ALTEON ANSIBLE AUTOMATION DEMO LAB

Owner: Chen Sagi, Advanced Solutions

Version: 10.1

February 1, 2023

TABLE OF CONTENTS

LAB OVERVIEW

Purpose

The purpose of this document is to describe and demonstrate how to use the Alteon Ansible automation demo environment.

Please note that the playbooks used in this demo are just an example and customers can write their playbooks according to their needs.

The document covers the following scenarios:

1. **Installing Radware Ansible modules**

   In this scenario, we will demonstrate the steps needed to install Radware Ansible modules.

2. **Running a playbook**

   In this scenario, we will run our first playbook.

   This playbook will configure the l3 interface and SLB in the Alteon.

3. **Changes in Playbooks**

   In this scenario, we will change our playbook and see how it affects our Alteon configuration.

4. **Error Handling**

   In this scenario, we will see how errors from Alteon shown in the Ansible modules.

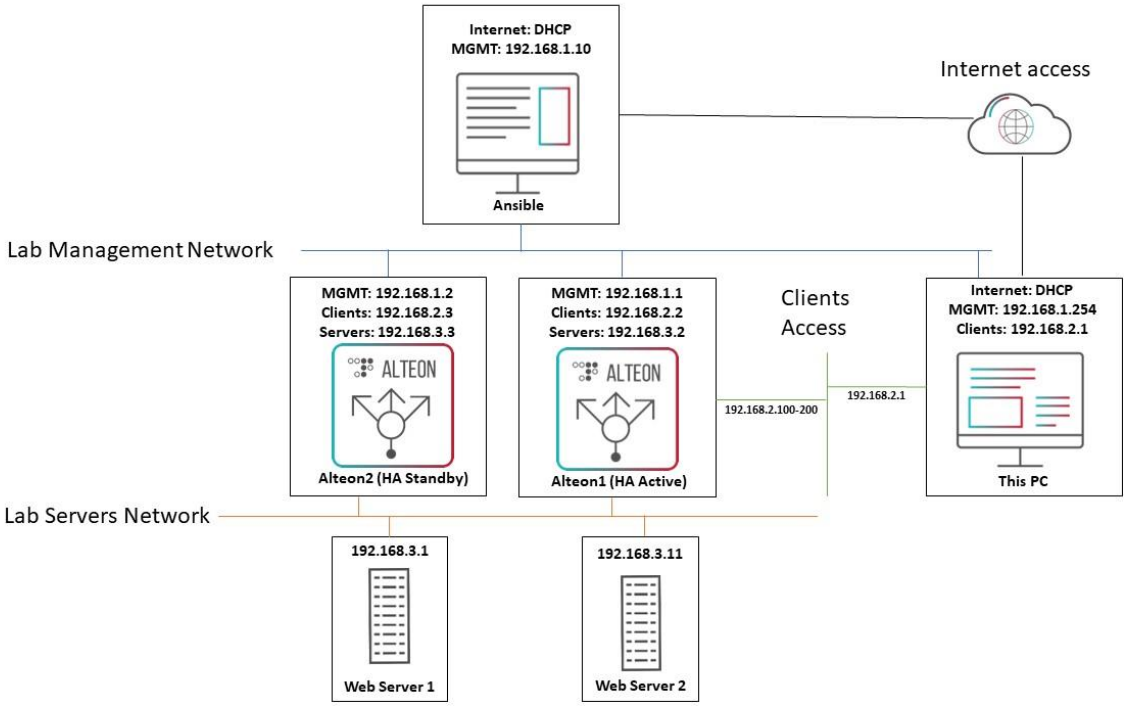
5. **Delete the Configuration**

   In this scenario, we will clean up the configuration from our playbook.

6. **Creating Multiple Services**

   In this scenario, we will create multiple services in one playbook using a variable file.

Lab Topology



Lab Environment Credentials

Device	*User*	*Password*
Management station	radware	radware
Alteon	admin	Radware1!
Ansible	radware	radware
Web Server	radware	radware

Management (MGMT) Station

This demonstration is performed from the management station, which includes:

- SSH Access to Ansible Server with MobaXterm. To access the MobaXterm click the icon in the desktop.
- SSH Access to Alteon with MobaXterm.
- Web Access to the Alteon with Google Chrome.
- Web Access to environment devices

The management station includes three network interfaces: one interface connects to the internet, the second connects to the lab data segment, and the third interface connects to the management segment.

OVERVIEW

Ansible Introduction

What Is Ansible?

Ansible is a powerful IT automation tool that can be used to configure systems, deploy software, and automate complex IT tasks.
It is written in Python and uses a simple, human-readable language called YAML to define automation tasks, called "playbooks."
These playbooks are then executed by the Ansible automation engine, which communicates with the target systems over SSH.
With Ansible, you can automate repetitive tasks, quickly deploy applications and infrastructure, and gain greater control over your IT environment.
Ansible also features a large and growing collection of pre-built modules that can be used to perform common tasks, such as installing software or creating users.
Additionally, Ansible's simple architecture means that it can be easily extended to work with custom modules and plugins, making it highly adaptable to a wide range of use cases.

Why Ansible?

- Simple: Ansible is easy to use and understand with simple commands written in YAML language.
- Agentless: Ansible doesn't require any additional software on the target systems, so it's easy to use on different types of systems.
- Declarative: Ansible defines the final state of the system, and it finds the best way to reach it. This makes it easy to specify complex tasks and handle errors automatically.
- Flexibility: Ansible has a wide collection of pre-built modules to perform common tasks, and it's also possible to write custom modules to handle specific needs.
- Scalability: Ansible can handle automation tasks for thousands of devices and it's easy to scale up or down according to organization's needs.

- Community and Support: Ansible has a big community with many resources like documentation, tutorials, and examples that you can use. It's also supported by Red Hat that provides enterprise level support and training.

Ansible Modules

The units of code Ansible executes. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device. You can invoke a single module with a task, or invoke several different modules in a playbook.

Most of the modules are "idempotent" - means that they only do something when a change is required

Tasks

Blocks that define a single procedure to be executed, e.g. Install a package.

Each task uses an Ansible module.

Ansible Playbooks

Ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share, and understand.

YAML

YAML (stands for YAML Ain't Markup Language) is a serialization language that has steadily increased in popularity over the last few years. It's often used as a format for configuration files, but its object serialization abilities make it a viable replacement for languages like JSON.

For a YAML example please refer to Appendix 1 – YAML Example.

Radware Modules

Radware Ansible modules are in ***Ansible Galaxy*** hub under Radware's namespace. Our Ansible Collection is called ***radware_ modules***.

Ansible Galaxy

Ansible Galaxy is Ansible's official hub for sharing Ansible content.

It's the source of community and vendor-provided Ansible roles and modules to help you get started faster.

radware_ modules

'radware_modules' is an ansible collection for managing and automating your Radware Alteon.

It consists of a set of modules for performing tasks related to Radware Alteon configuration.

Requirements

- Ansible >= 2.9
- Python >= 3.6
- alteon-sdk python package

Visit the Galaxy Page

You can get the latest release from the Ansible Galaxy page.

Read the Read Me and go the content tab to see all the modules available.

ANSIBLE DEMO LAB SCENARIOS

Radware Modules Installation

In this scenario, we will demonstrate the steps needed to install Radware Ansible modules.

1. Use MobaXterm to login to the Ansible server.
2. Install the alteon-sdk python package using pip command.

   ```
   pip3 install alteon-sdk
   ```

3. Install Radware modules from Galaxy.

   ```
   ansible-galaxy collection install radware.radware_modules
   ```

4. Verify that the collection is installed using the ansible-doc command.

   ```
   ansible-doc -t module radware.radware_modules.alteon_config_server
   ```

   You should see the module documentation.

5. Browse the Ansible Galaxy page and click on '***Content'*** to show the list of modules.

Using Playbooks to Configure the Alteon

Run you first playbook

This playbook configuring the I3 interface and SLB (2 real servers, group, virt, and HTTP service) in the Alteon.

Note:

Refer to Appendix 2 – demo.yaml to view the playbook.

1. Use the bookmark in Chrome to login to the Alteon (admin:radware) and show that there is nothing configured under these tables:

   Network > Layer 3 > IP Interfaces

   Application Delivery > Server Resources > Real Servers

   Application Delivery > Server Resources > Server Groups

   Application Delivery > Virtual Services

2. Login to the Ansible server and go to */opt/demolab/basic-demo* folder.

   ```
   cd /opt/demolab/basic-demo
   ```

3. Open the playbook file with your favorite editor and go over it to show how it works.

   ```
   vim demo.yaml
   ```

   Note: The playbook uses 2 external variables:

   - *"radware_provider"* located in vars/alteon_params.yml file. This variable contains the Alteon connection details.
   - *"state"* which we will enter at the playbook run command. This variable contains the Ansible state which we will use to create/delete Alteon object.

4. Close the demo.yaml file (press esc then write ":q" and press enter)

5. Open vars/alteon_params.yml go over it to show *radware_provider* variable which contains the Alteon connection details

```
vim vars/alteon params.yml
```

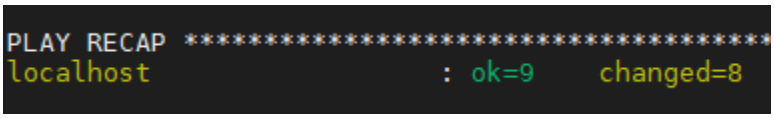
Note:

Refer to Appendix 3 – alteon_params.yml to view the file.

6. Close alteon_params file.
7. Run the playbook using ansible-playbook command

```
ansible-playbook demo.yaml -e "state=present"
```

8. Verify the output of the command is:



9. Go to Alteon-01 and show the configuration was created under:

Network > Layer 3 > IP Interfaces

Application Delivery > Server Resources > Real Servers

Application Delivery > Server Resources > Server Groups

Application Delivery > Virtual Services

10. Go to chrome and browse to the VIP address to show that the service we created works.

You can use the bookmark "Demo App" to reach the application we now deployed.

Using Playbooks to Modify Alteon Configuration

The following scenario will show how to change the real server IP address.

1. Go to the Alteon and check the real server IP address.

   Application Delivery > Server Resources > Real Servers

2. Login to the Ansible server and go to */opt/demolab/basic-demo* folder.

   ```
   cd /opt/demolab/basic-demo
   ```

3. Open the playbook file with your favorite editor.

   ```
   vim demo.yaml
   ```

4. Change the server IP address (Edit the Yaml by pressing 'I', save by pressing the esc button, write ':wq' and press enter)

   ```
   - name: real1 config
     radware.radware_modules.alteon_config_server:
       provider: "{{ radware_provider }}"
       state: "{{ state }}"
       parameters:
         index: real1
         state: enabled
         ip_address: 192.168.3.12
   ```

5. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook demo.yaml -e "state=present"
   ```

6. Verify the output of the command is:

   ```
   PLAY RECAP *********************************************
   localhost                  : ok=9    changed=2
   ```

7. Go to the Alteon and show that the real server IP address changed.

   Application Delivery > Server Resources > Real Servers

Error Handling

The following scenario will show error messages when using incorrect input in the playbook.

1. Login to the Ansible server and go to */opt/demolab/basic-demo* folder.

```
cd /opt/demolab/basic-demo
```

2. Open the playbook file with your favorite editor.

```
vim demo.yaml
```

3. Change the real server IP to invalid IP address.

```
- name: real config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: real1
      state: enabled
      ip_address: 12.3..4.56
```

4. Run the playbook using ansible-playbook command.

```
ansible-playbook demo.yaml -e "state=present"
```

5. Show the error message to understand what was wrong.

Using Playbooks to Delete Alteon Configuration

The following scenario will delete the configuration from the Alteon.

1. Login to the Ansible server and go to */opt/demolab/basic-demo* folder.

   ```
   cd /opt/demolab/basic-demo
   ```

2. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook demo.yaml -e "state=absent"
   ```

3. Go to the Alteon and show the configuration under these tables:

   Network > Layer 3 > IP Interfaces

   Application Delivery > Server Resources > Real Servers

   Application Delivery > Server Resources > Server Groups

   Application Delivery > Virtual Services

Creating Multiple Services

The following scenario will create multiple services in one playbook.

1. Login to the Ansible server and go to */opt/demolab/long-demo* folder.

   ```
   cd /opt/demolab/long-demo
   ```

2. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook long_demo.yaml -e "state=present"
   ```

   Note:

   Refer to Appendix 4 – long_demo.yaml to view the file.

3. Go to the Alteon and show the configuration under these tables:

   Network > Layer 3 > IP Interfaces

   Application Delivery > Server Resources > Real Servers

   Application Delivery > Server Resources > Server Groups

   Application Delivery > Virtual Services

4. Cleanup the environment for next scenario by running

   ```
   ansible-playbook long_demo.yaml -e "state=absent"
   ```

Configure Alteon High Availability

The following scenario will create HA configuration for both Alteons in one playbook.

1. Login to the Ansible server and go to */opt/demolab/ha-demo* folder.

   ```
   cd /opt/demolab/ha-demo
   ```

2. Open the playbook file with your favorite editor and go over it to show how it works.

   ```
   vim ha-demo.yaml
   ```

3. Close the demo.yaml file

4. Open vars/alteon_params.yml go over it to show *radware_provider* variable which contains both Alteon connection details as a list,

   ```
   vim vars/alteon_params.yml
   ```

   Note:

   Refer to Appendix 6- alteon_params.yml (HA Demo) to view the file.

5. Close alteon_params file.

6. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook ha-demo.yaml -e "state=present"
   ```

   Note:

   Refer to Appendix 5 – ha-demo.yaml to view the file.

7. Go to the Alteon-01 and Alteon-02 and show the configuration under these tables:

   Network > Layer 3 > IP Interfaces

   Network > Layer 3 > High Availability

   Network > Layer 3 > High Availability > Configuration Sync

8. Cleanup the environment for next scenario by running

   ```
   ansible-playbook ha-demo.yaml -e "state=absent"
   ```

Configure Alteon SSL Offload

The following scenario will create service with SSL offload configured in one playbook.

1. Login to the Ansible server and go to */opt/demolab/ssl-demo* folder.

   ```
   cd /opt/demolab/ssl-demo
   ```

2. Open the playbook file with your favorite editor and go over it to show how it works.

   ```
   vim ssl-demo.yaml
   ```

3. Close the demo.yaml file

4. Open vars/alteon_params.yml go over it to show *radware_provider* variable which contains both Alteon connection details as a list,

   ```
   vim vars/alteon_params.yml
   ```

   Note:

   Refer to Appendix 3 – alteon_params.yml to view the file.

5. Close alteon_params file.

6. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook ssl-demo.yaml -e "state=present"
   ```

   Note:

   Refer to Appendix 7 – ssl-demo.yaml to view the file.

11. Go to Alteon-01 and show the configuration was created under:

    Network > Layer 3 > IP Interfaces

    Application Delivery > Server Resources > Real Servers

    Application Delivery > Server Resources > Server Groups

    Application Delivery > Virtual Services

12. Go to chrome and browse to the VIP address with HTTPS to show that the service we created works.

    You can use the bookmark “Secure Demo App” to reach the application we now deployed.

7. Cleanup the environment for next scenario by running

```
ansible-playbook ssl-demo.yaml -e "state=absent"
```

Configure Alteon Global Load Balancer (Advanced Ansible)

The following scenario will create GSLB on both Alteon ADC devices.

1. Login to the Ansible server and go to */opt/demolab/gslb-demo* folder.

   ```
   cd /opt/demolab/gslb-demo
   ```

2. Open the playbook file with your favorite editor and go over it to show how it works.

   ```
   vim gslb-demo.yaml
   ```

   Note:

   Some of the tasks are using the 'radware.radware_modules.alteon_config_alteon_cli_command' module, as there are some configurations that aren't supported by radware modules, this can be extended to any configuration and provide ansible solution to any use case.

3. Close the gslb-demo.yaml file

4. Open vars/alteon_params.yml go over it to show *radware_provider* variable which contains both Alteon connection details as a list, and the necessary variables to configure GSLB.

   ```
   vim vars/alteon params.yml
   ```

   Note:

   Refer to Appendix 9 – alteon_params.yml (GSLB Demo) to view the file.

5. Close alteon_params file.

6. Run the playbook using ansible-playbook command.

   ```
   ansible-playbook gslb-demo.yaml -e "state=present"
   ```

   Note:

   Refer to Appendix 8 – gslb-demo.yaml to view the file.

7. Go to the Alteon and show the configuration under these tables:

   Network > Layer 3 > IP Interfaces

Application Delivery > Virtual Services

Application Delivery > Server Resources > Real Servers

Application Delivery > Server Resources > Server Groups

Application Delivery > Global Traffic Redirection

Application Delivery > Global Traffic Redirection > Remote Sites

Application Delivery > Global Traffic Redirection > DNS Redirection Rules

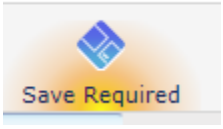
8. Go to the web and try out the gslb service we just created by navigating to http://app.demo.lab/ or choose the '**GSLB Demo App'** bookmark
9. **Save** the configuration and **reboot** alteon-01

CLI: Go to MobaXterm and log in to 

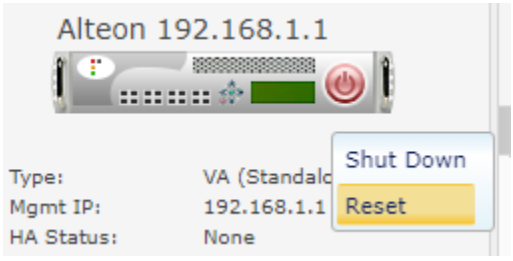
>> Main# **save**

>> Main# **/boot/reset**

UI: Navigate with Chrome using the Bookmark Alteon-01



Click on the Power Button Icon and Choose Reset:



10. Navigate to the Demo app and show that you are getting 'Server2' with Address 192.168.3.11:80 which is set to be in Alteon-02 environment.
11. Wait for Alteon-01 to be online again to continue to the next step.
12. (Optional) Do the same thing with Alteon-02 and show that you are getting 'Server' with Address 192.168.3.1:80, Wait for Alteon-02 to be online again to continue to the next step.
13. Edit the script variables under vars/alteon_params.yml so that *'fqdn'* is set to be yournameapp.demo.lab

```
vim vars/alteon_params.yml
```

14. Run the playbook using ansible-playbook command.

```
ansible-playbook gslb-demo.yaml -e "state=present"
```

15. Navigate to http://yournameapp.demo.lab/ and see that the site name has changed successfully
16. Cleanup the environment for next scenario by running:

```
ansible-playbook gslb-demo.yaml -e "state=absent"
```

APPENDIX

Appendix 1 – YAML Example

YAML Example

```
---
  doe: "a deer, a female deer"
  ray: "a drop of golden sun"
  pi: 3.14159
  xmas: true
  french-hens: 3
  calling-birds:
    - huey
    - dewey
    - louie
    - fred
  xmas-fifth-day:
    calling-birds: four
    french-hens: 3
    golden-rings: 5
    partridges:
      count: 1
      location: "a pear tree"
    turtle-doves: two
```

Appendix 2 – demo.yaml

```
- hosts: localhost
  vars_files:
    - vars/alteon_params.yml
  tasks:
    - name: interface 1 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          ip4_address: 192.168.3.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 1

    - name: interface 2 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 2
          state: enabled
          ip4_address: 192.168.2.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 2
```

```
- name: real1 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: real1
      state: enabled
      ip_address: 192.168.3.1

- name: real2 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: real2
      state: enabled
      ip_address: 192.168.3.11

- name: group config
  radware.radware_modules.alteon_config_server_group:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    revert_on_error: true
    parameters:
      index: group_test
      slb_metric: roundRobin
      health_check_id: tcp
      server_names:
        - real1
        - real2
```

```
- name: virt config
  radware.radware_modules.alteon_config_virtual_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: virt_test
      state: enabled
      ip_address: 192.168.2.100

- name: service config
  radware.radware_modules.alteon_config_virtual_service:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: virt_test
      service_index: 1
      service_port: 80
      server_group_name: group_test
      application_type: http
      nat_mode: address
      nat_address: 192.168.3.100

- name: apply configuration
  radware.radware_modules.alteon_mng_config:
    provider: "{{ radware_provider }}"
    command: apply
```

Appendix 3 – alteon_params.yml (used for basic, long and SSL demos)

```
radware_provider:
  server: 192.168.1.1
  user: admin
  password: admin
  validate_certs: no
  https_port: 443
  ssh_port: 22
  timeout: 10

services:
 - ip: 192.168.2.100
   nat: 192.168.3.100
   port: 80
 - ip: 192.168.2.101
   nat: 192.168.3.101
   port: 8080
 - ip: 192.168.2.102
   nat: 192.168.3.102
   port: 8081
 - ip: 192.168.2.103
   nat: 192.168.3.103
   port: 8082
 - ip: 192.168.2.104
   nat: 192.168.3.104
   port: 8083
state: present
```

```
- ip: 192.168.2.105
   nat: 192.168.3.105
   port: 8084
 - ip: 192.168.2.106
   nat: 192.168.3.106
   port: 8085
```

Appendix 4 – long_demo.yaml

```
- hosts: localhost
  vars_files:
    - vars/alteon_params.yml
  tasks:
    - name: interface 1 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          ip4_address: 192.168.3.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 1

    - name: interface 2 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 2
          state: enabled
          ip4_address: 192.168.2.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 2
```

```
- name: real1 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: "real1_{{ loop_index + 1 }}"
      state: enabled
      ip_address: 192.168.3.1
  loop: "{{ services }}"
  loop_control:
    index_var: loop_index

- name: real2 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: "real2_{{ loop_index + 1 }}"
      state: enabled
      ip_address: 192.168.3.11
  loop: "{{ services }}"
  loop_control:
    index_var: loop_index
```

```
- name: group config
  radware.radware_modules.alteon_config_server_group:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    revert_on_error: true
    parameters:
      index: "group_test_{{ loop_index + 1 }}"
      slb_metric: roundRobin
      health_check_id: tcp
      server_names:
        - "real1_{{ loop_index + 1 }}"
        - "real2_{{ loop_index + 1 }}"
  loop: "{{ services }}"
  loop_control:
    index_var: loop_index

- name: virt config
  radware.radware_modules.alteon_config_virtual_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: "virt_test_{{ loop_index + 1 }}"
      state: enabled
      ip_address: "{{ item.ip }}"
  loop: "{{ services }}"
  loop_control:
    index_var: loop_index
```

```
- name: service config
  radware.radware_modules.alteon_config_virtual_service:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: "virt_test_{{ loop_index + 1 }}"
      service_index: 1
      service_port: "{{ item.port }}"
      server_group_name: "group_test_{{ loop_index + 1 }}"
      application_type: http
      nat_mode: address
      nat_address: "{{ item.nat }}"
  loop: "{{ services }}"
  loop_control:
    index_var: loop_index

- name: apply configuration
  radware.radware_modules.alteon_mng_config:
    provider: "{{ radware_provider }}"
    command: apply
```

Appendix 5 – ha-demo.yaml

```
- hosts: localhost
  vars_files:
    - vars/alteon_params.yml
  tasks:
    - name: active alteon interface 1 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider[0] }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          ip4_address: "{{ ha.peer_interface_va_1 }}"
          peer_ip: "{{ ha.peer_interface_va_2 }}"
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 1
   - name: backup alteon interface 1 config
         radware.radware_modules.alteon_config_l3_interface:
           provider: "{{ radware_provider[1] }}"
           state: "{{ state }}"
           parameters:
             index: 1
             state: enabled
             ip4_address: "{{ ha.peer_interface_va_2 }}"
             peer_ip: "{{ ha.peer_interface_va_1 }}"
             ip4_subnet: 255.255.255.0
             ip_ver: ipv4
```

```
- name: configure active ha
      radware.radware_modules.alteon_config_high_availability:
        provider: "{{ radware_provider[0] }}"
        state: "{{ state }}"
        parameters:
          mode: switch
          fail_back_mode: always
          preferred_state: active
          advertising_interfaces:
          - 1
          tracked_interfaces:
          - 1

    - name: configure standby ha
      radware.radware_modules.alteon_config_high_availability:
        provider: "{{ radware_provider[1] }}"
        state: "{{ state }}"
        parameters:
          mode: switch
          fail_back_mode: always
          preferred_state: standby
          advertising_interfaces:
          - 1
          tracked_interfaces:
          - 1
```

```
- name: active config sync command
      radware.radware_modules.alteon_config_ha_config_sync:
        provider: "{{ radware_provider[0] }}"
        state: "{{ state }}"
        parameters:
          automatic_sync: disabled
          filter_sync: enabled
          ip_interface_sync: enabled
          gateway_sync: enabled
          static_route_sync: enabled
          certificate_sync: enabled
          certificate_passphrase: radware
          peer_authentication_mode: passphrase
          authentication_passphrase: radware
          sync_peers:
            - state: enable
              ip_ver: ipv4
              ip4_address: "{{ ha.peer_interface_va_2 }}"

- name: backup config sync command
      radware.radware_modules.alteon_config_ha_config_sync:
        provider: "{{ radware_provider[1] }}"
        state: "{{ state }}"
        parameters:
          automatic_sync: enabled
          filter_sync: enabled
          ip_interface_sync: enabled
          gateway_sync: enabled
          static_route_sync: enabled
```

```
        certificate_sync: enabled
         certificate_passphrase: radware
         peer_authentication_mode: passphrase
         authentication_passphrase: radware
         sync_peers:
           - state: enable
             ip_ver: ipv4
             ip4_address: "{{ ha.peer_interface_va_1 }}"

  - name: apply configuration
    radware.radware_modules.alteon_mng_config:
      provider: "{{ item }}"
      command: apply
    loop: "{{ radware_provider }}"
```

Appendix 6 – alteon_params.yml (HA Demo)

```
radware_provider:
  - server: 192.168.1.1
    user: admin
    password: radware
    validate_certs: no
    https_port: 443
    ssh_port: 22
    timeout: 10
  - server: 192.168.1.2
    user: admin
    password: radware
    validate_certs: no
    https_port: 443
    ssh_port: 22
    timeout: 10
state: present

ha:
  peer_interface_va_1: 192.168.3.2 # ha ip interface for alteon 1 (ipv4)
  peer_interface_va_2: 192.168.3.3 # ha ip interface for alteon 2 (ipv4)

   port: 8083
```

Appendix 7 – ssl-demo.yaml

```
- hosts: localhost
  vars_files:
    - vars/alteon_params.yml
  tasks:
    - name: interface 1 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          ip4_address: 192.168.3.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 1

    - name: interface 2 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: 2
          state: enabled
          ip4_address: 192.168.2.2
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 2
```

```
- name: real1 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: real1
      state: enabled
      ip_address: 192.168.3.1

- name: real2 config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: real2
      state: enabled
      ip_address: 192.168.3.11

- name: group config
  radware.radware_modules.alteon_config_server_group:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    revert_on_error: true
    parameters:
      index: group_test
      slb_metric: roundRobin
      health_check_id: tcp
      server_names:
        - real1
        - real2
```

```
- name: virt config
      radware.radware_modules.alteon_config_virtual_server:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: virt_test
          state: enabled
          ip_address: 192.168.2.100

    - name: service config
      radware.radware_modules.alteon_config_virtual_service:
        provider: "{{ radware_provider }}"
        state: "{{ state }}"
        parameters:
          index: virt_test
          service_index: 1
          service_port: 80
          server_group_name: group_test
          application_type: http
          nat_mode: address
          nat_address: 192.168.3.100
```

```
- name: TLS offload service config
  radware.radware_modules.alteon_config_virtual_service:
    provider: "{{ radware_provider }}"
    state: "{{ state }}"
    parameters:
      index: virt_test
      service_index: 2
      service_port: 443
      server_port: 80
      server_group_name: group_test
      application_type: https
      ssl_policy_name: Outbound_FE_SSL_Inspection
      delayed_binding: forceproxy
      server_cert_name: WebManagementCert
      nat_mode: address
      nat_address: 192.168.3.100

- name: apply configuration
  radware.radware_modules.alteon_mng_config:
    provider: "{{ radware_provider }}"
    command: apply
```

Appendix 8 – gslb-demo.yaml

```
- hosts: localhost
  vars_files:
    - vars/alteon_params.yml
  tasks:
    - name: creating local variables
      set_fact:
        reals_site_a: |-
          [{% for srv in gslb_srv.site_a.local_servers %}
            "{{ srv.index }}",
          {% endfor %}]
        reals_site_b: |-
          [{% for srv in gslb_srv.site_b.local_servers %}
            "{{ srv.index }}",
          {% endfor %}]
    - name: site A interface 1 config
      radware.radware_modules.alteon_config_l3_interface:
        provider: "{{ radware_provider[0] }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          ip4_address: "{{ gslb_srv.site_a.srvr_int }}"
          ip4_subnet: 255.255.255.0
          ip_ver: ipv4
          vlan: 1
```

```
- name: site A interface 2 config
   radware.radware_modules.alteon_config_l3_interface:
     provider: "{{ radware_provider[0] }}"
     state: "{{ state }}"
     parameters:
       index: 2
       state: enabled
       ip4_address: "{{ gslb_srv.site_a.clnt_int }}"
       ip4_subnet: 255.255.255.0
       ip_ver: ipv4
       vlan: 2

 - name: site B interface 1 config
   radware.radware_modules.alteon_config_l3_interface:
     provider: "{{ radware_provider[1] }}"
     state: "{{ state }}"
     parameters:
       index: 1
       state: enabled
       ip4_address: "{{ gslb_srv.site_b.srvr_int }}"
       ip4_subnet: 255.255.255.0
       ip_ver: ipv4
       vlan: 1

- name: site B interface 2 config
   radware.radware_modules.alteon_config_l3_interface:
     provider: "{{ radware_provider[1] }}"
     state: "{{ state }}"
```

```
    parameters:
      index: 2
      state: enabled
      ip4_address: "{{ gslb_srv.site_b.clnt_int }}"
      ip4_subnet: 255.255.255.0
      ip_ver: ipv4
      vlan: 2

- name: site A reals config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider[0] }}"
    state: "{{ state }}"
    parameters:
      state: enabled
      index: "{{ item.index }}"
      ip_address: "{{ item.address }}"
  loop: "{{ gslb_srv.site_a.local_servers }}"

- name: site B reals config
  radware.radware_modules.alteon_config_server:
    provider: "{{ radware_provider[1] }}"
    state: "{{ state }}"
    parameters:
      state: enabled
      index: "{{ item.index }}"
      ip_address: "{{ item.address }}"
  loop: "{{ gslb_srv.site_b.local_servers }}"
```

```
- name: site A remote real
   radware.radware_modules.alteon_config_server:
     provider: "{{ radware_provider[0] }}"
     state: "{{ state }}"
     parameters:
       state: enabled
       index: "other-va-vip"
       health_check_id: "dssp"
       server_type: "remote_server"
       ip_address: "{{ gslb_srv.site_b.vip }}"

 - name: site B remote real
   radware.radware_modules.alteon_config_server:
     provider: "{{ radware_provider[1] }}"
     state: "{{ state }}"
     parameters:
       state: enabled
       index: "other-va-vip"
       health_check_id: "dssp"
       server_type: "remote_server"
       ip_address: "{{ gslb_srv.site_a.vip }}"
```

```
- name: site A group config
  radware.radware_modules.alteon_config_server_group:
    provider: "{{ radware_provider[0] }}"
    state: "{{ state }}"
    revert_on_error: true
    parameters:
      index: group_gslb
      slb_metric: roundRobin
      health_check_id: tcp
      server_names: "{{ reals_site_a + ['other-va-vip'] }}"

- name: site B group config
  radware.radware_modules.alteon_config_server_group:
    provider: "{{ radware_provider[1] }}"
    state: "{{ state }}"
    revert_on_error: true
    parameters:
      index: group_gslb
      slb_metric: roundRobin
      health_check_id: tcp
      server_names: "{{ reals_site_b + ['other-va-vip'] }}"
```

```
- name: site A dns responder
   radware.radware_modules.alteon_config_dns_responders:
     provider: "{{ radware_provider[0] }}"
     state: "{{ state }}"
     parameters:
       dns_responders:
       - name: responder_dns_1
         ip_ver: ipv4
         ip4_address: "{{ gslb_srv.site_a.dns_responder }}"
         return_to_src_mac: disable

 - name: site B dns responder
   radware.radware_modules.alteon_config_dns_responders:
     provider: "{{ radware_provider[1] }}"
     state: "{{ state }}"
     parameters:
       dns_responders:
       - name: responder_dns_1
         ip_ver: ipv4
         ip4_address: "{{ gslb_srv.site_b.dns_responder }}"
         return_to_src_mac: disable
```

```
    - name: turn on GSLB on both sites
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ item }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/ on"
      when: state == "present"
      loop: "{{ radware_provider }}"

    - name: turn off GSLB on both sites
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ item }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/ off"
      when: state == "absent"
      loop: "{{ radware_provider }}"

    - name: config GSLB remote site site_a -> site_b
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ radware_provider[0] }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/site 1/ prima {{
gslb_srv.site_b.clnt_int }}"
      when: state == "present"
```

```
   - name: config GSLB remote site site_b -> site_a
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ radware_provider[1] }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/site 1/ prima {{
gslb_srv.site_a.clnt_int }}"
      when: state == "present"

    - name: config GSLB remote site enabled on both sites
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ item }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/site 1/ ena"
      when: state == "present"
      loop: "{{ radware_provider }}"

    - name: delete GSLB remote site config on both sites
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ item }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/site 1/ del"
      when: state == "absent"
      loop: "{{ radware_provider }}"
```

```
   - name: enable GSLB dns responder on both sites
      radware.radware_modules.alteon_config_alteon_cli_command:
        provider: "{{ item }}"
        state: present
        parameters:
          alteon_cli_command: "/c/slb/gslb/dnsresvip DnsResp1,DnsResp2/
ena"
      when: state == "present"
      loop: "{{ radware_provider }}"

    - name: site A GSLB rules config
      radware.radware_modules.alteon_config_gslb_rule:
        provider: "{{ item }}"
        state: "{{ state }}"
        parameters:
          index: 1
          state: enabled
          dns_ttl: 180
          description: gslb_rule
          rule_metrics:
            - metric: phash
              priority: 1
            - metric: roundrobin
              priority: 8
      loop: "{{ radware_provider }}"
```

```
- name: site A virt config
  radware.radware_modules.alteon_config_virtual_server:
    provider: "{{ radware_provider[0] }}"
    state: "{{ state }}"
    parameters:
      index: gslb_virt
      state: enabled
      domain_name: "{{ gslb_srv.fqdn }}"
      ip_address: "{{ gslb_srv.site_a.vip }}"

- name: site B virt config
  radware.radware_modules.alteon_config_virtual_server:
    provider: "{{ radware_provider[1] }}"
    state: "{{ state }}"
    parameters:
      index: gslb_virt
      state: enabled
      domain_name: "{{ gslb_srv.fqdn }}"
      ip_address: "{{ gslb_srv.site_b.vip }}"

- name: site A service config
  radware.radware_modules.alteon_config_virtual_service:
    provider: "{{ radware_provider[0] }}"
    state: "{{ state }}"
    parameters:
      index: gslb_virt
      service_index: 1
      service_port: 80
      server_group_name: group_gslb
      application_type: http
      nat_mode: address
      nat_address: 192.168.3.100
```

```
- name: site B service config
   radware.radware_modules.alteon_config_virtual_service:
     provider: "{{ radware_provider[1] }}"
     state: "{{ state }}"
     parameters:
       index: gslb_virt
       service_index: 1
       service_port: 80
       server_group_name: group_gslb
       application_type: http
       nat_mode: address
       nat_address: 192.168.3.100

 - name: apply configuration
   radware.radware_modules.alteon_mng_config:
     provider: "{{ item }}"
     command: apply
   loop: "{{ radware_provider }}"
```

Appendix 9 – alteon_params.yml (GSLB Demo)

```
radware_provider:
  - server: 192.168.1.1
    user: admin
    password: radware
    validate_certs: no
    https_port: 443
    ssh_port: 22
    timeout: 10
  - server: 192.168.1.2
    user: admin
    password: radware
    validate_certs: no
    https_port: 443
    ssh_port: 22
    timeout: 10

gslb_srv:
    fqdn: app.demo.lab
    gslb_metric: phash
    site_a:
      clnt_int: 192.168.2.2
      srvr_int: 192.168.3.2
      vip: 192.168.2.101
      dns_responder: 192.168.2.200
          index: real31
```

```
    site_b:
      clnt_int: 192.168.2.3
      srvr_int: 192.168.3.3
      vip: 192.168.2.102
      dns_responder: 192.168.2.201
      local_servers:
        - address: 192.168.3.11
          index: real311

state: present
```

North America

Radware Inc.

575 Corporate Drive

Mahwah, NJ 07430

Tel: +1-888-234-5763

International

Radware Ltd.

22 Raoul Wallenberg St.

Tel Aviv 69710, Israel

Tel: 972 3 766 8666