01. Kubernetes The Hard Way

SharingTechLab-DevOps/MSA

Exported on Jun 11, 2020

Table of Contents

1 100. install hardway-centos 4

1.1 100.kubernetes 배우기 5

1.2 101. base env 10

1.3 102. vm configuration 14

1.4 103. CA 프로비저닝 및 TLS 인증서 생성 19

1.4.1 kube-api서버와 통신을 위한 PKI 인프라 프로비저닝 19

1.5 104. kubeconfig 구성파일 배포 33

1.6 105.5분 읽을 거리 38

1.7 106. Data 암호화 42

1.8 107. etcd 클러스터 구성 43

1.9 108-0. Kubernetes 컨트롤 플레인 설치 47

1.9.1 Kubernetes Controller 바이너리 다운로드 및 설치 48

1.9.2 Kubernetes API 서버 구성 49

1.9.3 Kubernetes Controller Manager 구성 50

1.9.4 Kubernetes 스케줄러 구성 51

1.9.5 컨트롤러 서비스 시작 52

1.10 108-1. Kubelet 인증을위한 RBAC 52

1.11 109. 쿠버네티스 worker node 구성 55

1.11.1 CNI 네트워킹 구성 56

1.11.2 컨테이너 구성 57

1.11.3 Kubelet 구성 58

1.11.4 kubelet-config.yaml구성 파일을 작성 59

1.11.5 Kubernetes 프록시 구성 60

1.11.6 서비스 시작 61

1.12 1009. 5분 휴식 61

1.13 1010. 쿠버네티스 worker node-2 구성 62

1.14 1011.원격 액세스를위한 kubectl 구성 62

1.15 1012.네트워크 프로비저닝 65

1.15.1 Provisioning Pod Network Routes 66

1.16 1014. DNS Cluster Add-on 배포 68

1.17 1015.deployments 70

1.18 1016. nginx 배포 72

* [100. install hardway-centos](#scroll-bookmark-2)

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

# 100. install hardway-centos

|  |
| --- |
| 이 튜토리얼은 Kubernetes를 어렵게 설정하는 과정을 안내합니다. 이 가이드는 Kubernetes 클러스터를 시작하기 위해 자동화 된 명령을 찾는 사람들을 위한 것이 아닙니다.  이 문서는 모든 과정을 손수 체크해 가면서 설치하는 과정으로, Kubernetes The Hard Way는 학습에 최적화되어 있으므로 Kubernetes 클러스터를 부트 스트랩하는 데 필요한 각 작업을 이해하기 위해 사용해야합니다.  실제 자동화된 설치과정은 <https://kubernetes.io/ko/docs/setup/>  에 잘 나와있으며, kubeadm을 기본으로 ansible을 이용한 kubespray까지 많은 실제 구축 가이드를 찾아보실수 있습니다.  여기서 우리는 쿠버네티스의 구조를 이해하며, 장애시 파악할수 있는 문제들에 대해서 고민해 보고자 합니다. |

|  |
| --- |
| 원본 : <https://github.com/kelseyhightower/kubernetes-the-hard-way>         환경 : ubuntu based on GCP   * [kubernetes](https://github.com/kubernetes/kubernetes) 1.15.3 * [containerd](https://github.com/containerd/containerd) 1.2.9 * [coredns](https://github.com/coredns/coredns) v1.6.3 * [cni](https://github.com/containernetworking/cni) v0.7.1 * [etcd](https://github.com/coreos/etcd) v3.4.0   수정본 : <https://github.com/mmumshad/kubernetes-the-hard-way>  환경 : ubuntu based on VirtualBox   * [Kubernetes](https://github.com/kubernetes/kubernetes) 1.13.0 * [Docker Container Runtime](https://github.com/containerd/containerd) 18.06 * [CNI Container Networking](https://github.com/containernetworking/cni) 0.7.5 * [Weave Networking](https://www.weave.works/docs/net/latest/kubernetes/kube-addon/) * [etcd](https://github.com/coreos/etcd) v3.3.9 * [CoreDNS](https://github.com/coredns/coredns) v1.2.2   우리 환경 : <https://github.com/osci-khoj/kubernetes-the-hard-way-centos> (virtualbox)  환경 : centos based on VirtualBox   * [kubernetes](https://github.com/kubernetes/kubernetes) 1.15.3 * [containerd](https://github.com/containerd/containerd) 1.2.9 * [coredns](https://github.com/coredns/coredns) v1.6.3 * [cni](https://github.com/containernetworking/cni) v0.7.1 * [etcd](https://github.com/coreos/etcd) v3.4.0   새로운 환경  : <https://github.com/alosadagrande/kubernetes-the-hard-way-libvirt-kvm> (kvm)  환경 : centos based on libvirtd   * [Kubernetes](https://github.com/kubernetes/kubernetes) 1.16.2 * [containerd Container Runtime](https://github.com/containerd/containerd) 1.3.0 * [coredns](https://github.com/coredns/coredns) v1.6.2 * [cni](https://github.com/containernetworking/cni) v0.7.3 * [etcd](https://github.com/coreos/etcd) v3.4.0 |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 100.kubernetes 배우기

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **kubernetes 기본구성** | | |  |  | | --- | --- | | 마스터 노드는  Kubernetes 클러스터의 상태를 관리하는 컨트롤 플레인을위한 실행 환경을 제공하며, 클러스터 내부의 모든 작업 뒤에 두뇌   * API Server (관제탑)   + 모든 관리 작업은 마스터 노드에서 실행되는 중앙 제어 플레인 구성 요소 인 kube-apiserver에 의해 조정 * Scheduler (할당)   + kube-scheduler 의 역할은 Pod,서비스등 각 리소스들을 적절한 노드에 할당하는 역할 * Controller Managers (상태점검 및 조절)   + 제어기 매니저 는 Kubernetes 클러스터의 상태를 조절하는 제어기를 실행하는 마스터 노드에 대한 control plain 요소   + 각각의 컨트롤러를 생성하고 각 노드에 배포하며 이를 관리 * etcd (모든 정보)   + etcd 는 Kubernetes 클러스터의 상태를 유지하는 데 사용되는 분산 키 - 값 데이터 저장소  |  | | --- | | 전체 복제: 클러스터의 모든 노드에서 전체 저장소를 사용할 수 있음 고가용성: Etcd는 하드웨어 또는 네트워크 문제 발생 시 단일 장애 지점을 방지하도록 설계됨 일관성: 모든 읽기가 여러 호스트에서 가장 최근의 쓰기를 반환함 단순성: 잘 정의된 사용자 대면 API(gRPC) 포함 보안: 선택적 클라이언트 인증서 인증을 사용하여 자동 TLS 구현 빠른 속도: 초당 10,000개의 쓰기 속도로 벤치마크 신뢰할 수 있는: 래프트 알고리즘을 사용하여 스토어가 적절하게 배포됨 |   작업자 노드는 클라이언트 응용 프로그램을위한 실행 환경을 제공   * Container runtime (엔진)   + [Docker](https://www.docker.com/) - containerd 를 컨테이너 런타임으로 사용하는 컨테이너 플랫폼이지만 Kubernetes에서 가장 널리 사용되는 컨테이너 런타임입니다.   + [CRI-O](https://cri-o.io/) - Kubernetes의 경량 컨테이너 런타임으로 Docker 이미지 레지스트리도 지원합니다.   + [containerd](https://containerd.io/) - 견고성을 제공하는 간단하고 휴대용 컨테이너 런타임 * kubelet (선장)   + 각 노드에서 실행되는 에이전트와 마스터 노드로부터의 컨트롤 플레인 구성 요소와 통신 * kube-proxy   + 노드의 모든 네트워킹 규칙의 동적 업데이트 및 유지 보수에 대한 책임을 각 노드에서 실행되는 네트워크 에이전트 * Addons for DNS, Dashboard, cluster-level monitoring and logging   + DNS - 클러스터 DNS는 Kubernetes 객체 및 자원에 DNS 레코드를 할당하는 데 필요한 DNS 서버입니다.   + Dashboard - 클러스터 관리를위한 일반적인 목적의 웹 기반 사용자 인터페이스   + 모니터링 - 클러스터 수준의 컨테이너 메트릭을 수집하여 중앙 데이터 저장소에 저장합니다.   + 로깅 - 클러스터 수준 컨테이너 로그를 수집하여 분석을 위해 중앙 로그 저장소에 저장합니다. |   **이중화 작업  (Set up High-Availability Kubernetes Masters)**   |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | Figure 1 ha-master-gce | * **etcd 인스턴스 : 모든 인스턴스는  함께 클러스터됩니다.** * API 서버 앞에는 외부 및 내부 트래픽을 라우팅하는로드 밸런서가 있습니다. * API 서버 : 각 서버는 로컬 etcd와 통신합니다. 클러스터의 모든 API 서버를 사용할 수 있습니다. * 컨트롤러, 스케줄러 및 클러스터 자동 스케일러 :  각 인스턴스 중 하나만 클러스터에서 활성화됩니다. * 애드온 관리자 : 각 관리자는 애드온을 동기화 상태로 유지하려고 독립적으로 작업합니다. | | |  |  | | --- | | **우리의 최종 구성도** | | |  |  | | --- | --- | |  |  |   실제 프로젝트에서 구성 시  image2019-8-5_23-16-27.png  Figure 2 image2019-8-5\_23-16-27.png  _scroll_external/attachments/image2019-8-6_2-11-53-f41f4242405300c49c1969cc6a28bbe7416dc2e3effe3b35944f9522babba0e1.png | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 101. base env

|  |
| --- |
| **목표구성도** |
|  |

|  |
| --- |
| **기본 구성사항** |
| **VM Hardware Requirements**  8 GB of RAM (Preferebly 16 GB) 50 GB Disk space   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **VM Name** | **Purpose** | **OS** | **vCPUs** | **Memory** | **Disk** | | master00 | controller | CentOS 7 | 2 | 2 GB | 20 GB | | master01 | controller | CentOS 7 | 2 | 2 GB | 20 GB | | master02 | controller | CentOS 7 | 2 | 2 GB | 20 GB | | worker00 | controller | CentOS 7 | 1 | 1 GB | 20 GB | | worker01 | controller | CentOS 7 | 1 | 1 GB | 20 GB | | worker02 | controller | CentOS 7 | 1 | 1 GB | 20 GB | | loadbalancer | balancer | CentOS 7 | 2 | 2 GB | 20GB |   **libvirt 구성**   |  | | --- | | yum groupinstall "Virtualization Host"  yum install libvirt-client  systemctl enable libvirtd --now |   **LibvirtCLI**  인프라를 실행하는 데 필요한 모든 가상 디바이스를 배포하기 위해 virsh명령줄을 사용할 수 있습니다. virsh프로그램은 virsh게스트 도메인을 관리하기 위한 기본 인터페이스입니다. 이 프로그램을 사용하여 도메인을 생성, 일시 중지 및 종료할 수 있습니다. 또한 현재 도메인을 나열하는 데도 사용할 수 있습니다.  그러나 가상 인프라를 구축하기 위해 kc[li](https://kcli.readthedocs.io/en/master/)를 사용하는 것이 훨씬 더 쉽습니다. **Kcli**는 기존 가상화 공급자(libvirt, Kubevirt, Ovirt, Openstack, GCp및 Lell, vsphere)와 상호 작용하고 클라우드 이미지에서 vms를 쉽게 배포하고 사용자 지정하기 위한 도구입니다.  이러한 vms(list, info, ssh, start, stop, delete, Console, serialconsole, add/dedeletedisk, add/dedeletenic등)와 상호 작용할 수도 있습니다. 또한 미리 정의된 프로파일을 사용하여 VM을 한번에 배포할 수 있으며, 여러 VM은 계획 파일을 사용하거나 계획이 이미 생성된 전체 제품을 사용합니다.  **2. kcli 설치**   * container install  |  | | --- | | # docker pull karmab/kcli    alias kcli='docker run --net host -it --rm --security-opt label=disable -v $HOME/.ssh:/root/.ssh -v $HOME/.kcli:/root/.kcli -v /var/lib/libvirt/images:/var/lib/libvirt/images -v /var/run/libvirt:/var/run/libvirt -v $PWD:/workdir -v /var/tmp:/ignitiondir karmab/kcli'    alias kclishell='docker run --net host -it --rm --security-opt label=disable -v $HOME/.ssh:/root/.ssh -v $HOME/.kcli:/root/.kcli -v /var/lib/libvirt/images:/var/lib/libvirt/images -v /var/run/libvirt:/var/run/libvirt -v $PWD:/workdir -v /var/tmp:/ignitiondir --entrypoint=/bin/sh karmab/kcli'    alias kweb='docker run -p 9000:9000 --net host -it --rm --security-opt label=disable -v $HOME/.ssh:/root/.ssh -v $HOME/.kcli:/root/.kcli -v /var/lib/libvirt/images:/var/lib/libvirt/images -v /var/run/libvirt:/var/run/libvirt -v $PWD:/workdir -v /var/tmp:/ignitiondir --entrypoint=/usr/bin/kweb karmab/kcli' |  * 로컬 libvirt를 관리하도록 kcli를 구성  |  | | --- | | # ssh-keygen -t rsa -b 2048  # kcli create host kvm -H 127.0.0.1 |  * 풀 만들기  |  | | --- | | # kcli create pool -p /data/kcli\_volume default  Creating pool default...  # sudo setfacl -m u:$(id -un):rwx /data/kcli\_volume        # kcli info host  Connection: qemu:///system  Host: osc-ceph02-openinfraday  Cpus: 24  Vms Running: 8  Memory Used: 88192MB of 163829MB    Storage:images Type: dir Path:/home/images Used space: 194.18GB Available space: 68.3GB  Storage:candidate Type: dir Path:/home/candidate Used space: 194.18GB Available space: 68.3GB  Storage:Downloads Type: dir Path:/root/Downloads Used space: 194.18GB Available space: 68.3GB  Storage:default Type: dir Path:/data/kcli\_volume Used space: 291.85GB Available space: 266.75GB  Storage:data Type: dir Path:/data/kvm-vol-pool/vm-candidate-pool Used space: 291.56GB Available space: 267.04GB  Storage:root Type: dir Path:/root Used space: 194.18GB Available space: 68.3GB  Storage:vms Type: dir Path:/data/vms Used space: 291.53GB Available space: 267.08GB  Network: br0 Type: bridged |  |  |  | | --- | --- | | **_scroll_external/attachments/image2020-6-9_23-34-11-f225c5df874209f4c99649f00ae7fb119dfd5894c6aefe463bc7eba06bf36e1b.png** | **kcli download image centos7**  kcli create vm -i centos7 vm1  kcli create vm -i centos7 -P memory=2048 -P numcpus=2 vm1  kcli create vm -i centos7 -P disks=[10,20] -P nets=[default,default] -P cmds=[yum -y install nc] vm1  kcli get keyword  remove vm  kcli list vms | grep up | awk '{ print "kcli delete -y vm " $2}' | sh -v |   kcli command   * + List vms     - kcli list vm   + List cloud images     - kcli list images   + Create vm from a profile named base7     - kcli create vm -p base7 myvm   + Create vm from profile base7 on a specific client/host named twix     - kcli -C twix create vm -p base7 myvm   + Delete vm     - kcli delete vm vm1   + Get detailed info on a specific vm     - kcli info vm vm1   + Start vm     - kcli start vm vm1   + Stop vm     - kcli stop vm vm1   + Switch active client/host to bumblefoot     - kcli switch host bumblefoot   + Get remote-viewer console     - kcli console vm vm1   + Get serial console (over TCP). It will only work with vms created with kcli and will require netcat client to be installed on hypervisor     - kcli console vm -s vm1   + Deploy multiple vms using plan x defined in x.yml file     - kcli create plan -f x.yml x   + Delete all vm from plan x     - kcli delete plan x   + Add 5GB disk to vm1, using pool named images     - kcli create vm-disk -s 5 -p images vm1   + Delete disk named vm1\_2.img from vm1     - kcli create disk -d -n vm1\_2.img  vm1   + Update to 2GB memory vm1     - kcli update vm -m 2048 vm1   + Clone vm1 to new vm2     - kcli clone vm -b vm1 vm2   + Connect by ssh to the vm     - kcli ssh vm vm1   + Add a new network     - kcli create network -c 192.168.7.0/24 --dhcp mynet   + Add a new pool     - kcli create pool -t dir -p /hom/images images   + Add a new nic from network default     - kcli create nic -n default myvm   + Delete nic eth2 from vm     - kcli delete nic -i eth2 myvm   + Create snapshot snap of vm:     - kcli snapshot vm -n vm1 snap1   + Get info on your kvm setup     - kcli info host   + Export vm:     - kcli export vm vm1  |  | | --- | |  | |

## 102. vm configuration

|  |
| --- |
| **vm 만들기** |
| make the vm   * network  |  | | --- | | [root@ct kcli\_volume]# kcli list network  Listing Networks...  +---------+--------+------------------+------+---------------+------+  | Network | Type | Cidr | Dhcp | Domain | Mode |  +---------+--------+------------------+------+---------------+------+  | default | routed | 192.168.122.0/24 | True | default | nat |  +---------+--------+------------------+------+---------------+------+ |   모든 Kubernetes 클러스터 리소스를 배치 할 새 가상 네트워크를 만들려고합니다.   * + 네트워크 주소 범위는 192.168.111.0/24입니다.   + 네트워크 이름은 k8s-net입니다   + 이 네트워크의 도메인은 k8s-thw.local입니다. 이 가상 네트워크의 모든 VM은이 도메인 이름을 정규화 된 이름의 일부로 가져 오기 때문에 중요합니다.  |  | | --- | | [root@ct kcli\_volume]# kcli list network  Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.  Listing Networks...  +---------+--------+------------------+------+---------------+------+  | Network | Type | Cidr | Dhcp | Domain | Mode |  +---------+--------+------------------+------+---------------+------+  | default | routed | 192.168.122.0/24 | True | default | nat |  | k8s-net | routed | 192.168.111.0/24 | True | k8s-thw.local | nat |  +---------+--------+------------------+------+---------------+------+ |  * **이미지**   인스턴스를 만들려면 이미지가 제공되어야합니다. 이 가이드에서는 CentOS 7을 모든 VM의 기본 운영 체제로 사용합니다. kcli를 사용하면 매우 쉽습니다. kcli 명령 줄을 사용하여 클라우드 CentOS 7 이미지를 다운로드하십시오.   |  | | --- | | [root@ct kcli\_volume]# kcli download image centos7 --pool default  Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.  Using url https://cloud.centos.org/centos/7/images/CentOS-7-x86\_64-GenericCloud.qcow2...  Grabbing image centos7...  Image CentOS-7-x86\_64-GenericCloud.qcow2 already there.Leaving...  Image centos7 Added  Updating profile 127.0.0.1\_centos7 with image CentOS-7-x86\_64-GenericCloud.qcow2 |   **DNS**  모든 VM의 직접 및 역방향 쿼리를 해결해야하는 적절한 DNS 구성이 필요합니다. 다른 유사한 자습서와 달리 kcli를 사용하면 각 VM의 적절한 DNS 확인을 매우 쉽게 구성 할 수 있습니다. 새 인스턴스를 생성 할 때마다 **베어 메탈** 호스트에서 실행 되는 **libvirt dnsmasq에** DNS 레코드를 생성 할 수 있습니다 .  **SSH 액세스 구성**  SSH는로드 밸런서, 컨트롤러 및 작업자 인스턴스를 구성하는 데 사용됩니다. kcli를 활용하면 모든 인스턴스간에 ssh 키를 수동으로 교환 할 필요가 없습니다. Kcli는 생성시 베어 메탈 서버에서 모든 인스턴스에 퍼블릭 ssh 키를 자동으로 주입 (cloudinit 사용)합니다. 따라서 인스턴스가 시작되고 실행되면 쉽게 실행할 수 있습니다.kcli ssh vm\_name  Kubernetes **컨트롤 플레인을** 호스팅 할 세 개의 컴퓨팅 인스턴스를 만듭니다 . 기본적으로 다음과 같이 구성된 3 개의 새 인스턴스를 만듭니다.   * OS로서의 CentOS 이미지 * 50GB 디스크 * k8s-net에 연결됨 (192.168.111.0/24) * 16GB 메모리 및 4 vCPus * libvirt의 dnsmasq에 포함될 DNS 레코드 (이 경우 $ {node} .k8s-thw.local)를 작성하십시오 (reservedns = true). * 다른 VM에서 사용할 수 없도록 IP를 예약하십시오 (reserveip = true). * 베어 메탈 서버의 / etc / host에 레코드를 작성하여 가상 네트워크 도메인 외부에서도 도달 할 수 있도록하십시오. (예약 = 예) * 서버가 시작되고 실행되면 "yum update -y"를 실행하십시오. 이 명령은 cloudinit에 삽입되므로 처음부터 모든 인스턴스가 최신 상태입니다.  |  | | --- | | for node in master00 master01 master02; do  kcli create vm -i centos7 -P disks=[50] -P nets=[k8s-net] -P memory=16384 -P numcpus=4 \  -P cmds=["yum -y update"] -P reservedns=true -P reserveip=true -P reservehost=true ${node}  done |   **로드 밸런서**  적절한 Kubernetes고가용성 환경을 유지하려면 API로드를 분산하기 위해 로드 밸런서가 필요합니다. 이 경우 HAProxyloadbalancer서비스를 실행할 특정 인스턴스를 생성할 것입니다. 먼저 로드 밸런서 서비스를 호스트 할 인스턴스를 생성합니다. 아래에서는 다음을 사용하여 새 인스턴스를 생성하려고 합니다.   |  | | --- | | kcli create vm -i centos7 -P disks=[20] -P nets=[k8s-net] -P memory=2048 -P numcpus=2 \  -P cmds=["yum -y update"] -P reserverdns=true -P reserverip=true -P reserverhost=true loadbalancer |   **로드 밸런서 서비스**  다음 단계에서는 이전에 생성된 인스턴스에서 HAProxy를 사용하여 로드 밸런서 서비스를 설치하는 방법을 보여 줍니다.  먼저 다음 인스턴스에 연결합니다.   |  | | --- | | kcli ssh loadbalancer |   HAProxy설치 및 구성:   |  | | --- | | # export DOMAIN="k8s-thw.local"  # yum install -y haproxy bind-utils  # for node in master00 master01 master02 ; do export IP\_${node}="$(dig +short ${node})"; done      # sudo tee /etc/haproxy/haproxy.cfg << EOF  global  log 127.0.0.1 local2  chroot /var/lib/haproxy  pidfile /var/run/haproxy.pid  maxconn 4000  user haproxy  group haproxy  daemon  stats socket /var/lib/haproxy/stats    defaults  log global  option httplog  option dontlognull  option http-server-close  option redispatch  retries 3  timeout http-request 10s  timeout queue 1m  timeout connect 10s  timeout client 1m  timeout server 1m  timeout http-keep-alive 10s  timeout check 10s  maxconn 3000    listen stats :9000  stats enable  stats realm Haproxy\ Statistics  stats uri /haproxy\_stats  stats auth admin:password  stats refresh 30  mode http    frontend main \*:6443  default\_backend mgmt6443  option tcplog    backend mgmt6443  balance source  mode tcp  # MASTERS 6443  server master00.${DOMAIN} ${IP\_master00}:6443 check  server master01.${DOMAIN} ${IP\_master01}:6443 check  server master02.${DOMAIN} ${IP\_master02}:6443 check  EOF |   Kubernetes포트는 6443이므로, haproxy가 특정 포트에서 수신 대기할 수 있도록 selinux정책을 수정해야 합니다.   |  | | --- | | sudo semanage port --add --type http\_port\_t --proto tcp 6443 |   모든 항목이 올바르게 구성되었는지 확인합니다.   |  | | --- | | haproxy -c -V -f /etc/haproxy/haproxy.cfg |   서비스를 시작하고 사용하도록 설정합니다.   |  | | --- | | sudo systemctl enable haproxy --now |   worker 구성   |  | | --- | | kcli create vm -i centos7 -P disks=[50] -P nets=[k8s-net] -P memory=16384 -P numcpus=4 \  -P cmds=["yum -y update",'echo "10.200.0.0/24" > /home/centos/pod\_cidr.txt'] -P reservedns=true -P reserveip=true -P reservehost=true worker00    kcli create vm -i centos7 -P disks=[50] -P nets=[k8s-net] -P memory=16384 -P numcpus=4 \  -P cmds=["yum -y update",'echo "10.200.0.0/24" > /home/centos/pod\_cidr.txt'] -P reservedns=true -P reserveip=true -P reservehost=true worker01  kcli create vm -i centos7 -P disks=[50] -P nets=[k8s-net] -P memory=16384 -P numcpus=4 \  -P cmds=["yum -y update",'echo "10.200.0.0/24" > /home/centos/pod\_cidr.txt'] -P reservedns=true -P reserveip=true -P reservehost=true worker02 |   check   |  | | --- | | [root@osc-ceph02-openinfraday ~]# kcli ssh master01 ping loadbalancer  Warning: Permanently added '192.168.111.175' (ECDSA) to the list of known hosts.  PING loadbalancer.k8s-thw.local (192.168.111.202) 56(84) bytes of data.  64 bytes from loadbalancer.k8s-thw.local (192.168.111.202): icmp\_seq=1 ttl=64 time=0.483 ms  64 bytes from loadbalancer.k8s-thw.local (192.168.111.202): icmp\_seq=2 ttl=64 time=0.444 ms    [root@osc-ceph02-openinfraday ~]# kcli list vms | grep up  | loadbalancer | up | 192.168.111.202 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | master00 | up | 192.168.111.154 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | master01 | up | 192.168.111.175 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | master02 | up | 192.168.111.232 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | worker00 | up | 192.168.111.82 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | worker01 | up | 192.168.111.53 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |  | worker02 | up | 192.168.111.91 | CentOS-7-x86\_64-GenericCloud.qcow2 | kvirt | centos7 |    전체 reboot    for node in loadbalancer master00 master01 master02 worker00 worker01 worker02  do  kcli restart vm $node  done | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 103. CA 프로비저닝 및 TLS 인증서 생성

[PKI 인프라](https://en.wikipedia.org/wiki/Public_key_infrastructure) 를 프로비저닝 한 다음이를 사용하여 인증 기관을 부트 스트랩하고 다음 구성 요소에 대한 TLS 인증서를 생성

대상 : etcd, kube-apiserver, kube-controller-manager, kube-scheduler, kubelet 및 kube-proxy.

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-04-certificate-authority-d102717c6fc4670fecc235e168bb6cf46d27e9555a263946cb11c6d62b36ca4c.png |

### kube-api서버와 통신을 위한 PKI 인프라 프로비저닝

[PKI 인프라](https://en.wikipedia.org/wiki/Public_key_infrastructure) 를 프로비저닝 한 다음이를 사용하여 인증 기관을 부트 스트랩하고 etcd, kube-apiserver, kube-controller-manager, kube-scheduler, kubelet 및 kube-proxy에 대한 TLS 인증서를 생성합니다

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | * 먼저 Certificate Authority(인증기관)을 만듬   + 추가적인 TLS 인증서를 생성해줌   + CA에 인증서 서명 요청을 하고, 이를 이용해 private key를 생성   _scroll_external/remote/tls-handshaking-with-certificates-and-keys-7bc741638e5a4c57d59593be2a5f7642e3078adaa69ab05812d3c356addbfb34.png_scroll_external/attachments/screenshot-from-2020-06-10-10-23-23-45b97e975fd49dac3a14597e4ed4f505f5248e9e5920a132f0240c8d44748422.png_scroll_external/attachments/screenshot-from-2020-06-10-10-26-53-8cbb24d88ccb1ccff8439874ad07c093483123df9ed6ac9029fdd1a2bdaa920c.png_scroll_external/attachments/screenshot-from-2020-06-10-10-27-24-46409f6b9c0db85666a9dcd17a8f8defdbdf475cf27ac99f8647bd191deded80.png_scroll_external/attachments/screenshot-from-2020-06-10-11-30-55-fbee81a4935b7b0d2ff274f9880c161b14c964fa787337adf91fa128138a687d.png  출처: <https://mcuoneclipse.com/2017/04/14/enable-secure-communication-with-tls-and-the-mosquitto-broker/> |   openssl vs cfssl   |  |  |  |  | | --- | --- | --- | --- | | **1** | **mkdir** openssl && cd openssl | **1 2 3 4 5 6** | **curl** https://pkg.cfssl.org/R1.2/cfssl\_linux-amd64 \        -o /usr/local/bin/cfssl curl https://pkg.cfssl.org/R1.2/cfssljson\_linux-amd64 \        -o /usr/local/bin/cfssljson curl https://pkg.cfssl.org/R1.2/cfssl-certinfo\_linux-amd64 \        -o /usr/local/bin/cfssl-certinfo | | 1 | openssl genrsa -out ca.key 2048 | 1 2 3 | chmod +x /usr/local/bin/cfssl \          /usr/local/bin/cfssljson \          /usr/local/bin/cfssl-certinfo | | 1 2 3 | openssl req -x509 -new -nodes \      -key ca.key -sha256 \      -days 1825 -out ca.crt | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | cat > ca-csr.json <<EOF {     "CN": "Demo CA",     "key": {       "algo": "rsa",       "size": 2048     },     "names": [       {         "C": "US",         "L": "California",         "ST": "Milpitas"       }     ]   } EOF | | 1 2 3 | openssl req -x509 -new -nodes \       -key ca.key -subj "/CN=scriptcrunch/C=US/L=CALIFORNIA" \       -days 1825 -out ca.crt | 1 | cfssl gencert -initca ca-csr.json | cfssljson -bare ca – | | 1 | openssl genrsa -out server.key 2048 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | cat > ca-config.json <<EOF  {     "signing": {       "default": {         "expiry": "8760h"       },       "profiles": {         "web-servers": {           "usages": [             "signing",             "key encipherment",             "server auth",             "client auth"           ],           "expiry": "8760h"         }       }     }   } EOF | | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 | cat > csr.conf <<EOF [ req ] default\_bits = 2048 prompt = no default\_md = sha256 req\_extensions = req\_ext distinguished\_name = dn   [ dn ] C = US ST = California L = San Fransisco O = Scriptcrunch OU = Scriptcrunch Dev CN = scriptcrunch.com   [ req\_ext ] subjectAltName = @alt\_names   [ alt\_names ] DNS.1 = scriptcrunch DNS.2 = scriptcrunch.com IP.1 = 10.34.12.5 IP.2 = 10.34.12.5   EOF | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | cat > server-csr.json <<EOF  {     "CN": "scriptcrunch",     "hosts": [         "[scriptcrunch.com](http://scriptcrunch.com)",         "[www.scriptcrunch.com](http://www.scriptcrunch.com)"     ],     "key": {         "algo": "rsa",         "size": 2048     },     "names": [         {             "C": "US",             "L": "CA",             "ST": "San Francisco"         }     ] } EOF | | 1 | openssl req -new -key server.key -out server.csr -config csr.conf | 1 2 3 4 5 6 | cfssl gencert \   -ca=ca.pem \   -ca-key=ca-key.pem \   -config=ca-config.json \   -profile=web-servers \   server-csr.json | cfssljson -bare server | | 1 2 3 | openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key \ -CAcreateserial -out server.crt -days 10000 \ -extfile csr.conf |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **What is contained in a CSR?**   |  |  |  | | --- | --- | --- | | **Name** | **Explanation** | **Examples** | | Common Name | The fully qualified domain name (FQDN) of your server. This must match exactly what you type in your web browser or you will receive a [name mismatch error](https://www.sslshopper.com/ssl-certificate-name-mismatch-error.html). | \*.[google.com](http://google.com) [mail.google.com](http://mail.google.com) | | Organization | The legal name of your organization. This should not be abbreviated and should include suffixes such as Inc, Corp, or LLC. | Google Inc. | |  * CFSSL  |  | | --- | | CFSSL 설치  cfssl및 cfssljson명령 행 유틸리티 제공에 사용되는 PKI 인프라 및 TLS 인증서를 생성합니다.    다운로드 및 설치 cfssl및 cfssljson:    wget -q --timestamping \  https://storage.googleapis.com/kubernetes-the-hard-way/cfssl/linux/cfssl \  https://storage.googleapis.com/kubernetes-the-hard-way/cfssl/linux/cfssljson  chmod +x cfssl cfssljson  sudo mv cfssl cfssljson /usr/local/bin/    확인 cfssl및 cfssljson버전 1.3.4 이상이 설치되어 있습니다 :    cfssl version  산출    Version: 1.3.4  Revision: dev  Runtime: go1.13  cfssljson --version  Version: 1.3.4  Revision: dev  Runtime: go1.13 |  * Certificate Authority * 추가 TLS인증서를 생성하는 데 사용할 수 있는 인증 기관을 프로비저닝 합니다.  CA구성 파일, 인증서 및 개인 키 생성:  |  | | --- | | {    cat > ca-config.json <<EOF  {  "signing": {  "default": {  "expiry": "8760h"  },  "profiles": {  "kubernetes": {  "usages": ["signing", "key encipherment", "server auth", "client auth"],  "expiry": "8760h"  }  }  }  }  EOF    cat > ca-csr.json <<EOF  {  "CN": "Kubernetes",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "Kubernetes",  "OU": "CA",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert -initca ca-csr.json | cfssljson -bare ca    }  결과:    ca-key.pem  ca.pem |  * **클라이언트 및 서버 인증서**   이 섹션에서는 각 Kubernetes구성 요소에 대한 클라이언트 및 서버 인증서와 Kubernetes에 대한 클라이언트 인증서를 생성합니다.admin사용자.   * + API 서버와 통신을 하기 위한 클라이언트 서버 인증서 생성  |  | | --- | | {    cat > admin-csr.json <<EOF  {  "CN": "admin",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "system:masters",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -profile=kubernetes \  admin-csr.json | cfssljson -bare admin    }    결과 :  admin-key.pem  admin.pem |   Code Block 1 Admin 인증서   |  | | --- | | admin 사용자는 **system : masters** 그룹의 일부입니다. 그래서, kubectl 유틸리티를 사용하여 Kubernetes 클러스터에서 관리 작업을 수행할수 있습니다.  Kubernetes는 Node Authorizer ( <https://kubernetes.io/docs/reference/access-authn-authz/node/>)라는 특수 목적의 권한 부여 모드를 사용한다.  이 모드는 Kubelets의 API 요청을 구체적으로 승인한다. 노드 인증 자에 의해 권한을 부여하기 위해 Kubelets는 사용자 이름이 system:node :<nodeName> 인 system :nodes 그룹에 있는 것으로 식별하는 자격 증명을 사용해야 한다. 노드 권한 부 여자 요구 사항을 충족하는 각 Kubernetes 작업자 노드에 대한 인증서를 만든다.  각 Kubernetes 작업자 노드에 대한 인증서 및 개인 키를 생성한다. |   **Kubelet클라이언트 인증서**  [Kubelets](https://kubernetes.io/docs/concepts/overview/components/#kubelet)에서 작성한 API요청을 구체적으로 승인하는 NodeAuthorizer라는 특수 [목적의 인증 모드](https://kubernetes.io/docs/admin/authorization/node/)를 사용합니다. 노드 인증자의 승인을 받으려면 Kubelets는 데이터베이스에 있는 것으로 식별되는 자격 증명을 사용해야 합니다.system:nodes사용자 이름을 가진 그룹system:node:<nodeName>이 섹션에서는 노드 인증자 요구 사항을 충족하는 각 Kubernetes작업자 노드에 대한 인증서를 생성합니다.   |  | | --- | | DOMAIN="k8s-thw.local"  for instance in worker00 worker01 worker02; do  cat > ${instance}-csr.json <<EOF  {  "CN": "system:node:${instance}.${DOMAIN}",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "system:nodes",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    NODE\_IP=$(kcli info vm ${instance} | grep "ip:" | awk {'print$2'})    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -hostname=${instance}.${DOMAIN},${instance},${NODE\_IP} \  -profile=kubernetes \  ${instance}-csr.json | cfssljson -bare ${instance}  done    결과:    worker00-key.pem  worker00.pem  worker01-key.pem  worker01.pem  worker02-key.pem  worker02.pem |   **컨트롤러 관리자 클라이언트 인증서**  생성kube-controller-manager클라이언트 인증서 및 개인 키:   |  | | --- | | {    cat > kube-controller-manager-csr.json <<EOF  {  "CN": "system:kube-controller-manager",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "system:kube-controller-manager",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -profile=kubernetes \  kube-controller-manager-csr.json | cfssljson -bare kube-controller-manager    }    결과:    kube-controller-manager-key.pem  kube-controller-manager.pem |   **Kube프록시 클라이언트 인증서**  Kuber-proxy클라이언트 인증서와 개인 키를 생성합니다   |  | | --- | | {    cat > kube-proxy-csr.json <<EOF  {  "CN": "system:kube-proxy",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "system:node-proxier",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -profile=kubernetes \  kube-proxy-csr.json | cfssljson -bare kube-proxy    }    결과:    kube-proxy-key.pem  kube-proxy.pem |  * + **스케줄러 클라이언트 인증서**   생성kube-scheduler클라이언트 인증서 및 개인 키:   |  | | --- | | {    cat > kube-scheduler-csr.json <<EOF  {  "CN": "system:kube-scheduler",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "system:kube-scheduler",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -profile=kubernetes \  kube-scheduler-csr.json | cfssljson -bare kube-scheduler    }    결과:    kube-scheduler-key.pem  kube-scheduler.pem |  * + **KubernetesAPI서버 인증서**   그kubernetes-the-hard-way정적 IP주소는 KubernetesAPIServer인증서의 제목 대체 이름 목록에 포함됩니다. 이렇게 하면 원격 클라이언트에서 인증서의 유효성을 검사할 수 있습니다.   * + kcli가 이상해서 /etc/hosts에 등록을 못하므로 직접 해주어야 함.  |  | | --- | | [root@osc-ceph02-openinfraday renault\_edu]# kcli list vms | grep 111 | awk '{ print $6 " " $2}' >> /etc/hosts |  |  | | --- | | kubernetes-the-hard-way 정적 IP 주소는 Kubernetes API 서버 인증서의 hostname에 포함됩니다. 이렇게 하면 원격 클라이언트가 인증서를 확인할 수 있습니다. kube-apiserver 인증서는 다양하게 접근하는 모든 이름에 대해서 인증해줄 필요가 있습니다.  DNS로 들어오거나, IP로 들어오거나, localhost로 들어오거나 다양한 방법에 가지고 들어올수 있도록 조치해야 합니다. |  |  | | --- | | {    KUBERNETES\_BAREMETAL\_ADDRESS=192.168.192.42  KUBERNETES\_HOSTNAMES=kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,kubernetes.svc.cluster.local  for node in loadbalancer master00 master01 master02 ; do export IP\_${node}=$(getent hosts ${node} | cut -d" " -f1) ;done    #/etc/hosts에 넣어야 함.    cat > kubernetes-csr.json <<EOF  {  "CN": "kubernetes",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "Kubernetes",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -hostname=loadbalancer.${DOMAIN},10.32.0.1,${IP\_loadbalancer},${IP\_master00},${IP\_master01},${IP\_master02},${KUBERNETES\_BAREMETAL\_ADDRESS},127.0.0.1,${KUBERNETES\_HOSTNAMES} \  -profile=kubernetes \  kubernetes-csr.json | cfssljson -bare kubernetes    }    결과:    kubernetes-key.pem  kubernetes.pem |   KubernetesAPI서버는 자동으로 할당됩니다.kubernetes내부 DNS이름. 첫번째 IP주소에 연결됩니다.10.32.0.1()주소 범위에서()10.32.0.0/24)제어[부 부트 스트랩](https://github.com/alosadagrande/kubernetes-the-hard-way-libvirt-kvm/blob/master/docs/08-bootstrapping-kubernetes-controllers.md#configure-the-kubernetes-api-server)연구소에서 내부 클러스터 서비스를 위해 예약됩니다.   * + **서비스 계정 키 쌍**   KubernetesControllerManager는 키 쌍을 사용하여 [서비스 계정 토큰을 생성하고 서명합니다(서비스 계정 관리](https://kubernetes.io/docs/admin/service-accounts-admin/)설명서 참조).  생성service-account인증서 및 개인 키:   |  | | --- | | {    cat > service-account-csr.json <<EOF  {  "CN": "service-accounts",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "ES",  "L": "Barcelona",  "O": "Kubernetes",  "OU": "Kubernetes The Hard Way",  "ST": "Spain"  }  ]  }  EOF    cfssl gencert \  -ca=ca.pem \  -ca-key=ca-key.pem \  -config=ca-config.json \  -profile=kubernetes \  service-account-csr.json | cfssljson -bare service-account    }    결과:    service-account-key.pem  service-account.pem |   **클라이언트 및 서버 인증서 배포**  각 작업자 인스턴스에 적절한 인증서와 개인 키를 복사합니다.   |  | | --- | | for node in worker00 worker01 worker02; do  for key in ${node}-key.pem ${node}.pem kube-proxy-key.pem kube-proxy.pem ca.pem ; do  kcli scp ${key} ${node}:~  done  done |   각 컨트롤러 인스턴스에 적절한 인증서와 개인 키를 복사합니다.   |  | | --- | | for node in master00 master01 master02; do  for key in ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem service-account-key.pem service-account.pem; do  kcli scp ${key} ${node}:~  done  done |   master1/2에 서버 / 클라이언트 인증서 배포   |  | | --- | | kube-proxy, kube-controller-manager, kube-scheduler, 및 kubelet클라이언트 인증서는  클라이언트 인증 구성 파일을 생성하는 데 사용 | |

kubernetes는 아래의 인증서가 필요합니다.

* API 서버에 클러스터 관리자(admin) 인증을 위한 클라이언트 인증서
* API 서버에서 kubelet과 통신을 위한 클라이언트 인증서
* 컨트롤러 매니저와 API 서버 간의 통신을 위한 클라이언트 인증서/kubeconfig
* kube-proxy를 위한 클라이언트 인증서
* 스케줄러와 API 서버간 통신을 위한 클라이언트 인증서/kubeconfig
* API 서버 엔드포인트를 위한 서버 인증서

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 104. kubeconfig 구성파일 배포

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-05-kubernetes-configuration-files-a04bf9a73866ab83bc57fb0e5131f1a444b5400a1be313b8fa11976008c0a542.png |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **클라이언트 인증 구성**  kuberconfigs라고도하는[Kubernetes 구성 파일](https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/)을 생성하여 Kubernetes 클라이언트가 Kubernetes API 서버를 찾고 인증 할 수 있도록합니다.   |  | | --- | | 여기서는 controller manager, kubelet, kube-proxy, 및 scheduler클라이언트와 admin사용자를 위한 kubeconfig파일을 생성할 것입니다.  이 모든 것들은 kub-api 서버의 client입니다. 즉 모든 kubeconfig 파일들은 kube-api서버와 통신해야 합니다.  우리는 여기서 kube-api서버와 통신하기위해서는 Loadbalacer를 두어서 통신하도록 하였습니다. 그래서 그것과 통신하는 client module(kube-proxy)의 경우는 LB를 통해서 통신할 것이고, schuedler,controller-manager, admin은 loopback 통신을 할것입니다.   * kubectl config set-cluster : 클러스터 위치에 대한 구성을 설정 * kubectl config set-credentials : 인증에 사용되는 사용자 이름 및 클라이언트 인증서를 설정 * kubectl config set-context default : 디폴트 컨텍스트를 설정 * kubectl config use-context default : 위 구성으로 현재 컨텍스트를 설정 |   각 kubeconfig들은 Kubernetes API 서버가 연결되어 있어야합니다. 고 가용성을 지원하기 위해로드 밸런서에 할당 된 IP 주소가 사용됩니다  PUBLIC\_ADDRESS는 LoadbalancerIP주소입니다. 이것은 Kubernetes클러스터 내의 모든 인스턴스와 베어 메탈 자체의 모든 API요청에 대한 진입점입니다.   |  | | --- | | [root@osc-ceph02-openinfraday renault\_edu]# KUBERNETES\_PUBLIC\_ADDRESS=$(kcli info vm loadbalancer | grep "ip:" | awk '{print$2}' | tr -d '[:space:]')  [root@osc-ceph02-openinfraday renault\_edu]# echo $KUBERNETES\_PUBLIC\_ADDRESS  192.168.111.202 |   Kubelets에 대한 Kubeconfig파일을 생성할 때 Kubelet의 노드 이름과 일치하는 클라이언트 인증서를 사용해야 합니다. 이렇게 하면 Kubelets가 Kubernetes[NodeAuthorizer](https://kubernetes.io/docs/admin/authorization/node/)에 의해 올바르게 인증됩니다.   |  | | --- | | for instance in worker00 worker01 worker02; do  kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://${KUBERNETES\_PUBLIC\_ADDRESS}:6443 \  --kubeconfig=${instance}.kubeconfig    kubectl config set-credentials system:node:${instance}.${DOMAIN} \  --client-certificate=${instance}.pem \  --client-key=${instance}-key.pem \  --embed-certs=true \  --kubeconfig=${instance}.kubeconfig    kubectl config set-context default \  --cluster=kubernetes-the-hard-way \  --user=system:node:${instance}.${DOMAIN} \  --kubeconfig=${instance}.kubeconfig    kubectl config use-context default --kubeconfig=${instance}.kubeconfig  done    결과:  -rw------- 1 root root 6433 Jun 11 02:40 worker00.kubeconfig  -rw------- 1 root root 6437 Jun 11 02:40 worker01.kubeconfig  -rw------- 1 root root 6437 Jun 11 02:40 worker02.kubeconfig |   **Kub proxy Kubernetes구성 파일**  Kubeconfig파일에서 Kube-proxy서비스를 생성합니다.   |  | | --- | | {    kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://${KUBERNETES\_PUBLIC\_ADDRESS}:6443 \  --kubeconfig=kube-proxy.kubeconfig    kubectl config set-credentials system:kube-proxy \  --client-certificate=kube-proxy.pem \  --client-key=kube-proxy-key.pem \  --embed-certs=true \  --kubeconfig=kube-proxy.kubeconfig    kubectl config set-context default \  --cluster=kubernetes-the-hard-way \  --user=system:kube-proxy \  --kubeconfig=kube-proxy.kubeconfig    kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig  } |   kube-controller-manager 구성   |  | | --- | | {  kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://127.0.0.1:6443 \  --kubeconfig=kube-controller-manager.kubeconfig    kubectl config set-credentials system:kube-controller-manager \  --client-certificate=kube-controller-manager.pem \  --client-key=kube-controller-manager-key.pem \  --embed-certs=true \  --kubeconfig=kube-controller-manager.kubeconfig    kubectl config set-context default \  --cluster=kubernetes-the-hard-way \  --user=system:kube-controller-manager \  --kubeconfig=kube-controller-manager.kubeconfig    kubectl config use-context default --kubeconfig=kube-controller-manager.kubeconfig  } |   kube-scheduler 구성   |  | | --- | | {  kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://127.0.0.1:6443 \  --kubeconfig=kube-scheduler.kubeconfig    kubectl config set-credentials system:kube-scheduler \  --client-certificate=kube-scheduler.pem \  --client-key=kube-scheduler-key.pem \  --embed-certs=true \  --kubeconfig=kube-scheduler.kubeconfig    kubectl config set-context default \  --cluster=kubernetes-the-hard-way \  --user=system:kube-scheduler \  --kubeconfig=kube-scheduler.kubeconfig    kubectl config use-context default --kubeconfig=kube-scheduler.kubeconfig  } |   admin 구성   |  | | --- | | {  kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://127.0.0.1:6443 \  --kubeconfig=admin.kubeconfig    kubectl config set-credentials admin \  --client-certificate=admin.pem \  --client-key=admin-key.pem \  --embed-certs=true \  --kubeconfig=admin.kubeconfig    kubectl config set-context default \  --cluster=kubernetes-the-hard-way \  --user=admin \  --kubeconfig=admin.kubeconfig    kubectl config use-context default --kubeconfig=admin.kubeconfig  } |   **Kubernetes구성 파일 배포**  해당 항목 복사kubelet그리고kube-proxy**각 작업자 인스턴스**에 대한 kubeconfig파일:   |  | | --- | | for node in worker00 worker01 worker02  do  kcli scp ${node}.kubeconfig ${node}:~  kcli scp kube-proxy.kubeconfig ${node}:~  done |   해당 항목 복사kube-controller-manager그리고kube-scheduler**각 컨트롤러 인스턴스**에 대한 파일   |  | | --- | | for node in master00 master01 master02; do  for kubeconfig in admin.kubeconfig kube-controller-manager.kubeconfig kube-scheduler.kubeconfig; do  kcli scp ${kubeconfig} ${node}:~  done  done | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 105.5분 읽을 거리

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Component | Type | Certificate Path | CN Name | ALT Names | Organization | Issuer | Expiration | File Type | Purpose | Description |
| Certificate Authority | Server | /etc/kubernetes/pki/ca.crt | kubernetes |  |  | kubernetes | May  9 11:21:40 2029 GMT | Certificate |  | CA server root certificates for Kubernetes API Server |
| Certificate Authority | Server | /etc/kubernetes/pki/ca.key |  |  |  |  |  | Key |  | CA server root certificate key for Kubernetes API Server |
| kube-apiserver | Server | /etc/kubernetes/pki/apiserver.crt | kube-apiserver | DNS:master DNS:kubernetes DNS:kubernetes.default DNS:kubernetes.default.svc DNS:kubernetes.default.svc.cluster.local IP Address:10.96.0.1 IP Address:172.17.0.27 |  | kubernetes | Feb 11 05:39:20 2020 GMT | Certificate | Server Certificate | Certificate to serve Kube-api server |
| kube-apiserver | Server | /etc/kubernetes/pki/apiserver.key |  |  |  |  |  | Key | Server Key | Key to serve Kube-api server |
| kube-apiserver | Server | /etc/kubernetes/pki/ca.crt | kubernetes |  |  | kubernetes | Feb  8 05:39:19 2029 GMT | Certificate | Server CA Certificate | CA Certificate to validate clients connecting to Kube-API Server |
| kube-apiserver | Client (Kubelet) | /etc/kubernetes/pki/apiserver-kubelet-client.crt | kube-apiserver-kubelet-client | | system:masters | kubernetes | Feb 11 05:39:20 2020 GMT | Certificate | Client Cert: Kube API Server to Kubelet | Client Certificate for Kube-API Server to connect to ETCD Server |
| kube-apiserver | Client (Kubelet) | /etc/kubernetes/pki/apiserver-kubelet-client.key |  |  |  |  |  | Key | Client Key: Kube API Server to Kubelet | Client Key for Kube-API Server to connect to ETCD Server |
| kube-apiserver | Client (Etcd) | /etc/kubernetes/pki/apiserver-etcd-client.crt | kube-apiserver-etcd-client | | system:masters | kubernetes | Feb 11 05:39:22 2020 GMT | Certificate | Client Cert: Kube API Server to ETCD | Client Certificate for Kube-API Server to connect to ETCD Server |
| kube-apiserver | Client (Etcd) | /etc/kubernetes/pki/apiserver-etcd-client.key |  |  |  |  |  | Key | Client Key: Kube API Server to ETCD | Client Key for Kube-API Server to connect to ETCD Server |
| kube-apiserver | Client (Etcd) | /etc/kubernetes/pki/etcd/ca.crt | kubernetes |  |  | kubernetes | Feb  8 05:39:21 2029 GMT | Certificate | Client CA File: Kube API Server to ETCD | CA File to validate Kube-API server to ETCD Server Connectivity. The ETCD setup can have a separate CA |
| kubelet | Server | /var/lib/kubelet/pki/kubelet.crt | [node01@1557660157](mailto:node01@1557660157) | |  |  |  | Certificate |  |  |
| kubelet | Server | /var/lib/kubelet/pki/kubelet.key |  |  |  |  |  | Key |  |  |
| kubelet | Client | /var/lib/kubelet/pki/kubelet-client-2019-05-12-11-22-38.pem | system:node:node01 | | system:nodes | kubernetes | May 11 11:18:00 2020 GMT | Certificate |  |  |
| kubelet | Client |  |  |  |  |  |  | Key |  |  |
| Certificate Authority (ETCD) | Server | /etc/kubernetes/pki/etcd/ca.crt | kubernetes |  |  | kubernetes | May  9 11:21:42 2029 GMT | Certificate |  | CA Server root certificates for ETCD Server. (This could be the same as kube-api server or a separate one of its own.) |
| Certificate Authority (ETCD) | Server | /etc/kubernetes/pki/etcd/ca.key |  |  |  |  |  | Key |  | CA Server root certificate key for ETCD Server. (This could be the same as kube-api server or a separate one of its own.) |
| etcd-server |  |  |  |  |  |  |  | Certificate |  |  |
| etcd-server |  |  |  |  |  |  |  | Key |  |  |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 106. Data 암호화

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-06-data-encryption-keys-cfdd7bec4d69173af75822f18b4c7e093bedfcf717092455a261f24efb7364c5.png |

Kubernetes는 클러스터 상태, 응용 프로그램 구성 및 secrets 등 다양한 데이터를 ETCD에 저장합니다.   
Kubernetes는 클러스터 데이터 를 [암호화](https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data) 하는 기능을 지원합니다 .  
Kubernetes Secrets 암호화에 적합한 암호화 키와 [암호화 구성](https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/#understanding-the-encryption-at-rest-configuration) 을 생성합니다 .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 암호화 키를 생성   |  | | --- | | $ ENCRYPTION\_KEY=$(head -c 32 /dev/urandom | base64)  $ echo $ENCRYPTION\_KEY  uAIkNfROR+F/JjQXiXs0CYP6EPkbp/Cin7KhQ4ePXxU= |   encryption-config.yaml암호화 구성 파일을 작성   |  | | --- | | $ cat> encryption-config.yaml <<EOF  kind: EncryptionConfig  apiVersion: v1  resources:  - resources:  - secrets  providers:  - aescbc:  keys:  - name: key1  secret: ${ENCRYPTION\_KEY}  - identity: {}  EOF |  |  | | --- | | 우리는 이 configuration file을 kube-api서버를 구성할때 사용할것이다. |   master 에 파일배포   |  | | --- | | for node in master00 master01 master02; do  kcli scp encryption-config.yaml $node:~  done | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 107. etcd 클러스터 구성

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-07-bootstrapping-etcd-1775741d3f05448404b11c4f2cb17f085d87c9be7455069bf387470a589adeff.png |

Kubernetes 구성 요소는 stateless 합니다. 그러한 클러스터 상태를 [etcd에](https://github.com/coreos/etcd) 저장합니다 . 두개의 노드에  etcd 클러스터를 구성하고, 고 가용성 및 안전한 원격 액세스를 위해 구현합니다.

* etcd
  + 기능 : 분산 시스템의 중요 데이터를 저장하기 위한 신뢰할 수 있는 key-value 저장소입니다.
  + 역할 : 클러스터 내 분산된 머신에 데이터를 저장하고, 모든 머신에서 데이터가 동기화되는지 확인하는 방법을 제공합니다.
* 쿠버네티스 내에서 etcd 역할
  + 쿠버네티스는 etcd를 사용하여 클러스터 상태에 대한 모든 내부 데이터를 저장합니다.
  + 저장된 데이터는 클러스터의 모든 마스터 노드에서 안정적으로 동기화되어야 합니다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 각 클러스터 구성원의 FQDN (정규화 된 도메인 이름)은 DNS를 통해 확인할 수 있어야합니다 (직접 및 역방향 조회) - no problem   |  | | --- | | DOMAIN=k8s-thw.local  for node in master00 master01 master02; do  kcli ssh $node hostnamectl set-hostname ${node}.${DOMAIN}  kcli ssh $node "hostname -f"  done |   **시간 동기화**  분산 시스템은 구성 요소가 동기화되어 있어야합니다. 그러나 인프라의 모든 VM을 따라 시간 동기화를 구성하는 것이 좋습니다. 이를 위해 CentOS에서 패키지로 제공되는 Ansible 및 시스템 역할을 활용할 것입니다.  먼저 베어 메탈 서버에 rhel-system-roles 패키지를 설치합니다   |  | | --- | | yum install rhel-system-roles.noarch ansible |   inventory실제로 인프라의 일부인 모든 서버를 포함하는 기본 인벤토리를 작성   |  | | --- | | [k8s-thw:children]  masters  workers  lb    [masters]  master00  master01  master02    [workers]  worker00  worker01  worker02    [lb]  loadbalancer |   timesync라는 시스템 역할을 호출하는 것으로 구성된 플레이 북을 만듭니다.   |  | | --- | | cat > timesync.yml <<EOF    - hosts: all  become: yes  vars:  timesync\_ntp\_provider: chrony  timesync\_ntp\_servers:  - hostname: pool.ntp.org  iburst: yes  roles:  - rhel-system-roles.timesync    EOF |  |  | | --- | | ansible-playbook -i inventory timesync.yml |   시계가 동기화되어 있는지 확인   |  | | --- | | ansible -i inventory all -a "date" |    master00, master01,와 master02  로그인   |  | | --- | | kcli ssh centos@master00.${DOMAIN} |   etcd 다운로드   |  | | --- | | $ sudo yum install -y wget bind-utils  $ wget https://github.com/etcd-io/etcd/releases/download/v3.4.0/etcd-v3.4.0-linux-amd64.tar.gz  Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.95.139|:443... connected.  HTTP request sent, awaiting response... 200 OK  Length: 17250172 (16M) [application/octet-stream]  Saving to: ‘etcd-v3.4.0-linux-amd64.tar.gz’    100%[===================================================================================>] 17,250,172 3.97MB/s in 5.3s    ${  tar -xvf etcd-v3.4.0-linux-amd64.tar.gz  sudo mv etcd-v3.4.0-linux-amd64/etcd\* /usr/local/bin  } |   etcd 설치   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | CA / etcd 인증서 복사   |  | | --- | | $ {  sudo mkdir -p /etc/etcd /var/lib/etcd  sudo cp ca.pem kubernetes-key.pem kubernetes.pem /etc/etcd/  } |   마스터 (etcd) 노드의 내부 IP 주소 지정   |  | | --- | | INTERNAL\_IP=$(hostname --ip-address)  for node in master00 master01 master02 ; do export IP\_${node}=$(dig +short ${node}) ;done |   호스트네임 설정   |  | | --- | | $ ETCD\_NAME=$(hostname -s)  $ echo $ETCD\_NAME |   시스템 등록   |  | | --- | | cat <<EOF | sudo tee /etc/systemd/system/etcd.service  [Unit]  Description=etcd  Documentation=https://github.com/coreos    [Service]  ExecStart=/usr/local/bin/etcd \\  --name ${ETCD\_NAME} \\  --cert-file=/etc/etcd/kubernetes.pem \\  --key-file=/etc/etcd/kubernetes-key.pem \\  --peer-cert-file=/etc/etcd/kubernetes.pem \\  --peer-key-file=/etc/etcd/kubernetes-key.pem \\  --trusted-ca-file=/etc/etcd/ca.pem \\  --peer-trusted-ca-file=/etc/etcd/ca.pem \\  --peer-client-cert-auth \\  --client-cert-auth \\  --initial-advertise-peer-urls https://${INTERNAL\_IP}:2380 \\  --listen-peer-urls https://${INTERNAL\_IP}:2380 \\  --listen-client-urls https://${INTERNAL\_IP}:2379,https://127.0.0.1:2379 \\  --advertise-client-urls https://${INTERNAL\_IP}:2379 \\  --initial-cluster-token etcd-cluster-0 \\  --initial-cluster master00=https://${IP\_master00}:2380,master01=https://${IP\_master01}:2380,master02=https://${IP\_master02}:2380 \\  --initial-cluster-state new \\  --data-dir=/var/lib/etcd  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |  |  | | --- | | $ {  sudo systemctl daemon-reload  sudo systemctl enable etcd  sudo systemctl start etcd  }  Created symlink from /etc/systemd/system/multi-user.target.wants/etcd.service to /etc/systemd/system/etcd.service. |  |  | | --- | | $ sudo ETCDCTL\_API=3 /usr/local/bin/etcdctl member list \  --endpoints=https://127.0.0.1:2379 \  --cacert=/etc/etcd/ca.pem \  --cert=/etc/etcd/kubernetes.pem \  --key=/etc/etcd/kubernetes-key.pem | | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 108-0. Kubernetes 컨트롤 플레인 설치

* [컨트롤 플레인](https://kubernetes.io/ko/docs/concepts/#%ec%bf%a0%eb%b2%84%eb%84%a4%ed%8b%b0%ec%8a%a4-%ec%bb%a8%ed%8a%b8%eb%a1%a4-%ed%94%8c%eb%a0%88%ec%9d%b8)
  + 쿠버네티스 클러스터를 제어하는 ​​서비스 (by 컨트롤 플레인 컴포넌트)
  + 쿠버네티스 오브젝트의 레코드와 상태를 관리
  + 클러스터에 대한 글로벌한 의사결정, 이벤트에 대한 탐지 및 응답 수행
* 컨트롤 플레인 컴포넌트 (마스터)
  + **kube-apiserver** : 쿠버네티스 API 제공. 사용자 - 클러스터 간 인터페이스
  + **etcd** : 쿠버네티스 클러스터 데이터 저장소
  + **kube-scheduler** : 가용한 워커 노드에 파드(pods) 스케줄링
  + **kube-controller-manager** : 다양한 기능을 제공하는 4가지 종류의 컨트롤러를 실행

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-08-bootstrapping-kubernetes-controllers-29482d342abdb83e592391394c48736a6348e58bc338be498e192746e5d7c369.png |

이 실습에서는 3 개의 컴퓨팅 인스턴스에서 Kubernetes 컨트롤 플레인을 부트 스트랩하고 고 가용성을 위해 구성합니다. 또한 사용 [이미 제공된 부하 분산](https://github.com/alosadagrande/kubernetes-the-hard-way-libvirt-kvm/blob/master/docs/03-compute-resources.md#load-balancer-service) 원격 클라이언트로는 Kubernetes의 API 서버를 노출합니다. 다음 구성 요소는 각 마스터 또는 컨트롤러에 설치됩니다 Kubernetes API Server, Scheduler그리고 Controller Manager.

**모든 컨트롤러 노드 cli** 에서 selinux를 비활성화

|  |
| --- |
| sudo sed -i -e 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config |

### Kubernetes Controller 바이너리 다운로드 및 설치

|  |
| --- |
| wget \  "https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kube-apiserver" \  "https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kube-controller-manager" \  "https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kube-scheduler" \  "https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl" |

Kubernetes 바이너리를 설치

|  |
| --- |
| {  chmod +x kube-apiserver kube-controller-manager kube-scheduler kubectl  sudo mv kube-apiserver kube-controller-manager kube-scheduler kubectl /usr/local/bin/  } |

### Kubernetes API 서버 구성

|  |
| --- |
| {  sudo mkdir -p /var/lib/kubernetes/    sudo mv ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \  service-account-key.pem service-account.pem \  encryption-config.yaml /var/lib/kubernetes/  } |

인스턴스 내부 IP 주소는 API 서버를 클러스터 멤버에게 알리는 데 사용

|  |
| --- |
| INTERNAL\_IP=$(hostname --ip-address)  for node in master00 master01 master02 ; do export IP\_${node}=$(dig +short ${node}) ;done |

kube-apiserver.service단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/kube-apiserver.service  [Unit]  Description=Kubernetes API Server  Documentation=https://github.com/kubernetes/kubernetes    [Service]  ExecStart=/usr/local/bin/kube-apiserver \\  --advertise-address=${INTERNAL\_IP} \\  --allow-privileged=true \\  --apiserver-count=3 \\  --audit-log-maxage=30 \\  --audit-log-maxbackup=3 \\  --audit-log-maxsize=100 \\  --audit-log-path=/var/log/audit.log \\  --authorization-mode=Node,RBAC \\  --bind-address=0.0.0.0 \\  --client-ca-file=/var/lib/kubernetes/ca.pem \\  --enable-admission-plugins=NamespaceLifecycle,NodeRestriction,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota \\  --etcd-cafile=/var/lib/kubernetes/ca.pem \\  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\  --etcd-servers=https://${IP\_master00}:2379,https://${IP\_master01}:2379,https://${IP\_master02}:2379 \\  --event-ttl=1h \\  --encryption-provider-config=/var/lib/kubernetes/encryption-config.yaml \\  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\  --kubelet-client-certificate=/var/lib/kubernetes/kubernetes.pem \\  --kubelet-client-key=/var/lib/kubernetes/kubernetes-key.pem \\  --kubelet-https=true \\  --runtime-config=api/all \\  --service-account-key-file=/var/lib/kubernetes/service-account.pem \\  --service-cluster-ip-range=10.32.0.0/24 \\  --service-node-port-range=30000-32767 \\  --tls-cert-file=/var/lib/kubernetes/kubernetes.pem \\  --tls-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \\  --v=2  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |

### Kubernetes Controller Manager 구성

|  |
| --- |
| sudo mv kube-controller-manager.kubeconfig /var/lib/kubernetes/ |

시스템 kube-controller-manager.service단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/kube-controller-manager.service  [Unit]  Description=Kubernetes Controller Manager  Documentation=https://github.com/kubernetes/kubernetes    [Service]  ExecStart=/usr/local/bin/kube-controller-manager \\  --address=0.0.0.0 \\  --allocate-node-cidrs=true \\  --cluster-cidr=10.200.0.0/16 \\  --cluster-name=kubernetes \\  --cluster-signing-cert-file=/var/lib/kubernetes/ca.pem \\  --cluster-signing-key-file=/var/lib/kubernetes/ca-key.pem \\  --kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\  --leader-elect=true \\  --root-ca-file=/var/lib/kubernetes/ca.pem \\  --service-account-private-key-file=/var/lib/kubernetes/service-account-key.pem \\  --service-cluster-ip-range=10.32.0.0/24 \\  --use-service-account-credentials=true \\  --v=2  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |

### Kubernetes 스케줄러 구성

|  |
| --- |
| sudo mv kube-scheduler.kubeconfig /var/lib/kubernetes/ |

kube-scheduler.yaml구성 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/kubernetes/config/kube-scheduler.yaml  apiVersion: kubescheduler.config.k8s.io/v1alpha1  kind: KubeSchedulerConfiguration  clientConnection:  kubeconfig: "/var/lib/kubernetes/kube-scheduler.kubeconfig"  leaderElection:  leaderElect: true  EOF |

시스템 kube-scheduler.service단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/kube-scheduler.service  [Unit]  Description=Kubernetes Scheduler  Documentation=https://github.com/kubernetes/kubernetes    [Service]  ExecStart=/usr/local/bin/kube-scheduler \\  --config=/etc/kubernetes/config/kube-scheduler.yaml \\  --v=2  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |

### 컨트롤러 서비스 시작

|  |
| --- |
| {  sudo systemctl daemon-reload  sudo systemctl enable kube-apiserver kube-controller-manager kube-scheduler --now  } |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 108-1. Kubelet 인증을위한 RBAC

Kubernetes API 서버가 각 작업자 노드의 Kubelet API에 액세스 할 수 있도록 RBAC 권한을 구성합니다

* RBAC (Role-Based Access Control) : 사용자의 역할을 기반으로 리소스에 대한 액세스를 제어하는 ​​방법
* 예 : kube-apiserver API를 통해 파드(Pod)에서 메트릭, 로그 및 실행 명령을 검색하려면 Kubelet API에 액세스해야합니다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 포드 관리와 관련된 가장 일반적인 작업을 Kubelet API에 액세스하고 수행 할 수있는 권한을 가진  the system:kube-apiserver-to-kubelet  [ClusterRole](https://kubernetes.io/docs/admin/authorization/rbac/#role-and-clusterrole) 구성합니다.   |  | | --- | | cat <<EOF | kubectl apply --kubeconfig admin.kubeconfig -f -  apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRole  metadata:  annotations:  rbac.authorization.kubernetes.io/autoupdate: "true"  labels:  kubernetes.io/bootstrapping: rbac-defaults  name: system:kube-apiserver-to-kubelet  rules:  - apiGroups:  - ""  resources:  - nodes/proxy  - nodes/stats  - nodes/log  - nodes/spec  - nodes/metrics  verbs:  - "\*"  EOF |  |  | | --- | | [clusterrole.rbac.authorization.k8s.io/system:kube-apiserver-to-kubelet](http://clusterrole.rbac.authorization.k8s.io/system:kube-apiserver-to-kubelet) created |   system:kube-apiserver-to-kubeletClusterRole을 kubernetes사용자 에게 바인딩   |  | | --- | | cat <<EOF | kubectl apply --kubeconfig admin.kubeconfig -f -  apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRoleBinding  metadata:  name: system:kube-apiserver  namespace: ""  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: system:kube-apiserver-to-kubelet  subjects:  - apiGroup: rbac.authorization.k8s.io  kind: User  name: kube-apiserver  EOF |  |  | | --- | | [clusterrolebinding.rbac.authorization.k8s.io/system:kube-apiserver](http://clusterrolebinding.rbac.authorization.k8s.io/system:kube-apiserver) created |  |  | | --- | | kubectl get componentstatuses --kubeconfig admin.kubeconfig -o yaml | egrep "kind|name|message" |  |  | | --- | | kind: ComponentStatus  name: controller-manager  - message: ok  status: "True"  kind: ComponentStatus  name: scheduler  - message: '{"health":"true"}'  status: "True"  kind: ComponentStatus  name: etcd-2  - message: '{"health":"true"}'  status: "True"  kind: ComponentStatus  name: etcd-1  - message: '{"health":"true"}'  status: "True"  kind: ComponentStatus  name: etcd-0 |   베어 메탈 노드에서 로드 밸런서가 올바르게 구성되었는지 확인   |  | | --- | | KUBERNETES\_PUBLIC\_ADDRESS=$(kcli ssh loadbalancer "hostname --ip-address")  curl --cacert ca.pem https://${KUBERNETES\_PUBLIC\_ADDRESS}:6443/version    {  "major": "1",  "minor": "16",  "gitVersion": "v1.16.2",  "gitCommit": "c97fe5036ef3df2967d086711e6c0c405941e14b",  "gitTreeState": "clean",  "buildDate": "2019-10-15T19:09:08Z",  "goVersion": "go1.12.10",  "compiler": "gc",  "platform": "linux/amd64"  } | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 109. 쿠버네티스 worker node 구성

컨트롤 플레인 컴포넌트 (워커 노드)

* [**kubelet**](https://kubernetes.io/docs/admin/kubelet) : 워커 노드에서 실행되는 에이전트. Kubelet은 파드 스펙(PodSpec)을 받아, 파드에서 컨테이너가 동작하도록 관리합니다.
* [**kube-proxy**](https://kubernetes.io/docs/concepts/cluster-administration/proxies) : 워커 노드에서 실행되는 네트워크 프록시로서, 호스트의 네트워크 규칙(iptables)을 관리하고 요청에 대한 포워딩을 책임집니다. (NodePort로 들어온 트래픽을 클러스터 내의 적절한 파드로 리다이렉팅)
* 컨테이너 런타임 : 컨테이너 실행을 담당하는 소프트웨어

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-09-bootstrapping-kubernetes-workers-7ee93616e6589f2b0d6a577be18aaa4988860b61a75dcf1bffb3be7d519ebdd0.png |

docker가 구성되어 이는 노드에 kubelet과 kube-proxy를 구성함.

FQDN (정규화 된 도메인 이름)은 DNS를 통해 확인

|  |
| --- |
| DOMAIN=k8s-thw.local  for node in worker00 worker01 worker02; do  kcli ssh $node hostnamectl set-hostname ${node}.${DOMAIN}  kcli ssh $node "hostname -f"  done |

Docker를 설치

|  |
| --- |
| {  sudo yum install epel-release -y  sudo yum install socat conntrack ipset wget jq vim -y  sudo yum-config-manager --disable epel  } |

|  |
| --- |
| sudo setenforce 0  sudo sed -i -e 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config |

|  |
| --- |
| wget \  https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.16.1/crictl-v1.16.1-linux-amd64.tar.gz \  https://github.com/opencontainers/runc/releases/download/v1.0.0-rc9/runc.amd64 \  https://github.com/containernetworking/plugins/releases/download/v0.8.2/cni-plugins-linux-amd64-v0.8.2.tgz \  https://github.com/containerd/containerd/releases/download/v1.3.0/containerd-1.3.0.linux-amd64.tar.gz \  https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl \  https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kube-proxy \  https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubelet |

설치 디렉토리 구성   
**worker**

|  |
| --- |
| sudo mkdir -p \  /etc/cni/net.d \  /opt/cni/bin \  /var/lib/kubelet \  /var/lib/kube-proxy \  /var/lib/kubernetes \  /var/run/kubernetes |

|  |
| --- |
| {  mkdir containerd  tar -xvf crictl-v1.16.1-linux-amd64.tar.gz  tar -xvf containerd-1.3.0.linux-amd64.tar.gz -C containerd  sudo tar -xvf cni-plugins-linux-amd64-v0.8.2.tgz -C /opt/cni/bin/  sudo mv runc.amd64 runc  chmod +x crictl kubectl kube-proxy kubelet runc  sudo mv crictl kubectl kube-proxy kubelet runc /usr/local/bin/  sudo mv containerd/bin/\* /bin/  } |

### CNI 네트워킹 구성

|  |
| --- |
| POD\_CIDR=$(cat /home/centos/pod\_cidr.txt) |

브릿지 네트워크 구성 파일을 작성하십시오.

|  |
| --- |
| cat <<EOF | sudo tee /etc/cni/net.d/10-bridge.conf  {  "cniVersion": "0.3.1",  "name": "bridge",  "type": "bridge",  "bridge": "cnio0",  "isGateway": true,  "ipMasq": true,  "ipam": {  "type": "host-local",  "ranges": [  [{"subnet": "${POD\_CIDR}"}]  ],  "routes": [{"dst": "0.0.0.0/0"}]  }  }  EOF |

루프백 네트워크 구성 파일을 작성하십시오.

|  |
| --- |
| cat <<EOF | sudo tee /etc/cni/net.d/99-loopback.conf  {  "cniVersion": "0.3.1",  "name": "lo",  "type": "loopback"  }  EOF |

### 컨테이너 구성

|  |
| --- |
| sudo mkdir -p /etc/containerd/ |

|  |
| --- |
| cat << EOF | sudo tee /etc/containerd/config.toml  [plugins]  [plugins.cri.containerd]  snapshotter = "overlayfs"  [plugins.cri.containerd.default\_runtime]  runtime\_type = "io.containerd.runtime.v1.linux"  runtime\_engine = "/usr/local/bin/runc"  runtime\_root = ""  EOF |

시스템 containerd.service단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/containerd.service  [Unit]  Description=containerd container runtime  Documentation=https://containerd.io  After=network.target    [Service]  ExecStartPre=/sbin/modprobe overlay  ExecStart=/bin/containerd  Restart=always  RestartSec=5  Delegate=yes  KillMode=process  OOMScoreAdjust=-999  LimitNOFILE=1048576  LimitNPROC=infinity  LimitCORE=infinity    [Install]  WantedBy=multi-user.target  EOF |

### Kubelet 구성

|  |
| --- |
| {  SHORTNAME=$(hostname -s)  sudo mv ${SHORTNAME}-key.pem ${SHORTNAME}.pem /var/lib/kubelet/  sudo mv ${SHORTNAME}.kubeconfig /var/lib/kubelet/kubeconfig  sudo mv ca.pem /var/lib/kubernetes/  } |

### kubelet-config.yaml구성 파일을 작성

|  |
| --- |
| POD\_CIDR=$(cat /home/centos/pod\_cidr.txt)  SHORTNAME=$(hostname -s)    cat <<EOF | sudo tee /var/lib/kubelet/kubelet-config.yaml  kind: KubeletConfiguration  apiVersion: kubelet.config.k8s.io/v1beta1  authentication:  anonymous:  enabled: false  webhook:  enabled: true  x509:  clientCAFile: "/var/lib/kubernetes/ca.pem"  authorization:  mode: Webhook  clusterDomain: "cluster.local"  clusterDNS:  - "10.32.0.10"  podCIDR: "${POD\_CIDR}"  resolvConf: "/etc/resolv.conf"  runtimeRequestTimeout: "15m"  tlsCertFile: "/var/lib/kubelet/${SHORTNAME}.pem"  tlsPrivateKeyFile: "/var/lib/kubelet/${SHORTNAME}-key.pem"  EOF |

kubelet.service단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/kubelet.service  [Unit]  Description=Kubernetes Kubelet  Documentation=https://github.com/kubernetes/kubernetes  After=containerd.service  Requires=containerd.service    [Service]  ExecStart=/usr/local/bin/kubelet \\  --config=/var/lib/kubelet/kubelet-config.yaml \\  --container-runtime=remote \\  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\  --image-pull-progress-deadline=2m \\  --kubeconfig=/var/lib/kubelet/kubeconfig \\  --network-plugin=cni \\  --register-node=true \\  --v=2  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |

### Kubernetes 프록시 구성

|  |
| --- |
| sudo mv kube-proxy.kubeconfig /var/lib/kube-proxy/kubeconfig |

|  |
| --- |
| cat <<EOF | sudo tee /var/lib/kube-proxy/kube-proxy-config.yaml  kind: KubeProxyConfiguration  apiVersion: kubeproxy.config.k8s.io/v1alpha1  clientConnection:  kubeconfig: "/var/lib/kube-proxy/kubeconfig"  mode: "iptables"  clusterCIDR: "10.200.0.0/16"  EOF |

kube-proxy.service 시스템 단위 파일을 작성

|  |
| --- |
| cat <<EOF | sudo tee /etc/systemd/system/kube-proxy.service  [Unit]  Description=Kubernetes Kube Proxy  Documentation=https://github.com/kubernetes/kubernetes    [Service]  ExecStart=/usr/local/bin/kube-proxy \\  --config=/var/lib/kube-proxy/kube-proxy-config.yaml  Restart=on-failure  RestartSec=5    [Install]  WantedBy=multi-user.target  EOF |

### 서비스 시작

|  |
| --- |
| {  sudo systemctl daemon-reload  sudo systemctl enable containerd kubelet kube-proxy --now  } |

check

|  |
| --- |
| kcli ssh master00 \  "kubectl get nodes --kubeconfig admin.kubeconfig -o wide" |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 1009. 5분 휴식

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 1010. 쿠버네티스 worker node-2 구성

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | |  | | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 1011.원격 액세스를위한 kubectl 구성

|  |
| --- |
| **목표구성도** |
| _scroll_external/remote/kthw-10-configuring-kubectl-72ffd0024b9138cd006e95d52f7f0c43f0fe9e18957641aba494157675826bc0.png |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 전체 환경 체크   |  | | --- | | vagrant@master-1:~$ kubectl get nodes --kubeconfig admin.kubeconfig  NAME STATUS ROLES AGE VERSION  worker-1 NotReady <none> 7h49m v1.13.0  worker-2 NotReady <none> 87s v1.13.0  vagrant@master-1:~$ kubectl get componentstatuses  NAME STATUS MESSAGE ERROR  controller-manager Healthy ok  scheduler Healthy ok  etcd-0 Healthy {"health":"true"}  etcd-1 Healthy {"health":"true"} |    admin용 kubeconfig 파일 구성   |  | | --- | | {  KUBERNETES\_PUBLIC\_ADDRESS=$(kcli ssh loadbalancer "hostname --ip-address")    kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://${KUBERNETES\_PUBLIC\_ADDRESS}:6443    kubectl config set-credentials admin \  --client-certificate=admin.pem \  --client-key=admin-key.pem    kubectl config set-context kubernetes-the-hard-way \  --cluster=kubernetes-the-hard-way \  --user=admin    kubectl config use-context kubernetes-the-hard-way  } |   Code Block 2 master-1에서 작업함  check   |  | | --- | | kubectl get componentstatuses -o yaml | egrep "name:|kind:|message:"    - message: ok  kind: ComponentStatus  name: scheduler  - message: ok  kind: ComponentStatus  name: controller-manager  - message: '{"health":"true"}'  kind: ComponentStatus  name: etcd-0  - message: '{"health":"true"}'  kind: ComponentStatus  name: etcd-1  - message: '{"health":"true"}'  kind: ComponentStatus  name: etcd-2  kind: List |   Code Block 3 내부 사용용 admin.kubeconfig  **external access**  이 경우 랩톱 또는 서버로 작업하고 Kubernetes 클러스터에 액세스하려는 사람에게 액세스 권한을 부여하려고합니다. 이 서버 또는 랩톱은 모든 VM이 실행중인 베어 메탈 서버에 도달 할 수 있어야합니다.   |  | | --- | | 각 kubeconfig에는 Kubernetes API 서버가 연결되어 있어야합니다. 고 가용성을 지원하기 위해 Kubernetes API 서버 앞에있는 외부로드 밸런서에 할당 된 IP 주소가 사용됩니다.    베어 메탈 호스트의 IP 주소를 가져옵니다. 원격 사용자의 장치에서 확인할 수 있고 접근 가능한지 확인하십시오.    # getent hosts smc-master  10.19.138.41 smc-master.cloud.lab  admin사용자 인증에 적합한 kubeconfig 파일을 생성하십시오 .      {  REMOTESERVER=smc-master  KUBERNETES\_PUBLIC\_ADDRESS=$(getent hosts smc-master.cloud.lab | awk '{print$1}' | tr -d '[:space:]')    kubectl config set-cluster kubernetes-the-hard-way \  --certificate-authority=ca.pem \  --embed-certs=true \  --server=https://${KUBERNETES\_PUBLIC\_ADDRESS}:6443 \  --kubeconfig=${REMOTESERVER}.kubeconfig    kubectl config set-credentials admin \  --client-certificate=admin.pem \  --client-key=admin-key.pem \  --embed-certs=true \  --kubeconfig=${REMOTESERVER}.kubeconfig    kubectl config set-context kubernetes-the-hard-way \  --cluster=kubernetes-the-hard-way \  --user=admin \  --kubeconfig=${REMOTESERVER}.kubeconfig    kubectl config use-context kubernetes-the-hard-way --kubeconfig=${REMOTESERVER}.kubeconfig  }  원격 사용자의 경우 Kubernetes 클러스터 (KUBERNETES\_PUBLIC\_ADDRESS)의 시작점이 베어 메탈 호스트입니다.    kubeconfig 파일을 만든 후에는로드 밸런서 VM을 외부에 노출시켜야합니다. 가장 쉬운 방법은 iptables foo를 활용하는 것입니다. 기본적으로 모든 요청을 베어 메탈 서버 포트 tcp / 6443에로드 밸런서 포트 tcp / 6443으로 전달합니다. 동일한 방식으로 요청을 세 가지 마스터 중 하나로 리디렉션합니다.    참고 :    로드 밸런서 VM의 IP는 192.168.111.68입니다.  베어 메탈 서버의 IP는 10.19.138.41입니다.  가상 서브넷의 Libvirt 인터페이스는 k8s-net입니다. 기본적으로 192.168.111.0/24 서브넷이 생성되는 인터페이스  em1은 베어 메탈 서버에 도달 할 수있는 인터페이스이며 10.19.138.41에 할당됩니다.  6443 / tcp는 Kubernetes API 서버 포트입니다.  베어 메탈 호스트에 다음 방화벽 규칙을 적용하십시오.    # connections from outside  $ iptables -I FORWARD -o k8s-net -d 192.168.111.68 -j ACCEPT  $ iptables -t nat -I PREROUTING -p tcp --dport 6443 -j DNAT --to 192.168.111.68:6443    # Masquerade local subnet  $ iptables -I FORWARD -o k8s-net -d 192.168.111.68 -j ACCEPT  $ iptables -t nat -A POSTROUTING -s 192.168.111.0/24 -j MASQUERADE  $ iptables -A FORWARD -o k8s-net -m state --state RELATED,ESTABLISHED -j ACCEPT  $ iptables -A FORWARD -i k8s-net -o em1 -j ACCEPT  $ iptables -A FORWARD -i k8s-net -o lo -j ACCEPT  그런 다음 적용되면 kubeconfig 파일 ($ {REMOTESERVER} .kubeconfig)을 클러스터에 도달하려는 원격 사용자에게 보낼 수 있습니다.    원격 장치는 최소한 베어 메탈 서버의 포트 tcp / 6443에 도달 할 수 있어야합니다.    마지막으로 랩톱에서 클러스터를 관리 할 수 ​​있는지 확인하십시오 (내 랩탑 @ Katatonic에서이 명령을 실행하고 있는지 확인하십시오).    alosadag@Katatonic $ kubectl get nodes --kubeconfig smc-master.kubeconfig  NAME STATUS ROLES AGE VERSION  worker00.k8s-thw.local Ready <none> 4h14m v1.16.2  worker01.k8s-thw.local Ready <none> 4h14m v1.16.2  worker02.k8s-thw.local Ready <none> 4h14m v1.16.2 |   Code Block 4 example |

## 1012.네트워크 프로비저닝

|  |
| --- |
| **목표구성도** |
|  |

### Provisioning Pod Network Routes

노드에 예약 된 포드는 노드의 포드 CIDR 범위에서 IP 주소를받습니다. 이 시점에서 포드는 가상 서버의 네트워크 경로가 누락되어 다른 노드에서 실행중인 다른 포드와 통신 할 수 없습니다.

이 실습에서는 노드의 포드 CIDR 범위를 노드의 내부 IP 주소에 매핑하는 각 worker  노드에 대한 경로를 만듭니다.

#### 라우팅 테이블

|  |
| --- |
| for node in worker00 worker01 worker02; do  kcli ssh ${node} "hostname -f; hostname -i; cat /home/centos/pod\_cidr.txt" | tr '\r\n' ' '  done      worker00.k8s-thw.local 192.168.111.198 10.200.0.0/24  worker01.k8s-thw.local 192.168.111.253 10.200.1.0/24  worker02.k8s-thw.local 192.168.111.158 10.200.2.0/24 |

#### Routes

 인스턴스에 대한 네트워크 경로를 만듭니다.

* worker00

|  |
| --- |
| [root@worker00 ~]# ip route add 10.200.2.0/24 via 192.168.111.158  [root@worker00 ~]# ip route add 10.200.1.0/24 via 192.168.111.253 |

* worker01

|  |
| --- |
| [root@worker01 ~]# ip route add 10.200.2.0/24 via 192.168.111.158  [root@worker01 ~]# ip route add 10.200.0.0/24 via 192.168.111.198 |

* worker02

|  |
| --- |
| [root@worker02 ~]# ip route add 10.200.1.0/24 via 192.168.111.253  [root@worker02 ~]# ip route add 10.200.0.0/24 via 192.168.111.198 |

* route information

|  |
| --- |
| Kernel IP routing table  Destination Gateway Genmask Flags Metric Ref Use Iface  0.0.0.0 192.168.111.1 0.0.0.0 UG 0 0 0 eth0  10.200.0.0 192.168.111.198 255.255.255.0 UG 0 0 0 eth0  10.200.1.0 0.0.0.0 255.255.255.0 U 0 0 0 cnio0  10.200.2.0 192.168.111.158 255.255.255.0 UG 0 0 0 eth0  169.254.0.0 0.0.0.0 255.255.0.0 U 1002 0 0 eth0  192.168.111.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0 |

======================================================================================================================================================================================

CNI-weave를 구성할 예정임.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cni plugin 다운로드  - worker1 / worker2   |  | | --- | | wget https://github.com/containernetworking/plugins/releases/download/v0.8.2/cni-plugins-linux-amd64-v0.8.2.tgz  sudo tar -xzvf cni-plugins-linux-amd64-v0.8.2.tgz --directory /opt/cni/bin/ |   weave network 배포 - master-1   |  | | --- | | vagrant@master-1:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')" |  |  | | --- | | serviceaccount/weave-net created [clusterrole.rbac.authorization.k8s.io/weave-net](http://clusterrole.rbac.authorization.k8s.io/weave-net) created [clusterrolebinding.rbac.authorization.k8s.io/weave-net](http://clusterrolebinding.rbac.authorization.k8s.io/weave-net) created [role.rbac.authorization.k8s.io/weave-net](http://role.rbac.authorization.k8s.io/weave-net) created [rolebinding.rbac.authorization.k8s.io/weave-net](http://rolebinding.rbac.authorization.k8s.io/weave-net) created daemonset.extensions/weave-net created |  |  | | --- | | Weave uses POD CIDR of 10.32.0.0/12 by default. |  |  | | --- | | [vagrant@master-1 ~]$ kubectl get pods -n kube-system  NAME READY STATUS RESTARTS AGE  weave-net-glq6r 0/2 ContainerCreating 0 8h  weave-net-m7cp9 0/2 ContainerCreating 0 8h | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 1014. DNS Cluster Add-on 배포

[CoreDNS](https://coredns.io/)를  배포 합니다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | kubectl apply -f https://storage.googleapis.com/kubernetes-the-hard-way/coredns.yaml |  |  | | --- | | serviceaccount/coredns created clusterrole.rbac.authorization.k8s.io/system:coredns created clusterrolebinding.rbac.authorization.k8s.io/system:coredns created configmap/coredns created deployment.apps/coredns created service/kube-dns created |  |  | | --- | | vagrant@master-1:~$ kubectl get pods -l k8s-app=kube-dns -n kube-system |  |  | | --- | | NAME READY STATUS RESTARTS AGE coredns-69cbb76ff8-mwn45 1/1 Running 0 57s coredns-69cbb76ff8-tzvz6 1/1 Running 0 57s |   확인  busybox 배포   |  | | --- | | $ kubectl run --generator=run-pod/v1 busybox --image=busybox:1.28 --command -- sleep 3600  pod/busybox created  vagrant@master-1:/$ kubectl get pods -l run=busybox  NAME READY STATUS RESTARTS AGE  busybox 1/1 Running 0 35s    Server: 10.32.0.10  Address 1: 10.32.0.10 kube-dns.kube-system.svc.cluster.local    Name: kubernetes  Address 1: 10.32.0.1 kubernetes.default.svc.cluster.local |  |  | | --- | | **busybox란?**  **리눅스 상에서 자주 사용되는 명령어들만 모은 압축파일.**  Busybox에서는 각각에 함수들을 최소 사이즈로 다시 구현하였으며 제한된 자원을 가진 임베디드 플랫폼에서는 기반 툴로 사용됩니다. 안드로이드 기본 커널에서는 파일복사 명령어 등이 없기 때문에 system에 접근해서 시스템 파일을 복사하기 위해서는 이 명령어가 필요합니다 | |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |

## 1015.deployments

nginx 배포

|  |
| --- |
| vagrant@master-1:/$ kubectl create deployment nginx --image=nginx  deployment.apps/nginx created  vagrant@master-1:/$ kubectl get pods -l app=nginx  NAME READY STATUS RESTARTS AGE  nginx-5c7588df-6478h 0/1 ContainerCreating 0 5s  vagrant@master-1:/$ kubectl expose deploy nginx --type=NodePort --port 80  service/nginx exposed  vagrant@master-1:/$ PORT\_NUMBER=$(kubectl get svc -l app=nginx -o jsonpath="{.items[0].spec.ports[0].nodePort}")  vagrant@master-1:/$ echo $PORT\_NUMBER  30217  vagrant@master-1:/$ curl http://worker-1:$PORT\_NUMBER  <!DOCTYPE html>  <html>  <head>  <title>Welcome to nginx!</title>  <style>  body {  width: 35em;  margin: 0 auto;  font-family: Tahoma, Verdana, Arial, sans-serif;  }  </style>  </head>  <body>  <h1>Welcome to nginx!</h1>  <p>If you see this page, the nginx web server is successfully installed and  working. Further configuration is required.</p>    <p>For online documentation and support please refer to  <a href="http://nginx.org/">nginx.org</a>.<br/>  Commercial support is available at  <a href="http://nginx.com/">nginx.com</a>.</p>    <p><em>Thank you for using nginx.</em></p>  </body>  </html> |

|  |
| --- |
| vagrant@master-1:/$ POD\_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")  vagrant@master-1:/$ kubectl logs $POD\_NAME  10.32.0.1 - - [15/Sep/2019:03:14:53 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.58.0" "-"  10.40.0.0 - - [15/Sep/2019:03:15:00 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.58.0" "-"  vagrant@master-1:/$ kubectl get pods -l app=nginx  NAME READY STATUS RESTARTS AGE  nginx-5c7588df-6478h 1/1 Running 0 16m |

Code Block 5 log 보기

|  |
| --- |
| vagrant@master-1:/$ kubectl exec -ti $POD\_NAME -- nginx -v  nginx version: nginx/1.17.3  vagrant@master-1:/$ kubectl exec -ti $POD\_NAME /bin/bash  root@nginx-5c7588df-6478h:/# ls  bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var |

## 1016. nginx 배포

* yaml file 만들기

|  |
| --- |
| [root@idc01 ansible-vbox-vagrant-kubernetes]# cat nginx.yaml  apiVersion: apps/v1  kind: Deployment  metadata:  name: nginx-deployment  spec:  selector:  matchLabels:  app: nginx  replicas: 3  template:  metadata:  labels:  app: nginx  spec:  containers:  - name: my-echo  image: gcr.io/google\_containers/echoserver:1.8  ---    apiVersion: v1  kind: Service  metadata:  name: nginx-service-np  labels:  name: nginx-service-np  spec:  type: NodePort  ports:  - port: 8082 # Cluster IP http://10.109.199.234:8082  targetPort: 8080 # Application port  nodePort: 30000 # (EXTERNAL-IP VirtualBox IPs) http://192.168.50.11:30000/ http://192.168.50.12:30000/ http://192.168.50.13:30000/  protocol: TCP  name: http  selector:  app: nginx    #NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  #...  #service/nginx-service-np NodePort 10.109.199.234 <none> 8082:30000/TCP 18m |

Code Block 6 nginx.yaml

* 서비스 확인

|  |
| --- |
| [vagrant@master-1 ~]$ kubectl get service  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 6h9m  nginx NodePort 10.96.0.187 <none> 80:32557/TCP 4h24m  nginx-service-np NodePort 10.96.0.209 <none> 8082:30000/TCP 173m    [vagrant@master-1 ~]$ kubectl get pod  NAME READY STATUS RESTARTS AGE  busybox 1/1 Running 2 178m  nginx-554b9c67f9-jx2lk 1/1 Running 2 4h26m  nginx-deployment-7bd8658df8-fq85h 1/1 Running 0 166m  nginx-deployment-7bd8658df8-qns6s 1/1 Running 0 166m  nginx-deployment-7bd8658df8-vg28v 1/1 Running 0 166m |

* 실행 확인 및 외부연결

|  |
| --- |
| [vagrant@master-1 ~]$ curl http://192.168.25.21:30000/      Hostname: nginx-deployment-7bd8658df8-vg28v    Pod Information:  -no pod information available-    Server values:  server\_version=nginx: 1.13.3 - lua: 10008    Request Information:  client\_address=10.32.0.1  method=GET  real path=/  query=  request\_version=1.1  request\_uri=http://192.168.25.21:8080/    Request Headers:  accept=\*/\*  host=192.168.25.21:30000  user-agent=curl/7.29.0    Request Body:  -no body in request- |

* check

|  |
| --- |
| [vagrant@master-1 ~]$ curl http://192.168.25.21:30000/ | grep -i host  % Total % Received % Xferd Average Speed Time Time Time Current  Dload Upload Total Spent Left Speed  100 415 0 415 0 0 243k 0 --:--:-- --:--:-- --:--:-- 405k  Hostname: nginx-deployment-7bd8658df8-vg28v  host=192.168.25.21:30000    [vagrant@master-1 ~]$ curl http://192.168.25.22:30000/ | grep -i host  % Total % Received % Xferd Average Speed Time Time Time Current  Dload Upload Total Spent Left Speed  100 415 0 415 0 0 227k 0 --:--:-- --:--:-- --:--:-- 405k  Hostname: nginx-deployment-7bd8658df8-fq85h  host=192.168.25.22:30000 |

* haproxy 구성

|  |
| --- |
| sudo vi /etc/haproxy/haproxy.cfg    listen nginx  bind 192.168.25.30:30000  mode tcp  balance roundrobin  option tcp-check  server worker-1 192.168.25.21:30000 check fall 3 rise 2  server worker-2 192.168.25.22:30000 check fall 3 rise 2 |

* 브라우저에서 LB 확인

|  |
| --- |
|  |

* edit deployment

|  |
| --- |
| kubectl edit deployment nginx-deployment  replicas: 4    67L, 2302C written  deployment.extensions/nginx-deployment edited    [vagrant@master-1 ~]$ kubectl get rs  NAME DESIRED CURRENT READY AGE  nginx-554b9c67f9 1 1 1 5h5m  nginx-deployment-7bd8658df8 4 4 4 3h25m |

* nodes down

|  |
| --- |
| [vagrant@worker-1 ~]$ sudo shutdown -h now    [vagrant@master-1 ~]$ kubectl get nodes  NAME STATUS ROLES AGE VERSION  worker-1 NotReady <none> 6h27m v1.15.3  worker-2 Ready <none> 5h50m v1.15.3    [vagrant@master-1 ~]$ curl http://192.168.25.30:30000      Hostname: nginx-deployment-7bd8658df8-fq85h    Pod Information:  -no pod information available-    Server values:  server\_version=nginx: 1.13.3 - lua: 10008    Request Information:  client\_address=10.40.0.0  method=GET  real path=/  query=  request\_version=1.1  request\_uri=http://192.168.25.30:8080/    Request Headers:  accept=\*/\*  host=192.168.25.30:30000  user-agent=curl/7.29.0    Request Body:  -no body in request-    [vagrant@master-1 ~]$ curl http://192.168.25.30:30000      Hostname: nginx-deployment-7bd8658df8-48l2z    Pod Information:  -no pod information available-    Server values:  server\_version=nginx: 1.13.3 - lua: 10008    Request Information:  client\_address=10.40.0.0  method=GET  real path=/  query=  request\_version=1.1  request\_uri=http://192.168.25.30:8080/    Request Headers:  accept=\*/\*  host=192.168.25.30:30000  user-agent=curl/7.29.0    Request Body:  -no body in request- |

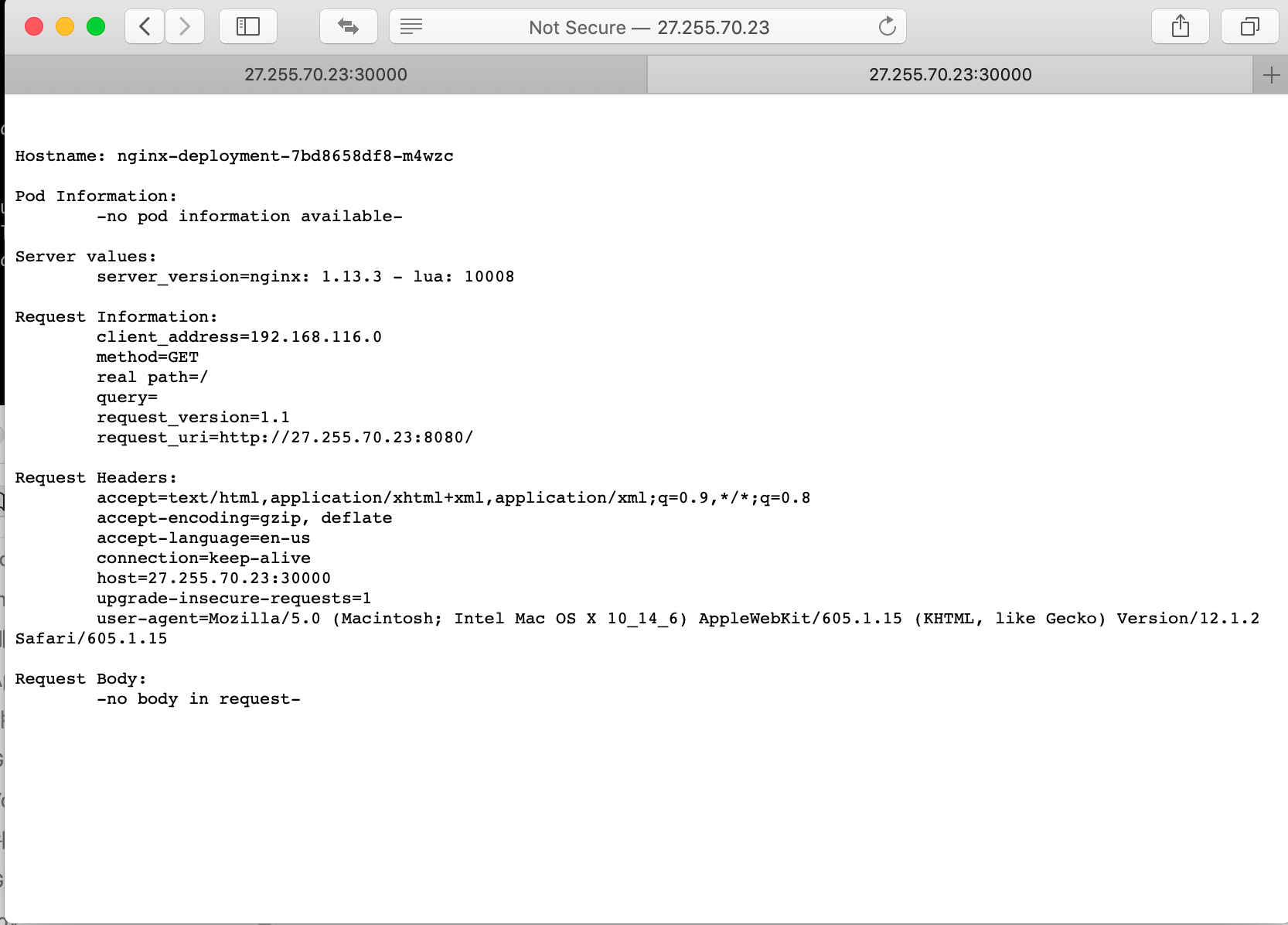
* curl network traffic

|  |
| --- |
| * vboxnet 라우터가 192.168.50.12:30000 (노드 1 포트 NodePort) 액세스 요청을 수신합니다. * 포트 30000에서 Kubernetes 노드 1의 외부 IP로 라우팅됩니다. * 요청이 Kubernetes 프록시로 전송됩니다 * Kubernetes 프록시는 nginx-services-np (ngingx service NodePort)에 할당 된 클러스터 IP 10.99.33.14 및 포트 8082로 요청을 보냅니다. * ngingx 서비스 NodePort는 nginx 애플리케이션에 대해 사용 가능한 포드 중 하나로 요청을 전송합니다. 동일한 Kubernetes 노드 또는 다른 노드 (K8N-N-2와 같은)에있는 포드 일 수 있습니다. |

* 외부와 연결하기

|  |
| --- |
| [root@idc01 ansible-vbox-vagrant-kubernetes]# vagrant suspend  ==> k8s-m-1: Saving VM state and suspending execution...  ==> k8s-n-1: Saving VM state and suspending execution...  ==> k8s-n-2: Saving VM state and suspending execution...    [root@idc01 ansible-vbox-vagrant-kubernetes]# vboxmanage modifyvm ansible-vbox-vagrant-kubernetes\_k8s-m-1\_1567750791434\_56475 --natpf1 "nginx,tcp,,30000,,30000"  [root@idc01 ansible-vbox-vagrant-kubernetes]# vboxmanage modifyvm ansible-vbox-vagrant-kubernetes\_k8s-n-1\_1567751192001\_63965 --natpf1 "nginx,tcp,,30000,,30000"  [root@idc01 ansible-vbox-vagrant-kubernetes]# vboxmanage modifyvm ansible-vbox-vagrant-kubernetes\_k8s-n-2\_1567751502461\_79428 --natpf1 "nginx,tcp,,30000,,30000"    [root@idc01 lab\_itworld]# VBoxManage showvminfo "ansible-vbox-vagrant-kubernetes\_k8s-m-1\_1567750791434\_56475" | grep NIC    NIC 1 Rule(0): name = nginx, protocol = tcp, host ip = , host port = 30000, guest ip = , guest port = 30000  NIC 1 Rule(1): name = ssh, protocol = tcp, host ip = 127.0.0.1, host port = 2222, guest ip = , guest port = 22    [root@idc01 ansible-vbox-vagrant-kubernetes]# vagrant resume  ==> k8s-m-1: Resuming suspended VM...  ==> k8s-m-1: Booting VM...  ==> k8s-m-1: Waiting for machine to boot. This may take a few minutes...  k8s-m-1: SSH address: 127.0.0.1:2222  k8s-m-1: SSH username: vagrant  k8s-m-1: SSH auth method: private key  ==> k8s-m-1: Machine booted and ready!  ==> k8s-m-1: Machine already provisioned. Run `vagrant provision` or use the `--provision`  ==> k8s-m-1: flag to force provisioning. Provisioners marked to run always will still run.  ==> k8s-n-1: Resuming suspended VM...  ==> k8s-n-1: Booting VM...  ==> k8s-n-1: Waiting for machine to boot. This may take a few minutes...  k8s-n-1: SSH address: 127.0.0.1:2200  k8s-n-1: SSH username: vagrant  k8s-n-1: SSH auth method: private key  ==> k8s-n-1: Machine booted and ready!  ==> k8s-n-1: Machine already provisioned. Run `vagrant provision` or use the `--provision`  ==> k8s-n-1: flag to force provisioning. Provisioners marked to run always will still run.  ==> k8s-n-2: Resuming suspended VM...  ==> k8s-n-2: Booting VM...  ==> k8s-n-2: Waiting for machine to boot. This may take a few minutes...  k8s-n-2: SSH address: 127.0.0.1:2201  k8s-n-2: SSH username: vagrant  k8s-n-2: SSH auth method: private key  ==> k8s-n-2: Machine booted and ready!  ==> k8s-n-2: Machine already provisioned. Run `vagrant provision` or use the `--provision`  ==> k8s-n-2: flag to force provisioning. Provisioners marked to run always will still run. |

* 화면에서 check



* auto-healing 기능 / node down / app down

|  |
| --- |
| [root@idc01 ansible-vbox-vagrant-kubernetes]# vagrant status  Current machine states:    k8s-m-1 running (virtualbox)  k8s-n-1 running (virtualbox)  k8s-n-2 running (virtualbox)    [root@idc01 ansible-vbox-vagrant-kubernetes]# kubectl describe pod/nginx-deployment-7bd8658df8-m4wzc  Node: k8s-n-1/192.168.50.12    [root@idc01 ansible-vbox-vagrant-kubernetes]# vagrant suspend k8s-n-1  ==> k8s-n-1: Saving VM state and suspending execution...    [root@idc01 ansible-vbox-vagrant-kubernetes]# vagrant status  Current machine states:    k8s-m-1 running (virtualbox)  k8s-n-1 saved (virtualbox)  k8s-n-2 running (virtualbox)    [root@idc01 ansible-vbox-vagrant-kubernetes]# kubectl get all  NAME READY STATUS RESTARTS AGE  pod/nginx-deployment-7bd8658df8-d2jvp 1/1 Running 4 13h  pod/nginx-deployment-7bd8658df8-gp97s 1/1 Running 0 11m  pod/nginx-deployment-7bd8658df8-m4wzc 1/1 Running 1 75m      NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 29h  service/nginx-service-np NodePort 10.99.33.14 <none> 8082:30000/TCP 13h      NAME READY UP-TO-DATE AVAILABLE AGE  deployment.apps/nginx-deployment 3/3 3 3 13h    NAME DESIRED CURRENT READY AGE  replicaset.apps/nginx-deployment-7bd8658df8 3 3 3 13h    [root@idc01 ansible-vbox-vagrant-kubernetes]# kubectl describe pod/nginx-deployment-7bd8658df8-gp97s  Events:  Type Reason Age From Message  ---- ------ ---- ---- -------  Normal Scheduled 13m default-scheduler Successfully assigned default/nginx-deployment-7bd8658df8-gp97s to k8s-n-2  Normal Pulled 13m kubelet, k8s-n-2 Container image "gcr.io/google\_containers/echoserver:1.8" already present on machine  Normal Created 13m kubelet, k8s-n-2 Created container my-echo  Normal Started 13m kubelet, k8s-n-2 Started container my-echo |

|  |
| --- |
| listen nginx  bind 192.168.25.30:30000  mode tcp  balance roundrobin  option tcp-check  server worker-1 192.168.25.21:30000 check fall 3 rise 2  server worker-2 192.168.25.22:30000 check fall 3 rise 2 |

|  |
| --- |
| **License Warning**  You are using an UNLICENSED copy of **Scroll Word Exporter**.  Do you find Scroll Word Exporter useful? Consider purchasing it today: [https://www.k15t.com/software/scroll-word-exporter](https://www.k15t.com/software/scroll-word-exporter?utm_campaign=Scroll%20Word%20Exporter&utm_medium=Exported%20Document&utm_source=Word%20Evaluation) |