F5-Ansible hands-on Lab

March 28, 2017

Authors:

PK Iyer – p.iyer@f5.com

Ilya Revich – irevich@f5.com

Welcome to the Agility Roadshow Lab day! We'll start off by checking access to our environment. Each student has an independent lab environment, that can be accessed as follows:

1) In a browser, go to:

https://goo.gl/2GOQ4A

Login Credentials:
Username: student_X (X = number on your lab guide)
Password: passw0rd (all lowercase)

2) Click on 'Ansible Lab …'
3) You will see 4 VMs running. Look for 'control' and on the far right, click on 'Console'.
4) If a window does not open up, unblock pop ups.
5) You will see an Ubuntu logon page for user 'Ubuntu' in the tab.

Login Credentials:
Username: Ubuntu (should already be populated)
Password: ravelloCloud

6) Launch 'xterm' by double-clicking the xterm icon. xterm is a terminal emulator, similar to Putty or gitbash
7) Go to the Ansible directory by typing → `cd /home/ubuntu/ansible/`
8) Run `ls -la` and look at the files and directories inside the ansible directory.

Introduction:

This lab is intended to get the participant familiar with Ansible and its various building blocks. This lab talks about

1. Installing Ansible
2. Using Ansible command line
3. Using Ansible to bootstrap a webserver
4. Managing configuration on a BIG-IP

About the environment:

This lab is setup in the Ravello environment. You log into your 'app environment' in Ravello to start using the lab. To do this:

1. There are 4 components in this environment. They are:
    1. an F5 BIG-IP and
    2. 3 Ubuntu-server machines (2 webservers and 1 control)
2. There are 2 networks in the environment:
    1. 10.10.10.X network for mgmt-traffic

2. 192.168.1.X network for Data-traffic
3. The BIG-IP has a Web-UI and SSH access via Public IP. We will be using the Web UI to view our configurations.

Installing Ansible

1. Ansible will be installed on the 'Ansible' Component of the lab. Access the Ansible host via Shell or SSH and run the following commands to install

   Ansible install commands

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

NOTE:

Press Enter when prompted while adding repository (Command 2).

You will install an older version of Ansible if you don't run the update on apt after adding the repository (command 3).

Check Ansible Installation:

To ensure Ansible has been properly installed on the host, let us check the version to make sure we have the proper version installed.

- **check Ansible Version**

  ```
  root@ubuntu:~# ansible --version
  ```

We can see that Ansible ***2.2.1.0*** has been installed on the host and the config file points to ***/home/ubuntu/ansible/ansible.cfg***

2. After installing Ansible we will need the **f5-sdk module** which implements an SDK for the iControl® REST interface for BIG-IP®. This library uses **python** to automate a BIG-IP via its REST API. Run the command below to install f5-sdk on the Ansible host.

- **Install f5-sdk library**

  ```
  sudo pip install f5-sdk
  ```

We now have a working F5-SDK in the /home/ubuntu/ansible/ directory.

Using Ansible

1. Modify the /home/ubuntu/ansible/ansible.cfg file for your environment

   ```
   sudo nano ansible.cfg
   ```

   Copy/Paste the following lines in the **/home/ubuntu/ansible/ansible.cfg** file (You can also uncomment/manually add)

   ansible.cfg

   ```
   inventory         = /home/ubuntu/ansible/hosts
   remote_tmp        = $HOME/.ansible/tmp
   pattern           = *
   forks             = 5
   poll_interval     = 15
   sudo_user         = root
   transport         = smart
   module_lang       = C
   gathering         = implicit
   host_key_checking = False

   sudo_exe          = sudo
   timeout           = 10
   remote_user       = root
   roles             =/home/ubuntu/ansible/playbooks/roles #add if this line is
   not present.
   ```

2. Create entries in the **/home/ubuntu/ansible/hosts** file. This will be your Ansible 'Inventory'. The host file must look like below for this lab.

   ```
   sudo nano hosts
   ```

   Copy/Paste the following lines in the hosts file

   ```
   # This is the default ansible 'hosts' file.
   #
   # It should live in /home/ubuntu/ansible/hosts
   #
   #   - Comments begin with the '#' character
   #   - Blank lines are ignored
   #   - Groups of hosts are delimited by [header] elements
   #   - You can enter hostnames or ip addresses
   #   - A hostname/ip can be a member of multiple groups

   #information on the BIG-IP
   [bigip]
   10.10.10.15

   # BIG-IP SSH login credentials
   [bigip:vars]
   ansible_user=root
   ```

```
ansible_ssh_pass=default


# webserver information
[webservers]
web01.internal ansible_ssh_host="10.10.10.21"
web02.internal ansible_ssh_host="10.10.10.22"

[webservers:vars]
ansible_user=root
ansible_ssh_pass=default

[local]
localhost
```

3. Make sure config and inventory works. Run the below commands on your Ansible hosts.

   sample commands

   ```
   ansible webservers[0] -a 'uname -a'
   ansible webservers[1] -a 'uname -a'
   ansible bigip -a 'tmsh show sys version'
   ```

NOTE: This is the simplest way to run remote commands/scripts on a remote host using Ansible. Each of the above command is executed on its respective host (derived from the /home/ubuntu/ansible/hosts file). Webservers[0] points to web01.internal, webservers[1] points to web02.internal. bigip is the BIG-IP host.

Using Ansible to bootstrap a Web-server

For this Lab, we will be using 2 ubuntu servers as our pool members. We will use Ansible to install Apache on both the webservers and create a custom webpage for each web server. We will explore how Ansible roles are used to achieve this.

1. Create working directories in /home/ubuntu/ansible/ for playbooks and roles.

   create directories for playbooks and roles

   ```
   cd /home/ubuntu/ansible
   sudo mkdir playbooks
   ```

2. We will first define the role apache in /home/ubuntu/ansible/roles. This role can be applied to any number of Ansible-nodes.
   1. Each role will have Ansible-task(s) defined in it and a template if needed.
   2. In this case, we will use a template to create the index pages for each of the web-servers.

      create directories for playbooks and roles

```
cd playbooks/

sudo mkdir roles

cd /home/ubuntu/ansible/playbook/roles

sudo mkdir apache
cd apache
sudo mkdir tasks
sudo mkdir templates
cd tasks/
sudo nano main.yaml
```

add the following code to main.yaml

main.yaml

```
#main.yaml
---

- name: Install Apache webserver
  apt:
      name: "apache2"
      state: "present"
      force: "yes"

- name: Copy out index page to box
  template:
       src: "index.html"
       dest: "/var/www/html/index.html"
```

Now define template for index.html

defining template for index.html

```
cd /home/ubuntu/ansible/roles/apache/templates
sudo nano index.html
```

add the following line in index.html, save and exit file.

index.html

```
page from {{ inventory_hostname }}
```

3. Create a new playbook called webserver_bootstrap.yaml and use this role install apache on both the nodes.

create webserver_bootstrap.yaml playbook

```
cd /home/ubuntu/ansible/playbooks
```

```
sudo nano webserver_bootstrap.yaml
```

add the following to the webserver_bootstrap.yaml file and save and exit.

webserver_bootstrap.yaml

```
---

- name: Bootstrap webserver
  hosts: webservers
  become: yes
  become_user: root
  roles:
      - apache
```

4. Once we have the roles defined and playbook written, we can run the playbook. Running the playbook will take a few minutes as this requires new installation.

   running webserver_bootstrap.yaml playbook.

```
cd /home/ubuntu/ansible/playbooks
ansible-playbook webserver_bootstrap.yaml
```

   Sample output when the playbook is run:

   sample output

```
<example output>
PLAY [Bootstrap webserver]
*****************************************************

TASK [setup]
*******************************************************************
ok: [web01.internal]
ok: [web02.internal]

TASK [apache : Install Apache webserver]
***************************************
changed: [web01.internal]
changed: [web02.internal]

TASK [apache : Copy out index page to box]
*************************************
changed: [web01.internal]
changed: [web02.internal]

PLAY RECAP
*********************************************************************
web01.internal             : ok=3    changed=2    unreachable=0
failed=0
web02.internal             : ok=3    changed=2    unreachable=0
failed=0
```

5. If there are no failures above, you have now successfully configured apache on web01.internal and web02.internal. You can login to both servers and check your installation.

   check web server installation

   ```
   On web01.internal
   -----------------
   root@ubuntu:~# cd /var/www/html/
   root@ubuntu:/var/www/html# cat index.html
   page from web01.internal

   root@ubuntu:~# curl http://localhost
   page from web01.internal

   ------
   # check similarly on web02.internal
   ```

Managing configuration on a BIG-IP

On the BIG-IP we will use ansible to:

1. Bootstrap BIG-IP and create a pool with web server
2. Create an HTTPS-Application

Bootstrap BIG-IP and create a pool with web-server

To boot strap the BIG-IP we will create another role for the pool members. And write a playbook to network the BIGIP and also create a pool with the pool members.

1. Create role for Pool Members.

   pool-member role

   ```
   cd /home/ubuntu/ansible/roles/
   sudo mkdir pool-member
   cd pool-member
   sudo mkdir tasks
   cd tasks
   sudo nano main.yaml
   ```

   Add the following to the main.yaml task, save and exit file. Notice that this file has 3 tasks in it.

   main.yaml

   ```
   ---

   # the 'delegate_to' keyword on a task is used if:
   # you want to perform a task on one host with reference to other hosts
   # This is ideal for placing nodes in a load balanced pool, or removing them.
   ```

```
- name: Create node # task1
  bigip_node: # module1
      host: "{{ hostvars[inventory_hostname]['ansible_eth1']['ipv4']['address'] }}"
      name: "{{ inventory_hostname }}"
      server: "10.10.10.15"
      password: "admin"
      validate_certs: "no"
      user: "admin"
      monitors:
          - "icmp"
  delegate_to: localhost

- name: Add node to pool # task2
  bigip_pool_member: # module2
      host: "{{ hostvars[inventory_hostname]['ansible_eth1']['ipv4']['address'] }}"
      port: "80"
      server: "10.10.10.15"
      user: "admin"
      password: "admin"
      validate_certs: "no"
      name: "{{ inventory_hostname }}"
      pool: "http_pool" # we will create a pool called http_pool in playbook
  delegate_to: localhost

- name: Create HTTP monitor # task3
  bigip_monitor_http: #module3
      name: "monitor-{{ inventory_hostname }}"
      user: "admin"
      server: "10.10.10.15"
      password: "admin"
      receive: ""
      send: "GET /"
      state: "present"
      validate_certs: "no"
  delegate_to: localhost
```

Now that we have role for the pool-member defined, let us create a playbook for bootstrapping the BIG-IP

create bigip_bootstrap.yaml

```
cd /home/ubuntu/ansible/playbooks
sudo nano bigip_bootstrap.yaml
```

add the following to bigip_bootstrap.yaml, save and exit file.

bigip_bootstrap.yaml

```
# Playbook: bigip_bootstrap.yaml
# This playbook creates an internal vlan and assigns a self-ip
# This playbook also creates a Pool and adds pool members to the pool
# 2 plays, 3 tasks and 1 role used for this playbook

- name: Bootstrap BIGIP # play1
  hosts: bigip
  connection: local

  tasks: # 3 tasks in this play
```

```
          - name: create internal vlan # task1
            bigip_vlan: # module1
                server: "{{ inventory_hostname }}"
                user: "admin"
                password: "admin"
                tag: "10"
                tagged_interfaces:
                    - 1.1
                name: "internal_vlan"

          - name: create internal self-ip
            bigip_selfip: # module2
                server: "{{ inventory_hostname }}"
                user: "admin"
                password: "admin"
                address: "10.10.10.15"
                netmask: "255.255.255.0"
                vlan: "internal_vlan"
                allow_service: "default"
                name: "int_selfip"

          - name: create Pool for webservers
            bigip_pool: # module3
                lb_method: "ratio_member"
                server: "{{ inventory_hostname }}"
                user: "admin"
                password: "admin"
                name: "http_pool"
                validate_certs: "no"
                monitors:
                    - "/Common/http"


    - name: Put webservers in pool # play2
      hosts: webservers
      connection: ssh
      roles: # role being used
          - pool-member
```

We can run the playbook above to bootstrap the BIG-IP. We also need **bigsuds** for running the big-ip module.

run bigip_bootstrap.yaml playbook

```
pip install bigsuds
ansible-playbook bigip_bootstrap.yaml -v
```

Sample output when the playbook is run

sample output when playbook run

```
Using /home/ubuntu/ansible/ansible.cfg as config file
```

```
PLAY [Bootstrap BIGIP]
*********************************************************

TASK [setup]
*******************************************************************
ok: [10.10.10.15]

TASK [create internal vlan]
****************************************************
ok: [10.10.10.15] => {"changed": false}


TASK [create internal self-ip]
*************************************************
ok: [10.10.10.15] => {"changed": false}

TASK [create Pool for webservers]
**********************************************
ok: [10.10.10.15] => {"changed": false}

PLAY [Put webservers in pool]
**************************************************

TASK [setup]
*******************************************************************
ok: [web01.internal]
ok: [web02.internal]

TASK [pool-member : Create node]
***********************************************
changed: [web01.internal -> localhost] => {"changed": true}
changed: [web02.internal -> localhost] => {"changed": true}

TASK [pool-member : Add node to pool]
******************************************
changed: [web01.internal -> localhost] => {"changed": true}
changed: [web02.internal -> localhost] => {"changed": true}

TASK [pool-member : Create HTTP monitor]
***************************************
changed: [web02.internal -> localhost] => {"changed": true}
changed: [web01.internal -> localhost] => {"changed": true}


PLAY RECAP
*********************************************************************
10.10.10.15                   : ok=4    changed=0    unreachable=0
failed=0
web01.internal             : ok=4    changed=3    unreachable=0    failed=0
web02.internal             : ok=4    changed=3    unreachable=0    failed=0
```

Login to the BIG-IP via the Web-GUI to make sure its configured as expected.

Lab 1

Create an HTTPS-Application

Now that you're familiar with how to write simple plays and playbooks, for this lab, we will create a Virtual server with the pool we just configured. From your ansible host:

create https application playbook

```
cd /home/ubuntu/ansible/playbooks
sudo nano bigip_https_vs.yaml
```

Add the following content to bigip_https_vs.yaml, save file and exit.

bigip_https_vs.yaml

```
- name: creating HTTPS application
  hosts: bigip
  connection: ssh
  gather_facts: true

  tasks:
      - name: create a virtual server
        bigip_virtual_server:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            name: "https_vs"
            snat: "Automap"
            validate_certs: "no"
            destination: "10.10.10.16"
            port: 443
            all_profiles:
                - http
                - clientssl
            pool: "http_pool"
        delegate_to: localhost
```

Run the bigip_https_vs.yaml playbook from the Ansible Host.

run bigip_https_vs.yaml playbook

```
ansible-playbook bigip_https_vs.yaml -v
```

Sample output when playbook is run:

sample output when playbook run

```
root@ubuntu:/home/ubuntu/ansible/playbooks# ansible-playbook
bigip_https_vs.yaml -v
Using /home/ubuntu/ansible/ansible.cfg as config file

PLAY [creating HTTPS application]
*********************************************

TASK [setup]
*****************************************************************
ok: [10.10.10.15]

TASK [create a virtual server]
************************************************
changed: [10.10.10.15 -> localhost] => {"changed": true}

PLAY RECAP
*******************************************************************
10.10.10.15                : ok=2    changed=1    unreachable=0
failed=0
```

You can check if your BIG-IP is configured and if your application is running as expected. To do this using curl from your Ansible host to GET the webpages.

Lab 1 Notes:

In this Lab, we used ansible to execute simple command line functionality in BIG-IP and on ubuntu machines. We examined the concept of Roles and Templates of Ansible. We also wrote simple plays and tasks using Ansible's BIG-IP modules and used them to write simple playbooks.

Lab 2

Create an HTTPs virtual server with clientssl and serverssl profile

create https application playbook

```
cd /home/ubuntu/ansible/playbooks
sudo nano new_bigip_https_vs.yaml
```

Add the following content to new_bigip_https_vs.yaml, save file and exit.

```
- name: creating HTTPS application
  hosts: bigip
  connection: ssh
  gather_facts: true

  tasks:
      - name: create a virtual server
        bigip_virtual_server:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            name: "new_https_vs"
            snat: "Automap"
            validate_certs: "no"
            destination: "10.10.10.17"
            port: 443
            all_profiles:
                - http
                - clientssl
                -  serverssl
            pool: "http_pool"
        delegate_to: localhost
```

Run the new_bigip_https_vs.yaml playbook from the Ansible Host.

```
ansible-playbook new_bigip_https_vs.yaml -v
```

Lab 2 Notes:

In this Lab, we attempted to create a 2nd virtual server with additional profiles. In doing so, we needed to remove a rogue white space, after which the playbook runs successfully.

Optional

Lab 3

Options to consider:

1 – Create 10 virtual servers

2 – Create 10 pools

3 – Add 10 nodes to existing pools

For more information on Ansible:

Check out http://go/ansible

Published modules: http://docs.ansible.com/ansible/list_of_network_modules.html#f5

On going work: https://github.com/search?q=ansible-f5

Please feel free to reach out to DevOpsBD@f5.com with any queries or questions.