Kubernetes

Entity

-------------

* Container
* Pod : pod can have one or more container
* Service : exposing to outside world

Install : [How to Install Kubernetes on Rocky Linux {Manual or via Ansible} (phoenixnap.com)](https://phoenixnap.com/kb/install-kubernetes-on-rocky-linux)

* Install docker or some other container runtime
* sudo dnf config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo> //set repo
* sudo dnf makecache //refresh cache
* sudo dnf install -y containerd.io
* sudo mv /etc/containerd/config.toml /etc/containerd/config.toml.bak
* containerd config default > config.toml
* sudo vi config.toml : find and change SystemdCgroup = true
* sudo systemctl enable --now containerd.service
* sudo nano /etc/modules-load.d/k8s.conf and add below two items

overlay

br\_netfilter

* sudo modprobe overlay

sudo modprobe br\_netfilter

* sudo setenforce 0 //selinux config change
* sudo sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=permissive/g' /etc/sysconfig/selinux
* Sestatus
* Some firewall rules
  + master

sudo firewall-cmd --permanent --add-port=6443/tcp

sudo firewall-cmd --permanent --add-port=2379-2380/tcp

sudo firewall-cmd --permanent --add-port=10250/tcp

sudo firewall-cmd --permanent --add-port=10251/tcp

sudo firewall-cmd --permanent --add-port=10259/tcp

sudo firewall-cmd --permanent --add-port=10257/tcp

sudo firewall-cmd --permanent --add-port=179/tcp

sudo firewall-cmd --permanent --add-port=4789/udp

* On worker

sudo firewall-cmd --permanent --add-port=179/tcp

sudo firewall-cmd --permanent --add-port=10250/tcp

sudo firewall-cmd --permanent --add-port=30000-32767/tcp

sudo firewall-cmd --permanent --add-port=4789/udp

* sudo firewall-cmd --reload
* Vi /etc/sysctl.d/k8s.conf and add below entry

net.ipv4.ip\_forward = 1

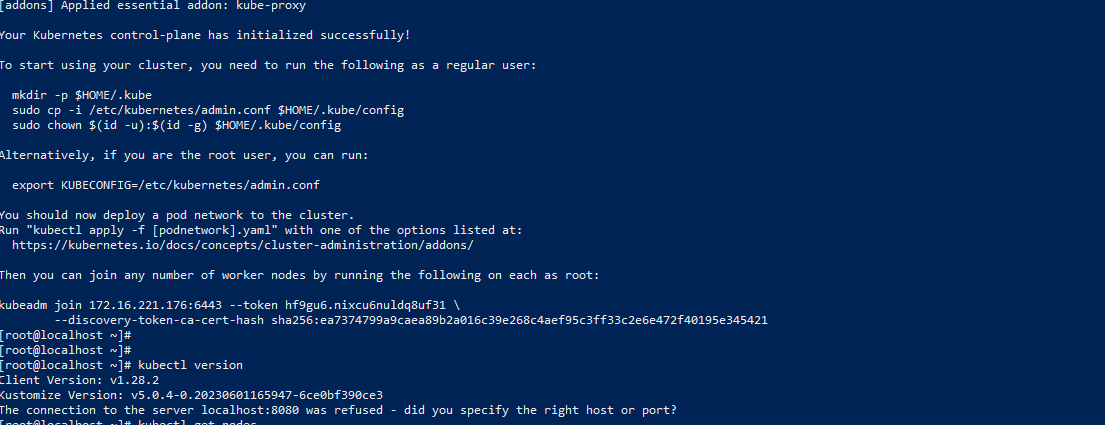
net.bridge.bridge-nf-call-ip6tables = 1

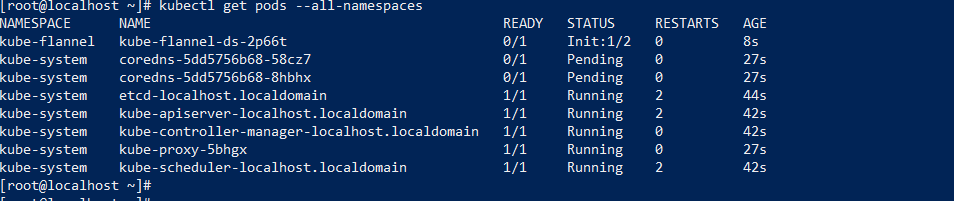
net.bridge.bridge-nf-call-iptables = 1

* sudo sysctl --system
* Modify fewlinux config
  + sudo swapoff -a
  + sudo sed -e '/swap/s/^/#/g' -i /etc/fstab
* sudo dnf config-manager --add-repo <https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64>
* sudo dnf config-manager --add-repo <https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/> //alternate repo
* sudo dnf install -y kubectl kubelet kubeadm
* You can install workers and connect to server

INIT

----------

* sudo kubeadm init // on master
* At the end of above command you get instruction on how to join master
* 
* Add above mentioned commands
* sudo kubeadm join <master-ip>:<port> --token <token> --discovery-token-ca-cert-hash <hash> // join workers
* kubectl get nodes // to check nodes
* kubectl describe node <nodename>
* Kubectl apply -f <https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml> or
* kubectl apply -f <https://raw.githubusercontent.com/flannel-io/flannel/v0.20.2/Documentation/kube-flannel.yml>
* kubectl get pods --all-namespaces // after install few automatic pods will always be running



* By default kubernetes don’t schedule nodes on naster node , to allw that run bellow command
  + kubectl taint node <node> node-role.kubernetes.io/master=:NoSchedule // to allow scheduling of pod on master node
  + kubectl taint nodes --all node-role.kubernetes.io/master-
  + kubectl taint nodes --all node-role.kubernetes.io/control-plane-
* To use local docker images , set up the registry by following command
  + docker run -d -p 5000:5000 --restart=always --name registry registry:2
  + docker build . -t localhost:5000/my-nginx-appp //create image
  + docker push localhost:5000/my-nginx-appp //push image

Commands

------------------

* Kubectl exec -it <pod-name> --bash //to go inside pod
* Exit // to come out of pod
* Kubectl log <podname>
* Kubectl delete -f <yamlfile>
* kubectl get service
* kubectl describe service <servicename>

Run

------------

* kubectl run nginx-pod<pod-name> --image=nginx-app <image from docker repo>
* Kubctl describe pod <pod-name>
* Kubectl delete pod <pod name>
* Create yaml file name my-<appname>.yml with below content

apiVersion: v1

kind: Pod

metadata:

name: my-nginx-pod

spec:

containers:

- name: my-app

image: localhost:5000/my-nginx-appp

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

* kubectl create -f my-pod.yaml
* kubectl port-forward my-nginx-pod 8083:80 // forward the node port to pod port
* **To expose service outside kubernetes cluster**
  + Add label inside yaml

apiVersion: v1

kind: Pod

metadata:

name: my-nginx-pod

labels:

zone: pod

version: v1

spec:

containers:

- name: my-app

image: localhost:5000/my-nginx-appp

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

* kubectl expose pod my-nginx-pod --name=hello-world-svc --port=80 --type=NodePort //to expose pod outside cluster
* kubectl describe service <servicename>
* Read the nodeport from above output and you can access the service using nodeip:nodeport

Ingress

--------------

* Used to direct traffic to service based on req
* Ingress can do below
  + Provide externally-reachable URLs for services.
  + Load balance traffic.
  + Terminate SSL/TLS.
  + Offer name-based virtual hosting.

Gateway

----------------

* Advanced version of ingress , with lots of added functionality

Notes

--------

* A pod can have one or more containers, this is the smallest deployable unit in kubernetes
* Having multiple container in pod keeps them on same node (sidecar containers are packed with its master)
* Autoscale should create more pods not more container in pod
* Containers in same pod communicates using localhost:port, containers in different pod use ip:port
* To access pod outside kubernetes cluster
* To auto scale pod
  + Yaml parameters can be used to autoscale pods based on various parameters like cpu utilization, num of packets etc
  + Ex : [tutorials/lessons/071/k8s/2-hpa.yaml at main · antonputra/tutorials (github.com)](https://github.com/antonputra/tutorials/blob/main/lessons/071/k8s/2-hpa.yaml)
* For HPA(horizontal pod autoscale) , service act as load balanced to pass traffic to different nodes, using below algo
  + Round robin
  + Ip hash
  + Random
  + Weighted round robin
  + Least connection
  + For example , see below service yaml

apiVersion: v1

kind: Service

metadata:

name: my-loadbalancer-service

spec:

selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 8080

type: LoadBalancer

**sessionAffinity: ClientIP // this actually deside the load balance type**