**Install Kubernetes Cluster using kubeadm**

Documentation to set up a Kubernetes cluster on **CentOS 7** Virtual machines. Setting up cluster with one master node and two worker nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Role | Host | Ip | OS | RAM | CPU |
| Master | HYD-DEVOPSCOS03.HAICPM.COM | 192.168.136.124 | CentOS 7.7.1908 | 4 GB | 2 |
| Worker | HYD-DEVOPSCOS04.HAICPM.COM | 192.168.136.125 | CentOS 7.7.1908 | 4 GB | 2 |
| worker |  |  |  |  |  |
|  |  |  |  |  |  |

**On both master and worker**

### **Pre-requisites**

##### **Install, enable and start docker service**

##### **Disable SELinux**

|  |
| --- |
| setenforce 0  sed -i --follow-symlinks 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/sysconfig/selinux |

##### **Disable Firewall**

|  |
| --- |
| systemctl disable firewalld  systemctl stop firewalld |

##### **Disable swap**

|  |
| --- |
| sed -i '/swap/d' /etc/fstab  swapoff -a |

##### Update sysctl settings for Kubernetes networking

|  |
| --- |
| cat >>/etc/sysctl.d/kubernetes.conf<<EOF  net.bridge.bridge-nf-call-ip6tables = 1  net.bridge.bridge-nf-call-iptables = 1  EOF  sysctl --system |

### **Kubernetes Setup**

### **Add yum repository**

|  |
| --- |
| cat >>/etc/yum.repos.d/kubernetes.repo<<EOF  [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  EOF |

##### **Install Kubernetes**

|  |
| --- |
| yum install -y kubeadm kubelet kubectl  yum -y install nfs-common cifs-utils nfs4-acl-tools nfs-utils ( check dynamic NFS provisionar) |

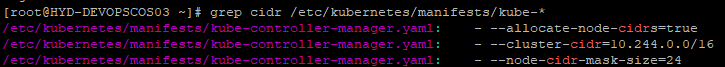
##### **Enable and Start kubelet service**

|  |
| --- |
| systemctl enable kubelet  systemctl start kubelet |

## On master

##### **Initialize Kubernetes Cluster**

|  |
| --- |
| kubeadm init --apiserver-advertise-address=192.168.136.124 --pod-network-cidr=10.244.0.0/16 |



**Create new user**

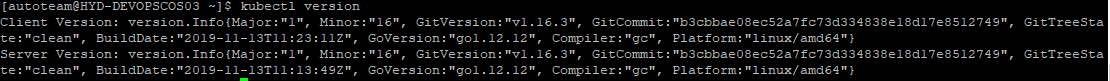
|  |
| --- |
| useradd -d /opt/autoteam autoteam  usermod -aG docker autoteam  passwd autoteam |

Make autoteam as Sudo user:

|  |
| --- |
| cat /etc/sudoers | grep autoteam  %autoteam ALL=(ALL:ALL) ALL |

To be able to use kubectl command to connect and interact with the cluster, the user needs kube config file.

|  |
| --- |
| mkdir /opt/autoteam/.kube  cp -r /etc/kubernetes/admin.conf /opt/autoteam/config  chown -R autoteam:autoteam /opt/autoteam/.kube |



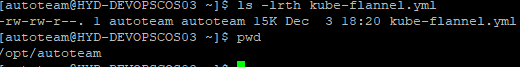
## Kubernetes networking model

##### **Deploy Flannel network**

<https://github.com/coreos/flannel/blob/master/Documentation/kube-flannel.yml>

For Kubernetes v1.7+

 kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



Kubectl apply -f kube-flannel.yml

**Custer Join Command**

|  |
| --- |
| kubeadm token create --print-join-command |

**On Worker**

##### **Join the cluster**

Use the output from **kubeadm token create** command in previous step from the master server and run here

## Verifying the cluster

##### **Get Nodes status**

|  |
| --- |
| kubectl get nodes |

##### **Get component status**

|  |
| --- |
| kubectl get cs |
| kubectl get cs -o yaml |

**Application Controllers**

* Deployment/statefulset/deamonset
* Configmap
* Services(clusterport,nodeport,load balancer)
* Ingress
* Horizontal Pod autotscaler

**Deployment**

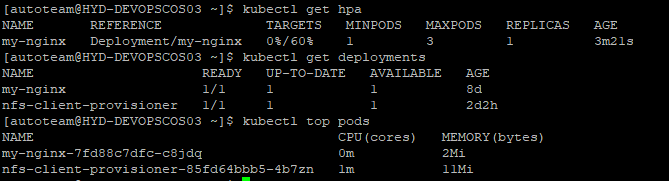
|  |
| --- |
| apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2  kind: Deployment  metadata:  name: my-nginx  spec:  selector:  matchLabels:  app: my-nginx  replicas: 2 # tells deployment to run 2 pods matching the template  template:  metadata:  labels:  app: my-nginx  spec:  containers:  - name: my-nginx  image: nginx:1.7.9  resources:  requests:  # memory: "${vis\_memory}"  cpu: "100m"  limits:  # memory: "${vis\_memory\_limit}"  cpu: "100m"  ports:  - containerPort: 80  # volumeMounts:  # - mountPath: /usr/share/nginx/html  # name: test-volume  # volumes:  # - name: test-volume  # hostPath:  # # directory location on host,/usr/share/nginx/html/index.html in container  # path: /opt/autoteam |

**Svc.yml**

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: my-nginx  labels:  app: my-nginx  spec:  type: NodePort  ports:  - port: 8080  targetPort: 80  protocol: TCP  name: http  selector:  app: my-nginx |

**Horizontal pod autoscaler**

|  |
| --- |
| >kubectl autoscale deployment my-nginx --cpu-percent=60 --min=1 --max=3  kubectl describe hpa my-nginx |



If “Kubectl top pods” doesn’t give valid out put, one need to configure metrics-server(refer below for steps) or Heapster (deprecated)

**Ingress**

The concept of Ingress is split in two parts:

* The [Ingress Controller](https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/), it’s some kind of wrapper for an HTTP proxy,
* Ingress resources/rules that expose HTTP and HTTPS routes from outside the cluster to services within the cluster, depending on traffic rules.

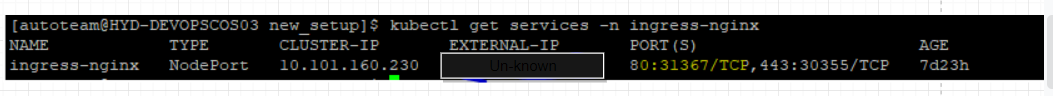
**Ingress Controller Installation**

Install a nginx ingress controller

> kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/mandatory.yaml

Ingress Controller is started, but not yet accessible externally from our K8s cluster. We need to create a NodePort Service to expose it to the outside world.

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/provider/baremetal/service-nodeport.yaml>



External-IP - <un-known>

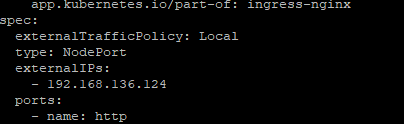
Access with any node Ip and port 31367 we will see a NGINX 404 page

Our **NGINX controller** is responding!

**Hack for local Dev Setup**

As we do not have any public Ip to be exposed as external-ip for the nginx ingress service controller

Update the external ip section in “**service-nodeport.yaml**”

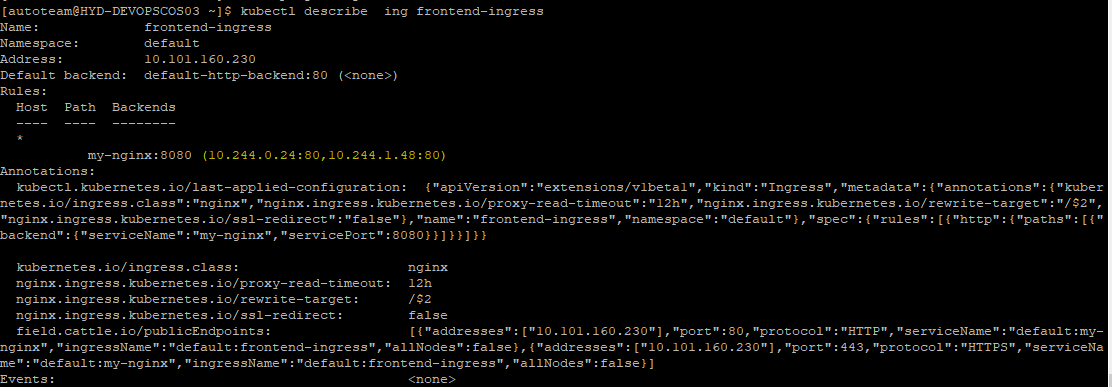




**Create Ingress to public expose application**

frontend-ingress.yaml

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  annotations:  kubernetes.io/ingress.class: "nginx"  nginx.ingress.kubernetes.io/rewrite-target: /$2  nginx.ingress.kubernetes.io/proxy-read-timeout: "12h"  nginx.ingress.kubernetes.io/ssl-redirect: "false"  name: frontend-ingress  namespace: default  spec:  rules:  # - host: hyd-devopscos04.haicpm.com  - http:  paths:  - backend:  serviceName: my-nginx  servicePort: 8080 |





AS we added External Ip address to Ingress controller node port service now we can access ”my-nginx” application using external Ip

Reference :



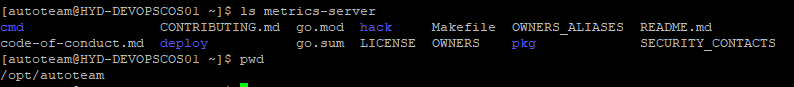
# **Kubernetes Metrics Server**

# Resource usage metrics, such as container CPU and memory usage, are available in Kubernetes through the Metrics API. These metrics can be either accessed directly by user, for example by using kubectl top command, or used by a controller in the cluster, e.g. Horizontal Pod Autoscaler, to make decisions.

* Through the **Metrics API** you can get the amount of resource currently used by a given node or a given pod
* **Metric server** is a cluster-wide aggregator of resource usage data

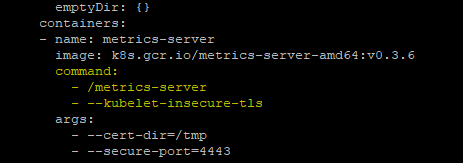


Refernce :

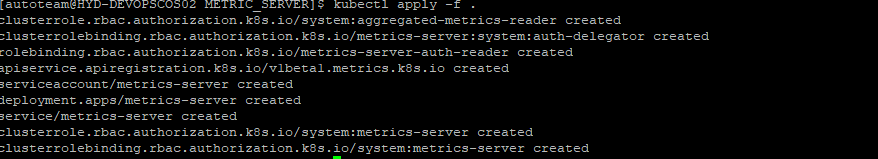


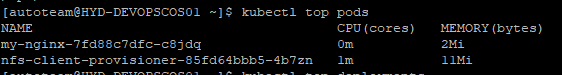
* git clone <https://github.com/kubernetes-sigs/metrics-server>
* >cd metrics-server/deploy/1.8+/

Update “metrics-server-deployment.yaml” as shown below to accept insecure tls



* kubectl create -f .



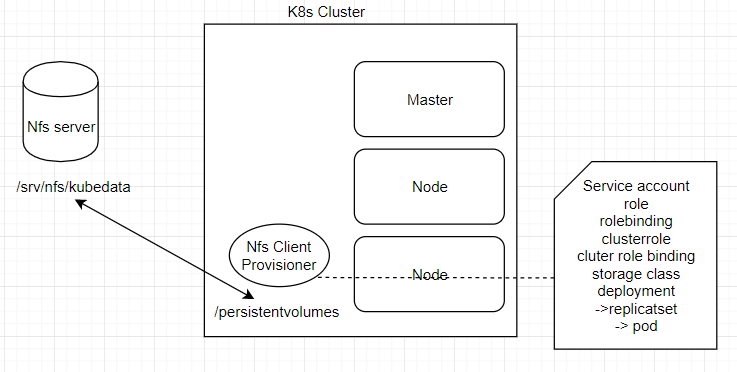


# **Dynamically provision NFS persistent Volumes in Kubernetes**

Kubernetes has **persistent volumes**. Persistent volumes are long-term storage in your Kubernetes cluster. Persistent volumes exist beyond containers, pods, and nodes .A pod uses a persistent volume claim to to get read and write access to the persistent volume.

**NFS Client Provisioner** creates a pod in k8’s cluster, which mounts /srv/nfs/kubedata to /persistentvolumes

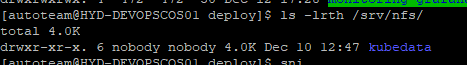
Also create a storage class and point to **NFS Client Provisioner pod**



**NFS-server setup**

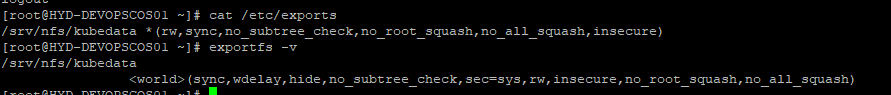
>git clone <https://github.com/kubernetes-incubator/external-storage>

* >cd external-storage/nfs-client/deploy
* > mkdir –p /srv/nfs/kubedata ( can be on any external Machine)
* >sudo chown nobody: /srv/nfs/kubedata



* sudo yum install nfs-utils ( yum -y install nfs-common cifs-utils nfs4-acl-tools nfs-utils on all nodes)
* systemctl enable nfs-server
* systemctl start nfs-server (assume firewall disable)
* cat /etc/exports

/srv/nfs/kubedata \*(rw,sync,no\_subtree\_check,no\_root\_squash,no\_all\_squash,insecure)



**On any worker node:**

* mount -t nfs 192.168.136.15:/srv/nfs/kubedata /mnt

(any issue :> yum -y install nfs-common cifs-utils nfs4-acl-tools nfs-utils on all nodes)

* >mount | grep kubedata
* >umount /mnt

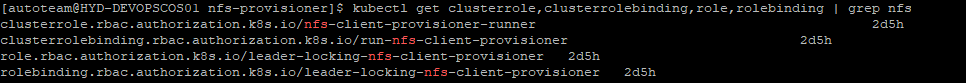
Verification successful

**Deploy client provisioner pod**

* git clone <https://github.com/justmeandopensource/kubernetes>
* cd Kubernetes/yamls/nfs-provisioner

->rbac.yaml – creates service account,role,rolebinding,clusterrole,clusterrole binding

* Kubectl create -f rbac.yml



->class.yaml – creates storage class

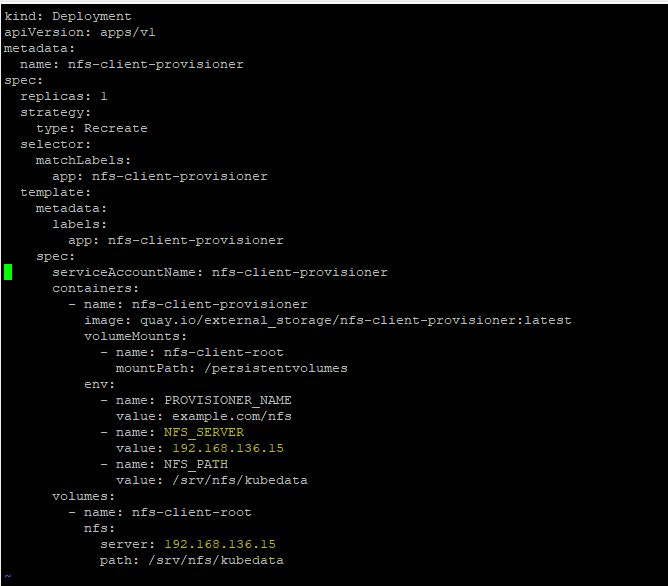
* kubectl create -f class.yml

( name will be there about storage class, it should be mentioned in persistent storage claim)

->deployment.yaml – create deploymet,replicatser

Update NfS server Ip as shown below

* kubectl create -f deployment.yml
* kubectl describe pod nfs-client-provisioner-85fd64bbb5-4b7zn

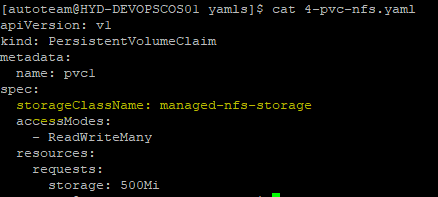


* kubectl get pv,pvc

no resource found

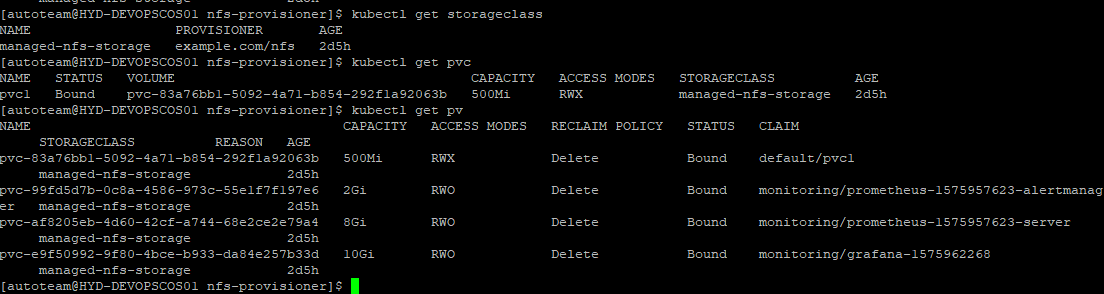
**Create persistent Volume claim(pvc)**

Storage class name show be updated as created earlier



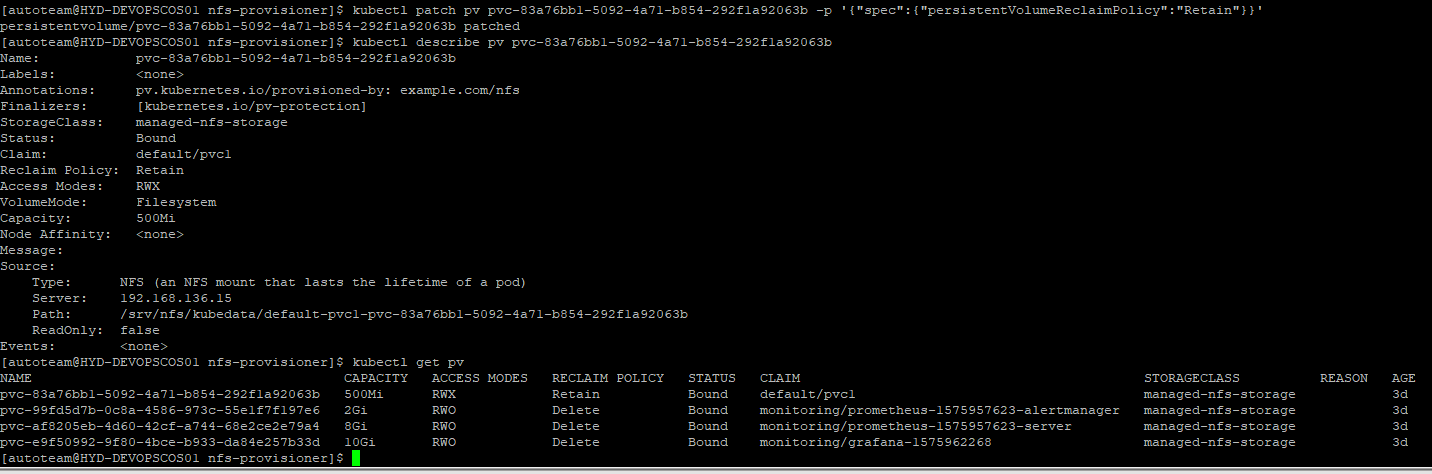
* Kubectl apply -f 4-pvc-nfs.yaml

Now pvc will trigger NFS provisioner and will create pv intern



**Change Recalim policy to Retain**

$ kubectl patch pv <pv name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'



**Create pod and consume persistent Volume**

deployment.yaml

|  |
| --- |
| apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2  kind: Deployment  metadata:  name: my-nginx  spec:  selector:  matchLabels:  app: my-nginx  replicas: 2 # tells deployment to run 2 pods matching the template  template:  metadata:  labels:  app: my-nginx  spec:  volumes:  - name: host-volume  persistentVolumeClaim:  claimName: pvc1  containers:  - name: my-nginx  image: nginx:1.7.9  resources:  requests:  # memory: "${vis\_memory}"  cpu: "10m"  limits:  # memory: "${vis\_memory\_limit}"  cpu: "100m"  ports:  - containerPort: 80  volumeMounts:  - mountPath: /var/log/nginx/  name: host-volume |

# <https://kubernetes.io/docs/concepts/services-networking/add-entries-to-pod-etc-hosts-with-host-aliases/>

apiVersion: v1

kind: Pod

metadata:

name: hostaliases-pod

spec:

restartPolicy: Never

hostAliases:

- ip: "127.0.0.1"

hostnames:

- "foo.local"

- "bar.local"

- ip: "10.1.2.3"

hostnames:

- "foo.remote"

- "bar.remote"

containers:

- name: cat-hosts

image: busybox

command:

- cat

args:

- "/etc/hosts"

# **Monitoring using Prometheus and Grafana**