Probleemoplossing en evaluatie van NDO-bronnen

Inhoud

[Inleiding](#)
[NDO QuickStart](#)
[Kubernetes met NDO-crashprogramma](#)
[NDO - Overzicht met Kubernetes-opdrachten](#)
[Aanmelden voor CLI-toegang](#)
[NDO-namespacesbeoordeling](#)
[NDO-implementatiebeoordeling](#)
[NDO Replica Set (RS) - beoordeling](#)
[NDO Pod Review](#)
[Gebruik-case Pod is niet gezond](#)
[CLI-probleemoplossing voor ongezonde pods](#)
[Hoe te om het Netwerk te leiden zuiver Opdrachten van binnen een Container](#)
[Inspecteer de Pod Kubernetes (K8s) ID](#)
[Hoe de PID te inspecteren vanuit de Container Runtime](#)
[Hoe te gebruiken nsenter om netwerk debug commands in een container uit te voeren](#)

Inleiding

Dit document beschrijft hoe NDO met de kubectl en containerruntime CLI te bekijken en problemen op te lossen.

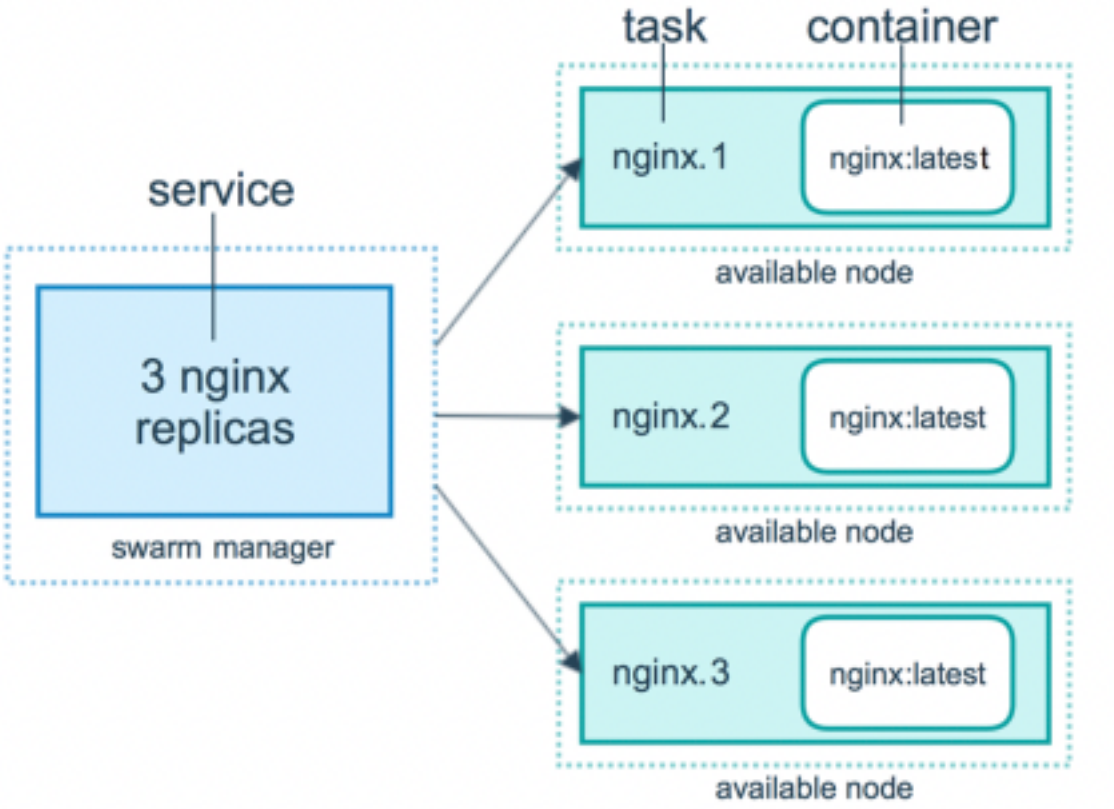
NDO QuickStart

De Cisco Nexus Dashboard Orchestrator (NDO) is een tool voor fabricbeheer, waarmee gebruikers verschillende soorten stoffen kunnen beheren, waaronder Cisco® Application Centric Infrastructure (Cisco ACI®)-sites, Cisco Cloud ACI-sites en Cisco Nexus Dashboard Fabric Controller (NDFC)-sites, waarbij elk wordt beheerd door zijn eigen controller (APIC-cluster, NDFC-cluster of Cloud APIC-instanties in een openbare cloud).

NDO biedt consistente netwerk- en beleidsorkestratie, schaalbaarheid en noodherstel via meerdere datacenters via één venster.

In de eerdere dagen werd de MSC (Multi-Site Controller) ingezet als een drie-knooppunt cluster met VMWare Open Virtual Appliances (OVA's) die klanten in staat stelde om een Docker Swarm cluster en de MSC-services te initialiseren. Dit Swarm cluster beheert de MSC microservices als Docker containers en diensten.

Dit beeld toont een vereenvoudigde weergave van hoe de Docker Swarm de microdiensten beheert als replica's van dezelfde container om een hoge beschikbaarheid te bereiken.



De Docker Swarm was verantwoordelijk voor het behoud van het verwachte aantal replica's voor elk van de microdiensten in de MSC-architectuur. Vanuit het oogpunt van Docker Swarm was de Multi-Site Controller de enige containerinzet om te orkestreren.

Nexus Dashboard (ND) is een centrale beheerconsole voor meerdere datacentersites en een gemeenschappelijk platform dat Cisco-datacenterbewerkingen host, waaronder Nexus Insight en MSC versie 3.3 en de naam heeft gewijzigd in Nexus Dashboard Orchestrator (NDO).

Terwijl de meeste microservices die de MSC architectuur omvatten hetzelfde blijven, wordt NDO ingezet in een Kubernetes (K8s) cluster in plaats van in een Docker Swarm. Hierdoor kan ND meerdere toepassingen of implementaties orkestreren in plaats van slechts één.

Kubernetes met NDO-crashprogramma

Kubernetes is een opensource-systeem voor automatische implementatie, schaalbaarheid en beheer van containertoepassingen. Kubernetes werkt als Docker met de containertechnologie, maar is niet verbonden met Docker. Dit betekent dat Kubernetes andere containerplatforms ondersteunt (Rkt, PodMan).

Een belangrijk verschil tussen Swarm en Kubernetes is dat Kubernetes niet direct met containers werkt, maar met een concept van op elkaar geplaatste containers, Pods genaamd.

De containers in een Pod moeten in dezelfde knoop lopen. Een groep Pods wordt een Implementatie genoemd. Een Kubernetes-implementatie kan een hele toepassing beschrijven.

Kubernetes stelt de gebruikers ook in staat om ervoor te zorgen dat een bepaalde hoeveelheid resources beschikbaar zijn voor een bepaalde toepassing. Dit gebeurt met het gebruik van replicatie-controllers om er zeker van te zijn dat het aantal pods consistent is met de toepassingsmanifest.

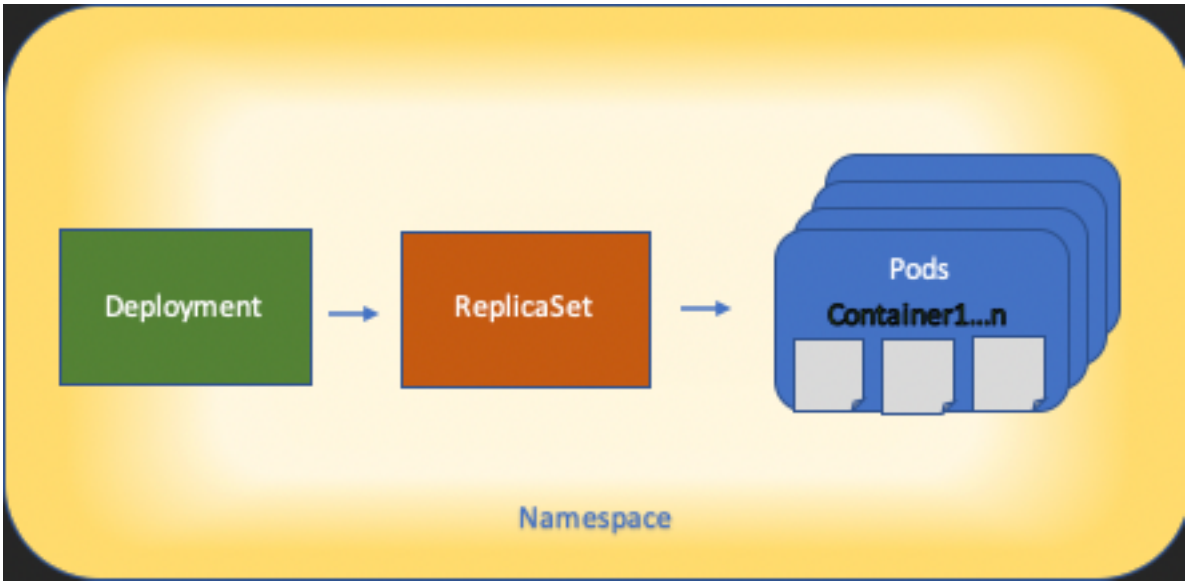
Een Manifest is een YAML-geformatteerd bestand dat een resource beschrijft die door de Cluster moet worden geïmplementeerd. De bron kan een van de eerder beschreven of andere bronnen beschikbaar voor gebruikers zijn.

De applicatie kan extern worden benaderd met een of meer services. Kubernetes bevat een taakverdeling om dit te realiseren.

Kubernetes biedt ook een manier om verschillende hulpbronnen te isoleren met het concept van Namespaces. In de ND wordt Namespaces gebruikt om verschillende toepassingen en clusterservices op unieke wijze te identificeren. Wanneer CLI-opdrachten worden uitgevoerd, specificeert u altijd de Namespace.

Hoewel een grondige kennis van Kubernetes niet nodig is om problemen met ND of NDO op te lossen, is een fundamenteel begrip van de Kubernetes architectuur vereist om de bronnen met problemen of die aandacht nodig hebben goed te identificeren.

De grondbeginselen van de Kubernetes resource architectuur worden getoond in dit diagram:



Het is belangrijk om te onthouden hoe elk type resource met de anderen communiceert, en het speelt een belangrijke rol in het review- en probleemoplossingsproces.

NDO - Overzicht met Kubernetes-opdrachten

Aanmelden voor CLI-toegang

Voor CLI-toegang van SSH tot NDO wordt de **admin-user** wachtwoord vereist is. In plaats daarvan gebruiken we echter de **rescue-user** wachtwoord. Zoals 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 ~]$
```

Dit is de standaardmodus en -gebruiker voor CLI-toegang en de meeste informatie is beschikbaar om te zien.

NDO-namespacesbeoordeling

Dit K8s concept maakt het mogelijk om verschillende bronnen over de cluster te isoleren. De volgende opdracht kan worden gebruikt om de verschillende geïmplementeerde Namespaces te bekijken:

```
[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
```

De vetgedrukte vermeldingen behoren tot Toepassingen in de NDO, terwijl de entiteiten die beginnen met het prefix **kube** behoren tot het Kubernetes-cluster. Elke Namespace heeft zijn eigen onafhankelijke implementaties en Pods

De kubectl CLI staat toe om een naamruimte te specificeren met de **--namespace** optie, als een opdracht zonder deze wordt uitgevoerd, gaat de CLI ervan uit dat de Namespace **default** (Naamruimte voor 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.
```

De kubectl CLI staat verschillende soorten formaten toe voor de output, zoals yaml, JSON, of een op maat gemaakte tabel. Dit wordt bereikt met de **-o** [Format] optie. Voorbeeld:

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

Van de vorige tekst, is de output een woordenboek waar één van zijn sleutels **punten** wordt genoemd en de waarde is een **lijst** van woordenboeken waar elk **woordenboek** voor een **ingang** **Namespace** rekenschap geeft en zijn eigenschappen zeer belangrijk-waarde paarwaarde in het woordenboek of genestelde woordenboeken zijn.

Dit is relevant omdat K8s gebruikers de optie biedt om jsonpath als uitvoer te selecteren, dit maakt

complexe bewerkingen mogelijk voor een JSON data array. Bijvoorbeeld, van de vorige output, als wij tot de waarde van toegang hebben **name** voor Namespaces, moeten wij tot de waarde van puntenlijst, toen toegang hebben metadata woordenboek, en krijg de waarde van de toets **name**. Dit kan met deze opdracht worden gedaan:

```
[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 ~]$
```

De beschreven hiërarchie wordt gebruikt om de vereiste specifieke informatie op te halen. In principe zijn alle items toegankelijk in de **items** lijst met **items[*]**, dan de toets metadata en **name** met **metadata.name**, kan de query andere weer te geven waarden bevatten.

Hetzelfde geldt voor de optie van aangepaste kolommen, die een soortgelijke manier gebruiken om de informatie uit de dataserie op te halen. Bijvoorbeeld, als we een tabel maken met de informatie over de **name** en de UID waarden, kunnen we de opdracht toepassen:

```
[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
```

De uitvoer vereist een naam voor elke kolom om te tonen en dan de waarde voor de output toe te wijzen. In dit voorbeeld zijn er twee kolommen: **NAME** en **UID**. Deze waarden behoren tot .metada.name en .metadata.uid respectievelijk. Meer informatie en voorbeelden zijn te vinden op:

Ondersteuning van JSONPath

Aangepaste kolommen

NDO-implementatiebeoordeling

Een inzet is een K8s object dat een samengevoegde ruimte biedt om ReplicaSet en Pods te beheren. De implementaties behandelen de uitrol van alle Peul die tot een Toepassing behoren en het verwachte aantal exemplaren van elke één.

De kubectl CLI bevat een opdracht om de implementaties voor een bepaalde Namespace te controleren:

```
[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
```

We kunnen dezelfde douanetabel gebruiken met het gebruik van deployment in plaats **namespace** en de **-n** optie om dezelfde informatie te zien als voorheen. Dit komt doordat de output op een zelfde manier gestructureerd is.

```
[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
```

Houd in gedachten het aantal weergegeven kopieën is voor de implementatie, niet het aantal

Pods voor elke microservice.

We kunnen het trefwoord gebruiken **describe** in plaats **get** om meer gedetailleerde informatie over een middel, in dit geval de schemaservice plaatsing te tonen:

```
[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 ~]$
```

Het **describe** opdracht maakt ook de opname van de **--show-events=true** de optie om alle relevante gebeurtenissen voor de inzet te tonen.

Spoiler

NDO Replica Set (RS) - beoordeling

Spoiler

DIT IS ALLEEN BESCHIKBAAR VOOR HOOFDGEBRUIKER

Een Replica Set (RS) is een K8s object met als doel een stabiel aantal replica Pods te behouden. Dit object detecteert ook wanneer er een ongezond aantal replica's worden gezien met een periodieke sonde naar de Pods.

De RS zijn ook in naamruimten georganiseerd.

```
[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
```

Het describe de optie omvat de informatie over de URL, de poort die de sonde gebruikt en de periodiciteit van tests en storingsdrempel.

```
[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) Review ### Dit is alleen beschikbaar voor ROOT USER ###### Een Replica Set (RS) is een K8s object met als doel een stabiel aantal replica pods te behouden. Dit object detecteert ook wanneer er een ongezond aantal replica's worden gezien met een periodieke sonde naar de Pods. De RS zijn ook in naamruimten georganiseerd. [root@MxNDsh01 ~]# kubectl get rs -n cisco-msoNAME WENSELIJKE HUIDIGE KLAAR AGEauditservice-648cd4c6f8 1 1 1 3d22hbackupservice-64b755b44c 1 1 1 3d22hcloudsecservice-7df465576 1 1 1 3d22hconsistencyservice-c98955599 1 1 1 3d22hdcnmworker-5d4d5cbb64 1 3d22heeworker-5 6f9fb9ddb 1 1 3d22hendpointservice-7df9d5599c 1 1 1 3d22hexecutionservice-58ff89595f 1 1 3d22hfluentd-86785f89bd 1 1 1 3d22himportservice-8bcc8547 1 1 3d22hjobschemerservice-5d4fdfd696 1 1 3d22hnotierservice -75c988cfd4 1 1 3d22hpctagvnidservice-64b755596 1 1 1 3d22hplatformservice-65cddb946f 1 1 3d22hplatformservice2-6796576659 1 1 1 3d22hpolicyservice-545b9c7d9c 1 1 3d22hasservice-7597ff4c5 1 3d22hsservice Service-5f477dd8c7 1 1 3d22hsdwanservice-6f87cd999d 1 1 1 3d22hsiteservice-86bb756585 1 1 1 3d22hsiteupgrade-7d578f9b6d 1 1 3d22hsyncengine-5b8bd6b45 1 1 3d22htemplateeng-5cbf9fdc48 1 22hui-84588b7c96 1 1 3d22huserservice-87846f7c6 1 1 3d22h De beschrijvingsoptie bevat de informatie over de URL, de poort en de sonde, en de periodiciteit van tests en storingsdrempel. [root@MxNDsh01 ~]# kubectl beschrijf rs -n cisco-mso schemaservice-7597ff4c5Naam: schemaservice-7597ff4c5Namespace: cisco-msoSelector: k8s-app=schemaservice, pod-template-hash=7597ff4c5Labels: cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=aservice

memory.resource.case.cncf.io/schemaservice=mem-xlg-service schemaservice schemd-template-hash=7597ff4c5Annotaties: deployment.kubernetes.io/desired-replicas: 1deployment.kubernetes.io/max-replicas/Controlled By: ServiceReplica's: 1 huidige / 1 gewenstePods Status: 1 Running / 0 Waiting / 0 Succeded / 0 MisluktePod Template: Labels: deployment.kubernetes.io/revision k8s-app=schemaservice cpu.resource.case.cncf.io/schemaservice=cpu-lg-service pod-template-hash=7597ff4c5 Service Account: cisco-mso-sa Init Containers: init-msc: Afbeelding: cisco-mso/tools:3.7.1j Poort: <none> Host Port: <none> Opdracht: memory.resource.case.cncf.io/schemaservice=mem-xlg-service Environment: <none> Mount: /secrets from infracerts (rw) Cisco-mso/schemaservice:3.7.1j Poorten: 8080/TCP, 8080/UDP Host Ports: 0/TCP, 0/UDP Opdracht: /check_mongo.sh schemaservice Liften: http-get /launchscala.sh time=300s timeout=20s periode=30s #success=1 #failure=3 Milieu: JAVA_OPTS: -XX:+IdleTuningGcOnIdle Mount: /jwtsecrets from jwtsecrets (rw) /logs van logboeken infracerts (rw) msc-schemasservice-ssl: Afbeelding: cisco-mso/sslcontainer:3.7.1j Poorten: 443/UDP, 443/TCP-hostpoorten: 0/UDP, 0/TCP-opdracht: http://:8080/api/v1/schemas/health /wrapper.sh

NDO Pod Review

Een Pod is een groep nauw verwante containers die in dezelfde Linux Namespace (anders dan K8s Namespace) en in dezelfde K8s-knooppunt draaien. Dit is het meest atomaire object K8s handvatten, omdat het niet interageert met containers. De toepassing kan bestaan uit één enkele container of complexer zijn met vele containers. Met de volgende opdracht kunnen we de Pods van een bepaalde naamruimte controleren:

```
[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 ~]$
```

Het nummer in de tweede kolom verwijst naar het aantal containers voor elke Pod.

Het **describe** De optie is ook beschikbaar, die gedetailleerde informatie over de containers op elke Pod omvat.

```
[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
```

De weergegeven informatie omvat het containerbeeld voor elke container en toont de gebruikte Container Runtime. In dit geval is CRI-O (**cri-o**), vorige versies van ND gebruikt om met Docker te werken, dit beïnvloedt hoe te aan een container te bevestigen.

Spoiler

Bijvoorbeeld, wanneer **cri-o** wordt gebruikt, en we willen via een interactieve sessie verbinding maken met een container (via de exec -it optie) aan de container van de vorige output; maar in plaats van de docker het bevel, gebruiken wij het bevel **crictl**:

```
schemaservice:

   Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac

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

Wij gebruiken deze opdracht:

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

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

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

root
```

Voor latere ND-releases is de te gebruiken Container ID verschillend. Eerst moeten we de opdracht gebruiken **crictl ps** om een lijst te maken van alle containers die op elke knoop lopen. We kunnen het resultaat filteren zoals vereist.

```
[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 ~]#
```

Met de waarde uit de eerste kolom, kunnen we dan toegang tot de Container-run-time met dezelfde opdracht als voorheen:

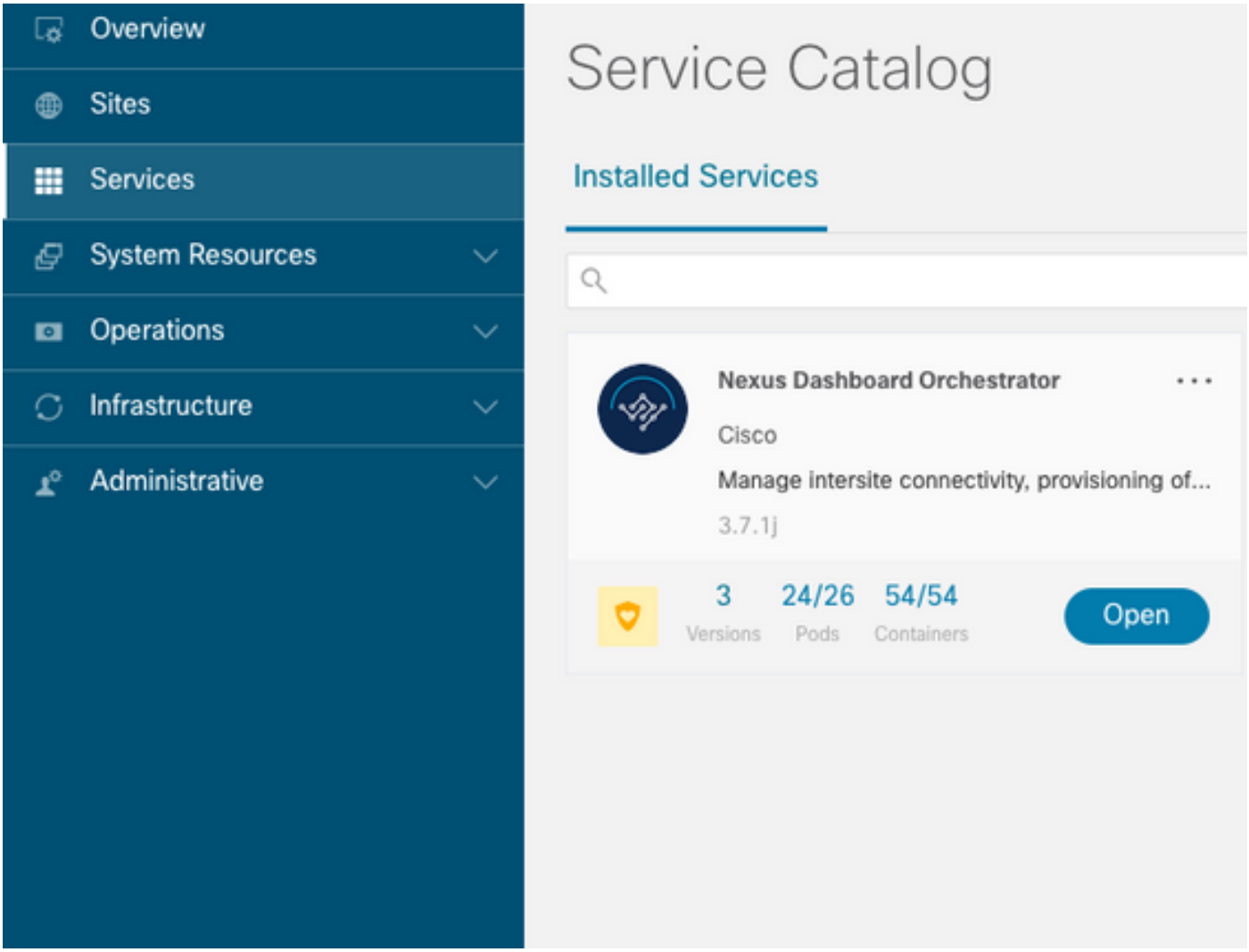
```
[root@singleNode ~]# crictl exec -it 4b26f67fc10cf bash
root@backupservice-8c699779f-j9jtr:/# pwd
/
```

Bijvoorbeeld, wanneer cri-o wordt gebruikt, en we willen door een interactieve sessie met een container verbinden (via de exec -it optie) met de container van de vorige output; maar in plaats van de docker commando, gebruiken we de criclt commando: schemaservice: Container ID: cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac Afbeelding: cisco-mso/schemaservice:3.7.1j We gebruiken deze opdracht: [root@MxNDsh01 ~]# crictl exec -it d2287f8659dec648c0100b7d22 aebd506f3f77af660238ca0c9c7e8946f4ac bashroot@schemaservice-7597ff4c5-w4x5d:/#root@schemaservice-7597ff4c5-w4x5d:/# whoamiroot Voor latere ND-releases is de te gebruiken Container-ID anders. Eerst moeten we het commando crictal ps gebruiken om alle containers op te sommen die op elke knoop lopen. We kunnen het resultaat filteren zoals vereist. [root@singleNode ~]# crictl ps| grep backupa9bb161d67295 10.31.125.241:30012/cisco-mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 2 dagen geleden die msc-backupservice-ssl 0 84b3c691cfc2b4b26f67fc10cf 10.31.125.241:30012/cisco-mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2 dagen geleden Running backupservice 0 84b3c691cfc2b[root@singleNode ~]# Met de waarde uit de eerste kolom, kunnen we dan toegang krijgen tot de Container run-time met dezelfde opdracht als voorheen: [root@singleNode ~]# exec -it 4b26f67fc10cf bashroot@backupservice-8c699779f-j9jtr:/# pwd/

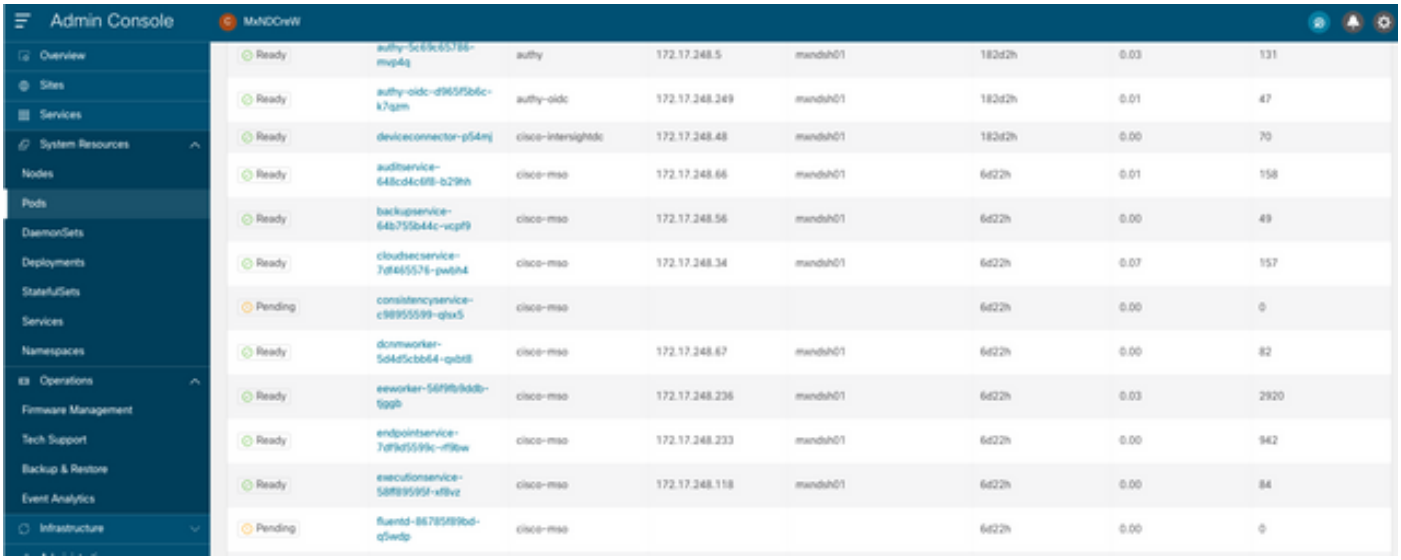
Gebruik-case Pod is niet gezond

We kunnen deze informatie gebruiken om problemen op te lossen waarom Pods van een implementatie niet gezond zijn. De Nexus Dashboard versie is 2.2-1d en de betreffende Application is Nexus Dashboard Orchestrator (NDO).

De NDO GUI geeft een onvolledige set Pods weer vanuit de Serviceweergave. In dit geval 24 van de 26 Pods.



Een andere weergave beschikbaar onder de **System Resources -> Pods** bekijken waar de Pods een andere status tonen dan **Ready**.



CLI-probleemoplossing voor ongezonde pods

Met het bekende feit de Namespace is cisco-mso (hoewel bij problemen heet, is het hetzelfde voor andere apps/namespaces) de weergave van de Pod toont als er ongezonde:

```
[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
```

Dit bijvoorbeeld, focussen we in de consistencyservice Pods. Vanuit de JSON-output kunnen we de specifieke informatie krijgen uit de statusvelden, met behulp van 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 ~]$
```

We zien het statuswoordenboek en in een lijst genaamd voorwaarden met woordenboeken als items met de toetsboodschap en waarde, is het deel {"\n} om een nieuwe regel te creëren aan het einde:

```
[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 ~]$
```

Deze opdracht laat zien hoe u het product vanaf de **get Pod** voor 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
```

Met de get pods Het bevel, kunnen wij de Peul-ID met kwesties krijgen die met moeten aanpassen van de vorige output. In dit voorbeeld **consistencyservice-c98955599-qlsx5**.

Het JSON-uitvoerformaat biedt ook de mogelijkheid om specifieke informatie te controleren vanaf de gegeven uitvoer.

```
[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 ~]$
```

De JSON-uitvoer moet informatie bevatten over de status in het kenmerk met dezelfde naam. Het bericht bevat informatie over de reden.

```
[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 ~]$
```

We hebben toegang tot informatie over de status en de vereisten voor de Pods:

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

Hier is het belangrijk om te vermelden hoe de waarde wordt berekend. In dit voorbeeld, verwijst cpu **500m** naar **500 milicores**, en de **1G** in geheugen is voor GB.

Het **Describe** de optie voor de knooppunt toont de beschikbare bron voor elke K8s-werknemer in het cluster (host of 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 ~]$
```

In het gedeelte **Allocatable wordt** het totaal aan bronnen in CPU, geheugen en opslag weergegeven dat voor elk knooppunt beschikbaar is. In het **toegewezen** gedeelte worden de reeds in gebruik zijnde bronnen vermeld. De waarde **13** voor CPU verwijst naar **13 kernen** of **13.000 (13K) millicores**.

Dit bijvoorbeeld, de knoop is **oversubscribed**, wat verklaart waarom de Pod niet kan initiëren. Nadat we de ND hebben uitgeklaard met de schrapping van ND APP's of toevoeging van VM Resources.

De Cluster probeert voortdurend om hangende beleidslijnen te implementeren, zodat als de middelen vrij zijn, de Pods kunnen worden ingezet.

```
[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
```

Met de opdracht die wordt gebruikt voor resource check, bevestigen we dat de Cluster beschikbare Resource voor CPU heeft:

```
[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 ~]$
```

De plaatsingsdetails omvatten een bericht met informatie over de huidige voorwaarden voor Pods:

```
[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

Hoe te om het Netwerk te leiden zuiver Opdrachten van binnen een Container

Omdat de containers slechts de minimale bibliotheken en de gebiedsdelen specifiek voor de Peul omvatten, zijn de meeste netwerk zuiveren hulpmiddelen (pingelen, ip route, en ip adres) niet beschikbaar binnen de container zelf.

Deze opdrachten zijn zeer nuttig wanneer er een behoefte is om netwerkproblemen op te lossen voor een service (tussen ND-knooppunten) of verbinding naar de APICS omdat verschillende microservices moeten communiceren met de controllers met de Data interface (**bond0** of **bond0br**).

Het **nsenter** hulpprogramma (root user only) staat ons toe om netwerkopdrachten uit te voeren vanaf de ND-knooppunt, aangezien het zich in de container bevindt. Vind hiervoor de proces-ID (PID) van de container die we willen zuiveren. Dit wordt bereikt met de Pod K8s ID tegen de lokale informatie van de Container Runtime, zoals Docker voor legacy versies, en **cri-o** voor nieuwere als standaard.

Inspecteer de Pod Kubernetes (K8s) ID

Uit de lijst met Pods in Cisco-mso Namespace, kunnen we de container selecteren om problemen op te lossen:

```
[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
```

De Pods moeten draaien in dezelfde K8s knooppunt. Voor productieomgevingen kunnen we de **-o wide** optie aan het eind om de knoop te weten te komen elke Peul loopt. Met de Pod K8s-ID (in het vorige uitvoervoorbeeld) kunnen we het proces (PID) controleren dat is toegewezen door de Container Runtime.

Hoe de PID te inspecteren vanuit de Container Runtime

De nieuwe standaard Container Runtime is CRI-O voor Kubernetes. Het document volgt op die regel voor de opdrachten. De door CRI-O toegewezen proces-ID (PID) kan uniek zijn in het K8s-knooppunt, dat kan worden ontdekt met de crictl het nut.

Het **ps** De optie onthult de ID gegeven door CRI-O aan elke container die de Pod, twee voor het sitesevice voorbeeld bouwt:

```
[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 ~]#
```

Met deze informatie kunnen we de inspect CRIO-ID optie om de daadwerkelijke PID te zien die aan elke container wordt gegeven. Deze informatie is nsenter opdracht:

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

Hoe te gebruiken nsenter om netwerk debug commands in een container uit te voeren

Met de PID van de output hierboven, kunnen wij als doel in de volgende bevelsyntaxis gebruiken:

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

Het **--net** de optie staat ons toe om opdrachten uit te voeren in het netwerk Namespaces, zodat het aantal beschikbare opdrachten beperkt is.

Voorbeeld:

```
[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
```

De ping is ook beschikbaar, en het test connectiviteit van de container naar buiten, in plaats van alleen de K8s knooppunt.

```
[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]
```

Hoe te om het Netwerk in werking te stellen zuiver Opdrachten van binnen een Container Omdat de containers slechts de minimale bibliotheken en de gebiedsdelen specifiek voor de Peul omvatten, zijn de meeste van netwerk zuiveren hulpmiddelen (pingelen, ip route, en ip adres) niet beschikbaar binnen de container zelf. Deze opdrachten zijn zeer nuttig wanneer er een behoefte is om netwerkproblemen op te lossen voor een service (tussen ND-knooppunten) of verbinding naar de APICS omdat verschillende microservices moeten communiceren met de controllers met de Data interface (bond0 of bond0br). Met het hulpprogramma Nententer (alleen rootgebruiker) kunnen we netwerkopdrachten uitvoeren vanaf de ND-knooppunt, omdat het zich in de container bevindt. Vind hiervoor de proces-ID (PID) van de container die we willen zuiveren. Dit wordt bereikt met de Pod K8s ID tegen de lokale informatie van de Container Runtime, zoals Docker voor legacy-versies, en cri-o voor nieuwere als standaard. Controleer de Pod Kubernetes (K8s) ID Uit de lijst met pods in de cisco-mso Namespace, kunnen we de container selecteren om problemen op te lossen: [root@MxNDsh01 ~]# kubectl get pod -n cisco-msoNAME KLAAR STATUS RESTARTS AGEconsistencyservice-569bdf5969-xkwpg 3/3 Running 0 9heeworker-65dc5dd849-485tq 2/2 Running 0 16pointservice-5db657884hkf5g 2/2 Running 0 9hexecutionservice-6c4894d4f7-p8fzk 2/2 Running 0 9hsiteservice-64dfcdf658-lvbr4 3/3 Running 0 9hsiteupgrade-68bcf987cc-tn7h 2/2 Running 0 9h De Pods moeten lopen in dezelfde K8s knooppunt. Voor productieomgevingen kunnen we de -o brede optie aan het eind toevoegen om te weten te komen welke knoop elke Pod draait. Met de Pod K8s-ID (in het vorige uitvoervoorbeeld) kunnen we het proces (PID) controleren dat is toegewezen door de Container Runtime. Hoe de PID van de Container Runtime te inspecteren De nieuwe standaard Container Runtime is CRI-O voor Kubernetes. Het document volgt op die regel voor de opdrachten. De door CRI-O toegewezen proces-ID (PID) kan uniek zijn in het K8s-knooppunt, dat met de cricktal kan

worden ontdekt. De ps-optie onthult de ID gegeven door CRI-O aan elke container die de Pod, twee voor het sitesdevice voorbeeld bouwt: [root@MxNDsh01 ~]# crictl ps |grep siteservicefb560763b06f2 172.31.0.0:30012/cisco-mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10 uur geleden Running msc-siteservice2-ssl 0 074727b4e9f51ad2d42aae1ad9 1d0195292f7fcc62f38529e135a1315c358067004a086cfed7e059986ce615b0 10 uur geleden Running siteservice-leader-election 0 074727b4b4b0 f5129b0b6d41d1e3 172.31.0.0:30012/cisco-mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10 uur geleden Running siteservice 0 074727b4e9f51[root@MxNDsh01 ~]# Met deze informatie kunnen we dan de inspect CRIO-ID optie gebruiken om de werkelijke PID gegeven aan elke container te zien. Deze informatie is nodig voor de opdracht nsenter: [root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid"pid": 239563,"pids": {"type": "pid" Hoe je nsenter kunt gebruiken om Network Debug commando's in een container uit te voeren Met de PID van de bovenstaande uitvoer, kunnen we als doel gebruiken in de volgende opdrachtsyntaxis: nsenter —target <PID> —net <NETWORK COMMANDO> De —net optie stelt ons in staat om opdrachten uit te voeren in het netwerk Namespaces, zodat het aantal beschikbare opdrachten beperkt is. Bijvoorbeeld: [root@MxNDsh01 ~]# nsenter —target 239563 —net ifconfigeth0: flags=4163<UP,BROADCAST,RUN,MULTICAST> mtu 1450inet 172.17.248.146 netmask 255.255.0.0 broadcast 0.0.0inet6 fe80:984f:32ff:fe72:7bfb prefixlen 64 scopeid 0x20 Andere 9a:4f:32:72:7b:fb txwachtelen 0 (Ethernet)RX-pakketten 916346 bytes 271080553 (258,5 MiB)RX-fouten 0 gevallen 183 overschrijdingen 0 frame 0TX-pakketten 828016 bytes 307255950 (293.0 MiB)TX-fouten 0 gevallen 0 overschrijdingen 0 carrier 0 botsingen 0lo: vlaggen=73<UP,LOOPBACK,RUN> mtu 65536inet 127.0.0.1 netmasker 25.0.0inet6:1 prefixlen 28 scopeid 0x10<host>loop txwachtelen 1000 (Local Loopback)RX-pakketten 42289 bytes 14186082 (13.5 MiB)RX fouten 0 gevallen 0 uitloop 0 frame 0TX-pakketten 42289 bytes 14186082 (13.5 MiB)TX fouten 0 gevallen 0 uitloop 0 carrier 0 botsingen 0 Het ping is ook beschikbaar en het test connectiviteit van de container naar buiten, in plaats van alleen de K8s-knooppunt. [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.WAARSCHUWING: kan 1xx.2xx.3xx.4xx's certificate, afgegeven door "/C=US/ST=CA/O=Cisco System/CN=APIC" niet verifiëren:Kan de autoriteit van de emittent niet lokaal verifiëren.WAARSCHUWING: certificaat algemene naam "APIC" komt niet overeen met de gevraagde hostnaam "1xx.2xx.3xx.4xx".HTTP verzoek verzonden. 200 OKLengte: 3251 (3.2K) [text/html]Opslaan naar: "index.html"100%[==========>] 3.251 —.-K/s in 0s2023-01-24 23:46:04 (548 MB/s) - "index.html" opgeslagen [3251/3251]

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.