NDO資源的故障排除和檢查

目錄

簡介

本文檔介紹如何使用kubectl和container runtime CLI檢查並排除NDO故障。

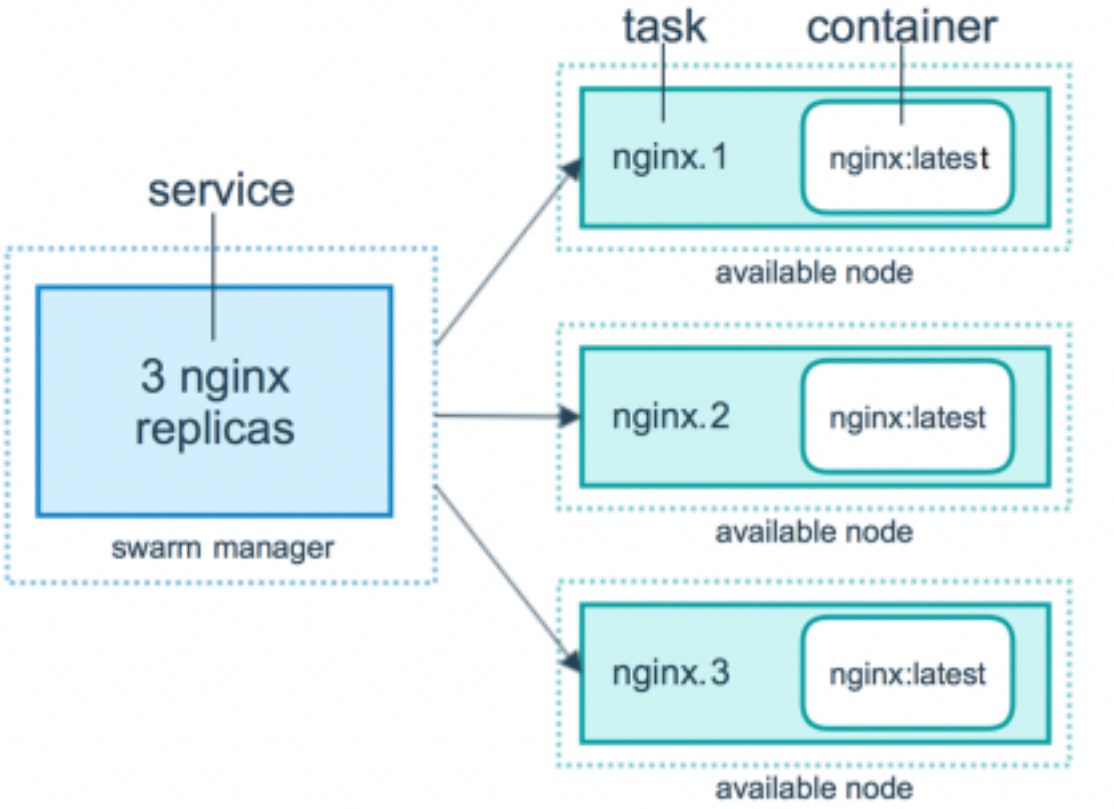
NDO快速啟動

Cisco Nexus Dashboard Orchestrator(NDO)是一個交換矩陣管理工具，允許使用者管理各種交換矩陣，包括思科®以應用為中心的基礎設施(Cisco ACI®)站點、思科雲ACI站點和思科Nexus控制面板交換矩陣控制器(NDFC)站點，每個站點都由其自己的控制器（APIC群集、NDFC群集或公共雲中的雲APIC例項）管理。

NDO通過單一平台跨多個資料中心提供一致的網路和策略協調、可擴充性和災難恢復。

在早期，MSC(Multi-Site Controller)被部署為帶有VMWare Open Virtual Appliances(OVA)的三節點集群，允許客戶初始化Docker Swarm集群和MSC服務。此Swarm群集將MSC微服務作為Docker容器和服務進行管理。

此圖片顯示了有關Docker Swarm如何管理作為同一容器副本的微服務以實現高可用性的簡化檢視。



Docker Swarm負責維護MSC架構中每個微服務的預期副本數。從Docker Swarm的角度來看，多站點控制器是唯一需要協調的容器部署。

Nexus Dashboard(ND)是適用於多個資料中心站點的中央管理控制檯，也是託管思科資料中心運營服務（包括Nexus Insight和MSC 3.3版）的通用平台，並將名稱更改為Nexus Dashboard Orchestrator(NDO)。

雖然構成MSC架構的大多數微服務仍然相同，但NDO部署在Kubernetes(K8s)集群中，而不是部署在Docker Swarm集群中。這允許ND協調多個應用程式或部署，而不是僅協調一個應用程式或部署。

Kubernetes with NDO Crash-Course

Kubernetes是一個開源系統，用於自動化容器化應用的部署、可擴充性和管理。作為Docker，Kubernetes與容器技術合作，但不與Docker合作。這意味著Kubernetes支援其他容器平台(Rkt、PodMan)。

Swarm和Kubernetes之間的一個關鍵區別是，後者並不直接與容器一起使用，而是與容器共處的概念，稱為Pods。

Pod中的容器必須在同一節點上運行。一組Pod稱為部署。Kubernetes部署可以描述整個應用程式。

Kubernetes還允許使用者確保一定數量的資源可用於任何給定的應用程式。這通過使用複製控制器來完成，以確保Pod的數量與應用程式清單一致。

清單是描述Cluster要部署的資源的YAML格式檔案。資源可以是前面描述的任何資源或其他可供使用者使用的資源。

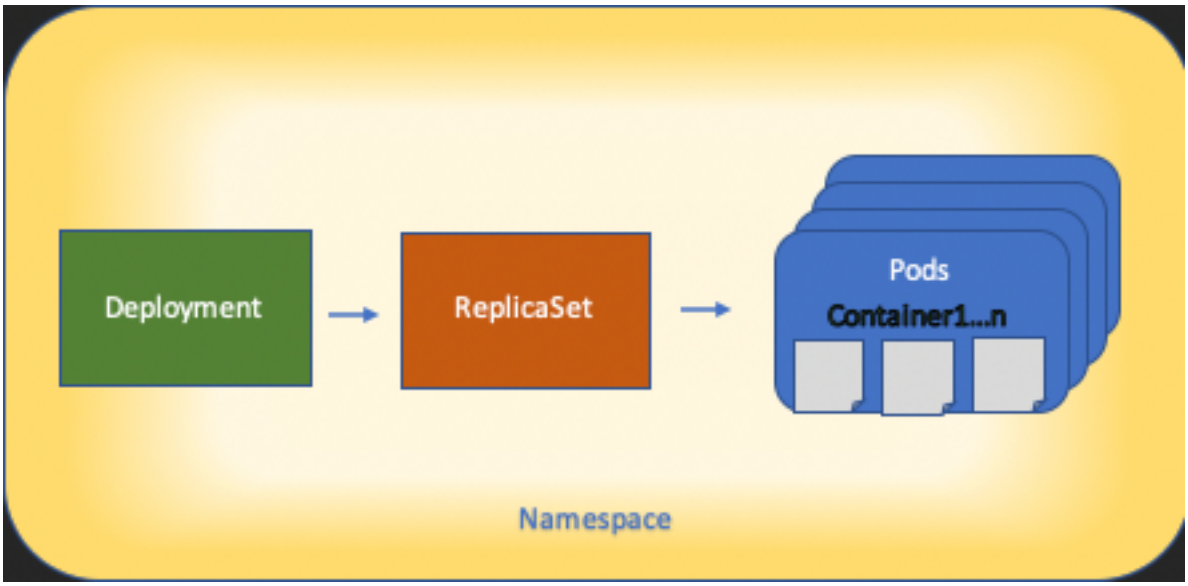
可以通過一個或多個服務從外部訪問應用程式。Kubernetes包括一個負載平衡器選項來完成此操作

。

Kubernetes還提供了一種通過名稱空間概念隔離不同資源的方法。ND使用名稱空間來唯一標識不同的應用程式和群集服務。運行CLI命令時，請始終指定Namespace。

雖然對ND或NDO進行故障排除時不需要深入瞭解Kubernetes，但需要對Kubernetes架構有基本瞭解，才能正確識別有問題或需要注意的資源。

Kubernetes資源體系結構的基本資訊如下圖所示：



記住各種資源與其他資源的互動方式是很重要的，並且它在審閱和故障排除過程中起著重要作用。

使用Kubernetes命令的NDO概述

CLI存取登入

對於通過SSH訪問NDO的CLI， **admin-user** 需要密碼。但是，我們使用 **rescue-user** 密碼。例如：

```
ssh rescue-user@ND-mgmt-IP
rescue-user@XX.XX.XX.XX's password:
[rescue-user@MxNDsh01 ~]$ pwd
/home/rescue-user
[rescue-user@MxNDsh01 ~]$
```

這是CLI訪問的預設模式和使用者，大部分資訊都可供檢視。

NDO名稱空間審查

此K8概念允許隔離群集中的不同資源。下一個命令可用於檢視部署的不同名稱空間：

```
[rescue-user@MxNDsh01 ~]$ kubectl get namespace
NAME                 STATUS   AGE
authy                Active   177d
authy-oidc           Active   177d
cisco-appcenter      Active   177d
cisco-intersightdc   Active   177d
```

```
cisco-mso            Active   176d
cisco-nir            Active   22d
clicks               Active   177d
confd                Active   177d
default              Active   177d
elasticsearch        Active   22d
eventmgr             Active   177d
firmwared            Active   177d
installer            Active   177d
kafka                Active   177d
kube-node-lease      Active   177d
kube-public          Active   177d
kube-system          Active   177d
kubese               Active   177d
maw                  Active   177d
mond                 Active   177d
mongodb              Active   177d
nodemgr              Active   177d
ns                   Active   177d
rescue-user          Active   177d
securitymgr          Active   177d
sm                   Active   177d
statscollect         Active   177d
ts                   Active   177d
zk                   Active   177d
```

粗體條目屬於NDO中的應用程式，而以字首開頭的實體屬於Kubernetes群集。每個名稱空間都有自己的獨立部署和Pod

kubectl CLI允許使用 **--namespace** 選項，如果沒有使用該選項運行命令，則CLI會假定名稱空間為 **default** （k8的名稱空間）：

```
[rescue-user@MxNDsh01 ~]$ kubectl get pod --namespace cisco-mso
NAME                                  READY   STATUS    RESTARTS   AGE
auditservice-648cd4c6f8-b29hh         2/2     Running   0          44h
…

[rescue-user@MxNDsh01 ~]$ kubectl get pod
No resources found in default namespace.
```

kubectl CLI允許輸出採用不同型別的格式，如yaml、JSON或自定義表。這是通過 **-o** [format]選項。例如：

```
[rescue-user@MxNDsh01 ~]$ kubectl get namespace -o JSON

{
   "apiVersion": "v1",
   "items": [
       {
           "apiVersion": "v1",
           "kind": "Namespace",
           "metadata": {
               "annotations": {
```

```
                    "kubectl.kubernetes.io/last-applied-configuration":
"{\"apiVersion\":\"v1\",\"kind\":\"Namespace\",\"metadata\":{\"annotations\":{},\"labels\":{\"se
rviceType\":\"infra\"},\"name\":\"authy\"}}\n"

                },

                "creationTimestamp": "2022-03-28T21:52:07Z",

                "labels": {

                    "serviceType": "infra"

                },

                "name": "authy",

                "resourceVersion": "826",

                "selfLink": "/api/v1/namespaces/authy",

                "uid": "373e9d43-42b3-40b2-a981-973bdddccd8d"

            },

        }

    ],

    "kind": "List",

    "metadata": {

        "resourceVersion": "",

        "selfLink": ""

    }

}
```


在上一文本中，輸出是一個字典，其中它的鍵之一被稱為**項**，值是一個詞典列**表，其中每個**字典都為Namespace條目的帳戶，其屬性是字典或巢狀字典中的鍵 — 值對值。

這是相關的，因為K8為使用者提供了選擇jsonpath作為輸出的選項，這允許對JSON資料陣列執行複雜的操作。例如，從上一個輸出中，如果我們訪問 **name** 對於名稱空間，我們需要訪問專案清單的值，然後 metadata ，然後獲取金鑰的值 **name**.可以使用以下命令來完成此操作：

```
[rescue-user@MxNDsh01 ~]$ kubectl get namespace -o=jsonpath='{.items[*].metadata.name}'

authy authy-oidc cisco-appcenter cisco-intersightdc cisco-mso cisco-nir clicks confd default
elasticsearch eventmgr firmwared installer kafka kube-node-lease kube-public kube-system kubese
maw mond mongodb nodemgr ns rescue-user securitymgr sm statscollect ts zk

[rescue-user@MxNDsh01 ~]$
```

所描述的層次結構用於獲取所需的特定資訊。基本上，所有專案都可在 **items** 列舉**項[*]**，然後鍵 metadata 和 **name** 使用**metadata.name**，查詢可以包含要顯示的其他值。

這同樣適用於自定義列的選項，該選項使用類似的方法從資料陣列中提取資訊。例如，如果我們建

立一個表，其中包含有關 **name** 和 UID 值，我們可以應用以下命令：

```
[rescue-user@MxNDsh01 ~]$ kubectl get namespace -o custom-
columns=NAME:.metadata.name,UID:.metadata.uid

NAME                 UID

authy                373e9d43-42b3-40b2-a981-973bdddccd8d

authy-oidc           ba54f83d-e4cc-4dc3-9435-a877df02b51e

cisco-appcenter      46c4534e-96bc-4139-8a5d-1d9a3b6aefdc

cisco-intersightdc   bd91588b-2cf8-443d-935e-7bd0f93d7256

cisco-mso            d21d4d24-9cde-4169-91f3-8c303171a5fc

cisco-nir            1c4dba1e-f21b-4ef1-abcf-026dbe418928

clicks               e7f45f6c-965b-4bd0-bf35-cbbb38548362

confd                302aebac-602b-4a89-ac1d-1503464544f7

default              2a3c7efa-bba4-4216-bb1e-9e5b9f231de2

elasticsearch        fa0f18f6-95d9-4cdf-89db-2175a685a761
```

輸出要求顯示每列的名稱，然後為輸出分配值。在此示例中，有兩列： **NAME** 和 **UID**.這些值屬於 .metada.name 和 .metadata.uid 分別.更多資訊和示例可從以下網址獲得：

JSONPath支援

自定義列

NDO部署稽核

部署是一個K8s對象，提供用於管理ReplicaSet和Pod的連線空間。部署處理屬於某個應用程式的所有Pod的展開以及每個應用程式的預期副本數。

kubectl CLI包含一個命令，用於檢查任何給定名稱空間的部署：

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso

NAME                 READY   UP-TO-DATE   AVAILABLE   AGE

auditservice         1/1     1            1           3d22h

backupservice        1/1     1            1           3d22h

cloudsecservice      1/1     1            1           3d22h

consistencyservice   1/1     1            1           3d22h

dcnmworker           1/1     1            1           3d22h
```

```
eeworker              1/1     1            1           3d22h

endpointservice       1/1     1            1           3d22h

executionservice      1/1     1            1           3d22h

fluentd               1/1     1            1           3d22h

importservice         1/1     1            1           3d22h

jobschedulerservice   1/1     1            1           3d22h

notifyservice         1/1     1            1           3d22h

pctagvnidservice      1/1     1            1           3d22h

platformservice       1/1     1            1           3d22h

platformservice2      1/1     1            1           3d22h

policyservice         1/1     1            1           3d22h

schemaservice         1/1     1            1           3d22h

sdaservice            1/1     1            1           3d22h

sdwanservice          1/1     1            1           3d22h

siteservice           1/1     1            1           3d22h

siteupgrade           1/1     1            1           3d22h

syncengine            1/1     1            1           3d22h

templateeng           1/1     1            1           3d22h

ui                    1/1     1            1           3d22h

userservice           1/1     1            1           3d22h
```

我們可以使用相同的自定義表 deployment **而不是 namespace 和 -n 選項以檢視與之前相同的資訊。 這是因為輸出採用類似的方式。**

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso -o custom-
columns=NAME:.metadata.name,UID:.metadata.uid

NAME                  UID

auditservice          6e38f646-7f62-45bc-add6-6e0f64fb14d4

backupservice         8da3edfc-7411-4599-8746-09feae75afee

cloudsecservice       80c91355-177e-4262-9763-0a881eb79382

consistencyservice    ae3e2d81-6f33-4f93-8ece-7959a3333168

dcnmworker            f56b8252-9153-46bf-af7b-18aa18a0bb97

eeworker              c53b644e-3d8e-4e74-a4f5-945882ed098f

endpointservice       5a7aa5a1-911d-4f31-9d38-e4451937d3b0

executionservice      3565e911-9f49-4c0c-b8b4-7c5a85bb0299
```

```
fluentd               c97ea063-f6d2-45d6-99e3-1255a12e7026

importservice         735d1440-11ac-41c2-afeb-9337c9e8e359

jobschedulerservice   e7b80ec5-cc28-40a6-a234-c43b399edbe3

notifyservice         75ddb357-00fb-4cd8-80a8-14931493cfb4

pctagvnidservice      ebf7f9cf-964e-46e5-a90a-6f3e1b762979

platformservice       579eaae0-792f-49a0-accc-d01cab8b2891

platformservice2      4af222c9-7267-423d-8f2d-a02e8a7a3c04

policyservice         d1e2fff0-251a-447f-bd0b-9e5752e9ff3e

schemaservice         a3fca8a3-842b-4c02-a7de-612f87102f5c

sdaservice            d895ae97-2324-400b-bf05-b3c5291f5d14

sdwanservice          a39b5c56-8650-4a4b-be28-5e2d67cae1a9

siteservice           dff5aae3-d78b-4467-9ee8-a6272ee9ca62

siteupgrade           70a206cc-4305-4dfe-b572-f55e0ef606cb

syncengine            e0f590bf-4265-4c33-b414-7710fe2f776b

templateeng           9719434c-2b46-41dd-b567-bdf14f048720

ui                    4f0b3e32-3e82-469b-9469-27e259c64970

userservice           73760e68-4be6-4201-959e-07e92cf9fbb3
```

請記住，顯示的副本數量是針對部署的，而不是每個微服務的Pod數量。

我們可以使用關鍵字 describe 而不是 get 要顯示有關資源的更多詳細資訊，在此情況下為架構服務部署：

```
[rescue-user@MxNDsh01 ~]$ kubectl describe deployment -n cisco-mso schemaservice

Name:               schemaservice

Namespace:          cisco-mso

CreationTimestamp:  Tue, 20 Sep 2022 02:04:58 +0000

Labels:             k8s-app=schemaservice

                   scaling.case.cncf.io=scale-service

Annotations:        deployment.kubernetes.io/revision: 1

                   kubectl.kubernetes.io/last-applied-configuration:

                     {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"c
reationTimestamp":null,"labels":{"k8s-app":"schemaservice","sca...

Selector:           k8s-app=schemaservice

Replicas:           1 desired | 1 updated | 1 total | 1 available | 0 unavailable
```

```
StrategyType:       Recreate

MinReadySeconds:    0

Pod Template:

 Labels:           cpu.resource.case.cncf.io/schemaservice=cpu-lg-service

                   k8s-app=schemaservice

                   memory.resource.case.cncf.io/schemaservice=mem-xlg-service

 Service Account:  cisco-mso-sa

 Init Containers:

  init-msc:

   Image:      cisco-mso/tools:3.7.1j

   Port:       <none>

   Host Port:  <none>

   Command:

     /check_mongo.sh

   Environment:  <none>

   Mounts:

     /secrets from infracerts (rw)

 Containers:

  schemaservice:

   Image:       cisco-mso/schemaservice:3.7.1j

   Ports:       8080/TCP, 8080/UDP

   Host Ports:  0/TCP, 0/UDP

   Command:

     /launchscala.sh

     schemaservice

   Liveness:  http-get http://:8080/api/v1/schemas/health delay=300s timeout=20s period=30s
#success=1 #failure=3

   Environment:

     JAVA_OPTS:  -XX:+IdleTuningGcOnIdle

   Mounts:

     /jwtsecrets from jwtsecrets (rw)

     /logs from logs (rw)
```

```
      /secrets from infracerts (rw)
  msc-schemaservice-ssl:
   Image:       cisco-mso/sslcontainer:3.7.1j
   Ports:       443/UDP, 443/TCP
   Host Ports:  0/UDP, 0/TCP
   Command:
     /wrapper.sh
   Environment:
     SERVICE_PORT:  8080
   Mounts:
     /logs from logs (rw)
     /secrets from infracerts (rw)
  schemaservice-leader-election:
   Image:      cisco-mso/tools:3.7.1j
   Port:       <none>
   Host Port:  <none>
   Command:
     /start_election.sh
   Environment:
     SERVICENAME:  schemaservice
   Mounts:
     /logs from logs (rw)
 Volumes:
  logs:
   Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
   ClaimName:  mso-logging
   ReadOnly:   false
  infracerts:
   Type:        Secret (a volume populated by a Secret)
   SecretName:  cisco-mso-secret-infra
   Optional:    false
  jwtsecrets:
```

```
    Type:        Secret (a volume populated by a Secret)
    SecretName:  cisco-mso-secret-jwt
    Optional:    false
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
Events:          <none>
[rescue-user@MxNDsh01 ~]$
```

其 **describe** 命令還允許包含 **--show-events=true** 選項以顯示部署的任何相關事件。

[擾流器]

NDO副本集(RS)檢查

[擾流器]

####僅對根使用者#####可用

Replica Set(RS)是一個K8s對象，其目的是保持穩定數量的副本Pod。此對象還檢測通過定期探測到Pod發現不正常數量的複製副本的時間。

RS也以名稱空間組織。

```
[root@MxNDsh01 ~]#  kubectl get rs  -n cisco-mso
NAME                              DESIRED   CURRENT   READY   AGE
auditservice-648cd4c6f8           1         1         1       3d22h
backupservice-64b755b44c          1         1         1       3d22h
cloudsecservice-7df465576         1         1         1       3d22h
consistencyservice-c98955599      1         1         1       3d22h
dcnmworker-5d4d5cbb64             1         1         1       3d22h
eeworker-56f9fb9ddb               1         1         1       3d22h
endpointservice-7df9d5599c        1         1         1       3d22h
executionservice-58ff89595f       1         1         1       3d22h
fluentd-86785f89bd                1         1         1       3d22h
importservice-88bcc8547           1         1         1       3d22h
jobschedulerservice-5d4fdfd696    1         1         1       3d22h
```

```
notifyservice-75c988cfd4         1         1         1       3d22h
pctagvnidservice-644b755596      1         1         1       3d22h
platformservice-65cddb946f       1         1         1       3d22h
platformservice2-6796576659      1         1         1       3d22h
policyservice-545b9c7d9c         1         1         1       3d22h
schemaservice-7597ff4c5          1         1         1       3d22h
sdaservice-5f477dd8c7            1         1         1       3d22h
sdwanservice-6f87cd999d          1         1         1       3d22h
siteservice-86bb756585           1         1         1       3d22h
siteupgrade-7d578f9b6d           1         1         1       3d22h
syncengine-5b8bdd6b45            1         1         1       3d22h
templateeng-5cbf9fdc48           1         1         1       3d22h
ui-84588b7c96                    1         1         1       3d22h
userservice-87846f7c6            1         1         1       3d22h
```

其 describe 選項包括有關URL、探測器使用的埠以及測試週期和故障閾值的資訊。

```
[root@MxNDsh01 ~]#  kubectl describe rs  -n cisco-mso schemaservice-7597ff4c5
Name:           schemaservice-7597ff4c5
Namespace:      cisco-mso
Selector:       k8s-app=schemaservice,pod-template-hash=7597ff4c5
Labels:         cpu.resource.case.cncf.io/schemaservice=cpu-lg-service
                k8s-app=schemaservice
                memory.resource.case.cncf.io/schemaservice=mem-xlg-service
                pod-template-hash=7597ff4c5
Annotations:    deployment.kubernetes.io/desired-replicas: 1
                deployment.kubernetes.io/max-replicas: 1
                deployment.kubernetes.io/revision: 1
Controlled By:  Deployment/schemaservice
Replicas:       1 current / 1 desired
Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
 Labels:           cpu.resource.case.cncf.io/schemaservice=cpu-lg-service
```

```
                  k8s-app=schemaservice
                  memory.resource.case.cncf.io/schemaservice=mem-xlg-service
                  pod-template-hash=7597ff4c5
Service Account:  cisco-mso-sa
Init Containers:
 init-msc:
  Image:      cisco-mso/tools:3.7.1j
  Port:       <none>
  Host Port:  <none>
  Command:
    /check_mongo.sh
  Environment:  <none>
  Mounts:
    /secrets from infracerts (rw)
Containers:
 schemaservice:
  Image:       cisco-mso/schemaservice:3.7.1j
  Ports:       8080/TCP, 8080/UDP
  Host Ports:  0/TCP, 0/UDP
  Command:
    /launchscala.sh
    schemaservice
   Liveness:  http-get http://:8080/api/v1/schemas/health delay=300s timeout=20s period=30s
#success=1 #failure=3
  Environment:
    JAVA_OPTS:  -XX:+IdleTuningGcOnIdle
  Mounts:
    /jwtsecrets from jwtsecrets (rw)
    /logs from logs (rw)
    /secrets from infracerts (rw)
 msc-schemaservice-ssl:
  Image:       cisco-mso/sslcontainer:3.7.1j
  Ports:       443/UDP, 443/TCP
```

```
    Host Ports:  0/UDP, 0/TCP

    Command:

      /wrapper.sh
```

NDO副本集(RS)審閱#####僅對根使用者#####可用A副本集(RS)是一個K8s對象，目標是維護穩定數量的副本Pod。此對象還檢測通過定期探測到Pod發現不正常數量的複製副本的時間。 RS也以名稱空間組織。[root@MxNDsh01 ~]# kubectl get rs -n cisco-msoNAME DESIRED CURRENT READY AUDITSERVICE-648cd4c6f8 1 1 1 3d22hbackupservice-64b755b44c 1 1 1 3d22hcloudsecservice-7df465576 1 1 1 1 3d22hconsistencyservice-c98955599 1 1 1 3d22hdcnmworker-5d5cbb64 1 1 3d22heeworker-56f fb9ddb 1 1 1 3d22hendpointservice-7df9d5599c 1 1 1 3d22hexecutionservice-58ff89595f 1 1 3d22hfluentd-86785f89bd 1 1 1 3d22himportservice-88bcc8547 1 1 1 3d22hjobschedulerservice-5d4fdfd696 1 1 1 3d22hnotifyservice-75c984 1 1 3d22hpctagvnidservice-644b755596 1 1 1 3d22platformservice-65cddb946f 1 1 1 3d22platformservice2-6796576659 1 1 1 3d22hpolicyservice-545b9c7d9c 1 1 1 3d22hschemaservice-7597ff4c5 1 1 3d22hsdaservice-5f478d7 1 1 1 3d26hsdwanservice f87cd999d 1 1 1 3d22hteservice-86bb756585 1 1 1 3d22hteupgrade-7d578f9b6d 1 1 3d22hsyncengine-5b8bdd6b45 1 1 1 3d22htemplateeng-5cbf9fdc48 1 1 1 3d22hui-84588b7c96 1 1 1 3d22huserservice-87846f7c6 1 1 3d22h describe選項包含有關URL、探測器使用的埠以及測試週期性和故障閾值的資訊。[root@MxNDsh01 ~]# kubectl describe rs rs -n cisco-mso schemaservice-7597ff4c5名稱： schemaservice-7597ff4c5名稱空間： cisco-mso選擇器： k8s-app=schemaservice，pod-template-hash=7597ff4c5標籤： cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service pod-template-hash=7597ff4c5註釋： deployment.kubernetes.io/desired-replicas: deployment.kubernetes.io/max-replicas: deployment.kubernetes.io/revision: 1控制者： Deployment/schemaservice副本：1 / 1 Pods狀態：1正在運行/0等待/0成功/0失敗Pod模板：標籤：cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service pod-template-hash=7597ff4c5服務帳戶：cisco-mso-sa Init容器：init-msc：映像：cisco-mso/tools:3.7.1j埠：<none>主機埠：<none>命令：/check_mongo.sh/launchscala.sh http://:8080/api/v1/schemas/health /wrapper.sh環境：<none>裝載：/infracerts(rw)容器的機密：schemaservice：映像： cisco-mso/schemaservice:3 7.1j埠：8080/TCP、8080/UDP主機埠：0/TCP、0/UDP命令：.....schemaservice活動性：http-get延遲=300s超時=20s週期=30s #success=1 #failure=3環境：JAVA_OPTS: -XX:+IdleTuningGcOnIdle掛載： /jwtsecrets(rw)日誌中的日誌(rw)/infracerts(rw)msc-schemts service-ssl：影象：cisco-mso/sslcontainer:3.7.1j埠：443/UDP，443/TCP主機埠：0/UDP，0/TCP命令：

NDO Pod稽核

Pod是在同一個Linux名稱空間（不同於K8s名稱空間）和同一個K8s節點中運行的一組緊密相關的容器。這是K8處理的最具原子性的對象，因為它不與容器互動。該應用程式可由單個容器組成，或者對於許多容器而言更複雜。使用下一個命令，我們可以檢查任何給定名稱空間的Pod:

```
[rescue-user@MxNDsh01 ~]$ kubectl get pod --namespace cisco-mso

NAME                                  READY   STATUS    RESTARTS   AGE

auditservice-648cd4c6f8-b29hh         2/2     Running   0          2d1h

backupservice-64b755b44c-vcpf9        2/2     Running   0          2d1h

cloudsecservice-7df465576-pwbh4       3/3     Running   0          2d1h
```

```
consistencyservice-c98955599-qlsx5     3/3     Running   0          2d1h
dcnmworker-5d4d5cbb64-qxbt8            2/2     Running   0          2d1h
eeworker-56f9fb9ddb-tjggb              2/2     Running   0          2d1h
endpointservice-7df9d5599c-rf9bw       2/2     Running   0          2d1h
executionservice-58ff89595f-xf8vz      2/2     Running   0          2d1h
fluentd-86785f89bd-q5wdp               1/1     Running   0          2d1h
importservice-88bcc8547-q4kr5          2/2     Running   0          2d1h
jobschedulerservice-5d4fdfd696-tbvqj   2/2     Running   0          2d1h
mongodb-0                              2/2     Running   0          2d1h
notifyservice-75c988cfd4-pkkfw         2/2     Running   0          2d1h
pctagvnidservice-644b755596-s4zjh      2/2     Running   0          2d1h
platformservice-65cddb946f-7mkzm       3/3     Running   0          2d1h
platformservice2-6796576659-x2t8f      4/4     Running   0          2d1h
policyservice-545b9c7d9c-m5pbf         2/2     Running   0          2d1h
schemaservice-7597ff4c5-w4x5d          3/3     Running   0          2d1h
sdaservice-5f477dd8c7-l5jn7            2/2     Running   0          2d1h
sdwanservice-6f87cd999d-6fjb8          3/3     Running   0          2d1h
siteservice-86bb756585-5n5vb           3/3     Running   0          2d1h
siteupgrade-7d578f9b6d-7kqkf           2/2     Running   0          2d1h
syncengine-5b8bdd6b45-2sr9w            2/2     Running   0          2d1h
templateeng-5cbf9fdc48-fqwd7           2/2     Running   0          2d1h
ui-84588b7c96-7rfvf                    1/1     Running   0          2d1h
userservice-87846f7c6-lzctd            2/2     Running   0          2d1h

[rescue-user@MxNDsh01 ~]$
```

第二列中顯示的數字表示每個Pod的容器數。

其 describe 選項也可用，其中包括有關每個Pod上容器的詳細資訊。

```
[rescue-user@MxNDsh01 ~]$ kubectl describe pod  -n cisco-mso schemaservice-7597ff4c5-w4x5d
Name:         schemaservice-7597ff4c5-w4x5d
Namespace:    cisco-mso
Priority:     0
Node:         mxndsh01/172.31.0.0
Start Time:   Tue, 20 Sep 2022 02:04:59 +0000
```

```
Labels:       cpu.resource.case.cncf.io/schemaservice=cpu-lg-service
              k8s-app=schemaservice
              memory.resource.case.cncf.io/schemaservice=mem-xlg-service
              pod-template-hash=7597ff4c5
Annotations:  k8s.v1.cni.cncf.io/networks-status:
                [{
                    "name": "default",
                    "interface": "eth0",
                    "ips": [
                        "172.17.248.16"
                    ],
                    "mac": "3e:a2:bd:ba:1c:38",
                    "dns": {}
                }]
              kubernetes.io/psp: infra-privilege
Status:       Running
IP:           172.17.248.16
IPs:
 IP:           172.17.248.16
Controlled By:  ReplicaSet/schemaservice-7597ff4c5
Init Containers:
 init-msc:
   Container ID:  cri-o://0c700f4e56a6c414510edcb62b779c7118fab9c1406fdac49e742136db4efbb8
   Image:         cisco-mso/tools:3.7.1j
   Image ID:      172.31.0.0:30012/cisco-
mso/tools@sha256:3ee91e069b9bda027d53425e0f1261a5b992dbe2e85290dfca67b6f366410425
   Port:          <none>
   Host Port:     <none>
   Command:
     /check_mongo.sh
   State:          Terminated
     Reason:       Completed
```

```
      Exit Code:    0
      Started:      Tue, 20 Sep 2022 02:05:39 +0000
      Finished:     Tue, 20 Sep 2022 02:06:24 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /secrets from infracerts (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from cisco-mso-sa-token-tn45l (ro)
Containers:
 schemaservice:
    Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac
    Image:         cisco-mso/schemaservice:3.7.1j
    Image ID:      172.31.0.0:30012/cisco-
mso/schemaservice@sha256:6d9fae07731cd2dcaf17c04742d2d4a7f9c82f1fc743fd836fe59801a21d985c
    Ports:         8080/TCP, 8080/UDP
    Host Ports:    0/TCP, 0/UDP
    Command:
      /launchscala.sh
      schemaservice
    State:          Running
      Started:      Tue, 20 Sep 2022 02:06:27 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     8
      memory:  30Gi
    Requests:
      cpu:     500m
      memory:  2Gi
```

顯示的資訊包括每個容器的容器影象並顯示使用的容器運行時。在本例中，CRI-O(cri-o)，早期版本的ND用於Docker，這會影響如何附加到容器。

[擾流器](#)

例如， **cri-o** ，我們想通過互動式會話連線到容器(通過 exec -it 選項)；而不是從輸出的 docker 命令，則使用**crictl**命令：

```
schemaservice:

   Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac

   Image:         cisco-mso/schemaservice:3.7.1j
```

使用以下命令：

```
[root@MxNDsh01 ~]# crictl exec -it
d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac bash

root@schemaservice-7597ff4c5-w4x5d:/#

root@schemaservice-7597ff4c5-w4x5d:/# whoami

root
```

對於以後的ND版本，要使用的容器ID不同。首先，我們需要使用命令 **crictl ps** 列出每個節點上運行的所有容器。我們可以根據需要過濾結果。

```
[root@singleNode ~]# crictl ps| grep backup
a9bb161d67295 10.31.125.241:30012/cisco-
mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 2 days
ago that run msc-backupservice-ssl 0 84b3c691cfc2b
4b26f67fc10cf 10.31.125.241:30012/cisco-
mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2 days
ago Running backupservice 0 84b3c691cfc2b
[root@singleNode ~]#
```

使用第一列中的值，我們可以使用與之前相同的命令來訪問Container run-time:

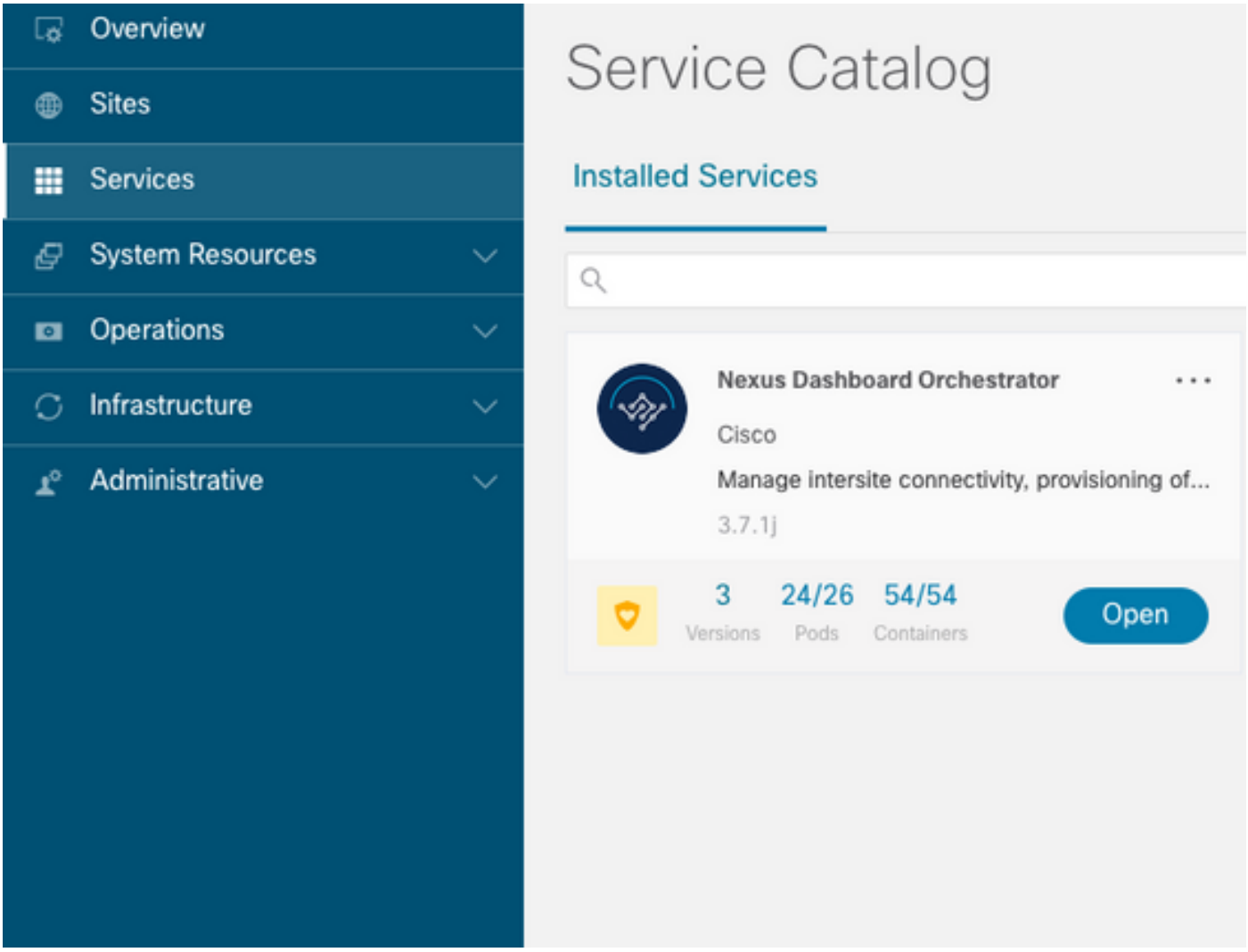
```
[root@singleNode ~]# crictl exec -it 4b26f67fc10cf bash
root@backupservice-8c699779f-j9jtr:/# pwd
/
```

例如，使用cri-o時，我們希望通過互動式會話將容器（通過exec -it選項）連線到上一個輸出的容器；但不要使用docker命令，而是使用crictl命令：schemaservice：容器ID:cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac 映像：cisco-mso/schemaservice:3.7.1j我們使用以下命令：[root@MxNDsh01 ~]# crictl exec -it d2287f8659dec6848c0100b7d24aeebd50 6f3f77af660238ca0c9c7e8946f4ac bashroot@schemaservice-7597ff4c5-w4x5d:/#root@schemaservice-7597ff4c5-w4x5d:/# whoamiroot對於以後的ND版本，要使用的容器ID不同。首先，我們需要使用命令crictl ps列出在每個節點上運行的所有容器。我們可以根據需要過濾結果。[root@singleNode ~]# crictl ps| grep backupa9bb161d67295 10.31.125.241:30012/cisco-mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 2天前運行msc-backupservice-ssl 0 84b3c691cfc2b4b26f67fc10cf 10.31.125.241:30012/cisco-mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2天前運行backupservice 0 84b3c691cfc2b[root@singleNode ~]#使用第一列中的值，然後可以使用與之前相同的命令訪問Container運行時：[root@singleNode ~]# cricexec -it 4b26f67 10cf bashroot@backupservice-8c699779f-j9jtr:/# pwd/

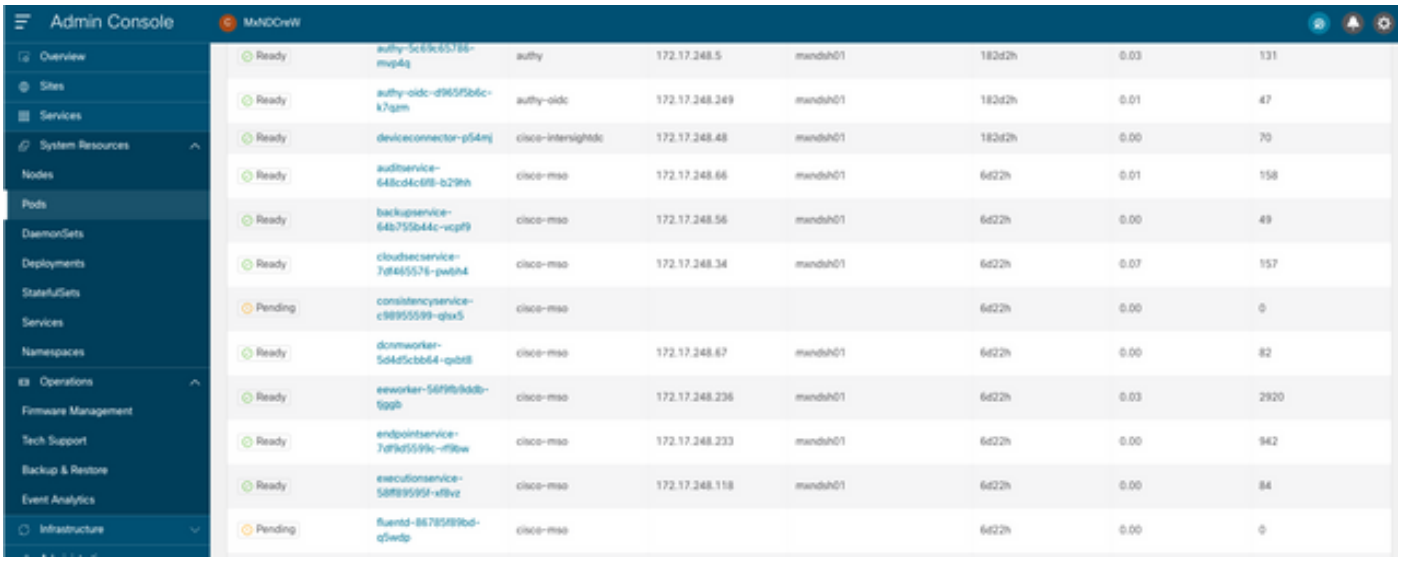
使用案例Pod不正常

我們可以使用此資訊來排除部署中的Pod不正常的原因。在本示例中，Nexus Dashboard版本為2.2-1d，受影響的應用程式為Nexus Dashboard Orchestrator(NDO)。

NDO GUI從「服務」檢視顯示一組不完整的Pod。本例中26個Pod中的24個。



另一個檢視位於 **System Resources -> Pods** 檢視Pod顯示不同於以下狀態的位置 **Ready**.



不正常Pod的CLI故障排除

如果已知名稱空間為cisco-mso（儘管進行故障排除時，其他應用/名稱空間也一樣），則如果存在不正常的Pod檢視，則會顯示Pod檢視：

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso
NAME READY UP-TO-DATE AVAILABLE AGE
auditservice 1/1 1 1 6d18h
backupservice 1/1 1 1 6d18h
cloudsecservice 1/1 1 1 6d18h
consistencyservice 0/1 1 0 6d18h <---
fluentd 0/1 1 0 6d18h <---
syncengine 1/1 1 1 6d18h
templateeng 1/1 1 1 6d18h
ui 1/1 1 1 6d18h
userservice 1/1 1 1 6d18h
```

在本例中，我們將重點放在一致性服務Pod中。通過JSON輸出，我們可以使用jsonpath從狀態欄位獲取特定資訊：

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso consistencyservice -o json
{
<--- OUTPUT OMITTED ---->
"status": {
"conditions": [
{
"message": "Deployment does not have minimum availability.",
"reason": "MinimumReplicasUnavailable",
},
{
"message": "ReplicaSet \"consistencyservice-c98955599\" has timed out progressing.",
"reason": "ProgressDeadlineExceeded",
}
],
}
}
[rescue-user@MxNDsh01 ~]$
```

我們看到**status**字典，在一個名為**conditions**的清單中，該清單包含詞典，作為包含**key message**和**value**的專案，其中{"\n"}部分是在結尾建立新行：

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso consistencyservice -
o=jsonpath='{.status.conditions[*].message}{"\n"}'
Deployment does not have minimum availability. ReplicaSet "consistencyservice-c98955599" has
timed out progressing.
[rescue-user@MxNDsh01 ~]$
```

此命令顯示如何從 **get Pod** 對於名稱空間：

```
[rescue-user@MxNDsh01 ~]$ kubectl get pods -n cisco-mso
NAME READY STATUS RESTARTS AGE
consistencyservice-c98955599-qlsx5 0/3 Pending 0 6d19h
executionservice-58ff89595f-xf8vz 2/2 Running 0 6d19h
fluentd-86785f89bd-q5wdp 0/1 Pending 0 6d19h
importservice-88bcc8547-q4kr5 2/2 Running 0 6d19h
jobschedulerservice-5d4fdfd696-tbvqj 2/2 Running 0 6d19h
```

```
mongodb-0 2/2 Running 0 6d19h
```

使用 get pods 命令，我們可以獲取與之前輸出中的問題匹配的Pod ID。在此範例中 **consistencyservice-c98955599-qlsx5**.

JSON輸出格式還提供如何從給定輸出檢查特定資訊的方法。

```
[rescue-user@MxNDsh01 ~]$ kubectl get pods -n cisco-mso consistencyservice-c98955599-qlsx5 -o
json
{
<--- OUTPUT OMITTED ---->
"spec": {
<--- OUTPUT OMITTED ---->
"containers": [
{
<--- OUTPUT OMITTED ---->
"resources": {
"limits": {
"cpu": "8",
"memory": "8Gi"
},
"requests": {
"cpu": "500m",
"memory": "1Gi"
}
},
<--- OUTPUT OMITTED ---->
"status": {
"conditions": [
{
"lastProbeTime": null,
"lastTransitionTime": "2022-09-20T02:05:01Z",
"message": "0/1 nodes are available: 1 Insufficient cpu.",
"reason": "Unschedulable",
"status": "False",
"type": "PodScheduled"
}
],
"phase": "Pending",
"qosClass": "Burstable"
}
}
[rescue-user@MxNDsh01 ~]$
```

JSON輸出必須在具有相同名稱的屬性中包含有關狀態的資訊。該消息包含有關原因的資訊。

```
[rescue-user@MxNDsh01 ~]$ kubectl get pods -n cisco-mso consistencyservice-c98955599-qlsx5 -
o=jsonpath='{.status}{"\n"}'
map[conditions:[map[lastProbeTime:<nil> lastTransitionTime:2022-09-20T02:05:01Z message:0/1
nodes are available: 1 Insufficient cpu. reason:Unschedulable status:False type:PodScheduled]]
phase:Pending qosClass:Burstable]
[rescue-user@MxNDsh01 ~]$
```

我們可以訪問有關Pod的狀態和要求的資訊：

```
[rescue-user@MxNDsh01 ~]$ kubectl get pods -n cisco-mso consistencyservice-c98955599-qlsx5 -
o=jsonpath='{.spec.containers[*].resources.requests}{"\n"}'
map[cpu:500m memory:1Gi]
```

在此必須提及如何計算價值。在本例中，cpu 500m指500 miligores，記憶體中的1G表示GB。

其 **Describe** 節點的選項顯示集群中每個K8工作執行緒（主機或VM）可用的資源：

```
[rescue-user@MxNDsh01 ~]$ kubectl describe nodes | egrep -A 6 "Allocat"
Allocatable:
cpu: 13
ephemeral-storage: 4060864Ki
hugepages-1Gi: 0
hugepages-2Mi: 0
memory: 57315716Ki
pods: 110
--
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource Requests Limits
-------- -------- ------
cpu 13 (100%) 174950m (1345%)
memory 28518Mi (50%) 354404Mi (633%)
ephemeral-storage 0 (0%) 0 (0%)
>[rescue-user@MxNDsh01 ~]$
```

Allocatable部分顯示每個節點可用的CPU、記憶體和儲存中的總資源。Allocated部分顯示已在使用的資源。CPU的**值13**指**13** 個核心或13,000(13K)毫核**核**。

在本例中，節點超額使用，**這解釋了為什麼Pod無法啟動。**通過刪除ND APP或新增VM資源清除ND後。

集群會不斷嘗試部署任何掛起的策略，因此，如果資源是空閒的，則可以部署Pod。

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso
NAME READY UP-TO-DATE AVAILABLE AGE
auditservice 1/1 1 1 8d
backupservice 1/1 1 1 8d
cloudsecservice 1/1 1 1 8d
consistencyservice 1/1 1 1 8d
dcnmworker 1/1 1 1 8d
eeworker 1/1 1 1 8d
endpointservice 1/1 1 1 8d
executionservice 1/1 1 1 8d
fluentd 1/1 1 1 8d
importservice 1/1 1 1 8d
jobschedulerservice 1/1 1 1 8d
notifyservice 1/1 1 1 8d
pctagvnidservice 1/1 1 1 8d
platformservice 1/1 1 1 8d
platformservice2 1/1 1 1 8d
policyservice 1/1 1 1 8d
schemaservice 1/1 1 1 8d
sdaservice 1/1 1 1 8d
sdwanservice 1/1 1 1 8d
siteservice 1/1 1 1 8d
siteupgrade 1/1 1 1 8d
syncengine 1/1 1 1 8d
templateeng 1/1 1 1 8d
ui 1/1 1 1 8d
userservice 1/1 1 1 8d
```

使用用於資源檢查的命令，我們確認集群具有可用的CPU資源：

```
[rescue-user@MxNDsh01 ~]$ kubectl describe nodes | egrep -A 6 "Allocat"
```

```
Allocatable:
cpu: 13
ephemeral-storage: 4060864Ki
hugepages-1Gi: 0
hugepages-2Mi: 0
memory: 57315716Ki
pods: 110
--
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource Requests Limits
-------- -------- ------
cpu 12500m (96%) 182950m (1407%)
memory 29386Mi (52%) 365668Mi (653%)
ephemeral-storage 0 (0%) 0 (0%)
[rescue-user@MxNDsh01 ~]$
```

部署詳細資訊包括一條消息，其中包含有關Pod的當前條件的資訊：

```
[rescue-user@MxNDsh01 ~]$ kubectl get deployment -n cisco-mso consistencyservice -
o=jsonpath='{.status.conditions[*]}{"\n"}'
map[lastTransitionTime:2022-09-27T19:07:13Z lastUpdateTime:2022-09-27T19:07:13Z
message:Deployment has minimum availability. reason:MinimumReplicasAvailable status:True
type:Available] map[lastTransitionTime:2022-09-27T19:07:13Z lastUpdateTime:2022-09-27T19:07:13Z
message:ReplicaSet "consistencyservice-c98955599" has successfully progressed.
reason:NewReplicaSetAvailable status:True type:Progressing]
[rescue-user@MxNDsh01 ~]$
```

[擾流器]

如何在容器內部運行網路Debug命令

由於容器僅包含特定於Pod的最小庫和依賴項，因此大多數網路調試工具（ping、ip route和ip addr）在容器本身內不可用。

當需要對服務（在ND節點之間）或與Apic的連線進行網路故障排查時，這些命令非常有用，因為多個微服務需要通過資料介面(**bond0**或**bond0br**)與控制器進行通訊。

其 **nsenter** 實用程式（僅根使用者）允許從ND節點運行網路命令，因為它位於容器中。為此，請從要調試的容器中找到進程ID(PID)。這是使用Pod K8的ID根據容器運行時的本地資訊完成的，例如舊版的Docker，以及 **cri-o** 作為預設選項。

檢查Pod Kubernetes(K8s)ID

從cisco-mso名稱空間內的Pod清單中，我們可以選擇要進行故障排除的容器：

```
[root@MxNDsh01 ~]# kubectl get pod -n cisco-mso
NAME READY STATUS RESTARTS AGE
consistencyservice-569bdf5969-xkwpg 3/3 Running 0 9h
eeworker-65dc5dd849-485tq 2/2 Running 0 163m
endpointservice-5db6f57884-hkf5g 2/2 Running 0 9h
executionservice-6c4894d4f7-p8fzk 2/2 Running 0 9h
siteservice-64dfcdf658-lvbr4 3/3 Running 0 9h
siteupgrade-68bcf987cc-ttn7h 2/2 Running 0 9h
```

Pod必須在同一個K8節點上運行。對於生產環境，我們可以新增 **-o wide** 選項，查詢每個Pod運行的節點。使用Pod K8的ID（在前面的輸出示例中加粗了），我們可以檢查由容器運行時分配的進程(PID)。

如何從容器運行時檢查PID

對於Kubernetes，新的預設容器運行時是CRI-O。因此文檔在命令的規則之後。CRI-O分配的進程ID(PID)在K8s節點中可以是唯一的，可以使用 crictl 公用事業。

其 **ps** 選項顯示CRI-O為構建Pod的每個容器提供的ID，其中兩個用於站點服務示例：

```
[root@MxNDsh01 ~]# crictl ps |grep siteservice
fb560763b06f2 172.31.0.0:30012/cisco-
mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10
hours ago Running msc-siteservice2-ssl 0 074727b4e9f51
ad2d42aae1ad9 1d0195292f7fcc62f38529e135a1315c358067004a086cfed7e059986ce615b0 10 hours ago
Running siteservice-leader-election 0 074727b4e9f51
29b0b6d41d1e3 172.31.0.0:30012/cisco-
mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10 hours
ago Running siteservice 0 074727b4e9f51
[root@MxNDsh01 ~]#
```

有了此資訊，我們就可以使用 inspect CRIO-ID 選項可檢視賦予每個容器的實際PID。以下項需要此資訊： nsenter 指令:

```
[root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid
"pid": 239563,
"pids": {
"type": "pid"
```

如何使用nsenter在容器內運行網路Debug命令

使用上面輸出的PID，我們可以使用作為下一個命令語法中的目標：

```
nsenter --target <PID> --net <NETWORK COMMAND>
```

其 **--net** 選項允許我們在網路名稱空間中運行命令，因此可用的命令數量是有限的。

例如：

```
[root@MxNDsh01 ~]# nsenter --target 239563 --net ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450
inet 172.17.248.146 netmask 255.255.0.0 broadcast 0.0.0.0
inet6 fe80::984f:32ff:fe72:7bfb prefixlen 64 scopeid 0x20<link>
ether 9a:4f:32:72:7b:fb txqueuelen 0 (Ethernet)
RX packets 916346 bytes 271080553 (258.5 MiB)
RX errors 0 dropped 183 overruns 0 frame 0
TX packets 828016 bytes 307255950 (293.0 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 42289 bytes 14186082 (13.5 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 42289 bytes 14186082 (13.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

ping也可用，它測試從容器到外部的連線，而不只是測試K8的節點。

```
[root@MxNDsh01 ~]# nsenter --target 239563 --net wget --no-check-certificate
https://1xx.2xx.3xx.4xx
--2023-01-24 23:46:04-- https://1xx.2xx.3xx.4xx/
Connecting to 1xx.2xx.3xx.4xx:443... connected.
WARNING: cannot verify 1xx.2xx.3xx.4xx's certificate, issued by '/C=US/ST=CA/O=Cisco
System/CN=APIC':
Unable to locally verify the issuer's authority.
WARNING: certificate common name 'APIC' doesn't match requested host name '1xx.2xx.3xx.4xx'.
HTTP request sent, awaiting response... 200 OK
Length: 3251 (3.2K) [text/html]
Saving to: 'index.html'

100%[==========================================================================================
==========================================================>] 3,251 --.-K/s in 0s

2023-01-24 23:46:04 (548 MB/s) - 'index.html' saved [3251/3251]
```

如何從容器內部運行網路調試命令由於容器僅包含特定於Pod的最小庫和依賴項，因此大多數網路調試工具（ping、ip route和ip addr）在容器本身內不可用。 當需要對服務（在ND節點之間）或與Apic的連線進行網路故障排查時，這些命令非常有用，因為多個微服務需要通過資料介面（bond0或bond0br）與控制器進行通訊。 Nsenter實用程式（僅根使用者）允許從ND節點運行網路命令，因為它位於容器中。為此，請從要調試的容器中找到進程ID(PID)。這是使用Pod K8的ID根據容器運行時的本地資訊完成的，例如舊版的Docker，以及新版的cri-o（預設值）。 檢查Pod Kubernetes(K8s)ID從cisco-mso名稱空間內的Pod清單中，我們可以選擇要進行故障排除的容器：[root@MxNDsh01 ~]# kubectl get pod -n cisco-msoNAME READY STATUS RESTARTS AGEconsistencyservice-569bdf5969-xkwpg 3/3運行0 9heeworker-65dc5dd849-485tq 2/2運行0 163mendpointservice-5db6f57884-2 5g 2/2運行0 9hexecutionservice-6c4894d4f7-p8fzk 2/2運行0 9shteservice-64dfcdf658-lvbr4 3/3運行0 9shteupgrade-68bcf987cc-ttn7h 2/2運行0 9h電源箱必須在同一個K8s節點中運行。對於生產環境，我們可以在末尾新增 — o wide選項以查詢每個Pod運行的節點。使用Pod K8的ID（在前面的輸出示例中加粗了），我們可以檢查由容器運行時分配的進程(PID)。 如何從Container Runtime檢查PID對於Kubernetes，新的預設容器運行時是CRI-O。因此文檔在命令的規則之後。CRI-O分配的進程ID(PID)在K8s節點中可以是唯一的，可以使用crictl實用程式發現該節點。 ps選項顯示CRI-O為構建Pod的每個容器提供的ID，其中兩個用於站點服務示例：[root@MxNDsh01 ~]# crictl ps |grep siteservicefb560763b06f2 172.31.0.0:30012/cisco-mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10小時前Running msc-siteservice2-ssl 0074727b4e9f51ad2d42aae1ad9 1d0195292f7fcc62f38529e135a1315c358067004a086cfed7e059986ce615b0 10小時前Running siteservice-leader-election 0 074727b4e9f5129b0b6d 1d1e3 172.31.0.0:30012/cisco-mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10小時前Running siteservice 0 074727b4e9f51[root@MxNDsh01 ~]#使用此資訊，我們可以使用inspect CRIO-ID選項檢視為每個容器指定的實際PID。nsenter命令需要此資訊：[root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid"pid": 239563,"pids": {"type": "pid"如何使用nsenter在容器內運行網路調試命令通過上述輸出的PID，我們可以在下一個命令語法中用作目標： nsenter —target <PID> —net <NETWORK COMMAND> —net選項允許我們在網路名稱空間內運行命令，因此可用的命令數量有限。例如：[root@MxNDsh01 ~]# nsenter —target 239563 —net ifconfiguration0: flags=4163<UP，BROADCAST，RUNNING，MULTICAST> mtu 1450inet 172.17.248.146 netmask 255.255.0.0 broadcast 0.0.0.0inet6 fe80::984f:32ff:fe72:7bfb prefixlen 64 scopeid 0x20<link>ether:4a:4f 32:72:7b:fb txqueuelen 0(Ethernet)RX packets 916346 bytes 271080553(258.5 MiB)RX errors 0 dropped 183 overruns 0 frame 0TX packets 828016 bytes 307255950(293.0 MiB)TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0lo:flags=73<UP，LOOPBACK，RUNNING> mtu 65536inet 125.0.0inet6 ::1 prefixlen 12 scopeid 0x10<host>loop txqueuelen 1000（本地環回）RX packets 42289 bytes 14186082(13.5 MiB)RX errors 0 dropped 0 overruns 0 frame 0TX packets 42289 bytes 14186082(13.5 MiB)TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0也可用ping，它測試從容器到外部的連通性，而不是僅測試K8s節點。[root@MxNDsh01 ~]# nsenter —target 239563 —net wget —no-check-certificate https://1xx.2xx.3xx.4xx--2023-01-24 23:46:04— https://1xx.2xx.3xx.4xx/Connecting到

1xx.2xx.3xx.4xx:443... connected.警告：無法驗證1xx.2xx.3xx.4xx的證書（由「/C=US/ST=CA/O=Cisco System/CN=APIC」頒發）：無法本地驗證頒發者的許可權。警告：證書公用名稱「APIC」與請求的主機名「1xx.2xx.3xx」不匹配.HTTP請求已傳送，等待響應…… 200 OKLength: 3251(3.2K)[text/html]儲存到：'index.html'100%[==========>] 3,251 —.-K/s in 0s2023-01-24 23:46:04(548 MB/s)- 'index.html'已儲存[3251/3251]

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。