

DEPLOYMENT GUIDE

# Automate Infoblox Infrastructure Using Ansible



# Table of Contents

Overview	3
Need for Network Automation	3
Ansible	3
Ansible Offerings	3
Usage	3
Key terms	4
Infoblox Collections	4
Ansible-Infoblox Integration	5
Modules	5
Plugins	5
Use Cases	5
Manage DNS Records, Networks and IP Addresses for VMs	5
Automate Deployment of Virtual Infoblox Appliances	5
Deployment	6
Requirements	6
Initial Setup	6
Ansible	6
Picking an Ansible Version	6
Installation	7
Verify Installation	7
Inventory	7
Infoblox	7
Supported Versions	7
Cloud Admin user (Optional)	8
Ansible-Infoblox Integration	10
Getting Started	10
Installing the Infoblox Collections	10
Installation from Ansible Galaxy	10
Installation from Git	11
Writing the Playbooks	11
Create a Network View	12
Create a Network	12
Create a Host Record	13
Create an A Record	13

Add entry for a Grid Member	14
Delete A Record	14
Delete Network	15
Delete Member	15
Sample lookup plugin	16
Preparing your Playbooks	16
Running your Playbooks	16
URI Module	16
Example URI module usage to start DNS services on the Grid	17
Conclusion	17
Additional Information	18
Appendix	18
Troubleshooting	18
NIOS Module Command Help	18

## Overview

Environments are becoming extremely dynamic as virtualization of hardware becomes more and more prevalent. To keep up with that, many organizations depend heavily on tools to automate, or orchestrate, tasks as much as possible.

Automation is an essential and strategic component of modernization and digital transformation. Modern, dynamic environments need a new type of management solution that can improve speed, scale and stability across the enterprise IT environment.

## Need for Network Automation

Traditionally, IP Address Management (IPAM) has been done using spreadsheets. The network administrators maintained the IP address landscape, where they manually added or deleted IP addresses from the spreadsheets whenever hosts are added or deleted from the network. That doesn't work in today's dynamic environments and makes the entire IPAM lifecycle management tedious and prone to error. In large organizations, end-users might have to submit internal tickets for IP address record fulfillments that can take hours, if not days. And then there is the cleanup – again, a time consuming and error-prone effort. The result? These legacy approaches often delay workload deployments and diminish the automation and orchestration advantages that organizations wanted.

## Ansible

Ansible is a simple, yet powerful IT automation engine that thousands of companies are using to drive complexity out of their environments and accelerate DevOps initiatives.

It is used for IT tasks such as configuration management, application deployment, intra-service orchestration and provisioning. It is both light weight and simple to deploy, manage and use. The Ansible platform makes it easy for administrators and developers to automate many tasks, including applying updates to machines on the network to managing devices on the network.

## Ansible Offerings

Ansible has three offerings:

- **Ansible Project:** A free, open-source automation product that is built by the community ([ansible.com/community](https://ansible.com/community)) for the benefit of the community.
- **Ansible Engine:** Open-source technology that organizations can use to access the tools and innovations available from the underlying Ansible technology in a hardened, enterprise-grade manner. Ansible Engine relies on the massive, global community behind the Ansible project, and adds in the capabilities and assurance from Red Hat.
- **Ansible Tower:** An enterprise offering which gives you a graphical interface and enables integration with other services and tools. [Tower](#) gives permission control and will also save a record of all Ansible playbook activity, useful for auditing purposes.

In this deployment guide, we use the Ansible project.

## Usage

You can leverage the capabilities of Ansible in multiple ways:

- **Ad-Hoc:** Issue ansible tasks direct from the command line. This is a good place to start to understand the basics of what Ansible can do prior to learning the playbooks language – ad-hoc commands can also be used to do quick things that you might not necessarily want to write a full playbook for.
- **Playbooks:** These are automation scripts. Playbooks are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process.
- **Automation Framework:** Requires the Ansible Tower.

- **Check Mode:** An option for running ad-hoc commands or playbooks without making changes. This deployment guide walks you through working with playbooks.

## Key terms

- **Controller Machine:** The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.
- **Inventory:** An initialization file that contains information about the servers you are managing.
- **Host:** In Ansible, a host is a remote machine that is assigned to individual variables and they are further grouped together. Each host has a dedicated name or unique IP address to make its identification easy and quick. They can be given simple port number too if you don't have to access them over SSH connection.
- **Ansible automation language:** The structure used when writing playbooks and other resources for Ansible. The Ansible automation language uses [YAML](#) and is intended to be both human and machine readable.
- **Playbook:** The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format. These are plain text files written in the Ansible automation language which describe the intended end-state of a deployment or task being executed.
- **Play:** A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.
- **Task:** A block that defines a single procedure to be executed, e.g. Install a package. It is used within a play to call modules and run-in order.
- **Module:** Also referred to as 'task plugins' or 'library plugins', a module typically abstracts a system task, like dealing with packages or creating and changing files. Ansible has a multitude of built-in modules, but you can also create custom ones.
- **Role:** A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.
- **Facts:** Global variables containing information about the system, like network interfaces or operating system.
- **Plug-ins:** They are the special pieces of code that help to write code quickly. Plug-ins automate the development tasks and help to speed up the deployment work to the maximum level.
- **Ansible Galaxy:** This refers to the [Galaxy](#) website where users can share roles, and to a command line tool for installing, creating and managing roles.
- **Collections:** An Ansible Content Collection, or "collection" for short, is a new directory structure, and complementary tooling in Ansible to consume content from that structure. This new structure accommodates multiple types of content, such as modules, plugins, roles, and others in a singular portable format.

## Infoblox Collections

The [Infoblox NIOS Collection for Ansible Automation](#) Platform is a package of modules and plugins that allows managing Infoblox Network Identity Operating System (NIOS) objects and functions through APIs leveraging Ansible playbooks.

Dynamic inventory is one of the powerful features in Red Hat Ansible Tower, which allows Ansible to query external systems and use the response data to construct its inventory.

The combined Infoblox/Red Hat solution enables network professionals to overcome the burden of maintaining a static registry of devices and gain a centralized and highly efficient way to manage DNS, DHCP, and IPAM (DDI) automation of VMs and containerized workloads deployed across multiple platforms.

## Ansible-Infoblox Integration

The Infoblox NIOS Collection for Ansible provides 16 modules and 4 plugins included with Ansible 2.9. It enables networking teams to leverage Ansible NIOS modules and plugins to automate Infoblox Core Network Services for IPAM, DNS, and inventory tracking for workloads deployed across multiple platforms. It frees network administrators from frequent repetitive requests or tasks with high error rates, including IP address assignments, DNS record creation, and cleanup of everything once a resource is no longer needed.

The `nios_modules` collection provides modules and plugins for managing the networks, IP addresses, and DNS records in NIOS. This collection is hosted on Ansible Galaxy under `infoblox.nios_modules`.

### Modules

- `nios_a_record` – Configure Infoblox NIOS A records
- `nios_aaaa_record` – Configure Infoblox NIOS AAAA records
- `nios_cname_record` – Configure Infoblox NIOS CNAME records
- `nios_dns_view` – Configure Infoblox NIOS DNS views
- `nios_fixed_address` – Configure Infoblox NIOS DHCP Fixed Address
- `nios_host_record` – Configure Infoblox NIOS host records
- `nios_member` – Configure Infoblox NIOS members
- `nios_mx_record` – Configure Infoblox NIOS MX records
- `nios_naptr_record` – Configure Infoblox NIOS NAPTR records
- `nios_network` – Configure Infoblox NIOS network object
- `nios_network_view` – Configure Infoblox NIOS network views
- `nios_nsgroup` – Configure Infoblox DNS Nameserver Groups
- `nios_ptr_record` – Configure Infoblox NIOS PTR records
- `nios_srv_record` – Configure Infoblox NIOS SRV records
- `nios_txt_record` – Configure Infoblox NIOS txt records
- `nios_zone` – Configure Infoblox NIOS DNS zones

### Plugins

- `nios_inventory` – List all the hosts with records created in NIOS
- `nios_lookup` – Look up queries for NIOS database objects
- `nios_next_ip` – Returns the next available IP address for a network
- `nios_next_network` – Returns the next available network addresses for a given network CIDR

### Use Cases

#### Manage DNS Records, Networks and IP Addresses for VMs

Ansible enables the automation for creating and deleting VM's that are deployed across multiple platforms. Integration with Infoblox is powered by the NIOS module in Ansible that provides the framework for managing the networks, IP addresses, and DNS records in NIOS.

#### Automate Deployment of Virtual Infoblox Appliances

Organizations can use Ansible to automate the creation (and deletion) of virtual Infoblox appliances. You can leverage this module for autoscaling grid members based on DNS traffic.

## Deployment

Ansible is an agentless automation tool that by default manages machines over the SSH protocol. You only need to install it on one machine (which could easily be a laptop) and it can manage an entire fleet of remote machines from that central point.

## Requirements

Ansible is available for Linux based operating systems (include MacOS) and can be installed on physical or virtual hosts.

This section lists the (minimum) system requirements for installing and using Ansible. For more details refer to the official documentation present [here](#):

- For the 'Control' machine, any distribution of Linux with Python 2.7 or newer, or 3.5 or newer.
- For the 'managed nodes', you need Python 2.6 or newer, or 3.5 or newer.
- PIP, the package management system for Python. If not already present, this can be installed, as below, depending on the Python version you use.

```
For Python2:  
sudo apt install python-pip
```

```
For Python3:  
sudo apt install python3-pip
```

- The infoblox-client WAPI package for python.

```
For Python2:  
sudo pip install infoblox-client
```

```
For Python3:  
sudo pip3 install infoblox-client
```

- If using MacOS, run the following command to avoid the error "Too many files open".

```
sudo launchctl limit maxfiles unlimited
```

- Internet access and working DNS on the system where Ansible is being installed (the 'Control' machine).

## Initial Setup

### Ansible

Ansible is supported on multiple Linux distributions so the installation steps will vary depending on the flavor that you are installing it on.

When getting started, it is recommended to use the OS packages for EPEL and APT; although, Ansible is available through multiple sources, including Pypi and GitHub. For installation instructions for your OS (operating system), refer to [https://docs.ansible.com/ansible/latest/installation\\_guide/intro\\_installation.html](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html).

### Picking an Ansible Version

Which Ansible version to install is based on your particular needs. You can choose any of the following ways to install Ansible:

- Install the latest release with your OS package manager (for Red Hat Enterprise Linux (TM), CentOS, Fedora, Debian, or Ubuntu).
- Install with `pip` (the Python package manager).
- Install `ansible-base` from source to access the development (devel) version to develop or test the latest features.

Please note that to get the Infoblox Collections to work as described in the guide, the minimum required version is Ansible 2.9.

In this guide, we demonstrate the installation of the latest release of Ansible on Ubuntu using APT.

## Installation

To install Ansible on Ubuntu, run the following commands:

```
sudo apt update
sudo apt install software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Note: This process generally only takes a few minutes to complete.

## Verify Installation

To verify that Ansible has been successfully installed, run the following command:

```
ansible --version
```

```
tme@ubuntu20:~$ ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/tme/.ansible/plugins/modules',
  '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.8.5 (default, Jul 28 2020, 12:59:40) [GCC 9.3.0]
```

## Inventory

Ansible uses an ‘inventory’ to identify all servers that it manages. This can be done using a static ‘hosts’ file (found in `/etc/ansible/` by default) or a dynamically generated inventory list. To update the static inventory and add your Infoblox appliance, use the following command examples:

1. `sudo vi /etc/ansible/hosts`
2. `<shift-G>` (move to the bottom of the file)
3. `i` (to enter interactive mode)
4. Type the name or IP address for your Infoblox appliance.
5. `<esc>`
6. `:wq`

## Infoblox

### Supported Versions

When preparing your playbooks, it is important to set the WAPI version to the version used by your version of NIOS by specifying the **wapi\_version: x.x** parameter.

You can verify the WAPI version used by your Infoblox appliance by appending “[/wapidoc/#backward-compatibility](#)” to the end of the URL used to connect to your Infoblox Grid Manager GUI.

Example:

```
https://grid-master.demo.com/wapidoc/#backward-compatibility
```

**Infoblox**

**Table Of Contents**

- Infoblox WAPI documentation
  - Introduction
    - Notation
    - Transport and Authentication
    - Backward Compatibility
    - General Syntax and Options
    - Naming and Values
    - Object Reference
    - Function Calls
    - Extensible Attributes
    - Use Flags
    - Data Formats
    - XML Format
    - Error Handling
  - Methods
    - GET
    - POST
    - PUT
    - DELETE
  - Objects
  - Structs
  - Additional Information and Examples

**Next topic**

ad\_auth\_service : Active Directory Authentication Service object.

## Backward Compatibility

The Infoblox WAPI has a versioning scheme that is independent of the NIOS versioning scheme. The current WAPI version is 2.12.

A current WAPI version is backward compatible with WAPI releases that have the same major WAPI version or with designated earlier major versions. Though the protocol itself may not be strictly backward compatible, the server emulates the correct behavior, when necessary.

For example, a client that uses WAPI version *X* behaves the same way in version *Y* if *X* is supported by *Y* (that is *X* is lower than *Y* and *X* has the same major version as *Y* or *X* uses a major version that is supported by *Y*).

The WAPI protocol is versioned (see URL in [General Syntax and Options](#)) independently from NIOS. Refer to the release notes for information about the WAPI version.

Requirements and exceptions:

- Rely on errors returned by HTTP Error Status only, not by text messages or other components.
- New objects and fields may exist in a later WAPI version. Thus, additional fields may be returned and must be ignored.
- New syntaxes and values may be supported. Do not rely on receiving errors for illegal usage.
- In the URL, use the WAPI version that corresponds to the behavior you expect. Do not combine requests using different WAPI versions in the same session or connection.

The default for the WAPI version is set to 2.1, which corresponds to NIOS version 7.1. Some operations may require newer WAPI versions. For example, the minimum version required for the NIOS member module to work is 2.2

### Cloud Admin user (Optional)

The plugin will authenticate with NIOS using an account specified in its config file/playbook or environment parameters. For this to work, this account must first be created in NIOS.

This can be a regular admin account, or a cloud-api enabled account, with the appropriate permissions.

To create a cloud-api enabled admin account:

1. Login to your Infoblox Grid Manager GUI if not already logged in.
2. Navigate to **Administration** → **Administrators** → **Admins**.
3. Click on the **+** (Add) button.
4. Specify the username in the Login field, along with the desired password in the two corresponding text boxes.
5. Click **Select** and choose the **cloud-api-only** group.
6. Click **Save & Close**.

**Note:** For the **cloud-api-only** group to be available, you need to have the Cloud Network Automation license enabled on your NIOS appliance.

Add Administrator Wizard > Step 1 of 2

Authentication Type: Local

**Credentials**

\*Login: cloudadmin

\*Password: .....

\*Confirm Password: .....

Password must contain at least 4 characters.

Email Address:

\*Admin Group: cloud-api-only

Comment:

Permissions must also be defined which will allow the plugin to make changes. To set the permissions:

1. Navigate to **Administration** → **Administrators** → **Permissions**.
2. Under the **Groups** column, select **cloud-api-only**.
3. Click on the **+** (Add) button. If the menu expands, select **Global Permissions** (clicking on the icon itself will default to this menu option).
4. Set the permissions as required. For lab purposes and getting started, allow Read/Write access for the following:
  - a. **DNS Permissions** -> **All DNS Views**
  - b. **DHCP Permissions** -> **All Network Views**
  - c. **Grid Permissions** -> **All Members**

**Note:** Permissions are inherited. Unless overridden at a lower level, they apply to all objects underneath.

Navigation: Dashboards | Data Management | Cloud | Smart Folders | Grid | Administration

Sub-navigation: Administrators | Workflow | Logs | Network Views | Extensible Attributes | Cloud | Authentication Server Groups | Named ACLs

Admins | Groups | Roles | Permissions | Authentication Policy | SNMPv3 Users

**+ - Create New Permission**

Admins [Show All](#) | Groups [Show All](#)

cloudadmin | cloud-api-only

**cloud-api-only Permissions**

Group/Role	Permission Type	Resource	Resource Type	Permission
cloud-api-only	Grid Permissions	All Members	Member	RW
cloud-api-only	DHCP Permissions/IPAM Permissi...	All Network Views	Network view	RW
cloud-api-only	DNS Permissions	All DNS Views	DNS View	RW

## Ansible-Infoblox Integration

### Getting Started

As mentioned in the requirements section, please make sure you the Python Infoblox Client installed. The infoblox-client WAPI package for python can be installed as below, depending on the Python version used by Ansible.

```
For Python2:  
sudo pip install infoblox-client
```

```
For Python3:  
sudo pip3 install infoblox-client
```

To identify which version of Python Ansible is using, you can check the results of the `ansible --version` command.

```
tme@ubuntu20:~$ ansible --version  
ansible 2.9.6  
  config file = /etc/ansible/ansible.cfg  
  configured module search path = ['/home/tme/.ansible/plugins/modules',  
  '/usr/share/ansible/plugins/modules']  
  ansible python module location = /usr/lib/python3/dist-packages/ansible  
  executable location = /usr/bin/ansible  
  python version = 3.8.5 (default, Jul 28 2020, 12:59:40) [GCC 9.3.0]
```

In the above example, you would need to use the command for python3 mentioned above. To verify if you have the client installed, you can run the following command.

```
For Python2:  
sudo pip show infoblox-client
```

```
For Python3:  
sudo pip3 show infoblox-client
```

```
tme@ubuntu20:~$ sudo pip3 show infoblox-client  
Name: infoblox-client  
Version: 0.5.0  
Summary: Client for interacting with Infoblox NIOS over WAPI  
Home-page: https://github.com/infobloxopen/infoblox-client  
Author: John Belamaric  
Author-email: jbelamaric@infoblox.com  
License: Apache  
Location: /usr/local/lib/python3.8/dist-packages  
Requires: requests, urllib3, setuptools, oslo.serialization, oslo.log, six  
Required-by:
```

### Installing the Infoblox Collections

The `nios_modules` collection can be installed either from Ansible Galaxy or directly from git. It is recommended to install collections from Ansible Galaxy as those are more stable than the ones in the git branch.

In this deployment guide, we will install the collections from Ansible Galaxy.

#### Installation from Ansible Galaxy

To directly install the `nios_modules` collection from Ansible Galaxy, run the following command:

```
ansible-galaxy collection install infoblox.nios_modules
```

The collection folder would be installed at

```
~/ansible/collections/ansible_collections/infoblox/nios_modules
```

```
tme@ubuntu20:~$ ansible-galaxy collection install infoblox.nios_modules
Process install dependency map
Starting collection install process
Installing 'infoblox.nios_modules:1.0.2' to '/home/tme/ansible/collections/
ansible_collections/infoblox/nios_modules'
```

## Installation from Git

To git clone and install from this repo, follow these steps:

1. Clone the repo:

```
git clone https://github.com/infobloxopen/infoblox-ansible.git
```

2. Build the collection:

To build a collection, run the following command from inside the root directory of the collection:

```
cd infoblox-ansible/ansible_collections/infoblox/nios_modules
ansible-galaxy collection build
```

This creates a tarball of the built collection in the current directory.

3. Install the collection:

```
ansible-galaxy collection install <collection-name>.tar.gz -p
./collections
```

## Writing the Playbooks

Developing playbooks that use the Infoblox NIOS modules can enable complex operations when automating IPAM functions for device management. Infoblox ships a few sample playbooks that can be used as a reference.

You can find these sample playbooks inside the root directory of the collection under the `playbooks/` directory.

Depending on your method of installation, navigate to the collection root directory. If you installed from Ansible Galaxy, navigate to `~/ansible/collections/ansible_collections/infoblox/nios_modules` and you would find it as below.

```
tme@ubuntu20:~/ansible/collections/ansible_collections/infoblox/nios_modules$ pwd
/home/tme/ansible/collections/ansible_collections/infoblox/nios_modules
tme@ubuntu20:~/ansible/collections/ansible_collections/infoblox/nios_modules$ ls
CHANGELOG.rst  CONTRIBUTING.md  MANIFEST.json  playbooks  README.md
changelogs    FILES.json      meta          plugins    tests
```

```
cd ~/ansible/collections/ansible_collections/infoblox/nios_modules/playbooks
ls
```

```
tme@ubuntu20:~/ansible/collections/ansible_collections/infoblox/nios_modules/playbooks$ ls
create_a_record.yaml      create_txt_record.yaml    delete_network.yml
create_cname_record.yaml  create_zone.yml           delete_txt_record.yaml
create_dns_view.yml       delete_cname_record.yaml  delete_zone.yml
create_mx_record.yaml     delete_dnsview.yml        update_a_record.yml
create_network_view.yml   delete_mx_record.yaml
create_network.yml        delete_network_view.yml
```

You can copy it to your working directory, modify them and execute them.

A select number of example playbooks are included as part of this deployment guide for your reference below.

## Create a Network View

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: create network view
      infoblox.nios_modules.nios_network_view:
        name: demo
        extattrs:
          Site: Demo Site
        comment: Created with Ansible
        state: present
        provider: "{{ nios_provider }}"
```

## Create a Network

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: create network
      infoblox.nios_modules.nios_network:
        network: 10.0.0.0/24
        network_view: demo
        options:
          - name: domain-name
            value: infoblox-ansible.com
        extattrs:
          Site: Test Site
        comment: Created with Ansible
        state: present
        provider: "{{ nios_provider }}"
```

## Create a Host Record

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: create host record
      infoblox.nios_modules.nios_host_record:
        name: host.demo.com
        view: demoDNSView
        ipv4addrs:
          - ipv4addr: "{{ lookup('nios_next_ip', '10.0.0.0/24',
provider=nios_provider)[0] }}"
        ipv6addrs:
          - ipv6addr: fd00::2
        ttl: 3600
        extattrs:
          Site: Demo Site
        comment: Created with Ansible
        state: present
        provider: "{{ nios_provider }}"
```

## Create an A Record

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: Create NIOS A record
      infoblox.nios_modules.nios_a_record:
        name: test-server.demo.com
        view: demoDNSView
        ipv4: 192.168.11.251
        comment: Created with Ansible
        state: present
        provider: "{{ nios_provider }}"
```

## Add entry for a Grid Member

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: create member
      infoblox.nios_modules.nios_member:
        host_name: member01.ansible-demo.com
        vip_setting:
          - address: 192.168.1.71
            subnet_mask: 255.255.255.0
            gateway: 192.168.1.1
        config_addr_type: IPV4
        platform: VNIOS
        comment: Created with Ansible
        state: present
        provider: "{{ nios_provider }}"
```

## Delete A Record

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: delete A record
      infoblox.nios_modules.nios_a_record:
        name: test-server.demo.com
        view: demoDNSView
        state: absent
        provider: "{{ nios_provider }}"
```

## Delete Network

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: delete network
      infoblox.nios_modules.nios_network:
        network: 10.0.0.0/24
        network_view: demo
        state: absent
        provider: "{{ nios_provider }}"
```

## Delete Member

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: delete member
      infoblox.nios_modules.nios_member:
        host_name: member01.ansible-demo.com
        state: absent
        provider: "{{ nios_provider }}"
```

## Sample lookup plugin

```
---
- hosts: localhost
  vars:
    nios_provider:
      host: grid-master.demo.com
      username: cloudadmin
      password: pwd
      wapi_version: '2.12'

  connection: local
  tasks:
    - name: get member list
      set_fact:
        members: "{{ lookup('nios', 'member', provider=nios_provider) }}"
    - name: display all members
      debug:
        msg: "{{ members }}"
```

## Preparing your Playbooks

Once your environment has been setup, the first step before running your playbooks is to make sure that all variables are updated for your environment. In the examples provided in this guide, the variables which may require modification have been highlighted in red.

## Running your Playbooks

Once your playbooks have been updated with any changes required to make them work in your environment, you are ready to begin working with them. To run the playbook, use the `ansible-playbook` command.

```
ansible-playbook <name of playbook>.yaml
```

```
tme@ubuntu20:~/ansible-playbooks$ ansible-playbook create_a_record.yaml
PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Create NIOS A record] *****
changed: [localhost]

PLAY RECAP *****
localhost : ok=2   changed=1   unreachable=0   failed=0   skipped=
0   rescued=0   ignored=0
```

To increase verbosity of the output, you can add `-v` to the `ansible-playbook` command.

## URI Module

In case, you need to automate Infoblox objects that are not a part of the 16 modules supported in the collections, you may use the `uri` module that is part of `ansible-base` and included in all Ansible installations. It allows users to interact with HTTP and HTTPS web services. You can find more information on how to use it [here](#).

Below is a sample playbook that uses the `uri` module can be used against any of the REST APIs supported by the Infoblox Grid. It uses the `grid:dns` endpoint to start the DNS service on the grid.

### Example URI module usage to start DNS services on the Grid

```
---
- hosts: localhost
  vars:
    host: grid-master.demo.com
    username: cloudadmin
    password: pwd
    wapi_version: 'v2.12'

  connection: local
  tasks:
    - name: Get object reference of the DNS service
      uri:
        url: "https://{{ host }}/wapi/{{ wapi_version
        }}/member:dns?host_name=infoblox.localdomain"
        user: "{{ username }}"
        password: "{{ password }}"
        validate_certs: no
        return_content: yes
        method: GET
        force_basic_auth: yes
        status_code: 200
        register: content

    - name: Print object reference
      debug:
        var: content.json[0]["_ref"]

    - name: Use object reference to start the DNS service
      uri:
        url: "https://{{ host }}/wapi/{{ wapi_version }}/{{
        content.json[0][\"_ref\"] }}"
        user: "{{ username }}"
        password: "{{ password }}"
        validate_certs: no
        return_content: yes
        method: PUT
        force_basic_auth: yes
        status_code: 201, 302, 200
        headers:
          Content-Type: "application/json"
        body:
          enable_dns: true
          body_format: json
```

## Conclusion

With the [Infoblox NIOS Collection for Ansible](#), organizations can confidently handle the most challenging DDI requirements in every type of network, data center, and hybrid cloud environment.

The Ansible modules allow you to configure Infoblox, and plugins allow you to grab information from Infoblox to use in subsequent tasks. With these modules, you can now automate your Infoblox infrastructure.

## Additional Information

<https://www.ansible.com/>

<http://docs.ansible.com/ansible/latest/>

<http://docs.ansible.com/ansible/latest/YAMLSyntax.html>

<https://community.infoblox.com/>

[Community Blog: Infoblox vNIOS Autoscaling on Openstack using Ansible](#)

[Community Blog: What is new with Ansible 2.8](#)

[Community Blog: Infoblox is Pleased to Announce the Brand New Infoblox NIOS Collection for Ansible](#)

[Reference Guide: Infoblox REST API](#)

## Appendix

### Troubleshooting

#### NIOS Module Command Help

If the infoblox-client package for Python has not been installed, you will see an error confirming that it is required. Example:

```
python infoblox.py
infoblox-client is required but does not appear to be installed.  It can be
installed using the command `pip install infoblox-client`
```

Sometimes, even if you have the infoblox-client installed, you may encounter this error. In that case, ensure that the infoblox-client and Ansible are using the same python version.

```
tme@ubuntu20:~/ansible-playbooks$ ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/tme/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.8.5 (default, Jul 28 2020, 12:59:40) [GCC 9.3.0]
```

```
tme@ubuntu20:~/ansible-playbooks$ pip3 show infoblox-client
Name: infoblox-client
Version: 0.5.0
Summary: Client for interacting with Infoblox NIOS over WAPI
Home-page: https://github.com/infobloxopen/infoblox-client
Author: John Belamaric
Author-email: jbelamaric@infoblox.com
License: Apache
Location: /home/tme/.local/lib/python3.8/site-packages
Requires: oslo.serialization, urllib3, six, oslo.log, requests, setuptools
Required-by:
```



Infoblox is the leader in modern, cloud-first networking and security services. Through extensive integrations, its solutions empower organizations to realize the full advantages of cloud networking today, while maximizing their existing infrastructure investments. Infoblox has over 12,000 customers, including 70 percent of the Fortune 500.

Corporate Headquarters | 2390 Mission College Boulevard, Ste. 501 | Santa Clara, CA | 95054  
+1.408.986.4000 | [info@infoblox.com](mailto:info@infoblox.com) | [www.infoblox.com](http://www.infoblox.com)



© 2021 Infoblox, Inc. All rights reserved. Infoblox logo, and other marks appearing herein are property of Infoblox, Inc. All other marks are the property of their respective owner(s).