Kubernetes Configuration Guide

Electronic Service Agent for Multivendor Systems

November 2024

Table of Contents

Introduction

Kubernetes is an open-source container orchestration engine for automating deployment, scaling, and management of containerized applications. **IBM Electronic Service Agent** can now be installed and run on Kubernetes. To run ESA on Kubernetes, you must first create a Kubernetes cluster and deploy ESA.

After successful deployment of Kubernetes cluster and ESA installation, you can access ESA from your local host and continuously monitor, collect, and submit hardware problem information to the IBM Electronic Support website. IBM Electronic Service Agent can also routinely collect and submit hardware, software and system configuration information, which might help IBM Support in diagnosing problems.

ESA Kubernetes Setup

The following is the list of prerequisites for Kubernetes setup:

- Docker
- Kubectl
- Kind - This is used to test Kubernetes cluster on one single server. In production we need to configure nodes on different servers.
- Nginx - Nginx is used to for reverse proxy, load balancing and caching. Also verify the internal IP addresses of the worker nodes in conf.d/lb.conf file.

ESA Kubernetes setup includes the following steps:

- Create a Kubernetes cluster
- Configure NFS
- Create a persistent volume
- Create a persistent volume claim
- Create ESA deployment
- Create ESA service
- Use SW load balancer

Create a Kubernetes Cluster

To create a Kubernetes cluster, you need to first create the cluster.yaml file and then execute the **`create cluster`** command as shown below.

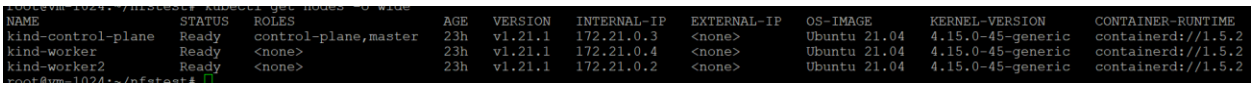
The command below creates a Kubernetes cluster with one 1 control plane node and 2 worker nodes:

`cluster.yaml`

```
--------------
# this config file contains all config fields with comments
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
networking:
  apiServerAddress: 9.114.195.16
  apiServerPort: 6443
# patch the generated kubeadm config with some extra settings
kubeadmConfigPatches:
```

```
- |
  apiVersion: kubelet.config.k8s.io/v1beta1
  kind: KubeletConfiguration
  evictionHard:
    nodefs.available: "0%"
# patch it further using a JSON 6902 patch
kubeadmConfigPatchesJSON6902:
- group: kubeadm.k8s.io
  version: v1beta3
  kind: ClusterConfiguration
  patch: |
    - op: add
      path: /apiServer/certSANs/-
      value: my-hostname
# 1 control plane node and 2 workers
nodes:
# the control plane node config
- role: control-plane
  kubeadmConfigPatches:
  extraPortMappings:
  - containerPort: 5024
    hostPort: 5024
    protocol: TCP
# the two workers
- role: worker
- role: worker

kind create cluster --config cluster.yaml # to create a cluster
kubectl get nodes -o wide # to check the cluster
```



NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
kind-control-plane	Ready	control-plane,master	23h	v1.21.1	172.21.0.3	<none>	Ubuntu 21.04	4.15.0-45-generic	containerd://1.5.2
kind-worker	Ready	<none>	23h	v1.21.1	172.21.0.4	<none>	Ubuntu 21.04	4.15.0-45-generic	containerd://1.5.2
kind-worker2	Ready	<none>	23h	v1.21.1	172.21.0.2	<none>	Ubuntu 21.04	4.15.0-45-generic	containerd://1.5.2

Configure NFS

```
apt install nfs-kernel-server
mkdir -p /mnt/nfs_share
chown -R nobody:nogroup /mnt/nfs_share/
chmod 777 /mnt/nfs_share/
mkdir -p /nfsstorage

vi /etc/exports and add the below lines
/nfsstorage *(rw,async,all_squash)
/mnt/nfs_share *(rw,sync,all_squash)

systemctl restart nfs-kernel-server
```

Create mount on all worker nodes

```
docker container ls #will display the node details
docker container exec -it <containerId>
mkdir -p /mnt/nfs_share
mount <HOSTIP>:/mnt/nfs_share /mnt/nfs_share
```

Create a Persistent Volume

Create a persistent volume as shown below and provide "**accessmode**" as "**ReadWriteMany**"

Note: The volumeName given in PV config would be used in creating a PVC (persistent volume claim)

pv-nfs.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nfs-3
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  storageClassName: standard
  nfs:
    path: /mnt/nfs_share
    server: <HOSTIP> # change the IP to NFS server IP
    readOnly: false

kubectl apply -f pv-nfs.yaml
```

Create a Persistent Volume Claim

pvc-nfs.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nfs-3
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
```

```
      storage: 1Gi
  volumeName: pv-nfs-3

kubectl apply -f pvc-nfs.yaml
```

Create ESA deployment

1. Check if there are any existing ESA deployments by the command:
   kubectl get deployments
2. Make sure that there no files in **mnt/nfs_share**.
3. Delete the files by using the command:
   rm –rf *
4. Verify the deployment file details in '**new-esa-deployment-withpvc-nfs.yaml' file.**
5. Apply the configuration details for ESA deployment by using the command:
   kubectl apply -f new-esa-deployment-withpvc-nfs.yaml

esa-deployment-withpvc-nfs.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: esa-tests
spec:
  replicas: 1
  selector:
    matchLabels:
      app: esa
  template:
    metadata:
      labels:
        app: esa
    spec:
      topologySpreadConstraints:
        - maxSkew: 1
          topologyKey: kubernetes.io/hostname
          whenUnsatisfiable: DoNotSchedule
          labelSelector:
            matchLabels:
              app: esa
      volumes:
        - name: esa-data
          persistentVolumeClaim:
            claimName: pvc-nfs-3
```

```
      containers:
      - name: esa
        image: docker.io/ibmcom/ibmesa:1.0.9
        env:
        - name: HOST
          value: "<HOSTIP>"
        - name: IP
          value: "<HOSTIP>"
        - name: CLUSTER_ENV
          value: "TRUE"
        imagePullPolicy: Never
        ports:
        - name: port5024
          containerPort: 5024
        - name: port162
          containerPort: 162
        volumeMounts:
        - mountPath:
opt/ibm/esa/workspace/.metadata/.plugins/com.ibm.esa.core/config
          name: esa-data
          subPath: workspace-core
        - mountPath:
/opt/ibm/esa/workspace/.metadata/.plugins/com.ibm.esa.central
          name: esa-data
          subPath: workspace-central
        - mountPath: /opt/ibm/esa/locks
          name: esa-data
          subPath: locks
        - mountPath: /opt/ibm/esa/ecc/data/cred
          name: esa-data
          subPath: ecc
        - mountPath: /opt/ibm/esa/workspace/cache
          name: esa-data
          subPath: esa-cache
        - mountPath: /opt/ibm/esa/conf/common
          name: esa-data
          subPath: esa-common-conf
        - mountPath: /opt/ibm/esaclient/conf/common
          name: esa-data
          subPath: client-common-conf
        - mountPath: /opt/ibm/esaclient/data/common
          name: esa-data
          subPath: client-common-data
        - mountPath: /opt/ibm/esa/workspace/h2db
          name: esa-data
          subPath: esa-db
        - mountPath: /opt/ibm/esa/workspace/data
           name: esa-data
```

```
            subPath: esa-workspace-data

kubectl apply -f esa-deployment-withpvc-nfs.yaml
```

Check the ESA deployment file details by using the command:

```
kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
esa-tests   1/1     1            1           2d20h
```

Create ESA Service

1. To create an ESA Service, use the below command:

esa-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: esa-service
spec:
  selector:
    app: esa
  ports:
  - name: http
    port: 5024
    targetPort: 5024
  - name: udp
    port: 162
    targetPort: 162
    protocol: UDP
  type: NodePort
  externalTrafficPolicy: Local
```

2. To apply the ESA service configurations, use the command:

   kubectl apply -f esa-service.yaml

3. Verify the ESA service by using the command:

   kubectl get services

4. Check if the ESA application is up by using the command:

   curl -k https://<worker IP :<SERVICE PORT>/esa/login.html

The < SERVICE PORT> could be fetched from output of the command - **kubectl get services -o wide**

30612 is the < SERVICE PORT> in the below sample output.

```
NAME         TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)         AGE SELECTOR
esa-service NodePort  10.96.168.249 <none>        5024:30612/TCP 23h app=esa
Kubernetes  ClusterIP 10.96.0.1     <none>        443/TCP         23h <none>
```

Use a SW Load Balancer

Install nginx

Nginx is used to for reverse proxy, load balancing and caching. Also verify the internal IP addresses of the worker nodes in conf.d/lb.conf file.

install nginx

/etc/nginx/nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
include /etc/nginx/conf.d/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}
```

/etc/nginx/conf.d/lb.conf

```
stream {
    upstream stream_backend {
        least_conn;
        server 172.18.0.3:31752;
        server 172.18.0.4:31752;
    }

    upstream dns_servers {
        least_conn;
        server 172.18.0.3:30659;
        server 172.18.0.4:30659;
    }

    server {
        listen        5024;
        proxy_pass    stream_backend;
        proxy_timeout 120s;
        proxy_connect_timeout 120s;
    }
```

```
    server {
        listen     162 udp;
        proxy_pass dns_servers;
        #proxy_protocol on;
         proxy_bind $remote_addr transparent;
    }

}
systemctl restart nginx
```