

Deployment Guide

Managing Infoblox vNIOS for Azure Appliances Using the Azure CLI



Table of Contents

Introduction	3
Prerequisites	3
Limitations	3
Concepts	3
Azure Basics	3
Infoblox Use Cases for the Azure CLI	4
The Custom Deployment Use Case	4
The Automation Use Case	4
Getting Started With Azure CLI	4
Installing the Azure CLI	4
Windows	5
macOS	5
Linux	5
Cloud Shell	5
Basic Usage	7
Login	7
Shutdown an Infoblox vNIOs for Azure Appliance	8
Start an Infoblox vNIOs for Azure Appliance	9
Restart an Infoblox vNIOs for Azure Appliance	9
Virtual Machine Details	9
Scripting With the Azure CLI	10
Scripting Basics	10
Bash	10
JSON	10
Example Script: Deploy vNIOs Using ARM Template	10
Script Structure	10

Bash Script	11
JSON Parameter Template	11
Running the Script	15
PowerShell for Azure	16
Installing PowerShell and the Azure Module	16
Windows	17
macOS	17
Linux	17
Azure PowerShell Module	18
Azure PowerShell Command Examples	19
Login	19
Shutdown an Infoblox vNIOs for Azure Appliance	20
Start an Infoblox vNIOs for Azure Appliance	21
Restart an Infoblox vNIOs for Azure Appliance	21
Virtual Machine Details	21
Additional Resources	22

Introduction

Infoblox vNIOS for Azure is a virtualized Infoblox appliance designed for deployment as a VM (virtual machine) in Microsoft Azure. Infoblox vNIOS for Azure enables you to deploy robust, manageable, and cost effective Infoblox appliances in the Microsoft Cloud. Infoblox NIOS is the underlying software running on Infoblox appliances and provides core network services and a framework for integrating all the components of the modular Infoblox solution. It provides integrated, secure, and easy-to-manage DNS (Domain Name System), IPAM (IP address management) and other services.

Azure CLI is a command line based application which enables automation and command line management of Microsoft Azure services. The Azure CLI is also leveraged when managing Microsoft Azure services via scripting (API), providing for additional functions beyond what may be available through the standard GUI based Azure Portal and automation of tasks through the use of scripts.

PowerShell is a command line based shell with its own scripting language, commonly used by system administrators and other users for automation of tasks, execution of advanced commands or for performing administrative actions not available through the standard GUI based interfaces.

Prerequisites

The following are prerequisites for managing an Infoblox vNIOS for Azure appliance using the Azure CLI and/or PowerShell:

- Valid subscription and credentials for Azure.
- A supported client computer platform (Windows, macOS, or Linux).
- Appropriate permissions in Azure to deploy, stop, and start virtual machines.
- A working internet connection.

Limitations

The Azure CLI and Azure PowerShell modules are constantly evolving. This guide is written using version 2.11.1 of the CLI and the Azure PowerShell Az module version 4.6.1. Commands and syntax can change at any time and without notice. For information on current versions of the Azure CLI, refer to Microsoft documentation at <https://docs.microsoft.com/en-us/cli/azure/?view=azure-cli-latest>. For information on current Azure PowerShell Modules, refer to Microsoft documentation at <https://docs.microsoft.com/en-us/powershell/azure>. Additionally, this guide covers only a subset of common use cases and should not be considered a definitive list.

Concepts

Azure Basics

Before implementing the Azure CLI or Azure PowerShell to manage an Infoblox vNIOS for Azure VM, an administrator should understand common terms and objects available in Azure related to the deployment and management of vNIOS or usage of Azure CLI and PowerShell. The following are common objects and terms:

- **Azure CLI:** A set of commands used in a terminal or command line session to provision, manage, and deprovision Azure resources.
- **PowerShell:** A task automation and configuration management tool which consists of a command line shell and a scripting language.
- **Az Module:** A set of PowerShell commands (referred to as cmdlets) used for working with Azure resources from the PowerShell command line.

- **.NET:** An open source developer platform created by Microsoft. .NET is a prerequisite for using Azure PowerShell.
- **ARM Template:** Azure Resource Manager templates are files used to define Infrastructure as Code (IaC) for Microsoft Azure. ARM template files are written using JavaScript Object Notation (JSON). Deployments usually require a set of two files, the template and a parameters file.
- **Azure Subscription:** An account which is used to access Azure services and through which billing is managed.
- **VNet:** A virtual network where individual subnets and other network settings (such as security groups) are applied.
- **Availability Set:** Maintain maximum availability of servers/applications by placing more than one in an availability set.
- **Storage Account:** Holds the image files for the OS or boot diagnostics for a VM.
- **Resource Group:** A container which holds objects such as VM's and their related resources and can be used to simplify management of all objects within that resource group.

Infoblox Use Cases for the Azure CLI

The following are common use cases for using the Azure CLI or Azure PowerShell to manage Infoblox vNIOS for Azure VMs:

- You are deploying an Infoblox vNIOS for Azure appliance and require a configuration not supported by using the Azure Marketplace for deployment.
- You need to automate the deployment, shutdown, restart, or termination of an Infoblox vNIOS for Azure appliance.

The Custom Deployment Use Case

In this use case, you must deploy an Infoblox vNIOS for Azure VM with parameters that may not be configurable or allowed in the template called by the wizard when creating the VM in the Azure Portal. Examples include deploying the VMs into an availability set and deploying into a resource group that contains other resources.

The Automation Use Case

For this use case, an administrator can automate the deployment, shutdown, restart, or termination of an Infoblox vNIOS for Azure appliance. This can be useful in lab or QA environments where appliance availability requirements can change rapidly and would allow for administrators to automate changes as required.

Getting Started With Azure CLI

The Azure CLI provides a simple command line interface to the Azure cloud, enabling administrators to execute commands for operations which may not be available through the Azure web portal (<https://portal.azure.com/>) and to automate many operations, helping to reduce both the administrator's workload and errors that can occur when commands are executed manually. The extensions added during the installation of the Azure CLI are also leveraged for scripting access, such as by enabling a script to authenticate with the Azure cloud platform and login to a subscription.

Installing the Azure CLI

The Azure CLI version 2 is available for many operating systems. Instructions and download information for the supported platforms can be found at <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>. The following are basic installation methods for Windows, macOS, and Linux operating systems.

Windows

The Azure CLI is available as an MSI. Download and run the MSI available at <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest&tabs=azure-cli>. Or, follow instructions provided on the same link to install the CLI using PowerShell.

macOS

To install the Azure CLI on macOS, Microsoft recommends using the Homebrew package manager. Use the following command to install the CLI on macOS using Homebrew:

```
brew update && brew install azure-cli
```

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

Linux

The Azure CLI can be installed on many Linux distributions using their native package managers. Refer to <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli> for details of installation on the various distributions. For Debian and Ubuntu, Microsoft also provides an all-in-one script to run the install commands for you. To use this script, run the following command in your terminal:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

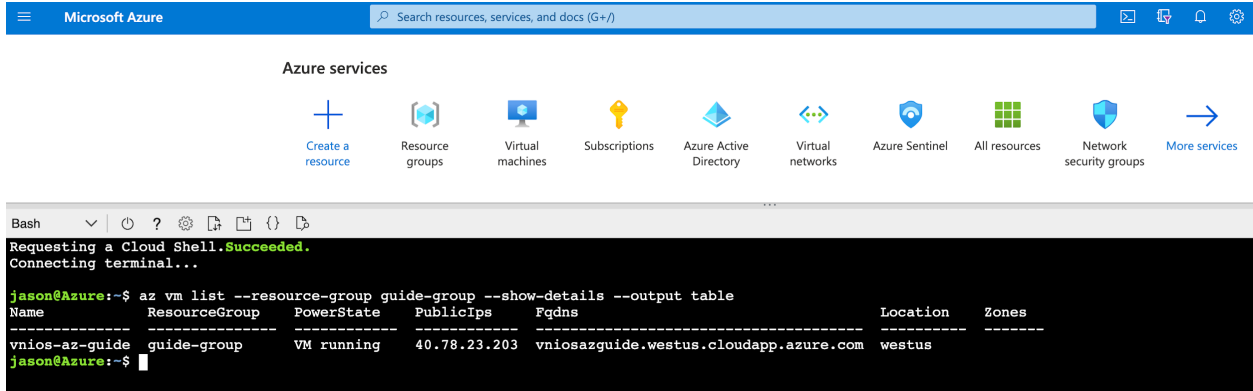
```
infoblox@az-cli-vm:~$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
export DEBIAN_FRONTEND=noninteractive
apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [48.9 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe i386 Packages [639 kB]
```

Cloud Shell

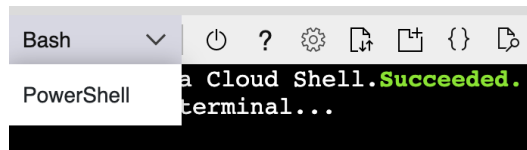
Microsoft also provides Azure Cloud Shell as an alternative method to utilize the Azure CLI. Azure Cloud Shell is a browser based command line application for Azure CLI and PowerShell. To access the Cloud Shell, navigate to <https://shell.azure.com> or select the Cloud Shell icon in the top menu bar of the Azure Portal, <https://portal.azure.com>.



The Cloud Shell will open in the bottom portion of your browser window. Since the shell is part of your Azure Portal session, you do not need to login again, and can begin running commands.



The Cloud Shell can also be used for Azure PowerShell. To switch into a PowerShell session, click the dropdown arrow next to Bash and select **PowerShell**.



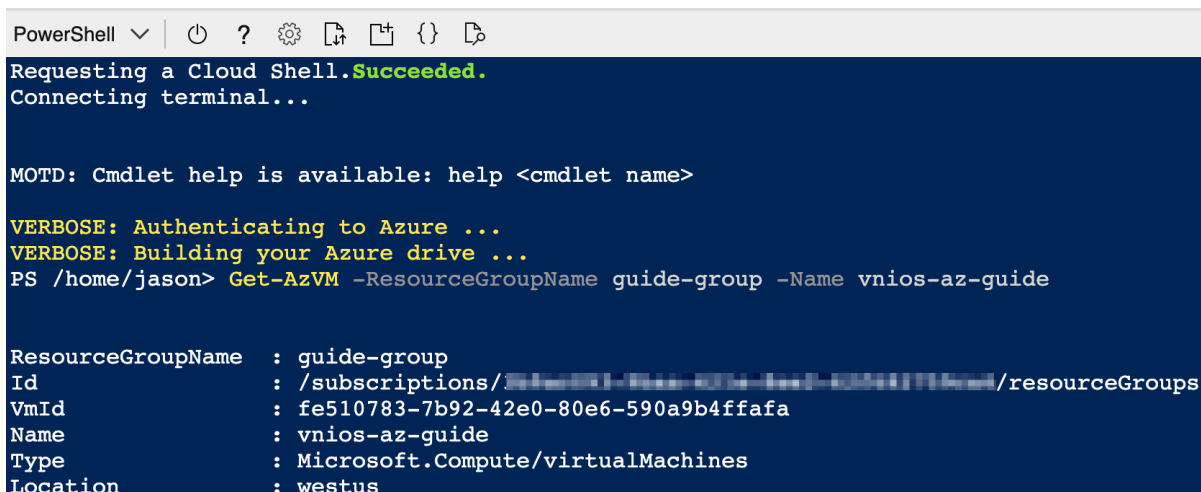
Click the **Confirm** button when prompted.

Switch to PowerShell in Cloud Shell

Switching will reconnect to the same container with a new PowerShell session.
Any running processes in active Bash sessions will go to background.



Once the PowerShell session is established, you can run Azure PowerShell cmdlets.



Basic Usage

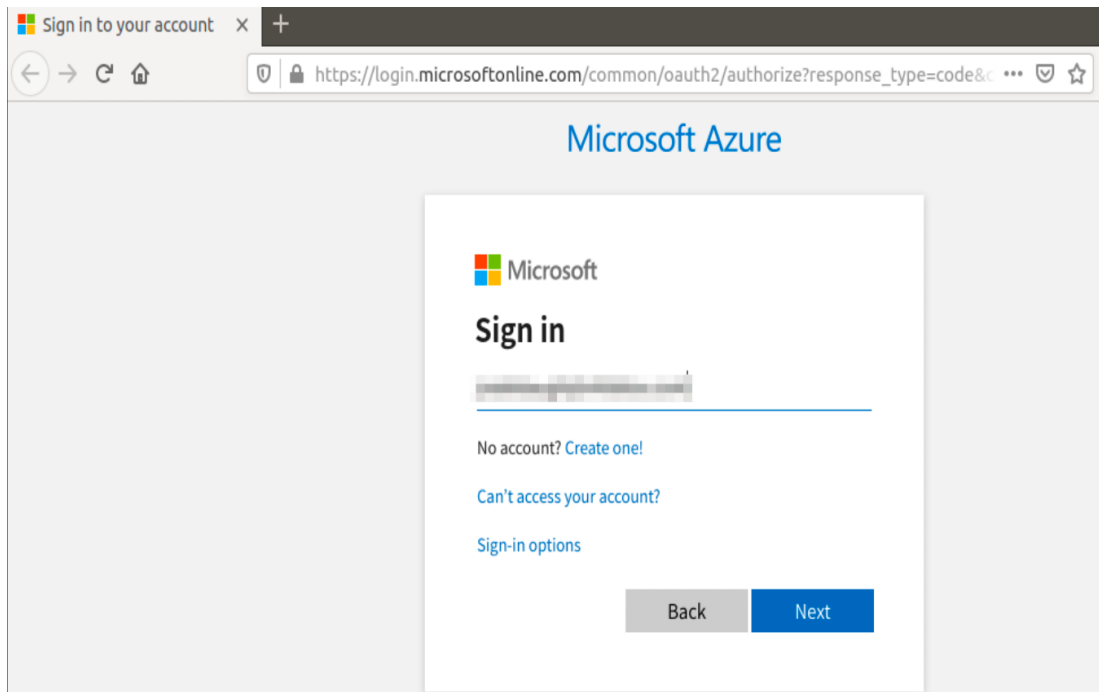
The installation of the Azure CLI adds libraries to your computer which are executed through either a terminal window for *nix based computers or through a command prompt for Windows based computers. The basic structure for CLI commands follows a pattern of **az <command> <subcommand> <parameters(optional)>**. For example, **az account show --output table**, which will show details of your Azure environment output in a table format. For more details on command syntax and usage, refer to Microsoft documentation at <https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli?view=azure-cli-latest>.

Login

In order to interact with Azure using the CLI, you will first need to login. To login to Azure using the CLI, run the command:

az login

If your system allows it, this will open your default browser to an Azure sign-in page. Follow the prompts to login to Azure.



Once you finish logging in through the browser, the CLI will list Azure subscriptions you have access to.

az vm stop --resource-group <group_name> --name <vm_name> --skip-shutdown.

```
infoblox@az-cli-vm:~$ az vm stop --resource-group guide-group --name vnios-az-guide --skip-shutdown
About to power off the specified VM...
It will continue to be billed. To deallocate a VM, run: az vm deallocate.
infoblox@az-cli-vm:~$
```

As the warning in the above screenshot shows, in order to deallocate resources for the VM, we need to use the **az vm deallocate** command. This command will shutdown the VM and deallocate compute and memory resources so you are not billed for them while the VM is not running. The following is an example of this command:

az vm deallocate --resource-group <group_name> --name <vm_name>.

```
infoblox@az-cli-vm:~$ az vm deallocate --resource-group guide-group --name vnios-az-guide
infoblox@az-cli-vm:~$
```

No feedback is provided when running this command. You will be returned to the prompt when it finishes executing.

Start an Infoblox vNIOS for Azure Appliance

To start an Infoblox vNIOS for Azure VM that has been stopped or deallocated, use the command:

az vm start --resource-group <group_name> --name <vm_name>.

```
infoblox@az-cli-vm:~$ az vm start --resource-group guide-group --name vnios-az-guide
infoblox@az-cli-vm:~$
```

No feedback is provided when running this command. You will be returned to the prompt when it finishes executing.

Restart an Infoblox vNIOS for Azure Appliance

To restart an Infoblox vNIOS for Azure VM, use the command:

az vm restart --resource-group <group_name> --name <vm_name>.

```
infoblox@az-cli-vm:~$ az vm restart --resource-group guide-group --name vnios-az-guide
infoblox@az-cli-vm:~$
```

No feedback is provided when running this command. You will be returned to the prompt when it finishes executing.

Virtual Machine Details

To get a detailed listing of VMs running in your Azure subscription, use the **az vm list** command. This command by itself outputs large amounts of information on all VMs in your subscription. You can limit the list to specific VMs by adding the **--resource-group** parameter. You can further refine the output using other available parameters. The following example is used to find information on my Infoblox vNIOS for Azure VM, including public IP and power state, and output them in a table format:

az vm list --resource-group <group_name> --show-details --output table.

```
infoblox@az-cli-vm:~$ az vm list --resource-group guide-group --show-details --output table
Name                ResourceGroup  PowerState  PublicIps  Fqdns                Location
-----
vnios-az-guide      guide-group    VM running  104.40.0.51  vniosazguide.westus.cloudapp.azure.com  westus
```

Alternatively, the **az vm show** command can be used to return the same information for individual VMs. For example, **az vm show --resource-group <group_name> --name <vm_name> --show-details --output table**, will return a table identical to the one shown in the screenshot above.

Scripting With the Azure CLI

Scripting can be very useful when attempting to automate administrative tasks or execute commands that may not be available through other means (such as through the Azure Portal). Azure provides libraries through the Azure CLI which enable connectivity into the Azure cloud platform using a variety of different scripting platforms. This allows for highly flexible and customizable scripts to be used for completing nearly any required task and using nearly any desired scripting language. These scripting languages can include (but are not limited to) Bash (as demonstrated in this guide) and Python.

Scripting Basics

Bash

Bash (Bourne-Again Shell), and its corresponding scripting language, is a 'shell' program used by default on many Linux based computers. Bash is a powerful shell, providing a vast array of features and capabilities, from very simple operations to complex commands and scripts. In this guide, example scripts which were written using bash are provided and these can be executed from any computer which supports the bash shell and has all required dependencies installed. Many of these dependencies are frequently met with default installations, though this will vary from computer to computer depending on how the operating system was installed.

JSON

JSON (JavaScript Object Notation) provides a specification used for the formatting of text in a format that is easy for both humans and computers to read and write. In the Azure CLI, the output of properties from many different commands is returned in JSON format by default. ARM templates used to deploy Infoblox vNIOS for Azure appliances also use the JSON format for the parameter and template files.

Example Script: Deploy vNIOS Using ARM Template

In the following script example, we will create a Resource Group, VNet, and vNIOS for Azure Virtual Machine using ARM templates. For details on creating ARM templates for deployment of vNIOS for Azure VMs, refer to the Infoblox Deployment Guide: Deploy vNIOS in Azure Using ARM Templates. The example script in this section uses a **parameters.json** file created using this guide:

<https://insights.infoblox.com/resources-deployment-guides/infoblox-deployment-guide-deploy-vnios-in-azure-using-arm-templates>.

Script Structure

The script used to deploy an Infoblox vNIOS for Azure VM consists of two pieces:

- The Bash script which contains the commands and logic to create the deployment. This script also sets some of the dynamic values that will be applied such as the Resource Group name. This script should be saved with a **.sh** filename extension.

- The JSON formatted parameters file which defines the configuration of the VM and the environment that it is being deployed in. The name for the VM, network configuration, and other options are set in this file. This parameters file should be saved with a `.json` filename extension.

Bash Script

```
#!/bin/bash
AZURE_TEMPLATE_URI=https://catalogartifact.azureedge.net/publicartifacts/infoblox.infoblox-nios-843-for-c8fff0dc-7421-466f-8f15-a5460f9b508d-vnios/Artifacts/mainTemplate.json
PARAMETERS_DIR="/home/infoblox"
PREFIX="guide-"
RESOURCE_GROUP="${PREFIX}group"
LOCATION="westus"
az group create --name "${RESOURCE_GROUP}" --location "${LOCATION}"
echo "-----"
echo -e "Creating Infoblox vNIOS for Azure appliance"
az deployment group create \
  --template-uri "${AZURE_TEMPLATE_URI}" \
  --parameters "${PARAMETERS_DIR}/parameters.json" \
  --resource-group "${RESOURCE_GROUP}"
echo "-----"
```

In the above example, the values highlighted in green should be changed to reflect the names or values that you want to use when the corresponding objects are created in Azure. This includes the directory where the parameters file is located, the prefix that will be used when objects specified in the script are created, name to be used for the Resource Group (appended with the prefix label), the location and the name of the parameters file.

The template uniform resource identifier (URI) above is for the current version of Infoblox vNIOS for Azure available at this writing. To find the template URI for other versions of vNIOS available on Azure, follow steps in the Deployment Guide:

<https://insights.infoblox.com/resources-deployment-guides/infoblox-deployment-guide-deploy-vnios-in-azure-using-arm-templates> to download templates from the Azure Portal. The URI for the main template can be found in the Parameters section, as the default value for the `artifactsBaseUrl`, of the `template.json` file.

```
"parameters": {
  "artifactsBaseUrl": {
    "type": "string",
    "metadata": {
      "description": "The base URL for dependent assets",
      "artifactsBaseUrl": ""
    },
    "defaultValue": "https://catalogartifact.azureedge.net/publicartifacts/infoblox.infoblox-nios-843-for-c8fff0dc-7421-466f-8f15-a5460f9b508d-vnios/Artifacts/mainTemplate.json"
  }
}
```

JSON Parameter Template

To support the script, a JSON formatted parameters file must also be used. The following table lists the supported parameters and allowed values that are used in the parameters file:

Parameter	Description	Allowed values
baseUrl	Base URL for dependent assets	
location	Location of resources.	<i>Must be a supported Azure region.</i>

vmName	Name for the Virtual Machine.	
vmSize	Size of the Virtual Machine. (corresponds to vNIOS model virtual hardware requirements)	"Standard_DS2","Standard_DS2_v2", "Standard_DS3", "Standard_DS3_v2","Standard_DS11_v2", "Standard_DS12_v2", "Standard_DS13_v2"
niosModel	vNIOS appliance model.	"IB-V825","IB-V1425", "IB-V2225","cp-v805","cp-v1405", "cp-v2205"
niosVersion	Version of NIOS software to use.	"latest", "843.383835.0"
adminPassword	Password for the command line and web interfaces.	Must be between 6 and 64 characters long and contains from at least three of the following groups: upper case character, lower case character, number, and special character.
virtualNetworkName	VNET name	
virtualNetworkExistingRGName	Resource Group containing existing network	
virtualNetworkAddressPrefix	Virtual Network Address prefix, using CIDR block notation	
vnetNewOrExisting	Identifies whether to use new or existing Virtual Network	
subnet1Name	Subnet 1 Name	
subnet1Prefix	Subnet 1 Prefix, using CIDR block notation	
subnet1StartAddress	Subnet 1 Starting IP Address	<i>Must be an unused IP address in the subnet</i>
subnet2Name	Subnet 2 Name	
subnet2Prefix	Subnet 2 Prefix, using CIDR block notation	
subnet2StartAddress	Subnet 2 Starting IP Address	<i>Must be an unused IP address in the subnet</i>
newStorageAccountName	Unique Name for Storage Account where the Virtual Machine's disks will be placed.	
storageAccountType	The type of storage account created.	"Premium_LRS"
storageAccountNewOrExisting	Identifies whether to use new or existing Storage Account	"new", "existing"
storageAccountExistingRG	Resource Group containing existing	

	storage account	
newStorageAccountForLogsName	Unique Name for Storage Account where the Virtual Machine's boot diagnostics will be placed.	
storageAccountForLogsType	The type of storage account created for boot diagnostics.	"Standard_LRS"
storageAccountForLogsNewOrExisting	Identifies whether to use new or existing Storage Account for boot diagnostics	"new", "existing"
storageAccountForLogsExistingRG	Resource Group containing existing storage account for boot diagnostics	
publicIPAddressName	Name of the Public IP Address	
publicIPDnsName	Unique DNS Prefix for the Public IP used to access the Virtual Machine.	
publicIPNewOrExistingOrNone	Indicates whether the Public IP is new or existing	"new", "existing", "none"
publicIPExistingRGName	Resource Group containing existing public IP	
availabilitySetNewOrExistingOrNone	Indicates whether the availability Set is new, none or existing	"new", "existing", "none"
availabilitySetName	Availability set name	
tempLicenseOption	Temporary license options.	"none", "TE", "CP", "TE-SoT", "CP-SoT"

The following is an example of the working parameters.json file used for this guide:

```
{
  "$schema":
    "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "value": "vnios-az-guide"
    },
    "location": {
      "value": "westus"
    },
    "vmSize": {
      "value": "Standard_DS11_v2"
    },
    "niosModel": {
      "value": "IB-V825"
    },
    "niosVersion": {
```

```

    "value": "843.383835.0"
  },
  "adminPassword": {
    "value": "Guide1ibx!"
  },
  "virtualNetworkName": {
    "value": "guide-vnet"
  },
  "virtualNetworkExistingRGName": {
    "value": "guide-group"
  },
  "virtualNetworkAddressPrefix": {
    "value": "10.10.0.0/16"
  },
  "vnetNewOrExisting": {
    "value": "new"
  },
  "subnet1Prefix": {
    "value": "10.10.0.0/24"
  },
  "subnet1Name": {
    "value": "lan1"
  },
  "subnet1StartAddress": {
    "value": "10.10.0.4"
  },
  "subnet2Prefix": {
    "value": "10.10.1.0/24"
  },
  "subnet2Name": {
    "value": "mgmt"
  },
  "subnet2StartAddress": {
    "value": "10.10.1.4"
  },
  "newStorageAccountName": {
    "value": "vniosazguidestor"
  },
  "storageAccountType": {
    "value": "Premium_LRS"
  },
  "storageAccountNewOrExisting": {
    "value": "new"
  },
  "storageAccountExistingRG": {
    "value": "guide-group"
  },
  "newStorageAccountForLogsName": {
    "value": "vniosazguideboot"
  },
  "storageAccountForLogsType": {
    "value": "Standard_LRS"
  },
  "storageAccountForLogsNewOrExisting": {
    "value": "new"
  },
  "storageAccountForLogsExistingRG": {
    "value": "guide-group"
  }

```

```

    },
    "storageAccountForLogsNeedFix": {
      "value": false
    },
    "publicIPAddressName": {
      "value": "vniosazguideip"
    },
    "publicIPDnsName": {
      "value": "vniosazguide"
    },
    "publicIPNewOrExistingOrNone": {
      "value": "new"
    },
    "publicIPExistingRGName": {
      "value": "guide-group"
    },
    "availabilitySetNewOrExistingOrNone": {
      "value": "none"
    },
    "tempLicenseOption": {
      "value": "TE-SoT"
    },
    "customData": {
      "value": ""
    }
  }
}

```

In the above example, the values shown can be used as long as no conflict occurs. In case of conflict or preference, replace the values highlighted in green with the values you want to use for your deployment (using allowable values where specified in the table).

Running the Script

To execute the script, be sure that the examples have been saved to files on the computer where you will be running them from. The script should use a .sh extension and be called using bash or other compatible program, while the parameters file should use a .json extension. Be sure that the parameters file is located in the correct directory (update the path for this in the script file as required). Prior to running the script, login to Azure using the **az login** command.

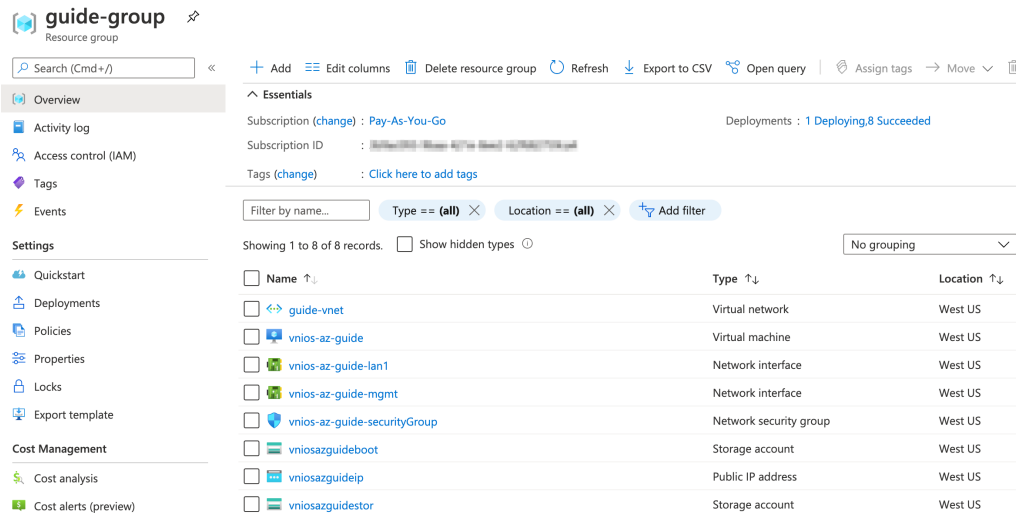
Once ready to run the script, call it by entering the full or relative path to your script followed by the file name, for example: **./deploy_vnios_vm.sh**.

```

infoblox@az-cli-vm:~$ ./deploy_vnios_vm.sh
{
  "id": "/subscriptions/████████████████████/resourceGroups/guide-group",
  "location": "westus",
  "managedBy": null,
  "name": "guide-group",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
-----
Creating Infoblox vNIOS for Azure appliance
{~ Finished ..
  "id": "/subscriptions/████████████████████/resourceGroups/guide-group/providers/Microsoft.Resources/deployments/mainTemplate",
  "location": null,
  "name": "mainTemplate",

```


When the deployment is complete, a detailed summary of the actions taken and resources created will be output in the terminal. The first few lines of this are shown in the screenshot above. The Infoblox vNIOS for Azure VM and other resources will now be available in the Azure Portal and via the CLI.



PowerShell for Azure

PowerShell is an automation and configuration management framework that consists of a command line shell and scripting language. PowerShell can be used to automate tasks an administrator may need to complete when managing an Infoblox vNIOS for Azure appliance. Commands executed in PowerShell are referred to as cmdlets, the names of which are generally structured with an action verb followed by a noun describing the object which will be acted on. PowerShell is built on Microsoft .NET framework and allows you to call .NET classes and methods directly from your PowerShell console. Cmdlets provide shortcuts to the .Net classes and will return the corresponding .NET object. As an example, executing the cmdlet **Get-Date** would return the **System.DateTime** .NET object, showing that executing the **Get-Date** cmdlet is equivalent to executing **[System.DateTime]::Now** command, both of which can be run in the PowerShell console and return the same results.

```

Administrator: Windows PowerShell
PS C:\> get-date
Friday, September 18, 2020 8:27:57 PM

PS C:\> [System.DateTime]::Now
Friday, September 18, 2020 8:28:01 PM

```

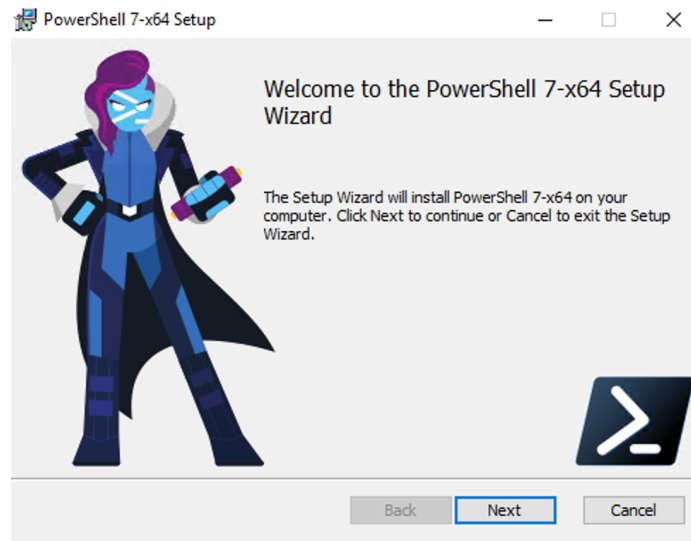
Installing PowerShell and the Azure Module

Versions of PowerShell are available for many operating systems. For this guide we will use PowerShell 7 which Microsoft recommends for use with the Azure PowerShell module. Instructions and download information

for the supported platforms can be found at <https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell?view=powershell-7>. The following are basic installation methods for Windows, macOS, and Linux.

Windows

Most Microsoft Windows operating systems come with PowerShell version 5.1 already installed. To install PowerShell 7, download the MSI package from GitHub: <https://github.com/PowerShell/PowerShell/releases>. Double click the installer and follow the prompts.



macOS

To install PowerShell on macOS, Microsoft recommends using the Homebrew package manager. Use the following command to install PowerShell on macOS using Homebrew:

brew cask install powershell

To run PowerShell from your macOS terminal after installation, use the command **pwsh**.

```
PowerShell 7.0.3  
Copyright (c) Microsoft Corporation. All rights reserved.  
  
https://aka.ms/powershell  
Type 'help' to get help.  
  
PS /> █
```

Linux

PowerShell can be installed on many Linux distributions using their native package managers. Refer to <https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-7> for details of installation on the various distributions. PowerShell can also be installed on Linux via direct download. For example, the following shows the process to install PowerShell 7 on Ubuntu 18.04:

1. Download the PowerShell package for Ubuntu from GitHub:
<https://github.com/PowerShell/PowerShell/releases/tag/v7.0.3>.



2. In a terminal run the following commands:

```
sudo dpkg -i powershell-lts_7.0.3-1.ubuntu.18.04_amd64.deb
sudo apt-get install -f
```

```
infoblox@gz-cli-vm:~$ sudo dpkg -i powershell-lts_7.0.3-1.ubuntu.18.04_amd64.deb
Selecting previously unselected package powershell-lts.
(Reading database ... 189642 files and directories currently installed.)
Preparing to unpack powershell-lts_7.0.3-1.ubuntu.18.04_amd64.deb ...
Unpacking powershell-lts (7.0.3-1.ubuntu.18.04) ...
dpkg: dependency problems prevent configuration of powershell-lts:
 powershell-lts depends on libltnng-ust0; however:
  Package libltnng-ust0 is not installed.

dpkg: error processing package powershell-lts (--install):
 dependency problems - leaving unconfigured
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Errors were encountered while processing:
 powershell-lts
infoblox@gz-cli-vm:~$ sudo apt-get install -f
Reading package lists... Done
Building dependency tree
Reading state information... Done
Correcting dependencies... Done
```

3. To run PowerShell from your Linux terminal after installation, use the command **pwsh**.

```
infoblox@gz-cli-vm:~$ pwsh
PowerShell 7.0.3
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/powershell
Type 'help' to get help.

PS /home/infoblox>
```

Azure PowerShell Module

The Azure PowerShell module can be installed using the PowerShellGet cmdlets. For Windows, macOS, and/or Linux run the following command from a PowerShell session:

```
if ($PSVersionTable.PSEdition -eq 'Desktop' -and (Get-Module -Name AzureRM -ListAvailable)) {
    Write-Warning -Message ('Az module not installed. Having both the AzureRM and ' +
    'Az modules installed at the same time is not supported.')
} else {
```

Install-Module -Name Az -AllowClobber -Scope CurrentUser

```
}
```

```
PS /home/infoblox> if ($PSVersionTable.PSEdition -eq 'Desktop' -and (Get-Module -Name AzureRM -ListAvailable)) {
>> Write-Warning -Message ('Az module not installed. Having both the AzureRM and ' +
>> 'Az modules installed at the same time is not supported.')
>> } else {
>> Install-Module -Name Az -AllowClobber -Scope CurrentUser
>> }
```

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): y
PS /home/infoblox> █

If prompted, enter **y** to install the module from the PSGallery repository.

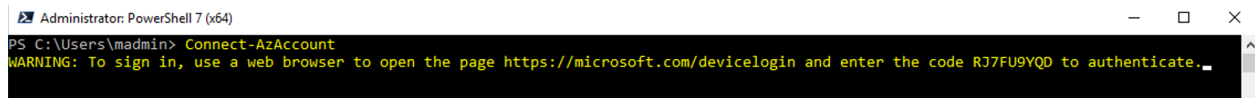
Azure PowerShell Command Examples

The command examples provided here are intended to allow an administrator to restart, shutdown, or terminate an Infoblox vNIOS for Azure appliance using PowerShell. These are not Infoblox specific and can be applied to any Virtual Machine operating in the Microsoft Azure cloud platform. Since PowerShell is a constantly evolving tool, these examples are subject to change without notice and are provided without any warranty.

Login

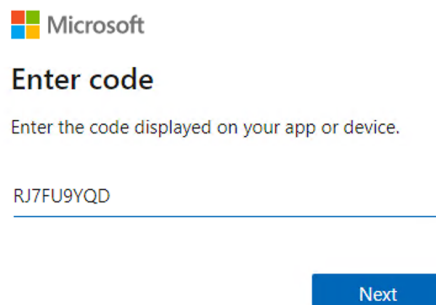
In order to interact with Azure using PowerShell, you will need to login to your Azure account. To login to Azure, open a PowerShell window and run the following cmdlet:

Connect-AzAccount



```
Administrator: PowerShell 7 (x64)
PS C:\Users\madmin> Connect-AzAccount
WARNING: To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code RJ7FU9YQD to authenticate.
```

You will see a message similar to the one shown in the above screenshot. Navigate to the URL it shows. Enter your code and click **Next**.



Follow additional prompts in your browser to sign in to your Azure account. Once you complete the process, return to your PowerShell window which will now show details of the Azure account you signed in with.

```
PS C:\Users\madmin> Connect-AzAccount
WARNING: To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code RJ7FU9YQD to authenticate.
Account          SubscriptionName TenantId          Environment
-----
@infoblox.com Pay-As-You-Go
PS C:\Users\madmin> _
```

If you have more than one subscription, use the following command to set the subscription you want to work in:

Set-AzContext -SubscriptionID <subscription_ID>

Use the following command to view which subscription is currently set and other information on your Azure PowerShell session:

Get-AzContext | Format-List

```
PS C:\Users\madmin> Set-AzContext -SubscriptionId
Name          Account          SubscriptionName Environment          TenantId
-----
Pay-As-You-Go (
@infoblox.com Pay-As-You-Go     AzureCloud
PS C:\Users\madmin> Get-AzContext | Format-List
Name          : Pay-As-You-Go (
Account       : @infoblox.com
Environment   : AzureCloud
Subscription  :
Tenant       :
TokenCache    : Microsoft.Azure.Commands.Common.Authentication.Core.ProtectedFileTokenCache
VersionProfile :
ExtendedProperties : {}
```

Shutdown an Infoblox vNIOS for Azure Appliance

To stop or shutdown an Infoblox vNIOS for Azure VM, use the following cmdlet:

Stop-AzVM -ResourceGroupName <group_name> -Name <vm_name>

Enter **y** or **yes** when prompted.

```
PS C:\Users\madmin> Stop-AzVM -ResourceGroupName "guide-group" -Name "vnios-az-guide"
Virtual machine stopping operation
This cmdlet will stop the specified virtual machine. Do you want to continue?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
OperationId : 1b1a0630-f164-4bc7-9379-8ffd00341dbe
Status      : Succeeded
StartTime   : 9/21/2020 6:26:37 PM
EndTime     : 9/21/2020 6:38:28 PM
Error       :
```

By default, the **Stop-AzVM** cmdlet deallocates resources from your VM so you will not be charged for them while the VM is not running. To run the cmdlet without prompting you for confirmation, add the **-Force** parameter. This cmdlet may take a long time to complete. To start the operation and return to the PowerShell prompt without waiting for it to complete, add the **-NoWait** parameter.

Start an Infoblox vNIOs for Azure Appliance

To start an Infoblox vNIOs for Azure VM that has been stopped or deallocated, use the cmdlet:

```
Start-AzVM -ResourceGroupName <group_name> -Name <vm_name>
```

```
PS C:\Users\madmin> Start-AzVM -ResourceGroupName "guide-group" -Name "vnios-az-guide"

OperationId : 626c4c0c-5896-465b-911d-e53856f238f7
Status      : Succeeded
StartTime   : 9/21/2020 6:45:48 PM
EndTime     : 9/21/2020 6:46:53 PM
Error       :
```

This cmdlet may take a long time to complete. To start the operation and return to the PowerShell prompt without waiting for it to complete, add the **-NoWait** parameter.

Restart an Infoblox vNIOs for Azure Appliance

To restart an Infoblox vNIOs for Azure VM, use the following cmdlet:

```
Restart-AzVM -ResourceGroupName <group_name> -Name <vm_name>
```

```
PS C:\Users\madmin> Restart-AzVM -ResourceGroupName "guide-group" -Name "vnios-az-guide"

OperationId : 9fd0b2c5-5db6-4d52-b16c-23a56470b789
Status      : Succeeded
StartTime   : 9/21/2020 6:51:24 PM
EndTime     : 9/21/2020 6:51:55 PM
Error       :
```

This cmdlet may take a long time to complete. To start the operation and return to the PowerShell prompt without waiting for it to complete, add the **-NoWait** parameter.

Virtual Machine Details

To find details of an Infoblox vNIOs for Azure VM, use the following cmdlet:

```
Get-AzVM ResourceGroupName <group_name> -Name <VM_name>
```

```

PS C:\Users\madmin> Get-AzVM -ResourceGroupName "guide-group" -Name "vnios-az-guide"

ResourceGroupName : guide-group
Id                : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/guide
                  -group/providers/Microsoft.Co
                  mpute/virtualMachines/vnios-az-guide
VmId              : fe510783-7b92-42e0-80e6-590a9b4ffafa
Name              : vnios-az-guide
Type              : Microsoft.Compute/virtualMachines
Location          : westus
Tags              : {}
DiagnosticsProfile : {BootDiagnostics}
HardwareProfile   : {VmSize}
NetworkProfile    : {NetworkInterfaces}
OSProfile         : {ComputerName, AdminUsername, LinuxConfiguration, Secrets}
Plan              : {Name, Publisher, Product}
ProvisioningState : Succeeded
StorageProfile    : {ImageReference, OsDisk, DataDisks}

```

By default, this cmdlet provides the model view, showing user specified properties. To see the instance view, which includes detailed status of the VM, add the **-Status** parameter.

```

PS C:\Users\madmin> Get-AzVM -ResourceGroupName "guide-group" -Name "vnios-az-guide" -Status

ResourceGroupName : guide-group
Name              : vnios-az-guide
HyperVGeneration : V1
BootDiagnostics   :
  ConsoleScreenshotBlobUri : https://vniosazguideboot.blob.core.windows.net/bootdiagnostics-
vniosazgu-fe510783-7b92-42e0-80e6-590a9b4ffafa/vnios-az-guide.fe510783-7b92-42e0-80e6-590a9b
4ffafa.screenshot.bmp
  SerialConsoleLogBlobUri  : https://vniosazguideboot.blob.core.windows.net/bootdiagnostics-
vniosazgu-fe510783-7b92-42e0-80e6-590a9b4ffafa/vnios-az-guide.fe510783-7b92-42e0-80e6-590a9b
4ffafa.serialconsole.log
Disks[0]          :
  Name             : vnios-az-guide2waqrd7o6tfsm
  Statuses[0]      :
    Code           : ProvisioningState/succeeded
    Level          : Info
    DisplayStatus  : Provisioning succeeded
    Time           : 9/21/2020 6:45:49 PM
VMAgent           :
  VmAgentVersion   : Unknown
  Statuses[0]      :
    Code           : ProvisioningState/Unavailable
    Level          : Warning
    DisplayStatus  : Not Ready
    Message        : VM status blob is found but not yet populated.
    Time           : 9/21/2020 7:01:50 PM
  Statuses[0]      :
    Code           : ProvisioningState/succeeded
    Level          : Info
    DisplayStatus  : Provisioning succeeded
    Time           : 9/21/2020 6:51:25 PM
  Statuses[1]      :
    Code           : PowerState/starting
    Level          : Info
    DisplayStatus  : VM starting

```

Additional Resources

- Azure CLI overview: <https://docs.microsoft.com/en-us/cli/azure/?view=azure-cli-latest>
- Azure CLI - az vm command and subcommand documentation: <https://docs.microsoft.com/en-us/cli/azure/vm?view=azure-cli-latest>

- PowerShell overview: <https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7>
- Azure PowerShell Module: <https://docs.microsoft.com/en-us/powershell/azure/?view=azps-4.6.1>
- Deployment Guide, Deploy vNIOS in Azure Using ARM Templates: <https://insights.infoblox.com/resources-deployment-guides/infoblox-deployment-guide-deploy-vnios-in-azure-using-arm-templates>
- Azure Cloud Shell overview: <https://docs.microsoft.com/en-us/azure/cloud-shell/overview>



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 | 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).