Infoblox
NEXT LEVEL NETWORKING

Infoblox IPAM Driver for Kubernetes User's Guide

Overview

CNI (Container Network Interface) is the generic plugin-based networking layer for supporting container runtime environments. CNI-Infoblox is an IPAM driver for CNI that interfaces with Infoblox to provide IP Address Management service.

This guide explains how to install and configure CNI-Infoblox IPAM Driver for Kubernetes. For information on how to use the driver, see *Network and IP Address Management for Containers Using CNI-Infoblox IPAM Driver for Kubernetes*.

Prerequisites

To use the plugin, you need access to the physical or virtual Infoblox DDI product, NIOS or vNIOS. For evaluation purposes, you can download a virtual version of the product from the *Infoblox Download Center* and you can also assign temporary license by logging into the Infoblox DDI appliance console with the `set temp_license` command.

If you are an existing Infoblox customer, you can download appliance images from the *Infoblox Support portal*. For information about downloading and setting up vNIOS, see *Downloading vNIOS* and *Setting Up vNIOS*.

Downloading vNIOS

A vNIOS appliance is the Infoblox virtual appliance that you can download from the *Infoblox Download Center*.

Complete the following steps to download vNIOS

1. Open a web browser and go to *Infoblox Download Center*.
2. Navigate to the section **Infoblox DDI (DNS, DHCP, IPAM)**.
3. Click **Try it Now** to download the Infoblox DDI product.
4. When the registration is complete, you will receive an email with the link that takes you to the Product Evaluation Portal. In the Product Evaluation Portal, under the **Required Downloads** section, download Infoblox DDI for VMware. In the Product Evaluation Portal, you can find download links as well as instructional videos to set up vNIOS.

   Note: Examples, instructions and demonstrations use VMware as the deployment platform. Steps may differ when using other platforms.

   ---

5. After the download is complete, install vNIOS.

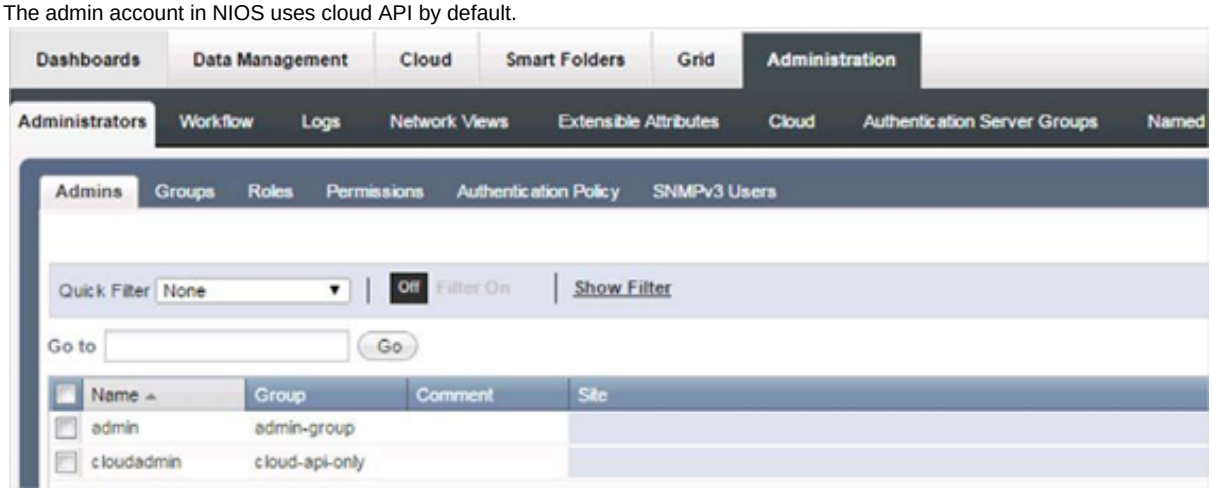
Setting up vNIOS

In vNIOS, you need to set up a user account as described in the procedure below and activate the vNIOS Cloud Network Automation feature.

To activate Cloud Network Automation, follow one of the instructional videos In the Product Evaluation Portal as mentioned in the procedure above. After you activate this feature, you need to add cloud extensible attributes in NIOS as described in *Configuring Cloud Extensible Attributes*.

To set up a user account:

1. In the Product Evaluation Portal, scroll down to **Related Resources** -> **Video 1: Infoblox Cloud Network Automation Installation and Setup**.
2. Activate the vNIOS Cloud Network Automation feature by following the instruction in the later part of the video. You can skip over the section on configuring DHCP and DNS, as well as the section on vRealize Orchestrator.
3. The admin account in NIOS uses cloud API by default.



4. Give **cloud-api admin user** permission to create and modify DNS views. Instructions on how to add permission to **cloud-api-only** group are included in the video. Follow the same instructions to add **All DNS Views** permission under the **DNS Permissions** permission type.

Configuring Cloud Extensible Attributes

CNI-Infoblox IPAM Driver for Kubernetes uses cloud extensible attributes configured in NIOS. When your Cloud Network Automation license is activated, you can define cloud extensible attributes by using the `create-ea-defs` tool in the infoblox/docker-ipam-tools Docker image.

To run the `create-ea-defs` tool, use the following command:

```
docker run infoblox/container-ipam-tool:0.0.1 create-ea-defs --debug --grid-host <infoblox-appliance-ip>
--wapi-username=<username> --wapi-password=<password> --wapi-version=2.5
```

To use the configuration file for `create-ea-defs`, use the following command:

```
docker run -v /etc/infoblox:/etc/infoblox infoblox/container-ipam-tool:0.0.1 create-ea-defs --debug
--conf-file docker-infoblox.conf
```

Configuring CNI-Infoblox IPAM Driver for Kubernetes

You need to update the following driver configurations, based on the vNIOS settings:

- Set **grid-host** to the management IP address of vNIOS.
- Set **WAPI username** and **password** to that for the cloud admin user on vNIOS.

To apply these configurations, edit the `run-daemon.sh` example shell scripts.

Note: By default, CNI-Infoblox IPAM Driver for Kubernetes assumes that the Cloud Network Automation licensed feature is activated in NIOS. If this is not the case, see *Configuring Cloud Extensible Attributes*.

Infoblox WAPI user permissions

Infoblox WAPI user should have the following permissions:

Permission Type	Resource	Resource Type	Permission
[DHCP]	All IPv4 DHCP Fixed Addresses/Reservations	IPv4 DHCP fixed address	RW
[DNS, DHCP, IPAM]	All Hosts	Host	RW
[DHCP, DNS, IPAM]	All IPv4 Host Addresses	IPv4 Host address	RW
[GRID]	All Members	Member	RW
[DHCP, IPAM]	All IPv4 Networks	IPv4 Network	RW
[DHCP, IPAM]	All Network Views	Network view	RW
[CLOUD]	All Tenants	Tenant	RW
[DNS]	All DNS Views	DNS View	RW

Supported Versions

The following are the minimum versions required in each node of kubernetes cluster for testing:

- Host - Ubuntu 16.04.3 LTS
- Docker - 17.12.1-ce
- kubeadm - 1.9.4
- kubectl - 1.9.4
- kubelet - 1.9.4
- kubernetes-cni - 0.6.0-00
- CNI source used to build the plugin and daemon - 0.6.0
- WAPI version - 2.5 and above

NIOS Version: 8.3.x, 8.2.x, 8.1.x.

Installing and Configuring CNI-Infoblox IPAM Driver for Kubernetes

This section provides information about installing and configuring CNI-Infoblox IPAM driver for Kubernetes. This includes the following topics:

Setting up a Cluster

If you are new to Kubernetes, use `kubeadm` to setup a kubernetes cluster. For information about setting up Kubernetes Cluster, see *Install Kubeadm* and *Create cluster*.

Following is the URL to the CNI-Infoblox repository:

https://github.com/infobloxopen/cni-infoblox

Installing CNI-Infoblox IPAM Driver for Kubernetes

It is recommended that you run the **Infoblox IPAM Daemon** as a daemonset in kubernetes cluster.

1. Create the daemonset before starting the plugin.
   `kubectl create -f k8s/cni-infoblox-daemon.yaml`
   A docker image is available in *infoblox/cni-infoblox-daemon:1.0* Docker Hub. This image contains a binary form of cni-infoblox-daemon.

   Note: Update base64 encoded wapi-password in `k8s/cni-infoblox-daemon.yaml`

2. Deploy the cni-infoblox-plugin daemonset in either of the following way:
   a. Infoblox plugin + CNI network configuration file:
      `kubectl create -f k8s/cni-infoblox-plugin.yaml`
   b. Infoblox plugin only:
      `kubectl create -f k8s/cni-infoblox-plugin-without-net-conf.yaml`

You can use the preceding commands to create a cni-infoblox-plugin daemonset in kubernetes cluster. It requires a docker image that is available at *infoblox/cni-infoblox-plugin*. If you use `k8s/cni-infoblox-plugin.yaml`, it will install the infoblox plugin binary and network configuration file in the locations `/opt/cni/bin` and `/etc/cni/net.d` respectively on all worker nodes. If you use `k8s/cni-infoblox-plugin-without-net-conf.yaml`, it will copy the infoblox plugin binary only..

If you want to make any changes in the network configuration, do either of the following:

- Change the network config file contents in the cni-infoblox plugin as shown below:
  `kubectl apply -f k8s/cni-infoblox-plugin.yaml`
- Change the configmap once the daemonset is created:
  `kubectl edit configmap infoblox-cni-cfg --namespace=kube-system`

Note: It takes about one minute to reflect the configmap changes using the `configmap edit` command. You should name the network configuration file with proper order. If there are multiple CNI configuration files in the kubernetes network config directory (i.e. `/etc/cni/net.d`), then the first one in lexicographic order of file name is used.

Example:

In the following example, the filename is given as `infoblox-ipam.conf`, which should match the value of the key `ipam_conf_file_name`.

```
ipam_conf_file_name: infoblox-ipam.conf

 ## Network Config file contents##

 ## This key should match the value of the key 'ipam_conf_file_name'##

 infoblox-ipam.conf: |

{

"name": "infoblox-ipam-network",

"type": "macvlan",

"master":"eth0",

"ipam": {

    "type": "infoblox",

    "subnet": "10.0.0.0/24",

    "gateway":"10.0.0.1",

    "network-view": "cni_view"

    }

}
```

Configuring CNI-Infoblox IPAM Driver for Kubernetes

To instruct the CNI library to execute the Infoblox IPAM plugin for IP allocation from specified network, specify `infoblox` as the IPAM `type` in the CNI network configuration file (`netconf`). CNI configuration files in a kubernetes environment are typically located in `/etc/cni/net.d`

Example:

```
(/etc/cni/net.d/infoblox-ipam.conf)
{
"name": "infoblox-ipam-network",
"type": "macvlan",
"master": "eth0",
"ipam": {
    "type": "infoblox",
    "subnet": "10.0.0.0/24",
    "gateway": "10.0.0.1",
       "network-view": "cni_view"
}
}
```

Note: To run a macvlan network, enable promiscuous mode on the master interface (e.g. `eth0`) for each node in a kubernetes cluster. This can be done using the command `ip link set eth0 promisc on`.

Following are the IPAM attributes:

IPAM Attributes	Description
type (Required)	Specifies the plugin type. This is also the file name of the plugin executable.
subnet (Optional)	Specifies the CIDR to be used for the network. This is a well-known CNI attribute and is used by the driver.
gateway(Optional)	Specifies the gateway for the network. This is a well-known CNI attribute and is simply passed through to CNI. If this attribute is mentioned, the gateway is created.
routes (Optional)	If **type** is set to **bridge**, then the **routes** attribute specifies the routes for the network. This is a well-known CNI attribute and is simply passed through to CNI.
network-view (Optional)	Specifies the Infoblox network view to use for this network. This is an Infoblox IPAM driver specific attribute. Other Infoblox specific attributes are not shown in the example configuration.

Note: The Gateway defined in the configuration file needs to be reserved as a reservation IP. You should not use this reserved IP for other purposes.

Configuring IPAM Daemon

The IPAM Daemon accepts the following command line arguments to set the Infoblox Grid settings, IPAM Driver settings, and IPAM Policy settings, respectively. Each of the IPAM Policy settings will be used when the same setting is not specified in the network configuration file.

You can configure the following settings in `cni-infoblox-daemon.yaml`:

```
## Infoblox Grid Settings ##

--grid-host string
IP of Infoblox Grid Host (default "192.168.124.200")
--wapi-port string
Infoblox WAPI Port (default "443")
--wapi-username string
Infoblox WAPI Username (default "")
--wapi-version string
Infoblox WAPI Version (default "2.5")
--ssl-verify string
Specifies whether (true/false) to verify server certificate. If a file path is specified, it is assumed to
be a certificate file and will be used to verify server certificate. (default "false")
--cluster-name
   User defined cluster name to identify the deployment (default "cluster-1")

## IPAM Driver Settings ##
--socket-dir string
Directory in which Infobox IPAM daemon socket is created (default "/run/cni")
--driver-name string
Name of the IPAM driver. This is the file name used to create Infoblox IPAM daemon socket, and has to match
the name specified as IPAM type in the CNI configuration. (default "infoblox")

## IPAM Policy Settings ##
--network-view string
Infoblox Network View (default "default")
--network string
       Network cidr to be used to assign ip address for pods if cidr info is not provided in cni network
conf file ( default "172.18.0.0/16" )
```

You should pass the WAPI-password via kubernetes secrets. For more information, refer to *K8s-Secrets*.

The following command demonstrates how to encode your password using `base64`.

```
$ echo -n "infoblox" | tr -d '\n' | base64
aW5mb2Jsb3g=


This base64 encoded value must be used in k8s/cni-infoblox-daemon.yaml, as referenced in the following
section:
---
apiVersion: v1
kind: Secret
metadata:
 name: infoblox-secret
 namespace: kube-system
type: Opaque
data:
wapi-password: <UPDATE YOUR ENCODED VALUE>
```

Note: WAPI Version should be 2.5 or above

Configuring CNI Infoblox IPAM plugin

The Infoblox IPAM plugin supports the following IPAM configuration types when assigning IP addresses for pods:

- *One CIDR for Pod across all Nodes*
- *An Unique CIDR for Pod on each Node*

One CIDR for Pod across all Nodes

CNI network conf needs to be identical across all nodes. You can use the `k8s/cni-infoblox-plugin.yaml` file to update the following section and deploy cni-infoblox-plugin daemonset. This will copy the infoblox plugin binary and net conf mentioned in the config map section to all the worker nodes.

Assume your pod network is 10.0.0.0/12, then:

```
infoblox-ipam.conf: |
   {
   "name": "infoblox-ipam-network",
   "type": "macvlan",
   "master":"eth0",
   "ipam": {
       "type": "infoblox",
       "subnet": "10.0.0.0/12",
       "gateway":"10.0.0.1",
       "network-view": "flat_view"
       }
   }
```

or

If the subnet is not mentioned, a network is automatically allocated by CNI-Infoblox daemon:

```
infoblox-ipam.conf: |
   {
   "name": "infoblox-ipam-network",
   "type": "macvlan",
   "master":"eth0",
   "ipam": {
       "type": "infoblox",
       "network-view": "flat_view"
       }
   }
```

An Unique CIDR for Pod on each Node

If you opt for different pod CIDRs on each worker node, the network conf on each node will be different.

The following three parameters should be different:

- Name
- Subnet
- Gateway

Parameter	Description
Name	You cannot have a network with the same name and different subnet as the relevant network is created in Infoblox appliances.
Subnet	The subnet/CIDR and gateway are different because the routes & iptables are configured in the background for each node and its respective subnet/CIDR.
Gateway	

You need to manually update the CNI network conf file by using the `k8s/cni-infoblox-plugin-without-net-conf.yaml` file to deploy cni-infoblox-plugin daemonset. It copies the infoblox plugin binary to all the worker nodes.

Example:
Node 1.

```
infoblox-ipam.conf: |
   {
   "name": "infoblox-ipam-network_10",
   "type": "macvlan",
   "master":"eth0",
   "ipam": {
       "type": "infoblox",
       "subnet": "192.168.10.0/24",
       "gateway":"192.168.10.1",
       "network-view": "node_view"
       }
   }
```

Node 2

```
infoblox-ipam.conf: |
   {
   "name": "infoblox-ipam-network_11",
   "type": "macvlan",
   "master":"eth0",
   "ipam": {
       "type": "infoblox",
       "subnet": "192.168.11.0/24",
       "gateway":"192.168.11.1",
       "network-view": "node_view"
       }
   }
```

1. To use the IPAM Driver, start the daemonset as described in the **Running the IPAM Daemon** section.
2. Put the netconf file and plugin binary in specified locations, as described in the *Configuring CNI-Infoblox IPAM Driver for Kubernetes* and *Configuring CNI Infoblox IPAM plugin* respectively.
3. Test pod connectivity by deploying apps in the kubernetes cluster. Use the following example and save it in `test-app.yaml`.
   Example:
   ```
   #vi test-app.yaml
   apiVersion: apps/v1beta1
   kind: Deployment
   metadata:
   name: test-infoblox-deployment
   spec:
   replicas: 2
   template:
   metadata:
   labels:
   app: test-infoblox
   spec:
   containers:
   - name: test-infoblox
   image: ianneub/network-tools
   command: ["/bin/sh"]
   args: ["-c", "sleep 10000; echo 'I m dying' "]

   kubectl create -f example/test-app.yaml
   ```
   This will start the test-infoblox-deployment with two pods.
4. When the pods appear, you can verify by using the `ifconfig` inside the pod to check whether the IP has been successfully provisioned from Infoblox. To verify the pod connectivity, ping the second pod from the first pod.

Using Existing Network

By default, the required network view & network will be created based on the configuration available in `infoblox-ipam.conf`.

If you want to use the existing network view & networks available on NIOS, update the following extensible attributes:(EAs):

- Network Name
- CMP Type
- Cloud API Owned
- Tenant ID.

Example:
If your network name is `ABCNET` in the network configuration, define the EAs as follows:

```
"Network Name" = "ABCNET"

"CMP Type" = "Kubernetes"

"Cloud API Owned" = "True"

"Tenant ID" = "Testing"
```

Related Documentation

Other Infoblox documentation related to IPAM Driver for Kubernetes:

- *Release Notes for the CNI-Infoblox IPAM Driver for Kubernetes*

See also the following Infoblox NIOS documentation

- *Infoblox NIOS Administrator Guide*

Technical Support

Infoblox Technical Support provides assistance via the Web, e-mail, and telephone. The *Infoblox Support Site* provides access to product documentation and release notes, but requires the user ID and password you receive when you register your product online at: *http://www.infoblox.com/support/customer/evaluation-and-registration*