Infoblox
CONTROL YOUR NETWORK

User Guide
Infoblox IPAM Driver for Docker

Version 1.1

Copyright Statements

Trademark Statements

Company Information

http://www.infoblox.com/contact/

Product Information

Hardware Models

Infoblox Advanced Appliances: PT-1400, PT-1405, PT-2200, PT-2205, PT-2205-10GE, PT-4000, and PT-4000-10GE

Network Insight Appliances: ND-800, ND-805, ND-1400, ND-1405, ND-2200, ND-2205, and ND-4000

Trinzic Appliances: TE-100, TE-810, TE-815, TE-820, TE-825, TE-1410, TE-1415, TE-1420, TE-1425, TE-2210, TE-2215, TE-2220, TE-2225, IB-4010, and IB-4020

Cloud Network Automation: CP-V800, CP-V1400, and CP-V2200

Trinzic Reporting: TR-800, TR-805, TR-1400, TR-1405, TR-2200, TR-2205, and TR-4000

DNS Cache Acceleration Appliances: IB-4030 and IB-4030-10GE

NetMRI: NetMRI-1102-A, NT-1400, NT-2200, and NT-4000

Document Number: 400-0693-001 Rev. A

Document Updated: October 11, 2017

Warranty Information

Your purchase includes a 90-day software warranty and a one year limited warranty on the Infoblox appliance, plus an Infoblox Warranty Support Plan and Technical Support. For more information about Infoblox Warranty information, refer to the Infoblox Web site, or contact Infoblox Technical Support.

Contents

Overview

Docker libnetwork is a robust Container Network Model that provides a consistent programming interface and the required network abstractions for applications. Docker can use third-party IPAM solutions through the libnetwork IPAM driver. Infoblox IPAM Driver for Docker is a Docker libnetwork driver that interfaces with the Infoblox DDI product to provide centralized IP address management services.

Organizations typically have multi-container environments including VMs, physical hosts, etc. This leads to the need to manage IP addresses in a holistic way across the organization. In this complex container deployment, Infoblox IPAM Driver for Docker helps maintain consistency in a very dynamic multi-host environment dealing with network container, network creation and deletion, and IP address allocation and release in a network.

With this approach, Infoblox IPAM Driver for Docker provides solutions to the following use cases:

- Allocating IP addresses on a given Docker network for a microservice container.
- Attaching containers on to pre-defined networks or VLANs within your environment.
- Creating a common shared network across multiple hosts of cooperating applications and associated microservices.
- Creating separate networks for different microservices-based applications across multiple hosts that do not need to interact, therefore isolating the traffic between containers.
- Enabling Docker services to have Infoblox IPAM Driver to allocate IP addresses to service containers using Docker's Swarm mode.

This Guide explains how to install and configure Infoblox IPAM Driver for Docker in the section *Installing and Configuring Infoblox IPAM Driver for Docker* on page 3. For information on how to use the Driver, see *Network and IP Address Management for Containers Using Infoblox IPAM Driver for Docker* on page 4 and *Using Infoblox IPAM Driver for Docker in Swarm Mode with Swarm Scope Networks* on page 6.

Before Installing Infoblox IPAM Plugin for Docker

To use the Driver, 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 at *https://www.infoblox.com/infoblox-download-center*. If you are an existing Infoblox customer, you can download it from the Support site. For information about downloading and setting up vNIOS, see *Downloading vNIOS* and *Setting Up vNIOS*.

As another prerequisite, you must have a working installation of Docker. You also need to create a Docker host to interact with vNIOS. For information, see Docker documentation at *https://docs.docker.com/engine/userguide/*. For installation of Docker Swarm, see *https://docs.docker.com/engine/swarm/swarm-tutorial/*.

System Requirements

Following are the system requirements for the Infoblox IPAM Driver for Docker version 1.1:

- Docker Version: 17.06.0-ce
- Ubuntu Version: 16.04
- NIOS Version: 8.2.x, 8.1.x, 8.0.x, 7.3.x, 7.2.x, 7.1.x

Downloading vNIOS

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

Do the following:

1. Point your browser to *https://www.infoblox.com/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: It is strongly recommended that you download the VMware version of the product as VMware is the platform on which the videos are based.

5. After the download is complete, install vNIOS.

See the next section for information on setting up vNIOS.

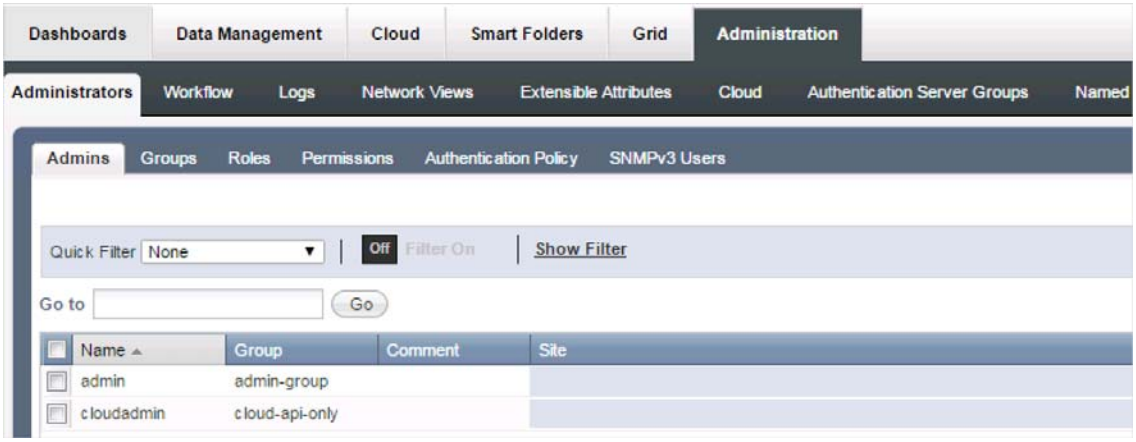
Setting Up vNIOS

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* on page 2.

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. In NIOS, create an admin account that uses cloud API.



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

Infoblox IPAM Driver for Docker 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.

This tool adds the following extensible attribute definitions in NIOS:

- Cloud API Owned
- CMP Type
- Tenant ID
- VM ID
- Docker-Plugin-Lock
- Docker-Plugin-Lock-Time

To run the “create-ea-defs” tool:

```
docker run infoblox/docker-ipam-tools:1.1.0 create-ea-defs --debug --grid-host 10.120.21.150
--wapi-username=admin --wapi-password=infoblox --wapi-version=2.3
```

To use the configuration file for “create-ea-defs”:

```
docker run -v /etc/infoblox:/etc/infoblox infoblox/docker-ipam-tools:1.1.0 create-ea-defs --debug
--conf-file docker-infoblox.conf
```

Installing and Configuring Infoblox IPAM Driver for Docker

You can install Infoblox IPAM Driver for Docker by pulling the infoblox/ipam-plugin from the Docker store and setting its environment variables. In the Docker swarm mode, you must install the Driver on all the swarm nodes.

To configure the Driver, use any of the following:

- Configuration file. For information, see *Installing and Configuring Infoblox IPAM Driver for Docker Using Configuration File* on page 3.
- Environment variables. For information, see *Installing and Configuring Infoblox IPAM Driver for Docker Using Environment Variables* on page 4.

Note the following configurations of the Driver that you need to update when running the Driver, based on the vNIOS settings:

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

To apply these configurations, edit the run.sh and run-container.sh shell scripts.

Note: By default, Infoblox IPAM Driver for Docker assumes that the Cloud Network Automation licensed feature is activated in NIOS. If this is not the case, see *Configuring Cloud Extensible Attributes* on page 2.

Installing and Configuring Infoblox IPAM Driver for Docker Using Configuration File

To install and configure the Driver, add parameters in the configuration file and set the CONF_FILE_NAME environment variable to the configuration file name. For the list of configuration file parameters, see *Appendix A, "Configuration Options",* on page 9.

Do the following:

1. Create a file `docker-infoblox.conf` (configurable via the `CONF_FILE_NAME` parameter) in `/etc/infoblox/` directory and specify the configuration options in the file.

   A sample configuration file for Infoblox IPAM Driver for Docker would look as follows:

   ```
   [grid-config]
   grid-host="10.120.21.150"
   wapi-port="443"
   wapi-username="infoblox"
   wapi-password="infoblox"
   wapi-version="2.0"
   ssl-verify="false"
   http-request-timeout=60
   http-pool-connections=10

   [ipam-config]
   global-view="global_view"
   global-network-container="172.18.0.0/16"
   ```

```
global-prefix-length=24
local-view="local_view"
local-network-container="192.168.0.0/20,192.169.0.0/22"
local-prefix-length=25
```

2. Set the `CONF_FILE_NAME` variable to the configuration file name while installing the Driver:

```
$ docker plugin install --alias infoblox store/infoblox/ipam-plugin:1.1.0 \
CONF_FILE_NAME=docker-infoblox.conf

Plugin "infoblox/ipam-plugin:1.1.0" is requesting the following privileges:
 - network: [host]
 - mount: [/etc/infoblox]
 - mount: [/var/run]
Do you grant the above permissions? [y/N]
```

The Driver requests the following privileges:

- Access host network.
- Mount the directory `/etc/infoblox` on the host as a volume for the container to read its configuration file.
- Mount the directory`/var/run` on the host as a volume for the container to access the Docker socket file.

To bypass the privileges request prompt, use the `--grant-all-permissions` option:

```
$ docker plugin install --grant-all-permissions --alias infoblox \
store/infoblox/ipam-plugin:1.1.0 CONF_FILE_NAME=docker-infoblox.conf
```

Installing and Configuring Infoblox IPAM Driver for Docker Using Environment Variables

You can install and configure the Driver by setting all the Driver environment variables while installing it:

```
$ docker plugin install --grant-all-permissions --alias infoblox \
store/infoblox/ipam-plugin:1.1.0 GRID_HOST=10.120.21.150 \
WAPI_USERNAME=admin WAPI_PASSWORD=infoblox GLOBAL_VIEW=global_view \
GLOBAL_NETWORK_CONTAINER=172.18.0.0/16 LOCAL_VIEW=local_view \
LOCAL_NETWORK_CONTAINER=192.168.0.0/20 LOCAL_PREFIX_LENGTH=25
```

For the list of environment variables, see *Appendix A, "Configuration Options",* on page 9.

To override the configuration file option with the Driver environment variable, use the following:

```
$ docker plugin install --grant-all-permissions --alias infoblox \
store/infoblox/ipam-plugin:1.1.0 CONF_FILE_NAME=docker-infoblox.conf \
LOCAL_NETWORK_CONTAINER=172.16.10.0/24
```

Here `LOCAL_NETWORK_CONTAINER` overrides the `local-network-container` option in the configuration file.

In order to set the Driver log level as debug, set the `DEBUG` variable:

```
$ docker plugin install --grant-all-permissions --alias infoblox \
store/infoblox/ipam-plugin:1.1.0 CONF_FILE_NAME=docker-infoblox.conf DEBUG=true
```

Network and IP Address Management for Containers Using Infoblox IPAM Driver for Docker

To start using the Driver, you should first create a Docker network specifying the Driver using the `--ipam-driver` option:

```
$ docker network create --ipam-driver=infoblox:latest priv-net
```

This creates a Docker network called “priv-net” which uses “infoblox” as the IPAM driver and the default “bridge” driver as the network driver. A network will be automatically allocated from the list of network containers specified during the Driver installation.

If necessary, you can create a container without using the IPAM Driver:

```
$ docker run --net net-1 -d --name=test-container ubuntu /bin/sh -c "while true; do echo Hello world;
sleep 1; done" 8a2c469f3841913fed26129fdfa1fd18041526f50034942e2fcdbf4e6817c56e
```

To connect the network and container, you can use the `network connect` option:

```
$ docker network connect net-2 test-container
```

or

```
$ docker ps
CONTAINER ID   IMAGE   COMMAND                  CREATED        STATUS      PORTS   NAMES
8a2c469f3841   ubuntu  "/bin/sh -c 'while..."   2 minutes ago  Up 2 minutes        test-container
$ docker network connect net-2 8a2c469f3841
```

To disconnect the network and container, you can use the `network disconnect` option:

```
$ docker network disconnect net-2 test-container
```

or

```
$ docker network connect net-2 8a2c469f3841
```

For the list of command line input parameters and configuration file options, see *Appendix A, "Configuration Options",* on page 9.

By default, the network will be created using the default prefix length specified during the Driver installation. You can override this using the `--ipam-opt` option. For example:

```
$ docker network create --ipam-driver=infoblox:latest --ipam-opt="prefix-length=24" priv-net-2
```

Additionally, if you are deploying containers in a cluster on different hosts, you can specify “network-name” using the `--ipam-opt` option. This will be used as an identifier so that Docker networks created on different Docker hosts can share the same IP address space. For example:

```
$ docker network create --ipam-driver=infoblox:latest --ipam-opt="network-name=blue" blue-net
```

This will allocate a network, for example, 192.168.10.0/24, from the default address pool. Additionally, the network will be tagged in Infoblox with the network name “blue”. When the same command is issued on a different host, the Driver will look for a network on Infoblox tagged with the same name, “blue”, and will share the same network, 192.168.10.0/24, instead of allocating a new one.

After the network is created, Docker containers can be started attaching to the “priv-net” network created above. For example, the following command runs the container from the “alpine” image:

```
$ docker run -it --network priv-net alpine /bin/sh

/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
49: eth0@if50: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ae:aa:e4:1c brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.2/25 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:aeff:feaa:e41c/64 scope link
       valid_lft forever preferred_lft forever
/ #
```

When the container is deployed, verify that the IP address assigned to the container is allocated by the Driver from NIOS by using the `ifconfig` command.

In order to use an existing NIOS network as a Docker network, add the extensible attribute “Network Name” to that network and then pass the EA value as `ipam-opt` while creating the Docker network. For example, if there exists a network 172.56.121.0/24 with gateway 172.56.121.1, add the EA “Network Name” with the value “SomeNetwork” for the network in NIOS. Then create a Docker network:

```
docker network create --ipam-driver=infoblox:latest --subnet 172.56.121.0/24 --gateway 172.56.121.1
--ipam-opt="network-name
```

When the network is no longer needed, you can delete it with the Docker `network rm` command. At that point, Infoblox IPAM Driver for Docker will receive a `ReleasePool` API call so that the subnet can be de-allocated.

Using Infoblox IPAM Driver for Docker in Swarm Mode with Swarm Scope Networks

Currently, the Infoblox IPAM Driver for Docker supports only the MACVLAN network driver with swarm scope. Before using the Driver in the swarm mode, make sure that it is installed on all swarm nodes.

Note: The minimum Docker version required for use with MACVLAN driver in swarm mode is 17.06.0.

To use the Driver in the swarm mode:

1. Create a config-only network on all the nodes:

```
sudo docker network create --config-only -o parent=eth1 --ipam-driver=infoblox:latest
--ipam-opt="network-name=docker-macvlan" mv-config
```

2. Create MACVLAN network with swarm scope on the swarm manager node:

```
sudo docker network create -d macvlan --scope=swarm --config-from mv-config --attachable
swarm-macvlan
```

3. Run service with the swarm network:

```
sudo docker service create --replicas 3 --network swarm-macvlan --name swarm-macvlan-test
chrch/docker-pets:1.0
```

4. Verify the IP addresses allocated to the containers of the service:

```
master $ sudo docker network inspect --verbose --format '{{json .Services}}' swarm-macvlan | python
-m json.tool
{
    "swarm-macvlan-test": {
        "LocalLBIndex": 262,
        "Ports": [],
        "Tasks": [
            {
                "EndpointID": "18c19d799d117a8afa20fcf5bb51c8927a4921abe121913e0b080284da459080",
                "EndpointIP": "192.168.9.196",
                "Info": null,
                "Name": "swarm-macvlan-test.2.xqlym8mbvpo9qw6g9b6bwbm5j"
            },
            {
                "EndpointID": "ac6d449822bac91916ed2aa9ab6dd02001cd324ec902fc14358cdebad3f9ca5e",
                "EndpointIP": "192.168.9.198",
                "Info": null,
                "Name": "swarm-macvlan-test.3.kopddbmdmavmmd1mbmo4s9i4v"
            },
            {
                "EndpointID": "c399388cbc2ca94c5a64bb0c7fef6d398d58baf01c6dec3ede26b58f976c0eb7",
                "EndpointIP": "192.168.9.197",
                "Info": null,
                "Name": "swarm-macvlan-test.1.cfjzh0hnnroguzire8xpskjrb"
            }
        ],
        "VIP": "<nil>"
    }
}
```

5. Verify the connectivity between the containers by running these commands on the swarm manager node:

```
master $ sudo docker service ls
ID              NAME              MODE             REPLICAS          IMAGE                PORTS
huj54859rbgr       swarm-macvlan-test   replicated        3/3             chrch/docker-pets:1.0


master $ sudo docker service ps huj54859rbgr
```

```
ID                  NAME                  IMAGE                  NODE              DESIRED STATE
CURRENT STATE          ERROR                 PORTS
cfjzh0hnnrog        swarm-macvlan-test.1  chrch/docker-pets:1.0  worker-1          Running
Running 7 days ago
xqlym8mbvpo9        swarm-macvlan-test.2  chrch/docker-pets:1.0  worker-2          Running
Running 7 days ago
kopddbmdmavm        swarm-macvlan-test.3  chrch/docker-pets:1.0  master            Running
Running 7 days ago


master $ sudo docker exec -it swarm-macvlan-test.3.kopddbmdmavmmd1mbmo4s9i4v sh
/app # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
40: eth0@if3: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 02:42:05:4d:d0:a1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.9.198/26 scope global eth0
       valid_lft forever preferred_lft forever
/app #
/app # ping swarm-macvlan-test.1.cfjzh0hnnroguzire8xpskjrb
PING swarm-macvlan-test.1.cfjzh0hnnroguzire8xpskjrb (192.168.9.197): 56 data bytes
64 bytes from 192.168.9.197: seq=0 ttl=64 time=0.868 ms
64 bytes from 192.168.9.197: seq=1 ttl=64 time=0.728 ms
64 bytes from 192.168.9.197: seq=2 ttl=64 time=0.759 ms
64 bytes from 192.168.9.197: seq=3 ttl=64 time=0.783 ms

--- swarm-macvlan-test.1.cfjzh0hnnroguzire8xpskjrb ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.728/0.784/0.868 ms
/app #
/app # ping swarm-macvlan-test.2.xqlym8mbvpo9qw6g9b6bwbm5j
PING swarm-macvlan-test.2.xqlym8mbvpo9qw6g9b6bwbm5j (192.168.9.196): 56 data bytes
64 bytes from 192.168.9.196: seq=0 ttl=64 time=1.129 ms
64 bytes from 192.168.9.196: seq=1 ttl=64 time=0.427 ms
64 bytes from 192.168.9.196: seq=2 ttl=64 time=0.636 ms
64 bytes from 192.168.9.196: seq=3 ttl=64 time=0.744 ms
64 bytes from 192.168.9.196: seq=4 ttl=64 time=0.655 ms

--- swarm-macvlan-test.2.xqlym8mbvpo9qw6g9b6bwbm5j ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.427/0.718/1.129 ms
/app #
/app # curl swarm-macvlan-test.2.xqlym8mbvpo9qw6g9b6bwbm5j:5000
<html>
  <head>
    <link rel='stylesheet' type='text/css' href="../static/style.css">
    <title>Docker PaaS</title>
  </head> ...
```

Note: In the swarm mode, if the Docker network is created without specifying a gateway address, Infoblox IPAM Driver for Docker creates a new gateway address for each host. If a gateway address is specified while creating the network, then the Driver uses that address as the gateway on all the nodes.

Logging

You can find the logs of Infoblox IPAM Driver for Docker in the Docker daemon logs.

To check the logs, identify the Driver ID:

```
$ docker plugin inspect infoblox:latest -f '{{ .ID }}'

980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
```

Search for this ID in the Docker daemon logs (default is `/var/log/syslog`) to find the Driver logs:

```
$ grep 'plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3' /var/log/syslog

Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Loading IPAM Configuration from the file\" "
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Found Configuration file
/etc/infoblox/docker-infoblox.conf"
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info msg="\" "
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Loading IPAM Configuration from the environment
variables\" " plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Configuration successfully loaded"
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info msg="\" "
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Socket File:
'/run/docker/plugins/infoblox.sock'\" "
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
Jun  8 16:08:40 ishant dockerd[23649]: time="2017-06-08T16:08:40+05:30" level=info
msg="time=\"2017-06-08T10:38:40Z\" level=info msg=\"Docker id is
'DVZR:HNJZ:42OG:XTRO:YOHD:VDYA:EBKK:UDO7:ILEA:JF7R:KYGG:QCIO'"
plugin=980b5a3befebc1a64d6d788b4c1e78676ed3e2632e78b9b38a370a9e71d73ee3
```

Related Documentation

Other Infoblox IPAM Driver for Docker documentation:

- *Release Notes for the Infoblox IPAM Driver for Docker*

See also related Infoblox NIOS documentation:

- *Infoblox NIOS Administrator Guide*
- *Infoblox CLI Guide*
- *Infoblox API Documentation*

Technical Support

Infoblox Technical Support provides assistance via the Web, e-mail, and telephone. The Infoblox Support web site at https://support.infoblox.com 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.

Appendix A Configuration Options

Following is the list of configuration file parameters and environment variables for Infoblox IPAM Driver for Docker.

Environment Variable	Configuration File Option	Type	Description
CONF_FILE_NAME	—	—	Configuration file name in /etc/infoblox directory.
DEBUG	—	—	Log level to debug.
DOCKER_API_VERSION	—	—	Docker API version to use. The default is 1.22.
GRID_HOST	grid-host	string	The IP address of the Infoblox Grid host.
GLOBAL_NETWORK_CONTAINER	global-network-container	string	The container from which subnets will be allocated when no subnet is specified during network creation. The default is “172.18.0.0/16”.
GLOBAL_PREFIX_LENGTH	global-prefix-length	unit	The default CIDR prefix length when allocating a global subnet. The default is “24”.
GLOBAL_VIEW	global-view	string	Infoblox network view for global address space. The default is “default”.
LOCAL_NETWORK_CONTAINER	local-network-container	string	The container from which subnets will be allocated when no subnet is specified during network creation. The default is “192.168.0.0/16”.
LOCAL_PREFIX_LENGTH	local-prefix-length	unit	The default CIDR prefix length when allocating a local subnet. The default is 24.
LOCAL_VIEW	local-view	string	Infoblox network view for local address space. The default is “default”.
SSL_VERIFY	ssl-verify	string	Specifies whether to verify server certificate (true/false). If a file path is specified, it is assumed to be a certificate file and is used to verify server certificate.
WAPI_PASSWORD	wapi-password	string	Infoblox WAPI user account password.
WAPI_PORT	wapi-port	string	Infoblox WAPI port. The default is “443”.
WAPI_USERNAME	wapi-username	string	Infoblox WAPI user account name.
WAPI_VERSION	wapi-version	string	Infoblox WAPI version. The default is “2.0”.
HTTP_REQUEST_TIMEOUT	http-request-timeout	string	Infoblox WAPI request timeout in seconds. The default is “60”.
HTTP_POOL_CONNECTIONS	http-pool-connections	string	Infoblox WAPI request connection pool size. The default is “10”.