Setup Kubernetes 1.14 Cluster on CentOS 7.6

In this tutorial I will demonstrate how to setup Kubernetes 1.14 Cluster to take advantage of new features that many organizations and community were waiting for. As specially windows containers. Kubernetes now offers windows containers out of the box and allows you to add windows node to Kubernetes cluster.

Lab Setup:

Hostname	IP Address	RAM	CPU	Role
Master.example.com	10.0.2.10	2 GB	2	Kubernetes Master
Node1.example.com	10.0.2.11	2 GB	2	Kubernetes Worker
Node2.example.com	10.0.2.12	2 GB	2	Kubernetes Worker
Win-node1.example.com	10.0.2.51	2 GB	2	Kubernetes Worker

Step 1: Do the below tasks to prepare the hosts for Kubernetes Cluster

1. Configure IP Address
2. Configure Hostname
3. Configure /etc/hosts file to resolve hostnames
4. Stop Firewall
5. Disable SELinux
6. Disable SWAP Memory
7. Update the System
8. Reboot the system

Step 2: Once you have prepared the hosts by performing the above mentioned task, follow the below instructions:

1. Install Docker Package on all Nodes
2. Start and Enable Docker Service on all Nodes

```
~]#  yum install -y docker
~]# systemctl start docker
~]# systemctl enable docker
~]# docker version
```

Step 3: Configure hosts file to resolve hostnames on all nodes

~]# vim /etc/hosts

10.0.2.10	master.example.com	master
10.0.2.11	node1.example.com	node1
10.0.2.12	node2.example.com	node2
10.0.2.51	win-node1.example.com	win-node1

:wq (save and exit)

Step 4: Configure YUM Package Repository to Install Kubernetes 1.14 Cluster Packages on all Nodes

~]# vim /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

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

:wq (save and exit)

Step 5: Install Kubernetes Packages and Start & Enable kubelet service on all Nodes.

~]# yum install -y kubelet kubeadm kubectl

~]# systemctl enable kubelet && sudo systemctl start kubelet

Step 6: configure Network Bridge Setting for iptables on all Nodes.

~]# vim /etc/sysctl.d/k8s.conf

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

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

:wq (save and exit)

~]# sysctl --system

Step 7: Stop and Disable the Swap Partition on all nodes.

~]# swapoff -a && sed -i 's/.*swap.*/#&/' /etc/fstab

~]# swapon -s

Step 8: Change SELinux Mode to Permissive on all nodes.

~]# setenforce 0

~]# sed -i --follow-symlinks "s/^SELINUX=enforcing/SELINUX=permissive/g" /etc/sysconfig/selinux

Step 9: Install Docker Package and Start Docker Service on all Nodes

~]# yum install -y docker

~]# systemctl enable docker && sudo systemctl start docker

~]# docker version

Step 10: Configure Kubernetes Master on master node

[root@master ~]# kubeadm config images pull

```
[root@master ~]# kubeadm config images pull
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.14.1
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.14.1
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.14.1
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.14.1
[config/images] Pulled k8s.gcr.io/pause:3.1
[config/images] Pulled k8s.gcr.io/etcd:3.3.10
[config/images] Pulled k8s.gcr.io/coredns:1.3.1
[root@master ~]#
```

[root@master ~]# kubeadm init --apiserver-advertise-address=10.0.2.10 --pod-network-cidr 10.244.0.0/16

```
Your Kubernetes control-plane 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/

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

kubeadm join 10.0.2.10:6443 --token tfrgrs.3658k3ylk5pnxxfz \
    --discovery-token-ca-cert-hash sha256:fa30f08dc57f148b3068c6a978b3ef4798db3f871693c84e507fc16a9014a8ab
[root@master ~]#
```

[root@master ~]# mkdir -p $HOME/.kube

[root@master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config

Step 11: Configure Kubernetes Master on master node

Before we setup worker nodes, we need to ensure pod networking is functional.

Pod networking is also a dependency for kube-dns pod to manage pod dns.

[root@master ~]# wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

[root@master ~]# ls *.yml

```
[root@master ~]# wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
--2019-05-09 11:26:33--  https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.156.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.156.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12306 (12K) [text/plain]
Saving to: 'kube-flannel.yml'

100%[==========================================================================================================================>] 12,306      --.-K/s   in 0.003s

2019-05-09 11:26:33 (4.31 MB/s) - 'kube-flannel.yml' saved [12306/12306]

[root@master ~]# ls *.yml
kube-flannel.yml
[root@master ~]#
```

[root@master ~]# kubectl get pods --all-namespaces

```
[root@master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
kube-system   coredns-fb8b8dccf-q5c7l                   0/1     Pending   0          2m33s
kube-system   coredns-fb8b8dccf-scrgb                   0/1     Pending   0          2m33s
kube-system   etcd-master.example.com                   1/1     Running   0          2m
kube-system   kube-apiserver-master.example.com         1/1     Running   0          105s
kube-system   kube-controller-manager-master.example.com   1/1     Running   0          116s
kube-system   kube-proxy-c7j8h                          1/1     Running   0          2m33s
kube-system   kube-scheduler-master.example.com         1/1     Running   0          105s
[root@master ~]#
```

[root@master ~]# kubectl create -f kube-flannel.yml

```
[root@master ~]# kubectl create -f kube-flannel.yml
podsecuritypolicy.extensions/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
[root@master ~]#
```

[root@master ~]# kubectl get pods --all-namespaces

```
[root@master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
kube-system   coredns-fb8b8dccf-q5c7l                   1/1     Running   0          4m8s
kube-system   coredns-fb8b8dccf-scrgb                   1/1     Running   0          4m8s
kube-system   etcd-master.example.com                   1/1     Running   0          3m35s
kube-system   kube-apiserver-master.example.com         1/1     Running   0          3m20s
kube-system   kube-controller-manager-master.example.com   1/1     Running   0          3m31s
kube-system   kube-flannel-ds-amd64-qjd8k               1/1     Running   0          45s
kube-system   kube-proxy-c7j8h                          1/1     Running   0          4m8s
kube-system   kube-scheduler-master.example.com         1/1     Running   0          3m20s
[root@master ~]#
```

[root@master ~]# kubectl get nodes

```
[root@master ~]# kubectl get nodes
NAME                 STATUS   ROLES    AGE    VERSION
master.example.com   Ready    master   5m6s   v1.14.1
[root@master ~]#
```

Step 12: Configure Kubernetes Worker nodes

[root@node1 ~]# kubeadm join 10.0.2.10:6443 --token tfrgrs.3658k3ylk5pnxxfz \

--discovery-token-ca-cert-hash sha256:fa30f08dc57f148b3068c6a978b3ef4798db3f871693c84e507fc16a9014a8ab

```
[root@node1 ~]# kubeadm join 10.0.2.10:6443 --token tfrgrs.3658k3ylk5pnxxfz \
>     --discovery-token-ca-cert-hash sha256:fa30f08dc57f148b3068c6a978b3ef4798db3f871693c84e507fc16a9014a8ab
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.14" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[root@node1 ~]#
```

[root@node2 ~]# kubeadm join 10.0.2.10:6443 --token tfrgrs.3658k3ylk5pnxxfz \

--discovery-token-ca-cert-hash sha256:fa30f08dc57f148b3068c6a978b3ef4798db3f871693c84e507fc16a9014a8ab

```
[root@node2 ~]# kubeadm join 10.0.2.10:6443 --token tfrgrs.3658k3ylk5pnxxfz \
>     --discovery-token-ca-cert-hash sha256:fa30f08dc57f148b3068c6a978b3ef4798db3f871693c84e507fc16a9014a8ab
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.14" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[root@node2 ~]#
```

Step 13: Go to master Node and run the following command to get the node list:

[root@master ~]# kubectl get nodes

```
[root@master ~]# kubectl get nodes
NAME                 STATUS   ROLES    AGE     VERSION
master.example.com   Ready    master   7m49s   v1.14.1
node1.example.com    Ready    <none>   116s    v1.14.1
node2.example.com    Ready    <none>   109s    v1.14.1
[root@master ~]#
```

Congratulations, You have successfully configured Kubernetes cluster

Adding a Windows Node to Kubernetes Cluster

Step1: Configuring Flannel in VXLAN mode on Master Node

There are two sections you should modify to enable the vxlan networking backend:

[root@master ~]# vim kube-flannel.yml

in the net-conf.json section of kube-flannel.yml should look as follows:

```
net-conf.json: |
  {
    "Network": "10.244.0.0/16",
    "Backend": {
      "Type": "vxlan",
      "VNI" : 4096,
      "Port": 4789
    }
  }
```

```
net-conf.json: |
  {
    "Network": "10.244.0.0/16",
    "Backend": {
      "Type": "vxlan"
      "VNI" : 4096,
      "Port": 4789
    }
  }
```

In the cni-conf.json section of your kube-flannel.yml, change the network name to "vxlan0"

cni-conf.json: |

{

"name": "vxlan0",

```
cni-conf.json: |
  {
    "name": "vxlan0",
    "plugins": [
      {
```

Step2: Apply the Flannel yml and Validate it.

[root@master ~]# kubectl apply -f kube-flannel.yml

```
[root@master ~]# kubectl apply -f kube-flannel.yml
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
podsecuritypolicy.extensions/psp.flannel.unprivileged configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
clusterrole.rbac.authorization.k8s.io/flannel configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
clusterrolebinding.rbac.authorization.k8s.io/flannel configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
serviceaccount/flannel configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
configmap/kube-flannel-cfg configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset.extensions/kube-flannel-ds-amd64 configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset.extensions/kube-flannel-ds-arm64 configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset.extensions/kube-flannel-ds-arm configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset.extensions/kube-flannel-ds-ppc64le configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset.extensions/kube-flannel-ds-s390x configured
[root@master ~]#
```

Since the Flannel pods are Linux-based, apply a NodeSelector patch, to the Flannel DaemonSet pod:

[root@master ~]# wget https://raw.githubusercontent.com/microsoft/SDN/1d5c055bb195fecba07ad094d2d7c18c188f9d2d/Kubernetes/flannel/l2bridge/manifests/node-selector-patch.yml

[root@master ~]# kubectl patch ds/kube-flannel-ds-amd64 --patch "$(cat node-selector-patch.yml)" -n=kube-system

```
[root@master ~]# wget https://raw.githubusercontent.com/microsoft/SDN/1d5c055bb195fecba07ad094d2d7c18c188f9d2d/Kubernetes/flannel/l2bridge/manifests/node-selector-patch.yml
--2019-05-09 11:42:36--  https://raw.githubusercontent.com/microsoft/SDN/1d5c055bb195fecba07ad094d2d7c18c188f9d2d/Kubernetes/flannel/l2bridge/manifests/node-selector-patch.yml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.156.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.156.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 88 [text/plain]
Saving to: 'node-selector-patch.yml'

100%[======================================================================================================================>] 88          --.-K/s   in 0s

2019-05-09 11:42:36 (7.20 MB/s) - 'node-selector-patch.yml' saved [88/88]

[root@master ~]#
[root@master ~]# kubectl patch ds/kube-flannel-ds-amd64 --patch "$(cat node-selector-patch.yml)" -n=kube-system
daemonset.extensions/kube-flannel-ds-amd64 patched
[root@master ~]#
```

After a few minutes, you should see all the pods as running if the Flannel pod network was deployed.

[root@master ~]# kubectl get pods --all-namespaces

```
[root@master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                          READY   STATUS    RESTARTS   AGE
kube-system   coredns-fb8b8dccf-q5c7l                       1/1     Running   0          19m
kube-system   coredns-fb8b8dccf-scrgb                       1/1     Running   0          19m
kube-system   etcd-master.example.com                       1/1     Running   0          19m
kube-system   kube-apiserver-master.example.com             1/1     Running   0          19m
kube-system   kube-controller-manager-master.example.com    1/1     Running   0          19m
kube-system   kube-flannel-ds-amd64-flrrb                   1/1     Running   0          14m
kube-system   kube-flannel-ds-amd64-qjd8k                   1/1     Running   0          16m
kube-system   kube-flannel-ds-amd64-tsc6f                   1/1     Running   0          14m
kube-system   kube-proxy-c2m6h                              1/1     Running   0          14m
kube-system   kube-proxy-c7j8h                              1/1     Running   0          19m
kube-system   kube-proxy-s4mbr                              1/1     Running   0          14m
kube-system   kube-scheduler-master.example.com             1/1     Running   0          19m
[root@master ~]#
```

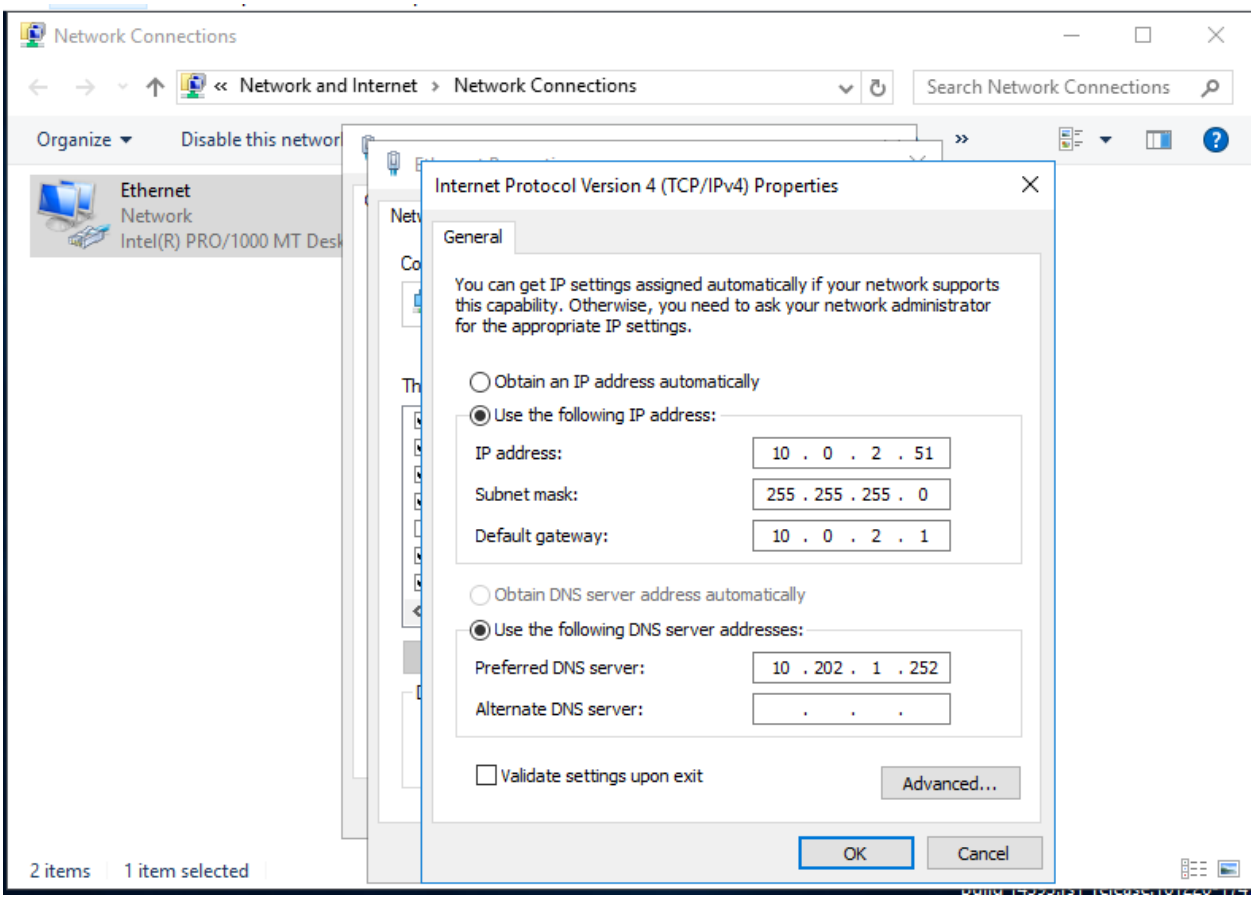
Verify that the Flannel DaemonSet has the NodeSelector applied.

[root@master ~]# kubectl get ds -n kube-system | grep kube-flannel-ds-amd64

```
[root@master ~]# kubectl get ds -n kube-system | grep kube-flannel-ds-amd64
kube-flannel-ds-amd64     3         3         3       0            3           beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux   17m
[root@master ~]#
```

Step3: Preparing a Windows Node:

3.1: Configure IP Address, Netmask, Gateway and DNS



3.2: Install Container Feature (requires a system reboot)

Open the PowerShell and Run the below command to Install and Enable Containers.

```
Enable-WindowsOptionalFeature -FeatureName Containers
Restart-Computer -Force
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Package -Name Docker -ProviderName DockerMsftProvider
Start-Service docker
docker version
```

3.3: Create a “Kubernetes for Windows” directory to store Kubernetes binaries as well as any deployment scripts and config files.

mkdir c:\k

Step4: Copy the Kubernetes certificate file $HOME/.kube/config from the Linux controller to this new "C:\k" directory on your Windows node.

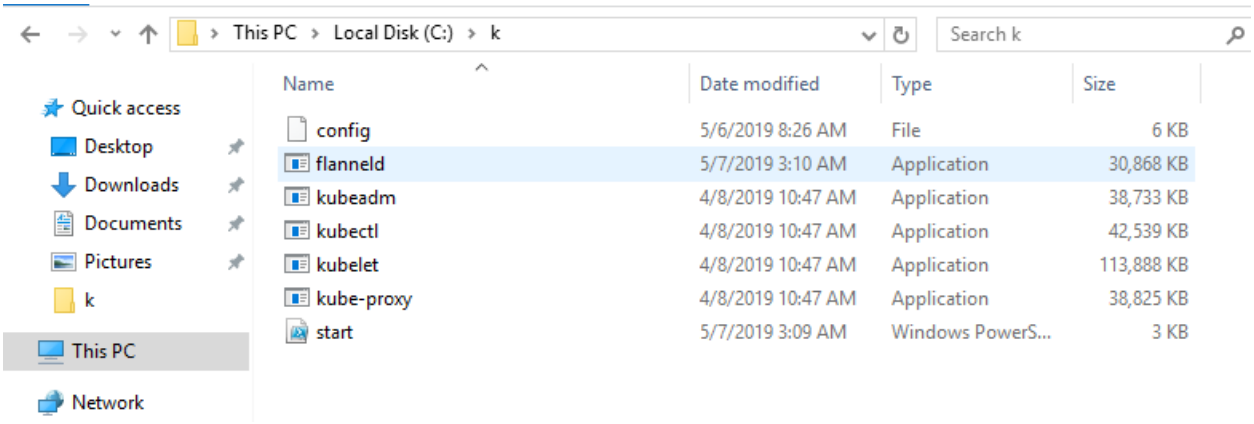
Step5: Download Kubernetes "kubelet" and "kube-proxy" binaries from below URL and place the binaries into "C:\k"

https://dl.k8s.io/v1.14.1/kubernetes-node-windows-amd64.tar.gz

Step6. Join the Windows node to the Flannel cluster, So First you need to download "start.ps1" and "flanneld.exe" from below URL to "C:\k".

https://github.com/Microsoft/SDN/blob/master/Kubernetes/flannel/start.ps1

https://github.com/coreos/flannel/releases/download/v0.11.0/flanneld.exe



Step7. Once Downloaded, Open PowerShell and Run the below command to Join Kubernetes Cluster with Flannel Network.

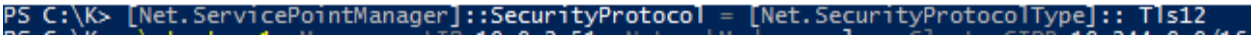
cd c:\k

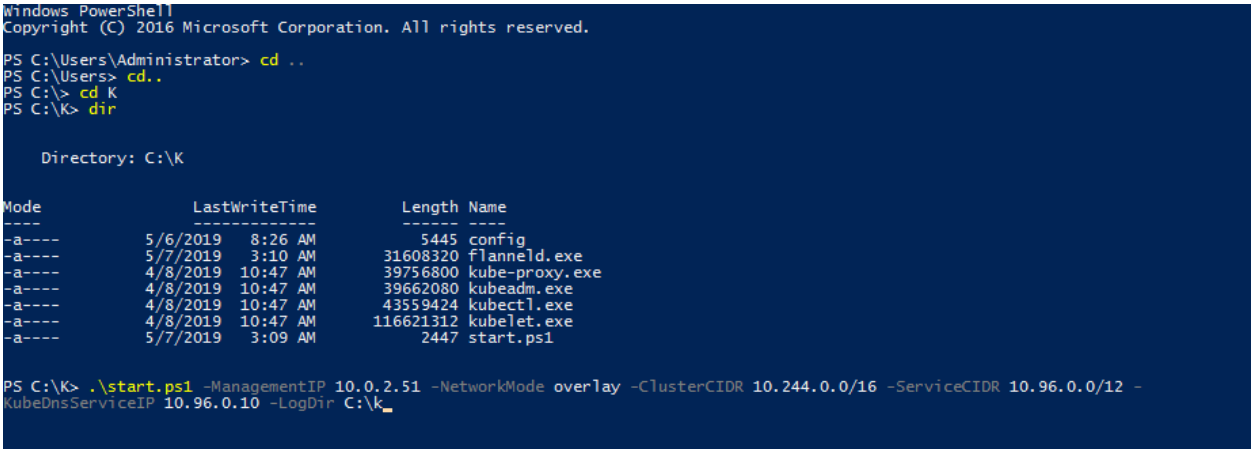
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

.\start.ps1 -ManagementIP <Windows Node IP> -NetworkMode overlay -ClusterCIDR <Cluster CIDR> -ServiceCIDR <Service CIDR> -KubeDnsServiceIP <Kube-dns Service IP> -LogDir <Log directory>

Example:

.\start.ps1 -ManagementIP 10.0.2.51 -NetworkMode overlay -ClusterCIDR 10.244.0.0/16 -ServiceCIDR 10.96.0.0/12 -KubeDnsServiceIP 10.96.0.10 -LogDir C:\k





8. Now you can view the Windows nodes in your cluster by running the following:

[root@master ~]# kubectl get nodes

```
[root@master ~]# kubectl get nodes
NAME                STATUS   ROLES    AGE     VERSION
master.example.com  Ready    master   84m     v1.14.1
node1.example.com   Ready    <none>   78m     v1.14.1
node2.example.com   Ready    <none>   78m     v1.14.1
win2k16-s1          Ready    <none>   4m33s   v1.14.1
[root@master ~]#
```

Congratulations, You have successfully added Windows Node in Kubernetes cluster.

You can follow the instructions from below URL:

"https://kubernetes.io/docs/setup/windows/user-guide-windows-nodes/" URL.

###This Document is created by Suresh Chandra (RHCA, Sr. DevOps Engineer) ###