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

## **Preface**

1. For Linux, more detailed documentation that uses similar commands to mine can be found [here](https://kubernetes.io/docs/setup/independent/install-kubeadm/).
2. You are recommended to use **Ubuntu 16.04** and **Windows Server 2019** [**Insider Builds**](https://blogs.windows.com/blog/tag/windows-insider-program/) for these instructions.
   1. Other Linux distributions where the Master was setup using “kubeadm” **should** work as well. Just skip ahead to the “Launch Flannel” section in this guide after “kubeadm” initialization.
      1. The exception this guide does **not** support ***as of time of writing*** is Ubuntu 18.04, as updated “kubeadm” binaries for Ubuntu 18.04 have not been released, necessitating workarounds.
   2. **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.

## **(Optional, only 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](https://docs.docker.com/install/linux/docker-ce/ubuntu/) 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

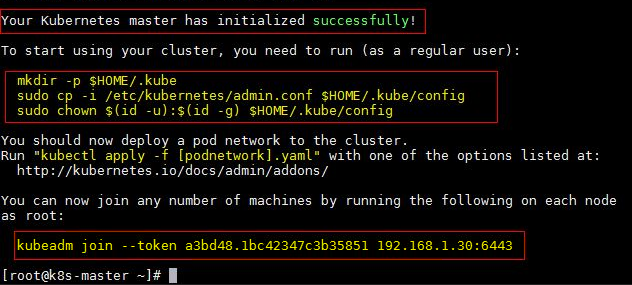
EOF

# 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



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](https://github.com/Microsoft/SDN/tree/master/Kubernetes/flannel/l2bridge/manifests/kube-flannel-example.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](https://raw.githubusercontent.com/coreos/flannel/%3cv0.9.1/Documentation/kube-flannel.yml)

(Applies to option 2 only) Since DaemonSets are not supported on Windows, 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"

}

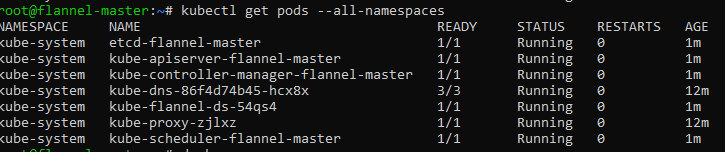
}

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



## **Edit Kube-Proxy DaemonSet**

Confirm that the update strategy of DaemonSet is set to [RollingUpdate](https://kubernetes.io/docs/tasks/manage-daemon/update-daemon-set/):

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

Next, patch the DaemonSet by downloading [this nodeSelector](https://github.com/Microsoft/SDN/tree/master/Kubernetes/flannel/l2bridge/manifests/node-selector-patch.yml) 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



## **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](https://docs.docker.com/install/linux/docker-ce/ubuntu/) 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

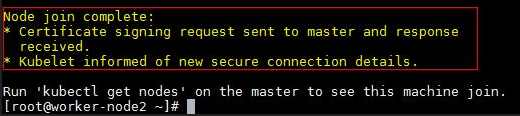
## **Join Node to Master**

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>



## **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:



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](https://hub.docker.com/r/microsoft/windowsservercore-insider/)
* [microsoft/nanoserver-insider](https://hub.docker.com/r/microsoft/nanoserver-insider/)

Step 1: Pull the image for your build. Try one of the following:

* mcr.microsoft.com/nanoserver-insider:<your\_build> or
* microsoft.com/nanoserver-insider:<your\_build>

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

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

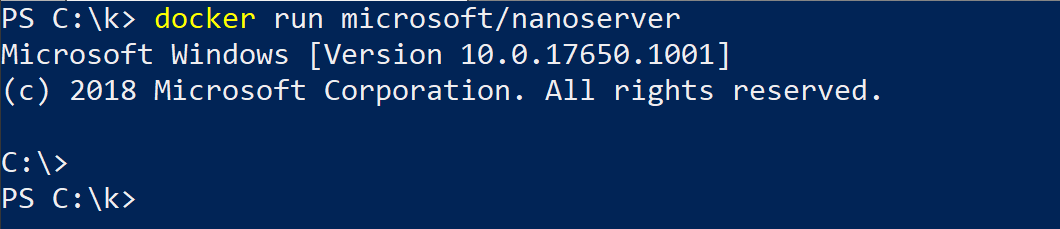
Step 2: Tag the Image as “microsoft/nanoserver:latest”

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:



If you don’t please see: [matching container host version with container image](https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/version-compatibility#matching-container-host-version-with-container-image-versions).

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**

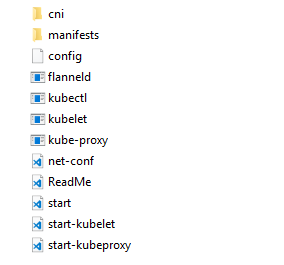
Create Kubernetes for Windows directory

PS C:> mkdir c:\k

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

* [Kubernetes binaries](https://storage.googleapis.com/kubernetes-release/release/v1.10.2/kubernetes-node-windows-amd64.tar.gz) (kubelet.exe, kubectl.exe, kube-proxy.exe)
  + As of time of writing, latest stable release was v1.10.2. Check [K8s releases](https://github.com/kubernetes/kubernetes/releases/) and changelogs for updates.
* [Files in Flannel Windows (host-gw) directory](https://github.com/Microsoft/SDN/tree/master/Kubernetes/flannel/l2bridge)
  + **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:



## **Join Windows Node to Master**

PS C:> cd c:\k

PS C:\k> .\start.ps1 -ManagementIP <Windows Host 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.
* 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](https://github.com/Microsoft/SDN/tree/master/Kubernetes/flannel/l2bridge/manifests/simpleweb.yml) (don’t forget to make sure the container image pulled matches your host OS).