Alibaba Cloud Container Service for Kubernetes

使用者指南

檔案版本：20180929

目錄

1 Kubernetes 叢集

1.1 簡介

1.1.1 概述

Kubernetes 是流行的開源容器編排技術。為了讓使用者可以方便地在阿里雲上使用 Kubernetes 管理容器應用，阿里雲Container Service提供了 Kubernetes 叢集支援。

您可以通過Container Service管理主控台建立一個安全高可用的 Kubernetes 叢集，整合阿里雲虛擬化、儲存、網路和安全能力，提供高效能可伸縮的容器應用管理能力，簡化叢集的搭建和擴容等工作，讓您專註於容器化的應用的開發與管理。

Kubernetes 支援對容器化應用程式的部署、擴充和管理。它具有以下功能：

- 彈性擴充和自我修複
- 服務發現和負載平衡
- 服務發布與復原
- 機密和組態管理

使用限制

- 目前 Kubernetes 叢集只支援 Linux 容器，對 Kubernetes 的 Windows 容器的支援在計劃中。
- 目前 Kubernetes 叢集只支援 VPC 網路。您可以在部署 Kubernetes 叢集時選擇建立一個新的 VPC 或者使用已有的 VPC。

相關開源項目

- 阿里雲 Kubernetes Cloud Provider 實現：*https://github.com/AliyunContainerService/kubernetes*
- Flannel 的阿里雲 VPC 網路驅動：*https://github.com/coreos/flannel/blob/master/Documentation/alicloud-vpc-backend.md*

如果您對相關項目有問題或者建議，歡迎在社區提交 Issue 或者 Pull Request。

1.1.2 阿里雲 Kubernetes VS 自建 Kubernetes

阿里雲 **Kubnernetes** 的優勢

便捷

- 通過 Web 介面一鍵建立 Kubernetes 叢集。

- 通過 Web 介面一鍵完成 Kubernetes 叢集的升級。

  您在使用自建 Kubernetes 叢集的過程中，可能需要同時處理多個版本的叢集（包括 1.8.6、1.9.4、以及未來的 1.10）。每次升級叢集的過程都是一次大的調整和巨大的營運負擔。Container Service的升級方案使用鏡像滾動升級以及完整元資料的備份策略，允許您方便地復原到先前版本。
- 通過 Web 介面輕鬆地實現 Kubernetes 叢集的擴容和縮容。

  使用Container Service Kubernetes 叢集可以方便地一鍵垂直伸縮容來快速應對資料分析業務的峰值。

強大

功能	說明
網路	• 高效能 VPC 網路外掛程式。 • 支援 network policy 和流控。 Container Service可以為您提供持續的網路整合和最佳的網路最佳化。
負載平衡	支援建立Server Load Balancer執行個體（公網、內網）。 如果您在使用自建 Kubernetes 叢集的過程使用自建的 Ingress 實現，業務發布頻繁可能會造成 Ingress 的配置壓力並增加出錯機率。Container Service的 SLB 方案支援原生的阿里雲高可用負載平衡，可以自動完成網路設定的修改和更新。該方案經曆了大量使用者長時間的使用，穩定性和可靠性大大超過使用者自建的入口實現。
儲存	整合阿里雲雲端硬碟、Network Attached Storage、區塊存放裝置EBS，提供標準的 FlexVolume 驅動。 自建 Kubernetes 叢集無法使用雲上的儲存資源，阿里雲Container Service提供了最佳的無縫整合。
營運	• 整合阿里雲Log Service、Cloud Monitor • 支援Auto Scaling
鏡像倉庫	• 高可用，支援大並發。 • 支援鏡像加速。 • 支援 p2p 分發。

功能	說明
	您如果使用自建的鏡像倉庫，在百萬級的用戶端同時拉取鏡像的時候，會存在鏡像倉庫崩潰的可能性。使用Container Service鏡像倉庫的專有雲版本來提高鏡像倉庫的可靠性，減少營運負擔和升級壓力。
穩定	• 專門的團隊保障容器的穩定性。 • 每個 Linux 版本，每個 Kubernetes 版本都會在經過嚴格測試之後之後才會提供給使用者。 Container Service提供了 Docker CE 兜底和推動 Docker 修複的能力。當您遇到 Docker Engine hang、網路問題、核心相容等問題時，Container Service可以為您提供最佳實務。
高可用	• 提供多可用性區域支援。 • 支援備份和容災。
支援人員	• 提供 Kubernetes 升級能力，新版本一鍵升級。 • 阿里雲容器團隊負責解決在環境中遇到的各種容器問題。

自建 **Kubernetes** 的成本和風險

- 搭建叢集繁瑣。

  您需要手動設定 kubernetes 相關的各種組件、設定檔、認證、密鑰、相關外掛程式和工具，整個叢集搭建工作需要專業人員數天到數周的時間。
- 在公用雲上，需要投入大量的成本實現和雲產品的整合。

  和阿里雲上其他產品的整合，需要您自己投入成本來實現，如Log Service、監控服務和儲存管理等。
- 容器是一個系統性工程，涉及網路、儲存、作業系統、編排等各種技術，需要專門的人員投入。
- 容器技術一直在不斷髮展，版本迭代快，需要不斷的踩坑、升級、測試。

1.2 授權管理

1.2.1 使用子帳號

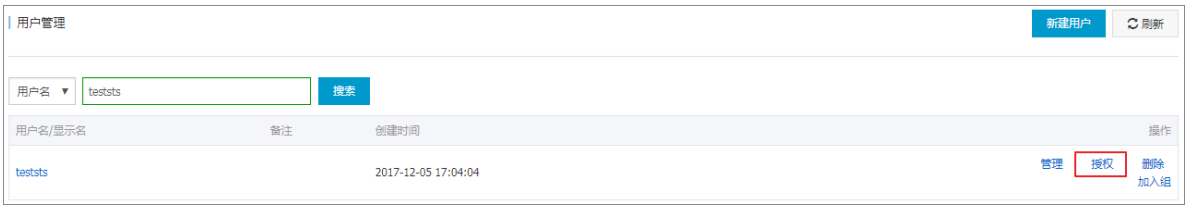
使用子帳號登入Container Service控制台並進行相關操作之前，您需要賦予子帳號相應的許可權。

步驟 1 建立子帳號並開啟控制台登入

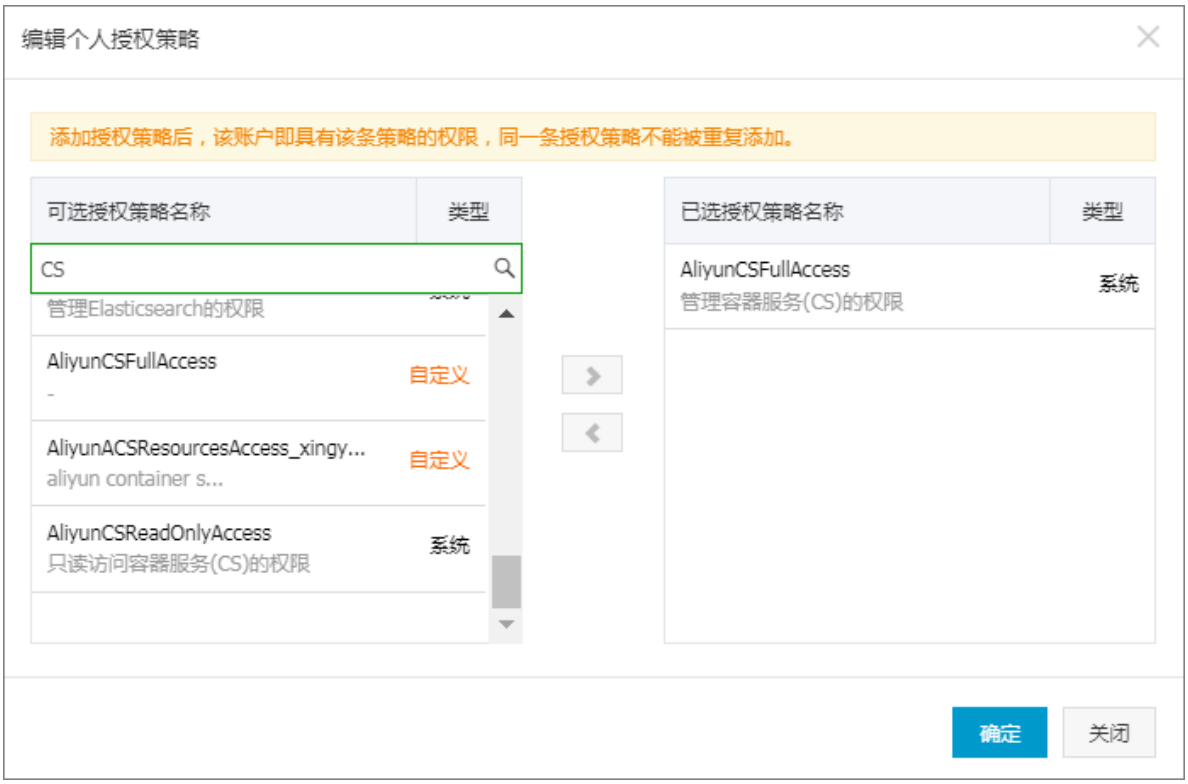
1. 登入 #########。
2. 單擊左側導覽列中的使用者管理並單擊頁面右上方的建立使用者。
3. 填寫子帳號的名稱並單擊確定。
4. 在使用者管理頁面，選擇建立的子帳號，單擊右側的管理。
5. 在Web控制台登入管理中，單擊啟用控制台登入。
6. 輸入登入密碼並單擊確定。

步驟 2 授予子帳號訪問Container Service的許可權

1. 在使用者管理頁面，選擇建立的子帳號，單擊右側的授權。



2. 將所需的策略授權給子帳號。



您可以使用系統預設的授權策略。

- AliyunCSFullAccess：Container Service的系統管理權限。
- AliyunCSReadOnlyAccess：Container Service的唯讀許可權。

或者根據您自己的需要自訂授權策略並授權給子帳號，參見########。

步驟 3 子帳號登入**Container Service**控制台

使用子帳號登入 *Container Service#####*。

如果您之前已經給主帳號授予了 AliyunCSDefaultRole 和 AliyunCSClusterRole 角色，子帳號可以直接登入到Container Service管理主控台，並進行相應的操作。

如果之前您沒有給主帳號授予 AliyunCSDefaultRole 和 AliyunCSClusterRole 角色，則會看到如下提示。單擊同意授權。

完成以上授權後，重新整理Container Service控制台，然後就可以進行操作了。

1.2.2 建立自訂授權策略

Container Service提供的系統授權策略的授權粒度比較粗，如果這種粗粒度授權策略不能滿足您的需要，那麼您可以建立自訂授權策略。比如，您想控制對某個具體的叢集的操作許可權，您必須使用自訂授權策略才能滿足這種細粒度要求。

建立自訂授權策略

在建立自訂授權策略時，您需要瞭解授權策略語言的基本結構和文法，相關內容的詳細描述請參考########。

本文檔以授予子帳號查詢、擴容和刪除叢集的許可權為例進行說明。

操作步驟

1. 使用主帳號登入*RAM#####*。
2. 單擊左側導覽列中的策略管理並單擊頁面右上方的建立授權策略。
3. 選擇一個模板，填寫授權策略名稱並編寫您的授權策略內容。


```
{
 "Statement": [{
     "Action": [
         "cs:Get*",
         "cs:ScaleCluster",
         "cs:DeleteCluster"
     ],
     "Effect": "Allow",
     "Resource": [
         "acs:cs:*:*:cluster/叢集ID"
     ]
 }],
 "Version": "1"
}
```

其中：

- Action 處填寫您所要授與權限。

  > 说明：
  > 所有的 Action 均支援萬用字元。

- Resource 有如下配置方式。

- 授予單叢集許可權

```
"Resource": [
     "acs:cs:*:*:cluster/叢集ID"
 ]
```

- 授予多個叢集許可權

```
"Resource": [
     "acs:cs:*:*:cluster/叢集ID",
     "acs:cs:*:*:cluster/叢集ID"
 ]
```

- 授予您所有叢集的許可權

```
"Resource": [
     "*"
 ]
```

其中，叢集ID 需要替換為您要授權的真實的叢集 ID。

4. 編寫完畢後，單擊建立授權策略。

表 **1-1: Container ServiceRAM Action**

Action	說明
CreateCluster	建立叢集
AttachInstances	向叢集中添加已有ECS執行個體
ScaleCluster	擴容叢集
GetClusters	查看叢集列表
GetClusterById	查看叢集詳情
ModifyClusterName	修改叢集名稱
DeleteCluster	刪除叢集
UpgradeClusterAgent	升級叢集Agent
GetClusterLogs	查看叢集的動作記錄
GetClusterEndpoint	查看叢集存取點地址
GetClusterCerts	下載叢集認證
RevokeClusterCerts	吊銷叢集認證
BindSLB	為叢集綁定Server Load Balancer執行個體
UnBindSLB	為叢集解除綁定Server Load Balancer執行個體
ReBindSecurityGroup	為叢集重新綁定安全性群組

Action	說明
CheckSecurityGroup	檢測叢集現有的安全性群組規則
FixSecurityGroup	修複叢集的安全性群組規則
ResetClusterNode	重設叢集中的節點
DeleteClusterNode	移除叢集中的節點
CreateAutoScale	建立節點Auto Scaling規則
UpdateAutoScale	更新節點Auto Scaling規則
DeleteAutoScale	刪除節點Auto Scaling規則
GetClusterProjects	查看叢集下的應用
CreateTriggerHook	為應用建立觸發器
GetTriggerHook	查看應用的觸發器列表
RevokeTriggerHook	刪除應用的觸發器
CreateClusterToken	建立 Token

1.2.3 子帳號Kubernetes應用許可權配置指導

本文旨在協助您瞭解如何通過Container Service控制台配置子帳號對應的Kubernetes RAM叢集許可權和在叢集內相應的Kubernetes RBAC應用許可權。

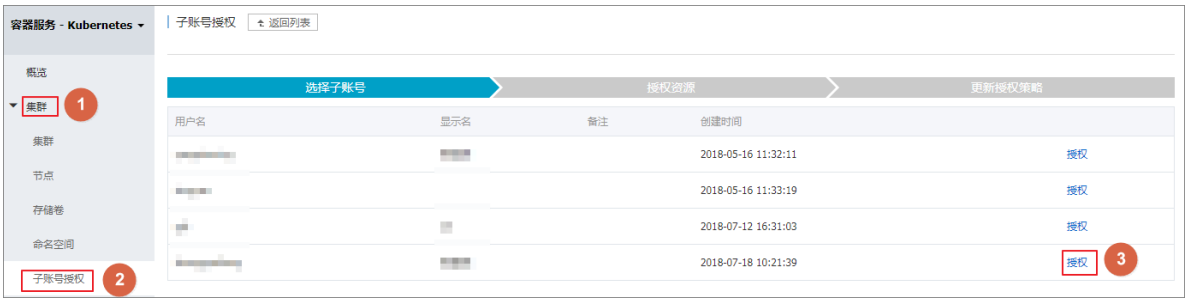
配置說明

- 子帳號授權頁面僅主帳號可見。您需要擁有一個阿里雲主帳號，並有一個或若干個子帳號。
- 由於阿里雲RAM的安全限制，當您通過Container Service控制台的授權配置涉及到子帳號RAM授權的修改時，需要您按照頁面上給出的參考策略內容和操作說明，在RAM控制台進行目標子帳號的手動授權。

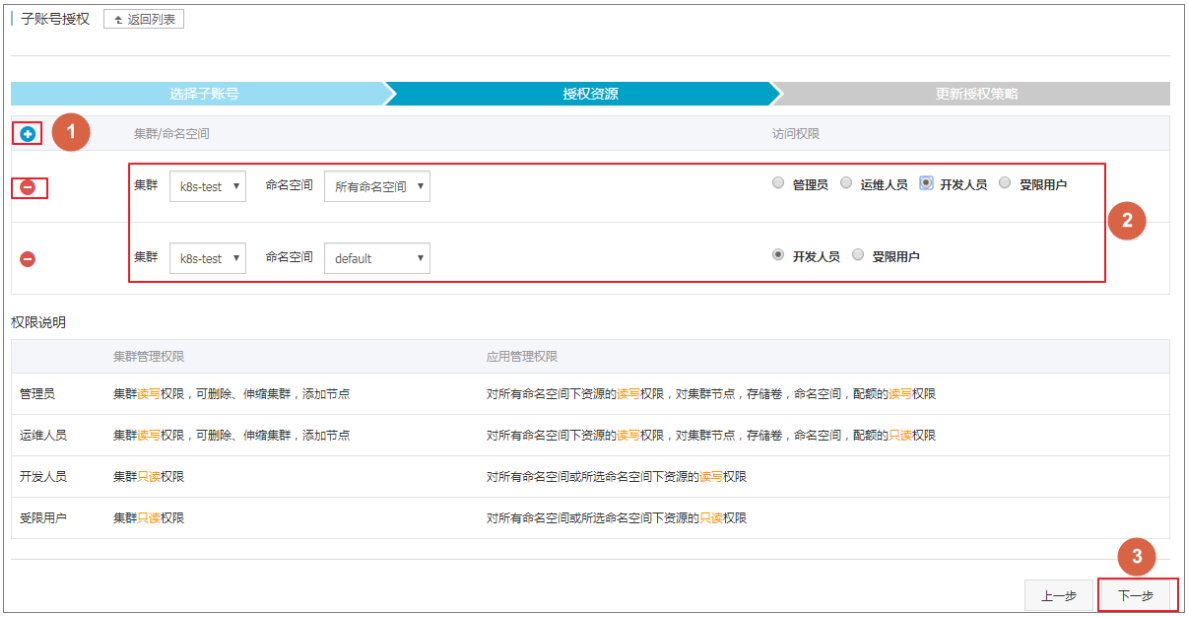
操作步驟


说明：
因為子帳號授權頁面僅主帳號可見，您需要用主帳號登入Container Service控制台。

1. 登入*Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的叢集 > 子帳號授權，進入子帳號授權頁面。
3. 在子帳號列表中選擇需要授權的子帳號，單擊授權。



4. 進入授權資源頁面，您可以通過單擊表格左上方的加號添加叢集或命名空間層級的許可權配置，並選擇相應的預置角色；也可以單擊配置行首的減號刪除目標角色。



叢集和命名空間的預置角色定義可查看下面的角色許可權說明：

表 1-2: 角色許可權說明

	叢集系統管理權限	應用系統管理權限
管理員	叢集讀寫權限，可刪除、伸縮叢集，添加節點	對所有命名空間下資源的讀寫權限，對叢集節點，儲存卷，命名空間，配額的讀寫權限
營運人員	叢集讀寫權限，可刪除、伸縮叢集，添加節點	對所有命名空間下資源的讀寫權限，對叢集節點，儲存卷，命名空間，配額的唯讀許可權
開發人員	叢集唯讀許可權	對所有命名空間或所選命名空間下資源的讀寫權限
受限人員	叢集唯讀許可權	對所有命名空間或所選命名空間下資源的唯讀許可權

5. 完成配置後，如果涉及目標子帳號RAM叢集許可權的變更，在更新授權策略頁面會展示配置項對應的Kubernetes叢集RAM許可權參考配置，您可以根據頁面中的指導在RAM控制台完成子帳號授權的更新。

補充說明

為了不影響子帳號對當前存量可訪問kubernetes叢集的正常使用，Container Service控制台會暫時相容舊的叢集存取權限控制，在一段時間對原有子帳號可訪問kubernetes叢集不進行RBAC的應用許可權校正，如果您是子帳號使用者，請您根據叢集類型和相容方式，及時聯絡主帳號進行授權操作。

對於由當前子帳號自身建立的存量叢集，可以通過在叢集詳情頁面單擊升級當前叢集授權資訊，完成叢集應用許可權的自動升級。

在公告期結束後，仍舊沒有由主帳號授權或進行許可權管理升級的子帳號，子帳號將被禁止訪問叢集對應的應用控制台。

表 **1-3:** 相容叢集說明

相容叢集類型	相容方式
存量子帳號建立叢集	提示許可權管理升級公告，提供一鍵升級連結，子帳號可通過點擊升級連結完成應用授權
存量RAM授權訪問叢集	提示許可權管理升級公告，請及時聯絡主帳號完成應用授權
建立RAM授權訪問叢集	提示許可權管理升級公告，請及時聯絡主帳號完成應用授權

1.3 叢集管理

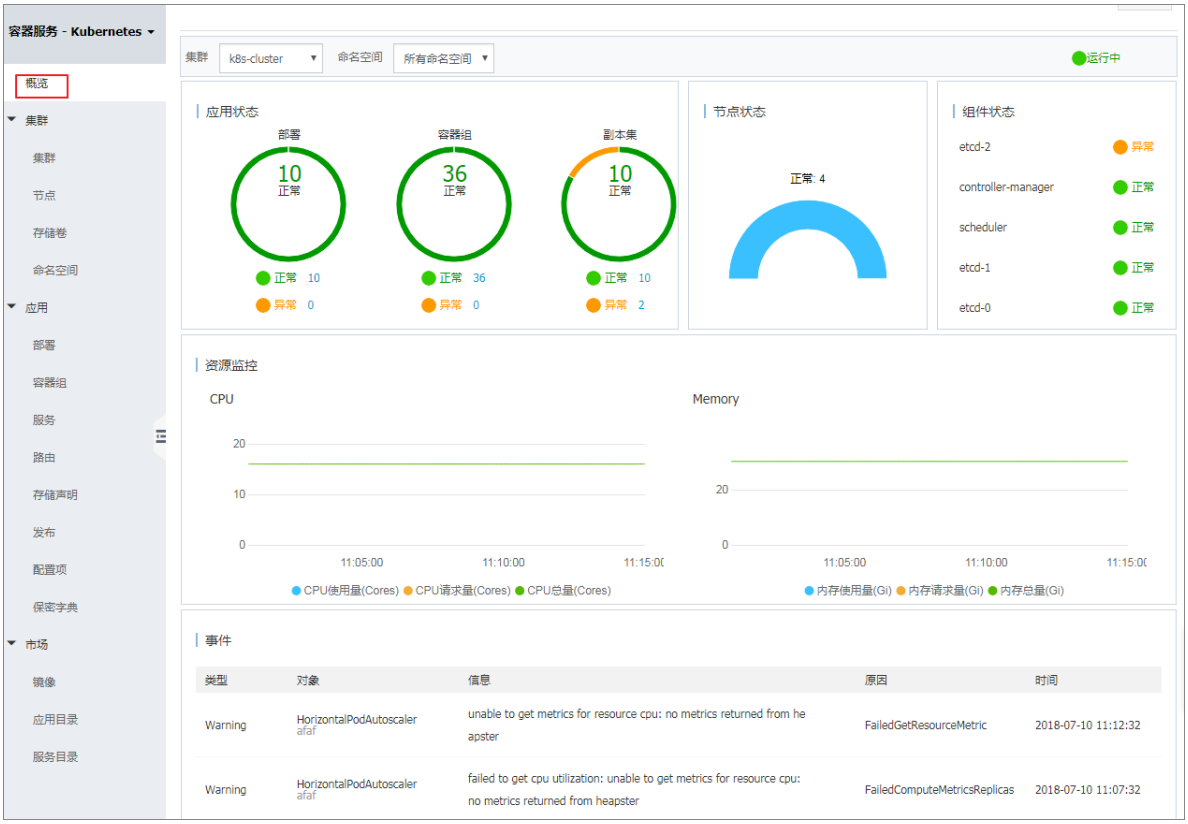
1.3.1 查看叢集概覽

阿里雲Container Service Kubernetes 叢集提供叢集概覽功能，提供應用狀態、組件狀態和資源監控等功能，方便您快速瞭解叢集的健康狀態資訊。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的概覽，進入 Kubernetes 叢集概覽頁面。
3. 選擇所需的叢集和命名空間，您可查看應用狀態、組件狀態和資源監控圖表。

- 應用狀態：顯示當前啟動並執行部署、容器組和複本集的狀態示意圖，綠色表徵圖代表正常，黃色表徵圖代表異常。
- 節點狀態：顯示當前叢集的節點狀態。
- 組件狀態：Kubernetes 叢集的組件通常部署在 kube-system 命名空間下，包括 scheduler、controller-manager和 etcd 等核心組件。
- 資源監控：提供 CPU 和記憶體的監控圖表。CPU 統計單位為 Cores（核），可顯示小數點後 3 位，最小統計單位是 millcores，即一個核的 1/1000；記憶體的統計單位是 Gi，顯示小數點後 3 位。更多相關資訊，請參見 *Meaning of CPU*和 *Meaning of memory*。
- 事件：顯示叢集的事件資訊，例如警告和錯誤事件等。



1.3.2 建立Kubernetes叢集

您可以通過Container Service管理主控台非常方便地快速建立 Kubernetes 叢集。

使用須知

建立叢集過程中，Container Service會進行如下操作：

- 建立 ECS，組態管理節點到其他節點的 SSH 的公開金鑰登入，通過 CloudInit 安裝配置 Kubernetes 叢集。
- 建立安全性群組，該安全性群組允許 VPC 入方向全部 ICMP 連接埠的訪問。

- 如果您不使用已有的 VPC 網路，會為您建立一個新的 VPC 及 VSwitch，同時為該 VSwitch 建立 SNAT。
- 建立 VPC 路由規則。
- 建立 NAT Gateway及 EIP。
- 建立 RAM 子帳號和 AK，該子帳號擁有 ECS 的查詢、執行個體建立和刪除的許可權，添加和刪除雲端硬碟的許可權，SLB 的全部許可權，Cloud Monitor的全部許可權，VPC 的全部許可權，Log Service的全部許可權，NAS 的全部許可權。Kubernetes 叢集會根據使用者部署的配置相應的動態建立 SLB，雲端硬碟，VPC路由規則。
- 建立內網 SLB，暴露 6443 連接埠。
- 建立公網 SLB，暴露 6443、8443和 22 連接埠（如果您在建立叢集的時候選擇開放公網 SSH 登入，則會暴露 22 連接埠；如果您選擇不開放公網 SSH 訪問，則不會暴露 22 連接埠）。

前提條件

您需要開通Container Service、Resource Orchestration Service（ROS）服務和存取控制（RAM）服務。

登入 *Container Service#####*、*ROS #####* 和 *RAM #####* 開通相應的服務。

说明：

Container Service Kubernetes 叢集部署依賴阿里雲Resource Orchestration Service 的應用部署能力，所以建立 Kubernetes 叢集前，您需要開通 ROS。

使用限制

- 隨叢集一同建立的Server Load Balancer執行個體只支援隨用隨付的方式。
- Kubernetes 叢集僅支援專用網路 VPC。
- 每個帳號預設可以建立的雲資源有一定的配額，如果超過配額建立叢集會失敗。請在建立叢集前確認您的配額。如果您需要提高您的配額，請提交工單申請。
  - 每個帳號預設最多可以建立 5 個叢集（所有地區下），每個叢集中最多可以添加 40 個節點。如果您需要建立更多的叢集或者節點，請提交工單申請。
  - 每個帳號預設最多可以建立 100 個安全性群組。
  - 每個帳號預設最多可以建立 60 個隨用隨付的Server Load Balancer執行個體。
  - 每個帳號預設最多可以建立 20 個EIP。
- ECS 執行個體使用限制：
  - 僅支援 CentOS 作業系統。

— 支援建立隨用隨付和訂用帳戶的ECS執行個體。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的叢集，進入叢集列表頁面。
3. 單擊頁面右上方的建立 **Kubernetes** 叢集，進入建立 **Kubernetes** 叢集頁面。

預設進入**kubernetes**叢集配置頁面。

4. 填寫叢集的名稱。

   叢集名稱應包含1-63個字元，可包含數字、漢字、英文字元或連字號（-）。

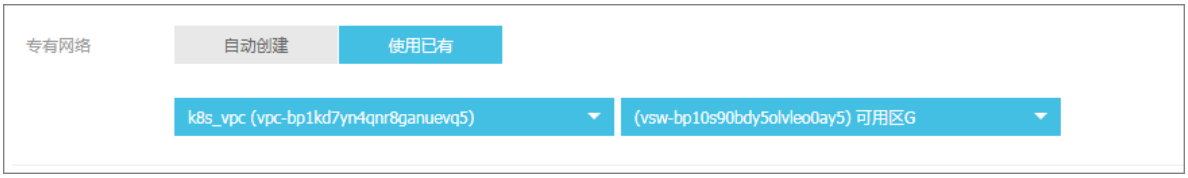
5. 選擇叢集所在的地區和可用性區域。

6. 設定叢集的網路。Kubernetes 叢集僅支援專用網路。

   專用網路：您可以選擇自動建立（建立 Kubernetes 叢集時，同步建立一個 VPC）或者使用已有（使用一個已有的 VPC）。選擇使用已有後，您可以在已有 VPC 列表中選擇所需的 VPC 和交換器。

   - 選擇自動建立，建立叢集時，系統會自動為您的 VPC 建立一個 NAT Gateway。
   - 選擇使用已有，如果您使用的 VPC 中當前已有 NAT Gateway，Container Service會使用已有的 NAT Gateway；如果 VPC 中沒有 NAT Gateway，系統會預設自動為您建立一個 NAT Gateway。如果您不希望系統自動建立 NAT Gateway，可以取消勾選頁面下方的為專用網路配置 **SNAT**。

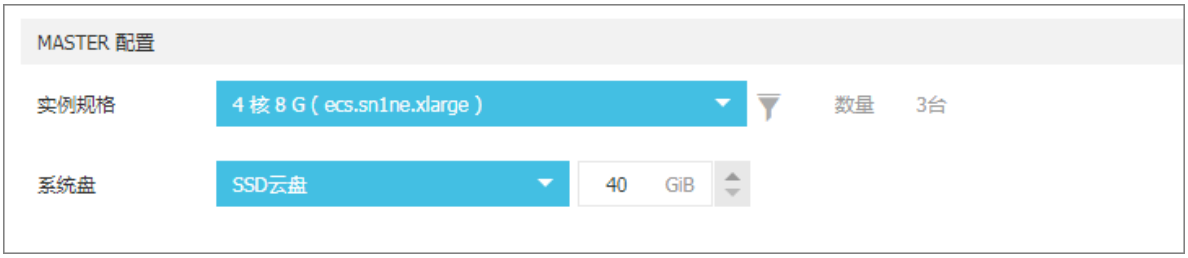
> 说明：
> 若選擇不自動建立 NAT Gateway，您需要自行配置 NAT Gateway實現 VPC 安全訪問公網環境，或者手動設定 SNAT，否則 VPC 內執行個體將不能正常訪問公網，會導致叢集建立失敗。



7. 設定節點類型，Container Service支援隨用隨付和訂用帳戶兩種節點類型。
8. 設定 Master 節點的配置資訊。

   您需要選擇 Master 節點的系列和規格。

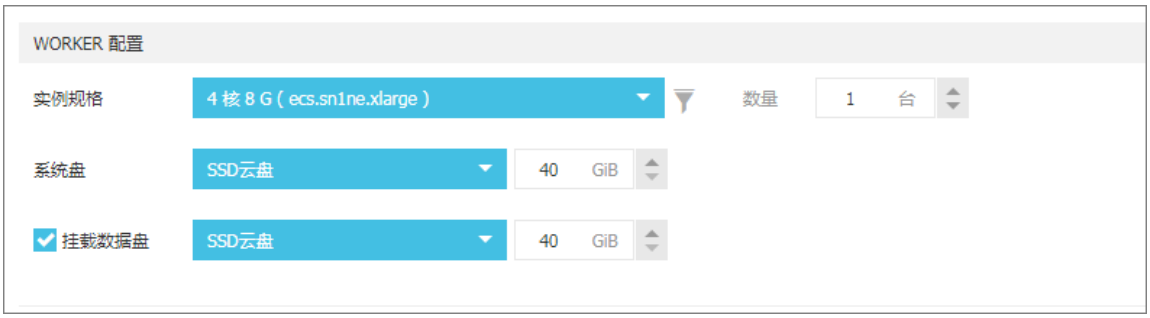
> 说明：
> - 目前僅支援 CentOS 作業系統。
> - 目前僅支援建立 3 個 Master 節點。
> - 支援為Master節點掛載系統硬碟，支援SSD雲端硬碟和高效雲端硬碟。



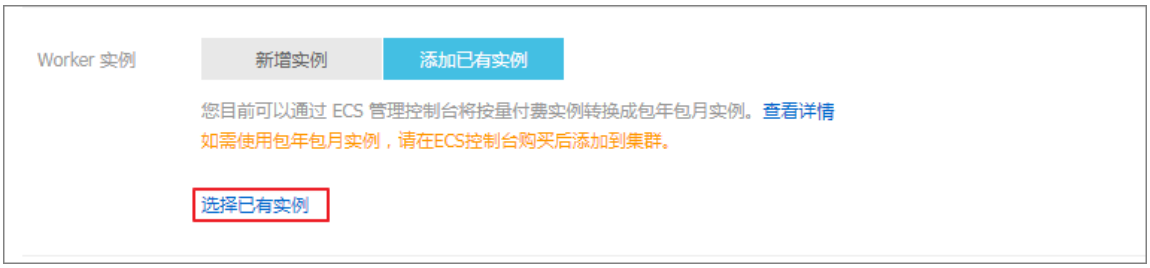
9. 設定 Worker 節點的配置資訊。您可選擇新增執行個體或添加已有執行個體。

> 说明：
> - 目前僅支援 CentOS 作業系統。
> - 每個叢集最多可包含 37 個 Worker 節點。如果您需要建立更多的節點，請提交工單申請。
> - 支援為Worker節點掛載一個資料盤，支援SSD雲端硬碟、高效雲端硬碟和普通雲端硬碟

   a. 若您選擇新增執行個體，則需要選擇 Worker 節點的系列和規格，以及需要建立的 Worker 節點的數量（本樣本建立 1 個 Worker 節點）。



b. 若您選擇添加已有執行個體，則需要預先在此地區下建立 ECS 雲端服務器。



10.配置登入方式。

- 設定密鑰。

  您需要在建立叢集的時候選擇金鑰組登入方式，單擊建立金鑰組，跳轉到ECS雲端服務器控制台，建立金鑰組，參見*## SSH ###*。金鑰組建立完畢後，設定該金鑰組作為登入叢集的憑據。

- 設定密碼。
  - 登入密碼：設定節點的登入密碼。
  - 確認密碼：確認設定的節點登入密碼。

11.設定Pod網路 CIDR 和Service CIDR。

> 说明：
> 該選項僅在選擇使用已有VPC時出現。

您需要指定**Pod 網路 CIDR**和**Service CIDR**，兩者都不能與 VPC 及 VPC 內已有 Kubernetes 叢集使用的網段重複，建立成功後不能修改。而且 Service 位址區段也不能和 Pod 位址區段重複，有關 kubernetes 網路位址區段規劃的資訊，請參考*VPC# Kubernetes #########*。

12.設定是否為專用網路配置 SNAT Gateway。

> 说明：
> 若您選擇自動建立 VPC 時必須配置 SNAT；若您選擇使用已有VPC，可選擇是否自動設定 SNAT Gateway。若選擇不自動設定 SNAT，您可自行配置NAT Gateway實現 VPC 安全訪問公網環境；或者手動設定 SNAT，否則 VPC 內執行個體將不能正常訪問公網，會導致叢集建立失敗。

13.設定是否開放公網 SSH 登入。

- 選擇開放公網 SSH 登入，您可以 SSH 訪問叢集。
- 選擇不開放公網 SSH 登入，將無法通過 SSH 訪問叢集，也無法通過 kubectl 串連 叢集。如果您需要通過 SSH 訪問叢集執行個體，可以手動為 ECS 執行個體綁定 EIP，並配置安全性群組規則，開放 SSH（22）連接埠，具體操作參見*SSH##Kubernetes##*。

14.設定是否啟用Cloud Monitor外掛程式。

您可以選擇在 ECS 節點上安裝Cloud Monitor外掛程式，從而在Cloud Monitor控制台查看所建立 ECS 執行個體的監控資訊。

15.設定是否啟用Log Service，您可使用已有Project或建立一個Project。

勾選使用**SLS**，會在叢集中自動設定Log Service外掛程式。建立應用時，您可通過簡單配置，快速使用Log Service，詳情參見*Kubernetes#Log Service#####*。

16.是否啟用進階選項。

a. 設定啟用的網路外掛程式，支援Flannel和Terway網路外掛程式。

- Flannel：簡單穩定的社區的Flannel cni外掛程式。
- Terway：阿里雲Container Service自研的網路外掛程式，支援將阿里雲的彈性網卡分配給容器，支援Kubernetes的**`NetworkPolicy`**來定義容器間的存取原則，支援對單個容器做頻寬的限流，目前處於公測階段。

b. 設定節點 Pod 數量，是指單個節點可運行 Pod 數量的上限，建議保持預設值。

c. 設定是否選擇自訂鏡像。或不選擇自訂鏡像，則 ECS 執行個體會安裝預設的 CentOS 版本。

目前您只能選擇基於 CentOS 的鏡像來快速部署您需要的環境，如基於 CentOS 7.4 的 LAMP 部署測試的鏡像。

d. 設定是否使用自訂叢集**CA**。如果勾選自訂叢集 CA，可以將 CA 憑證添加到 kubernetes 叢集中，加強服務端和用戶端之間資訊互動的安全性。

17.單擊建立叢集，啟動部署。

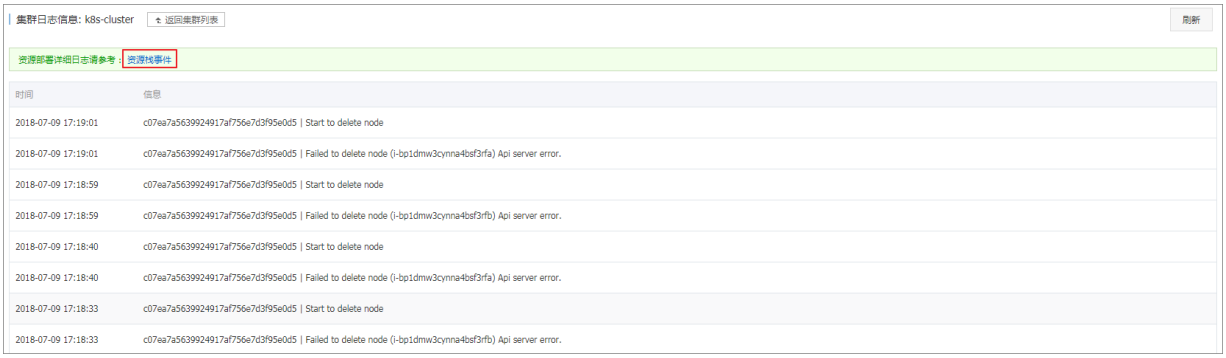
说明：

一個包含多節點的 Kubernetes 叢集的建立時間一般需要十幾分鐘。

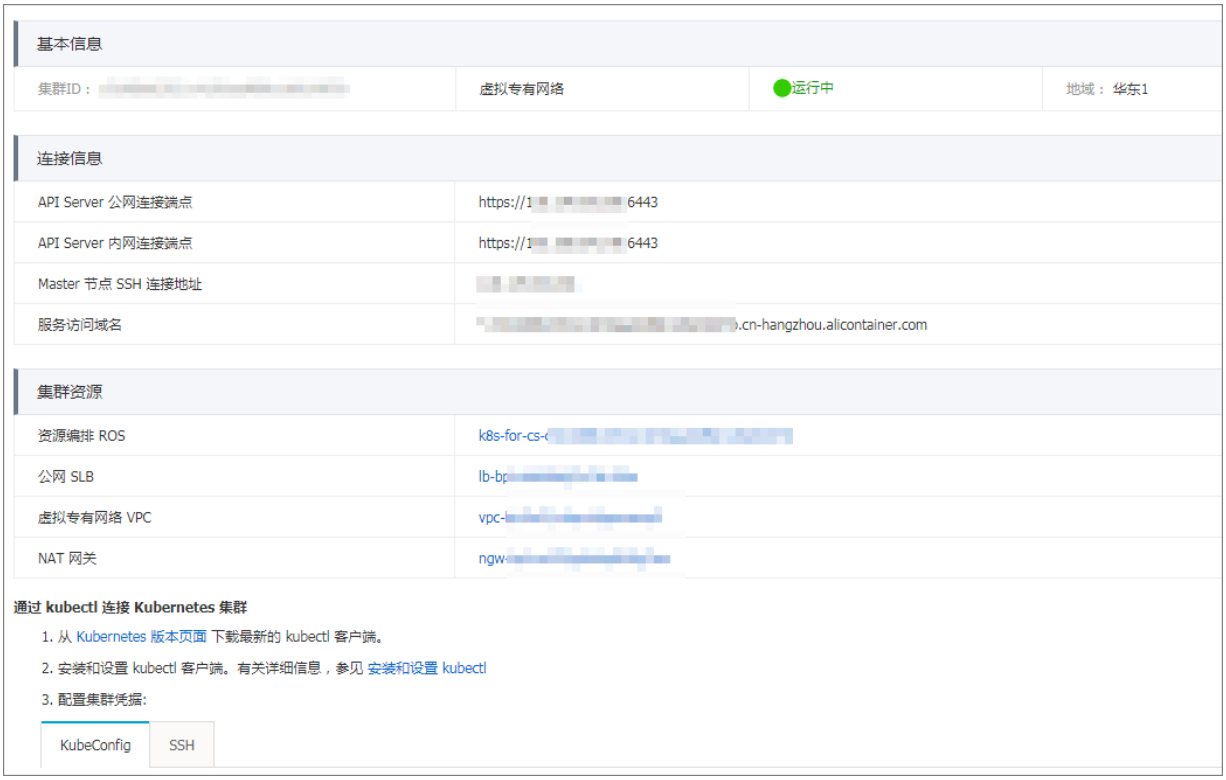
查看叢集部署結果

叢集建立成功後，您可以在Container Service管理主控台的 Kubernetes 叢集列表頁面查看所建立的叢集。

您可以單擊右側的查看日誌查看叢集的日誌資訊，您可單擊資源棧事件查看更詳細的資訊。



您可以單擊右側的管理，查看叢集的基本資料和串連資訊。



其中：

- **API Server** 公網串連端點：Kubernetes 的 API server 對公網提供服務的地址和連接埠，可以通過此服務在使用者終端使用 kubectl 等工具管理叢集。

- **API Server** 內網串連端點：Kubernetes 的 API server 對叢集內部提供服務的地址和連接埠。此 IP 為負載平衡的地址，後端有 3 台 Master 提供服務。
- **Master** 節點 **SSH** 串連地址：可以直接通過 SSH 登入到 Master 節點，以便對叢集進行日常維護。
- 服務訪問網域名稱：為叢集中的服務提供測試用的訪問網域名稱。服務訪問網域名稱尾碼是<cluster_id>.<region_id>.alicontainer.com。

例如，您可以通過 SSH 登入到 Master 節點，執行`kubectl get node`查看叢集的節點資訊。

```
login as: root
root@1[redacted]3's password:

Welcome to Alibaba Cloud Elastic Compute Service !

[root@iZbp1d7yvpa3j183u0urllZ ~]# kubectl get node
NAME                                STATUS    ROLES     AGE       VERSION
cn-hangzhou.i-[redacted]            Ready     <none>    17m       v1.8.4
cn-hangzhou.i-[redacted]            Ready     master    19m       v1.8.4
cn-hangzhou.i-[redacted]            Ready     master    24m       v1.8.4
cn-hangzhou.i-[redacted]            Ready     master    22m       v1.8.4
[root@iZbp1d7yvpa3j183u0urllZ ~]#
```

可以發現，一共有 4 個節點，包括 3 個 Master 節點和我們在參數設定步驟填寫的 1 個 Worker 節點。

1.3.3 VPC下 Kubernetes 的網路位址區段規劃

在阿里雲上建立 Kubernetes 叢集時，通常情況下，可以選擇自動建立專用網路，使用預設的網路地址即可。在某些複雜的情境下，需要您自主規劃 ECS 地址、Kubernetes Pod 地址和 Service 地址。本文將介紹阿里雲 VPC 環境下 Kubernetes 裡各種地址的作用，以及位址區段該如何規劃。

Kubernetes 網段基本概念

首先來看幾個和 IP 位址有關的概念。

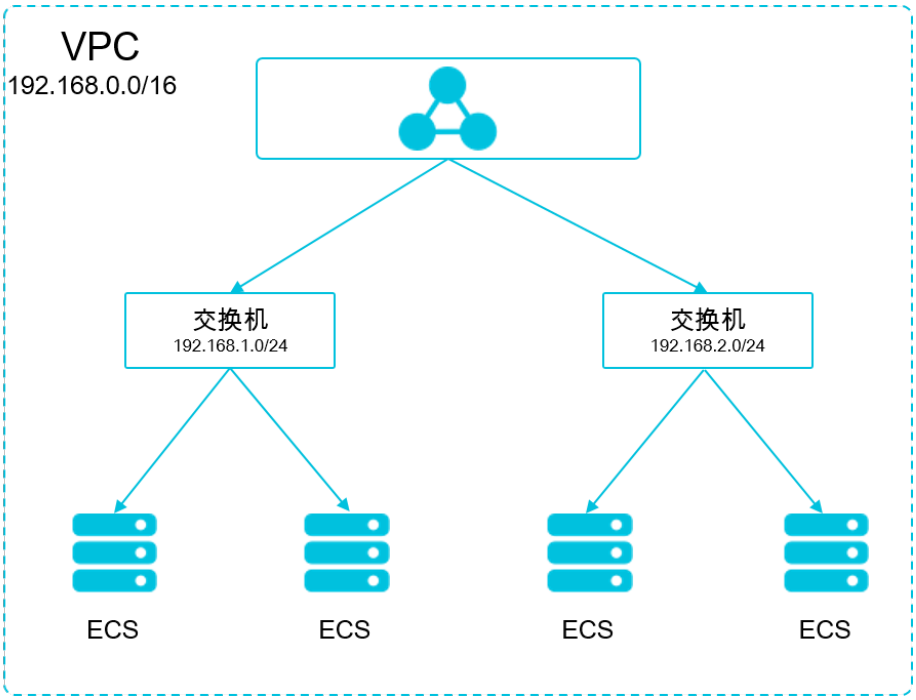
VPC 網段

您在建立 VPC 選擇的位址區段。只能從 10.0.0.0/8、172.16.0.0/12、192.168.0.0/16 三者當中選擇一個。

交換器網段

在 VPC 裡建立交換器時指定的網段，必須是當前 VPC 網段的子集（可以跟 VPC 網段地址一樣，但不能超過）。交換器下面的 ECS 所分配到的地址，就是從這個交換器位址區段內擷取的。一個 VPC 下，可以建立多個交換器，但交換器網段不能重疊。

VPC 網段結構如下圖。



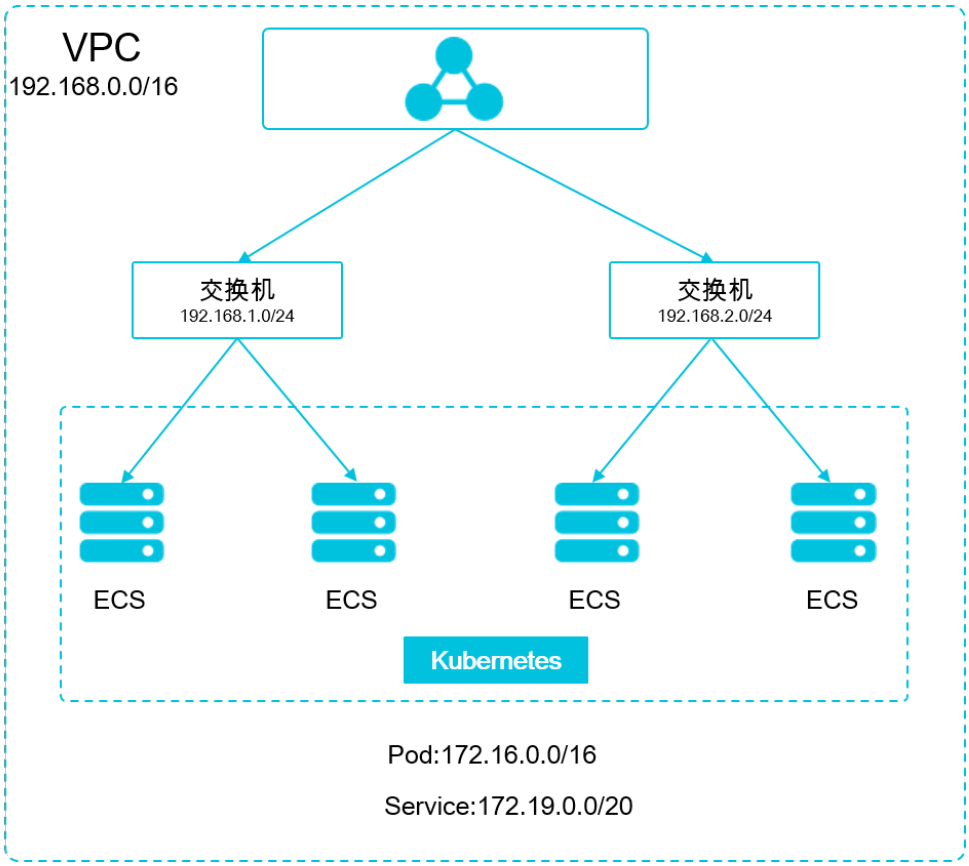
Pod 位址區段

Pod 是 Kubernetes 內的概念，每個 Pod 具有一個 IP 位址。在阿里雲Container Service上建立 Kubernetes 叢集時，可以指定Pod 的位址區段，不能和 VPC 網段重疊。比如 VPC 網段用的是 172.16.0.0/12，Kubernetes 的 Pod 網段就不能使用 172.16.0.0/16，不能使用 172.17.0.0/16…，這些地址都涵蓋在 172.16.0.0/12 裡了。

Service 位址區段

Service 也是 Kubernetes 內的概念，每個 Service 有自己的地址。同樣，Service 位址區段也不能和 VPC 位址區段重合，而且 Service 位址區段也不能和 Pod 位址區段重合。Service 地址只在 Kubernetes 叢集內使用，不能在叢集外使用。

Kubernetes 網段和 VPC 網段關係如下圖。



如何選擇位址區段

單 VPC+單 Kubernetes 叢集情境

這是最簡單的情形。VPC 地址在建立 VPC 的時候就已經確定，建立 Kubernetes 叢集時，選擇和當前 VPC 不一樣的位址區段就可以了。

單 VPC+多 Kubernetes 叢集情境

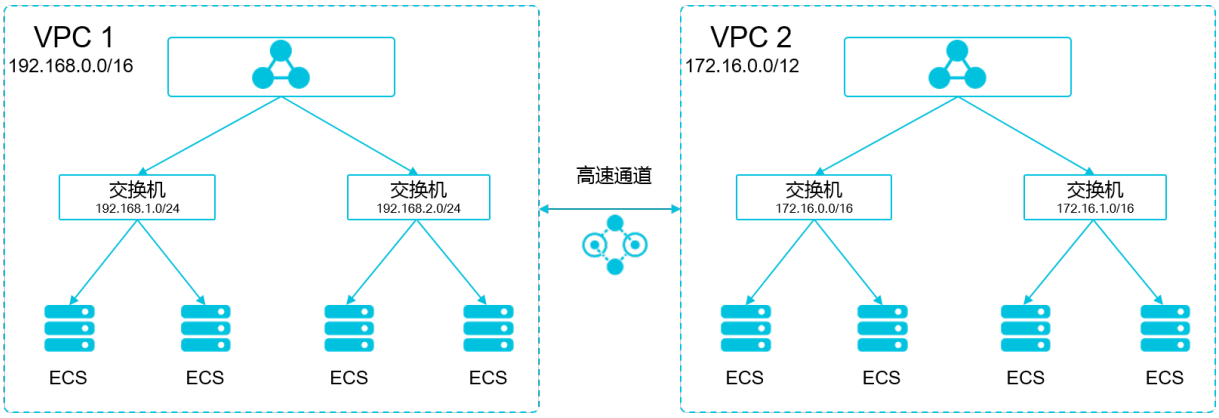
一個 VPC 下建立多個 Kubernetes 叢集。在預設的網路模式下（Flannel），Pod 的報文需要通過 VPC 路由轉寄，Container Service會自動在 VPC 路由上配置到每個 Pod 位址區段的路由表。所有 Kubernetes 叢集的 Pod 位址區段不能重疊，但 Service 位址區段可以重疊。

VPC 地址還是建立 VPC 的時候確定的，建立 Kubernetes 的時候，為每個 Kubernetes 叢集選擇一個不重疊的位址區段，不僅不能和 VPC 地址重疊，也不和其他 Kubernetes Pod 段重疊。

需要注意的是，這種情況下 Kubernetes 叢集部分互連，一個叢集的 Pod 可以直接存取另外一個叢集的 Pod 和 ECS，但不能訪問另外一個叢集的 Service。

VPC 互聯情境

兩個 VPC 網路互聯的情況下，可以通過路由表配置哪些報文要發送到對端 VPC 裡。以下面的情境為例，VPC 1 使用位址區段 192.168.0.0/16，VPC 2 使用位址區段 172.16.0.0/12，我們可以通過路由表，指定在 VPC 1 裡把 目的地址為 172.16.0.0/12 的報文都發送到 VPC 2。



在這種情況下，VPC 1 裡建立的 Kubernetes 叢集，首先不能和 VPC 1 的位址區段重疊，同時其位址區段也不能和 VPC 2 的位址區段重疊。在 VPC 2 上建立 Kubernetes 叢集也類似。這個例子中，Kubernetes 叢集 Pod 位址區段可以選擇 10.0.0.0/8 下的某個子段。

> 说明：
> 這裡要特別關注“路由到 VPC 2” 的位址區段，可以把這部分地址理解成已經佔用的地址，Kubernetes叢集不能和已經佔用的地址重疊。

如果 VPC 2 裡要訪問 VPC 1 的 Kubernetes Pod，則需要在 VPC 2 裡配置到 VPC1 Kubernetes叢集pod地址的路由。

VPC 網路到 IDC 的情境

和 VPC 互聯情境類似，同樣存在 VPC 裡部分位址區段路由到 IDC ，Kubernetes 叢集的 Pod 地址就不能和這部分地址重疊。IDC 裡如果需要訪問 Kubernetes 裡的 Pod 地址，同樣需要在 IDC 端配置到專線 VBR 的路由表。

1.3.4 通過 kubectl 串連 Kubernetes 叢集

如果您需要從用戶端電腦串連到 Kubernetes 叢集，請使用 Kubernetes 命令列用戶端 *kubectl*。

操作步驟

1. 從*Kubernetes ####* 下載最新的 kubectl 用戶端。
2. 安裝和設定 kubectl 用戶端。

   有關詳細資料，參見 *##### kubectl*。
3. 配置叢集憑據。

您可以使用**scp**命令安全地將主節點的配置從 Kubernetes 叢集主 VM 中的 `/etc/kubernetes/kube.conf` 複製到本機電腦的 `$HOME/.kube/config`（`kubectl` 預期憑據所在的位置）。

- 如果您在建立叢集時選擇密碼方式登入，請用以下方式拷貝 kubectl 設定檔。

```
mkdir $HOME/.kube
scp root@<master-public-ip>:/etc/kubernetes/kube.conf $HOME/.kube/config
```

- 如果您在建立叢集時選擇金鑰組方式登入，請用以下方式拷貝 kubectl 設定檔。

```
mkdir $HOME/.kube
scp -i [.pem 私密金鑰檔案在本地機器上的儲存路徑] root@:/etc/kubernetes/kube.conf $HOME/.kube/config
```

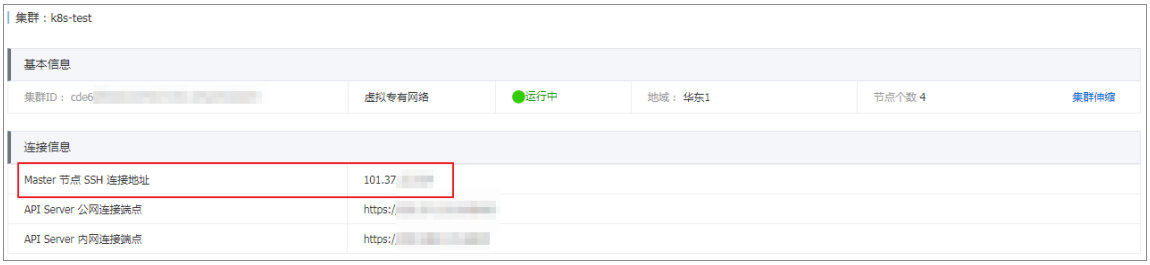
您可以在叢集資訊頁面查看叢集的 `master-public-ip`。

a) 登入*Container Service#####*。

b) 單擊**Kubernetes**進入 Kubernetes 叢集列表頁面。

c) 選擇所需的叢集並單擊右側的管理

您可以在串連資訊處查看叢集的串連地址。

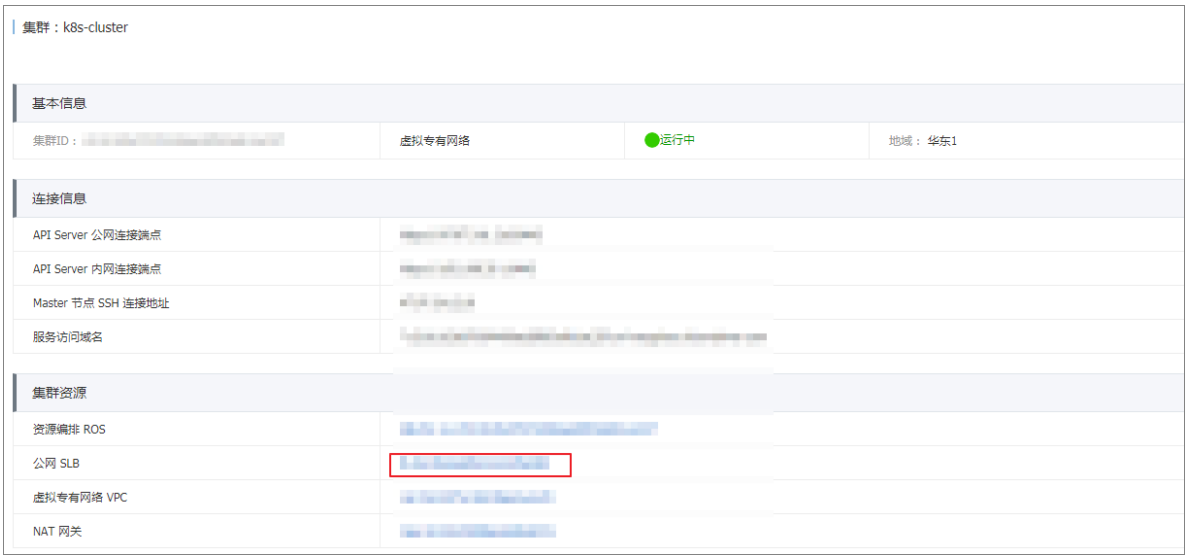


1.3.5 SSH訪問Kubernetes叢集

如果您在建立叢集時，選擇不開放公網 SSH 訪問，您將無法 SSH 訪問 Kubernetes 叢集，而且也無法通過 kubectl 串連 Kubernetes 叢集。如果建立叢集後，您需要 SSH 訪問您的叢集，可以手動為 ECS 執行個體綁定 EIP，並配置安全性群組規則，開放 SSH（22）連接埠。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的管理。
4. 在叢集資源中，找到您的公網Server Load Balancer執行個體，並單擊執行個體 ID，頁面跳轉至您的公網Server Load Balancer執行個體的詳細資料頁面。



5. 單擊左側導覽列中的執行個體 > 執行個體管理，單擊添加監聽。
6. 添加 SSH 監聽規則。
   a. 前端協議（連接埠）選擇 TCP：22。
   b. 後端協議（連接埠）選擇 TCP：22。
   c. 選擇 使用伺服器組 並選擇虛擬伺服器組。
   d. 伺服器組ID 選擇**sshVirtualGroup**。
   e. 單擊下一步 並單擊確認，建立監聽。

7. 監聽建立完成後，您就可以使用該負載平衡的服務地址 SSH 訪問您的叢集了。

1.3.6 SSH金鑰組訪問Kubernetes叢集

阿里雲Container Service支援通過 SSH 金鑰對的方式登入叢集，保障遠程SSH訪問的安全性。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 單擊右上方的建立 **Kubernetes** 叢集。

4. 對叢集登入方式進行配置，設定通過密鑰的登入方式。對叢集其他的參數進行配置，參見*## Kubernetes##*，最後單擊建立。
   1. 若您已在ECS雲端服務器控制台建立了密鑰，只需在金鑰組下拉框中進行選擇。
   2. 若您沒有密鑰，單擊建立金鑰組，前往ECS雲端服務器控制台建立金鑰組，參見*## SSH ##*。

5. 叢集建立成功後，在叢集列表中找到該叢集，單擊叢集右側的管理，在串連資訊中查看**Master**節點**SSH**串連地址。

6. 下載 .*pem*私密金鑰檔案，按照本地作業系統環境，如 Windows 和 linux 作業系統，進行相關配置，參見*##SSH#####Linux####*。以 linux 環境為例。

   a) 找到您所下載的.pem私密金鑰檔案在本地機上的儲存路徑，如 `/root/xxx.pem`

   b) 運行命令修改私密金鑰檔案的屬性：`chmod 400 [.pem私密金鑰檔案在本地機上的儲存路徑]`，如 `chmod 400 /root/xxx.pem`。

   c) 運行命令串連至叢集：`` `ssh -i [.pem私密金鑰檔案在本地機上的儲存路徑] root@[master-public-ip]`，其中 master-public-ip 即是 Master 節點 SSH 串連地址。如 `ssh -i /root/xxx.pem root@10.10.10.100`。

1.3.7 升級叢集

您可以通過Container Service管理主控台升級您叢集的 Kubernetes 版本。

您可以在 Kubernetes 叢集列表頁面查看您的叢集的 Kubernetes 版本。

注意事項

- 叢集升級需要機器可以公網訪問，以便下載升級所需的軟體包。
- 叢集升級 Kubernetes 過程中，可能會有升級失敗的情況，為了您的資料安全，強烈建議您先打快照然後再升級。有關 ECS 打快照的操作參見####。
- 叢集升級 Kubernetes 過程中，叢集上部署中的服務會中斷，同時無法進行叢集和應用的操作，請您在升級之前安排好相關事宜。升級時間大約 5-30 分鐘， 升級完成後叢集會變成運行中狀態。

準備工作

請在叢集升級前檢查叢集的健康情況，並且確保叢集健康。

登入 Master 節點，參見*SSH##Kubernetes##*和*## kubectl ## Kubernetes ##*。

1. 執行`kubectl get cs`命令，確保所有模組都處於健康狀態。

```
NAME                 STATUS    MESSAGE              ERROR
 scheduler           Healthy   ok
 controller-manager  Healthy   ok
 etcd-0              Healthy   {"health": "true"}
 etcd-1              Healthy   {"health": "true"}
 etcd-2              Healthy   {"health": "true"}
```

2. 執行 `kubectl get nodes` 命令，確保所有節點都處於 Ready 狀態。

```
kubectl get nodes
 NAME                                  STATUS    ROLES     AGE
VERSION
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
```

如果節點不正常可以自行修複，也可以通過提交工單，請阿里雲工程師協助修複。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集，並單擊更多 > 叢集升級 。

4. 在彈出的對話方塊中，單擊升級。

   系統開始升級 Kubernetes 的版本。

升級完成後，您可以在 Kubernetes 叢集列表頁面查看叢集 Kubernetes 的版本，確認升級成功。

1.3.8 擴容和縮容叢集

通過Container Service管理主控台，您可以根據實際業務需要對 Kubernetes 叢集的 Worker 節點進行擴容和縮容。

背景信息

- 目前不支援叢集中 Master 節點的擴容和縮容。
- 叢集縮容只能縮減叢集建立和擴容時增加的 Worker 節點，通過######功能添加到叢集中的 Worker 節點不能被縮減。
- 縮減規則是按建立時間進行的，最新擴容出來的節點會被先回收。
- 必須有大於1個非手動添加的節點，才能進行縮容。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的叢集伸縮。

4. 選擇擴容 或者縮容，設定 Worker 節點的數量。

   本樣本進行叢集擴容，將叢集的 Worker 節點數由 1 個擴容到 4 個。

5. 填寫節點的登入密碼。

> 说明：
> 由於升級過程依賴登入到 ECS 來拷貝配置資訊，所以叢集伸縮時填寫的密碼必須和建立叢集時填寫的密碼一致。

6. 單擊提交。

后续操作

伸縮完成後，單擊左側導覽列中的叢集 > 節點，查看節點列表頁面，您可以看到 Worker 節點的數量從 1 變成了 4。

1.3.9 自動調整叢集

根據叢集負載的工作情況，自動調整叢集。

前提條件

您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。

背景資訊

自動調整叢集與傳統基於資源閾值的*#######*不同，根據配置好的伸縮條件，當叢集工作負載的工作情況達到配置條件時，叢集可進行自動調整。

操作步驟

執行自動調整

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入Kubernetes叢集列表頁面。
3. 選擇所需的叢集並單擊操作列的更多 > 自動調整。

授權

- 開通ESS服務

1. 單擊彈出對話方塊中的第一個連結，進入Auto Scaling服務 ESS頁面。

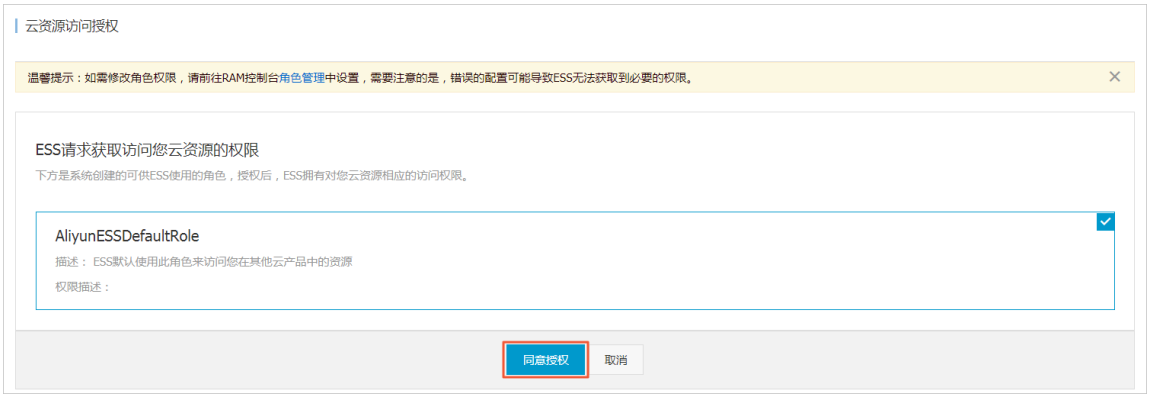
2. 單擊開通ESS服務，進入雲產品開通頁。

3. 選中我已閱讀並同意複選框，單擊立即開通。

4. 開通成功後，在開通完成頁簽，單擊管理主控台，進入**Auto Scaling**服務 **ESS**頁面。

5. 單擊前往授權，進入雲資源訪問授權頁面，配置對雲資源的存取權限。

6. 單擊同意授權。



預期結果

頁面自動跳轉至Auto Scaling控制台，說明授權成功。關閉頁面，繼續配置授權角色。

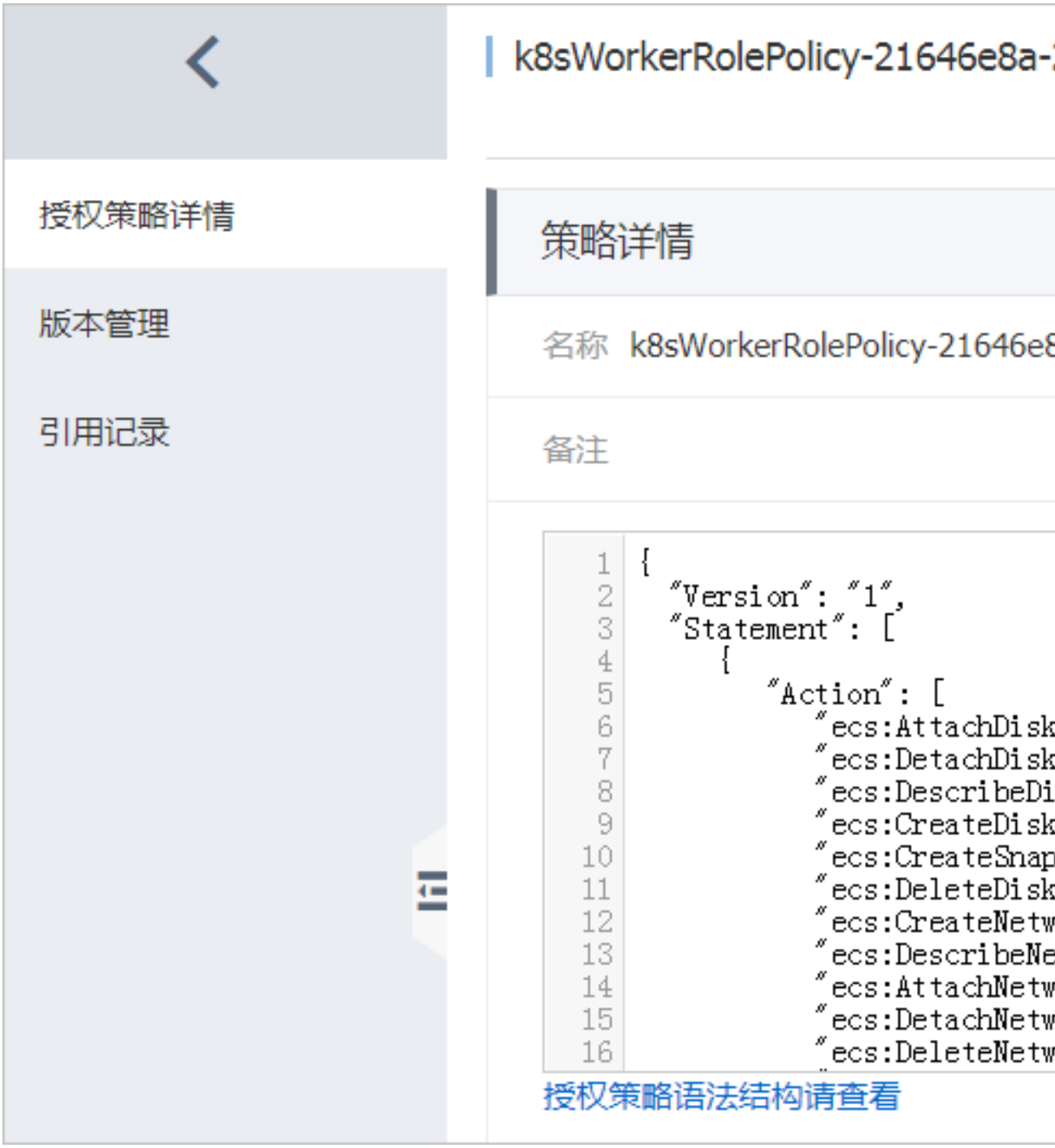
- 授權角色

1. 單擊彈出對話方塊中的第二個連結，進入角色授權策略頁面。

> 说明：
> 此處需要以主帳號登入控制台。

2. 選擇目標授權策略名稱並單擊操作列查看許可權，進入授權策略詳情頁面。

3. 單擊頁面右上方修改授權策略。



4. 在策略內容的`Action`欄位，補充以下策略：

```
"ess:Describe*",
"ess:CreateScalingRule",
"ess:ModifyScalingGroup",
"ess:RemoveInstances",
"ess:ExecuteScalingRule",
"ess:ModifyScalingRule",
"ess:DeleteScalingRule",
"ecs:DescribeInstanceTypes",
```

```
"ess:DetachInstances"
```

> 说明：
> 需要在Action欄位的最後一行補充“，”，再添加以上內容。

5. 單擊修改策略。

配置自動調整

1. 在**Auto Scaling**配置頁面，填寫以下資訊，配置自動縮容：

配置	說明
叢集	目的地組群名稱。
縮容閾值	叢集中負載申請的資源量與叢集資源量的比值。當叢集中負載申請的資源值小於等於配置的縮容閾值時，叢集會自動縮容。預設情況下，縮容閾值為50%。
縮容觸發時延	叢集滿足配置的縮容閾值時，在配置的縮容觸發時延到達後，叢集開始縮容。單位：分鐘。預設情況下是10分鐘。
靜默時間	在叢集添加或刪除節點後，在靜默時間內，叢集不會再次觸發擴容或縮容，單位 ：分鐘。預設情況下是10分鐘。

2. 根據所需要Auto Scaling的資源類型（CPU/GPU），單擊操作列建立。

在伸縮配置頁面填寫以下資訊，建立對應的伸縮組：

配置	說明
地區	所建立伸縮組將要部署到的地區。與伸縮組所在叢集的地區一致，不可變更。
可用性區域	所建立伸縮組的可用性區域。
專用網路	所建立伸縮組的網路，與伸縮組所在叢集的網路一致。

配置 Worker 節點的資訊。

配置	說明
執行個體規格	伸縮組內執行個體的規格。
系統硬碟	伸縮組的系統硬碟。
掛載資料盤	是否在建立伸縮組時掛載資料盤，預設情況下不掛載。
執行個體數	伸縮組所包含的執行個體數量。 说明： • 執行個體不包含客戶已有的執行個體。 • 預設情況，執行個體的最小值是0台，超過0台的時候，叢集會預設向伸縮組中添加執行個體，並將執行個體加入到伸縮組對應的Kubernetes叢集中。
金鑰組	登入伸縮後的節點時所使用的金鑰組。可以在ECS控制台建立金鑰組。 说明： 目前只支援通行金鑰對方式登入。
RDS白名單	Auto Scaling後的節點可以訪問的RDS執行個體。

3. 單擊確定，建立伸縮組。

預期結果

- 在自動調整頁面，可以看到CPU下已建立好一個伸縮組。

- 1. 單擊左側導覽列應用 > 部署，進入部署列表頁面。
  2. 選擇目的地組群和kube-system命名空間，可以看到名稱為cluster-autoscaler的組件已建立成功。

1.3.10 刪除叢集

您可以通過Container Service管理主控台刪除不再使用的叢集。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的更多 > 刪除。

后续操作

刪除叢集失敗

如果您在 ROS 建立的資源下手動添加了一些資源，ROS 是沒有許可權刪除這些資源的。比如在 ROS 建立的 VPC 下手動添加了一個 VSwitch，這樣在就會導致 ROS 刪除時無法處理該 VPC，從而最終刪除叢集失敗。

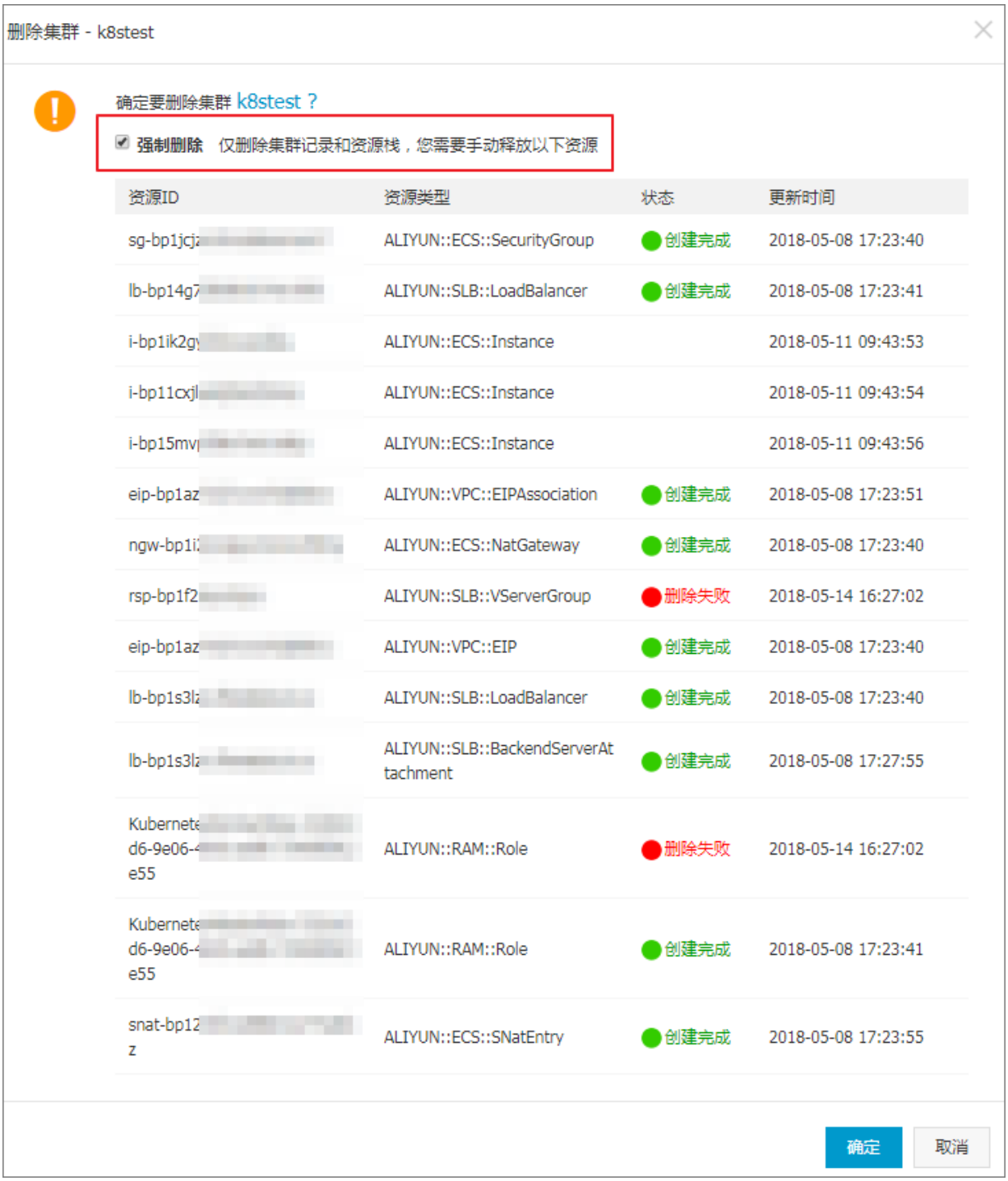
Container Service提供了強制移除叢集的功能。通過強制移除功能，您可以在叢集刪除失敗後，強制移除叢集記錄和 ROS 資源棧。但是，強制移除操作不會自動釋放您手動建立的這些資源，您需要手動進行釋放。

叢集刪除失敗時，會顯示如下資訊。

單擊刪除失敗的叢集對應的更多 > 刪除，在彈出的對話方塊裡，您可以看到刪除失敗的資源，勾選強制移除 並單擊 確定，即可刪除叢集和 ROS 資源棧。

說明：

該操作不會釋放這些資源，您需要手動釋放資源。有關如何如果排查不能釋放的資源，可參見*## Kubernetes #####ROS stack ####*。



1.4 節點管理

1.4.1 添加已有節點

您可以向已經建立的 Kubernetes 叢集中添加已有的 ECS 執行個體。目前，僅支援添加 Worker 節點。

前提条件

- 如果之前沒有建立過叢集，您需要先*##Kubernetes##*。
- 需要先把待添加的 ECS 執行個體添加到 Kubernetes 叢集的安全性群組裡。

背景信息

- 預設情況下，每個叢集中最多可包含 40 個節點。如果您需要添加更多節點，請提交工單申請。
- 添加的雲端服務器必須與叢集在同一地區同一 VPC 下。
- 添加已有雲端服務器時，請確保您的雲端服務器有EIP（專用網路），或者相應 VPC 已經配置了 NAT Gateway。總之，需要確保相應節點能正常訪問公網，否則，添加雲端服務器會失敗。
- Container Service不支援添加不同帳號下的雲端服務器。
- 僅支援添加作業系統為 CentOS 的節點。

操作步驟

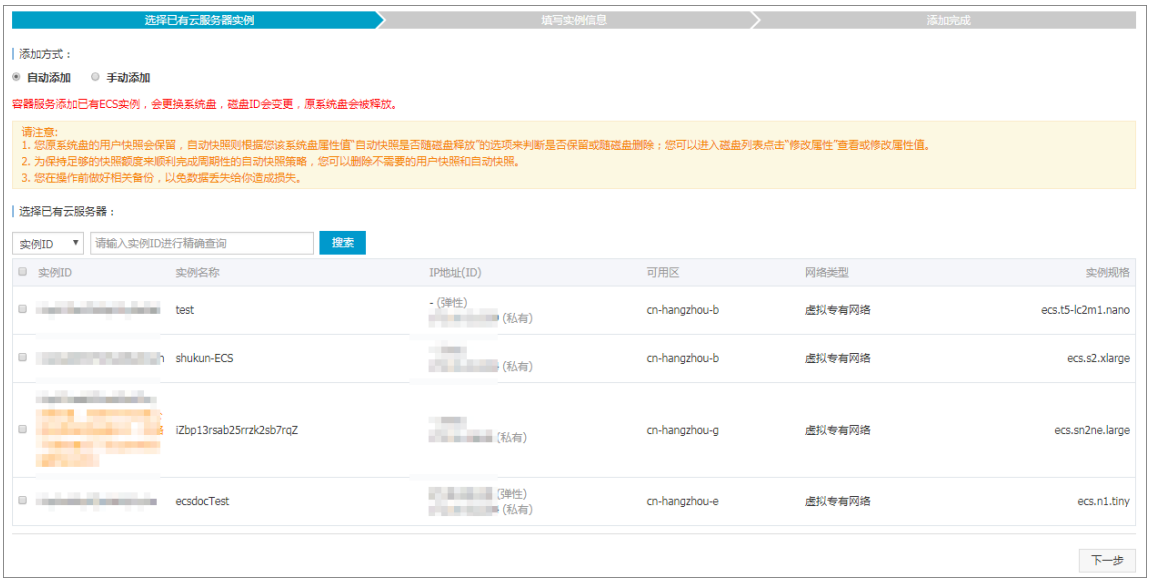
1. 登入 *Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的更多 > 添加已有節點。

   進入添加節點頁面，您可以選擇自動添加或手動添加的方式，添加現有雲端服務器執行個體。

   自動添加方式會列出當前帳號下可用的 ECS 雲端服務器，在 Web 介面進行選擇，安裝部署，並自動添加到叢集中；手動添加方式要求您擷取安裝命令，登入到對應 ECS 雲端服務器上進行安裝，每次只能添加一個 ECS 雲端服務器。

4. 您可選擇自動添加的方式，您可以一次性添加多個ECS雲端服務器。

   a) 在已有雲端服務器的列表中，選擇所需的ECS雲端服務器，然後單擊下一步。

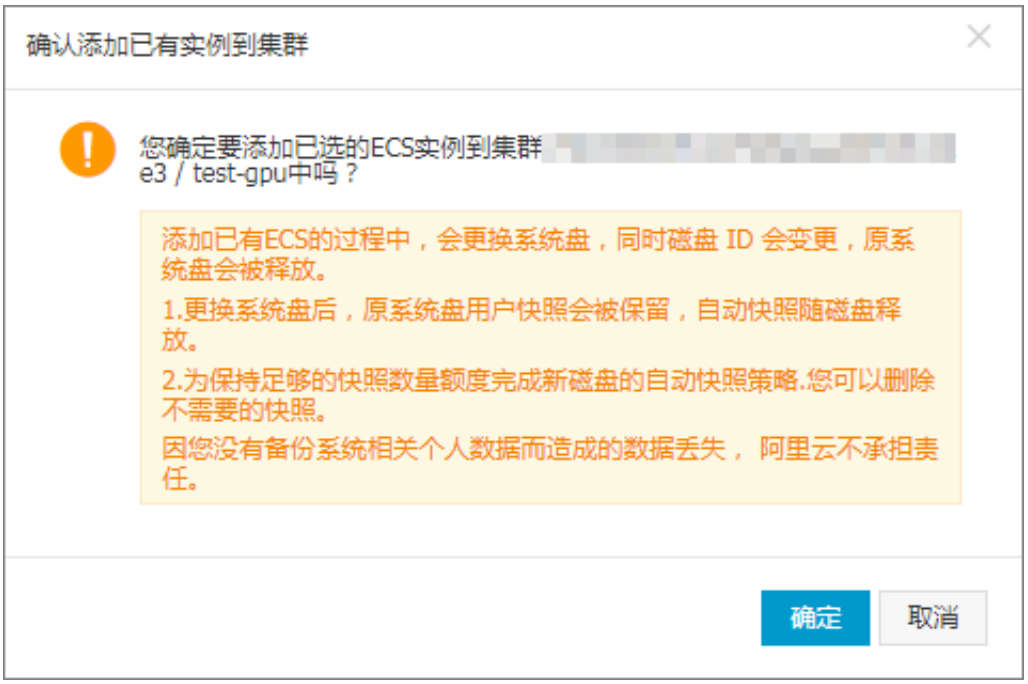


b) 填寫執行個體資訊，設定登入密碼，然後單擊下一步。

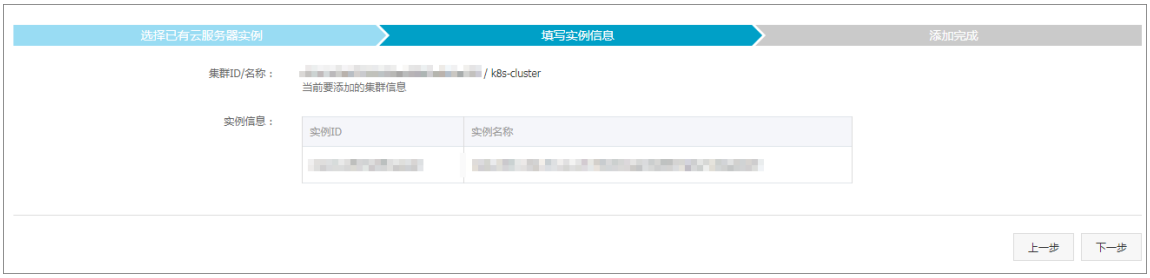
c) 在彈出的對話方塊中，單擊確定，選擇的 ECS 雲端服務器會自動添加到該叢集中。

5. （可选）選擇手動添加的方式。

a) 選擇所需的 ECS 雲端服務器，單擊下一步。您一次只能添加一個 ECS 雲端服務器。



b) 進入執行個體資訊頁面，確認無誤後，單擊下一步。

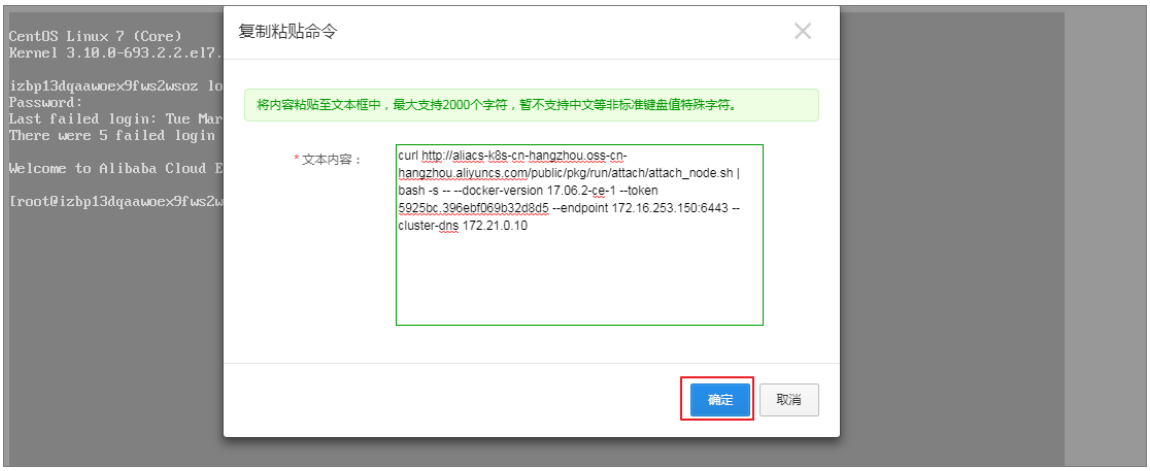


c) 進入添加節點頁面，您需要複製其中的執行命令。

d) 最後單擊完成。

e) 登入 *ECS #####*，單擊左側導覽列中的執行個體，選擇叢集所在的地區，選擇需要添加的 ECS 執行個體。

f) 單擊 ECS 執行個體右側的遠端連線。進入 ECS 執行個體遠端連線介面，根據頁面指導，輸入遠端連線密碼並單擊確定，成功後，輸入上面儲存的命令，單擊確定 ，開始執行指令碼。



g) 等待指令碼執行成功，該雲端服務器即添加成功。您可以在叢集列表頁面單擊叢集的 ID 查看該叢集下的節點列表。查看節點是否成功添加到叢集中。

1.4.2 查看節點列表

您可以通過命令、Container Service管理主控台的節點列表頁面或者 Kubernetes Dashboard 的節點列表頁面查看 Kubernetes 叢集的節點列表。

通過命令查看

> 说明：
> 使用命令查看 Kubernetes 叢集的節點列表頁面之前，您需要先設定*## kubectl ## Kubernetes ## #*。

通過 kubectl 串連到 Kubernetes 叢集後，運行以下命令查看叢集中的節點。

```
kubectl get nodes
```

輸出樣本：

```
$ kubectl get nodes
NAME                        STATUS    AGE       VERSION
iz2ze2n6ep53tch701yh9zz     Ready     19m       v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39e5az     Ready     7m        v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39e5bz     Ready     7m        v1.6.1-2+ed9e3d33a07093
iz2zef4dnn9nos8elyr32kz     Ready     14m       v1.6.1-2+ed9e3d33a07093
iz2zeitvvo8enoreufstkmz     Ready     11m       v1.6.1-2+ed9e3d33a07093
```

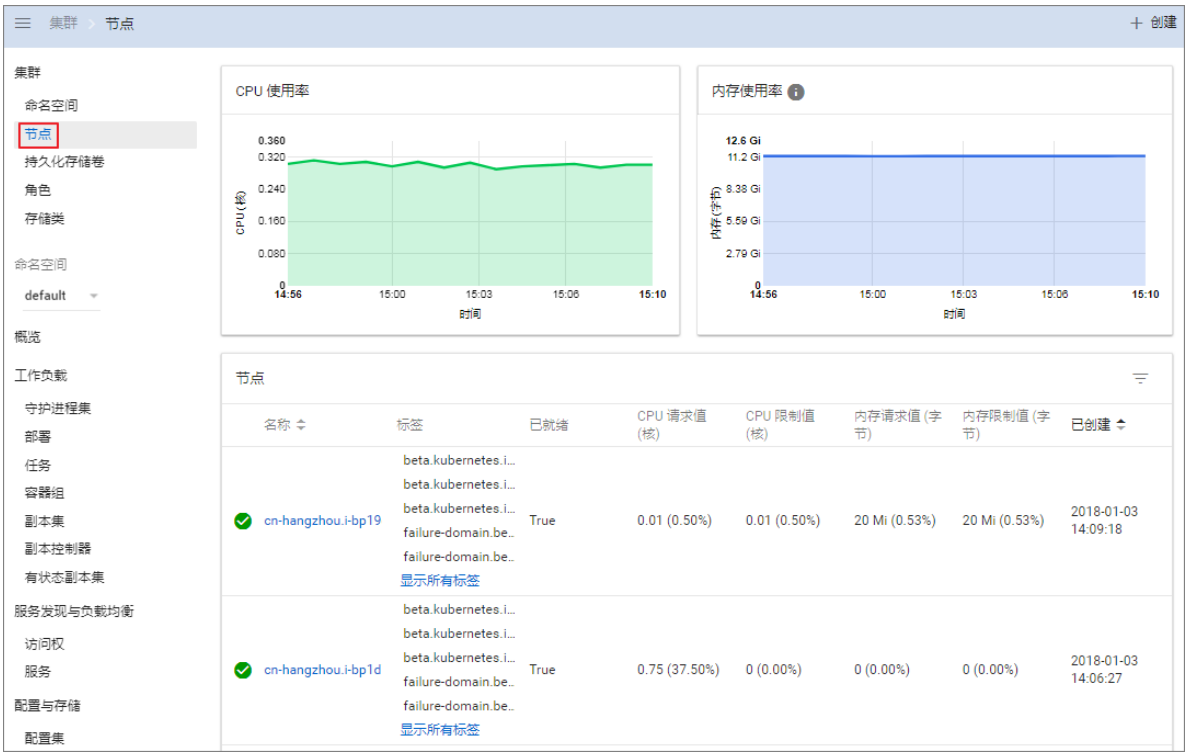
通過**Container Service**管理主控台查看

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點，進入節點列表頁面。
3. 選擇所需的叢集，您可以查看該叢集下的節點列表。

通過 Kubernetes Dashboard 查看

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的控制台，進入 Kubernetes Dashboard。

4. 在 Kubernetes Dashboard 中，單擊左側導覽列中的節點。即可查看叢集中的節點列表。



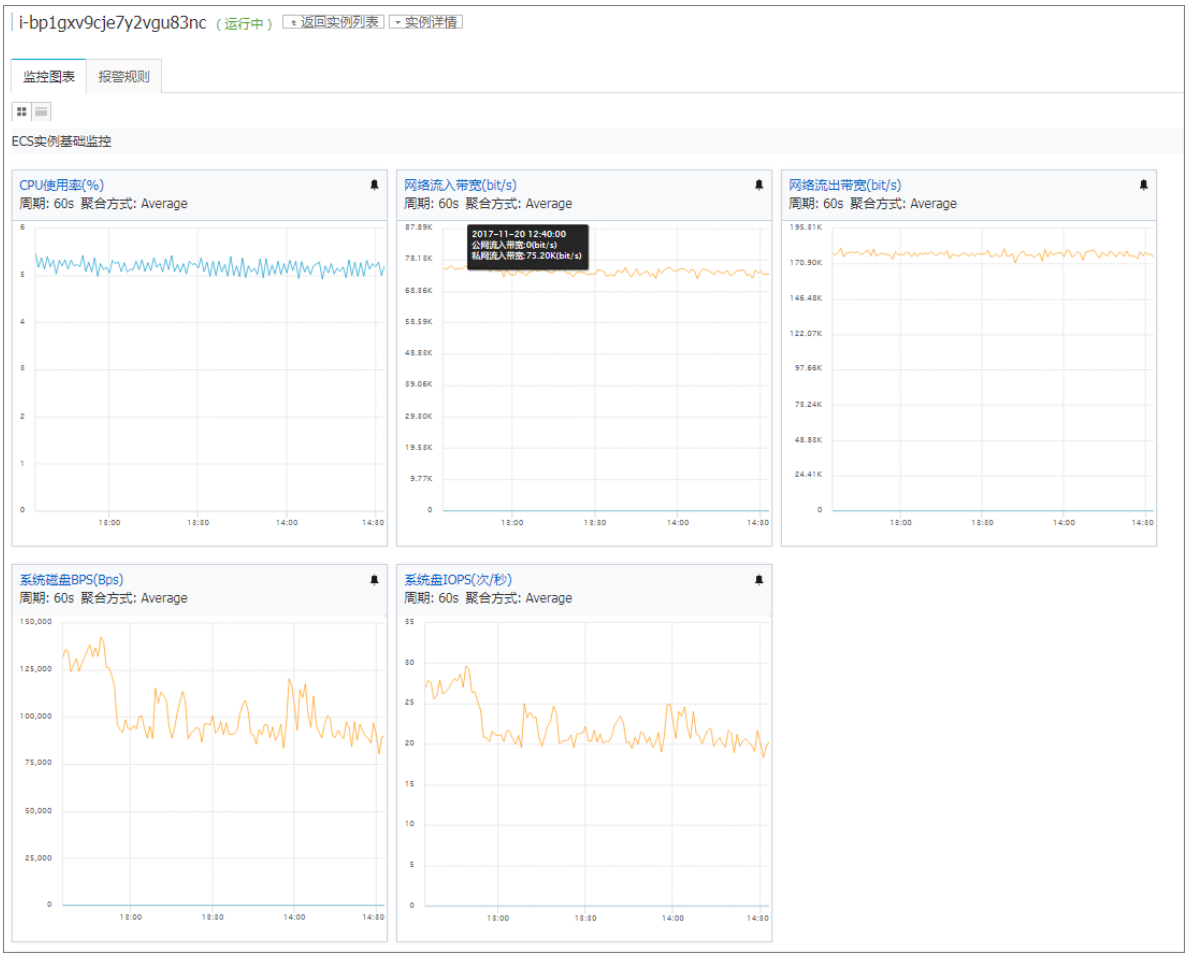
1.4.3 節點監控

kubernetes 叢集與阿里雲監控服務無縫整合，您可以查看 kubernetes 節點的監控資訊，瞭解Kubernetes叢集下 ECS 執行個體的節點監控指標。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點，進入節點列表頁面。
3. 選擇所需的叢集，在該叢集下選擇所需的節點。
4. 單擊監控，查看對應節點的監控資訊。

5. 進入Cloud Monitor管理主控台，查看對應 ECS 執行個體的基本監控資訊，包括 CPU使用率、網路流入頻寬、網路流出頻寬、系統磁碟 BPS、系統硬碟 IOPS 等指標。



后续操作

要想查看關於作業系統層級的監控指標，需要安裝Cloud Monitor組件。參見######。

Kubernetes 叢集新增了關於應用分組的監控功能，您可以參見##############進行升級。

1.4.4 節點標籤管理

您可以通過Container Service Web 介面對節點進列標籤管理，包括大量新增節點標籤、通過標籤篩選節點和快速刪除節點標籤。

關於如何使用節點標籤實現節點調度，請參見######。

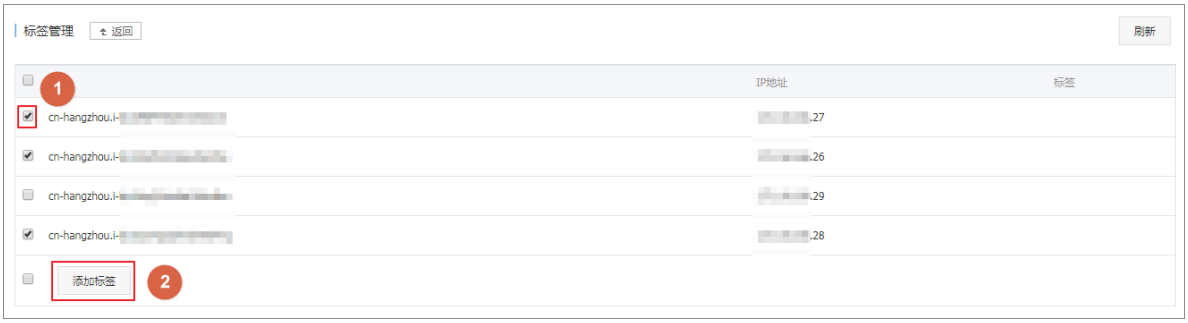
前提條件

您已經成功建立一個 Kubernetes 叢集，參見##Kubernetes##。

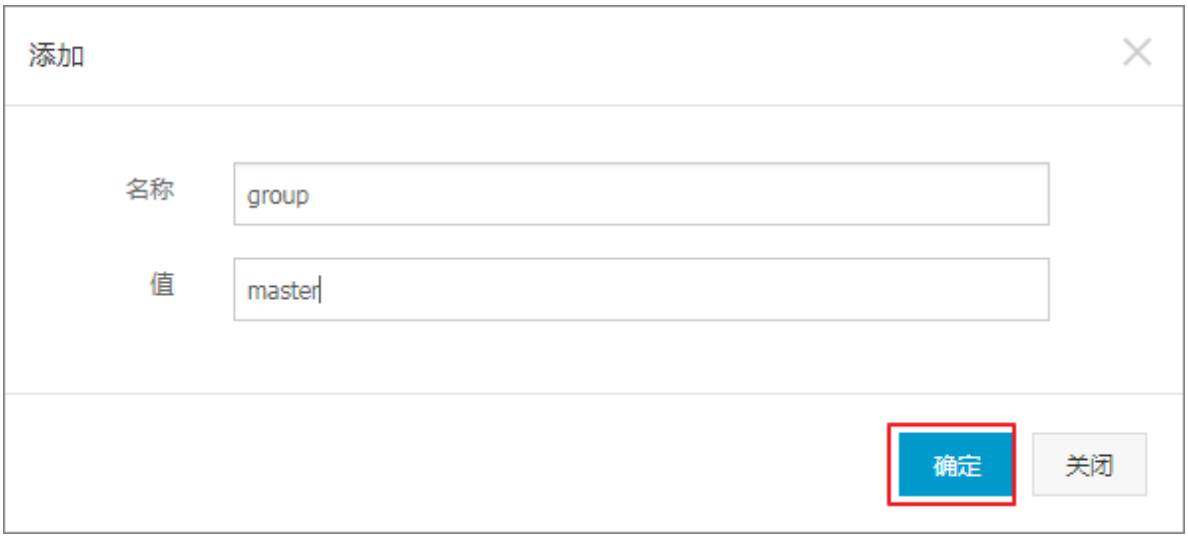
大量新增節點標籤

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點，進入節點列表頁面。
3. 選擇所需的叢集，在頁面右上方單擊標籤管理。

4. 在節點列表中，批量選擇節點，然後單擊添加標籤。



5. 在彈出的添加標籤對話方塊中，輸入標籤的名稱和值，然後單擊確定。



您可以在標籤管理頁面，看到批量節點具有相同的標籤。

通過標籤篩選節點

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點，進入節點列表頁面。
3. 選擇所需的叢集，在頁面右上方單擊標籤管理。

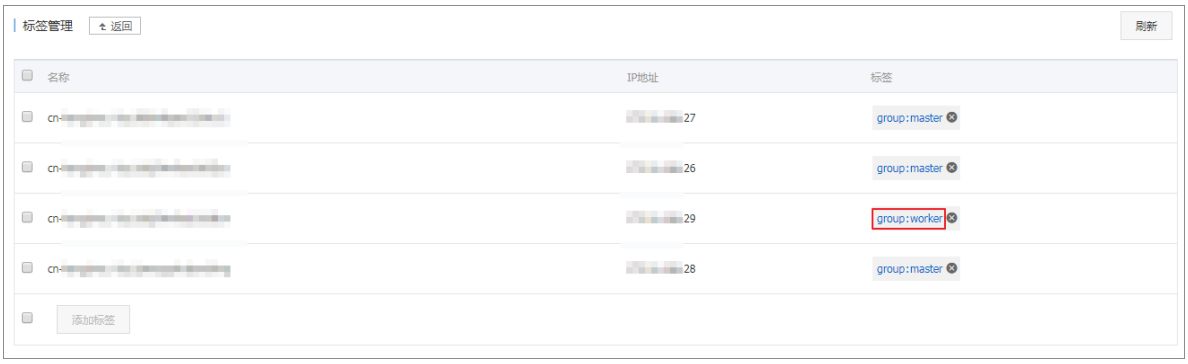
4. 選擇某個節點，單擊右側的標籤，如 `group:worker`，可通過標籤來篩選節點。

   您可看到通過 `group:worker` 標籤成功篩選出所需的節點。

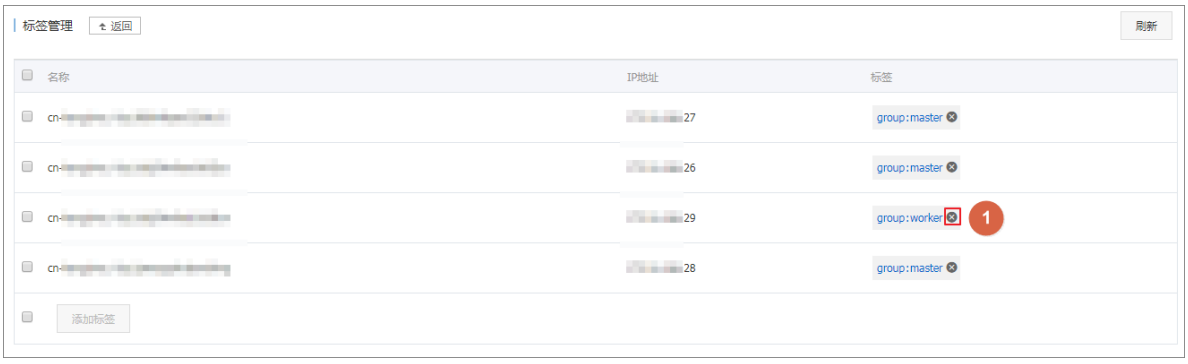
刪除節點標籤

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點，進入節點列表頁面。
3. 選擇所需的叢集，在頁面右上方單擊標籤管理。

4. 選擇某個節點，單擊標籤的刪除表徵圖，如 `group:worker`。



您可以看到該節點右側的標籤消失，節點標籤被刪除。



1.5 命名空間管理

1.5.1 設定資源配額和限制

您可通過Container Service控制台，設定命名空間的資源配額和限制。

前提条件

- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已成功建立一個樣本的命名空間**test**，參見*######*。
- 您已成功串連到叢集的Master節點地址，參見*## kubectl ## Kubernetes ##*。

背景信息

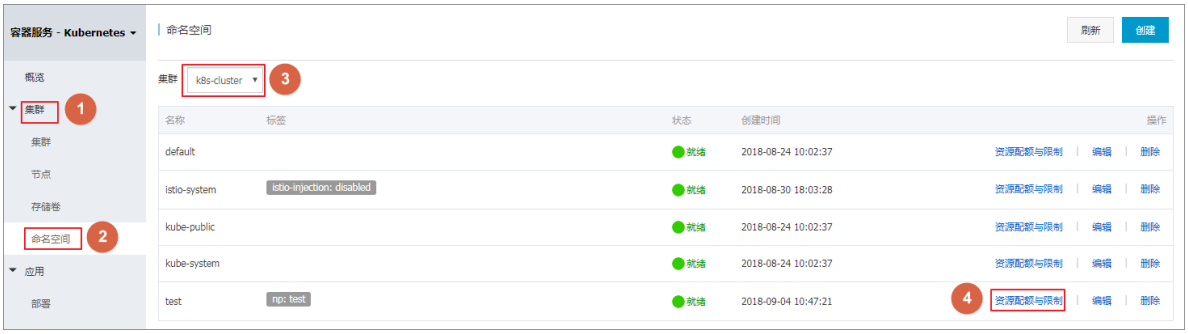
在預設的情況下，運行中的 Pod 可以無限制的使用 Node 上的 CPU 和記憶體，這意味著任意一個 Pod 都可以無節制地使用叢集的計算資源，某個命名空間的 Pod 可能會耗盡叢集的資源。

命名空間的一個重要的作用是充當一個虛擬叢集，用於多種工作用途，滿足多使用者的使用需求，因此，為命名空間配置資源額度是一種最佳實務。

您可為命名空間配置包括 CPU、記憶體、Pod 數量等資源的額度，更多資訊請參見 *Resource Quotas*。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的叢集 > 命名空間，選擇所需的叢集，然後單擊命名空間**test**右側的資源配額與限制。

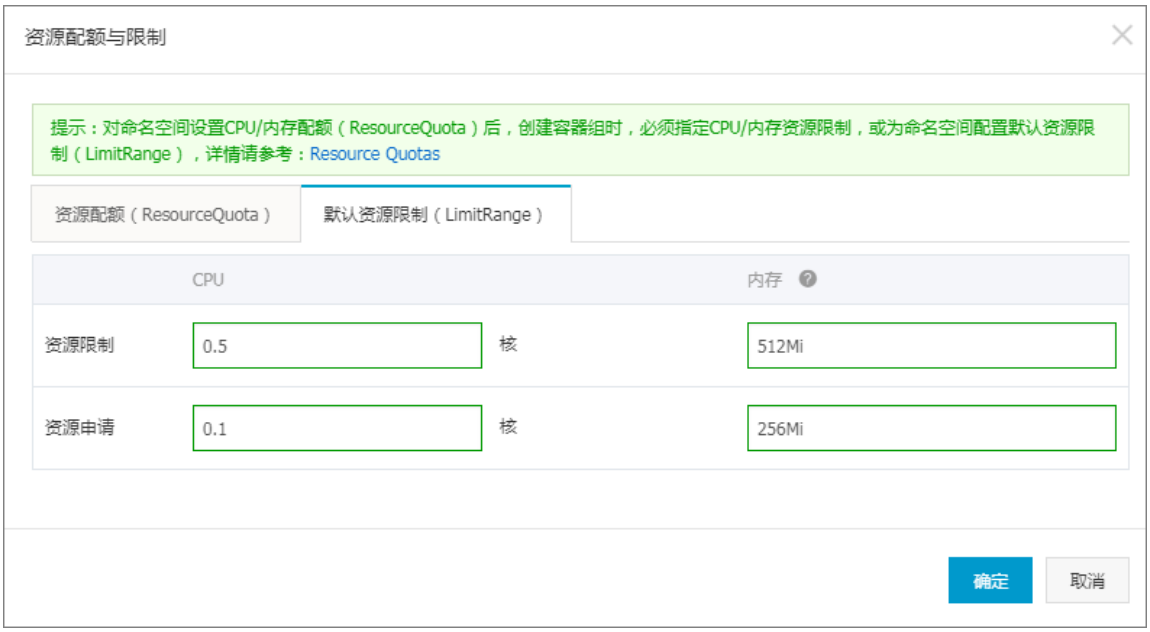


3. 在資源配額與限制對話方塊中，您可快速設定資源配額和預設資源限制。

> 说明：
> 對命名空間設定CPU/記憶體配額（ResourceQuota）後，建立容器組時，必須指定CPU/記憶體資源限制，或為命名空間配置預設資源限制（LimitRange），詳情請參考：*Resource Quotas*。

a) 您可為命名空間配置資源限額（ResourceQuota）。

b) 您可為該命名空間下的容器設定資源限制和資源申請（defaultRequest），從而控制容器的開銷。詳情參見*https://kubernetes.io//memory-default-namespace/*。



4. 您已為該命名空間建立資源配額和限制，串連到 Master 節點串連地址，執行以下命令，查看該命名空間的資源情況。

```
#kubectl get limitrange,ResourceQuota -n test
NAME AGE
limitrange/limits 8m

NAME AGE
resourcequota/quota 8m


# kubectl describe limitrange/limits resourcequota/quota -n test
Name: limits
Namespace: test
Type Resource Min Max Default Request Default Limit Max Limit/
Request Ratio
---- -------- --- --- --------------- -------------
 -----------------------
Container cpu - - 100m 500m -
Container memory - - 256Mi 512Mi -

Name: quota
Namespace: test
Resource Used Hard
-------- ---- ----
configmaps 0 100
limits.cpu 0 2
limits.memory 0 4Gi
persistentvolumeclaims 0 50
pods 0 50
requests.storage 0 1Ti
secrets 1 10
services 0 20
services.loadbalancers 0 5
```

1.5.2 編輯命名空間

前提条件

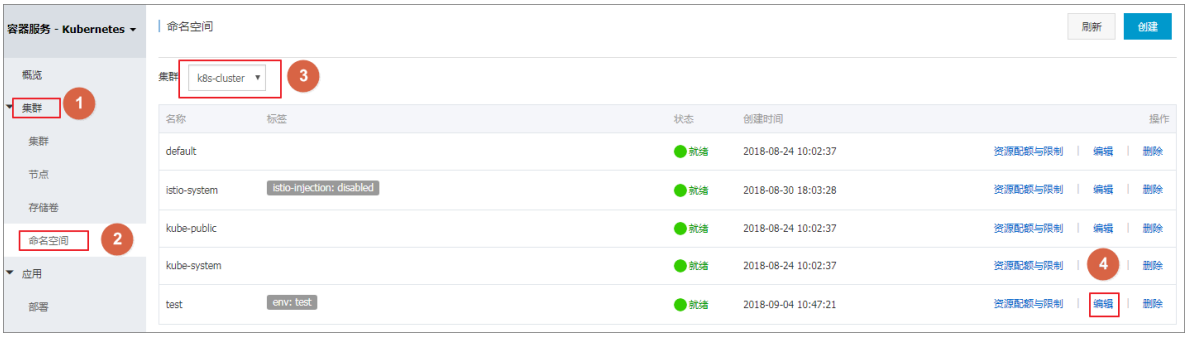
- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已成功建立一個樣本的命名空間**test**，參見*######*。

背景信息

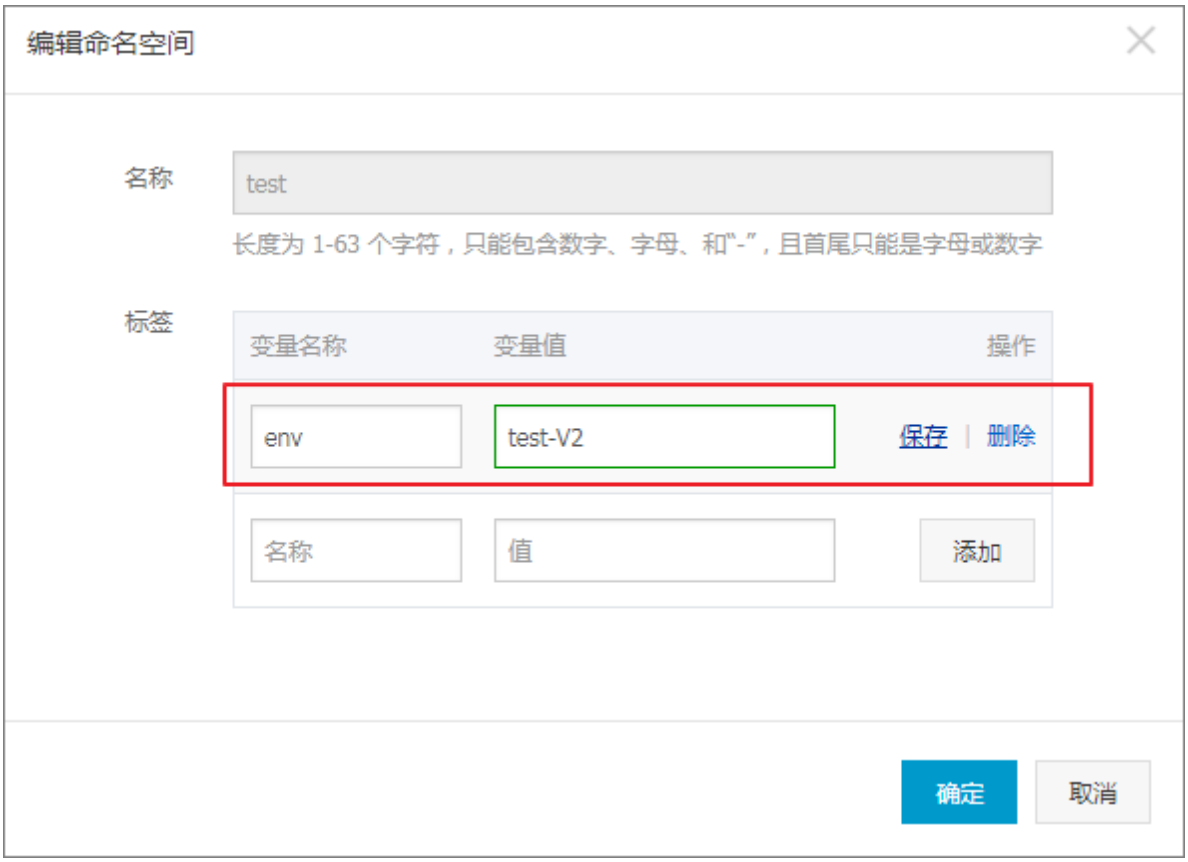
您可對命名空間進行編輯，對命名空間的標籤進行增、刪、改等操作。

操作步驟

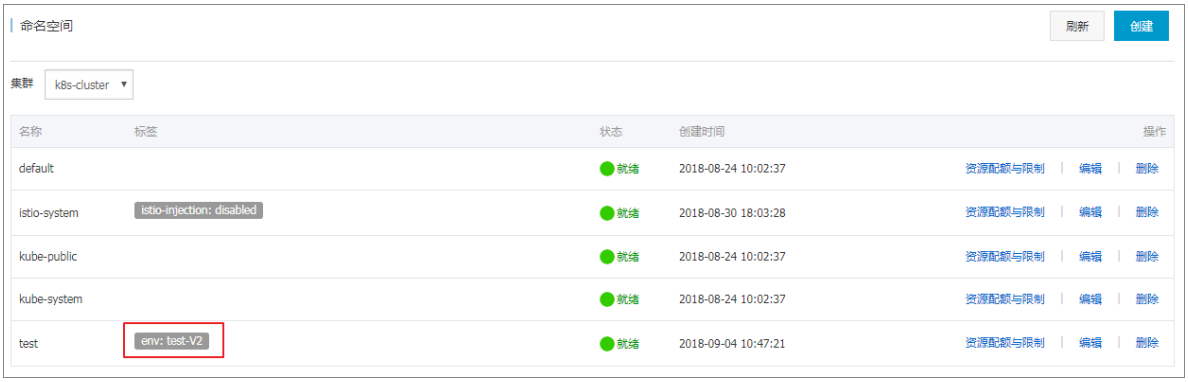
1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的叢集 > 命名空間，進入命名空間列表頁面。
3. 選擇所需的叢集，選擇所需的命名空間，單擊右側的編輯。



4. 在彈出的對話方塊中，單擊編輯，對命名空間的標籤進行修改，如將標籤修改為`env:test-V2`，然後單擊儲存。



5. 單擊確定，返回命名空間列表，該命名空間的標籤發生變化。



1.6 應用管理

1.6.1 使用鏡像建立無狀態Deployment應用

您可以使用鏡像建立一個可公網訪問的nginx應用。

前提条件

建立一個 Kubernetes 叢集。詳情請參見*##Kubernetes##*。

操作步驟

1. 登入*Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 部署，然後單擊頁面右上方的使用鏡像建立。
3. 設定應用程式名稱、部署叢集 和 命名空間、副本數量和類型，副本數量即應用程式套件含的Pod數量。然後單擊下一步 進入容器配置頁面。

> 说明：
> 本例中選擇無狀態類型，即Deployment類型。

如果您不設定命名空間，系統會預設使用 default 命名空間。

4. 設定容器配置。

> 说明：
> 您可為應用的Pod設定多個容器。

a) 設定容器的基本配置。

- 鏡像名稱：您可以單擊選擇鏡像，在彈出的對話方塊中選擇所需的鏡像並單擊確定，本例中為 nginx。

  您還可以填寫私人 registry。填寫的格式為`domainname/namespace/imagename:tag`
- 鏡像版本：您可以單擊選擇鏡像版本 選擇鏡像的版本。若不指定，預設為 latest。
- 總是拉取鏡像：為了提高效率，Container Service會對鏡像進行緩衝。部署時，如果發現鏡像 Tag 與本機快取的一致，則會直接複用而不重新拉取。所以，如果您基於上層業務便利性等因素考慮，在做代碼和鏡像變更時沒有同步修改 Tag ，就會導致部署時還是使用本機快取內舊版本鏡像。而勾選該選項後，會忽略緩衝，每次部署時重新拉取鏡像，確保使用的始終是最新的鏡像和代碼。

- 資源限制：可指定該應用所能使用的資源上限，包括 CPU 和 記憶體兩種資源，防止佔用過多資源。其中，CPU 資源的單位為 millicores，即一個核的千分之一；記憶體的單位為 Bytes，可以為 Gi、Mi 或 Ki。
- 所需資源：即為該應用預留資源額度，包括 CPU 和 記憶體兩種資源，即容器獨佔該資源，防止因資源不足而被其他服務或進程爭搶資源，導致應用不可用。
- **Init Container**：勾選該項，表示建立一個Init Container，Init Container包含一些實用的工具，具體參見*https://kubernetes.io/docs/concepts/workloads/pods/init-containers/*。

b) （可选）配置資料卷資訊。

支援配置本機存放區和雲端儲存。

- 本機存放區：支援主機目錄（hostpath）、配置項（configmap）、保密字典（secret）和臨時目錄，將對應的掛載源掛載到容器路徑中。更多資訊參見 *volumes*。
- 雲端儲存：支援雲端硬碟/NAS/OSS三種雲端儲存類型。

本例中配置了一個雲端硬碟類型的資料卷，將該雲端硬碟掛載到容器中`/tmp` 路徑下，在該路徑下產生的容器資料會儲存到雲端硬碟中。

c) （可选）配置**Log Service**，您可進行採集配置和自訂Tag設定。

> 说明：
> 請確保已部署Kubernetes叢集，並且在此叢集上已安裝日誌外掛程式。

您可對日誌進行採集配置：

- 日誌庫：即在Log Service中產生一個對應的logstore，用於儲存採集到的日誌。
- 容器內日誌路徑：支援stdout和文本日誌。
  - **stdout**： stdout 表示採集容器的標準輸出日誌。
  - 文本日誌：表示收集容器內指定路徑的日誌，本例中表示收集/var/log/nginx下所有的文本日誌，也支援萬用字元的方式。

您還可設定自訂 tag，設定tag後，會將該tag一起採集到容器的日誌輸出中。自訂 tag 可協助您給容器日誌打上tag，方便進行日誌統計和過濾等分析操作。

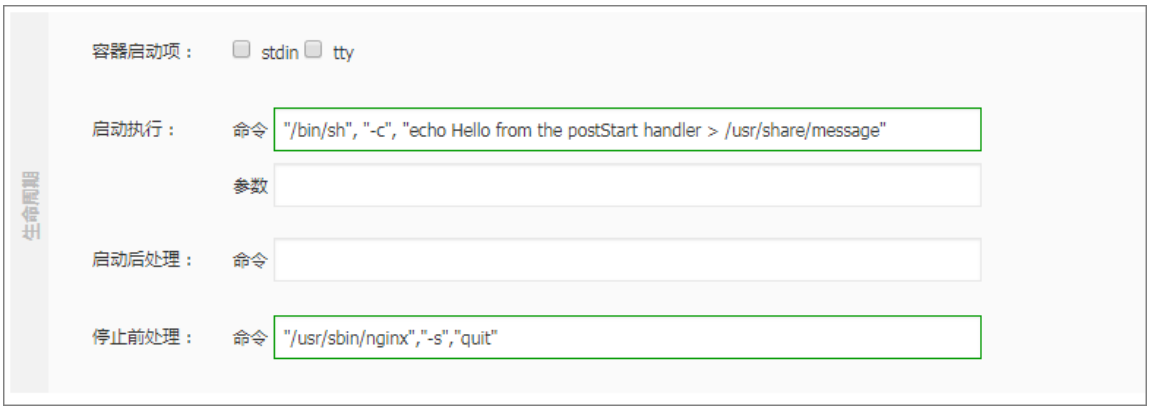
d) （可选）配置環境變數。

支援通過索引值對的形式為 Pod 配置環境變數。用於給 Pod 添加環境標誌或傳遞配置等，具體請參見 *Pod variable*。

e) 配置生命週期。

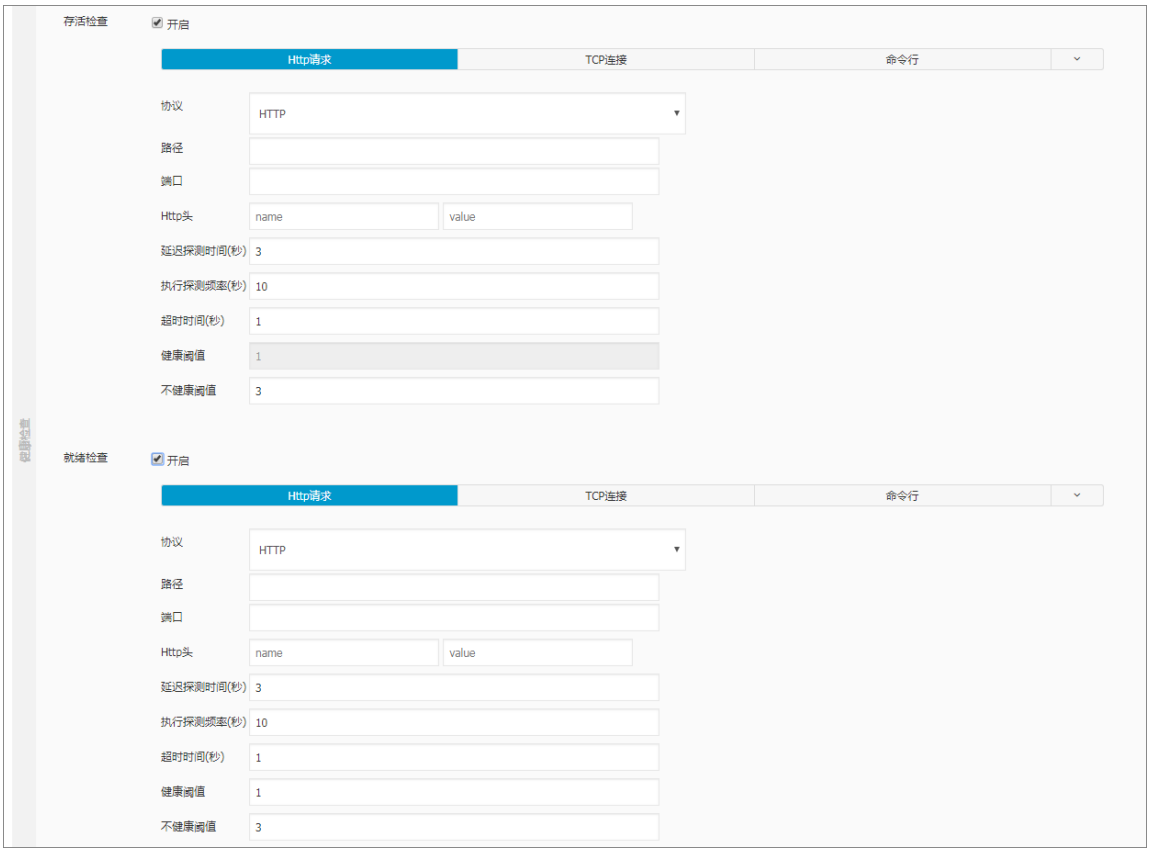
您可以為容器的生命週期配置容器啟動項、啟動執行、啟動後處理和停止前處理。具體參見 *https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/*。

- 容器啟動項：勾選 stdin 表示為該容器開啟標準輸入；勾選 tty 表示為該容器分配一個虛擬終端，以便於向容器發送訊號。通常這兩個選項是一起使用的，表示將終端 (tty) 綁定到容器的標準輸入（stdin）上，比如一個互動程式從使用者擷取標準輸入，並顯示到終端中。
- 啟動執行：為容器設定啟動前置命令和參數。
- 啟動後處理：為容器設定啟動後的命令。
- 停止前處理：為容器設定預結束命令。



f) （可选）設定健全狀態檢查

支援存活檢查（liveness）和就緒檢查（Readiness）。存活檢查用於檢測何時重啟容器；就緒檢查確定容器是否已經就緒，且可以接受流量。關於健全狀態檢查的更多資訊，請參見*https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes*。



請求類型	配置說明
HTTP請求	即向容器發送一個HTTPget 請求，支援的參數包括： • 協議：HTTP/HTTPS

請求類型	配置說明
	• 路徑：訪問HTTP server 的路徑 • 連接埠：容器暴露的訪問連接埠或連接埠名，連接埠號碼必須介於1~65535。 • HTTP頭：即HTTPHeaders，HTTP請求中自訂的要求標頭，HTTP允許重複的header。支援索引值對的配置方式。 • 延遲探測時間（秒）：即**initialDelaySeconds**，容器啟動後第一次執行探測時需要等待多少秒，預設為3秒。 • 執行探測頻率（秒）：即**periodSeconds**，指執行探測的時間間隔，預設為10s，最低為1s。 • 逾時時間（秒）：即**timeoutSeconds**，探測逾時時間。預設1秒，最小1秒。 • 健康閾值：探測失敗後，最少連續探測成功多少次才被認定為成功。預設是1，最小值是1。對於存活檢查（liveness）必須是1。 • 不健康閾值：探測成功後，最少連續探測失敗多少次才被認定為失敗。預設是3。最小值是1。
TCP串連	即向容器發送一個TCP Socket，kubelet將嘗試在指定連接埠上開啟容器的通訊端。 如果可以建立串連，容器被認為是健康的，如果不能就認為是失敗的。支援的參數包括： • 連接埠：容器暴露的訪問連接埠或連接埠名，連接埠號碼必須介於1~65535。 • 延遲探測時間（秒）：即**initialDelaySeconds**，容器啟動後第一次執行探測時需要等待多少秒，預設為15秒。 • 執行探測頻率（秒）：即**periodSeconds**，指執行探測的時間間隔，預設為10s，最低為1s。 • 逾時時間（秒）：即**timeoutSeconds**，探測逾時時間。預設1秒，最小1秒。 • 健康閾值：探測失敗後，最少連續探測成功多少次才被認定為成功。預設是1，最小值是1。對於存活檢查（liveness）必須是1。

請求類型	配置說明
	• 不健康閾值：探測成功後，最少連續探測失敗多少次才被認定為失敗。預設是3。最小值是1。
命令列	通過在容器中執行探針檢測命令，來檢測容器的健康情況。支援的參數包括： • 命令列：用於檢測容器健康情況的探測命令。 • 延遲探測時間（秒）：即**initialDelaySeconds**，容器啟動後第一次執行探測時需要等待多少秒，預設為5秒。 • 執行探測頻率（秒）：即**periodSeconds**，指執行探測的時間間隔，預設為10s，最低為1s。 • 逾時時間（秒）：即**timeoutSeconds**，探測逾時時間。預設1秒，最小1秒。 • 健康閾值：探測失敗後，最少連續探測成功多少次才被認定為成功。預設是1，最小值是1。對於存活檢查（liveness）必須是1。 • 不健康閾值：探測成功後，最少連續探測失敗多少次才被認定為失敗。預設是3。最小值是1。

5. 完成容器配置後，單擊 下一步。
6. 進行進階設定。
   a) 設定訪問設定。

      您可以設定暴露後端Pod的方式，最後單擊建立。本例中選擇ClusterIP服務和路由（Ingress），構建一個可公網訪問的nginx應用。

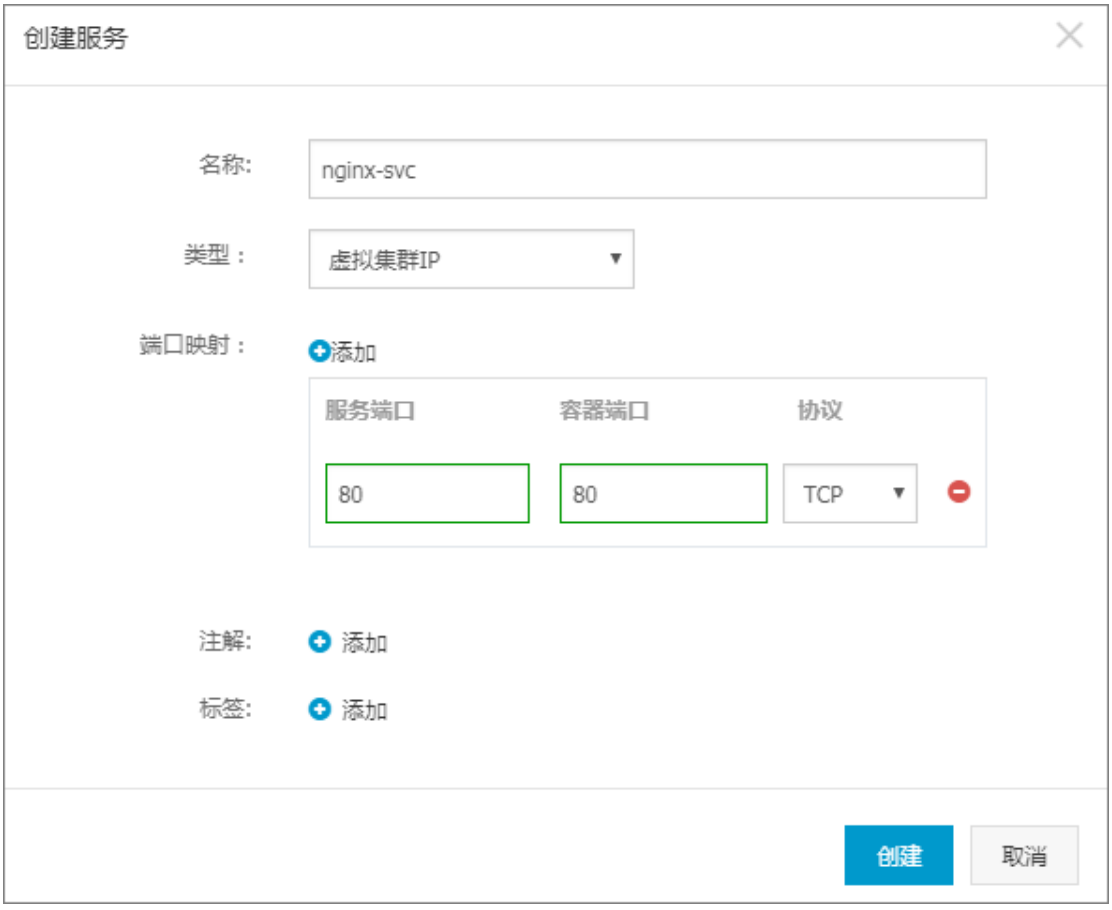
      說明：

      針對應用的通訊需求，您可靈活進行訪問設定：

      - 內部應用：對於只在叢集內部工作的應用，您可根據需要建立ClusterIP或NodePort類型的服務，來進行內部通訊。
      - 外部應用：對於需要暴露到公網的應用，您可以採用兩種方式進行訪問設定：
        - 建立LoadBalancer類型的服務：使用阿里雲提供的負載平衡服務（Server Load Balancer，SLB），該服務提供公網訪問能力。

— 建立ClusterIP、NodePort類型的服務，以及路由（Ingress）：通過路由提供公網訪問能力，詳情參見*https://kubernetes.io/docs/concepts/services-networking/ingress/*。

1. 在服務欄單擊建立，在彈出的對話方塊中進行配置，最後單擊建立。



- 名稱：您可自主設定，預設為`applicationname-svc`。
- 類型：您可以從下面 3 種服務類型中進行選擇。

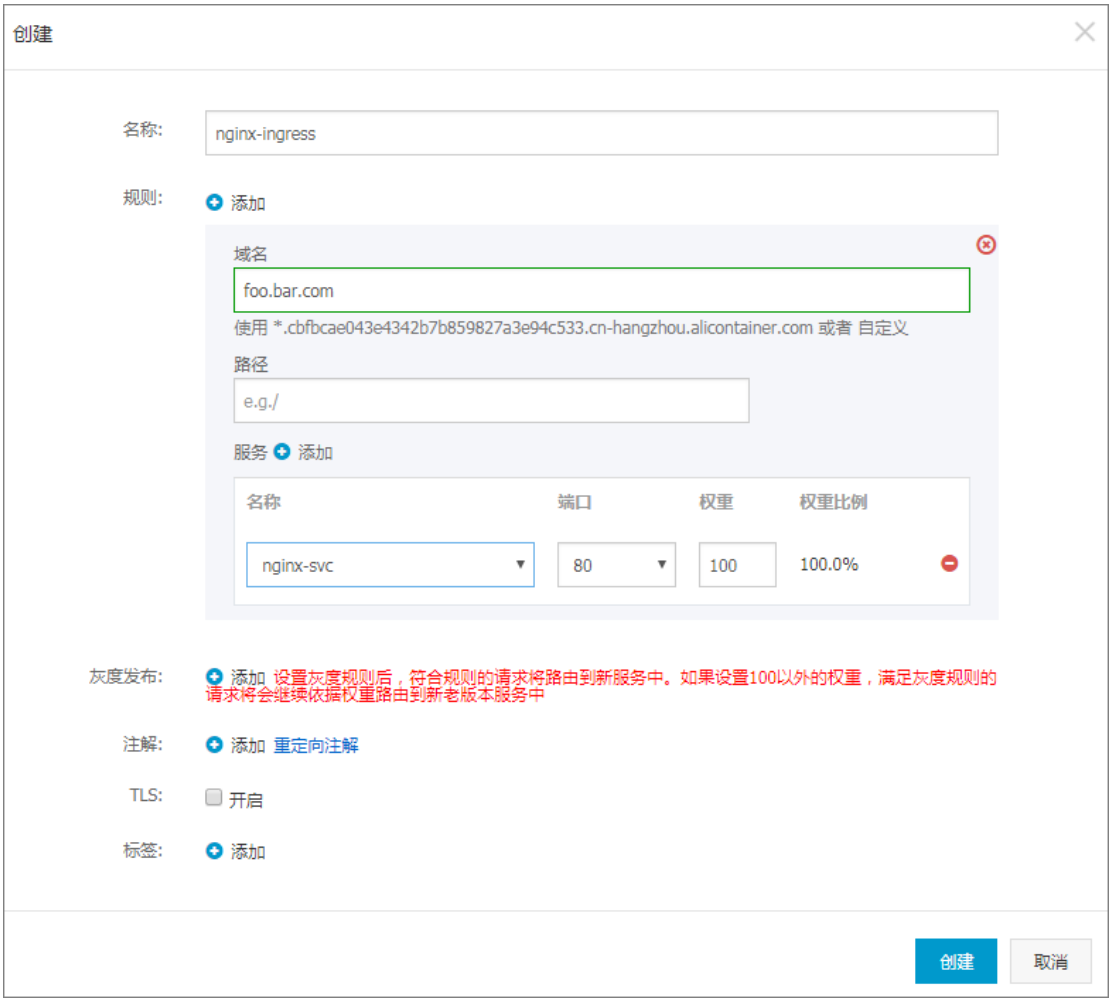
  - 虛擬叢集 IP：即 ClusterIP，指通過叢集的內部 IP 暴露服務，選擇該項，服務只能夠在叢集內部可以訪問。
  - 節點連接埠：即 NodePort，通過每個 Node 上的 IP 和靜態連接埠（NodePort）暴露服務。NodePort 服務會路由到 ClusterIP 服務，這個 ClusterIP 服務會自動建立。通過請求 `<NodeIP>:<NodePort>`，可以從叢集的外部存取一個 NodePort 服務。
  - 負載平衡：即 LoadBalancer，是阿里雲提供的負載平衡服務，可選擇公網訪問或內網訪問。負載平衡可以路由到 NodePort 服務和 ClusterIP 服務。
- 連接埠映射：您需要添加服務連接埠和容器連接埠，若類型選擇為節點連接埠，還需要自己設定節點連接埠，防止連接埠出現衝突。支援 TCP/UDP 協議。
- 註解：為該服務添加一個註解（annotation），支援負載平衡配置參數，參見###### *#Server Load Balancer#####*。
- 標籤：您可為該服務添加一個標籤，標識該服務。

2. 在路由欄單擊建立，在彈出的對話方塊中，為後端Pod配置路由規則，最後單擊建立。更多詳細的路由配置資訊，請參見######。

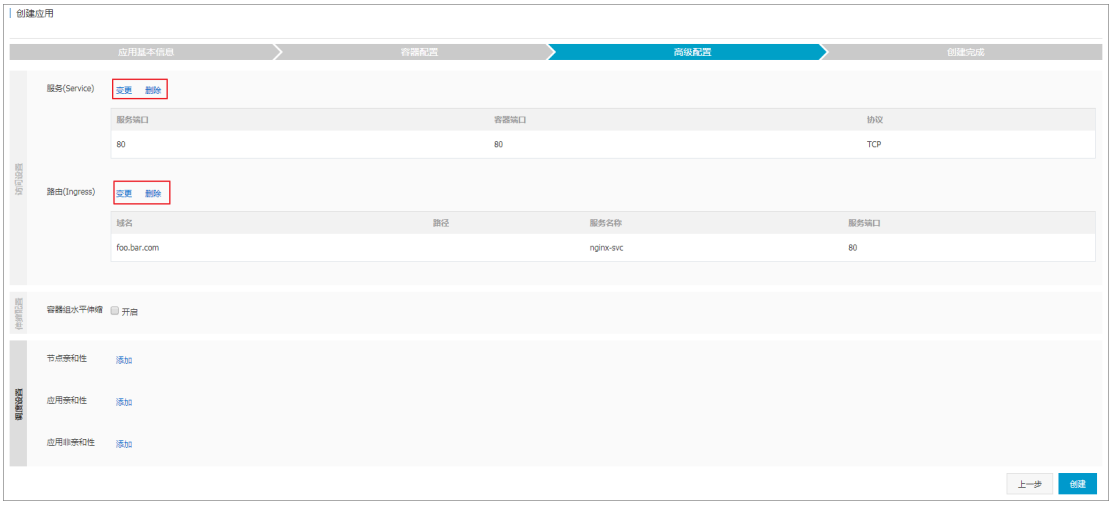
说明：

通過鏡像建立應用時，您僅能為一個服務建立路由（Ingress）。本例中使用一個虛擬機器主機名稱作為測試網域名稱，您需要在hosts中添加一條記錄。在實際工作情境中，請使用備案網域名稱。

```
101.37.224.146    foo.bar.com     #即ingress的IP
```



3. 在訪問設定欄中，您可看到建立完畢的服務和路由，您可單擊變更和刪除進行二次配置。



b) （可选）容器組水平伸縮。

您可勾選是否開啟容器組水平伸縮，為了滿足應用在不同負載下的需求，Container Service支援服容器組（Pod）的Auto Scaling，即根據容器 CPU 和記憶體資源佔用情況自動調整容器組數量。

说明：

若要啟用自動調整，您必須為容器設定所需資源，否則容器自動調整無法生效。參見容器基本配置環節。

- 指標：支援CPU和記憶體，需要和設定的所需資源類型相同。
- 觸發條件：資源使用率的百分比，超過該使用量，容器開始擴容。
- 最大容器數量：該Deployment可擴容的容器數量上限。
- 最小容器數量：該Deployment可縮容的容器數量下限。

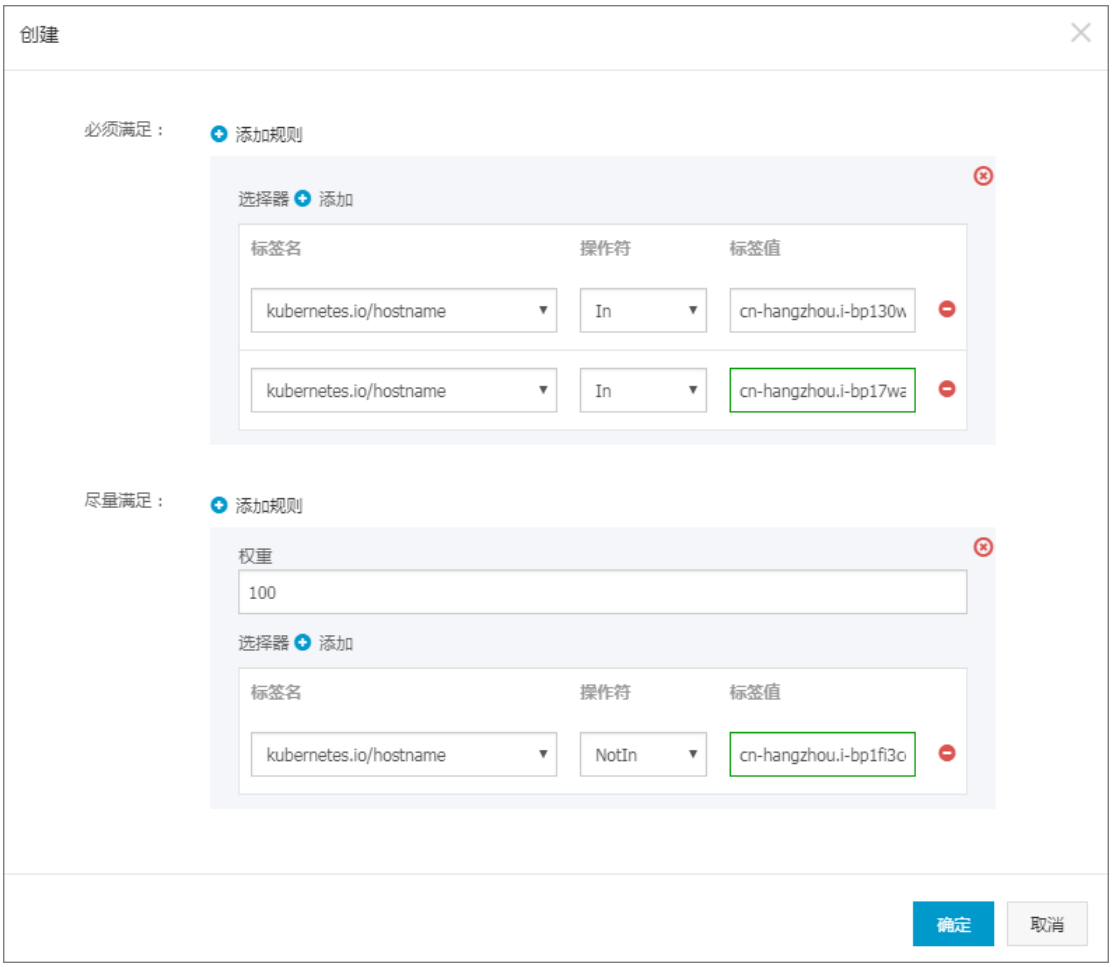
c) （可选）設定調度親和性。

您可設定節點親和性、應用親和性和應用非親和性，詳情參見*https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity*。

说明：

親和性調度依賴節點標籤和Pod標籤，您可使用內建的標籤進行調度；也可預先為節點、Pod配置相關的標籤。

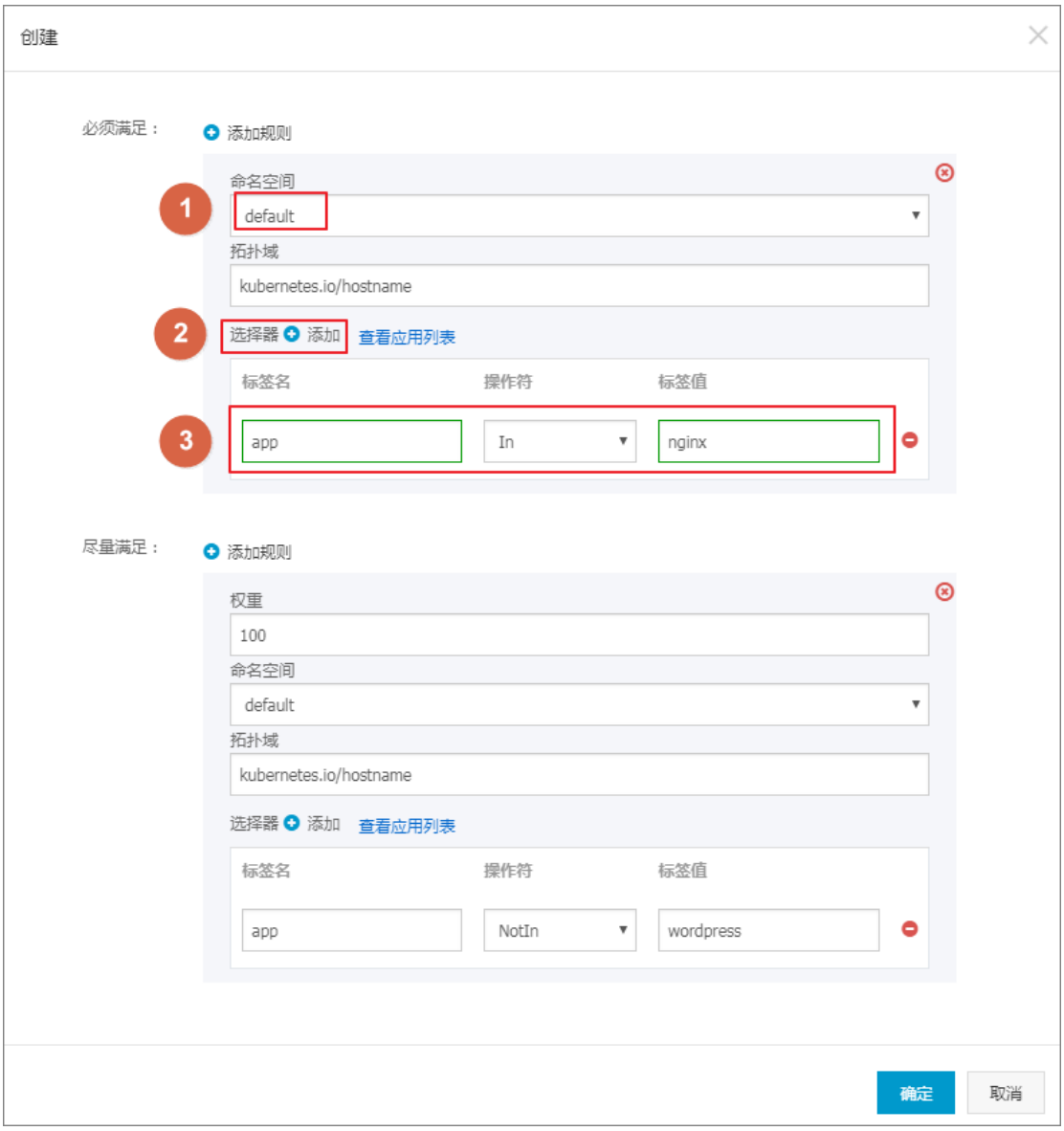
1. 設定節點親和性，通過Node節點的Label標籤進行設定。



節點調度支援硬約束和軟約束（Required/Preferred），以及豐富的匹配運算式（In, NotIn, Exists, DoesNotExist. Gt, and Lt）：

- 必須滿足，即硬約束，一定要滿足，對應**requiredDuringSchedulingIgnoredDuringExecution**，效果與`NodeSelector`相同。本例中Pod只能調度到具有對應標籤的Node節點。您可以定義多條硬約束規則，但只需滿足其中一條。
- 盡量滿足，即軟約束，不一定滿足，對應**preferredDuringSchedulingIgnoredDuringExecution**。本例中，調度會盡量不調度Pod到具有對應標籤的Node節點。您還可為軟約束規則設定權重，具體調度時，若存在多個合格節點，權重最高的節點會被優先調度。您可定義多條軟約束規則，但必須滿足全部約束，才會進行調度。

2. 設定應用親和性調度。決定應用的Pod可以和哪些Pod部署在同一拓撲域。例如，對於相互連信的服務，可通過應用親和性調度，將其部署到同一拓撲域（如同一個主機）中，減少它們之間的網路延遲。



根據節點上啟動並執行Pod的標籤（Label）來進行調度，支援硬約束和軟約束，匹配的運算式有：`In, NotIn, Exists, DoesNotExist`。

- 必須滿足，即硬約束，一定要滿足，對應**`requiredDuringSchedulingIgnoredDuringExecution`**，Pod的親和性調度必須要滿足後續定義的約束條件。
  - 命名空間：該策略是依據Pod的Label進行調度，所以會受到命名空間的約束。
  - 拓撲域：即topologyKey，指定調度時範圍，這是通過Node節點的標籤來實現的，例如指定為`kubernetes.io/hostname`，那就是以Node節點為區分範圍；如果指定為`beta.kubernetes.io/os`，則以Node節點的作業系統類型來區分。
  - 選取器：單擊選取器右側的加號按鈕，您可添加多條硬約束規則。
  - 查看應用列表：單擊應用列表，彈出對話方塊，您可在此查看各命名空間下的應用，並可將應用的標籤匯入到親和性配置頁面。

— 硬約束條件：設定已有應用的標籤、操作符和標籤值。本例中，表示將待建立的應用調度到該主機上，該主機啟動並執行已有應用具有`app:nginx`標籤。

- 盡量滿足，即軟約束，不一定滿足，對應**`preferredDuringSchedulingIgnoredDuringExecution`**。Pod的親和性調度會盡量滿足後續定義的約束條件。對於軟約束規則，您可配置每條規則的權重，其他配置規則與硬約束規則相同。

  
  说明：
  權重：設定一條軟約束規則的權重，介於1-100，通過演算法計算滿足軟約束規則的節點的權重，將Pod調度到權重最高的節點上。

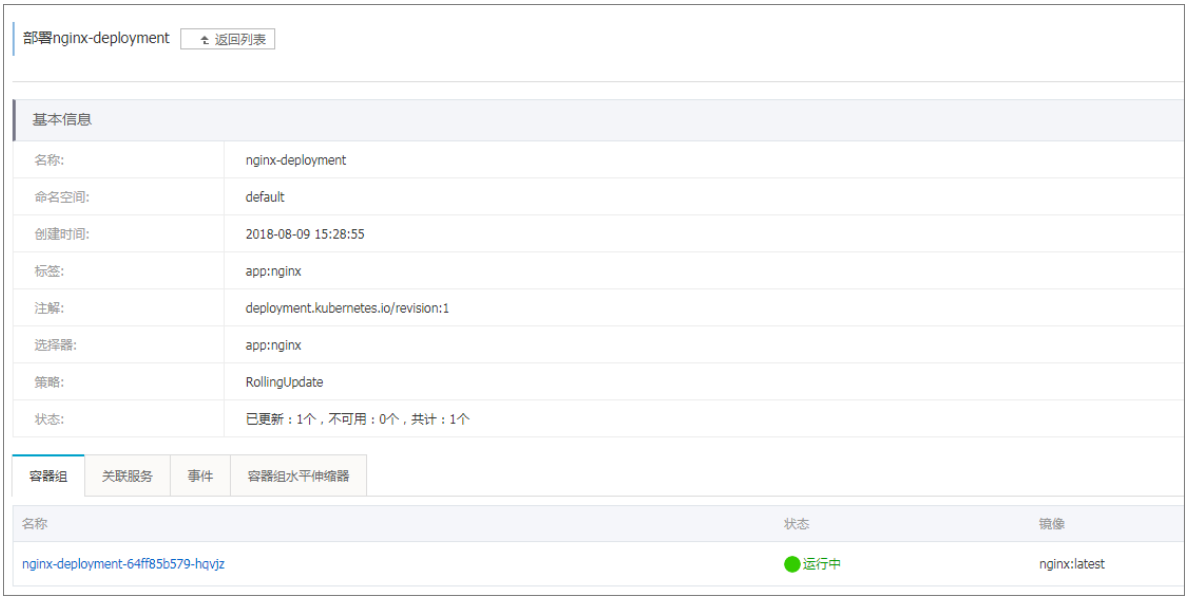
3. 設定應用非親和性調度，決定應用的Pod不與哪些Pod部署在同一拓撲域。應用非親和性調度的情境包括：

   - 將一個服務的Pod分散部署到不同的拓撲域（如不同主機）中，提高服務本身的穩定性。
   - 給予Pod一個節點的獨佔存取權限來保證資源隔離，保證不會有其它Pod來分享節點資源。
   - 把可能會相互影響的服務的Pod分散在不同的主機上。

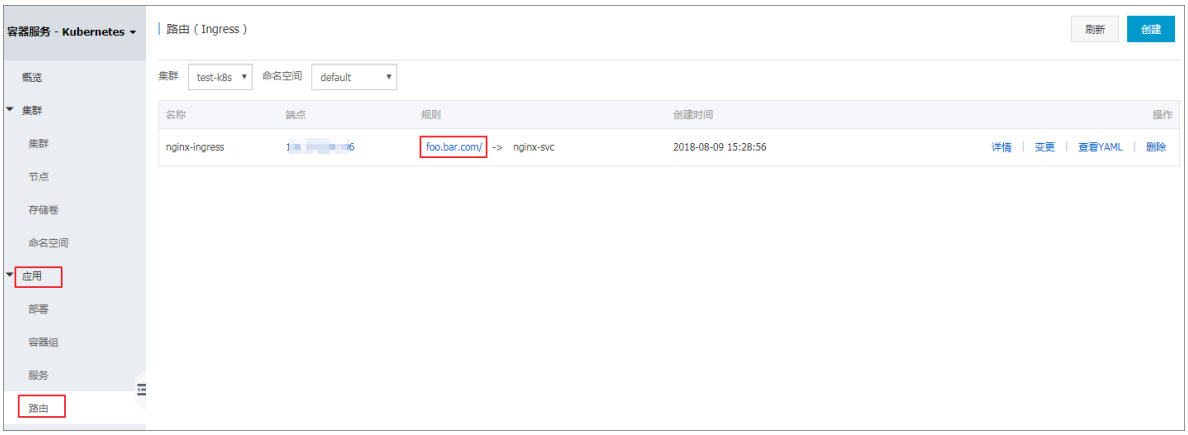
   
   说明：
   應用非親和性調度的設定方式與親和性調度相同，但是相同的調度規則代表的意思不同，請根據使用情境進行選擇。

7. 最後單擊建立。
8. 建立成功後，預設進入建立完成頁面，會列出應用程式套件含的對象，您可以單擊查看應用詳情進行查看。

預設進入建立的nginx-deployment的詳情頁面。



9. 單擊左側導覽列的應用 > 路由，可以看到路由列表下出現一條規則。



10.在瀏覽器中訪問路由測試網域名稱，您可訪問 nginx 歡迎頁面。

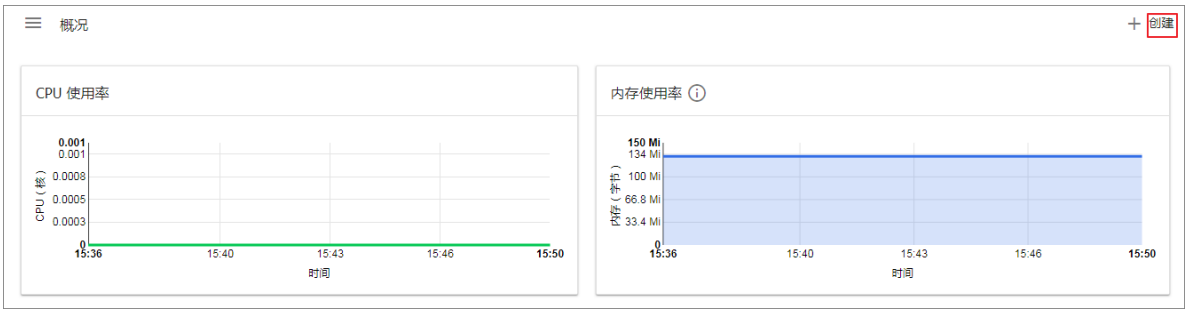
1.6.2 通過 Kubernetes Dashboard 建立應用

您可以通過 Kubernetes Dashboard 建立應用。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的控制台，進入 Kubernetes Dashboard。

4. 在 Kubernetes Dashboard 中，單擊頁面右上方的建立。



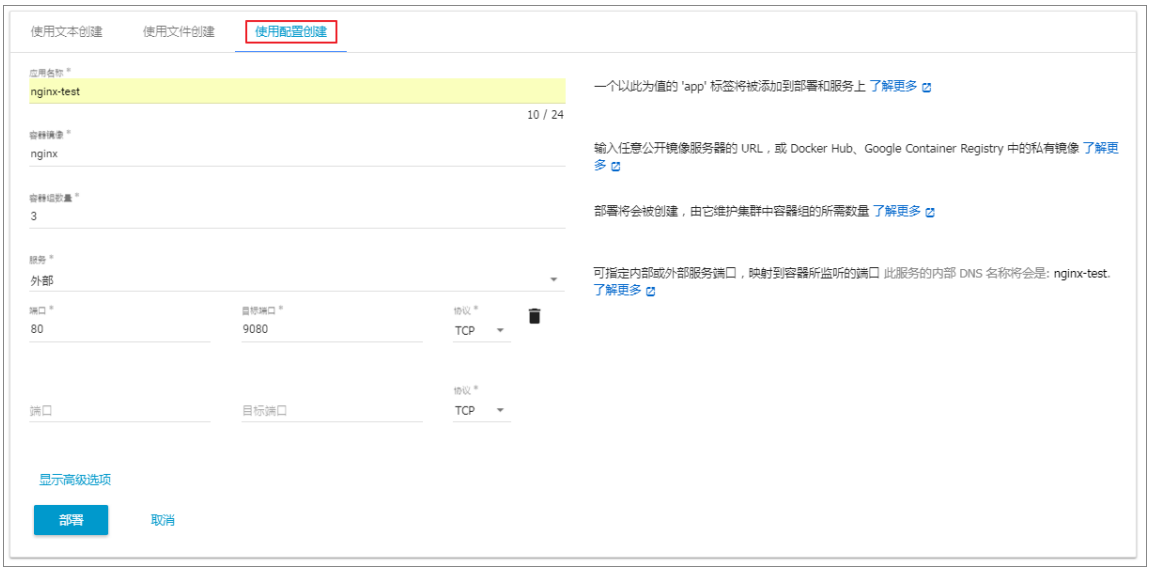
5. 在彈出的對話方塊中，設定應用的資訊。

   您可以通過以下 3 種方法之一建立應用：

   - 使用文本建立：直接輸入 YAML 或 JSON 格式的編排代碼建立應用。您需要瞭解對應的編排格式，如下所示是一個 YAML 格式的編排模板。

   - 使用檔案建立：通過匯入已有的 YAML 或 JSON 設定檔建立應用。
   - 使用配置建立：本例中通過指定下邊的設定建立應用。

     - 應用程式名稱：所建立應用的名稱。本樣本中為 `nginx`。
     - 容器鏡像：所要使用的鏡像的 URL。本樣本使用的是 Docker *Nginx*。
     - 容器組數量：建立的應用的 pod 個數。
     - 服務：可設定為外部 或內部。外部表示建立一個可以從叢集外部存取的服務；內部表示建立一個叢集內部可以訪問的服務。
     - 進階選項：您可以選擇顯示進階選項，對標籤、環境變數等選項進行配置。 此設定將流量負載平衡到三個 Pod。



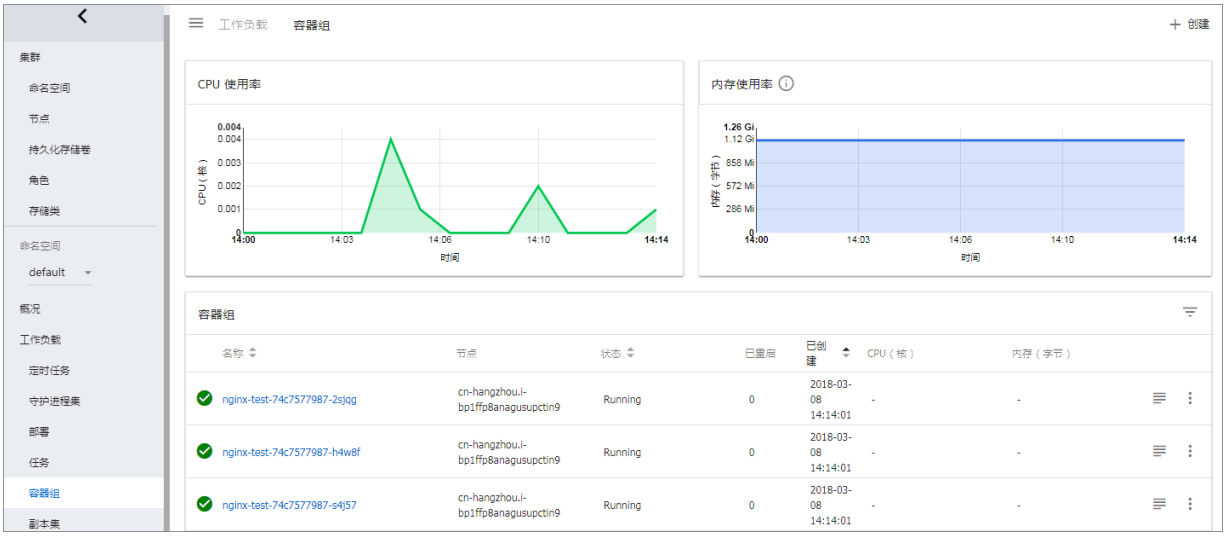
6. 單擊部署 部署這些容器和服務。

   您也可以單擊顯示進階選項 展開進階選項進一步配置相關參數。

后续操作

單擊部署後，您可以查看應用的服務或查看應用的容器。

單擊左側導覽列中的容器，您可以通過左側的表徵圖查看每個 Kubernetes 對象的狀態。 表示對象仍然處於部署狀態。 表示對象已經完成部署。



1.6.3 通過編排模板建立應用

在Container Service kubernetes 模板編排中，您需要自己定義一個應用運行所需的資來源物件，通過標籤選取器等機制，將資來源物件組合成一個完整的應用。

前提条件

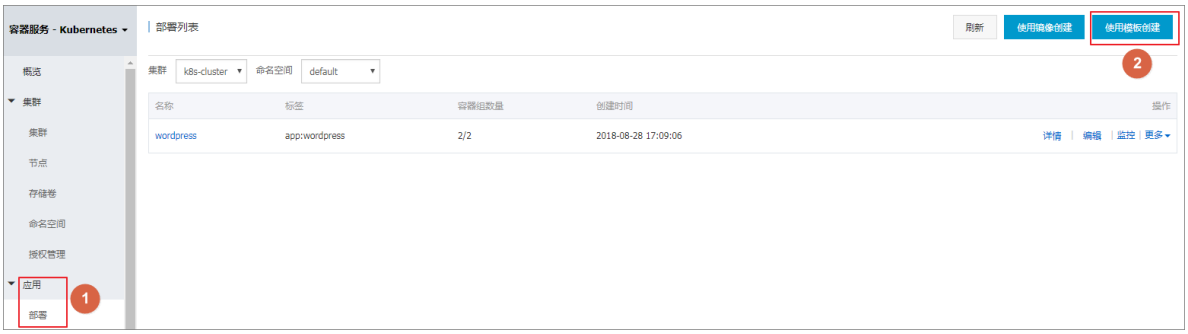
建立一個 kubernetes 叢集，參見*##Kubernetes##*。

背景信息

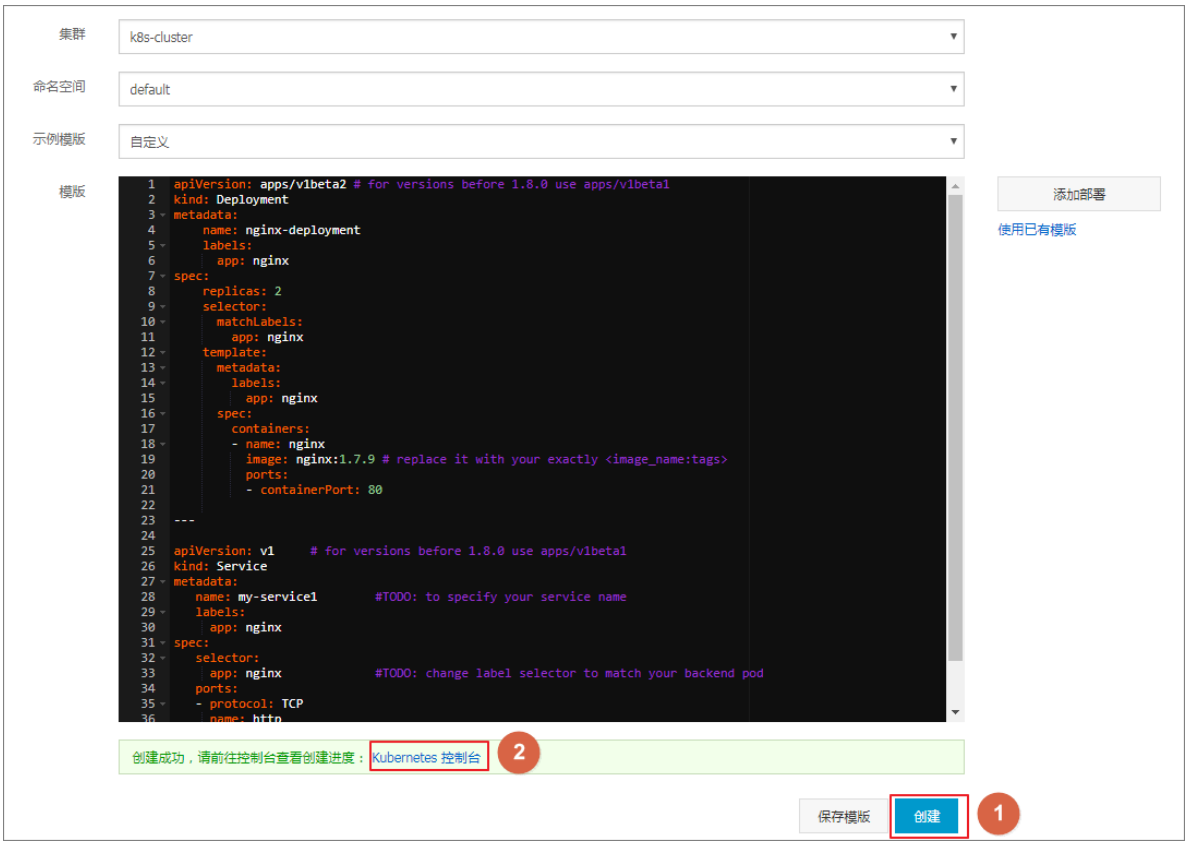
本例示範如何通過一個編排模板建立 nginx 應用，包含一個 Deployment 和 Service，後端 Deployment會建立Pod 資來源物件， Service 會綁定到後端 Pod 上，形成一個完整的 nginx 應用。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面。
3. 單擊頁面右上方的使用模板建立。



4. 對模板進行相關配置，完成配置後單擊建立。
   - 叢集：選擇目的地組群。資來源物件將部署在該叢集內。
   - 命名空間：選擇資來源物件所屬的命名空間，預設是 default。除了節點、持久化儲存卷等底層計算資源以外，大多數資來源物件需要作用於命名空間。
   - 樣本模板：阿里雲Container Service提供了多種資源類型的 Kubernetes yaml 樣本模板，讓您快速部署資來源物件。您可以根據 Kubernetes Yaml 編排的格式要求自主編寫，來描述您想定義的資源類型。
   - 添加部署：您可通過此功能快速定義一個Yaml模板。
   - 使用已有模板：您可將已有編排模板匯入到模板配置頁面。



下面是一個 nginx 應用的樣本編排，基於Container Service內建的編排模板。通過該編排模板，即可快速建立一個屬於 nginx 應用的 deployment 。

说明：

Container Service支援Kubernetes Yaml編排，支援通過---符號將資來源物件分隔，從而通過一個模板建立多個資來源物件。

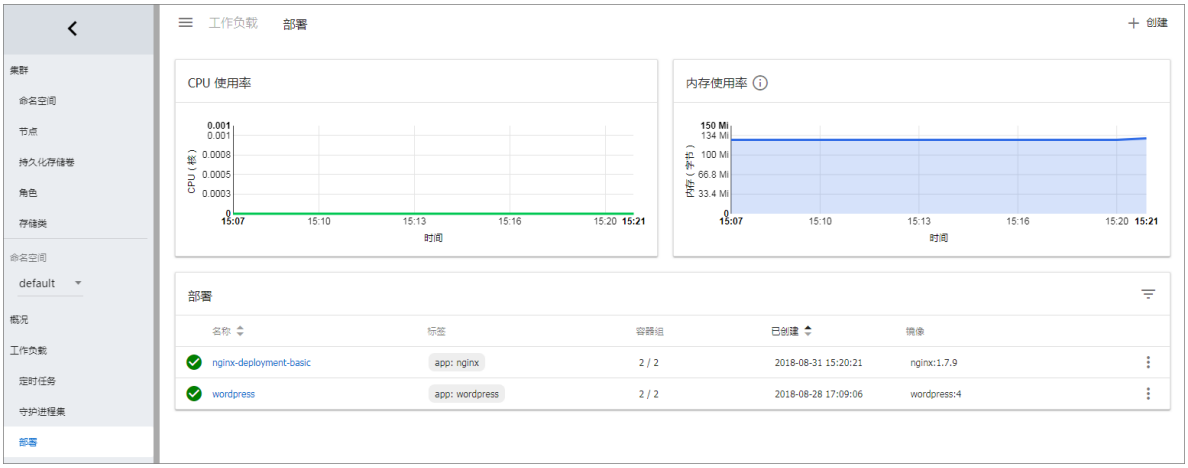
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
    name: nginx-deployment
    labels:
      app: nginx
spec:
    replicas: 2
    selector:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
```

```
          image: nginx:1.7.9 # replace it with your exactly <
image_name:tags>
          ports:
          - containerPort: 80

---

apiVersion: v1      # for versions before 1.8.0 use apps/v1beta1
kind: Service
metadata:
   name: my-service1        #TODO: to specify your service name
   labels:
     app: nginx
spec:
   selector:
     app: nginx             #TODO: change label selector to match
your backend pod
   ports:
   - protocol: TCP
     name: http
     port: 30080            #TODO: choose an unique port on each
node to avoid port conflict
     targetPort: 80
   type: LoadBalancer        ##本例中將type從Nodeport修改為LoadBalancer
```

5. 單擊建立後。會提示部署狀態資訊。成功後，單擊**Kubernetes** 控制台前往Kubernetes Dashboard 查看部署進度。



6. 在 Kubernetes Dashboard 裡，您可以看到 my-service1 服務已成功部署，並暴露了外部入口。單擊外部入口的訪問地址。

7. 您可以在瀏覽器中訪問 nginx 服務歡迎頁面。

后续操作

您也可返回Container Service首頁，單擊左側導覽列中的應用 > 服務，查看該nginx的服務。

1.6.4 通過命令管理應用

您可以通過命令建立應用或者查看應用的容器。

前提條件

在本地使用命令前，您需要先設定*## kubectl ## Kubernetes ##*。

通過命令建立應用

可以通過運行以下語句來運行簡單的容器（本樣本中為 Nginx Web 服務器）：

```
 # kubectl run -it nginx --image=registry.aliyuncs.com/spacexnice/
netdia:latest
```

此命令將為該容器建立一個服務入口，指定 `--type=LoadBalancer` 將會為您建立一個阿里雲負載平衡路由到該 Nginx 容器。

```
# kubectl expose deployment nginx --port=80 --target-port=80 --type=
LoadBalancer
```

通過命令查看容器

運行如下命令列出所有 default 命名空間裡正在啟動並執行容器。

```
root@master # kubectl get pods
NAME                                READY     STATUS    RESTARTS
AGE
nginx-2721357637-dvwq3              1/1       Running   1
9h
```

1.6.5 建立服務

Kubernetes Service 定義了這樣一種抽象：一個 Pod 的邏輯分組，一種可以訪問它們的策略，通常稱為微服務。這一組 Pod 能夠被 Service 訪問到，通常是通過 Label Selector 來實現的。

在 Kubernetes 中，pod 雖然擁有獨立的 IP， 但 pod 會快速地建立和刪除，因此，通過 pod 直接對外界提供服務不符合高可用的設計準則。通過 service 這個抽象，Service 能夠解耦 frontend（前端）和 backend（後端） 的關聯，frontend 不用關心 backend 的具體實現，從而實現松耦合的微服務設計。

更多詳細的原理，請參見 *Kubernetes service*。

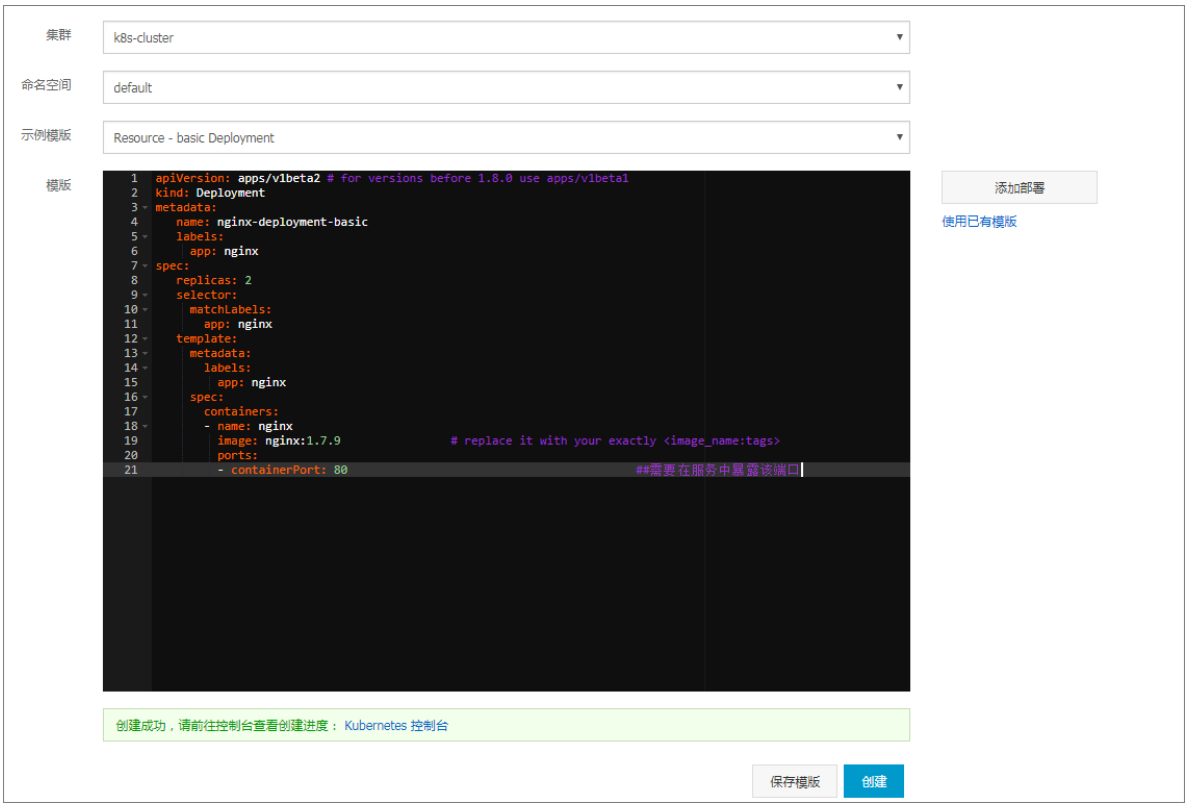
前提條件

您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。

步驟1 建立 deployment

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，單擊頁面右上方的使用模板建立。

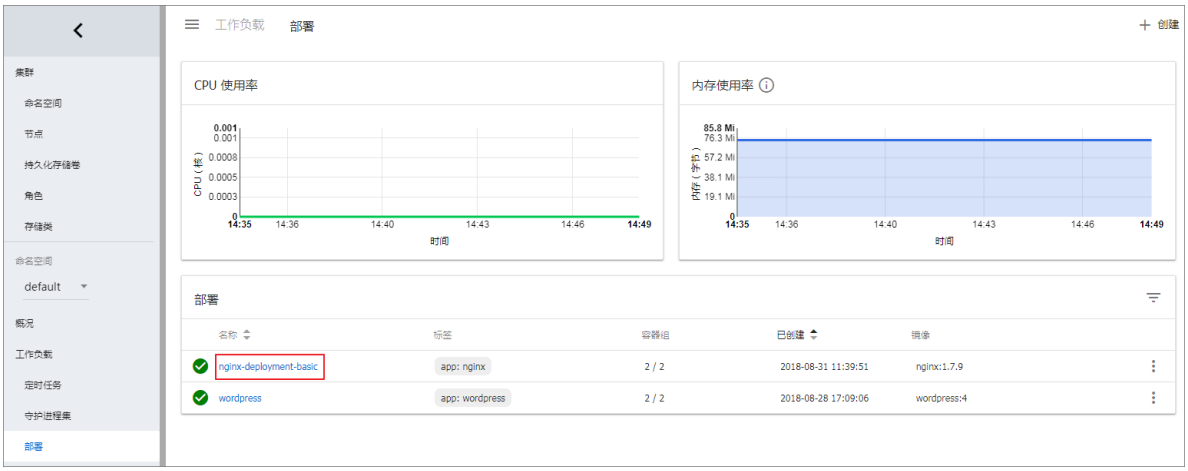
3. 選擇所需的叢集，命名空間，選擇範例模板或自訂，然後單擊建立。



本例中，樣本模板是一個 nginx 的 deployment。

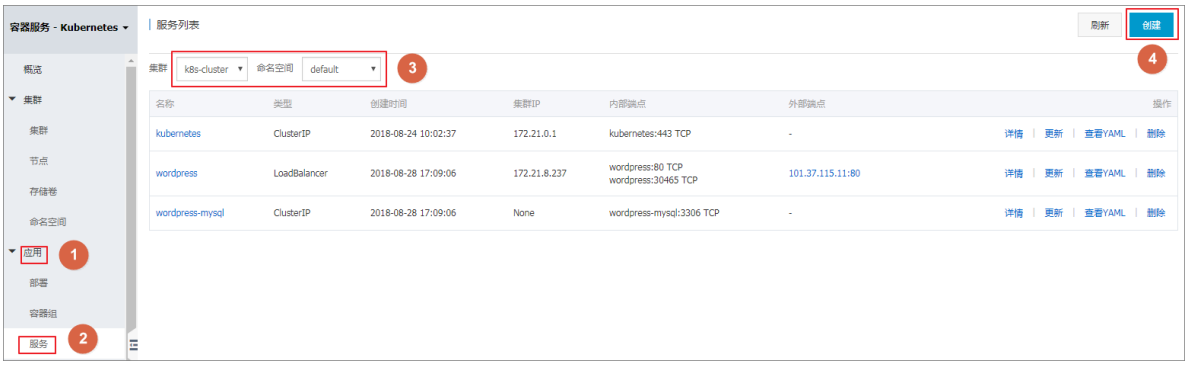
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
   name: nginx-deployment-basic
   labels:
     app: nginx
spec:
   replicas: 2
   selector:
     matchLabels:
       app: nginx
   template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
         image: nginx:1.7.9                # replace it with your
exactly <image_name:tags>
         ports:
         - containerPort: 80
  ##需要在服務中暴露該連接埠
```

4. 單擊**Kubernetes** 控制台，進入 Kubernetes Dashboard，查看該 Deployment 的運行狀態。



步驟2 建立服務

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 服務，進入服務列表頁面。
3. 選擇所需的叢集和命名空間，單擊頁面右上方的建立。



4. 在彈出的建立服務對話方塊中，進行配置。

- 名稱：輸入服務的名稱，本例中為 nginx-vc。
- 類型：選擇服務類型，即服務訪問的方式，包括：
  - 虛擬叢集 IP：即 ClusterIP，指通過叢集的內部 IP 暴露服務，選擇該值，服務只能夠在叢集內部可以訪問，這也是預設的 ServiceType。

- 節點連接埠：即 NodePort，通過每個 Node 上的 IP 和靜態連接埠（NodePort）暴露服務。NodePort 服務會路由到 ClusterIP 服務，這個 ClusterIP 服務會自動建立。通過請求 `<NodeIP>:<NodePort>`，可以從叢集的外部存取一個 NodePort 服務。
- 負載平衡：即 LoadBalancer，指阿里雲提供的負載平衡服務（SLB），可選擇公網訪問或內網訪問。阿里雲負載平衡服務可以路由到 NodePort 服務和 ClusterIP 服務。

- 關聯部署：選擇服務要綁定的後端對象，本例中是前面建立的 nginx-deployment-basic 。若不進行關聯部署，則不會建立相關的 Endpoints 對象，您可自己進行綁定，參見 *services-without-selectors*。
- 連接埠映射：添加服務連接埠和容器連接埠，容器連接埠需要與後端的 pod 中暴露的容器連接埠一致。
- 註解：為該服務添加一個註解（annotation），配置負載平衡的參數，例如設定`service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20`表示將該服務的頻寬峰值設定為20Mbit/s，從而控制服務的流量。更多參數請參見*#######Server Load Balancer#####*。
- 標籤：您可為該服務添加一個標籤，標識該服務。

5. 單擊建立，nginx-svc 服務出現在服務列表中。

6. 您可查看服務的基本資料，在瀏覽器中訪問 nginx-svc 的外部端點。

至此，您完成如何建立一個關聯到後端的 deployment 的服務，最後成功訪問 Nginx 的歡迎頁面。

1.6.6 服務伸縮

應用建立後，您可以根據自己的需求來進行服務擴容或縮容。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。

3. 選擇所需的叢集並單擊右側的控制台，進入 Kubernetes Dashboard。

4. 在 Kubernetes Dashboard 中，單擊左側導覽列中的部署，查看部署的 Deployment。

5. 選擇所需的 Deployment，單擊右側的操作表徵圖並單擊伸縮。

6. 在彈出的對話方塊中，將所需容器組數修改為 2，並單擊確定。

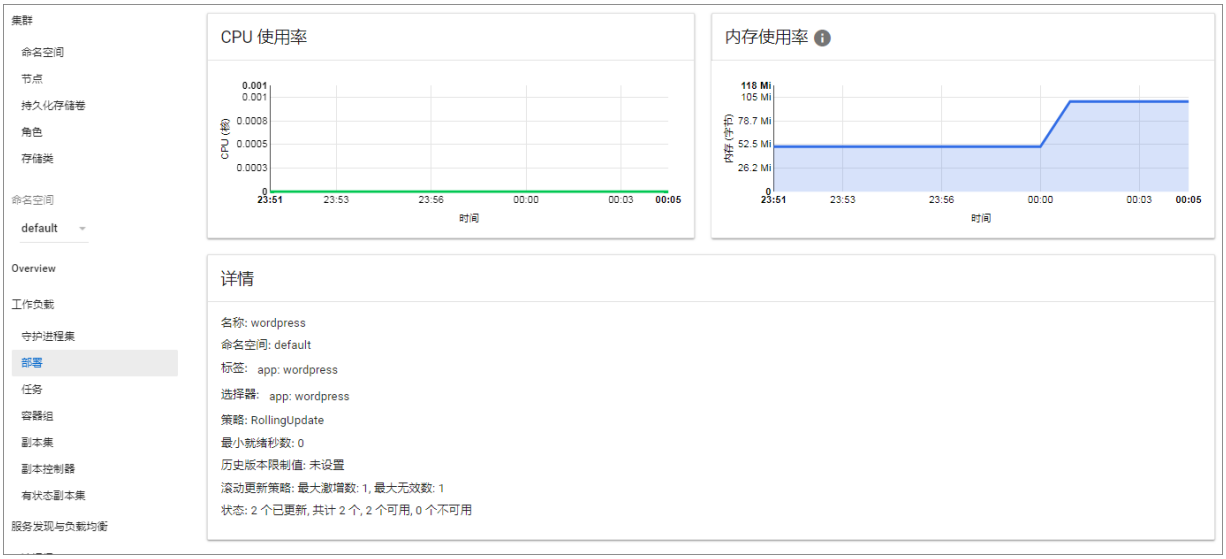
   此操作會擴容一個 Pod，將副本數升到 2。

后续操作

您可以通過左側的表徵圖查看每個 Kubernetes 對象的狀態。 表示對象仍然處於部署狀態。 表示對象已經完成部署。

應用部署完成後，您可以單擊某個部署的名稱查看正在啟動並執行 Web 服務的詳細資料。您可以查看部署中的複本集以及這些複本集所使用的 CPU 和 Memory 資源的相關資訊。

> 说明：
> 如果看不到資源，請耐心等待幾分鐘。



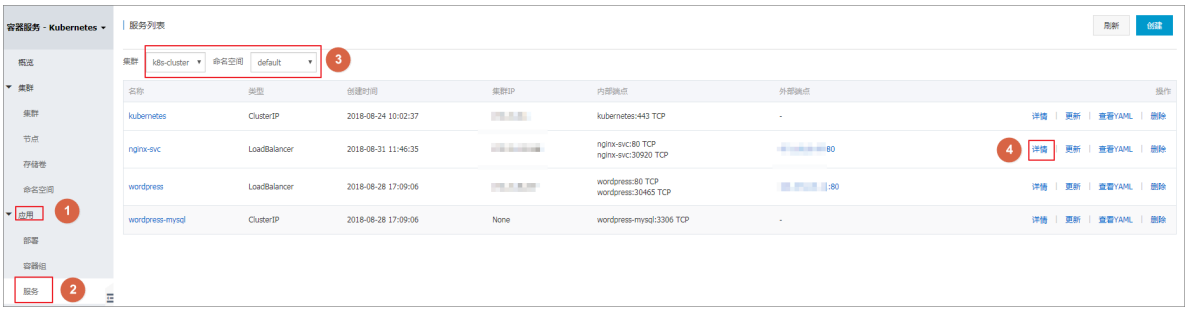
1.6.7 查看服務

背景信息

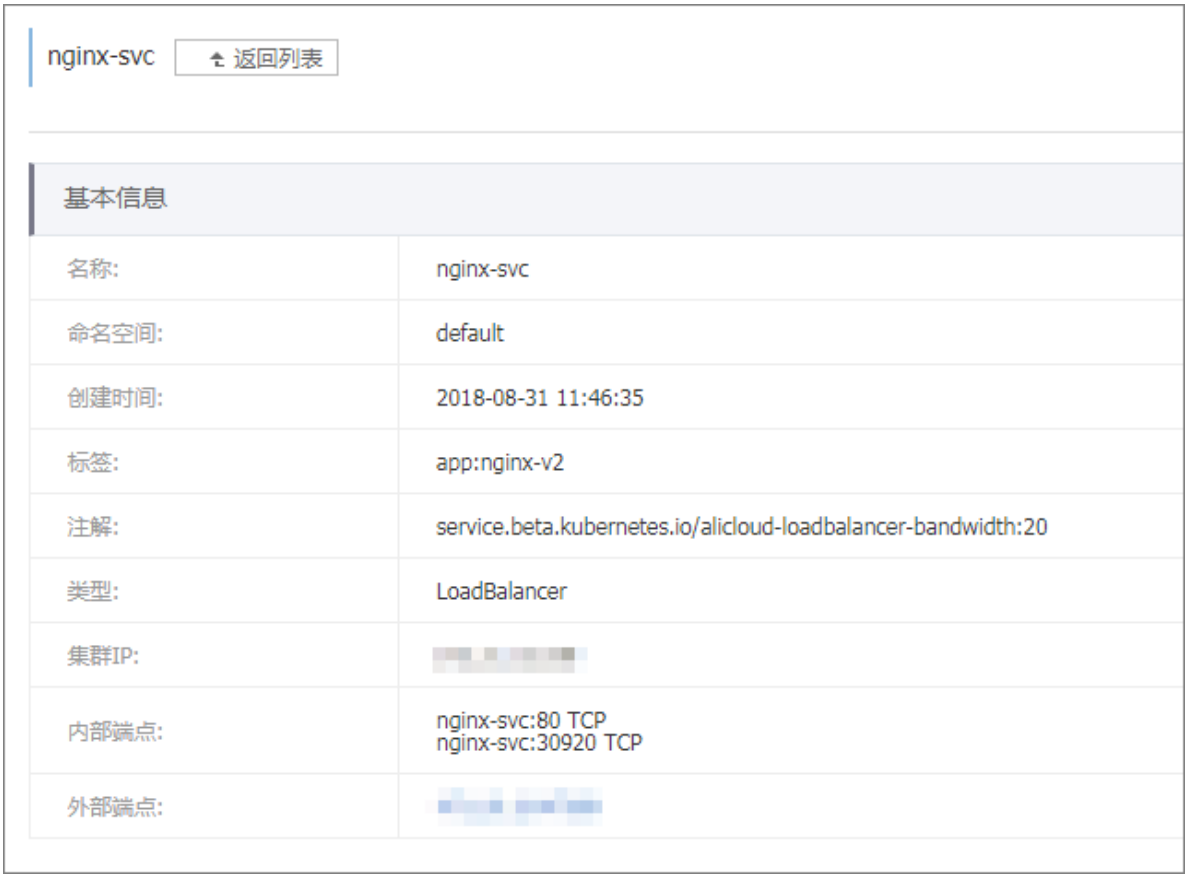
您在建立應用時，如果配置了外部服務，除了運行容器，Kubernetes Dashboard 還會建立外部 Service，用於預配負載平衡器，以便將流量引入到叢集中的容器。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 服務，進入服務列表頁面。
3. 您可以選擇所需的叢集和命名空間，選擇所需的服務，單擊右側的詳情。



您可查看服務的名稱、類型、建立時間、叢集 IP 以及外部端點等資訊。本例中您可看到分配給服務的外部端點（IP 位址）。您可以單擊該 IP 位址訪問nginx應用。



4. （可选）您也可進入叢集的 Kubernetes Dashboard。在左側導覽列中單擊服務，查看所有服務。

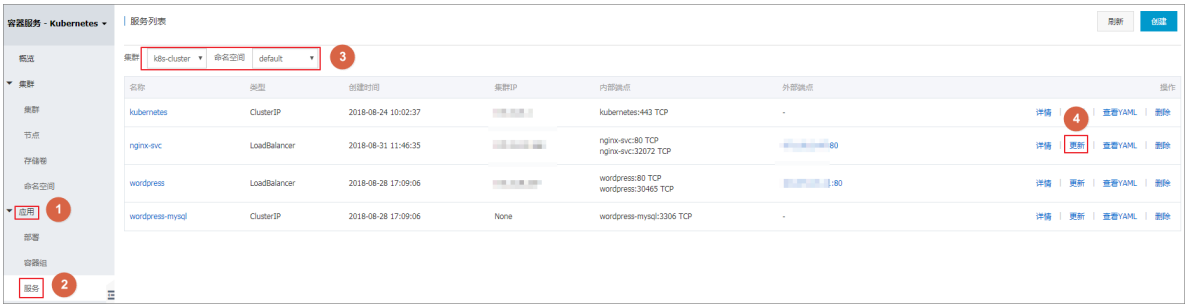
1.6.8 更新服務

您可以通過Container Service管理主控台的服務列表頁面或者 Kubernetes Dashboard 變更服務的配置。

通過Container Service控制台服務列表頁面

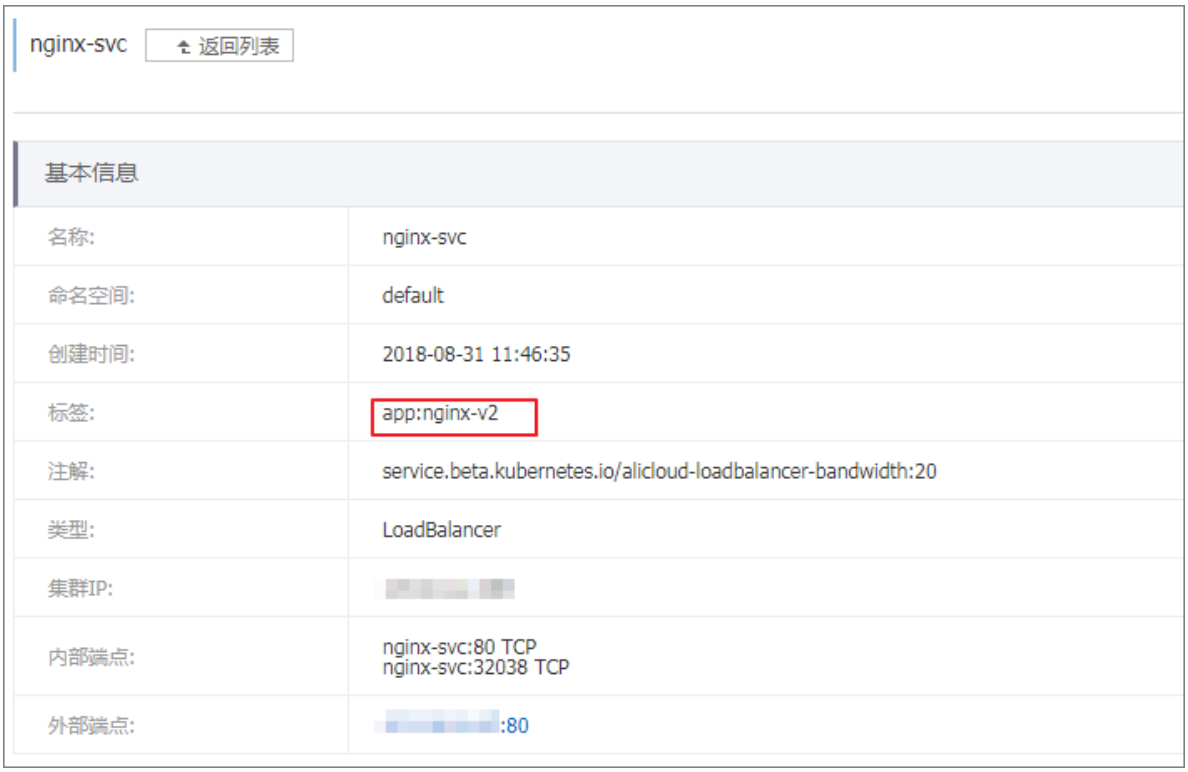
1. 登入*Container Service#####*。

2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 服務，進入服務列表頁面。
3. 選擇叢集和命名空間，選擇所需的服務（本樣本中選擇 nginx-svc），單擊右側的更新。



4. 在彈出的更新對話方塊中，進行配置修改，然後單擊確定。

5. 在服務列表中，找到所需的服務，單擊右側的詳情，查看服務變化的情況。本例中，對服務的標籤進行修改。

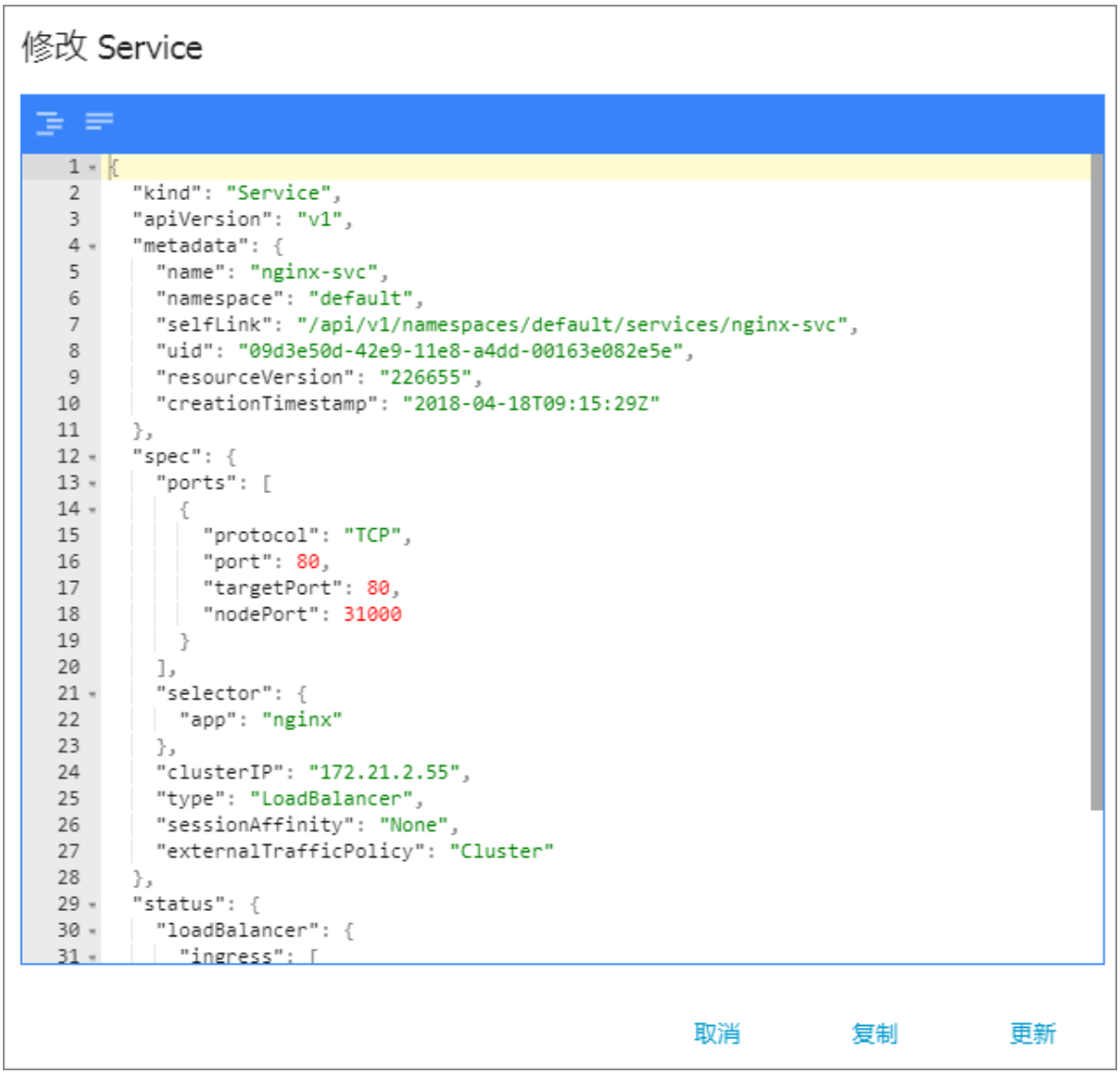


通過 Kubernetes Dashboard

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的控制台，進入 Kubernetes Dashboard。

4. 在 Kubernetes Dashboard 中，選擇所需的命名空間，單擊左側導覽列中的服務。
5. 選擇所需的服務，單擊右側的操作表徵圖並單擊查看/編輯**YAML**。

6. 在彈出的更新對話方塊中，修改配置資訊，如將 nodePort 修改為 **31000**，然後單擊更新。



1.6.9 删除服務

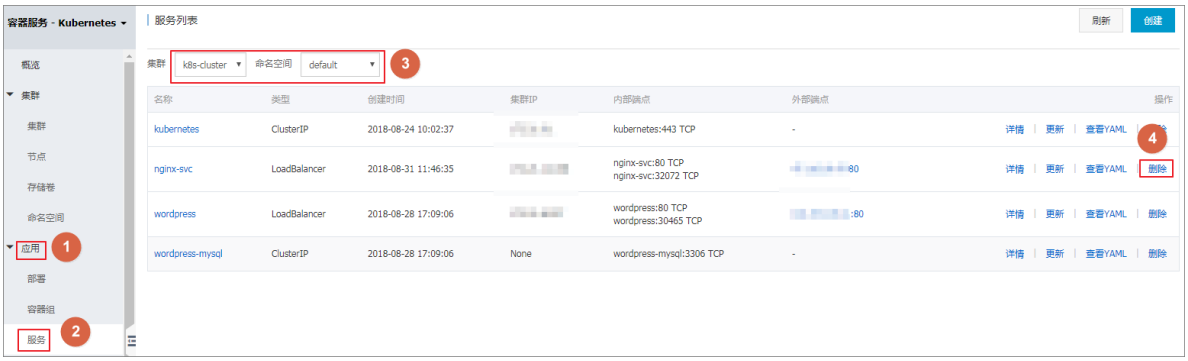
阿里雲 Kuberntes 服務提供 Web 介面，讓您快速對服務進行刪除。

前提条件

- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已經成功建立一個服務，參見*####*。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 服務，進入服務列表頁面。
3. 選擇叢集和命名空間，選擇所需的服務（本樣本中選擇 nginx-svc），單擊右側的刪除。



4. 在彈出的視窗中，單擊確定，確認刪除，該服務在服務列表中消失。

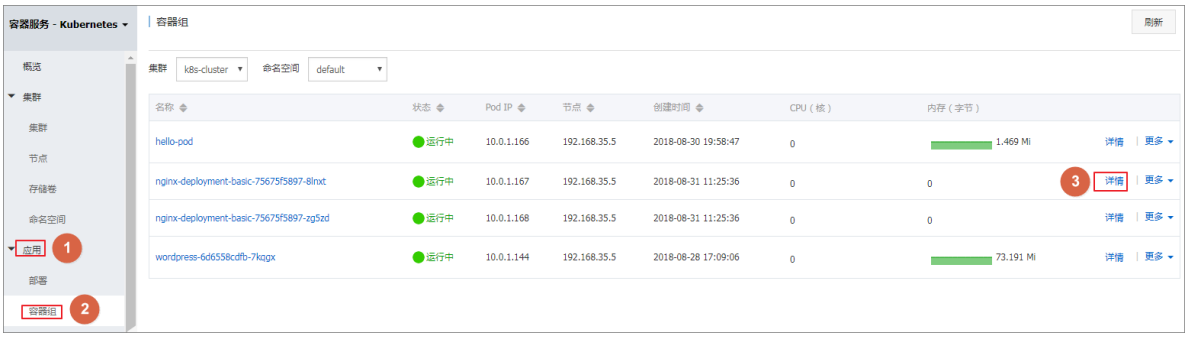
1.6.10 查看容器

您可以通過Container Service管理主控台的容器組頁面或者 Kubernetes Dashboard 查看 Kubernetes 叢集的容器組頁面。

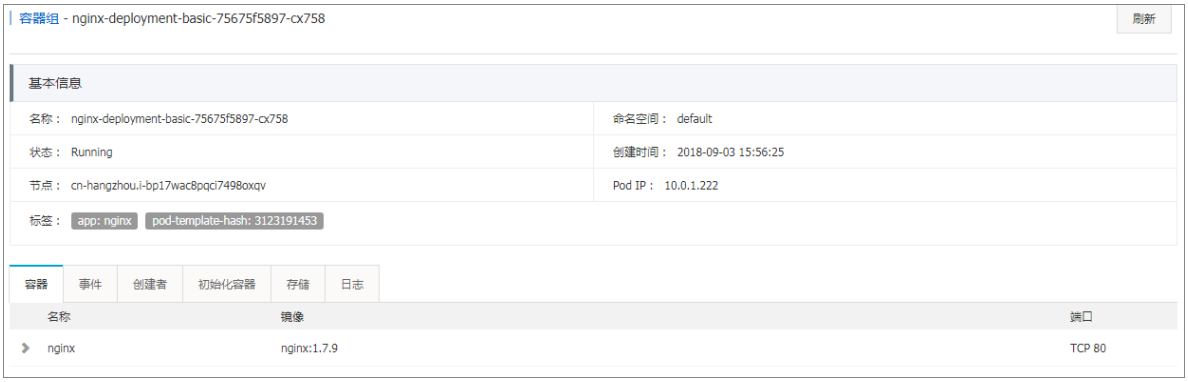
通過容器組頁面進行查看

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 容器組，進入容器組頁面。
3. 選擇所需的叢集和命名空間，選擇所需的容器組，單擊右側的詳情。

> 说明：
> 您可對容器組進行更新和刪除操作，對於通過部署（deployment）建立的容器組，建議您通過 deployment 進行管理。



4. 進入容器組的詳情頁，您可查看該容器組的詳情資訊。

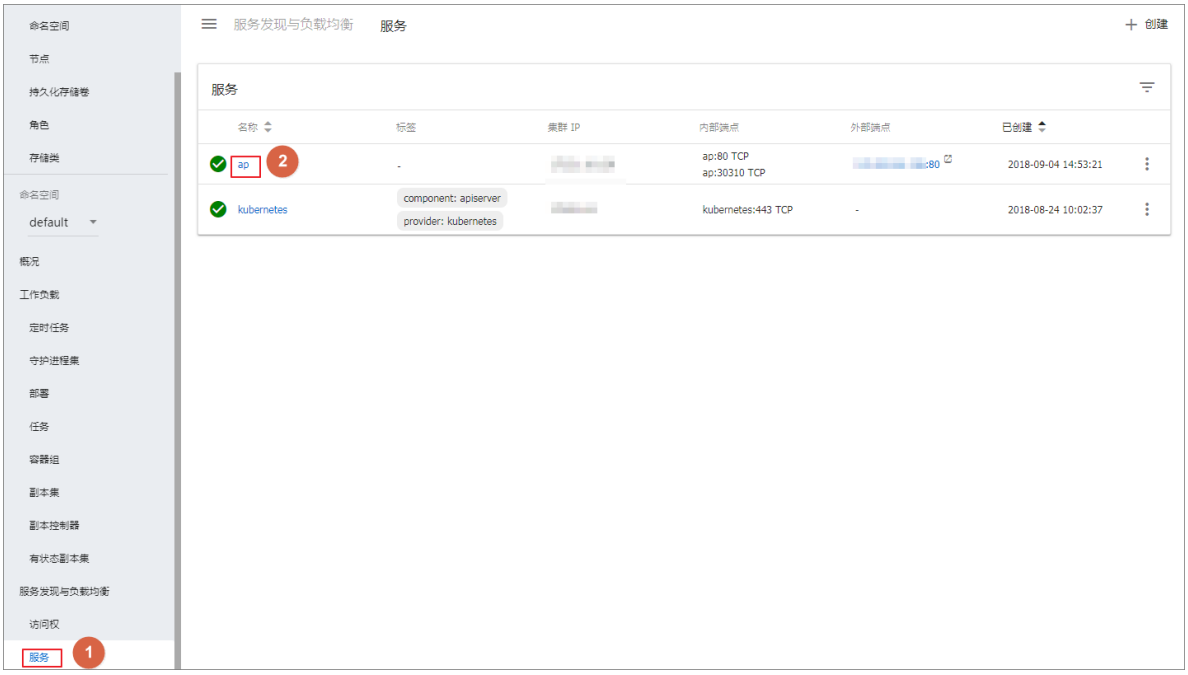


通過 Kubernetes Dashboard 查看容器組

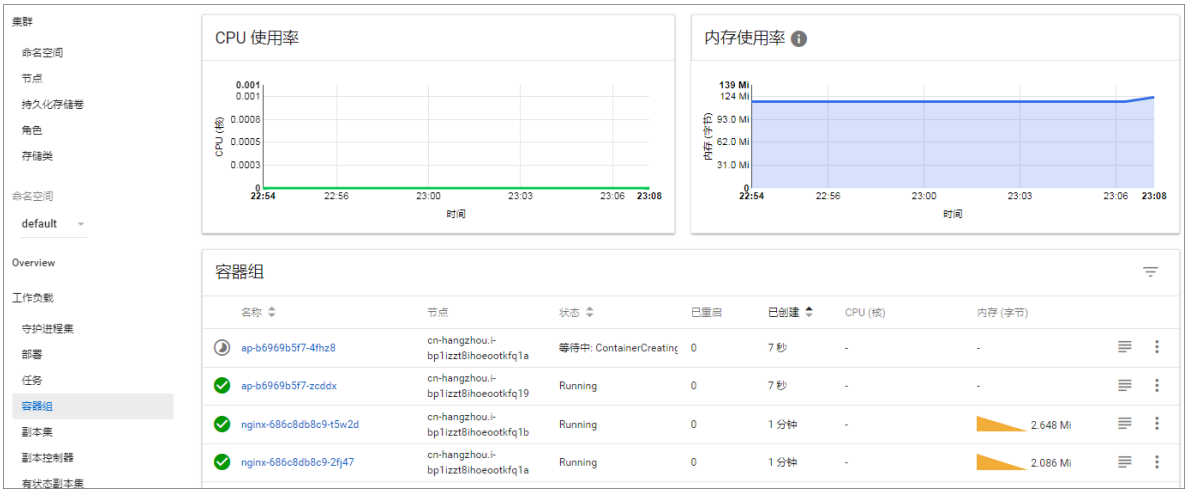
1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的控制台，進入 Kubernetes Dashboard。

4. 單擊左側導覽列中的容器組，查看叢集中的容器組。

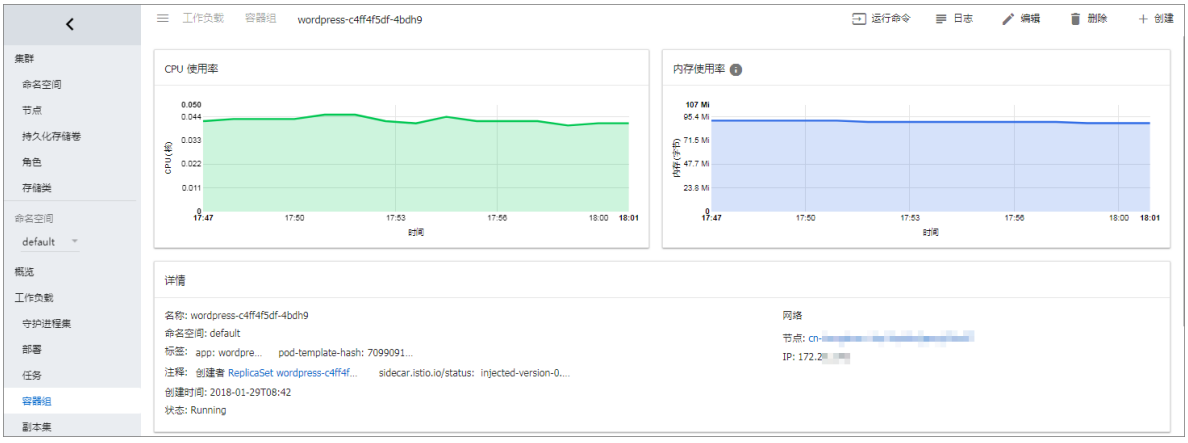
   或者，您可以單擊左側導覽列中的服務，選擇所需的服務並單擊服務的名稱，查看該服務下的容器組。



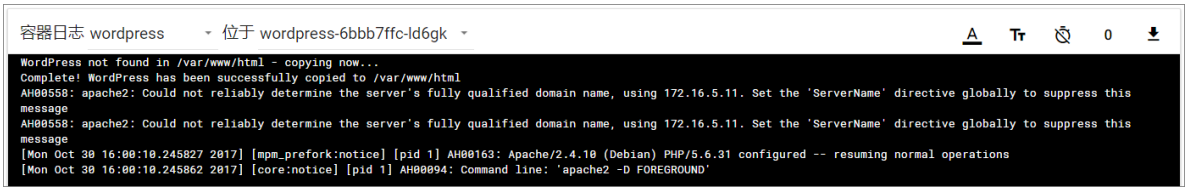
5. 您可以通過左側的表徵圖查看每個 Kubernetes 對象的狀態。 表示對象仍然處於部署狀態。 表示對象已經完成部署。



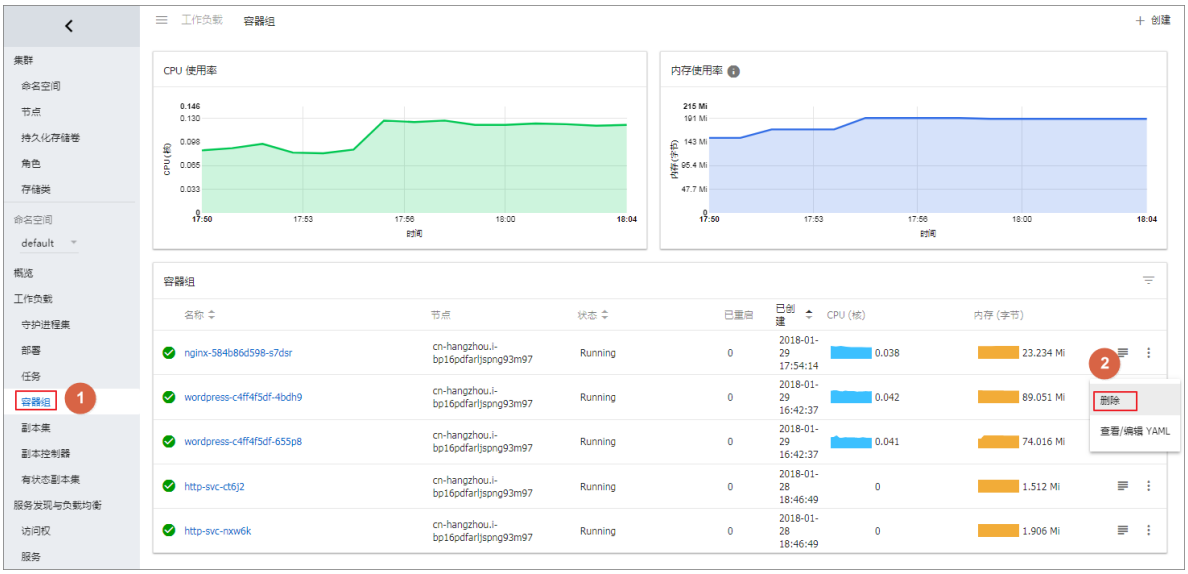
6. 選擇所需的容器組，單擊容器組的名稱，您可以查看容器組的詳細資料以及容器組使用的 CPU 和 Memory。



7. 選擇所需的容器組，單擊右上方的日誌查看容器的日誌資訊。



8. 您可以單擊右側的刪除 刪除該容器組。

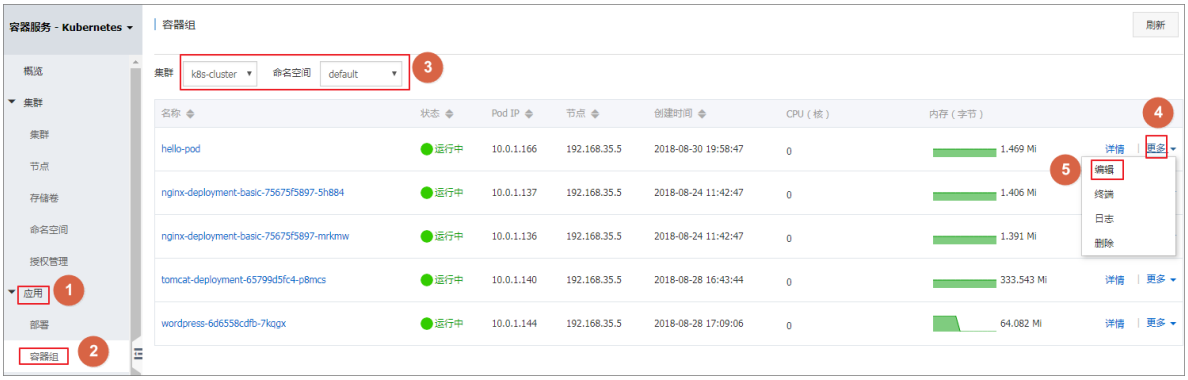


1.6.11 變更容器配置

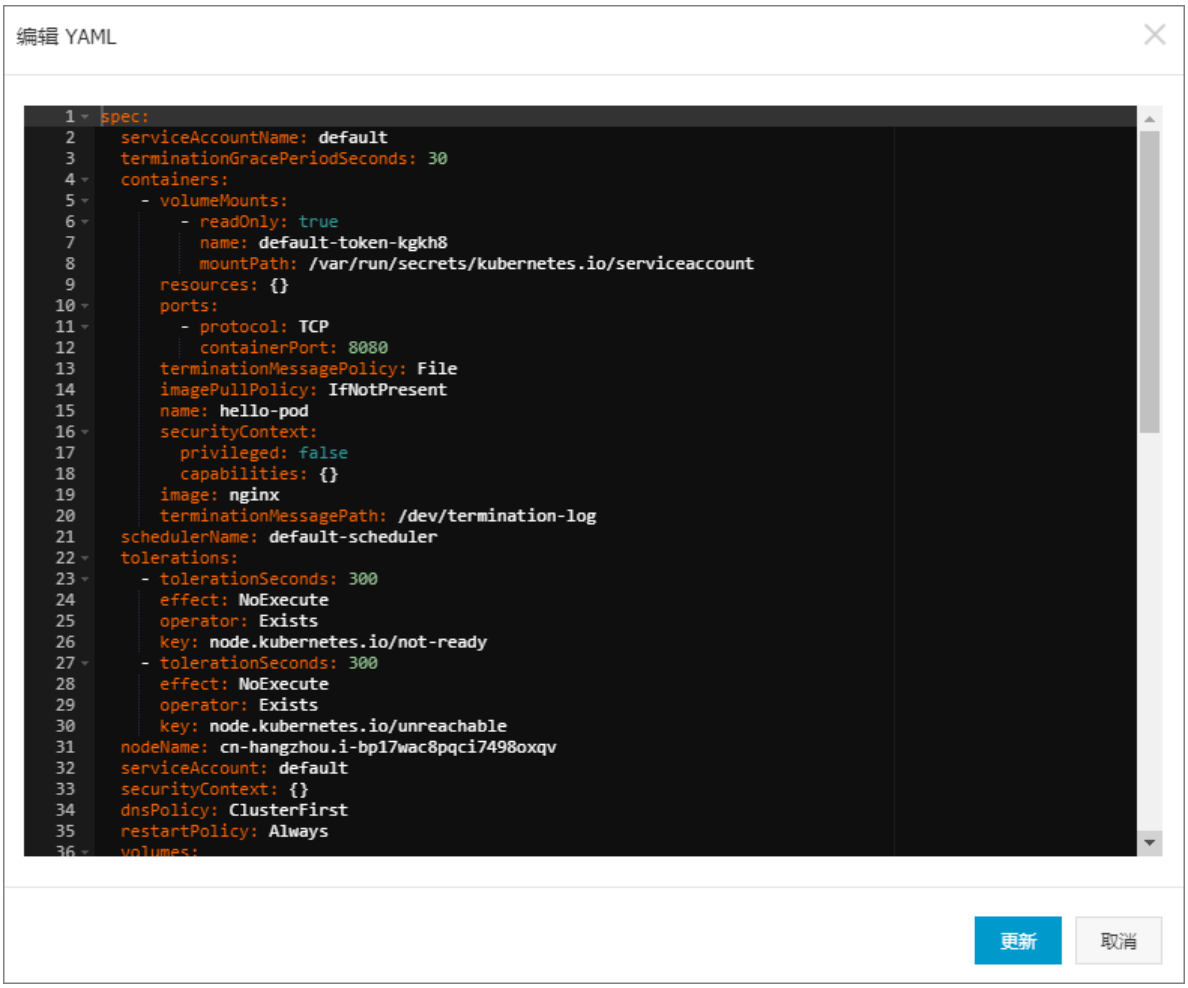
您可以通過Container Service管理主控台變更容器組（Pod）的配置。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 容器組，進入容器組列表。
3. 選擇所需的叢集和命名空間，選擇所需的容器組，單擊右側的更多 > 編輯。



4. 更新容器組的配置並單擊更新。



1.6.12 指定節點調度

您可通過為節點設定節點標籤，然後通過配置 `nodeSelector` 對 pod 調度進行強制限制式，將 pod 調度到指定的 Node 節點上。關於 nodeSelector 的詳細實現原理，請參見 *nodeselector*。

出於業務情境需要，如您需要將管控服務部署到 Master 節點；或者將某些服務部署到具有 SSD 盤的機器上。您可以採用這種方式實現指定節點調度。

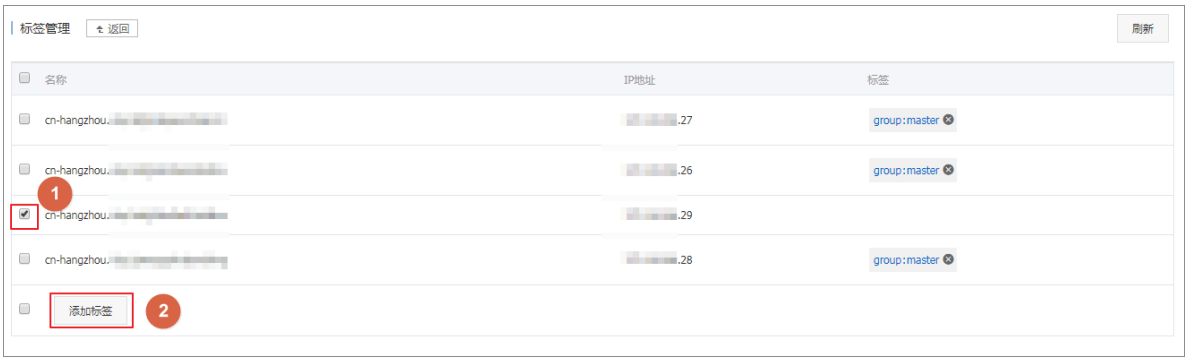
前提條件

您已經成功部署一個 Kubernetes 叢集，參見*##Kubernetes##*。

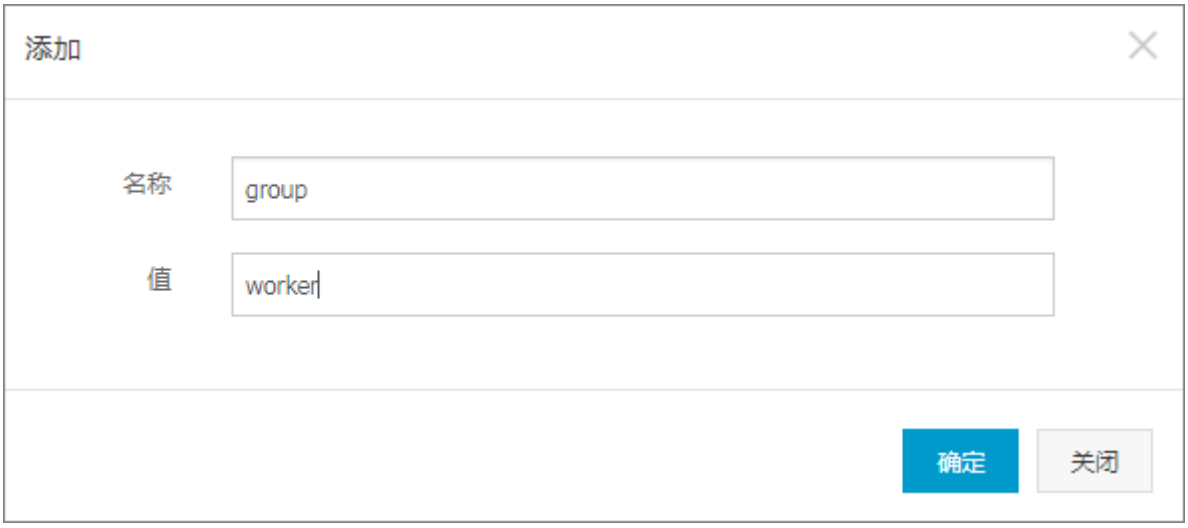
步驟1 為節點添加標籤

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 節點 ，進入節點列表頁面。
3. 選擇所需的叢集，在頁面右上方單擊標籤管理。

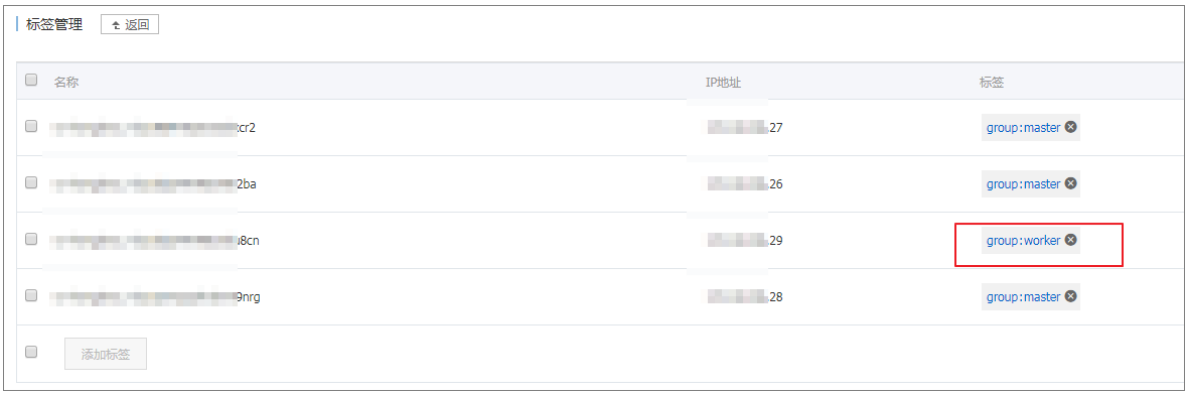
4. 在節點列表中，選擇所需節點，然後單擊添加標籤。本例中選擇一個 worker 節點。



5. 在彈出的添加標籤對話方塊中，輸入標籤的名稱和值，然後單擊確定。



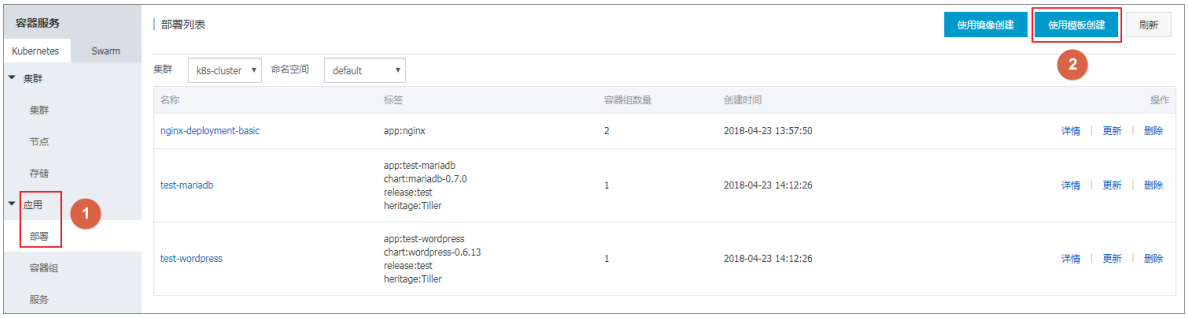
您可以在標籤管理頁面，看到該節點具有 `group:worker`標籤。



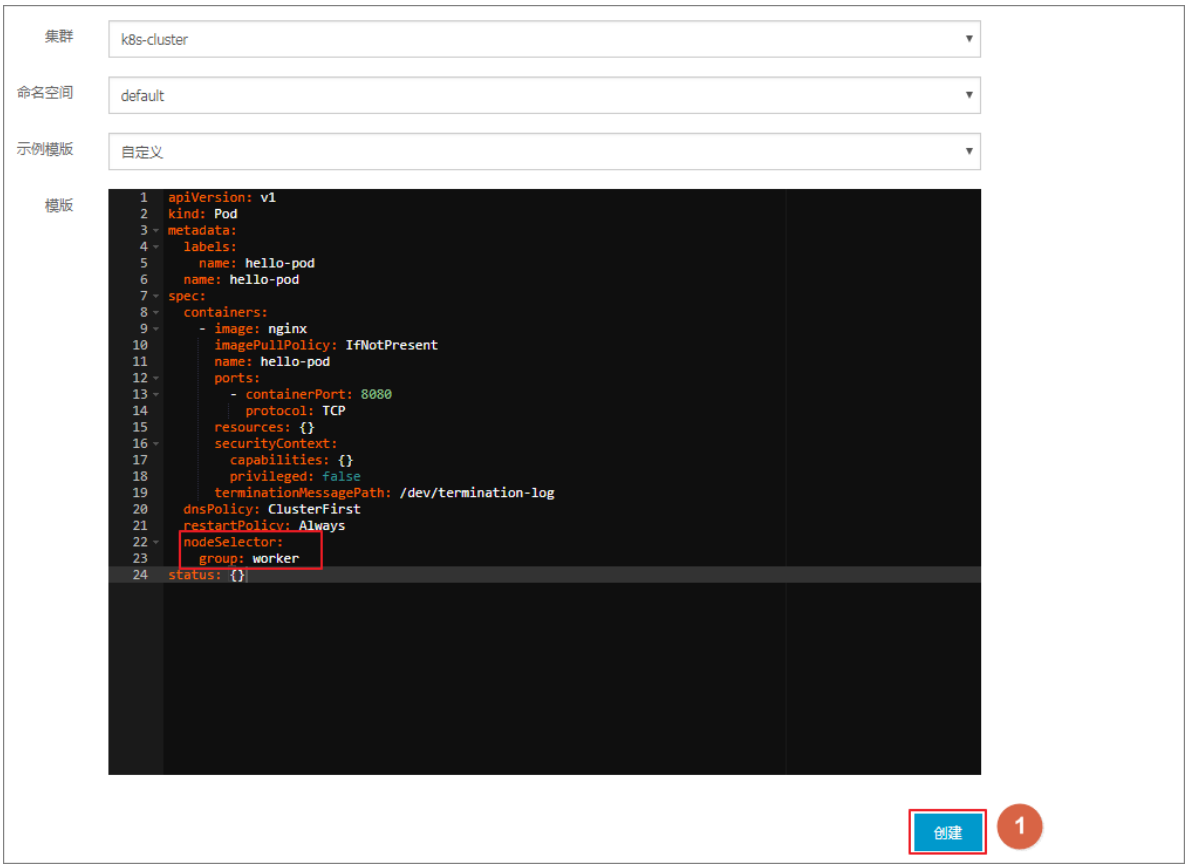
您也可以通過命令 `kubectl label nodes <node-name> <label-key>=<label-value>` 為節點添加標籤。

步驟2 將 pod 部署到指定節點

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署 ，進入部署列表頁面。
3. 單擊頁面右上方的使用模板建立。



4. 對模板進行相關配置，部署一個 pod，完成配置後，單擊建立。
   - 叢集：選擇所需的叢集。
   - 命名空間：選擇資來源物件所屬的命名空間，本例中是 default。
   - 樣本模板：本樣本選擇自訂模板。

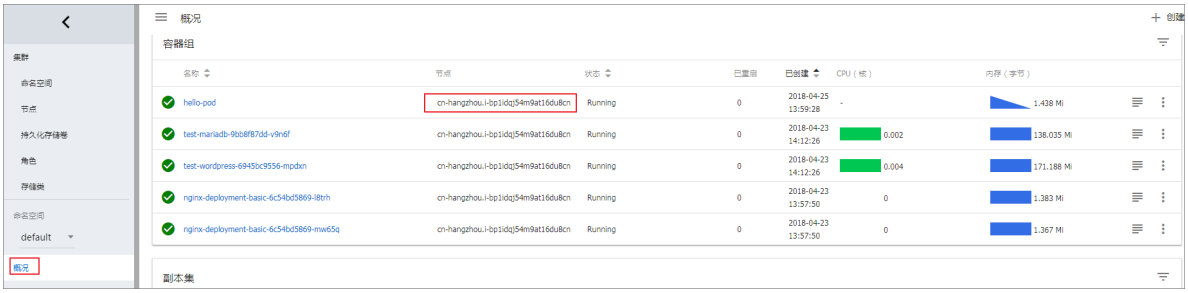


本樣本的編排模板如下。

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    name: hello-pod
  name: hello-pod
spec:
  containers:
    - image: nginx
      imagePullPolicy: IfNotPresent
      name: hello-pod
      ports:
        - containerPort: 8080
          protocol: TCP
      resources: {}
      securityContext:
        capabilities: {}
        privileged: false
      terminationMessagePath: /dev/termination-log
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  nodeSelector:
    group: worker                                ##注意與前面配置的節點標籤一致
```

```
  status: {}
```

5. 單擊建立 後，會提示部署狀態資訊。 成功後，單擊**Kubernetes** 控制台前往控制台查看部署狀態。



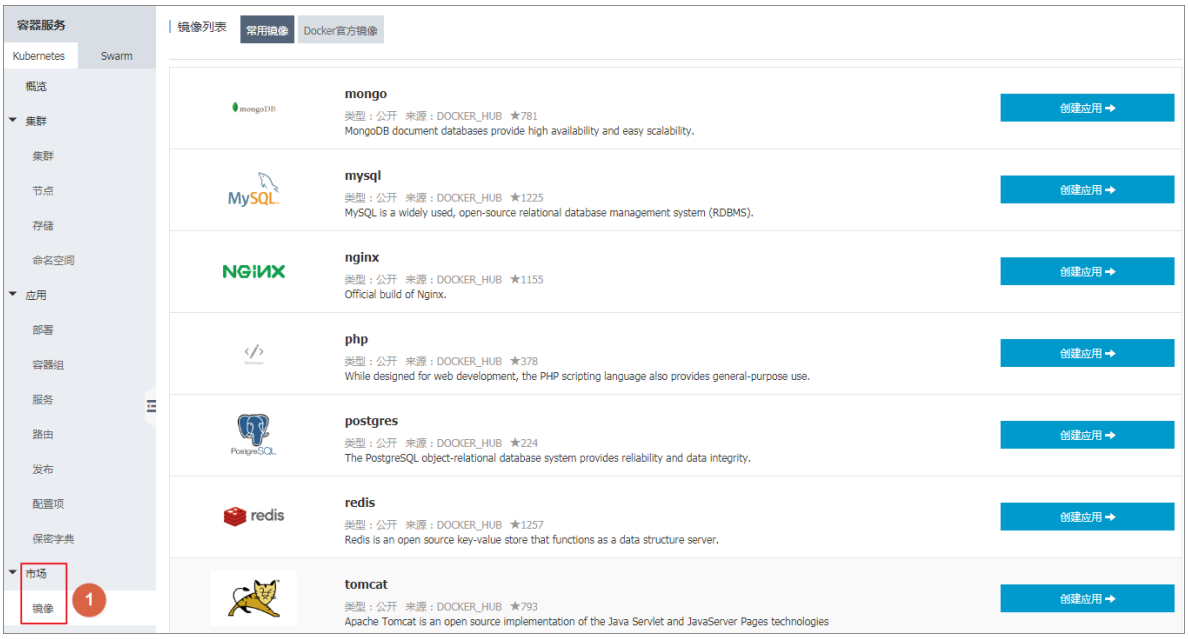
6. 您可單擊 pod 名稱，進入詳情頁，瞭解 pod 詳情。

   您可看到 pod 的標籤、所處的節點 ID 等資訊，表明該 pod 已經成功部署到具有 `group: worker` 標籤的指定節點上。

1.6.13 查看鏡像列表

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中市場 > 鏡像，進入鏡像列表頁面。



您可以查看鏡像的種類過。

- 常用鏡像： Container Service推薦的一些常用鏡像。
- **Docker** 官方鏡像： Docker Hub 提供的官方鏡像。

1.7 負載平衡及路由管理

1.7.1 概述

Kubernetes 叢集提供了多種訪問容器應用的方式，支援通過阿里雲的負載平衡服務（Server Load Balancer）或者 Ingress 方式來訪問內部服務的和實現負載平衡。

1.7.2 通過負載平衡（Server Load Balancer）訪問服務

您可以使用阿里雲負載平衡來訪問服務。

說明：

如果您的叢集的cloud-controller-manager版本大於等於v1.9.3，對於指定已有SLB的時候，系統預設不再為該SLB處理監聽，使用者需要手動設定該SLB的監聽規則。

執行以下命令，可查看cloud-controller-manager的版本。

```
root@master # kubectl get po -n kube-system -o yaml|grep image:|grep
cloud-con|uniq
```

```
  image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/cloud-controller-
manager-amd64:v1.9.3
```

通過命令列操作

1. 通過命令列工具建立一個 Nginx 應用。

```
root@master # kubectl run nginx --image=registry.aliyuncs.com/acs/
netdia:latest
root@master # kubectl get po
NAME                                READY     STATUS    RESTARTS
  AGE
nginx-2721357637-dvwq3              1/1       Running   1
  6s
```

2. 為 Nginx 應用建立阿里雲負載平衡服務，指定 `type=LoadBalancer` 來向外網使用者暴露 Nginx 服務。

```
root@master # kubectl expose deployment nginx --port=80 --target-
port=80 --type=LoadBalancer
root@master # kubectl get svc
NAME                  CLUSTER-IP      EXTERNAL-IP      PORT(S)
                 AGE
nginx                 172.19.10.209   101.37.192.20   80:31891/TCP
                4s
```

3. 在瀏覽器中訪問 `http://101.37.192.20`，來訪問您的 Nginx 服務。

通過 **Kubernetes Dashboard** 操作

1. 將下面的 yml code 儲存到 `nginx-svc.yml`檔案中。

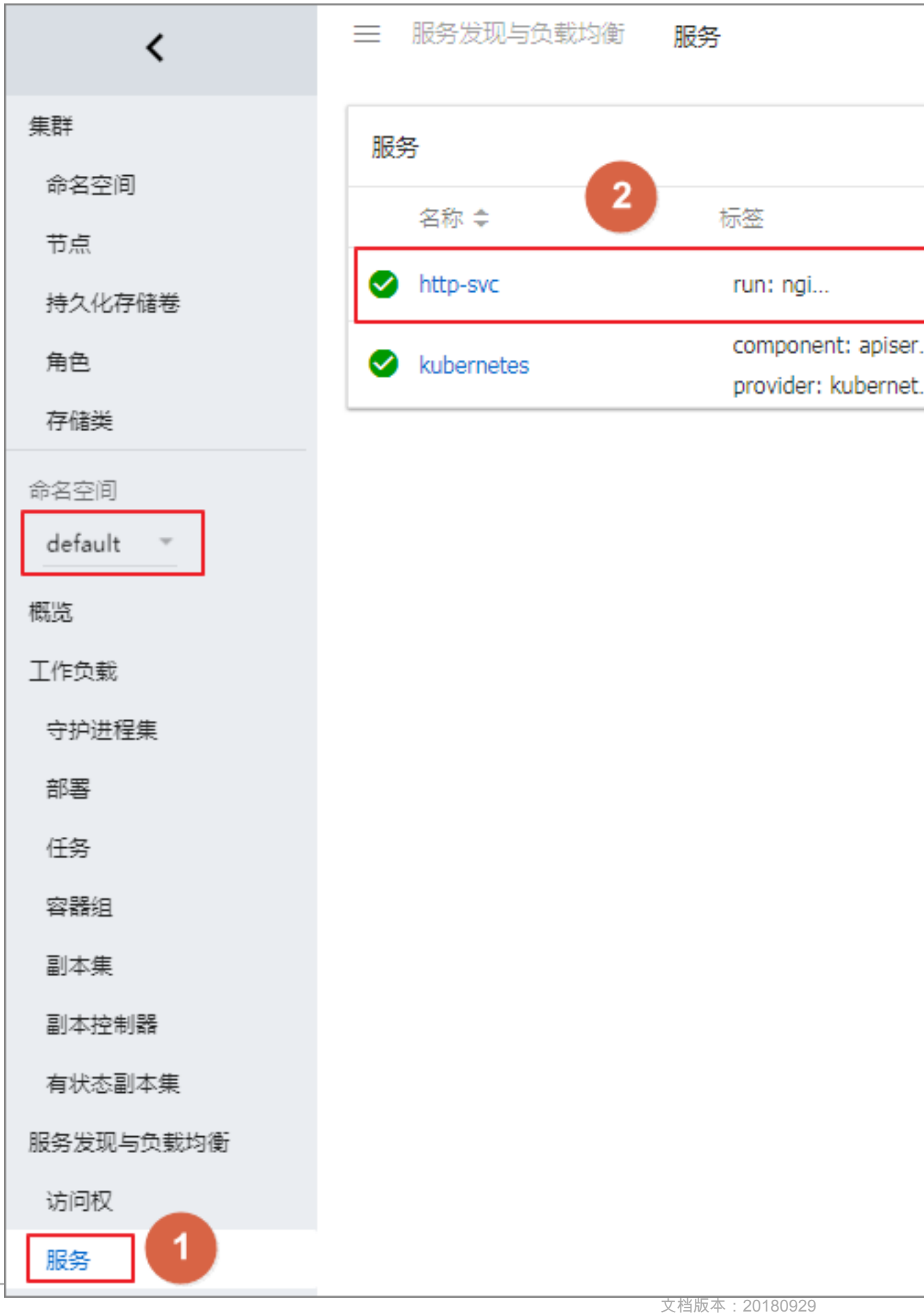
```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: http-svc
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

2. 登入*Container Service#####*，單擊目的地組群右側的控制台，進入 Kubernetes Dashboard 頁面。
3. 單擊建立，開始建立應用。

4. 單擊使用檔案建立。選擇剛才儲存的*nginx-svc.yml* 檔案
5. 單擊上傳。

   這樣會建立一個阿里雲Server Load Balancer執行個體指向建立的 Nginx 應用，服務的名稱為 `http-svc`。
6. 在 Kubernetes Dashboard 上定位到 default 命名空間，選擇服務。

   可以看到剛剛建立的 `http-svc` 的 Nginx 服務和機器的負載平衡地址 `http://114.55.79.24:80`。


集群
命名空间
节点
持久化存储卷
角色
存储类
命名空间
default
概览
工作负载
守护进程集
部署
任务
容器组
副本集
副本控制器
有状态副本集
服务发现与负载均衡
访问权
服务
1
服务发现与负载均衡
服务
服务
2
名称
标签
http-svc
run: ngi...
kubernetes
component: apiser.
provider: kubernet.

7. 將該地址拷貝到瀏覽器中即可訪問該服務。

更多資訊

阿里雲負載平衡還支援豐富的配置參數，包含健全狀態檢查、收費類型、負載平衡類型等參數。詳細資料參見#########。

注釋

阿里雲可以通過注釋的形式支援豐富的負載平衡功能。

使用已有的內網 **SLB**

需要指定兩個annotation。注意修改成您自己的 Loadbalancer-id。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type:
intranet
    service.beta.kubernetes.io/alicloud-loadbalancer-id: your-
loadbalancer-id
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

然後儲存為 slb.svc 後，執行 `kubectl apply -f slb.svc`。

建立 **HTTPS** 類型的 **Loadbalancer**

先在阿里雲控制台上建立一個認證並記錄 cert-id，然後使用如下 annotation 建立一個 HTTPS 類型的 SLB。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: your-
cert-id
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
https:443"
  labels:
    run: nginx
  name: nginx
```

```
  namespace: default
spec:
  ports:
  - name: web
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

说明：

注釋的內容是區分大小寫。

注釋	描述	預設值
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	多個值之間由逗號分隔，比如：https:443,http:80	無
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	取值可以是 internet 或者 intranet	internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	負載平衡的網路類型，取值可以是 classic 或者 vpc	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	取值可以是 paybytraffic 或者 paybybandwidth	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	Server Load Balancer執行個體的 ID。通過 loadbalancer-id 指定您已有的 SLB，已有 listener 會被覆蓋， 刪除 service 時該 SLB 不會被刪除。	無
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	通過 label 指定 SLB 後端掛哪些節點。	無
service.beta.kubernetes.io/alicloud-loadbalancer-region	負載平衡所在的地區	無
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	負載平衡的頻寬	50

注釋	描述	預設值
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	阿里雲上的認證 ID。您需要先上傳認證	“”
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	取值是 on 或者 off	預設為 off。TCP 不需要改參數。因為 TCP 預設開啟健全狀態檢查，使用者不可設定。
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	參見*CreateLoadBalancerTCPListener*	無
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	參見*CreateLoadBalancerTCPListener*	無

1.7.3 Ingress 支援

在 Kubernetes 叢集中，Ingress是授權入站串連到達叢集服務的規則集合，為您提供七層負載平衡能力。您可以給 Ingress 配置提供外部可訪問的 URL、負載平衡、SSL、基於名稱的虛擬機器主機等。

前置條件

為了測試複雜路由服務，本例中建立一個 nginx 的樣本應用，您需要事先建立 nginx 的 deployment，然後建立多個 Service，用來觀察路由的效果。實際測試請替換成自己的服務。

```
root@master # kubectl run nginx --image=registry.cn-hangzhou.aliyuncs.
com/acs/netdia:latest

root@master # kubectl expose deploy nginx --name=http-svc --port=80 --
target-port=80
root@master # kubectl expose deploy nginx --name=http-svc1 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc2 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc3 --port=80
 --target-port=80
```

簡單的路由服務

通過以下命令建立一個簡單的 Ingress，所有對 */svc* 路徑的訪問都會被路由到名為 http-svc 的服務。`nginx.ingress.kubernetes.io/rewrite-target: /`會將*/svc*路徑重新導向到後端服務能夠識別的/路徑上面。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
simple          *             101.37.192.211   80        11s
```

現在訪問 `http://101.37.192.211/svc` 即可訪問到 Nginx 服務。

基於網域名稱的簡單扇出路由

如果您有多個網域名稱對外提供不同的服務，您可以產生如下的配置達到一個簡單的基於網域名稱的扇出效果。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout
```

```
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
  - host: foo.example.com
    http:
      paths:
      - path: /film
        backend:
          serviceName: http-svc3
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
simple-fanout   *             101.37.192.211   80        11s
```

這時您可以通過 `http://foo.bar.com/foo` 訪問到 `http-svc1` 服務；通過 `http://foo.bar.com/bar` 訪問到 `http-svc2` 服務；通過 `http://foo.example.com/film` 訪問到 `http-svc3` 服務。

 说明：

- 如果是生產環境，您需要將您的這個網域名稱指向上面返回的 ADDRESS `101.37.192.211`。
- 如果是測試環境測試，您可以修改 `hosts` 檔案添加一條網域名稱映射規則。

```
101.37.192.211 foo.bar.com
101.37.192.211 foo.example.com
```

簡單路由預設網域名稱

如果您沒有網域名稱地址也沒有關係，Container Service為 Ingress 服務綁定了一個預設網域名稱，您可以通過這個網域名稱來訪問服務。網域名稱的格式如下：`*.[cluster-id].[region-id].alicontainer.com`。您可以直接在控制台叢集基本資料頁擷取到該地址。

您可以通過下面的配置藉助該預設網域名稱暴露兩個服務。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: shared-dns
spec:
  rules:
```

```
  - host: foo.[cluster-id].[region-id].alicontainer.com  ##替換為您叢集
預設的服務訪問網域名稱
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc1
          servicePort: 80
  - host: bar.[cluster-id].[region-id].alicontainer.com  ##替換為您叢集
預設的服務訪問網域名稱
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
shared-dns   foo.[cluster-id].[region-id].alicontainer.com,bar.[
cluster-id].[region-id].alicontainer.com             47.95.160.171
80        40m
```

這時您可以通過 `http://foo.[cluster-id].[region-id].alicontainer.com/` 訪問到 `http-svc1` 服務；通過 `http://bar.[cluster-id].[region-id].alicontainer.com` 訪問到 `http-svc2` 服務。

配置安全的路由服務

支援多認證管理，為您的服務提供安全防護。

1. 準備您的服務憑證。

   如果沒有認證，可以通過下面的方法產生測試認證。

   > 说明：
   > 網域名稱與您的 Ingress 配置要一致。

   ```
   root@master # openssl req -x509 -nodes -days 365 -newkey rsa:2048 -
   keyout tls.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
   ```

   上面命令會產生一個認證檔案 *tls.crt*、一個私密金鑰檔案*tls.key*。

   然後用該認證和私密金鑰建立一個名為*foo.bar* 的 Kubernetes Secret。建立 Ingress 時需要引用這個 Secret。

   ```
   root@master # kubectl create secret tls foo.bar --key tls.key --cert
    tls.crt
   ```

2. 建立一個安全的 Ingress 服務。

   ```
   root@master # cat <<EOF | kubectl create -f -
   apiVersion: extensions/v1beta1
   ```

```
kind: Ingress
metadata:
  name: tls-fanout
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: foo.bar
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
tls-fanout      *             101.37.192.211   80        11s
```

3. 按照 基於網域名稱的簡單扇出路由 中的注意事項，配置 `hosts` 檔案或者設定網域名稱來訪問該 tls 服務。

   您可以通過 `http://foo.bar.com/foo` 訪問到 `http-svc1` 服務；通過 `http://foo.bar.com/bar` 訪問到 `http-svc2` 服務。

   您也可以通過 HTTP 的方式訪問該 HTTPS 的服務。Ingress 預設對配置了 HTTPS 的 HTTP 訪問重新導向到 HTTPS 上面。所以訪問 `http://foo.bar.com/foo` 會被自動重新導向到 `https://foo.bar.com/foo`。

通過 Kubernetes Dashboard 部署 Ingress

1. 將下面的 yml code 儲存到 `nginx-ingress.yml` 檔案中。

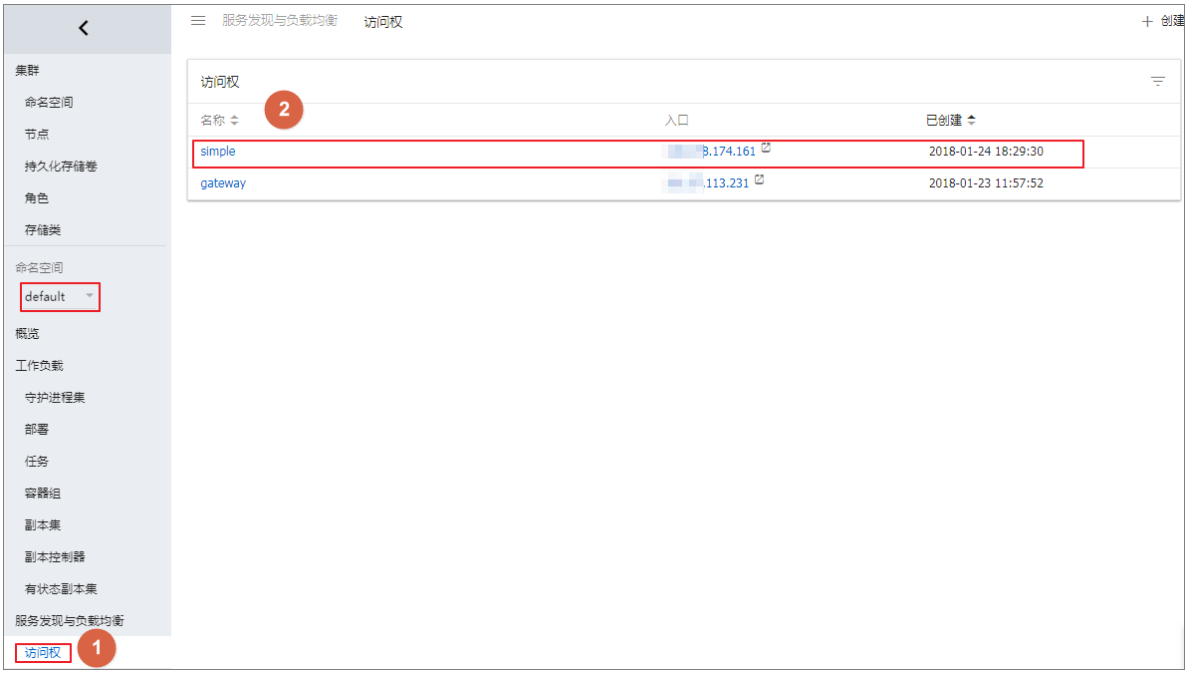
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
```

2. 登入*Container Service#####*，在 Kubernetes 菜單下，在叢集列表頁面中，單擊目的地組群右側的控制台，進入 Kubernetes Dashboard 頁面。
3. 單擊建立，開始建立應用。

4. 單擊使用檔案建立。選擇剛才儲存的 `nginx-ingress.yml` 檔案。
5. 單擊上傳。

   這樣就建立了一個 Ingress 的七層代理路由到 `http-svc` 服務上。
6. 在 Kubernetes Dashboard 上定位到 default 命名空間，選擇訪問權。

   可以看到您剛剛建立的 Ingress 資源及其訪問地址 `http://118.178.174.161/svc`。



7. 開啟瀏覽器輸入該地址即可訪問前面建立的 `http-svc` 服務。

1.7.4 通過 Web 介面建立路由

阿里雲Container Service Web 介面整合了路由（Ingress）服務，您可通過 Web 介面快速建立路由服務，構建靈活可靠的流量接入層。

前提條件

- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*，並且叢集中 Ingress controller 正常運行。

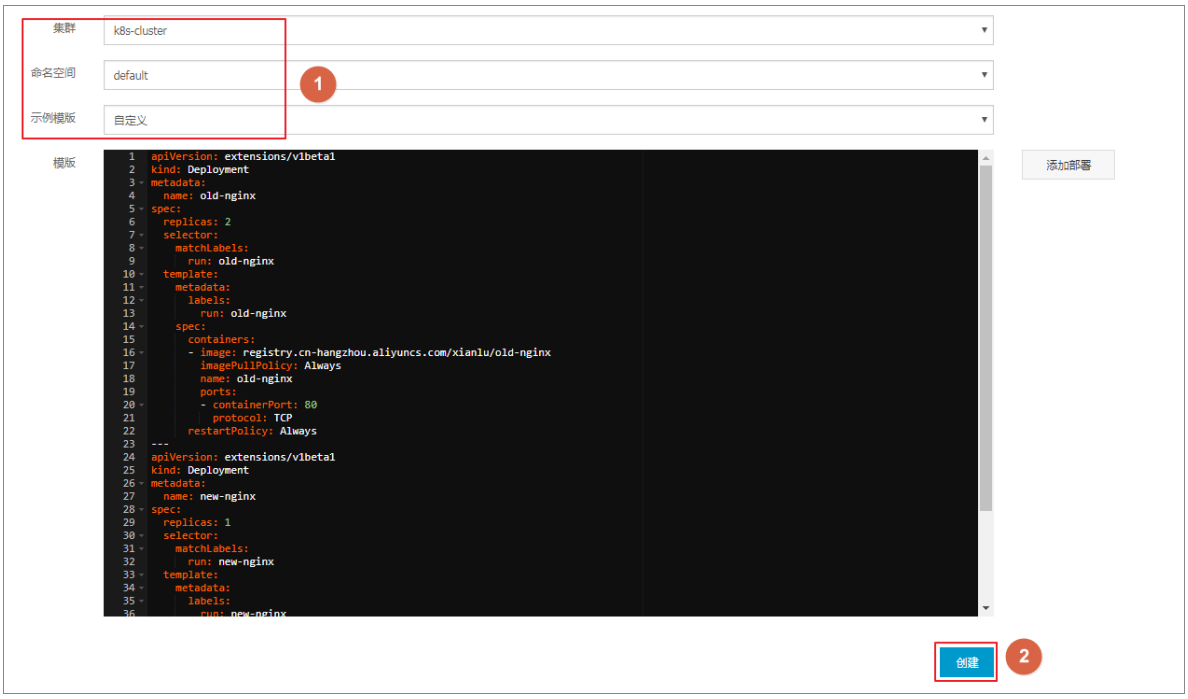
- SSH 登入到 Master 節點，參見*SSH##Kubernetes##*。

步驟1 建立 **deployment** 和服務

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面。
3. 單擊頁面右上方使用模板建立。

4. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

本例中，樣本中建立3個nginx應用，一個代表舊的應用old-nginx，一個代表新的應用 new-nginx，此外建立一個domain-nginx應用，用於測試叢集訪問網域名稱。



old-nginx的編排模板如下所示：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: old-nginx
spec:
```

```
  replicas: 2
  selector:
    matchLabels:
      run: old-nginx
  template:
    metadata:
      labels:
        run: old-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
        imagePullPolicy: Always
        name: old-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: old-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: old-nginx
  sessionAffinity: None
  type: NodePort
```

new-nginx的編排模板如下所示：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: new-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      run: new-nginx
  template:
    metadata:
      labels:
        run: new-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
        imagePullPolicy: Always
        name: new-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: new-nginx
```

```
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: new-nginx
  sessionAffinity: None
  type: NodePort
```

domain-nginx應用的編排模板如下所示：

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
  name: domain-nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <
image_name:tags>
        ports:
        - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: domain-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  sessionAffinity: None
  type: NodePort
```

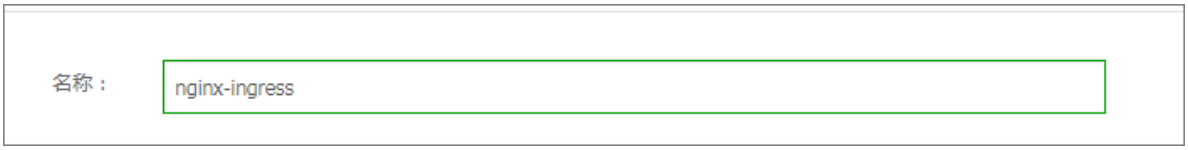
5. 單擊左側導覽列中的應用 > 服務 ，進入服務列表頁面。

   等待服務建立完成後，在服務列表，您可看到本樣本建立的服務。

步驟2 建立路由

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 路由，進入路由頁面。
3. 選擇所需的叢集和命名空間，單擊頁面右上方的建立。

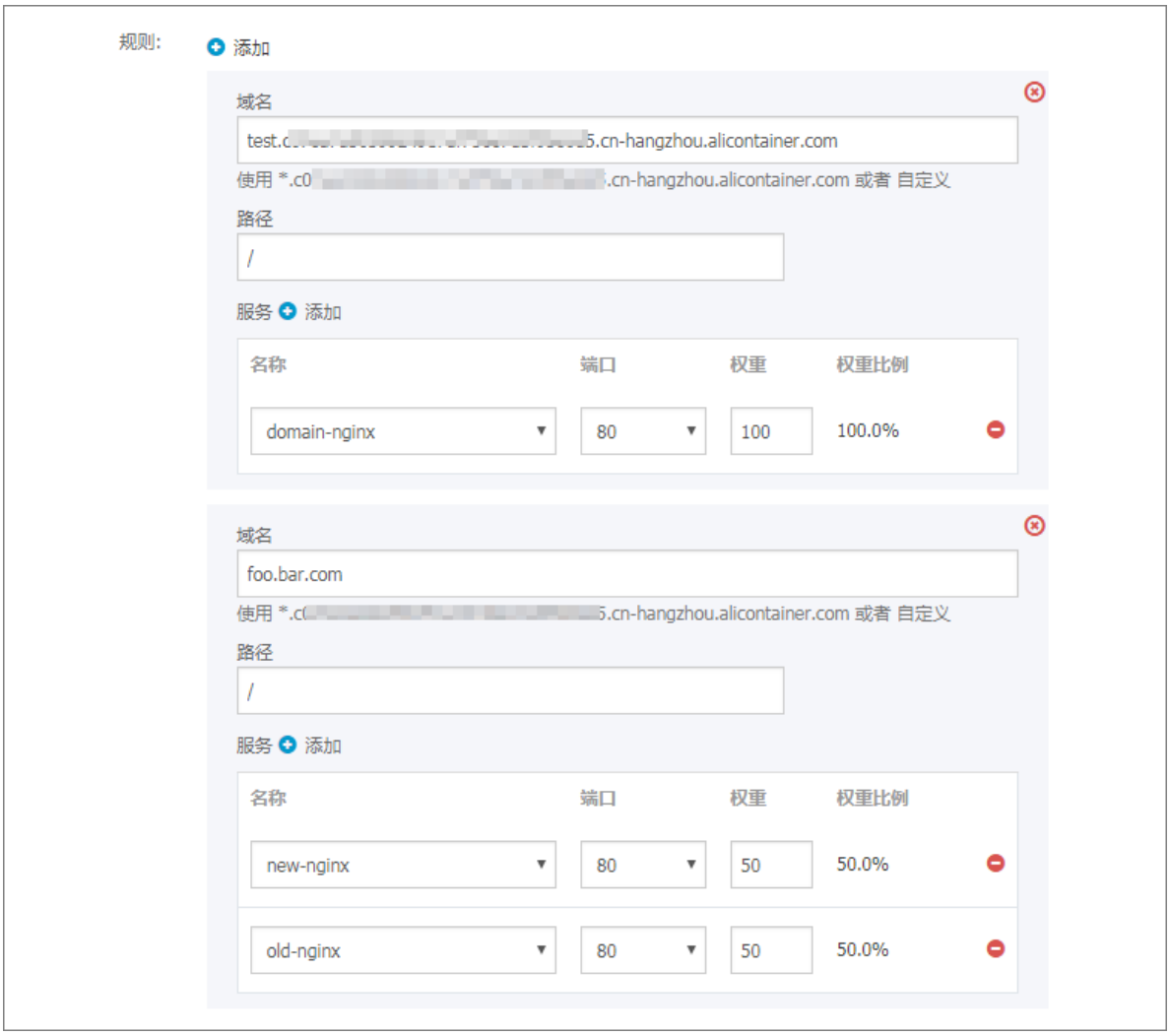
4. 在彈出的路由建立對話方塊中，首先配置路由名稱，本例為 nginx-ingress。



5. 對路由規則進行配置。

   路由規則是指授權入站到達叢集服務的規則，支援 http/https 規則，配置項包括網域名稱(虛擬機器主機名稱)、URL 路徑、服務名稱、連接埠配置和路由權重等。詳細的資訊請參見######。

   本例中配置添加一條複雜的路由規則，配置叢集預設的測試網域名稱和虛擬機器主機名稱，展示基於網域名稱的路由服務。



- 基於預設網域名稱的簡單路由，即使用叢集的預設網域名稱對外提供訪問服務。
  - 網域名稱配置：使用叢集的預設網域名稱，本例中是 `test.[cluster-id].[region-id].alicontainer.com`。

    在建立路由對話方塊中，會顯示該叢集的預設網域名稱，網域名稱格式是 `*.[cluster-id].[region-id].alicontainer.com`；您也可在叢集的基本資料頁面中擷取。
  - 服務配置：佈建服務的訪問路徑、名稱以及連接埠。
    - 訪問路徑配置：您可指定服務訪問的 URL 路徑，預設為根路徑/，本例中不做配置。每個路徑（path）都關聯一個 backend（服務），在阿里雲 SLB 將流量轉寄到 backend 之前，所有的入站請求都要先匹配網域名稱和路徑。
    - 服務配置：支援服務名稱、連接埠、服務權重等配置，即 backend 配置。同一個訪問路徑下，支援多個服務的配置，Ingress的流量會被切分，並被轉寄到它所匹配的 backend。

- 基於網域名稱的簡單扇出路由。本例中使用一個虛擬主機名稱作為測試網域名稱對外提供訪問服務，為兩個服務配置路由權重，並為其中一個服務設定灰階發布規則。若您在生產環境中，可使用成功備案的網域名稱提供訪問服務。
  - 網域名稱配置：本例中使用測試網域名稱 `foo.bar.com`。

    您需要修改 hosts 檔案添加一條網域名稱映射規則。

    ```
    118.178.108.143 foo.bar.com       #IP即是Ingress的Address
    ```

  - 服務配置：佈建服務的訪問路徑、服務名稱、服務連接埠和服務權重。
    - 訪問路徑配置：指定服務訪問的 URL 路徑。本例中不做配置，保留根路徑/。
    - 服務名稱：本例中設定新舊兩個服務 nginx-new 和 nginx-old 。
    - 服務連接埠：暴露80連接埠。
    - 權重設定：設定該路徑下多個服務的權重。服務權重採用相對值計算方式，預設值為100，如本例中所示，新舊兩個版本的服務權重值都是50，則表示兩個服務的權重比例都是50%。

6. 配置灰階發布。

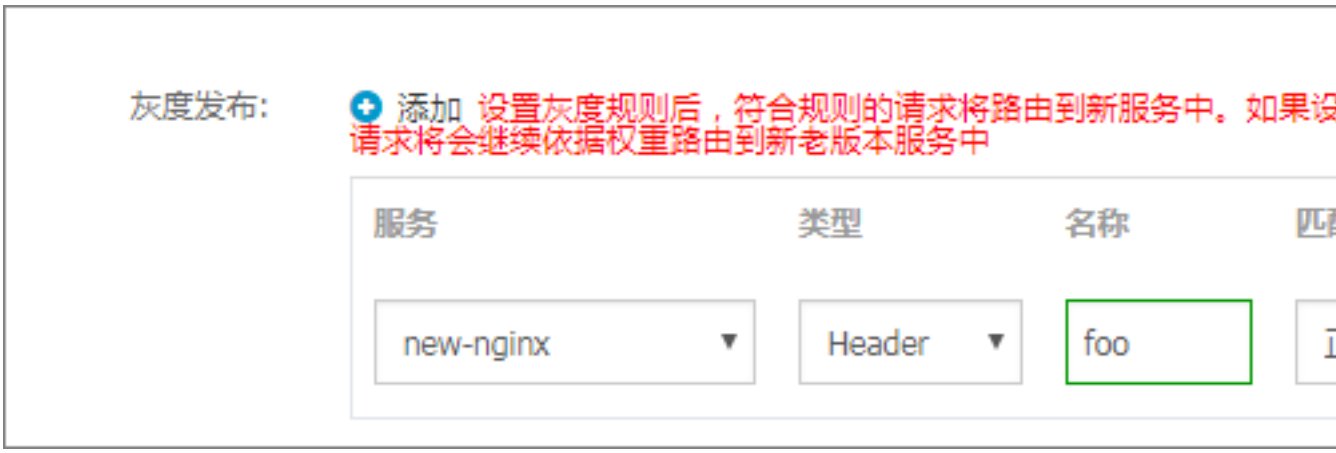
   说明：
   目前阿里雲Container ServiceKubernetes Ingress Controller需要**0.12.0-5**及其以上版本才支援流量切分特性。

   Container Service支援多種流量切分方式，適用於灰階發布以及AB測試情境。

   1. 基於Request Header的流量切分
   2. 基於Cookie的流量切分
   3. 基於Query Param的流量切分

   設定灰階規則後，要求標頭中滿足灰階發布匹配規則的請求才能被路由到新版本服務new-nginx中。如果該服務設定了100%以下的權重比例，滿足灰階規則的請求會繼續依據權重比例路由到對應服務。

   在本例中，設定Header要求標頭帶有`foo=^bar$`的灰階發布規則，僅帶有該要求標頭的用戶端請求才能訪問到new-nginx 服務。



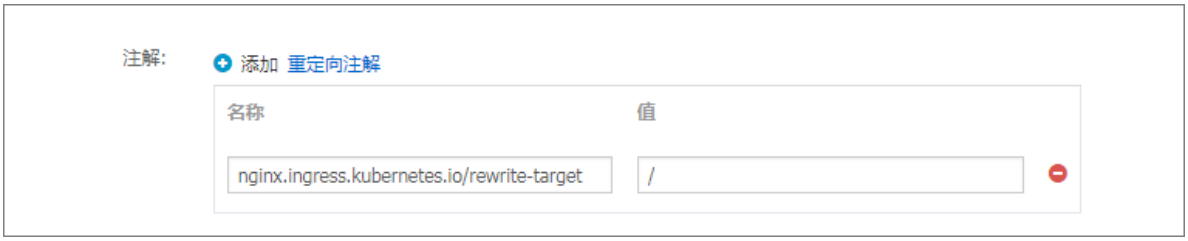
- 服務：路由規則配置的服務。
- 類型：支援Header（要求標頭）、Cookie和Query（請求參數）的匹配規則。
- 名稱和匹配值：使用者自訂的請求欄位，名稱和匹配值為索引值對。
- 匹配規則：支援正則匹配和完全符合。

7. 配置註解。

   單擊重新導向註解，可為路由添加一條典型的重新導向註解。即 `nginx.ingress.kubernetes.io/rewrite-target：/`，表示將*/path* 路徑重新導向到後端服務能夠識別的根路徑/上面。

   > 说明：
   > 本例中未對服務配置訪問路徑，因此不需要配置重新導向註解。重新導向註解的作用是使Ingress以根路徑轉寄到後端，避免訪問路徑錯誤配置而導致的404錯誤。

   您也可單擊添加按鈕，輸入註解名稱和值，即Ingress的annotation索引值對，Ingress的註解參見*https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/*。



8. 配置 TLS。勾選開啟 TLS，配置安全的路由服務。具體可參見#########。

   - 您可選擇使用已有密鑰。

1. 登入 master 節點，建立 *tls.key* 和 *tls.crt*。

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

2. 建立一個 secret。

```
kubectl create secret tls foo.bar --key tls.key --cert tls.crt
```

3. 執行命令 `kubectl get secret`，您可看到該 secret 已經成功建立。在 Web 介面可選擇建立的*foo.bar*這個 secret。

- 您可選擇在 TLS 介面上利用已建立的 TLS 私密金鑰和認證，一鍵建立 secret。

1. 登入 master 節點，建立 *tls.key* 和 *tls.crt*。

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

2. 執行 `vim tls.key` 和 `vim tls.crt` 擷取產生的私密金鑰和認證。
3. 將認證和私密金鑰的內容複寫到 TLS 建立密鑰的面板。

9. 添加標籤。

   標籤的作用是為Ingress添加對應的標籤，標示該Ingress的特點。

10. 最後單擊建立，返迴路由列表。

    等待一段時間，可以看到一條路由。

11.單擊路由中的訪問網域名稱 `test.[cluster-id].[region-id].alicontainer.com`，以及 `foo.bar.com` ，可訪問 nginx 的歡迎頁面。

單擊指向new-nginx服務的路由地址，發現指向了old-nginx應用的頁面。

> 说明：
> 在瀏覽器中訪問路由地址，預設情況下，要求標頭（Header）中沒有前面步驟中定義的`foo=^bar$`，因此流量會導向old-nginx應用。

12.SSH登入到Master節點，執行以下命令，類比帶有特定要求標頭的訪問結果。

```
curl -H "Host: foo.bar.com" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" http://47.107.20.35
 #類似於瀏覽器的訪問請求
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
     #類比帶有特有header的訪問請求，會根據路由權重返回結果
new
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
```

```
new
```

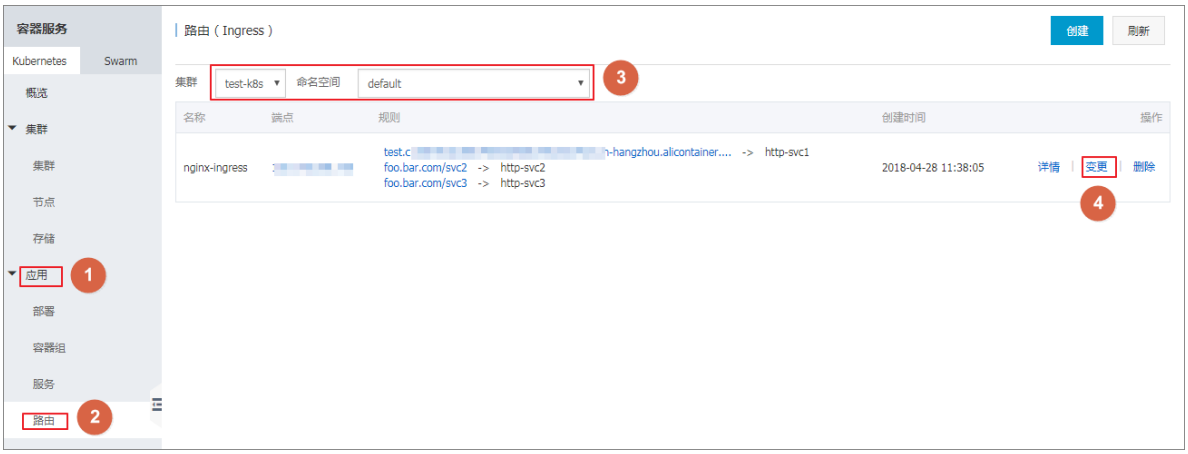
1.7.5 變更路由

前提条件

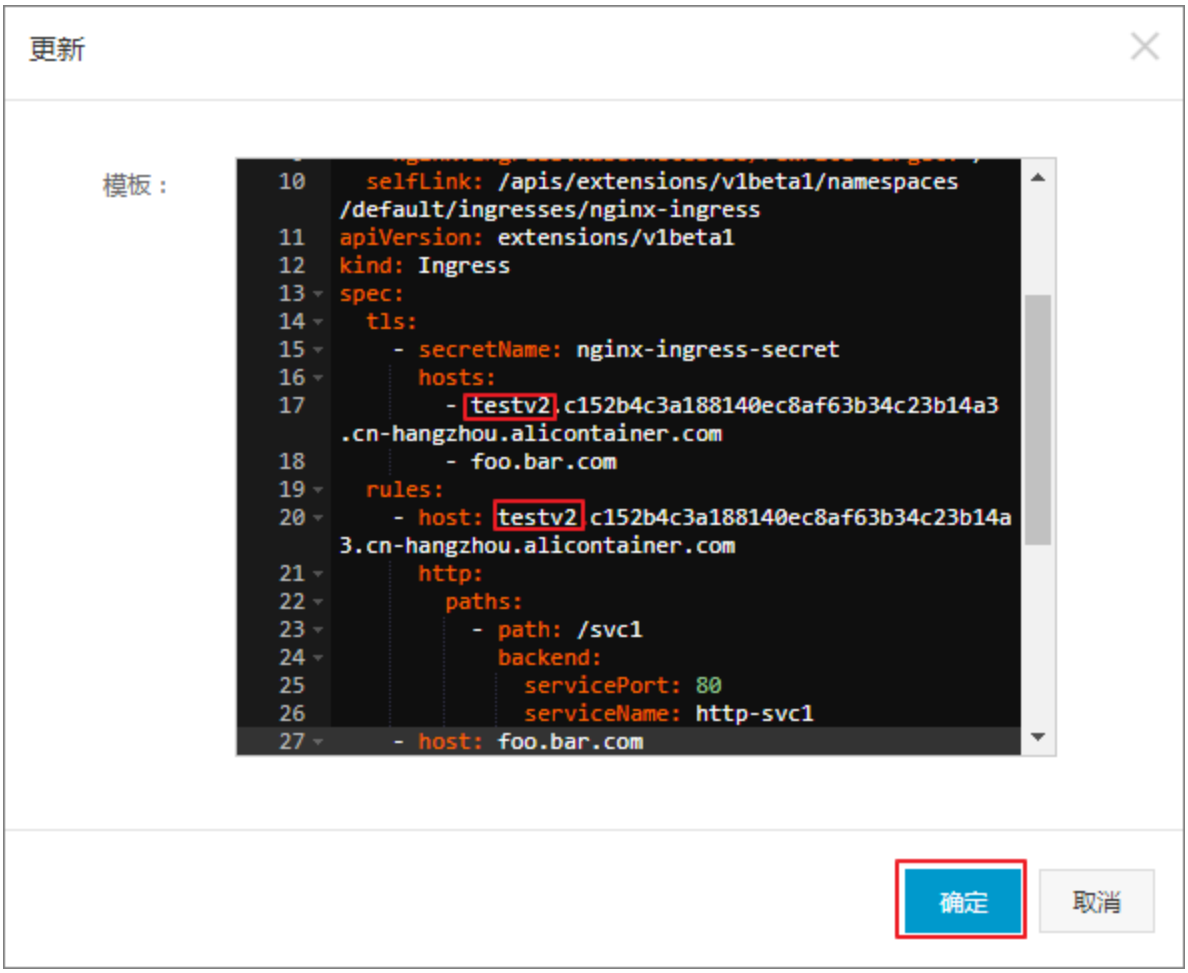
- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*，並且叢集中 Ingress controller 正常運行。
- 您已經成功建立一個路由，參見*## Web ######*。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 路由，進入路由頁面。
3. 選擇所需的叢集和命名空間，選擇所需的路由，然後單擊路由右側的變更。

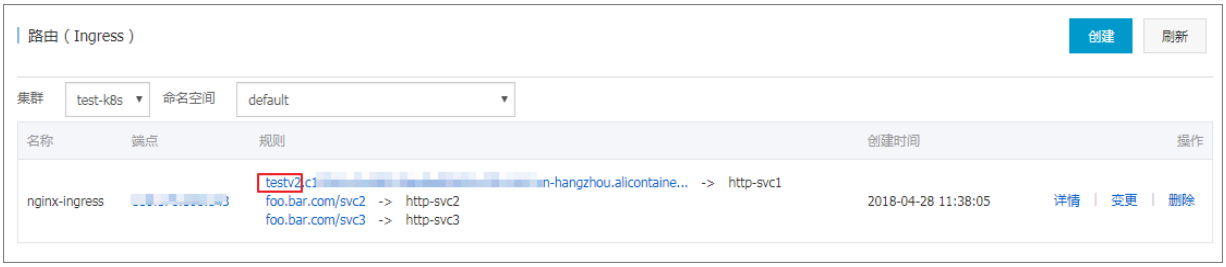


4. 在彈出的對話方塊中，對路由的相關參數進行變更，然後單擊確定 。本例中將 `test.[cluster-id].[region-id].alicontainer.com` 修改為 `testv2.[cluster-id].[region-id].alicontainer.com`。



后续操作

返迴路由列表，您可看到該路由的其中一條路由規則發生變化。



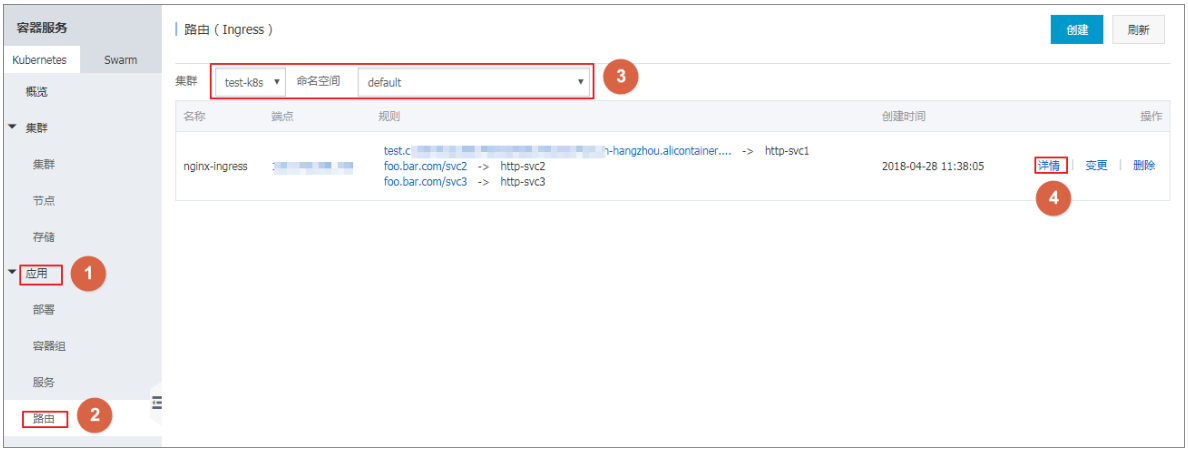
1.7.6 查看路由

前提条件

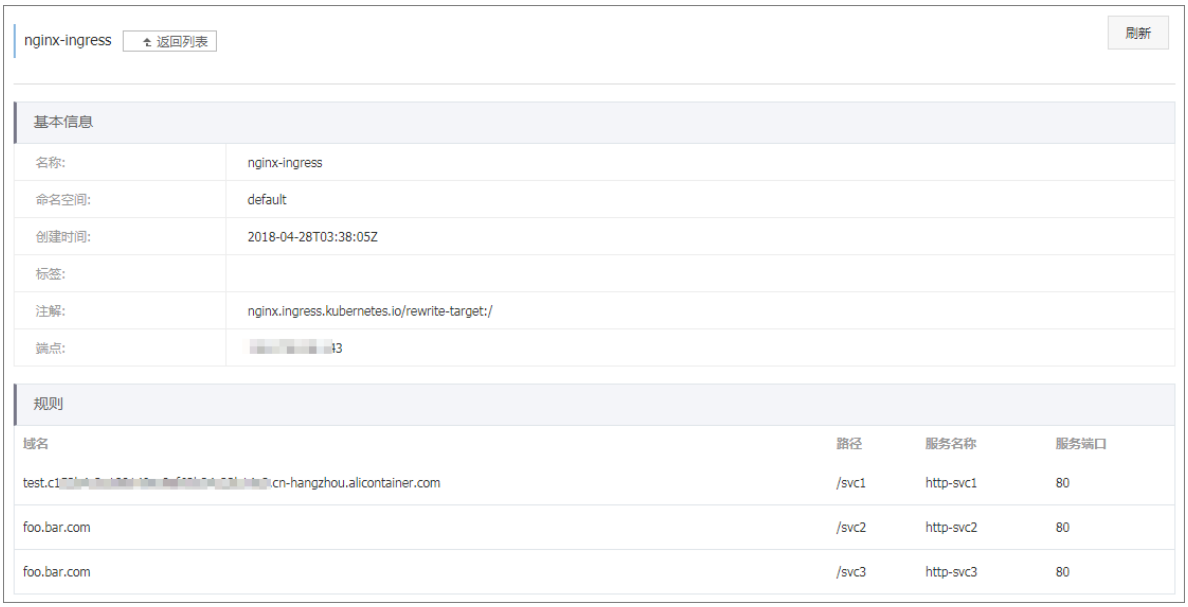
- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*，並且叢集中 Ingress controller 正常運行。
- 您已經成功建立一個路由，參見*## Web ######*。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的 應用 > 路由，進入路由頁面。
3. 選擇所需的叢集和命名空間，選擇所需的路由，然後單擊路由右側的詳情。



進入路由詳情頁面，您可查看該路由的基本資料以及路由規則。



1.7.7 刪除路由

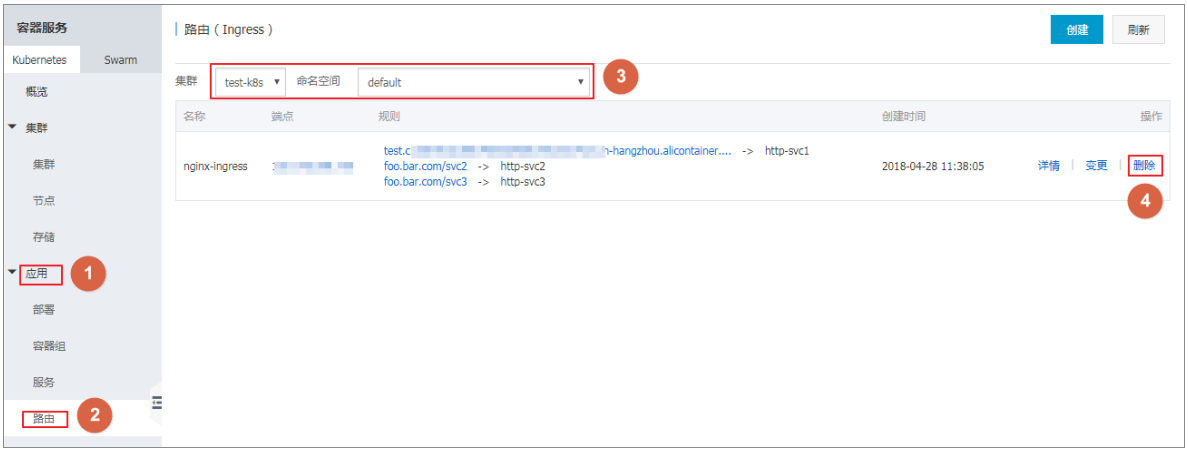
前提条件

- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*，並且叢集中 Ingress controller 正常運行。
- 您已經成功建立一個路由，參見*## Web ######*。

操作步驟

1. 登入*Container Service#####*。

2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 路由，進入路由頁面。
3. 選擇所需的叢集和命名空間，選擇所需的路由，然後單擊路由右側的刪除。



4. 在彈出的對話方塊中，單擊確定，完成刪除。

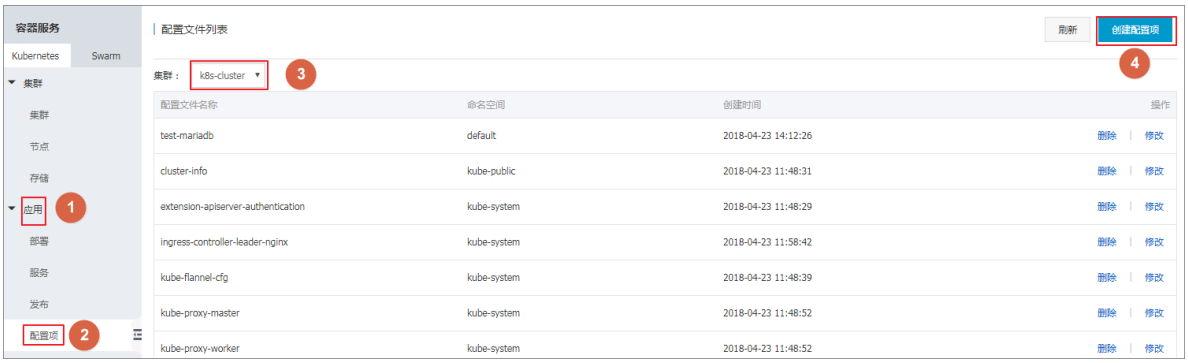
1.8 配置項及密鑰管理

1.8.1 建立配置項

在Container Service管理主控台上，您可以通過配置項菜單或使用模板來建立配置項。

通過配置項建立

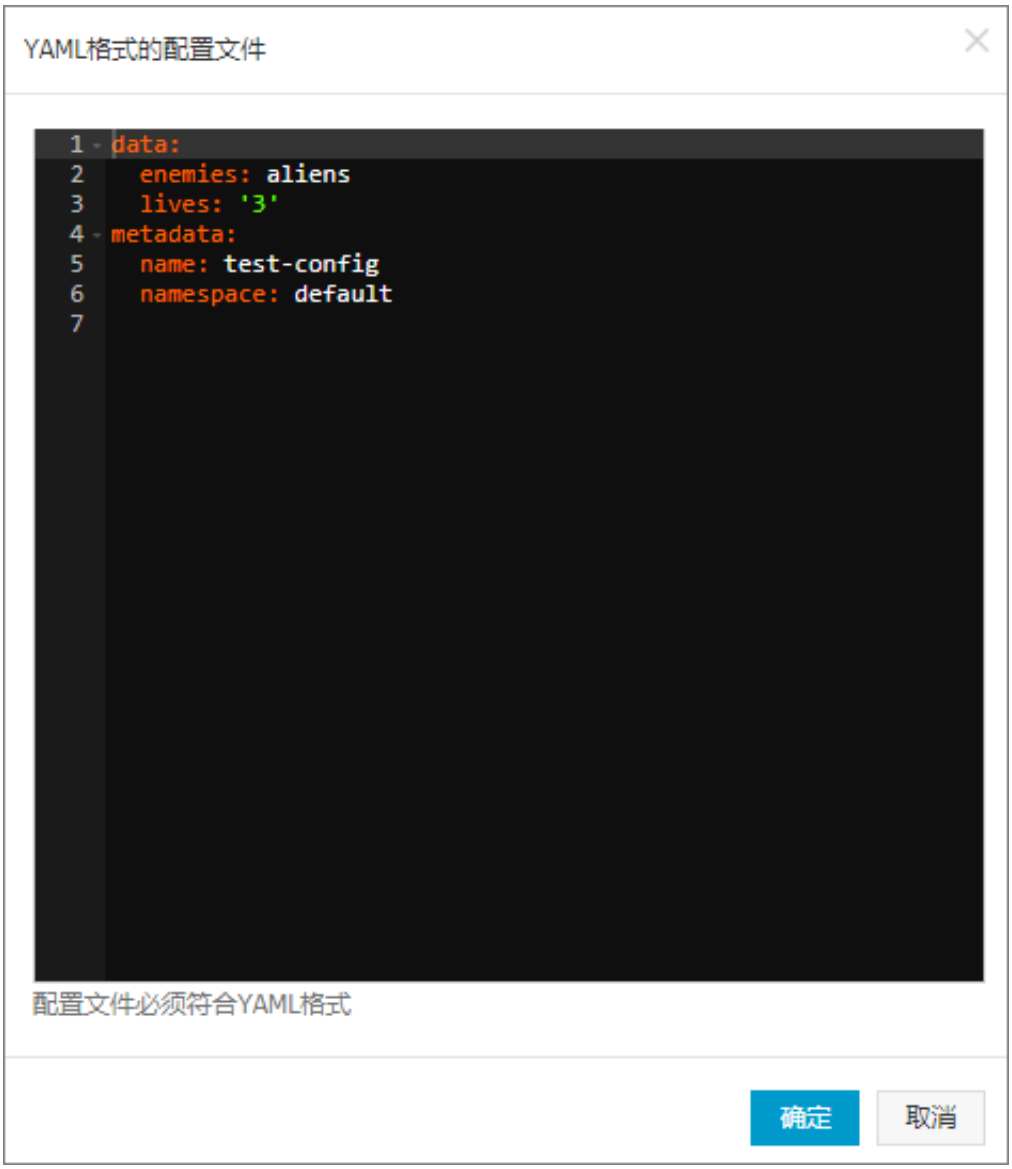
1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 配置項，進入設定檔列表。
3. 在設定檔列表頁面，選擇需要建立配置項的叢集，然後單擊建立配置項。



4. 填寫設定檔的資訊並單擊確定。

- 命名空間：選擇該配置項所屬的命名空間。配置項（ConfigMap）是 kubernetes 資來源物件，需要作用於命名空間。
- 設定檔名：指定配置項的檔案名稱，名稱可以包含小寫字母、數字、連字號（-）或者點號（.），名稱不可為空。其他資來源物件需要引用設定檔名來擷取配置資訊。
- 配置項:填寫變數名稱和變數值後，需要單擊右側的添加。您也可以單擊編輯設定檔 在彈出的對話方塊裡編寫配置項並單擊確定。

本樣本中設定了 enemies 和 lives 變數，分別用於傳遞 aliens 和 3 這兩個參數。



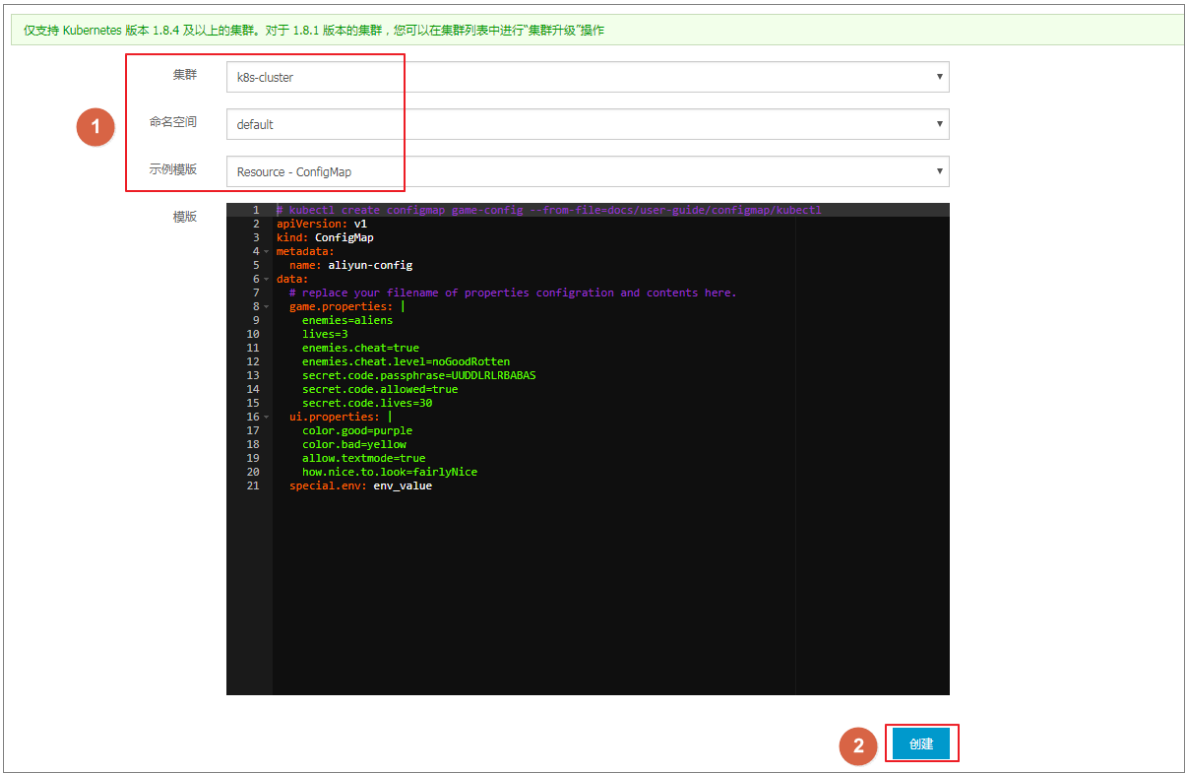
5. 單擊確定後，您可以在設定檔列表中看到 test-config 設定檔。

使用模板建立

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表。
3. 單擊頁面右上方的使用模板建立。

4. 在使用模版部署的頁面，設定設定檔的資訊並單擊部署。

   - 叢集：選擇需要建立配置項的叢集。
   - 命名空間：選擇該配置項所屬的命名空間。配置項（ConfigMap）是 kubernetes 資來源物件，需要作用於命名空間。
   - 樣本模板：您可以選擇自訂，根據 Kubernetes yaml 文法規則編寫 ConfigMap 檔案，或者選擇樣本模板 **resource-ConfigMap**。該樣本模板的 ConfigMap 名稱為 aliyun-config，包含兩個變數檔案 `game.properties` 和 `ui.properties`，您可以在此基礎上進行修改，然後單擊部署。



5. 部署成功後，您可以在設定檔列表下看到 aliyun-config 設定檔。

1.8.2 在 pod 中使用配置項

您可以在 Pod 中使用配置項，有多種使用情境，主要包括：

- 使用配置項定義 pod 環境變數
- 通過配置項設定命令列參數
- 在資料卷中使用配置項

更多關於配置項的資訊，可以參見 *Configure a Pod to Use a ConfigMap*。

使用限制

您在 pod 裡使用配置項時，需要兩者處於同一叢集和命名空間中。

建立配置項

本樣本建立配置項 special_config，包含 `SPECIAL_LEVEL: very` 和 `SPECIAL_TYPE: charm` 兩個索引值對。

使用編排模板建立配置項

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，然後單擊右上方的使用模板建立。
3. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

   您可以使用如下 yaml 樣本模板建立配置項。

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: special-config
   namespace: default
data:
   SPECIAL_LEVEL: very
   SPECIAL_TYPE: charm
```

通過 **Web** 介面建立配置項

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 配置項，進入配置項列表頁面。
3. 選擇所需的叢集和命名空間，然後單擊右上方的使用模板建立。

4. 輸入配置項名稱，然後單擊添加，輸入配置項，最後單擊確定。

使用配置項定義 pod 環境變數

使用配置項的資料定義 pod 環境變數

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，然後單擊右上方的使用模板建立。
3. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

   您可以在 pod 中定義環境變數，使用 `valueFrom` 引用 SPECIAL_LEVEL 的 value 值，從而定義 pod 的環境變數。

   下面是一個編排樣本。

```
apiVersion: v1
kind: Pod
metadata:
   name: config-pod-1
spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "env" ]
       env:
         - name: SPECIAL_LEVEL_KEY
           valueFrom:                                  ##使用valueFrom來指
定env引用配置項的value值
             configMapKeyRef:
               name: special-config                    ##引用的設定檔名稱
               key: SPECIAL_LEVEL                      ##引用的配置項key
   restartPolicy: Never
```

同理，如果您需要將多個配置項的 value 值定義為 pod 的環境變數值，您只需要在 pod 中添加多個 env 參數即可。

將配置項的所有 **key/values** 配置為 **pod** 的環境變數

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，然後單擊右上方的使用模板建立。
3. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

   如果您想在一個 pod 中將配置項的所有 key/values 索引值對配置為 pod 的環境變數，可以使用 envFrom 參數，配置項中的 key 會成為 Pod 中的環境變數名稱。

   下面是一個編排樣本。

```
apiVersion: v1
kind: Pod
metadata:
   name: config-pod-2
spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "env" ]
       envFrom:                ##引用 sepcial-config 設定檔的所有 key/
values 索引值對
       - configMapRef:
           name: special-config
   restartPolicy: Never
```

通過配置項設定命令列參數

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，然後單擊右上方的使用模板建立。
3. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

   您可以使用配置項設定容器中的命令或者參數值，使用環境變數替換文法 `$(VAR_NAME)`來進行。

   下面是一個編排樣本。

```
apiVersion: v1
kind: Pod
metadata:
   name: config-pod-3
spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(
SPECIAL_TYPE_KEY)" ]
       env:
         - name: SPECIAL_LEVEL_KEY
           valueFrom:
             configMapKeyRef:
               name: special-config
               key: SPECIAL_LEVEL
```

```
          - name: SPECIAL_TYPE_KEY
            valueFrom:
              configMapKeyRef:
                name: special-config
                key: SPECIAL_TYPE
   restartPolicy: Never
```

運行這個 pod 後，會輸出如下結果。

```
very charm
```

在資料卷中使用配置項

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 部署，然後單擊右上方的使用模板建立。
3. 選擇所需的叢集和命名空間，選擇範例模板或自訂，然後單擊建立。

   您也可以在資料卷裡面使用配置項，在 volumes 下指定配置項名稱，會將 key/values 的資料存放區到 mountPath 路徑下（本例中是 */etc/config*）。最終產生以 key 為檔案名稱，values 為檔案內容的設定檔。

   下面是一個編排樣本。

```
apiVersion: v1
kind: Pod
metadata:
   name: config-pod-4
spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "ls /etc/config/" ]   ##列出該目錄
下的檔案名稱
       volumeMounts:
       - name: config-volume
         mountPath: /etc/config
   volumes:
     - name: config-volume
       configMap:
         name: special-config
   restartPolicy: Never
```

   運行 pod 後，會輸出配置項的 key。

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

1.8.3 修改配置項

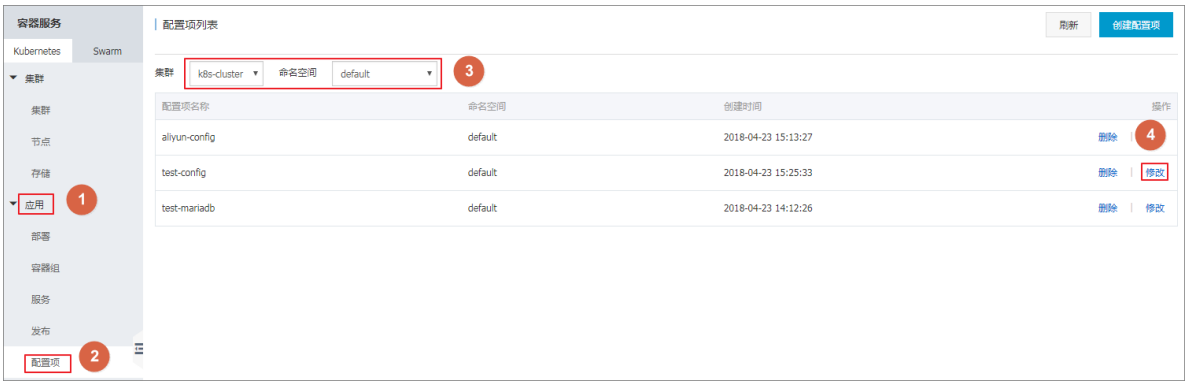
您可以修改配置項的配置。

 说明：

修改設定檔會影響使用該設定檔的應用。

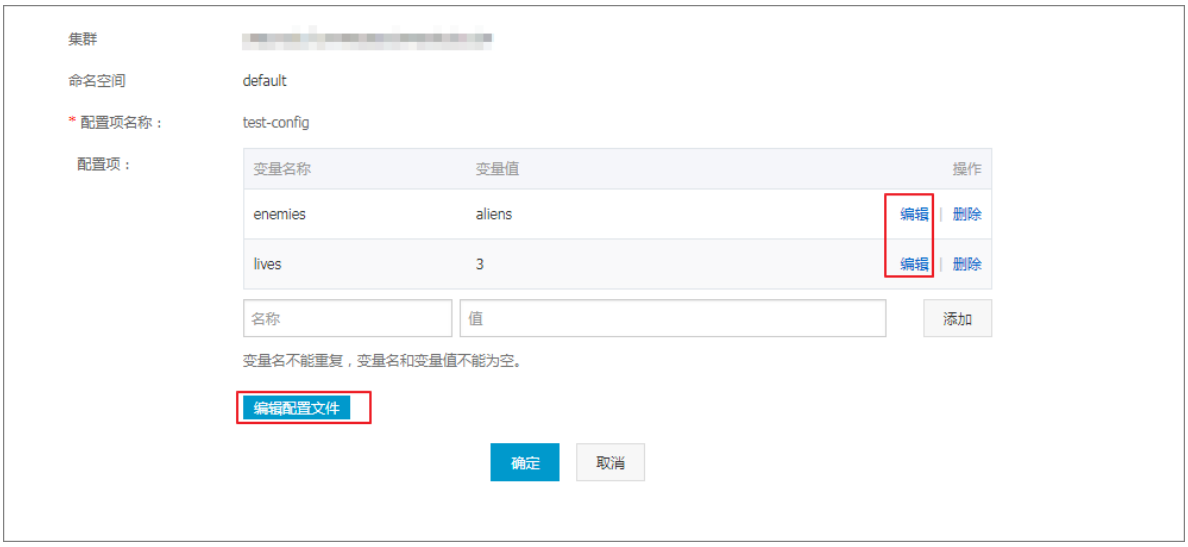
通過配置項進行修改

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 配置項，進入設定檔列表。
3. 選擇所需的叢集和命名空間，選擇需要修改的配置項並單擊右側的修改。



4. 在彈出的確認對話方塊中，單擊確定。

5. 修改配置項。
   - 選擇需要修改的配置項並單擊右側的編輯，修改配置後單擊儲存。
   - 或者單擊編輯設定檔，完成編輯後單擊確定。

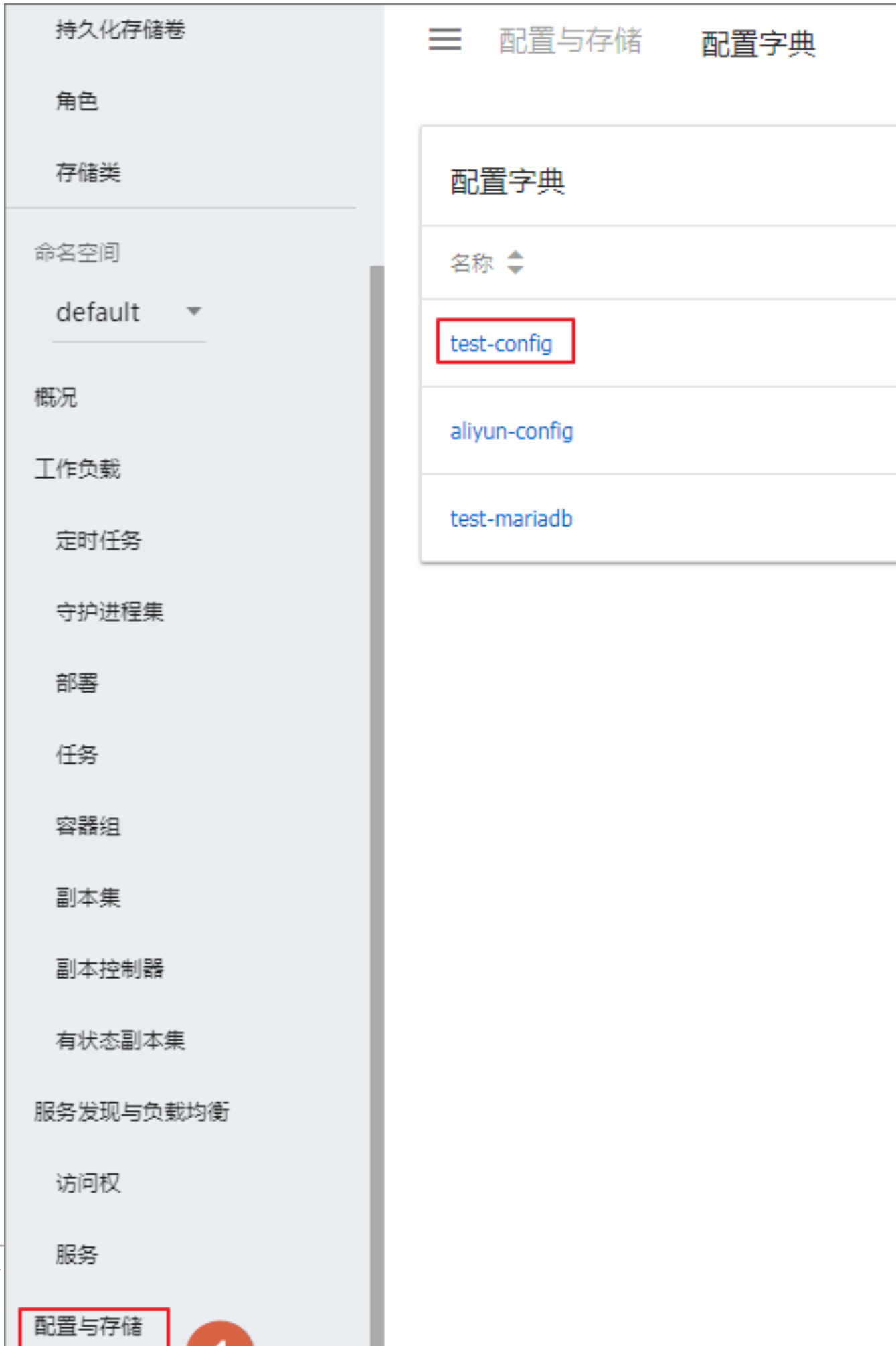


6. 完成配置項修改後，單擊確定。

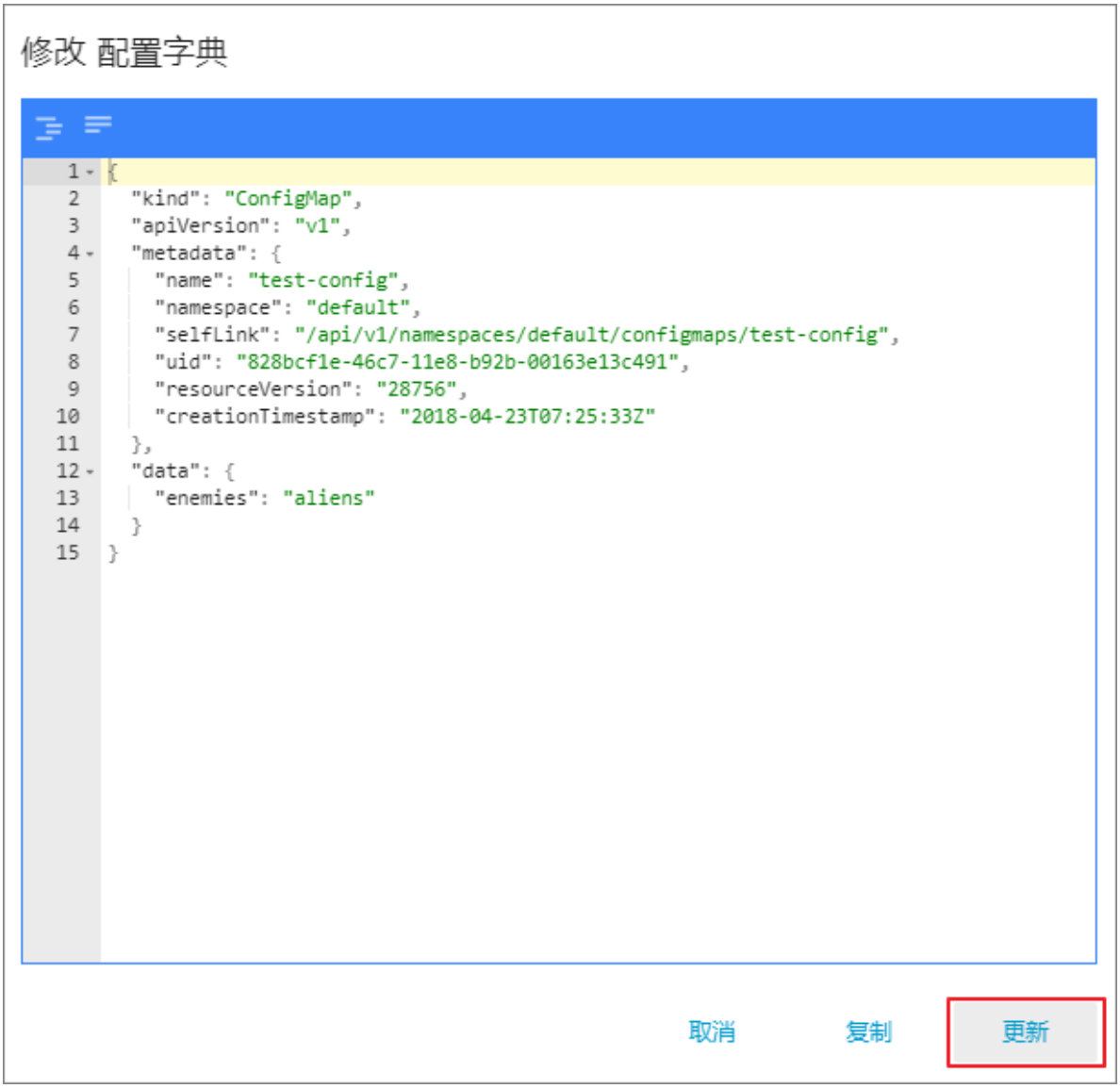
通過 Kubernetes Dashboard 進行修改

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入叢集列表。
3. 選擇所需的叢集，並單擊右側的控制台。

4. 在 Kubernetes Dashboard 頁面，單擊左側導覽列中的配置與儲存 > 配置字典，選擇所需的配置字典，單擊右側的操作 > 查看/編輯 **YAML** 。


持久化存储卷
角色
存储类
命名空间
default
概况
工作负载
定时任务
守护进程集
部署
任务
容器组
副本集
副本控制器
有状态副本集
服务发现与负载均衡
访问权
服务
配置与存储
配置与存储
配置字典
配置字典
名称
test-config
aliyun-config
test-mariadb

5. 彈出修改配置字典對話方塊，對組態變數進行修改後，單擊更新。

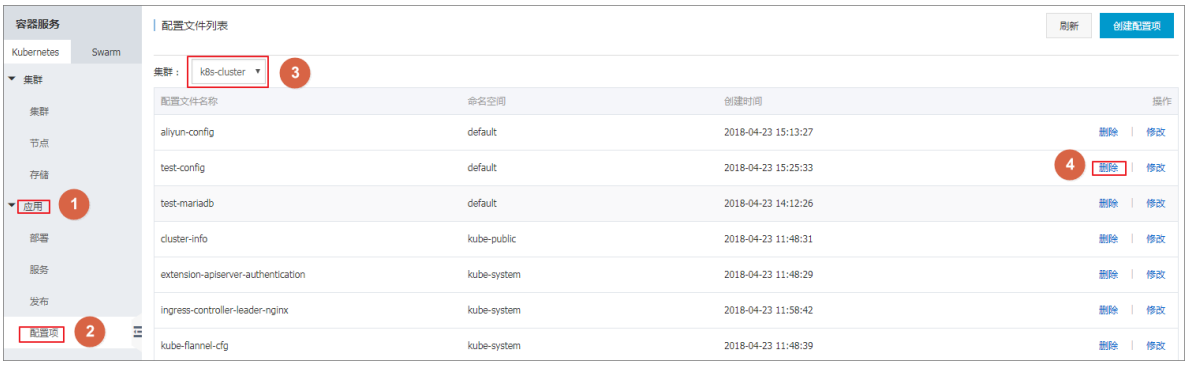


1.8.4 刪除配置項

您可以刪除不再使用的配置項。

通過配置項菜單進行刪除

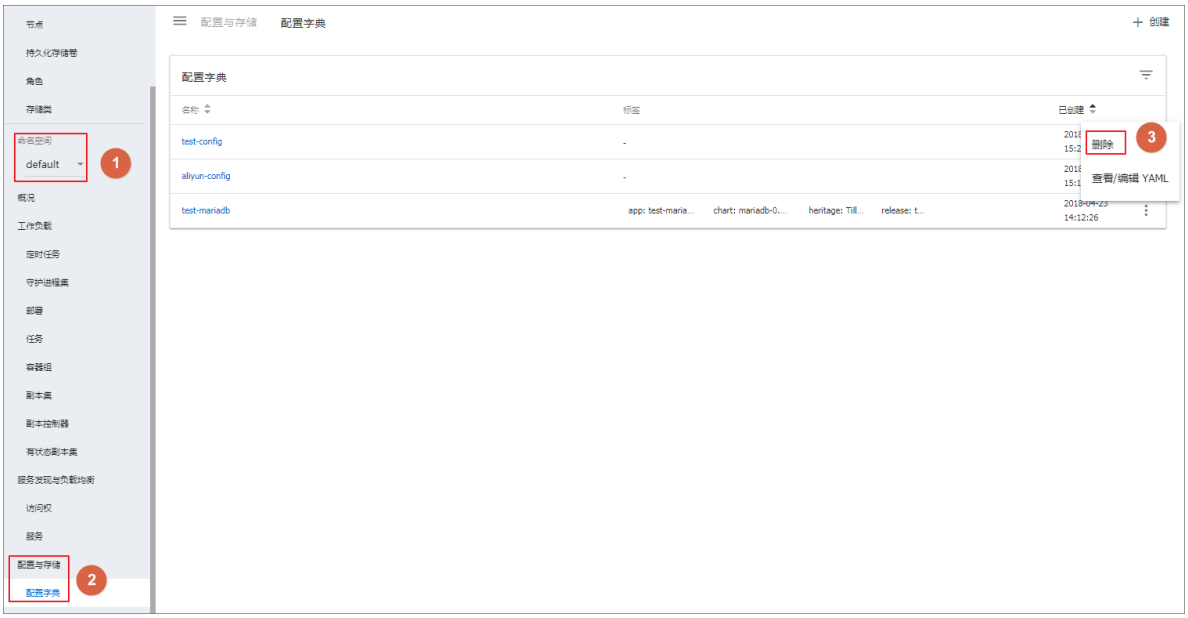
1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 配置項，進入設定檔列表。
3. 選擇目的地組群，選擇需要修改的配置項並單擊右側的刪除。



通過 Kubernetes Dashboard 進行刪除

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 單擊左側導覽列中的叢集，選擇目的地組群，單擊右側的控制台。

4. 在 Kubernetes Dashboard 頁面，選擇所需的命名空間，單擊左側導覽列中的配置與儲存 > 配置字典，單擊右側的的操作按鈕，在下拉框中單擊刪除 。



5. 在彈出的對話方塊中，單擊刪除。

1.8.5 建立密鑰

前提条件

您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。

背景信息

若您需要在 Kubernetes 叢集中使用一些敏感的配置，比如密碼、認證等資訊時，建議使用密鑰（secret），即保密字典。

密鑰有多種類型，例如：

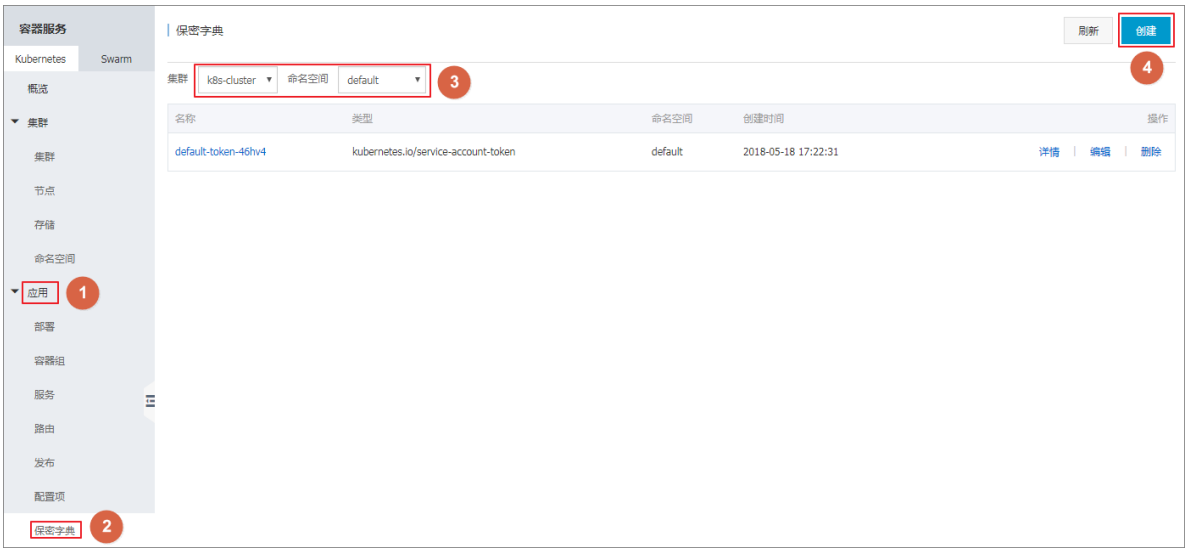
- Service Account：用來訪問 Kubernetes API，由 Kubernetes 自動建立，並且會自動掛載到 Pod 的`/run/secrets/kubernetes.io/serviceaccount`目錄中。
- Opaque：base64 編碼格式的 Secret，用來儲存密碼、認證等敏感資訊。

預設情況下，通過 Web 介面只能建立 Opaque 類型的密鑰。Opaque 類型的資料是一個 map 類型，要求value 是 base64 編碼格式。阿里雲Container Service提供一鍵建立密鑰的功能，自動將明文資料編碼為 base64 格式。

您也可通過命令列手動建立密鑰，請參見 *kubernetes secret* 瞭解更多資訊。

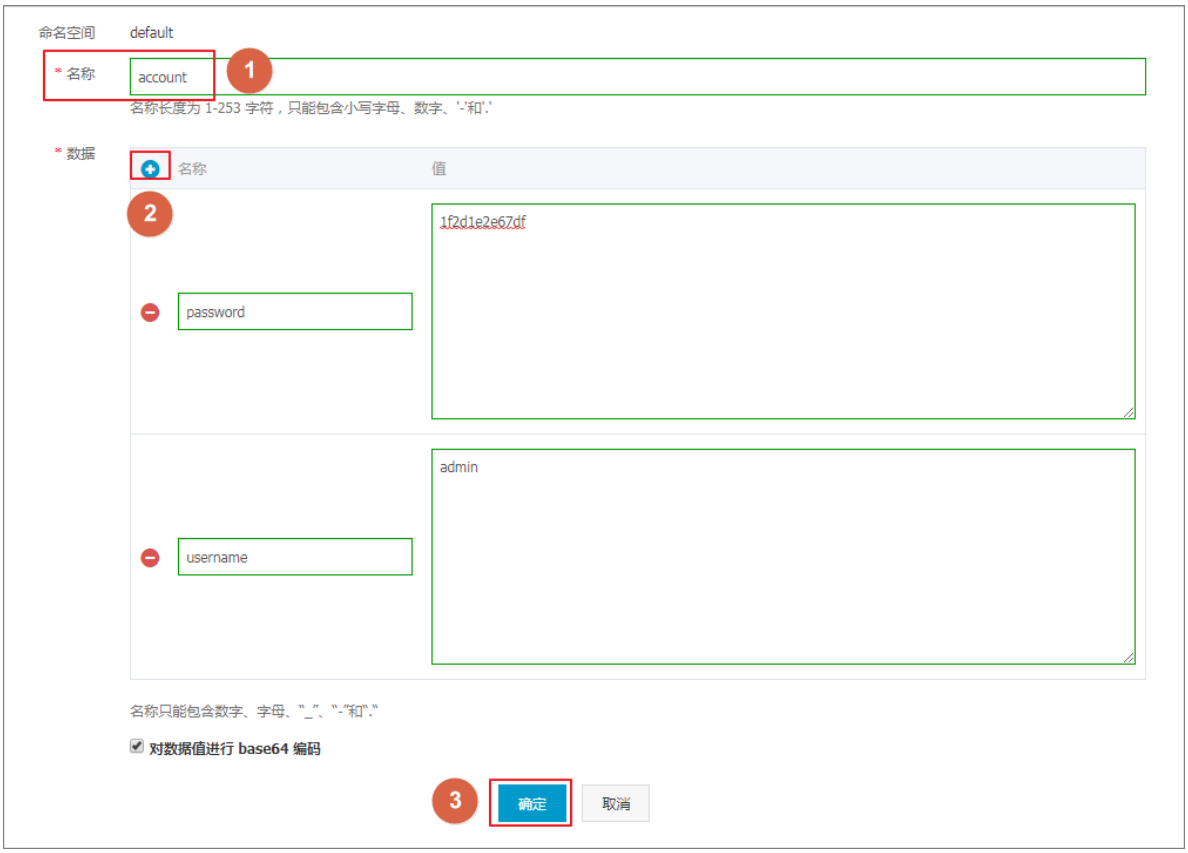
操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 保密字典，進入保密字典頁面。
3. 選擇所需的叢集和命名空間，單擊右上方的建立。



4. 配置新的保密字典。

> 说明：
> 若您輸入密鑰的明文資料，請注意勾選對資料值進行**base64編碼**



1. 名稱：輸入密鑰的名稱。名稱長度為 1-253 字元，只能包含小寫字母、數字、'-'和'.'
2. 配置密鑰的資料。單擊添加，在彈出的對話方塊中，輸入密鑰名稱和值，即索引值對。本例中建立的密鑰包含兩個 value，`username:admin`和 `passwrod：1f2d1e2e67df`。
3. 單擊確定。

5. 預設返回保密字典頁面，您可看到建立的密鑰出現在列表中。

1.8.6 查看密鑰

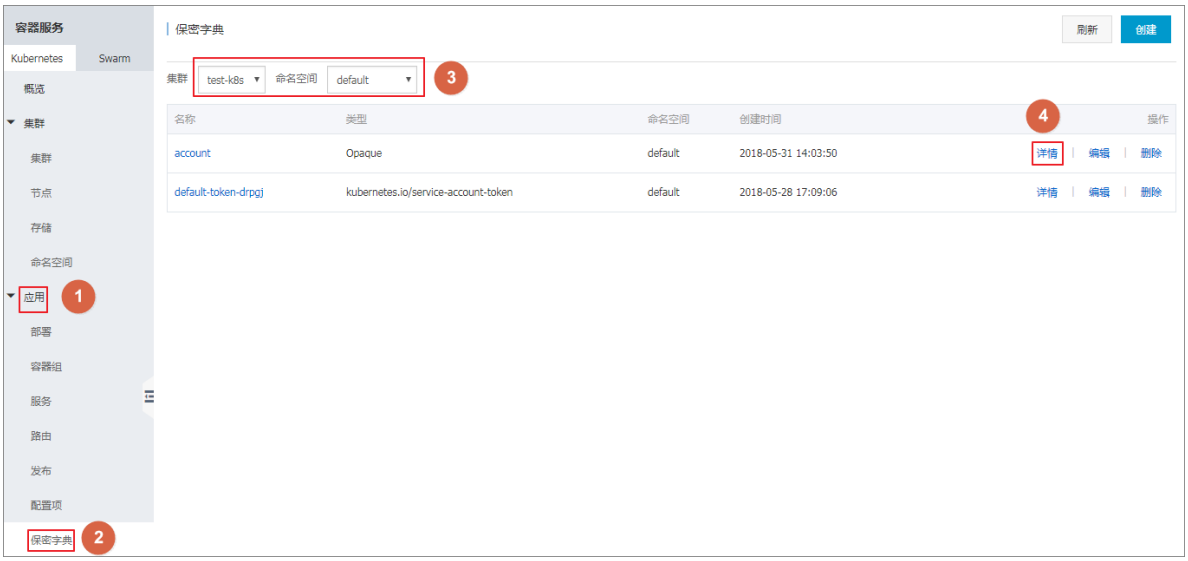
您可通過阿里雲Container Service Web 介面查看已建立的密鑰。

前提条件

- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已建立一個密鑰，參見*####*。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 保密字典，進入保密字典頁面。
3. 選擇所需的叢集和命名空間，選擇所需的密鑰，並單擊右側的詳情。



4. 進入密鑰的詳情頁面，您可查看該密鑰的基本資料，以及密鑰包含的資料資訊。

   單擊詳細資料中資料名稱右側的表徵圖，可查看資料的明文。

1.8.7 編輯密鑰

您可通過阿里雲Container Service Web 介面直接編輯已有密鑰。

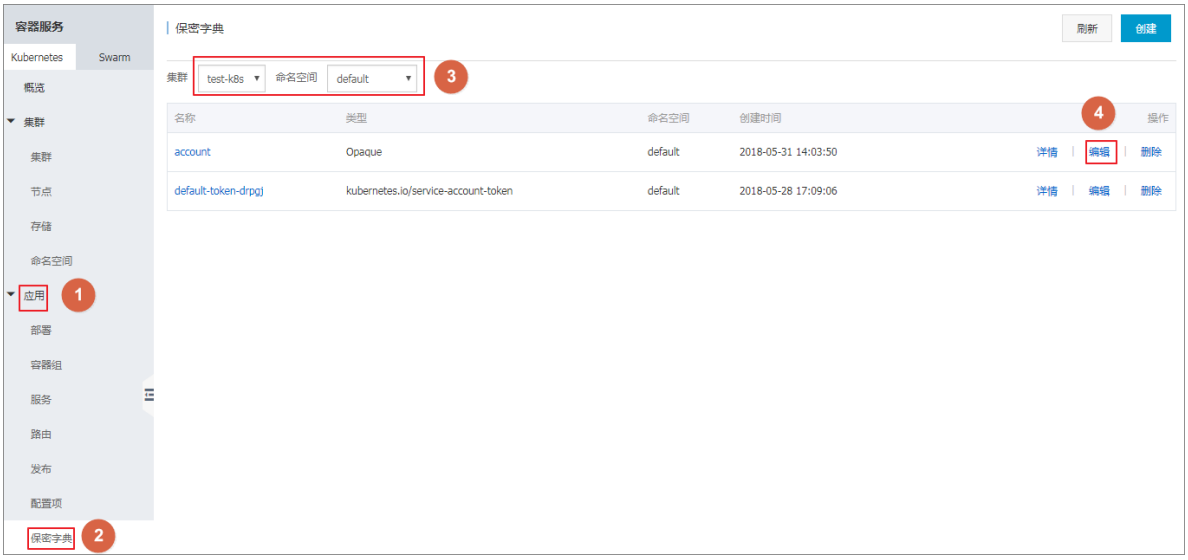
前提條件

- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已建立一個密鑰，參見*####*。

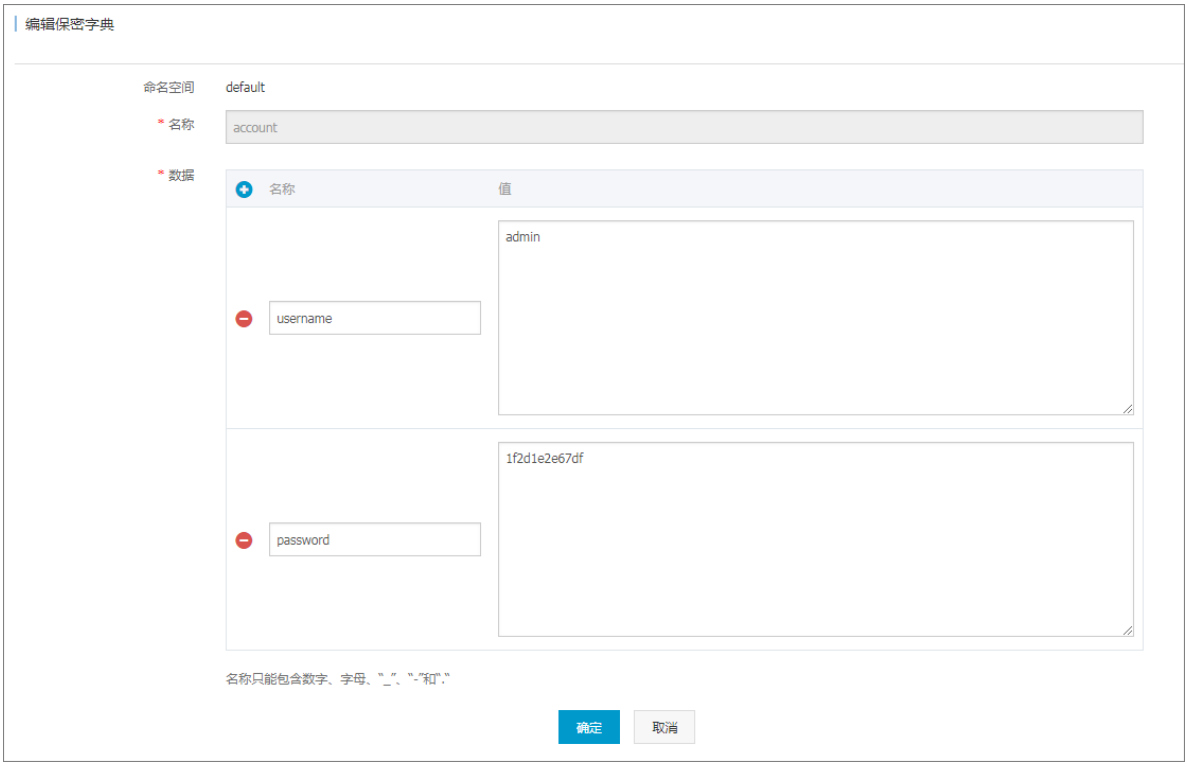
操作步驟

1. 登入 *Container Service#####*。

2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 保密字典，進入保密字典頁面。
3. 選擇所需的叢集和命名空間，選擇所需的密鑰，並單擊右側的編輯。



4. 在編輯密鑰的頁面，可對密鑰的資料進行編輯。



5. 最後單擊確定。

1.8.8 刪除密鑰

您可通過阿里雲Container Service Web 介面刪除已有密鑰。

前提条件

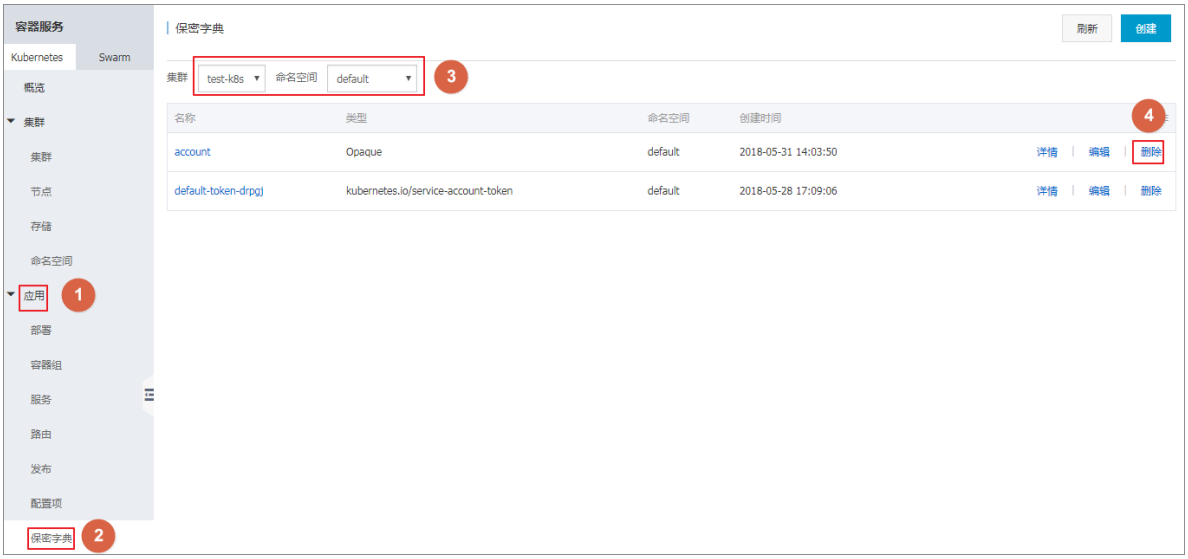
- 您已成功建立一個 Kubernetes 叢集，參見##Kubernetes##。
- 您已建立一個密鑰，參見####。

背景信息

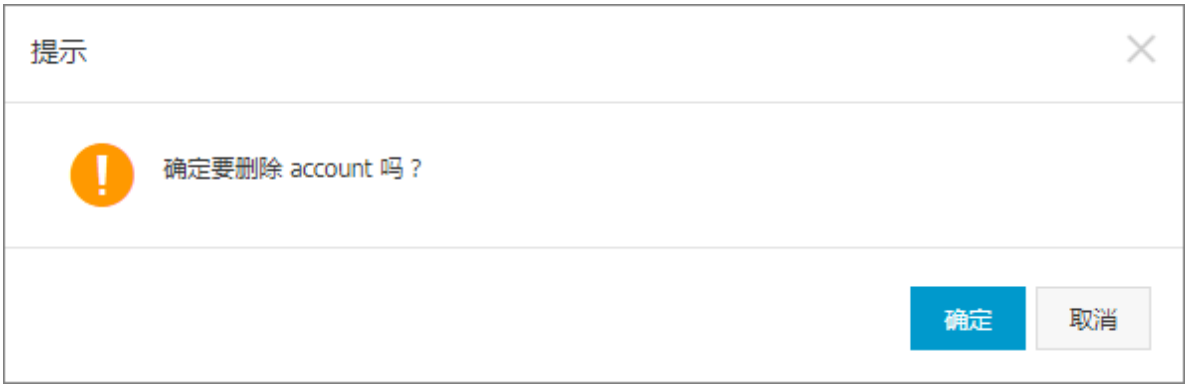
> 说明：
> 叢集建立過程中產生的密鑰請勿刪除。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 保密字典，進入保密字典頁面。
3. 選擇所需的叢集和命名空間，選擇所需的密鑰，並單擊右側的刪除。



4. 在彈出的對話方塊中，單擊確定，完成刪除。



1.9 儲存管理

1.9.1 概述

Container Service支援 kubernetes Pod 自動綁定阿里雲雲端硬碟、NAS、 OSS 儲存服務。

目前，支援靜態儲存卷和動態儲存裝置卷。每種資料卷的支援情況如下：

阿里雲儲存	待用資料卷	動態資料卷
阿里雲雲端硬碟	您可以通過以下兩種方式使用雲端硬碟靜態儲存卷： • 直接通過 volume 使用 • 通過 PV/PVC 使用	支援
阿里雲 NAS	NAS 靜態儲存卷又支援下面兩種使用方式： • 通過 flexvolume 外掛程式使用 — 通過 volume 方式使用 — 使用 PV/PVC • 通過 Kubernetes 的 NFS 驅動使用	支援
阿里雲 OSS	您可以通過以下兩種方式使用 OSS 靜態儲存卷： • 直接使用 volume 方式 • 使用 PV/PVC	不支援

1.9.2 安裝外掛程式

通過下面的 yaml 配置部署阿里雲 Kubernetes 儲存外掛程式。

 说明：

如果您的 Kubernetes 叢集是在 2018 年 2 月 6 日之前建立的，那麼您使用資料卷之間需要先安裝阿里雲 Kubernetes 儲存外掛程式；如果您的 Kubernetes 叢集是在 2018 年 2 月 6 日之後建立的，那麼您可以直接使用資料卷，不需要再安裝阿里雲 Kubernetes 儲存外掛程式。

使用限制

目前支援 CentOS 7 作業系統。

注意事項

- 使用 flexvolume 需要 kubelet 關閉 `--enable-controller-attach-detach` 選項。預設阿里雲 Kubernetes 叢集已經關閉此選項。

- 在 kube-system 使用者空間部署 flexvolume。

驗證安裝完成：

在 master 節點上：

- 執行命令：`kubectl get pod -n kube-system | grep flexvolume` 。輸出若干（節點個數）Running 狀態的 Pod 列表。
- 執行命令：`kubectl get pod -n kube-system | grep alicloud-disk-controller`。輸出一個 Running 狀態的 Pod 列表。

安裝步驟

安裝 Flexvolume：

```
apiVersion: apps/v1 # for versions before 1.8.0 use extensions/v1beta1
kind: DaemonSet
metadata:
  name: flexvolume
  namespace: kube-system
  labels:
    k8s-volume: flexvolume
spec:
  selector:
    matchLabels:
      name: acs-flexvolume
  template:
    metadata:
      labels:
        name: acs-flexvolume
    spec:
      hostPID: true
      hostNetwork: true
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
      - name: acs-flexvolume
        image: registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.9.7
-42e8198
        imagePullPolicy: Always
        securityContext:
          privileged: true
        env:
        - name: ACS_DISK
          value: "true"
        - name: ACS_NAS
          value: "true"
        - name: ACS_OSS
          value: "true"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
```

```
        volumeMounts:
        - name: usrdir
          mountPath: /host/usr/
        - name: etcdir
          mountPath: /host/etc/
        - name: logdir
          mountPath: /var/log/alicloud/
      volumes:
      - name: usrdir
        hostPath:
          path: /usr/
      - name: etcdir
        hostPath:
          path: /etc/
      - name: logdir
       hostPath:
          path: /var/log/alicloud/
```

安裝 Disk Provisioner：

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-common
provisioner: alicloud/disk
parameters:
  type: cloud
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-efficiency
provisioner: alicloud/disk
parameters:
  type: cloud_efficiency
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-ssd
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-available
provisioner: alicloud/disk
parameters:
  type: available
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: alicloud-disk-controller-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
```

```
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alicloud-disk-controller
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-alicloud-disk-controller
subjects:
  - kind: ServiceAccount
    name: alicloud-disk-controller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: alicloud-disk-controller-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: alicloud-disk-controller
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: alicloud-disk-controller
    spec:
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      nodeSelector:
         node-role.kubernetes.io/master: ""
      serviceAccount: alicloud-disk-controller
      containers:
        - name: alicloud-disk-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-disk-
controller:v1.9.3-ed710ce
          volumeMounts:
            - name: cloud-config
              mountPath: /etc/kubernetes/
            - name: logdir
              mountPath: /var/log/alicloud/
```

```
      volumes:
        - name: cloud-config
          hostPath:
            path: /etc/kubernetes/
        - name: logdir
          hostPath:
            path: /var/log/alicloud/
```

1.9.3 使用阿里雲雲端硬碟

您可以在阿里雲Container Service Kubernetes 叢集中使用阿里雲雲端硬碟儲存卷。

目前，阿里雲雲端硬碟提供兩種 Kubernetes 掛載方式：

- #####

  您可以通過以下兩種方式使用雲端硬碟靜態儲存卷：

  - ####volume##
  - ## PV/PVC ##
- #######

> 说明：
> 對建立的雲端硬碟容量有如下要求：
> - 普通雲端硬碟：最小5Gi
> - 高效雲端硬碟：最小20Gi
> - SSD雲端硬碟：最小20Gi

靜態儲存卷

您可以直接通過volume使用阿里雲雲端硬碟儲存卷或者通過 PV/PVC 使用阿里雲雲端硬碟儲存卷。

前提條件

使用雲端硬碟資料卷之前，您需要先在 ECS 管理主控台上建立雲端硬碟。有關如何建立雲端硬碟，參見#####。

使用說明

- 雲端硬碟為非共用儲存，只能同時被一個 pod 掛載。
- 使用雲端硬碟儲存卷前需要先申請一個雲端硬碟，並獲得磁碟 ID。參見 #####。
- volumeId： 表示所掛載雲端硬碟的磁碟ID；volumeName、PV Name要與之相同。
- 叢集中只有與雲端硬碟在同一個可用性區域（Zone）的節點才可以掛載雲端硬碟。

直接通過 **volume** 使用

使用*disk-deploy.yaml*檔案建立 Pod。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-disk-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-flexvolume-disk
        image: nginx
        volumeMounts:
          - name: "d-bp1j17ifxfasvts3tf40"
            mountPath: "/data"
      volumes:
        - name: "d-bp1j17ifxfasvts3tf40"
          flexVolume:
            driver: "alicloud/disk"
            fsType: "ext4"
            options:
              volumeId: "d-bp1j17ifxfasvts3tf40"
```

通過 **PV/PVC** 使用

步驟 **1** 建立雲端硬碟類型的 **PV**

您可以使用 yaml 檔案或者控制台介面建立雲端硬碟類型的 PV。

通過 **yaml** 檔案建立 **PV**

使用 `disk-pv.yaml` 檔案建立 PV。

 说明：

pv name 要與阿里雲盤 ID 相同。

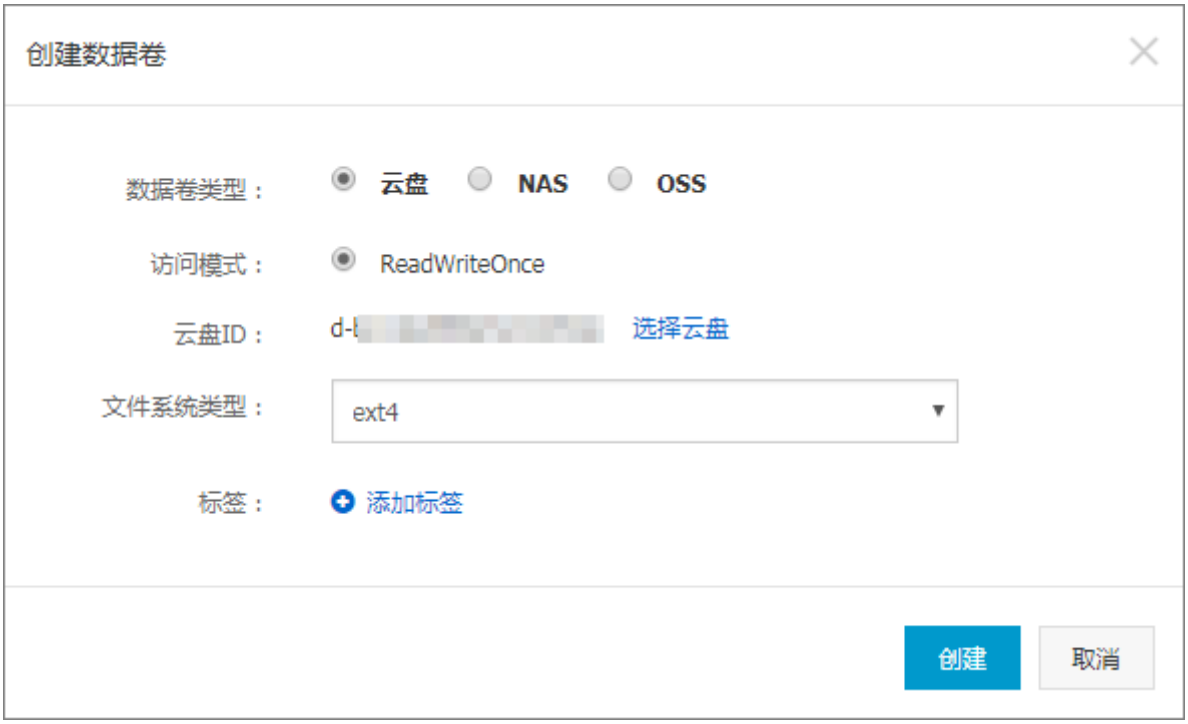
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: d-bp1j17ifxfasvts3tf40
  labels:
    failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
    failure-domain.beta.kubernetes.io/region: cn-hangzhou
spec:
  capacity:
    storage: 20Gi
  storageClassName: disk
  accessModes:
    - ReadWriteOnce
  flexVolume:
    driver: "alicloud/disk"
    fsType: "ext4"
    options:
```

```
      volumeId: "d-bp1j17ifxfasvts3tf40"
```

通過控制台介面建立雲端硬碟資料卷

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 儲存卷，進入資料卷列表頁面。
3. 選擇所需的叢集，單擊頁面右上方的建立。

4. 在建立資料卷對話方塊中，配置資料卷的相關參數。
   - 資料卷類型：本樣本中為雲端硬碟。
   - 訪問模式：預設為 ReadWriteOnce。
   - 雲端硬碟 ID：您可以選擇與叢集屬於相同地區和可用性區域下處於待掛載狀態的雲端硬碟。
   - 檔案系統類型：您可以選擇以什麼資料類型將資料存放區到雲端硬碟上，支援的類型包括 ext4、ext3、xfs、vfat。預設為 ext4。
   - 標籤：為該資料卷添加標籤。



5. 完成配置後，單擊建立。

步驟 2 建立 PVC

使用 disk-pvc.yaml 檔案建立 PVC。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-disk
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: disk
  resources:
    requests:
      storage: 20Gi
```

步驟 **3** 建立 **Pod**

使用*disk-pod.yaml*檔案建立 pod。

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-alicloud-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
        - name: pvc-disk
          mountPath: "/data"
  volumes:
  - name: pvc-disk
    persistentVolumeClaim:
      claimName: pvc-disk
```

動態儲存裝置卷

動態儲存裝置卷需要您手動建立 StorageClass，並在PVC中通過 storageClassName 來指定期望的雲端硬碟類型。

建立 **StorageClass**

```
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-common-hangzhou-b
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
  regionid: cn-hangzhou
  zoneid: cn-hangzhou-b
```

參數說明：

- provisioner：配置為 alicloud/disk，標識使用阿里雲雲端硬碟 Provsioner 外掛程式建立。

- type：標識雲端硬碟類型，支援 cloud、cloud_efficiency、cloud_ssd、available 四種類型；其中 available 會對高效、SSD、普通雲端硬碟依次嘗試建立，直到建立成功。
- regionid：期望建立雲端硬碟的地區。
- reclaimPolicy: 雲端硬碟的回收策略，預設為Delete，支援Retain。
- zoneid：期望建立雲端硬碟的可用性區域。

建立服務

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: disk-common
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: alicloud-disk-common-hangzhou-b
  resources:
    requests:
      storage: 20Gi
---
kind: Pod
apiVersion: v1
metadata:
  name: disk-pod-common
spec:
  containers:
  - name: disk-pod
    image: nginx
    volumeMounts:
      - name: disk-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: disk-pvc
      persistentVolumeClaim:
        claimName: disk-common
```

預設選項

叢集預設提供了下面幾種 StorageClass，可以在單 AZ 類型的叢集中使用。

- alicloud-disk-common：普通雲端硬碟。
- alicloud-disk-efficiency：高效雲端硬碟。
- alicloud-disk-ssd：SSD雲端硬碟。
- alicloud-disk-available：提供高可用選項，先試圖建立高效雲端硬碟；如果相應AZ的高效雲端硬碟資源售盡，再試圖建立SSD盤；如果SSD售盡，則試圖建立普通雲端硬碟。

使用雲端硬碟建立多執行個體**StatefulSet**

使用 volumeClaimTemplates 的方式來建立，這樣會動態建立多個 PVC 和 PV 並綁定。

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: disk-common
          mountPath: /data
  volumeClaimTemplates:
  - metadata:
      name: disk-common
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "alicloud-disk-common"
      resources:
        requests:
          storage: 10Gi
```

1.9.4 使用阿里雲 NAS

您可以在Container Service Kubernetes 叢集中使用阿里雲 NAS 資料卷。

目前阿里雲 NAS 提供兩種 Kubernetes 掛載方式：

- #####

  其中，靜態儲存卷又支援下面兩種使用方式：

- 通過 flexvolume 外掛程式使用
  - 通過 volume 方式使用
  - 使用 PV/PV
- 通過 Kubernetes 的 NFS 驅動使用
- #######

前提條件

使用 NAS 資料卷之前，您需要先在 NAS 管理主控台上建立檔案系統，並在檔案系統中添加 Kubernetes 叢集的掛載點。建立的 NAS 檔案系統需要和您的叢集位於同一 VPC。

靜態儲存卷

您可以通過阿里雲提供的 flexvolume 外掛程式使用阿里雲 NAS 檔案儲存體服務或者通過 Kubernetes 的 NFS 驅動使用阿里雲 NAS 檔案儲存體服務。

通過 **flexvolume** 外掛程式使用

使用 flexvolume 外掛程式，您可以通過 volume 方式使用阿里雲 NAS 資料卷或者通過 PV/PVC 方式使用阿里雲 NAS 資料卷。

 说明：

- NAS 為共用儲存，可以同時為多個 Pod 提供共用儲存服務。
- server：為 NAS 資料盤的掛載點。
- path：為串連 NAS 資料卷的掛載目錄，支援掛載 NAS 子目錄，且當子目錄不存在時，會自動建立子目錄並掛載。
- vers：定義 nfs 掛載協議的版本號碼，支援3.0 和 4.0。
- mode：定義掛載目錄的存取權限，注意掛載 NAS 盤根目錄時不能配置掛載許可權。當 NAS 盤中資料量很大時，配置 mode 會導致執行掛載非常慢，甚至掛載失敗。

通過 **volume** 方式使用

使用 `nas-deploy.yaml` 檔案建立 Pod。

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-nas-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
```

```
        - name: "nas1"
          mountPath: "/data"
  volumes:
    - name: "nas1"
      flexVolume:
        driver: "alicloud/nas"
        options:
          server: "0cd8b4a576-grs79.cn-hangzhou.nas.aliyuncs.com"
          path: "/k8s"
          vers: "4.0"
```

使用 **PV/PVC**

步驟 **1** 建立 **PV**

您可以使用 yaml 檔案或者通過阿里雲Container Service控制台介面建立 NAS 資料卷。

- 使用 yaml 檔案建立 PV

  使用 `nas-pv.yaml` 檔案建立 PV。

  ```
  apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: pv-nas
  spec:
    capacity:
      storage: 5Gi
    storageClassName: nas
    accessModes:
      - ReadWriteMany
    flexVolume:
      driver: "alicloud/nas"
      options:
        server: "0cd8b4a576-uih75.cn-hangzhou.nas.aliyuncs.com"
        path: "/k8s"
        vers: "4.0"
  ```

- 通過控制台介面建立 NAS 資料卷

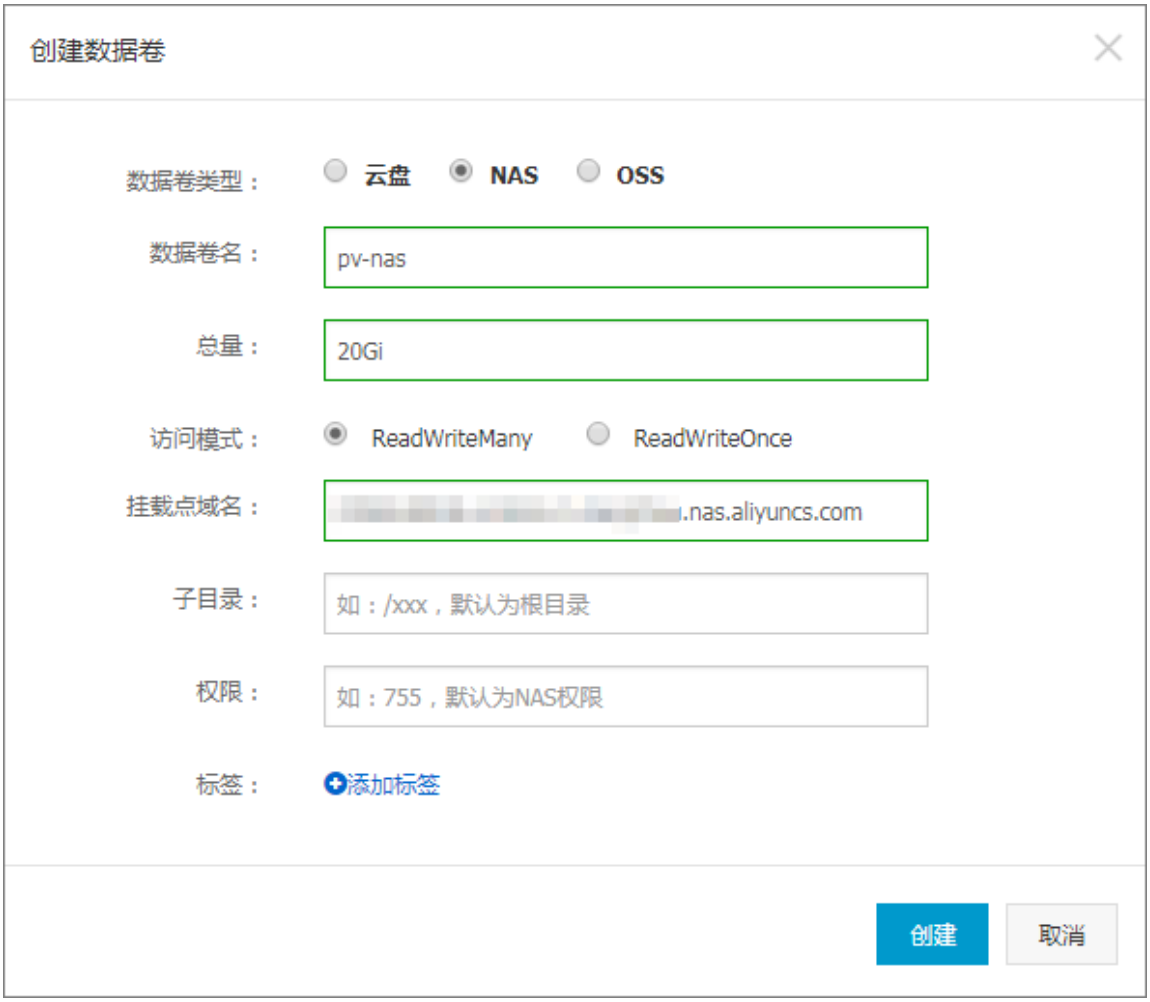
  1. 登入*Container Service#####*。
  2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 儲存，進入資料卷列表頁面。
  3. 選擇所需的叢集，單擊頁面右上方的建立。

  4. 在建立資料卷對話方塊中，配置資料卷的相關參數。

     - 資料卷類型：本樣本中為NAS。
     - 資料卷名：建立的資料卷的名稱。資料卷名在叢集內必須唯一。本例為 pv-nas。

- 資料卷總量：所建立資料卷的容量。注意不能超過磁碟容量。
- 訪問模式：預設為 ReadWriteMany。
- 掛載點網域名稱：叢集在 NAS 檔案系統中掛載點的掛載地址。
- 子目錄：NAS 路徑下的子目錄，以 ／ 開頭，設定後資料卷將掛載到指定的子目錄。
  - 如果 NAS 根目錄下沒有此子目錄，會預設建立後再掛載。
  - 您可以不填此項，預設掛載到 NAS 根目錄。
- 許可權：設定掛載目錄的存取權限，例如：755、644、777 等。
  - 只有掛載到 NAS 子目錄時才能設定許可權，掛載到根目錄時不能設定。
  - 您可以不填此項，預設許可權為 NAS 檔案原來的許可權。
- 標籤：為該資料卷添加標籤。



5. 完成配置後，單擊建立。

步驟 2 建立 PVC

使用 *nas-pvc.yaml* 檔案建立 PVC。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: nas
  resources:
    requests:
      storage: 5Gi
```

步驟 **3** 建立 **Pod**

使用*nas-pod.yaml*檔案建立 pod。

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-nas-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
          - name: pvc-nas
            mountPath: "/data"
  volumes:
  - name: pvc-nas
    persistentVolumeClaim:
        claimName: pvc-nas
```

通過 **Kubernetes** 的 **NFS** 驅動使用

步驟 **1** 建立 **NAS** 檔案系統

登入 ########## , 建立一個 NAS 檔案系統。

 说明：

建立的 NAS 檔案系統需要和您的叢集位於同一地區。

假設您的掛載點為 `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com`。

步驟 **2** 建立 **PV**

您可以使用編排模板或者通過阿里雲Container Service控制台介面建立 NAS 資料卷。

- 使用編排模板

  使用*nas-pv.yaml*檔案建立 PV。

執行以下命令建立一個類型為 NAS 的 PesistentVolume。

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
   name: nas
spec:
   capacity:
     storage: 8Gi
   accessModes:
     - ReadWriteMany
   persistentVolumeReclaimPolicy: Retain
   nfs:
     path: /
     server: 055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com
EOF
```

- 通過控制台介面建立 **NAS** 資料卷

  具體操作參見 *##PV/PVC* 中關於通過控制台介面建立 NAS 資料卷的內容。

步驟 2 建立 PVC

建立一個 PersistentVolumeClaim 來請求綁定該 PersistentVolume。

```
root@master # cat << EOF | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
     name: nasclaim
spec:
     accessModes:
       - ReadWriteMany
     resources:
       requests:
         storage: 8Gi
EOF
```

步驟 3 建立 Pod

建立一個應用來申明掛載使用該資料卷。

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
     name: mypod
spec:
     containers:
       - name: myfrontend
         image: registry.aliyuncs.com/spacexnice/netdia:latest
         volumeMounts:
         - mountPath: "/var/www/html"
           name: mypd
     volumes:
       - name: mypd
         persistentVolumeClaim:
```

```
            claimName: nasclaim
EOF
```

至此，您就將 NAS 遠程檔案系統掛載到了您的 Pod 應用當中了。

動態儲存裝置卷

使用動態 NAS 儲存卷需要您手動安裝驅動外掛程式，並配置 NAS 掛載點。

安裝外掛程式

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-nas
provisioner: alicloud/nas
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-alicloud-nas-controller
subjects:
  - kind: ServiceAccount
    name: alicloud-nas-controller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: alicloud-disk-controller-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: alicloud-nas-controller
    spec:
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      nodeSelector:
         node-role.kubernetes.io/master: ""
      serviceAccount: alicloud-nas-controller
```

```
      containers:
        - name: alicloud-nas-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-nas-
controller:v1.8.4
          volumeMounts:
          - mountPath: /persistentvolumes
            name: nfs-client-root
          env:
            - name: PROVISIONER_NAME
              value: alicloud/nas
            - name: NFS_SERVER
              value: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            - name: NFS_PATH
              value: /
      volumes:
        - name: nfs-client-root
          nfs:
            server: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            path: /
```

使用動態儲存裝置卷

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  volumeClaimTemplates:
  - metadata:
      name: html
    spec:
      accessModes:
        - ReadWriteOnce
      storageClassName: alicloud-nas
      resources:
        requests:
          storage: 2Gi
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        volumeMounts:
        - mountPath: "/usr/share/nginx/html/"
          name: html
```

1.9.5 使用阿里雲 OSS

您可以在阿里雲Container Service Kubernetes 叢集中使用阿里雲 OSS 資料卷。

目前，僅支援 OSS 靜態儲存卷，不支援 OSS 動態儲存裝置卷。您可以通過以下方式使用 OSS 靜態儲存卷：

- 直接使用 volume 方式
- 使用 PV/PVC

前提條件

使用 OSS 靜態儲存卷之前，您需要先在 OSS 管理主控台上建立 Bucket。

使用說明

- OSS 為共用儲存，可以同時為多個 Pod 提供共用儲存服務。
- bucket：目前只支援掛載 Bucket，不支援掛載 Bucket 下面的子目錄或檔案。
- url: OSS endpoint，掛載 OSS 的接入網域名稱。
- akId: 使用者的 access id 值。
- akSecret：使用者的 access secret 值。
- otherOpts: 掛載 OSS 時支援定製化參數輸入，格式為: `-o *** -o ***`。

注意事項

如果您的 Kubernetes 叢集是在 2018 年 2 月 6 日之前建立的，那麼您使用資料卷之間需要先安裝#####。使用 OSS 資料卷必須在部署 flexvolume 服務的時候建立 Secret，並輸入 AK 資訊。

使用 **OSS** 靜態卷

直接使用 **volume** 方式

使用*oss-deploy.yaml* 檔案建立 Pod。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-oss-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-flexvolume-oss
        image: nginx
        volumeMounts:
          - name: "oss1"
            mountPath: "/data"
      volumes:
        - name: "oss1"
          flexVolume:
            driver: "alicloud/oss"
            options:
              bucket: "docker"
              url: "oss-cn-hangzhou.aliyuncs.com"
              akId: ***
```

```
              akSecret: ***
              otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

使用 **PV/PVC**（目前不支援動態 **pv**）

步驟 **1** 建立 **PV**

您可以使用 yaml 檔案或者通過Container Service控制台介面建立 PV。

使用 **yaml** 檔案建立 **PV**

使用*oss-pv.yaml*檔案建立 PV。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-oss
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  storageClassName: oss
  flexVolume:
    driver: "alicloud/oss"
    options:
      bucket: "docker"
      url: "oss-cn-hangzhou.aliyuncs.com"
      akId: ***
      akSecret: ***
      otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

通過控制台介面建立 **OSS** 資料卷

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集 > 儲存，進入資料卷列表頁面。
3. 選擇所需的叢集，單擊頁面右上方的建立。

4. 在建立資料卷對話方塊中，配置資料卷的相關參數。
   - 資料卷類型：本樣本中為 OSS。
   - 資料卷名：建立的資料卷的名稱。資料卷名在叢集內必須唯一。本例為 pv-oss。
   - 資料卷總量：所建立資料卷的容量。
   - 訪問模式：預設為 ReadWriteMany。
   - **AccessKey ID**、**AccessKey Secret**：訪問 OSS 所需的 AccessKey。

- **Bucket ID**：您要使用的 OSS bucket 的名稱。單擊選擇**Bucket**，在彈出的對話方塊中選擇所需的 bucket 並單擊選擇。
- 訪問網域名稱：如果 Bucket 和 ECS 執行個體位於不同地區（Region），請選擇外網網域名稱；如果位於相同地區，需要根據叢集網路類型進行選擇，若是 VPC 網路，請選擇**VPC**網域名稱，若是傳統網路，請選擇內網網域名稱。
- 標籤：為該資料卷添加標籤。

5. 完成配置後，單擊建立。

步驟 **2** 建立 **PVC**

使用*oss-pvc.yaml*檔案建立 PVC。

```
kind: PersistentVolumeClaim
```

```
apiVersion: v1
metadata:
  name: pvc-oss
spec:
  storageClassName: oss
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

步驟 3 建立 **Pod**

使用*oss-pod.yaml* 建立 Pod。

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-oss-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
          - name: pvc-oss
            mountPath: "/data"
  volumes:
  - name: pvc-oss
    persistentVolumeClaim:
        claimName: pvc-oss
```

使用 **OSS** 動態磁碟區

目前暫不支援。

1.9.6 建立持久化儲存卷聲明

您可通過Container Service控制台建立持久化儲存卷聲明（PVC）。

前提条件

- 您已建立一個Kubernetes叢集，參見*##Kubernetes##*。
- 您已建立一個儲存卷，本例中使用雲端硬碟建立一個雲端硬碟儲存卷，參見*#########*。

  預設根據標籤**alicloud-pvname**將儲存聲明和儲存卷綁定，通過Container Service控制台建立資料卷時，會預設給儲存卷打上該標籤。如果儲存卷上沒有該標籤，您需要添加標籤後才可以選擇關聯這個儲存卷。

背景信息

操作步驟

1. 登入*Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 儲存聲明，進入儲存聲明列表頁面。

3. 選擇所需的叢集和命名空間，單擊右上方的建立。

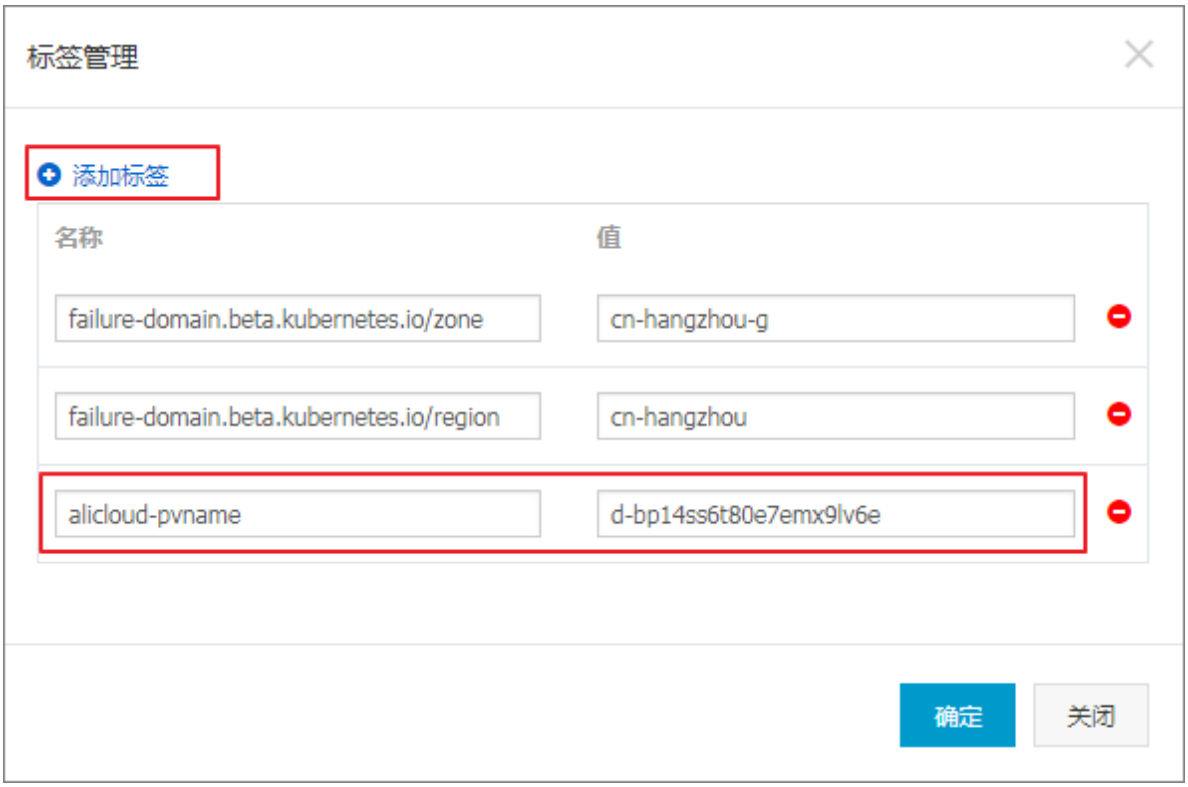
4. 在建立儲存聲明對話方塊中進行配置，最後單擊建立。

- 儲存宣告類型： 和儲存卷一致， 包括雲端硬碟/NAS/OSS
- 名稱： 輸入儲存卷聲明名稱
- 分配模式： 目前只支援已有儲存卷
- 已有儲存卷： 選擇該類型下的儲存卷綁定
- 總量： 聲明使用量，不能大於儲存卷的總量

说明：

若您的叢集中已有儲存卷，且未被使用，但在選擇已有儲存卷無法找到，則可能是未定義**alicloud-pvname**標籤。

若無法找到可用的儲存卷，您可在左側導覽列中單擊叢集 > 儲存卷，找到所需儲存卷，單擊右側的標籤管理，添加對應標籤，其中名稱為**alicloud-pvname**，值為儲存卷的名稱，雲端硬碟儲存卷預設以雲端硬碟ID作為儲存卷的名稱。



5. 返回儲存聲明列表，您可看到建立的儲存聲明出現在列表中。

1.9.7 使用持久化儲存卷聲明

您可通過Container Service控制台，通過鏡像或範本部署應用，從而使用持久化儲存聲明。本例中使用鏡像來建立應用，若您想通過模板使用持久化儲存卷聲明，請參見#########。

前提条件

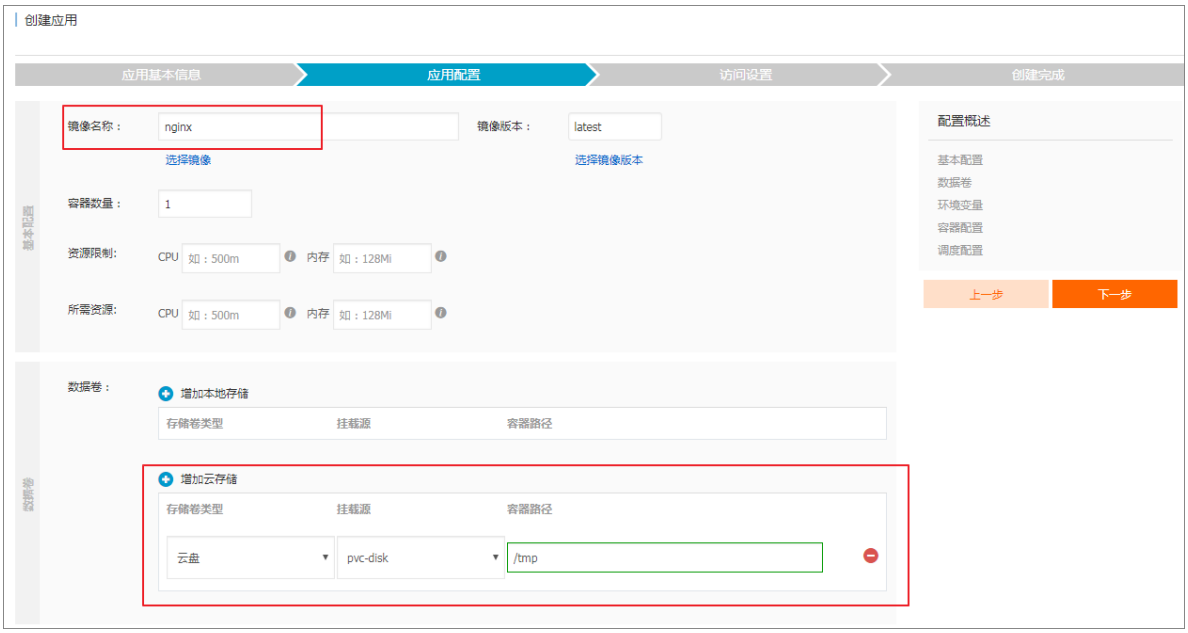
- 您已建立一個Kubernetes叢集，參見*##Kubernetes##*。
- 您已建立一個儲存卷聲明，本例中使用雲端硬碟建立一個雲端硬碟儲存卷聲明pvc-disk，參見*########*。

操作步驟

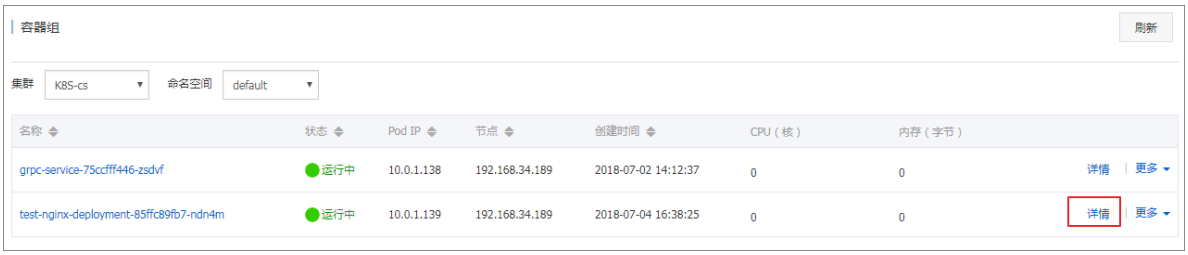
1. 登入*Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面，單擊右上方的使用鏡像建立。

3. 在應用基本資料配置頁面，設定應用程式名稱、部署叢集和命名空間，然後單擊下一步。

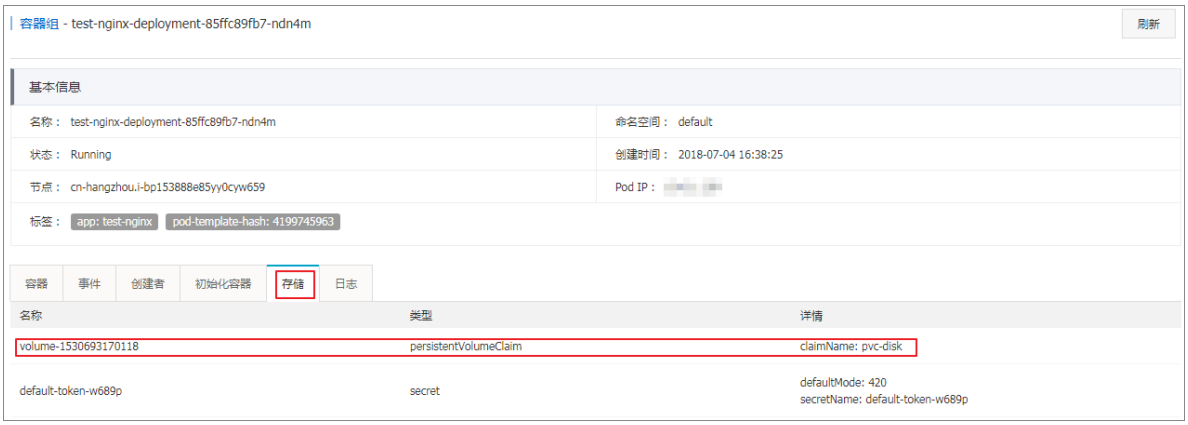
4. 在應用配置頁面，選擇鏡像，然後配置雲端儲存類型的資料卷，支援雲端硬碟/NAS/OSS 三種類型。本例中使用準備好的雲端硬碟儲存卷聲明，最後單擊下一步。



5. 配置test-nginx應用的服務，最後單擊建立。
6. 應用建立完畢後，單擊左側導覽列中的應用 > 容器組，找到該應用所屬容器組，單擊詳情。



7. 在容器組詳情頁面，單擊儲存，您可看到該容器組正確綁定了pvc-disk。



1.10 日誌管理

1.10.1 概述

阿里雲Container Service Kubernetes 叢集提供給您多種方式進行應用的日誌管理。

- 通過阿里雲Log Service*#unique_97*，您可以方便地使用Log Service採集應用日誌，從而更好地利用阿里雲Log Service提供給您的各種日誌統計分析等功能。
- 通過阿里雲Container Service提供的開源 *Log-pilot* 項目，*#unique_98*，您可以方便地搭建自己的應用日誌叢集。

1.10.2 查看叢集日誌

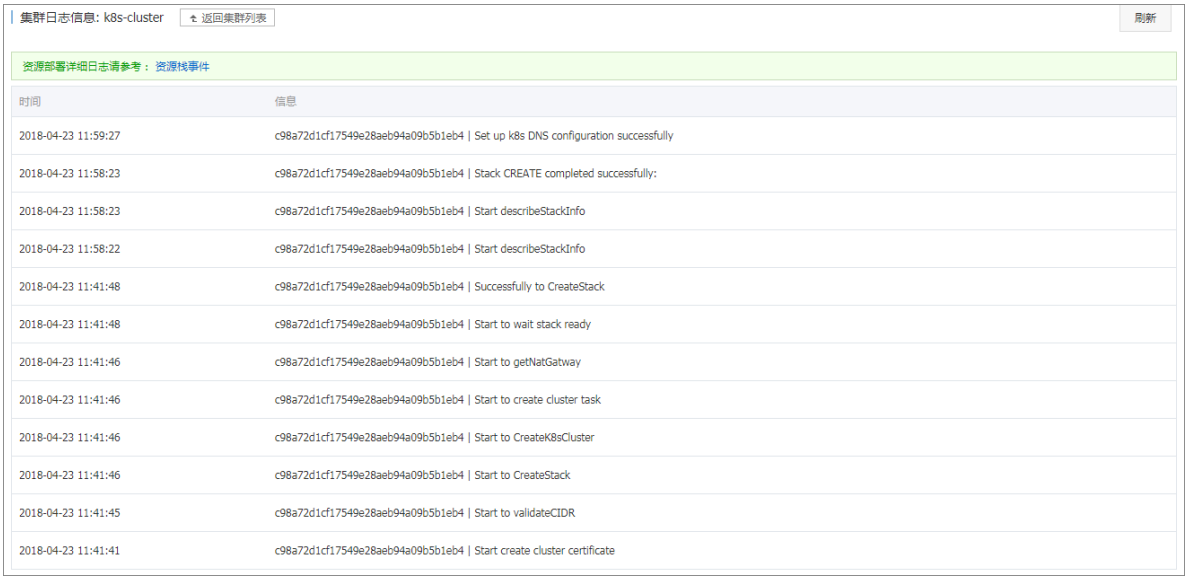
背景信息

您可以通過Container Service的簡單Log Service查看叢集的動作記錄。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，進入 Kubernetes 叢集列表頁面。
3. 選擇所需的叢集並單擊右側的查看日誌。

您可以查看該叢集的操作資訊。



1.10.3 為 Kubernetes 和Log Service配置 Log4JAppender

Log4j 是 Apache 的一個開放原始碼項目。Log4j 由三個重要組件構成：日誌資訊的優先順序、日誌資訊的輸出目的地、日誌資訊的輸出格式。通過配置 Log4jAppender，您可以控制日誌資訊輸送的目的地是控制台、檔案、GUI 組件、甚至是套介面伺服器、NT 的事件記錄器、UNIX Syslog 守護進程等。

本文介紹在不需要修改應用代碼的前提下，通過配置一個 yaml 檔案，將阿里雲Container Service Kubernetes 叢集中產生的日誌輸出到阿里雲Log Service。此外，通過在 Kubernetes 叢集上部署一個樣本 API 程式，來進行示範。

前提條件

- 您已經開通Container Service，並且建立了 Kubernetes 叢集。

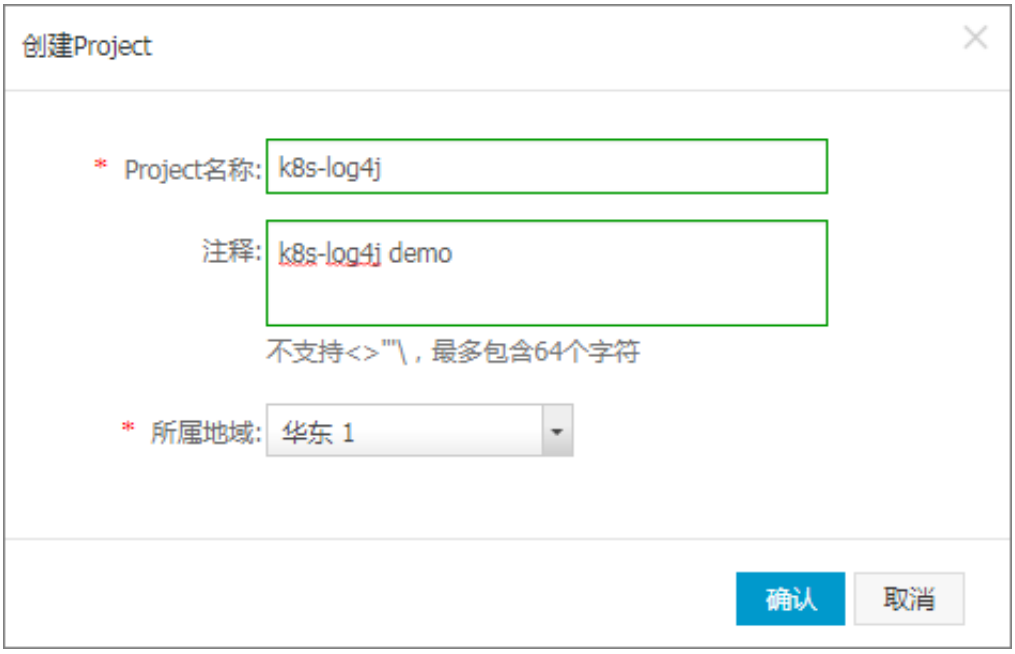
  本樣本中，建立的 Kubernetes 叢集位於華東 1 地區。
- 啟用 AccessKey 或 RAM，確保有足夠的存取權限。本例中使用 AccessKey。

步驟 1 在阿里雲**Log Service**上配置 **Log4jAppender**

1. 登入 *Log Service#####*。
2. 單擊頁面右上方的建立 **Project**，填寫 Project 的基本資料並單擊確認進行建立。

本樣本建立一個名為 k8s-log4j，與 Kubernetes 叢集位於同一地區（華東 1）的 Project。

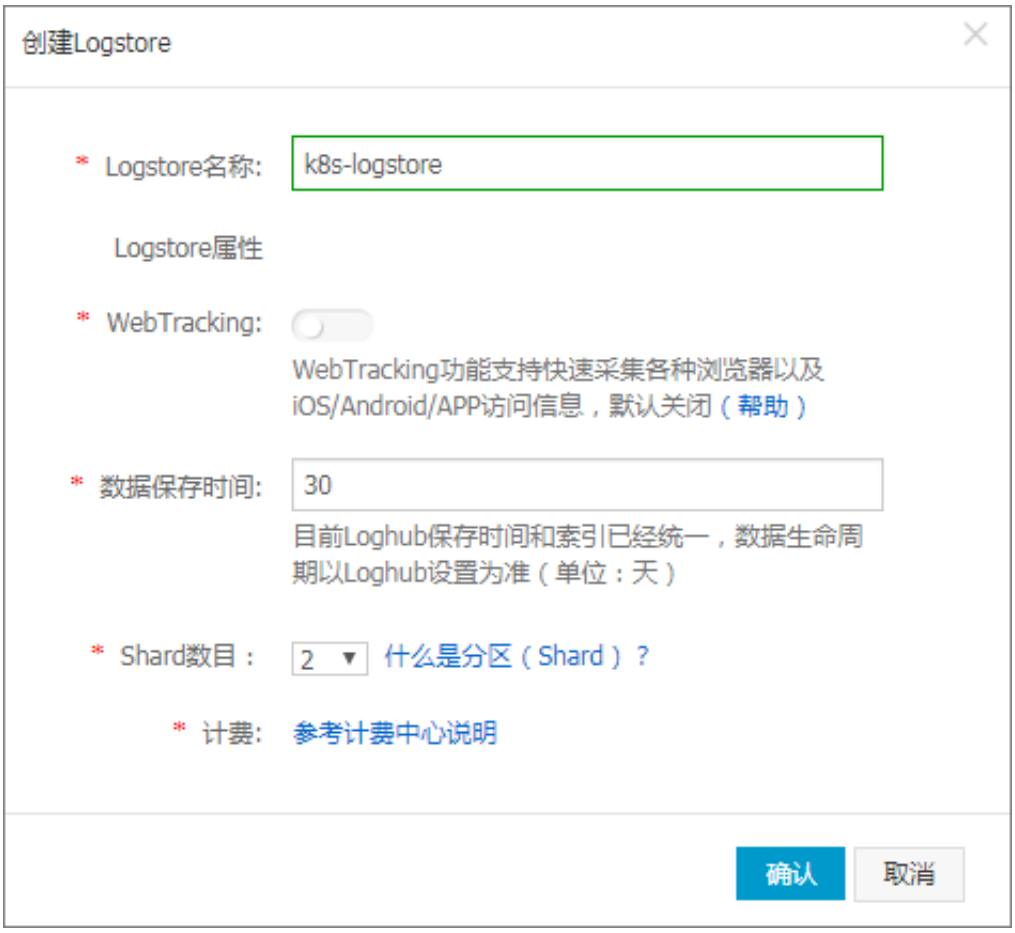
> 说明：
> 在配置時，一般會使用與 Kubernetes 叢集位於同一地區的Log Service Project。因為當 Kubernetes 叢集和Log Service Project 位於同一地區時，日誌資料會通過內網進行傳輸，從而避免了因地區不一致而導致的資料轉送外網頻寬費用和耗時，從而實現即時採集、快速檢索的最佳實務。



3. 建立完成後，k8s-log4j 出現在 project 列表下，單擊該 project 名稱，進入 project 詳情頁面。
4. 預設進入日誌庫頁面，單擊右上方的建立。

5. 填寫日誌庫配置資訊並單擊確認。

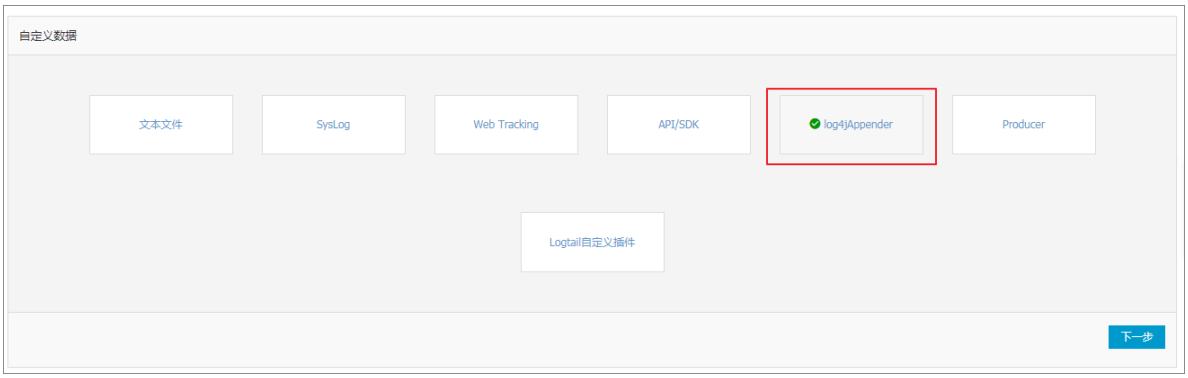
   本樣本建立名為 k8s-logstore 的日誌庫。



6. 建立完畢後，頁面會提示您建立資料接入嚮導。

7. 選擇自訂資料下的log4jAppender，根據頁面引導進行配置。

   本樣本使用了預設配置，您可根據日誌資料的具體使用情境，進行相應的配置。



步驟 2 在 Kubernetes 叢集中配置 log4j

本樣本使用 *demo-deployment* 和 *demo-Service* 樣本 yaml 檔案進行示範。

1. 串連到您的 Kubernetes 叢集。

   具體操作參見*SSH##Kubernetes##* 或 *## kubectl ## Kubernetes ##*。

2. 擷取*demo-deployment.yaml*檔案並配置環境變數 `JAVA_OPTS` 設定 Kubernetes 叢集日誌的採集。

   demo-deployment.yaml 檔案的樣本編排如下。

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: log4j-appender-demo-spring-boot
  labels:
    app: log4j-appender
spec:
  replicas: 1
  selector:
    matchLabels:
      app: log4j-appender
  template:
    metadata:
    labels:
      app: log4j-appender
  spec:
    containers:
    - name: log4j-appender-demo-spring-boot
      image: registry.cn-hangzhou.aliyuncs.com/jaegertracing/log4j-
appender-demo-spring-boot:0.0.2
      env:
      - name: JAVA_OPTS                    ##注意
        value: "-Dproject={your_project} -Dlogstore={your_logstore
} -Dendpoint={your_endpoint} -Daccess_key_id={your_access_key_id} -
Daccess_key={your_access_key_secret}"
      ports:
      - containerPort: 8080
```

其中：

- `-Dproject`：您所使用的阿里雲Log Service Project 的名稱。本樣本中為 k8s-log4j。

- `-Dlogstore`：您所使用的阿里雲Log Service Logstore 的名稱。本樣本中為 k8s-logstore。
- `-Dendpoint`：Log Service的服務入口，使用者需要根據日誌 Project 所屬的地區，配置自己的服務入口，參見####進行查詢。本樣本中為 cn-hangzhou.log.aliyuncs.com。
- `-Daccess_key_id`：您的 AccessKey ID。
- `-Daccess_key`：您的 AccessKey Secret。

3. 在命令列中執行以下命令，建立 deployment。

```
kubectl create -f demo-deployment.yaml
```

4. 擷取*demo-Service.yaml*檔案，並運行以下命令建立 service。

   您不需要修改*demo-Service.yaml*中的配置。

```
kubectl create -f demo-service.yaml
```

步驟 3 測試產生 **Kubernetes** 叢集日誌

您可以使用 `kubectl get` 命令查看資來源物件部署狀況，等待 deployment 和 service 部署成功後，執行 `kubectl get svc` 查看 service 的外部存取 IP，即 EXTERNAL-IP。

```
$ kubectl get svc
NAME                         TYPE           CLUSTER-IP    EXTERNAL-IP
  PORT(S)          AGE
log4j-appender-demo-spring-boot-svc   LoadBalancer   172.21.XX.XX
120.55.XXX.XXX   8080:30398/TCP   1h
```

在本樣本中，通過運行`login`命令來測試產生 Kubernetes 叢集日誌。其中 `K8S_SERVICE_IP` 即為 `EXTERNAL-IP`。

说明：

您可以在 *GitHub log4j-appender-demo* 中查看完整的 API 集合。

```
curl http://${K8S_SERVICE_IP}:8080/login?name=bruce
```

步驟 4 在阿里雲**Log Service**中查看日誌

登入 *Log Service#####*。

進入對應的 Project 的詳情頁面，選擇對應的日誌庫 k8s-logstore，並單擊右側的查詢分析 **-** 查詢，查看 Kubernetes 叢集輸出的日誌，如下所示。

日誌的輸出內容對應上面的命令。本樣本示範了將樣本應用產生的日誌輸出到阿里雲Log Service的操作。通過這些操作，您可以在阿里雲上配置 Log4JAppender，並通過阿里雲Log Service，實現日誌即時搜集、資料過濾、檢索等進階功能。

1.10.4 利用 log-pilot + elasticsearch + kibana 搭建 kubernetes 日誌解決方案

開發人員在面對 kubernetes 分布式叢集下的日誌需求時，常常會感到頭疼，既有容器自身特性的原因，也有現有日誌採集工具的桎梏，主要包括：

- 容器本身特性：
  - 採集目標多：容器本身的特性導致採集目標多，需要採集容器內日誌、容器 stdout。對於容器內部的檔案日誌採集，現在並沒有一個很好的工具能夠去動態發現採集。針對每種資料來源都有對應的採集軟體，但缺乏一站式的工具。
  - Auto Scaling難：kubernetes 是分布式的叢集，服務、環境的Auto Scaling對於日誌採集帶來了很大的困難，無法像傳統虛擬機器環境下那樣，事先配置好日誌的採集路徑等資訊，採集的動態性以及資料完整性是非常大的挑戰。
- 現有日誌工具的一些缺陷：
  - 缺乏動態配置的能力。目前的採集工具都需要事先手動設定好日誌採集方式和路徑等資訊，因為它無法能夠自動感知到容器的生命週期變化或者動態漂移，所以它無法動態地去配置。
  - 日誌採集重複或丟失的問題。因為現在的一些採集工具基本上是通過 tail 的方式來進行日誌採集的，那麼這裡就可能存在兩個方面的問題：一個是可能導致日誌丟失，比如採集工具在重啟的過程中，而應用依然在寫日誌，那麼就有可能導致這個視窗期的日誌丟失；而對於這種情況一般保守的做法就是，預設往前多採集 1M 日誌或 2M 的日誌，那麼這就又會可能引起日誌採集重複的問題。
  - 未明確標記日誌源。因為一個應用可能有很多個容器，輸出的應用日誌也是一樣的，那麼當我們將所有應用日誌收集到統一日誌儲存後端時，在搜尋日誌的時候，我們就無法明確這條日誌具體是哪一個節點上的哪一個應用程式容器產生的。

本文檔將介紹一種 Docker 日誌收集工具 log-pilot，結合 Elasticsearch 和 kibana 等工具，形成一套適用於 kubernetes 環境下的一站式日誌解決方案。

log-pilot 介紹

log-Pilot 是一個智能容器日誌採集工具，它不僅能夠高效便捷地將容器日誌採集輸出到多種儲存日誌後端，同時還能夠動態地發現和採集容器內部的記錄檔。

針對前面提出的日誌採集難題，log-pilot 通過聲明式配置實現強大的容器事件管理，可同時擷取容器標準輸出和內部檔案日誌，解決了動態伸縮問題，此外，log-pilot 具有自動探索機制，CheckPoint 及控制代碼保持的機制，自動日誌資料打標，有效應對動態配置、日誌重複和丟失以及日誌源標記等問題。

目前 log-pilot 在 Github 完全開源，項目地址是 *https://github.com/AliyunContainerService/log-pilot* 。您可以深入瞭解更多實現原理。

針對容器日誌的聲明式配置

Log-Pilot 支援容器事件管理，它能夠動態地監聽容器的事件變化，然後依據容器的標籤來進行解析，組建記錄檔採集設定檔，然後交由採集外掛程式來進行日誌採集。

在 kubernetes 下，Log-Pilot 可以依據環境變數 `aliyun_logs_$name = $path` 動態地組建記錄檔採集設定檔，其中包含兩個變數：

- $name 是我們自訂的一個字串，它在不同的情境下指代不同的含義，在本情境中，將日誌採集到 Elasticsearch 的時候，這個 $name 表示的是 Index。
- 另一個是 $path，支援兩種輸入形式，stdout 和容器內部記錄檔的路徑，對應日誌標準輸出和容器內的記錄檔。
  - 第一種約定關鍵字 stdout 表示的是採集容器的標準輸出日誌，如本例中我們要採集 tomcat 容器日誌，那麼我們通過配置標籤 `aliyun.logs.catalina=stdout` 來採集 tomcat 標準輸出日誌。
  - 第二種是容器內部記錄檔的路徑，也支援萬用字元的方式，通過配置環境變數 `aliyun_logs_access=/usr/local/tomcat/logs/*.log`來採集 tomcat 容器內部的日誌。當然如果你不想使用 aliyun 這個關鍵字，Log-Pilot 也提供了環境變數 PILOT_LOG_PREFIX 可以指定自己的聲明式日誌配置首碼，比如 `PILOT_LOG_PREFIX: "aliyun,custom"`。

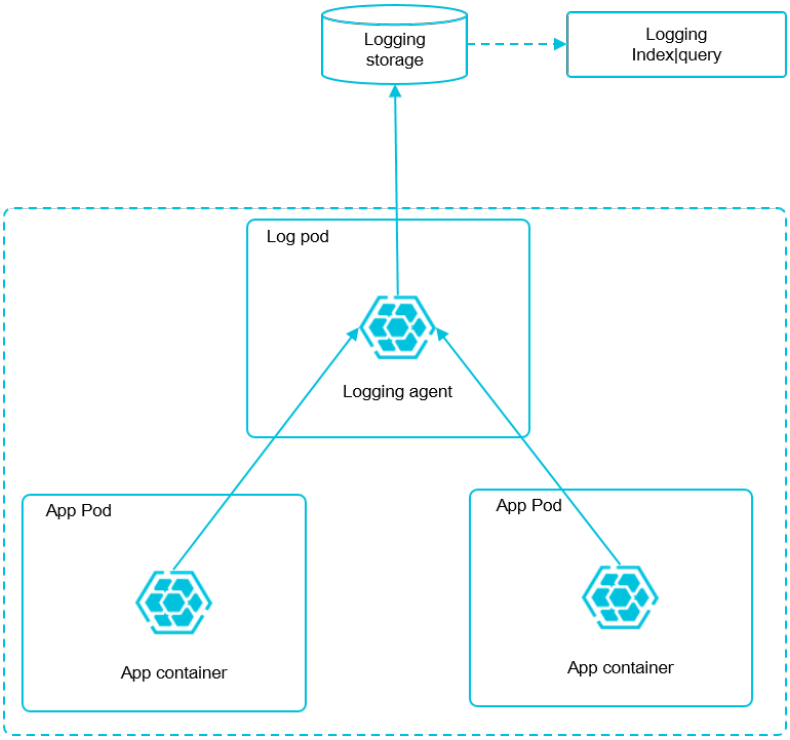
此外，Log-Pilot 還支援多種日誌解析格式，通過 `aliyun_logs_$name_format=<format>` 標籤就可以告訴 Log-Pilot 在採集日誌的時候，同時以什麼樣的格式來解析日誌記錄，支援的格式包括：none、json、csv、nginx、apache2 和 regxp。

Log-Pilot 同時支援自訂 tag，我們可以在環境變數裡配置 `aliyun_logs_$name_tags="K1=V1,K2=V2"`，那麼在採集日誌的時候也會將 K1=V1 和 K2=V2 採集到容器的日誌輸出中。自訂 tag 可協助您給日誌產生的環境打上 tag，方便進行日誌統計、日誌路由和日誌過濾。

日誌採集模式

本文檔採用 node 方式進行部署，通過在每台機器上部署一個 log-pilot 執行個體，收集機器上所有 Docker 應用日誌。

該方案跟在每個 Pod 中都部署一個 logging 容器的模式相比，最明顯的優勢就是佔用資源較少，在叢集規模比較大的情況下表現出的優勢越明顯，這也是社區推薦的一種模式。



前提條件

您已經開通Container Service，並建立了一個 kubernetes 叢集。本樣本中，建立的 Kubernetes 叢集位於華東 1 地區。

步驟1 部署 **elasticsearch**

1. 串連到您的 Kubernetes 叢集。具體操作參見通過*SSH##Kubernetes##* 或 *## kubectl ## Kubernetes ##*。

2. 首先部署 elasticsearch 相關服務，該編排模板包含一個 elasticsearch-api 的服務、elasticsearch-discovery 的服務和 elasticsearch 的狀態集，這些對象都會部署在 kube-system 命名空間下。

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
elasticsearch.yml
```

3. 部署成功後，kube-system 命名空間下會出現相關對象，執行以下命令查看運行情況。

```
$ kubectl get svc,StatefulSet -n=kube-system
 NAME                           TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)                        AGE
 svc/elasticsearch-api          ClusterIP      172.21.5.134    <none>
          9200/TCP                      22h
 svc/elasticsearch-discovery    ClusterIP      172.21.13.91    <none>
          9300/TCP                      22h
 ...
 NAME                           DESIRED   CURRENT   AGE
 statefulsets/elasticsearch     3      3      22h
```

步驟2 部署 **log-pilot** 和 **kibana** 服務

1. 部署 log-pilot 日誌採集工具，如下所示：

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
log-pilot.yml
```

2. 部署 kibana 服務，該編排樣本包含一個 service 和一個 deployment。

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
kibana.yml
```

步驟3 部署測試應用 **tomcat**

在 elasticsearch + log-pilot + Kibana 這套日誌工具部署完畢後，現在開始部署一個日誌測試應用 tomcat，來測試日誌是否能正常採集、索引和顯示。

編排模板如下。

```
apiVersion: v1
kind: Pod
metadata:
  name: tomcat
  namespace: default
  labels:
    name: tomcat
spec:
  containers:
  - image: tomcat
    name: tomcat-test
    volumeMounts:
    - mountPath: /usr/local/tomcat/logs
      name: accesslogs
    env:
     - name: aliyun_logs_catalina
```

```
        value: "stdout"                                                    ##
採集標準輸出日誌
      - name: aliyun_logs_access
        value: "/usr/local/tomcat/logs/catalina.*.log"                     ##
 採集容器內記錄檔
  volumes:
    - name: accesslogs
      emptyDir: {}
```

tomcat 鏡像屬於少數同時使用了 stdout 和檔案日誌的 Docker 鏡像，適合本文檔的示範。在上面的編排中，通過在 pod 中定義環境變數的方式，動態地組建記錄檔採集設定檔，環境變數的具體說明如下：

- `aliyun_logs_catalina=stdout`表示要收集容器的 stdout 日誌。
- `aliyun_logs_access=/usr/local/tomcat/logs/catalina.*.log` 表示要收集容器內 */usr/local/tomcat/logs/* 目錄下所有名字匹配 *catalina.*.log* 的檔案日誌。

在本方案的 elasticsearch 情境下，環境變數中的 `$name` 表示 Index，本例中 `$name`即是 catalina 和 access 。

步驟 **4** 將 **kibana** 服務暴露到公網

上面部署的 kibana 服務的類型採用 NodePort，預設情況下，不能從公網進行訪問，因此本文檔配置一個 ingress 實現公網訪問 kibana，來測試日誌資料是否正常索引和顯示。

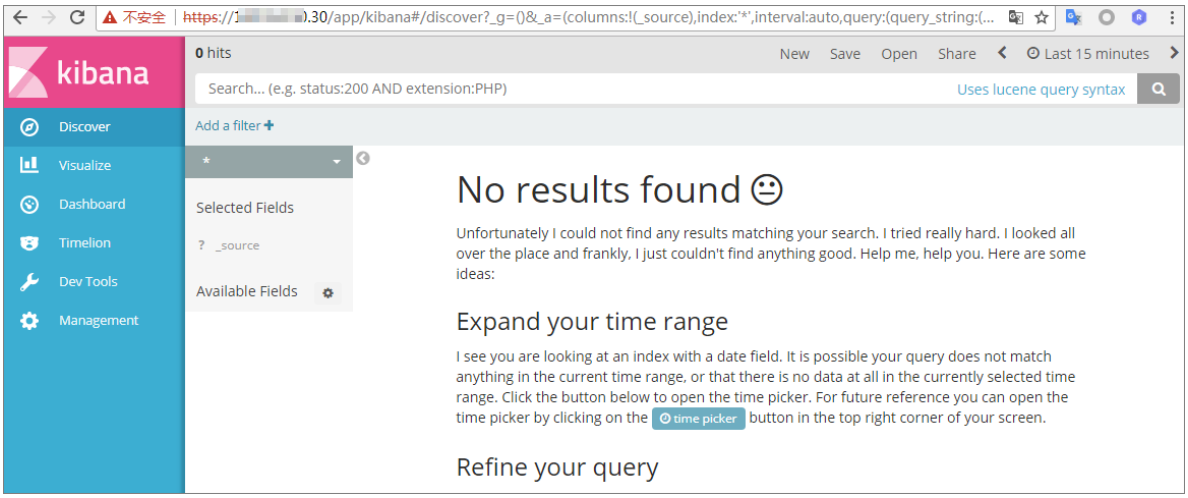
1. 通過配置 ingress 來實現公網下訪問 kibana 。本樣本選擇簡單的路由服務來實現，您可參考*Ingress ##* 擷取更多方法。該 ingress 的編排模板如下所示。

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
   name: kibana-ingress
   namespace: kube-system                          #要與 kibana 服務處
於同一個 namespace
spec:
   rules:
   - http:
       paths:
       - path: /
         backend:
           serviceName: kibana                     #輸入 kibana 服務的
名稱
           servicePort: 80                         #輸入 kibana 服務暴
露的連接埠
```

2. 建立成功後，執行以下命令，擷取該 ingress 的訪問地址。

```
$ kubectl get ingress -n=kube-system
 NAME         HOSTS     ADDRESS         PORTS     AGE
 shared-dns   *         120.55.150.30   80        5m
```

3. 在瀏覽器中訪問該地址，如下所示。

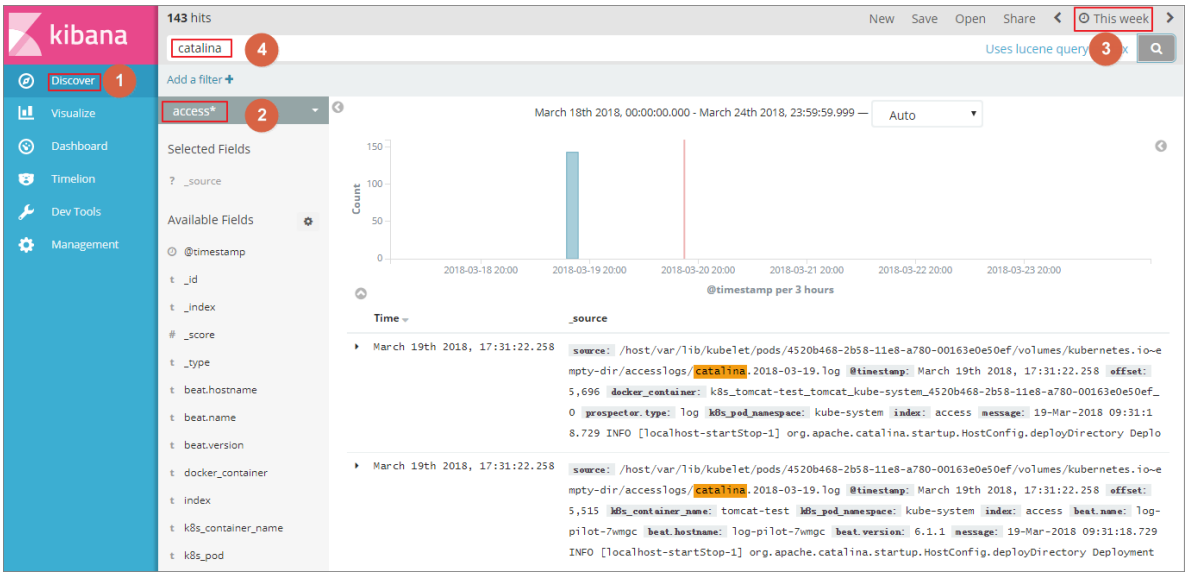


4. 單擊左側導覽列中的**management** ，然後單擊**Index Patterns** > **Create Index Pattern**。具體的索引名稱會在 `$name` 變數尾碼一個時間字串，您可以配合萬用字元 `*` 進行建立。本例中使用 `$name*` 來建立 Index Pattern。

   您也可以執行以下命令，進入 elasticsearch 對應的 pod，在 index 下列出 elasticsearch 的所有索引。

```
$ kubectl get pods -n=kube-system                    #找到 elasticsearch
 對應的 pod
   ...
$ kubectl exec -it  elasticsearch-1 bash              #進入
elasticsearch 的一個 pod
 ...
$ curl 'localhost:9200/_cat/indices?v'                  ## 列出所有索
引
 health status index                uuid                   pri rep
docs.count docs.deleted store.size pri.store.size
 green  open   .kibana             x06jj19PS4Cim6Ajo51PWg   1   1
       4            0     53.6kb         26.8kb
 green  open   access-2018.03.19   txd3tG-NR6-guqmMEKKzEw   5   1
     143            0    823.5kb        411.7kb
 green  open   catalina-2018.03.19 ZgtWd16FQ7qqJNNWXxFPcQ   5   1
     143            0    915.5kb        457.5kb
```

5. 索引建立完畢後，單擊左側導覽列中的**Discover**，然後選擇前面建立的 Index，選擇合適的時間段，在搜尋欄輸入相關欄位，就可以查詢相關的日誌。



至此，在阿里雲 Kubernetes 叢集上，我們已經成功測試基於 log-pilot、elasticsearch 和 kibana 的日誌解決方案，通過這個方案，我們能有效應對分布式 kubernetes 叢集日誌需求，可以協助提升營運和運營效率，保障系統持續穩定運行。

1.10.5 Kubernetes與Log Service整合與使用

阿里雲Container ServiceKubernetees叢集整合了Log Service（SLS），您可在建立叢集時啟用Log Service，隨後只要對應用進行簡單配置，即可無縫使用阿里雲Log Service。

前提條件

請確保開通了阿里雲*Log Service*。

步驟1 安裝日誌組件

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的叢集，進入叢集列表頁面，單擊右上方建立**Kubernetes**叢集。

3. 在建立叢集頁面，選擇**Kubernetes**叢集模式。勾選使用**SLS**，您可選擇已有日誌project；或者直接建立一個，預設建立名稱為`k8s-log-{ClusterID}`的`project`。其他叢集配置參數，請參見*##Kubernetes##*。

4. 最後單擊右上方建立叢集，啟動部署，等叢集部署完成。
5. 單擊左側導覽列中應用 > 部署，進入部署列表，選擇所需的叢集和kube-system命名空間，您可看到alibaba-log-controller組件為運行狀態。

6. 進入Log Service管理主控台，您可看到日誌project被建立出來。

步驟2 在部署頁配置採集資訊

1. 登入 *Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面，單擊右上方使用鏡像建立。
3. 進入應用基本資料頁面，配置應用基本資料，包括應用程式名稱、部署叢集和命名空間，最後單擊下一步。

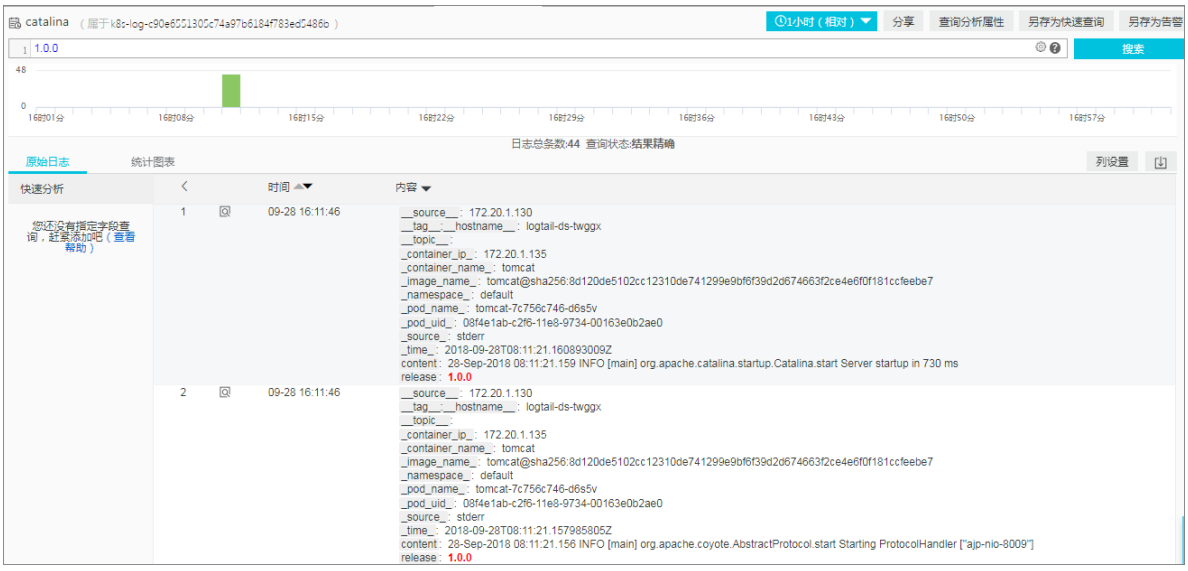
4. 在應用配置頁面，首先配置鏡像名稱、鏡像版本、容器數量。本例中建立一個tomcat應用，來測試收集容器標準輸出日誌，以及容器內文本日誌。

> 说明：
> tomcat 鏡像屬於少數同時使用 stdout 和檔案日誌的容器鏡像，適合本例中的示範。

5. 配置資料卷，將容器內的日誌目錄掛載到臨時目錄，用於儲存文本日誌。

> 说明：
> 您也可使用其他儲存儲存容器文本日誌，如雲端硬碟、NAS等雲端儲存。

6. 配置Log Service，您可進行採集配置和自訂Tag設定。



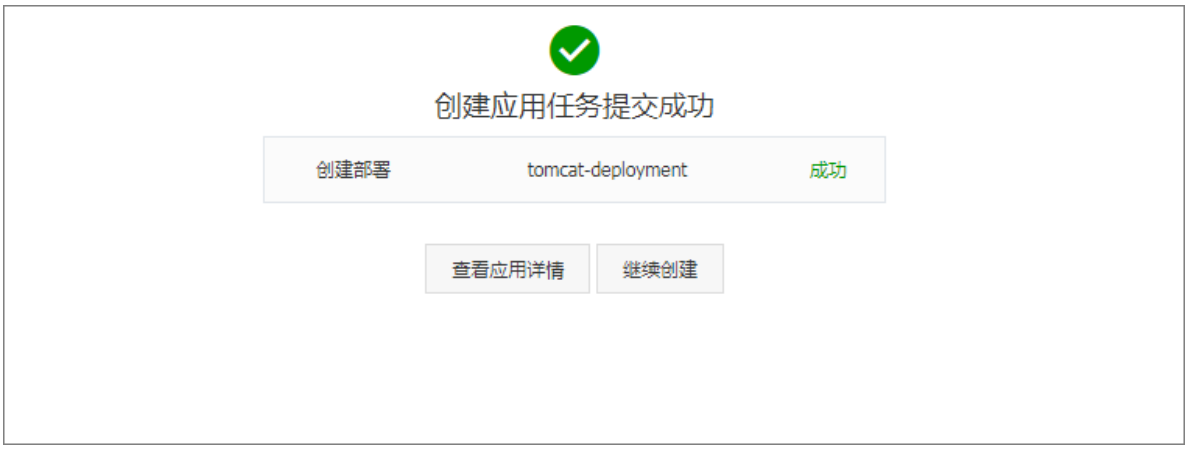
您可對日誌進行採集配置：

- 日誌庫：即在Log Service中產生一個對應的logstore，用於儲存採集到的日誌。

- 容器內日誌路徑：支援stdout和文本日誌。
  - **stdout**： stdout 表示採集容器的標準輸出日誌。
  - 文本日誌：表示收集容器內指定路徑的日誌，支援萬用字元的方式，本例中表示收集符合/usr/local/tomcat/logs/catalina.*.log名稱規則的文本日誌。

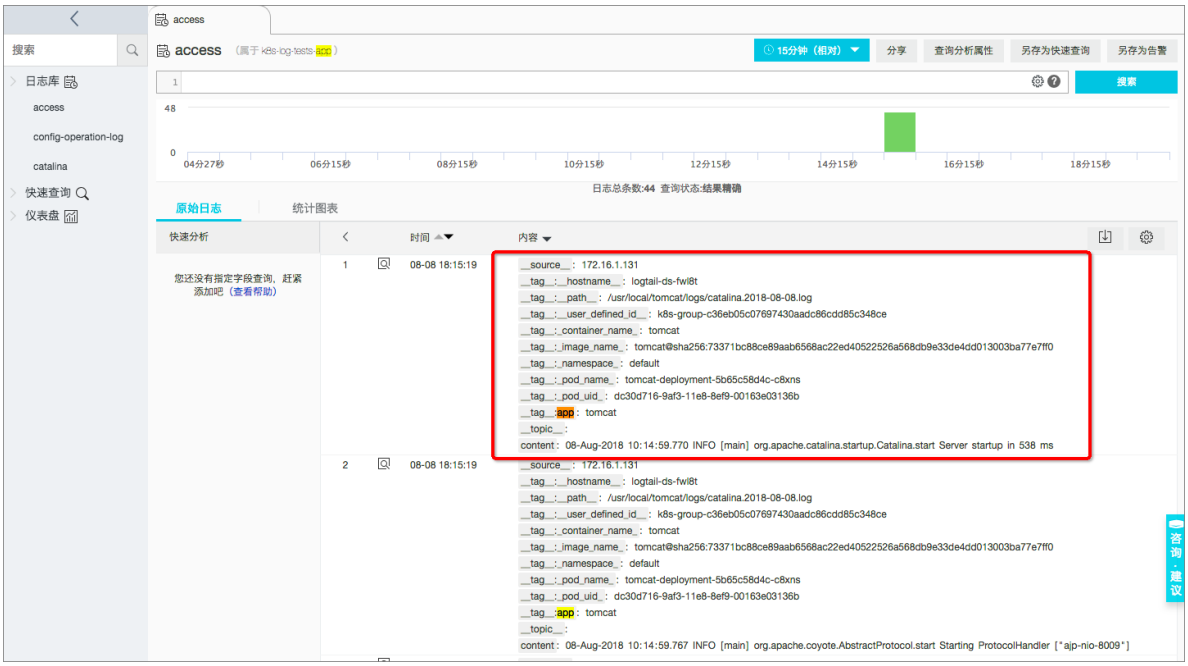
  您還可設定自訂 tag，設定tag後，會將該tag一起採集到容器的日誌輸出中。自訂 tag 可協助您給容器日誌打上tag，方便進行日誌統計和過濾等分析操作。

7. 完成應用配置後，單擊下一步。
8. 進入訪問設定頁面，本例中不進行訪問設定，單擊建立，頁面提示建立成功。



9. 進入Log Service控制台，在對應的日誌project中，您可看到成功建立的logstore。選擇所需的logstore，單擊查詢。

10. 頁面跳轉到日誌列表頁，選擇日誌查詢時間，每條日誌都被打上了tag `app=tomcat`。



1.11 監控管理

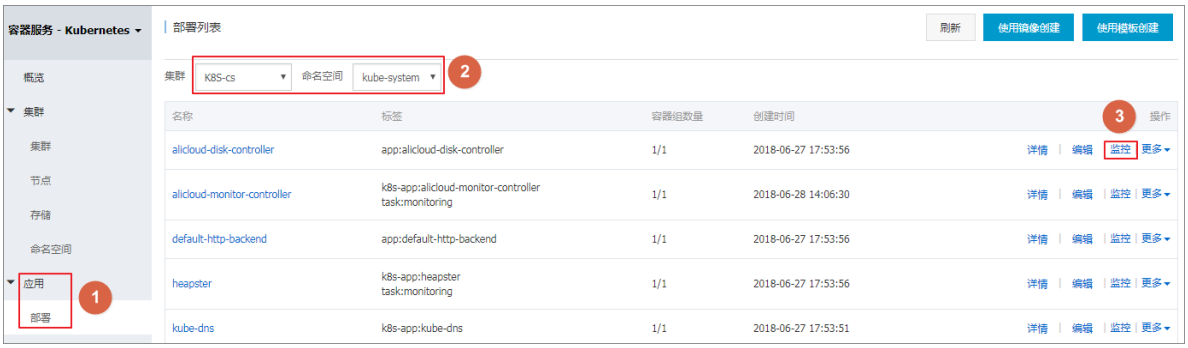
1.11.1 與Cloud Monitor整合與使用

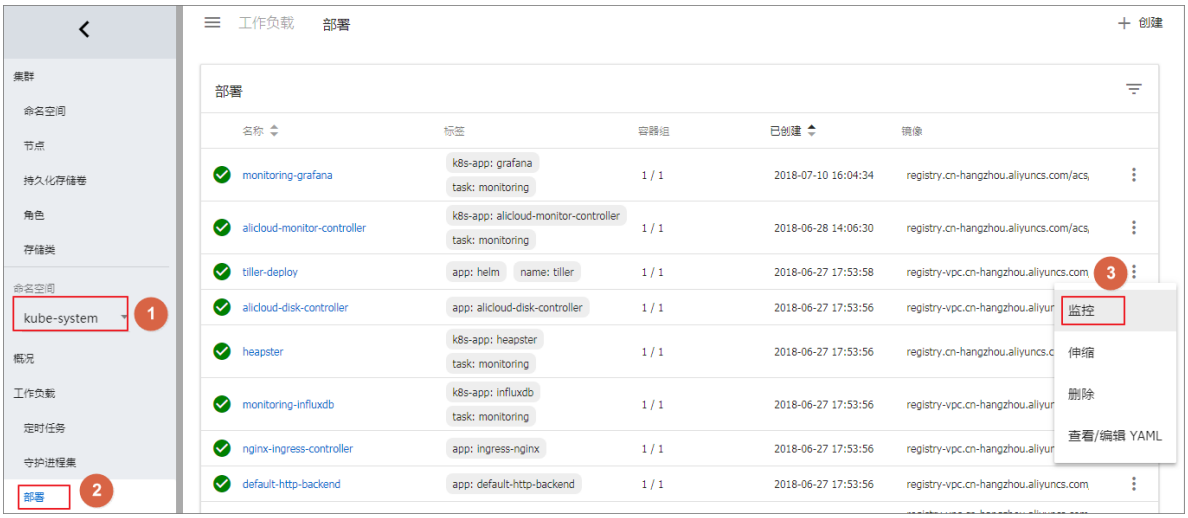
前提條件

請先檢查kube-system命名空間下是否已經部署了alicloud-monitor-controller，若未部署，請進行舊版本叢集升級。

使用方式

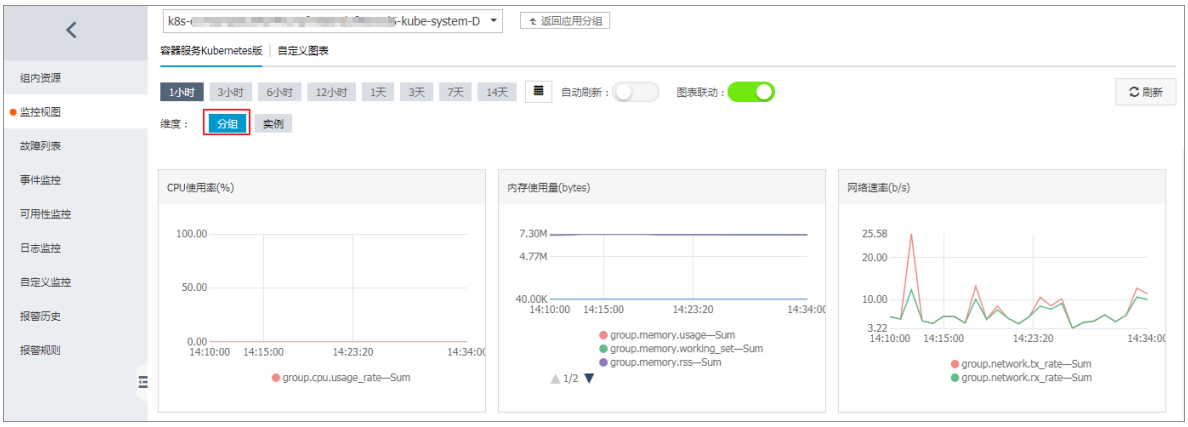
1. 登入 *Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的部署，進入部署列表頁面。
3. 選擇所需的deployment，單擊右側的監控，或者在內建的Kubernetes Dashboard的部署頁面中單擊監控。

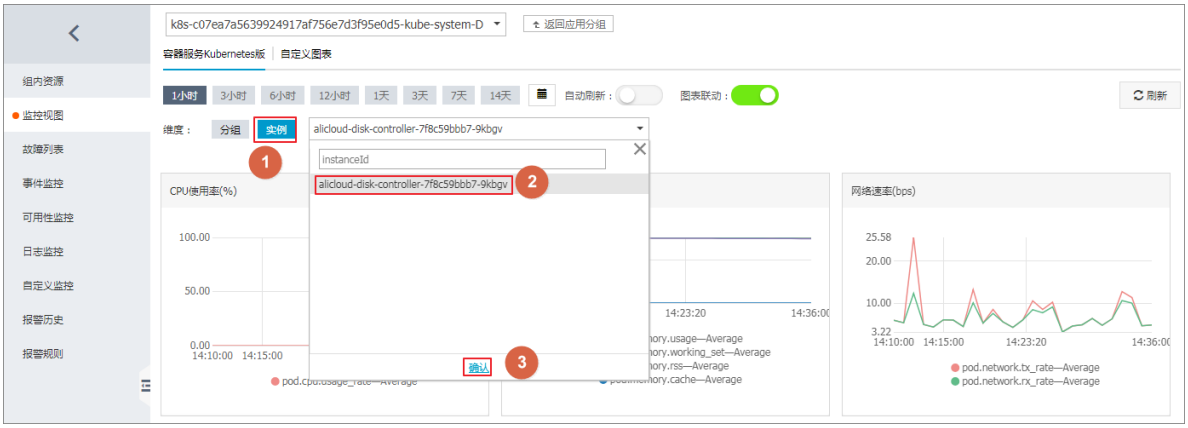




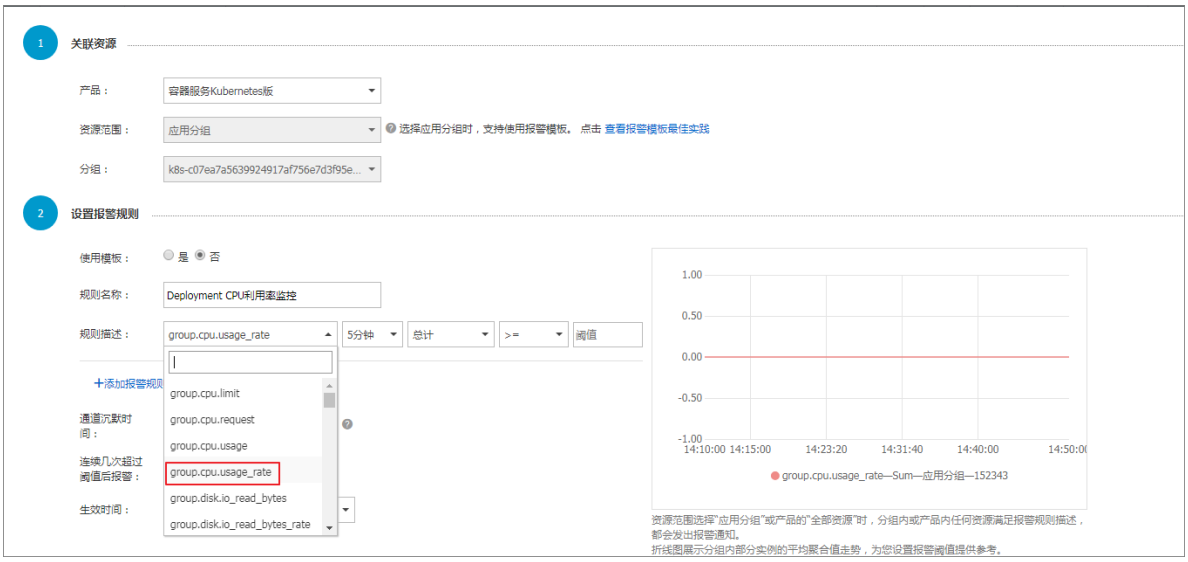
此時會跳轉到Cloud Monitor的相應的監控視圖頁面。

4. 應用分組支援分組和執行個體兩個維度監控。





5. 如需警示設定，分組層級的指標以**group**開頭，執行個體層級的指標以**pod**開頭。

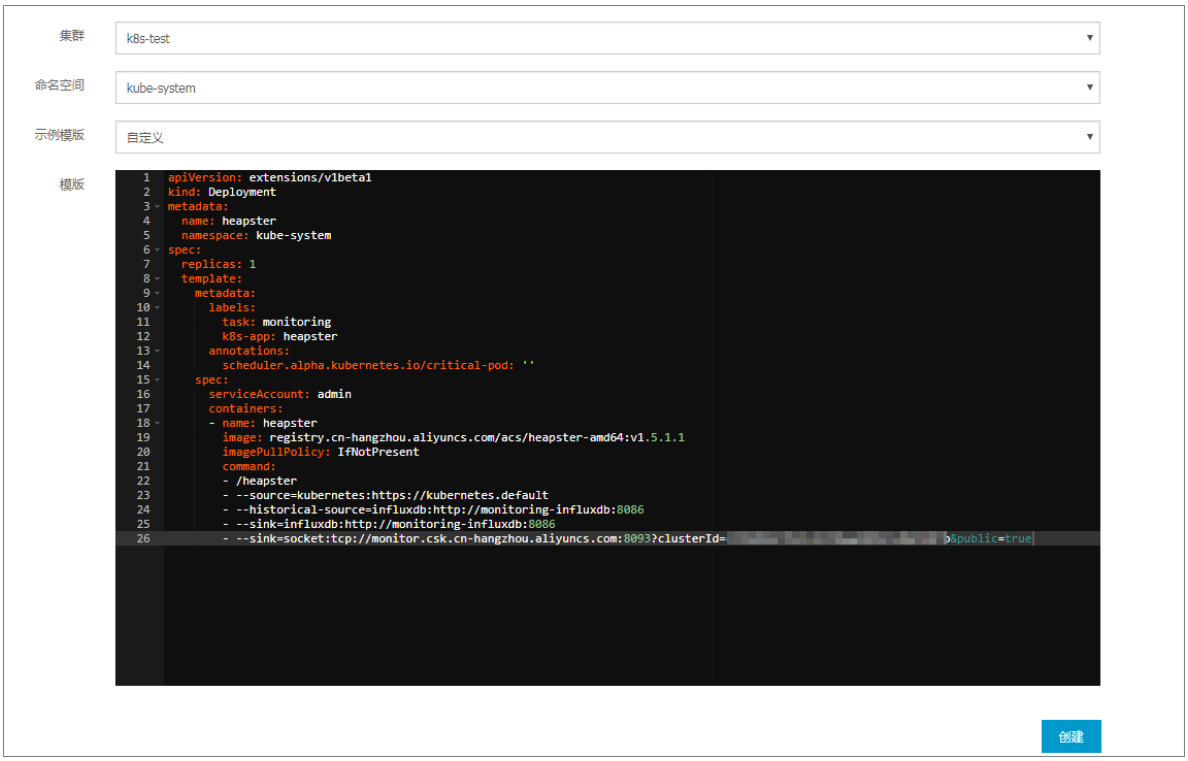


舊版本叢集升級

1. 登入 *Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面，單擊右上方的使用模板建立。

3. 選擇所需的叢集，Kube-system命名空間，使用以下的樣本模板，然後單擊建立。

> 说明：
> 根據自己的叢集替換**REGION**與**CLUSTER_ID**，並重新部署Heapster的yaml編排。



heapster樣本編排模板如下。若叢集中已有舊版本的heapster，您也可登入到Kubernetes叢集，執行`kubectl apply -f xxx.yaml`命令進行更新。

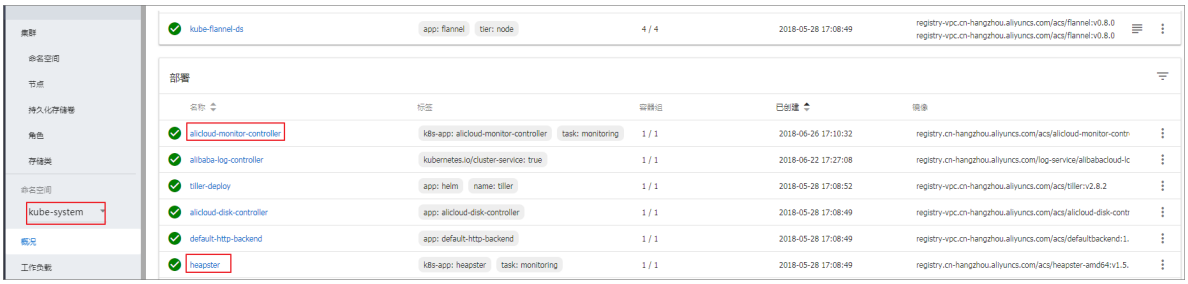
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heapster
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: heapster
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      serviceAccount: admin
      containers:
      - name: heapster
        image: registry.##REGION##.aliyuncs.com/acs/heapster-amd64:
v1.5.1.1
        imagePullPolicy: IfNotPresent
        command:
        - /heapster
        - --source=kubernetes:https://kubernetes.default
        - --historical-source=influxdb:http://monitoring-influxdb:
8086
        - --sink=influxdb:http://monitoring-influxdb:8086
```

```
        - --sink=socket:tcp://monitor.csk.##REGION##.aliyuncs.com:
8093?clusterId=##CLUSTER_ID##&public=true
```

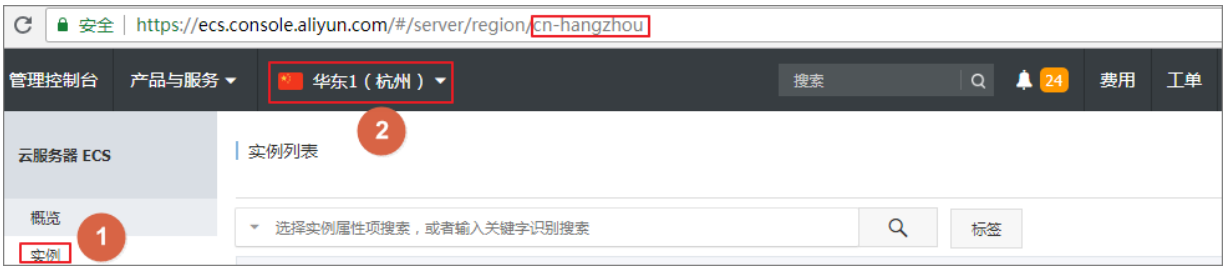
alicloud-monitor-controller 的樣本編排如下，執行`kubectl create -f xxx.yaml`命令部署alicloud-monitor-controller。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: alicloud-monitor-controller
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: alicloud-monitor-controller
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      hostNetwork: true
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      serviceAccount: admin
      containers:
      - name: alicloud-monitor-controller
        image: registry.##REGION##.aliyuncs.com/acs/alicloud-monitor
-controller:v1.0.0
        imagePullPolicy: IfNotPresent
        command:
        - /alicloud-monitor-controller
        - agent
        - --regionId=##REGION##
        - --clusterId=##CLUSTER_ID##
        - --logtostderr
        - --v=4
```

4. 更新完畢後，進入Kubernetes 控制台，在kube-system命名空間中，可看到這兩個Deployment處於運行中，即升級完畢。



對於不清楚自己REGION資訊的開發人員，可以通過如下的方式快速查詢，開啟ECS控制台，選擇自己叢集所在的地區，頁面地址URL中最後一段即是REGION。



1.11.2 使用 Grafana 展示監控資料

前提条件

- 您已經成功部署一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 本樣本使用的 Grafana 的鏡像地址是 `registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4`，內建了相關監控模板。

背景信息

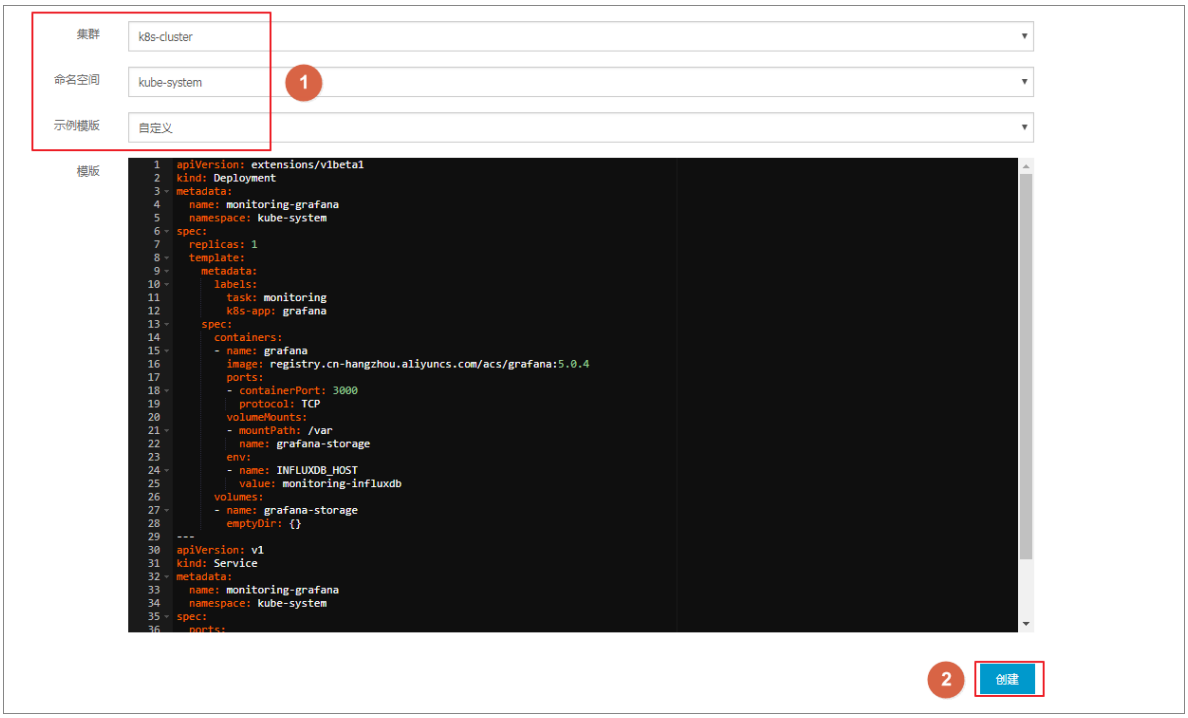
在 kubernetes 的監控方案中，Heapster+Influxdb+Grafana 的組合相比 prometheus 等開源方案而言更為簡單直接。而且 Heapster 在 kubernetes 中承擔的責任遠不止監控資料的採集，控制台的監控介面、HPA的 POD Auto Scaling等都依賴於 Heapster 的功能。因此 Heapster 成為 kubernetes 中一個必不可少的組件，在阿里雲的 Kubernetes 叢集中已經內建了 Heapster+Influxdb 的組合，如果需要將監控的資料進行展示，只需要配置一個可用的 Grafana 與相應的 Dashboard 即可。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 部署，進入部署列表頁面。
3. 單擊頁面右上方的使用模板建立。

4. 對模板進行相關配置，部署 Grafana 的 deployment 和 service，完成配置後，單擊建立。
   - 叢集：選擇所需的叢集。
   - 命名空間：選擇資來源物件所屬的命名空間，必須是**`kube-system`**。

- 樣本模板：本樣本選擇自訂模板，其中包含一個 deployment 和 service。

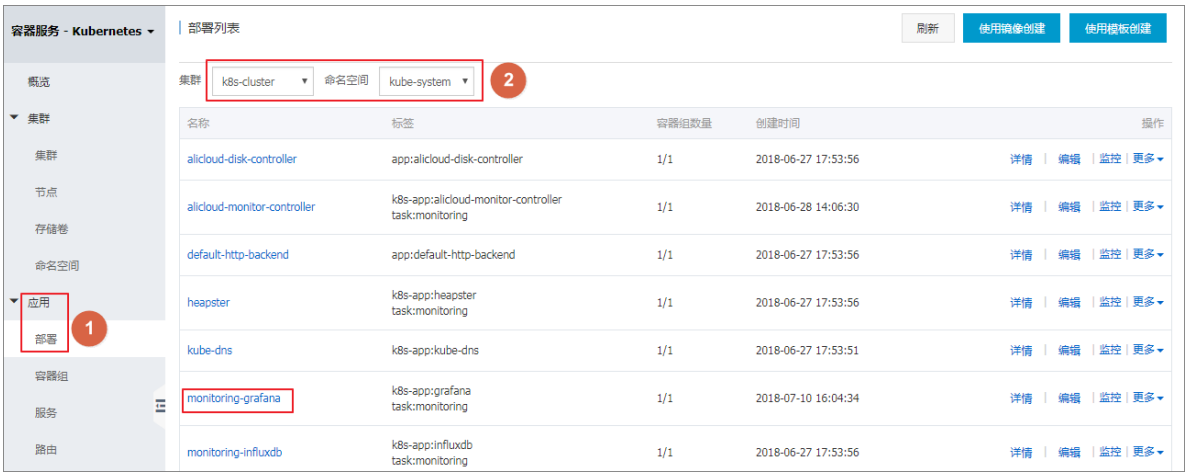


本樣本的編排模板如下。

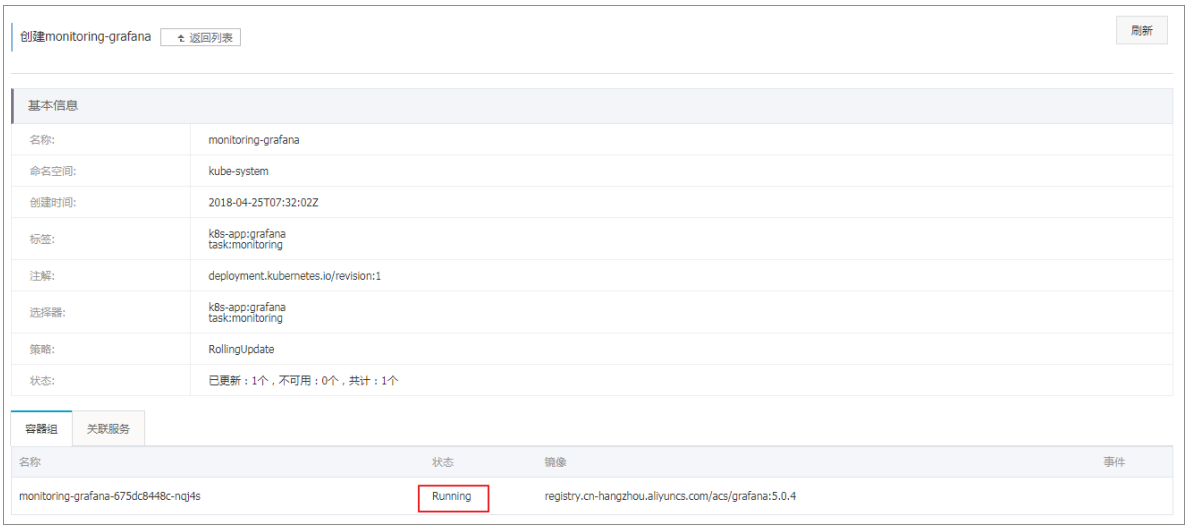
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   replicas: 1
   template:
     metadata:
       labels:
         task: monitoring
         k8s-app: grafana
     spec:
       containers:
       - name: grafana
         image: registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4
         ports:
         - containerPort: 3000
           protocol: TCP
         volumeMounts:
         - mountPath: /var
           name: grafana-storage
         env:
         - name: INFLUXDB_HOST
           value: monitoring-influxdb
       volumes:
       - name: grafana-storage
         emptyDir: {}
 ---
apiVersion: v1
kind: Service
```

```
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   ports:
   - port: 80
     targetPort: 3000
   type: LoadBalancer
   selector:
     k8s-app: grafana
```

5. 完成部署後，返回部署頁面，選擇所需叢集，然後選擇 kube-system，查看其下部署的應用。

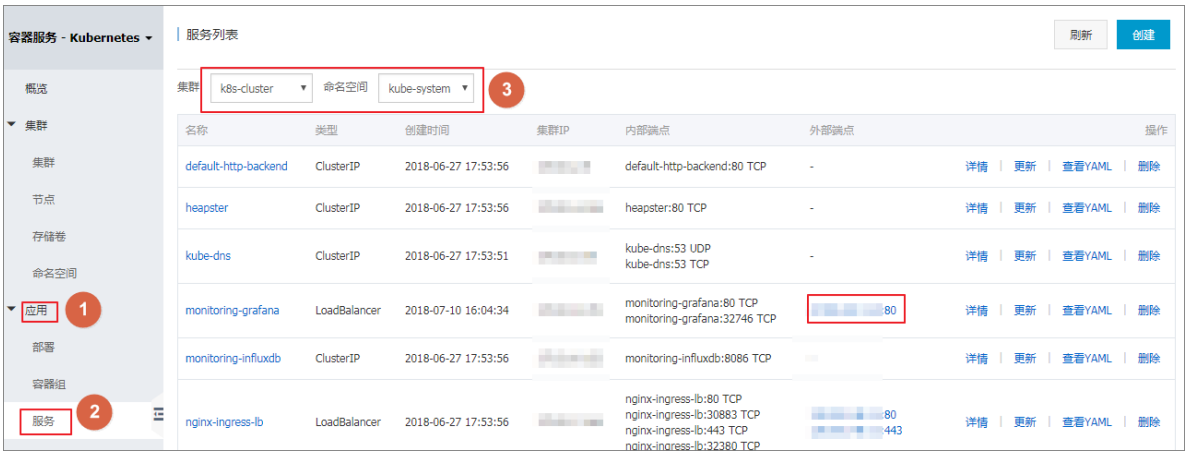


6. 單擊 monitoring-grafana 的名稱，查看部署狀態，等待運行狀態變為 running。



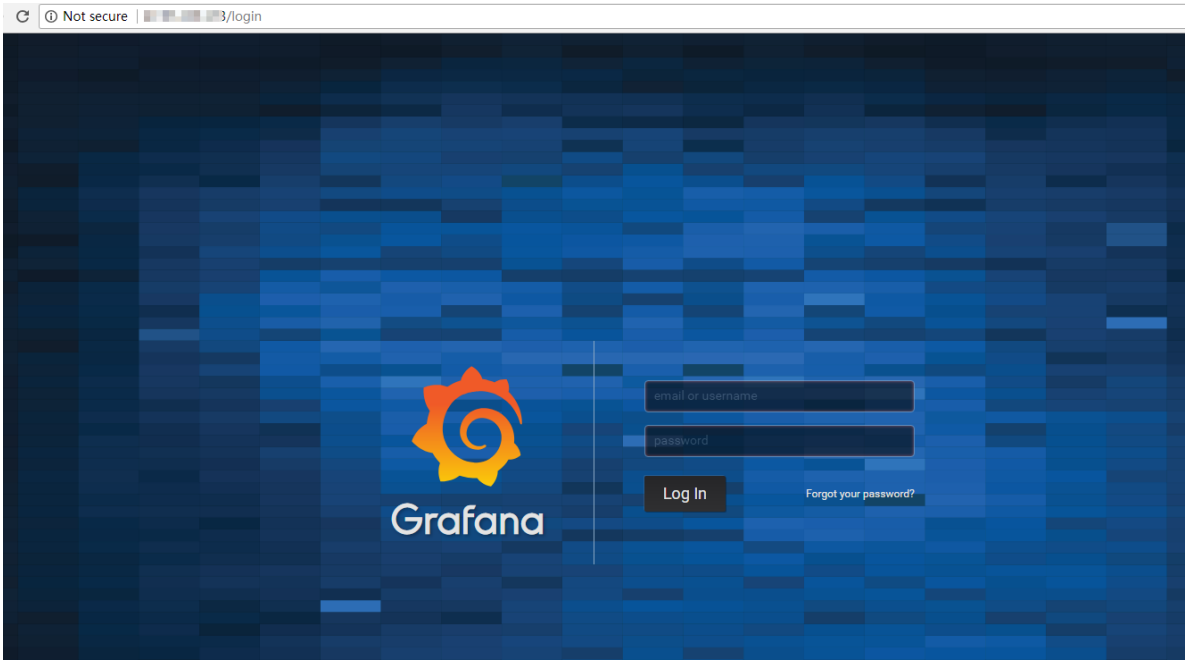
7. 單擊左側導覽列中的應用 > 服務，進入服務列表，選擇所需的叢集和命名空間（kube-system），查看外部端點。

   這個地址是通過 LoadBalancer 類型的 service 自動建立的，對於要求更安全存取原則的開發人員而言，建議考慮添加 IP 白名單或者使用配置認證等方式增強安全性。



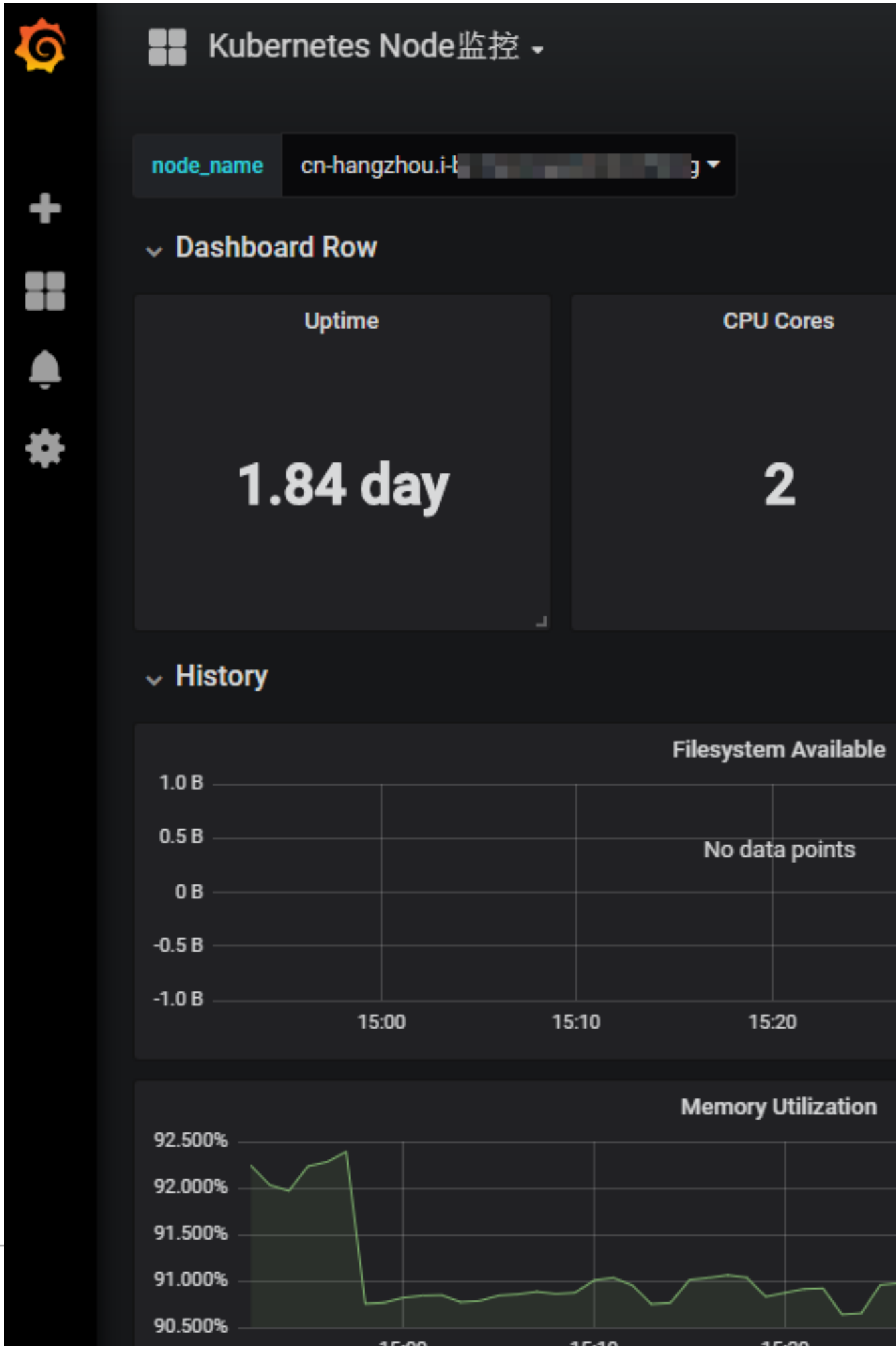
8. 選擇 monitoring-grafana 服務，單擊右側的外部端點，登入 Grafana 監控介面。

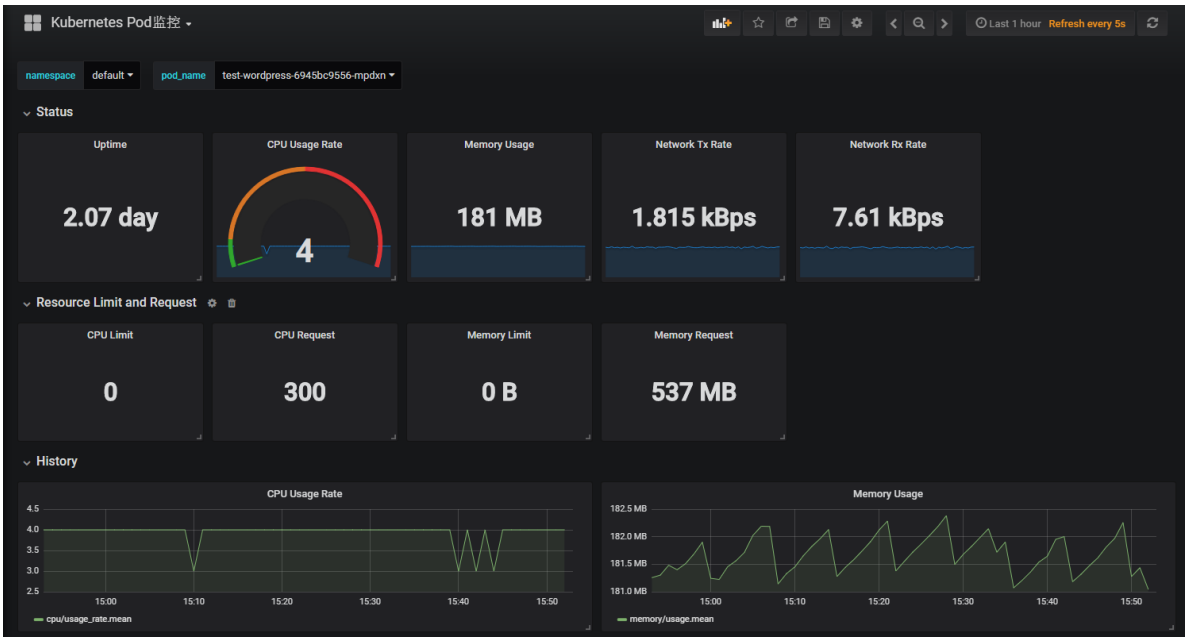
   預設的 Grafana 的使用者名和密碼都是 admin，建議在登入後先修改為更複雜的密碼。



9. 您可選擇內建的監控模板 ，查看 Pod 和 Node 的監控 Dahsboard。

   本樣本使用的 Grafana 版本內建了兩個模板，一個負責展示節點層級的實體資源，一個負責展示 Pod 相關的資源。開發人員也可以通過添加自訂的 Dashboard 的方式進行更複雜的展現，也可以基於 Grafana 進行資源的警示等。


Kubernetes Node监控
node_name
cn-hangzhou.i-b
Dashboard Row
Uptime
1.84 day
CPU Cores
2
History
Filesystem Available
1.0 B
0.5 B
0 B
-0.5 B
-1.0 B
No data points
15:00
15:10
15:20
Memory Utilization
92.500%
92.000%
91.500%
91.000%
90.500%



1.11.3 使用HPAAuto Scaling容器

阿里雲Container Service支援在控制台介面上快速建立支援HPA的應用，實現容器資源的Auto Scaling。您也可通過定義HPA(Horizontal Pod Autoscaling)的yaml配置來進行配置。

前提條件

- 您已成功建立一個Kubernetes叢集，參見*##Kubernetes##*。
- 您已成功串連到Kubernetes叢集的Master節點。

方法1 通過**Container Service**控制台建立**HPA**應用

在阿里雲Container Service中，已經整合了HPA，開發人員可以非常簡單地通過Container Service控制台進行建立。

1. 登入 *Container Service#####*。
2. 在Kubernetes菜單下，單擊左側導覽列中的應用 > 部署，單擊右上方的使用鏡像建立。

3. 填寫應用的名稱，設定應用部署叢集和命名空間，單擊下一步。
4. 首先進行應用設定，設定副本數量，然後勾選開啟自動調整，設定伸縮的條件和配置。

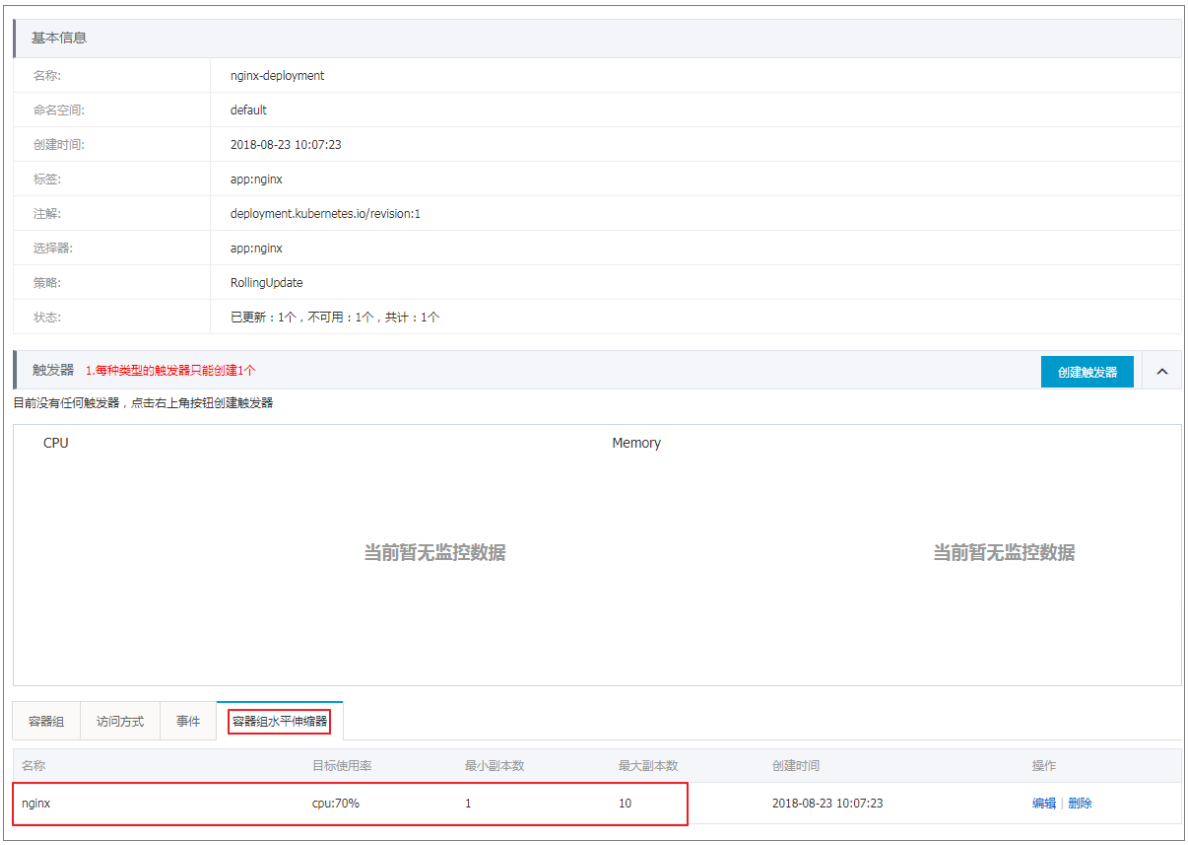
- 指標：支援CPU和記憶體，需要和設定的所需資源類型相同。
- 觸發條件：資源使用率的百分比，超過該使用量，容器開始擴容。
- 最大容器數量：該Deployment可擴容的容器數量上限。
- 最小容器數量：該Deployment可縮容的容器數量下限。

5. 進行容器設定，選擇鏡像，並設定所需的資源。然後單擊下一步

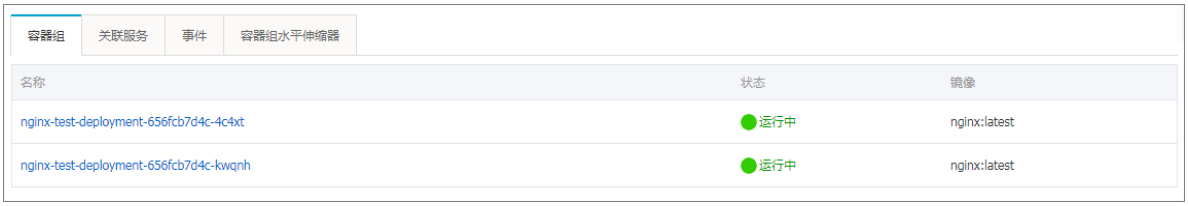
> 说明：
> 您必須為Deployment設定所需資源，否則無法進行容器自動調整。

6. 進入訪問設定頁面，本例中不進行訪問設定，單擊建立。

   此時一個支援HPA的Deployment就已經建立完畢，您可在部署的詳情中查看伸縮組資訊。



7. 在實際使用環境中，應用會根據CPU負載進行伸縮。您也可在測試環境中驗證Auto Scaling，通過給Pod進行CPU壓測，可以發現Pod在半分鐘內即可完成水平的擴充。



方法2 通過**kubectl**命令進行使用

您也可通過編排模板來手動建立HPA，並將其綁定到要伸縮的Deployment對象上，通過**`kubectl`**命令實現容器自動調整配置。

下面針對一個Nginx應用進行舉例，Deployment的編排模板如下，執行**`kubectl create -f xxx.yml`**命令進行建立。

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
```

```
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:
tags>
        ports:
        - containerPort: 80
        resources:
          requests:                          ##必須設定，不然HPA無法運行
            cpu: 500m
```

然後建立HPA，通過**scaleTargetRef**設定當前HPA綁定的對象，在本例中綁定是名叫nginx的Deployment。

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
  namespace: default
spec:
  scaleTargetRef:                              ##綁定名為nginx的
Deployment
    apiVersion: apps/v1beta2
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
```

說明：

HPA需要給Pod設定request資源，如果沒有request資源，HPA不會運行。

執行**kubectl describe hpa [name]**會發現有類似如下的warnning。

```
Warning  FailedGetResourceMetric       2m (x6 over 4m)  horizontal-pod
-autoscaler  missing request for cpu on container nginx in pod default
/nginx-deployment-basic-75675f5897-mqzs7
```

```
Warning  FailedComputeMetricsReplicas  2m (x6 over 4m)  horizontal-pod
-autoscaler  failed to get cpu utilization: missing request for cpu on
 container nginx in pod default/nginx-deployment-basic-75675f5
```

建立好HPA後，再次執行**kubectl describe hpa [name]**命令，可以看到如下資訊，則表示HPA已經正常運行。

```
Normal SuccessfulRescale 39s horizontal-pod-autoscaler New size: 1;
reason: All metrics below target
```

此時當Nginx的Pod的利用率超過本例中設定的50%利用率時，則會進行水平擴容，低於50%的時候會進行縮容。

1.11.4 通過資源分組進行監控與警示

前提条件

- 如果之前沒有建立過叢集，您需要*##Kubernetes##*。
- Kubernetes 版本需要在 1.8.4 及以上，若叢集版本過低，您可以先對叢集進行升級，然後通過升級監控服務的方式快速建立資源警示分組。

背景信息

在 IT 系統基礎設施營運中，監控警示一直是保證可靠性和安全性的基礎，有助於日常營運、系統監測、故障排除和調試。

在 Kubernetes 情境下，傳統的容器監控方案通過靜態配置化的監控 agent 或中心化的 server 進行資源監控和警示，會遇到很大的難題。例如，由於容器更多的是在資源集區中調度，宿主機部署監控 agent 會造成缺乏必要資訊來識別監控對象；容器的生命週期與傳統應用相比而言會更加短暫，而由容器抽象的上層概念如 kubernetes 中的 ReplicaSet、Deployment 等則沒有太好的辦法從採集的資料中進行反向的抽象，造成單純的容器監控資料無法有效地進行監控資料的彙總和警示，一旦應用的發布可能會導致原有的監控與警示規則無法生效。

阿里雲Container Service Kubernetes 與Cloud Monitor進行了深度整合，用應用分組來抽象邏輯概念，實現邏輯的概念和物理概念在監控資料、生命週期上面的統一。此外，阿里雲Cloud Monitor服務提供豐富的功能特性和自訂工具，協助您快速實現 Kuberbetes 資源監控和警示的最佳實務。

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的叢集，在叢集列表下找到目的地組群。
3. 查看 kubernetes 版本和叢集 ID 等資訊，單擊叢集右側的更多 > 升級監控服務 ，快速建立資源警示分組。

> 说明：
> 若您的叢集版本低於 1.8.4，請先單擊叢集升級進行升級。

4. 登入 *Cloud Monitor#####*。在左側導覽列中單擊應用分組，在應用分組中，可以看到包含叢集 ID 資訊的 kubernetes 資源分組。

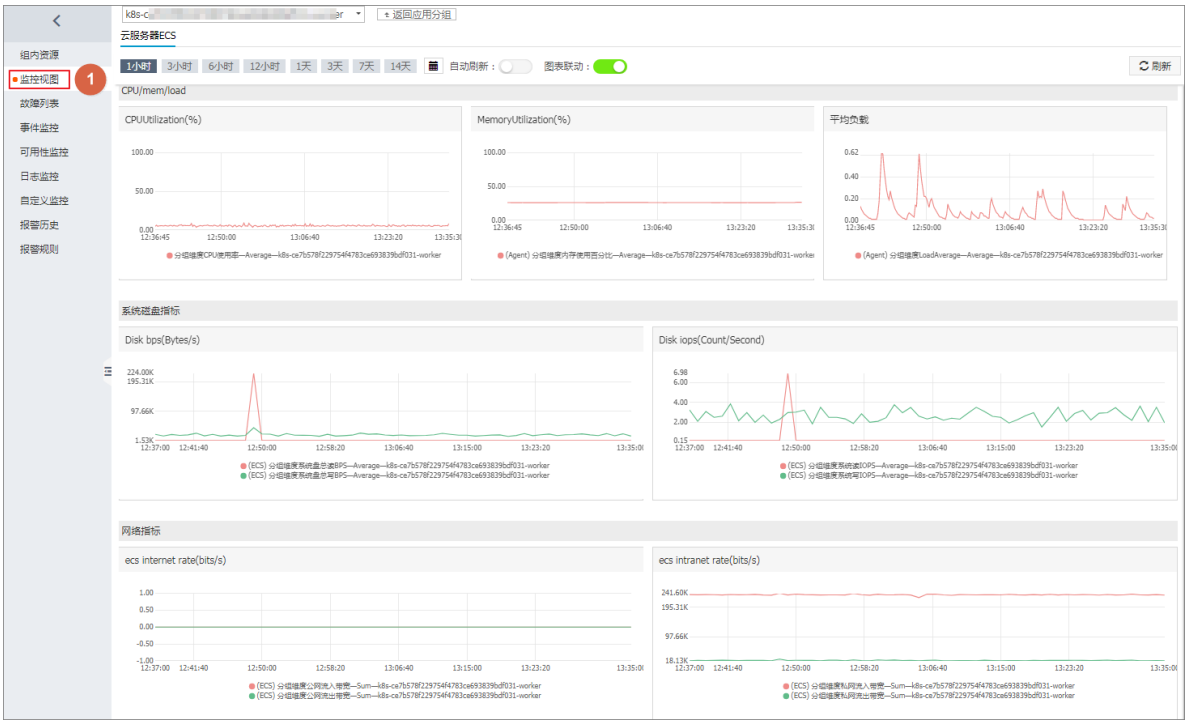
5. 單擊分組名稱，進入具體的分組頁面，您可以查看組內的各項監控視圖。以 kubernetes 的 master 分組為例，預設顯示分組的拓撲視圖。

   Kubernetes 節點從職能上分為 Worker 和 Master 兩種不同的節點。Master 節點上面通常會部署管控類型的應用，整體的資源要求以強魯棒性為主；而 Worker 節點更多的承擔實際的 Pod 調度，整體的資源以調度能力為主。當你建立資源警示分組時，Container Service會為你自動建立兩個資源分組，一個是 Master 組，一個是 Worker 組。Master 組中包含了 Master 節點以及與其相關的負載平衡器；Worker 組包含了所有的工作節點。

6. 您若想瞭解分組內詳細的機器資訊，請單擊列表視圖的表徵圖，您可以查看分組內的各雲產品的詳細資料。

7. 單擊右下角的添加監控圖表，您可以添加更多的監控圖表。

8. 在左側導覽列中單擊監控視圖，您可以查看分組內各雲產品的詳細監控指標。



9. 在左側導覽列中單擊警示規則，該頁面列出當前分組中已有的警示規則列表，預設會在 Master 分組中設定所有節點的核心組件的健全狀態檢查。

   1. 您可以單擊建立警示規則，根據業務需求，來建立屬於該分組的警示規則。

   2. 進入警示規則頁面，您需要設定警示規則。

      - 您可選擇警示關聯對象。
      - 您可選擇是否使用模板建立警示規則。若選擇使用模板建立警示規則，您可以在選擇模板下拉框中選擇已有的警示模板；或者可以單擊建立警示模板，建立新的自訂警示模板，參見 ，然後再進行選擇。
      - 您可設定通知方式，如通過釘釘、郵件、SMS的方式在第一時間擷取到 Kubernetes 的叢集狀態。

3. 最後單擊確認，本例中建立的警示規則會出現在規則列表下。

后续操作

在左側導覽列中，您可以探索更多符合自己資源監控需求的功能特性，如故障列表、事件監控、可用性監控、日誌監控等。

1.12 安全管理

1.12.1 概述

授權管理

Kubernetes 叢集支援叢集層級的操作的子帳號授權。

詳細操作參見#####。

全鏈路 **TLS** 認證

在Container Service提供的 Kubernetes 叢集中，存在的以下通訊鏈路，均會進行 TLS 認證校正，以保證通訊不被竊聽或篡改。

- 位於 Worker 節點上的 `kubelet` 主動串連位於 Master 節點上的 `apiserver` 時
- 位於 Master 節點上的 `apiserver` 主動串連位於 Worker 節點上的 `kubelet` 時

在初始化過程中，發起初始化的 Master 節點會通過 SSH 隧道的方式串連到其他節點的 SSH 服務（22 連接埠）進行初始化。

原生 **Secret&RBAC** 支援

Kubernetes Secret 用於儲存密碼、OAuth Token、SSH Key 等敏感資訊。明文地將這些敏感資訊寫在 Pod YAML 檔案中或者寫在 Dockerfile 固化到鏡像中，會有資訊泄露的可能，使用 Secret 能有效地避免這些安全隱患。

詳細資料參見*Secret*。

Role-Based Access Control (RBAC) 使用 Kubernetes 內建的 API 組來驅動授權鑒權管理，您可以通過 API 來管理不同的 Pod 對應到不同的角色，以及各自的角色擁有的存取權限。

詳細資料參見*Using RBAC Authorization*。

網路隔離

Kubernetes 叢集中不同節點間 Pod 預設是可以互相訪問的。在部分情境中，不同業務之間不應該網路互連，為了減少風險，您需要引入網路隔離（Network Policy）。在 Kubernetes 叢集中，您可以使用 Canal 網路驅動實現網路隔離支援。

鏡像安全掃描

Kubernetes 叢集可使用Container Registry進行鏡像管理，鏡像服務支援鏡像安全掃描。

鏡像安全掃描可以快速識別鏡像中存在的安全風險，減少 Kubernetes 叢集上啟動並執行應用被攻擊的可能性。

詳細描述參見 ######。

安全性群組與公網訪問

每個建立的叢集會被預設分配一個新的、安全風險最小化的安全性群組。該安全性群組對於公網入方向僅允許 ICMP。

建立叢集預設不允許公網 SSH 連入，您可以參考*SSH##Kubernetes##* 配置通過公網 SSH 連入到叢集節點中。

叢集節點通過 NAT Gateway訪問公網，可進一步減少安全風險。

1.12.2 Kube-apiserver審計日誌

在Kubernetes叢集中，apiserver的審計日誌可以協助叢集管理員記錄或追溯不同使用者的日常操作，是叢集安全營運中重要的環節。本文旨在協助您瞭解阿里雲Kubernetes叢集apiserver審計日誌的相關配置，以及如何通過SLSLog Service收集和搜尋指定的日誌內容。

配置介紹

當前建立Kubernetes叢集會預設開啟apiserver審計功能，相關的參數配置功能如下：

说明：

登入到Master節點，apiserver設定檔的目錄是*/etc/kubernetes/manifests/kube-apiserver.yaml*。

配置	說明
audit-log-maxbackup	審計日誌最大分區儲存10個記錄檔
audit-log-maxsize	單個審計日誌最大size為100MB
audit-log-path	審計日誌輸出路徑為*/var/log/kubernetes/kubernetes.audit*
audit-log-maxage	審計日誌最多儲存期為7天
audit-policy-file	審計日誌配置策略檔案，檔案路徑為：*/etc/kubernetes/audit-policy.yml*

登入Master節點機器，審計配置策略檔案的目錄是*/etc/kubernetes/audit-policy.yml*，內容如下：

```
apiVersion: audit.k8s.io/v1beta1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived
stage.
omitStages:
  - "RequestReceived"
rules:
  # The following requests were manually identified as high-volume and
 low-risk,
  # so drop them.
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: "" # core
```

```
        resources: ["endpoints", "services"]
  - level: None
    users: ["system:unsecured"]
    namespaces: ["kube-system"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["configmaps"]
  - level: None
    users: ["kubelet"] # legacy kubelet identity
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes"]
  - level: None
    userGroups: ["system:nodes"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes"]
  - level: None
    users:
      - system:kube-controller-manager
      - system:kube-scheduler
      - system:serviceaccount:kube-system:endpoint-controller
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: "" # core
        resources: ["endpoints"]
  - level: None
    users: ["system:apiserver"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["namespaces"]
  # Don't log these read-only URLs.
  - level: None
    nonResourceURLs:
      - /healthz*
      - /version
      - /swagger*
  # Don't log events requests.
  - level: None
    resources:
      - group: "" # core
        resources: ["events"]
  # Secrets, ConfigMaps, and TokenReviews can contain sensitive &
binary data,
  # so only log at the Metadata level.
  - level: Metadata
    resources:
      - group: "" # core
        resources: ["secrets", "configmaps"]
      - group: authentication.k8s.io
        resources: ["tokenreviews"]
  # Get repsonses can be large; skip them.
  - level: Request
    verbs: ["get", "list", "watch"]
    resources:
      - group: "" # core
      - group: "admissionregistration.k8s.io"
```

```
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
      - group: "policy"
      - group: "rbac.authorization.k8s.io"
      - group: "settings.k8s.io"
      - group: "storage.k8s.io"
  # Default level for known APIs
  - level: RequestResponse
    resources:
      - group: "" # core
      - group: "admissionregistration.k8s.io"
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
      - group: "policy"
      - group: "rbac.authorization.k8s.io"
      - group: "settings.k8s.io"
      - group: "storage.k8s.io"
  # Default level for all other requests.
  - level: Metadatak8s.io"
      - group: "storage.k8s.io"
  # Default level for all other requests.
  - level: Metadata
```

 说明：

- 在收到請求後不立即記錄日誌，當返回體header發送後才開始記錄。
- 對於大量冗餘的kube-proxy watch請求，kubelet和system:nodes對於node的get請求，kube組件在kube-system下對於endpoint的操作，以及apiserver對於namespaces的get請求等不作審計。
- 對於*/healthz**，*/version**, */swagger**等唯讀url不作審計。
- 對於可能包含敏感資訊或二進位檔案的secrets, configmaps, tokenreviews介面的日誌等級設為metadata，該level只記錄請求事件的使用者、時間戳記、請求資源和動作，而不包含請求體和返回體。
- 對於一些如authenticatioin、rbac、certificates、autoscaling、storage等敏感介面，根據讀寫記錄相應的請求體和返回體。

採集和檢索

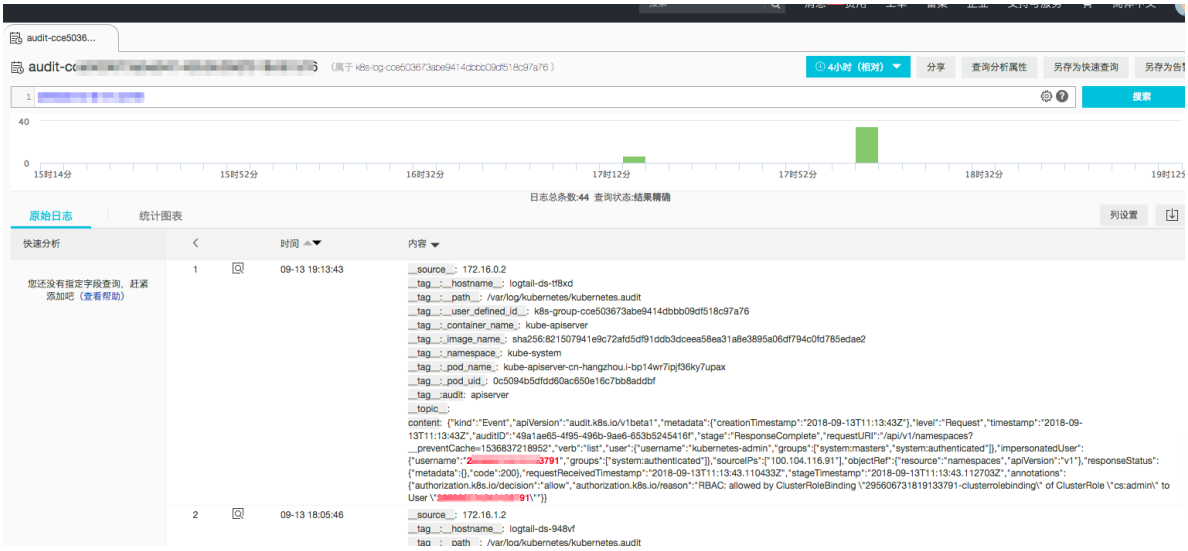
在使用Kube-apiserver審計日誌之前，請確保在建立叢集的配置中開啟了SLSLog Service，並成功建立對應的日誌Project和Logstore。

1. 登入 *Log Service#####*。
2. 單擊左側導覽列中**Project**管理，選擇建立叢集時設定的日誌Project，單擊名稱進入日誌Project頁面。

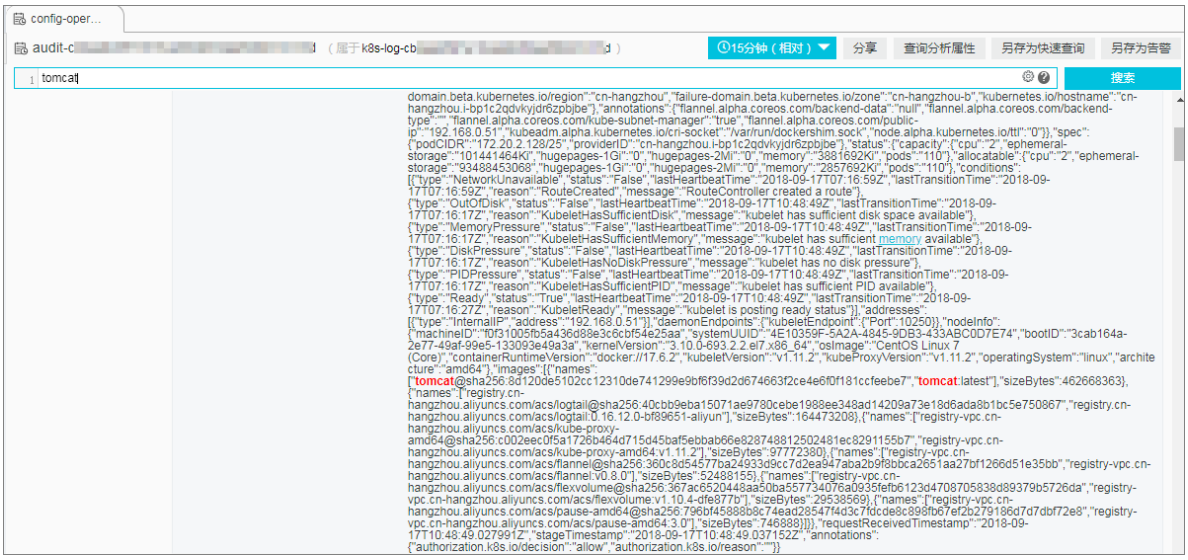
3. 在Project詳情頁中，預設進入日誌庫頁面，查看名為audit-${clustered}的日誌庫（logstore），單擊右側的查詢，叢集對應的審計日誌會收集在該日誌庫中。

> 说明：
> 在您建立過程中，指定的日誌Project中會自動添加一個名稱為audit-${clustered}的日誌庫。

4. 當叢集管理員需要關注某一子帳號的行為時，可以輸入相應子帳號id，追溯其操作。



5. 當叢集管理員關注某一具體資來源物件時，可以輸入相應的資源名稱，檢索時間段內的指定操作。



支援第三方日誌解決

您可以在叢集Master各節點，在`/var/log/kubernetes/kubernetes.audit` 路徑下找到審計日誌的源檔案。該檔案是標準的json格式，您可以在部署叢集時選擇不使用阿里雲的Log Service，根據需要對接其他的日誌解決方案，完成相關審計日誌的採集和檢索。

1.13 發行管理

1.13.1 基於Heml的發行管理

在阿里雲 Kubernetes 服務提供了豐富的雲市場，其中的應用目錄和服務類別目錄功能整合了 helm 包管理工具，協助您快速構建雲上應用，因為 chart 可以多次發布（release），這就帶來一個發布版本管理的問題。因此，阿里雲 Kubernetes 服務提供了發布功能，通過 Web 介面的方式對通過 helm 發布的應用進行管理。

前提條件

- 您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已經使用應用目錄或服務類別目錄功能，安裝了 helm 應用，參見*## Helm ######*。本例中是一個 tf-model 應用。

查看發布的詳情

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 發布，選擇所需的叢集，進入發布列表頁面。

   您可看到該叢集下通過 helm 包管理工具發布的應用及服務等。

3. 以 tf-model 為例，您可查看發布的詳情資訊，單擊右側的詳情，進入該發布的詳情頁面。

   你可以查看該發布的目前的版本和曆史版本等資訊，目前的版本為1，無曆史版本。您還可查看 tf-model 的資源資訊，如資源名稱，資源類型，以及查看 YMAL 資訊。

   
   说明：
   單擊資源名稱，可進入 Kubernetes Dashboard 頁面，查看對應資源的詳細運行狀態。

4. 單擊參數，你可查看該 helm 包安裝的參數配置。

更新發布的版本

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 發布，選擇所需的叢集，進入發布列表頁面。

   您可看到該叢集下通過 helm 包管理工具發布的應用及服務等。

3. 以 tf-model 為例，您可更新該發布，單擊右側的更新，彈出更新發布對話方塊。

4. 在對話方塊中修改相關參數，隨後單擊更新，可對該發布進行更新。

   在發布列表頁面，您可以看到目前的版本變為 2，您可以在曆史版本菜單下找到版本1，單擊 復原到該版本，可進行復原。

刪除發布

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的應用 > 發布，選擇所需的叢集，進入發布列表頁面。

   您可看到該叢集下通過 helm 包管理工具發布的應用及服務等。

3. 以 tf-model 為例，您可刪除該發布，單擊右側的刪除，彈出刪除對話方塊。

4. 勾選是否清除發布記錄，然後單擊確定，您可以刪除 tf-model 應用，其包含的 service、deployment 等資源都會一併刪除。

1.14 Istio管理

1.14.1 概述

Istio是一個提供串連、保護、控制以及觀測微服務功能的開放平台。

微服務目前被越來越多的IT企業重視。微服務是將複雜的應用切分為若干服務，每個服務均可以獨立開發、部署和伸縮；微服務和容器組合使用，可進一步簡化微服務的交付，提升應用的可靠性和延展性。

隨著微服務的大量應用，其構成的分布式應用架構在營運、調試、和安全管理等維度變得更加複雜，開發人員需要面臨更大的挑戰，如：服務發現、負載平衡、故障恢複、指標收集和監控，以及A/B測試、灰階發布、藍綠髮布、限流、存取控制、端到端認證等。

Istio應運而生。Istio是一個提供串連、保護、控制以及觀測微服務功能的開放平台，提供了一種簡單的建立微服務網路的方式，並提供負載平衡、服務間認證以及監控等能力，同時Istio不需要修改服務即可實現以上功能。

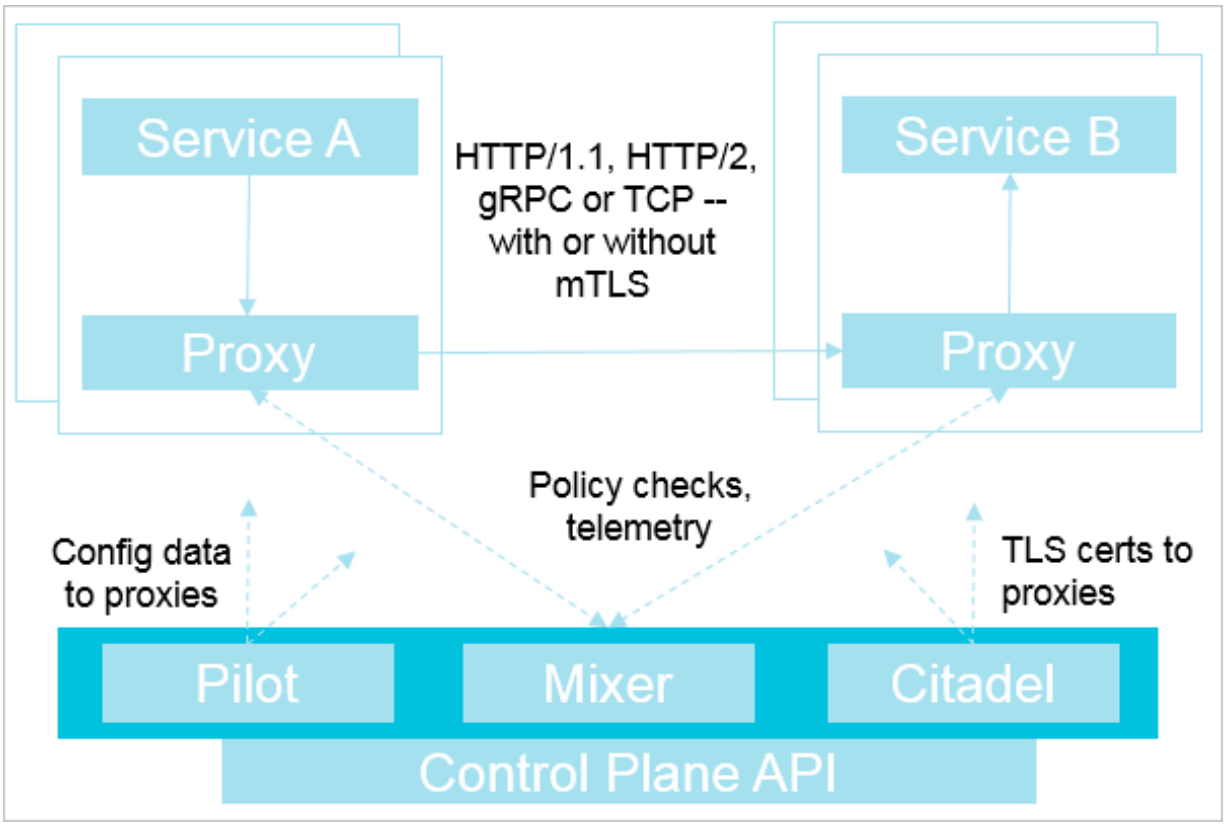
Istio提供如下功能：

- 流量管理：控制服務之間的流量及API調用。增強系統的可靠性。
- 鑒權及安全保護：為網格中的服務提供身分識別驗證，並保護服務的流量。增強系統的安全性。
- 策略執行：控制服務之間的存取原則，且不需要改動服務。
- 可觀察性： 擷取服務之間的流量分布及調用關係。 快速定位問題。

Istio架構

Istio在邏輯上分為控制層面和資料層面：

- 控制層面：管理代理程式（預設為Envoy），用於管理流量路由、運行時策略執行等。
- 資料層面：由一系列代理（預設為Envoy）組成，用於管理和控制服務之間的網路通訊。



Istio主要由以下組件構成：

- Istio管理器（Pilot）：負責收集和驗證配置，並將其傳播到各種Istio組件。它從策略執行模組（Mixer）和智能代理（Envoy）中抽取環境特定的實現細節，為他們提供使用者服務的抽象表示，獨立於底層平台。此外，流量管理規則（即通用4層規則和7層HTTP/gRPC路由規則）可以在運行時通過Pilot進行編程。
- 策略執行模組（Mixer）：負責在服務網格上執行存取控制和使用原則，並從智能代理（Envoy）和其他服務收集遙測資料。依據智能代理（Envoy）提供的屬性執行策略。
- Istio安全模組：提供服務間以及使用者間的認證，確保在不需要修改服務代碼的前提下，增強服務之間的安全性。包括3個組件：
  - 身份識別：當Istio運行在Kubernetes時，根據容器Kubernetes提供的服務帳號，識別運行服務的主體。
  - Key管理：提供CA自動化產生和管理key和認證。
  - 通訊安全：通過智能代理（Envoy）在用戶端和服務端提供通道（tunnel）保證服務的安全性。
- 智能代理（Envoy）：作為一個獨立的組件與相關微服務部署在同一個Kubernetes的pod上，並提供一系列的屬性給策略執行模組（Mixer）。策略執行模組（Mixer）以此作為執行策略的依據，並發送到監控系統。

1.14.2 部署Istio

為解決微服務的分布式應用架構在營運、調試、和安全管理等維度存在的問題，可通過部署Istio建立微服務網路，並提供負載平衡、服務間認證以及監控等能力，同時Istio不需要修改服務即可實現以上功能。

前提條件

您已經成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。

背景資訊

- 阿里雲Container ServiceKubernetes 1.10.4及之後版本支援部署Istio，如果是1.10.4之前的版本，請先升級到1.10.4或之後版本。
- 一個叢集中，worker節點數量需要大於等於3個，保證資源充足可用。

操作步驟

通過叢集介面部署**Istio**

1. 登入 *Container Service#####*。
2. 單擊左側導覽列中的叢集，進入叢集列表頁面。
3. 選擇所需的叢集並單擊操作列更多 > 部署**Istio**。

4. 根據如下資訊，部署Istio：

配置	說明
叢集	部署Istio的目的地組群。
命名空間	部署Istio的命名空間。
發布名稱	發布的Istio名稱。
啟用 Prometheus 度量日誌收集	是否啟用Prometheus 收集度量日誌。預設情況下啟用。
啟用 Grafana 度量展示	是否啟用Grafana 展示指標的度量資料。預設情況下啟用。

配置	說明
啟用 ServiceGraph 可視化部署	是否啟用ServiceGraph 進行可視化部署。預設情況下啟用。
啟用 Sidecar 自動注入	是否啟用 Sidecar 進行容器的自動注入。預設情況下啟用。
啟用阿里雲Log Service SLS 及 Jaeger	是否啟用阿里雲的Log ServiceSLS 及 Jaeger。預設情況下不啟用。 服務入口地址（Endpoint）：根據配置的Log Service所在region選擇對應的地址。具體請參見#### 項目名稱（Project）：採集日誌所在的項目名稱。 日誌庫名稱（Logstore）：採集日誌所在的日誌庫名稱。 AccessKeyID：訪問Log Service時，所使用的訪問秘鑰ID。 说明： 需要選擇有許可權訪問Log Service的AccessKeyID。 AccessKeySecret：訪問Log Service時，所使用的訪問秘鑰Secret。

5. 單擊部署 **Istio**，啟動部署。

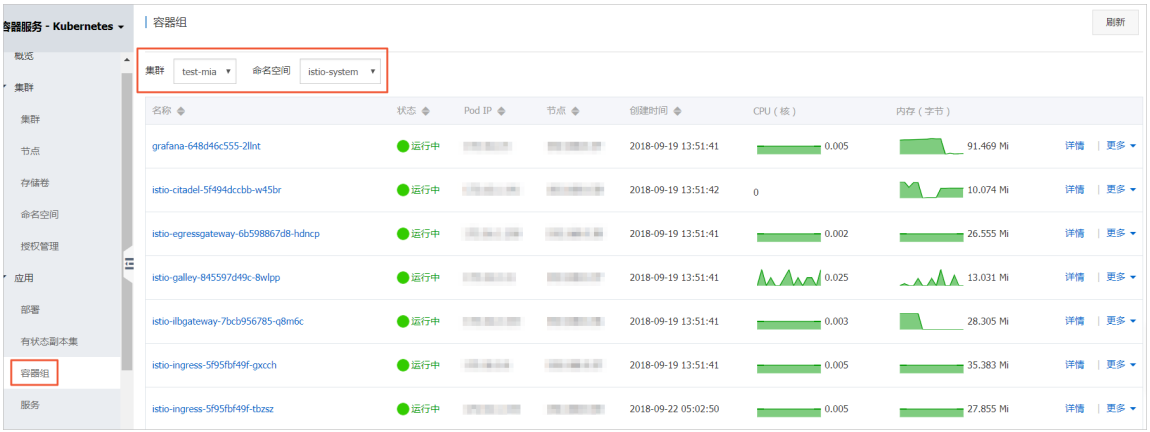
   在部署頁面下方，可即時查看部署進展及狀態。

預期結果

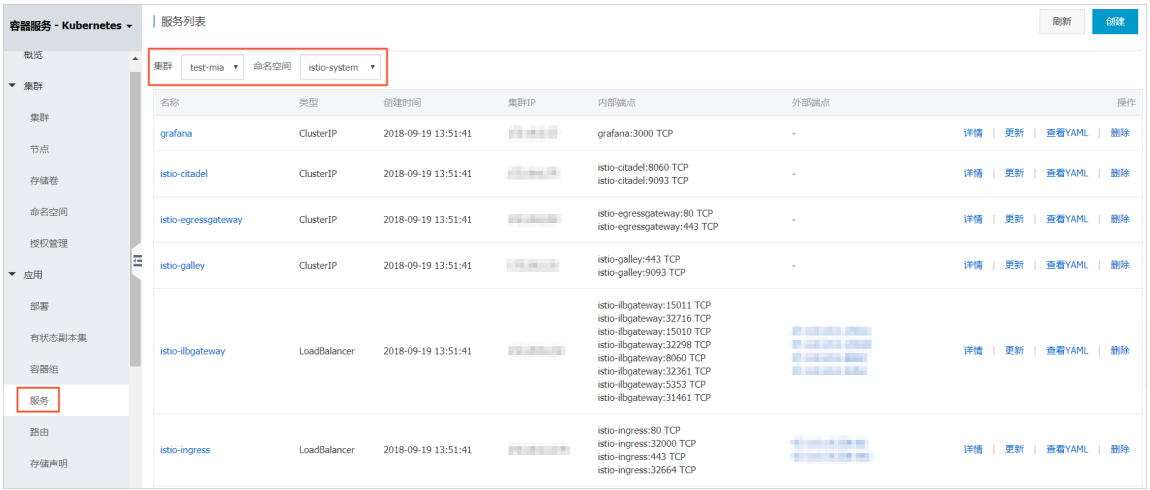
可通過以下方法查看部署是否成功：

- 在部署 **Istio**頁面下方，部署 **Istio**變為已部署。

- • 單擊左側導覽列應用 > 容器組，進入容器組頁面。
  - 選擇部署Istio的叢集及命名空間，可查看到已經部署Istio的相關容器組。

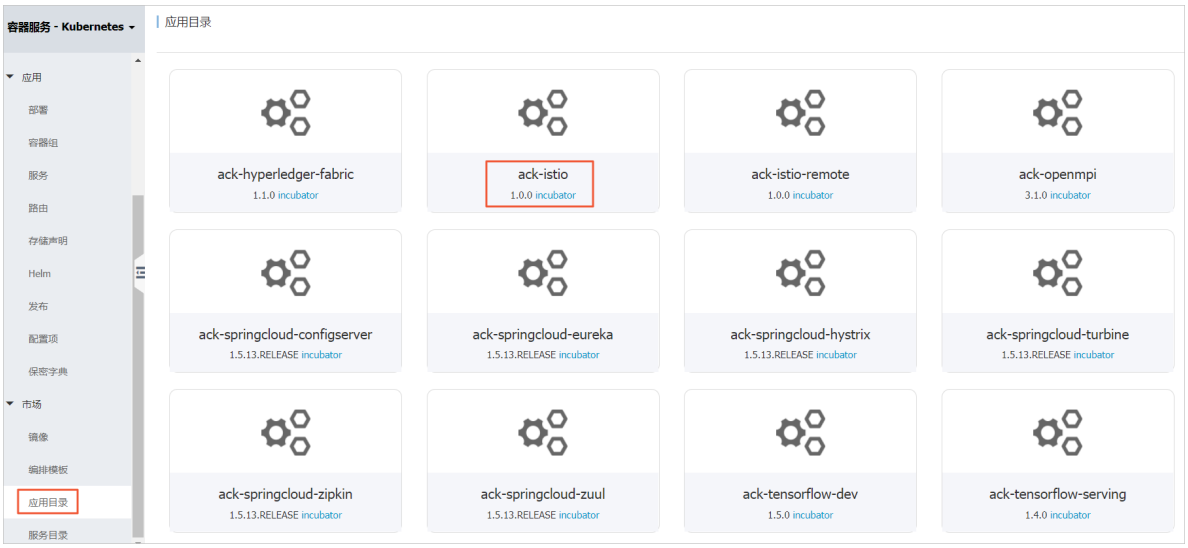


- • 單擊左側導覽列應用 > 服務，進入服務列表頁面。
  - 選擇部署Istio的叢集及命名空間，可查看到已經部署Istio相關服務所提供的訪問地址。

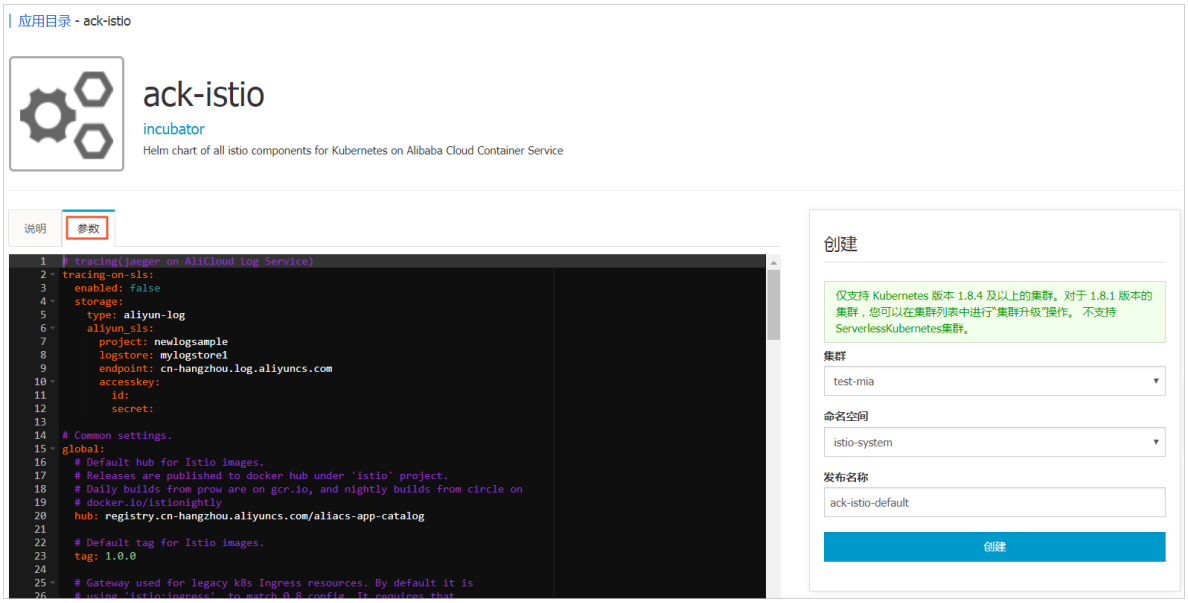


通過應用目錄部署Istio

1. 登入 *Container Service#####*。
2. 單擊左側導覽列中的市場 > 應用目錄，進入應用目錄頁面。



3. **單擊ack-istio，進入應用目錄 - ack-istio頁面。**
4. **單擊參數頁簽，進行參數配置。**



说明：

- 通用參數的含義、取值及預設情況，可參考說明頁簽**Configuration**欄位。
- 也可以針對特定參數進行定製化配置，例如：是否啟用**grafana**、**prometheus**、**tracing**、**weave-scope**以及**kiali**等，可參考：

```
#
# addons configuration
#
grafana:
enabled: true
replicaCount: 1
image: istio-grafana
service:
```

```
name: http
type: ClusterIP
externalPort: 3000
internalPort: 3000
....
prometheus:
enabled: true
replicaCount: 1
image:
repository: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog/
istio-prometheus
tag: latest
....
tracing:
enabled: true
jaeger:
enabled: true
....
weave-scope:
enabled: true
global:
# global.image: the image that will be used for this release
image:
repository: weaveworks/scope
tag: "1.9.0"
# global.image.pullPolicy: must be Always, IfNotPresent, or Never
pullPolicy: "IfNotPresent"
....
kiali:
enabled: true
replicaCount: 1
image:
repository: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog/
istio-kiali
tag: dev
```

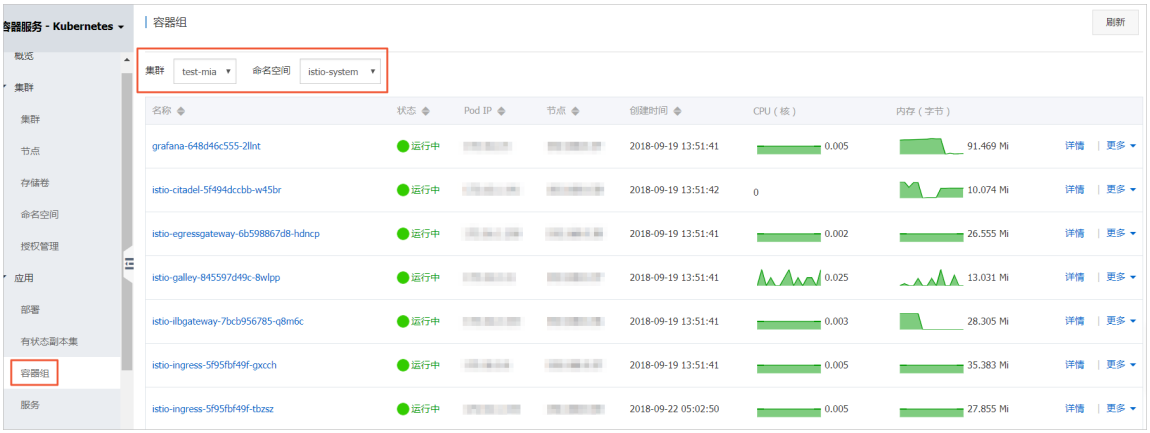
5. 在右側建立地區，填寫以下基本資料：

配置	說明
叢集	部署Istio的目的地組群。
命名空間	部署Istio的命名空間。預設情況下為 default。
發布名稱	發布的Istio名稱。

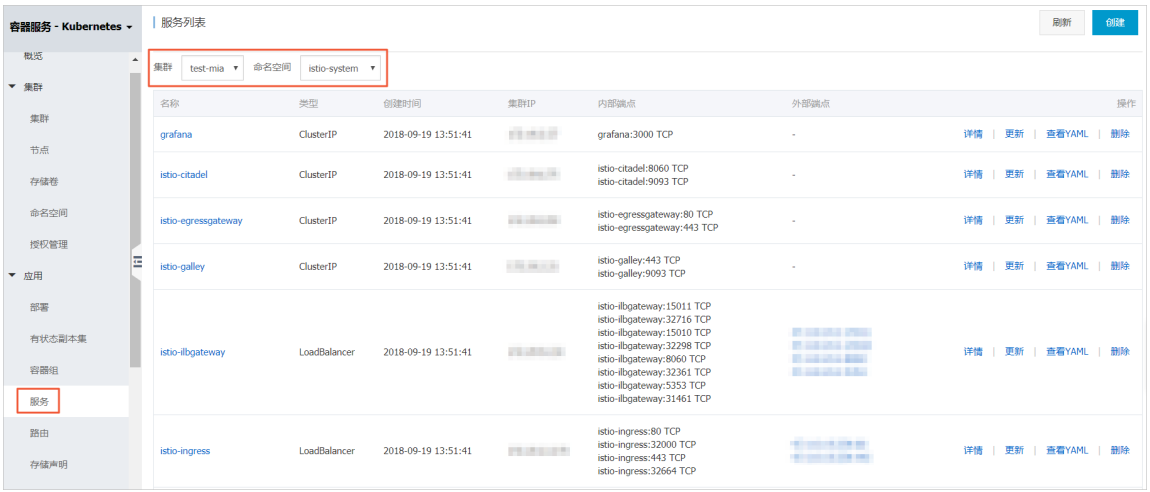
6. 單擊建立，啟動部署。

預期結果

- • 單擊左側導覽列應用 > 容器組，進入容器組頁面。
  - 選擇部署Istio的叢集及命名空間，可查看到已經部署Istio的相關容器組。



- • 單擊左側導覽列應用 > 服務，進入服務列表頁面。
  - 選擇部署Istio的叢集及命名空間，可查看到已經部署Istio相關服務所提供的訪問地址。



1.14.3 更新Istio

您可以通過更新操作，編輯已部署的Istio。

前提条件

- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已成功建立一個**Istio**，參見*##Istio*。

操作步驟

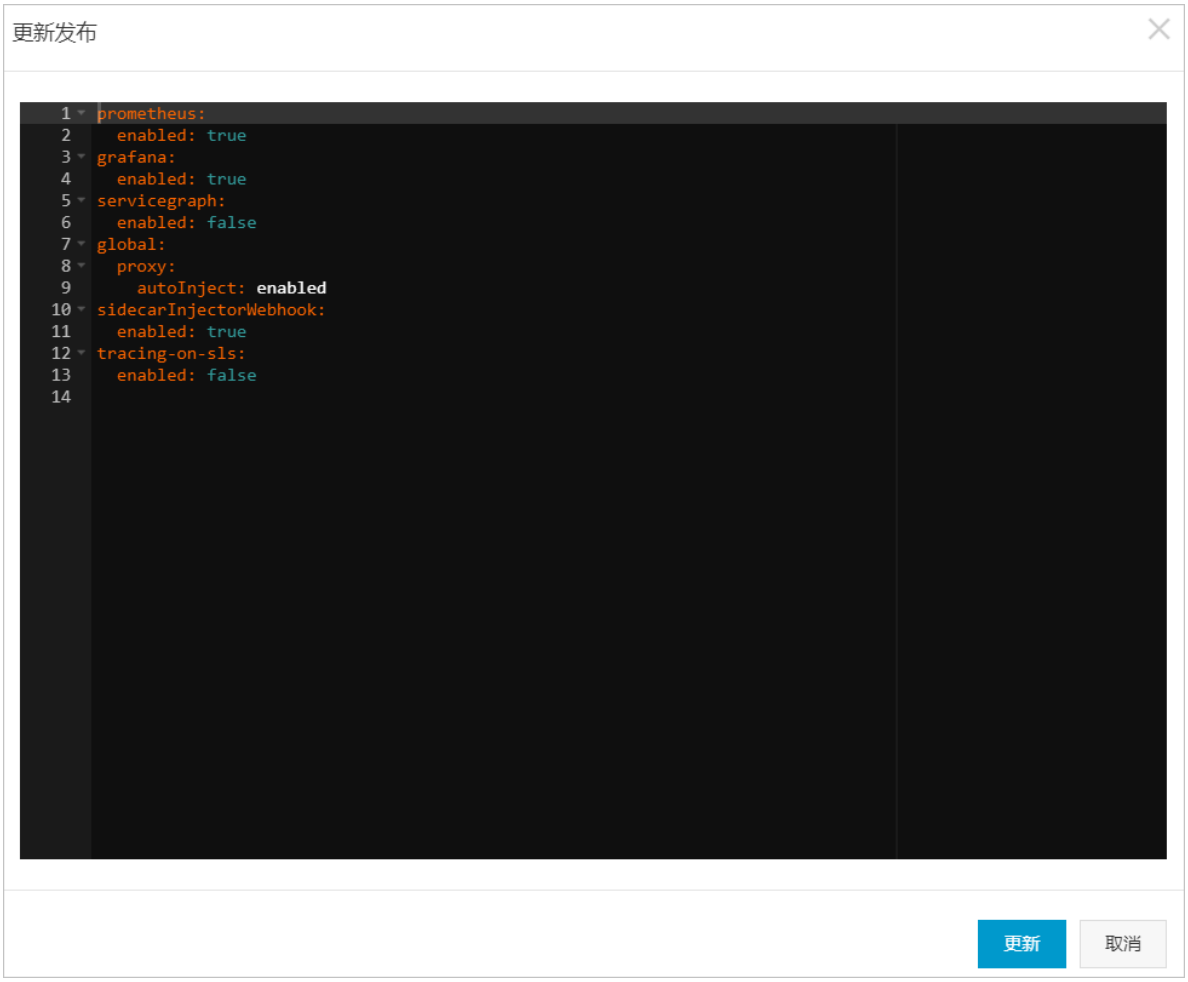
1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的應用 > **Helm**，進入發布列表頁面。
3. 選擇所需的叢集，選擇待更新的Istio，單擊操作列的更新。

> 说明：
>
> - 通過叢集介面部署的Istio，發布名稱為istio，更新的內容與部署時的選項一致。

- 通過應用目錄部署的Istio，發布名稱為使用者建立時指定的名稱，更新的內容與部署時的參數一致。

4. 在彈出的對話方塊中，對Istio的參數進行編輯後，單擊更新。

   此處以更新通過叢集頁面部署的Istio為例：



預期結果

可通過以下兩種方式，查看更新後的內容：

- 更新完成後，頁面自動跳轉到發布列表頁面，在資源頁簽，可以查看更新的內容。
- 在 Kubernetes 菜單下，單擊左側導覽列的應用 > 容器組，進入容器組頁面，選擇目的地組群和目標空間進行查看。

1.14.4 刪除Istio

您可以通過刪除操作，刪除已部署的Istio。

前提条件

- 您已成功建立一個 Kubernetes 叢集，參見*##Kubernetes##*。
- 您已成功建立一個**Istio**，參見*##Istio*。

操作步驟

1. 登入 *Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列的應用 > **Helm**，進入發布列表頁面。
3. 選擇所需的叢集及待刪除的Istio，單擊操作列的刪除。

4. 在彈出的對話方塊中，單擊確定。

说明：

- 不勾選清除發布記錄：
  - 發布記錄不會被刪除：

  - 此Istio的名稱不可被再次使用。

通過叢集介面重新部署Istio時，顯示已完成。

通過應用目錄部署Istio時，提示：已存在同名的部署或資源，請修改名稱。

- 勾選清除發布記錄，會同時刪除所有發布記錄，且此Istio的名稱可被再次使用。

建議保持預設狀態，勾選清除發布記錄。

預期結果

返回發布列表頁面，可以看到該Istio已被刪除。

1.15 應用目錄管理

1.15.1 應用目錄概述

微服務是容器時代的主題，應用微服務化給部署和管理帶來極大的挑戰。通過將龐大的單體應用拆分成一個個微服務，從而使各個微服務可被獨立部署和擴充，實現敏捷開發和快速迭代。雖然微服務帶來了很大的好處，但同時，由於應用拆分成許多組件，對應著龐大數量的微服務，開發人員不得不面對這些微服務的管理問題，如資源管理、版本管理、組態管理等。

針對 Kubernetes 編排下微服務管理問題，阿里雲Container Service引入 Helm 開源項目並進行整合，協助簡化部署和管理 Kubernetes 應用。

Helm 是 Kubernetes 服務編排領域的開源子項目，是 Kubernetes 應用的一個包管理工具， Helm 通過軟體打包的形式，支援發布的版本管理和控制，簡化了 Kubernetes 應用部署和管理的複雜性。

阿里雲應用目錄功能

阿里雲Container Service應用目錄功能整合了 Helm，提供了 Helm 的相關功能，並進行了相關功能擴充，如提供圖形化介面、阿里雲官方 Repository 等。

應用目錄首頁 chart 列表的資訊包含：

- chart 名稱： 一個 Helm 包，對應一個目標應用，其中包含了運行一個應用所需要的鏡像、依賴和資源定義等。
- 版本：chart 的版本號碼。
- Repository：用於發布和儲存 Chart 的倉庫，例如官方倉庫 stable、incubator 等。

各個 chart 詳情頁包含的資訊不盡相同，例如，可能包含：

- chart 簡介
- chart 詳細資料
- chart 安裝到叢集的前提條件，如預先配置持久化儲存卷(pv) 。
- chart 安裝命令
- chart 卸載命令
- chart 參數配置項

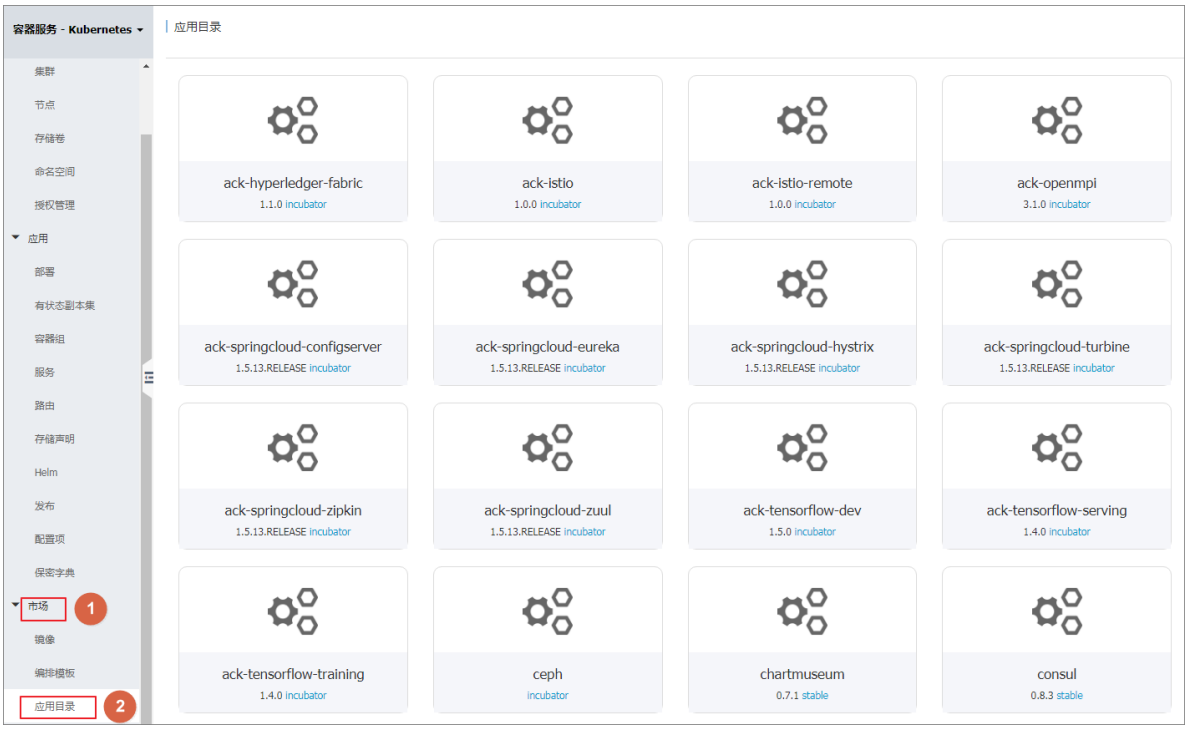
目前，您可以通過 helm 工具部署和管理應用目錄中的 chart，具體請參見*## Helm ######*。

1.15.2 查看應用目錄列表

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的市場 > 應用目錄，進入應用目錄列表。

   您可以查看該頁面下 chart 列表，每個 chart 對應一個目標應用，包含一些基本資料，如應用程式名稱、版本號碼和來源倉庫等資訊。



后续操作

您可進入單個 chart，瞭解具體的 chart 資訊，可根據相關資訊，利用 Helm 工具部署應用，具體請參見*## Helm ######* 。

1.16 服務類別目錄管理

1.16.1 概述

雲平台上啟動並執行應用需要使用一些基礎服務，如資料庫、應用伺服器等通用的基礎軟體。譬如一個 wordpress 應用，作為一個 web 應用，後端需要一個資料庫服務，如 MariaDB。傳統方式是在 wordpress 應用的編排中也建立應用依賴的 MariaDB 服務，並與 Web 應用進行整合。這種雲上應用開發的方式，就需要開發人員花費精力解決所依賴的基礎設施軟體的部署和配置，增加應用託管和遷移的成本。

阿里雲Container Service支援並整合了服務類別目錄的功能，該功能旨在接入和管理 Service Broker，使 kubernetes 上啟動並執行應用可以使用 service broker 所代理的託管服務。服務類別目錄功能將支援一系列基礎設施軟體，應用開發人員可以不用關心這些軟體的可用性和伸縮能力，也不用對其進行管理，開發人員可以簡單的將其作為服務使用，只需專註開發核心的應用程式。

服務類別目錄通過 Kubernetes 的 Open Service Broker API 與 Service Broker 進行通訊，並作為 Kubernetes API Server 的中介，以便協商首要規定（initial provisioning）並擷取應用程式使用託管服務的必要憑據。關於服務類別目錄的具體實現原理，請參考 *service catalog*。

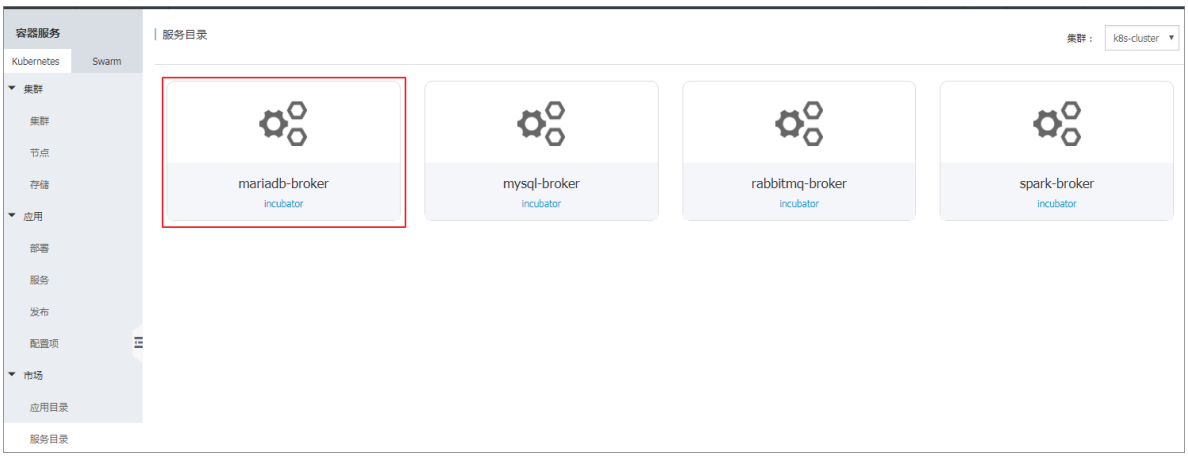
1.16.2 開通服務類別目錄

操作步驟

1. 登入*Container Service#####*。
2. 在 Kubernetes 菜單下，單擊左側導覽列中的市場 > 服務類別目錄，在右上方選擇目的地組群。
3. 若您還未部署服務類別目錄功能，介面上會提示您先進行安裝。

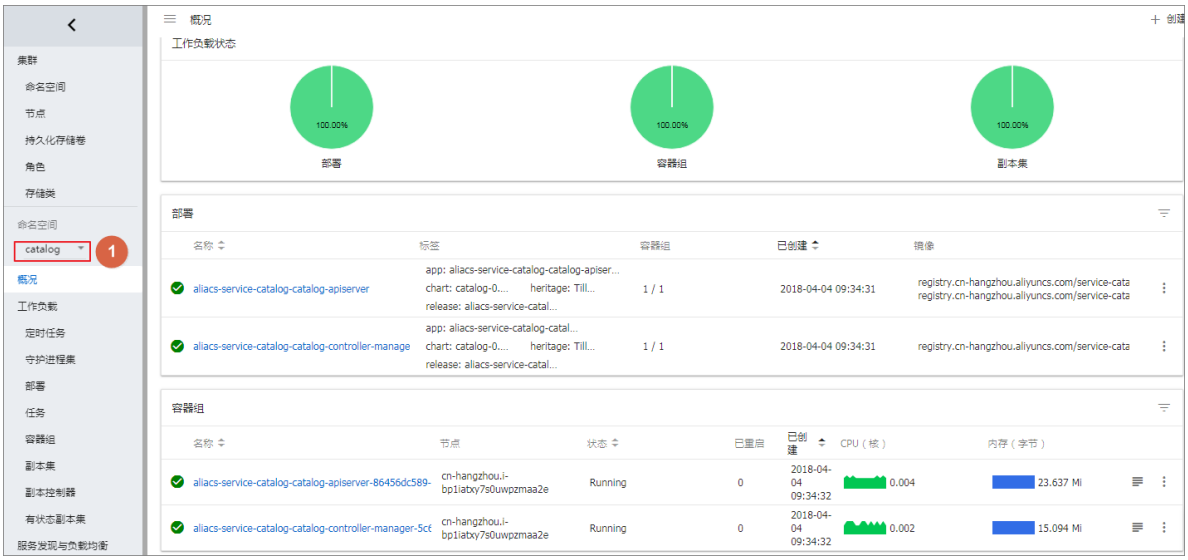
4. 安装完成後，服務類別目錄頁面中會顯示預設安裝的 Service Broker，您可以進入 mariadb-broker 瞭解詳情。

> 说明：
> 服務類別目錄具體實現為一個擴充 API server 和一個 controller，阿里雲Container Service安裝服務類別目錄功能後，會建立一個 catalog 命名空間。



5. 在左側導覽列中單擊叢集，單擊目的地組群右側的控制台。

6. 進入 Kubernetes Dashboard 頁面，在命名空間中選擇**catalog**，可以看到該命名空間下安裝了 catalog apiserver 和 controller 相關的資來源物件。



后续操作

至此，您已經成功開通服務類別目錄功能。接下來可以通過服務類別目錄下的 Service Broker 來建立託管服務的執行個體，並將其應用到您自己的應用程式中。