

ETCD Cluster Initialize

[Kakao Cloud School S1 Dev. CptBluebear]

실습환경: Ubuntu 20.04 LTS

제약 및 참고사항

1. etcdadm 을 실행 시 root 권한으로 수행해야 함. 이는 system service를 만들기 때문이다.
2. etcd v.2는 지원하지 않는다.
3. 리눅스의 기본적인 환경의 구성은 완료되었다고 가정한다. (시간 동기화, hostname, hosts 등)
4. 별도의 언급이 없다면 상대경로의 기준은 사용자의 home directory다.
5. etcd 클러스터 구성 시 클러스터를 구성하는 노드의 개수는 홀수여야 한다.
6. Hosts 파일은 다음과 같다. 이를 기반으로 hostname을 설정한다. ip는 각자 환경에 맞게 설정한다. 핵심은 클러스터간 hosts 파일의 동기화이다.

```
cptbluebear@etcd-01:~$ cat /etc/hosts
127.0.0.1        localhost
127.0.1.1        etcd-01

192.168.131.201  etcd-01
192.168.131.202  etcd-02
```

[사전 환경 구성]

1. golang 컴파일러, make 설치

etcdadm의 경우 golang으로 소스코드가 작성되어있다. 해당 소스코드를 실행 바이너리로 만들어 사용하기 위해서는 실행 환경에 맞게 컴파일을 수행해야 한다.

다음과 같은 명령을 수행하여 golang 컴파일러 및 make를 ubuntu 환경에 설치한다.

```
sudo apt update
sudo apt install -y golang make
```

```
cptbluebear@etcd-01:~$ sudo apt update
[sudo] password for cptbluebear:
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Hit:2 http://kr.archive.ubuntu.com/ubuntu focal InRelease
cptbluebear@etcd-01:~$ sudo apt install -y golang make
Reading package lists... Done
Building dependency tree
```

그 후, 설치 확인을 위해 다음과 같은 명령어를 실행하여 결과를 확인한다.

```
golang version
make --version
```

```
cptbluebear@etcd-01:~$ go version
go version go1.13.8 linux/amd64
cptbluebear@etcd-01:~$ make --version
GNU Make 4.2.1
```

ETCD Cluster Initialize

[etcdadm 설치 및 ETCD Cluster 구성]

etcdadm은 etcd-cluster를 구성하기 위한 커맨드 라인 도구이며, etcd 클러스터를 보다 쉽게 구성하고 관리할 수 있도록 도와주는 프로그램이다. 해당 프로그램은 kubespray, krew 등을 개발한 Kubernetes SIGs 에서 개발하였으며, kubeadm의 사용자 경험을 기반으로 etcdadm을 개발하였다고 밝혔다.

1. etcdadm 소스코드 다운로드

etcdadm의 github repository의 주소는 다음과 같다.

<https://github.com/kubernetes-sigs/etcdadm>

해당 git repository를 본인이 원하는 경로에 배치한다. 그 뒤 소스코드 디렉터리로 이동한다. 소스코드를 다운로드 받기 위해 git clone 명령을 이용할 수 있다.

```
git clone https://github.com/kubernetes-sigs/etcdadm.git
```

2. etcd 빌드

etcd를 빌드하기 위해 소스 폴더 내부에 있는 MakeFile을 이용한다. 빌드를 하는 방법에는 host환경을 이용하는 것과 docker를 이용해 컨테이너 환경에서 빌드한 뒤 실행 바이너리를 host 환경으로 가져오는 방법이 있다. 본 워크북에서는 전자를 사용한다.

다음과 같은 명령을 실행하여 etcdadm을 빌드한다.

* 주의! 사전에 golang 및 make가 환경에 설치되어 있어야 한다.

```
sudo make etcdadm
```

```
cptbluebear@etcd-01:~/etcdadm$ sudo make etcdadm
G0111MODULE=on go build -ldflags "-X 'k8s.io/component-base/version.buildDate
d7c6502c63f7aff35ce342c4e8' -X 'k8s.io/component-base/version.gitTreeState=cl
k8s.io/component-base/version.gitMajor=0' -X 'k8s.io/component-base/version.g
```

ETCD Cluster Initialize

3. etcdadm 실행 바이너리 절대경로에 추가

etcdadm 파일을 절대경로에 추가한다. 본 워크북에서는 /usr/local/sbin 디렉터리에 etcdadm 실행 바이너리를 추가한다.

```
sudo cp ./etcdadm /usr/local/sbin/  
ls -al /usr/local/sbin/ | grep etcdadm
```

```
cptbluebear@etcd-01:~/etcdadm$ sudo cp ./etcdadm /usr/local/sbin/  
cptbluebear@etcd-01:~/etcdadm$ ls -al /usr/local/sbin/ | grep etcdadm  
-rwxr-xr-x 1 root root 23237070 10월  6 17:28 etcdadm
```

4. etcdadm을 통한 etcd cluster 구성

etcdadm 실행 바이너리를 이용하여 etcd 클러스터를 구성한다.

etcdadm의 상세 사용법은 [etcdadm help] 명령을 통해 확인할 수 있다.

다음과 같은 명령으로 etcd를 설치한다. 작성일 22.10.06 기준 최신 버전인 v3.5.1이 설치된다.

- * 주의! 아이피는 각자 환경에 맞게 변경해야한다. 또한, etcdadm은 해당 시스템에 하나의 NIC만 있다면 상관 없지만, 두개 이상의 NIC가 있을 경우 원하는 IP로 서비스가 바인드 되지 않을 수 있기에 별도로 --bind-address 플래그를 이용해 직접 바인딩될 IP를 지정해야 한다.
또한, 실행 마지막 줄의 etcdadm join 명령을 복사해놓는다.

```
sudo etcdadm init --bind-address 192.168.131.201
```

```
cptbluebear@etcd-01:~/etcdadm$ sudo etcdadm init --bind-address 192.168.131.201  
INFO[0000] [install] extracting etcd archive /var/cache/etcdadm/etcd/v3.5.1/etcd-v3.5.1-linux-amd64.tar.gz to /tmp/etcd509073984  
INFO[0000] [install] verifying etcd 3.5.1 is installed in /opt/bin/  
INFO[0000] [certificates] creating PKI assets  
INFO[0000] creating a self signed etcd CA certificate and key files  
[certificates] Generated ca certificate and key.  
INFO[0000] creating a new server certificate and key files for etcd  
[certificates] Generated server certificate and key.  
[certificates] server serving cert is signed for DNS names [etcd-01] and IPs [192.168.131.201 127.0.0.1]  
INFO[0000] creating a new certificate and key files for etcd peering  
[certificates] Generated peer certificate and key.  
[certificates] peer serving cert is signed for DNS names [etcd-01] and IPs [192.168.131.201]  
INFO[0000] creating a new client certificate for the etcdctl  
[certificates] Generated etcdctl-etcd-client certificate and key.  
INFO[0000] creating a new client certificate for the apiserver calling etcd  
[certificates] Generated apiserver-etcd-client certificate and key.  
[certificates] valid certificates and keys now exist in "/etc/etcd/pki"  
INFO[0002] [health] Checking local etcd endpoint health  
INFO[0002] [health] Local etcd endpoint is healthy  
INFO[0002] To add another member to the cluster, copy the CA cert/key to its certificate dir and run:  
INFO[0002] etcdadm join https://192.168.131.201:2379
```

etcdadm이 수행해주는 동작은 다음과 같다.

1. etcd 바이너리 다운로드 후 압축 해제, /opt/bin/ 경로에 배치
2. CA 인증서 등, etcd에서 필요한 여러 인증서와 키 생성
3. etcd 서비스 생성 및 systemd에 등록
4. etcd 서비스 활성화 및 실행
5. etcd 정상 실행 여부를 확인하기 위해 endpoint health로 health check 수행

ETCD Cluster Initialize

5. etcdctl 설치

etcd를 제어하기 위한 etcdctl을 설치한다. 다음과 같은 명령을 통해 설치한다.

```
sudo apt install -y etcd-client
```

```
cptbluebear@etcd-01:~/etcdadm$ sudo apt install -y etcd-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

6. etcdctl 관련 환경 설정 (옵션)

etcd에 작업을 전달하는 etcdctl은 사용 시, etcd 서버에 대한 인증서, 인증키가 필요하다. 또한 etcdctl API Version 또한 지정해주어야 하는데 이러한 요소를 사전에 설정함으로써 이후 etcdctl 사용시 편의성을 얻을 수 있다. 만약 이 설정을 하지 않는다면 etcdctl 실행 시 항상 인증키를 플래그 옵션으로 넣어주어야 한다. 인증키 관련 설명은 kubeadm 워크북에서 서술한다.

아래의 명령을 실행하여 /etc/bash.bashrc에 alias를 등록한 뒤 쉘에 적용한다.

```
sudo su
echo alias sudo=W'sudo W'>> /etc/bash.bashrc
echo alias etcdctl=W'ETCDCTL_API=3 etcdctl --cacert /etc/etcd/pki/ca.crt --cert /etc/etcd/pki/etcdctl-etcd-client.crt --key /etc/etcd/pki/etcdctl-etcd-client.key W'>> /etc/bash.bashrc
exit
source /etc/bash.bashrc
```

7. 인증서 복제

ETCD Cluster의 각 노드는 동일한 인증서를 보유해야 한다. 그렇기 때문에 첫번째 노드에서 사용중인 ca.crt 파일과 ca.key 파일을 다른 노드들로 배포해야 한다. 해당 파일들을 적절히 다른 노드들의 /etc/etcd/pki/ 경로에 복제한다. scp를 이용하는 것을 추천한다.

* 주의! 옮기기 전/후 키 파일의 권한이 적절한지, 해당 경로가 생성되어있는지 확인하여야 한다.

ETCD Cluster Initialize

예시)

기본적으로 scp를 이용한 복제 시 root ssh 로그인은 차단되어 있다. 그렇기에 1번 서버의 인증키 파일을 일반 유저 권한으로 복제 후 이를 scp를 통해 2번 서버에 전송한다. 그 뒤, 2번 서버에서 해당 파일의 권한을 다시 root로 변경한 후 적절한 위치에 배치하면 작업이 완료된다.

아래는 필자가 scp를 이용해서 키 파일을 옮길 때 사용한 명령어이다.

scp 명령어 말고 rsync 명령어를 이용해도 좋다. 본 작업은 etcd에 종속되는 작업이 아니라 리눅스 운영체제를 이용하는 작업이니 어떻게든 해당 인증키 파일들을 적절한 위치에, 적절한 권한을 가지게 하여 배치하기만 하면 된다.

```
sudo cp -r /etc/etcd/pki ./
sudo chown -R <일반유저>.<일반유저> ./pki
scp -r ./pki/ <일반유저>@[etcd주소]:/home/<일반유저>/pki
sudo mkdir -p /etc/etcd/pki
sudo cp ./pki/ca.crt /etc/etcd/pki/
sudo cp ./pki/ca.key /etc/etcd/pki/
sudo chown root.root /etc/etcd/pki
```

8. etcdadm join 을 통한 etcd 노드를 클러스터에 추가

두 번째 etcd 노드 또한 본 워크북의 1-3번 과정을 통해 etcdadm이 준비되었다고 가정한다.

앞서 복사해둔 etcdadm join 명령어를 수행한다. join의 경우에도 NIC가 두개 이상인 경우를 대비하여 --bind-address 플래그 옵션을 붙인다.

```
sudo etcdadm join https://192.168.131.201:2379 --bind-address 192.168.131.202
```

```
cpptbluebear@etcd-02:/etc/etcd/pki$ sudo etcdadm join https://192.168.131.201:2379 --bind-address 192.168.131.202
INFO[0000] [certificates] creating PKI assets
INFO[0000] creating a self signed etcd CA certificate and key files
[certificates] Using the existing ca certificate and key.
INFO[0000] creating a new server certificate and key files for etcd
[certificates] Generated server certificate and key.
[certificates] server serving cert is signed for DNS names [etcd-02] and IPs [192.168.131.202 127.0.0.1]
INFO[0000] creating a new certificate and key files for etcd peering
[certificates] Generated peer certificate and key.
[certificates] peer serving cert is signed for DNS names [etcd-02] and IPs [192.168.131.202]
INFO[0000] creating a new client certificate for the etcdctl
[certificates] Generated etcdctl-etcd-client certificate and key.
INFO[0000] creating a new client certificate for the apiserver calling etcd
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] valid certificates and keys now exist in "/etc/etcd/pki"
INFO[0000] [membership] Checking if this member was added
INFO[0000] [membership] Member was not added
INFO[0000] Removing existing data dir "/var/lib/etcd"
INFO[0000] [membership] Adding member
INFO[0000] [membership] Checking if member was started
INFO[0000] [membership] Member was not started
INFO[0000] [membership] Removing existing data dir "/var/lib/etcd"
INFO[0000] [install] extracting etcd archive /var/cache/etcdadm/etcd/v3.5.1/etcd-v3.5.1-linux-amd64.tar.gz to /tmp/etcd250439659
INFO[0000] [install] verifying etcd 3.5.1 is installed in /opt/bin/
INFO[0009] [health] Checking local etcd endpoint health
INFO[0009] [health] Local etcd endpoint is healthy
```

ETCD Cluster Initialize

9. 첫 번째 노드에서 수행했던 부가 작업들을 필요에 따라 수행한다.

그리고 8번 같은 과정을 통해 다른 노드들 또한 ETCD Cluster에 추가할 수 있다.

10. ETCD Cluster의 각 노드의 상태를 확인한다.

```
sudo etcdctl -w table --endpoints=etcd-01:2379,etcd-02:2379,etcd-03:2379 endpoint status
```

```
cptbluebear@etcd-02:~$ sudo etcdctl -w table --endpoints=etcd-01:2379,etcd-02:2379,etcd-03:2379 endpoint status
```

ENDPOINT	ID	VERSION	DB SIZE	IS LEADER	RAFT TERM	RAFT INDEX
etcd-01:2379	94b39c67e7d19097	3.5.1	20 kB	true	11	21
etcd-02:2379	44e7e0130befb77	3.5.1	20 kB	false	11	21
etcd-03:2379	f45f2dbe1b0059d0	3.5.1	25 kB	false	11	21

ETCD Cluster Initialize

[ETCD Cluster 모니터링]

etcd는 기반 스토리지로 Kubernetes Cluster를 구성하는 요소 중 가장 중요한 것이라고 볼 수 있다. Kubernetes Cluster에서 사용되는 모든 데이터가 etcd에 보관되기 때문이다. 그렇기 때문에 ETCD Cluster를 모니터링 하여 클러스터의 상태나, 변화, 가용성에 이상이 발생할 여지가 있는지 지속적으로 확인해야 한다.

대표적인 모니터링 도구라면 Prometheus 와 Grafana를 생각할 수 있다. Prometheus는 시계열 데이터를 처리 가능한 데이터베이스이며, Grafana는 Prometheus DB에서 데이터를 읽어 시각화 해주는 도구이다.

ETCD Cluster에서는 Prometheus가 각 노드에서 데이터를 수집하여 Prometheus에 저장한 뒤 이를 Grafana를 통해 모니터링을 수행하는 것이 주요 모니터링 방법 중 하나이다. 주의할 점은, 각 노드에서 정보를 보내주는 것이 아닌, Prometheus가 정보를 읽어온다는 것이다.

그렇다면 각 노드는 이러한 정보 - 즉, Metric 데이터를 보여줘야 한다는 것인데 일반적으로 Exporter라는 것을 시스템에 추가하여 처리한다. 하지만, etcd는 자체적으로 metric 데이터를 보여주는 EndPoint를 제공하고 있다.

본 워크북에서는 EndPoint를 활성화 하여 Prometheus 및 Grafana와 연동한다.

1. etcd 실행 시의 ENV 수정 후 서비스 재시작

Metric EndPoint를 활성화 하기 위해서는 etcd를 실행할 때 특정 환경변수를 부여해야 한다. 우선 etcd.service를 확인해본다. 여기서 중요하게 볼 것은 8번 라인이다.

```
1 [Unit]
2 Description=etcd
3 Documentation=https://github.com/coreos/etcd
4 Conflicts=etcd-member.service
5 Conflicts=etcd2.service
6
7 [Service]
8 EnvironmentFile=/etc/etcd/etcd.env
9 ExecStart=/opt/bin/etcd
10
11 Type=notify
12 TimeoutStartSec=0
13 Restart=on-failure
14 RestartSec=5s
15
```

해당 라인을 해석해보면, etcd가 실행될 때 /etc/etcd/etcd.env 파일에서 환경변수를 읽어들인다. 그렇기에 해당 파일을 수정하여 Metric EndPoint를 활성화 한다.

ETCD Cluster Initialize

/etc/etcd/etcd.env 파일의 적당한 위치에 다음과 같이 추가한다. 아래의 그림에서 30번째 줄을 참고하면 된다. 본 워크북은 2381 포트를 이용한다.

```
ETCD_LISTEN_METRICS_URLS=http://192.168.131.201:2381
```

```
26 # Other
27 ETCD_DATA_DIR=/var/lib/etcd
28 ETCD_STRICT_RECONFIG_CHECK=true
29 GOMAXPROCS=4
30 ETCD_LISTEN_METRICS_URLS=http://192.168.131.201:2381
31
```

환경변수를 수정하였기에 다음과 같은 명령을 수행하여 etcd 서비스를 재시작 한다.

```
sudo systemctl restart etcd.service
```

그렇다면 입력한 주소에서 metric 정보를 확인할 수 있다. 직접 확인하고 싶다면 다음과 같이 접속한다.

<http://<Cluster-IP>:2381/metrics>

```
# HELP etcd_cluster_version Which version is running. 1 for 'cluster_version' label with current cluster version
# TYPE etcd_cluster_version gauge
etcd_cluster_version{cluster_version="3.5"} 1
# HELP etcd_debugging_auth_revision The current revision of auth store.
# TYPE etcd_debugging_auth_revision gauge
etcd_debugging_auth_revision 1
# HELP etcd_debugging_disk_backend_commit_rebalance_duration_seconds The latency distributions of commit.rebalance called by bbolt db backend.
# TYPE etcd_debugging_disk_backend_commit_rebalance_duration_seconds histogram
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.001"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.002"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.004"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.008"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.016"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.032"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.064"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.128"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.256"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="0.512"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="1.024"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="2.048"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="4.096"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="8.192"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_bucket{le="+Inf"} 4
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_sum 0
etcd_debugging_disk_backend_commit_rebalance_duration_seconds_count 4
```

해당 EndPoint에서 etcd 노드의 Metric 정보를 확인할 수 있다.

이 작업을 ETCD Cluster 내부의 모든 노드에서 수행하여 Metric 정보를 조회할 수 있도록 한다.

ETCD Cluster Initialize

2. 모니터링 서버에 Prometheus 설치

Prometheus 공식 홈페이지에 가서 다운로드 링크를 얻는다.(<https://prometheus.io/download/>)

본 워크북 작성일 22.10.06 기준 LTS 버전인 2.37.1을 선택한다.

현재 환경에 알맞은 linux-amd64를 선택하여 다운로드 링크를 얻는다.

<https://github.com/prometheus/prometheus/releases/download/v2.37.1/prometheus-2.37.1.linux-amd64.tar.gz>

그 후, 모니터링 서버에서 해당 파일을 다운로드 받아 압축을 해제하기 위해 다음 명령을 수행한다.

wget

<https://github.com/prometheus/prometheus/releases/download/v2.37.1/prometheus-2.37.1.linux-amd64.tar.gz>

tar xzf prometheus-2.39.0.linux-amd64.tar.gz

cd prometheus-2.39.0.linux-amd64

```
cptbluebear@monitoring:~/Desktop$ wget https://github.com/prometheus/prometheus/releases/download/v2.37.1/prometheus-2.37.1.linux-amd64.tar.gz
--2022-10-07 01:35:40-- https://github.com/prometheus/prometheus/releases/download/v2.37.1/prometheus-2.37.1.linux-amd64.tar.gz
Resolving github.com (github.com)... 20.200.245.247
Connecting to github.com (github.com)|20.200.245.247|:443... connected.
HTTP request sent, awaiting response... 302 Found
cptbluebear@monitoring:~/Desktop$ tar xzf prometheus-2.37.1.linux-amd64.tar.gz
cptbluebear@monitoring:~/Desktop$ cd prometheus-2.37.1.linux-amd64/
cptbluebear@monitoring:~/Desktop/prometheus-2.37.1.linux-amd64$ ls
console_libraries  consoles  LICENSE  NOTICE  prometheus  prometheus.yml  promtool
```

해당 디렉터리 내에는 Prometheus의 실행 바이너리가 이미 포함되어 있기에, 별도로 컴파일 과정을 거치지 않아도 된다.

prometheus.yml 파일을 수정한다.

마지막 줄의 scrape_configs.static_config.target 항목에 이전에 생성해둔 etcd노드의 Metric 정보를 제공하는 EndPoint의 주소를 기입한 뒤 저장한다.

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["192.168.131.201:2381", "192.168.131.202:2381", "192.168.131.203:2381"]
```

ETCD Cluster Initialize

prometheus 실행 파일을 절대 경로에 추가한다. 또한, 설정파일 또한 적절한 위치에 배치한다. 이를 위해 다음과 같은 명령을 수행한다.

```
sudo cp ./prometheus /usr/local/sbin/  
sudo mkdir -p /etc/prometheus  
sudo cp ./prometheus.yml /etc/prometheus/
```

그 후, prometheus 유저를 생성하여 필요한 파일들에 권한을 부여한다. 또한, prometheus의 작업 영역을 위해 추가적인 디렉터리도 생성한다. 이를 위해 다음과 같은 명령을 수행한다.

```
sudo useradd -m -s /bin/bash prometheus  
sudo mkdir /prometheus  
sudo chown -R prometheus.prometheus /etc/prometheus/ /usr/local/sbin/prometheus /prometheus
```

그리고 prometheus를 systemd service로 등록하기 위해 서비스 파일을 추가한다.

```
sudo vi /etc/systemd/system/prometheus.service
```

```
=====
```

[Unit]

Description=Prometheus Server

Documentation=<https://prometheus.io/docs/introduction/overview/>

After=network-online.target

[Service]

User=prometheus

WorkingDirectory=/prometheus

ExecStart=/usr/local/sbin/prometheus ₩

--config.file=/etc/prometheus/prometheus.yml ₩

[Install]

WantedBy=multi-user.target

```
=====
```

ETCD Cluster Initialize

이후 서비스 등록 및 기동을 위해 다음과 같은 명령을 수행한다.

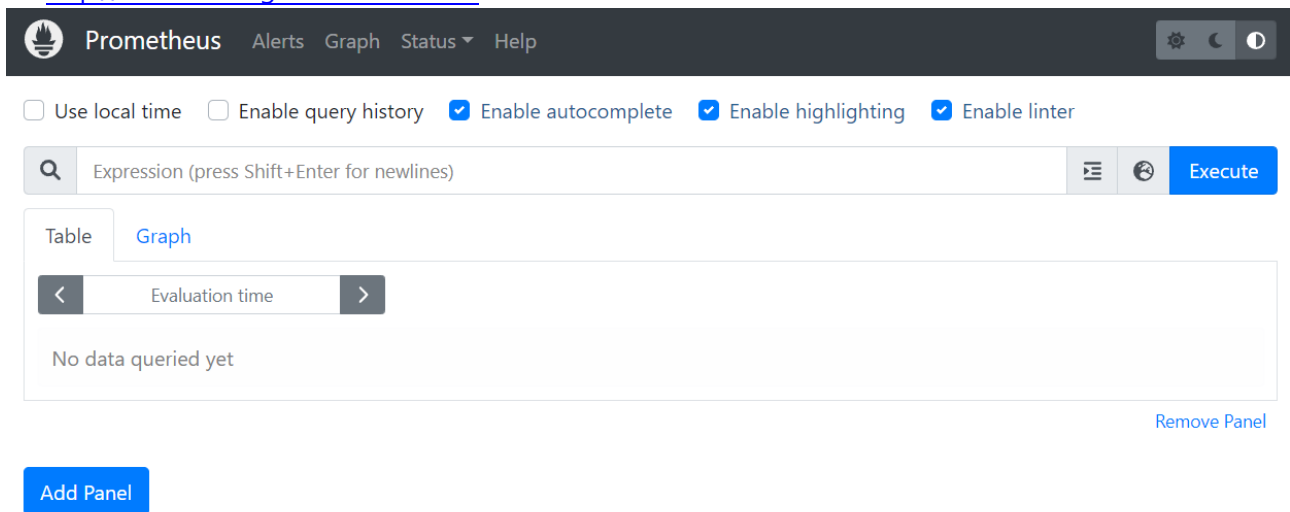
```
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl restart prometheus
sudo systemctl status prometheus
```

```
cpbluebear@monitoring:~$ sudo systemctl status prometheus.service
● prometheus.service - Prometheus Server
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-10-07 10:43:29 KST; 1s ago
     Docs: https://prometheus.io/docs/introduction/overview/
    Main PID: 18181 (prometheus)
      Tasks: 9 (limit: 4577)
     Memory: 15.9M
    CGroup: /system.slice/prometheus.service
            └─18181 /usr/local/sbin/prometheus --config.file=/etc/prometheus/prometheus.yml

10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.949Z caller=head.go:601 level=info component=tsdb msg="Replaying WAL, this may take a while"
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.950Z caller=head.go:672 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=1
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.951Z caller=head.go:672 level=info component=tsdb msg="WAL segment loaded" segment=1 maxSegment=2
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.951Z caller=head.go:709 level=info component=tsdb msg="WAL replay completed" checkpoints=1
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.953Z caller=main.go:1002 level=info fs_type=EXT4_SUPER_MAGIC
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.953Z caller=main.go:1005 level=info msg="TSDB started"
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.953Z caller=main.go:1185 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.954Z caller=main.go:1222 level=info msg="Completed loading of configuration file" filename=/etc/prometheus/prometheus.yml
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.954Z caller=main.go:966 level=info msg="Server is ready to receive web requests."
10:07 07 10:43:29 monitoring prometheus[18181]: ts=2022-10-07T01:43:29.954Z caller=manager.go:941 level=info component="rule manager" msg="Starting rule manager"
```

이 과정을 거치면 prometheus 서비스가 활성화 된다. 다음 주소를 통해 확인할 수 있다.

<http://<Monitoring-Server-IP>:9090>



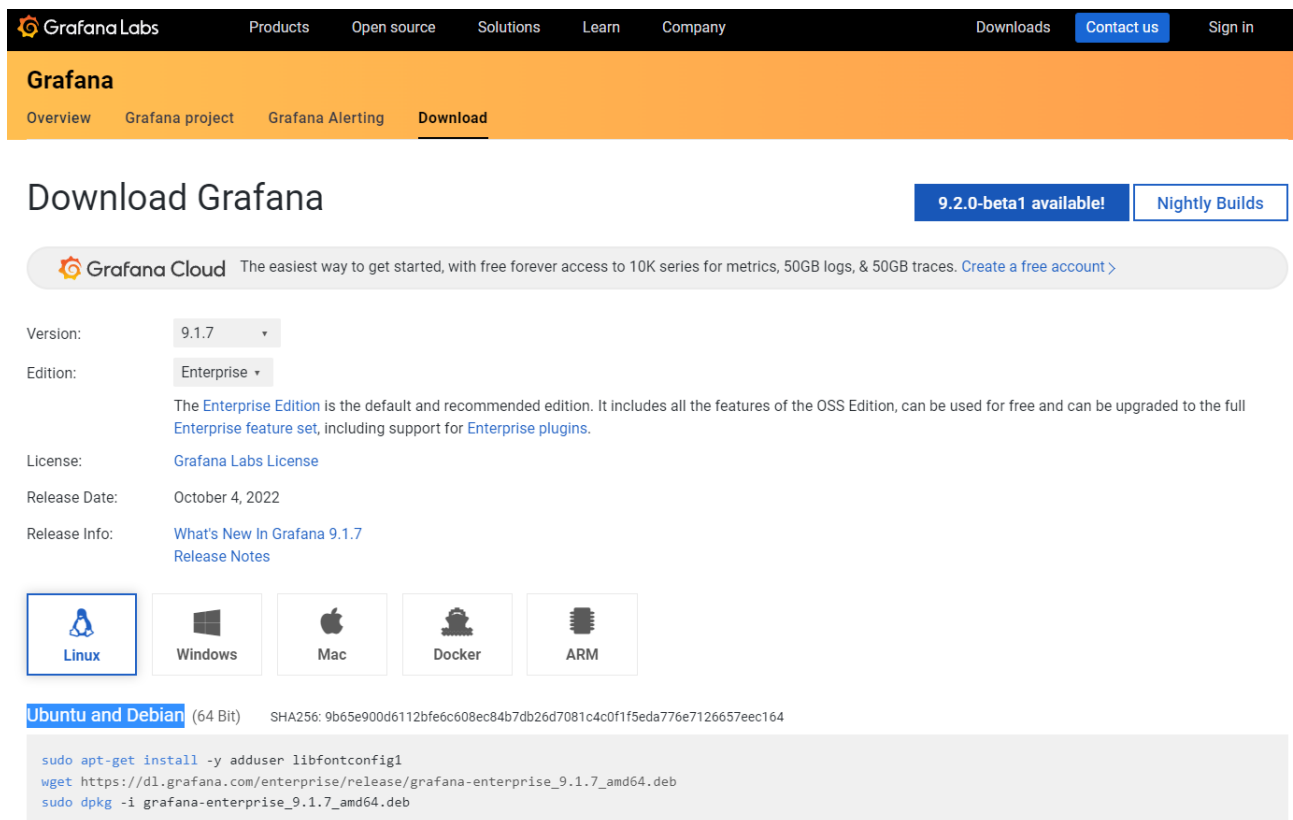
ETCD Cluster Initialize

3. 모니터링 서버에 Grafana 설치

Grafana는 공식 홈페이지에서 기본적으로 리눅스 패키지 관리도구를 통한 설치 파일을 제공한다. 본 워크북은 deb 파일을 이용하여 Grafana 9.17 버전을 설치한다.

<https://grafana.com/grafana/download>

위 사이트에서 9.17 / Enterprise / Linux를 선택한 뒤 아래의 Ubuntu and Debian 항목의 명령을 그대로 본인의 모니터링 서버에 수행한다.



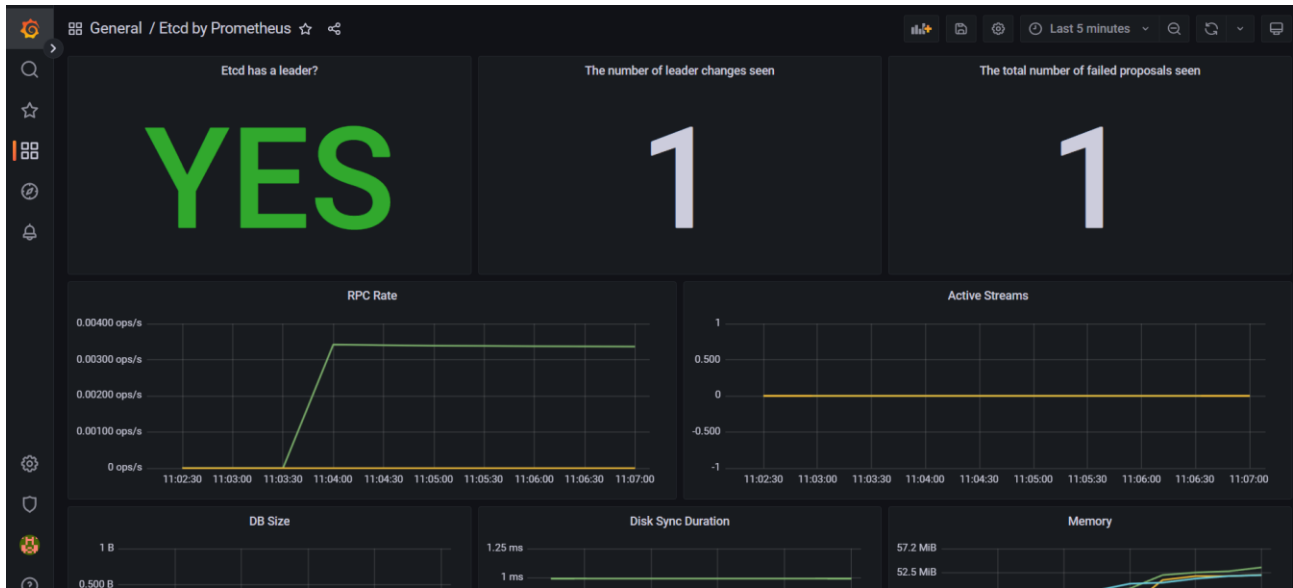
```
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/enterprise/release/grafana-enterprise_9.1.7_amd64.deb
sudo dpkg -i grafana-enterprise_9.1.7_amd64.deb
```

그렇다면 자동으로 Grafana가 설치된다. 다음 주소에 접근하여 Grafana 서비스에 접근할 수 있다. 초기 계정은 admin/admin이다.

<http://<Monitoring-Server-IP>:3000>

예시로 사용할만한 Dashboard ID는 3070이다.

ETCD Cluster Initialize



ETCD Cluster Initialize

[ETCD Cluster 백업과 복구]

etcd는 Kubernetes Cluster를 구성하는 가장 중요한 요소이다. Kubernetes Cluster의 모든 데이터는 etcd에 저장되며, 이 데이터만 살아있다면, Kubernetes Cluster에 장애가 발생하더라도 복구 가능하다. 그렇기 때문에 etcd 데이터베이스에 대한 백업이 매우 중요하며, 이러한 기능 자체를 etcdctl에서 지원한다.

etcdctl을 이용한 etcd cluster의 백업은 다음과 같은 명령으로 수행 가능하다. 이렇게 백업을 수행한다면, ETCD Cluster의 snapshot을 생성한다.

```
sudo etcdctl --endpoints=etcd-01:2379,etcd-02:2379,... snapshot save <백업경로 및 파일명>
```

```
cptbluebear@monitoring:/BACKUP$ sudo etcdctl --endpoints=etcd-01:2379,etcd-02:2379,etcd-03:2379 snapshot save /BACKUP/snapshot-$(date '+%Y%m%d%H%M%S').db
Snapshot saved at /BACKUP/snapshot-20221011194600.db
cptbluebear@monitoring:/BACKUP$ ls -alhF
total 32K
drwxr-xr-x  2 root root 4.0K 10월 11 19:46 ./
drwxr-xr-x 22 root root 4.0K 10월 11 19:43 ../
-rw-r--r--  1 root root 21K 10월 11 19:46 snapshot-20221011194600.db
cptbluebear@monitoring:/BACKUP$
```

이렇게 생성한 snapshot 백업 파일을 통해 불의의 사고가 일어났을 경우 재해 복구가 가능하다. 이러한 기능 또한 etcdctl에서 제공하고 있다. 다만, snapshot save와는 다르게 조금 더 신경써야 할 부분이 많다.

재해 복구를 수행한다는 가정이기에, 이 과정 동안 새로운 데이터의 추가/삭제를 배제한다. 만약 실제로 재해 복구를 수행한다면 우선 ETCD Cluster의 모든 노드에서 etcd service를 중단한 후 수행하는 것이 바람직 할 것이다. 본 워크북에서도 재해 복구 동안에는 etcd service를 전부 중단한 뒤 수행한다고 가정한다.

etcdctl의 snapshot restore는 스냅샷으로 저장된 데이터를 이용해 etcd 데이터 디렉토리를 만든다. 이렇게 생성된 디렉토리를 적절한 경로에 배치한 뒤, etcd가 실행될 때 이 디렉토리를 data_dir로 설정하면 snapshot을 기반으로 복구가 가능하다. 물론 기존에 데이터를 덮어쓰고 복구 또한 가능하다.

본 워크북은 별도의 디렉토리에 snapshot 데이터를 복구한 뒤 etcd 실행 인자를 변경하는 방식으로 진행한다.

다음과 같은 명령을 통해 스냅샷 복원이 가능하다. ETCD Cluster를 구성하는 노드의 수는 3개라 가정한다.

ETCD Cluster Initialize

=====

각 노드에서 `sudo systemctl stop etcd.service` 명령으로 etcd service를 중지한다. 그 뒤 각 노드에서 다음과 같이 명령을 수행한다.

1번 노드에서

```
etcdctl snapshot restore [Snapshot File Name] --name etcd-01 --initial-cluster
etcd-01=https://etcd-01:2380,etcd-02=https://etcd-02:2380,etcd-03=https://etcd-03:2380
--initial-cluster-token <Token Value> --initial-advertise-peer-urls https://etcd-01:2380 -- data-dir
/var/lib/etcd/01.etcd/
```

2번 노드에서

```
etcdctl snapshot restore [Snapshot File Name] --name etcd-02 --initial-cluster
etcd-01=https://etcd-01:2380,etcd-02=https://etcd-02:2380,etcd-03=https://etcd-03:2380
--initial-cluster-token <Token Value> --initial-advertise-peer-urls https://etcd-02:2380 -- data-dir
/var/lib/etcd/02.etcd/
```

3번 노드에서

```
etcdctl snapshot restore [Snapshot File Name] --name etcd-03 --initial-cluster
etcd-01=https://etcd-01:2380,etcd-02=https://etcd-02:2380,etcd-03=https://etcd-03:2380
--initial-cluster-token <Token Value> --initial-advertise-peer-urls https://etcd-03:2380 --data-dir
/var/lib/etcd-03.etcd/
```

각각 명령을 수행한 후 각 노드에 존재하는 `/etc/etcd/etcd.env` 파일에서 `ETCD_DATA_DIR`의 값을 위 명령에서 `--data-dir` 옵션으로 준 경로로 변경한다. 그 후, 각 노드에서 `sudo systemctl restart etcd.service` 명령을 통해 다시 서비스를 활성화 한다.

* 주의! etcd노드를 동시에 활성화 하지 말고 천천히 하나씩 활성화 한다. 예를 들어 1번 노드에서 서비스를 활성화 시킨 뒤 반응이 없을 수 있는데 당황하지 말고 2번 노드의 서비스를 활성화 하면 두 etcd 노드가 연동되면서 정상적으로 서비스가 실행된다. 이후 3번 노드까지 활성화 시키면 된다.

=====

`etcdctl snapshot` 명령에서 각 플래그 옵션의 의미를 설명하자면 다음과 같다.

1. `--name` : etcd 노드에 대한 이름을 지정
2. `--initial-cluster` : 클러스터를 구성하는 노드를 기입한다.
3. `--initial-cluster-token` : 클러스터간 노드가 같은 클러스터라고 판단하기 위한 토큰 값이다. 해당값은 최초 etcd 노드의 환경변수 파일인 `/etc/etcd/etcd.env` 파일에서도 확인할 수 있다.
4. `--initial-advertise-peer-urls` : 복구 과정을 수행하는 초기 클러스터를 지정한다. 자기 자신을 지정한다.
5. `--data-dir` : snapshot으로부터 추출된 etcd 데이터가 생성될 경로를 지정한다.

ETCD Cluster Initialize

이 외의 재해복구 방법은 다음과 같다.

1. 하나의 노드에서 etcd 복구를 진행한 뒤, 다른 노드의 etcd를 초기화 하고 etcdadm으로 다시 join
2. etcd 노드를 초기화 한 뒤, etcdadm init 시 사용할 수 있는 플래그 옵션인 --snapshot 옵션을 사용
3. snapshot이 아닌 etcd 데이터 디렉터리 자체를 사용하는 방법. 다만 이는 checksum을 제공하지 않음
4. etcd 공식 문서에서 추천하고 있는 etcd-backup-restore 프로그램 사용
(<https://github.com/gardener/etcd-backup-restore>)

이렇듯 여러가지 재해복구 방법이 존재하니 사용하여 복구하는 방법도 존재하니 상황에 맞추어 적절한 재해 복구 방법을 선택해야 한다.