**FIDO（Fast IDentity Online）叢集安裝部署手冊**

2024年9月

**前言**

FIDO叢集安裝部署手冊提供基於Kubernetes環境的多個FIDO容器實例運作說明，並可提供實體機/虛擬機節點的高可用性、負載平衡及資料庫同步，以支援來自客戶端的高峰流量及維運時需要的容錯。

**一、FIDO叢集架構及節點服務功能**

圖為FIDO叢集架構範例，我們將依此範例作為手冊安裝設置的對象，接下來說明每個實體機/虛擬機(host)節點上提供的服務功能，在下個章節會逐步說明該服務的安裝設置步驟。

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

自動產生的描述

host1為前端客戶的應用服務例如web，host2為硬體支援的高可用性及負載平衡設備，gitlab/harbor為安裝在外部網路的FIDO 容器映像檔案儲存庫。手冊將說明如何在外部/內部網路環境架設host3~host9節點的服務以支援運作多個FIDO容器實例並提供節點的高可用性、負載平衡及資料庫同步

**k8s**：k8s host3與host4節點提供多個FIDO容器實例運作管理功能，主要有容器自動化部署和擴展。兩個k8s節點搭配host2的硬體設備可同時支援FIDO容器的高可用性及負載平衡，除了可以分擔來自客戶端的流量外，當其中一個k8s節點不能運作時，另一個k8s節點可以獨立維持FIDO容器管理功能的正常運作。

[註] 硬體支援的高可用性及負載平衡設備也可以使用軟體或是容器方式來實現。

**keepalived**：host5與host6節點以keepalived容器方式提供後端節點(host7~host9)的高可用性，例如使用者可定義keepalived1的優先權為100、keepalived2的優先權為90及Virtual IP address(VIP)，VIP會是一個host3~host9共同網段中未使用的IP address，當前端客戶的資料要透過FIDO容器讀/寫到後端資料庫時都會對應到這個VIP。啟動兩個keepalived容器服務時，keepalived1有較高優先權會是Master節點除了自有的IP address外也同時拿到VIP，keepalived1 host5節點會將由FIDO容器來的資料庫讀/寫請求交給haproxy1容器服務再傳送到後端資料庫。keepalived2為較低優先權會是Backup節點只會有自有的IP address，只有當keepalived1容器服務或是 host5節點不能正常運作時，keepalived2 host6節點會接手為Master並取得VIP以維持運作處理由FIDO容器來的資料庫讀/寫請求。

**haproxy**： host5與host6節點以haproxy容器方式提供後端資料庫的負載平衡。當keepalived1 host5節點運作時，haproxy1容器可以設置聽取來自FIDO容器對3306 port的mysql資料庫讀/寫請求，再將此請求輪流分配到host7~host9上的對應的mysql資料庫容器服務以達到負載平衡的目的。

[註] 對host7~host9節點上的任一mysql資料庫容器寫入資料後，三台的資料庫都會同步一致。

**mysql**：在host7節點的mysql master node1容器負責mysql叢集間的溝通協調，host8節點的mysql slave node2容器及host9節點的mysql slave node3容器皆為mysql叢集的成員。每個mysql容器都可以接受來自FIDO容器的資料庫讀/寫請求，資料有異動後會再同步到另兩個節點的mysql資料庫。

**二、FIDO叢集節點的安裝設置**

**系統環境準備**

* 每個實體機/虛擬機節點安裝Ubuntu 24.04.1 LTS桌面版本/伺服器版本
* 關閉DHCP服務，在每個Ubuntu 24.04節點上手動綁定IP位址
* 停用節點之間的防火牆，執行命令sudo ufw status應該顯示Status: inactive
* 確認內部網段/外部網段/管理網段之間的可連通性

容器服務的安裝設置，建議依序由底層的mysql叢集容器、haproxy容器、keepalived容器、k8s管理及FIDO容器逐步安裝及驗證容器功能是否正常運作。

**預安裝工具套件**

$ sudo apt install openssh-server net-tools -y

$ sudo apt install vim dos2unix -y

**預安裝Docker 套件**

$ sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

$ sudo docker --version **# Docker version 27.1.2, build d01f264**

$ sudo usermod -aG docker ${USER} **# 例如 /home/k3d，${USER}=k3d**

$ sudo reboot

1. **安裝k8s管理**

在host3與host4 分別建立k8s 環境，host3 建立k8s node1，host4建立k8s node2。

**下載k3d安裝檔**

wget -q -O - https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh | bash

**準備 k3d 設定檔 config.yaml，設定docker image倉庫repo連線**

# k3d configuration file, saved as e.g. /home/me/myk3dcluster.yaml

apiVersion: k3d.io/v1alpha5 # this will change in the future as we make everything more stable

kind: Simple # internally, we also have a Cluster config, which is not yet available externally

metadata:

name: prod1 # name that you want to give to your cluster (will still be prefixed with `k3d-`)

servers: 1 # same as `--servers 1`

agents: 1 # same as `--agents 2`

kubeAPI: # same as `--api-port myhost.my.domain:6445` (where the name would resolve to 127.0.0.1)

hostIP: "0.0.0.0" # where the Kubernetes API will be listening on

hostPort: "6445"

ports:

- port: 80:80 # same as `--port '8080:80@loadbalancer'`

nodeFilters:

- loadbalancer

- port: 443:443

nodeFilters:

- loadbalancer

registries: # define how registries should be created or used

config: | # define contents of the `registries.yaml` file (or reference a file); same as `--registry-config /path/to/config.yaml`

mirrors:

"xxx.xxx.xxx.xxx:3000": #fido docker image倉庫

endpoint:

- "http://xxx.xxx.xxx.xxx:3000" #fido docker image倉庫

configs:

"xxx.xxx.xxx.xxx:3000": #fido docker image倉庫

auth:

username: xxxx #fido docker image倉庫帳戶

password: xxxxxxxx #fido docker image倉庫密碼

**建立k3d cluster**

sudo k3d cluster create –config config.yaml

**安裝kubectl指令，後續使用kubectl管理k3d**

**下載kubectl**

curl -LO [https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

**下載 sha256編碼檔**

curl -LO [https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)

**比較kubectl主檔與編碼檔**

echo "$(cat kubectl.sha256) kubectl" | sha256sum –check

**安裝kubectl主檔並給予root權限**

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**安裝argocd，後續使用argocd進行k8s中pod的監控。(argocd也是k8s中的一個pod)**

**建立argocd在k8s中的命名空間**

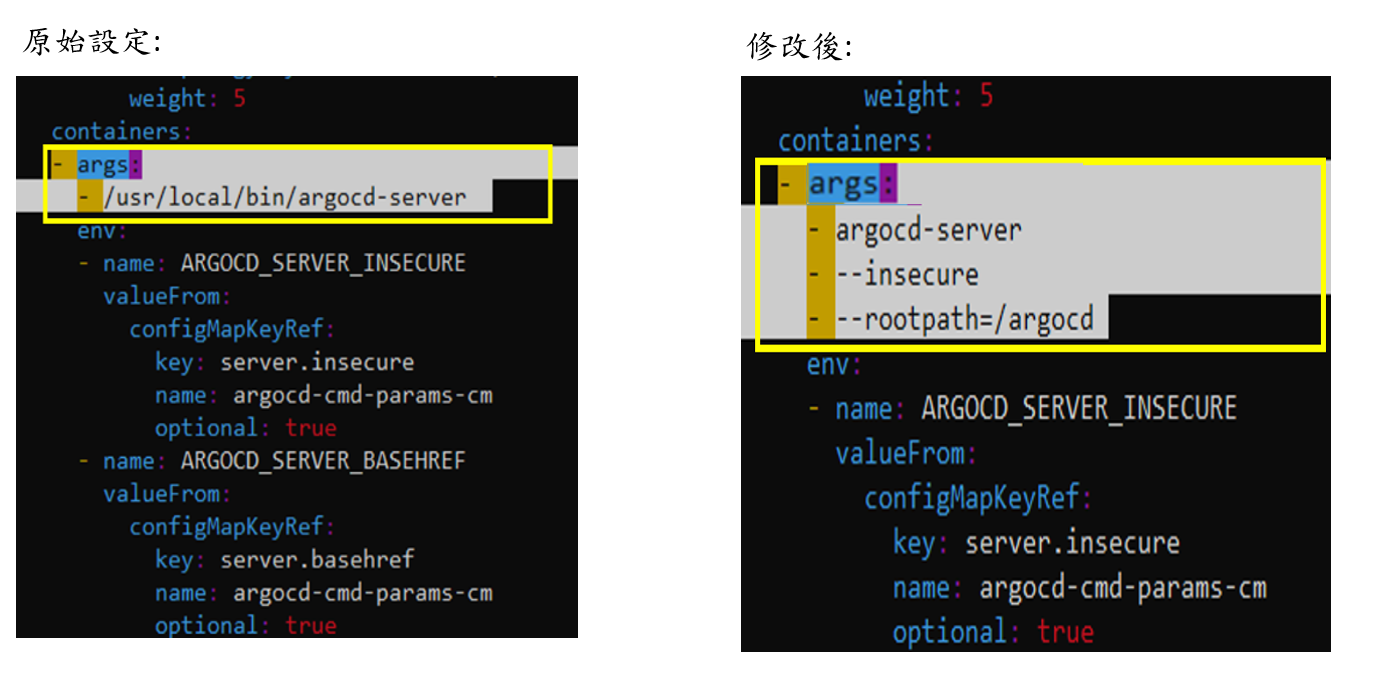
sudo kubectl create namespace argocd

**安裝argocd**

sudo kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

**修改 argocd佈署文件**

sudo kubectl edit deployment argocd-server -n argocd



**準備argocd路由檔案 ingress.yaml**

apiVersion: traefik.containo.us/v1alpha1

kind: IngressRoute

metadata:

name: argocd-server

namespace: argocd

spec:

entryPoints:

- web #websecure

routes:

- kind: Rule

match: Host(`localhost`) && PathPrefix(`/argocd`)

priority: 10

services:

- name: argocd-server

port: 80

- kind: Rule

match: Host(`localhost`) && PathPrefix(`/argocd`) && Headers(`Content-Type`, `application/grpc`)

priority: 11

services:

- name: argocd-server

port: 80

scheme: h2c

**安裝argocd路由檔案**

sudo kubectl apply -n argocd -f ingress.yaml

**得到argocd 初始admin的密碼**

sudo kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

**登入argocd，進行fido容器的部署**

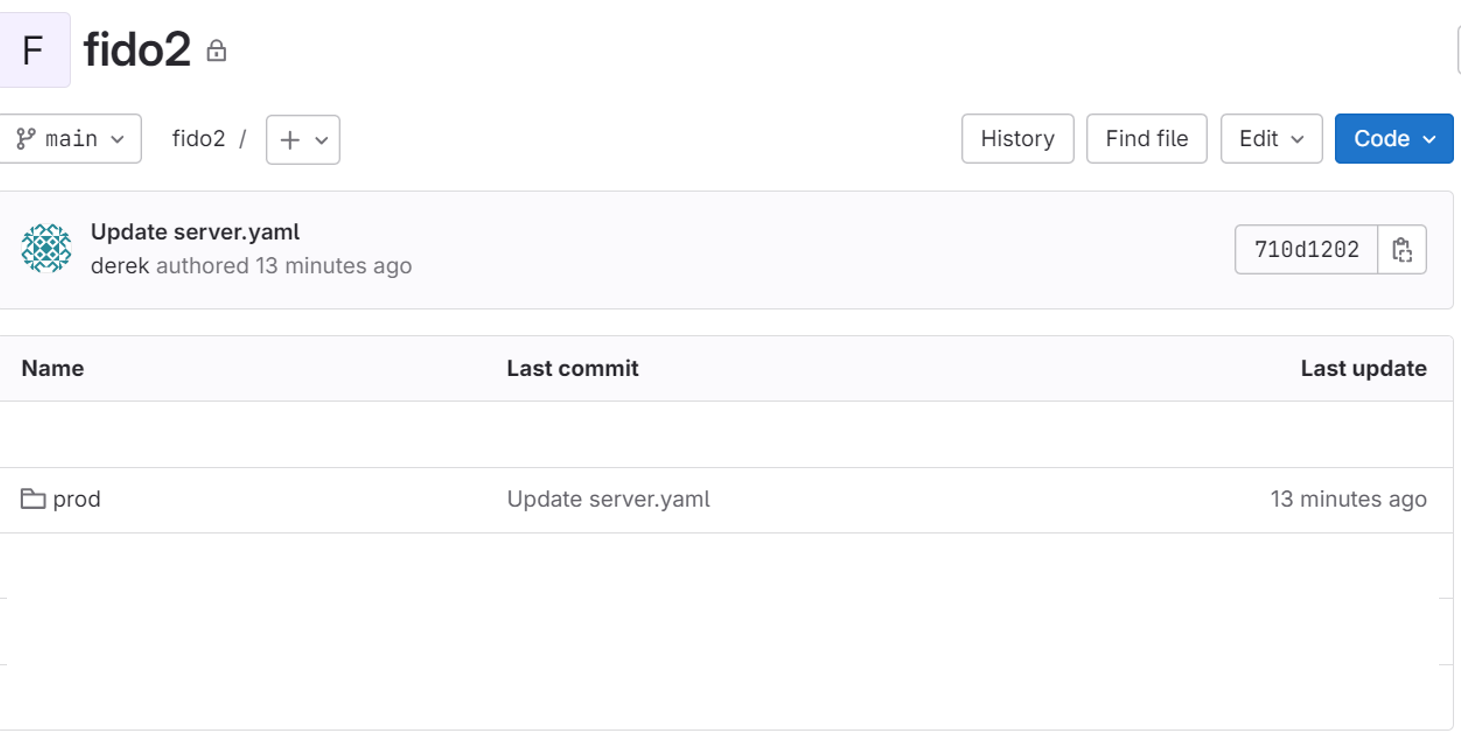
argocd預設帳戶為 admin 密碼透過上一步獲得

過瀏覽器使用host3 與host4各自進入argocd，進行pod的部署

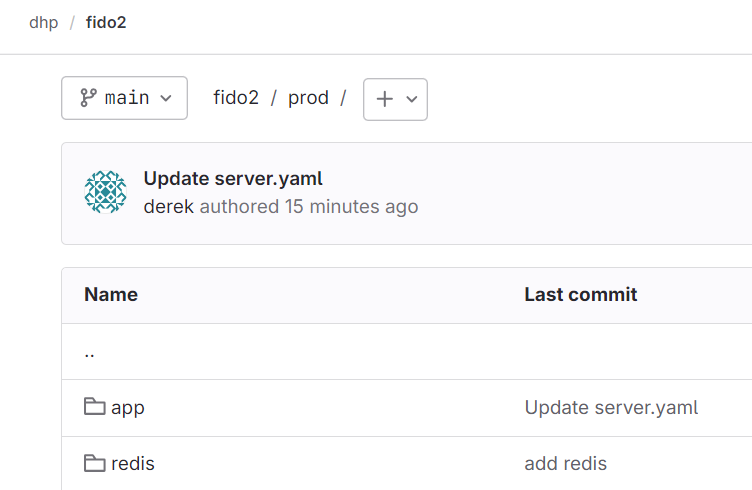
1. **設置FIDO 儲存庫連線**

**請先準備k8s部屬專案於gitlab上，該專案主要是 fido server與 redis在k8s上的部屬設定檔**

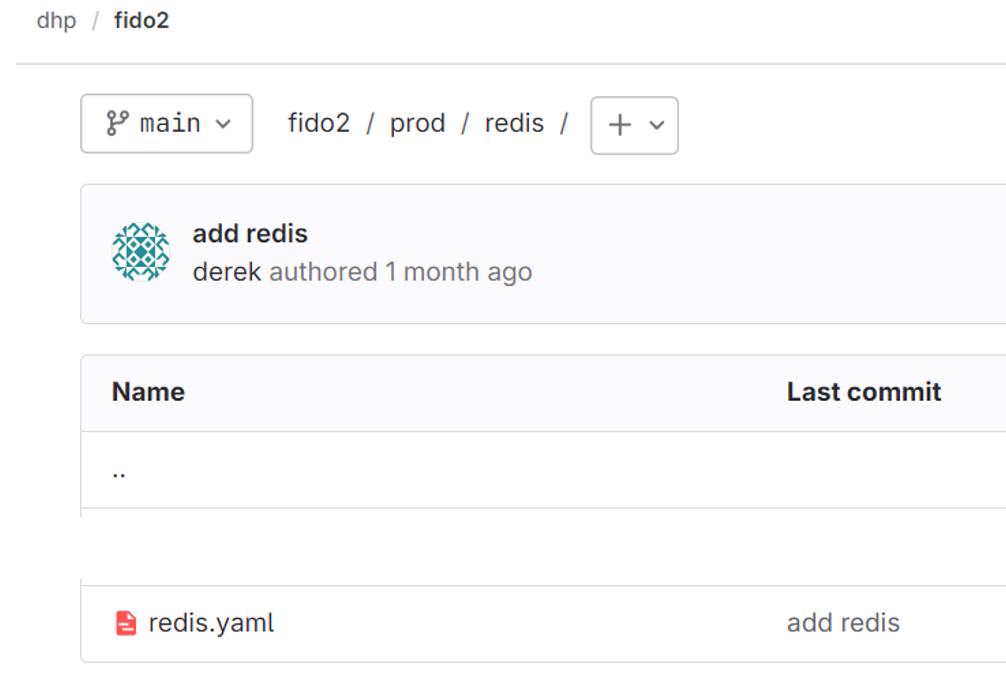
**假設專案 為** [**http://xxxxxxx/dhp/fido2.git**](http://xxxxxxx/dhp/fido2.git)

****

**部屬專案結構**

****

**Redis部分**

****

**Redis.yaml**

apiVersion: v1

kind: Namespace

metadata:

name: fido2

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: fido2-redis

namespace: fido2

spec:

serviceName: redis

replicas: 1

selector:

matchLabels:

app: redis

template:

metadata:

labels:

app: redis

spec:

containers:

- name: redis

image: redis:latest

ports:

- containerPort: 6379

volumeMounts:

- name: redis-data

mountPath: /data

volumeClaimTemplates:

- metadata:

name: redis-data

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 1Gi

---

apiVersion: v1

kind: Service

metadata:

name: redis

namespace: fido2

spec:

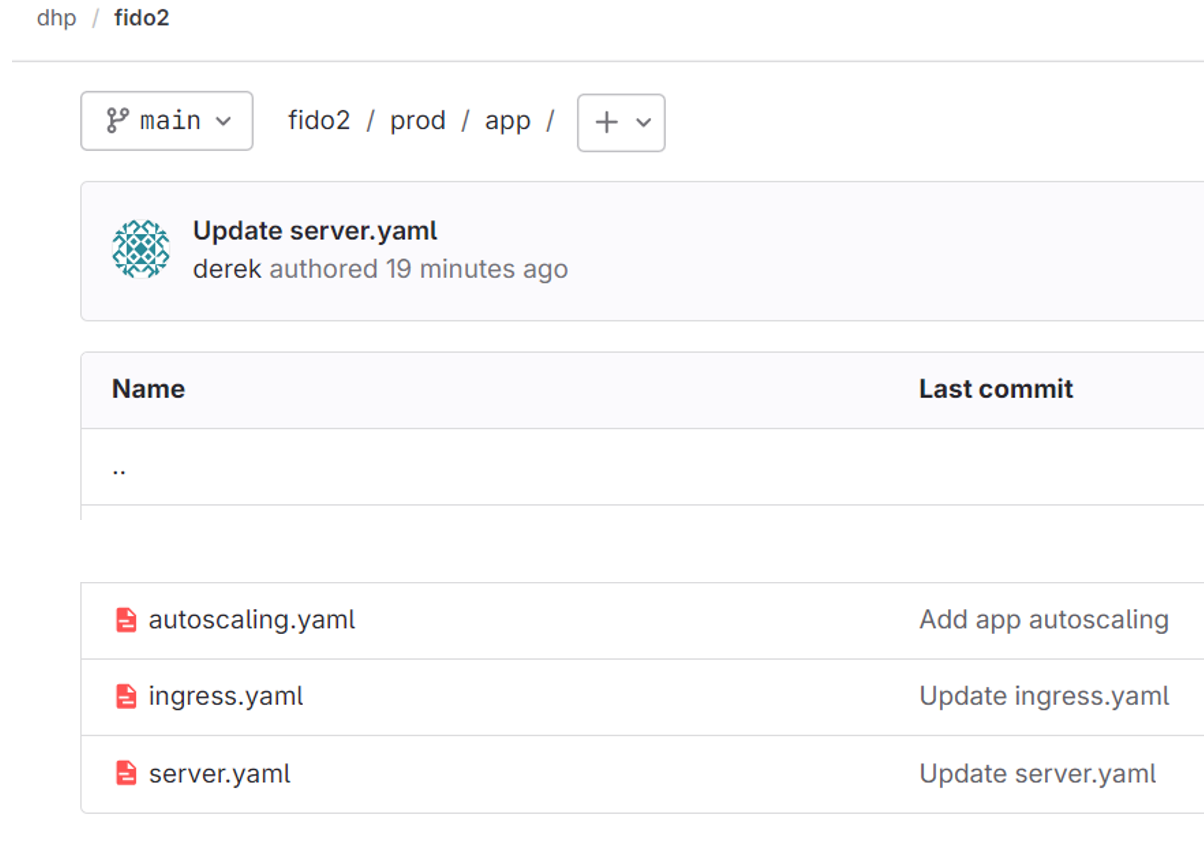
ports:

- port: 6379

selector:

app: redis

**fido部分**

****

**Autoscaling.yaml**

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: simple-go-web-autoscaler

namespace: fido2

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: simple-go-web

minReplicas: 3

maxReplicas: 6

targetCPUUtilizationPercentage: 70

**ingress.yaml**

apiVersion: traefik.containo.us/v1alpha1

kind: IngressRoute

metadata:

name: simple-go-web-route

namespace: fido2

spec:

entryPoints:

- web #web

routes:

- match: Host(`fido2 server 網域`)

priority: 3

kind: Rule

services:

- name: simple-go-web

port: 8080

- match: Host(`fido2 server 網域) && PathPrefix(`/api`)

priority: 4

kind: Rule

services:

- name: simple-go-web

port: 8080

**server.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: simple-go-web

namespace: fido2

labels:

app: simple-go-web

spec:

replicas: 3

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

minReadySeconds: 5

selector:

matchLabels:

app: simple-go-web

template:

metadata:

labels:

app: simple-go-web

spec:

containers:

- name: simple-go-web

image: docker image倉庫位置:3000/fido2/fido2-server:1.933.2

ports:

- containerPort: 8080

env:

- name: GIN\_MODE

value: release

---

apiVersion: v1

kind: Service

metadata:

name: simple-go-web

namespace: fido2

spec:

selector:

app: simple-go-web

ports:

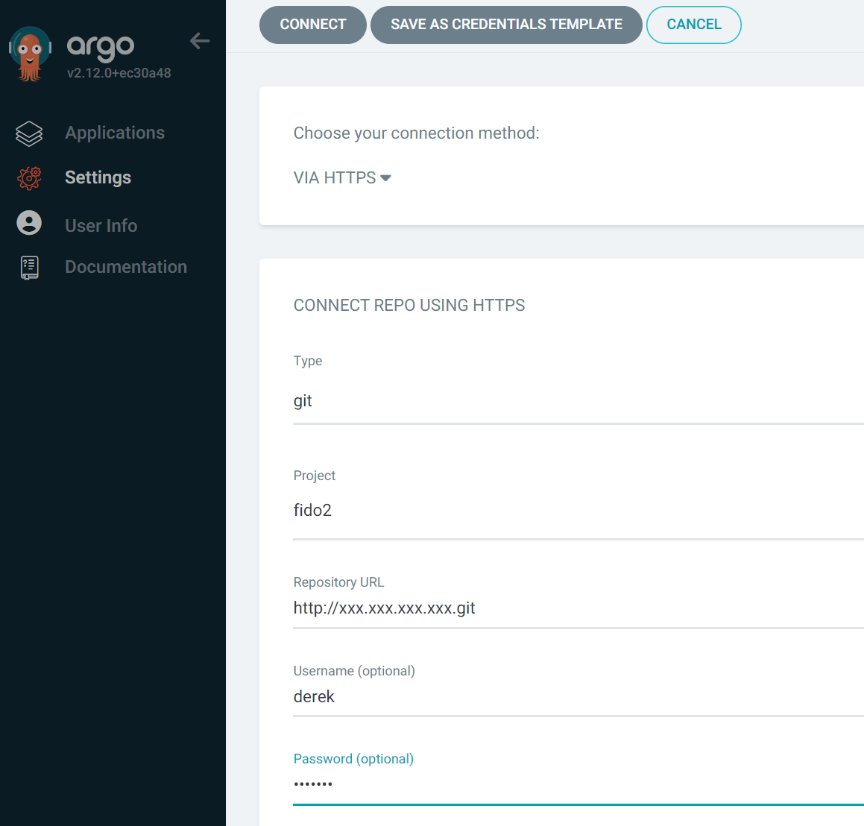
- protocol: TCP

port: 8080

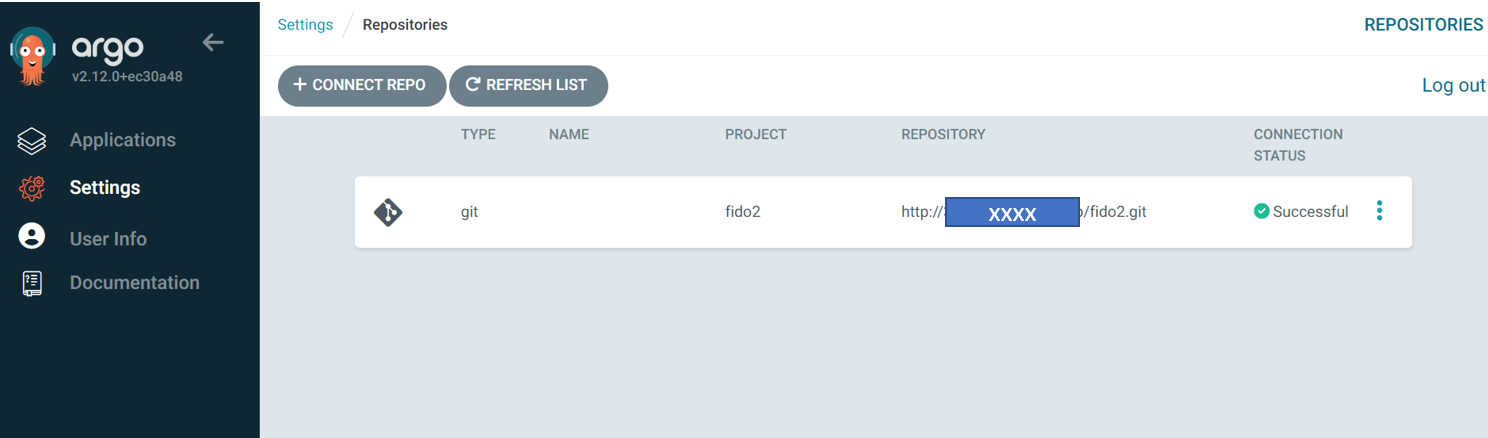
targetPort: 8080

**進行部屬檔與argocd的連線**

**設定gitlab repo連線**



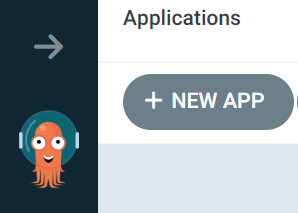
設定連線完成

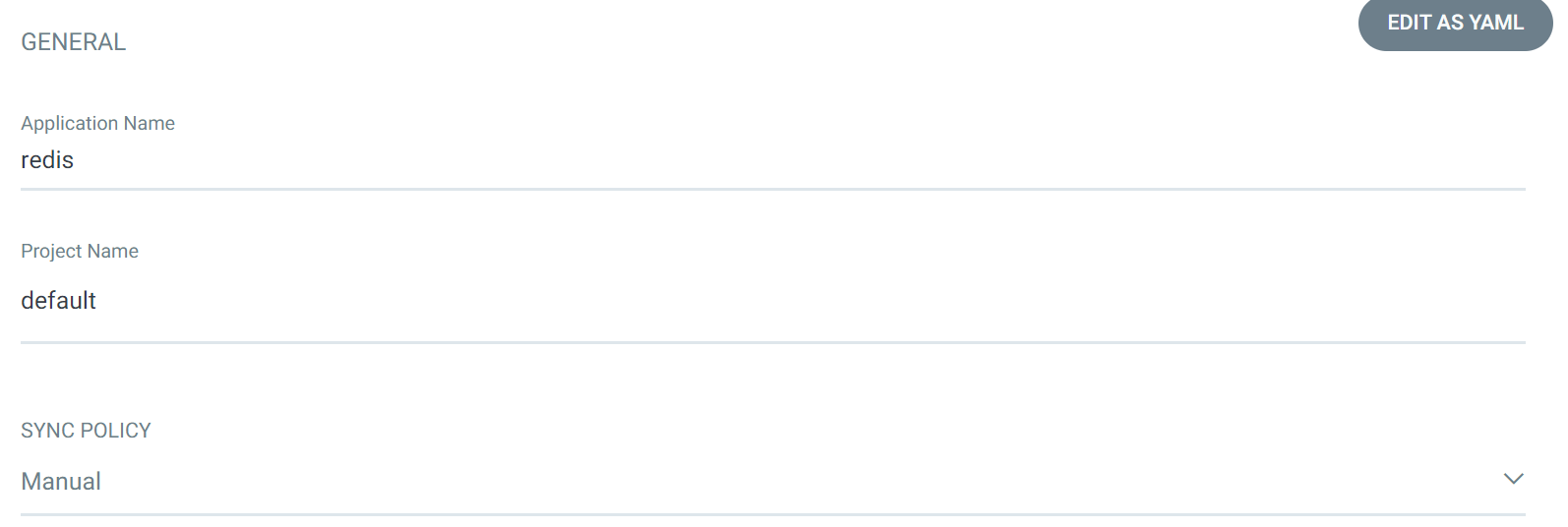


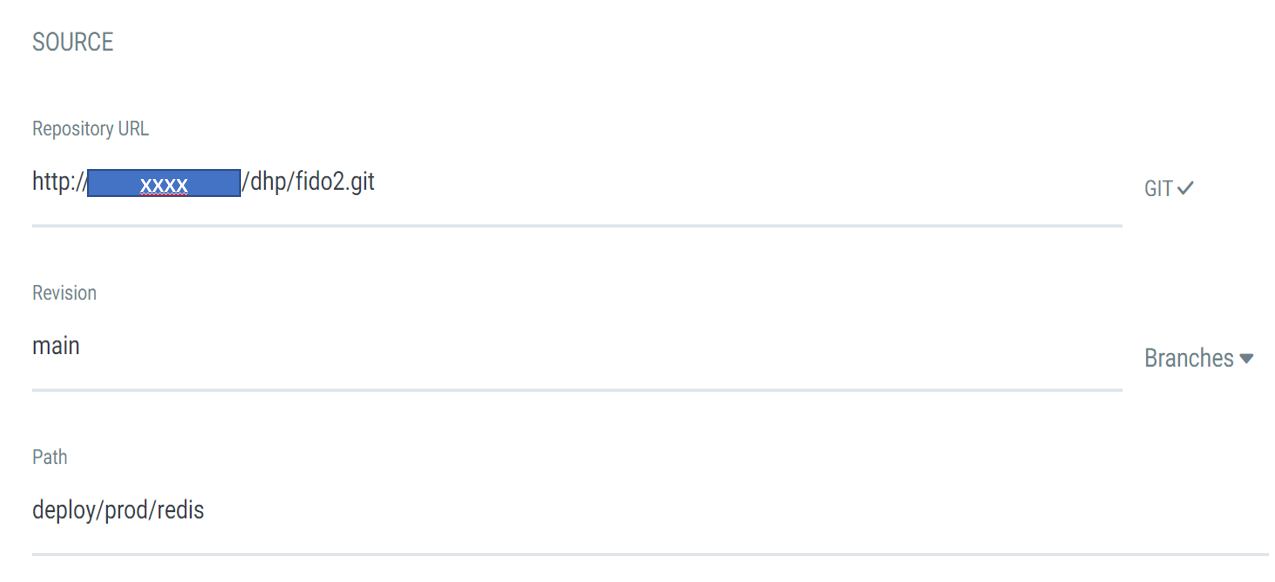
1. **部署FIDO容器**

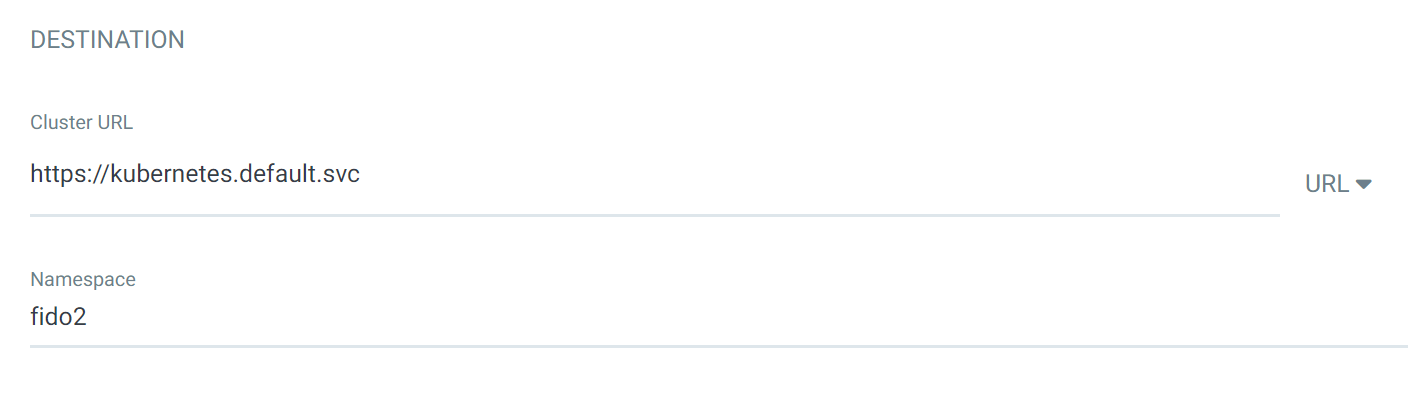
進入host3 (k8s node1) 與 host4 (k8s node2) 的argocd控制台

**先部屬Redis的部分， 按下 + NEW APP按鈕**



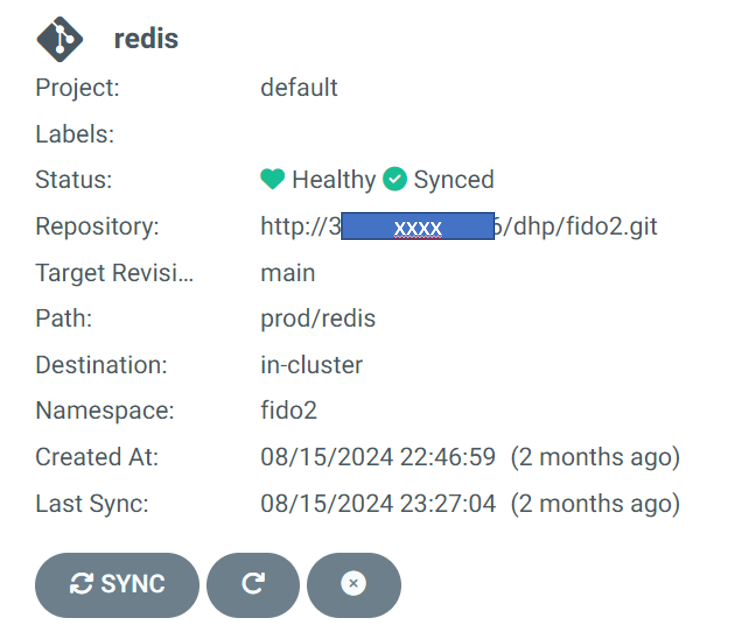




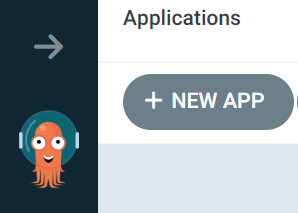


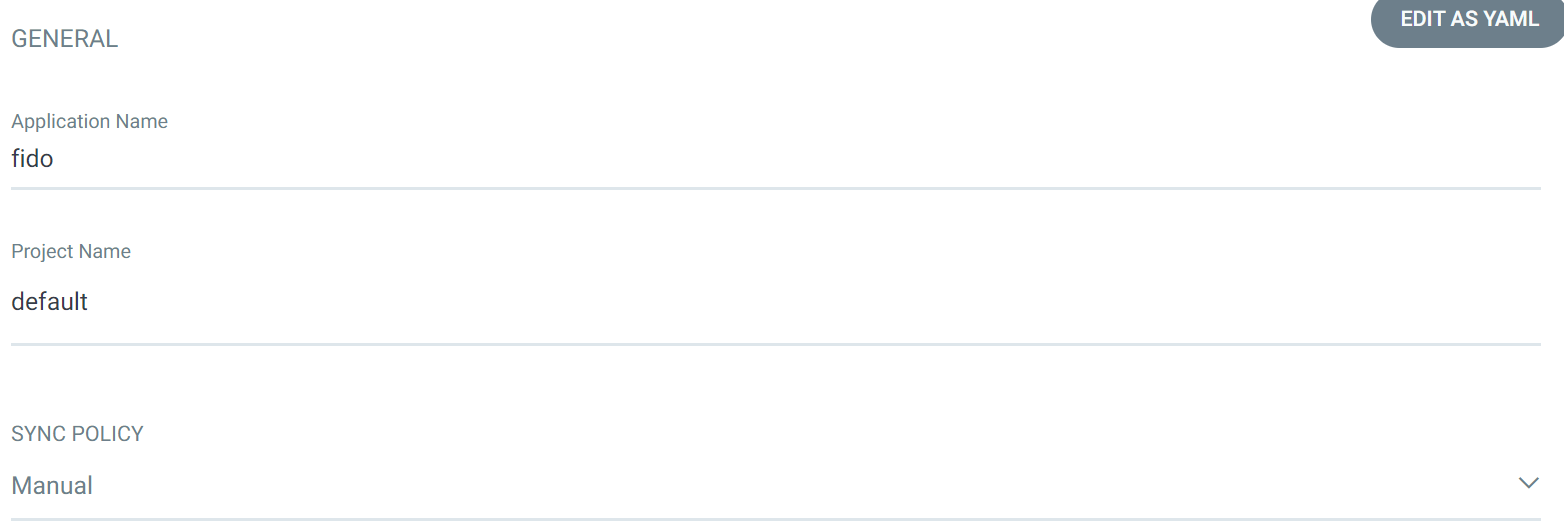
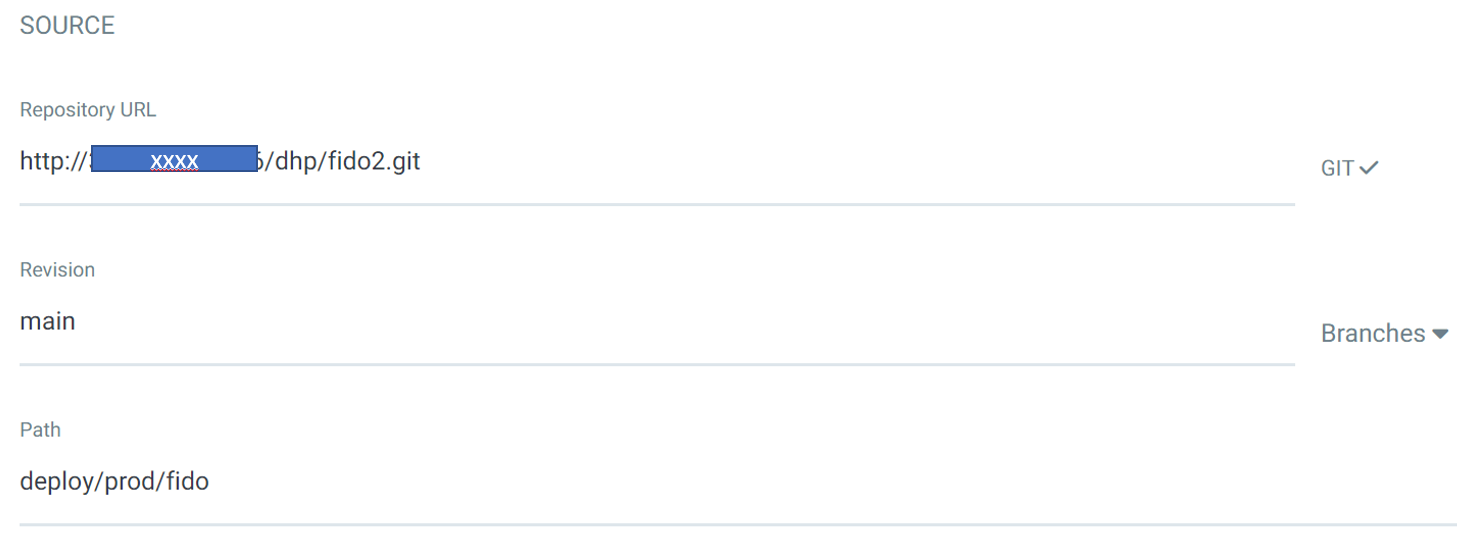
**設定redis pod與github上deploy commit 同步，完成部屬**

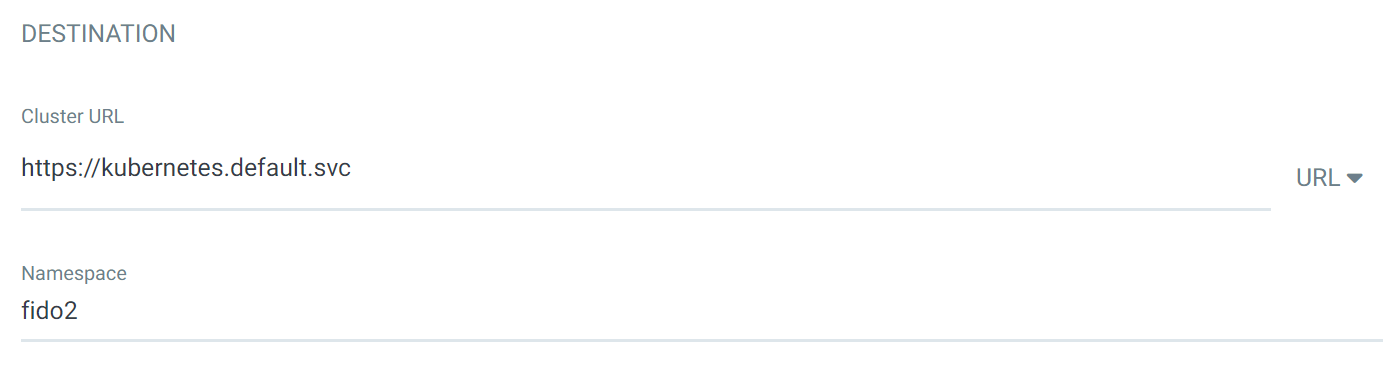
按下 sync按鈕，進行同步



**接著部署fido， 按下 + NEW APP按鈕**

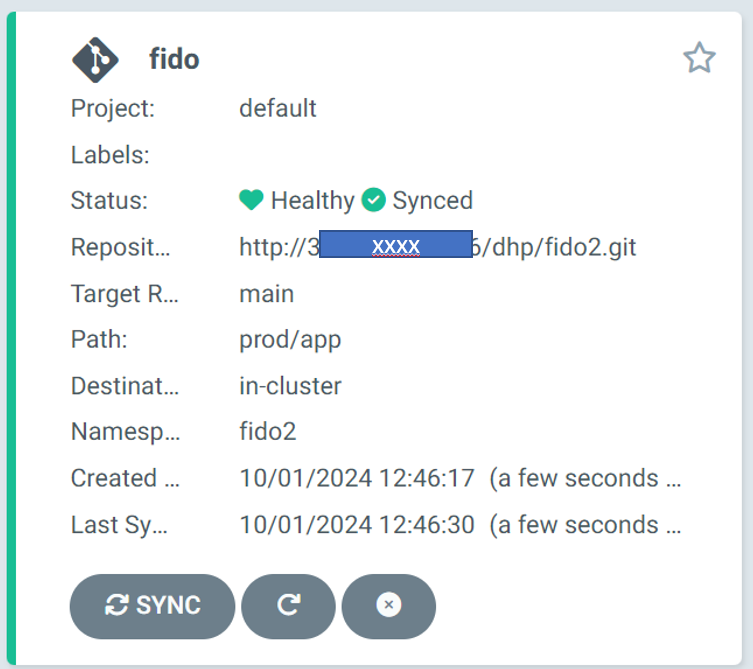


設定欄位  
  


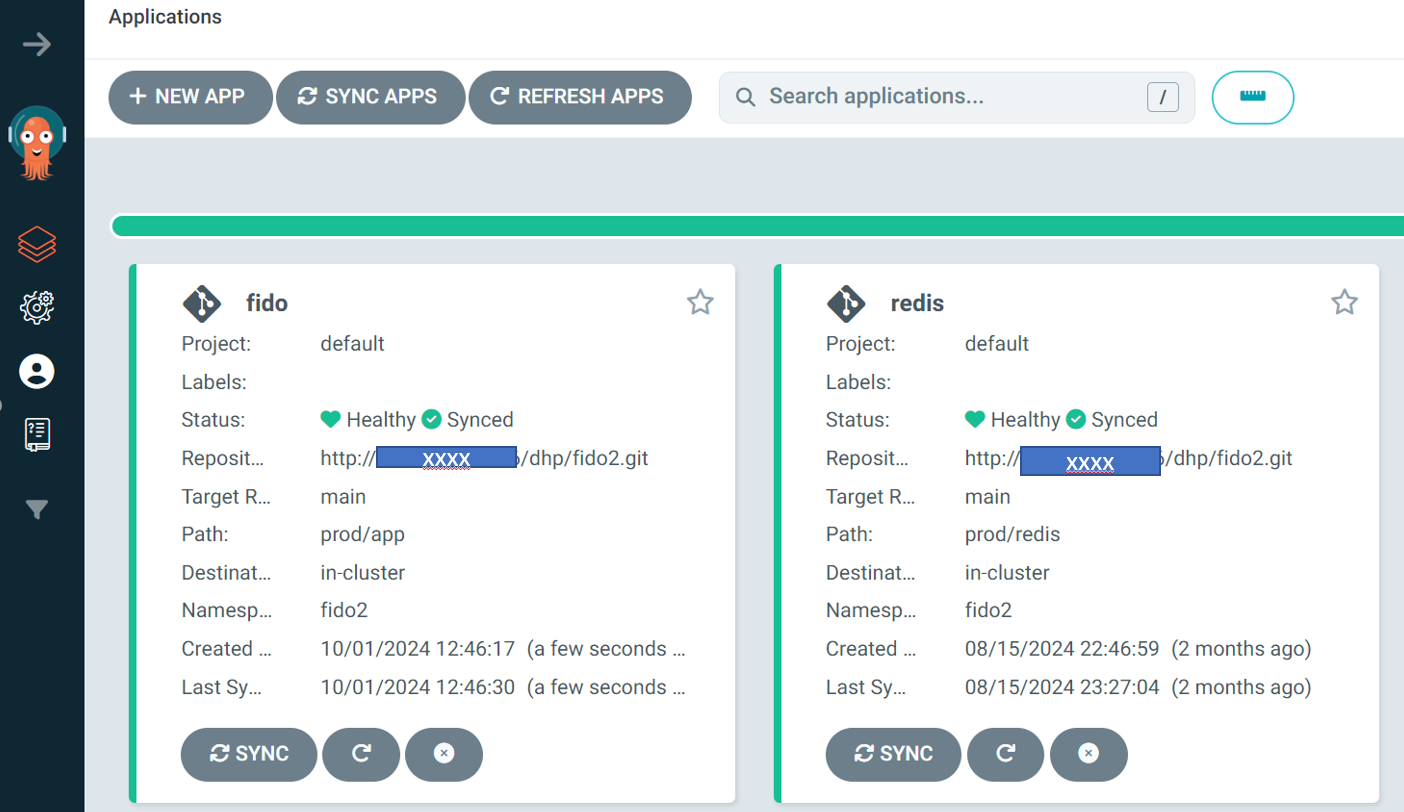


**設定fido app與github上deploy commit 同步，完成部屬**

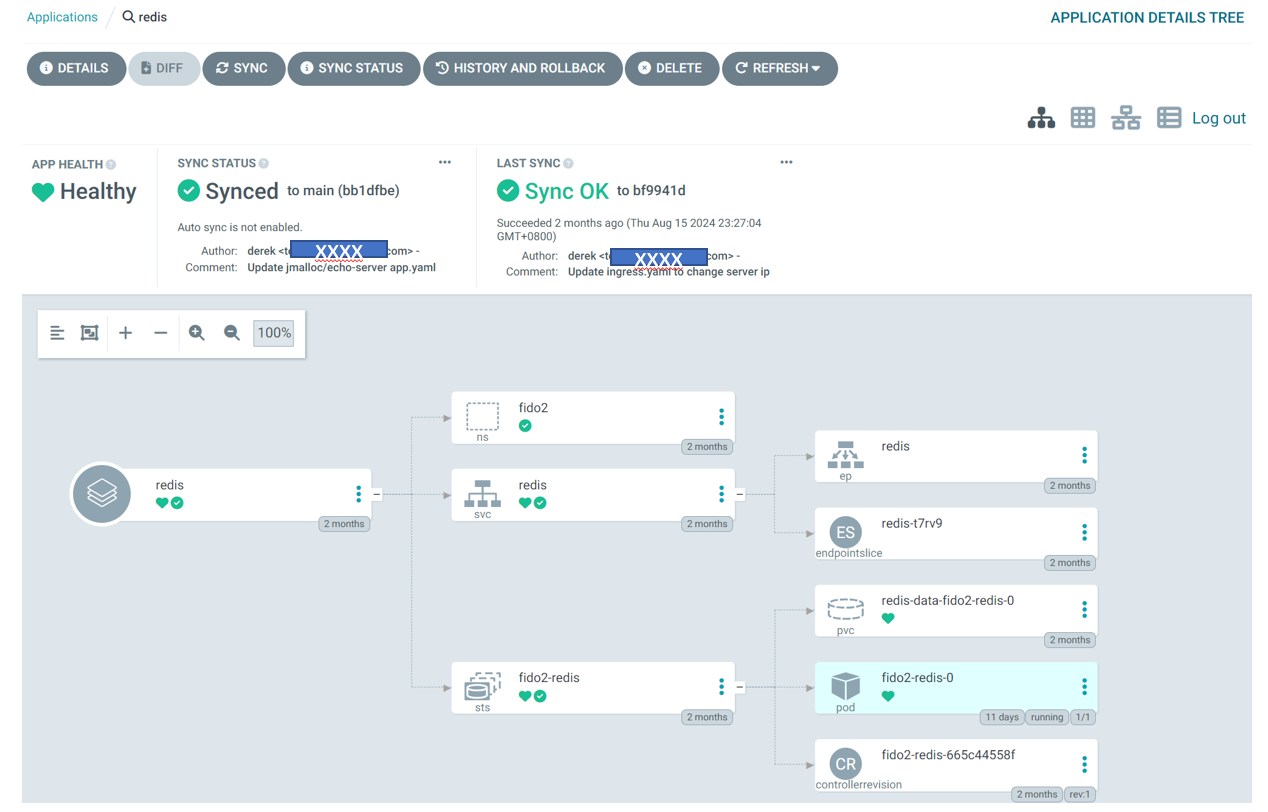
按下 sync按鈕，進行同步



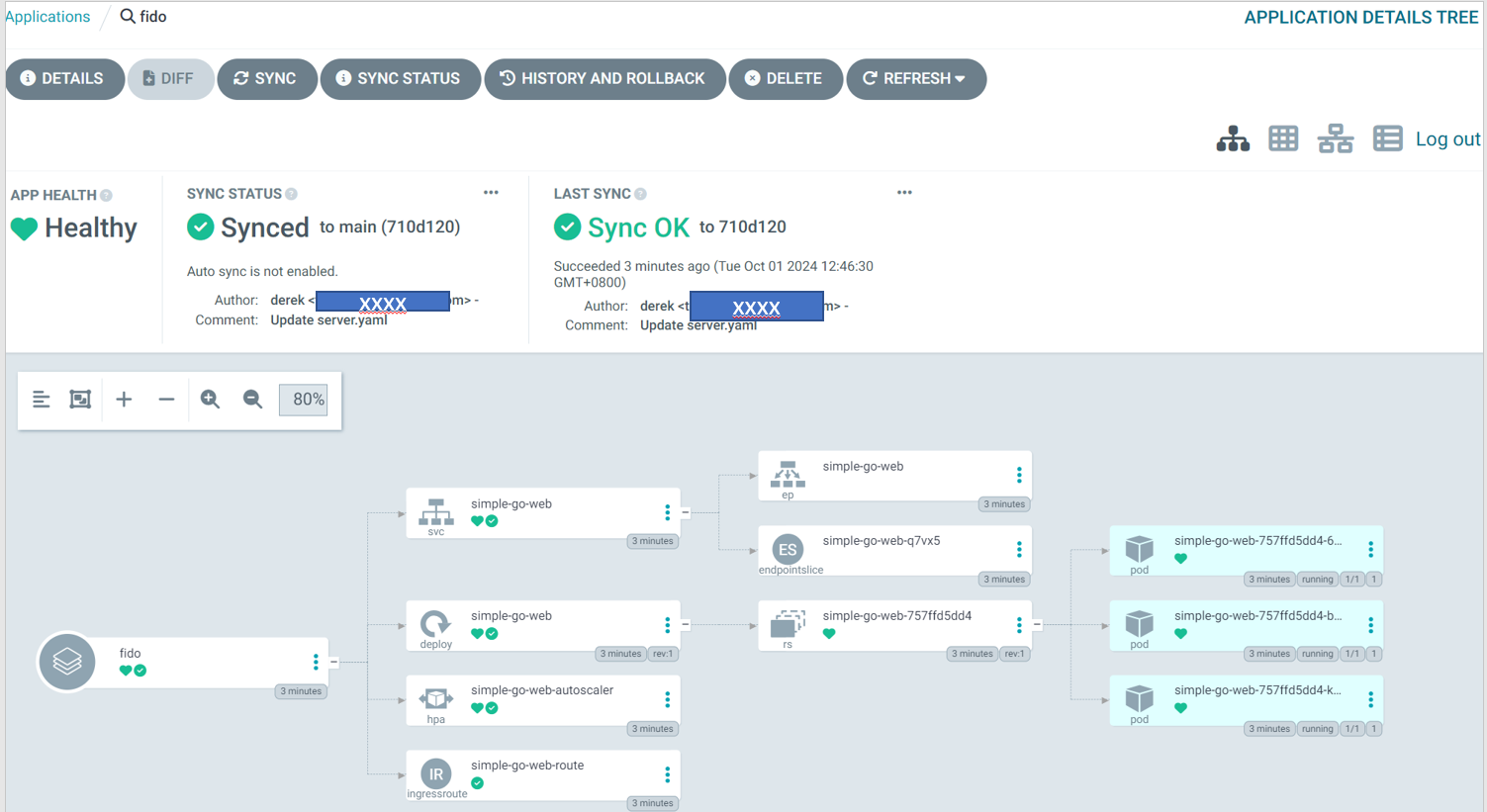
**完成redis pod與 fido app pod的部屬後，可以在argocd控制台看到運作情況**



**Redis pod的部分**



**Fido pod的部分**



1. **安裝keepalived容器**

將keepalived安裝資料夾複製到host5及host6節點，在host5節點中將檔案keepalived\_master.conf裡的interface及VIP的預設值更改成實際環境的對應值；host6節點中將檔案keepalived\_backup.conf裡的interface及VIP的預設值更改成實際環境的對應值。

…

interface **enp0s8**

…

virtual\_ipaddress {

**192.168.50.11** # Same VIP as server1

}

…

在host5節點中將檔案run\_keepalived1.sh裡的host volume路徑及interface

的預設值更改成實際環境的對應值；host6節點中將檔案run\_keepalived2.sh

裡的interface及VIP的預設值更改成實際環境的對應值。

…

docker run \

-d \

--name keepalived**1** \

--network host \

--restart unless-stopped \

--cap-add=NET\_ADMIN \

--cap-add=NET\_BROADCAST \

--cap-add=NET\_RAW \

-v **/home/k3d/keepalived/**keepalived.conf:/usr/local/etc/keepalived/keepalived.conf \

-v **/home/k3d/keepalived/**check\_haproxy.sh:/usr/local/etc/keepalived/check\_haproxy.sh \

osixia/keepalived:latest

…

ip addr show **enp0s8**

…

ip addr show **enp0s8**

改完對應值後，host5節點安裝資料夾中執行下面指令以啟動keepalived1容器服務；host6節點的安裝資料夾中用相同方式改成啟動keepalived2容器服務(在host6將標示1改成2)。

$ sudo chmod +x ./run\_keepalived**1**.sh

$ ./run\_keepalived**1**.sh

1. **安裝haproxy容器**

將haproxy安裝資料夾複製到host5及host6節點，在host5節點中將檔案

haproxy1.yaml裡的host volumes路徑的預設值更改為實際環境的對應值；host6節點中將檔案haproxy2.yaml裡的host volumes路徑的預設值更改成實際環境的對應值。

…

volumes:

- **/home/k3d/haproxy/**haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro

…

在host5及host6節點的haproxy安裝資料夾中將haproxy.cfg裡三組資料庫節點的IP address預設值改成實際環境的對應值。

…

backend mysql\_servers

mode tcp

balance roundrobin

server pxc\_mysql1 **192.168.50.8**:3306 check

server pxc\_mysql2 **192.168.50.9**:3306 check

server pxc\_mysql3 **192.168.50.10**:3306 check

改完對應值後host5節點安裝資料夾中執行下面指令以啟動haproxy1容器服務；host6節點安裝資料夾中用相同方式改成啟動haproxy2容器服務(在host6將標示1改成2)。

$ cp haproxy**1**.yaml docker-compose.yaml

$ docker compose up -d

1. **安裝mysql容器**

將mysql-cluster安裝資料夾複製到host7~host9節點，在host7節點中將檔案

pxc\_mysql1.yaml裡MYSQL\_ROOT\_PASSWORD及XTRABACKUP\_PASSWORD的預設密碼(兩組密碼要相同)改成自定義密碼，MYSQL\_PASSWORD為fido使用者密碼也由預設密碼改成自定義密碼；相同方式也在host8節點的檔案

pxc\_mysql2.yaml及host9節點的檔案pxc\_mysql3.yaml更改為自定義密碼，另外也將CLUSTER\_JOIN的預設值改成實際環境的對應值(host7的IP address)。

…

environment:

MYSQL\_ROOT\_PASSWORD: **Qazxsw23edc!!!**

MYSQL\_DATABASE: fido

MYSQL\_USER: fido

MYSQL\_PASSWORD: **Qazx**

CLUSTER\_NAME: PXC

XTRABACKUP\_PASSWORD: **Qazxsw23edc!!!**

CLUSTER\_JOIN: **192.168.50.8**

…

改完密碼及IP address後，在host7節點的安裝資料夾中依照下面指令啟動pxc\_mysql1容器服務；在host8及host9節點的安裝資料夾中用相同方式改成啟動pxc\_mysql2及pxc\_mysql3容器服務(在host8將標示1改成2，host9將標示1改成3)。

$ cd mysql-cluster

$ sudo mkdir ./v1pxc

$ sudo chown -R 1001:1001 ./v1pxc

$ sudo chmod -R 775 ./v1pxc

$ cp pxc\_mysql**1**.yaml docker-compose.yaml

$ docker compose up -d

**三、故障排除**

1. 當容器服務不正常運作時，要如何偵錯?

使用docker ps可以確認目前運作中的容器狀態並查取容器名稱例如為

haproxy1，再使用docker logs -f haproxy1可以監看haproxy1執行過程中所有產

生的相關或是錯誤訊息。

2. 編輯haproxy.cfg檔案後，啟動容器時出現該檔案的編碼錯誤訊息如何處理?

使用dos2unix haproxy.cfg轉成正確編碼。

3. mysql的使用者fido對資料庫fido無法讀寫?

參考mysql官方文件對使用者fido設定資料庫的存取權限。