NDO 리소스 문제 해결 및 검토

목차

소개

이 문서에서는 kubctl 및 컨테이너 런타임 CLI를 사용하여 NDO를 검토하고 문제를 해결하는 방법에 대해 설명합니다.

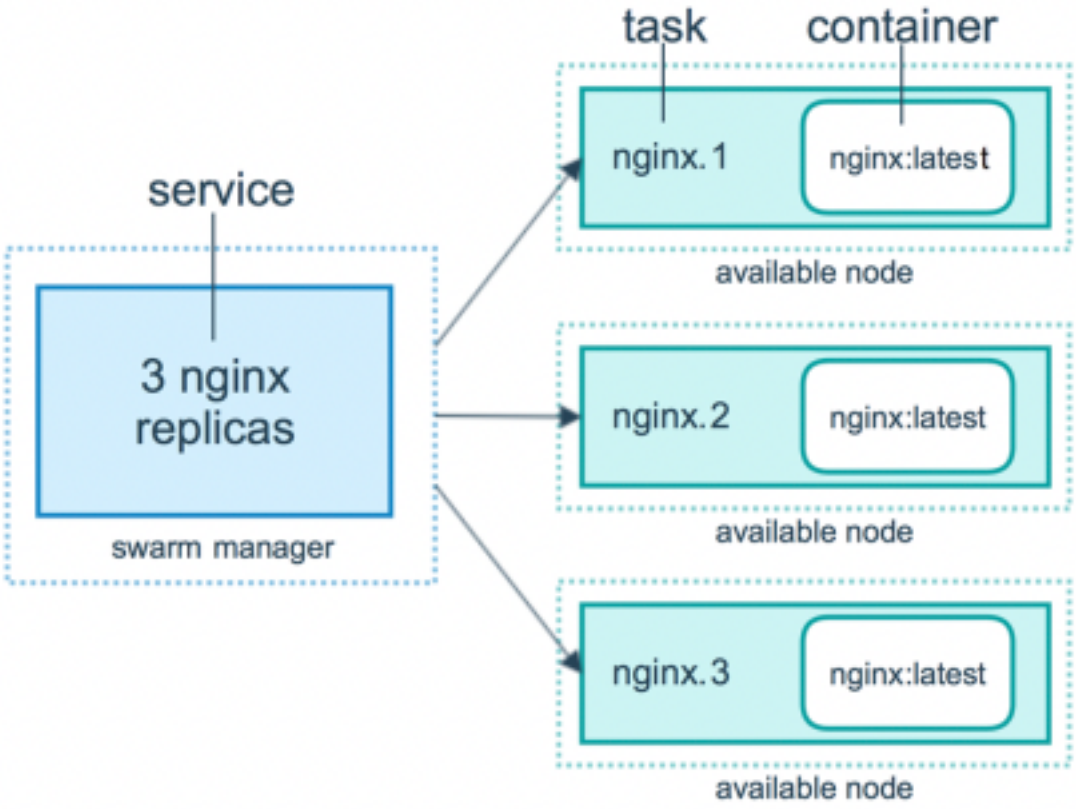
NDO 빠른 시작

Cisco Nexus Dashboard Orchestrator(NDO)는 패브릭 관리 도구로서, 사용자가 Cisco® ACI®(Application Centric Infrastructure) 사이트, Cisco Cloud ACI 사이트, Cisco Nexus NDFC(Dashboard Fabric Controller) 사이트를 포함하는 다양한 종류의 패브릭을 관리할 수 있으며, 각 패브릭은 자체 컨트롤러(APIC 클러스터, NDFC 클러스터 또는 퍼블릭 클라우드의 클라우드 APIC 인스턴스)에 의해 관리됩니다.

NDO는 단일 창을 통해 여러 데이터 센터 전반에 걸쳐 일관된 네트워크 및 정책 오케스트레이션, 확장성, 재해 복구를 제공합니다.

이전 기간에는 MSC(Multi-Site Controller)가 VMWare OVA(Open Virtual Appliance)와 함께 3노드 클러스터로 구축되어 고객이 Docker Swarm 클러스터 및 MSC 서비스를 초기화할 수 있도록 했습니다. 이 Swarm 클러스터는 MSC 마이크로서비스를 Docker 컨테이너 및 서비스로 관리합니다.

이 그림에서는 Docker Swarm이 고가용성을 위해 동일한 컨테이너의 복제본으로 마이크로 서비스를 관리하는 방법을 간단하게 보여 줍니다.



Docker Swarm은 MSC 아키텍처의 각 마이크로서비스에 대해 예상되는 복제본 수를 유지 관리합니다. Docker 스웜의 관점에서 볼 때 멀티 사이트 컨트롤러는 오케스트레이션할 수 있는 유일한 컨테이너 구축이었습니다.

Nexus Dashboard(ND)는 여러 데이터 센터 사이트를 위한 중앙 관리 콘솔이며 Cisco 데이터 센터 운영 서비스를 호스팅하는 공통 플랫폼입니다. 여기에는 Nexus Insight 및 MSC 버전 3.3 이상이 포함되며 이름을 Nexus Dashboard Orchestrator(NDO)로 변경했습니다.

MSC 아키텍처를 구성하는 대부분의 마이크로서비스는 동일하게 유지되지만, NDO는 Docker Swarm이 아닌 Kubernetes(K8s) 클러스터에 구축됩니다. 이를 통해 ND는 여러 애플리케이션 또는 구축을 하나만 오케스트레이션하는 것이 아니라 여러 개의 애플리케이션 또는 구축을 오케스트레이션할 수 있습니다.

Kubernetes with NDO Crash-Course

Kubernetes는 컨테이너화된 애플리케이션의 배포, 확장성 및 관리를 자동화하는 오픈 소스 시스템입니다. Docker로서 Kubernetes는 컨테이너 기술을 사용하지만 Docker와 연결되어 있지 않습니다. 이는 Kubernetes가 다른 컨테이너 플랫폼(Rkt, PodMan)을 지원함을 의미합니다.

스웜과 쿠베르네테스의 핵심적인 차이점은 후자는 컨테이너와 직접 연동되지 않고, 대신 Pod라고 불리는 컨테이너의 공동 배치 그룹 개념과 연동된다는 점이다.

Pod의 컨테이너는 동일한 노드에서 실행되어야 합니다. Pod 그룹을 구축이라고 합니다. Kubernetes 배포는 전체 응용 프로그램을 설명할 수 있습니다.

또한 Kubernetes는 사용자가 지정된 애플리케이션에 대해 사용할 수 있는 리소스의 양을 확인할 수 있도록 합니다. 이 작업은 Replication Controller를 사용하여 Pod 수가 애플리케이션 매니페스트와 일치하는지 확인합니다.

매니페스트는 클러스터에서 배포할 리소스를 설명하는 YAML 형식의 파일입니다. 리소스는 앞에서

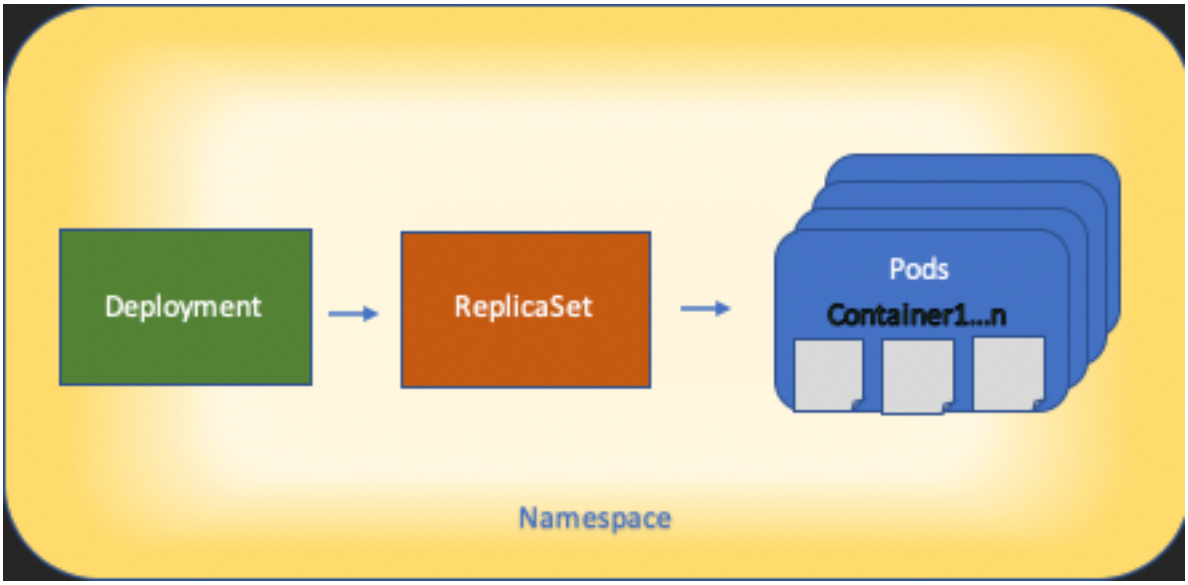
설명한 리소스 중 하나이거나 사용자가 사용할 수 있는 리소스일 수 있습니다.

애플리케이션은 하나 이상의 서비스를 통해 외부에서 액세스할 수 있습니다. Kubernetes에는 이러한 작업을 수행하는 로드 밸런서 옵션이 포함되어 있습니다.

Kubernetes는 Namespaces 개념으로 다양한 리소스를 격리하는 방법도 제공합니다. ND는 네임스페이스를 사용하여 서로 다른 응용 프로그램 및 클러스터 서비스를 고유하게 식별합니다. CLI 명령이 실행되면 항상 Namespace를 지정합니다.

ND 또는 NDO의 트러블슈팅에는 Kubernetes에 대한 깊은 지식이 필요하지 않지만 문제가 있거나 주의가 필요한 리소스를 제대로 식별하려면 Kubernetes 아키텍처에 대한 기본적인 이해가 필요합니다.

Kubernetes 리소스 아키텍처의 기본 사항은 다음 다이어그램에 나와 있습니다.



각 종류의 리소스가 다른 리소스와 상호 작용하는 방식을 기억하는 것이 중요하며, 검토 및 문제 해결 과정에서 중요한 역할을 합니다.

Kubernetes 명령을 사용한 NDO 개요

CLI 액세스 로그인

SSH를 통해 NDO에 액세스하는 경우 **admin-user** 비밀번호가 필요합니다. 하지만 대신 **rescue-user** 비밀번호입니다. 다음과 같이

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

이는 CLI 액세스를 위한 기본 모드 및 사용자이며 대부분의 정보를 볼 수 있습니다.

NDO 네임스페이스 검토

이 K8s 개념은 클러스터 전체에서 서로 다른 리소스를 격리할 수 있도록 합니다. 다음 명령을 사용

하여 구축된 여러 네임스페이스를 검토할 수 있습니다.

```
[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
```

굵게 표시된 항목은 NDO의 Applications에 속하며, 접두사 kube로 시작하는 **엔터티는** Kubernetes 클러스터에 속합니다. 각 네임스페이스에는 고유한 독립적인 구축 및 포드가 있습니다.

kubectl CLI에서는 **--namespace** 옵션, 명령을 실행하지 않고 실행하면 CLI는 Namespace를 **default** (k8의 네임스페이스):

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

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

kubectl CLI는 yaml, JSON 또는 사용자 정의 테이블 등 출력에 대해 서로 다른 종류의 형식을 허용합니다. 이는 **-o** [format] 옵션. 예를 들면 다음과 같습니다.

```
[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": ""

    }

}
```


이전 텍스트에서 출력은 해당 키 중 하나가 **항목(items)**인 사전이며 값은 각 **사전 계정**이 네임스페이스(Namespace) 항목**을** 차지하고 해당 특성은 **사전 또는 중첩된 사전의 키-값 쌍 값인 사전 목록입니다.**

이는 K8이 사용자에게 jsonpath를 출력으로 선택할 수 있는 옵션을 제공하기 때문에 JSON 데이터 어레이에 대한 복잡한 작업이 가능하기 때문에 의미가 있습니다. 예를 들어, 이전 출력에서 **name** 네임스페이스의 경우 항목 목록의 값에 액세스한 다음 metadata 사전 및 키 값 가져오기 **name**. 이 명령은 다음 명령으로 수행할 수 있습니다.

```
[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 ~]$
```

설명된 계층은 필요한 특정 정보를 가져오는 데 사용됩니다. 기본적으로 모든 항목은 **items** **항목[*]이** 있는 목록을 선택한 다음 metadata 및 **name** metadata.**name**을 사용하면 쿼리에 표시할 다른 값을 포함할 수 있습니다.

데이터 배열에서 정보를 가져오는 것과 유사한 방법을 사용하는 사용자 지정 열 옵션도 마찬가지입니다. 예를 들어, FirePOWER Services에 대한 **name** 및 UID 다음 명령을 적용할 수 있습니다.

```
[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
```

출력에는 표시할 각 열의 이름이 필요하며 출력 값을 지정해야 합니다. 이 예제에서는 두 개의 열이 있습니다. **NAME** 및 **UID**. 이 값은 다음에 속합니다. .metada.name 및 .metadata.uid 각각. 자세한 내용 및 예제는 다음 사이트에서 확인할 수 있습니다.

JSONPath 지원

사용자 지정 열

NDO 구축 검토

구축은 ReplicaSet 및 Pod를 관리하기 위해 조인된 공간을 제공하는 K8s 객체입니다. 구축에서는 애플리케이션에 속하는 모든 Pod의 롤아웃과 각 Pod의 예상 복사본 수를 처리합니다.

kubectl CLI에는 지정된 네임스페이스에 대한 구축을 확인하는 명령이 포함되어 있습니다.

```
[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
```

동일한 사용자 지정 테이블을 사용하여 deployment 대신 **namespace** 및 **-n** 옵션을 선택하면 이전과 동일한 정보를 볼 수 있습니다. 산출물이 유사한 구조로 되어 있기 때문이다.

```
[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
```

각 마이크로서비스의 Pod 수가 아니라 구축에 사용할 수 있는 복제본 수를 확인합니다.

이 키워드는 **describe** 대신 **get** 리소스에 대한 자세한 정보를 표시하려면(이 경우 schemaservice 구축)

```
[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 ~]$
```

이 **describe** 또한 이 명령을 통해 **--show-events=true** 구축과 관련된 이벤트를 표시할 수 있습니다.

스포일러

NDO RS(Replica Set) 검토

스포일러

루트 사용자 #####에만 사용할 수 있습니다.

RS(Replica Set)는 안정적인 복제본 포드(Pod) 수를 유지하는 것을 목표로 하는 K8s 객체입니다. 또한 이 개체는 Pod에 대한 주기적인 프로브를 통해 비정상적인 복제본 수가 표시되는 경우를 탐지합니다.

또한 RS는 네임스페이스로 구성됩니다.

```
[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
```

이 describe 옵션은 URL, 프로브가 사용하는 포트, 테스트 주기 및 실패 임계값에 대한 정보를 포함합니다.

```
[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 RS(Replica Set) 검토 #### 이 옵션은 ROOT 사용자 #####에만 사용할 수 있습니다. RS(Replica Set)는 안정적인 수의 복제본 Pod를 유지 관리하는 것을 목적으로 하는 K8s 개체입니다. 또한 이 개체는 Pod에 대한 주기적인 프로브를 통해 비정상적인 복제본 수가 표시되는 경우를 탐지합니다. 또한 RS는 네임스페이스로 구성됩니다. [root@MxNDsh01 ~]# kubctl get rs -n cisco-msoNAME DESIRED CURRENT READY AGEauditservice-648cd4c6f8 1 1 3d22hbackupservice-64b755b44c 1 1 1 3d22hcloudsecservice-7df465576 1 1 1 1 3d22hconsistencyservice-c98955599 1 1 1 1 3d22hdcnmworker-5d4d5cbb64 1 1 1 3d22heeworker-56f9fb9ddb 1 1 3d22hendpointservice-7d9d5599c 1 1 1 1 3d2 executionservice-58ff89595f 1 1 1 3d22hfluentd-86785f89bd 1 1 3d22himportservice-88bcc8547 1 1 3d22hjobschedulerservice-5d4fdfd696 1 1 1 3d22hnotifyservice-75c988cfd4 1 1 1 3d22hpctagvnidservice-644b755596 1 1 1 3d22hplatformservice-65cddb946f 1 1 1 3d22hplatformservice2-6796576659 1 1 3d22hpolicyservice-54 b9c7d9c 1 1 3d22hschemaservice-7597ff4c5 1 1 3d22hsdaservice-5f477dd8c7 1 1 3d22hsdwanservice-6f87cd999d 1 1 1 3d22hsiteservice-86bb756585 1 1 1 3d22hsiteupgrade-7d578f9b6d1 1 3d22hsyncengine-5b8bdd6b45 1 1 1 1 3d22htemplateng-5cbf9fdc481 1 3d22hui-84588b5 c96 1 1 1 3d22huserservice-87846f7c6 1 1 1 3d22h describe 옵션은 URL, 프로브가 사용하는 포트, 테스트의 주기 및 실패 임계값에 대한 정보를 포함합니다. [root@MxNDsh01 ~]# kubectl describe rs -n cisco-mso schemaservice-7597ff4c5Name: schemaservice-7597ff4c5Namespace: cisco-msoSelector: k8s-app=schemaservice,pod-template-hash=7597ff4c5Labels: cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service pod-template-hash=7597ff4c5Annotations: deployment.kubernetes.io/desired-replicas: 1 deployment.kubernetes.io/max-replicas: 1 deployment.kubernetes.io/revision: 1로 제어됨: 배포 /schemaservice복제본: 1 현재 / 1 원하는 Pod 상태: 1 실행 / 0 대기 / 0 성공 / 0 실패 Pod 템플릿: 레이블: cpu.resource.case.cncf.io/schemaservice=cpu-lg-service k8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service - pod template=75 7ff4c5 서비스 계정: cisco-mso-sa Init Containers: init-msc: 이미지: cisco-mso/tools:3.7.1j 포트: <none> 호스트 포트: <none> 명령: /check_mongo.sh 환경: <none> 마운트: /secrets from infracerts (rw) 컨테이너: schemaservice: 이미지: cisco-mso/schemaservice:3.7.1j 포트: 8080/TCP, 8080/UDP 호스트 포트: 0/TCP, 0/UDP 명령: /launchscala.sh schemaservice 라이브니스: http-get http://:8080/api/v1/schemas/health delay=300s timeout=20s period=30s #success=1 #failure=3 환경: JAVA_OPTS: ID:+IdleTuningGcOnIdle Mounts: /jwtsecrets from jwtsecrets (rw) /logs from logs (rw) /secrets from infra (rw) msc-schemaservice-ssl: 이미지: cisco-mso/sslcontainer:3.7.1j Ports: 443/UDP, 443/TCP Host Ports: 0/UDP, 0/TCP Command: /wrapper.sh

NDO Pod 검토

Pod는 동일한 Linux 네임스페이스(K8s 네임스페이스와 다름) 및 동일한 K8s 노드에서 실행되는 밀접하게 관련된 컨테이너 그룹입니다. 이것은 컨테이너와 상호 작용하지 않기 때문에 K8s 핸들이 가장 원자적인 개체이다. 어플리케이션은 단일 컨테이너로 구성될 수 있거나 또는 많은 컨테이너들과 더 복잡할 수 있다. 다음 명령을 사용하여 지정된 네임스페이스의 Pod를 확인할 수 있습니다.

```
[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 ~]$
```

두 번째 열에서 볼 수 있는 숫자는 각 Pod의 컨테이너 수를 나타냅니다.

이 **describe** 각 Pod의 컨테이너에 대한 세부 정보가 포함된 옵션도 제공됩니다.

```
[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
```

표시되는 정보에는 각 컨테이너의 컨테이너 이미지가 포함되며 사용된 컨테이너 런타임이 표시됩니다. 이 경우 CRI-O(**cri-o**)의 이전 버전인 ND는 Docker와 함께 작동하기 때문에 컨테이너에 연결하는 방법에 영향을 줍니다.

[스포일러]

예를 들어, **cri-o** 인터랙티브 세션을 통해 컨테이너에 연결하고자 합니다. exec -it (옵션) 컨테이너를 이전 출력에서 가져온 다음 docker 이 명령은 criclt 명령을 **사용하**는 것입니다.

```
schemaservice:

   Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac

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

다음 명령을 사용합니다.

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

root@schemaservice-7597ff4c5-w4x5d:/#

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

root
```

이후 ND 릴리스의 경우 사용할 컨테이너 ID가 다릅니다. 먼저, 이 명령을 사용하여 **crictl ps** 각 노드에서 실행되는 모든 컨테이너를 나열합니다. 필요에 따라 결과를 필터링할 수 있습니다.

```
[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 ~]#
```

첫 번째 열의 값을 사용하여 이전과 동일한 명령으로 컨테이너 런타임에 액세스할 수 있습니다.

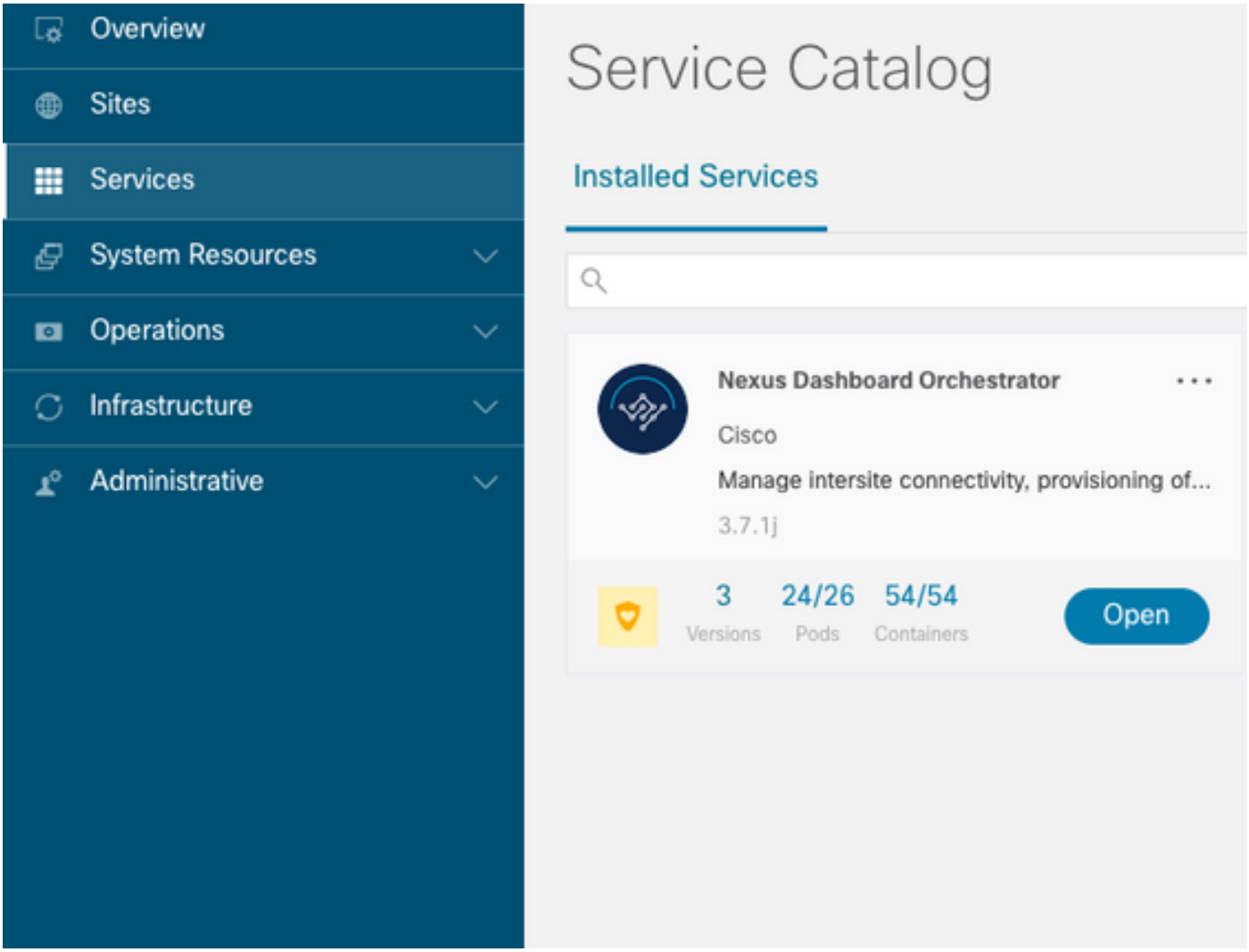
```
[root@singleNode ~]# crictl exec -it 4b26f67fc10cf bash
root@backupservice-8c699779f-j9jtr:/# pwd
```

/
예를 들어, cri-o를 사용하고, 대화형 세션을 통해 컨테이너에 이전 출력의 컨테이너에 연결하려고 하지만 docker 명령 대신 crictl 명령을 사용합니다. schemaservice: Container ID: cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac Image: cisco-mso/schemaservice:3.7.1j 이 명령을 사용합니다. [root@MxNDsh01 ~]# crictl exec -it d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9e8946f4ac bashroot@schemaservice-7597ff4c5-w4x5d/#root@schemaservice-7597ff4c5-w4x5d/# whoamiroot 릴리스에서는 사용할 컨테이너 ID가 다릅니다. 먼저, 명령 crictl ps를 사용하여 각 노드에서 실행되는 모든 컨테이너를 나열해야 합니다. 필요에 따라 결과를 필터링할 수 있습니다. [root@singleNode ~]# crictl ps| grep backupa9bb161d67295 10.31.125.241:30012/cisco-mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 msc-backupservice-ssl 0 84b3c691cfc2b4b26f67fc10cf 10.31.125.241:30012/cisco-mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2일 전 backupservice 0 84b3c691cfc2b[root@singleNode ~]# 첫 번째 열의 값을 사용하여 이전과 동일한 명령으로 컨테이너 런타임 기능에 액세스할 수 있습니다. [root@singleNode ~]# crictl exec -it 4b26f67fc10cf bashroot@backupservice-8c699779f-j9jtr:/# pwd/

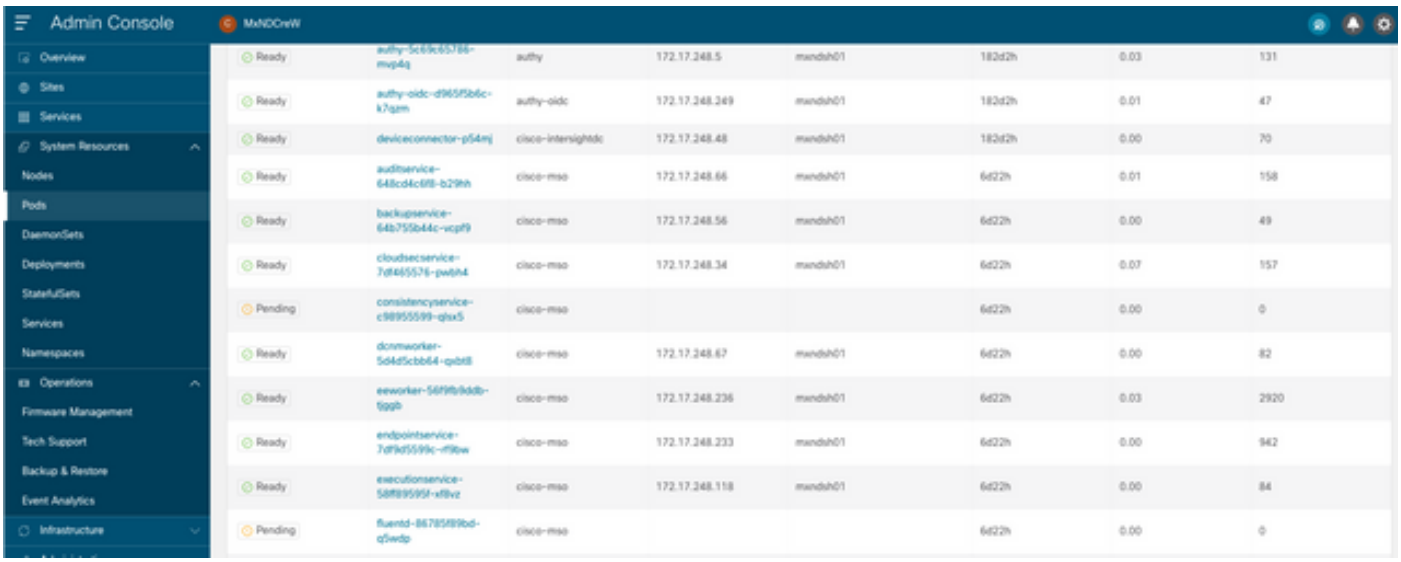
사용 사례 포드가 정상이 아님

이 정보를 사용하여 구축의 Pod가 정상적이지 않은 이유를 트러블슈팅할 수 있습니다. 이 예에서 Nexus 대시보드 버전은 2.2-1d이고 영향을 받는 애플리케이션은 Nexus Dashboard Orchestrator(NDO)입니다.

NDO GUI는 서비스 보기에서 불완전한 포드 집합을 표시합니다. 이 경우에는 Pod 26개 중 24개가 사용됩니다.



다음 아래에서 사용할 수 있는 다른 보기 **System Resources -> Pods** 포드가 다른 상태를 표시하는 위치 보기 **Ready**.



비정상 Pod에 대한 CLI 트러블슈팅

알려진 사실에 따라 네임스페이스는 cisco-mso입니다(트러블슈팅할 경우 다른 앱/네임스페이스에 대해 동일하지만). 비정상 네임스페이스가 있는 경우 Pod 보기가 표시됩니다.

```
[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
```

이 예에서는 일관성 서비스 Pod에 중점을 둡니다. JSON 출력에서 jsonpath를 사용하여 상태 필드에서 특정 정보를 가져올 수 있습니다.

```
[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 ~]$
```

상태 사전이 표시되고 **조건** 및 사전을 키 **메시지** 및 값과 함께 항목으로 **부르는 목록**에서 {"\n"} 부분은 끝에 새 줄을 만드는 것입니다.

```
[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 ~]$
```

이 명령은에서 확인하는 방법을 보여줍니다. **get Pod** 네임스페이스:

```
[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
```

를 사용하여 get pods 명령을 사용하면 이전 출력의 포드 ID와 일치해야 하는 문제가 있는 포드 ID를 가져올 수 있습니다. 이 예에서는 **consistencyservice-c98955599-qlsx5.**

JSON 출력 형식은 또한 지정된 출력에서 특정 정보를 확인하는 방법을 제공합니다.

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

JSON 출력에는 같은 이름의 특성에 있는 상태에 대한 정보가 포함되어야 합니다. 메시지에는 이유에 대한 정보가 포함되어 있습니다.

```
[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 ~]$
```

Pod의 상태 및 요구 사항에 대한 정보에 액세스할 수 있습니다.

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

여기서 그 값이 어떻게 산출되는지 언급하는 것이 중요하다. 이 예에서, cpu 500m은 500개의 밀리 코어를 가리키며, 메모리의 1G는 GB를 위한 것이다.

이 Describe 노드에 대한 옵션은 클러스터(호스트 또는 VM)의 각 K8s 작업자에 대해 사용 가능한 리소스를 표시합니다.

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

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

Allocatable **섹션**은 각 노드에 대해 사용 가능한 CPU, 메모리 및 스토리지의 총 리소스를 보여줍니다. Allocated(**할당됨**) 섹션에는 이미 사용 중인 리소스가 표시됩니다. CPU 값 **13**은 **13개 코어 또**는 **13,000(13K) 밀리코어를 의미합니다**.

이 예에서 노드는 초과 **가입되어** 있으므로 Pod를 시작할 수 없는 이유를 설명합니다. ND APP를 삭제하거나 VM 리소스를 추가하여 ND를 지운 후

클러스터는 보류 중인 정책을 계속 구축하려고 하므로 리소스가 사용 가능한 경우 포드를 구축할 수 있습니다.

```
[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
```

리소스 확인에 사용되는 명령을 사용하여 클러스터에 CPU에 사용할 수 있는 리소스가 있는지 확인합니다.

```
[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 ~]$
```

구축 세부 정보에는 Pod의 현재 조건에 대한 정보가 포함된 메시지가 포함됩니다.

```
[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 ~]$
```

스포일러

컨테이너 내부에서 네트워크 디버그 명령을 실행하는 방법

컨테이너에는 Pod에 특정한 최소 라이브러리와 종속성만 포함되므로, 대부분의 네트워크 디버그 도구(ping, ip 경로 및 ip 주소)는 컨테이너 자체 내에서 사용할 수 없습니다.

이러한 명령은 여러 마이크로서비스가 데이터 인터페이스(bond0 또는 bond0br)를 사용하여 컨트롤러와 통신해야 하기 때문에 서비스(ND 노드 간) 또는 Apic에 대한 연결에 대한 네트워크 문제를 해결해야 하는 경우에 매우 유용합니다.

이 **nsenter** 유틸리티(루트 사용자만 해당)를 사용하면 컨테이너 내에서 ND 노드의 네트워크 명령을 실행할 수 있습니다. 이를 위해 디버깅할 컨테이너에서 프로세스 ID(PID)를 찾습니다. 이 작업은 컨테이너 런타임의 로컬 정보에 대한 Pod K8s ID(예: 레거시 버전의 Docker)를 사용하여 수행됩니다. **cri-o** 기본적으로 새 를 선택합니다.

Pod Kubernetes(K8s) ID 검사

cisco-mso 네임스페이스 내의 Pod 목록에서 트러블슈팅할 컨테이너를 선택할 수 있습니다.

```
[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
```

Pod는 동일한 K8s 노드에서 실행해야 합니다. 운영 환경의 경우 **-o wide** 각 Pod에서 실행되는 노드를 찾을 수 있습니다. 이전 출력 예에서 굵게 표시된 Pod K8s ID를 사용하여 컨테이너 런타임에서

할당한 프로세스(PID)를 확인할 수 있습니다.

컨테이너 런타임에서 PID를 검사하는 방법

새로운 기본 컨테이너 런타임은 Kubernetes에 대한 CRI-O입니다. 따라서 이 문서는 명령에 대한 규칙을 따릅니다. CRI-O에 의해 할당된 프로세스 ID(PID)는 K8s Node에서 고유할 수 있으며, crictl 유틸리티.

이 **ps** 옵션은 CRI-O가 제공한 ID를 Pod를 구축하는 각 컨테이너에 표시합니다(사이트 서비스 예시: 2개).

```
[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 ~]#
```

이 정보를 사용하여 inspect CRIO-ID 각 컨테이너에 제공된 실제 PID를 확인할 수 있습니다. 이 정보는 nsenter 명령을 사용합니다:

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

컨테이너 내에서 Network Debug 명령을 실행하기 위해 프린터를 사용하는 방법

위 출력의 PID를 사용하면 다음 명령 구문에서 타겟으로 사용할 수 있습니다.

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

이 **--net** 옵션을 사용하면 네트워크 네임스페이스에서 명령을 실행할 수 있으므로 사용 가능한 명령 수가 제한됩니다.

예를 들면 다음과 같습니다.

```
[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
```

ping도 사용할 수 있으며 K8s 노드만 테스트하는 것이 아니라 컨테이너에서 외부로의 연결을 테스트합니다.

```
[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]
```

컨테이너 내부에서 네트워크 디버그 명령을 실행하는 방법 컨테이너에는 Pod에 특정한 최소 라이브러리와 종속성만 포함되므로 대부분의 네트워크 디버그 도구(ping, ip 경로 및 ip 주소)를 컨테이너 내부에서 사용할 수 없습니다. 이러한 명령은 여러 마이크로서비스가 데이터 인터페이스(bond0 또는 bond0br)를 사용하여 컨트롤러와 통신해야 하므로 서비스에 대한 네트워크 문제(ND 노드 간) 또는 Apic에 대한 연결 문제를 해결해야 할 때 매우 유용합니다. Enter 유틸리티(루트 사용자만 해당)를 사용하면 컨테이너 내에서 ND 노드의 네트워크 명령을 실행할 수 있습니다. 이를 위해 디버깅할 컨테이너에서 프로세스 ID(PID)를 찾습니다. 레거시 버전의 경우 Docker, 새로운 버전의 경우 cri-o 등 Container Runtime의 로컬 정보에 대한 Pod K8s ID로 이를 수행합니다. Pod Kubernetes(K8s) ID 검사 cisco-mso 네임스페이스 내의 Pod 목록에서 트러블슈팅할 컨테이너를 선택할 수 있습니다. [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 163mentpointservice-5db6f57884-hkf5g 2/2 Running 0 9hexecutionservice-6c4894d4f7-p8fzk 2/2 Running 9hhwise teservice-64dfcdf658-lvbr4 3/3 Running 0 9hsiteupgrade-68bcf987cc-ttn7h 2/2 Running 0 9h Pod는 동일한 K8s 노드에서 실행되어야 합니다. 프로덕션 환경의 경우 끝에 -o wide 옵션을 추가하여 각 Pod에서 실행되는 노드를 확인할 수 있습니다. 이전 출력 예에서 굵게 표시된 Pod K8s ID를 사용하여 컨테이너 런타임에서 할당한 프로세스(PID)를 확인할 수 있습니다. 컨테이너 런타임에서 PID를 검사하는 방법 새로운 기본 컨테이너 런타임은 Kubernetes용 CRI-O입니다. 따라서 이 문서는 명령에 대한 규칙을 따릅니다. CRI-O에서 할당한 PID(Process ID)는 K8s Node에서 고유할 수 있으며, 이는 crictl 유틸리티로 검색할 수 있습니다. ps 옵션은 CRI-O가 Pod를 구축하는 각 컨테이너에 제공한 ID를 나타냅니다. 사이트 서비스 예제는 [root@MxNDsh01 ~]# crictl ps입니다. |grep siteservicefb560763b06f2 172.31.0.0:30012/cisco-mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10시간 전 Running msc-siteservice2-ssl 0 074727b4e9f51ad2d42aae1ad9 1d0195292f7fcc62f38529e135a1315c358067004a086cfed7e059986ce615b0 10시간 전 Running siteservice-leader-selection 0 074727b4e9f5129b6d1d1e3 172.31.0.0:30012/cisco-mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10시간 전 Siteservice 0 074727e9f551[root@MxNDsh01 ~] 이 정보를 사용하여 각 컨테이너에 제공된 실제 PID를 확인하기 위해 inspect CRIO-ID 옵션을 사용할 수 있습니다. 이 정보는 nsenter 명령에 필요합니다. [root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid"pid": 239563,"pid": {"type": "pid" How to Use nsenter to Run Network Debug Commands Inside a Container Inside a Container 위의 출력의 PID를 사용하여 다음 명령 구문에서 대상으로 사용할 수 있습니다. nsenter —target <PID> —net <NETWORK COMMAND> —net 옵션을 사용하면 Namespaces에서 명령을 실행할 수 있으므로 사용 가능한 명령 수가 제한됩니다. 예:

[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>ether 9a:4f:72:7b:fb txqueuelen 0 (Ethernet)RX packets 916346 bytes 271080553 (25 B)RX 오류 0 삭제됨 183 오버런 0 프레임 0TX 패킷 828016 바이트 307255950 (293.0 MiB)TX 오류 0 삭제됨 0 오버런 0 캐리어 0 충돌 0lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536inet 127.0.0.1 netmask 255.0.0.0inet6 ::1 prefixlen 128 scopeid 0x10<host>loop txqueuelen 1000 (Local Loopback)RX 패킷 42289 바이트 14186082 (13.5 MiB)RX 오류 0 삭제됨 0 프레임 0TX 패킷 42289 바이트 14186082 (13.5.5 MiMiMiMiMiB TX errors 0 dropped 0 overruns 0 carrier 0 collision 0 ping도 사용할 수 있으며 K8s 노드만 테스트하는 것이 아니라 컨테이너에서 외부로의 연결을 테스트합니다.
[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: 1xx.2xx.3xx.4xx의 인증서를 확인할 수 없습니다. '/C=US/ST=CA/O=Cisco System/CN=APIC'에서 발급됨: 발급자의 권한을 로컬로 확인할 수 없습니다.경고: 인증서 일반 이름 'APIC'이(가) 요청된 호스트 이름 '1xx.2xx.3xx.4xx'와(과) 일치하지 않습니다. HTTP 요청 전송, 응답 대기 중.. 200 OKLength: 3251 (3.2k) text/html]저장 위치: 'index.html'100%[==========>] 3,251 —.-K/s in 0s2023-01-24 23:46:04(548MB/s) - 'index.html'이 저장되었습니다[3251/3251]

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.