OpenEBS

1. 온프레미스에서 OpenEBS 설치

1.1. OpenEBS 설치 전제 조건

* Admin context가 있는지 확인한다. 클러스터에 대한 관리자 권한이 없는 경우, 쿠버네티스 클러스터 관리자에게 OpenEBS 설치에 도움을 요청하거나 클러스터의 소유자인 경우 새 admin context를 만들고 OpenEBS를 설치한다.
* 쿠버네티스 1.18 버전 이상
* OpenEBS는 iSCSI 프로토콜을 통해 블록 볼륨 지원을 제공한다. 따라서 모든 쿠버네티스 노드에서 iSCSI initiator가 필요하다.

1.2. iSCSI initiator 설치

iSCSI initiator가 노드에 설치되지 않은 경우, 아래 명령을 통하여 설치한다.

Ubuntu

$ sudo apt-get update

$ sudo apt-get install open-iscsi

$ sudo systemctl enable –now iscsid

CentOS

$ yum install iscsi-initiator-utils -y

iSCSI initiator가 노드에 설치 후, iSCSI 서비스의 상태를 조회한다.

$ systemctl status iscsid

그림 1과 같이 iSCSI 데몬이 동작하는지 확인한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 1 iSCSI 상태 조회

1.3. OpenEBS 설치

OpenEBS는 쿠버네티스 클러스터에서 OpenEBS의 설정을 수정하고 실행할 수 있는 YAML 파일을 제공한다. Openebs-operator YAML 파일을 다운로드하고 구성을 업데이트한 후 아래 kubectl 명령에서 YAML 파일을 사용하여 설치한다. 이 명령은 Jiva와 로컬 PV (Persistent Volume) 구성 요소를 설치한다.

$ kubectl apply -f https://openebs.github.io/charts/openebs-operator.yaml

OpenEBS 네임스페이스의 파드 목록을 조회한다.

$ kubectl get pods -n openebs

그림 2와 같이 OpenEBS의 파드들이 동작하고 있음을 확인한다.

테이블이(가) 표시된 사진

자동 생성된 설명

그림 2 OpenEBS 파드 조회

1.4. cStor 데이터 엔진 설치

OpenEBS 설치 후 cStor 데이터 엔진을 설치한다.

$ kubectl apply -f https://openebs.github.io/charts/cstor-operator.yaml

1.5. cStor 풀 클러스터 생성

cStor 풀을 설정하는 데 사용해야 하는 노드와 디바이스의 세부 정보를 지정하여 CStorPoolCluster (CSPC)라는 쿠버네티스 CR (Custom Resource)을 만든다. 그림 3은 CSPC YAML 파일 예시이다. 클러스터의 세부 사항을 수정하여 사용한다.



그림 3 cspc.yaml

그림 4는 아래 명령어로 클러스터에 있는 모든 노드 라벨을 조회한다. CSPC yaml을 수정하는 데 노드 라벨이 필요하다.

$ kubectl get nodes –show-labels

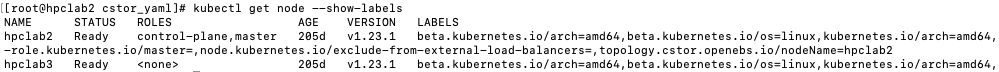


그림 4 노드 라벨 조회

CSPC YAML 파일을 수정하여 스토리지 풀을 생성할 노드에 연결된 블록 디바이스를 지정해야 한다. 그림 5는 아래 명령으로 각 워커 노드에서 사용 가능한 블록 디바이스를 확인하다.

$ kubectl get bd -n openebs

테이블이(가) 표시된 사진

자동 생성된 설명

그림 5 블록 디바이스 조회

수정한 CSPC YAML 파일을 이용하여 CSPC를 생성하고 생성된 CSPC를 조회한다.

$ kubectl apply -f cspc.yaml

$ kubectl get cspc -n openebs

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 6 CSPC 생성 및 조회

cStor 풀 인스턴스 (CSPI)를 조회한다.

$ kubectl get cspi -n openebs



그림 7 CSPI 조회

1.6. 스토리지 클래스 생성

1. 스토리지 클래스를 만들 CSPC를 설정한다.
2. 요구 사항/작업량에 따라 레플리카 카운트 (replica Count)를 설정한다. OpenEBS는 레플리카 카운트 설정을 제한하지 않지만, 최대 5개의 레플리카 카운트가 허용된다. 사용자가 어떻게 구성하는지에 따라 다르지만, 적어도 (n/2+1) 볼륨의 가용성을 위해 레플리카 카운트는 대상에 연결되어야 하며, 여기서 n은 레플리카 카운트다. 레플리카 카운트는 항상 CSPI 수보다 작거나 같아야 한다. 다음은 몇 가지 예시이다.

* 사용자가 레플리카 카운트를 2로 구성한 경우, 볼륨 작업을 수행하기 위해 항상 2개의 복제본을 사용할 수 있어야 한다.
* 사용자가 레플리카 카운트를 3으로 구성한 경우, 볼륨을 작동하려면 최소 2개의 복제본을 사용할 수 있어야 한다.
* 사용자가 레플리카 카운트를 5로 구성한 경우, 볼륨을 작동하려면 최소 3개의 복제본을 사용할 수 있어야 한다.

그림 8은 OpenEBS에서 제공하는 스토리지 클래스 YAML 파일이다. 제공된 파일을 수정하여 풀, 레플리카 카운트 및 기타 설정을 업데이트한다. 그림 8의 YAML 파일을 사용하면 3개의 OpenEBS cStor 복제본이 생성된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 8 cstor-csi-disk.yaml

스토리지 클래스 YAML 파일을 이용하여 스토리지 클래스를 생성한다.

$ kubectl apply -f cstor-csi-disk.yaml

생성된 스토리지 클래스의 상태를 조회한다.

$ kubectl get sc

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 9 스토리지 클래스 조회

1. AWS Storage를 사용한 OpenEBS 설치

2.1. 전제 조건

* EC2, S3 및 VPC에 대한 전체 액세스 권한이 있는 AWS 계정
* Ubuntu 16.04
* KOPS 설치
* AWS 계정 액세스를 위한 AWS CLI 설치
* EC2 인스턴스에 액세스하기 위해 생성된 SSH 키

2.2. AWS 환경 설정

가상 사설망 (VPC) 서비스로 가서 VPC를 생성한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 10 VPC 생성

인터넷 게이트웨이를 만들고 VPC를 이 인터넷 게이트웨이와 연결한다. 연결되면, 이 VPC 아래의 모든 노드는 외부 연결이 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 11 인터넷 게이트웨이 생성

로컬 Ubuntu에서 다음을 수행한다.

1. 로컬 컴퓨터에서 AWS CLI 유틸리티를 다운로드한다.
2. 아래 명령을 실행하여 AWS 계정에 연결한다.

$ aws configure

참고로 구성 세부 사항을 유지하려면 AWS 액세스 키, 시크릿 액세스 키, 기본 리전 이름 및 기본 출력 형식을 지정해야 한다.

클러스터 구성 세부 정보를 저장할 S3 버킷을 만든다.

$ aws s3 mb s3://<bucket\_name>

S3 버킷 세부 정보를 내보낸다.

$ export KOPS\_STATE\_STORE=s3://<bucket\_name>

클러스터를 생성한다.

$ kops create cluster – name=<cluster\_name>.k8s.local – vpc=<vpc\_id> - zones=<zone\_name>

명령 세트를 사용하여 클러스터 이름 변경, 노드용 인스턴스 그룹 및 마스터 등과 같은 클러스터 구성을 변경한다.

노드 구성을 수정한다.

$ kops edit ig – name=<cluster\_name>.k8s.local nodes

마스터 인스턴스 유형과 머신 수를 수정한다.

$ kops edit ig – name=<cluster\_name>.k8s.local master-<zone\_name>

사용자 지정이 완료되면, 변경 사항을 업데이트한다.

$ kops update cluster <cluster\_name>.k8s.local – yes

위의 단계는 AWS에 OpenEBS 클러스터를 배포한다. EC2 인스턴스 페이지를 찾고 해당 지역을 선택하여 인스턴스 생성 상태를 확인한다.

EC2 인스턴스 페이지에서 각 인스턴스 유형 Public IP를 조회한다.

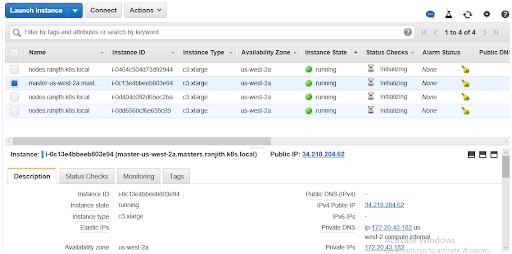


그림 12 Public IP 조회

EC2 페이지의 **Launch Configuration** 섹션으로 이동하여 노드의 Launch Configuration 사본을 가져온다. Configuration for Node group을 선택하고 **Actions**를 클릭한다.

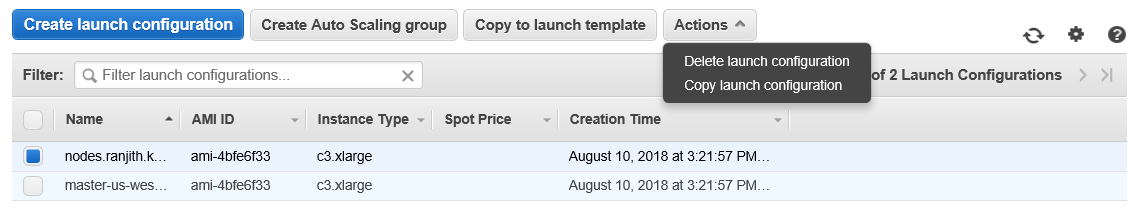


그림 13 노드 그룹 구성 선택

**Configure Details** 섹션에서 다음과 같이 설정한다.

필요한 경우, 새로운 configuration 이름으로 변경한다. **Advanced Details** 섹션을 수정하고 **User data** 섹션 끝에 아래 항목을 추가한다.

#!/bin/bash

$set -x  
$ date  
$ apt-get install open-iscsi  
$ grep “@reboot root sleep 120;service open-iscsi restart” /etc/crontab || sudo sh -c ‘echo “@reboot root sleep 120;service open-iscsi restart” >> /etc/crontab’  
$ systemctl enable open-iscsi  
$ sh -c ‘echo “/dev/xvdd /mnt/openebs\_xvdd auto defaults,nofail,comment=cloudconfig 0 2” >> /etc/fstab’  
$ reboot  
$ set -x  
$ umount /mnt/openebs/xvdd  
$ mount /dev/xvdd /mnt/openebs\_xvdd

**Skip**을 클릭하여 launch configuration 생성을 검토한다.

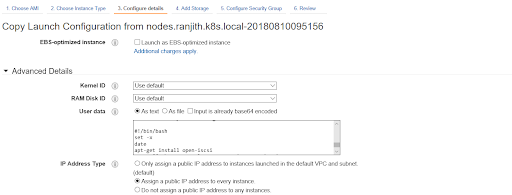


그림 14 Configure Details 설정

EC2 페이지의 Auto Scale Group 섹션으로 이동한다. 노드 그룹의 configuration을 선택하고 Actions을 클릭하여 Launch Configuration을 수정한다. 새로운 Launch Configuration으로 설정을 변경하고 새 설정을 저장한다.

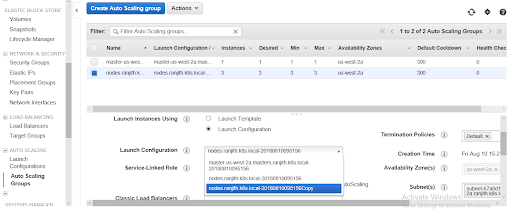


그림 15 Launch Configuration 설정

Public 키를 사용하여 각 노드에서 SSH를 실행한다.

$ ssh -i ~/ .ssh/id\_rsa admin@<public\_ip>

2.3. iSCSI initiator 설치 및 AWS 스토리지 마운트

OpenEBS가 설치될 모든 노드에 SSH를 실행하고, 다음 명령을 실행하여 재부팅 중에 iSCSI 패키지를 설치하고 로컬 디스크를 자동으로 마운트한다.

$ sudo apt-get update  
$ sudo apt-get install open-iscsi  
$ sudo service open-iscsi restart  
$ sudo cat /etc/iscsi/initiatorname.iscsi  
$ sudo service open-iscsi status  
$ sudo sudo sh -c ‘echo “/dev/xvdd /mnt/openebs\_xvdd auto defaults,nofail,comment=cloudconfig 0 2” >> /etc/fstab’  
$ grep “@reboot root sleep 120;service open-iscsi restart” /etc/crontab || sudo sh -c ‘echo “@reboot root sleep 120;service open-iscsi restart” >> /etc/crontab’  
$ sudo reboot

* 1. OpenEBS 설치

마스터 노드에서 다음 명령을 실행하여 OpenEBS YAML, 스토리지 클래스 YAML 파일을 가져온다.

$ wget https://raw.githubusercontent.com/openebs/openebs/v0.6/k8s/openebs-operator.yaml

$ wget https://raw.githubusercontent.com/openebs/openebs/v0.6/k8s/openebs-storageclasses.yaml

Openebs-operator.yaml을 수정하고 아래 항목을 추가한다. 호스트에 연결된 로컬 디스크 중 하나에 스토리지 풀이 생성된다.

apiVersion: openebs.io/v1alpha1  
kind: StoragePool  
metadata:  
name: jivaawspool  
type: hostdir  
spec:  
path: “/mnt/openebs\_xvdd”  
 — -

수정한 openebs-operator.yaml을 적용하여 OpenEBS를 사용한다.

$ kubectl apply -f openebs-operator.yaml

* 1. cStor 데이터 엔진 설치

cStor 데이터 엔진을 설치하기 위해 아래 명령어를 실행한다.

$ kubectl apply -f https://openebs.github.io/charts/cstor-operator.yaml

* 1. cStor 풀 클러스터 생성

cspc.yaml을 적용하여 CSPC를 생성한다.

$ kubectl apply -f cspc.yaml

* 1. 스토리지 클래스 생성

해당 스토리지 클래스에 아래 항목을 추가하여 openebs-storageclasses.yaml을 수정한다.

openebs.io/storage-pool: “jivaawspool”

apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
name: openebs-percona  
provisioner: openebs.io/provisioner-iscsi  
parameters:  
openebs.io/storage-pool: “default”  
openebs.io/jiva-replica-count: “3”  
openebs.io/volume-monitor: “true”  
openebs.io/capacity: 5G  
openebs.io/storage-pool: “jivaawspool”  
— -

openebs-storageclasses.yaml

수정한 openebs-storageclasses.yaml을 적용하여 스토리지 클래스를 생성한다.

$ kubectl apply -f openebs-storageclasses.yaml