

Kubernetes install with kubeadm

Network Example

```
10.10.11.20 kubemaster kubemaster.myhypervisor.ca
10.10.11.30 kube1 kube1.myhypervisor.ca
10.10.11.36 kube2 kube2.myhypervisor.ca
```

Disable SELinux.

```
setenforce 0
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

Enable the `br_netfilter` module for cluster communication.

```
modprobe br_netfilter
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Disable swap to prevent memory allocation issues.

```
swapoff -a
vim /etc/fstab
#Remove swap from fstab
```

Setup NTP

```
yum install -y ntp
systemctl enable ntpd
systemctl start ntpd
```

Install Docker CE.

Install the Docker prerequisites.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

Add the Docker repo and install Docker.

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo  
yum install -y docker-ce
```

Add the Kubernetes repo.

```
cat << EOF | tee /etc/yum.repos.d/kubernetes.repo  
[kubernetes]  
name=Kubernetes  
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64  
enabled=1  
gpgcheck=0  
repo_gpgcheck=0  
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg  
        https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg  
EOF
```

Install Kubernetes.

```
yum install -y kubelet kubeadm kubectl
```

Reboot.

Enable and start Docker and Kubernetes.

```
systemctl enable docker  
systemctl enable kubelet  
systemctl start docker  
systemctl start kubelet
```

Check the group Docker is running in.

```
docker info | grep -i cgroup
```

****Note: Complete the following section on the MASTER ONLY!***

Initialize the cluster using the IP range for Flannel.

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Copy the `kubeadm join` output.

Create standard user

```
useradd kubeuser
usermod -aG wheel kubeuser
passwd kubeuser
su kubeuser
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Deploy Flannel.

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Check the cluster state.

```
kubectl get pods --all-namespaces
```

Note: Complete the following steps on the NODES ONLY!

Run the `join` command that you copied earlier, then check your nodes from the master.

```
kubectl get nodes
```

Create a service account (for kube dashboard)

Copy the token in a safe location, you will be able to use that token for services such as the k8s dashboard

Creating a admin / service account user called k

```
kubectl create serviceaccount k8sadmin -n kube-system
```

Give the user admin privileges

```
kubectl create clusterrolebinding k8sadmin --clusterrole=cluster-admin --serviceaccount=kube-system:k8sadmin
```

Get the token

```
kubectl -n kube-system describe secret $(sudo kubectl -n kube-system get secret | (grep k8sadmin || echo "$_") | awk '{print $1}') | grep token: | awk '{print $2}'
```

Installing MetalLB

```
kubectl apply -f  
https://raw.githubusercontent.com/google/metallb/v0.8.3/manifests/metallb.yaml
```

Create a file called metallb.yml

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  namespace: metallb-system  
  name: config  
data:  
  config: |  
    address-pools:  
    - name: default  
      protocol: layer2  
      addresses:  
      - 10.10.11.150-10.10.11.160
```

```
kubectl create -f metallb.yml
```

Installing Dashboard

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta8/aio/deploy/recommended.yaml
```

Create a file called kube-dashboard-service.yml

```
apiVersion: v1  
kind: Service  
metadata:  
  labels:  
    app.kubernetes.io/name: load-balancer-dashboard
```

```
name: dashboard-service
namespace: kubernetes-dashboard
spec:
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  type: LoadBalancer
```

```
kubectl create -f kube-dashboard-service.yml
kubectl get all -A
```

With the command "kubectl get all -A" you will be able to find the external IP provided by metallb, only https is supported, use the token from the previous step to login to the dashboard.

(Heketi Install guide here:

<https://wiki.mypervisor.ca/books/linux/page/glusterfs-using-ubuntu1604-c74>)

Add Heketi for Dynamic Volumes

Create a file called heketi-storage.yml

```
apiVersion: v1
kind: Secret
metadata:
  name: heketi-secret
  namespace: default
type: "kubernetes.io/glusterfs"
data:
  # base64 encoded password. E.g.: echo -n "password" | base64
  key: cGFzc3dvcmQ=
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gluster-vol-default
provisioner: kubernetes.io/glusterfs
```

```

parameters:
  resturl: "http://10.10.11.161:8080"
  restuser: "admin"
  secretNamespace: "default"
  secretName: "heketi-secret"
  clusterid: "eba4f23d2eb41f894590cbe3ee05e51e"
allowVolumeExpansion: true
---
apiVersion: v1
kind: Endpoints
metadata:
  name: glusterfs-cluster
subsets:
- addresses:
  - ip: 10.10.11.200
  ports:
  - port: 5000
- addresses:
  - ip: 10.10.11.201
  ports:
  - port: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: glusterfs-cluster
spec:
  ports:
  - port: 5000

```

Install Glusterfs-Client, The version needs to be the same as the Glusterfs Server.

```

apt-get install software-properties-common
add-apt-repository ppa:gluster/glusterfs-7
apt install glusterfs-client

```

Add a PVC to the storage (Example at the bottom is for a pihole deployment)

```

---
apiVersion: v1

```

```

kind: PersistentVolumeClaim
metadata:
  name: pihole-dnsmasq-volume
  annotations:
    volume.beta.kubernetes.io/storage-class: gluster-vol-default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pihole-etc-volume
  annotations:
    volume.beta.kubernetes.io/storage-class: gluster-vol-default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi

```

Deploy container

Deploy your application (example below is pihole)

```

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pihole
  labels:
    app: pihole
spec:
  replicas: 1

```

```
selector:
  matchLabels:
    app: pihole
template:
  metadata:
    labels:
      app: pihole
      name: pihole
  spec:
    containers:
      - name: pihole
        image: pihole/pihole:latest
        imagePullPolicy: Always
        env:
          - name: TZ
            value: "America/New_York"
          - name: WEBPASSWORD
            value: "secret"
        volumeMounts:
          - name: pihole-etc-volume
            mountPath: "/etc/pihole"
          - name: pihole-dnsmasq-volume
            mountPath: "/etc/dnsmasq.d"
    volumes:
      - name: pihole-etc-volume
        persistentVolumeClaim:
          claimName: pihole-etc-volume
      - name: pihole-dnsmasq-volume
        persistentVolumeClaim:
          claimName: pihole-dnsmasq-volume
---
apiVersion: v1
kind: Service
metadata:
  name: pihole-udp
spec:
  type: LoadBalancer
  ports:
    - port: 53
```



```
      targetPort: 53
      protocol: UDP
      name: dns-udp
    selector:
      app: pihole
  ---
apiVersion: v1
kind: Service
metadata:
  name: pihole-tcp
spec:
  type: LoadBalancer
  ports:
    - port: 53
      targetPort: 53
      protocol: TCP
      name: dns-tcp
    - protocol: TCP
      name: web
      port: 80
  selector:
    app: pihole
```

Install Helm Chart

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 > get_helm.sh
chmod 700 get_helm.sh
./get_helm.sh

helm repo add stable https://kubernetes-charts.storage.googleapis.com/
helm repo update
```

traefik

```
apiVersion: v1
kind: ServiceAccount
metadata:
```

```
namespace: default
name: traefik-ingress-controller
---
kind: Deployment
apiVersion: apps/v1
metadata:
  namespace: default
  name: traefik
  labels:
    app: traefik
spec:
  replicas: 1
  selector:
    matchLabels:
      app: traefik
  template:
    metadata:
      labels:
        app: traefik
    spec:
      serviceAccountName: traefik-ingress-controller
      containers:
        - name: traefik
          image: traefik:v2.0
          args:
            - --api.insecure
            - --accesslog
            - --entrypoints.web.Address=:80
            - --providers.kubernetescrd
          ports:
            - name: web
              containerPort: 80
            - name: admin
              containerPort: 8080
---
kind: Deployment
apiVersion: apps/v1
metadata:
  namespace: default
  name: whoami
```

```
  labels:
    app: whoami
spec:
  replicas: 2
  selector:
    matchLabels:
      app: whoami
  template:
    metadata:
      labels:
        app: whoami
    spec:
      containers:
        - name: whoami
          image: containous/whoami
          ports:
            - name: web
              containerPort: 80
```

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: ingressroutes.traefik.containo.us
```

```
spec:
  group: traefik.containo.us
  version: v1alpha1
  names:
    kind: IngressRoute
    plural: ingressroutes
    singular: ingressroute
  scope: Namespaced
```

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: ingressroutetcp.traefik.containo.us
```

```
spec:
```

```
group: traefik.containo.us
version: v1alpha1
names:
  kind: IngressRouteTCP
  plural: ingressroutetcps
  singular: ingressroutetcp
scope: Namespaced

---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: middlewares.traefik.containo.us

spec:
  group: traefik.containo.us
  version: v1alpha1
  names:
    kind: Middleware
    plural: middlewares
    singular: middleware
  scope: Namespaced

---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: tlsoptions.traefik.containo.us

spec:
  group: traefik.containo.us
  version: v1alpha1
  names:
    kind: TLSOption
    plural: tlsoptions
    singular: tlsoption
  scope: Namespaced

---
apiVersion: apiextensions.k8s.io/v1beta1
```

```
kind: CustomResourceDefinition
metadata:
  name: traefikservices.traefik.containo.us

spec:
  group: traefik.containo.us
  version: v1alpha1
  names:
    kind: TraefikService
    plural: traefikservices
    singular: traefikservice
    scope: Namespaced

---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: traefik-ingress-controller

rules:
- apiGroups:
  - ""
  resources:
    - services
    - endpoints
    - secrets
  verbs:
    - get
    - list
    - watch
- apiGroups:
  - extensions
  resources:
    - ingresses
  verbs:
    - get
    - list
    - watch
- apiGroups:
  - extensions
```

```
resources:
  - ingresses/status
verbs:
  - update
- apiGroups:
  - traefik.containo.us
resources:
  - middlewares
verbs:
  - get
  - list
  - watch
- apiGroups:
  - traefik.containo.us
resources:
  - ingressroutes
verbs:
  - get
  - list
  - watch
- apiGroups:
  - traefik.containo.us
resources:
  - ingressroutetcp
verbs:
  - get
  - list
  - watch
- apiGroups:
  - traefik.containo.us
resources:
  - tlsoptions
verbs:
  - get
  - list
  - watch
- apiGroups:
  - traefik.containo.us
resources:
  - traefikservices
```

verbs:

- get
- list
- watch

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: simpleingressroute
  namespace: default
spec:
  entryPoints:
    - web
  routes:
    - match: Host(`pihole.myhypervisor.ca`) && PathPrefix(`/`)
      kind: Rule
      services:
        - name: pihole-web
          port: 80
```

Revision #16

Created 10 February 2019 19:24:25 by Dave

Updated 14 February 2020 09:54:02 by Dave