# Ansible F5 Agility 2021 Workshop

## Overview

We make every effort to ensure the lab guide has step by step instructions to complete the tasks, however some topics are not covered in depth due to a lack of time. Throughout the labs we hope to share features, elements, good and bad practices to using Ansible Tower for Automation.

You will be required to modify some files during the course of this workshop. You will not be required to write your own playbooks. The playbooks used are open source and thus free to use and modify. That being said, writing playbooks and running them in a test environment is the best way to learn.

***Outline***

* Part 1: Getting setup
  + Fork Git Repo
  + Connecting to your student F5
* Part 2: Tower lab 1 (Imperative)
  + Connect to Ansible Tower
  + Explore Ansible Tower
  + Create a project
  + Create an inventory
  + Create job templates
  + Create a workflow
  + Add rescue logic to a workflow
* Part 3: Tower lab 2 – AS3 (Declarative)
  + Overview of AS3 Ansible Components
  + Install AS3 plugin
  + Create job templates
  + Create a workflow

**Note: Assume the output shown in the examples below will be different to yours.**

## Part 1: Getting setup

### ***Overview***

* Fork the lab github repository to your own repository so you can edit and modify.

## *Fork the Sirius ansible networking GitHub repository*

### Step 1

## Login in to Github at <https://github.com>

### Step 2

## Go to <https://github.com/mysidlabs/f5-agility-2021-tower-labs>

### Step 3

## Click on the Fork button in upper right.

## Graphical user interface, text, application Description automatically generated

## Note: Once forked you can modify all files within GitHub

*Connecting to your F5*

### Step 1

## Open your web browser (Chrome or Firefox)

### Step 2

Put the address of your F5 in the address bar including the port number 8443. NOTE: Your student F5 will have a different number!

## https://siduser155.f5.mysidlabs.com:8443

### Step 3

## Proceed through the warnings regarding the SSL certificate not being valid

## Note: Some browsers will not allow you to connect/bypass the security warning due to settings and the fact that the SSL certificate is a self-signed certificate and cannot be validated. You may have to change browsers in order to bypass the security warning.

### Step 4

Record the F5 Private IP address

A screenshot of a social media post

Description automatically generated

### Step 5

## Login into the F5 BigIP

## Username: admin

## Password: Mys1dlabspw!

## Part 2: Basic F5 labs

### Topology

The topology is simple for the sake of learning some ansible basics. The diagram below is an example, the XXX in the hostname is your Student ID. If you are student 199 then the hostname for the F5 would be siduser199.f5.mysidlabs.com.

Diagram

Description automatically generated

## Part 2: Tower lab 1

Lab 4.0: Explore Ansible Tower

### Step 1:

Open web browser and go to [https://tower.mysidlabs.com](https://tower.mysidlabs.com/) and enter in your username and password.

A screenshot of a cell phone

Description automatically generated

### Step 2:

Once logged in explore the interface

A screenshot of a computer

Description automatically generated

### Step 3:

Click on the “i” information button on the top right of the user interface.



You will get something similar to the following:

A picture containing bird, flower

Description automatically generated

### Step 4:

Click on the Inventories Tab on the left side of the page

A close up of a logo

Description automatically generated

Click on Network-Lab-Instructor-Inventory



You will see the following

A screenshot of a cell phone

Description automatically generated

### Step 5:

Click on the Projects Tab on the left side of the page. Note that projects point to your source code repository where your playbooks exist.



Click on any of the available project. Below I selected Instructor-ansible-network-labs as an example, but it may not exist when you do this.



You will see the following

A screenshot of a cell phone

Description automatically generated

### Step 6:

Click on the Credentials Tab on the left side of the page

A picture containing knife

Description automatically generated

Type network in the search field and click on the magnifying glass icon

A screenshot of a cell phone

Description automatically generated

Click on network-lab-key



You will see the following

A screenshot of a cell phone

Description automatically generated

### Step 7:

Click on the Templates Tab on the left side of the page

A picture containing table

Description automatically generated

Click on any of the available templates. Below I selected Instructor-SNMP as an example, but it may not exist when you do this.



You will see the following

A screenshot of a computer

Description automatically generated

### Lab 4.1: Create an Ansible Tower job template

### Step 1:

Create a Credential

Click on Credentials section using the left navigation bar.

A close up of a logo

Description automatically generated

Click on the green plus button on the right-hand side to create a new Credential

templates link

Fill out the following fields as follows, and click SAVE

|  |  |
| --- | --- |
| Name | <<siduserID>>.f5.mysidlabs.com |
| Credential Type | Network |
| Username | admin |
| Password | Mys1dlabspw! |

A screenshot of a cell phone

Description automatically generated

### Step 2:

Create a Project

Click on Projects section using the left navigation bar.

A picture containing table

Description automatically generated

Click on the green plus button on the right-hand side

templates link

Fill out the following fields as follows, and click SAVE

|  |  |
| --- | --- |
| Name | <<siduserID>>.f5.mysidlabs.com |
| Organization | sid-org |
| SCM Type | git |
| SCM URL | https://github.com/<<your-github-user>>/f5-agility-2021-tower-labs |
| Clean/Delete on Update/  Update Revision on Launch | Checked |

A screenshot of a cell phone

Description automatically generated

### Step 3:

Create an Inventory

Click on Inventories section using the left navigation bar.

A close up of a logo

Description automatically generated

Click on the green plus button on the right-hand side and click Inventory to create a new inventory

A picture containing clock

Description automatically generated

Fill out the following fields and click SAVE

|  |  |
| --- | --- |
| Name | <<siduserID>>.f5.mysidlabs.com |
| Organization | sid-org |

A screenshot of a cell phone

Description automatically generated

Click on the Sources tab, click the green plus button

A screenshot of a cell phone

Description automatically generated

Fill out the following fields and click SAVE

|  |  |
| --- | --- |
| Name | <<siduserID>>.f5.mysidlabs.com |
| Source | Amazon EC2 |
| Credentials | aws-api |
| Regions | US East Ohio |
| Instance Filters | tag:Name=F5-Lab-<<siduserID>>  ex: tag:Name=F5-Lab-siduser155 |
| Only Group By | Tags |
| Overwrite/Update on Launch | Checked |

A screenshot of a social media post

Description automatically generated

Click on the Sources link at the top left



Click on the green plus button on the right-hand side to add another source

A screenshot of a cell phone

Description automatically generated

Fill out the following fields and click SAVE

|  |  |
| --- | --- |
| Name | <<siduserID>>.web.mysidlabs.com |
| Source | Amazon EC2 |
| Credentials | aws-api |
| Regions | US East Ohio |
| Instance Filters | tag:Name=F5-Web\* |
| Only Group By | Tags |
| Overwrite/Update on Launch | Checked |

A screenshot of a social media post

Description automatically generated

Click on Sources > Sync All

Graphical user interface, text, application, email

Description automatically generated

You should have green clouds next to your sources once sync is complete

Click on the <<siduserID>>.f5.mysidlabs.com link at the top left



Click on Hosts and verify you have 3 hosts

A screenshot of a cell phone

Description automatically generated

### Step 4:

Create a Job Template

Click on Inventories section using the left navigation bar.



Click on the green plus button on the right-hand side and click Job Template to create a new inventory

A screenshot of a cell phone

Description automatically generated

Fill out the following fields and click SAVE and then LAUNCH

|  |  |
| --- | --- |
| Name | <<siduserID>>.f5.mysidlabs.com |
| Job Type | Run |
| Inventory | <<siduserID>>.f5.mysidlabs.com |
| Project | <<siduserID>>.f5.mysidlabs.com |
| Playbook | 4.1-bigip-create-vs.yaml |
| Credentials | Select Credential Type: Network  <<siduserID>>.f5.mysidlabs.com |

A screenshot of a social media post

Description automatically generated

Examine the Details pane

A screenshot of a cell phone

Description automatically generated

### Step 4:

Verifying that the playbook did what you expected. Login to the F5 with your web browser to see what was configured.

Click on the Local Traffic on the left-hand menu

Click on Virtual Servers.

The Virtual Server will be displayed.

This time it will be Green (Available (Enabled) - The virtual server is available)

Check Pools for app\_pool that both web servers are set to port 80 for their service\_port

If everything looks good from an F5 standpoint, verifying the web servers

Open up the public IP of the F5 load balancer in your web browser:

https://<<siduserID>>.f5.mysidlabs.com/

Each time you refresh the host will change between web1 and web2.

Alternate Verification Method

You can use the curl command on the jump station to access public IP.

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

**<< output omitted >>**

## Part 4: F5 AS3 labs

Lab 3.0: Intro to AS3

### Step 1:

**Make sure the BIG-IP configuration is clean, run exercise 2.1 delete-configuration before proceeding**

siduser250@toolkit ~/ansible-f5-labs # **ansible-playbook 2.1-bigip-delete-configuration.yaml**

**<< output omitted >>**

### Step 2:

Download AS3 RPM at

https://github.com/F5Networks/f5-appsvcs-extension/releases/download/v3.22.0/f5-appsvcs-3.22.0-2.noarch.rpm

### Step 3:

Install AS3

Login to the F5 BIG-IP through your web browser.

Click on the iApps button, Click the Package Management LX, Click the Import… button

A picture containing screenshot

Description automatically generated

Click Browse… button, Select the RPM, Click the Upload button, then wait for upload to finish

A screenshot of a social media post

Description automatically generated

Make sure that f5-appsvcs is installed. You may need too refresh the screen a few times.

A picture containing screenshot

Description automatically generated

If this is not working, please ask your instructor for help.

### Step 4:

Before starting we look at the Playbooks, it is important to understand how AS3 works. AS3 requires a JSON template to be sent as an API call to the F5. You do not need to fully understand every parameter or create these templates from scratch.

Look at the file called 30\_as3\_template.j2

siduser250@toolkit ~/ansible-f5-labs **# cat 30\_as3\_template.j2**

{

"class": "AS3",

"action": "deploy",

"persist": false,

"declaration": {

"class": "ADC",

"schemaVersion": "3.22.0",

"id": "123abc",

"label": "Sample 2",

"remark": "HTTPS with round robin pool",

"{{ tenant\_name }}": {

"class": "Tenant",

"A1": {

"class": "Application",

"service": {

"class": "Service\_HTTPS",

"virtualAddresses": [

"{{ private\_ip }}"

],

"profileMultiplex": {

"bigip": "/Common/oneconnect"

},

"pool": "web\_pool",

"persistenceMethods": [],

"serverTLS": "webtls"

},

"web\_pool": {

"class": "Pool",

"loadBalancingMode": "round-robin",

"monitors": [

"http"

],

"members": [{

"servicePort": 443,

"serverAddresses": [

{% set comma = joiner(",") %}

{% for mem in pool\_members %}

{{comma()}} "{{ hostvars[mem]['ansible\_host'] }}"

{% endfor %}

]

**<< output omitted >>**

This template is a JSON representation of the Web Application. The important parts to note are:

The template is a Jinja2 template which allows us to use variables. In this case we have a few variable such as "{{ tenant\_name }}" and "{{ private\_ip }}"

The jinja2 template can use loops to grab all the pool members (which points to our web servers group).

### Step 4:

Look at the file called 3.0-as3.yaml

siduser250@toolkit ~/ansible-f5-labs **# cat 3.0-as3.yaml**

---

- name: AS3

hosts: lb

connection: local

gather\_facts: false

vars:

pool\_members: "{{ groups['webservers'] }}"

vars\_prompt:

- name: "tenant\_name"

prompt: "Enter the Tenant name:"

private: no

default: "WorkshopExample"

tasks:

- name: Push AS3

uri:

url: "https://{{ ansible\_host }}:8443/mgmt/shared/appsvcs/declare"

method: POST

body: "{{ lookup('template', '30\_as3\_template.j2', split\_lines=False) }}"

status\_code: 200

timeout: 300

body\_format: json

force\_basic\_auth: true

user: "{{ username }}"

password: "{{ lb\_password }}"

validate\_certs: false

### Step 5:

Review the output and see what information is in the output.

**connection:** - local tells the Playbook to run locally (rather than SSHing to itself)

**uri:** - this task is calling the uri module

**url: "https://{{ ansible\_host }}:8443/mgmt/shared/appsvcs/declare"** - webURL (API) for AS3

**method: POST** - HTTP method of the request, must be uppercase. Module documentation page has list of all options.

**body: "{{ lookup('template','j2/ as3\_template.j2', split\_lines=False) }}"** - This passed the template as the body for the API request

**status\_code: 200** - A valid, numeric, HTTP status code that signifies success of the request. This can be comma separated list of status codes. In this instance we use 200 which means OK, this is a standard response for successful HTTP requests

Step 3:

Run the playbook:

siduser250@toolkit ~/ansible-f5-labs # **ansible-playbook 3.0-as3.yaml**

**<< output omitted >>**

Step 4:

Login to the F5 with your web browser to see what was configured.

Click on the Local Traffic on the left-hand menu

Click on Virtual Servers.

On the top right, click on the drop-down menu titled Partition and select WorkshopExample, or whatever tenant name you used

The Virtual Server will be displayed.

Check Pools for app\_pool

Notice that it is not working.

### Lab 3.1: Operational changes with AS3

Step 1:

Login to the F5 with your web browser to see what was configured.

Click on the Pools under Local Traffic

Click on web\_pool

Click on the Members button

Look at the Service Port for each web server

The port 443 is incorrect. The two web servers are only running on port 80. This is why they are showing down.

### Step 2:

We need to get this up and working so we need to make sure that we are using port 80 not 443. Start by looking at the file called 31\_as3\_template.j2

siduser250@toolkit ~/ansible-f5-labs **# cat 31\_as3\_template.j2**

**<< output omitted >>**

"web\_pool": {

"class": "Pool",

"loadBalancingMode": "round-robin",

"monitors": [

"http"

],

"members": [{

**"servicePort": 80,**

"serverAddresses": [

{% set comma = joiner(",") %}

{% for mem in pool\_members %}

{{comma()}} "{{ hostvars[mem]['ansible\_host'] }}"

{% endfor %}

]

**<< output omitted >>**

While understanding the F5 AS3 JSON is out of scope for this class we can see that the servicePort parameter under the members section is set to port 80 instead of port 443 like it was in the 3.0 lab. To find out more about the AS3 formatting an option you can look at the F5 documentation:

<https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/>

### Step 3:

Look at the file called 3.1-as3.yaml

siduser250@toolkit ~/ansible-f5-labs **# cat 3.1-as3.yaml**

**<< output omitted >>**

### Step 4:

Run the playbook:

siduser250@toolkit ~/ansible-f5-labs # **ansible-playbook 3.1-as3.yaml**

Step 5:

Verifying that the playbook did what you expected. Login to the F5 with your web browser to see what was configured.

Click on the Local Traffic on the left-hand menu

Click on Virtual Servers.

On the top right, click on the drop-down menu titled Partition and select WorkshopExample, or whatever tenant name you used

The Virtual Server will be displayed.

This time it will be Green (Available (Enabled) - The virtual server is available)

Check Pools for app\_pool that both web servers are set to port 80 for their service\_port

If everything looks good from an F5 standpoint, verifying the web servers

Open up the public IP of the F5 load balancer in your web browser:

https://<<siduserID>>.f5.mysidlabs.com/

Each time you refresh the host will change between web1 and web2.

Alternate Verification Method

You can use the curl command on the jump station to access public IP

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

siduser250@toolkit ~/ansible-f5-labs # curl https:// <<siduserID>>.f5.mysidlabs.com:443 --insecure

**<< output omitted >>**

## Lab 3.2: Deleting a web application

### Step 1:

Look at the file called 3.2-as3-delete.yaml

siduser250@toolkit ~/ansible-f5-labs **# cat 3.2-as3-delete.yaml**

---

- name: AS3

hosts: lb

connection: local

gather\_facts: false

vars\_prompt:

- name: "tenant\_name"

prompt: "Enter the Tenant name:"

private: no

default: "WorkshopExample"

tasks:

- name: Remove Tenant AS3

uri:

url: "https://{{ ansible\_host }}:8443/mgmt/shared/appsvcs/declare/{{ tenant\_name }}"

method: DELETE

status\_code: 200

timeout: 300

body\_format: json

force\_basic\_auth: true

user: "{{ username }}"

password: "{{ lb\_password }}"

validate\_certs: false

### Step 2:

Review the output and see what information is in the output.

**method: DELETE** – this tells AS3 to delete the tenant

Step 3:

Run the playbook:

siduser250@toolkit ~/ansible-f5-labs # **ansible-playbook 3.2-as3-delete.yaml**

**<< output omitted >>**

Step 4:

Verifying that the playbook did what you expected. Login to the F5 with your web browser to see that the configuration was saved.

## Success - Congratulations.

## Picture 2095273455

**Appendix A:**

Useful resource links and information

Links:

Ansible Best Practices

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html>

Ansible Network troubleshooting

<https://docs.ansible.com/ansible/latest/network/user_guide/network_debug_troubleshooting.html>

Ansible cli\_command module information

https://www.ansible.com/blog/deep-dive-on-cli-command-for-network-automation

Variable precedence

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable>

Additional Notes:

* Remember YAML is very sensitive to correct indentation
* **Hostvars** allow us to access meta-data about our inventory hosts.
* The use of an Ansible role is best practice when there is a well-defined scope with a high possibility of re-use.
* If you copy and paste text for a playbook you may get indentation issues. Ansible provides a simple syntax checker, try ansible-playbook --syntax-check backup.yml to verify. A Best Practice is to use a linter, for example ansible-review. Ansible provides excellent online documentation, which is also available from the command line, for example ansible-doc ios\_config. For a full list of modules try ansible-doc –l
* There where multiple ways of implementing a playbook where specific tasks or groups of tasks execute against specific hosts. For example, we could have used 1 playbook for configuring every router in the lab utilizing the “when:” statement to ensure specific tasks are only applied to a specific router. Although this is not necessarily following best practices.
* The use of handlers: which can be used in any playbook. A handler is a special way of calling a task whenever an action needs to be taken after a previous task. For example, both installing and configuring an application may require a restart. A handler would be notified by both tasks but would only run once when the playbook finishes.