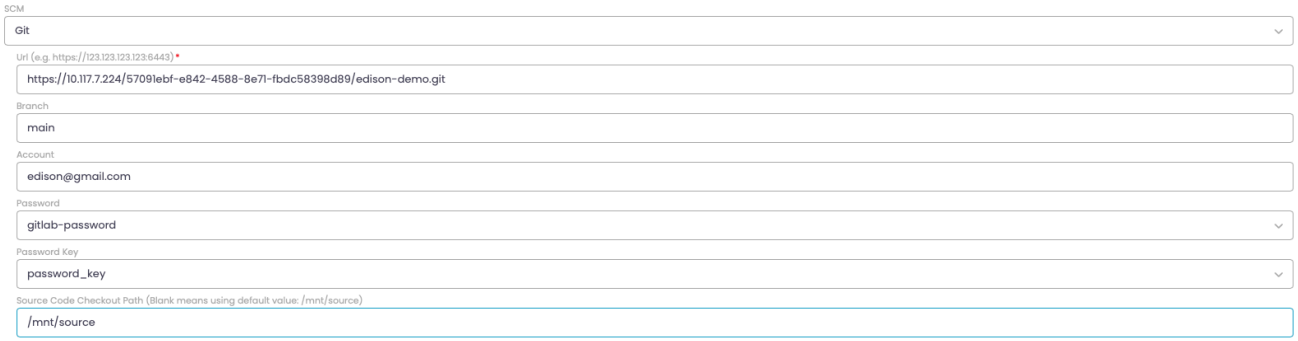
  - Job Description：此Job的額外描述。
  - Job Tags：此Job之標籤。使用者可輸入多個標籤，設定標籤的目的主要是為了在後續的流程中查看特定標籤的執行結果。
  - Environment Variable：若使用者需要額外的環境變數，可點擊add more來增加環境變數。每個環境變數的欄位有三個，若有不需要的環境變數，可點擊變數旁的垃圾桶刪除：

        - Variable Name：此變數的名稱。
        - Value/Secret：針對環境變數，可選擇兩種方式輸入。如果要直接輸入值，可在輸入值後按Enter確定，如果要以Data Secret的方式輸入，則選擇對應的Secret。
        - Key：若選擇透過Data Secret的方式輸入，則需指定所選Secret中的Key。

    當所有資料都填寫完畢後，即可點擊下方的Submit按鈕建立Job。

- Advanced Setting
  Advanced Setting中可設定此Template的進階功能，包含：

    - SCM：使用者可於Source Code Management(SCM)中設定是否要整合Gitlab專案，預設的選項為None，即不整合Gitlab專案，若使用者選擇Git後，在執行CICD時，Nomos會自動將所設定的Gitlab專案 Source Code Checkout到執行CICD工作的環境中。需額外輸入相關資訊，包含：

        - URL：欲整合之Gitlab專案的URL，需確認此Gitlab服務處於Nomos可連接的網路環境。
        - Branch：欲整合之Gitlab專案的Branch。
        - Account：使用者於該Gitlab上所使用的帳號。
        - Password：使用者事先建立的Data Secret，使用者需於此Data Secret中儲存所使用的Gitlab帳號所對應的密碼。Data Secret的建立方式將於之後的章節說明。

- Passowrd Key：該Data Secret中對應Gitlab密碼的Key。
- Source Code Checkout Path：Source Code Checkout Path可指定使用者欲將Source Code Checkout到CICD環境中的哪個路徑下，預設值為/mnt/source。

- Webhook：使用者可於Webhook中設定是否要整合Gitlab的Webhook功能，預設的選項為None，即不整合Gitlab Webhook，若使用者選擇Git後，當Gitlab專案有對應的事件發生，例如Commit、Merge Request等，則Nomos會接到通知，並自動執行所設定的Template。需要額外輸入相關資訊，包含：

  - Secret：使用者事先建立的Data Secret，使用者需於此Data Secret中儲存自定義的Gitlab Webhook Token，並於Gitlab上輸入對應的Token，Webhook的機制才會生效。Gitlab上的設定方式請參考Gitlab官方文件(https://docs.gitlab.com/ee/user/project/integrations/webhooks.html)。
  - Secret Key：該Data Secret中對應Gitlab Webhook Token的Key。

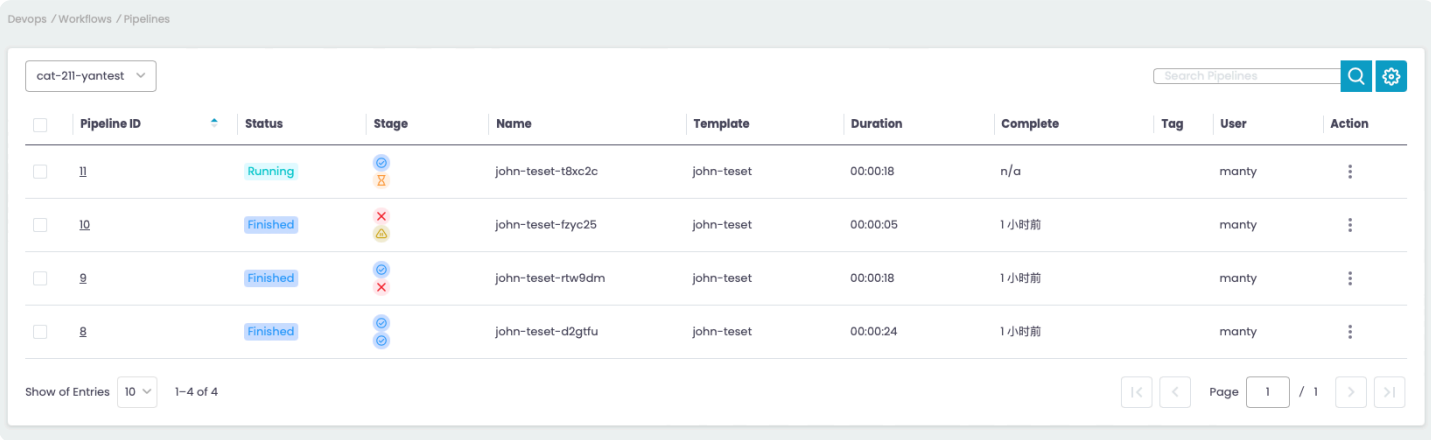
- Notification：使用者可於Notification中設定是否要設定事件通知，預設的選項為None，即不進行事件通知，若使用者選擇Email後，當Template執行結果符合條件時，即會透過Email通知指定的使用者。若選擇Email需額外輸入相關資訊，包含：

  - Email：欲通知人員的Email名單。
  - Event：使用者可選擇不同的通知事件，目前有All Event與Error Event兩種事件。
    - All Event：每次Template執行後都會通知使用者。

    - Error Event：當Template執行失敗後會通知使用者。

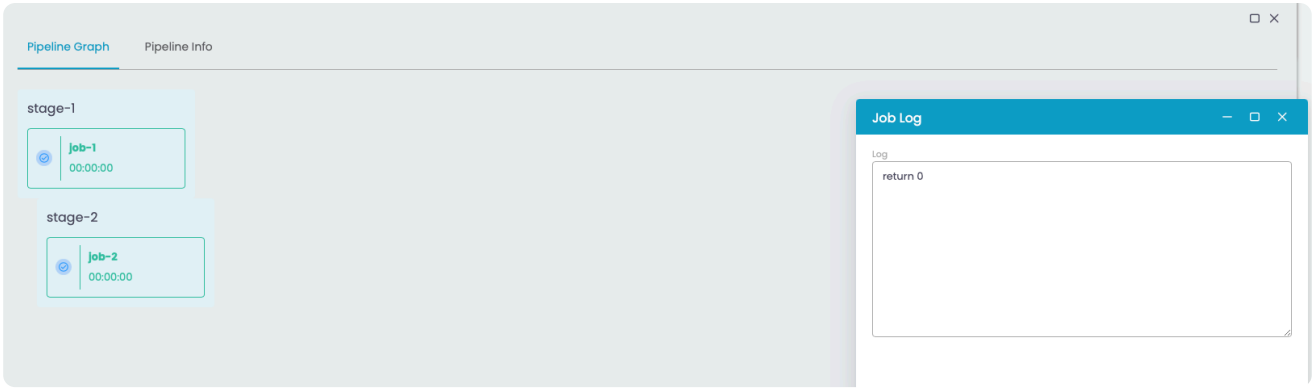
當所有資料都填寫完畢後，即可點擊下方的Confirm按鈕建立Template。

Pipelines

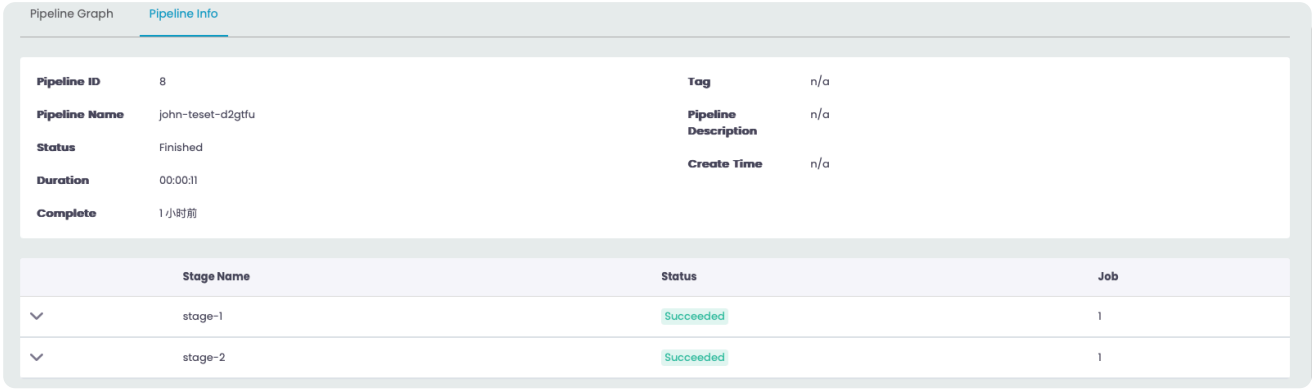


Pipeline為每次Template執行後所產生的結果紀錄，於Pipelines頁面中，我們可以查看所有的Pipeline狀態。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Pipelines，每個Pipeline包含數個資訊，說明如下：

- Pipeline ID：此Pipeline的ID。點擊ID可查看此Pipeline的細部資料，包括
    - Pipeline Graph：以圖形化的介面呈現此Pipeline的執行結果。點擊特定Job可跳出關於該Job的執行Log。



- Pipeline Info：提供該Pipeline的基本資訊，以及此Pipeline中每個Job各別的執行結果。

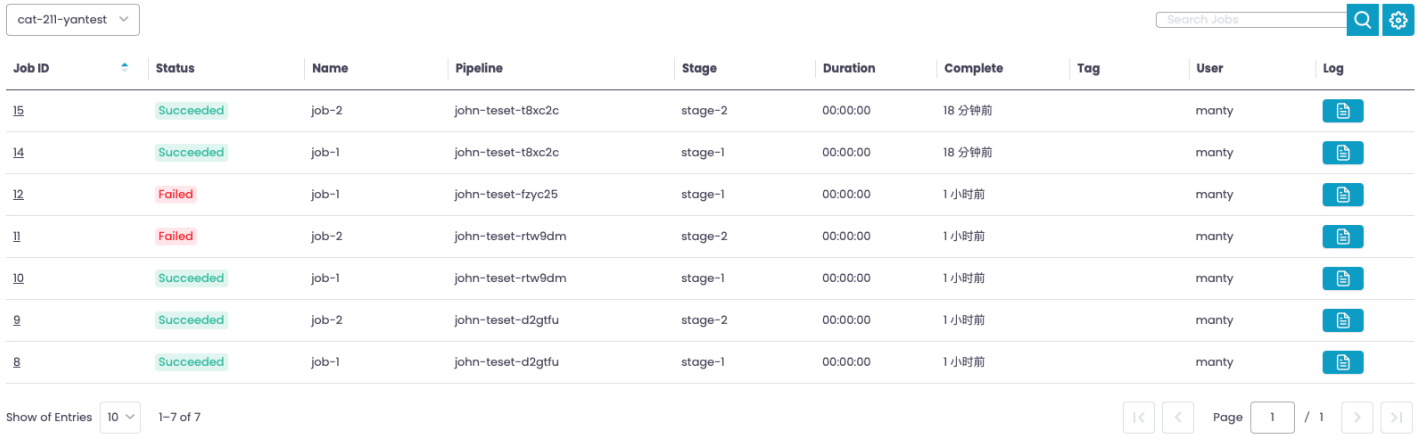


點擊特定Job，系統會跳出該Job的細部資訊。

- Status：此Pipeline的狀態，包含Running和Finished等。
- Stage：此Pipeline各Stage的執行狀態，包含Running、Pedning、Failed、Cancelled和Succeeded等。
- Name：此Pipeline的名稱，若無指定則由系統根據Template名稱自動產生。
- Template：此Pipeline所屬Template。
- Duration：此Pipeline執行時間。
- Complete：此Pipeline結束的時間。

- Tag：此Pipeline之標籤，於Template中所定義。使用者可透過標籤搜尋相關的Pipeline紀錄。
- User：建立此Pipeline所屬Template之使用者。
- Action：此Pipeline可執行的動作，目前只有Delete這個選項。刪除Pipeline會一併刪除屬於該Pipeline的Job。

Jobs

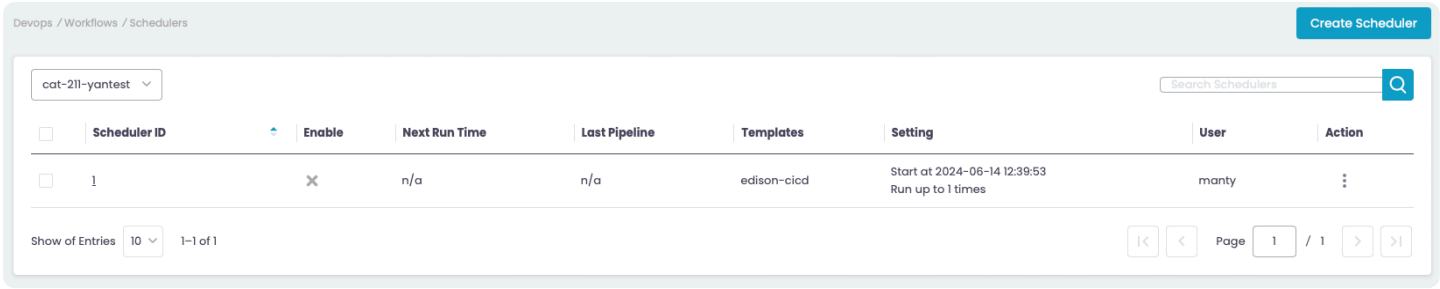


Pipeline由數個Job所組成，當使用者想直接查看關於Job的詳細資料，則可於Jobs頁面查看。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Job，每個Job包含數個資訊，說明如下：

- Job ID：此Job的ID。點擊ID可查看Job的細部資訊，包括當初建立Job所指定的參數。

- Status：此Job的Status，包括Succeeded、Failed、Running等。
- Name：此Job的名稱，乃於Template中定義。
- Pipeline：此Job所屬Pipeline名稱。
- Stage：此Job屬於Pipeline中的哪個Stage。
- Duration：此Job執行時間。
- Complete：此Job結束的時間。
- Tag：此Job之標籤，於Template或Job中所定義。使用者可透過標籤搜尋相關的Pipeline紀錄。
- User：建立此Job所屬Template之使用者。
- Log：點擊Log，可查看此Job執行時所產生的Log。

Schedulers

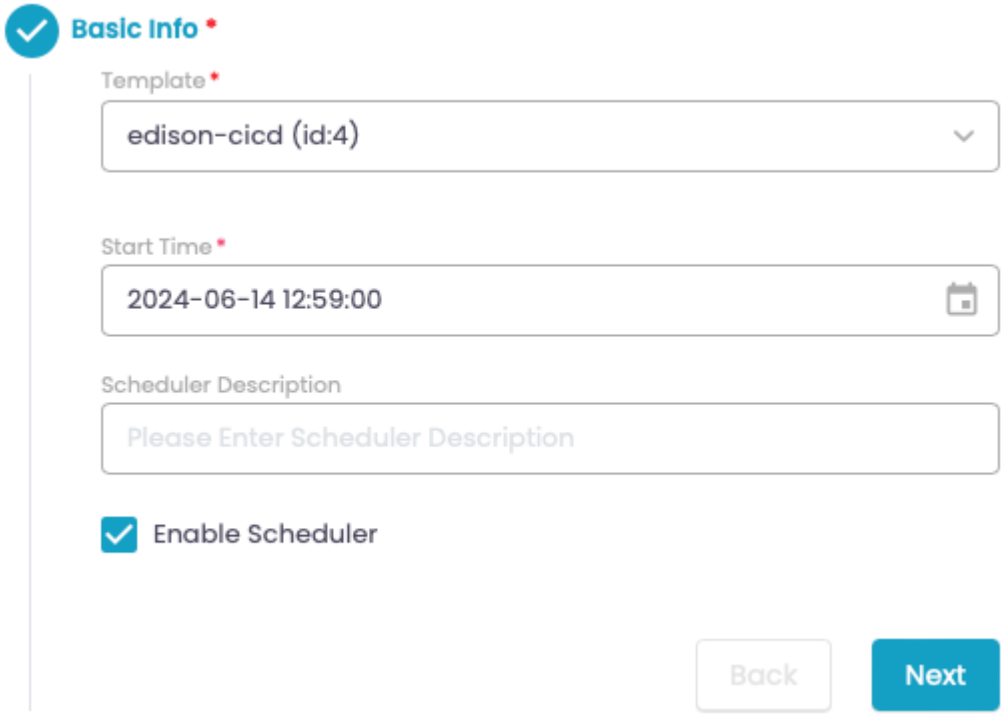


於Schedulers頁面中，可管理使用者所建立的Scheduler。Scheduler的目的是可以依照排程定時或於特地時間執行特定的Template。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Scheduler，每個Schuduler包含數個資訊，說明如下：

- Scheduler ID：此Scheduler的ID。
- Enable：此Scheduler是否啟用。只有Scheduler啟用時，才會於指定的時間執行，當Scheduler執行時，就會觸發對應的Template執行。
- Next Run Time：此Scheduler下次的執行時間。此資訊會根據使用者設定的條件即時呈現下次Scheduler執行時間，以供使用者參考。
- Last Pipeline：此Scheduler於上次觸發Template執行時，所對應的Piepline資訊。
- Templates：此Scheduler所對應的Template。當Scheduler執行時，會觸發對應的Template執行。
- Setting：此Scheduler的執行條件說明。
- User：建立此Scheduler的使用者。
- Action：此Scheduler可執行的操作，包括：
  - Update：即修改此Scheduler的設定。
  - Delete：即刪除此Scheduler。

在Schedulers頁面的右上角有Create Scheduler的按鈕，點擊後即可出現Scheduler的建立表單，使用者可根據自身需求設定必要的參數。Template建立表單主要分為三個部分，Basic Setting、Frequency Setting、Advanced Setting，以下分別就各部分進行說明。

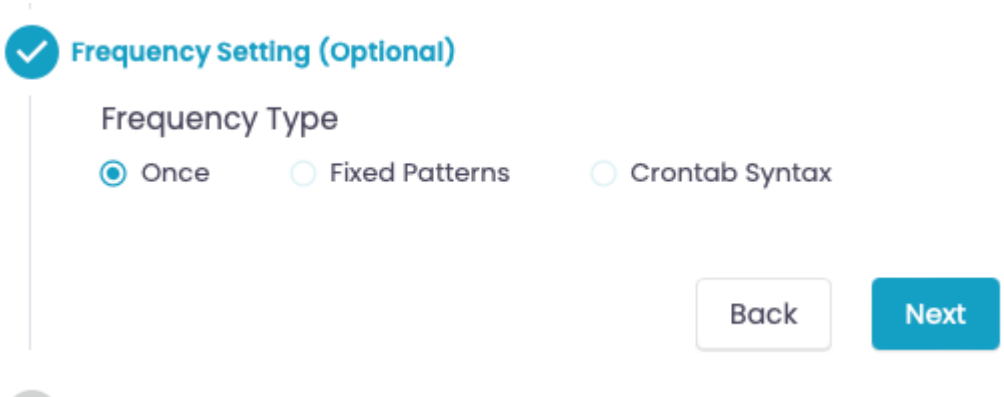
- Basic Setting：此Scheduler的基本資訊，包含：



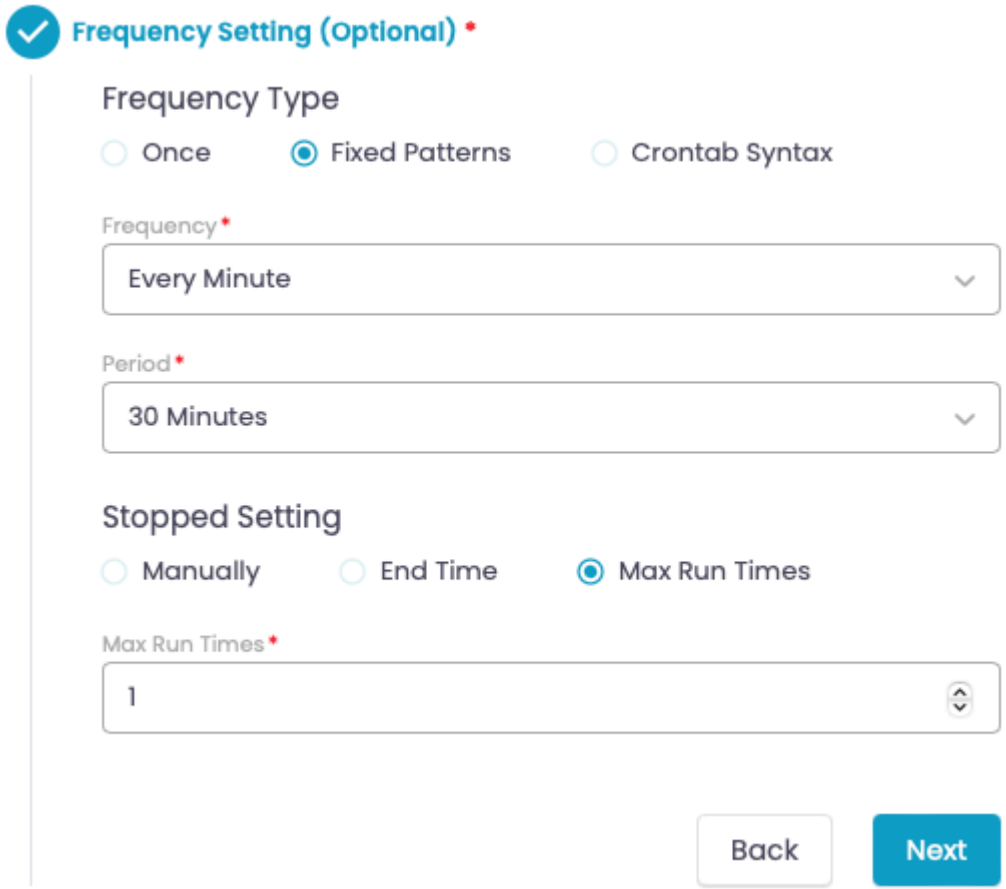
  - Template：由下拉選單選擇此Scheduler所欲關聯之Template。
  - Start Time：此Scheduler開始運作的時間。例如當使用者設定了Scheduler每天執行一次，但他希望這個設定在三天後才開始運作，則可透過Start Time來控制。
  - Scheduler Description：此Scheduler的額外描述。
  - Enable Scheduler：此Scheduler是否啟用。例如當使用者設定了Scheduler每天執行一次，但他還不確定此設定要何時開始運作，則可透過Enable來控制。

- Frequency Setting：設定此Scheduler要以怎樣的模式和條件來運作。可分為三種類型：

- Once：此Scheuler只執行一次，即Start Time那次。



- Fixed Patterns：透過下拉選單選擇Frequency以及Period來決定Scheduler執行的頻率或時間。系統亦可設置Stopped Setting，自動停用Scheduler，可選擇的方式有三種：



- Manually：此Scheduler一但設置，只能透過使用者手動停用。
- End Time：一但超過End Time，此Scheduler自動停用。
- Max Run Times：透過此Scheduler觸發的Template執行次數超過指定次數後，該Scheduler自動停用。

◦ Crontab Syntax：以crontab的語法來指定執行的頻率或時間。Stopped Setting的說明同Fixed Patterns。

Frequency Setting (Optional) *

Frequency Type

- ○ Once
- ○ Fixed Patterns
- ◉ Crontab Syntax

Crontab Syntax*

Please Enter Crontab Syntax (e.g. */5 * * * *)

Stopped Setting

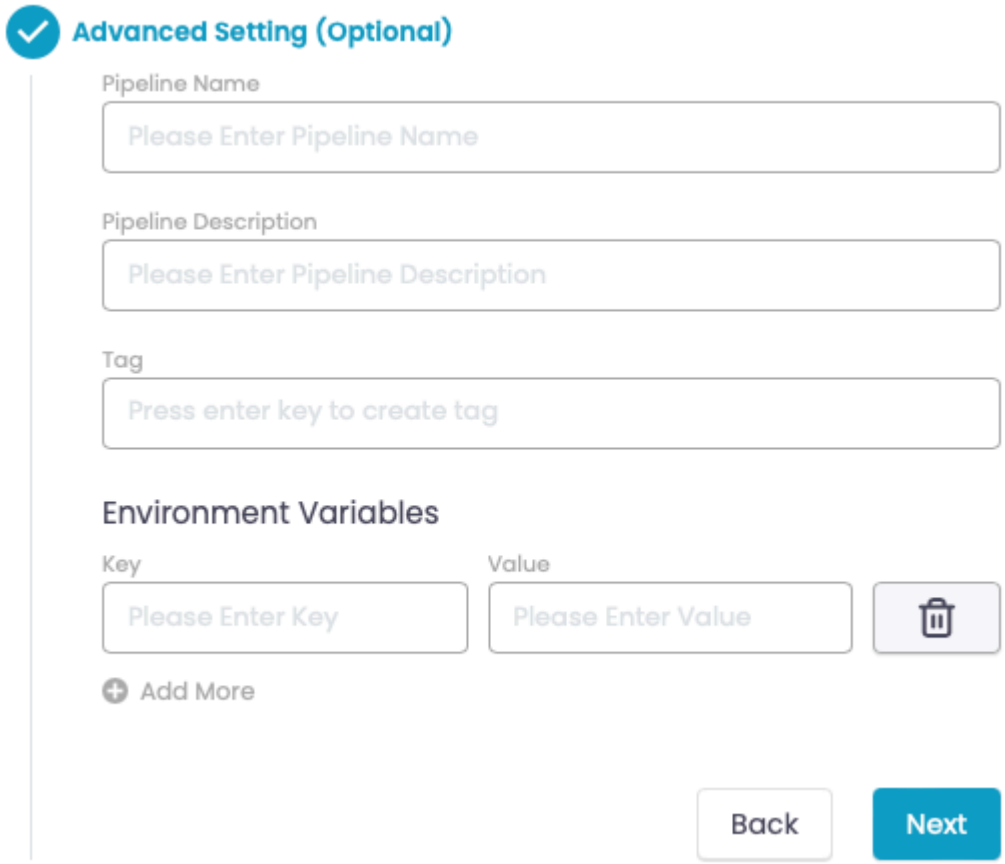
- ○ Manually
- ○ End Time
- ◉ Max Run Times

Max Run Times*

1

Back Next

- Advanced Setting：可在透過此Scheduler觸發的Template給予特別的資訊，包含：



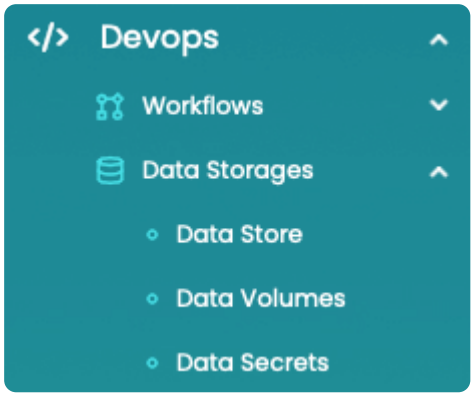
  - Pipeline Name：觸發Template所建立的Pipeline名稱。系統會給予預設的名稱，但若使用者有需求的話，則可以指定。
  - Pipeline Description：觸發Template所建立的Pipeline描述。
  - Tag：觸發Template所建立的Pipeline之標籤。使用者可輸入多個標籤，設定標籤的目的主要是為了在後續的流程中查看特定標籤的執行結果。
  - Environment Variable：若使用者需要額外的環境變數，可點擊add more來增加環境變數。每個環境變數的欄位有三個，若有不需要的環境變數，可點擊變數旁的垃圾桶刪除：
    - Variable Name：此變數的名稱。

- Value/Secret：針對環境變數，可選擇兩種方式輸入。如果要直接輸入值，可在輸入值後按Enter確定，如果要以Data Secret的方式輸入，則選擇對應的Secret。
- Key：若選擇透過Data Secret的方式輸入，則需指定所選Secret中的Key。

當所有資料都填寫完畢後，即可點擊下方的Submit按鈕建立Scheduler。

Data Storages

Data Storage中包含三類型的資源，主要用在輔助Devops Job執行。例如Data Volume可以掛載到Job的執行環境，提供資料存取的機制。Data Secret可將一些機敏資料封裝後設定在Job的參數中。以下將針對各資源的使用進行詳細說明。

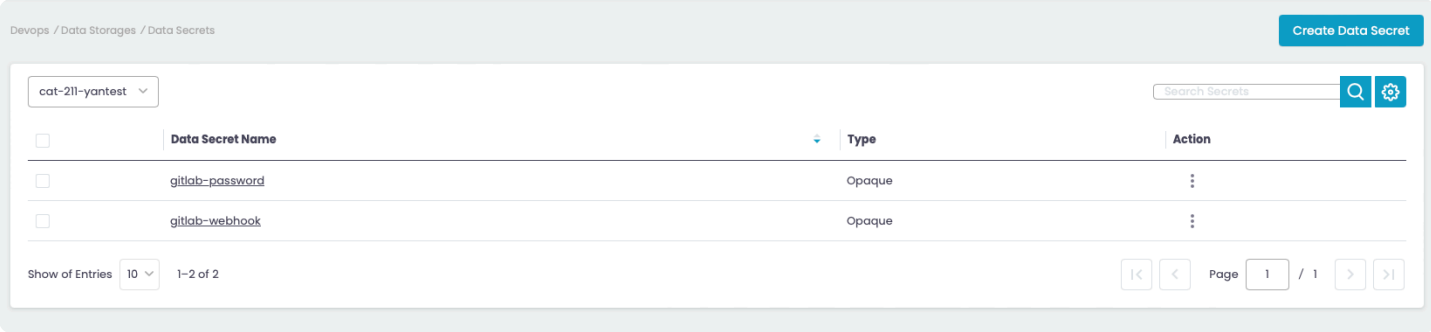


Data Store

Data Volumes

Data Secrets

> Data Secret因存在機敏資料，因此無論Project Admin或Project User皆只能查看、修改、刪除由自己所建立的Data Secret。

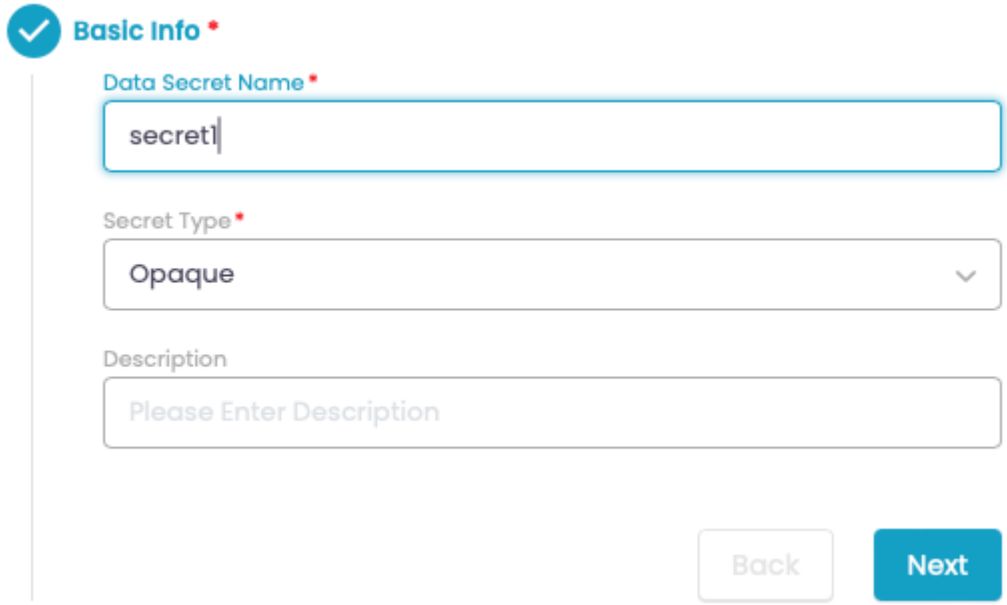


Data Secrets的目的是將一些機敏資料或參數檔封裝成Secret的形式，封裝後的資料可成為Template的參數，亦可以Secret Volume的形式掛載到Job的執行環境後做進一步的應用。於Data Secrets頁面，使用者可以管理使用者所建立的Data Secret。如圖所示，使用者可透過左上角的下拉選單，選擇指定的Kubernetes Cluster，接著畫面就會顯示該Cluster下所建立的Data Secret，每個Data Secret包含數個資訊，說明如下：

- Data Secret Name：此Data Secret的名稱。
- Type：此Data Secret的類型，包含Opaque、TLS、Docker Registry和Basic Auth四種。我們將在稍後的章節說明四種Data Secret的使用時機。
- Action：此Data Secret可執行的操作，包括：
  - Update：即修改此Data Secret的設定。
  - Delete：即刪除此Data Secret。

在Data Secret頁面的右上角有Create Data Secret的按鈕，點擊後即可出現Data Secret的建立表單，使用者可根據自身需求設定必要的參數。Data Secret建立表單主要分為兩個部分，Basic Info、Secret Setting，以下分別就各部分進行說明。

- Basic Info：此Data Secret的基本資訊，包括：

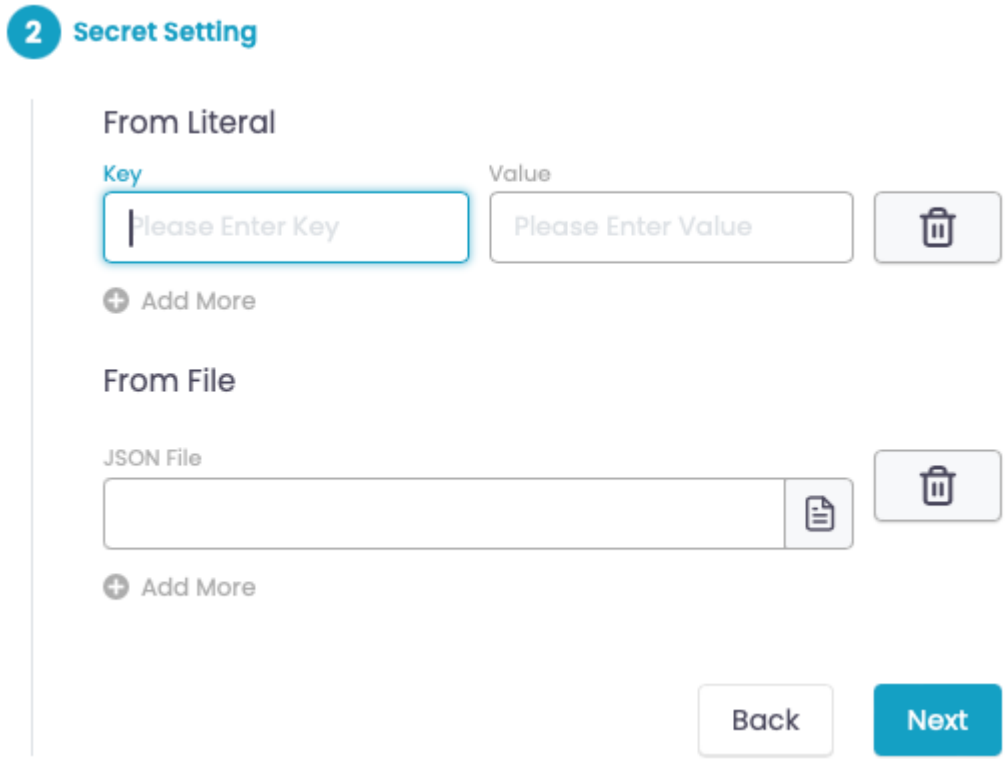


  - Data Secret Name：此Data Secret的名稱。
  - Secret Type：此Data Secret的類型，包含Opaque、TLS、Docker Registry和Basic Auth四種。根據使用者所選擇的類型，Secret Setting部分需要填寫的資訊也會改變，詳情我們在Secret Setting的章節說明。
  - Description：此Data Secret的額外描述。

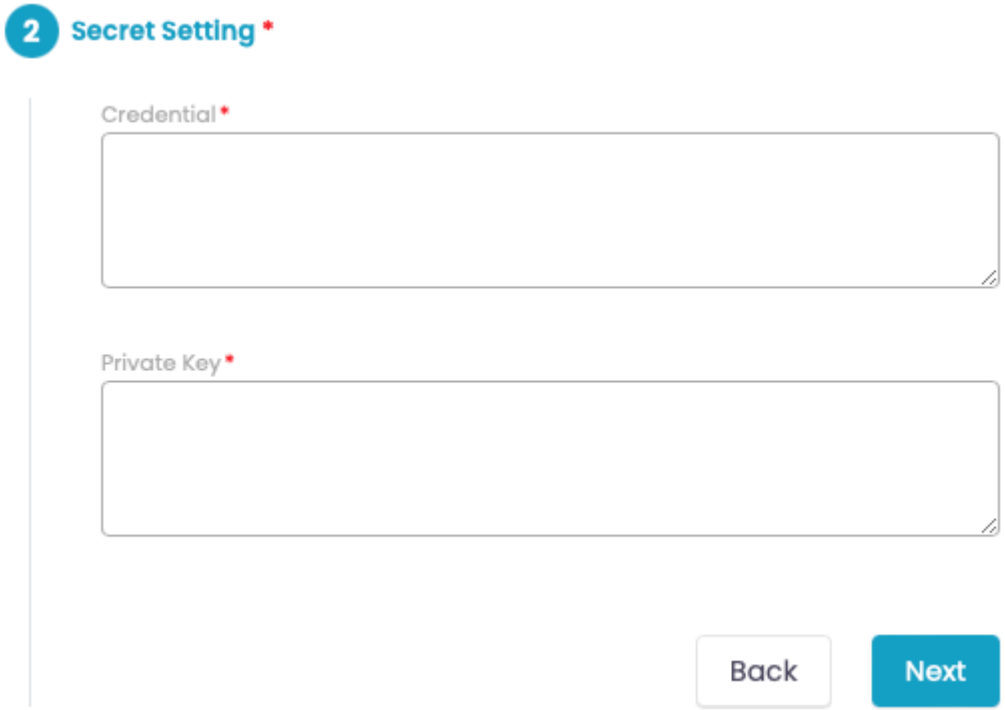
- Secret Setting：

  - Opaque：最通用的類型，以Key-Value的形式儲存使用者的資料，例如應用程式的參數檔。使用者可透過UI介面提供Key-Value的資訊，若有多筆資訊，可點擊Add More來新

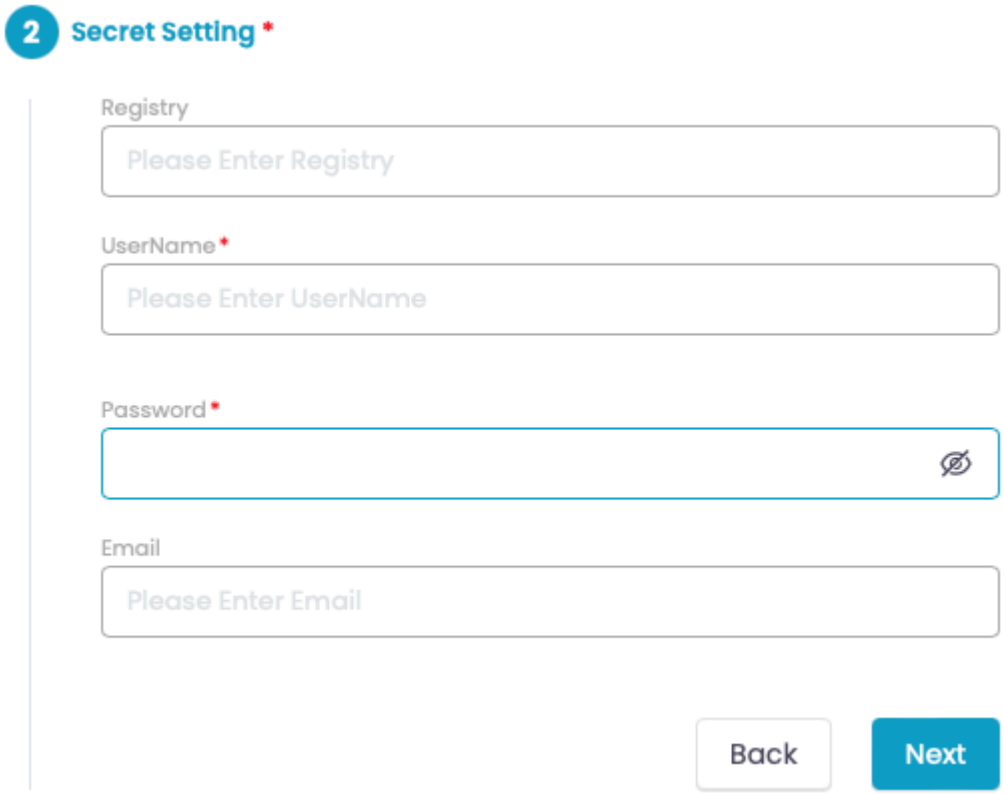
增。此外，使用者亦可直接上傳Json格式的檔案來提供Data Secret的內容。



- TLS：主要用來儲存TLS的資訊，所需填寫的資訊包括：

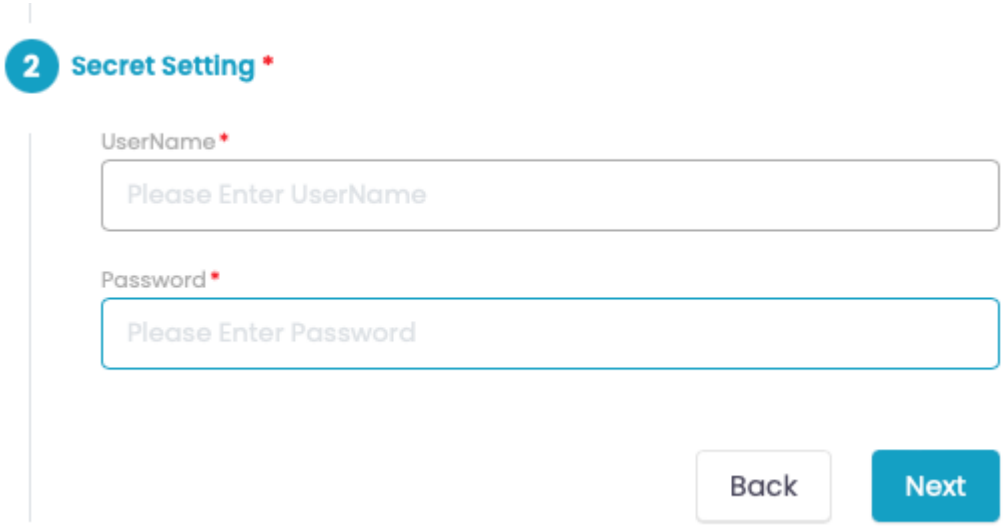


  - Credential：憑證的Credential。
  - Private Key：憑證的Private Key。
- Docker Registry：主要用來儲存Docker Registry的認證資訊，以供使用者在從Docker Registry Pull Image時使用。所需填寫的資訊包括：



- Registry：欲Pull Image之Docker Registry URL。
- UserName：使用者於該Docker Registry上的帳號。
- Password：使用者於該Docker Registry上的密碼。
- Email：使用者於該Docker Registry上的Email。

- Basic Auth：最基本的帳號密碼認證機制。所需填寫的資料包括：



  - UserName：使用者的帳號
  - Password：使用者的密碼。

當所有資料都填寫完畢後，即可點擊下方的Submit按鈕建立Scheduler。

使用者下拉選單

在Nomos介面的右上角，會顯示使用者的身份，點擊使用者下拉選單

Grafana Dashboard