**Kubeadm : Bring Your Own Nodes (BYON)**

This documents describes how to setup kubernetes from scratch on your own nodes, without using a managed service. This setup uses **kubeadm** to install and configure kubernetes cluster.

Compatibility

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

The below steps are applicable for the below mentioned OS

* OS: **Ubuntu**
* Version:**16.04**
* Codename:**Xenial**

**Base Setup (Skip if using vagrant)**

**Skip this step and scroll to Initializing Master if you have setup nodes with vagrant**

On all nodes which would be part of this cluster, you need to do the base setup as described in the following steps. To simplify this, you could also user script

**Create Kubernetes Repository**

We need to create a repository to download Kubernetes.

1. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
2. cat <<EOF > /etc/apt/sources.list.d/kubernetes.list
3. deb http://apt.kubernetes.io/ kubernetes-xenial main
4. EOF

**Installation of the packages**

We should update the machines before installing so that we can update the repository.

1. apt-get update -y

Installing all the packages with dependencies:

1. apt-get install -y docker.io kubelet kubeadm kubectl kubernetes-cni
2. rm -rf /var/lib/kubelet/\*

**Setup sysctl configs**

**//Start from here if you are using .sh file**

In order for many container networks to work, the following needs to be enabled on each node.

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

The above steps has to be followed in all the nodes.

**Initializing Master**

This tutorial assumes **kube-01** as the master and used kubeadm as a tool to install and setup the cluster. This section also assumes that you are using vagrant based setup provided along with this tutorial. If not, please update the IP address of the master accordingly.

To initialize master, run this on kube-01

1. kubeadm init --apiserver-advertise-address 192.168.12.10 --pod-network-cidr=192.168.0.0/16

**Initialization of the Nodes (Previously Minions)**

After master being initialized, it should display the command which could be used on all worker/nodes to join the k8s cluster.

e.g.

1. kubeadm join --token c04797.8db60f6b2c0dd078 192.168.12.10:6443 --discovery-token-ca-cert-hash sha256:88ebb5d5f7fdfcbbc3cde98690b1dea9d0f96de4a7e6bf69198172debca74cd0

Copy and paste it on all node.

**Troubleshooting Tips**

If you lose the join token, you could retrieve it using

1. kubeadm token list

On successfully joining the master, you should see output similar to following,

1. root@kube-03:~# kubeadm join --token c04797.8db60f6b2c0dd078 159.203.170.84:6443 --discovery-token-ca-cert-hash sha256:88ebb5d5f7fdfcbbc3cde98690b1dea9d0f96de4a7e6bf69198172debca74cd0
2. [kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
3. [preflight] Running pre-flight checks
4. [discovery] Trying to connect to API Server "159.203.170.84:6443"
5. [discovery] Created cluster-info discovery client, requesting info from "https://159.203.170.84:6443"
6. [discovery] Requesting info from "https://159.203.170.84:6443" again to validate TLS against the pinned public key
7. [discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "159.203.170.84:6443"
8. [discovery] Successfully established connection with API Server "159.203.170.84:6443"
9. [bootstrap] Detected server version: v1.8.2
10. [bootstrap] The server supports the Certificates API (certificates.k8s.io/v1beta1)
12. Node join complete:
13. \* Certificate signing request sent to master and response
14. received.
15. \* Kubelet informed of new secure connection details.
17. Run 'kubectl get nodes' on the master to see this machine join.

**Setup the admin client - Kubectl**

On Master Node

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

**Installing CNI with Weave**

Installing overlay network is necessary for the pods to communicate with each other across the hosts. It is necessary to do this before you try to deploy any applications to your cluster.

There are various overlay networking drivers available for kubernetes. Like flannel, calico. We are going to use **Weave Net**. Run on master

1. export kubever=$(kubectl version | base64 | tr -d '\n')
2. kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"

**Validating the Setup**

You could validate the status of this cluster, health of pods and whether all the components are up or not by using a few or all of the following commands.

To check if nodes are ready

1. kubectl get nodes
2. kubectl get cs // component status

[ Expected output ]

1. root@kube-01:~# kubectl get nodes
2. NAME STATUS ROLES AGE VERSION
3. kube-01 Ready master m v1.8.2
4. kube-02 Ready <none> m v1.8.2
5. kube-03 Ready <none> m v1.8.2

**Additional Status Commands**

1. kubectl version
3. kubectl cluster-info
5. kubectl get pods -n kube-system
7. kubectl get events

**It will take a few minutes to have the cluster up and running with all the services.**

**Possible Issues**

* Nodes are node in Ready status
* kube-dns is crashing constantly
* Some of the systems services are not up

Most of the times, kubernetes does self-heal, unless it’s an issue with system resources not being adequate. Upgrading resources or launching it on bigger capacity VM/servers solves it. However, if the issues persist, you could try following techniques,

Troubleshooting Tips

Check events

1. kubectl get events

Check Logs

1. kubectl get pods -n kube-system
3. [get the name of the pod which has a problem]
5. kubectl logs <pod> -n kube-system

**e.g.**

1. root@kube-01:~# kubectl logs kube-dns-545bc4bfd4-dh994 -n kube-system
2. Error from server (BadRequest): a container name must be specified for pod kube-dns-545bc4bfd4-dh994, choose one of:
3. [kubedns dnsmasq sidecar]

6. root@kube-01:~# kubectl logs kube-dns-545bc4bfd4-dh994 kubedns -n kube-system
7. I1106 14:41:15.542409 1 dns.go:48] version: 1.14.4-2-g5584e04
8. I1106 14:41:15.543487 1 server.go:70] Using
10. ....

***Enable Kubernetes Dashboard***

After the Pod networks is installed, We can install another add-on service which is Kubernetes Dashboard.

Installing Dashboard:

1. kubectl apply -f https://gist.githubusercontent.com/initcron/32ff89394c881414ea7ef7f4d3a1d499/raw/4863613585d05f9360321c7141cc32b8aa305605/kube-dashboard.yaml

This will create a pod for the Kubernetes Dashboard.

To access the Dashboard in the browser, run the below command

1. kubectl describe svc kubernetes-dashboard -n kube-system

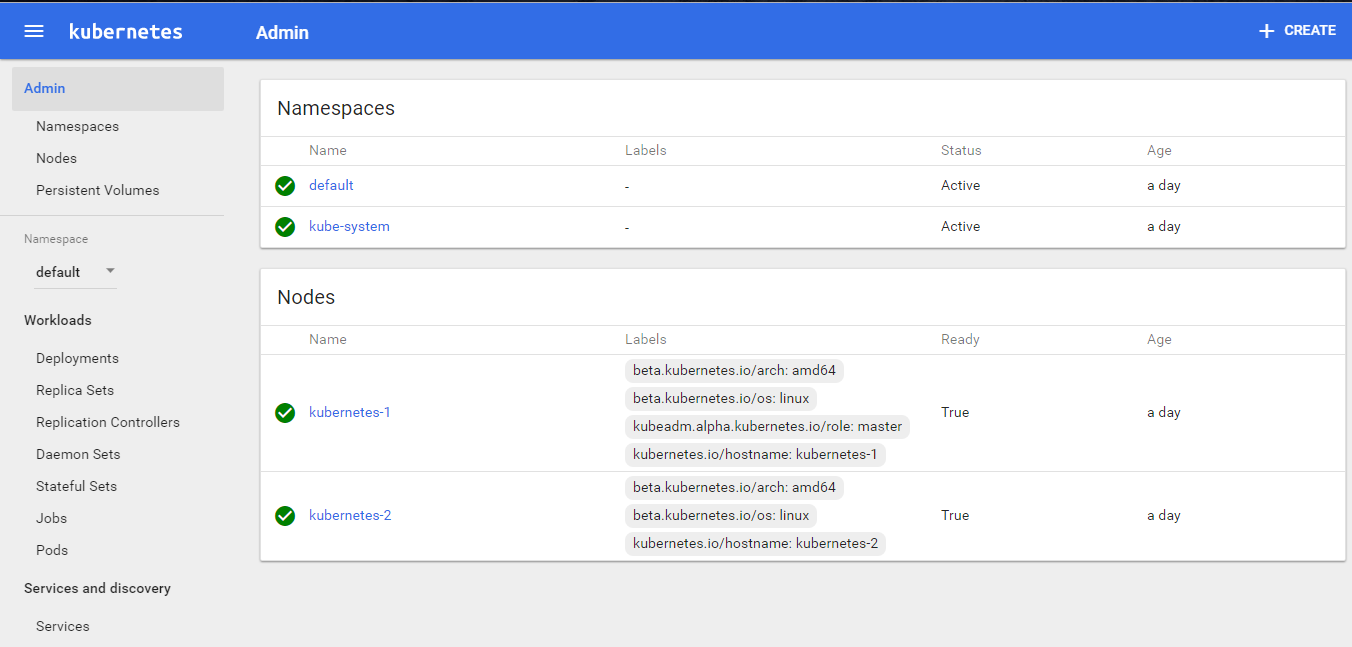
Sample output:

1. kubectl describe svc kubernetes-dashboard -n kube-system
2. Name: kubernetes-dashboard
3. Namespace: kube-system
4. Labels: app=kubernetes-dashboard
5. Selector: app=kubernetes-dashboard
6. Type: NodePort
7. IP: 10.98.148.82
8. Port: <unset> 80/TCP
9. NodePort: <unset> 31000/TCP
10. Endpoints: 10.40.0.1:9090
11. Session Affinity: None

Now check for the node port, here it is 31000, and go to the browser,

1. masterip:32756

The Dashboard Looks like:



**Check out the supporting code**

1. git clone https://github.com/chandanchanchal/k8s-code.git