ELIOT Blueprints Installation Overview

ELIOT Deployments in Virtual Environments Overview

This document provides a general approach to set up ELIOT ecosystem in virtual environments. The virtual environment taken are X86_64 , ARM64 and ARM32 servers.

X86 Servers with CentOS - 7 and ARM - 64 with Ubuntu 17.10 versions are chosen for setup.

ELIOT support Kubernetes and kubeedge orchestrator based two deployment model.

ELIOT Kubernetes based setup:

ELIOT Deployment in X86_64/AMD64 Server with CentOS - 7.5 version.

Pre-Installation steps to be executed on ELIOT Manager and ELIOT Node

Disable SELinux:

```
$ setenforce 0

$ sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

Disable swap:

```
$ swapoff -a

Note : To make sure on reboot of server the swap isn't re-enabled, comment the swap line UUID in /etc/fstab file.

$ vi /etc/fstab

 # /dev/mapper/centos-swap swap swap defaults 0 0
```

Enable br_netfilter:

```
$ modprobe br_netfilter
$ echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Install Docker:

Set up the repository
Install the required packages.

```
$ sudo yum install -y yum-utils \
device-mapper-persistent-data \
lvm2
```

Set up stable repository

```
$ sudo yum-config-manager \
-add-repo \
 https://download.docker.com/linux/centos/docker-ce.repo
```

Install Docker CE

To install latest execute step a or else if specific version has to be installed execute step b.

a) Install the latest version of Docker CE and containerd

```
$ sudo yum install docker-ce docker-ce-cli containerd.io
```

b) List and sort specific available in your repo.

```
$ yum install docker-ce --showduplicates | sort -r
$ sudo yum install docker-ce-<VERSION_STRING> docker-ce-cli-<VERSION_STRING> containerd.io

Example :
$ yum list docker-ce --showduplicates | sort -r
docker-ce.x86_64  3:18.09.1-3.el7                  docker-ce-stable
docker-ce.x86_64  3:18.09.0-3.el7                  docker-ce-stable
docker-ce.x86_64  18.06.1.ce-3.el7                 docker-ce-stable

$ sudo yum install docker-ce-18.09.1 docker-ce-cli-18.09.1 containerd.io
```

Start Docker

```
$ sudo systemctl start docker
```

Install Kubernetes

Add kubernetes repository in Cent OS system

```
$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[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 packages kubeadm, kubelet and kubectl.

```
$ yum install -y kubelet kubeadm kubectl
```

After installation is complete restart the ELIOT Manager and ELIOT Node

```
$ sudo reboot
```

Login to the servers and start the services, docker and kubelet

```
$ systemctl start docker && systemctl enable docker
$ systemctl start kubelet && systemctl enable kubelet
```

Change the cgroup-driver (Need to make sure docker-ce and kubernetes are using same cgroup)

Check docker cgroup

```
$ docker info | grep -i cgroup
```

It will display docker is using 'cgroupfs' as a cgroup-driver
Run the below command to change the kubernetes cgroup-driver to 'cgroupfs' and Reload the systemd system and restart the kubelet service

```
$ sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g' /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

$ systemctl daemon-reload
$ systemctl restart kubelet
```

Kubernetes Cluster Initialization

Login to ELIOT Manager Server ; Initialize the Kubernetes Master

```
$ kubeadm init --apiserver-advertise-address=<ELIOT Manager Server IP Address> --pod-network-cidr=10.244.0.0/16
```

⚠ Note :

--apiserver-advertise-address = determines which IP Address Kubernetes should advertise its API server on.
--pod-network-cidr=10.244.0.0/16 - Pod network range. Pod network must not overlap with any of the host networks as this can cause issues. If you find a collision between your network plugin?s preferred Pod network and some of your host networks, you should think of a suitable CIDR replacement and use that during kubeadm init with --pod-network-cidr and as a replacement in your network plugin?s YAML.

To start using your cluster, you need to run (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
        or
(for root user)
$ export KUBECONFIG=/etc/kubernetes/admin.conf
```

Adding ELIOT Node to the Cluster

Execute the command in the ELIOT Node

```
$ kubeadm join --token <token> <master-ip>:6443 --discovery-token-ca-cert-hash sha256:<hash>

[The Kuberadm join command string is displayed after successfull installation of Kubernetes Master (ELIOT Manager)]
```

Deploy Flannel network in ELIOT Manager (Kubernetes Cluster)

```
$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/a70459be0084506e4ec919aa1c114638878db11b/Documentation/kube-flannel.yml
```

After join command is executed successfully , you could check the cluster node and pods

```
$ kubectl get nodes
$ kubectl get pods --all-namespaces
```

⚠ Note : You will get the 'k8s-master' node is running as a 'master' cluster with status 'ready', and you will get all pods that are needed for the cluster, including the 'kube-flannel-ds' for network pod configuration
Make sure all kube-system pod status is running.

ELIOT KubeEdge based setup:

Eliot support Light weight orchestrator KubeEdge.

For more details refer

https://wiki.akraino.org/display/AK/KubeEdge

Kube Edge Building from source

- `git clone https://github.com/kubeedge/kubeedge.git $GOPATH/src/github.com/kubeedge/kubeedge`
- `cd $GOPATH/src/github.com/kubeedge/kubeedge/keadm`
- make

Binary `keadm` is available in current path

Installing KubeEdge Master Node (on the Cloud) component

```
keadm init --docker-version=<expected version> --kubernetes-version=<expected version> --kubeedge-version=<expec
ted version>


Flags:
      --docker-version string[="18.06.0"]          Use this key to download and use the required Docker version
(default "18.06.0")
  -h, --help                                       help for init
      --kubeedge-version string[="0.3.0-beta.0"]   Use this key to download and use the required KubeEdge
version (default "0.3.0-beta.0")
      --kubernetes-version string[="1.14.1"]       Use this key to download and use the required Kubernetes
version (default "1.14.1")
```

Installing KubeEdge Worker Node (at the Edge) component

```
keadm join --edgecontrollerip=<ip address> --edgenodeid=<unique string as edge identifier>


Flags:
      --docker-version string[="18.06.0"]          Use this key to download and use the required Docker version
(default "18.06.0")
  -e, --edgecontrollerip string                    IP address of KubeEdge edgecontroller
  -i, --edgenodeid string                          KubeEdge Node unique identification string, If flag not used
then the command will generate a unique id on its own
  -h, --help                                       help for join
  -k, --k8sserverip string                         IP:Port address of K8S API-Server
      --kubeedge-version string[="0.3.0-beta.0"]   Use this key to download and use the required KubeEdge
version (default "0.3.0-beta.0")
```

Reset KubeEdge Master and Worker nodes

```
keadm reset [flags]



Flags:
  -h, --help                help for reset
  -k, --k8sserverip string  IP:Port address of cloud components host/VM
```

For more details, please follow below link:

https://kubeedge.readthedocs.io/en/latest/setup/installer_setup.html

ELIOT VM setup on ARM64

This section provides steps about setting the Virtual Machine with Ubuntu OS on ARM64 servers
Environment Details :

- ARM 64 servers with Host OS as Ubuntu 17.10 version.
- QEMU-KVM 2.10

Set up VM on ARM64 servers

Install QEMU-KVM 2.10
(Qemu-kvm is used for creating vms.)

```
$ sudo apt-get install qemu-kvm libvirt-bin
```

After qemu-kvm 2.10 is installed successful, install virtinst, followed by qemu-system-arm and qemu-efi

```
$ sudo apt-get install virtinst
```

Install QEMU and EFI image for QEMU

```
$ sudo apt-get install qemu-system-arm qemu-efi
```

After successfull instalation of qemu-efi, now need to create pflash volumes for UEFI. Two volumes are required, one static one for the UEFI firmware, and another dynamic one to store variables. Both need to be exactly 64M in size

```
$ dd if=/dev/zero of=flash0.img bs=1M count=64
$ dd if=/usr/share/qemu-efi/QEMU_EFI.fd of=flash0.img conv=notrunc
$ dd if=/dev/zero of=flash1.img bs=1M count=64
```

Set Search Permissions for libvirt:
Change contents of qemu.conf found in below path /etc/libvirt/qemu.conf
uncomment user = "root" and group = "root"

Restart libvirtd

```
$ service libvirtd restart
```

⚠ Note: if you closed the terminal and reopened you need to repeat the previous step once.

Start QEMU VMs with LIBVIRT:
Download the below image from the internet by executing the below command

```
$ wget http://cdimage.ubuntu.com/releases/16.04/release/ubuntu-16.04.4-server-arm64.iso?_ga=2.265907727.1041693553.1521178200-1580291815.1512480357
```

Rename the ISO file to ubuntu-16.04.4-server-arm64.iso.

```
$ mv ubuntu-16.04.4-server-arm64.iso?_ga=2.265907727.1041693553.1521178200-1580291815.1512480357 ubuntu-16.04.4-server-arm64.iso
```

Execute the below command for VM creation:

```
$ sudo virt-install --name <VM Name> --ram 8192 --disk path=dut1.img,size=<Storage Space> --vcpus <Number of CPU's> --os-type linux --os-variant generic --cdrom './ubuntu-16.04.4-server-arm64.iso' --network default

Sample :-
$ sudo virt-install --name dut1 --ram 8192 --disk path=dut1.img,size=50 --vcpus 4 --os-type linux --os-variant generic --cdrom './ubuntu-16.04.4-server-arm64.iso' --network default
```

Note: In the above command VM name is dut1 , memory = 50 GB and CPUS = 4
After --name attribute will be your VM name.
After --ram will be your ram size
After --vcpus attribute will be your CPU?s for your VM.
After --cdrom attribute within the single quotes will b path for .iso file.
(The above step is also suitable with disk image also)
Repeat the above step (virt-install) only if you need more than one VM.
After creating the VM you can see your IP of your VM by executing command ifconfig.

If you unexpectedly closes the terminal before knowing your IP address of VM , execute arp -n in terminal.
It will give all the IP addresses of your main machine along with all the VM's you created.