f5-rs-docs Documentation

Release latest

Dec 31, 2020

User Guide

This is a community project. it is not maintained or sponsered by F5. use at your own responsability !

designed to provide a common framework for deploying and developing F5 solutions as code.

You can use these modules to create, edit, update, and delete configuration objects on BIG-IP or cloud infrastructure.

CHAPTER 1

About

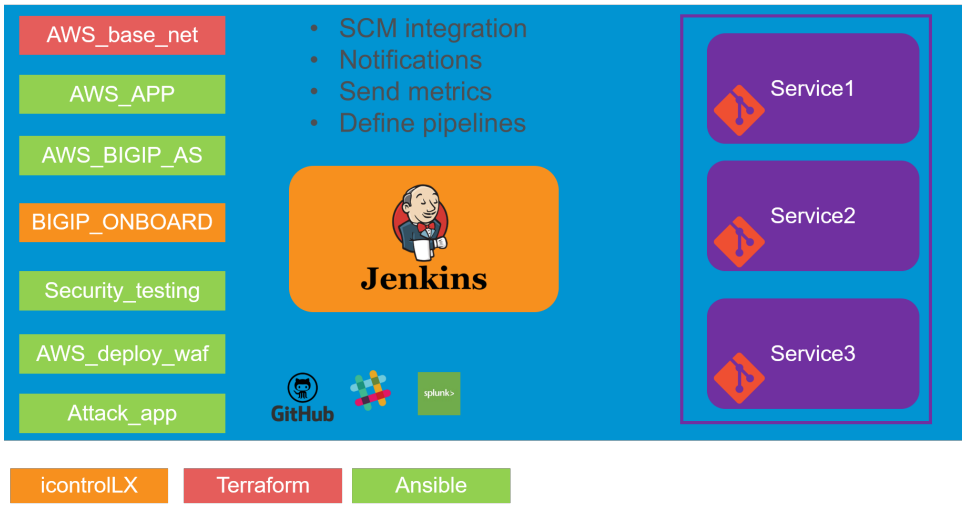
1.1 General

This project is a community driven effort to enable F5 users to build/experiment/test F5 services in their own cloud environments. The intent is to make it easier and faster to test the advanced security/ADC services offered by F5 by delivering modular pieces of automation code/scripts.

1.2 About the framework

The f5-rs framework is built from the following components:

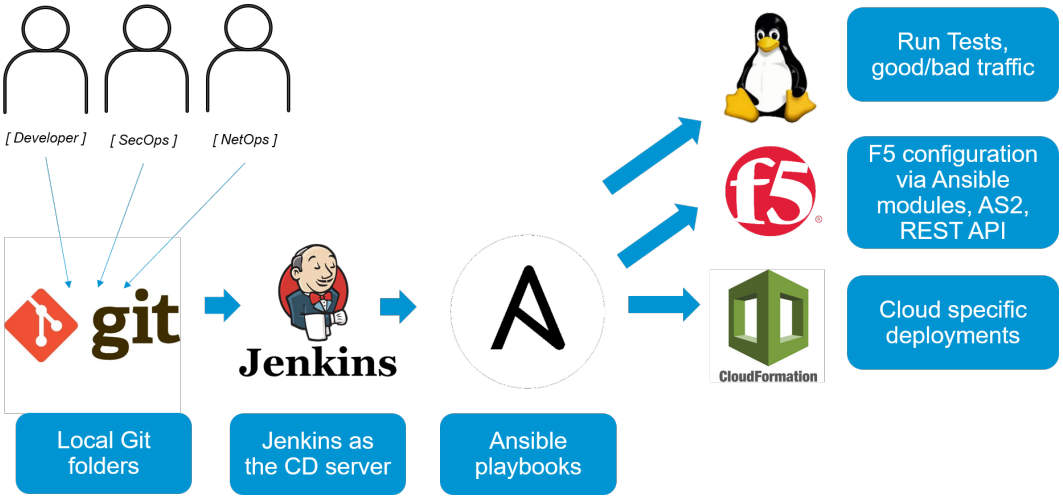
- F5-rs-container
- runs the tools we use and their dependencies (for example - f5-sdk, aws python sdk..)
- Shared infrastructure
- bigiq for licensing
- DNS
- Centralized logging platform
- automation modules



1.3 Tools

The framework leverages several automation tools, one of the automation guidelines is to use F5 supported solutions where possible,

- AWS cloud formation templates are used to deploy resources into AWS (network, app, BIGIP)
- for more information on CFT , https://aws.amazon.com/cloudformation/
- F5 supported CFT's , https://github.com/F5Networks/f5-aws-cloudformation
- Ansible modules are used to control BIGIP configuration (Profiles, waf policy upload, iApp)
- more info on F5 supported ansible modules http://clouddocs.f5.com/products/orchestration/ansible/devel/
- F5 REST API calls are used when no ansible module is available (for example, update a DOSL7 profile)
- more info on F5 iControl REST, https://devcentral.f5.com/Wiki/Default.aspx?Page=HomePage&NS=iControlREST
- Jenkins is used to create a full pipeline that ties several ansible playbooks together.
- Each Jenkins job correlates to one ansible playbook/Role
- Jenkins is also used for ops notifications (Slack)
- Git is used as the SCM
- All references in the lab itself are to the local copy of the repos that is on /home/snops/



1.4 Getting started

You can run the container from any docker host, follow the instructions here:

1.4.1 Running the container on your docker host

Note: The following instructions will create a volume on your docker host and will instruct you to store private information in the host volume. the information in the volume will persist on the host even after the container is terminated.

1. run the rs-container

```
docker pull f5usecases/f5-rs-container:1.1
docker run -it --name rs-container -v config:/home/snops/host_volume -v jenkins:/var/
↪jenkins_home -p 2222:22 -p 10000:8080 --rm f5usecases/f5-rs-container:1.1
```

The container exposes the following access methods:

- SSH to RS-CONTAINER ssh://localhsot:2222
- HTTP Access to Jenkins http://localhost:10000 (only available after you start the lab)

1.1 Connect using SSH to the RS-CONTAINER

- SSH to dockerhost:2222
- username: *root*
- password: *default*

1.2 initial setup or skip to solutions if already completed the initial setup

- Move on to configure the container:

Initial setup

1. Configure the rs-container

The entire lab is built from code hosted in this repo. To run the deployments you need to configure your personal information and credentials.

Note: You will be asked to configure sensitive parameters like AWS credentials. those are used to deploy resources on your account. those cloud resources will appear on your cloud account

it is your responsibility to use it responsibly and shut down the instances when done.

1.1 Configure credentials and personal information

The following steps are required only in the first time you run the container on a host, this information persists on the host and will be available for you on any subsequent runs.

1.1.1 Create an AWS credentials file

- create an AWS credentials file by typing:

```
mkdir -p /home/snops/host_volume/aws
vi /home/snops/host_volume/aws/credentials
```

- Copy and paste the following (and change to your keys):

```
[default]
aws_access_key_id = CHANGE_TO_ACCESS_KEY
aws_secret_access_key = CHANGE_TO_SECRET
```

1.1.2 Create a personal SSH key

The SSH key will be used when creating EC2 instances. we will store them in the host-volume so they will persist any container restart:

```
mkdir -p /home/snops/host_volume/sshkeys
ssh-keygen -f /home/snops/host_volume/sshkeys/id_rsa -t rsa -N ''
```

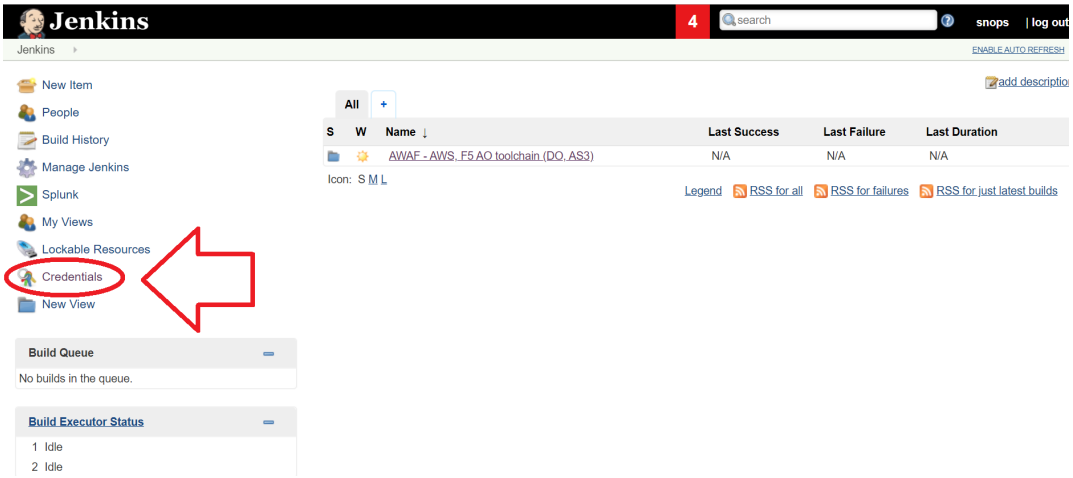
1.1.3 open jenkins

on your laptop:

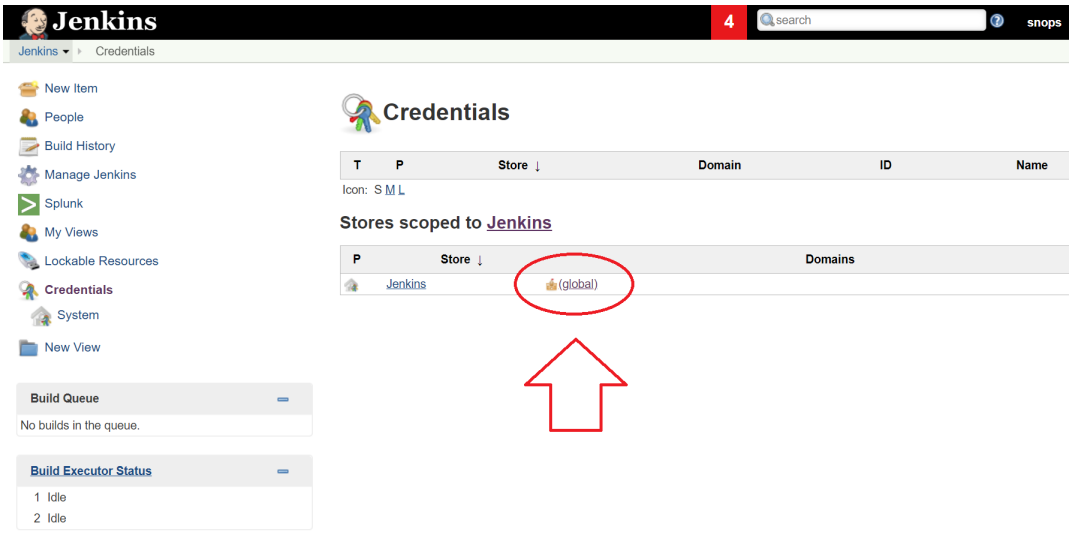
- open http://localhost:10000
- *username:* `snops` , *password:* `default`

1.1.4 add credentials

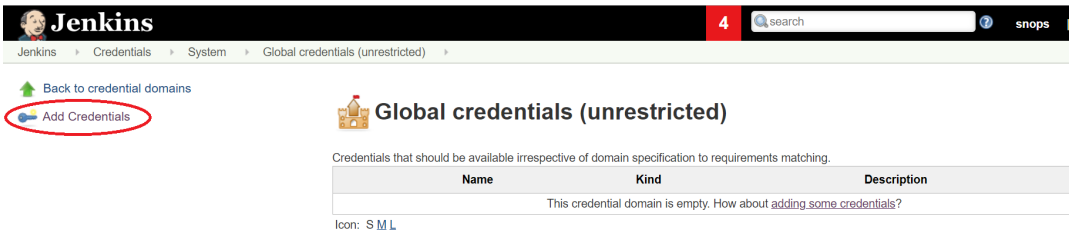
- You will now configure some paramaters as 'jenkins credentials', those paramaters are used when deploying the solutions.
- In jenkins, Navigate to 'credentilas' on the left side



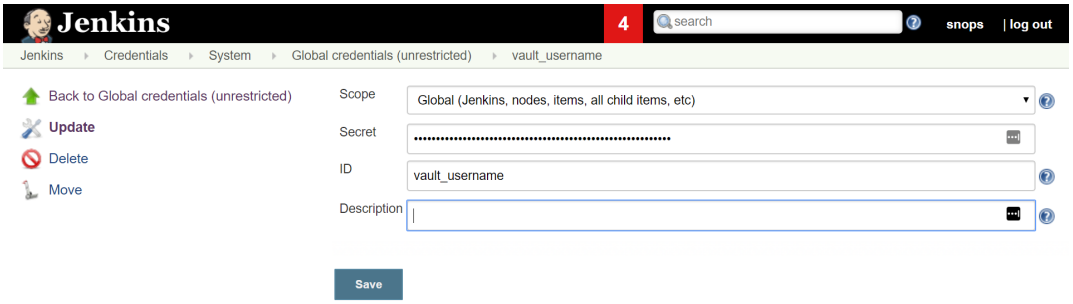
- Click on 'global'



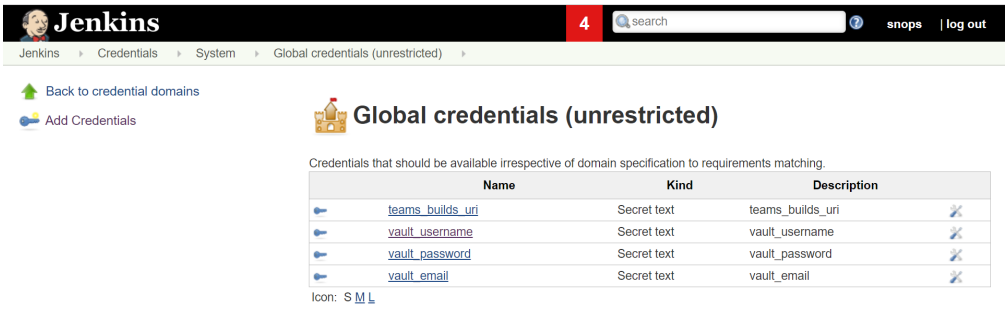
- Click on 'Add Credentials' on the left side



- Change the 'kind' to 'secret text'



- **Add the following credentials:**
  - **Secret: 'USERNAME' , ID: 'vault_username'**
    - USERNAME: used as the username for instances that you launch. also used to tag instances. example johnw. please follow BIGIP password complexity guide https://techdocs.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/big-ip-system-secure-password-policy-14-0-0/01.html
- **Add the following credentials:**
  - **Secret: 'EMAIL' , ID: 'vault_email'**
    - EMAIL: your EMAIL address
- **Add the following credentials:**
  - **Secret: 'YOUR_SECRET_PASSWORD' , ID: 'vault_password'**
    - USERNAME: used as the password for instances that you launch. needs to be a secure password.
- **Add the following credentials:**
  - **Secret: 'TEAMS_WEBHOOK' , ID: 'teams_builds_uri'**
    - TEAMS_WEBHOOK: webhook from your teams channel.
    - open teams, click on the channel options (3 points next to the channel name)
    - configure an Incoming Webhook



1.2 Run the container startup script

- Run the container startup script with the following command:
- The script will download the repos again and copy files from the host volume you just populated to the relevant directories

```
/snopsboot/start
```

2. Start a solution

List of available solutions:

F5 AWAF in AWS with DO/AS3

This lab covers the following topics:

- Deploying a vpc to AWS with the required subnets
- Deploying a juiceshop application in an autoscale group
- Deploying a bigip to AWS and onboarding it using declarative onboarding
- Deploying an F5 AWAF to protect juiceshop application
- Declaring the F5 service using AS3

Here are the lab steps:

Lab Environment

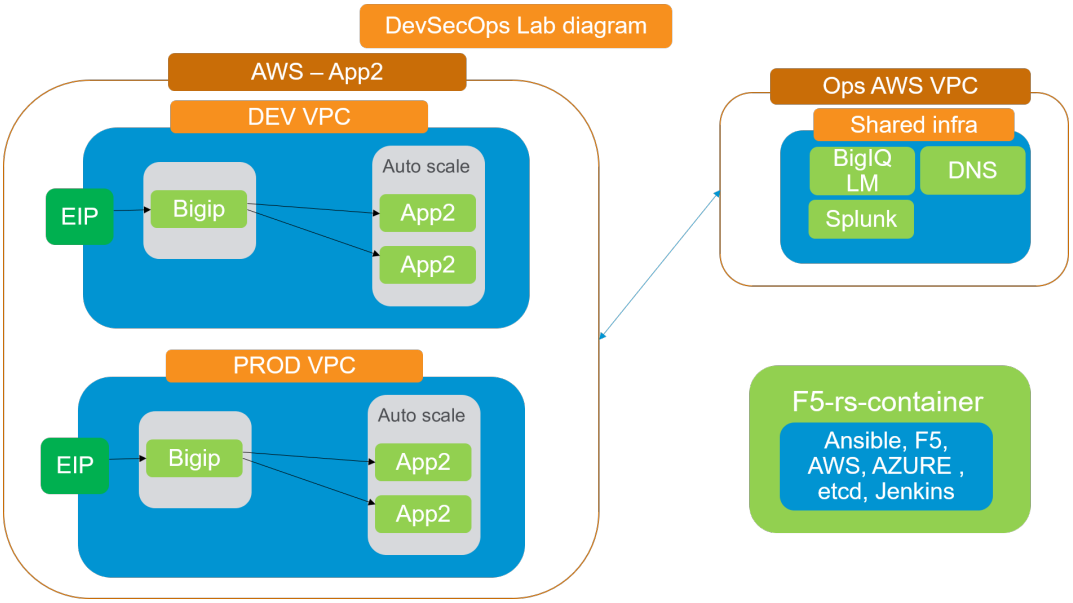
Lab Environment

this lab is intended to represent an app team that deploys their app on their own AWS VPC. while most of the components are dedicated for their app and separated from the rest of the netwrok, there are some services that the enterprise provides to this app team which are shared and are pre-built:

- Centralized logging server - Splunk server
- Bigiq License manager to license the bigips
- slack account

The application lab environment will be built in AWS, we are going to create two environments - DEV and PROD both environments have the exact same topology. in each environment we are deploying:

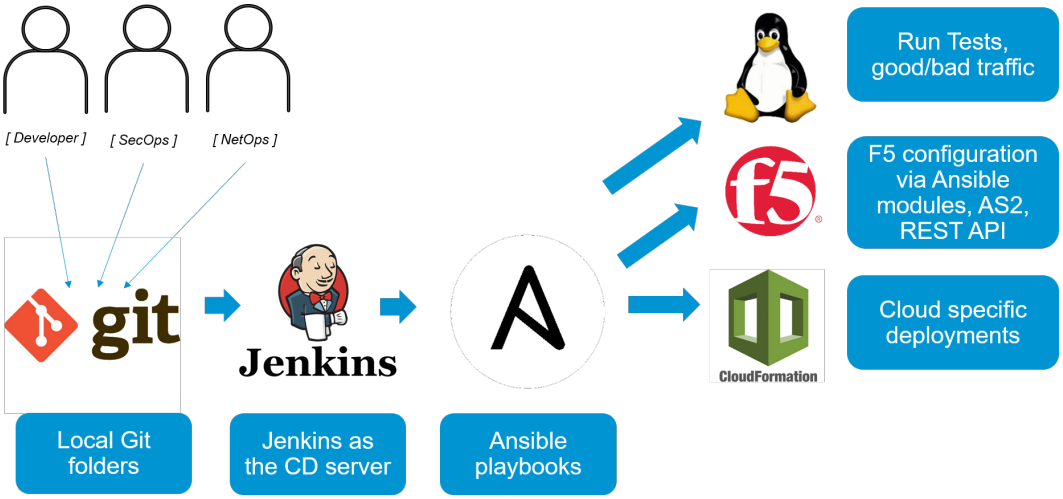
- VPC with subnets, security groups and Internet gateway.
- 1 x F5 BIG-IP VE (latest cloud version)
- An autoscale group of application servers running DOCKER with a dockerized Hackazone app running on them.



Automation workflow

This lab leverages several automation tools, one of the automation guidelines is to use F5 supported solutions where possible,

- AWS cloud formation templates are used to deploy resources into AWS (network, app, BIGIP)
- for more information on CFT , https://aws.amazon.com/cloudformation/
- F5 supported CFT's , https://github.com/F5Networks/f5-aws-cloudformation
- Ansible modules are used to control BIGIP configuration (Profiles, waf policy upload, iApp)
- more info on F5 supported ansible modules http://clouddocs.f5.com/products/orchestration/ansible/devel/
- F5 REST API calls are used when no ansible module is available (for example, update a DOSL7 profile)
- more info on F5 iControl REST, https://devcentral.f5.com/Wiki/Default.aspx?Page=HomePage&NS=iControlREST
- Jenkins is used to create a full pipeline that ties several ansible playbooks together.
- Each Jenkins job correlates to one ansible playbook/Role
- Jenkins is also used for ops notifications (Slack)
- Git is used as the SCM
- All references in the lab itself are to the local copy of the repos that is on /home/snops/



Lab 1: Deploy app and bigip

Task 1.1 - Configure jenkins credentials

1.1.1 open jenkins

on your laptop:

- open http://localhost:10000
- *username:* `snops` , *password:* `default`

1.1.2 Verify that credentials are configured

- **verify the following credentials exists:**
  - **Secret: 'USERNAME' , ID: 'vault_username'**
    - USERNAME: used as the username for instances that you launch. also used to tag instances. example johnw
- **Add the following credentials:**
  - **Secret: 'EMAIL' , ID: 'vault_email'**
    - EMAIL: your EMAIL address
- **Add the following credentials:**
  - **Secret: 'YOUR_SECRET_PASSWORD' , ID: 'vault_password'**
    - USERNAME: used as the password for instances that you launch. needs to be a secure password.
- **Add the following credentials:**
  - **Secret: 'teams_builds_uri' , ID: 'teams_builds_uri'**
    - USERNAME: uri used for teams

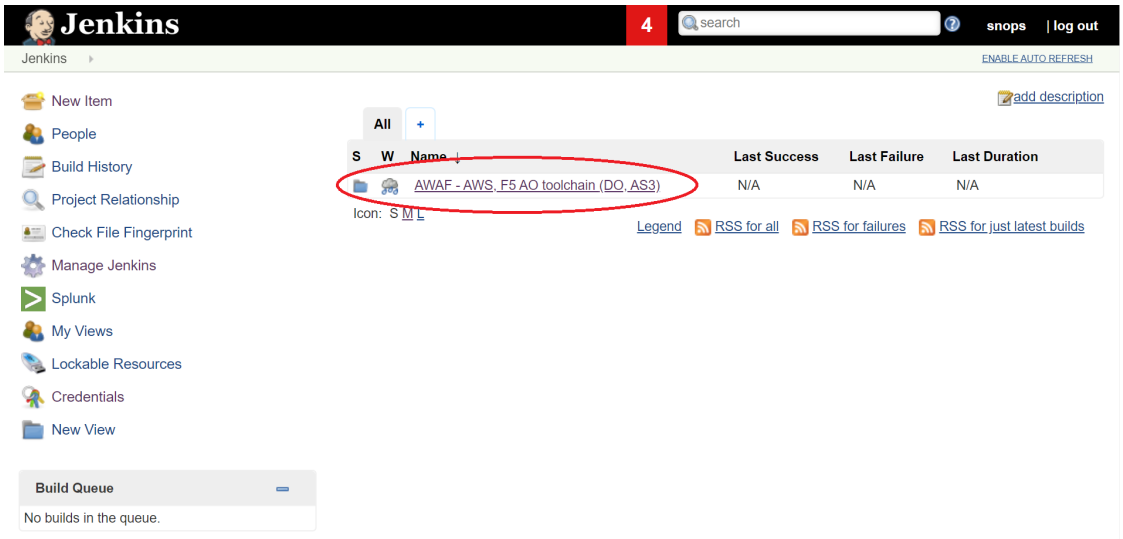
Task 1.2 - Deploy environment

1.2.1 Open Jenkins:

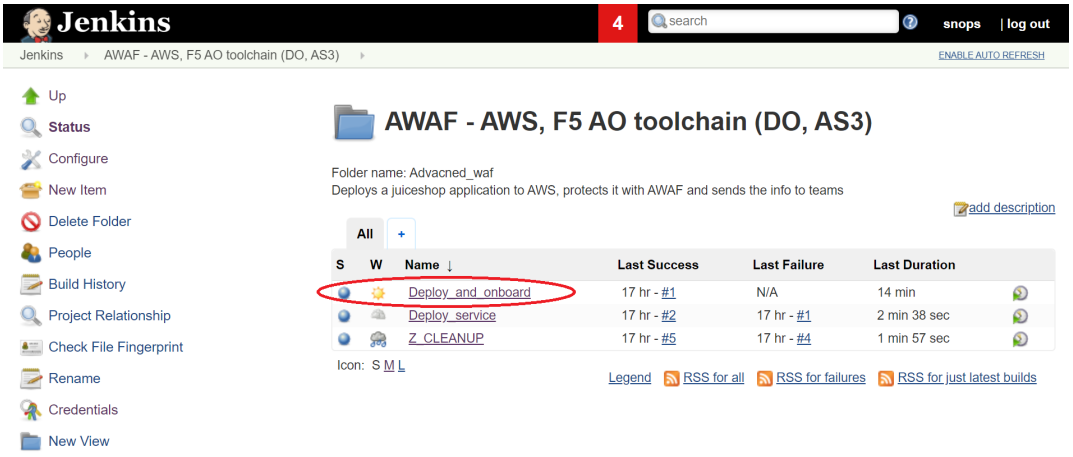
- LOCAlL: open http://localhost:10000
- *username:* `snops` , *password:* `default`

1.2.2 start the 'Deployment Pipeline':

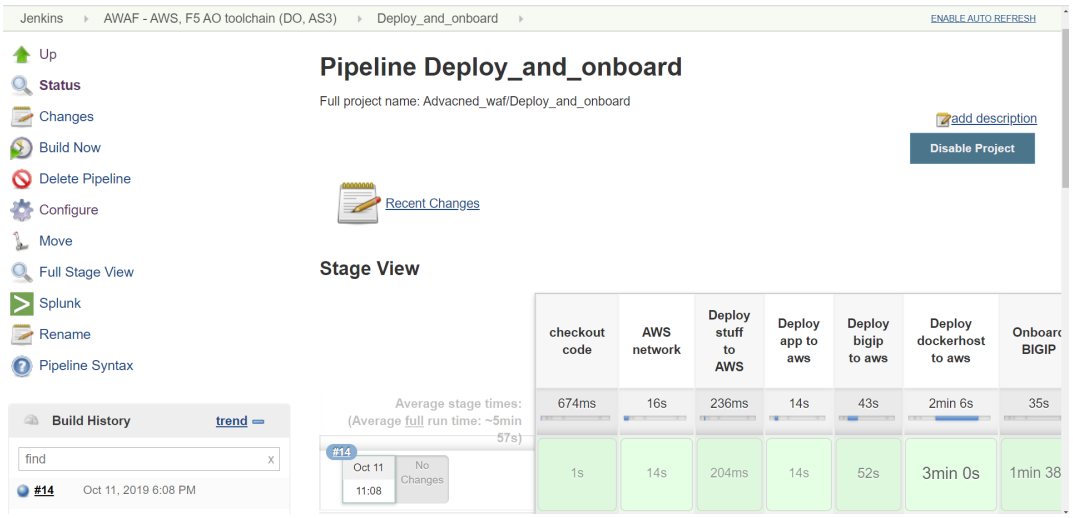
in jenkins open the *AWAF - AWS, F5 AO toolchain (DO, AS3)* folder, the lab jobs are all in this folder we will start by deploying a full environment in AWS.



- click on the 'Deploy_and_onboard' job.



- click on *Build Now* button on the left side.



Task 1.3 - Review the deployed environment

1.3.1 review jobs output:

- you can review the output of each job while its running, click on any of the green square and then click on *logs* icon

1.3.2 let the jobs run until the pipeline finishes:

- wait until all of the jobs have finished (turned green).

1.3.3 open teams channel and extract BIG-IP info:

- open the teams channel you've configured in the 'initial setup' section
- jenkins will send to this channel the BIG-IP address.
- username is the 'vault_username' that was configured in jenkins credentials
- password is the 'vault_password' that was configured in jenkins credentials

1.3.4 login to the BIG-IP:

- use the address from the slack notification (look for your username in the *builds* channel)
- username is the 'vault_username' that was configured in jenkins credentials
- password is the 'vault_password' that was configured in jenkins credentials

explore the objects that were created:

- AS3 and DO installed

Task 1.4 - Deploy services:

1.4.1 Open Jenkins:

- LOCAlL: open http://localhost:10000
- *username:* `snops` , *password:* `default`

1.4.2 start the 'service deployment Pipeline':

in jenkins open the *AWAF - AWS, F5 AO toolchain (DO, AS3)* folder, the lab jobs are all in this folder

- click on the 'Deploy_service' job.
- click on *Build Now* button on the left side.

Task 1.5 - Review the deployed application

1.5.1 review jobs output:

- you can review the output of each job while its running, click on any of the green square and then click on *logs* icon

1.5.2 let the jobs run until the pipeline finishes:

- wait until all of the jobs have finished (turned green).

1.5.3 open teams channel and extract application information info:

- open the teams channel you've configured in the 'initial setup' section
- jenkins will send the application access information to this channel

1.6 Go over WAF logs:

1.6.1 open WAF logs:

- Open the BIGIP
- Switch to App10 partition
- Go over the 'application event log', go over the 'brute force event log'

Lab 2 (BIGIP):

Task 1.2 - Explore the app repo

1.2.1 explore the infrastructure as code parameters file:

1.2.2 view git branches in the application repo:

on the container CLI type the following command to view git branches:

```
cd /home/snops/f5-rs-app10
git branch -a
```

1.2.3 explore files in the app repo:

```
more iac_parameters.yaml
```

the infrastructure of the environments is deployed using ansible playbooks that were built by devops/netops. those playbooks are being controlled by jenkins which takes the iac_parameters.yaml file and uses it as parameters for the playbooks.

- You can choose the AWS region you want to deploy in
- You can also control the WAF blocking state using this file

Task 1.3 - Update the AWS region for the DEV environment (Optional)

1.3.1 Update git with your information:

Configure your information in git, this information is used by git (in this lab we use local git so it only has local meaning) - on the RS-CONTAINER CLI

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

1.3.2 verify you edit the dev branch:

- go to the container CLI
- go to the application git folder (command below)
- check which branches are there and what is the active branch. (command below)

```
cd /home/snops/f5-rs-app10
git branch
```

1.3.3 Update the infrastructure as code parameters file:

edit the iac_parameters.yaml file to the desired AWS region. then add the file to git and commit.

- change line: aws_region: “us-west-2”
- to: aws_region: “your_region”

```
vi iac_parameters.yaml
git add iac_parameters.yaml
git commit -m "changed aws region"
```

DevSecOps - Advanced WAF in a CI/CD Workflow

This lab covers the following topics:

- Shifting WAF policies left, closer to Dev
- Declarative Advanced WAF

Lab Goals:

- Describe the main DevSecOps concepts and how they translate into an actual environment
- Describe the various roles in a DevSecOps workflow (SecOps, Dev, DevOps)
- Describe the workflow with F5 Application Security integrated into the pipeline

Roles in the Lab:

- SecOps - Represents an application security engineer
- Dave - Represents a guy from the application / end to end team, responsible for the app and infrastructure code required to build the app.
- DevOps / Automation / SRE - aren't represented in the lab. Their role is to build the tools we utilize in this lab (the automation pipeline of infrastructure and application security)

OUT OF SCOPE:

- The "how-to" and the mechanics of the automation components
- Please refer to the F5 Super-NetOps Training for the above

Expected time to complete: **1 hours**

To continue, please review the information about the Lab Environment.

Lab info

Lab Environment

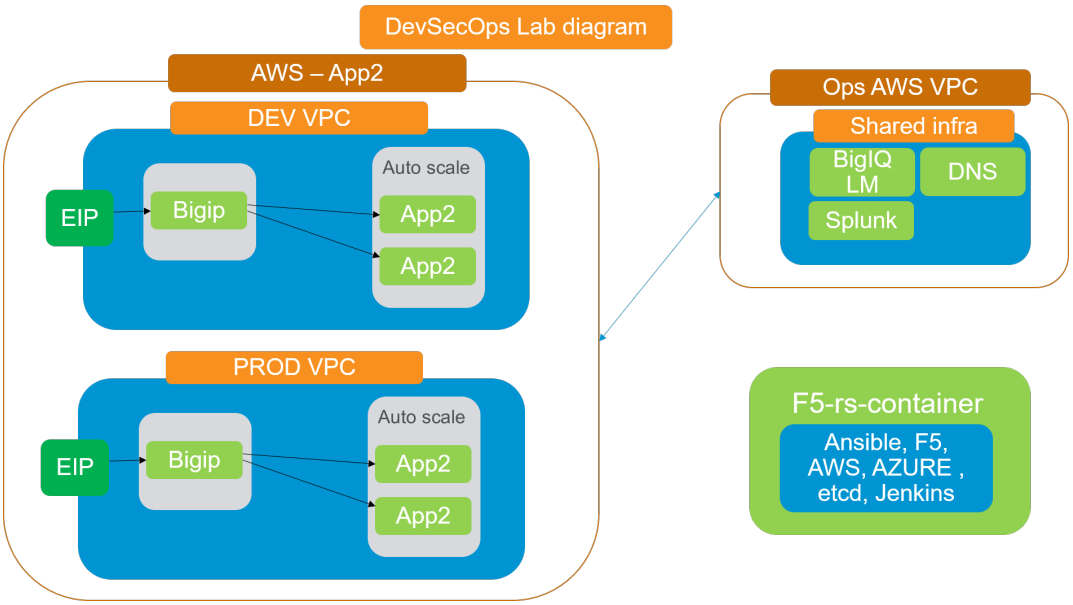
this lab is intended to represent an app team that deploys their app on their own AWS VPC. while most of the components are dedicated for their app and separated from the rest of the netwrok, there are some services that the enterprise provides to this app team which are shared and are pre-built:

- Centralized logging server - Splunk server
- Bigiq License manager to license the bigips
- slack account

The application lab environment will be built in AWS, we are going to create two environments - DEV and PROD both environments have the exact same topology. in each environment we are deploying:

- VPC with subnets, security groups and Internet gateway.
- 1 x F5 BIG-IP VE (latest cloud version)

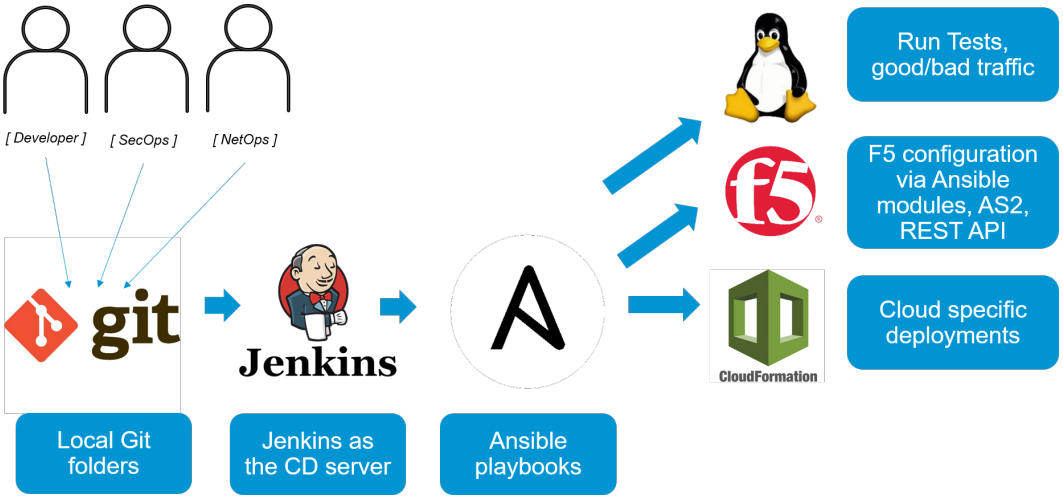
- An autoscale group of application servers running DOCKER with a dockerized Hackazone app running on them.



Automation workflow

This lab leverages several automation tools, one of the automation guidelines is to use F5 supported solutions where possible,

- AWS cloud formation templates are used to deploy resources into AWS (network, app, BIGIP)
- for more information on CFT , https://aws.amazon.com/cloudformation/
- F5 supported CFT's , https://github.com/F5Networks/f5-aws-cloudformation
- Ansible modules are used to control BIGIP configuration (Profiles, waf policy upload, iApp)
- more info on F5 supported ansible modules http://clouddocs.f5.com/products/orchestration/ansible/devel/
- F5 REST API calls are used when no ansible module is available (for example, update a DOSL7 profile)
- more info on F5 iControl REST, https://devcentral.f5.com/Wiki/Default.aspx?Page=HomePage&NS=iControlREST
- Jenkins is used to create a full pipeline that ties several ansible playbooks together.
- Each Jenkins job correlates to one ansible playbook/Role
- Jenkins is also used for ops notifications (Slack)
- Git is used as the SCM
- All references in the lab itself are to the local copy of the repos that is on /home/snops/



Accessing the lab

The lab is built from code, to run it you need a docker host (can be your laptop), and an AWS account with API access (access and secret keys):

Module 0 - initial setup

Note: This environment is currently available for F5 employees only

---

Determine how to start your deployment:

- **Official Events (ISC, SSE Summits):** Please follow the instructions given by your instructor to join the UDF Course.
- **Self-Paced/On Your Own:** Login to UDF, *Deploy* the *Security Lab: DevSecOps* Blueprint and *Start* it.

1. Connecting to the Environment

To connect to the lab environment we will use SSH to the jumphost.

SSH key has to be configured in UDF in order to access the jumphost.

The lab environment provides several access methods to the Jumphost:

- SSH to RS-CONTAINER
- SSH to the linux host
- HTTP Access to Jenkins (only available after you start the lab)

1.1 Connect using SSH to the RS-CONTAINER

1. In UDF navigate to the *Deployments*
2. Click the *Details* button for your DevSecOps Deployment

3. Click the *Components* tab
4. Find the `Linux Jumphost` Component and click the the *ACCESS* button.
5. use your favorite SSH client to connect to *RS-CONTAINER* using your UDF private key. username is *root*

1.2 Configure the rs-container

The entire lab is built from code hosted in this repo, the container that you are connecting to runs on the linux host and is publicly available. to run the deployments you need to configure it with personal information and credentials.

1.3 Configure credentials and personal information

1.3.1 Copy ssh key, aws credentials and global parameters file

the SSH key will be used when creating EC2 instances. we will store them in the Jenkins SSH folder so that Jenkins can use them to access instances.

Copy credentials and parameters files from the host folder using the following script:

```
/home/snops/host_volume/udf_startup.sh
```

1.3.2 Edit the global parameters file with your personal information

- Edit the encrypted global parameters file `/home/snops/f5-rs-global-vars-vault.yaml` by typing:

```
ansible-vault edit --vault-password-file /var/jenkins_home/.vault_pass.txt /home/
↪snops/f5-rs-global-vars-vault.yaml
```

- Once in edit mode - type `i` to activate INSERT mode and configure your personal information by changing the following variables: `vault_dac_user`, `vault_dac_email` and `vault_dac_password`
- Use your student# from Teams for `vault_dac_user` - used as a Tenant ID to differentiate between multiple deployments
- Choose your own (secure) value for `vault_dac_password` - ** this is the password for the `admin` user of the BIG-IP **
- There are a number of special characters that you should avoid using in passwords for F5 products. See https://support.f5.com/csp/article/K2873 for details

For example:

```
vault_dac_user: "student01" // username IS case sensitive
vault_dac_email: "yossi@f5.com"
vault_dac_password: "Sup3rsecur3Passw0rd1"
```

- Press the `ESC` key and save the file by typing: `:wq`

1.3.3 Configure Jenkins and reload it

Run the following command to configure jenkins with your personal information and reload it:

```
ansible-playbook --vault-password-file /var/jenkins_home/.vault_pass.txt /home/snops/
↪f5-rs-jenkins/playbooks/jenkins_config.yaml
```

- Start: *Module 1: Shifting WAF policy left, closer to DEV.*

Module 1: Shifting WAF policy left, closer to DEV.

In this module you will review the lab environment, practice some of the concepts discussed in class:

- break down the silos, enable dev to deploy securely with minimum friction.
- introduce security as early on in the dev chain as possible
- automated security tests
- roles of secops and dev in our lab model and deploy an app to prod with WAF protection.

Lab 1 (Dave): Deploy app to DEV environment

Background:

Security team has created some security policies templates, those were built based on the F5 templates with some modifications to the specific enterprise. in this lab we don't cover the 'how to' of the security templates. we focus on the operational side and the workflows.

The Tasks are split between the two roles:

- SecOps
- Dave - a person from the 'end to end' team. a team that's responsible for the application code and running it in production.

Lab scenario:

New app - App2 is being developed. the app is an e-commerce site. code is ready to go into 'DEV' environment. for lab simplicity there are only two environments - DEV and PROD. Dave should deploy their new code into a DEV environment that is exactly the same as the production environment. run their application tests and security tests.

Note: Pipeline is broken to DEV and PROD for lab simplicity. from a workflow perspective the pipelines are the same. it is broken up to two for a better lab flow.

Note: OUT OF SCOPE - a major part of the app build process is out of scope for this lab, Building the app code and publish it as a container to the registry. this process is done using DOCKERHUB.

Task 1.1 - review Dave's repo

- Make sure you've completed the setup section - http://f5-rs-docs.readthedocs.io/en/latest/solutions/devsecops/labinfo/udf.html

1.1.1 view git branches in the application repo:

on the container CLI type the following command to view git branches:

```
cd /home/snops/f5-rs-app2
git branch
```

the app repo has two branches, dev and master. we are now working on the dev branch.

Note: the lab builds two environments, dev and prod. the dev environment deploys the code on the dev branch the prod environment deploys the code on the master branch.

1.1.2 view files in the application repo:

on the container CLI type the following commands to view the files in the repo:

```
cd /home/snops/f5-rs-app2
ls
```

- application code under the 'all-in-one-hackazon' folder.
- infrastructure code maintained in the 'iac_parameters.yaml' file.

1.1.3 explore the infrastructure as code parameters file:

```
more iac_parameters.yaml
```

the infrastructure of the environments is deployed using ansible playbooks that were built by devops/netops. those playbooks are being controlled by jenkins which takes the iac_parameters.yaml file and uses it as parameters for the playbooks.

- that enables Dave to choose the AWS region in which to deploy, the name of the app and more.
- Dave can also control the deployment of the security policies from his repo as we will see.

Task 1.2 - Deploy dev environment

Note: Jenkins can be configured to run the dev pipeline based on code change in dave's app repo. in this lab we are manually starting the Full stack pipeline in Jenkins to visualize the process.

1.2.1 Open Jenkins:

go to UDF, on the *jumphost* click on *access* and *jenkins*

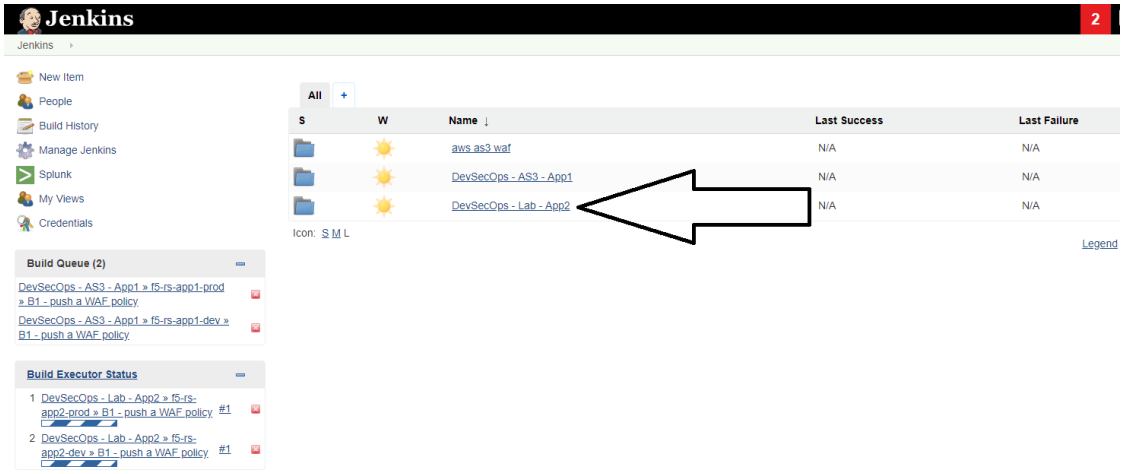
username: `snops` , *password:* `default`

Note: when you open jenkins you should see some jobs that have started running automatically, jobs that contain: 'Push a WAF policy', this happens because jenkins monitors the repo and start the jobs.

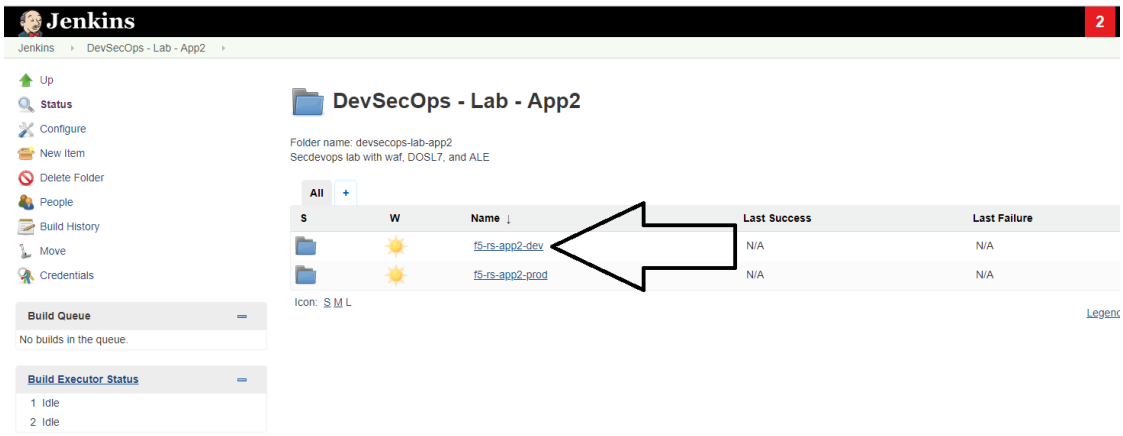
you can cancel the jobs or let them fail.

1.2.2 start the 'Full stack pipeline':

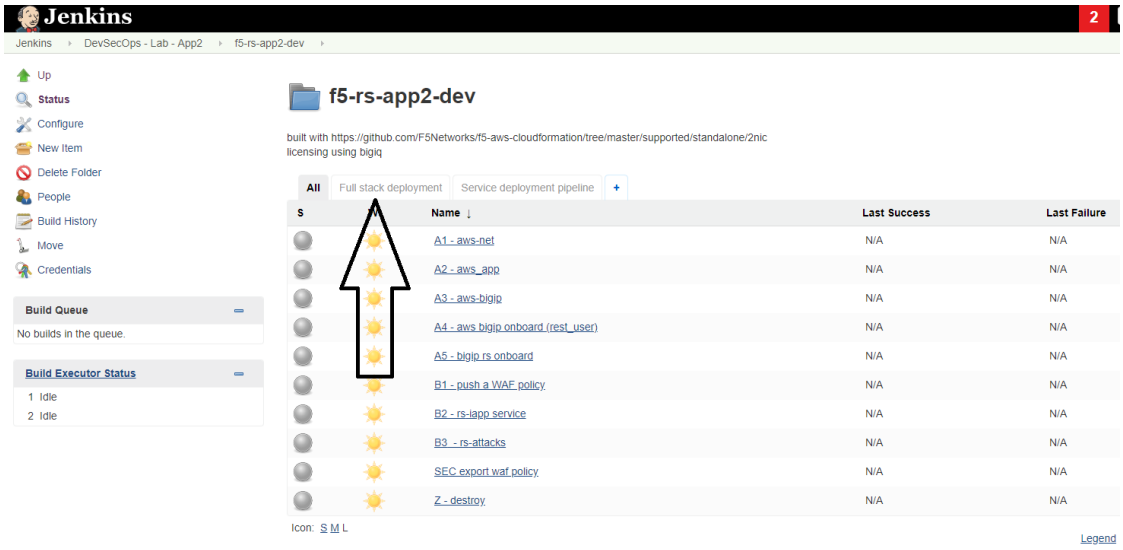
in jenkins open the *DevSecOps - Lab - App2* folder, the lab jobs are all in this folder we will start by deploying a DEV environment, you will start a pipeline that creates a full environment in AWS.



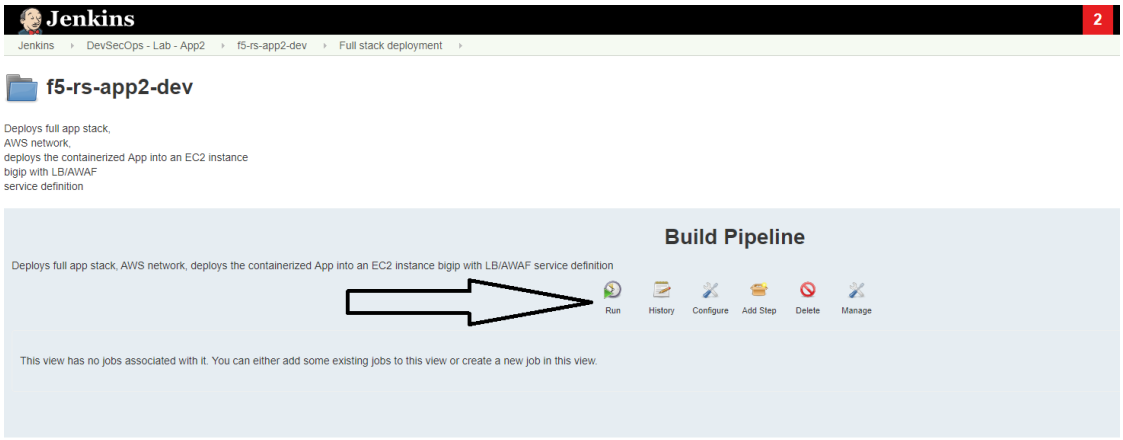
click on the 'f5-rs-app2-dev' folder. here you can see all of the relevant jenkins jobs for the dev environment.



click on 'Full stack deployment' , that's the pipeline view for the same folder.



click on 'run' to start the dev environment pipeline.

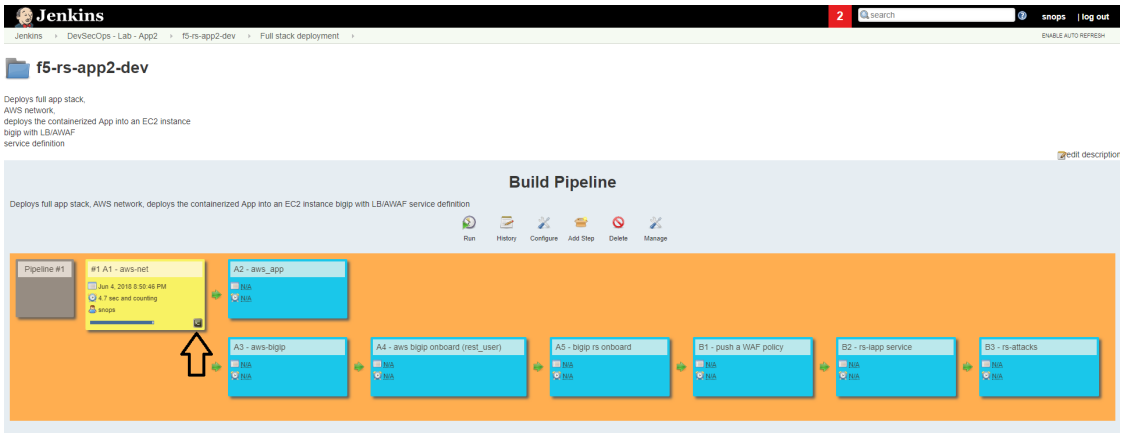


Task 1.3 - Review the deployed environment

Note: Jenkins doesn't automatically refresh the page, either refresh manually to see the progress or click on the 'ENABLE AUTO REFRESH' on the upper right side.

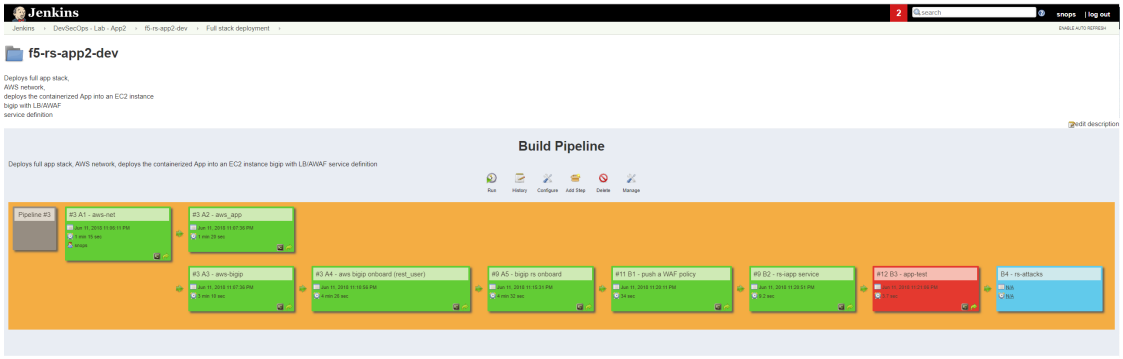
1.3.1 review jobs output:

you can review the output of each job while its running, click on the small *console output* icon as shown in the screenshot:



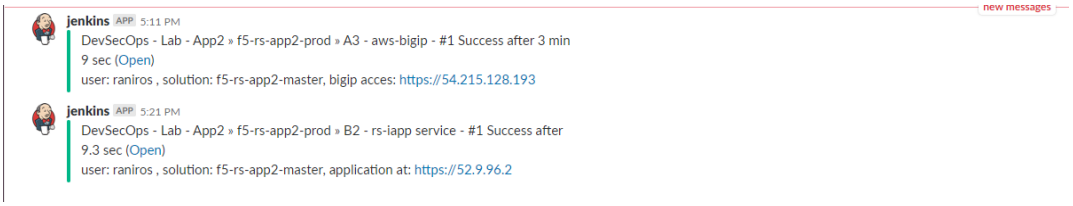
1.3.2 let the jobs run until the pipeline finishes:

wait until all of the jobs have finished (turned green and the app-test one is red).



1.3.3 open slack and extract BIG-IP and application info:

- open slack - https://f5-rs.slack.com/messages/C9WLUB89F/ (if you don't already have an account you can set it up with an F5 email)
- go to the *builds* channel.
- use the search box on the upper right corner and filter by your username (student#). replace you student# in this string: "user: student# , solution: f5-rs-app2-dev, bigip acces:"
- jenkins will send to this channel the BIG-IP and the application address.



1.3.4 login to the BIG-IP:

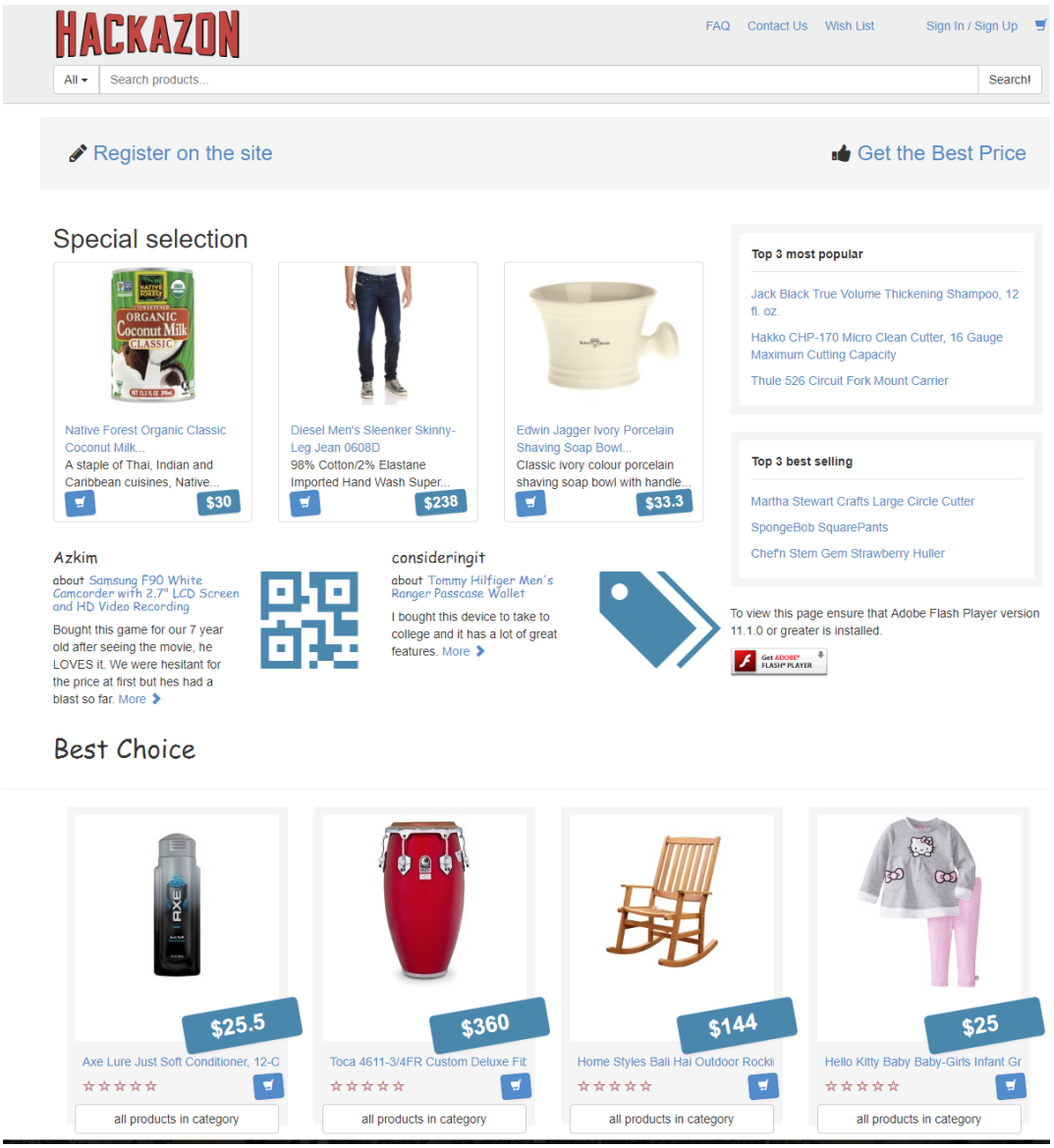
- use the address from task 1.3.3

- username: *admin*
- password: the personal password you defined in the global parameters file in the vault_dac_password parameter.

explore the objects that were created:

1.3.5 Access the App:

- open slack - https://f5-rs.slack.com/messages/C9WLUB89F/ (if you don't already have an account you can set it up with an F5 email)
- go to the *builds* channel.
- use the search box on the upper right corner and filter by your username (student#). replace you student# in this string: "user: student# , solution: f5-rs-app2-dev, application at:"
- try to access the app using the ip provided in the slack channel - that's the Elastic ip address that's tied to the VIP on the BIG-IP.
- after ignoring the ssl error (because the certificate isn't valid for the domain) you should get to the Hackazone mainpage



1.3.6 Summary - Jobs roles:

A1 - aws-net:

- Builds an AWS VPC with subnets and security groups.
- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo. (like which region)
- Ansible playbook takes the parameters and use them to deploy a cloud formation template
- cloud formation template deploys all resources in AWS subscription

A2 - aws_app:

- Deploys an AWS autoscale group with a containerized app

- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo. (like container name)
- Jenkins uses the VPC / subnets information from previews job
- Ansible playbook takes the parameters and use them to deploy a cloud formation template
- cloud formation template deploys all resources in AWS subscription

A3 - aws-bigip:

- Deploys a BIG-IP to AWS
- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo. (like which region)
- Jenkins uses the VPC / subnets information from previews job
- Ansible playbook takes the parameters and use them to deploy a cloud formation template
- cloud formation template deploys all resources in AWS subscription

A4 - aws bigip onboard (rest_user):

- Connects to the BIG-IP over SSH with private key (only way to connect to an AWS instance).
- configures rest user and password for future use

A5 - bigip rs onboard:

- deploys the 'enterprise' default profiles, for example: HTTP, analytics, AVR, DOSL7, iapps etc.
- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo.
- Ansible playbook takes the parameters and uses them to deploy a configuration to the BIG-IP using the F5 supported ansible modules and API's.

B1 - push a WAF policy:

- deploys the 'application specific' profiles, for example: DOSL7, waf policy
- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo. (which waf policy to use, dosl7 parameters)
- Ansible playbook takes the parameters and uses them to deploy a configuration to the BIG-IP using the F5 supported ansible modules and API's.

B2 - rs-iapp service:

- deploys the 'service definition' uses AS2 API
- Jenkins runs a shell command that kicks off an ansible playbook with parameters from the application repo.
- Jenkins uses the application autoscale group name from previous jobs
- Ansible playbook takes the parameters and uses them to deploy a configuration to the BIG-IP using the F5 supported ansible modules and API's.

- AS2 turns the service definition into objects on the BIG-IP

B3 - app-test:

- Send HTTP requests to the application to test it
- Jenkins runs a shell command that kicks off an ansible playbook with parameters
- Ansible playbook takes the parameters and uses them to run HTTP requests to our APP.

B4 - rs-attacks:

- Test app vulnerabilities
- Jenkins runs a shell command that kicks off an ansible playbook with parameters
- Ansible playbook takes the parameters and uses them to run HTTP requests to our APP.

SEC export waf policy:

- Pulls a policy from a BIG-IP and stores in a git repo
- Jenkins runs a shell command that kicks off an ansible playbook with parameters
- Ansible playbook takes the parameters and uses them to run F5 modules (Created by Fouad Chmainy <F.Chmainy@F5.com>) to pull the waf policy from the BIG-IP

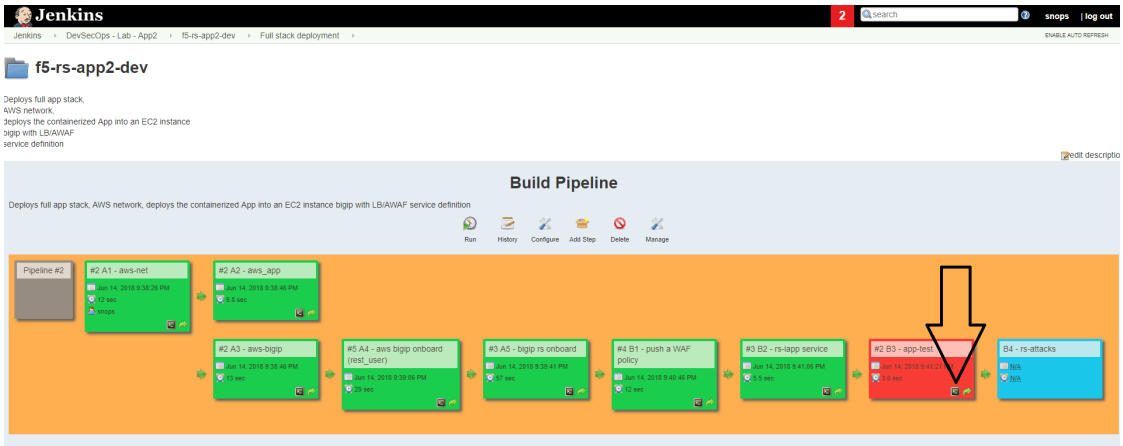
Z - destroy:

- Destroy the environment

Task 1.4 - Go over the test results

1.4.1 view the test results:

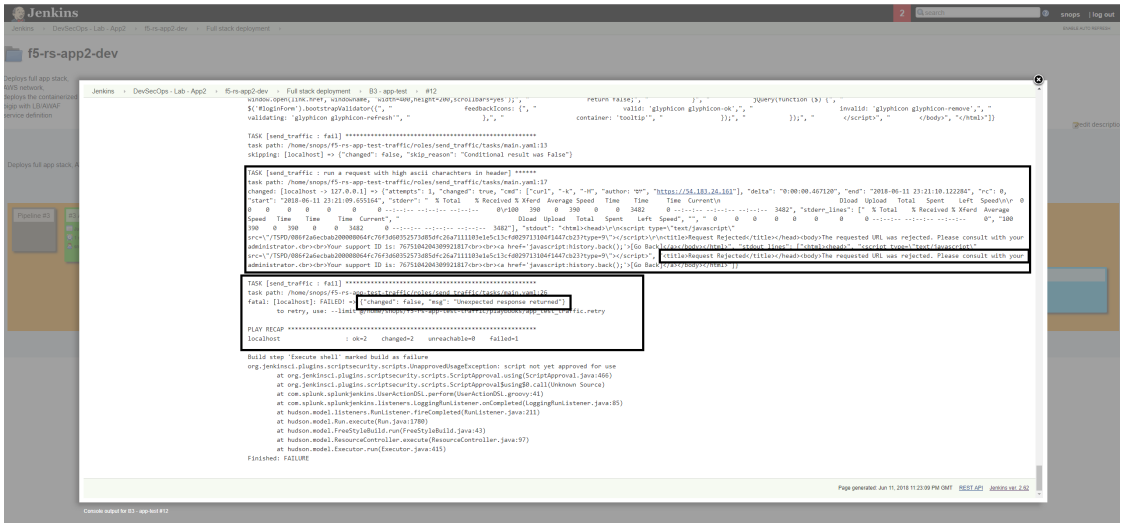
the deployment process failed because not all of the application tests completed successfully. review the app-test job *console output*



1.4.2 identify the WAF blocked page response:

scroll to the bottom of the page, you should see the response with *request rejected*, and the failure reason as *unexpected response returned*

this is an indication that ASM has blocked the request. in our case it is a false positive.



Note: in this lab secops uses the same WAF policy template for many apps. we don't want to create a 'snowflake' waf policy. so with this failure dave will escalate to secops. that ensures that the setting will be reviewed and if needed the policy template will get updated. we don't want to create a 'snowflake' waf policy. so with this failure Dave will escalate to secops. this ensures that the setting will be reviewed and if needed the policy template will get updated.

Lab 2 (Secops): Tune/fix security policy

Background:

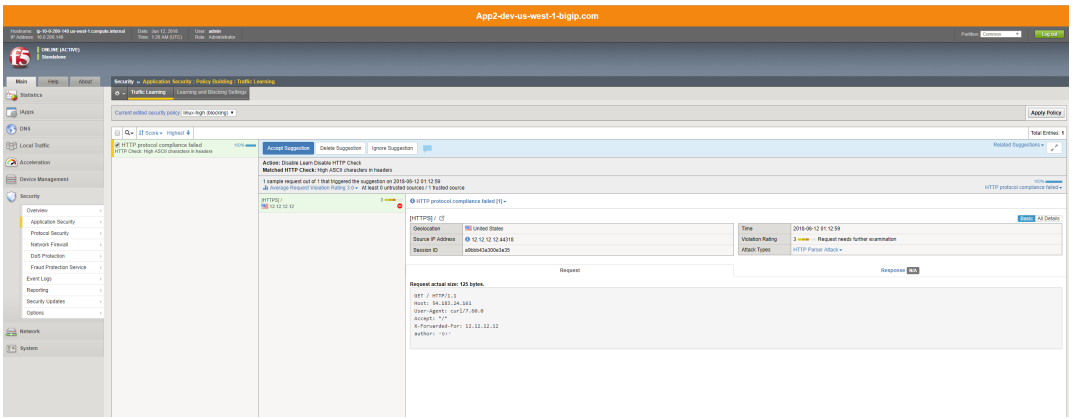
The application team tests came back and some of the tests have failed. the test result came back with the WAF blocking page.

Task 2.1 - Find which requests were blocked and resolve false-positive

2.1.1 Clear false positive:

- log on to the 'DEV' bigip. (username: admin , password: your personal password that you set in the lab setup) see section 1.3.3
- log on to the 'DEV' BIG-IP.
- go to 'traffic learning',
- make sure you are editing the 'linux-high' policy.
- check the requests that triggered suggestions.
- you should see a suggestion on 'High ASCII characters in headers' , examine the request. this is a false positive.

- the app uses a different language in the header and it is legitimate traffic.
- you can also see that the request comes from a trusted ip.
- accept the suggestion.



2.1.2 Apply the policy :

- apply the policy.

Note: you are applying the policy to DEV, secops shouldn't change the waf policy running in production outside of the ci/cd workflow ** unless there is a true emergency

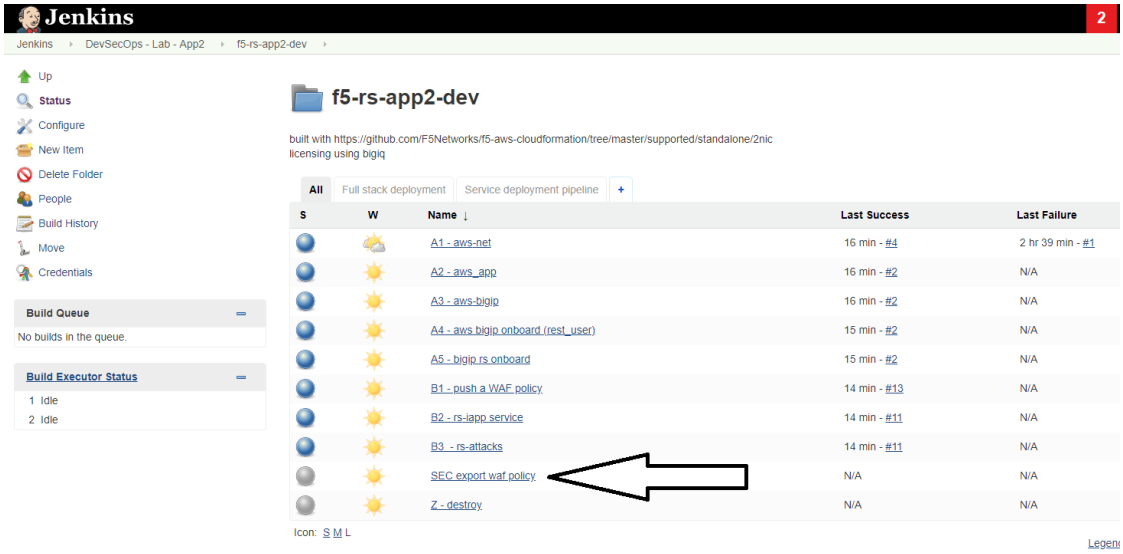
---

Task 2.2 - Save the WAF policy to the templates repo (managed by secops)

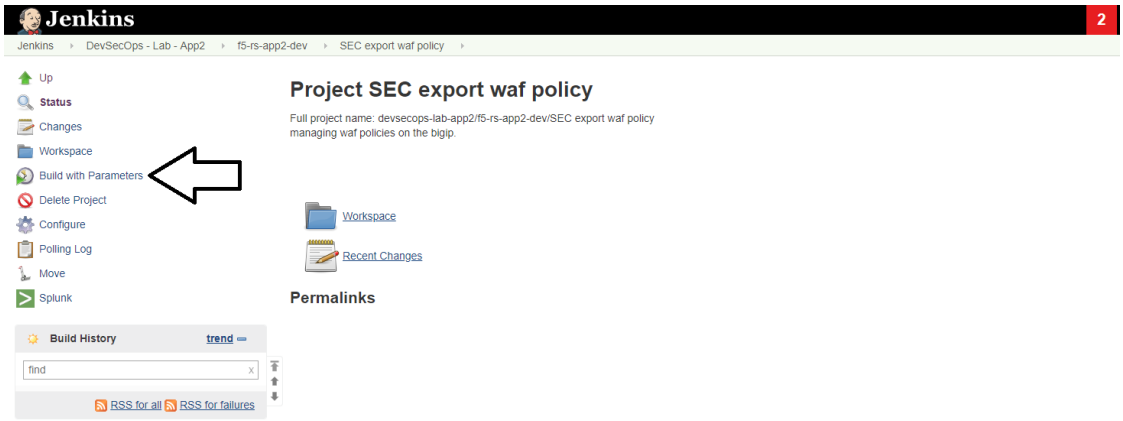
- secops have updated the policy with a setting that makes sense to update on the general template.
- we will now export the policy from the BIG-IP to the waf-policies repo (managed by secops)

2.2.1 Pull WAF policy from the BIG-IP :

go back to jenkins, under the 'f5-rs-app2-dev' there is a job that will export the policy and save it to the git repo - *SEC export waf policy*



click on this job and choose *Build with Parameters* from the left menu.



you can leave the defaults, it asks for two parameters. the first parameter is the name of the policy on the BIG-IP and the other is the new policy name in the git repo.

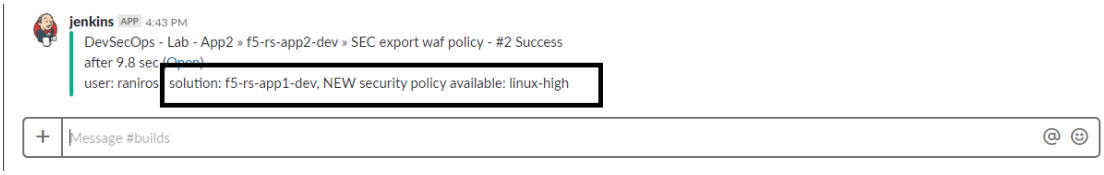
Note: why saving the template with a different version ? changes should be tracked, more than that we should allow app teams to 'control their own destiny' allowing them to choose the right time and place to update the waf policy in their environment. by versioning the policies we ensure their control over which template gets deployed.

---

click on 'build'

2.2.2 Check slack channel notification :

check the slack channel - you should see a message about the new security policy that's ready. this illustrates how chatops can help communicate between different teams.



the security admin role ends here. it's now up to Dave to run the pipeline again.

Lab 3: (Dave) Deploy with a new WAF policy

Background:

secops found a false positive on the waf policy template, they fixed it and created a new version for that policy.

Task 3.1 - Update the WAF policy deployed in DEV

we (Dave) got the message on a new waf template, we need to deploy the new template to the DEV environment. to do so we will edit the 'infrastructure as code' parameters file in Dave's app2 repo.

3.1.1 Update git with your information:

Configure your information in git, this information is used by git (in this lab we use local git so it only has local meaning) - on the RS-CONTAINER CLI

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

3.1.2 verify you edit the dev branch:

- go to the container CLI
- go to the application git folder (command below)
- check which branches are there and what is the active branch. (command below)
- you should be on the 'dev' branch. the files you see belong to the dev branch.

```
cd /home/snops/f5-rs-app2
git branch
```

3.1.3 Update the infrastructure as code parameters file:

edit the iac_parameters.yaml file to point the deployment to the new WAF policy (linux-high-v01). then add the file to git and commit.

- change line: waf_policy_name: "linux-high"
- to: waf_policy_name: "linux-high-v01"

```
vi iac_parameters.yaml
git add iac_parameters.yaml
git commit -m "changed asm policy"
```

```
[jenkins@f5-super-netops] [/home/snops] $ pwd
/home/snops
[jenkins@f5-super-netops] [/home/snops] $ cd f5-rs-app2
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ git branch
* dev
  master
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ vi iac_parameters.yaml
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ more iac_parameters.yaml
service_name: "App2"
aws_region: "us-west-1"
rs_container_name: "mutzel/all-in-one-hackazon:postinstall supervisord -n"
waf_policy_name: "linux-high"
# enables or disables proactive bot mitigation - Sends JS challange
# allowed values: disabled / always
proactive_autometed_attack_prevention: "disabled"
#enables or disables login page paramaters encryption
#login_page_file and login_page_path are made of the login URL, for example /user/login will have login_page_file: "login", login_page_path: "user"
login_password_encryption: "disabled"
login_page_file: "login"
login_page_path: "user"

[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ vi iac_parameters.yaml
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ git add iac_parameters.yaml
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $ git commit -m "enabled proactive bot protection"
[dev 264a72b] enabled proactive bot protection
 1 file changed, 1 insertion(+), 1 deletion(-)
[jenkins@f5-super-netops] [/home/snops/f5-rs-app2] $
```

3.1.4 service deployment update:

Note:

- we now have an active DEV environment, the app, network and BIG-IP shouldn't change. the only change is to the SERVICE deployed on the BIG-IP.
- we have a dedicated pipeline view for the Service deployment.
- **jenkins is set up to monitor the application repo. when a 'commit' is identified jenkins will start an automatic pipeline to c**
  - that way it will update the WAF policy on the BIG-IP.

- go back to jenkins and open the *f5-rs-app2-dev* folder. choose the *Service deployment pipeline* tab , it takes up to a minute for jenkins to start the pipeline. **you should see that the tasks start to run and the pipeline finishes successfully (all tasks are now green).**
- Don't forget to refresh the page

3.1.5 view changes on the BIG-IP :

- log on to the BIG-IP again, check which WAF policies are there and which policy is attached to the 'App2 VIP' check the 'traffic learning' for the security policy and verify you no longer see the 'high ascii charachters'

this concludes the tests in the 'dev' environment. we are now ready to push the changes to production.

Lab 4 (Dave): Deploy the PROD environment

Background:

we completed tests in DEV, both functional tests and security tests have passed.

Task 4.1 - merge infrastructure as code file from dev

- we will 'merge' the app2 dev branch with the master branch. so that the production deployment will use the correct policy.
- on the /home/snops/f5-rs-app2 folder:

```
git checkout master
git merge dev -m "changed asm policy"
```

Note: the merge will trigger a job in Jenkins that's configured to monitor this repo - 'Push waf policy', since the environment isn't deployed yet it will fail, either cancel the job or let it fail.

Task 4.2 deploy PROD:

Note: in this lab we manually deploy PROD after the tests have completed. this manual step can easily be automated. what are the metrics that we need to verify successful deployment ?

How can splunk analytics / BIG-IQ 6.0 help with that ?

- go to the 'f5-rs-app2-prod' folder, choose the 'Full stack deployment' view and run the pipeline.
- open slack - https://f5-rs.slack.com/messages/C9WLUB89F/
- go to the *builds* channel.
- use the search box on the upper right corner and filter by your username (student#). replace you student# in this string: "user: student# , solution: f5-rs-app2-master, bigip acces:"
- open the BIG-IP and verify that you don't see the 'high ascii' false positive.
- verify the security policy that's attached to the VIP.

Module 2: Declarative advanced waf

In this module you will use declarative security controls that controls the F5 advanced waf. the Lab doesn't cover how to configure the automation tools, just how to operate them and the workflow. in this lab we cover:

- automated attacks prevention
- application layer encryption (OPTIONAL)

Lab 1: (Dave) protect the app from automated attacks

Background:

after the app was launched we started identifying an abnormal activity, some specific products were added to the cart until the stock was out but were never purchased. in addition we identified an abuse of our coupons that every new member gets.

in an effort to mitigate those unwanted requests the secops engineer suggests the use of 'proactive bot defense', he configures a template DOSL7 profile with some values as defaults.

he then exposes the option of enabling / disabling proactive bot defense from the 'iac_paramaters' file.

it is up to Dave now to deploy the new feature in dev and promote to PROD when it makes sense for him.

Task 1.1 - Enable proactive bot defense in the DEV environment

1.1.1 Review iac_paramaters file in the app repo:

- Open the container CLI
- go to the application git folder.
- check which branches are there and what is the active branch. (git branch)
- you should be on the 'dev' branch. the files you see belong to the dev branch.

```
cd /home/snops/f5-rs-app2
git checkout dev
git branch
```

1.1.2 Edit the iac_paramaters file in the app repo:

- edit the iac_parameters.yaml file to enable proactive bot defense,
- change the setting from:
- proactive_autometed_attack_prevention: "disabled"
- To
- Proactive_autometed_attack_prevention: "always"
- change the setting from: `proactive_autometed_attack_prevention: "disabled"` to `proactive_autometed_attack_prevention: "always"`

```
vi iac_parameters.yaml
```

1.1.3 Add the file to git and commit:

- add the file to git and commit

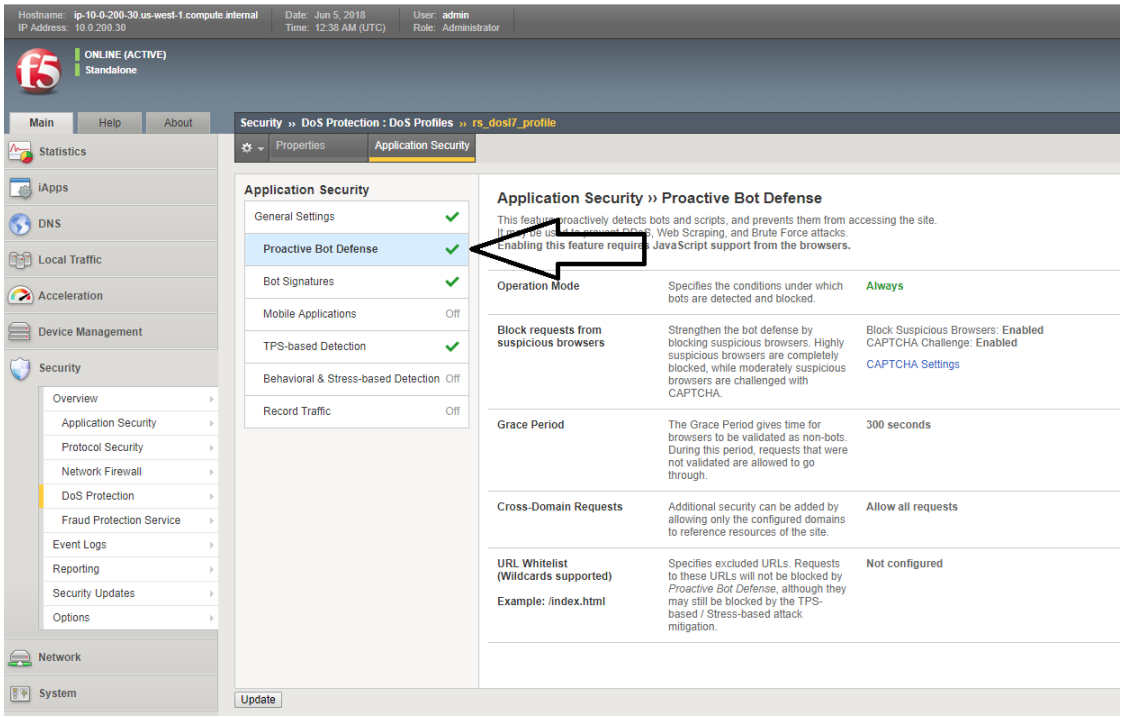
```
git add iac_parameters.yaml
git commit -m "enabled proactive bot defense"
```

1.1.4 View the automatic pipeline:

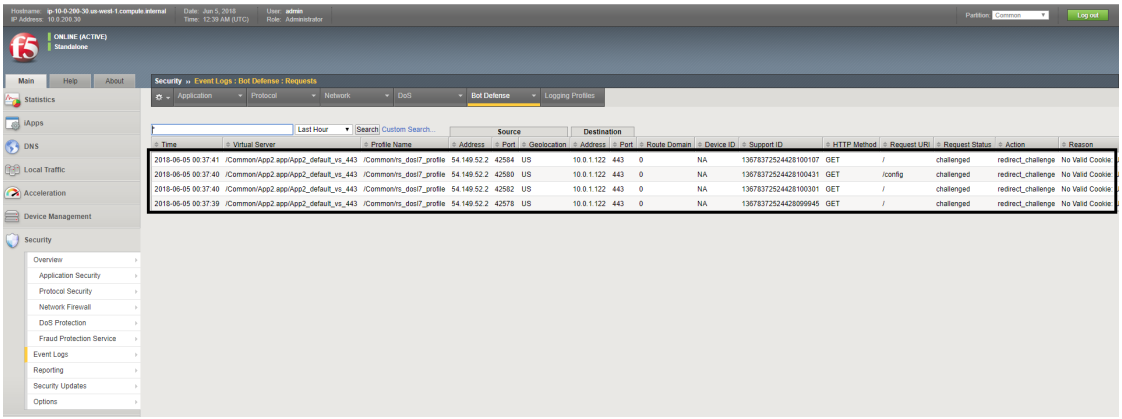
- go back to jenkins and open the *f5-rs-app2-dev* folder. choose the *Service deployment pipeline* tab ,
- go back to jenkins and open the *f5-rs-app2-dev* folder. choose the *Service deployment pipeline* tab, jenkins is set up to monitor the application repo. when a 'commit' is identified jenkins will start an automatic pipeline to deploy the service. it takes up to a minute for jenkins to start the pipeline. jenkins takes the parametes from the git repo and uses them to deploy/update the service.
- OPTIONAL - Log on to splunk (logon details in the UDF documentation), navigate to your app and look under the 'Security - DDoS' tab for proactive mitigation.

Task 1.2 - (Secops) Verify bot defense configuration and logs on the BIG-IP

while all of the logs are sent to splunk where they can be viewed by Dave, part of the lab is to verify the change on the BIG-IP. this task doesn't represents an actual step of the deployment. it is just for lab purpose log on to the dev BIG-IP again, check the setting on the dos profile named rs_dosl7, verify that proactive bot defense is now enabled.



on the bigip, check the bot request log, verify that requests are being challenged



this concludes the tests in the 'DEV' environment. we are now ready to push the changes to production.

Task 1.3 - Enable proactive bot defense in the PROD environment

we will 'merge' the app2 dev branch with the master branch so that the production deployment will use the correct policy.

1.3.1 Merge app2 dev to master :

on the /home/snops/f5-rs-app2 folder:

```
git checkout master
git merge dev -m "enabled proactive bot defense"
```

1.3.2 view the automatic pipeline :

the merge will trigger a job in Jenkins that's configured to monitor this repo - *Push WAF policy*, open the *f5-rs-app2-prd* folder and navigate to the *Service deployment pipeline* , you should see the jobs running in up to a minute.

open the PRODUCTION BIG-IP, check that the DOSL7 profile named rs_dosl7 has the 'proactive bot defense' enabled.

check that requests are getting challenged in the bot event log.

Lab 2: (Dave) Account takeover protection (app layer encryption) - OPTIONAL

Background:

Application is up and running, sales on the site have seen a big growth. our support center started getting complaints from customers that their account is abused and they are charged with purcheses they never did. after further investigation it turns out that the user's credentials were stolen by a malware on the client side.

secops engineer suggests to turn on F5's application encryption on the login page, he configured a template profile with some settings that make sense for the enterprise. exposing the login page paramters (URI), and a choice to enable/disable.

Task 4 - Enable application layer encryption

it is up to Dave now to deploy the new feature in DEV and promote to PROD when it makes sense for him.

- Open the container CLI
- go to the application git folder. check which branches are there and what is the active branch. (git branch)
- you should be on the 'dev' branch. the files you see belong to the dev branch.

```
cd /home/snops/f5-rs-app2
git checkout dev
git branch
```

- edit the iac_parameters.yaml file to enable login password encryption,
- change the setting from:
  - login_password_encryption: "disabled"
  - to:
  - login_password_encryption: "enabled"
- add the file to git and commit

```
vi iac_parameters.yaml
git add iac_parameters.yaml
git commit -m "enabled login password encryption"
```

- go back to jenkins and open the 'f5-rs-app2-dev ' folder. choose the 'Service deployment pipeline' tab , jenkins is set up to monitor the application repo. when a 'commit' is identified jenkins will start an automatic pipeline to deploy the service. it takes up to a minute for jenkins to start the pipeline.
- jenkins takes the parametes from the git repo and uses them to deploy/update the service.
- log on to the dev BIG-IP again, check the setting on the FPS profile.

this concludes the tests in the 'dev' environment. we are now ready to push the changes to production. we will 'merge' the app2 dev branch with the master branch so that the production deployment will use the correct policy. on the /home/snops/f5-rs-app2 folder:

```
git checkout master
git merge dev -m "enabled login password encryption"
```

the merge will trigger a job in jenkins that's configured to monitor this repo - 'Push waf policy', open the f5-rs-app2-prd folder and navigate to the 'service deployment pipeline' , you should see the jobs running in up to a minute.

open the PRODUCTION BIG-IP, check that the FPS profile named rs_fps has the 'login_password_encryption' enabled.

2. Start a solution

CHAPTER 2

Running the container on your docker host

Note: The following instructions will create a volume on your docker host and will instruct you to store private information in the host volume. the information in the volume will persist on the host even after the container is terminated.

2.1 1. run the rs-container

```
docker pull f5usecases/f5-rs-container
docker run -it --name rs-container -v config:/home/snops/host_volume -v jenkins:/var/
↪jenkins_home -p 2222:22 -p 10000:8080 --rm f5usecases/f5-rs-container
```

The container exposes the following access methods:

- SSH to RS-CONTAINER ssh://localhsot:2222
- HTTP Access to Jenkins http://localhost:10000 (only available after you start the lab)

2.1.1 1.1 Connect using SSH to the RS-CONTAINER

- SSH to dockerhost:2222
- username: *root*
- password: *default*

2.1.2 1.2 initial setup or skip to solutions if already completed the initial setup

- Move on to configure the container:

2.2 2. Start a solution

CHAPTER 3

Module Index

3.1 f5_rs_aws_net - Deploys vpc and network objects to an AWS region

New in version 0.9.

3.1.1 Synopsis

- Deploys vpc and network objects to an AWS region.

3.1.2 Requirements (on host that executes module)

- f5-sdk >= 3.0.9
- ansible >= 2.4
- boto3 >= 1.6.4

3.1.3 Options

3.1.4 Examples

Deploy:

ansible-playbook –vault-password-file ~/.vault_pass.txt -i inventory/hosts playbooks/aws_net_deploy.yaml -e "deploymentName=yossir100 aws_region=us-west-2"

Destroy:

ansible-playbook –vault-password-file ~/.vault_pass.txt -i inventory/hosts playbooks/aws_net_deploy.yaml -e "deploymentName=yossir100 aws_region=us-west-2 cft_state=absent"

3.1.5 Return Values

Return values are stored in the following etcd path:

f5_rs_aws_net/<deploymentName>/

3.1.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.

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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.2 f5_rs_aws_app - Creates an application in an auto-scale group

New in version 2.4.

3.2.1 Synopsis

- Creates an application in an auto-scale group in a given AWS vpc

3.2.2 Requirements (on host that executes module)

- aws

3.2.3 Options

3.2.4 Examples

Deploy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_app_deploy.yaml -e “aws_region=”$(etcdctl get f5-rs-aws-net/yossir100/aws_region)” applicationSubnets=”$(etcdctl get f5-rs-aws-net/yossir100/applicationSubnets)” deploymentName=yossir100 service_name=App1 vpc=”$(etcdctl get f5-rs-aws-net/yossir100/vpc)”“

Destroy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_app_deploy.yaml -e “aws_region=”$(etcdctl get f5-rs-aws-net/yossir100/aws_region)” applicationSubnets=”$(etcdctl get f5-rs-aws-net/yossir100/applicationSubnets)” deploymentName=yossir100 service_name=App1 vpc=”$(etcdctl get f5-rs-aws-net/yossir100/vpc)” rs_state=absent”

3.2.5 Return Values

Return values are stored in the following etcd path:

f5_rs_aws_app/<deploymentName>/

3.2.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.

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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.3 f5_rs_aws_bigip - deploys bigip in AWS using CFT

New in version 2.4.

3.3.1 Synopsis

- deploys bigip in AWS using CFT, currently supports only the WAF-autoscale CFT

3.3.2 Requirements (on host that executes module)

- aws

3.3.3 Options

3.3.4 Examples

Deploy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_bigip_deploy.yaml -e ” deploymentName=yossir100 service_name=App1 aws_region=”$(etcdctl get f5-rs-aws-net/yossir100/aws_region)” vpc=”$(etcdctl get f5-rs-aws-net/yossir100/vpc)” subnets=”$(etcdctl

get f5-rs-aws-net/yossir100/subnets)" bigipElasticLoadBalancer="$(etcdctl get f5-rs-aws-external-lb/yossir100/bigipElasticLoadBalancer)" applicationPoolTagValue="$(etcdctl get f5-rs-aws-app/yossir100/appAutoscaleGroupName)""

Destroy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_bigip_deploy.yaml -e " deploymentName=yossir100 service_name=App1 aws_region="$(etcdctl get f5-rs-aws-net/yossir100/aws_region)" vpc="$(etcdctl get f5-rs-aws-net/yossir100/vpc)" subnets="$(etcdctl get f5-rs-aws-net/yossir100/subnets)" bigipElasticLoadBalancer="$(etcdctl get f5-rs-aws-external-lb/yossir100/bigipElasticLoadBalancer)" applicationPoolTagValue="$(etcdctl get f5-rs-aws-app/yossir100/appAutoscaleGroupName)" state=absent"

3.3.5 Return Values

Return values are stored in the following etcd path:

f5_rs_aws_app/<deploymentName>/

3.3.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.

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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.4 bigip_command - Run arbitrary command on F5 devices

New in version 2.4.

3.4.1 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.

3.4.2 Requirements (on host that executes module)

- f5-sdk >= 3.0.9

3.4.3 Options

3.4.4 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
  register: result
  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
```

(continues on next page)

(continued from previous page)

```
- 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
  register: result
  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

- name: Delete all LTM nodes in Partition1, assuming no dependencies exist
  bigip_command:
    commands:
      - delete ltm node all
    chdir: Partition1
    server: lb.mydomain.com
    password: secret
    user: admin
    validate_certs: no
  delegate_to: localhost
```

3.4.5 Return Values

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

3.4.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.
- 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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.5 f5_rs_aws_external_lb - Creates an ELB on a given AWS vpc

New in version 2.4.

3.5.1 Synopsis

- Creates an ELB on a given AWS vpc

3.5.2 Requirements (on host that executes module)

- f5-sdk >= 3.0.9

3.5.3 Options

3.5.4 Examples

Deploy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_external_elb_deploy.yaml -e “deploymentName=yossir100 service_name=App1 aws_region=”$(etcdctl get f5-rs-aws-net/yossir100/aws_region)” vpc=”$(etcdctl get f5-rs-aws-net/yossir100/vpc)” subnets=”$(etcdctl get f5-rs-aws-net/yossir100/subnets)””

Destroy:

ansible-playbook –vault-password-file ~/.vault_pass.txt playbooks/aws_external_elb_deploy.yaml -e “deploymentName=yossir100 service_name=App1 aws_region=”$(etcdctl get f5-rs-aws-net/yossir100/aws_region)” vpc=”$(etcdctl get f5-rs-aws-net/yossir100/vpc)” subnets=”$(etcdctl get f5-rs-aws-net/yossir100/subnets)” state=absent”

3.5.5 Return Values

Return values are stored in the following etcd path:

f5_rs_aws_external_lb/<deploymentName>/

3.5.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.

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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.6 bigip_command - Run arbitrary command on F5 devices

New in version 2.4.

– *Support*

3.6.1 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.

3.6.2 Requirements (on host that executes module)

- f5-sdk >= 3.0.9

3.6.3 Options

3.6.4 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
  register: result
  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
```

(continues on next page)

(continued from previous page)

```
    wait_for:
      - result[0] contains BIG-IP
      - result[1] contains my-vs
    server: lb.mydomain.com
    password: secret
    user: admin
    validate_certs: no
  register: result
  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

- name: Delete all LTM nodes in Partition1, assuming no dependencies exist
  bigip_command:
    commands:
      - delete ltm node all
    chdir: Partition1
    server: lb.mydomain.com
    password: secret
    user: admin
    validate_certs: no
  delegate_to: localhost
```

3.6.5 Return Values

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

3.6.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.
- 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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.7 f5_rs_attacks - Run attacks on an HTTP/S target

New in version 2.4.

3.7.1 Synopsis

- Run attacks on an HTTP/S target

3.7.2 Requirements (on host that executes module)

- curl

3.7.3 Options

3.7.4 Examples

```
Run:
ansible-playbook -vv --vault-password-file ~/.vault_pass.txt \
playbooks/http_attacks/cmd_attack.yaml -e "\
https_target="$(etcdctl get f5-rs-aws-external-lb/yossir100/bigipELBDnsName)"
 ↪"
```

3.7.5 Return Values

Returns the HTTP response from the server

3.7.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.
- 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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

3.8 bigip_command - Run arbitrary command on F5 devices

New in version 2.4.

3.8.1 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.

3.8.2 Requirements (on host that executes module)

- f5-sdk >= 3.0.9

3.8.3 Options

3.8.4 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
  register: result
  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
  register: result
  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
```

(continues on next page)

(continued from previous page)

```
    user: admin
    validate_certs: no
  delegate_to: localhost

- name: Delete all LTM nodes in Partition1, assuming no dependencies exist
  bigip_command:
    commands:
      - delete ltm node all
    chdir: Partition1
    server: lb.mydomain.com
    password: secret
    user: admin
    validate_certs: no
  delegate_to: localhost
```

3.8.5 Return Values

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

3.8.6 Notes

Note:

- For more information on using Ansible to manage F5 Networks devices see https://www.ansible.com/integrations/networks/f5.
- 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 /usage/support

For help developing modules, should you be so inclined, please read *Getting Involved*, Writing a Module and *Guidelines*.

CHAPTER 4

BIG-IP versions

F5 does not currently support the F5 Modules for Ansible. However, F5 provides informal support through a number of channels. For details, see support.

The informal support F5 provides is for BIG-IP version 12.0.0 and later.

For a detailed list of BIG-IP versions that are currently supported, see .

When a version of BIG-IP reaches end of technical support, it is supported until the next Ansible release.

For example, if a version of BIG-IP reaches end of technical support on January 1, and Ansible releases a new version on March 1, then the F5 Modules for Ansible are supported on that version of BIG-IP until March 1.

F5 does not back-port changes to earlier versions of Ansible.

F5 develops the Ansible modules in tandem with the REST API, and newer versions of BIG-IP provide better support for the REST API.

CHAPTER 5

Experimental vs. production modules

F5 modules are included when you install Ansible. These modules are informally supported by F5 employees.

F5 modules are also in the . These modules are also informally supported by F5 employees, but you should consider these modules to be experimental and not production-ready.

However, if an experimental module's DOCUMENTATION block has a completed `Tested platforms` section, then the module is likely complete and ready for use. You can file issues against modules that are complete.

```
# Tested platforms:
#
#     - 12.0.0
#
```

CHAPTER 6

How to get involved

Thank you for getting involved with this project.

You can contribute in a number of different ways.

Here is some information that can help set your expectations.

6.1 Developing and supporting your module

When you develop a module, it goes through review before F5 accepts it. This review process may be difficult at times, but it ensures the published modules are good quality.

You should *stay up to date* with this site's documentation about module development. As time goes on, things change and F5 and the industry adopt new practices; F5 tries to keep the documentation updated to reflect these changes.

If you develop a module that uses an out-of-date convention, F5 will let you know, and you should take the initiative to fix it.

6.2 What to work on

While module/solution development is the primary focus of most contributors, it's understandable that you may not know how to create modules, or may not have any interest in creating modules to begin with.

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

6.2.1 Documentation

Documentation help is always needed. F5 encourages you to submit documentation improvements.

6.2.2 Unit tests

The unit tests in the *test/* directory can always use work. Unit tests run fast and are not a burden on the test runner.

F5 encourages you to add more test cases for your particular usage scenarios or any other scenarios that are missing tests.

F5 adds enough unit tests to be reasonably comfortable that the code will execute correctly. This, unfortunately, does not cover many of the functional test cases. Writing unit test versions of functional tests is hugely beneficial.

6.2.3 New modules

Modules do not cover all of the ways you might use F5 products. If you find that a module is missing from the repo and you think F5 should add it, put those ideas on the Github Issues page.

6.2.4 New functionality for an existing module

If a module is missing a parameter that you think it should have, raise the issue and F5 will consider it.

6.2.5 Postman collections

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

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

If you provide F5 with the Postman collections, F5 can easily write the module itself.

And you get bonus points for collections that address differences in APIs between versions of BIG-IP.

6.2.6 Bugs

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

During the development process, F5 writes tests with specific user personas in mind. Your usage patterns may not reflect those personas.

Using the modules is the best way to get good code and documentation. If the documentation isn't clear to you, it's probably not clear to others.

Righting those wrongs helps you and future users.

CHAPTER 7

Guidelines

Follow these guidelines when developing F5 modules for Ansible.

7.1 Which API to use

In Ansible 2.2 and later, all new F5 modules must use the `f5-sdk`.

Prior to 2.2, modules used `bigsuds` (SOAP) or `requests` (REST).

To maintain backward compatibility of older modules, you can continue to extend modules that use `bigsuds`. `bigsuds` and `f5-sdk` can co-exist, but F5 recommends that you write all new features, and fix all bugs by using `f5-sdk`.

7.2 Module naming convention

Base the name of the module on the part of BIG-IP that the module manipulates. A good rule of thumb is to refer to the API the `f5-sdk` uses.

Don't further abbreviate names. If something is a well-known abbreviation because it is a major component of BIG-IP, you can use it, but don't create new ones independently (e.g., LTM, GTM, ASM, etc. are fine).

7.3 Adding new APIs

If a module you need does not exist yet, the REST API in the `f5-sdk` may not exist yet.

Refer to the following GitHub project to determine if the REST API exists:

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

If you want F5 to write an API, open an issue with this project.

7.4 Using the f5-sdk

Follow these guidelines for using the `f5-sdk` in the modules you develop. Here are the most common scenarios that you will encounter.

7.4.1 Importing

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

```
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')
```

You might wonder why you are doing this.

The answer is that Ansible runs automated tests specifically against your module, and they use an environment that doesn't include your module's dependencies.

Therefore, without the appropriate exception handlers, your PR will fail to pass when Ansible runs these upstream tests.

Example tests include, but are not limited to:

- ansible-test sanity –test import –python 2.6
- ansible-test sanity –test import –python 2.7
- ansible-test sanity –test import –python 3.5
- ansible-test sanity –test import –python 3.6

7.4.2 Connecting to BIG-IP

Connecting to an F5 product is automatic. You can control which product you are communicating with by changing the appropriate value in your *ArgumentSpec* class.

For example, to specify that your module is one that communicates with a BIG-IP, here is the minimum viable *ArgumentSpec*:

```
class ArgumentSpec(object):
    def __init__(self):
        self.argument_spec = dict()
        self.f5_product_name = 'bigip'
```

Note the special key *f5_product_name*. By changing this value, you are able to change the *ManagementRoot* that your module uses.

The following is a list of allowed values for this key:

- bigip

- bigiq
- iworkflow

Inside your module, the *ManagementRoot* is in the *ModuleManager* under the *self.client.api* object.

Use the object in the same way that you normally use the *ManagementRoot* of an `f5-sdk` product.

For example, this code snippet illustrates a "normal" method of using the `f5-sdk`:

```
mr = ManagementRoot("localhost", "admin", "admin", port='10443')
vs = mr.tm.ltm.virtuals.virtual.load(name='asdf')
```

The equivalent Ansible module code is:

```
# Assumes you provided "bigip" in your ArgumentSpec
vs = self.client.api.tm.ltm.virtuals.virtual.load(name='asdf')
```

7.4.3 Exception handling

If the code throws an exception, it is up to you to decide how to handle it.

For raising exceptions, use the exception class, *F5ModuleError*, provided with the *f5-sdk*, exclusively.

```
# Module code
...

try:
    result = self.want.api.tm.ltm.pools.pool.create(foo='bar')
except iControlUnexpectedHTTPError as ex:
    raise F5ModuleError(str(ex))

...
# End of module code
```

In all cases in which you encounter it, it is correct to catch internal exceptions and re-raise them (if necessary) with the *F5ModuleError* class.

7.5 Python compatibility

The Python code underlying the Ansible modules should 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

7.6 Automated testing

F5 recommends that you use the testing facilities paired with this repository. When you open PR's, F5's testing tools will run the PR against supported BIG-IP versions.

Because F5 has test harnesses, you do not need your own devices or VE instances to test (although if you do that's fine).

F5 currently has the following devices in the test harness:

- 12.0.0 (BIGIP-12.0.0.0.0.606)
- 12.1.0 (BIGIP-12.1.0.0.0.1434)
- 12.1.0-hf1 (BIGIP-12.1.0.1.0.1447-HF1)
- 12.1.0-hf2 (BIGIP-12.1.0.2.0.1468-HF2)
- 12.1.1 (BIGIP-12.1.1.0.0.184)
- 12.1.1-hf1 (BIGIP-12.1.1.1.0.196-HF1)
- 12.1.1-hf2 (BIGIP-12.1.1.2.0.204-HF2)
- 12.1.2 (BIGIP-12.1.2.0.0.249)
- 12.1.2-hf1 (BIGIP-12.1.2.1.0.264-HF1)
- 13.0.0 (BIGIP-13.0.0.0.0.1645)
- 13.0.0-hf1 (BIGIP-13.0.0.1.0.1668-HF1)