User Documentation

Introduction

This document describes how to deploy blueprints from Akraino's KNI Blueprint Family. It is common to all blueprints in that family, unless otherwise noted.

License

All our code is released under Apache license: https://www.apache.org/licenses/LICENSE-2.0.html

How to use this document

This document describes the generic installation for our KNI blueprint family. Specific documentation is provided for Provider Access Edge and Industrial Edge blueprints. See KNI PAE Installation Guide

Deployment architecture

See KNI PAE Architecture document

Pre-Installation Requirements

Resource Requirements

The resource requirements for deployment depend on the specific blueprint and deployment target. Please see:

- Provider Access Edge (PAE)
- Industrial Edge blueprint

Pre-Requisites for Deploying to AWS

For deploying a KNI blueprint to AWS, you need to

- add a public hosted DNS zone for the cluster to Route53,
- validate your AWS quota in the chosen region is sufficient,
- set up an API user account with the necessarily IAM privileges.

Please see the upstream documentation for details.

Store the `aws-access-key-id` and `aws-secret-access-key` in a credentials file inside $HOME/.aws, with the following format:

```
[default]
aws_access_key_id=xxx
aws_secret_access_key=xxx
```

Pre-Requisites for Deploying to Bare Metal

The baremetal UPI install can be optionally automated when using knictl (see below). When attempting a manual baremetal UPI install, however, please be sure to read: https://docs.openshift.com/container-platform/4.4/installing/installing_bare_metal/installing-bare-metal.html

Minimum hardware requirements

This is minimal configuration example where only 3 servers are used. Servers and their role are given in below table.

Server#	Role	Purpose
1	Installer node	This host is used for remotely installing and configuring master and worker node. This server also hosts bootstrap node on KVM-QEMU using libvirt. Several components like- HAProxy, DNS server, DHCP server for provisioning and baremetal network, CoreDNS, Matchbox, Terraform, IPMItool, TFTPboot are configured on this server. Since cluster coreDNS is running from here, this node will be required later as well.
2	Master node	This is control plane or master node of K8s cluster that is based on openshift 4.x.
3	Worker node	This is worker node which hosts the application.

4	Bootstrap node	Bootstrap node **runs as VM on installer node** and it exists only during the installation and later automatically deleted by installer.

Other installation requirements

Network requirements

Each server should have 3 Ethernet ports configured, purpose of these is listed below. **These three are in addition to IPMI port, which is required for PXE boot.**

Interface	Purpose
Management interface	Remote root login from this interface is used for entire setup. This interface needs to have internet connectivity to download various files. This can be shared with external interface. This only needs to be present on the Installer node
External interface	Interface on the installer node that has internet network connectivity. All external traffic from masters/workers is redirected to the external interface of the installer node.
Baremetal interface	This interface is for baremetal network, also known as SDN network. This interface doesn't need internet connectivity.
Provisioning interface	This interface is for PXE boot. This interface doesn't need internet connectivity.

These can be independent NICs or VLANs.

Configure required network interfaces as explained earlier. Be sure that each server has the NIC for PXE configured properly, matching to the interface that you are setting for this deployment. You can set it by entering the BIOS setup, and entering into the NIC configuration of your BIOS setup menu.

Collect IPs and MAC addresses of all the nodes, one sample is given below. This information will be required to populate config files:

Role	iDRAC IP/IPMI port IP	Provisioning network IP	Baremetal network IP	Management network IP	Provisioning network port & mac	Baremetal network port & mac	Management network port & mac
Installer	xx.xx.xx.xx	xx.xx.xx.xx	xx.xx.xx.xx	xx.xx.xx.xx	em1 / 21:02:0E:DC:BC:27	em2/ 21:02:0E:DC:BC:28	em3/ 21:02:0E:DC:BC:29
master-0							
worker-0							

Enable IPMI over LAN for all master and worker nodes. This is required for remote PXE boot from installer node. Different servers have different ways to enable it.

In absence of this setting, following kind of errors are thrown from installer.

> Error: Error running command '   ipmitool -I lanplus -H x.x.x.x -U xxx -P xxxxx chassis bootdev pxe;
>
>   ipmitool -I lanplus -H x.x.x.x -U xxx -P xxxxx power cycle || ipmitool -I lanplus -H x.x.x.x -U xxx -P xxxxx power on;
>
> ': exit status 1. Output: Error: Unable to establish IPMI v2 / RMCP+ session
>
> Error: Unable to establish IPMI v2 / RMCP+ session
>
> Error: Unable to establish IPMI v2 / RMCP+ session
>
> *Depending on servers, RMCP session needs to be enabled on security settings of the management console.*

After enabling this setting, you can run below command to verify that it is working as expected. Give IP address, username and password.

> ipmitool -I lanplus -H x.x.x.x -U xxx -P xxxxx chassis status

(where x.x.x.x is IPMI port IP of your master/worker node, this is followed by root username and password for IPMI e.g. iDRAC)

Bare metal node requirements

Node Role	OS requirement

Installer	CentOS 7.6 and above
Bootstrap	RHCOS (Redhat CoreOS)
Master	RHCOS (Redhat CoreOS)
Worker	RHCOS/RHEL/CentOS/CentOS-rt

Pre-Requisites for Deploying to Google Cloud Platform

For deploying a KNI blueprint to GCP, you need to:

- enable service APIs
- setup DNS
- ensure sufficient quota
- create a installer service account

Please, see the upstream documentation for details. As mentioned in the KNI installer repo, the service account JSON file should be located inside $HOME/.gcp with the name osServiceAccount.json.

Pre-Requisites for Deploying to Libvirt

Minimum hardware requirements

Only one server is needed, that will be acting as a virthost. Master and worker VMs will be created there

Server#	Role	Purpose
1	Installer node	This host is used for remotely installing and configuring master and worker node. This server also hosts bootstrap node on KVM-QEMU using libvirt. Several components like- HAProxy, DNS server, DHCP server for provisioning and baremetal network, CoreDNS, Matchbox, Terraform, IPMItool, TFTPboot are configured on this server. Since cluster coreDNS is running from here, this node will be required later as well.

Network requirements

Network connectivity will be the same as the baremetal case, but these can be dummy interfaces as all the network connectivity will be just inside the same host:

Interface	Purpose
Management interface	Remote root login from this interface is used for entire setup. This interface needs to have internet connectivity to download various files. This can be shared with external interface. This only needs to be present on the Installer node
External interface	Interface on the installer node that has internet network connectivity. All external traffic from masters/workers is redirected to the external interface of the installer node.
Baremetal interface	This interface is for baremetal network, also known as SDN network. This interface doesn't need internet connectivity.
Provisioning interface	This interface is for PXE boot. This interface doesn't need internet connectivity.

Jump host requirements

Node Role	OS requirement
Installer	CentOS 7.6 and above

Installation high level overview

Virtual deployment guide

Create site for AWS and GCP

In order to deploy a blueprint, you need to create a repository with a site. The site configuration is based in kustomize, and needs to use our blueprints as base, referencing that properly. Sample sites for deploying on libvirt, AWS and baremetal can be seen on: https://github.com/akraino-edge-stack/kni-blueprint-pae/tree/master/sites.
Site needs to have this structure:

> *.*
> *00_install-config*
> *install-config.name.patch.yaml*
> *install-config.patch.yaml*
> *kustomization.yaml*
> *site-config.yaml*
> *01_cluster-mods*
> *kustomization.yaml*
> *manifests*
> *openshift*
> *02_cluster-addons*
> *kustomization.yaml*
> *03_services*
> *kustomization.yaml*

00_install-config

This folder will contain the basic settings for the site, including the base blueprint/profile, and the site name/domain. The following files are needed:

- kustomization.yaml: key file, where it will contain a link to the used blueprint/profile, and a reference to the used patches to customize the site bases:

  ```
  bases:
  - git::https://gerrit.akraino.org/r/kni/blueprint-pae.git//profiles/production.aws/00_install-config

  patches:
  - install-config.patch.yaml

  patchesJson6902:
  - target:
  version: v1
  kind: InstallConfig
  name: cluster
  path: install-config.name.patch.yaml

  transformers:
  - site-config.yaml
  ```

  The entry in bases needs to reference the blueprint being used (in this case blueprint-pae), and the profile install-config file (in this case production.aws/00_install-config). The other entries need to be just written literally.
- install-config.patch.yaml: is a patch to modify the domain from the base blueprint. You need to customize with the domain you want to give to your site

  ```
  apiVersion: v1
  kind: InstallConfig
  metadata:
  name: cluster
  baseDomain: devcluster.openshift.com
  ```

- install-config.name.patch.yaml: is a patch to modify the site name from the base blueprint. You need to customize with the name you want to give to your site

```
- op: replace
  path: "/metadata/name"
  value: kni-site
```

- site-config.yaml: site configuration file, you can add entries in config to override behaviour of knictl (currently just releaseImageOverride is supported)

```
apiVersion: kni.akraino.org/v1alpha1
kind: SiteConfig
metadata:
 name: notImportantHere
 config:
   releaseImageOverride: registry.svc.ci.openshift.org/origin/release:4.4
```

NOTE: If you are deploying on baremetal, specific configuration needs to be set. This is going to be covered in an specific section for it

01_cluster_mods

This is the directory that will contain all the customizations for the basic cluster deployment. You could create patches for modifying number of masters /workers, network settings... everything that needs to be modified on cluster deployment time. It needs to have a basic **kustomization.yaml** file, that will reference the same level file for the blueprint. And you could create additional patches following kustomize syntax:

```
bases:
- git::https://gerrit.akraino.org/r/kni/blueprint-pae.git//profiles/production.aws/01_cluster-mods
```

02_cluster_addons and 03_services

Follow same structure as 01_cluster_mods, but in this case is for adding additional workloads after cluster deployment. They also need to have a **kustomization.yaml** file that references the file of the same level for the blueprint, and can include additional resources and patches.

How to deploy on AWS and GCP

The whole deployment workflow is based on knictl CLI tool that this repository is providing.

CLI tool

The current KNI blueprints use the `openshift-install` tool from the OKD Kubernetes distro to stand up a minimal Kubernetes cluster. All other Day 1 and Day 2 operations are then driven purely through manipulation of declarative Kubernetes manifests. To use this in the context of Akraino KNI blueprints, the project has created a helper CLI tool that needs to be installed first on Installer Node.

If necessary, install golang binary (incl. GOPATH var) using following steps, you can use latest version instead of the one given below.

```
wget https://golang.org/dl/go1.13.4.linux-amd64.tar.gz

tar -C /usr/local -xzf go1.13.4.linux-amd64.tar.gz

export PATH=$PATH:/usr/local/go/bin
```

Next, install the following dependencies:

```
sudo yum install -y make gcc libvirt-devel
```

Then install the knictl:

```
mkdir -p $GOPATH/src/gerrit.akraino.org/kni
cd $GOPATH/src/gerrit.akraino.org/kni
git clone https://gerrit.akraino.org/r/kni/installer
cd installer
make build
mkdir -p $GOPATH/bin/
cp knictl $GOPATH/bin/

cp knictl /usr/local/go/bin/
```

Secrets

Most secrets (TLS certificates, Kubernetes API keys, etc.) will be auto-generated for you, but you need to provide at least two secrets yourself:

- a public SSH key
- a pull secret

The public SSH key is automatically added to every machine provisioned into the cluster and allows remote access to that machine. In case you don't have / want to use an existing key, you can create a new key pair using:

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa
```

The pull secret is used to download the container images used during cluster deployment. Unfortunately, the OKD Kubernetes distro used by the KNI blueprints does not (yet) provide pre-built container images for all of the deployed components. Instead of going through the hassle of building those from source, we use the ones made available by openshift.com. Therefore, you need to go to https://cloud.redhat.com/openshift/install/metal/user-provisioned, log in (creating a free account, if necessary), and hit "Download Pull Secret".

Create a $HOME/.kni folder and copy the following files:

- id_rsa.pub needs to contain the public key that you want to use to access your nodes
- pull-secret.json needs to contain the pull secret previously copied

1. Fetch requirements for a site.

You need to have a site repository with the structure described above. Then, first thing is to fetch the requirements needed for the blueprint that the site references. This is achieved by:

```
./knictl fetch_requirements github.com/site-repo.git
```

Where the first argument references a site repository, following https://github.com/hashicorp/go-getter syntax. This will download the site repository, and will create a folder with the site name inside *$HOME/.kni* . It will also fetch all the binaries needed, and will store them inside *$HOME/.kni/$SITE_NAME /requirements* folder.

2. Prepare manifests for a site

NOTE: Before performing this step, you must copy your OpenShift pull secret into your build path (i.e. to ~/.kni/pull-secret.json).

Next step is to run a procedure to prepare all the manifests for deploying a site. This is achieved by applying kustomize on the site repository, combining that with the base manifests for the blueprint, and doing a merge with the manifests generated by the installer at runtime. This is achieved by the following command:

```
./knictl prepare_manifests $SITE_NAME
```

This will generate a set of manifests ready to apply, and will be stored on *$HOME/.kni/$SITE_NAME/final_manifests* folder. Along with manifests, a *profile.env* file has been created also in *$HOME/.kni/$SITE_NAME* folder. It includes environment vars that can be sourced before deploying the cluster. Current vars that can be exported are:

- OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE : used when a new image is wanted, instead of the default one
- TF_VAR_libvirt_master_memory, TF_VAR_libvirt_master_vcpu: Used in the libvirt case, to define the memory and CPU for the vms.

3. Deploy the cluster

Before starting the deployment, it is recommended to source the env vars from profile.env . You can achieve it with:

```
source $HOME/.kni/$SITE_NAME/profile.env
```

If you are deploying on AWS or libvirt, then you need to deploy the cluster. This can be achieved with:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install create cluster --dir=$HOME/.kni/$SITE_NAME/final_manifests
```

This will deploy a cluster based on the specified manifests. You can learn more about how to manage cluster deployment and how to interact with it on https://docs.openshift.com/container-platform/4.4/welcome/index.html

Specific instructions for baremetal are going to be provided later.

4. Apply workloads

After the cluster has been generated, the extra workloads that have been specified in manifests (like kubevirt), need to be applied. This can be achieved by:

```
./knictl apply_workloads $SITE_NAME
```

This will execute kustomize on the site manifests and will apply the output to the cluster. After that, the site deployment can be considered as finished.

Bare metal deployment guide

Create site for Baremetal

First step to start a baremetal deployment is to have a site defined, with all the network and baremetal settings defined in the yaml files. A sample of site using this baremetal automation can be seen here .
In order to define the settings for a site, the first section 00_install-config needs to be used.
Start by creating a kustomization file like the following: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/community.baremetal.edge-sites.net/00_install-config/kustomization.yaml

```
bases:
- git::https://gerrit.akraino.org/r/kni/blueprint-pae.git//profiles/production.baremetal/00_install-config

patches:
- install-config.patch.yaml

patchesJson6902:
- target:
    version: v1
    kind: InstallConfig
    name: cluster
  path: install-config.name.patch.yaml

transformers:
- site-config.yaml
```

In this kustomization file we are patching the default install-config, and also adding some extra files to define networking (site-config.yaml).

ha-lab-ipmi-creds.yaml:

This file is not shown on the site structure as it contains private content. This file should be present with given name in **00_install-config.** It needs to have following structure:

```
apiVersion: v1
kind: Secret
metadata:
  name: community-lab-ipmi
stringdata:
  username: xxx <- base64 encoded IPMI username
  password: xxx <- base64 encoded IPMI password

type: Opaque
```

install-config.name.patch.yaml: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/community.baremetal.edge-sites.net/00_install-config/install-config.name.patch.yaml

```
- op: replace
  path: "/metadata/name"
  value: community <- replace with your cluster name here
```

install-config.patch.yaml : https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/community.baremetal.edge-sites.net/00_install-config/install-config.patch.yaml

```
apiVersion: v1
kind: InstallConfig
baseDomain: baremetal.edge-sites.net <- domain of your site
compute:
 - name: worker
   replicas: 2 <- number of needed workers
controlPlane:
   name: master
   platform: {}
   replicas: 1 <- number of needed masters (1/3)
metadata:
   name: cluster  <- Do not change this value as this is not cluster name
networking:
  clusterNetworks:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
   none: {}
   apiVIP: 192.168.111.4  <- IP for Kubernetes api endpoint, needs to be on the range of your baremetal network
   ingressVIP: 192.168.111.3 <- IP for the Kubernetes ingress endpoint, needs to be on the range of your
baremetal network
   dnsVIP: 192.168.111.2 <- IP for the Kubernetes DNS endpoint, needs to be on the range of your baremetal network
   hosts:
      # Master nodes are always RHCOS
      -  name: master-0
         role: master
         bmc:
            address: ipmi://10.11.7.12 <- ipmi address for master
            credentialsName: community-lab-ipmi <- this needs to reference the name of the secret provided in
credentials.yaml
         bootMACAddress: 3C:FD:FE:CD:98:C9  <- mac address for the provisioning interface of your master
         sdnMacAddress: 3C:FD:FE:CD:98:C8   <- mac address for the baremetal interface of your master
         # sdnIPAddress: 192.168.111.11     <- Optional -- Set static IP on your baremetal for your master
         hardwareProfile: default
         osProfile:
            # With role == master, the osType is always rhcos
            # And with type rhcos, the following are settings are available
            type: rhcos
            pxe: bios         <- pxe boot type either bios (default if not specified) or uefi
            install_dev: sda  <- where to install the operating system (sda is the default)
      # Worker nodes can be either rhcos (default) || centos (7.x) || rhel (8.x)
      -  name: worker-0
         role: worker
         bmc:
            address: ipmi://10.11.7.13
            credentialsName: community-lab-ipmi
         bootMACAddress: 3C:FD:FE:CD:9E:91
         sdnMacAddress: 3C:FD:FE:CD:9E:90
         hardwareProfile: default
         provisioning_interface: enp134s0f1 <- specify that if the provisioning interface is different than the
one you will provide on next site-config.yaml
         baremetal_interface: enp134s0f0 <- specify that if the baremetal interface is different than the one you
will provide on next site-config.yaml
         # If an osProfile/type is not defined, rhe node defaults to RHCOS
         osProfile:
            type: centos7
            # With type: rhcos the following are settings are available
            pxe: bios    # pxe boot type either bios (default if not specified) or uefi
            install_dev: sda  # where to install the operating system (sda is the default)
      -  name: worker-1
         role: worker
         bmc:
            address: ipmi://10.11.7.14
            credentialsName: community-lab-ipmi
         bootMACAddress: 3C:FD:FE:CD:9B:81
         sdnMacAddress: 3C:FD:FE:CD:9B:80
         hardwareProfile: default
         # If an osProfile/type is not defined, rhe node defaults to RHCOS
         # osProfile:
            # type: rhcos
            # With type: rhcos the following are settings are available
            # pxe: bios|uefi    # pxe boot type either bios (default if not specified) or uefi
            # install_dev: sda  # where to install the operating system (sda is the default)
pullSecret: 'PULL_SECRET'  <- Do not change anything here as this is automatically pulled from installer node
sshKey: |
   SSH_PUB_KEY             <- Do not change anything here as this is automatically pulled from installer node
```

site-config.yaml: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/community.baremetal.edge-sites.net/00_install-config/site-config.yaml

```
apiVersion: kni.akraino.org/v1alpha1
kind: SiteConfig
metadata:
  name: notImportantHere
config: {}
provisioningInfrastructure:
  hosts:
    # interface to use for provisioning on the masters
    masterBootInterface: ens787f1 <- name of the provisioning interface for the masters
    # interface to use for provisioning on the workers
    workerBootInterface: ens787f1 <- name of the provisioning interface for the workers
    # interface to use for baremetal on the masters
    masterSdnInterface: ens787f0 <- name of the baremetal interface for the masters
    # interface to use for baremetal on the workers
    workerSdnInterface: ens787f0 <- name of the baremetal interface for the workers

  network:
    # The provisioning network's CIDR
    provisioningIpCidr: 172.22.0.0/24 <- range of the provisioning network
    # PXE boot server IP
    # DHCP range start (usually provHost/interfaces/provisioningIpAddress + 1)
    provisioningDHCPStart: 172.22.0.11 <- DHCP start range of the provisioning network
    provisioningDHCPEnd: 172.22.0.51 -> DHCP end range

    # The baremetal networks's CIDR
    baremetalIpCidr: 192.168.111.0/24 <- range of the baremetal network
    # Address map
    # bootstrap: baremetalDHCPStart   i.e. 192.168.111.10
    # master-0: baremetalDHCPStart+1  i.e. 192.168.111.11
    # master-1: baremetalDHCPStart+2  i.e. 192.168.111.12
    # master-2: baremetalDHCPStart+3  i.e. 192.168.111.13
    # worker-0: baremetalDHCPStart+5  i.e. 192.168.111.15
    # worker-N: baremetalDHCPStart+5+N
    baremetalDHCPStart: 192.168.111.10 <- DHCP start range of the baremetal network. Needs to start with an IP
that does not conflict with previous baremetal VIP definitions
    baremetalDHCPEnd: 192.168.111.50 <- DHCP end range
    # baremetal network default gateway, set to proper IP if /provHost/services/baremetalGateway == false
    # if /provHost/services/baremetalGateway == true, baremetalGWIP with be located on provHost/interfaces
/baremetal
    # and external traffic will be routed through the provisioning host
    baremetalGWIP: 192.168.111.4
    dns:
      # cluster DNS, change to proper IP address if provHost/services/clusterDNS == false
      # if /provHost/services/clusterDNS == true, cluster (IP) with be located on provHost/interfaces/provisioning
      # and DNS functionality will be provided by the provisioning host
      cluster: 192.168.111.3
      # Up to 3 external DNS servers to which non-local queries will be directed
      external1: 8.8.8.8
#     external2: 10.11.5.19
#     external3: 10.11.5.19

  provHost:
    interfaces:
      # Interface on the provisioning host that connects to the provisioning network
      provisioning: enp136s0f1 <- it typically needs to be a nic, not a vlan (unless your system supports pxe
booting from vlans)
      # Must be in provisioningIpCidr range
      # pxe boot server will be at port 8080 on this address
      provisioningIpAddress: 172.22.0.1
      # Interface on the provisioning host that connects to the baremetal network
      baremetal: enp136s0f0.3009
      # Must be in baremetalIpCidr range
      baremetalIpAddress: 192.168.111.1
      # Interface on the provisioning host that connects to the internet/external network
      external: enp136s0f0.3008
    bridges:
      # These bridges are created on the bastion host
      provisioning: provisioning <- typically leave those fixed names
      baremetal: baremetal
    services:
      # Does the provsioning host provide DHCP services for the baremetal network?
      baremetalDHCP: true <- set it to false just if you have your own DHCP for the baremetal network
      # Does the provisioning host provide DNS services for the cluster?
      clusterDNS: true <- set it to false just if you have your own DNS in the baremetal network and you can
configure your names properly
      # Does the provisioning host provide a default gateway for the baremetal network?
      baremetalGateway: true
```

Setup installer node

Install CentOS operating system there. Once you have it, configure your NIC/VLANS properly (management/external, provisioning, baremetal, ipmi). Be sure that you collect the information of interfaces/vlans.

Configure the system properly to run knictl on it: Install knictl

Fetch requirements

Inside knictl path (typically $HOME/go/src/gerrit.akraino.org/kni/installer), run the fetch-requirements command, pointing to the github repo of the site you created

```
./knictl fetch_requirements <site repo URI>
```

For example:

```
./knictl fetch_requirements github.com/akraino-edge-stack/kni-blueprint-pae/tree/master/sites/community.baremetal.
edge-sites.net
```

Prepare manifests

Run the prepare manifests command, using as a parameter the name of your site

./knictl prepare_manifests $SITE_NAME

For example:
./knictl prepare_manifests community.baremetal.edge-sites.net

Remember that the generated files there have a validity of 24 hours. If you don't finish the installation on that time, you'll need to re-run this command.

Deploy masters

```
./knictl deploy_masters $SITE_NAME
```

This will deploy a bootstrap VM and begin to bring up your master nodes. Once the masters have reached the ready state, you can then deploy your workers. You can monitor the process of installation with:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install wait-for bootstrap-complete --dir $HOME/.kni/$SITE_NAME
/baremetal_automation/ocp/
```

When all master nodes are shown as ready, you can start deployment of your workers

Deploy workers

```
./knictl deploy_workers $SITE_NAME
```

This will begin to bring up your worker nodes.
You will need to destroy the bootstrap VM once the *deploy_workers* command is initiated with:

```
virsh destroy <bootstrap_vm_name>
```

Monitor your worker nodes are you normally would during this process. If the deployment doesn't hit any errors, you will then have a working baremetal cluster. You can monitor the state of the cluster with:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install wait-for install-complete --dir $HOME/.kni/$SITE_NAME
/baremetal_automation/ocp/
```

It may happen that the monitor of this process stops at 93%-94%. This is fine, you can just launch again, or simply start using the cluster, as mostly all operators will come online over the time. Follow

https://docs.openshift.com/container-platform/4.4/installing/installing_bare_metal/installing-bare-metal.html#installation-registry-storage-config_installing-bare-metal to fix image registry operator.

Accessing the Cluster

After the deployment finishes, a `kubeconfig` file will be placed inside auth directory:

```
export KUBECONFIG=$HOME/.kni/$SITE_NAME/final_manifests/auth/kubeconfig
```

> *NOTE: When using automated baremetal deployment, the `kubeconfig` will be found here instead:*

```
     export KUBECONFIG=$HOME/.kni/$SITE_NAME/baremetal_automation/ocp/auth/kubeconfig
```

Then cluster can be managed with the kubectl or oc (drop-in replacement with advanced functionality) CLI tools.

To verify a correct setup, you can check again the nodes, and see if masters and workers are ready:

```
$HOME/.kni/$SITE_NAME/requirements/oc get nodes
```

You also can check if the cluster is available:

```
$HOME/.kni/$SITE_NAME/requirements/oc get clusterversion
```

You can also verify that the console is working, the console url is the following:

```
 https://console-openshift-console.apps.$CLUSTER_NAME.$CLUSTER_DOMAIN
```

You can enter the console with **kubeadmin** user and the password that is shown at the end of the install.

libvirt deployment guide

Create site for virtual baremetal

The procedure for virtual baremetal is the same as for the baremetal case, but adding extra flags to indicate that the process is going to be virtual.

First step to start a virtual baremetal deployment is to have a site defined, with all the network and baremetal settings defined in the yaml files. A sample of site using this baremetal automation can be seen here .
In order to define the settings for a site, the first section 00_install-config needs to be used.
Start by creating a kustomization file like the following: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/community.baremetal.edge-sites.net/00_install-config/kustomization.yaml

```
bases:
- git::https://gerrit.akraino.org/r/kni/blueprint-pae.git//profiles/production.baremetal/00_install-config

patches:
- install-config.patch.yaml

patchesJson6902:
- target:
    version: v1
    kind: InstallConfig
    name: cluster
  path: install-config.name.patch.yaml

transformers:
- site-config.yaml
```

In this kustomization file we are patching the default install-config, and also adding some extra files to define networking (site-config.yaml).

install-config.name.patch.yaml: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/testing.baremetal.edge-sites.net/00_install-config/install-config.name.patch.yaml

```
- op: replace
  path: "/metadata/name"
  value: testing <- replace with your cluster name here
```

install-config.patch.yaml : https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/testing.baremetal.edge-sites.net/00_install-config/install-config.patch.yaml

```
apiVersion: v1
kind: InstallConfig
baseDomain: baremetal.edge-sites.net <- domain for your site
compute:
- name: worker
replicas: 2
controlPlane:
  name: master
  platform: {}
  replicas: 1
metadata:
  name: cluster
networking:
  clusterNetworks:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
  apiVIP: 192.168.111.4
  ingressVIP: 192.168.111.3
  dnsVIP: 192.168.111.2
  hosts: {} <- see it's empty, this will be created automatically as it's virtual
pullSecret: 'PULL_SECRET' <- leave like that, it will be replaced in runtime
sshKey: |
  SSH_PUB_KEY <- leave like that, it will be replaced in runtime
```

site-config.yaml: https://github.com/akraino-edge-stack/kni-blueprint-pae/blob/master/sites/testing.baremetal.edge-sites.net/00_install-config/site-config.yaml

```
apiVersion: kni.akraino.org/v1alpha1
kind: SiteConfig
metadata:
  name: notImportantHere
config:
  virtualizedInstall: "true" <- this will tell the installer to deploy with virtual baremetal
provisioningInfrastructure:
  hosts:
    # interface to use for provisioning on the masters
    masterBootInterface: eno1
    # interface to use for provisioning on the workers
    workerBootInterface: eno1
    # interface to use for baremetal on the masters
    masterSdnInterface: eno2
    # interface to use for baremetal on the workers
    workerSdnInterface: eno2

  network:
    # The provisioning network's CIDR
    provisioningIpCidr: 172.22.0.0/24
    # PXE boot server IP
    # DHCP range start (usually provHost/interfaces/provisioningIpAddress + 1)
    provisioningDHCPStart: 172.22.0.11
    provisioningDHCPEnd: 172.22.0.51

    # The baremetal networks's CIDR
    baremetalIpCidr: 192.168.111.0/24
    # Address map
    # bootstrap: baremetalDHCPStart   i.e. 192.168.111.10
    # master-0: baremetalDHCPStart+1  i.e. 192.168.111.11
    # master-1: baremetalDHCPStart+2  i.e. 192.168.111.12
    # master-2: baremetalDHCPStart+3  i.e. 192.168.111.13
    # worker-0: baremetalDHCPStart+5  i.e. 192.168.111.15
    # worker-N: baremetalDHCPStart+5+N
    baremetalDHCPStart: 192.168.111.10
    baremetalDHCPEnd: 192.168.111.50
    # baremetal network default gateway, set to proper IP if /provHost/services/baremetalGateway == false
    # if /provHost/services/baremetalGateway == true, baremetalGWIP with be located on provHost/interfaces
/baremetal
    # and external traffic will be routed through the provisioning host
    baremetalGWIP: 192.168.111.4
    dns:
      # cluster DNS, change to proper IP address if provHost/services/clusterDNS == false
      # if /provHost/services/clusterDNS == true, cluster (IP) with be located on provHost/interfaces/provisioning
      # and DNS functionality will be provided by the provisioning host
      cluster: 192.168.111.3
      # Up to 3 external DNS servers to which non-local queries will be directed
      external1: 10.10.160.1
      external2: 10.10.160.2

  provHost:
    interfaces:
      # Interface on the provisioning host that connects to the provisioning network
      provisioning: dummy0
      # Must be in provisioningIpCidr range
      # pxe boot server will be at port 8080 on this address
      provisioningIpAddress: 172.22.0.1
      # Interface on the provisioning host that connects to the baremetal network
      baremetal: eno1
      # Must be in baremetalIpCidr range
      baremetalIpAddress: 192.168.111.199
      # Interface on the provisioning host that connects to the internet/external network
      external: eno3
    bridges:
      # These bridges are created on the bastion host
      provisioning: provisioning
      baremetal: baremetal
    services:
      # Does the provsioning host provide DHCP services for the baremetal network?
      baremetalDHCP: true
      # Does the provisioning host provide DNS services for the cluster?
      clusterDNS: true
      # Does the provisioning host provide a default gateway for the baremetal network?
      baremetalGateway: true
```

Setup installer node

Install CentOS operating system there. Once you have it, configure your NIC/VLANS properly. You can make use of dummy interfaces if you need it, as the network will all be virtualized.

Configure the system properly to run knictl on it: Install knictl

Fetch requirements

Inside knictl path (typically $HOME/go/src/gerrit.akraino.org/kni/installer), run the fetch-requirements command, pointing to the github repo of the site you created

```
 ./knictl fetch_requirements <site repo URI>
```

For example:

```
./knictl fetch_requirements github.com/akraino-edge-stack/kni-blueprint-pae/tree/master/sites/testing.baremetal.
edge-sites.net
```

Prepare manifests

Run the prepare manifests command, using as a parameter the name of your site

./knictl prepare_manifests $SITE_NAME

For example:
./knictl prepare_manifests testing.baremetal.edge-sites.net

Remember that the generated files there have a validity of 24 hours. If you don't finish the installation on that time, you'll need to re-run this command.

Deploy masters

```
./knictl deploy_masters $SITE_NAME
```

This will deploy a bootstrap VM and begin to bring up your master nodes. Once the masters have reached the ready state, you can then deploy your workers. You can monitor the process of installation with:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install wait-for bootstrap-complete --dir $HOME/.kni/$SITE_NAME
/baremetal_automation/ocp/
```

When all master nodes are shown as ready, you can start deployment of your workers

Deploy workers

```
./knictl deploy_workers $SITE_NAME
```

This will begin to bring up your worker nodes. Monitor your worker nodes are you normally would during this process. If the deployment doesn't hit any errors, you will then have a working baremetal cluster.

You can monitor the state of the cluster with:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install wait-for install-complete --dir $HOME/.kni/$SITE_NAME
/baremetal_automation/ocp/
```

After masters and workers are up, you can apply the workloads using the general procedure with:

```
./knictl apply_workloads $SITE_NAME --kubeconfig $HOME/.kni/$SITE_NAME/baremetal_automation/ocp/auth/kubeconfig
```

Verifying the setup

After the deployment finishes, a `kubeconfig` file will be placed inside auth directory:

```
export KUBECONFIG=$HOME/.kni/$SITE_NAME/final_manifests/auth/kubeconfig
```

> *NOTE: When using automated baremetal deployment, the* `kubeconfig` *will be found here instead:*
>
> `export KUBECONFIG=$HOME/.kni/$SITE_NAME/baremetal_automation/ocp/auth/kubeconfig`

Then cluster can be managed with the kubectl or `oc` (drop-in replacement with advanced functionality) CLI tools.

To verify a correct setup, you can check again the nodes, and see if masters and workers are ready:

```
$HOME/.kni/$SITE_NAME/requirements/oc get nodes
```

You also can check if the cluster is available:

```
$HOME/.kni/$SITE_NAME/requirements/oc get clusterversion
```

You can also verify that the console is working, the console url is the following:

```
 https://console-openshift-console.apps.$CLUSTER_NAME.$CLUSTER_DOMAIN
```

You can enter the console with **kubeadmin** user and the password that is shown at the end of the install.

Developer guide and troubleshooting

Developer guide -See Developer Documentation

Troubleshooting guide - Please see the upstream documentation for details.

Uninstall guide

Manual

When needed, the site can be destroyed with the openshift-install command, using the following syntax:

```
$HOME/.kni/$SITE_NAME/requirements/openshift-install destroy cluster --dir $HOME/.kni/$SITE_NAME/final_manifests
```

Automated (Baremetal / virtual baremetal only)

A baremetal UPI cluster that was deployed using knictl's automation commands (*deploy_masters / deploy_workers)* can be destroyed like so:

```
./knictl destroy_cluster $SITE_NAME
```