CMM-D (CXL Memory) 활용가이드

For Samsung SMRC

Table of Contents

1. 본 문서에 대하여

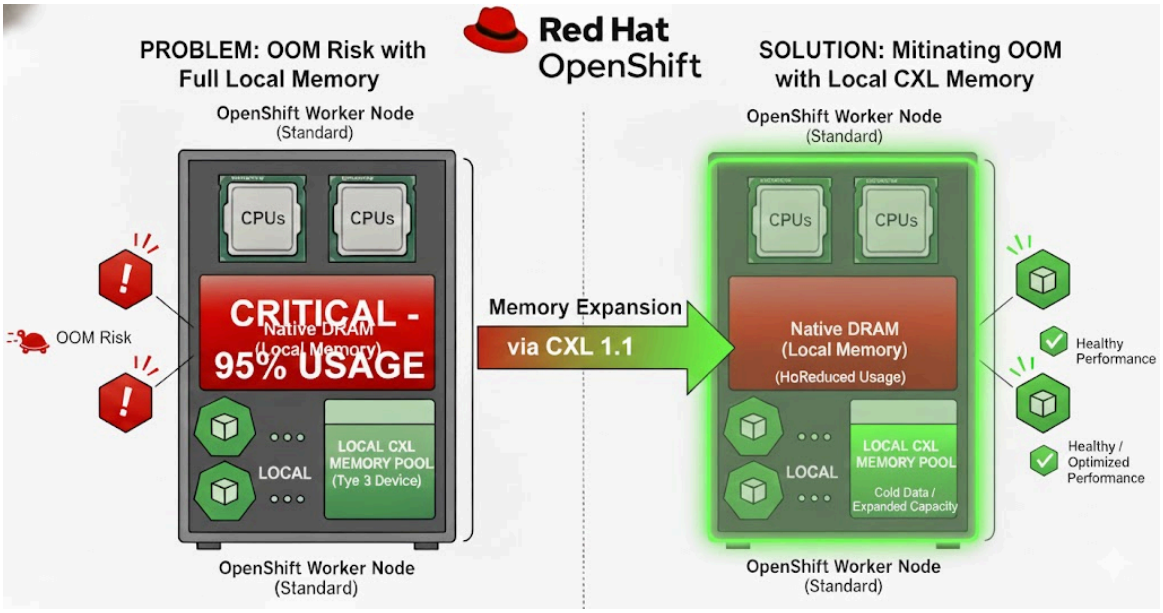
본 문서는 삼성전자 **CMM-D** 를 이용하여 Baremetal 환경의 AMD CPU 또는 INTEL CPU의 Red Hat OpenShift Enterprise Application Platform 환경에서 CMM-D 메모리 정보, 구성, 기능, 사용법 등을 확인하는 목적으로 구성하는 방법을 가이드하기 위해 작성된 문서입니다.

본 가이드는 삼성전자 **SMRC** 에서 제공되는 하드웨어 및 네트워크 환경에서 검증되었습니다.

2. 시스템 아키텍처

2.1 CXL 개요 및 환경구성 아키텍처

가상머신 및 컨테이너 기반의 클라우드 환경이 증가함에 따라 응용프로그램에서 요구되는 메모리의 양이 급격히 늘어나고 있습니다. 또한, AI/ML 에서 사용되는 캐시메모리 등 새로운 기술이 등장함에 따라 메모리의 사용량도 비례하여 증가되고 있습니다. 그러나, CPU의 Core수가 증가하고 있음에도 불구하고 메모리 구성 컴포넌트는 확장성은 제한되고 있는 실정이어서 메모리 확장성의 한계에 직면하고 있습니다.



이를 개선하기 위한 솔루션인 삼성전자 **CMM-D** 는 최대 256GB의 DDR5 DRAM 메모리로 제공되어 서버 메모리 용량을 수십 테라바이트대로 확장하는 동시에 메모리 대역폭을 초당 몇 테라바이트대로 증가시킬 수 있으며, CMM-D Memory는 x8 PCIe 5.0 인터페이스를 사용해서 레인당 최대 32GT/s의 전송 속도로 CPU에 연결할 수 있습니다.

이번 활용가이드에서 검증한 [삼성전자 **CMM-D**] 를 활용한 환경구성 아키텍처는 AMD CPU 또는 INTEL CPU가 장착된 SuperMicro Baremetal System에 RHEL 10 OS 및 OpenShift 4.14, OpenShift Virtualization 4.18를 구성하여 Container와 VirtualMachine 환경에서의 CMM-D Memory 정보, 구성, 기능, 사용법 등을 확인하였으며, 삼성전자 CMM-D를 활용하여 Container, VirtualMachine에서 사용할 수 있는 기본적인 옵션을 설명합니다.

2.2 CXL 환경구성 정보

본 가이드의 기본환경은 Supermicro 시스템의 AMD, INTEL x86 CPU 기반으로 진행하였으며, 삼성전자 CMM-D의 Red Hat OpenShift Enterprise application platform (OCP, OV) 환경에서 인식여부, Container 및 VirtualMachine에 자원할당 등 다양한 시나리오로 진행을 하였습니다.

2.2.1 Baremetal Host information

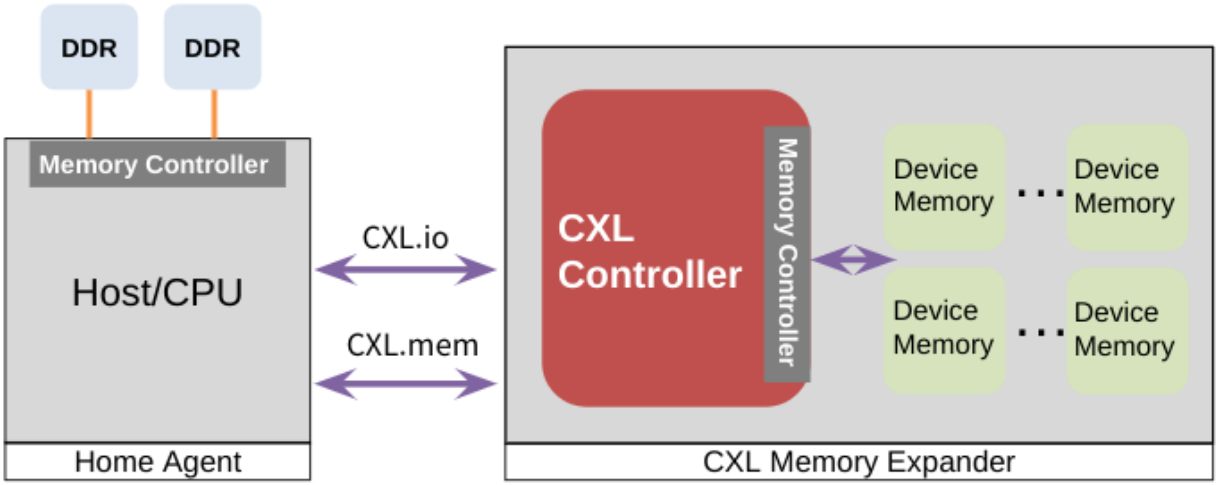
TYPE	Server Model	BIOS rev	CPU	Memory	Disk
AMD	Supermicro Super Server (Base Board : H13SSF)	07/07/2023	48 core 1 Socket	128 GB (1 NUMA)	894.3 GB
INTEL	Supermicro SSG-121E-NE316R (Base Board : X13DSF-A)	08/14/2023	24 core 2 Socket	128 GB (2 NUMA)	894.3 GB

2.2.2 CPU Model information

TYPE	CPU Model	OEM Strings
AMD	AMD EPYC 9454P 48-Core Processor	AMD EPYC Soc/Genoa Supermicro Motherboard-H13 Series
INTEL	Intel(R) Xeon(R) Gold 6442Y	Intel Sapphire Rapids/Emmitsburg/EagleStream Supermicro motherboard-X13 Series

2.2.3 Samsung CMM-D information

TYPE	CMM-D Model
AMD	CXL: Montage Technology Co., Ltd. Device c000
INTEL	CXL: Montage Technology Co., Ltd. Device c000



2.2.4 Red Hat Openshift Compatibility

삼성전자 CMM-D (1.1)는 INTEL, AMD CPU 환경에서 인식이 가능하며, OS Kernel의 CXL 지원여부를 사전에 파악해야 합니다. 아래는 검증을 위한 OCP 지원표입니다.

TYPE	OCP Version
AMD	Red Hat Openshift 4.14 (tested on 4.14.16) Red Hat Openshift Virtualization 4.18 or later(tested on 4.18)
Intel	Red Hat Openshift 4.14 (tested on 4.14.16) Red Hat Openshift Virtualization 4.18 or later(tested on 4.18)

3. AMD, INTEL CPU 환경 CXL 구성

삼성전자 CMM-D 장치는 Flexbus 레인을 통해 CXL 루트 포트 또는 CXL 스위치 다운스트림 포트에 연결될 수 있으며, CXL 장치는 표준 PCIe 메커니즘을 사용하여 MMCFG 및 MMIO 영역에 매핑됩니다. 이를 위해 사전 BIOS 설정작업은 중요하며, 아키텍처 별 설정이 필요합니다.

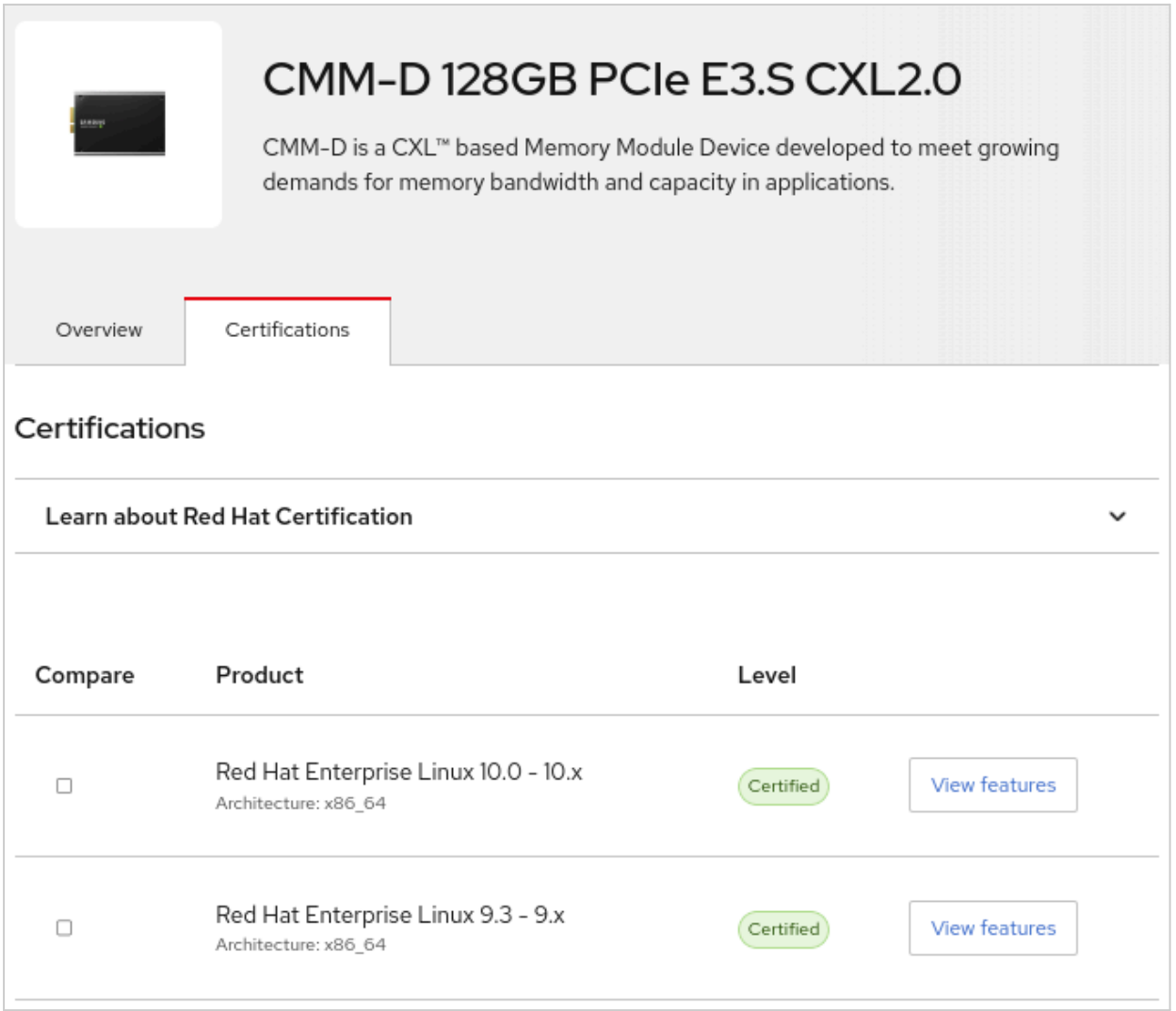
3.1 INTEL 및 AMD CPU 환경에 BIOS 구성

2024년에 공개된 RHEL 9 CXL 활용가이드 for Samsung SMRC 문서에 Supermicro 시스템의 AMD, INTEL BIOS 구성에 관하여 상세히 안내되어 있습니다.
해당 내용을 참고해서 사전에 H/W BIOS 구성이 필요합니다.

4. Red Hat Ecosystem (인증)

4.1 RHEL 9, RHEL 10

RHEL 9 및 RHEL 10 환경에서 삼성전자 CMM-D가 레드햇 인증을 완료했습니다.
삼성전자 CMM-D는 AI, 머신러닝, 클라우드 인프라, 고성능 컴퓨팅(HPC) 등에서 급증하는 메모리 대역폭과 용량 요구를 충족하기 위해 개발된 Compute Express Link™(CXL™) 기반 메모리 모듈입니다. 이번 인증을 통해 삼성전자 CMM-D가 다양한 IT 인프라 환경에 안정적으로 적용될 수 있는 기반이 마련되었으며, 앞으로도 신규 메모리 모듈 출시와 동시에 레드햇 솔루션과의 호환성을 지속적으로 확보해 나갈 계획입니다.



https://catalog.redhat.com/en/hardware/components/detail/253627#certifications

- **RHEL** 기반의 지원 **:** OpenShift Virtualization은 Red Hat Enterprise Linux(RHEL) 기반 위에서 운영됩니다. 특정 하드웨어 구성 요소가 RHEL에서 공식적으로 'CMM-D'로 인증되었다는 것은, 해당 장치가 RHEL 커널 수준에서 안정적으로 인식되고 지원된다는 의미입니다.

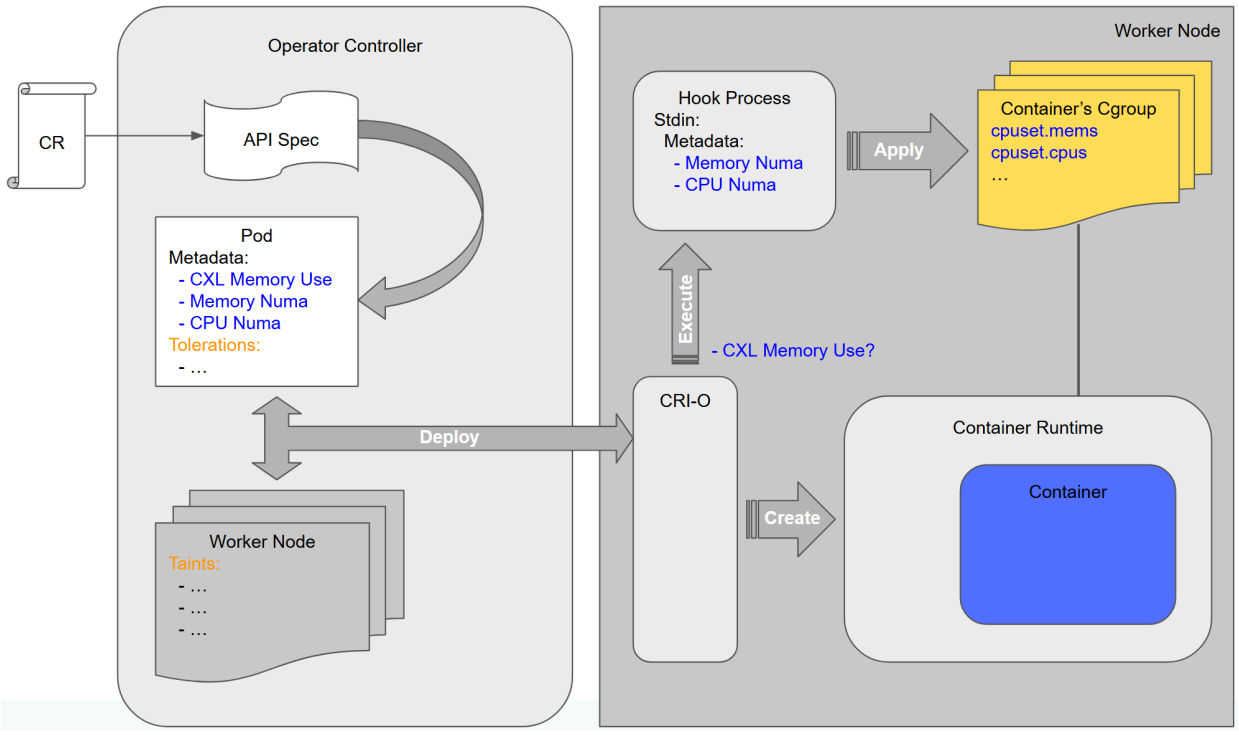
- **Openshift OV**로의 확장 **:** RHEL에서 삼성전자 CMM-D 장치에 대한 근본적인 하드웨어 및 메모리 티어링(Memory Tiering) 지원이 확립되었다면, OV를 구성하는 KubeVirt/KVM 하이퍼바이저 레이어는 이 CMM-D Memory 리소스를 활용할 수 있습니다. 즉, 호스트 OS(RHEL)의 지원이 OpenShift Virtualization 환경의 기반을 마련해 줍니다.

- 고성능 활용 가능성 **:** 제품 설명에는 RHEL의 메모리 티어링 기능과 삼성전자 CMM-D 를 함께 사용하여 성능이 중요한 데이터(Hot Memory)를 로컬 메모리에, 자주 사용되지 않는 데이터를 CXL 티어에 할당하여 메모리 활용을 극대화할 수 있습니다.

5. OpenShift CXL Operator

5.1 OpenShift CXL Operator (SOCMMD)

Samsung Operator for CMM-D는 Red Hat OpenShift 환경 내에서 서버의 로컬 메모리(DRAM)와 삼성전자 CMM-D 장치를 통합합니다. 오퍼레이터는 사용자가 필요에 따라 통합된 이기종 메모리를 효율적으로 활용할 수 있도록 지원합니다. 로컬 메모리 또는 확장된 삼성전자 CMM-D 리소스를 선택하고 할당하여 Red Hat OpenShift Container Platform의 제어 하에 안전하고 효율적인 사용을 보장합니다. 또한, CoreOS (커널)나 OpenShift를 수정할 필요가 없으므로 Red Hat의 OpenShift 지원에는 영향을 미치지 않습니다. 이 가이드는 삼성 오퍼레이터 for CMM-D를 사용하기 위한 하드웨어 및 소프트웨어 환경 구성에 대한 간략한 개요와 오퍼레이터 설치 및 사용 지침을 제공합니다.



[Red Hat OpenShift CXL Operator Flow]

5.1.1 Hardware Configuration

Node	H/W	Usage	CPU	Local Memory	CXL
node #1	Dell	KVM	Intel 96 Core	512 GB	NA
node #2	SMC	OCP Worker #1	Intel 96 Core	64 GB	512 GB(128 GB * 4ea)
node #3	SMC	OCP Worker #2	AMD 96 Core	128 GB	512 GB(128 GB * 4ea)

node #3	SMC	OCP Worker #3	AMD 96 Core	64 GB	512 GB(128 GB * 4ea)

5.1.2 Software Configuration

소프트웨어 버전 정보는 다음과 같습니다:

```
$ oc version
Client Version: 4.14.16
Kustomize Version: v5.0.1
Server Version: 4.14.16
Kubernetes Version: v1.27.10+c79e5e2

$ oc debug node/cmmd1
sh-4.4# chroot /host
sh-5.1# grep -e "OPENSHIFT_VERSION" -e "RHEL_VERSION" -e "^VERSION="
/etc/os-release
VERSION="414.92.202403051622-0"
OPENSHIFT_VERSION="4.14"
RHEL_VERSION="9.2"
```

5.1.3 Install Operator (SOCMMD)

5.1.3.1 CMM-D 환경 Samsung Operator 설정

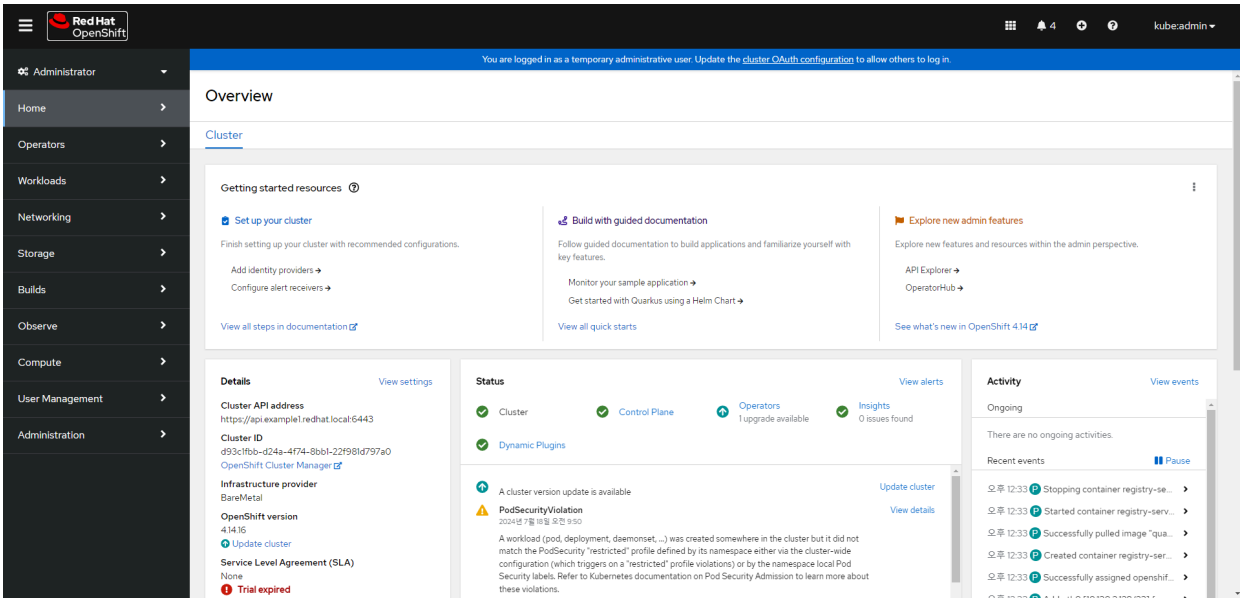
1) **Openshift** 클러스터 설치

CMM-D Operator를 테스트하려면 위의 하드웨어 및 소프트웨어 환경에 설명된 대로 CMM-D가 구성된 서버 환경에 OpenShift 클러스터를 설치해야 합니다. 자세한 설치 지침은 Red Hat OpenShift 설치 가이드를 참조하십시오.

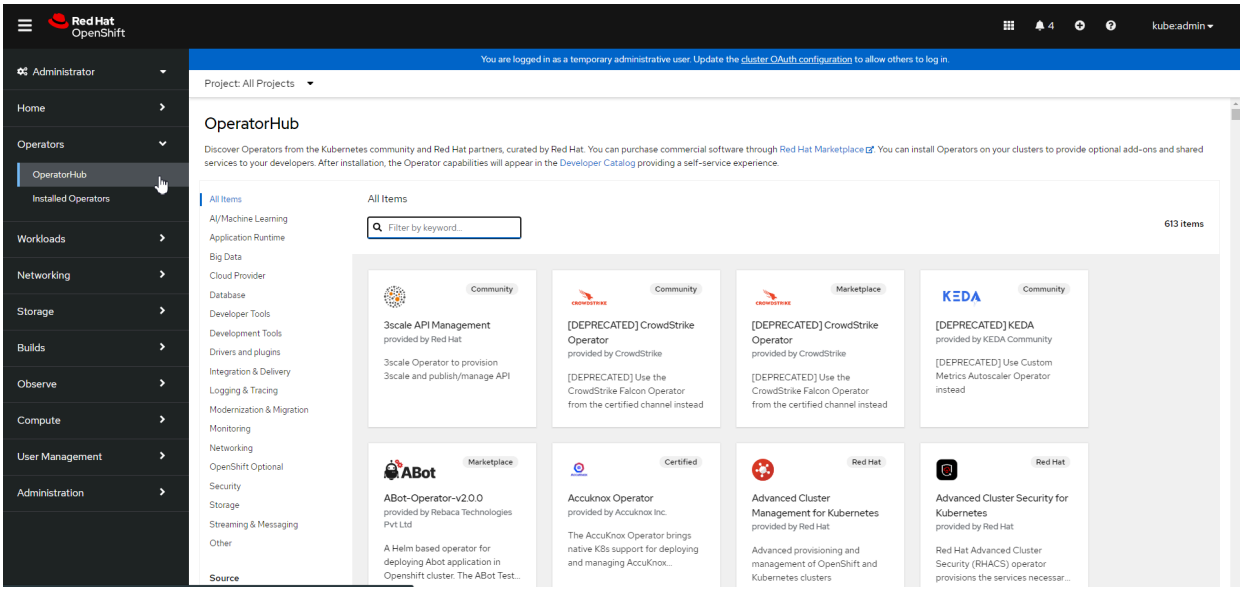
5.1.3.2 OCP 웹 콘솔을 통한 운영자 설치

1) **Operator** 검색

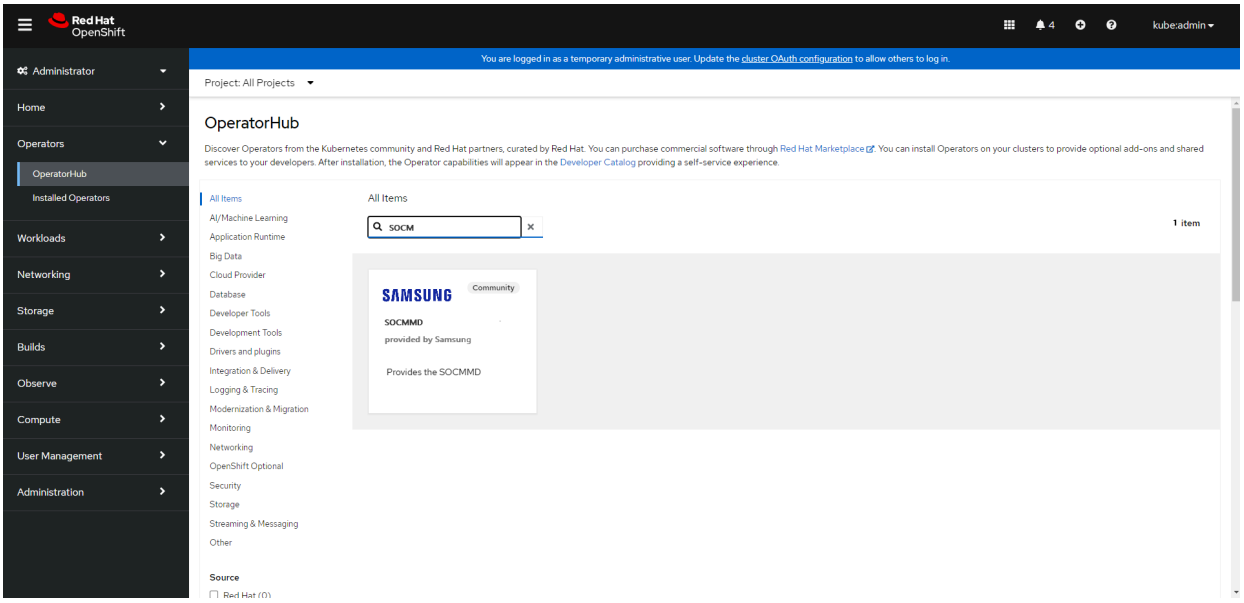
아래 이미지는 Red Hat OCP 웹 콘솔에 접속했을 때의 첫 번째 화면을 보여줍니다.



아래 이미지는 Red Hat OCP 웹 콘솔에서 Operator를 검색하기 위해 OperatorHub 메뉴를 클릭하면 나타나는 화면을 보여줍니다.

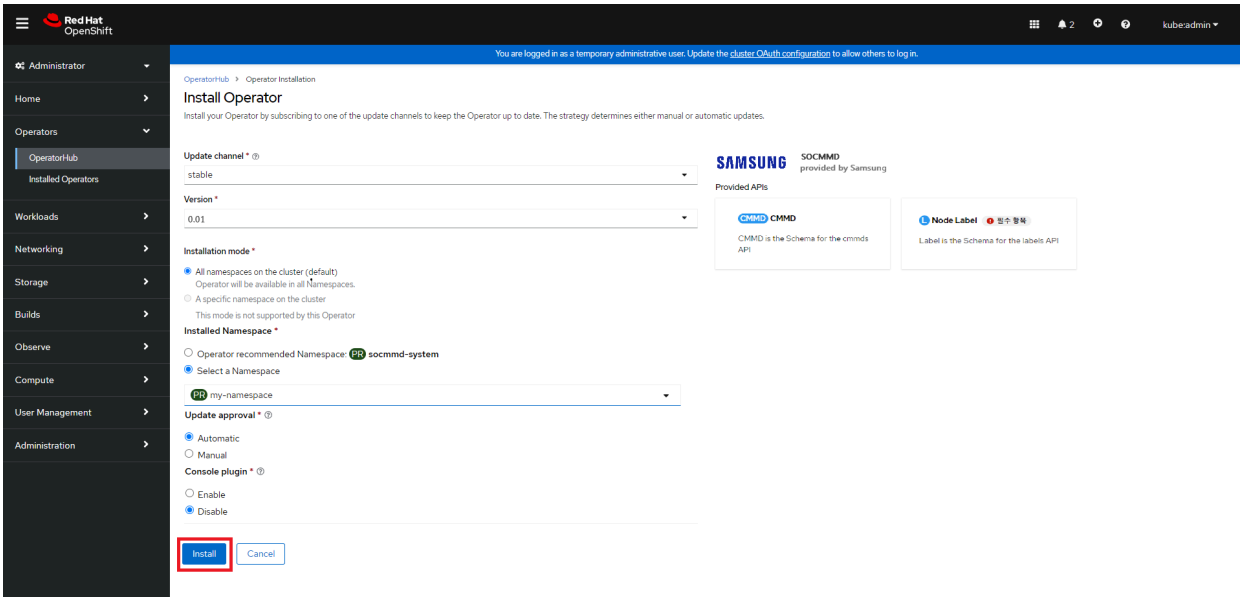


아래 이미지는 CMM-D용 Samsung Operator를 검색하는 화면입니다. 검색어를 입력하면 일치하는 Operator가 표시됩니다.



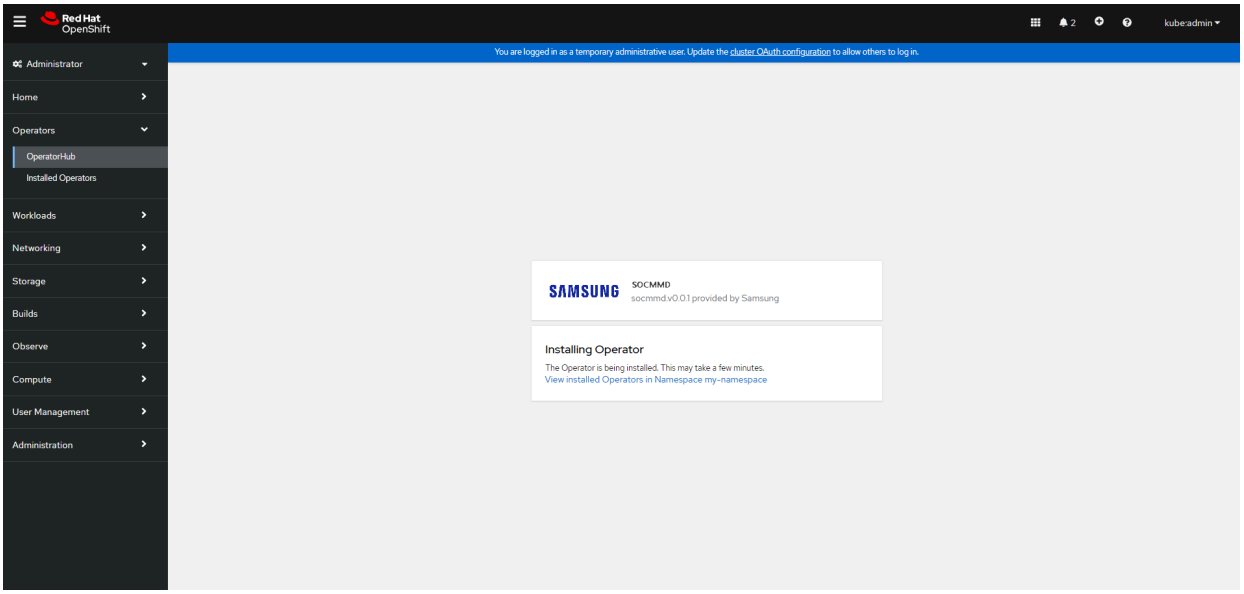
2) Operator 선택 및 설치 옵션 지정

아래 이미지는 1단계에서 찾은 Operator를 클릭하면 나타나는 설치 화면을 보여줍니다. 필요한 설치 옵션을 선택한 후, '설치' 버튼을 클릭하여 계속 진행하세요.

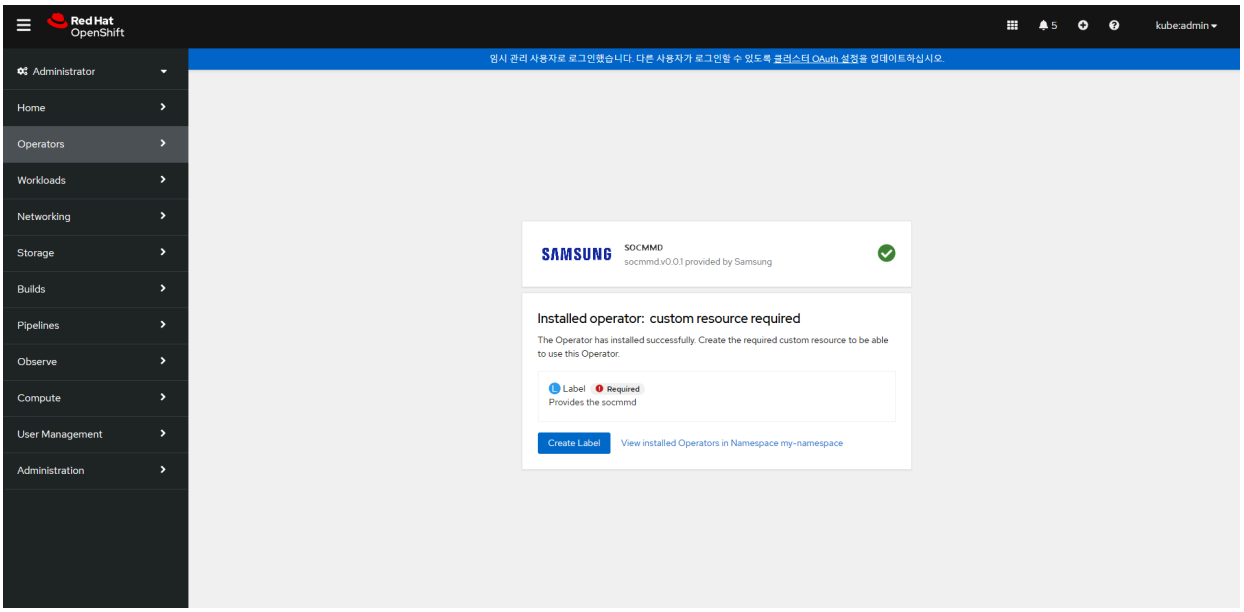


3) Operator 설치

아래 이미지는 이전 이미지에서 보여준 것과 같이 '설치' 버튼을 클릭하면 나타나는 설치 화면을 보여줍니다.



아래 이미지는 Operator 설치가 완료된 후 표시되는 화면을 보여줍니다.



4) 운영자를 설치한 후 필수 단계로 노드 레이블을 설정합니다.

CXL(Compute Express Link) 메모리 확장 장치인 CMM-D를 OpenShift 클러스터에 성공적으로 통합하고 활용하기 위해서는 Machine Config Pool (MCP)을 통한 호스트 소프트웨어 구성 관리가 필수적입니다. 이 과정은 CMM-D 리소스를 운영자를 통해 배포된 Pod나 가상 머신(VM)에서 접근 가능하도록 준비하는 핵심 단계입니다.

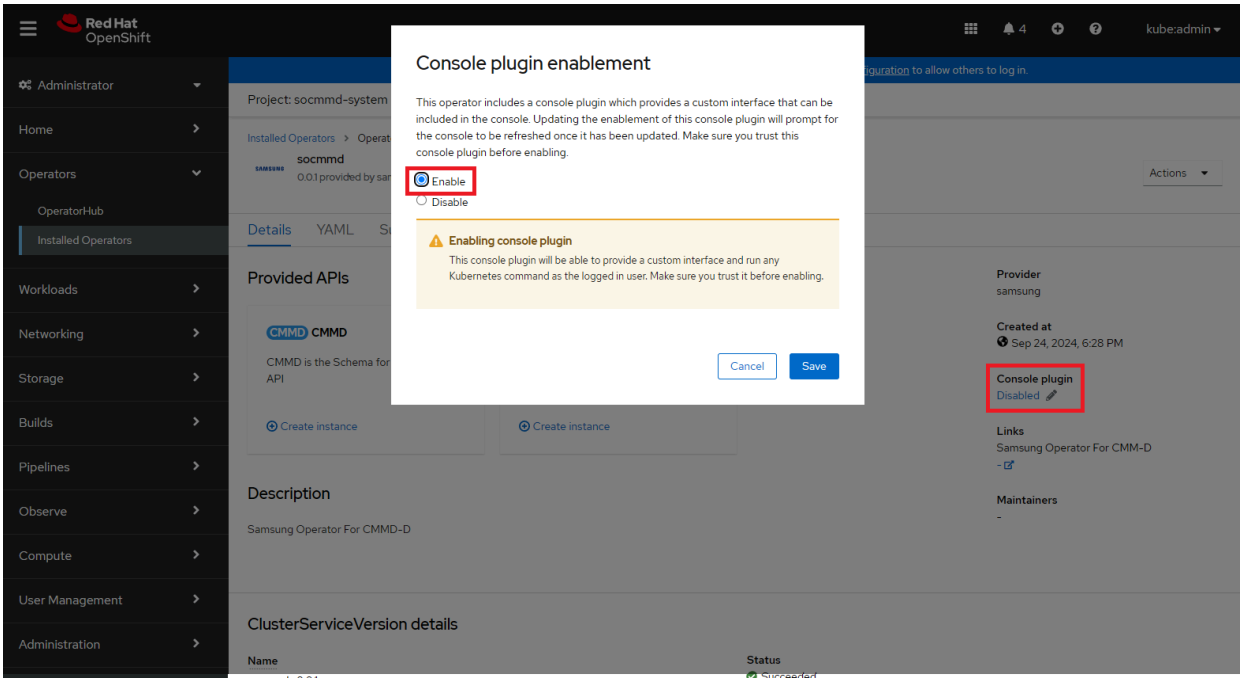
1. Machine Config Pool (MCP)을 통한 소프트웨어 구성 관리

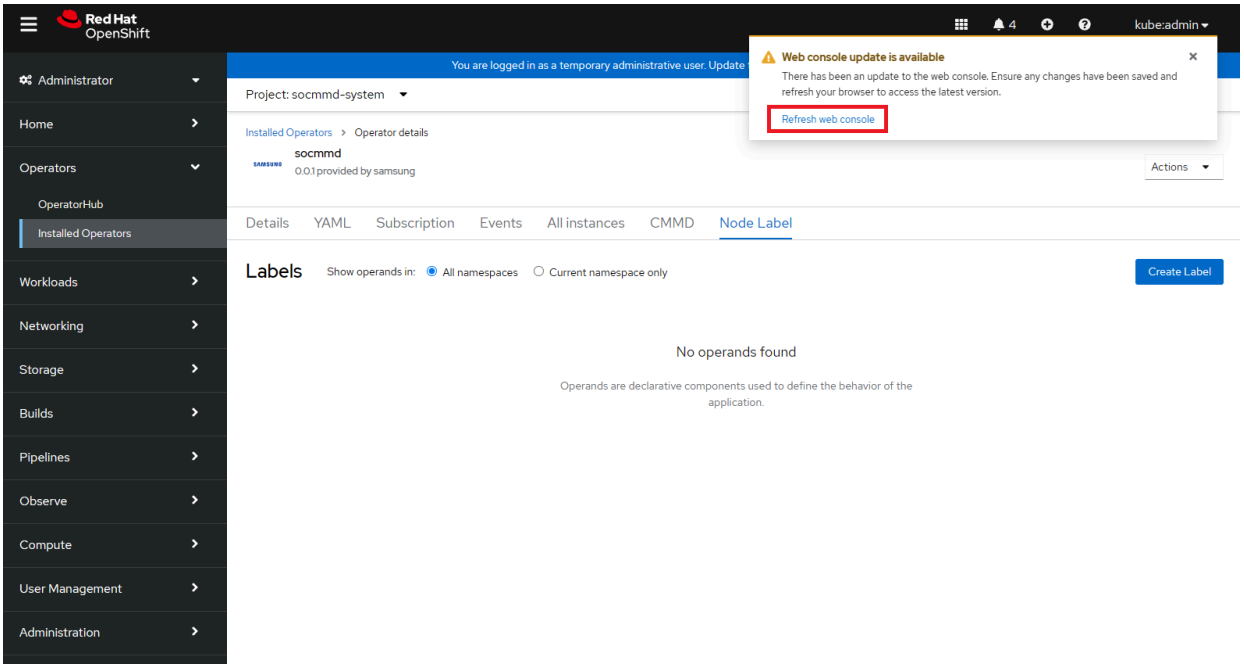
OpenShift 웹 콘솔에서는 노드를 그룹화하기 위해 '레이블'을 사용하는 것처럼 보이지만, 실제로 특정 하드웨어 구성(여기서는 CMM-D 지원)에 필요한 소프트웨어 변경 사항을 관리하고 적용하는 주체는 Machine Config Pool (MCP)입니다. MCP는 해당 풀에 속한 워커 노드의 소프트웨어 구성을 정의합니다. 따라서 사용자의 운영 환경이 기본 설정과 다르거나 CMM-D를 특정 노드 그룹에만 적용하려면, 해당 환경에 맞게 MCP 내용을 세부적으로 수정해야 합니다.

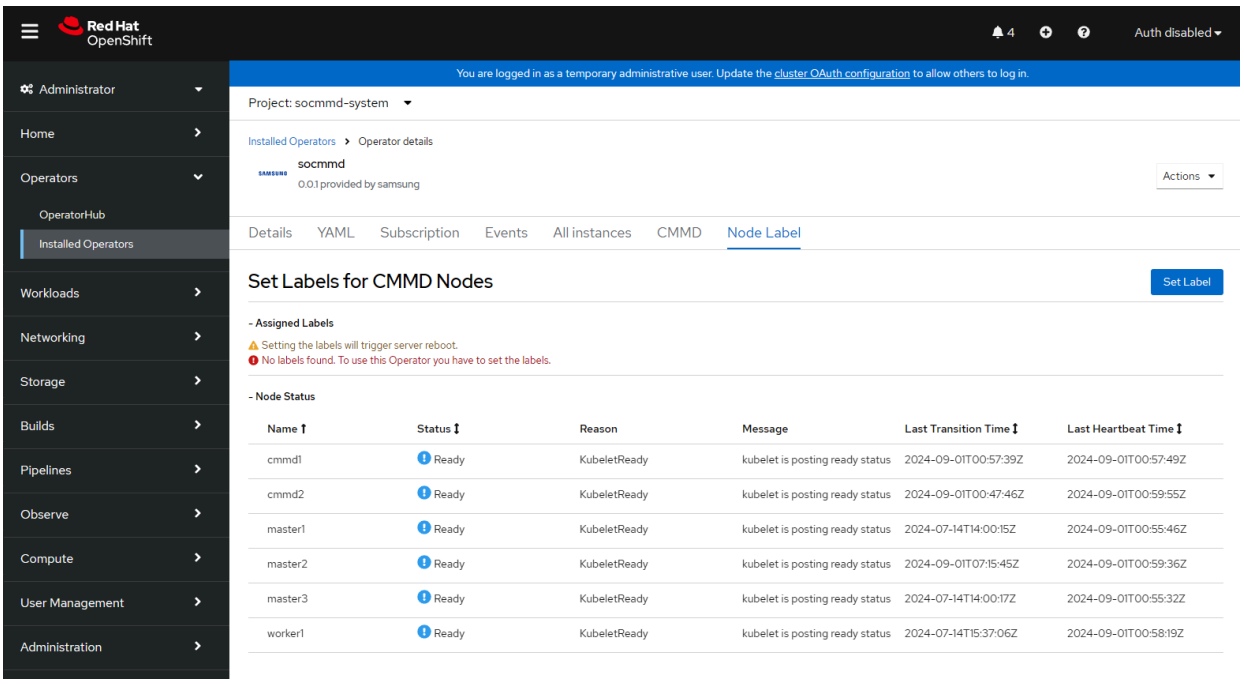
2. 대상 워커 노드 레이블링 및 **Machine Config** 적용
CMM-D가 장착된 워커 노드에 정확한 레이블을 입력해야 필요한 Machine Config가 해당 노드에 자동으로 적용됩니다. 콘솔에서 SetLabel 버튼을 클릭하여 레이블 입력 팝업이 나타나면, CMM-D가 장착된 노드를 관리할 MCP의 이름을 사용하여 다음과 같은 형식으로 레이블을 지정합니다
machineconfiguration.openshift.io/role=<CMM-D가 장착된 MCP 이름>. 예를 들어, CMM-D 전용 풀을 cxl-worker로 정의했다면 이 노드에 해당 레이블이 적용됩니다. 이 레이블이 설정되지 않고 Machine Config가 적용되지 않은 워커 노드에서는 운영자(Operator)를 통해 배포된 Pod가 CMM-D 리소스를 활용할 수 없게 됩니다.

3. 구성 적용, 재부팅 및 운영자 활성화
레이블 지정이 완료되면, Machine Config는 대상 워커 노드에 순차적으로 적용되기 시작하며, 이 과정에서 노드는 필수적으로 재부팅됩니다. CMM-D를 관리하는 운영자(Operator)가 제대로 작동하고 새로운 CMM-D Memory 리소스를 인식하여 활용하기 위해서는 대상 워커 노드 전체가 이 구성을 적용하고 재부팅되어야 합니다. 재부팅이 완료되면 CMM-D를 위한 소프트웨어 환경이 완전히 준비된 것입니다. 현재 시스템의 MCP 목록과 진행 상태는 oc get mcp 명령을 통해 확인할 수 있습니다.

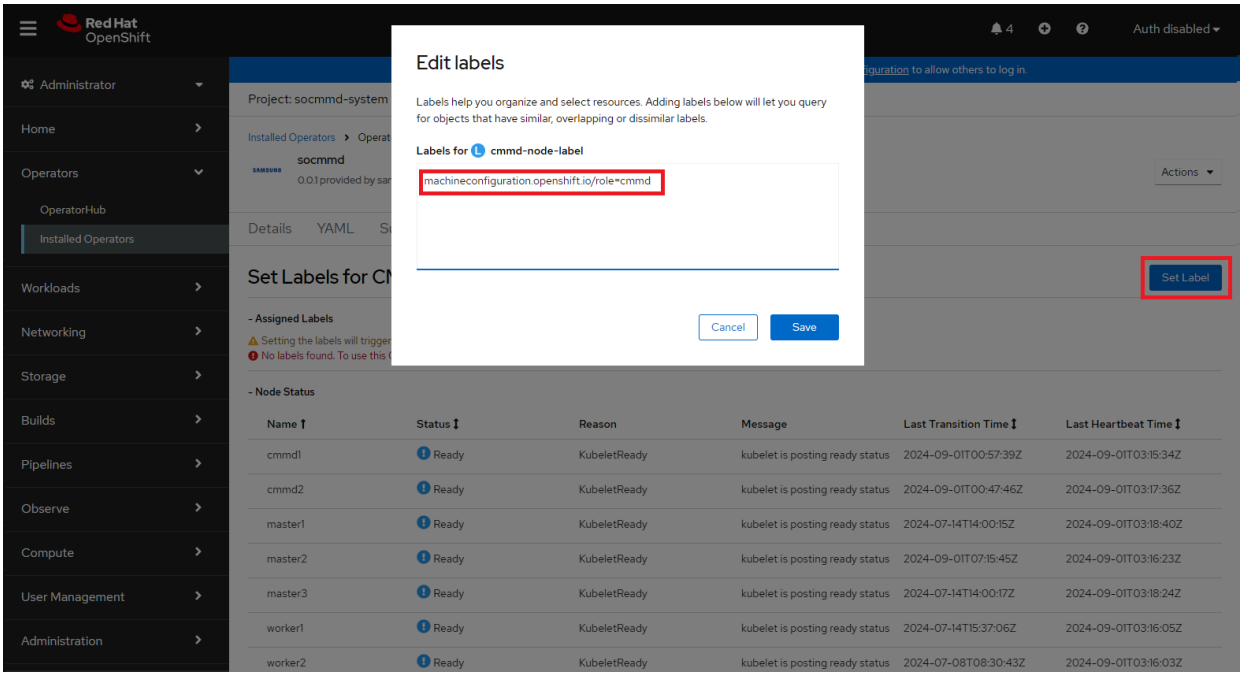
4. 웹 콘솔 플러그인 활성화 및 **UI** 확인
마지막 단계로, 사용자 편의를 위해 웹 콘솔에서 CMM-D 관련 기능을 시각적으로 확인할 수 있는 플러그인을 활성화합니다. 콘솔의 플러그인 항목을 클릭한 다음, 팝업 메뉴에서 사용(Enable)을 선택하고 저장합니다. 사용 설정 후 잠시 기다리면 새로 고침 알림이 팝업되는데, 웹 콘솔 새로 고침을 클릭하여 UI를 업데이트합니다. 화면이 새로 고쳐지면 CMM-D의 상태나 활용도를 나타내는 새로운 UI 요소가 추가되어 관리 및 모니터링이 용이해집니다.

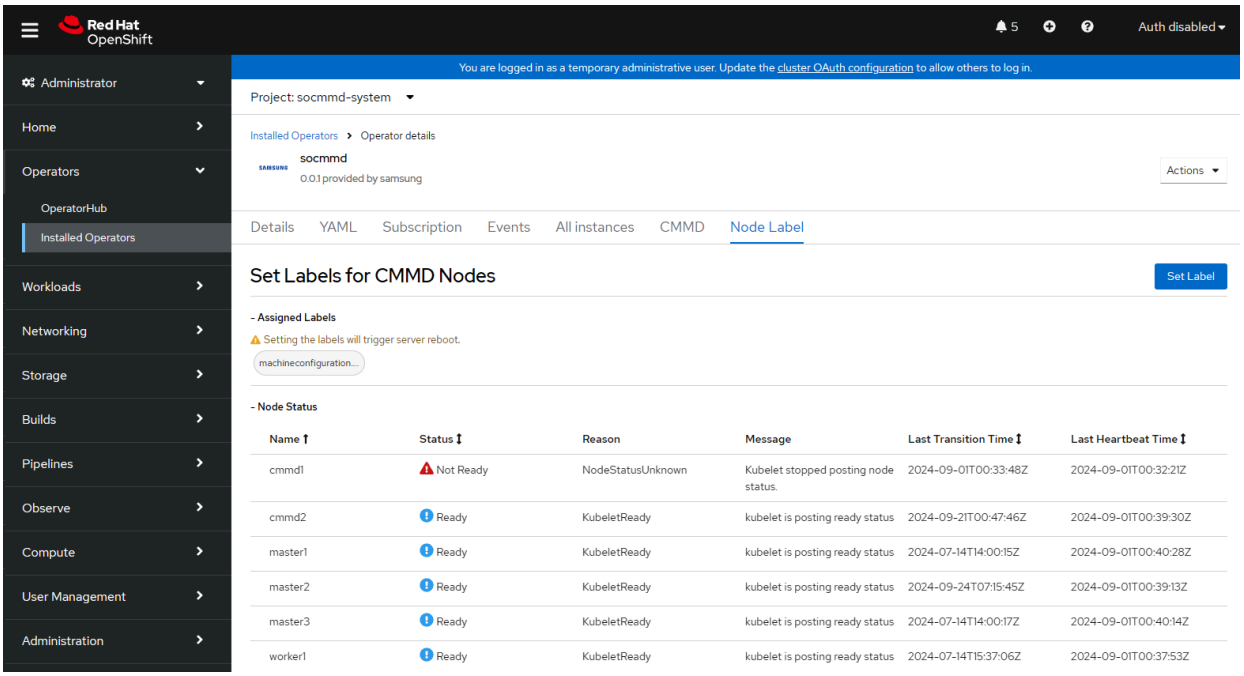

Red Hat OpenShift
kube:admin
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Pipelines
Observe
Compute
User Management
Administration
Project: socmmd-system
Installed Operators
socmmd
Actions
Details
YAML
Provided APIs
CMMD
Create instance
Description
Samsung Operator For CMMD-D
ClusterServiceVersion details
Name
Status
Provider
samsung
Created at
Sep 24, 2024, 6:28 PM
Console plugin
Disabled
Links
Samsung Operator For CMM-D
Maintainers
Console plugin enablement
This operator includes a console plugin which provides a custom interface that can be included in the console. Updating the enablement of this console plugin will prompt for the console to be refreshed once it has been updated. Make sure you trust this console plugin before enabling.
Enable
Disable
Enabling console plugin
This console plugin will be able to provide a custom interface and run any Kubernetes command as the logged in user. Make sure you trust it before enabling.
Cancel
Save






```
[kni@bastion ~]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED   MACHINECOUNT   REA
cmmd     rendered-cmmd-ab5eba95f41bb008c64f67c18ebc53d7     True      False      False      4              4
master   rendered-master-d428a812ed34921c5d0fe6b30250ddd8   True      False      False      3              3
worker   rendered-worker-f419a3437b805bdd4fba033a991769a2   True      False      False      2              2
```





5) 추가 작업 요소

운영자는 각 워커 노드에서 정보를 수집하기 위해 **"cmmdc"**라는 데몬 셋을 배포합니다.
데몬 셋이 제대로 배포되지 않으면 운영자가 제대로 작동하지 않을 수 있습니다.
워커 노드에 테인트가 설정된 경우 다음 프로세스를 통해 톨러레이션을 적용할 수 있습니다.
운영자와 동일한 네임스페이스에 **"cmmd-config"**라는 이름의 **ConfigMap**을 생성하고 적용할 톨러레이션을 **CMMD_ND_TOLERATIONS** 항목에 추가합니다.

아래 이미지는 예시를 보여줍니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cmmd-config
  namespace: <Same Namespace As The Opeartor>
data:
  CMMD_ND_TOLERATIONS: |
    - key: node-role.kubernetes.io/cmmd
      operator: Equal
      value: "true"
      effect: NoSchedule
    - key: node-role.kubernetes.io/cmmd
      operator: Equal
      value: "true"
      effect: NoExecute
```

5.1.4 Create Container (Application)

테스트를 위하여 앞서 설명한 환경을 준비하십시오.

- 지정된 하드웨어 및 소프트웨어 요구 사항을 충족하는 환경을 구축합니다.
- Openshift 클러스터 설치
- 웹 콘솔을 통해 Operator를 설치합니다.
- CRD(Custom Resource Definition) 설정

다음은 Samsung Operator for CMM-D를 사용하는 데 필요한 사용자 정의 리소스 정의(CRD)에 대한 설명과 CRD를 기반으로 사용자 정의 리소스(CR)를 제출하는 방법, 그리고 CR을 통해 생성된 Pod를 확인하고 삭제하는 방법을 두 가지 예를 통해 설명합니다.

1) CRD 설정

CRD(Custom Resource Definition)는 Operator를 통해 Pod를 생성하는 데 필수적입니다. 아래에서 이에 대해 설명합니다.

```
apiVersion: apiextensions.k8s.io/v1                <1>
 kind: CustomResourceDefinition                    <2>
 metadata:
     annotations:
         controller-gen.kubebuilder.io/version: v0.11.1
     name: cmmds.cmmd.samsung.com
 spec:
     group: cmmd.samsung.com
```

```
        names:
            kind: CMMD
            listKind: CMMDList
            plural: cmmds
            singular: cmmd
        scope: Namespaced
        versions:                                   <3>
        - name: v1                                  <4>
            schema:                                 <5>
            openAPIV3Schema:
                description: CMMD is the Schema for the cmmds API
                properties:
                    apiVersion:
                        type: string
                    kind:
                        type: string
                    metadata:
                        type: object
```

❶ API 버전을 정의합니다.
❷ 사용자 정의 리소스 정의(CRD) 문서로 지정된 YAML 문서 유형을 정의합니다.
❸ CRD에서 지원하는 버전 목록을 정의합니다.
❹ 해당 CRD 버전의 이름을 정의합니다.
❺ 리소스 스키마를 정의합니다.

```
spec:                                   <1>
     description: CMMDSpec defines the desired state of CMMD
     properties:                        <2>
         allocate:                      <3>
             description: Specify resource allocation manually
             properties:
                 cpu:                   <4>
  type: string
                 memory:                <5>
  type: string
                 nodeName:              <6>
  type: string
                 type: object
         allocateMode:                  <7>
             description: Specify resource allocation mode
             enum:
             - auto
             - manual
             type: string
         enable:  <8>
             description: Whether use of CMMD
             type: boolean
         payload: <9>
             description: Specify user resource manifest. e.g.) Pod, Deployment
             type: object
```

```
            x-kubernetes-embedded-resource: true
            x-kubernetes-preserve-unknown-fields: true
        required:<10>
        - allocateMode
        - enable
        - payload
        type: object
```

❶ 원하는 CMM-D에 대한 사양을 정의합니다
❷ 사양의 속성을 정의합니다
❸ CPU, 메모리, nodeName을 포함하여 리소스 할당을 수동으로 할당하기 위한 속성을 정의합니다.
❹ CPU 리소스 할당을 문자열 유형으로 지정합니다.(cpuset.cpu 값을 설정합니다.)
❺ 메모리 에 대한 리소스 할당을 숫자를 사용하여 문자열 유형으로 지정합니다.(cpuset.mems 값을 설정합니다.)
❻ 리소스가 할당될 노드의 이름을 문자열 형식으로 지정합니다.
❼ 리소스 할당 모드를 auto(자동 리소스 할당) 또는 manual(수동 리소스 할당)로 지정합니다.
❽ CMM-D를 사용할지 여부를 정의합니다.
❾ 사용자 지정 리소스(CR)에 대한 매니페스트를 지정합니다.
❿ Spec 속성 중 필수 필드를 나열합니다. 여기서 allocateMode, enable 및 payload는 필수입니다 .

2) 사용자 정의 리소스**(CR)** 샘플 **1** 제출 **-** 수동 모드

2-1) 사용자 정의 리소스**(CR)** 제출 시나리오

사용자가 사용자 정의 리소스(CR)에 의해 정의된 속성을 수동으로 지정하고 제출하는 방법의 예입니다.
CMM-D가 구성된 테스트 환경에서 워커 노드(cmmd1, cmmd2, cmmd3)에서 수동 모드로 CPU 및 메모리 NUMA를 지정하여 지정된 CPU 및 메모리가 올바르게 활용되는지 테스트합니다.

2-2) 실행 세부 정보

- Target Node : cmmd1 (intel cpu node)
- allocationMode : manual
- memory : 2 (CMMD Numa)
- cpu : 0-23 (cpu 0)

- Resource Quota : 100 GB
- Pod 생성 및 삭제를 통한 리소스 회수

2-3) CR(사용자 정의 리소스)

```
apiVersion: cmmd.samsung.com/v1
 kind: CMMD
 metadata:
     name: my-app
     namespace: work
 spec:
     allocateMode: manual                          <1>
     allocate:
         cpu: "0-23"                               <2>
         memory: "2"                               <3>
         nodeName: "cmmd1"                         <4>
     enable: true                                  <5>
     payload:
         apiVersion: v1
         kind: Pod
         metadata:
             name: my-app
         spec:
             containers:
             - name: stress
               image: <stress image>
               args: ["tail", "-f", "/dev/null"]
               resources:                          <6>
               requests:
                   memory: 100Gi
               limits:
                   memory: 100Gi
```

❶ 리소스 할당 모드(Manual: 리소스를 지정하여 명시적으로 할당하는 모드, Auto: 가장 효율적인 노드를 선택하여 리소스를 자동으로 할당하는 모드)에 대한 기술적인 설명입니다.
❷ 수동 모드에서 CPU 코어 정의
❸ 수동 모드에서 메모리 NUMA 정의
❹ 생성할 Pod의 대상 노드 이름 정의
❺ CMMD 장치 활용 여부 (true: 사용, false: 사용하지 않음)
❻ 요청/제한된 리소스를 설명합니다.

2-4) 사용자 정의 리소스 제출(Pod 생성)

```
[kni@bastion work]$ oc apply -f my-app.yaml
cmmd.cmmd.samsung.com/my-app created
```

2-5) 생성된 **Pod** 확인

```
[kni@bastion work]$ oc get pod
NAME      READY   STATUS    RESTARTS   AGE
my-app    1/1     Running   0          1s
```

2-6) 생성된 **Pod** 삭제

```
[kni@bastion work]$ oc delete -f my-app.yaml
cmmd.cmmd.samsung.com/my-app deleted
```

2-7) 삭제된 **Pod** 확인

```
[kni@bastion work]$ oc get pod
No resources found in work namespace
```

3) 사용자 정의 리소스**(CR)** 샘플 **2** 제출 **-** 자동 모드

3-1) 사용자 정의 리소스**(CR)** 제출 시나리오

사용자가 CR(Custom Resource)에 자동 모드(Auto Mode) 속성을 지정하여 Pod 생성을 요청하는 예시입니다. 자동 모드는 Pod 생성 시 가장 효율적인 노드를 자동으로 지정하는 스케줄링 기능을 제공합니다.

CMM-D가 설정된 클라우드 환경에서 이 기능을 검증합니다. cmmd1, cmmd2, cmmd3 워커 노드 중 cmmd2에 메모리 할당 400GB Pod를 할당하여 메모리 압력을 높인 후, 200GB Pod 생성 요청을 자동 모드로 제출합니다.

이 테스트는 3개 노드 중 메모리 압력(Pressure)이 가장 적은 노드에 Pod가 자동으로 할당되는지 확인하는 것을 목표로 합니다.

3-2) 사전 설정 세부 정보

- Target Node : cmmd2
- Create Permission : ServiceAccount 생성 및 권한 할당을 진행합니다.
- Resource Quota : cmmd2 노드의 CMM-D에 400GB Pod를 할당합니다.

3-3) 실행 세부 정보

- Target Node : All nodes (cmmd1, cmmd2, cmmd3)
- allocationMode : auto

- Resource Quota : 200 GB
- Pod 생성 및 삭제를 통한 리소스 회수

3-4) CR(사용자 정의 리소스)

```
apiVersion: cmmd.samsung.com/v1
kind: CMMD
metadata:
    name: my-app
    namespace: work
spec:
    allocateMode: auto                <1>
    enable: true                      <2>
    payload:
        apiVersion: apps/v1
        kind: Deployment
        metadata:
            name: my-app
        spec:
            replicas: 1
            selector:
                matchLabels:
                    name: my-app
            template:
                metadata:
                    labels:
                        name: my-app
                spec:
                    nodeSelector:
                        node-role.kubernetes.io/cmmd: ""
                    containers:
                    - name: stress01
                      image: <stress image>
                      args: ["tail", "-f", "/dev/null"]
                      resources:
                      requests:
                          memory: 200Gi
                      limits:
                          memory: 200Gi
                    serviceAccount: stress01
                    serviceAccountName: stress01
```

❶ 리소스 할당 모드(Manual: 리소스를 명시적으로 지정하여 할당하는 모드, Auto: 가장 효율적인 노드를 선택하여 자동으로 리소스를 할당하는 모드)에 대한 기술적인 설명입니다.
❷ CMMD 장치 활용 여부(true: 사용, false: 미사용)

3-5) 사용자 정의 리소스 제출(Pod 생성)

```
[kni@bastion work]$ oc apply -f my-app.yaml
cmmd.cmmd.samsung.com/my-app created
```

3-6) 생성된 **Pod** 확인

```
[kni@bastion work]$ oc get pod
NAME                      READY    STATUS    RESTARTS    AGE
my-app-6f856d99d8x6zrd    1/1      Running   0           1s
```

3-7) 생성된 **Pod** 삭제

```
[kni@bastion work]$ oc delete -f my-app.yaml
cmmd.cmmd.samsung.com/my-app deleted
```

3-8) 삭제된 **Pod** 확인

```
[kni@bastion work]$ oc get pod
No resources found in work namespace
```

6. OpenShift Virtualization

6.1 OpenShift Virtualization with CXL (Sidecar)

OpenShift와 마찬가지로 podman 등 Container runtime에서는 이미 cpuset.mems 와 같은 메모리 노드 (NUMA)에 대한 설정을 제공하지만 OpenShift Virtualization 4.18 테스트 결과 CMM-D Memory 환경과 같은 Zero-CPU NUMA에 대한 Memory Allocation (binding)을 지원하지 않고 있습니다.

Kubevirt 프로젝트는 게스트에게 가상화 컴퓨팅 기능을 제공하기 위해 적극적으로 노력하고 있습니다. 하지만, VirtualMachine 객체 정의에서 이러한 기능을 지원하지 않아 사용자 지정 옵션이 제한됩니다. 이번 활용가이드에서는 KubeVirt의 Hook Sidecar 기능을 사용하여 이러한 제한을 우회하여 CMM-D Memory가 할당된 가상머신 생성방법에 대해 안내하고 있습니다.

KubeVirt의 Hook Sidecar는 가상 머신이 초기화되기 전에 사용자 지정을 적용하는 컨테이너입니다. 프로덕션 환경에서는 권장되지 않지만, OpenShift Virtualization 환경에서 CrossNUMA 형태의 가상머신을 생성하여 개념이해 및 다양한 실험을 진행할 수 있습니다.

6.1.1 Hardware Configuration

Node	H/W	Usage	CPU	Local Memory	CXL
node #1	SMC	KVM	AMD 96 Core	128 GB	NA
node #2	SMC	OCP SNO #1	Intel 96 Core	1 TB	256 GB(128 GB * 2ea)

6.1.2 Software Configuration

소프트웨어 버전 정보는 다음과 같습니다:

```
# oc version
Client Version: 4.18.27
Kustomize Version: v5.4.2
Server Version: 4.18.27
Kubernetes Version: v1.31.13

# oc debug node/master1.ocp4.exp.com
sh-5.1# chroot  /host
sh-5.1# grep -e "OPENSHIFT_VERSION" -e "RHEL_VERSION" -e "^VERSION="
/etc/os-release
VERSION="418.94.202510230424-0"
OPENSHIFT_VERSION="4.18"
```

```
RHEL_VERSION=9.4
```

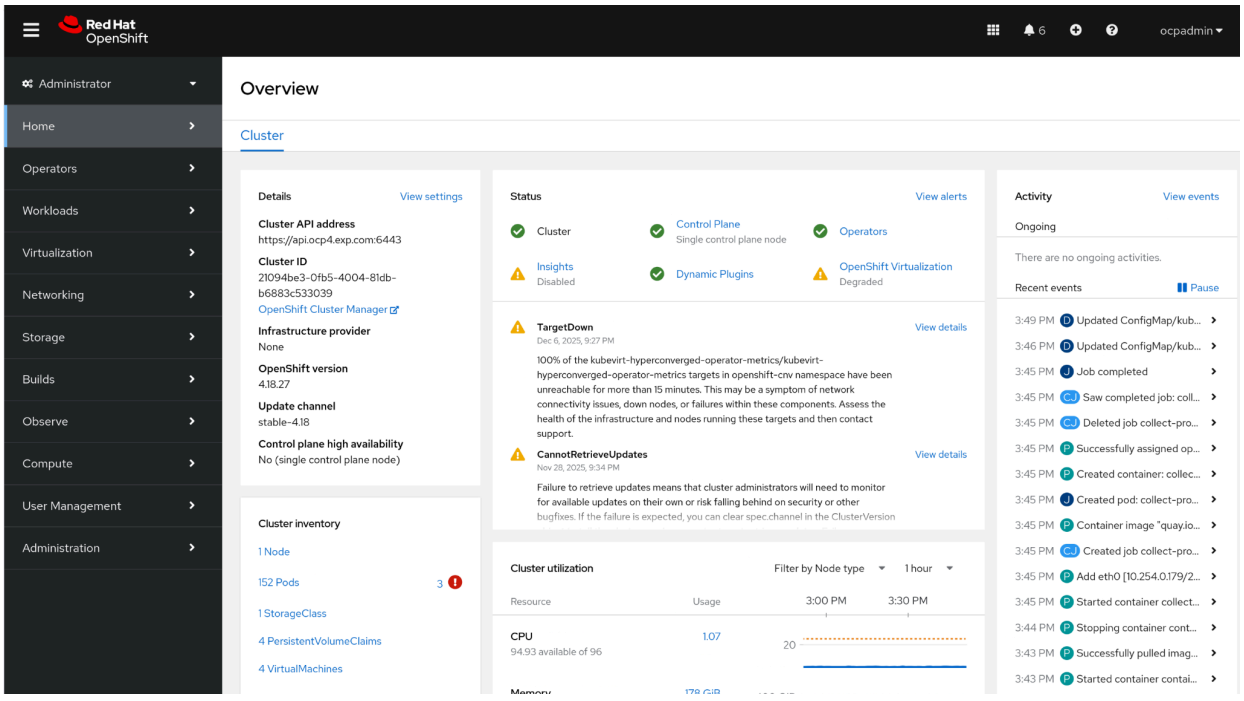
6.1.3 Install Operator (Openshift Virtualization)

현대의 IT 인프라는 급격하게 컨테이너 기반으로 이동하고 있지만, 여전히 수많은 레거시 애플리케이션과 특정 워크로드는 가상 머신(Virtual Machine, VM) 위에서 구동되어야 합니다. 개발자와 운영팀에게 가장 큰 고민은 바로 이 두 가지 세상(전통적인 VM 환경과 현대적인 컨테이너 환경)을 어떻게 효율적으로 공존시키느냐 하는 것입니다.

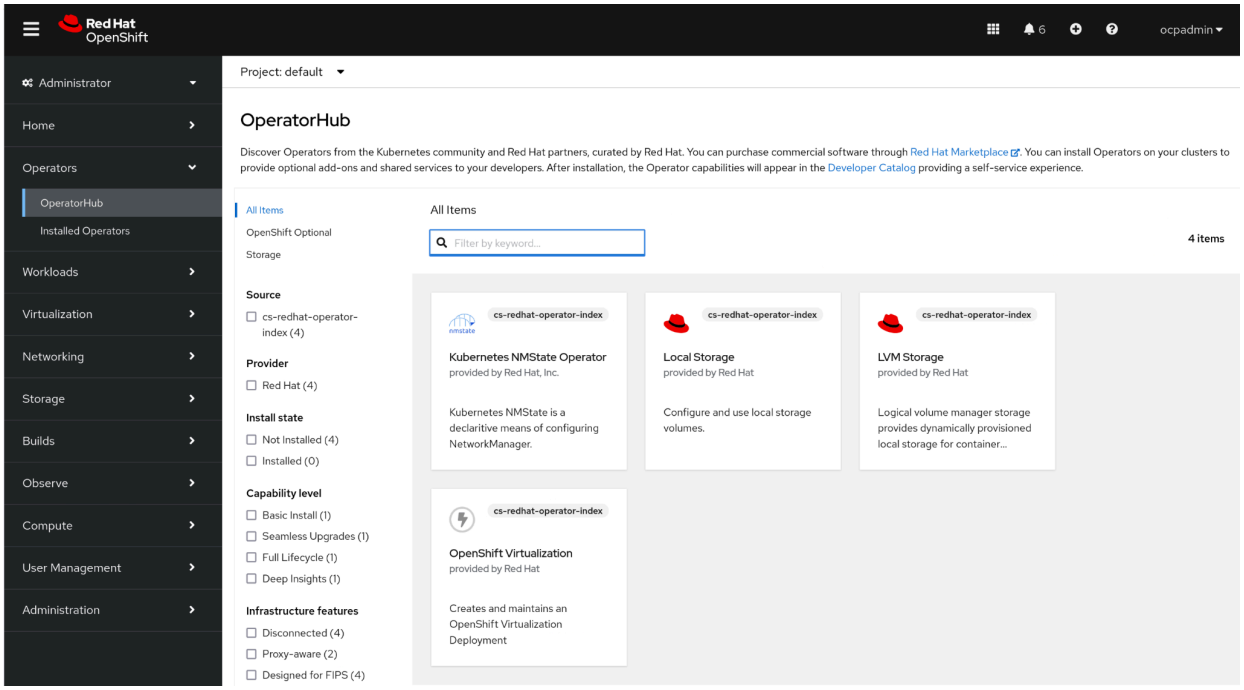
Red Hat OpenShift는 이러한 고민에 대한 명쾌한 해답으로 OpenShift Virtualization을 제시합니다. 이것은 단순히 VM을 생성하는 도구에 그치지 않고, VM을 쿠버네티스(Kubernetes) 네이티브 리소스로 취급하여 컨테이너와 동일한 파이프라인 안에서 관리할 수 있게 해주는 혁신적인 기능입니다.

1) Operator 검색

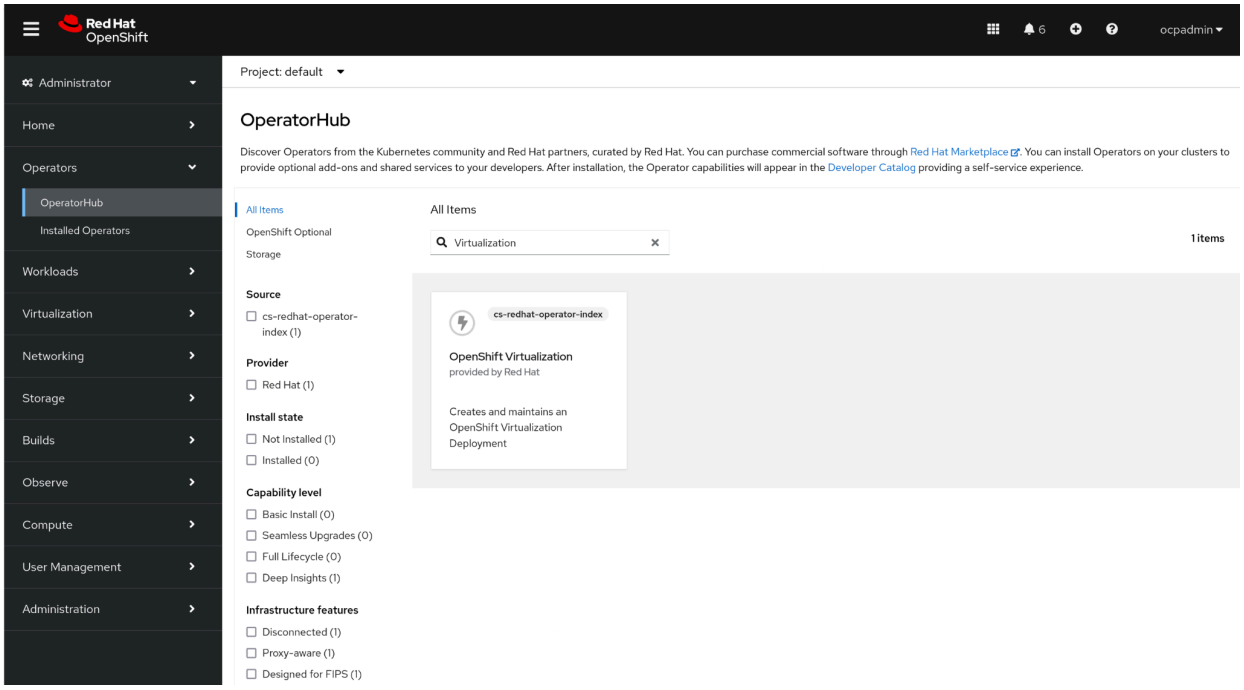
아래 이미지는 Red Hat OCP 웹 콘솔에 접속했을 때의 첫 번째 화면을 보여줍니다.



아래 이미지는 Red Hat OCP 웹 콘솔에서 Operator를 검색하기 위해 OperatorHub 메뉴를 클릭하면 나타나는 화면을 보여줍니다.

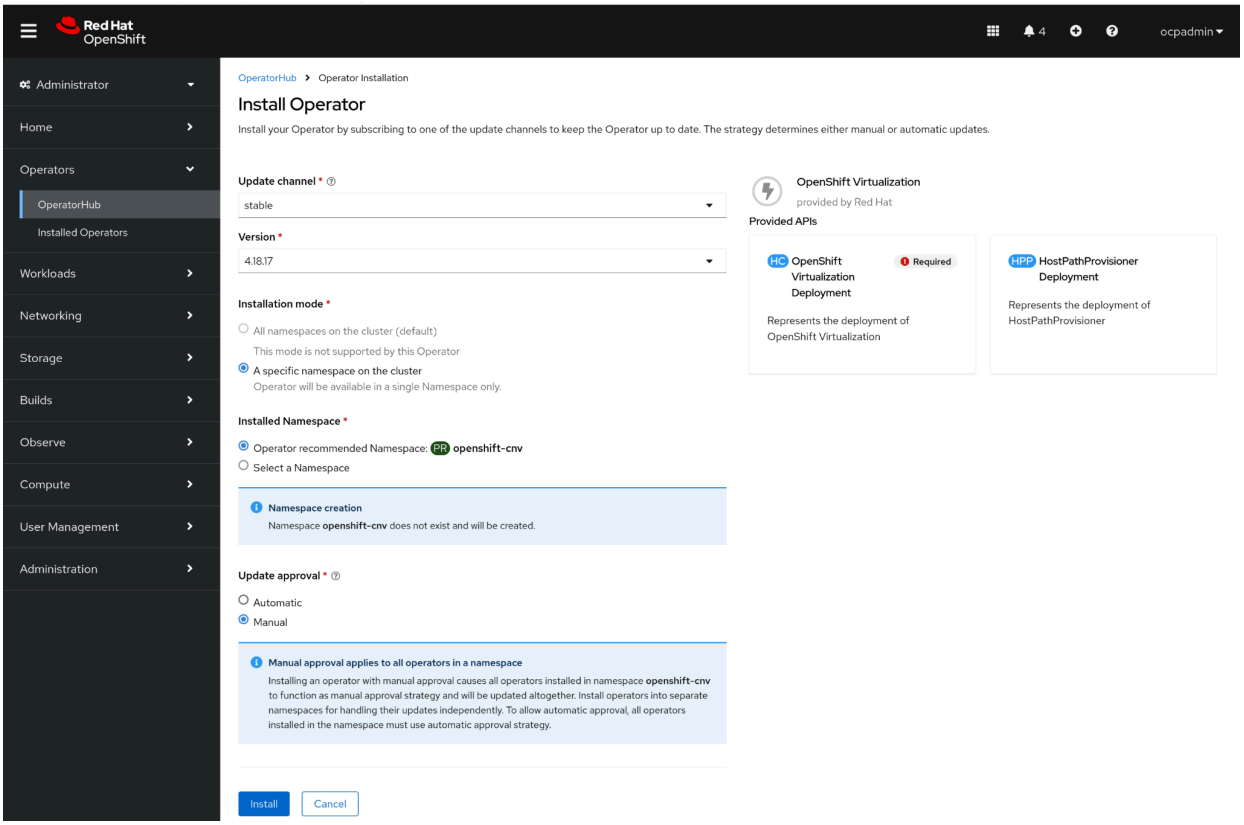


아래 이미지는 OpenShift Virtualization Operator를 검색하는 화면입니다. 검색어를 입력하면 일치하는 Operator가 표시됩니다.



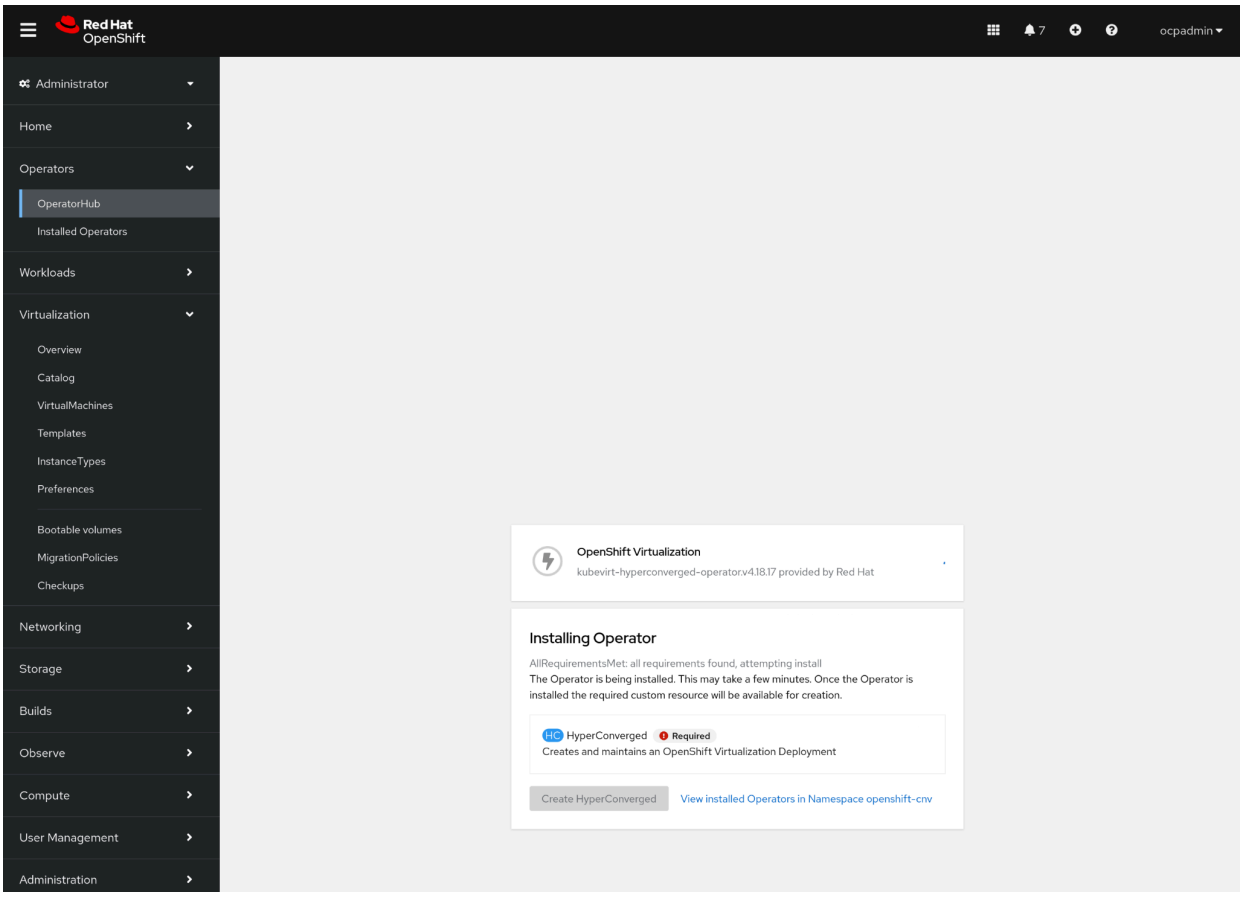
Operator 선택 및 설치 옵션 지정

아래 이미지는 1단계에서 찾은 Operator를 클릭하면 나타나는 설치 화면을 보여줍니다. 필요한 설치 옵션을 선택한 후, '설치' 버튼을 클릭하여 계속 진행하세요.

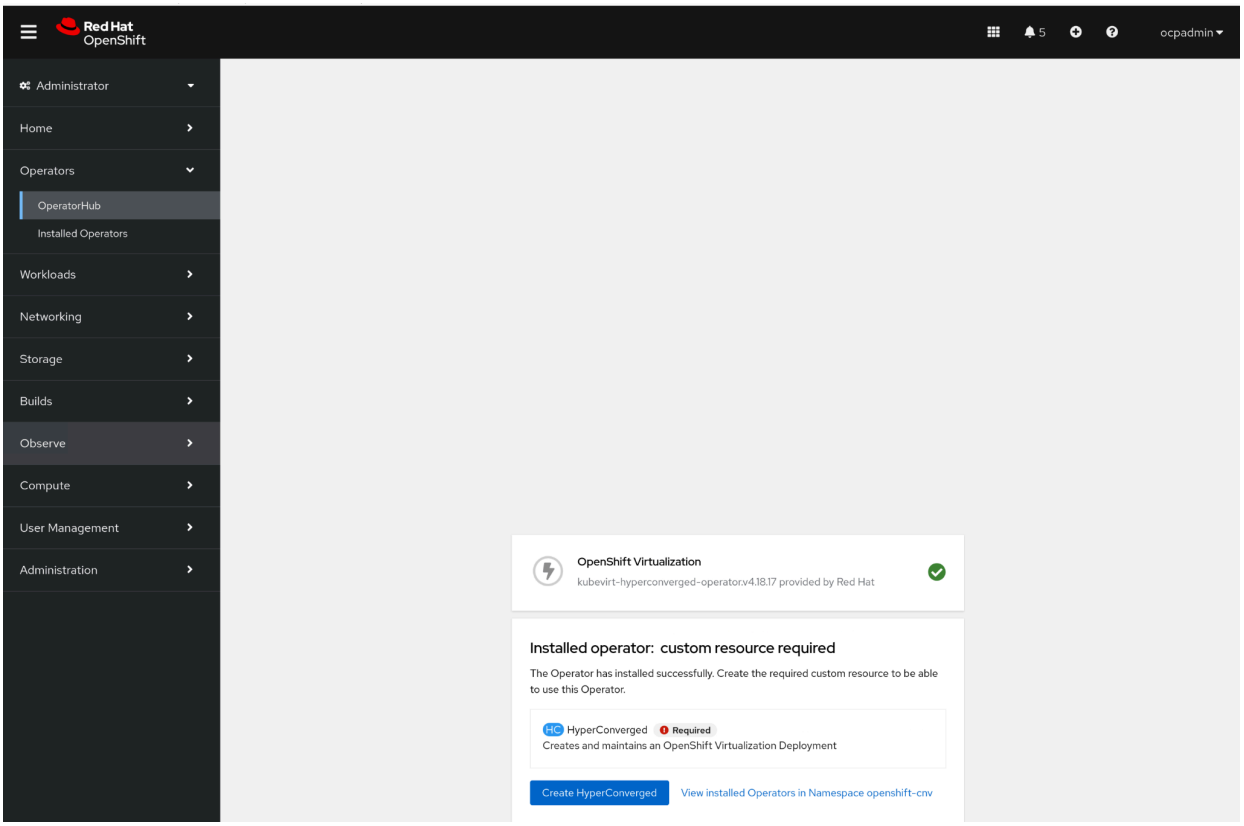


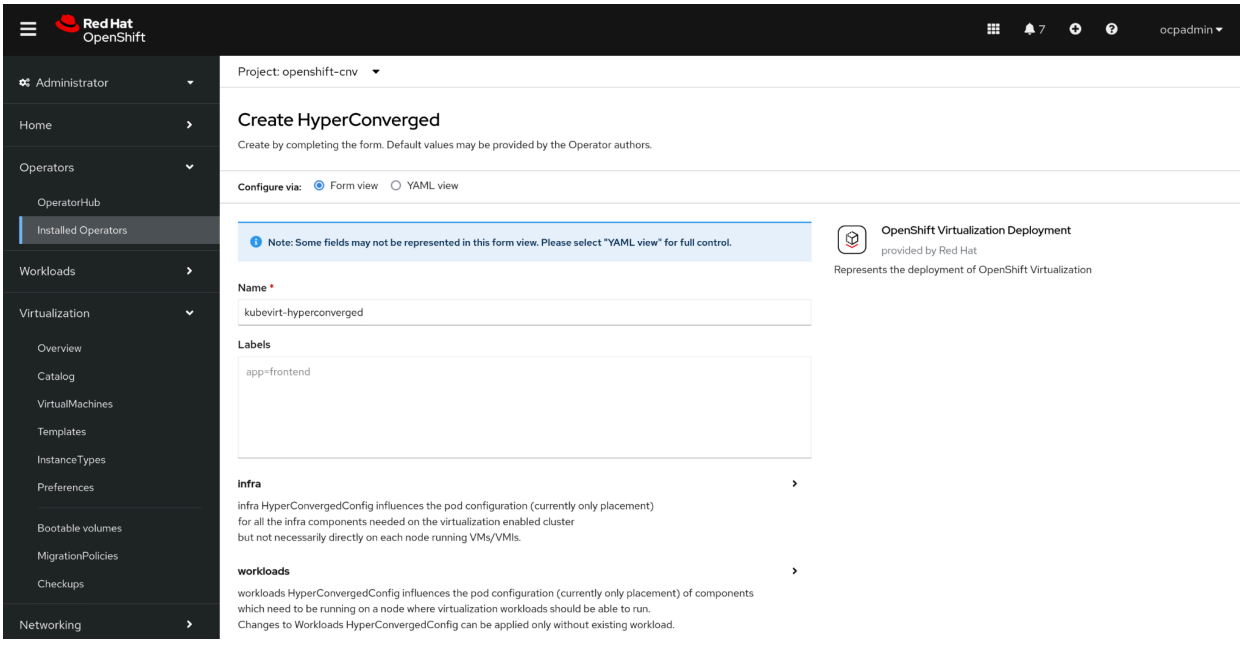
2) Operator 설치

아래 이미지는 이전 이미지에서 보여준 것과 같이 '설치' 버튼을 클릭하면 나타나는 설치 화면을 보여줍니다.

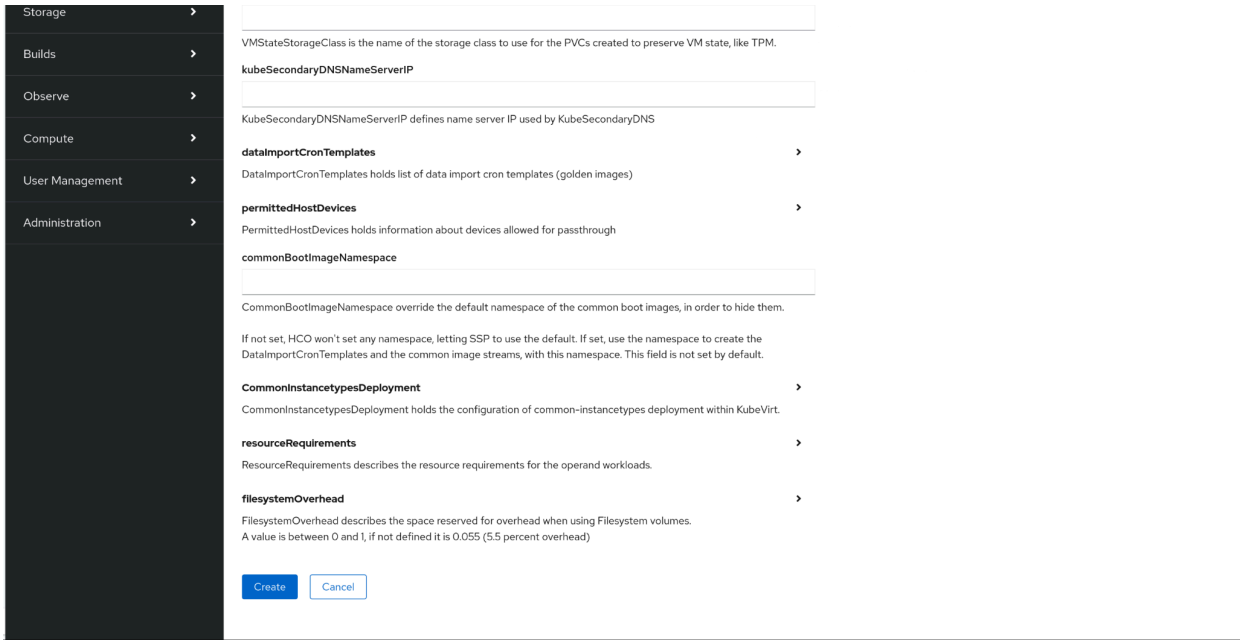


3) 운영자를 설치한 후 필수 단계로 **Create HyperConverged** 설치를 기본 제공하는 옵션으로 수정 하지않고 **Create** 진행합니다.


Red Hat OpenShift
5
ocpadmin
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OpenShift Virtualization
kubevirt-hyperconverged-operator.v4.18.17 provided by Red Hat
Installed operator: custom resource required
The Operator has installed successfully. Create the required custom resource to be able to use this Operator.
HC HyperConverged Required
Creates and maintains an OpenShift Virtualization Deployment
Create HyperConverged
View installed Operators in Namespace openshift-cnv

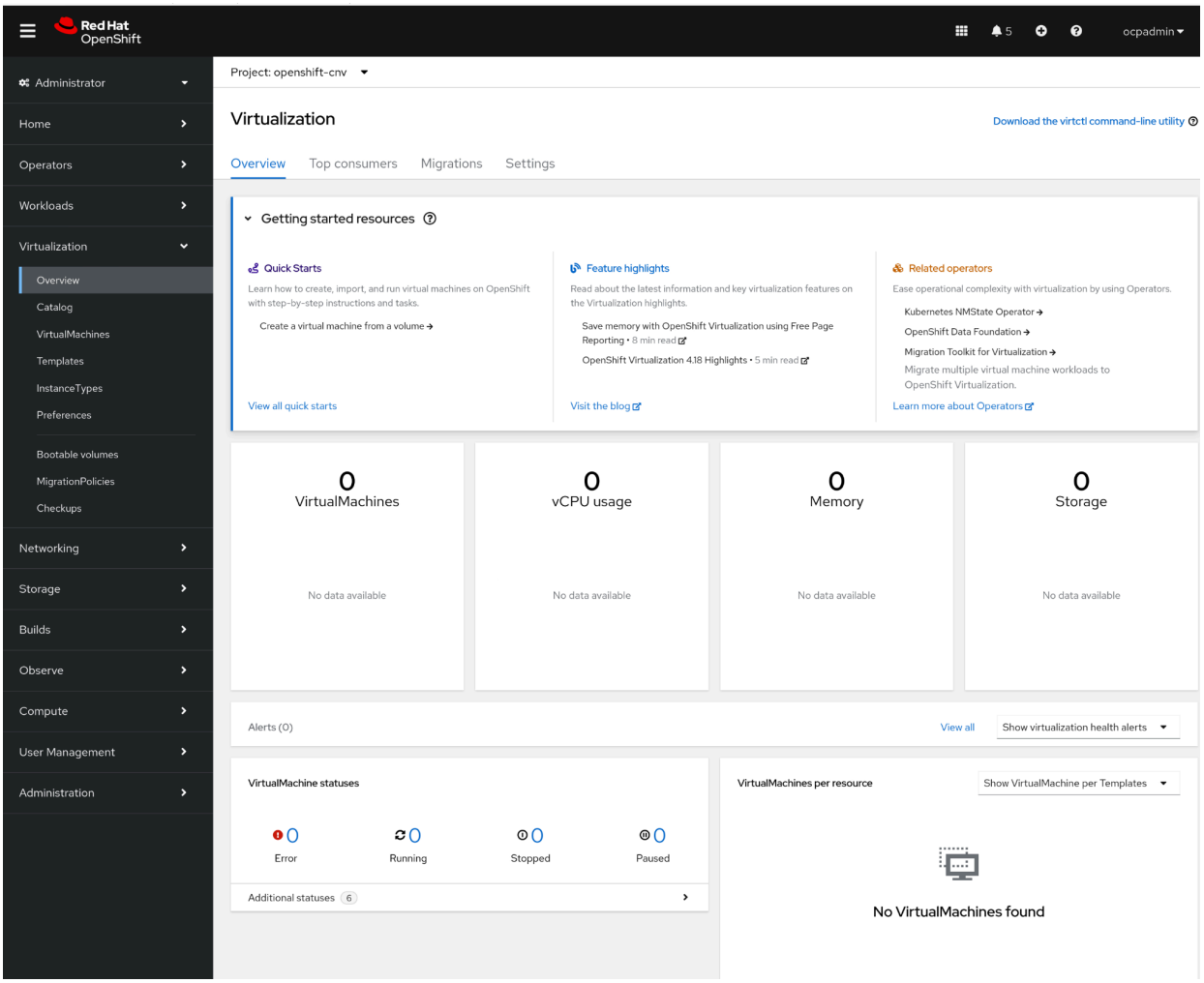

Red Hat OpenShift
7
ocpadmin
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Virtualization
Overview
Catalog
VirtualMachines
Templates
InstanceTypes
Preferences
Bootable volumes
MigrationPolicies
Checkups
Networking
Project: openshift-cnv
Create HyperConverged
Create by completing the form. Default values may be provided by the Operator authors.
Configure via: Form view YAML view
Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.
Name *
kubevirt-hyperconverged
Labels
app=frontend
infra
infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarily directly on each node running VMs/VMIs.
workloads
workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components which need to be running on a node where virtualization workloads should be able to run. Changes to Workloads HyperConvergedConfig can be applied only without existing workload.
OpenShift Virtualization Deployment
provided by Red Hat
Represents the deployment of OpenShift Virtualization

...



4) 웹 콘솔 플러그인 활성화 및 UI 확인

마지막 단계로, 사용자 편의를 위해 웹 콘솔에서 Virtualization 관련 기능을 시각적으로 확인할 수 있는 플러그인을 활성화합니다. 사용 설정 후 잠시 기다리면 새로 고침 알림이 팝업 되는데, 웹 콘솔 새로 고침을 클릭하여 UI를 업데이트합니다. 화면이 새로 고쳐지면 Virtualization의 상태나 활용도를 나타내는 새로운 UI 요소가 추가되어 관리 및 모니터링이 가능합니다.


Red Hat OpenShift
5
ocpadmin
Administrator
Home
Operators
Workloads
Virtualization
Overview
Catalog
VirtualMachines
Templates
InstanceTypes
Preferences
Bootable volumes
MigrationPolicies
Checkups
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: openshift-cnv
Virtualization
Download the virtctl command-line utility
Overview
Top consumers
Migrations
Settings
Getting started resources
Quick Starts
Learn how to create, import, and run virtual machines on OpenShift with step-by-step instructions and tasks.
Create a virtual machine from a volume
View all quick starts
Feature highlights
Read about the latest information and key virtualization features on the Virtualization highlights.
Save memory with OpenShift Virtualization using Free Page Reporting • 8 min read
OpenShift Virtualization 4.18 Highlights • 5 min read
Visit the blog
Related operators
Ease operational complexity with virtualization by using Operators.
Kubernetes NMState Operator
OpenShift Data Foundation
Migration Toolkit for Virtualization
Migrate multiple virtual machine workloads to OpenShift Virtualization.
Learn more about Operators
0
VirtualMachines
No data available
0
vCPU usage
No data available
0
Memory
No data available
0
Storage
No data available
Alerts (0)
View all
Show virtualization health alerts
VirtualMachine statuses
0
Error
0
Running
0
Stopped
0
Paused
Additional statuses 6
VirtualMachines per resource
Show VirtualMachine per Templates
No VirtualMachines found

6.1.4 Workaround Sidecar

KubeVirt Hook Sidecar 참고 링크
https://kubevirt.io/user-guide/user_workloads/hook-sidecar/

KubeVirt Hook Sidecar 컨테이너 적용 검증
Kubevirt의 Sidecar 기능을 사용하면 가상 머신 Pod에 추가 컨테이너를 연결할 수 있습니다. 이러한 Sidecar 컨테이너는 VM 컨테이너와 함께 실행되어 핵심 VM 자체를 수정하지 않고도 모니터링, 로깅, 디버깅 및 VM 환경의 사용자 지정 수정과 같은 향상된 기능을 제공합니다.

1) **OV Sidecar** 활성

OpenShift 환경에서는 Sidecar 기능이 기본적으로 활성화되어 있지 않으므로, 해당 기능을 사용하기 위해서는 별도의 활성화 절차가 필요합니다.

1-1) Sidecar 기능 게이트 활성화

사이드카 기능은 명시적으로 활성화되어야 하는 기능 게이트에 의해 제어됩니다.

```
[root@bastion ~]# kubectl annotate --overwrite -n openshift-cnv hco
kubevirt-hyperconverged \
  kubevirt.kubevirt.io/jsonpatch='[{"op": "add", "path":
"/spec/configuration/developerConfiguration/featureGates/-", "value": "Sidecar"}]'
hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged annotated
```

1-2) Sidecar 활성화 확인

기능 게이트가 성공적으로 추가되었는지 확인하려면 다음을 수행하십시오.

```
[root@bastion ov_yaml]# kubectl get kubevirt kubevirt-kubevirt-hyperconverged -n
openshift-cnv   -o
jsonpath='{.spec.configuration.developerConfiguration.featureGates}' | jq .
[
  "CPUManager",
  "Snapshot",
  "HotplugVolumes",
  "ExpandDisks",
  "GPU",
  "HostDevices",
  "VMExport",
  "DisableCustomSELinuxPolicy",
```

```
  "KubevirtSeccompProfile",
  "VMPersistentState",
  "NetworkBindingPlugins",
  "VMLiveUpdateFeatures",
  "DynamicPodInterfaceNaming",
  "VolumesUpdateStrategy",
  "VolumeMigration",
  "WithHostModelCPU",
  "HypervStrictCheck",
  "Sidecar"
]
```

1-3) Hook ConfigMap 구성

ConfigMap은 onDefineDomain 후크를 사용하여 VM의 libvirt XML 정의에 사용자 정의 메타데이터를 추가하여 VM 구성을 수정하는 스크립트를 제공합니다.

```
# configmap 적용
[root@bastion ov_yaml]# cat single_nodeset_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cmmd-single-nodeset
data:
  script.sh: |
    #!/bin/sh
    tempFile=$(mktemp --dry-run)
    echo "$4" > "$tempFile"
    sed -i "/<\/cpu>/a <numatune>\n  <memory mode='strict'
nodeset='2'/>\n</numatune>" "$tempFile"
    cat "$tempFile"

[root@bastion ov_yaml]# oc create -f single_nodeset_configmap.yaml
configmap/cmmd-single-nodeset created
```

1-4) VM 사이드카 설정

VM에 후크 사이드카를 추가하려면 후크 구성을 지정하는 필수 주석을 포함하도록 VM 매니페스트를 수정해야 합니다. 스크립트가 포함된 ConfigMap은 주석에서 참조되어야 합니다.

```
[root@bastion ov_yaml]# vim single_nodeset_vm.yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
```

```
metadata:
  name: rhel9-single-nodeset
  namespace: default
  labels:
    app: rhel9-single-nodeset
    vm.kubevirt.io/os: rhel9
    vm.kubevirt.io/workload: server
spec:
  runStrategy: RerunOnFailure
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1beta1
    kind: DataVolume
    metadata:
      name: rhel9-single-nodeset
    spec:
      source:
        http:
          url: http://172.16.18.221:8081/rhel-9.6-x86_64-kvm.qcow2
      storage:
        resources:
          requests:
            storage: 50Gi
        storageClassName: hostpath-csi
  template:
    metadata:
      annotations:
        hooks.kubevirt.io/hookSidecars: '[{"args": ["--version", "v1alpha2"],
          "configMap": {"name": "cmmd-single-nodeset", "key": "script.sh",
"hookPath": "/usr/bin/onDefineDomain"}}]'
      labels:
        kubevirt.io/domain: rhel9-single-nodeset
        kubevirt.io/size: small
        network.kubevirt.io/headlessService: headless
    spec:
      domain:
        cpu:
          cores: 8
          sockets: 1
          threads: 1
        memory:
          guest: 16Gi
        devices:
          disks:
          - name: rootdisk
            disk:
              bus: virtio
          - name: cloudinitdisk
            disk:
              bus: virtio
          interfaces:
          - name: default
            masquerade: {}
            model: virtio
            macAddress: 02:f7:a7:00:00:19
        rng: {}
        features:
          acpi: {}
```

```
      smm:
        enabled: true
    firmware:
      bootloader:
        efi: {}
    machine:
      type: pc-q35-rhel9.4.0
    resources: {}
  networks:
  - name: default
    pod: {}
  nodeSelector:
    kubernetes.io/hostname: master1.ocp4.exp.com
  terminationGracePeriodSeconds: 180
  volumes:
  - name: rootdisk
    dataVolume:
      name: rhel9-single-nodeset
  - name: cloudinitdisk
    cloudInitNoCloud:
      userData: |
        #cloud-config
        user: cloud-user
        password: !redhat123
        chpasswd: { expire: False }
```

For onDefineDomain hook:
Argument 1: The hook name (onDefineDomain)
Argument 2: The --version parameter (e.g., v1alpha2)
Argument 3: The --vmi parameter with VMI information as JSON string
Argument 4: The --domain parameter with the current domain XML

6.1.5 Create VirtualMachine (VM)

1) VirtualMachine 생성

앞서 활성화된 OpenShift의 Sidecar 기능을 활용하여, 이 기능을 사용하는 가상 머신(VM)을 생성합니다.

```
[root@bastion ov_yaml]# oc create -f single_nodeset_vm.yaml
virtualmachine.kubevirt.io/rhel9-single-nodeset created

[root@bastion ov_yaml]# oc get pod
NAME                                      READY   STATUS    RESTARTS   AGE
virt-launcher-rhel9-single-nodeset-4wc2k   2/2     Running   0          18s
```

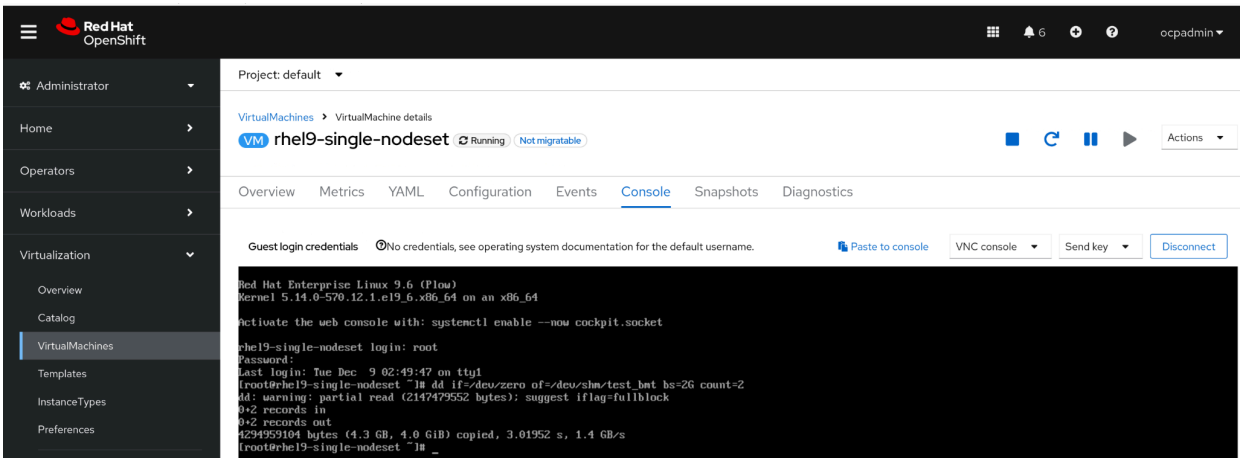
2) VirtualMachine with CMM-D 검증

Openshift Sidecar 기능으로 생성된 가상 머신(VM)이 정상적으로 CMM-D에 영역에 할당되었는지 확인하기 위하여 VM XML 상세내역을 확인할 수 있습니다.

```
[root@bastion ov_yaml]# oc exec virt-launcher-rhel9-single-nodeset-4wc2k -n default
-- virsh dumpxml 1 | grep -A3 numatune
  <numatune>
    <memory mode='strict' nodeset='2'/>  <-해당 라인 추가 확인
  </numatune>
  <sysinfo type='smbios'>
    <system>
      <entry name='manufacturer'>Red Hat</entry>
```

3) NUMA Node 2 Memory 부하 테스트

dd 명령어를 사용하여 /dev/shm 디렉터리에 메모리 사용 부하 작업을 수행합니다. 이는 시스템의 공유 메모리 영역을 직접 활용하여 master1 노드의 메모리 리소스를 인위적으로 점유하고, 특히 메모리 NUMA Node 2(CMM-D) 영역의 메모리 사용이 증가하는지 검증하기 위함입니다.



4) CXL 메모리 사용률 확인

이러한 부하 작업 수행과 동시에, Node (CMM-D Memory가 연결된 노드)의 CMM-D Memory 사용률을 모니터링합니다. 이는 노드에서 메모리 압력이 발생했을 때, CMM-D Memory가 확장 메모리 풀로써 정상적으로 활용되고 있는지 또는 NUMA Node 2의 부하가 CMM-D Memory 사용으로 이어지는지 확인하는 중요한 지표가 됩니다.

```
[root@master1 ~]# toolbox                <1>
[root@toolbox /]# numastat -cm

Per-node system memory usage (in MBs):
                 Node 0 Node 1 Node 2 Node 3   Total
                 ------ ------ ------ ------ -------
MemTotal         515719 516036 131072 131072 1293899
MemFree          474622 482126 125661 131072 1213482
MemUsed           41096  33910   5411      0   80417
SwapCached            0      0      0      0       0
Active            11252  10593   5406      0   27251
Inactive          21600  20320      0      0   41920
Active(anon)      10618   9348   5406      0   25372
Inactive(anon)        0      0      0      0       0
Active(file)        634   1245      0      0    1879
Inactive(file)    21600  20320      0      0   41920
Unevictable         112     48      0      0     160
Mlocked             109     48      0      0     157
Dirty                 0      0      0      0       0
Writeback             0      0      0      0       0
FilePages         22331  21613      0      0   43944
Mapped             2838   2589      0      0    5427
AnonPages          9124   7844   5404      0   22372
Shmem                96     41      0      0     136
KernelStack          85     83      0      0     168
PageTables           87     82      0      0     169
SecPageTables         0      0      0      0       0
NFS_Unstable          0      0      0      0       0
Bounce                0      0      0      0       0
```

```
WritebackTmp           0      0      0      0       0
KReclaimable         747    795      0      0    1542
Slab                2196   1804      0      0    4000
SReclaimable         747    795      0      0    1542
SUnreclaim          1449   1008      0      0    2457
AnonHugePages       6446   5126   5374      0   16946
ShmemHugePages         0      0      0      0       0
ShmemPmdMapped         0      0      0      0       0
FileHugePages          0      0      0      0       0
FilePmdMapped          0      0      0      0       0
Unaccepted             0      0      0      0       0
HugePages_Total        0      0      0      0       0
HugePages_Free         0      0      0      0       0
HugePages_Surp         0      0      0      0       0
```

❶ Toolbox는 일반적으로 registry.redhat.io/rhel8/support-tools 이미지를 기반으로 실행되는 특수 목적의 컨테이너입니다. RHCOS 호스트에는 설치되어 있지 않은, 그러나 시스템 분석에는 필수적인 수많은 진단 도구들이 이 컨테이너를 이용해서 모니터링할 수 있습니다.

7. 기타 CMM-D 활용방법

7.1 OpenShift Virtualization with HugePage (CXL)

Sidecar 기능의 추가 활성화 과정 없이도, 특수 목적을 위해 CMM-D가 관리하는 영역의 메모리를 Hugepage 형태로 가상 머신(VM)에 직접 할당하여 사용할 수 있습니다.
이 방식은 Sidecar와 같은 추가적인 기능 사용 없이 CMM-D Memory 리소스를 VM에 최적화된 대용량 메모리 페이지로 제공하여 활용하는 것을 가능하게 합니다.

1) **Hugepage** 설정

1-1) tuneD 설정

OpenShift Container Platform(OCP)에서 TuneD는 클러스터 노드(서버)의 운영체제와 하드웨어 설정을 워크로드 목적에 맞게 자동으로 최적화해주는 튜닝 데몬입니다. 노드가 최고의 성능을 낼 수 있도록 리눅스 커널과 시스템 설정을 자동으로 조율해 주는 조율사 역할을 하며 해당 기능을 이용하여 Hugepage 설정을 합니다.

```
# node Hugepages 설정 내용 확인
[root@worker3 ~]# cat
/sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_hugepages
0

# node tuned profile 적용전 확인
[root@bastion ov_yaml]# oc get profile -n openshift-cluster-node-tuning-operator
NAME                   TUNED                       APPLIED   DEGRADED   MESSAGE
AGE
master1.ocp4.exp.com   openshift-control-plane   True      False      TuneD profile
applied.   9d
master2.ocp4.exp.com   openshift-control-plane   True      False      TuneD profile
applied.   10d
master3.ocp4.exp.com   openshift-control-plane   True      False      TuneD profile
applied.   10d
worker1.ocp4.exp.com   openshift-node            True      False      TuneD profile
applied.   10d
worker2.ocp4.exp.com   openshift-node            True      False      TuneD profile
applied.   10d
worker3.ocp4.exp.com   openshift-node            True      False      TuneD profile
applied.   10d

## 주의: hugepages의 개수와 용량이 너무 크게 설정되면 core OS 부팅 불가 문제가 발생 할 수
있으므로 주의 필요

[root@bastion yaml]# vim hugepages-tuned-sysfs.yaml
apiVersion: tuned.openshift.io/v1
```

```
kind: Tuned
metadata:
  name: hugepages-numa1
  namespace: openshift-cluster-node-tuning-operator
spec:
  profile:
  - data: |
      [main]
      summary=Runtime configuration for hugepages on NUMA node 1
      include=openshift-node

      [sysfs]
      /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_hugepages=120000
    name: openshift-node-hugepages-numa1
  recommend:
  - machineConfigLabels:
      machineconfiguration.openshift.io/role: "cmmd"
    priority: 30
    profile: openshift-node-hugepages-numa1

[root@bastion yaml]# oc create -f hugepages-tuned-sysfs.yaml
tuned.tuned.openshift.io/hugepages-numa1 created
```

1-2) tuneD 설정 후 Hugepage 설정 확인

worker node에 직접 접속하여 Hugepage가 정상 설정이 되었는지 확인을 합니다.

```
[root@worker3 ~]# cat
/sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_hugepages
120000

[root@worker3 ~]# cat /proc/meminfo|grep -i huge
AnonHugePages:    729088 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:   120000
HugePages_Free:    120000
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:        245760000 kB

[root@toolbox /]# numastat -cm
Per-node system memory usage (in MBs):
                 Node 0 Node 1  Total
                 ------ ------ ------
MemTotal         128511 260096 388607
MemFree          117468  20096 137564
MemUsed           11043 240000 251043
...
HugePages_Total       0 240000 240000
HugePages_Free        0 240000 240000
```

```
HugePages_Surp        0       0       0


[root@bastion yaml]# oc get nodes
NAME                   STATUS                     ROLES                  AGE
VERSION
master1.ocp4.exp.com   Ready                      control-plane,master   10d
v1.31.12
master2.ocp4.exp.com   Ready                      control-plane,master   10d
v1.31.12
master3.ocp4.exp.com   Ready                      control-plane,master   10d
v1.31.12
worker1.ocp4.exp.com   Ready                      worker                 10d
v1.31.12
worker2.ocp4.exp.com   Ready                      worker                 10d
v1.31.12
worker3.ocp4.exp.com   Ready    cmmd                  10d   v1.31.12
```

1-3) CMM-D에 설정 된 Hugepage를 이용한 VM 생성

Hugepage 설정이 적용된 TuneD 프로파일을 활용하여 가상 머신(VM)을 생성함으로써, VM이 최적화된 대용량 메모리를 통해 CPU와 메모리의 NUMA 정책을 일치시켜 효율적으로 구동되도록 합니다.

```
## worker node에 수동으로 반영한 내용 OCP Cluster에 반영을 위해 kublet.service 재 시작
[root@worker3 ~]# systemctl restart kubelet.service

## Hugepage가 worker3에 반영 여부 확인
[root@bastion ov_yaml]# oc describe node worker3.ocp4.exp.com | grep -A15 Capacity
Capacity:
  cpu:                            96
  devices.kubevirt.io/kvm:        1k
  devices.kubevirt.io/tun:        1k
  devices.kubevirt.io/vhost-net:  1k
  ephemeral-storage:              936709572Ki
  hugepages-1Gi:                  0
  hugepages-2Mi:                  240000Mi
  memory:                         397933332Ki
  pods:                           250
Allocatable:
  cpu:                            95500m
  devices.kubevirt.io/kvm:        1k
  devices.kubevirt.io/tun:        1k
  devices.kubevirt.io/vhost-net:  1k
  ephemeral-storage:              862197798302

[root@bastion ov_yaml]# vim you_vm.yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
```

```
metadata:
  name: rhel9-you-vm
  namespace: default
  labels:
    app: rhel9-you-vm
    vm.kubevirt.io/os: rhel9
    vm.kubevirt.io/workload: server
spec:
  runStrategy: RerunOnFailure
  dataVolumeTemplates:
  - apiVersion: cdi.kubevirt.io/v1beta1
    kind: DataVolume
    metadata:
      name: rhel9-you-vm
    spec:
      source:
        http:
          url: http://172.16.18.221:8081/rhel-9.6-x86_64-kvm.qcow2
      storage:
        resources:
          requests:
            storage: 50Gi
        storageClassName: hostpath-csi
  template:
    metadata:
      labels:
        kubevirt.io/domain: rhel9-you-vm
        kubevirt.io/size: small
        network.kubevirt.io/headlessService: headless
    spec:
      domain:
        cpu:
          cores: 1
          sockets: 2
          threads: 1
          dedicatedCpuPlacement: true
          numa:
            guestNUMA:
              numaNodeCount: 1
        memory:
          guest: 16Gi
          hugepages:
            pageSize: "2Mi"
        devices:
          disks:
          - name: rootdisk
            disk:
              bus: virtio
          - name: cloudinitdisk
            disk:
              bus: virtio
          interfaces:
          - name: default
            masquerade: {}
            model: virtio
            macAddress: 02:f7:a7:00:10:11
          rng: {}
```

```
        features:
          acpi: {}
          smm:
            enabled: true
        firmware:
          bootloader:
            efi: {}
        machine:
          type: pc-q35-rhel9.4.0
        resources:
          requests:
            cpu: "2"
            memory: "16Gi"
            hugepages-2Mi: "16Gi"                    <1>
          limits:
            cpu: "2"
            memory: "16Gi"
            hugepages-2Mi: "16Gi"                    <2>
      networks:
      - name: default
        pod: {}
      nodeSelector:
        kubernetes.io/hostname: worker3.ocp4.exp.com
      terminationGracePeriodSeconds: 180
      volumes:
      - name: rootdisk
        dataVolume:
          name: rhel9-you-vm
      - name: cloudinitdisk
        cloudInitNoCloud:
          userData: |
            #cloud-config
            user: cloud-user
            password: mocf-443w-3klx
            chpasswd: { expire: False }

[root@bastion ov_yaml]# oc create -f you_vm.yaml
virtualmachine.kubevirt.io/rhel9-you-vmcreated
```

❶ Hugepage 사용 용량 정의
❷ Hugepage 사용 Limit 용량 정의

1-4) CMM-D Hugepage 사용 VM 확인

구동된 VM이 CMM-D 메모리 영역 NUMA Node 1의 메모리를 사용하여 잘 구동이 되었는지 확인을 합니다.

```
[root@bastion ov_yaml]# oc get vm
NAME           AGE     STATUS    READY
rhel9-you-vm   4m46s   Running   True

[root@bastion ov_yaml]# oc get vmi -o wide
NAME           AGE    PHASE     IP            NODENAME               READY
LIVE-MIGRATABLE   PAUSED
rhel9-you-vm   5m     Running   10.254.4.15   worker3.ocp4.exp.com   True    False

## VM 구동 후 worker3에서 Hugepage 사용 확인
[root@toolbox /]# numastat  -cm

Per-node system memory usage (in MBs):
                 Node 0 Node 1  Total
                 ------ ------ ------
MemTotal         128511 260096 388607
MemFree          116615  20096 136711
MemUsed           11895 240000 251895
…
HugePages_Total       0 240000 240000
HugePages_Free        0 223616 223616
HugePages_Surp        0      0      0
```