K8s Cluster 및 자동화 운영 구성

목차

[1 서비스 구성도 4](#_Toc24715563)

[1.1 Kubernetes 클러스터 구성도 (L3 Network, Flannel 적용) 4](#_Toc24715564)

[2 서버 구성 정보 5](#_Toc24715565)

[2.1 서버 구성 정보 5](#_Toc24715566)

[3 OS 설치 및 환경 구성 (All Node without HA Node) 5](#_Toc24715567)

[3.1 CENTOS 7.7 설치 minimum 5](#_Toc24715568)

[3.2 OS 환경 - 불필요한 서비스 disable 5](#_Toc24715569)

[3.3 Selinux Enforcing 서비스 6](#_Toc24715570)

[3.4 NTP 설치 및 설정 6](#_Toc24715571)

[3.5 Hostname 수정 (처음 설치 할 때 안했으면 모든 호스트 호스트네임 올바르게 수정) 6](#_Toc24715572)

[3.6 Hosts 설정 7](#_Toc24715573)

[3.7 Yum cache update (Master01 만 해도 된다.) 7](#_Toc24715574)

[3.8 OS Version 확인 7](#_Toc24715575)

[3.9 Kernel Version 확인 7](#_Toc24715576)

[3.10 Package 설치 전 개수 확인 7](#_Toc24715577)

[3.11 Networking 설정 7](#_Toc24715578)

[3.12 날짜 확인 및 변경 8](#_Toc24715579)

[3.13 파일시스템 정보 및 생성 8](#_Toc24715580)

[3.14 Swap 파일 시스템 Disable (Only Master01~03, Worker01~02) 8](#_Toc24715581)

[3.15 파일 복사 간편을 위한 준비 (Only Kubemaster01) 8](#_Toc24715582)

[4 HAProxy 서버 설치 9](#_Toc24715583)

[4.1 HAProxy yum install & Configuration (Only Kubelb) 9](#_Toc24715584)

[5 Kubernetes 설치 (All Node without HA Node) 11](#_Toc24715585)

[5.1 K8s Bridge Network sysctl 설정 11](#_Toc24715586)

[5.2 Docker 설치 11](#_Toc24715587)

[5.3 Kebernetes Repo 등록 및 설치 12](#_Toc24715588)

[5.4 Kubernetes 및 관련 툴 설치 12](#_Toc24715589)

[5.5 환경 변수 설정 12](#_Toc24715590)

[5.6 Kubeadm을 통한 초기화 (반드시 순서를 지켜서 할 것 kubemaster01~03) 13](#_Toc24715591)

[5.7 Kubernetes 관련 툴 설치 (On Worker01, Worker02) 16](#_Toc24715592)

[5.8 K8s Worker 노드 설정 후 각각 추가 (On Worker01, Worker02) 16](#_Toc24715593)

[5.9 Kubeadm Package 설치 확인 (On Worker01) 16](#_Toc24715594)

[5.10 K8s Master, Worker 노드 상태 확인 (On Master01) 16](#_Toc24715595)

[5.11 Pod 실행 상태 확인 (On Master01) 17](#_Toc24715596)

[6 Kubernetes ETCD 삼중화 구성(구 버전 방식으로 보임) 17](#_Toc24715597)

[6.1 ETCD 삼중화 설정 파일 구성 (On Master01) 17](#_Toc24715598)

[6.2 인증 기관 생성 (On Master01) 19](#_Toc24715599)

[6.3 각 ETCD 멤버에 대한 인증서 작성 (On Master01) 19](#_Toc24715600)

[6.4 생성된 파일 서버로 복사 (On Master01) 20](#_Toc24715601)

[6.5 정적 PoD 매니페스트 작성 (ALL Master) 21](#_Toc24715602)

[6.6 상태 체크 21](#_Toc24715603)

[7 Docker Docker-Distribution Local Repository 21](#_Toc24715604)

[7.1 Docker-distribution 설치 및 보안 레지스트리 구성 (On KubeLB) 21](#_Toc24715605)

[7.2 익명사용자가 Local Registry 접근을 막기 위한 http 설정 (On KubeLB) 22](#_Toc24715606)

[7.3 방화벽 설정 (On KubeLB) 22](#_Toc24715607)

[7.4 Docker 호스트에 Secured Docker Regitry 등록 (Kubemaster01~03 Kuberworkd01~02) 23](#_Toc24715608)

[7.5 Docker 호스트에 TLS/SSL 인증서 설치 (Kubemaster01~03 Kubeworker01~02) 23](#_Toc24715609)

[7.6 DockerHub 에서 이미지 가져와서 테그 붙이기 (아무 호스트 테스트) 23](#_Toc24715610)

[7.7 Local Registry 로그인 및 푸시 23](#_Toc24715611)

[8 Kubernetes Docker-Distribution Local Repository Create Secret 24](#_Toc24715612)

[8.1 Docker Reposity 간 Secret 생성 방법1 도커 로그인 정보 이용 (On KubeMaster01) 24](#_Toc24715613)

[8.2 Docker Reposity 간 Secret 생성 방법2 명령어 생성 (On KubeMaster01) 24](#_Toc24715614)

[8.3 생성된 Secret 정보 확인(On KubeMaster01) 25](#_Toc24715615)

[8.4 실제 Docker Local Repository 에서 Pulling 하여 PoD 생성 테스트 (On KubeMaster01) 25](#_Toc24715616)

[9 Kubernetes Dashboard 설치 27](#_Toc24715617)

[9.1 Addon 설치 - Dashboard (On Master01) 27](#_Toc24715618)

[9.2 접속 방법1 Proxy 로 접속 (On KubeMaster01) 28](#_Toc24715619)

[9.3 접속 방법2 NordPort 로 접속 (On KubeMaster01) 28](#_Toc24715620)

[9.4 접속 방법3 API 서버로 접속 (On KubeMaster01) 28](#_Toc24715621)

[9.5 Token 생성 (On KubeMaster01) 32](#_Toc24715622)

[10 Kubernetes Monitoring Prometheus 설치 34](#_Toc24715623)

[10.1 Namespaces 및 Role, Configmap 생성 (On Master01) 34](#_Toc24715624)

[10.2 Deployment Prometheus (On Master01) 38](#_Toc24715625)

[11 Prometheus 를 위한 kube-state-metrics 설치 40](#_Toc24715626)

[11.1 Kube-state-mertics 설치 (On KubeMaster01) 40](#_Toc24715627)

[12 Kubernetes Monitoring Grafana 설치 41](#_Toc24715628)

[12.1 Namespaces 및 Role, Configmap 생성 (On Master01) 41](#_Toc24715629)

[13 Horizontal Pod Autoscaler 테스트 44](#_Toc24715630)

[13.1 Metrics-server 설치 및 Autoscaler YAML파일 작성 (On Master01) 44](#_Toc24715631)

[13.2 AutoSclaer 테스트 (On Master01) 46](#_Toc24715632)

제 개정 이력

|  |  |  |  |
| --- | --- | --- | --- |
| 개정 번호 | 개정 페이지 및 내용 | 개정 일자 | 작성자 |
| 1.0.0 | Kubernetes 구축 | 2019/11/04 | 윤종연 |
|  | Docker Local Repository 구성 및 Kubernetes 연결 | 2019/11/12 | 윤종연 |
|  | Kubernetes Dashboard 구성 | 2019/11/13 | 윤종연 |
|  | Prometheus 구성 | 2019/11/14 | 윤종연 |
|  | Grafana with Prometheus 구성 | 2019/11/14 | 윤종연 |
|  | Kubernetes Autoscaler Scale-in/out 테스트 | 2019/11/15 | 윤종연 |
|  | Kubernetes pipe line CI/CD 구성 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 서비스 구성도

## Kubernetes 클러스터 구성도 (L3 Network, Flannel 적용)

안그려 귀찮네

# 서버 구성 정보

## 서버 구성 정보

|  |  |  |  |
| --- | --- | --- | --- |
| HOSTNAME | VERSION | IP | 용도 |
| Kubelb | OS : Centos 7.7.1908  HAProxy 1.5.18 | 192.168.0.10 | KubeMaster API LB  Docker Repository |
| Kubemaster01 | OS : Centos 7.7.1908  Kubernetes : 1.16.2  Docker : 19.03.4 | 192.168.0.11 | Kubernetes Master |
| Kubemaster02 | OS : Centos 7.7.1908  Kubernetes : 1.16.2  Docker : 19.03.4 | 192.168.0.12 | Kubernetes Master |
| Kubemaster03 | OS : Centos 7.7.1908  Kubernetes : 1.16.2  Docker : 19.03.4 | 192.168.0.13 | Kubernetes Master |
| Keberworker01 | OS : Centos 7.7.1908  Kubernetes : 1.16.2  Docker : 19.03.4 | 192.168.0.14 | Kubernetes Worker |
| Keberworker02 | OS : Centos 7.7.1908  Kubernetes : 1.16.2  Docker : 19.03.4 | 192.168.0.15 | Kubernetes Worker |

# OS 설치 및 환경 구성 (All Node without HA Node)

## CENTOS 7.7 설치 minimum

OS 설치 완료 후 추가적으로 아래 패키지 설치

|  |
| --- |
| yum -y install net-tools  yum -y groupinstall "Development Tools"  yum -y groupinstall "Compatibility Libraries" |

## OS 환경 - 불필요한 서비스 disable

|  |
| --- |
| systemctl disable postfix.service  systemctl disable cups.service  systemctl disable NetworkManager.service  systemctl disable bluetooth.service  systemctl disable ModemManager.service  systemctl disable firewalld.service  systemctl disable dnsmasq.service  systemctl disable iptables.service  systemctl disable ip6tables.service |

## Selinux Enforcing 서비스

|  |
| --- |
| # vi /etc/selinux/config  SELINUX=disabled  # sed -i 's/^SELINUX=enforcing$/SELINUX=disabled/' /etc/selinux/config  # cat /etc/selinux/config  # setenforce 0 |

## NTP 설치 및 설정

|  |
| --- |
| # yum -y install chrony  # vi /etc/chrony.conf  server time.bora.net iburst  # systemctl enable chronyd && systemctl restart chronyd  # chronyc -a makestep  200 OK  # ^\*로 안나오고 ^? 로 나올시 재부팅 필요.  # chronyc sources  210 Number of sources = 1  MS Name/IP address Stratum Poll Reach LastRx Last sample  ===============================================================================  ^\* time.bora.net 3 6 77 23 -36us[ -75us] +/- 31ms  # chronyc tracking  Reference ID : CBF8F08C (time.bora.net)  Stratum : 4  Ref time (UTC) : Tue Jun 18 00:35:17 2019  System time : 0.000000154 seconds slow of NTP time  Last offset : +0.000171008 seconds  RMS offset : 0.000124821 seconds  Frequency : 8.274 ppm fast  Residual freq : +0.070 ppm  Skew : 2.778 ppm  Root delay : 0.016457681 seconds  Root dispersion : 0.022239704 seconds  Update interval : 65.1 seconds  Leap status : Normal |

## Hostname 수정 (처음 설치 할 때 안했으면 모든 호스트 호스트네임 올바르게 수정)

|  |
| --- |
| [root@kubemaster01 ~]# vi /etc/hostname  Kubemaster01  [root@kubemaster01 ~]# hostname kubemaster01 |

## Hosts 설정

|  |
| --- |
| # vi /etc/hosts  127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  192.168.0.10 kubelb  192.168.0.11 kubemaster01  192.168.0.12 kubemaster02  192.168.0.13 kubemaster03  192.168.0.14 kubeworker01  192.168.0.14 kubeworker02  127.0.0.1 부분을 지우면 안됨 |

## Yum cache update (Master01 만 해도 된다.)

|  |
| --- |
| # vi /etc/yum.conf  keepcache=1 (default : 0)  # sed -i 's/^keepcache=0$/keepcache=1/' /etc/yum.conf |

## OS Version 확인

|  |
| --- |
| # cat /etc/redhat-release  CentOS Linux release 7.7.1908 (Core) |

## Kernel Version 확인

|  |
| --- |
| # uname -a  Linux kubemaster01 3.10.0-1062.el7.x86\_64 #1 SMP Wed Aug 7 18:08:02 UTC 2019 x86\_64 x86\_64 x86\_64 GNU/Linux |

## Package 설치 전 개수 확인

|  |
| --- |
| # rpm -qa|wc  425 425 13597 |

## Networking 설정

|  |
| --- |
| # ifconfig  eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  inet 192.168.0.11 netmask 255.255.255.0 broadcast 192.168.0.255  inet6 fe80::cfe1:b863:1938:e5ff prefixlen 64 scopeid 0x20<link>  ether 00:15:5d:00:02:01 txqueuelen 1000 (Ethernet)  RX packets 4268 bytes 3751363 (3.5 MiB)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 1906 bytes 177983 (173.8 KiB)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  inet 127.0.0.1 netmask 255.0.0.0  inet6 ::1 prefixlen 128 scopeid 0x10<host>  loop txqueuelen 1000 (Local Loopback)  RX packets 0 bytes 0 (0.0 B)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 0 bytes 0 (0.0 B)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 |

## 날짜 확인 및 변경

|  |
| --- |
| # ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime  # date  Mon Nov 4 13:29:42 KST 2019 |

## 파일시스템 정보 및 생성

|  |
| --- |
| # df -h  Filesystem Size Used Avail Use% Mounted on  devtmpfs 876M 0 876M 0% /dev  tmpfs 887M 0 887M 0% /dev/shm  tmpfs 887M 8.4M 879M 1% /run  tmpfs 887M 0 887M 0% /sys/fs/cgroup  /dev/mapper/centos-root 27G 1.6G 26G 6% /  /dev/sda2 1014M 141M 874M 14% /boot  /dev/sda1 200M 12M 189M 6% /boot/efi  tmpfs 178M 0 178M 0% /run/user/0 |

## Swap 파일 시스템 Disable (Only Master01~03, Worker01~02)

|  |
| --- |
| # swapoff -a  # vi /etc/fstab  Swap 파티션 # 주석 처리 |

## 파일 복사 간편을 위한 준비 (Only Kubemaster01)

|  |
| --- |
| # root 계정사용  # ssh-keygen -t rsa  인증서가 제대로 만들어 졌다면, ${HOME}/.ssh/ 위치에 id\_rsa 와 id\_rsa.pub 두개의 키가 생성되어있다  # ssh-copy-id kubemaster02  # ssh-copy-id kubemaster03  잘되었다면  # ssh -A kubemaster02  # ssh -A kubemaster03  하여 패스워드없이 들어가는 것 확인 |

# HAProxy 서버 설치

## HAProxy yum install & Configuration (Only Kubelb)

|  |
| --- |
| # HAProxy 설치  [root@kubelb ~]# yum -y install haproxy  # Disable SELINUX  [root@kubelb ~]# sed -i 's/^SELINUX=enforcing$/SELINUX=disabled/' /etc/selinux/config  [root@kubelb ~]# setenforce 0  # Configu HAProxy.cnf  [root@ kubelb HAProxy]# cat /etc/haproxy/haproxy.cfg  #---------------------------------------------------------------------  # Example configuration for a possible web application. See the  # full configuration options online.  #  # http://haproxy.1wt.eu/download/1.4/doc/configuration.txt  #  #---------------------------------------------------------------------  #---------------------------------------------------------------------  # Global settings  #---------------------------------------------------------------------  global  # to have these messages end up in /var/log/haproxy.log you will  # need to:  #  # 1) configure syslog to accept network log events. This is done  # by adding the '-r' option to the SYSLOGD\_OPTIONS in  # /etc/sysconfig/syslog  #  # 2) configure local2 events to go to the /var/log/haproxy.log  # file. A line like the following can be added to  # /etc/sysconfig/syslog  #  # local2.\* /var/log/haproxy.log  #  log 127.0.0.1 local2  chroot /var/lib/haproxy  pidfile /var/run/haproxy.pid  maxconn 4000  user haproxy  group haproxy  daemon  # turn on stats unix socket  stats socket /var/lib/haproxy/stats  #---------------------------------------------------------------------  # common defaults that all the 'listen' and 'backend' sections will  # use if not designated in their block  #---------------------------------------------------------------------  defaults  mode tcp  timeout connect 5000ms  timeout client 50000ms  timeout server 50000ms  #---------------------------------------------------------------------  # main frontend which proxys to the backends  #---------------------------------------------------------------------  frontend k8s-api-front  # bind 192.168.255.37:6443  bind \*:6443  #option tcplog  default\_backend k8s-api-back  #---------------------------------------------------------------------  # static backend for serving up images, stylesheets and such  #---------------------------------------------------------------------  #---------------------------------------------------------------------  # round robin balancing between the various backends  #---------------------------------------------------------------------  backend k8s-api-back  #mode tcp  option tcplog  option tcp-check  balance roundrobin  default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100  server kubemaster01 192.168.0.11:6443 check  server kubemaster02 192.168.0.12:6443 check  server kubemaster03 192.168.0.13:6443 check  # HAProxy Restart  [root@kubelb HAProxy]# systemctl enable haproxy  [root@kubelb HAProxy]# systemctl restart haproxy  # 확인  [root@kubelb HAProxy]# netstat -anp|grep 6443  tcp 0 0 0.0.0.0:6443 0.0.0.0:\* LISTEN 4350/haproxy |

# Kubernetes 설치 (All Node without HA Node)

## K8s Bridge Network sysctl 설정

|  |
| --- |
| #바꿈  [root@kubemaster01 ~]# vi /etc/sysctl.d/k8s.conf  net.bridge.bridge-nf-call-iptables = 1  net.bridge.bridge-nf-call-ip6tables = 1  [root@kubemaster01 ~]# sysctl --system  #모듈 체크  [root@kubemaster01 ~]# lsmod | grep br\_netfilter  [root@kubemaster01 ~]# modprobe br\_netfilter |

## Docker 설치

|  |
| --- |
| [root@kubemaster01 ~]# yum update -y #필요에 따라하면됨  [root@kubemaster01 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2  [root@kubemaster01 ~]# yum install -y epel-release  [root@kubemaster01 ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo  [root@kubemaster01 ~]# yum list docker-ce --showduplicates | sort -r #버전확인 3:19.03.4-3.el7일경우  [root@kubemaster01 ~]# yum -y install docker-ce-19.03.4 docker-ce-cli-19.03.4 containerd.io  [root@kubemaster01 ~]# systemctl enable docker  [root@kubemaster01 ~]# systemctl start docker  #Docker cgroup 확인  [root@kubemaster01 ~]# docker info|grep Cgroup  Cgroup Driver: cgroupfs  #Cgroup 변경 #이작업을 반드시 해야되는지 검색을 해보고 하길 바람  [root@kubemaster01 ~]# vi /etc/docker/daemon.json  {  "exec-opts": ["native.cgroupdriver=systemd"],  "log-driver": "json-file",  "log-opts": {  "max-size": "100m"  },  "storage-driver": "overlay2",  "storage-opts": [  "overlay2.override\_kernel\_check=true"  ]  }  #변경 적용  [root@kubemaster01 ~]# systemctl daemon-reload  [root@kubemaster01 ~]# systemctl restart docker  #변경 확인  [root@kubemaster01 ~]# docker info|grep Cgroup  Cgroup Driver: systemd  #문제 있을경우 삭제  [root@kubemaster01 ~]# yum remove docker-ce  [root@kubemaster01 ~]# rm -rf /var/lib/docker |

## Kebernetes Repo 등록 및 설치

|  |
| --- |
| [root@kubemaster01 ~]# vi /etc/yum.repos.d/kubernetes.repo  [kubernetes]  name=Kubernetes  baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64  enabled=1  gpgcheck=1  repo\_gpgcheck=1  gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg  exclude=kube\* |

## Kubernetes 및 관련 툴 설치

|  |
| --- |
| [root@kubemaster01 ~]# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes  [root@kubemaster01 ~]# systemctl enable kubelet && systemctl start kubelet |

## 환경 변수 설정

|  |
| --- |
| # kubernetes version 확인  # kubelet --version  Kubernetes v1.16.2  # kubeadm version  kubeadm version: &version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2", GitCommit:"c97fe5036ef3df2967d086711e6c0c405941e14b", GitTreeState:"clean", BuildDate:"2019-10-15T19:15:39Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}  # 참고한것 기본 설정인듯하다 이렇게 하니 성공 1.15버전 이상  [root@kubemaster01 ~]# vi ./kubeadm-config.yaml  apiVersion: kubeadm.k8s.io/v1beta1  kind: ClusterConfiguration  kubernetesVersion: 1.16.2  controlPlaneEndpoint: "192.168.0.10:6443"  networking:  podSubnet: 10.244.0.0/16 |

## Kubeadm을 통한 초기화 (반드시 순서를 지켜서 할 것 kubemaster01~03)

|  |
| --- |
| # 이것이 1.11 버전 1.15 이상은 적용하지않음  #환경변수 작성  export KUBERNETES\_VER=v1.16.2  export LOAD\_BALANCER\_DNS=192.168.255.34  export LOAD\_BALANCER\_PORT=6443  export CP1\_HOSTNAME=kubemaster01  export CP1\_IP=192.168.255.38  export CP2\_HOSTNAME=kubemaster02  export CP2\_IP=192.168.255.39  export CP3\_HOSTNAME=kubemaster03  export CP3\_IP=192.168.255.40  [root@kubemaster01 ~]# vi ./kubeadm-config.yaml  apiVersion: kubeadm.k8s.io/v1alpha2  kind: MasterConfiguration  kubernetesVersion: ${KUBERNETES\_VER}  apiServerCertSANs:  - "${LOAD\_BALANCER\_DNS}"  api:  controlPlaneEndpoint: "$LOAD\_BALANCER\_DNS:$LOAD\_BALANCER\_PORT"  etcd:  local:  extraArgs:  listen-client-urls: "https://127.0.0.1:2379,https://${CP1\_IP}:2379"  advertise-client-urls: "https://${CP1\_IP}:2379"  listen-peer-urls: "https://${CP1\_IP}:2380"  initial-advertise-peer-urls: "https://${CP1\_IP}:2380"  initial-cluster: "${CP1\_HOSTNAME}=https://${CP1\_IP}:2380"  serverCertSANs:  - $CP1\_HOSTNAME  - $CP1\_IP  peerCertSANs:  - $CP1\_HOSTNAME  - $CP1\_IP  networking:  # This CIDR is a Flannel default. Substitute or remove for your CNI provider.  podSubnet: "10.244.0.0/16"  # ipv4 포워딩 활성화  # echo 1 > /proc/sys/net/ipv4/ip\_forward  # 초기화 실행 1.15 이상 버전 아래와 같이 실행  # kubeadm init --config=./kubeadm-config.yaml --upload-certs  Your Kubernetes control-plane has initialized successfully!  To start using your cluster, you need to run the following as a regular user:  mkdir -p $HOME/.kube  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  sudo chown $(id -u):$(id -g) $HOME/.kube/config  You should now deploy a pod network to the cluster.  Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:  https://kubernetes.io/docs/concepts/cluster-administration/addons/  You can now join any number of the control-plane node running the following command on each as root:  kubeadm join 192.168.0.10:6443 --token rfthql.q908317x2pfz9jn3 \  --discovery-token-ca-cert-hash sha256:0958f00d84b5bed5ca88e71e99dea5a845eba2e88851d5c8000c59c8241b6776 \  --control-plane --certificate-key 5a2ce911805091e624e1df3caca9f752de4824ae6293fa9e9597f8917a3273be  Please note that the certificate-key gives access to cluster sensitive data, keep it secret!  As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use  "kubeadm init phase upload-certs --upload-certs" to reload certs afterward.  Then you can join any number of worker nodes by running the following on each as root:  kubeadm join 192.168.0.10:6443 --token rfthql.q908317x2pfz9jn3 \  --discovery-token-ca-cert-hash sha256:0958f00d84b5bed5ca88e71e99dea5a845eba2e88851d5c8000c59c8241b6776 [root@kubemaster01 ~]#  [root@kubemaster01 ~]# mkdir -p $HOME/.kube  [root@kubemaster01 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  [root@kubemaster01 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config  # CNI 설치  [root@kubemaster01 ~]# wget https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml  [root@kubemaster01 ~]# kubectl apply -f kube-flannel.yml  #노드확인  [root@kubemaster01 ~]# kubectl get nodes  NAME STATUS ROLES AGE VERSION  kubemaster01 Ready master 8m21s v1.16.2  [root@kubemaster01 ~]#  #kubemaster02 에서 실행  [root@kubemaster02 ~]# kubeadm join 192.168.0.10:6443 --token rfthql.q908317x2pfz9jn3 --discovery-token-ca-cert-hash sha256:0958f00d84b5bed5ca88e71e99dea5a845eba2e88851d5c8000c59c8241b6776 --control-plane --certificate-key 5a2ce911805091e624e1df3caca9f752de4824ae6293fa9e9597f8917a3273be  [root@kubemaster02 ~]# mkdir -p $HOME/.kube  [root@kubemaster02 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  [root@kubemaster02 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config  #kubemaster03 에서 실행  [root@kubemaster03 ~]# kubeadm join 192.168.255.34:6443 --token s00xr5.duti4gieqhqlzyhv --discovery-token-ca-cert-hash sha256:2211f70a4a4b4760418d7db2ad00e2886044cef97b16dd4b1ac14ffaac642df1 --control-plane --certificate-key c900aa808f93be6ba573718f489e9fdd91b372cc4cd982c71509b8e16a9b181d  [root@kubemaster03 ~]# mkdir -p $HOME/.kube  [root@kubemaster03 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  [root@kubemaster03 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config  #완료후 노드확인  [root@kubemaster01 ~]# kubectl get nodes  NAME STATUS ROLES AGE VERSION  kubemaster01 Ready master 5m17s v1.16.2  kubemaster02 Ready master 3m23s v1.16.2  kubemaster03 Ready master 1m10s v1.16.2  [root@kubemaster01 ~]#  #실패시  $ kubeadm reset  $ systemctl stop kubelet  $ systemctl stop docker  $ rm -rf /var/lib/cni/  $ rm -rf /var/lib/kubelet/\*  $ rm -rf /etc/cni/  $ /sbin/ifconfig cni0 down  $ /sbin/ifconfig flannel.1 down  $ /sbin/ifconfig docker0 down  $ /sbin/ip link delete cni0  $ /sbin/ip link delete flannel.1  $ systemctl start docker  $ systemctl start kubelet  다시 재구칙 진행 |

참고)

--pod-network-cidr : Pod Container의 네트워크 사용 대역

--service-cidr : 서버 클러스터 네트워크 입력안하면 10.96.0.0/16 으로 잡히며 dns는 10.96.0.10 으로 기본 셋팅 위같이 셋팅시 10.128.0.10 이 자동으로 dns로 잡힌다. 바꾸려면 10-kubeadm.conf 파일을 통해 바꿔야 하는듯 하다.

--service-dns-domain : 도메인 이름 입력 안하면 cluster.local 로 잡힌다.

--apiserver-advertise-address : 마스터 서버IP

## Kubernetes 관련 툴 설치 (On Worker01, Worker02)

|  |
| --- |
| [root@kubeworker01 ~]# yum install -y kubelet kubeadm kubernetes-cni --disableexcludes=kubernetes |

## K8s Worker 노드 설정 후 각각 추가 (On Worker01, Worker02)

|  |
| --- |
| [root@kubeworker01 ~]# systemctl enable kubelet && systemctl start kubelet  [root@kubeworker01 ~]# kubeadm join 192.168.0.10:6443 --token rfthql.q908317x2pfz9jn3 --discovery-token-ca-cert-hash sha256:0958f00d84b5bed5ca88e71e99dea5a845eba2e88851d5c8000c59c8241b6776 |

참고)

Token 값은 새로운 노드를 추가할 때 재사용 가능하며, 분실시 또는 expire 시 에는 Master에서 kubeadm token list 로 알아낼 수 있다

## Kubeadm Package 설치 확인 (On Worker01)

|  |
| --- |
| [root@kubeworker01 ~]# netstat -nlpte  Active Internet connections (only servers)  Proto Recv-Q Send-Q Local Address Foreign Address State User Inode PID/Program name  tcp 0 0 127.0.0.1:10248 0.0.0.0:\* LISTEN 0 59060 19198/kubelet  tcp 0 0 127.0.0.1:10249 0.0.0.0:\* LISTEN 0 61646 19394/kube-proxy  tcp 0 0 127.0.0.1:44942 0.0.0.0:\* LISTEN 0 59057 19198/kubelet  tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 0 22231 3560/sshd  tcp6 0 0 :::10250 :::\* LISTEN 0 52765 19198/kubelet  tcp6 0 0 :::10256 :::\* LISTEN 0 61642 19394/kube-proxy  tcp6 0 0 :::22 :::\* LISTEN 0 22233 3560/sshd |

## K8s Master, Worker 노드 상태 확인 (On Master01)

|  |
| --- |
| [root@kubemaster01 ~]# kubectl get nodes  NAME STATUS ROLES AGE VERSION  kubemaster01 Ready master 12m v1.16.2  kubemaster02 Ready master 7m52s v1.16.2  kubemaster03 Ready master 6m16s v1.16.2  kubeworker01 Ready <none> 46s v1.16.2  kubeworker02 Ready <none> 36s v1.16.2  [root@kubemaster01 ~]# |

## Pod 실행 상태 확인 (On Master01)

|  |
| --- |
| [root@kubemaster01 ~]# kubectl get pods --all-namespaces  NAMESPACE NAME READY STATUS RESTARTS AGE  kube-system coredns-5644d7b6d9-ktfx4 1/1 Running 0 12m  kube-system coredns-5644d7b6d9-z4955 1/1 Running 0 12m  kube-system etcd-kubemaster01 1/1 Running 0 12m  kube-system etcd-kubemaster02 1/1 Running 0 8m27s  kube-system etcd-kubemaster03 1/1 Running 0 6m51s  kube-system kube-apiserver-kubemaster01 1/1 Running 0 11m  kube-system kube-apiserver-kubemaster02 1/1 Running 0 8m27s  kube-system kube-apiserver-kubemaster03 1/1 Running 0 6m52s  kube-system kube-controller-manager-kubemaster01 1/1 Running 1 12m  kube-system kube-controller-manager-kubemaster02 1/1 Running 0 8m27s  kube-system kube-controller-manager-kubemaster03 1/1 Running 0 6m51s  kube-system kube-flannel-ds-amd64-6ctlg 1/1 Running 0 82s  kube-system kube-flannel-ds-amd64-ch9kq 1/1 Running 1 8m28s  kube-system kube-flannel-ds-amd64-d86m2 1/1 Running 0 11m  kube-system kube-flannel-ds-amd64-zf9v5 1/1 Running 0 6m52s  kube-system kube-proxy-4glbz 1/1 Running 0 8m28s  kube-system kube-proxy-7dzgr 1/1 Running 0 6m52s  kube-system kube-proxy-jtpk2 1/1 Running 0 82s  kube-system kube-proxy-rcdvt 1/1 Running 0 12m  kube-system kube-scheduler-kubemaster01 1/1 Running 1 12m  kube-system kube-scheduler-kubemaster02 1/1 Running 0 8m27s  kube-system kube-scheduler-kubemaster03 1/1 Running 0 6m51s  [root@kubemaster01 ~]# |

# Kubernetes ETCD 삼중화 구성(구 버전 방식으로 보임)

## ETCD 삼중화 설정 파일 구성 (On Master01)

|  |
| --- |
| #서버별 환경 설정을 위한 폴더 만들기  [root@kubemaster01 ~]# export HOST01=192.168.0.11  [root@kubemaster01 ~]# export HOST02=192.168.0.12  [root@kubemaster01 ~]# export HOST03=192.168.0.13  [root@kubemaster01 ~]# mkdir -p /tmp/${HOST01}/ /tmp/${HOST02}/ /tmp/${HOST03}/  [root@kubemaster01 ~]# ll /tmp  total 0  drwxr-xr-x 2 root root 6 Nov 5 10:31 192.168.0.11  drwxr-xr-x 2 root root 6 Nov 5 10:31 192.168.0.12  drwxr-xr-x 2 root root 6 Nov 5 10:31 192.168.0.13  [root@kubemaster01 ~]#  [root@kubemaster01 ~]# ETCDHOSTS=(${HOST01} ${HOST02} ${HOST03})  [root@kubemaster01 ~]# NAMES=("kubemaster01" " kubemaster02" " kubemaster03")  for i in "${!ETCDHOSTS[@]}"; do  HOST=${ETCDHOSTS[$i]}  NAME=${NAMES[$i]}  cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml  apiVersion: "kubeadm.k8s.io/v1beta2"  kind: ClusterConfiguration  etcd:  local:  serverCertSANs:  - "${HOST}"  peerCertSANs:  - "${HOST}"  extraArgs:  initial-cluster: ${NAMES[0]}=https://${ETCDHOSTS[0]}:2380,${NAMES[1]}=https://${ETCDHOSTS[1]}:2380,${NAMES[2]}=https://${ETCDHOSTS[2]}:2380  initial-cluster-state: new  name: ${NAME}  listen-peer-urls: https://${HOST}:2380  listen-client-urls: https://${HOST}:2379, https://127.0.0.1:2379  advertise-client-urls: https://${HOST}:2379  initial-advertise-peer-urls: https://${HOST}:2380  EOF  done  # done 까지 복붙해야 파일작성이 끝남  # 파일 생성 확인  [root@kubemaster01 ~]# ls -al /tmp/192.168.0.\*/kubeadmcfg.yaml  -rw-r--r-- 1 root root 680 Nov 5 11:05 /tmp/192.168.0.11/kubeadmcfg.yaml  -rw-r--r-- 1 root root 681 Nov 5 11:05 /tmp/192.168.0.12/kubeadmcfg.yaml  -rw-r--r-- 1 root root 681 Nov 5 11:05 /tmp/192.168.0.13/kubeadmcfg.yaml  [root@kubemaster01 ~]# cat /tmp/192.168.0.11/kubeadmcfg.yaml  apiVersion: "kubeadm.k8s.io/v1beta2"  kind: ClusterConfiguration  etcd:  local:  serverCertSANs:  - "192.168.0.11"  peerCertSANs:  - "192.168.0.11"  extraArgs:  initial-cluster: kubemaster01=https://192.168.0.11:2380, kubemaster02=https://192.168.0.12:2380, kubemaster03=https://192.168.0.13:2380  initial-cluster-state: new  name: kubemaster01  listen-peer-urls: https://192.168.0.11:2380  listen-client-urls: https://192.168.0.11:2379, https://127.0.0.1:2379  advertise-client-urls: https://192.168.0.11:2379  initial-advertise-peer-urls: https://192.168.0.11:2380  [root@kubemaster01 ~]# cat /tmp/192.168.0.12/kubeadmcfg.yaml  apiVersion: "kubeadm.k8s.io/v1beta2"  kind: ClusterConfiguration  etcd:  local:  serverCertSANs:  - "192.168.0.12"  peerCertSANs:  - "192.168.0.12"  extraArgs:  initial-cluster: kubemaster01=https://192.168.0.11:2380, kubemaster02=https://192.168.0.12:2380, kubemaster03=https://192.168.0.13:2380  initial-cluster-state: new  name: kubemaster02  listen-peer-urls: https://192.168.0.12:2380  listen-client-urls: https://192.168.0.12:2379, https://127.0.0.1:2379  advertise-client-urls: https://192.168.0.12:2379  initial-advertise-peer-urls: https://192.168.0.12:2380  [root@kubemaster01 ~]# cat /tmp/192.168.0.13/kubeadmcfg.yaml  apiVersion: "kubeadm.k8s.io/v1beta2"  kind: ClusterConfiguration  etcd:  local:  serverCertSANs:  - "192.168.0.13"  peerCertSANs:  - "192.168.0.13"  extraArgs:  initial-cluster: kubemaster01=https://192.168.0.11:2380, kubemaster02=https://192.168.0.12:2380, kubemaster03=https://192.168.0.13:2380  initial-cluster-state: new  name: kubemaster03  listen-peer-urls: https://192.168.0.13:2380  listen-client-urls: https://192.168.0.13:2379, https://127.0.0.1:2379  advertise-client-urls: https://192.168.0.13:2379  initial-advertise-peer-urls: https://192.168.0.13:2380  [root@kubemaster01 ~]# |

## 인증 기관 생성 (On Master01)

|  |
| --- |
| [root@kubemaster01 ~]# kubeadm init phase certs etcd-ca  # 아래 두 파일 생성 확인  # /etc/kubernetes/pki/etcd/ca.crt  # /etc/kubernetes/pki/etcd/ca.key |

## 각 ETCD 멤버에 대한 인증서 작성 (On Master01)

|  |
| --- |
| #HOST03 부터  [root@kubemaster01 ~]# find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  [root@kubemaster01 ~]# kubeadm init phase certs etcd-server --config=/tmp/${HOST03}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-peer --config=/tmp/${HOST03}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST03}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST03}/kubeadmcfg.yaml  [root@kubemaster01 ~]# cp -R /etc/kubernetes/pki /tmp/${HOST03}/  # cleanup non-reusable certificates  [root@kubemaster01 ~]# find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST02  [root@kubemaster01 ~]# kubeadm init phase certs etcd-server --config=/tmp/${HOST02}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-peer --config=/tmp/${HOST02}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST02}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST02}/kubeadmcfg.yaml  [root@kubemaster01 ~]# cp -R /etc/kubernetes/pki /tmp/${HOST02}/  # cleanup non-reusable certificates  [root@kubemaster01 ~]# find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST01  [root@kubemaster01 ~]# kubeadm init phase certs etcd-server --config=/tmp/${HOST01}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-peer --config=/tmp/${HOST01}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST01}/kubeadmcfg.yaml  [root@kubemaster01 ~]# kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST01}/kubeadmcfg.yaml  #자기자신 HOST01 이기에 pki 이동 할 필요 없다.  #필요 없는 ca.key 삭제  [root@kubemaster01 ~]# find /tmp/${HOST03} -name ca.key -type f -delete  [root@kubemaster01 ~]# find /tmp/${HOST02} -name ca.key -type f -delete |

## 생성된 파일 서버로 복사 (On Master01)

|  |
| --- |
| #각 환경 변수는 알아서 바꿔서 쓸 수 있다. 특히 USER  [root@kubemaster01 ~]# USER=root  [root@kubemaster01 ~]# scp -r /tmp/${HOST03}/\* ${USER}@${HOST03}:  [root@kubemaster01 ~]# scp -r /tmp/${HOST02}/\* ${USER}@${HOST02}:  #SSH HOST03  [root@kubemaster01 ~]# ssh ${USER}@${HOST03}  #pki 폴더 권한 확인 폴더와 하위 파일들 모두 root:root 것이여야 한다.  만약 root것이 아니라면 chown –R root:root pki 명령으로 변경  [root@kubemaster03 ~]# ll  total 8  -rw-------. 1 root root 1724 Oct 19 20:22 anaconda-ks.cfg  -rw-r--r-- 1 root root 681 Nov 5 11:29 kubeadmcfg.yaml  drwxr-xr-x 3 root root 98 Nov 5 11:29 pki  #인증서 이동  [root@kubemaster03 ~]# mv pki /etc/kubernetes/  #SSH HOST02  [root@kubemaster01 ~]# ssh ${USER}@${HOST02}  #pki 폴더 권한 확인 폴더와 하위 파일들 모두 root:root 것이여야 한다.  만약 root것이 아니라면 chown –R root:root pki 명령으로 변경  [root@kubemaster02 ~]# ll  total 8  -rw-------. 1 root root 1724 Oct 19 20:22 anaconda-ks.cfg  -rw-r--r-- 1 root root 681 Nov 5 11:29 kubeadmcfg.yaml  drwxr-xr-x 3 root root 98 Nov 5 11:29 pki  #인증서 이동  [root@kubemaster02 ~]# mv pki /etc/kubernetes/ |

## 정적 PoD 매니페스트 작성 (ALL Master)

|  |
| --- |
| root@HOST01 # kubeadm init phase etcd local --config=/tmp/${HOST01}/kubeadmcfg.yaml  root@HOST02 # kubeadm init phase etcd local --config=/root/kubeadmcfg.yaml  root@HOST03 # kubeadm init phase etcd local --config=/root/kubeadmcfg.yaml |

## 상태 체크

|  |
| --- |
| # 컴포넌트 상태 체크  # kubectl get componentstatuses  #클러스터 정보  # ubectl cluster-info  # ETCD HealthCheck  # docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes k8s.gcr.io/etcd:3.3.15-0 etcdctl --cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc/kubernetes/pki/etcd/peer.key --ca-file /etc/kubernetes/pki/etcd/ca.crt --endpoints https://192.168.0.11:2379 cluster-health |

# Docker Docker-Distribution Local Repository

## Docker-distribution 설치 및 보안 레지스트리 구성 (On KubeLB)

|  |
| --- |
| # Docker-distribution 설치  [root@kubelb ~]# yum install -y docker-distribution  # Docker-distribution 보안 레지스트리 구성 SSL/TLS 인증서 작성  [root@kubelb ~]# openssl req -newkey rsa:2048 -nodes -sha256 -x509 -days 365 -keyout /etc/pki/tls/private/registry.key -out /etc/pki/tls/registry.crt  Country Name (2 letter code) [XX]:  State or Province Name (full name) []:  Locality Name (eg, city) [Default City]:  Organization Name (eg, company) [Default Company Ltd]:  Organizational Unit Name (eg, section) []:  Common Name (eg, your name or your server's hostname) []:kubelb [여긴 반드시 입력해야되며 IP로 하면 안된다]  Email Address []: Email Address []:  [root@kubelb ~]# |

## 익명사용자가 Local Registry 접근을 막기 위한 http 설정 (On KubeLB)

|  |
| --- |
| # Http-tools 설치  [root@kubelb ~]# yum install -y httpd-tools  #접근 계정 생성  [root@kubelb ~]# htpasswd -c -B /etc/docker-distribution/dockerpasswd admin  New password:  Re-type new password:  Adding password for user admin  [root@kubelb ~]#  #Docker Registry 구성 파일 편집  [root@kubelb ~]# vi /etc/docker-distribution/registry/config.yml  version: 0.1  log:  fields:  service: registry  storage:  cache:  layerinfo: inmemory  filesystem:  rootdirectory: /var/lib/registry  http:  addr: kubelb:5000  tls:  certificate: /etc/pki/tls/registry.crt  key: /etc/pki/tls/private/registry.key  auth:  htpasswd:  realm: example.com  path: /etc/docker-distribution/dockerpasswd  # 서비스 활성화 및 기동  [root@kubelb ~]# systemctl enable docker-distribution  [root@kubelb ~]# systemctl start docker-distribution |

## 방화벽 설정 (On KubeLB)

|  |
| --- |
| # Iptables 나 Firewalld 를 사용안하면 딱히 하지 안아도 된다.  [root@kubelb ~]# firewall-cmd --permanent --add-port=5000/tcp  [root@kubelb ~]# firewall-cmd --reload |

## Docker 호스트에 Secured Docker Regitry 등록 (Kubemaster01~03 Kuberworkd01~02)

|  |
| --- |
| # 위에서 인증서 생성시 common name을 도메인으로 한경우 아래 와같이 hosts에 박아놓은다 그렇지 않으면 패스  # cat >> /etc/hosts << EOF  > 192.168.0.10 kubelb  > EOF |

## Docker 호스트에 TLS/SSL 인증서 설치 (Kubemaster01~03 Kubeworker01~02)

|  |
| --- |
| # 디렉토리 생성후 인증서 복사 인증서가 없으면 이미지 풀링이 안된다.  # mkdir -p /etc/docker/certs.d/kubelb:5000  # scp root@192.168.0.10:/etc/pki/tls/registry.crt /etc/docker/certs.d/kubelb\:5000/  The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.  root@192.168.0.10's password:  registry.crt 100% 1261 1.4MB/s 00:00  # |

## DockerHub 에서 이미지 가져와서 테그 붙이기 (아무 호스트 테스트)

|  |
| --- |
| # 디렉토리 생성후 인증서 복사  # docker pull alpine  Using default tag: latest  latest: Pulling from library/alpine  8e402f1a9c57: Pull complete  Digest: sha256:644fcb1a676b5165371437feaa922943aaf7afcfa8bfee4472f6860aad1ef2a0  Status: Downloaded newer image for alpine:latest  # docker tag alpine kubelb:5000/alpine |

## Local Registry 로그인 및 푸시

|  |
| --- |
| # 이미지 푸시를 위해 로컬 레지스트리 로그인  # docker login kubelb:5000  Username: admin  Password:admin  WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  Configure a credential helper to remove this warning. See  https://docs.docker.com/engine/reference/commandline/login/#credentials-store  Login Succeeded  만약 아래와 같은 에러가 나오면 인증서 관련 문제 이니 인증서 복사가 잘되었나 확인  Error response from daemon: Get https://kubelb:5000/v2/: x509: certificate signed by unknown authority  # docker push kubelb:5000/alpine  The push refers to repository [kubelb:5000/alpine]  bcf2f368fe23: Pushed  latest: digest: sha256:d05ecd4520cab5d9e5d877595fb0532aadcd6c90f4bbc837bc11679f704c4c82 size: 528 |

# Kubernetes Docker-Distribution Local Repository Create Secret

## Docker Reposity 간 Secret 생성 방법1 도커 로그인 정보 이용 (On KubeMaster01)

|  |
| --- |
| #해당 방법은 기존 Docker를 로그인한 정보를 바탕으로 Secret 을 만드는 것이다.  # Kubernetes Secret 생성 .docker 폴더가 없다면 docker login 을 한번도 하지 안았기 때문 이다 수동으로 docker login 하여 로그인을 하면 생성됨  [root@kubemaster01 ~]# kubectl create secret generic regcred --from-file=.dockerconfigjson=/root/.docker/config.json --type=kubernetes.io/dockerconfigjson  secret/regcred created  [root@kubemaster01 ~]#  #config.json 파일에는 아래와 같은 정보가 들어 있다  cat config.json  {  "auths": {  "kubelb:5000": {  "auth": "YWRtaW46YWRtaW4="  }  },  "HttpHeaders": {  "User-Agent": "Docker-Client/19.03.4 (linux)"  }  } |

## Docker Reposity 간 Secret 생성 방법2 명령어 생성 (On KubeMaster01)

|  |
| --- |
| #해당 방법은 명령어를 이용하여 직접 Secret을 만드는 것이다.  #명령어 참고 kubectl create secret docker-registry regcred --docker-server=<your-registry-server> --docker-username=<your-name> --docker-password=<your-pword> --docker-email=<your-email>  #생성 email은 안넣어도 된다.  [root@kubemaster01 ~]# kubectl create secret docker-registry regcred-cli --docker-server=https://kubelb:5000 --docker-username=admin --docker-password=admin  secret/regcred-cli created  [root@kubemaster01 ~]# kubectl get secret  NAME TYPE DATA AGE  default-token-k4r76 kubernetes.io/service-account-token 3 6d23h  **regcred** kubernetes.io/dockerconfigjson 1 14m  **regcred-cli** kubernetes.io/dockerconfigjson 1 6s  [root@kubemaster01 ~]# |

## 생성된 Secret 정보 확인(On KubeMaster01)

|  |
| --- |
| #생성된 정보를 디코딩하여 보여준다.  [root@kubemaster01 ~]# kubectl get secret regcred-cli --output="jsonpath={.data.\.dockerconfigjson}" | base64 --decode  {"auths":{"https://kubelb:5000":{"username":"admin","password":"admin","auth":"YWRtaW46YWRtaW4="}}}  [root@kubemaster01 ~]# |

## 실제 Docker Local Repository 에서 Pulling 하여 PoD 생성 테스트 (On KubeMaster01)

|  |
| --- |
| #pod yaml 파일 작성  [root@kubemaster01 ~]# vi testapachepod.yaml  apiVersion: v1  kind: Pod  metadata:  name: test-apache  spec:  containers:  - name: test-apache  image: kubelb:5000/apacheload:0.1  imagePullSecrets:  - name: **regcred-cli**  #생성  [root@kubemaster01 ~]# kubectl apply -f testapachepod.yaml  pod/private-reg created  #확인  [root@kubemaster01 ~]# kubectl get pods  NAME READY STATUS RESTARTS AGE  test-apache 0/1 ContainerCreating 0 9s  [root@kubemaster01 ~]# kubectl get pods  NAME READY STATUS RESTARTS AGE  test-apache 1/1 Running 0 10s  [root@kubemaster01 ~]# kubectl describe pods test-apache  Name: test-apache  Namespace: default  Priority: 0  Node: kubeworker02/192.168.0.15  Start Time: Wed, 13 Nov 2019 14:27:39 +0900  Labels: <none>  Annotations: kubectl.kubernetes.io/last-applied-configuration:  {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"test-apache","namespace":"default"},"spec":{"containers":[{"image":"k...  Status: Running  IP: 10.244.4.31  IPs:  IP: 10.244.4.31  Containers:  test-apache:  Container ID: docker://64e84179fedcfac6d09968cd3da38e7298ebac68736da739054a4837d7b04b9f  Image: kubelb:5000/apacheload:0.1  Image ID: docker-pullable://kubelb:5000/apacheload@sha256:86bb7e6d0bccc7e618cd82bc5c3a8f7b98e572a4fecb34dfe66a5b6876b32c52  Port: <none>  Host Port: <none>  State: Running  Started: Wed, 13 Nov 2019 14:27:49 +0900  Ready: True  Restart Count: 0  Environment: <none>  Mounts:  /var/run/secrets/kubernetes.io/serviceaccount from default-token-k4r76 (ro)  Conditions:  Type Status  Initialized True  Ready True  ContainersReady True  PodScheduled True  Volumes:  default-token-k4r76:  Type: Secret (a volume populated by a Secret)  SecretName: default-token-k4r76  Optional: false  QoS Class: BestEffort  Node-Selectors: <none>  Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s  node.kubernetes.io/unreachable:NoExecute for 300s  Events:  Type Reason Age From Message  ---- ------ ---- ---- -------  Normal Scheduled <unknown> default-scheduler Successfully assigned default/test-apache to kubeworker02  Normal Pulling 24s kubelet, kubeworker02 Pulling image "kubelb:5000/apacheload:0.1"  Normal Pulled 16s kubelet, kubeworker02 Successfully pulled image "kubelb:5000/apacheload:0.1"  Normal Created 15s kubelet, kubeworker02 Created container test-apache  Normal Started 15s kubelet, kubeworker02 Started container test-apache  [root@kubemaster01 ~]# |

# Kubernetes Dashboard 설치

## Addon 설치 - Dashboard (On Master01)

\* 다음 링크를 통해 진행함

아래 쿠버네티스 공식 홈페이지에서 dashboard deploy 참조

☞ https://kubernetes.io/ko/docs/tasks/access-application-cluster/web-ui-dashboard/

|  |
| --- |
| [root@kubemaster01 ~]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta4/aio/deploy/recommended.yaml  namespace/kubernetes-dashboard created  serviceaccount/kubernetes-dashboard created  service/kubernetes-dashboard created  secret/kubernetes-dashboard-certs created  secret/kubernetes-dashboard-csrf created  secret/kubernetes-dashboard-key-holder created  configmap/kubernetes-dashboard-settings created  role.rbac.authorization.k8s.io/kubernetes-dashboard created  clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created  rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created  deployment.apps/kubernetes-dashboard created  service/dashboard-metrics-scraper created  deployment.apps/dashboard-metrics-scraper created  [root@kubemaster01 ~]#  #NodePort 로 변경  [root@kubemaster01 ~]# kubectl edit service -n kubernetes-dashboard  - apiVersion: v1  kind: Service  metadata:  annotations:  kubectl.kubernetes.io/last-applied-configuration: |  {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"k8s-app":"kubernetes-dashboard"},"name":"kubernetes-dashboard","namespace":"kubernetes-dashboard"},"spec":{"ports":[{"port":443,"targetPort":8443}],"selector":{"k8s-app":"kubernetes-dashboard"}}}  creationTimestamp: "2019-11-07T01:35:50Z"  labels:  k8s-app: kubernetes-dashboard  name: kubernetes-dashboard  namespace: kubernetes-dashboard  resourceVersion: "32631"  selfLink: /api/v1/namespaces/kubernetes-dashboard/services/kubernetes-dashboard  uid: 9a15bde6-2e8f-44ba-8032-87a3198e417f  spec:  clusterIP: 10.111.133.19  ports:  - port: 443  protocol: TCP  targetPort: 8443  selector:  k8s-app: kubernetes-dashboard  sessionAffinity: None  type: NodePort  status:  loadBalancer: {}  kind: List  metadata: {}  저장  service/dashboard-metrics-scraper skipped  service/kubernetes-dashboard edited  [root@kubemaster01 ~]# kubectl get service -n kubernetes-dashboard  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  dashboard-metrics-scraper ClusterIP 10.99.55.42 <none> 8000/TCP 6m31s  kubernetes-dashboard NodePort 10.111.133.19 <none> 443:32349/TCP 6m31s  [root@kubemaster01 ~]#  브라우저 접속  http://192.168.0.11:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/  #대시보드 접근 롤 부여 |

## 접속 방법1 Proxy 로 접속 (On KubeMaster01)

|  |
| --- |
| #대시보드를 연결하는 방법은 3가지가 있지만 프록시와 API서버를 통한 것만 소개한다.  #첫번째 프록시로 포워딩 하도록 설정  [root@kubemaster01 ~]# kubectl proxy --port=8001 --address=192.168.0.11 --accept-hosts='^\*$'  브라우저 접속  http://192.168.0.11:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/ |

## 접속 방법2 NordPort 로 접속 (On KubeMaster01)

|  |
| --- |
| #대시보드를 연결하는 방법은 3가지가 있지만 프록시와 API서버를 통한 것만 소개한다.  #첫번째 프록시로 포워딩 하도록 설정  [root@kubemaster01 ~]# kubectl proxy --port=8001 --address=192.168.0.11 --accept-hosts='^\*$'  브라우저 접속  http://192.168.0.11:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/ |

## 접속 방법3 API 서버로 접속 (On KubeMaster01)

|  |
| --- |
| #API 서버를 통한 로그인  .kube/config 파일의 client certificate 와 client key 를 추출 하여 각각 kubectf.crt 와 kubecfg.key 파일로 생성  [root@kubemaster01 ~]# grep 'client-certificate-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d  -----BEGIN CERTIFICATE-----  MIIC8jCCAdqgAwIBAgIIR9yug+NPjMUwDQYJKoZIhvcNAQELBQAwFTETMBEGA1UE  AxMKa3ViZXJuZXRlczAeFw0xOTExMDYwNTA0MzRaFw0yMDExMDUwNTA0MzVaMDQx  FzAVBgNVBAoTDnN5c3RlbTptYXN0ZXJzMRkwFwYDVQQDExBrdWJlcm5ldGVzLWFk  bWluMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAldz1oA6dniCcav+0  A5sRiqxcNu+iHXVitnkZ/dgmKOyb+3jLvCtdS6y14k3b92MdqgKBenTh+n7icAU2  Ks3GmjRFxF5xp4HdnJRvw4D3BosVlw95gS6HlNK5GRYgKUanf3XH1cSeyvFFQIFY  1uS0ObrYEeRg1tZ3vzQblv0ktiEs0tFA3tZb4++g5IxlOZAgofVMM+4NLW7aInsK  S2iKtDxghc5ubYnjsX2j+9BsBFK+/1Rg16yp0Tv3X9Lbgwr4aBTUzc/AzOL2xrsT  gWrsi1torjqF3vAXedhBOoNNPhyp7iK3LVyH1Dc2cIzk05DAyRDa+3C2UYkiwyxt  VHzUBQIDAQABoycwJTAOBgNVHQ8BAf8EBAMCBaAwEwYDVR0lBAwwCgYIKwYBBQUH  AwIwDQYJKoZIhvcNAQELBQADggEBAJzil8GI5yNqBrJYCaV3BhbxATVy8aHaqJs4  UQQ92QY+kOkDBwlB47x4G3u7fvJbqezIZuW2j30/lUehwW/b8SUWZUdwMSsxGsru  eg3BQGx2mphN+5ANOSdz+Bydt9YrJTzw82qANxpAmh9Pbj8lPcF1N4lxtmPhXy+s  WHZcFW4PKsVyquPJV9cHOyJgzs9Utbg6AacC5Ap7oXV4Koxn2cCE+Uj1VogpsHOz  X1aOeYZ0eo2dDUlAZBlCiqFxcU7cGFR8NzKXSeRnO1CxkO1M5Iza80pOi+vrL2lV  YpGsE5qV/NjtKMzaLFPHqyN2rUWITFnsjwMXP/9K862YXIF2nJc=  -----END CERTIFICATE-----  [root@kubemaster01 ~]# grep 'client-certificate-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.crt  [root@kubemaster01 ~]#  [root@kubemaster01 ~]# grep 'client-key-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d  -----BEGIN RSA PRIVATE KEY-----  MIIEowIBAAKCAQEAldz1oA6dniCcav+0A5sRiqxcNu+iHXVitnkZ/dgmKOyb+3jL  vCtdS6y14k3b92MdqgKBenTh+n7icAU2Ks3GmjRFxF5xp4HdnJRvw4D3BosVlw95  gS6HlNK5GRYgKUanf3XH1cSeyvFFQIFY1uS0ObrYEeRg1tZ3vzQblv0ktiEs0tFA  3tZb4++g5IxlOZAgofVMM+4NLW7aInsKS2iKtDxghc5ubYnjsX2j+9BsBFK+/1Rg  16yp0Tv3X9Lbgwr4aBTUzc/AzOL2xrsTgWrsi1torjqF3vAXedhBOoNNPhyp7iK3  LVyH1Dc2cIzk05DAyRDa+3C2UYkiwyxtVHzUBQIDAQABAoIBAQCKohcO87ZXW2+X  g+gxFbhKglvj7hYYDkk86BZlPxk97nP1U6+U6MCrip/PM5ty5NvCNAM3Fv0dT1eF  rSyIuEcqy4Rmvb+B557shFxHjPv3r55PWd9a44JE6BKW5GLoDBZS0NAGLpd+wWcA  3ych0/vtE61iD9SNqWSpGjkTmiXl3KxAfdzYkGnkg2RzM6iW15olbQq50l0z5RTW  E1N6J6nNUiS7c0gcixt1ChRk2KMQohdIAdCX8GYnm1bZvRYYvf/xhkHuIjnkzpMQ  AaswL8TcTw9TcumbRrZ5wcmv8NgF+7C+qb5Cp+HdKYd5ZoVaAN0ef9zDVDTs1W8U  bv8QGQkBAoGBAMZ42pDl3VitxEaknrI9NE3IFGm9X9Jf/8YuezrLC4q+o1M8DhRz  Ba1Pr59Zt02Icy4S99guq0uSae0uHbVrn3LG9cknTqIQmE63lV2j4bG63B5vhUAc  qZTsgxBXgp4weOC/7XPRGePYbazlfG3fCOuzRvWt8Dy/GWbBGZMf4SkhAoGBAMFN  MaecJ4bitU9CjfThuL4DK6qaTtxiQR/Y+U9DTtZe9e6mifl79ayJ3MY4Ft9+el4M  gf6KJG6r2949nZqtsaGebShczWFbAlcxl596Ss6DOUoDyOc2vfxOCw1+c51u3MVp  lZgHzrghInRO2ZN0op/S49E03xo/FKLb2anX8lplAoGAJHPsCPL/LZt00ljIAQq2  wZZ/EVDD+brCGp9M6OaELy6GhvIfz+e0Yjmhkw4CWc43AX5gYSdpNM8mf3aZ5Elp  FsauPAzpJ2EO6F5Vo8ZG3MLqTPjEq5rYFs8931GDXxSeDwGRVlVQgQWuVzQrH274  j3n1DYit+CpNOw86aEhRraECgYAgKtJqjAff4HY/nFm3OKDfrfNkTo11Czy/KgJC  +HZ0IhVXA4vOYpx9e+uraMYVpxyB7h7r1t09NYg/MY26odD/jjeoEdlzrAt1qyEK  PvWX7CpqcLrMD0YKaIpaUoK23i4LBAkK5bG23ny3Ui/O5Af0BMo4lE3UNk8e0usj  4nXIKQKBgG7HIG4qdZp8kuVeyQpfRWyGR7S6ku9OaEnIJnLXJqFe8oloDxoHyiPt  nD+AxQ0q6hvPfdMHmQLwuzaGBQQi1DscTgTkr2y+Jf7IjjT6ikbnArAq4BEICkco  gbNzeTO9H4gosBpAlrbeDHYk9/mtHm4i2i2uAntjJLYwRnfAmggV  -----END RSA PRIVATE KEY-----  [root@kubemaster01 ~]# grep 'client-key-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.key  [root@kubemaster01 ~]#  #생성한 키를 기반으로 p12 인증서 파일 생성  [root@kubemaster01 ~]# openssl pkcs12 -export -clcerts -inkey kubecfg.key -in kubecfg.crt -out kubecfg.p12 -name "kubernetes-admin"  Enter Export Password: admin  Verifying - Enter Export Password: admin  [root@kubemaster01 ~]#  [root@kubemaster01 ~]# ls -al kubec\*  -rw-r--r-- 1 root root 1082 Nov 13 15:16 kubecfg.crt  -rw-r--r-- 1 root root 1675 Nov 13 15:19 kubecfg.key  -rw-r--r-- 1 root root 2454 Nov 13 15:24 kubecfg.p12  [root@kubemaster01 ~]#  #인증서 설치를 위해 두개의 파일을 다운로드 접속할 단말에 다운로드 합니다.  하나는 방금 생성한 kubecfg.p12 다른 하나는 /etc/kubernetes/pki/ca.crt 파일 입니다.    먼저 .ca.crt 를 현재 컴퓨터->신뢰할수 있는 루트 인증기관에 설치 합니다.    완료후 kubecfg.p12 를 현재 사용자->개인용 위치에 설치 합니다.  생성시 입력한 패스워드 입력    개인용에 설치    설치가 다되면 모든 브라우를 종료하고 아래 주소로 접속 합니다.  <https://192.168.0.10:6443/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#!/login>  확인을 눌러 넘깁니다.    정상 페이지 확인 이후 접속을 위한 토큰키 생성을 작업합니다. |

## Token 생성 (On KubeMaster01)

|  |
| --- |
| #서비스 계정 생성  [root@kubemaster01 ~]# cat <<EOF | kubectl create -f -  > apiVersion: v1  > kind: ServiceAccount  > metadata:  > name: admin-user  > namespace: kube-system  > EOF  serviceaccount/admin-user created  [root@kubemaster01 ~]#  #서비스 계정 롤 부여  [root@kubemaster01 ~]# cat <<EOF | kubectl create -f -  > apiVersion: rbac.authorization.k8s.io/v1  > kind: ClusterRoleBinding  > metadata:  > name: admin-user  > roleRef:  > apiGroup: rbac.authorization.k8s.io  > kind: ClusterRole  > name: cluster-admin  > subjects:  > - kind: ServiceAccount  > name: admin-user  > namespace: kube-system  > EOF  clusterrolebinding.rbac.authorization.k8s.io/admin-user created  [root@kubemaster01 ~]#  #토큰 확인  [root@kubemaster01 ~]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')  Name: admin-user-token-j655s  Namespace: kube-system  Labels: <none>  Annotations: kubernetes.io/service-account.name: admin-user  kubernetes.io/service-account.uid: 52d1ac25-bb3e-48bf-b6a4-c8c4ee12ab8d  Type: kubernetes.io/service-account-token  Data  ====  ca.crt: 1025 bytes  namespace: 11 bytes  token: eyJhbGciOiJSUzI1NiIsImtpZCI6IjdvaWJ0bEpSaU1qWUlYaTc2OFZTMTFOUWZPS2twT1FNVEFSZGM4V1o3MmMifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLWo2NTVzIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI1MmQxYWMyNS1iYjNlLTQ4YmYtYjZhNC1jOGM0ZWUxMmFiOGQiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.LmtW2IaHwH5QiVERVbNeXsGJ5YKWgXhOi3B5jaEyvcWnxgyAtvvh3Un4krMPt3shqPb2VPvrOWSh\_91xRJefY\_l7Nxz6wExwBf39wQEvPOwDGcFpRlOcXyNFDmtSbUxqhfz7\_UcDEF0djisnWWGejewK35p57EwGgmdgMJW9\_bwHC9HAQZuCfGVBNym3dXnDDlOMgJN0Bf5cMGLg1gfdLBLTIt2D4AjbJR2a0brf91tlbp5suvSikd5A3MPz5HbfGtbQt6A4WJlmqw\_KExMZypri1S3sOHiFLtqorEGvKZnk\_-n1XjHwpLbWPDFzZCjCMjBDxKXzxWd1UThQrRBY5A  [root@kubemaster01 ~]#  #위 토근 잡을 가지고 대시보드에 접속 한다. |

# Kubernetes Monitoring Prometheus 설치

## Namespaces 및 Role, Configmap 생성 (On Master01)

|  |
| --- |
| #모니터링 네임스페이스 구성 모니터링을 위한 파드들은 moniroing namespaces로 설정 하기로 한다.  [root@kubemaster01 ~]# cat monitoring\_namespaces.yaml  {  "kind": "Namespace",  "apiVersion": "v1",  "metadata": {  "name": "monitoring",  "labels": {  "name": "monitoring"  }  }  }  [root@kubemaster01 ~]# kubectl apply -f monitoring\_namespaces.yaml  namespace/monitoring created  [root@kubemaster01 ~]#  #모니터링을 위한 클러스터 롤 생성 및 바인딩  [root@kubemaster01 ~]# cat monitoring\_cluster\_roles.yaml  apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRole  metadata:  name: prometheus  rules:  - apiGroups: [""]  resources:  - nodes  - nodes/proxy  - services  - endpoints  - pods  verbs: ["get", "list", "watch"]  - apiGroups:  - extensions  resources:  - ingresses  verbs: ["get", "list", "watch"]  - nonResourceURLs: ["/metrics"]  verbs: ["get"]  ---  apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRoleBinding  metadata:  name: prometheus  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: prometheus  subjects:  - kind: ServiceAccount  name: default  namespace: monitoring  [root@kubemaster01 ~]# kubectl apply -f monitoring\_cluster\_roles.yaml  clusterrole.rbac.authorization.k8s.io/prometheus created  clusterrolebinding.rbac.authorization.k8s.io/prometheus created  [root@kubemaster01 ~]#  #Configmap 작성 및 생성  [root@kubemaster01 ~]# cat prometheus\_config\_map.yaml  apiVersion: v1  kind: ConfigMap  metadata:  name: prometheus-server-conf  labels:  name: prometheus-server-conf  namespace: monitoring  data:  prometheus.yml: |-  global:  scrape\_interval: 5s  evaluation\_interval: 5s  scrape\_configs:  - job\_name: 'kubernetes-apiservers'  kubernetes\_sd\_configs:  - role: endpoints  scheme: https  tls\_config:  ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt  bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token  relabel\_configs:  - source\_labels: [\_\_meta\_kubernetes\_namespace, \_\_meta\_kubernetes\_service\_name, \_\_meta\_kubernetes\_endpoint\_port\_name]  action: keep  regex: default;kubernetes;https  - job\_name: 'kubernetes-nodes'  scheme: https  tls\_config:  ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt  bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token  kubernetes\_sd\_configs:  - role: node  relabel\_configs:  - action: labelmap  regex: \_\_meta\_kubernetes\_node\_label\_(.+)  - target\_label: \_\_address\_\_  replacement: kubernetes.default.svc:443  - source\_labels: [\_\_meta\_kubernetes\_node\_name]  regex: (.+)  target\_label: \_\_metrics\_path\_\_  replacement: /api/v1/nodes/${1}/proxy/metrics  - job\_name: 'kubernetes-pods'  kubernetes\_sd\_configs:  - role: pod  relabel\_configs:  - source\_labels: [\_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_scrape]  action: keep  regex: true  - source\_labels: [\_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_path]  action: replace  target\_label: \_\_metrics\_path\_\_  regex: (.+)  - source\_labels: [\_\_address\_\_, \_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_port]  action: replace  regex: ([^:]+)(?::\d+)?;(\d+)  replacement: $1:$2  target\_label: \_\_address\_\_  - action: labelmap  regex: \_\_meta\_kubernetes\_pod\_label\_(.+)  - source\_labels: [\_\_meta\_kubernetes\_namespace]  action: replace  target\_label: kubernetes\_namespace  - source\_labels: [\_\_meta\_kubernetes\_pod\_name]  action: replace  target\_label: kubernetes\_pod\_name  - job\_name: 'kubernetes-cadvisor'  scheme: https  tls\_config:  ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt  bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token  kubernetes\_sd\_configs:  - role: node  relabel\_configs:  - action: labelmap  regex: \_\_meta\_kubernetes\_node\_label\_(.+)  - target\_label: \_\_address\_\_  replacement: kubernetes.default.svc:443  - source\_labels: [\_\_meta\_kubernetes\_node\_name]  regex: (.+)  target\_label: \_\_metrics\_path\_\_  replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor  - job\_name: 'kubernetes-service-endpoints'  kubernetes\_sd\_configs:  - role: endpoints  relabel\_configs:  - source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_scrape]  action: keep  regex: true  - source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_scheme]  action: replace  target\_label: \_\_scheme\_\_  regex: (https?)  - source\_labels: [\_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_path]  action: replace  target\_label: \_\_metrics\_path\_\_  regex: (.+)  - source\_labels: [\_\_address\_\_, \_\_meta\_kubernetes\_service\_annotation\_prometheus\_io\_port]  action: replace  target\_label: \_\_address\_\_  regex: ([^:]+)(?::\d+)?;(\d+)  replacement: $1:$2  - action: labelmap  regex: \_\_meta\_kubernetes\_service\_label\_(.+)  - source\_labels: [\_\_meta\_kubernetes\_namespace]  action: replace  target\_label: kubernetes\_namespace  - source\_labels: [\_\_meta\_kubernetes\_service\_name]  action: replace  target\_label: kubernetes\_name  [root@kubemaster01 ~]# kubectl apply -f prometheus\_config\_map.yaml  configmap/prometheus-server-conf created  [root@kubemaster01 ~]#  kubernetes-apiservers: Kubernetes API에 대한 메트릭을 가져옵니다.  kubernetes-nodes: Kubernetes 노드에서 메트릭을 가져옵니다.  kubernetes-pods: 메타 데이터에 prometheus.io/scrape 및 prometheus.io/port 주석이 정의 된 포드에서 메트릭을 가져옵니다.  kubernetes-cadvisor: Kubernetes 클러스터에서보고 된 cAdvisor 지표를 가져옵니다.  kubernetes-service-endpoints: 메타 데이터에 정의 된 prometheus.io/scrape및 prometheus.io/port주석 이있는 서비스에서 메트릭을 가져 옵니다. |

## Deployment Prometheus (On Master01)

|  |
| --- |
| #Prometheus 용 Service 작성  [root@kubemaster01 ~]# cat service\_prometheus.yaml  apiVersion: v1  kind: Service  metadata:  name: prometheus-service  namespace: monitoring  annotations:  prometheus.io/scrape: 'true'  prometheus.io/port: '9090'  spec:  selector:  app: prometheus-server  type: NodePort  ports:  - port: 8080  targetPort: 9090  nodePort: 30001  [root@kubemaster01 ~]# kubectl apply -f service\_prometheus.yaml  service/prometheus-service created  [root@kubemaster01 ~]#  #Prometheus 용 deployment yaml 파일 작성  [root@kubemaster01 ~]# cat deployment\_Prometheus.yaml  apiVersion: apps/v1  kind: Deployment  metadata:  name: prometheus-deployment  namespace: monitoring  spec:  replicas: 1  selector:  matchLabels:  app: prometheus-server  template:  metadata:  labels:  app: prometheus-server  spec:  containers:  - name: prometheus  image: prom/prometheus:v2.2.1  args:  - "--config.file=/etc/prometheus/prometheus.yml"  - "--storage.tsdb.path=/prometheus/"  ports:  - containerPort: 9090  volumeMounts:  - name: prometheus-config-volume  mountPath: /etc/prometheus/  - name: prometheus-storage-volume  mountPath: /prometheus/  volumes:  - name: prometheus-config-volume  configMap:  defaultMode: 420  name: prometheus-server-conf  - name: prometheus-storage-volume  emptyDir: {}  [root@kubemaster01 ~]# kubectl apply -f deployment\_Prometheus.yaml  deployment.apps/prometheus-deployment created  [root@kubemaster01 ~]#  #설명  prometheus-config-volumeConfigMap을 사용하여 관리 할 예정 prometheus.yml이며, 이는 볼륨 섹션에 반영됩니다.  이것이 prometheus-server-confPrometheus 배포와 함께 ConfigMap 을 사용하는 방법 입니다. 의 경우 prometheus-storage-volume, Prometheus 데이터를 저장하기 위해 emptyDir을 작성하고 있습니다.  이 볼륨은 일시적이며 포드로 생성 및 소멸됩니다. 즉, 어떤 이유로 든 포드를 삭제하면 해당 포드의 데이터 prometheus-storage-volume가 삭제됩니다. 이 데이터를 지속적으로 유지하려면 영구 볼륨을 대신 사용해야합니다.  #접속 해본다  http://192.168.0.11:30001/  #상태 모두 UP상태 확인  http://192.168.0.11:30001/targets |

# Prometheus 를 위한 kube-state-metrics 설치

## Kube-state-mertics 설치 (On KubeMaster01)

|  |
| --- |
| #git 에서 파일 다운  [root@kubemaster01 ~]# git clone https://github.com/devopscube/kube-state-metrics-configs.git  Cloning into 'kube-state-metrics-configs'...  remote: Enumerating objects: 10, done.  remote: Counting objects: 100% (10/10), done.  remote: Compressing objects: 100% (9/9), done.  remote: Total 10 (delta 0), reused 7 (delta 0), pack-reused 0  Unpacking objects: 100% (10/10), done.  [root@kubemaster01 ~]#  #설치  [root@kubemaster01 ~]# kubectl apply -f kube-state-metrics-configs/  clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created  clusterrole.rbac.authorization.k8s.io/kube-state-metrics created  deployment.apps/kube-state-metrics created  serviceaccount/kube-state-metrics created  service/kube-state-metrics created  [root@kubemaster01 ~]#  #확인  [root@kubemaster01 ~]# kubectl get deployments kube-state-metrics -n kube-system  NAME READY UP-TO-DATE AVAILABLE AGE  kube-state-metrics 0/1 1 0 14s  [root@kubemaster01 ~]#  #pod에 대한 그래프가 정상적으로 안나올것이다…위 프로메테우스 셋팅할 때 생성한 Configmap에 구문을 추가한다.  [root@kubemaster01 ~]# kubectl edit configmap prometheus-server-conf -n monitoringtarget\_label: kubernetes\_namespace  - source\_labels: [\_\_meta\_kubernetes\_service\_name]  action: replace  target\_label: kubernetes\_name  **- job\_name: 'kube-state-metrics'**  **static\_configs:**  **- targets: ['kube-state-metrics.kube-system.svc.cluster.local:8080']**  kind: ConfigMap  metadata:  annotations:  kubectl.kubernetes.io/last-applied-configuration: |  #저장하고 나간후 Prometheus pod 가 configmap을 다시 가져올 수 있게 pod를 죽이고 자동으로 올라온걸 확인한다.  #확인  http://192.168.0.11:30001/config 접속 아래와 같은 것이 추가 되었는지 확인  - job\_name: kube-state-metrics  scrape\_interval: 5s  scrape\_timeout: 5s  metrics\_path: /metrics  scheme: http  static\_configs:  - targets:  - kube-state-metrics.kube-system.svc.cluster.local:8080 |

# Kubernetes Monitoring Grafana 설치

## Namespaces 및 Role, Configmap 생성 (On Master01)

|  |
| --- |
| #모니터링 네임스페이스 구성 모니터링을 위한 파드들은 moniroing namespaces로 설정 하기로 한다.  [root@kubemaster01 ~]# cat service\_grafana.yaml  apiVersion: v1  kind: Service  metadata:  labels:  kubernetes.io/name: grafana-app  name: grafana-app  namespace: monitoring  spec:  ports:  - port: 3000  targetPort: 3000  nodePort: 30002  selector:  k8s-app: grafana  type: NodePort  [root@kubemaster01 ~]# kubectl apply -f service\_grafana.yaml  service/grafana-app created  [root@kubemaster01 ~]#  #grafana deployment  [root@kubemaster01 ~]# cat deployment\_grafana.yaml  apiVersion: apps/v1  kind: Deployment  metadata:  name: grafana-app  namespace: monitoring  spec:  replicas: 1  selector:  matchLabels:  k8s-app: grafana  template:  metadata:  labels:  k8s-app: grafana  spec:  containers:  - name: grafana  image: grafana/grafana:5.2.3  ports:  - containerPort: 3000  protocol: TCP  env:  - name: GF\_SERVER\_HTTP\_PORT  value: "3000"  - name: GF\_AUTH\_BASIC\_ENABLED  value: "false"  - name: GF\_AUTH\_ANONYMOUS\_ENABLED  value: "true"  - name: GF\_AUTH\_ANONYMOUS\_ORG\_ROLE  value: Admin  - name: GF\_SERVER\_ROOT\_URL  value: /  [root@kubemaster01 ~]# kubectl apply -f deployment\_grafana.yaml  deployment.apps/grafana-app created  [root@kubemaster01 ~]#  #웹페이지 접속  http://192.168.0.11:30002/  프로메테우스 데이터 소스 추가    #이후 그래프들은 기 만들어진 그래프들을 가져다 쓴다. |

# Horizontal Pod Autoscaler 테스트

## Metrics-server 설치 및 Autoscaler YAML파일 작성 (On Master01)

|  |
| --- |
| #Metrics-server 설치  [root@kubemaster01 ~]# git clone https://github.com/kubernetes-sigs/metrics-server.git  Cloning into 'metrics-server'...  remote: Enumerating objects: 4, done.  remote: Counting objects: 100% (4/4), done.  remote: Compressing objects: 100% (4/4), done.  remote: Total 11349 (delta 0), reused 1 (delta 0), pack-reused 11345  Receiving objects: 100% (11349/11349), 12.18 MiB | 2.79 MiB/s, done.  Resolving deltas: 100% (5912/5912), done.  [root@kubemaster01 ~]# cd /root/metrics-server/deploy/  [root@kubemaster01 deploy]# kubectl apply -f 1.8+/  clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created  clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created  rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created  apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created  serviceaccount/metrics-server created  deployment.apps/metrics-server created  service/metrics-server created  clusterrole.rbac.authorization.k8s.io/system:metrics-server created  clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created  [root@kubemaster01 deploy]#  [root@kubemaster01 ~]# cat autosclaer\_nginx.yaml  apiVersion: autoscaling/v1  kind: HorizontalPodAutoscaler  metadata:  name: kubernetes-autoscaler-nginx  namespace: default  spec:  maxReplicas: 10  minReplicas: 1  scaleTargetRef:  apiVersion: apps/v1beta2  kind: Deployment  name: nginx  targetCPUUtilizationPercentage: 30  [root@kubemaster01 ~]# kubectl apply -f autosclaer\_nginx.yaml  horizontalpodautoscaler.autoscaling/kubernetes-autoscaler-nginx created  [root@kubemaster01 ~]#  #확인 unknown 으로 나올거인데 정상  [root@kubemaster01 ~]# kubectl get HorizontalPodAutoscaler  NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE  kubernetes-autoscaler-nginx Deployment/nginx **<unknown>**/30% 2 10 0 18s  [root@kubemaster01 ~]#  #unkown 해결 방법1  [root@kubemaster01 ~]# kubectl edit deployment metrics-server -n kube-system  아래와 같이 된 것에 추가를 한다.  spec:  containers:  - args:  - --cert-dir=/tmp  - --secure-port=4443  **- --kubelet-insecure-tls**  **- --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname**  image: k8s.gcr.io/metrics-server-amd64:v0.3.6  imagePullPolicy: Always  name: metrics-server  #그래도 해결이 안되면 deploy 한 app의 spec을 정하여 명시해준다.  #unkwown 해결 방법2 deployment에 pod당 cpu와 mem자원을 spec에 명시해야 한다.(스펙은 알아서 예시임)  [root@kubemaster01 ~]# kubectl edit deployment nginx  이렇게 바꿔준다.  spec:  containers:  - image: nginx  imagePullPolicy: Always  name: nginx  **resources:**  **limits:**  **cpu: 500m**  **memory: 1Gi**  **requests:**  **cpu: 200m**  **memory: 256Mi**  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  dnsPolicy: ClusterFirst  3분정도 후 확인해보면 정상 출력되는 것을 확인 할 수 있다.  [root@kubemaster01 ~]# kubectl get hpa  NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE  kubernetes-autoscaler-nginx Deployment/nginx **0%**/30% 1 10 1 5m45s  [root@kubemaster01 ~]#  #API 정상확인  [root@kubemaster01 ~]# kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes  {"kind":"NodeMetricsList","apiVersion":"metrics.k8s.io/v1beta1","metadata":{"selfLink":"/apis/metrics.k8s.io/v1beta1/nodes"},"items":[{"metadata":{"name":"**kubemaster01**","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/kubemaster01","creationTimestamp":"2019-11-15T02:29:31Z"},"timestamp":"2019-11-15T02:29:08Z","window":"30s","usage":{"cpu":"237598966n","memory":"1052500Ki"}},{"metadata":{"name":"kubemaster02","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/**kubemaster02**","creationTimestamp":"2019-11-15T02:29:31Z"},"timestamp":"2019-11-15T02:29:09Z","window":"30s","usage":{"cpu":"233739754n","memory":"845736Ki"}},{"metadata":{"name":"kubemaster03","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/**kubemaster03**","creationTimestamp":"2019-11-15T02:29:31Z"},"timestamp":"2019-11-15T02:29:11Z","window":"30s","usage":{"cpu":"169646338n","memory":"842180Ki"}},{"metadata":{"name":"kubeworker01","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/**kubeworker01**","creationTimestamp":"2019-11-15T02:29:31Z"},"timestamp":"2019-11-15T02:29:02Z","window":"30s","usage":{"cpu":"57385292n","memory":"445984Ki"}},{"metadata":{"name":"kubeworker02","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/**kubeworker02**","creationTimestamp":"2019-11-15T02:29:31Z"},"timestamp":"2019-11-15T02:29:07Z","window":"30s","usage":{"cpu":"98632634n","memory":"712128Ki"}}]}  [root@kubemaster01 ~]# |

## AutoSclaer 테스트 (On Master01)

|  |
| --- |
| #PoD 부하 테스트 떠 있는 어플리케이션의 IP를 확인 합니다.(예제는 nginx 입니다.)  [root@kubemaster01 ~]# kubectl describe pods nginx-786654f4bb-vhggp | grep IP  IP: 10.244.4.47  #해당 IP 80포트로 curl로 niginx 확인  [root@kubemaster01 ~]# curl http://10.244.4.47:80  <!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>  #부하 시작  [root@kubemaster01 ~]# while true; do wget -q -O- http://10.244.4.47:80; done  #잠시후 부하로 인한 0% -> 48% Replicas 1->2 로 변경    # pod 2개로 인한 %감소    #다른 노드에서 추가 부하 발생 시키기 Replicas 2->3 변경    # 30%밑으로 내려가므로 증가 안함 테스트 종료    #부하 중단 후 몇분 이지나면 다시 원래대로 돌아온다 |