Infoblox Guide

Topics

This guide describes how to use Ansible with the Infoblox Network Identity Operating System (NIOS). With Ansible integration, you can use Ansible playbooks to automate Infoblox Core Network Services for IP address management (IPAM), DNS, and inventory tracking.

You can review simple example tasks in the documentation for any of the :ref:`NIOS modules <nios_net tools_modules>` or look at the Use cases with modules section for more elaborate examples. See the Infoblox website for more information on the Infoblox product.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 11); *backlink*
>
> Unknown interpreted text role "ref".

> **Note**
>
> You can retrieve most of the example playbooks used in this guide from the network-automation/infoblox_ansible GitHub repository.

Prerequisites

Before using Ansible `nios` modules with Infoblox, you must install the `infoblox-client` on your Ansible control node:

```
$ sudo pip install infoblox-client
```

> **Note**
>
> You need an NIOS account with the WAPI feature enabled to use Ansible with Infoblox.

Credentials and authenticating

To use Infoblox `nios` modules in playbooks, you need to configure the credentials to access your Infoblox system. The examples in this guide use credentials stored in `<playbookdir>/group_vars/nios.yml`. Replace these values with your Infoblox credentials:

```
---
nios_provider:
  host: 192.0.0.2
  username: admin
  password: ansible
```

NIOS lookup plugins

Ansible includes the following lookup plugins for NIOS:

- :ref:`nios <nios_lookup>` Uses the Infoblox WAPI API to fetch NIOS specified objects, for example network views, DNS views, and host records.

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 46); *backlink*
  >
  > Unknown interpreted text role "ref".

- :ref:`nios_next_ip <nios_next_ip_lookup>` Provides the next available IP address from a network. You'll see an example of this in Creating a host record.

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 47); *backlink*
  >
  > Unknown interpreted text role "ref".

- :ref:`nios_next_network <nios_next_network_lookup>` - Returns the next available network range for a network-container.

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 48); *backlink*
  >
  > Unknown interpreted text role "ref".

You must run the NIOS lookup plugins locally by specifying `connection: local`. See :ref:`lookup plugins <lookup_plugins>` for more detail.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 50); ***backlink***

Unknown interpreted text role "ref".

Retrieving all network views

To retrieve all network views and save them in a variable, use the :ref:`set_fact <set_fact_module>` module with the :ref:`nios <nios_lookup>` lookup plugin:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 56); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 56); ***backlink***

Unknown interpreted text role "ref".

```
---
- hosts: nios
  connection: local
  tasks:
    - name: fetch all networkview objects
      set_fact:
        networkviews: "{{ lookup('nios', 'networkview', provider=nios_provider) }}"

    - name: check the networkviews
      debug:
        var: networkviews
```

Retrieving a host record

To retrieve a set of host records, use the `set_fact` module with the `nios` lookup plugin and include a filter for the specific hosts you want to retrieve:

```
---
- hosts: nios
  connection: local
  tasks:
    - name: fetch host leaf01
      set_fact:
         host: "{{ lookup('nios', 'record:host', filter={'name': 'leaf01.ansible.com'}, provider=nios_provider) }}"

    - name: check the leaf01 return variable
      debug:
        var: host

    - name: debug specific variable (ipv4 address)
      debug:
        var: host.ipv4addrs[0].ipv4addr

    - name: fetch host leaf02
      set_fact:
        host: "{{ lookup('nios', 'record:host', filter={'name': 'leaf02.ansible.com'}, provider=nios_provider) }}"

    - name: check the leaf02 return variable
      debug:
        var: host
```

If you run this `get_host_record.yml` playbook, you should see results similar to the following:

System Message: WARNING/2 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 107)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    $ ansible-playbook get_host_record.yml

    PLAY [localhost] ***************************************************************************************

    TASK [fetch host leaf01] ******************************************************************************
    ok: [localhost]

    TASK [check the leaf01 return variable] *************************************************************
    ok: [localhost] => {
    < ...output shortened...>
        "host": {
            "ipv4addrs": [
                {
                    "configure_for_dhcp": false,
                    "host": "leaf01.ansible.com",
                }
            ],
            "name": "leaf01.ansible.com",
            "view": "default"
        }
    }
```

```
TASK [debug specific variable (ipv4 address)] ******************************************************
ok: [localhost] => {
    "host.ipv4addrs[0].ipv4addr": "192.168.1.11"
}

TASK [fetch host leaf02] ******************************************************************************
ok: [localhost]

TASK [check the leaf02 return variable] *************************************************************
ok: [localhost] => {
< ...output shortened...>
    "host": {
        "ipv4addrs": [
            {
                "configure_for_dhcp": false,
                "host": "leaf02.example.com",
                "ipv4addr": "192.168.1.12"
            }
        ],
    }
}

PLAY RECAP ******************************************************************************************
localhost                  : ok=5    changed=0    unreachable=0    failed=0
```

The output above shows the host record for `leaf01.ansible.com` and `leaf02.ansible.com` that were retrieved by the `nios` lookup plugin. This playbook saves the information in variables which you can use in other playbooks. This allows you to use Infoblox as a single source of truth to gather and use information that changes dynamically. See :ref:`playbooks_variables` for more information on using Ansible variables. See the :ref:`nios <nios_lookup>` examples for more data options that you can retrieve.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 156); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 156); *backlink*

Unknown interpreted text role "ref".

You can access these playbooks at Infoblox lookup playbooks.

Use cases with modules

You can use the `nios` modules in tasks to simplify common Infoblox workflows. Be sure to set up your :ref:`NIOS credentials<nios_credentials>` before following these examples.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 163); *backlink*

Unknown interpreted text role "ref".

Configuring an IPv4 network

To configure an IPv4 network, use the :ref:`nios_network <nios_network_module>` module:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 168); *backlink*

Unknown interpreted text role "ref".

```
---
- hosts: nios
  connection: local
  tasks:
    - name: Create a network on the default network view
      nios_network:
        network: 192.168.100.0/24
        comment: sets the IPv4 network
        options:
          - name: domain-name
            value: ansible.com
        state: present
        provider: "{{nios_provider}}"
```

Notice the last parameter, `provider`, uses the variable `nios_provider` defined in the `group_vars/` directory.

Creating a host record

To create a host record named *leaf03.ansible.com* on the newly-created IPv4 network:

```
---
- hosts: nios
  connection: local
  tasks:
    - name: configure an IPv4 host record
      nios_host_record:
        name: leaf03.ansible.com
        ipv4addrs:
          - ipv4addr:
              "{{ lookup('nios_next_ip', '192.168.100.0/24', provider=nios_provider)[0] }}"
        state: present
provider: "{{nios_provider}}"
```

Notice the IPv4 address in this example uses the :ref:`nios_next_ip <nios_next_ip_lookup>` lookup plugin to find the next available IPv4 address on the network.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 208); ***backlink***

Unknown interpreted text role "ref".

Creating a forward DNS zone

To configure a forward DNS zone use, the `nios_zone` module:

```
---
- hosts: nios
  connection: local
  tasks:
    - name: Create a forward DNS zone called ansible-test.com
      nios_zone:
        name: ansible-test.com
        comment: local DNS zone
        state: present
        provider: "{{ nios_provider }}"
```

Creating a reverse DNS zone

To configure a reverse DNS zone:

```
---
- hosts: nios
  connection: local
  tasks:
    - name: configure a reverse mapping zone on the system using IPV6 zone format
      nios_zone:
        name: 100::1/128
        zone_format: IPV6
        state: present
        provider: "{{ nios_provider }}"
```

Dynamic inventory script

You can use the Infoblox dynamic inventory script to import your network node inventory with Infoblox NIOS. To gather the inventory from Infoblox, you need two files:

- infoblox.yaml - A file that specifies the NIOS provider arguments and optional filters.
- infoblox.py - The python script that retrieves the NIOS inventory.

Note

Please note that the inventory script only works when Ansible 2.9, 2.10 or 3 have been installed. The inventory script will eventually be removed from community.general, and will not work if *community.general* is only installed with *ansible-galaxy collection install*. Please use the inventory plugin from infoblox.nios_modules instead.

To use the Infoblox dynamic inventory script:

1. Download the `infoblox.yaml` file and save it in the `/etc/ansible` directory.
2. Modify the `infoblox.yaml` file with your NIOS credentials.
3. Download the `infoblox.py` file and save it in the `/etc/ansible/hosts` directory.
4. Change the permissions on the `infoblox.py` file to make the file an executable:

```
$ sudo chmod +x /etc/ansible/hosts/infoblox.py
```

You can optionally use `./infoblox.py --list` to test the script. After a few minutes, you should see your Infoblox inventory in JSON format. You can explicitly use the Infoblox dynamic inventory script as follows:

```
$ ansible -i infoblox.py all -m ping
```

You can also implicitly use the Infoblox dynamic inventory script by including it in your inventory directory (`etc/ansible/hosts` by default). See :ref:`dynamic_inventory` for more details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 279); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_infoblox.rst`, line 281)

Unknown directive type "seealso".

```
.. seealso::

  `Infoblox website <https://www.infoblox.com//>`_
      The Infoblox website
  `Infoblox and Ansible Deployment Guide <https://www.infoblox.com/resources/deployment-guides/infoblox-and-ansibl
      The deployment guide for Ansible integration provided by Infoblox.
  `Infoblox Integration in Ansible 2.5 <https://www.ansible.com/blog/infoblox-integration-in-ansible-2.5>`_
      Ansible blog post about Infoblox.
  :ref:`Ansible NIOS modules <nios_net tools_modules>`
```

```
    The list of supported NIOS modules, with examples.
`Infoblox Ansible Examples <https://github.com/network-automation/infoblox_ansible>`_
    Infoblox example playbooks.
```