

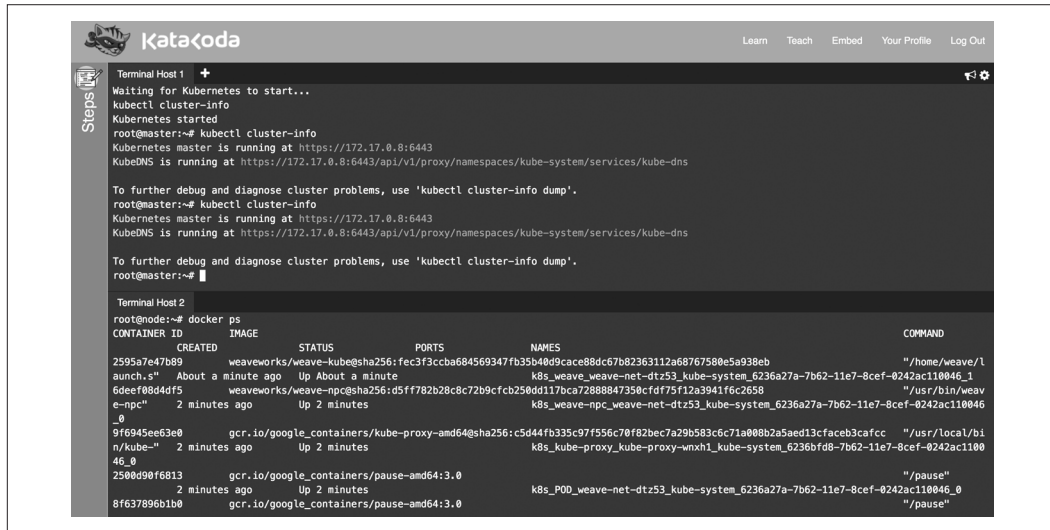


圖 1-1 Katacoda 提供的 Kubernetes Playground

注意你在 playground 啟動的環境，只能維持一段有限的時間（目前只有一小時），但它是免費的，而且只要有瀏覽器就能操作。

1.2 安裝 Kubernetes 的指令列介面 kubectl

問題

你想安裝 Kubernetes 的指令列介面，以便操作 Kubernetes 叢集。

解法

請以下列方式之一安裝 kubectl：

- 下載打包過的原始安裝檔（tarballs）。
- 利用套件管理員（package manager）來安裝。
- 自己從原始碼建置（參閱招式 13.1）。

請參閱文件 (<https://kubernetes.io/docs/tasks/kubectl/install/>)，裡面標註了數種取得 kubectl 的方式。最簡單的莫過於下載最新的官方發行版本。例如，在 Linux 系統裡，若要取得最新的穩定版本，請輸入：

```
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/\
$(curl -s https://storage.googleapis.com/kubernetes-release/\
release/stable.txt)\
/bin/linux/amd64/kubectl

$ chmod +x ./kubectl

$ sudo mv ./kubectl /usr/local/bin/kubectl
```

如果是 macOS 的使用者，可以用 Homebrew 來取得 kubectl：

```
$ brew install kubectl
```

使用 Google Kubernetes Engine 的人（參閱招式 2.7），只需先裝好 gcloud 命令，就可以連帶取得 kubectl。以筆者自己機器為例：

```
$ which kubectl
/Users/sebgoa/google-cloud-sdk/bin/kubectl
```

此外也注意，最新版的 Minikube（參閱招式 1.3）套件裡也包含了 kubectl，而且安裝時還會順便更改路徑變數 \$PATH。

在繼續讀下去之前，請先試著查詢你的 kubectl 版本，確認它可以運作。這道指令同時也會試著取得預設 Kubernetes 叢集的版本：

```
$ kubectl version
Client Version: version.Info{Major:"1", \
    Minor:"7", \
    GitVersion:"v1.7.0", \
    GitCommit:"fff5156...", \
    GitTreeState:"clean", \
    BuildDate:"2017-03-28T16:36:33Z", \
    GoVersion:"go1.7.5", \
    Compiler:"gc", \
    Platform:"darwin/amd64"}
...
```

探討

要建立一個應用程式，請按下右上角的建立（Create）按鈕，並為應用程式命名，然後指定你要使用的 Docker 映像檔。然後按下部署（Deploy）按鈕，你就會在另一個新畫面中看到部署成果和抄本集合（replica sets），稍後還會看到一個 pod。這些都是關鍵的基本 API，在本書接下來的章節裡我們還會進一步處理它們。

圖 1-3 呈現的，是在使用 Redis 容器建立單一應用程式後的典型儀表板外觀。

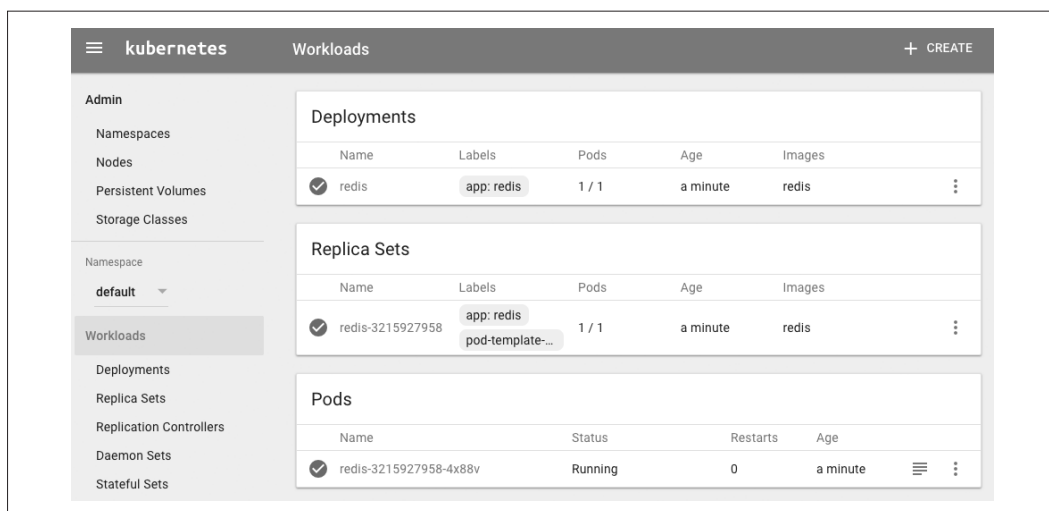


圖 1-3 一個 Redis 應用程式的儀表板外觀

如果你回到終端機使用指令列用戶端觀察，就會發現一樣的內容：

```
$ kubectl get pods,rs,deployments
```

```
NAME                                READY    STATUS    RESTARTS   AGE
po/redis-3215927958-4x88v          1/1      Running   0           24m
```

```
NAME                                DESIRED   CURRENT   READY    AGE
rs/redis-3215927958                1         1         1        24m
```

```
NAME            DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/redis    1         1         1             1           24m
```

2.3 從 GitHub 下載 Kubernetes 發行內容

問題

想下載官方發行的 Kubernetes，不想自己用原始碼編譯。

解法

請遵照手動程序，前往 GitHub 發行頁面（<https://github.com/kubernetes/kubernetes/releases>）。選取你要下載的發行版本、或是潛在的搶鮮版。然後選擇你需要編譯的原始碼包裝，或是下載 *kubernetes.tar.gz* 檔案。

抑或是利用 GitHub API 檢視最新發行版本標籤，就像這樣：

```
$ curl -s https://api.github.com/repos/kubernetes/kubernetes/releases | \
jq -r .[].assets[].browser_download_url
https://github.com/kubernetes/kubernetes/releases/download/v1.9.0/
kubernetes.tar.gz
https://github.com/kubernetes/kubernetes/releases/download/v1.9.0-beta.2/
kubernetes.tar.gz
https://github.com/kubernetes/kubernetes/releases/download/v1.8.5/
kubernetes.tar.gz
https://github.com/kubernetes/kubernetes/releases/download/v1.9.0-beta.1/
kubernetes.tar.gz
https://github.com/kubernetes/kubernetes/releases/download/v1.7.11/
kubernetes.tar.gz
...
```

然後下載你需要的 *kubernetes.tar.gz* 發行套件。例如，要取得 1.7.11 版時：

```
$ wget https://github.com/kubernetes/kubernetes/releases/download/\
v1.7.11/kubernetes.tar.gz
```

如果要從原始碼編譯 Kubernetes，請參閱招式 13.1。



別忘了要驗證 *kubernetes.tar.gz* 檔案的安全雜湊值（secure hash）。GitHub 發行頁面會列有 SHA256 hash 值。下載檔案後，請自行產生雜湊值並進行比對。就算發行版本未經 GPG 簽署，驗證雜湊值還是會有驗明檔案正身的效果。

現在來看看部署後的情形：

```
$ kubectl create -f fancyapp.yaml
deployment "fancyapp" created
```

```
$ kubectl get deploy
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
fancyapp	5	5	5	0	8s

```
$ kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
fancyapp-1223770997	5	5	0	13s

```
$ kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
fancyapp-1223770997-18msl	0/1	ContainerCreating	0	15s
fancyapp-1223770997-1zdg4	0/1	ContainerCreating	0	15s
fancyapp-1223770997-6rqn2	0/1	ContainerCreating	0	15s
fancyapp-1223770997-7bnbh	0/1	ContainerCreating	0	15s
fancyapp-1223770997-qxg4v	0/1	ContainerCreating	0	15s

如果過個幾秒鐘再做同樣的動作：

```
$ kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
fancyapp-1223770997-18msl	1/1	Running	0	1m
fancyapp-1223770997-1zdg4	1/1	Running	0	1m
fancyapp-1223770997-6rqn2	1/1	Running	0	1m
fancyapp-1223770997-7bnbh	1/1	Running	0	1m
fancyapp-1223770997-qxg4v	1/1	Running	0	1m



當你要清除某個部署、以及它所監督的副本集合和 pod，請執行像是 `kubectl delete deploy/fancyapp` 這樣的命令。切勿嘗試直接刪除個別的 pod，因為部署還會把它們還原回來。這是初學者常犯的錯誤。

你可以透過部署任意調整應用程式規模（參閱招式 9.1），也可以藉此推出新版本、甚至退回到先前的舊版本。通常這種方式僅適用於無狀態（stateless）、需要所有 pod 都具備同等特性的應用程式。

探討

一份部署負責監管若干個 pod 和抄本集合（replica set，簡稱 RS），透過部署，你可以鉅細靡遺地掌控新版 pod 的推出方式和時間、或者是還原至先前的狀態。通常你不會在意一份部署監督哪些 RS 和 pod，除非需要替某個 pod 除錯（參閱招式 12.5）。圖 4-1 便說明了如何在不同部署間來回調整。

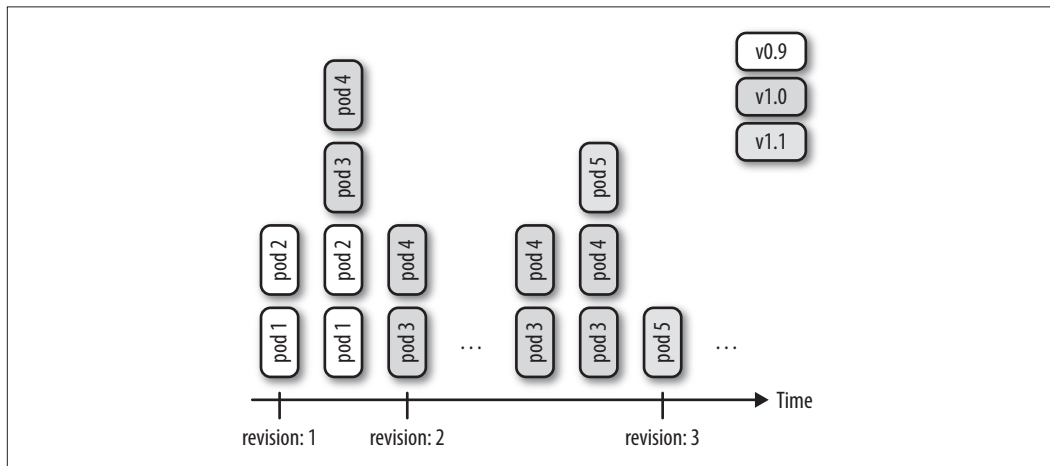


圖 4-1 部署的調整

注意 RS 還是會持續發展，並取代原有的抄本控制器（replication controller，RC），因此你最好以 RS 為考量重點、而非 RC。在這個節骨眼，唯一的差別只在於 RS 可以支援以集合為基礎的標示 / 查詢，但我們可以確信的是，隨著 RS 功能的逐漸增加，RC 則終將淘汰。

最後，為了產生一個項目清單，你可以使用 `kubectl create` 指令和 `--dry-run` 選項。這樣可以產生出 YAML 或 JSON 格式的項目清單，然後儲存起來以待後用。舉例來說，若要用 Docker 映像檔 `nginx` 建立一個部署用的項目清單，檔名為 `fancy-app`，就這般下令：

```
$ kubectl create deployment fancyapp --image nginx -o json --dry-run
{
  "kind": "Deployment",
  "apiVersion": "extensions/v1beta1",
  "metadata": {
    "name": "fancy-app",
    "creationTimestamp": null,
```

服務的處理

本章要探討的是叢集裡的 pod 如何溝通、應用程式又是如何尋找彼此的存在，還有如何將 pod 對外公開、以便讓人從叢集外部取用它們。

此處我們所仰賴的原理稱為一個 Kubernetes 的服務（service, <https://kubernetes.io/docs/concepts/services-networking/service/>），如圖 5-1 所示。

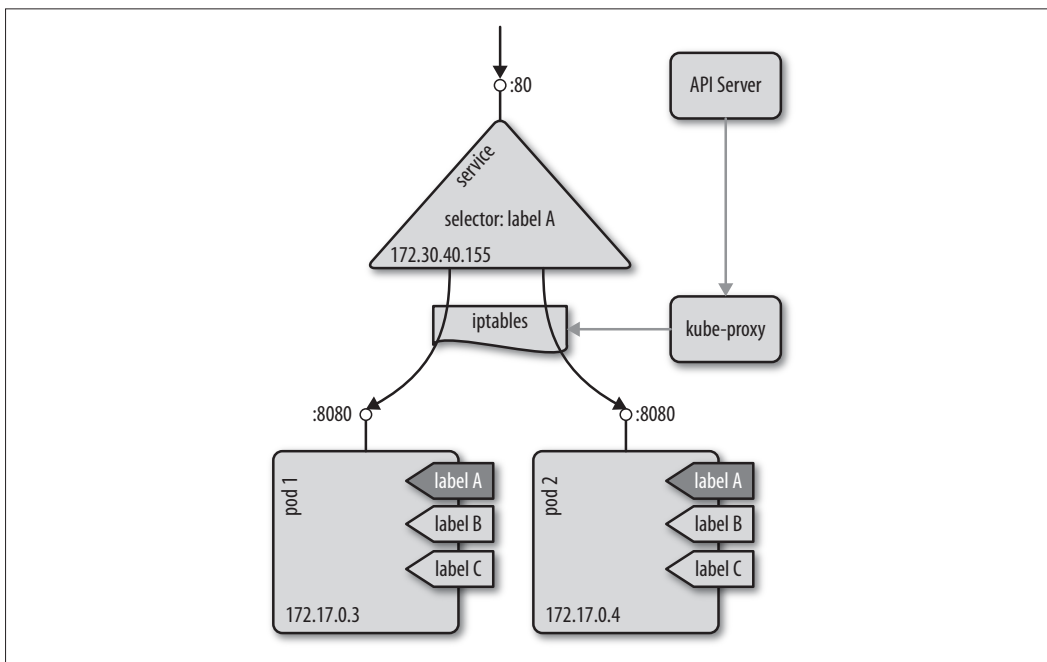


圖 5-1 Kubernetes 的服務概念

```

working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

```

探討

一般來說，入口的作用方式如圖 5-4 所示：入口控制器會傾聽 API 伺服器的 `/ingresses` 端點，並得知新規範的存在。然後據此設定路徑，讓外部流量找到特定的（叢集之內的）服務（即本例中位於 9876 通訊埠的 `service1`）。

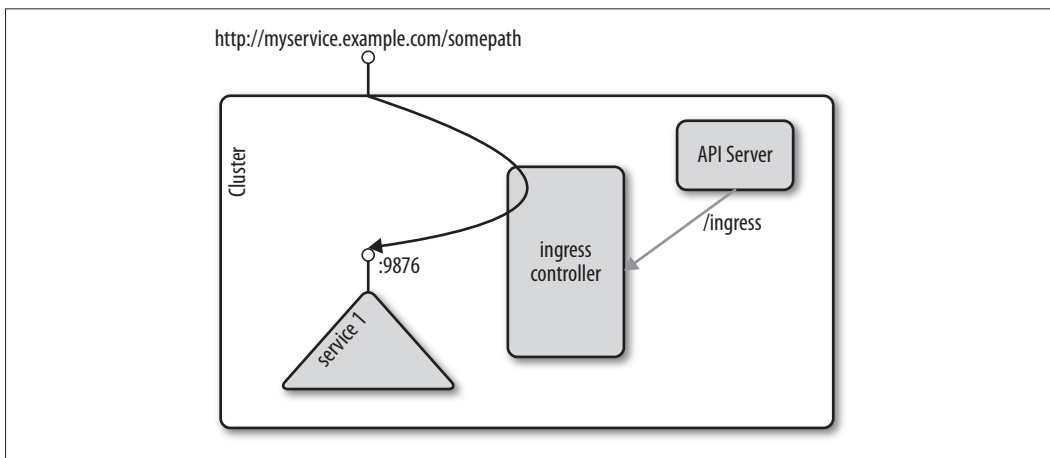


圖 5-4 入口的概念



這個招式使用了 Minishift，它是一種現成的入口控制器附加功能。通常你需要自行設置入口控制器；相關指示的範例請參閱 GitHub（<https://github.com/kubernetes/ingress-nginx>）。



請檢視完整的指令說明（`kubectl label --help`）。從中可以學到如何移除標籤、覆寫既有標籤、以及如何標記某個命名空間裡的所有資源。

探討

在 Kubernetes 裡，你可以透過標籤，以非階層的彈性方式來編排物件。一個標籤由一對鍵 / 值組成，毋須在 Kubernetes 裡預先定義。換句話說，系統不會去解讀鍵 / 值的內容。你可以用標籤來表達某種歸屬關係（例如物件 X 屬於 ABC 部門）、或是某種環境（如某種在正式環境中執行的服務）、或是任何你自訂的物件編排方式。注意標籤的名稱長度及其許可值仍有限制存在。²

6.6 利用標籤來查詢

問題

想要有效率地查詢物件。

解法

利用 `kubectl get --selector` 指令即可。例如，有以下的 pod 要查詢：

```
$ kubectl get pods --show-labels
NAME                                READY   ...   LABELS
cockroachdb-0                      1/1     ...   app=cockroachdb,
cockroachdb-1                      1/1     ...   app=cockroachdb,
cockroachdb-2                      1/1     ...   app=cockroachdb,
jump-1247516000-sz87w              1/1     ...   pod-template-hash=1247516000,run=jump
nginx-4217019353-462mb             1/1     ...   pod-template-hash=4217019353,run=nginx
nginx-4217019353-z3g8d             1/1     ...   pod-template-hash=4217019353,run=nginx
prom-2436944326-pr60g             1/1     ...   app=prom,pod-template-hash=2436944326
```

² 參閱 Kubernetes 官網「標籤與選擇器：語法和字元集」（“Labels and Selectors: Syntax and character set”，<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set>）。

```

Name:          fluentd
Selector:      app=fluentd
Node-Selector: <none>
Labels:        app=fluentd
Annotations:   <none>
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:   1 Running / 0 Waiting / 0 Succeeded / 0 Failed
...

```

探討

注意以上的輸出，由於指令是在 Minikube 裡執行的，你只會看到一個 pod 在運行，這是因為 Minikube 原本就只設置一個節點之故。如果你的叢集裡有 15 個節點，你就會得到總共 15 個 pod、而且剛好每個節點運行 1 個 pod。你也可以透過 DaemonSet 項目清單 spec 裡的 nodeSelector 區段，把服務侷限在特定的節點上。

7.4 管理有狀態的 Leader/Follower 應用程式

問題

運行一支應用程式，其 pod 擁有彼此互異的潛在特質，像是某個資料庫，由一個 leader 來處理讀寫；其他數個 follower 則只接受讀取。你無法只靠部署達成這種配置，因為部署只能配置完全一致的 pod，但你需要的是一個監督者，由它來處理不同的 pod，就像是寵物跟家畜的差別一般。

解法

利用 StatefulSet，它可以透過獨特的網路名稱啟用工作負載，溫和地進行部署 / 調整 / 終止等動作、或是持久性儲存。舉例來說，若你想要運行廣受喜愛的可延展資料儲存 CockroachDB，可以利用下例²，其核心包含以下的 StatefulSet 段落：

² GitHub 上的 Kubernetes cockroachdb 範例檔 cockroachdb-statefulset.yaml (<https://github.com/k8s-cookbook/recipes/blob/master/ch07/cockroachdb-statefulset.yaml>)。

7.5 影響 Pod 的啟動行為

問題

你的 pod 必須仰賴若干其他服務，只有這些服務堪用時 pod 才能正常運作。

解法

利用 init 容器 (<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>) 來影響 pod 的啟動行為。

設想你要啟動一個 nginx 網頁伺服器，而且必須倚靠後台的服務來提供內容。因此你想要確保 nginx 的 pod 只會在後台服務已經正常運作的前提下才會啟動。

首先，建立一個網頁伺服器所需仰賴的後台服務：

```
$ kubectl run backend --image=mhausenblas/simple-service:0.5.0
deployment "backend" created

$ kubectl expose deployment backend --port=80 --target-port=9876
```

然後你就可以使用以下的項目清單 `nginx-init-container.yaml` 來啟動 nginx 的實例，且確保前者只有在 `backend` 部署已畢、可以提供資料時，方才啟動：^{譯註}

```
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
```

^{譯註} 從本書的 GitHub 頁面下載 <https://github.com/k8s-cookbook/recipes/blob/master/ch07/nginx-init-container.yaml> 範例後，記得把 `initContainers` 段落的 `image:busybox` 改成 `image:busybox:1.28`，這樣才能避開 busybox 最近版本的 DNS 查詢問題（參閱招式 5.2 的譯註）。

探討

Secret 以命名空間為存在背景，因此在設置或使用它時均須考慮到這一點。

你可以透過以下方式，從運行在 pod 上的容器中取用 secret：

- 卷冊（如以上解法說明，其內容存放在 `tmpfs` 的卷冊裡）
- 環境變數（<https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets-as-environment-variables>）

還有，請注意 secret 的大小不能超過 1 MB。

除了使用者自訂的 secret，Kubernetes 還會替服務帳號自動建立 secret，以便取用 API。舉例來說，如果安裝了 Prometheus（參閱招式 11.6），你就會在 Kubernetes 儀表板看到像圖 8-1 所示的畫面。

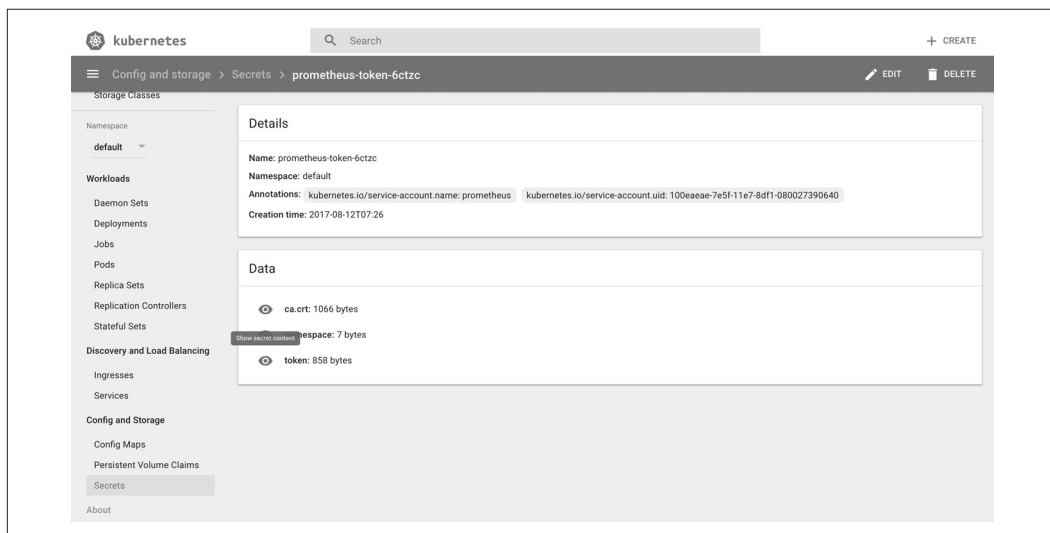


圖 8-1 Prometheus 服務帳號的 secret 畫面