

Managing networks hasn't changed much in 30 years

- Networks are mission critical
- Every network is a unique snowflake
- Tactical, opportunistic changes proliferate
- Vendor specific implementations
- Testing is expensive/impossible



F5 and Ansible: Top customer outcomes



AUTOMATE F5 CONFIGURATIONS



CONTINUOUS DELIVERY



OPERATIONAL AGILITY



CONSISTENT RELIABILITY AND
SECURITY ACROSS CLOUDS

AUTOMATE F5 CONFIGURATIONS

CHALLENGE

Modern ephemeral environments require easy, consistent ways to manage configurations.

SOLUTION

F5 modules on Ansible and F5 roles on Ansible Galaxy.

OUTCOMES

- Manage F5 application services using Ansible Playbooks and roles.
- Write Ansible tasks for F5 once and run them over and over.
- Retrieve and manage F5 inventory ad hoc or in bulk.
- Version Ansible Playbooks to state check and validate by comparing running configurations to desired configurations.

RELATED ARTICLE

[Getting started with Ansible](#) ›



F5 and Ansible: Top customer outcomes



AUTOMATE F5 CONFIGURATIONS



CONTINUOUS DELIVERY



OPERATIONAL AGILITY



CONSISTENT RELIABILITY AND
SECURITY ACROSS CLOUDS

CONTINUOUS DELIVERY

CHALLENGE

Release application code quicker with improved efficiency, control, and zero down time.

SOLUTION

F5 BIG-IP VE, F5 DNS, F5 LTM and Ansible integration.

OUTCOMES

- Supports blue-green deployments.
- Programmatic validation before making changes makes the process safer.
- Low-risk deployments and upgrades without scheduling and announcing maintenance windows.
- Faster time to resolution because NetOps can move traffic out of a data center that's experiencing issues.

RELATED ARTICLES

[Customer story: Surescripts Speeds DevOps Work with Red Hat Ansible Automation >](#)

[Tackling Blue-Green Deployments in the Private Cloud >](#)

F5 and Ansible: Top customer outcomes



AUTOMATE F5 CONFIGURATIONS



CONTINUOUS DELIVERY



OPERATIONAL AGILITY



CONSISTENT RELIABILITY AND
SECURITY ACROSS CLOUDS

OPERATIONAL AGILITY

CHALLENGE

Need to reduce time to production for all necessary application services, including network services.

SOLUTION

[F5 Automation Tool Chain, F5 modules for Ansible](#)

OUTCOMES

- Develop, test, and release with enterprise-grade F5 application services by including them as part of CI/CD workflow.
- Admin-defined F5 deployment and configuration templates for the rest of your organization to choose from.
- Scheduled F5 deployment and configuration using [Red Hat Ansible Tower](#).

RELATED ARTICLE

[F5 Automation Toolchain and CI/CD with Ansible Tower](#) >

F5 and Ansible: Top customer outcomes



AUTOMATE F5 CONFIGURATIONS



CONTINUOUS DELIVERY



OPERATIONAL AGILITY



CONSISTENT RELIABILITY AND
SECURITY ACROSS CLOUDS

CONSISTENT RELIABILITY AND SECURITY ACROSS CLOUDS

CHALLENGE

As your app infrastructure expands across multiple clouds, forgetting, or missing a device when patching or deploying secure configurations can cause outages and even expose your apps to attackers.

SOLUTION

F5 Automation Tool Chain, Ansible F5 modules, F5 roles on Ansible Galaxy, F5 Cloud Formation Templates (CFT), and F5 Azure Resource Manager (ARM) templates

OUTCOMES

- Manage hybrid F5 environments with ease and consistency.
- Red Hat Ansible Tower provides single pane-of-glass management, helping to reduce the time and resources required to manage a distributed infrastructure.
- Create Ansible tasks and workflows to configure F5 devices once, and then apply the same workflows across multiple F5 infrastructures (cloud, virtual, and physical).
- Deploy pre-configured traffic and security policies consistently across multiple cloud environments.

RELATED ARTICLE

[Example: Deploying F5 Application Security Manager with Ansible >](#)

What You Will Learn

- What is Ansible, its common use cases
- How Ansible works and terminology
- Running Ansible playbooks
- Network modules
- An introduction to roles
- An introduction to Ansible Galaxy



**MANAGING NETWORKS
HASN'T CHANGED
IN 30 YEARS.**

Automation considerations

- Compute is no longer the slowest link in the chain
- Businesses demand that networks deliver at the speed of cloud
- Automation of repeatable tasks
- Bridge silos

What is Ansible?

Red Hat Ansible network automation is enterprise software for automating and managing IT infrastructure.

As a vendor agnostic framework Ansible can automate F5 (BIG-IP, BIG-IQ), Arista (EOS), Cisco (IOS, IOS XR, NX-OS), Juniper (JunOS), Open vSwitch and VyOS.

Ansible Tower is an enterprise framework for controlling, securing and managing your Ansible automation with a UI and RESTful API.

According to Gartner



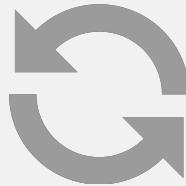
Figure 1

Primary Method for Making Network Changes

Source: Gartner, *Look Beyond Network Vendors for Network Innovation*. January 2018. Gartner ID: G00349636. (n=64)



SIMPLE



POWERFUL



AGENTLESS

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

Gather information and audit

Configuration management

Workflow orchestration

Manage ALL IT infrastructure

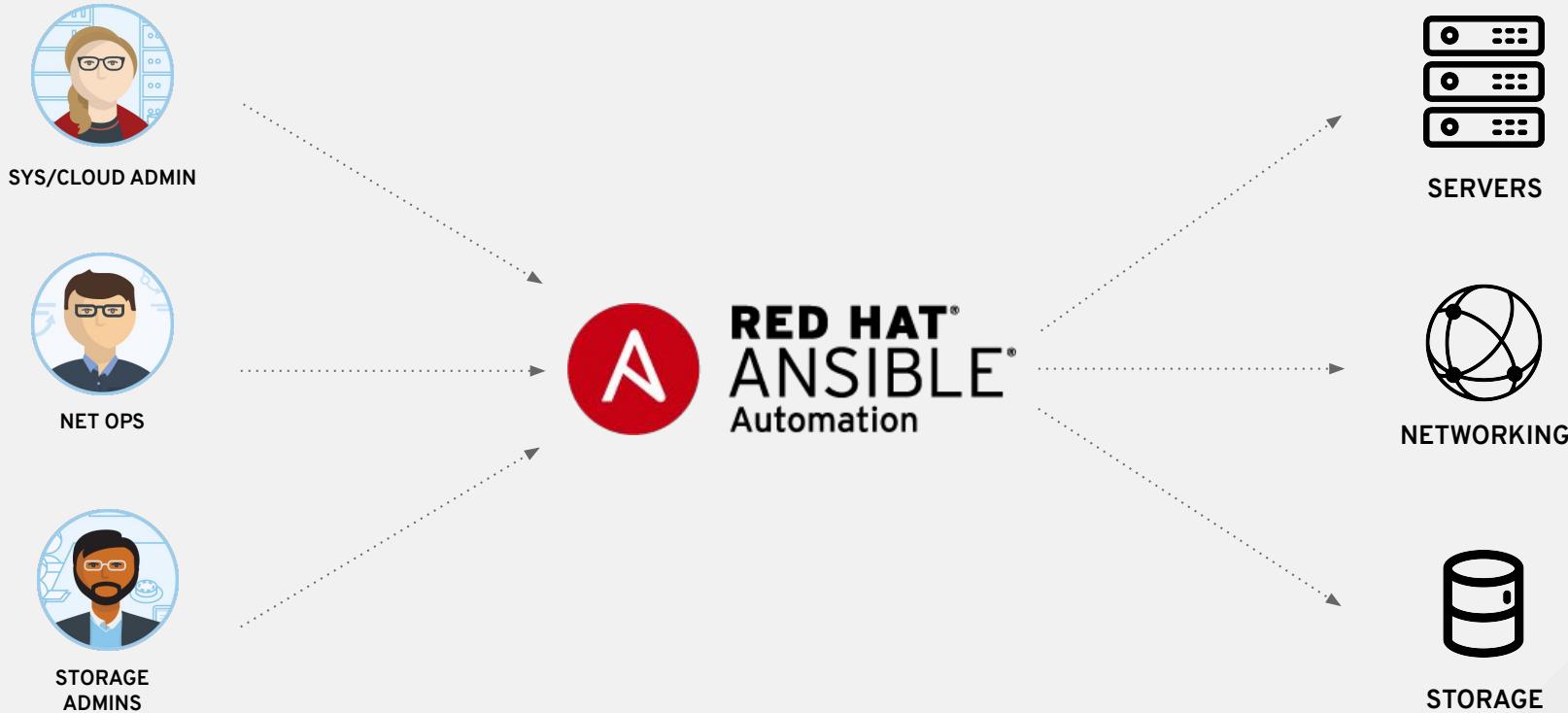
Agentless architecture

Uses OpenSSH and paramiko

No agents to exploit or update

More efficient & more secure

Ansible: The Universal Automation Framework



ANSIBLE NETWORK AUTOMATION

50

Networking
platforms

700+

Networking
modules

ansible.com/networking

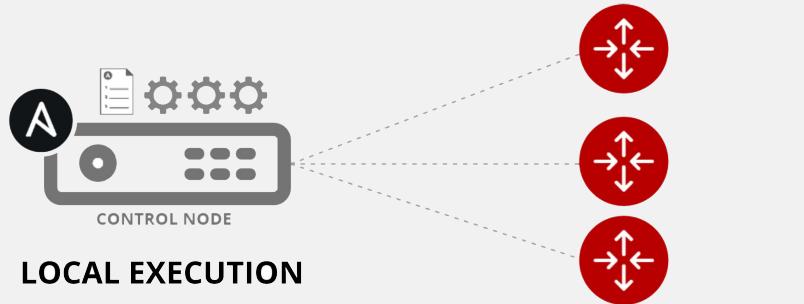
Common use cases

- Backup and restore device configurations
- Upgrade network device OS
- Ensure configuration compliance
- Apply patches to address CVE
- Generate dynamic documentation

Basically anything an operator can do manually, Ansible can automate.

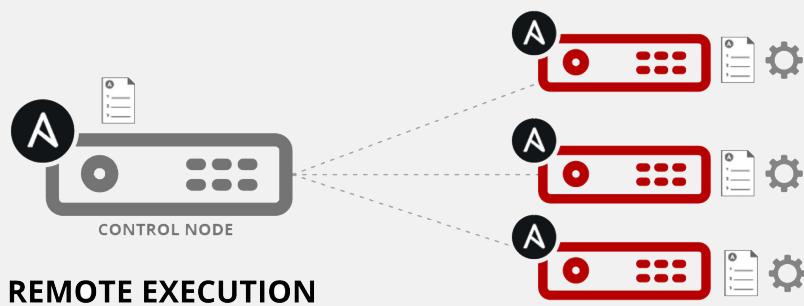
How Ansible Works

Module code is executed locally on the control node

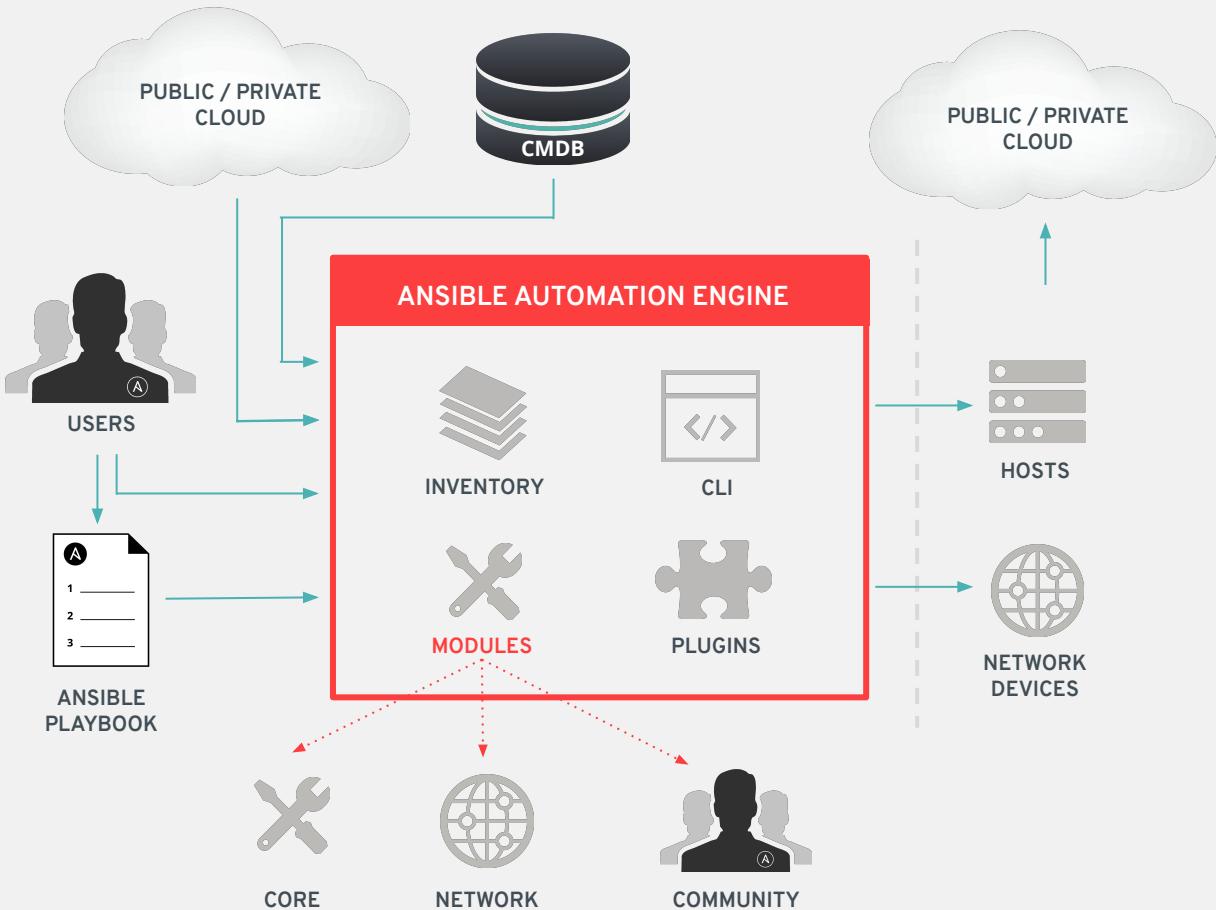


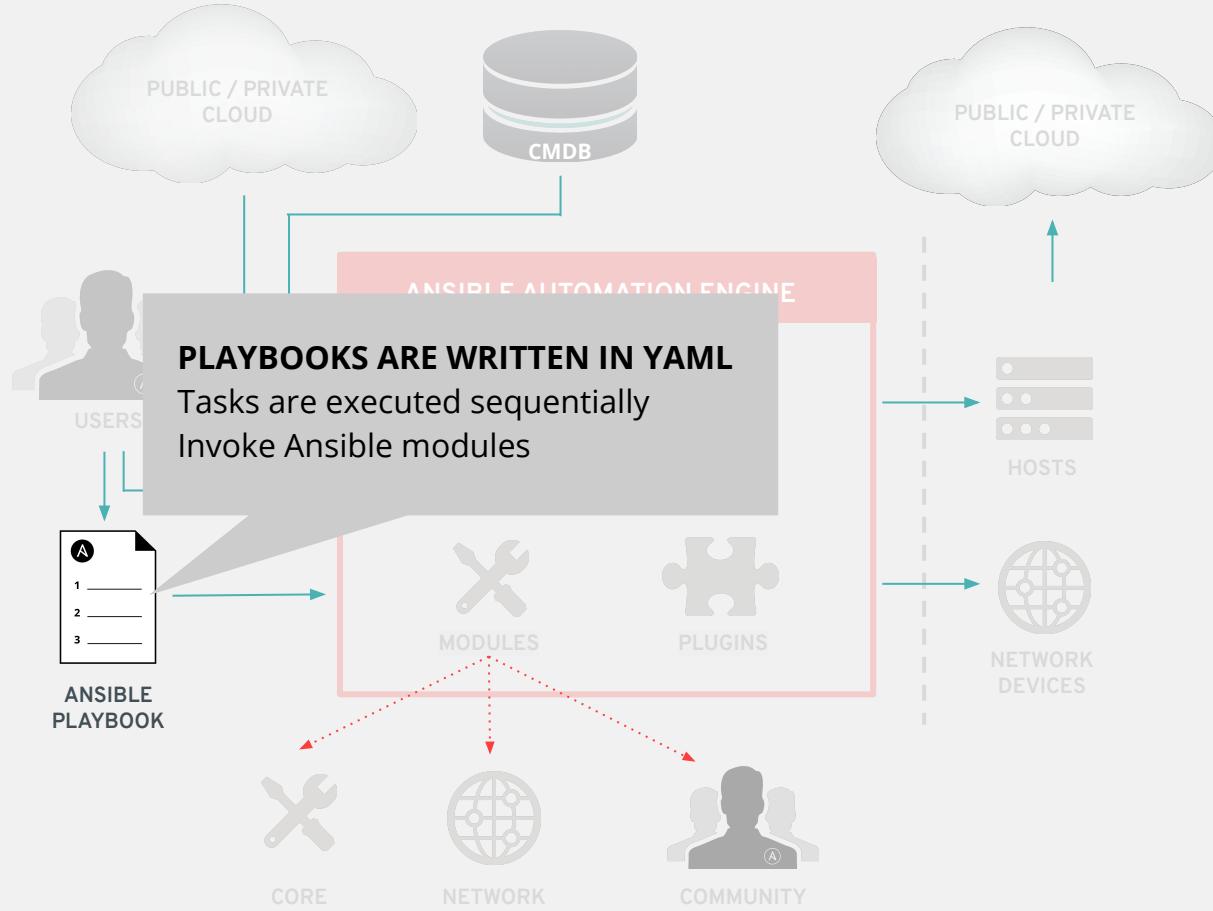
NETWORKING DEVICES

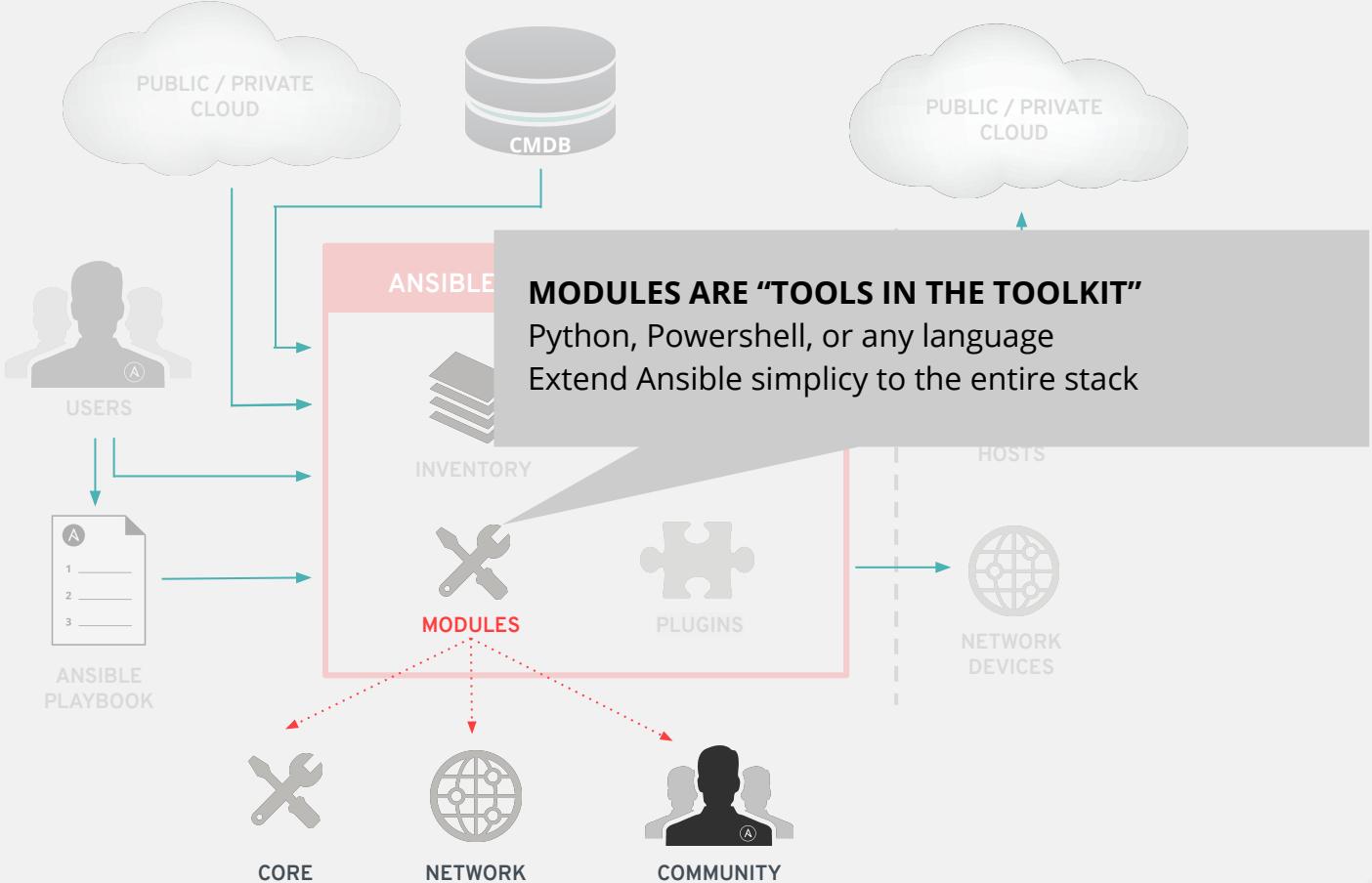
Module code is copied to the managed node, executed, then removed

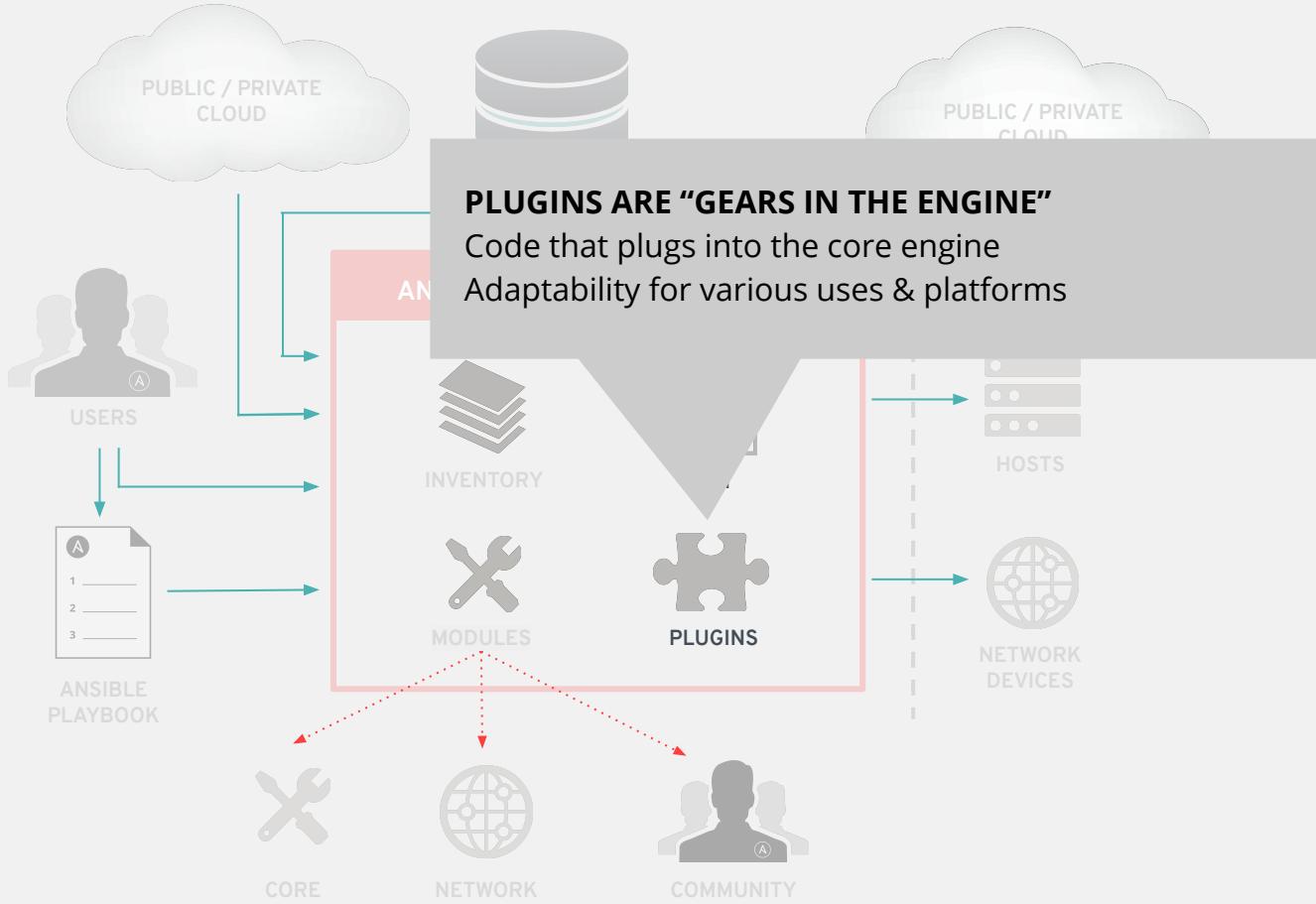


LINUX/WINDOWS HOSTS









Understanding Inventory

10.1.1.2

10.1.1.3

172.16.1.1

172.16.1.2

192.168.1.2

192.168.1.3

Understanding Inventory

There is always a group called "**all**" by default

```
[lb]

f5 ansible_host=34.199.128.69


[control]

ansible ansible_host=107.23.192.217


[webservers]

host1 ansible_host=107.22.141.4

host2 ansible_host=54.146.162.192
```

Groups can be nested

```
[DC:children]

lb

webservers


[rhel:children]

control

webservers
```

Inventory - variables

```
[all:vars]
ansible_user=student2
ansible_ssh_pass=ansible
ansible_port=22

[lb]
f5 ansible_host=34.199.128.69 ansible_user=admin private_ip=172.16.26.136 ansible_ssh_pass=admin

[webservers]
host1 ansible_host=107.22.141.4 ansible_user=ec2-user private_ip=172.16.170.190
host2 ansible_host=54.146.162.192 ansible_user=ec2-user private_ip=172.16.160.13
```

Group variables apply for all devices
in that group

Host variables apply to the host and
override group vars

A Sample Playbook

```
---  
- name: BIG-IP SETUP  
  hosts: lb  
  connection: local  
  gather_facts: false  
  
  tasks:  
  
    - name: CREATE NODES  
      bigip_node:  
        server: "f5.ansible.com"  
        user: "admin"  
        password: "admin"  
        server_port: "8443"  
        host: 192.168.0.1  
        name: "webserver01"
```

- Playbook is a list of plays.
- Each play is a list of tasks.
- Tasks invoke modules.
- A playbook can contain more than one play.

Lab Time

Exploring the Lab Environment

In this lab you will explore the lab environment and build familiarity with the lab inventory.

Approximate time: 10 mins

Playbook definition for network automation

- Target play execution using hosts
- Define the connection : local
- About `gather_facts`

Running a playbook

```
[student1@ansible ~]$ ansible-navigator run bigip-facts.yml -m stdout

PLAY [GRAB F5 FACTS] ****
TASK [COLLECT BIG-IP FACTS] ****
ok: [f5]

PLAY RECAP ****
f5 : ok=1     changed=0     unreachable=0    failed=0
```

Displaying output

Use the optional **verbose** flag during playbook execution

Limiting Playbook execution

Playbook execution can be limited to a subset of devices using the --limit flag.

```
$ ansible-navigator run bigip-facts.yml -m stdout --limit f5node1
```

Forget a flag / option ?
Just type ansible-playbook then press enter

Use the **--help** flag

Quick Refresher on JSON

Structured Data is easy to work with

```
"system_info": {  
    "base_mac_address": "0a:54:53:51:86:fc",  
    "chassis_serial":  
"685023ec-071e-3fa0-3849dcf70dff",  
    "product_version": "13.1.0.7",  
}
```

```
bigip_facts['system_info']['base_mac_address']
```

00a:54:53:51:86:fc



Registering the output

The register parameter is used to collect the output of a task execution. The output of the task is 'registered' in a variable which can then be used for subsequent tasks.

```
- name: COLLECT BIG-IP FACTS
  bigip_device_facts:
    gather_subset:
      - system_info
    server: "{{private_ip}}"
    user: "{{ansible_user}}"
    password: "{{ansible_ssh_pass}}"
    server_port: 8443
  register: bigip_device_facts
```

Displaying output - The "debug" module

The debug module is used like a "print" statement in most programming languages.

```
- name: DISPLAY ONLY THE MAC ADDRESS
  debug:
    var: bigip_device_facts['system_info']['base_mac_address']
```

```
TASK [DISPLAY ONLY THE MAC ADDRESS] ****
ok: [f5] => {
    "bigip_device_facts['system_info']['base_mac_address']": "0a:54:53:51:86:fc"
}
```

Limiting tasks within a play

- Tags allow the user to selectively execute tasks within a play.
- Multiple tags can be associated with a given task.
- Tags can also be applied to entire plays or roles.

```
- name: DISPLAY THE VARIABLE OUTPUT
  debug:
    var: output_variable
  tags: debug
```

Tags are invoked using the --tags flag while running the playbook

```
[user@ansible]$ ansible-navigator run bigip-facts.yml -m stdout --tags=debug
```

Limiting tasks within a play - or skip them!

- --skip-tags allows you to skip everything

```
- name: DISPLAY THE VARIABLE OUTPUT
  debug:
    var: output_variable
  tags: debugtask
```

Tags are invoked using the --tags flag while running the playbook

```
[user@ansible] $ ansible-navigator run bigip-facts.yml -m stdout
--skip-tags=debugtask
```

A note about variables

Other than the user defined variables, Ansible supports many inbuilt variables. For example:

Variable	Explanation
ansible_*	Output of fact gathering
inventory_hostname	magic inbuilt variable that is the name of the host as defined in inventory
hostvars	magic inbuilt variable dictionary variable whose key is <code>inventory_hostname</code> e.g. <code>hostvars[webserver1].my_variable</code>

Lab Time

Exercise 1.1 -Using Ansible to gather data from F5 BIG-IP

In this lab you will write your first playbook and run it to gather facts from a F5 BIG-IP load balancer.

Approximate time: 15 mins

Modules

Modules do the actual work in Ansible, they are what gets executed in each playbook task.

- Typically written in Python (but not limited to it)
- Modules are idempotent
- Modules take user input in the form of parameters

Network modules

Ansible modules for network automation typically references the vendor OS followed by the module name.

- `*_facts`
- `*_command`
- `*_config`

More modules depending on platform

Arista EOS = `eos_*`

Cisco IOS/IOS-XE = `ios_*`

Cisco NX-OS = `nxos_*`

Cisco IOS-XR = `iosxr_*`

F5 BIG-IP = `bigip_*`

F5 BIG-IQ = `bigiq_*`

Juniper Junos = `junos_*`

VyOS = `vyos_*`

Modules Documentation

<https://console.redhat.com/ansible/automation-hub>

The screenshot shows the Red Hat Hybrid Cloud Console interface. The left sidebar is titled "Ansible Automation Platform" and includes sections for Overview, Automation Hub (with Collections, Partners, Repo Management, Connect to Hub), Automation Services Catalog, Insights, Reports, Savings Planner, Automation Calculator, Organization Statistics, Job Explorer, Clusters, Notifications, and Documentation. The Documentation section is currently selected. The main content area is titled "f5_modules" and shows the "Documentation" tab selected. It displays the "f5networks.f5_modules" collection, which includes a "Readme" file and 177 modules. The modules listed include bigip_aspm_ad, bigip_aspm_network_access, bigip_aspm_policy_fetch, bigip_aspm_policy_import, bigip_asm_advanced_settings, bigip_asm_dos_application, bigip_asm_policy_fetch, bigip_asm_policy_import, bigip_asm_policy_manage, bigip_asm_policy_server_technol..., bigip_cgnat_lsn_pool, bigip_cli_alias, bigip_cli_script, bigip_command, and bigip_config. The page also contains sections for Requirements (listing ansible >= 2.9 and ipaddress (python < 3.5)), Installation (instructions for ansible-galaxy collection install), and Semantic Versioning examples. It also includes a Deprecation Notice about Python 3.5 and an Example Usage section.

Red Hat Hybrid Cloud Console | All apps and services

Ansible Automation Platform

Partners > f5networks > f5_modules > Documentation

f5_modules

Documentation (1) Readme

Modules (177)

- bigip_aspm_ad
- bigip_aspm_network_access
- bigip_aspm_policy_fetch
- bigip_aspm_policy_import
- bigip_asm_advanced_settings
- bigip_asm_dos_application
- bigip_asm_policy_fetch
- bigip_asm_policy_import
- bigip_asm_policy_manage
- bigip_asm_policy_server_technol...
- bigip_cgnat_lsn_pool
- bigip_cli_alias
- bigip_cli_script
- bigip_command
- bigip_config

Requirements

- ansible >= 2.9
- ipaddress (python < 3.5)

Installation

To install in ansible default or defined paths use:

```
ansible-galaxy collection install f5networks.f5_modules
```

To specify the installation location use -p. If specifying a folder, make sure to update the ansible.cfg so ansible will check this folder as well.

```
ansible-galaxy collection install f5networks.f5_modules -p collections/
```

To specify the version of the collection to install, include it at the end of the collection with ::=1.0.0:

```
ansible-galaxy collection install f5networks.f5_modules::=1.0.0
```

Semantic Versioning examples below: - Increment major (for example: x in x.y.z) version number for an incompatible API change. - Increment minor (for example: y in x.y.z) version number for new functionality in a backwards compatible manner. - Increment patch (for example: z in x.y.z) version number for backwards compatible bug fixes.

Deprecation Notice

Support for versions of Python earlier than 3.5 is being deprecated and will be removed at a future date.

Example Usage

To use a module from a collection, reference the full namespace, collection, and module name that you want to use:

Modules Documentation

Documentation right on the command line

```
[user@ansible]$ ansible-navigator doc bigip_device_facts -m stdout
```

```
> BIGIP_DEVICE_FACTS  (/usr/lib/python2.7/site-packages/ansible/modules/network/f5/bigip_device_facts.py)
```

```
    Collect facts from F5 BIG-IP devices.
```

```
OPTIONS (= is mandatory):
```

```
= gather_subset
```

```
    When supplied, this argument will restrict the facts returned to a given subset.
```

```
    Can specify a list of values to include a larger subset.
```

```
.
```

Inventory - Revisiting Variables

```
[lb]
```

```
f5 ansible_host=34.199.128.69 ansible_user=admin private_ip=172.16.26.136  
ansible_ssh_pass=admin
```

ansible_host	34.199.128.69
ansible_user	admin
private_ip	172.16.26.136
ansible_ssh_pass	admin

Using the F5 bigip_node module

```
- name: CREATE NODES

  bigip_node:

    server: "{{private_ip}}"
    user: "{{ansible_user}}"
    password: "{{ansible_ssh_pass}}"
    server_port: "8443"
    validate_certs: "no"
    host: "{{hostvars[item].ansible_host}}"
    name: "{{hostvars[item].inventory_hostname}}"

  loop: "{{ groups['webservers'] }}"
```

Using the F5 bigip_node module

- **name**: CREATE NODES

```
bigip_node:
```

```
  server: "{{private_ip}}"
  user: "{{ansible_user}}"
  password: "{{ansible_ssh_pass}}"
  server_port: "8443"
  validate_certs: "no"
```

Information for connecting
to F5 BIG-IP load balancer

```
  host: "{{hostvars[item].ansible_host}}"
  name: "{{hostvars[item].inventory_hostname}}"
  loop: "{{ groups['webservers'] }}"
```

Using the F5 bigip_node module

```
- name: CREATE NODES  
  
  bigip_node:  
  
    server: "{{private_ip}}"  
    user: "{{ansible_user}}"  
    password: "{{ansible_ssh_pass}}"  
    server_port: "8443"  
  
    validate_certs: "no"  
  
    host: "{{hostvars[item].ansible_host}}"  
    name: "{{hostvars[item].inventory_hostname}}"  
  
loop: "{{ groups['webservers'] }}"
```

nodes being added

- host refers to the web server IP address
- name is a human identifiable trait can be the DNS name but does not depend on it

Using the F5 bigip_node module

```
- name: CREATE NODES  
  
  bigip_node:  
  
    server: "{{private_ip}}"  
    user: "{{ansible_user}}"  
    password: "{{ansible_ssh_pass}}"  
    server_port: "8443"  
    validate_certs: "no"  
  
    host: "{{hostvars[item].ansible_host}}"  
    name: "{{hostvars[item].inventor_hostname}}"  
  
  loop: "{{ groups['webservers'] }}"
```

Loops over all the web servers in the **group** webservers

Lab Time

Exercise 1.2 -Adding nodes to F5 BIG-IP

In this lab you will creating a playbook that makes use of the BIG-IP node module to add two RHEL (Red Hat Enterprise Linux) web servers as nodes for the BIG-IP load balancer.

Approximate time: 15 mins

Using the F5 bigip_pool module

```
- name: CREATE POOL  
  
  bigip_pool:  
<<login info removed for brevity>>  
    name: "http_pool"  
    lb_method: "round-robin"  
    monitors: "/Common/http"  
    monitor_type: "and_list"
```

Using the F5 bigip_pool module

```
- name: CREATE POOL  
  
  bigip_pool:  
<<login info removed for brevity>>  
    name: "http_pool"  
  
    lb_method: "round-robin"  
  
    monitors: "/Common/http"  
  
    monitor_type: "and_list"
```

The **name** is a user defined name that we will add nodes to in a later exercise

Using the F5 bigip_pool module

```
- name: CREATE POOL  
  
  bigip_pool:  
<<login info removed for brevity>>  
  
    name: "http_pool"  
    lb_method: "round-robin"  
  
    monitors: "/Common/http"  
  
    monitor_type: "and_list"
```

The **lb_method** refers to the load balancing method, a full list is provided on the module documentation

Using the F5 bigip_pool module

```
- name: CREATE POOL  
  bigip_pool:  
<<login info removed for brevity>>  
    name: "http_pool"  
    lb_method: "round-robin"  
    monitors: "/Common/http"  
    monitor_type: "and_list"
```

The **monitors** parameter refers to the protocol that the F5 BIG-IP load balancer will be listening on

Using the F5 bigip_pool module

```
- name: CREATE POOL  
  
  bigip_pool:  
<<login info removed for brevity>>  
    name: "http_pool"  
    lb_method: "round-robin"  
    monitors: "/Common/http"  
    monitor_type: "and_list"
```

This **monitor_type** parameter is technically the default. We can actually configure multiple monitors (protocols) simultaneously

F5 Web GUI

The screenshot shows the F5 BIG-IP Web GUI interface. The top navigation bar displays the URL <https://34.199.128.69:8443/xui/>, a warning about the connection being not secure, and the user Sean Cavanaugh. The top status bar shows the hostname ip-172-16-26-136.ec2.internal, IP address 172.16.26.136, date Aug 3, 2018, time 7:29 PM (UTC), user admin, role Administrator, and partition Common. A green "Log out" button is also present.

The main menu on the left includes links for Apps, Cisco Systems IOS, community/network..., What networking pl..., NETCONF and YAN..., OpenConfig, Data..., OpenConfig - Home, Log out, Main, Help, and About. The Local Traffic section is expanded, showing Network Map, Virtual Servers, Policies, Profiles, Ciphers, iRules, Pools (which is selected and highlighted in yellow), Nodes, Monitors, Traffic Class, and Address Translation.

The central content area is titled "Local Traffic > Pools : Pool List". It features a "Pool List" tab and a "Statistics" tab. A search bar and a "Create..." button are at the top of the pool list table. The table has columns for Status, Name, Description, Application, Members, and Partition / Path. One row is visible: http_pool, which is healthy (green diamond icon), has 0 members, and is associated with the Common partition.

F5 Web GUI - Configuration

Click on the pool to get more information.
Monitor ‘http’ assigned to the pool.

Configuration: Basic ▾

	Active	Available
Health Monitors	/Common http	/Common gateway_icmp http_head_f5 https https_443

<< >>

Lab Time

Exercise 1.3 -Adding a load balancing pool

Demonstrate use of the BIG-IP pool module to configure a load balancing pool in BIG-IP device. A load balancing pool is a logical set of devices, such as web servers, that you group together to receive and process traffic.

Approximate time: 15 mins

Using the F5 bigip_pool_member module

```
- name: ADD POOL MEMBERS

  bigip_pool_member:
<<login info removed for brevity>>

    state: "present"

    name: "{{ hostvars[item].inventory_hostname }}"
    host: "{{ hostvars[item].ansible_host }}"
    port: "80"
    pool: "http_pool"

  loop: "{{ groups['webservers'] }}"
```

F5 BIG-IP Web GUI

The web servers are now configured and can be found under the Members tab of **http_pool**

The screenshot shows the F5 BIG-IP Web GUI interface. The top navigation bar displays the host information: Hostname: ip-172-16-26-136.ec2.internal, IP Address: 172.16.26.136, Date: Aug 3, 2018, Time: 7:33 PM (UTC), User: admin, Role: Administrator, and Partition: Common. The title of the main window is "Local Traffic > Pools : Pool List > http_pool". The left sidebar menu includes options like Network Map, Virtual Servers, Policies, Profiles, Ciphers, iRules, Pools (which is selected and highlighted in yellow), Nodes, Monitors, Traffic Class, and Address Translation. Below the sidebar is an "Acceleration" icon. The main content area shows the "Members" tab for the "http_pool". Under "Load Balancing", the "Load Balancing Method" is set to "Round Robin" and "Priority Group Activation" is set to "Disabled". The "Current Members" table lists two hosts:

Status	Member	Address	Service Port	FQDN	Ephemeral	Ratio	Priority Group	Connection Limit	Partition / Path
<input checked="" type="checkbox"/>	host2:80	54.146.162.192	80		No	1	0 (Active)	0	Common
<input checked="" type="checkbox"/>	host1:80	107.22.141.4	80		No	1	0 (Active)	0	Common

Buttons at the bottom of the table include "Enable", "Disable", "Force Offline", and "Remove". A "Add..." button is located in the top right corner of the table header.

Parsing the output

JSON Query Filters:

https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#json-query-filter

```
. . .<<Get output using bigip_device_facts and store in variable>>

- name: "View complete output"
  debug: "msg={{bigip_device_facts}}"

- name: "Show members belonging to pool"
  debug: "msg={{item}}"
  loop: "{{bigip_device_facts.ltm_pools | json_query(query_string) }}"
  vars:
    query_string: "[?name=='http_pool'].members[*].name[]"
```

Lab Time

Exercise 1.4 -Adding members to a pool on F5

Demonstrate use of the BIG-IP pool member module to tie web server nodes into the load balancing pool http_pool created in the previous exercises.

Approximate time: 15 mins

Using the F5 bigip_virtual_server module

```
- name: ADD VIRTUAL SERVER

  bigip_virtual_server:
<<login info removed for brevity>>

    name: "vip"
    destination: "{{private_ip}}"
    port: "443"
    enabled_vlans: "all"
    all_profiles: ['http','clientssl','oneconnect']
    pool: "http_pool"
    snat: "Automap"
```

F5 BIG-IP Web GUI

The virtual server can be found under Local Traffic -> Virtual Servers

The screenshot shows the F5 BIG-IP Web GUI interface. At the top, there's a header bar with the title "BIG-IP® - ip-172-16-26-136.ec2.internal", a timestamp ("Date: Aug 5, 2018 Time: 12:18 AM (UTC)"), and user information ("User: admin Role: Administrator"). On the right side of the header, there are links for "Partition: Common" and "Log out". Below the header is a navigation bar with tabs for "Main", "Help", and "About". The main content area is titled "Local Traffic > Virtual Servers : Virtual Server List". It features a search bar and a table with columns: Status, Name, Description, Application, Destination, Service Port, Type, Resources, and Partition / Path. A single row is visible in the table, showing a green status icon, the name "vip", the IP address "172.16.26.136", the service port "443 (HTTPS)", and the partition "Common". At the bottom of the table are buttons for "Enable", "Disable", and "Delete...". To the left of the main content area, there's a sidebar titled "Local Traffic" with a list of options: Network Map, Virtual Servers (which is currently selected and highlighted in yellow), Policies, Profiles, Ciphers, iRules, Pools, Nodes, Monitors, Traffic Class, and Address Translation. At the very bottom of the sidebar is a "Acceleration" section.

Lab Time

Exercise 1.5 -Adding a virtual server

Demonstrate use of the BIG-IP virtual server module to create a VIP (virtual IP). The VIP will be tied to the http_pool created in earlier exercises. Use a web browser to demonstrate the F5 load balancing between host1 and host2.

Approximate time: 15 mins

Using the F5 bigip_irule module

```
vars:  
  
    irules: ['irule1','irule2']  
  
  
tasks:  
- name: ADD iRules  
  
    bigip_irule:  
        <<login info removed for brevity>>  
        module: "ltm"  
        name: "{{item}}"  
        content: "{{lookup('file','{{item}}')}}"  
        with_items: "{{irules}}"
```

Lab Time

Exercise 1.6 -Adding a iRule

Demonstrate use of the BIG-IP irule module to upload irules to the BIG-IP and then attach those iRules to the Virtual Server created earlier .

Approximate time: 15 mins

Using the F5 bigip_config module

```
- name: SAVE RUNNING CONFIG ON BIG-IP
  bigip_config:
    server: "{{private_ip}}"
    user: "{{ansible_user}}"
    password: "{{ansible_ssh_pass}}"
    server_port: "8443"
    validate_certs: "no"
    save: yes
```

Lab Time

Exercise 1.7 - Saving running configuration

Demonstrate use of the BIG-IP config module to save the running BIG-IP configuration to disk

Approximate time: 15 mins

Using Provider

Use provider to avoid setting the connection details in every module, set it as a fact once as a task and then re-use it.

- provider

A dict object containing connection details.

suboptions:

password:

server:

server_port:

user:

validate_certs:

<<not a complete list>>

Using Provider Example

```
tasks:  
  - name: Setup provider  
    set_fact:  
      provider:  
        server: "{{private_ip}}"  
        user: "{{ansible_user}}"  
        password: "{{ansible_ssh_pass}}"  
        server_port: "8443"  
        validate_certs: "no"
```

```
- name: Query BIG-IP facts  
  bigip_device_facts:  
    provider: "{{provider}}"  
    gather_subset:  
      - ltm-pools  
    register: bigip_facts  
  
- name: SAVE RUNNING CONFIG  
  bigip_config:  
    provider: "{{provider}}"  
    save: yes
```

Operational Automation

- Dynamically grab node information from F5 BIG-IP
 - What pools are present?
 - What pool members are part of the pools and what are their IP addresses and Port numbers?
- Disable particular pool member or all pool members
- Verify with Web UI and Ansible Playbooks

Lab Time

Exercise 2.0 - Disabling a pool member

Demonstrate disabling of a node member:port from the pool.

Approximate time: 25 mins

Deleting with the F5 bigip_node module

```
- name: DELETE NODES  
  
bigip_node:  
  
    server: "{{private_ip}}"  
    user: "{{ansible_user}}"  
    password: "{{ansible_ssh_pass}}"  
    server_port: "8443"  
  
    validate_certs: "no"  
  
    name: "{{item}}"  
  
    state: absent
```

Using the **state** parameter with absent, the module will make sure the specified configuration is not existent (deleted)

Lab Time

Exercise 2.1 - Deleting F5 BIG-IP Configuration

Demonstrate use of the Ansible state parameter for modules. The state parameter will remove a configuration from the F5 BIG-IP load balancer.

Approximate time: 15 mins

Block

```
- name: BLOCK  
  block:  
    - debug:  
        msg: 'Task 1!'  
    - debug:  
        msg: 'Task 2!'  
    - debug:  
        msg: 'Task 3!'
```

Block

```
- name: BLOCK

block:
  - debug:
      msg: 'Task 1!'

  - debug:
      msg: 'Task 2!'

when:
  - ' "Xeon" in check_model'
  - ' "E5-2670" in check_model'
```

Block - Rescue

```
- name: Attempt and graceful roll back demo

block:
  - debug:
    msg: 'I execute normally'
  - command: /bin/false
  - debug:
    msg: 'I never execute, due to the above task failing'

rescue:
  - debug:
    msg: 'I caught an error'
  - command: /bin/false
  - debug:
    msg: 'I also never execute :-('
```

Block - Rescue

What happens when?

- If a task fails in the block, it will immediately go to **rescue**.
- If there is no **rescue** stanza, the Playbook will stop executing for the host it failed on.
- If there is a **rescue** stanza, the tasks under the rescue stanza will execute.
 - If any tasks under **rescue** fail, the Playbook will stop executing for the host it failed on.
 - If everything executes successfully under the **rescue** the Playbook will continue on like no failures happened. The failure will be recorded in the Play Recap.

Lab Time

Exercise 2.2 - Advanced: Error Handling

Demonstrate the use of the block and the rescue functionality for Ansible Playbooks. This exercise will also tie the previous exercises into one holistic Playbook.

Approximate time: 30 mins

Roles

Roles are Playbooks

- Roles help simplify playbooks.
- Think of them as callable functions for repeated tasks.
- Roles can be distributed/shared; similar to libraries.

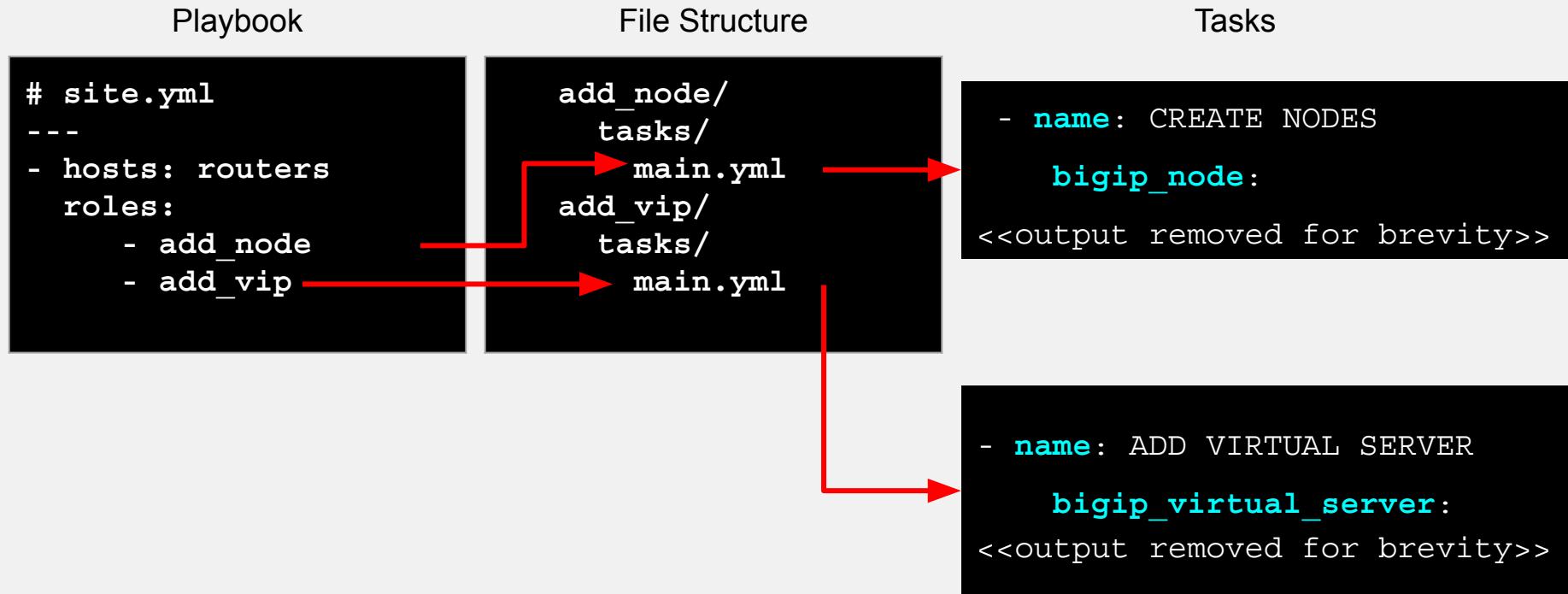
Example Playbook

```
# site.yml
---
- hosts: DC
  roles:
    - add_node
    - add_vip
```

Directory Structure

```
site.yml
roles/
  add_node/
    tasks/
      main.yml
  add_vip/
    tasks/
      main.yml
```

Roles - really simple, but powerful



Ansible Galaxy

<http://galaxy.ansible.com>

- Ansible Galaxy is a hub for finding, reusing and sharing Ansible roles.
- Jump-start your automation project with content contributed and reviewed by the Ansible community.

The screenshot shows the Ansible Galaxy search interface. The search bar at the top contains the keyword 'f5'. Below the search bar, there are filters for 'Keyword' (set to 'f5'), 'Best Match', and a dropdown menu showing '10 Results'. The main content area displays four search results, each with a thumbnail of the author, the role name, a brief description, and metrics like downloads, watchers, and stars. To the right of the search results is a sidebar titled 'Popular Tags' and another titled 'Popular Platforms'.

Tag	Count
system	4,987
development	2,493
web	2,118
monitoring	1,113
networking	891
database	872
cloud	802
packaging	676
docker	562
ubuntu	540

Platform	Count
Ubuntu	65,987

App Services 3 Extension (AS3)

The Application Services 3 Extension uses a declarative model, meaning you send a declaration file using a single Rest API call.



Simple JSON

- Declaration not ordered, nor sequenced
- Variables can be used easily within the AS3 template
- Incremental Declaration capable

```
"web_app": {
    "class": "Application",
    "template": "http",
    "serviceMain": {
        "class": "Service_HTTP",
        "virtualAddresses": [
            "{{private_ip}}"
        ],
        "pool": "app_pool"
    },
    "app_pool": {
        "class": "Pool",
        "monitors": [
            "http"
        ],
        "members": [
<<snippet, output removed for brevity>>>
```

Pushing a Template

Module coming in Ansible 2.7 (Today!)

```
- name: PUSH AS3
uri:
  url: "https://{{ ansible_host }}:8443/mgmt/shared/appsvcs/declare"
  method: POST
  body: "{{ lookup('template','j2/tenant_base.j2', split_lines=False) }}"
  status_code: 200
  timeout: 300
  body_format: json
  force_basic_auth: yes
  user: "{{ ansible_user }}"
  password: "{{ ansible_ssh_pass }}"
  validate_certs: no
```

Lab Time

Exercise 3.0 - Intro to AS3

Demonstrate building a virtual server (exactly like the Section 1 Ansible F5 Exercises) with F5 AS3

Approximate time: 15 mins

Lab Time

Exercise 3.1 - Operational Change with AS3

Demonstrate changing an existing Web Application AS3 template. There is a problem with the existing template, the serviceMain is showing red. What is wrong?

Approximate time: 15 mins

Lab Time

Exercise 3.2 - Deleting a Web Application

Demonstrate deleting a Web Application with AS3 and the uri module.

Approximate time: 15 mins

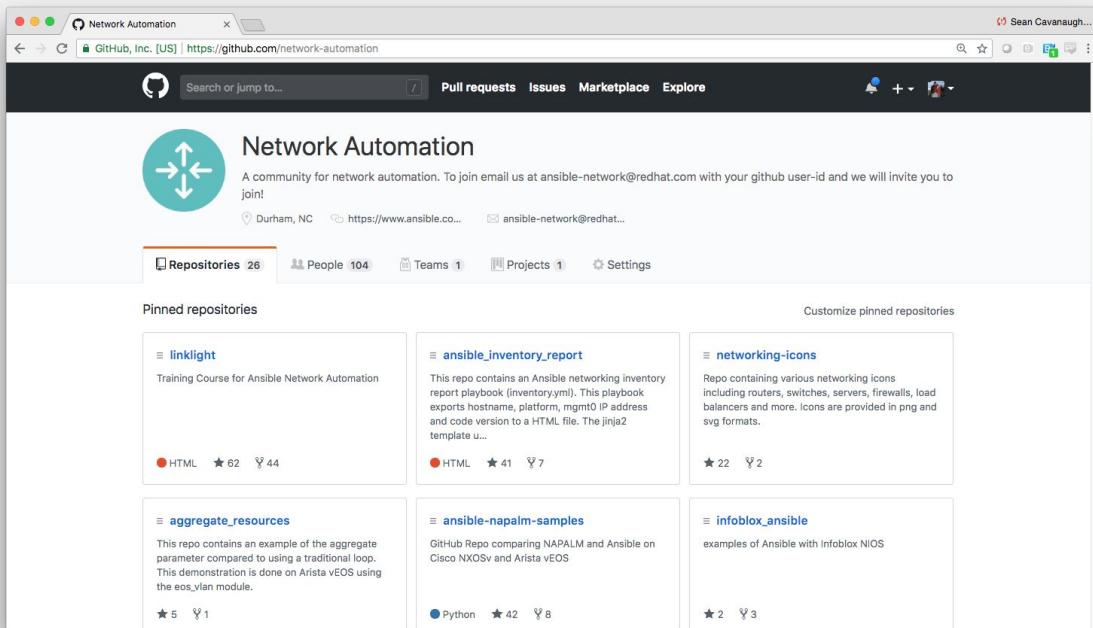
Next Steps

Thanks so much for joining the class. Here are some next steps on how to get more information and join the community!

Bookmark the GitHub Project

<https://www.github.com/network-automation>

- Examples, samples and demos
- Run network topologies right on your laptop



Chat with us

Engage with the community

- **Slack**

<https://ansiblenetwork.slack.com>

Join by clicking here <https://bit.ly/2OfNEBr>

- **IRC**

#ansible-network on freenode

<http://webchat.freenode.net/?channels=ansible-network>

Next Steps

- It's easy to get started

<https://ansible.com/get-started>

- Learn about Ansible & F5

<https://ansible.com/f5>

- Instructor Led Classes

Class D0457: Ansible for Network Automation

<https://red.ht/2MiAgvA>

