

# Bonus Connect Windows Worker Node to Ubuntu Master Node

[Edition 1]

[Last Update 200827]

For any issues/help contact : [support@k21academy.com](mailto:support@k21academy.com)

**Note:** This is a bonus optional Guide only for Overview.

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>2</b>	<b>Documentation.....</b>	<b>4</b>
2.1	Kubernetes Documentation .....	4
2.2	Linux Commands and VIM Commands.....	4
<b>3</b>	<b>Pre-Requisite .....</b>	<b>5</b>
<b>4</b>	<b>Configure master Node.....</b>	<b>6</b>
<b>5</b>	<b>Create Windows Server 2019 machine on Azure.....</b>	<b>8</b>
5.1	Connecting to Windows Machine .....	8
<b>6</b>	<b>Networking Configuration In Master Node .....</b>	<b>9</b>
6.1	Configuring Flannel.....	9
<b>7</b>	<b>Install Docker On Windows Server On Azure cloud.....</b>	<b>12</b>
<b>8</b>	<b>Joining a Windows worker node.....</b>	<b>14</b>
<b>9</b>	<b>Summary.....</b>	<b>17</b>

## INTRODUCTION

You can use Kubernetes to run a mixture of Linux and Windows nodes, so you can mix Pods that run on Linux on with Pods that run on Windows. This page shows how to register Windows nodes to your cluster.

This guide Covers:

- Pre-Requisite
- Configure master Node
- Create Windows Server 2019 machine on Azure
- Networking Configuration In Master Node
- Install Docker On Windows Server On Azure cloud
- Joining a Windows worker node

## DOCUMENTATION

### 2.1 Kubernetes Documentation

1. Install Docker on Windows Server 2019  
<https://4sysops.com/archives/install-docker-on-windows-server-2019/>
2. Adding Windows nodes  
<https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/adding-windows-nodes/#joining-a-windows-worker-node>
3. Connect and sign on to an Azure virtual machine running Windows  
<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/connect-logon>

### 2.2 Linux Commands and VIM Commands

1. Basic Linux Commands  
<https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners>  
<https://www.hostinger.in/tutorials/linux-commands>
2. Basic VIM Commands  
<https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started>
3. Popular VIM Commands  
<https://www.keycdn.com/blog/vim-commands>

## PRE-REQUISITE

Your **Kubernetes server must be at or later than version 1.17.** To check the version, enter `kubectl version`.

- Obtain a **Windows Server 2019 license (or higher)** in order to configure the Windows node that hosts Windows containers.
- A Linux-based Kubernetes kubeadm cluster in which you have access to the control plane.

## CONFIGURE MASTER NODE

**Important Note:** Read Pre-requisite Because Performing this Guide.

First we need to configure Master Node For that please follow Guide **AG\_Bootstrap\_Kubernetes\_Cluster\_Using\_Kubeadm\_Guide\_ed\*\*** from portal

### 6. INSTALLING DOCKER, KUBEADM AND OTHER KUBECTL PACKAGES

**Note:** First perform Step 1 to 5 Steps on the Master node then repet same Step 1 to 5 on the both worker Nodes.

1. SSH to the virtual machine with the username and password you used while creating the VM

```
$ ssh root@publicipaddress
```

2. Switch to root user in case you aren't logged as root

```
$ sudo su
```

```
ubuntu@kubeadm-master:~$  
ubuntu@kubeadm-master:~$ sudo su  
root@kubeadm-master:~#
```

3. Install docker package using the following command

```
$ apt-get update && apt-get install -y docker.io
```

```
root@kubeadm-master:~# apt-get update && apt-get install -y docker.io  
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease  
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]  
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
```

Verify the docker version installed

```
$ docker --version
```

4. Install http-transport

```
$ apt-get update && apt-get install -y apt-transport-https
```

```
root@kubeadm-master:~#  
root@kubeadm-master:~# apt-get update && apt-get install -y apt-transport-https  
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
```

## 7 KUBEADM TO CREATE AND INITIALISE A CLUSTER

1. Initialising the control-plane node run the below command on the **(master node)**

```
$ kubeadm init
```

```
root@kubeadm-master:/home/ubuntu#  
root@kubeadm-master:/home/ubuntu# kubeadm init  
W0519 02:29:21.597345 2870 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io  
kubeproxy.config.k8s.io]  
[init] Using Kubernetes version: v1.18.2  
[preflight] Running pre-flight checks  
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please fol  
low the guide at https://kubernetes.io/docs/setup/cri/  
[preflight] Pulling images required for setting up a Kubernetes cluster
```

2. If cluster initialisation has succeeded you will see a cluster join command. Copy and save that for future reference. **This command would be used by the worker nodes to join the cluster**

**Note:** Best practice is to save this box command in note pad for future Reference.

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 10.0.0.4:6443 --token 9amey0.szuruforpi62ulj0 \  
--discovery-token-ca-cert-hash sha256:bb3e85d5f582591aeb24321e1e58d82eaddbddd0e217ee8fc160ae5635017989
```

## CREATE WINDOWS SERVER 2019 MACHINE ON AZURE

**Note: Create Windows Server 2019 Machine on Azure in same network in which your master Node deployed.**

### Create a virtual machine

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Resource group \* ⓘ  [Create new](#)

#### Instance details

Virtual machine name \* ⓘ    
✖ Virtual machine name must be unique in the current resource group.

Region \* ⓘ

Availability options ⓘ

Image \* ⓘ    
[Browse all public and private images](#)

Azure Spot instance ⓘ ☐ Yes ☒ No

Size \* ⓘ    
[Select size](#)

✖ 2 vCPUs are needed for this configuration, but only 0 vCPUs (of 4) remain for the Standard DSv3 Family vCPUs.

### 5.1 Connecting to Windows Machine

Note: Please Follow Official doc to connect to Azure Windows machine.

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/connect-logon>



## NETWORKING CONFIGURATION IN MASTER NODE

### 6.1 Configuring Flannel

1. Prepare Kubernetes control plane for Flannel

Some minor preparation is recommended on the Kubernetes control plane in our cluster. It is recommended to enable bridged IPv4 traffic to iptables chains when using Flannel. The following command must be run on all Linux nodes:

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

```
root@master-ubuntu:/home/ubuntu# sysctl net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
```

2. Download & configure Flannel for Linux

```
wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

```
root@master-ubuntu:/home/ubuntu# wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
--2020-08-26 16:41:03-- https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.208.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.208.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14662 (14K) [text/plain]
Saving to: 'kube-flannel.yml'

kube-flannel.yml  100%[=====>] 14.32K  --.-KB/s   in 0.007s

2020-08-26 16:41:03 (1.97 MB/s) - 'kube-flannel.yml' saved [14662/14662]
```

3. Modify the kube-flannel.yml

```
vim kube-flannel.yml
```

Modify the `net-conf.json` section of the flannel manifest in order to set the VNI to 4096 and the Port to 4789. It should look as follows:

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

```
    },
    {
      "type": "portmap",
      "capabilities": {
        "portMappings": true
      }
    }
  ]
}
net-conf.json: |
{
  "Network": "10.244.0.0/16",
  "Backend": {
    "Type": "vxlan"
    "VNI" : 4096,
    "Port": 4789
  }
}
---
apiVersion: apps/v1
```

#### 4. Apply the Flannel manifest and validate

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

```
root@master-ubuntu:/home/ubuntu# kubectl apply -f kube-flannel.yml
podsecuritypolicy.policy/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.apps/kube-flannel-ds-amd64 created
daemonset.apps/kube-flannel-ds-arm64 created
daemonset.apps/kube-flannel-ds-arm created
daemonset.apps/kube-flannel-ds-ppc64le created
daemonset.apps/kube-flannel-ds-s390x created
```

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

```
kubectl get pods -n kube-system
```

```
root@master-ubuntu:/home/ubuntu# kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-66bff467f8-j4x41	0/1	Pending	0	8m15s
coredns-66bff467f8-vkj8s	0/1	Pending	0	8m15s
etcd-master-ubuntu	1/1	Running	0	8m22s
kube-apiserver-master-ubuntu	1/1	Running	0	8m22s
kube-controller-manager-master-ubuntu	1/1	Running	0	8m22s
kube-flannel-ds-amd64-k57gk	1/1	Running	1	11s
kube-proxy-gxncv	1/1	Running	0	8m15s
kube-scheduler-master-ubuntu	1/1	Running	0	8m21s

## 6. Add Windows Flannel and kube-proxy DaemonSets

**Note:** Now you can add Windows-compatible versions of Flannel and kube-proxy. In order to ensure that you get a compatible version of kube-proxy, you'll need to substitute the tag of the image. The following example shows usage for Kubernetes v1.18.0, but you should adjust the version for your own deployment.

```
curl -L https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/kube-proxy.yml | sed 's/VERSION/<v1.18.0>/g' | kubectl apply -f - (Example)
curl -L https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/kube-proxy.yml | sed 's/VERSION/v1.18.8/g' | kubectl apply -f -
kubectl apply -f https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/flannel-overlay.yml
```

```
root@master-ubuntu:/home/ubuntu# kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.8", GitCommit:"9f2892aabb98fe339f3bd70e3c470144299398ace", GitTreeState:"clean", BuildDate:"2020-08-13T16:12:48Z", GoVersion:"go1.13.15", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.8", GitCommit:"9f2892aabb98fe339f3bd70e3c470144299398ace", GitTreeState:"clean", BuildDate:"2020-08-13T16:04:18Z", GoVersion:"go1.13.15", Compiler:"gc", Platform:"linux/amd64"}
root@master-ubuntu:/home/ubuntu# curl -L https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/kube-proxy.yml | sed 's/VERSION/v1.18.8/g' | kubectl apply -f -
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100  158    100   158    0     0   1136      0 --:--:-- --:--:-- --:--:--  1136
100   640    100   640    0     0  2191      0 --:--:-- --:--:-- --:--:--  2191
100  2844    100  2844    0     0  8056      0 --:--:-- --:--:-- --:--:--  8056
configmap/kube-proxy-windows created
daemonset.apps/kube-proxy-windows created
root@master-ubuntu:/home/ubuntu# kubectl apply -f https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/flannel-overlay.yml
configmap/kube-flannel-windows-cfg created
daemonset.apps/kube-flannel-ds-windows-amd64 created
```

## INSTALL DOCKER ON WINDOWS SERVER ON AZURE CLOUD

**Note:** To perform all the tasks please open **Windows power shell in Administrator role**.

1. Install the containers feature

**Note:** For containerization to work, you need to install the Windows container feature on the Windows container host. Use the command below to install the containers feature and **reboot the computer**.

Install-WindowsFeature containers -Restart

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

PS C:\windows\system32> Get-WindowsFeature containers

Display Name      Name      Install State
-----
[ ] Containers    Containers Available

PS C:\windows\system32> Install-WindowsFeature containers -Restart
```

2. Install Docker

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

```
PS C:\Users\windows> Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet
provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or
'C:\Users\windows\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by
running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install
and import the NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
```

3. We can use the commands below to view the installed package provider and the Docker package made available through it.

Get-PackageProvider -ListAvailable

```
PS C:\Users\windows> Get-PackageProvider -ListAvailable

Name      Version      DynamicOptions
-----
DockerMsftProvider 1.0.0.8      Update
msi        3.0.0.0      AdditionalArguments
msu        3.0.0.0
NuGet      2.8.5.208    Destination, ExcludeVersion, Scope, SkipDependencies, Headers, FilterOnTag...
PowerShellGet 1.0.0.1      PackageManagementProvider, Type, Scope, AllowClobber, SkipPublisherCheck, ...
Programs   3.0.0.0      IncludeWindowsInstaller, IncludeSystemComponent
```

- Next, we will use the PackageManagement PowerShell module command Install-Package to install the latest version of Docker.

```
Install-Package -Name docker -ProviderName DockerMsftProvider
```

```
PS C:\Users\windows> Install-Package -Name docker -ProviderName DockerMsftProvider

The package(s) come(s) from a package source that is not marked as trusted.
Are you sure you want to install software from 'DockerDefault'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y

Name                Version      Source        Summary
-----
Docker              19.03.11    DockerDefault Contains Docker EE for use with Windows Server.
```

- Docker verification

```
Start-Service Docker
```

Also, we can verify the Docker virtual network creation using the Docker command below. The default name of the bridge or switch in a Windows environment is NAT.

```
docker network ls
```

```
PS C:\Users\windows> Start-Service Docker
PS C:\Users\windows> docker network ls

NETWORK ID          NAME                DRIVER              SCOPE
ad767a49c619        nat                 nat                 local
b40b8b8b54f5        none               null                local
```

- we can run the Docker version command to check the details of our deployment setup. We can verify the Docker engine and client version from the command output.

```
Docker version
```

```
PS C:\Users\windows> docker version
Client: Docker Engine - Enterprise
Version:      19.03.11
API version:  1.40
Go version:   go1.13.11
Git commit:   0da829ac52
Built:        06/26/2020 17:20:46
OS/Arch:     windows/amd64
Experimental: false

Server: Docker Engine - Enterprise (Unlicensed - not for production workloads)
Engine:
Version:      19.03.11
API version:  1.40 (minimum version 1.24)
Go version:   go1.13.11
Git commit:   0da829ac52
Built:        06/26/2020 17:19:32
OS/Arch:     windows/amd64
Experimental: false
```

## JOINING A WINDOWS WORKER NODE

1. Install wins, kubelet, and kubeadm

```
curl.exe -LO https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/PrepareNode.ps1
```

```
PS C:\Users\windows> curl.exe -LO https://github.com/kubernetes-sigs/sig-windows-tools/releases/latest/download/PrepareNode.ps1
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100	159	100	159	0	0	0:00:01	726
100	641	100	641	0	0	0:00:01	641
100	4137	100	4137	0	0	0:00:01	4137

```
.\PrepareNode.ps1 -KubernetesVersion v1.18.0
```

```
PS C:\Users\windows> .\PrepareNode.ps1 -KubernetesVersion v1.18.0
Using Kubernetes version: v1.18.0

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          8/26/2020   5:30 PM             k
Downloading https://dl.k8s.io/v1.18.0/bin/windows/amd64/kubelet.exe to C:\k\kubelet.exe
Downloading https://dl.k8s.io/v1.18.0/bin/windows/amd64/kubeadm.exe to C:\k\kubeadm.exe
Downloading https://github.com/rancher/wins/releases/download/v0.0.4/wins.exe to C:\k\wins.exe
Creating Docker host network
381e4240db435718a98cae8196d0abc70b15d8b7356eba111968b8dda9de562a
Registering wins service

Directory: C:\var\log

Mode                LastWriteTime         Length Name
----                -
d-----          8/26/2020   5:30 PM             kubelet

Directory: C:\var\lib\kubelet\etc

Mode                LastWriteTime         Length Name
----                -
d-----          8/26/2020   5:30 PM             kubernetes

Directory: C:\etc\kubernetes

Mode                LastWriteTime         Length Name
----                -
d-----          8/26/2020   5:30 PM             pki

Directory: C:\var\lib\kubelet\etc\kubernetes

Mode                LastWriteTime         Length Name
```

```
Registering kubelet service
Service "kubelet" installed successfully!
Set parameter "DependOnService" for service "kubelet".

Caption          :
Description      :
ElementName      : kubelet
InstanceID       : kubelet
CommonName       :
PolicyKeywords   :
Enabled          : True
PolicyDecisionStrategy : 2
PolicyRoles      :
ConditionListType : 3
CreationClassName : MSFT|FW|FirewallRule|kubelet
ExecutionStrategy : 2
Mandatory        :
PolicyRuleName    :
Priority          :
RuleUsage         :
SequencedActions  : 3
SystemCreationClassName :
SystemName        :
Action            : Allow
Direction         : Inbound
DisplayGroup      :
DisplayName       : kubelet
EdgeTraversalPolicy : Block
EnforcementStatus  : NotApplicable
LocalOnlyMapping   : False
LooseSourceMapping : False
Owner             :
Platforms         : {}
PolicyStoreSource  : PersistentStore
PolicyStoreSourceType : Local
PrimaryStatus     : OK
Profiles          : 0
RuleGroup         :
Status            : The rule was parsed successfully from the store. (65536)
StatusCode        : 65536
PSComputerName    :
Name              : kubelet
ID                : kubelet
Group             :
Profile           : Any
Platform          : {}
LSM               : False
```

2. Run join command to join Worker node to master node:



```
PS C:\Users\windows> kubeadm join 10.0.0.6:6443 --token 01190n.yffrodjqvnjt7tgk --discovery-token-ca-cert-hash sha256:c2fb90daccfd71b187d7af3cc62557adb98766d5c37501682b32477696eac9c5
W0826 17:32:35.772600 1340 join.go:346] [preflight] WARNING: JoinControlPlane.controlPlane settings will be ignored when control-plane flag is not set.
[preflight] Running pre-flight checks
[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 19.03.11. Latest validated version: 19.03
[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'
W0826 17:32:38.605569 1340 utils.go:26] The recommended value for "authentication.x509.clientCAFile" in "KubeletConfiguration" is: \etc\kubernetes\pki\ca.crt; the provided value is: /etc/kubernetes/pki/ca.crt
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.18" 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] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiservert 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.

PS C:\Users\windows>
```

### 3. Verify If Worker node is connected or nor on master node

```
root@master-ubuntu:/home/ubuntu# kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
master-ubuntu       Ready     master   128m    v1.18.8
windows-worker      Ready     <none>   72m     v1.18.0
root@master-ubuntu:/home/ubuntu#
```

  

```
root@master-ubuntu:/home/ubuntu# kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE     VERSION    INTERNAL-IP    EXTERNAL-IP    OS-IMAGE                                     KERNEL-VERSION    CONTAINER-RUNTIME
master-ubuntu       Ready     master   129m    v1.18.8    10.0.0.6       <none>         Ubuntu 18.04.5 LTS                         5.3.0-1035-azure  docker://19.3.6
windows-worker      Ready     <none>   72m     v1.18.0    10.0.0.8       <none>         Windows Server 2019 Datacenter              10.0.17763.1397  docker://19.3.11
root@master-ubuntu:/home/ubuntu#
```



## SUMMARY

In this guide we Covered:

- Pre-Requisite
- Configure master Node
- Create Windows Server 2019 machine on Azure
- Networking Configuration In Master Node
- Install Docker On Windows Server On Azure cloud
- Joining a Windows worker node