Setup Kubernetes Cluster using Kubeadm on Ubuntu

We are using the following Hostnames & IP Assignments:

Kubernetes Master Node

```
— kube-master : 10.10.121.56
•4 Kubernetes Worker Nodes
—kube-node1: 10.10.121.57
— kube-node2 : 10.10.121.71
— kube-node3 : 10.10.121.84
— kube-node4 : 10.10.121.25
```

install Docker on Master (10.10.121.56)

references : https://docs.docker.com/engine/install/ubuntu/

Install kubelet,kubeadm,kubectl and containerd.io on Master and worker nodes

At the end of 2021 kubernetes (release v1.22) plans to end support with docker,

as can be seen on the kubernetes website https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/

install containerd and Configure Private Registry for Kubernetes cluster running with containerd :

```
sudo modprobe overlay
sudo modprobe br_netfilter
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
```

```
overlay
br_netfilter
EOF
wget
https://github.com/containerd/containerd/releases/download/v1.7.0/conta
inerd-1.7.0-linux-amd64.tar.gz
tar -xvzf containerd-1.7.0-linux-amd64.tar.gz
cd bin/
cp * /usr/local/bin
cd /etc/systemd/system/
```

For containerd service file : https://raw.githubusercontent.com/containerd/containerd/main/containerd.service

```
vi containerd.service
```

```
# Copyright The containerd Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

[Unit]
Description=containerd container runtime
Documentation=https://containerd.io
After=network.target local-fs.target

[Service]
#uncomment to enable the experimental sbservice (sandboxed) version of
containerd/cri integration
#Environment="ENABLE_CRI_SANDBOXES=sandboxed"
ExecStartPre=-/sbin/modprobe overlay
ExecStart=/usr/local/bin/containerd

Type=notify
Delegate=yes
KillMode=process
Restart=always
```

```
RestartSec=5
# Having non-zero Limit*s causes performance problems due to accounting
overhead
# in the kernel. We recommend using cgroups to do container-local
accounting.
LimitNPROC=infinity
LimitCORE=infinity
LimitNOFILE=infinity
# Comment TasksMax if your systemd version does not supports it.
# Only systemd 226 and above support this version.
TasksMax=infinity
OOMScoreAdjust=-999

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

1. ____

```
systemctl daemon-reload
```

```
wget
https://github.com/opencontainers/runc/releases/download/v1.1.6/runc.am
d64
install -m 755 runc.amd64 /usr/local/sbin/runc
wget
https://github.com/containernetworking/plugins/releases/download/v1.2.0
/cni-plugins-linux-amd64-v1.2.0.tgz
mkdir -p /opt/cni/bin
tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.2.0.tgz
mkdir -p /etc/containerd
containerd config default > /etc/containerd/config.toml
systemctl restart containerd
```

```
vi /etc/containerd/config.toml
```

```
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]

  [plugins."io.containerd.gc.v1.scheduler"]
    deletion_threshold = 0
    mutation_threshold = 100
    pause_threshold = 0.02
    schedule_delay = "0s"
    startup_delay = "100ms"

  [plugins."io.containerd.grpc.v1.cri"]
    cdi_spec_dirs = ["/etc/cdi", "/var/run/cdi"]
    device_ownership_from_security_context = false
    disable_apparmor = false
    disable_cgroup = false
    disable_hugetlb_controller = true
    disable_proc_mount = false
    disable_tcp_service = true
    drain_exec_sync_io_timeout = "0s"
    enable_cdi = false
    enable_selinux = false
    enable_tls_streaming = false
    enable_unprivileged_icmp = false
    enable_unprivileged_ports = false
    ignore_image_defined_volumes = false
    image_pull_progress_timeout = "1m0s"
    max_concurrent_downloads = 3
    max_container_log_line_size = 16384
    netns_mounts_under_state_dir = false
    restrict_oom_score_adj = false
    sandbox_image = "registry.k8s.io/pause:3.9"
    selinux_category_range = 1024
    stats_collect_period = 10
    stream_idle_timeout = "4h0m0s"
    stream_server_address = "127.0.0.1"
    stream_server_port = "0"
    systemd_cgroup = false
    tolerate_missing_hugetlb_controller = true
    unset_seccomp_profile = ""

    [plugins."io.containerd.grpc.v1.cri".cni]
      bin_dir = "/opt/cni/bin"
      conf_dir = "/etc/cni/net.d"
      conf_template = ""
      ip_pref = ""
      max_conf_num = 1
      setup_serially = false

    [plugins."io.containerd.grpc.v1.cri".containerd]
      default_runtime_name = "runc"
      disable_snapshot_annotations = true
      discard_unpacked_layers = false
      ignore_blockio_not_enabled_errors = false
      ignore_rdt_not_enabled_errors = false
      no_pivot = false
      snapshotter = "overlayfs"

      [plugins."io.containerd.grpc.v1.cri".containerd.default_runtime]
        base_runtime_spec = ""
        cni_conf_dir = ""
        cni_max_conf_num = 0
        container_annotations = []
        pod_annotations = []
        privileged_without_host_devices = false
        privileged_without_host_devices_all_devices_allowed = false
        runtime_engine = ""
        runtime_path = ""
        runtime_root = ""
        runtime_type = ""
        sandbox_mode = ""
        snapshotter = ""


[plugins."io.containerd.grpc.v1.cri".containerd.default_runtime.options
]

      [plugins."io.containerd.grpc.v1.cri".containerd.runtimes]

        [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
          base_runtime_spec = ""
          cni_conf_dir = ""
          cni_max_conf_num = 0
          container_annotations = []
          pod_annotations = []
          privileged_without_host_devices = false
          privileged_without_host_devices_all_devices_allowed = false
          runtime_engine = ""
          runtime_path = ""
          runtime_root = ""
          runtime_type = "io.containerd.runc.v2"
          sandbox_mode = "podsandbox"
          snapshotter = ""


[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
            BinaryName = ""
            CriuImagePath = ""
            CriuPath = ""
            CriuWorkPath = ""
            IoGid = 0
            IoUid = 0
            NoNewKeyring = false
            NoPivotRoot = false
            Root = ""
            ShimCgroup = ""
            SystemdCgroup = true


[plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runt
ime]
        base_runtime_spec = ""
        cni_conf_dir = ""
        cni_max_conf_num = 0
        container_annotations = []
        pod_annotations = []
        privileged_without_host_devices = false
        privileged_without_host_devices_all_devices_allowed = false
        runtime_engine = ""
        runtime_path = ""
        runtime_root = ""
        runtime_type = ""
        sandbox_mode = ""
        snapshotter = ""


[plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runt
ime.options]

    [plugins."io.containerd.grpc.v1.cri".image_decryption]
      key_model = "node"

    [plugins."io.containerd.grpc.v1.cri".registry]
      config_path = ""

      [plugins."io.containerd.grpc.v1.cri".registry.auths]

      [plugins."io.containerd.grpc.v1.cri".registry.configs]

[plugins."io.containerd.grpc.v1.cri".registry.configs."10.10.121.29:808
5".tls]
            insecure_skip_verify = true

[plugins."io.containerd.grpc.v1.cri".registry.configs."10.10.121.29:808
5".auth]
            auth = "***************=="

      [plugins."io.containerd.grpc.v1.cri".registry.headers]

      [plugins."io.containerd.grpc.v1.cri".registry.mirrors]

[plugins."io.containerd.grpc.v1.cri".registry.mirrors."10.10.121.29:808
5"]
            endpoint = ["http://10.10.121.29:8085"]

    [plugins."io.containerd.grpc.v1.cri".x509_key_pair_streaming]
      tls_cert_file = ""
      tls_key_file = ""
```

```
[plugins."io.containerd.internal.v1.opt"]
  path = "/opt/containerd"

[plugins."io.containerd.internal.v1.restart"]
  interval = "10s"

[plugins."io.containerd.internal.v1.tracing"]
  sampling_ratio = 1.0
  service_name = "containerd"

[plugins."io.containerd.metadata.v1.bolt"]
  content_sharing_policy = "shared"

[plugins."io.containerd.monitor.v1.cgroups"]
  no_prometheus = false

[plugins."io.containerd.nri.v1.nri"]
  disable = true
  disable_connections = false
  plugin_config_path = "/etc/nri/conf.d"
  plugin_path = "/opt/nri/plugins"
  plugin_registration_timeout = "5s"
  plugin_request_timeout = "2s"
  socket_path = "/var/run/nri/nri.sock"

[plugins."io.containerd.runtime.v1.linux"]
  no_shim = false
  runtime = "runc"
  runtime_root = ""
  shim = "containerd-shim"
  shim_debug = false

[plugins."io.containerd.runtime.v2.task"]
  platforms = ["linux/amd64"]
  sched_core = false

[plugins."io.containerd.service.v1.diff-service"]
  default = ["walking"]

[plugins."io.containerd.service.v1.tasks-service"]
  blockio_config_file = ""
  rdt_config_file = ""

[plugins."io.containerd.snapshotter.v1.aufs"]
  root_path = ""

[plugins."io.containerd.snapshotter.v1.btrfs"]
  root_path = ""

[plugins."io.containerd.snapshotter.v1.devmapper"]
  async_remove = false
  base_image_size = ""
  discard_blocks = false
  fs_options = ""
  fs_type = ""
  pool_name = ""
  root_path = ""
  [plugins."io.containerd.snapshotter.v1.native"]
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.overlayfs"]
    root_path = ""
    upperdir_label = false

  [plugins."io.containerd.snapshotter.v1.zfs"]
    root_path = ""

  [plugins."io.containerd.tracing.processor.v1.otlp"]
    endpoint = ""
    insecure = false
    protocol = ""

  [plugins."io.containerd.transfer.v1.local"]

[proxy_plugins]

[stream_processors]

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env =
["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprov
ider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar"

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env =
["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprov
ider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar+gzip"

[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
_______________________________________________________________________
____________________________________________
```

Note : you need to replace ip and token

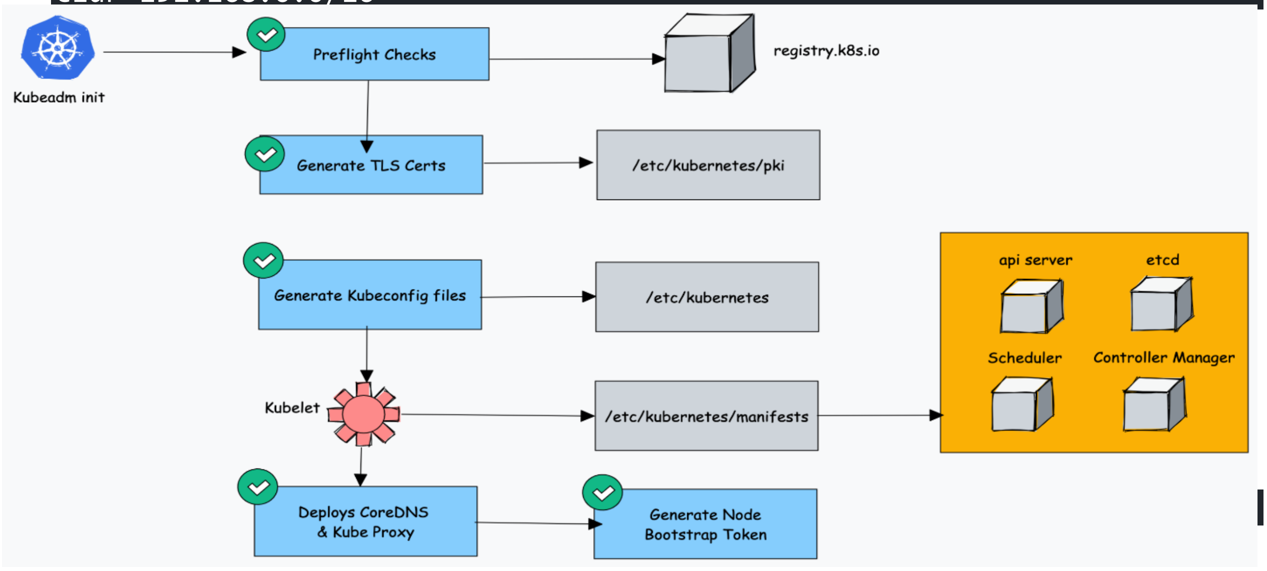
Install Kubernetes

```
swapoff -a
```

Kubernetes references : https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

In Master node

```
kubeadm init --apiserver-advertise-address=10.10.121.56 --pod-network-
cidr=192.168.0.0/16
```



Message after Kubeadm init is successful:

```
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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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/

Then you can join any number of worker nodes by running the following
on each as root:

kubeadm join 10.10.121.56:6443 --token zxen48.z5ruut0pwf8rxb4i --
```

To View the Join command:

```
kubeadm token create --print-join-command
```

Join the workers with join command.

Install flannel Network Plugin for Pod Networking:

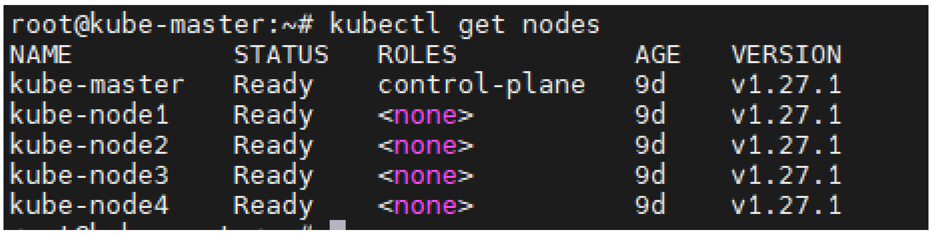
Reference : https://kubernetes.io/docs/concepts/cluster-administration/addons/#networking-and-network-policy

```
wget https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
```

```
kubectl apply -f kube-flannel
```

Note 1 : Before above 'apply' command, we have to check CIDR range of cluster in kube-flannel.yaml file. If it does not match with that of mentioned in 'kubeadm init' command…..we have to change it.
Node 2 : We have to wait till all the nodes get into 'Ready' state.

Output:



```
root@kube-master:~# kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
kube-master    Ready    control-plane   9d    v1.27.1
kube-node1     Ready    <none>          9d    v1.27.1
kube-node2     Ready    <none>          9d    v1.27.1
kube-node3     Ready    <none>          9d    v1.27.1
kube-node4     Ready    <none>          9d    v1.27.1
```

Install metal lb :

Reference : https://metallb.universe.tf/installation/

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.9/config/manifests/metallb-native.yaml
```

```
vi l2advertisement.yaml
```

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: example
  namespace: metallb-system
spec:
  ipAddressPools:
```

```
  - first-pool
```

```
vi ipaddresspoll.yaml
```

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: first-pool
  namespace: metallb-system
spec:
  addresses:
  - 10.10.XXXX-10.10.XXXXX
```

```
kubectl create -f ipaddresspoll.yaml
```

```
kubectl create -f l2advertisement.yaml
```

Install Helm:

```
wget https://get.helm.sh/helm-v3.11.2-linux-amd64.tar.gz
```

```
 tar -zxvf helm-v3.11.2-linux-amd64.tar.gz
```

```
mv linux-amd64/helm /usr/local/bin/helm
```

Install Ingress :

```
git clone https://github.com/nginxinc/kubernetes-ingress.git --branch
v3.1.0
```

```
cd kubernetes-ingress/deployments/helm-chart
```

```
helm repo add nginx-stable https://helm.nginx.com/stable
kubectl apply -f crds/
```

```
helm install nginx-controller-one nginx-stable/nginx-ingress --set
controller.ingressClass=nginx-one --namespace ingress-nginx-one --
create-namespace
```

```
kubectl get svc -n ingress-nginx-one
```