How To Guide: Kubernetes for Windows Flannel (Host-Gateway)

Preface

1. For Linux, more detailed documentation that uses similar commands can be found here.
2. You are recommended to use **Ubuntu 16.04** and **Windows Server 2019** **Insider Builds** for these instructions.
    a. Other Linux distributions where the Master was setup using "kubeadm" should also work. Just skip ahead to the "Launch Flannel" section in this guide after initialization.
    b. **Windows Server, version 1803** will work with these instructions as well.
3. "$" means a command was run as regular user whereas "#" denotes a command that was run as root.

(Required for Windows VMs) Prepare guest VM(s)

Ensure MAC address spoofing/promiscuity mode and virtualization is enabled for the Windows container host VMs (guests). To achieve this, you should run the following as Administrator **on the VM host** server (example given for Hyper-V manager):

```
PS C:> Set-VMProcessor -VMName "<name>" -ExposeVirtualizationExtensions $true

PS C:> Get-VMNetworkAdapter -VMName "<name>" | Set-VMNetworkAdapter -MacAddressSpoofing On
```

All following commands in this how-to guide need to be executed on the container host machines (guests) directly.

K8s MASTER

Linux Ubuntu

To get to a root shell, you can use:

```
$ sudo –s
```

Make sure your machine is up to date:

```
# apt-get update && apt-get upgrade
```

Install Docker

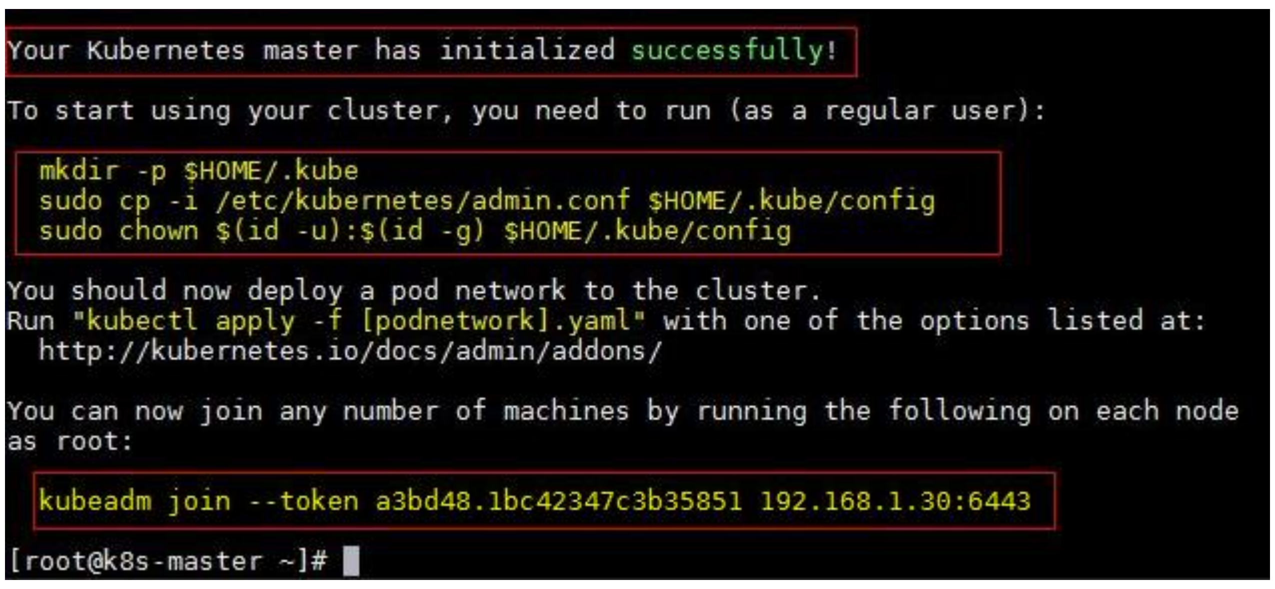
To get the most recent version, you can use these instructions for Docker installation.

Install K8s using kubeadm

```
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

# apt-get update && apt-get install -y kubelet kubeadm kubectl
```

`# nano /etc/fstab` (*remove a line referencing 'swap.img'* , if it exists)

```
# swapoff -a

# kubeadm init --pod-network-cidr=10.244.0.0/16
```



```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run (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:
  http://kubernetes.io/docs/admin/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join --token a3bd48.1bc42347c3b35851 192.168.1.30:6443

[root@k8s-master ~]#
```

1. Note down kubeadm join command. We will need this later. For example: "kubeadm join <Master_IP>:6443 --token <some_token> --discovery-token-ca-cert-hash <some_hash>"
2. Note down pod network CIDR (also known as cluster CIDR) being used (e.g. 10.244.0.0/16)

Finally, to use kubectl, **as a regular user**, run:

```
$ mkdir -p $HOME/.kube

$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Launch Flannel

Enable passing bridged IPv4 traffic to iptables chains:

```
# sysctl net.bridge.bridge-nf-call-iptables=1
```

Option 1: Deploy my example kube-flannel.yml (v0.9.1)

Option 2: Edit it yourself, if a newer Flannel version greater than v0.9.1 has been released:

```
$ wget https://raw.githubusercontent.com/coreos/flannel/<version_here>/Documentation/kube-flannel.yml
```

(Applies to option 2 only) Since the Flannel pods are only runnable on Linux, add a NodeSelector to `kube-flannel.yml` into `kube-flannel-ds` DaemonSet to only target Linux:

```
spec:
  template:
    spec:
      nodeSelector:
        beta.kubernetes.io/os: linux
```

Whichever option you chose, double-check that the type of network backend being used is set to "host-gw" and that the cluster CIDR (e.g. "10.244.0.0/16") conforms with what you put into the "`kubeadm init`" command when initializing the Master earlier.

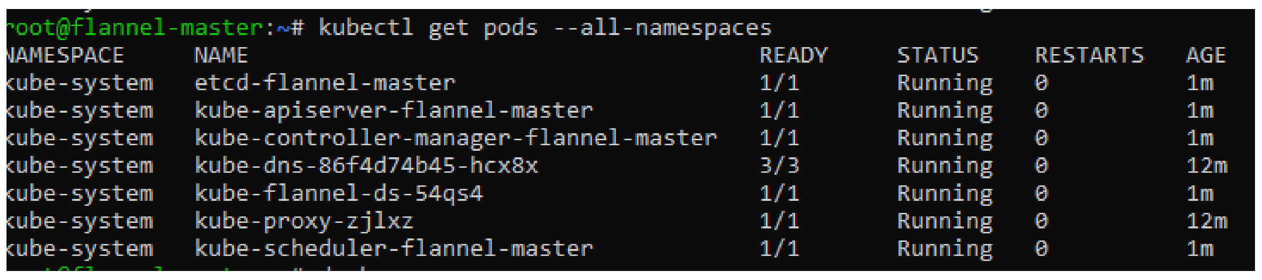
```
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "host-gw"
      }
    }
```

Launch Flannel using:

```
$ kubectl apply -f kube-flannel.yml
```

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

```
$ kubectl get pods --all-namespaces
```



```
oot@flannel-master:~# kubectl get pods --all-namespaces
AMESPACE     NAME                                     READY     STATUS    RESTARTS   AGE
ube-system   etcd-flannel-master                      1/1       Running   0          1m
ube-system   kube-apiserver-flannel-master            1/1       Running   0          1m
ube-system   kube-controller-manager-flannel-master   1/1       Running   0          1m
ube-system   kube-dns-86f4d74b45-hcx8x                3/3       Running   0          12m
ube-system   kube-flannel-ds-54qs4                    1/1       Running   0          1m
ube-system   kube-proxy-zjlxz                         1/1       Running   0          12m
ube-system   kube-scheduler-flannel-master            1/1       Running   0          1m
```

Edit Kube-Proxy DaemonSet

Confirm that the update strategy of DaemonSet is set to RollingUpdate:

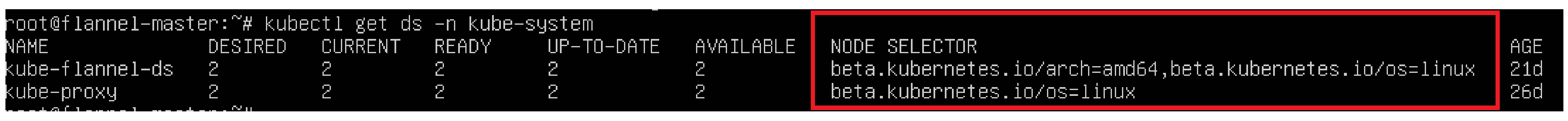
```
$ kubectl get ds/kube-proxy -o go-template='{{.spec.updateStrategy.type}}{{"\n"}}' --namespace=kube-system
```

Next, patch the DaemonSet by downloading this nodeSelector and apply it to only target Linux:

```
$ kubectl patch ds/kube-proxy --patch "$(cat node-selector-patch.yml)" -n=kube-system
```

Once successful, you should see "Node Selectors" of DaemonSets set to **beta.kubernetes.io/os=linux**

```
$ kubectl get ds -n kube-system
```



```
root@flannel-master:~# kubectl get ds -n kube-system
NAME              DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR                                                 AGE
kube-flannel-ds   2         2         2         2            2           beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux   21d
kube-proxy        2         2         2         2            2           beta.kubernetes.io/os=linux                                   26d
```

Later in the "Join Windows Node to Master" section we will launch kube-proxy as a process using a Powershell script.

Collect Information to join Workers

To summarize, the following information will be needed from the Kubernetes Master later:

- `Kubeadm join` command
  - For example, "kubeadm join <Master_IP>:6443 --token <some_token> --discovery-token-ca-cert-hash <some_hash>"
- Cluster CIDR defined during `kubeadm init`
  - For example, "10.244.0.0/16"
- Config file generated during `kubeadm init`
  - This can be found in one of either:
    - `/etc/kubernetes/admin.conf`
    - `$HOME/.kube/config`
- Service CIDR being used (can be found using `kubectl cluster-info dump | grep -i service-cluster-ip-range`)
  - For example, "10.96.0.0/12"
- Kube-DNS service VIP being used (can be found in "IP" field using `kubectl describe svc/kube-dns -n kube-system`)
  - For example, "10.96.0.10"

K8s WORKER

Linux Ubuntu

To get to a root shell, you can use:

```
$ sudo –s
```

Make sure your machine is up to date:

```
# apt-get update && apt-get upgrade
```

Install Docker

To get the most recent version, you can use these instructions for Docker installation.

Install K8s, kubeadm

```
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt-get update && apt-get install -y kubelet kubeadm kubectl
```

`# nano /etc/fstab` (*remove a line referencing 'swap.img'*)
`# swapoff -a`

Distribute config file from Master

As regular user, run:

```
$ mkdir -p $HOME/.kube
```

Copy (I used scp) config file ~/.kube/config from Master into `$HOME/.kube/config` on worker

```
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

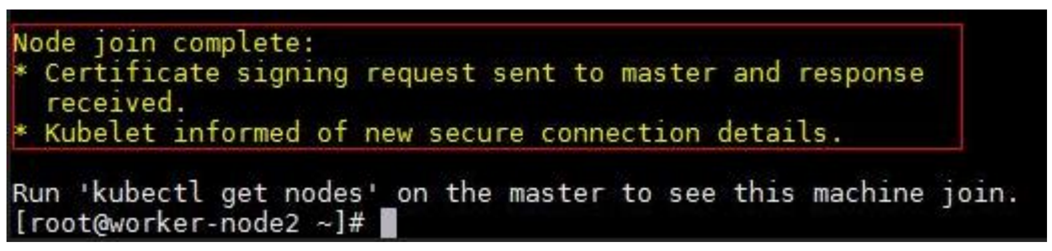
Join Node to Master

Enable passing bridged IPv4 traffic to iptables chains:

```
# sysctl net.bridge.bridge-nf-call-iptables=1
```

As root, run: kubeadm join command we noted down during Master setup. Eg:

```
# kubeadm join <Master_IP>:6443 --token <some_token> --discovery-token-ca-cert-hash <some_hash>
```



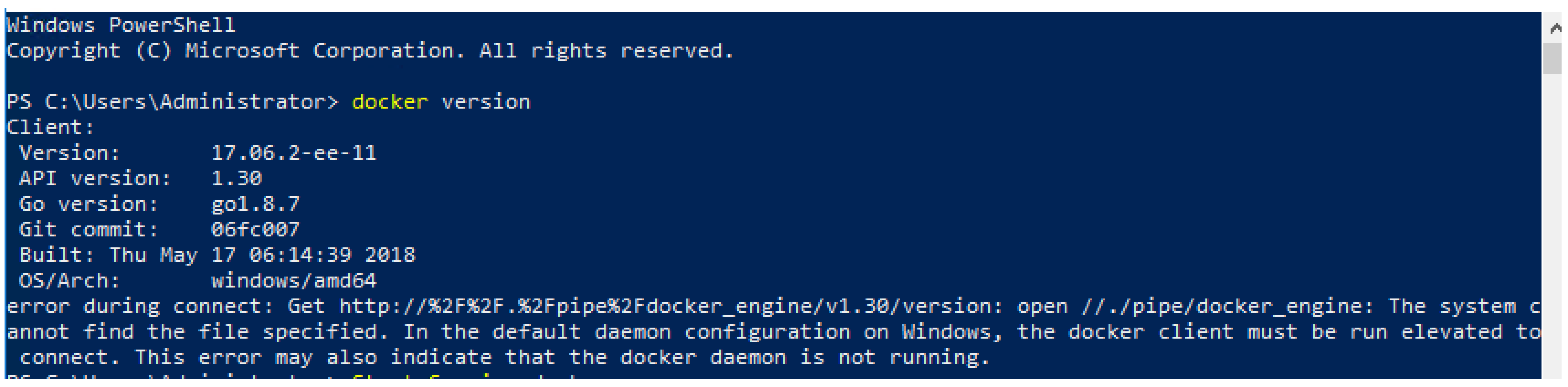
```
Node join complete:
* Certificate signing request sent to master and response
  received.
* Kubelet informed of new secure connection details.

Run 'kubectl get nodes' on the master to see this machine join.
[root@worker-node2 ~]#
```

Windows Server

Install Docker (requires reboot)

```
PS C:> Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
PS C:> Install-Package -Name Docker -ProviderName DockerMsftProvider
PS C:> Restart-Computer -Force
```

If after reboot you see the following error:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> docker version
Client:
 Version:       17.06.2-ee-11
 API version:   1.30
 Go version:    go1.8.7
 Git commit:    06fc007
 Built: Thu May 17 06:14:39 2018
 OS/Arch:       windows/amd64
error during connect: Get http://%2F%2F.%2Fpipe%2Fdocker_engine/v1.30/version: open //./pipe/docker_engine: The system c
annot find the file specified. In the default daemon configuration on Windows, the docker client must be run elevated to
 connect. This error may also indicate that the docker daemon is not running.
```

Then start the docker service:

```
PS C:> Start-Service docker
```

Prepare Infrastructure Image

I recommend you pick an image and double-check that it works for your specific build. Otherwise, your pods may later be stuck in "ContainerCreating" status indefinitely. There are three steps to this: pulling the image, tagging it as microsoft/nanoserver:latest, and running it. For Windows Server 2019 images simply adjust the docker pull command below to match your specific insider build #:

- microsoft/windowsservercore-insider
- microsoft/nanoserver-insider

For example, if you are on Windows Server 2019 build 17650, you can do the following:

Step 1: Pull the image for your build (either `mcr.microsoft.com/nanoserver-insider:<your_build>` or `microsoft.com/nanoserver-insider:<your_build>`)

```
PS C:> docker pull mcr.microsoft.com/nanoserver-insider:10.0.17650.1001
```

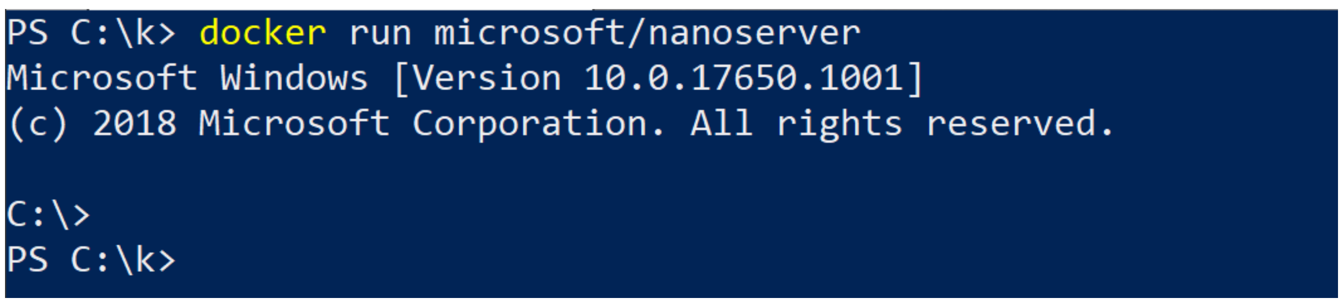
Step 2: Tag the Image as "microsoft/nanoserver:latest"

```
PS C:> docker tag mcr.microsoft.com/nanoserver-insider:10.0.17650.1001 microsoft/nanoserver:latest
```

Step 3: Double-check that the container runs on your computer:

```
PS C:> docker run microsoft/nanoserver:latest
```

You should see something like this:



```
PS C:\k> docker run microsoft/nanoserver
Microsoft Windows [Version 10.0.17650.1001]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\>
PS C:\k>
```

If you don't please see: matching container host version with container image.

For **Windows Server, version 1803** simply replace the docker pull command in step 1 with `docker pull microsoft/nanoserver:1803` and make sure you tag your image as `microsoft/nanoserver:latest` in step 2.

Download Flannel Launch Scripts and K8s Binaries

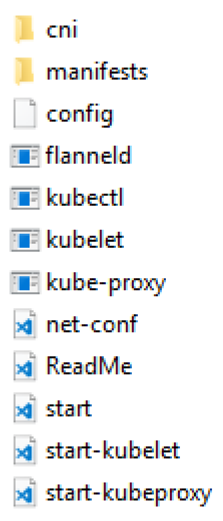
Create Kubernetes for Windows directory

```
PS C:> mkdir c:\k
```

Download the following **into** `c:\k`:

- Kubernetes binaries (kubelet.exe, kubectl.exe, kube-proxy.exe)
  - As of time of writing, latest stable release was v1.10.2. Check K8s releases and changelogs for updates.
- Files in Flannel Windows (host-gw) directory
  - **Ensure cluster CIDR (e.g. check "10.244.0.0/16") is correct in:**
    - `net-conf.json`
- Copy config file `$HOME/.kube/config` from master into `c:\k` directory on Windows worker.

Once you are done, the `c:\k` directory should look as follows:



```
cni
manifests
config
flanneld
kubectl
kubelet
kube-proxy
net-conf
ReadMe
start
start-kubelet
start-kubeproxy
```

Join Windows Node to Master

```
PS C:> cd c:\k
```

```
PS C:\k> .\start.ps1 -ManagementIP <Windows Node IP> -ClusterCIDR <Cluster CIDR> -ServiceCIDR <Service CIDR> -
KubeDnsServiceIP <Kube-dns Service IP>
```

- This script will download additional files such as flanneld executable and the Dockerfile used to prepare the `kubeletwin/pause` image (*and run those for you*).
- Wait a couple minutes and this script will launch Flannel, kubelet, kube-proxy, and join the node to the Master.
  - Kubelet and kube-proxy will be visible in two separate powershell windows.
- You noted down the arguments `<Cluster CIDR>`, `<Service CIDR>`, `<Kube-dns Service IP>` from the Linux master in section **"Collect information to join Workers"**
- There may be a few seconds of network outage while the new pod network is being created.
- Afterwards, double check that all the values look correct in: `cni/config/cni.conf`.
  - *You can edit this file on-the-fly, and the configuration will apply automatically to any newly deployed Kubernetes resources.*

Now you can view the joined Windows node using `kubectl get nodes` or try scheduling an example Windows service (don't forget to make sure the container image pulled matches your host OS).