infoblox.

DEPLOYMENT GUIDE

# Managing Infoblox vNIOS for AWS Appliances Using the AWS CLI

## Table of Contents

## Introduction

Infoblox vNIOS for Amazon Web Services (AWS) is a virtualized Infoblox appliance designed for deployment as a VM (virtual machine) instance in AWS, a collection of integrated cloud services in the Amazon Cloud.

Infoblox vNIOS for AWS enables you to deploy robust, manageable, and cost effective Infoblox appliances in the Amazon 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), DHCP (Dynamic Host Configuration Protocol), IPAM (IP address management) and other services.

AWS is a highly available cloud platform that provides multiple services, including the ability to operate a virtualized computer (VM) and applications that can be made available globally or from specific networks. Amazon provides the AWS CLI (Command Line Interface) and AWS Shell as separate applications which are simple to install and use for management of available AWS services.

## Prerequisites

The following are prerequisites for managing an Infoblox vNIOS for AWS appliance using the AWS CLI (v2 of the CLI is used for this guide):

- Valid subscription and credentials for AWS.
- A supported client computer platform (64 bit Windows XP or later, Linux, Unix or macOS).
- Appropriate permissions in AWS to stop and start VM instances.
- A working Internet connection.

## Limitations

The AWS CLI or AWS Shell may not support all services or operations. For further information on which services and operations are and are not supported, refer to the AWS CLI command reference (https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html). Additional operations may be available through the AWS Management Console (web based GUI), or through scripting using the AWS API.

## Concepts

### Introduction to the AWS CLI, AWS Shell, and the REST API

The AWS CLI and AWS Shell provide an alternative to the AWS Management Console (https://console.aws.amazon.com/console/) and to the AWS API for managing resources and services in AWS. The CLI is installed as a single application and runs in a standard command prompt or terminal window. Commands are not as flexible as what is available through the API; however, the CLI gives an easy to use interface which does not require specialized expertise to use and will generally also have faster response times.

The Infoblox REST API, also referred to as the WAPI (or web API), provides an interface for NIOS (or vNIOS instances) that uses HTTP methods for operations and supports input and output in both JSON and XML formats. The WAPI is a NIOS interface, not part of the AWS CLI; basic WAPI usage is included in this guide in support of automating various tasks to manage your Infoblox vNIOS for AWS instances.

While not covered in this guide, AWS modules are also available for Windows PowerShell. Refer to https://aws.amazon.com/powershell/ for more details regarding Windows PowerShell.

## AWS Basics

Before implementing the AWS CLI or AWS Shell to manage an Infoblox vNIOS for AWS EC2 instance, an administrator should understand common terms or objects available in AWS related to the implementation of vNIOS or the usage of the AWS CLI and AWS Shell. The following are common objects and terms:

- **AWS CLI**: The AWS Command line interface is a single application which installs on your computer and enables you to easily manage supported resources and services in AWS.
- **AWS Shell**: Installed under Python, the AWS Shell functions as a replacement for the AWS CLI but works essentially the same as the CLI with the exception that you do not need to specify the top level command (aws) each time you run a command.
- **GitHub**: An online repository and hosting service which allows developers to share both public and private source code and other development related tasks. The source code for both the AWS CLI and AWS Shell is available on GitHub.
- **Python**: A commonly used programming language that is supported on many different operating systems and allows developers to develop fully functional programs or scripts.
- **PIP**: The package management system used for installing and managing software packages in python.
- **Boto**: An AWS SDK (software developer kit) for python. Boto is required when using the Amazon API in python for scripting purposes.
- **JSON**: JavaScript Object Notation, a human readable and open-standard format commonly used for large blocks of data that is supported by many programming languages. JSON is the default output format for the AWS CLI.
- **EC2**: Elastic Compute Cloud, the service which provides virtualized computing resources in the Amazon cloud. This is the service which Infoblox vNIOS for AWS appliances operate under.

- **VPC**: Virtual Private Cloud, a network service which provides network connectivity and isolation for any applications and resources that you are running in the Amazon cloud. A VPC is your own network where you will define firewalls, routing, and subnets that enable you to connect in and out of the Amazon cloud.

## AWS CLI Applications

The AWS CLI version 2 is available for many operating systems. Detailed installation instructions can be found in the user guide at https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html. The following are basic installation methods for each operating system.

### Windows

The AWS CLI version 2 is available in an MSI installer for 64 bit Windows operating systems. Download and run the MSI available at https://aws.amazon.com/cli/.

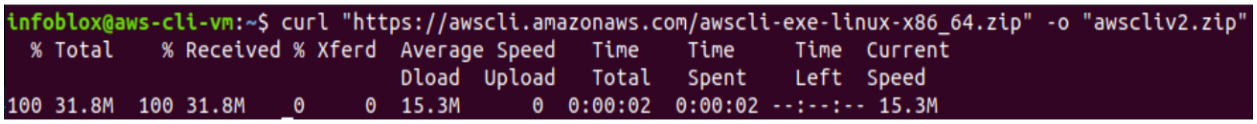
### macOS

The AWS CLI version 2 is available for 64 bit macOS versions as a PKG file. Download the PKG from https://aws.amazon.com/cli/. Double click the PKG to install.

### Linux

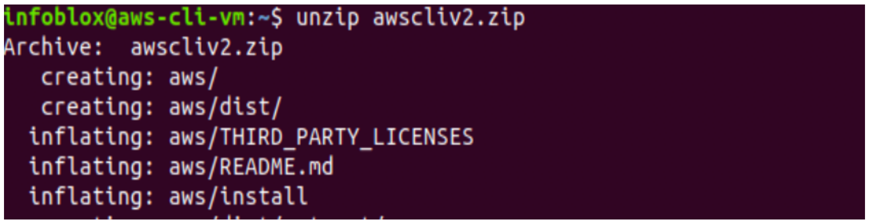
The AWS CLI version 2 is available for both 64 bit and ARM versions of Linux. Download the zip file from https://aws.amazon.com/cli/. Unzip and run the install file. The following is an example of installation on Ubuntu Linux:

1. First, use the curl command to download the CLI in a zip file.



```
infoblox@aws-cli-vm:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 31.8M  100 31.8M    0     0  15.3M      0  0:00:02  0:00:02 --:--:-- 15.3M
```

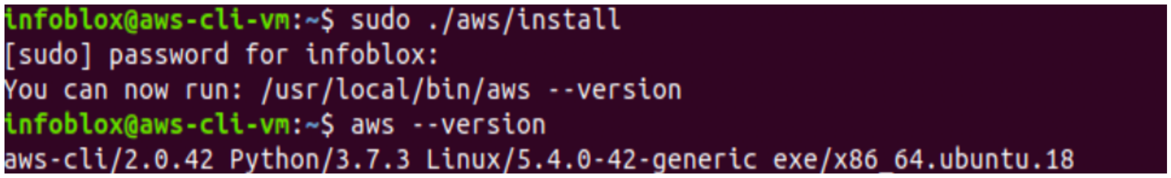
2. Unzip the file.



```
infoblox@aws-cli-vm:~$ unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/THIRD_PARTY_LICENSES
  inflating: aws/README.md
  inflating: aws/install
```

3. Run the install program.

4. Verify the installation with the **aws --version** command.



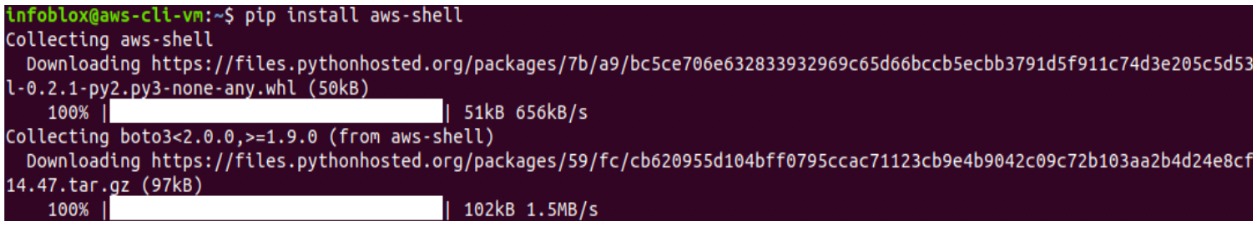
```
infoblox@aws-cli-vm:~$ sudo ./aws/install
[sudo] password for infoblox:
You can now run: /usr/local/bin/aws --version
infoblox@aws-cli-vm:~$ aws --version
aws-cli/2.0.42 Python/3.7.3 Linux/5.4.0-42-generic exe/x86_64.ubuntu.18
```

### Docker

The AWS CLI version 2 is also available on an official Docker image, supported and maintained by Amazon. To download the latest image, use the docker command: **docker pull amazon/aws-cli:latest**. Refer to AWS documentation for usage of the docker image: [https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-docker.html](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-docker.html).

### AWS Shell

The AWS Shell is available as a python module. It can be found on GitHub along with usage instructions at [https://github.com/awslabs/aws-shell](https://github.com/awslabs/aws-shell). The module can be installed using the pip package manager for python: **pip install aws-shell**.



```
infoblox@aws-cli-vm:~$ pip install aws-shell
Collecting aws-shell
  Downloading https://files.pythonhosted.org/packages/7b/a9/bc5ce706e632833932969c65d66bccb5ecbb3791d5f911c74d3e205c5d53
l-0.2.1-py2.py3-none-any.whl (50kB)
    100% |████████████████████████████████| 51kB 656kB/s
Collecting boto3<2.0.0,>=1.9.0 (from aws-shell)
  Downloading https://files.pythonhosted.org/packages/59/fc/cb620955d104bff0795ccac71123cb9e4b9042c09c72b103aa2b4d24e8cf
14.47.tar.gz (97kB)
    100% |████████████████████████████████| 102kB 1.5MB/s
```

## Infoblox Use Cases for the AWS CLI

The following are common use cases for using the AWS CLI in managing Infoblox vNIOS for AWS appliances:

- You do not have access to an appliance and must shutdown, restart or terminate it.
- You need to automate management of Infoblox appliances and servers.

### The Offline Appliance Use Case

In this use case, the Infoblox vNIOS for AWS appliance is no longer reachable. This may happen due to a service issue with the appliance itself, or a general connectivity issue and can happen frequently in lab or testing environments. In such cases, it may not be desired to use the web-based AWS management console. As an alternative, the AWS CLI can be used to complete the desired task.

### The Automation Use Case

For this use case, an administrator can automate the shutdown and restart or termination of an Infoblox vNIOS for AWS appliance. This can be helpful in lab or QA environments where systems are spun up and shutdown as needed and would allow for administrators to make the systems available as required.

## Usage

### AWS CLI

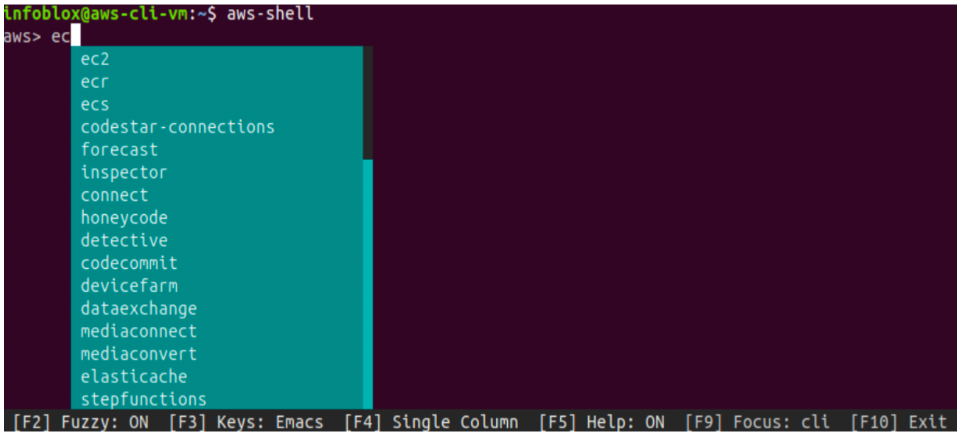
Amazon provides a large number of services which can be managed through the AWS CLI, including the EC2 service which Infoblox vNIOS for AWS appliances operate under. When running a command for the AWS CLI, you will begin with the base call (aws), followed by the top-level command which is generally the service that you are managing. Next in the command structure would be the subcommand, or the actual operation that you wish to perform and then any options or additional parameters that you need to pass in the command would be specified next.

**$ aws <command> <subcommand> [options and parameters]**

Tab completion (auto completion) for commands is available for some platforms but is not enabled by default. Refer to http://docs.aws.amazon.com/cli/latest/userguide/cli-command-completion.html for more details regarding this feature.

### AWS Shell

To launch the AWS Shell, run the command: aws-shell. This will bring you to the aws prompt. With the exception that you do not enter the top level command, the syntax for commands is similar to that of the AWS CLI. The AWS Shell provides the advantage of fuzzy completion and an integrated help function. This makes for a much friendlier interface, especially useful to administrators who may not be as familiar with the available commands.



```
infoblox@aws-cli-vm:~$ aws-shell
aws> ec
        ec2
        ecr
        ecs
        codestar-connections
        forecast
        inspector
        connect
        honeycode
        detective
        codecommit
        devicefarm
        dataexchange
        mediaconnect
        mediaconvert
        elasticache
        stepfunctions
 [F2] Fuzzy: ON  [F3] Keys: Emacs  [F4] Single Column  [F5] Help: ON  [F9] Focus: cli  [F10] Exit
```

### Infoblox REST API

Using a supported application (such as curl or even a web browser), you can send http GET, POST, PUT and DELETE requests to a supporting server. This enables cross-functional API access to manage systems such as your Infoblox services. In the following example, we first pass the **command** name. Next, we can see multiple options that may be set in the command and are prefixed with a – (hyphen), the **credentials** to be used for the connection to the endpoint that the command is being sent to, then the type of **action** (which may use either **GET, POST, PUT or DELETE**). Following that, we see the **URL** which sets where the command is being sent to (and includes the WAPI version to be used for the endpoint the command is being sent to) and the resource it will act on (for example vdiscoverytask). Lastly, you will see any data or function calls set after the URL. Note: the strings highlighted in red are common parameters that you must update to match your environment, such as the username and password.

**$ curl -k -u admin:infoblox -X GET https://<GM_address>/wapi/v2.11/<resource>**

## Command Examples

The examples provided here are intended to allow an administrator to restart, shutdown, or terminate an Infoblox vNIOS for AWS appliance using the AWS CLI. These examples are not Infoblox specific and can be applied to any Virtual Machine operating in the Amazon AWS cloud platform. When using these examples in the AWS Shell, simply omit the "aws" portion of the command.

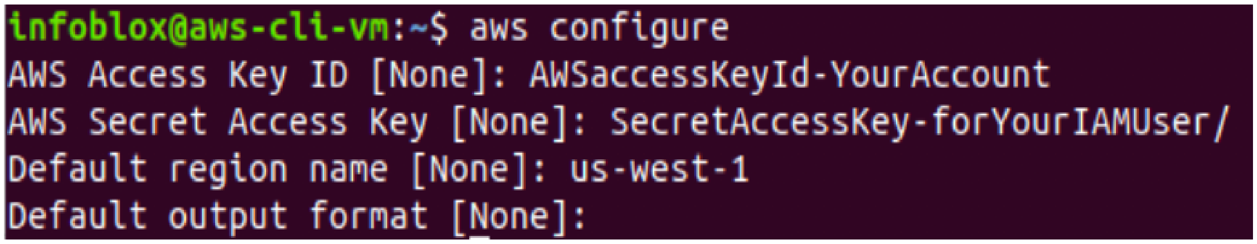
*Note: The examples included in this guide are subject to change without notice and are provided without any warranty.*

### Environment Setup

Prior to using the AWS CLI or AWS Shell to manage Infoblox vNIOS for AWS appliances, you will first need to set up your environment. This includes specifying your credentials (keys) which will be used to login to AWS, and the default region that you will connect to. Using the Infoblox REST API does not require any special configuration when using a web browser. The ability to run commands through a command line interface/terminal may require additional software components, such as CURL (which is open source and freely available).

#### AWS CLI

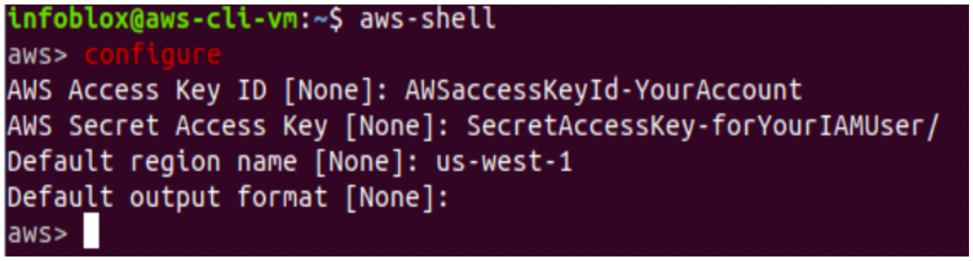
To set your access key ID, secret access key, default region, and default output format, run the command: **aws configure**. Enter the appropriate values at each prompt.



```
infoblox@aws-cli-vm:~$ aws configure
AWS Access Key ID [None]: AWSaccessKeyId-YourAccount
AWS Secret Access Key [None]: SecretAccessKey-forYourIAMUser/
Default region name [None]: us-west-1
Default output format [None]:
```

#### AWS Shell

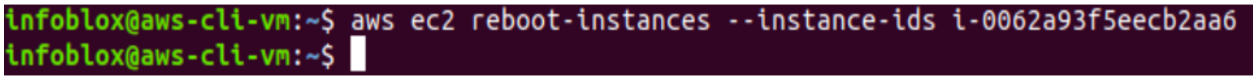
To set up your environment in the AWS Shell, run the shell and use the command: **configure**. Enter the appropriate values at each prompt.



```
infoblox@aws-cli-vm:~$ aws-shell
aws> configure
AWS Access Key ID [None]: AWSaccessKeyId-YourAccount
AWS Secret Access Key [None]: SecretAccessKey-forYourIAMUser/
Default region name [None]: us-west-1
Default output format [None]:
aws>
```

## Restart an Infoblox vNIOS for AWS Appliance

Once you have configured your CLI environment with credentials for the account where your Infoblox vNIOS for AWS instance is running, you can restart it using this command: **aws ec2 reboot-instances --instance-ids <instance_id>**.

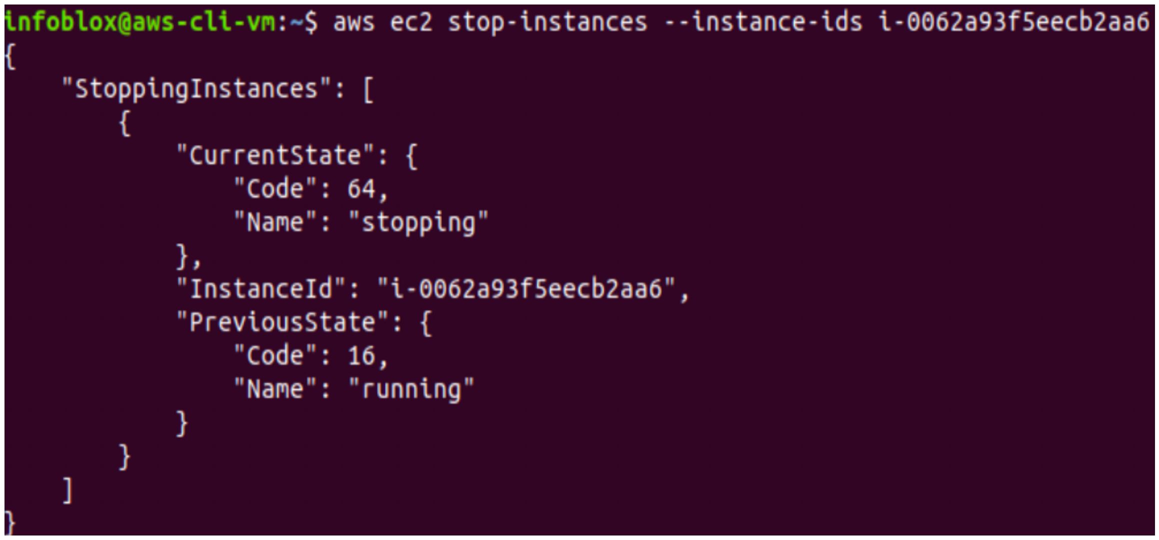


```
infoblox@aws-cli-vm:~$ aws ec2 reboot-instances --instance-ids i-0062a93f5eecb2aa6
infoblox@aws-cli-vm:~$
```

No feedback is provided when running this command. You will be returned back to the prompt once it finishes executing. Multiple instances can be restarted by adding additional instance IDs, for example: **aws ec2 reboot-instances --instance-ids i-111111111 i-2222222 i-nnnnnnn**. If you need to find the instance IDs for your instances you can use the **aws ec2 describe-instances** command.

## Shutdown an Infoblox vNIOS for AWS Appliance

There may be cases where you wish to be able to shut down an EC2 instance, such as if it is used for testing purposes and you do not wish to leave it running while it is not being used. To shut down the VM instance, use the following command: **aws ec2 stop-instances --instance-ids <instance_id>**.

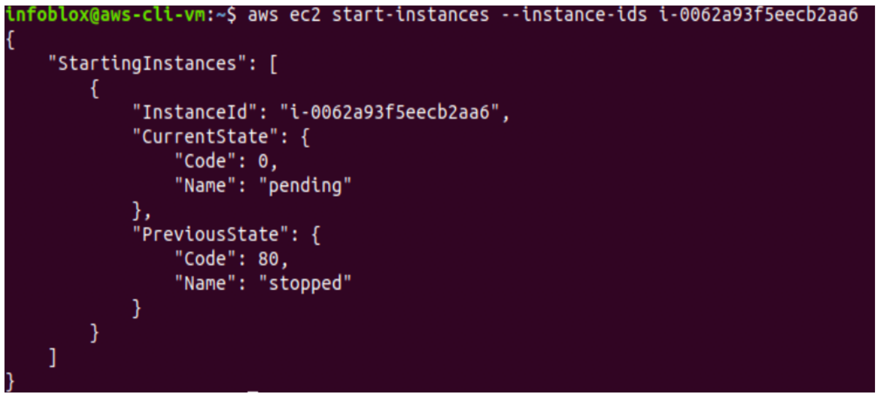


```
infoblox@aws-cli-vm:~$ aws ec2 stop-instances --instance-ids i-0062a93f5eecb2aa6
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-0062a93f5eecb2aa6",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

The screenshot above shows the output for a successful run. Multiple instances can be stopped with a single command by adding additional instance IDs, for example: **aws ec2 stop-instances --instance-ids i-111111111 i-2222222 i-nnnnnnn**.

## Start an Infoblox vNIOS for AWS Appliance

To start an Infoblox vNIOS for AWS EC2 instance that has been shut down, use the following command: a**ws ec2 start-instances --instance-ids <instance_id>**.

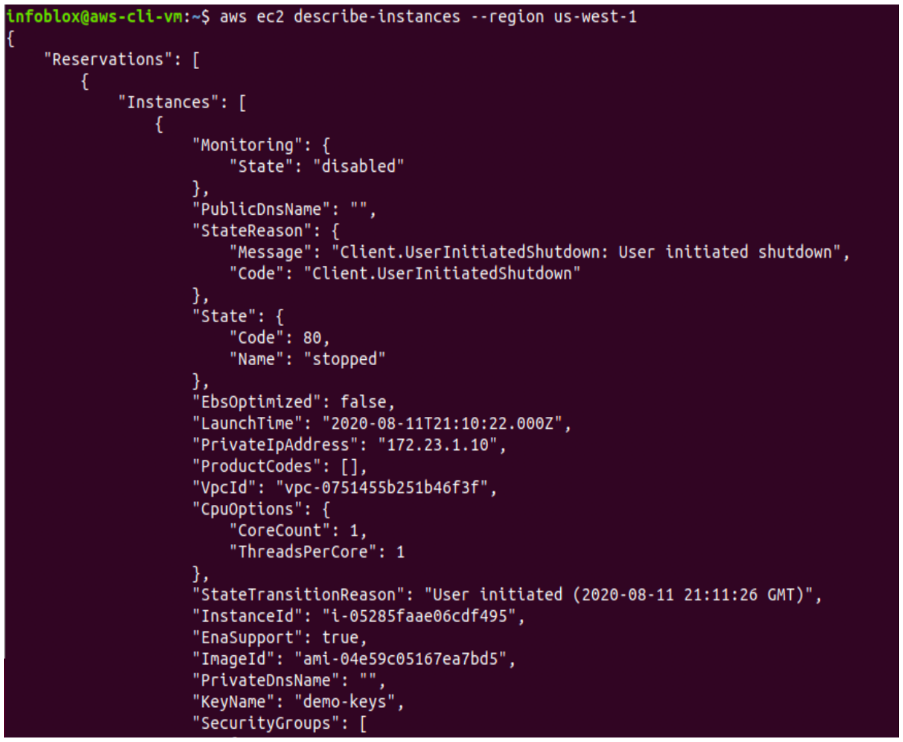


```
infoblox@aws-cli-vm:~$ aws ec2 start-instances --instance-ids i-0062a93f5eecb2aa6
{
    "StartingInstances": [
        {
            "InstanceId": "i-0062a93f5eecb2aa6",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}
```

The screenshot above shows the output for a successful run. Multiple instances can be started with a single command by adding additional instance IDs, for example: **aws ec2 start-instances --instance-ids i-111111111 i-2222222 i-nnnnnnn**.

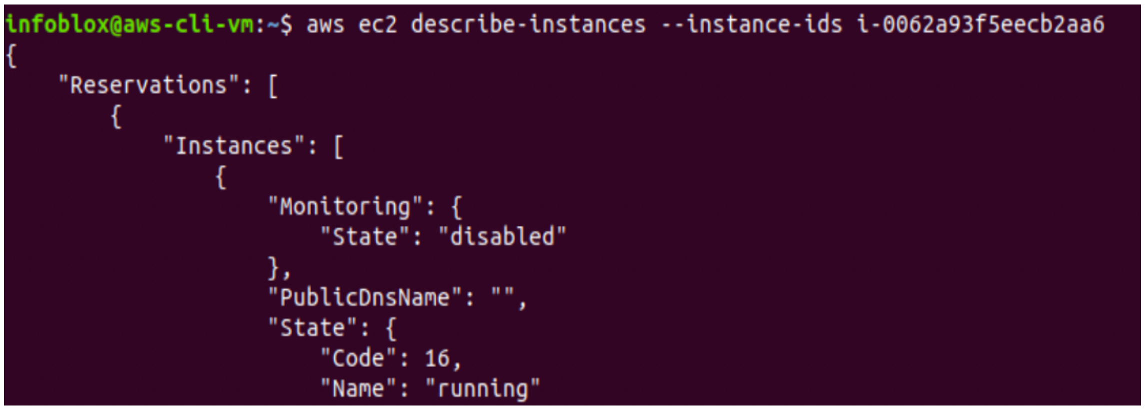
## Virtual Machine Details

To get a detailed listing of all your EC2 instances in a particular region, use the following command: **aws ec2 describe-instance --region <region>**. If the region argument is omitted, the command will return information on instances in the default region set for your environment.



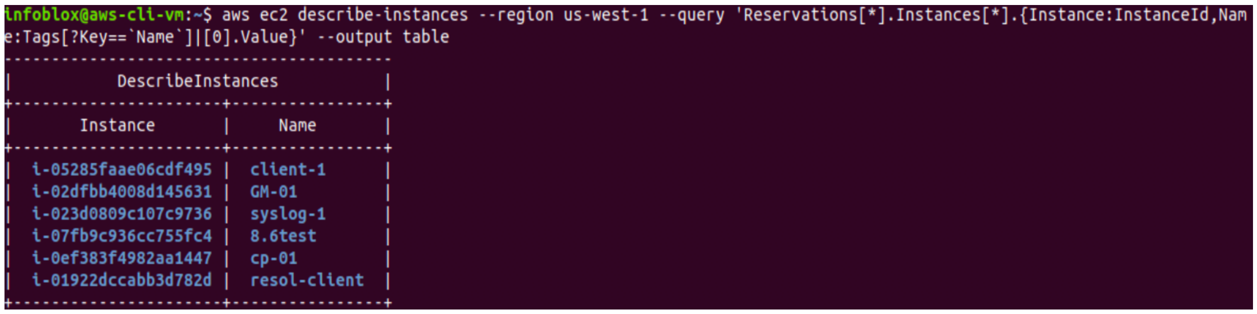
```
infoblox@aws-cli-vm:~$ aws ec2 describe-instances --region us-west-1
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "",
                    "StateReason": {
                        "Message": "Client.UserInitiatedShutdown: User initiated shutdown",
                        "Code": "Client.UserInitiatedShutdown"
                    },
                    "State": {
                        "Code": 80,
                        "Name": "stopped"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2020-08-11T21:10:22.000Z",
                    "PrivateIpAddress": "172.23.1.10",
                    "ProductCodes": [],
                    "VpcId": "vpc-0751455b251b46f3f",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 1
                    },
                    "StateTransitionReason": "User initiated (2020-08-11 21:11:26 GMT)",
                    "InstanceId": "i-05285faae06cdf495",
                    "EnaSupport": true,
                    "ImageId": "ami-04e59c05167ea7bd5",
                    "PrivateDnsName": "",
                    "KeyName": "demo-keys",
                    "SecurityGroups": [
```

The screenshot above shows the first lines of the detailed information returned for the first VM instance in this region. To return information on only specific VM instances, add the **--instance-ids** parameter filled by the IDs of instances. For example: **aws ec2 describe-instance --instance-ids i-1111111111111111**.



```
infoblox@aws-cli-vm:~$ aws ec2 describe-instances --instance-ids i-0062a93f5eecb2aa6
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
```

To return only certain attributes for each of the instances being described, use the **--query** parameter. For example: **aws ec2 describe-instance --region us-west-1 --query ‘Reservations[*].Instances[*].{Instance:InstanceId,Name:Tags[?Key==`Name`]|[0].Value}’ --output table**. This example also uses the **table** value for the **--output** parameter to format the returned information.

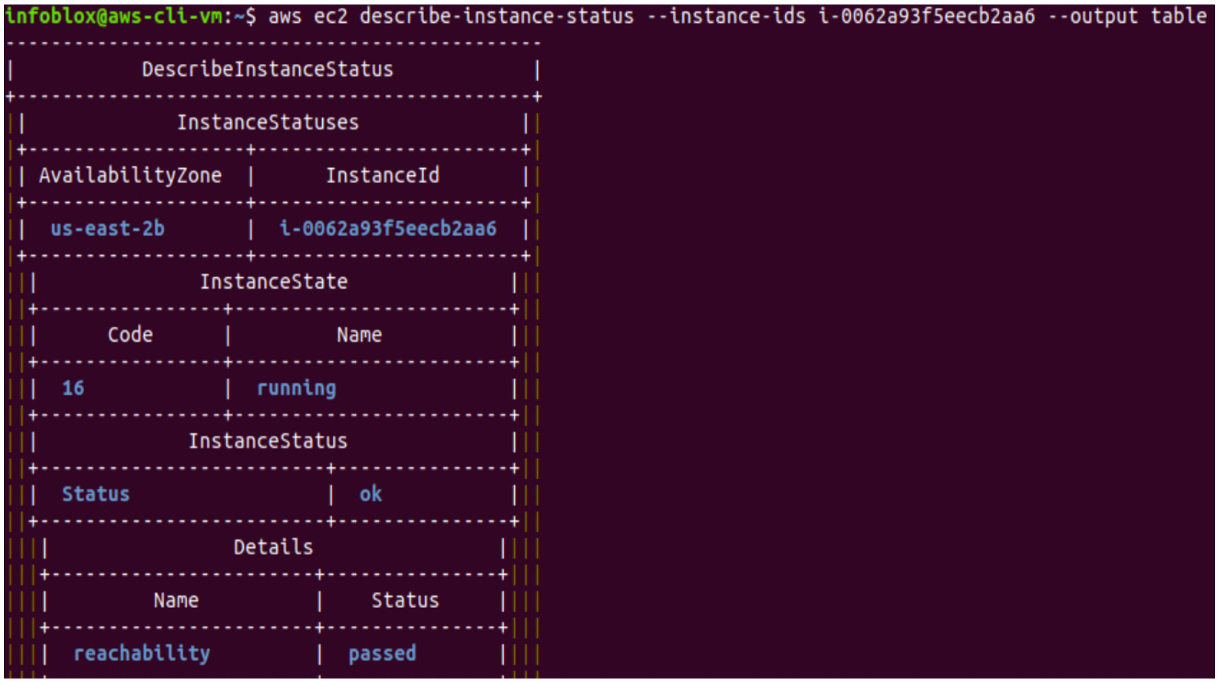


```
infoblox@aws-cli-vm:~$ aws ec2 describe-instances --region us-west-1 --query 'Reservations[*].Instances[*].{Instance:InstanceId,Nam
e:Tags[?Key==`Name`]|[0].Value}' --output table
-----------------------------------------
|           DescribeInstances           |
+----------------------+----------------+
|       Instance       |      Name      |
+----------------------+----------------+
|  i-05285faae06cdf495 |  client-1      |
|  i-02dfbb4008d145631 |  GM-01         |
|  i-023d0809c107c9736 |  syslog-1      |
|  i-07fb9c936cc755fc4 |  8.6test       |
|  i-0ef383f4982aa1447 |  cp-01         |
|  i-01922dccabb3d782d |  resol-client  |
+----------------------+----------------+
```

For further information and examples of using the --query parameter, refer to the documentation for describe-instances: https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html.

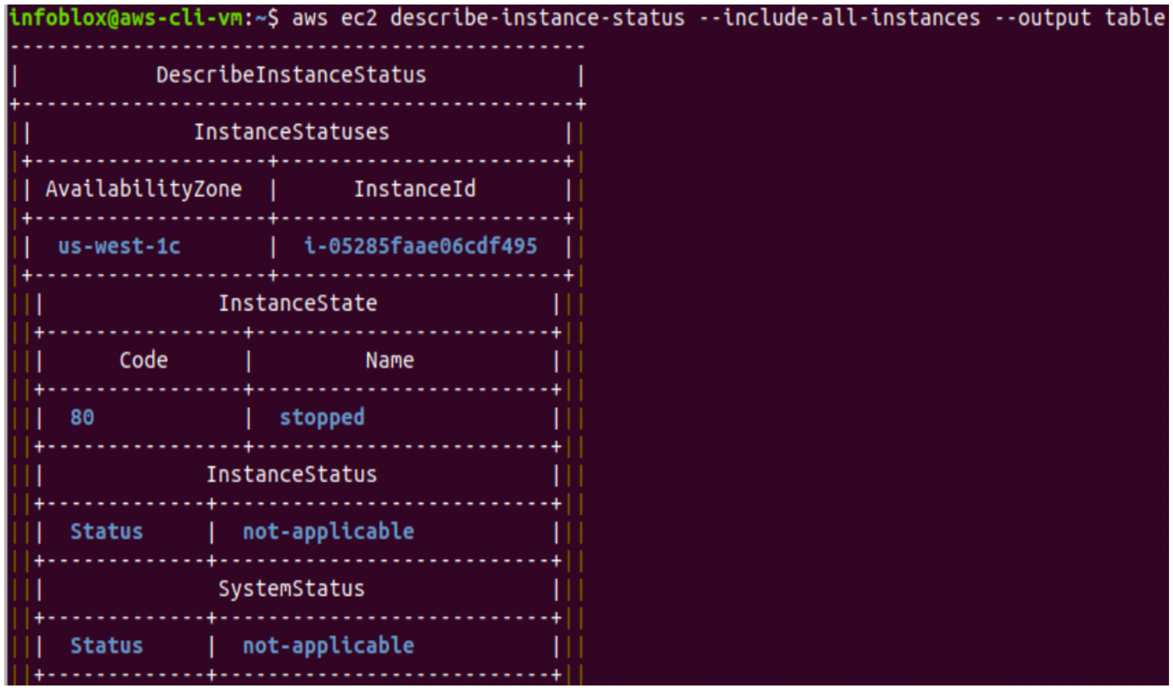
## Virtual Machine Status

To find the status of your virtual machine instances, use the command: **aws ec2 describe-instance-status**. By default, this command only describes running instances and performs status checks on them to identify hardware or software issues. To get the status for only specific instances, use the **--instance-ids** parameter: **aws ec2 describe-instance-status --instance-ids <instance_id>**.



```
infoblox@aws-cli-vm:~$ aws ec2 describe-instance-status --instance-ids i-0062a93f5eecb2aa6 --output table
----------------------------------------------
|           DescribeInstanceStatus           |
+--------------------------------------------+
||             InstanceStatuses             ||
|+-------------------+----------------------+|
|| AvailabilityZone  |      InstanceId      ||
|+-------------------+----------------------+|
||  us-east-2b       |  i-0062a93f5eecb2aa6 ||
|+-------------------+----------------------+|
|||              InstanceState             |||
||+---------------+------------------------+||
|||     Code      |          Name          |||
||+---------------+------------------------+||
|||  16           |  running               |||
||+---------------+------------------------+||
|||             InstanceStatus             |||
||+--------------------------+-------------+||
|||  Status                  |  ok         |||
||+--------------------------+-------------+||
||||               Details                ||||
|||+----------------------+---------------+|||
||||         Name         |    Status     ||||
|||+----------------------+---------------+|||
||||  reachability        |  passed       ||||
```

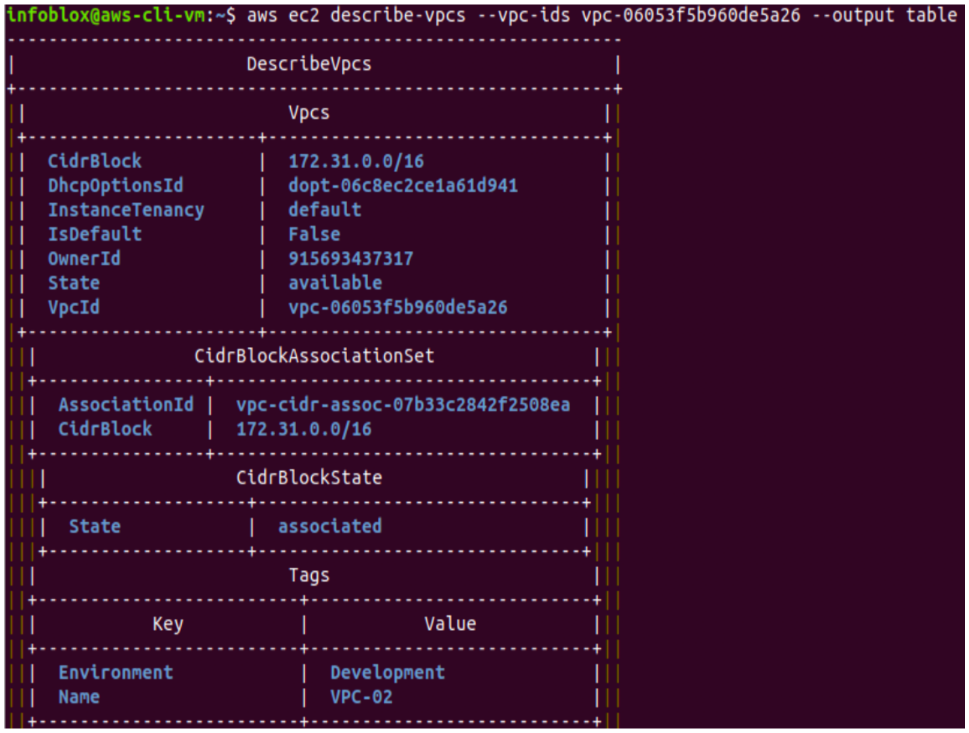
To see the status of all instances, including stopped instances, add the **--include-all-instances** parameter.



```
infoblox@aws-cli-vm:~$ aws ec2 describe-instance-status --include-all-instances --output table
----------------------------------------------
|           DescribeInstanceStatus           |
+--------------------------------------------+
||             InstanceStatuses             ||
|+-------------------+----------------------+|
|| AvailabilityZone  |      InstanceId      ||
|+-------------------+----------------------+|
||  us-west-1c       |  i-05285faae06cdf495 ||
|+-------------------+----------------------+|
|||              InstanceState             |||
||+---------------+------------------------+||
|||     Code      |          Name          |||
||+---------------+------------------------+||
|||  80           |  stopped               |||
||+---------------+------------------------+||
|||             InstanceStatus             |||
||+------------+---------------------------+||
|||  Status    |  not-applicable           |||
||+------------+---------------------------+||
|||              SystemStatus              |||
||+------------+---------------------------+||
|||  Status    |  not-applicable           |||
||+------------+---------------------------+||
```

## VPC Details

To gather details on all of your VPCs, such as CIDR block, DHCP options set, and more, use the command: **aws ec2 describe-vpcs**. To find information for a specific VPC, add the **--vpc-ids** parameter as shown below.



```
infoblox@aws-cli-vm:~$ aws ec2 describe-vpcs --vpc-ids vpc-06053f5b960de5a26 --output table
-----------------------------------------------------------
|                       DescribeVpcs                      |
+---------------------------------------------------------+
||                          Vpcs                         ||
|+----------------------+--------------------------------+|
||  CidrBlock           |  172.31.0.0/16                 ||
||  DhcpOptionsId       |  dopt-06c8ec2ce1a61d941        ||
||  InstanceTenancy     |  default                       ||
||  IsDefault           |  False                         ||
||  OwnerId             |  915693437317                  ||
||  State               |  available                     ||
||  VpcId               |  vpc-06053f5b960de5a26         ||
|+----------------------+--------------------------------+|
|||               CidrBlockAssociationSet               |||
||+---------------+-------------------------------------+||
|||  AssociationId |  vpc-cidr-assoc-07b33c2842f2508ea  |||
|||  CidrBlock     |  172.31.0.0/16                     |||
||+---------------+-------------------------------------+||
||||                   CidrBlockState                  ||||
|||+-------------------+-------------------------------+|||
||||  State            |  associated                   ||||
|||+-------------------+-------------------------------+|||
|||                        Tags                         |||
||+--------------------------+--------------------------+||
|||           Key            |          Value           |||
||+--------------------------+--------------------------+||
|||  Environment             |  Development             |||
|||  Name                    |  VPC-02                  |||
||+--------------------------+--------------------------+||
```

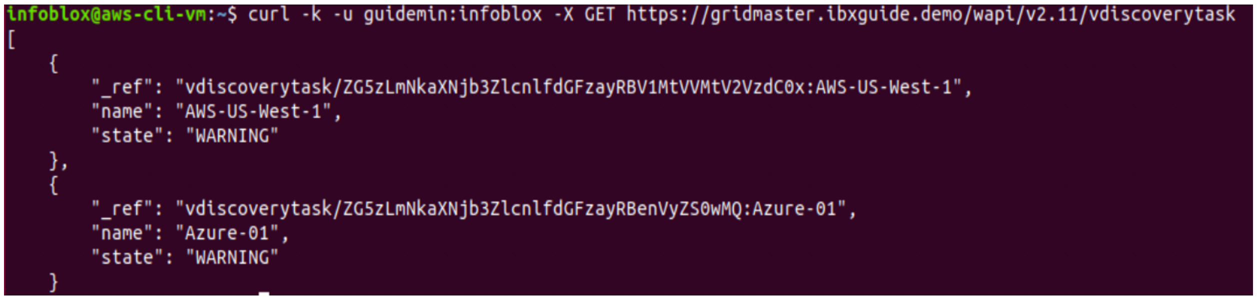
## Infoblox API Example

Infoblox provides a WAPI Reference Guide which provides information regarding available commands and their structure. One of the first steps when running commands is to first obtain the reference ID for the object that you want to manage. In this example, we will use the curl command to manage vDiscovery as this is a commonly sought after feature.

### vDiscovery

To get the reference ID for any vDiscovery tasks that you have configured, use the following example: c**url -k -u username:password -X GET https://<Grid_Master>/wapi/v2.11/vdiscoverytask**.

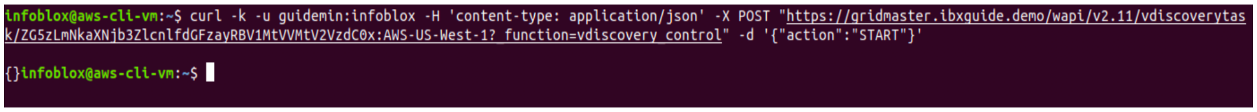
*Note: This command example is using WAPI version 2.11. Adjust this for your version of NIOS as required. Refer to Infoblox WAPI documentation for further information.*



```
infoblox@aws-cli-vm:~$ curl -k -u guidemin:infoblox -X GET https://gridmaster.ibxguide.demo/wapi/v2.11/vdiscoverytask
[
    {
        "_ref": "vdiscoverytask/ZG5zLmNkaXNjb3ZlcnlfdGFzayRBV1MtVVMtV2VzdC0x:AWS-US-West-1",
        "name": "AWS-US-West-1",
        "state": "WARNING"
    },
    {
        "_ref": "vdiscoverytask/ZG5zLmNkaXNjb3ZlcnlfdGFzayRBenVyZS0wMQ:Azure-01",
        "name": "Azure-01",
        "state": "WARNING"
    }
```

In the above example, the reference ID for one of the vDiscovery tasks is:
**vdiscoverytask/ZG5zLmNkaXNjb3ZlcnlfdGFzayRBV1MtVVMtV2VzdC0x:AWS-US-West-1**

Using this reference ID, we can build a command to start this vDiscovery task (this command should be a single line): **curl -k -u username:password -H ‘content-type: application/json’ -X POST “https://<Grid_Master>/wapi/v2.11/vdiscoverytask/<task_ID>:<Task_Name>?_function=vdiscovery_control” -d ‘{“action”:”START”}’**



```
infoblox@aws-cli-vm:~$ curl -k -u guidemin:infoblox -H 'content-type: application/json' -X POST "https://gridmaster.ibxguide.demo/wapi/v2.11/vdiscoverytask/ZG5zLmNkaXNjb3ZlcnlfdGFzayRBV1MtVVMtV2VzdC0x:AWS-US-West-1?_function=vdiscovery_control" -d '{"action":"START"}'

{}infoblox@aws-cli-vm:~$ 
```

## Additional Resources

- AWS Management Console: https://console.aws.amazon.com/console/
- AWS CLI and AWS Shell main page: https://aws.amazon.com/cli/
- AWS CLIv2 Command Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html
- AWS Tools for PowerShell: https://aws.amazon.com/powershell/
- Infoblox REST API Reference: https://www.infoblox.com/wp-content/uploads/infoblox-deployment-infoblox-rest-api.pdf
- Infoblox WAPI and Other Documentation: https://docs.infoblox.com/

Infoblox unites networking and security to deliver unmatched performance and protection. Trusted by Fortune 100 companies and emerging innovators, we provide real-time visibility and control over who and what connects to your network, so your organization runs faster and stops threats earlier.

Corporate Headquarters
2390 Mission College Blvd, Ste. 501
Santa Clara, CA 95054
+1.408.986.4000
www.infoblox.com