Install - Single (On-demand)

Kubernetes Install

Installing Docker

Docker .

Docker Ref : https://docs.docker.com/install/linux/docker-ce/centos/

Docker 17.03.2.ce .

```
# yum update
# yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
# yum-config-manager \
  --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
# yum install -y docker
# yum install --setopt=obsoletes=0 \
  docker-ce-17.03.2.ce-1.el7.centos \
  docker-ce-selinux-17.03.2.ce-1.el7.centos
# systemctl enable docker && systemctl start docker
```

Installing kubeadm, kubelet and kubectl

Kubeadm : .

Kubelet : .

Kubectl : CLI .

Kubernetes Ref : https://kubernetes.io/docs/setup/independent/install-kubeadm/

Install .

```
# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
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
EOF
# setenforce 0
# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
# systemctl enable kubelet && systemctl start kubelet
```

(Optional) Configure cgroup driver used by kubelet on Master Node

Kubelet Cgroup Docker Cgroup .

Kubectl 1.10 .

```
# docker info | grep -i cgroup
# cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
# sed -i "s/cgroup-driver=systemd/cgroup-driver=cgroupfs/g" /etc/systemd/system/kubelet.service.d/10-kubeadm.
conf
# systemctl daemon-reload
# systemctl restart kubelet
```

Docker Kubeadm Kubelet cgroup /var/lib/kubelet/kubeadm-flags.env .

CRI cgroup-driver /etc/default/kubelet , .

```
# KUBELET_KUBEADM_EXTRA_ARGS=--cgroup-driver=<value>
```

CRIX cgroup cgroupfs kgrouplet .

Kubeadm init (K8s Version 1.10)

```
# kubeadm init
[init] Using Kubernetes version: v1.10.3
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks.
        [WARNING Firewalld]: firewalld is active, please ensure ports [6443 10250] are open or your cluster
may not function correctly
        [WARNING FileExisting-crictl]: crictl not found in system path
Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [localhost.localdomain kubernetes kubernetes.
default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.0.2.15]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
...
Your Kubernetes master 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 machines by running the following on each node
as root
  kubeadm join ...
```

Kubeadm init (K8s Version 1.11)

```
[init] using Kubernetes version: v1.11.2
[preflight] running pre-flight checks
I0810 15:52:27.254403   14590 kernel_validator.go:81] Validating kernel version
I0810 15:52:27.254663   14590 kernel_validator.go:96] Validating kernel config
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [sbion-kubernetes69 kubernetes kubernetes.
default ...
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
```

```
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [sbion-kubernetes69 localhost] and IPs
[127.0.0.1 ::1]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names ...
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-
apiserver.yaml"
[controlplane] wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests
/kube-controller-manager.yaml"
[controlplane] wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-
scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"
[init] waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes
/manifests"
[init] this might take a minute or longer if the control plane images have to be pulled
[apiclient] All control plane components are healthy after 41.502496 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.11" in namespace kube-system with the configuration for the
kubelets in the cluster
[markmaster] Marking the node ...
[markmaster] Marking the node ...
[patchnode] Uploading the CRI Socket information ...
[bootstraptoken] using token: ...
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get
long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a
Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the
cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master 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 machines by running the following on each node
as root:

  kubeadm join ...
```

Master Setup

```
# mkdir -p $HOME/.kube
# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
# sudo chown $(id -u):$(id -g) $HOME/.kube/config

# kubectl version
Client Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.3", GitCommit:"
2bba0127d85d5a46ab4b778548be28623b32d0b0", GitTreeState:"clean", BuildDate:"2018-05-21T09:17:39Z",
GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.3", GitCommit:"
2bba0127d85d5a46ab4b778548be28623b32d0b0", GitTreeState:"clean", BuildDate:"2018-05-21T09:05:37Z",
GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}
# kubectl version
Client Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.1", GitCommit:"
b1b29978270dc22fecc592ac55d903350454310a", GitTreeState:"clean", BuildDate:"2018-07-17T18:53:20Z",
GoVersion:"go1.10.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.2", GitCommit:"
bb9ffb1654d4a729bb4cec18ff088eacc153c239", GitTreeState:"clean", BuildDate:"2018-08-07T23:08:19Z",
GoVersion:"go1.10.3", Compiler:"gc", Platform:"linux/amd64"}

# kubectl get nodes
NAME                     STATUS     ROLES     AGE       VERSION
localhost.localdomain   NotReady   master    4m        v1.10.3

# kubectl cluster-info
Kubernetes master is running at https://10.0.2.15:6443
KubeDNS is running at https://10.0.2.15:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

# kubectl get pods --all-namespaces
NAMESPACE     NAME                                          READY     STATUS    RESTARTS   AGE
kube-system   etcd-localhost.localdomain                    1/1       Running   0          4m
kube-system   kube-apiserver-localhost.localdomain          1/1       Running   0          4m
kube-system   kube-controller-manager-localhost.localdomain   1/1       Running   0          4m
kube-system   kube-dns-86f4d74b45-275td                     0/3       Pending   0          5m
kube-system   kube-proxy-rp79j                              1/1       Running   0          5m
kube-system   kube-scheduler-localhost.localdomain          1/1       Running   0          4m
```

kube-dns Pending .

.. Join .

, 3 .

Installing a pod network add-on

Pod .

(K8s Version 1.10)

```
# export kubever=$(kubectl version | base64 | tr -d '')
# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount "weave-net" created
clusterrole.rbac.authorization.k8s.io "weave-net" created
clusterrolebinding.rbac.authorization.k8s.io "weave-net" created
role.rbac.authorization.k8s.io "weave-net" created
rolebinding.rbac.authorization.k8s.io "weave-net" created
daemonset.extensions "weave-net" created
```

(K8s Version 1.11)

```
# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.extensions/weave-net created
```

, get nodes NotReady Ready .

```
# kubectl get nodes
NAME                    STATUS    ROLES     AGE       VERSION
localhost.localdomain   Ready     master    33m       v1.10.3
```

Master Isolation

. (: Kubernetes

```
# kubectl taint nodes --all node-role.kubernetes.io/master-
node "test-01" untainted
taint "node-role.kubernetes.io/master:" not found
taint "node-role.kubernetes.io/master:" not found
```

Joining your nodes

```
# kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hash sha256:<hash>
[preflight] running pre-flight checks
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the
follovs_sh ip_vs ip_vs_rr] or no builtin kernel ipvs support: map[ip_vs_rr:{} ip_vs_wrr:{} ip_vs_sh:{}
nf_connt
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
 2. Provide the missing builtin kernel ipvs support

I0810 16:30:39.529114   17494 kernel_validator.go:81] Validating kernel version
I0810 16:30:39.529419   17494 kernel_validator.go:96] Validating kernel config
[discovery] Trying to connect to API Server "180.70.98.69:6443"
[discovery] Created cluster-info discovery client....
[discovery] Requesting info from....
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roo
[discovery] Successfully established connection with API Server....
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.11" ConfigMap in the kube-s
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "sbion-

This node has joined the cluster:
* Certificate signing request was sent to master and a response
  was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster
```

.

```
# kubeadm token list
TOKEN                     TTL  EXPIRES              USAGES           DESCRIPTION            EXTRA GROUPS
8ewj1p.9r9hcjoqgajrj4gi   23h  2018-06-12T02:51:28Z authentication,  The default bootstrap  system:
                                                    signing          token generated by     bootstrappers:
                                                                     'kubeadm init'.        kubeadm:
                                                                                            default-node-token
```

24 , .

```
# kubeadm token create
5didvk.d09sbcov8ph2amjw
```

Minikube

- Kubernetes Local , .
- DNS, Dashboards, CNI, NodePorts, ConfigMaps Secrets kubernetes .
- Minikube Docker .
- Kubernetes .

Minikube for Windows

- BIOS VT-x AMD-v .
- VirtualBox .
  - https://www.virtualbox.org/wiki/Downloads
- VirtualBox windows / Hyper-V disable
- Docker-toolbox for window . (virtualbox ,) - https://docs.docker.com/toolbox/toolbox_install_windows/
- Kubectl
  - Chocolatey https://chocolatey.org/ ( )
  - Choco install kubernetes-cli
  - Kubectl version .

- Minikube-windows-amd64.exe
  - https://github.com/kubernetes/minikube/releases
- minikube.exe windows system path (C:\install , path)
- minikube-installer.exe

---

crictl not found in system path

Kubenetes `kubeadm init` `crictl not found in system path` (Warning) .

crictl .

crictl .

Go Go , crictl .

```
go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
```

crictl .

Suggestion . , go .

.

```
crictl not found in system path
Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
```

The connection to the server localhost:8080 was refused

kubectl .

```
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

admin.conf kubeadm init .

,master kubectl , kubectl master

dmin.conf kubectl .

[ERROR docker] docker version is greater than the most recently validated version. Docker version: 17.12.0-ce. Max validated version: 17.03

, 17.03 .. ... .

, 17.12 17.03 .

```
# yum list docker-ce --showduplicates | sort -r | grep 17.03
# yum install docker-ce-<VERSION STRING>
# yum install --setopt=obsoletes=0 \
  docker-ce-17.03.2.ce-1.el7.centos \
  docker-ce-selinux-17.03.2.ce-1.el7.centos
# systemctl start docker && systemctl enable docker
```

[ERROR Swap]: running with swap on is not supported. Please disable swap

Swap .

```
# swapoff -a
# cat /etc/fstab
# Created by anaconda on Thu Feb  8 11:02:14 2018
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=4978c64f-38be-4e76-9061-132b6dc77015 /                       xfs     defaults        0 0
UUID=152043c4-03ea-4755-acc0-9a13982cb755 /boot                   xfs     defaults        0 0
UUID=8fefa47f-cd08-44aa-9f5c-c57e10eefb0c swap                    swap    defaults        0 0

### weblog backup ###
SYSBACKUP1:/vol/vol1/Permanent/APPLICATION_LOG          /LOG_BACKUP             nfs     defaults        1 2

### system backup ###
SYSBACKUP2:/vol/vol2/SYSTEM_UNIX_BAK                    /system_backup_UNIX     nfs     defaults        1 2

# cp fstab fstab_bacup
# rm fstab
# systemctl daemon-reload
# systemctl restart kubelet
```

[ERROR FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists

[ERROR FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists

[ERROR FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists

[ERROR FileAvailable--etc-kubernetes-manifests-etcd.yaml]: /etc/kubernetes/manifests/etcd.yaml already exists

.

.

.

```
error: failed to run Kubelet: failed to create kubelet: misconfiguration: kubelet cgroup driver: "systemd"
```

.

```
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=systemd"  Environment="KUBELET_CGROUP_ARGS=--cgroup-
driver=cgroupfs"
systemctl daemon-reload
kubeadm reset
kubeadm init
```

.

```
In order to set /proc/sys/net/bridge/bridge-nf-call-iptables by editing /etc/sysctl.conf. There you can add
[1]
```

.

```
net.bridge.bridge-nf-call-iptables = 1
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

sudo sysctl -p .