# IEC Blueprints Installation Overview

## Contents


- IEC Reference Foundation Overview
  - Networking and HW Prerequisites
  - Methods of Installation
  - Kubernetes Install for Ubuntu
    - Install Docker as Prerequisite
    - Turn off all swap devices and files with: Disable swap on your machine
    - Install Kubernetes with Kubeadm
  - Install the Calico CNI Plugin to Kubernetes Cluster
    - Install the Etcd Database
    - Install Calico to system
    - Remove the taints on master node
  - Verification for the Work of Kubernetes
  - Helm Install on Arm64
- Alternative Methods of Installation
  - Script Based Installation
  - OPNFV Installers
    - OPNFV Fuel
  - Heat Orchestration Templates


# IEC Reference Foundation Overview

This document provides a general description about the reference foundation of IEC. The Integrated Edge Cloud (IEC) will enable new functionalities and business models on the network edge. The benefits of running applications on the network edge are - Better latencies for end users - Less load on network since more data can be processed locally - Fully utilize the computation power of the edge devices

Currently, the chosen operating system(OS) is Ubuntu 16.04 and/or 18.04. The infrastructure orchestration of IEC is based on Kubernetes, which is a production-grade container orchestration with rich running eco-system. The current container network interface(CNI) solution chosen for Kubernetes is project Calico, which is a high performance, scalable, policy enabled and widely used container networking solution with rather easy installation and arm64 support. In the future, Contiv/VPP or OVN-Kubernetes would also be candidates for Kubernetes networking.

## Networking and HW Prerequisites

For a virutal deploy, minimum hardware requirement is 1 baremetal server (either x86_64 or aarch64) for a deploy with 3 VMs on it.

For a baremetal deploy minimum hardware requirement is 3 baremetal servers.

Networking requirements - TBD

## Methods of Installation

To address a large variety of setups, multiple methods of deployment should be supported. Deployment works both on x86_64 and aarch64 hw.

| Method | Pros (current state) | Cons (current state) | Prerequisites |
|---|---|---|---|
| Manual installation | • Full control over each step<br>• Easy to understand and replicate<br>• Already available (see next chapter on this page) | • Requires user intervention<br>• Requires certain prerequisites be met on cluster nodes apriori | • preinstalled operating system (Ubuntu 16.04/18.04) on all involved nodes |
| Script-based installation | • High degree of flexibility via arguments<br>• Portable<br>• Can be used in CI/CD, assuming baremetal nodes are pre-provisioned, e.g. for shorter test cycles like a patch verify job where we'd want to avoid reinstalling the operating system each time | • Implementation currently in progress<br>• Fixed number of nodes (1 master + 1 worker)<br>• Requires certain prerequisites be met on cluster nodes apriori | • preinstalled operating system (Ubuntu 16.04/18.04) on all involved nodes<br>• user with passwordless sudo access already available on the target nodes |

| | | | |
|---|---|---|---|
| OPNFV-based installer (s) | • Unified and standardized input configuration files (PDF/IDF)<br>• Can be used in CI/CD<br>• Can handle OS provisioning on its own, for virtual, baremetal or hybrid PODs | • Not yet implemented<br>• Requires hardware descriptor files (PDF /IDF) | • Jumpserver (installer) node preinstalled<br>• XDF (PDF/IDF) available for the target lab |
| Heat stack | • Portable | • Not tested on aarch64 yet<br>• Uses VMs rather than baremetal | • Openstack cloud preinstalled |
| Other installer solutions (e.g. Airship) | • Alignment with industry standard installer solutions for K8s | • Not implemented<br>• More complex design and configuration<br>• Might be overkill for IEC, at least with the current requirements | • TBD |

# Kubernetes Install for Ubuntu

## Install Docker as Prerequisite

Please follow docker installation guide for Ubuntu arm64 to install Docker CE:

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Please select the 18.06 version since it is the latest version kubelet supported.

```
$DOCKER_VERSION=18.06.1
$ARCH=arm64
$curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$sudo apt-key fingerprint 0EBFCD88
$sudo add-apt-repository \
"deb [arch=${ARCH}] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"

$sudo apt-get update
$sudo apt-get install -y docker-ce=${DOCKER_VERSION}~ce~3-0~ubuntu
```

## Turn off all swap devices and files with: **Disable swap on your machine**

```
$ sudo swapoff -a
```

## Install Kubernetes with Kubeadm

kubeadm helps you bootstrap a minimum viable Kubernetes cluster that conforms to best practices which is a preferred installation method for IEC currently. Now we choose v1.13.0 as a current stable version of Kubernetes for arm64. Usually the current host(edge server/gateway)'s management interface is chosen as the Kubeapi-server advertise address which is indicated here as `$MGMT_IP`.

The common installation steps for both Kubernetes master and slave node are given as Linux shell scripts:

```
$ sudo bash
$ apt-get update && apt-get install -y apt-transport-https curl
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
$ cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
$ deb https://apt.kubernetes.io/ kubernetes-xenial main
$ EOF
$ apt-get update
$ apt-get install -y kubelet=1.13.0-00 kubeadm=1.13.0-00 kubectl=1.13.0-00
$ apt-mark hold kubelet kubeadm kubectl
$ sysctl net.bridge.bridge-nf-call-iptables=1
```

For host setup as Kubernetes master:

> **Attention**
>
> --pod-network-cidr is the subnet of the pod network and must not overlap with any of the host networks
> (see https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#pod-network for more details)
>
> --apiserver-advertise-address is the IP on the master node to use for advertising the master's IP
> (see https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#initializing-your-master for more details)
>
> --service-cidr can be set up to be used for service VIPs and must not overlap with any of the host networks.
> The default value is "10.96.0.0/12". If you change the default, make sure you update the clusterIP address
> in the yaml file when installing etcd (in the steps below)

```
$ sudo kubeadm config images pull
$ MGMT_IP=<ip-address>
$ sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=$MGMT_IP \
--service-cidr=172.16.1.0/24
```

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

```
$ export KUBECONFIG=/etc/kubernetes/admin.conf
```

if you are the ``root`` user.

For hosts setup as Kubernetes slave:

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

which will skip ca-cert verification.
After the `slave` joining the Kubernetes cluster, in the master node, you could check the cluster node with the command:

```
$ kubectl get nodes
```

## Install the Calico CNI Plugin to Kubernetes Cluster

Now we install a Calico network add-on so that Kubernetes pods can communicate with each other. The network must be deployed before any applications. Kubeadm only supports Container Networking Interface(CNI) based networks for which Calico has supported.

### Install the Etcd Database

Please use the following command to install etcd database.

```
$ wget https://raw.githubusercontent.com/iecedge/iec/master/src/foundation/scripts/cni/calico/etcd.yaml
$ sed -i "s/10.96.232.136/${CLUSTER_IP}/" ./etcd.yaml
$ kubectl apply -f etcd.yaml
```

## Install the RBAC Roles required for Calico

```
$ kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/rbac.yaml
```

### Install Calico to system

Firstly, we should get the configuration file from web site and modify the corresponding image from amd64 to arm64 version. Then, by using kubectl, the calico pod will be created.

```
$ wget https://raw.githubusercontent.com/iecedge/iec/master/src/foundation/scripts/cni/calico/calico.yaml
```

Since the "quay.io/calico" image repo does not support does not multi-arch, we have to replace the “quay.io/calico” image path to "calico" which supports multi-arch.

```
$ sed -i "s@10.96.232.136@${CLUSTER_IP}@; s@192.168.0.0/16@${POD_NETWORK_CIDR}@" ./calico.yaml
```

Deploy the Calico using following command:

```
$ kubectl apply -f calico.yaml
```

**Attention**

```
In calico.yaml file, there is an option "IP_AUTODETECTION_METHOD" about choosing
network interface. The default value is "first-found" which means the first valid
IP address (except local interface, docker bridge). So if the number of network-interface
is more than 1 on your server, you should configure it depends on your networking
environments. If it does not configure it properly, there are some error about
calico-node pod: "BGP not established with X.X.X.X".
```

### Remove the taints on master node

```
$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

# Verification for the Work of Kubernetes

Now we can verify the work of Kubernetes and Calico with Kubernets pod and service creation and accessing based on Nginx which is a widely used web server.

Firstly, create a file named nginx-app.yaml to describe a Pod and service by:

```
$ cat <<EOF >~/nginx-app.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
    name: http
  selector:
    app: nginx
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        #image: arm64v8/nginx:stable
        image: nginx
        ports:
        - containerPort: 80
EOF
```

then test the Kubernetes working status with the script:

**Bash**

```
   set -ex

   kubectl create -f ~/nginx-app.yaml
   kubectl get nodes
   kubectl get services
   kubectl get pods
   kubectl get rc

   r="0"
   while [ $r -ne "2" ]
   do
      r=$(kubectl get pods | grep Running | wc -l)
      sleep 60
   done

   svcip=$(kubectl get services nginx  -o json | grep clusterIP | cut -f4 -d'"')
   sleep 10
   wget http://$svcip
   #kubectl delete -f ./examples/nginx-app.yaml
   kubectl delete -f ./nginx-app.yaml
   kubectl get rc
   kubectl get pods
   kubectl get services
```

# Helm Install on Arm64

Helm is a tool for managing Kubernetes charts. Charts are packages of pre-configured Kubernetes resources. The installation of Helm on arm64 is as follows:

```
$ wget https://storage.googleapis.com/kubernetes-helm/helm-v2.12.3-linux-arm64.tar.gz
$ tar -xvf helm-v2.12.3-linux-arm64.tar.gz
$ sudo cp linux-arm64/helm /usr/bin
$ sudo cp linux-arm64/tiller /usr/bin
```

# Further Information

We would like to provide a walk through shell script (described in the following chapter) to automate the installation of Kubernetes and Calico in the future. But this README is still useful for IEC developers and users.

For issues or anything on the reference foundation stack of IEC, you could contact:

Trevor Tao: trevor.tao@arm.com

Jingzhao Ni: jingzhao.ni@arm.com

Jianlin Lv: jianlin.lv@arm.com

# Alternative Methods of Installation

## Script Based Installation

Akraino IEC repository now provides an automated method, based on sh scripts, that handles all above steps.

Prerequisites:

- 2 nodes (virtual machines or baremetal) with a preinstalled operating system (Ubuntu 16.04/18.04) and passwordless-sudo capable user on them (password-based login via SSH enabled);

The following snippet will automatically handle all steps described above in the previous chapter:

```
$ git clone https://gerrit.akraino.org/r/iec
# iec/scripts/startup.sh [master ip] [worker ip] [user] [password]
$ iec/scripts/startup.sh 10.169.40.171 10.169.41.172 iec 123456
```

## OPNFV Installers

### OPNFV Fuel

OPNFV Fuel installer can be leveraged to automate the IEC prerequisites setup (e.g. baremetal operating system provisioning for baremetal clusters), as well as the IEC installation itself.

Prerequisites:

- 1 jumpserver node with preinstalled operating system (Ubuntu 16.04/18.04 or CentOS7) - will also be used as a hypvervisor for the IEC VMs - for single hypervisor deployments;
- 1 jumpserver node with preinstalled operating system + 3 baremetal nodes for multiple hypervisor deployments;

Supported configurations include, but are not limited to:

- single hypervisor node running 3 VMs dedicated to IEC;
- 3 baremetal nodes dedicated IEC (K8 directly on baremetal);
- 3 baremetal nodes running a virtual control plane (each baremetal node has 1 VM dedicated to IEC);

Upstream Fuel patch is currently undergoing final stages of review and is expected to be merged soon.

Once the Fuel patch lands upstream, deploying IEC (including handling its prerequisites, like creating the required VMs on the hypervisor) can be done using (e.g. for an AArch64 single-hypervisor POD):

```
$ git clone -b stable/hunter https://github.com/opnfv/fuel
$ fuel/ci/deploy.sh -l arm -p virtual2 -s k8-nosdn-iec-noha -S /var/lib/opnfv/tmpdir/ -D |& tee deploy.log
```

## Heat Orchestration Templates

Prerequisites:

- Openstack Ocata or latest

Recommended configuration:

- 2 or more compute nodes with enough RAM and disk (128 GB of RAM, 2 TB disk space)
- DPDK is optional but it is recommended

The scripts and templates can be found in the Akraino iec git repository:

**IEC HOT**

```
$ git clone https://gerrit.akraino.org/r/iec
$ cd iec/src/foundation/hot
$ # [has_dpdk=true] [skip_k8s_net=1] [skip_k8s_master=1] [skip_k8s_slaves=1] external_net=<external_net> .
/control.sh <start|stop>
$ has_dpdk=true external_net=external ./control.sh start
```

More useful information can be found in the README in the same directory