The useful commands for kubespray on Centos

Following basic procedure of kubespray, other commands are good to take notes.

1. Before Install

Read those articles for knowledge updating.

1.1 Reference

- https://ottodeng.io/post/kubespray/
- https://www.jianshu.com/p/45b9707b4567

1.2 Linux command

- Netstat command

Check Pods Routing.

```
$ ipvsadm -L -n

$ netstat -tulpn | grep LISTEN
$ sudo lsof -i -P -n
$ sudo lsof -i -P -n | grep LISTEN
$ doas lsof -i -P -n | grep LISTEN ### [OpenBSD] ###
```

- Find a bigger disk space

```
findmnt -n -o SOURCE --target /opt
```

1.3 Antivirus ClamAV setup

https://hostpresto.com/community/tutorials/how-to-install-clamav-on-centos-7/

```
git clone https://github.com/geerlingguy/ansible-role-clamav

modify the tasks and vars files to support Centos
$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "yum -y update && yum -y
install epel-release && yum -y update && yum clean all && yum -y install clamav-
server clamav-data clamav-update clamav-filesystem clamav clamav-scanner-systemd
clamav-devel clamav-lib clamav-server-systemd"
$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "setsebool -P
antivirus_can_scan_system 1 && setsebool -P clamd_use_jit 1"
$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "cp
/etc/clamd.d/scan.conf /etc/clamd.d/scan.conf.backup && sed -i -e
"s/^Example/#Example/" /etc/clamd.d/scan.conf"
$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "cat /etc/passwd | grep
clam"
$ EDIT  vim /etc/clamd.d/scan.conf,,, default User clamscan
```

Uncomment the line #LocalSocket **/var/run/clamd.scan/clamd.sock** to

```
LocalSocket /var/run/clamd.scan/clamd.sock
```

Freshclam is used to update the database of virus definitions into the server.

```
$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "cp /etc/freshclam.conf
/etc/freshclam.conf.bakup && sed -i -e "s/^Example/#Example/" /etc/freshclam.conf"

$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "freshclam"
```

Create a new file /usr/lib/systemd/system/freshclam.service

```
# Run the freshclam as daemon
[Unit]
Description = freshclam scanner
After = network.target

[Service]
Type = forking
ExecStart = /usr/bin/freshclam -d -c 4
Restart = on-failure
PrivateTmp = true

[Install]
WantedBy=multi-user.target
```

Execute those service command to start

```
systemctl enable freshclam.service
systemctl start freshclam.service

$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "systemctl start
clamd@scan &&  systemctl enable clamd@scan && systemctl status clamd@scan"

$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "systemctl daemon-
reload"

$ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "clamscan --infected --
remove --recursive /home /root"
```

1.4 SSH Tunnel Setup

```
  sudo groupadd devops-group
  sudo useradd -G devops-group devops
  chmod +w /etc/sudoers && echo "devops ALL=(ALL)       NOPASSWD: ALL" >>
/etc/sudoers && chmod -w /etc/sudoers

  su devops && cd ~
  ssh-keygen -t rsa -P ""
  cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
  chmod +700 .ssh
```

```
  chmod 600 .ssh/*rsa
  [devops@Redhat-Ansible ]$ chmod 644 .ssh/authorized_keys
  [devops@Redhat-Ansible ]$ chmod 644 .ssh/id_rsa.pub
  [devops@Redhat-Ansible ]$ chmod 644 .ssh/known_hosts

  [devops@Redhat-Ansible ]$ ssh-copy-id -i ~/.ssh/id_rsa.pub devops@10.0.0.5


  service sshd restart
```

2. Start Install

2.1 Use The standard Procedures

- Prepare the Hosts.yaml

```
  # Install dependencies from ``requirements.txt``
  sudo pip install -r requirements.txt

  # Copy ``inventory/sample`` as ``inventory/mycluster``
  cp -rfp inventory/sample inventory/mycluster

  # Update Ansible inventory file with inventory builder
  declare -a IPS=(10.10.1.3 10.10.1.4 10.10.1.5)
  CONFIG_FILE=inventory/mycluster/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

Ensure Disable Some Settings for K8s cluster

Disable Swapoff

```
  遇到的问题wait for the apiserver to be running
  $ ansible -i inventory/mycluster/hosts.yaml all -m raw -a "swapoff -a && free -
m"
```

Network Settings

```
  ansible -i inventory/mycluster/hosts.yaml all -m raw -a "systemctl stop
firewalld && systemctl disable firewalld"
  ansible -i inventory/mycluster/hosts.yaml all -m raw -a "setenforce 0"
  ansible -i inventory/mycluster/hosts.yaml all -m raw -a "sed -i --follow-
symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux"

  ipv4网络设置
  ansible -i inventory/mycluster/hosts.yaml all -m raw -a "modprobe br_netfilter
&& echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables && sysctl -w
net.ipv4.ip_forward=1"
```

- Clean Install

```
  ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root
cluster.yml
```

- Reset and Uninstall All

  But you can also reset the entire cluster for fresh installation:

```
  $  ansible-playbook -i inventory/devopscluster/hosts.yaml reset.yml

  # Run clean command for files are removing from nodes.
  $ k8s-uninstall.sh
```

Remember to keep the “hosts.ini” updated properly.

- Remove Nodes You can remove node by node from your cluster simply adding specific node do section [kube-node] in inventory/mycluster/hosts.ini file (your hosts file) and run command:

```
  $ ansible-playbook -i inventory/devopscluster/hosts.yaml remove-node.yml
```

Install with Debug use –b (become), -i (inventory) and –v (verbose)

```
 $ ansible-playbook -v -b -i inventory/devopscluster/hosts.ini cluster.yml
```

Install with Azure Automation scritps

```
https://github.com/kubernetes-sigs/kubespray/tree/master/contrib/azurerm

$ ansible-playbook -i contrib/azurerm/inventory -u devops --become -e
"@inventory/sample/group_vars/all.yml" cluster.yml
```

2.2 Use Automation Deployment onto Cloud, e.g. Azure

2.3 Create the PVCs

```
ansible -i inventory/mycluster/hosts.yaml all -m raw -a "yum install -y nfs-utils"
ansible -i inventory/mycluster/hosts.yaml all -m raw -a "chmod -R 755
/var/nfsshare && chown nfsnobody:nfsnobody /var/nfsshare"
ansible -i inventory/mycluster/hosts.yaml all -m raw -a "systemctl enable rpcbind
```

```
&& systemctl enable nfs-server && systemctl enable nfs-lock && systemctl enable
nfs-idmap"
ansible -i inventory/mycluster/hosts.yaml all -m raw -a "systemctl start rpcbind
&& systemctl start nfs-server && systemctl start nfs-lock && systemctl start nfs-
idmap"
```

https://www.kubeflow.org/docs/other-guides/kubeflow-on-multinode-cluster/#in-case-of-existing-kubeflow-installation

2.4 Tune Parameters

2.4.1 Disable nodellocalDns

Not Using the nodellocalDns

```
# Set manual server if using a custom cluster DNS server
# manual_dns_server: 10.x.x.x
# Enable nodelocal dns cache
enable_nodelocaldns: false
#nodelocaldns_ip: 169.254.25.10
```

2.4.2 Modify yaml according to Swapoff

```
vim roles/download/tasks/download_container.yml
75 - name: Stop if swap enabled
76   assert:
77     that: ansible_swaptotal_mb == 0
78   when: kubelet_fail_swap_on|default(false)
```

2.4.3 K8s Dashboard Version > v2.0

Due to the version conflict problem with k8s 1.16, the Dashboard version must > v2.0.0-beta5

https://github.com/kubernetes/dashboard/releases

```
dashboard_image_repo: "{{ gcr_image_repo }}/google_containers/kubernetes-
dashboard-{{ image_arch }}"
dashboard_image_tag: "v2.0.0-beta6"
```

2.4.4 User accounts

Kubespray sets up two Kubernetes accounts by default: **root** and **kube**. Their passwords default to changeme. You can set this by changing **kube_api_pwd**.

```
roles/kubespray-defaults/defaults/main.yaml
  |-> kube_api_pwd:
```

2.4.5 DNS variables

By default, dnsmasq gets set up with 8.8.8.8 as an upstream DNS server and all other settings from your existing /etc/resolv.conf are lost. Set the following variables to match your requirements.

- upstream_dns_servers - Array of upstream DNS servers configured on host in addition to Kubespray deployed DNS
- nameservers - Array of DNS servers configured for use in dnsmasq
- searchdomains - Array of up to 4 search domains
- skip_dnsmasq - Don't set up dnsmasq (use only KubeDNS)

```
inventory/devopscluster/group_vars/all/all.yml
  |-> upstream_dns_servers:
```

2.4.6 Enable metrics to fetch

```
inventory/devopscluster/group_vars/all/all.yml
  |-> kube_read_only_port: 10255
```

2.4.7 Bootstrap OS

```
inventory/devopscluster/group_vars/all/all.yml
  |-> boostrap_os: centos
```

2.4.8 Configure kubectl to access the cluster

```
'inventory/devopscluster/group_vars/k8s-cluster/k8s-cluster.yml'
  |-> kubeconfig_localhost: true
```

2.4.9 Select CNI

```
'inventory/devopscluster/group_vars/k8s-cluster/k8s-cluster.yml'
  |-> kube_network_plugin: calico
```

2.4.10 Persistench Volume, ONLY support OpenStack

```
inventory/devopscluster/group_vars/k8s-cluster/addons.yml
  |->local_volume_provisioner_enabled: true
    cert_manager_enabled: true
```

3. Debug Command

3.1 Cannot Use kubectl Command

Error -- "The connection to the server lb-apiserver.kubernetes.local:8443 was refused" Error -- "The connection to the server localhost:8080 was refused - did you specify the right host or port?"

```
sudo cp /etc/kubernetes/admin.conf $HOME/ && sudo chown $(id -u):$(id -g)
$HOME/admin.conf && export KUBECONFIG=$HOME/admin.conf

export KUBECONFIG=/etc/kubernetes/kubelet.conf

sudo -i
swapoff -a
exit
strace -eopenat kubectl version
```

3.2 Solve the problem of timeout

Error trying to reach service: 'dial tcp 10.233.70.1:8443:

```
sudo route add -net <kubernetes-dashboard_Endpoints_ip> netmask 255.255.255.255 gw
<worker_node_ip>
```

4. Kubernetes Dashboard Problem

In order to show the kube dashboard, need to create admin RBAC account, and use its token or kubeconfig to login. In another hand, to simplify the login process, importing cert locally also can help login. The way is first exporting the cert of kubeconfig into customer's machine, and then importing into browser see - .[4.2 Export the Certificate](!4.2 Export the Certificate)

4.1 Create Admin Account

- admin-role.yaml

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
```

```
metadata:
  name: admin
  annotations:
    rbac.authorization.kubernetes.io/update: "true"
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: admin
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin
  namespace: kube-system
  labels:
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
```

- Commands to get token

```
    # Create the admin role.
    kubectl create -f admin-role.yaml

    # find the secret start with "admin-token"
    kubectl  -n kube-system get secret
    # Retrieve corresponding token
    kubectl -n kube-system get secret admin-token-tmh9v -o jsonpath=
{.data.token}|base64 -d

    # also can exec  "kubectl -n kube-system describe secret admin-token-tmh9" to
get the token

    # Dashboard URL: https://<first_master>:6443/api/v1/namespaces/kube-
system/services/https:kubernetes-dashboard:/proxy/#!/login
    # use above token to login this web URL
    # succeed.
```

4.2 Export the Certificate

```
```
cd .kube/
  # export the ppk
  $ grep 'client-certificate-data' ~/.kube/config | head -n 1 | awk '{print $2}' |
base64 -d >> kubecfg.crt
```
```

```
  # export the public key
  $ grep 'client-key-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64
-d >> kubecfg.key

  # Generate into p12 format cert.
  $ openssl pkcs12 -export -clcerts -inkey kubecfg.key -in kubecfg.crt -out
kubecfg.p12
  # type the password fpr the certificate

  # copy certs back to machine will visit the dashboard
  scp devop@ip:~/.kube/kubecfg.p12 .~~~~
```

5. kube-apiserver HA

```
yum install -y haproxy keepalived
vim /etc/haproxy/haproxy.cfg
listen kubernetes-apiserver-https
  bind *:8443
  option ssl-hello-chk
  mode tcp
  timeout client 3h
  timeout server 3h
  server master1 192.168.10.81:6443
  server master2 192.168.10.82:6443
  server master3 192.168.10.83:6443
  balance roundrobin
```

6. Detail Parameters Explaination

6.1 Common vars that are used in Kubespray

- calico_version - Specify version of Calico to use
- calico_cni_version - Specify version of Calico CNI plugin to use
- docker_version - Specify version of Docker to used (should be quoted string)
- etcd_version - Specify version of ETCD to use
- ipip - Enables Calico ipip encapsulation by default
- hyperkube_image_repo - Specify the Docker repository where Hyperkube resides
- hyperkube_image_tag - Specify the Docker tag where Hyperkube resides
- kube_network_plugin - Sets k8s network plugin (default Calico)
- kube_proxy_mode - Changes k8s proxy mode to iptables mode
- kube_version - Specify a given Kubernetes hyperkube version
- searchdomains - Array of DNS domains to search when looking up hostnames
- nameservers - Array of nameservers to use for DNS lookup

6.2 Cluster variables

Kubernetes needs some parameters in order to get deployed. These are the following default cluster paramters:

- cluster_name - Name of cluster (default is cluster.local)
- domain_name - Name of cluster DNS domain (default is cluster.local)
- kube_network_plugin - Plugin to use for container networking
- kube_service_addresses - Subnet for cluster IPs (default is 10.233.0.0/18). Must not overlap with kube_pods_subnet
- kube_pods_subnet - Subnet for Pod IPs (default is 10.233.64.0/18). Must not overlap with kube_service_addresses.
- kube_network_node_prefix - Subnet allocated per-node for pod IPs. Remainin bits in kube_pods_subnet dictates how many kube-nodes can be in cluster.
- dns_setup - Enables dnsmasq
- dns_server - Cluster IP for dnsmasq (default is 10.233.0.2)
- skydns_server - Cluster IP for KubeDNS (default is 10.233.0.3)
- cloud_provider - Enable extra Kubelet option if operating inside GCE or OpenStack (default is unset)
- kube_hostpath_dynamic_provisioner - Required for use of PetSets type in Kubernetes
- Note, if cloud providers have any use of the 10.233.0.0/16, like instances' private addresses, make sure to pick another values for kube_service_addresses and kube_pods_subnet, for example from the 172.18.0.0/16.

6.3 Addressing variables

- ip - IP to use for binding services (host var)
- access_ip - IP for other hosts to use to connect to. Often required when deploying from a cloud, such as OpenStack or GCE and you have separate public/floating and private IPs.
- ansible_default_ipv4.address - Not Kubespray-specific, but it is used if ip and access_ip are undefined
- loadbalancer_apiserver - If defined, all hosts will connect to this address instead of localhost for kube-masters and kube-master[0] for kube-nodes. See more details in the HA guide.
- loadbalancer_apiserver_localhost - makes all hosts to connect to the apiserver internally load balanced endpoint. Mutual exclusive to the loadbalancer_apiserver. See more details in the HA guide.