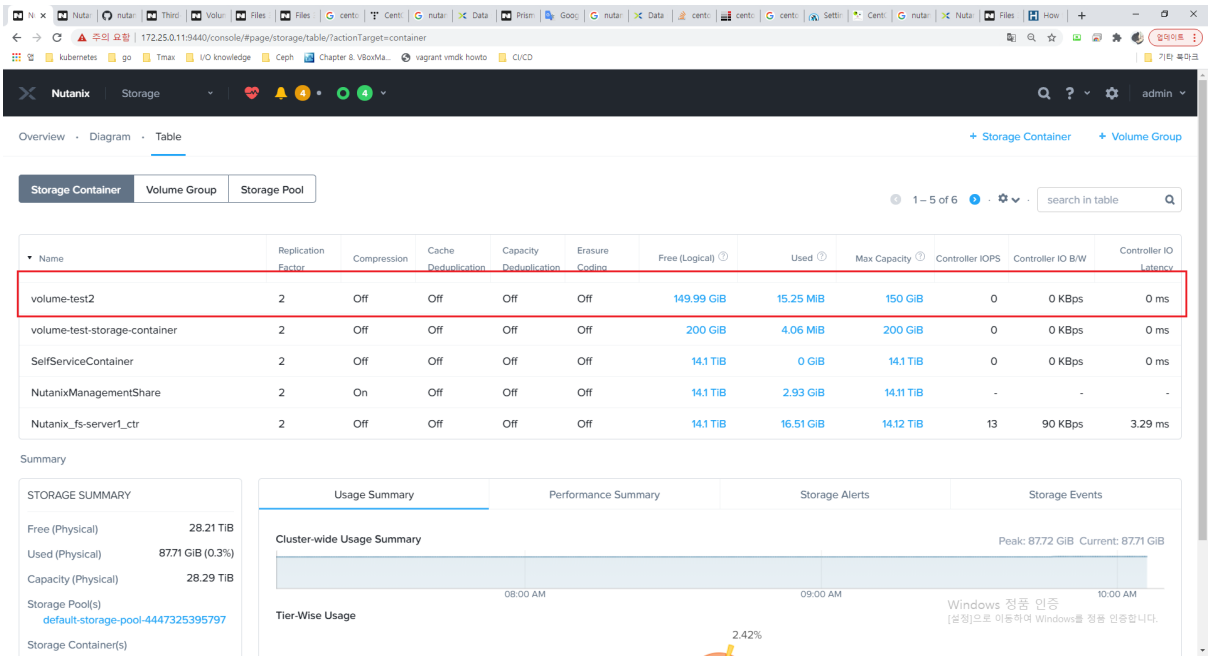
Nutanix Volume CSI 사용성 조사

본 문서는 Nutanix Volume 스토리지를 K8s에서 CSI를 이용하여 사용하는 방법을 다룹니다.

[Prerequisite]

1. Nutanix 클러스터 및 Prism이 구성되어 있어야 합니다.
   - AOS version: 5.15.5.1 LTS

2. Nutanix Volume을 provisioning할 충분한 크기의 storage container가 생성되어 있어야 합니다.
   - 본 예제에서 사용하는 storage container는 아래 그림의 volume-test2와 같이 생성되어 있습니다.



3. Nutanix VM으로 구성한 K8s 클러스터가 있어야 합니다.
   - 실험 환경의 VM은 CentOS 7.9입니다
   - 단일 VM에 minikube로 K8s를 구성했고, K8s 버전은 1.19.4입니다.

[iSCSI Volume Provisioning]

Nutanix Volume Group을 iSCSI 프로토콜을 통해 VM 노드 및 K8s에 provisioning 합니다.

1. iscsi utility 패키지를 설치합니다.
   - CMD: yum install -y iscsi-initiator-utils

2. Nutanix에서 제공하는 CSI 에제 yaml 파일들을 다운받습니다.
   - 본 예제에서는 Nutanix CSI 2.3.1을 사용했습니다. (참고 https://github.com/nutanix/csi-plugin)
   - CMD: wget http://download.nutanix.com/csi/v2.3.1/csi-v2.3.1.tar.gz
   - CMD: tar xvf csi-v2.3.1.tar.gz && cd CSI-v2.3.1

3. Nutanix CSI 관련 RBAC을 생성합니다.
   - CMD: kubectl apply -f ntnx-csi-rbac.yaml

4. Provisioner가 접근하기 위한 secret을 생성합니다.
   - secret은 아래와 같은 내용으로 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: ntnx-secret
  namespace: kube-system
data:
  key: MTcyLjI1LjAuMTE6OTQ0MDphZG1pbjpOdXRhbml4MSE=
```

   - key 값은 base64 인코딩된 “prism-ip:prism-port:admin:password” 값입니다.
     - CMD: echo -n '172.25.0.11:9440:admin:Nutanix1!' | base64

5. Nutanix CSI provisioner가 작동하는 데 이용하는 CSIDriver를 생성합니다.
   - CMD: kubectl apply -f csi-driver.yaml

6. Nutanix CSI node provisioner DaemonSet을 생성합니다.
   - CMD: kubectl apply -f ntnx-csi-node.yaml

7. Nutanix CSI provisioner를 생성합니다.
   - CMD: kubectl apply -f ntnx-csi-provisioner.yaml

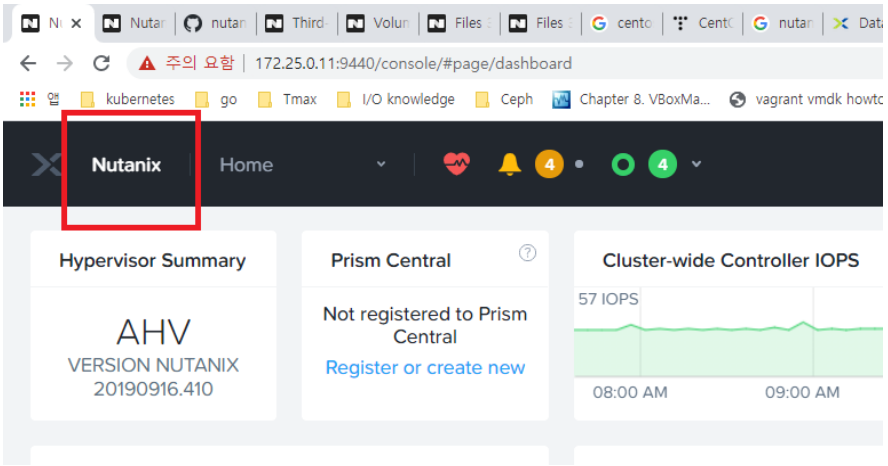
진행 후 Pod 결과는 아래와 같습니다.

```
[root@localhost csi-v2.3.1]# kubectl get pod -n kube-system
NAME                                          READY   STATUS    RESTARTS   AGE
coredns-f9fd979d6-jbxds                       1/1     Running   0          111m
csi-node-ntnx-plugin-brs2s                    3/3     Running   0          108m
csi-provisioner-ntnx-plugin-0                 5/5     Running   0          106m
etcd-localhost.localdomain                    1/1     Running   0          112m
kube-apiserver-localhost.localdomain          1/1     Running   0          112m
kube-controller-manager-localhost.localdomain 1/1     Running   0          112m
kube-proxy-6nswf                              1/1     Running   0          111m
kube-scheduler-localhost.localdomain          1/1     Running   0          112m
storage-provisioner                           1/1     Running   0          112m
```

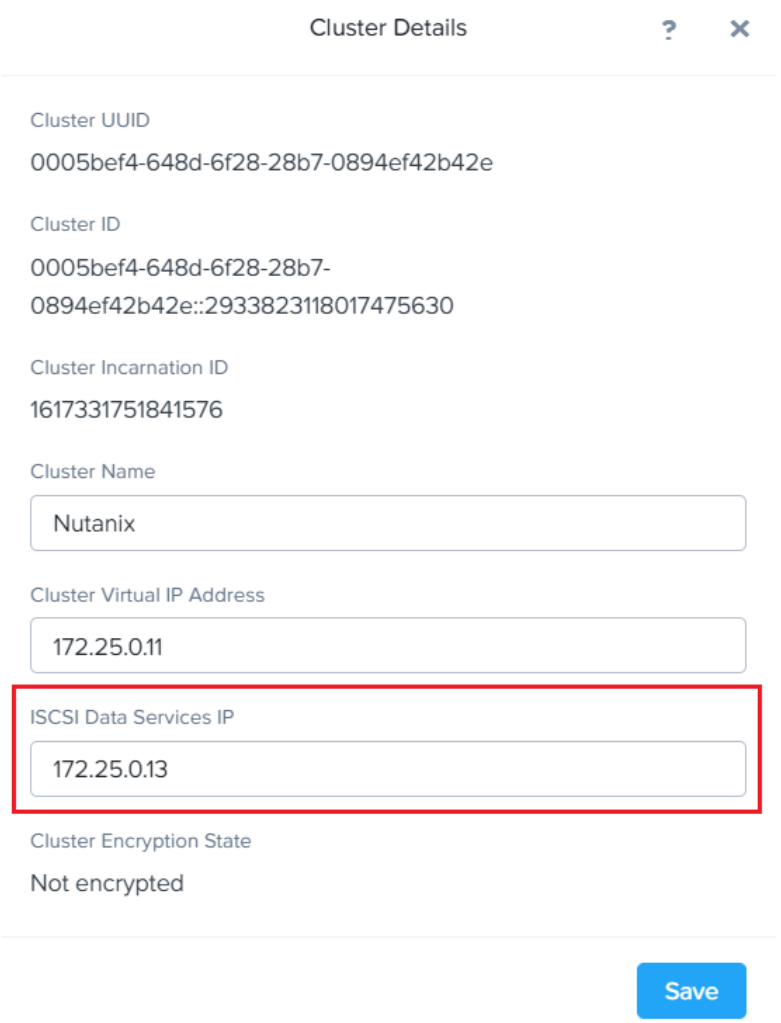
8. Nutanix Volume의 StorageClass를 생성합니다.
   - CMD: cd example/ABS
   - StorageClass는 아래와 같이 설정합니다.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: acs-abs
provisioner: csi.nutanix.com
parameters:
    csi.storage.k8s.io/provisioner-secret-name: ntnx-secret
    csi.storage.k8s.io/provisioner-secret-namespace: kube-system
    csi.storage.k8s.io/node-publish-secret-name: ntnx-secret
    csi.storage.k8s.io/node-publish-secret-namespace: kube-system
    csi.storage.k8s.io/controller-expand-secret-name: ntnx-secret
    csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
    csi.storage.k8s.io/fstype: ext4
    dataServiceEndPoint: 172.25.0.13:3260
    storageContainer: volume-test2
    storageType: NutanixVolumes
allowVolumeExpansion: true
reclaimPolicy: Delete
```

   - csi.storage.k8s.io API의 provisioner, node-publish, controller-expand 필드들에는 위에서 생성한 secret 이름과 namespace를 입력합니다.
   - parameters.dataServiceEndPoint에는 Prism의 data service endpoint를 입력합니다.
     - Data service의 endpoint IP를 확인하는 방법은 아래 그림과 같습니다.
     - Data service의 endpoint port는 default로 3260 입니다.



   - 위 Prism 홈 화면의 Nutanix를 클릭합니다.



- parameters.storageContainer에는 Nutanix Volume을 생성할 storage container를 입력합니다.
  - 본 예제에서는 volume-test2

9. 해당 StorageClass로부터 볼륨을 할당받는 PVC를 생성합니다.
   - CMD: kubectl apply -f claim1.yaml

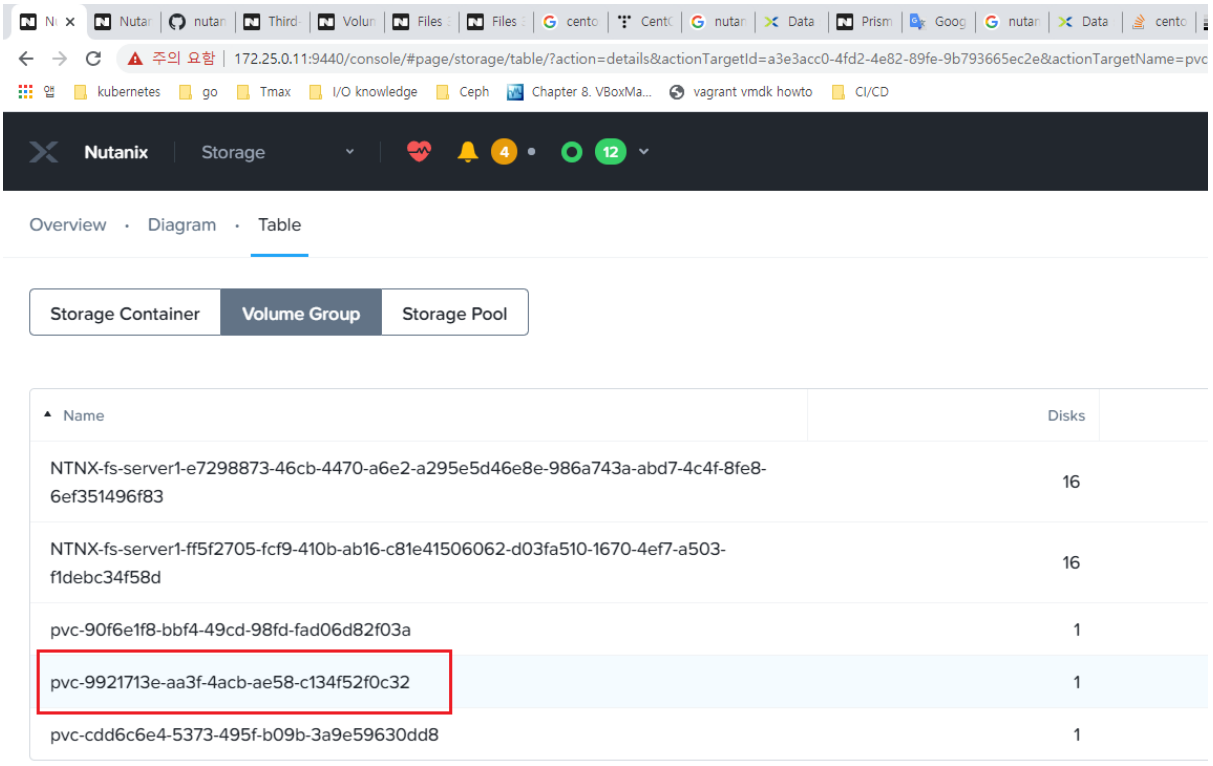
```
[root@control-plane ABS]# kubectl get pvc
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim1-test   Bound    pvc-9921713e-aa3f-4acb-ae58-c134f52f0c32   6Gi        RWO            acs-abs        4m24s
```

10. 위 PVC를 사용하는 Pod을 띄워 테스트합니다.
    - CMD: kubectl apply -f rc-nginx.yaml

```
[root@control-plane ABS]# kubectl get pod
NAME            READY   STATUS    RESTARTS   AGE
server-vgrnx    1/1     Running   0          4m51s
```

11. Provisioning된 volume 및 해당 volume이 VM node와 Pod에 attach 됐는지 확인합니다.
    - (Provisioning 체크) PVC Volume의 이름으로 새로 생성된 Prism Storage의 Volume Group을 확인합니다.

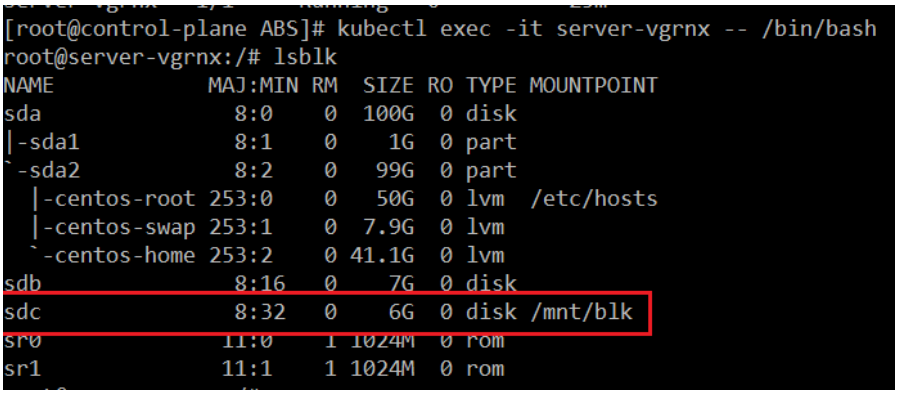
```
[root@control-plane ABS]# kubectl get pvc
NAME          STATUS   VOLUME                                     CAPACITY
claim1-test   Bound    pvc-9921713e-aa3f-4acb-ae58-c134f52f0c32   6Gi
```



- (Attach 체크) VM node에 해당 PVC volume이 block device로 인식되는지 체크합니다.

```
[root@control-plane ABS]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   99G  0 part
  ├─centos-root 253:0    0   50G  0 lvm  /
  ├─centos-swap 253:1    0  7.9G  0 lvm
  └─centos-home 253:2    0 41.1G  0 lvm  /home
sdb               8:16   0    7G  0 disk
sdc               8:32   0    6G  0 disk /var/lib/kubelet/p
ods/df383e81-bc7f-4b16-922e-41363407f10a/volumes/kubernetes
.io~csi/pvc-9921713e-aa3f-4acb-ae58-c134f52f0c32/mount
sr0              11:0    1 1024M  0 rom
sr1              11:1    1 1024M  0 rom
[root@control-plane ABS]#
```

- (Attach 체크) Pod에서 해당 PVC volume이 block device로 인식되는지 체크합니다.



[LVM Volume Provisioning]

Volume Group을 Nutanix Node의 LVM으로 관리하도록 provisioning 합니다.

1. iSCSI Volume Provisioning 방법의 7번까지 진행합니다.

2. LVM volume provisioning을 위한 StorageClass를 생성합니다.
   - StorageClass를 아래와 같이 설정한 후 생성합니다.
   - CMD: kubectl apply -f lvm/sc-lvm.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: lvm-sc
provisioner: csi.nutanix.com
parameters:
    csi.storage.k8s.io/provisioner-secret-name: ntnx-secret
    csi.storage.k8s.io/provisioner-secret-namespace: kube-system
    csi.storage.k8s.io/node-publish-secret-name: ntnx-secret
    csi.storage.k8s.io/node-publish-secret-namespace: kube-system
    csi.storage.k8s.io/controller-expand-secret-name: ntnx-secret
    csi.storage.k8s.io/controller-expand-secret-namespace: kube-system
    csi.storage.k8s.io/fstype: ext4
    dataServiceEndPoint: 172.25.0.13:3260
    storageContainer: volume-test2
    storageType: NutanixVolumes
    isLVMVolume: "true"
    numLVMDisks: "8"
allowVolumeExpansion: true
reclaimPolicy: Delete
```

   - iSCSI Volume StorageClass와 공통되는 필드는 iSCSI Volume과 마찬가지로 작성합니다.
   - parameters.isLVMVolume에는 “true”를 입력합니다.

- parameteres.numLVMDisks에는 해당 Volume을 위해 생성할 vDisk 개수를 입력합니다.

3. 해당 StorageClass로부터 볼륨을 할당받는 PVC를 생성합니다.
   - CMD: kubectl apply -f lvm/claim_lvm.yaml

```
[root@control-plane lvm]# kg pvc
NAME          STATUS   VOLUME                                     CAPACITY
claim-lvm     Bound    pvc-81c71dbd-e227-4f4f-99d1-bc0e88a1b3c0   2Gi
claim1-test   Bound    pvc-9921713e-aa3f-4acb-ae58-c134f52f0c32   6Gi
```

4. 해당 PVC를 사용하는 Pod을 iSCSI Volume Provisioning과 같은 방법으로 띄워 테스트합니다.

- (Attach 체크) VM Node에 해당 PVC volume이 생성되는 형태는 아래와 같습니다.

```
[root@control-plane lvm]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                   8:0    0  100G  0 disk
├─sda1                                8:1    0    1G  0 part /boot
└─sda2                                8:2    0   99G  0 part
  ├─centos-root                     253:0    0   50G  0 lvm  /
  ├─centos-swap                     253:1    0  7.9G  0 lvm
  └─centos-home                     253:2    0 41.1G  0 lvm  /home
sdb                                   8:16   0    7G  0 disk
sdc                                   8:32   0    6G  0 disk /var/lib/kubelet/pods/df383e81-bc7f-4b16-922e-41363407f10a/volumes/kubernetes.io~csi/pvc-9921713e-aa3f-4acb-ae58-c134
sdd                                   8:48   0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sde                                   8:64   0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdf                                   8:80   0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdg                                   8:96   0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdh                                   8:112  0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdi                                   8:128  0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdj                                   8:144  0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
sdk                                   8:160  0  256M  0 disk
└─d888aceb_b910_44e7_9810_6d88fab9b012-ntnxLV
                                    253:3    0    2G  0 lvm  /var/lib/kubelet/pods/920e3286-1bfd-4833-9d86-08942d746179/volumes/kubernetes.io~csi/pvc-81c71dbd-e227-4f4f-99d1-bc0e
```