Risoluzione dei problemi e revisione delle risorse NDO

Sommario

Introduzione

In questo documento viene descritto come esaminare e risolvere i problemi relativi a NDO con la CLI di runtime di kubectl e contenitore.

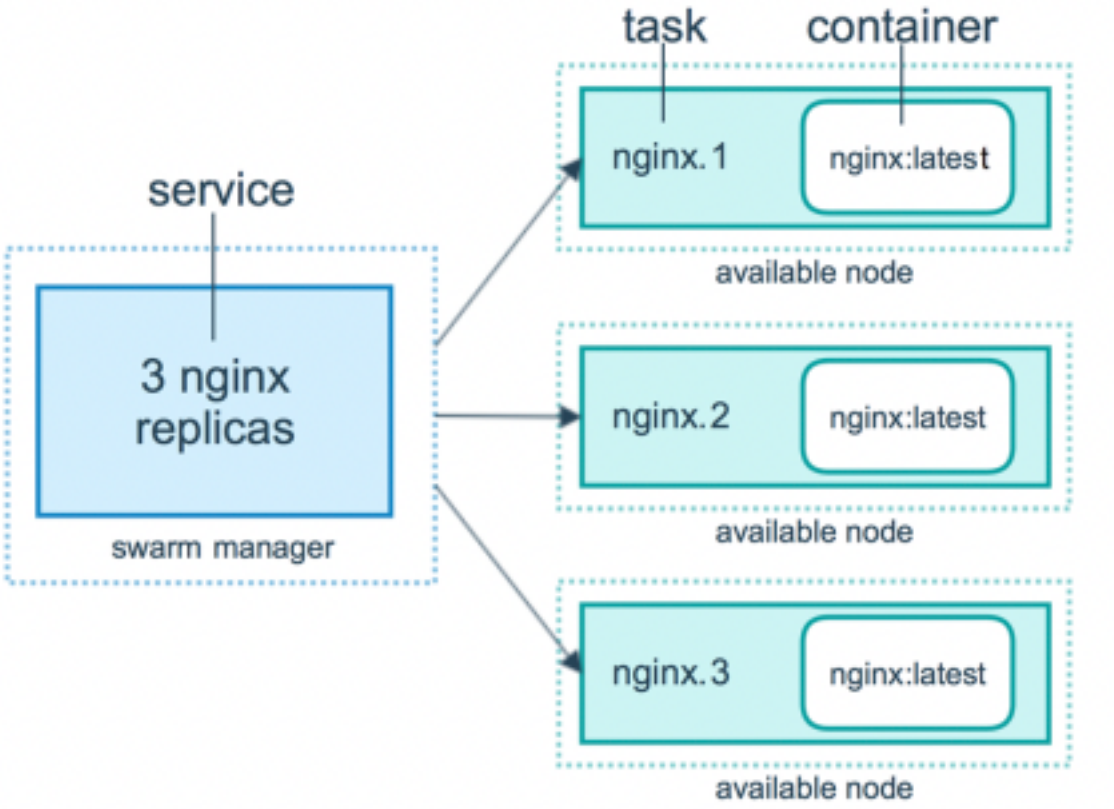
Avvio rapido NDO

Cisco Nexus Dashboard Orchestrator (NDO) è uno strumento di amministrazione del fabric, che consente agli utenti di gestire diversi tipi di fabric, tra cui i siti Cisco® Application Centric Infrastructure (Cisco ACI®), i siti Cisco Cloud ACI e i siti Cisco Nexus Dashboard Fabric Controller (NDFC), ciascuno gestito dal proprio controller (cluster APIC, cluster NDFC o istanze Cloud APIC in un cloud pubblico).

La tecnologia NDO garantisce l'organizzazione coerente della rete e delle regole, la scalabilità e il disaster recovery in più data center attraverso un'unica console.

Nei giorni precedenti, MSC (Multi-Site Controller) è stato implementato come cluster a tre nodi con VMWare Open Virtual Appliance (OVA) che ha consentito ai clienti di inizializzare un cluster Docker Swarm e i servizi MSC. Questo cluster Swarm gestisce i microservizi MSC come contenitori e servizi Docker.

Questa immagine mostra una visualizzazione semplificata di come Docker Swarm gestisce i microservizi come repliche dello stesso contenitore per ottenere un'elevata disponibilità.



Il Docker Swarm è stato responsabile del mantenimento del numero previsto di repliche per ciascuno dei microservizi dell'architettura MSC. Dal punto di vista di Docker Swarm, il controller multisito è stato l'unico tipo di distribuzione di contenitori per l'organizzazione.

Nexus Dashboard (ND) è una console di gestione centrale per più siti di centri dati e una piattaforma comune che ospita i servizi operativi dei centri dati Cisco, che includono Nexus Insight e MSC versione 3.3 e successive, e ha cambiato il nome in Nexus Dashboard Orchestrator (NDO).

Mentre la maggior parte dei microservizi che comprendono l'architettura MSC rimangono gli stessi, NDO viene implementato in un cluster Kubernetes (K8s) piuttosto che in un cluster Docker Swarm. Questo consente alla divisione di amministrazione di coordinare più applicazioni o implementazioni invece di una sola.

Kubernetes con NDO Crash-Course

Kubernetes è un sistema open-source per l'automazione dell'installazione, della scalabilità e della gestione delle applicazioni containerizzate. Come Docker, Kubernetes lavora con la tecnologia dei container, ma non è legato a Docker. Questo significa che Kubernetes supporta altre piattaforme container (Rkt, PodMan).

Una differenza chiave tra Swarm e Kubernetes è che quest'ultimo non funziona direttamente con i container, funziona invece con un concetto di gruppi di contenitori collocati, chiamati Pods.

I contenitori in un POD devono essere eseguiti nello stesso nodo. Un gruppo di dispositivi POD è denominato Distribuzione. Una distribuzione Kubernetes può descrivere un'intera applicazione.

Kubernetes consente inoltre agli utenti di garantire che una certa quantità di risorse sia disponibile per qualsiasi applicazione. A tale scopo, è possibile utilizzare i controller di replica per garantire che il numero di pod sia coerente con i manifesti dell'applicazione.

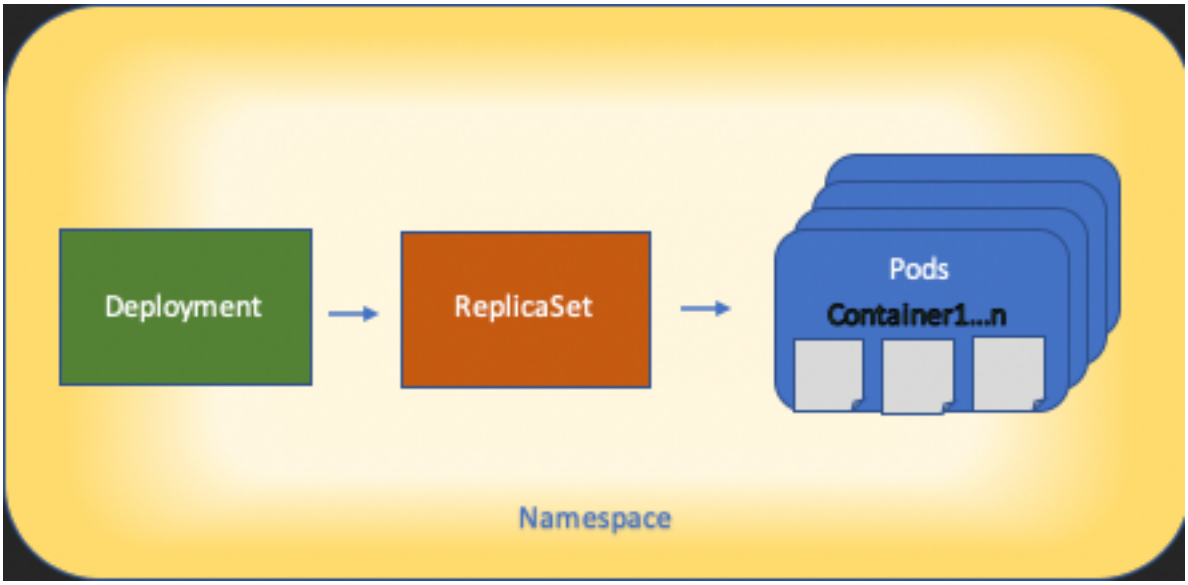
Un manifesto è un file in formato YAML che descrive una risorsa da distribuire tramite il cluster. La risorsa può essere una qualsiasi delle risorse descritte in precedenza o altre disponibili per gli utenti.

Èpossibile accedere all'applicazione esternamente con uno o più servizi. Kubernetes include un'opzione Load Balancer per eseguire questa operazione.

Kubernetes offre anche un modo per isolare diverse risorse con il concetto di Spazi dei nomi. La funzione ND utilizza gli spazi dei nomi per identificare in modo univoco applicazioni e servizi cluster diversi. Quando si eseguono i comandi CLI, specificare sempre Namespace.

Sebbene non sia richiesta una conoscenza approfondita di Kubernetes per la risoluzione dei problemi relativi a ND o NDO, è necessaria una conoscenza di base dell'architettura di Kubernetes per identificare correttamente le risorse con problemi o che richiedono attenzione.

Le nozioni di base dell'architettura delle risorse Kubernetes sono illustrate nel seguente diagramma:



Èimportante ricordare come ogni tipo di risorsa interagisce con gli altri e svolge un ruolo importante nel processo di revisione e risoluzione dei problemi.

Panoramica di NDO con i comandi Kubernetes

Accesso CLI

Per l'accesso CLI di SSH a NDO, **admin-user** password necessaria. Tuttavia, al suo posto utilizziamo **rescue-user** password. Come in:

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

Questa è la modalità e l'utente predefiniti per l'accesso CLI e la maggior parte delle informazioni è disponibile per la visualizzazione.

Revisione spazi dei nomi NDO

Questo concetto di K8s consente l'isolamento di diverse risorse all'interno del cluster. Il comando successivo può essere utilizzato per esaminare i diversi spazi dei nomi distribuiti:

```
[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
```

Le voci in grassetto appartengono ad Applicazioni nella NDO, mentre le entità che iniziano con il prefisso **kube** appartengono al cluster Kubernetes. Ogni spazio dei nomi dispone di distribuzioni e pod indipendenti

La CLI di kubectl consente di specificare uno spazio dei nomi con **--namespace** se un comando viene eseguito senza di esso, la CLI presuppone che lo spazio dei nomi sia **default** (Spazio dei nomi per k8s):

```
[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.
```

La CLI di kubectl consente diversi tipi di formati per l'output, ad esempio yaml, JSON o una tabella personalizzata. Ciò è possibile grazie alla **-o** opzione [format] (formato). Ad esempio:

```
[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": ""
   }
}
```


Dal testo precedente, l'output è un dizionario in cui una delle chiavi è denominata **elementi** e il valore è un **elenco** di dizionari in cui ogni **dizionario** tiene conto di una **voce Namespace** e i relativi attributi sono valori di coppia chiave-valore nel dizionario o nei dizionari nidificati.

Questo è importante perché K8s offre agli utenti la possibilità di selezionare jsonpath come output, consentendo operazioni complesse per un array di dati JSON. Ad esempio, dall'output precedente, se si accede al valore **name** per Spazi dei nomi, è necessario accedere all'elenco dei

valori degli elementi, quindi metadata e ottenere il valore della chiave **name**. A tale scopo, è possibile usare questo comando:

```
[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 ~]$
```

La gerarchia descritta viene utilizzata per recuperare le informazioni specifiche richieste. In pratica, è possibile accedere a tutti gli elementi **items** elenca con **elementi[*]**, quindi il tasto metadata e **name** con **metadata.name**, la query può includere altri valori da visualizzare.

Lo stesso vale per l'opzione delle colonne personalizzate, che utilizzano un metodo simile per recuperare le informazioni dall'array di dati. Ad esempio, se si crea una tabella con le informazioni **name** e UID , è possibile applicare il comando:

```
[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
```

L'output richiede un nome per ogni colonna da visualizzare e quindi assegnare il valore per l'output. In questo esempio sono presenti due colonne: **NAME** e **UID**. Questi valori appartengono a .metada.name e .metadata.uid rispettivamente. Ulteriori informazioni ed esempi sono disponibili all'indirizzo:

Supporto JSONPath

Colonne personalizzate

Revisione distribuzione NDO

Una distribuzione è un oggetto K8s che fornisce uno spazio unito per la gestione di ReplicaSet e Pod. Le distribuzioni gestiscono l'implementazione di tutti i Pod appartenenti a un'applicazione e il numero previsto di copie di ciascuno di essi.

La CLI di kubectl include un comando per controllare le distribuzioni per un determinato spazio dei nomi:

```
[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
```

Èpossibile utilizzare la stessa tabella personalizzata con deployment anziché **namespace** e **-n** per visualizzare le stesse informazioni di prima. Questo perché l'output è strutturato in modo simile.

```
[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
```

Tenete presente che il numero di copie visualizzate è per la distribuzione, non il numero di Pod per ogni microservizio.

Possiamo usare la parola chiave **describe** anziché **get** per visualizzare informazioni più dettagliate su una risorsa, in questo caso la distribuzione schemaservice:

```
[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 ~]$
```

OSPF (Open Shortest Path First) **describe** consente inoltre l'inclusione di **--show-events=true** opzione per mostrare gli eventi rilevanti per la distribuzione.

Spoiler

Revisione set di repliche NDO (RS)

Spoiler

DISPONIBILE SOLO PER L'UTENTE ROOT

Un set di repliche (RS) è un oggetto K8s con l'obiettivo di mantenere un numero stabile di repliche pod. Questo oggetto rileva anche quando viene rilevato un numero non integro di repliche con una sonda periodica per i pod.

I gruppi di risorse sono inoltre organizzati in spazi dei nomi.

```
[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
```

OSPF (Open Shortest Path First) describe Questa opzione include le informazioni sull'URL, la porta utilizzata dal probe e la periodicità dei test e la soglia di errore.

```
[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 Replica Set (RS) Revisione #### DISPONIBILE SOLO PER L'UTENTE ROOT ##### Un Replica Set (RS) è un oggetto K8s con l'obiettivo di mantenere un numero stabile di repliche Pod. Questo oggetto rileva anche quando viene rilevato un numero non integro di repliche con una sonda periodica per i pod. I gruppi di risorse sono inoltre organizzati in spazi dei nomi. [root@MxNDsh01 ~]# kubectl get rs -n cisco-msoNAME DESIRED CURRENT READY AGEauditservice-648cd4c6f8 1 1 3d22hbackupservice-64b755b44c 1 1 3d22hcloudsecservice-7df465576 1 1 3d22hconsistencyservice-c9895599 1 1 1 3d22hdcnmworker-worker d4d5cbb64 1 1 1 3d22heeworker-56f9fb9ddb 1 1 3d22hendpointservice-7df9d599c 1 1 3d22hexecutionservice-58ff89595f 1 1 3d22hfluentd-86785f89bd 1 1 3d22himportservice-88bcc8547 1 1 3d22hjobschedulerservice-5d4fdfd696 1 1 3d22hnotifyservice-75c988cfd4 1 1 3d22hpctagvnidservice-644b75596 1 1 3d22hplatformservice-65cdb946f 1 1 1 3d22hplatform service2-6796576 659 1 1 1 3d22hpolicyservice-545b9c7d9c 1 1 3d22hschemaservice-7597ff4c5 1 1 3d22hsdaservice-5f477dd8c7 1 1 3d22hsdwanservice-6f87cd999d 1 1 3d22hsiteservice-86bb75658 1 1 1 3d22hsiteupgrade-7d578f9b6d 1 1 1 3d22hsyncengine-5b8bdd6b45 1 1 3d22htemplateeng-5cbf9fdc48 1 1 3d22hui-84588b7c96 1 1 3d22huserservice-87846f7c6 1 1 3d22h L'opzione include le informazioni di descrizione informazioni sull'URL, la porta utilizzata dal probe e la periodicità dei test e la soglia di errore. [root@MxNDsh01 ~]# kubectl descrive rs -n cisco-mso schemaservice-7597ff4c5Nome: schemaservice-7597ff4c5Spazio dei nomi: cisco-msoSelector: k8s-app=schemaservice,pod-template-hash=7597ff4c5Etichette: cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service pod-template-hash=7597ff4c5Annotazioni: deployment.kubernetes.io/desired-replicas: 1 deployment.kubernetes.io/max-replicas: 1Controllato da:/schemaserviceReplicas: 1 corrente / 1

desiredPods Status: 1 In esecuzione / 0 In attesa / 0 Completato / 0 FailedPod Template: Etichette: deployment.kubernetes.io/revision k8s-app=schemaservice cpu.resource.case.cncf.io/schemaservice=cpu-lg-service pod-template-hash=7597ff4c5 Account del servizio: cisco-mso-sa Init Containers: init-msc: Immagine: cisco-mso/tools:3.7.1j Porta: <nessuno> Porta host: <nessuno> Comando: memory.resource.case.cncf.io/schemaservice=mem-xlg-service Environment: <nessuno> Mounts: /secrets from infracerts (schrw) Containers: emaservice: Immagine: cisco-mso/schemaservice:3.7.1j Porte: 8080/TCP, 8080/UDP Porte host: 0/TCP, 0/UDP Comando: /check_mongo.sh schemaservice Liveness: http-get /launchscala.sh delay=300s timeout=20s period=30s #success=1 #failure=3 Ambiente: JAVA_OPTS: -XX:+IdleTuningGcOnIdle Mounts: /jwtsecrets from jwtsecrets (rw) from logs (rw) /secrets from infracerts (rw) msc-schemaservice-ssl: Immagine: cisco-mso/sslcontainer:3.7.1j Porte: 443/UDP, 443/TCP Porte host: 0/UDP, 0/TCP Comando: http://:8080/api/v1/schemas/health /wrapper.sh

Revisione NDO Pod

Un POD è un gruppo di contenitori strettamente correlati che vengono eseguiti nello stesso spazio dei nomi Linux (diverso dallo spazio dei nomi K8s) e nello stesso nodo K8s. Questo è l'oggetto più atomico degli handle K8s, in quanto non interagisce con i contenitori. L'applicazione può essere costituita da un singolo contenitore o essere più complessa con molti contenitori. Con il comando successivo, è possibile controllare i pod di qualsiasi spazio dei nomi:

```
[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 ~]$
```

Il numero visualizzato nella seconda colonna si riferisce al numero di contenitori per ciascun POD.

OSPF (Open Shortest Path First) **describe** è disponibile anche l'opzione, che include informazioni dettagliate sui contenitori su ciascun POD.

```
[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
```

Le informazioni visualizzate includono l'immagine del contenitore per ogni contenitore e mostrano il runtime del contenitore utilizzato. In questo caso, CRI-O (**cri-o**), versioni precedenti di ND utilizzato per lavorare con Docker, questo influenza come collegare a un contenitore.

Spoiler

Ad esempio, quando **cri-o** e si desidera connettersi a un contenitore tramite una sessione interattiva (tramite exec -it al contenitore dall'output precedente, ma anziché docker , viene utilizzato il comando **crictl**:

```
schemaservice:

   Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac

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

Viene utilizzato questo comando:

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

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

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

root
```

Nelle versioni più recenti di ND, l'ID contenitore da utilizzare è diverso. Per prima cosa, è necessario utilizzare il comando **crictl ps** per elencare tutti i contenitori in esecuzione in ogni nodo. È possibile filtrare il risultato in base alle esigenze.

```
[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 ~]#
```

Con il valore della prima colonna, è quindi possibile accedere al runtime Container con lo stesso comando utilizzato in precedenza:

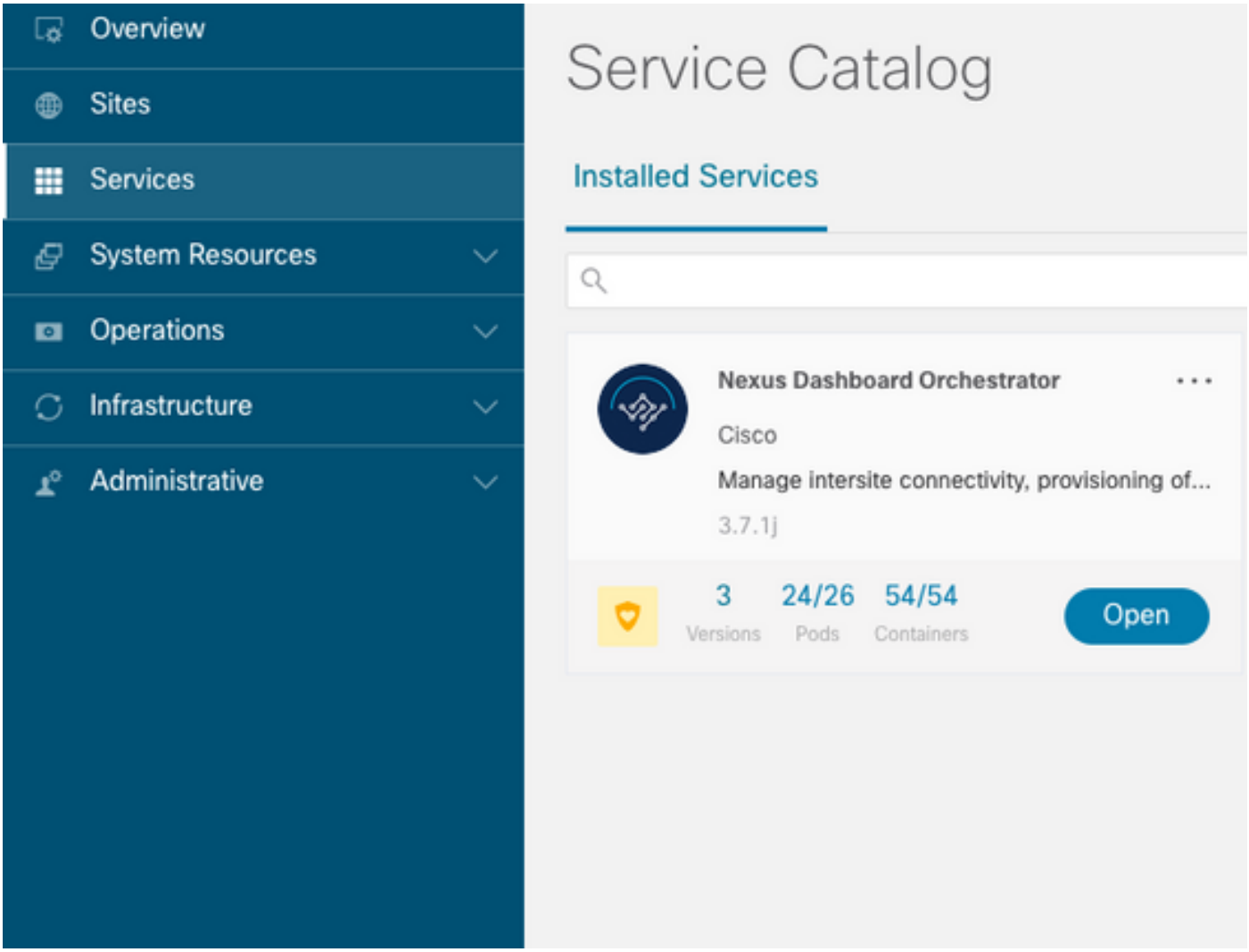
```
[root@singleNode ~]# crictl exec -it 4b26f67fc10cf bash
root@backupservice-8c699779f-j9jtr:/# pwd
/
```

Ad esempio, quando si utilizza cri-o e si desidera stabilire una connessione al contenitore mediante una sessione interattiva (tramite l'opzione exec -it) dall'output precedente, ma anziché il comando docker, viene utilizzato il comando criclt: schemaservice: ID contenitore: cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac Immagine: cisco-mso/schemaservice:3.7.1j Viene utilizzato questo comando: [root@MxNDsh01 ~]# crictl exec -it d2287f8659dec6848c0100b7d 24aeebd506f3f77af660238ca0c9c7e8946f4ac bashroot@schemaservice-7597ff4c5-w4x5d:/#root@schemaservice-7597ff4c5-w4x5d:/# whoamiroot Nelle versioni più recenti di ND, l'ID contenitore da utilizzare è diverso. Innanzitutto, è necessario utilizzare il comando crictl ps per elencare tutti i contenitori in esecuzione su ogni nodo. È possibile filtrare il risultato in base alle esigenze. [root@singleNode ~]# crictl ps| grep backupa9bb161d67295 10.31.125.241:30012/cisco-mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 2 giorni fa che eseguono msc-backupservice-ssl 0 84b3c691cfc2b4b26f67fc10cf 10.31.125.241:30012/cisco-mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2 giorni fa Esecuzione backupservice 0 84b3c691cfc2b[root@singleNode ~]# Con il valore della prima colonna, è possibile accedere al runtime del contenitore con lo stesso comando precedente: [root@singleNode ~]# crictl exec -it 4b26f67fc10cf bashroot@backupservice-8c699779f-j9jtr:/# pwd/

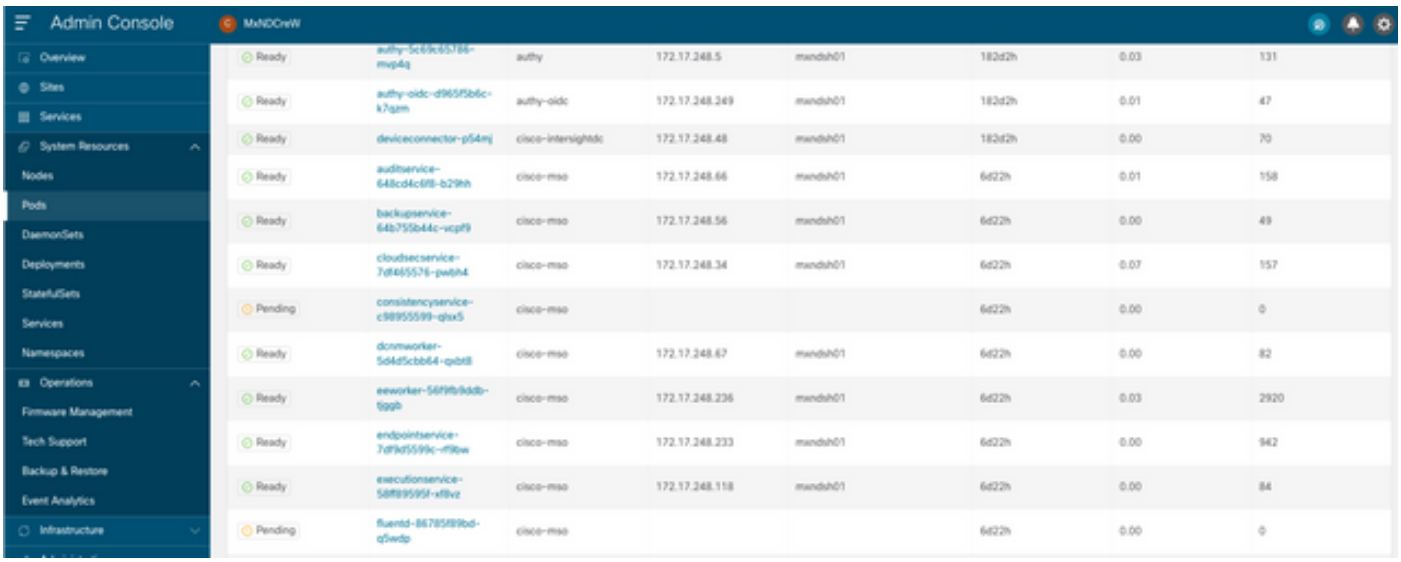
Il POD con Use Case non è integro

Èpossibile utilizzare queste informazioni per risolvere i problemi relativi all'integrità dei Pod di una distribuzione. Per questo esempio, la versione di Nexus Dashboard è 2.2-1d e l'applicazione interessata è Nexus Dashboard Orchestrator (NDO).

L'interfaccia utente grafica di NDO mostra un set incompleto di Pod dalla visualizzazione Servizio. In questo caso 24 su 26 Pod.



Un'altra visualizzazione disponibile nel **System Resources -> Pods** visualizzazione in cui i Pod mostrano uno stato diverso da **Ready**.



Risoluzione dei problemi della CLI per i Pod non integri

Con il fatto noto che lo spazio dei nomi è cisco-mso (anche se in caso di risoluzione dei problemi è lo stesso per altre app/spazi dei nomi) la visualizzazione del Pod mostra eventuali app non integre:

```
[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
```

Per questo esempio, è possibile concentrarsi sui pacchetti consistencyservice. Dall'output JSON è possibile ottenere informazioni specifiche dai campi di stato, utilizzando jsonpath:

```
[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 ~]$
```

Il dizionario di **stato** e all'interno di un elenco denominato **condizioni** con dizionari come elementi con le chiavi **messaggio** e **valore**, la parte {"\n"} consiste nel creare una nuova riga alla fine:

```
[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 ~]$
```

Questo comando mostra come eseguire il controllo **get Pod** per Namespace:

```
[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
```

Con il get pods , è possibile ottenere l'ID del Pod con problemi che devono corrispondere a quello dell'output precedente. In questo esempio **consistencyservice-c98955599-qlsx5**.

Il formato di output JSON fornisce inoltre la modalità di controllo di informazioni specifiche dall'output specificato.

```
[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 ~]$
```

L'output JSON deve includere informazioni sullo stato nell'attributo con lo stesso nome. Il messaggio contiene informazioni sul motivo.

```
[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 ~]$
```

Possiamo accedere alle informazioni sullo stato e sui requisiti per i Pod:

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

In questo caso è importante menzionare il modo in cui viene calcolato il valore. Nell'esempio, la CPU da **500 m** si riferisce a **500 milicore**, mentre la **memoria da 1G** è per GB.

OSPF (Open Shortest Path First) **Describe** per il nodo mostra la risorsa disponibile per ogni fase K8 nel cluster (host o VM):

```
[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 ~]$
```

La sezione **Allocabile** mostra le risorse totali in CPU, memoria e spazio di archiviazione disponibili per ogni nodo. La sezione **Allocate** mostra le risorse già in uso. Il valore **13** per CPU si riferisce a **13 core** o **13.000 (13K) millicores**.

In questo esempio, il nodo è **sovrascritto**, il che spiega perché il Pod non può essere avviato. Dopo aver cancellato la DN con l'eliminazione delle APP di ND o l'aggiunta di risorse VM.

Il cluster tenta costantemente di distribuire qualsiasi criterio in sospeso, quindi se le risorse sono libere, è possibile distribuire i 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
```

Con il comando utilizzato per il controllo delle risorse, si conferma che il cluster dispone di risorse per la CPU:

```
[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 ~]$
```

I dettagli di distribuzione includono un messaggio con informazioni sulle condizioni correnti per i dispositivi di distribuzione:

```
[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 ~]$
```

Spoiler

Come eseguire i comandi di debug della rete da un contenitore

Poiché i contenitori includono solo le librerie minime e le dipendenze specifiche per il POD, la maggior parte degli strumenti di debug di rete (ping, ip route e ip addr) non sono disponibili all'interno del contenitore stesso.

Questi comandi sono molto utili quando è necessario risolvere problemi di rete per un servizio (tra nodi ND) o una connessione verso le appliance, in quanto diversi microservizi devono comunicare con i controller tramite l'interfaccia dati (**bond0** o **bond0br**).

OSPF (Open Shortest Path First) **nsenter** L'utilità (solo per utenti root) consente di eseguire i comandi di rete dal nodo ND così come si trova all'interno del contenitore. A tale scopo, individuare l'ID processo (PID) dal contenitore di cui si desidera eseguire il debug. Ciò è possibile grazie all'ID Pod K8s confrontato con le informazioni locali del Container Runtime, come Docker per le versioni precedenti, e **cri-o** per quelli più recenti come impostazione predefinita.

Ispezionare l'ID Pod Kubernetes (K8s)

Dall'elenco di Pod all'interno dello spazio dei nomi cisco-mso, è possibile selezionare il contenitore per la risoluzione dei problemi:

```
[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
```

I pod devono essere eseguiti nello stesso nodo K8s. Per gli ambienti di produzione, è possibile aggiungere **-o wide** alla fine per scoprire il nodo su cui è in esecuzione ciascun Pod. Con l'ID POD K8s (in grassetto nell'esempio di output precedente) è possibile controllare il processo (PID) assegnato dal runtime del contenitore.

Come ispezionare il PID dal runtime del contenitore

Il nuovo Runtime contenitore predefinito è CRI-O per Kubernetes. Il documento segue la regola per i comandi. L'ID processo (PID) assegnato da CRI-O può essere univoco nel nodo K8s, che può essere rilevato con crictl utilità.

OSPF (Open Shortest Path First) **ps** Questa opzione consente di visualizzare l'ID assegnato da CRI-O a ciascun contenitore che crea il Pod, due per l'esempio di dispositivo di sito:

```
[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 ~]#
```

Grazie a queste informazioni, è possibile utilizzare inspect CRIO-ID per visualizzare il PID effettivo assegnato a ciascun contenitore. Queste informazioni sono necessarie per nsenter comando:

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

Come utilizzare l'invio per eseguire i comandi di debug della rete all'interno di un contenitore

Con il PID dell'output precedente, è possibile utilizzare come destinazione nella sintassi del comando seguente:

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

OSPF (Open Shortest Path First) **--net** consente di eseguire comandi negli spazi dei nomi di rete, pertanto il numero di comandi disponibili è limitato.

Ad esempio:

```
[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
```

Il comando ping è disponibile anche e verifica la connettività dal contenitore all'esterno, anziché solo dal nodo K8s.

```
[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]
```

Come eseguire i comandi di debug di rete dall'interno di un contenitore Poiché i contenitori includono solo le librerie minime e le dipendenze specifiche del POD, la maggior parte degli strumenti di debug di rete (ping, ip route e ip addr) non è disponibile all'interno del contenitore stesso. Questi comandi sono molto utili quando è necessario risolvere problemi di rete per un servizio (tra nodi ND) o una connessione verso le appliance, in quanto diversi microservizi devono comunicare con i controller tramite l'interfaccia dati (bond0 o bond0br). L'utilità nsenter (solo utente root) consente di eseguire i comandi di rete dal nodo ND così come si trova all'interno del contenitore. A tale scopo, individuare l'ID processo (PID) dal contenitore di cui si desidera eseguire il debug. Ciò è possibile grazie all'ID Pod K8s confrontato con le informazioni locali del Container Runtime, come Docker per le versioni precedenti e cri-o per quelle più recenti come impostazione predefinita. Ispezionare l'ID Pod Kubernetes (K8s) Dall'elenco di Pod all'interno dello spazio dei nomi cisco-mso, è possibile selezionare il contenitore da risolvere: [root@MxNDsh01 ~]# kubectl get pod -n cisco-msoNAME READY STATUS RESTARTS AGEconsistencyservice-569bdf5969-xkwpg 3/3 Running 0 9heeworker-65dc5dd849-485tq 2/2 Running 0 163mendpointservice-5db6f5 7884-hkf5g 2/2 Esecuzione 0 9hexecutionservice-6c4894d4f7-p8fzk 2/2 Esecuzione 0 9hsiteservice-64dfcdf658-lvbr4 3/3 Esecuzione 0 9hsiteupgrade-68bcf987cc-ttn7h 2/2 Esecuzione 0 9h I Pod devono essere eseguiti nello stesso nodo K8s. Per gli ambienti di produzione, possiamo aggiungere l'opzione -o wide alla fine per scoprire il nodo che ciascun Pod esegue. Con l'ID POD K8s (in grassetto nell'esempio di output precedente) è possibile controllare il processo (PID) assegnato dal runtime del contenitore. Ispezione del PID dal runtime del contenitore Il nuovo runtime predefinito del contenitore è CRI-O per Kubernetes. Il documento segue la regola per i comandi. L'ID processo (PID) assegnato da CRI-O può essere univoco nel nodo K8s, che può essere rilevato con l'utilità crictl. L'opzione ps indica l'ID assegnato da CRI-O a ciascun contenitore che costruisce il Pod, due per l'esempio di siteservice: [root@MxNDsh01 ~]# crictl ps |grep siteservicefb560763b06f2

172.31.0.0:30012/cisco-mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10 ore fa Esecuzione di msc-siteservice2-ssl 0 074727b4e9f51ad2d42aae1ad91d0195292f7fcc62f38529e135a1315c35806 7004a086cfed7e059986ce615b0 10 ore fa Esecuzione siteservice-leader-election 0 074727b4e9f5129b0b6d41d1e3 172.31.0.0:30012/cisco-mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10 ore fa Esecuzione siteservice 0 074727b4e9f51[root@MxNDsh01 ~]# Con queste informazioni, possiamo quindi utilizzare il software inspect cric opzione-ID per visualizzare il PID effettivo assegnato a ogni contenitore. Queste informazioni sono necessarie per il comando nsenter: [root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid"pid": 239563,"pids": {"type": "pid" Come utilizzare nsenter per eseguire i comandi di debug di rete all'interno di un contenitore Con il PID dell'output precedente, è possibile utilizzare come destinazione nella sintassi del comando seguente: nsenter —target <PID> —net <NETWORK COMMAND> L'opzione —net consente di eseguire i comandi nella rete Namesper , per cui il numero di comandi disponibili è limitato. Ad esempio: [root@MxNDsh01 ~]# nsenter —target 239563 —net ifconfigeth0: 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>etere 9a:4f:32:72:7b:fb txqueuelen 0 (Ethernet)Pacchetti RX 916346 byte 271080553 (258,5 MiB)Errori RX 0 ignorati 183 sovraccarichi 0 frame 0TX pacchetti 828016 byte 307255950 (293,0 MiB)TX ignorato 0 sovraccarichi 0 portante 0 collisioni 0lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536inet 127.0.0.1 netmask 255.0.0.0inet6 ::1 prefix 128 scopeid 0x10<host>loop txqueuelen 1000 (loopback locale)Pacchetti RX 42289 byte 14186082 (13,5B)X errori 0 ignorati 0 sovraccarichi 0 frame 0TX pacchetti 42289 byte 14186082 (13.5 MiB)Errori TX 0 ignorati 0 sovraccarichi 0 portante 0 collisioni 0 Il ping è disponibile anche e verifica la connettività dal contenitore all'esterno, anziché solo dal nodo K8s. [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: impossibile verificare il certificato di 1xx.2xx.3xx.4xx, rilasciato da ‘/C=US/ST=CA/O=Cisco System/CN=APIC’:Impossibile verificare localmente l'autorità dell'emittente.WARNING: il nome comune del certificato ‘APIC’ non corrisponde al nome host richiesto ‘xx.2xx.3xx.4xx".Richiesta HTTP inviata, in attesa di risposta... 200 OKLength: 3251 (3.2K) [testo/html]Salvataggio in: 'index.html'100%[==========>] 3.251 —.-K/s in 0s2023-01-24 23:46:04 (548 MB/s) - "index.html" salvato [3251/3251]

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).