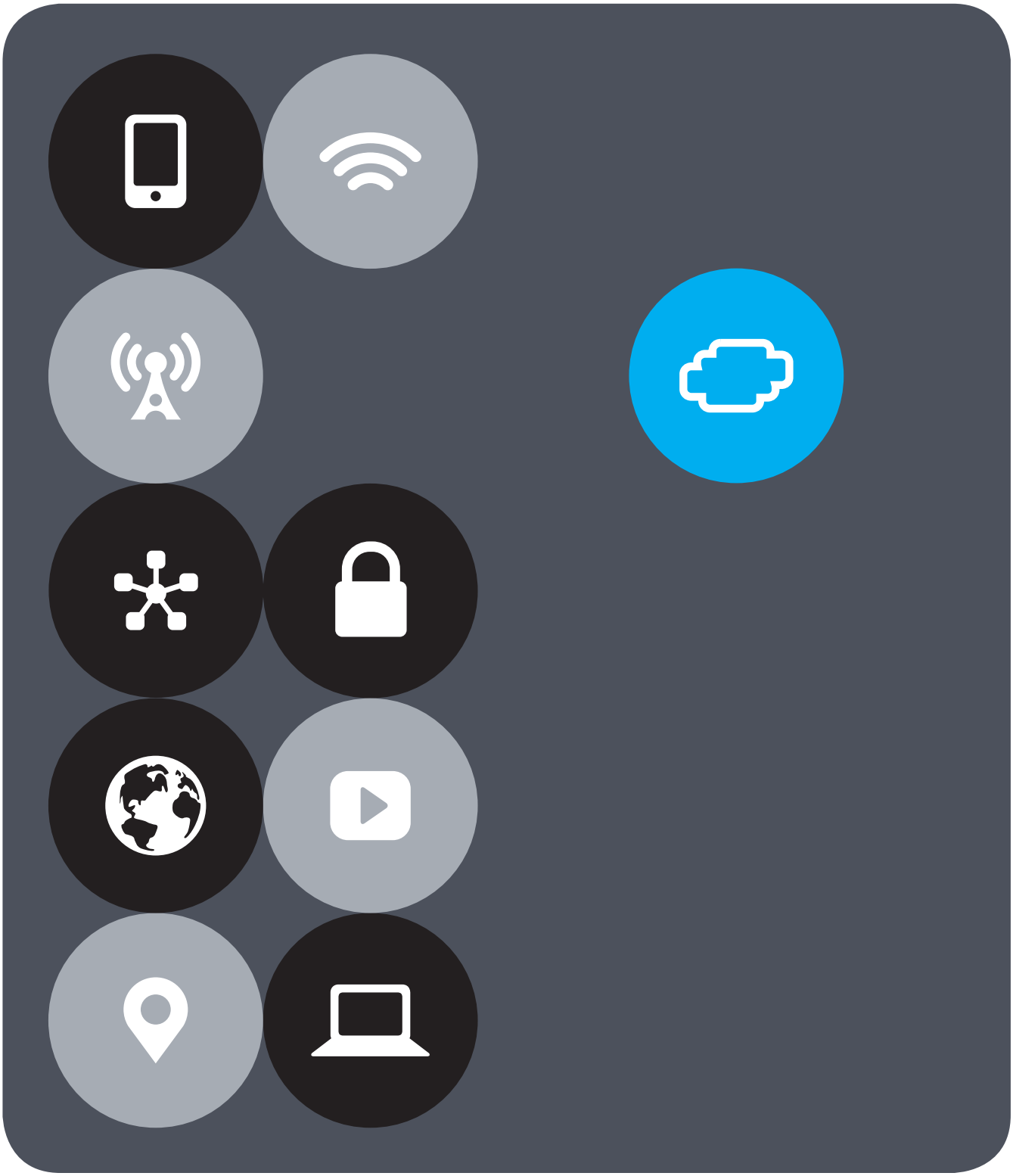
F5 + ANSIBLE

TECHNICAL WHITE PAPER

Automating the F5 BIG-IP Platform with Ansible



Contents

Introduction

"The key drivers for the use of DevOps-related frameworks and toolsets remain scalability and reduction of operational expenses."[1]

Traditionally, organizations deploy infrastructure and applications using a combination of various documents such as deployment guides along with many manual processes and operations. It's a time-consuming approach that doesn't align with ever-increasing requirements for speed and agility. The need to deploy full application stacks and services more quickly and more often in a repeatable manner has driven both development and operational teams toward automation and orchestration. In addition to enabling organizations to better manage applications, infrastructure deployments, and the process of provisioning and de-provisioning, automation reduces the amount of IT resources required and ensures increased reliability, efficiency, and agility.

The programmable network

Through a deep understanding of best practices for networking and application delivery, F5 empowers organizations to take advantage of the benefits of automation and programmability as they configure and manage devices on the BIG-IP platform. Both the hardware and virtual editions of F5 Application Delivery Controllers (ADCs) feature flexible and programmable management, control and data planes.

F5 ADCs achieve programmability through the following features:

1. Traffic Management Shell (tmsh): Allows complete access to configure system features and set up and manage network elements.
2. iControl: Utilizes SOAP/XML/REST to ensure open communications between dissimilar systems and automate BIG-IP functionality.
3. iApps: Templated sets of functionalities helps automate configuration and application deployment on F5 devices.
4. iRules: Tcl-based scripting language is a highly customizable and programmatic way to inspect, analyze, modify, route, re-direct or manipulate traffic in real time. The next evolution, iRules Language eXtensions (LX) enables node.js capabilities on the BIG-IP platform (v12.1).

In addition to programmability, F5's rich, supported APIs improve operational agility, responsiveness, and the effectiveness of managing infrastructure and application deployment, which is essential in DevOps and cloud environments.

1 https://f5.com/about-us/news/the-state-of-application-delivery

A comprehensive joint solution

F5 and Ansible address the modern business need for agility in support of continuous integration and continuous deployment through a comprehensive solution. Ansible provides a simple yet powerful multi-tier automation and orchestration platform, which can help organizations automate application deployment and configuration management, while ensuring repeatability. Unlike other agent-based tools in the DevOps community, Ansible does not require installing any agent or software on the BIG-IP platform. Ansible communicates and manages most devices over SSH or through API calls.

Furthermore, Ansible version 2.2 and above offer F5 modules for managing the BIG-IP platform. These modules use SOAP and REST APIs to control and manage BIG-IP devices whether they are hardware or virtual editions. To make use of the APIs, it is necessary to install the Python libraries of f5-sdk (for iControl REST) and bigsuds (for iControl SOAP) on the Ansible host.

They can both be easily installed via pip: `pip install f5-sdk bigsuds`

Ansible F5 modules make it simple to automate BIG-IP onboarding configurations like hostname, DNS, and NTP—and even network the BIG-IP device. Ansible offers F5 modules for managing F5 BIG-IP DNS and BIG-IP Local Traffic Manager (LTM) objects, including deploying and managing HTTP and HTTPS virtual servers, as well as configuring and managing pools and pool members. All the Ansible F5 modules follow best practices, use BIG-IP APIs, can be used with TMOS version 12.0 and above.

Use case: configure an HTTPS application on the BIG-IP platform

Configuring a HTTPS application on a BIG-IP device requires a certain amount of F5 expertise and typically entails the following high-level tasks.

1. Adding the backend servers as BIG-IP nodes
2. Assigning node monitors
3. Adding these nodes to a pool(s)
4. Adding pool monitors
5. Configuring a ClientSSL profile
6. Configuring a virtual server and adding the above items to the virtual server

This means that a BIG-IP expert must visit 8–10 different GUI sections on the BIG-IP device, and configure each section manually. This may seem manageable without automation on a single BIG-IP device, but a modern application seldom runs on a single BIG-IP appliance. The task of configuring several BIG-IP devices and accomplishing this across multiple data centers quickly becomes a highly time- and labor-intensive process. However, by using task-oriented Ansible playbooks, organizations can manage and automate secure application deployment on multiple BIG-IP devices across data centers.

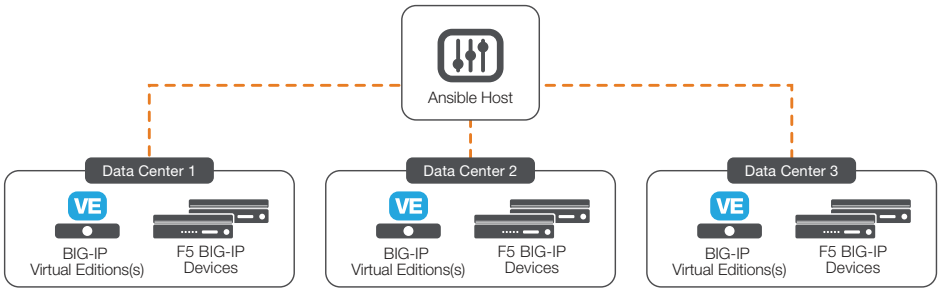


Figure 1: BIG-IP devices across various data centers can be configured and managed by an Ansible host.

The sample Ansible playbook shown below assumes that the user's Ansible "Inventory" file has a list of BIG-IP devices under the "bigips" group along with each device's corresponding group. When the playbook is run, the HTTPS application will be configured on the BIG-IPs in the "bigips" group. In the example below, the "bigips" group includes a BIG-IP device with management IP 172.27.74.50.

```
cat /etc/ansible/hosts

[bigips]
172.27.74.50

[bigips:vars]
ansible_user=root
ansible_ssh_pass=********
ui_uname=admin
ui_pass=**********

[webservers]
y.y.y.y

[webservers:vars]
ansible_user=root
ansible_ssh_pass=**********
```

This example features one main play, which is accomplished through four tasks:

- The play: deploy an HTTPS application on hosts under the “f5” group in the Inventory file.
- Task 1: An Ansible F5 module called “bigip_node” adds a node member.
- Task 2: The “bigip_pool” module creates a pool.
- Task 3: The “bigip_pool_member” module adds the pool member created in the first task to the pool created in the second task.
- Task 4: The “bigip_virtual_server” module creates a virtual server that has “http” and “ClientSSL” profiles; secure network address translation (SNAT) configured to “Automap”; and the pool created in the second task.

Here’s a sample playbook for configuring an HTTPS application on the BIG-IP platform:

```
- name: creating HTTPS application
  hosts: bigips

  tasks:
      - name: Create a HTTP node
        bigip_node:
            server: “{{ inventory_hostname }}”
            user: “admin”
            password: “admin”
            host: “10.1.10.98”
            name: “http_node”
            validate_certs: “no”
        delegate_to: localhost

      - name: Create a web-pool
        bigip_pool:
            server: “{{ inventory_hostname }}”
            user: “admin”
            password: “admin”
            lb_method: “ratio_member”
            monitors: http
            name: “web-pool”
            validate_certs: “no”
        delegate_to: localhost

      - name: Add http node to web-pool
        bigip_pool_member:
            description: “HTTP Webserver-1”
            host: “{{ item.host }}”
            name: “{{ item.name }}”
            user: “admin”
            password: “admin”
            pool: “web-pool”
            port: “80”
            server: “{{ inventory_hostname }}”
            validate_certs: “no”
```

```
    with_items:
        - host: "10.1.10.98"
          name: "http_node"
    delegate_to: localhost

  - name: Create a virtual server
    bigip_virtual_server:
        description: "Secure web application"
        server: "{{ inventory_hostname }}"
        user: "admin"
        password: "admin"
        name: "htttp_vs"
        destination: "10.10.20.100"
        port: 443
        snat: "Automap"
        all_profiles:
            - http
            - clientssl
        pool: "web-pool"
        validate_certs: "no"
    delegate_to: localhost
```

As shown in the playbook above, it is recommended to use the "delegate_to: localhost" parameter with F5 modules, which ensures that Ansible uses the supporting Python modules on the Ansible host and does not depend on the BIG-IP device.

Before running the playbook, there are no virtual servers, pools, or nodes configured on the BIG-IP device. As shown below, all the tasks in the play "creating HTTPS application" were completed successfully with no failures.

```
ansible-playbook create_https_app.yaml

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

TASK [setup] *******************************************************************
ok: [172.27.74.50]

TASK [Create a HTTP node] ******************************************************
changed: [172.27.74.50 -> localhost]

TASK [Create a web-pool] *******************************************************
changed: [172.27.74.50 -> localhost]

TASK [Add http node to web-pool] ***********************************************
changed: [172.27.74.50 -> localhost] => (item={u'host': u'10.1.10.98', u'name':
u'http_node'})

TASK [Create a virtual server] *************************************************
changed: [172.27.74.50 -> localhost]

PLAY RECAP *********************************************************************
172.27.74.50               : ok=5    changed=4    unreachable=0    failed=0
```

Also notice the play recap and specifically "changed=4," which shows that the four tasks in the playbook configured four objects on the BIG-IP device. Ansible has now created a new, fully configured virtual server (shown in Figure 2), and this server is load balancing this application across a newly created pool (shown in Figure 3).

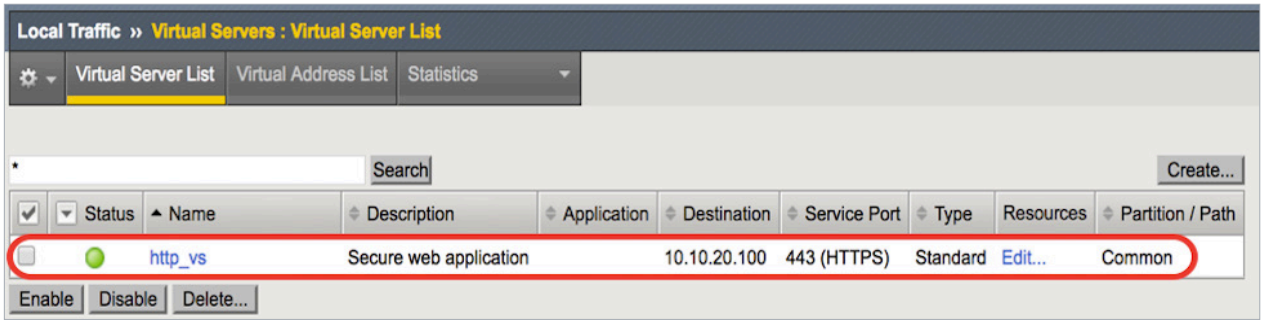


Figure 2: A new virtual server ("http_vs") has been configured using Ansible.

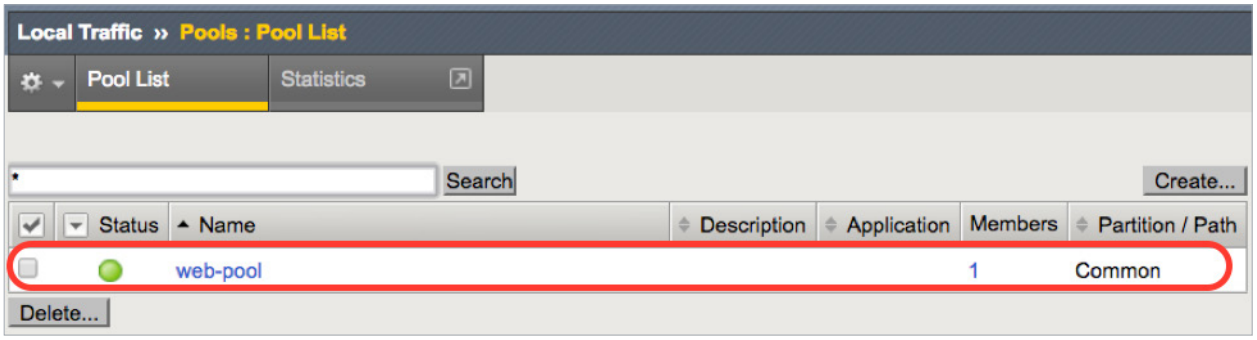


Figure 3: The BIG-IP pool, created by Ansible.

And recall that all the Ansible F5 modules are idempotent, which means that if the same playbook is run again, Ansible does not reconfigure these objects. The play recap will read "changed=0" unless something has changed in the playbook.

Conclusion

With applications transitioning into the cloud and organizations embracing agility, automating infrastructure and application deployments is essential to ensure businesses stay ahead. Over the years, F5 has fully embraced the idea of using automation technologies and programmability to help automate the network. The rich APIs of F5 devices combined with the agentless architecture of Ansible makes it easy to automate application deployments and manage infrastructure in both public and private clouds.

For more information on automation and how the F5 and Ansible partnership can help your organization, visit www.ansible.com/ansible-f5.

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 f5.com

Americas
info@f5.com

Asia-Pacific
apacinfo@f5.com

Europe/Middle-East/Africa
emeainfo@f5.com

Japan
f5j-info@f5.com