kubernetes install by kubespray

2024.04.02 슈어데이터랩 조현우

개요

kubespray를 활용하여 kubernetes를 구축하고, rook을 사용하여 ceph을 올리기까지의 작업순서와 에러 대처방법 정리

버전정보

OS : xUbuntu_22.04 CRI-O : 1.25 kubernetes : v1.25.6 kubespray : 2.21

순서

0. 설치할 각종 소프트웨어의 호환성 조사
1. memory swap off
2. master node에 kubespray 설치
3. master node의 ssh key 생성 및 다른 node들에 copy
4. kubespray 설정(inventory.ini 혹은 host.yaml 사용)
5. ansible-playbook 명령어로 7에서 설정된 내용대로 클러스터 생성
6. Rook 설치, 설정
7. ceph 설정
8. ceph 올리기

설치할 각종 소프트웨어 호환성 조사

가장 중요한 것은, 각 소프트웨어의 버전이 서로 호환되어야 한다는 것이다.

필자의 경우 이를 확인하지 않고 설치하였는데, kubespray 2.2에서 설치하여 사용되는 calico의 버전이 kubernetes 1.25 버전과 호환되지 않아 오랜 시간 헤맸다.

이런 문제가 발생할 경우, 에러 로그가 엉뚱히 나와 직접적인 호환성과의 연관성을 찾기 매우 어려울 수 있으므로,

이를 방지하기 위해서 각 홈페이지에서 각 소프트웨어가 서로 어떤 버전이 호환되는지를 반드시 먼저 파악하고 정리한 후 시작해야 한다.

그 예시로, calico홈페이지의 `Kubernetes requirements` 탭에서, 호환되는 kubernetes의 버전을 확인할 수 있다.

memory swap off

쿠버네티스를 사용하려면 가장 기본적으로 모든 노드의 memory swap이 off 되어있어야 한다. 각 노드에서 아래 명령어로 swap을 끈다

```
sudo swapoff -a
```

개발서버 or 로컬에 kubespray 설치

kubespray가 알아서 kubectl, kubelet, kubeadm, cri-o, calico 등 Kubernetes 클러스터 구성에 필요한 소프트웨어를 설치하므로, 워커노드나 master node에 따로 각 소프트웨어를 다운받을 필요가 없다. 아래 명령어를 통해 kubespray를 로컬 혹은 개발서버에 설치한다.

```
sudo apt install python3
sudo apt update
sudo apt install -y python3-pip
sudo apt install -y git


git clone https://github.com/kubernetes-sigs/kubespray.git
cd kubespray/
sudo pip install -r requirements.txt
```

로컬 혹은 개발서버의 ssh key 생성 및 다른 node들에 copy

지금 실행할 것은 아니지만, 최종적으로 세팅이 완료된 후에 실행할 명령어는

`ansible-playbook -i inventory/test-cluster/inventory.ini --become --become-user=root cluster.yml`

이 명령어가 정상적으로 동작하는 조건은 다음과 같음

1. 명령어를 실행하는 로컬 혹은 개발서버에서, inventory.ini에 적어둔 모든 노드의 ip들에 ssh로 비밀번호 없이, ssh-key를 통해 접근이 가능할 것
2. 그렇게 접근한 유저가 sudo 명령어를 비밀번호 입력 없이 실행할 수 있을 것

따라서 1번부터 하나씩 차근차근 설정해보자.

명령어를 실행하는 로컬 혹은 개발서버에서, inventory.ini에 적어둔 모든 노드의 ip들에 ssh로 비밀번호 없이, ssh-key를 통해 접근이 가능하도록 설정

master node에서 ssh-key를 생성하고 대상 노드들에 복사한다.

```
ssh-keygen
ssh-copy-id 마스터노드1ip
ssh-copy-id 마스터노드2ip
ssh-copy-id 마스터노드3ip
ssh-copy-id 워커노드1ip
ssh-copy-id 워커노드2ip
ssh-copy-id 워커노드3ip
```

로컬 혹은 개발서버에서 ssh로 타 node들에 비밀번호 없이 접속되는지 확인

```

```

```
ssh 마스터노드1ip
ssh 마스터노드2ip
ssh 마스터노드3ip
ssh 워커노드1ip
ssh 워커노드2ip
ssh 워커노드3ip
```

각 노드의 해당 user가 sudo 명령어를 비밀번호 입력 없이 실행할 수 있도록 설정

각 노드에서, 다음 명령어로 파일을 연다

```
sudo visudo /etc/sudoers
```

그러면 다음과 같이 파일 내용이 나온다.

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/snap/bin"
Defaults        use_pty

# This preserves proxy settings from user environments of root
# equivalent users (group sudo)
#Defaults:%sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy
no_proxy"

# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults:%sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults:%sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults:%sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_*"

# Per-user preferences; root won't have sensible values for them.
#Defaults:%sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"
```

```
# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) NOPASSWD:ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

이 중 중요한 내용은 밑에있는 두 줄,

`%sudo ALL=(ALL:ALL) ALL` 와 `@includedir /etc/sudoers.d` 이다.

이 부분에서 `%sudo ALL=(ALL:ALL) ALL` 다음 줄에, 다음 내용을 추가해야 한다.

`유저이름 ALL=(ALL) NOPASSWD:ALL`

`%sudo ALL=(ALL:ALL) ALL` 보다 아래에 위치에 추가하는 이유는, 아마도 이 유저가 이미 sudo group 소속일텐데, `%sudo ALL=(ALL:ALL) ALL` 보다 위에 저 내용을 추가해버리면, 더 아래줄에 적힌 내용이 덮어씌워져버리므로 의미가 없어져버리기 때문이다.

하지만 위 방법처럼 추가하지 말고, 왠만하면 `@includedir /etc/sudoers.d` 부분을 활용하는 것이 좋다.

sudoers.d 디렉토리는 유저가 다양할 때 파일로 나누어 관리하기 위한 것으로, 유저가 많을 시 본 가이드에서 아까 제시한 방식 대신 이 방법으로 관리하는것이 적극 권장된다.

따라서 해당 디렉토리 내에 해당 유저명 혹은 그룹명으로 파일을 만들어 그 안에 관련 설정들을 넣어 관리하는것이 좋다.

이번 예시의 경우라면, 필자의 유저명은 `test`이므로,

test 라는 파일을 만들고, 내부에 `test ALL=(ALL) NOPASSWD:ALL` 라는 내용을 추가하면 된다.

물론 파일명은 `~` 혹은 `.` 으로 시작하지않는 모든 파일을 포함하기때문에 편하게 만들어도된다.

그리고 가능은 되지만 하지 말아야 할 방법을 소개하겠다.

아마 이미 해당 유저가 sudo 그룹에 소속되어으니, `%sudo ALL=(ALL:ALL) ALL` 부분을

`%sudo ALL=(ALL:ALL) NOPASSWD:ALL`

로 수정한다면 sudo group에 포함된 모든 사용자가 비밀번호 없이 사용이 가능할 것이다. 하지만 모든 sudo 그룹 사용자가 이렇게 되는 것 보단 위 방법이 좋으니, 그렇게 하자.

kubespray 설정(inventory.ini 혹은 hosts.yaml 사용)

본 예시에서는 inventory.ini를 사용하겠다. kubespray 디렉토리 위치에서, 다음 명령들을 순서대로 실행한다.

```
cp -rfp inventory/sample/ inventory/test-cluster
cd inventory/test-cluster
declare -a IPS=(마스터노드1ip 마스터노드2ip 마스터노드3ip 워커노드1ip 워커노드2ip 워커노드
3ip)
CONFIG_FILE=inventory/test-cluster/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

물론 위 워커노드 ip들은 실제 자신의 노드 ip에 맞게 설정해야 한다.

이러고 나면 `inventory/test-cluster/hosts.yaml` 파일을 열어보면 해당 사항들이 반영되어있는 것을 확인할 수 있을 것이다.

이 내용에 맞춰, `inventory/test-cluster/inventory.ini` 파일을 다음과 같이 수정하자.

```
[kube_control_plane]
[etcd]
```

두 그룹안에 각각 마스터노드와 etcd로 사용할 노드를 배치하고, `[kube_node]` 안에 워커노드로 사용할 노드를 배치하면 된다.

아래는 그 구체적 예시이다.

```
[all]
master1 ansible_host=192.168.***.***
master2 ansible_host=192.168.***.***
master3 ansible_host=192.168.***.***
node1 ansible_host=192.168.***.***
node2 ansible_host=192.168.***.***
node3 ansible_host=192.168.***.***


[kube_control_plane]
master1 ansible_host=192.168.***.***
master2 ansible_host=192.168.***.***
master3 ansible_host=192.168.***.***

[etcd]
master1 ansible_host=192.168.***.***
master2 ansible_host=192.168.***.***
master3 ansible_host=192.168.***.***
```

```
[kube_node]
node1 ansible_host=192.168.***.***
node2 ansible_host=192.168.***.***
node3 ansible_host=192.168.***.***

[calico_rr]

[k8s_cluster:children]
kube_control_plane
kube_node
calico_rr
```

중요 그 후 본 kubespray v2.21 만의 오류로 수정해야 하는 부분이 하나 더 있다.

`vi inventory/test-cluster/group_vars/k8s_cluster/k8s-cluster.yml` 를 열어서,

`remove_default_searchdomains: false` 이 부분을 찾아 주석을 해제해야 한다.

그 이유는, 다음 사이트에서 확인할 수 있는데,

단순히 kubespray 개발자들이 저 부분의 기본값을 실수로 true로 설정해두었기 때문이다.

crio 사용 설정

kubespray v2.21은 기본적으로 containerd를 사용하게 되어있다. 이를 cri-o로 수정하기 위해서는 다음과 같이 파일을 수정해야 한다.

`inventory/test-cluster/group_vars/all/all.yml` 파일에서,

```
download_container: false
skip_downloads: false
etcd_deployment_type: host # optionally kubeadm
```

로 수정 혹은 내용 추가를 해야 한다.

또한, `inventory/test-cluster/group_vars/k8s_cluster/k8s_cluster.yml` 파일에서,

`container_manager: crio` 로 수정해야한다. 기본값으로는 `container_manager: containerd` 로 되어있을 것이다.

`inventory/test-cluster/group_vars/all/crio.yml` 파일의 내용을 다음과 같이 수정해야 한다.

```
crio_registries:
  - prefix: docker.io
    insecure: false
    blocked: false
    location: registry-1.docker.io
    unqualified: false
    mirrors:
```

```
        - location: 192.168.100.100:5000
          insecure: true
        - location: mirror.gcr.io
          insecure: false
```

공식문서

필요한 경우 pids_limit을 올리려면 `roles/container-engine/cri-o/defaults/main.yml` 에 있는 `crio_pids_limit` 값을 수정하면 된다.

ansible-playbook 명령어로 7에서 설정된 내용대로 클러스터 생성

위 설정에 혹시 예전에 이미 클러스터를 만든게 있다면, 그걸 삭제하기 위해 다음 명령어를 실행한다.

```
ansible-playbook -i inventory/test-cluster/inventory.ini --become --become-user=root reset.yml
```

이때 reset 혹은 을 했다면, 필자의 경우 이유는 모르지만 일정 확률로 master node나 workder node중 몇몇 도메인 서버 정보가 삭제되어 nslookup naver.com 등이 작동되지 않게 되는 현상이 있었다. 그런 경우,

Truble shooting

이때 ansible이 각 노드의 dns 설정을 건드리며 아래와 같은 오류가 발생할 수 있다. 증상과 해결방법은 다음과 같다.

1. 일정 확률로, 도메인 서버가 고장남

이 경우를 확인하는 것은, 다음 명령어로 확인이 가능하다.

```
nslookup google.com ping 8.8.8.8 dig +dnssec google.com
```

nslookup으로 정상적으로 구글의 ip가 반환되지 않고 에러가 나면서, `ping 8.8.8.8` 은 제대로 수행되고,

`dig +dnssec google.com` 에서 status가 SERVFAIL이 나온다면, ansible가 dns 설정을 중간에 잘못 건드려 발생한 오류이다.

이 경우 다음과 같이 다시 고칠 수 있다.

1. `sudo nano /etc/systemd/resolved.conf` 로 파일을 열고, `DNS=` 의 주석을 해제하고 해당 줄을 `DNS=8.8.8.8`로 수정한다.
2. `sudo systemctl restart systemd-resolved`를 입력하여 해당사항을 반영하고, `nslookup google.com` 을 입력하여 정상적으로 값이 나오는 것을 확인한다.
3. 1번에서 주석을 해제했던 부분을 원상복귀(주석처리)하고, 다시 `sudo systemctl restart systemd-resolved` 를 입력하여 해당 사항을 적용한다.

이러고 나면 정상적으로 다시 nslookup google.com이 수행되는 것을 확인할 수 있다.

계속 진행하자.

이제 클러스터를 생성하는 명령어를 실행한다.(물론 아까와 동일하게 kubespray 디렉토리에서 입력)

```
ansible-playbook -i inventory/test-cluster/inventory.ini --become --become-user=root cluster.yml
```

정상적으로 설치되었는지 확인

```
mkdir ~/.kube
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
kubectl get nodes
```

이때, 다음과 같이 출력되면 정상적으로 설치 된 것이다.

```
NAME     STATUS   ROLES           AGE     VERSION
master1  Ready    control-plane   4h53m   v1.25.6
master2  Ready    control-plane   4h53m   v1.25.6
master3  Ready    control-plane   4h53m   v1.25.6
node1    Ready    <none>          4h53m   v1.25.6
node2    Ready    <none>          4h53m   v1.25.6
node3    Ready    <none>          4h53m   v1.25.6
```

하지만 클러스터가 정상적으로 만들어졌다 하더라도, 아직은 안심할 수 없다. 각종 kubernetes의 시스템을 구성하는 요소들이 잘 초기화되고 실행되는지 확인해야 한다.

따라서 k9s를 설치하여 나머지 Kube-system 항목들이 잘 초기화되고있는지 확인하자

Node를 더 추가하려면?

kubernetes를 사용하는 것이니 당연히 Node를 추가하거나 줄일 일이 있을 수 있다. 이 경우, 다음과 같이 한다.

1. 추가할 node에 ssh-key-copy
2. inventory,hosts.yaml 수정
3. (추가하는 경우)추가할 Node에도 해당 사용자가 sudo 명령어를 비밀번호 없이 사용할 수 있도록 설정
4. ansible로 inventory의 수정사항을 반영하여 scale하기

하나씩 보자

inventory 수정

```
declare -a IPS=(ip1 ip2 ip3 ip4)
CONFIG_FILE=inventory/mycluster/hosts.yaml python3 contrib/inventory_builder/inventory.py ${IPS[@]}
```

(추가하는 경우)추가할 Node에도 해당 사용자가 sudo 명령어를 비밀번호 없이 사용할 수 있도록 설정

알아서 위에 작성된 가이드를 참고하여 설정하자.

ansible로 inventory의 수정사항을 반영하여 scale하기

inventory.ini파일도 수정한 후, 다음 명령어를 통해 scale할 수 있다.

```
ansible-playbook -i inventory/test-cluster/inventory.ini --become --become-user=root scale.yml
```

다만 필자의 경우 여기서 에러가 발생하며 정상적으로 동작하지 않았는데, 그 이유는 필자가 사용한 main branch의 kubespray에 버그가 있었다.

오류내용은 다음과 같다.

```
'kubeadm_images_raw' is undefined
... 이하 생략
```

만약 필자와 동일한 오류가 있다면(아직은 main에 merge되지 않아 해당 버그가 존재하나, 이후 버전에서는 해결될 것으로 보인다)다음과 같이 수정해주면 된다.

Rook 설치, ceph 올리기

홈피링크

```
git clone --single-branch --branch v1.13.7
https://github.com/rook/rook.git
cd rook/deploy/examples
kubectl create -f crds.yaml -f common.yaml -f operator.yaml
kubectl create -f cluster.yaml
```

유틸 설치, 동작 확인

```
kubectl -n rook-ceph get pod
cd ../../

(rook 디렉토리에서 다음을 실행)

kubectl create -f deploy/examples/toolbox.yaml
kubectl -n rook-ceph rollout status deploy/rook-ceph-tools
```

모니터링 쉘 실행

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
```

해당 쉘 내에서,

```
ceph status
```

를 통해 상태를 확인할 수 있다.

만약, worker node가 3개보다 적다면, ceph의 정책상 동작이 정상적으로 돌아가고있지 않음을 확인할 수 있다.

default 설정으로 최소 3개의 워커노드에 각각의 모니터가 running상태로 등록되어야하는데, 이것이 잘 되고있는지는 다음 명령어로 확인할 수 있다.

```
kubectl -n rook-ceph get pods -l app=rook-ceph-mon
```