How to Set-up and Install an Edge Cluster Agent

Deploy the Open Horizon All-in-1 Horizon Management Hub

JP> NOTE: If you already have a Hub, skip to the next section: installing and configuring a cluster.

1. Install the Management Hub without the native agent:

   ```
   curl -sSL https://raw.githubusercontent.com/open-horizon/devops/master/mgmt-hub/deploy-mgmt-hub.sh | bash -s -- -A
   ```

   Note: You can also optionally export HZN_LISTEN_IP to an external IP that can be reached outside of the local network before running the deploy-mgmt-hub.sh script
2. The end of the command output will include a summary of steps performed. In step 2, you will find a list of passwords and tokens that were automatically generated. These include the exchange root password and Hub admin password, so it is important you write these down somewhere safe.

   The final 2 lines of output will list the HZN_ORG_ID and HZN_EXCHANGE_USER_AUTH environment variables and prompt you to export them. Exporting these will allow us to continue the tutorial without the need to specify them later.

   If you would like to use different credentials to connect your agent, use the hzn exchange org create and hzn exchange user create commands to add a new org and user, respectively. Export these variables and/or take note of them.

   Note: For more information about the all-in-1 hub, please follow this link: https://github.com/open-horizon/devops/tree/master/mgmt-hub

JP> NOTE: If you already have a cluster installed, skip to the next section: Installing the Cluster Agent.

JP> NOTE: The following section on installing and configuring a cluster currently has separate directions for two solutionsL k3s and microk8s. Please choose one of those two supported solutions or use your own and translate our directions accordingly.

JK> Due to limitations with LFEdge text editor, many of the commands and input files use characters that are not unicode, so they will fail. For now, all commands and inputs should be types manually into the terminal.

Install and configure a k3s edge cluster

<DAB> This looks pretty close to what's documented in OH, what are the differences? <DAB>

This content provides a summary of how to install k3s, a lightweight and small Kubernetes cluster, on Ubuntu 18.04. For more information, see the k3s documentation.

This tutorial is for installing a k3s cluster on the same device that is running the Hub.

Note: If installed, uninstall kubectl before completing the following steps.

1. Either login as root or elevate to root with sudo -i
2. The full hostname of your machine must contain at least two dots. Check the full hostname:

   ```
   hostname
   ```

3. Install k3s:

   ```
   curl -sfL https://get.k3s.io | sh -
   ```

4. Create the image registry service: <DAB> The formatting of these yaml files is incorrect. I think we should put these files into a source controlled repo, and create a script that performs these steps.</DAB>

   a. Create a file called k3s-persistent-claim.yml with this content

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: docker-registry-pvc
   spec:
     storageClassName: "local-path"
     accessModes:
       - ReadWriteOnce
     resources:
       requests:
         storage: 10Gi
   ```

   b. Create the persistent volume claim:

```
kubectl apply -f k3s-persistent-claim.yml
```

c. Verify that the persistent volume claim was created, and it is in “Pending” status

```
kubectl get pvc
```

d. Create a file called k3s-registry-deployment.yml with this content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: docker-registry
  labels:
    app: docker-registry
spec:
  replicas: 1
  selector:
    matchLabels:
      app: docker-registry
  template:
    metadata:
      labels:
        app: docker-registry
    spec:
      volumes:
      - name: registry-pvc-storage
        persistentVolumeClaim:
          claimName: docker-registry-pvc
      containers:
      - name: docker-registry
        image: registry
        ports:
        - containerPort: 5000
        volumeMounts:
        - name: registry-pvc-storage
          mountPath: /var/lib/registry
---
apiVersion: v1
kind: Service
metadata:
  name: docker-registry-service
spec:
  selector:
    app: docker-registry
  type: NodePort
  ports:
    - protocol: TCP
      port: 5000
```

e. Create the registry deployment and service:

```
kubectl apply -f k3s-registry-deployment.yml
```

f. Verify that the docker-registry deployment and docker-registry-service service were created:

```
kubectl get deployment
kubectl get service
```

g. Define the registry endpoint:

```
export REGISTRY_ENDPOINT=$(kubectl get service docker-registry-service | grep docker-registry-service | awk
'{print $3;}'):5000
cat << EOF >> /etc/rancher/k3s/registries.yaml
mirrors:
"$REGISTRY_ENDPOINT":
    endpoint:
    - "http://$REGISTRY_ENDPOINT"
EOF
```

h. Restart k3s to pick up the change to /etc/rancher/k3s/registries.yaml:

```
systemctl restart k3s
```

5. Install docker (if not already installed):

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
apt-get install docker-ce docker-ce-cli containerd.io
```

6. Install jq (if not already installed):

```
apt-get install jq
```

7. Define this registry to docker as an insecure registry:

   a. Run the following to define an insecure registry route using the value of the $REGISTRY_ENDPOINT environment variable obtained in the last step and append it to the /etc/docker/daemon.json file.

   ```
   echo "{
       \"insecure-registries\": [ \"$REGISTRY_ENDPOINT\" ]
   }" >> /etc/docker/daemon.json
   ```

   b. Restart docker to pick up the change:

   ```
   systemctl restart docker
   ```

Install and configure a microk8s edge cluster

This content provides a summary of how to install microk8s, a lightweight and small Kubernetes cluster, on Ubuntu 18.04. (For more information, see the microk8s documentation.)

This tutorial is for installing a microk8s cluster on the same device that is running the Hub.

Note: This type of edge cluster is meant for development and test because a single worker node Kubernetes cluster does not provide scalability or high availability.

1. Install microk8s:

   ```
   sudo snap install microk8s --classic --channel=stable
   ```

2. If you are not running as root, add your user to the microk8s group:

   ```
   sudo usermod -a -G microk8s $USER
   sudo chown -f -R $USER ~/.kube
   su - $USER
   ```

3. Enable dns and storage modules in microk8s:

   ```
   microk8s.enable dns
   microk8s.enable storage
   ```

   Note: Microk8s uses 8.8.8.8 and 8.8.4.4 as upstream name servers by default. If these name servers cannot resolve the management hub hostname, you must change the name servers that microk8s is using:

   a. Retrieve the list of upstream name servers in /etc/resolv.conf or /run/system/resolve/resolv.conf

   b. Edit coredns configmap in the kube-system namespace. Set the upstream nameservers in the forward section

   ```
   microk8s.kubectl edit -n kube-system cm/coredns
   ```

4. Check the status:

   ```
   microk8s.status --wait-ready
   ```

5. The microk8s kubectl command is called microk8s.kubectl to prevent conflicts with an already install kubectl command. Assuming that kubectl is not installed, add this alias for microk8s.kubectl:

   ```
   echo 'alias kubectl=microk8s.kubectl' >> ~/.bash_aliases
   source ~/.bash_aliases
   ```

6. Enable the container registry and configure docker to tolerate the insecure registry:

   a. Enable the container registry

   ```
   microk8s.enable registry
   export REGISTRY_ENDPOINT=localhost:32000
   ```

   b. Install docker (if not already installed):

   ```
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
   add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
   apt-get install docker-ce docker-ce-cli containerd.io
   ```

   c. Install jq (if not already installed):

   ```
   apt-get install jq
   ```

d. Define this registry as insecure to docker. Create or add to /etc/docker/daemon.json (replacing with the value of the $REGISTRY_ENDPOINT environment variable you obtained in a previous step).

```
{
        "insecure-registries": [ "<registry-endpoint>" ]
}
```

e. (optional) Verify that docker is on your machine:

```
curl -fsSL get.docker.com | sh
```

f. Restart docker to pick up the change:

```
sudo systemctl restart docker
```

Install Agent on Edge Cluster

This content describes how to install the Open Horizon agent on k3s or microk8s - lightweight and small Kubernetes cluster solutions.

1. Log in to your edge cluster as root
2. Export your Hub exchange credentials if not already done:

   ```
   export HZN_EXCHANGE_USER_AUTH=<your-exchange-username>:<your-exchange-password>
   export HZN_ORG_ID=<your-exchange-organization>
   ```

3. Run the following commands to export the HZN_EXCHANGE_URL, HZN_FSS_CSSURL, HZN_AGBOT_URL and HZN_SDO_SVC_URL needed to configure the agent to be able to talk to the Hub resources.

   ```
   export HZN_EXCHANGE_URL=http://$(docker network inspect hzn_horizonnet | jq -r '.[].Containers | to_entries
   [] | select (.value.Name == "exchange-api") | .value.IPv4Address' | cut -d'/' -f1):8080/v1
   export HZN_FSS_CSSURL=http://$(docker network inspect hzn_horizonnet | jq -r '.[].Containers | to_entries[]
   | select (.value.Name == "css-api") | .value.IPv4Address' | cut -d'/' -f1):9443/
   export HZN_AGBOT_URL=http://$(docker network inspect hzn_horizonnet | jq -r '.[].Containers | to_entries[]
   | select (.value.Name == "agbot") | .value.IPv4Address' | cut -d'/' -f1):3111
   export HZN_SDO_SVC_URL=http://$(docker network inspect hzn_horizonnet | jq -r '.[].Containers | to_entries
   [] | select (.value.Name == "sdo-owner-services") | .value.IPv4Address' | cut -d'/' -f1):9008/api
   ```

4. Download the latest agent-install.sh script to your new edge cluster

   ```
   curl -sSLO https://github.com/open-horizon/anax/releases/latest/download/agent-install.sh
   chmod +x agent-install.sh
   ```

5. The agent-install.sh script will store the Open Horizon agent in the edge cluster image registry. Set the full image path (minus the tag) that should be used.

   On k3s: <DAB> The REGISTRY_ENDPOINT var is duplicate of step already performed. </DAB>

   ```
   REGISTRY_ENDPOINT=$(kubectl get service docker-registry-service | grep docker-registry-service | awk
   '{print $3;}'):5000
   export IMAGE_ON_EDGE_CLUSTER_REGISTRY=$REGISTRY_ENDPOINT/openhorizon-agent/amd64_anax_k8s
   ```

   On microk8s:

   ```
   export IMAGE_ON_EDGE_CLUSTER_REGISTRY= localhost:32000/openhorizon-agent/amd64_anax_k8s
   ```

6. Instruct agent-install.sh to use the default storage class:

   On k3s:

   ```
   export EDGE_CLUSTER_STORAGE_CLASS=local-path
   ```

   On microk8s:

   ```
   export EDGE_CLUSTER_STORAGE_CLASS=microk8s-hostpath
   ```

7. Run agent-install.sh to get the necessary files from Github, install and configure the Horizon agent, and register your edge cluster with policy.

   Since this install guide does not enable SSL on the All-in-1 Management Hub, run the following commands to create a fake certificate and install the agent. (The install script currently requires a certificate to be present even if SSL is disabled)

   ```
   touch agent-install.crt
   ./agent-install.sh -D cluster -i 'anax:'
   ```

8. Verify that the agent pod is running:

```
kubectl get namespaces
kubectl -n openhorizon-agent get pods
```

9. Use the following command to connect to a bash instance on the agent pod to execute hzn commands

```
kubectl exec -it $(kubectl get pod -l app=agent -n openhorizon-agent | grep "agent-" | cut -d " " -f1) -n
openhorizon-agent – bash
```

10. As a test, execute the following hzn command on the agent pod:

```
hzn node ls
```

11. The OpenHorizon cluster agent is now successfully installed and ready to deploy services