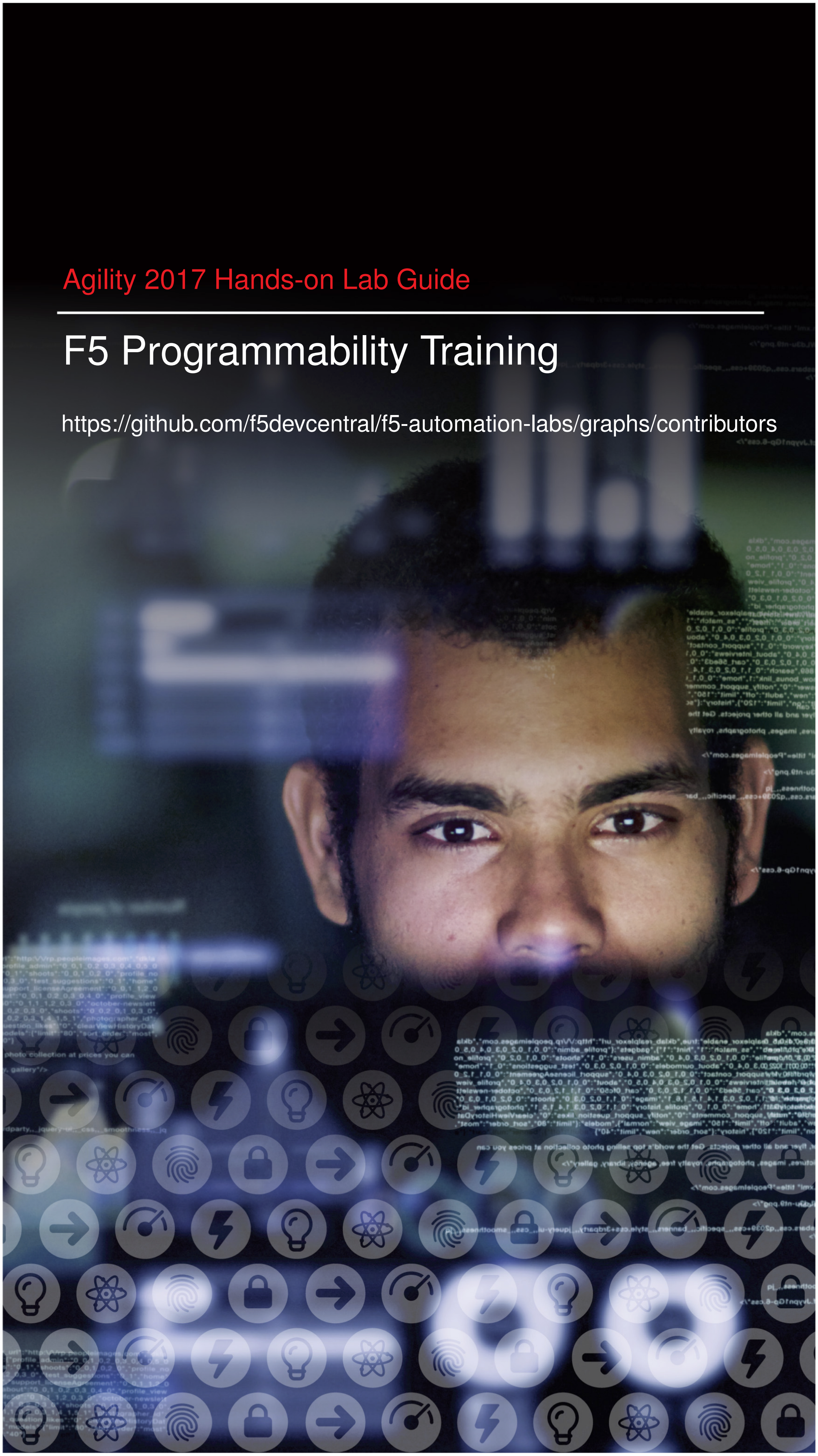


Agility 2017 Hands-on Lab Guide

F5 Programmability Training

https://github.com/f5devcentral/f5-automation-labs/graphs/contributors

Contents:

1 Welcome

Welcome to F5's Automation, Orchestration and Programmability Training series. The intended audience for these labs are Super NetOps and DevOps engineers that would like to leverage the various programmability tools offered by the F5 platform. If you require a pre-built lab environment please contact your F5 account team and they can provide access to environments on an as-needed basis.

The content contained here adheres to a DevOps methodology and automation pipeline. All content contained here is sourced from the following GitHub repository:

https://github.com/f5devcentral/f5-automation-labs/

Bugs and Requests for enhancements are handled in two ways:

- Fork the Github Repo, fix or enhance as required and submit a Pull Request
  - https://help.github.com/articles/creating-a-pull-request-from-a-fork/
- Open an Issue within the repository.

2

Class 1: Introduction to Automation & Orchestration

This introductory class covers the following topics:

- Imperative Automation using the F5 BIG-IP iControl REST API
- Service Abstraction and Automation using F5 iApp templates
- Building Declarative Interfaces with the F5 iWorkflow product

Expected time to complete: **4 hours**

To continue please review the information about the Lab Environment. Additionally, if you are new to the F5 BIG-IP Platform we've created an overview in the BIG-IP Basics section.

2.1 Lab Environments & Topology

Note: All work for this lab will be performed exclusively from the Linux Jumphost. No installation or interaction with your local system is required.

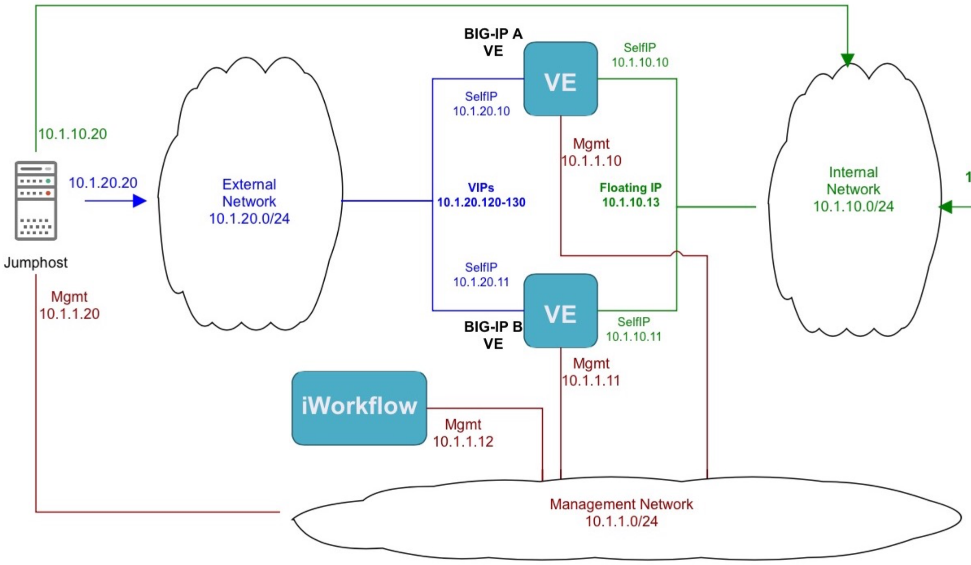
All pre-built environments implement the Lab Topology shown below. Please review the topology first, then find the section matching the lab environment you are using for connection instructions.

2.1.1 Lab Topology

The network topology implemented for this lab is very simple. Since the focus of the lab is Control Plane programmability rather than Data Plane traffic flow we can keep the data plane fairly simple. The following components have been included in your lab environment:

- 2 x F5 BIG-IP VE (v12.1.x)
- 1 x F5 iWorkflow VE (v2.3)
- 1 x Linux Webserver
- 1 x Linux Jumphost

Lab Topology



The following table lists VLANS, IP Addresses and Credentials for all components:

Component	Management IP	VLAN/IP Address(es)	Credentials
Linux Jumphost	10.1.1.20	**Internal:** 10.1.10.20 **External:** 10.1.20.20	ubuntu/supernetops
BIG-IP A	10.1.1.10	**Internal:** 10.1.10.10 **Internal (Float):** 10.1.10.13 **External:** 10.1.20.10 **External (VIPs):** 10.1.20.120-130	admin/admin root/default
BIG-IP B	10.1.1.11	**Internal:** 10.1.10.11 **Internal (Float):** 10.1.10.13 **External:** 10.1.20.11 **External (VIPs):** 10.1.20.120-130	admin/admin root/default
iWorkflow	10.1.1.12	N/A	admin/admin root/default
Linux Server	10.1.1.15	**Internal:** 10.1.10.100-103	root/default

2.1.2 Lab Environments

In order to complete this class you will need to utilize a specific **Lab Environment**. You can consume this training in a couple of ways:

- Pre-built Environment using a Ravello Blueprint
  - Used at official F5 events such as F5 Agility, F5 Agility Roadshows, User Groups, MeetUps, etc.
  - Access can be provided by your F5 Account Team

- Pre-built Environment using an Amazon AWS CloudFormation Template (CFT)
  - Access is on-demand and uses *your* AWS account
- Pre-built Environment using the F5 Unified Demo Framework (UDF)
  - This environment is currently available for F5 employees only
- Self-built Environment on your own infrastructure
  - Review the Topology for details

Select the Environment from the list below to get started:

Amazon AWS Lab Environment Guide

Warning: The AWS CFT will run in your account. The template includes components and instances that will incur a charge. This charge will be billed to your account.

Warning: The AWS CFT is currently in testing. You can complete Modules 1 & 2 of this class using the template at this time.

You can use an Amazon CloudFormation Template (CFT) to launch your own lab environment in AWS. This guide assumes the following:

- Pre-existing Amazon AWS account
- Access to create AWS Instances and Resources
- You have created an AWS Key Pair:
  - http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html
- *You are responsible for all charges incurred*

More information about AWS can be found here:

https://aws.amazon.com/

Task 1 - Determine your Source IP Address

The AWS lab environment restricts access based on your Source IP Address. We will use a website to determine your Source IP for use in the next Task.

Note: If you Source IP address changes you will lose access to your environment.

Perform the following steps to complete this task:

1. Open a web browser window or tab and navigate to https://www.whatismyip.com/
2. Copy the IP Address shown in the *Your IP Address is:* box into your clipboard

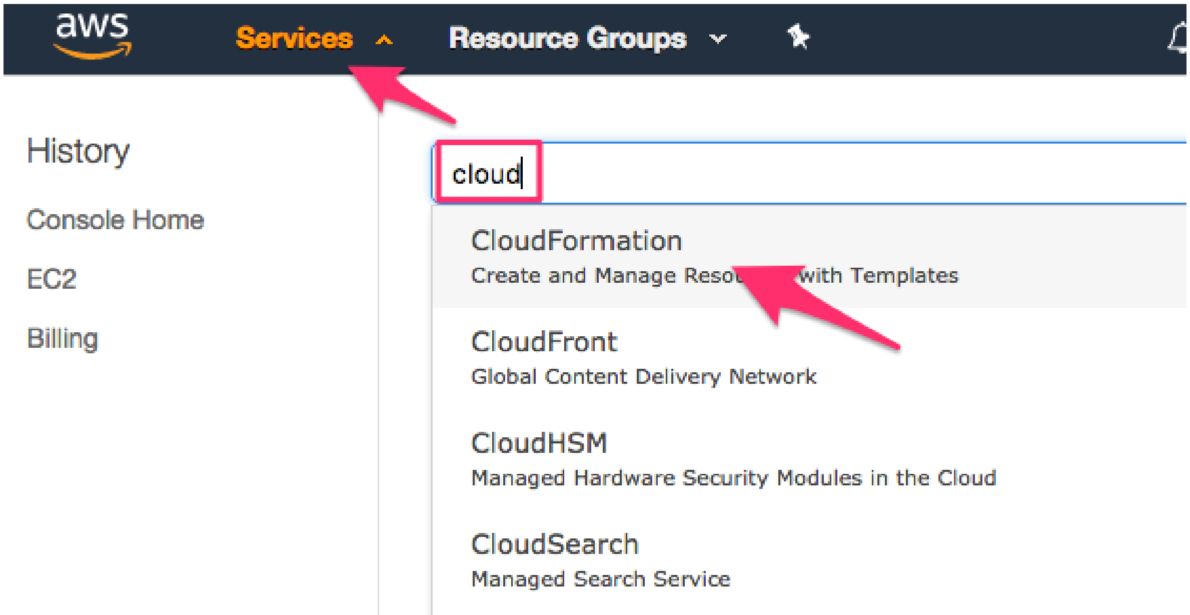
Task 2 - Launch the CloudFormation Template

Perform the following steps to complete this task:

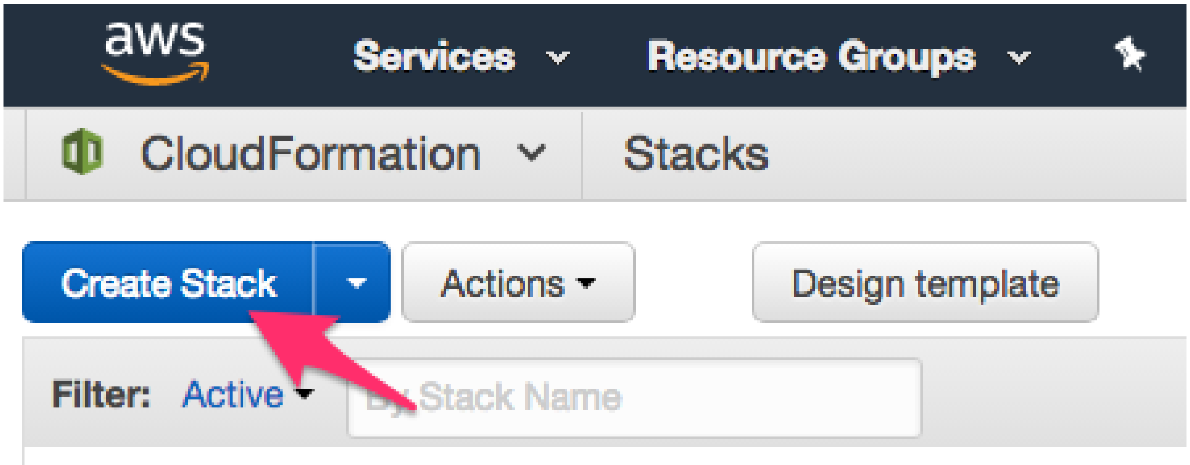
1. Login to your AWS Management Console

> **Note:** Access to the console is determined by your individual account setup.
>
> If you are using a personal account you should be able to login using https://console.aws.amazon.com/console/home
>
> If you are using a corporate account please contant your IT Help Desk

2. Click *Services* and the top of the windows. Then type `cloud` into the search box and find the *CloudFormation* item. Click the *CloudFormation* item:



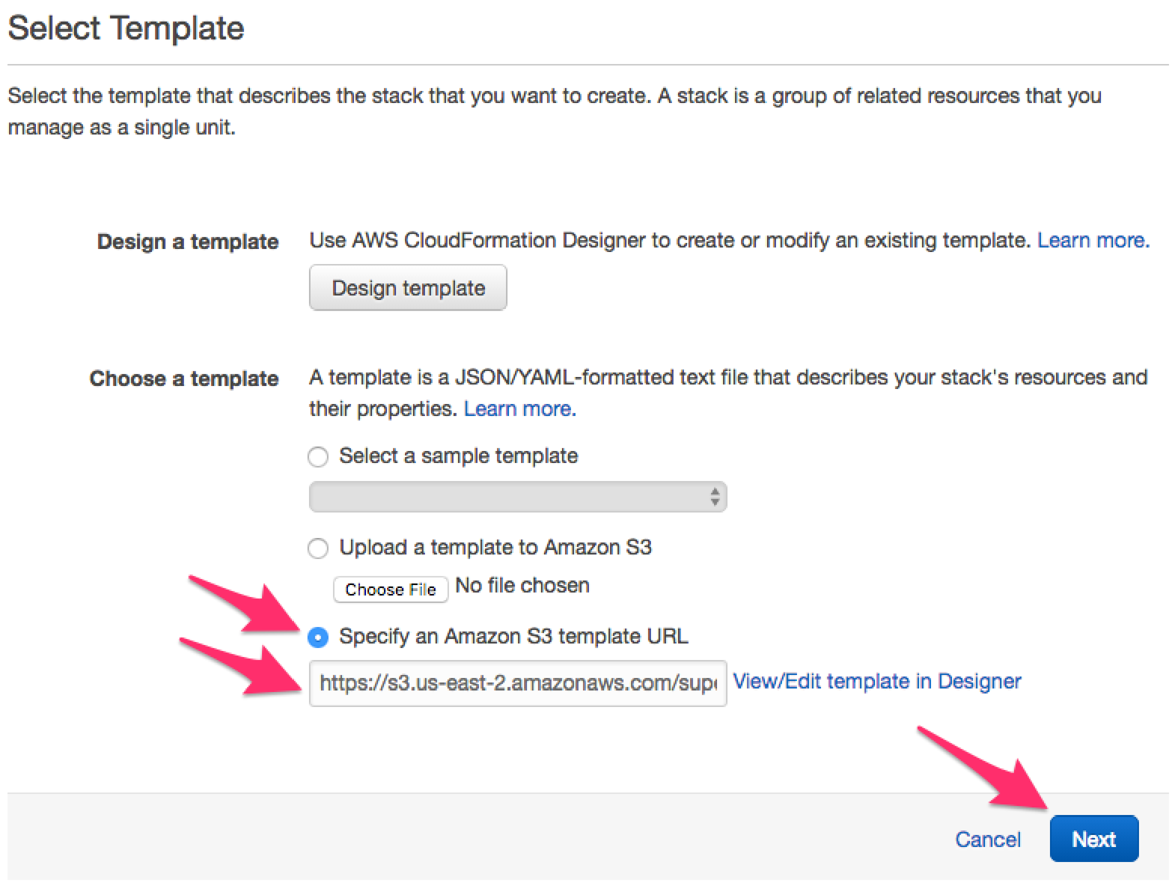
3. Click the *Create Stack* button:



4. On the *Select Template* screen select the *Specify an Amazon S3 template URL* option. Copy and paste the URL below into the box:

```
https://s3.us-east-2.amazonaws.com/supernetops-cf-templates/class1.template
```

Click the *Next* button



Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. Learn more.

Choose a template A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. Learn more.

○ Select a sample template

○ Upload a template to Amazon S3

No file chosen

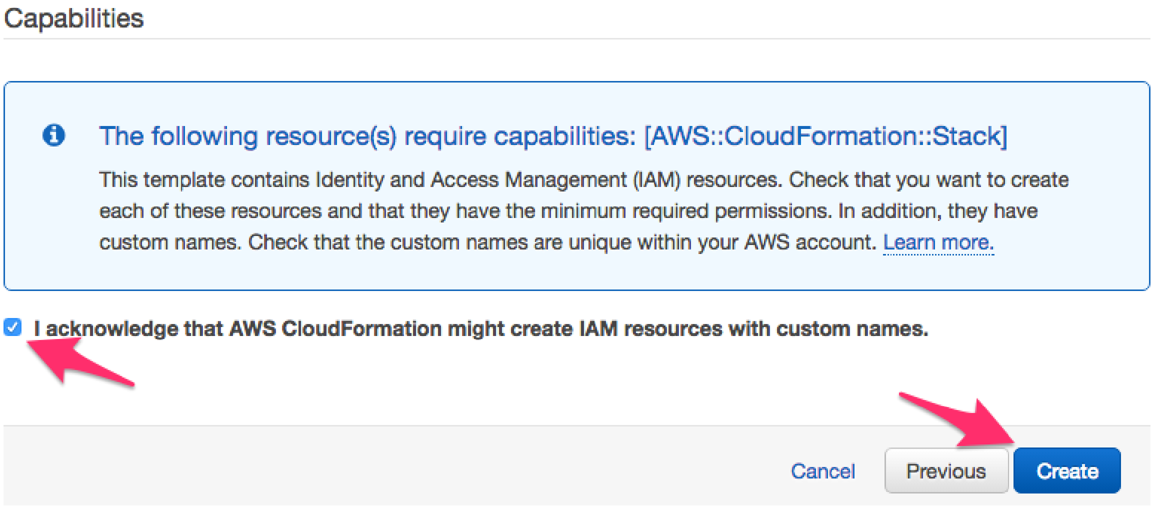
● Specify an Amazon S3 template URL

https://s3.us-east-2.amazonaws.com/supe

5. Complete the form in the *Specify Details* screen:
   - *Stack Name*: `Super-NetOps-Lab` or a name of your choice
   - *Branch*: `master`
   - *InstanceType*: `t2.medium`
   - *KeyName*: Select your AWS Key Pair
   - *UserIP*: Paste the IP Address from Task 1 and add `/32` to to the end.

   Note: You can also specify a CIDR formatted Subnet in this field

6. Click the *Next* button
7. On the *Options* screen click the *Next* button at the bottom of the screen
8. On the *Review* screen check the `I acknowledge that AWS CloudFormation might create IAM resources with custom names` field and click the *Create* button at the bottom of the screen



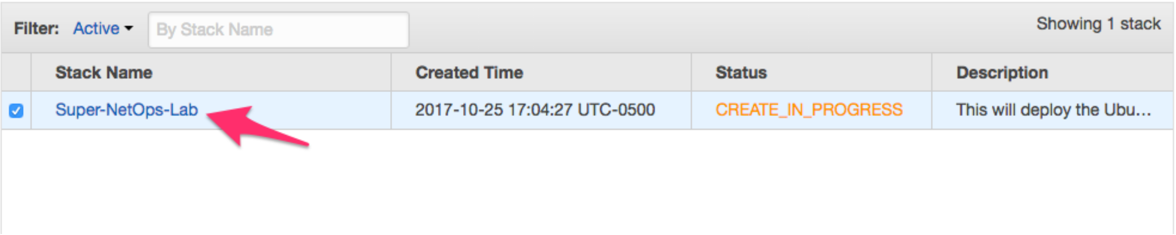
Capabilities

The following resource(s) require capabilities: [AWS::CloudFormation::Stack]

This template contains Identity and Access Management (IAM) resources. Check that you want to create each of these resources and that they have the minimum required permissions. In addition, they have custom names. Check that the custom names are unique within your AWS account. Learn more.

☑ **I acknowledge that AWS CloudFormation might create IAM resources with custom names.**

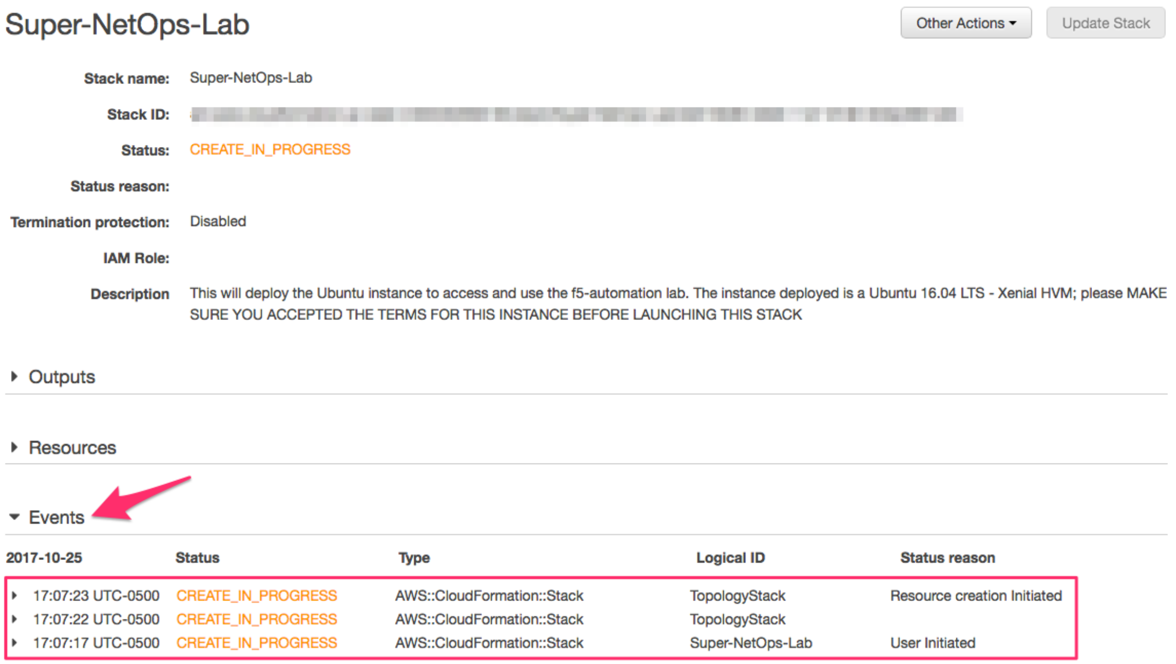
9. Click the `Super-NetOps-Lab` stack to view details of the deployment



Filter: Active ▾ By Stack Name — Showing 1 stack

	Stack Name	Created Time	Status	Description
☑	Super-NetOps-Lab	2017-10-25 17:04:27 UTC-0500	CREATE_IN_PROGRESS	This will deploy the Ubu…

10. Monitor the *Events* section of the page as the Stack deploys:



Super-NetOps-Lab

Stack name: Super-NetOps-Lab

Stack ID: [redacted]

Status: CREATE_IN_PROGRESS

Status reason:

Termination protection: Disabled

IAM Role:

Description This will deploy the Ubuntu instance to access and use the f5-automation lab. The instance deployed is a Ubuntu 16.04 LTS - Xenial HVM; please MAKE SURE YOU ACCEPTED THE TERMS FOR THIS INSTANCE BEFORE LAUNCHING THIS STACK

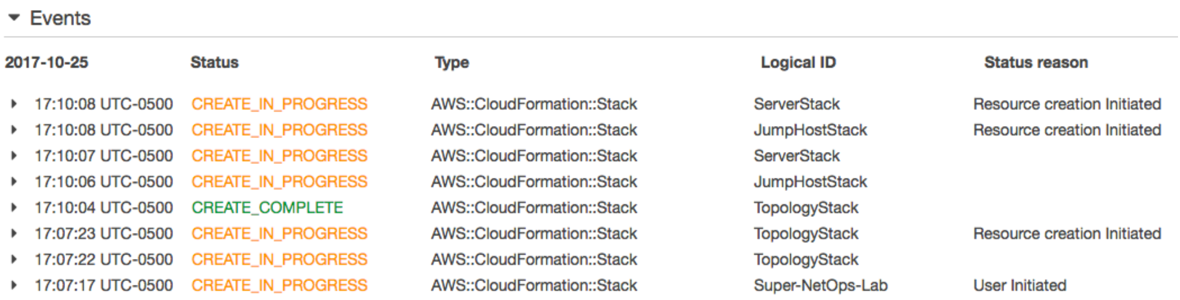
▸ Outputs

▸ Resources

▾ Events

2017-10-25	Status	Type	Logical ID	Status reason
▸ 17:07:23 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	TopologyStack	Resource creation Initiated
▸ 17:07:22 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	TopologyStack	
▸ 17:07:17 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	Super-NetOps-Lab	User Initiated

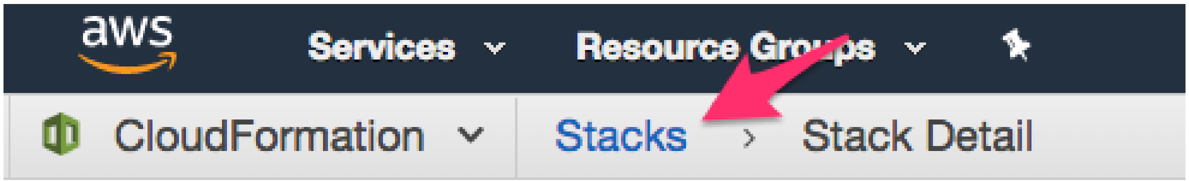
11. The CFT used performs a nested deployment, leveraging other CFT's. The *Events* will notify you when new status messages are available. Total deployment time varies. As the Stack is being deployed you will see periodic `CREATE_COMPLETED` messages:

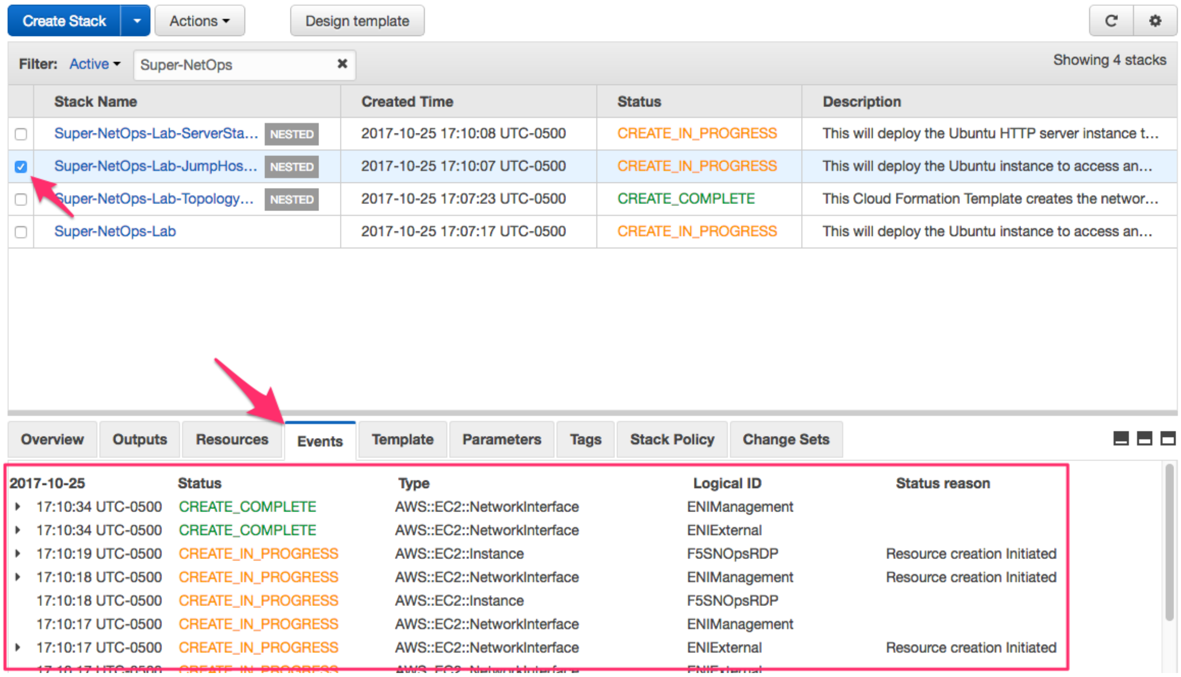


▾ Events

2017-10-25	Status	Type	Logical ID	Status reason
▸ 17:10:08 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	ServerStack	Resource creation Initiated
▸ 17:10:08 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	JumpHostStack	Resource creation Initiated
▸ 17:10:07 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	ServerStack	
▸ 17:10:06 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	JumpHostStack	
▸ 17:10:04 UTC-0500	CREATE_COMPLETE	AWS::CloudFormation::Stack	TopologyStack	
▸ 17:07:23 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	TopologyStack	Resource creation Initiated
▸ 17:07:22 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	TopologyStack	
▸ 17:07:17 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	Super-NetOps-Lab	User Initiated

12. You can also go back to the page listing Stacks and monitor the progres of the nested templates from there:



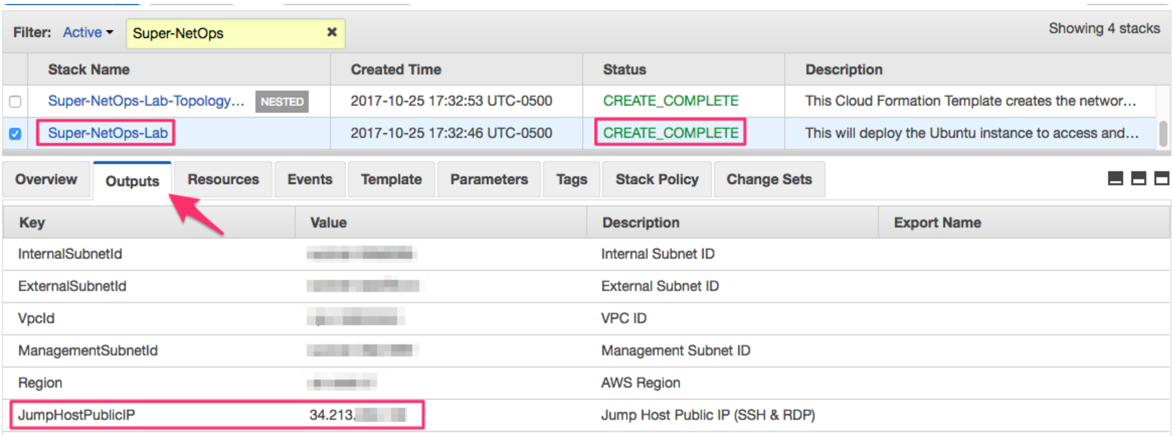


Filter: Active ▾ Super-NetOps ✖ Showing 4 stacks

	Stack Name	Created Time	Status	Description
□	Super-NetOps-Lab-ServerSta... NESTED	2017-10-25 17:10:08 UTC-0500	CREATE_IN_PROGRESS	This will deploy the Ubuntu HTTP server instance t...
☑	Super-NetOps-Lab-JumpHos... NESTED	2017-10-25 17:10:07 UTC-0500	CREATE_IN_PROGRESS	This will deploy the Ubuntu instance to access an...
□	Super-NetOps-Lab-Topology... NESTED	2017-10-25 17:07:23 UTC-0500	CREATE_COMPLETE	This Cloud Formation Template creates the networ...
□	Super-NetOps-Lab	2017-10-25 17:07:17 UTC-0500	CREATE_IN_PROGRESS	This will deploy the Ubuntu instance to access an...

2017-10-25	Status	Type	Logical ID	Status reason
▸ 17:10:34 UTC-0500	CREATE_COMPLETE	AWS::EC2::NetworkInterface	ENIManagement	
▸ 17:10:34 UTC-0500	CREATE_COMPLETE	AWS::EC2::NetworkInterface	ENIExternal	
▸ 17:10:19 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	F5SNOpsRDP	Resource creation Initiated
▸ 17:10:18 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::NetworkInterface	ENIManagement	Resource creation Initiated
17:10:18 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	F5SNOpsRDP	
17:10:17 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::NetworkInterface	ENIManagement	
▸ 17:10:17 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::NetworkInterface	ENIExternal	Resource creation Initiated
17:10:17 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::NetworkInterface	ENIExternal	

13. Once the *Status* of the `Super-NetOps-Lab` **root** stack shows `CREATE_COMPLETED` click the *Outputs* tab. You will see a *Key* named `JumpHostPublicIP`. The *Value* is the IP Address you can use to connect to the Lab Jumphost using RDP, HTTPS or SSH (diagnostics only).



Filter: Active ▾ Super-NetOps ×

Showing 4 stacks

	Stack Name	Created Time	Status	Description
□	Super-NetOps-Lab-Topology... NESTED	2017-10-25 17:32:53 UTC-0500	CREATE_COMPLETE	This Cloud Formation Template creates the networ...
☑	Super-NetOps-Lab	2017-10-25 17:32:46 UTC-0500	CREATE_COMPLETE	This will deploy the Ubuntu instance to access and...

Key	Value	Description	Export Name
InternalSubnetId	[redacted]	Internal Subnet ID	
ExternalSubnetId	[redacted]	External Subnet ID	
VpcId	[redacted]	VPC ID	
ManagementSubnetId	[redacted]	Management Subnet ID	
Region	[redacted]	AWS Region	
JumpHostPublicIP	34.213.[redacted]	Jump Host Public IP (SSH & RDP)	

14. You can now connect to the Jumphost using RDP or HTTPS:
    - RDP: Configure your RDP client to connect to the `JumpHostPublicIP`
    - HTTPS: Using an HTML5 browser connect to `https://<JumpHostPublicIP>`
15. Select how you would like to continue:
    - Review: *BIG-IP Basics (optional)*
    - Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

Ravello

If you are taking this class at a Meetup, User Group, F5 Agility or another official event, access details will be provided by your instructor.

If you would like to take this class using our lab environment please contact your F5 Account Team for access.

Once you have connected to your environment you can select how you would like to continue:

- Review: *BIG-IP Basics (optional)*
- Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

F5 Unified Demo Framework (UDF)

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 `Intro to Automation & Orchestion` Blueprint and *Start* it.

Connecting to the Environment

The lab environment provides two methods of access to the Jumphost:

- RDP Connection using an RDP Client
- HTML5 Browser based VNC Connection using noVNC
  - Chrome
  - Firefox
  - Safari
  - EDGE

Connect using RDP

1. In the UDF navigate to your *Deployments*
2. Click the *Details* button for your Deployment
3. Click the *Components* tab
4. Find the `Linux Jumphost` Component and click the the *Access* button. Then click the *RDP* option. An RDP file will be downloaded to your system.

   Note: The RDP file opens the session in Full Screen mode by default. You may want to open the file in an RDP client and adjust these settings as needed to match your preference.

5. If you have the official Microsoft Remote Desktop Client installed please install it using the instructions at https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients
6. **Warning:** If you have a HiDPI display please refer to the *Using HiDPI Displays (RDP & Windows)* instructions below
7. Open the RDP file in the Remote Desktop Client and connect. If you have any problems please ask your instructor for help
8. Select how you would like to continue:
   - Review: *BIG-IP Basics (optional)*
   - Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

Connect using an HTML5 Browser

1. In the UDF navigate to your *Deployments*
2. Click the *Details* button for your Deployment
3. Click the *Components* tab
4. Find the `Linux Jumphost` Component and click the the *Access* button. Then click the *NOVNC* option. A new browser window/tab will be opened.
5. In the new browser window/tab click the *Connect* button followed by the *Send Password* button. You should now be connected. If you have any problems please ask your instructor for help
6. Select how you would like to continue:

- Review: *BIG-IP Basics (optional)*
- Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

Using HiDPI Displays (RDP & Windows)

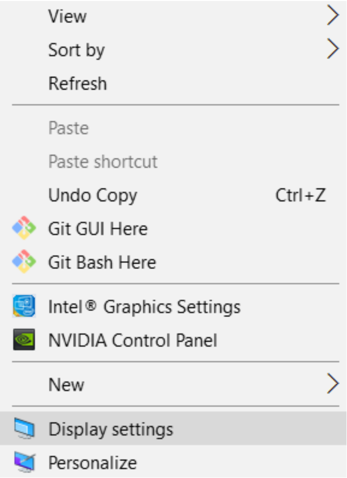
Warning: Do these steps BEFORE you connect via RDP. Choose ONE. Do not do both.

If you are using a Hi Resolution Display on Windows we recommend that you either:

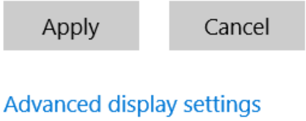
- **RECOMMENDED:** Resize your display to 1080p (1920 x 1080)
- Use your RDP clients "Zoom" funcitonality to increase the size

Resize your display (Windows 10) - RECOMMENDED

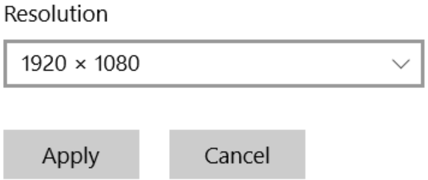
1. Right click on your Desktop and select Display Settings.



2. Click on *Advanced Display Settings*



3. Change the Resolution to `1920 x 1080` and click the *Apply* button


Resolution

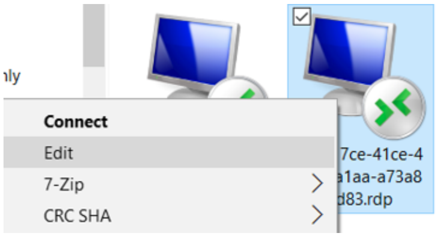
1920 × 1080

4. Connect to the RDP session

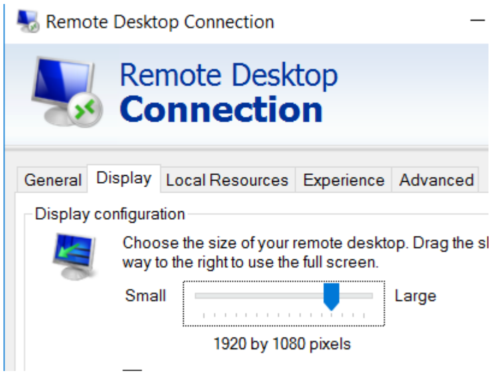
5. Select how you would like to continue:
   - Review: *BIG-IP Basics (optional)*
   - Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

Use RDP Zoom

1. Right click on the RDP file and click on *Edit*



2. Under the *Display* tab change the resolution to `1920x1080`, then click *Connect*


Remote Desktop Connection

Remote Desktop Connection

General | Display | Local Resources | Experience | Advanced

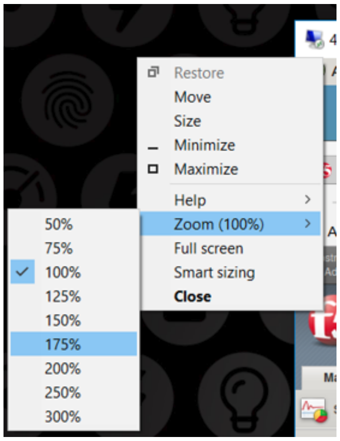
Display configuration

Choose the size of your remote desktop. Drag the sl
way to the right to use the full screen.

Small | Large

1920 by 1080 pixels

3. After you connect, access the menu at the top left of your RDP Window and change the Zoom level (i.e. 175%).



4. Select how you would like to continue:
   - Review: *BIG-IP Basics (optional)*
   - Start: *Module 1: Imperative Automation with the BIG-IP iControl REST API*

2.2 BIG-IP Basics (optional)

Just in case you're new to the F5 BIG-IP platform (or need a refesher) we've included some links and videos below that will help get you started.

2.2.1 What is BIG-IP

Source: https://devcentral.f5.com/articles/lightboard-lessons-what-is-big-ip-26793

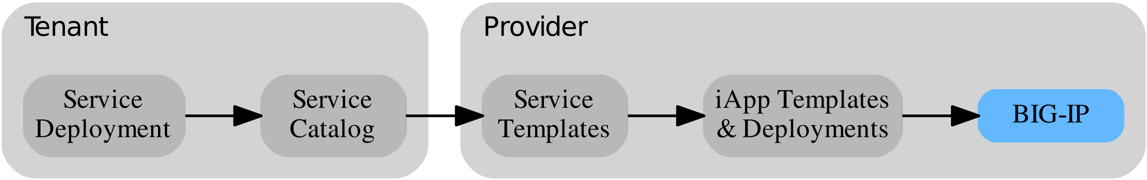
2.2.2 BIG-IP Basic Nomenclature

Source: https://devcentral.f5.com/articles/lightboard-lessons-big-ip-basic-nomenclature-26144

2.2.3 F5 DevCentral BIG-IP Basics Articles

BIG-IP Basics Articles: https://devcentral.f5.com/articles?tag=devcentral+basics

2.3 Module 1: Imperative Automation with the BIG-IP iControl REST API



In this module you will learn the basic concepts required to interact with the BIG-IP iControl REST API. Additionally, you will walk through a typical Device Onboarding workflow that results in a fully functional BIG-IP Active/Standby pair. It's important to note that this module will focus on showing an **Imperative** approach to automation.

Note: The Lab Deployment for this lab includes two BIG-IP devices. For most of the labs we will focus on configuring only the BIG-IP-A device (management IP and licensing have already been completed). BIG-IP-B already has some minimal configuration loaded. In a real-world environment it would be necessary to perform Device Onboarding functions on ALL BIG-IP devices. We are only performing them on a single device in this lab so we are able to cover all topics in the time allotted.

Note: In order to confirm the results of REST API calls made in this lab, it's beneficial to have GUI/SSH sessions open to BIG-IP and iWorkflow devices. By default, BIG-IP and iWorkflow will log all REST API related events locally to **restjavad.0.log** and also can be configured to log to a remote syslog server (see https://support.f5.com/csp/article/K13080). Additionally, the **ltm** log file on BIG-IP will contain log messages that pertain specifically to BIG-IP local traffic management events. These log file locations are below:

- BIG-IP:
  - /var/log/ltm
  - /var/log/restjavad.0.log
- iWorkflow:
  - /var/log/restjavad.0.log

2.3.1 Lab 1.1: Exploring the iControl REST API

BIG-IP
REST Basics
Authentication
Global Settings
Networking
Clustering
Tran

Task 1 - Explore the API using the TMOS Web Interface

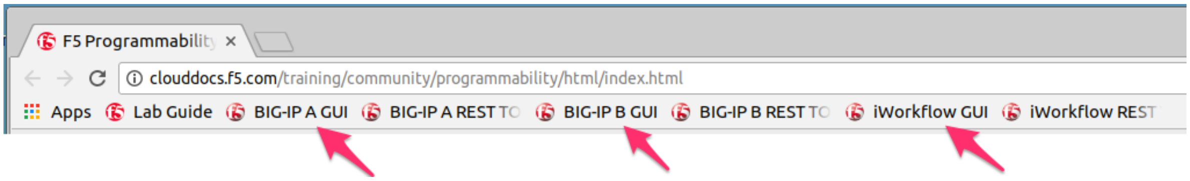
In this lab, we will explore the API using an interface that is built into TMOS. This utility is useful for understanding how TMOS objects map to the REST API. The interfaces implement full Create, Read, Update and Delete (CRUD) functionality, however, in most practical use cases it's far easier to use this interface as a 'Read' tool rather than trying to Create objects directly from it. You can use TMUI or TMSH to create the object as needed and then use this tool to view the created object with all the correct attributes already populated.

Note: This guide may require you to Copy/Paste information from the guide to your jumphost. To make this easier you can open a copy of the guide by using the **Lab Guide** bookmark in Chrome.

1. Open Google Chrome and navigate to the following bookmarks: **BIG-IP A GUI**, **BIG-IP B GUI** and **iWorkflow GUI**. Bypass any SSL errors that appear and ensure you see the login screen for each bookmark.

   Warning: Skipping this step will result in errors in subsequent steps

Warning: We are using self-signed certificate in this lab. In your environment you must make sure that you use certificate issued by your certificate authority for both production and lab equipments. Not doing so would make it possible for an attacker to do a man-in-the-middle attack and allow him the ability to steal passwords and tokens.



F5 Programmabilit ×

clouddocs.f5.com/training/community/programmability/html/index.html

2. Navigate to the URL `https://10.1.1.10/mgmt/toc` (or click the BIG-IP A REST TOC bookmark). The `/mgmt/toc` path in the URL is available on all TMOS versions 11.6 or newer.
3. Authenticate to the interface using the default credentials (`admin/admin`)
4. You will now be presented with a top-level list of various REST resources. At the top of the page there is a search box that can be used to find items on the page. Type `net` in the search box and then click on the 'net' link under iControl REST Resources:

filter worker...

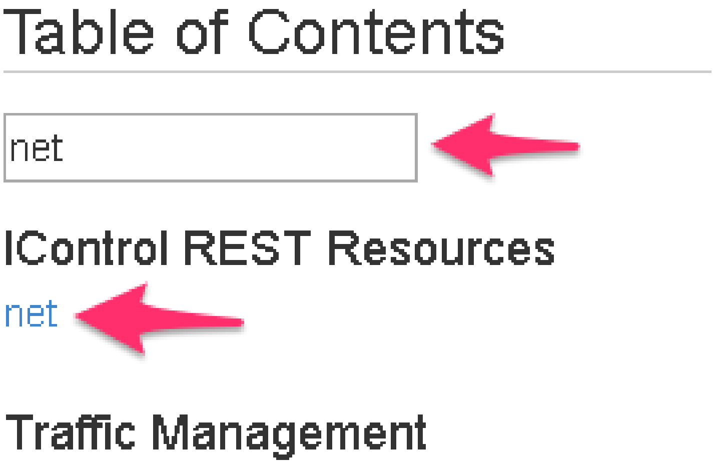


Table of Contents

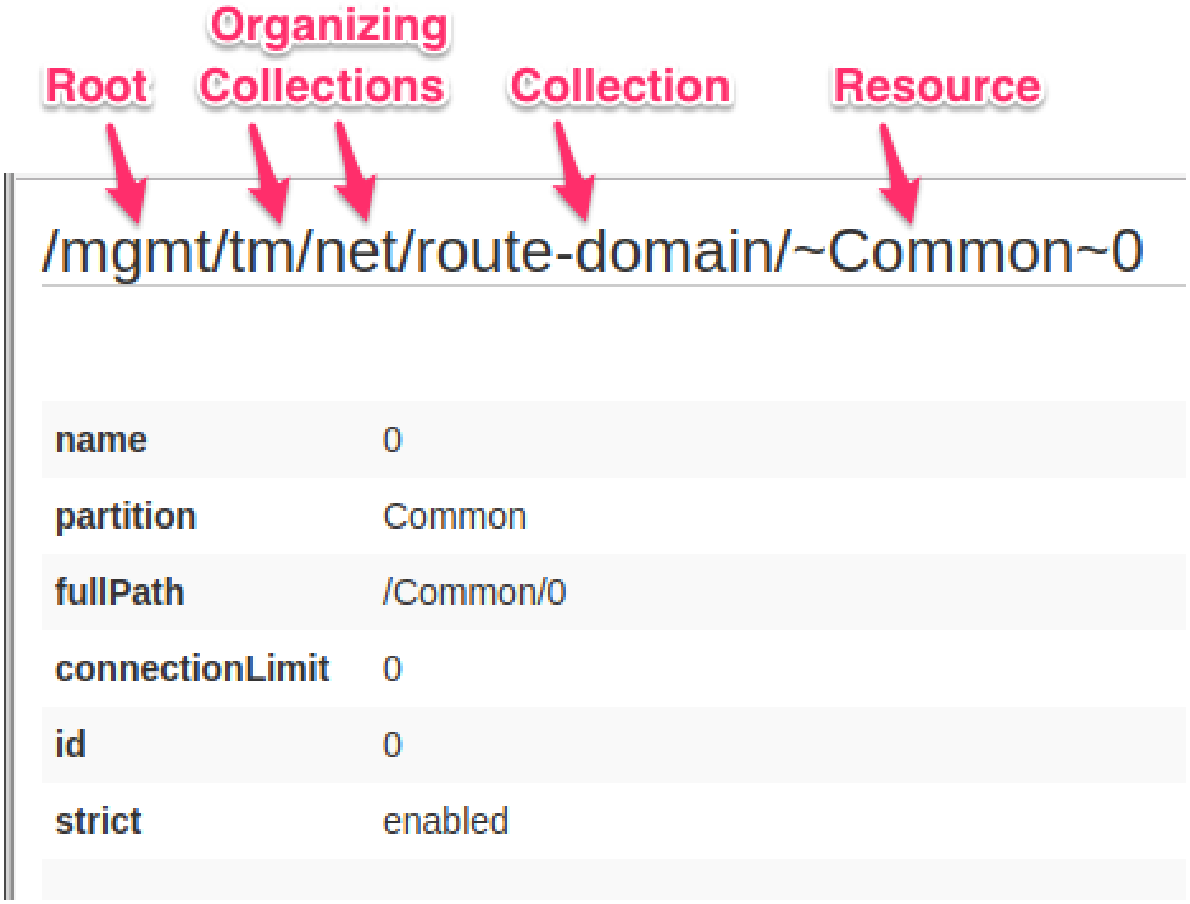
net

IControl REST Resources

net

Traffic Management

5. Find the `/mgmt/tm/net/route-domain` **Collection** and click it.
6. You will now see a listing of the **Resources** that are part of the route-domain(s) collection. As you can see the default route domain of `0` is listed. You can also create new objects by clicking the ➕ button. Additionally resources can be deleted using the 🗑 button or edited using the ✏ button. The ↪ is used to copy JSON formatted resource with Ctrl+C. This can be useful when you want to slightly change an existing resource
7. Click the `0` resource to view the attributes of route-domain 0 on the device:



/mgmt/tm/net/route-domain/~Common~0

name	0
partition	Common
fullPath	/Common/0
connectionLimit	0
id	0
strict	enabled

Note: If you would like to learn more about the iControl REST API be sure to read the **Demystifying iControl REST** article series at https://devcentral.f5.com/wiki/icontrolrest.homepage.ashx

2.3.2 Lab 1.2: REST API Authentication & 'example' Templates

BIG-IP
REST Basics
Authentication
Global Settings
Networking
Clustering
Tran

One of the many basic concepts related to interaction with REST API's is how a particular consumer is authenticated to the system. BIG-IP and iWorkflow support two types of authentication: **HTTP BASIC** and **Token-Based (TBA)**. It's important to understand both of these authentication mechanisms, as consumers of the API will often make use of both types depending on the use case. This lab will demonstrate how to interact with both types of authentication.

Throughout this and other classes in the series we will make use of the Postman REST API Client. You can find more information about Postman at https://getpostman.com

Task 1 - Import the Postman Collection & Environment

In this task you will Import a Postman Collection & Environment for this lab. Perform the following steps to complete this task:

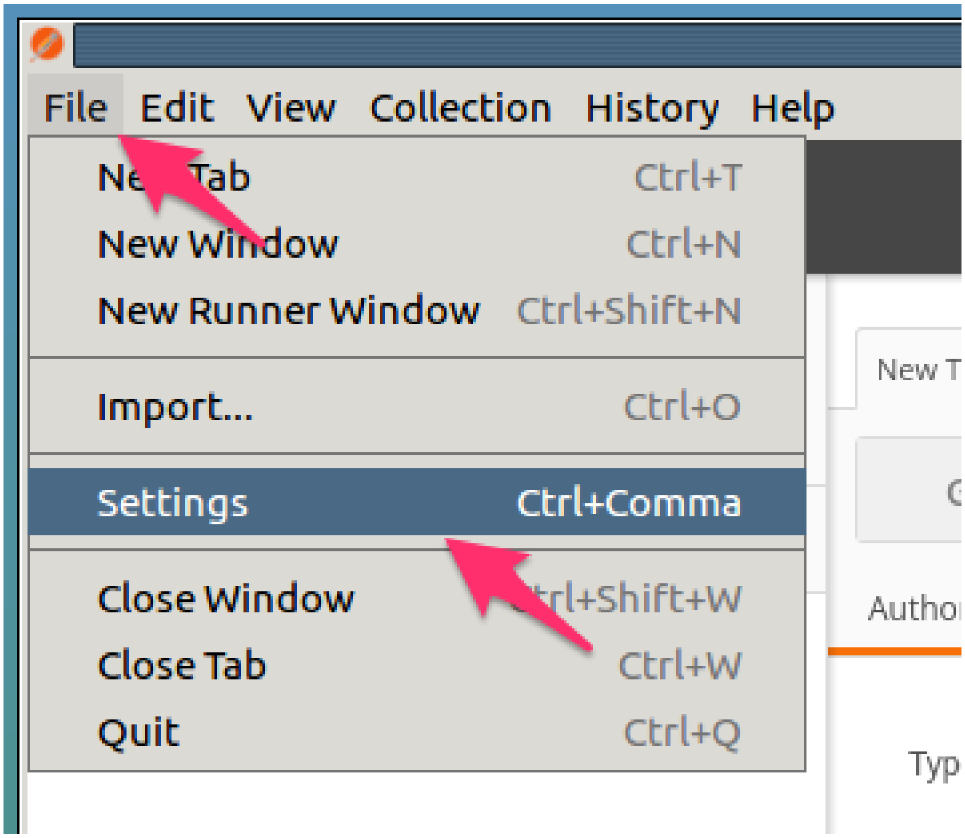
1. Open the Postman tool by clicking the Postman icon of the desktop of your Linux Jumphost. **The initial window may take a few moments to appear.**

   Note: The Postman client receives very frequent updates. If you are prompted to update the client please click the *Remind me later* button to skip updating the version installed in your lab environment

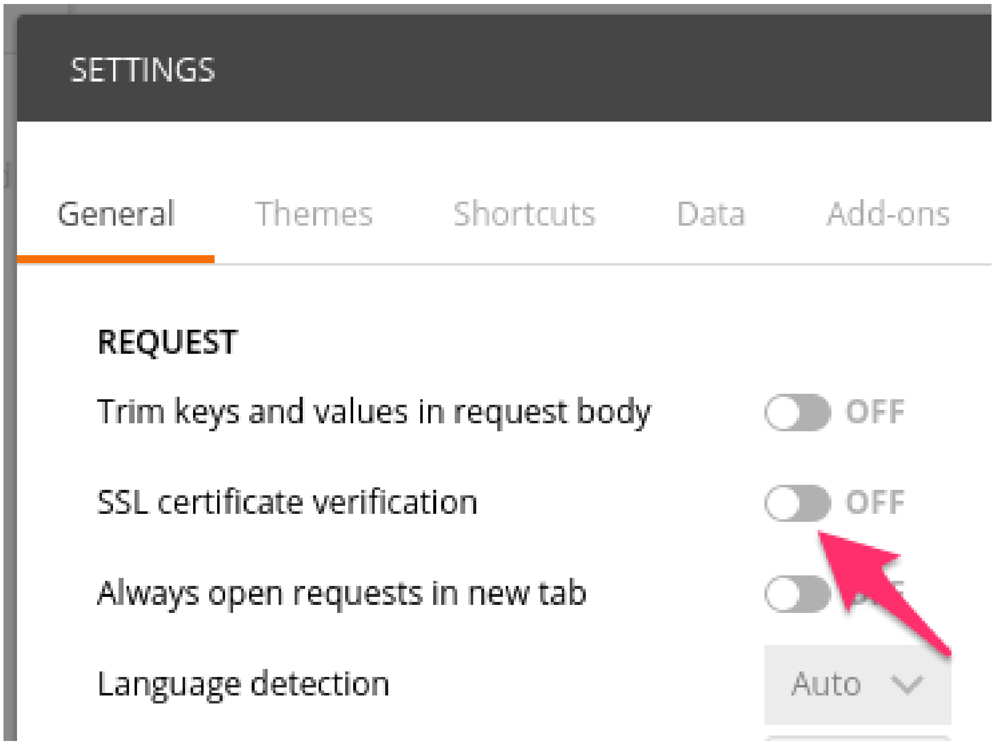
   ---

2. By default the Postman clients requires verification of SSL/TLS Certificates to a public Root Certificate Authority. By default BIG-IP, and many other, devices use a Self-Signed Certificate for SSL/TLS connections. To allow connections with Self-Signed Certificates we need to modify the default settings of Postman.

   - Open the Postman Settings windows by click *File* → *Settings*:



- Verify your client is configured to allow self-signed certificates by setting `SSL certificate verification` to `OFF`



REQUEST

Trim keys and values in request body OFF

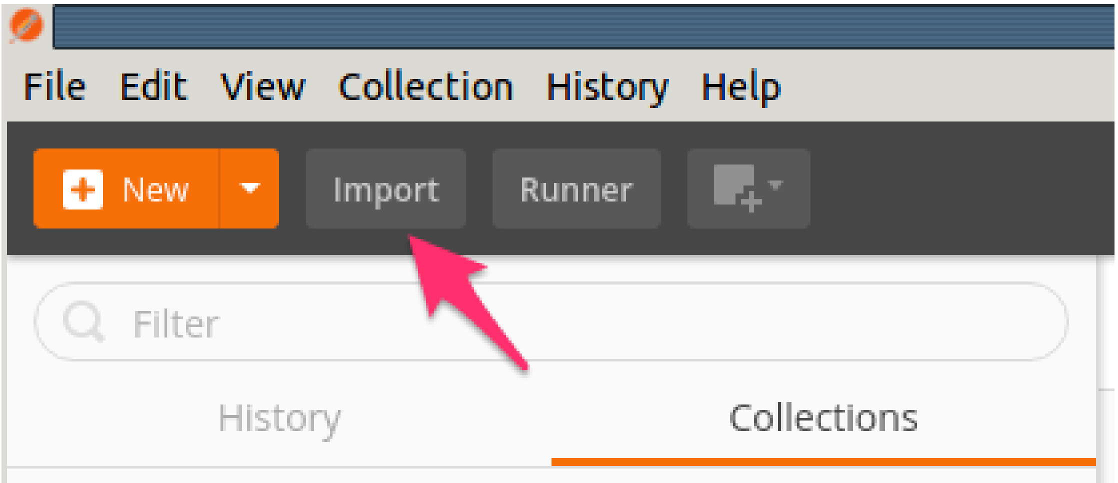
SSL certificate verification OFF

Always open requests in new tab OFF

Language detection Auto

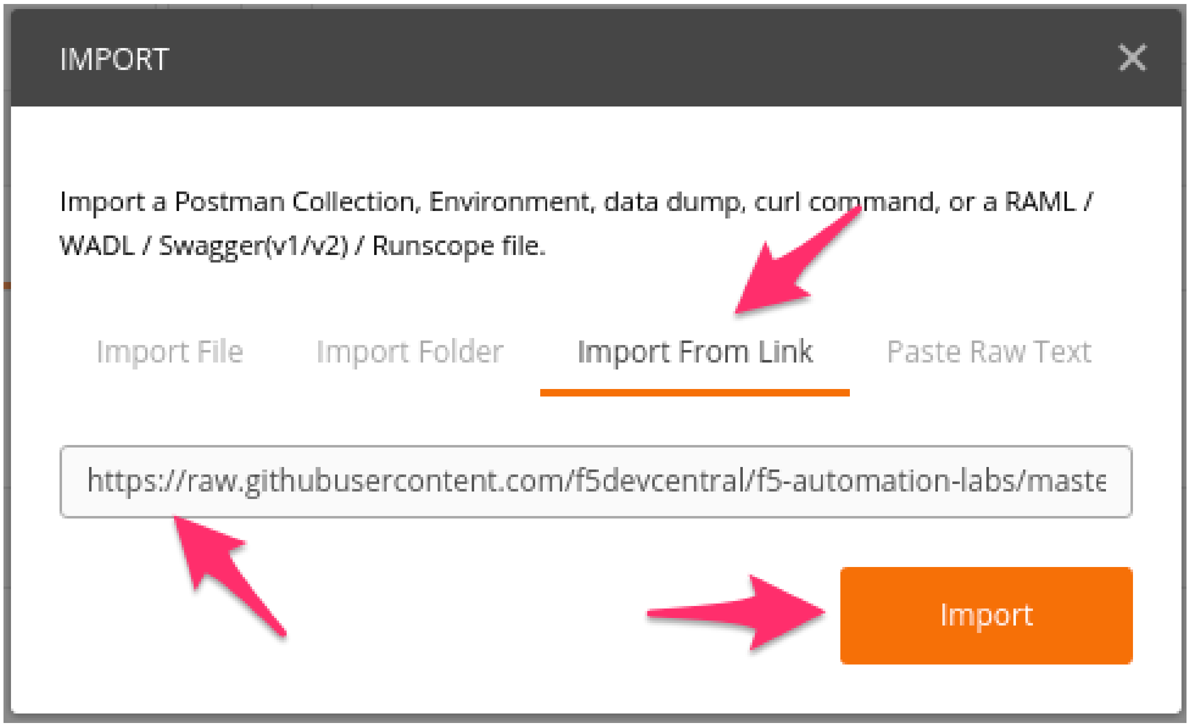
- Click the **X** in the top right of the Settings window

3. A Postman Collection lets you group individual REST requests together. This Postman collection can then be shared and imported. To import a Postman Collection, click the *Import* button in the top left of the Postman window



4. Click the *Import from Link* tab. Paste the following URL into the text box and click *Import*

```
https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/postman_collections/Class_1.postman_collection.json
```

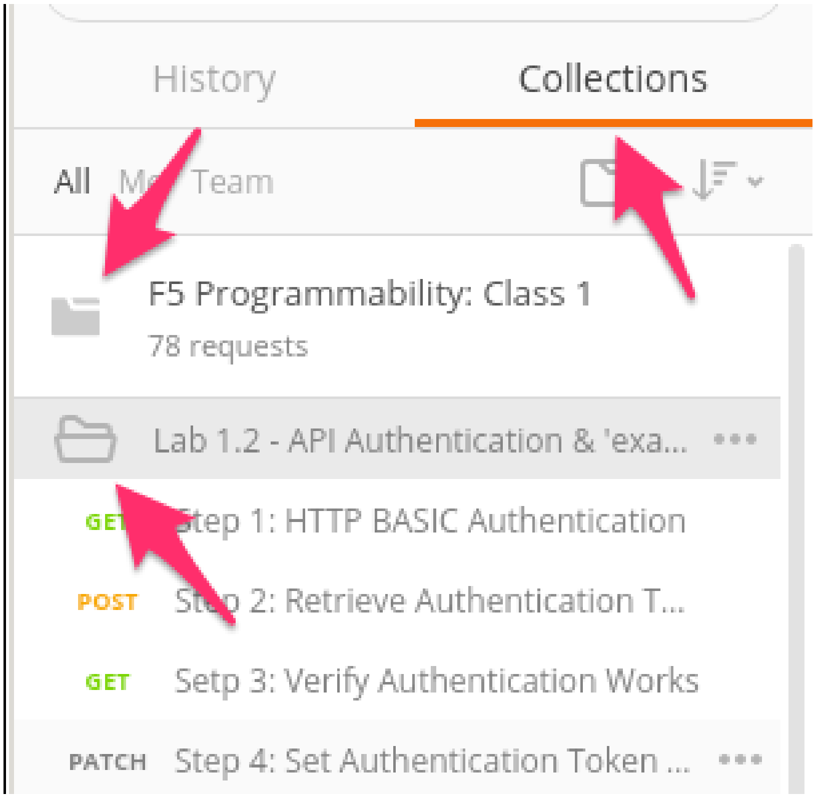


IMPORT

Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.

https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/maste

5. You should now see a collection named `F5 Programmability:  Class 1` in your Postman Collections sidebar. Postman automatically resizes its GUI depending on its window size. It might be necessary to use the short `Ctrl + \` (on Windows) or click the show sidebar icon at the bottom left corner of postman if you do not see the sidebar.



F5 Programmability: Class 1
78 requests

Lab 1.2 - API Authentication & 'exa...

GET Step 1: HTTP BASIC Authentication

POST Step 2: Retrieve Authentication T...

GET Setp 3: Verify Authentication Works

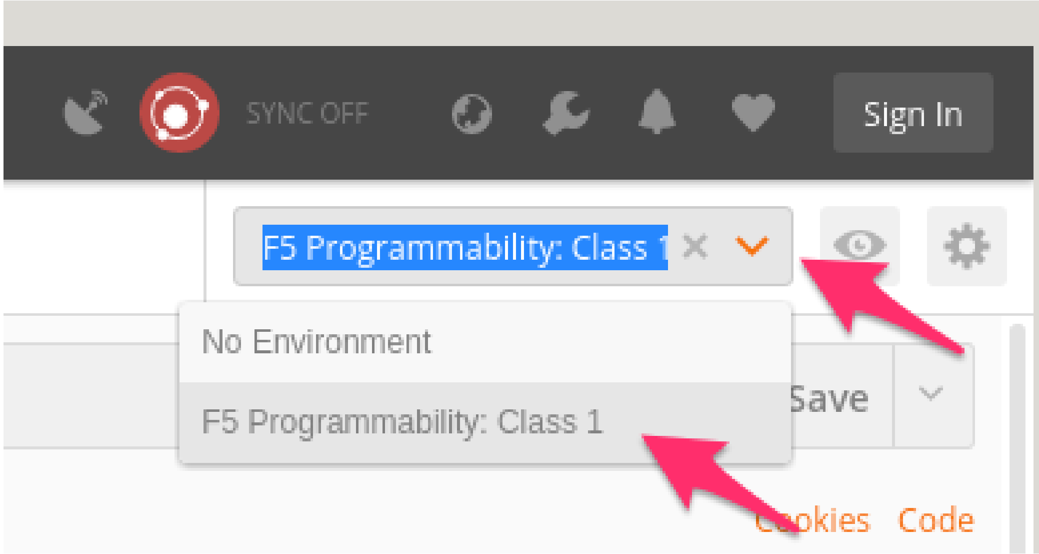
PATCH Step 4: Set Authentication Token ...

6. To assist in multi-step procedures we make heavy use of the **Environments** capability in Postman.

This capability allows us to set various global variables that are then substituted into a request before it's sent. Import the Environment file by clicking *Import* → *Import from Link* and pasting the following URL and clicking *Import*:

```
https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/
postman_collections/Class_1.postman_environment.json
```

7. Set your environment to `F5 Programmability: Class 1` by using the menu at the top right of your Postman window:



Task 2 - HTTP BASIC Authentication

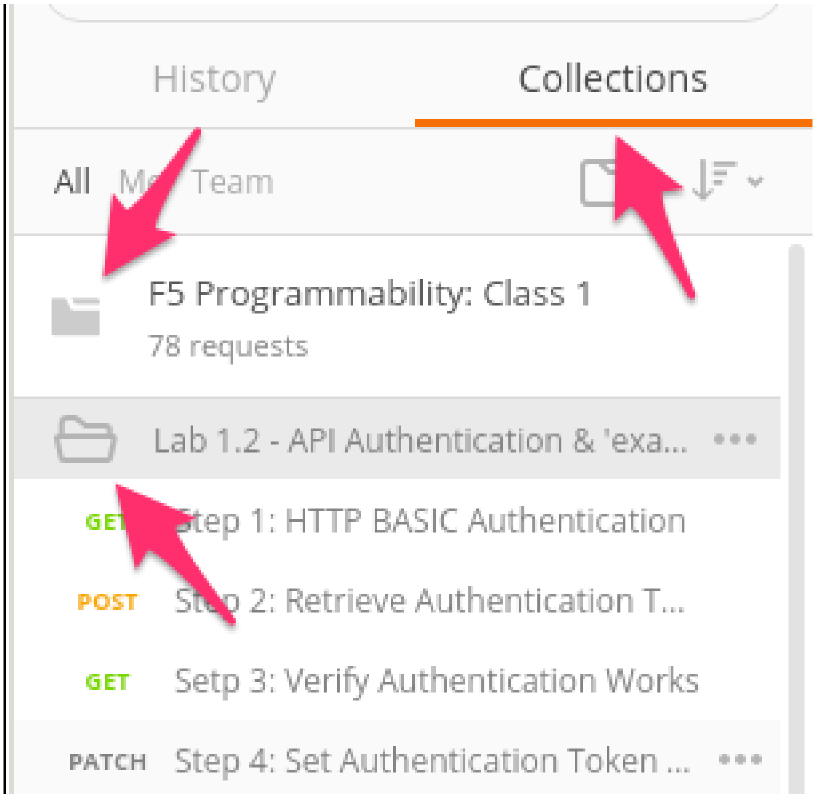
In this task, we will use the Postman client to send API requests using HTTP BASIC authentication. As its name implies this method of authentication encodes the user credentials via the existing BASIC authentication method provided by the HTTP protocol. The mechanism this method uses is to insert an HTTP header named 'Authorization' with a value that is built by Base 64 encoding the string `<username>:<password>`. The resulting header takes this form:

```
Authorization: Basic YWRtaW46YWRtaW4=
```

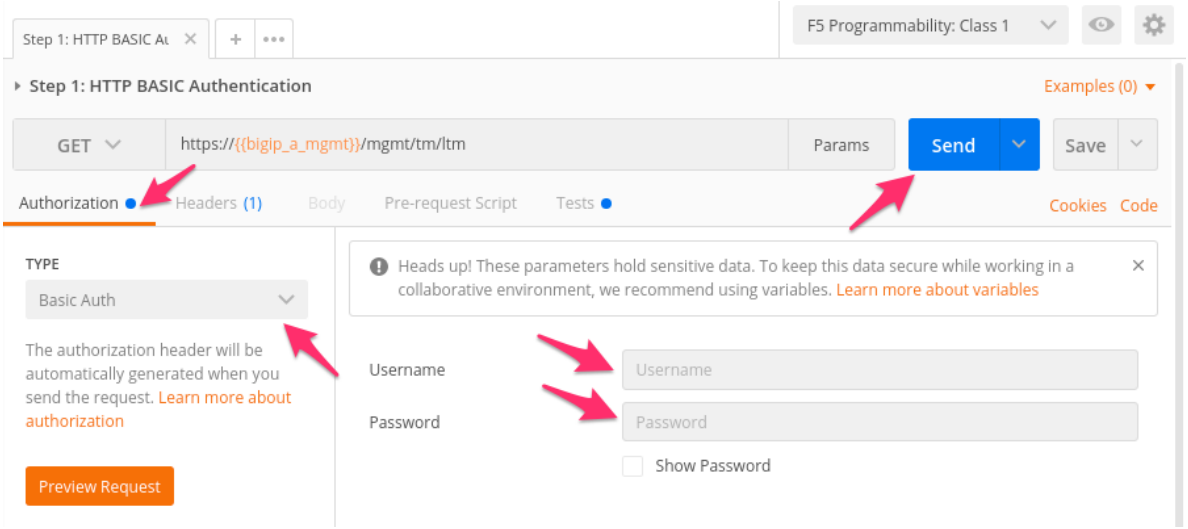
It should be noted that cracking the method of authentication is TRIVIAL; as a result API calls should always be performed using HTTPS encryption (F5 default) with a certificate signed by an authority rather than HTTP.

Perform the following steps to complete this task:

1. Click the *Collections* tab on the left side of the screen, expand the `F5 Programmability: Class 1` collection on the left side of the screen, expand the `Lab 1.2 - API Authentication & 'example' Templates` folder:


History
Collections
All
My Team
F5 Programmability: Class 1
78 requests
Lab 1.2 - API Authentication & 'exa... •••
GET Step 1: HTTP BASIC Authentication
POST Step 2: Retrieve Authentication T...
GET Setp 3: Verify Authentication Works
PATCH Step 4: Set Authentication Token ... •••

2. Click the `Step 1:  HTTP BASIC Authentication` item. Click the *Authorization* tab and select `Basic Auth` as the Type. Fill in the username and password (`admin/admin`) and click the *Send* button:



https://{{bigip_a_mgmt}}/mgmt/tm/ltm

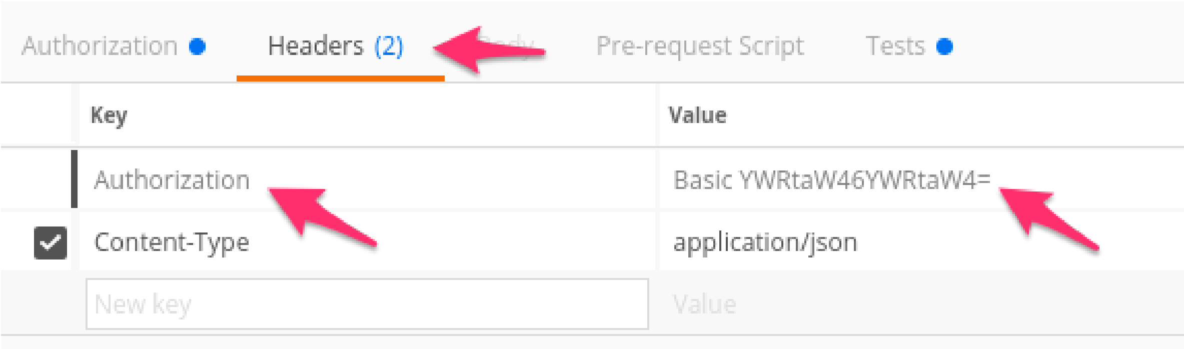
TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

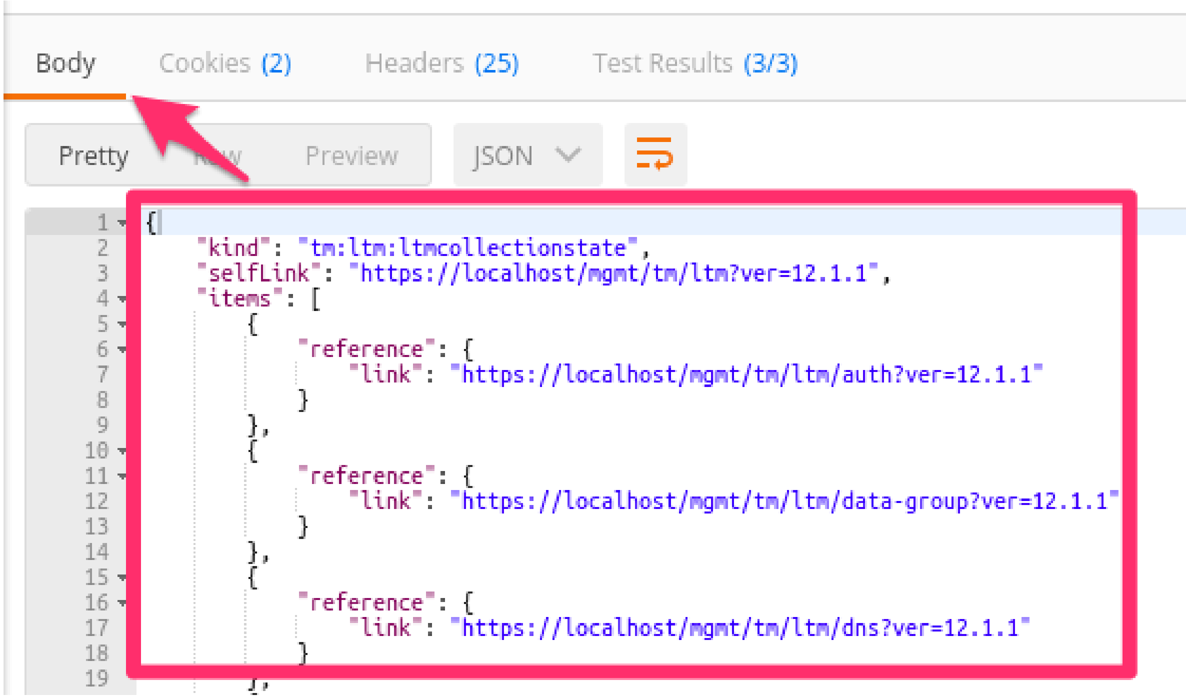
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

3. Click the *Headers* tab and examine the HTTP header. Notice that the number of Headers in the Headers tab changed from `1` to `2`. This is because Postman automatically created the HTTP header and updated your request to include it.



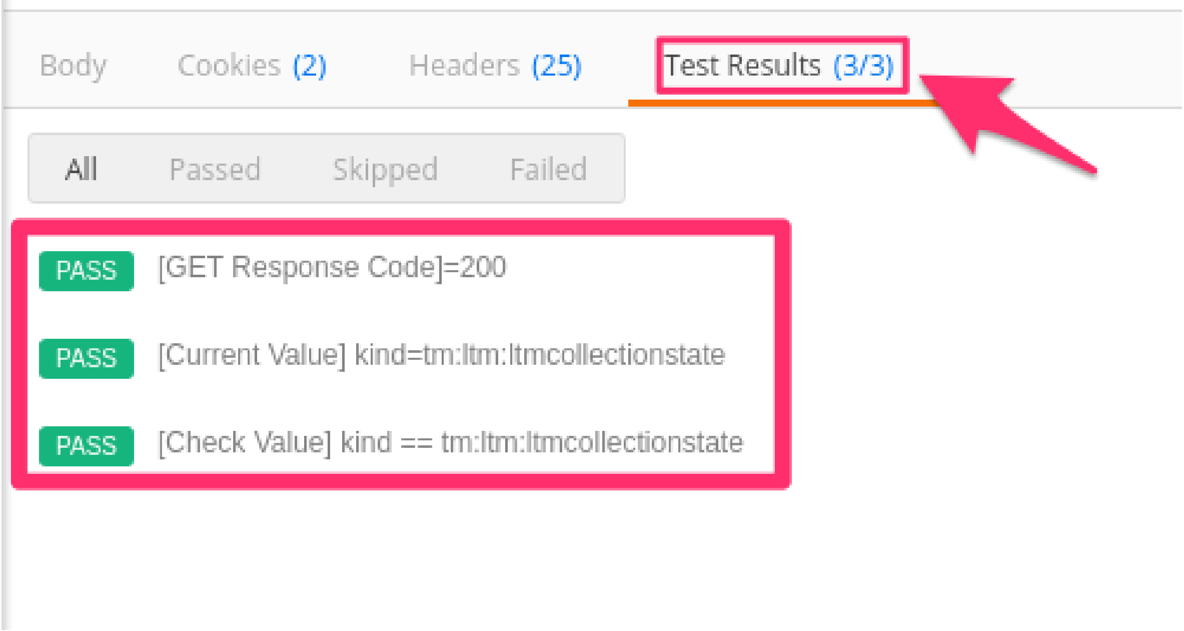
	Key	Value
	Authorization	Basic YWRtaW46YWRtaW4=
☑	Content-Type	application/json
	New key	Value

4. Click the *Body* tab, if the request succeeded you should be presented with a listing of the `/mgmt/tm/ltm` Organizing Collection:



```
{
    "kind": "tm:ltm:ltmcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/ltm?ver=12.1.1",
    "items": [
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/ltm/auth?ver=12.1.1"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/ltm/data-group?ver=12.1.1"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/ltm/dns?ver=12.1.1"
            }
        },
```

5. Click the *Test Results* tab and ensure all the tests for this request have passed:

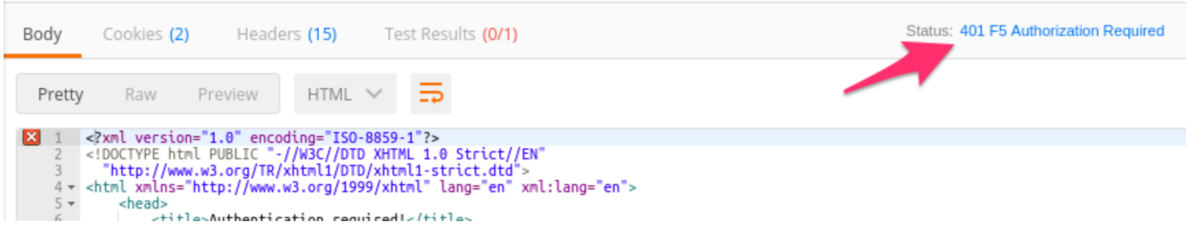


PASS [GET Response Code]=200

PASS [Current Value] kind=tm:ltm:ltmcollectionstate

PASS [Check Value] kind == tm:ltm:ltmcollectionstate

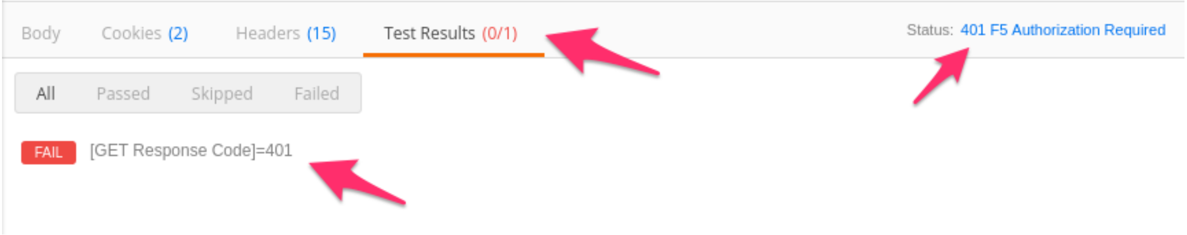
6. Update the credentials and specify an INCORRECT password. Send the request again and examine the response:



Status: 401 F5 Authorization Required

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
    <head>
        <title>Authentication required!</title>
```

7. Check the *Test Results* tab and notice that our *Unit Tests* for this request are now failing (as expected):



Status: 401 F5 Authorization Required

FAIL [GET Response Code]=401

Important: As you progress through this lab be sure to check the *Test Results* tab. We have included *Unit Tests* where applicable to help you verify the requests being sent are succeeding. If you notice a test has failed please double check your input or ask for help.

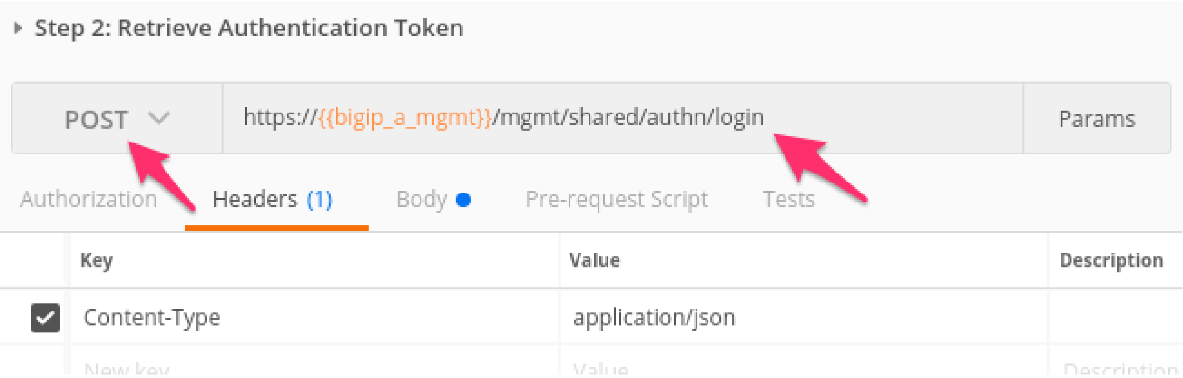
Task 3 - Token Based Authentication

One of the disadvantages of BASIC Authentication is that credentials are sent with each and every request. This can result in a much greater attack surface being exposed unnecessarily. As a result, **Token Based Authentication (TBA)** is preferred in many cases. TBA only sends the credentials once, on the first request. The system then responds with a unique token for that session and the consumer then uses that token for all subsequent requests. BIG-IP and iWorkflow support token-based authentication that drops down to the underlying authentication subsystems available in TMOS. As a result, the system can be configured to support external authentication providers (Active Directory, RADIUS, TACACS, etc) and those authentication methods can flow through to the REST API. In this task we will demonstrate TBA using the local authentication database, however, authentication to external providers is fully supported.

Note: For more information about external authentication providers see the section titled **About external authentication providers with iControl REST** in the iControl REST API User Guide available at https://devcentral.f5.com

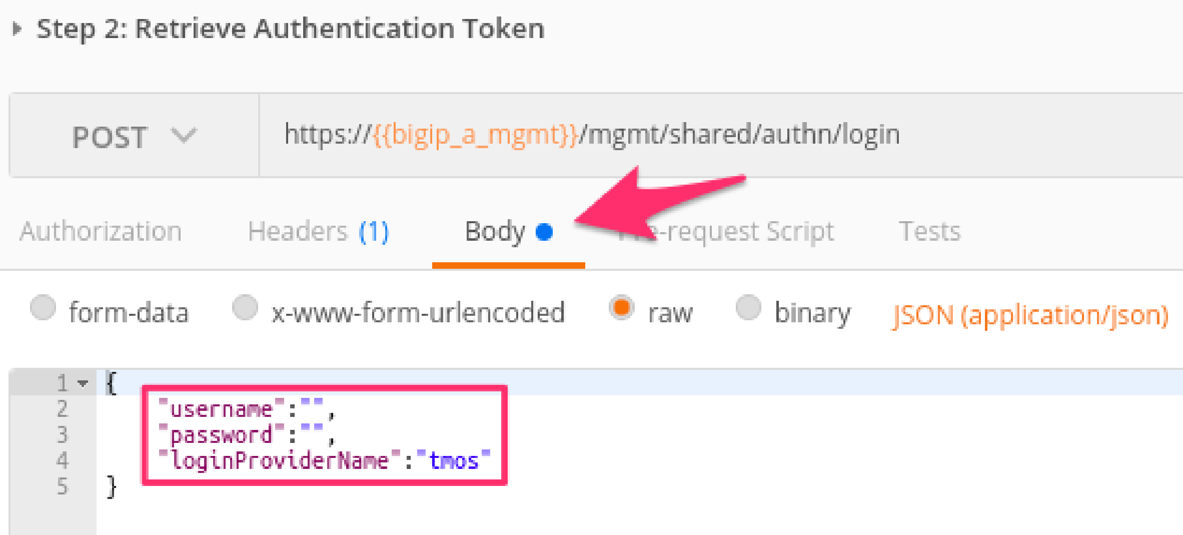
Perform the following steps to complete this task:

1. Click the `Step 2: Retrieve Authentication Token` item in the Lab 1.2 Folder
2. Notice that we send a `POST` request to the `/mgmt/shared/authn/login` endpoint.


Step 2: Retrieve Authentication Token
POST
https://{{bigip_a_mgmt}}/mgmt/shared/authn/login
Params
Authorization
Headers (1)
Body
Pre-request Script
Tests

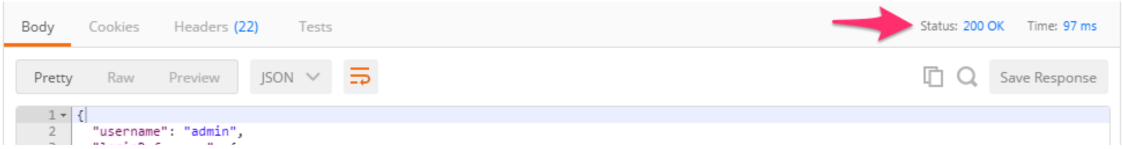
	Key	Value	Description
☑	Content-Type	application/json	
	New key	Value	Description

3. Click the *Body* tab and examine the JSON that we will send to BIG-IP to provide credentials and the authentication provider:



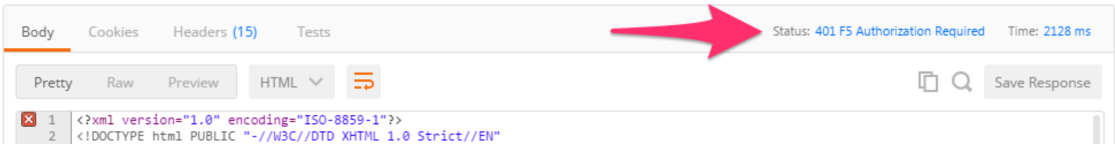
```
{
    "username":"",
    "password":"",
    "loginProviderName":"tmos"
}
```

4. Modify the JSON body and add the required credentials (`admin/admin`). Then click the *Send* button.
5. Examine the response status code. If authentication succeeded and a token was generated the response will have a `200 OK` status code. If the status code is `401` then check your credentials:
    - **Successful:**



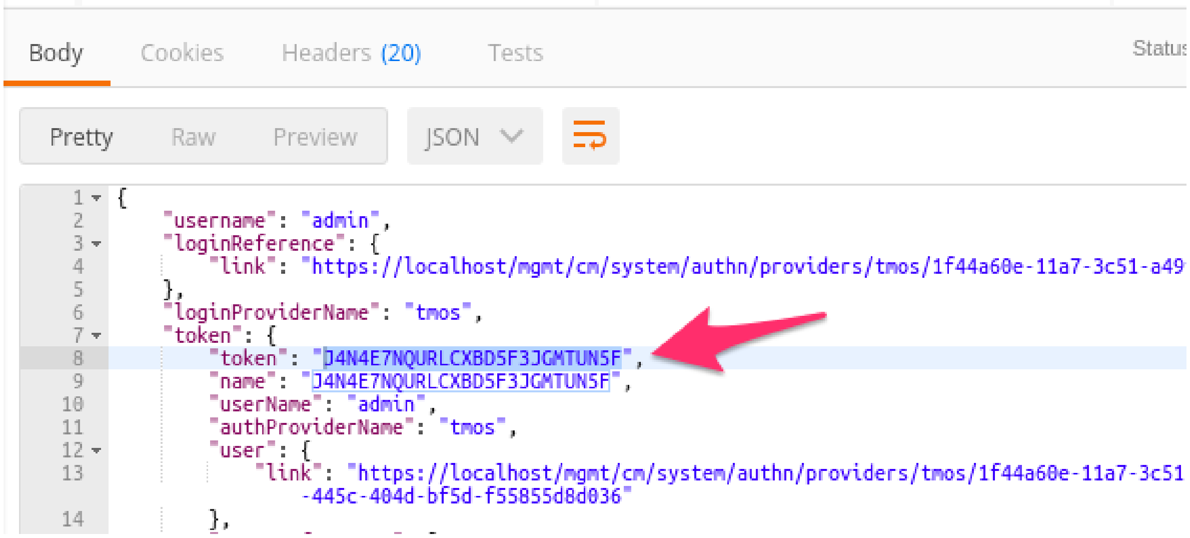
```
{
  "username": "admin",
```

- **Unsuccessful:**



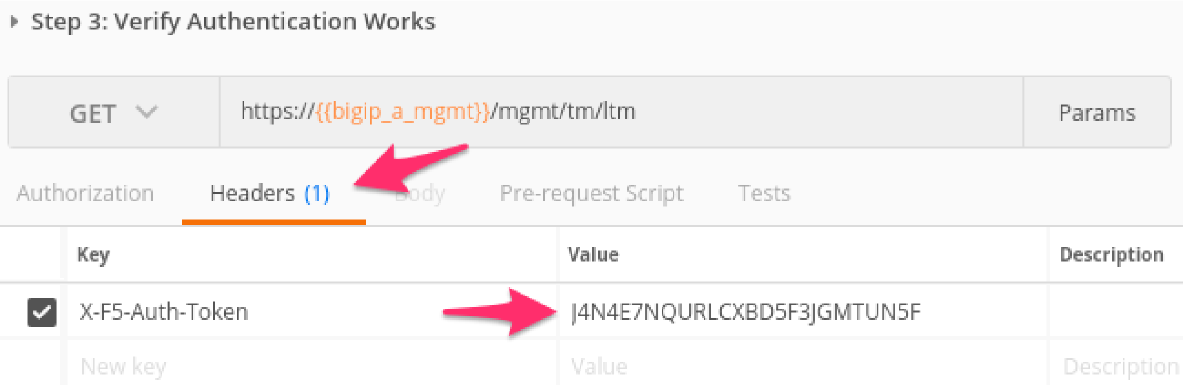
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

6. Once you receive a `200 OK` status code examine the response body. The various attributes show the parameters assigned to the particular token. Find the `token` attribute and copy it into your clipboard (`Ctrl+c`) for use in the next step.



```
{
    "username": "admin",
    "loginReference": {
        "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49
    },
    "loginProviderName": "tmos",
    "token": {
        "token": "J4N4E7NQURLCXBD5F3JGMTUN5F",
        "name": "J4N4E7NQURLCXBD5F3JGMTUN5F",
        "userName": "admin",
        "authProviderName": "tmos",
        "user": {
            "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51
-445c-404d-bf5d-f55855d8d036"
        },
```

7. Click the `Step 3: Verify Authentication Works` item in the Lab 1.2 Postman collection. Click the *Headers* tab and paste the token value copied above as the VALUE for the `X-F5-Auth-Token` header. This header is required to be sent on all requests when using token-based authentication.

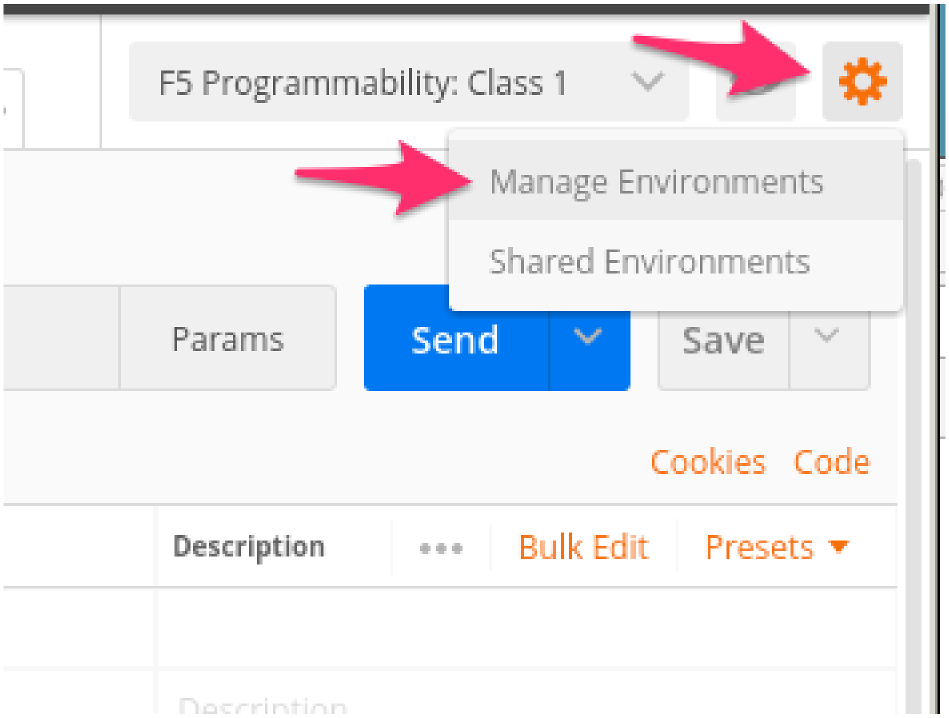


GET https://{{bigip_a_mgmt}}/mgmt/tm/ltm

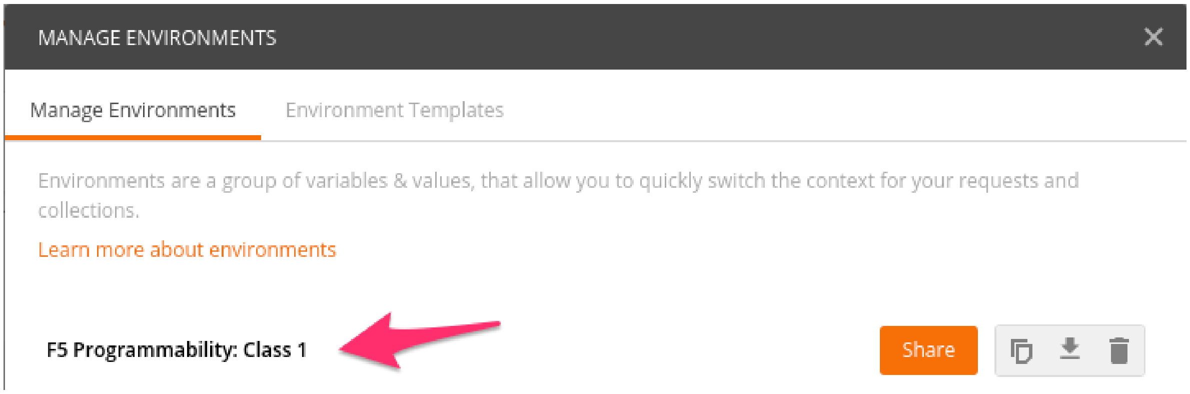
	Key	Value	Description
☑	X-F5-Auth-Token	J4N4E7NQURLCXBD5F3JGMTUN5F	

8. Click the *Send* button. If your request is successful you should see a `200 OK` status and a listing of the `ltm` Organizing Collection.

9. We will now update your Postman environment to use this auth token for the remainder of the lab. Click the Environment menu in the top right of the Postman window and click *Manage Environments*:



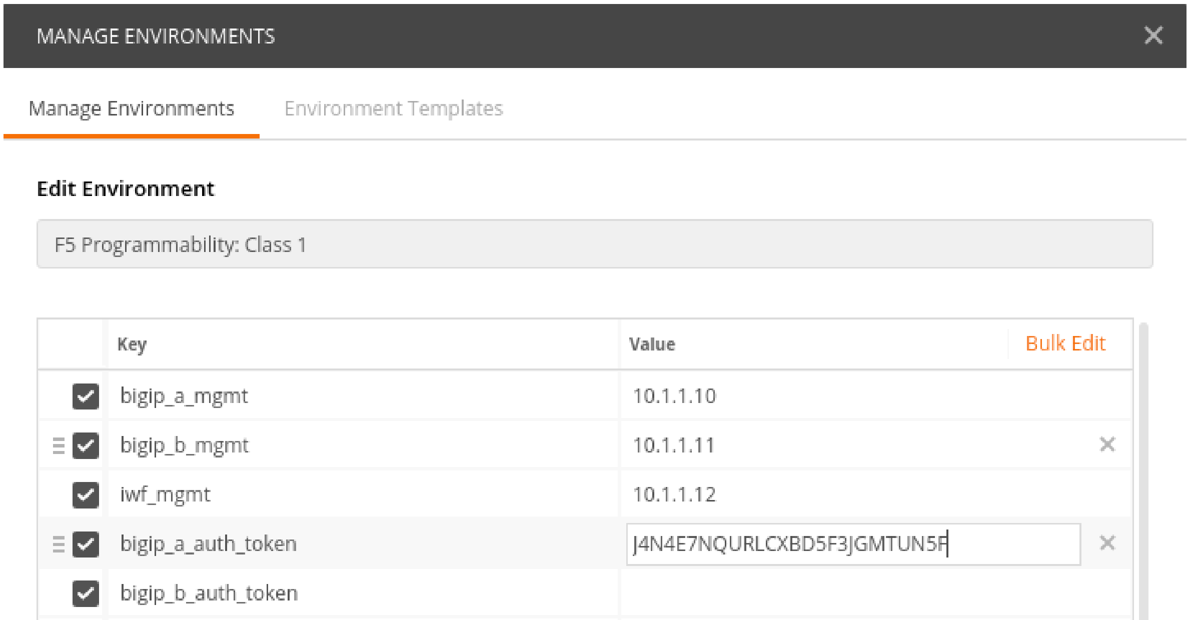
10. Click the `F5 Programmability: Class 1` item:



Environments are a group of variables & values, that allow you to quickly switch the context for your requests and collections.

F5 Programmability: Class 1

11. Update the value for `bigip_a_auth_token` by Pasting (`Ctrl+v`) in your auth token:



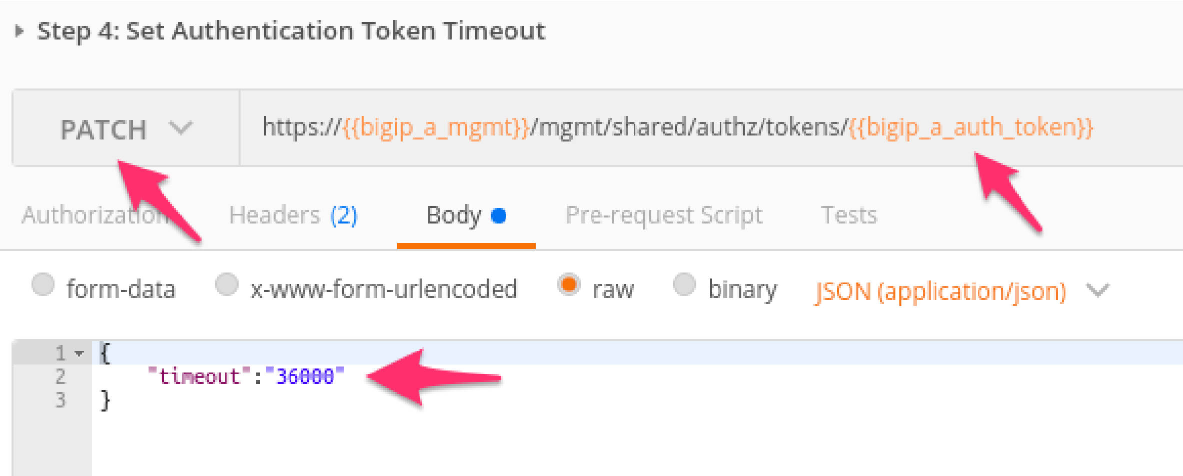
Edit Environment

F5 Programmability: Class 1

	Key	Value
☑	bigip_a_mgmt	10.1.1.10
☑	bigip_b_mgmt	10.1.1.11
☑	iwf_mgmt	10.1.1.12
☑	bigip_a_auth_token	J4N4E7NQURLCXBD5F3JGMTUN5F
☑	bigip_b_auth_token	

12. Click the `Update` button and then close the *Manage Environments* window. Because the subsequent requests refer to the `{{bigip_a_auth_token}}` variable, you will not have to set the token in the header of the following requests.

13. Click the `Step 4: Set Authentication Token Timeout` item in the Lab 1.2 Postman folder. This request will `PATCH` your token Resource (notice the URI) and update the timeout attribute so we can complete the lab easily. Examine the request type and JSON *Body* and then click the *Send* button. Verify that the timeout has been changed to `36000` in the response:



PATCH https://{{bigip_a_mgmt}}/mgmt/shared/authz/tokens/{{bigip_a_auth_token}}

● raw JSON (application/json)

```
{
    "timeout":"36000"
}
```

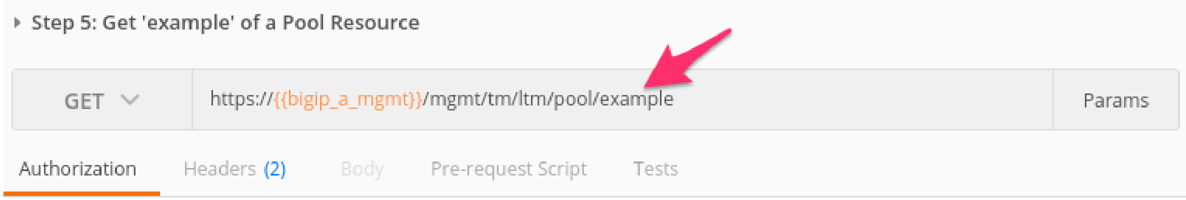
Task 4 - Get a pool 'example' Template

In order to assist with REST API interactions, you can request a template of the various attributes of a Resource type in a Collection. This template can then be used as the body of a `POST`, `PUT` or `PATCH` request as needed.

Perform the following steps:

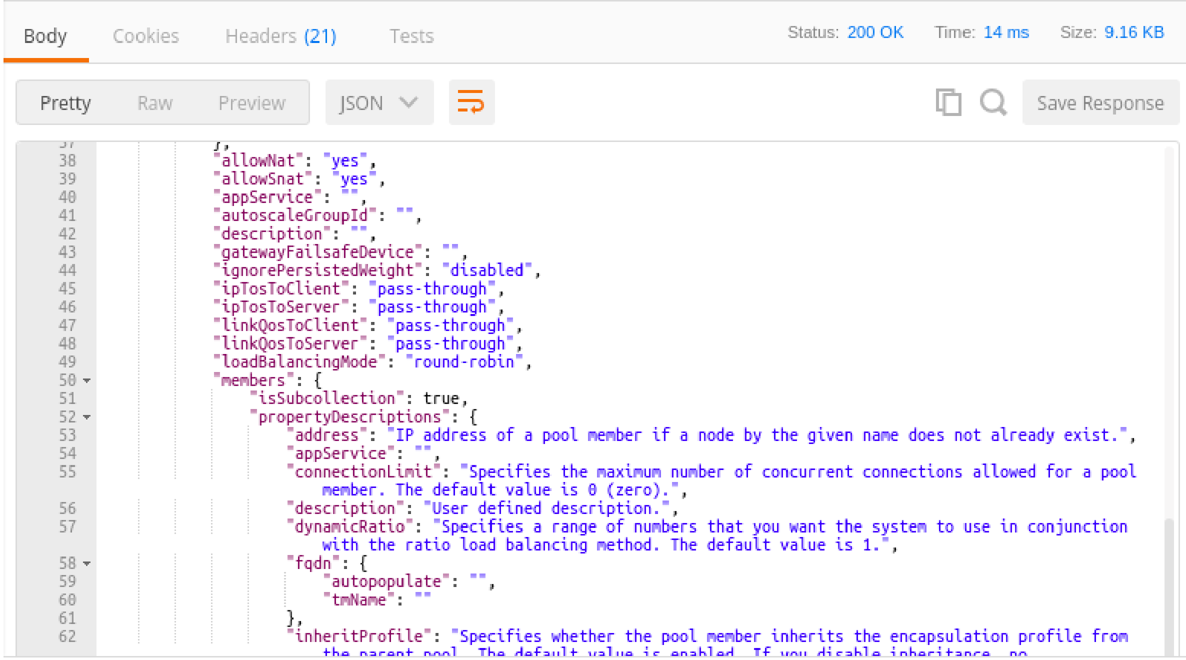
1. Click the `Step 5: Get 'example' of a Pool Resource` item in the Lab 1.2 Postman collection

2. Examine the URI. Notice the addition of `example` at the end of the collection name:



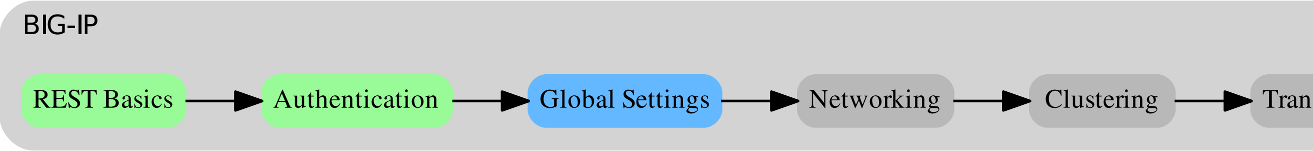
```
https://{{bigip_a_mgmt}}/mgmt/tm/ltm/pool/example
```

3. Click *Send* and examine the FULL response. You will see descriptions and then all the attributes for the *Pool* resource type. The response also shows the default values for the attributes if applicable:



```
        },
        "allowNat": "yes",
        "allowSnat": "yes",
        "appService": "",
        "autoscaleGroupId": "",
        "description": "",
        "gatewayFailsafeDevice": "",
        "ignorePersistedWeight": "disabled",
        "ipTosToClient": "pass-through",
        "ipTosToServer": "pass-through",
        "linkQosToClient": "pass-through",
        "linkQosToServer": "pass-through",
        "loadBalancingMode": "round-robin",
        "members": {
            "isSubcollection": true,
            "propertyDescriptions": {
                "address": "IP address of a pool member if a node by the given name does not already exist.",
                "appService": "",
                "connectionLimit": "Specifies the maximum number of concurrent connections allowed for a pool
                    member. The default value is 0 (zero).",
                "description": "User defined description.",
                "dynamicRatio": "Specifies a range of numbers that you want the system to use in conjunction
                    with the ratio load balancing method. The default value is 1.",
                "fqdn": {
                    "autopopulate": "",
                    "tmName": ""
                },
                "inheritProfile": "Specifies whether the pool member inherits the encapsulation profile from
                    the parent pool. The default value is enabled. If you disable inheritance, no
```

2.3.3 Lab 1.3: Review/Set Device Settings



All devices are already licensed so we can focus on configuring the basic infrastructure related settings to complete the Device Onboarding process. The remaining items include (list not exhaustive):

- Device Settings

  - **NTP/DNS Settings**
  - Remote Authentication
  - **Hostname**
  - **Admin Credentials**
- L1-3 Networking
  - Physical Interface Settings
  - L2 Connectivity (**VLAN**, VXLAN, etc.)
  - L3 Connectivity (**Self IPs, Routing**, etc.)
- HA Settings
  - **Global Settings**
    - **Config Sync IP**
    - **Mirroring IP**
    - **Failover Addresses**
  - **CMI Device Trusts**
  - **Device Groups**
  - **Traffic Groups**
  - **Floating Self IPs**

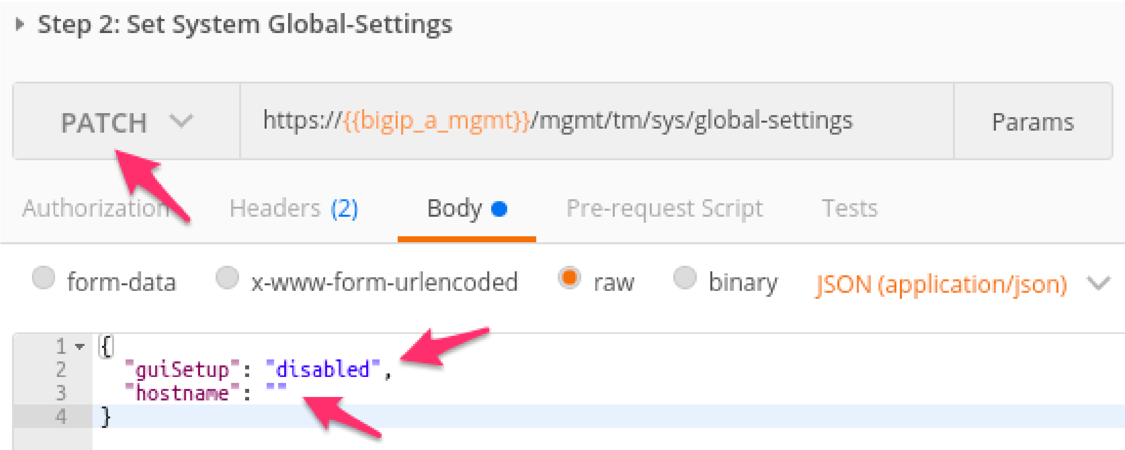
We will specifically cover the items in **BOLD** above in the following labs. It should be noted that many permutations of the Device Onboarding process exist due to the nature of real-world environments. This class is designed to teach enough information so that you can then apply the knowledge learned and help articulate and/or deliver a specific solution for your environment.

Task 1 - Set Device Hostname & Disable GUI Setup Wizard

In this task we will modify the device hostname and disable the GUI Setup Wizard. The Resource that contains these settings is `/mgmt/tm/sys/global-settings`.

Perform the following steps to complete this task:

1. Expand the `Lab 1.3 - Review/Set Device Settings` folder in the Postman collection
2. Click the `Step 1:  Get System Global-Settings` request. Click the *Send* button and review the response body to see what the current settings on the device are. Examine the resulting response to understand what settings are currently applied.
3. Click the `Step 2:  Set System Global-Settings` request. This item uses a `PATCH` request to the `global-settings` resource to modify the attributes contained within it. We will update the `guiSetup` and `hostname` attribute.
   - Click on body. Review the JSON body and modify the `hostname` attribute to set the hostname to `bigip-a.f5.local`
   - Also notice that we are disabling the GUI Setup Wizard as part of the same request:



https://{{bigip_a_mgmt}}/mgmt/tm/sys/global-settings

```
{
  "guiSetup": "disabled",
  "hostname": ""
}
```

4. Click the *Send* button and review the response body. You should see that the attributes modified above have been updated by looking at the response. You can also `GET` the `global-settings` by sending the `Step 1: Get System Global-Settings` request again to verify they have been updated.

Task 2 - Modify DNS/NTP Settings

Note: This task will make use of JSON arrays. The syntax for defining a JSON array is:

`myArray: [ Object0, Object1 ... ObjectX ]`

To define an array consisting of Strings the syntax is:

`myStringArray: [ "string0", "string1" ... "stringX" ]`

Much like the previous task we can update system DNS and NTP settings by sending a PATCH request to the correct resource in the `sys` Organizing Collection. The relevant Resources for this task are:

URL	Type
`/mgmt/tm/sys/dns`	DNS Settings
`/mgmt/tm/sys/ntp`	NTP Settings

Perform the following steps to complete this task:

1. Click the `Step 3: Get System DNS Settings` item in the folder. Click *Send* and review the current settings
2. Click the `Step 4: Set System DNS Settings` item in the folder. Click body. Review the JSON body to verify the name server IPs `4.2.2.2` and `8.8.8.8` are listed. Additionally, add a search domain of `f5.local`. You will modify a JSON array to add a search domain.
3. Click the *Send* button and verify the requested changes were successfully implemented by looking at the response or by sending the `Step 3: Get System DNS Settings` request again.
4. Click the `Step 5: Get System NTP Settings` item in the folder. Click *Send* and review the current settings
5. Click the `Step 6: Set System NTP Settings` item in the folder. Click *Body*. Review the JSON body to verify the NTP servers with hostnames `0.pool.ntp.org` and `1.pool.ntp.org` are contained in the `servers` attribute (another JSON array!).

6. Click the *Send* button and verify the requested changes were successfully implemented by looking at the response or sending the `Step 5: Get System NTP Settings` again.

Task 3 - Update default user account passwords

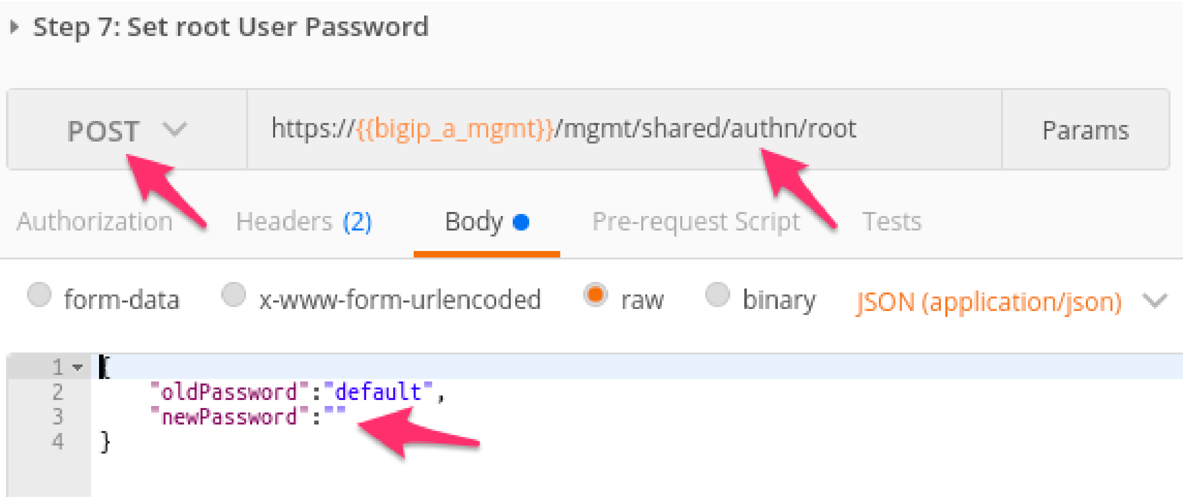
In this task we will update the passwords for the `root` and `admin` accounts. The process for updating the root account is different than other system accounts because it is used by underlying Linux OS.

To update the root account password we will use a `POST` to the `/mgmt/shared/authn/root` REST endpoint.

To update all other system accounts we will `PATCH` the `/mgmt/tm/auth/user/<username>` Resource

Perform the following steps to change the `root` user password:

1. Click the `Step 7: Set root User Password` item in the folder.
2. We are performing a POST operation to change the root user password and have to specify the `oldPassword` because the REST implementation on the BIG-IP uses the underlying Linux mechanism. Click *Body*. Modify the JSON body to update the password to the value `newdefault` and click the *Send* button.

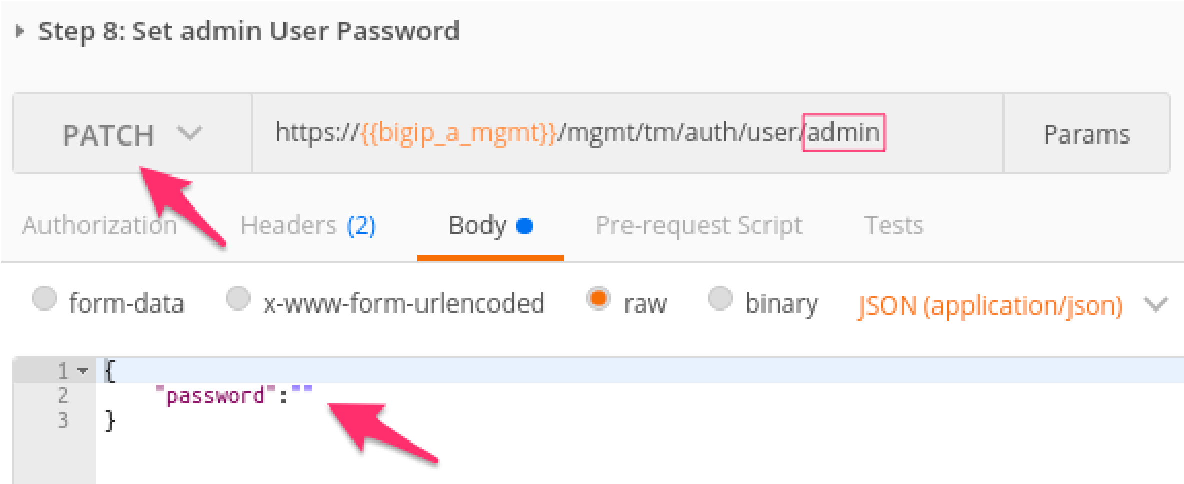


```
{
    "oldPassword":"default",
    "newPassword":""
}
```

3. You can verify the password was changed by opening an SSH session to BIG-IP-A. A shortcut to a terminal is included on the desktop of the Linux jumphost. To open an SSH connection to BIG-IP A open a terminal window and execute `ssh root@10.1.1.10`
4. **Repeat the procedure above to change the password back to** `default`

Perform the following steps to change the **admin** user password:

1. Click the `Step 8: Set admin User Password` item in the collection.
2. We are performing a `PATCH` operation to admin user Resource. Click *Body* and modify the JSON body to update the password to the value `newadmin` and click the *Send* button.



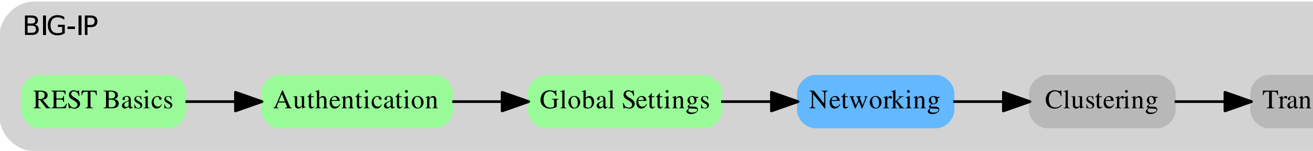
Step 8: Set admin User Password

PATCH https://{{bigip_a_mgmt}}/mgmt/tm/auth/user/admin

```
{
    "password":""
}
```

3. **You can verify the password was changed by opening an SSH session** OR by logging into TMUI (HTTP GUI) to BIG-IP-A in a Chrome browser tab.
4. **Repeat the procedure above to change the password back to** `admin`

2.3.4 Lab 1.4: Basic Network Connectivity



This lab will focus on configuration of the following items:

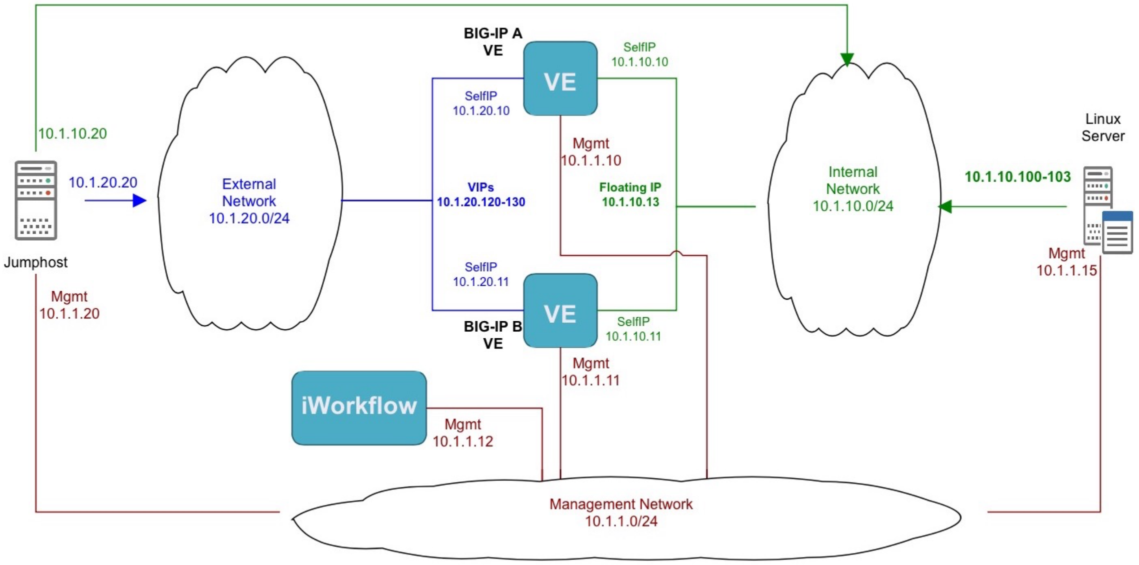
- L1-3 Networking
  - Physical Interface Settings
  - L2 Connectivity (**VLAN**, VXLAN, etc.)
  - L3 Connectivity (**Self IPs, Routing**, etc.)

We will specifically cover the items in **BOLD** above in the following labs. It should be noted that many permutations of the Device Onboarding process exist due to the nature of different organizations. This class is designed to teach enough information so that you can then apply the knowledge learned and help articulate and/or deliver a specific solution for your environment.

The following table and diagram lists the L2-3 network information used to configure connectivity for BIG-IP-A:

Type	Name	Details
VLAN	Internal	**Interface**: 1.1 **Tag:** 10
VLAN	External	**Interface**: 1.2 **Tag:** 20
Self IP	Self-Internal	**Address**: 10.1.10.10/24 **VLAN:** Internal
Self IP	Self-External	**Address**: 10.1.20.10/24 **VLAN:** External
Route	Default	**Network:** 0.0.0.0/0 **GW:** 10.1.20.1

Lab Topology



Task 1 - Create VLANs

Note: This lab shows how to configure VLAN tags, but does not deploy tagged interfaces. To use tagged interfaces the `tagged` attribute needs to have the value `true`

Perform the following steps to configure the VLAN objects/resources:

1. Expand the `Lab 1.4 - Basic Network Connectivity` folder in the Postman collection.
2. Click the `Step 1: Create a VLAN` request in the folder. Click *Body* and examine the JSON request body; the values for creating the Internal VLAN have already been populated.
3. Click the *Send* button to create the VLAN
4. **Repeat Step 1**, however, this time modify the JSON body to create the External VLAN using the parameters shown in the table above. In order to do so you can replace the following:
   - `name`: `Internal` –> `External`

- `tag`: `10` –> `20`
- `interfaces[] -->` `name`: `1.1` –> `1.2`

5. Click the `Step 2: Get VLANs` request in the folder. Click the *Send* button to `GET` the VLAN collection. Examine the response to make sure both VLANs have been created.

Task 2 - Create Self IPs

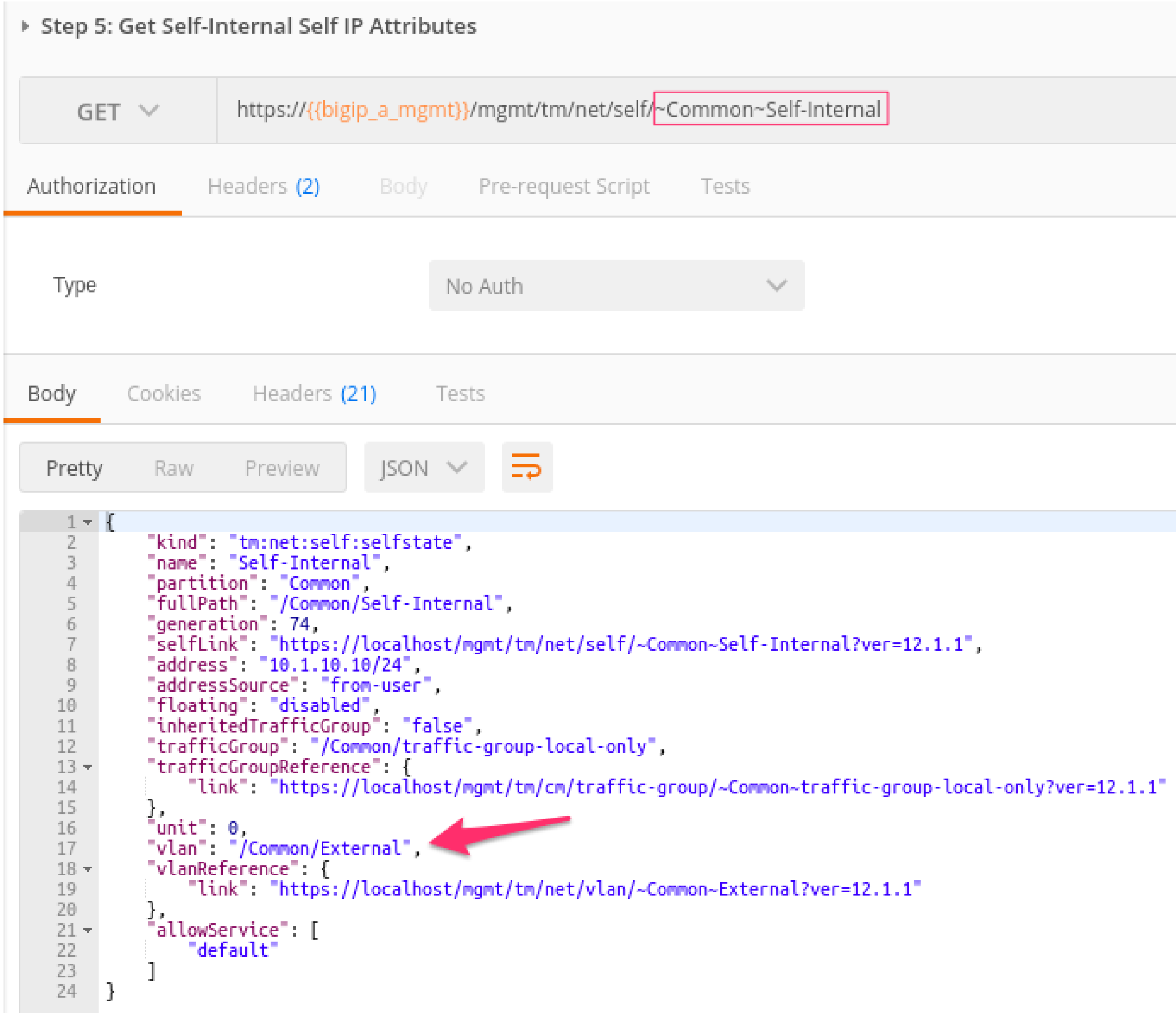
Perform the following steps to configure the Self IP objects/resources:

1. Click the `Step 3: Create Internal Self IP` request in the folder. Examine the JSON body; the values for creating the Self-Internal Self IP have already been populated.

   Note: The JSON body sets the VLAN to `/Common/External` on purpose. You will modify this value in the steps below. Please do not change the value.

2. Click the *Send* button to create the Self IP
3. Click the `Step 4: Create External Self IP` request in the folder and click *Send*
4. Click the `Step 5: Get Self-Internal Self IP Attributes` request in the folder and click the *Send* button. Examine the VLAN settings of the Resource. As noted above the Self IP has been assigned to the wrong VLAN (intentionally)

   Note: Postman has the ability to check the responses for specific values to verify if the result of a request is what it is expected to be. The :guilabel:`Test Results` for this request will show a failure for the `[Check Value] vlan == /Common/Internal` value. This is intentional and you should continue to the next section.



Step 5: Get Self-Internal Self IP Attributes

GET https://{{bigip_a_mgmt}}/mgmt/tm/net/self/~Common~Self-Internal

Type: No Auth

```
{
    "kind": "tm:net:self:selfstate",
    "name": "Self-Internal",
    "partition": "Common",
    "fullPath": "/Common/Self-Internal",
    "generation": 74,
    "selfLink": "https://localhost/mgmt/tm/net/self/~Common~Self-Internal?ver=12.1.1",
    "address": "10.1.10.10/24",
    "addressSource": "from-user",
    "floating": "disabled",
    "inheritedTrafficGroup": "false",
    "trafficGroup": "/Common/traffic-group-local-only",
    "trafficGroupReference": {
        "link": "https://localhost/mgmt/tm/cm/traffic-group/~Common~traffic-group-local-only?ver=12.1.1"
    },
    "unit": 0,
    "vlan": "/Common/External",
    "vlanReference": {
        "link": "https://localhost/mgmt/tm/net/vlan/~Common~External?ver=12.1.1"
    },
    "allowService": [
        "default"
    ]
}
```

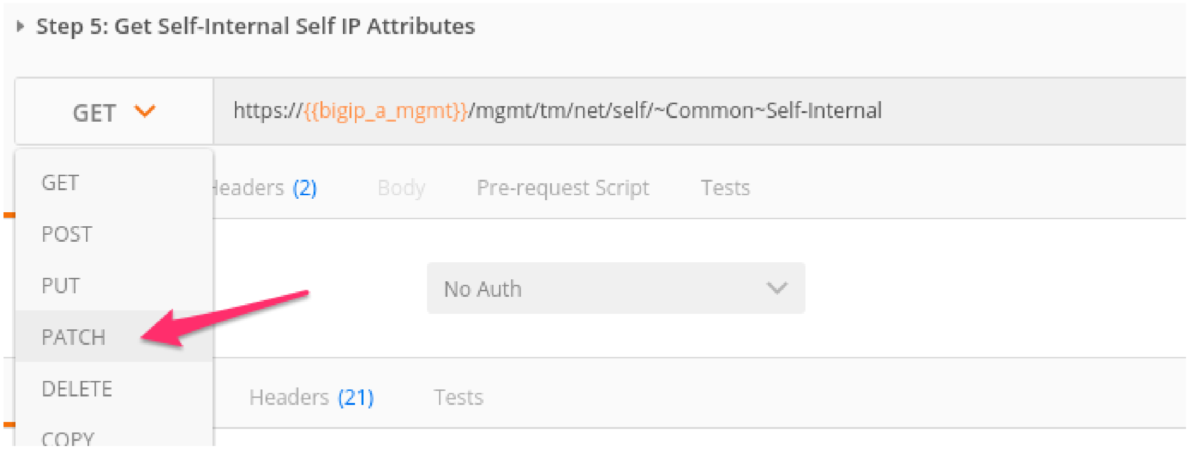
Task 3 - Modify Existing Self IP Resource

In order to modify an existing object via the REST API, the URI path has to be changed. In the previous examples we used a `POST` to create Resources under a Collection, therefore, the URI used was that of the Collection itself. If you wish to update/modify a Resource you must refer to the Resource directly.

For example, the Collection URI for Self IPs is `/mgmt/tm/net/self`.

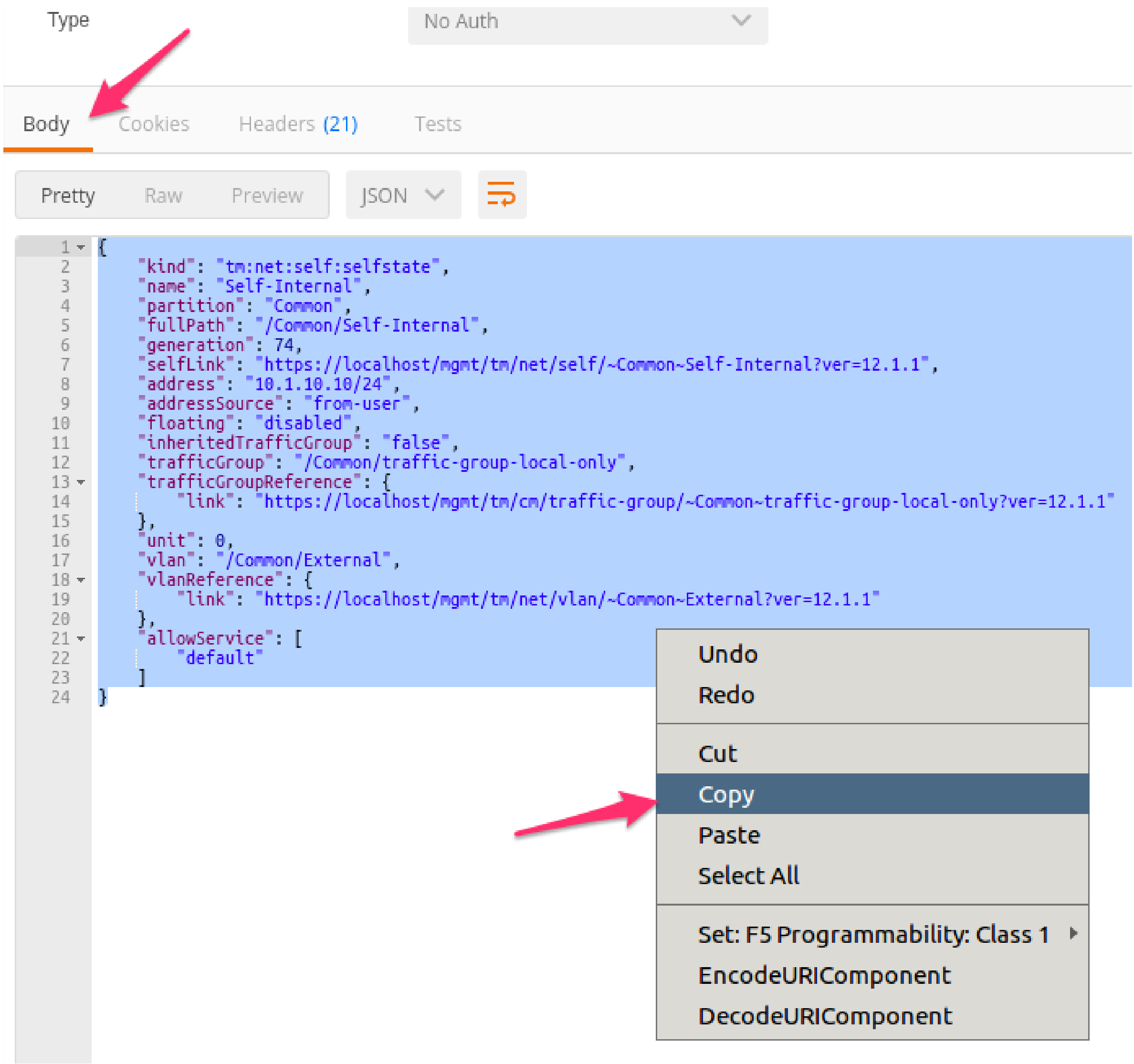
The Resource URI for the `Self-Internal` Self IP is `/mgmt/tm/net/self/~Common~Self-Internal`. Notice that the BIG-IP partition and object name has been added to the Collection URI to for the Resource URI.

1. On the open `Step 5: Get Self-Internal Self IP Attributes` request change the request method from `GET` to `PATCH`. The `PATCH` method is used to modify the attributes of an existing Resource.



https://{{bigip_a_mgmt}}/mgmt/tm/net/self/~Common~Self-Internal

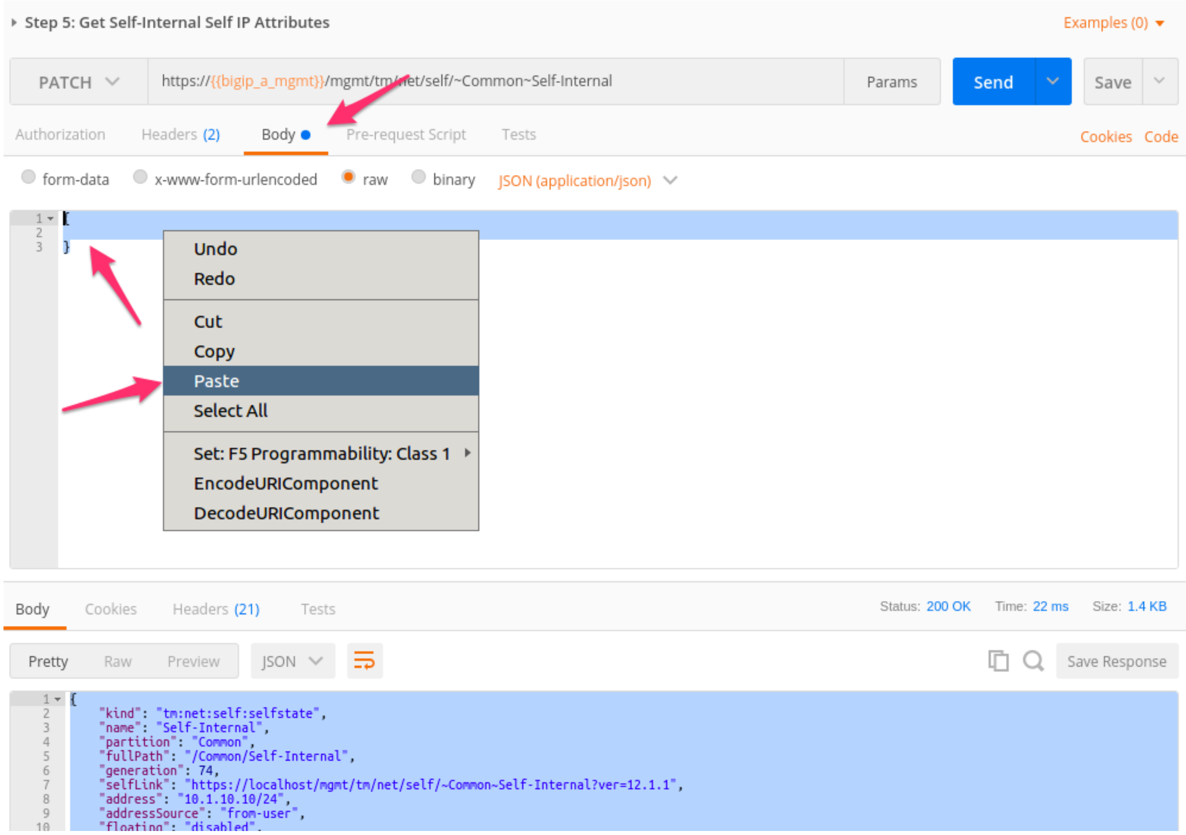
2. Copy the entire JSON **RESPONSE** from the previous `GET` request



```
{
    "kind": "tm:net:self:selfstate",
    "name": "Self-Internal",
    "partition": "Common",
    "fullPath": "/Common/Self-Internal",
    "generation": 74,
    "selfLink": "https://localhost/mgmt/tm/net/self/~Common~Self-Internal?ver=12.1.1",
    "address": "10.1.10.10/24",
    "addressSource": "from-user",
    "floating": "disabled",
    "inheritedTrafficGroup": "false",
    "trafficGroup": "/Common/traffic-group-local-only",
    "trafficGroupReference": {
        "link": "https://localhost/mgmt/tm/cm/traffic-group/~Common~traffic-group-local-only?ver=12.1.1"
    },
    "unit": 0,
    "vlan": "/Common/External",
    "vlanReference": {
        "link": "https://localhost/mgmt/tm/net/vlan/~Common~External?ver=12.1.1"
    },
    "allowService": [
        "default"
    ]
}
```

3. Paste the text into JSON Request body:

Note: Be sure to highlight any existing text and replace it while pasting



Step 5: Get Self-Internal Self IP Attributes

PATCH https://{{bigip_a_mgmt}}/mgmt/tm/net/self/~Common~Self-Internal

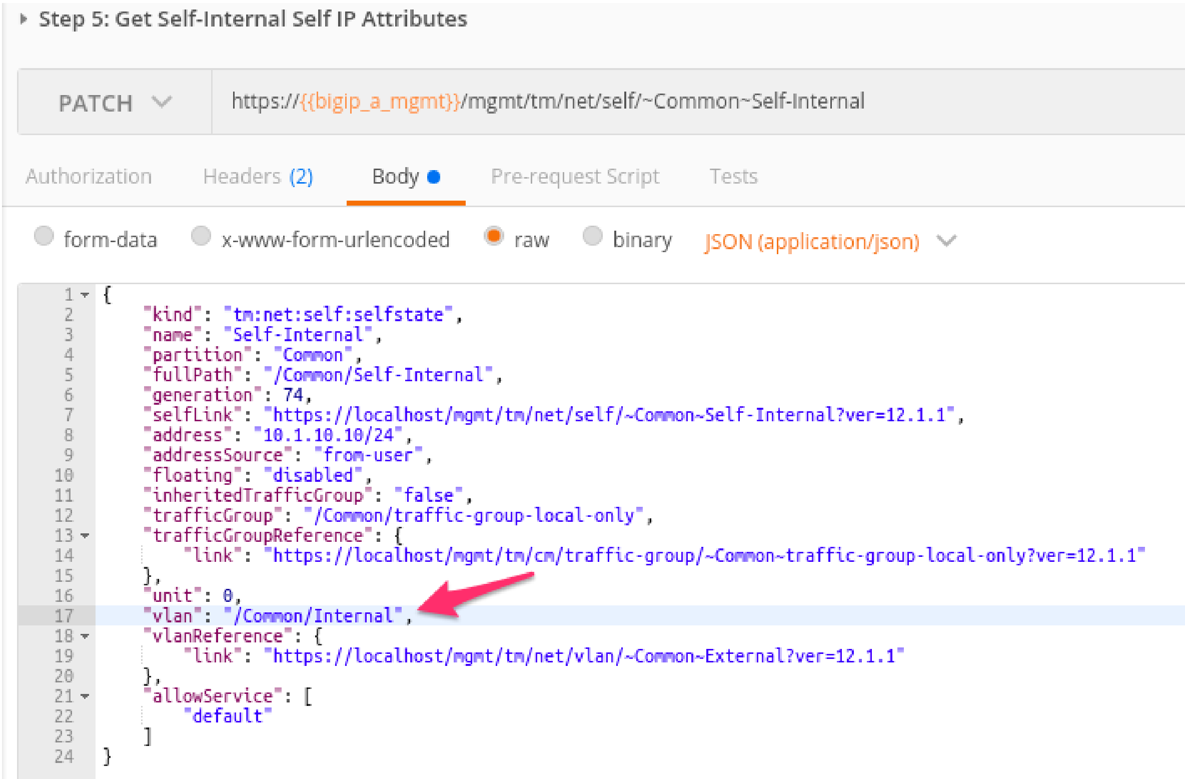
● raw JSON (application/json)

```
{
}
```

Status: 200 OK Time: 22 ms Size: 1.4 KB

```
{
    "kind": "tm:net:self:selfstate",
    "name": "Self-Internal",
    "partition": "Common",
    "fullPath": "/Common/Self-Internal",
    "generation": 74,
    "selfLink": "https://localhost/mgmt/tm/net/self/~Common~Self-Internal?ver=12.1.1",
    "address": "10.1.10.10/24",
    "addressSource": "from-user",
    "floating": "disabled",
```

4. In the JSON body change the `vlan` attribute to `/Common/Internal` and click `Send`:



```
{
    "kind": "tm:net:self:selfstate",
    "name": "Self-Internal",
    "partition": "Common",
    "fullPath": "/Common/Self-Internal",
    "generation": 74,
    "selfLink": "https://localhost/mgmt/tm/net/self/~Common~Self-Internal?ver=12.1.1",
    "address": "10.1.10.10/24",
    "addressSource": "from-user",
    "floating": "disabled",
    "inheritedTrafficGroup": "false",
    "trafficGroup": "/Common/traffic-group-local-only",
    "trafficGroupReference": {
        "link": "https://localhost/mgmt/tm/cm/traffic-group/~Common~traffic-group-local-only?ver=12.1.1"
    },
    "unit": 0,
    "vlan": "/Common/Internal",
    "vlanReference": {
        "link": "https://localhost/mgmt/tm/net/vlan/~Common~External?ver=12.1.1"
    },
    "allowService": [
        "default"
    ]
}
```

5. Click the `Step 6: Get Self IPs` item in the collection. Click the `Send` button to GET the Self IP collection. Examine the response to make sure both Self IPs have been created and belong to the appropriate vlan.

Task 4 - Create Routes

Perform the following steps to configure the Route object/resource:

1. Before creating the route, we double check the content of the routing table. Click the `Step 7: Get Routes` item in the collection. Click the `Send` button to `GET` the routes collection. Examine the response to make sure there is no route.
2. Click the `Step 8: Create a Route` item in the collection. Examine the JSON body; the values for creating the Default Route have already been populated.
3. Click the `Send` button to create the Route
4. Click the `Step 9: Get Routes` item in the collection again. Click the `Send` button to `GET` the routes collection. Examine the response to make sure the route has been created.

2.3.5 Lab 1.5: Building Imperative Workflows with Postman Collections

BIG-IP
REST Basics
Authentication
Global Settings
Networking
Clustering
Tran

As you have seen in the previous lab's we can use the Collections and Folders features of the Postman client to group REST requests logically. Additionally, as you've seen most of the examples so far have consisted of executing a sequence of REST request to achieve some outcome.

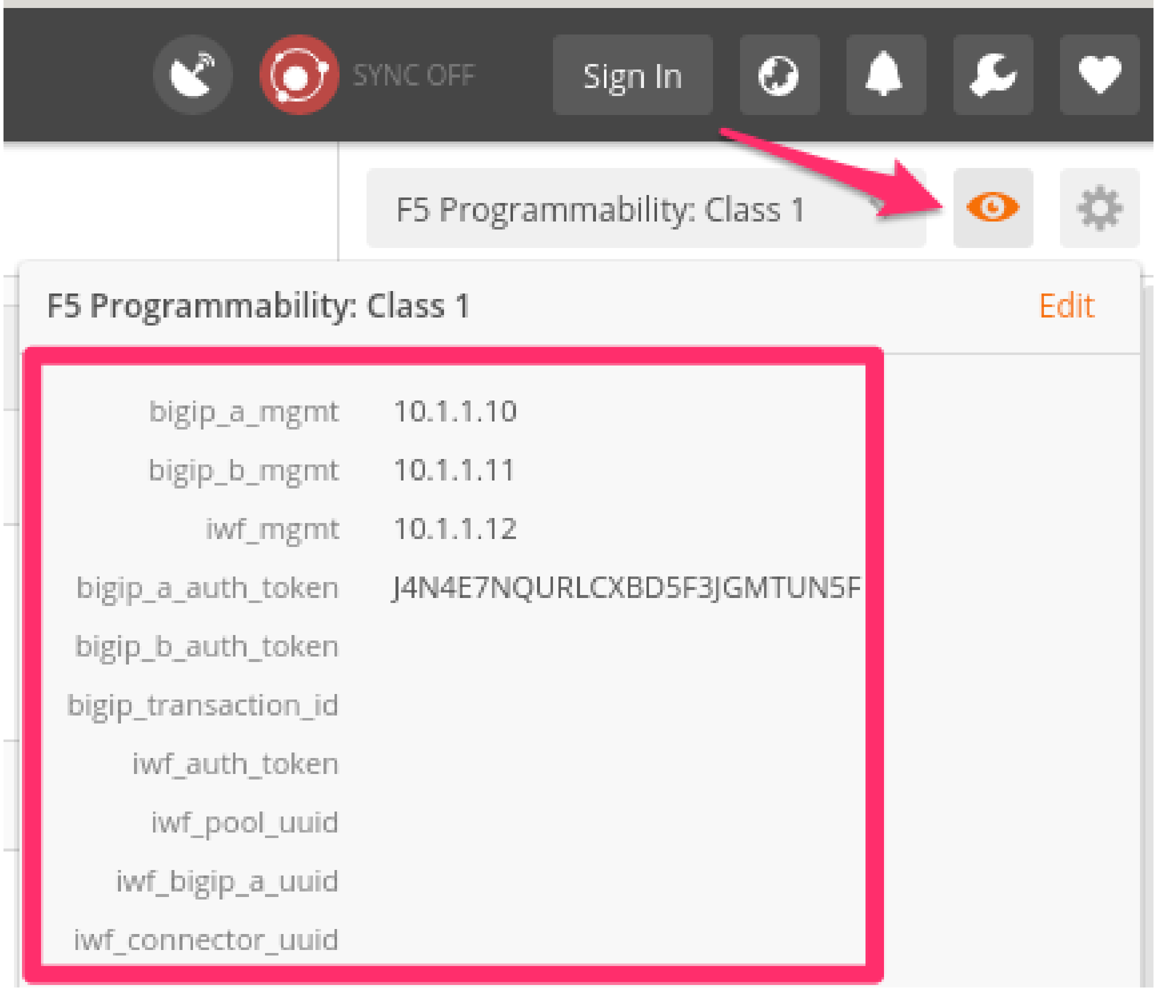
In this lab, we will use a feature in Postman called the **Collection Runner (Runner)** to execute a sequence of REST requests. Using the Runner we can rapidly prototype REST requests into an **Imperative Workflow** that can be executed without user intervention.

The purpose of this exercise is to provide an example of how new workflows can be built from scratch or existing workflows can be modified.

Additionally, we will use some Postman Javascript Tests to programmatically populate environment variables with the output of our workflow.

Task 1 - Open and Run a Collection

1. The collection we will run in this task will populate some environment variables with various data about the BIG-IP system. First, let's examine the Environment Variables that are currently set. Click the 👁 icon in the top right of the Postman window. Notice that there are no variables starting with the name `lab1.5_`:

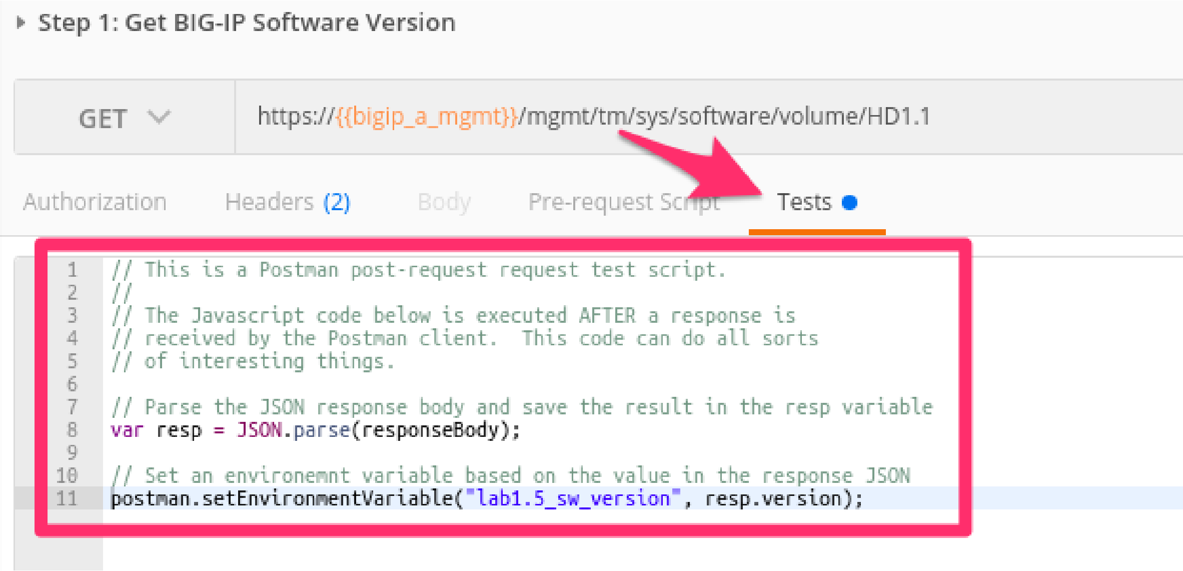


F5 Programmability: Class 1

F5 Programmability: Class 1 Edit

bigip_a_mgmt	10.1.1.10
bigip_b_mgmt	10.1.1.11
iwf_mgmt	10.1.1.12
bigip_a_auth_token	J4N4E7NQURLCXBD5F3JGMTUN5F
bigip_b_auth_token	
bigip_transaction_id	
iwf_auth_token	
iwf_pool_uuid	
iwf_bigip_a_uuid	
iwf_connector_uuid	

2. Click the `Lab 1.5 – Building Imperative Workflows` folder to expand it
3. Click the `Step 1: Get BIG-IP Software Version` request. Click the *Tests* tab and examine the Javascript code and comments:

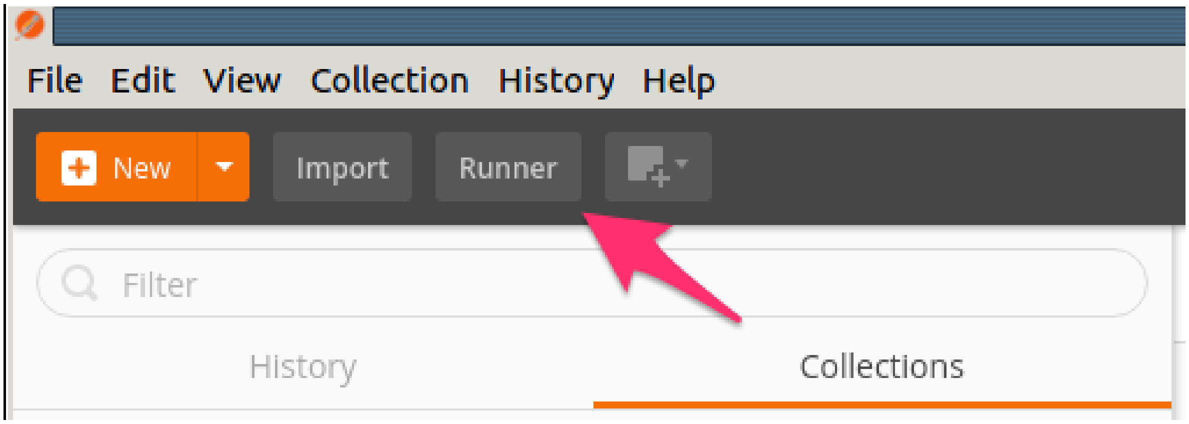


https://{{bigip_a_mgmt}}/mgmt/tm/sys/software/volume/HD1.1

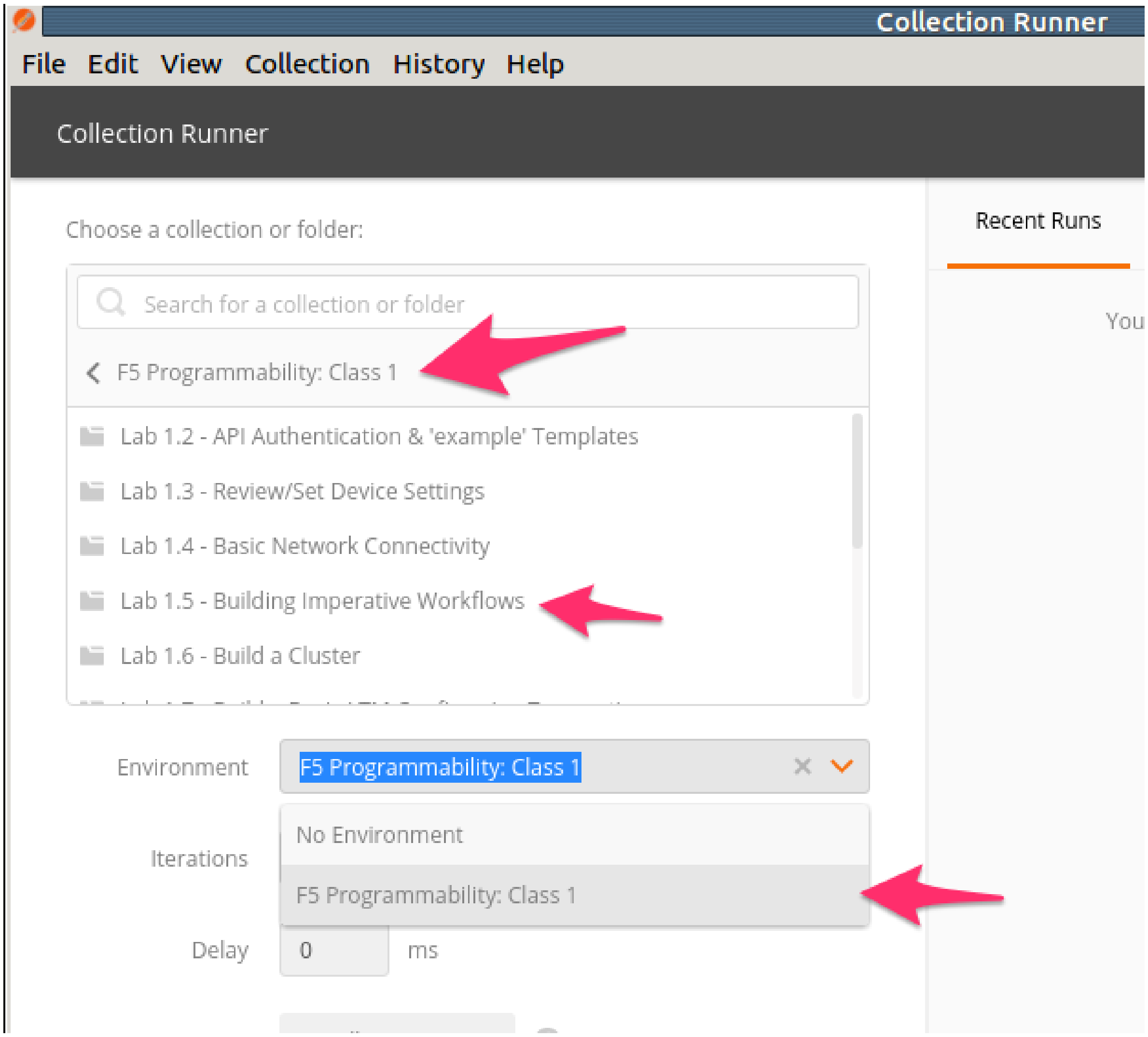
```
// This is a Postman post-request request test script.
//
// The Javascript code below is executed AFTER a response is
// received by the Postman client.  This code can do all sorts
// of interesting things.

// Parse the JSON response body and save the result in the resp variable
var resp = JSON.parse(responseBody);

// Set an environemnt variable based on the value in the response JSON
postman.setEnvironmentVariable("lab1.5_sw_version", resp.version);
```

The Javascript code in the Test script will populate an environment variable based on the response from the BIG-IP system.

4. Click the *Runner* button at the top right of your Postman window:



5. Select the `F5 Programmability: Class 1` Collection then the `Lab 1.5 – Building Imperative Workflows` folder. Next, be sure the environment is set to `F5 Programmability: Class 1`:



Choose a collection or folder:

Search for a collection or folder

F5 Programmability: Class 1

- Lab 1.2 - API Authentication & 'example' Templates
- Lab 1.3 - Review/Set Device Settings
- Lab 1.4 - Basic Network Connectivity
- Lab 1.5 - Building Imperative Workflows
- Lab 1.6 - Build a Cluster

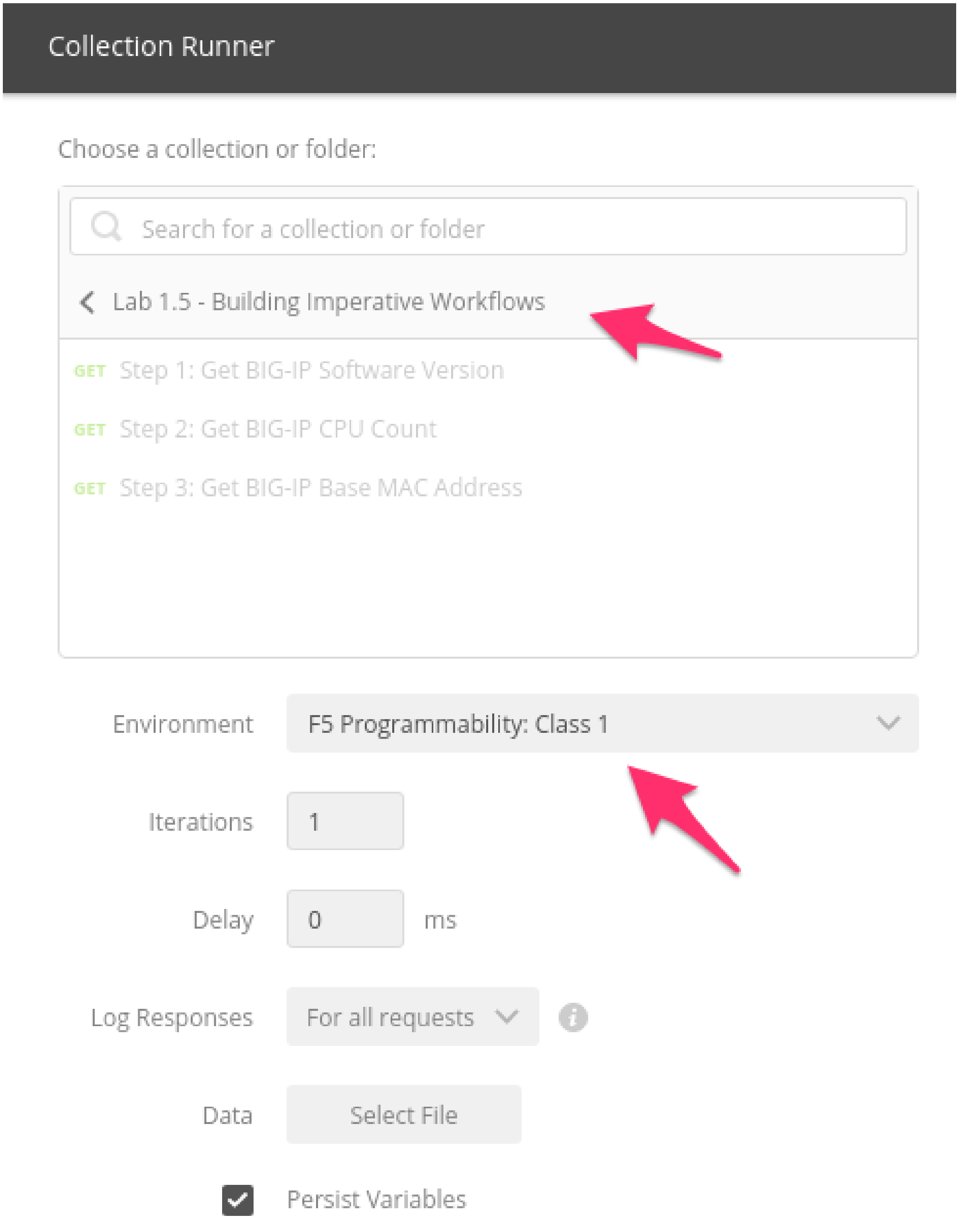
Environment: F5 Programmability: Class 1

- No Environment
- F5 Programmability: Class 1

Iterations

Delay: 0 ms

Your Runner window should look like:



Collection Runner
Choose a collection or folder:
Search for a collection or folder

Lab 1.5 - Building Imperative Workflows

GET Step 1: Get BIG-IP Software Version

GET Step 2: Get BIG-IP CPU Count

GET Step 3: Get BIG-IP Base MAC Address

Environment: F5 Programmability: Class 1

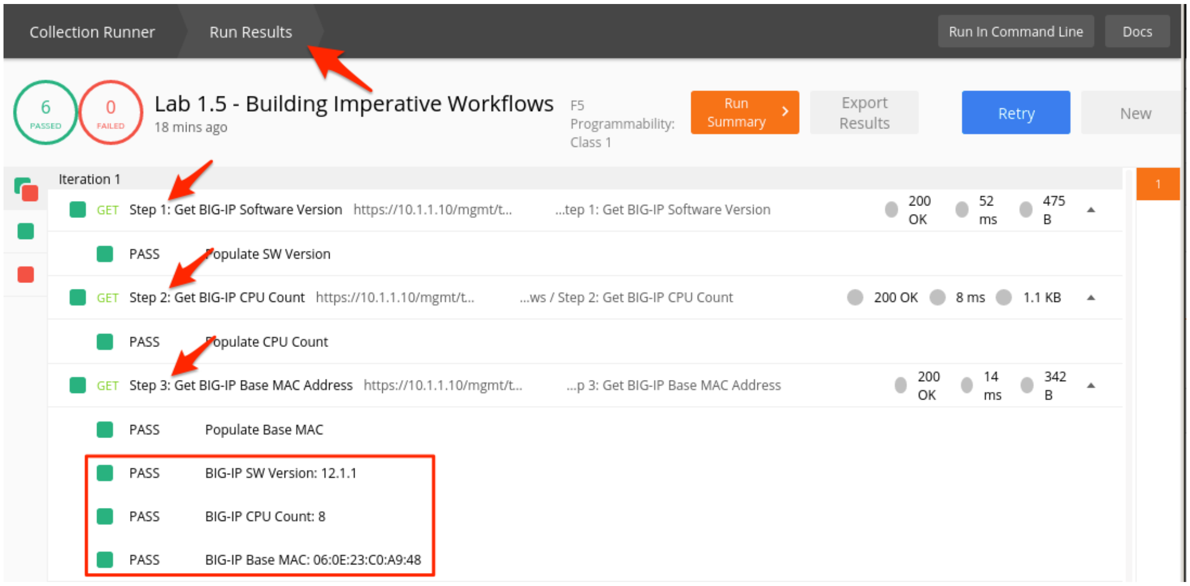
Iterations: 1

Delay: 0 ms

Log Responses: For all requests

☑ Persist Variables

6. Click the *Run Lab 1.5 - Buil. . .* button
7. The results window will now populate. You will see each request in the folder is sent and it's associated test results are displayed on the screen. The last request in the folder includes some Javascript code to dump the results to the screen:

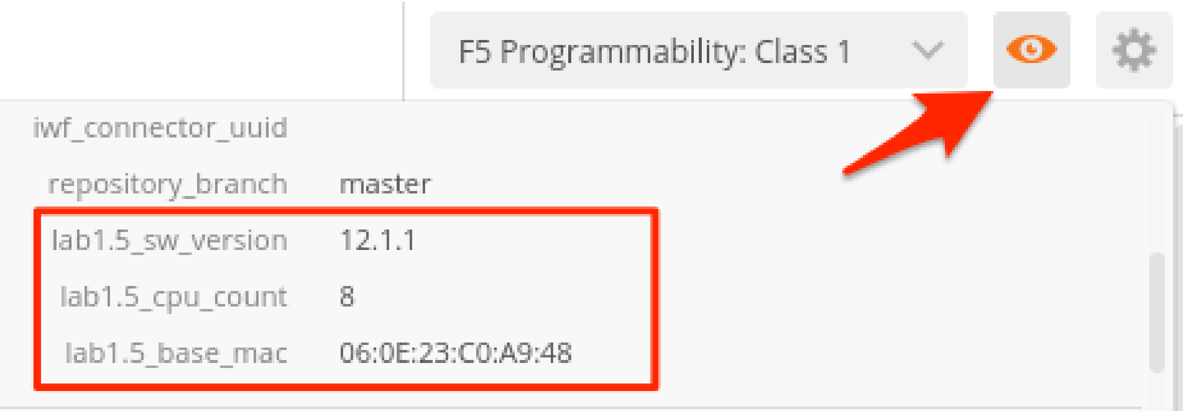


Lab 1.5 - Building Imperative Workflows
18 mins ago
F5 Programmability: Class 1

Iteration 1

GET	Step 1: Get BIG-IP Software Version	https://10.1.1.10/mgmt/t...	...tep 1: Get BIG-IP Software Version	200 OK	52 ms	475 B
PASS	Populate SW Version					
GET	Step 2: Get BIG-IP CPU Count	https://10.1.1.10/mgmt/t...	...ws / Step 2: Get BIG-IP CPU Count	200 OK	8 ms	1.1 KB
PASS	Populate CPU Count					
GET	Step 3: Get BIG-IP Base MAC Address	https://10.1.1.10/mgmt/t...	...p 3: Get BIG-IP Base MAC Address	200 OK	14 ms	342 B
PASS	Populate Base MAC					
PASS	BIG-IP SW Version: 12.1.1					
PASS	BIG-IP CPU Count: 8					
PASS	BIG-IP Base MAC: 06:0E:23:C0:A9:48					

8. Next, switch back to the main Postman window. Click the 👁 button again and examine the environment variables. Notice that three new variables starting with the name `lab1.5_` have been populated:



iwf_connector_uuid	
repository_branch	master
lab1.5_sw_version	12.1.1
lab1.5_cpu_count	8
lab1.5_base_mac	06:0E:23:C0:A9:48

Note: It is normal for the values of Software Version, CPU Count and Base MAC Address be different than the screenshot(s).

In this lab, we demonstrated running a simple Imperative Workflow using the Postman Collection Runner. In subsequent labs, we will expand on this simple use case to perform more complex functions. As you continue through the labs be sure to take time to explore the details of the requests being sent. The Postman Collection used in this class can also serve as a starting point for building your own collections or modifying existing ones.

As we move through the rest of this module you will see the complexity involved in building Imperative Workflows. While these types of workflows are incredibly powerful, they are also time-consuming to build from scratch. As we move into Module 2 you will see the importance of leveraging **Abstraction** and **Declarative Interfaces** to minimize the amount of time spent building Imperative Workflows.

2.3.6 Lab 1.6: Build a BIG-IP Cluster using a Collection

In this lab, we will build an active-standby cluster between BIG-IP-A and BIG-IP-B using the REST API. As mentioned previously, to save time, BIG-IP-B is already licensed and had its device-level settings configured. This lab will use the Postman Runner functionality introduced in the previous lab. We will run the requests in a Collection Folder to build the cluster. If you examine the `Lab 1.6 - Build a Cluster` folder in the Collection you can see how complex **Imperative** processes can become. Clustering is one of the *transition* points for most customers to move into the **Declarative** model (if not already done) due to the need to abstract device/vendor level specifics from Automation consumers.

The high-level procedure required to create the cluster is:

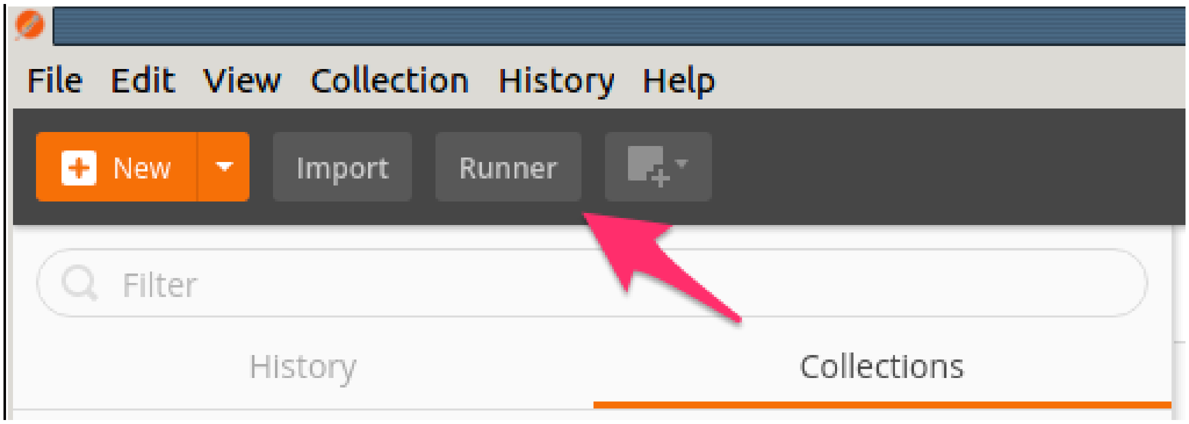
1. Obtain Authentication Tokens for BIGIP A & B
2. Check that both devices are licensed and ready to configure
3. Configure Device Level settings on both devices
4. Configure Networking on BIGIP-B (remember this was already done in Lab 1.4 for BIGIP-A)
5. Set BIGIP-A & BIGIP-B CMI Parameters (Config Sync IP, Failover IPs, Mirroring IP)
6. Add BIGIP-B as a trusted peer on BIGIP-A
7. Check the status of the Sync Groups
8. Create a sync-failover Device Group
9. Check the status of the created Device Group
10. Perform initial sync of the Device Group
11. Check status (again)
12. Change the Traffic Group to use HA Order failover
13. Create Floating Self IPs
14. Failover the Traffic Group to make BIGIP-A the Active device

Task 1 - Build a Cluster using Runner

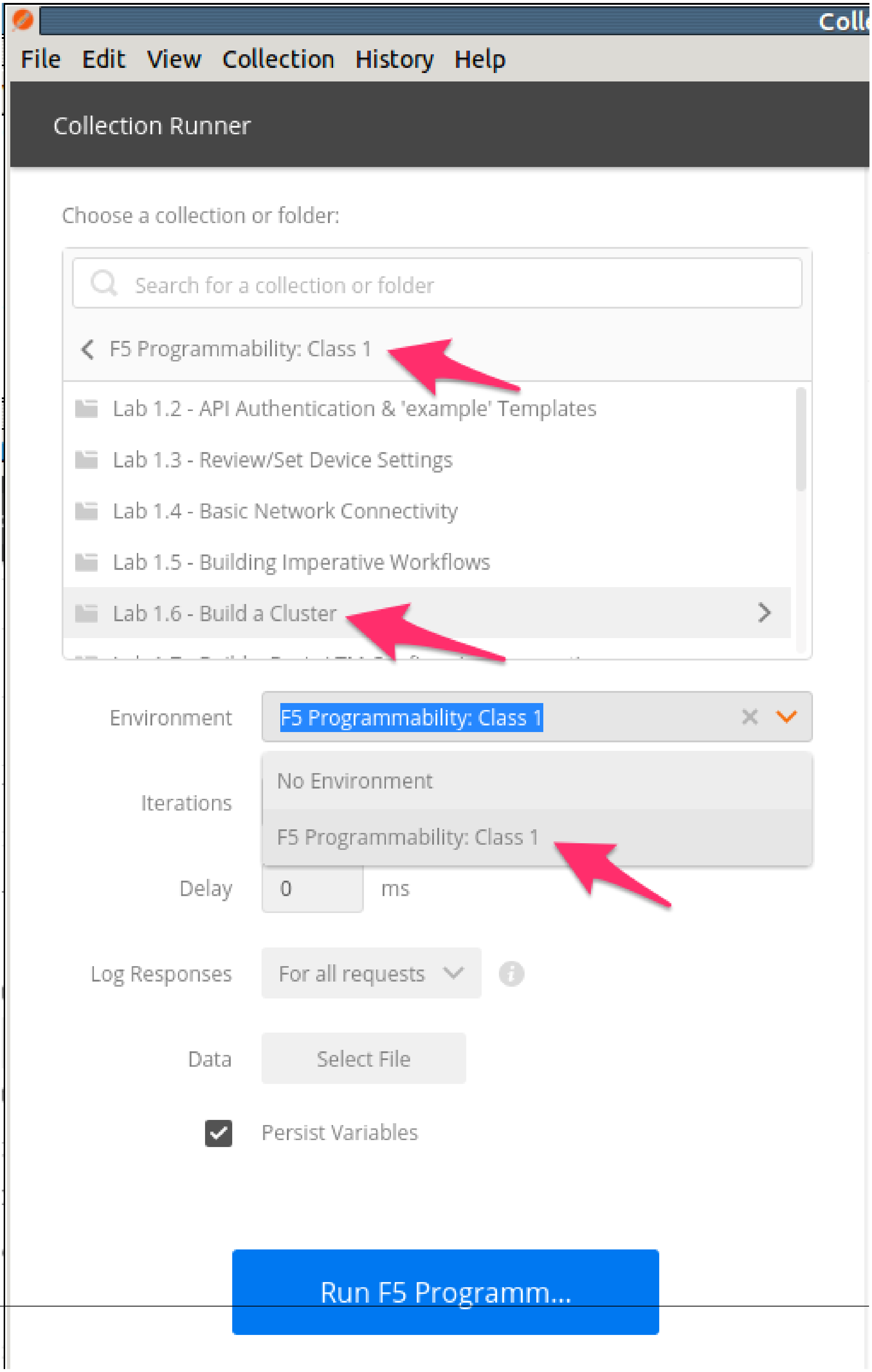
In this task we will use the *Runner* to execute a series of requests contained in the `Lab 1.6 - Build a Cluster` folder. As mentioned previously this folder contains a large number of REST requests required to build an Active/Standby cluster. Additionally, we will make use of a JavaScript framework called `f5-postman-workflows` that extends the Postman client to include common test and polling functions.

Perform the following steps to build the cluster:

1. Click the *Runner* button at the top right of your Postman window:



2. Select the `F5 Programmability: Class 1` Collection then the `Lab 1.6 - Build a Cluster` folder. Next, be sure the environment is set to `F5 Programmability: Class 1`:



Choose a collection or folder:

F5 Programmability: Class 1

- Lab 1.2 - API Authentication & 'example' Templates
- Lab 1.3 - Review/Set Device Settings
- Lab 1.4 - Basic Network Connectivity
- Lab 1.5 - Building Imperative Workflows
- Lab 1.6 - Build a Cluster

Environment: F5 Programmability: Class 1

- No Environment
- F5 Programmability: Class 1

Iterations

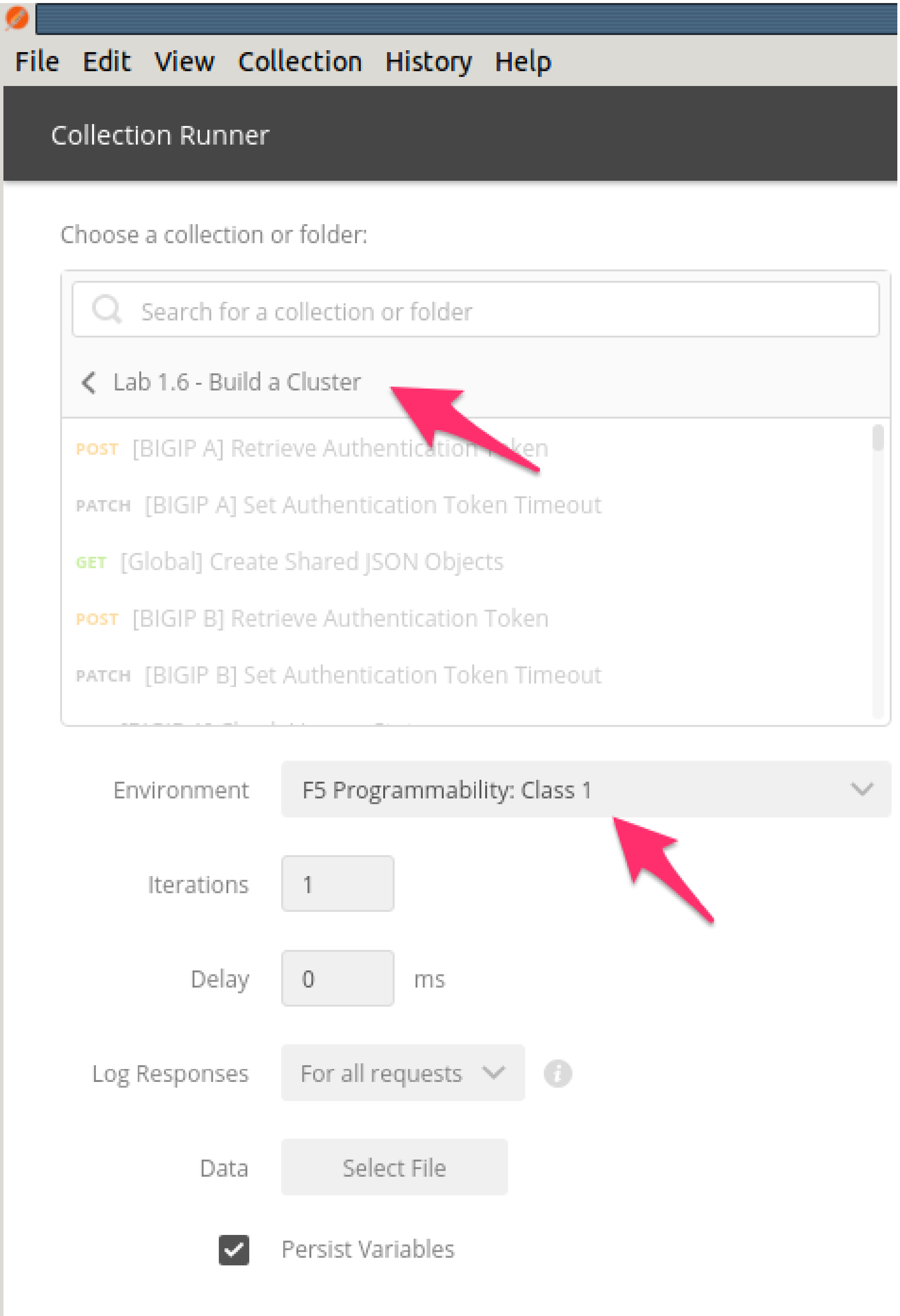
Delay: 0 ms

Log Responses: For all requests

Data: Select File

☑ Persist Variables

Your Runner window should look like:



Choose a collection or folder:

Lab 1.6 - Build a Cluster

POST [BIGIP A] Retrieve Authentication Token

PATCH [BIGIP A] Set Authentication Token Timeout

GET [Global] Create Shared JSON Objects

POST [BIGIP B] Retrieve Authentication Token

PATCH [BIGIP B] Set Authentication Token Timeout

Environment: F5 Programmability: Class 1

Iterations: 1

Delay: 0 ms

Log Responses: For all requests

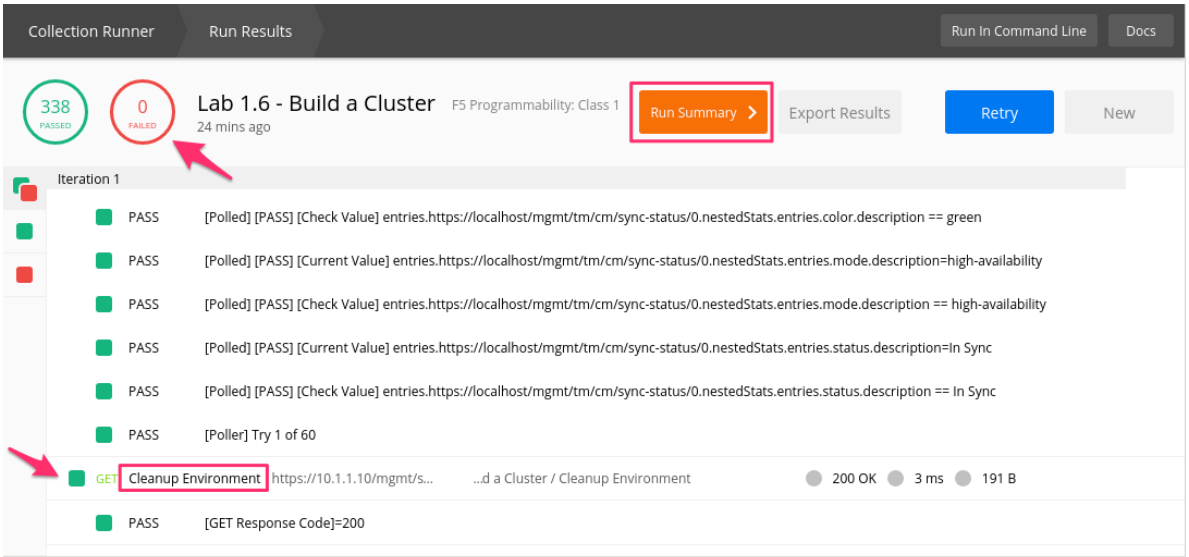
Data

☑ Persist Variables

3. Click the *Run Lab 1.6 - Buil. . .* button

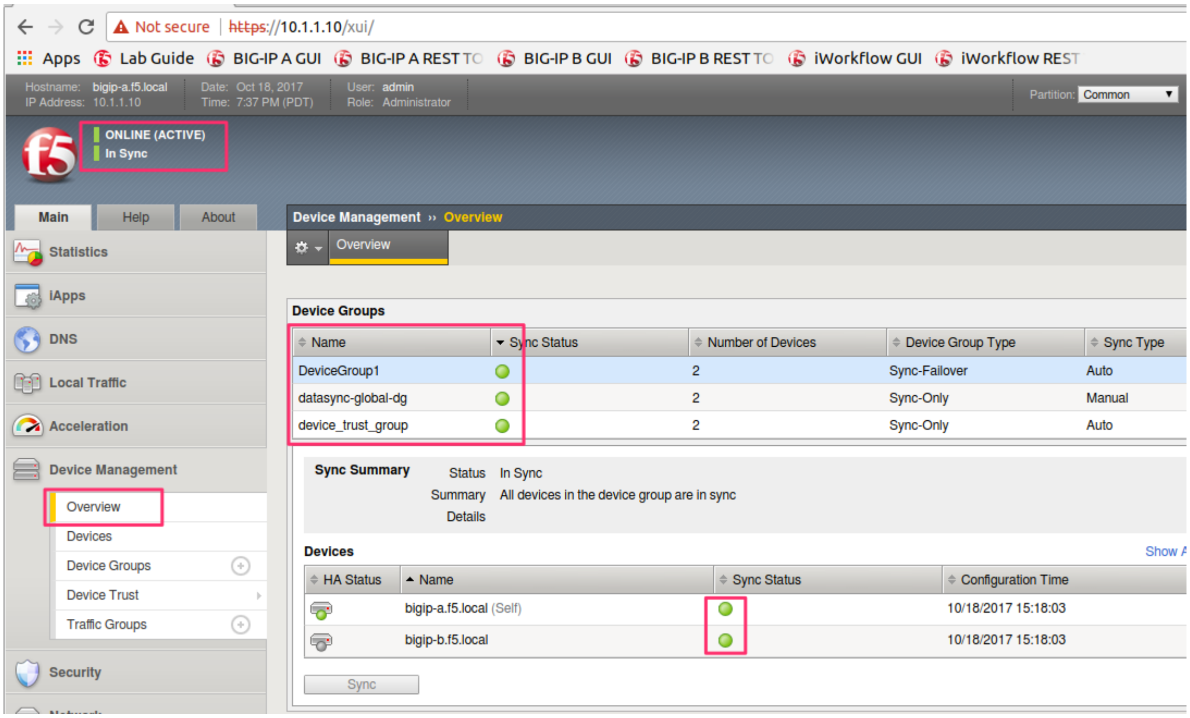
4. The results window will now populate. You will see each request in the folder is sent and it's associated test results are displayed on the screen. Building the cluster can take a few minutes. You can follow the progress by scrolling down the results window.

5. Once the *Run Summary* button appears the folder has finished running. You should have 0 failures and the last item in the request list should be named `Cleanup Environment`



```
PASS   [Polled] [PASS] [Check Value] entries.https://localhost/mgmt/tm/cm/sync-status/0.nestedStats.entries.color.description == green
PASS   [Polled] [PASS] [Current Value] entries.https://localhost/mgmt/tm/cm/sync-status/0.nestedStats.entries.mode.description=high-availability
PASS   [Polled] [PASS] [Check Value] entries.https://localhost/mgmt/tm/cm/sync-status/0.nestedStats.entries.mode.description == high-availability
PASS   [Polled] [PASS] [Current Value] entries.https://localhost/mgmt/tm/cm/sync-status/0.nestedStats.entries.status.description=In Sync
PASS   [Polled] [PASS] [Check Value] entries.https://localhost/mgmt/tm/cm/sync-status/0.nestedStats.entries.status.description == In Sync
PASS   [Poller] Try 1 of 60
GET Cleanup Environment https://10.1.1.10/mgmt/s...   ...d a Cluster / Cleanup Environment   200 OK   3 ms   191 B
PASS   [GET Response Code]=200
```

6. At this point you can log into BIG-IP A using Chrome at `https://10.1.1.10` and verify the cluster was built by using the menu in the BIG-IP GUI to navigate to *Device Management* → *Overview* and verifying the cluster and failover status indicators are all Green



Hostname: bigip-a.f5.local
IP Address: 10.1.1.10
Date: Oct 18, 2017
Time: 7:37 PM (PDT)
User: admin
Role: Administrator

Device Groups

Name	Sync Status	Number of Devices	Device Group Type	Sync Type
DeviceGroup1	🟢	2	Sync-Failover	Auto
datasync-global-dg	🟢	2	Sync-Only	Manual
device_trust_group	🟢	2	Sync-Only	Auto

Sync Summary
Status: In Sync
Summary: All devices in the device group are in sync
Details:

Devices

HA Status	Name	Sync Status	Configuration Time
	bigip-a.f5.local (Self)	🟢	10/18/2017 15:18:03
	bigip-b.f5.local	🟢	10/18/2017 15:18:03

2.3.7 Lab 1.7: Build a Basic LTM Config using REST Transactions

BIG-IP
REST Basics
Authentication
Global Settings
Networking
Clustering
Tran

In this lab we will build a basic LTM Config using iControl REST API Transactions.

Task 1 - Create a Transaction

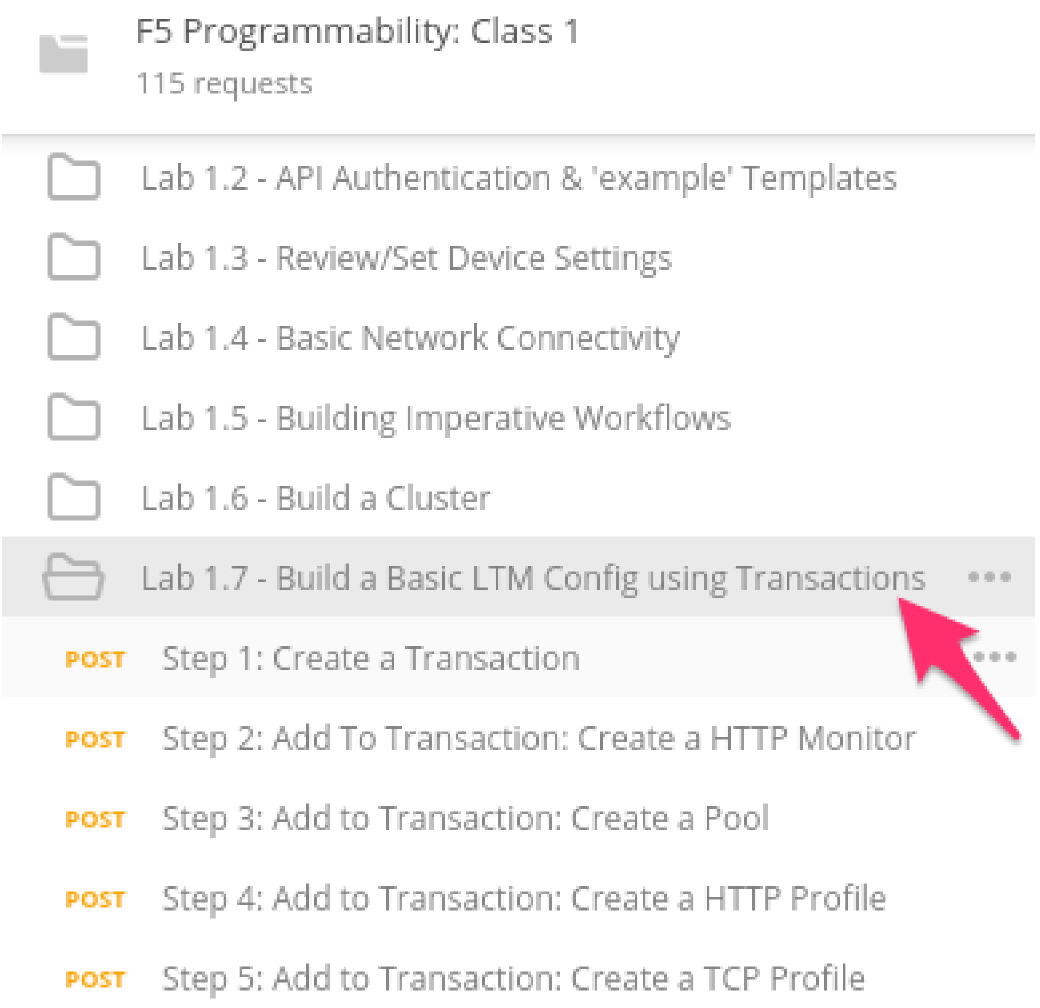
Transactions are very useful in cases where you would want discrete REST operations to act as a batch operation. As a result, the nature of a transaction is that either all the operations succeed or none of them do (all-or-nothing). This is very useful when creating a configuration that is linked together because it allows the roll back of operations in case one fails. All the commands issued are queued one after the other in the transaction. We will also review how to change the order of a queued command or remove a single command from the queued list before committing.

Note: Transactions are essential to ensure that an Imperative process is **Atomic** in nature.

Warning: Transactions have a default timeout of 120 seconds. Taking longer than the timeout to execute the transaction will result in its automatic deletion. To avoid having to redo the steps in this task, please read the steps below first and then execute each one in a timely manner.

Perform the following steps to complete this task:

1. Expand the `Lab 1.7 - Build a Basic LTM Config using Transactions` folder in the Postman collection:



F5 Programmability: Class 1

115 requests

Lab 1.2 - API Authentication & 'example' Templates

Lab 1.3 - Review/Set Device Settings

Lab 1.4 - Basic Network Connectivity

Lab 1.5 - Building Imperative Workflows

Lab 1.6 - Build a Cluster

Lab 1.7 - Build a Basic LTM Config using Transactions •••

POST Step 1: Create a Transaction •••

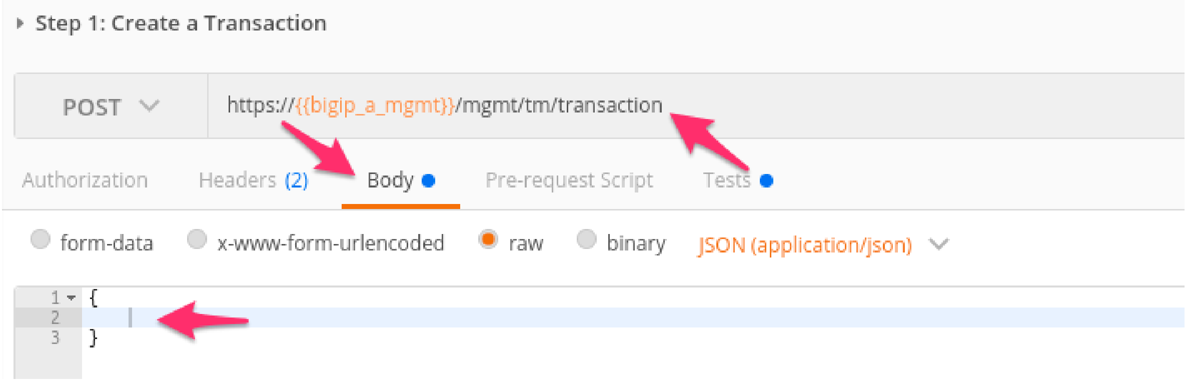
POST Step 2: Add To Transaction: Create a HTTP Monitor

POST Step 3: Add to Transaction: Create a Pool

POST Step 4: Add to Transaction: Create a HTTP Profile

POST Step 5: Add to Transaction: Create a TCP Profile

2. Click the `Step 1: Create a Transaction` item. Examine the URL and JSON body. We will send a `POST` to the `/mgmt/tm/transaction` endpoint with an empty JSON body to create a new transaction.



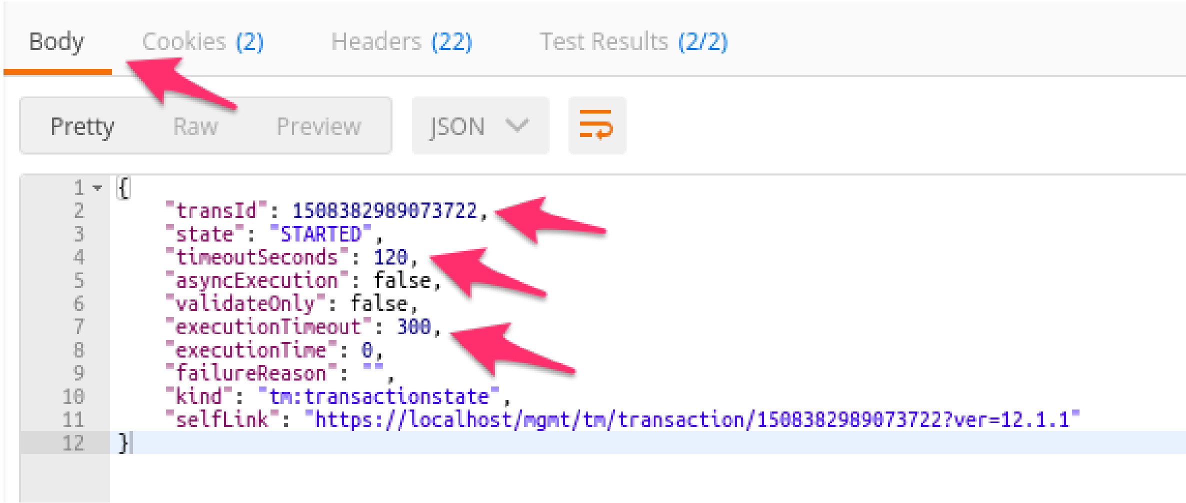
▸ Step 1: Create a Transaction

POST ∨ https://{{bigip_a_mgmt}}/mgmt/tm/transaction

○ form-data ○ x-www-form-urlencoded ● raw ○ binary JSON (application/json) ∨

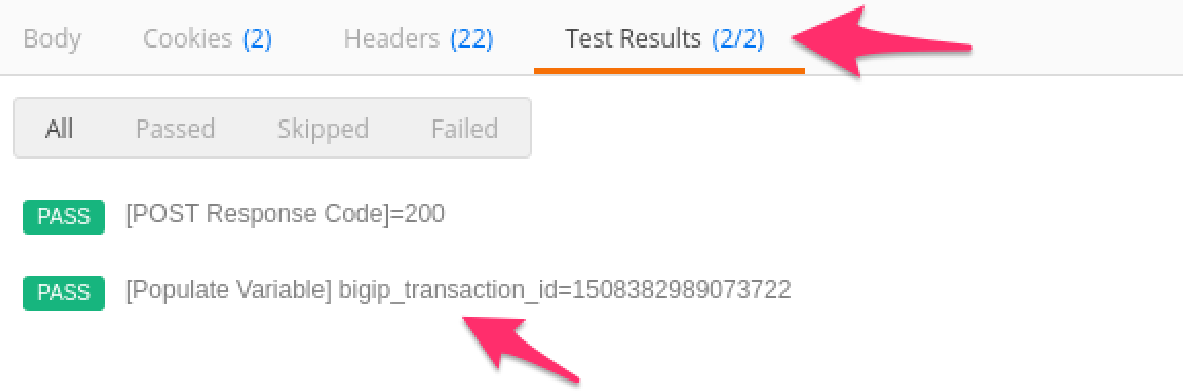
```
{

}
```

3. Click the *Send* button to send the request. Examine the response and find the `transId` attribute. Additionally, notice that there are timeouts for both the submission of the transaction and how long it should take to execute. Be aware that after the `timeoutSeconds` value, this `transId` will be silently removed:



```
{
    "transId": 1508382989073722,
    "state": "STARTED",
    "timeoutSeconds": 120,
    "asyncExecution": false,
    "validateOnly": false,
    "executionTimeout": 300,
    "executionTime": 0,
    "failureReason": "",
    "kind": "tm:transactionstate",
    "selfLink": "https://localhost/mgmt/tm/transaction/1508382989073722?ver=12.1.1"
}
```

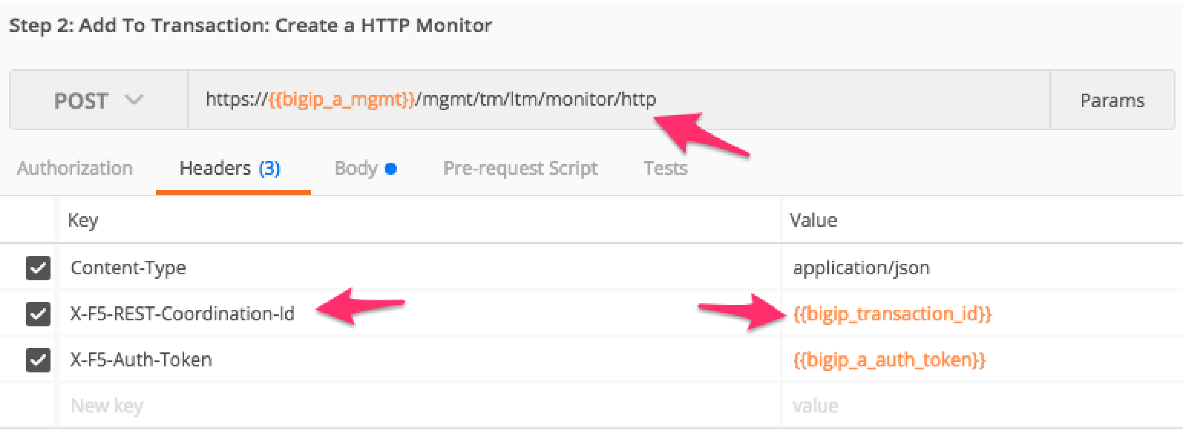
The `transId` value has been automatically populated for you in the `bigip_transaction_id` environment variable:



PASS [POST Response Code]=200

PASS [Populate Variable] bigip_transaction_id=1508382989073722

4. Click the `Step 2: Add to Transaction: Create a HTTP Monitor` item in the Postman collection. This request is the same as a non-transaction enabled request in terms of the `POST` request method, URI and JSON body. The difference is we add a `X-F5-REST-Coordination-Id` header with a value of the `transId` attribute to add it to the transaction:


Step 2: Add To Transaction: Create a HTTP Monitor

POST https://{{bigip_a_mgmt}}/mgmt/tm/ltm/monitor/http

	Key	Value
☑	Content-Type	application/json
☑	X-F5-REST-Coordination-Id	{{bigip_transaction_id}}
☑	X-F5-Auth-Token	{{bigip_a_auth_token}}
	New key	value

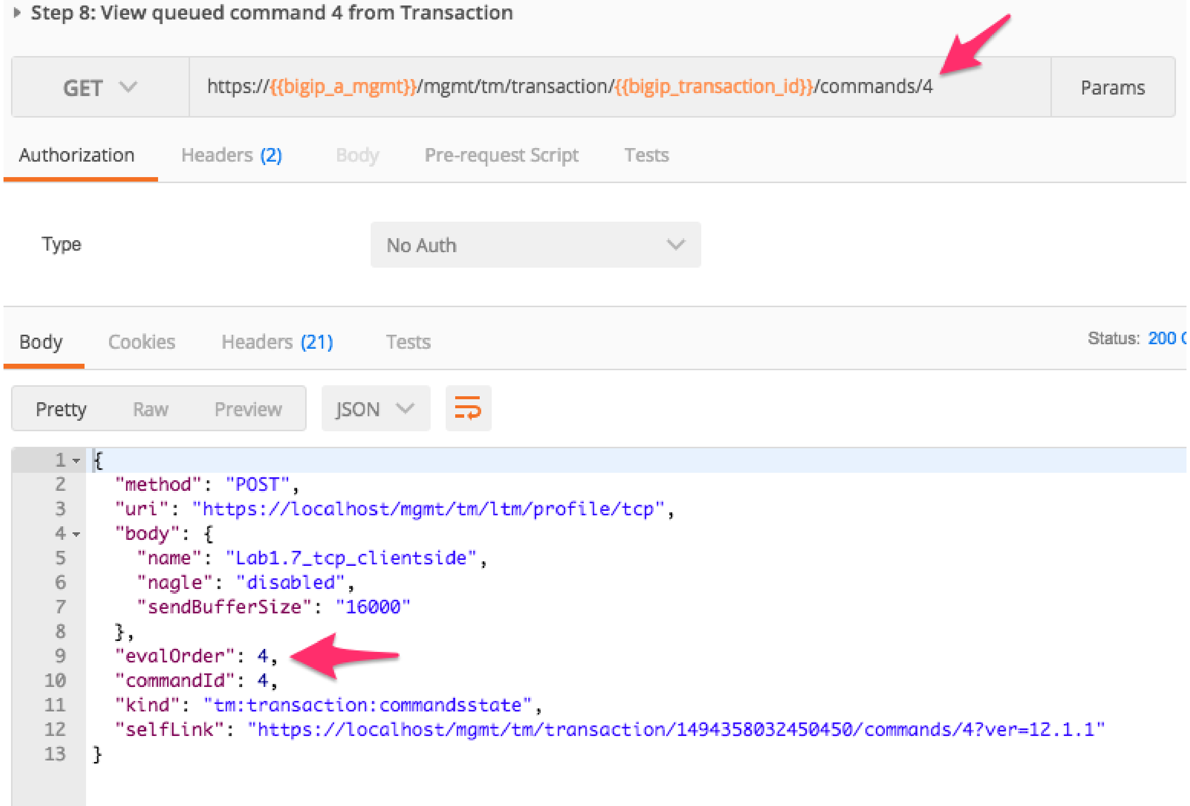
5. Click the *Send* button and examine the response

6. Examine and click *Send* on **Steps 3-6** in the folder

7. Click `Step 7:  View the Transaction Queue`. Examine the request type and URI and click *Send*. This request allows you to see the current list of ordered commands in the transaction.

Task 2 - Modify a Transaction

1. Click the `Step 8:  View Queued Command 4 from Transaction` item in the folder. Examine the request method and URI. We will `GET` command number **4** from the transaction queue.



```
https://{{bigip_a_mgmt}}/mgmt/tm/transaction/{{bigip_transaction_id}}/commands/4
```

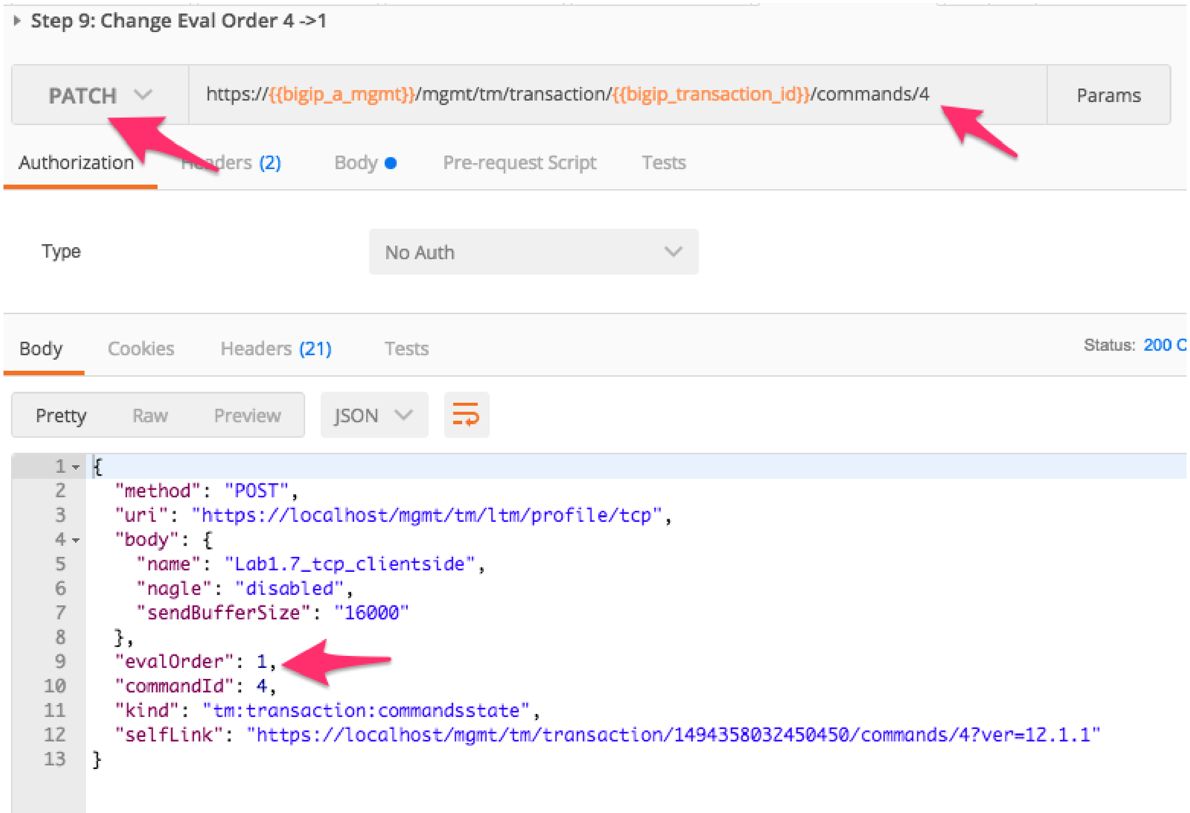
```
{
  "method": "POST",
  "uri": "https://localhost/mgmt/tm/ltm/profile/tcp",
  "body": {
    "name": "Lab1.7_tcp_clientside",
    "nagle": "disabled",
    "sendBufferSize": "16000"
  },
  "evalOrder": 4,
  "commandId": 4,
  "kind": "tm:transaction:commandsstate",
  "selfLink": "https://localhost/mgmt/tm/transaction/1494358032450450/commands/4?ver=12.1.1"
}
```

2. Click the `Step 9:  Change Eval Order 4 -> 1` item in the folder. Examine the request method, URI and JSON body. We will PATCH our transaction resource and change the value of the `evalOrder` attribute from `4` to `1` to move at the first position of the transaction queue:

Note: Requests in the ordered transaction queue must obey the order of operations present in the underlying BIG-IP system.

Warning: When sending the Header `X-F5-REST-Coordination-Id`, the system assumes you want to **ADD** an entry in the transaction queue. You **MUST** remove this header if you want to issue transaction queue changes (like deleting an entry from the queue, changing the order, committing a transaction). If you don't remove the header, the system will respond with a `400` HTTP error code with the following error text:

```
"message":  "Transaction XXXXX operation .... is not allowed to be
added to transaction."
```

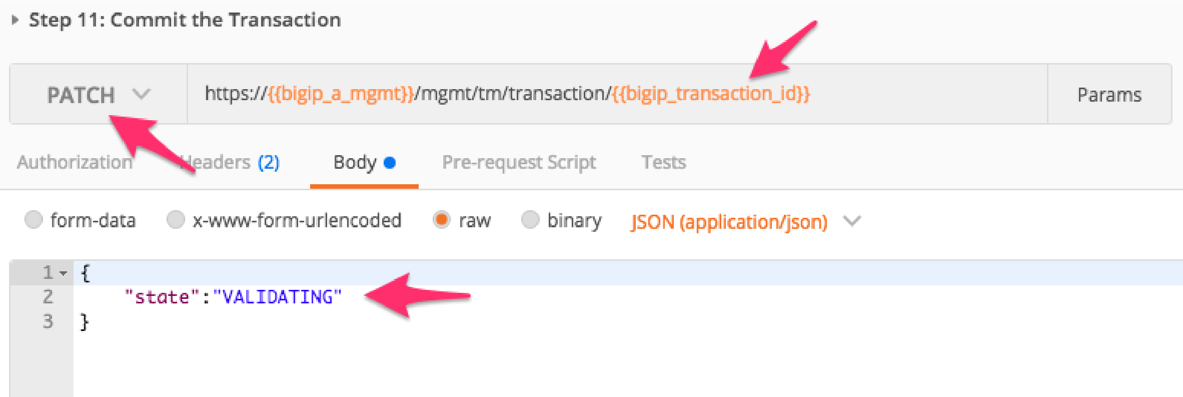


```
{
  "method": "POST",
  "uri": "https://localhost/mgmt/tm/ltm/profile/tcp",
  "body": {
    "name": "Lab1.7_tcp_clientside",
    "nagle": "disabled",
    "sendBufferSize": "16000"
  },
  "evalOrder": 1,
  "commandId": 4,
  "kind": "tm:transaction:commandsstate",
  "selfLink": "https://localhost/mgmt/tm/transaction/1494358032450450/commands/4?ver=12.1.1"
}
```

3. Click the `Step 10: View the Transaction Queue Changes` item in the folder. Verify that request number `4` has moved into position `1` and the order of all other requests has been updated accordingly.

Task 3 - Commit a Transaction

1. Click the `Step 11: Commit the Transaction` item in the folder. Examine the request type, URI and JSON body. We will `PATCH` our transaction resource and change the value of the `state` attribute to submit the transaction:



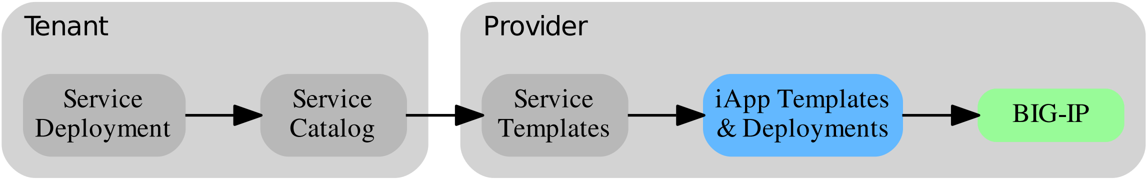
```
{
    "state":"VALIDATING"
}
```

2. Click the *Send* button and examine the response. The `state` may already be `COMPLETED`, however, it's good practice to explicitly check for this.

3. Click the `Step 12: View the Transaction Status` item in the folder and click the *Send* button. Verify that the `state` of the transaction is `COMPLETED`

4. You can verify the configuration was created on the BIG-IP device via the BIG-IP A GUI at `https://10.1.1.10`
5. Verify the virtual server works by opening `http://10.1.20.120` in the Chrome web browser

2.4 Module 2: Abstracting Services using iApp Templates



In this Module, we will continue working with the BIG-IP REST interface. However, we will now introduce F5 Declarative Interfaces built with F5 iApp templates.

iApps are a user-customizable framework for deploying applications that enables you to templatize sets of functionality on your F5 devices. For example, you can automate the process of adding virtual servers or build a custom iApp to manage your iRules inventory.

iApps are commonly thought of as a Wizard style deployment helper, but they are actually a Declarative Interface (like REST Transactions).

When an iApp deploys, a **single** call - declaring the desired deployment - is processed on the BIG-IP in the correct order of operations. All created objects are associated with an Application Service Object (ASO). The ASO model identifies which objects belong to the iApp service deployment. Upon service deletion, all service related objects are recursively deleted.

We will be using the **F5 App Services Integration iApp** (App Services iApp for short).

For further information about the App Services iApp see:

- **GitHub Repository:** https://github.com/F5Networks/f5-application-services-integration-iApp
- **User Guide:** https://devcentral.f5.com/wiki/iApp.AppSvcsiApp_userguide_userguide.ashx

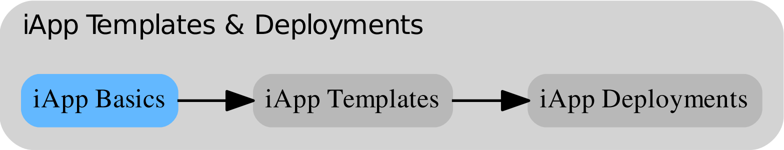
An overview of iApps and different iApp templates that available can be found at:

- https://devcentral.f5.com/iapps

Note: This module requires the underlying network configuration that was completed in Module 1. Additionally, **BIG-IP A** must be the **Active** node in the cluster

Note: This module deploys the configuration to BIG-IP A. iApp deployments leverage the underlying config-sync mechanisms in the cluster. Once deployed on BIG-IP A, the configuration will be automatically synced to BIG-IP B

2.4.1 Lab 2.1: Exploring iApps



iApp Templates & Deployments

A BIG-IP device has multiple ways to install an iApp onto its platform, including TMOS Shell (TMSH), the GUI (TMUI), and the REST Interface. All mechanisms are valid and, if needed, can be used in conjunction with each other.

For instance, you can install an iApp template from the GUI and then deploy a new service via iControl REST using tools like cURL, Postman and Ansible.

Note: Redeployment of iApp templates makes use of an underlying mechanism in the BIG-IP platform that allows safe changes to the configuration without interrupting existing user traffic.

F5 iApps were introduced in TMOS Version 11, they can interact within, and across different F5 modules providing full Layer 4-7 Application Service capability. The **iApp Template** is used to drive an **iApp Deployment** that creates a configuration under an **Application Service Object (ASO)**. The ASO model identifies which objects belong to the iApp service deployment. Upon service deletion, all service related objects are recursively deleted.

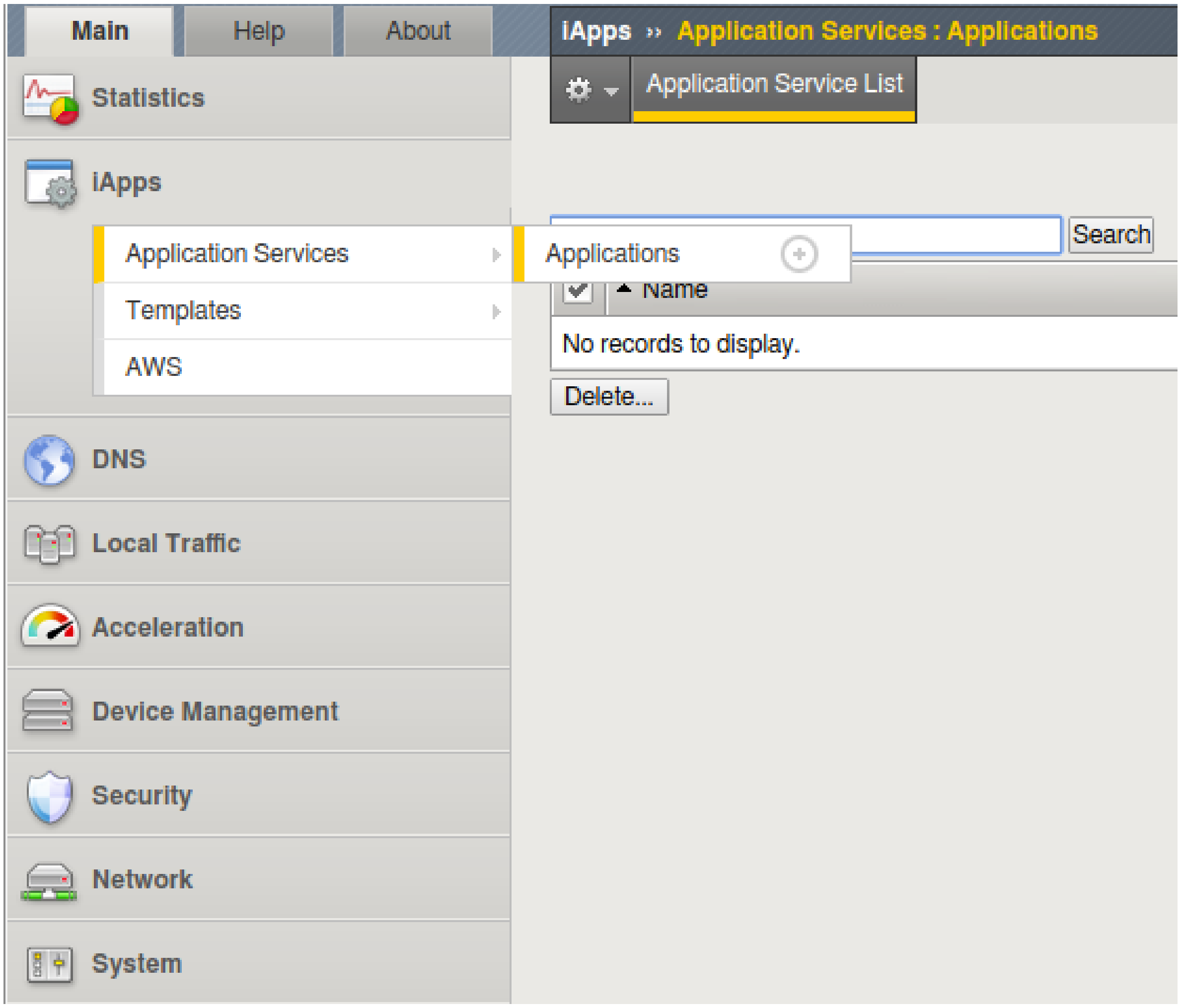
Some examples of the modules we can use iApp templates to configure:

- Local Traffic Manager
- Advanced Firewall Manager
- Application Security Manager
- Access Policy Manager

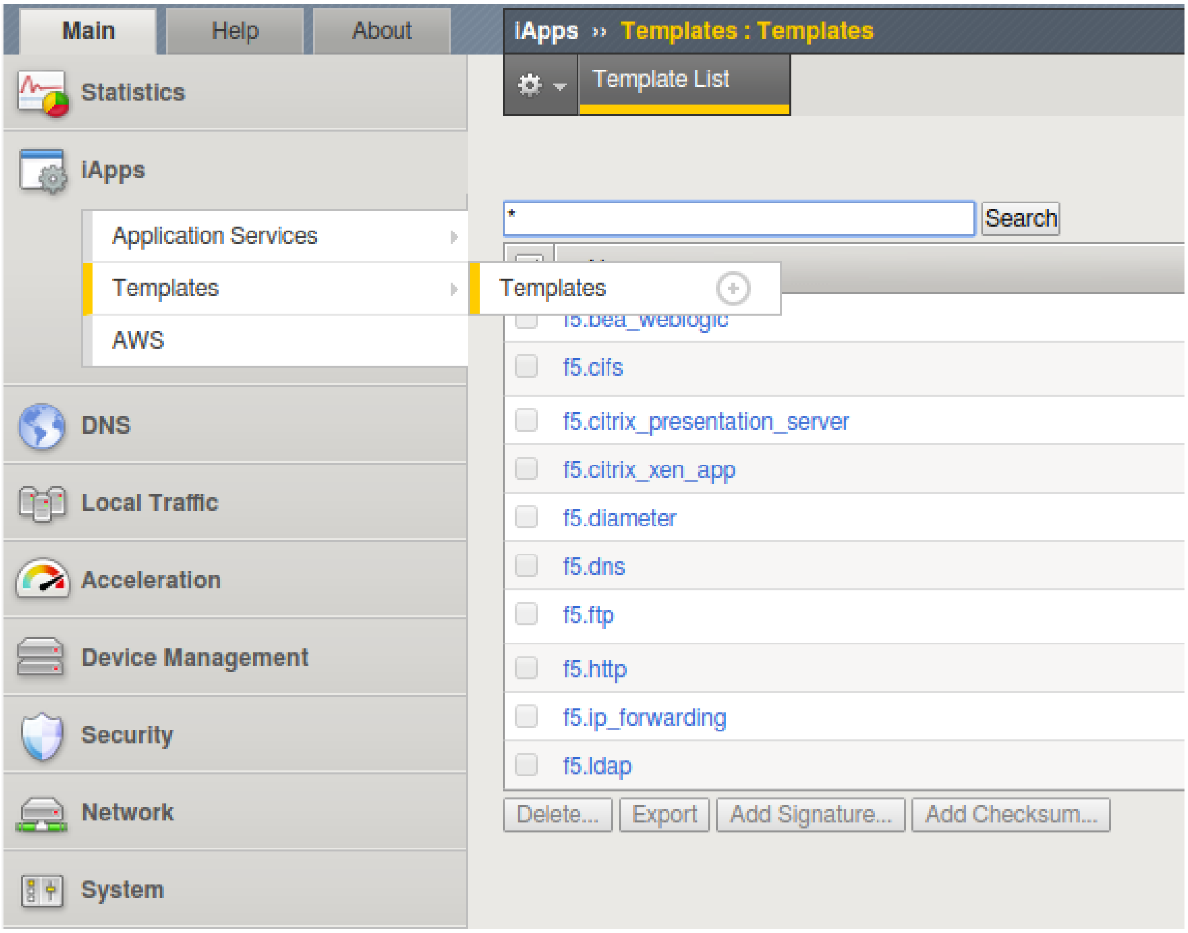
Note: `Application Service` in the GUI and `service` in the REST API are the same objects. The name is slightly abbreviated in the API.

You can find the GUI representation of iApps on the left-hand side of the UI under *iApps*. iApp deployments are located under *Application Services*, while iApp templates are located under *Templates* on the system.

- *Application Services* (iApp deployments)



- *Templates* (iApp templates)



The associated REST API endpoints are:

- **iApp Deployments**: `/mgmt/tm/cloud/services/iapp`
- **iApp Templates**: `/mgmt/tm/sys/application/template`

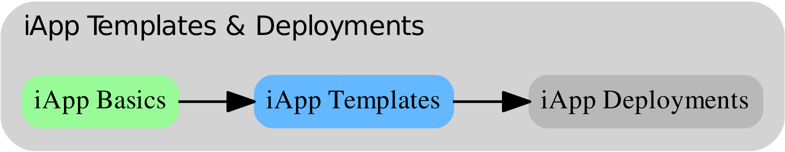
iApp Deployments and Source-of-Truth

By default, iApp technology implements a strict source-of-truth preservation mechanism called **Strict Updates**. The App Service iApp allows granular configuration of the underlying TMOS objects **without** disabling the Strict Updates mechanism, however, not all iApp templates implement this functionality.

In non-automated environments, the impact of this can be justified. However, in automated environments we must always guarantee that the template inputs are the Source-of-Truth for an underlying deployment. As a result, **Strict Updates should not be disabled**.

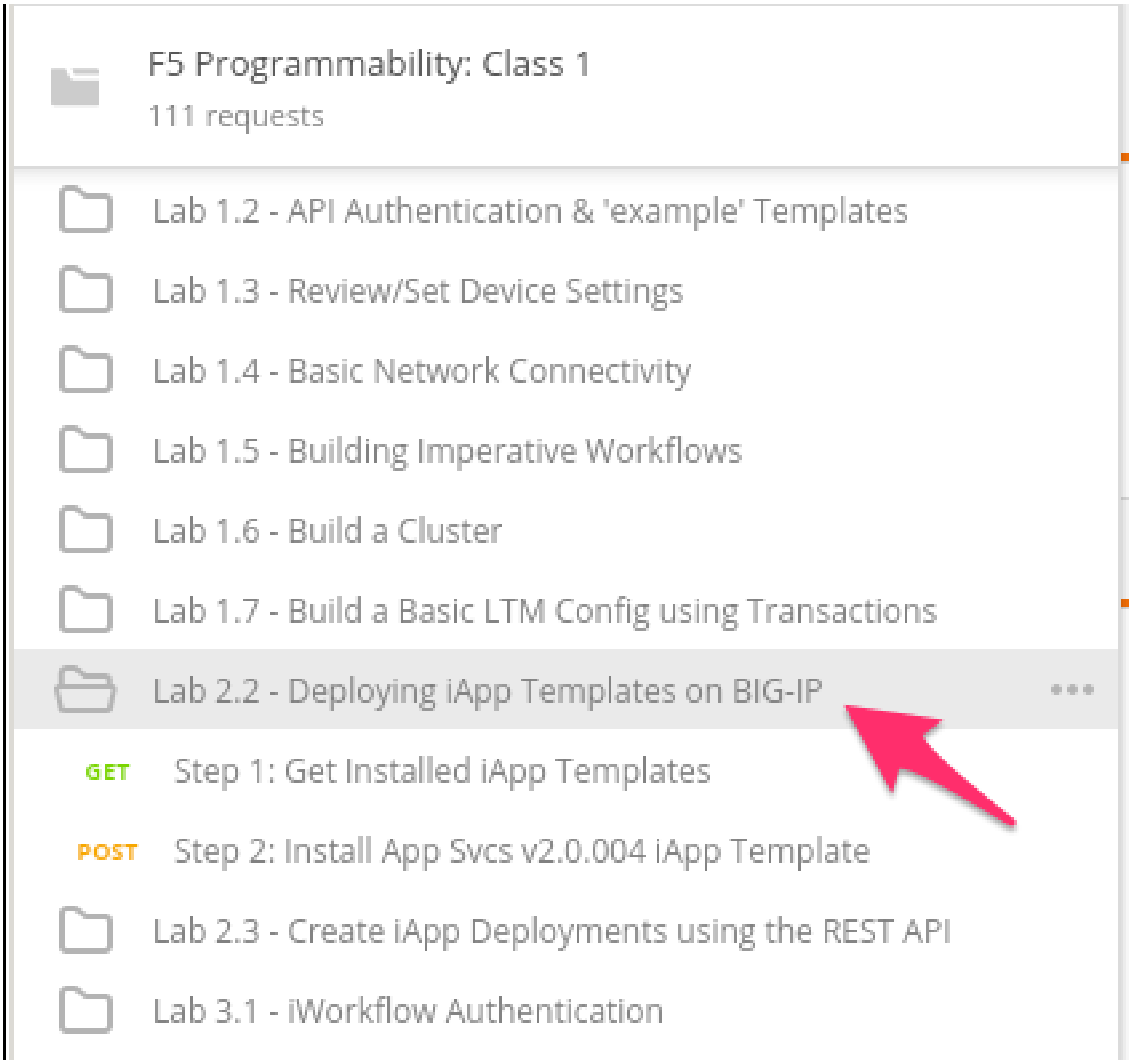
For example, creating an iApp deployment, disabling Strict Updates and then modifying the underlying configuration results in a Source-of-Truth violation because redeployment of the iApp would result in the changes made directly to the configuration being overwritten. It is because the direct modification of the configuration moved the Source-of-Truth to the object itself, rather than the iApp deployment input values that automation tools are interacting with.

2.4.2 Lab 2.2: Deploying iApp Templates on BIG-IP



iApps typically come in the form of a `.tmpl` file, which contains the content needed for the BIG-IP to utilize it as a Service framework. Different toolkits will install iApps in different ways. We'll be using the REST API in a raw form, so the content of the file is what we need. As a result, we need to ensure that the content of the iApp is URL encoded to make sure the BIG-IP reads the payload correctly. This is specific for the iApp deployment over REST API. When using other tools like Ansible, the whole `.tmpl` file can be uploaded, removing the need for encoding.

Note: This lab work will be performed from `Lab 2.2 - Deploying iApp Templates on BIG-IP` folder in the Postman Collection



F5 Programmability: Class 1

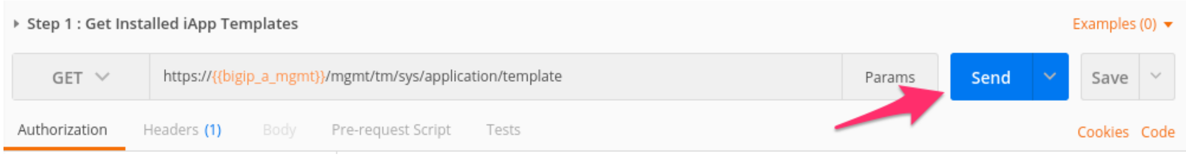
111 requests

- Lab 1.2 - API Authentication & 'example' Templates
- Lab 1.3 - Review/Set Device Settings
- Lab 1.4 - Basic Network Connectivity
- Lab 1.5 - Building Imperative Workflows
- Lab 1.6 - Build a Cluster
- Lab 1.7 - Build a Basic LTM Config using Transactions
- Lab 2.2 - Deploying iApp Templates on BIG-IP
  - GET Step 1: Get Installed iApp Templates
  - POST Step 2: Install App Svcs v2.0.004 iApp Template
- Lab 2.3 - Create iApp Deployments using the REST API
- Lab 3.1 - iWorkflow Authentication

Task 1 - View Installed iApp Templates

Perform the following steps to complete this task:

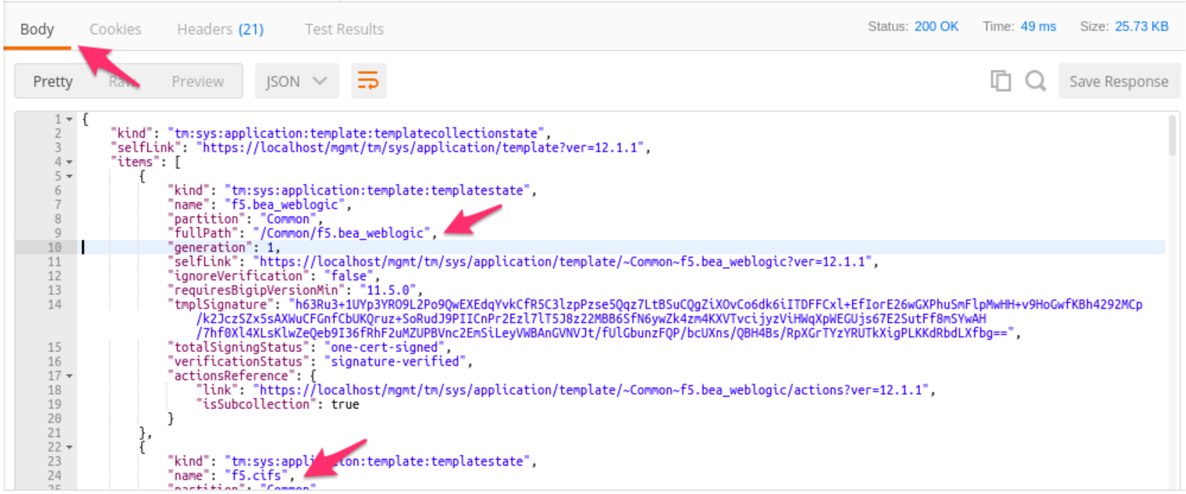
1. *Send* the `Step 1: Get Installed iApp Templates` request to view iApp templates installed on the BIG-IP device:



Step 1 : Get Installed iApp Templates

GET https://{{bigip_a_mgmt}}/mgmt/tm/sys/application/template

2. Review the JSON response *Body*. The JSON payload shows a iApp templates that are installed by default on the BIG-IP device:

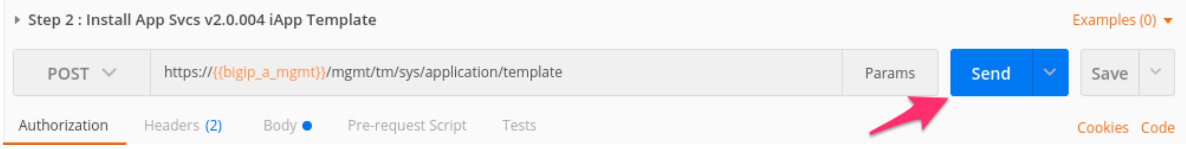


```
{
    "kind": "tm:sys:application:template:templatecollectionstate",
    "selfLink": "https://localhost/mgmt/tm/sys/application/template?ver=12.1.1",
    "items": [
        {
            "kind": "tm:sys:application:template:templatestate",
            "name": "f5.bea_weblogic",
            "partition": "Common",
            "fullPath": "/Common/f5.bea_weblogic",
            "generation": 1,
            "selfLink": "https://localhost/mgmt/tm/sys/application/template/~Common~f5.bea_weblogic?ver=12.1.1",
            "ignoreVerification": "false",
            "requiresBigipVersionMin": "11.5.0",
            "tmplSignature": "h63Ru3+1UYp3YRO9L2Po9QwEXEdqYvkCfR5C3lzpPze5Qqz7LtBSuCQgZiXOvCo6dk6iITDFFCxl+EfIorE26wGXPhuSmFlpMwHH+v9HoGwfKBh4292MCp
                /k2JczSZx5sAXWuCFGnfCbUKQruz+SoRudJ9PIICnPr2Ezl7lT5J8z22MBB6SfN6ywZk4zm4KXVTvcijyzViHWqXpWEGUjs67E2SutFf8mSYwAH
                /7hf0Xl4XLsKlwZeQeb9I36fRhF2uMZUPBVnc2EmSiLeyVWBAnGVNVJt/fUlGbunzFQP/bcUXns/QBH4Bs/RpXGrTYzYRUTkXigPLKKdRbdLXfbg==",
            "totalSigningStatus": "one-cert-signed",
            "verificationStatus": "signature-verified",
            "actionsReference": {
                "link": "https://localhost/mgmt/tm/sys/application/template/~Common~f5.bea_weblogic/actions?ver=12.1.1",
                "isSubcollection": true
            }
        },
        {
            "kind": "tm:sys:application:template:templatestate",
            "name": "f5.cifs",
            "partition": "Common",
```

Task 2 - Install the App Services iApp Template

Perform the following steps to complete this task:

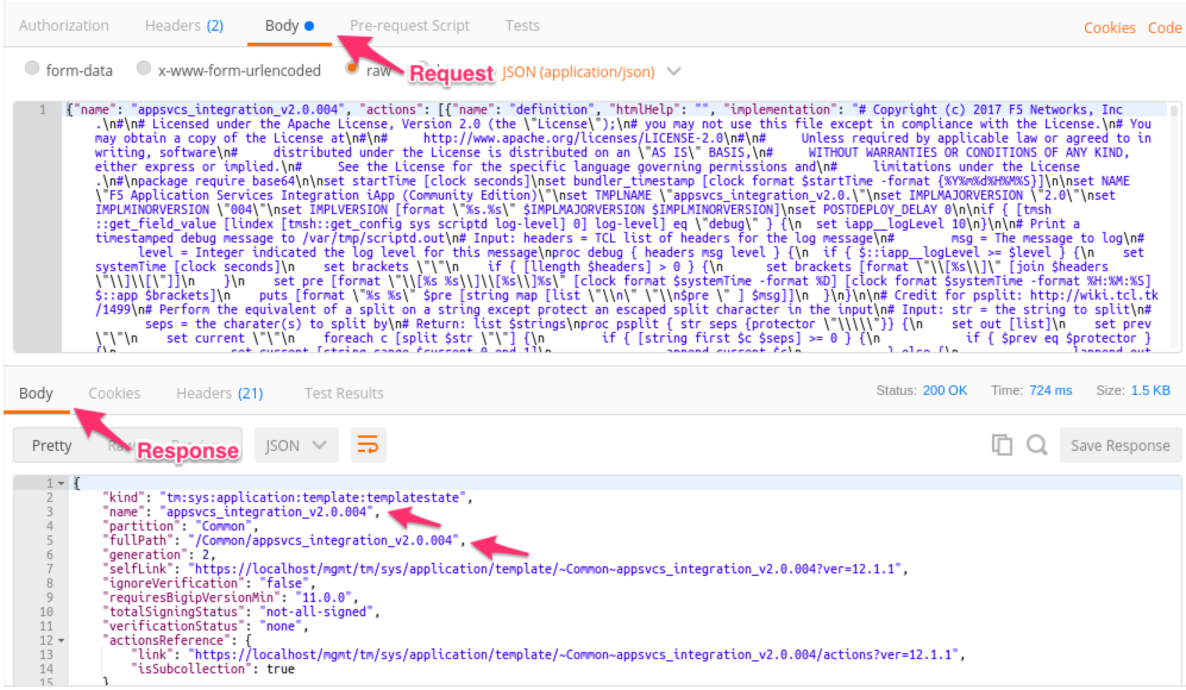
1. *Send* the `Step 2: Install App Svcs v2.0.004 iApp Template` request to install the iApp template:



https://{{bigip_a_mgmt}}/mgmt/tm/sys/application/template

2. Review the **Request** JSON *Body*, and the **Response** JSON *Body*. In this task we installed the App Services iApp Template and the BIG-IP sent back a response that the iApp was installed with its object name.

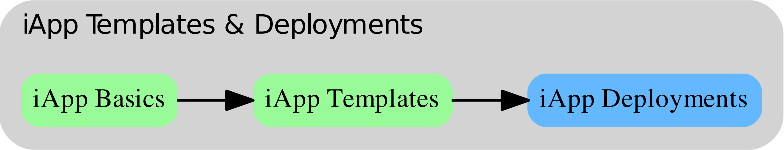
Note: The JSON body in the **Request** portion is automatically generated as part of the build process for the App Services iApp and the request in the Postman Collection was copied from a pre-built collection that ships with releases of the App Services iApp template.



```
{"name": "appsvcs_integration_v2.0.004", "actions": [{"name": "definition", "htmlHelp": "", "implementation": "# Copyright (c) 2017 F5 Networks, Inc
    .\n#\n# Licensed under the Apache License, Version 2.0 (the \"License\");\n# you may not use this file except in compliance with the License.\n# You
    may obtain a copy of the License at\n#\n#     http://www.apache.org/licenses/LICENSE-2.0\n#\n#     Unless required by applicable law or agreed to in
    writing, software\n#     distributed under the License is distributed on an \"AS IS\" BASIS,\n#     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
    either express or implied.\n#     See the License for the specific language governing permissions and\n#     limitations under the License
    .\n#\npackage require base64\n\nset startTime [clock seconds]\nset bundler_timestamp [clock format $startTime -format {%Y%m%d%H%M%S}]\n\nset NAME
    \"F5 Application Services Integration iApp (Community Edition)\"\nset TMPLNAME \"appsvcs_integration_v2.0.\"\nset IMPLMAJORVERSION \"2.0\"\nset
    IMPLMINORVERSION \"004\"\nset IMPLVERSION [format \"%s.%s\" $IMPLMAJORVERSION $IMPLMINORVERSION]\nset POSTDEPLOY_DELAY 0\n\nif { [tmsh
    ::get_field_value [lindex [tmsh::get_config sys scriptd log-level] 0] log-level] eq \"debug\" } {\n  set iapp__logLevel 10\n}\n\n# Print a
    timestamped debug message to /var/tmp/scriptd.out\n# Input: headers = TCL list of headers for the log message\n#        msg = The message to log\n#
          level = Integer indicated the log level for this message\nproc debug { headers msg level } {\n  if { $::iapp__logLevel >= $level } {\n    set
    systemTime [clock seconds]\n    set brackets \"\"\n    if { [llength $headers] > 0 } {\n      set brackets [format \"\\[%s\\]\" [join $headers
    \"\\]\\[\"]]\n    }\n    set pre [format \"\\[%s %s\\]\\[%s\\]%s\" [clock format $systemTime -format %D] [clock format $systemTime -format %H:%M:%S]
    $::app $brackets]\n    puts [format \"%s %s\" $pre [string map [list \"\\n\" \"\\n$pre \" ] $msg]]\n  }\n}\n\n# Credit for psplit: http://wiki.tcl.tk
    /1499\n# Perform the equivalent of a split on a string except protect an escaped split character in the input\n# Input: str = the string to split\n#
           seps = the charater(s) to split by\n# Return: list $strings\nproc psplit { str seps {protector \"\\\\\"}} {\n    set out [list]\n    set prev
    \"\"\n    set current \"\"\n    foreach c [split $str \"\"] {\n        if { [string first $c $seps] >= 0 } {\n            if { $prev eq $protector }
```

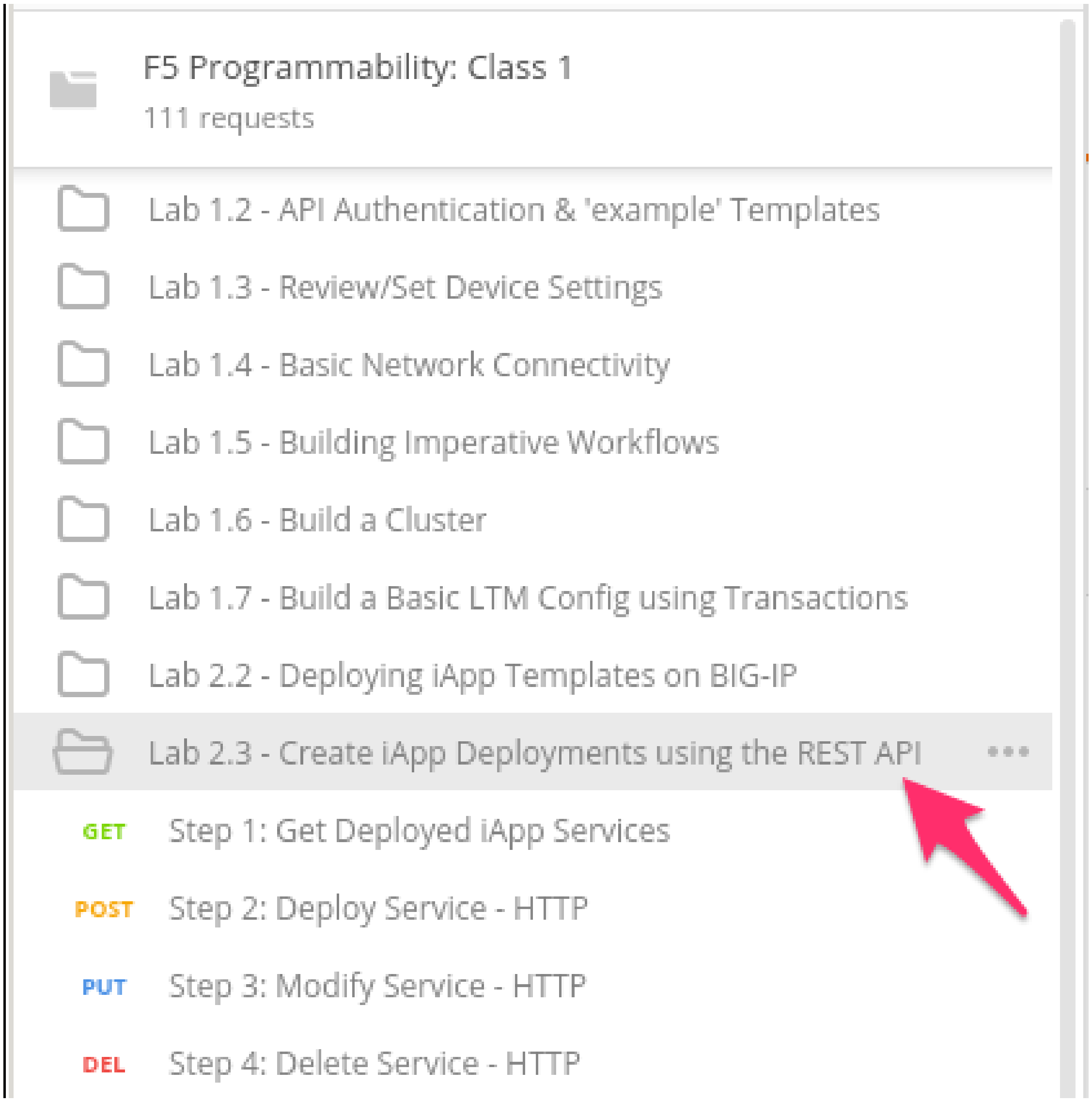
```
{
    "kind": "tm:sys:application:template:templatestate",
    "name": "appsvcs_integration_v2.0.004",
    "partition": "Common",
    "fullPath": "/Common/appsvcs_integration_v2.0.004",
    "generation": 2,
    "selfLink": "https://localhost/mgmt/tm/sys/application/template/~Common~appsvcs_integration_v2.0.004?ver=12.1.1",
    "ignoreVerification": "false",
    "requiresBigipVersionMin": "11.0.0",
    "totalSigningStatus": "not-all-signed",
    "verificationStatus": "none",
    "actionsReference": {
        "link": "https://localhost/mgmt/tm/sys/application/template/~Common~appsvcs_integration_v2.0.004/actions?ver=12.1.1",
        "isSubcollection": true
    }
```

2.4.3 Lab 2.3: Create iApp Deployments using the REST API



Now that the App Services iApp template is installed, we can deploy a new Layer 4-7 Service. The service in this lab will go through different iterations, we'll start with **Creating** a Basic HTTP Service, show **Modifying** the service by changing the node state, and then **Delete** the whole service. Once we've seen this first **Mutation**, we'll introduce some more complex deployments options with iRules, Custom Profiles, Certificates, and an ASM Policy.

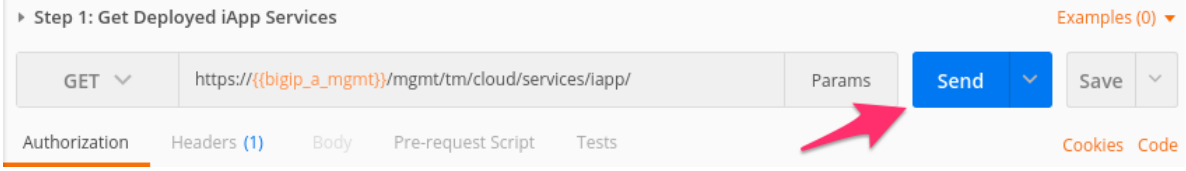
Note: This lab work will be performed from `Lab 2.3 - Create iApp Deployments using the REST API` folder in the Postman Collection


F5 Programmability: Class 1
111 requests
Lab 1.2 - API Authentication & 'example' Templates
Lab 1.3 - Review/Set Device Settings
Lab 1.4 - Basic Network Connectivity
Lab 1.5 - Building Imperative Workflows
Lab 1.6 - Build a Cluster
Lab 1.7 - Build a Basic LTM Config using Transactions
Lab 2.2 - Deploying iApp Templates on BIG-IP
Lab 2.3 - Create iApp Deployments using the REST API
GET Step 1: Get Deployed iApp Services
POST Step 2: Deploy Service - HTTP
PUT Step 3: Modify Service - HTTP
DEL Step 4: Delete Service - HTTP

Task 1 - View Deployed Services

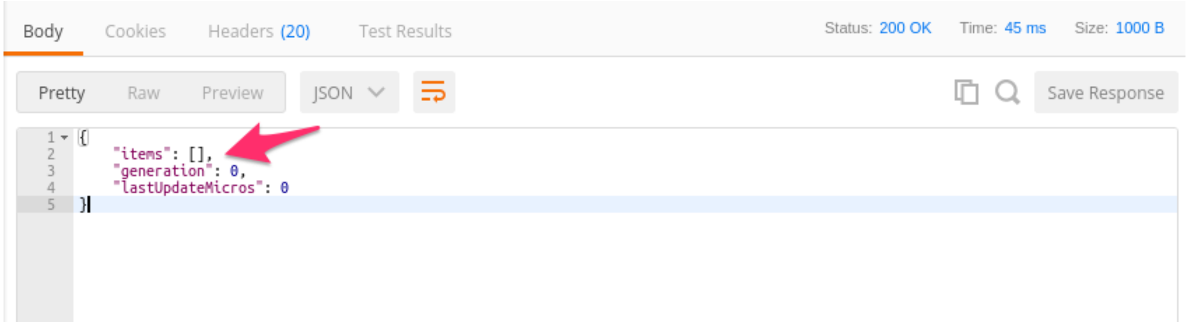
Perform the following steps to complete this task:

1. *Send* the `Step 1: Get Deployed iApp Services` request to view current iApp deployments on the BIG-IP device:


Step 1: Get Deployed iApp Services

GET https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/

2. Review the JSON Response *Body*. The BIG-IP device does not have any iApp deployments. As a result the `items` array is empty (`[]`):

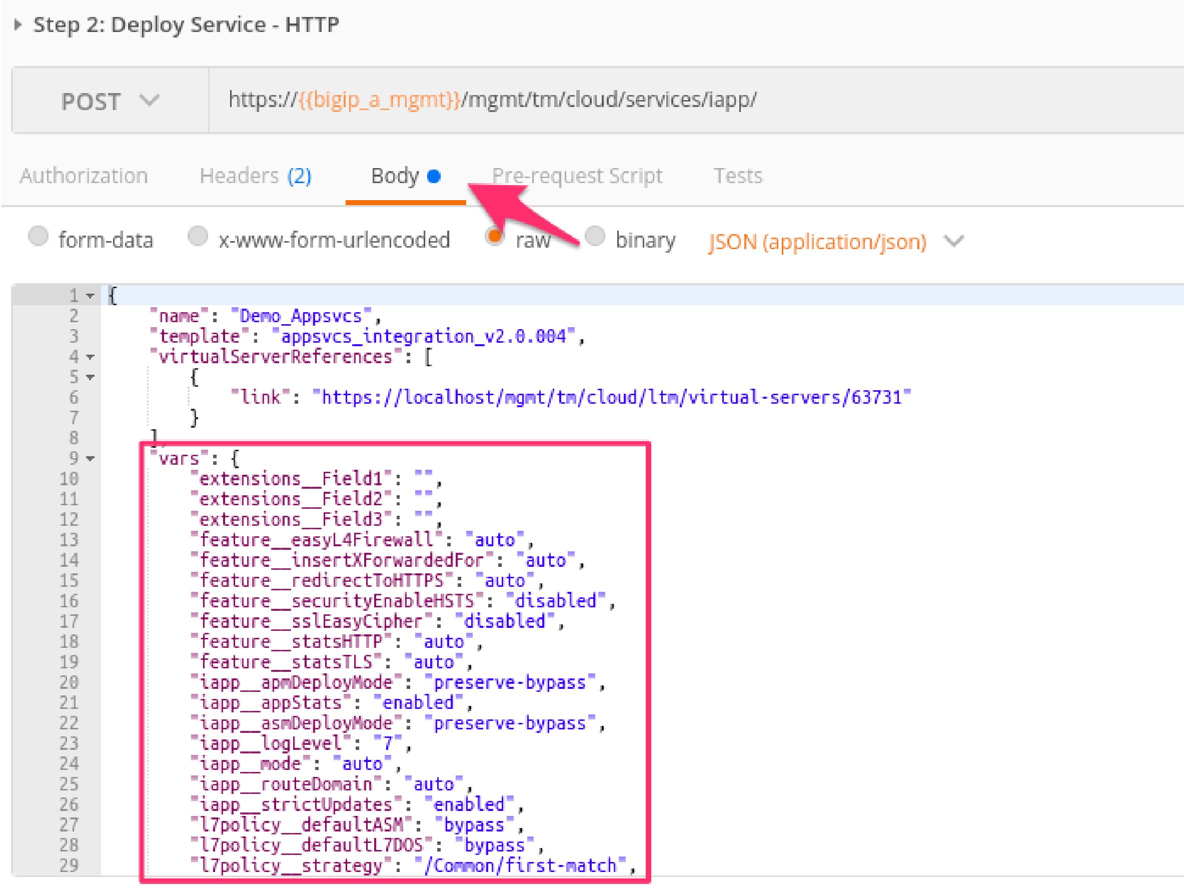


```
{
    "items": [],
    "generation": 0,
    "lastUpdateMicros": 0
}
```

Task 2 - Deploy Basic HTTP Service

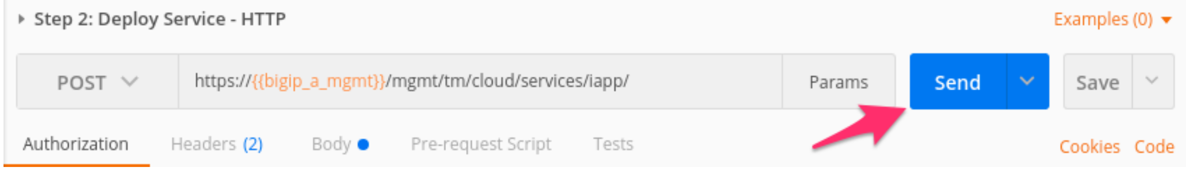
Perform the following steps to complete this task:

1. Click `Step 2: Deploy Service - HTTP`. Review the **Request** JSON *Body*. The JSON body of the POST contains the input for the iApp template to drive the deployment of the service.

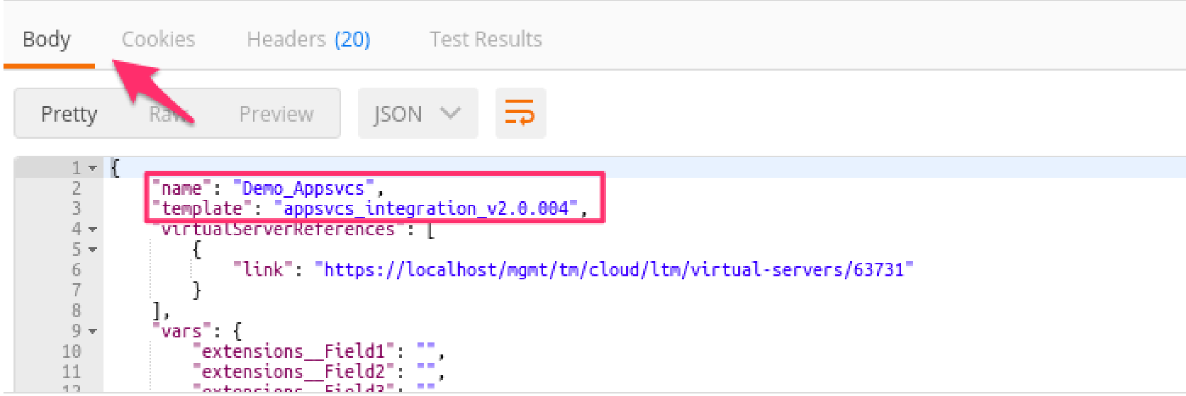


```
{
    "name": "Demo_Appsvcs",
    "template": "appsvcs_integration_v2.0.004",
    "virtualServerReferences": [
        {
            "link": "https://localhost/mgmt/tm/cloud/ltm/virtual-servers/63731"
        }
    ],
    "vars": {
        "extensions__Field1": "",
        "extensions__Field2": "",
        "extensions__Field3": "",
        "feature__easyL4Firewall": "auto",
        "feature__insertXForwardedFor": "auto",
        "feature__redirectToHTTPS": "auto",
        "feature__securityEnableHSTS": "disabled",
        "feature__sslEasyCipher": "disabled",
        "feature__statsHTTP": "auto",
        "feature__statsTLS": "auto",
        "iapp__apmDeployMode": "preserve-bypass",
        "iapp__appStats": "enabled",
        "iapp__asmDeployMode": "preserve-bypass",
        "iapp__logLevel": "7",
        "iapp__mode": "auto",
        "iapp__routeDomain": "auto",
        "iapp__strictUpdates": "enabled",
        "l7policy__defaultASM": "bypass",
        "l7policy__defaultL7DOS": "bypass",
        "l7policy__strategy": "/Common/first-match",
```

2. Click the *Send* button to **Create** a Basic HTTP Service:



In this task, we deployed our first service. Review the **Response** JSON *Body* to verify if the Service has been deployed.



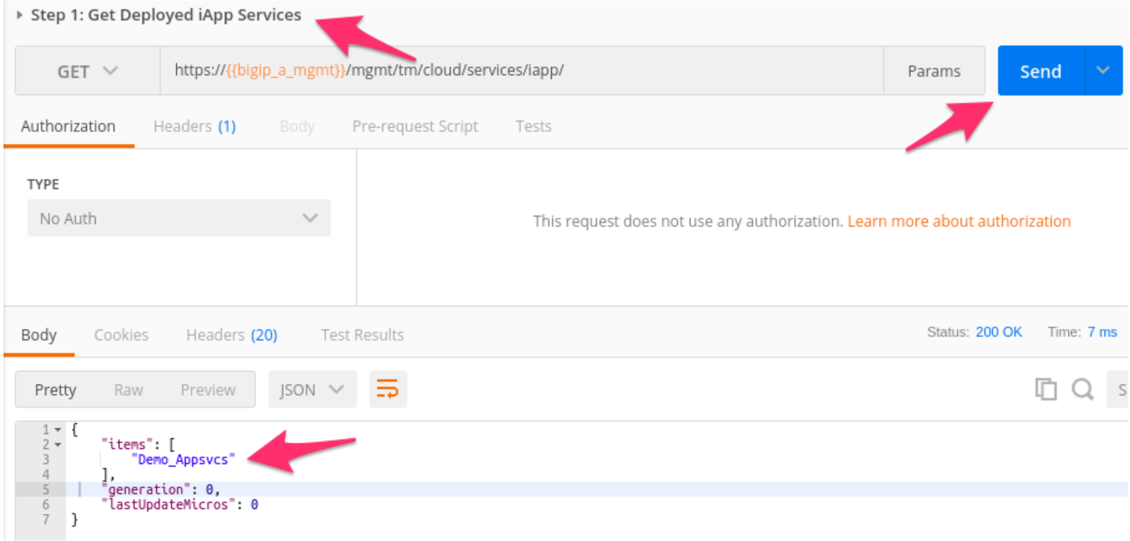
```
{
    "name": "Demo_Appsvcs",
    "template": "appsvcs_integration_v2.0.004",
    "virtualServerReferences": [
        {
            "link": "https://localhost/mgmt/tm/cloud/ltm/virtual-servers/63731"
        }
    ],
    "vars": {
        "extensions__Field1": "",
        "extensions__Field2": "",
        "extensions__Field3": ""
```

Note: We've just progressed into a **Declarative** instantiation, by defining the end state and letting the BIG-IP handle the order of operations and configuration of the specific objects. By doing this, we have drastically reduced the **Domain Specific Knowledge** requirement to interact with the device. In the next module, we will combine this concept with **Abstraction** to further simplify the interface.

---

3. Now that the service has been deployed, let's review the BIG-IP configuration. You can review via REST by sending the `Step 1: Get Deployed iApp Services` request again, or you can login to the BIG-IP A GUI and see the service deployment via TMUI:

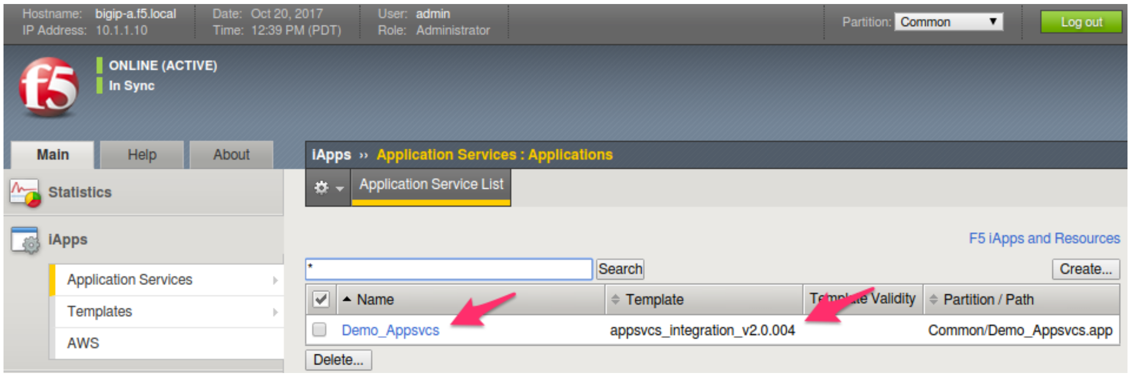
   - **REST**: *Send* `Step 1: Get Deployed iApp Services` request:



This request does not use any authorization. Learn more about authorization

```
{
    "items": [
        "Demo_Appsvcs"
    ],
    "generation": 0,
    "lastUpdateMicros": 0
}
```

   - **TMUI GUI**: *iApps* → *Application Services* → *Applications*



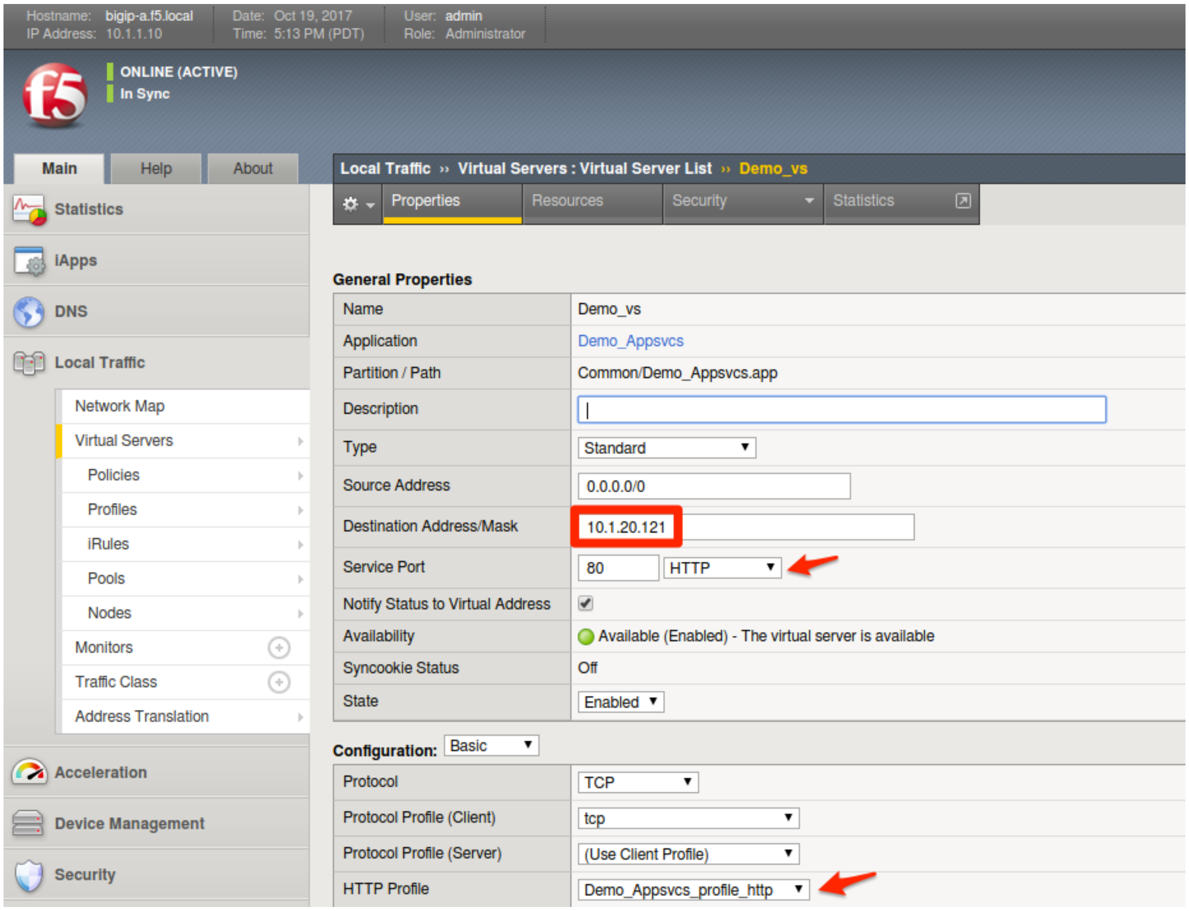
Hostname: bigip-a.f5.local
IP Address: 10.1.1.10
Date: Oct 20, 2017
Time: 12:39 PM (PDT)
User: admin
Role: Administrator
Partition: Common

ONLINE (ACTIVE)
In Sync

iApps » Application Services : Applications

	Name	Template	Template Validity	Partition / Path
□	Demo_Appsvcs	appsvcs_integration_v2.0.004		Common/Demo_Appsvcs.app

4. From the TMUI GUI, examine the Virtual Server that was created from this deployment by clicking *Local Traffic* → *Virtual Servers* → *Virtual Server List* → *Demo_vs*. The configuration is simple, but it does contain the key components for an HTTP service (Listener, HTTP Profile, Monitor, Pool, and Pool Members):



Hostname: bigip-a.f5.local
IP Address: 10.1.1.10
Date: Oct 19, 2017
Time: 5:13 PM (PDT)
User: admin
Role: Administrator

ONLINE (ACTIVE)
In Sync

Local Traffic » Virtual Servers : Virtual Server List » Demo_vs

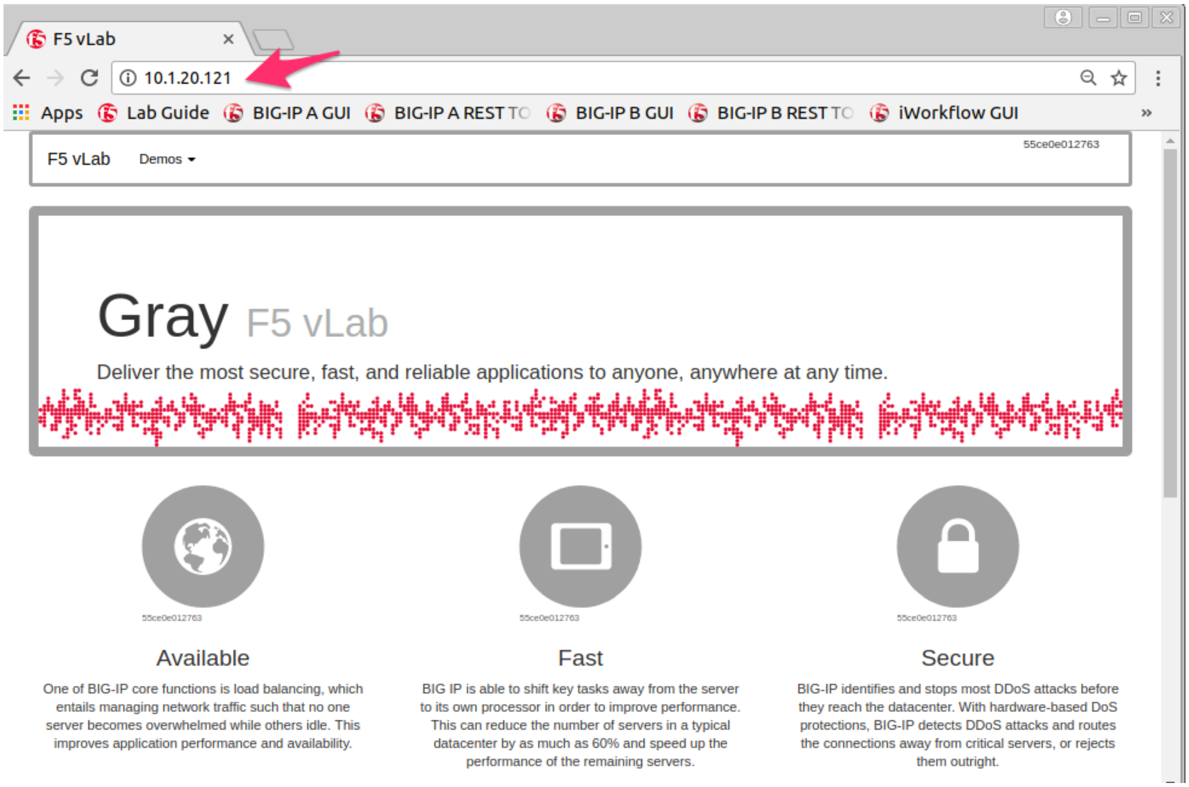
General Properties

Name	Demo_vs
Application	Demo_Appsvcs
Partition / Path	Common/Demo_Appsvcs.app
Description	
Type	Standard
Source Address	0.0.0.0/0
Destination Address/Mask	10.1.20.121
Service Port	80 HTTP
Notify Status to Virtual Address	☑
Availability	● Available (Enabled) - The virtual server is available
Syncookie Status	Off
State	Enabled

Configuration: Basic

Protocol	TCP
Protocol Profile (Client)	tcp
Protocol Profile (Server)	(Use Client Profile)
HTTP Profile	Demo_Appsvcs_profile_http

5. The service is available and active, you can connect to the Virtual Server using Chrome at `http://10.1.20.121` and examine its response:



Gray F5 vLab

Deliver the most secure, fast, and reliable applications to anyone, anywhere at any time.

Available

One of BIG-IP core functions is load balancing, which entails managing network traffic such that no one server becomes overwhelmed while others idle. This improves application performance and availability.

Fast

BIG IP is able to shift key tasks away from the server to its own processor in order to improve performance. This can reduce the number of servers in a typical datacenter by as much as 60% and speed up the performance of the remaining servers.

Secure

BIG-IP identifies and stops most DDoS attacks before they reach the datacenter. With hardware-based DoS protections, BIG-IP detects DDoS attacks and routes the connections away from critical servers, or rejects them outright.

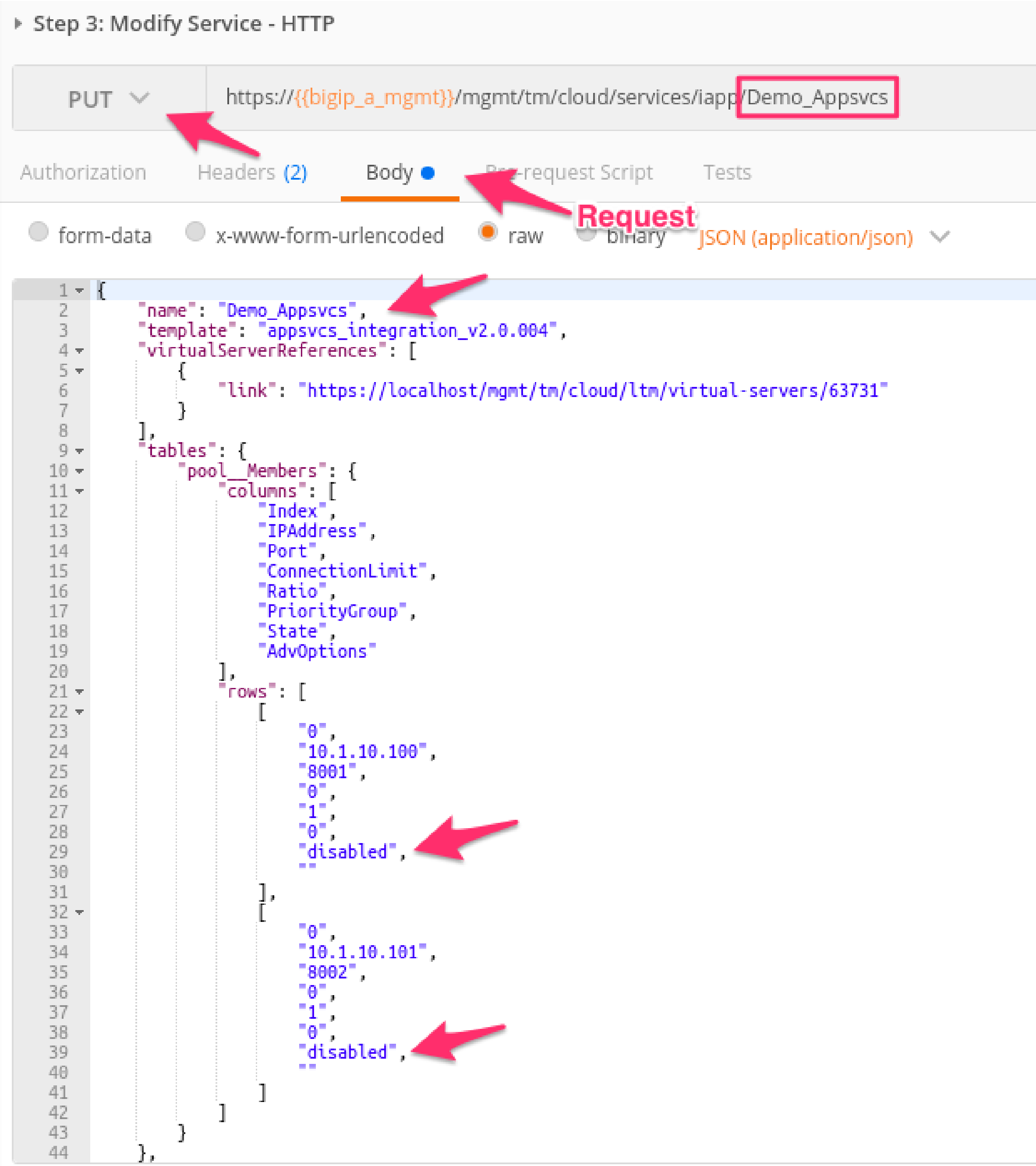
Note: The colors of the text, images, and borders may vary based on which back-end server was selected during the load balancing process.

Task 3 - Modify our Deployed Service

In this task, we will modify the existing service. We will disable all pool members and bring the service down.

Perform the following steps to complete this task:

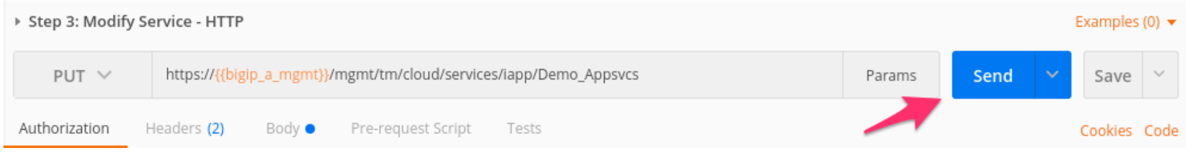
1. Click on `Step 3:  Modify Service - HTTP`. Review the **Request** URL and JSON *Body*. Notice that we specified the **Resource** URL for our deployment. Modifying or *Redeploying* a service is handled by sending **only** the updated JSON to the specific Resource (our service) using a `PUT` request method. We set the state of the pool members to `disabled` which forces the service offline.



https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/Demo_Appsvcs

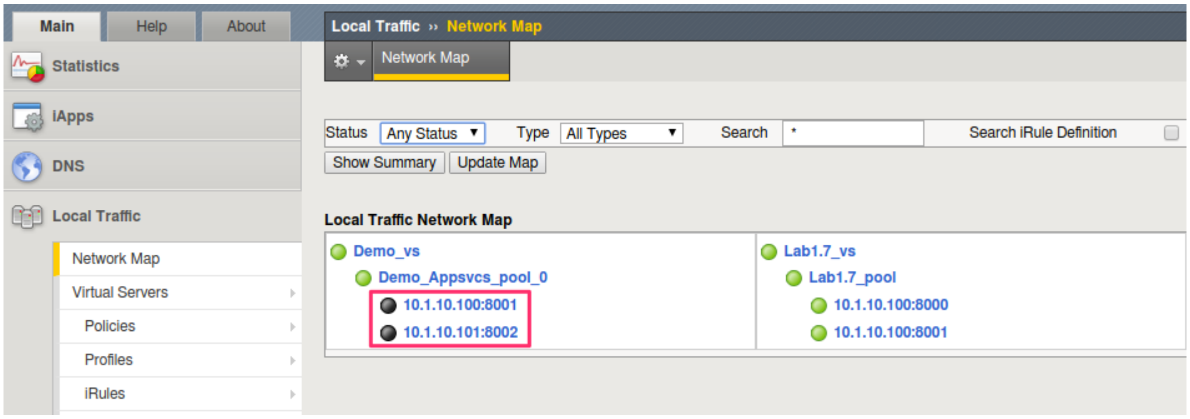
```
{
    "name": "Demo_Appsvcs",
    "template": "appsvcs_integration_v2.0.004",
    "virtualServerReferences": [
        {
            "link": "https://localhost/mgmt/tm/cloud/ltm/virtual-servers/63731"
        }
    ],
    "tables": {
        "pool__Members": {
            "columns": [
                "Index",
                "IPAddress",
                "Port",
                "ConnectionLimit",
                "Ratio",
                "PriorityGroup",
                "State",
                "AdvOptions"
            ],
            "rows": [
                [
                    "0",
                    "10.1.10.100",
                    "8001",
                    "0",
                    "1",
                    "0",
                    "disabled",
                    ""
                ],
                [
                    "0",
                    "10.1.10.101",
                    "8002",
                    "0",
                    "1",
                    "0",
                    "disabled",
                    ""
                ]
            ]
        }
    },
```

2. Click the *Send* button to **Modify** the previously deployed Basic HTTP Service:



https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/Demo_Appsvcs

3. In the BIG-IP GUI click *Local Traffic* → *Network Map* to view the new state of the Pool Members (Black indicators reflect the disabled state). The state has been updated to reflect the state we declared in our call. The Virtual Server is no longer passing traffic at `http://10.1.20.121` because all the Members in the Pool are disabled:



Status: Any Status Type: All Types Search: *

Local Traffic Network Map

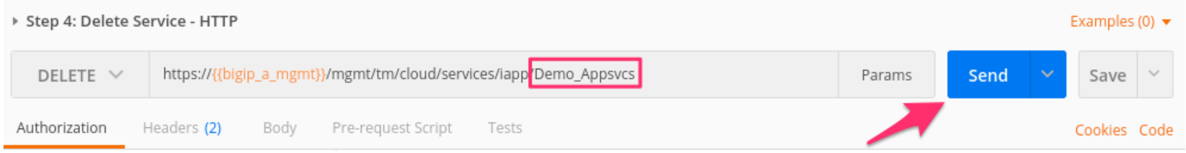
- Demo_vs
  - Demo_Appsvcs_pool_0
    - 10.1.10.100:8001
    - 10.1.10.101:8002
- Lab1.7_vs
  - Lab1.7_pool
    - 10.1.10.100:8000
    - 10.1.10.100:8001

Task 4 - Delete our Deployed Service

The lifecycle of a service also includes the service removal. We will now delete an existing service.

Perform the following steps to complete this task:

1. *Send* the `Step 4: Delete Service - HTTP` request to **Delete** the previously deployed Basic HTTP Service:



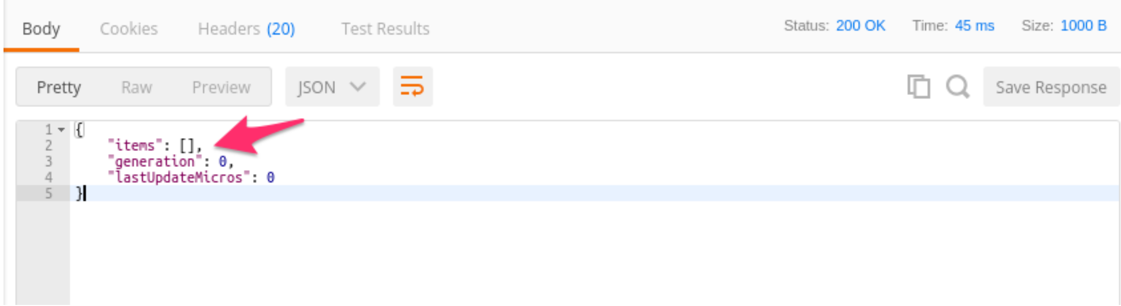
```
DELETE  https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/Demo_Appsvcs
```

2. Like modification, the deletion of a service is performed on the **Resource** URL. When we created the service we defined a Declarative state to the iApp template. The template then created the configuration and all the associated objects. With a `DELETE` request, the BIG-IP will processes the removal of all objects linked to the ASO in the correct order. This is crucial to Application Lifecycle Management as it provides a mechanism to make sure all parts of the service are removed successfully.

Note: There is no JSON body to a `DELETE` call, as the HTTP Method is defining the action.

Now that the service has been deleted, let's review the BIG-IP configuration. You can review via REST by sending the `Step 1: Get Deployed iApp Services` request again, or you can login to the BIG-IP A GUI and see the service deployment via TMUI:

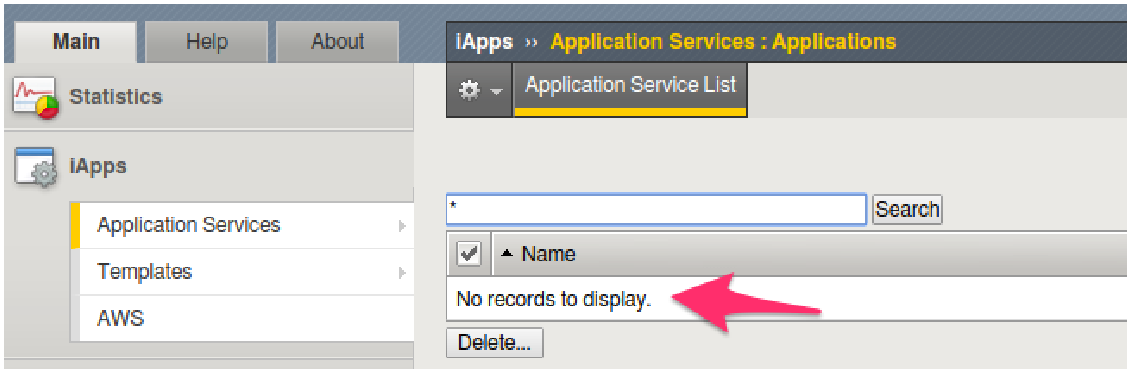
- **REST**: *Send* `Step 1: Get Deployed iApp Services` request:



Status: 200 OK Time: 45 ms Size: 1000 B

```
{
    "items": [],
    "generation": 0,
    "lastUpdateMicros": 0
}
```

- **TMUI GUI**: *iApps → Application Services → Applications*

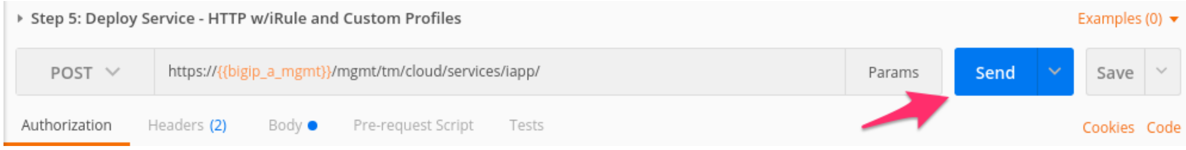


No records to display.

Task 5 - Deploy an HTTP Service with Custom created Profile and a referenced iRule

Perform the following steps to complete this task:

1. *Send* the `Step 5: Deploy Service - HTTP w/ iRule and Custom Profiles` request to deploy an HTTP Service with Custom Profiles and an iRule:



Step 5: Deploy Service - HTTP w/iRule and Custom Profiles

POST https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/

2. The App Services iApp can *Create* or *Reference* various objects. In this deployment we perform two actions:

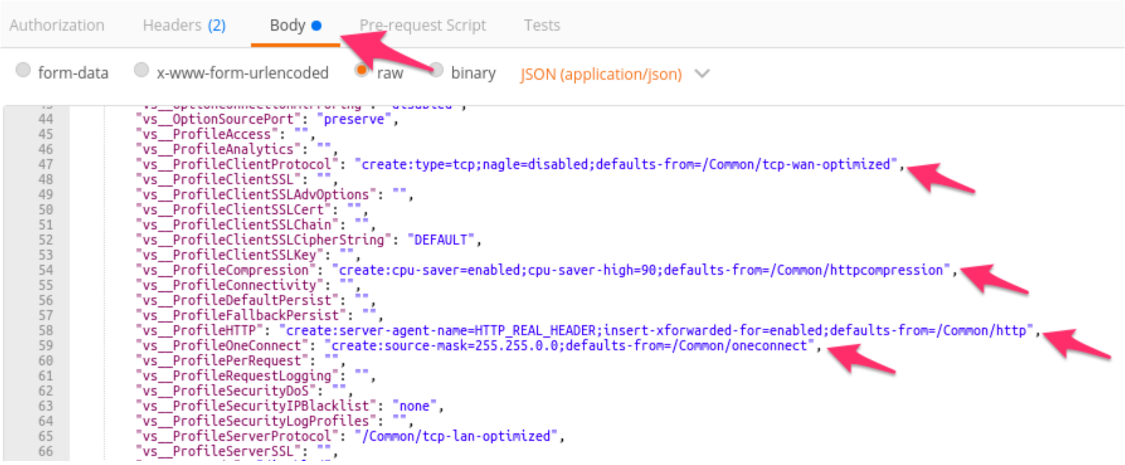
   (a) Create custom profiles on the BIG-IP device with various options specified. These profiles do not exist on the BIG-IP but are created dynamically during the deployment.

   (b) Create an iRule on the BIG-IP device by using a **URL Reference**. The App Services iApp downloads the iRule resource from the URL and then creates a new iRule object on the system. The iRule object is then automatically linked to the Virtual Server

   > **Warning:** When using URL references, it is important to properly secure the repository which hosts the resource(s). The example in this lab uses a publicly readable repository, however, most environments should use a private repository with appropriate access control.

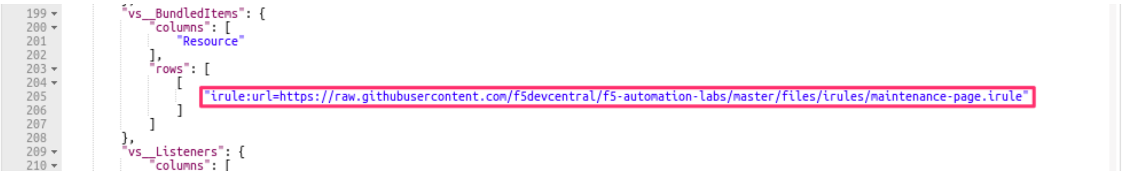
3. Review the **Request** JSON *Body* to see how the desired outcomes above were declared:

   - **Custom Profiles:**



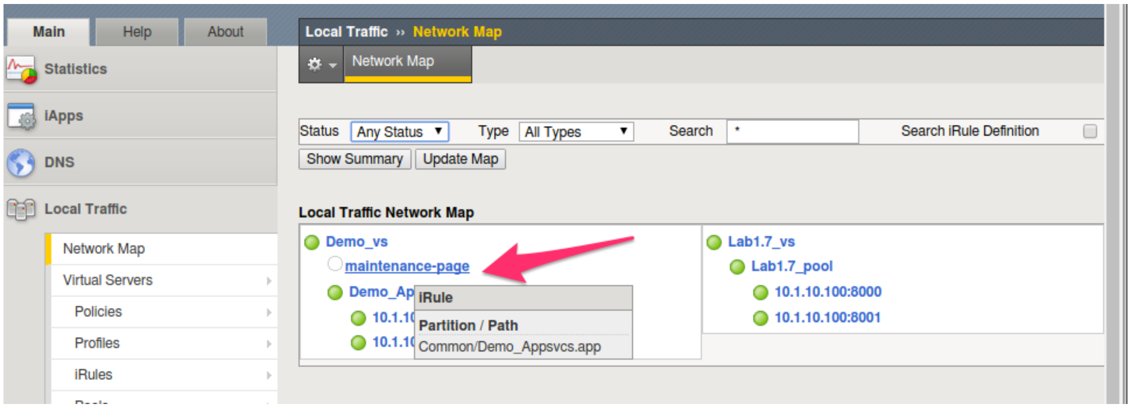
```
        "vs__OptionSourcePort": "preserve",
        "vs__ProfileAccess": "",
        "vs__ProfileAnalytics": "",
        "vs__ProfileClientProtocol": "create:type=tcp;nagle=disabled;defaults-from=/Common/tcp-wan-optimized",
        "vs__ProfileClientSSL": "",
        "vs__ProfileClientSSLAdvOptions": "",
        "vs__ProfileClientSSLCert": "",
        "vs__ProfileClientSSLChain": "",
        "vs__ProfileClientSSLCipherString": "DEFAULT",
        "vs__ProfileClientSSLKey": "",
        "vs__ProfileCompression": "create:cpu-saver=enabled;cpu-saver-high=90;defaults-from=/Common/httpcompression",
        "vs__ProfileConnectivity": "",
        "vs__ProfileDefaultPersist": "",
        "vs__ProfileFallbackPersist": "",
        "vs__ProfileHTTP": "create:server-agent-name=HTTP_REAL_HEADER;insert-xforwarded-for=enabled;defaults-from=/Common/http",
        "vs__ProfileOneConnect": "create:source-mask=255.255.0.0;defaults-from=/Common/oneconnect",
        "vs__ProfilePerRequest": "",
        "vs__ProfileRequestLogging": "",
        "vs__ProfileSecurityDoS": "",
        "vs__ProfileSecurityIPBlacklist": "none",
        "vs__ProfileSecurityLogProfiles": "",
        "vs__ProfileServerProtocol": "/Common/tcp-lan-optimized",
        "vs__ProfileServerSSL": "",
```

- **URL Referenced iRule:**

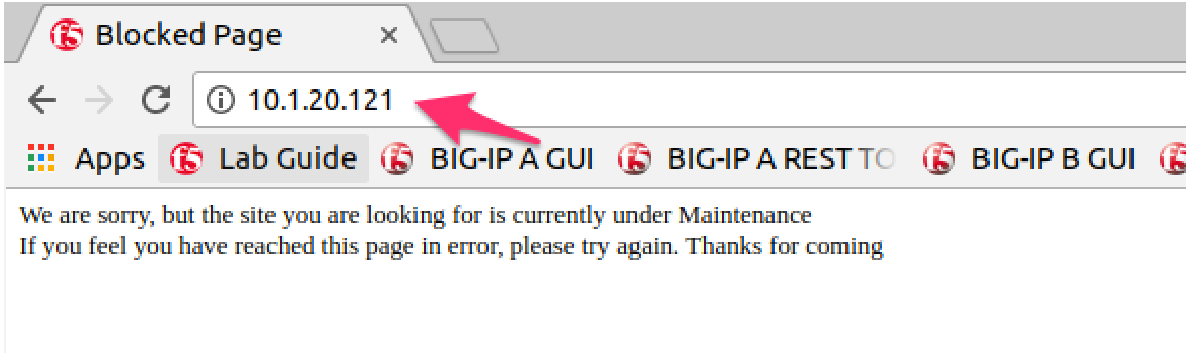


```
    "vs__BundledItems": {
        "columns": [
            "Resource"
        ],
        "rows": [
            [
                "irule:url=https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/files/irules/maintenance-page.irule"
            ]
        ]
    },
    "vs__Listeners": {
        "columns": [
```

- **iRule linked to Virtual Server:** (*Local Traffic* → *Network Map*)



4. Open Chrome and connect to the Virtual Server at `http://10.1.20.121`. The iRule that was attached to the service contains an `HTTP_RESPOND` event, which responds with a simple Maintenance Page.

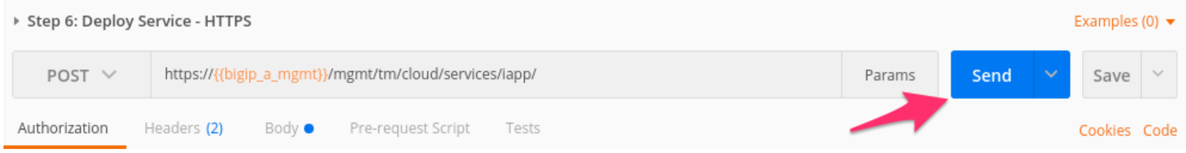


We are sorry, but the site you are looking for is currently under Maintenance
If you feel you have reached this page in error, please try again. Thanks for coming

Task 6 - Deploy an HTTPS Service

Perform the following steps to complete this task:

1. *Send* the `Step 6: Deploy Service - HTTPS` request to deploy an HTTPS Service using **URL Resources** for the SSL/TLS Key, Certificate and Certificate Bundle.



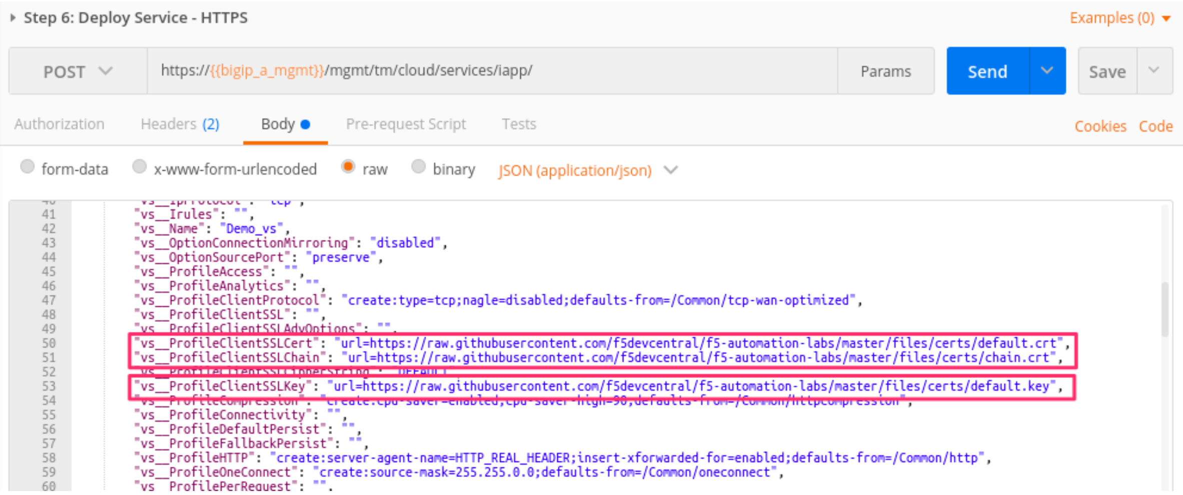
2. iApps are a Declarative interface, allowing us to modify deployment without the need to delete it (this also means we can re-name objects **if** we needed too). For this service we will:

   - Use the same custom profiles
   - Remove the iRule
   - Change the Listener port to `443` (HTTPS)
   - Use URL Resources to obtain the SSL/TLS Key, Certificate and Certificate Bundle

   > **Warning:** When using URL references, it is important to properly secure the repository which hosts the resource(s). The example in this lab uses a publicly readable repository. However, most environments should use a private repository with appropriate access control.

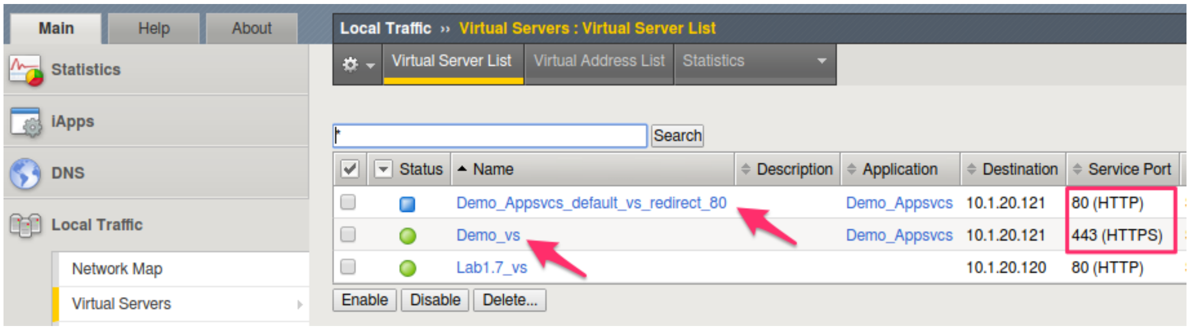
   - Create and apply a Client SSL Profile

3. Review the **Request** JSON *Body* to see how the desired outcomes above were declared:



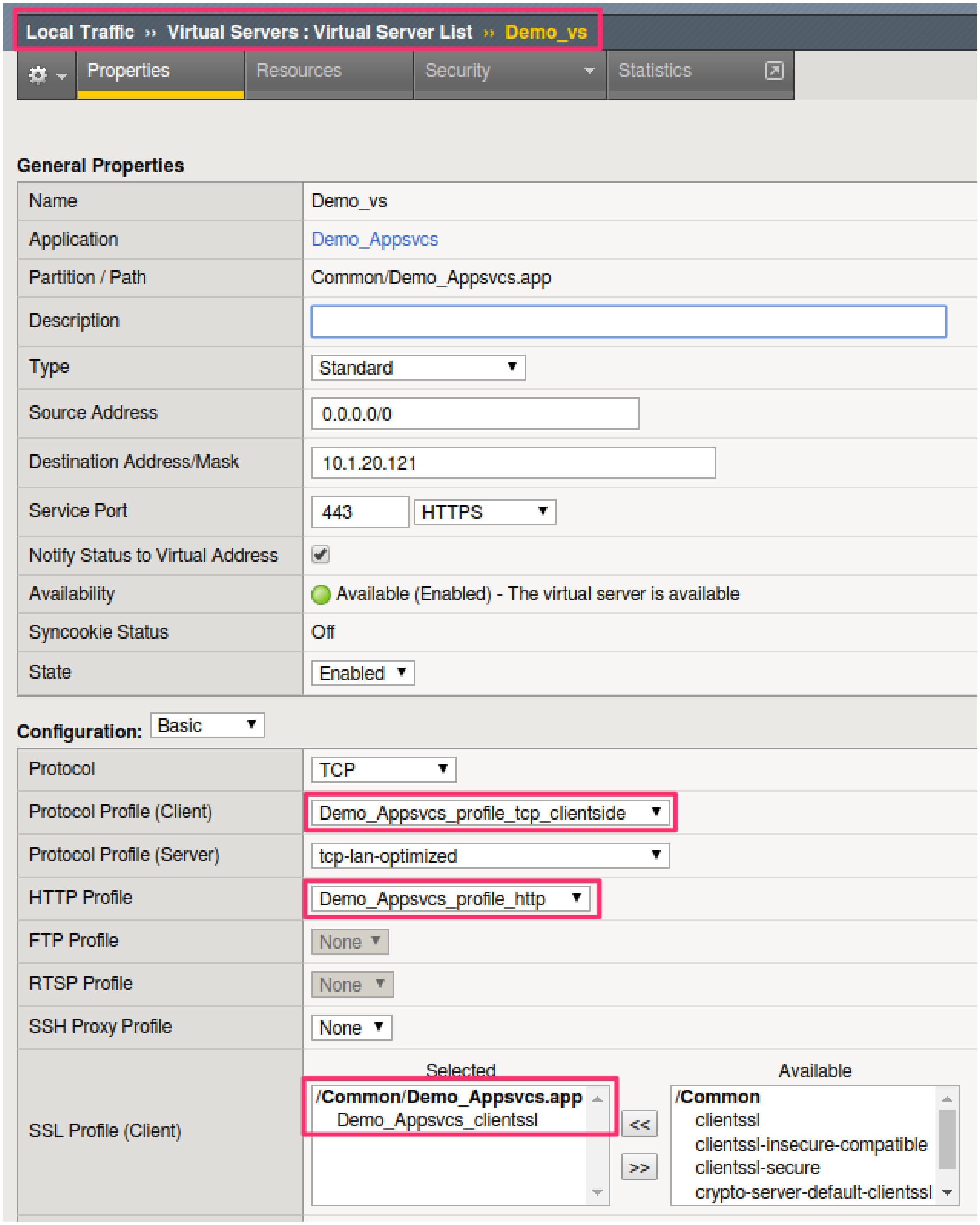
```
        "vs__IpProtocol": "tcp",
        "vs__Irules": "",
        "vs__Name": "Demo_vs",
        "vs__OptionConnectionMirroring": "disabled",
        "vs__OptionSourcePort": "preserve",
        "vs__ProfileAccess": "",
        "vs__ProfileAnalytics": "",
        "vs__ProfileClientProtocol": "create:type=tcp;nagle=disabled;defaults-from=/Common/tcp-wan-optimized",
        "vs__ProfileClientSSL": "",
        "vs__ProfileClientSSLAdvOptions": "",
        "vs__ProfileClientSSLCert": "url=https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/files/certs/default.crt",
        "vs__ProfileClientSSLChain": "url=https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/files/certs/chain.crt",
        "vs__ProfileClientSSLCipherString": "DEFAULT",
        "vs__ProfileClientSSLKey": "url=https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/files/certs/default.key",
        "vs__ProfileCompression": "create:cpu-saver=enabled;cpu-saver-high=90;defaults-from=/Common/httpcompression",
        "vs__ProfileConnectivity": "",
        "vs__ProfileDefaultPersist": "",
        "vs__ProfileFallbackPersist": "",
        "vs__ProfileHTTP": "create:server-agent-name=HTTP_REAL_HEADER;insert-xforwarded-for=enabled;defaults-from=/Common/http",
        "vs__ProfileOneConnect": "create:source-mask=255.255.0.0;defaults-from=/Common/oneconnect",
        "vs__ProfilePerRequest": "",
```

4. Review the configured Virtual Servers in the TMUI GUI. The App Services iApp created a new Virtual Server to redirect `TCP/80` traffic to `TCP/443` and reconfigured the Virtual Server to listen on `TCP/443`



☑	Status	Name	Description	Application	Destination	Service Port
□	■	Demo_Appsvcs_default_vs_redirect_80		Demo_Appsvcs	10.1.20.121	80 (HTTP)
□	●	Demo_vs		Demo_Appsvcs	10.1.20.121	443 (HTTPS)
□	●	Lab1.7_vs			10.1.20.120	80 (HTTP)

5. The configuration of the Virtual Server now uses an SSL Client profile containing our imported SSL Resources. The deployment is now providing SSL Offload for the backend compute nodes.


Local Traffic ›› Virtual Servers : Virtual Server List ›› Demo_vs
Properties
Resources
Security
Statistics

General Properties

Name	Demo_vs
Application	Demo_Appsvcs
Partition / Path	Common/Demo_Appsvcs.app
Description	
Type	Standard
Source Address	0.0.0.0/0
Destination Address/Mask	10.1.20.121
Service Port	443 HTTPS
Notify Status to Virtual Address	☑
Availability	● Available (Enabled) - The virtual server is available
Syncookie Status	Off
State	Enabled

Configuration: Basic

Protocol	TCP
Protocol Profile (Client)	Demo_Appsvcs_profile_tcp_clientside
Protocol Profile (Server)	tcp-lan-optimized
HTTP Profile	Demo_Appsvcs_profile_http
FTP Profile	None
RTSP Profile	None
SSH Proxy Profile	None
SSL Profile (Client)	Selected: /Common/Demo_Appsvcs.app Demo_Appsvcs_clientssl Available: /Common clientssl clientssl-insecure-compatible clientssl-secure crypto-server-default-clientssl

6. Open Chrome and access the service with `http://10.1.20.121`. It should redirect you to `https://10.1.20.121`.

Note: We are using self signed certificates in the lab so an SSL warning will be shown.

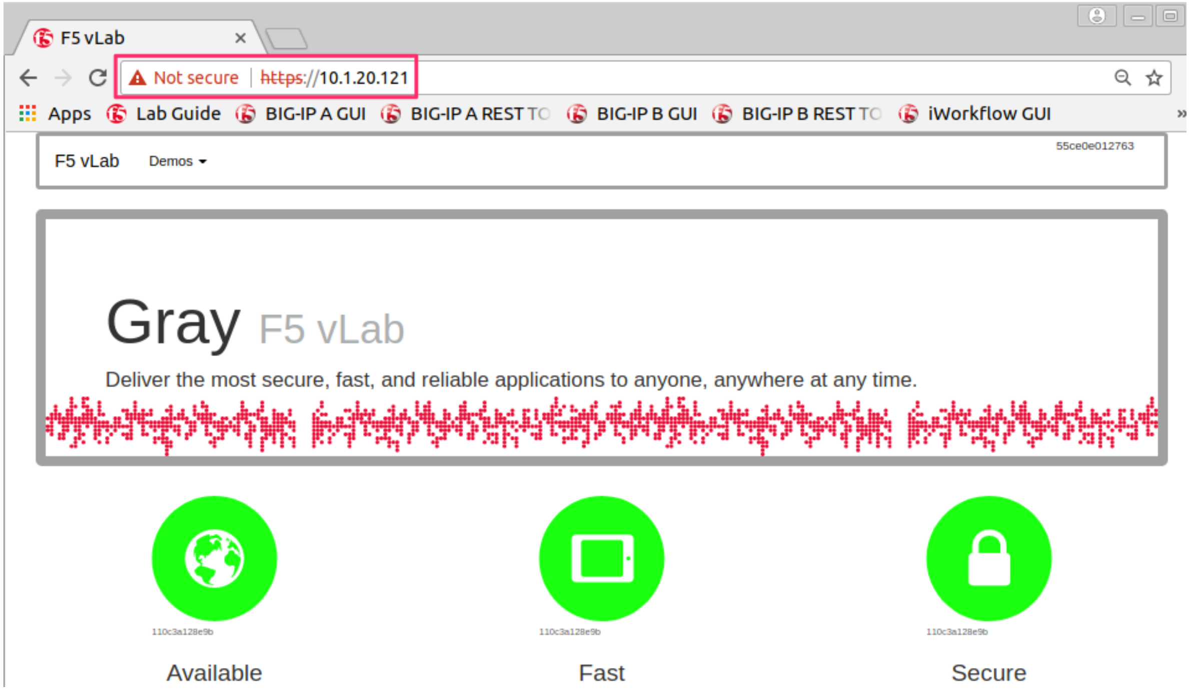
Warning: When you open this page you may continue to keep the Maintence Page from the previous Task. This occurs because of two reasons:

(a) Chrome keeps HTTP connections open in the background to improve network performance

(b) BIG-IP maintains a fully versioned configuration internally. Stateful connections, like HTTP, are then pinned to a specific version of the configuration for the lifetime of the connection.

As a result, because Chrome has not closed the actual TCP connection, BIG-IP still processes traffic with the configuration that was present when the connection was originally created.

You can open an Incognito Chrome Window (Ctrl-Shift-N) and try to connect to `http://10.1.20.121` again. The connection in the Incognito window should behave as expected because it's a new connection and therefore uses the most recent configuration.

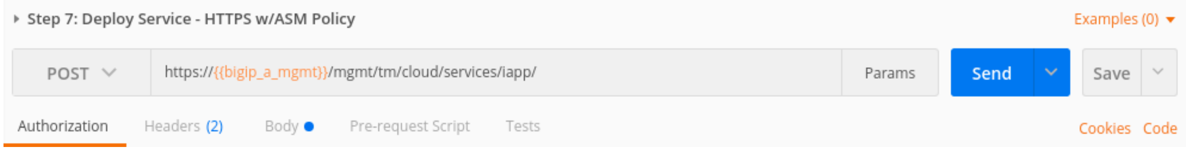


Task 7 - Deploy an HTTPS Service with an Web Application Firewall Policy

Another advantage of Service Deployment using iApp Templates is that they can deploy advanced Layer 4-7 services from various F5 modules. In this task we will deploy a service that includes a Web Application Firewall policy with the base HTTPS offload and load balancing features.

Perform the following steps to complete this task:

1. *Send* the `Step 7: Deploy Service - HTTPS w/ WAF Policy` request to deploy an HTTPS Service using **URL Resources** for a Web Application Firewall policy that will be used with the Application Security Manager (ASM) module.

   Step 7: Deploy Service - HTTPS w/ASM Policy

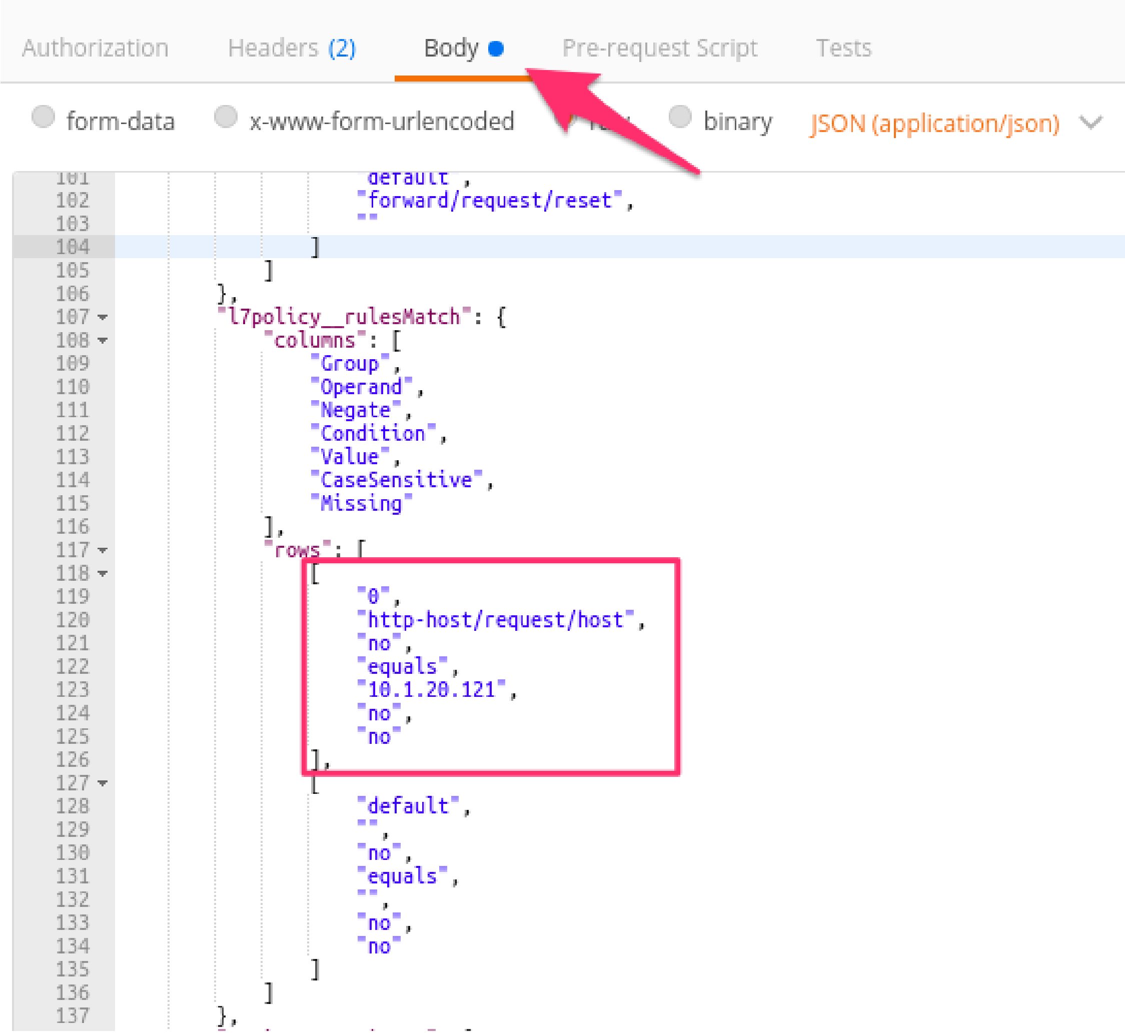
   POST https://{{bigip_a_mgmt}}/mgmt/tm/cloud/services/iapp/

2. This final iApp deployment will build upon our service by having the iApp load a WAF policy Resource from our repository. The App Services iApp will then create a Layer 7 Traffic Policy and apply it to the Virtual Server.

   This deployment recognizes the need for Security from the beginning of the application lifecycle. It lays the groundwork for **Continuous Improvement** by having the policy reside in a repository. It allows us to treat resources as code leading to an Infrastructure as Code (IaC) methodology. As the policy is updated in the repository additional automation and orchestration can be enabled to deploy the policy into the environment. The result is an ability to rapidly build, test and iterate Layer 7 security policies and guarantee deployment into the environment.

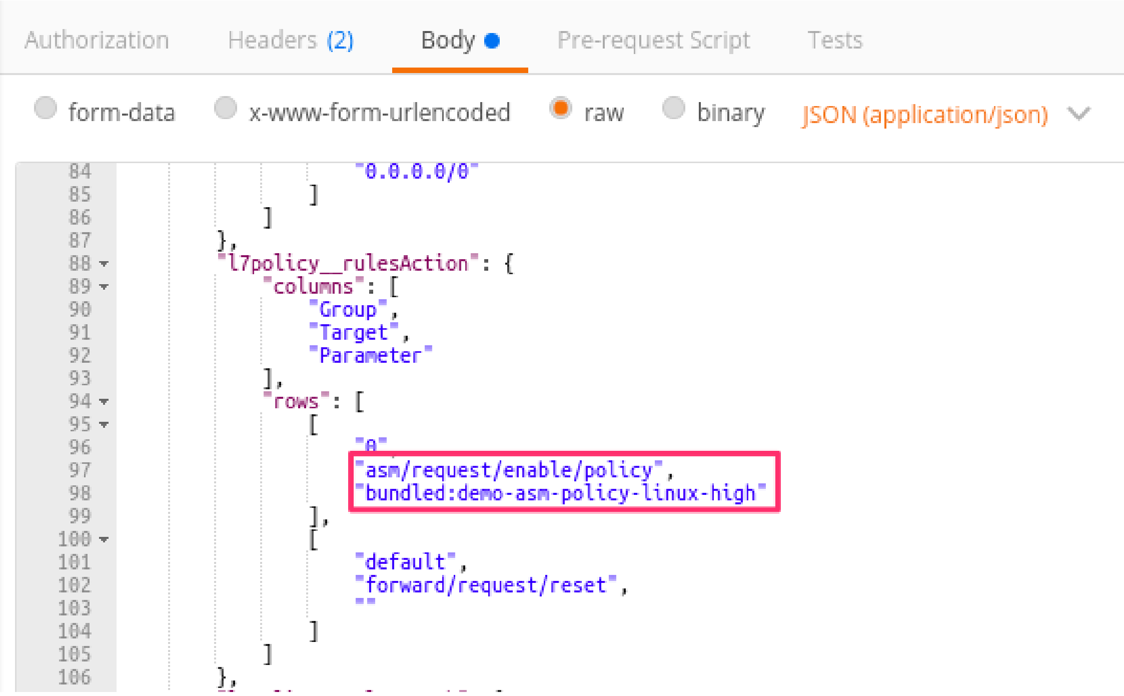
3. Review the **Request** JSON *Body* to see how the desired outcomes above were declared:

   - **Layer 7 Policy Rules:**



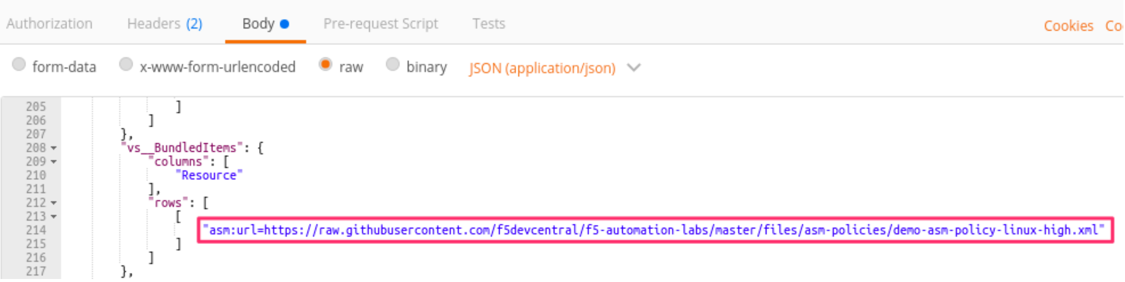
```
                "default",
                "forward/request/reset",
                ""
            ]
        ]
    },
    "l7policy__rulesMatch": {
        "columns": [
            "Group",
            "Operand",
            "Negate",
            "Condition",
            "Value",
            "CaseSensitive",
            "Missing"
        ],
        "rows": [
            [
                "0",
                "http-host/request/host",
                "no",
                "equals",
                "10.1.20.121",
                "no",
                "no"
            ],
            [
                "default",
                "",
                "no",
                "equals",
                "",
                "no",
                "no"
            ]
        ]
    },
```

   - **Layer 7 Policy Actions:**



```
                "0.0.0.0/0"
            ]
        ]
    },
    "l7policy__rulesAction": {
        "columns": [
            "Group",
            "Target",
            "Parameter"
        ],
        "rows": [
            [
                "0",
                "asm/request/enable/policy",
                "bundled:demo-asm-policy-linux-high"
            ],
            [
                "default",
                "forward/request/reset",
                ""
            ]
        ]
    },
```

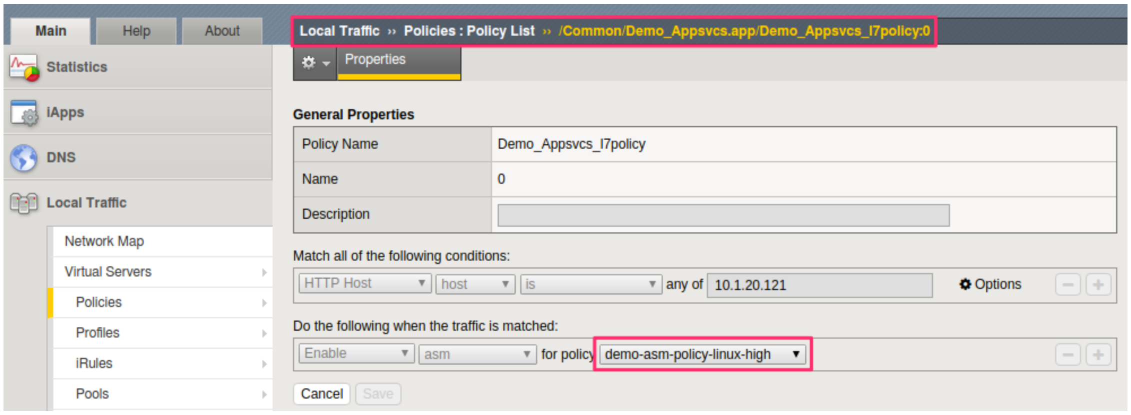
- **ASM Policy URL:**



```
        ]
    ]
},
"vs__BundledItems": {
    "columns": [
        "Resource"
    ],
    "rows": [
        [
            "asm:url=https://raw.githubusercontent.com/f5devcentral/f5-automation-labs/master/files/asm-policies/demo-asm-policy-linux-high.xml"
        ]
    ]
},
```

4. In the TMUI GUI, we can see the Layer 7 policy applied to the Virtual Server. In the *Application Security*, we can see the details of the policy which was dynamically fetched, applied, and set to Blocking mode.

   - **Layer 7 Policy:**



Local Traffic ›› Policies : Policy List ›› /Common/Demo_Appsvcs.app/Demo_Appsvcs_l7policy:0

General Properties

Policy Name	Demo_Appsvcs_l7policy
Name	0
Description	

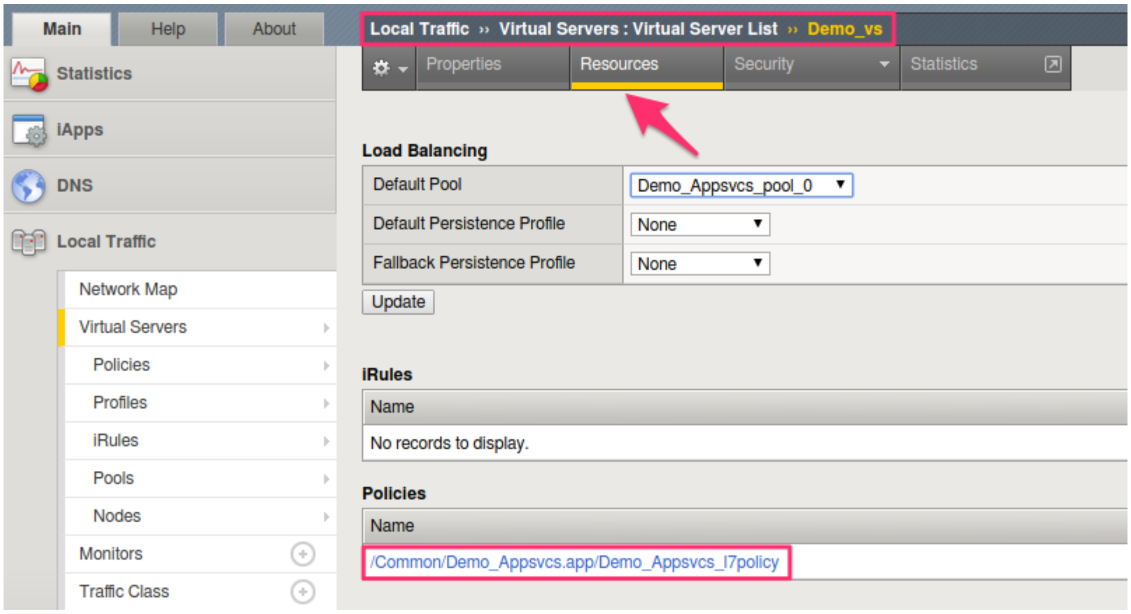
Match all of the following conditions:

HTTP Host ▾ host ▾ is ▾ any of 10.1.20.121

Do the following when the traffic is matched:

Enable ▾ asm ▾ for policy demo-asm-policy-linux-high ▾

   - **Layer 7 Policy attached to Virtual Server:**



Local Traffic » Virtual Servers : Virtual Server List » Demo_vs

Load Balancing

Default Pool	Demo_Appsvcs_pool_0
Default Persistence Profile	None
Fallback Persistence Profile	None

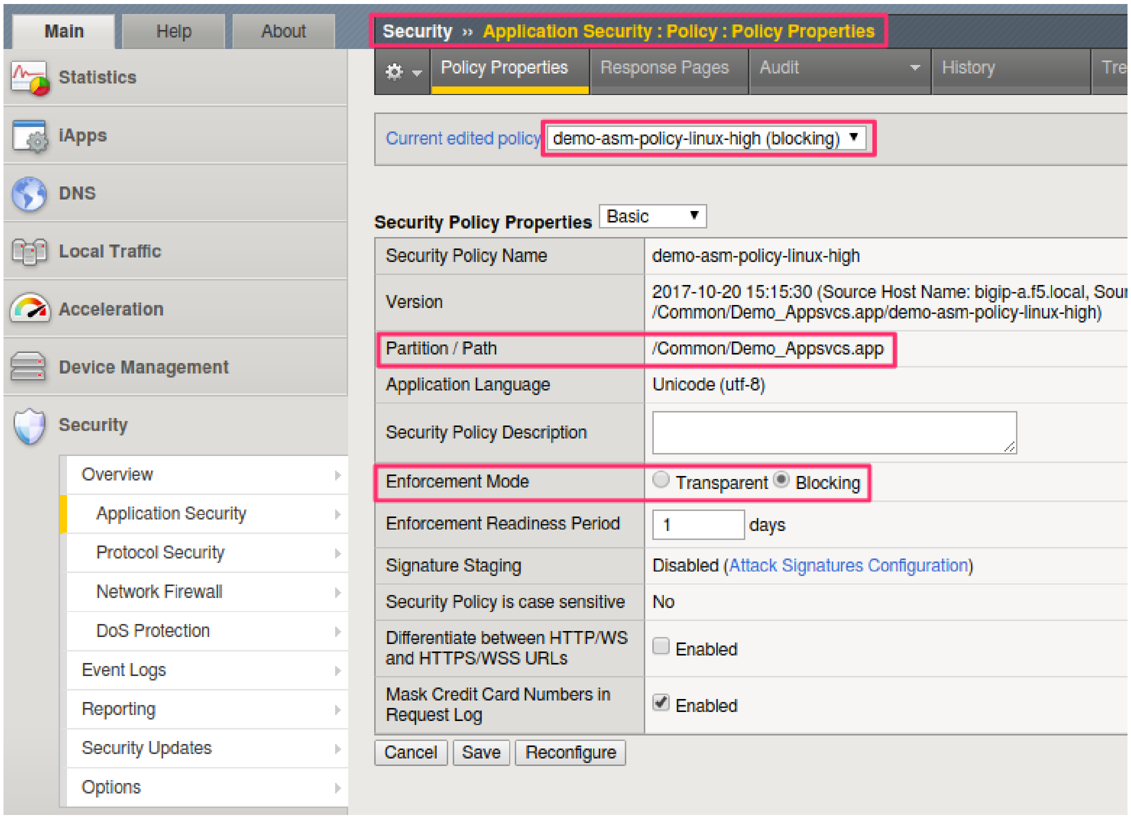
iRules

Name
No records to display.

Policies

Name
/Common/Demo_Appsvcs.app/Demo_Appsvcs_l7policy

- **ASM WAF Policy:**



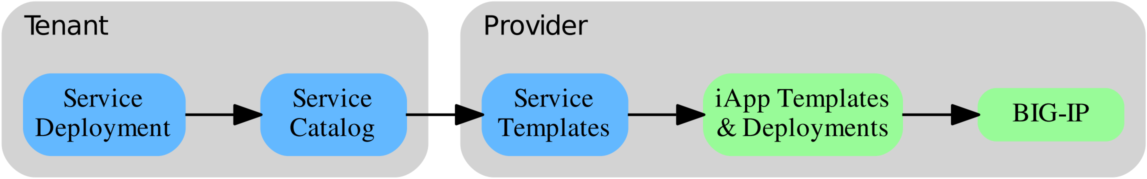
Security » Application Security : Policy : Policy Properties

Current edited policy demo-asm-policy-linux-high (blocking)

Security Policy Properties Basic

Security Policy Name	demo-asm-policy-linux-high
Version	2017-10-20 15:15:30 (Source Host Name: bigip-a.f5.local, Sou /Common/Demo_Appsvcs.app/demo-asm-policy-linux-high)
Partition / Path	/Common/Demo_Appsvcs.app
Application Language	Unicode (utf-8)
Security Policy Description	
Enforcement Mode	○ Transparent ● Blocking
Enforcement Readiness Period	1 days
Signature Staging	Disabled (Attack Signatures Configuration)
Security Policy is case sensitive	No
Differentiate between HTTP/WS and HTTPS/WSS URLs	□ Enabled
Mask Credit Card Numbers in Request Log	☑ Enabled

2.5 Module 3: Creating Declarative Service Interfaces with iWorkflow



In this module we will explore how to use F5's iWorkflow platform to further abstract application services and deliver those services to tenants. iWorkflow has two main purposes in the Automation & Orchestration toolchain:

- Provide simplified but customizable Device Onboarding workflows
- Provide a tenant/provider interface for L4 - L7 service delivery

When moving to an iWorkflow based toolchain it's important to understand that L1-3 Automation (Device Onboarding, Networking, etc) and L4-7 (Deployment of Virtual Servers, Pools, etc) are separated and delivered by different features.

L1-3 Networking and Device Onboarding are delivered by **Cloud Connectors** that provide an abstracted interface to BIG-IP Onboarding in different environments.

L4-7 service delivery is accomplished by:

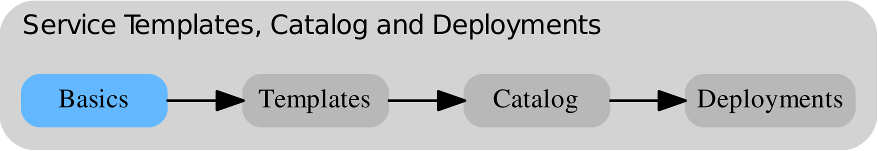
- **Declarative:** Consuming F5 iApp templates from BIG-IP devices and creating a Service Catalog.
- **Imperative:** Consuming the iWorkflow REST Proxy to drive API calls to BIG-IP devices

The labs in the module will focus on the high level features in place to achieve full L1-7 automation. As mentioned above, iApps are a key component of this toolchain.

In this Module we will focus on building a **Service Catalog** using the App Services iApp template you learned about in Module 2. The focus in Module 2 was showing how to drive rich deployments, however, a large amount of F5 **Domain Specific Knowledge** was still required to drive the deployments. From a conceptual view iApp templates alone do not fully satisfy the requirement for a fully **Declarative** interface because while the iApp template simplifies the underlying **Imperative** actions it does not allow the adminstrator to build an **Interface** that minimizes or eliminates the need for **Domain Specific Knowledge**.

For example, we deployed a service that enabled HTTP Traffic Management with an iRule attached and Profile Customizations. To the F5 administrator these are all very familar terms, however, to a consumer, such as an Application Owner, the terms *Virtual Server*, *iRule*, *Profile*, etc. are foreign concepts. To solve this problem iWorkflow allows the adminstrator to create a **Service Catalog Template** that is an **Abstraction** of the iApp templates input fields. By doing this the F5 administrator can **create an interface tailored to the use case and knowledge level of the CONSUMER rather than the ADMINSTRATOR**, enabling full featured and complex Layer 4-7 Application and Security services that are tailored to business need and use case rather than the technical implementation. Additionally, the **Service Abstraction** acheived when creating the **Service Catalog** enables the easy integration of F5 services with third-party tools and methodologies such as DevOps.

2.5.1 Lab 3.1: iWorkflow Onboarding



In this lab we will use the *Runner*, introduced in previous labs to complete the onboarding of the F5 iWorkflow device. The onboarding process creates the initial configuration required to start creation of Service Catalog Templates.

iWorkflow Overview

Before looking at the details of the onboarding process, lets discuss the new components iWorkflow introduces to our toolchain.

Device Discovery

In order for iWorkflow to interact with a BIG-IP device it must be discovered by iWorkflow. The device discovery process leverages the existing CMI Device Trust infrastructure on BIG-IP. Currently there is a limitation that a single BIG-IP device can only be 'discovered' by ONE of iWorkflow or BIG-IQ CM at a time. In this lab will we discover the existing BIG-IP devices from your lab environment.

Tenants & Connectors

iWorkflow implements a Tenant/Provider interface to enable abstracted deployment of L4-7 into various environment. In conjuction iWorkflow Connectors serve as the L1-3 Network and Device Onboarding automation component in the automation toolchain. In this lab we will create a 'BIG-IP Connector' for the BIG-IP devices in the lab environment. This connector will then allow you to drive a fully automated deployment from the iWorkflow Service Catalog.

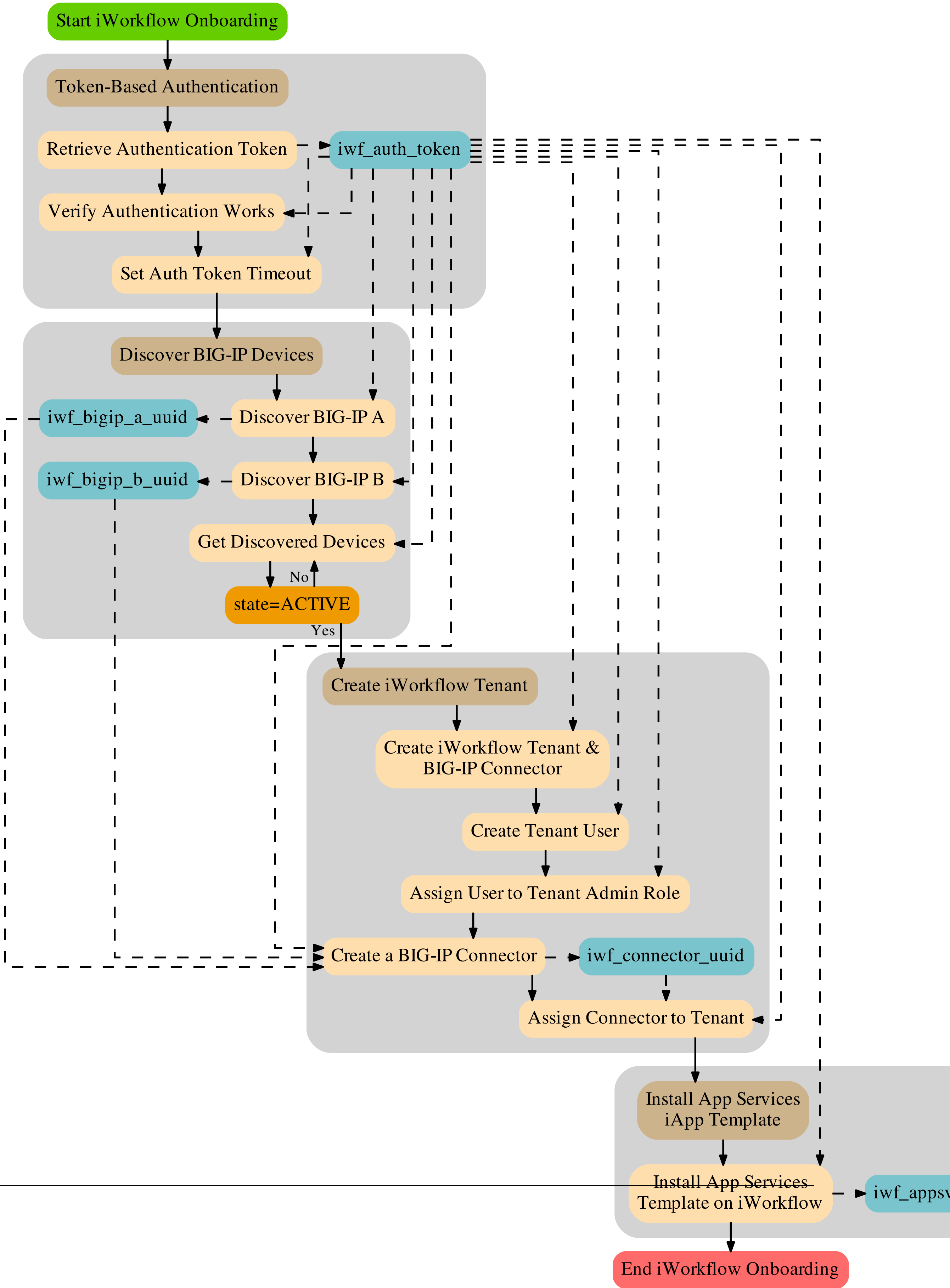
iApp Templates

iWorkflow serves as an iApp Template Source-of-Truth for discovered BIG-IP devices. This allows an F5 administrator to manage iApp templates in a single place with iWorkflow installing required templates on BIG-IP devices as required **during** service deployment.

Onboarding Process Overview

The process implemented in the `Lab 3.1 - iWorkflow Onboarding` folder of the Postman collection is diagrammed below.

Note: The diagram below represents environment variables in blue. You can follow the lines on each variable to understand which request populates the variable and how they are subsequently used.

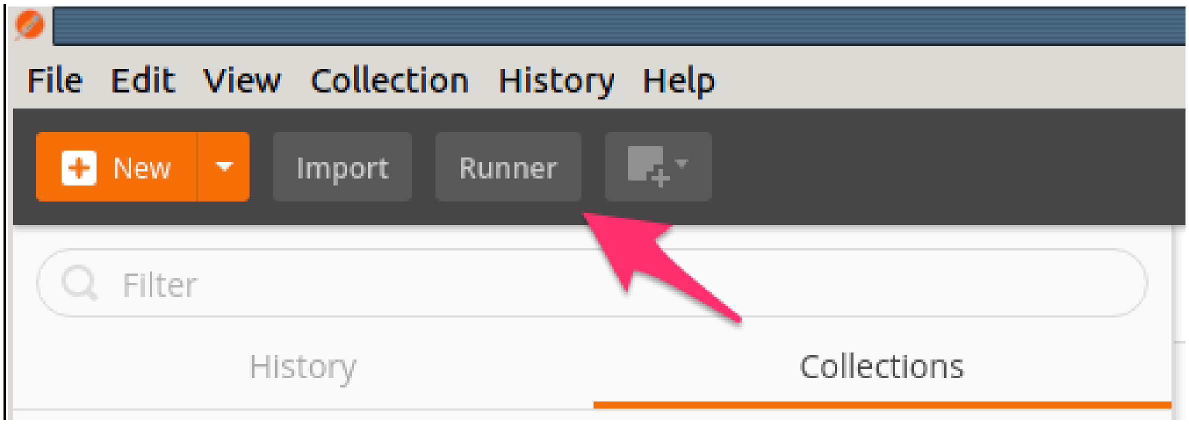

Start iWorkflow Onboarding
Token-Based Authentication
Retrieve Authentication Token
iwf_auth_token
Verify Authentication Works
Set Auth Token Timeout
Discover BIG-IP Devices
iwf_bigip_a_uuid
Discover BIG-IP A
iwf_bigip_b_uuid
Discover BIG-IP B
Get Discovered Devices
No
state=ACTIVE
Yes
Create iWorkflow Tenant
Create iWorkflow Tenant &
BIG-IP Connector
Create Tenant User
Assign User to Tenant Admin Role
Create a BIG-IP Connector
iwf_connector_uuid
Assign Connector to Tenant
Install App Services
iApp Template
Install App Services
Template on iWorkflow
iwf_appsv
End iWorkflow Onboarding

Task 1 - Onboard iWorkflow using Runner

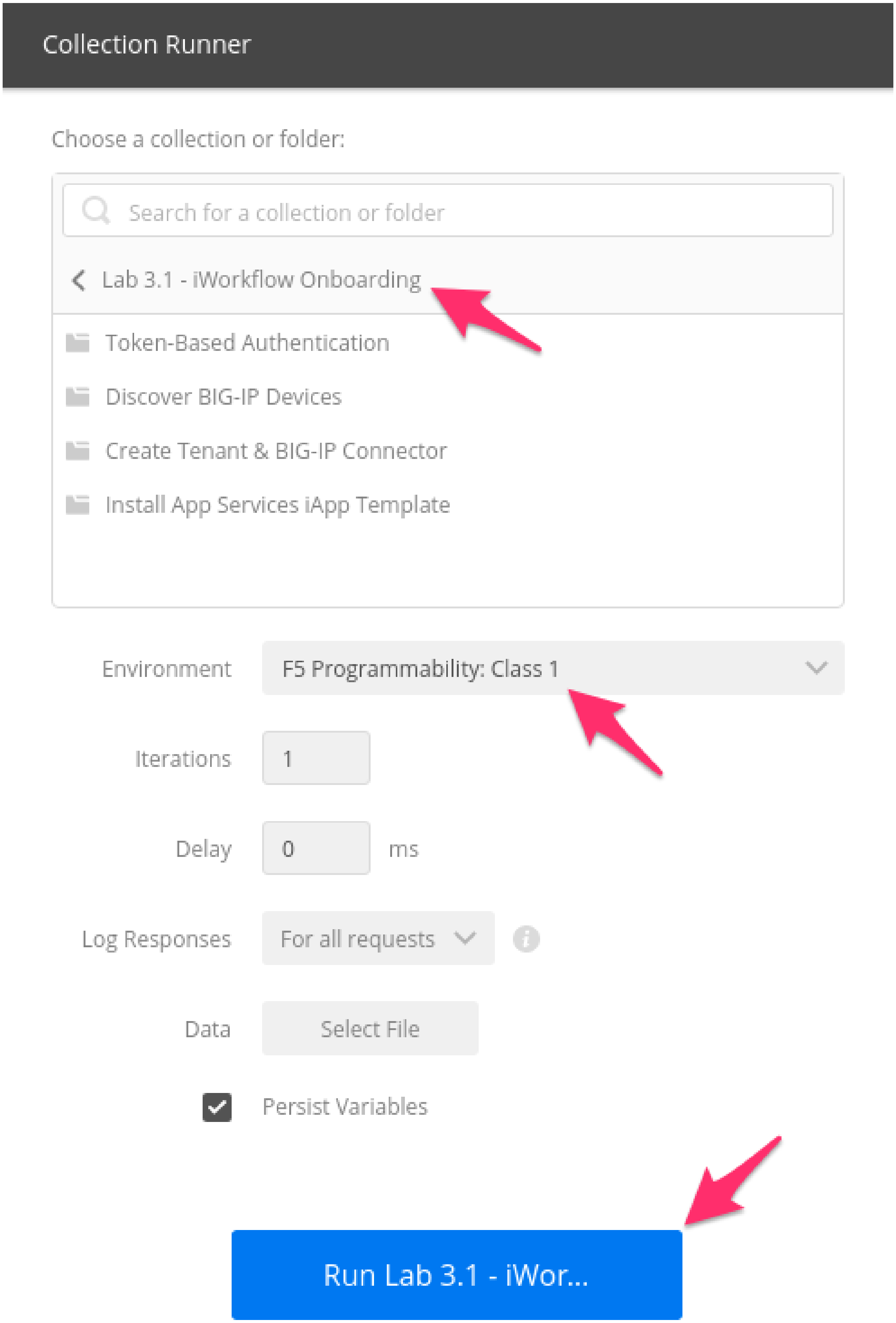
In this task we will use the *Runner* to execute a series of requests contained in the `Lab 3.1 - iWorkflow Onboarding` folder.

Perform the following steps to build the cluster:

1. Click the *Runner* button at the top right of your Postman window:



2. Select the `F5 Programmability: Class 1` Collection then the `Lab 3.1 - iWorkflow Onboarding` folder. Next, be sure the environment is set to `F5 Programmability: Class 1`:



Collection Runner

Choose a collection or folder:

Search for a collection or folder

Lab 3.1 - iWorkflow Onboarding

- Token-Based Authentication
- Discover BIG-IP Devices
- Create Tenant & BIG-IP Connector
- Install App Services iApp Template

Environment: F5 Programmability: Class 1

Iterations: 1

Delay: 0 ms

Log Responses: For all requests

Data: Select File

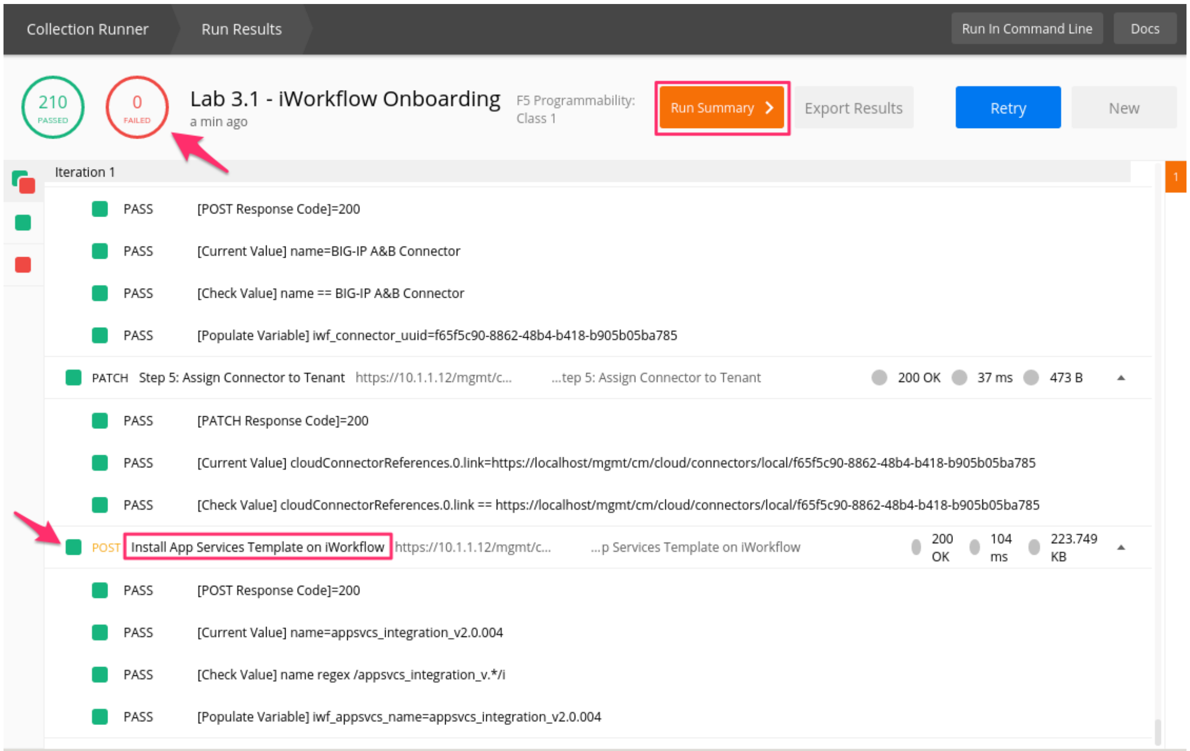
☑ Persist Variables

Run Lab 3.1 - iWor...

3. Click the *Run Lab 3.1 - iWor. . .* button

4. The results window will now populate. You will see each request in the folder is sent and it's associated test results are displayed on the screen. Onboarding iWorkflow can take a few minutes. You can follow the progress by scrolling down the results window.

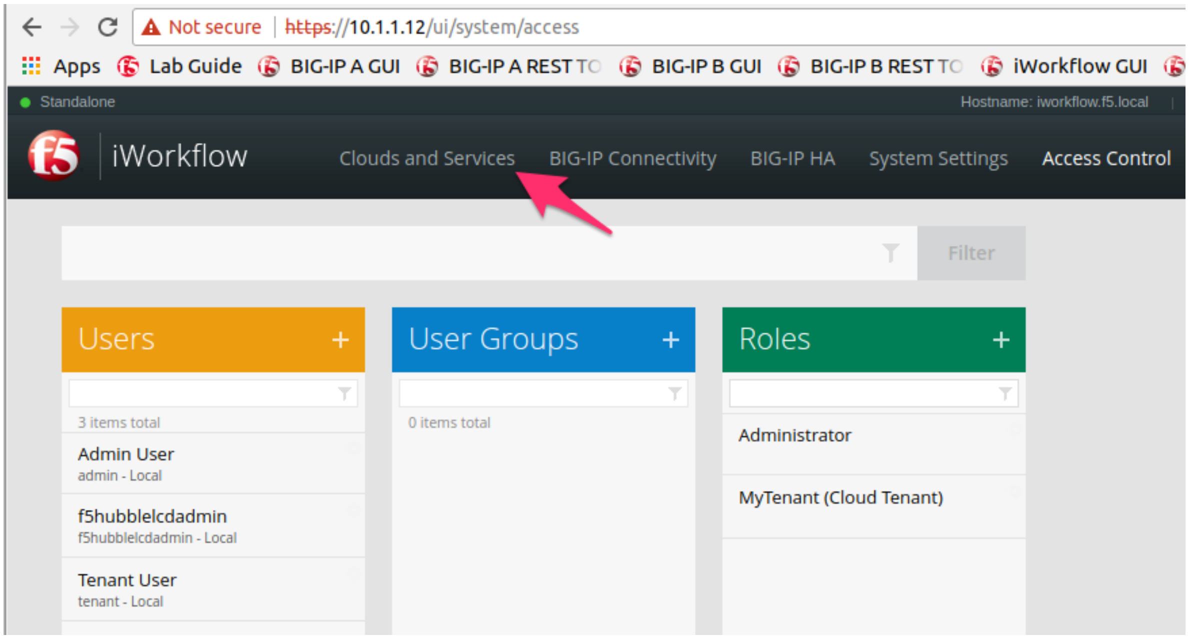
5. Once the *Run Summary* button appears the folder has finished running. You should have 0 failures and the last item in the request list should be named `Install App Services Template on iWorkflow`



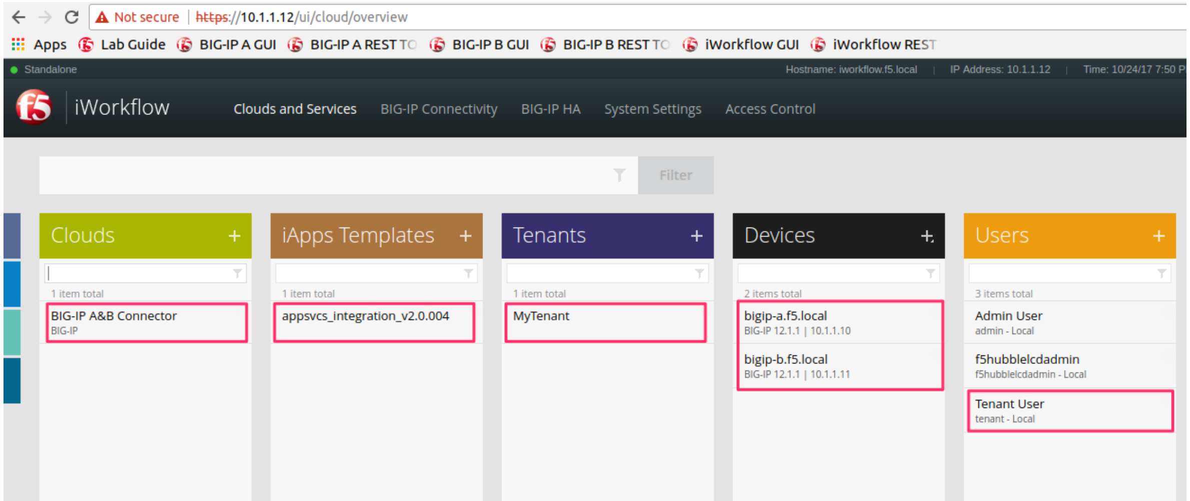
Lab 3.1 - iWorkflow Onboarding
a min ago
F5 Programmability: Class 1

```
Iteration 1
    PASS    [POST Response Code]=200
    PASS    [Current Value] name=BIG-IP A&B Connector
    PASS    [Check Value] name == BIG-IP A&B Connector
    PASS    [Populate Variable] iwf_connector_uuid=f65f5c90-8862-48b4-b418-b905b05ba785
PATCH  Step 5: Assign Connector to Tenant   https://10.1.1.12/mgmt/c...   ...tep 5: Assign Connector to Tenant   200 OK   37 ms   473 B
    PASS    [PATCH Response Code]=200
    PASS    [Current Value] cloudConnectorReferences.0.link=https://localhost/mgmt/cm/cloud/connectors/local/f65f5c90-8862-48b4-b418-b905b05ba785
    PASS    [Check Value] cloudConnectorReferences.0.link == https://localhost/mgmt/cm/cloud/connectors/local/f65f5c90-8862-48b4-b418-b905b05ba785
POST  Install App Services Template on iWorkflow  https://10.1.1.12/mgmt/c...   ...p Services Template on iWorkflow   200 OK   104 ms   223.749 KB
    PASS    [POST Response Code]=200
    PASS    [Current Value] name=appsvcs_integration_v2.0.004
    PASS    [Check Value] name regex /appsvcs_integration_v.*/i
    PASS    [Populate Variable] iwf_appsvcs_name=appsvcs_integration_v2.0.004
```

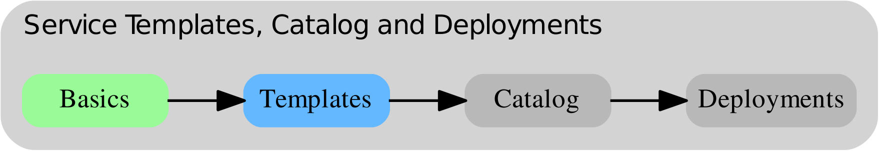
6. At this point you can log into iWorkflow using Chrome at `https://10.1.1.12` and `admin/admin` credentials. Click *Clouds and Services* at the top of the window:



7. Browse the various panes to see what was created:



2.5.2 Lab 3.2: Create a Declarative Service Catalog



In the introduction to this module we discussed the importance of using **Service Templates** to build a **Declarative Service Catalog**. This lab will show to create a few examples of **Service Templates** (Templates). It's important to understand that while the packaged examples used in this lab are great starting points, you should use them as a starting point for creating your own **Service Catalog** that meets the requirements of your environment.

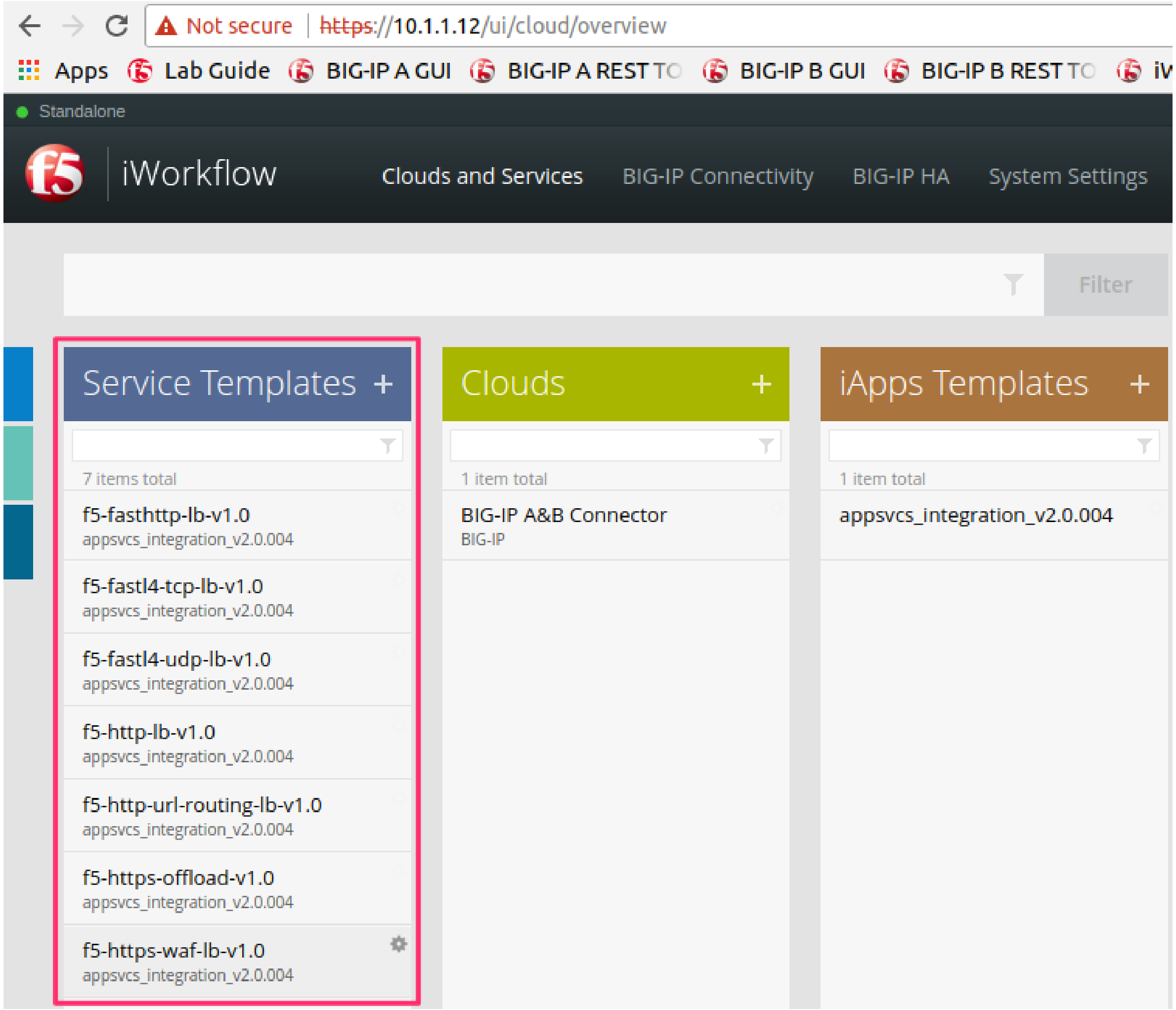
We will explore the first example in depth so you can gain an understanding of how the templates are structured. For the remaining templates you can just repeat the steps used with the first example.

The templates used in this lab all have a version number appended to the name (Example: `f5-http-lb-v1.0`). It's important that this pattern is followed in your environment. Explicitly versioning the templates allows for migration between template versions in a stable manner. Without versioning any changes to the template could result in *every* deployment associated with the template being modified at the same time. With versioning the application owner or F5 administrator can choose to either migrate all deployments at the same time OR perform the migration on a per deployment manner.

Task 1 - Create the Service Templates

In this task we will use the Runner to quickly create our sample Service Templates. Perform the following steps to complete this task:

1. Click the *Runner* button at the top right of your Postman window.
2. Select *F5 Programmability: Class 1 → Lab 3.2 - Create a Declarative Service Catalog* folder.
3. Select the `F5 Programmability: Class 1` environment
4. Click the *Run Lab 3.2 - Crea...* button and wait for the run to complete. Verify no errors were encountered.
5. Open the iWorkflow GUI in Chrome by navigating to `https://10.1.1.12`
6. Expand the *Service Templates* panel and verify all the templates have been created:



Service Templates

7 items total

- f5-fasthttp-lb-v1.0 — appsvcs_integration_v2.0.004
- f5-fastl4-tcp-lb-v1.0 — appsvcs_integration_v2.0.004
- f5-fastl4-udp-lb-v1.0 — appsvcs_integration_v2.0.004
- f5-http-lb-v1.0 — appsvcs_integration_v2.0.004
- f5-http-url-routing-lb-v1.0 — appsvcs_integration_v2.0.004
- f5-https-offload-v1.0 — appsvcs_integration_v2.0.004
- f5-https-waf-lb-v1.0 — appsvcs_integration_v2.0.004

Clouds

1 item total

- BIG-IP A&B Connector — BIG-IP

iApps Templates

1 item total

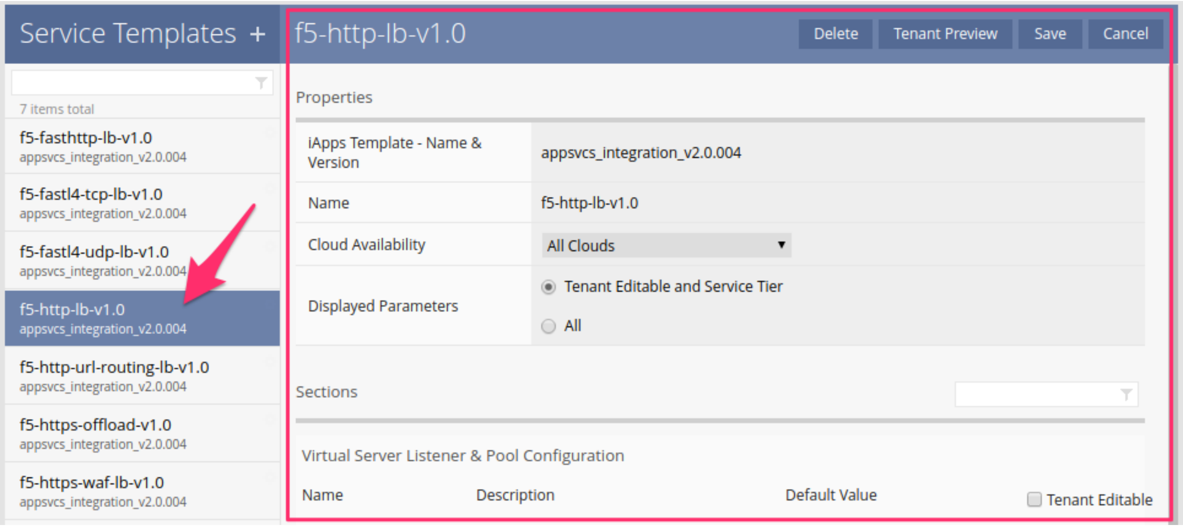
- appsvcs_integration_v2.0.004

Task 2 - Explore the f5-http-lb-v1.0 Template

Now that we've created our Templates let's review one of them in depth.

Perform the following steps to complete this task:

1. Open the `f5-http-lb-v1.0` Template by double clicking it:



7 items total

- f5-fasthttp-lb-v1.0 appsvcs_integration_v2.0.004
- f5-fastl4-tcp-lb-v1.0 appsvcs_integration_v2.0.004
- f5-fastl4-udp-lb-v1.0 appsvcs_integration_v2.0.004
- f5-http-lb-v1.0 appsvcs_integration_v2.0.004
- f5-http-url-routing-lb-v1.0 appsvcs_integration_v2.0.004
- f5-https-offload-v1.0 appsvcs_integration_v2.0.004
- f5-https-waf-lb-v1.0 appsvcs_integration_v2.0.004

Properties

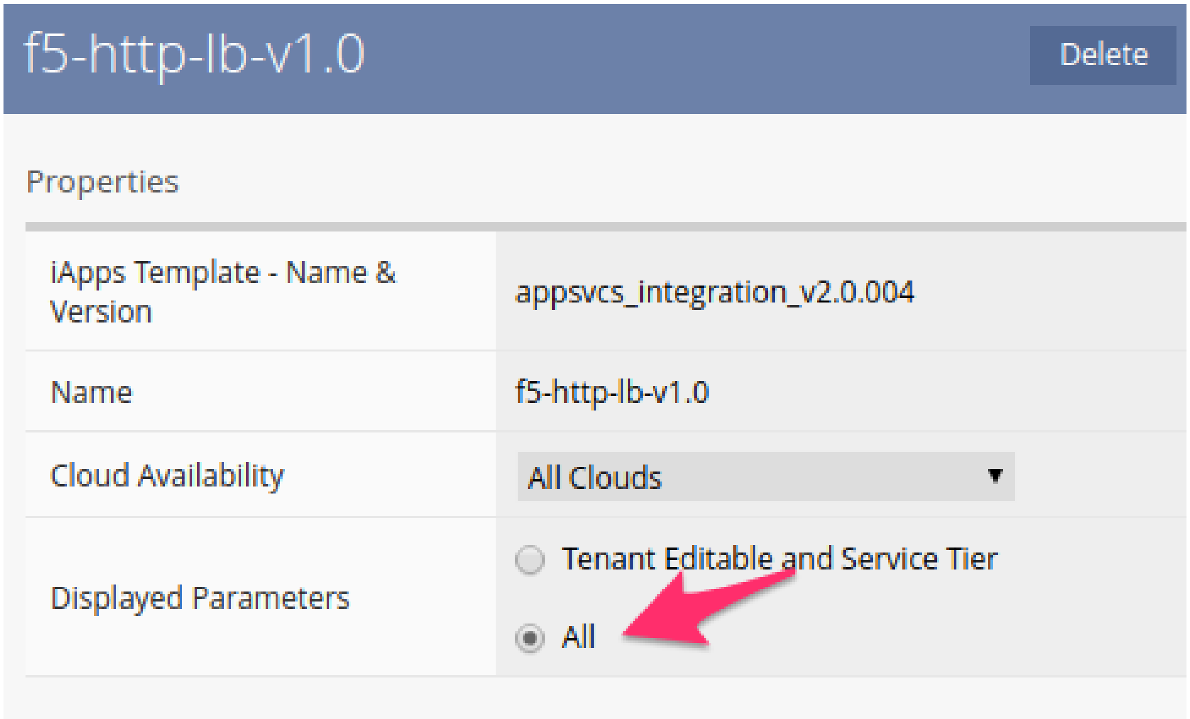
iApps Template - Name & Version	appsvcs_integration_v2.0.004
Name	f5-http-lb-v1.0
Cloud Availability	All Clouds ▾
Displayed Parameters	● Tenant Editable and Service Tier ○ All

Sections

Virtual Server Listener & Pool Configuration

Name	Description	Default Value	□ Tenant Editable

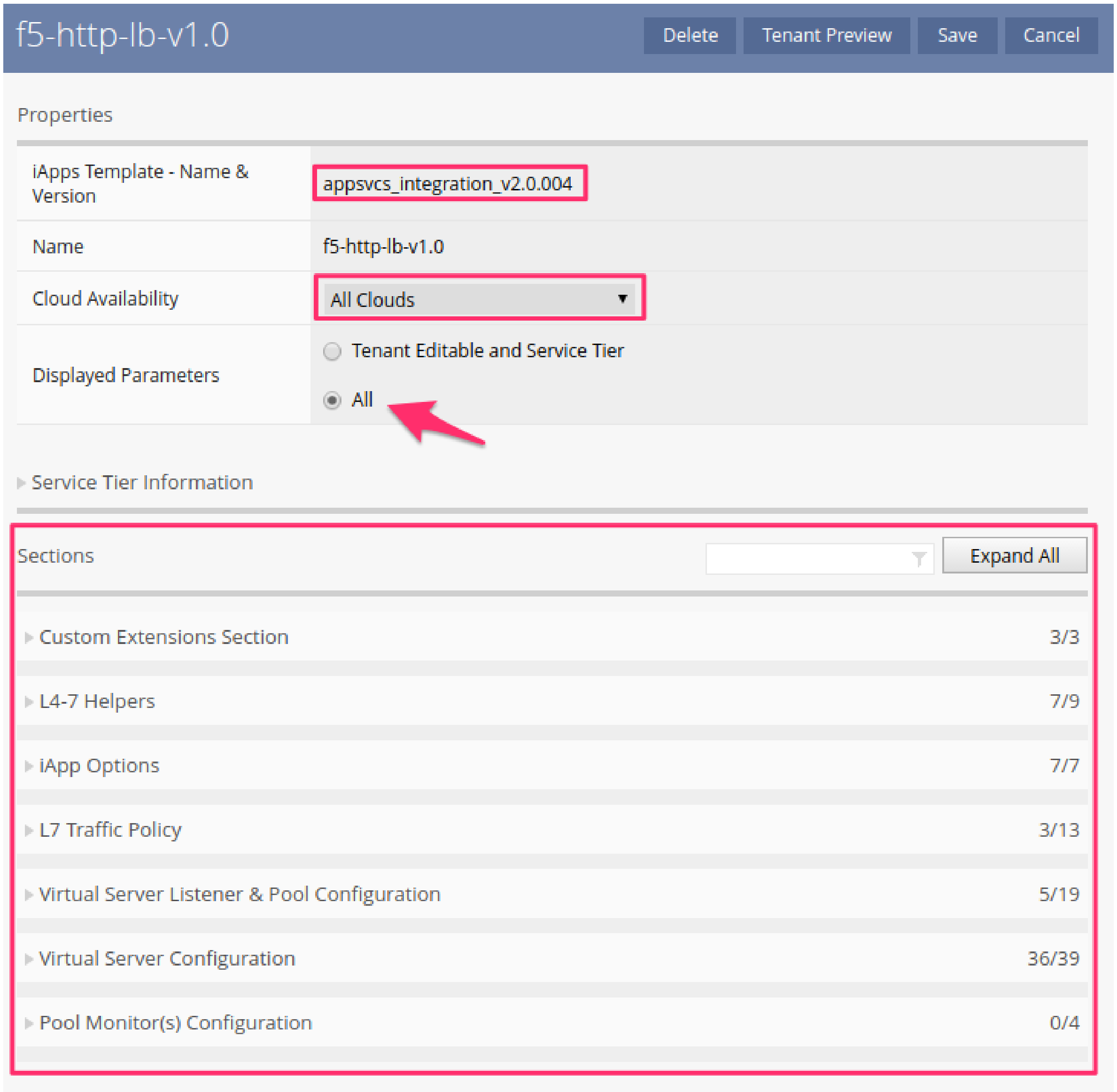
2. Let’s examine the *Properties* pane.
3. Select *All* in the *Displayed Parameters* section:



Properties

iApps Template - Name & Version	appsvcs_integration_v2.0.004
Name	f5-http-lb-v1.0
Cloud Availability	All Clouds ▾
Displayed Parameters	○ Tenant Editable and Service Tier ● All

4. This pane shows detailed information about the Template such as:
   - iApp Template Name & Version the Service Template is using
   - The Connectors/Clouds that may use this template
   - A control that toggles which Parameters are displayed in the pane
   - The input Sections and Fields (collapsed in screeshot) for the iApp Template



f5-http-lb-v1.0

Properties

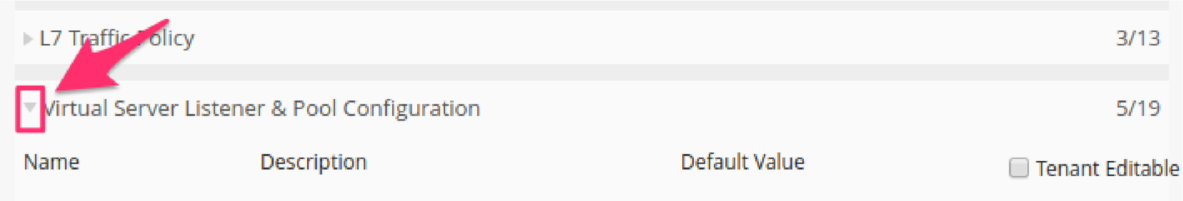
iApps Template - Name & Version	appsvcs_integration_v2.0.004
Name	f5-http-lb-v1.0
Cloud Availability	All Clouds ▾
Displayed Parameters	○ Tenant Editable and Service Tier ● All

▸ Service Tier Information

Sections

▸ Custom Extensions Section	3/3
▸ L4-7 Helpers	7/9
▸ iApp Options	7/7
▸ L7 Traffic Policy	3/13
▸ Virtual Server Listener & Pool Configuration	5/19
▸ Virtual Server Configuration	36/39
▸ Pool Monitor(s) Configuration	0/4

5. In the *Sections* portion of the pane, find the *Virtual Server Listener & Pool Configuration* section. Click the triangle to expand the section:



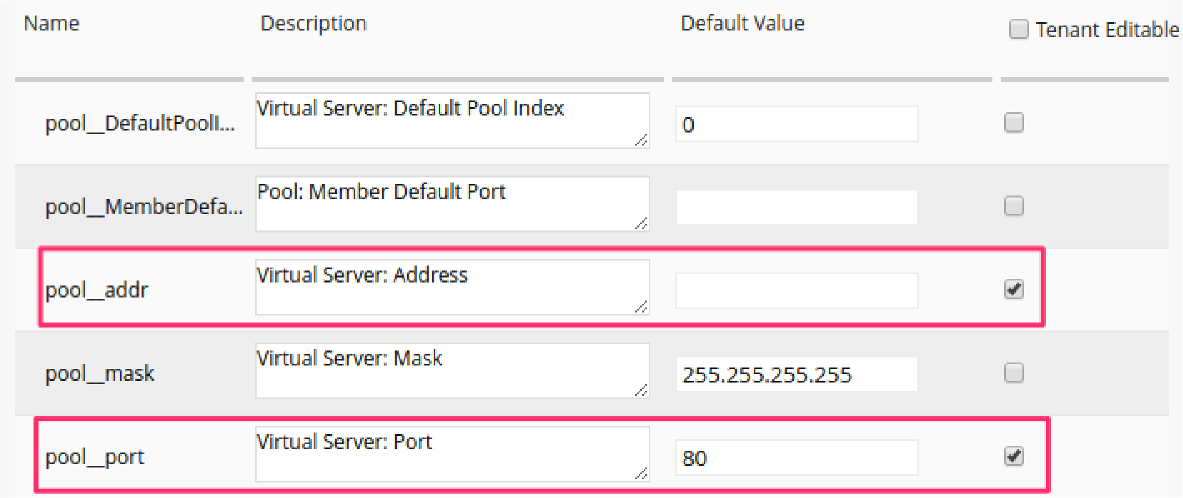
▸ L7 Traffic Policy	3/13
▾ Virtual Server Listener & Pool Configuration	5/19

Name	Description	Default Value	□ Tenant Editable

6. You can now see all the input fields associated with this section of the iApp template. These fields are defined by the iApp Template itself. In the previous lab, when we installed the App Services iApp Template, iWorkflow created a internal representation of the input fields used in the iApp template. iWorkflow then allows you to create a template that:

   - Define which fields are `Tenant Editable`, therefore exposed to the Tenant interface
   - Setting a default value for the field

- If the field is NOT `Tenant Editable` the default value is sent during a Service Deployment, however, the Tenant cannot see or modify the value
- If the field is `Tenant Editable` the default value is populated for the Tenant and the Tenant may edit it during a Service Deployment



Name	Description	Default Value	☐ Tenant Editable
pool__DefaultPoolI...	Virtual Server: Default Pool Index	0	☐
pool__MemberDefa...	Pool: Member Default Port		☐
pool__addr	Virtual Server: Address		☑
pool__mask	Virtual Server: Mask	255.255.255.255	☐
pool__port	Virtual Server: Port	80	☑

In the case of the fields shown in the example:

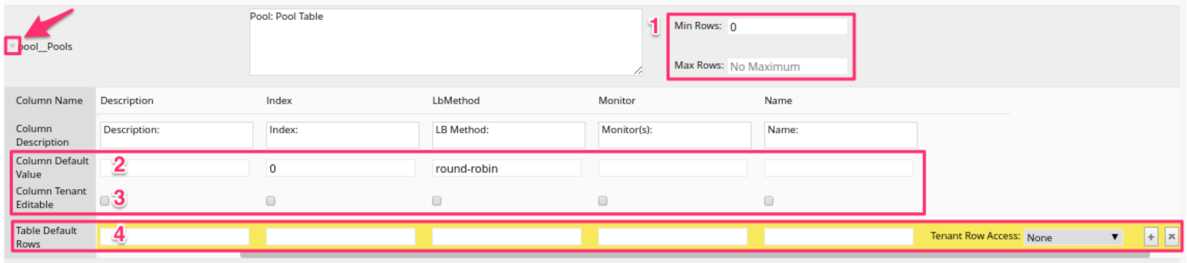
- `pool__DefaultPoolIndex`: A value of `0` will be sent during a deployment
- `pool__MemberDefaultPort`: Nothing will be sent
- `pool__addr`: Tenant will be allowed to populate the field with a value
- `pool__mask`: A value of `255.255.255.255` will be sent
- `pool__port`: Tenant will see `80` and can change the field

By combining different combinations of **Default Values** and `Tenant Editable` fields you can create many different types of templates to match your requirements.

Note: The App Services iApp Template has been specifically designed to integrate with iWorkflow and Automation use cases. While any iApp template that is properly versioned can be used with iWorkflow, you should consider whether the template was designed for Automation use cases or not. Many iApp templates were designed for a GUI or Wizard based interaction through the BIG-IP TMUI GUI. As a result those templates may not present a good API interface.

7. In addition to simple text fields, iApp templates also support table based input. The App Services iApp uses this capability to allow input of more complex data such as Pools, Pool Members and Layer 7 Routing Policies. iWorkflow allows you to have granular control over how the Tenant can interact with a table. Let's find the `pool__Pools` table and click the triangle to expand it:

Note: To accomodate screen size this screenshot does not show all the columns in the table.



pool__Pools

Pool: Pool Table

Min Rows: 0

Max Rows: No Maximum

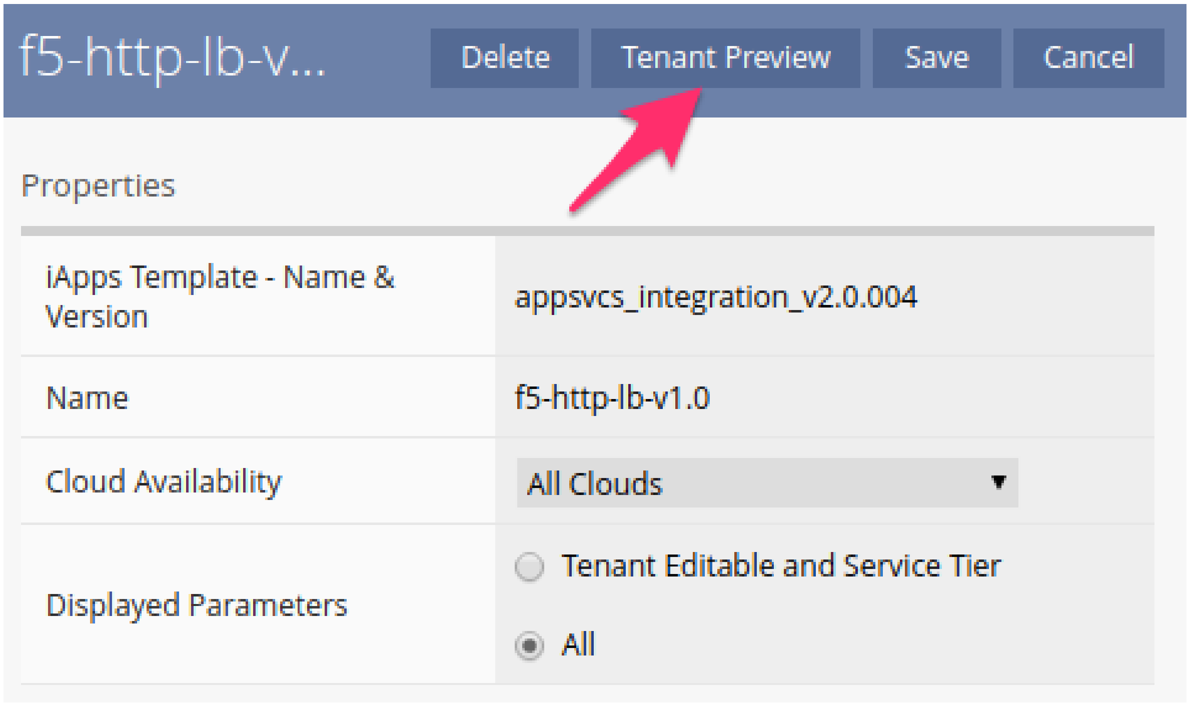
Column Name	Description	Index	LbMethod	Monitor	Name
Column Description	Description:	Index:	LB Method:	Monitor(s):	Name:
Column Default Value		0	round-robin		
Column Tenant Editable	□	□	□	□	□
Table Default Rows					

Tenant Row Access: None ▾ + ×

The highlighted sections in the image above correspond to the capabilities in the list below:

- [1] Definition of the *Min* and *Max* number of rows in a table
  - Example: Define a fixed number or limit for the number of Pools a Tenant can deploy
- [2] *Default Values* for each column in a table
  - Example: Define a default Load Balancing Method for deployed Pools
- [3] *Tenant Editable* flag for each column in the table
  - Example: Only allow the Tenant to control the Load Balancing Method and Name of a Pool, while defaulting all other values.
- [4] *Default Rows* that auto-populate a desired input for the Tenant. Each row can have a No Access, Read-Only or Write ACL applied.
  - Example: Define a Service that allows URL Based Content Routing to only two pools.
    - Define 2 *Default Rows* in the Pools table
    - Set the *Min* & *Max* value to 2

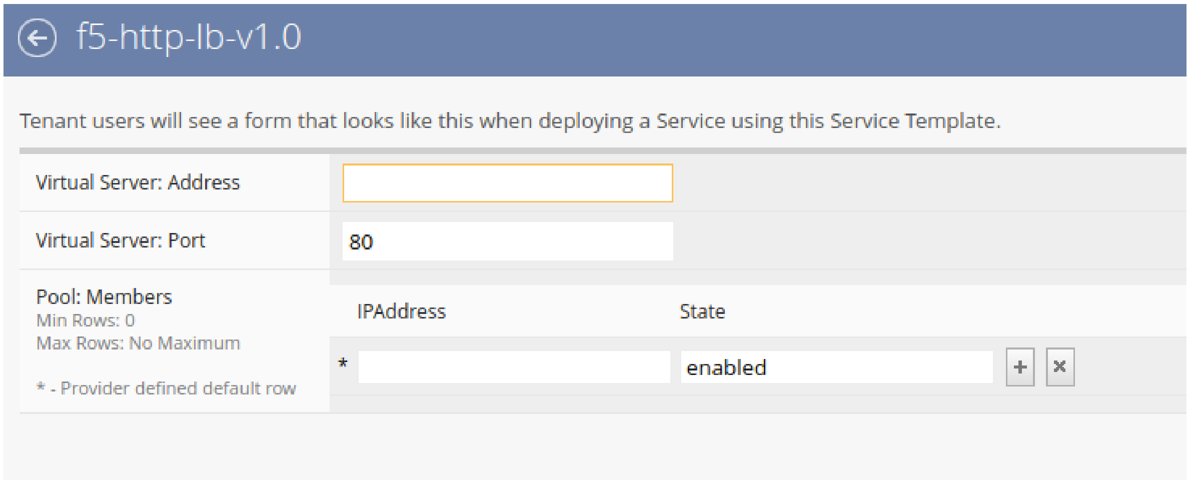
8. Finally, to assist in designing a Tenant interface, iWorkflow allows you to preview what the Tenant UI would look like for a Service Template. To view preview for click the *Tenant Preview* button:



Properties

iApps Template - Name & Version	appsvcs_integration_v2.0.004
Name	f5-http-lb-v1.0
Cloud Availability	All Clouds ▾
Displayed Parameters	○ Tenant Editable and Service Tier ● All

9. The preview window shows how the Tenant UI would present the Service Template. As you can see the interface is vastly simplified and only *Tenant Editable* fields are shown. Because the true deployment details are filtered from the Tenant, the Service Deployment requires much less **Domain Specific Knowledge**. Keep in mind that while the Tenant interface may be simple, you can leverage advanced functionality in the Service Template.



f5-http-lb-v1.0

Tenant users will see a form that looks like this when deploying a Service using this Service Template.

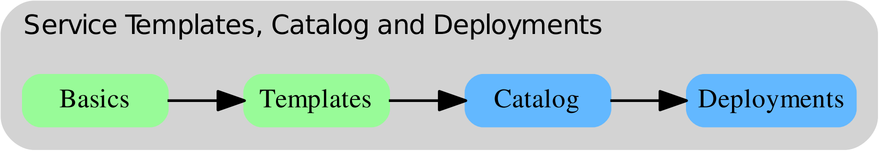
Virtual Server: Address		
Virtual Server: Port	80	
Pool: Members Min Rows: 0 Max Rows: No Maximum * - Provider defined default row	IPAddress	State
	*	enabled

Task 3 - Explore the Remaining Service Templates

Using the pattern in the last task explore the other Service Templates that were created earlier. A description of each Service Template is included in the table below. In all cases the Template has been configured with the appropriate Monitors, Profiles and Options for the use case.

Service Template	**Description**
`f5-http-lb-v1.0`	HTTP Load Balancing to a Single Pool
`f5-https-offload-v1.0`	HTTPS Offload and Load Balancing to a Single Pool
`f5-fasthttp-lb-v1.0`	Performance-enhanced HTTP Load Balancing to a Single Pool
`f5-fastl4-udp-lb-v1.0`	Generic L4 TCP Load Balancing to a Single Pool
`f5-fastl4-udp-lb-v1.0`	Generic L4 UDP Load Balancing to a Single Pool
`f5-http-url-routing-lb-v1.0`	HTTP Load Balancing with URL Based Content Routing to Multiple Pools
`f5-https-waf-lb-v1.0`	HTTPS Offload, Web Application Firewall Protection and Load Balancing to a Single Pool

2.5.3 Lab 3.3: Deploy L4-7 Services



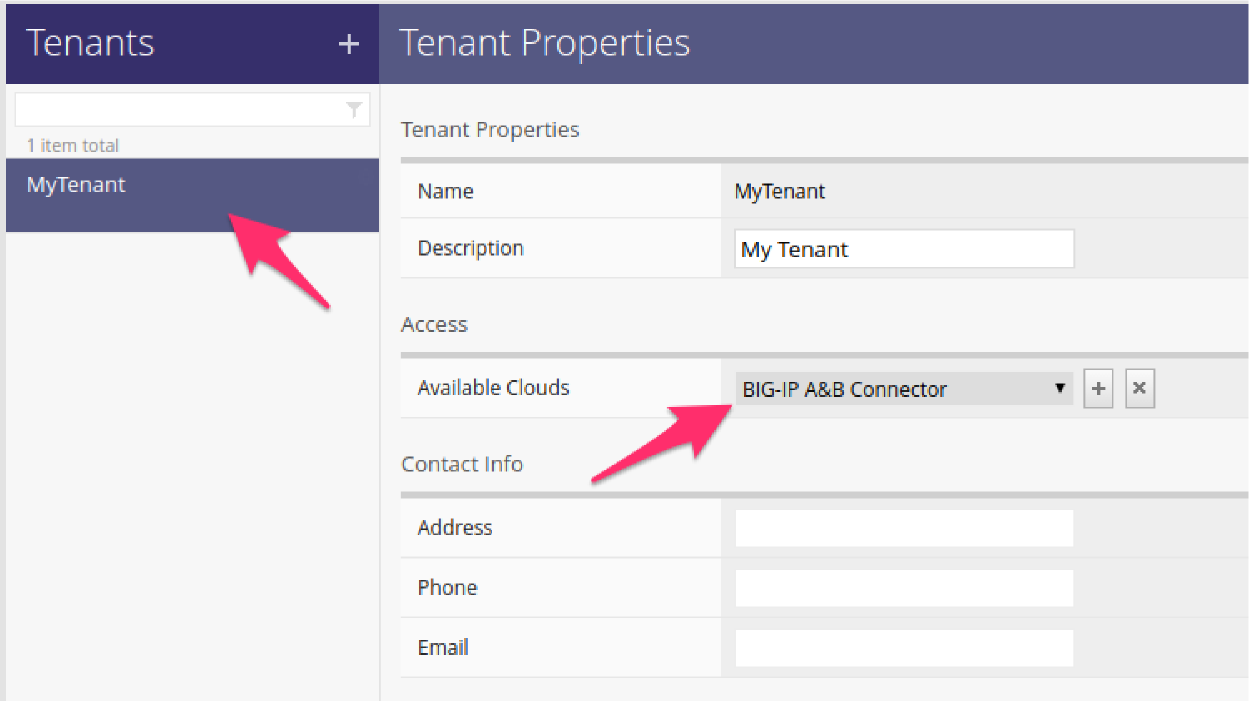
Up to this point we have spent a lot of time building our toolchain to create a Declarative Service Catalog. We are now at the point where we can perform a Declarative, Abstracted Service Deployment using the iWorkflow Tenant Service Catalog, Tenant API and optionally the built-in Tenant GUI.

As we did in the previous lab we will explore the first deployment in depth so you can implement a full Service Lifecycle: Create, Read, Update and Delete (CRUD) operations. For the remaining deployments you can just repeat the steps used with the first example.

Tenant Overview

iWorkflow Tenants allow Consumers to perform Service Lifecycle operations in an isolated environment. All actions performed prior to this lab have been in what's called the `Provider` space and, by nature, are masked from Tenants unless specifically exposed. As a result of the Tenant isolation, each Tenant maintains its own set of Users and Roles associated with those users, allowing each Tenant full control of the actions Tenant Users can perform.

During our iWorkflow Onboarding process in Lab 3.1 we created a *Tenant* named `MyTenant` and an associated *Tenant User* with a username of `tenant`. Additionally we gave `MyTenant` access to the *BIG-IP Connector* named `BIG-IP A&B Connector`:



Tenants

1 item total

MyTenant

Tenant Properties

Tenant Properties

Name	MyTenant
Description	My Tenant

Access

Available Clouds	BIG-IP A&B Connector

Contact Info

Address	
Phone	
Email	

This gives the `tenant` user the ability to perform CRUD operations on Service Deployments.

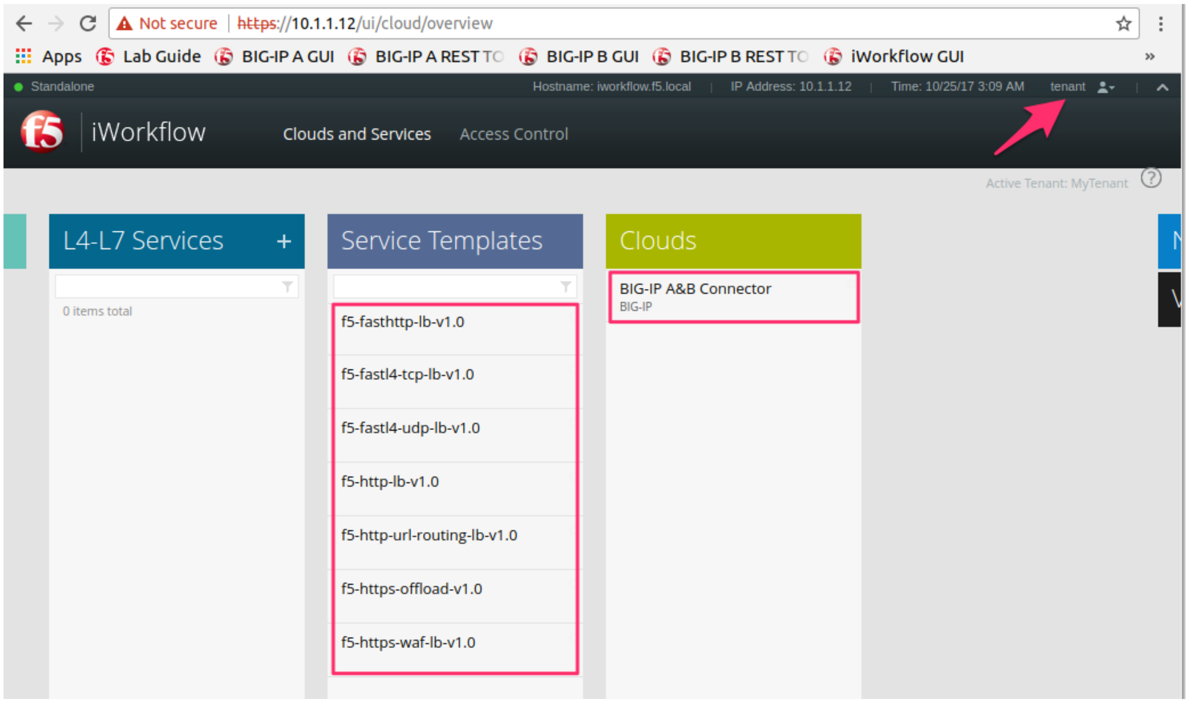
Note: Service Templates can also be assigned to specific Cloud Connectors, allowing you to restrict the use of Templates to a specific Tenant and set of BIG-IP resources.

Task 1 - Login to the iWorkflow Tenant UI

iWorkflow provides a Tenant UI that can act as a simple self-service portal for Tenants. In this lab we'll use the Tenant UI to monitor the results of various actions we take via the iWorkflow Tenant API.

Perform the following steps to complete this task:

1. Open a new Chrome window/tab and connect to `https://10.1.1.12`
2. Use the `MyTenant` Tenant User credentials to login:
   - Username: `tenant`
   - Password: `tenant`
3. You will see a user interface that looks similar to the Provider UI, however, the access is limited to Tenant specific objects. You can see a list of available *Service Templates* and *Clouds* with their associated Connectors:



Task 2 - Authenticate to the iWorkflow Tenant API

As described above, the Tenant interfaces to iWorkflow maintain their own access control mechanisms. As a result, when performing operations via the Tenant API you must authenticate with a Tenant User (`tenant` in this case).

Perform the following steps to complete this task:

1. In Postman expand the `Lab 3.3 - Deploy L4-7 Services` folder in the collection
2. Click the `Authenticate and Obtain Token for Tenant User` request and examine the JSON request *Body*. Notice that we are sending the credentials for the Tenant User (`tenant`). This request will automatically populate the `iwf_tenant_auth_token` variable in the Postman environment so it can be used by subsequent requests.
3. Click the *Send* button on the `Authenticate and Obtain Token for Tenant User` request. Check the *Test Results* tab to ensure the token was populated.
4. Click the `Set Tenant Authentication Token Timeout` request and click the *Send* button. This request will increase the timeout value for the token so we can complete the lab without having to re-authenticate.

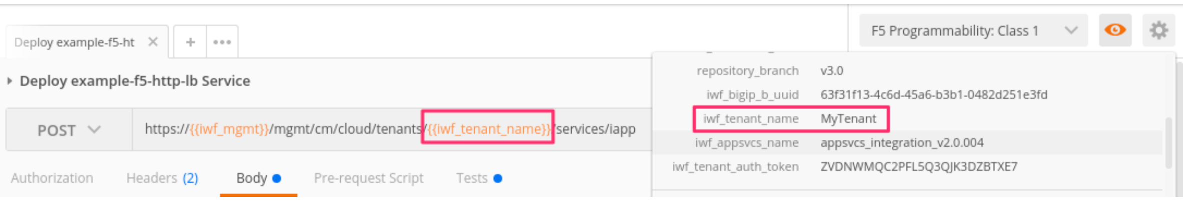
Task 3 - Perform Service Lifecycle Operations

In this task we will perform CRUD operations on Service Deployments demonstrating a full Service Lifecycle for a Tenant Service.

Create

Perform the following steps to complete this task:

1. Click the `Deploy example-f5-http-lb Service` request in the folder.
2. Examine the URI. Notice that the variable `iwf_tenant_name` is used to specify the Tenant we are performing the operation on. In this case `iwf_tenant_name` is set to `MyTenant` in the Postman environment:



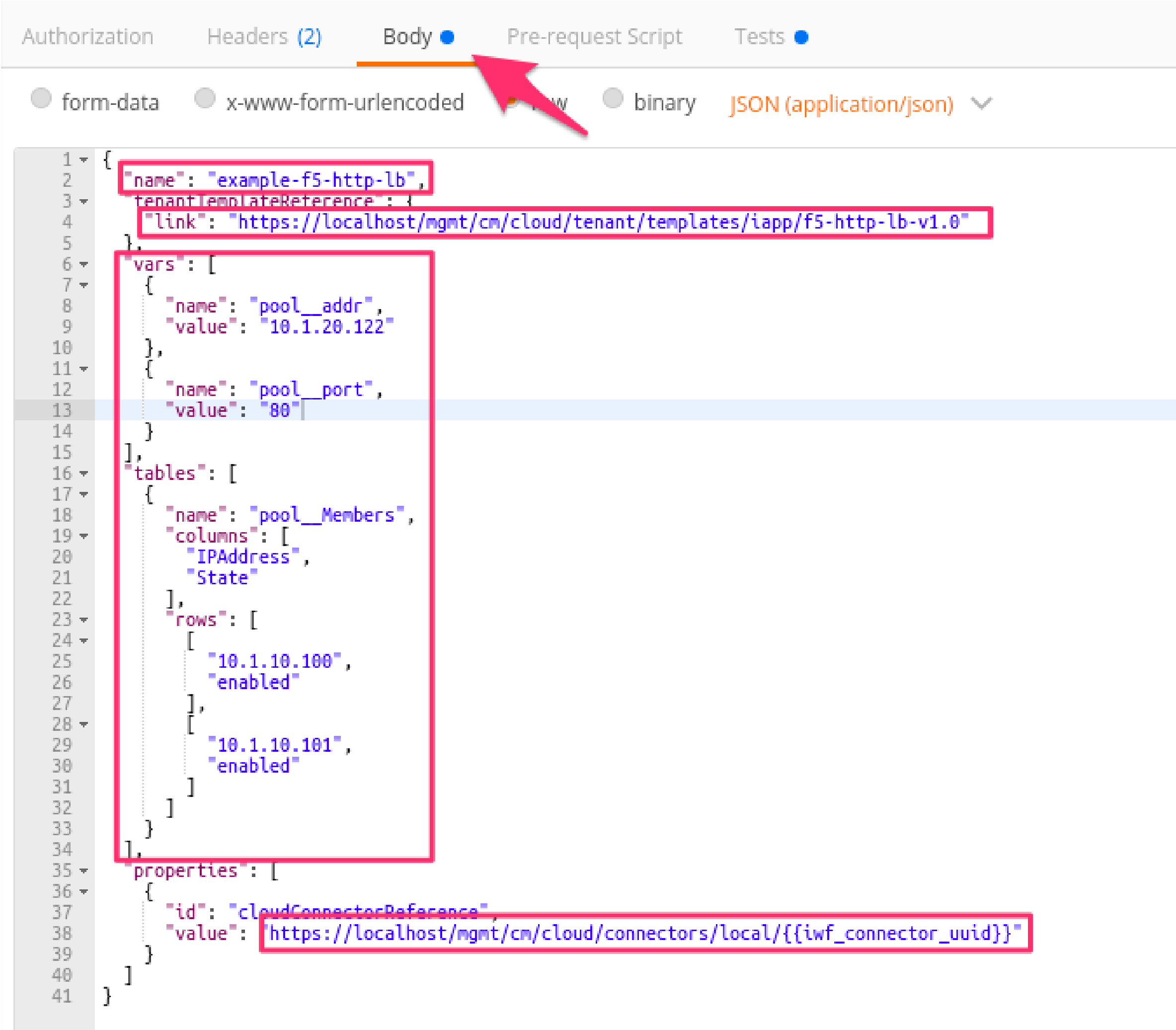
Deploy example-f5-http-lb Service

POST https://{{iwf_mgmt}}/mgmt/cm/cloud/tenants/{{iwf_tenant_name}}/services/iapp

repository_branch	v3.0
iwf_bigip_b_uuid	63f31f13-4c6d-45a6-b3b1-0482d251e3fd
iwf_tenant_name	MyTenant
iwf_appsvcs_name	appsvcs_integration_v2.0.004
iwf_tenant_auth_token	ZVDNWMQC2PFL5Q3QJK3DZBTXE7

3. Examine the JSON Request *Body*; it contains the following data:
   - Deployment `name`
   - A URI Reference to the Service Template `f5-http-lb-v1.0`
   - The input `vars` and `tables` for the deployment. These fields were marked `Tenant Editable` in the Service Template
   - A URI Reference to the Connector to use for deployment. This specifies which BIG-IP devices will be used for this deployment

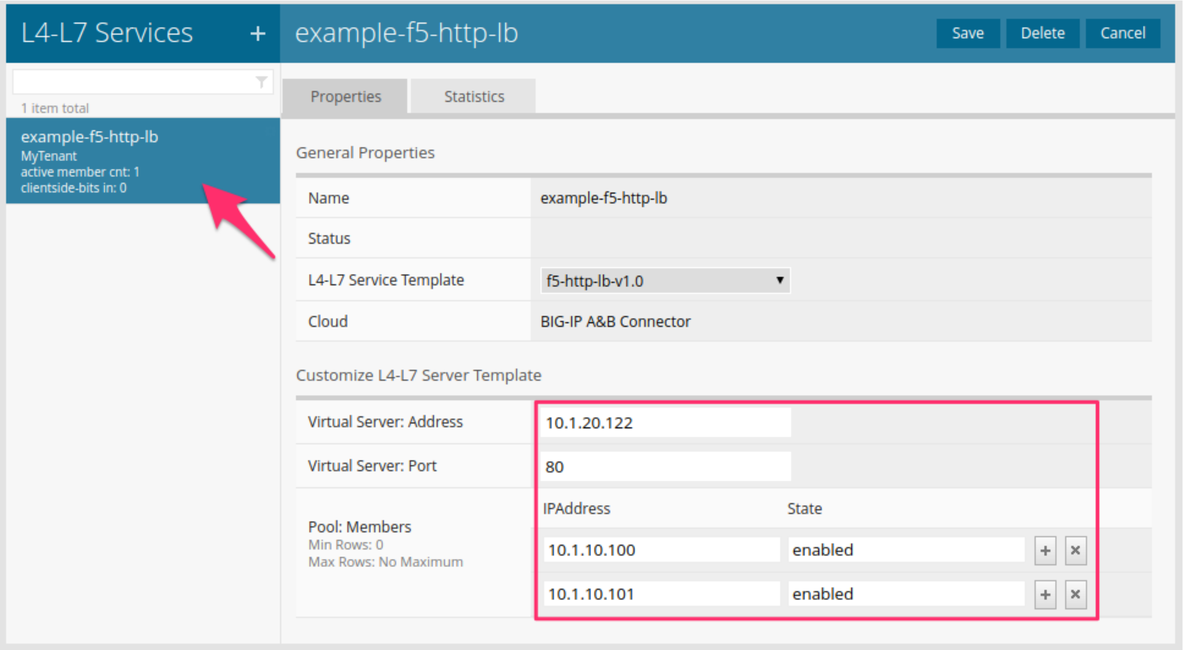
   The data in the list above is higlighted below:



```
{
  "name": "example-f5-http-lb",
  "tenantTemplateReference": {
    "link": "https://localhost/mgmt/cm/cloud/tenant/templates/iapp/f5-http-lb-v1.0"
  },
  "vars": [
    {
      "name": "pool__addr",
      "value": "10.1.20.122"
    },
    {
      "name": "pool__port",
      "value": "80"
    }
  ],
  "tables": [
    {
      "name": "pool__Members",
      "columns": [
        "IPAddress",
        "State"
      ],
      "rows": [
        [
          "10.1.10.100",
          "enabled"
        ],
        [
          "10.1.10.101",
          "enabled"
        ]
      ]
    }
  ],
  "properties": [
    {
      "id": "cloudConnectorReference",
      "value": "https://localhost/mgmt/cm/cloud/connectors/local/{{iwf_connector_uuid}}"
    }
  ]
}
```

4. Click the *Send* button to **Create** the Service Deployment

5. Switch to the Chrome iWorkflow Tenant UI window. The `example-f5-http-lb` Service is now present in the *L4-L7 Services* pane. Double click the Service and examine its properties. You can compare the values in the UI to the JSON Request *Body* from the step above.



example-f5-http-lb

1 item total

example-f5-http-lb
MyTenant
active member cnt: 1
clientside-bits in: 0

General Properties

Name	example-f5-http-lb
Status	
L4-L7 Service Template	f5-http-lb-v1.0
Cloud	BIG-IP A&B Connector

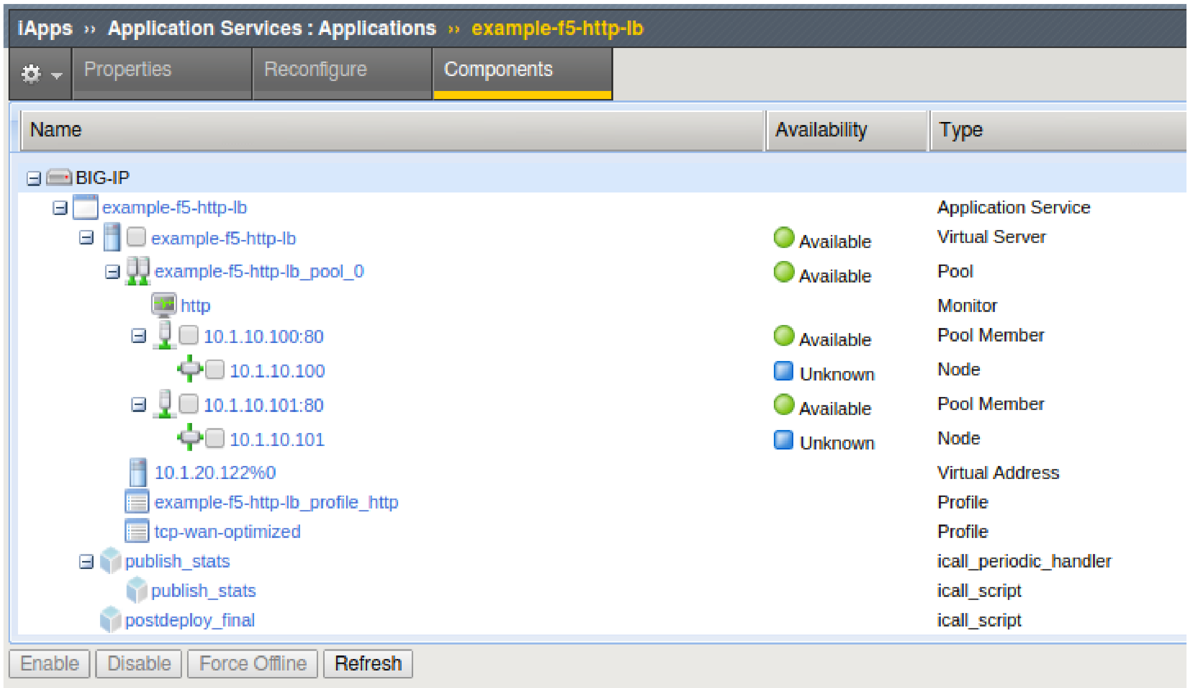
Customize L4-L7 Server Template

Virtual Server: Address	10.1.20.122
Virtual Server: Port	80

Pool: Members
Min Rows: 0
Max Rows: No Maximum

IPAddress	State
10.1.10.100	enabled
10.1.10.101	enabled

6. Open a Chrome window/tab to the BIG-IP A GUI at `https://10.1.1.10` and login with `admin/admin` credentials. Navigate to *iApps* → *Application Services*. Select `example-f5-http-lb` from the list of deployed services and examine the *Components* of the deployed service:



iApps ›› Application Services : Applications ›› example-f5-http-lb
Properties
Reconfigure
Components
Enable
Disable
Force Offline
Refresh

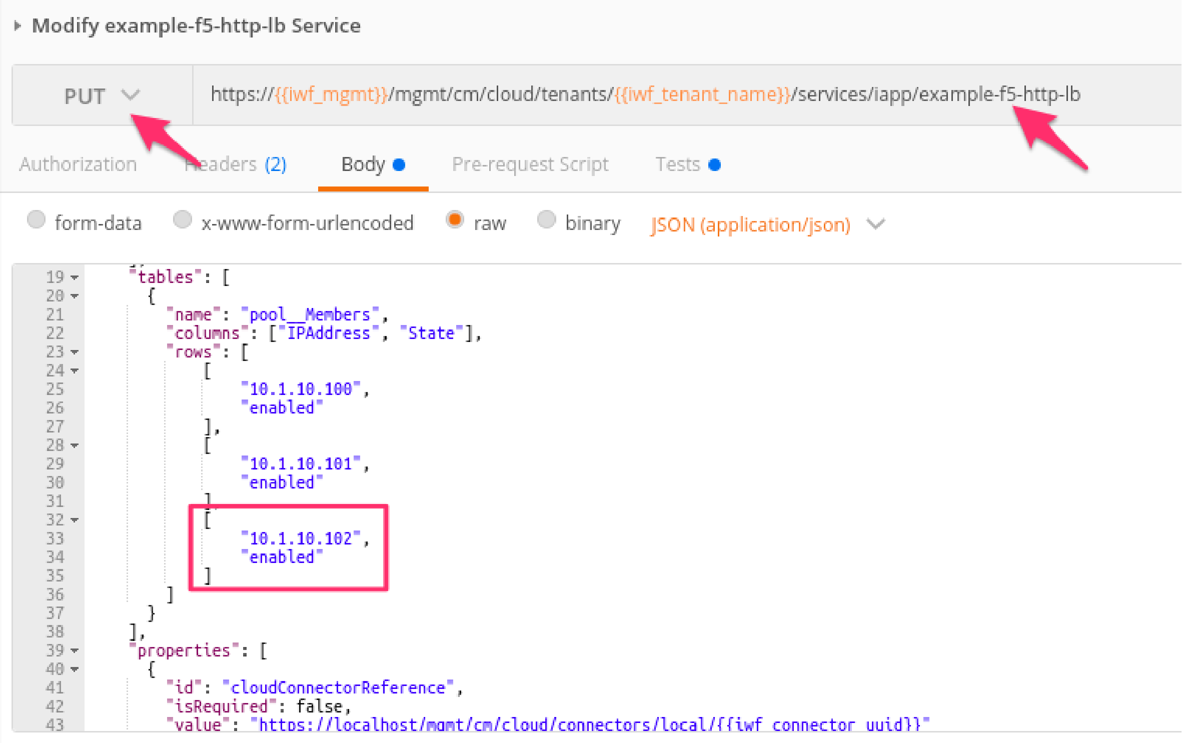
Name	Availability	Type
BIG-IP		
example-f5-http-lb		Application Service
example-f5-http-lb	Available	Virtual Server
example-f5-http-lb_pool_0	Available	Pool
http		Monitor
10.1.10.100:80	Available	Pool Member
10.1.10.100	Unknown	Node
10.1.10.101:80	Available	Pool Member
10.1.10.101	Unknown	Node
10.1.20.122%0		Virtual Address
example-f5-http-lb_profile_http		Profile
tcp-wan-optimized		Profile
publish_stats		icall_periodic_handler
publish_stats		icall_script
postdeploy_final		icall_script

Update

Perform the following steps to complete this task:

1. Click the `Modify example-f5-http-lb Service` request in the folder.

2. We will send a `PUT` request to the Resource URI for the existing deployment and add a Pool Member as shown in the JSON Request *Body*:

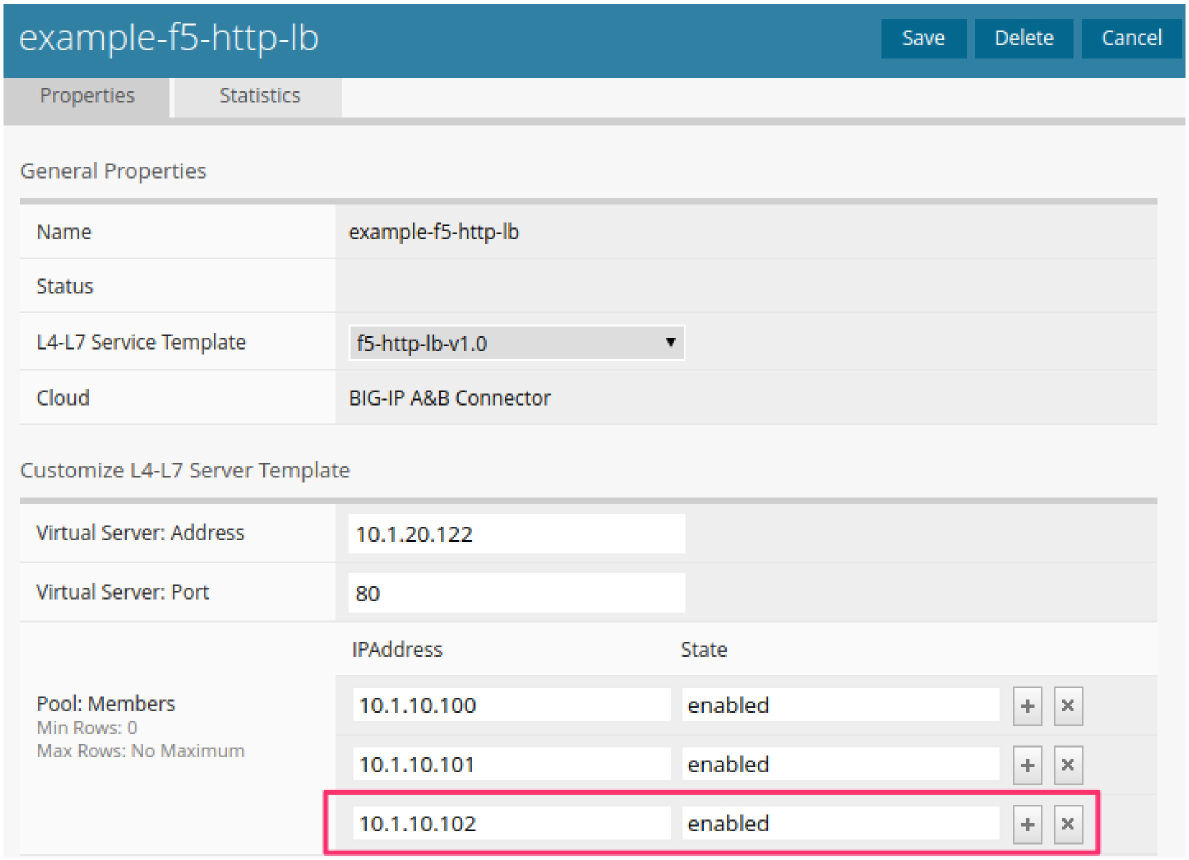


```
https://{{iwf_mgmt}}/mgmt/cm/cloud/tenants/{{iwf_tenant_name}}/services/iapp/example-f5-http-lb
```

```
    "tables": [
      {
        "name": "pool__Members",
        "columns": ["IPAddress", "State"],
        "rows": [
          [
            "10.1.10.100",
            "enabled"
          ],
          [
            "10.1.10.101",
            "enabled"
          ]
          [
            "10.1.10.102",
            "enabled"
          ]
        ]
      }
    ],
    "properties": [
      {
        "id": "cloudConnectorReference",
        "isRequired": false,
        "value": "https://localhost/mgmt/cm/cloud/connectors/local/{{iwf_connector_uuid}}"
```

3. Click the *Send* button to **Update** the Service Deployment

4. Update the iWorkflow Tenant UI and notice that the Service has been updated:



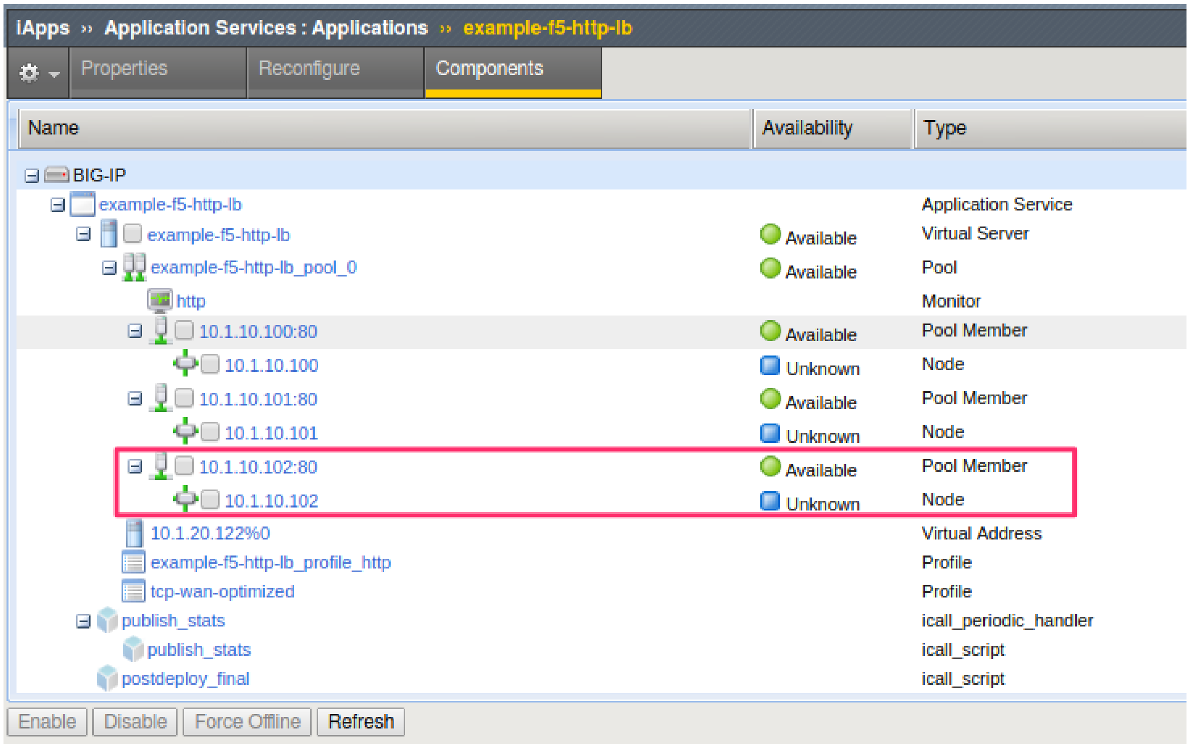
General Properties

Name	example-f5-http-lb
Status	
L4-L7 Service Template	f5-http-lb-v1.0 ▾
Cloud	BIG-IP A&B Connector

Customize L4-L7 Server Template

Virtual Server: Address	10.1.20.122		
Virtual Server: Port	80		
	IPAddress	State	
Pool: Members Min Rows: 0 Max Rows: No Maximum	10.1.10.100	enabled	+ ×
	10.1.10.101	enabled	+ ×
	10.1.10.102	enabled	+ ×

5. Update the BIG-IP GUI and notice that the *Components* tree has been updated:

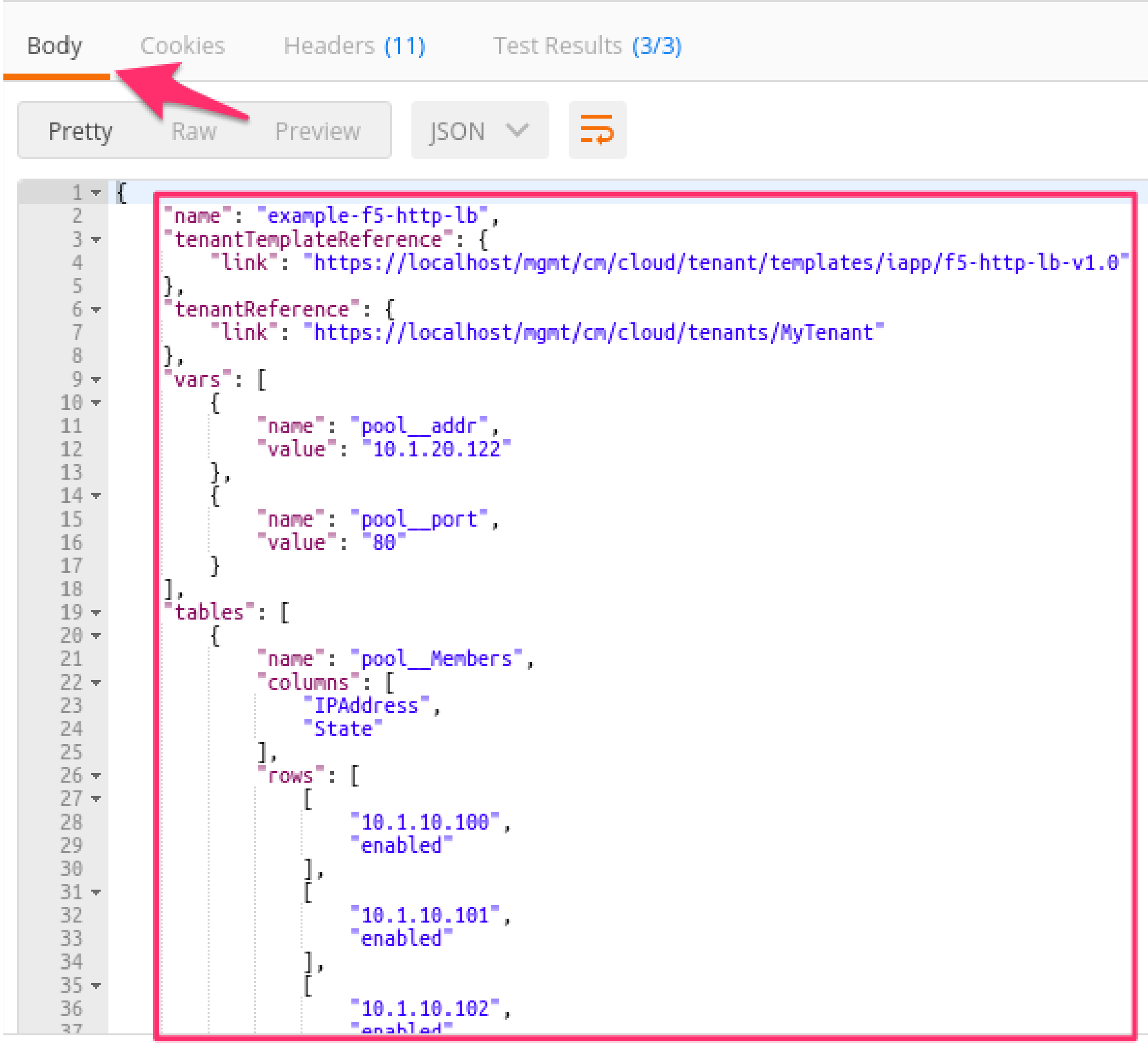

iApps ›› Application Services : Applications ›› example-f5-http-lb
Properties
Reconfigure
Components
Enable
Disable
Force Offline
Refresh

Name	Availability	Type
⊟ BIG-IP		
⊟ example-f5-http-lb		Application Service
⊟ example-f5-http-lb	● Available	Virtual Server
⊟ example-f5-http-lb_pool_0	● Available	Pool
http		Monitor
⊟ 10.1.10.100:80	● Available	Pool Member
10.1.10.100	■ Unknown	Node
⊟ 10.1.10.101:80	● Available	Pool Member
10.1.10.101	■ Unknown	Node
⊟ 10.1.10.102:80	● Available	Pool Member
10.1.10.102	■ Unknown	Node
10.1.20.122%0		Virtual Address
example-f5-http-lb_profile_http		Profile
tcp-wan-optimized		Profile
⊟ publish_stats		icall_periodic_handler
publish_stats		icall_script
postdeploy_final		icall_script

Read

Perform the following steps to complete this task:

1. Click the `Get example-f5-http-lb Service` request in the folder.
2. We will send a `GET` request to the Resource URI for the existing deployment.
3. Click the *Send* button to **Read** the Service Deployment
4. Examine the JSON Response *Body* to see the state of the current Service Deployment:



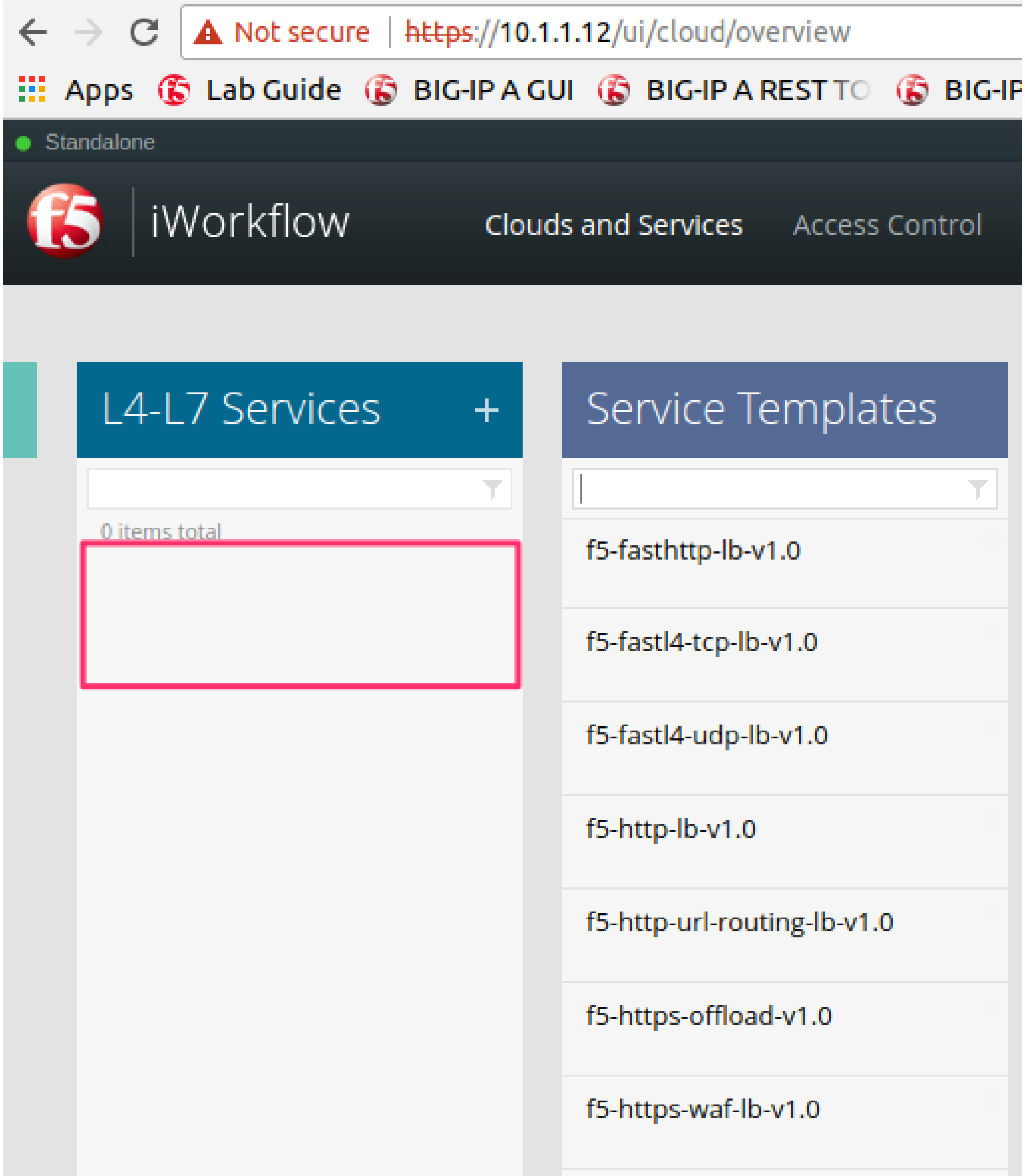
```
{
    "name": "example-f5-http-lb",
    "tenantTemplateReference": {
        "link": "https://localhost/mgmt/cm/cloud/tenant/templates/iapp/f5-http-lb-v1.0"
    },
    "tenantReference": {
        "link": "https://localhost/mgmt/cm/cloud/tenants/MyTenant"
    },
    "vars": [
        {
            "name": "pool__addr",
            "value": "10.1.20.122"
        },
        {
            "name": "pool__port",
            "value": "80"
        }
    ],
    "tables": [
        {
            "name": "pool__Members",
            "columns": [
                "IPAddress",
                "State"
            ],
            "rows": [
                [
                    "10.1.10.100",
                    "enabled"
                ],
                [
                    "10.1.10.101",
                    "enabled"
                ],
                [
                    "10.1.10.102",
                    "enabled"
```

Delete

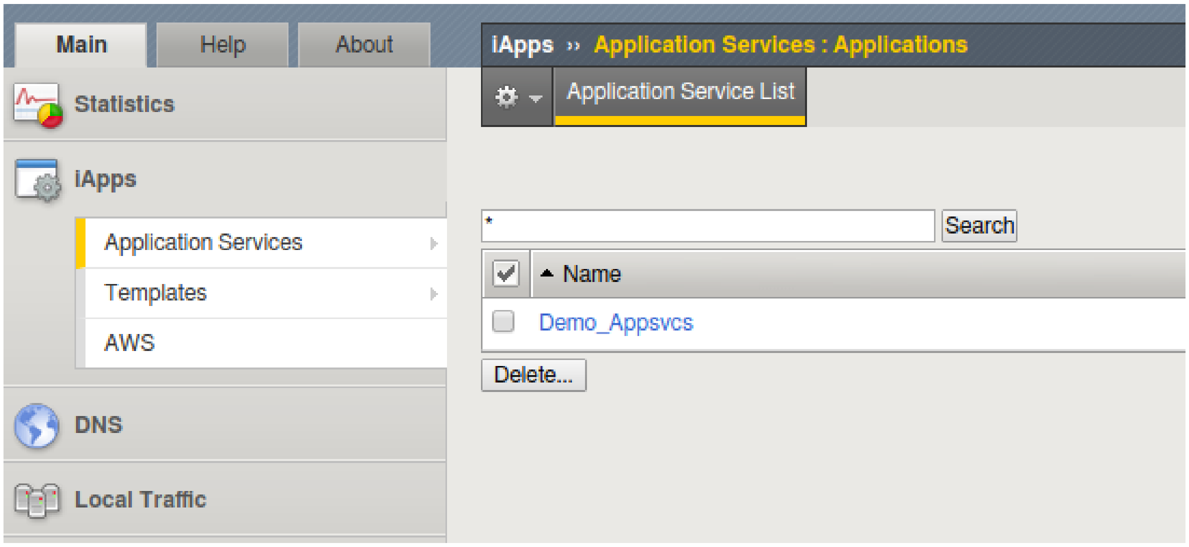
Perform the following steps to complete this task:

1. Click the `Delete example-f5-http-lb Service` request in the folder.
2. We will send a `DELETE` request to the Resource URI for the existing deployment.
3. Click the *Send* button to **Delete** the Service Deployment

4. Update the iWorkflow Tenant UI and verify that the Service has been deleted:



5. In the BIG-IP GUI navigate to *iApps* → *Application Services* and verify the service was deleted.

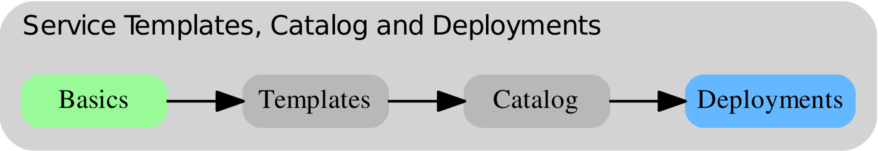


Task 3 - Deploy Additional Services

Examples **Create** requests are included in the `Lab 3.3 – Deploy L4-7 Services` folder. For the remaining services refer to the table below to see which ones apply most to your specific use cases. You can repeat the steps in Task 2 for the additional services by modifying the requests as needed.

Service Name	Description
`f5-http-lb`	HTTP Load Balancing to a Single Pool
`f5-https-offload`	HTTPS Offload and Load Balancing to a Single Pool
`f5-fasthttp-lb`	Performance-enhanced HTTP Load Balancing to a Single Pool
`f5-fastl4-udp-lb`	Generic L4 TCP Load Balancing to a Single Pool
`f5-fastl4-udp-lb`	Generic L4 UDP Load Balancing to a Single Pool
`f5-http-url-routing-lb`	HTTP Load Balancing with URL Based Content Routing to Multiple Pools
`f5-https-waf-lb`	HTTPS Offload, Web Application Firewall Protection and Load Balancing to a Single Pool

2.5.4 Lab 3.4: iWorkflow REST Proxy



While the focus so far has been on building **Declarative Interfaces** with iWorkflow, it's important to note iWorkflow can also help simplify **Imperative** operations to BIG-IP devices when needed.

iWorkflow includes a REST proxy that allows pass-through of REST requests to devices discoverd on iWorkflow. The REST proxy feature allows customers to simplify **Imperative** Automation by:

- Providing a centralized API endpoint for BIG-IP infrastructure
  - No need to communicate with individual BIG-IP devices, only with iWorkflow
- Simplified authentication
  - Strong authentication can be implemented at iWorkflow rather than on each BIG-IP
- Simplified RBAC
  - RBAC can be implemented at iWorkflow for all devices rather than on individual devices in the environment

The REST proxy works by passing data sent to a specific URL through to the BIG-IP device. The root URL for a particular devices REST proxy is:

```
/mgmt/shared/resolver/device-groups/cm-cloud-managed-devices/devices/
<device\_uuid>/rest-proxy/
```

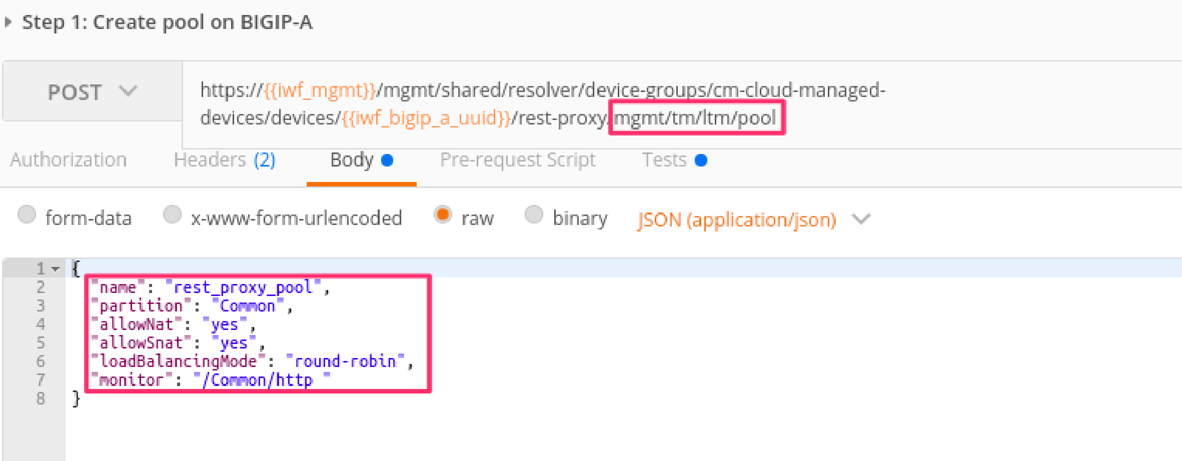
Any URL segments included after `.../rest-proxy/` are forwarded unaltered to the BIG-IP device. Query parameters (e.g. `?expandSubcollections=true`) are also passed unaltered along with the request type and request body.

Task 1 - Perform REST operations via the REST Proxy

In this task we will perform a sample CRUD operation utilizing the REST Proxy. The intent of this task is to show the basic mechanism used to perform these tasks. Simply changing the URL to include the iWorkflow REST Proxy root for that device could easily change all the **Imperative** operations we have completed in this lab to use the REST Proxy.

Perform the following steps to complete this task:

1. Expand the `Lab 3.4 – iWorkflow REST Proxy` folder in the Postman collection.
2. Click the `Step 1: Create pool on BIG-IP A`. Examine the request type, URL and JSON body. Essentially we are performing a POST to the '/mgmt/tm/ltm/pool' collection on BIG-IP A. The last part of the URL includes this URI path (the part after `.../rest-proxy/`). The JSON body and all other parameters are passed unaltered. Also, notice that we are still using our iWorkflow Token to Authenticate, not the BIG-IP one in the *Headers* tab.



Step 1: Create pool on BIGIP-A

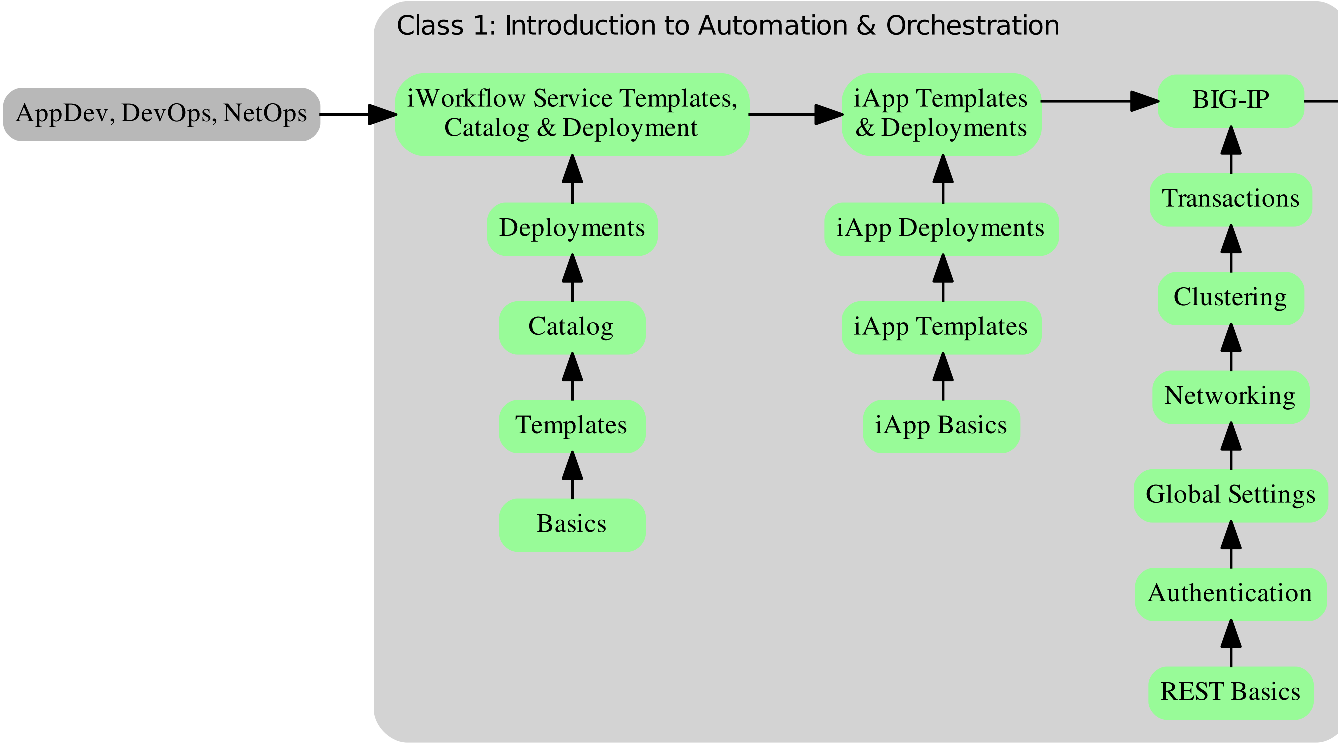
POST https://{{iwf_mgmt}}/mgmt/shared/resolver/device-groups/cm-cloud-managed-devices/devices/{{iwf_bigip_a_uuid}}/rest-proxy/mgmt/tm/ltm/pool

○ form-data ○ x-www-form-urlencoded ● raw ○ binary JSON (application/json)

```
{
  "name": "rest_proxy_pool",
  "partition": "Common",
  "allowNat": "yes",
  "allowSnat": "yes",
  "loadBalancingMode": "round-robin",
  "monitor": "/Common/http "
}
```

3. Click the *Send* button and examine the response.
4. Complete Steps 2-5 for the remaining items in the `Lab 2.5 - iWorkflow REST Proxy` collection. Examine each request carefully so you understand what is happening.

2.6 Conclusion



In this class we learned the base concepts and skills required to effectively automate the F5 BIG-IP platform. The diagram above shows a high-level view of the different components to this base level of knowledge. In subsequent classes we will expand on the core concepts and knowledge learned in this class.

This content has been created with a DevOps methodology and fully Continuous Toolchain. All content contained here is sourced from the following GitHub repository:

https://github.com/f5devcentral/f5-automation-labs/

Bug Reports and Requests for Enhancement are handled in two ways:

- Fork the Github Repo, fix or enhance as required, and submit a Pull Request
  - https://help.github.com/articles/creating-a-pull-request-from-a-fork/
- Open an Issue within the repository.

Lastly, this content would not be possible without the contributions from many F5 Employees, Partners, and Customers. A full list of contributors to this content can be found at:

https://github.com/f5devcentral/f5-automation-labs/graphs/contributors

3 Class 2: Building Continuous Delivery Pipelines

This class covers the following topics:

- Continuous Integration(CI) and Continuous Delivery(CD) Concepts
- F5 Automation Toolkits:
  - F5-Super-NetOps-Container
  - F5 Postman Collections and f5-postman-workflows extensions
  - F5 f5-newman-wrapper for Automating Workflows
- Building CI/CD Pipelines with Jenkins
- Team Collaboration with Automated Slack Notifications

Expected time to complete: **3 hours**

3.1 Module 1: f5-super-netops-container Toolkit

In this module, we will explore how to use the **f5-super-netops-container** toolkit to easily integrate various F5 Automation, Orchestration, Super Netops and DevOps tools, along with framework technologies.

The f5-super-netops-container is meant to provide a simple way for users to quickly duplicate a standard automation and orchestration environment in your local machine/lab environment. The container is **continuously updated** to include the latest tools and documentation.

The labs in this module will show you how to install the f5-super-netops-container image, start it in your local environment and access various tools and documentation.

To install the f5-super-netops-container, your system must support running Docker Community Edition (CE). Please refer to https://docs.docker.com/engine/installation/#platform-support-matrix for more information.

This toolkit is fully open source and is on GitHub at https://github.com/f5devcentral/f5-super-netops-container

3.1.1 Lab 1.1: Install Docker Community Edition (CE)

To use the f5-super-netops-container you first need to install Docker Community Edition on your system.

Note: If you are using an F5 provided lab environment, RDP to the **Windows Jumphost** from there you can access the already installed Docker CE service on the host named 'Docker Server'. SSH to the Docker Sercer via Putty to that host a execute all `docker` commands there.

Task 1 - Install Docker CE

Note: User Credentials to Docker Server: User `root` and Password `default`

Please follow the instructions at https://docs.docker.com/engine/installation/ to install Docker CE.

Once you have completely installed, and successfully run the `hello-world` test you can continue to the next lab.

To test your setup with the `hello-world` container, you just need to run the following command

`docker run --rm hello-world`

Example output:

```
$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

Note: The `--rm` option will delete the container as soon as it stops running.

If you see this message: **Cannot connect to the Docker daemon. Is the docker daemon running on this host?**, it is likely that you don't have enough privileges with your user, try to use **sudo** when executing docker commands.

If you want to remove the hello-world container, you can run the command `sudo docker rmi hello-world` If your container is running, you cannot remove the image. You can issue the following commands in that case (this will stop ALL your container instances): `sudo docker stop $(docker ps -aq)`

3.1.2 Lab 1.2: Obtain & Start the f5-super-netops-container Image

In this lab we will use the `docker` cli tools to obtain and start the f5-super-netops-container image.

Task 1 - Obtain and verfiy the container image

Perform the following steps to complete this task:

1. Open a Command Prompt

Note: If you are using an F5 provided lab environment please SSH to the 'Docker Server' host and execute the following commands.

---

1. Execute `docker pull f5devcentral/f5-super-netops-container:jenkins`

   Example output:

```
$ docker pull f5devcentral/f5-super-netops-container:jenkins
jenkins: Pulling from f5devcentral/f5-super-netops-container
019300c8a437: Pull complete
2d6b38b56ae7: Pull complete
5fab9174d5b4: Pull complete
fc0251c85d81: Pull complete
d5c1476cba25: Pull complete
3f563aeb530f: Pull complete
56717b902584: Pull complete
3a973f5ee17d: Pull complete
68d52f474d41: Pull complete
604d6366bf0b: Pull complete
b3b4184aef22: Pull complete
2cebe1f5955c: Pull complete
2b7bce9d0d9e: Pull complete
259f696f7766: Pull complete
6d5f2e57c5b3: Pull complete
985706ad6d05: Pull complete
a29f68892227: Pull complete
7420ee096abd: Pull complete
0907797bbe90: Pull complete
5b8f2518bf01: Pull complete
2940be145e35: Pull complete
f2cb35cbf665: Pull complete
5cdfa1779954: Pull complete
61c1367b68d8: Pull complete
5bcd8c5223bb: Pull complete
b0defdb83b82: Pull complete
Digest: sha256:27563f98bf58c9d26eb5989acaf540a9ad7fb1806e4a4c373ad28769ebe63ef4
Status: Downloaded newer image for f5devcentral/f5-super-netops-container:jenkins
```

2. Execute `docker images`

   Example output:

```
$ docker images
REPOSITORY                               TAG                 IMAGE ID           ␣
 ↪CREATED             SIZE
f5devcentral/f5-super-netops-container   jenkins             b71fc40407e4       ␣
 ↪2 weeks ago         490MB
```

Task 2 - Start the container image

To start using the container we will execute the command:

1. Execute `docker run -p 8080:80 -p 2222:22 -p 10000:8080 --rm -it -e SNOPS_GH_BRANCH=master f5devcentral/f5-super-netops-container:jenkins`

Note: The image requires Internet connectivity to download the latest versions of tools and documentation. Please ensure you have proper connectivity from your host prior to starting the image. If you need to use a proxy please refer to the documentation at https://docs.docker.com

The image will now start and load resources from the Internet. This process may take a while depending on the speed of your connection. When the startup process is complete you will be presented in the `root` user prompt. You can interact with the image with standard Linux commands. In the next lab we will connect to the image via SSH and HTTP.

The `-p` option publishes a L4 port from the container to the host. For example the `-p 8080:80` option will redirect port `8080` on the host system to port `80` in the container.

The `-it` option will make the session interactive and allocate a pseudo-TTY

The `-e` option will specify a Github Branch, in this case we are pulling from `master`

The `f5devcentral/f5-super-netops-container:jenkins` option is the name associated with the image we obtained in Task 1.

Example startup output:

```
container:jenkins
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] done.
[services.d] starting services
[services.d] done.
[environment] SNOPS_HOST_SSH=2222
[environment] SNOPS_REPO=https://github.com/f5devcentral/f5-super-netops-container.git
[environment] SNOPS_AUTOCLONE=1
[environment] SNOPS_HOST_IP=172.17.0.2
[environment] SNOPS_ISALIVE=1
[environment] SNOPS_GIT_HOST=github.com
[environment] SNOPS_REVEALJS_DEV=0
[environment] SNOPS_HOST_HTTP=8080
[environment] SNOPS_IMAGE=jenkins
[environment] SNOPS_GH_BRANCH=master
Reticulating splines...
Becoming self-aware...
[cloneGitRepos] Retrieving repository list from https://github.com/f5devcentral/f5-
↪super-netops-container.git#develop
[updateRepos] Processing /tmp/snops-repo/images/jenkins/fs/etc/snopsrepo.d/jenkins.
↪json
[updateRepos]  Processing /tmp/snops-repo/images/base/fs/etc/snopsrepo.d/base.json
[updateRepos] Processing /tmp/user_repos.json
[cloneGitRepos] Loading repositories from /home/snops/repos.json
[cloneGitRepos] Found 7 repositories to clone...
[cloneGitRepos][1/7] Cloning f5-sphinx-theme#master from https://github.com/
↪f5devcentral/f5-sphinx-theme.git
[cloneGitRepos][1/7]  Installing f5-sphinx-theme#master
```

```
[cloneGitRepos][2/7] Cloning f5-super-netops-container#develop from https://github.
↪com/f5devcentral/f5-super-netops-container.git
[cloneGitRepos][2/7]  Installing f5-super-netops-container#develop
[cloneGitRepos][3/7] Cloning f5-application-services-integration-iApp#develop from␣
↪https://github.com/F5Networks/f5-application-services-integration-iApp.git
[cloneGitRepos][3/7]  Installing f5-application-services-integration-iApp#develop
[cloneGitRepos][4/7] Cloning f5-postman-workflows#develop from https://github.com/
↪0xHiteshPatel/f5-postman-workflows.git
[cloneGitRepos][4/7]  Installing f5-postman-workflows#develop
[cloneGitRepos][5/7] Cloning f5-automation-labs#master from https://github.com/
↪f5devcentral/f5-automation-labs.git
[cloneGitRepos][5/7]  Installing f5-automation-labs#master
[cloneGitRepos][6/7] Cloning ultimate-vimrc#master from https://github.com/amix/vimrc.
↪git
[cloneGitRepos][6/7]  Installing ultimate-vimrc#master
[cloneGitRepos][7/7] Cloning reveal-js#master from https://github.com/hakimel/reveal.
↪js.git
[cloneGitRepos][7/7]  Installing reveal-js#master
                                .----------.
                               /          /
                              /   ______.'
                        _.._ /   /_
                      .' .._/      '''--.
                      | '  '___          `.
                    __| |__    `'.         |
                   |__   __|      )        |
                      | | ......-'        /
                      | | \          _..'`
                      | |  '------'''
                      | |                      _
                      |_|                     | |
 ___ _   _ _ __   ___ _ __          _ __   ___| |_ ___  _ __  ___
/ __| | | | '_ \ / _ \ '__| ______ | '_ \ / _ \ __/ _ \| '_ \/ __|
\__ \ |_| | |_) |  __/ |   |______|| | | |  __/ || (_) | |_) \__ \
|___/\__,_| .__/ \___|_|           |_| |_|\___|\__\___/| .__/|___/
          | |                                          | |
          |_|                                          |_|

Welcome to the f5-super-netops-container.  This image has the following
services running:

 SSH  tcp/22
 HTTP tcp/80

To access these services you may need to remap ports on your host to the
local container using the command:

 docker run -p 8080:80 -p 2222:22 -it f5devcentral/f5-super-netops-container:base

From the HOST perspective, this results in:

 localhost:2222 -> f5-super-netops-container:22
 localhost:8080 -> f5-super-netops-container:80

You can then connect using the following:

 HTTP: http://localhost:8080
 SSH:  ssh -p 2222 snops@localhost
```

```

Default Credentials:

 snops/default
 root/default

Go forth and automate!

(you can now detach by using Ctrl+P+Q)

[root@f5-super-netops] [/] #
```

Task 3 - Detach/Re-attach the Container

When running containers it’s important to understand that it will exit if the foreground process (in this case the shell) exits. For example, if you typed the `exit` command in the running container it will shutdown. In order to avoid this you should detach from the container once it has completed booting. You can still perform functions by using SSH to access the container as explained in the next lab.

Its likely that the installation of the f5-super-netops-container will not be on a localhost while running in a large environment, the steps below show how you can leave this instance running as a background process, if needed.

Detach the Container

1. Issue a `Ctrl+p+q` in the running TTY.

   Example output:

   ```
   [root@f5-super-netops] [/] #
   [root@f5-super-netops] [/] #
   [root@f5-super-netops] [/] # <enter Ctrl+p+q>
   hostname:~ user$
   ```

2. Verify the container is still running by entering `docker ps`

   Example output:

   ```
   hostname:~ user$ docker ps
   $ docker ps
   CONTAINER ID        IMAGE                                                    ␣
   ↪COMMAND                  CREATED             STATUS              PORTS          ␣
   ↪                                                                               ␣
   ↪NAMES
   4cf75944bfbc        f5devcentral/f5-super-netops-container:jenkins          "/
   ↪init /snopsboot/..."   2 minutes ago       Up 2 minutes        8000/tcp, 50000/
   ↪tcp, 0.0.0.0:2222->22/tcp, 0.0.0.0:8080->80/tcp, 0.0.0.0:10000->8080/tcp  ␣
   ↪loving_montalcini
   ```

Re-attach the Container

1. Execute `docker ps`

Example output:

```
hostname:~ user$ docker ps
$ docker ps
CONTAINER ID        IMAGE                                                          ␣
↪COMMAND                  CREATED             STATUS              PORTS              ␣
↪                                                                                   ␣
↪NAMES
4cf75944bfbc        f5devcentral/f5-super-netops-container:jenkins          "/
↪init /snopsboot/..."   2 minutes ago       Up 2 minutes        8000/tcp, 50000/
↪tcp, 0.0.0.0:2222->22/tcp, 0.0.0.0:8080->80/tcp, 0.0.0.0:10000->8080/tcp  ␣
↪loving_montalcini
|------------|
  ^- YOUR CONTAINER ID
```

2. Copy the value under the `CONTAINER ID` column that correspond to the f5devcentral/f5-super-netops-container:jenkins image.

3. Execute `docker attach <container_id>`

4. You may have to hit `<Enter>` twice to display the command prompt

5. Detach the container again by entering `<Ctrl+p+q>`

3.1.3 Lab 1.3: Connect to f5-super-netops-container

In the previous lab we started the container image and were presented with a root user terminal. In order to use the container and its associated tools properly we will connect via SSH and/or HTTP.

Task 1 - Connect via SSH

To connect to the image via SSH we must use the published port specified in the `docker run` command. **To review** the command used to start the container was:

```
docker run -p 8080:80 -p 2222:22 -p 10000:8080 --rm -it -e
SNOPS_GH_BRANCH=master f5devcentral/f5-super-netops-container:jenkins
```

This will publish the standard SSH service on `TCP/22` to `TCP/2222` on the Docker host. In the case of the SSH service the following mapping applies:

```
localhost:2222 -> f5-super-netops-container:22
```

Note: If you are using an F5 provided lab environment please use the SSH client and connect to the 'f5-super-netops-container SSH' item

The container includes the `snops` user with a password of `default`. If you are not using the F5 Lab environment connect to the container execute the following command or it's OS-specific equivalent:

```
ssh -p 2222 snops@localhost
```

Note: The host SSH keys for our environment are regenerated each time the container boots, you may receive an error when trying to connect indicating the host key has changed. This error is safe to ignore in this case and can be resolved by removing the key from `~/.ssh/known_hosts`. You can also configure your local SSH config by adding the following to `~/.ssh/config`:

```
Host localhost
Port 2222
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
```

Example output of connecting to the container:

```
$ ssh -p 2222 snops@localhost
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.
snops@localhost's password:
                                .----------.
                               /          /
                              /   ______.'
                        _.._ /   /_
                      .' .._/      '''--.
                      | '  '___          `.
                    __| |__    `'.         |
                   |__   __|      )        |
                      | | ......-'        /
                      | | \          _..'`
                      | |  '------'''
                      | |                      _
                      |_|                     | |
 ___ _   _ _ __   ___ _ __          _ __   ___| |_ ___  _ __  ___
/ __| | | | '_ \ / _ \ '__| ______ | '_ \ / _ \ __/ _ \| '_ \/ __|
\__ \ |_| | |_) |  __/ |   |______|| | | |  __/ || (_) | |_) \__ \
|___/\__,_| .__/ \___|_|           |_| |_|\___|\__\___/| .__/|___/
          | |                                          | |
          |_|                                          |_|

Welcome to the f5-super-netops-container.  This image has the following
services running:

 SSH  tcp/22
 HTTP tcp/80

To access these services you may need to remap ports on your host to the
local container using the command:

 docker run -p 8080:80 -p 2222:22 -it f5devcentral/f5-super-netops-container:base

From the HOST perspective, this results in:

 localhost:2222 -> f5-super-netops-container:22
 localhost:8080 -> f5-super-netops-container:80

You can then connect using the following:

 HTTP: http://localhost:8080
 SSH:  ssh -p 2222 snops@localhost

Default Credentials:

 snops/default
 root/default

Go forth and automate!
```

```
[snops@f5-super-netops] [~] $
```

Task 2 - Connect via HTTP

To connect to the image via HTTP we use the published port specified in the `docker run` command. **To review** the command used to start the container was:

```
docker run -p 8080:80 -p 2222:22 -p 10000:8080 --rm -it -e
SNOPS_GH_BRANCH=master f5devcentral/f5-super-netops-container:jenkins
```

This will publish the standard HTTP service on `TCP/80` to `TCP/8080` on the Docker host. In the case of the HTTP service the following mapping applies:

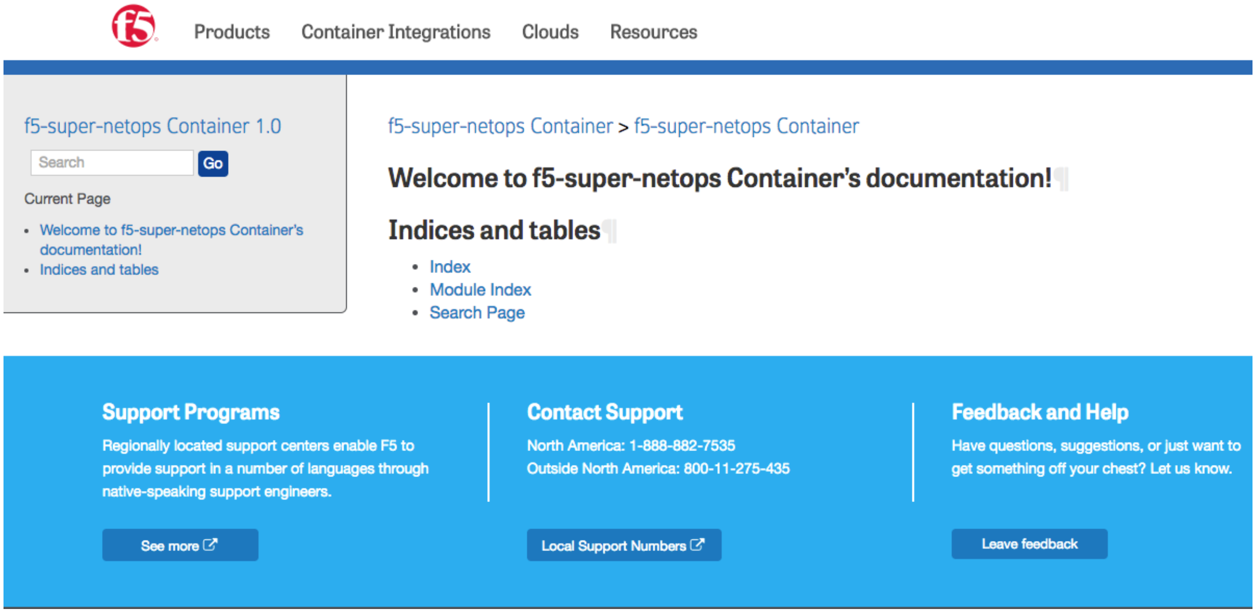
```
localhost:8080 -> f5-super-netops-container:80
```

Note: If you are using an F5 provided lab environment please use the browser and click the 'Super Netops Container' bookmark.

To connect outside of the F5 Lab environment via HTTP, open a web browser and enter the URL:

```
http://(YourDockerSever):8080/start
```

You should see a page like this:



f5-super-netops Container 1.0

Current Page

- Welcome to f5-super-netops Container's documentation!
- Indices and tables

f5-super-netops Container > f5-super-netops Container

Welcome to f5-super-netops Container's documentation!

Indices and tables

- Index
- Module Index
- Search Page

Support Programs

Regionally located support centers enable F5 to provide support in a number of languages through native-speaking support engineers.

Contact Support

North America: 1-888-882-7535
Outside North America: 800-11-275-435

Feedback and Help

Have questions, suggestions, or just want to get something off your chest? Let us know.

Task 3 - Connect via Jenkins

To connect to the image via Jenkins we use the published port specified in the `docker run` command. **To review** the command used to start the container was:

```
docker run -p 8080:80 -p 2222:22 -p 10000:8080 --rm -it -e
SNOPS_GH_BRANCH=master f5devcentral/f5-super-netops-container:jenkins
```

This will publish the standard Jenkins service on `TCP/8080` to `TCP/10000` on the Docker host. In the case of the Jenkins service the following mapping applies:

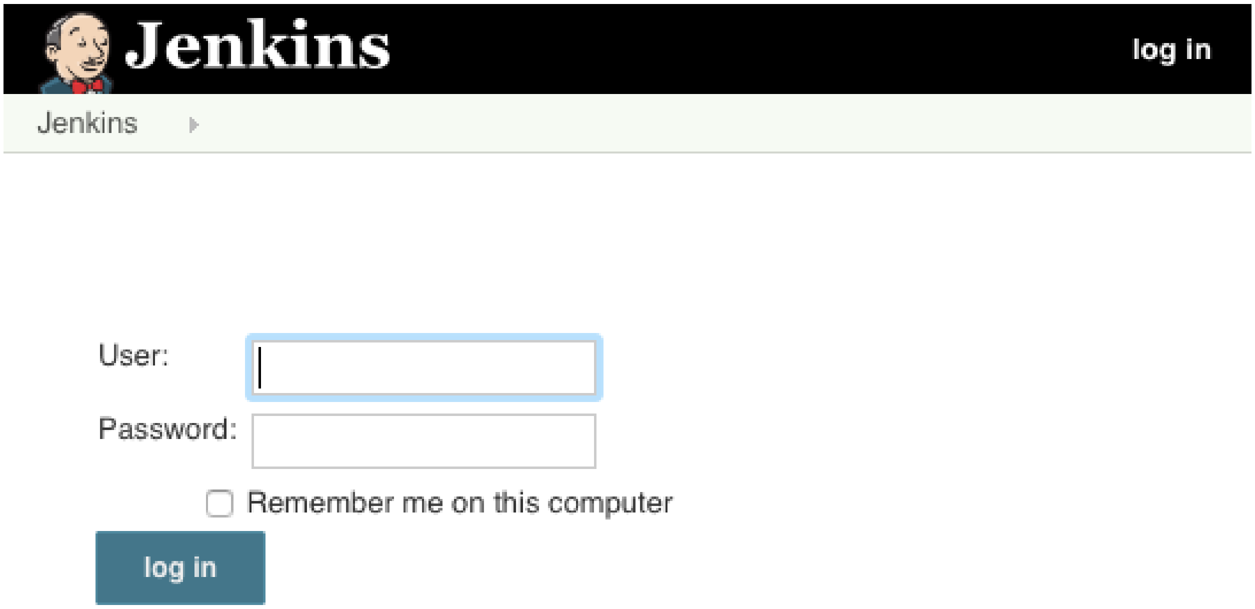
```
10.1.1.8:10000 -> f5-super-netops-container:8080
```

Note: If you are using an F5 provided lab environment please use the browser and click the 'Jenkins' bookmark.

To connect via HTTP open a web browser and enter the URL:

`http://(YourDockerSever):10000`

You should see a page like this:



3.2 Module 2: F5 f5-postman-workflows & f5-newman-wrapper

In the previous Class you may have found the tasks associated with checking various response values and populating environment variables **very** tedious. In addition to being tedious, these tasks are not fundamentally automatable due to the requirement for human interaction.

In order to assist users with automating the F5 BIG-IP platform we have developed a set of tools that can be used with the Postman REST Client (http://getpostman.com). The purpose of the tools are:

- f5-postman-workflows
  - Provide re-usable JavaScript functions that ease testing of API responses and populating environment variables
  - Implement a delay-based polling mechanism
- f5-newman-wrapper
  - Allow users to easily assemble Postman collections into workflows
  - Enabled integration with third-party tools such as Ansible, Chef & Puppet

This framework allows collection developers to create automatable collections that include full testing of response values, population of environment variables to establish chains of requests and time-based polling to allow long-lived API processes time to complete.

Users can then interact with these collections via the Postman GUI client, run the collections with the Postman Runner or the Newman CLI client.

This lab module will walk you through using the tools. If you are interested in developing collections using the f5-postman-workflows framework please visit the official GitHub repository at https://github.com/0xHiteshPatel/f5-postman-workflows

3.2.1 Lab 2.1: Install the f5-postman-workflows Framework

In this lab you will walk through installing the f5-postman-workflows framework into the Postman REST Client.

Task 1 - Import the f5-postman-workflows Postman Collection

In this task you will Import a Postman Collection that contains Installation helpers, Examples and a automated Test Framework. The collection is installed from the f5-postman-workflows GitHub repository.

Perform the following steps to complete this task:

1. Open the Postman Client on your jumphost by clicking the Postman icon
2. Click the 'Import' button in the top left of the Postman window
3. Click the 'Import from Link' tab. Paste the following URL into the text box and click 'Import'

   ```
   https://raw.githubusercontent.com/0xHiteshPatel/f5-postman-workflows/
   master/F5_Postman_Workflows.postman_collection.json
   ```

4. You should now see a collection named `F5_Postman_Workflows` in your Postman Collections sidebar, in some cases the Collections dont appear until Postman has been closed an relaunched.

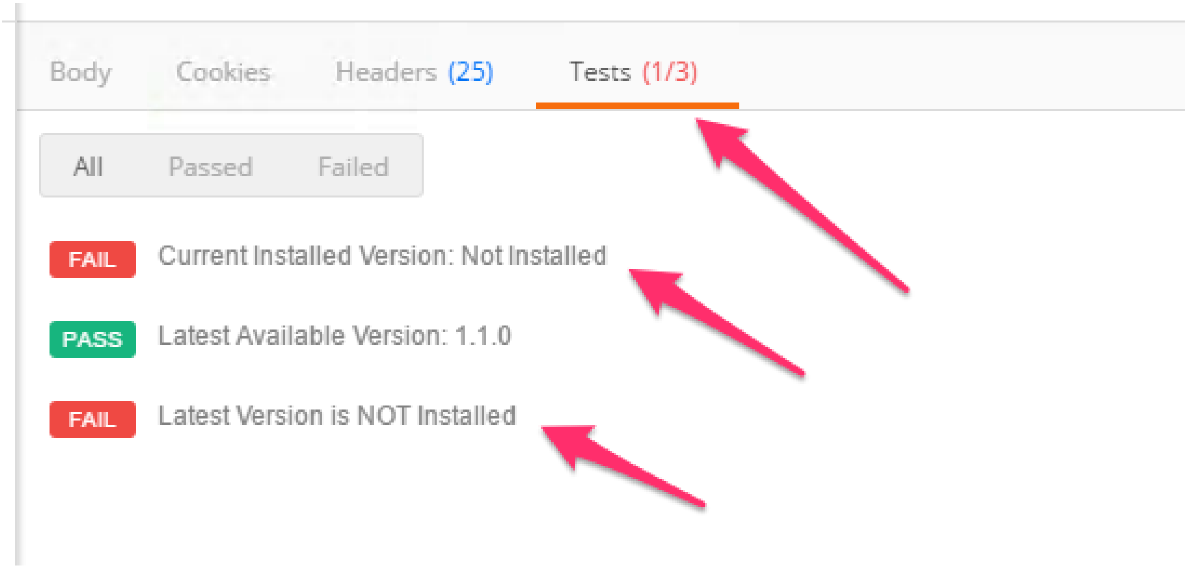
Task 2 - Install f5-postman-workflows into your Postman Client

To utilize the helper functions the framework includes, we must install those functions into the Postman Client. The installation helpers perform the following tasks:

1. Determine the most current version of the framework
2. Dynamically minify the JavaScript code from the f5-postman-workflows GitHub repository using Google's Closure Compiler
3. Install the minified JS code into a Postman Global Variable
4. Set a number of Global variables that allow you to configure various options

To install the framework complete the following tasks:

1. Open the `F5_Postman_Workflows` collection
2. Open the `Install` folder
3. Select the `Check f5-postman-workflows Version` item and click 'Send'
4. Examine the 'Tests' portion of the **RESPONSE**:

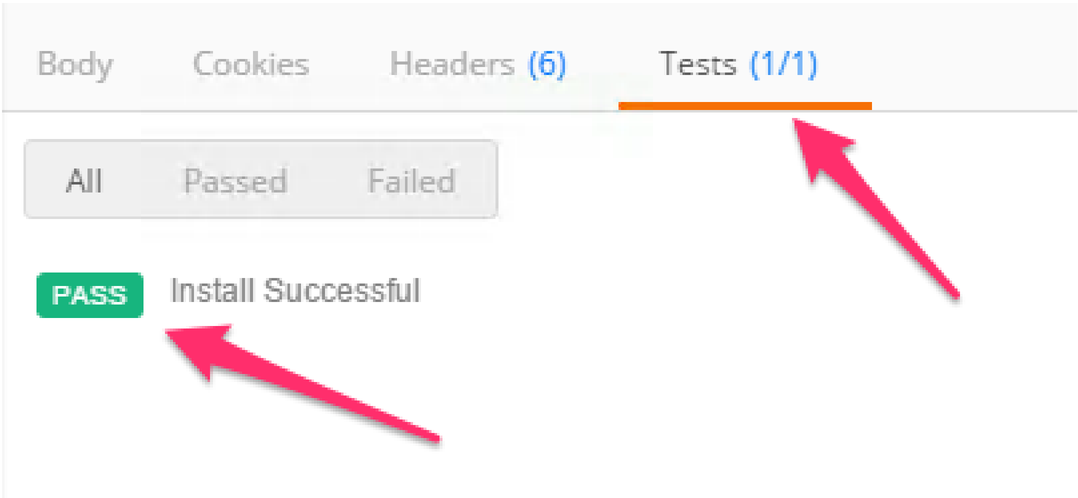


FAIL Current Installed Version: Not Installed

PASS Latest Available Version: 1.1.0

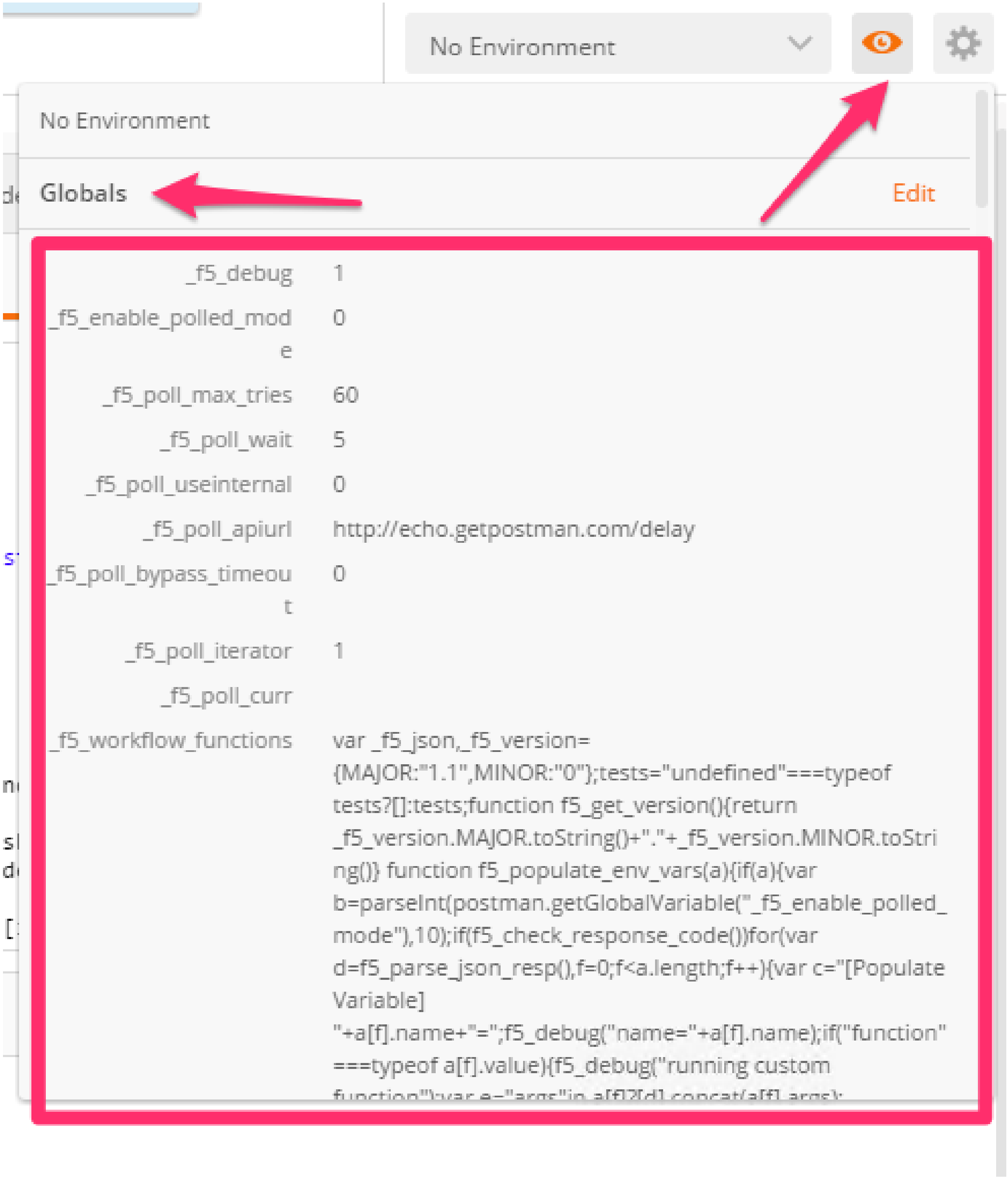
FAIL Latest Version is NOT Installed

5. Select the `Install/Upgrade f5-postman-workflows` item and click ‘Send’
6. Examine the ‘Tests’ again and ensure that Installation was successful:



PASS Install Successful

7. Click the ‘Eye’ button in the top right of the Postman window and examine the Global variables that have been populated



No Environment

Globals

_f5_debug	1
_f5_enable_polled_mode	0
_f5_poll_max_tries	60
_f5_poll_wait	5
_f5_poll_useinternal	0
_f5_poll_apiurl	http://echo.getpostman.com/delay
_f5_poll_bypass_timeout	0
_f5_poll_iterator	1
_f5_poll_curr	
_f5_workflow_functions	var _f5_json,_f5_version={MAJOR:"1.1",MINOR:"0"};tests="undefined"===typeof tests?[]:tests;function f5_get_version(){return _f5_version.MAJOR.toString()+"."+_f5_version.MINOR.toString()} function f5_populate_env_vars(a){if(a){var b=parseInt(postman.getGlobalVariable("_f5_enable_polled_mode"),10);if(f5_check_response_code())for(var d=f5_parse_json_resp(),f=0;f<a.length;f++){var c="[Populate Variable] "+a[f].name+"=";f5_debug("name="+a[f].name);if("function"===typeof a[f].value){f5_debug("running custom function");var e="args"in a[f]?[d].concat(a[f].args);

The f5-postman-workflows framework is now installed in your Postman client.

3.2.2 Lab 2.2: Manually Execute a Workflow

In this lab we will walk through how to obtain two collections, and then we'll use the f5-postman-workflows framework to execute a simple workflow using the Postman GUI client. The f5-postman-workflows GitHub repository is continually updated with new collections that can be used as is, or customized, to automate the F5 platform. Additionally, the f5-super-netops-container automatically downloads these and other tools so users can rapidly execute workflows in their environments.

Postman collections also serve as a reference example of how various tasks can be accomplished using an **Imperative** process. When executing a collection you are actually providing a **Declarative** input to an **Imperative** process.

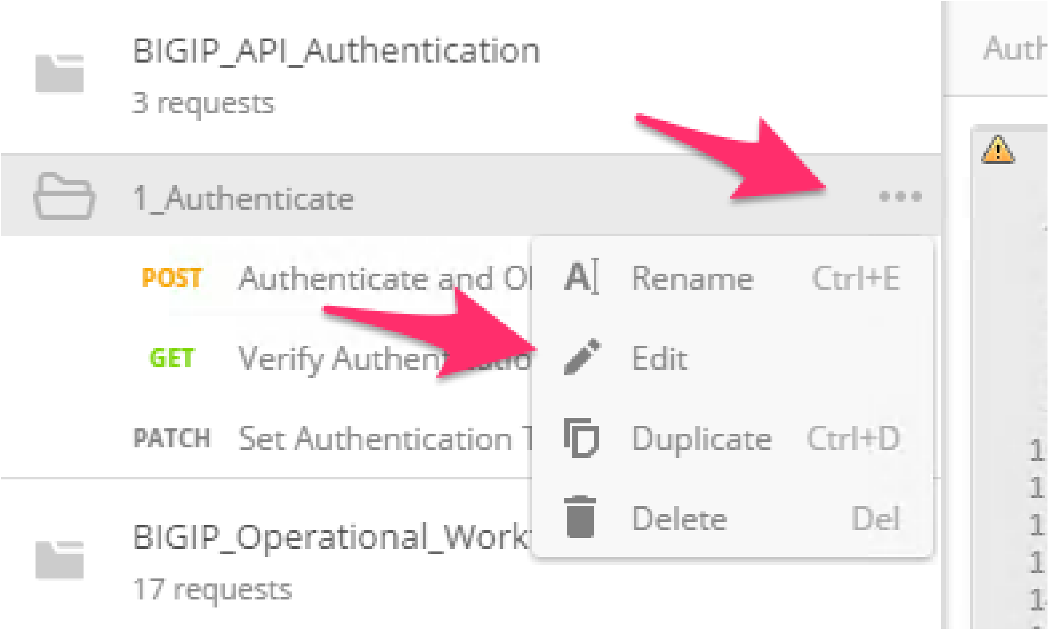
Collections are self-documenting, and we will explore how to access the included documentation to assemble a workflow from beginning to end. In the next lab we will use this base knowledge to create workflows as JSON templates that can be executed by the f5-newman-wrapper on the f5-super-netops-container image (or any system that has Newman installed)

Task 1 - Import and Explore BIG-IP Collections

First, we will import two collections to Postman using the same steps in the previous labs. The **two different** collections will allow us to perform REST **API Authentication** to BIG-IP devices and then execute **Operational** actions on the BIG-IP device. We are stitching together two **Imperative** process's.

Execute the following steps to complete this task:

1. Click Import -> Import from Link and import both of these collection URLs:
    - `https://raw.githubusercontent.com/0xHiteshPatel/f5-postman-workflows/master/collections/BIG_IP/BIGIP_API_Authentication.postman_collection.json`
    - `https://raw.githubusercontent.com/0xHiteshPatel/f5-postman-workflows/master/collections/BIG_IP/BIGIP_Operational_Workflows.postman_collection.json`
2. You should now have two additional Collections in the sidebar:
    - BIGIP_API_Authentication
    - BIGIP_Operational_Workflows
3. Expand the `BIGIP_API_Authentication` collection. Within the collection you will see one folder named `1_Authenticate`. Folders are used to organize various workflows within a collection. In this case this collection performs exactly one task, authentication, therefore one folder is present.
4. Expand the `1_Authenticate` folder. Notice there are three requests in the folder. These three requests will be executed in a synchronous manner (one-after-another).
5. Click the `...` icon on the `1_Authenticate` folder, then click `Edit`



6. In the opened window you will see documentation explaining what the requests in this folder accomplish. Additionally you will see a series of Input and Output variables. Unless marked otherwise it is assumed that all Input variables **are** required. In this case the `bigip_token_timeout` variable is optional.

   Folders may also contain output variables that are set to pass data between requests in different collections (A Waterfall). In this case the output variable is named `bigip_token` and contains the authentication token that can be sent in the `X-F5-Auth-Token` HTTP header to perform authentication.

7. Close the window by clicking 'Cancel'
8. Repeat the steps above and explore the `BIGIP_Operational_Workflows` collection, specifically the `4A_Get_BIGIP_Version` folder

Task 2 - Manually Chain Folders into a Workflow

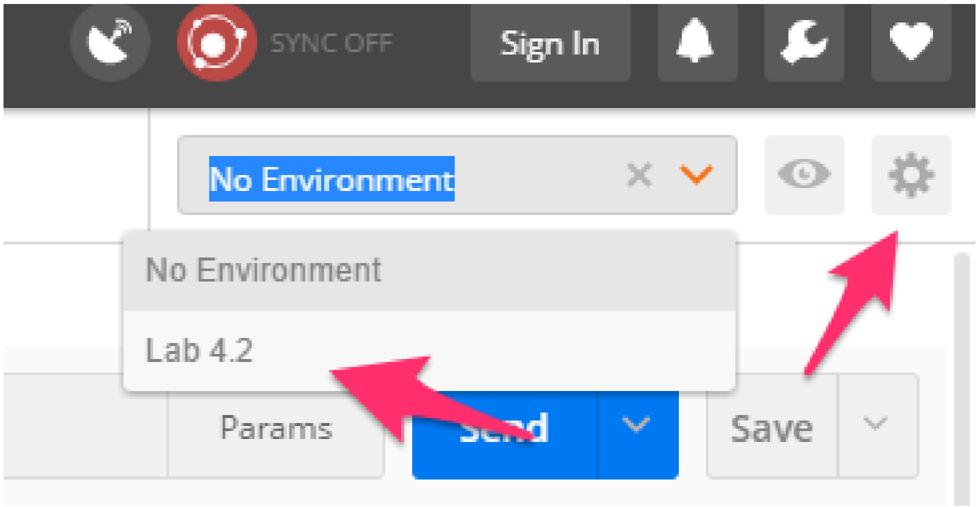
In this task we will explore how to chain two folders together and manually execute a workflow. This example is simple, but should help illustrate how we can use folders as building blocks that can be assembled or chained together to construct a workflow.

We will use the `1_Authenticate` folder in the `BIGIP_API_Authentication` collection and then pass the authentication token to the `4A_Get_BIGIP_Version` folder in the `BIGIP_Operational_Workflows` collection.

Execute the following steps to complete this task:

1. Create a new Postman environment by clicking the Gear icon -> Manage Environments -> Add.
2. Name the environment `Lab 2.2` and populate the following key/value pairs:
   - **bigip_mgmt**: 10.1.1.4
   - **bigip_username**: admin
   - **bigip_password**: admin
3. Click the 'Add' button, then close the 'Manage Environments' window.

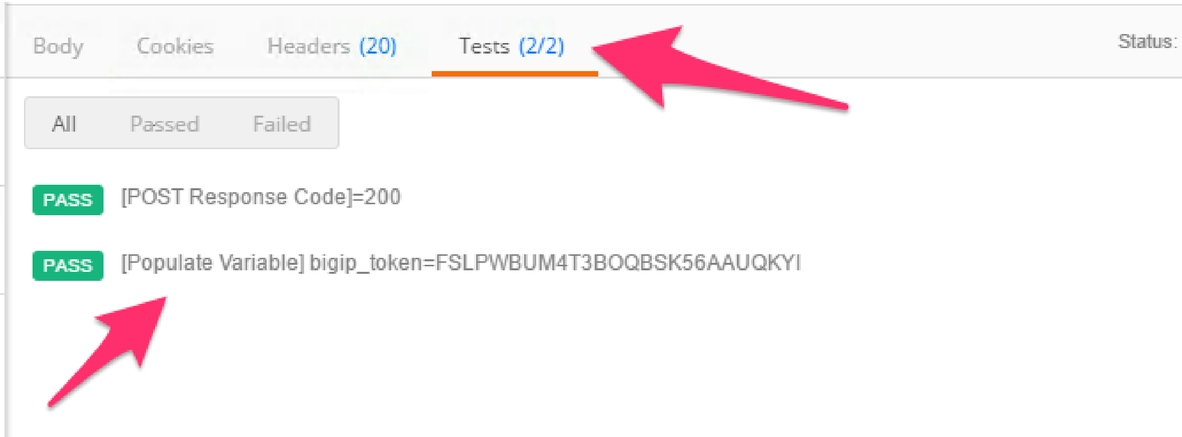
4. Select the `Lab 2.2` environment:



The preceding steps configured the Input Variables required for all the folders that comprise our workflow. We will now manually execute all the requests in the folders.

1. Expand the `BIGIP_API_Authentication` -> `1_Authenticate` folder.
2. Select the `Authenticate and Obtain Token` item and click `Send`
3. Examine the `Tests` in the response portion of the request. All the tests should be passing. Additionally you should see a test similar to:

   `[Populate Variable] bigip_token=....`

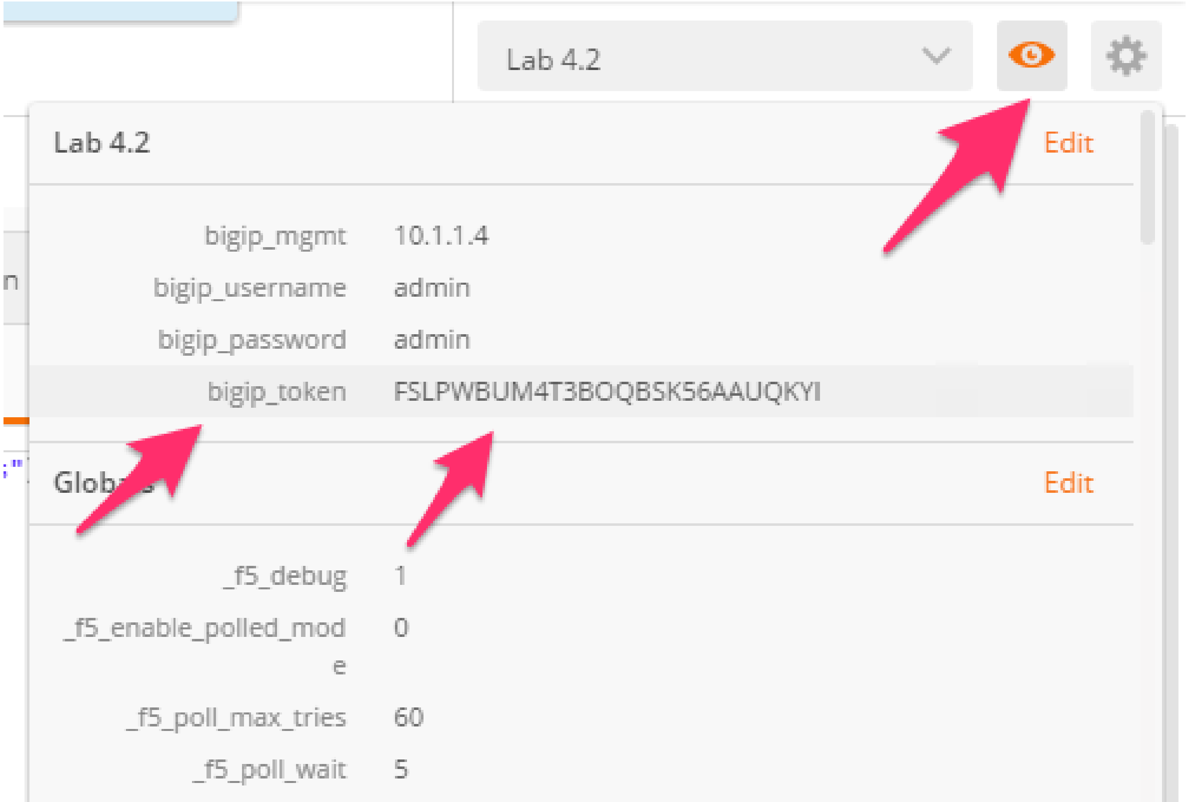
   

   PASS [POST Response Code]=200

   PASS [Populate Variable] bigip_token=FSLPWBUM4T3BOQBSK56AAUQKYI

   These test items and their corresponding actions (populating a variable in this case) are generated by the f5-postman-workflows framework.

4. Examine your Postman Environment variables and confirm that the `bigip_token` variable is present and populated.



Lab 4.2

bigip_mgmt	10.1.1.4
bigip_username	admin
bigip_password	admin
bigip_token	FSLPWBUM4T3BOQBSK56AAUQKYI

Globals

_f5_debug	1
_f5_enable_polled_mode	0
_f5_poll_max_tries	60
_f5_poll_wait	5

5. Select the `Verify Authentication Works` request in the folder and click ‘Send’. Examine the Tests and ensure they are all passing

6. Select the `Set Authentication Token Timeout` request, click *Send* and verify all Tests pass.

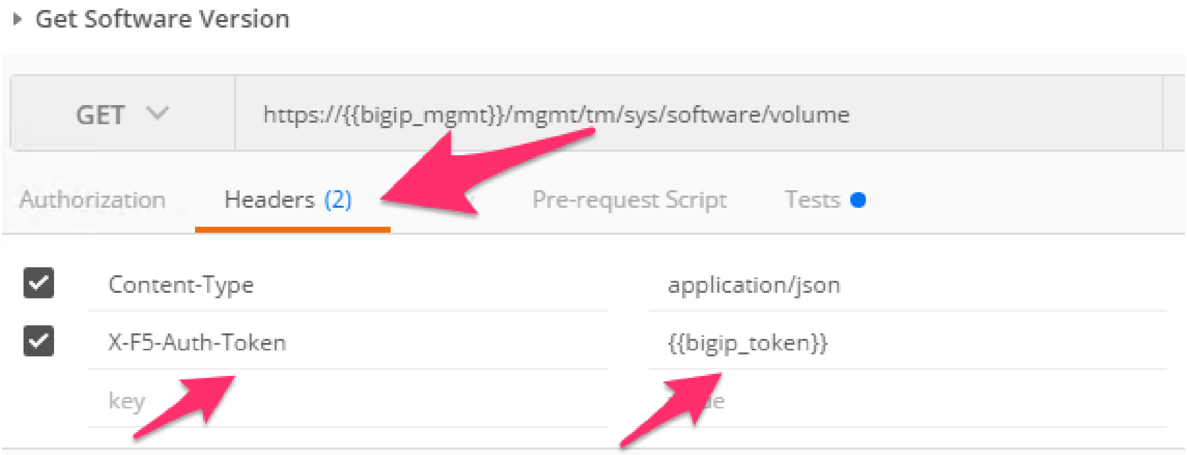
At this point we have successfully authenticated to our device and stored the authentication token in the `bigip_token` environment variable. We will now execute a request in a **different** collection and folder that uses the `bigip_token` variable value to authenticate and perform its actions.

1. Expand the `BIGIP_Operational_Workflows` -> `4A_Get_BIGIP_Version` folder.

2. Click the `Get Software Version` request.

3. Click the ‘Headers’ tab. Notice that the value for the `X-F5-Auth-Token` header is populated with the `bigip_token` variable value.

Note: Postman uses the `{{variable_name}}` syntax to perform variable value substitution.



GET https://{{bigip_mgmt}}/mgmt/tm/sys/software/volume

☑	Content-Type	application/json
☑	X-F5-Auth-Token	{{bigip_token}}
	key	value

4. Click 'Send' to send the request. Examine the Tests and ensure all tests have passed.
5. Examine your environment variables and note that the `bigip_version` and `bigip_build` variables are now populated.

While the example above was simple, it should show how we can chain together different collections and folders to assemble custom workflows. The key concepts to understand are:

- The f5-postman-workflows framework and collection test-code performs unit tests on the response data, and verifies the request executed successfully.
- The framework also populates Output Variables as documented so they can be used in subsequent requests as Inputs to assemble a workflow

Next, we will explore how to use this base knowledge to assemble various collections and folders into workflows using Newman and the f5-newman-wrapper.

3.2.3 Lab 2.3: f5-newman-wrapper Introduction

As shown in the previous lab, we can manually execute collections and folders using the Postman GUI to achieve end results on BIG-IP devices. While this capability is important in a test/prototyping phase, we need to ensure we can execute these manual steps as an automated process.

To achieve this goal we can use the f5-newman-wrapper tool. This tool allows a user to specify a workflow in a JSON formatted file, this includes Input Variables, the collections and folders, and executes various output options to provide feedback and run details in a programmatic fashion.

The core element of a workflow that can be executed by f5-newman-wrapper is a `JSON formatted` input file. In this lab we will introduce the file format.

Task 1 - Explore the workflow JSON format

To introduce the format of the workflow file we will use an example that recreates the simple workflow we executed manually in the previous lab. We will explore the file in sections followed by showing the whole file.

Define Name and Description

```
{
     "name":"Wrapper_Demo_1",
     "description":"Execute a chained workflow that authenticates to a BIG-IP and␣
↪retrieves it's software version"
}
```

Define Global Settings for the Run

This section defines how f5-newman-wrapper will run this workflow. The attributes are explained in the table below.

```
{
     "globalEnvVars":"../framework/f5-postman-workflows.postman_globals.json",
     "globalOptions": {
         "insecure":true,
         "reporters":["cli"]
     },
     "saveEnvVars":true,
     "outputFile":"Wrapper_Demo_1-run.json",
     "envOutputFile":"Wrapper_Demo_1-env.json"
 }
```

Attribute	Description
`globalEnvVars`	This is the file that contains the Global environment variables used by Newman. This file is generated by the f5-postman-workflows build scripts and contains the same global variables as we saw in the previous lab that installed the framework into the Postman GUI client
`globalOptions`	Specify the global options for newman. Available options are documented at: https://github.com/postmanlabs/newman#api-reference **Note:** Removing the `cli` option from the `reporters` array will disable verbose CLI output
`saveEnvVars`	Save the environment variables at the end of the run to a file
`outputFile`	The file to save the run details to.
`envOutputFile`	The file to save the environment variables at the end of the run to.

Define Input Variables

This section specifies the Input Variables for the workflow. The name `globalVars` conveys that the variables defined here will be present for each request in the defined workflow (the global scope from a workflow perspective). Variables can also be defined within each item in a workflow (the local scope from a item perspective). In the case of a global and local variable that is named identically, the local scope variable will take precedence.

```
 {
     "globalVars": {
         "bigip_mgmt": "10.1.1.4",
         "bigip_username":"admin",
         "bigip_password":"admin"
     }
 }
```

Define the Workflow Collections and Ordering

This section defines the workflow and collections and folders that it is comprised of. The `workflow` attribute is an ordered array that contains objects defining each collection and folder to run.

```
 {
    "workflow": [
        {
            "name":"Authenticate to BIG-IP",
            "options": {
                "collection":".. /collections/BIG_IP/BIGIP_API_Authentication.postman_
↪collection.json",
                "folder":"1_Authenticate"
            }
        },
        {
            "name":"Get BIG-IP Software Version",
            "options": {
                "collection":"../collections/BIG_IP/BIGIP_Operational_Workflows.
↪postman_collection.json",
                "folder":"4A_Get_BIGIP_Version"
            }
        }
    ]
}
```

Lets look at the item in the workflow that performs authentication:

```
                {
                        "name":"Authenticate to BIG-IP",
                        "options": {
                                "collection":".. /collections/BIG_IP/BIGIP_API_
↪Authentication.postman_collection.json",
                                "folder":"1_Authenticate"
                        }
                }
```

The `name` attribute specifies the name for this item. The `options` object specifies the parameters used to execute this particular item. In our case the `collection` attribute refers to the file containing the `BIGIP_API_Authentication` collection. The `folder` attribute specifies the name of the folder to run in the collection.

By default all output variables from a collection or folder are passed to the next item in the workflow. This allows us to chain collections together as needed to build workflows.

Final Workflow JSON

```
{
        "name":"Wrapper_Demo_1",
        "description":"Execute a chained workflow that authenticates to a BIG-IP and retrieves it's software version",
        "globalEnvVars":"../framework/f5-postman-workflows.postman_globals.json",
        "globalOptions": {
                "insecure":true,
                "reporters":["cli"]
        },
        "globalVars": {
                "bigip_mgmt": "10.1.1.4",
                "bigip_username":"admin",
                "bigip_password":"admin"
        },
        "saveEnvVars":true,
        "outputFile":"Wrapper_Demo_1-run.json",
        "envOutputFile":"Wrapper_Demo_1-env.json",
        "workflow": [
                {
                        "name":"Authenticate to BIG-IP",
                        "options": {
                                "collection":"..   /collections/BIG_IP/BIGIP_API_Authentication.   postman_collection.json",
                                "folder":"1_Authenticate"
                        }
                },
                {
                        "name":"Get BIG-IP Software Version",
                        "skip":false,
                        "options": {
                                "collection":"..   /collections/BIG_IP/BIGIP_Operational_Workflows.   postman_collection.json",
                                "folder":"4A_Get_BIGIP_Version"
                        }
                }
        ]
}
```

3.2.4 Lab 2.4: Run a workflow with f5-newman-wrapper

In this lab we will use the f5-super-netops-container to run the workflow we reviewed in the previous lab. The advantage of using the f5-super-netops Container is that all the tools, collections and frameworks are pre-installed and ready to use.

Task 1 - Run a f5-newman-wrapper Workflow

1. Return to, or open an SSH session as described in the *previous lab*
2. Run `cd f5-postman-workflows/local`
3. Run `cp ../workflows/Wrapper_Demo_1.json .`
4. Edit the `Wrapper_Demo_1.json` file with `vim` and enter the `10.1.1.4` for the value of the `bigip_mgmt` variable

```
"globalVars": {
        "bigip_mgmt": "10.1.1.4",
        "bigip_username":"admin",
        "bigip_password":"admin"
},
```

5. Run `f5-newman-wrapper Wrapper_Demo_1.json`
6. Examine the output to see how the workflow was executed. Notice that the same tests that we saw when using Postman are present during this run.

   Example output:

```
[snops@f5-super-netops] [~/f5-postman-workflows/local] $ f5-newman-wrapper␣
↪Wrapper_Demo_1.json
[Wrapper_Demo_1-2017-03-30-04-08-12] starting run
[Wrapper_Demo_1-2017-03-30-04-08-12] [runCollection][Authenticate to BIG-IP]␣
↪running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 108ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=WYKIVPHCNASNVEC55ZDVNH5OO2

? Verify Authentication Works
  GET https://10.1.1.4/mgmt/shared/authz/tokens/WYKIVPHCNASNVEC55ZDVNH5OO2 [200␣
↪OK, 1.23KB, 8ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=WYKIVPHCNASNVEC55ZDVNH5OO2
  ✓  [Check Value] token == WYKIVPHCNASNVEC55ZDVNH5OO2

? Set Authentication Token Timeout
  PATCH https://10.1.1.4/mgmt/shared/authz/tokens/WYKIVPHCNASNVEC55ZDVNH5OO2 [200␣
↪OK, 1.23KB, 14ms]
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 297ms                     |
?-------------------------------?
| total data received: 1.71KB (approx)          |
?-------------------------------?
| average response time: 43ms                   |
?-------------------------------?
[Wrapper_Demo_1-2017-03-30-04-08-12] [runCollection][Get BIG-IP Software Version]␣
↪running...
```

```
newman

BIGIP_Operational_Workflows

? 4A_Get_BIGIP_Version
? Get Software Version
  GET https://10.1.1.4/mgmt/tm/sys/software/volume [200 OK, 1.32KB, 16ms]
  ✓  [GET Response Code]=200
  ✓  [Populate Variable] bigip_version=12.1.1
  ✓  [Populate Variable] bigip_build=1.0.196
[Wrapper_Demo_1-2017-03-30-04-08-12] run completed

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        1 |        0 |
?-----------------?-------?-------?
|            test-scripts |        1 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        0 |        0 |
?-----------------?-------?-------?
|              assertions |        3 |        0 |
?-----------------?-------?-------?
| total run duration: 58ms                      |
?-------------------------------?
| total data received: 611B (approx)            |
?-------------------------------?
| average response time: 16ms                   |
?-------------------------------?
```

7. Examine the environment variables that were saved at the end of the run by executing `cat Wrapper_Demo_1-env.json`

   Example output:

```
{
  "id": "c0550892-36d4-4412-bf35-a1d9aa8d2efe",
  "values": [
    {
      "type": "any",
      "value": "10.1.1.4",
      "key": "bigip_mgmt"
    },
    {
      "type": "any",
      "value": "admin",
      "key": "bigip_username"
    },
    {
      "type": "any",
      "value": "admin",
      "key": "bigip_password"
    },
    {
      "type": "any",
      "value": "WYKIVPHCNASNVEC55ZDVNH5OO2",
      "key": "bigip_token"
    },
    {
      "type": "any",
      "value": "1200",
      "key": "bigip_token_timeout"
    },
    {
      "type": "any",
      "value": "12.1.1",
      "key": "bigip_version"
    },
    {
      "type": "any",
      "value": "1.0.196",
      "key": "bigip_build"
    }
  ]
}
```

Notice that the `bigip_version` and `bigip_build` variables were saved, similar to how this was shown in the Postman GUI Environment Variables. This file is JSON formatted and can easily be used directly by other tools to drive further automation.

3.2.5 Lab 2.5: Building Complex Workflows

In the previous lab we reviewed and ran a very simple workflow. To support more complex use cases f5-newman-wrapper includes features to help build more complex workflows.

These features allow users to:

- Create infinitely nested items
- Rename/remap variables name pre and post run of an item
- Load variables from a saved environment file
- Define variables in the global (workflow) or local (item) scope

To explore all the available options currently implemented please refer to https://raw.githubusercontent.com/0xHiteshPatel/f5-postman-workflows/master/framework/f5-newman-wrapper/workflow-schema.json

Task 1 - Explore Nested Workflows & Variable Remapping

By using the 'children' array within an item in a workflow you can create nested items. In this task, we will create a more advanced version of the workflow we used in the previous lab. This workflow will perform authentication to two BIG-IP devices and then retrieve the software version running on each.

We will implement a workflow that is best depicted by the following branch diagram:

```
Start
  |
  |- Authenticate
  |  |- Authenticate to BIG-IP A
  |  |- Authenticate to BIG-IP B
  |
  |- Get BIGIP Version
  |  |- Get BIGIP Version on BIG-IP A
  |  |- Get BIGIP Version on BIG-IP B
  |
Stop
```

To implement this workflow we need to consider how Input Variables are passed to each item in the workflow. Previously, we saw that the following variables are required to the the `1_Authenticate` folder in the `BIGIP_API_Authentication` collection:

- `bigip_mgmt`
- `bigip_username`
- `bigip_password`

The issue we encounter when building this workflow is that we, at a minimum, have different values for `bigip_mgmt` because we are trying to communicate with two BIG-IP devices. To address this issue, we could define our input variables as follows:

- `bigip_a_mgmt = 10.1.1.4`
- `bigip_b_mgmt = 10.1.1.5`
- `bigip_username = admin`
- `bigip_password = admin`

This solves the problem of providing both BIG-IP management addresses, however, it introduces another issue. The `1_Authenticate` folder requires that the management IP address be passed in the `bigip_mgmt` input variable. To solve this issue, we will use variable name remapping to remap a globalVar to a different name before the `1_Authenticate` folder is run for each BIG-IP device. To illustrate this, we will add more information to our diagram:

```
Start
  |
  |- Define globalVars
  |  |- bigip_a_mgmt = 10.1.1.4
  |  |- bigip_b_mgmt = 10.1.1.5
  |  |- bigip_username = admin
  |  |- bigip_password = admin
  |
  |- Authenticate
  |  |- Authenticate to BIG-IP A
  |  |  | Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
  |  |  |     Run: 1_Authenticate folder
  |  |
  |  |- Authenticate to BIG-IP B
  |  |  | Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
  |  |  |     Run: 1_Authenticate folder
  |
  |- Get BIGIP Version
  |  |- Get BIGIP Version on BIG-IP A
  |  |- Get BIGIP Version on BIG-IP B
  |
Stop
```

We've now addressed our issues regarding defining and passing the BIG-IP management address, but have to consider one last problem. The **output variable** of the `1_Authenticate` folder is `bigip_token`. By default f5-newman-wrapper will store all output variables from one folder and automatically pass them to the next item. In this case, an issue occurs because the `Authenticate to BIG-IP B` item will overwrite the bigip_token variable that was outputted by the Authenticate to BIG-IP A item. To resolve this issue, we can remap variables **AFTER** or post-run of an item. We can modify our diagram to handle this issue like this:

```
Start
  |
  |- Define globalVars
  |  |- bigip_a_mgmt = 10.1.1.4
  |  |- bigip_b_mgmt = 10.1.1.5
  |  |- bigip_username = admin
  |  |- bigip_password = admin
  |
  |- Authenticate
  |  |- Authenticate to BIG-IP A
  |  |  |  Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
  |  |  |      Run: 1_Authenticate folder
  |  |  | Post-run: Remap bigip_token -> bigip_a_token
  |  |
  |  |- Authenticate to BIG-IP B
  |  |  |  Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
  |  |  |      Run: 1_Authenticate folder
  |  |  | Post-run: Remap bigip_token -> bigip_b_token
  |
  |- Get BIGIP Version
  |  |- Get BIGIP Version on BIG-IP A
  |  |- Get BIGIP Version on BIG-IP B
  |
Stop
```

The last step is to perform some additional pre-run remapping to pass the correct token to the 4A_Get_BIGIP_Version folder to get our BIG-IP software version. Additionally, we will perform some post-run remaps so we can save the output variables for each device:

```
Start
  |
  |- Define globalVars
  |  |- bigip_a_mgmt = 10.1.1.4
  |  |- bigip_b_mgmt = 10.1.1.5
  |  |- bigip_username = admin
  |  |- bigip_password = admin
  |
  |- Authenticate
  |  |- Authenticate to BIG-IP A
  |  |  |  Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
  |  |  |      Run: 1_Authenticate folder
  |  |  | Post-run: Remap bigip_token -> bigip_a_token
  |  |
  |  |- Authenticate to BIG-IP B
  |  |  |  Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
  |  |  |      Run: 1_Authenticate folder
  |  |  | Post-run: Remap bigip_token -> bigip_b_token
  |
  |- Get BIGIP Version
  |  |- Get BIGIP Version on BIG-IP A
  |  |  |  Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
  |  |  |  Pre-run: Remap bigip_a_token -> bigip_token
  |  |  |      Run: 4A_Get_BIGIP_Version folder
  |  |  | Post-run: Remap bigip_version -> bigip_a_version
  |   |   | Post-run: Remap bigip_build -> bigip_a_build
  |   |
  |   |- Get BIGIP Version on BIG-IP B
  |   |   |  Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
  |       |  Pre-run: Remap bigip_b_token -> bigip_token
  |       |      Run: 4A_Get_BIGIP_Version folder
  |       | Post-run: Remap bigip_version -> bigip_b_version
  |       | Post-run: Remap bigip_build -> bigip_b_build
  |
  |- Save globarVars to file
  |
Stop
```

Note: Collections and folders that are designed to act on multiple devices are designed to automatically use the `bigip_a_...` and `bigip_b_...` syntax to avoid having to remap variables. In this case the `BIGIP_Operational_Workflows` collection is designed to perform actions on **one** device at a time, thus the need for remapping of the `bigip_token` input variables.

Note: Another option that is available to solve this issue is to define all variables in the local scope for each item. This method is not preferred because it decreases portability and increases complexity in definition of input variables.

Task 2 - Build Complex Workflow JSON

Define Global Settings & Variables:

```
{
  "name":"Wrapper_Demo_2",
  "description":"Execute a chained workflow that authenticates to two BIG-IPs and retrieves their software version",
  "globalEnvVars":"../framework/f5-postman-workflows.postman_globals.json",
  "globalOptions": {
    "insecure":true,
    "reporters":["cli"]
  },
  "globalVars": {
    "bigip_a_mgmt": "10.1.1.4",
    "bigip_b_mgmt": "10.1.1.5",
    "bigip_username":"admin",
    "bigip_password":"admin"
  },
  "saveEnvVars":true,
  "outputFile":"Wrapper_Demo_2-run.json",
  "envOutputFile":"Wrapper_Demo_2-env.json"
}
```

Define Authentication Items

Note: As shown below, we can use the `skip: true` attribute to signal f5-newman-wrapper to not run that particular item. The items `children` will still be processed. The `skip` attribute can be used to create a container for similar requests.

```
{
  "workflow": [
    {
      "name":"Authenticate to BIG-IPs",
      "skip":true,
      "children": [
        {
          "name":"Authenticate to BIG-IP A",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_API_Authentication.postman_
↪collection.json",
            "remapPreRun": {
              "bigip_a_mgmt": "bigip_mgmt"
            },
            "folder":"1_Authenticate",
            "remapPostRun": {
              "bigip_token": "bigip_a_token"
            }
          }
        },
        {
          "name":"Authenticate to BIG-IP B",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_API_Authentication.postman_
↪collection.json",
            "remapPreRun": {
              "bigip_b_mgmt": "bigip_mgmt"
            },
            "folder":"1_Authenticate",
            "remapPostRun": {
              "bigip_token": "bigip_b_token"
            }
          }
        }
      ]
    }
  ]
}
```

The JSON above implements the following part of our branch diagram:

```
|- Authenticate
   |- Authenticate to BIG-IP A
   |  |  Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
   |  |      Run: 1_Authenticate folder
   |  | Post-run: Remap bigip_token -> bigip_a_token
   |
   |- Run: Authenticate to BIG-IP B
   |  |  Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
   |  |      Run: 1_Authenticate folder
   |  | Post-run: Remap bigip_token -> bigip_b_token
```

Specifically, note the use of the `skip` attribute on line 5 to create a container to group the items together.

Define Get Software Version Items

```
{
   "workflow": [
     {
       "name":"Get BIG-IP Software Versions",
       "skip":true,
       "children": [
         {
           "name":"Get BIG-IP A Software Version",
           "options": {
             "collection":"../collections/BIG_IP/BIGIP_Operational_Workflows.postman_
↪collection.json",
             "remapPreRun": {
               "bigip_a_mgmt": "bigip_mgmt",
               "bigip_a_token": "bigip_token"
             },
             "folder":"4A_Get_BIGIP_Version",
             "remapPostRun": {
               "bigip_version": "bigip_a_version",
               "bigip_build": "bigip_a_build"
             }
           }
         },
         {
           "name":"Get BIG-IP B Software Version",
           "options": {
             "collection":"../collections/BIG_IP/BIGIP_Operational_Workflows.postman_
↪collection.json",
             "remapPreRun": {
               "bigip_b_mgmt": "bigip_mgmt",
               "bigip_b_token": "bigip_token"
             },
             "folder":"4A_Get_BIGIP_Version",
             "remapPostRun": {
               "bigip_version": "bigip_b_version",
               "bigip_build": "bigip_b_build"
             }
           }
         }
       ]
     }
   ]
}
```

The JSON above implements the following part of our branch diagram:

```
|- Get BIGIP Version
   |- Get BIGIP Version on BIG-IP A
   |   |  Pre-run: Remap bigip_a_mgmt -> bigip_mgmt
   |   |  Pre-run: Remap bigip_a_token -> bigip_token
   |   |      Run: 4A_Get_BIGIP_Version folder
   |   | Post-run: Remap bigip_version -> bigip_a_version
   |   | Post-run: Remap bigip_build -> bigip_a_build
   |
   |- Get BIGIP Version on BIG-IP B
   |   |  Pre-run: Remap bigip_b_mgmt -> bigip_mgmt
      |  Pre-run: Remap bigip_b_token -> bigip_token
      |      Run: 4A_Get_BIGIP_Version folder
      | Post-run: Remap bigip_version -> bigip_b_version
      | Post-run: Remap bigip_build -> bigip_b_build
```

Final Workflow JSON

```
{
  "name":"Wrapper_Demo_2",
  "description":"Execute a chained workflow that authenticates to two BIG-IPs and retrieves their software version",
  "globalEnvVars":"../framework/f5-postman-workflows.postman_globals.json",
  "globalOptions": {
    "insecure":true,
    "reporters":["cli"]
  },
  "globalVars": {
    "bigip_a_mgmt": "",
    "bigip_b_mgmt": "",
    "bigip_username":"admin",
    "bigip_password":"admin"
  },
  "saveEnvVars":true,
  "outputFile":"Wrapper_Demo_2-run.json",
  "envOutputFile":"Wrapper_Demo_2-env.json",
  "workflow": [
    {
      "name":"Authenticate to BIG-IPs",
      "skip":true,
      "children": [
        {
          "name":"Authenticate to BIG-IP A",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_API_Authentication.postman_collection.json",
            "remapPreRun": {
              "bigip_a_mgmt": "bigip_mgmt"
            },
            "folder":"1_Authenticate",
            "remapPostRun": {
              "bigip_token": "bigip_a_token"
            }
          }
        },
        {
          "name":"Authenticate to BIG-IP B",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_API_Authentication.postman_collection.json",
            "remapPreRun": {
              "bigip_b_mgmt": "bigip_mgmt"
            },
            "folder":"1_Authenticate",
            "remapPostRun": {
              "bigip_token": "bigip_b_token"
            }
          }
        }
      ]
    },
    {
      "name":"Get BIG-IP Software Versions",
      "skip":true,
      "children": [
        {
          "name":"Get BIG-IP A Software Version",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_Operational_Workflows.postman_collection.json",
            "remapPreRun": {
              "bigip_a_mgmt": "bigip_mgmt",
              "bigip_a_token": "bigip_token"
            },
            "folder":"4A_Get_BIGIP_Version",
            "remapPostRun": {
              "bigip_version": "bigip_a_version",
              "bigip_build": "bigip_a_build"
            }
          }
        },
        {
          "name":"Get BIG-IP B Software Version",
          "options": {
            "collection":"../collections/BIG_IP/BIGIP_Operational_Workflows.postman_collection.json",
            "remapPreRun": {
              "bigip_b_mgmt": "bigip_mgmt",
              "bigip_b_token": "bigip_token"
            },
            "folder":"4A_Get_BIGIP_Version",
            "remapPostRun": {
              "bigip_version": "bigip_b_version",
              "bigip_build": "bigip_b_build"
            }
          }
        }
      ]
    }
  ]
}
```

Task 3 - Run the Workflow

1. Open an SSH session as described in the *previous lab*
2. Run `cd f5-postman-workflows/local`
3. Run `cp ../workflows/Wrapper_Demo_2.json .`
4. Edit the `Wrapper_Demo_2.json` file and enter you BIG-IP management addresses

```
{
  "globalVars": {
          "bigip_a_mgmt": "10.1.1.4",
          "bigip_b_mgmt": "10.1.1.5",
          "bigip_username":"admin",
          "bigip_password":"admin"
  }
}
```

5. Run `f5-newman-wrapper Wrapper_Demo_2.json`
6. Examine the output to see how the workflow was executed.

   Example output:

```
[snops@f5-super-netops] [~/f5-postman-workflows/local] $ f5-newman-wrapper␣
↪Wrapper_Demo_2.json
[Wrapper_Demo_2-2017-03-30-19-22-52] starting run
[Wrapper_Demo_2-2017-03-30-19-22-52] [runCollection][Authenticate to BIG-IP A]␣
↪running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 570ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=UE7W5CXWM5SJ6SZEV5A7KTAI5Q

? Verify Authentication Works
  GET https://10.1.1.4/mgmt/shared/authz/tokens/UE7W5CXWM5SJ6SZEV5A7KTAI5Q [200␣
↪OK, 1.23KB, 9ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=UE7W5CXWM5SJ6SZEV5A7KTAI5Q
  ✓  [Check Value] token == UE7W5CXWM5SJ6SZEV5A7KTAI5Q

? Set Authentication Token Timeout
  PATCH https://10.1.1.4/mgmt/shared/authz/tokens/UE7W5CXWM5SJ6SZEV5A7KTAI5Q [200␣
↪OK, 1.23KB, 13ms]
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 740ms                     |
?-------------------------------?
```

```
| total data received: 1.71KB (approx)          |
?-------------------------------?
| average response time: 197ms                  |
?-------------------------------?
[Wrapper_Demo_2-2017-03-30-19-22-52] [runCollection][Authenticate to BIG-IP B]␣
↪running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.5/mgmt/shared/authn/login [200 OK, 1.41KB, 350ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=ONQXOQPNCVOHZELKIFSPHETL3I

? Verify Authentication Works
  GET https://10.1.1.5/mgmt/shared/authz/tokens/ONQXOQPNCVOHZELKIFSPHETL3I [200␣
↪OK, 1.23KB, 9ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=ONQXOQPNCVOHZELKIFSPHETL3I
  ✓  [Check Value] token == ONQXOQPNCVOHZELKIFSPHETL3I

? Set Authentication Token Timeout
  PATCH https://10.1.1.5/mgmt/shared/authz/tokens/ONQXOQPNCVOHZELKIFSPHETL3I [200␣
↪OK, 1.23KB, 12ms]
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 472ms                     |
?-------------------------------?
| total data received: 1.71KB (approx)          |
?-------------------------------?
| average response time: 123ms                  |
?-------------------------------?
[Wrapper_Demo_2-2017-03-30-19-22-52] [runCollection][Get BIG-IP A Software␣
↪Version] running...
newman

BIGIP_Operational_Workflows

? 4A_Get_BIGIP_Version
? Get Software Version
  GET https://10.1.1.4/mgmt/tm/sys/software/volume [200 OK, 1.32KB, 207ms]
  ✓  [GET Response Code]=200
  ✓  [Populate Variable] bigip_version=12.1.1
  ✓  [Populate Variable] bigip_build=1.0.196

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        1 |        0 |
?-----------------?-------?-------?
|            test-scripts |        1 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        0 |        0 |
?-----------------?-------?-------?
|              assertions |        3 |        0 |
?-----------------?-------?-------?
| total run duration: 250ms                     |
?-------------------------------?
| total data received: 611B (approx)            |
?-------------------------------?
| average response time: 207ms                  |
?-------------------------------?
[Wrapper_Demo_2-2017-03-30-19-22-52] [runCollection][Get BIG-IP B Software␣
↪Version] running...
newman

BIGIP_Operational_Workflows

? 4A_Get_BIGIP_Version
? Get Software Version
  GET https://10.1.1.5/mgmt/tm/sys/software/volume [200 OK, 1.32KB, 191ms]
  ✓  [GET Response Code]=200
  ✓  [Populate Variable] bigip_version=12.1.1
  ✓  [Populate Variable] bigip_build=1.0.196

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        1 |        0 |
?-----------------?-------?-------?
|            test-scripts |        1 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        0 |        0 |
?-----------------?-------?-------?
|              assertions |        3 |        0 |
?-----------------?-------?-------?
| total run duration: 230ms                     |
?-------------------------------?
| total data received: 611B (approx)            |
?-------------------------------?
| average response time: 191ms                  |
?-------------------------------?
[Wrapper_Demo_2-2017-03-30-19-22-52] run completed in 3s, 316.921 ms
```

7. Examine the environment variables that were saved at the end of the run by executing `cat`

Wrapper_Demo_2-env.json. The resulting BIG-IP software versions are now present and have been highlighted below.

Example output:

```
{
  "id": "d459e491-4936-4be7-a910-567f711a636a",
  "values": [
    {
      "type": "any",
      "value": "10.1.1.4",
      "key": "bigip_a_mgmt"
    },
    {
      "type": "any",
      "value": "10.1.1.5",
      "key": "bigip_b_mgmt"
    },
    {
      "type": "any",
      "value": "10.1.1.5",
      "key": "bigip_mgmt"
    },
    {
      "type": "any",
      "value": "admin",
      "key": "bigip_username"
    },
    {
      "type": "any",
      "value": "admin",
      "key": "bigip_password"
    },
    {
      "type": "any",
      "value": "UE7W5CXWM5SJ6SZEV5A7KTAI5Q",
      "key": "bigip_a_token"
    },
    {
      "type": "any",
      "value": "ONQXOQPNCVOHZELKIFSPHETL3I",
      "key": "bigip_b_token"
    },
    {
      "type": "any",
      "value": "ONQXOQPNCVOHZELKIFSPHETL3I",
      "key": "bigip_token"
    },
    {
      "type": "any",
      "value": "12.1.1",
      "key": "bigip_a_version"
    },
    {
      "type": "any",
      "value": "1.0.196",
      "key": "bigip_a_build"
    },
    {
      "type": "any",
      "value": "1200",
      "key": "bigip_token_timeout"
    },
    {
      "type": "any",
      "value": "12.1.1",
      "key": "bigip_b_version"
    },
    {
      "type": "any",
      "value": "1.0.196",
      "key": "bigip_b_build"
    }
  ]
}
```

3.3 Module 3: Stitching Workflows from Class 1 into new Orchestratable Collections

In the previous module we saw the example of stitching together the Authentication Folder and some facts gathering. We will now stitch together the Postman Collection from Class 1 and the Authentication Collection from Module 2. Once we validate the new file we'll use f5-newman-wrapper to execute.

In review, to assist users with automating the F5 BIG-IP platform we have developed a Collection of calls that can be used with the Postman REST Client (http://getpostman.com). The purpose of the tools are:

- f5-postman-workflows
  - Provide re-usable JavaScript functions that ease testing of API responses and populating environment variables
  - Implement a delay-based polling mechanism
- F5_Automation_Orchestration_Intro (Class 1)
  - F5's training collection for **Onboarding** BIG-IP
  - F5's training collection for **Operating** BIG-IP

Stitching together the collections and workflows allows Super-NetOps engineers the ability to start quickly Orchestrating calls running Automation workflows. This also allows BIG-IP to be Orchestrated from upper level orchestration toolkits.

Using this structure allows you to build your own solutions, to manage BIG-IP quickly as native REST calls are used.

From the previous labs you should already have your Super-NetOps-Container running, if it's not please refer to Class 2 Module 2 on starting your service.

3.3.1 Lab 3.1 - Files used and locations

The `f5-super-netops-container` is a self contained toolkit, meaning everything we will be using is already in the solution. It will also always be updated (CD), received either from restarting the container or updating via git clones scripts.

Note: If you are running in the provided training infrastructure, the `f5-super-netops-container` can be accessed via Putty, all commands from this point will be run from within the container

Task 1 - Review the super-netops-container files and collections used

1. Return to or open a new session to the `super-netops-container` user credentials are `snops` and `default`
2. During the installation of the `super-netops-container` there were several github repositories cloned, all of which are mapped to the `/home/snops/` directory.

Let's make sure all repositories were mapped correctly.

Execute: `cd /home/snops`

The Collections we will be using are located here:

```
|- /f5-automation-labs
   |- /postman_collections
   |   |  f5-programmability-class-2.postman_collection.json
```

```
|- /f5-postman-workflows
  |- /collections
  |   |  /BIG_IP
  |   |      BIGIP_Operational_Workflows.postman_collection.json
```

The f5-newman-wrapper configuration files are located here:

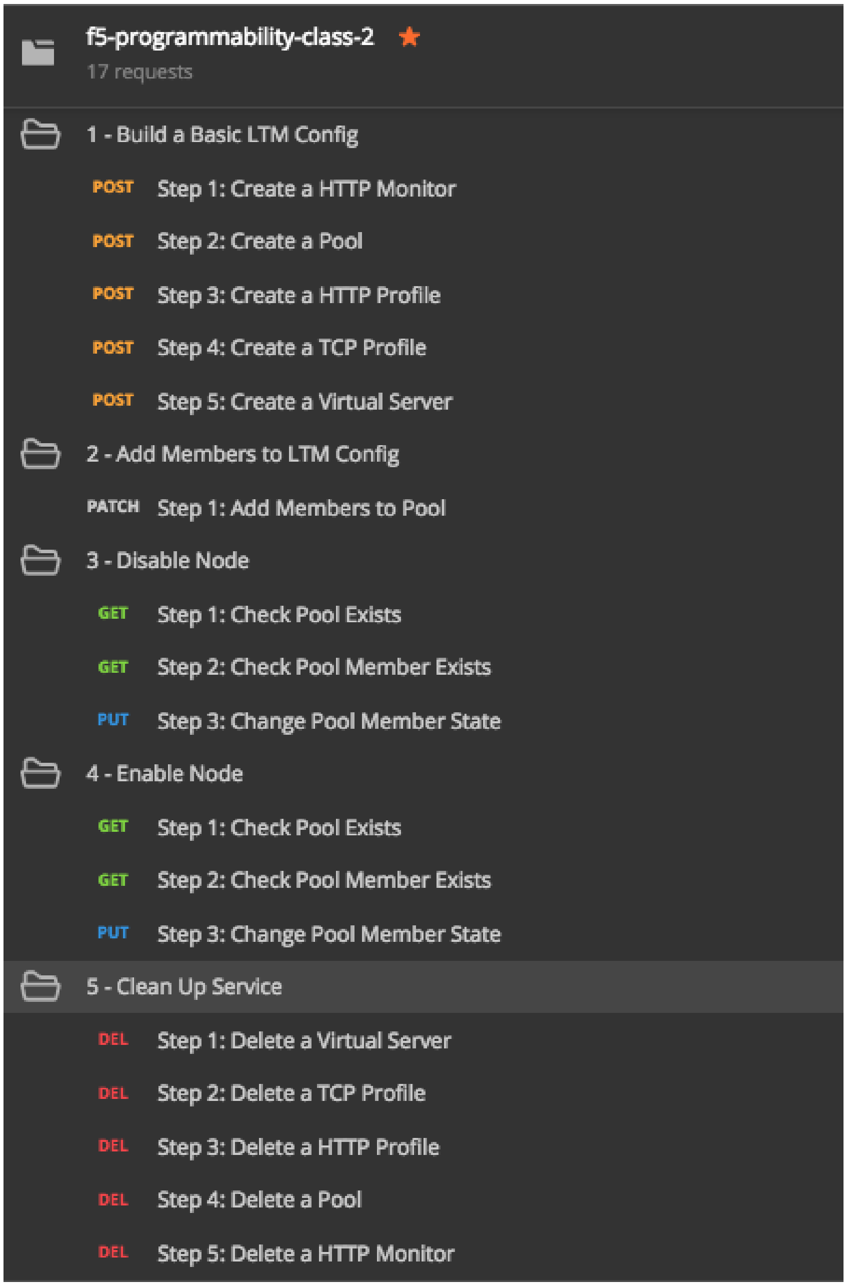
```
|- /f5-automation-labs
   |- /jenkins
   |   |  /f5-newman-build
   |   |      f5-newman-build-1
   |   |      f5-newman-build-2
   |   |      f5-newman-build-5
   |   |  /f5-newman-operation
   |   |      f5-newman-build-3
   |   |      f5-newman-build-4
```

3.3.2 Lab 3.2 - Execute f5-newman-wrapper for a Build Workflow

Your environment has already been seeded with 5 `f5-newman-wrapper` files, these files will execute against the collections noted in the previous lab. This lab will cover the **Build** aspect, creating a Virtual Server Framework containing all the pieces required for this demo service.

Note: This is a Postman Collection, and can also be imported into the Postman GUI client for viewing

For a visual reference of what f5-programmability-class-2.postman_collection.json looks like:



f5-programmability-class-2 ★
17 requests

1 - Build a Basic LTM Config
- POST Step 1: Create a HTTP Monitor
- POST Step 2: Create a Pool
- POST Step 3: Create a HTTP Profile
- POST Step 4: Create a TCP Profile
- POST Step 5: Create a Virtual Server

2 - Add Members to LTM Config
- PATCH Step 1: Add Members to Pool

3 - Disable Node
- GET Step 1: Check Pool Exists
- GET Step 2: Check Pool Member Exists
- PUT Step 3: Change Pool Member State

4 - Enable Node
- GET Step 1: Check Pool Exists
- GET Step 2: Check Pool Member Exists
- PUT Step 3: Change Pool Member State

5 - Clean Up Service
- DEL Step 1: Delete a Virtual Server
- DEL Step 2: Delete a TCP Profile
- DEL Step 3: Delete a HTTP Profile
- DEL Step 4: Delete a Pool
- DEL Step 5: Delete a HTTP Monitor

Note: You do not need to have all these operations individually broken out, it is shown this way to educate that Workflows can be as small (update a pool member) or as large (deploy a whole service) as needed

Task 1 - Examine f5-newman-build-1

Note: The contents of this folder contain files for this lab and upcoming labs in this class

1. Return to or open a new session to the `super-netops-container` user credentials are `snops` and `default`

2. Navigate to the location containing the f5-newman-wrapper files `cd ~/f5-automation-labs/jenkins/f5-newman-build`

3. Let's examine the contents of the first f5-newman-wrapper file `cat f5-newman-build-1`

```
{
       "name":"f5-newman-build-1",
       "description":"Execute a chained workflow that authenticates to a BIG-IP and builds configuration",
       "globalEnvVars":"/home/snops/f5-postman-workflows/framework/f5-postman-workflows.postman_globals.json",
       "globalOptions": {
               "insecure":true,
               "reporters":["cli"]
       },
       "globalVars": {
               "bigip_mgmt": "10.1.1.4",
               "bigip_username":"admin",
               "bigip_password":"admin",
               "bigip_partition":"Common",
               "bigip_pool_name":"module_3_pool",
               "bigip_pool_member":"75.67.228.133:80",
               "bigip_object_state":"user-up",
               "bigip_object_session":"user-enabled",
               "bigip_vs_name":"module_3_vs",
               "bigip_vs_destination":"10.1.20.129:80",
               "bigip_node_name":"75.67.228.133",
               "bigip_http_monitor":"module_3_http_monitor",
               "bigip_http_profile":"module_3_http",
               "bigip_tcp_profile":"module_3_tcp_clientside"
       },
       "workflow": [
               {
                       "name":"Authenticate to BIG-IP",
                       "options": {
                               "collection":"/home/snops/f5-postman-workflows/collections/BIG_IP/BIGIP_API_Authentication.postman_collection.json",
                               "folder":"1_Authenticate"
                       }
               },  (REMOVE THIS TEXT AND ADD YOUR CODE BELOW)

           }
       ]
 }
```

#. The above f5-newman-wrapper file only has the `Authenticate to BIG-IP` Collection/Folder referenced, **we will need to add in another collection**. You are going to add this code snippet after the last `},`. This shows the method for chaining together multiple calls from multiple sources, shown in a previous lab. For editing files VIM/VI is installed on the container, if you **do not know** how to use VIM/VI please let the instructor know.

```
{
     "name":"1 - Build a Basic LTM Config",
     "skip":false,
     "options": {
             "collection":"/home/snops/f5-automation-labs/postman_collections/f5-programmability-class-2.postman_collection.json",
             "folder":"1 - Build a Basic LTM Config"
   }
```

1. Now that you have the full file you can see what it will look like with `cat f5-newman-build-1`. The environment variables will float into both Collections, and the returned Global Variables will persist during the whole run.

Example of a complete file:

```
{
       "name":"f5-newman-build-1",
       "description":"Execute a chained workflow that authenticates to a BIG-IP and builds configuration",
       "globalEnvVars":"/home/snops/f5-postman-workflows/framework/f5-postman-workflows.postman_globals.json",
       "globalOptions": {
               "insecure":true,
               "reporters":["cli"]
       },
       "globalVars": {
               "bigip_mgmt": "10.1.1.4",
               "bigip_username":"admin",
               "bigip_password":"admin",
               "bigip_partition":"Common",
               "bigip_pool_name":"module_3_pool",
               "bigip_pool_member":"75.67.228.133:80",
               "bigip_object_state":"user-up",
               "bigip_object_session":"user-enabled",
               "bigip_vs_name":"module_3_vs",
               "bigip_vs_destination":"10.1.20.129:80",
               "bigip_node_name":"75.67.228.133",
               "bigip_http_monitor":"module_3_http_monitor",
               "bigip_http_profile":"module_3_http",
               "bigip_tcp_profile":"module_3_tcp_clientside"
       },
       "workflow": [
               {
                       "name":"Authenticate to BIG-IP",
                       "options": {
                               "collection":"/home/snops/f5-postman-workflows/collections/BIG_IP/BIGIP_API_Authentication.postman_collection.json",
                               "folder":"1_Authenticate"
                       }
               },
               {
                    "name":"1 - Build a Basic LTM Config",
                    "skip":false,
                    "options": {
                            "folder":"1 - Build a Basic LTM Config"
                            "collection":"/home/snops/f5-automation-labs/postman_collections/f5-programmability-class-2.postman_collection.json",
                  }
           }
       ]
 }
```

Task 2 - Execute the first f5-newman-wrapper file

1. Login to your BIG-IP lab machine and verify you do not have any Virtual Servers or Pools

Note: If you are using the F5 lab systems there are already shortcuts in your Chrome browser called `BIG-IP A GUI`, if you receive a certificate warning accept and add exception (the BIG-IP has a self-signed cert which violates Chrome's security). BIG-IP A Login credentials are `admin\admin`

2. `f5-newman-build-1` now contains the needed calls to build the Framework of an Application Service (Virtual Server, Pool and needed Profiles), **it doesn't however include any pool members**.

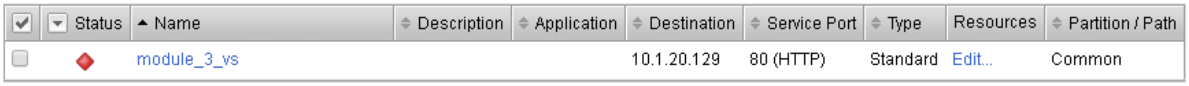
   Execute: `f5-newman-wrapper f5-newman-build-1`

   Output should look like:

```
$ f5-newman-wrapper f5-newman-build-1
[f5-newman-build-1-2017-07-26-08-23-00] starting run
[f5-newman-build-1-2017-07-26-08-23-00] [runCollection][Authenticate to BIG-IP] running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 505ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=MB4YMPICV3XEZ3B47LJRQKGHTJ

? Verify Authentication Works
 GET https://10.1.1.4/mgmt/shared/authz/tokens/MB4YMPICV3XEZ3B47LJRQKGHTJ [200 OK, 1.23KB, 17ms]
 ✓  [GET Response Code]=200
 ✓  [Current Value] token=MB4YMPICV3XEZ3B47LJRQKGHTJ
 ✓  [Check Value] token == MB4YMPICV3XEZ3B47LJRQKGHTJ

? Set Authentication Token Timeout
 PATCH https://10.1.1.4/mgmt/shared/authz/tokens/MB4YMPICV3XEZ3B47LJRQKGHTJ [200 OK, 1.23KB, 50ms]
 ✓  [PATCH Response Code]=200
 ✓  [Current Value] timeout=1200
 ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 1197ms                     |
?-------------------------------?
| total data received: 1.71KB (approx)          |
?-------------------------------?
| average response time: 190ms                  |
?--------------------------------?
[f5-newman-build-1-2017-07-26-08-23-00] [runCollection][1 - Build a Basic LTM Config] running...
newman

f5-programmability-class-2

? 1 - Build a Basic LTM Config
? Step 1: Create a HTTP Monitor
 POST https://10.1.1.4/mgmt/tm/ltm/monitor/http [200 OK, 1.32KB, 625ms]

? Step 2: Create a Pool
 POST https://10.1.1.4/mgmt/tm/ltm/pool [200 OK, 1.56KB, 157ms]

? Step 3: Create a HTTP Profile
 POST https://10.1.1.4/mgmt/tm/ltm/profile/http [200 OK, 1.96KB, 183ms]

? Step 4: Create a TCP Profile
 POST https://10.1.1.4/mgmt/tm/ltm/profile/tcp [200 OK, 2.68KB, 64ms]

? Step 5: Create a Virtual Server
 POST https://10.1.1.4/mgmt/tm/ltm/virtual [200 OK, 1.9KB, 230ms]

?-----------------?-------?-------?
|                          | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        5 |        0 |
?-----------------?-------?-------?
|            test-scripts |        0 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        0 |        0 |
?-----------------?-------?-------?
|              assertions |        0 |        0 |
?-----------------?-------?-------?
| total run duration: 1406ms                    |
?--------------------------------?
| total data received: 5.79KB (approx)          |
?--------------------------------?
| average response time: 251ms                  |
?--------------------------------?
[f5-newman-build-1-2017-07-26-08-23-00] run completed in 6s, 90.207 ms
```

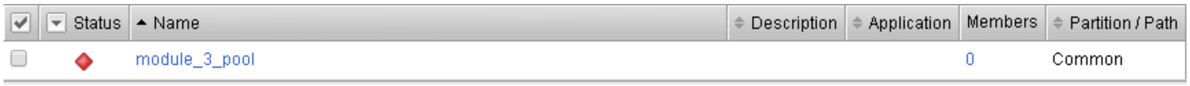
Note: Notice the 200 OK responses, the number of requests ect, we're building in testing and logging, look back at `BIGIP-A` for the newly created Application Service Framework

3. On BIG-IP A, examine Virtual Server `module_3_vs`:



☑	Status	Name	Description	Application	Destination	Service Port	Type	Resources	Partition / Path
□	◆	module_3_vs			10.1.20.129	80 (HTTP)	Standard	Edit...	Common

4. On BIG-IP A, examine Pool `module_3_pool`:



☑	Status	▲ Name	Description	Application	Members	Partition / Path
□	◆	module_3_pool			0	Common

Task 3 - Execute the second f5-newman-wrapper file

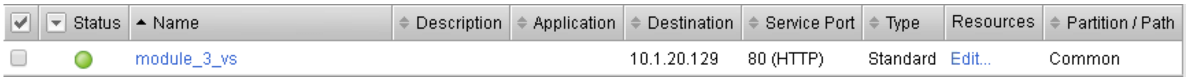
1. `f5-newman-build-2` contains calls to add pool members to the Application Service Framework created above; this is done independently of the build, to show Service staging as a possible use case.

   Execute: `f5-newman-wrapper f5-newman-build-2`

   Output should look like:

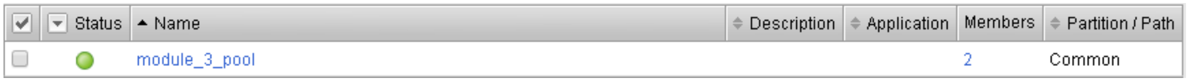
```
$ f5-newman-wrapper f5-newman-build-2
[f5-newman-build-2-2017-07-26-08-40-52] starting run
[f5-newman-build-2-2017-07-26-08-40-52] [runCollection][Authenticate to BIG-IP] running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
 POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 272ms]
 ✓  [POST Response Code]=200
 ✓  [Populate Variable] bigip_token=WSNAXWTCWNZGJG7MDBVF6CRXTB

? Verify Authentication Works
 GET https://10.1.1.4/mgmt/shared/authz/tokens/WSNAXWTCWNZGJG7MDBVF6CRXTB [200 OK, 1.23KB, 15ms]
 ✓  [GET Response Code]=200
 ✓  [Current Value] token=WSNAXWTCWNZGJG7MDBVF6CRXTB
 ✓  [Check Value] token == WSNAXWTCWNZGJG7MDBVF6CRXTB

? Set Authentication Token Timeout
 PATCH https://10.1.1.4/mgmt/shared/authz/tokens/WSNAXWTCWNZGJG7MDBVF6CRXTB [200 OK, 1.23KB, 61ms]
 ✓  [PATCH Response Code]=200
 ✓  [Current Value] timeout=1200
 ✓  [Check Value] timeout == 1200

?------------------?--------?--------?
|                          | executed |   failed |
?------------------?--------?--------?
|               iterations |        1 |        0 |
?------------------?--------?--------?
|                 requests |        3 |        0 |
?------------------?--------?--------?
|             test-scripts |        3 |        0 |
?------------------?--------?--------?
|       prerequest-scripts |        1 |        0 |
?------------------?--------?--------?
|               assertions |        8 |        0 |
?------------------?--------?--------?
| total run duration: 1034ms                     |
?-------------------------------?
| total data received: 1.71KB (approx)           |
?-------------------------------?
| average response time: 116ms                  |
?-------------------------------?
[f5-newman-build-2-2017-07-26-08-40-52] [runCollection][2 - Add Members to LTM Config] running...
newman

f5-programmability-class-2

? 2 - Add Members to LTM Config
? Step 1: Add Members to  Pool
 PATCH https://10.1.1.4/mgmt/tm/ltm/pool/module_3_pool [200 OK, 1.52KB, 143ms]

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        1 |        0 |
?-----------------?-------?-------?
|            test-scripts |        0 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        0 |        0 |
?-----------------?-------?-------?
|              assertions |        0 |        0 |
?-----------------?-------?-------?
| total run duration: 182ms                     |
?-------------------------------?
| total data received: 818B (approx)            |
?-------------------------------?
| average response time: 143ms                  |
?-------------------------------?
[f5-newman-build-2-2017-07-26-08-40-52] run completed in 4s, 328.497 ms
```

2. On BIG-IP A examine Virtual Server `module_3_vs`, the Virtual Server should be healthy and Green:



☑	Status	Name	Description	Application	Destination	Service Port	Type	Resources	Partition / Path
□	●	module_3_vs			10.1.20.129	80 (HTTP)	Standard	Edit...	Common

3. On BIG-IP A examine Pool `module_3_pool`:



☑	Status	Name	Description	Application	Members	Partition / Path
□	●	module_3_pool			2	Common

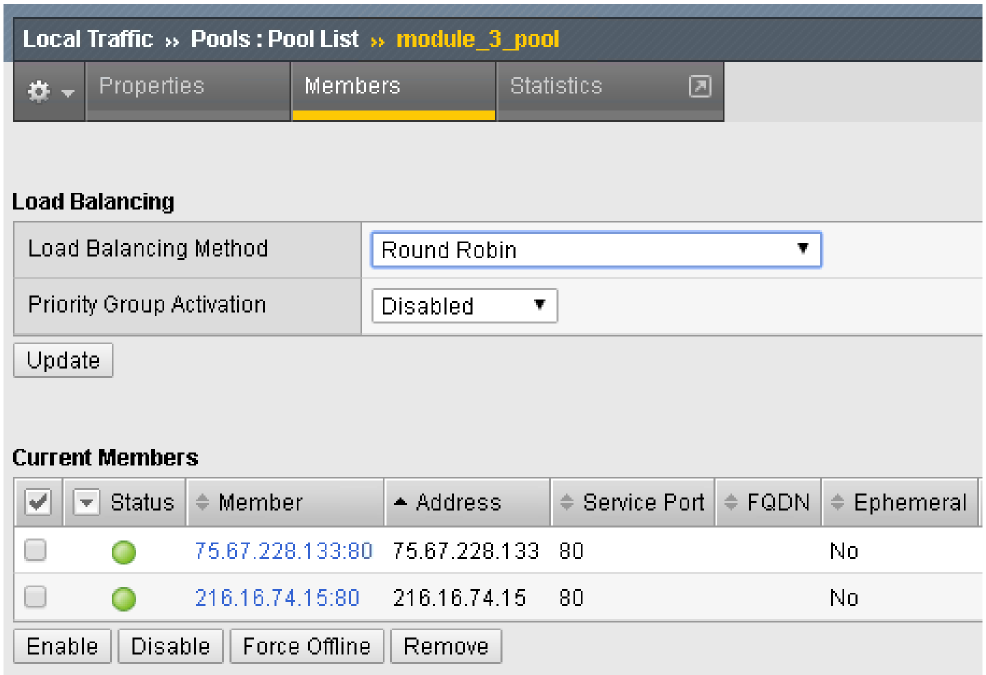
3.3.3 Lab 3.3 - Execute f5-newman-wrapper for an Operations Workflow

In the last lab we walked through creating an Application Service Framework, and then updating the Service Framework in a separate call. This lab has 2 f5-newman-files also, one used to user-down a pool member, and another to user-up the same member. These could be used as individual calls from another toolkit (which we'll see later) or run independently as a single commands.

Task 1 - Execute f5-newman-build-3

1. Return to or open a new session to the `super-netops-container` user credentials are `snops` and `default`

2. Navigate to the location containing the f5-newman-wrapper files `cd ~/f5-automation-labs/jenkins/f5-newman-operation`
3. On BIGIP-A, examine the pool `module_3_pool`, you should see 2 active (Green) pool members:



Load Balancing

Load Balancing Method	Round Robin ▼
Priority Group Activation	Disabled ▼

Current Members

☑	Status	Member	Address	Service Port	FQDN	Ephemeral
□	●	75.67.228.133:80	75.67.228.133	80		No
□	●	216.16.74.15:80	216.16.74.15	80		No

4. `f5-newman-build-3` contains calls to change the node state to `user-down` for `"bigip_pool_member":"75.67.228.133:80"`, both of these are specified as variables in the f5-newman-wrapper files.

   Execute: `f5-newman-wrapper f5-newman-build-3`

   Output should look like:

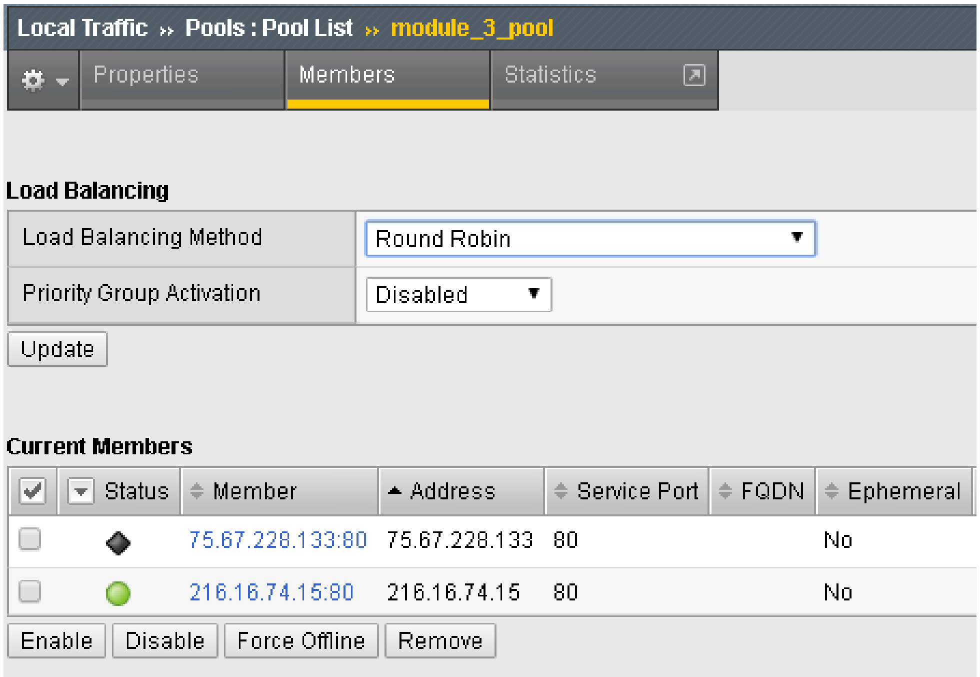
```
$ f5-newman-wrapper f5-newman-build-3
[f5-newman-build-3-2017-07-26-09-06-53] starting run
[f5-newman-build-3-2017-07-26-09-06-53] [runCollection][Authenticate to BIG-IP]␣
↪running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 267ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=JFN6TNIRAWEKNR5QPM26VT4QFE

? Verify Authentication Works
  GET https://10.1.1.4/mgmt/shared/authz/tokens/JFN6TNIRAWEKNR5QPM26VT4QFE [200␣
↪OK, 1.23KB, 22ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=JFN6TNIRAWEKNR5QPM26VT4QFE
  ✓  [Check Value] token == JFN6TNIRAWEKNR5QPM26VT4QFE

? Set Authentication Token Timeout
  PATCH https://10.1.1.4/mgmt/shared/authz/tokens/JFN6TNIRAWEKNR5QPM26VT4QFE [200␣
↪OK, 1.23KB, 26ms]
```

```
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 1243ms                    |
?--------------------------------?
| total data received: 1.71KB (approx)          |
?--------------------------------?
| average response time: 105ms                  |
?--------------------------------?
[f5-newman-build-3-2017-07-26-09-06-53] [runCollection][3 - Disable Node] running.
↪..
newman

f5-programmability-class-2

? 3 - Disable Node
? Step 1: Check Pool Exists
  GET https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool [200 OK, 1.56KB,␣
↪39ms]
  ✓  [GET Response Code]=200

? Step 2: Check Pool Member Exists
  GET https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool/members/~Common~75.
↪67.228.133:80 [200 OK, 1.25KB, 33ms]
  ✓  [GET Response Code]=200

? Step 3: Change Pool Member State
  PUT https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool/members/~Common~75.
↪67.228.133:80 [200 OK, 1.25KB, 298ms]
  ✓  [PUT Response Code]=200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        3 |        0 |
?-----------------?-------?-------?
```

```
| total run duration: 1092ms                      |
?-------------------------------?
| total data received: 1.89KB (approx)            |
?-------------------------------?
| average response time: 123ms                    |
?-------------------------------?
[f5-newman-build-3-2017-07-26-09-06-53] run completed in 6s, 564.868 ms
```

Note: Notice the 200 OK responses, as it completed successfully

5. Log back into BIG-IP A examine the pool `module_3_pool` status page:



Load Balancing

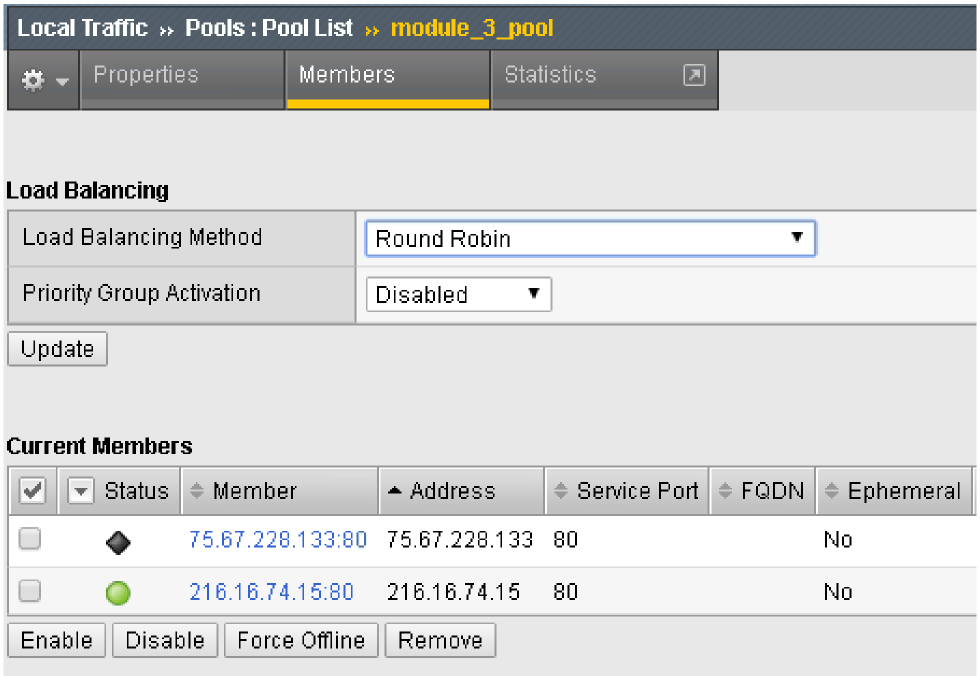
Load Balancing Method	Round Robin
Priority Group Activation	Disabled

Current Members

☑	Status	Member	Address	Service Port	FQDN	Ephemeral
□	◆	75.67.228.133:80	75.67.228.133	80		No
□	●	216.16.74.15:80	216.16.74.15	80		No

Task 2 - Execute f5-newman-build-4

1. Return to or open a new session to the `super-netops-container` user credentials are `snops` and `default`
2. Navigate to the location containing the f5-newman-wrapper files `cd ~/f5-automation-labs/jenkins/f5-newman-operation`
3. On BIG-IP A examine the pool `module_3_pool`, you should show only 1 Active and Green:



Local Traffic » Pools : Pool List » module_3_pool

Load Balancing

Load Balancing Method	Round Robin
Priority Group Activation	Disabled

Current Members

☑	Status	Member	Address	Service Port	FQDN	Ephemeral
□	◆	75.67.228.133:80	75.67.228.133	80		No
□	●	216.16.74.15:80	216.16.74.15	80		No

4. `f5-newman-build-3` contains calls to user-up variable node `"bigip_pool_member":"75.67.228.133:80"`

   Execute: `f5-newman-wrapper f5-newman-build-4`

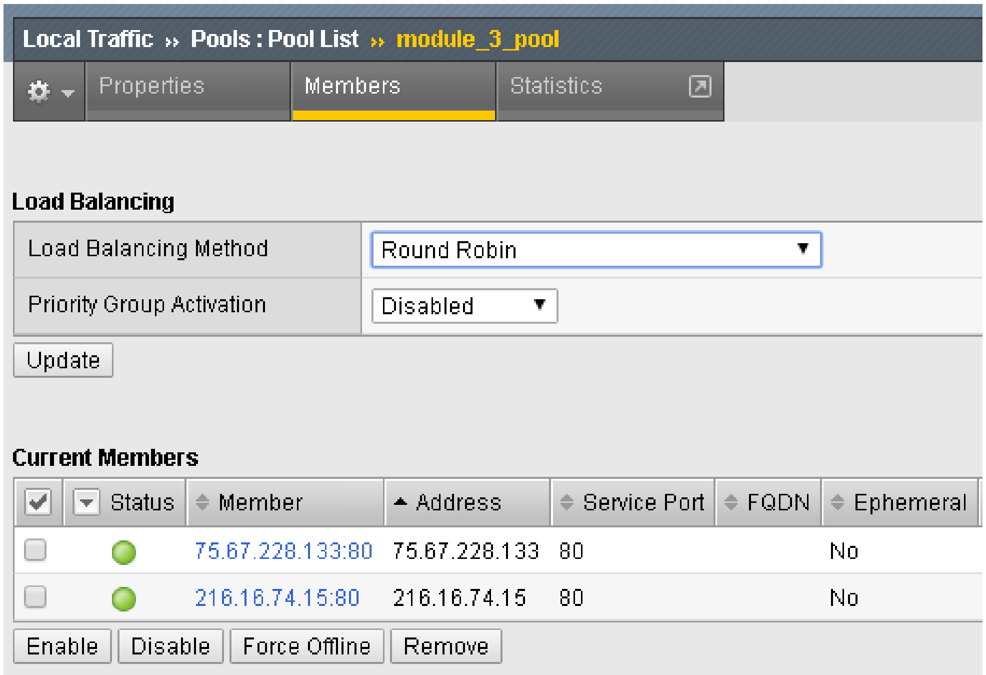
   Output should look like:

```
$ f5-newman-wrapper f5-newman-build-4
[f5-newman-build-4-2017-07-26-09-12-47] starting run
[f5-newman-build-4-2017-07-26-09-12-47] [runCollection][Authenticate to BIG-IP] running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 240ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=LN5IEBCKW5TTNXZLX5VYRUTOW5

? Verify Authentication Works
  GET https://10.1.1.4/mgmt/shared/authz/tokens/LN5IEBCKW5TTNXZLX5VYRUTOW5 [200 OK, 1.23KB, 15ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=LN5IEBCKW5TTNXZLX5VYRUTOW5
  ✓  [Check Value] token == LN5IEBCKW5TTNXZLX5VYRUTOW5

? Set Authentication Token Timeout
  PATCH https://10.1.1.4/mgmt/shared/authz/tokens/LN5IEBCKW5TTNXZLX5VYRUTOW5 [200 OK, 1.23KB, 27ms]
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
```

```
|                         | executed |   failed |
?----------------?-------?-------?
|              iterations |        1 |        0 |
?----------------?-------?-------?
|                requests |        3 |        0 |
?----------------?-------?-------?
|            test-scripts |        3 |        0 |
?----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?----------------?-------?-------?
|              assertions |        8 |        0 |
?----------------?-------?-------?
| total run duration: 922ms                      |
?------------------------------?
| total data received: 1.71KB (approx)           |
?------------------------------?
| average response time: 94ms                    |
?------------------------------?
[f5-newman-build-4-2017-07-26-09-12-47] [runCollection][4 - Enable Node] running..
↪.
newman

f5-programmability-class-2

? 4 - Enable Node
? Step 1: Check Pool Exists
  GET https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool [200 OK, 1.56KB,␣
↪31ms]
  ✓  [GET Response Code]=200

? Step 2: Check Pool Member Exists
  GET https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool/members/~Common~75.
↪67.228.133:80 [200 OK, 1.25KB, 28ms]
  ✓  [GET Response Code]=200

? Step 3: Change Pool Member State
  PUT https://10.1.1.4/mgmt/tm/ltm/pool/~Common~module_3_pool/members/~Common~75.
↪67.228.133:80 [200 OK, 1.25KB, 62ms]
  ✓  [PUT Response Code]=200

?----------------?-------?-------?
|                         | executed |   failed |
?----------------?-------?-------?
|              iterations |        1 |        0 |
?----------------?-------?-------?
|                requests |        3 |        0 |
?----------------?-------?-------?
|            test-scripts |        3 |        0 |
?----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?----------------?-------?-------?
|              assertions |        3 |        0 |
?----------------?-------?-------?
| total run duration: 519ms                      |
?------------------------------?
| total data received: 1.89KB (approx)           |
?------------------------------?
| average response time: 40ms                    |
?-------------------------------?
[f5-newman-build-4-2017-07-26-09-12-47] run completed in 4s, 510.429 ms
```

Note: Notice the 200 OK responses, as it completed successfully

5. On BIG-IP A examine Pool `module_3_pool` all Nodes should be back to the beginning state:



Load Balancing

Load Balancing Method	Round Robin
Priority Group Activation	Disabled

Current Members

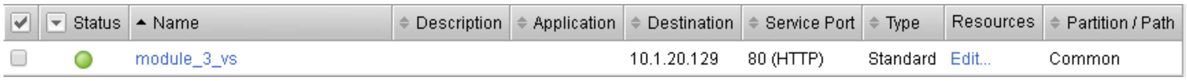
☑	Status	Member	Address	Service Port	FQDN	Ephemeral
□	●	75.67.228.133:80	75.67.228.133	80		No
□	●	216.16.74.15:80	216.16.74.15	80		No

3.3.4 Lab 3.3 - Execute an f5-newman-wrapper for Teardown

To get ready for the next module, we're going to execute one last f5-newman-wrapper directly. This file is designed to delete the framework and service we created in the last few labs. We used 2 f5-newman-wrapper files to create our service, but for the deletion we will only use 1.

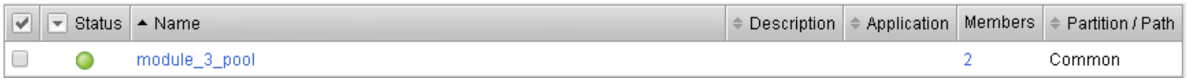
Task 1 - Execute f5-newman-build-5

1. Return to or open a new session to the `super-netops-container` user credentials are `snops` and `default`
2. Navigate to the location containing the f5-newman-wrapper files `cd ~/f5-automation-labs/jenkins/f5-newman-build`
3. On BIG-IP A examine the virtual server `module_3_vs`, it should be active and Green:



☑	Status	Name	Description	Application	Destination	Service Port	Type	Resources	Partition / Path
□	●	module_3_vs			10.1.20.129	80 (HTTP)	Standard	Edit...	Common

4. On BIGIP-A examine the pool `module_3_pool`, you should show 2 active members Green:



☑	Status	Name	Description	Application	Members	Partition / Path
□	●	module_3_pool			2	Common

5. `f5-newman-build-5` contains calls to delete all items we've created in the last few modules

   Execute: `f5-newman-wrapper f5-newman-build-5`

   Output should look like:

```
$ f5-newman-wrapper f5-newman-build-5
[f5-newman-build-5-2017-07-26-09-28-13] starting run
[f5-newman-build-5-2017-07-26-09-28-13] [runCollection][Authenticate to BIG-IP] running...
newman

BIGIP_API_Authentication

? 1_Authenticate
? Authenticate and Obtain Token
  POST https://10.1.1.4/mgmt/shared/authn/login [200 OK, 1.41KB, 194ms]
  ✓  [POST Response Code]=200
  ✓  [Populate Variable] bigip_token=NGEHHD6ZDJFD2MNF2UL3UXTGVH

? Verify Authentication Works
  GET https://10.1.1.4/mgmt/shared/authz/tokens/NGEHHD6ZDJFD2MNF2UL3UXTGVH [200 OK, 1.23KB, 16ms]
  ✓  [GET Response Code]=200
  ✓  [Current Value] token=NGEHHD6ZDJFD2MNF2UL3UXTGVH
  ✓  [Check Value] token == NGEHHD6ZDJFD2MNF2UL3UXTGVH

? Set Authentication Token Timeout
  PATCH https://10.1.1.4/mgmt/shared/authz/tokens/NGEHHD6ZDJFD2MNF2UL3UXTGVH [200 OK, 1.23KB, 17ms]
  ✓  [PATCH Response Code]=200
  ✓  [Current Value] timeout=1200
  ✓  [Check Value] timeout == 1200

?-----------------?-------?-------?
|                         | executed |   failed |
?-----------------?-------?-------?
|              iterations |        1 |        0 |
?-----------------?-------?-------?
|                requests |        3 |        0 |
?-----------------?-------?-------?
|            test-scripts |        3 |        0 |
?-----------------?-------?-------?
|      prerequest-scripts |        1 |        0 |
?-----------------?-------?-------?
|              assertions |        8 |        0 |
?-----------------?-------?-------?
| total run duration: 835ms                      |
?-------------------------------?
| total data received: 1.71KB (approx)           |
?-------------------------------?
| average response time: 75ms                    |
?-------------------------------?
[f5-newman-build-5-2017-07-26-09-28-13] [runCollection][5 - Clean Up Service] running...
newman

f5-programmability-class-2

```

```
? 5 - Clean Up Service
? Step 1: Delete a Virtual Server
  DELETE https://10.1.1.4/mgmt/tm/ltm/virtual/module_3_vs [200 OK, 740B, 57ms]

? Step 2: Delete a TCP Profile
  DELETE https://10.1.1.4/mgmt/tm/ltm/profile/tcp/module_3_tcp_clientside [200 OK,
↪ 740B, 88ms]

? Step 3: Delete a HTTP Profile
  DELETE https://10.1.1.4/mgmt/tm/ltm/profile/http/module_3_http [200 OK, 740B,␣
↪56ms]

? Step 4: Delete a Pool
  DELETE https://10.1.1.4/mgmt/tm/ltm/pool/module_3_pool [200 OK, 740B, 47ms]

? Step 5: Delete a HTTP Monitor
  DELETE https://10.1.1.4/mgmt/tm/ltm/monitor/http/module_3_http_monitor [200 OK,␣
↪740B, 59ms]

?------------------?--------?-------?
|                         | executed |   failed |
?------------------?--------?-------?
|              iterations |        1 |        0 |
?------------------?--------?-------?
|                requests |        5 |        0 |
?------------------?--------?-------?
|            test-scripts |        0 |        0 |
?------------------?--------?-------?
|      prerequest-scripts |        0 |        0 |
?------------------?--------?-------?
|              assertions |        0 |        0 |
?------------------?--------?-------?
| total run duration: 445ms                      |
?--------------------------------?
| total data received: 0B (approx)               |
?--------------------------------?
| average response time: 61ms                    |
?--------------------------------?
[f5-newman-build-5-2017-07-26-09-28-13] run completed in 4s, 267.464 ms
```

Note: Notice the 200 OK responses, as it completed successfully

6. On BIG-IP A examine Virtual `module_3_vs` and Pool `module_3_pool` are deleted

3.4 Module 4: Continuous Integration / Continuous Delivery

This Module will continue to build up our Infrastructure to a Self-Service or CI/CD goal. We will be building on the code that was utilized in the previous modules and labs, though now we'll use Jenkins to provide a CI/CD mechanism. This lab will also use Slack to notify users of changes going on in real time.

Tools we will be using:

- f5-newman-wrapper & previous workflows
  - The previous 5 wrapper workflows files will be executed, but from a Jenkins Pipeline

- f5-super-netops-container
  - Continuing delivery of F5 configuration from a self contained toolkit
  - This version or **variant** of the container has Jenkins installed for you, this is depicted from tag associated to the Docker Image `f5devcentral/f5-super-netops-container:jenkins`
- Slack
  - There has been a Slack channel already setup on your behalf, which we will all be monitoring for environment changes
  - Any person with an @f5.com email address can join the Slack Channel. To join and view the transactions use https://f5agilitydevops.slack.com/signup
- Jenkins
  - Jenkins is installed on the f5-super-netops-container, accessed via `http://10.1.1.8:10000` (Web) user credentials are `admin/default`

From the previous labs you should already have your Super-NetOps-Container already running, if it's not please refer to Class 2 Module 2 on starting the service.

3.4.1 Lab 4.1 - File Locations and Jenkins setup

We've been executing all our commands locally from Automated scripts; we are now going to take the different toolkits and tie them together to form a **Pipeline**. Pipelines will vary in deployments and even within solutions. Our lab will show you just one way one could be utilized.

Task 1 - Locating the Jenkins files and how they are setup

1. Return to or open a new session to the `super-netops-container`, user credentials are `snops` and `default`
2. During the installation of the f5-super-netops-container there were several github repositories cloned, all of which are mapped to the `/home/snops/` directory. Lets make sure the Jekins files were mapped correctly.

   Execute: `cd ~/f5-automation-labs/jenkins` to access our folder containing the Jenkins Pipeline Files
3. The Jenkins files are located alongside the f5-newman-wrapper files we've used in the previous labs, setup this way was for ease of learning. You may place tools in different structures in your environment.

   File Locations:

   ```
   |- /f5-automation-labs
      |- /jenkins
      |   |   /f5-newman-build
      |   |       Jenkinsfile1-2
      |   |       Jenkinsfile5
      |   |   /f5-newman-operation
      |   |       Jenkinsfile3
      |   |       Jenkinsfile4
   ```
4. Lets review the first Jenkins file, from the current folder structure execute `cat Jenkinsfile1-2`

   File output:

```
node {
   stage('Testing') {
      //Run the tests
      //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-1"
      //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-2"
   }
   stage('Frameword-Deployment') {
       //Run SNOPS Container Newman Package Virtual and Pool
      sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-1"
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'good',
         message: 'Super-NetOps Engineer is about to deploy an F5 Service Framework, Approval Needed!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
   stage('Approval') {
      //Gate the process and require approval
      input 'Proceed?'
      //chatops slack message that run has completed
      slackSend(
          channel: '#jenkins_builds',
          color: 'good',
          message: 'Super-NetOps Engineer just approved a new F5 Service Framework, thats some serious Continuous Delivery!',
          teamDomain: 'f5agilitydevops',
          token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
          )
   }
   stage('Add-Sevice-Node') {
       //Run SNOPS Container Newman Package add Node to Pool
      sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-2"
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'good',
         message: 'Super-NetOps Engineer just added a Node to a Service, Production is Online!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
}
```

- This is a Jenkins Pipeline file, which we will be inputing into a Pipeline deployment via our Jenkins Toolkit.
- The file should be human readable even without Jenkins experience, a `stage` can be thought of as a step in the Pipeline (or a work-center in manufacturing terms); right after the stage is its name, followed by some commands. Since the super-netops-container is running this Jenkins installation locally, we can use local mappings to file structure.

- In more common deployments the Jenkins file would be stored in a SCM (like Github) and called during an Event (Build/Pull Request) or a Polling Timer, or even some other kind of scripting launch.
- Testing in Pipeline before executing code with tools like `linter` or python scripts can make sure formatting is valid, reducing errors from happening during builds.

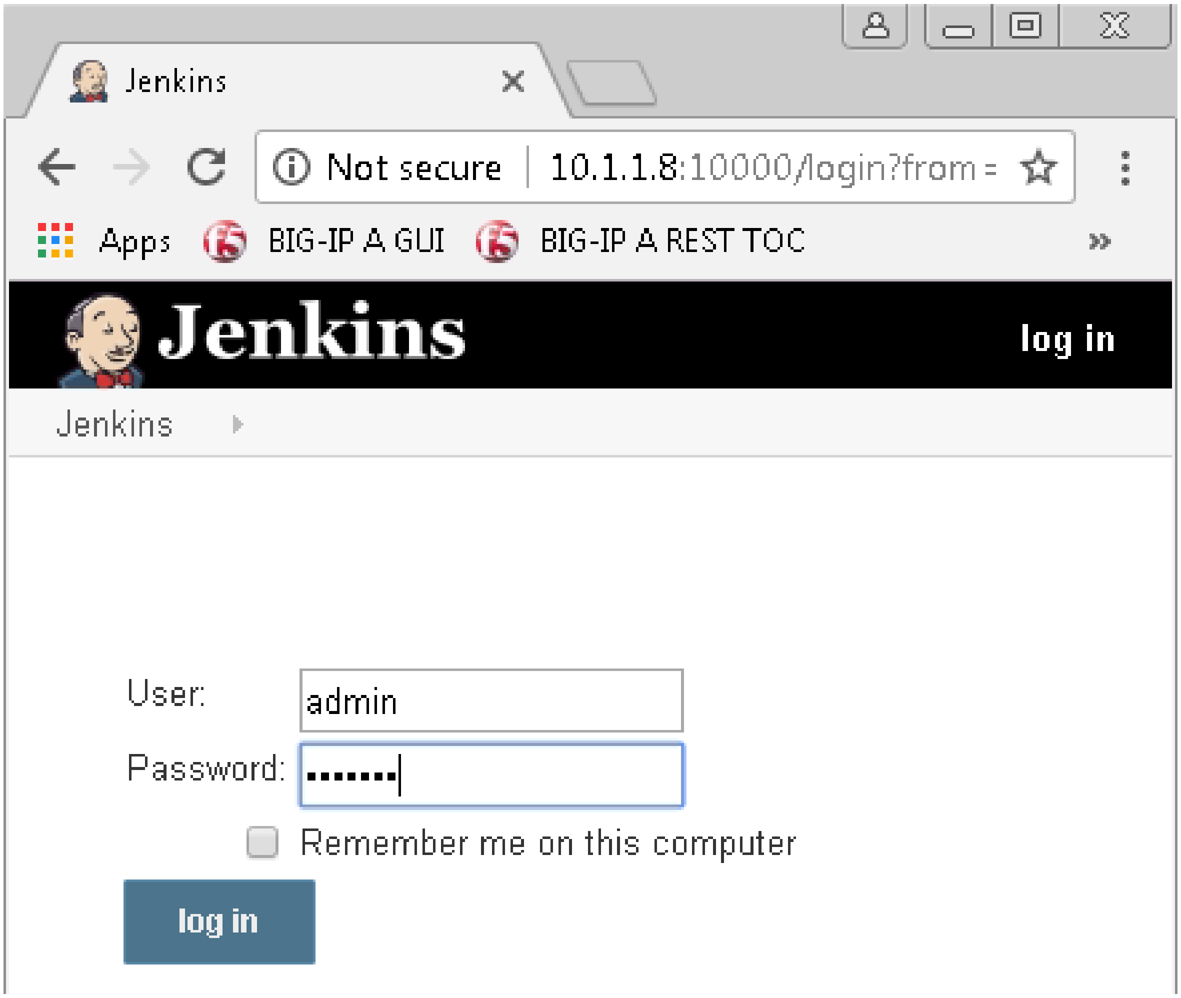
Our installation also has some Slack calls. Which we will setup next.

Task 2 - Accessing Jenkins and Installing the Slack-Notifier Plug-in

Slack is a ChatOps toolkit, think of Skype, Teams, Messenger, or IIRC! Except Slack also has the ability to take in bots. slackbots are used to interact with services, they might query for something when asked, or give you information when they notice something. In our case our Jenkins Pipeline file will use Slack to notify all of us when an action happens, collaborative teamwork.

> **Note:** In the Jenkins Files, the `message` piece is sent to the Slack channel, if you would like to modify your messages for our lab **change the text!**

1. Access Jenkins via Chrome, there is already a bookmark `Jenkins` created on your behalf , the user credentials are `admin\default`.

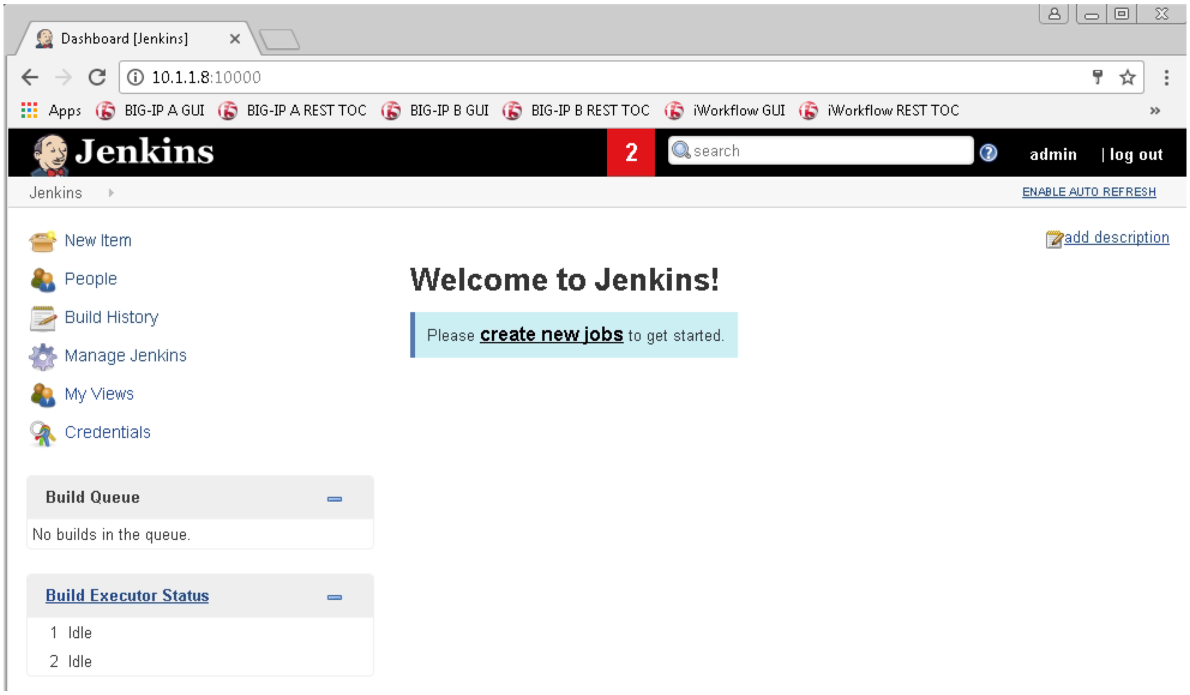


User: admin

Password: •••••••

□ Remember me on this computer

2. Once you are logged into Jenkins it should look like below



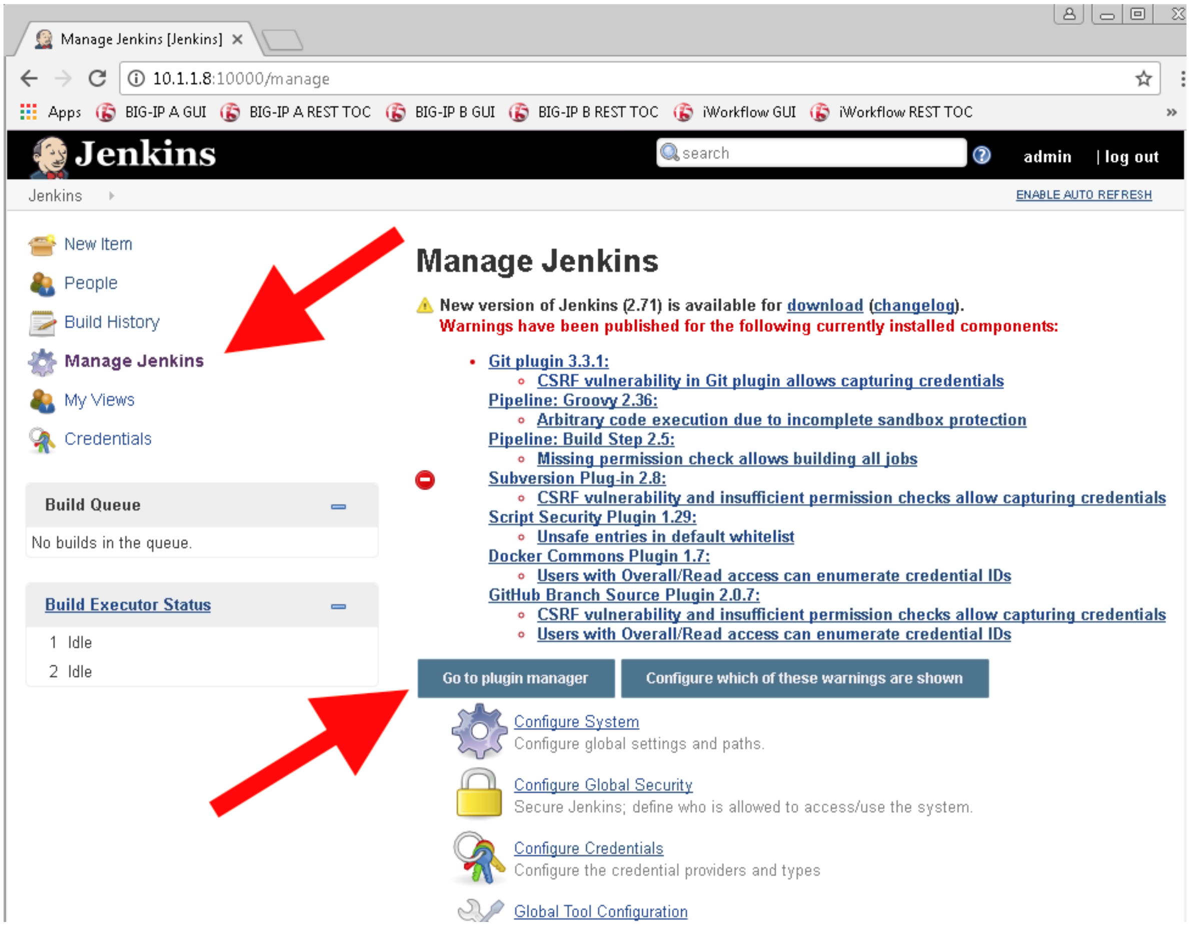
Welcome to Jenkins!

Please **create new jobs** to get started.

No builds in the queue.

1 Idle
2 Idle

3. Click on Manage Jenkins



Manage Jenkins

⚠ New version of Jenkins (2.71) is available for download (changelog).

Warnings have been published for the following currently installed components:

- Git plugin 3.3.1:
  - CSRF vulnerability in Git plugin allows capturing credentials
- Pipeline: Groovy 2.36:
  - Arbitrary code execution due to incomplete sandbox protection
- Pipeline: Build Step 2.5:
  - Missing permission check allows building all jobs
- Subversion Plug-in 2.8:
  - CSRF vulnerability and insufficient permission checks allow capturing credentials
- Script Security Plugin 1.29:
  - Unsafe entries in default whitelist
- Docker Commons Plugin 1.7:
  - Users with Overall/Read access can enumerate credential IDs
- GitHub Branch Source Plugin 2.0.7:
  - CSRF vulnerability and insufficient permission checks allow capturing credentials
  - Users with Overall/Read access can enumerate credential IDs

No builds in the queue.

1 Idle
2 Idle

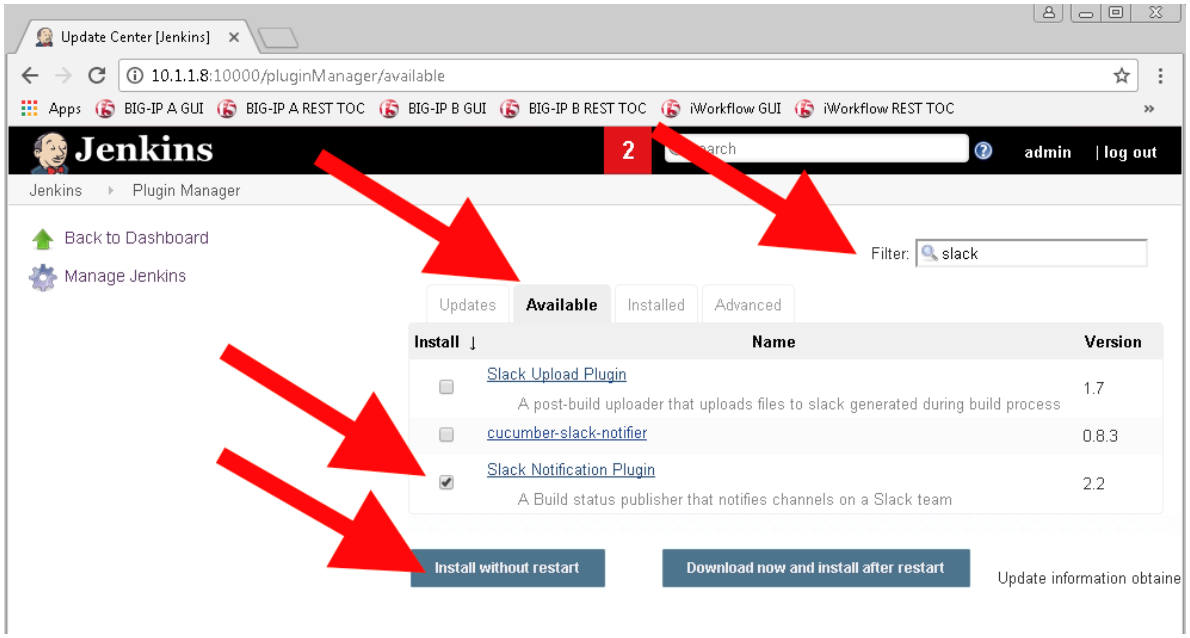
Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration

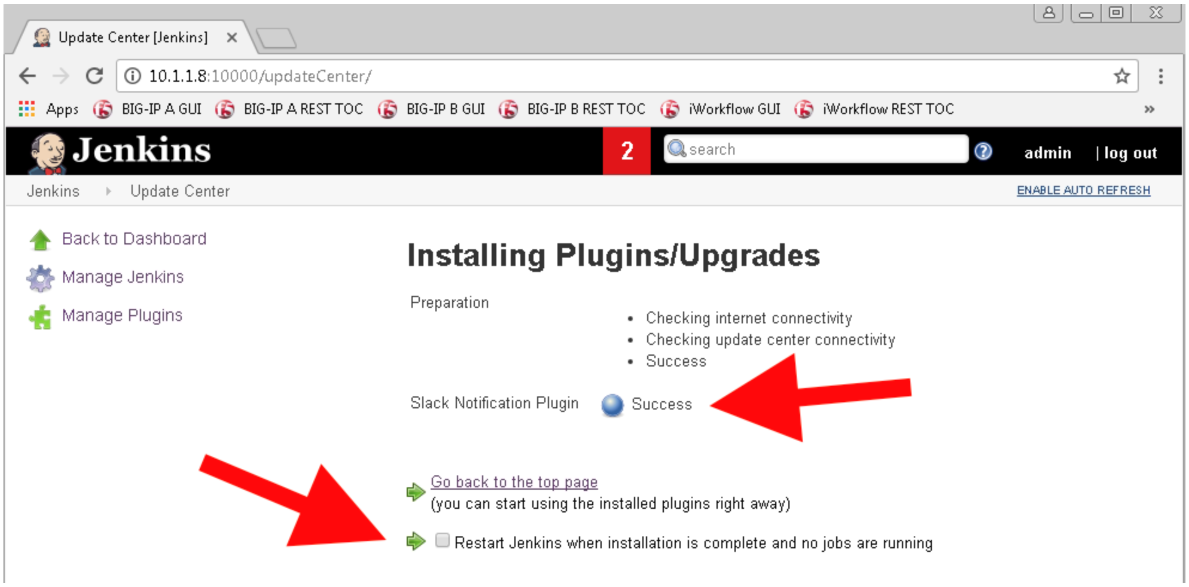
4. On the Manage Jenkins tab Select `Available` then filter on `slack`, once the filter is complete choose `Slack Notification Plugin` and execute `Install without Restart`



Filter: slack

Install ↓	Name	Version
□	Slack Upload Plugin A post-build uploader that uploads files to slack generated during build process	1.7
□	cucumber-slack-notifier	0.8.3
☑	Slack Notification Plugin A Build status publisher that notifies channels on a Slack team	2.2

5. Once the Slack Notification Plugin has changed to `Success`, tick the radio button for `Restart Jenkins when installation is complete and no jobs are running`



Installing Plugins/Upgrades

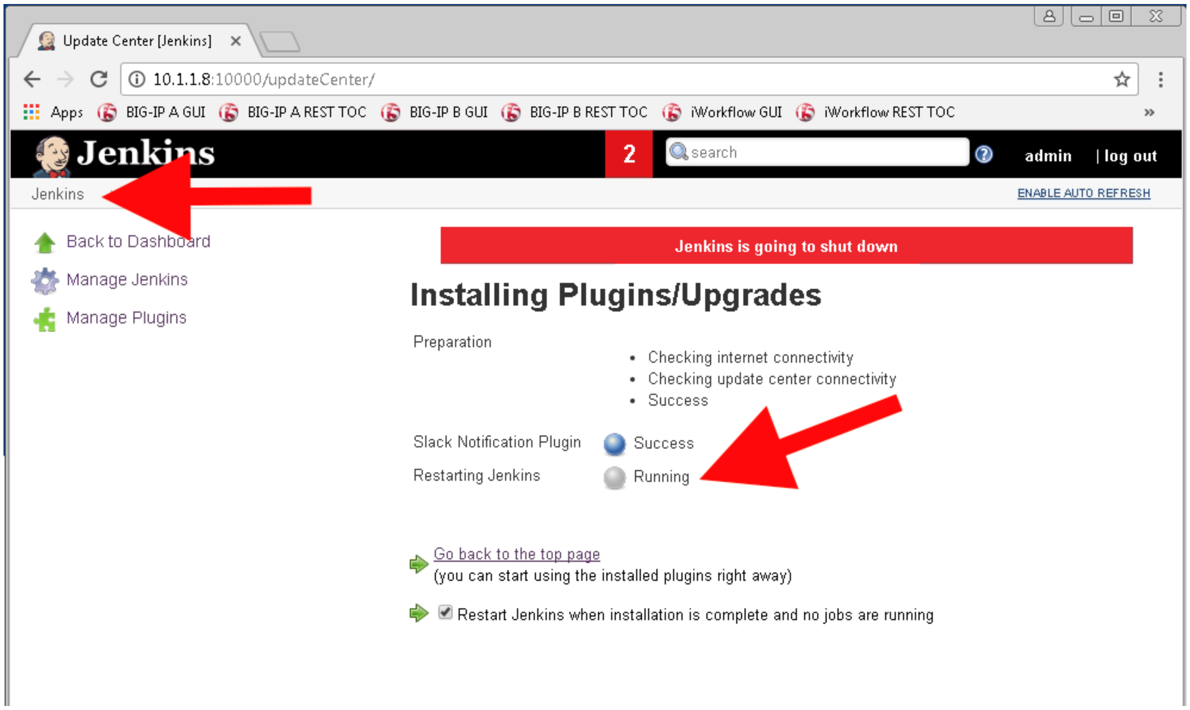
Preparation
- Checking internet connectivity
- Checking update center connectivity
- Success

Slack Notification Plugin ● Success

Go back to the top page
(you can start using the installed plugins right away)

□ Restart Jenkins when installation is complete and no jobs are running

6. Slack can take a few minutes to install in the background (give it 30 seconds), once the `Restarting Jenkins` globe is grey and the status is `Running` go back to Jenkins Home



Jenkins is going to shut down

Installing Plugins/Upgrades

Preparation
- Checking internet connectivity
- Checking update center connectivity
- Success

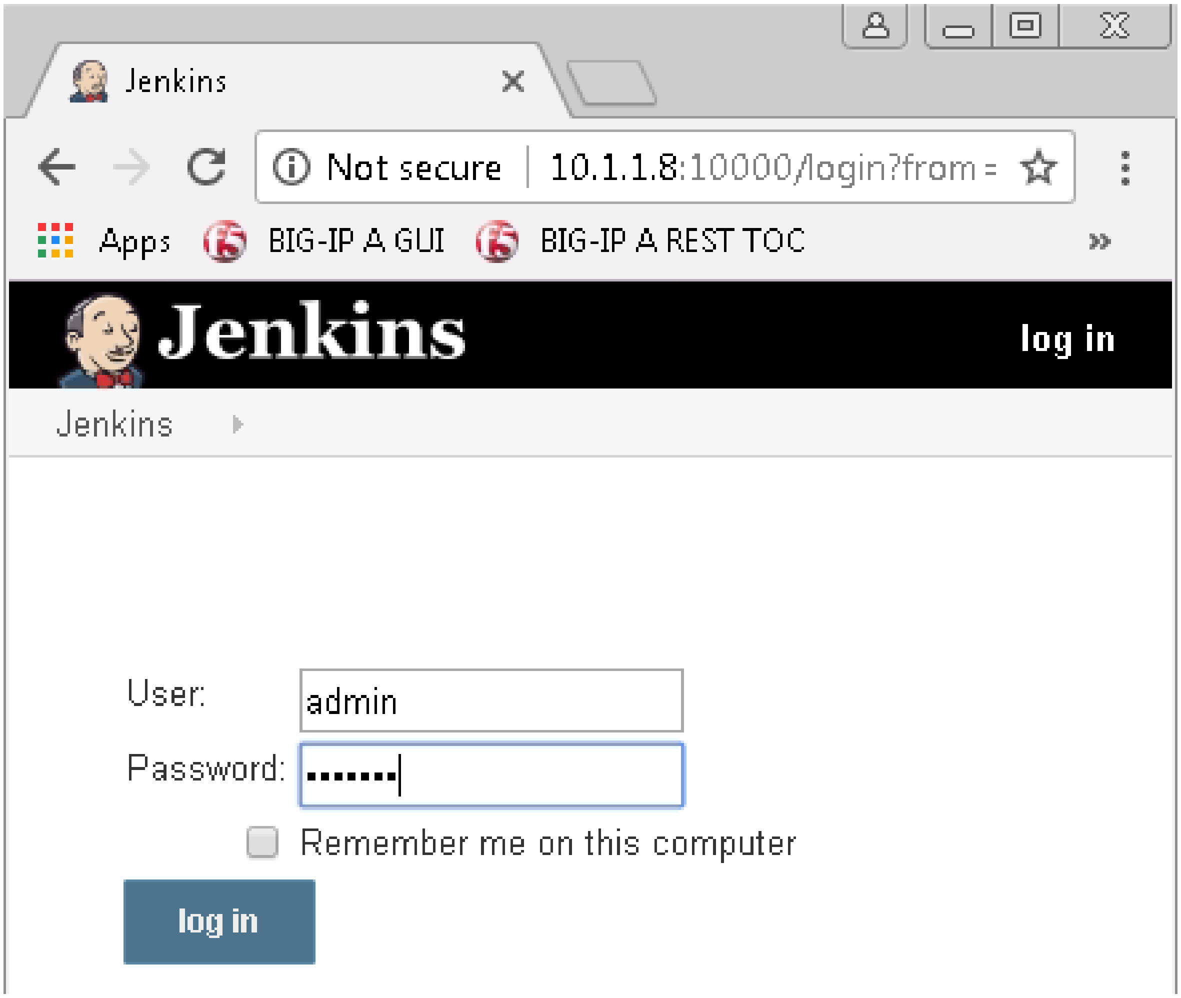
Slack Notification Plugin ● Success

Restarting Jenkins ○ Running

Go back to the top page
(you can start using the installed plugins right away)

☑ Restart Jenkins when installation is complete and no jobs are running

7. Executing a restart of Jenkins will stop your session, you will need to log back into the system



User: admin

Password: ••••••••

□ Remember me on this computer

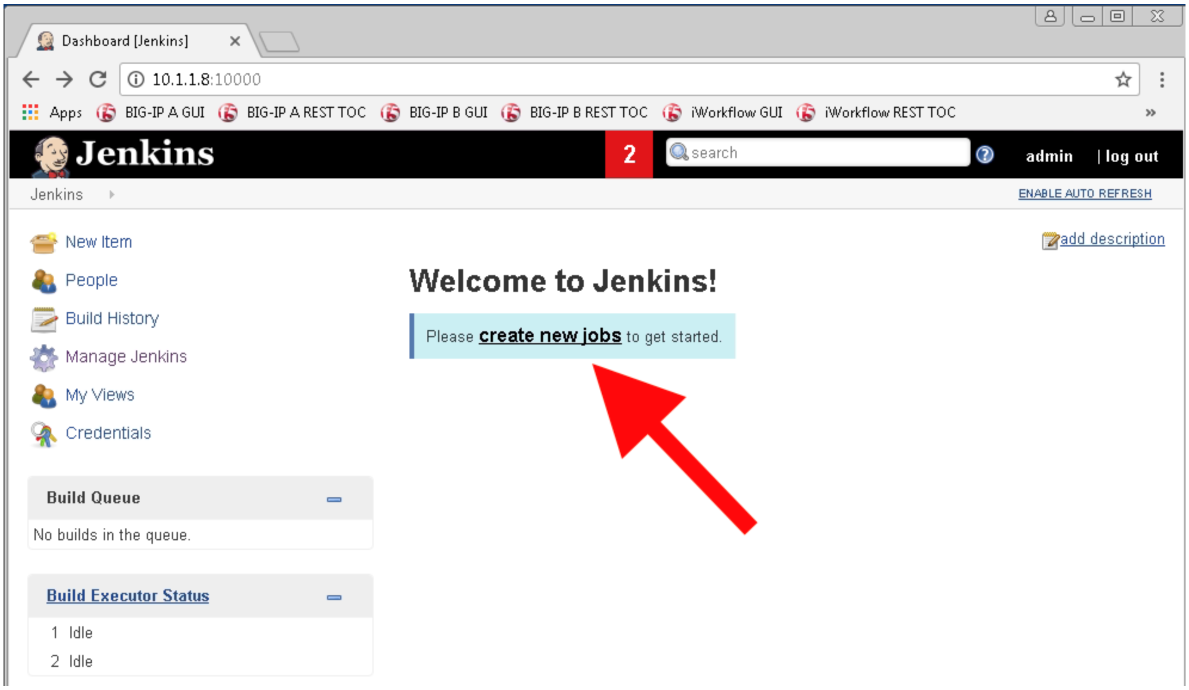
3.4.2 Lab 4.2 - Executing Jenkins Jobs for Creation or Modify

Now that we have Jenkins running, and the dependent Slack Plugin installed we can utilize our Jenkins Pipeline Scripts successfully.

Task 1 - Building the Application Service Framework via Jenkins

This step is executing the f5-newman-wrapper files. Instead of having to run the two different builds (Application Service Framework and Pool member add) individually we'll us a pause. Jenkins has a pause functionality which pausing the deployment looking for an approval to continue. After the approving step, the node to be added, still using 2 f5-newman-wrapper files but in conjunction with a single solution (Jenkins). Jenkins will continue to update the class via Slack as people are progressing. Jenkins does also keep a running console for logging, which we will also review.

1. From the Jenkins Dashboard click on `create new jobs`



Welcome to Jenkins!

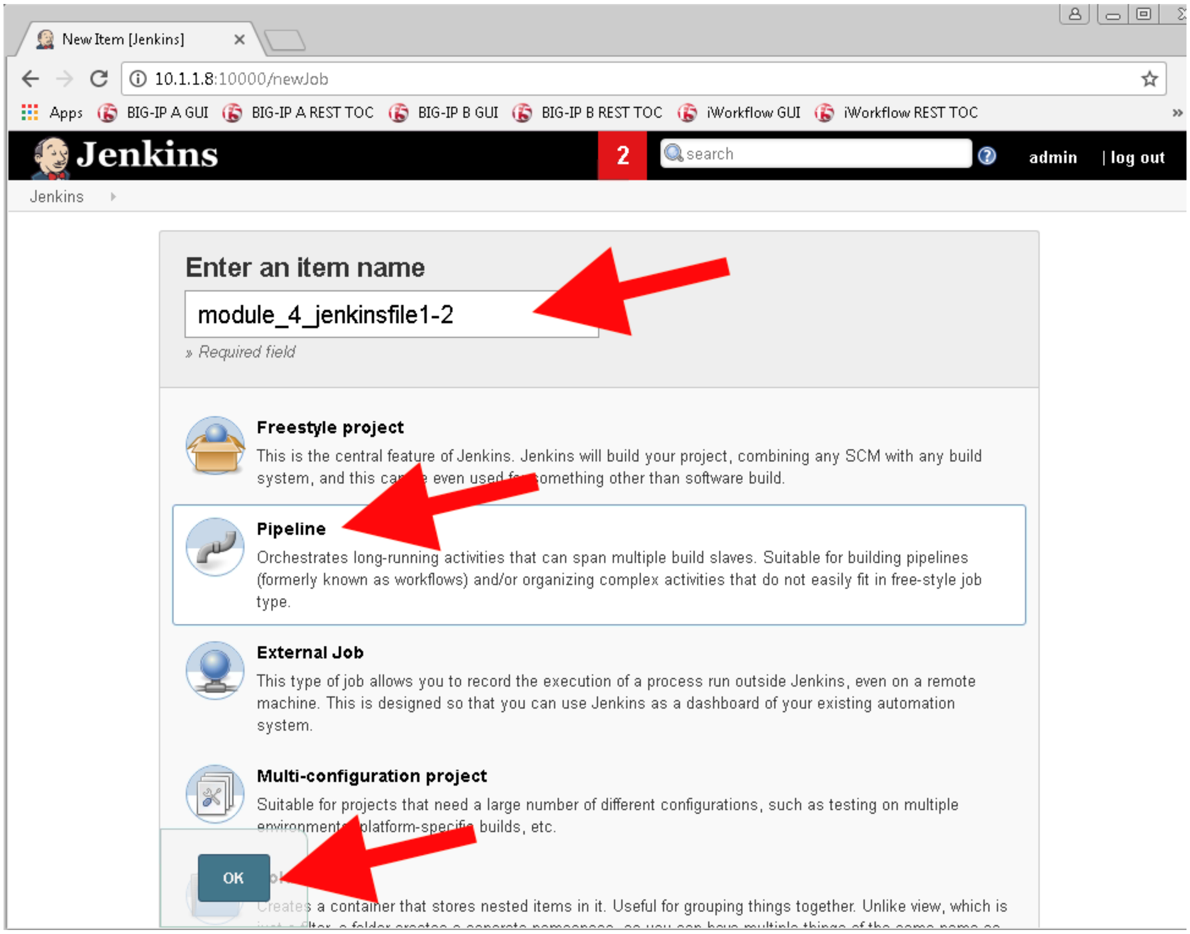
Please **create new jobs** to get started.

No builds in the queue.

1 Idle

2 Idle

2. We are going to create our first Pipeline Job. Name the item `module_4_jenkinsfile1-2`, choose the `Pipeline` project style and select `OK`



Enter an item name

module_4_jenkinsfile1-2

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job

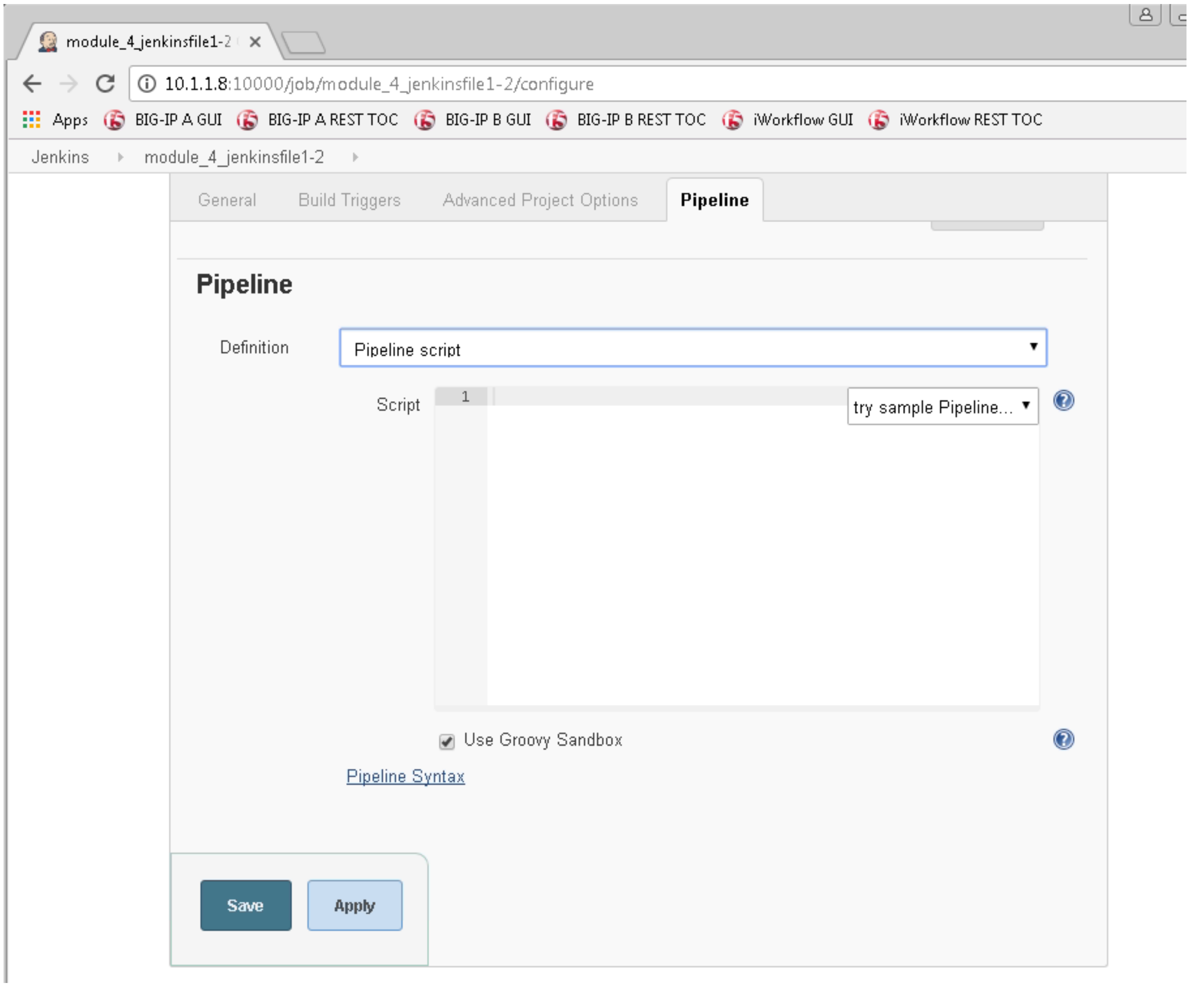
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is

3. We are going to be using the raw `Jenkinsfile1-2` right in the `Pipeline Script` option at the end of the config page. Scroll to the bottom of the page but **please look at the other options** which can deploy a Pipeline. The different options in here are for an SCM (like GitHub), the `Polling` or `Commit` methods enable Continuous Deployment, as Jenkins will deploy the change on an event basis. Tie this with automatic testing to make sure you're not breaking the build!



10.1.1.8:10000/job/module_4_jenkinsfile1-2/configure

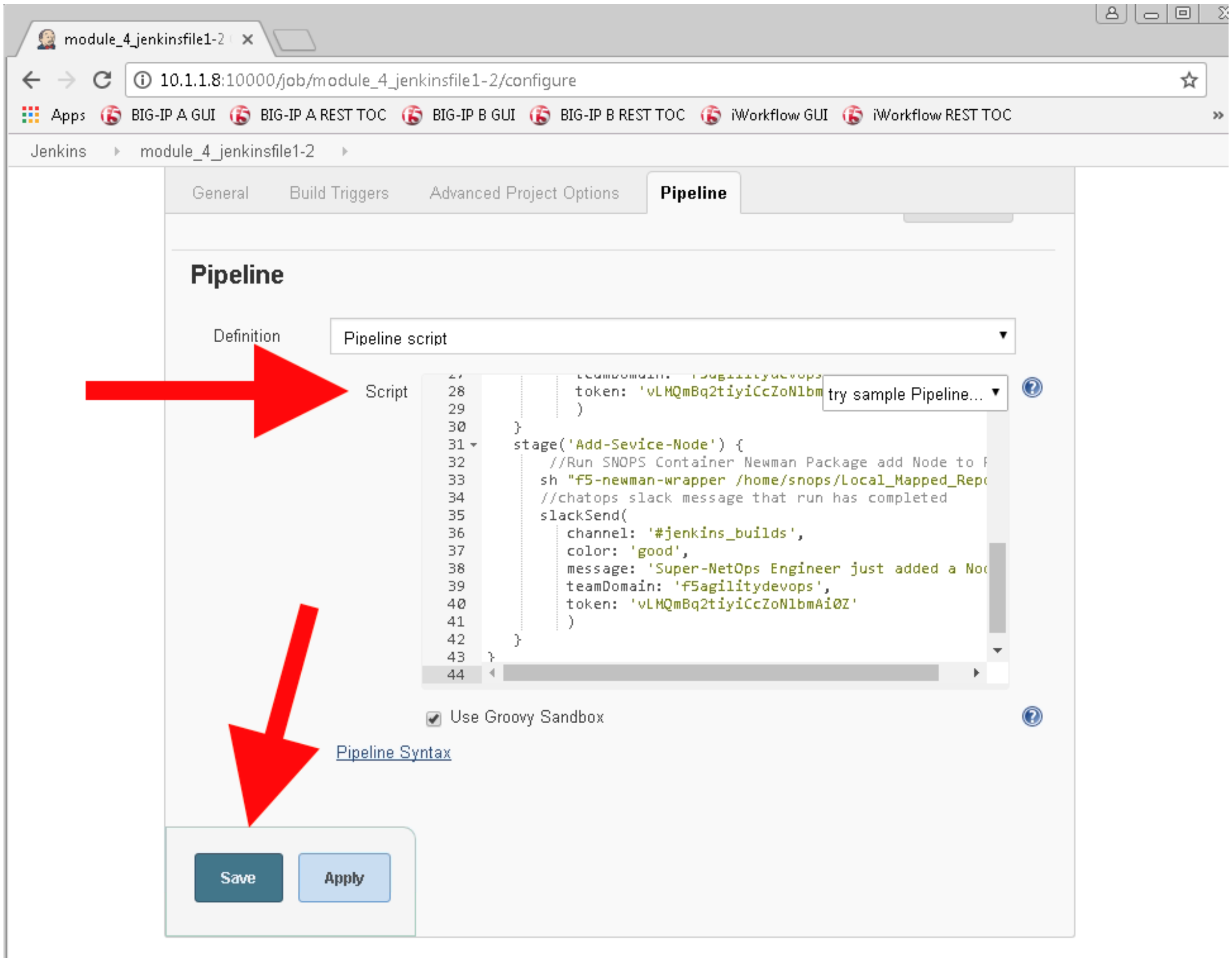
Definition: Pipeline script

☑ Use Groovy Sandbox

4. We need to enter the contents of the `Jenkinsfile1-2` into the `Script` section under Pipeline. After the contents are added click the `Save` Option.

```
node {
   stage('Testing') {
      //Run the tests
      //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-1"
      //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-2"
   }
   stage('Frameword-Deployment') {
       //Run SNOPS Container Newman Package Virtual and Pool
      sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-1"
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'good',
         message: 'Super-NetOps Engineer is about to deploy an F5 Service Framework,␣
↪Approval Needed!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
   stage('Approval') {
      //Gate the process and require approval
      input 'Proceed?'
      //chatops slack message that run has completed
      slackSend(
          channel: '#jenkins_builds',
          color: 'good',
          message: 'Super-NetOps Engineer just approved a new F5 Service Framework,␣
↪thats some serious Continuous Delivery!',
          teamDomain: 'f5agilitydevops',
          token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
          )
   }
   stage('Add-Sevice-Node') {
       //Run SNOPS Container Newman Package add Node to Pool
      sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-build/
↪f5-newman-build-2"
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'good',
         message: 'Super-NetOps Engineer just added a Node to a Service, Production␣
↪is Online!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
}
```

Contents in Pipeline:



Pipeline

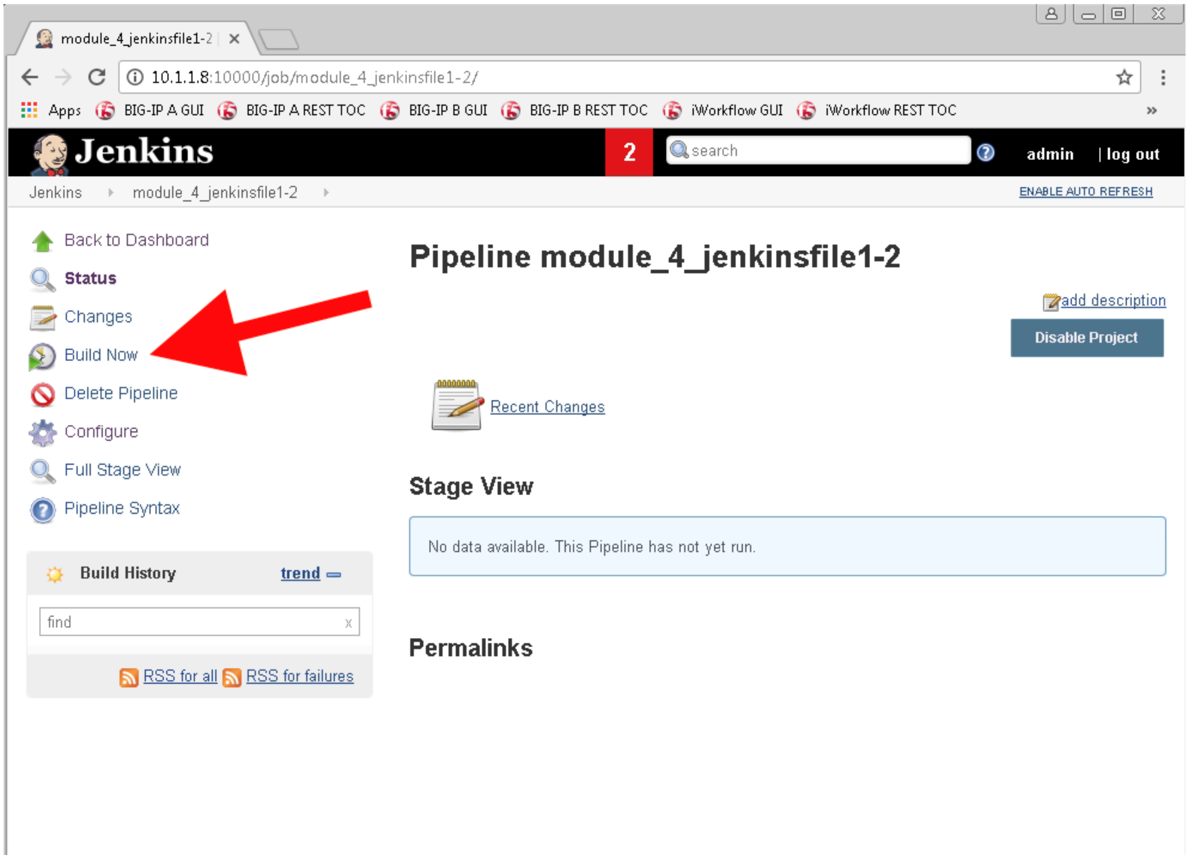
Definition: Pipeline script

Script

```
                teamDomain: 'f5agilitydevops',
                token: 'vLMQmBq2tiyiCcZoNlbm
                )
    }
    stage('Add-Sevice-Node') {
         //Run SNOPS Container Newman Package add Node to F
       sh "f5-newman-wrapper /home/snops/Local_Mapped_Repo
       //chatops slack message that run has completed
       slackSend(
           channel: '#jenkins_builds',
           color: 'good',
           message: 'Super-NetOps Engineer just added a Noc
           teamDomain: 'f5agilitydevops',
           token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
           )
    }
}
```

☑ Use Groovy Sandbox

1. Once the Job is saved, you will be taken to the stage view page, from here we are going to execute our Pipeline build, choose the `Build Now` option.



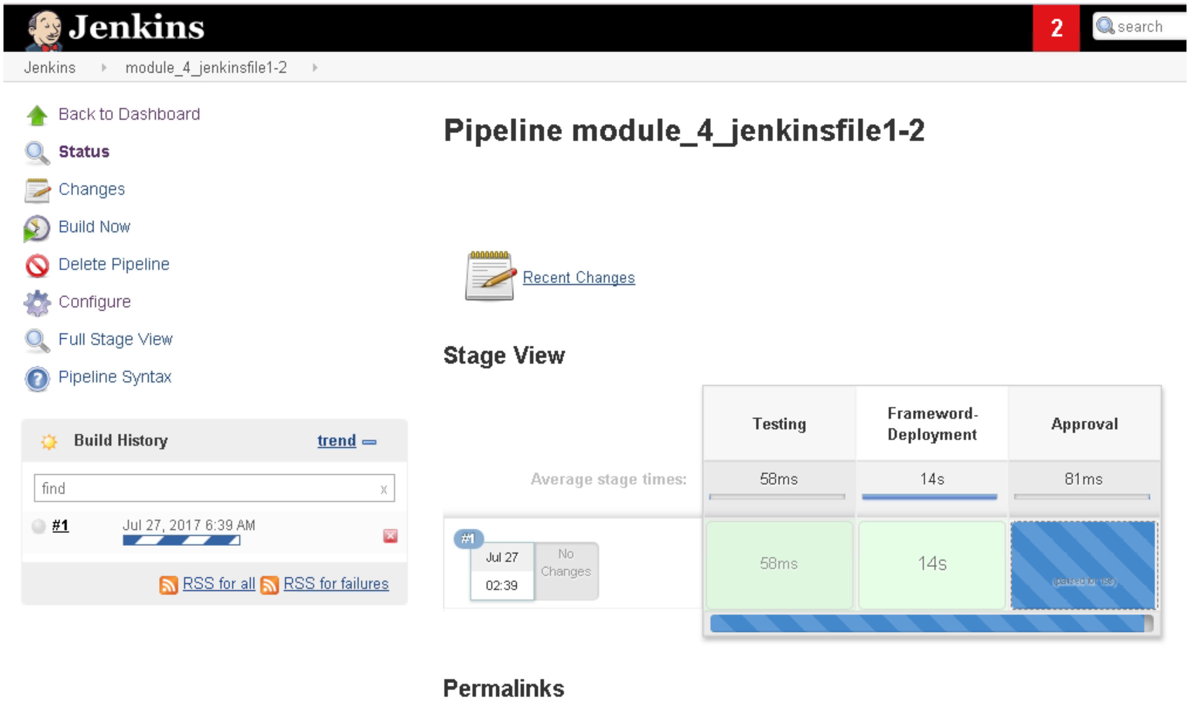
Pipeline module_4_jenkinsfile1-2

Stage View

No data available. This Pipeline has not yet run.

Permalinks

2. The Build is now running, and the stages are being executed in order. However, on our third stage we have a **pause** and an approval **needed**. Also at the same time Slack has began to notify us that a new service is being deployed, and someone needs to approve it.

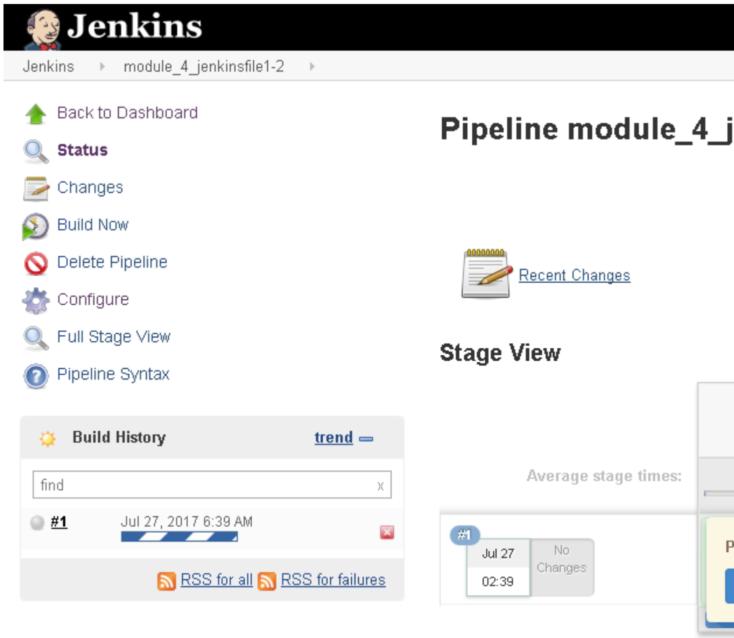


Pipeline module_4_jenkinsfile1-2

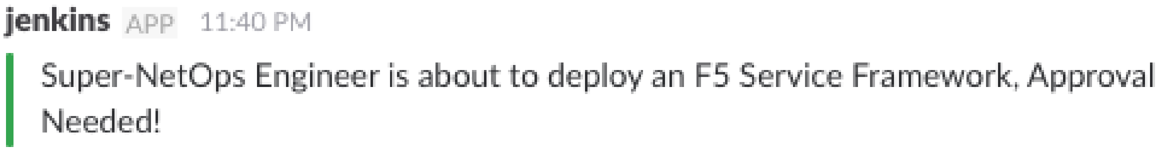
Stage View

	Testing	Frameword-Deployment	Approval
Average stage times:	58ms	14s	81ms
#1 Jul 27 02:39 No Changes	58ms	14s	

Permalinks

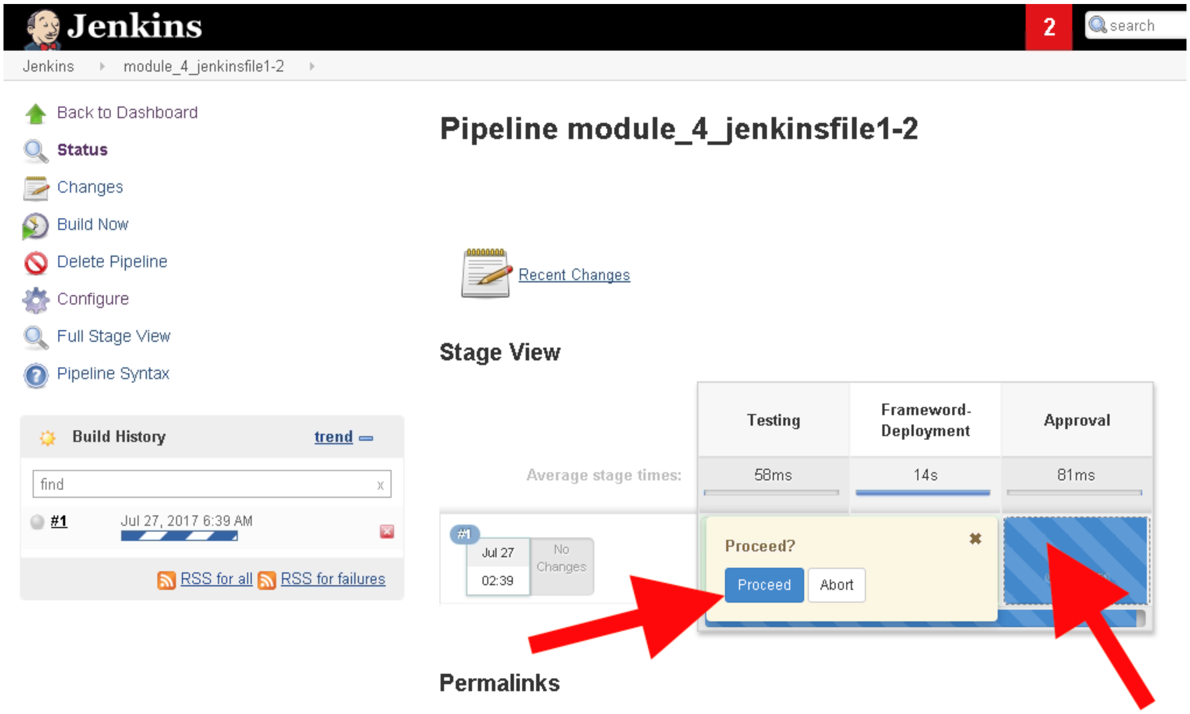


Highlight over the third Stage to prompt for the Approval



Super-NetOps Engineer is about to deploy an F5 Service Framework, Approval Needed!

3. Approve the change in Jenkins to allow the build to finish. Once this is done, the approval and finished Slack notification will be sent.



Jenkins 2 search

Jenkins > module_4_jenkinsfile1-2

- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Full Stage View
- Pipeline Syntax

Build History trend

find

#1 Jul 27, 2017 6:39 AM

RSS for all RSS for failures

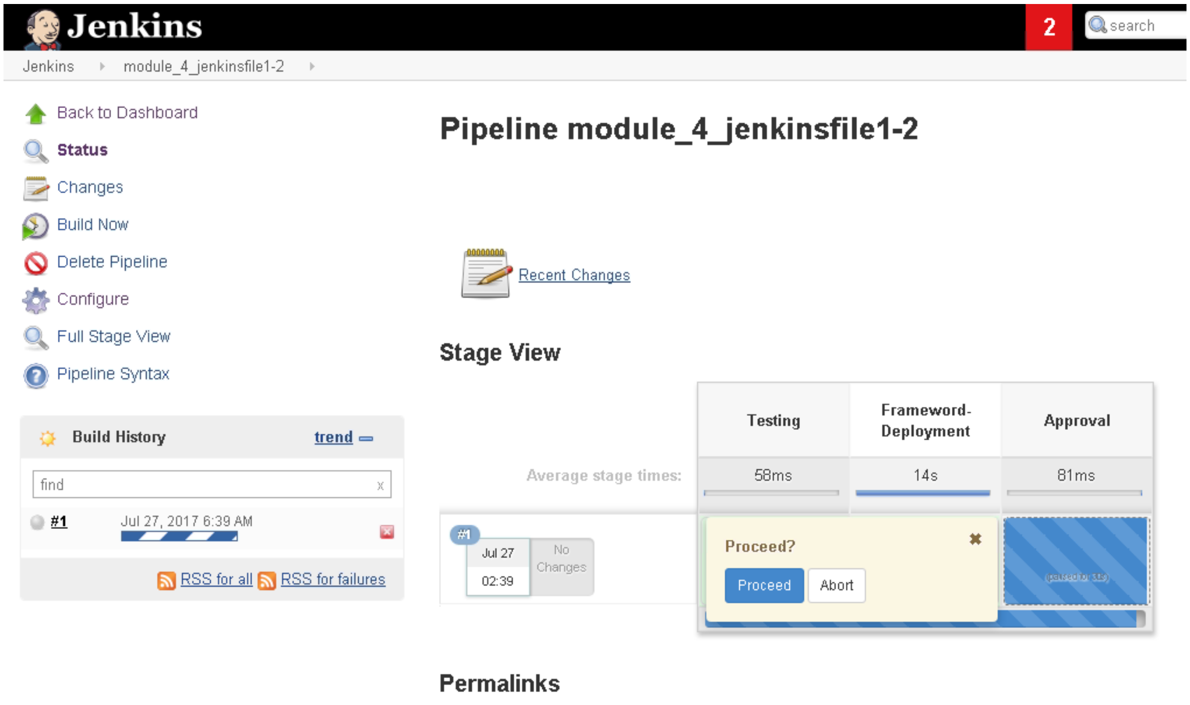
Pipeline module_4_jenkinsfile1-2

Recent Changes

Stage View

	Testing	Framework-Deployment	Approval
Average stage times:	58ms	14s	81ms
#1 Jul 27 02:39 No Changes	Proceed? Proceed Abort		

Permalinks



Jenkins 2 search

Jenkins > module_4_jenkinsfile1-2

- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Full Stage View
- Pipeline Syntax

Build History trend

find

#1 Jul 27, 2017 6:39 AM

RSS for all RSS for failures

Pipeline module_4_jenkinsfile1-2

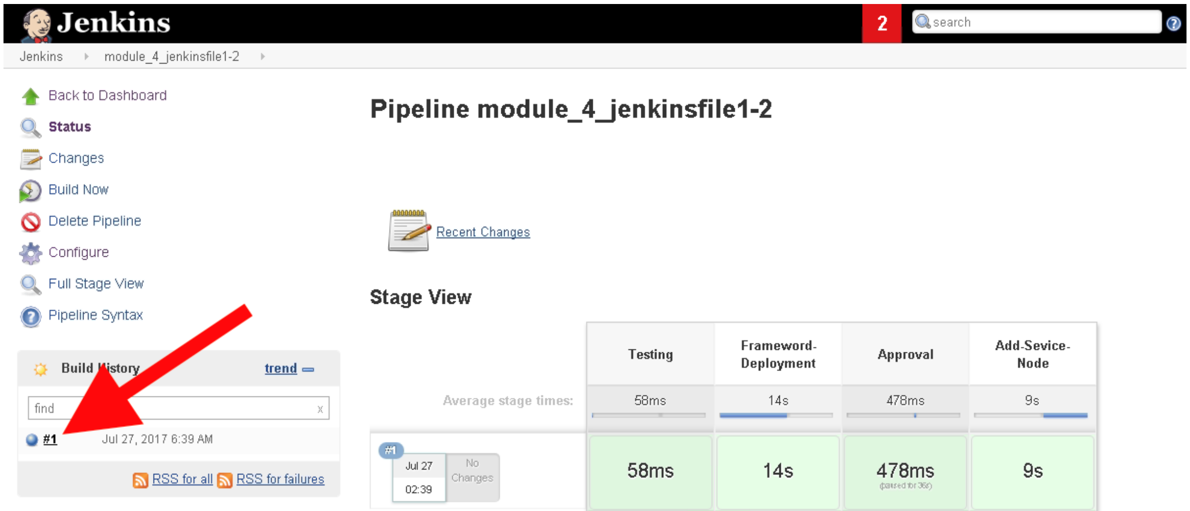
Recent Changes

Stage View

	Testing	Framework-Deployment	Approval
Average stage times:	58ms	14s	81ms
#1 Jul 27 02:39 No Changes	Proceed? Proceed Abort		

Permalinks

4. At the end of the Build event (success or failure) there is a console output from Jenkins. Select the blue globe on the left to see the outputs



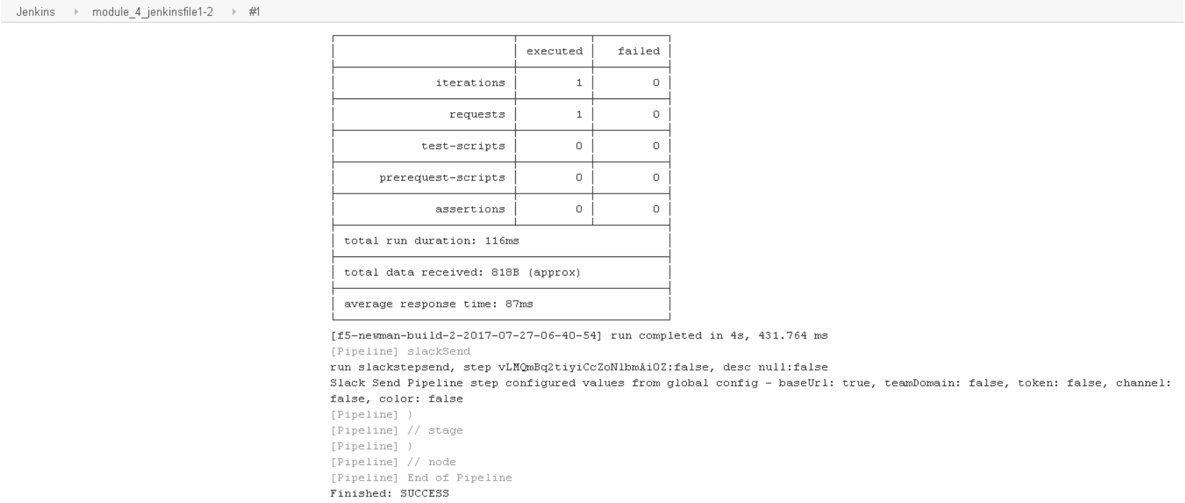
Pipeline module_4_jenkinsfile1-2

#1 Jul 27, 2017 6:39 AM

Stage View

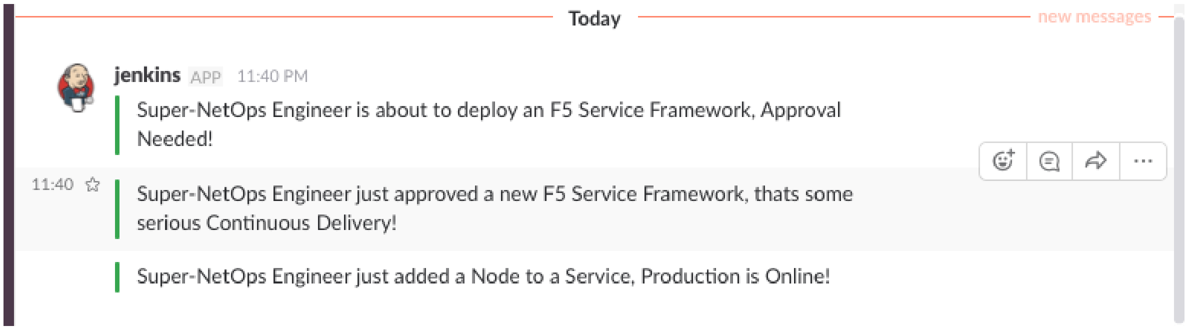
	Testing	Framework-Deployment	Approval	Add-Sevice-Node
Average stage times:	58ms	14s	478ms	9s
#1 Jul 27 02:39 No Changes	58ms	14s	478ms (paused for 36s)	9s

5. The Console Output file not only contains the Jenkins output from the Build, but also the f5-newman-wrapper toolkit logs for easy troubleshooting



```
┌─────────────────────────┬──────────┬──────────┐
│                         │ executed │   failed │
├─────────────────────────┼──────────┼──────────┤
│              iterations │        1 │        0 │
├─────────────────────────┼──────────┼──────────┤
│                requests │        1 │        0 │
├─────────────────────────┼──────────┼──────────┤
│            test-scripts │        0 │        0 │
├─────────────────────────┼──────────┼──────────┤
│      prerequest-scripts │        0 │        0 │
├─────────────────────────┼──────────┼──────────┤
│              assertions │        0 │        0 │
├─────────────────────────┴──────────┴──────────┤
│ total run duration: 116ms                     │
├───────────────────────────────────────────────┤
│ total data received: 818B (approx)            │
├───────────────────────────────────────────────┤
│ average response time: 87ms                   │
└───────────────────────────────────────────────┘
[f5-newman-build-2-2017-07-27-06-40-54] run completed in 4s, 431.764 ms
[Pipeline] slackSend
run slackstepsend, step vLMQmBq2tiyiCcZoNlbmAiOZ:false, desc null:false
Slack Send Pipeline step configured values from global config - baseUrl: true, teamDomain: false, token: false, channel:
false, color: false
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

6. Check Slack for the completion of everything!



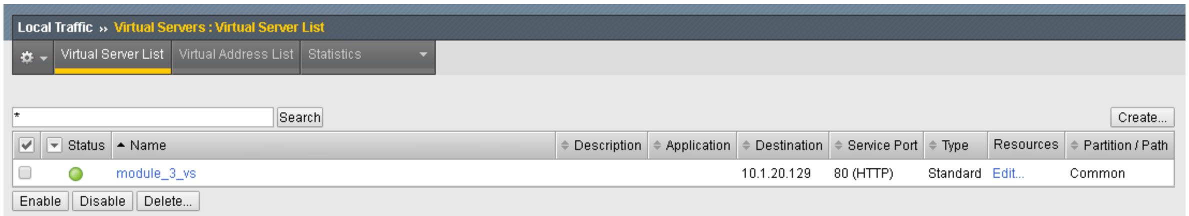
jenkins APP 11:40 PM

Super-NetOps Engineer is about to deploy an F5 Service Framework, Approval Needed!

11:40 Super-NetOps Engineer just approved a new F5 Service Framework, thats some serious Continuous Delivery!

Super-NetOps Engineer just added a Node to a Service, Production is Online!

7. Verify on the BIG-IP that the pool `module_3_vs` has been created and the services are Green

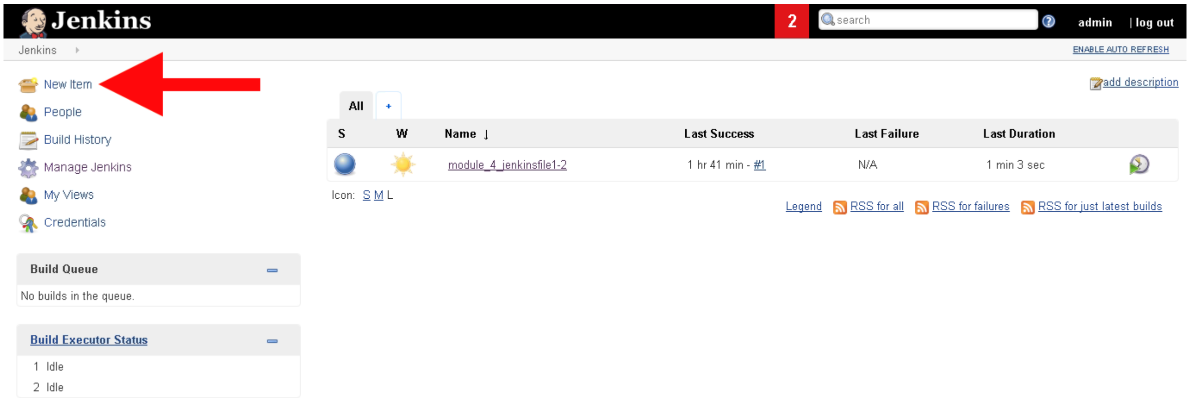


	Status	Name	Description	Application	Destination	Service Port	Type	Resources	Partition / Path
□	●	module_3_vs			10.1.20.129	80 (HTTP)	Standard	Edit...	Common

Task 2 - Jenkinsfile3 and Jenkinsfile4

These two Jenkins files were completed to show the ability of creating smaller deployments. In our case we will use the f5-newman-wrapper toolkit to again change the user selected state of a pool member. The different Pipelines notifications also have different Slack Color depictions, helping to quickly identify issues to team members.

1. Return to the Jenkins Dashboard and select `New Item`



S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	module_4_jenkinsfile1-2	1 hr 41 min - #1	N/A	1 min 3 sec

2. Repeat steps 2 & 3 of the last module, creating 2 new Jenkins jobs, one for each desired node state.

3. Create and Execute `module_4_jenkinsfile_3` for a down node

   Pipeline Job Name: `module_4_jenkinsfile_3`

```
node {
  stage('Testing') {
    //Run the tests
    //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-operation/f5-newman-build-3"
  }
  stage('Disable-Node') {
     //Run SNOPS Container Newman Package Virtual and Pool
    sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-operation/f5-newman-build-3"
    //chatops slack message that run has completed
    slackSend(
       channel: '#jenkins_builds',
       color: 'bad',
       message: 'Super-NetOps Engineer just disabled a Service Node!',
       teamDomain: 'f5agilitydevops',
       token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
       )
  }
}
```

4. Verify on the BIG-IP that the pool `module_3_pool` has a down node
5. Create and Execute `module_4_jenkinsfile_4` for an up node

   Pipeline Job Name: `module_4_jenkinsfile_4`

```
node {
  stage('Testing') {
     //Run the tests
     //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-operation/f5-newman-build-4"
  }
  stage('Enable-Node') {
      //Run SNOPS Container Newman Package Virtual and Pool
     sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-operation/f5-newman-build-4"
     //chatops slack message that run has completed
     slackSend(
        channel: '#jenkins_builds',
        color: 'good',
        message: 'Super-NetOps Engineer just enabled a Service Node!',
        teamDomain: 'f5agilitydevops',
        token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
        )
  }
}
```

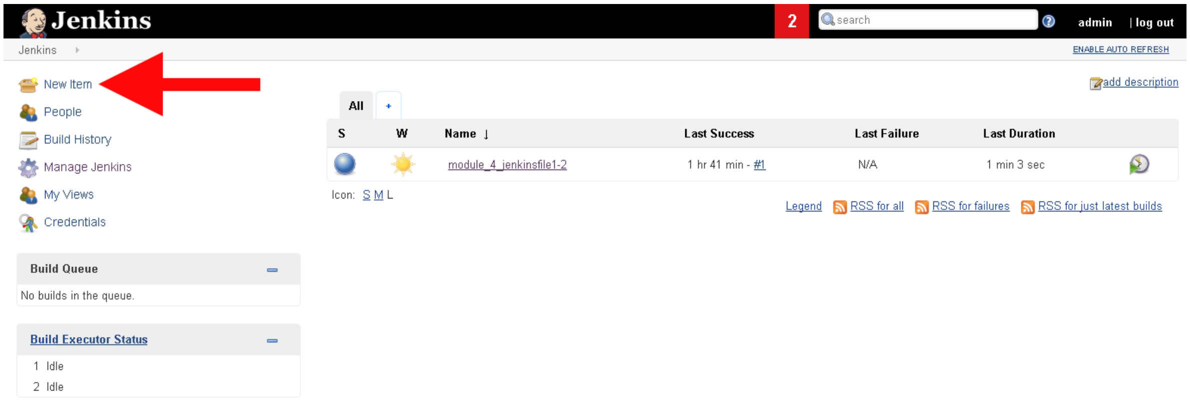
6. Verify on the BIG-IP that the pool `module_3_pool` has an up node

3.4.3 Lab 4.3 - Destroying a Service with Jenkins

For the last module we will teardown the Application Service we've been working with today. Destruction of an Application Services is an easy step often overlooked because of the perceived complexity it takes to reverse engineer a build. Utilizing Postman and the f5-newman-wrapper this is actually a very easy step, and will be incredibly valuable to teams seeking to strive for better Application Lifecycle Management. Because Postman and Newman operate in sequential order, simply reversing the order of creation will result in the correct teardown order. Also, since we're using the native F5 REST endpoints, all you need is a Resource with a DELETE method.

Task 1 - Teardown the Application via Jenkins

1. Return to the Jenkins Dashboard and select `New Item`



S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	module_4_jenkinsfile1-2	1 hr 41 min - #1	N/A	1 min 3 sec

2. Repeat steps 2 & 3 of the first module, creating the final Jenkins job
3. Create and Execute `module_4_jenkinsfile_5` to destroy our Application Service

   Pipeline Job Name: `module_4_jenkinsfile_5`

```
node {
   stage('Testing') {
      //Run the tests
      //sh "python -m /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-5"
   }
   stage('Removal-Notification') {
      //Run SNOPS Container Newman Package Delete Service
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'bad',
         message: 'Super-NetOps Engineer is about to remove an F5 Service!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
   stage('Approval') {
      //Gate the process and require approval
      input 'Delete?'
   }
   stage('Service-Delete') {
      //Run SNOPS Container Newman Package add Node to Pool
      sh "f5-newman-wrapper /home/snops/f5-automation-labs/jenkins/f5-newman-build/f5-newman-build-5"
      //chatops slack message that run has completed
      slackSend(
         channel: '#jenkins_builds',
         color: 'good',
         message: 'Super-NetOps Engineer removed an F5 Service successfully!',
         teamDomain: 'f5agilitydevops',
         token: 'vLMQmBq2tiyiCcZoNlbmAi0Z'
         )
   }
}
```

4. Verify the on the BIG-IP the service has been Deleted
5. Class 2 is Complete! if you have extra time, please give us feedback! https://www.surveymonkey.com/r/W2SZDYK

4

Class 3: Introduction to SecDevOps

This hands-on lab will demonstrate how to secure applications programmatically using a BIG-IP's iControl based REST API.

Leveraging programmability to deploy security policies and/or adhere to best practices during an application's lifecycle reduces the operational (e.g. time and money) cost of a defense in depth strategy. Programmatic workflows can be developed and deployed for specific security use cases, and integrated into the SDLC process, allowing for the protection of an application to iterate in parallel with the development of the application.

This course will feature the following topics.

- General interaction with tmm via BIG-IPs REST APIs
- Create, modify and assign an AFM policy
- Create, modify and assign an ASM policy

Lab Guide

This lab is divided into three modules. Each module of the lab, will require configuration of the BIG-IP, AFM, or ASM using the iControl REST based API. It is recommended that each lab be executed in order.

To perform the steps required in the lab, Postman will be used from the Windows jump box.

Prior to beginning the exercises, it is recommended to review the *Lab Topology*.

1. Module 1: Configuring BIG-IP
2. Module 2: Configuring AFM (Advanced Firewall Module)
3. Module 3: Configuring ASM (Application Security Module)

Support

Bugs and enhancements can be made by opening an issue within the GitHub repository.

Getting Started

Please follow the instructions provided by the instructor to start your lab and access your jump host.

Note: All work for this lab will be performed exclusively from the Windows jumphost. No software installation or interaction with your local system is required.

Expected time to complete: **3 hours**

4.1 Lab Topology

The network topology implemented for this lab is very simple. Since the focus of the lab is Control Plane programmability rather that Data Plane traffic flow we can keep the data plane fairly simple. The following components have been included in your lab environment:

- 1 x F5 BIG-IP (v13.0)
- 1 x Linux webserver (Ubuntu 16.04)
- 1 x Windows 7 jump box

The following table lists VLANS, IP Addresses and Credentials for all components:

Component	VLAN/IP Address(es)	Credentials
Windows Jump Box	• **Management:** 10.1.1.250 • **External:** 10.1.10.250	external_user/*available in instance details*
BIG-IP	• **Management:** 10.1.1.5 • **External:** 10.1.10.5 • **Internal:** 10.1.20.5	admin/admin
Linux Server	• **Management:** 10.1.1.10 • **Internal:** 10.1.20.10	ubuntu/ubuntu

4.2 Module 1: iControl REST API Refresher

- Explore the iControl REST API on a BIG-IP
- Use Postman to interact with the iControl REST API
- Authenticate to the BIG-IP using a username/password and token
- Modify the authentication token timeout
- Build a basic LTM configuration

4.2.1 Lab 1.1: Exploring iControl

The iControl REST API available via TMOS can be directly accessed and endpoints explored.

1. Open Google Chrome and navigate to the following bookmarks: BIG-IP A GUI and BIG-IP API ToC. Accept any SSL warnings/errors that appear and ensure that you can access both login prompts.
2. Click on the **BIG-IP API ToC** bookmark to access the API Table of Contents for BIG-IP A. The `/mgmt/toc` path in the URL is available on all TMOS versions 11.6 or newer.
3. Authenticate using the default **admin/admin** credentials.
4. After successfully authenticating, you will be presented with a top-level list of REST resources available on the BIG-IP. At the top of the page is a search box that can be used to search for specific REST resources.

4.2.2 Lab 1.2: API Authentication

This lab utilizes the Postman Chrome extension to facilitate the sending data to and receiving data from the iControl REST API.

REST API Authentication

One of the many basic concepts related to interaction with REST API's is how a particular consumer is authenticated to the system. BIG-IP, BIG-IQ and iWorkflow support two types of authentication: HTTP BASIC and Token based. It's important to understand both of these authentication mechanisms, as consumers of the API will often make use of both types depending on the use case. This lab will demonstrate how to interact with both types of authentication.

Task 1 - Basic Authentication

> **Warning:** Prior to performing any of the below steps, ensure that you can log into the BIG-IP with Chrome after accepting the invalid certificate. Postman relies on the Chrome certificate store and if the self-signed cert has not been accepted via Chrome, this extension will not work properly.

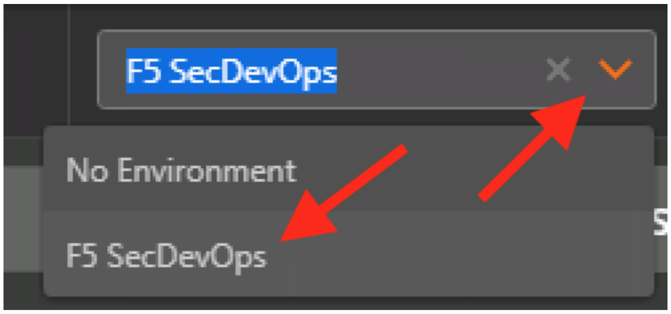
In this task we will use the Postman tool to send API requests using HTTP BASIC authentication. As its name implies this method of authentication encodes the user credentials via the existing BASIC authentication method provided by the HTTP protocol. The mechanism this method uses is to insert an HTTP header named 'Authorization' with a value that is built by Base 64 encoding the string "<username>:<password>". The resulting header takes this form:

```
Authorization: Basic YWRtaW46YWRtaW4=
```

It should be noted that cracking the method of authentication is TRIVIAL; as a result API calls should always be performed using HTTPS (F5 default) rather than HTTP.

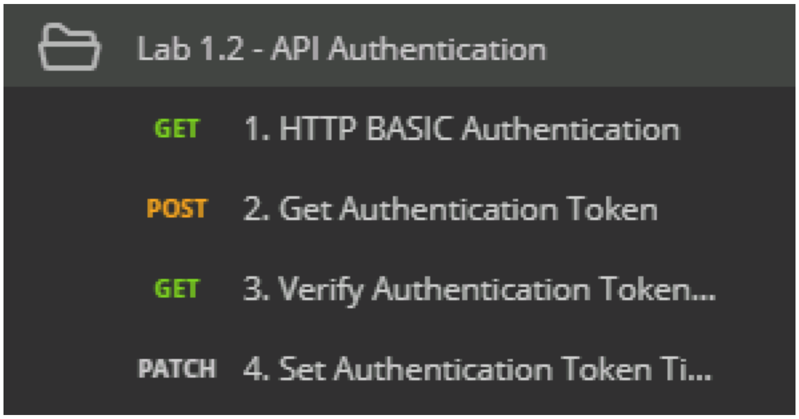
Perform the following steps to complete this task:

1. Open the Postman Client on your jumphost by clicking the icon.
2. To assist in multi-step procedures we make heavy use of the 'Environments' capability in Postman. This capability allows us to set various global variables that are then substituted into a request before it's sent. When you open Postman please verify that your environment is set the **F5 SecDevOps** environment:



3. Click the ‘Collections’ tab on the left side of the screen, expand the ‘F5 SecDevOps’ collection on the left side of the screen, expand the **Lab 1.2 - API Authentication** folder:

(Ignore the # of requests on the screen below versus what you might see, the # of requests will grow and change as this lab grows)



Lab 1.2 - API Authentication

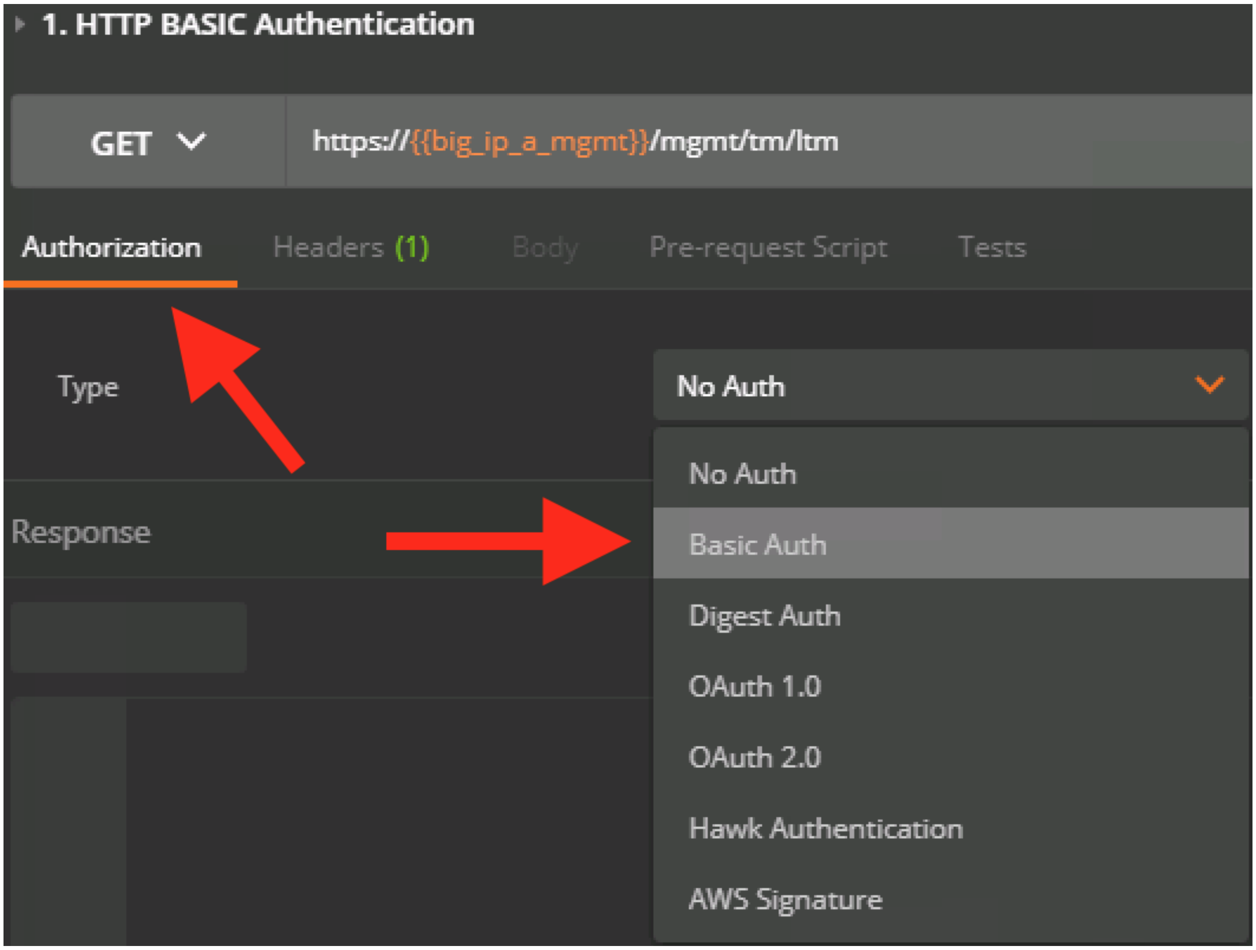
GET 1. HTTP BASIC Authentication

POST 2. Get Authentication Token

GET 3. Verify Authentication Token...

PATCH 4. Set Authentication Token Ti...

4. Click the **1. HTTP BASIC Authentication** item. Click the ‘Authorization’ tab and select ‘Basic Auth’ as the Type. Fill in the username and password (admin/admin) and click the ‘Update Request’ button. Notice that the number of Headers in the Headers tab changed from 1 to 2. This is because Postman automatically created the HTTP header and updated your request to include it. Click the ‘Headers’ tab and examine the HTTP header:



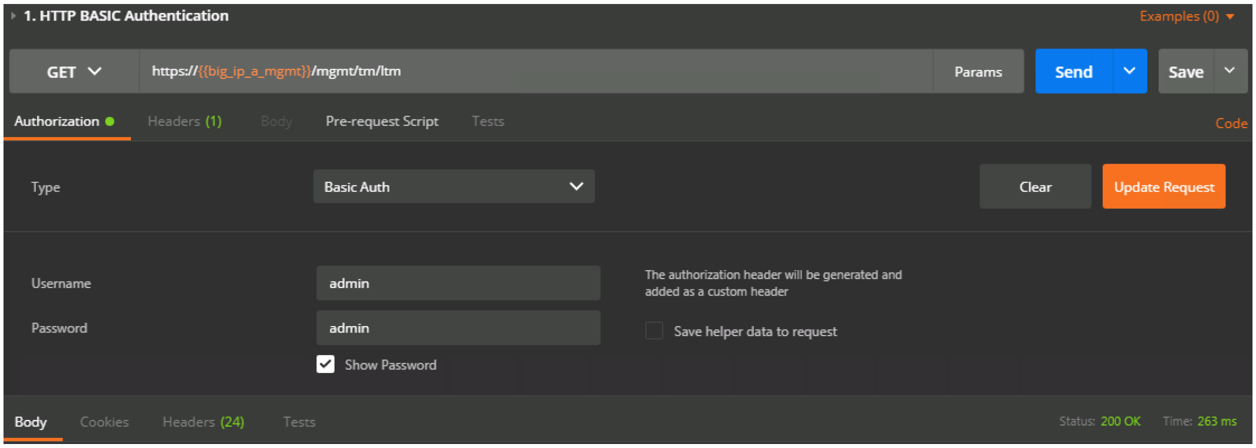
1. HTTP BASIC Authentication

GET https://{{big_ip_a_mgmt}}/mgmt/tm/ltm

Type: No Auth

5. Click the ‘Send’ button to send the request. If the request succeeds you should be presented with a listing of the ‘/mgmt/tm/ltm’ Organizing Collection.

Tip: Pay attention to the Status response i.e. **200 OK**



1. HTTP BASIC Authentication

GET https://{{big_ip_a_mgmt}}/mgmt/tm/ltm

Type: Basic Auth

Username: admin

Password: admin

☑ Show Password

The authorization header will be generated and added as a custom header

□ Save helper data to request

Status: 200 OK Time: 263 ms

Task 2 - Token Based Authentication

One of the disadvantages of BASIC Authentication is that credentials are sent with each and every request. This can result in a much greater attack surface being exposed unnecessarily. As a result Token Based Authentication (TBA) is preferred in many cases. This method only sends the credentials once, on the first request. The system then responds with a unique token for that session and the consumer then uses that token for all subsequent requests. BIG-IP, BIG-IQ and iWorkflow support token-based authentication that drops down to the underlying authentication subsystems available in TMOS. As a result the system can be configured to support external authentication providers (RADIUS, TACACS, AD, etc) and those authentication methods can flow through to the REST API. In this task we will demonstrate TBA using the local authentication database, however, authentication to external providers is fully supported.

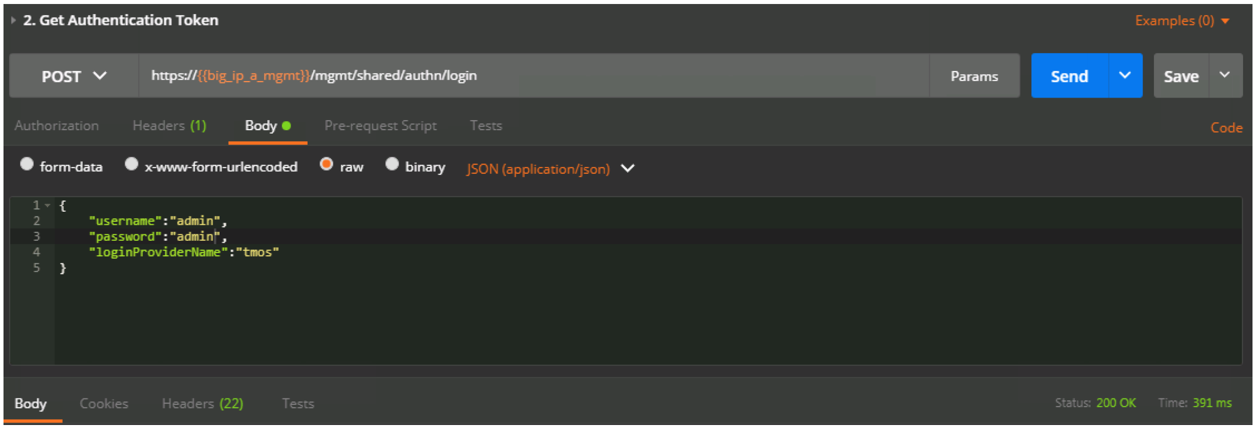
Tip: For more information about external authentication providers see the section titled “About external authentication providers with iControl REST” in the iControl REST API User Guide available at https://devcentral.f5.com

---

Perform the following steps to complete this task:

1. Click the **2: Get Authentication Token** item in the **Lab 1.2 - API Authentication** Postman Collection
2. Notice that we send a POST request to the ‘/mgmt/shared/authn/login’ endpoint. Note that BASIC authentication is NOT required for this step. The token is provided based on the credentials located within the JSON payload.
3. Click the ‘Body’ tab and examine the JSON that we will send to BIG-IP to provide credentials and the authentication provider:
4. Modify the JSON body and add the required credentials (admin/admin). Then click the ‘Send’ button.
5. Examine the response status code. If authentication succeeded and a token was generated, the response will have a 200 OK status code. If the status code is 401 then check your credentials. View the response body to see the token that was provided:

Successful:

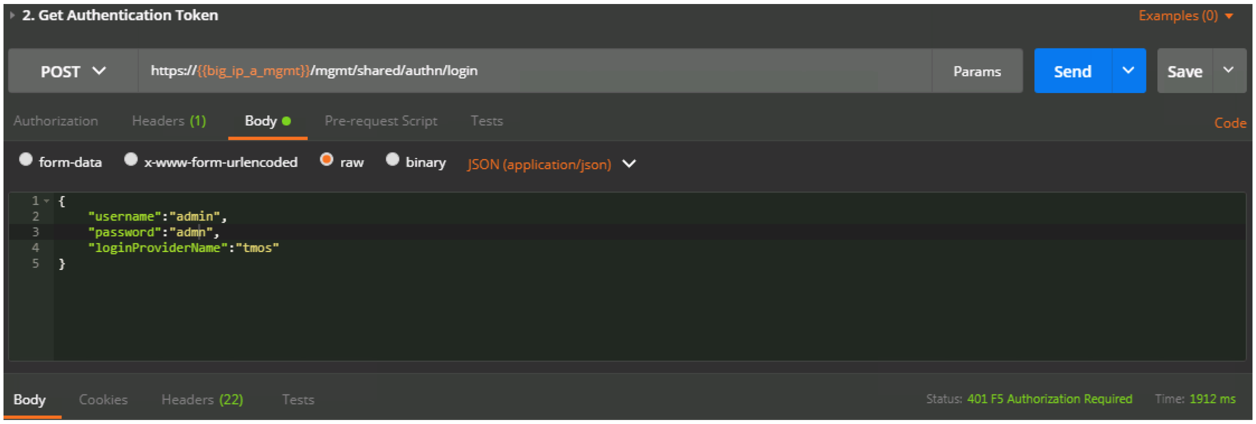


POST https://{{big_ip_a_mgmt}}/mgmt/shared/authn/login

```
{
    "username":"admin",
    "password":"admin",
    "loginProviderName":"tmos"
}
```

Status: 200 OK Time: 391 ms

Unsuccessful:

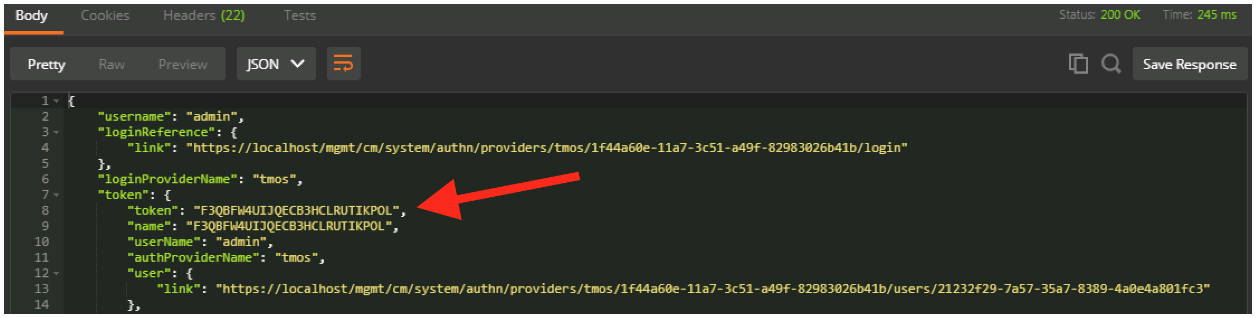


POST https://{{big_ip_a_mgmt}}/mgmt/shared/authn/login

```
{
    "username":"admin",
    "password":"admn",
    "loginProviderName":"tmos"
}
```

Status: 401 F5 Authorization Required Time: 1912 ms

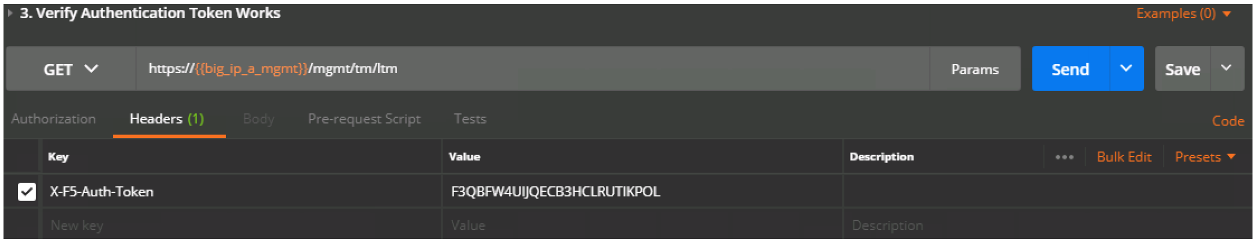
6. Once you receive a 200 OK status code examine the response body. The various attributes show the parameters assigned to the particular token. Find the 'token' attribute and copy it into your clipboard (Ctrl+c) for use in the next step:



Status: 200 OK Time: 245 ms

```
{
    "username": "admin",
    "loginReference": {
        "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/login"
    },
    "loginProviderName": "tmos",
    "token": {
        "token": "F3QBFW4UIJQECB3HCLRUTIKPOL",
        "name": "F3QBFW4UIJQECB3HCLRUTIKPOL",
        "userName": "admin",
        "authProviderName": "tmos",
        "user": {
            "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/users/21232f29-7a57-35a7-8389-4a0e4a801fc3"
        },
```

7. Click the **3: Verify Authentication Works** item in the **Lab 1.2 - API Authentication** Postman collection. Click the 'Headers' tab and paste the token value copied above as the VALUE for the 'X-F5-Auth-Token' header. This header is required to be sent on all requests when using token based authentication.



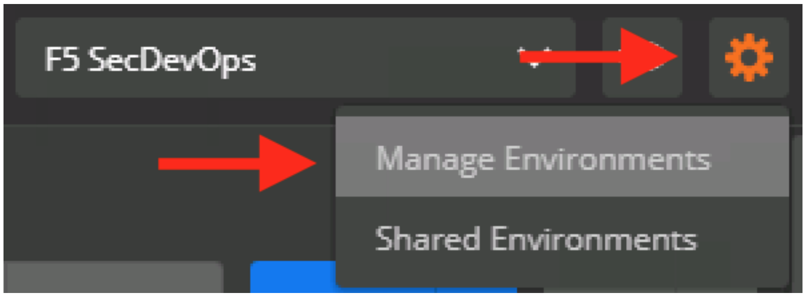
3. Verify Authentication Token Works

GET https://{{big_ip_a_mgmt}}/mgmt/tm/ltm

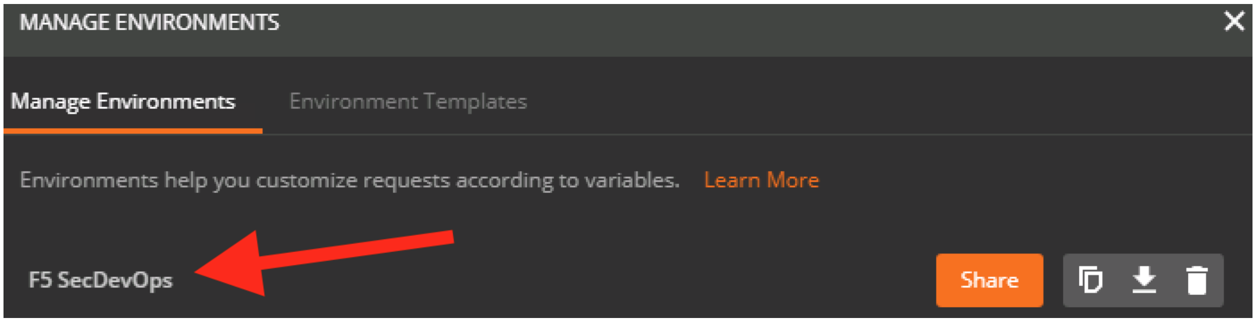
	Key	Value	Description
☑	X-F5-Auth-Token	F3QBFW4UIJQECB3HCLRUTIKPOL	
	New key	Value	Description

8. Click the 'Send' button. If you're request is successful you should see a '200 OK' status and a listing of the 'ltm' Organizing Collection.

9. We will now update your Postman environment to use this auth token for the remainder of the lab. Click the Environment menu in the top right of the Postman window and click 'Manage Environments':



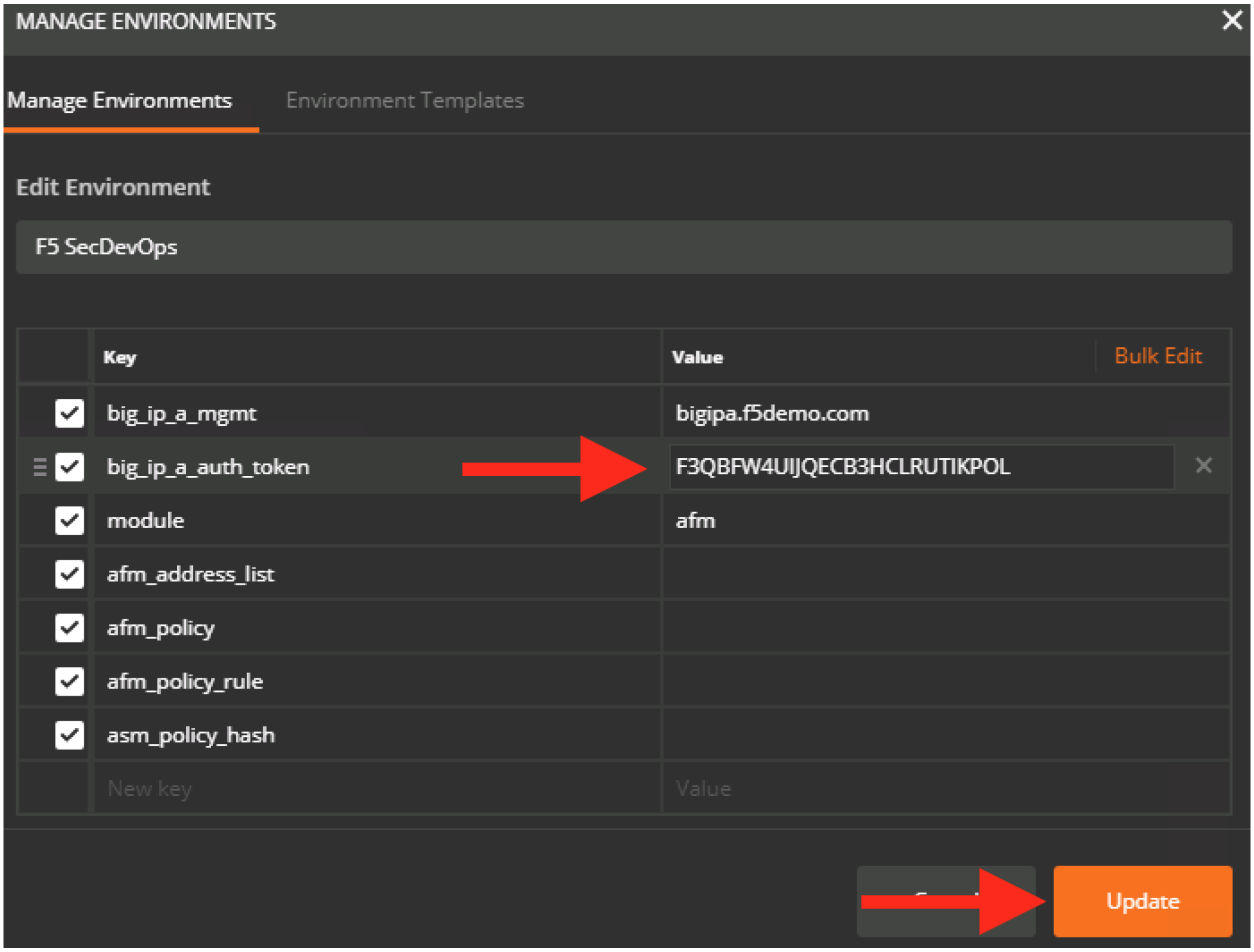
10. Click the **F5 SecDevOps** item:



Environments help you customize requests according to variables.

11. Update the value for 'big_ip_a_auth_token' by Pasting (Ctrl-v) in your auth token:

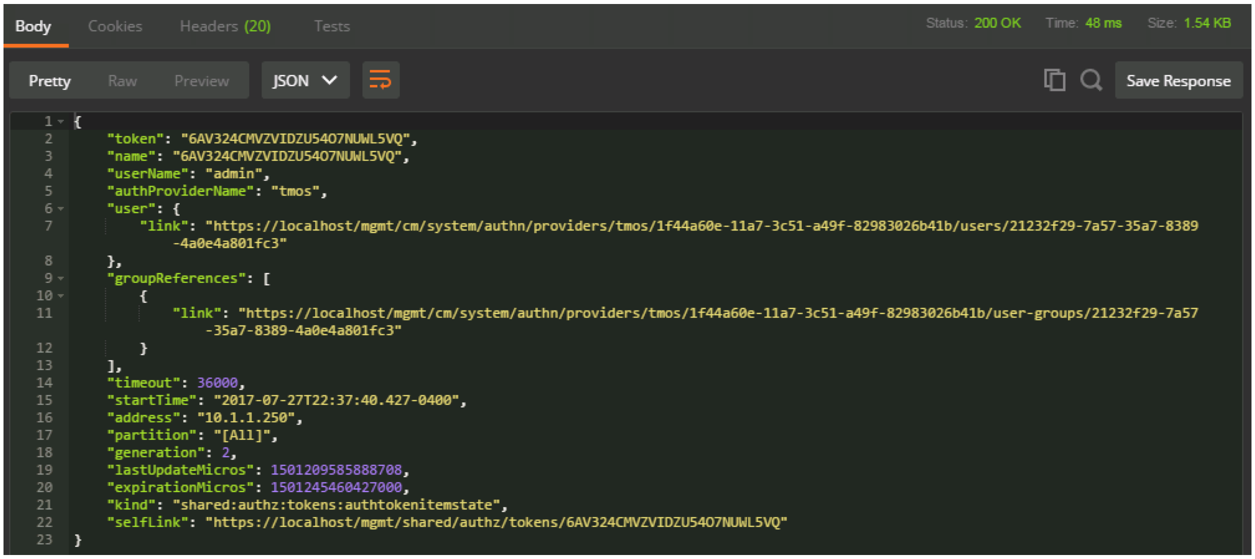
12. Click the 'Update' button and then close the 'Manage Environments' window. You're subsequent requests will now automatically substitue the token's value where the **{{big_ip_a_auth_token}}** environmental variable is used.



F5 SecDevOps

	Key	Value
☑	big_ip_a_mgmt	bigipa.f5demo.com
☑	big_ip_a_auth_token	F3QBFW4UIJQECB3HCLRUTIKPOL
☑	module	afm
☑	afm_address_list	
☑	afm_policy	
☑	afm_policy_rule	
☑	asm_policy_hash	
	New key	Value

13. Click the **4: Set Authentication Token Timeout** item in the **Lab 1.2 - API Authentication** Postman collection. This request will PATCH your token Resource (check the URI) and update the timeout attribute so we can complete the lab easily. Examine the request type and JSON Body and then click the 'Send' button. Verify that the timeout has been changed to '36000' in the response:



Status: 200 OK Time: 48 ms Size: 1.54 KB

```
{
    "token": "6AV324CMVZVIDZU54O7NUWL5VQ",
    "name": "6AV324CMVZVIDZU54O7NUWL5VQ",
    "userName": "admin",
    "authProviderName": "tmos",
    "user": {
        "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/users/21232f29-7a57-35a7-8389
            -4a0e4a801fc3"
    },
    "groupReferences": [
        {
            "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/user-groups/21232f29-7a57
                -35a7-8389-4a0e4a801fc3"
        }
    ],
    "timeout": 36000,
    "startTime": "2017-07-27T22:37:40.427-0400",
    "address": "10.1.1.250",
    "partition": "[All]",
    "generation": 2,
    "lastUpdateMicros": 1501209585888708,
    "expirationMicros": 1501245460427000,
    "kind": "shared:authz:tokens:authtokenitemstate",
    "selfLink": "https://localhost/mgmt/shared/authz/tokens/6AV324CMVZVIDZU54O7NUWL5VQ"
}
```

4.2.3 Lab 1.3: Building a Basic LTM Config

Overview

In this lab, the iControl REST API will be used to build a basic monitor, node, pool, and virtual server configuration on the BIG-IP.

Specific Instructions

Prior to performing the below steps, validate that the Hackazon web site is not accessible via the Windows jump box by clicking on the Hackazon bookmark in the Chrome toolbar.

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

Attention: Some response content has been removed for brevity.

1. Create an HTTP Monitor

An HTTP POST to the `/mgmt/tm/ltm/monitor/http` endpoint with a body containing the monitor configuration creates a monitor.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/monitor/http
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"hackazon_monitor",
    "send":"GET /\r\n"
}
```

Example Response

```
{
    "kind": "tm:ltm:monitor:http:httpstate",
    "name": "hackazon_monitor",
    "partition": "Common",
    "fullPath": "/Common/hackazon_monitor",
    "generation": 0,
    "selfLink": "https://localhost/mgmt/tm/ltm/monitor/http/~Common~hackazon_monitor?
↪ver=13.0.0",
    "adaptive": "disabled",
    "adaptiveDivergenceType": "relative",
    "adaptiveDivergenceValue": 25,
    "adaptiveLimit": 200,
    "adaptiveSamplingTimespan": 300,
    "defaultsFrom": "/Common/http",
    "destination": "*:*",
    "interval": 5,
    "ipDscp": 0,
    "manualResume": "disabled",
    "reverse": "disabled",
    "send": "GET / HTTP/\r\n",
    "timeUntilUp": 0,
    "timeout": 16,
    "transparent": "disabled",
    "upInterval": 0
}
```

2. Create a Pool

An HTTP POST to the `/mgmt/tm/ltm/pool` endpoint with a body containing the configuration creates a pool with a node(s).

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/pool
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"hackazon_pool",
    "monitor":"/Common/hackazon_monitor",
    "members": ["10.1.20.10:80"]
}
```

Example Response

```
{
    "kind": "tm:ltm:pool:poolstate",
    "name": "hackazon_pool",
    "partition": "Common",
    "fullPath": "/Common/hackazon_pool",
    "generation": 10781,
    "selfLink": "https://localhost/mgmt/tm/ltm/pool/~Common~hackazon_pool?ver=13.0.0",
    "allowNat": "yes",
    "allowSnat": "yes",
    "ignorePersistedWeight": "disabled",
    "ipTosToClient": "pass-through",
    "ipTosToServer": "pass-through",
    "linkQosToClient": "pass-through",
    "linkQosToServer": "pass-through",
    "loadBalancingMode": "round-robin",
    "minActiveMembers": 0,
    "minUpMembers": 0,
    "minUpMembersAction": "failover",
    "minUpMembersChecking": "disabled",
    "monitor": "/Common/hackazon_monitor ",
    "queueDepthLimit": 0,
    "queueOnConnectionLimit": "disabled",
    "queueTimeLimit": 0,
    "reselectTries": 0,
    "serviceDownAction": "none",
    "slowRampTime": 10,
    "membersReference": {
        "link": "https://localhost/mgmt/tm/ltm/pool/~Common~hackazon_pool/members?
↪ver=13.0.0",
        "isSubcollection": true
    }
}
```

3. Create a HTTP Profile

An HTTP POST to the `/mgmt/tm/ltm/profile/http` endpoint with a body containing the configuration creates a profile.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/profile/http
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"hackazon_http_profile",
    "insertXforwardedFor":"enabled",
    "serverAgentName":"hackazon"
}
```

Example Response

```
{
    "kind": "tm:ltm:profile:http:httpstate",
    "name": "hackazon_http_profile",
    "partition": "Common",
    "fullPath": "/Common/hackazon_http_profile",
    "generation": 10783,
    "selfLink": "https://localhost/mgmt/tm/ltm/profile/http/~Common~hackazon_http_
↪profile?ver=13.0.0",
    "acceptXff": "disabled",
    "appService": "none",
    "basicAuthRealm": "none",
    "defaultsFrom": "/Common/http",
    "defaultsFromReference": {
        "link": "https://localhost/mgmt/tm/ltm/profile/http/~Common~http?ver=13.0.0"
    },
    "description": "none",
    "encryptCookies": [],
    "insertXforwardedFor": "enabled",
    "serverAgentName": "hackazon"
}
```

4. Create a TCP profile

An HTTP POST to the `/mgmt/tm/ltm/profile/tcp` endpoint with a body containing the configuration creates a TCP profile.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/profile/tcp
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"hackazon_tcp_clientside_profile",
    "nagle":"disabled",
    "sendBufferSize":"16000"
}
```

Example Response

```
{
    "kind": "tm:ltm:profile:tcp:tcpstate",
    "name": "hackazon_tcp_clientside_profile",
    "partition": "Common",
    "fullPath": "/Common/hackazon_tcp_clientside_profile",
    "generation": 10784,
    "selfLink": "https://localhost/mgmt/tm/ltm/profile/tcp/~Common~hackazon_tcp_
↪clientside_profile?ver=13.0.0",
    "abc": "enabled",
    "ackOnPush": "enabled",
    "appService": "none",
    "autoProxyBufferSize": "disabled",
    "autoReceiveWindowSize": "disabled",
    "autoSendBufferSize": "disabled",
    "closeWaitTimeout": 5,
    "cmetricsCache": "enabled",
    "cmetricsCacheTimeout": 0,
    "congestionControl": "high-speed",
    "defaultsFrom": "/Common/tcp",
    "defaultsFromReference": {
        "link": "https://localhost/mgmt/tm/ltm/profile/tcp/~Common~tcp?ver=13.0.0"
    },
    "keepAliveInterval": 1800,
    "nagle": "disabled",
    "sendBufferSize": 16000
}
```

5. Create a Virtual Server

An HTTP POST to the `/mgmt/tm/ltm/virtual` endpoint with a body containing the configuration creates a virtual server.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/virtual
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"hackazon_vs",
    "destination":"10.1.10.10:80",
    "ipProtocol":"tcp",
    "pool":"hackazon_pool",
    "sourceAddressTranslation": { "type":"automap" },
    "profiles": [
        { "name":"/Common/hackazon_tcp_clientside_profile", "context":"clientside" },
        { "name":"/Common/tcp-wan-optimized", "context":"serverside" },
        "/Common/hackazon_http_profile"
    ]
}
```

Example Response

Note: The profiles for this virtual server is a subcollection. This collection can be access by performing a GET on the profiles endpoint for this specific virtual server `https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/virtual/~Common~hackazon_vs/profiles`.

```
{
    "kind": "tm:ltm:virtual:virtualstate",
    "name": "hackazon_vs",
    "partition": "Common",
    "fullPath": "/Common/hackazon_vs",
    "generation": 10785,
    "selfLink": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs?ver=13.0.0
↪",
    "addressStatus": "yes",
    "autoLasthop": "default",
    "cmpEnabled": "yes",
    "connectionLimit": 0,
    "destination": "/Common/10.1.10.20:80",
    "enabled": true,
    "gtmScore": 0,
    "ipProtocol": "tcp",
    "mask": "255.255.255.255",
    "mirror": "disabled",
    "mobileAppTunnel": "disabled",
    "nat64": "disabled",
    "pool": "/Common/hackazon_pool",
    "poolReference": {
        "link": "https://localhost/mgmt/tm/ltm/pool/~Common~hackazon_pool?ver=13.0.0"
    },
    "rateLimit": "disabled",
    "rateLimitDstMask": 0,
    "rateLimitMode": "object",
    "rateLimitSrcMask": 0,
    "serviceDownImmediateAction": "none",
    "source": "0.0.0.0/0",
    "sourceAddressTranslation": {
        "type": "automap"
    },
    "sourcePort": "preserve",
    "synCookieStatus": "not-activated",
    "translateAddress": "enabled",
    "translatePort": "enabled",
    "vlansDisabled": true,
    "vsIndex": 9,
    "policiesReference": {
        "link": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs/policies?
↪ver=13.0.0",
        "isSubcollection": true
    },
    "profilesReference": {
        "link": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs/profiles?
↪ver=13.0.0",
        "isSubcollection": true
    }
}
```

6. Retrieve VS information

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/ltm/virtual/~Common~hackazon_vs/
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:ltm:virtual:virtualstate",
    "name": "hackazon_vs",
    "partition": "Common",
    "fullPath": "/Common/hackazon_vs",
    "generation": 10785,
    "selfLink": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs?ver=13.0.0
↪",
    "addressStatus": "yes",
    "autoLasthop": "default",
    "cmpEnabled": "yes",
    "connectionLimit": 0,
    "destination": "/Common/10.1.10.20:80",
    "enabled": true,
    "gtmScore": 0,
    "ipProtocol": "tcp",
    "mask": "255.255.255.255",
    "mirror": "disabled",
    "mobileAppTunnel": "disabled",
    "nat64": "disabled",
    "pool": "/Common/hackazon_pool",
    "poolReference": {
        "link": "https://localhost/mgmt/tm/ltm/pool/~Common~hackazon_pool?ver=13.0.0"
    },
    "rateLimit": "disabled",
    "rateLimitDstMask": 0,
    "rateLimitMode": "object",
    "rateLimitSrcMask": 0,
    "serviceDownImmediateAction": "none",
    "source": "0.0.0.0/0",
    "sourceAddressTranslation": {
        "type": "automap"
    },
    "sourcePort": "preserve",
    "synCookieStatus": "not-activated",
    "translateAddress": "enabled",
    "translatePort": "enabled",
    "vlansDisabled": true,
    "vsIndex": 9,
    "policiesReference": {
        "link": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs/policies?
↪ver=13.0.0",
        "isSubcollection": true
    },
    "profilesReference": {
        "link": "https://localhost/mgmt/tm/ltm/virtual/~Common~hackazon_vs/profiles?
↪ver=13.0.0",
        "isSubcollection": true
    }
}
```

7. Validate the virtual server

Click on the Hackazon bookmark in the Chrome toolbar and validate that the Hackazon web site is now accessible.

4.3 Module 2: Programmatic Control of Firewall Services

- Provision AFM module on BIG-IP
- Interact with AFM related REST endpoints on a BIG-IP
- Create and modify an AFM address list
- Create and modify an AFM policy

4.3.1 Lab 2.1: Provisioning AFM

Overview

In this lab, the iControl REST API will be used to provision a module on the BIG-IP. More specifically, the Advanced Firewall Manager (AFM) module will be provisioned for use in **Module 2: Configuring AFM (Advanced Firewall Module)**.

Specific Instructions

Prior to performing the below steps, validate the **{{module}}** Postman environment variable. The **{{module}}** should be set to **afm**.

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

> **Attention:** Some response content has been removed for brevity.

1. Retrieve all module provision states

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

> **Note:** The **afm** module is currently provisioned for **none** while the **ltm** module is provisioned for **nominal**.

```
{
    "kind": "tm:sys:provision:provisioncollectionstate",
    "selfLink": "https://localhost/mgmt/tm/sys/provision?ver=13.0.0",
    "items": [
        {
            "kind": "tm:sys:provision:provisionstate",
            "name": "afm",
            "fullPath": "afm",
            "generation": 5609,
            "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
            "cpuRatio": 0,
            "diskRatio": 0,
            "level": "none",
            "memoryRatio": 0
        },
        {
            "kind": "tm:sys:provision:provisionstate",
            "name": "ltm",
            "fullPath": "ltm",
            "generation": 1,
            "selfLink": "https://localhost/mgmt/tm/sys/provision/ltm?ver=13.0.0",
            "cpuRatio": 0,
            "diskRatio": 0,
            "level": "nominal",
            "memoryRatio": 0
        }
    ]
}
```

2. Retrieve single module provision state

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: The **afm** module should still be provisioned after performing the steps in Lab 1.

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "afm",
    "fullPath": "afm",
    "generation": 5609,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "none",
    "memoryRatio": 0
}
```

3.1. Provision module (OPTIONAL)

Warning: This step is optional and should only be performed if **afm** is not provisioned.

The **afm** module is provisioned using an HTTP PATCH with a body containing a provisioning level to the REST endpoint for `mgmt/tm/sys/provision/{{module}}`.

Note: Performing a provision/deprovion operation takes some time to complete. If the original request is still being processed, the below error may be encountered.

```
{
    "code": 400,
    "message": "01071003:3: A previous provisioning operation is in progress. Try␣
↪again when the BIGIP is active.",
    "errorStack": [],
    "apiError": 3
}
```

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "level":"nominal"
}
```

Example Response

Note: The **afm** module has been provisioned with a **level** of **nominal**.

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "afm",
    "fullPath": "afm",
    "generation": 10636,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "nominal",
    "memoryRatio": 0
}
```

3.2. Deprovision module

This request will serve as an example of how to deprovision a BIG-IP module.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "level":"none"
}
```

Example Response

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "afm",
    "fullPath": "afm",
    "generation": 10714,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "none",
    "memoryRatio": 0
}
```

3.3. Re-provision module

Re-provision the **afm** module if previously deprovisioned.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "level":"nominal"
}
```

Example Response

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "afm",
    "fullPath": "afm",
    "generation": 10636,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "nominal",
    "memoryRatio": 0
}
```

4.3.2 Lab 2.2: Create AFM Address List

Overview

In this lab, the iControl REST based API will be used to create an address list that will be used with an AFM policy in a later lab.

Specific Instructions

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

Attention: Some response content has been removed for brevity.

1. List Firewall Policies

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: A test policy has already been created on the BIG-IP for demonstration purposes.

```
{
    "kind": "tm:security:firewall:policy:policycollectionstate",
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy?ver=13.0.0",
    "items": [
        {
            "kind": "tm:security:firewall:policy:policystate",
            "name": "block_all",
            "partition": "Common",
            "fullPath": "/Common/block_all",
            "generation": 5789,
            "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~
↪block_all?ver=13.0.0",
            "rulesReference": {
                "link": "https://localhost/mgmt/tm/security/firewall/policy/~Common~
↪block_all/rules?ver=13.0.0",
                "isSubcollection": true
            }
        }
    ]
}
```

2. List all Firewall Address Lists

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/address-list
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: A test address list has already been created on the BIG-IP for demonstration purposes.

```
{
    "kind": "tm:security:firewall:address-list:address-listcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/security/firewall/address-list?ver=13.0.0",
    "items": [
        {
            "kind": "tm:security:firewall:address-list:address-liststate",
            "name": "test_address_list",
            "partition": "Common",
            "fullPath": "/Common/test_address_list",
            "generation": 6326,
            "selfLink": "https://localhost/mgmt/tm/security/firewall/address-list/~
↪Common~test_address_list?ver=13.0.0",
            "addresses": [
                {
                    "name": "1.1.1.1"
                }
            ]
        }
    ]
}
```

3. Create an Address List

An HTTP POST to the `/mgmt/tm/security/firewall/address-list/` endpoint with a body containing the configuration creates an address list that can be used with a firewall policy.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/address-list/
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name": "google-dns_address_list",
    "addresses": [
        {
            "name": "8.8.4.4"
        }
    ]
}
```

Example Response

Note: Copy the **name** of the address list, highlighted below, from the response into the **afm_address_list** Postman environment variable.

```
{
    "kind": "tm:security:firewall:address-list:address-liststate",
    "name": "google-dns_address_list",
    "partition": "Common",
    "fullPath": "/Common/google-dns_address_list",
    "generation": 11436,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/address-list/~Common~
↪google-dns_address_list?ver=13.0.0",
    "addresses": [
        {
            "name": "8.8.4.4"
        }
    ]
}
```

4. List Single Firewall Address List

Note: Ensure that the **afm_address_list** Postman environment variable has been populated with the name of the address list.

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/address-list/{{afm_address_
↪list}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:security:firewall:address-list:address-liststate",
    "name": "google-dns_address_list",
    "partition": "Common",
    "fullPath": "/Common/google-dns_address_list",
    "generation": 11436,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/address-list/~Common~
↪google-dns_address_list?ver=13.0.0",
    "addresses": [
        {
            "name": "8.8.4.4"
        }
    ]
}
```

5. Update Firewall Address List

An HTTP PATCH to the `/mgmt/tm/security/firewall/address-list/{{afm_address_list}}` endpoint with a body containing all addresses that should exist in the address list will update this collection.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/address-list/{{afm_address_
↪list}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Note: Include the contents of the address list plus the new address(es) to ensure that the contents are not overwritten.

Body

Warning: When patching an address list, be sure to include all addresses (e.g. existing and new) to ensure that the list does not get overwritten.

```
{
    "addresses": [
        {
            "name": "8.8.4.4"
        },
        {
            "name": "8.8.8.8"
        }
    ]
}
```

Example Response

```
{
    "kind": "tm:security:firewall:address-list:address-liststate",
    "name": "google-dns_address_list",
    "partition": "Common",
    "fullPath": "/Common/google-dns_address_list",
    "generation": 11436,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/address-list/~Common~
↪google-dns_address_list?ver=13.0.0",
    "addresses": [
        {
            "name": "8.8.4.4"
        },
        {
            "name": "8.8.8.8"
        }
    ]
}
```

4.3.3 Lab 2.3: Create AFM Policy

Overview

In this lab, the iControl REST based API will be used to create a firewall policy that will leverage the previously created address list.

Specific Instructions

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

Attention: Some response content has been removed for brevity.

1. List AFM policies

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:security:firewall:policy:policycollectionstate",ƒ
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy?ver=13.0.0",
    "items": [
        {
            "kind": "tm:security:firewall:policy:policystate",
            "name": "block_all",
            "partition": "Common",
            "fullPath": "/Common/block_all",
            "generation": 5789,
            "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~
↪block_all?ver=13.0.0",
            "rulesReference": {
                "link": "https://localhost/mgmt/tm/security/firewall/policy/~Common~
↪block_all/rules?ver=13.0.0",
                "isSubcollection": true
            }
        }
    ]
}
```

2. Create AFM policy

An HTTP POST to the `/mgmt/tm/security/firewall/policy` endpoint with a body containing just a policy name creates a firewall policy.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name": "global_default_deny"
}
```

Example Response

Note: Copy the full policy name as it appears in the `"selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_default_deny?ver=13.0.0"` line of the response and populate the **{{afm_policy}}** Postman environment variable. In this case, the name of the policy is `~Common~global_default_deny`.

```
{
    "kind": "tm:security:firewall:policy:policystate",
    "name": "global_default_deny",
    "partition": "Common",
    "fullPath": "/Common/global_default_deny",
    "generation": 11451,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny?ver=13.0.0",
    "rulesReference": {
        "link": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules?ver=13.0.0",
        "isSubcollection": true
    }
}
```

3. List AFM policy rules

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: There will be no rules listed in the newly created policy. Rules are populated in the `"items": []` sub collection.

```
{
    "kind": "tm:security:firewall:policy:rules:rulescollectionstate",
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules?ver=13.0.0",
    "items": []
}
```

4. Add default deny rule to policy

An HTTP POST to the `/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules` endpoint with a body containing a new rule will add the rule to the firewall policy.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name": "default_deny",
    "fullPath": "default_deny",
    "action": "drop",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "no",
    "status": "enabled",
    "destination": { }
    "place-before": "none"
}
```

Example Response

```
{
    "kind": "tm:security:firewall:policy:rules:rulesstate",
    "name": "default_deny",
    "fullPath": "default_deny",
    "generation": 11464,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules/default_deny?ver=13.0.0",
    "action": "drop",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "no",
    "status": "enabled",
    "destination": {},
    "source": {
        "identity": {}
    }
}
```

5. Add address list rule to policy

An HTTP POST to the `/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules` endpoint with a body containing a new rule will add the rule to the firewall policy. The status of the rule can be specified when the POST is made.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name": "allow_google-dns",
    "fullPath": "allow_google-dns",
    "action": "accept",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "no",
    "status": "enabled",
    "placeBefore": "default_deny",
    "destination": {
        "addressLists": [
        "/Common/google-dns_address_list"
        ]
    }
}
```

Example Response

Note: Copy the newly created rule name `allow_google-dns` and populate the {{afm_policy_rule}} Postman environment variable.

```
{
    "kind": "tm:security:firewall:policy:rules:rulesstate",
    "name": "allow_google-dns",
    "fullPath": "allow_google-dns",
    "generation": 13210,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules/allow_google-dns?ver=13.0.0",
    "action": "accept",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "no",
    "status": "enabled",
    "destination": {
        "addressLists": [
        "/Common/google-dns_address_list"
        ],
        "addressListsReference": [
        {
            "link": "https://localhost/mgmt/tm/security/firewall/address-list/~Common~
↪allow_google-dns?ver=13.0.0"
        }
        ]
    },
    "source": {
        "identity": {}
    }
}
```

6. List policy rules

The `"items"` sub collection will now be populated with the all the firewall rules when performing an HTTP GET on the rules endpoint of the **{{afm_policy}}**.

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:security:firewall:policy:rules:rulescollectionstate",
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules?ver=13.0.0",
    "items": [
        {
                "kind": "tm:security:firewall:policy:rules:rulesstate",
                "name": "allow_google-dns",
                "fullPath": "allow_google-dns",
                "generation": 11483,
                "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~
↪Common~global_default_deny/rules/allow_google-dns?ver=13.0.0",
                "action": "accept",
                "ipProtocol": "any",
                "iruleSampleRate": 1,
                "log": "yes",
                "status": "enabled",
                "destination": {
                    "addressLists": [
                    "/Common/google-dns_address_list"
                    ],
                    "addressListsReference": [
                    {
                        "link": "https://localhost/mgmt/tm/security/firewall/address-
↪list/~Common~google-dns_address_list?ver=13.0.0"
                    }
                    ]
                },
                "source": {
                    "identity": {}
                }
            },
            {
                "kind": "tm:security:firewall:policy:rules:rulesstate",
                "name": "default_deny",
                "fullPath": "default_deny",
                "generation": 11464,
                "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~
↪Common~global_default_deny/rules/default_deny?ver=13.0.0",
                "action": "drop",
                "ipProtocol": "any",
                "iruleSampleRate": 1,
                "log": "no",
                "status": "enabled",
                "destination": {},
                "source": {
                    "identity": {}
            }
        }
    ]
}
```

7. Disable Policy rule

An HTTP PATCH to the `/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules/{{afm_policy_rule}}` endpoint with a body containing a name of an existing rule can set the `"status": "disabled"` to deactivate a single rule.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules/
↪{{afm_policy_rule}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "status": "disabled"
}
```

Example Response

```
{
    "kind": "tm:security:firewall:policy:rules:rulesstate",
    "name": "allow_google-dns",
    "fullPath": "allow_google-dns",
    "generation": 11470,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules/allow_google-dns?ver=13.0.0",
    "action": "accept",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "no",
    "status": "disabled",
    "destination": {
        "addressLists": [
            "/Common/google-dns_address_list"
        ],
        "addressListsReference": [
            {
                "link": "https://localhost/mgmt/tm/security/firewall/address-list/~
↪Common~google-dns_address_list?ver=13.0.0"
            }
        ]
    },
    "source": {
        "identity": {}
    }
}
```

8. List policy rule

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/security/firewall/policy/{{afm_policy}}/rules/{
↪{afm_policy_rule}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:security:firewall:policy:rules:rulesstate",
    "name": "allow_google-dns",
    "fullPath": "allow_google-dns",
    "generation": 11483,
    "selfLink": "https://localhost/mgmt/tm/security/firewall/policy/~Common~global_
↪default_deny/rules/allow_google-dns?ver=13.0.0",
    "action": "accept",
    "ipProtocol": "any",
    "iruleSampleRate": 1,
    "log": "yes",
    "status": "disabled",
    "destination": {
        "addressLists": [
        "/Common/google-dns_address_list"
        ],
        "addressListsReference": [
        {
            "link": "https://localhost/mgmt/tm/security/firewall/address-list/~Common~
↪google-dns_address_list?ver=13.0.0"
        }
        ]
    },
    "source": {
        "identity": {}
    }
}
```

4.4 Module 3: Programmatic Control of Web Application Firewall Services

- Provision ASM module on BIG-IP
- Interact with ASM related REST endpoints on a BIG-IP
- Create and modify an ASM policy
- Apply the ASM policy to a virtual server

4.4.1 Lab 3.1: Provisioning ASM

Overview

In this lab, the iControl REST API will be used to provision a module on the BIG-IP. More specifically, the Application Security Manager (ASM) module will be provisioned for use in **Module 3: Configuring ASM (Application Security Module)**.

Specific Instructions

Prior to performing the steps below, validate the **{{module}}** Postman environment variable. The **{{module}}** should be set to **asm**.

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

> **Attention:** Some response content has been removed for brevity.

1. Deprovision AFM module

This request is will serve as an example of how to deprovision a BIG-IP module.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/afm
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "level":"none"
}
```

Example Response

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "afm",
    "fullPath": "afm",
    "generation": 10714,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/afm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "none",
    "memoryRatio": 0
}
```

2. Retrieve all module provision states

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: The **asm** module is currently provisioned for **none** while the **ltm** module is provisioned for **nominal**.

```
{
    "kind": "tm:sys:provision:provisioncollectionstate",
    "selfLink": "https://localhost/mgmt/tm/sys/provision?ver=13.0.0",
    "items": [
        {
            "kind": "tm:sys:provision:provisionstate",
            "name": "asm",
            "fullPath": "asm",
            "generation": 5609,
            "selfLink": "https://localhost/mgmt/tm/sys/provision/asm?ver=13.0.0",
            "cpuRatio": 0,
            "diskRatio": 0,
            "level": "none",
            "memoryRatio": 0
        },
        {
            "kind": "tm:sys:provision:provisionstate",
            "name": "ltm",
            "fullPath": "ltm",
            "generation": 1,
            "selfLink": "https://localhost/mgmt/tm/sys/provision/ltm?ver=13.0.0",
            "cpuRatio": 0,
            "diskRatio": 0,
            "level": "nominal",
            "memoryRatio": 0
        }
    ]
}
```

3. Retrieve single module provision state

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: The **asm** module is currently not provisioned.

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "asm",
    "fullPath": "asm",
    "generation": 5609,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/asm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "none",
    "memoryRatio": 0
}
```

4. Provision ASM module

The **asm** module is provisioned using an HTTP PATCH with a body containing a provisioning level to the REST endpoint for `mgmt/tm/sys/provision/{{module}}`.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/sys/provision/{{module}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "level":"nominal"
}
```

Example Response

Note: The **asm** module has been provisioned with a **level** of **nominal**.

```
{
    "kind": "tm:sys:provision:provisionstate",
    "name": "asm",
    "fullPath": "asm",
    "generation": 10636,
    "selfLink": "https://localhost/mgmt/tm/sys/provision/asm?ver=13.0.0",
    "cpuRatio": 0,
    "diskRatio": 0,
    "level": "nominal",
    "memoryRatio": 0
}
```

4.4.2 Lab 3.2: Interact with ASM

Overview

In this lab, the iControl REST based API will be used to explore some of the ASM related endpoints.

Specific Instructions

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

> **Attention:** Some response content has been removed for brevity.

1.0. Retrieve ASM resources

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:asmcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm?ver=13.0.0",
    "items": [
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/policies?ver=13.0.0"
            }
        },
            {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/server-technologies?ver=13.0.0"
            }
        }
    ]
}
```

1.1. Retrieve ASM server technologies

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/server-technologies
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:server-technologies:server-technologycollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/server-technologies?ver=13.0.0",
    "totalItems": 40,
    "items": [
        {
            "serverTechnologyDisplayName": "jQuery",
            "serverTechnologyName": "jQuery",
            "logoFileName": "jquery.png",
            "lastUpdateMicros": 1476919661000000,
            "description": "jQuery is a cross-platform JavaScript library designed to␣
↪simplify the client-side scripting of HTML.",
            "kind": "tm:asm:server-technologies:server-technologystate",
            "serverTechnologyReferences": [],
            "selfLink": "https://localhost/mgmt/tm/asm/server-technologies/9ZC0_aLDC-
↪KN08jDyvXHew?ver=13.0.0",
            "id": "9ZC0_aLDC-KN08jDyvXHew"
        },
        {
            "serverTechnologyDisplayName": "Java Servlets/JSP",
            "serverTechnologyName": "Java Servlets/JSP",
            "logoFileName": "java.png",
            "lastUpdateMicros": 1476919661000000,
            "description": "A Java servlet is a Java program that extends the␣
↪capabilities of a server.",
            "kind": "tm:asm:server-technologies:server-technologystate",
            "serverTechnologyReferences": [],
            "selfLink": "https://localhost/mgmt/tm/asm/server-technologies/
↪9ySigIBMpBbYU4r8FNAt4g?ver=13.0.0",
            "id": "9ySigIBMpBbYU4r8FNAt4g"
        }
    ]
}
```

2.0. Retrieve ASM policies

A test policy named `test_asm_policy` has already been created on the BIG-IP for demonstration purposes.

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

Note: Copy the ASM policy hash as it appears in the `"link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/plain-text-profiles?ver=13.0.0",` line of the response and populate the **{{asm_policy_hash}}** Postman environment variable.

```
{
    "kind": "tm:asm:policies:policycollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/policies?ver=13.0.0",
    "totalItems": 1,
    "items": [
        {
        "plainTextProfileReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪plain-text-profiles?ver=13.0.0",
            "isSubCollection": true
        },
        "dataGuardReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪data-guard?ver=13.0.0"
        }
    ]
}
```

2.1. Retrieve ASM policy

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-06-02T04:37:22Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": " Security Policy /Common/test_asm_policy [add]: Type was␣
↪set to Security.\nEncoding Selected was set to false.\nApplication Language was set␣
↪to utf-8.\nCase Sensitivity was set to Case Sensitive.\nTemplate was set to POLICY_
↪TEMPLATE_FUNDAMENTAL.\nActive was set to false.\nDifferentiate between HTTP and␣
↪HTTPS URLs was set to Protocol Specific.\nPolicy Name was set to /Common/test_asm_
↪policy.\nEnforcement Mode was set to Blocking. { audit: policy = /Common/test_asm_
↪policy, username = admin, client IP = 192.168.2.111 }",
    "name": "test_asm_policy",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/header-
↪settings?ver=13.0.0"
```

```
    },
    "versionPolicyName": "/Common/test_asm_policy",
    "generalReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪general?ver=13.0.0"
    }
}
```

3. Search for ASM policy

An HTTP GET to the `/mgmt/tm/asm/policies` endpoint with a parameter of `filter=name eq test`, allows ASM policies to be searched by name.

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies?filter=name eq test
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-06-02T04:37:22Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": " Security Policy /Common/test_asm_policy [add]: Type was␣
↪set to Security.\nEncoding Selected was set to false.\nApplication Language was set␣
↪to utf-8.\nCase Sensitivity was set to Case Sensitive.\nTemplate was set to POLICY_
↪TEMPLATE_FUNDAMENTAL.\nActive was set to false.\nDifferentiate between HTTP and␣
↪HTTPS URLs was set to Protocol Specific.\nPolicy Name was set to /Common/test_asm_
↪policy.\nEnforcement Mode was set to Blocking. { audit: policy = /Common/test_asm_
↪policy, username = admin, client IP = 192.168.2.111 }",
    "name": "test_asm_policy",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/header-
↪settings?ver=13.0.0"
    },
    "versionPolicyName": "/Common/test_asm_policy",
    "generalReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪general?ver=13.0.0"
    }
}
```

4.0. List ASM tasks

An HTTP GET to the `/mgmt/tm/asm/tasks/` endpoint lists the various ASM related tasks that can be performed via the iControl REST API.

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/tasks/
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:tasks",
    "selfLink": "https://localhost/mgmt/tm/asm/tasks?ver=13.0.0",
    "items": [
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/export-policy?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/resolve-vulnerabilities?
↪ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/check-signatures?ver=13.
↪0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/terminate-vulnerability-
↪assessment?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/apply-server-
↪technologies?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/associate-xml-validation-
↪files-to-xml-profile?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/export-policy-template?
↪ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/export-requests?ver=13.0.
↪0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/initiate-vulnerability-
↪assessment?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/import-policy-template?
↪ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/update-signatures?ver=13.
↪0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/import-policy?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/bulk?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/get-vulnerability-
↪assessment-subscriptions?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/apply-policy?ver=13.0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/export-signatures?ver=13.
↪0.0"
            }
        },
        {
            "reference": {
                "link": "https://localhost/mgmt/tm/asm/tasks/import-vulnerabilities?
↪ver=13.0.0"
```

```
            }
        }
    ]
}
```

4.1. List specific ASM task

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/tasks/export-policy
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:tasks:export-policy:export-policy-taskcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/tasks/export-policy?ver=13.0.0",
    "totalItems": 0,
    "items": []
}
```

5. Retrieve ASM policy templates

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policy-templates
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:policy-templates:policy-templatecollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/policy-templates?ver=13.0.0",
    "totalItems": 32,
    "items": [
        {
        "policyType": "security",
        "name": "POLICY_TEMPLATE_SHAREPOINT_2007_HTTP",
        "description": "Generic template for SharePoint 2007 (http)",
        "kind": "tm:asm:policy-templates:policy-templatestate",
        "templateType": "application-ready",
        "selfLink": "https://localhost/mgmt/tm/asm/policy-templates/jmHjN-Fpm-
↪SGwYQsrZp57A?ver=13.0.0",
        "templateDefaults": {
            "caseInsensitive": true,
            "learningSpeed": {
            "untrustedTrafficSiteChangeTracking": {
                "maxDaysBetweenSamples": 7,
                "differentSources": 10,
                "minMinutesBetweenSamples": 5
            },
            "untrustedTrafficLoosen": {
                "maxDaysBetweenSamples": 7,
                "differentSources": 20,
                "minHoursBetweenSamples": 1
            },
            "trustedTrafficSiteChangeTracking": {
                "maxDaysBetweenSamples": 7,
                "differentSources": 1,
                "minMinutesBetweenSamples": 0
            },
            "trustedTrafficLoosen": {
                "maxDaysBetweenSamples": 7,
                "differentSources": 1,
                "minHoursBetweenSamples": 0
            },
            "trafficTighten": {
                "minDaysBetweenSamples": 1,
                "totalRequests": 15000,
                "maxModificationSuggestionScore": 50
            }
            },
            "enforcementReadinessPeriod": 7,
            "learningMode": "disabled",
            "applicationLanguage": "utf-8",
            "enforcementMode": "transparent",
            "signatureStaging": true,
            "type": "security",
            "protocolIndependent": false
        },
        "title": "SharePoint 2007 (http)",
        "id": "jmHjN-Fpm-SGwYQsrZp57A"
        }
    ]
}
```

6. Retrieve ASM signature sets

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/signature-sets
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:signature-sets:signature-setcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/signature-sets?ver=13.0.0",
    "totalItems": 29,
    "items": [
        {
        "filter": {
            "riskFilter": "all",
            "accuracyFilter": "all",
            "userDefinedFilter": "all",
            "lastUpdatedFilter": "all",
            "accuracyValue": "all",
            "riskValue": "all",
            "signatureType": "all"
        },
        "isUserDefined": false,
        "name": "Generic Detection Signatures",
        "assignToPolicyByDefault": true,
        "lastUpdateMicros": 0,
        "kind": "tm:asm:signature-sets:signature-setstate",
        "selfLink": "https://localhost/mgmt/tm/asm/signature-sets/pBeUaadz6x-Z55_
↪GkLxfsg?ver=13.0.0",
        "defaultAlarm": true,
        "systems": [
            {
                "systemReference": {
                    "link": "https://localhost/mgmt/tm/asm/signature-systems/
↪EStDgGiP9nSPgKBhSlDyvQ?ver=13.0.0"
                }
                },
                {
                "systemReference": {
                    "link": "https://localhost/mgmt/tm/asm/signature-systems/
↪rMiBJmL6DLmnfmW_pXHmdw?ver=13.0.0"
                }
                },
                {
                "systemReference": {
                    "link": "https://localhost/mgmt/tm/asm/signature-systems/
↪b9hI1sIulARJ09bbdy0VQw?ver=13.0.0"
                }
            }
        ],
        "id": "pBeUaadz6x-Z55_GkLxfsg",
        "type": "filter-based",
        "signatureReferences": [
            {
            "link": "https://localhost/mgmt/tm/asm/signatures/nHU-8zUxj8ldUevwMgFpvw?
↪ver=13.0.0"
            },
            {
            "link": "https://localhost/mgmt/tm/asm/signatures/RTFj6E66sH7g7XMa9ihQOQ?
↪ver=13.0.0"
            }
        ],
        "category": "User-defined",
        "defaultBlock": true,
        "defaultLearn": true
        }
    ]
}
```

7. Retrieve ASM signature systems

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/signature-systems
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:signature-systems:signature-systemcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/signature-systems?ver=13.0.0",
    "totalItems": 44,
    "items": [
        {
            "kind": "tm:asm:signature-systems:signature-systemstate",
            "selfLink": "https://localhost/mgmt/tm/asm/signature-systems/
↪EStDgGiP9nSPgKBhSlDyvQ?ver=13.0.0",
            "name": "General Database",
            "id": "EStDgGiP9nSPgKBhSlDyvQ"
        },
        {
            "kind": "tm:asm:signature-systems:signature-systemstate",
            "selfLink": "https://localhost/mgmt/tm/asm/signature-systems/
↪rMiBJmL6DLmnfmW_pXHmdw?ver=13.0.0",
            "name": "Various systems",
            "id": "rMiBJmL6DLmnfmW_pXHmdw"
        }
    ]
}
```

8. Retrieve ASM attack types

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/attack-types
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:attack-types:attack-typecollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/attack-types?ver=13.0.0",
    "totalItems": 37,
    "items": [
        {
            "kind": "tm:asm:attack-types:attack-typestate",
            "selfLink": "https://localhost/mgmt/tm/asm/attack-types/9yL3q5_
↪pO0E3pK1Uz9x2cw?ver=13.0.0",
            "name": "Remote File Include",
            "id": "9yL3q5_pO0E3pK1Uz9x2cw",
            "description": "Remote File Inclusion attacks allow attackers to run␣
↪arbitrary code on a vulnerable website."
        },
        {
            "kind": "tm:asm:attack-types:attack-typestate",
            "selfLink": "https://localhost/mgmt/tm/asm/attack-types/
↪ufg0smEkZrpmkoDHfSPGdQ?ver=13.0.0",
            "name": "Non-browser Client",
            "id": "ufg0smEkZrpmkoDHfSPGdQ",
            "description": "An attempt is made by a non-browser client to explore the␣
↪site."
        }
    ]
}
```

9. Retrieve ASM policy urls

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}/urls
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:policies:urls:urlcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/urls?
↪ver=13.0.0",
    "totalItems": 2,
    "items": [
        {
        "protocol": "http",
        "wildcardIncludesSlash": true,
        "lastLearnedNewEntityDatetime": "2017-06-02T04:37:25Z",
        "html5CrossOriginRequestsEnforcement": {
            "enforcementMode": "disabled"
        },
        "kind": "tm:asm:policies:urls:urlstate",
        "selfLink": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪urls/faiefv884qtHRU3Qva2AbQ?ver=13.0.0",
        "methodsOverrideOnUrlCheck": false,
        "id": "faiefv884qtHRU3Qva2AbQ",
        "isAllowed": true,
        "metacharsOnUrlCheck": false,
        "name": "*",
        "lastUpdateMicros": 1496378251000000,
        "description": "",
        "parameterReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪urls/faiefv884qtHRU3Qva2AbQ/parameters?ver=13.0.0",
            "isSubCollection": true
        },
        "attackSignaturesCheck": true,
        "signatureOverrides": [],
        "clickjackingProtection": false,
        "urlContentProfiles": [
            {
            "headerValue": "*",
            "headerName": "*",
            "headerOrder": "default",
            "type": "apply-value-and-content-signatures"
            },
            {
            "headerValue": "*form*",
            "headerName": "Content-Type",
            "headerOrder": "1",
            "type": "form-data"
            },
            {
            "contentProfileReference": {
                "link": "https://localhost/mgmt/tm/asm/policies/W-
↪w3q351kYbr1A9OEaUOag/json-profiles/X8FbXF48VWJ5Tecp5ATd4A?ver=13.0.0"
            },
            "headerValue": "*json*",
            "headerName": "Content-Type",
            "headerOrder": "2",
            "type": "json"
            },
            {
            "contentProfileReference": {
                "link": "https://localhost/mgmt/tm/asm/policies/W-
↪w3q351kYbr1A9OEaUOag/xml-profiles/jwQd_XYZPfNGYnc3l7P4Pg?ver=13.0.0"
            },
            "headerValue": "*xml*",
            "headerName": "Content-Type",
            "headerOrder": "3",
            "type": "xml"
            }
        ],
        "performStaging": true,
        "type": "wildcard",
        "wildcardOrder": 2
        },
        {
        "protocol": "https",
        "wildcardIncludesSlash": true,
        "lastLearnedNewEntityDatetime": "2017-06-02T04:37:25Z",
        "html5CrossOriginRequestsEnforcement": {
            "enforcementMode": "disabled"
        },
        "kind": "tm:asm:policies:urls:urlstate",
        "selfLink": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪urls/N_a3D1S7OKDehYEPb-mgCg?ver=13.0.0",
        "methodsOverrideOnUrlCheck": false,
        "id": "N_a3D1S7OKDehYEPb-mgCg",
        "isAllowed": true,
        "metacharsOnUrlCheck": false,
        "name": "*",
        "lastUpdateMicros": 1496378251000000,
        "description": "",
        "parameterReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪urls/N_a3D1S7OKDehYEPb-mgCg/parameters?ver=13.0.0",
            "isSubCollection": true
        },
        "attackSignaturesCheck": true,
        "signatureOverrides": [],
        "clickjackingProtection": false,
        "urlContentProfiles": [
            {
            "headerValue": "*",
            "headerName": "*",
            "headerOrder": "default",
            "type": "apply-value-and-content-signatures"
            },
            {
            "headerValue": "*form*",
            "headerName": "Content-Type",
            "headerOrder": "1",
            "type": "form-data"
            },
            {
            "contentProfileReference": {
                "link": "https://localhost/mgmt/tm/asm/policies/W-
↪w3q351kYbr1A9OEaUOag/json-profiles/X8FbXF48VWJ5Tecp5ATd4A?ver=13.0.0"
            },
            "headerValue": "*json*",
            "headerName": "Content-Type",
            "headerOrder": "2",
            "type": "json"
            },
            {
            "contentProfileReference": {
                "link": "https://localhost/mgmt/tm/asm/policies/W-
↪w3q351kYbr1A9OEaUOag/xml-profiles/jwQd_XYZPfNGYnc3l7P4Pg?ver=13.0.0"
            },
            "headerValue": "*xml*",
            "headerName": "Content-Type",
            "headerOrder": "3",
            "type": "xml"
            }
        ],
        "performStaging": true,
        "type": "wildcard",
        "wildcardOrder": 1
        }
    ]
}
```

10. Retrieve ASM policy signature sets

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}/signature-sets
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:policies:signature-sets:signature-setcollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪signature-sets?ver=13.0.0",
    "totalItems": 1,
    "items": [
        {
        "signatureSetReference": {
            "link": "https://localhost/mgmt/tm/asm/signature-sets/pBeUaadz6x-Z55_
↪GkLxfsg?ver=13.0.0"
        },
        "lastUpdateMicros": 1496378251000000,
        "selfLink": "https://localhost/mgmt/tm/asm/policies/W-w3q351kYbr1A9OEaUOag/
↪signature-sets/xMpCOKC5I4INzFCab3WEmw?ver=13.0.0",
        "kind": "tm:asm:policies:signature-sets:signature-setstate",
        "alarm": true,
        "block": true,
        "id": "xMpCOKC5I4INzFCab3WEmw",
        "learn": true
        }
    ]
}
```

4.4.3 Lab 3.3: Create ASM Policy

Overview

In this lab, the iControl REST based API will be used to create both an ASM parent and child policy.

Specific Instructions

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

Attention: Some response content has been removed for brevity.

1. Retrieve ASM policy

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "kind": "tm:asm:policies:policycollectionstate",
    "selfLink": "https://localhost/mgmt/tm/asm/policies?ver=13.0.0",
    "totalItems": 1,
    "items": [
        {
        "plainTextProfileReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/8JuF2s3Lb26BYwLXpaHLIg/
↪plain-text-profiles?ver=13.0.0",
            "isSubCollection": true
        },
        "dataGuardReference": {
            "link": "https://localhost/mgmt/tm/asm/policies/8JuF2s3Lb26BYwLXpaHLIg/
↪data-guard?ver=13.0.0"
        }
    ]
}
```

2.0. Create ASM parent policy

An HTTP POST to the `/mgmt/tm/asm/policies` endpoint with a body containing basic policy configuration including `"type":"parent"` will create a new ASM parent policy which can then be used for inheritance when a child policy is created.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"API_ASM_POLICY_TEST",
    "description":"Test ASM policy",
    "applicationLanguage":"utf-8",
    "type":"parent",
    "enforcementMode":"transparent",
    "protocolIndependent":"true",
    "learningMode":"disabled",
    "serverTechnologyName": "Unix/Linux"
}
```

Example Response

Note: Copy the ASM policy hash for the newly created policy and populate the {{asm_policy_hash}} Postman environment variable. The hash in the example below is JEQPVWeJcdso_rEC7Xxo6Q

```
{
    "historyRevisionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪history-revisions?ver=13.0.0",
        "isSubCollection": true
    },
    "childPolicyCount": 0,
    "responsePageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪response-pages?ver=13.0.0",
        "isSubCollection": true
    },
    "policyBuilderReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/policy-
↪builder?ver=13.0.0"
    },
    "serverTechnologyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/server-
↪technologies?ver=13.0.0",
        "isSubCollection": true
    },
    "blockingSettingReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪blocking-settings?ver=13.0.0",
        "isSubCollection": true
    },
    "hostNameReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/host-
↪names?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/data-
↪guard?ver=13.0.0"
    },
    "selfLink": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q?ver=13.
↪0.0",
    "signatureReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪signatures?ver=13.0.0",
        "isSubCollection": true
    },
    "filetypeReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪filetypes?ver=13.0.0",
        "isSubCollection": true
    },
    "createdDatetime": "2017-05-30T15:02:11Z",
    "modifierName": "",
    "id": "JEQPVWeJcdso_rEC7Xxo6Q",
    "subPath": "/Common",
    "name": "API_ASM_POLICY_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/header-
↪settings?ver=13.0.0"
    }
}
```

2.1. Retrieve ASM parent policy

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "historyRevisionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪history-revisions?ver=13.0.0",
        "isSubCollection": true
    },
    "childPolicyCount": 0,
    "responsePageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/
↪response-pages?ver=13.0.0",
        "isSubCollection": true
    },
    "policyBuilderReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/policy-
↪builder?ver=13.0.0"
    },
    "serverTechnologyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q/server-
↪technologies?ver=13.0.0",
        "isSubCollection": true
    }
}
```

3.0. Create ASM child policy

An HTTP POST to the `/mgmt/tm/asm/policies` endpoint with a body containing basic policy configuration including `"parentPolicyName": "/Common/API_ASM_POLICY_TEST"` will create a new child policy which inherits a base configuration from the specified parent.

Request

```
POST https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "name":"API_ASM_POLICY_CHILD_TEST",
    "description":"Test ASM policy",
    "applicationLanguage":"utf-8",
    "parentPolicyName": "/Common/API_ASM_POLICY_TEST",
    "enforcementMode":"transparent",
    "protocolIndependent":"true",
    "learningMode":"automatic",
    "learningSpeed":"slow",
    "serverTechnologyName": "Apache Tomcat"
}
```

Example Response

Note: Take note of the ASM policy hash for the newly created policy. Copy this value into your Postman's collection environmental variable for {{asm_policy_hash}}

The hash in the example below is `zD8sehzULw6Ni7GJG2XwJQ`

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    },
    "sectionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/
↪sections?ver=13.0.0",
        "isSubCollection": true
    },
    "loginPageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/login-
↪pages?ver=13.0.0",
        "isSubCollection": true
    },
    "description": "Test ASM policy",
    "fullPath": "/Common/API_ASM_POLICY_CHILD_TEST",
    "policyBuilderParameterReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/policy-
↪builder-parameter?ver=13.0.0"
    },
    "hasParent": true,
    "partition": "Common",
    "parentPolicyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q?ver=13.
↪0.0"
```

```
    }
}
```

3.1. Retrieve ASM child policy

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": " Security Policy /Common/API_ASM_POLICY_CHILD_TEST [add]:␣
↪Parent Policy was set to /Common/API_ASM_POLICY_TEST.\nType was set to Security.
↪\nEncoding Selected was set to true.\nApplication Language was set to utf-8.\nCase␣
↪Sensitivity was set to Case Sensitive.\nSecurity Policy Description was set to␣
↪Fundamental Policy.\nLearning Mode was set to Automatic.\nActive was set to false.
↪\nDifferentiate between HTTP and HTTPS URLs was set to Protocol Specific.\nPolicy␣
↪Name was set to /Common/API_ASM_POLICY_CHILD_TEST.\nEnforcement Mode was set to␣
↪Blocking. { audit: policy = /Common/API_ASM_POLICY_CHILD_TEST, username = admin,␣
↪client IP = 192.168.2.112 }",
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    },
    "sectionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/
↪sections?ver=13.0.0",
        "isSubCollection": true
    },
    "loginPageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/login-
↪pages?ver=13.0.0",
        "isSubCollection": true
    },
    "description": "Test ASM policy",
    "fullPath": "/Common/API_ASM_POLICY_CHILD_TEST",
    "policyBuilderParameterReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/policy-
↪builder-parameter?ver=13.0.0"
    },
    "hasParent": true,
    "partition": "Common",
    "parentPolicyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q?ver=13.
↪0.0"
    },
    "webScrapingReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/web-
↪scraping?ver=13.0.0"
    },
    "csrfProtectionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/csrf-
↪protection?ver=13.0.0"
    },
    "policyAntivirusReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/
↪antivirus?ver=13.0.0"
    },
    "kind": "tm:asm:policies:policystate",
    "virtualServers": [],
    "policyBuilderCookieReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/policy-
↪builder-cookie?ver=13.0.0"
    }
}
```

4.4.4 Lab 3.4: Apply ASM Policy to VS

Overview

In this lab, the previously created ASM policy will be applied to a virtual server using the iControl REST API.

Specific Instructions

Follow the below steps in order found in the Postman collection to complete this portion of the lab. The requests and responses have been included below for reference.

> **Attention:** Some response content has been removed for brevity.

1. Apply ASM Policy to VS

An HTTP PATCH to the `/mgmt/tm/asm/policies/{{asm_policy_hash}}` endpoint with a body containing the name of a virtual server(s), in this case `"virtualServers":["/Common/hackazon_vs"]`, will apply the ASM policy.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "virtualServers":["/Common/hackazon_vs"]
}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": " Security Policy /Common/API_ASM_POLICY_CHILD_TEST [add]:␣
↪Parent Policy was set to /Common/API_ASM_POLICY_TEST.\nType was set to Security.
↪\nEncoding Selected was set to true.\nApplication Language was set to utf-8.\nCase␣
↪Sensitivity was set to Case Sensitive.\nSecurity Policy Description was set to␣
↪Fundamental Policy.\nLearning Mode was set to Automatic.\nActive was set to false.
↪\nDifferentiate between HTTP and HTTPS URLs was set to Protocol Specific.\nPolicy␣
↪Name was set to /Common/API_ASM_POLICY_CHILD_TEST.\nEnforcement Mode was set to␣
↪Blocking. { audit: policy = /Common/API_ASM_POLICY_CHILD_TEST, username = admin,␣
↪client IP = 192.168.2.112 }",
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    },
    "sectionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/
↪sections?ver=13.0.0",
        "isSubCollection": true
    },
    "loginPageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/login-
↪pages?ver=13.0.0",
        "isSubCollection": true
    },
    "description": "Test ASM policy",
    "fullPath": "/Common/API_ASM_POLICY_CHILD_TEST",
    "policyBuilderParameterReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/policy-
↪builder-parameter?ver=13.0.0"
    },
    "hasParent": true,
    "partition": "Common",
    "parentPolicyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q?ver=13.
↪0.0"
    },
}
```

2. Retrieve ASM policy

Request

```
GET https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": "Policy Building Settings Policy Building Settings [update]:␣
↪Internal Statistics have been updated { audit: policy = /Common/API_ASM_POLICY_
↪CHILD_TEST, component = Policy Builder }",
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    }
}
```

3. Remove ASM Policy from VS

An HTTP PATCH to the `/mgmt/tm/asm/policies/{{asm_policy_hash}}` endpoint with a body removing the name of a virtual server(s), in this case `"virtualServers":[""]`, will remove the ASM policy from the absent virtual serves.

Request

```
PATCH https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
Content-Type: application/json
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Body

```
{
    "virtualServers":[""]
}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": "Policy Building Settings Policy Building Settings [update]:␣
↪Internal Statistics have been updated { audit: policy = /Common/API_ASM_POLICY_
↪CHILD_TEST, component = Policy Builder }",
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    },
    "sectionReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/
↪sections?ver=13.0.0",
        "isSubCollection": true
    },
    "loginPageReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/login-
↪pages?ver=13.0.0",
        "isSubCollection": true
    },
    "description": "Test ASM policy",
    "fullPath": "/Common/API_ASM_POLICY_CHILD_TEST",
    "policyBuilderParameterReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/policy-
↪builder-parameter?ver=13.0.0"
    },
    "hasParent": true,
    "partition": "Common",
    "parentPolicyReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/JEQPVWeJcdso_rEC7Xxo6Q?ver=13.
↪0.0"
    }
}
```

4. Delete ASM policy

An HTTP DELETE to the `/mgmt/tm/asm/policies/{{asm_policy_hash}}` endpoint will delete the ASM policy from the BIG-IP.

Request

```
DELETE https://{{big_ip_a_mgmt}}/mgmt/tm/asm/policies/{{asm_policy_hash}}
```

Headers

```
X-F5-Auth-Token: {{big_ip_a_auth_token}}
```

Example Response

```
{
    "plainTextProfileReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/plain-
↪text-profiles?ver=13.0.0",
        "isSubCollection": true
    },
    "dataGuardReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/data-
↪guard?ver=13.0.0"
    },
    "createdDatetime": "2017-05-30T15:45:59Z",
    "cookieSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/cookie-
↪settings?ver=13.0.0"
    },
    "versionLastChange": "Policy Building Settings Policy Building Settings [update]:␣
↪Internal Statistics have been updated { audit: policy = /Common/API_ASM_POLICY_
↪CHILD_TEST, component = Policy Builder }",
    "name": "API_ASM_POLICY_CHILD_TEST",
    "caseInsensitive": false,
    "headerSettingsReference": {
        "link": "https://localhost/mgmt/tm/asm/policies/zD8sehzULw6Ni7GJG2XwJQ/header-
↪settings?ver=13.0.0"
    }
}
```

5 HOWTOs: Index

This section contains useful HOWTOs

5.1 HOWTO - Update Existing iApp templates to Work with iWorkflow v2.1

This HOWTO document describes the minimal changes required to update an existing iApp template and add a version number to the template name.

Adding the version number allows the iApp template to be used by iWorkflow v2.1 and later. Versioning is required to enable iApp templates to be installed across many BIG-IP devices in a production-safe manner.

Without version information it is possible that iApp templates could be overwritten leading to deployment failures and/or outages.

5.1.1 Task 1 - Export the existing iApp from BIG-IP

The iApp template can be exported from a BIG-IP system where it has been installed. The file has a `.tmpl` extension and is a plaintext, readable format.

Complete the following steps:

1. Login to the BIG-IP GUI with admin credentials
2. Click iApps -> Templates
3. Find the desired template in the list and click the template name to open it
4. Scroll to the bottom of the page and click the 'Export' button
5. Click the `Download: ...` button and save the file to your computer

5.1.2 Task 2 - Edit the Exported template

We will now edit the template name to add a version number. iWorkflow currently supports the following formats:

- `template_name_v1.0_0`

- `template_name.v.1.0.0`
- `/<partition>/template_name.v1.0.0`

Complete the following steps:

1. Open the previously saved `.tmpl` file in a text editor
2. Perform a text search for `sys application template`

   Example:

```
cli admin-partitions {
    update-partition Common
}

sys application template my_template_name {
    actions {
        definition {
            implementation {
```

3. Modify the template name to include a version number using one of the formats specified at the beginning of this task.

   Example:

```
cli admin-partitions {
    update-partition Common
}

sys application template my_template_name.v1.0.0 {
    actions {
        definition {
            implementation {
```

4. Save the file

5.1.3 Task 3 - Import the iApp template to iWorkflow

The updated iApp template is now ready to be imported to iWorkflow. Instructions on how to do this can be found at:

https://devcentral.f5.com/wiki/iWorkflow.iWorkflowOpsGuide_7.ashx

6 Appendices

6.1 Appendix A: Python SDK

This module will cover the newly released F5 Python SDK. This SDK is released and maintained as a public GitHub repository at https://github.com/F5Networks/f5-common-python

The goal of the Python SDK is to provide a simple interface that abstracts many of the F5-specific nuances of the iControl REST API away from the user. As you learned in Module 1, when interacting directly with the API, it's often necessary to build out requests in a very manual fashion. In order to provide a simpler interface, the SDK was developed to abstract away many of the eccentricities of the API and provide a clean, Pythonic interface.

For example, when creating a pool in, an Imperative automation model, without the SDK you would be required to do something like the following (this code is not complete):

```
import requests
import sys
base_url = "https://10.1.1.4/mgmt/tm/ltm/pool/"

pool_attributes = {
        "name": "test_pool",
        "partition": "Common",
        "loadBalancingMode": "least-connections-member",
        "minUpMembers": 1
}

s = requests.session()
s.auth = ("admin", "admin")

resp = s.post(base_url, data=json.dumps(pool_attributes))

if resp.status_code != requests.codes.ok:
        print "Error creating pool"

sys.exit(1)
```

When using the Python SDK the equivalent code is:

```
from f5.bigip import ManagementRoot
```

```
mgmt = ManagementRoot("10.1.1.4","admin","admin")

pool = mgmt.tm.ltm.pools.pool.create(partition="Common", name="test_pool")
pool.loadBalancingMode = "least-connections-member"
pool.minUpMembers = 1

pool.update()
```

As you can see, the code utilizing the SDK is much more condensed and far easier to read. This is a result of the SDK exposing abstracted methods to build the URL. Additionally the SDK creates standard CURDLE (create, update, refresh, delete, load, exists) methods that behave correctly depending on REST object type (Organizing Collection, Resource, etc.) you are interacting with (e.g., you cannot DELETE an Organizing Collection, therefore a delete() method is not available).

Full documentation for the API exists at here

For the purpose of this lab, your Windows Jumphost has everything pre-installed, however, since the SDK is a standard python package the process is trivial on any system (Windows, Linux, Mac, etc.) that has Python installed.

It's important to keep in mind, while going through this module, that we are only demonstrating what is possible with the SDK from a high level. For example, the same scripts used in this module are designed to run from the command line with arguments, however, they could easily be modified to use JSON files as the input mechanism.

6.1.1 Lab A.1: create_pool.py

In this lab we will review, line-by-line an example script that has been created to allow creation of a BIG-IP Pool with Pool Members directly from the command line.

Task 1 - Review create_pool.py

1. Open Notepad++ using the located in the Windows Taskbar.
2. Double click the file `create_pool.py` in the menu on the left side of the Notepad++ screen
3. We will now review the code line-by-line:

```
from f5.bigip import ManagementRoot
import pprint
import argparse
pp = pprint.PrettyPrinter(indent=3)
```

These lines import in various Python libraries. The first line imports the F5 Python SDK. The pprint and argparse libraries are standard Python libraries that aid in print data to the console and parsing command line arguments.

```
parser = argparse.ArgumentParser(description='Script to create a pool on a BIG-IP␣
↪device')
parser.add_argument("host", help="The IP/Hostname of the BIG-IP device")
parser.add_argument("pool_name", help="The name of the pool")
parser.add_argument("pool_members", help="A comma seperated string in the format <IP>:
↪<port>[,<IP>:<port>]")
parser.add_argument("-P", "--partition", help="The partition name", default="Common")
parser.add_argument("-u", "--username", help="The BIG-IP username", default="admin")
parser.add_argument("-p", "--password", help="The BIG-IP password", default="admin")
args = parser.parse_args()
```

These lines setup the command line arguments for the script and store those arguments in a python dictionary names ‘args’. The argparse library automatically generates help text, checks for required arguments, sets defaults, etc.

```
mgmt = ManagementRoot(args.host, args.username, args.password)
```

This line creates a new Python object that refers to the BIG-IP device. We are calling the ManagementRoot method with 3 arguments:

- The value of the `host` argument
- The value of the `username` argument
- The value of the `password` argument

This method automatically performs a test to ensure that we are able to reach the device and authenticate successfully.

```
pool_path = "/%s/%s" % (args.partition, args.pool_name)
```

This line just stores the human-readable path to the pool name for later use

```
if mgmt.tm.ltm.pools.pool.exists(partition=args.partition, name=args.pool_name):
raise Exception("Pool '%s' already exists" % args.pool_name)
```

This if statement checks to see if a pool with the same name already exists on the specified partition on the device. The return value of the exists() method is a Boolean value of True or False. In this case we want the Exception to execute if a pool DOES exist and stop execution of the script.

```
pool = mgmt.tm.ltm.pools.pool.create(partition=args.partition, name=args.pool_name)
print "Created pool %s" % pool_path
```

The first line in this block actually creates the new pool. The partition and name of the pool are specified as arguments to the create() method and the ‘pool’ variable represents an object that holds the created pool’s properties. The second line simply prints a message that the pool has been created.

```
member_list = args.pool_members.split(',')
```

This line uses a built-in python method called split() to separate the value of the command line argument into discrete strings using a ‘,’ as a separator. The return type of the split() is a python list (lists = arrays)

```
for member in member_list:
pool_member = pool.members_s.members.create(partition=args.partition, name=member)
print " Added member %s" % member
```

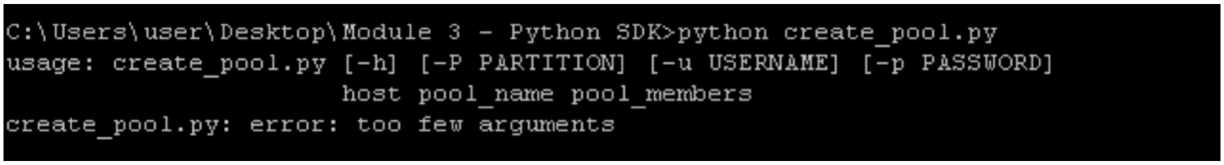
This for loop iterates over the elements in the list generated above and creates a new member in the pool.

Task 2 - Run create_pool.py

1. Open Console2 using the C:> icon on the Windows Taskbar
2. The console window automatically opens in the Desktop\Module 5 - Python SDK directory

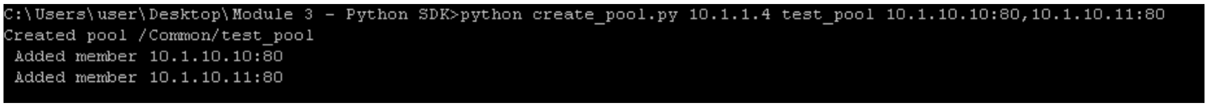
3. Type `set PYTHONWARNINGS=ignore` to disable the printing of SSL/TLS warnings about self-signed certificates.

4. Type `python create_pool.py` and examine the help output:



```
C:\Users\user\Desktop\Module 3 - Python SDK>python create_pool.py
usage: create_pool.py [-h] [-P PARTITION] [-u USERNAME] [-p PASSWORD]
                      host pool_name pool_members
create_pool.py: error: too few arguments
```

5. Type `python create_pool.py 10.1.1.4 test_pool 10.1.10.10:80,10.1.10.11:80` to create a new pool:



```
C:\Users\user\Desktop\Module 3 - Python SDK>python create_pool.py 10.1.1.4 test_pool 10.1.10.10:80,10.1.10.11:80
Created pool /Common/test_pool
 Added member 10.1.10.10:80
 Added member 10.1.10.11:80
```

6. Using Chrome open a tab to BIGIP-A (https://10.1.1.4). Examine the pool that was created.

6.1.2 Lab A.2: read_pool.py

In this lab we will review, line-by-line an example script that has been created to view the attributes of a BIG-IP Pool directly from the command line.

Task 1 - Review read_pool.py

1. Open `read_pool.py` in Notepad++

2. We will review the code. For brevity we have removed lines that are common with previous examples:

```
if not mgmt.tm.ltm.pools.pool.exists(partition=args.partition, name=args.pool_name):
raise Exception("Pool '%s' does not exist" % args.pool_name)
```

This if statement checks to see if a pool with the same name exists in the specified partition on the device. The key difference between this and the example in the previous lab is the inclusion of the ‘not’ keyword. This inverses the logic of the statement so that the Exception is raised when the pool DOES NOT exist

```
pool = mgmt.tm.ltm.pools.pool.load(partition=args.partition, name=args.pool_name)
```

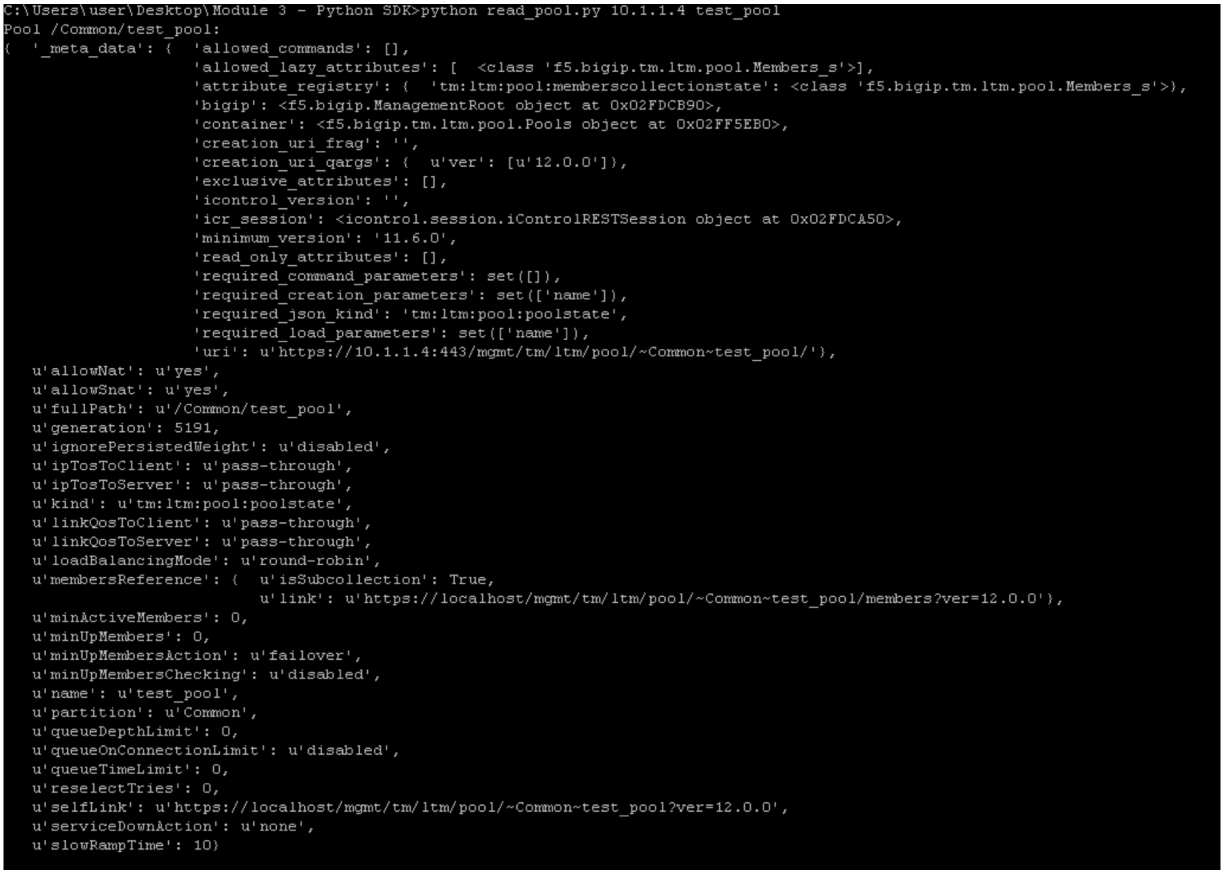
This line loads the configuration of the pool into a variable

```
print "Pool %s:" % pool_path
pp.pprint(pool.raw)
```

These lines print the human-readable pool path and then uses the PrettyPrint library to dump all the attributes associated with the pool

Task 2 - Run read_pool.py

1. In the command prompt type `python read_pool.py 10.1.1.4 test_pool` and examine the output:



```
C:\Users\user\Desktop\Module 3 - Python SDK>python read_pool.py 10.1.1.4 test_pool
Pool /Common/test_pool:
{   '_meta_data': {   'allowed_commands': [],
                      'allowed_lazy_attributes': [   <class 'f5.bigip.tm.ltm.pool.Members_s'>],
                      'attribute_registry': {   'tm:ltm:pool:memberscollectionstate': <class 'f5.bigip.tm.ltm.pool.Members_s'>},
                      'bigip': <f5.bigip.ManagementRoot object at 0x02FDCB90>,
                      'container': <f5.bigip.tm.ltm.pool.Pools object at 0x02FF5EB0>,
                      'creation_uri_frag': '',
                      'creation_uri_qargs': {   u'ver': [u'12.0.0']},
                      'exclusive_attributes': [],
                      'icontrol_version': '',
                      'icr_session': <icontrol.session.iControlRESTSession object at 0x02FDCA50>,
                      'minimum_version': '11.6.0',
                      'read_only_attributes': [],
                      'required_command_parameters': set([]),
                      'required_creation_parameters': set(['name']),
                      'required_json_kind': 'tm:ltm:pool:poolstate',
                      'required_load_parameters': set(['name']),
                      'uri': u'https://10.1.1.4:443/mgmt/tm/ltm/pool/~Common~test_pool/'},
    u'allowNat': u'yes',
    u'allowSnat': u'yes',
    u'fullPath': u'/Common/test_pool',
    u'generation': 5191,
    u'ignorePersistedWeight': u'disabled',
    u'ipTosToClient': u'pass-through',
    u'ipTosToServer': u'pass-through',
    u'kind': u'tm:ltm:pool:poolstate',
    u'linkQosToClient': u'pass-through',
    u'linkQosToServer': u'pass-through',
    u'loadBalancingMode': u'round-robin',
    u'membersReference': {   u'isSubcollection': True,
                             u'link': u'https://localhost/mgmt/tm/ltm/pool/~Common~test_pool/members?ver=12.0.0'},
    u'minActiveMembers': 0,
    u'minUpMembers': 0,
    u'minUpMembersAction': u'failover',
    u'minUpMembersChecking': u'disabled',
    u'name': u'test_pool',
    u'partition': u'Common',
    u'queueDepthLimit': 0,
    u'queueOnConnectionLimit': u'disabled',
    u'queueTimeLimit': 0,
    u'reselectTries': 0,
    u'selfLink': u'https://localhost/mgmt/tm/ltm/pool/~Common~test_pool?ver=12.0.0',
    u'serviceDownAction': u'none',
    u'slowRampTime': 10}
```

2. Notice the various attributes that are associated with the pool. Take note of the value of the `loadBalancingMode` attribute for the next lab

6.1.3 Lab A.3: update_pool.py

In this lab we will review, line-by-line an example script that has been created to allow updating any attribute of a pool using the command-line. This script is a good example of creating generic tools that enable many use cases. Rather than creating a script that just updates a specific attribute we created one that updates ANY pool attribute, greatly expanding it's potential use cases.

Task 1 - Review update_pool.py

1. Open `update_pool.py` in Notepad++
2. We will review the code. For brevity we have removed lines that are common with previous examples:

```
pool = mgmt.tm.ltm.pools.pool.load(partition=args.partition, name=args.pool_name)

pp.pprint("Current: %s=%s" % (args.attribute, getattr(pool, args.attribute)))
```

These lines load the pool from the device and print the current value of the attribute specified on the the command line. The second line is a little bit tricky because the SDK dynamically populates the objects attributes based on the type of object (pool, virtual server, etc.). Normally we could just use something like 'pool.loadBalancingMode' to get the current lb-method for the pool, however, since this script implements a way to change ANY attribute in the object we have to dynamically substitute the attribute name at run-time.

To do this we use the getattr() python built-in function to resolve the mapping at runtime and return the value of the attribute specified on the command line.

```
kwargs = {args.attribute: args.value}
```

This line creates a new python dictionary with one entry specifying a key-value pair using the command line arguments. For example if you were updated the loadBalancingMode attribute to 'least-connections-member' the dictionary would look like {"loadBalancingMode":"least-connections-member"}

```
pool.update(**kwargs)
```

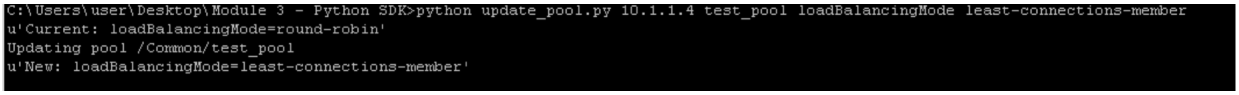
The first line updates the pool we loaded previously with the new value for the attribute. The **kwargs argument to the update() method triggers a special mechanism in python called 'keyword unpacking' which allows us to pass the attribute to be updated to the update() method.

```
pool.refresh()
pp.pprint("New: %s=%s" % (args.attribute, getattr(pool, args.attribute)))
```

The first line refreshes the data in the object from the BIG-IP device. The second line prints this refreshed information to the console so the user can verify the update completed successfully.

Task 2 - Run update_pool.py

1. In the command prompt type `python update_pool.py 10.1.1.4 test_pool loadBalancingMode least-connections-member` and examine the output:



```
C:\Users\user\Desktop\Module 3 - Python SDK>python update_pool.py 10.1.1.4 test_pool loadBalancingMode least-connections-member
u'Current: loadBalancingMode=round-robin'
Updating pool /Common/test_pool
u'New: loadBalancingMode=least-connections-member'
```

2. You can manually verify the load balancing method was changed via TMUI or by re-running `read_pool.py` (it's not required since the line that prints the new value forces a refresh())
3. Experiment with changing other pool attributes

6.1.4 Lab A.4: update_pool_member_state.py

One of the most common tasks asked for by customers is the ability to set a pool member's state via a script. We have included an example of such a script in the lab that can be used to see how easy it is to automate specific operational tasks.

Task 1 - Run update_pool_member_state.py

1. In the command prompt type `python update_pool_member_state.py 10.1.1.4 test_pool 10.1.10.10:80 disabled` and examine the output.
2. Verify the pool member was disabled via TMUI
3. Re-run the script with as `python update_pool_member_state.py --help` to see additional options.
4. Re-enable the pool member using the script

6.1.5 Lab A.5: delete_pool.py

In this lab we will review, line-by-line an example script that has been created to allow deletion of a pool using the command-line.

Task 1 - Review delete_pool.py

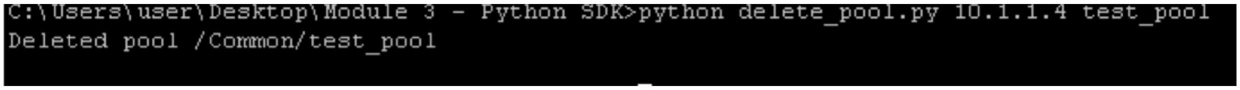
1. Open `delete_pool.py` in Notepad++
2. We will review the code. For brevity we have removed lines that are common with previous examples:

```
pool = mgmt.tm.ltm.pools.pool.load(partition=args.partition, name=args.pool\_name)
pool.delete()

print "Deleted pool %s" % pool\_path
```

These lines should be fairly self-explanatory at this point. First we load the pool and the we delete() it and print that we have done so.

Task 2 - Run delete_pool.py

1. In the command prompt type `python delete_pool.py 10.1.1.4 test_pool` and examine the output:



```
C:\Users\user\Desktop\Module 3 - Python SDK>python delete_pool.py 10.1.1.4 test_pool
Deleted pool /Common/test_pool
```

2. If desired verify the pool was deleted using TMUI or the `read_pool.py` script (it should return an error)

6.1.6 Lab A.6: Create a Python Script

In this lab we will use the 'Generate Code' feature of Postman to create a python script from a collection of requests.

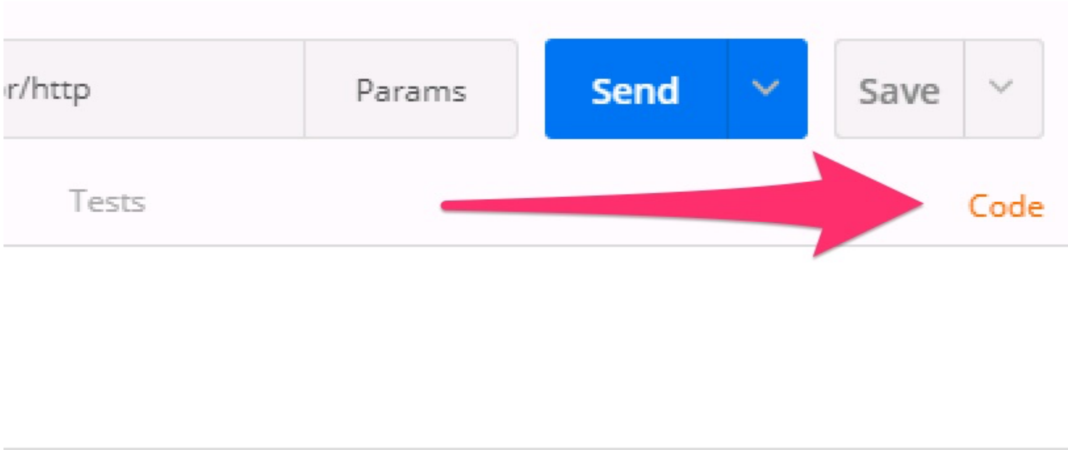
Task 1 - Create a simple script

Note: Remember to have the correct environment selected in Postman

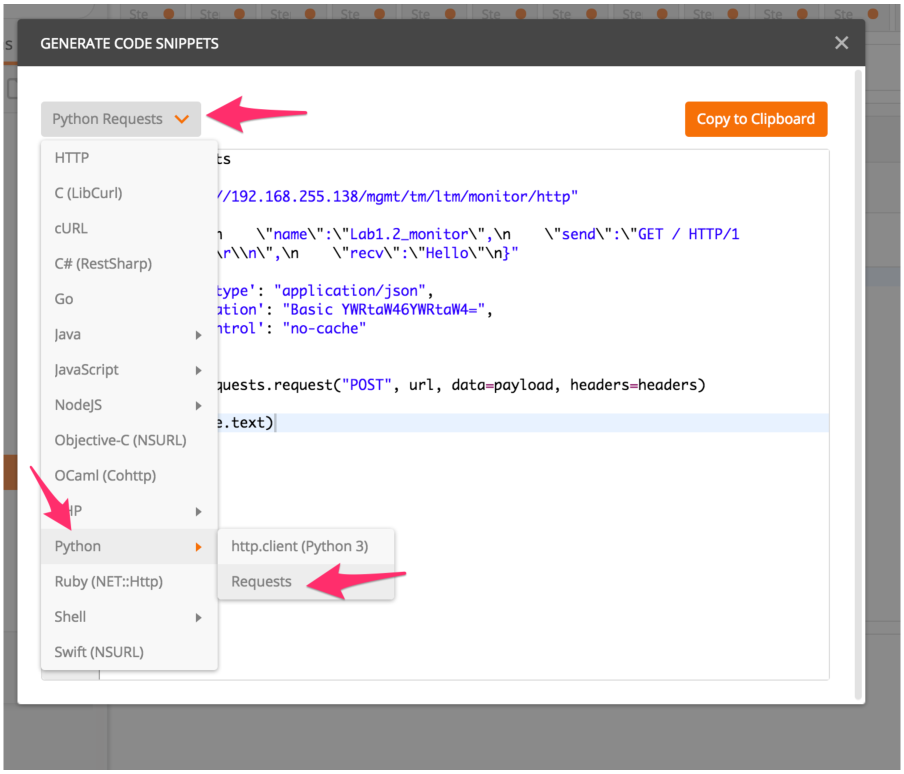
---

Perform the following steps to complete this task:

1. Expand the 'Lab 5.6 - Create a Python Script' folder in the Postman collection
2. Click the 'Step 1 - Create a HTTP Monitor' item in the collection
3. Click the 'Code' link in the Postman window:



4. Select Python -> Requests from the menu on the top right of the window:



```
ts

//192.168.255.138/mgmt/tm/ltm/monitor/http"

n    \"name\":\"Lab1.2_monitor\",\n    \"send\":\"GET / HTTP/1
\r\\n\",\n    \"recv\":\"Hello\"\n}"

type': "application/json",
ation': "Basic YWRtaW46YWRtaW4=",
ntrol': "no-cache"


quests.request("POST", url, data=payload, headers=headers)

e.text)
```

5. Examine the Python code that was generated. Click the 'Copy to Clipboard' button

6. Open a new text file and paste the generated code. We need to modify the line that sends the request to DISABLE SSL certificate verification. Find the following line:

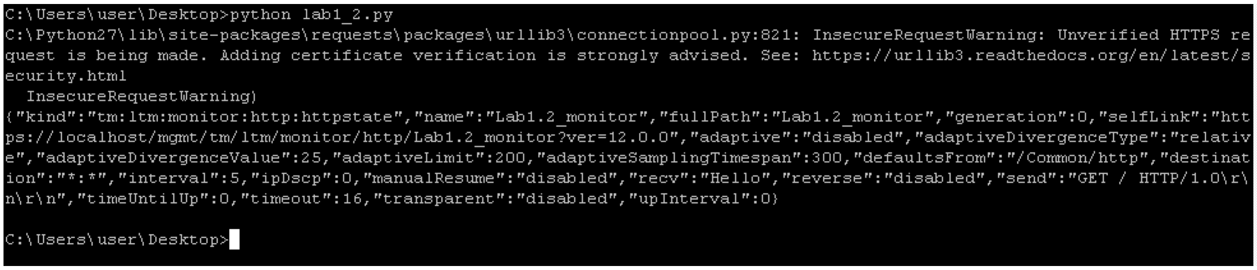
```
response = requests.request("POST", url, data=payload, headers=headers)
```

And add a verify=False option to it:

```
response = requests.request("POST", url, data=payload, headers=headers,␣
↪verify=False)
```

Save the file on your Desktop as lab5_6.py

7. Open a command prompt and run the script by typing `python lab5_6.py`:



```
C:\Users\user\Desktop>python lab1_2.py
C:\Python27\lib\site-packages\requests\packages\urllib3\connectionpool.py:821: InsecureRequestWarning: Unverified HTTPS re
quest is being made. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.org/en/latest/s
ecurity.html
  InsecureRequestWarning)
{"kind":"tm:ltm:monitor:http:httpstate","name":"Lab1.2_monitor","fullPath":"Lab1.2_monitor","generation":0,"selfLink":"htt
ps://localhost/mgmt/tm/ltm/monitor/http/Lab1.2_monitor?ver=12.0.0","adaptive":"disabled","adaptiveDivergenceType":"relativ
e","adaptiveDivergenceValue":25,"adaptiveLimit":200,"adaptiveSamplingTimespan":300,"defaultsFrom":"/Common/http","destinat
ion":"*:*","interval":5,"ipDscp":0,"manualResume":"disabled","recv":"Hello","reverse":"disabled","send":"GET / HTTP/1.0\r\
n\r\n","timeUntilUp":0,"timeout":16,"transparent":"disabled","upInterval":0}

C:\Users\user\Desktop>
```

8. Verify the monitor was created on BIG-IP
9. **Delete the monitor to prepare for the next task**

Task 2 - Chain together multiple requests

In this task we will repeat the process from Task 1 to chain together multiple requests.

Perform the following steps:

1. Repeat the procedure from Task 1 with each of the items in the 'Lab 5.6' postman collection. Append each snippet of code to your existing script until you have all 5 requests in the script. **You will need to remove the duplicate 'import requests' lines and update each request with the 'verify=False' option.**
2. Save the file
3. Run the script and verify the config was created.

6.1.7 Lab A.7: EXTRA CREDIT - Modify create_pool.py

This is an open-ended exercise. Copy `create_pool.py` to `create_vs.py` and modify it to create a Virtual Server. You could also cheat and look at `you_cheated.py`!

6.1.8 Lab A.8: EXTRA CREDIT - Review super_pool.py

This is an open-ended exercise. Review and run the `super_pool.py` script. This script allows bulk creation/deletion of pools using CSV files.

WE MAKE APPS GO FASTER. SMARTER. SAFER.

F5 Networks, Inc. | f5.com

US Headquarters: 401 Elliott Ave W, Seattle, WA 98119 | 888-882-4447 // Americas: info@f5.com // Asia-Pacific: apacinfo@f5.com // Europe/Middle East/Africa: emeainfo@f5.com // Japan: f5j-info@f5.com