F5 Ansible Documentation

Release 1.0.0

F5 Networks

Jul 25, 2017

User Documentation

This project implements a set of Ansible modules for the F5 Networks® BIG-IP® Users of these modules can create, edit, update, and delete configuration objects on a BIG-IP®. For more information on the basic principals that the modules use, see the usage/index.

The code is open source, and available on github. Additionally, some modules have been promoted to the Ansible core product and Ansible extras.

The main documentation for the modules is organized into several sections listed below.

CHAPTER 1

Getting Started

This document will show you how to begin using the F5 Ansible modules. You will create a pool, add two nodes to that pool, and finally assign a virtual server to serve requests to the nodes in the pool.

First, obtain Python 2.7 and Ansible if you do not already have them.

The version of Ansible that is required is at least 2.2.0.

Installing Ansible

Let's install Ansible to make it possible to use the modules.

First, make sure *ansible* is installed.

```
pip install ansible
```

Note: You should *only* install Ansible from *pip*. While you might find Ansible packaged for your operating system (for instance via *apt*, *yum*, or *brew*, the only official way to get Ansible is via *pip*. We cannot assist you if you have installed it any other way.

You should be able to verify that you are running Ansible by using the *–version* argument to the *ansible* command, like so.

```
ansible --version
```

You should be presented with output that resembles the following

```
(test1)SEA-ML-RUPP1:virtualenv trupp$ ansible --version
ansible 2.2.0
  config file =
  configured module search path = Default w/o overrides
```

With this ready, you can create your first playbook. We'll write the remainder of our Ansible playbooks in a file called `site.yaml`

Installing Modules

Refer to the documentation on installing the modules here.

Playbook

The remainder of this tutorial will walk you through the various steps of adding tasks to your playbook.

Note: I've broken each task into its own section to better explain it. This might lead to confusion though, so if you just want to get the whole file and then follow along, you can download it here.

Let's begin by placing the following in your `site.yaml`:

```
---

- name: Create a VIP, pool, pool members and nodes
  hosts: big-ip01.internal
  connection: local
```

Your BIG-IP is probably not called `big-ip01.internal`. It might be a different hostname or even IP address. Whichever it is, place it in the hosts line.

Add a pool

A pool represents a collection of resources. These resource typically deliver a service that is identical. By assigning them to a pool, the BIG-IP is able to distribute requests amongst all of them.

Add the following to your `site.yaml` to create a pool called `web`:

```
tasks:
   - name: Create a pool
     bigip_pool:
          lb_method: "ratio_member"
          name: "web"
          password: "admin"
          server: "big-ip01.internal"
          slow_ramp_time: "120"
          user: "admin"
          validate_certs: "no"
     delegate_to: localhost
```

Add two nodes

Now we want to create the nodes in our BIG-IP configuration. These represent the actual devices on your network. They could be physical gear, VMs, or other devices.

To add the two nodes, we'll put the following in our `site.yaml`

```
- name: Create node1
  bigip_node:
      host: "10.10.10.10"
      name: "node-1"
      password: "admin"
      server: "big-ip01.internal"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Create node2
  bigip_node:
      host: "10.10.10.20"
      name: "node-2"
      password: "admin"
      server: "big-ip01.internal"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Note: It is important that you correctly space over this and the remaining tasks so that they align vertically with the `Create a pool` task above. If you do not do this, Ansible will raise an error.

Add the nodes to the pool

With the pool created and your nodes in place, you not want to add those nodes to the pool. At this point we would refer to those nodes as pool members.

```
- name: Add nodes to pool
  bigip_pool_member:
      description: "webserver-1"
      host: "{{ item.host }}"
      name: "{{ item.name }}"
      password: "admin"
      pool: "web"
      port: "80"
      server: "big-ip01.internal"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
  with_items:
      - host: "10.10.10.10"
        name: "node-1"
      - host: "10.10.10.20"
        name: "node-2"
```

Add a virtual server

Now that our pool is set up and the nodes are members of that pool, we next want to create a VIP so that external requests can be delivered to the pool members.

The below example uses `172.16.10.108` as the external address, so you may need to change it for your own environment

To create a virtual server, add the following to you `site.yaml`:

```
- name: Create a VIP
  bigip_virtual_server:
      description: "foo-vip"
      destination: "172.16.10.108"
      password: "admin"
      name: "vip-1"
      pool: "web"
      port: "80"
      server: "big-ip01.internal"
      snat: "Automap"
      user: "admin"
      all_profiles:
           - "http"
           - "clientssl"
      validate_certs: "no"
  delegate_to: localhost
```

More info

Curious what else is possible with the current modules? Interested in test-driving the modules under development? Refer to the sidebar for links relevant to your interests.

CHAPTER 2

Versions

There are a number of versions of software the F5 officially supports that can be found here. This should not be considered the list that the Ansible modules support though.

We're doing our best to move more towards using the REST API, so with that in mind, we're also moving many of the Ansible modules along in terms of supported BIG-IP versions.

Generally speaking, the versions of BIG-IP that are tested and supported by the Ansible modules include:

- 11.6.0
- 12.0.0
- 12.1.0

So that the oldest that we're willing to aim for is when the REST interface went GA.

If you're seriously interested in the Ansible modules, then consider it a reason to upgrade if you have not yet done so.

CHAPTER 3

Support

The F5 Ansible modules are developed primarily with the REST API in mind. Due to this requirement, we often take the approach that newer versions of BIG-IP are those that are best supported.

The versions of BIG-IP that support these modules typically start at version 12.0.0, but may require later versions than that depending on the REST functionality that is needed by them.

Incentives

Fortunately or unfortunately, the Ansible modules are not recognized by F5 as a supported product at this time.

Due to this constraint, you could say that the maintainers of these modules have limited resources to focus on these full time. We do our best though, and offload many of the more mundane tasks to automation where possible.

With those constraints in mind, we hope you can see why we are not able to focus more on older releases of BIG-IP. Consider these modules as an incentive to upgrade your BIG-IP to a later version.

Finding assistance

If you need help with anything related to these modules, it is recommended that you open an issue on Github

https://github.com/F5Networks/f5-ansible/issues

We usually respond promptly and may ask you to contact us offline if we need to deal with something that would not be appropriate on a public forum.

When communicating with us on the Issues page, we generally recommend that you do it via Github's UI and not via email. The reason for this is that we have seen examples where email communication may expose the name of your company when communicating with us.

Whether your company has an issue with this or not, it's probably best if we stay vendor neutral when discussing the technical issues on a public forum. If you need more in-depth technical know-how, you're free to ask us to ping you offline and we can handle things there.

Credentials and secret things

It's considered bad form to expose credentials in a Github issue. Please be diligent of that!

We *do not need any* of the following task arguments to debug your issue

- user
- password
- server
- server_port

therefore, please be diligent and either

- Do not provide them (leave them empty with empty quotes "")
- Provide placeholders for them (such as "admin", "secret", and "lb.mydomain.com")

We are very well equipped in terms of physical and virtual BIG-IPs to be able to diagnose and test your problems. Therefore we never need this information to provide your with assistance.

How to know which modules are supported

Remember that, ultimately, this repository contains *experimental* code.

However, with that said, there is a quick way to figure out if a particular module you are interested in has been tested to work on a particular platform or not.

To find that out, look for your modules in the *tests/* directory.

Each module has a doc block which includes a "Tested platforms" section.

For example

The above doc block tells you that this particular module has not yet benn tested on any platforms. This is a fairly safe bet that the module is not complete yet.

Therefore, we would not recommend filing any bugs against this module.

Let's take a look at another module though.

This module specifies the versions of BIG-IP that it has been tested against. Therefore, this module can probably be considered "complete" and ready for use by anyone.

It is these modules that we will entertain bug reports on.

CHAPTER 4

Module Index

All Modules

bigip_command - Run arbitrary command on F5 devices.

New in version 2.4.

Synopsis

- Sends an arbitrary command to an BIG-IP node and returns the results read from the device. This module includes an argument that will cause the module to wait for a specific condition before returning or timing out if the condition is not met.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: run show version on remote devices
  bigip_command:
    commands: show sys version
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: run show version and check to see if output contains BIG-IP
  bigip_command:
    commands: show sys version
    wait_for: result[0] contains BIG-IP
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: run multiple commands on remote nodes
  bigip_command:
    commands:
      - show sys version
      - list ltm virtual
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: run multiple commands and evaluate the output
  bigip_command:
    commands:
      - show sys version
      - list ltm virtual
    wait_for:
      - result[0] contains BIG-IP
      - result[1] contains my-vs
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: tmsh prefixes will automatically be handled
  bigip_command:
    commands:
      - show sys version
      - tmsh list ltm virtual
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_config - Manage BIG-IP configuration sections.

New in version 2.4.

Synopsis

- Manages a BIG-IP configuration by allowing TMSH commands that modify running configuration, or merge SCF formatted files into the running configuration. Additionally, this module is of significant importance because it allows you to save your running configuration to disk. Since the F5 module only manipulate running configuration, it is important that you utilize this module to save that running config.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Save the running configuration of the BIG-IP
  bigip_config:
    save: yes
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: Reset the BIG-IP configuration, for example, to RMA the device
  bigip_config:
    reset: yes
    save: yes
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost

- name: Load an SCF configuration
  bigip_config:
    merge_content: "{{ lookup('file', '/path/to/config.scf') }}"
    server: "lb.mydomain.com"
    password: "secret"
    user: "admin"
    validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_configsync_actions - Perform different actions related to config-sync.

New in version 2.4.

Synopsis

- Allows one to run different config-sync actions. These actions allow you to manually sync your configuration across multiple BIG-IPs when those devices are in an HA pair.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Sync configuration from device to group
  bigip_configsync_actions:
      device_group: "foo-group"
      sync_device_to_group: yes
      server: "lb01.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: no
  delegate_to: localhost

- name: Sync configuration from most recent device to the current host
  bigip_configsync_actions:
      device_group: "foo-group"
```

```
      sync_most_recent_to_device: yes
      server: "lb01.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: no
  delegate_to: localhost

- name: Perform an initial sync of a device to a new device group
  bigip_configsync_actions:
      device_group: "new-device-group"
      sync_device_to_group: yes
      server: "lb01.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: no
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the objectpath Python package on the host. This is as easy as pip install objectpath.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_connectivity - Manages device IP configuration settings for HA on a BIG-IP.

New in version 2.4.

Synopsis

- Manages device IP configuration settings for HA on a BIG-IP. Each BIG-IP device has synchronization and failover connectivity information (IP addresses) that you define as part of HA pairing or clustering. This module allows you to configure that information.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Configure device connectivity for standard HA pair
  bigip_device_connectivity:
      config_sync_ip: "10.1.30.1"
      mirror_primary_address: "10.1.30.1"
      unicast_failover:
          - address: "10.1.30.1"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- This module is primarily used as a component of configuring HA pairs of BIG-IP devices.
- Requires BIG-IP >= 12.1.x.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_dns - Manage BIG-IP device DNS settings

New in version 2.2.

Synopsis

- Manage BIG-IP device DNS settings

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set the DNS settings on the BIG-IP
  bigip_device_dns:
      name_servers:
          - 208.67.222.222
          - 208.67.220.220
      search:
          - localdomain
          - lab.local
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_group - Manage device groups on a BIG-IP.

New in version 2.4.

Synopsis

- Managing device groups allows you to create HA pairs and clusters of BIG-IP devices. Usage of this module should be done in conjunction with the `bigip_configsync_actions` to sync configuration across the pair or cluster if auto-sync is disabled.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create a sync-only device group
  bigip_device_group:
      name: "foo-group"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Create a sync-only device group with auto-sync enabled
  bigip_device_group:
      name: "foo-group"
      auto_sync: "yes"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- This module is primarily used as a component of configuring HA pairs of BIG-IP devices.
- Requires BIG-IP >= 12.1.x.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_ntp - Manage NTP servers on a BIG-IP.

New in version 2.2.

Synopsis

- Manage NTP servers on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set NTP server
  bigip_device_ntp:
      ntp_servers:
          - "192.0.2.23"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Set timezone
  bigip_device_ntp:
      password: "secret"
      server: "lb.mydomain.com"
      timezone: "America/Los_Angeles"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_sshd - Manage the SSHD settings of a BIG-IP.

New in version 2.2.

Synopsis

- Manage the SSHD settings of a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set the banner for the SSHD service from a string
  bigip_device_sshd:
      banner: "enabled"
      banner_text: "banner text goes here"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Set the banner for the SSHD service from a file
  bigip_device_sshd:
      banner: "enabled"
      banner_text: "{{ lookup('file', '/path/to/file') }}"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Set the SSHD service to run on port 2222
  bigip_device_sshd:
      password: "secret"
      port: 2222
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host This is as easy as pip install f5-sdk.
- Requires BIG-IP version 12.0.0 or greater

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_device_trust - Manage the trust relationships between BIG-IPs.

New in version 2.5.

Synopsis

- Manage the trust relationships between BIG-IPs.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record - Manage DNS resource records on a BIG-IP

New in version 2.2.

Synopsis

- Manage DNS resource records on a BIG-IP

Requirements (on host that executes module)

- bigsuds
- distutils

Options

Examples

```
- name: Add an A record to organization.com zone
  bigip_dns_record:
      user: "admin"
      password: "secret"
      hostname: "lb.mydomain.com"
      type: "A"
      zone: "organization.com"
      state: "present"
      options:
          hostname: "elliot.organization.com"
          ip_address: "10.1.1.1"
  delegate_to: localhost
```

```
- name: Add an A record to organization.com zone
  local_action:
      module: bigip_dns_record
      user: "admin"
      password: "secret"
      hostname: "lb.mydomain.com"
      type: "A"
      zone: "organization.com"
      state: "present"
      ttl: "10"
      options:
          domain_name: "elliot.organization.com"
          ip_address: "10.1.1.1"
```

Notes

Note:

- Requires the bigsuds Python package on the remote host. This is as easy as pip install bigsuds

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record_facts - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk

---

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_zone - Manages DNS zones on a BIG-IP

New in version 2.2.

Synopsis

- This module manages DNS zones described in the iControl Management documentation

Requirements (on host that executes module)

- bigsuds
- distutils

Options

Examples

```
- name: Add a view, named "internal", to organization.com zone
  local_action:
      module: bigip_view
      username: 'admin'
      password: 'admin'
      hostname: 'bigip.organization.com'
      zone_names:
          - 'organization.com'
      state: 'present'
      options:
          - domain_name: elliot.organization.com
          ip_address: 10.1.1.1
```

Notes

Note:

- Requires the bigsuds Python package on the remote host. This is as easy as pip install bigsuds
- https://devcentral.f5.com/wiki/iControl.Management__Zone.ashx

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record_facts - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk

---

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_facts - Collect facts from F5 BIG-IP devices

New in version 1.6.

Synopsis

- Collect facts from F5 BIG-IP devices via iControl SOAP API

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: Collect BIG-IP facts
  bigip_facts:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      include: "interface,vlan"
  delegate_to: localhost
```

Notes

Note:

- Requires BIG-IP software version >= 11.4
- F5 developed module 'bigsuds' required (see http://devcentral.f5.com)
- Best run as a local_action in your playbook
- Tested with manager and above account privilege level
- `provision` facts were added in 2.2

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_gtm_datacenter - Manage Datacenter configuration in BIG-IP

New in version 2.2.

Synopsis

- Manage BIG-IP data center configuration. A data center defines the location where the physical network components reside, such as the server and link objects that share the same subnet on the network. This module is able to manipulate the data center definitions in a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Create data center "New York"
  bigip_gtm_datacenter:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
```

```
    name: "New York"
    location: "222 West 23rd"
delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_gtm_facts - Collect facts from F5 BIG-IP GTM devices.

New in version 2.3.

Synopsis

- Collect facts from F5 BIG-IP GTM devices.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Get pool facts
  bigip_gtm_facts:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      include: "pool"
      filter: "my_pool"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk

---

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_gtm_pool - Manages F5 BIG-IP GTM pools.

New in version 2.4.

Synopsis

- Manages F5 BIG-IP GTM pools.

Requirements (on host that executes module)

- f5-sdk
- netaddr

Options

Examples

```
- name: Create a GTM pool
  bigip_gtm_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_pool"
  delegate_to: localhost

- name: Disable pool
  bigip_gtm_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "disabled"
      name: "my_pool"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_gtm_virtual_server - Manages F5 BIG-IP GTM virtual servers

New in version 2.2.

Synopsis

- Manages F5 BIG-IP GTM virtual servers

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: Enable virtual server
  local_action: >
    bigip_gtm_virtual_server
    server=192.0.2.1
    user=admin
```

```
    password=mysecret
    virtual_server_name=myname
    virtual_server_server=myserver
    state=enabled
```

Notes

Note:

- Requires BIG-IP software version >= 11.4
- F5 developed module 'bigsuds' required (see http://devcentral.f5.com)
- Best run as a local_action in your playbook
- Tested with manager and above account privilege level

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_gtm_wide_ip - Manages F5 BIG-IP GTM wide ip.

New in version 2.0.

Synopsis

- Manages F5 BIG-IP GTM wide ip.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set lb method
  bigip_gtm_wide_ip:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      lb_method: "round-robin"
      name: "my-wide-ip.example.com"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_hostname - Manage the hostname of a BIG-IP.

New in version 2.3.

Synopsis

- Manage the hostname of a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set the hostname of the BIG-IP
  bigip_hostname:
      hostname: "bigip.localhost.localdomain"
      password: "admin"
      server: "bigip.localhost.localdomain"
      user: "admin"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_iapp_service - Manages TCL iApp services on a BIG-IP.

New in version 2.4.

Synopsis

- Manages TCL iApp services on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk
- deepdiff

Options

Examples

```
- name: Create HTTP iApp service from iApp template
  bigip_iapp_service:
      name: "foo-service"
      template: "f5.http"
      parameters: "{{ lookup('file', 'f5.http.parameters.json') }}"
      password: "secret"
      server: "lb.mydomain.com"
```

```
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Upgrade foo-service to v1.2.0rc4 of the f5.http template
  bigip_iapp_service:
      name: "foo-service"
      template: "f5.http.v1.2.0rc4"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Configure a service using parameters in YAML
  bigip_iapp_service:
      name: "tests"
      template: "web_frontends"
      password: "admin"
      server: "{{ inventory_hostname }}"
      server_port: "{{ bigip_port }}"
      validate_certs: "{{ validate_certs }}"
      state: "present"
      user: "admin"
      parameters:
          variables:
              - name: "var__vs_address"
                value: "1.1.1.1"
              - name: "pm__apache_servers_for_http"
                value: "2.2.2.1:80"
              - name: "pm__apache_servers_for_https"
                value: "2.2.2.2:80"
  delegate_to: localhost

- name: Re-configure a service whose underlying iApp was updated in place
  bigip_iapp_service:
      name: "tests"
      template: "web_frontends"
      password: "admin"
      force: yes
      server: "{{ inventory_hostname }}"
      server_port: "{{ bigip_port }}"
      validate_certs: "{{ validate_certs }}"
      state: "present"
      user: "admin"
      parameters:
          variables:
              - name: "var__vs_address"
                value: "1.1.1.1"
              - name: "pm__apache_servers_for_http"
                value: "2.2.2.1:80"
              - name: "pm__apache_servers_for_https"
                value: "2.2.2.2:80"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the deepdiff Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_iapp_template - Manages TCL iApp templates on a BIG-IP.

New in version 2.4.

Synopsis

- Manages TCL iApp templates on a BIG-IP. This module will allow you to deploy iApp templates to the BIG-IP and manage their lifecycle. The conventional way to use this module is to import new iApps as needed or by extracting the contents of the iApp archive that is provided at downloads.f5.com and then importing all the iApps with this module. This module can also update existing iApps provided that the source of the iApp changed while the name stayed the same. Note however that this module will not reconfigure any services that may have been created using the `bigip_iapp_service` module. iApps are normally not updated in production. Instead, new versions are deployed and then existing services are changed to consume that new template. As such, the ability to update templates in-place requires the `force` option to be used.

Options

Examples

```
- name: Add the iApp contained in template iapp.tmpl
  bigip_iapp_template:
      content: "{{ lookup('template', 'iapp.tmpl') }}"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Update a template in place
  bigip_iapp_template:
      content: "{{ lookup('template', 'iapp-new.tmpl') }}"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Update a template in place that has existing services created from␣
↪it.
  bigip_iapp_template:
      content: "{{ lookup('template', 'iapp-new.tmpl') }}"
      force: yes
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record_facts - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk
- https://localhost:10443/mgmt/shared/iapp/global-installed-packages/

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_irule - Manage iRules across different modules on a BIG-IP.

New in version 2.2.

Synopsis

- Manage iRules across different modules on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Add the iRule contained in template irule.tcl to the LTM module
  bigip_irule:
      content: "{{ lookup('template', 'irule.tcl') }}"
      module: "ltm"
      name: "MyiRule"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Add the iRule contained in static file irule.tcl to the LTM module
  bigip_irule:
      module: "ltm"
      name: "MyiRule"
      password: "secret"
      server: "lb.mydomain.com"
      src: "irule.tcl"
```

```
    state: "present"
    user: "admin"
delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_license - Manage license installation and activation on BIG-IP devices

New in version 2.2.

Synopsis

- Manage license installation and activation on BIG-IP devices

Requirements (on host that executes module)

- bigsuds
- requests
- suds
- paramiko

Options

Examples

```
- name: License BIG-IP using default license options
  bigip_license:
      server: "big-ip.domain.org"
      user: "admin"
      password: "MyPassword123"
      key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
  delegate_to: localhost

- name: License BIG-IP, specifying license options
  bigip_license:
      server: "big-ip.domain.org"
      key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
      user: "admin"
      password: "MyPassword123"
      license_options:
          email: 'joe.user@myplace.com'
          firstname: 'Joe'
          lastname: 'User'
          company: 'My Place'
          phone: '630-555-1212'
          jobtitle: 'Systems Administrator'
          address: '207 N Rodeo Dr'
          city: 'Beverly Hills'
          state: 'CA'
          postalcode: '90210'
          country: 'US'
  delegate_to: localhost

- name: Remove the license from the system
  bigip_license:
      server: "big-ip.domain.org"
      user: "admin"
      password: "MyPassword123"
      state: "absent"
  delegate_to: localhost

- name: Update the current license of the BIG-IP
  bigip_license:
      server: "big-ip.domain.org"
      user: "admin"
      password: "MyPassword123"
      key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
      state: "latest"
  delegate_to: localhost
```

Notes

Note:

- Requires the suds Python package on the host. This is as easy as pip install suds
- Requires the bigsuds Python package on the host. This is as easy as pip install bigsuds
- Requires the paramiko Python package on the host if using the `state absent`. This is as easy as pip install paramiko
- Requires the requests Python package on the host if using the `state absent`. This is as easy as pip install paramiko

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_monitor_http - Manages F5 BIG-IP LTM http monitors

New in version 1.4.

Synopsis

- Manages F5 BIG-IP LTM monitors via iControl SOAP API

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: BIGIP F5 | Create HTTP Monitor
  bigip_monitor_http:
      state: "present"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_http_monitor"
      send: "http string to send"
      receive: "http string to receive"
  delegate_to: localhost

- name: BIGIP F5 | Remove HTTP Monitor
  bigip_monitor_http:
    state: "absent"
    server: "lb.mydomain.com"
    user: "admin"
    password: "secret"
    name: "my_http_monitor"
  delegate_to: localhost
```

Notes

Note:

- Requires BIG-IP software version >= 11
- F5 developed module 'bigsuds' required (see http://devcentral.f5.com)
- Best run as a local_action in your playbook
- Monitor API documentation: https://devcentral.f5.com/wiki/iControl.LocalLB__Monitor.ashx

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_monitor_tcp - Manages F5 BIG-IP LTM tcp monitors.

New in version 1.4.

Synopsis

- M
- a
- n
- a
- g
- e
- s
-
- F
- 5
-
- B
- I
- G
- –
- I
- P
-
- L
- T
- M
-
- t
- c

- p
-
- m
- o
- n
- i
- t
- o
- r
- s
-
- v
- i
- a
-
- i
- C
- o
- n
- t
- r
- o
- l
-
- S
- O
- A
- P
-
- A
- P
- I
- .

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create TCP Monitor
  bigip_monitor_tcp:
      state: "present"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
      type: "tcp"
      send: "tcp string to send"
      receive: "tcp string to receive"
  delegate_to: localhost

- name: Remove TCP Monitor
  bigip_monitor_tcp:
      state: "absent"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP software version >= 12

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_monitor_tcp_echo - Manages F5 BIG-IP LTM tcp monitors.

New in version 2.4.

Synopsis

- M
- a
- n
- a
- g
- e
- s
-
- F
- 5
-
- B
- I
- G
- –
- I
- P
-
- L
- T
- M
-

- t
- c
- p
-
- m
- o
- n
- i
- t
- o
- r
- s
-
- v
- i
- a
-
- i
- C
- o
- n
- t
- r
- o
- l
-
- S
- O
- A
- P
-
- A
- P
- I
- .

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create TCP Echo Monitor
  bigip_monitor_tcp_echo:
      state: "present"
      server: "lb.mydomain.com"
      user: "admin"
      ip: 10.10.10.10
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost

- name: Remove TCP Echo Monitor
  bigip_monitor_tcp_echo:
      state: "absent"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP software version >= 12

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_monitor_tcp_half_open - Manages F5 BIG-IP LTM tcp monitors.

New in version 2.4.

Synopsis

- M
- a
- n
- a
- g
- e
- s
-
- F
- 5
-
- B
- I
- G
- –
- I
- P
-
- L
- T
- M
-

- t
- c
- p
-
- m
- o
- n
- i
- t
- o
- r
- s
-
- v
- i
- a
-
- i
- C
- o
- n
- t
- r
- o
- l
-
- S
- O
- A
- P
-
- A
- P
- I
- .

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create TCP Monitor
  bigip_monitor_tcp_half_open:
      state: "present"
      ip: "10.10.10.10"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost

- name: Remove TCP Monitor
  bigip_monitor_tcp_half_open:
      state: "absent"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP software version >= 12

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_node - Manages F5 BIG-IP LTM nodes

New in version 1.4.

Synopsis

- Manages F5 BIG-IP LTM nodes via iControl SOAP API.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Add node
  bigip_node:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      partition: "Common"
      host: "10.20.30.40"
      name: "10.20.30.40"
  delegate_to: localhost

- name: Add node with a single 'ping' monitor
  bigip_node:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      partition: "Common"
      host: "10.20.30.40"
      name: "mytestserver"
      monitors:
          - /Common/icmp
  delegate_to: localhost
```

```
- name: Modify node description
  bigip_node:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      partition: "Common"
      name: "10.20.30.40"
      description: "Our best server yet"
  delegate_to: localhost

- name: Delete node
  bigip_node:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "absent"
      partition: "Common"
      name: "10.20.30.40"
  delegate_to: localhost

- name: Force node offline
  bigip_node:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "disabled"
      partition: "Common"
      name: "10.20.30.40"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires BIG-IP software version >= 11
- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_partition - Manage BIG-IP partitions.

New in version 2.5.

Synopsis

- Manage BIG-IP partitions.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create partition "foo" using the default route domain
  bigip_partition:
      name: "foo"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Create partition "bar" using a custom route domain
  bigip_partition:
      name: "bar"
      route_domain: 3
      password: "secret"
```

```
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Change route domain of partition "foo"
  bigip_partition:
      name: "foo"
      route_domain: 8
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Set a description for partition "foo"
  bigip_partition:
      name: "foo"
      description: "Tenant CompanyA"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Delete the "foo" partition
  bigip_partition:
      name: "foo"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      state: "absent"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP software version >= 12

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_policy - Manage general policy configuration on a BIG-IP.

New in version 2.4.

Synopsis

- Manages general policy configuration on a BIG-IP. This module is best used in conjunction with the `bigip_policy_rule` module. This module can handle general configuration like setting the draft state of the policy, the description, and things unrelated to the policy rules themselves. It is also the first module that should be used when creating rules as the `bigip_policy_rule` module requires a policy parameter.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
vars:
    policy_rules:
        - name: rule1
          actions:
              - forward: "yes"
                select: "yes"
                pool: "pool-svrs"
          conditions:
              - http_uri: "yes"
                path: "yes"
                starts-with:
                    - /euro
          ordinal: 8
        - name: HomePage
          actions:
              - forward: yes
                select: yes
```

```
                pool: "pool-svrs"
            conditions:
                - http-uri: yes
                  path: yes
                  starts-with:
                      - /HomePage/
            ordinal: 4

- name: Create policies
  bigip_policy:
      name: "Policy-Foo"
      state: present
  delegate_to: localhost

- name: Add a rule to the new policy
  bigip_policy_rule:
      policy: "Policy-Foo"
      name: "ABC"
      ordinal: 11
      conditions:
          - http_uri: "yes"
            path: "yes"
            starts_with:
                - "/ABC"
      actions:
          - forward: "yes"
            select: "yes"
            pool: "pool-svrs"

- name: Add multiple rules to the new policy
  bigip_policy_rule:
      policy: "Policy-Foo"
      name: "{{ item.name }}"
      ordinal: "{{ item.ordinal }}"
      conditions: "{{ item.conditions }}"
      actions: "{{ item.actions }}"
  with_items:
      - policy_rules
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_policy_rule - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk
- https://localhost:10443/mgmt/shared/iapp/global-installed-packages/

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_pool - Manages F5 BIG-IP LTM pools.

Synopsis

- Manages F5 BIG-IP LTM pools via iControl REST API.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Create pool
  bigip_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      name: "my-pool"
      partition: "Common"
      lb_method: "least_connection_member"
      slow_ramp_time: 120
  delegate_to: localhost

- name: Modify load balancer method
  bigip_pool:
```

```
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      name: "my-pool"
      partition: "Common"
      lb_method: "round_robin"
  delegate_to: localhost

- name: Add pool member
  bigip_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      name: "my-pool"
      partition: "Common"
      host: "{{ ansible_default_ipv4['address'] }}"
      port: 80
  delegate_to: localhost

- name: Remove pool member from pool
  bigip_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "absent"
      name: "my-pool"
      partition: "Common"
      host: "{{ ansible_default_ipv4['address'] }}"
      port: 80
  delegate_to: localhost

- name: Delete pool
  bigip_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "absent"
      name: "my-pool"
      partition: "Common"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires BIG-IP software version >= 11.
- F5 developed module 'F5-SDK' required (https://github.com/F5Networks/f5-common-python).
- Best run as a local_action in your playbook.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_pool_member - Manages F5 BIG-IP LTM pool members

New in version 1.4.

Synopsis

- Manages F5 BIG-IP LTM pool members via iControl SOAP API

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: Add pool member
  bigip_pool_member:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
```

```
      pool: "my-pool"
      partition: "Common"
      host: "{{ ansible_default_ipv4["address"] }}"
      port: 80
      description: "web server"
      connection_limit: 100
      rate_limit: 50
      ratio: 2
  delegate_to: localhost

- name: Modify pool member ratio and description
  bigip_pool_member:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      pool: "my-pool"
      partition: "Common"
      host: "{{ ansible_default_ipv4["address"] }}"
      port: 80
      ratio: 1
      description: "nginx server"
  delegate_to: localhost

- name: Remove pool member from pool
  bigip_pool_member:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "absent"
      pool: "my-pool"
      partition: "Common"
      host: "{{ ansible_default_ipv4["address"] }}"
      port: 80
  delegate_to: localhost


# The BIG-IP GUI doesn't map directly to the API calls for "Pool ->
# Members -> State". The following states map to API monitor
# and session states.
#
# Enabled (all traffic allowed):
# monitor_state=enabled, session_state=enabled
# Disabled (only persistent or active connections allowed):
# monitor_state=enabled, session_state=disabled
# Forced offline (only active connections allowed):
# monitor_state=disabled, session_state=disabled
#
# See https://devcentral.f5.com/questions/icontrol-equivalent-call-for-b-
↪node-down

- name: Force pool member offline
  bigip_pool_member:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      session_state: "disabled"
```

```
    monitor_state: "disabled"
    pool: "my-pool"
    partition: "Common"
    host: "{{ ansible_default_ipv4["address"] }}"
    port: 80
delegate_to: localhost
```

Notes

Note:

- Requires BIG-IP software version >= 11
- F5 developed module 'bigsuds' required (see http://devcentral.f5.com)
- Best run as a local_action in your playbook
- Supersedes bigip_pool for managing pool members

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_provision - Manage BIG-IP module provisioning.

New in version 2.4.

Synopsis

- Manage BIG-IP module provisioning. This module will only provision at the standard levels of Dedicated, Nominal, and Minimum.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Provision PEM at "nominal" level
  bigip_provision:
      server: "lb.mydomain.com"
      module: "pem"
      level: "nominal"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Provision a dedicated SWG. This will unprovision every other module
  bigip_provision:
      server: "lb.mydomain.com"
      module: "swg"
      password: "secret"
      level: "dedicated"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_qkview - Manage qkviews on the device.

New in version 2.4.

Synopsis

- Manages creating and downloading qkviews from a BIG-IP. Various options can be provided when creating qkviews. The qkview is important when dealing with F5 support. It may be required that you upload this qkview to the supported channels during resolution of an SRs that you may have opened.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Fetch a qkview from the remote device
  bigip_qkview:
      asm_request_log: "yes"
      exclude:
          - audit
          - secure
      dest: "/tmp/localhost.localdomain.qkview"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- This module does not include the "max time" or "restrict to blade" options.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_remote_syslog - Manipulate remote syslog settings on a BIG-IP.

New in version 2.5.

Synopsis

- Manipulate remote syslog settings on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.2.0

Options

Examples

```
- name: Add a remote syslog server to log to
  bigip_remote_syslog:
      remote_host: "10.10.10.10"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "false"
  delegate_to: localhost

- name: Add a remote syslog server on a non-standard port to log to
  bigip_remote_syslog:
      remote_host: "10.10.10.10"
      remote_port: "1234"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "false"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_routedomain - Manage route domains on a BIG-IP

New in version 2.2.

Synopsis

- Manage route domains on a BIG-IP

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Create a route domain
  bigip_routedomain:
      id: "1234"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Set VLANs on the route domain
  bigip_routedomain:
      id: "1234"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
      vlans:
          - net1
          - foo
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_routedomain_facts - Retrieve route domain attributes from a BIG-IP

New in version 2.2.

Synopsis

- Retrieve route domain attributes from a BIG-IP

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Get the facts for a route domain
  bigip_routedomain_facts:
      id: "1234"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_selfip - Manage Self-IPs on a BIG-IP system

New in version 2.2.

- *Status*
- *Support*

Synopsis

- Manage Self-IPs on a BIG-IP system

Requirements (on host that executes module)

- netaddr
- f5-sdk

Options

Examples

```
- name: Create Self IP
  bigip_selfip:
      address: "10.10.10.10"
      name: "self1"
      netmask: "255.255.255.0"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "no"
      vlan: "vlan1"
  delegate_to: localhost

- name: Create Self IP with a Route Domain
  bigip_selfip:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: "no"
      name: "self1"
      address: "10.10.10.10"
      netmask: "255.255.255.0"
      vlan: "vlan1"
      route_domain: "10"
      allow_service: "default"
  delegate_to: localhost

- name: Delete Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

```
- name: Allow management web UI to be accessed on this Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
      allow_service:
          - "tcp:443"
  delegate_to: localhost

- name: Allow HTTPS and SSH access to this Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
      allow_service:
          - "tcp:443"
          - "tpc:22"
  delegate_to: localhost

- name: Allow all services access to this Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
      allow_service:
          - all
  delegate_to: localhost

- name: Allow only GRE and IGMP protocols access to this Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
      allow_service:
          - gre:0
          - igmp:0
  delegate_to: localhost

- name: Allow all TCP, but no other protocols access to this Self IP
  bigip_selfip:
      name: "self1"
      password: "secret"
      server: "lb.mydomain.com"
      state: "absent"
      user: "admin"
      validate_certs: "no"
```

```
    allow_service:
        - tcp:0
delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_service - Manage BIG-IP service states

New in version 2.2.

Synopsis

- Manage BIG-IP service states

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: Restart the BIG-IP sshd service
  bigip_service:
      server: "big-ip"
      name: "sshd"
      user: "admin"
      password: "admin"
      state: "restarted"
  delegate_to: localhost
```

Notes

Note:

- Requires the bigsuds Python package on the host if using the iControl interface. This is as easy as pip install bigsuds

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_snat_pool - Manage SNAT pools on a BIG-IP.

New in version 2.3.

Synopsis

- Manage SNAT pools on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Add the SNAT pool 'my-snat-pool'
  bigip_snat_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my-snat-pool"
      state: "present"
      members:
          - 10.10.10.10
          - 20.20.20.20
  delegate_to: localhost

- name: Change the SNAT pool's members to a single member
  bigip_snat_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my-snat-pool"
      state: "present"
      member: "30.30.30.30"
  delegate_to: localhost

- name: Append a new list of members to the existing pool
  bigip_snat_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my-snat-pool"
      state: "present"
      members:
```

```
          - 10.10.10.10
          - 20.20.20.20
  delegate_to: localhost

- name: Remove the SNAT pool 'my-snat-pool'
  bigip_snat_pool:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "johnd"
      state: "absent"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_snmp - Manipulate general SNMP settings on a BIG-IP.

New in version 2.4.

Synopsis

- Manipulate general SNMP settings on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.2.0

Options

Examples

```
- name: Set snmp contact
  bigip_snmp:
      contact: "Joe User"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "false"
  delegate_to: localhost

- name: Set snmp location
  bigip_snmp:
      location: "US West 1"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "false"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record_facts - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_snmp_trap - Manipulate SNMP trap information on a BIG-IP.

New in version 2.4.

Synopsis

- Manipulate SNMP trap information on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.2.0

Options

Examples

```
- name: Create snmp v1 trap
  bigip_snmp_trap:
      community: "general"
      destination: "1.2.3.4"
      name: "my-trap1"
      network: "management"
      port: "9000"
      snmp_version: "1"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
  delegate_to: localhost
```

```
- name: Create snmp v2 trap
  bigip_snmp_trap:
      community: "general"
      destination: "5.6.7.8"
      name: "my-trap2"
      network: "default"
      port: "7000"
      snmp_version: "2c"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- This module only supports version v1 and v2c of SNMP.
- The `network` option is not supported on versions of BIG-IP < 12.1.0 because the platform did not support that option until 12.1.0. If used on versions < 12.1.0, it will simply be ignored.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_dns_record_facts - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_software - Manage BIG-IP software versions and hotfixes

New in version 2.4.

Synopsis

- Manage BIG-IP software versions and hotfixes

Requirements (on host that executes module)

- f5-sdk
- isoparser

Options

Examples

```
- name: Remove uploaded hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      state: "absent"
  delegate_to: localhost

- name: Upload hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      state: "present"
  delegate_to: localhost

- name: Remove uploaded base image
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      state: "absent"
  delegate_to: localhost

- name: Upload base image
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
```

```
      state: "present"
  delegate_to: localhost

- name: Upload base image and hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      state: "present"
  delegate_to: localhost

- name: Remove uploaded base image and hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      state: "absent"
  delegate_to: localhost

- name: Install (upload, install) base image. Create volume if not exists
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      volume: "HD1.1"
      state: "installed"
  delegate_to: localhost

- name: Install (upload, install) base image and hotfix. Create volume if␣
↪not exists
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      volume: "HD1.1"
      state: "installed"
  delegate_to: localhost

- name: Activate (upload, install, reboot) base image. Create volume if not␣
↪exists
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      volume: "HD1.1"
      state: "activated"
  delegate_to: localhost

- name: Activate (upload, install, reboot) base image and hotfix. Create␣
↪volume if not exists
```

```
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      volume: "HD1.1"
      state: "activated"
  delegate_to: localhost

- name: Activate (upload, install, reboot) base image and hotfix. Reuse␣
↪inactive volume in volumes with prefix.
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "/root/BIGIP-11.6.0.0.0.401.iso"
      hotfix: "/root/Hotfix-BIGIP-11.6.0.3.0.412-HF3.iso"
      reuse_inactive_volume: yes
      state: "activated"
  delegate_to: localhost

- name: Activate (download, install, reboot, reuse_inactive_volume) base␣
↪image and hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso"
      hotfix_md5sum: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso.md5"
      software: "http://fake.com/BIGIP-12.1.2.0.0.249.iso"
      software_md5sum: "http://fake.com/BIGIP-12.1.2.0.0.249.iso.md5"
      build: "1.0.271"
      version: "12.1.2"
      remote_src: "yes"
      state: "activated"
      reuse_inactive_volume: True
  delegate_to: localhost

- name: Download hotfix image
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso"
      hotfix_md5sum: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso.md5"
      build: "1.0.271"
      version: "12.1.2"
      remote_src: "yes"
      state: "present"
  delegate_to: localhost

- name: Remove uploaded hotfix image
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso"
```

```
      build: "1.0.271"
      version: "12.1.2"
      remote_src: "yes"
  delegate_to: localhost

- name: Install (download, install) base image
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "http://fake.com/BIGIP-12.1.2.0.0.249.iso"
      software_md5sum: "http://fake.com/BIGIP-12.1.2.0.0.249.iso.md5"
      build: "0.0.249"
      version: "12.1.2"
      remote_src: "yes"
      volume: "HD1.1"
      state: "installed"
  delegate_to: localhost

- name: Install (download, install) base image and hotfix
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso"
      hotfix_md5sum: "http://fake.com/Hotfix-12.1.2.1.0.271-HF1.iso.md5"
      software: "http://fake.com/BIGIP-12.1.2.0.0.249.iso"
      software_md5sum: "http://fake.com/BIGIP-12.1.2.0.0.249.iso.md5"
      build: "1.0.271"
      version: "12.1.2"
      remote_src: "yes"
      state: "installed"
      volume: "HD1.2"
   delegate_to: localhost

   - name: Download hotfix image (name mismatch)
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      hotfix: "http://fake.com/12.1.2-HF1.iso"
      hotfix_md5sum: "http://fake.com/Hotfix-12.1.2HF1.md5"
      build: "1.0.271"
      version: "12.1.2"
      remote_src: "yes"
      state: "present"
  delegate_to: localhost

- name: Download software image (name mismatch)
  bigip_software:
      server: "bigip.localhost.localdomain"
      user: "admin"
      password: "admin"
      software: "http://fake.com/BIGIP-12.1.2.iso"
      software_md5sum: "http://fake.com/12.1.2.md5"
      build: "0.0.249"
      version: "12.1.2"
      remote_src: "yes"
```

```
    state: "present"
delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk
- Requires the isoparser Python package on the host. This can be installed with pip install isoparser
- Requires the lxml Python package on the host. This can be installed with pip install lxml

---

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_software_update - Manage the software update settings of a BIG-IP.

New in version 2.4.

Synopsis

- Manage the software update settings of a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host This is as easy as pip install f5-sdk

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_ssl_certificate - Import/Delete certificates from BIG-IP.

New in version 2.2.

– *Support*

Synopsis

- This module will import/delete SSL certificates on BIG-IP LTM. Certificates can be imported from certificate and key files on the local disk, in PEM format.

Requirements (on host that executes module)

- f5-sdk >= 1.5.0
- BIG-IP >= v12

Options

Examples

```
- name: Import PEM Certificate from local disk
  bigip_ssl_certificate:
      name: "certificate-name"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      cert_src: "/path/to/cert.crt"
      key_src: "/path/to/key.key"
  delegate_to: localhost

- name: Use a file lookup to import PEM Certificate
  bigip_ssl_certificate:
      name: "certificate-name"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      cert_content: "{{ lookup('file', '/path/to/cert.crt') }}"
      key_content: "{{ lookup('file', '/path/to/key.key') }}"
  delegate_to: localhost

- name: "Delete Certificate"
  bigip_ssl_certificate:
      name: "certificate-name"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "absent"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- This module does not behave like other modules that you might include in roles where referencing files or templates first looks in the role's files or templates directory. To have it behave that way, use the Ansible file or template lookup (see Examples). The lookups behave as expected in a role context.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_static_route - Manipulate static routes on a BIG-IP.

New in version 2.3.

Synopsis

- Manipulate static routes on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.2.3

Options

Examples

```
- name: Create static route with gateway address
  bigip_static_route:
      destination: "10.10.10.10"
      gateway_address: "10.2.2.3"
      name: "test-route
      password: "secret"
      server: "lb.mydomain.come"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_sys_db - Manage BIG-IP system database variables

New in version 2.2.

Synopsis

- Manage BIG-IP system database variables

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Set the boot.quiet DB variable on the BIG-IP
  bigip_sys_db:
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "boot.quiet"
      value: "disable"
  delegate_to: localhost

- name: Disable the initial setup screen
  bigip_sys_db:
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "setup.run"
      value: "false"
  delegate_to: localhost

- name: Reset the initial setup screen
  bigip_sys_db:
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "setup.run"
      state: "reset"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP version 12.0.0 or greater

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_sys_global - Manage BIG-IP global settings.

New in version 2.3.

Synopsis

- Manage BIG-IP global settings.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Disable the setup utility
  bigip_sys_global:
      gui_setup: "disabled"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      state: "present"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_ucs - Manage upload, installation and removal of UCS files.

New in version 2.4.

- *Status*
- *Support*

Synopsis

- Manage upload, installation and removal of UCS files.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Upload UCS
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "/root/bigip.localhost.localdomain.ucs"
      state: "present"
  delegate_to: localhost

- name: Install (upload, install) UCS.
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "/root/bigip.localhost.localdomain.ucs"
      state: "installed"
  delegate_to: localhost

- name: Install (upload, install) UCS without installing the license portion
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "/root/bigip.localhost.localdomain.ucs"
      state: "installed"
      no_license: "yes"
  delegate_to: localhost

- name: Install (upload, install) UCS except the license, and bypassing the␣
↪platform check
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "/root/bigip.localhost.localdomain.ucs"
      state: "installed"
      no_license: "yes"
      no_platform_check: "yes"
```

```
  delegate_to: localhost

- name: Install (upload, install) UCS using a passphrase necessary to load␣
↪the UCS
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "/root/bigip.localhost.localdomain.ucs"
      state: "installed"
      passphrase: "MyPassphrase1234"
  delegate_to: localhost

- name: Remove uploaded UCS file
  bigip_ucs:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      ucs: "bigip.localhost.localdomain.ucs"
      state: "absent"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Only the most basic checks are performed by this module. Other checks and considerations need to be taken into account. See the following URL. https://support.f5.com/kb/en-us/solutions/public/11000/300/sol11318.html
- This module does not handle devices with the FIPS 140 HSM
- This module does not handle BIG-IPs systems on the 6400, 6800, 8400, or 8800 hardware platform.
- This module does not verify that the new or replaced SSH keys from the UCS file are synchronized between the BIG-IP system and the SCCP
- This module does not support the 'rma' option
- This module does not support restoring a UCS archive on a BIG-IP 1500, 3400, 4100, 6400, 6800, or 8400 hardware platform other than the system from which the backup was created
- The UCS restore operation restores the full configuration only if the hostname of the target system matches the hostname on which the UCS archive was created. If the hostname does not match, only the shared configuration is restored. You can ensure hostnames match by using the `bigip_hostname` Ansible module in a task before using this module.
- This module does not support re-licensing a BIG-IP restored from a UCS
- This module does not support restoring encrypted archives on replacement RMA units.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_ucs_fetch - Fetches a UCS file from remote nodes

New in version 2.4.

Synopsis

- This module is used for fetching UCS files from remote machines and storing them locally in a file tree, organized by hostname. Note that this module is written to transfer UCS files that might not be present, so a missing remote UCS won't be an error unless fail_on_missing is set to 'yes'.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Download a new UCS
  bigip_ucs_fetch:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      src: "cs_backup.ucs"
      dest: "/tmp/cs_backup.ucs"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- BIG-IP provides no way to get a checksum of the UCS files on the system via any interface except, perhaps, logging in directly to the box (which would not support appliance mode). Therefore, the best this module can do is check for the existence of the file on disk; no checksumming.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_user - Manage user accounts and user attributes on a BIG-IP.

New in version 2.4.

Synopsis

- Manage user accounts and user attributes on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Add the user 'johnd' as an admin
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      username_credential: "johnd"
      password_credential: "password"
      full_name: "John Doe"
      partition_access: "all:admin"
      update_password: "on_create"
      state: "present"
  delegate_to: localhost

- name: Change the user "johnd's" role and shell
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      username_credential: "johnd"
      partition_access: "NewPartition:manager"
      shell: "tmsh"
      state: "present"
  delegate_to: localhost

- name: Make the user 'johnd' an admin and set to advanced shell
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "johnd"
      partition_access: "all:admin"
      shell: "bash"
      state: "present"
  delegate_to: localhost

- name: Remove the user 'johnd'
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "johnd"
      state: "absent"
  delegate_to: localhost

- name: Update password
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
```

```
      state: "present"
      username_credential: "johnd"
      password_credential: "newsupersecretpassword"
  delegate_to: localhost

# Note that the second time this task runs, it would fail because
# The password has been changed. Therefore, it is recommended that
# you either,
#
#   * Put this in its own playbook that you run when you need to
#   * Put this task in a `block`
#   * Include `ignore_errors` on this task
- name: Change the Admin password
  bigip_user:
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      state: "present"
      username_credential: "admin"
      password_credential: "NewSecretPassword"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP versions >= 12.0.0

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_user_facts - Retrieve user account attributes from a BIG-IP

New in version 2.2.

Synopsis

- Retrieve user account attributes from a BIG-IP

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Gather facts about user 'johnd'
  bigip_user_facts:
      name: "johnd"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Display the user facts
  debug:
    var: bigip
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk
- Facts are placed in the `bigip` variable

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_view - Manage ZoneRunner Views on a BIG-IP

New in version 2.2.

Synopsis

- Manage ZoneRunner Views on a BIG-IP. ZoneRunner is a feature of the GTM module. Therefore, this module should only be used on BIG-IP systems that have the GTM module enabled. The SOAP connection has a number of known limitations when it comes to updating Views. It is only possible to

Requirements (on host that executes module)

- bigsuds

Options

Examples

```
- name: Create a view foo.local
  local_action:
      module: "bigip_view"
      user: "admin"
```

```
      password: "admin"
      name: "foo.local"

- name: Assign zone "bar" to view "foo.local"
  local_action:
      module: "bigip_view"
      user: "admin"
      password: "admin"
      name: "foo.local"
      zones:
          - "bar"
```

Notes

Note:

- Requires the bigsuds Python package on the remote host. This is as easy as pip install bigsuds

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_virtual_address - Manage LTM virtual addresses on a BIG-IP.

New in version 2.4.

Synopsis

- Manage LTM virtual addresses on a BIG-IP.

Options

Examples

```
- name: Add virtual address
  bigip_virtual_address:
      server: "lb.mydomain.net"
      user: "admin"
      password: "secret"
      state: "present"
      partition: "Common"
      address: "10.10.10.10"
  delegate_to: localhost

- name: Enable route advertisement on the virtual address
  bigip_virtual_address:
      server: "lb.mydomain.net"
      user: "admin"
      password: "secret"
      state: "present"
      address: "10.10.10.10"
      use_route_advertisement: yes
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_virtual_server - Manage LTM virtual servers on a BIG-IP.

New in version 2.1.

Synopsis

- Manage LTM virtual servers on a BIG-IP

Requirements (on host that executes module)

- f5-sdk
- netaddr

Options

Examples

```
- name: Add virtual server
  bigip_virtual_server:
      server: lb.mydomain.net
      user: admin
      password: secret
      state: present
      partition: Common
      name: my-virtual-server
      destination: "10.10.10.10"
      port: 443
      pool: "my-pool"
      snat: Automap
      description: Test Virtual Server
      profiles_both:
          - http
          - fix
      profiles_server_side:
```

```
          - clientssl
      profiles_client_side:
          - ilx
      enabled_vlans:
          - /Common/vlan2
  delegate_to: localhost

- name: Modify Port of the Virtual Server
  bigip_virtual_server:
      server: lb.mydomain.net
      user: admin
      password: secret
      state: present
      partition: Common
      name: my-virtual-server
      port: 8080
  delegate_to: localhost

- name: Delete virtual server
  bigip_virtual_server:
      server: lb.mydomain.net
      user: admin
      password: secret
      state: absent
      partition: Common
      name: my-virtual-server
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires BIG-IP software version >= 11
- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_vlan - Manage VLANs on a BIG-IP system

New in version 2.2.

Synopsis

- Manage VLANs on a BIG-IP system

Requirements (on host that executes module)

- f5-sdk

Options

Examples

```
- name: Create VLAN
  bigip_vlan:
      name: "net1"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Set VLAN tag
  bigip_vlan:
      name: "net1"
      password: "secret"
      server: "lb.mydomain.com"
      tag: "2345"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

```
- name: Add VLAN 2345 as tagged to interface 1.1
  bigip_vlan:
      tagged_interface: 1.1
      name: "net1"
      password: "secret"
      server: "lb.mydomain.com"
      tag: "2345"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Add VLAN 1234 as tagged to interfaces 1.1 and 1.2
  bigip_vlan:
      tagged_interfaces:
          - 1.1
          - 1.2
      name: "net1"
      password: "secret"
      server: "lb.mydomain.com"
      tag: "1234"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires BIG-IP versions >= 12.0.0

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigiq_license_pool - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk
- https://localhost:10443/mgmt/shared/iapp/global-installed-packages/

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigiq_license_pool_member - foo

New in version 2.2.

Synopsis

- foo

Requirements (on host that executes module)

- f5-sdk

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk
- https://localhost:10443/mgmt/shared/iapp/global-installed-packages/

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_partition - Manage BIG-IP partitions

New in version 2.3.

Synopsis

- Manage BIG-IP partitions

Requirements (on host that executes module)

- bigsuds
- requests

Options

Examples

Notes

Note:

- Requires the bigsuds Python package on the host if using the iControl interface. This is as easy as pip install bigsuds

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_managed_device - Manipulate cloud managed devices in iWorkflow.

New in version 2.4.

Synopsis

- Manipulate cloud managed devices in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 1.5.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Discover a BIG-IP device with hostname lb.mydomain.com
  iworkflow_device:
      device: "lb.mydomain.com
      username_credential: "admin"
      password_credential: "admin"
      password: "secret"
      server: "mgmt.mydomain.com"
```

```
    user: "admin"
delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_iapp_template - Manages iApp templates

New in version 2.4.

Synopsis

- Manages TCL iApp services on a BIG-IP.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Add AppSvcs Integration to iWorkflow
  iworkflow_iapp_template:
      device: "my-bigip-1"
      template_content: "{{ lookup('file', 'appsvcs_integration_v2.0_001.tmpl
↪') }}"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Remove AppSvcs Integration from iWorkflow
  iworkflow_iapp_template:
      name: "appsvcs_integration_v2.0_001"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_license - Manage license of iWorkflow itself.

New in version 2.4.

- *Synopsis*

Synopsis

- Manage license of iWorkflow itself.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: License iWorkflow with a base license
  iworkflow_license:
      accept_eula: "yes"
      base_key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
      state: "present"
      server: "iwf.mydomain.com"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: License iWorkflow and provide add-on keys
  iworkflow_license:
      accept_eula: "yes"
      base_key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
      add_on_keys:
          - YYYYY-YYYYY-YYYYY-YYYYY-YYYYYYY
          - ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZZZ
      state: "present"
      server: "iwf.mydomain.com"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_license_pool - Manage license pools in iWorkflow.

New in version 2.4.

Synopsis

- Manage license pools in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Create license pool
  iworkflow_license_pool:
      accept_eula: "yes"
      name: "my-lic-pool"
      base_key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
      state: "present"
      server: "iwf.mydomain.com"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_license_pool_member - Manages members in a license pool.

New in version 2.4.

Synopsis

- Manages members in a license pool. By adding and removing members from a pool, you will implicitly be licensing and un-licensing them.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_bigip_connector - Manipulate cloud BIG-IP connectors in iWorkflow.

New in version 2.4.

Synopsis

- Manipulate cloud BIG-IP connectors in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Create cloud connector named Private Cloud
  iworkflow_bigip_connector:
      name: "Private Cloud"
      password: "secret"
      server: "iwf.mydomain.com"
      user: "admin"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_local_connector_device - Manipulate cloud local connector devices in iWorkflow.

New in version 2.4.

Synopsis

- Manipulate cloud local connector devices in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_local_connector_node - Manages L2/L3 configuration of a BIG-IP via iWorkflow.

New in version 2.3.

Synopsis

- Manages L2/L3 configuration of a BIG-IP via iWorkflow. This module is useful in the event that you have a new BIG-IP that does not yet have VLANs and Self-IPs configured on it. You can use this module on a discovered, managed, device to configure those settings via iWorkflow. You **do not** need touse this module if you have an existing BIG-IP that has its L2/L3 configuration already complete. In that cae, it is sufficient to just use the `iworkflow_managed_device` module and iWorkflow will automatically discover the node information for you.

Requirements (on host that executes module)

- f5-sdk >= 2.2.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Create node from managed device
  iworkflow_local_connector_node:
      device: "10.144.128.137"
      password_credential: "secret"
      username_credential: "admin"
      state: "present"
```

```
      connector: "Private OpenStack"
      hostname: "lb1.example.com"
      interfaces:
          - local_address: "10.144.128.137"
            subnet_address: "10.144.128/24"
          - local_address: "10.2.0.81"
            subnet_address: "10.2.0.0/24"
            name: "internal"
      server: "iwf.mydomain.com"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost

- name: Create node from managed device in Azure
  iworkflow_local_connector_node:
      device: "10.144.128.137"
      password_credential: "secret"
      username_credential: "admin"
      device_root_password: "default"
      state: "present"
      connector: "Public Azure West US"
      hostname: "lb1.example.com"
      interfaces:
          - local_address: "10.0.2.12"
            subnet_address: "10.0.2.0/24"
            virtual_address: "10.144.128.137"
          - local_address: "10.2.0.81"
            subnet_address: "10.2.0.0/24"
            name: "external"
      server: "iwf.mydomain.com"
      password: "secret"
      user: "admin"
      validate_certs: "no"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Requires the netaddr Python package on the host. This is as easy as pip install netaddr.
- This module does not support updating of existing nodes that were created with a `cli_password_credential`. The onboarding process will change your device's `cli_username_credential` password, which will prevent you from using this module (without knowing the password) a second time.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_service - Manages L4/L7 Services on iWorkflow.

New in version 2.4.

Synopsis

- Manages L4/L7 Service on iWorkflow. Services can only be created and otherwise managed by tenants on iWorkflow. Since all of the F5 modules assume the use of the administrator account, the user of this module will need to include the `tenant` option if they want to use this module with the admin account.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk.
- L4/L7 Services cannot be updated once they have been created. Instead, you must first delete the service and then re-create it.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_service_template - Manages Service Templates on iWorkflow.

New in version 2.4.

Synopsis

- Manages Service Templates on iWorkflow. Service templates are created by the iWorkflow administrator and are consumed by iWorkflow tenants in the form of L4/L7 services. The Service Template can be configured to allow tenants to change certain values of the template such as the IP address of a VIP, or the port that a Virtual Server listens on.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the remote host. This is as easy as pip install f5-sdk
- Requires the deepdiff Python package on the Ansible controller host. This is as easy as pip install deepdiff.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_system_setup - Manage system setup related configuration on iWorkflow

New in version 2.4.

Synopsis

- Manage system setup related configuration on iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.2.2
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Disable iWorkflow setup screen and set accounts as unchanged
  iworkflow_system_setup:
      is_admin_password_changed: "no"
      is_root_password_changed: "no"
      is_system_setup: "yes"
      password: "secret"
      server: "iwf.mydomain.com"
      user: "admin"
  delegate_to: localhost
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Required the netaddr Python package on the host. This is as easy as pip install netaddr.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_tenant - Manage tenants in iWorkflow.

New in version 2.4.

– *Support*

Synopsis

- Manage tenants in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Tenants are not useful unless you associate them with a connector using the `iworkflow_tenant_connector` module.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_tenant_connector - Manage connectors associated with tenants in iWorkflow.

New in version 2.4.

Synopsis

- Manage connectors associated with tenants in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

```
- name: Register connector to tenant
  iworkflow_tenant_connector:
      tenant: "tenant-foo"
      connector: "connector-foo"
      server: "iwf.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: "no"
      state: "present"

- name: Register multiple connectors to tenant
  iworkflow_tenant_connector:
      tenant: "tenant-foo"
      connector: "{{ item }}"
      server: "iwf.mydomain.com"
      user: "admin"
      password: "secret"
      validate_certs: "no"
      state: "present"
  with_items:
      - "connector-one"
      - "connector-two"

- name: Unregister connector from tenant
  iworkflow_tenant_connector:
      tenant: "tenant-foo"
      connector: "connector-foo"
```

```
    server: "iwf.mydomain.com"
    user: "admin"
    password: "secret"
    validate_certs: "no"
    state: "absent"
```

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.
- Tenants are not useful unless you associate them with a connector using the `iworkflow_tenant_connector` module.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

iworkflow_user - Manage users in iWorkflow.

New in version 2.4.

Synopsis

- Manage users in iWorkflow.

Requirements (on host that executes module)

- f5-sdk >= 2.3.0
- iWorkflow >= 2.1.0

Options

Examples

Notes

Note:

- Requires the f5-sdk Python package on the host. This is as easy as pip install f5-sdk.

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

bigip_partition - Manage BIG-IP partitions

New in version 2.3.

Synopsis

- Manage BIG-IP partitions

Requirements (on host that executes module)

- bigsuds
- requests

Options

Examples

Notes

Note:

- Requires the bigsuds Python package on the host if using the iControl interface. This is as easy as pip install bigsuds

---

Status

This module is flagged as **preview** which means that it is not guaranteed to have a backwards compatible interface.

Support

This module is community maintained without core committer oversight.

For more information on what this means please read modules_support

For help in developing on modules, should you be so inclined, please read community, dev_guide/developing_test_pr and dev_guide/developing_modules.

Note:

- (D): This marks a module as deprecated, which means a module is kept for backwards compatibility but usage is discouraged. The module documentation details page may explain more about this rationale.

Network Modules

F5

Note:

- (D): This marks a module as deprecated, which means a module is kept for backwards compatibility but usage is discouraged. The module documentation details page may explain more about this rationale.

CHAPTER 5

F5 Networks Contributor License Agreement

Before you start contributing to any project sponsored by F5 Networks, Inc. (F5) on GitHub, you will need to sign a Contributor License Agreement (CLA).

If you are signing as an individual, we recommend that you talk to your employer (if applicable) before signing the CLA since some employment agreements may have restrictions on your contributions to other projects. Otherwise by submitting a CLA you represent that you are legally entitled to grant the licenses recited therein.

If your employer has rights to intellectual property that you create, such as your contributions, you represent that you have received permission to make contributions on behalf of that employer, that your employer has waived such rights for your contributions, or that your employer has executed a separate CLA with F5.

If you are signing on behalf of a company, you represent that you are legally entitled to grant the license recited therein. You represent further that each employee of the entity that submits contributions is authorized to submit such contributions on behalf of the entity pursuant to the CLA.

Click the link below to download the PDF:

`F5 Contributor License Agreement (CLA)`

CHAPTER 6

Getting involved

So you want to get involved with this project. Great!

Becoming involved in this project can take a variety of angles. The maintainers spend their time camping this repository as well as contributing upstream to the Ansible core product. We can be reached on either forum.

Before you jump feet first in to coding, let me give you some idea of the workload involved in module development. This is not intended to scare you away, but only to set your expectations.

What is expected from you

First and foremost, we expect you to *bring a good attitude*. These modules are developed out of our own personal interests in Ansible; there is no official team here.

With that said, we want to keep the environment from getting bogged down in a bad mood.

The development of a module will require some effort on your part. Often times the countless reviews might seem frustrating, but believe us, we do it for good reason.

By becoming involved with developing a module, you're accepting that your contribution will probably not be accepted right off the bat. If you're willing to work with us though, and understand the direction we're headed in, you will more than likely find your module landing here.

We expect that you *stay up to date* with the documentation of this site concerning module development. As time goes on, things change and new practices are adopted. As this happens, we try to keep the documentation up to date with these changes.

If you develop a module use an out-of-date convention, we will tell you so upon review. It is then expected that you take the initiative to fix it.

Part of Ansible's requirements for this is that the people listed in the author field (usually those who wrote the module) take responsibility for the ongoing maintenance and support of the module.

Understandably this might be a big issue for you, so we are offering to assist. When your module is merged to our repo here, we lists ourselves as one of the authors of the code.

With this in place, and with the addition of us opening the PR with Ansible upstream, we think this will be sufficient to meet their needs for ongoing maintenance. This is a joint effort though, so lets work together to ensure that the module stays in Core. Modules that can no longer be supported by their authors are removed from Ansible.

What to work on

While module development is the primary focus of most contributors, it's understandable that you may not know how to write Python, or may not have any interest in writing code to begin with.

That's ok. Here are some existing things you can do to assist.

Documentation

This is always needed. I write documentation that focuses on many of the moving parts here, but I can't cover it all and will inevitably miss things.

Submitting documentation improvements is encouraged.

Unit tests

The unit tests in the *test/* directory can always use more love. Unit tests run fast and so more of them is not a burden on the test runner.

Add more test cases for your particular usage scenarios or any scenarios that may have been missed.

I personally only add enough unit tests to be reasonably comfortable that the code will execute right. This, unfortunately, does not cover many of the functional test cases. So writing unit test versions of functional tests is of huge benefit.

New module ideas

We don't have modules to cover all of the ways our products are used. If you find that a module is missing from the repo and you think needs to be added, I will entertain those ideas on the Github Issues page

New functionality for an existing module

Even the existing modules do not cover all the bells and whistles that customers use.

If a module is missing a parameter that you think it should have, raise and issue and we will consider it.

Postman collections

The Ansible modules make use of the F5 Python SDK for all of their work. In the SDK, all work is accomplished via the product REST APIs and this just happens to fit in perfectly with the tool Postman.

If you want to help us work on new modules without involving yourself in Python code, a great way to start is to write Postman collections for the APIs that configure on the BIG-IP what you want to configure.

If you provide us with the Postman collections, this makes it really easy for us to write the Ansible module itself.

This is the approach that many of the F5 teams who do not work in software land all day long take because it is super effective. Bonus points for collections that address differences in APIs between versions of BIG-IP

Finding bugs (via usage)

Using the modules is the best way to iron out bugs. By using the modules in the way that **you** expect them to work is a great way to find bugs.

During the development process, we write tests with specific user personas in mind. Your usage patterns may not reflect those personas though and that might break the module.

Using the modules is the best way to get both code and documentation correct. If it's not obvious to you via the documentation about how a module works, then I guarantee it is unclear to many more people.

Righting those wrongs helps you and future users.

More example playbooks

Playbooks show people how to make use of the module when paired with other modules. Playbooks also are the way that people inevitably use all these modules, so if you write playbooks that make use of them, this allows people to copy/paste the actual usage for their own benefit.

More roles for f5devcentral

Ansible's role features provide the opportunity for us to build new sets of configuration for an F5 product.

There has been some effort to begin writing role that would be useful for different scenarios (such as the *bigip-hardening* role) but this effort has not been fully tackle yet.

Anyone can write roles as they are just collections of files, templates, modules, and tasks that you are already writing for some purpose.

Writing more roles helps spread the usage of the modules.

More ways if you're at F5

If you're an F5 employee, there are even more ways to help. Refer to the *go/ansible* link for more details.

Conclusion

One final thing that will require effort on your part that, frankly, we cannot help with, is that of endorsement.

The Ansible work here is by no means supported by F5. If you want that to change, then you will need to initiate that change. Speaking with your SEs, AMs, SAs, etc etc, is the best way to drive that change.

When we run our mouths about orchestration tools, it falls on deaf ears. If this is valuable to your organization, then say so.

CHAPTER 7

Guidelines

Guidelines are written for you, the contributor, to understand what we expect from our contributors and to provide guidance on the direction we are taking the modules.

Getting Started

Since Ansible 2.2, it is a requirement that all new F5 modules are written to use the `f5-sdk`.

Prior to 2.2, modules may of been written in `bigsuds` or `requests` (for SOAP and REST respectively). Modules written using `bigsuds` can continue to be extended using it.

Backward compatibility of older modules should be maintained, but because `bigsuds` and `f5-sdk` can co-exist, it is recommended that all future features be written using `f5-sdk`.

BIG-IP's have two API interfaces; SOAP and REST. As a general rule of thumb, the SOAP API should be considered deprecated. While this is not an official stance from F5, there is clearly more of a push in the REST direction than the SOAP direction.

New functionality to the SOAP interface continues to be added, but only under certain circumstances.

Bug fixing

If you are writing a bugfix for a module that uses `bigsuds`, you should continue to use `bigsuds` to maintain backward compatibility.

If you are adding new functionality to an existing module that uses `bigsuds` but the new functionality requires `f5-sdk`, you may add it using `f5-sdk`.

Naming your module

Base the name of the module on the part of BIG-IP that the modules manipulates. (A good rule of thumb is to refer to the API being used in the `f5-sdk`).

Don't further abbreviate names - if something is a well known abbreviation due to it being a major component of BIG-IP, that's fine, but don't create new ones independently (e.g. LTM, GTM, ASM, etc. are fine)

Adding new features

If a module that you need does not yet exist, it is equally likely that the REST API in the f5-sdk has also not yet been developed. Please refer to the following github project

- https://github.com/F5Networks/f5-common-python

Open an Issue with that project to add the necessary APIs so that a proper Ansible module can be written to use them.

Using f5-sdk

The following guidelines pertain to how you should use the f5-sdk in the modules that you develop. We'll focus on the most common scenarios that you will encounter.

Importing

Wrap `import` statements in a try block and fail the module later if the import fails.

f5-sdk

```
try:
    from f5.bigip import ManagementRoot
    from f5.bigip.contexts import TransactionContextManager
    HAS_F5SDK = True
except ImportError:
    HAS_F5SDK = False

def main():

   if not HAS_F5SDK:
      module.fail_json(msg='f5-sdk required for this module')
```

Connecting to a BIG-IP

To connect to a BIG-IP, you should use the instantiate a *ManagementRoot* object, providing the credentials and options you wish to use for connecting.

REST

An example of connecting to big-ip01.internal is shown below.

```
from f5.bigip import ManagementRoot
from f5.bigip.contexts import TransactionContextManager

mr = ManagementRoot("localhost", "admin", "admin", port='10443')
tx = mr.tm.transactions.transaction
```

```
with TransactionContextManager(tx) as api:
    virt = api.tm.ltm.virtuals.virtual.load(name='asdf')
    tcp = virt.profiles_s.profiles.load(name='tcp')
    tcp.delete()
    virt.profiles_s.profiles.create(name='wom-tcp-wan-optimized')
```

Exception Handling

If an exception is thrown, it is up to you decide how to handle it but usually calling *fail_json* with the error message will suffice.

For raising exceptions you can use the exception class, *F5ModuleError*, provided with the *f5-sdk*. It can be used as such.

```
try:
    from f5.bigip import ManagementRoot
    HAS_F5SDK = True
except ImportError:
    HAS_F5SDK = False

# Connect to BIG-IP
...

# Make a call to BIG-IP
try:
    result = api.tm.ltm.pools.pool.create(foo='bar')
except F5ModuleError as e:
    module.fail_json(msg=e.message)
```

Helper functions

The helper functions available to you are included in the Ansible f5.py module_utils.

Code compatibility

The python code underlying the Ansible modules should be written to be compatible with both Python 2.7 and 3.

The travis configuration contained in this repo will verify that your modules are compatible with both versions. Use the following cheat-sheet to write compatible code.

- http://python-future.org/compatible_idioms.html

Automated testing

It is recommended that you use the testing facilities that we have paired with this repository. When you open PR's, our testing tools will run the PR against supported BIG-IP versions in our testing facilities.

By doing using our test harnesses, you do not need to have your own devices or VE instances to do your testing (although if you do that's fine).

We currently have the following devices in our test harness

- BIG-IP VE 11.6.0
- BIG-IP VE 12.0.0
- BIG-IP VE 12.1.0

CHAPTER 8

Architecture

CHAPTER 9

Code Conventions

The F5 modules attempt to follow a set of coding conventions that apply to all new and existing modules.

These conventions help new contributors quickly develop new modules. Additionally, they help existing contributors maintain the current modules.

Style checking

Where possible, we try to automate the validation of these coding conventions so that you are aware of mistakes and able to fix them yourself without having to have the maintainers intervene.

For more information on what tools perform these checks, refer to the tests page.

Conventions

When writing your modules and their accompanying tests and docs, please follow the below coding conventions.

Use the complex/structure map format

In reference to Jeff Geerling's page here, this format looks like this.

```
- name: Create a UCS
  bigip_ucs_fetch:
      dest: "/tmp/{{ ucs_name }}"
      password: "{{ bigip_password }}"
      server: "{{ inventory_hostname }}"
      src: "{{ ucs_name }}"
      user: "{{ bigip_username }}"
      validate_certs: "{{ validate_certs }}"
  register: result
```

There are several reasons that we use this format. Among them are Geerling's reasons.

- The structure is all valid YAML, using the structured list/map syntax mentioned in the beginning of this post.
- Strings, booleans, integers, octals, etc. are all preserved (instead of being converted to strings).
- Each parameter must be on its own line, so you can't chain together `mode: 0755, owner: root, user: root` to save space.
- YAML syntax highlighting works slightly better for this format than `key=value`, since each key will be highlighted, and values will be displayed as constants, strings, etc.

In addition to those reasons, there are also some situations that, if you use the simple `key=value` format will raise syntax errors. Finally, it saves on space and, in the maintainers opinion, is easier to read and know (by looking) what the arguments to the module are.

Alphabetize your module parameters

The parameters to your modules in the Roles and Playbooks that are developed here must be in alphabetic order.

GOOD

```
- name: My task
  bigip_module:
      alpha: "foo"
      beta: "bar"
      gamma: "baz"
```

BAD

```
- name: My task
  bigip_module:
      alpha: "foo"
      gamma: "baz"
      beta: "bar"
```

This provides for consistency amongst module usage as well as provides a way to see at a glance if a module has the correct parameters.

Double-quotes for Strings, no quotes for Numbers

Ansible supports a simple form of typing for your parameters. If there is a value that is a string, it should be represented as a string using double quotes.

GOOD

```
- name: My task
  bigip_module:
      alpha: "foo"
      beta: "bar"
```

BAD

```
- name: My task
  bigip_module:
      alpha: foo
      beta: bar
```

For numeric characters, you should not use any quotes because this can cause some modules to raise 'type' errors if the expected value is a number and you provide it with a number wrapped in quotes

GOOD

```
- name: My task
  bigip_module:
      alpha: 1
      beta: 100
```

BAD

```
- name: My task
  bigip_module:
      alpha: "1"
      beta: "100"
```

Begin YAML files with a triple-dash

A YAML file usually begins with three dashes. As such, you should have that be a part of your own YAML files.

GOOD

```
---

- name: My task
  bigip_module:
      alpha: 1
      beta: 100
```

BAD

```
- name: My task
  bigip_module:
      alpha: "1"
      beta: "100"
```

All tasks should have a name

When your Playbooks encounter errors, the name of the task is always called out in the failure. If you do not provide a name, then Ansible creates a name for you using the module call itself.

Naming your tasks allows you to quickly reference where a failure occurred.

GOOD

```
- name: My task
  bigip_module:
      alpha: 1
      beta: 100
```

BAD

```
- bigip_module:
      alpha: "1"
      beta: "100"
```

All modules must have a DOCUMENTATION variable

The DOCUMENTATION variable is also required by Ansible upstream as it serves as the source of the module documentation that is generated on their site.

Good documentation is essential to people being able to use the module so it must be included.

GOOD

```
DOCUMENTATION = '''
---
module: bigip_device_ntp
short_description: Manage NTP servers on a BIG-IP
description:
  - Manage NTP servers on a BIG-IP
version_added: "2.1"
options:
...
'''
```

BAD

```
Missing DOCUMENTATION variable
```

All modules must have an EXAMPLES variable

Useful and valid examples are crucial for people new to Ansible and to the module itself.

When providing examples, be mindful of what you provide. If you developed the module with a specific use case in mind, be sure to include that use case. It may be applicable to a large majority of users and, therefore, may eliminate a significant portion of their time that they would otherwise spend figuring out what is or is not needed.

GOOD

```
EXAMPLES = '''
- name: Set the banner for the SSHD service from a string
  bigip_device_sshd:
      banner: "enabled"
      banner_text: "banner text goes here"
      password: "admin"
      server: "bigip.localhost.localdomain"
      user: "admin"
  delegate_to: localhost
'''
```

BAD

```
Missing EXAMPLES variable
```

All modules must have a RETURN variable

The RETURN variable provides documentation essential to determining what, if any, information is returned by the operation of the module.

End users of the module will reference this documentation when they want to use the `register` keyword.

The `RETURN` field should include the parameters that have been changed by your module. If nothing has been changed, then no values need be returned.

GOOD

```
RETURN = '''
full_name:
    description: Full name of the user
    returned: changed
    type: string
    sample: "John Doe"
'''
```

BAD

```
Missing RETURN variable
```

If your module does not return any information, then an empty YAML string is sufficient

GOOD

..code-block:: python

    RETURN = ‘’‘# ’’‘

The author field must be a list

There is a good possibility that multiple people will work to maintain the module over time, so it is a good idea to make the `author` keyword in your module a list.

GOOD

```
author:
  - Tim Rupp (@caphrim007)
```

BAD

```
author: Tim Rupp (@caphrim007)
```

Author field should be Github handle

Both Ansible and this repository are maintained on Github. Therefore, for maintenance reasons we require your Github handle. Additionally, your email address may change over time.

GOOD

```
author:
  - Tim Rupp (@caphrim007)
```

BAD

```
author:
  - Tim Rupp <caphrim007@gmail.com>
```

Use 2 spaces in the DOCUMENTATION, EXAMPLES, and RETURN

This is a simple spacing convention to ensure that everything is properly spaced over.

GOOD

```
options:
  server:
    description:
      - BIG-IP host
    required: true
  user:
^^
```

BAD

```
options:
    server:
        description:
            - BIG-IP host
        required: true
    user:
^^^^
```

Use ansible lookup plugins where appropriate

Ansible provides existing facilities that can be used to read in file contents to a module's parameters.

If your module can accept a string or a file containing a string, then assume that users will be using the lookup plugins.

For example, SSL files are typically strings. SSH keys are also strings even if they are contained in a file. Therefore, you would delegate the fetching of the string data to a lookup plugin.

There should be no need to use the python `open` facility to read in the file.

GOOD

```
some_module:
    string_param: "{{ lookup('file', '/path/to/file') }}"
```

BAD

```
some_module:
    param: "/path/to/file"
```

Always expand lists in the various documentation variables

When listing examples or documentation in any of the following variables,

- DOCUMENTATION
- RETURN
- EXAMPLES

be sure to always expand lists of values if that key takes a list value.

GOOD

```
options:
  state:
    description:
      - The state of things
    choices:
      - present
      - absent
```

BAD

```
options:
  state:
    description:
      - The state of things
    choices: ['enabled', 'disabled']
```

Support for 12.0.0 or greater at this time

In the `DOCUMENTATION` section notes, you should specify what version of BIG-IP the module requires.

At this time, that version is 12.0.0, so your `DOCUMENTATION` string should reflect that.

GOOD

```
notes:
  - Requires BIG-IP version 12.0.0 or greater
```

BAD

```
Any version less than 12.0.0.
```

If your module requires functionality greater than 12.0.0 it is also acceptable to specify that in the `DOCUMENTATION` block.

Never raise a general Exception

General Exceptions are bad because they hide unknown errors from you, the developer. If a bug report comes in and is being caused by an exception that you do not handle, it will be exceedingly difficult to debug it.

Instead, only catch the *F5ModuleError* exception that is provided by the *f5-sdk*. Specifically raise this module and handle those errors. If an unknown error occurs, a full traceback will be produced that will more easily allow you to debug the problem.

GOOD

```
try:
    // do some things here that can cause an Exception
except bigsuds.OperationFailed as e:
    raise F5ModuleError('Error on setting profiles : %s' % e)
```

GOOD

```
if foo:
    // assume something successful happens here
else:
    raise F5ModuleError('Error on baz')
```

BAD

```
try:
    // do some things here that can cause an Exception
except bigsuds.OperationFailed as e:
    raise Exception('Error on setting profiles : %s' % e)
```

BAD

```
if foo:
    // assume something successful happens here
else:
    raise Exception('Error on baz')
```

All modules must support check mode

Check-mode allows Ansible to run your Playbooks in a dry-mode sort of operation. This is very handy when you want to run a set of tasks but are not sure what will happen when you do.

Since BIG-IPs are usually considered a sensitive device to handle, there should always be a check-mode implemented in your module.

Do not use local_action in your EXAMPLES

Some folks like local_action and some folks like delegation. Delegation is more applicable to general-purpose Ansible, so for that reason I want to get people in the habit of using and understanding it.

Therefore, do not use *local_action* when defining examples. Instead, use *delegate_to*.

GOOD

```
- name: Reset the initial setup screen
  bigip_sys_db:
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "setup.run"
      state: "reset"
  delegate_to: localhost
```

BAD

```
- name: Reset the initial setup screen
  local_action:
      module: "bigip_sys_db"
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "setup.run"
      state: "reset"
```

Default EXAMPLE parameters

For consistency, always using the following values for the given parameters

- user: “admin”
- password: “secret”
- server: “lb.mydomain.com”

This allows you to not have to overthink the inclusion of your example.

GOOD

```
- name: Reset the initial setup screen
  bigip_sys_db:
      user: "admin"
      password: "secret"
      server: "lb.mydomain.com"
      key: "setup.run"
      state: "reset"
  delegate_to: localhost
```

BAD

```
- name: Reset the initial setup screen
  bigip_sys_db:
      user: "joe_user"
      password: "admin"
      server: "bigip.host"
      key: "setup.run"
      state: "reset"
  delegate_to: localhost
```

Assign before returning

To enable easier debugging when something goes wrong, ensure that you assign values **before** you return those values.

GOOD

```
def exists(self):
    result = self.client.api.tm.gtm.pools.pool.exists(
        name=self.want.name,
        partition=self.want.partition
    )
    return result
```

BAD

```
def exists(self):
    return self.client.api.tm.gtm.pools.pool.exists(
        name=self.want.name,
        partition=self.want.partition
    )
```

The reason that the above **BAD** example is considered bad is that when it comes time to debug the value of a variable, it requires that you change the code to do an assignment operation anyway.

For example, using *q* to debug the value of the above requires that you implicitly assign the value of the API call before you do this,

```
...
result = self.client.api....
q.q(result)
...
```

When the code does not do a assignment, then you are required to change the code before you are able to debug the code.

Fixed Github issues should have an associated issue-xxxxx.yaml file

When a developer takes on a new issue that requires changes to code to get working, these changes should be tested with a new functional test yaml file located in the module's *test/integration/PRODUCT/targets* directory.

For example.

Consider the Github Issue 59 which is relevant to the *bigip_virtual_server* module.

The developer needed to add new code to the module. So to verify that the new code is tested, the developer should add a new file to the module's *targets* directory here

- *test/functional/bigip/bigip_virtual_server/tasks*

The name of the file should be

- *issue-59.yaml*

And inside of the file should be any and all work that is required to,

- Setup the test
- Perform the test
- Teardown the test

Any issues that are reported on github should follow the same pattern, however the filenames of those modules should be

- *ansible-xxxxx.yaml*

So-as not to step on the numeric namespace that is used natively in the *f5-ansible* repository.

RETURN value when there is no return

The correct way to set a RETURN variable whose module has no returnable things is like this according to *bcoca*.

Excluding code from unit test coverage

Ansible's test runner makes use of *pytest*, so the acceptable way of excluding lines from code coverage is documented here.

- http://coverage.readthedocs.io/en/coverage-4.2/excluding.html

The cases where you would want to use this include the various **_on_device* and **_from_device* methods in modules that make direct calls to the remote BIG-IPs.

CHAPTER 10

Upstreaming

Upstreaming refers to opening a PR with the Ansible core product.

Our goal with this repository is to serve as an incubator for modules to mature. Eventually, we hope that the modules here will find their way to the upstream Ansible product (core or extras).

Incubating modules

Modules that are currently in incubation are named as normal modules are that you would find in the Ansible product. They are easily distinguished by their filename **not** including a leading underscore.

These modules can only be obtained through the installation steps outlined in the **'Installation'_** docs.

An incubating module may or may not be working at any point in time. While we prefer to not break any of them during development, we recognize that sometimes that is part of the process.

Note: Just because a module is incubation does not mean that it is unstable. Many modules remain in the incubator because there has not been sufficient interest from the community either internal or external to release them.

A module in incubation should have an associated Issue in Github so that its progress can be tracked and so that others do not repeat work.

Qualifications for upstreaming

A module is considered to be in a mature state once it has met all of the standards that have been established for it.

- http://docs.ansible.com/ansible/dev_guide/testing.html
- **'upstreaming requirements'_**
- **'coding conventions'_**

At that point in its development, we will make the request to upstream Ansible for the module's inclusion.

Releases

After a module is upstreamed, there will continue to be a period of time that must pass before it is released as part of the core Ansible product.

That release schedule is shown in the *ROADMAP files* near the top of each file.

Depending on what version is currently stable, upstreamed modules will be part of the next major stable release.

For example.

If 2.3 is the current stable version and a module was upstreamed to core, it would not appear as part of *pip install ansible* until version 2.4 was released.

You can get the modules before that point in time, but you must do so manually. The link to the **'stable modules is here'_**.

CHAPTER 11

Upstreaming Process

The upstreaming process is summarized below. Only one person should be concerned with upstreaming things.

Upstream Github template

Ansible provides an Issue template that will be shown to you automatically when you create a new PR in Github. You should fill out the various fields in it, making sure to include the following in one of the "Summary", "Description", or related fields.

Below is an example of what has been provided in the past and you should use it as an example of what you too should provide.

Including the extra information that shows your due diligence in writing and testing the module is important because it helps ensure the Ansible maintainers, and our customers, that the code has been written well.

Upstream Window

Generally speaking the upstreaming window is open each one week around the times of the Networking meeting. Here is the Networking team's schedule.

- https://github.com/ansible/community/blob/master/MEETINGS.md#wednesdays

During that time, you will need to comment on the Ansible Networking Groups issue tracker for new PRs. This can be found here.

- https://github.com/ansible/community/issues/110

Upstream Meeting

The Networking team will address your PRs at their weekly meeting, which you are expected to attend.

The meeting can be found on IRC at the below location

- Server: irc.freenode.net
- Channel: #ansible-devel

CHAPTER 12

Writing a Module

Let's explore what it takes to write a module using the provided guidelines and standards.

Getting Started

The first step is to decide what you will call your module. For this tutorial we will recreate the functionality of the `bigip_device_sshd` module as it provides good examples of the common idioms you will encounter when developing or maintaining modules.

Because this module already exists, let's just slightly change the name of our module to the following

```
bigip_device_ssh
```

This name will additionally prevent you from tab'ing to the existing sshd module.

Create the directory layout

There are a number of files and directories that need to be created to hold the various tests and validation code in addition to just your module.

To create the necessary directories and files, an executable file is available for you to use to set these directories up automatically.

```
$> ./scripts/stub-new-module.py stub --module=bigip_device_ssh
```

When it finishes running, you will have the necessary files available to begin working on your module.

The module file

The module file that gets created is located here

‘‘library/bigip_device_ssh.py

Let’s open that file to get started

License header

The first things you you will put in the file is the license header.

Here is the common license header.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
# Copyright 2016 F5 Networks Inc.
#
# This file is part of Ansible
#
# Ansible is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# Ansible is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with Ansible.  If not, see <http://www.gnu.org/licenses/>.
```

If the module under development is your original work, then you can include your name in the copyright above.

If you are only contributing an existing module, then it is not necessary to include a copyright line at the top. Instead, accepting our CLA is sufficient to get code merged into our branch.

The ANSIBLE_METADATA variable

This variable should be included first in your module. It specifies metadata for the module itself. It can always look the same. Here is it as would be defined in code.

```
ANSIBLE_METADATA = {'status': ['preview'],
                    'supported_by': 'community',
                    'version': '1.0'}
```

After the metadata variable comes the *DOCUMENTATION* variable.

The DOCUMENTATION variable

The next chunk of code that you will insert describes the module, what parameters is accepts, who the authors/maintainers are, its dependencies, etc.

Let’s look at the code that we will add to our module.

```
DOCUMENTATION = '''
---
module: bigip_device_sshd
```

```
short_description: Manage the SSHD settings of a BIG-IP
description:
  - Manage the SSHD settings of a BIG-IP
version_added: "2.4"
options:
  banner:
    description:
      - Whether to enable the banner or not
    required: false
    choices:
      - enabled
      - disabled
  banner_text:
    description:
      - Specifies the text to include on the pre-login banner that displays
        when a user attempts to login to the system using SSH
    required: false
  inactivity_timeout:
    description:
      - Specifies the number of seconds before inactivity causes an SSH
        session to log out
    required: false
  log_level:
    description:
      - Specifies the minimum SSHD message level to include in the system log
    choices:
      - debug
      - debug1
      - debug2
      - debug3
      - error
      - fatal
      - info
      - quiet
      - verbose
  login:
    description:
      - Specifies, when checked C(enabled), that the system accepts SSH
        communications
    required: false
  port:
    description:
      - Port that you want the SSH daemon to run on
    required: false
notes:
  - Requires the f5-sdk Python package on the host This is as easy as pip
    install f5-sdk
extends_documentation_fragment: f5
requirements:
  - f5-sdk
author:
  - Tim Rupp (@caphrim007)
'''
```

Most documentation variables have a common set of keys and only differ in the values of those keys.

The keys that one commonly finds are

- `module`

- short_description
- description
- version_added
- options
- notes
- requirements
- author
- extends_documentation_fragment

Note: The *extends_documentation_fragment* key is special as it is what will automatically inject the variables *user*, *password*, *server*, *server_port* and *validate_certs* into your documentation. It should be used for all modules.

The EXAMPLES variable

The examples variable contains the most common use cases for this module.

I personally think that setting of the banner will be the most common case, but future authors are free to add to my examples.

These examples will also serve as a basis for the functional tests that we will write shortly.

For this module, our EXAMPLES variable looks like this.

```
EXAMPLES = '''
- name: Set the banner for the SSHD service from a string
  bigip_device_sshd:
      banner: "enabled"
      banner_text: "banner text goes here"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Set the banner for the SSHD service from a file
  bigip_device_sshd:
      banner: "enabled"
      banner_text: "{{ lookup('file', '/path/to/file') }}"
      password: "secret"
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost

- name: Set the SSHD service to run on port 2222
  bigip_device_sshd:
      password: "secret"
      port: 2222
      server: "lb.mydomain.com"
      user: "admin"
  delegate_to: localhost
'''
```

This variable should be placed __after__ the `DOCUMENTATION` variable.

The examples that you provide should always have the following

delegate_to: localhost

The BIG-IP modules are intended to run on the Ansible controller only. The best practice is to use this *delegate_to:* here so that users get in the habit of using it

common args

The common args as as follow

- *password* should always be set to *secret*
- *server* should always be set to *lb.mydomain.com*
- *user* should always be set to *admin*

The RETURN variable

The pattern which we follow is that we always return what changed in the module's parameters when the module has finished running.

The parameters that I am referring to here are the ones that are not considered to be the "standard" parameters to the F5 modules. Some exceptions to this rule apply. For example, where the *state* variable contains more states than just *absent* and *present*, such as in the *bigip_virtual_server* module.

For our module these include,

- `banner`
- `banner_text`
- `inactivity_timeout`
- `log_level`
- `login`

The `RETURN` variable describes these values, specifies when they are returned and provides examples of what the values returned might look like.

When the Ansible module documentation is generated, these values are presented in the form of a table. Here is the RETURN variable that we would place in our module file.

The import block

The next section in our code is the block of code where our 'import's happen.

This code usually just involves importing the *module_util* helper library, but may also include imports of other libraries if you are working with legacy code.

For this module our import block is the following

```
from ansible.module_utils.f5_utils import *
```

Module class

The next block of code is the skeleton for our module's class. We encapsulate all of our module's code inside a class for easy testing as well as for code re-use outside of this module.

For example, there are cases where third-parties want to re-use this code outside of Ansible.

The module class is where the specifics of your code will be. There are, however, a number of commonalities across all modules. The code outlined below includes those commonalities and leaves the implementation details specific to the module to your interpretation.

```
class BigIpDeviceSshd(object):
    def __init__(self, *args, **kwargs):
        if not HAS_F5SDK:
            raise F5ModuleError("The python f5-sdk module is required")

        self.params = kwargs
        self.api = ManagementRoot(kwargs['server'],
                                  kwargs['user'],
                                  kwargs['password'],
                                  port=kwargs['server_port'])

    def present(self):
        pass

    def absent(self):
        pass

    def update(self):
        pass

    def read(self):
        pass

    def flush(self):
        pass
```

For modules where settings are actively added or removed from the system, the modules **must** provide `present` and `absent` methods respectively.

Additionally, modules usually include an `update` method for those cases where `present` is being performed, but the value already exists and only an attribute of the setting is being changed.

The `flush` method exists to encapsulate the running of the `absent`, `present`, and `update` modules and should include the appropriate checks of the `state` parameter to decide which method to call.

For the implementation specifics, you can refer to the existing module.

Connecting to Ansible

With the implementation details of the module complete, we move on to the code that hooks the module up to Ansible itself.

The main function

This code begins with the definition of the `main` function.

This code should be placed __after__ the definition of your class which you wrote earlier. Here is how we begin.

```
def main():
```

Argument spec and instantiation

Next, we generate the common argument spec using a utility method of Ansible.

```
argument_spec = f5_argument_spec()
```

With the `argument_spec` generated, we update the values in it to match the `options` we declared in our `DOCUMENTATION` variable earlier.

The values that you must specify here are, again, the ones that are **not** common to all F5 modules. Below is the code we need to update our `argument_spec`

```
meta_args = dict(
    allow=dict(required=False, default=None),
    banner=dict(required=False, default=None, choices=CHOICES),
    banner_text=dict(required=False, default=None),
    inactivity_timeout=dict(required=False, default=None, type='int'),
    log_level=dict(required=False, default=None, choices=LEVELS),
    login=dict(required=False, default=None, choices=CHOICES),
    port=dict(required=False, default=None, type='int')
)
argument_spec.update(meta_args)
```

After the `argument_spec` has been updated, we instantiate an instance of our class, providing the `argument_spec` and the value that indicates we support Check mode.

```
module = AnsibleModule(
    argument_spec=argument_spec,
    supports_check_mode=True
)
```

All F5 modules **must** support Check Mode as it allows an administrator to determine whether a change will be made or not when the module is run against their devices.

Try and module execution

The next block of code that is added is a general execution of your class.

We wrap this execution inside of a try...except statement to ensure that we handle know errors and bubble up known errors.

Never include a general Exception handler here because it will hide the details of an unknown exception that we require when debugging an unhandled exception.

```
try:
    obj = BigIpDeviceSshd(check_mode=module.check_mode, **module.params)
    result = obj.flush()

    module.exit_json(**result)
except F5ModuleError as e:
    module.fail_json(msg=str(e))
```

Common imports

The following imports are common to all of the F5 modules. The `f5` import provides you with the helper functions that create the `argument_spec`.

The `basic` import is replaced by Ansible itself and provides helper functions and classes used to create the `Module` object (among other things).

```
from ansible.module_utils.basic import *
from ansible.module_utils.f5_utils import *
```

Common running

The final two lines in your module inform Python to execute the module's code if the script being run is itself executable.

```
if __name__ == '__main__':
    main()
```

Due to the way that Ansible works, this means that the `main` function will be called when the module is sent to the remote device (or run locally) but will not be called if the module is imported.

You would import the module if you were using it outside of Ansible, or in some sort of test environment where you do not want the module to actually run.

Testing

Providing tests with your module is a crucial step for having it merged and subsequently pushed upstream. We rely heavily on testing.

In this section I will go in to detail on how our tests are organized and how you can write your own to ensure that your modules works as designed.

flake8

We make use of the `flake8` command to ensure that our modules meet certain coding standards and compatibility across Python releases.

You can run the flake8 tests via the `make` command

```
make flake8
```

Before submitting your own module, it is recommended that your module pass the *flake8* tests we ship with the repository. We will ask you to update your code to meet these requirements if it does not.

Functional tests

This is probably the most important part of testing, so let's go in to detail on this part.

Functional tests are required during module submission so that we (F5) and you, the developer, can agree that a module works on a particular platform.

We will test your module on a variety of platforms automatically when a new PR is submitted, and from there provide feedback if something does not fly.

Structure of tests

Test file stubs are created for you automatically when you stub a new module.

To best show you how testing works, we will reference an existing module that provides complete tests; *bigip_device_sshd*.

First, let's look at the layout of a set of tests. A test is composed of a role whose name matches the name of the module that is being tested.

This role is placed in the *roles/* directory.

So, for our example, our test role looks like this.

- *roles/bigip_device_sshd/*

Inside of this role is everything that you would associate with a normal role in ansible.

Consider the following examples.

- if your test requires static files be used, then a *files/* directory should be in your role.
- if your test requires template data (for example iRules) for its input, then a *templates/* directory should be in your role.
- all roles will perform some work to test the module, so a *tasks/* directory should be in your role.

Now let's dig in to what a test should look like.

Test content

The test itself will follow the pattern below.

- perform some operation with the module
- assert a value

All of the tests work like this, and it is a decent smoke test for all modules until such time as we take the testing further.

Here is an example of a test from the *bigip_device_sshd* module.

```
---

- name: Set the SSHD allow string to a specific IP
  bigip_device_sshd:
      allow:
          - "{{ allow[0] }}"
      user: "{{ bigip_username }}"
      password: "{{ bigip_password }}"
      server: "{{ inventory_hostname }}"
      server_port: "{{ bigip_port }}"
      validate_certs: "no"
  register: result

- name: Assert Set the SSHD allow string to a specific IP
  assert:
      that:
          - result|changed
```

As you can see, pretty straightforward.

We use the module and then we check that the result we *register* was changed.

Test variables

All of the tests have access to the following variables by default.

- *{{ bigip_password }}*
- *{{ bigip_port }}*
- *{{ bigip_username }}*
- *{{ inventory_hostname }}*

All other information specific to the tests that you need to run should be put in the *defaults/main.yaml* file of your test role.

By putting them there, you allow individuals to override values in your test by providing arguments to the CLI at runtime.

The idempotent test

All tests that change data should also include a test right after it that tries to perform the same test, but whose result is expected to *not* change.

These are called idempotent tests because they ensure that the module only changes settings if the setting needs to be changed.

Here is an example of the previous test as an idempotent test

```
- name: Set the SSHD allow string to a specific IP - Idempotent check
  bigip_device_sshd:
      allow:
          - "{{ allow[0] }}"
      user: "{{ bigip_username }}"
      password: "{{ bigip_password }}"
      server: "{{ inventory_hostname }}"
      server_port: "{{ bigip_port }}"
      validate_certs: "no"
  register: result

- name: Assert Set the SSHD allow string to a specific IP - Idempotent check
  assert:
      that:
          - not result|changed
```

There are two things to note here.

First, the test code itself is identical to the previous test.

Second, note that we changed the name of the test to include the string ‘‘“- Idempotent check”‘. This gives reviewers the ability to visually note that this is an idempotent test.

Third, note that in our assertion, we are check that the result has *not* changed. This is the important part because it is what ensures that the test itself was idempotent.

Now lets look at how you call the test.

Calling the test

To call the test and run it, this repo includes a *make* command that is available for all modules. The name of the make target is the name of your module.

So, for our example, that *make* command would be.

- make bigip_device_ssh

This command will run the module functional tests for you in debug mode.

Including supplementary information

If you include files inside of the *files/*, *templates*, or other directories in which the content of that file was auto-generated or pulled from a third party source, you should include a *README.md* file in your role's directory.

Inside of this file, you can include steps to reproduce any of the input items that you include in the role subdirectories.

In addition, this place is also a good location to include references to third party file locations if you have included them in the tests. For example, if you were to include iRules or other things that you downloaded and included from DevCentral or similar.

The *README.md* is there for future developers to reference the information needed to re-create any of the inputs to your tests in case they need to.

Other testing notes

When writing your tests, there is no need to be concerned about "undoing" what you previously have done to the test environment.

Between the running of the tests, we destroy the VMs that ran the test so for each running of the test you can assume a pristine environment.

CHAPTER 13

Deprecating Code

Our Ansible modules follow a strategy of deprecation which is intended to give the user of the modules ample time to upgrade to a new version of Ansible.

New releases of Ansible happen, at the time of this writing, about once a quarter.

With this in mind, the following process should allow a user 3 to 6 months to upgrade their Ansible installation to the new code. If the user misses this 3 to 6 month period, then they can upgrade incrementally (2.1 -> 2.2 -> 2.3) instead of upgrading directly to the latest version and, in the process, test that the incremental versions work with their playbooks.

Deprecation process

Let's look at an example deprecation process

- 2.0 - version to deprecate
- 2.1 - deprecated version
- 2.2 - version with deprecated feature removed

During the second release noted above, the module developer *MUST* insert adequate warnings for the user to see. Ansible will color warning messages so that they stick out from regular messages.

Raising deprecated warnings

To raise warnings about deprecated functionality, the module developer should add the following method to their *ModuleManager* class.

Additionally, the module developer should add the calling of that method when they collect the *changes* to report to the user. For example,

And finally, the module developer should populate the *__warnings* key of their *changes* attribute as needed. For example, in the *bigip_gtm_wide_ip* module, the following is used in the *lb_method* property when it sees you are using an option name that is deprecated. For example,

The *changes* attribute is typically updated during the call to *_update_changed_options* during *update*, or *_set_changed_options* during *create*.

If your module needs to detect changes outside of those two general methods, you should be doing so inside of the *should_update* method.

Note: To do your own ad-hoc detection inside of *should_update*, you will need to overload the base classes' method. If you do this, you should decide whether or not it is still necessary to call the base classes' method during the call to your overloaded method.

With that in place, you will find yourself with *warning* messages being raised by Ansible when you use the deprecated functionality.

Deprecating parameters

Below is an excerpt from how one might deprecate an option that we no longer want to use. You may do this for a number of reasons, but in most cases it is due to the original name not making sense in the context it is used.

For example, you might have named the original option *rules* when the more appropriate name for the option would have been *irules*.

Note: Ansible allows for aliasing of options so that specifying one is equivalent to specifying another. This is *not* the situation that we are referring to here. It is still perfectly acceptable to use option aliases if you want to. These guidelines are for when you specifically want to *remove* options that are presumably already in use.

Here is a sample *ArgumentSpec* from the version where we made the mistake. Let's assume that this mistake was made in version 2.0.

Now, we wish to deprecate that option name. So, in version 2.1 of Ansible, we would do something like this.

Additionally, we would include the warnings necessary to make the user of the module aware that they are using deprecated functionality (the *rules* option).

Finally, during the release cycle of Ansible 2.2, we would want to change our spec to look like so.

Thus, removing the deprecated functionality.

Also, do not forget to remove any mention of the deprecation inside the actual module code. We don't want the legacy code to stick around. This helps keep technical debt at bay.

CHAPTER 14

Code Conventions

The following is a document that describes some of the design decisions that went into the F5 Ansible modules and why they were made. This document is intended to address any contributor, or customer, questions on why I did what I did and the known limitations for this design.

SSH is not used. REST is

This is a question that comes up rather frequently.

> Why don't the modules use ssh?

After all, the "other" networking modules use SSH for their communication with their remote devices.

This is a complicated question to answer because there are __many__ reasons why this was decided upon. I will try to explain all of those reasons below.

TMSH is not an API

At F5, regardless of what you might here or read online, *tmsh* is not considered to be a formal API.

Now, there are some people who will try to argue about this and justify their argument by saying that, "well, tmsh is a publicly available way to interact with the BIG-IP, therefore it is implicitly an API".

While *tmsh* is indeed a publicly available way to interact with the BIG-IP, it is not considered by anyone at F5 to be an API. You will notice, compared to other APIs that are formal, that it has none of the features of "real" APIs.

In addition to missing many features of "normal" APIs, it also is not guaranteed to be consistent across versions of BIG-IP.

You should not rely on *tmsh* for anything except in cases where you have no other choice. And in those cases, you must handle the versioning of it yourself.

The company had decided to put all their effort into REST

When the REST pattern of API development became popular, F5 put some work into making a REST wrapper around *tmsh*. That is what you have today; a wrapper around calls to *tmsh*.

To understand why the REST API is where the company has put more effort, you need to understand the history of the SOAP API at F5.

It is surprisingly difficult for a team to develop for the SOAP API. The REST API is easier to code for. Additionally, the REST APIs curiously map almost directly to the *tmsh* command you would use. Coincidence? Hardly. Remember, it's *tmsh*.

There is still new functionality being added to SOAP, but most of the engineers at F5 are focused on providing REST functionality.

- It's natively built into most all programming languages
- It's more easily supported in our future javascript-based iAppLX effort
- It's pretty close to native *tmsh*

SSH on BIG-IP can cause auth errors when under heavy load

For better or worse, while SSH is about as native today as telnet was in the past, it actually takes a fair amount of resources to establish an SSH connection to a remote device.

You see this in Ansible today with their use of *ControlMaster* and *pipelining*. Ansible saw that just the act of repeatedly connecting to the remote device over SSH caused a significant amount of lag.

A real-life example, a customer had been using modules (that they had developed) that used SSH under the hood to connecto the BIG-IP and were seeing "F5 Authorization Required" 95% of the time. With SOAP this dropped to 4% and with REST this dropped to 1%.

When we investigated things further, we noticed that when the BIG-IP was under load, the normal behavior of Ansible, namely this,

```
sshpass -d57 ssh -C -o ControlMaster=auto -o ControlPersist=60s
-o StrictHostKeyChecking=no -o User=ansible -o ConnectTimeout=10 -o
ControlPath=/tmp/ansible_tower_JR8zTS/cp/ansible-ssh-%h-%p-%r -tt
10.192.73.218 'tmsh delete sys connection'
```

When set in an infinite loop would cause failures rather consistently.

Diagnosis? The act of SSH'ing to frequently to BIG-IP causes intermittent authentication failures. Solution? Don't use SSH because it's resource-intensive "enough" to be unreliable for things that need to be reliable.

Now, this creates an issue. By giving up SSH, you also implicitly give up on SSH certificate based access. Many customers made a stink about this. Well, on one hand their frustration is justified. On the other hand though, it would be a herculean amount of work to support SSH.

First, because of the above problem, but also because,

- You don't always have root shell access to your box. The cloud versions of BIG-IP come with "Appliance mode" turned on which puts you directly into *tmsh*
- *tmsh* has no structured output format like JSON. There's no consistent way to parse *tmsh* output. This means we would have to commit serious amounts of code to, ultimately, poorly parsing free-form text output. This would impact how long it took us to bring new modules to market and how reliably we could do that.
- The *tmsh* commands and output changes across nearly all versions of BIG-IP
- There is no SDK to consistently interact with *tmsh*

REST is, frankly, easy to code for

This constraint is relevant to the developers who actually code these modules. For the first iteration of the modules, what I refer to 1st-gen, the modules were written by primarily two people who were not F5 employees.

For the 2nd-gen modules, they were all written by one guy; me, an F5 employee. This was when F5 began put more skin in the Ansible game.

For the 3rd-gen modules, they were written primarily by myself and one other F5 employee. Then, they were tested and promoted by several other F5 employees.

The reason that I bring these points up is to emphasize just how __few__ people are actually working on the modules that you are using.

There is no "Ansible team" at F5. Due to my limited resources, I placed a priority on ease-of-development and testing. I needed to churn reliable product out at a rather fast pace compared to the pace that all other F5 products are released. My timeline was weeks, not bi-yearly like BIG-IP releases.

SOAP has different APIs for every configuration point. Literally. If you need to set the description of a Virtual Server, you need to use the set_description API, but if you need to set a *destination* of the same Virtual Server you need to use the set_destination_v2 API. It's *v2* because there is also a set_destination API that must be used for anything before version 11 of BIG-IP.

This is kinda frustrating from a developer point of view because you need to know all these different APIs.

It's frustrating from the admin's point of view because each API is another round-trip to the BIG-IP. Each time we need to talk to BIG-IP it means a slow-down and a chance that a failure could happen.

This can be worked around through the use of Transactions in SOAP, but that just _another_ thing that the users of the API need to be aware of when writing any sort of integrations with BIG-IP.

REST configures based on a "resource" so many APIs are implicitly transactional without needing to use transactions. Additionally, these resources mean that you only need to refer to __one__ API when changing most things about a particular object in BIG-IP.

For example, using our virtual server example above, instead of 2 or more APIs, there is only one; */mgmt/tm/ltm/virtual*. Sending a *GET* request to the resource returns a single JSON payload where you can change the *description* or *destination* as needed and then send a *PATCH* back to the BIG-IP with those changed values.

Also, it works like this across __all__ of the resources in BIG-IP. Which means once you have learned how to use one resource, you've essentially learned how to use all of them.

Note: It should be noted that due to bugs in the REST API this is not __always__ true, but it is true enough that you can consider it "the way things are" and handle the edge cases as you encounter them. Indeed, we handle just such edge cases for you in the f5-sdk so that you don't need to care. That is one of the many reasons to use the SDK; we iron out the inconsistencies in the API.

From a developer point-of-view, this requirement to learn a convention instead of learning a library of API calls means that new developers can be onboarded more quickly and existing developers can more easily add new functionality and support existing functionality.

From an admin point of view, this means that we need to make fewer round-trips to your BIG-IP and this should therefore speed up the operations that we do perform on the BIG-IP.

The F5 Python SDK is built on REST

The tool underlying all future Ansible F5 module development is the F5 Python SDK.

First, some history of this SDK.

F5 is notorious for writing half-baked "SDKs". These "SDKs" always have the following things in common.

- Written by one engineer
- The "SDK" covers Pools, Virtuals, and pool members
- The engineer has left the company
- No resources were ever dedicated to the "SDK"

I didnt want this to be the same story with the Python SDK that had been developed by the OpenStack team. Since the OpenStack integration was a project at F5 that had real resources dedicated to it, and the OpenStack integration relied implicitly on the Python SDK, it was safe-enough to consider the Python SDK "supported".

I wanted to further re-enforce the need to keep this SDK alive though, so I chose to build all the Ansible modules to use it. My hope was that if one project (OpenStack) had resources dedicated to it, then maybe I could get a second major project (Ansible) to also get resources dedicated to it to give the SDK a greater chance of surviving.

I also wanted to focus developer effort and expertise instead of fragmenting it unnecessarily. My goal was that more engineers contributing to this SDK would negate the need for fragmenting this development effort and that we would ultimately be building everything off of this one SDK and dog-fooding it appropriately.

REST was also chosen because native ability to speak "REST via HTTP" is built into all programming languages these days. We were using Python in this case, but it is not much of a leap to expand this same functionality to Ruby or Go or JavaScript or any language you may be interested in. All of them have native support for speaking HTTP.

Another reason to use this REST SDK is that it is easy to debug JSON payloads with common toolchains. For instance, working with Chrome developer tools, Postman, or other REST clients is simple. SOAP envelopes are more difficult to humanly consume as they are usually in an XML formatted payload and it's not readily obvious what tools one would use to send payloads like this back and forth to a BIG-IP.

Finally, the Python packages *suds* and *bigsuds* are not Python 3 compatible, and (at least in *bigsuds* case) supported or used by anyone at F5. There was no demand for building an SDK that supported an API that only a minority of colleagues was using at F5 or in the community.

Other F5 products made REST a first-class citizen

BIG-IP is not F5's only product. BIG-IQ and iWorkflow are two other products that we make. Both of these products natively use REST API communication for __all__ of their functionality.

Indeed, if you use a network inspector like those built-into Chrome or Firefox, you can see the actual APIs these F5 products communicate with and the payloads that they use.

Ok, fine, but "these products ship on something that has SSH access" you might say. That's true, but in the future they won't. Teams developing these products are rapidly turning them into standalone applications; what we refer to as "TMOS independence".

So in the future they will __not__ have CLI's other than whatever is provided by the operating system that hosts them.

Also, each of these products provides functionality that allows they to proxy requests directly to the BIG-IPs that they manage. We refer to this as "REST Proxy". That these tools provide such native support is testament to how REST is considered to be a first-class citizen for configuring our devices.

Other vendor APIs are always REST-like

If you look at the API landscape, nearly every vendor API is REST-like. It's becoming increasingly uncommon to see SOAP APIs because, compared to JSON-over-HTTP using HTTP verbs, SOAP is just a little too heavy-handed.

Most applications can represent their data structures just fine using JSON. Its largely unnecessary to provide anything bigger than just a JSON payload. Languages can natively transform scalars, lists, and dictionaries to the data structures native to the language.

Indeed, even in a complicated system like BIG-IP, all of our data structures can be represented by a JSON payload.

To make the adoption of our APIs easier for those admining our box and integrating with it, it was important to use technology that was already familiar to them.

Since customers are already largely exposed to REST-like APIs from their dealings with other vendors, it was natural to make use of the REST API instead of some other format, or, direct SSH communication.

The people working on this codebase work with REST and the SDK every day

The F5 OpenStack team began the trend of SDK development with their work on our Openstack integration. This progressed to include my adopting their work in Ansible. Today, the people who are working on Ansible modules are the same developers who were initially working on the F5 Python SDK.

Furthermore, we are introducing more teams at F5 to the Python SDK so that they too may integrate it into their testing procedures.

So as you can see, the majority of the new work being done at F5 is being done by people who are familiar with REST.

There is a sizable amount of pre-existing work in test harnesses and other stuff at F5 that is based on SSH, but the experts that were involved in writing that have since left the company and no expertise exists to further develop it; nor do those teams want to put further development into it.

With this increasing body of knowledge around our REST API, it makes less sense to attempt to support SSH.

The Ansible persistent network connection was not mature at the time

Persistent network device connections was released in Ansible 2.3. A __significant__ amount of work on the modules however, had already been done prior to this release.

To expect that one guy (Tim) to,

- change all those 30 modules
- support both modes (API and SSH) of configuring the remote device
- that had taken multiple years to write

was not something I wanted to undertake.

I honestly leave this open as an exercise for the end user. If you are deeply interested in making SSH happen, then by all means go after it. Modules that come out of F5 directly though will remain REST based for the foreseeable future.

CHAPTER 15

Dealing with “replace all”

A common ask from customers and colleagues is to mimic the behavior of the “replace-all-with” *tmsh* functionality via Ansible modules. The following document discusses challenges and proposes solutions for this feature when it is requested.

Challenges

By it’s nature, Ansible is not designed to support something such as *replace-all-with*. Ansible modules are normally intended to be run per-device and therefore should, in most cases, be accepting a single item of configuration and applying it per that device.

For example,

The above task, iterated per host, would create a number of SNAT pools. There is the desire, however, to remove all the existing and replace with a new list. Suppose that you did not know what the existing SNAT pools were. In that case, how would you remove the existing to add new ones?

This pattern of “unit of work per host” becomes an anti-pattern when applied to *replace-all-with*. This is because there is no way to reliably tell the module to

- Delete all the existing
- Add what I give you

Since the module only knows about what it receives at time of execution and not about what the Play is doing as a whole (or even that its in a loop) you cannot specify, for example a *replace_all_with* parameter.

It’s also unacceptable to have something like an *append* parameter because, again, the module is not aware that it is in a loop or what the greater Play is doing.

There may be some ability to make the module aware by specifying these list squashing modules in Ansible’s default *squash_actions* configuration variable, but this is a very untenable solution because we would be either

- Changing core code
- Asking users to create custom *ansible.cfg* files every time they use BIG-IPs

Proposals

What I propose is to provide the user with the ability to know what exists so that they can use the *absent* state of a module to remove all existing instances. This will be done using a **_facts* module for the manipulation module in question.

Using the above example of *bigip_snat_pool*, the facts module would be *bigip_snat_pool_facts*. The user could provide filtering params such as those they can supply to the *bigip_snat_pool* module to return only values that meet those criteria.

The user can then store those returned values using a *register* variable and then loop over the values to delete them all before adding new ones.

For example,

Future additions

Additionally, I would like to pursue the development of modules to support transactions such as

- bigip_transaction

This could be used to ensure that the above example is done in a way that would tolerate a failure between deleting and re-creating SNAT pools. Thus, the *replace-all-with* functionality would essentially be retained.

For example,

Note the addition of Transactions above. This new functionality would be put into the *f5_utils* module_utils code inside of Ansible to be supported across all modules.

For modules that do not support it, a *@property* would be defined to return only *None*.

For example,

This is similar in how the *bigip_partition* module always returns *None* for the *partition* parameter because you cannot create partitions inside of partitions.

All properties should read from self._values[key]

All property.setters should write to self._values[key]

The 'key' in the self._values dictionary should match the API resource attribute. This allows us to easily determine

CHAPTER 16

Deprecating functionality

In a module, it may be necessary to raise warnings to the user due to deprecated functionality.

Normally, deprecations are done in the ArgumentSpec, such as when using the *removed_in_version* key shown below.

but that is only relevant when the **parameter itself** is deprecated. There are other deprecation scenarios for instance where the parameter is a list of choices and the **choices themselves** are deprecated.

For example, consider the following parameter

One may need to deprecate the values themselves in favor of other values.

This may seem like a simple thing that one could add code to fix, but doing so would also lead to an increase in technical debt. Having a mapping of old values to new values should be considered an anti-pattern and something that is a candidate for deprecation.

To announce deprecations (for all things) you can use the *removed_in_version* field mentioned above, but you can also have more customized deprecations raised by your module.

To do this, begin by amending the *__init__* method of your *Parameters* class to define a *__warnings* variable.

Next, we need to add a new method to the *ModuleManager*, or, class specific manager (such as those used when forking logic; ie, *bigip_gtm_pool*)

The definition of this method is the following

The third and final step is to actually make use of the deprecation code that you set up previously. Do do that, you want to **append** to the aforementioned *__warnings* field.

An example is show below.

CHAPTER 17

pycodestyle

Your modules should be flake free flake8

Your modules should conform to ansible's validate-modules code

CHAPTER 18

Design Patterns

These patterns are intended to

- make your time spent developing new modules shorter
- allow you to not need to decide “what to do”
- allow for easier unit testing
- allow for customizing the modules to meet edge cases easier
- allow for customizing the modules to meet feature requests easier
- allow for customizing the modules to address bug reports easier

If these patterns conflict with the above goals, the patterns should be re-evaluated and all modules should be changed to support the new patterns.

CRUDable

- bigip_static_route

Only Updatable

- bigip_snmp

Executable

- bigip_command

CRUDable Reference

iworkflow_tenant_connector

List item as member

- bigip_remote_syslog

CHAPTER 19

Class variables

The following class variables are common attributes that each *Parameters* class needs to define.

updatables

Specifies a list of *Parameters* properties to that are considered updatable by the module. This is used when doing *should_update()*‘ comparisons and setting properties in *self.changes*.

api_attributes

Specifies a list *Parameters* properties to provide to the *api_params()*‘ method when generating valid sets of attributes for resources in the REST API.

You will likely need to write adapter methods that call the properties used internally by the module when writing these. For example

The reason that we use this method instead of the map method is because there may be cases where the value used in *api_params()* is not a single property but a set of properties that need to be combined.

This is used by the *api_params* method to generate a valid set of attributes to provide to the REST API. Typically this dictionary does NOT provide the *name* and *partition* parameters. These values should be specified specifically in the (create|update|delete)_on_device methods

returnables

Specifies a list of Parameters properties for the *to_return()* method to iterate over when supplying “changed” options back to the user

api_map

We need to have a dictionary or a list of some stuff because there are times when the API parameters can not be written as methods. For example, the *bigip_device_dns* APIs parameters include

This attribute is mapped to *forwarders* in the Ansible module.

The pattern that I had been developing is to use methods decorated as properties in python and then to call those methods when setting values and getting values.

For example, the "dns.proxy.__iter__" API attribute would be mapped to the *_values* key "forwarders". Normally I would set the set the API attributes directly in the dictionary. I would need to get those API specific keys however when I am returning the values to compare. this makes the getters for the Module options look messy though.

Next I thought about having the API attributes have their own @property decorators, but this won't work in the "dns" case mention above.

NEED a pattern for a single Ansible Option Parameter that returns 2 API attributes. For example in the bigip_virtual_server module there is an option called enabled vlans. This, however, actually sets two (possibly 3) values in the API

- vlans (list
- vlansDisabled (boolean True)
- vlansEnabled (boolean True)

what is a pattern that, that supports that?

The pattern is that the api_attributes is an arbitrary list of attributes that you want to send to the API.

The api_params() method uses this list to iterate over the

param_api_map does not work for situations where the Ansible->API relationship is 1->n (bigip_virtual_server with enabled_vlans) param_api_map only works for 1->1

Requirements

- easy attribute comparison in Ansible parameters format with BIG-IP API values
- ability to consume API attributes that cannot be written as python functions (dns.proxy.__iter__ for example)

params_spec=dict(

cache='dns.cache', forwarders='dns.proxy.__iter__', name_servers='nameServers', search='search', ip_version='include'

)

updatables = ['cache', 'forwarders', 'name_servers', 'search', 'ip_version'

]

)

CHAPTER 20

Common classes

Nearly every module (see exceptions) should have the following classes. These classes are used to support the stated design patterns.

- Parameters
- ModuleManager
- ArgumentSpec

Exceptions to common classes

Exceptions to the above rules will happen when,

- the API that a particular module addresses, changes underneath it between versions of the software.
- the resources or collections that the module is manipulating become too numerous

Good examples of this include

- bigip_ssl_certificate
- bigip_gtm_wide_ip

CHAPTER 21

Defaulting to None

It should be noted that you should never specify default values in your *ArgumentSpec*. For example, the following is incorrect

But, shouldn't you be using the actual defaults?

> Answer: No

The reason that you provide no defaults is to support cases where the user does not specify a value for a particular option. If that happens, then you should not step on that parameter if it is preconfigured.

If a user had a setting that they want to keep and you specified a default value, then in the first opportunity that they forgot to specify that value, you would end up replacing that value with your default.

This is a bad idea.

Ansible defaults *required* to *False* and *default* to *None*. Therefore, there is no need to specify these default values.

CHAPTER 22

What is the layer of @property decorators all about?

The ‘‘@property‘ decorators you see represent an adapter pattern. Inside of the *ModuleManager*, when data needs to be compared (what you have vs what you want), that data is returned by these properties in a known format.

The API's resource attributes differ in structure and name from the options that a user can provide to a module.

For example, an API resource may have an attribute called *minSupportedBIGIPVersion*. The user facing portion of the module though, may refer to this attribute as *min_bigip_version*. There are a number of reasons to do this.

- it provides an abstraction of the API so the name of the thing being modified is not closely tied to the implementation of the API.
- many times the API attribute names are vague, this abstraction makes them more clear
- the Resource Attributes use camelCase variable naming, while some of python and nearly all of Ansible use snake_case variable naming.

For future developer clarity's sake, all of the attributes that we are interested in are typically compared by the option name that they would have in Ansible and not the Resource attribute name.

This allows a developer to look at the names of variables and match them to the names of the options in the Ansible module.

While the names of properties usually mirror the names of the module options available to the user, the values of those properties do not.

Values of the properties reflect the values that are accepted by the API resource. This is done because, ultimately, the values that we need to deal with at the values that are going to be used to update the API.

Therefore, when we receive options from the module, we transform them into the values that would appropriate for the API. When we receive values from the API, we might order them or cast some of their values to specific types so that comparisons can occur, but otherwise we dont really touch them.

So,

1. property name reflects module option
2. property getter reflects the appropriate Resource attribute value

CHAPTER 23

Why are they not all setters?

This is because there are some cases where you do not know ahead of time what the value of that property should be. Often it takes two or more options be set before another option can be known.

Consider a module that accepts an IP address option and a gateway mask option, but needs to return a CIDR representation of those two values. Without getting both values, we cannot produce the one value. That is who we calculate the necessary value at time of getattr, and not at the time of setattr.

CHAPTER 24

Use the module_utils test suite to verify AnsibleF5Parameters classes

This is important in case there is a pattern we miss for adapting API attributes and module params.

This test suite is found at the following location

test/misc/test_module_utils.py

CHAPTER 25

Never import *

9 times out of 10 you are doing this because you are using one of the following variables

- *BOOLEANS*
- *BOOLEANS_TRUE*
- *BOOLEANS_FALSE*

It is, however, an anti-pattern to import from * and it will be caught by the Ansible unit tests. Instead, specifically include each thing that you want to use.

CHAPTER 26

The Changes class

In many cases, the values that you process from the user will match the values that you send to BIG-IP.

For example, consider the following parameters to a module

Above, the module code that implements this is a collection of different adapters that collectively allow the module to convert the information the user provides to it into a format that it is able to send to the BIG-IP and vice-versa.

This class is a way for the module developer to complete the cycle of

> User (params) -> Module -> REST -> Module -> User (changed params)

Due to most of the adapters being concerned with how they should be adapting data to meet the format expect by the REST API, the *Changes* class is concerned with how to adapt the data to meet the format expected by the end user.

If there is a need to change the value to something that is more "human" so that the user can understand it, that job is undertaken by the *Changes* module.

An example of it in use is the *bigip_device_connectivity* module where it acts as a way to translate BIG-IP's representation of "none" (*any6*) to the human word "none"

Examples of modules that use the *Changes* class are,

- *bigip_gtm_datacenter*
- *bigip_device_connectivity*
- *bigip_device_group*

CHAPTER 27

The Difference class

When comparing values to detect changed-ness, there are situations where the default comparison method will not be appropriate for you use. The default comparison method essentially just does a simple comparison.

The source of this method illustrates its simplicity

As you can see, it is quite simple and does not take into consideration anything more complicated than simply the values being compared.

This differencing is not conducive to more complicated data structures or types of data.

The above fails to satisfy this simple (albeit erroneous due to established patterns) difference.

Note: This is logically incorrect because the Adapter pattern that you should be using for the *Parameters* class mandates that *@property* values should return a specific data type (in the above case *int*) and should never be non-deterministic.

To check for differences in more complicated data structures, the module author should make use of the *Difference* class.

The definition of the *Difference* class is the following

By default, it does nothing more than uses the simple comparison to diff the parameters provided, and discovered, by the module.

To make use of it, you must do the following.

First, define this class in your module.

Second, add *@property* methods for each of the values that you want to compare. Remember, the properties of the *Parameter* classes are the names that are exposed to the module user and not the names of REST API parameters themselves (unless it perfectly matches) because the REST API camel-cases all parameter names.

So, if we were interested in providing custom diffing for the *members* module parameter, we may add this as a *@property* to the *Difference* class like so.

These *@property* methods **must** be named after the Parameter you want to compare. Additionally, the return value of these *@property* definitions is one of two values.

- Python *None* if there is no difference
- The value of the difference if there is one. This value will later be reported as what was changed in the module invocation.

Finally, to make use of this new difference class, you must change the following method in the *ModuleManager* code,

- *_update_changed_options*

The new value of this method must include the usage of the *Difference* class as a new object. Example usage is provided below.

CHAPTER 28

API Map Adapter

This adapter pattern is useful for converting data values from user inputs to REST outputs. It's definition is,

> The API Map Adapter pattern adapts a known REST attribute to a predefined *Parameters* method. The return value of this method is a correct payload for the REST attribute.

This pattern is frequently used as a way for the module developer to translate the input provided to them by the user into a format that is consumable by the REST API.

The following is an example of this kind of adapter.

CHAPTER 29

1-to-1 Adapter

YAML represents the *banner* parameter as a simple key with a simple value. The actual REST payload contains an attribute called *banner* and it takes an actual value called *enabled*. This is represented in code by the *ArgumentSpec* class.

This is considered to be the most simple form of a parameter definition by the F5 Ansible modules because it is nearly a 1 to 1 translation of Ansible to F5.

The following is an example of this kind of adapter.