NDOリソースのトラブルシューティングとレビュー

内容

[概要](#)
[NDO QuickStart](#)
[Kubernetes with NDO Crash-Course](#)
[KubernetesコマンドによるNDOの概要](#)
[CLIアクセスログイン](#)
[NDO名前空間の確認](#)
[NDO導入のレビュー](#)
[NDOレプリカセット(RS)の確認](#)
[NDOポッドレビュー](#)
[ユースケースPodが正常でない](#)
[異常なポッドのCLIトラブルシューティング](#)
[コンテナ内からnetwork debugコマンドを実行する方法](#)
[Pod Kubernetes(K8s)IDの検査](#)
[コンテナランタイムからのPIDの検査方法](#)
[コンテナ内でnetwork debugコマンドを実行するためのセンサーの使用方法](#)

概要

このドキュメントでは、kubectlおよびコンテナランタイムCLIを使用してNDOを確認し、トラブルシューティングする方法について説明します。

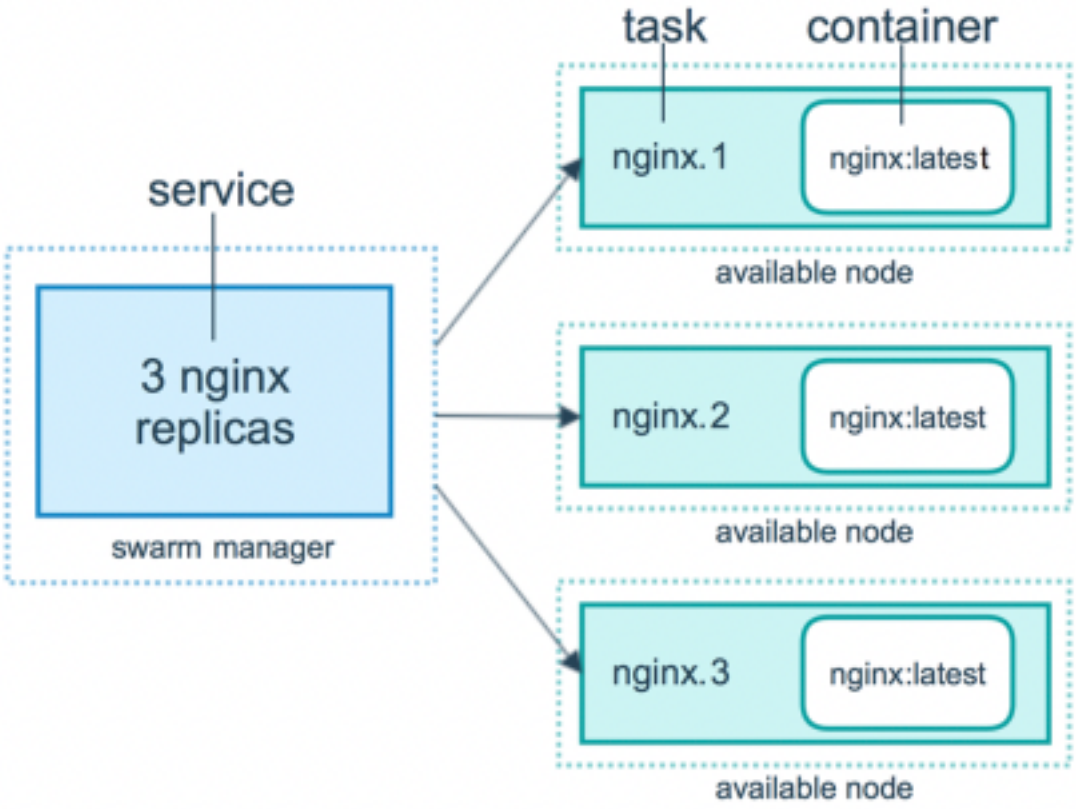
NDO QuickStart

Cisco Nexus Dashboard Orchestrator(NDO)は、ファブリック管理ツールです。このツールを使用すると、Cisco® Application Centric Infrastructure(Cisco ACI®)サイト、Cisco Cloud ACIサイト、Cisco Nexus Dashboard Fabric Controller(NDFC)サイトなどのさまざまな種類のファブリックを管理できます。各ファブリックは、独自のコントローラ (パブリッククラウド内のAPICクラスタ、NDFCクラスタ、またはCloud APICインスタンス) で管理されます。

NDOは、1つの画面で複数のデータセンター間で一貫したネットワークとポリシーのオーケストレーション、拡張性、およびディザスタリカバリを提供します。

初期には、VMWare Open Virtual Appliances(OVA)を使用した3ノードクラスタとしてMSC(Multi-Site Controller)が導入され、Docker SwarmクラスタとMSCサービスを初期化できました。このSwarmクラスタは、MSCマイクロサービスをDockerコンテナおよびサービスとして管理します。

この図は、Docker Swarmがハイアベイラビリティを実現するためにマイクロサービスを同じコンテナのレプリカとして管理する方法を簡単に示しています。



Docker Swarmは、MSCアーキテクチャ内の各マイクロサービスに対して予想される数のレプリカを維持する役割を担っていました。Docker Swarmの観点からは、マルチサイトコントローラがオーケストレーションを行う唯一のコンテナ展開でした。

Nexus Dashboard(ND)は、複数のデータセンターサイトに対応する中央管理コンソールであり、Nexus InsightやMSCバージョン3.3以降を含むシスコのデータセンター運用サービスをホストし、Nexus Dashboard Orchestrator(NDO)に名前を変更した共通のプラットフォームです。

MSCアーキテクチャを構成するマイクロサービスの大部分は同じままですが、NDOはDocker Swarmクラスタではなく Kubernetes(K8s)クラスタに導入されます。これにより、NDは1つのアプリケーションまたは導入ではなく、複数のアプリケーションまたは導入を調整できます。

Kubernetes with NDO Crash-Course

Kubernetesは、コンテナ化されたアプリケーションの導入、スケーラビリティ、および管理を自動化するオープンソースシステムです。Dockerとして、Kubernetesはコンテナテクノロジーで動作しますが、Dockerとは関係ありません。これは、Kubernetesが他のコンテナプラットフォーム(Rkt、PodMan)をサポートすることを意味します。

SwarmとKubernetesの主な違いは、後者はコンテナを直接操作するのではなく、代わりにポッドと呼ばれるコンテナのグループを同じ場所に配置するという概念で動作することです。

ポッド内のコンテナは同じノードで実行する必要があります。ポッドのグループは導入と呼ばれます。Kubernetesの導入では、アプリケーション全体を記述できます。

また、Kubernetesは、ユーザが特定のアプリケーションに対して一定量のリソースを確実に使用できるようにしています。これは、Replication Controllerを使用して行われ、Podの数がアプリケーションマニフェストと一致していることを確認します。

マニフェストは、クラスターによって展開されるリソースを記述するYAML形式のファイルです

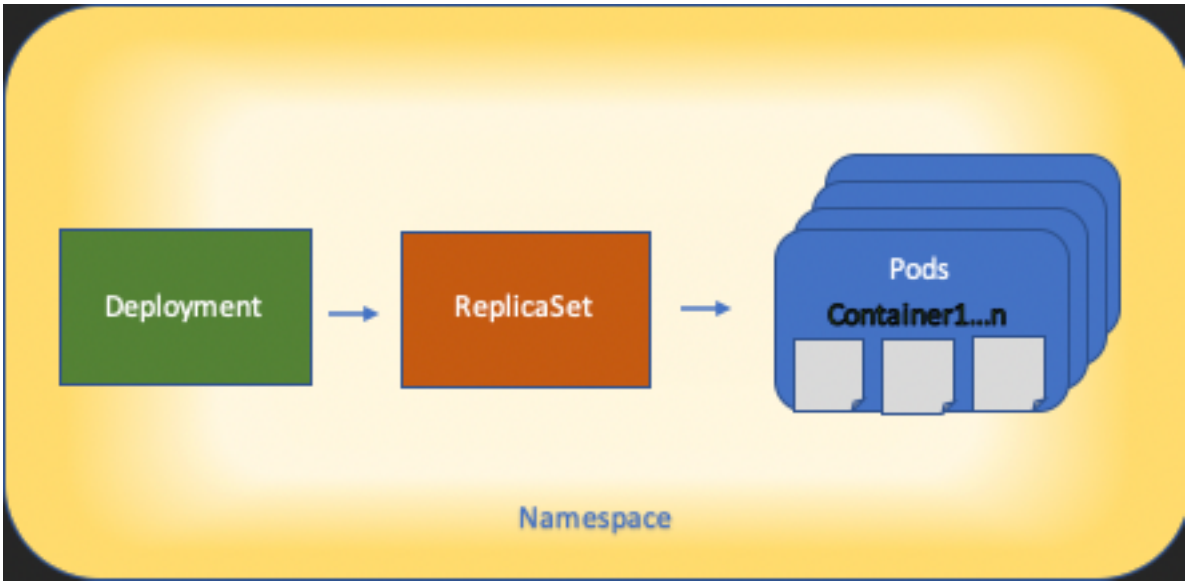
。リソースは、前に説明したリソースまたはユーザが使用できるその他のリソースです。

アプリケーションには、1つ以上のサービスを使用して外部からアクセスできます。
Kubernetesには、これを実現するためのロードバランサオプションが含まれています。

Kubernetesは、名前空間の概念を使用して異なるリソースを分離する方法も提供します。NDは名前空間を使用して、さまざまなアプリケーションとクラスターサービスを一意に識別します。
CLIコマンドを実行するときは、常にネームスペースを指定します。

NDまたはNDOのトラブルシューティングにはKubernetesに関する深い知識は必要ありませんが、問題のあるリソースや注意が必要なリソースを適切に特定するには、Kubernetesアーキテクチャに関する基本的な知識が必要です。

Kubernetesリソースアーキテクチャの基本を次の図に示します。



各リソースが他のリソースとどのように相互作用するかを覚えておくことが重要であり、レビューおよびトラブルシューティングプロセスで重要な役割を果たします。

KubernetesコマンドによるNDOの概要

CLIアクセスログイン

NDOへのSSHによるCLIアクセスの場合は、 **admin-user** パスワードが必要です。ただし、代わりに **rescue-user** password が base64 でエンコードされたものです。次のような場合：

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

これはCLIアクセスのデフォルトモードとユーザであり、ほとんどの情報を表示できます。

NDO名前空間の確認

このK8sの概念により、クラスタ全体で異なるリソースを分離できます。次のコマンドを使用し

て、導入されたさまざまなネームスペースを確認できます。

```
[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
```

太字のエントリはNDOのアプリケーションに属し、プレフィックスkubeで始まるエンティティはKubernetesクラスタに属しています。各名前空間には独自の展開とポッドがあります

kubectl CLIでは、 **--namespace** オプションを指定せずにコマンドを実行すると、CLIは名前空間が **default** （k8sの名前空間）：

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

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

kubectl CLIでは、yaml、JSON、またはカスタムメイドのテーブルなど、さまざまな種類の出力フォーマットを使用できます。これを実現するには、 **-o** [format]オプションを使用します。以下に、いくつかの例を示します。

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


前のテキストからの出力はディクショナリで、そのキーの1つがitemsと呼ばれ、値はディクショナリのリストで、各ディクショナリがNamespaceエントリを占め、その属性がディクショナリ内のキーと値のペア値またはネストされたディクショナリです。

これは、K8sでは出力としてjsonpathを選択するオプションがユーザに提供されるため、JSONデータ配列の複雑な操作が可能になります。たとえば、前の出力から、 **name** 名前空間の場合は、項目リストの値にアクセスし、 metadata キーの値を取得します。 **name**.次のコマンドを使用して実行できます。

```
[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 ~]$
```

説明されている階層は、必要な特定の情報を取得するために使用されます。基本的に、すべての項目は **items** **items[*]**でリストし、次にキーを使用して metadata と **name** **metadata.name**を使用すると、クエリに表示する他の値を含めることができます。

同様の方法でデータ配列から情報を取得するカスタム列のオプションについても同様です。たとえば、テーブルを作成する際に、 **name** および UID 値の場合は、次のコマンドを適用できます。

```
[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
```

出力では、表示する各列に名前を付け、出力に値を割り当てる必要があります。この例では、次の2つの列があります。 **NAME** と **UID**.これらの値は、 .metada.name と .metadata.uid それぞれに対応.詳細と例については、次のURLを参照してください。

JSONPathサポート

カスタム列

NDO導入のレビュー

Deploymentは、ReplicaSetとPodを管理するための結合スペースを提供するK8sオブジェクトです。導入では、アプリケーションに属するすべてのポッドのロールアウトと、各ポッドの予想コピー数が処理されます。

kubectl CLIには、任意の名前空間の展開を確認するコマンドが含まれています。

```
[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
```

同じカスタムテーブルを使用して、 deployment 代わりに **namespace** および **-n** オプションを選択すると、以前と同じ情報が表示されます。 これは、出力が同様の方法で構成されているためです。

```
[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
```

表示されるコピーの数は、各マイクロサービスのポッドの数ではなく、導入用であることに注意してください。

キーワードは **describe** 代わりに **get** リソースに関するより詳細な情報を表示するには、次の手順に従います。この場合、schemaservice配置は次のとおりです。

```
[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 ~]$
```

「 **describe** コマンドを使用すると、 **--show-events=true** オプションを選択して、導入に関連するイベントを表示します。

スポイラー

NDOレプリカセット(RS)の確認

スポイラー

####これはROOTユーザ#####でのみ使用できます。

レプリカ・セット(RS)は、安定した数のレプリカ・ポッドを維持することを目的としたK8sオブジェクトです。また、このオブジェクトは、ポッドへの定期的なプローブで異常な数のレプリカが検出されたときも検出します。

RSは名前空間でも構成されています。

```
[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
```

「 describe オプションには、URLに関する情報、プローブが使用するポート、およびテストと障害のしきい値の周期性が含まれます。

```
[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レプリカセット(RS)レビュー####これはROOTユーザーに対してのみ使用できます#####レプリカセット(RS)は、安定した数のレプリカPodを維持することを目的としたK8sオブジェクトです。また、このオブジェクトは、ポッドへの定期的なプローブで異常な数のレプリカが検出されたときも検出します。 RSは名前空間でも構成されています。[root@MxNDsh01 ~]# kubectl get rs -n cisco-msoNAME DESIRED CURRENT READY AGEauditservice-648cd4c6f8 1 3d22hbackupservice-64b755b44c 1 1 3d22hcloudsecservice-7df465576 1 1 3d22hconsistencyservice-c98955599 1 1 3d22hcnmworker-5d4d5cbb64 1 3d22eehez worker-56f9fb9ddb 1 1 1 3d22hendpointservice-7df9d5599c 1 1 3d22hexecutionservice-58ff89595f 1 1 1 3d22hfluentd-86785f89bd 1 1 3d22himportservice-88bcc8547 1 1 3d22hjobschedulerservice-5d4fdfd696 1 3d22hnofyfnoportservice service-75c988cfd4 1 1 3d22hpctagvnidservice-644b755596 1 1 1 3d22hplatformservice-65cddb946f 1 1 1 3d22hplatformservice2-6796576659 1 1 3d22hpolicyservice-545b9c7d9c 1 1 3d22hschemaservice-7597ff4c5 1 1 1 3d22hsdaservice-5ff477dd8c7 1 1 3d22hsdwanservice-6f87cd999d 1 1 3d22hsiteservice-86bb756585 1 1 3d22hsiteupgrade-7d578f9b6d 1 1 3d22hsyncengine-5b8bdd6b45 1 1 3d22htemplateeng-5cbf9fdc48 1 1 3d22hui-84588b7c 96 1 1 3d22huserservice-87846f7c6 1 1 1 3d22h describeオプションには、URLに関する情報、プローブが使用するポート、およびテストと障害のしきい値の周期性が含まれます。[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 8s-app=schemaservice memory.resource.case.cncf.io/schemaservice=mem-xlg-service pod-template-hash=7597ff4c5Annotations: deployment.kubernetes.io/desired-replicas: deployment.kubernetes.io/max-replicas: 1: 1Controlled by: Deployment/schemaservice: deployment.kubernetes.io/revisionレプリカ： 1現在/ 1望ましいポッドステータス： 1実行中/ 0待機中/ 0成功/ 0失敗ポッドテンプレート：ラベル： 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サービスアカウント： cisco-mso-sa Init Containers: init-msc: Image: cisco-mso/tools:3.7.1j Port: <none>ホストポート： <none>コマンド： /check_mongo.sh Environment: <none>マウント： /secrets from infracts (rw) Containers: schemaservice: image: cisco-mso/schemaservice:3.7.1j Ports: 8080/TCP, 8080/UDP Host Ports: 0/TCP, 0/UDP Command:

/launchscala.sh http://:8080/api/v1/schemas/health /wrapper.sh schemaservice Liveness: http-get-delay=300s timeout=20s period=30s #success=1 #failure=3環境： JAVA_OPTS: -XX:+IdleTuningGcOnIdle Mounts: /jwsecrets from jwsecrets (jwsecrets) 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コマンド：

NDOポッドレビュー

Podは、同じLinux名前空間（K8s名前空間とは異なる）および同じK8sノードで実行される、密接に関連したコンテナのグループです。これはコンテナと相互作用しないので、K8sが処理する最もアトミックオブジェクトです。アプリケーションは、単一のコンテナで構成することも、複数のコンテナを使用してより複雑にすることもできます。次のコマンドを使用して、任意のネームスペースのポッドを確認できます。

```
[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 ~]$
```

2列目の数値は、各ポッドのコンテナの数を示しています。

「 **describe** オプションも利用できます。これには、各ポッドのコンテナに関する詳細情報が含まれます。

```
[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
```

表示される情報には、各コンテナのコンテナイメージが含まれ、使用されるコンテナランタイムが表示されます。この場合、CRI-O(cri-o)、以前のバージョンのNDはDockerで動作していましたが、これはコンテナにアタッチする方法に影響を与えます。

スポイラー

たとえば、次の場合です cri-o を使用し、インタラクティブセッションでコンテナに接続する場合は、 exec -it オプション)を使用して、前の出力のコンテナに追加します。 docker crictlコマンドを使用します。

```
schemaservice:

  Container ID:  cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac

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

次のコマンドを使用します。

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

root@schemaservice-7597ff4c5-w4x5d:/#

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

root
```

それ以降のNDリリースでは、使用するコンテナIDが異なります。最初に、コマンドを使用する必要があります crictl ps をクリックして、各ノードで実行されているすべてのコンテナを一覧表示します。必要に応じて結果をフィルタリングできます。

```
[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 ~]#
```

最初の列の値を使用して、前と同じコマンドでコンテナのランタイムにアクセスできます。

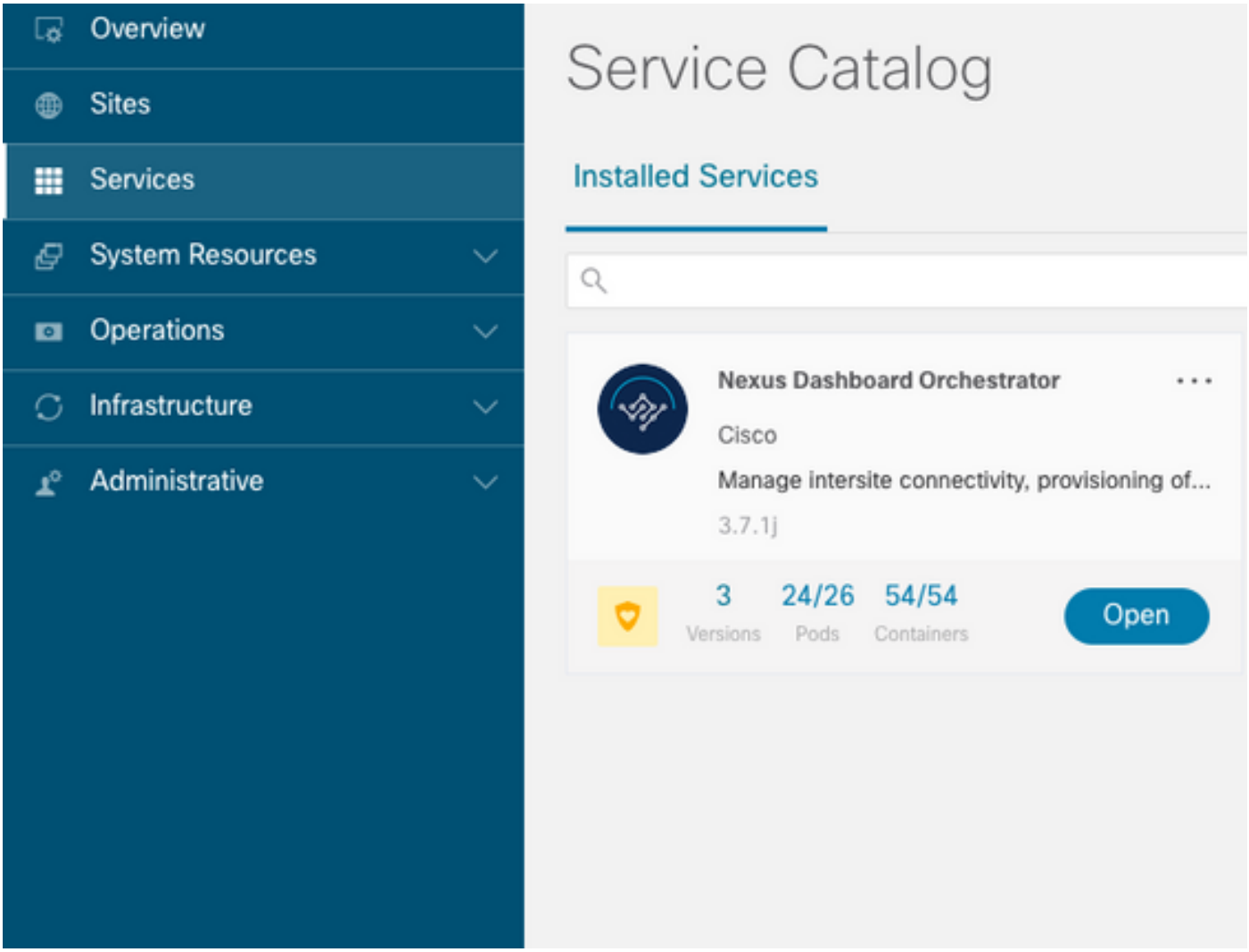
```
[root@singleNode ~]# crictl exec -it 4b26f67fc10cf bash
root@backupservice-8c699779f-j9jtr:/# pwd
/
```

たとえば、cri-oを使用し、対話型セッションで（exec -itオプションを使用して）以前の出力からコンテナに接続する場合、dockerコマンドの代わりにcricltコマンドを使用します。schemaservice:Container ID:cri-o://d2287f8659dec6848c0100b7d24aeebd506f3f77af660238ca0c9c7e8946f4ac　Image: cisco-mso/schemaservice:3.7.1j次のコマンドを使用します。[root@MxNDsh01 ~]# crictl exec -it d2287f859dec0100b7d2 4aeebd506f3f77af660238ca0c9c7e8946f4ac bashroot@schemaservice-7597ff4c5-w4x5d:/#root@schemaservice-7597ff4c5-w4x5d:/# whoamiroot以降のNDリリースでは、使用されるコンテナIDが異なります。最初に、コマンドcrictl psを使用して、各ノードで実行されるすべてのコンテナをリストする必要があります。必要に応じて結果をフィルタリングできます。[root@singleNode ~]# crictl ps| grep backupa9bb161d67295 10.31.125.241:30012/cisco-mso/sslcontainer@sha256:26581eebd0bd6f4378a5fe4a98973dbda417c1905689f71f229765621f0cee75 2日前にmsc-backupservice-ssl 0 84b3c691cfc2b4b26f67fc10cf 10.31.125.241:30012/cisco-mso/backupservice@sha256:c21f4cdde696a5f2dfa7bb910b7278fc3fb4d46b02f42c3554f872ca8c87c061 2日前にRunning backupservice 0 84b3c691cfc2b[root@singleNode ~]#最初の列の値を使用して、前と同じコマンドでコンテナのランタイムにアクセスできます。 [root@singleNode ~]# crictl -it b26f67fc10cf bashroot@backupservice-8c699779f-j9jtr:/# pwd/

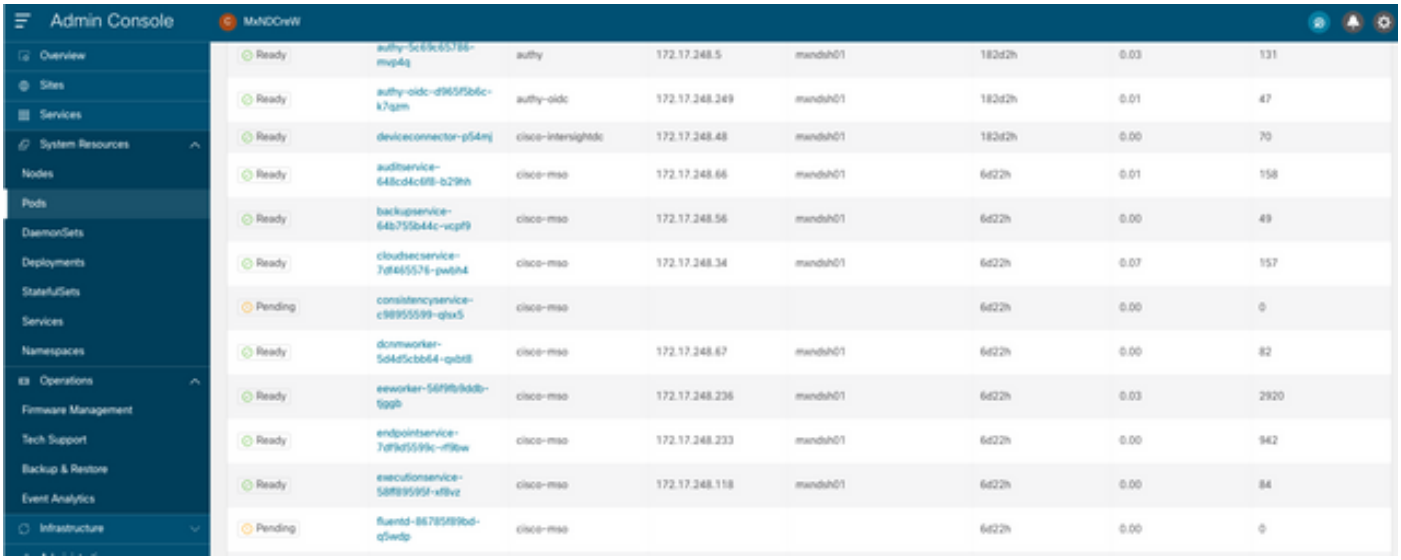
ユースケースPodが正常でない

この情報を使用して、導入のポッドが正常でない理由をトラブルシューティングできます。この例では、Nexus Dashboardのバージョンは2.2-1dで、影響を受けるアプリケーションはNexus Dashboard Orchestrator(NDO)です。

NDO GUIの[Service]ビューに、不完全なポッドのセットが表示されます。この例では、26個のポッドのうち24個です。



別のビューは、 **System Resources -> Pods** ポッドが示すステータスが異なる場所の表示 **Ready**.



異常なポッドのCLIトラブルシューティング

既知の事実では、名前空間はcisco-msoです（ただし、トラブルシューティングを行うと、他のアプリケーションや名前空間でも同じです）。Podビューは、異常な名前空間がある場合に表示されます。

```
[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
```

この例では、一貫したサービスポッドに焦点を当てます。JSON出力から、jsonpathを使用してステータスフィールドから特定の情報を取得できます。

```
[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 ~]$
```

statusディクショナリが表示されます。このリストの中のconditionsというディクショナリには、messageキーとvalueキーを持つ項目が含まれています。このリストの{"\n"}という部分は、最後に新しい行を作成することです。

```
[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 ~]$
```

次のコマンドは、 **get Pod** 名前空間の場合：

```
[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
```

Cisco Unified Computing System get pods コマンドを使用して、前の出力の問題と一致する必要がある問題を含むポッドIDを取得できます。この例の場合は **consistencyservice-c98955599-qlsx5.**

JSON出力形式は、与えられた出力から特定の情報をチェックする方法も提供します。

```
[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出力には、同じ名前の属性のステータスに関する情報が含まれている必要があります。メッセージには理由に関する情報が含まれています。

```
[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 ~]$
```

ポッドのステータスと要件に関する情報にアクセスできます。

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

ここでは、値の計算方法について説明することが重要です。この例では、cpu 500mは500ミリコアを指し、メモリ内の1GはGBを表します。

「 Describe ノードのオプションには、クラスタ（ホストまたはVM）内の各K8sワーカーに使用可能なリソースが表示されます。

```
[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] セクションには、各ノードで使用可能なCPU、メモリ、およびストレージ内のリソースの合計が表示されます。[Allocated] セクションには、すでに使用されているリソースが表示されます。CPUの値13は、13コアまたは13,000(13K)ミリコアを示します。

この例では、ノードがoversubscribedであり、これがポッドが開始できない理由を説明しています。NDをクリアした後、ND APPを削除するか、VMリソースを追加します。

クラスタは常に保留中のポリシーの展開を試みるため、リソースが空いている場合はポッドを展開できます。

```
[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
```

リソースチェックに使用するコマンドを使用して、クラスタにCPU用の使用可能なリソースがあることを確認します。

```
[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 ~]$
```

導入の詳細には、ポッドの現在の状態に関する情報を含むメッセージが含まれます。

```
[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 ~]$
```

スポイラー

コンテナ内からnetwork debugコマンドを実行する方法

コンテナにはPodに固有の最小限のライブラリと依存関係しか含まれていないため、ほとんどのネットワークデバッグツール(ping、ip route、ip addr)はコンテナ自体の内部では使用できません。

これらのコマンドは、サービス（NDノード間）またはApicへの接続に関するネットワーク問題をトラブルシューティングする必要がある場合に非常に便利です。複数のマイクロサービスがデータインターフェイス(**bond0**または**bond0br**)を使用してコントローラと通信する必要があるためです。

「 **nsenter** ユーティリティ（rootユーザのみ）を使用すると、コンテナ内にあるNDノードからネットワークコマンドを実行できます。このために、デバッグするコンテナからプロセスID(PID)を見つけます。これは、旧バージョンのDockerなど、コンテナランタイムからのローカル情報に対するPod K8s IDによって実現されます。 **cri-o** 新しいバージョンの場合はデフォルトです。

Pod Kubernetes(K8s)IDの検査

cisco-mso名前空間内のポッドのリストから、トラブルシューティングするコンテナを選択できます。

```
[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
```

ポッドは同じK8sノードで実行する必要があります。実稼働環境には、 **-o wide** 各ポッドが実行するノードを確認するオプションを最後に追加します。Pod K8s ID（前の出力例で太字で示されています）を使用して、コンテナランタイムによって割り当てられたプロセス(PID)を確認できます。

コンテナランタイムからPIDを検査する方法

新しいデフォルトのコンテナランタイムは、KubernetesのCRI-Oです。したがって、このドキュメントはコマンドに関するルールの後に続きます。CRI-Oによって割り当てられるプロセスID(PID)は、K8sノード内で一意にすることができます。これは、 crictl 使用できます。

「 **ps** このオプションは、CRI-Oによってポッドを構築する各コンテナに付与されたIDを表示します。サイトサービスの例では、2つのIDが表示されます。

```
[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 ~]#
```

この情報を使用して、 inspect CRIO-ID 各コンテナに与えられた実際のPIDを表示するオプション。この情報は、 nsenter コマンドにより、WLC CLI で明確に示されます。

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

コンテナ内でnetwork debugコマンドを実行するためのセンサーの使用方法

上記の出力のPIDを使用すると、次のコマンド構文でターゲットとして使用できます。

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

「 **--net** オプションを使用すると、ネットワークの名前空間でコマンドを実行できるため、使用可能なコマンドの数が制限されます。

以下に、いくつかの例を示します。

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

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

pingも使用でき、K8sノードだけでなく、コンテナから外部への接続をテストします。

```
[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]
```

コンテナ内からnetwork debugコマンドを実行する方法コンテナにはPodに固有の最小限のライブラリと依存関係しか含まれていないため、ほとんどのネットワークデバッグツール(ping、ip route、ip addr)はコンテナ内では使用できません。 これらのコマンドは、サービス（NDノード間）またはApicへの接続に関するネットワーク問題をトラブルシューティングする必要がある場合に非常に便利です。複数のマイクロサービスがデータインターフェイス（bond0またはbond0br）を使用してコントローラと通信する必要があるためです。 Enterユーティリティ（rootユーザのみ）を使用すると、コンテナ内にあるNDノードからネットワークコマンドを実行できます。このために、デバッグするコンテナからプロセスID(PID)を見つけます。これは、旧バージョンのDockerや新バージョンのcri-oなど、コンテナランタイムからのローカル情報に対するPod K8s IDによって実現されます。 Pod Kubernetes(K8s)IDの検査cisco-mso名前空間内のポッドのリストから、トラブルシューティングするコンテナを選択できます。 [root@MxNDsh01 ~]# kubectl get pod -n cisco-msoNAME READY STATUS RESTARTS AGEconsistencyservice-569bdf5969-xkwpg 3/3 Running 0 9heeworker-65dc5dd849-485tq 2 Running 163mendpointservice-5tservice-5 db6f57884-hkf5g 2/2 Running 0 9hexecutionservice-6c4894d4f7-p8fzk 2/2 Running 0 9hsiteservice-64dfcdf658-lvbr4 3/3 Running 0 9hsiteupgrade-68bcf987cc-ttn7h 2/2 Running 0 9h Podsは同じK8sノードで実行する必要があります。実稼働環境の場合は、各ポッドが実行するノードを確認するために、末尾に – o wideオプションを追加できます。Pod K8s ID（前の出力例で太字で示されています）を使用して、コンテナランタイムによって割り当てられたプロセス(PID)を確認できます。 コンテナランタイムからPIDを検査する方法新しいデフォルトのコンテナランタイムは、Kubernetes用のCRI-Oです。したがって、このドキュメントはコマンドに関するルールの後に続きます。CRI-Oによって割り当てられるプロセスID(PID)は、K8sノード内で一意にすることができます。このノードはcrictlユーティリティで検出できます。 psオプションは、CRI-OがPodを構築する各コンテナに与えたIDを明らかにします。2つはsiteserviceの例です。[root@MxNDsh01 ~]# crictl ps |grep siteservicefb560763b06f2 172.31.0.0:30012/cisco-mso/sslcontainer@sha256:2d788fa493c885ba8c9e5944596b864d090d9051b0eab82123ee4d19596279c9 10時間前msc-siteservice2-ssl 0 074727b4e9f51ad2d42aae1ad9 1d0195292f7fcc62f38529e135a1315c358067004a086cfed7e059986ce615b0 10時間前 siteservice-leader-election 0 074727b4e9f512を0時間前b0b6d41d1e3 172.31.0.0:30012/cisco-

mso/siteservice@sha256:80a2335bcd5366952b4d60a275b20c70de0bb65a47bf8ae6d988f07b1e0bf494 10時間前siteserviceの実行0 074727b4e9f51[root@MxNDsh01 ~]#この情報を基に、inspect CRIO-IDオプションを使用して、各コンテナに指定された実際のPIDを確認できます。この情報は、nsenterコマンドに必要です。[root@MxNDsh01 ~]# crictl inspect fb560763b06f2| grep -i pid"pid": 239563,"pid": {"type": "pid"コンテナ内でnetwork Debugコマンドを実行する方法コンテナ内でNsenterを使用する方法上記の出力のPIDを使用すると、次のコマンド構文のターゲットとして使用できます。例：[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:32:72:7b:fb txqueuelen 0 (Ethernet)RX packets 916346 bytes 271080553 (258.5 MiB)RX errors 0 dropped 183 overruns 0 frame 0TX packets 828016 bytes 307255950 (293.0 MiB)TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536inet 127.0.1 0inet6 ::1 prefixlen 128 scopeid 0x10<host>loop txqueuelen 1000 (Local Loopback)RXパケット42289バイト14186082 (13.5 MiB)RXエラー0ドロップ0オーバーラン0フレーム0TXパケット42289バイト14186082 (13.5 MiB)TXエラー0ドロップ0オーバーラン0キャリア0コリジョン0pingも使用でき、K8sノードだけでなく、コンテナから外部への接続をテストします。[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 t requested name '1xx .2xx.3xx.4xx'.HTTP要求が送信され、応答を待っています… 200 OK長： 3251 (3.2K) [text/html]保存先：
'index.html'100%[==========>] 3,251 —.-K/s in 0s2023-01-24 23:46:04 (548 MB/s) - 'index.html' saved [325] 1]

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。