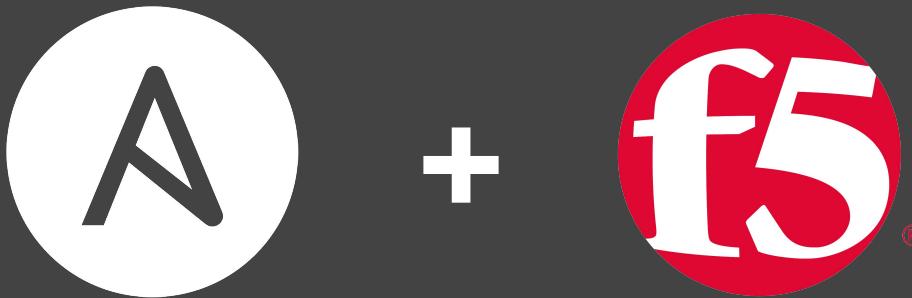


Ansible F5 Workshop



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.**

Managing networks hasn't changed in 30 years

- Networks are mission critical
- Every network is a unique snowflake
- Ad-hoc changes that proliferate
- Vendor specific implementations
- Testing is expensive/impossible

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)

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.



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

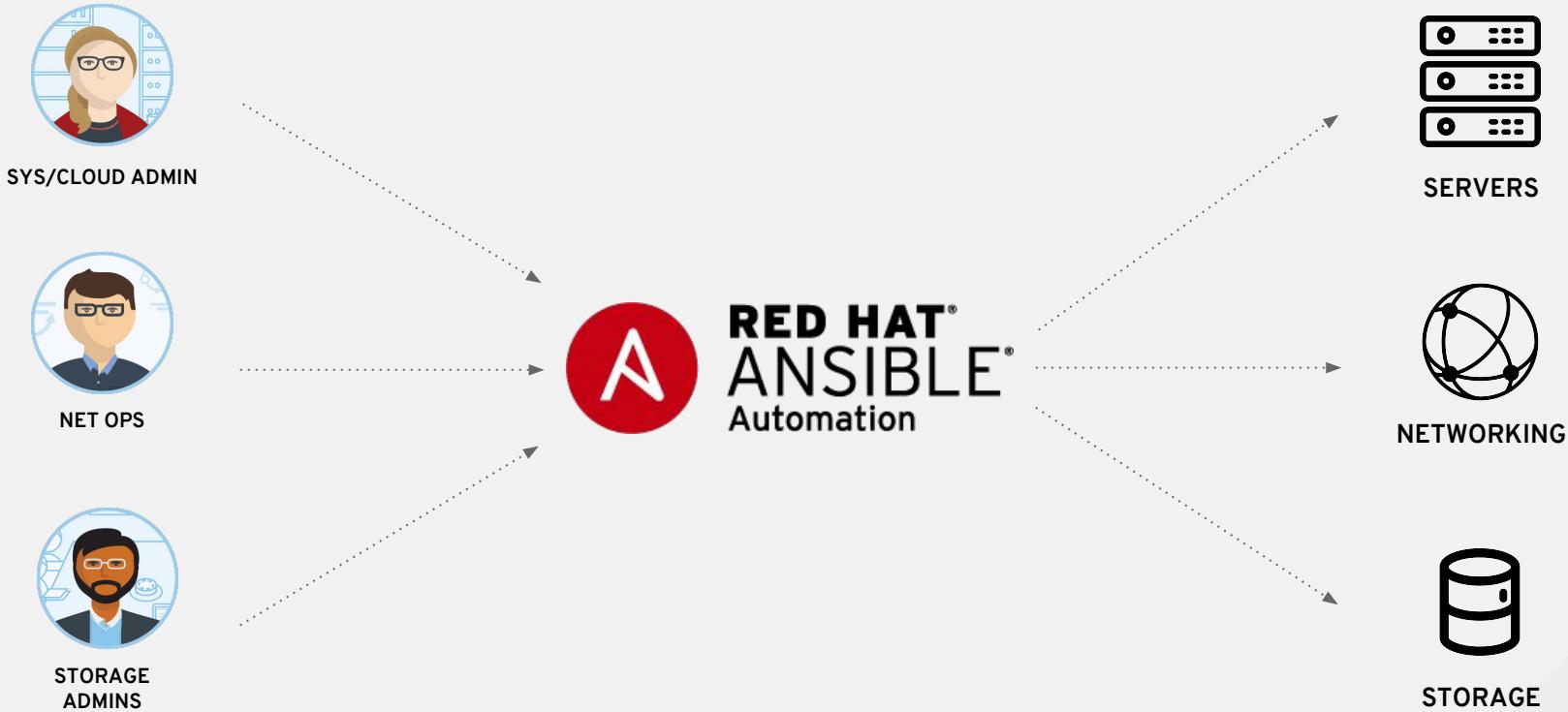
Agentless architecture

Uses OpenSSH and paramiko

No agents to exploit or update

More efficient & more secure

Ansible: The Universal Automation Framework



ANSIBLE NETWORK AUTOMATION

45

Networking
platforms

630+

Networking
modules

ansible.com/networking

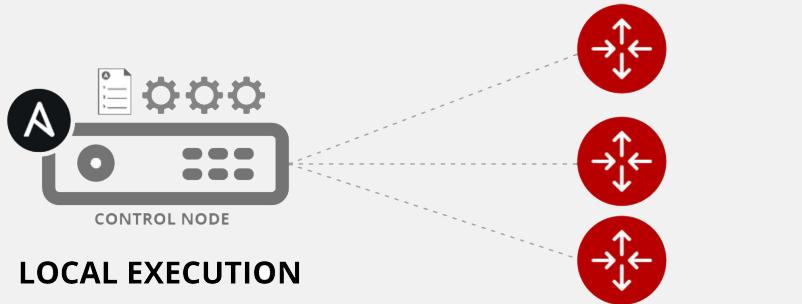
Common use cases

- 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.

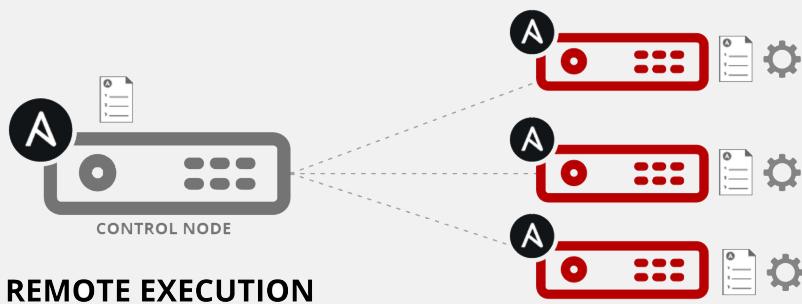
How Ansible Works

Module code is executed locally on the control node

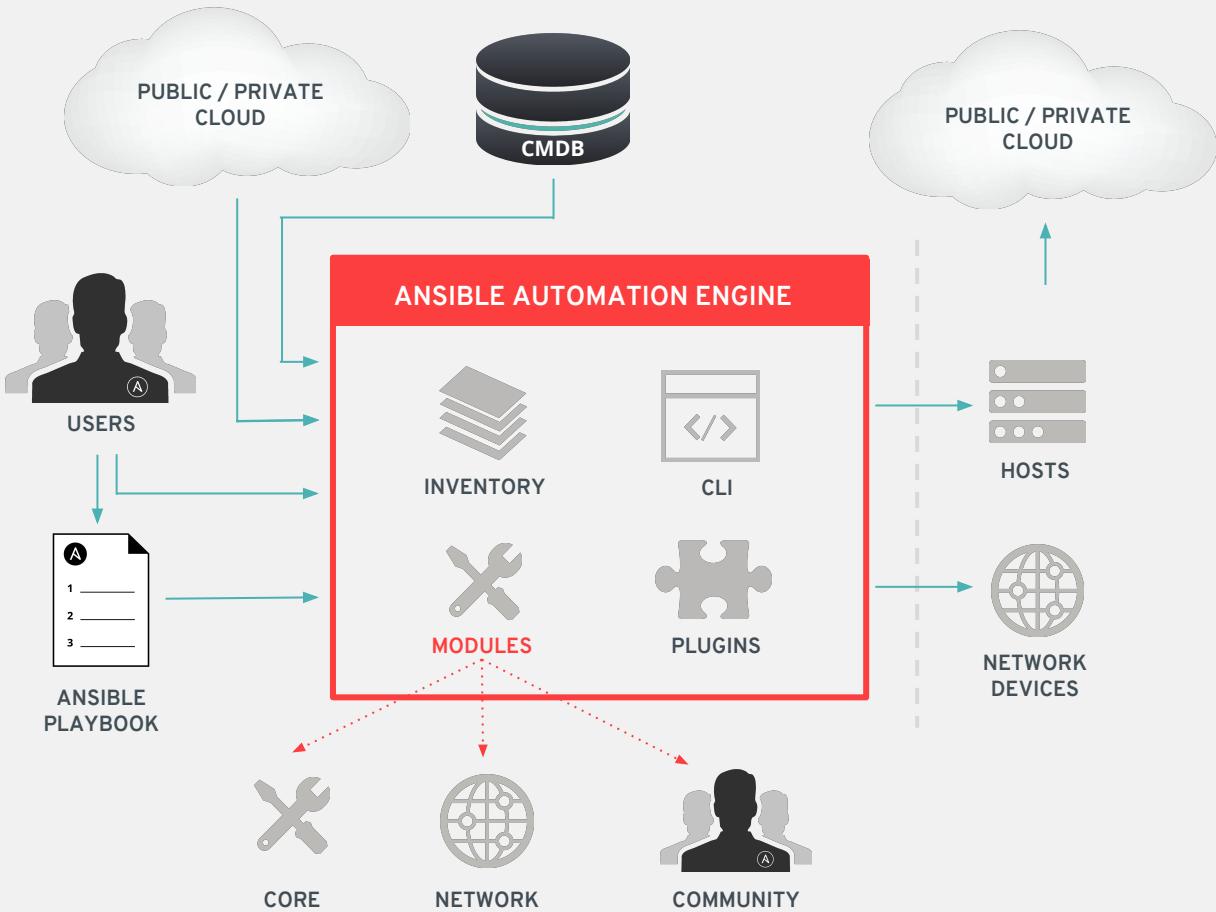


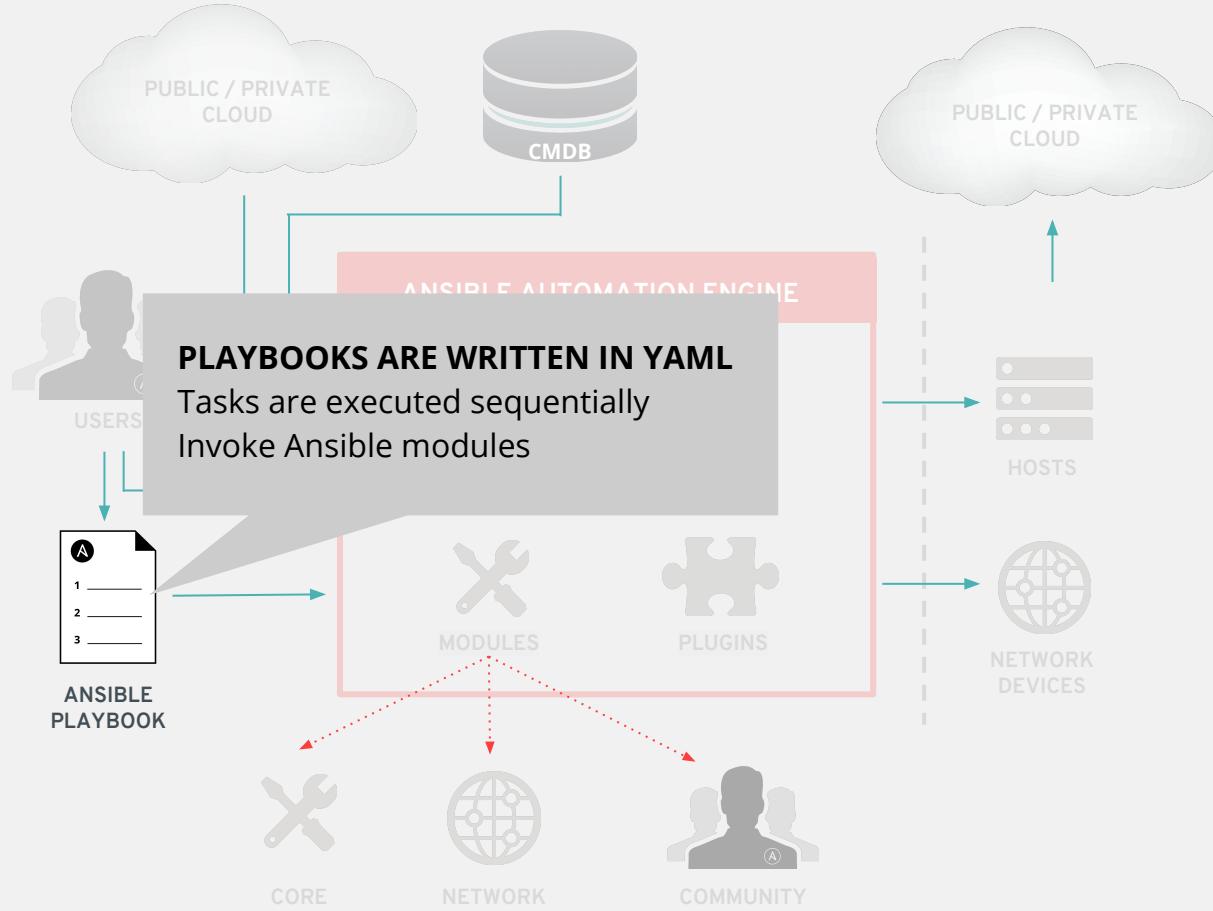
NETWORKING DEVICES

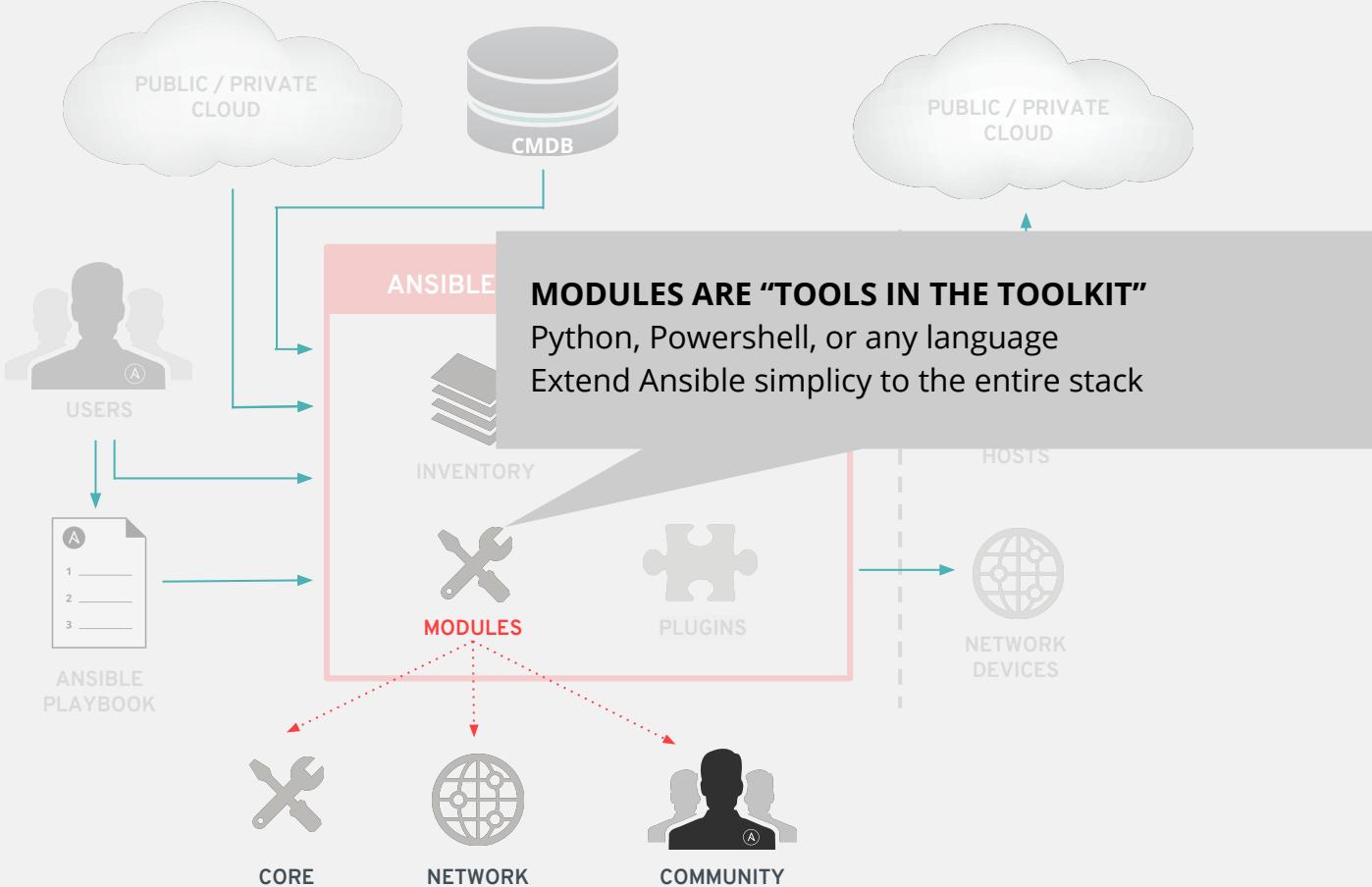
Module code is copied to the managed node, executed, then removed

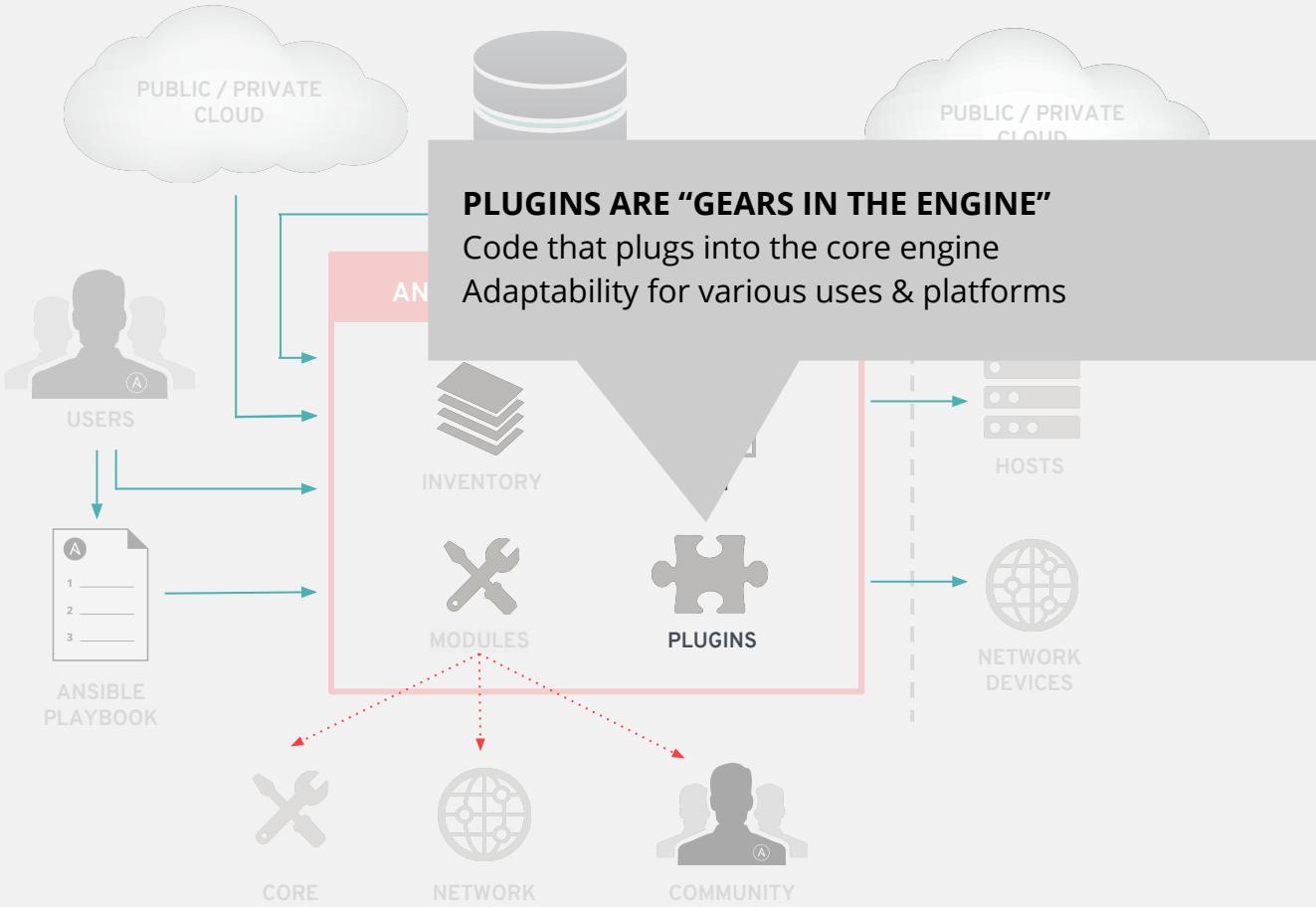


LINUX/WINDOWS HOSTS









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-playbook bigip-facts.yml

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-playbook bigip-facts.yml --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

```
"bigip_facts": {  
    "ansible_facts": {  
        "system_info": {  
            "base_mac_address": "0A:D1:27:C1:84:76"  
        }  
    }  
}
```

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

0A:D1:27:C1:84:76

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_facts:
    include: system_info
    server: "{{private_ip}}"
    user: "{{ansible_user}}"
    password: "{{ansible_ssh_pass}}"
    server_port: 8443
  register: bigip_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_facts['ansible_facts']['system_info']['base_mac_address']
```

```
TASK [DISPLAY ONLY THE MAC
ADDRESS] ****
ok: [f5] => {
    "bigip_facts['ansible_facts']['system_info']['base_mac_address']":
"0A:D1:27:C1:84:76"
}
```

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-playbook bigip-facts.yml --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: debug
```

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

```
[user@ansible]$ ansible-playbook bigip-facts.yml --skip-tags=debug
```

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://docs.ansible.com/>

Docs » Module Index

Module Index

- [All Modules](#)
- [Cloud Modules](#)
- [Clustering Modules](#)
- [Commands Modules](#)
- [Crypto Modules](#)
- [Database Modules](#)
- [Files Modules](#)
- [Identity Modules](#)
- [Inventory Modules](#)
- [Messaging Modules](#)
- [Monitoring Modules](#)
- [Network Modules](#)
- [Notification Modules](#)
- [Packaging Modules](#)
- [Remote Management Modules](#)
- [Source Control Modules](#)
- [Storage Modules](#)
- [System Modules](#)
- [Utilities Modules](#)
- [Web Infrastructure Modules](#)
- [Windows Modules](#)

service - Manage services.

▪ Synopsis
▪ Options
▪ Examples
 ◦ Status
 ◦ Support

Synopsis

▪ Controls services on remote hosts. Supported init systems include BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.

Options

parameter	required	default	choices	comments
arguments	no			Additional arguments provided on the command line
alias	yes			alias.eap
enabled	no		• yes • no	Whether the service should start on boot. At least one of state and enabled are required.
name	yes			Name of the service
pattern	no			If the service does not respond to the status command, name is substring to look for as would be found in the output of the ps command as a stand-in for a status result. If the service is running, it will be assumed to be running.
runlevel	no	default		For OpenRC init scripts (on Gentoo only) the runlevel this service belongs to.
sleep	no	(added in 1.8)		If the service is being <code>restart</code> , then sleep this many seconds between the stop and start commands. This helps to workaround badly behaving init scripts that exit immediately after stopping.
state	no		• started • stopped • restarted • reloaded	<code>started</code> / <code>stopped</code> are idempotent actions that will not run commands unless necessary. <code>restart</code> will always bounce the service. <code>reload</code> will always reload. At least one of state or enabled must be specified. If both are specified and the service is not already started, even if your chosen init system wouldn't normally.
use	no	auto		The service module actually uses system specific modules, normally through auto detection, but you can force a specific module. Normally it uses the value of the <code>ansible_service_mgr</code> fact and falls back to the old <code>service</code> module when none matching is found.

Modules Documentation

Documentation right on the command line

```
[user@ansible]$ ansible-doc bigip_facts
```

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

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

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

```
- filter
```

```
    Shell-style glob matching string used to filter fact keys. Not applicable for software, provision, and system_info fact categories.
```

```
.
```

```
.
```

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 host information: 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 "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, and Logout. The Local Traffic section is currently selected, showing sub-links for Network Map, Virtual Servers, Policies, Profiles, Ciphers, iRules, Pools, 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 located at the top of the pool list table. The table itself has columns for Status, Name, Description, Application, Members, and Partition / Path. One entry, "http_pool", is listed with a status of "ONLINE (ACTIVE)", 0 members, and the partition "Common".

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 "Load Balancing" section with "Load Balancing Method" set to "Round Robin" and "Priority Group Activation" set to "Disabled". The "Current Members" table lists two members: "host2:80" with address 54.146.162.192 and service port 80, and "host1:80" with address 107.22.141.4 and service port 80. Both members are marked as "0 (Active)" and belong to the "Common" priority group. At the bottom of the table are buttons for "Enable", "Disable", "Force Offline", and "Remove". A tooltip from a large black arrow points to the "Members" tab in the navigation bar.

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

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

A screenshot of the F5 BIG-IP Web GUI interface. The title bar shows "BIG-IP® - ip-172-16-26-136.ec2.internal" and the URL "https://34.199.128.69:8443/xui/". The top navigation bar includes "Hostname: ip-172-16-26-136.ec2.internal", "Date: Aug 5, 2018", "Time: 12:18 AM (UTC)", "User: admin", "Role: Administrator", "Partition: Common", and "Log out". A user "Sean Cavanaugh..." is logged in. The main menu on the left has tabs for "Main", "Help", and "About". Under "Local Traffic", the "Virtual Servers" tab is selected, showing a list of virtual servers. The list table has columns for Status, Name, Description, Application, Destination, Service Port, Type, Resources, and Partition / Path. One entry is visible: "vip" with status "ONLINE (ACTIVE)", destination "172.16.26.136", service port "443 (HTTPS)", type "Standard", resources "Common", and partition "Common". There are "Create...", "Enable", "Disable", and "Delete..." buttons at the bottom of the list.

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

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 1.6 - 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 1.7 - 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

Operational Automation

- Dynamically grab node information from F5 BIG-IP
 - What nodes are online?
 - What are their IP addresses?
- Remove Node
- Verify with Web UI or Ansible Playbooks

Lab Time

Exercise 2.0 - Disabling a pool member

Demonstrate the removal of a node from the pool.

Approximate time: 25 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

Playbook

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

File Structure

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

Tasks

```
- name: CREATE NODES
  bigip_node:
    <<output removed for brevity>>

- name: ADD VIRTUAL SERVER
  bigip_virtual_server:
    <<output removed for brevity>>
```

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 web interface. The URL in the address bar is https://galaxy.ansible.com/search?keywords=f5&order_by=-relevance&page_size=10. The search bar contains the keyword 'f5'. There are 10 results listed:

- module-f5bigip role** by erjac77 (66 Downloads, 5 Watchers, 3 Forks, Last Imported a month ago, Best Match 0.3231)
- bigip-ansible-virtual... role** by f5devcentral (32 Downloads, 5 Watchers, 0 Forks, Last Imported 7 months ago, Best Match 0.3110)
- bigip-toggle-nodeSta... role** by f5devcentral (27 Downloads, 5 Watchers, 0 Forks, Last Imported 7 months ago, Best Match 0.3077)
- rancid-git role** by rancid (RANCID 2 fork with Git support. Also includes customizations)

On the right side, there are two sections of popular tags and platforms:

- Popular Tags**: system (4,987), development (2,493), web (2,118), monitoring (1,113), networking (891), database (872), cloud (802), packaging (676), docker (562), ubuntu (540).
- Popular Platforms**: 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

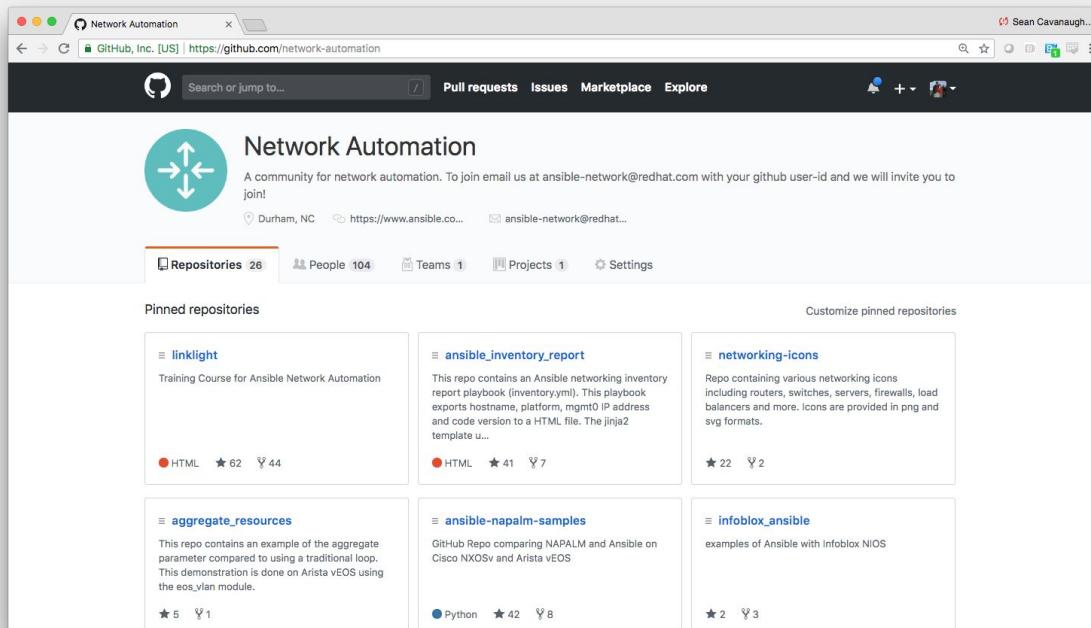
Next Steps

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>

