

Deployment Guide

Infoblox Terraform Provider v2



Table of Contents

Introduction	2
Infoblox NIOS and Infoblox Terraform Provider	2
Prerequisites	2
Key Terms	3
Infoblox Grid Configuration	3
DNS Zone	3
Terraform Installation	7
Windows	7
macOS	7
Linux	7
Terraform Commands	8
Installing and Authenticating With the Infoblox Terraform Provider	9
Common Fields	10
Comments	10
Extensible Attributes	10
Extensible Attributes for Cloud Objects	10
TTL	11
Examples	11
Create DNS Records	11
Apply Terraform Configuration File	12
Provision Network and VM in Azure	15
Apply Terraform Configuration Files	18
Additional Resources	22

Introduction

Terraform is an Open Source Infrastructure as Code (IaC) tool that is developed by HashiCorp. It enables predictable and consistent provisioning of infrastructure across many public and private cloud providers.

Infoblox NIOS and Infoblox Terraform Provider

Infoblox NIOS provides core network services that include integrated, secure, and easy-to-manage DDI services - DNS (Domain Name System), DHCP (Dynamic Host Configuration Protocol) and IPAM (IP address management).

The Infoblox Terraform provider interfaces with Infoblox NIOS through REST API to provide IP Address Management and DNS Services. Instead of manually provisioning IP addresses and DNS records for network devices and interfaces in your infrastructure, you can use the plugin to automate these steps with Infoblox NIOS.

The Infoblox Terraform provider includes the following Resource types:

- Network View
- IPv4 and IPv6 Network Containers
- IPv4 and IPv6 Networks
- IPv4 and IPv6 Address Allocation - Allocates IP and creates Host record
- DNS Records: A, AAAA, CNAME, PTR
- IPv4 and IPv6 Association - Updates Grid with VM data

The resources listed support Create, Update, and Delete operations.

The following Data Sources are included :

- IPv4 Networks
- DNS Records: A, CNAME

This deployment guide covers the use of the Infoblox Terraform provider, v2.0.1. Examples in this guide show integration with the Azure public cloud. Additional examples for all resource types and data sources as well as example integrations with AWS and VMware vSphere cloud platforms can be found in the Infobloxopen GitHub repository, <https://github.com/infobloxopen/terraform-provider-infoblox/tree/master/examples/v0.14>. Full documentation on all of the resources and data sources can be found at <https://docs.infoblox.com/display/ipamdriverterraform> or in the Terraform registry, <https://registry.terraform.io/providers/infobloxopen/infoblox/latest/docs>.

Prerequisites

The following are prerequisites for using the Infoblox Terraform provider:

- Terraform version 0.14 or newer.
- Infoblox NIOS or vNIOS appliance, version 8.5 or newer with required licenses: Grid, NIOS, DNS.

Key Terms

- **Provider:** Plugins for Terraform that interact with other tools such as cloud providers, SaaS applications, and other APIs.
- **Terraform Configuration:** A document written in the Terraform language, which tells Terraform how to manage a collection of infrastructure.
- **Resource Block:** Defines one or more infrastructure objects in a Terraform configuration. Each provider contains a collection of resources for managing its associated infrastructure.
- **Data Source:** Used in a Terraform configuration to pull in data from another source to be used by Terraform. Providers can include data sources to access information from their associated infrastructure.
- **Terraform Registry:** A repository of providers for Terraform. The registry is directly integrated with Terraform, allowing you to specify providers in your configuration files.

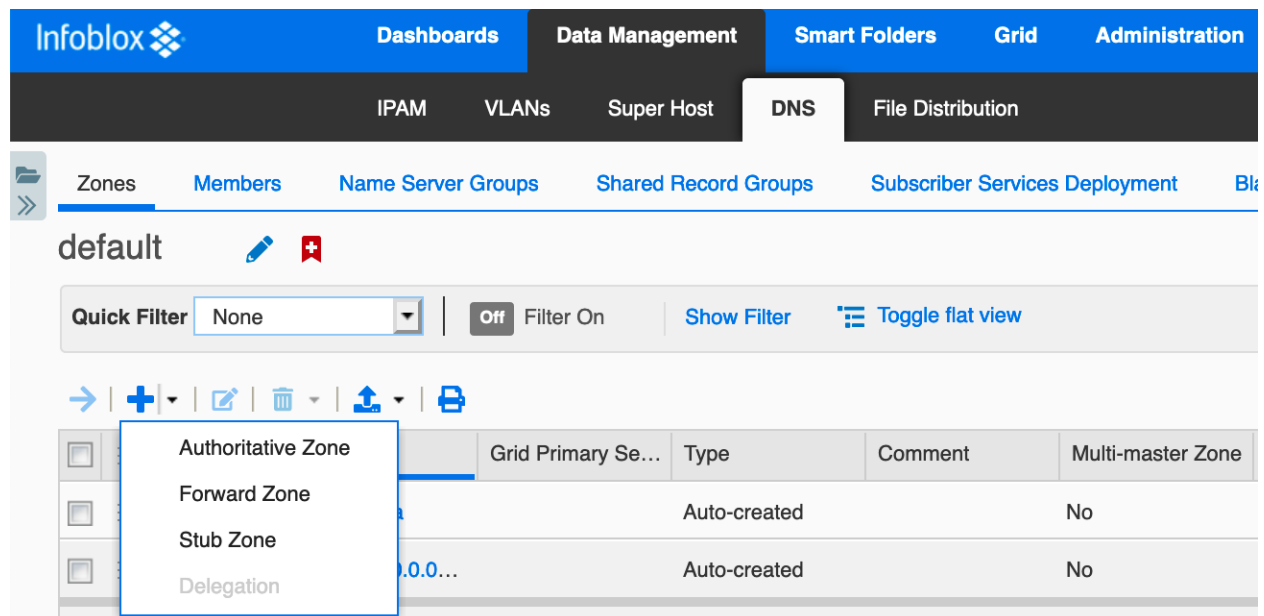
Infoblox Grid Configuration

To use the Infoblox Terraform provider, you will need an Infoblox Grid running NIOS 8.5 or newer. This guide assumes you have a Grid with at least one member already deployed. For instructions on deploying Infoblox NIOS or vNIOS, refer to the deployment guide for the appropriate platform. Deployment guides for many available platforms can be found at <https://www.infoblox.com/resources/>. NIOS and virtual appliance documentation can be found at <https://docs.infoblox.com>.

DNS Zone

To use all of the Infoblox Terraform provider DNS resource types, you will need at least one Authoritative Forward-Mapping Zone in your Infoblox Grid.

1. In the Grid Manager, navigate to the **Data Management** → **DNS** → **Zones** tab.
2. Open the **+** (Add) dropdown.
3. Select **Authoritative Zone**.



4. On Step 1 of the wizard, select **Add an authoritative forward-mapping zone**.
5. Click **Next**.

Add Authoritative Zone Wizard > Step 1 of 6

☒ Add an authoritative forward-mapping zone

☐ Add an authoritative IPv4 reverse-mapping zone

☐ Add an authoritative IPv6 reverse-mapping zone

?

<<

Cancel

Previous

Next

Schedule for Later

Save & Close

6. On Step 2, enter the name of your zone.
7. Click **Next**.

Add Authoritative Zone Wizard > Step 2 of 6

*Name

ibxdemo.com

Comment

Disable

☐

Disabling large amounts of data may take a longer time to execute.

?

<<

Cancel

Previous

Next

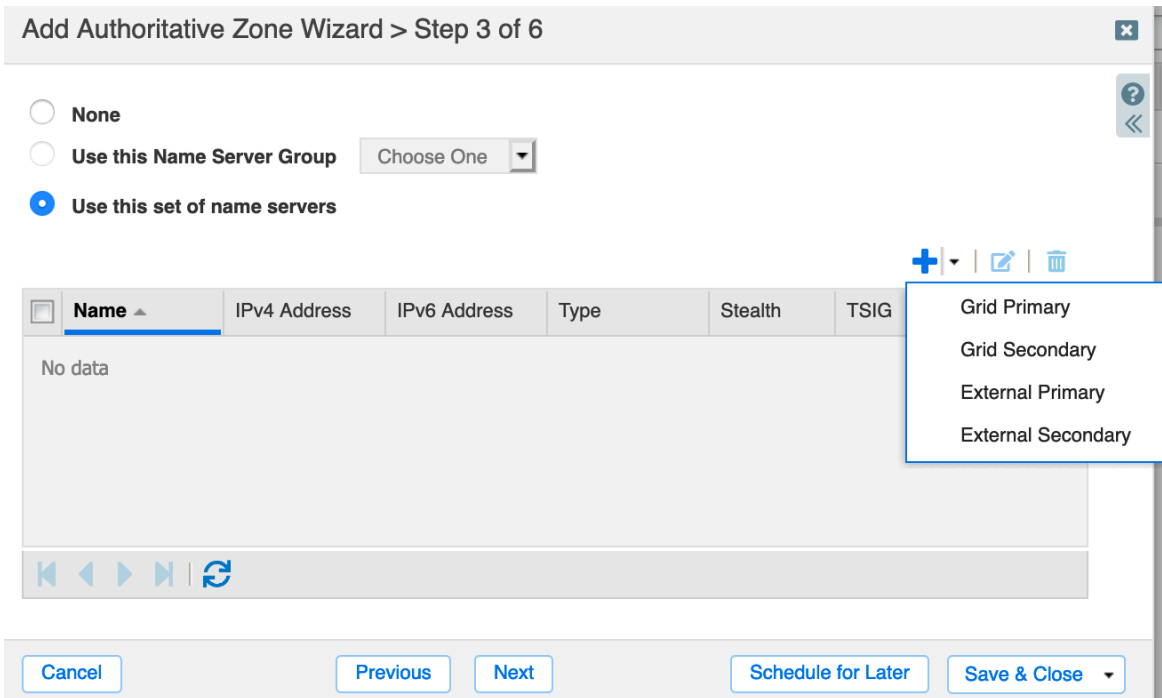
Schedule for Later

Save & Close

8. On Step 3, select **Use this set of name servers**.

9. Open the  (Add) dropdown.

10. Select **Grid Primary**.






Add Authoritative Zone Wizard > Step 3 of 6






☐ None

☐ Use this Name Server Group Choose One

☒ Use this set of name servers

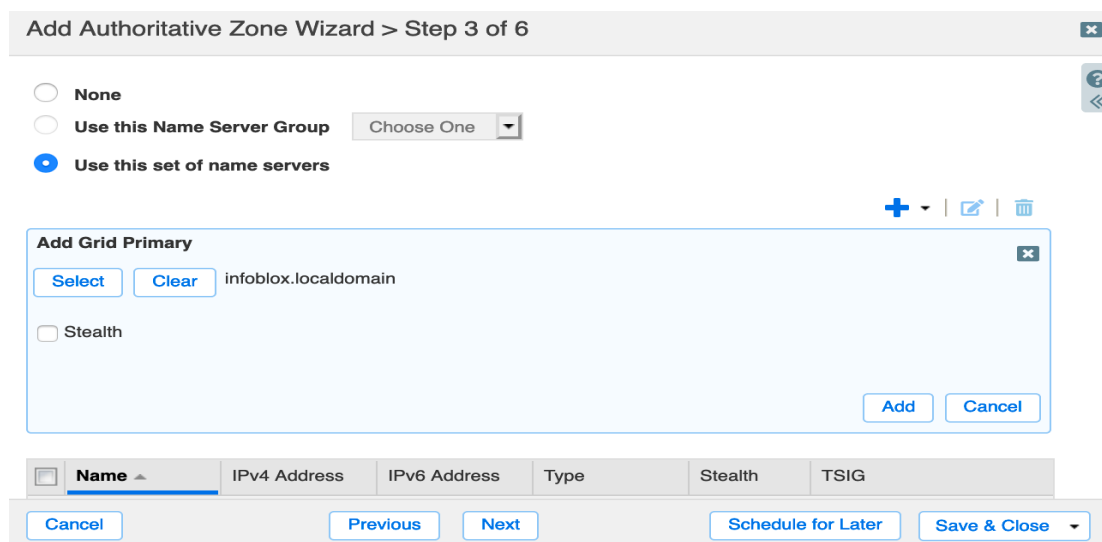
<input type="checkbox"/> Name	IPv4 Address	IPv6 Address	Type	Stealth	TSIG
No data					

Cancel Previous Next Schedule for Later Save & Close

11. Click **Select** in the Add Grid Primary panel.

12. For a Grid with only one member, it will be automatically selected. If you have multiple Grid members, select the one you want to use as a name server.






Add Authoritative Zone Wizard > Step 3 of 6

☐ None

☐ Use this Name Server Group Choose One

☒ Use this set of name servers

Add Grid Primary

Select Clear infoblox.localdomain

☐ Stealth

Add Cancel

<input type="checkbox"/> Name	IPv4 Address	IPv6 Address	Type	Stealth	TSIG
-------------------------------	--------------	--------------	------	---------	------

Cancel Previous Next Schedule for Later Save & Close

13. Click **Add**.

14. Click **Save & Close**.

Add Authoritative Zone Wizard > Step 3 of 6

☐ None

☐ Use this Name Server Group

Choose One

☒ Use this set of name servers

Name

IPv4 Address

IPv6 Address

Type

Stealth

TSIG

infoblox.local...

172.17.1.222

Grid Primary

No

No

Cancel

Previous

Next

Schedule for Later

Save & Close

15. Click **Restart** in the warning bar when prompted.

The configuration changes require a service restart to take effect. Click Restart to restart relevant services now or click Ignore to restart the services later.

Restart

View Changes

Ignore

Infoblox

Dashboards

Data Management

Smart Folders

Grid

Administration

IPAM

VLANs

Super Host

DNS

File Distribution

16. Click **Restart** in the Restart Grid Services window.

Restart Grid Services

Restart Grid Services

☒ If needed

☐ Force service restart

A forced restart may be delayed if there are pending restarts for the same service.

Restart Method

☒ Restart all Restart Groups

☐ Simultaneously for all members

☐ Sequentially for all members

Affected Members and Services

View Pending Changes

Member

To start polling, click the Poll Members icon above this table ...

Cancel

Restart

Infoblox Deployment Guide - Infoblox Terraform Provider v2 (August 2021)

6

Terraform Installation

Terraform is available for many operating systems. Instructions and download information for the supported platforms can be found at <https://learn.hashicorp.com/tutorials/terraform/install-cli>. The following are basic installation methods for Windows, macOS, and Debian/Ubuntu Linux operating systems.

Windows

To install Terraform on Windows, use the Chocolatey package manager command:

choco install terraform

For information on installing and using Chocolatey, refer to <https://chocolatey.org/>.

macOS

To install Terraform on macOS, use the Homebrew package manager command:

brew install hashicorp/tap/terraform

For information on installing and using Homebrew, refer to <https://brew.sh/>.

Linux

Terraform can be installed on many Linux distributions using their native package managers. To install Terraform on Ubuntu or Debian Linux, use the following steps:

1. Open a terminal window.
2. Add the HashiCorp GPG key with the command:

curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

```
infoblox@az-cli-vm:~$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
[sudo] password for infoblox:
OK
infoblox@az-cli-vm:~$
```

3. Add the HashiCorp Linux repository with the following command:

sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com \$(lsb_release -cs) main"

```
infoblox@az-cli-vm:~$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
Get:1 https://apt.releases.hashicorp.com bionic InRelease [4,421 B]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 https://packages.microsoft.com/repos/azure-cli bionic InRelease [3,965 B]
Get:5 https://apt.releases.hashicorp.com bionic/main amd64 Packages [11.7 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
```


4. Run the following command to update package information and install Terraform:

sudo apt-get update && sudo apt-get install terraform

```
infoblox@az-cli-vm:~$ sudo apt-get update && sudo apt-get install terraform
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 https://packages.microsoft.com/repos/azure-cli bionic InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:6 https://apt.releases.hashicorp.com bionic InRelease
Reading package lists... Done
```

Terraform Commands

The following are basic commands you will use while working with the Infoblox Terraform Provider. For additional information and reference for all Terraform commands, refer to the documentation at <https://www.terraform.io/docs/commands/index.html>. All commands will begin with **terraform**, followed by a subcommand. Screenshots of some commands are shown here; others will be shown in the Examples section.

- **terraform -help**: This command displays the common commands available for Terraform.

```
infoblox@az-cli-vm:~$ terraform -help
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
  apply          Builds or changes infrastructure
  console        Interactive console for Terraform interpolations
  destroy        Destroy Terraform-managed infrastructure
  env            Workspace management
  fmt            Rewrites config files to canonical format
  get            Download and install modules for the configuration
  graph          Create a visual graph of Terraform resources
  import         Import existing infrastructure into Terraform
  init           Initialize a Terraform working directory
```

- **terraform -version**: This command displays the version of Terraform installed on your computer.

```
infoblox@az-cli-vm:~/tf-guide$ terraform -version
Terraform v1.0.3
```

- **terraform init**: This command is used to initialize a working directory containing Terraform configuration files and install Terraform providers such as the Infoblox plugin. This command must be run in a directory prior to apply or plan. A screenshot is provided in the Examples section.
- **terraform validate**: This command is used to verify that configuration files in the working directory are valid. Configuration files are validated for syntax and internal consistency. This is applied only to the configuration and does not access any remote services such as provider APIs.

```
infoblox@az-cli-vm:~/tf-guide$ terraform validate
Success! The configuration is valid.
```

- **terraform plan -out=<path/file>**: The plan command creates an execution plan by determining the actions needed to achieve a desired state based on the configuration files in your directory. This command does not make any changes and is a good way to verify that your configuration files will give the desired result prior to executing. The **-out** parameter is optional and records the plan for later use when applying the configuration. A screenshot is provided in the Examples section.
- **terraform apply <plan_file>**: The apply command executes changes required to meet the desired state based on configuration files in your directory. Optionally, add the name of the file output by the plan command to execute the predetermined actions from the plan command. A screenshot is provided in the Examples section.
- **terraform destroy**: This command is used to de-provision Terraform managed infrastructure defined by the configuration files in your directory. The destroy command will ask for confirmation unless the **-auto-approve** argument is used. A screenshot is provided in the Examples section.

Installing and Authenticating With the Infoblox Terraform Provider

In Terraform version 0.13 and later, a required providers block, which must be nested inside a terraform block, is required to install the Infoblox provider (and most other providers). The following is an example of a required providers block to install the Infoblox provider v2.0.1:

```
terraform {
  required_providers {
    infoblox = {
      source = "infobloxopen/infoblox"
      version = "~2.0.1"
    }
  }
}
```

The terraform block containing the required providers block can be followed by a provider block to pass additional configuration for the provider such as credentials. The following is an example of a provider block for Infoblox:

```
provider "infoblox" {
  username = "admin"
  password = "password"
  server = "10.0.0.1"
}
```

The server, username, and password arguments specify the Infoblox credentials which will be used when provisioning and deprovisioning resources in your Infoblox Grid.

If you do not want to include credentials in your Terraform configuration file for Infoblox, they can be stored as environmental variables on your system instead. For example, you can make these credentials available on Linux and macOS systems using the following commands in your terminal:

```
export INFOBLOX_USERNAME="admin"
export INFOBLOX_PASSWORD="password"
export INFOBLOX_SERVER="10.0.0.1"
```

When you export the credentials as shown above, you can omit them from your provider block.

Common Fields

All of the Infoblox Terraform resources support fields for comments and extensible attributes. Additionally, DNS record resources support the TTL field. These fields are all optional.

Comments

The comment field is used to add a comment or description to the object created. Comments can be viewed in the Grid Manager GUI or returned as part of an API call. The comment field takes a string value. The following is an example of a Network View resource with a comment:

```
resource "infoblox_network_view" "new_view" {
  name = "New-View"
  comment = "This object is created by Terraform"
}
```

Extensible Attributes

Extensible attributes (EA) are identifiers applied to objects in the NIOS Grid, similar to tags used in Azure or AWS, that allow you to further define and track objects. The optional **ext_attrs** argument for all resources in the Infoblox provider allows you to add extensible attributes to the objects being created. You can add any extensible attribute to your resource that is defined in the Grid and allowed for that object type. The **ext_attr** argument contains a JSON encoded block of key/pair values. This example shows a Network resource with **ext_attrs** to add values for Location and Site EAs:

```
resource "infoblox_ipv4_network" "new_network" {
  cidr = "10.101.0.0/24"
  ext_attrs = jsonencode({
    "Location" = "California"
    "Site" = "HQ"
  })
}
```

For additional information on managing extensible attributes, refer to NIOS documentation, <https://docs.infoblox.com/display/ILP/NIOS>.

Extensible Attributes for Cloud Objects

The Infoblox provider for Terraform is commonly used with Infoblox Cloud Network Automation features to automate DNS and IPAM for private and public cloud platforms. To use Terraform with the Cloud API for CNA, or with Cloud Platform (CP) appliances, specific extensible attributes must be included in the **ext_attrs** argument. You must include values for: **CMP Type**, **Tenant ID**, and **Cloud API Owned**. The following is an example of a **ext_attrs** argument for a cloud object:

```
ext_attrs = jsonencode({
  "CMP Type" = "Terraform"
  "Tenant ID" = "tenant-1"
  "Cloud API Owned" = "True"
})
```

You can include other extensible attributes for cloud objects in addition to the three which are required.

TTL

The time to live (TTL) field is supported for all DNS record resources and data sources. TTL specifies the time that a name server is allowed to cache a record before updating the data. The TTL field takes an integer value and is expressed in seconds. A value of 0 indicates the record should not be cached. If no TTL field is specified, the record created will inherit this value from the parent zone. The following is an example of an A record resource which sets a TTL of 3600 seconds (1 hour):

```
resource "infoblox_a_record" "new_record" {
  fqdn = "newrecord.ibxdemo.com"
  ip_addr = "192.168.233.4"
  ttl = 3600
}
```

Examples

This section provides examples of Terraform configuration files for use with the Infoblox Terraform provider. Required file syntax and formatting may change at any time and without notice. These examples are provided as-is and without warranty. For additional information on Terraform configuration language including changes in new versions, refer to the Terraform documentation: <https://www.terraform.io/docs/configuration/index.html>. For additional information on usage of Infoblox plugin resources including those not shown in these examples, refer to the Infoblox Terraform provider on GitHub: <https://github.com/infobloxopen/terraform-provider-infoblox>.

Create DNS Records

This example demonstrates a configuration file used to create DNS records for a DNS server. The configuration will create an A record and CNAME record. In order for this configuration to be successfully applied, you will need to have an existing authoritative DNS zone. Copy the below example into a text file and save as **dns_records.tf**. Replace the server, username, and password values with credentials from your Grid Master. Values used for other arguments such as CIDRs and names can be changed as desired for your environment. Lines that begin with # are comments explaining the resources and are not read when applying the configuration. The terraform and nested required provider block is required for Terraform version 0.13 and later.

Terraform block containing the required providers block to install the Infoblox provider

```
terraform {
  required_providers {
    infoblox = {
      source = "infobloxopen/infoblox"
      version = "2.0.1"
    }
  }
}
```

Provider block for Infoblox credentials

```
provider "infoblox" {
  server = "1.2.3.4"
  username = "admin"
  password = "infoblox"
}
```

Create A record

```
resource "infoblox_a_record" "dynamic_record" {
```

```

    network_view = "default"
    dns_view = "default"
    fqdn = "server1.ibxdemo.com"
    # Dynamically allocate IP from an existing network. To statically assign an IP, use the ip_addr
argument instead of cidr.
    cidr = "192.168.208.0/24"
    ttl = 3600
}

resource "infoblox_cname_record" "cname_1" {
    dns_view = "default"
    canonical = infoblox_a_record.dynamic_record.fqdn
    alias = "server1-alias.ibxdemo.com"
    ttl = 3600
}

```

Apply Terraform Configuration File

To deploy resources using the example configuration file above, save the file in a directory on your computer. Open a terminal and navigate to that directory. Run **terraform init** to initialize and install the Infoblox provider.

```

infoblox@az-cli-vm:~/tf-guide/dns$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding infobloxopen/infoblox versions matching "2.0.1"...
- Installing infobloxopen/infoblox v2.0.1...
- Installed infobloxopen/infoblox v2.0.1 (signed by a HashiCorp partner, key ID B355854AA5DBB18E)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

```

Run the command **terraform plan -out=plan1** to verify your environment and create an execution plan.

```
infoblox@az-cli-vm:~/tf-guide/dns$ terraform plan -out=plan1

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# infoblox_a_record.dynamic_record will be created
+ resource "infoblox_a_record" "dynamic_record" {
  + cidr      = "192.168.208.0/24"
  + dns_view  = "default"
  + fqdn      = "server1.ibxdemo.com"
  + id        = (known after apply)
  + network_view = "default"
  + ttl       = 3600
}

# infoblox_cname_record.cname_1 will be created
+ resource "infoblox_cname_record" "cname_1" {
  + alias      = "server1-alias.ibxdemo.com"
  + canonical  = "server1.ibxdemo.com"
  + dns_view   = "default"
  + id         = (known after apply)
  + ttl        = 3600
}

Plan: 2 to add, 0 to change, 0 to destroy.
```

If there are any errors, make corrections in your configuration files. As long as there are no errors shown, you are ready to apply the configuration. Run the **terraform apply plan1** command to execute your plan.

```
infoblox@az-cli-vm:~/tf-guide/dns$ terraform apply plan1
infoblox_a_record.dynamic_record: Creating...
infoblox_a_record.dynamic_record: Creation complete after 0s [id=record:a/ZG5zLmJpbmRfYSQ
uX2RlZmF1bHQuY29tLmlieGRlbW8sc2VydMvyMSwxOTIuMTY4LjIwOC4y:server1.ibxdemo.com/default]
infoblox_cname_record.cname_1: Creating...
infoblox_cname_record.cname_1: Creation complete after 1s [id=record:cname/ZG5zLmJpbmRfY2
5hbWUkLl9kZWZhdWx0LmNvbS5pYnhkZW1vLnNlcnZlcjEtYWxpYXM:server1-alias.ibxdemo.com/default]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

To view the new DNS records in your Infoblox Grid, login to Grid Manager and navigate to the **Data Management** → **DNS** tab. Click on your DNS zone to view the records.

Name	Type	Data	Record Source
	SOA Record	Serial 264 MNAME infoblox.localdomain RNAME please_set_email.absolutely.nowhere Refresh 10800 Retry 3600 Expire 2419200 Negative Caching TTL 900	System
	NS Record	infoblox.localdomain	System
server1	A Record	192.168.208.2	Static
server1-alias	CNAME Record	server1.ibxdemo.com	Static

When you are ready to deprovision the example resources, back in your terminal run the **terraform destroy** command.

```
infoblox@az-cli-vm:~/tf-guide/dns$ terraform destroy -auto-approve
infoblox_a_record.dynamic_record: Refreshing state... [id=record:a/ZG5zLmJpbmRfYSQuX2RLZmF1bHQuY29tLmlieGRlbW8sc2VydMvyMSwxOTIuMTY4LjIwOC4y:server1.ibxdemo.com/default]
infoblox_cname_record.cname_1: Refreshing state... [id=record:cname/ZG5zLmJpbmRfY25hbWUkLl9kZWZhdWx0LmNvbS5pYnhkZW1vLnNlcjEtYWxpYXM:server1-alias.ibxdemo.com/default]

Terraform used the selected providers to generate the following execution plan. Resource
actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# infoblox_a_record.dynamic_record will be destroyed
- resource "infoblox_a_record" "dynamic_record" {
  - cidr      = "192.168.208.0/24" -> null
  - dns_view  = "default" -> null
  - fqdn      = "server1.ibxdemo.com" -> null
  - id        = "record:a/ZG5zLmJpbmRfYSQuX2RLZmF1bHQuY29tLmlieGRlbW8sc2VydMvyMSwxOTIuMTY4LjIwOC4y:server1.ibxdemo.com/default" -> null
  - network_view = "default" -> null
  - ttl       = 3600 -> null
}
```

The records will be removed from your Infoblox Grid.

Provision Network and VM in Azure

This example demonstrates configuration files used to provision a new virtual network and virtual machine in Azure. The example uses the Infoblox provider to add the network container and network into the Infoblox Grid, find the next available IP address to use for the VM, create a DNS Host record for the VM, and update the Grid with metadata for the VM. We will use two files, **infoblox.tf** and **vm.tf**.

The first file, **infoblox.tf** will contain the resource blocks to create and identify objects in the Infoblox Grid. Copy the below example into a text file and save as **infoblox.tf**. Replace the server, username, and password values with credentials for your Grid Master. Values used for other arguments such as CIDRs and names can be changed as desired for your environment. Lines that begin with **#** are comments explaining the resources and are not read when applying the configuration. The terraform and nested required provider block is required for Terraform version 0.13 and later.

Terraform block containing the required providers block to install the Infoblox provider

```
terraform {
  required_providers {
    infoblox = {
      source = "infobloxopen/infoblox"
      version = "2.0.1"
    }
    azurerm = {
      source = "hashicorp/azurerm"
      version = "2.68.0"
    }
  }
}
```

Provider block for Infoblox credentials

```
provider "infoblox" {
  username = "admin"
  password = "infoblox"
  server = "1.2.3.4"
}
```

Create Network Container in Grid

```
resource "infoblox_ipv4_network_container" "az_vnet" {
  network_view = "default"
  cidr = "192.168.233.0/24"
  comment = "New Azure VNet"
  ext_attrs = jsonencode({
    "Tenant ID" = "Azure-tenant"
    "CMP Type" = "Azure"
    "Cloud API Owned" = "True"
  })
}
```

Create network object in Grid

```
resource "infoblox_ipv4_network" "az_network" {
  network_view = "default"
  allocate_prefix_len = 25
  parent_cidr = infoblox_ipv4_network_container.az_vnet.cidr
}
```



```

reserve_ip = 2
ext_attrs = jsonencode({
  "Network Name" = "tfsub"
  "Tenant ID" = "Azure-tenant"
  "CMP Type" = "Azure"
  "Cloud API Owned" = "True"
})
}

# Allocate IP address from new network for a new VM
resource "infoblox_ipv4_allocation" "az_allocation" {
  network_view = "default"
  dns_view = "default"
  cidr = infoblox_ipv4_network.az_network.cidr
  fqdn = "az-vm1.ibxdemo.com"
  enable_dns = "true"
  enable_dhcp = "false"
  ext_attrs = jsonencode({
    "VM Name" = "az-vm1"
    "Tenant ID" = "Azure-tenant"
    "CMP Type" = "Azure"
    "Cloud API Owned" = "True"
  })
}

# Update Grid with VM data
resource "infoblox_ipv4_association" "az_associate" {
  network_view = infoblox_ipv4_allocation.az_allocation.network_view
  dns_view = infoblox_ipv4_allocation.az_allocation.dns_view
  fqdn = infoblox_ipv4_allocation.az_allocation.fqdn
  enable_dns = infoblox_ipv4_allocation.az_allocation.enable_dns
  enable_dhcp = infoblox_ipv4_allocation.az_allocation.enable_dhcp
  cidr = infoblox_ipv4_network.az_network.cidr
  mac_addr = azurerm_network_interface.ni.mac_address
  ip_addr = infoblox_ipv4_allocation.az_allocation.ip_addr
  ext_attrs = jsonencode({
    "Network Name" = azurerm_subnet.subnet.name
    "VM Name" = azurerm_virtual_machine.vm.name
    "VM ID" = azurerm_virtual_machine.vm.id
    "Tenant ID" = "Azure-tenant"
    "CMP Type" = "Azure"
    "Cloud API Owned" = "True"
  })
}

```

The second file, `vm.tf` will contain the resource blocks to create resources in Azure, using values provided by the Infoblox resources. Copy the below example into a text file and save as **vm.tf**. Replace the subscription ID, client ID, client secret, and tenant ID with credentials authorized to create resources in your Azure environment. To use different methods of authenticating with Azure, refer to the Azure provider documentation: <https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>. Values used for other arguments such as

regions and names can be changed as desired for your environment. Lines that begin with **#** are comments explaining the resources and are not read when applying the configuration.

Provider block for Azure credentials

```
provider "azurerm" {  
  features {}  
  subscription_id = "azure-subscription-id"  
  tenant_id = "azure-tenant-id"  
  client_id = "azure-app-registration-client-id"  
  client_secret = "azure-app-registration-client-secret"  
}
```

Create a resource group in Azure for the new resources

```
resource "azurerm_resource_group" "terraform" {  
  name = "tf-ibx-grp"  
  location = "centralus"  
}
```

Create a VNet

```
resource "azurerm_virtual_network" "vnet" {  
  name = "tfvnet"  
  address_space = [infoblox_ipv4_network_container.az_vnet.cidr]  
  location = azurerm_resource_group.terraform.location  
  resource_group_name = azurerm_resource_group.terraform.name  
}
```

#Create subnet - references infoblox_network for CIDR

```
resource "azurerm_subnet" "subnet" {  
  name = "tfsub"  
  resource_group_name = azurerm_resource_group.terraform.name  
  virtual_network_name = azurerm_virtual_network.vnet.name  
  address_prefixes = [infoblox_ipv4_network.az_network.cidr]  
}
```

Create public IP - OPTIONAL

```
resource "azurerm_public_ip" "ip" {  
  name = "tfip"  
  location = azurerm_resource_group.terraform.location  
  resource_group_name = azurerm_resource_group.terraform.name  
  allocation_method = "Dynamic"  
  domain_name_label = "ibxvmiplabel"  
}
```

Create network interface - references infoblox_ipv4_allocation for IP address

```
resource "azurerm_network_interface" "ni" {  
  name = "tfni"  
  location = azurerm_resource_group.terraform.location  
  resource_group_name = azurerm_resource_group.terraform.name  
  
  ip_configuration {  
    name = "ipconfiguration"  
  }  
}
```

```

        subnet_id = azurerm_subnet.subnet.id
        private_ip_address_allocation = "static"
        private_ip_address = infoblox_ipv4_allocation.az_allocation.ip_addr
        public_ip_address_id = azurerm_public_ip.ip.id
    }
}

# Create virtual machine - attaches network interface created in previous block
resource "azurerm_virtual_machine" "vm" {
    name = "az-vm1"
    location = azurerm_resource_group.terraform.location
    resource_group_name = azurerm_resource_group.terraform.name
    network_interface_ids = [azurerm_network_interface.ni.id]
    vm_size = "Standard_DS1_v2"
    storage_image_reference {
        publisher = "Canonical"
        offer = "UbuntuServer"
        sku = "18.04-LTS"
        version = "latest"
    }
    storage_os_disk {
        name = "myosdisk"
        caching = "ReadWrite"
        create_option = "FromImage"
    }
    os_profile {
        computer_name = "az-vm1"
        admin_username = "madmin"
        admin_password = "Infoblox_123"
    }
    os_profile_linux_config {
        disable_password_authentication = false
    }
}

```

Apply Terraform Configuration Files

To deploy resources using the example configuration files above, save them both in a single directory on your computer. Open a terminal and navigate to that directory.

```

infoblox@az-cli-vm:~/tf-guide$ ls -al
total 16
drwxr-xr-x  2 infoblox infoblox 4096 Jul 28 10:34 .
drwxr-xr-x 17 infoblox infoblox 4096 Jul 27 10:51 ..
-rw-rw-r--  1 infoblox infoblox 2133 Jul 27 10:50 infoblox.tf
-rw-rw-r--  1 infoblox infoblox 2422 Jul 27 10:50 vm.tf
infoblox@az-cli-vm:~/tf-guide$

```

Run **terraform init** to initialize and install the Infoblox and Azure providers.

```
infoblox@az-cli-vm:~/tf-guide$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding infobloxopen/infoblox versions matching "2.0.1"...
- Finding hashicorp/azurerm versions matching "2.68.0"...
- Installing infobloxopen/infoblox v2.0.1...
- Installed infobloxopen/infoblox v2.0.1 (signed by a HashiCorp partner, key ID B355854AA5DDB18E)
- Installing hashicorp/azurerm v2.68.0...
- Installed hashicorp/azurerm v2.68.0 (signed by HashiCorp)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Run the command **terraform plan -out=plan1** to verify your environment and create an execution plan.

```
infoblox@az-cli-vm:~/tf-guide$ terraform plan -out=plan1

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# azurerm_network_interface.ni will be created
+ resource "azurerm_network_interface" "ni" {
  + applied_dns_servers      = (known after apply)
  + dns_servers              = (known after apply)
  + enable_accelerated_networking = false
  + enable_ip_forwarding     = false
  + id                      = (known after apply)
  + internal_dns_name_label  = (known after apply)
  + internal_domain_name_suffix = (known after apply)
  + location                 = "centralus"
  + mac_address              = (known after apply)
  + name                     = "tfni"
  + private_ip_address       = (known after apply)
```

```

infoblox-az-ccli-vn:~/tf-guides$ terraform apply plan1
infoblox_ipv4_network.container.az_vnet: Creating...
infoblox_ipv4_network.container.az_vnet: Creation complete after 2s [id=networkcontainer/ZG5zLn5ldHdcvntfY29udGFbmVYJDE5M14xNjguMjMzJzJAVmJQVwMA:192.168.233.0/24/default]
infoblox_ipv4_network.az_network: Creating...
azurerm_resource_group.terraform: Creating...
azurerm_resource_group.terraform: Creation complete after 1s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp]
azurerm_virtual_network.vnet: Creating...
azurerm_public_ip.ip: Creating...
azurerm_public_ip.ip: Creation complete after 3s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/publicIPAddresses/tfip]
azurerm_virtual_network.vnet: Creation complete after 5s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/virtualNetworks/tfvnet]
infoblox_ipv4_network.az_network: Still creating... [10s elapsed]
infoblox_ipv4_network.az_network: Creation complete after 13s [id=network/ZG5zLn5ldHdcvnskMTkylE20C4yMzMUMCBYNS8w:192.168.233.0/25/default]
infoblox_ipv4_allocation.az_allocation: Creating...
azurerm_subnet.subnet: Creating...
infoblox_ipv4_allocation.az_allocation: Creation complete after 1s [id=record:host/ZG5zLmhvc3QkLl9kZWZhdWx0LmNvbS5pYnhkZW1vLnF6LXZtMQ:az-vn1.bxdemo.com/default]
azurerm_subnet.subnet: Creation complete after 4s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/virtualNetworks/tfvnet/subnets/tfsub]
azurerm_network_interface.ni: Creating...
azurerm_network_interface.ni: Creation complete after 2s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/networkInterfaces/tfnl]
azurerm_virtual_machine.vm: Creating...
azurerm_virtual_machine.vm: Still creating... [10s elapsed]
azurerm_virtual_machine.vm: Still creating... [20s elapsed]
azurerm_virtual_machine.vm: Creation complete after 21s [id=/subscriptions/XXXXXXXXXXXXXXXXXXXX/resourceGroups/tf-ibx-grp/providers/Microsoft.Compute/virtualMachines/az-vm1]
infoblox_ipv4_association.az_associate: Creating...
infoblox_ipv4_association.az_associate: Creation complete after 1s [id=record:host/ZG5zLmhvc3QkLl9kZWZhdWx0LmNvbS5pYnhkZW1vLnF6LXZtMQ:az-vm1.bxdemo.com/default]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

```

Infoblox

Dashboards

Data Management

Cloud

Smart Folders

Grid

Administration

default

IPAM

VLANs

Super Host

DHCP

DNS

File Distribution

default

Network View

Quick Filter

None

On Filter Off

Show Filter

Toggle flat view

→

+

↻

🗑️

🔗

📄

	Network	Cloud Usage	Owned By	Comment	Delegated To	IPAM Utilization
	🖨️ 192.168.233.0/24	Cloud from adapter	Cloud adapter	New Azure VNet		50.0%

IPAM Home

192.168.233.0/24

Cloud IPv4 Network Container

Net Map

List

Quick Filter

None

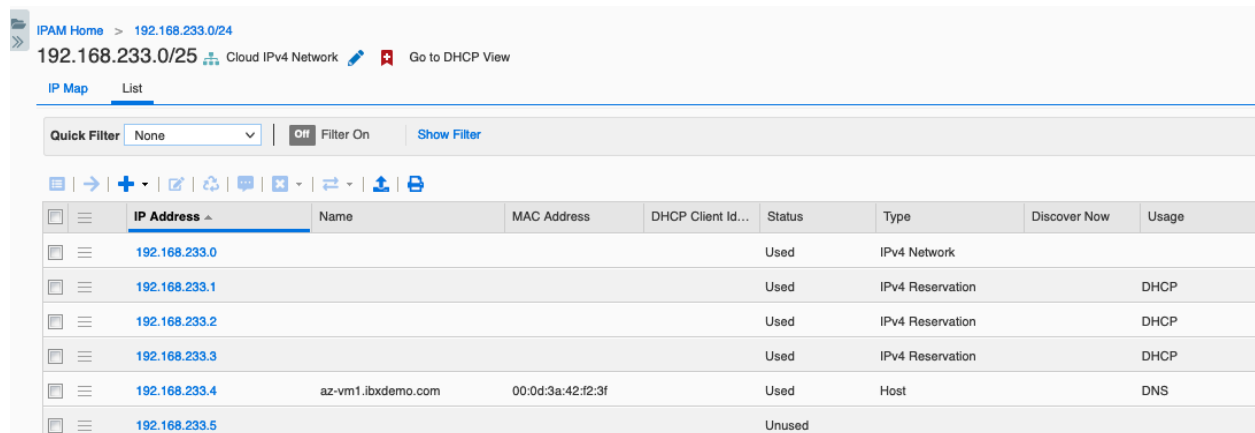
On Filter Off

Show Filter

Toggle flat view

		Network	Cloud Usage	Owned By	Comment	Delegated To	IPAM Utilization
		192.168.233.0/25	Cloud from adapter	Cloud adapter			3.9%

To view details of your network including the IP address allocated to your new VM, click on the network.



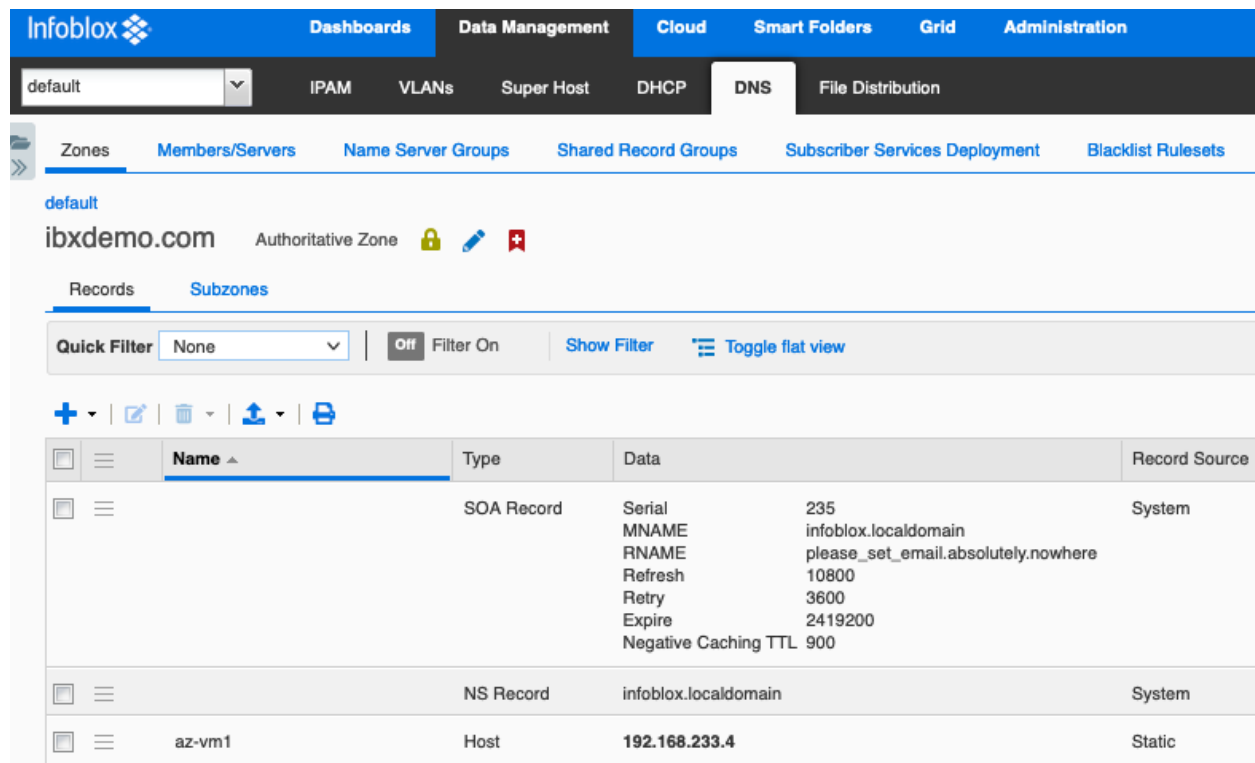
IPAM Home > 192.168.233.0/24
192.168.233.0/25 Cloud IPv4 Network Go to DHCP View

IP Map List

Quick Filter: None Filter On Show Filter

IP Address	Name	MAC Address	DHCP Client Id...	Status	Type	Discover Now	Usage
192.168.233.0				Used	IPv4 Network		
192.168.233.1				Used	IPv4 Reservation		DHCP
192.168.233.2				Used	IPv4 Reservation		DHCP
192.168.233.3				Used	IPv4 Reservation		DHCP
192.168.233.4	az-vm1.ibxdemo.com	00:0d:3a:42:f2:3f		Used	Host		DNS
192.168.233.5				Unused			

To view the DNS host record created for your VM, navigate to the **Data Management** → **DNS** → **Zones** and click on your DNS zone.



Infoblox Dashboards Data Management Cloud Smart Folders Grid Administration

default IPAM VLANs Super Host DHCP DNS File Distribution

Zones Members/Servers Name Server Groups Shared Record Groups Subscriber Services Deployment Blacklist Rulesets

default
ibxdemo.com Authoritative Zone

Records Subzones

Quick Filter: None Filter On Show Filter Toggle flat view

Name	Type	Data	Record Source
SOA Record		Serial 235 MNAME infoblox.localdomain RNAME please_set_email.absolutely.nowhere Refresh 10800 Retry 3600 Expire 2419200 Negative Caching TTL 900	System
NS Record		infoblox.localdomain	System
az-vm1	Host	192.168.233.4	Static

To view the resources created in your Azure subscription, login to the Azure portal and navigate to your new resource group.

tf-ibx-grp
Resource group

Search (Cmd+/) << + Create Edit columns Delete resource group Refresh Export to CSV Open query Assign to

Overview

Activity log

Access control (IAM)

Tags

Events

Settings

Deployments

Security

Policies

Properties

Locks

Cost Management

Essentials

Subscription (change) :
Subscription ID :
Tags (change) : [Click here to add tags](#)

Filter for any field... Type == all Location == all Add filter

Showing 1 to 5 of 5 records. Show hidden types

Name	Type
az-vm1	Virtual machine
myosdisk	Disk
tfip	Public IP address
tfni	Network interface
tfvnet	Virtual network

When you are ready to deprovision the example resources, back in your terminal run the **terraform destroy** command.

```
infoblox@az-c1i-vm:~/tf-guide$ terraform destroy -auto-approve
infoblox_ipv4_network_container.az_vnet: Refreshing state... [id=networkcontainer/ZG5zLn5ldHdvcmtfY29udGFpbmVyJDE5MT4xNjguMjMzLjAvMjQvQVMA:192.168.233.0/24/default]
infoblox_ipv4_network.az_network: Refreshing state... [id=network/ZG5zLn5ldHdvcn5kMTkyLjE2ODQyMzMuMUMC8yNS8w:192.168.233.0/25/default]
infoblox_ipv4_allocation.az_allocation: Refreshing state... [id=record:host/ZG5zLn5ldHdvcn5kMTkyLjE2ODQyMzMuMUMC8yNS8w:192.168.233.0/25/default]
azurerm_resource_group.terraform: Refreshing state... [id=/subscriptions/infoblox-open-azure-terraform-000000000000/resourceGroups/tf-ibx-grp]
azurerm_virtual_network.vnet: Refreshing state... [id=/subscriptions/infoblox-open-azure-terraform-000000000000/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/virtualNetworks/tfvnet]
azurerm_public_ip.ip: Refreshing state... [id=/subscriptions/infoblox-open-azure-terraform-000000000000/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/publicIPAddresses/tfip]
azurerm_subnet.subnet: Refreshing state... [id=/subscriptions/infoblox-open-azure-terraform-000000000000/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/virtualNetworks/tfvnet/subnets/tfsub]
azurerm_network_interface.ni: Refreshing state... [id=/subscriptions/infoblox-open-azure-terraform-000000000000/resourceGroups/tf-ibx-grp/providers/Microsoft.Network/networkInterfaces/tfni]
```

The IPAM and DNS objects will be removed from your Infoblox Grid and the Azure resources will be deleted.

Additional Resources

- Infoblox Terraform Provider Documentation: <https://docs.infoblox.com/display/ipamdriverterraform>
- Infoblox Plugin for Terraform GitHub: <https://github.com/infobloxopen/terraform-provider-infoblox>
- Infoblox Plugin for Terraform in the Terraform Registry: <https://registry.terraform.io/providers/infobloxopen/infoblox/latest>



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% of the Fortune 500.

Corporate Headquarters | 2390 Mission College Boulevard, Ste. 501 | Santa Clara, CA | 95054
+1.408.986.4000 | info@infoblox.com | 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).