

Agility 2018 Hands-on Lab Guide

Contents:

1	Class 1: Introduction to ADC Deployments with BIG-IP LTM	5
2	Class 2: Building the F5 Fabric	25
3	Class 3: BIG-IP® Local Traffic Manager (LTM) - v13.1 Lab Guide	57
4	Class 4: Troubleshoot with tcpdump and Wireshark	107
5	Resilient Data Center Architectures with F5 BIG-IP	123

Class 1: Introduction to ADC Deployments with BIG-IP LTM

Welcome to the ADC Deployments with BIG-IP LTM hands-on lab session. These labs are intended to guide you through creating basic ADC deployments and completing common administrative tasks. This guide is intended to complement lecture material provided during the ADC Deployments with BIG-IP LTM as well as a reference guide that can be referred to after the class.

1.1 Lab Network Setup

In the interest of focusing as much time as possible configuring your application delivery controller, we have provided some resources and basic setup ahead of time. These are:

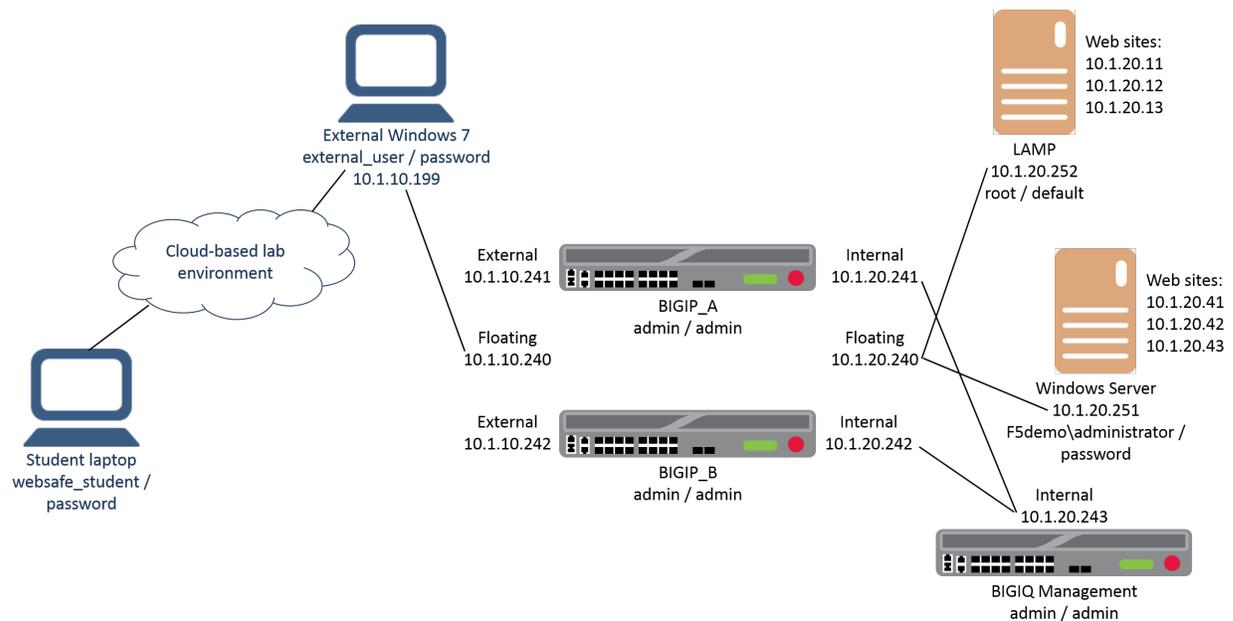
- Cloud-based lab environment complete with a Windows workstation, a virtual BIG-IP (VE), a virtual BIG-IQ acting as a logging node, a virtual BIG-IQ acting as a management node, and a back-end banking application running on a Linux web server.
- The virtual BIG-IP has been pre-licensed

If you wish to replicate these labs in your office you will need to perform these steps accordingly. Additional lab resources are provided as illustrated in the diagram on the next page.

To access the lab environment, you will require a web browser and Remote Desktop Protocol (RDP) client software. The web browser will be used to access the lab training portal. The RDP client will be used to connect to a Windows workstation, where you will be able to access the BIG-IP and BIG-IQ management interfaces (HTTPS, SSH).

Your class instructor will provide additional lab access details.

1.1.1 Lab Diagram



1.1.2 Timing for Labs

The time it takes to perform each lab varies and is mostly dependent on accurately completing steps. This can never be accurately predicted but we strived to derive an estimate among several people each having a different level of experience. Below is an estimate of how long it will take for each lab:

LAB Name (Description)	Time Allocated
LAB 1 – Configure Virtual Servers and Pools	35 minutes
LAB 2 – Work with SNAT, Profiles, and Monitors	45 minutes
LAB 3 – Use SSL Offload, Best Practices, and iApps	40 minutes
LAB 4 – Configure High Availability	30 minutes

1.2 Module 1: BIG-IP LTM Basic Configuration

In this module you will learn the basics of configuring BIG-IP Local Traffic Manager

1.2.1 Lab 1: Configure Virtual Servers and Pools

In this lab you will explore the BIG-IP configuration utility, create your first web application, and configure different types of virtual servers and load balancing methods.

Task 1 – Connect to Ravello and Examine the BIG-IP Configuration Utility

1. Use a browser to access **http://IP_address** with the IP address supplied by your instructor, and log in using the username and password supplied by your instructor.

2. For **ADC Implementations with LTM** click **View**.
3. Copy the IP address of the **Windows 7 External** VM, and then use RDP to access the IP address.
4. Log into the Windows workstation as **external_user / password**.
5. Open Chrome and click the **BIGIP_A** bookmark.
6. Log into the BIG-IP system as **admin / admin**.
7. From the left menu select **Local Traffic**.

The **Local Traffic** menu is where most ADC functions are performed.

8. From the left menu select **Network**.

The **Network** menu is where you configure elements for routing and switching.

9. From the left menu select **System**.

The **System** menu is where you configure DNS and NTP settings, manage licensing, perform software updates, and import SSL certificates.

10. Open the **Network > VLANs > VLAN List** page.

<input checked="" type="checkbox"/>	Name	Application	Tag	Untagged Interfaces	Tagged Interfaces	Partition / Path
<input type="checkbox"/>	external		4093	1.1		Common
<input type="checkbox"/>	internal		4094	1.2		Common

Two VLANs were already created, an **external** VLAN for outside access, and an **internal** VLAN for access to the internal network.

11. Open the **Network > Self IPs > Self IP List** page.

This BIG-IP system is configured with four self IP addresses. Each VLAN has a standard self IP address (ending in **.241**) and a floating self IP address (ending in **.240**). We'll use the floating self IP addresses during the high availability exercise.

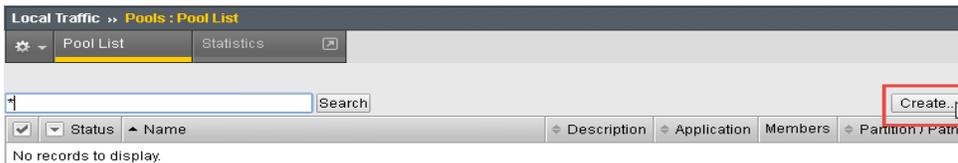
12. Open the **Network > Routes** page.

This BIG-IP system is configured with a default gateway route for outbound internet access (on **10.1.10.1**).

Task 2 – Create a Basic Web Application

Examine the lab diagram on page 2. We'll be creating a web application for an application that is stored on three web servers (at **10.1.20.11 – 10.1.20.13**).

1. Open the **Local Traffic > Pools > Pool List** page and click **Create**.



2. Use the following information for the new pool. For fields that are not specified, leave them set to the default settings.

Form field	Value
Name	http_pool
New Members	Node Name: node1 Address: 10.1.20.11 Service Port: 80 (Click Add)
	Node Name: node 2 Address 10.1.20.12 Service Port: 80 (Click Add)
	Node Name: node 3 Address: 10.1.20.13 Service Port: 80 (Click Add)

3. Click **Finished**.
4. Open the **Local Traffic > Virtual Servers > Virtual Server List** page and click **Create**.
5. Use the following information for the new virtual server, and then click **Finished**.

Form field	Value
Name	http_virtual
Destination Address/ Mask	10.1.10.20
Service Port	80
Resources > Default Pool	http_pool

6. Use a new tab to access **http://10.1.10.20**.
7. Use **Ctrl + F5** to reload the page several times.
You can see that page elements are coming from all three web servers. That's all it takes to create a basic web application on the BIG-IP system.
8. Close the tab.
9. In the Configuration Utility, open the **Local Traffic > Pools > Statistics** page.
10. Expand the **http_pool** by clicking on the **+** icon.



You use the **Statistics** page to identify the amount of traffic sent to the pool members. Notice that the requests are evenly distributed across all three web servers.

11. Select the **http_pool** checkbox, and then click **Reset**.



Task 3 – Create a Forwarding Virtual Server

1. Use a new tab to attempt direct access to an internal web server at **http://10.1.20.41**.
Currently you are unable to access resources on the internal network from the external Windows workstation.

2. Open the **Start** menu and type **cmd**, then right-click **cmd.exe** and select **Run as administrator**, and then click **Yes**.

3. At the command prompt, type (or copy and paste):

```
route add 10.1.20.0 mask 255.255.255.0 10.1.10.241
```

This adds a route to the **10.1.20.0** network through the external self IP address (**10.1.10.241**) of the BIG-IP system.

4. Reload the page directed at **http://10.1.20.41**.

The request fails again, as the BIG-IP system does not have a listener to forward this request to the internal network.

5. In the Configuration Utility, open the **Local Traffic > Virtual Servers > Virtual Server List** page and click **Create**.

6. Use the following information for the new virtual server, and then click **Finished**.

Form field	Value
Name	forward_virtual
Type	Forwarding (IP)
Destination Address/ Mask	10.1.20.0/24
Service Port	* All Ports
Protocol	* All Protocols

This virtual server provides access to the **10.1.20.0/24** network on all ports and all protocols.

7. Reload the page directed at **http://10.1.20.41**.

The request is successful. The BIG-IP system doesn't act as a full proxy, it simply forwards requests to the internal network.

8. Edit the URL to **https://10.1.20.32**.

9. Go to **Start > Remote Desktop Connection**.

10. Click **Show Options**, then select the **Display** tab, then change the **Display configuration** to **1024 by 768**.

11. Open the **General** tab and connect to **10.1.20.251** and log in as **administrator / password**.

12. On the Windows Server image go to **Start > Log off**.

You now have access to all ports and all protocols on the **10.1.20.0** network.

Task 4 – Create a Reject Virtual Server

1. In the Configuration Utility, on the **Virtual Server List** page click **Create**.

2. Use the following information for the new virtual server, and then click **Finished**.

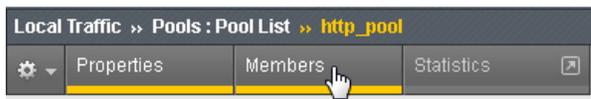
Form field	Value
Name	reject_win_server
Type	Reject
Destination Address/ Mask	10.1.20.251
Service Port	* All Ports
Protocol	* All Protocols

3. On the Lorax Intranet tab click **Corporate Tools**, and then close the tab.
4. Go to **Start > Remote Desktop Connection** and connect to **10.1.20.251**.
Although you still have access to the **10.1.20.0** network, you no longer have access to **10.1.20.251** (the Windows Server).
5. Close the **Remote Desktop Connection** window.
6. In the command prompt type the following, and then close the command prompt.

```
route DELETE 10.1.20.0
```
7. In the Configuration Utility, select the **forward_virtual** and **reject_win_server** checkboxes and then click **Delete** and **Delete** again.

Task 5 – Use Different Pool Options

1. Open the **Local Traffic > Pools > Pool List** page and click **http_pool**, and then open the **Members** page.



Currently the pool is using the default load balancing method: **Round Robin**.

2. From the **Load Balancing Method** list select **Ratio (member)**, and then click **Update**.
3. Examine the **Current Members** section.
Currently all three pool members have the same ratio value (**1**).
4. Click **node1:80**, then change the ratio value to **10**, and then click **Update**.
5. At the top of the page click **Members**, then click **node2:80**, then change the ratio value to **5**, and then click **Update**.
6. Click **Members** again and examine the **Current Members** section.
7. Use an incognito window to access **http://10.1.10.20**, then type **Ctrl + F5** at least 10 times to reload the page, and then close the page.



8. In the Configuration Utility, at the top of the page click **Statistics**.

Requests are now being distributed to the three pool members in a **10 – 5 – 1** ratio.

1.2.2 Lab 2: Work with SNAT, Profiles, and Monitors

In this lab you will experiment with using SNAT Auto Map for inbound requests as well as outbound requests from internal users. You'll also use an HTTP and stream profile to make global modifications to text within a web site. Finally you'll see how using health monitors ensures that you the BIG-IP knows which web servers are available for client requests.

Task 1 – Use SNAT AutoMap

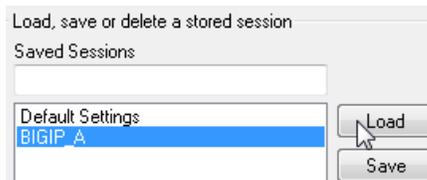
1. In the Configuration Utility, open the **Pool List** page and click **Create**.
2. Use the following information for the new pool, and then click **Finished**.

Form field	Value
Name	lorax_pool
New Members	Address: 10.1.20.41 Service Port: 80 (Click Add)
	Address 10.1.20.42 Service Port: 80 (Click Add)
	Address: 10.1.20.43 Service Port: 80 (Click Add)

3. Open the **Virtual Server List** page and click **Create**.
4. Use the following information for the new virtual server, and then click **Finished**.

Form field	Value
Name	lorax_virtual
Destination Address/ Mask	10.1.10.25
Service Port	80
Resources > Default Pool	lorax_pool

5. From the desktop open putty, and then connect to **BIGIP_A** and log in as **root / default**.



6. At the CLI type (or copy and paste):

```
tcpdump -i external port 80
```
7. Open a second putty session and connect to **BIGIP_A**.
8. At the CLI type (or copy and paste):

```
tcpdump -i internal port 80
```
9. Use a new tab to access **http://10.1.10.25**, and then close the tab.
The page displays as expected.
10. Examine the **tcpdump** windows.
On the external VLAN the communication is between the client IP address (**10.1.10.199**) and the virtual server (**10.1.10.25**).
On the internal VLAN the communication is between the client IP address (**10.1.10.199**) and a back-end web server (**10.1.20.x**).
11. In both **tcpdump** sessions press the **Enter** key several times to move the log entries to the top of the window.
12. In the Ravello window, open the Windows Server console and log in as **f5demo\administrator / password**.
13. In the Windows Server image, go to **Start > Control Panel**, then go to **Network and Internet > Network and Sharing Center**, and then click **Change adapter settings**.

14. Right-click on **Local Area Connection 3** and select **Properties**.

15. Select **Internet Protocol Version 4 (TCP/IPv4)** and select **Properties**.

Currently the Windows Server's default gateway is configured for the BIG-IP's internal self IP address (**10.1.20.241**). The network administrator has chosen to modify the default gateway to an external router.

16. Edit the **Default gateway** to **10.1.20.254**, then click **OK** and **Close**.

17. On the Windows desktop, use an incognito window to access **http://10.1.10.25**.

The page fails to load because the web server is now sending its responses to the external router, not the BIG-IP system.

18. Close the page, and then examine the **tcpdump** window.

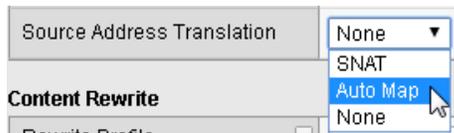
On the external VLAN the communication is still between the client IP address (**10.1.10.199**) and the virtual server (**10.1.10.25**).

On the internal VLAN the requests are from the client IP address to a back-end web server, however there are no responses from the web server.

19. Press the **Enter** key several times to move the log entries to the top of the window.

20. In the Configuration Utility, click **lorax_virtual**.

21. From the **Source Address Translation** list select **Auto Map**, and then click **Update**.



22. Use an incognito window to access **http://10.1.10.25**, and then close the window.

SNAT Auto Map ensures that responses to server request are always sent back to the BIG-IP system.

23. Examine the **tcpdump** window.

On the external VLAN the communication is still between the client IP address (**10.1.10.199**) and the virtual server (**10.1.10.25**).

On the internal VLAN the communication is now between the BIG-IP internal floating self IP address (**10.1.20.240**) and a back-end web server (**10.1.20.x**).

Task 2 – Create a SNAT for Internal Resources

1. Press the **Enter** key several times to move the log entries to the top of the window.

2. On the Windows server, change the default gateway to **10.1.20.240** (the BIG-IP internal floating self IP address).

3. On the Windows server, use Internet Explorer to access **www.f5.com**.

The request fails as the internal resource has no access to the WAN (or the Internet).

4. Close the page, then on the Windows desktop examine the **tcpdump** windows.

No requests are being sent to the Internet by the BIG-IP system on behalf of the internal resource.

5. In the Configuration Utility, open the **Local Traffic > Address Translation > SNAT List** page and click **Create**.

- Use the following information for the new SNAT, and then click **Finished**.

Form field	Value
Name	internal_snat
Translation	IP Address: 10.1.10.100
Origin	Address List
Address/Prefix Length	10.1.20.0/24 (Click Add)

- On the Windows server, use Internet Explorer to access **www.f5.com**.

The internal user now has public access to the internet using the SNAT IP address of **10.1.10.100**.

- On the Windows desktop, examine the **tcpdump** windows.

On the external VLAN the communication is between the SNAT IP address (**10.1.10.100**) and the Internet resources.

On the internal VLAN the communication is between the internal client (**10.1.20.251**) and the Internet resources.

- Close the **putty** sessions.

Task 3 – Use Profiles with a Virtual Server

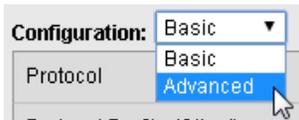
- Use a new tab to access **http://10.1.10.25**, and then select the links at the top of the page and examine the text on each page.

The pages make several references to the company name **Lorax Investments**. Lorax Investments has been acquired by **Smithy Financial**. Instead of updating all the web site code we'll use profiles on the BIG-IP system to update the web site.

- Close the tab.
- In the Configuration Utility, open the **Local Traffic > Profiles > Other > Stream** page and click **Create**.
- Use the following information for the profile, and then click **Finished**.

Form field	Value
Name	name_change
Source	Lorax Investments
Target	Smithy Financials

- Open the **Virtual Server List** page and click **lorax_virtual**.
- From the **Configuration** list select **Advanced**.



- From the **HTTP Profile** list select **http**.
- From the **Stream Profile** list select **name_change**.



9. In the **Acceleration** section, from the **HTTP Compression Profile** list select **httpcompression**.
10. From the **Web Acceleration Profile** list select **optimized-caching**, and then click **Update**.
11. Use an incognito window to access **http://10.1.10.25**, and then select the links at the top of the page.
Although the logo need to be updated, all the text on all pages now references **Smithy Financials**.

Task 4 – Work with Monitors

1. Edit the URL to **http://10.1.10.25/health_check.html**
We're going to use this web page to identify if the web server is up or down.
2. Close the health check page.
3. In the Configuration Utility, open the **Local Traffic > Monitors** page and click **Create**.
4. Use the following information for the monitor, and then click **Finished**.

Form field	Value
Name	lorax_monitor
Type	http
Interval	4
Timeout	13
Send String	GET /health_check.html\r\n
Receive String	Server_Up
Receive Disable String	Server_Down

5. Open the **Pool List** page and click **lorax_pool**.
6. Identify the current **Availability** status of the pool.
Unknown identifies when a pool or node doesn't have a configured monitor.
7. Add **lorax_monitor** to the **Active** list and click **Update**.
The **Availability** of the pool changes to **Available (Enabled)**.
8. Open the **Local Traffic > Nodes > Node List** page.
Notice that all the nodes currently display unknown.
9. Open the **Local Traffic > Nodes > Default Monitor** page.
10. Add **gateway_icmp** to the **Active** list and click **Update**.
11. Return to the **Nodes >ode List** page.
All nodes now display. This means that they are all sending **icmp** responses.
12. Open the **Local Traffic > Network Map** page and view the status for **lorax_virtual**.
The virtual server, pool, and all three pool members display available.
13. Use your mouse to hover over the pool members.
All three nodes also display available.

Sub-Task 1 – Take 10.1.20.41:80 Offline

1. On the Windows server go to **Start > Computer**, and then navigate to **C:\inetpub\wwwroot\lorax_public_site_41**.

This is the directory is used for pool member **10.1.20.41:80**. The **health_check.html** web page currently exists on this pool member.

2. Delete **health_check.html**.
3. Wait 13 seconds, and then in the Configuration Utility on the **Network Map** page click **Update Map**.



4. Use your mouse to hover over the pool members.

The first pool member is offline, and all three nodes display available.

Sub-Task 2 – Disable 10.1.20.42:80

1. On the Windows server navigate to **C:\inetpub\wwwroot\lorax_public_site_42**.
2. Right-click **health_check** and select **Open with > WordPad**.
3. In the **<p>** tag, edit the text to **Server_Down**, and then click **Save**.

This file is used by pool member **10.1.20.42:80**. This pool member will now match the disable string identified in the monitor.

4. Wait 13 seconds, and then in the Configuration Utility on the **Network Map** page click **Update Map**.

The second pool member is now disabled; however, the virtual server and pool still display available.

Sub-Task 3 – Take Node 10.1.20.43 Offline

1. On the Windows server, for **Local Area Connection 3** open the **Internet Protocol Version 4 (TCP/IPv4)** properties.
2. Click **Advanced**, and in the list of IP addresses scroll down to **10.1.20.43** and click **Remove**, then click **OK** three times and then click **Close**.
3. Wait 13 seconds, and then in the Configuration Utility on the **Network Map** page, click **Update Map**.
4. Use your mouse to hover over the pool members.



The virtual server and pool display disabled but available. Node **10.1.20.43** now displays offline, which causes pool member **10.1.20.43:80** to display offline.

Sub- Task 4 – Bring 10.1.20.42:80 Back Online

1. On the Windows server, in the **health_check** WordPad document, edit the text back to **Server_Up**, then click **Save**, and then close WordPad.
2. In the Configuration Utility on the **Network Map** page click **Update Map**.
Because pool member **10.1.20.42:80** is available, the virtual server and pool once again display available.
3. Use an incognito window to access **http://10.1.10.25**.
The page displays, with all page elements coming from **10.1.20.42:80**.
4. Close the page.

1.2.3 Lab 3: Use SSL Offload, Best Practices, and iApps

In this lab you will create an HTTPS web application and use the BIG-IP SSL offload feature to free up CPU resources from the web servers. You'll update the BIG-IP configuration by including some best practices. Finally you'll re-create the HTTPS web application by using BIG-IP iApps.

Task 1 – Use SSL Offload

1. In the Configuration Utility, open the **Pool List** page and click **Create**.
2. Use the following information for the new pool, and then click **Finished**.

Form field	Value
Name	https_pool
Health Monitors	https_443
New Members	Node List: node1 (10.1.20.11) Service Port: 443 (Click Add)
	Node List: node2 (10.1.20.12) Service Port: 443 (Click Add)
	Node List: node3 (10.1.20.13) Service Port: 443 (Click Add)

3. Open the **Virtual Server List** page and click **Create**.
4. Use the following information for the new virtual server, and then click **Finished**.

Form field	Value
Name	https_virtual
Destination Address/ Mask	10.1.10.20
Service Port	443
HTTP Profile	http
Acceleration > HTTP Compression Profile	httpcompression
Resources > Default Pool	https_pool

5. Use a new tab to access **https://10.1.10.20**.

The page fails to load. At this point the BIG-IP system is simply forwarding SSL requests to the downstream HTTPS web server without decrypting them first. This prevents the BIG-IP system from performing any HTTP functions (such as HTTP compression) which is why the page fails to load.

6. In the Configuration Utility, on the **Virtual Server List** page click **https_virtual**, and then for **SSL Profile (Client)** move **clientssl** to the **Selected** list.
7. For **SSL Profile (Server)** move **serverssl-insecure-compatible** to the **Selected** list, and then click **Update**.
8. Reload the **https://10.1.10.20** page.
The page displays as expected.
9. Identify the protocol used in the URL, and the port used on the downstream server (in the **Pool member address/port entry**).
Although the BIG-IP system is now decrypting requests, it is re-encrypting before sending to the HTTPS web servers, which means they must perform the CPU-intensive task of decrypting requests and encrypting responses.
10. In the Configuration Utility, on the **https_virtual** page, for **SSL Profile (Server)** move **serverssl-insecure-compatible** back to the **Available** list, and then click **Update**.
11. Open the **Resources** page.



12. From the **Default Pool** list select **http_pool**, and then click **Update**.
13. Reload the **https://10.1.10.20** page.
14. Identify the protocol used in the URL, and the port used on the downstream server (in the **Pool member address/port entry**).
The BIG-IP system is now performing SSL offload, sending all downstream requests using port 80. This means that the web servers no longer need to perform the CPU-intensive task of decrypting requests and encrypting responses.
15. Close the tab.

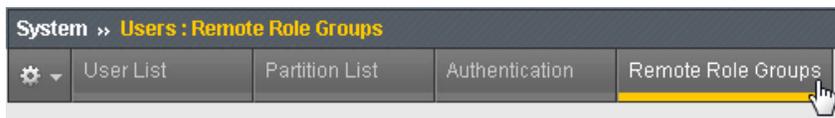
Task 2 – Configure BIG-IP Best Practices

1. Close the Configuration Utility, then open Internet Explorer and access **https://10.1.10.240**.
Currently the BIG-IP system can be accessed by the outside world using the external self IP address, which is not recommended.
2. Log into the BIG-IP system, and then open the **Network > Self IPs** page and click **10.1.10.240**.
3. From the **Port Lockdown** list select **Allow None**, and then click **Update**.
4. Return to the **Self IPs** page.
Why are you now unable to access the BIG-IP system?
5. Close Internet Explorer, then open Chrome and access **https://10.1.1.245** and then log into the BIG-IP system as **admin / admin**.
It is not recommended to use the default admin account.

6. Open the **System > Users > Authentication** page and click **Change**.
7. From the **User Directory** list select **Remote – Active Directory**.
8. Use the following information, and then click **Finished**.

Form field	Value
Host	10.1.20.251
Remote Directory Tree	DC=f5demo,DC=com
Bind DN	CN=Service Account,OU=Corporate,DC=f5demo,DC=com
Bind Password	password
Check Member Attribute	Enabled (selected)
Role	Guest

9. Open the **Remote Role Groups** page and click **Create**.



10. Use the following information, and then click **Finished**.

Form field	Value
Group Name	F5Admins
Line Order	10
Attribute String	memberOf=CN=loraxadmins,CN=Users,DC=f5demo,DC=com
Assigned Role	Administrator
Terminal Access	tmsh

11. Create another role group using the following information, and then click **Finished**.

Form field	Value
Group Name	F5ResourceAdmins
Line Order	15
Attribute String	memberOf=CN=resadmins,CN=Users,DC=f5demo,DC=com
Assigned Role	Resource Administrator
Terminal Access	Disabled

12. Create another role group using the following information, and then click **Finished**.

Form field	Value
Group Name	F5Operators
Line Order	20
Attribute String	memberOf=CN=operators,CN=Users,DC=f5demo,DC=com
Assigned Role	Operator
Terminal Access	Disabled

13. Open the **System > Users > User List** page.
14. Select the **admin** account and change the password to **admin-pass** and then click **Update**.
15. Log in as **bigip_operator / password**.

16. Notice the user's role at the top of the page.

Hostname: bigipA.f5demo.com	Date: Jun 6, 2017	User: bigip_operator
IP Address: 10.1.1.245	Time: 1:05 PM (PDT)	Role: Operator

17. Open the **Virtual Server List** page and examine the **Create** button.

This user can view all virtual servers and other BIG-IP system objects, but can't create or update objects.

18. Log out and then log back in as **bigip_ra / password**.

19. Notice the user's role at the top of the page.

20. Open the **Virtual Server List** page.

This user can see and manage all virtual servers.

21. Open the **System > Users > Authentication** page and examine the **Change** button.

22. Log out and then log back in as **bigip_admin / admin**. (NOTE: You are intentionally logging in with the wrong password.)

23. Log in as **bigip_admin / password**.

24. Open the **System > Logs > Audit > List** page, and then sort the list by the **Time** column in descending order.

Timestamp	User Name	Transaction	Event
Mon May 22 12:41:02 PDT 2017		0-0	pid=1496t cmd_data

25. Examine the login and logout details for the three users.

You can see when each user logged in, logged out, and failed to login correctly.

Task 3 – Re-create the Application using iApp

1. Open the **Virtual Server List** page, then select the **http_virtual** and **https_virtual** checkboxes, and then click **Delete** twice.
2. Open the **Pool List** page, then select the **http_pool** and **https_pool** checkboxes, and then click **Delete** twice.
3. Open the **Node List** page, then select the **node1**, **node2**, and **node3** checkboxes, and then click **Delete** twice.
4. Open the **iApps > Application Services > Applications** page and click **Create**.
5. Create an application using the following information, and then click **Finished**.

Form field	Value
User Name	https_app
Template	f5.http
Network > Do you want to use the latest TCP profiles?	Yes
SSL Encryption > How should the BIG-IP system handle SSL traffic?	Terminate SSL from clients, plaintext to servers
Virtual Server and Pools > What IP address do you want to use	10.1.10.20
FQDN	www.f5demo.com
Web servers	10.1.20.11: 80 (Click Add) 10.1.20.12: 80 (Click Add) 10.1.20.13: 80
Application Health > What HTTP URI	/index.php
Expected Response	Welcome

6. Open the **Virtual Server List** page.

iApp created two virtual servers for the web application. The port 80 virtual server is used to redirect requests to the port 443 virtual server.

7. Open the **Pool List** page.

iApp created a pool with three pool members and a monitor attached (which you can identify by it being identified as available).

8. Open the **Monitors** page and click **https_app_http_monitor**.

iApp created the custom HTTP monitor for the web application.

9. Use a new tab to access **http://10.1.10.20**.

Notice that the request is redirected to **https**. The requests are sent to the web servers on port **80**, identifying that SSL offload is taking place.

10. Close the tab.

1.2.4 Lab 4: Configure High-Availability

In this lab you will set up a high availability pair using two BIG-IP systems. You'll then test failover between the two HA members.

Task 1 – Set up a Device Group

1. Open a new tab and click the **BIGIP_B** bookmark and then log into the BIG-IP system.
2. Note the status of both BIG-IP systems.



On bigipA.f5demo.com

1. Open the **Device Management > Device Trust > Device Trust Members** page and click **Add**.
2. In the **Device IP Address** field, type **10.1.1.246**.

3. Enter **admin** for the **Administrator Username** and **Administrator Password**.
4. Click **Retrieve Device Information**.
5. Click **Device Certificate Matches**.
6. Verify that the **Name** value is **bigipB.f5demo.com** and click **Add Device**.

On bigipB.f5demo.com

1. Open the **Device Management > Device Trust > Device Trust Members** page.
This BIG-IP system sees **bigipA.f5demo.com** as a trusted peer.
2. Note the new status of both BIG-IP systems.



On bigipA.f5demo.com

1. Open the **Device Management > Device Groups** page and click **Create**. (ENSURE you are on **bigipA.f5demo.com**.)
2. Create a device group using the following information, and then click **Finished**.

Form field	Value
Name	lorax_device_group
Group Type	Sync-Failover
Members	bigipA.f5demo.com bigipB.f5demo.com
Sync Type	Manual with Incremental Sync

3. Note the new status of **bigipA.f5demo.com**.
4. Click **Awaiting Initial Sync**.
This directs you to the **Device Management > Overview** page.
5. In the **Devices** section, leave the **bigipA.f5demo.com (Self)** option selected.
6. For **Sync Options** leave **Push the selected device configuration to the group** selected and click **Sync**.
7. Note the status of **bigipA.f5demo.com**.
Both BIG-IP systems are now in sync with each other.

On bigipB.f5demo.com

1. Attempt to log in as **admin / admin**.
Why do you think your login failed?
2. Log in as **bigip_admin / password**.
3. Examine the **Virtual Server List** and **Pool List** pages.
4. Open the **iApps > Application Services > Applications** page and click **https_app**.
5. Open the **Reconfigure** page.
6. Add a new pool member for **10.1.20.14:80**, and then click **Finished**.
7. Note the updated status of **bigipB.f5demo.com**.

8. Click **Changes Pending**.
9. In the **Devices** section leave the default options selected and click **Sync**.

On **bigipA.f5demo.com**

1. Examine the **Pool List** page.
The **https_app_pool** now contains 4 members.

Task 2 – Test Failover

On **bigipA.f5demo.com**

1. Navigate to **Local Traffic > Virtual Servers** and right-click on **Statistics** and open the page in a new tab.
2. In the statistics tab reset the statistics for all three virtual servers.

On **bigipB.f5demo.com**

1. Navigate to **Local Traffic > Virtual Servers** and right-click on **Statistics** and open the page in a new tab.
2. Use a new tab to access **http://10.1.10.25**.
3. Use both statistics tabs (click **Refresh**) to identify which BIG-IP system processed the incoming request.
4. In the Configuration Utility tab for **bigipB.f5demo.com**, open the **Device Management > Traffic Groups** page and click **traffic-group-1**.

For this traffic group the **Current Device** is **bigipB.f5demo.com**.

5. Click **Force to Standby** twice, and then view the value in the **Active Device** column.
6. In the Lorax Investments page, edit the URL to **http://10.1.10.20**, then use both statistics tabs to identify which BIG-IP system processed the incoming request.

On **bigipA.f5demo.com**

1. In the Configuration Utility tab, open the **Device Management > Devices** page and click **bigipA.f5demo.com (Self)**.
2. Click **Force Offline** and then **OK**.

On **bigipB.f5demo.com**

1. Note the status of **bigipB.f5demo.com**.

On **bigipA.f5demo.com**

1. On the **Devices** page click **Release Offline** and then **OK**.

On **bigipB.f5demo.com**

1. Note the status of **bigipB.f5demo.com**.

When **bigipA.f5demo.com** comes back online it doesn't become the active device.

Task 3 – Create an Active / Active Pair

On **bigipA.f5demo.com**

1. Open the **Device Management > Traffic Groups** page and click **Create**.

2. Create a traffic group using the following information, and then click **Create Traffic Group**.

Form field	Value
Name	traffic-group-2
Failover Method	Preferred Device Order
Preferred Order	bigipA.f5demo.com bigipB.f5demo.com

3. Open the **Local Traffic > Virtual Servers > Virtual Address List** page and click **10.1.10.25**.
4. From the **Traffic Group** list select **traffic-group-2 (floating)**, and then click **Update**.

Traffic Group	<input type="checkbox"/> Inherit traffic group from current partition / path
Availability	<input type="radio"/> None <input checked="" type="radio"/> Common
State	<input type="checkbox"/> Enabled <input checked="" type="checkbox"/> Disabled
Auto Delete	<input checked="" type="checkbox"/>

traffic-group-1 (floating) ▼

traffic-group-1 (floating)

traffic-group-2 (floating)

traffic-group-local-only (non-floating)

5. Click **Changes Pending**.
 6. Leave the default options selected and click **Sync**.
 7. Note the status of both BIG-IP systems.
You now have an active / active pair.
 8. Reset both statistics pages.
 9. Access **https ://10.1.10.20** and identify which BIG-IP processed the request.
 10. Access **http://10.1.10.25** and identify which BIG-IP is processed the request.
- That concludes the hands-on exercises for the Introduction to ADC Deployments with LTM lab session.

Class 2: Building the F5 Fabric

Welcome to the Building the F5 Fabric lab session at F5 Agility 2018. These labs are intended to guide you through creating basic Device Service Cluster (DSC) using the GUI, TMSH and in an automated fashion.

This guide is intended to complement lecture material provided during the Agility event as well as a reference guide that can be referred to after the class.

2.1 DSC Lab Network Setup

In the interest of focusing as much time as possible configuring your application delivery controller, we have provided some resources and basic setup ahead of time. These are:

- Cloud-based lab environment complete with a Windows workstation, three virtual BIG-IPs (VE), and a back-end banking application running on a Linux web server.
- The virtual BIG-IP has been pre-licensed

If you wish to replicate these labs in your office you will need to perform these steps accordingly.

To access the lab environment, you will require a web browser and Remote Desktop Protocol (RDP) client software. The web browser will be used to access the lab training portal. The RDP client will be used to connect to a Windows workstation, where you will be able to access the BIG-IPs (HTTPS, SSH).

Your class instructor will provide additional lab access details.

2.1.1 Login info for lab jump box

Use RDP to connect to the DNS host provided to you in the Agility Portal. The user and password are:

Username: `student`

Password: `agility`

Bigip1: `10.128.1.245`

Bigip2: `10.128.1.246`

Bigip3: `10.128.1.247`

BIG-IP Usernames	BIG-IP Passwords
admin	admin
root	default

Timing for DSC Labs

The time it takes to perform each lab varies and is mostly dependent on accurately completing steps. This can never be accurately predicted but we strived to derive an estimate among several people each having a different level of experience. Below is an estimate of how long it will take for each lab:

LAB Name (Description)	Time Allocated
Lab 1 – Active/Active ScaleN Clustering Exercise	55 minutes
Lab 2 – Sync-Only Groups Exercise	30 minutes
Lab 3 – Device Service Cluster Automation	60 minutes

2.2 Module 1: BIG-IP Device Service Cluster Configuration

In this module you will learn the basics of configuring BIG-IP DSC

2.2.1 Lab 1: Active/Active ScaleN Clustering Exercise

Configuration of active/active device cluster utilizing multiple traffic groups.

Objective:

- Create a two node cluster, with a traffic groups active on both BIG-IPs.
- Create two standard virtual servers. Both virtual servers must be active on a different BIG-IP in the cluster.

Prerequisites and Notes:

To save some time, both VE images have been provided for you. They are licensed, and have basic network connectivity established on the three VLANs listed below.

- 1.1 = External Network Interface (WAN Side)
- 1.2 = Internal network interface (LAN Side)
- 1.3 = High Availability Network Interface *not yet configured

The following is the IP addressing scheme for every student.

- External: 10.128.10.0/24
- Internal: 10.128.20.0/24
- HA: 192.168.1.0/24

Lab Requirements:

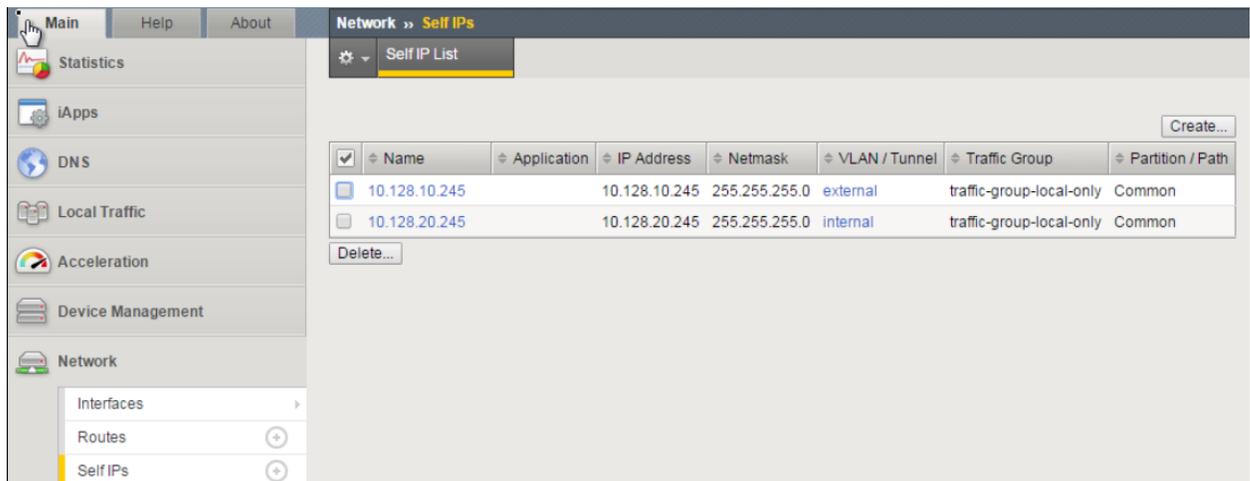
- Configure Floating Self-Ips
- Create HA VLAN
- Create additional Traffic Group

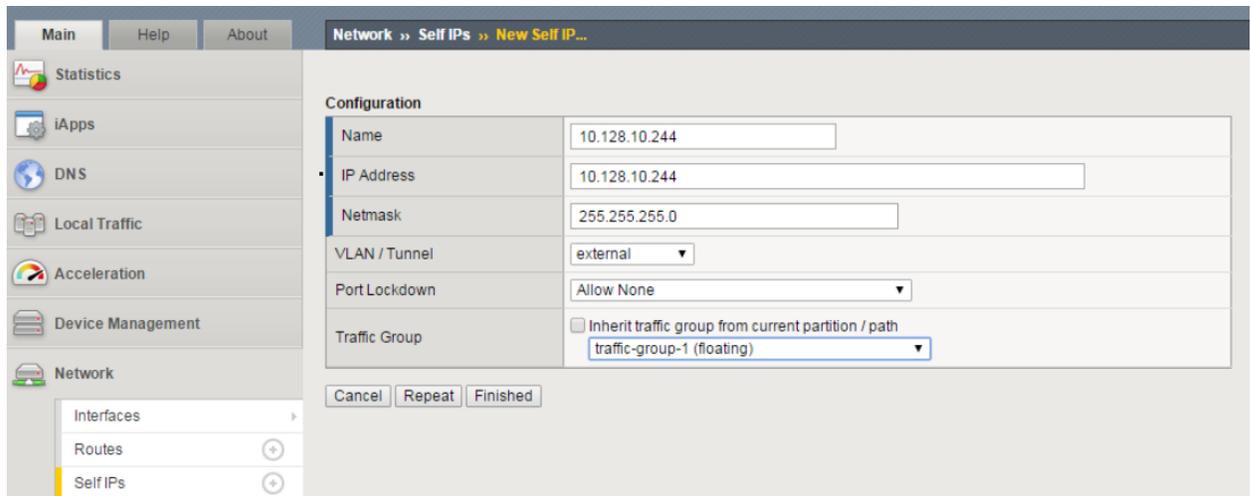
- Build a Device Service Cluster
- Create a pool of web servers
- Create three virtual servers
- Test failover

TASK 1 – Configure Floating Self IP's

Create Floating Self IPs

1. Open Chrome on the Windows jump box. Connect to bigip1.
 - `https://10.128.1.245` (accept certificate error)
 - Username: admin
 - Password: admin
2. Go to *Network* -> *Self Ips* -> *Create*
3. Create a Floating Self IP for External VLAN using the following values
 - Name/Address: 10.128.10.244
 - Netmask: 255.255.255.0
 - Port Lockdown: Allow None
 - Traffic group: traffic-group-1 (floating)





4. Click **Finished**

5. Create a Floating Self IP for Internal VLAN using the following values

- Name/Address: 10.128.20.244
- Netmask: 255.255.255.0
- Port Lockdown: None
- Traffic Group: traffic-group-1 (floating)

6. Click **Finished**

Note: Although Self IP's (non-floating) must be created on both F5's in a cluster (which was already completed for you), the creation of Floating Self IP's only needs to occur on a single F5, as they will "float" to the other F5 after HA has been successfully completed.

TASK 2 – Create HA VLAN

Create HA VLAN

1. Go to **Network -> VLANs -> VLAN List -> Create**
2. Create a VLAN using the following values:
 - Name: ha
 - Interface: 1.3 (untagged)

General Properties	
Name	ha
Description	
Tag	
Resources	
Interfaces	Interface: 1.1 ▼ Tagging: Untagged ▼ <input type="button" value="Add"/>
	<div style="border: 1px solid gray; padding: 2px;">1.3 (untagged)</div> <input type="button" value="Edit"/> <input type="button" value="Delete"/>

3. Click **Finished**

4. Repeat the above steps to create the ha vlan on **bigip2**

Create one Self IP on **bigip1** and one on **bigip2** for HA VLAN*

On bigip1:

1. Go to **Network -> Self IP's -> Create**

2. Create a Self IP using the following values:

- Name: 192.168.1.10
- IP Address: 192.168.1.10
- Netmask: 255.255.255.0
- VLAN: ha
- Port Lockdown: Allow Default
- Traffic Group: traffic-group-local-only (non-floating)

Configuration	
Name	192.168.1.10
IP Address	192.168.1.10
Netmask	255.255.255.0
VLAN / Tunnel	ha
Port Lockdown	Allow Default
Traffic Group	<input type="checkbox"/> Inherit traffic group from current partition / path traffic-group-local-only (non-floating)

3. Click **Finished**

On bigip2:

1. Go to **Network -> Self IP's -> Create**

2. Create a Self IP using the following values:

- Name: 192.168.1.11
- IP Address: 192.168.1.11
- Netmask: 255.255.255.0
- VLAN: ha
- Port Lockdown: Allow Default
- Traffic Group: traffic-group-local-only (non-floating)

Note: It is critical the Self IP on each big-ip be set to Allow Default for the ha VLAN. This is because failover communication will be configured to use those IP's in an upcoming task.

TASK 3 – Create Traffic Groups and Additional Floating Self IP's

Create 2 new traffic groups on bigip1

1. Go to **Device Management -> Traffic Groups -> Create**

2. Create a new Traffic Group using the following values:

- Name: lab_traffic_group_01
- MAC Masquerade: 02:00:00:01:00:00
- Failover Method = Failover using Preferred Device Order and then Load Aware

3. Click **Create Traffic Group**

4. Go to **Device Management -> Traffic Groups -> Create**

5. Create a new Traffic Group using the following values:

- Name: lab_traffic_group_02
- MAC Masquerade: 02:00:00:02:00:00
- Failover Method = Failover using Preferred Device Order and then Load Aware

6. Click **Create Traffic Group**

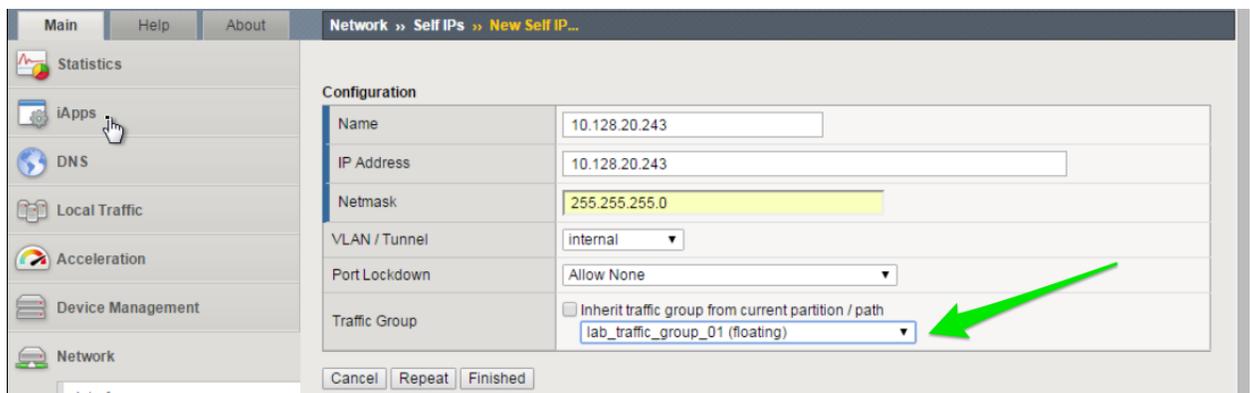
Create additional Floating Self IP's for the Internal VLAN for each traffic group

Create a Floating Self IP for Internal VLAN in Traffic Group 1

1. Go to **Network -> Self IP's -> Create**

2. Create a Floating Self IP using the following values:

- Name: 10.128.20.243
- IP Address: 10.128.20.243
- Netmask: 255.255.255.0
- VLAN: internal
- Port Lockdown: Allow None
- Traffic Group: lab_traffic_group_01 (floating)

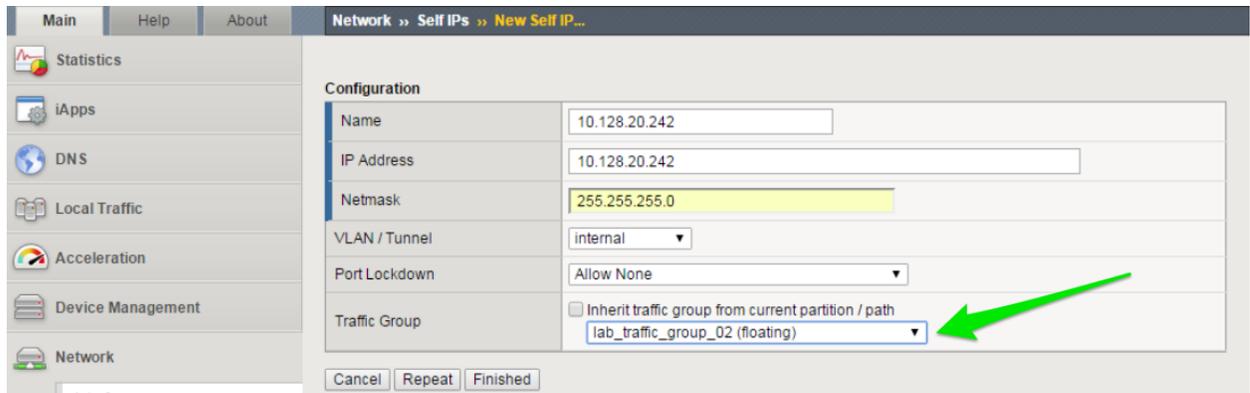


Create a Floating Self IP for Internal VLAN in Traffic Group 2

1. Go to **Network -> Self IP's -> Create**

2. Create a Floating Self IP using the following values:

- Name: 10.128.20.242
- IP Address: 10.128.20.242
- Netmask: 255.255.255.0
- VLAN: internal
- Port Lockdown: Allow None
- Traffic Group: lab_traffic_group_02 (floating)



TASK 4 – Configure Device Connectivity for HA Communication

Configure Local HA Address

1. Go to **Device Management -> Devices -> select your device (bigip1)**
2. Choose ConfigSync tab
3. Under Local Address
4. Choose the **HA address** and click **Update**

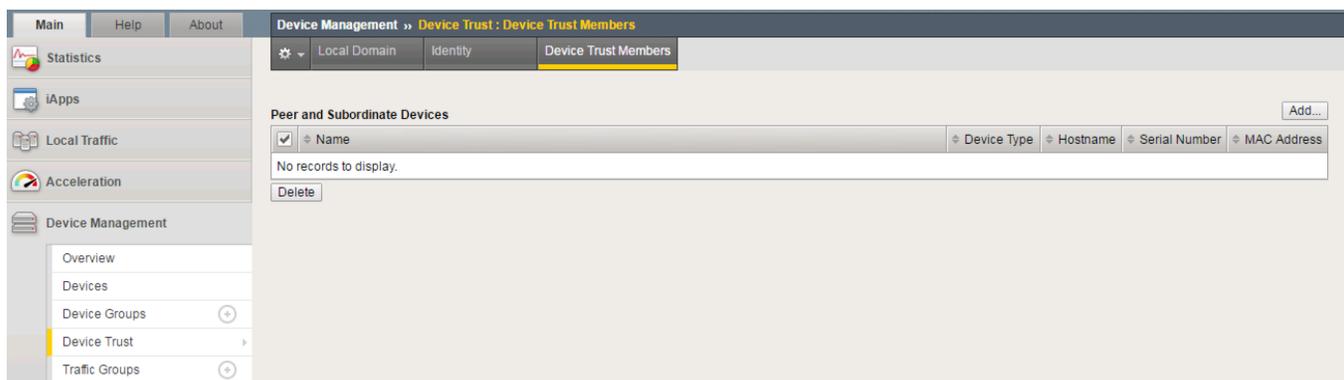
Configure Network Failover

1. From the Failover Network tab, choose Add
2. Add both the Management address as well as the HA VLAN address

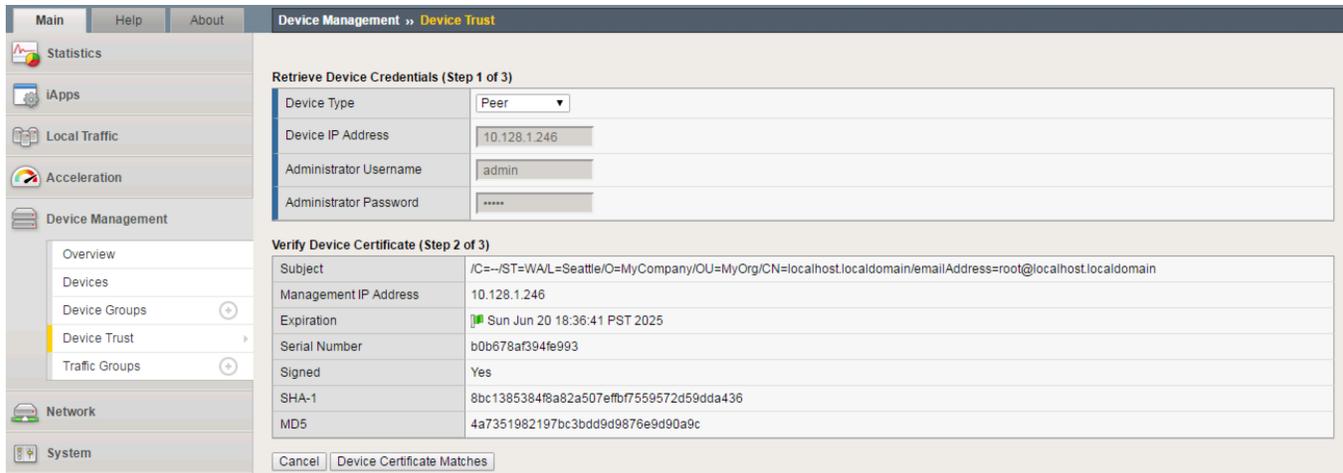
Important: Perform both of the above steps on **bigip2** as well.

On bigip1, add bigip2 to Peer List

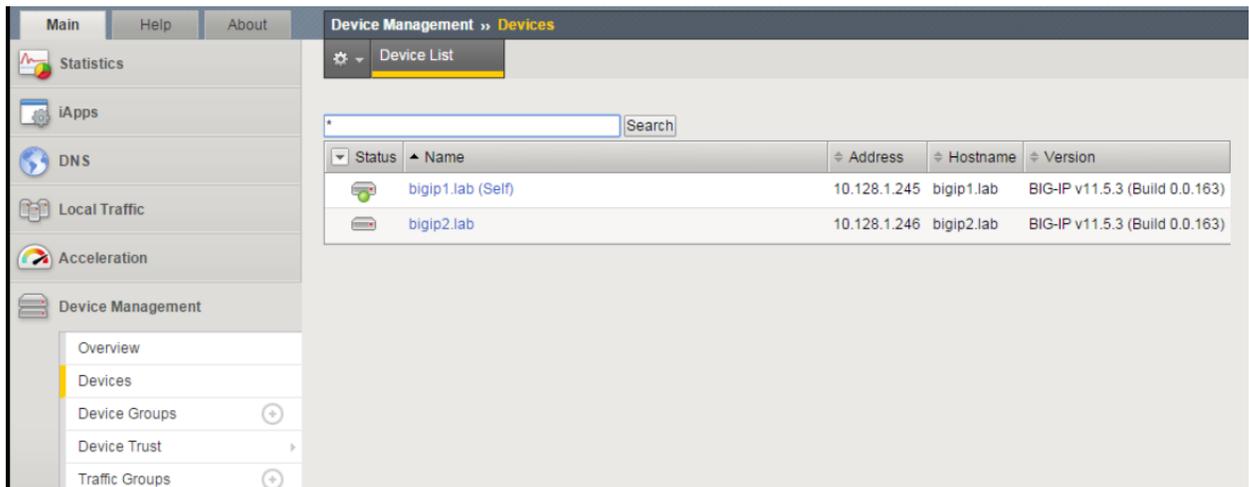
1. Go to *Device Management -> Device Trust -> Device Trust Members*



2. Click Add
3. Add the second F5's Management Address
4. Click Retrieve Device Information

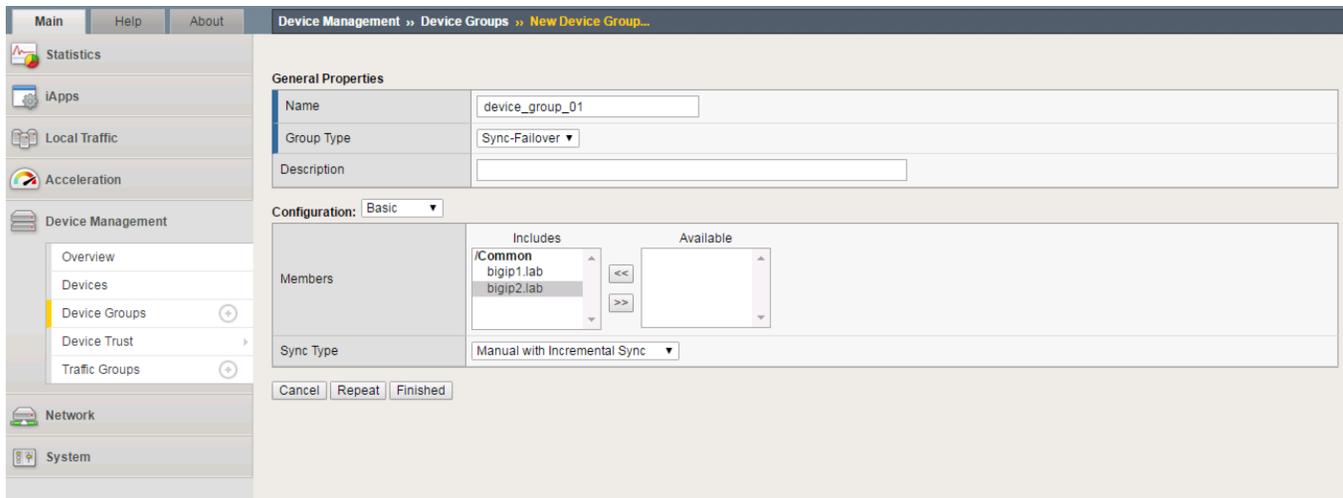


5. Click **Device Certificate Matches**
6. Click **Add Device**
7. Click on **Device Management -> Devices**. You will now see both F5's.



Create Device Group

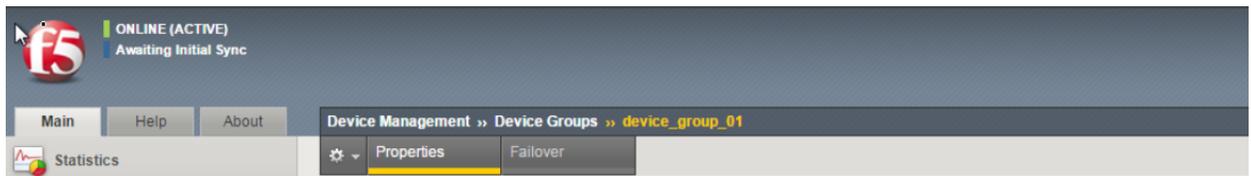
1. Go to **Device Management -> Device Groups -> Create**
2. Create a Device Group using the following values:
 - Name: device_group_01
 - Group Type: Sync-Failover
 - Add both devices as Members



3. Click **Finished**

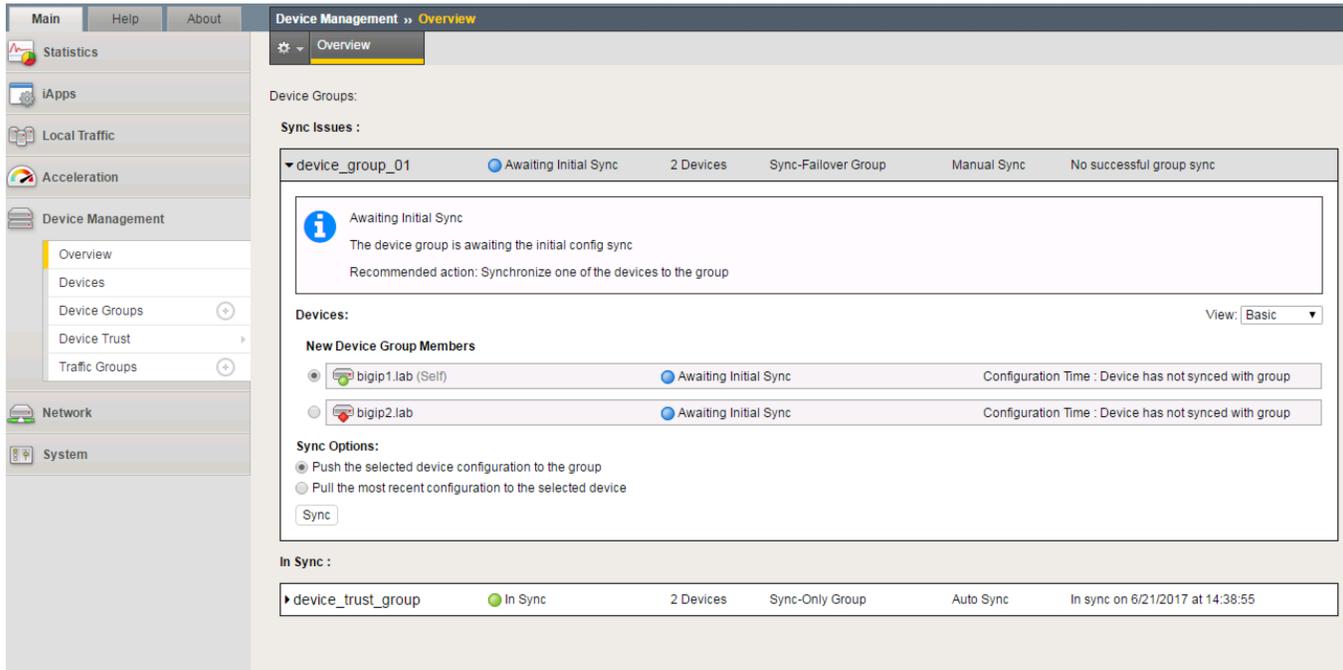
Perform the initial configuration sync

1. Click **Awaiting Initial Sync** in upper left corner of your screen



2. Select bigip1.lab (Self)

3. Choose Push the selected device configuration to the group



4. Click **Sync**. You will notice the change in the upper left status to “In Sync”

TASK 5 – Create Web Server Pool and Virtual Servers

Create a web server pool

1. Go to **Local Traffic -> Pools -> Create**
2. Create a Pool using the following values:
 - Name: lamp_pool
 - Health Monitor: http
 - Add first Node: - Node Name: 10.128.20.11 - Address: 10.128.20.11 - Port: 80
 - Add second Node: - Node Name: 10.128.20.12 - Address: 10.128.20.12 - Port 80
 - Add third Node: - Node Name: 10.128.20.13 - Address: 10.128.20.13 - Port 80

The screenshot shows the FortiGate configuration interface for creating a new pool. The left sidebar shows the navigation menu with 'Local Traffic' selected and 'Pools' highlighted. The main configuration area is titled 'Configuration: Pools' and contains the following fields and options:

- Name:** lamp_pool
- Description:** (empty)
- Health Monitors:** A list of health monitors is shown. The 'Active' monitor is '/Common/http' and the 'Available' monitor is '/Common/gateway_icmp'. Other available monitors include 'http_head_f5', 'https', and 'https_443'.
- Resources:**
 - Load Balancing Method:** Round Robin
 - Priority Group Activation:** Disabled
 - New Members:** A section for adding new nodes. The 'New Node' radio button is selected. The fields are: Node Name: 10.128.20.252 (Optional), Address: 10.128.20.252, Service Port: 80, and HTTP. An 'Add' button is present. Below the fields, a list of members is shown: R:1 P:0 C:0 10.128.20.252 10.128.20.252 :80. 'Edit' and 'Delete' buttons are also visible.

At the bottom of the configuration area, there are buttons for 'Cancel', 'Repeat', and 'Finished'.

3. Click **Finished**.

Create first Virtual Server

1. Go to **Local Traffic -> Virtual Servers -> Create**
2. Create a Virtual Server using the following values:
 - Name: http_vs_01
 - Destination (Host): 10.128.10.230
 - Port 80
 - Default Pool: lamp_pool
 - Source Address Translation (SNAT): Auto Map

3. Click **Finished**

Create second Virtual Server

1. Go to **Local Traffic -> Virtual Servers -> Create**
2. Create a Virtual Server using the following values:

- Name: http_vs_02
- Destination (Host): 10.128.10.231
- Port 80
- Default Pool: lamp_pool
- Source Address Translation (SNAT): Auto Map

3. Click **Finished**

4. You should now have two virtual servers created

✓	Status	Name	Application	Destination	Service Port	Type	Resources	Partition / Path
<input type="checkbox"/>	●	http_vs_01		10.128.10.230	80 (HTTP)	Standard	Edit...	Common
<input type="checkbox"/>	●	http_vs_02		10.128.10.231	80 (HTTP)	Standard	Edit...	Common

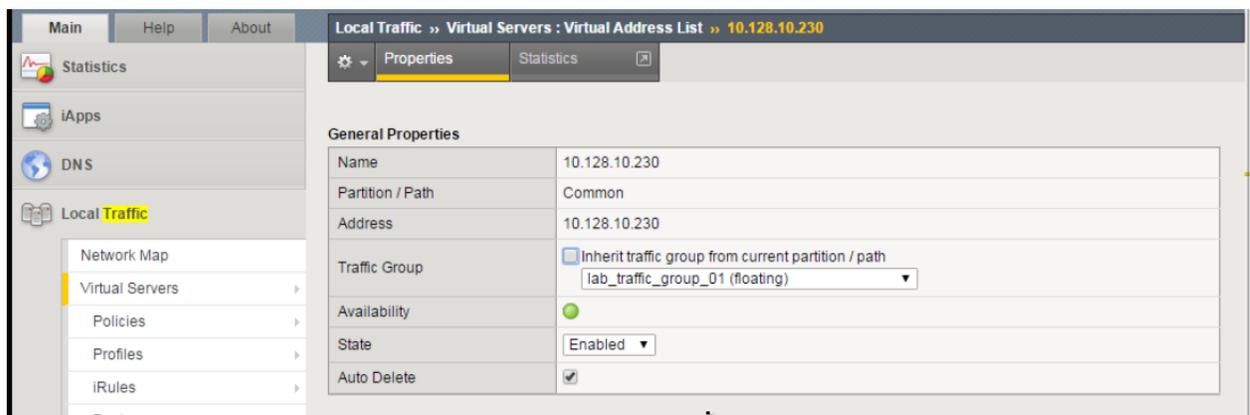
Buttons: Enable, Disable, Delete...

5. Sync your configuration.

TASK 6 – Configure Virtual Servers for Different Traffic Groups and Simulate Failover

Reconfigure the new Virtual Servers so that they reside in the 2 new traffic groups.

1. Go to **Local Traffic -> Virtual Servers -> Virtual Address List**
2. Click on 10.128.10.230
3. Change the Traffic Group to lab_traffic_group_01.



4. Click **Update**
5. Perform the same procedure for 10.128.10.231, but place in lab_traffic_group_02.
6. Sync Configuration

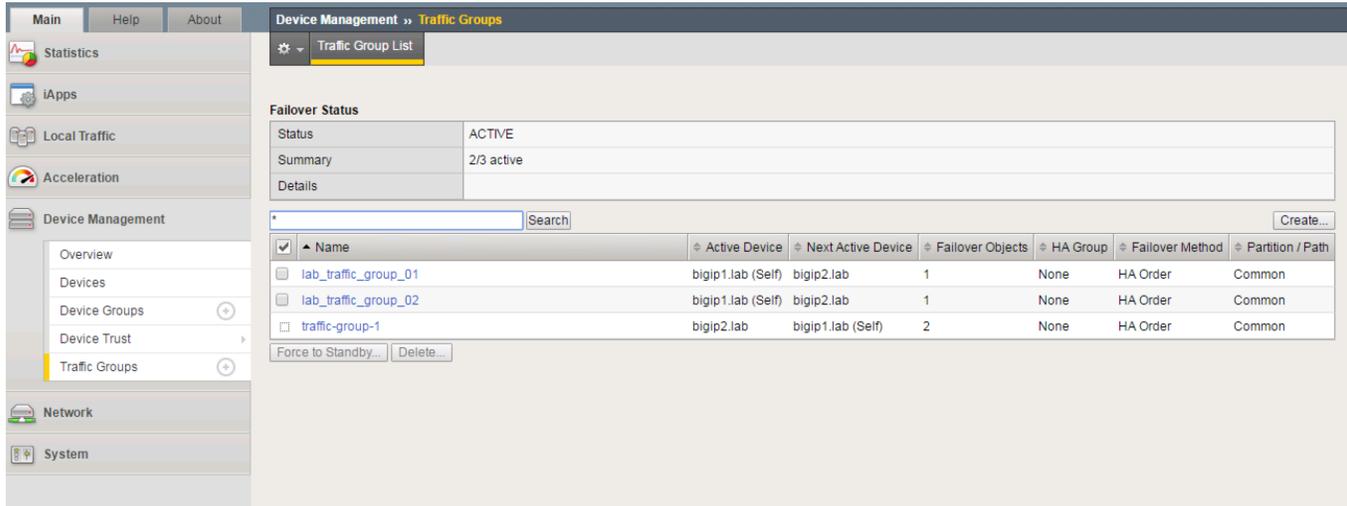
Check which objects are in each Traffic Group

1. Go to **Device Management -> Traffic Groups**
2. Select a group, and choose Failover Objects.

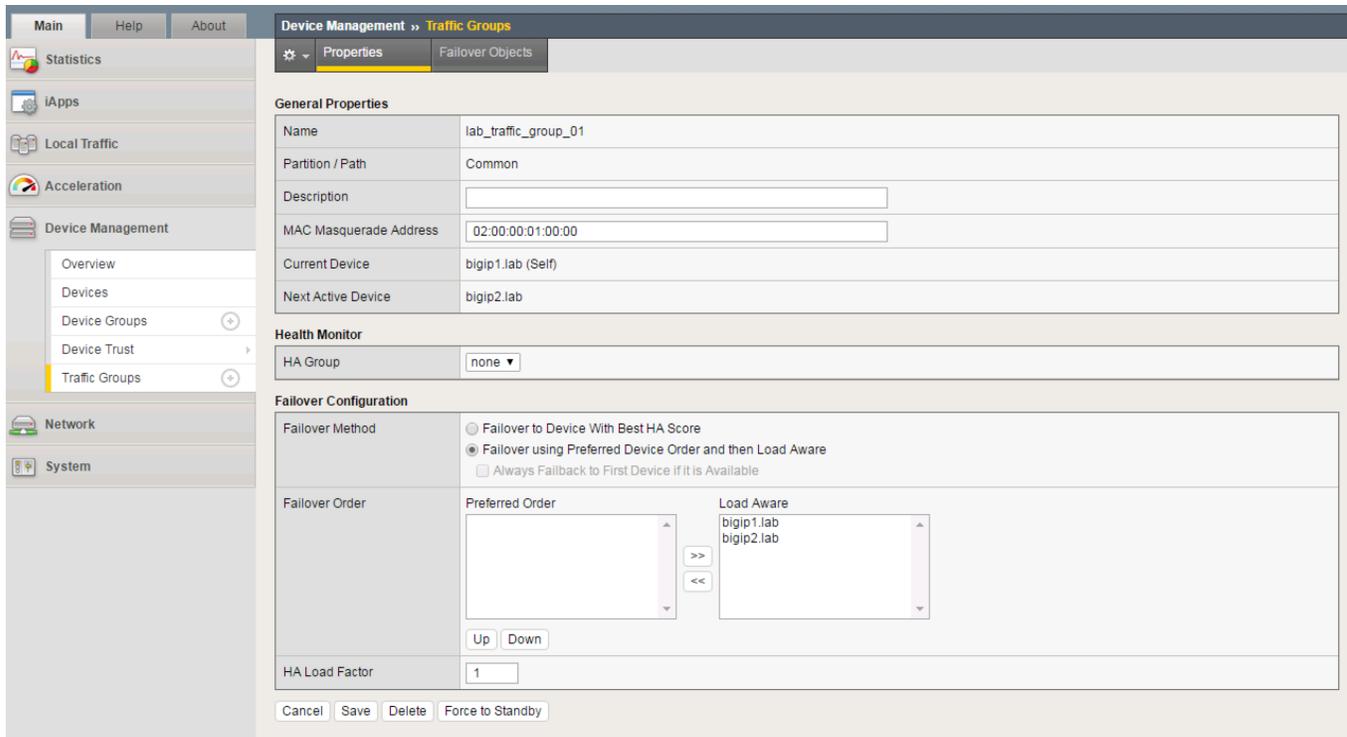
Simulate a failover within the Active/Active cluster

1. Go to **Device Management -> Traffic Groups**

- Take note of which devices are currently servicing each Traffic Group. If 1 device is servicing a particular traffic group, and the other device is servicing another traffic group, you will see that both **bigip1** and **bigip2** list their status as **ACTIVE** in the GUI



- From **bigip1**, choose any Traffic Group which is currently active on it.



- Select **Force to Standby** to manually fail this traffic group over to **bigip2**

Note: This is failing over the traffic group only, not the device. When all traffic groups have been failed-over to bigip2, bigip1 will be **STANDBY**, and bigip2 will be **ACTIVE**.

Why does bigip1 display as STANDBY when we never failed it over at the device level?

Are all VIP's still accessible?

What are some practical, real-world examples for what we just configured?

2.2.2 Lab 2: Sync Only exercise

Objective: Add a sync only device group. You have already configured two VE's in an Active/Active Configuration with two traffic groups. Add a 3rd VE. Create new Sync-Only group, and new Partition that will leverage new Sync-Only group. Create new SSL profile which will sync to all devices.

Prerequisites and Notes

Have at least 2 VE's in an Active/Active Failover Configuration.

Note that for this exercise we will use three network interfaces (as in in the previous failover exercise)

1.1 = External Network Interface (Wan Side)

1.2 = Internal network interface (LAN Side)

1.3 = High Availability Network Interface

TASK 1 – Add HA Self IP to bigip3.lab

1. Go to **Network -> Self Ips -> Create**

- Name: 192.168.1.12
- IP Address: 192.168.1.12
- Netmask: 255.255.255.0
- VLAN: HA
- Port Lockdown: Allow Default

2. Click **Finished**

The screenshot shows the configuration page for a new self IP. The breadcrumb path is "Network >> Self IPs >> New Self IP...". The configuration fields are as follows:

Configuration	
Name	192.168.1.12
IP Address	192.168.1.12
Netmask	255.255.255.0
VLAN / Tunnel	HA
Port Lockdown	Allow Default
Traffic Group	<input checked="" type="checkbox"/> Inherit traffic group from current partition / path traffic-group-local-only (non-floating)

Buttons at the bottom: Cancel, Repeat, Finished

TASK 2 – Update Config-Sync Properties on bigip3.lab

1. Go to **Device Management -> Devices**

2. Click on then `bigip3.lab` (Self) link
3. Go to **ConfigSync** tab
4. Select `192.168.1.12` (HA) in Local Address dropdown
5. Click **Update**

Device Management >> Devices >> bigip3.lab

Properties ConfigSync Failover Network Mirroring

ConfigSync Configuration

Local Address 192.168.1.12 (HA) ▼

Update

TASK 3 – Add bigip3 to peer list on bigip1

1. Go to **Device Management -> Device Trust -> Device Trust Members -> Add**

Device Management >> Device Trust

Retrieve Device Credentials (Step 1 of 3)

Device Type	Peer ▼
Device IP Address	10.128.1.247
Administrator Username	admin
Administrator Password

Cancel Retrieve Device Information

2. Click on **Retrieve Device Information**
3. Click **Device Certificate Matches**
4. Click **Add Device**

Device Management » Device Trust

Retrieve Device Credentials (Step 1 of 3)

Device Type	Peer
Device IP Address	10.128.1.247
Administrator Username	admin
Administrator Password	*****

Verify Device Certificate (Step 2 of 3)

Subject	/C=--/ST=WAL=Seattle/O=MyCompany/OU=MyOrg/CN=localhost.localdomain/emailAddress=root@localhost.localdomain
Management IP Address	10.128.1.247
Expiration	Sun Jun 20 18:36:41 PST 2025
Serial Number	b0b678af394fe993
Signed	Yes
SHA-1	8bc1385384f8a82a507effbf7559572d59dda436
MD5	4a7351982197bc3bdd9d9876e9d90a9c

Add Device (Step 3 of 3)

Name	bigip3.lab
------	------------

Cancel Add Device

5. Go to **bigip3** and verify **bigip1** and **bigip2** are now in the peer list

Device Management » Device Trust : Device Trust Members

Local Domain Identity **Device Trust Members**

Peer and Subordinate Devices Add...

<input checked="" type="checkbox"/>	Name	Device Type	Hostname	Serial Number	MAC Address
<input type="checkbox"/>	bigip1.lab	Peer	bigip1.lab	42378932-5196-5a7d-303c685f5fb2	00:50:56:b7:d7:65
<input type="checkbox"/>	bigip2.lab	Peer	bigip2.lab	42378932-5196-5a7d-303c685f5fb2	00:50:56:b7:d7:65

Delete

TASK 4 – Create New Sync Only Group

On bigip1:

Create a sync only group

- Go to **Device Management -> Device Groups -> Create**
 - Name = device_group_02_so
 - Group Type = Sync-Only
 - Members = All 3 bigip's

Device Management » Device Groups » **New Device Group...**

General Properties

Name:

Group Type: Sync-Only

Description:

Configuration: Basic

Members:

Includes	Available
<input type="checkbox"/> /Common <input type="checkbox"/> bigip1.lab <input checked="" type="checkbox"/> bigip2.lab <input type="checkbox"/> bigip3.lab	<input type="checkbox"/>

Sync Type: Manual with Incremental Sync

Cancel Repeat Finished

2. Click **Finished**

Perform initial sync

1. Click **Awaiting Initial Sync** in the upper-left of the GUI
2. Choose `device_group_02_so`, then choose `bigip1`.
3. Select **Push** the selected device configuration to the group and then click **Sync**

Device Management » **Overview**

Device Groups:

Sync Issues :

<p>device_group_02_so</p> <p> Awaiting Initial Sync</p> <p>3 Devices Sync-Only Group Manual Sync No successful group sync</p>

Awaiting Initial Sync

The device group is awaiting the initial config sync

Recommended action: Synchronize one of the devices to the group

Devices: View: Basic

New Device Group Members

Device	Status	Configuration Time
bigip1.lab (Self)	Awaiting Initial Sync	Configuration Time : Device has not synced with group
bigip2.lab	Awaiting Initial Sync	Configuration Time : Device has not synced with group
bigip3.lab	Awaiting Initial Sync	Configuration Time : Device has not synced with group

Sync Options:

Push the selected device configuration to the group
 Pull the most recent configuration to the selected device

Sync

In Sync :

<p>device_trust_group</p> <p> In Sync</p> <p>3 Devices Sync-Only Group Auto Sync In sync on 6/21/2017 at 14:55:07</p>
<p>device_group_01</p> <p> In Sync</p> <p>2 Devices Sync-Failover Group Manual Sync In sync on 6/21/2017 at 14:41:52</p>

TASK 5 – Create New Partition and SSL Profile, Configure for Sync-Only

On bigip1:

Create new Partition

1. Go to **System -> Users -> Partition List -> Create**

- Partition Name = partition_02_so
- Device Group = (uncheck “Inherit device group from root folder” box), device_group_02_so
- Traffic Group = None

System >> Users : Partition List >> New Partition...

Properties

Partition Name	partition_02_so
Partition Default Route Domain	0
Description	<input type="checkbox"/> Extend Text Area <input type="checkbox"/> Wrap Text

Redundant Device Configuration

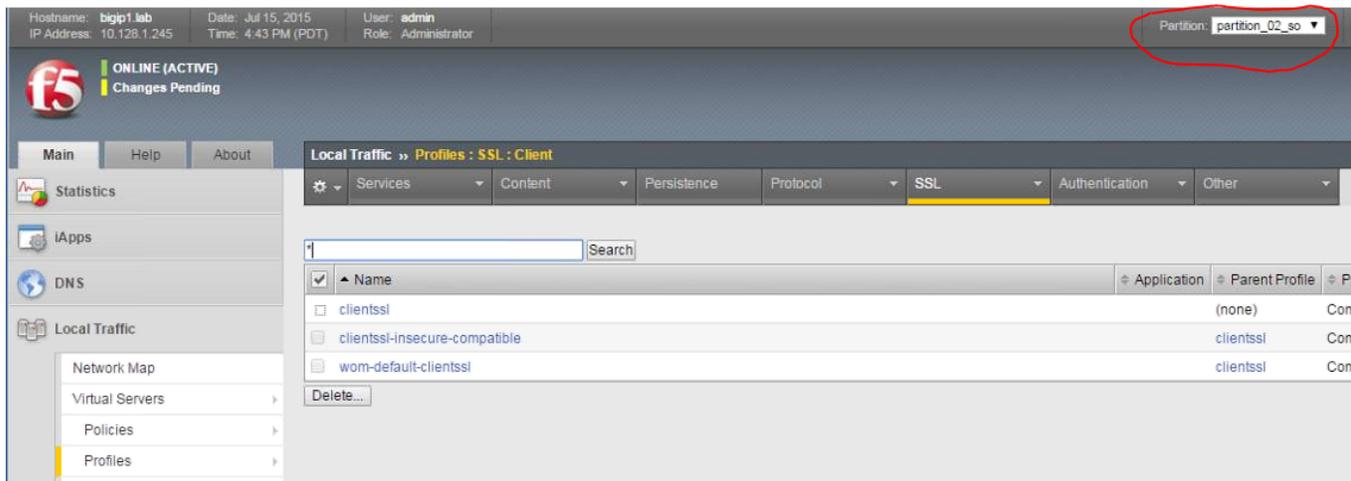
Device Group	<input type="checkbox"/> Inherit device group from root folder device_group_02_so (Sync-Only)
Traffic Group	None

Cancel Repeat Finished

2. Click **Finished**

Create new Client SSL Profile

1. Go to **Local Traffic -> Profiles -> SSL -> Client**
2. Change Partition to partition_02_so in the upper-right of the GUI



3. Click **Create**

- Name = `clientssl_02_lab`
- Accept all defaults

4. Click Finished

Sync Changes

1. On **bigip2** and **bigip3**, confirm this Sync-Only `clientssl` profile has synced
2. Go to **Local Traffic -> Profiles -> SSL -> Client**
3. Choose `partition_02_so`
4. Is `clientssl_02_lab` there?

What are some practical uses for Sync-Only device groups?

2.2.3 Lab 3: Device Service Cluster Automation

We'll be using POSTMAN and it is built upon Chrome libraries, therefore, you must use Chrome to visit every BIG-IP login page (i.e. <http://10.128.1.245/>) and bypass the security warning - otherwise the REST calls will fail. If you fail to bypass these security warnings because of the untrusted SSL certificate on the BIG-IP management interface you will receive an error when attempting to send REST calls.

Pre-lab work

To start this lab, we'll want to start with a blank state again - erasing all that hard work you've done. In order to do so, go to "SYSTEM -> Archives" and then click on the UCS archive shown. Once it is up, click to restore and start the process on both BIG-IP's in order to go back to the default Lab configuration.

Objective: At a high level, we'll be building a simple active/standby device service cluster (DSC). With some new DevOps oriented packages created by F5, the heavy lifting will be taken away, and we have some simple REST API calls to make to accomplish the DSC creation. The high level steps to do this are as follows:

1. Authenticate. We'll show basic and token based authentication - but we'll then use token authentication throughout the rest of the exercises.
2. Installation of the DevOps packages. We'll SCP the packages to the BIG-IP's, then via API, install them and verify they are installed.

3. Execute the device settings package in order to set the hostname, device name, NTP server and NTP timezone.
4. Execute the DSC configuration package in order to create an active/standby DSC.

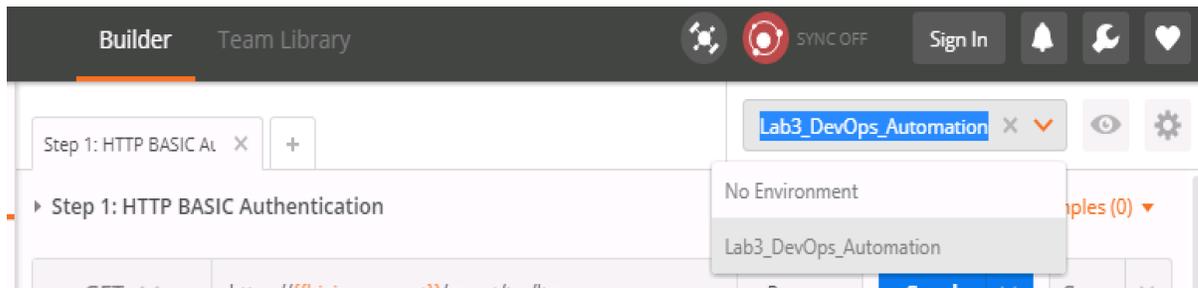
Hint: Note that when working with API's, generally once you create an object it is done via a POST. If you try to modify an object by using the POST method again, you will create another instance of what you've done, or receive an error citing an object with that name already exists. The "Blocks" that you submit via POST will be duplicated if you use the POST method as they will each receive a unique ID upon creation.

Modification is done via the PATCH HTTP method calling out the same name/ID. There are a few "Optional" tasks created within POSTMAN which you can experiment Unbinding and deleting application blocks.

TASK 1 – 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 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.

1. Open Postman from the Desktop and make sure the Environment is set to Lab3_DevOps_Automation



TASK 2 – HTTP BASIC Authentication

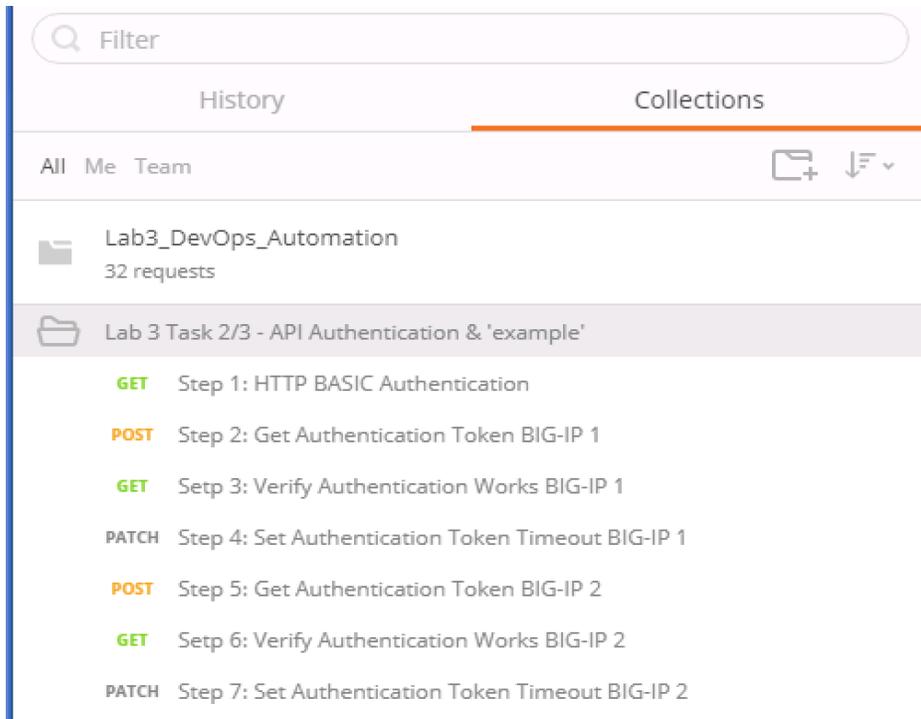
In this task we will use the Postman tool to send API requests using HTTP BASIC authentication. As its name implies this method of authentication encodes the user credentials via the existing BASIC authentication method provided by the HTTP protocol. The mechanism this method uses is to insert an HTTP 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=
```

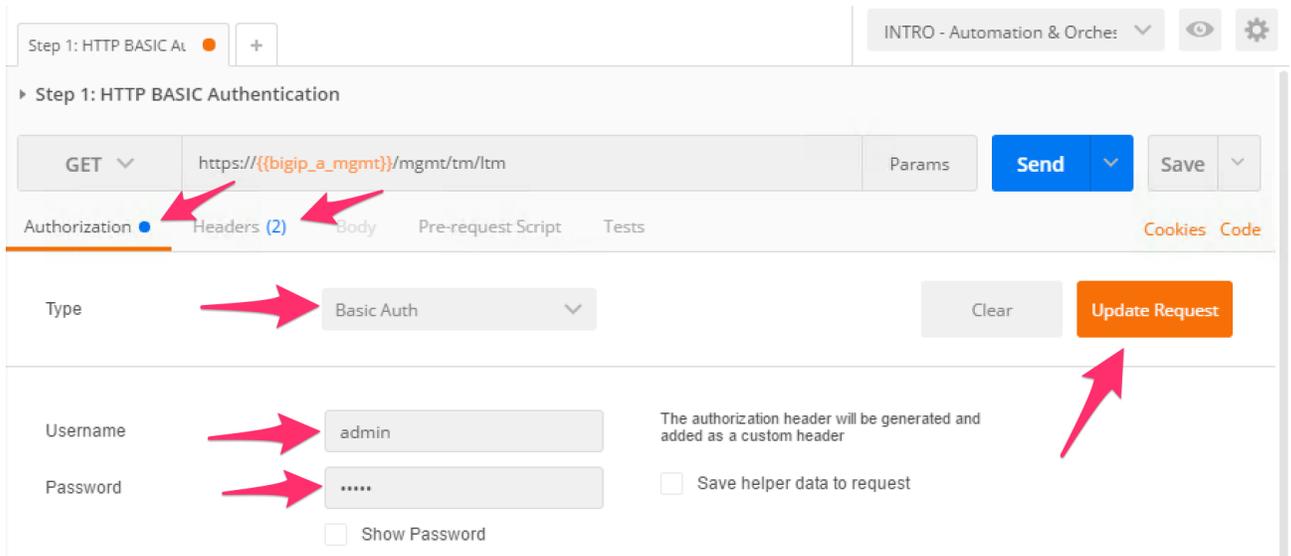
It should be noted that cracking the method of authentication is TRIVIAL; as a result API calls should always be performed using HTTPS (F5 default) rather than HTTP.

Perform the following steps to complete this task:

1. Open the Postman application from the link on the Desktop. Click the 'Collections' tab on the left side of the screen, expand the Lab3_DevOps_Automation collection on the left side of the screen, expand the Lab 3 Task 2/3 – API Authentication & 'example' folder:



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 '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:



3. Click the 'Send' button to send the request. If the request succeeds you should be presented with a listing of the /mgmt/tm/ltn Organizing Collection.
4. Update the credentials and specify an INCORRECT password. Send the request again and examine the response:



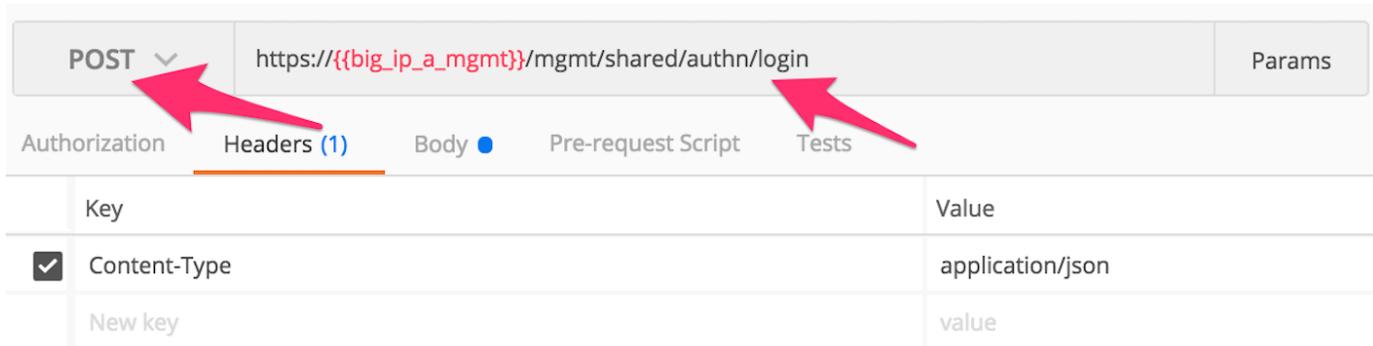
TASK 3 – Token Based Authentication

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. 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. Both 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 (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.

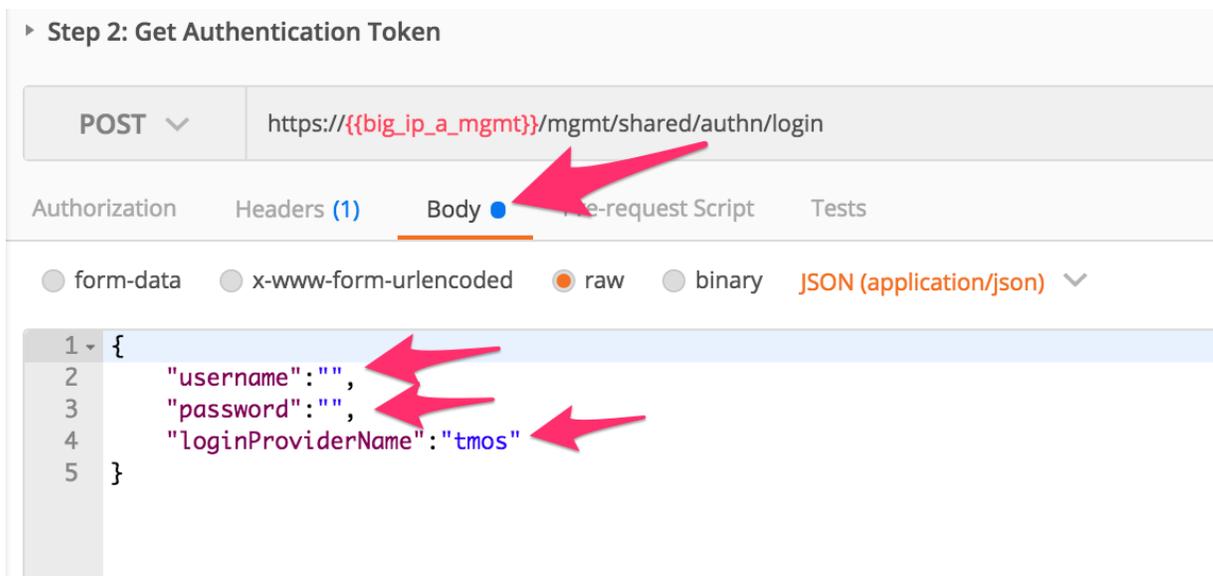
Hint: For more information about external authentication providers see the section titled **About external authentication providers with iControl REST** in the iControl REST API User Guide available at <https://devcentral.f5.com>

Perform the following steps to complete this task:

1. Click the 'Step 2: Get Authentication Token' item in the Lab 3.1 Postman Collection
2. Notice that we send a POST request to the `/mgmt/shared/authn/login` endpoint.



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:

Successful:



Unsuccessful:



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:

Body Cookies Headers (22) Tests

Pretty Raw Preview JSON

```

1 {
2   "username": "admin",
3   "loginReference": {
4     "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49",
5   },
6   "loginProviderName": "tmos",
7   "token": {
8     "token": "QJXK6HQWZFW35VC3JRZOXWNNNDQ",
9     "name": "QJXK6HQWZFW35VC3JRZOXWNNNDQ",
10    "userName": "admin",
11    "authProviderName": "tmos",
12    "user": {
13      "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a",
14      "b032-0b2f4a2523a1"
15    }
16  },
17 }

```

- Click the 'Step 3: Verify Authentication Works' item in the Lab 3.1 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.

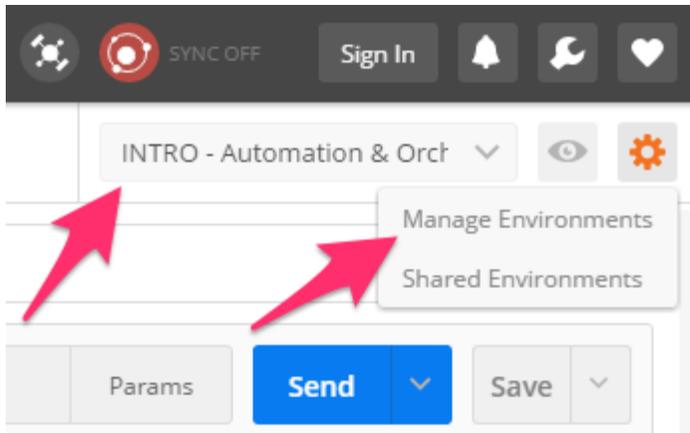
▶ Setp 3: Verify Authentication Works

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

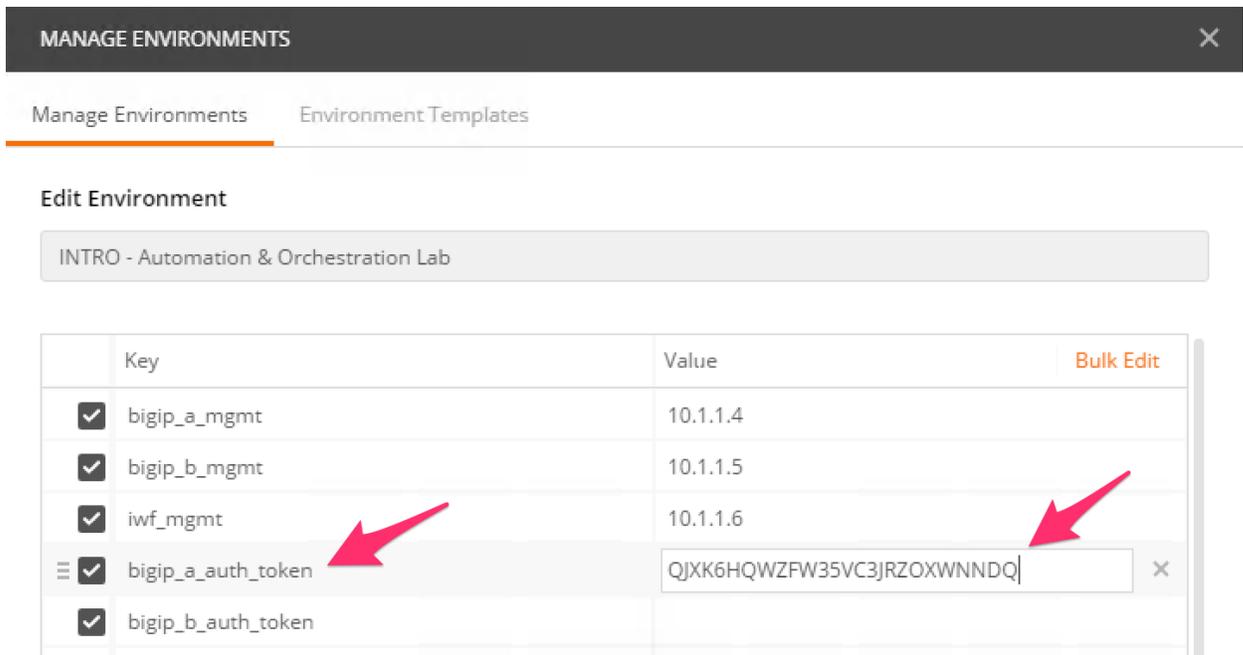
Authorization Headers (1) Body Pre-request Script Tests

Key	Value
<input checked="" type="checkbox"/> X-F5-Auth-Token	QJXK6HQWZFW35VC3JRZOXWNNNDQ
New key	value

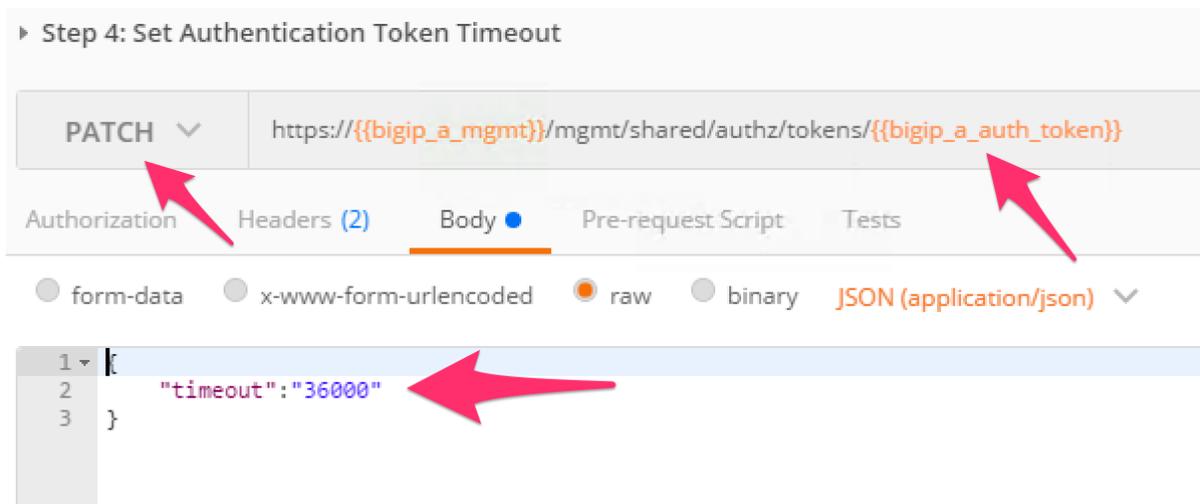
- Click the 'Send' button. If your request is successful you should see a '200 OK' status and a listing of the ltmOrganizing Collection.
- We will now update your Postman environment to use this auth token for the remainder of the lab. Click the Environment menu in the top right of the Postman window and click 'Manage Environments':



10. Click the `Lab3_DevOps_Automation` item
11. Update the value for `bigip_a_auth_token` by Pasting (`Ctrl-v`) in your auth token:



12. Click the 'Update' button and then close the 'Manage Environments' window. Your subsequent requests will now automatically include the token.
13. Click the 'Step 4: Set Authentication Token Timeout' item in the Lab 3.1 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:



- Repeat the same steps for setting the authentication token for BIG-IP 2 into the environment variable `bigip_b_auth_token`, ensuring the set the timeout as well.

TASK 4 – Install iApp RPM Packages

In this Task you'll be installing two DevOps packages developed by F5 built for the express purpose of quickly and easily taking a new set of BIG-IP's and bringing them online programmatically. You will install the packages on the two BIG-IP's using SCP, and then interact with the BIG-IP API.

- Use WinSCP to SCP from `~\Documents\LabFiles*.rpm` to `/var/config/rest/downloads` on BIG-IP 1 and BIG-IP 2.

The user is `root` and the password is `default`. WinSCP has been preconfigured for both BIG-IP's. You can exclude the DSC RPM on BIG-IP 2 if you desire, as we won't install it there.

- Expand the Lab 3 Task 4 'Install RPM Packages' collection in Postman and click Step 2: "Install device settings RPM package on BIGIP1".

Click the Body tab and note the command.

Click Send

You should receive a 200 OK response

- Expand the Lab 3 Task 4 'Install RPM Packages' collection in Postman and click Step 3: "Install DSC RPM package on BIGIP1."

Click the Body tab and note the command.

Click Send

You should receive a 202 Accepted response

- Expand the Lab 3 Task 4 'Install RPM Packages' collection in Postman and click Step 4: "Install device settings RPM package on BIGIP 2".

Click the Body tab and note the command.

Click Send

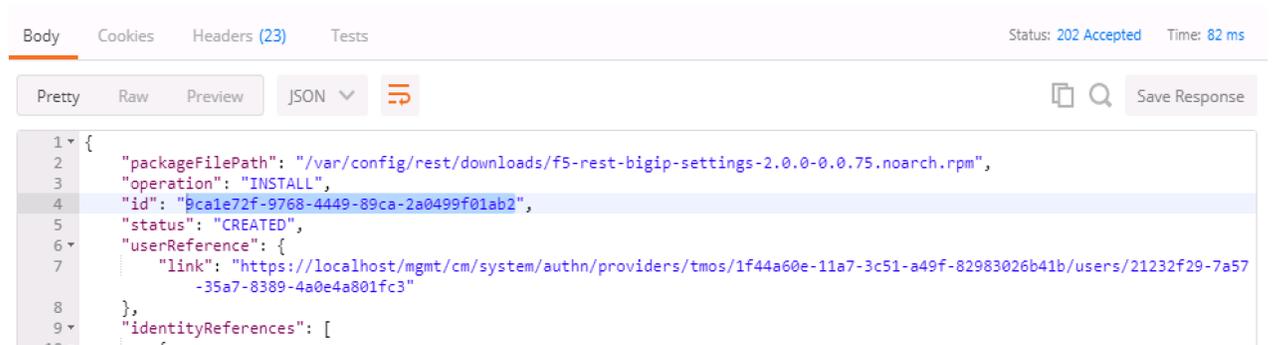
You should receive a 202 Accepted response

- Run Step 5 "Get Block Template IDs on BIG-IP 1." The output of this command will return two JSON templates, one for each package that was installed. At the top of the JSON payload is

the field `id`. Capture the unique ID for the `id` key and put it into the Environment Variables `bigip_a_settings_id` and `bigip_a_dsc_id`. Ensure that these ID's are mapped from the correct template to the correct environment variable or later calls will fail. If you get zero, or just one TEMPLATE result back, there are "OPTIONAL" calls as part of Task 4.

6. Run Step 6 "Get Block Template IDs on BIG-IP 2." Just like the previous step, take the "id" output for the device-settings TEMPLATE and put it into the environment variable `bigip_b_settings_id`. The DSC package was not installed on BIG-IP 2 so you will only see one block template.

Error: Running the "OPTIONAL" REST calls if you ran into trouble. There are three calls, one for each of the packages that get installed in the event you need to delete a "block". In order to run them, we require the task ID from the installation REST Call. The graphic below shows that ID. Take that ID and then move to the requisite task and put the ID at the end of the URI replacing `{guid}` and run the call. The resulting output will tell you what has happened. Most likely, the package isn't in the right directory and cannot be installed. If that's the case, move the file in place and run the install REST call again.

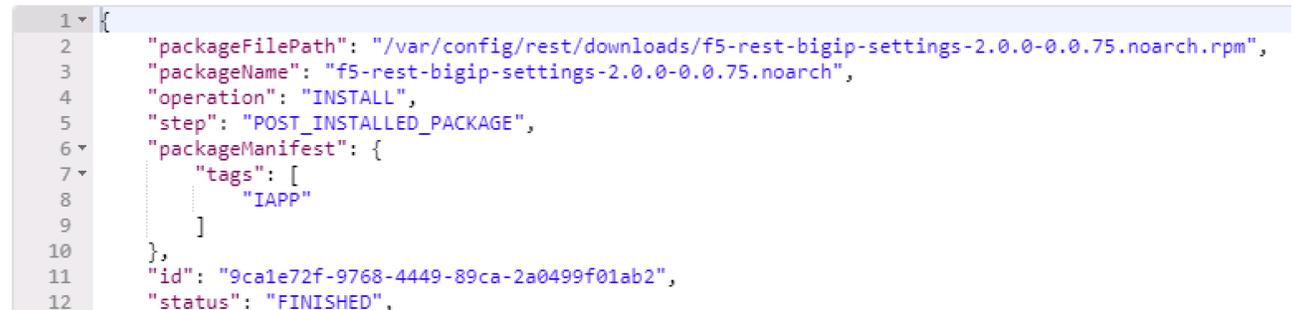


The screenshot shows a REST client interface with the following details:

- Body tab selected, Status: 202 Accepted, Time: 82 ms
- JSON view selected
- Response body (lines 1-10):

```
1 {
2   "packageFilePath": "/var/config/rest/downloads/f5-rest-bigip-settings-2.0.0-0.0.75.noarch.rpm",
3   "operation": "INSTALL",
4   "id": "9ca1e72f-9768-4449-89ca-2a0499f01ab2",
5   "status": "CREATED",
6   "userReference": {
7     "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/users/21232f29-7a57-35a7-8389-4a0e4a801fc3"
8   },
9   "identityReferences": [
10  ]
11 }
```

The response contains the updated status of the package management task. The JSON in the response contains the status property. When the value of status updates to "FINISHED", the package installation is complete. You can then find the added package at `/var/config/rest/iapps/f5-rest-bigip-settings` on the command line.



The screenshot shows a REST client interface with the following details:

- JSON view selected
- Response body (lines 1-12):

```
1 {
2   "packageFilePath": "/var/config/rest/downloads/f5-rest-bigip-settings-2.0.0-0.0.75.noarch.rpm",
3   "packageName": "f5-rest-bigip-settings-2.0.0-0.0.75.noarch",
4   "operation": "INSTALL",
5   "step": "POST_INSTALLED_PACKAGE",
6   "packageManifest": {
7     "tags": [
8       "IAPP"
9     ]
10  },
11   "id": "9ca1e72f-9768-4449-89ca-2a0499f01ab2",
12   "status": "FINISHED",
13 }
```

TASK 5 – Configure Device Settings

This task modifies base device settings. The DSC package requires NTP, DNS, and hostname to be set before it will work, so that is what this exercise is doing. This package can also license/re-license a device, or default the configuration in one REST call. Because of the lab setup, we will not be doing the licensing portion in the lab.

Expand the Lab 3 Task 5 'Device Settings – NTP/DNS/License' collection.

1. This step lists the specific “block” identified by `{{bigip_a_settings_id}}`, meaning the installed iApp packages and their associated properties. If you were to remove the `/{{bigip_a_settings_id}}` from the URI, you could see all configured blocks, Those with `TEMPLATE` as the status, as well as any in `BINDING`, `BOUND` or `ERROR` state.

To formulate your own call, you’d take the output of the `TEMPLATE` block and create your own “block” in order to change settings. We’ve already done that in Step 2, and the environment variables are being used to identify the parent `TEMPLATE/package`.

```

1 {
2   "items": [
3     {
4       "id": "cb0fae5-b497-3379-87f2-4e09d029ba5a",
5       "name": "bigip-dsc",
6       "inputProperties": [
7         {
8           "id": "clusterName",

```

2. We’ve formulated a valid call to BIG-IP 1 here, and we’ve modified settings like hostname, timezone, NTP and DNS (server) and search domain. Feel free to change some of these settings if you’d like, but make the changes simple as syntax is vitally important to the success of the call. We’re using IP addresses that will have no meaning in a real configuration (IP address of `127.27.1.1`), but are required to be set for the DSC RPM.

One item to note is the `selfLink` variable at the bottom of the call body. The correct ID needs to be identified and we do that by utilizing the `{{bigip_a_settings_id}}`. property.

The response for this call will give you a ID as the top line of the response. Capture that ID and move to Step 3.

```

1 {
2   "id": "60d7bd4f-0574-461c-9d4e-d9d0e312dc0a",
3   "name": "Lab_3.3 Settings",
4   "inputProperties": [
5     {
6       "id": "deviceReference",
7       "type": "REFERENCE",
8       "value": {
9         "link": "localhost"

```

3. In step 3 you will need to paste the ID captured in step 2 in place of the `{guid}` at the end of the URI.

▶ Step 3: Verify Device Details status BIG-IP 1 Exa

GET https://{{bigip_a_mgmt}}/mgmt/shared/iapp/blocks/60d7bd4f-0574-461c-9d4e-d9d0e312dc0a Params Send

Authorization Headers (2) Body Pre-request Script Tests

We are interested to see the state “`BOUND`” which is found near the end of the response. If the state still shows “`BINDING`,” continue to re-run this call until you see `BOUND` or an error. The errors should be self explanatory, but if you have trouble, see one of the proctors.

```

"configProcessorTimeoutSeconds": 500,
"statsProcessorTimeoutSeconds": 15,
"state": "BOUND",
"generation": 2,
"lastUpdateMicros": 1500395025391411,
"kind": "shared:iapp:blocks:blockstate",
"selfLink": "https://localhost/mgmt/shared/iapp/blocks/60d7bd4f-0574-461c-9d4e-d9d0e312dc0a"

```

Steps 4,5,6 - Repeat steps 1, 2, and 3 for bigip2, but use the POSTMAN steps 4, 5, and 6 that have been pre-set for BIG-IP 2.

Close all open tabs (Don't Save if prompted) at the top of POSTMAN to avoid unexpected crashes.



TASK 6 – Create the Device Service Cluster

Hint: The rest of the queries will be run against BIG-IP 1.

In this portion of the lab we'll be creating the DSC.

Expand the Lab 3.4 'Device Service Cluster' collection.

1. This step adds a device group called `lab3_4_group`. Run this step and ensure you get a 200 OK response code.

In Steps 2 and 3 we'll add both BIG-IP's to this group. Run these steps and ensure you get a 200 OK response code.

2. Run Step 2 REST call

3. Run Step 3 REST call

4. In Step 4 we'll query this device-group we just created and we'll verify that it was created with both devices in the group and we'll also need to grab the `machineId` values from the output and copy them into the BODY of the call in Step 6.

▶ New Step 4: GET machineId Examples (0) ▼

GET `https://{{(bigip_a_mgmt)}/mgmt/shared/resolver/device-groups/lab3_4_group/devices` Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

Type No Auth ▼

Body Cookies Headers (22) Tests Status: 200 OK Time: 84 ms

Pretty Raw Preview JSON ▼ Save Response

```

1 {
2   "items": [
3     {
4       "uuid": "6bb2d7fa-11f5-447a-9fd8-7a5b26a09ace",
5       "deviceUri": "https://10.128.1.245:443",
6       "machineId": "6bb2d7fa-11f5-447a-9fd8-7a5b26a09ace",
7       "state": "ACTIVE",
8       "address": "10.128.1.245",

```

- This step is a formality as we've already created a call for you, but this is how you'd gain the body of a call you formulated yourself. We've done that in Step 6, so go ahead and move to the next step.

▶ Step 5: Get Block ID Examples (0) ▼

GET Params

Authorization Headers (2) Body Pre-request Script Tests Code

Type

Body Cookies Headers (22) Tests Status: 200 OK Time: 118 ms

Pretty Raw Preview JSON

```
112     "state": "BOUND",
113     "generation": 2,
114     "lastUpdateMicros": 1500395025391411,
115     "kind": "shared:iapp:blocks:blockstate",
116     "selfLink": "https://localhost/mgmt/shared/iapp/blocks/60d7bd4f-0574-461c-9d4e-d9d0e312dc0a"
117   },
118 },
119 {
120   "id": "7cb0fae5-b497-3379-87f2-4e09d029ba5a",
121   "name": "bigip-dsc",
122   "inputReferences": [
```

- In this step, we'll be creating the sync-failover group `Lab_3.4_failover-cluster`, adding a `HA_VLAN` on interface `1.3` and creating Self-IP's for those VLANS on both BIG-IP's.

You'll need to capture the `machineId` from step 4 for both BIG-IP's and paste it into the `deviceReference` under the section for each BIG-IP. The item to replace will be labeled as `{machineId_BIGIP_X}` where X is 1 or 2.

Step 6: Create DSC Examples (0) ▾

POST ▾ `https://{{bigip_a_mgmt}}/mgmt/shared/iapp/blocks` Params Send ▾ Save ▾

Authorization Headers (2) **Body** ● Pre-request Script Tests Code

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

```

10  {
11      "id": "clusterMembers",
12      "type": "JSON",
13      "value": [
14          {
15              "hostname": "bigip1.lab",
16              "deviceReference": "http://localhost/mgmt/shared/resolver/device
17                  -groups/lab3_4_group/devices/{machineId_BIGIP_1}",
18              "ip": "10.128.1.245",
19              "username": "admin",
20              "password": "admin",
21              "ha_vlan": {
22                  "name": "HA_VLAN",
23                  "tag": 3000,
24                  "interface": "1.3",
25                  "tagged": false
26              },
27              "ha_selfip": {
28                  "name": "HA_SELFIP",
29                  "address": "192.168.1.10/24",
30                  "tagged": true
31              }
32          }
33      ]
34  }

```

Once you have these two properties updated, then please run the REST call. Take the ID from the top line of the response as we'll use that to verify that the status is `BOUND` in the following step.

- Take the ID saved from the response from Step 6 and pasted it at the end of the URI, replacing `{guid}` as you've done before. Run this step until you see the `state` show as `BOUND`. If it shows as `BINDING`, then you can continue to run this command until you see `BOUND` or `ERROR`. The error should be self explanatory, but if you have trouble, please see a proctor.

Note: At this point, you should have a valid DSC in active/standby state. Verify that you have a new VLAN (`HA_VLAN`) and SelfIP for that VLAN and the cluster should be "In-Sync."

2.2.4 Final Notes

Here are a few helpful Solution Articles which helped us create this lab:

SOL 13250 - Overview of Port Lockdown Behavior

- TMOS v10.x - v11.x: <https://support.f5.com/csp/article/K13250>
- TMOS v12.x - v13.x: <https://support.f5.com/csp/article/K17333>

Even if you specify **Port Lockdown** as `None` this Solution states the following: *ICMP traffic is always allowed, and if the BIG-IP systems are configured in a redundant pair, ports that are listed as exceptions are always allowed from the peer system.*

SOL 13946 - Troubleshooting ConfigSync and device service clustering issues (11.X)

<https://support.f5.com/csp/article/K13946>

Ever see “Awaiting Initial Sync,” “Sync Failure,” or “Changes Pending” and don’t know how to go about resolving the issue? This is a guide to keep handy if you’re running in HA mode with 11.X and above. It will give you simple tests do diagnose the issues you will face and how to resolve them.

Version 13.0 Release Notes

https://support.f5.com/kb/en-us/products/big-ip_ltm/releasesnotes/product/relnote-ltm-13-0-0.html

New Built-in TCP Profiles, auto Nagle, auto Buffer tuning

Default TCP profiles are becoming read only for “troubleshooting purposes.”

If you are using the default TCP profile, it is not optimal and can be less productive for your applications. Please see: <https://devcentral.f5.com/articles/stop-using-the-base-tcp-profile>

“When the latency penalty in using Nagle’s algorithm is small, the feature assembles data into full-size packets; when it is high, it stops doing that.”

Instead of analyzing traffic captures and setting send buffer/receive buffer sizes manually, customers can now select the auto option for send/receive buffer sizes and high/low watermarks for the TCP Proxy buffers

The content contained here leverages a full DevOps CI/CD pipeline and is sourced from the GitHub repository at <https://github.com/f5devcentral/f5-agility-labs-adc>. Bugs and Requests for enhancements can be made by opening an Issue within the repository.

Class 3: BIG-IP® Local Traffic Manager (LTM) - v13.1 Lab Guide

This lab guide is designed for you to get an understanding of the BIG-IP Local Traffic Manager (LTM) product.

3.1 Lab Overview

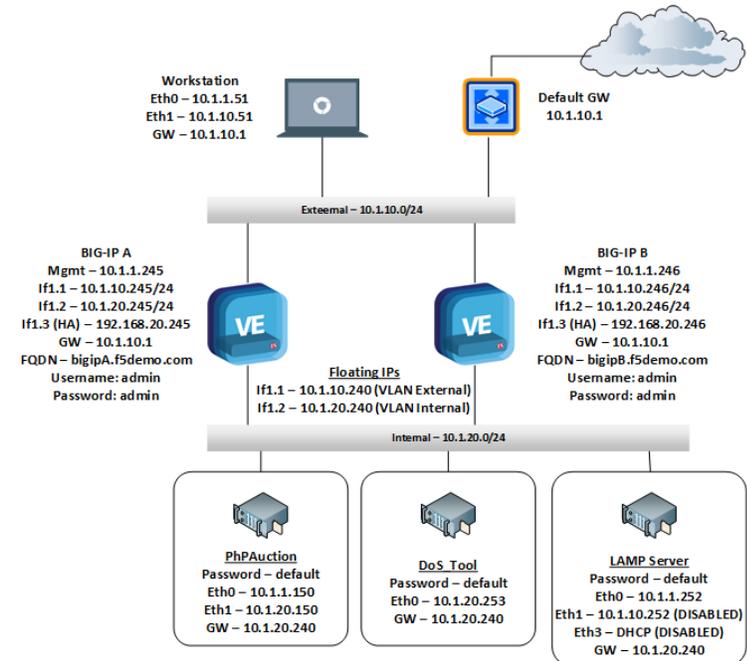
- F5 BIG-IP LTM VE, licensed using F5-BIG-VE-LAB-LIC
- Your BIG-IP is as close to factory default as possible. Only the following changes have been made:
- The management IP has already been configured
- The initial setup has been completed. (Licensing and Platform information)
- The Idle Timeout was modified from 1200 seconds to 7200 seconds
- The Welcome messages for the GUI and SSH were changed.
- An archive file base-setup-and-licensing.ucs was created allowing you to revert to the base settings above.

Note: Various directory and application services are available within the lab environment.

Warning: DO NOT COPY INFORMATION FROM THE SCREENSHOTS. THEY ARE FOR REFERENCE ONLY.

3.2 Scenario

Your customer has the following environment. The servers sit on the customers internal VLANs, the virtual servers will exist in another VLAN in the DMZ. The customer does not want to rework their networking and does not wish to use the LTM as the default gateway. Our solution will be to use SNATs to force traffic to pass through the BIG-IP to return through the LTM.



<p>LDAP 10.1.20.252:1389 or 389</p> <p>People</p> <ul style="list-style-type: none"> user.0-user.49 adminuser corpuser remoteuser <p>Groups</p> <ul style="list-style-type: none"> Employees <ul style="list-style-type: none"> user.0-user.49 corpuser remote <ul style="list-style-type: none"> user.11-user.20 admin <ul style="list-style-type: none"> adminuser user.0 user.10 user.20 user.30 <p>Password for ALL users: password</p>	<p>DNS 10.1.20.252:53</p> <ul style="list-style-type: none"> dojo.f5demo.com. 10.1.10.50 ldp1.f5demo.com. 10.1.10.100 App1.f5demo.com. 10.1.10.101 App2.f5demo.com. 10.1.10.102 ldp2.f5demo.com. 10.1.10.103 ldp3.f5demo.com. 10.1.10.104 Cert.f5demo.com. 10.1.20.11 Server1.f5demo.com. 10.1.20.11 Server2.f5demo.com. 10.1.20.12 Server3.f5demo.com. 10.1.20.13 Server4.f5demo.com. 10.1.20.14 Server5.f5demo.com. 10.1.20.15 App3.f5demo.com. 10.1.20.16 Dvwa.f5demo.com. 10.1.20.17 Peruggia.f5demo.com. 10.1.20.18 Wackopicko.f5demo.com. 10.1.20.19 Hackazon.f5demo.com. 10.1.20.20 Websafe.f5demo.com. 10.1.20.50 Auction.f5demo.com. 10.1.20.150 Lamp.f5demo.com. 10.1.20.252 Ldap.f5demo.com. 10.1.20.252 Lamp2.f5demo.com. 10.1.20.252 	<p>Apache</p> <p>Demo Servers</p> <ul style="list-style-type: none"> Server1.f5demo.com 10.1.20.11:80, 443, 8081 Server2.f5demo.com 10.1.20.12:80, 443, 8081 Server3.f5demo.com 10.1.20.13:80, 443, 8081 Server4.f5demo.com 10.1.20.14:80, 443, 8081 Server5.f5demo.com 10.1.20.15:80, 443, 8081 <p>Hackables</p> <ul style="list-style-type: none"> Dvwa.f5demo.com 10.1.20.17 admin/password Peruggia.f5demo.com 10.1.20.18 admin/password Wackopicko.f5demo.com 10.1.20.19 admin/admin Hackazon.f5demo.com 10.1.20.20 admin/admin Hackazon DB root/default <p>FPS</p> <ul style="list-style-type: none"> Websafe.f5demo.com 10.1.20.50 or 10.1.20.252 user/12345678 DemoTools http://10.1.1.252/demotools.php <p>Services</p> <ul style="list-style-type: none"> Webmin localhost:10000 root/default Splunk Web Interface 10.1.20.252:8000 admin/default Splunk Logging Port 10.1.20.252:5514 SimpleSAMLPHP localhost/simplesam/ OpenAM localhost:8080/openam Sert.f5demo.com server1.f5demo.com /CaPolicy/ /CaRevocation/ /Policy/ /Revocation/ 	<p>RADIUS 10.1.20.252:1812</p> <p>Clients: 10.1.20.240 10.1.20.241</p> <p>Secrets: f5networks (for clients) testing123 (for def tests)</p> <p>Users: user0/password user1/password</p> <p>Other</p> <ul style="list-style-type: none"> FTP/SCP 10.1.1.252 10.1.20.252 f5 or root/default SYSLOG port 514 NTP port 123 TinyCA cert.f5demo.com pwd: f5networks Splunk http://localhost:8000 admin/default VNC 10.1.1.252:5900 password: default
---	--	--	---

3.2.1 Lab 1: Access the Lab Environment

You will now gain access to your lab environment and your BIG-IP. The first agenda item is to log in to the training portal. After this, you will create a remote desktop session to the training environment and access your BIG-IP.

Access the Lab Environment

1. Using Firefox or Chrome open a connection to the training portal supplied by the instructor

- (a) Using the IP address supplied by the instructor log on to the training portal
- i. http://<ip_address_supplied_by_instructor/>

- ii. User: **ltm101studentX** (where **X** is your student number)
 - iii. Password: **PieIntheSKYltm101**
- (b) Open the blueprint or the view link.

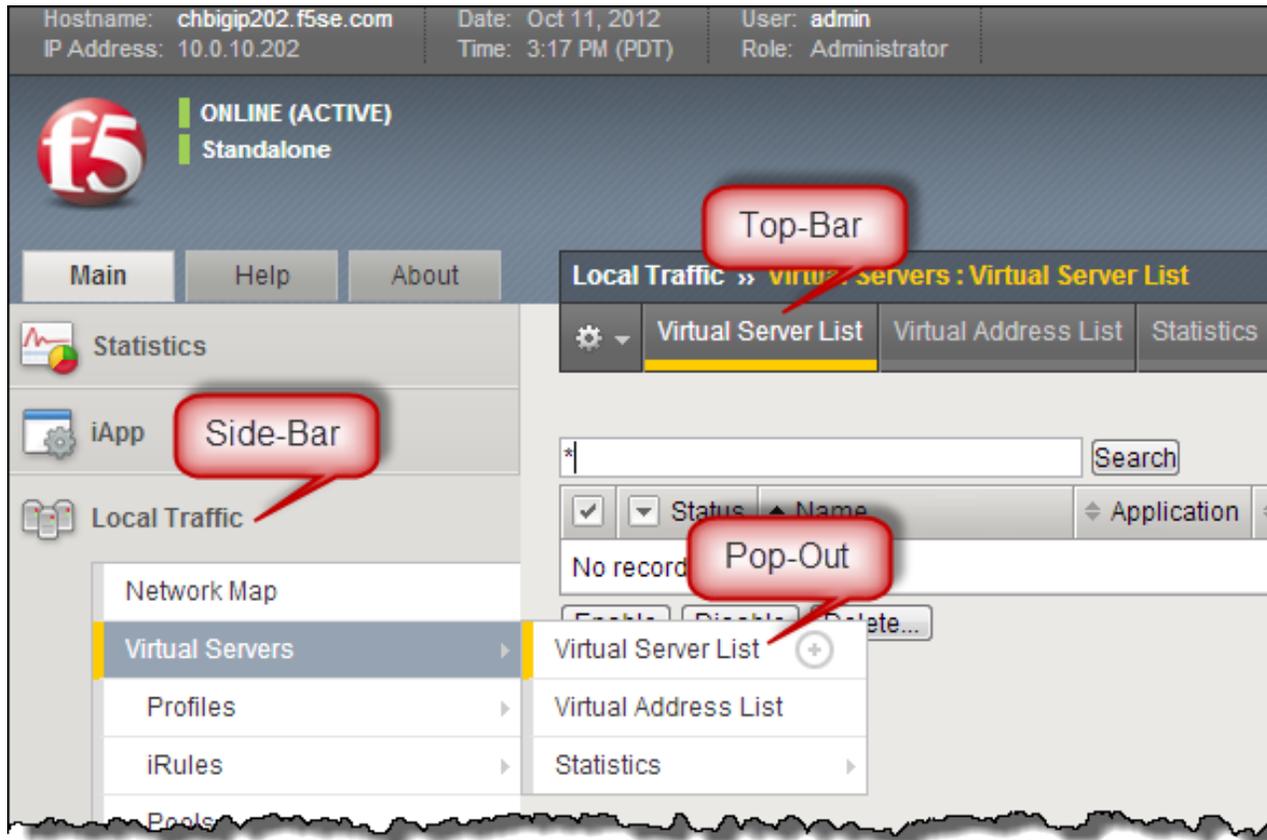
Name	# of VMs	# of running VMs	Actions
ltm-101-bp-v5	4	4	View

- (c) Find the Xunbutu jumpbox in the list of applications and select console to access the jumpbox via the browser. Alternately you can use the IP address associated with the jumpbox to access it via RDP.

<input type="checkbox"/>	VM name	Status	IP	DNS	Actions
<input type="checkbox"/>	bigip01-v11.5.3-v12.0-no-license	STARTED			Go! Console
<input type="checkbox"/>	bigip02-v11.5.3-v12.0-no-license	STARTED			Go! Console
<input type="checkbox"/>	LAMP_3.6b	STARTED			Go! Console
<input type="checkbox"/>	xbun-jumpbox-v8	STARTED	85.190.180.144	xbunjumpboxv8-ltm101v115xltm101...	Go! Console

- (d) Access the **xbun-jumpbox** using the **Console** (watch for pop-up blockers)
- i. User: **f5student**

- ii. Password: **f5DEMOs4u**
- 2. To access the BIG-IP open a new browser window and
 - (a) Click the bookmark **bigip01**
 - i. User: **admin**
 - ii. Nomenclature for GUI navigation begins on the side-bar and then goes to the pop-up or top-bar.



- iii. For example: **Go to Local Traffic >> Virtual Servers >> Virtual List**

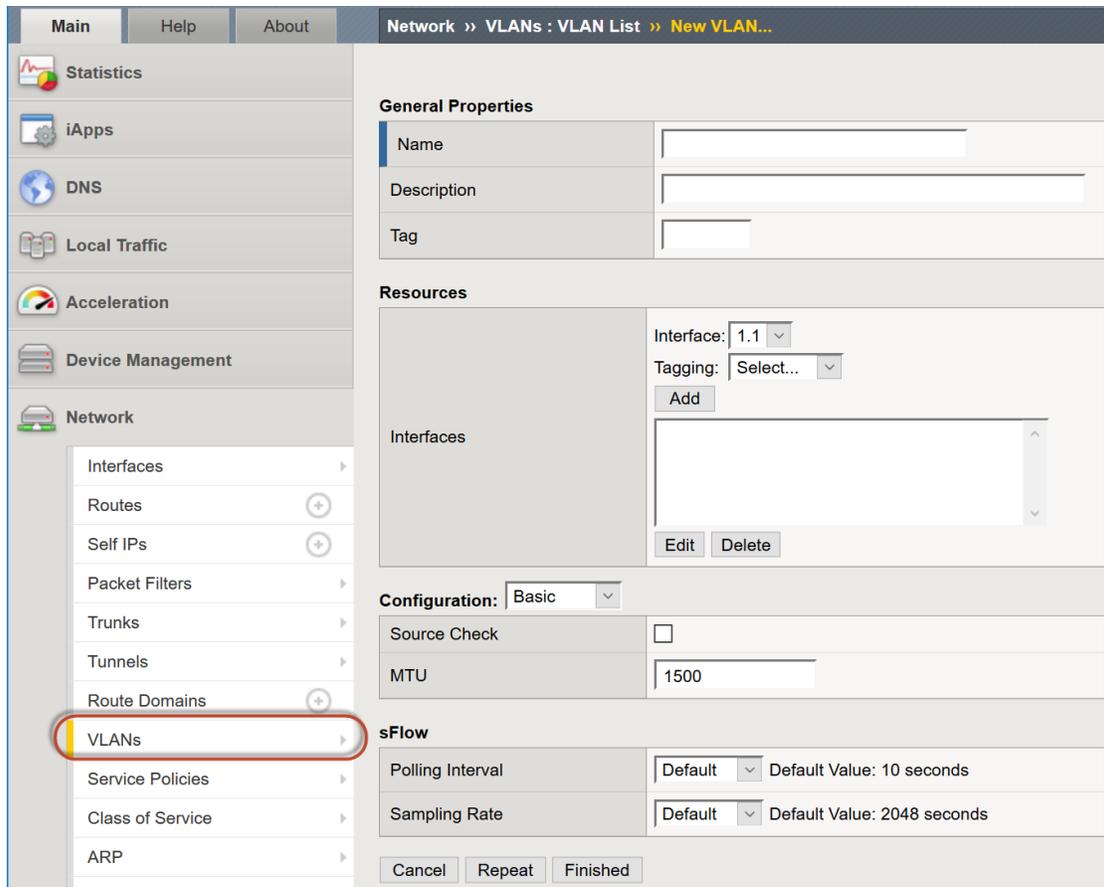
3.2.2 Lab 2: The Basics (Networking, Pools and Virtual Servers)

In this lab we will access the Management GUI. We will then create the VLANs and assign self IP addresses to our VLAN. As mentioned during our lecture portion, BIG-IPs may be put in-line or one-armed depending on your customer's requirements and topology.

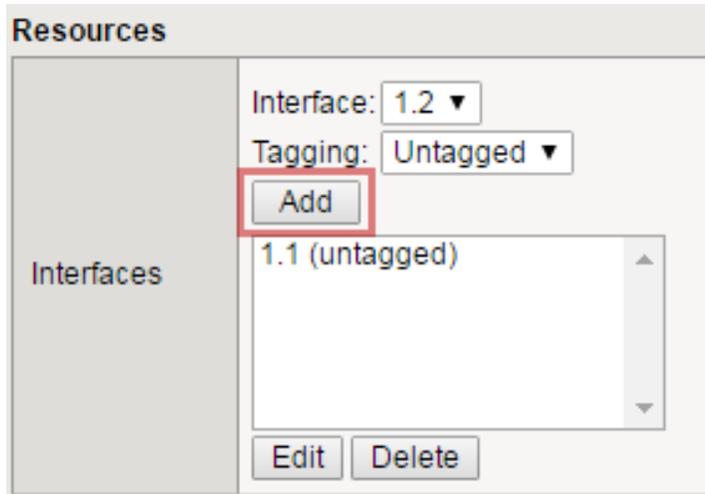
Creating VLANs

You will need create two untagged VLANs, one client-side VLAN (**client_vlan**) and one server-side VLAN (**server_vlan**) for the devices in your network.

1. From the sidebar select **Network >> VLANs** then select **Create**



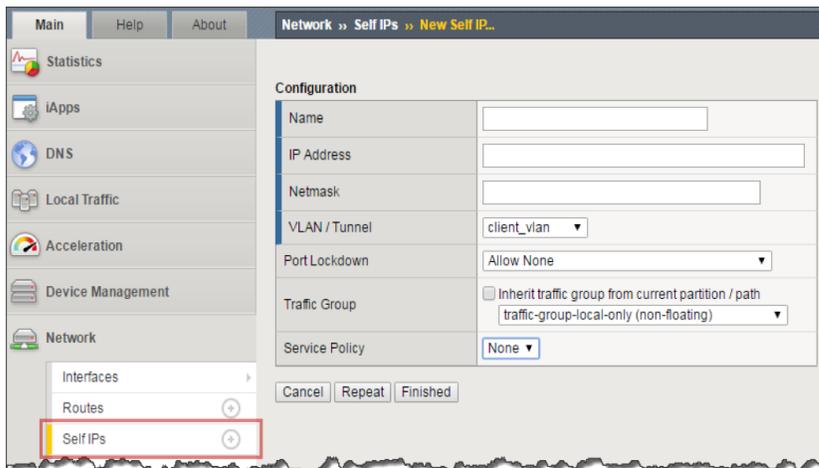
- (a) Under **General Properties**:
- i. **Name**: client_vlan
- (b) The name is for management purposes only, you could name them after your children or pets
- i. **Tag**: <leave blank>
 - A. Entering a tag is only required for “**Tagged**” (802.1q interfaces. “**Untagged**” interfaces will automatically get a tag which is used for internal L2 segmentation of traffic.
- (c) Under **Resources** in the **Interfaces** section:
- i. **Interface**: 1.1
 - ii. **Tagging**: Untagged
 - iii. Select the **Add** button. Leave all other items at the default setting.



When you have completed your VLAN configuration, hit the **Finished** button
 Create another untagged VLAN named **server_vlan** on interface **1.2**.

Assigning a Self IP addresses to your VLANs

Go to **Network >> Self IPs**, select **Create**.



1. Create a new self IP, for the **server_vlan** and **client_vlan** VLANs.

(a) In **Network >> Self IPs >> New Self IP**, under **Configuration** enter:

```

**Server-Side                               Client-side**

**Name** :                                   server_ip           client_ip

**IP Address** :                             10.1.20.245         10.1.10.245

**Netmask** :                                255.255.255.0       255.255.255.0

**VLAN** :                                   server_vlan          client_vlan

**Port** **Lockdown** : Allow None         Allow None

```

The default “**Allow None**” means the Self IP would respond only to ICMP.

- (b) The “**Allow Defaults**” selection opens the following on the self IP of the VLAN
 - i. TCP: ssh, domain, snmp, https
 - ii. TCP: 4353, 6699 (for F5 protocols, such as HA and iQuery)
 - iii. UDP: 520, cap, domain, f5-iquery, snmp
 - iv. PROTOCOL: ospf

Note: Even with “**Allow None**” chosen, traffic destined for a virtual server or object on the F5 (e.g. NAT) are able to pass through without issue as any object created on the F5 is by default allowed to pass through.

When you have completed your self-IP configuration, hit the **Finished** button. You should have something similar to the following

<input checked="" type="checkbox"/>	Name	Application	IP Address	Netmask	VLAN / Tunnel	Traffic Group	Par
<input type="checkbox"/>	client_ip		10.1.10.245	255.255.255.0	client_vlan	traffic-group-local-only	Comm
<input type="checkbox"/>	server_ip		10.1.20.245	255.255.255.0	server_vlan	traffic-group-local-only	Comm

Assigning the Default Gateway

1. Go to **Network > Routes** and then **Add**.
 - (a) Here is where we assign our default gateway (and other static routes as desired)

Properties	
Name	default_gateway
Description	
Destination	0.0.0.0
Netmask	0.0.0.0
Resource	Use Gateway...
Gateway Address	IP Address 10.1.10.1
MTU	

- (b) Under **Properties**

- i. **Name:** default_gateway
- ii. **Destination:** 0.0.0.0
- iii. **Netmask:** 0.0.0.0
- iv. **Resource:** Use Gateway...
- v. **Gateway Address:** 10.1.10.1
- vi. When you have completed defining your default gateway, hit the **Finished** button

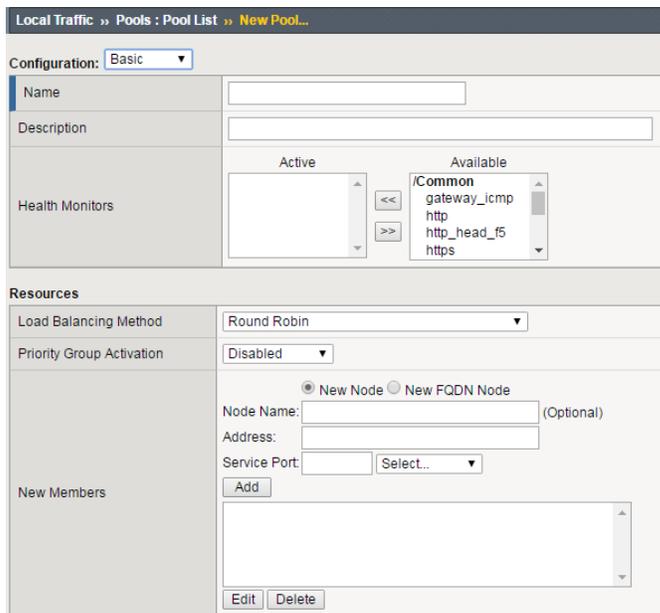
2. Verify your network configuration

- (a) Ping your client-side self IP (**10.1.10.245**) to verify connectivity
- (b) Use an SSH utility, such as puTTY, to access your BIG-IP management port at 10.1.1.245.
 - i. User: **root** Password: **default**
 - ii. Ping your default gateway, 10.1.10.1
 - iii. Ping a web server at 10.1.20.11.

Creating Pools

In this lab we will build a pool and virtual server to support our web site and verify our configurations by accessing our web servers through the BIG-IP. Verification will be performed visually and through various statistical interfaces.

- 1. From the sidebar, select **Local Traffic >> Pools** then select **Create**. Here we will create our new pool



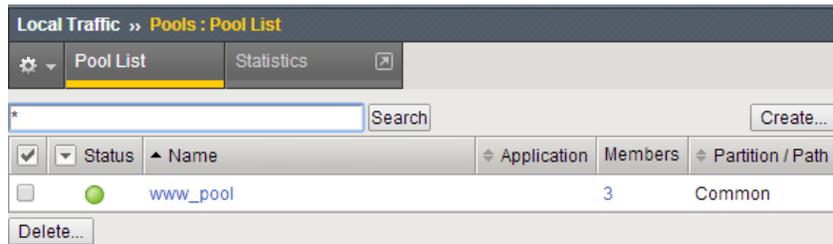
- (a) Under **Configuration:**
 - i. **Name:** www_pool
 - A. The name is for management purposes only, no spaces can be used
 - ii. **Description:** <optional>
 - iii. **Health Monitor:** http

(b) Under **Members:**

- i. **Load Balancing Method:** <leave at the default Round Robin>
- ii. **Priority Group Activation:** <leave at default>
- iii. **New Members:**

Address	Service Port
10.1.20.11	80
10.1.20.12	80
10.1.20.13	80

iv. As you enter each IP address and port combination, hit the **Add** button

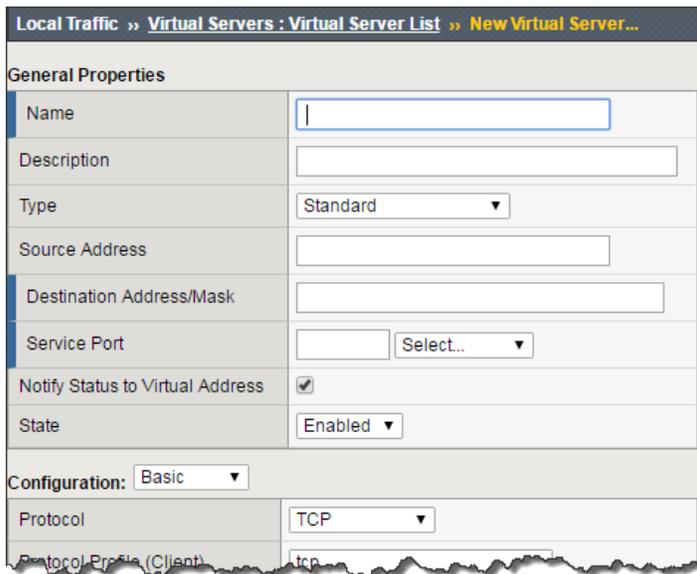


When you have completed your pool configuration, hit the **Finished** button

Creating Virtual Servers

Now let's build our virtual server

- 1. Under **Local Traffic >> Virtual Servers**, click the “+” icon



(a) Under **General Properties**

- i. **Name:** www_vs
- ii. **Description:** <optional>

- iii. **Type:** Standard
- iv. **Source/Address:** <leave blank>

Note: The default is 0.0.0.0/0, all source IP address are allowed

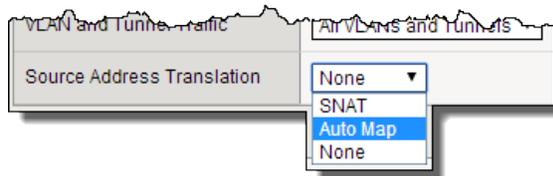
- v. **Destination Address/Mask:** 10.1.10.100

Note: The default mask is /32

- vi. **Service Port:** 80 or HTTP

(b) Under **Configurations**

- i. The web servers do not use the BIG-IP LTM as the default gateway. This means return traffic will route around the BIG-IP LTM and the TCP handshake will fail. To prevent this we can configure SNAT Automap on the Virtual Server. This will translate the client IP to the self IP of the egress VLAN and ensure the response returns to the BIG-IP.
- ii. **Source Address Translation:** Auto Map

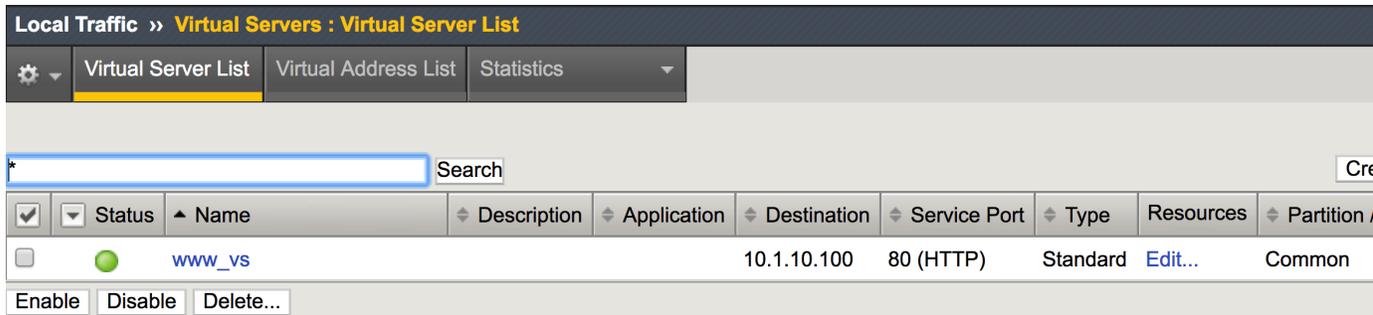


(c) Under **Resources**

- i. **iRules:** none
- ii. **Default Pool:** From the drop down menu, select the pool (**www_pool**) which you created earlier
- iii. **Default Persistence Profile:** None
- iv. **Fallback Persistence Profile:** None

When you have completed your virtual server configuration, hit the **Finished** button

You have now created a Virtual Server (Note: Items in blue are links)



1. Now let's see if our virtual server works!

- (a) Open the browser to the Virtual Server you just created

- (b) Refresh the browser screen several times (use “<ctrl>” F5)

Source: Node #3
F5 vLab Test Web Site

[Welcome](#) to F5 Networks and the F5E vLab Test Web Site. This Web site is designed to be used with F5 vLab (virtual environment) hands-on exercises and customer demonstrations.

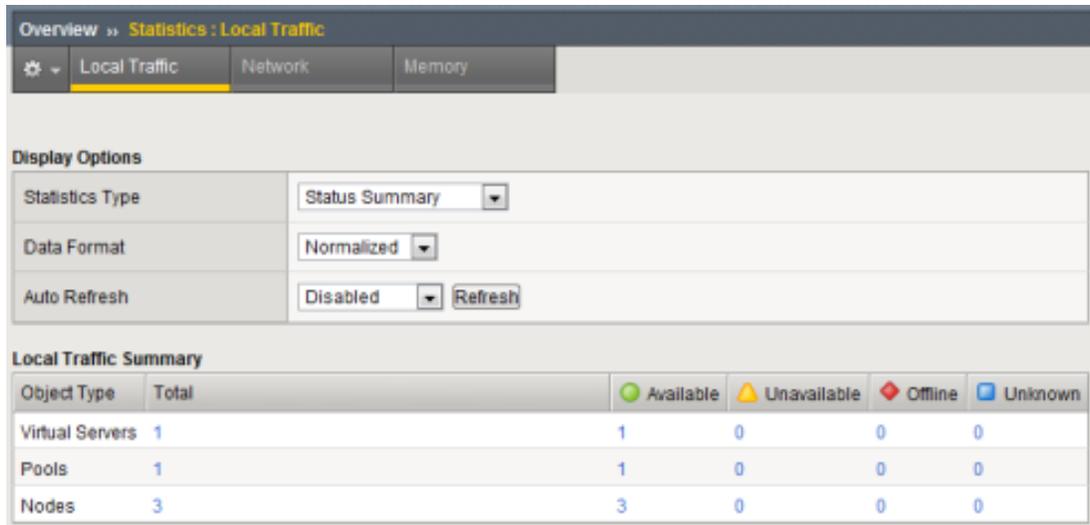


F5 Worldwide
Field Readiness
Node #1

Request Details

The *index.php* page is from **Node #1**
 Virtual server address: 10.1.10.100
 Pool member address/port: **10.1.20.11:80**
 Client IP address/port: 10.1.20.245:54432
 Requested URI: /

- Go to your BIG-IP and view the statistics for the **www_vs** virtual server and the **www_pool** pool and its associated members
- Go to **Statistics > Module Statistics > Local Traffic**
 - Choose **Virtual Servers** from drop down



Overview » Statistics : Local Traffic

Local Traffic | Network | Memory

Display Options

Statistics Type: Status Summary
 Data Format: Normalized
 Auto Refresh: Disabled Refresh

Object Type	Total	Available	Unavailable	Offline	Unknown
Virtual Servers	1	1	0	0	0
Pools	1	1	0	0	0
Nodes	3	3	0	0	0

- Go to **Local Traffic >> Virtual Servers >> Statistics**
- Go to **Local Traffic >> Pools >> Statistics**
 - Did each pool member receive the same number of connections?
 - Did each pool member receive approximately the same number of bytes?
 - Note the Source and Destination address when you go to directly and through the virtual server

- Let's archive our configuration in case we have to fall back later.
 - Go to **System >> Archives** and select **Create**.
 - * Name your archive **lab2_the_basics_net_pool_vs**

3.2.3 Lab 3: Load Balancing, Monitoring and Persistence

Objectives:

- Configure and review Ratio load balancing
- Build and test priority groups
- Build a content monitor that looks for a receive string and requires authentication
- Build and review simple (source IP) persistence and cookie persistence.

Ratio Load Balancing

1. Go to Local **Traffic >> Pools** and select **www_pool** and then **Members** from the top bar or you could click on the **Members** link in the Pool List screen.

<input checked="" type="checkbox"/>	Status	Name	Description	Application	Members	Partition / Path
<input type="checkbox"/>	●	fqdn_pool			4	Common
<input type="checkbox"/>	●	www_pool			3	Common

Note: When we created the pool, we performed all of our configurations on one page, but when we modify a pool the **Resource** information is under the **Members** tab

2. Under **Load Balancing** section
 - (a) Change the **Load Balancing Method** to **Ratio (Member)**
 - (b) As you look at the drop-down menu, notice most load balancing methods have two options: (Node) or (Member). Remember the difference?

Local Traffic » Pools : Pool List » www_pool

Properties Members Statistics

Load Balancing

Load Balancing Method
Ignore Persisted Weight
Priority Group Activation

Update

Current Members

Status Member

	Status	Member	Address	Service Port
<input type="checkbox"/>	●	10.1.20.11:80	10.1.20.11	80
<input type="checkbox"/>	●	10.1.20.12:80	10.1.20.12	80
<input type="checkbox"/>	●	10.1.20.13:80	10.1.20.13	80

Enable Disable Force Offline

Round Robin
 Ratio (member)
 Least Connections (member)
 Observed (member)
 Predictive (member)
 Ratio (node)
 Least Connections (node)
 Fastest (node)
 Observed (node)
 Predictive (node)
 Dynamic Ratio (node)
 Fastest (application)
 Least Sessions
 Dynamic Ratio (member)
 Weighted Least Connections (member)
 Weighted Least Connections (node)
 Ratio (session)
 Ratio Least Connections (member)
 Ratio Least Connections (node)

- (c) Don't forget the **Update** button
- (d) Then under **Current Members**
 - i. Select the first member in the pool **10.1.20.11:80**.
 - ii. Under the **Configuration** section
 - A. Change the **Ratio** of the member to 3

Local Traffic » Pools : Pool List » www_pool

Properties Members Statistics

Member Properties

Node Name	10.1.20.11
Address	10.1.20.11
Service Port	80
Partition / Path	Common
Description	
Parent Node	<input checked="" type="checkbox"/> 10.1.20.11
Availability	● Available (Enabled) - Pool member is available 2018-05-29 16:56:28
Health Monitors	● http
Monitor Logging	<input type="checkbox"/> Enable
Current Connections	0
State	<input checked="" type="radio"/> Enabled (All traffic allowed) <input type="radio"/> Disabled (Only persistent or active connections allowed) <input type="radio"/> Forced Offline (Only active connections allowed)

Configuration: Basic

Ratio	3
Priority Group	0
Connection Limit	0
Connection Rate Limit	0

- (e) Select the **Update** button

3. Verification

- (a) Check the pool statistics by selecting **Statistics** on the top bar, if you are still in **Local Traffic >> Pools**, or by going to **Statistics >> Module Statistics >> Local Traffic** and selecting **Pool** from **Statistics Type**.
- (b) Reset the statistics for your **www_pool** pool by checking the boxes next to the pool members and hitting the **Reset** button
 - i. Browse to your **www_vs (10.1.10.100)** virtual server
 - ii. Refresh the browser screen several times (use “<ctrl>” F5)
 - iii. Select the **Refresh** button on the **Statistics** screen
 - iv. How many total connections has each member taken?
 - v. Is the ratio of connections correct between the members?
- (c) Now go back and put the pool load balancing method back to Round Robin
 - i. Reset the statistics
 - ii. Refresh the virtual server page several times
 - iii. Refresh the statistics
 - iv. Does the ratio setting have any impact now?

Priority Groups Lab

Let's look at priority groups. In this scenario we will treat the **.13** server as if it were in a disaster recovery site that can be reached over a backhaul. The customer would like to maintain at least two members in the pool for redundancy and load. They would find this beneficial to allow connections to proceed during a maintenance window or during an outage.

1. Go to **Local Traffic >> Pools >> www_pool**

- (a) Select the **Members** tab.
 - i. Set the **Load Balancing Method** back to **Round Robin**
 - ii. Set the **Priority Group Activation** to **Less than ... 2 Available Members**.

Local Traffic >> Pools : Pool List >> www_pool

Properties Members Statistics

Load Balancing

Load Balancing Method: Round Robin

Priority Group Activation: Less than... 2 Available Member(s)

Update

Current Members

<input type="checkbox"/>	Status	Member	Address	Service Port	FQDN	Ephemeral	Ratio	Priority Group	Conn
<input type="checkbox"/>	●	10.1.20.11:80	10.1.20.11	80		No	1	0 (Active)	0
<input type="checkbox"/>	●	10.1.20.12:80	10.1.20.12	80		No	1	0 (Active)	0
<input type="checkbox"/>	●	10.1.20.13:80	10.1.20.13	80		No	1	0 (Active)	0

Enable Disable Force Offline Remove

- (b) Don't forget to hit the **Update** button
- (c) Select the pool members **10.128.20.11** and **10.128.20.12** and set their **Priority Group** to **2**.
 - i. This will allow you to change the priority on that particular member.

The screenshot shows the configuration page for a pool member. The 'Members' tab is selected. The 'Member Properties' section includes fields for Node Name, Address, Service Port, Partition / Path, Description, Parent Node, Availability, Health Monitors, Monitor Logging, Current Connections, and State. The 'Configuration' section includes fields for Ratio, Priority Group, Connection Limit, and Connection Rate Limit. The 'Priority Group' field is highlighted with a red box and contains the value '2'. Below the configuration fields are 'Update' and 'Delete' buttons.

2. Review your settings and let's see how load balancing reacts now
 - (a) Select the **Statistics** tab.
 - (b) Reset the pool statistics.
 - (c) Browse to your virtual server and refresh several times.
 - (d) Refresh your statistics.
 - (e) Are all members taking connections?
 - (f) Which member isn't taking connections?
3. Let's simulate a maintenance window or an outage by disabling a pool member in the highest priority group (2).

Note: F5 ranks priority from low number to high number. This means, a priority of 1 has a lower priority than 2, and onwards.

This should cause priority group activation to kick in, since the number of active members in our high priority group has dropped below one.

4. Select the member in the Priority Group 2 and Disable that pool member.
 - (a) Select the **Disable** button

Local Traffic » Pools : Pool List » www_pool

Properties Members Statistics

Load Balancing

Load Balancing Method: Round Robin

Priority Group Activation: Less than... 2 Available Member(s)

Update

Current Members

Status	Member	Address	Service Port	FQDN	Ephemeral	Ratio	Priority Group	Connection Limit
<input checked="" type="checkbox"/>	10.1.20.11:80	10.1.20.11	80		No	1	2 (Active)	0
<input type="checkbox"/>	10.1.20.12:80	10.1.20.12	80		No	1	2 (Active)	0
<input type="checkbox"/>	10.1.20.13:80	10.1.20.13	80		No	1	0 (Active)	0

Enable **Disable** Force Offline Remove

(b) The status indicator now goes to black, indicating the member has been disabled

- Once again, select **Statistics**, reset the pool statistics, browse to the virtual server and see which pool members are taking hits now.

Once you are done testing re-enable your disabled pool member.

Monitor Labs

Objective:

- Build a default monitor for nodes
- Build a content monitor for your pool

Default Monitors

- Go to **Local Traffic >> Nodes**, note the status of the nodes.
 - Note that the nodes exist in this table, even though they were never specifically configured in the Node section of the GUI. Each time a unique IP address is placed in a pool a corresponding node entry is added and assigned the default monitor (if any).
 - Select the **Default Monitors** tab.

Local Traffic » Nodes : Default Monitor

Node List Default Monitor Statistics

Configuration: Basic

Health Monitors

Active

Available

Common

- gateway_icmp
- https_443
- icmp
- real_server

Update

- Notice we have several options. For nodes you will want a generic monitor, so we will choose **icmp**.

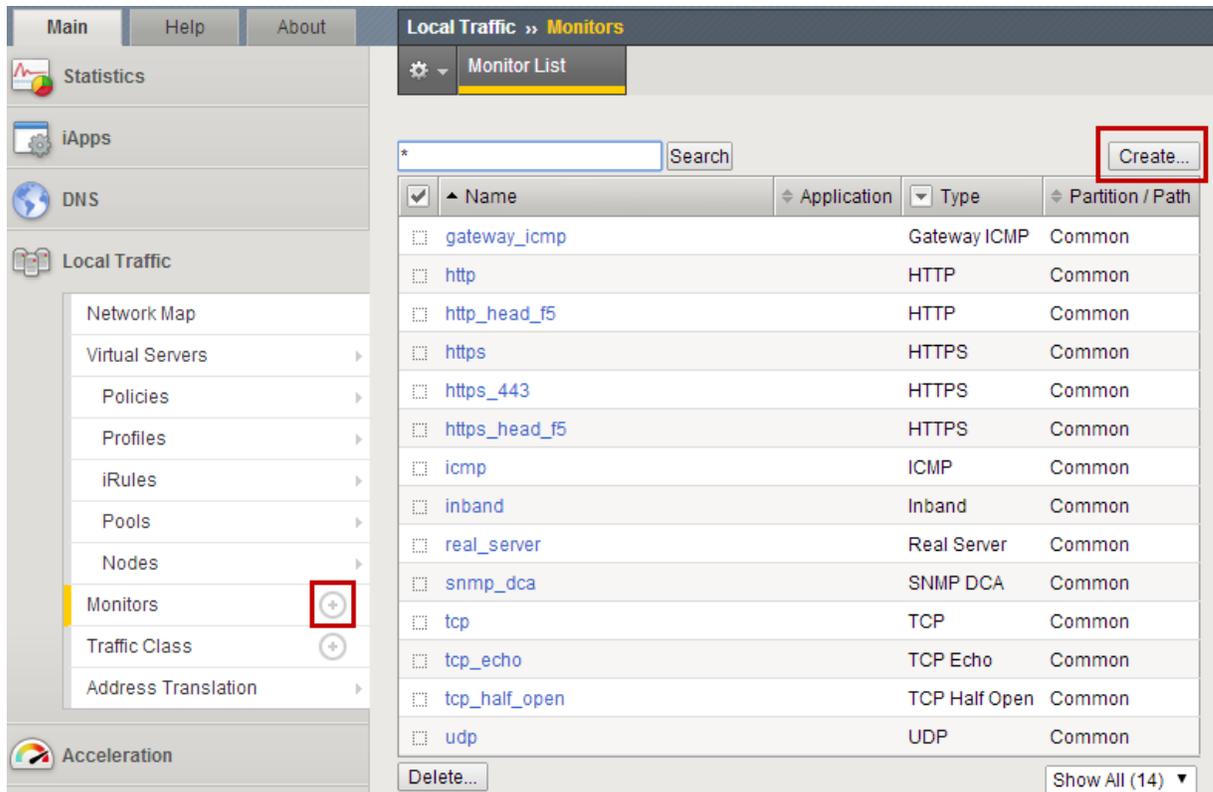
- (d) Select **icmp** from the **Available** box and hit  to place it in the **Active** box.
 - (e) Click on the **Update** button to finalize your changes.
2. Select **Node List** or **Statistics** from the top tab.
 - (a) What are your node statuses?
 3. Select **Statistics >> Module Statistics >> Local Traffic**
 - (a) What are the statuses of your nodes, pool and virtual server?

For those of you who did the **FQDN Pool** extra credit lab, you will notice your FQDN in the node list. The status should be **Available** (Green) even though there wasn't a monitor. This is because a good status indicates the BIG-IP successfully queried the DNS server for the name. Click on the FQDN node to see additional options, such as query interval.

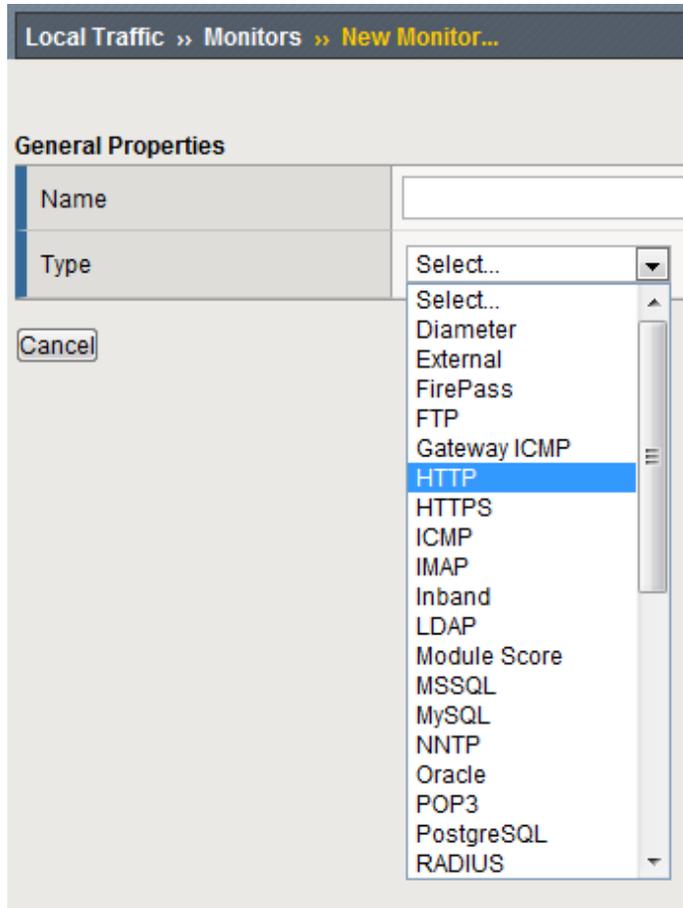
Content Monitors

The default monitor simply tells us the IP address is accessible, but we really don't know the status of the particular application the node supports. We are now going to create a monitor to specifically test the application we are interested in. We are going to check our web site and its basic authentication capabilities.

1. Browse to **http://10.1.10.100** and on the web page select the **Basic Authentication** link under **Authentication Examples**.
 - (a) User: **user.1**
 - (b) Password: **password**
 - (c) You could use text from this page or text within the source code to test for availability. You could also use HTTP statuses or header information. You will be looking for the HTTP status "**200 OK**" as your receive string to determine availability.
 - (d) Note the URI is **/basic**. You will need this for your monitor.
2. Select **Local Traffic >> Monitor** on the side-bar and select the plus (+) sign or **Create**



- (a) Now we can create a monitor to check the content of our web page to ensure things are running properly.
- i. **Name:** www_test
 - ii. **Type:** HTTP



- (b) Once you have selected your parent (**Type**) monitor, you can access the **Configuration** section
- i. **Send String**: Enter the command to retrieve the page you want “**GET /basic/r/n**” (no quotes)
 - ii. In the Receive String box put “**200 OK**” (no quotes)

Note: The receive string is not case sensitive.

- iii. Enter **user.1/password** for the **Username** and **Password**

Local Traffic >> Monitors >> New Monitor...

General Properties

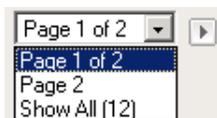
Name	www_test
Description	
Type	HTTP
Parent Monitor	http

Configuration: Basic

Interval	5 seconds
Timeout	16 seconds
Send String	GET /basic/\r\n
Receive String	200 OK
Receive Disable String	
User Name	user.1
Password
Reverse	<input type="radio"/> Yes <input checked="" type="radio"/> No
Transparent	<input type="radio"/> Yes <input checked="" type="radio"/> No
Alias Address	* All Addresses
Alias Service Port	* * All Ports

(c) Click **Finish** and you will be taken back to **Local Traffic >> Monitors**

3. Where is your new Monitor?



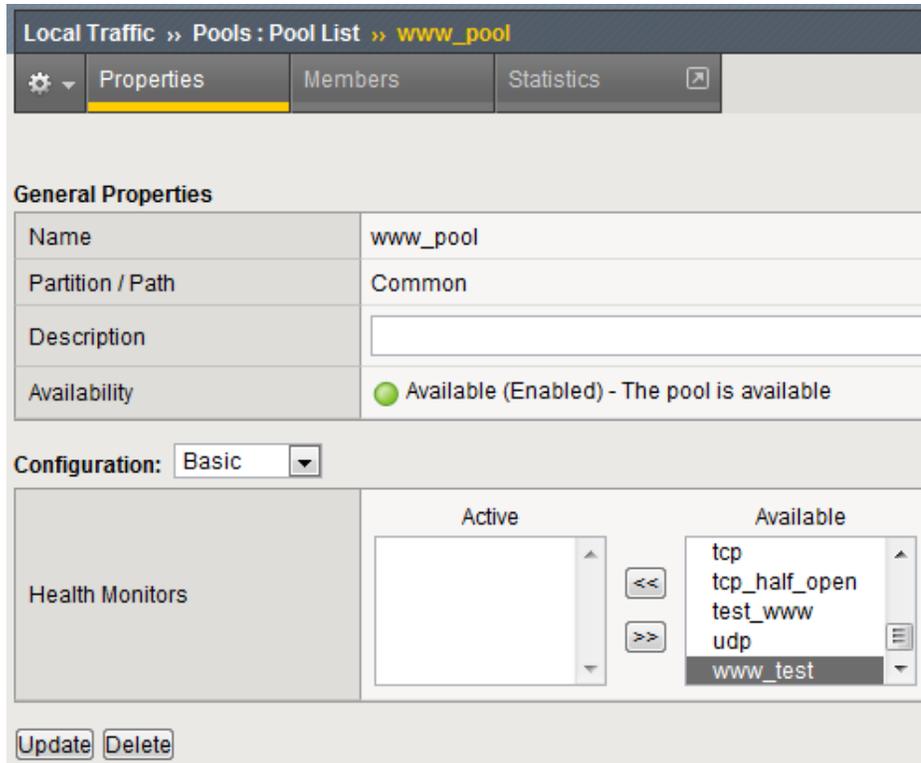
(a) **Hint:** Check the lower right hand corner of the Monitors list. Here you can go to the next page or view all Monitors

(b) You can change the number of records displayed per page in **System >> Preferences**

4. Go to **Local Traffic >> Pools >> www_pool** and choose **Properties** from the top bar

(a) Remove the **http** monitor from the Active box.

(b) Select the **www_test** monitor from the Available monitor's window in the **Configuration** section and move it to the Active window.



5. Once you have selected your parent (Type) monitor, you can access the **Configuration** section
 - (a) Select **Statistics** from the tabs.
 - (b) What is the status of the pool and its members?
6. Go to **Local Traffic >> Virtual Servers**. What is the status of your virtual server?
 - (a) Browse to your **www_vs** virtual server. Which members are taking traffic?
 - (b) Just for fun reverse the monitor. Now when **200 OK** is returned it indicates the server is not responding successfully. You can see where this would be useful if you were looking for a 404 (bad page) response.

Monitor Testing

There is now the ability to test monitors. This is tremendously helpful as you no longer need to create monitors and add them to false objects on the BIG-IP. The functionality is now built in to the monitor itself to be less invasive on your infrastructure, and less time consuming all together.

1. Go to **Local Traffic >> Pools >> www_pool**
 - (a) Under **Configuration**, move the active monitor to **Available**
2. Go to **Monitors** and click on **http**
 - (a) Click the **Test** tab
 - (b) Under **Address** plug in **10.1.20.11** and in the port field plug in **80**
 - (c) Click **Test**

Local Traffic » Monitors » http

Properties Instances **Test**

Test

Address :

Result

```

-----
LTM::Monitor /Common/http
-----
Monitor Test Result:
Destination: 10.1.20.11:80
Last state: up
State time: 2018.06.01 09:38:54 (0hr:0min:10sec ago)
Last result: Test Completed (UP)
See /var/log/monitors/Common_http-Common_monitor_test-80.log

```

- (d) Go back to **Local Traffic >> Pools >> www_pool**
- i. Once here, move **http** back to **Active**

Persistence Labs

In this lab we will configure a couple types of persistence and view their behavior. For persistence, profiles will have to be created and attached to our virtual server.

Lab Requirements:

- Prior to beginning the lab verify your **www_pool** has been set to the following parameters:
 - **Load Balancing Method:** Round Robin
 - **Priority Group Activation:** Disable
 - * The members **Ratio** and **Priority Group** mean nothing since we aren't using Ratio load balancing and Priority Groups are disabled.
 - Hit **Update**
 - Hit your virtual server several times, you should see all 3 servers respond.

Simple (Source Address) Persistence

1. Go to **Local Traffic >> Profiles** and select the **Persistence** tab.
 - (a) From the **Persistence Profiles** screen select the **Create** button.

Local Traffic » Profiles : Persistence

Services Content Databases Persistence Protocol

SSL Authentication Message Routing Other

* Search Create

<input checked="" type="checkbox"/>	Name	Application	Type	Parent Profile	Partition / Pat
<input type="checkbox"/>	cookie		Cookie	(none)	Common
<input type="checkbox"/>	dest_addr		Destination Address Affinity	(none)	Common
<input type="checkbox"/>	hash		Hash	(none)	Common
<input type="checkbox"/>	msrdp		Microsoft® Remote Desktop	(none)	Common
<input type="checkbox"/>	sip_info		SIP	(none)	Common
<input type="checkbox"/>	source_addr		Source Address Affinity	(none)	Common
<input type="checkbox"/>	ssl		SSL	(none)	Common
<input type="checkbox"/>	universal		Universal	(none)	Common

(b) At the **New Persistence Profile** screen enter:

- i. **Name:** my-src-persist
- ii. **Persistence Type:** Source Address Affinity

General Properties

Name	<input type="text"/>
Persistence Type	Source Address Affinity ▼
Parent Profile	source_addr ▼

Configuration Custom

Match Across Services	<input type="checkbox"/>	<input type="checkbox"/>
Match Across Virtual Servers	<input type="checkbox"/>	<input type="checkbox"/>
Match Across Pools	<input type="checkbox"/>	<input type="checkbox"/>
Hash Algorithm	Default ▼	<input type="checkbox"/>
Timeout	Specify... ▼ 180	seconds <input type="checkbox"/>
Prefix Length	None ▼	<input type="checkbox"/>
Map Proxies	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/>
Override Connection Limit	<input type="checkbox"/>	<input type="checkbox"/>

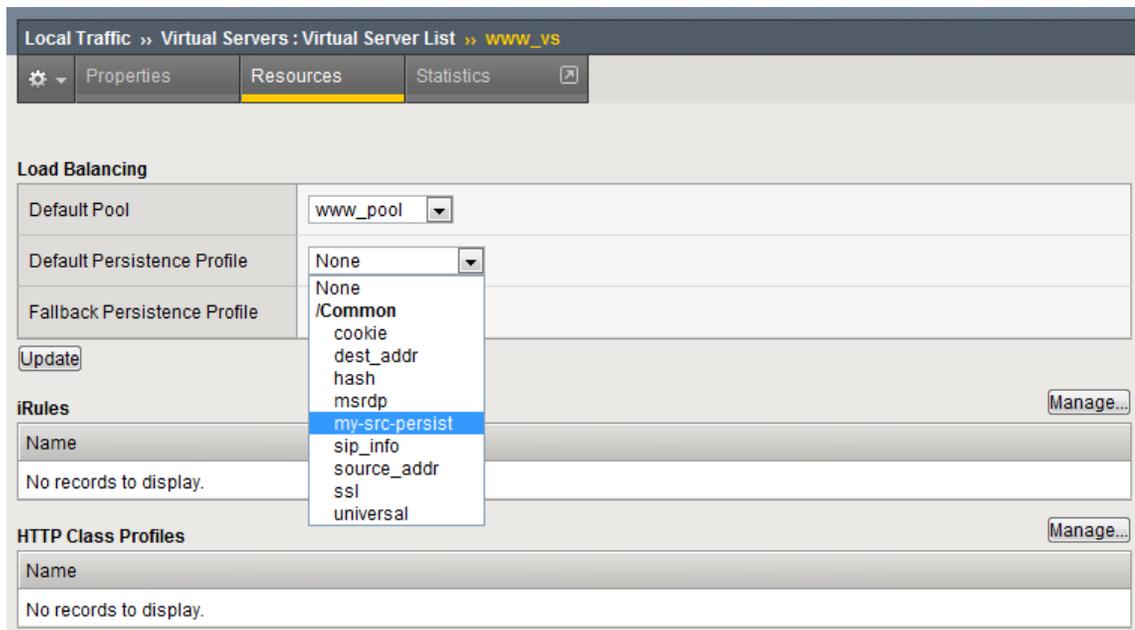
Cancel Repeat Finished

(c) This will add the **Configuration** section to the **General Properties** section.

- i. Note the parent profile.

(d) In the **Configuration** section, set the

- i. **Timeout:** 60 seconds
 - ii. **Prefix Length:** None
 - A. This is the default and is a /32 prefix (255.255.255.255 mask).
 - B. Each new IP address will create a new persistence record.
 - iii. **Hint:** You can't change these settings until you have checked the Custom box. This prevents unwanted or unauthorized changes from within the GUI, without explicitly allowing it. Also, it allows you to know what has changed from the default settings.
- (e) You have just created your first custom Profile.
- i. Note the check box for your new custom profile isn't grayed out and can be selected to allow you to delete the profile if desired.
2. Now let's attach our new profile to the virtual server.
- (a) Go to **Local Traffic >> Virtual Server** and
 - i. Select **www_vs** and the **Resources** tab or
 - ii. Take the shortcut directly to the **Resources** of the virtual server. (Can you find it?)
 - (b) Set the **Default Persistence Profile** to **my-src-persist**.



- (c) Don't forget to **Update** before leaving the page. (*Be careful, the reminders will stop!*)
- (d) Testing Source Address Affinity
 - i. At this point you may want to open a second browser window to the management GUI.
 - ii. From one management window go to **Statistics >> Module Statistic >> Local Traffic**
 - iii. Select **Persistence Records** for the **Statistics Type** menu

Statistics » Module Statistics : Local Traffic

Traffic Summary ▾ **Local Traffic** Network ▾ Memory

Display Options

Statistics Type Persistence Records ▾
 Data Format
 Auto Refresh

Local Traffic Summary

Object Type	Total	Available	Unavailable	Offline	Unknown
Virtual Servers	1	1	0	0	0
Pools	1	1	0	0	0
Nodes	3	3	0	0	0

Persistence Records ▾
 Status Summary
 Virtual Servers
 Virtual Addresses
 Policies
 Profiles Summary
 Profiles - Statistics
 Pools
 iRules
 Nodes
 SNATs
 SNAT Pools
 SNAT Translations
 NATs
Persistence Records
 DNS Express Zones
 DNS Cache

3. At this point you will see that the Persistence Records statistics display has been disabled (way back in v12.1). A TMSH database command is required to activate it.
 - (a) SSH to you BIG-IP at 10.1.1.245. Username: **root** Password: **default**
 - (b) At the prompt enter: **tmsH**
 - (c) At the TMSH prompt enter the command in the **Persistence Value** GUI.
 - i. **modify sys db ui.statistics.modulestatistics.localtraffic.persistence records value true**
 - A. Tab completion will make this a little easier
4. Now, in this window you can watch your persistence records. You may want to set **Auto Refresh** to 20 seconds.

Statistics » Module Statistics : Local Traffic » Persistence Records

Traffic Summary ▾ **Local Traffic** Subscriber Management Network Memory System

Display Options

Statistics Type Persistence Records ⇅
 Data Format Normalized ⇅
 Auto Refresh Disabled ⇅ Refresh

* Search

▲ Persistence Value	▼ Persistence Mode	Virtual Server	Pool	Pool Mem
10.1.10.51	Source Address Affinity	www_vs	www_pool	10.1.20.12:80

5. In your other management GUI window go to **www_pool** and clear the member statistics.
 - (a) Open a browser session to your virtual server and refresh several times.
 - (b) How many members are taking traffic?
 - (c) Check you **Persists Records** window. Are there any persistence records?
 - i. If you are not Auto Refreshing, don't forget to hit **Refresh**

(d) Refresh your web page prior to the **Age column** reaching **60**. What happens?

Cookie Persistence (Cookie Insert)

1. Go to **Local Traffic >> Profiles >> Persistence** tab and hit **Create**
 - (a) Let's name our profile **my_cookie_insert** (original isn't it)
 - (b) Our **Persistence Type** will be **Cookie**
 - (c) This brings us to the **Configuration** section.

General Properties	
Name	<input type="text"/>
Persistence Type	Cookie ▼
Parent Profile	cookie ▼

Configuration		Custom <input type="checkbox"/>
Cookie Method	HTTP Cookie Insert ▼	<input type="checkbox"/>
Cookie Name	<input type="text"/>	<input type="checkbox"/>
HTTPOnly Attribute	Enabled ▼	<input type="checkbox"/>
Secure Attribute	Enabled ▼	<input type="checkbox"/>
Always Send Cookie	<input type="checkbox"/>	<input type="checkbox"/>
Expiration	<input checked="" type="checkbox"/> Session Cookie	<input type="checkbox"/>
Cookie Encryption Use Policy	disabled ▼	<input type="checkbox"/>
Encryption Passphrase	<input type="text"/>	<input type="checkbox"/>
Override Connection Limit	<input type="checkbox"/>	<input type="checkbox"/>

2. As you can see, the default **Cookie Method** is **HTTP Cookie Insert**, so we won't have to modify the **Cookie Method**
 - (a) The BIG-IP will also create a cookie name for you using a combination of "**BIGipServer**" and the pool name the virtual server service. We will take this default also.
 - (b) We will use a **session** cookie. Which means the cookie is deleted when the browser is closed.
 - (c) Select **Finished**
 - (d) Now attach your cookie persistence profile to your virtual server's **Default Persistence Profile** by:
 - i. Go to **Local Traffic >> Virtual Server >> www_vs >> Resources** tab
 - ii. Set the **Default Persistence Profile** to **my_cookie_insert**
 - iii. Hit **Update**

(e) Whoa! Did you just get this error message?

The screenshot shows a web interface for configuring a virtual server. At the top, there is a breadcrumb trail: "Local Traffic >> Virtual Servers : Virtual Server List >> www_vs". Below this, there are tabs for "Properties", "Resources", and "Statistics". The "Resources" tab is currently selected. A yellow warning icon is displayed with the message: "01070309:3: Cookie persistence requires an HTTP or FastHTTP profile to be associated with the virtual server". Below the error message, the "Load Balancing" section is visible. It contains three dropdown menus: "Default Pool" set to "www_pool", "Default Persistence Profile" set to "cookie", and "Fallback Persistence Profile" set to "None". An "Update" button is located at the bottom of this section.

- (f) Remember what we said earlier about some Profiles requiring prerequisite Profiles? Since we are looking in the HTTP header for the cookie the prerequisite for the Cookie Profile is the HTTP profile.
3. We will have to go to the virtual server to add the HTTP profile, prior to adding the Cookie Persistence profile.
- (a) Select the **Properties** tab on your virtual server
 - (b) Go to **HTTP Profile** in the **Configuration** section and select the default HTTP (**http**) profile.

Local Traffic » Virtual Servers : Virtual Server List » **www_vs**

Properties
 Resources
 Statistics

General Properties

Name	www_vs
Partition / Path	Common
Description	
Type	Standard
Source Address	0.0.0.0/0
Destination Address/Mask	10.1.10.100
Service Port	80 HTTP
Notify Status to Virtual Address	<input checked="" type="checkbox"/>
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	<input checked="" type="checkbox"/> None <input type="checkbox"/> /Common <input type="checkbox"/> http <input type="checkbox"/> http-explicit <input type="checkbox"/> http-transparent
HTTP Proxy Connect Profile	
FTP Profile	
RTSP Profile	None

- (c) Hit the **Update** button
 - (d) Now we can go back to the **Resource** tab and add our cookie persistence profile.
4. Testing cookie persistence.
- (a) If you wish you can watch the member statistics to validate your persistence.
 - (b) Open a new browser session to your virtual server and refresh several times.
 - (c) Does the page ever change?
 - (d) Did you hit a different server?
 - (e) Refresh several times. Are you hitting the same server?
 - i. On the web page under **HTTP Request and Response Information** click the **Display Cookie** link.



Archive your work in the file: **lab3_lb_monitor_and_persist**

3.2.4 Lab 4: Accelerating Applications Lab

Objectives:

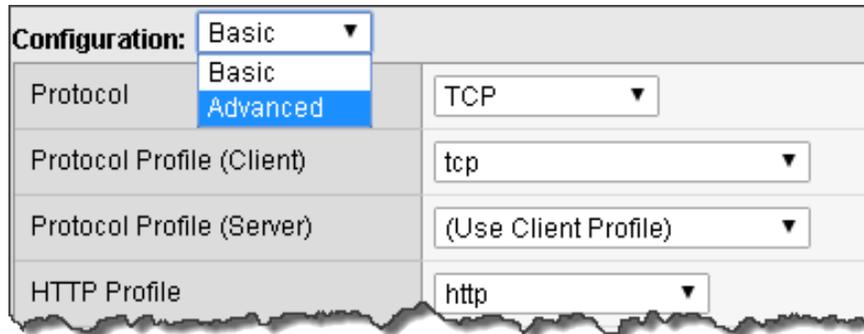
- Assign client-side and server-side profiles
- Set up caching for your web site
- Set up compression for your web site

Lab Prerequisites:

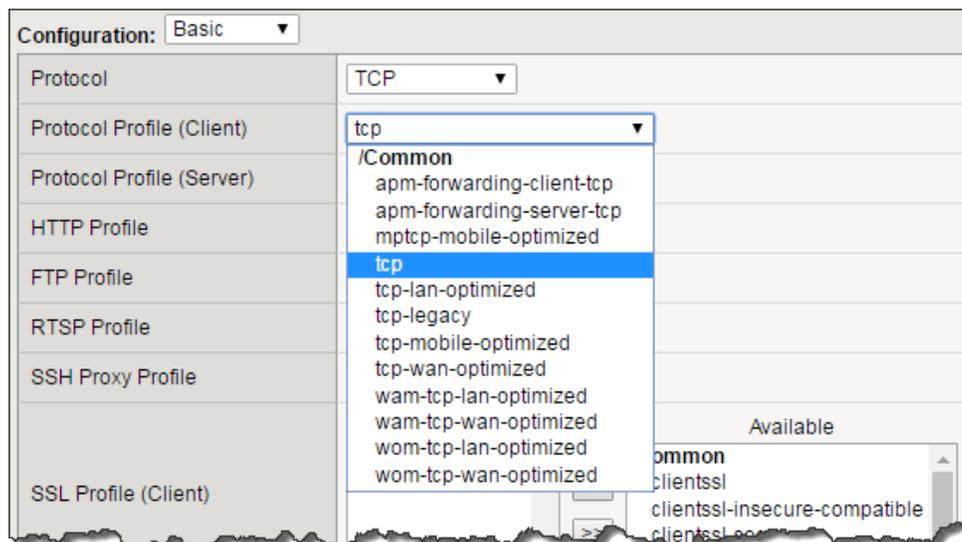
- Prior to starting this lab remove the cookie persistence profile from your virtual server.

TCP Express

1. Set client-side and server-side TCP profiles on your virtual server properties.
 - (a) In some earlier version you would be required to select the **Advanced** menu to see the Client and Server protocol profiles.



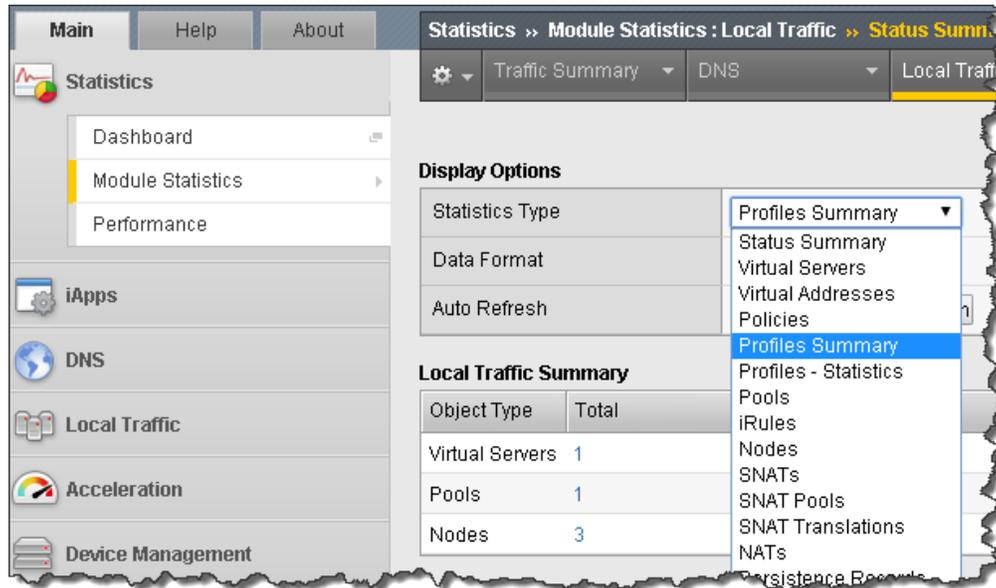
- (b) If you chose to use the **Advanced** menu you would see a whole array of new options. There are **Basic** and **Advanced** drop downs on many of the GUI menus. You can always see **Advanced** menus by changing the preferences in **System >> Preferences**.
- (c) From the drop-down menus place the **tcp-wan-optimized** profile on the client-side and the **tcp-lan-optimized** profile on the server-side.



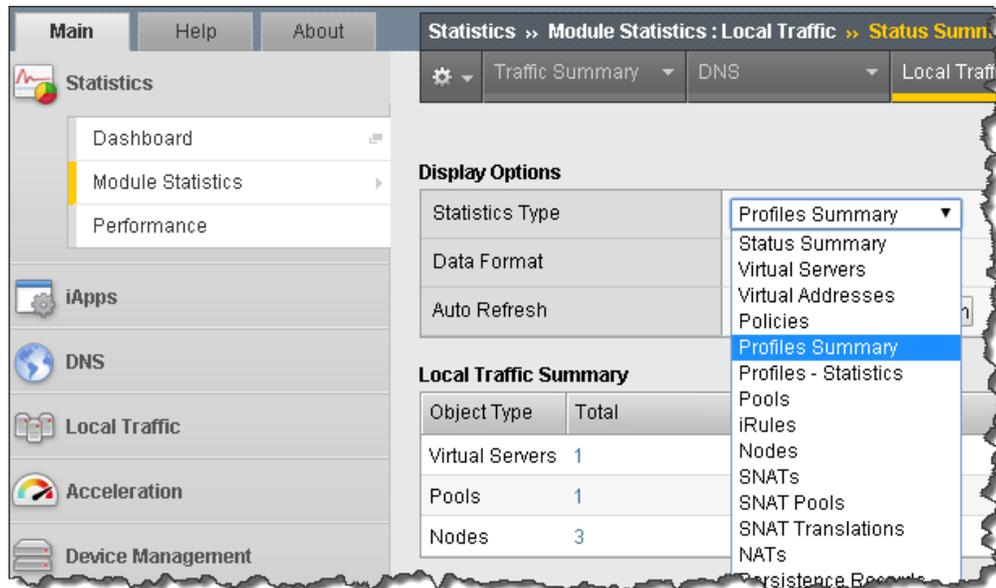
HTTP Optimization - RamCache Lab

1. Visit your virtual server's web page and refresh it several times. Note the Source Node for the pictures of the BIG-IPs. They change depending on where the connection is coming from. The Source Node information is part of the picture.
2. Go to **Local Traffic >> Profiles >> Services >> Web Acceleration** or **Acceleration >> Profiles >> Web Acceleration**
 - (a) Create a new profile named **www-opt-caching** using **optimized-caching** as the Parent Profile.
 - (b) Take all the defaults, no other changes are required.
3. Open up your **www_vs** virtual server.
 - (a) At the **HTTP Profile** drop down menu make sure **http** is selected.
 - (b) Under **Acceleration** at **Web Acceleration Profile** select your new caching profile; **www-opt-caching**

- (c) Clear the statistics on your pool and refresh the main web page several times.
- The pictures do not change. Why do you think that is?
 - Go to your pool. Are all pool members taking connections?
4. Now go to **Statistics >> Module Statistics >> Local Traffic** on the sidebar. From the **Statistics Type** drop-down menu select **Profiles Summary**



5. Select the View link next to the **Web Acceleration** profile type



Global Profile Statistics	
Profile Type	Details
HTTP	View...
HTTP Compression	View...
Web Acceleration	View...
RTSP	View...

6. You can get more detailed information on RamCache entries at the CLI level
 - (a) Log onto the CLI of your BIG-IP via SSH using the root account (user: **root** password: **default**).
 - (b) At the CLI go into **tmsh** at the **(tmos)#** prompt
 - (c) At the shell prompt enter **show ltm profile ramcache www-opt-caching**

HTTP Optimization - HTTP Compression Lab

1. Go to **Local Traffic >> Profiles >> Service >> HTTP Compression** or **Acceleration >> Profiles >> Web Acceleration**
 - (a) Create a new profile, **www-compress**, using the **wan-optimized-compression** default profile.
2. Open up your **www_vs** virtual server.
 - (a) At the **HTTP Profile** drop down menu make sure **http** is selected
 - (b) At the **Web Acceleration** drop-down menu select **None**
 - i. *For the purpose of this lab we don't want caching interfering with our response headers*
 - (c) At the **HTTP Compression** drop-down menu select the HTTP compression profile you just created
3. Now open your virtual server's web page and under **Content Examples on This Host** select the **HTTP Compress Example** and **Plaintext Compress Example** link
 - (a) Now off to the statistics on the sidebar. Under the **Local Traffic** drop-down menu select **Profiles Summary**
 - (b) Select the **View** link next to the **HTTP Compression** profile type

Cache	
Cache Size (bytes)	179.1K
Total Cached Items	11
Total Evicted Items	0

Cache Hits / Misses	Count	Size (bytes)
Hits	31	909.9K
Misses (Cacheable)	7	183.4K
localb.stats.Misses(Unacheable)	17	5.5K

- (c) On the web page under **HTTP Request and Response Information** select the **Request and Response Headers** link.
 - i. Notice you no longer see the **Accept-Encoding** header in the **Request Headers Received at the Server** section

Archive your work in a file called: **lb4_acceleration**

3.2.5 Lab 5: SSL Offload and Security

In this Lab we will configure client-side SSL processing on the BIG-IP

Objective:

- Create a self-signed certificate
- Create a client SSL profile
- Modify your HTTP virtual server to use HTTPS
- Add additional security to your HTTPS web server using the HTTP profile

We will create a self-signed certificate and key for a client SSL profile to attach to our virtual server

Creating a Self-signed certificate and key

1. Go to **System >> Certificate Management >> Traffic Certificate Management >> SSL Certificate List** and select **Create**

<input type="checkbox"/>	Status	Name	Contents	Key Security	Common Name	Organization	Expiration	Partition
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ca-bundle	Certificate Bundle				Mar 4, 2035 - Oct 6, 2046	Common
<input type="checkbox"/>	<input type="checkbox"/>	default	RSA Certificate & Key	Normal	localhost.localdomain	MyCompany	May 26, 2028	Common
<input type="checkbox"/>	<input type="checkbox"/>	f5-ca-bundle	RSA Certificate		Entrust Root Certificati...	Entrust, Inc.	Dec 7, 2030	Common
<input type="checkbox"/>	<input type="checkbox"/>	f5-irule	RSA Certificate		support.f5.com	F5 Networks	Aug 13, 2031	Common
<input type="checkbox"/>	<input type="checkbox"/>	f5_api_com	RSA Key	Password				Common

Note: The default key size is **2048**. You can save SSL resources on the **server-side** by lowering this key size

System » Certificate Management : Traffic Certificate Management : SSL Certificate List » **New SSL Certificate...**

General Properties

Name	<input type="text"/>
------	----------------------

Certificate Properties

Issuer	Self
Common Name	<input type="text"/>
Division	<input type="text"/>
Organization	<input type="text"/>
Locality	<input type="text"/>
State Or Province	<input type="text"/>
Country	United States <input type="text"/> US
E-mail Address	<input type="text"/>
Lifetime	365 days
Subject Alternative Name	<input type="text"/>

Key Properties

Key Type	RSA
Size	2048 bits

Cancel Finished

(a) Enter:

- i. **Name:** my-selfsigned-cert
- ii. **Issuer:** Self
- iii. **Common Name:** www.f5demo.com
- iv. Fill out the rest any way you would like

Creating SSL Client Profile

1. Go to **Local Traffic >> Profiles >> SSL >> Client** menu and select **Create**

- (a) Under **General Properties**
 - i. **Name:** my_clientssl_profile
- (b) Under **Configuration** in the **Certificate Key Chain** section, select the **Custom** box and hit **Add**
 - i. In the **Add SSL Certificate to Key Chain** pop-up select:
 - A. **Certificate:** my-selfsigned-cert
 - B. **Key:** my-selfsigned-cert
 - ii. Select **Add**

- (c) Hit **Finished**.

Building our New Secure Virtual Server

1. Go to **Local Traffic >> Virtual Servers** and hit the **Create** button or hit the “+” next to Virtual Servers
 - (a) **Name:** secure_vs
 - (b) **Destination Address/Mask:** 10.1.10.105
 - (c) **Port:** 443 or HTTPS
 - (d) **SSL Profile (Client):** my_clientssl_profile (the profile you just created)
 - (e) **Source Address Translation:** Auto Map (remember why we need this?)
 - (f) **Default Pool:** www_pool

- (g) Default all other settings. (Notice you did not require an HTTP profile)
 - (h) **Finish**
2. Test our secure server. Go to you **secure_vs** at **https://10.1.10.105**
 - (a) If you want to watch member traffic, go to the **www_pool** and reset the statistics
 - (b) Browse to your secure virtual server
 - (c) What port did your pool members see traffic on?

Securing Web Applications with the HTTP profile

1. Let's begin by creating a custom HTTP profile
 - (a) Go to **Local Traffic >> Profiles >> Services**, select **HTTP** and create a new profile
 - (b) Under **General Properties**
 - i. **Name**: secure-my-website
 - (c) Under **Settings**:
 - i. Set the **Fallback Host**: <http://10.1.1.252> (*this will take you an internal site*)
 - ii. **Fallback on Error Codes**: 404 (fallback site if a 404 error is received)
 - iii. **Response Headers Allowed**: Content-Type Set-Cookie Location
 - iv. **Insert XForwarded For**: Enabled (because we talked about it earlier)

Settings		Custom <input type="checkbox"/>
Basic Auth Realm	<input type="text"/>	<input type="checkbox"/>
Fallback Host	<input type="text"/>	<input type="checkbox"/>
Fallback on Error Codes	<input type="text"/>	<input type="checkbox"/>
Request Header Erase	<input type="text"/>	<input type="checkbox"/>
Request Header Insert	<input type="text"/>	<input type="checkbox"/>
Response Headers Allowed	<input type="text"/>	<input type="checkbox"/>
Request Chunking	Preserve ▼	<input type="checkbox"/>
Response Chunking	Selective ▼	<input type="checkbox"/>
OneConnect Transformations	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/>
Redirect Rewrite	None ▼	<input type="checkbox"/>
Encrypt Cookies	<input type="text"/>	<input type="checkbox"/>
Cookie Encryption Passphrase	<input type="text"/>	<input type="checkbox"/>
Confirm Cookie Encryption Passphrase	<input type="text"/>	
Insert X-Forwarded-For	Disabled ▼	<input type="checkbox"/>
LWS Maximum Columns	80	<input type="checkbox"/>

- (d) Attach your new HTTP Profile to your secure (HTTPS) virtual server
2. Browse to your secure virtual server.
 - (a) Do web pages appear normal?
 - (b) Now browse to a bad page
 - i. For example,
 - A. What is the result?
 - (c) Go to the **Request and Response Headers** page. You should see a sanitized server response at the bottom of the web page and the original client IP address
 - (d) You can compare the headers by accessing your HTTP virtual server at <http://10.1.10.100>
 - (e) While you are looking at the headers, check for the **X-Forwarded-For** header received by the server

Note: Even though the data is encrypted between your browser and the virtual server, the LTM can still modify the data (i.e. resource cloaking) because the data is unencrypted and decompressed within TMOS

Archive your work in a file called: **lab5_security**

3.2.6 Lab 6: BIG-IP Policies and iRules

When clients attempt to access your **secure_vs**, you don't want them to have to remember to type HTTPS before the web site, but you also don't want to open port 80 (HTTP) on your web servers as that is just asking for trouble. To avoid this issue, you will be creating an HTTP virtual server that will redirect HTTP to HTTPS and the **secure_vs**. Also, you will write an iRule and a BIG-IP policy that will retrieve images from a different pool of servers than the default pool attached to the virtual server. This will give you a simple comparison between the two methods. You will use a policy on the HTTP server and an iRule on the HTTPS virtual server.

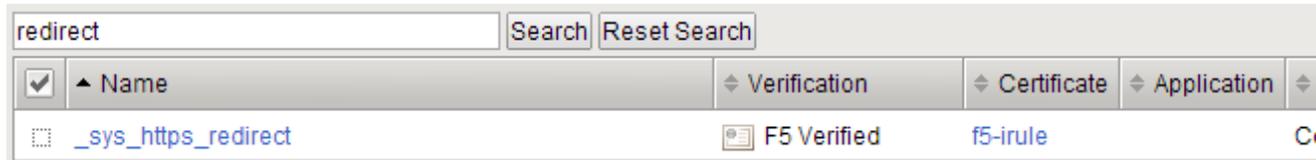
Using the Built-in `https_redirect` iRule

1. While it would be easy to write your own redirect iRule, note that F5 has one prebuilt that you can use
 - (a) Example of simple redirect iRule:

```
when HTTP_REQUEST {
    HTTP::redirect https://[HTTP::host][HTTP::uri]
}
```

2. Go to **Local Traffic >> iRules**

- (a) In the search box at the top of the list of iRules, type **redirect** and hit **Search**.



- (b) Open the iRule and take a quick look. This is an F5 Verified and supported iRule.
3. Create your HTTP-to-HTTPS redirect virtual server.
 - (a) Go to **Local Traffic >> Virtual Servers** and create a new virtual server.
 - i. **Name:** `redirect_to_secure_vs`
 - ii. **Destination:** `<same IP as secure_vs>`
 - iii. **Service Port:** 80 (HTTP)
 - iv. **Source Address Translation:** None `<you don't need this as this traffic is going nowhere>`
 - v. **iRule:** `_sys_https_redirect`
 - vi. Hit **Finished**
 - A. **WOW!** That didn't go too far did it. You just got an error. If you are going to redirect the HTTP request, you need the HOST and URI information and that requires the HTTP protocol
 - (b) In the **Configuration** section make sure the default **http** profile is added to the virtual server
 - (c) HTTP Profile: **http**
 - (d) Select **Finished**
 4. Test your policy by going to **http://<ip address of your virtual>**
 - (a) You should be redirected to the HTTPS virtual server
 - (b) As you can see, very small iRules can make a very big difference

Use a BIG-IP Policy to retrieve images from a different pool

1. **Create** a new pool named **image_pool**, use the **http** monitor for status, and add one member **10.1.20.14:80**
2. First you will create your policy container and set your match strategy
 - (a) Try to do this using the instructions, but a screen shot of the policy is available in the **Appendix** at the end of the lab guide if you would like it.
3. Go to **Local Traffic >> Policies >> Policy List** and select **Create**
 - (a) **Policy_Name:** access_image_pool
 - (b) **Strategy:** Execute **first** matching rule
 - (c) **Create Policy**

Local Traffic » Policies : Policy List » New...

Policies List Strategy List Statistics

General Properties

Policy Name	access_image_pool
Description	
Strategy	Execute first matching rule

Cancel Create Policy

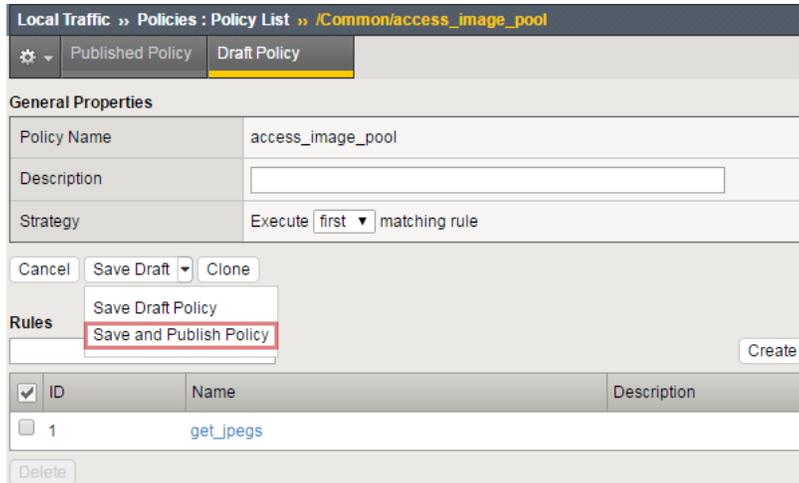
4. Now you can create/view policy rules by selecting **Create**
 - (a) **Name:** get_jpegs
 - (b) In the box under **Match all of the following conditions:** select the **+** to the right of **All Traffic**
 - i. Use the drop-down menus to look at the **HTTP URI** and check if it **ends_with** an image type
 - ii. Look for JPEGs by adding **jpg** in the box under the **any of** box and selecting **Add**

Match all of the following conditions:

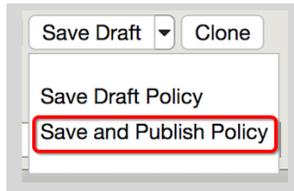
HTTP URI	extension	ends with	any of	jpg	at	request	time.
----------	-----------	-----------	--------	-----	----	---------	-------

Add

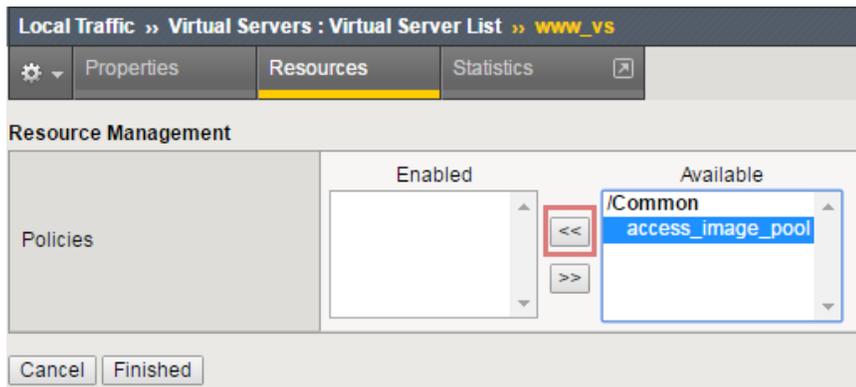
- (c) Under **Do the following when the traffic is** matched, build the following operation.
 - i. **Forward Traffic** to the **pool** named **image_pool**
- (d) **Save**



5. The policy is saved in **Draft** form and is not available until **Published**. To publish the policy:
 - (a) Select the **Save Draft Policy** drop-down menu and select **Save and Publish Policy**



6. Go to the **Resources** section of your **www_vs** virtual server and select **Managed** over the **Policies** box
 - (a) Move **access_image_pool** for the **Available** box to the **Enabled** box



7. Now test your change by browsing to <http://10.1.10.100>
 - (a) If your policy is working correctly, all of the images under **F5 Platform List** should be from **NODE #4**
 - (b) Other images are PNG images and have a different extension

Welcome to F5 Networks and the F5E vLab Test Web Site. This Web site is designed to be used with F5 vLab (virtual environment) hands-on exercises and customer demonstrations.

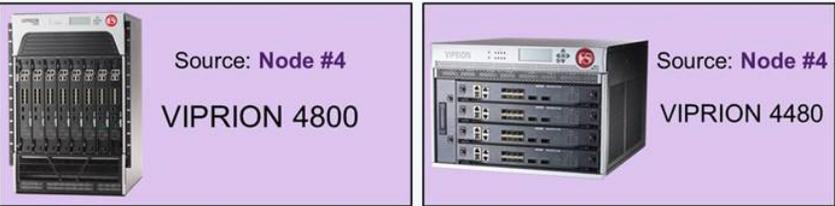


F5 Worldwide Field Readiness Node #3

Request Details

The *index.php* page is from Node #2
Virtual server address: 10.128.10.100
Pool member address/port: 10.128.20.12:80
Client IP address/port: 10.128.20.246:64687
Requested URI: /

F5 Platform List

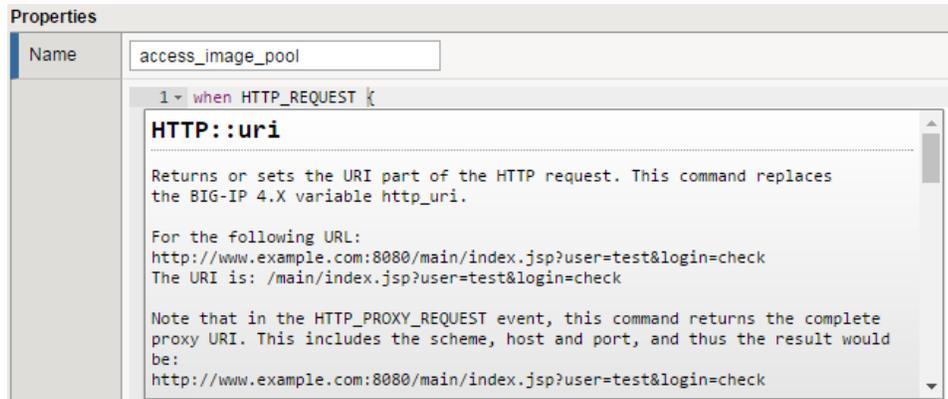


Use an iRule to Retrieve Images From a Different Pool

1. Now you will use an iRule to perform the same image retrieval. Your **image_pool** is already created
2. Go to **Local Traffic >> iRules** and select **Create**
 - (a) **Name:** access_image_pool
 - (b) In the **Definition** section enter the following:



- (c) This activity is not meant to be “cut and paste”. We want you to get comfortable and familiar with typing iRules inside the GUI.
 - i. Try hovering the cursor over a command or event, such as **HTTP_REQUEST** or **HTTP:uri**. You will see a definition of the item. For example:



3. Save your iRule and go to the **Resources** section of your **secure_vs** and select **iRules >> Manage**
 - (a) Move your **access_image_pool** iRule into the **Enabled** box
4. Test your **secure_vs** virtual by going to **https://10.1.10.105**
 - (a) The results should be the same as before
5. **Extra Credit!** Change both the policy and iRule to access the **image_pool** for **png** file types
 - (a) You should notice one is easier to update than the other

3.2.7 Lab 7: Support and Troubleshooting

In this lab, you will review your BIG-IP using iHealth and perform some basic troubleshooting commands

Objective:

- Collect a QKView and upload it to <http://ihealth.f5.com> and review the results
- Perform a TCPDump to watch traffic flow
- Obtain web page information via Curl

Archive the current configuration and perform a health check using a QKview

1. Obtain a **QKView**. Go to **System >> Support**
 - (a) Click **New Support Snapshot**

Note: If you have an iHealth account, you can choose to use the Generate and Upload QKview to iHealth option as long as your management port has Internet access. For this exercise, we will skip this.

- (b) From the drop-down, select **Generate QKview**
- (c) Select **Download** and download the QKView to your PC

Note: If you leave the **Support** page and return you will still the **Support Snapshot** screen. You **CANNOT** create another **QKView** or **TCP Dump** until the QKView is deleted.

2. Import the QKView into iHealth.

- (a) Go to <http://ihealth.f5.com>. **If you don't have an account, now is the time to create one and then skip to the next section "Troubleshoot using TCPDump or Curl" because it will take time for your account set up.**
- (b) Select the **Upload** button and upload the QKView file you download from your BIG-IP
- (c) Once the file is uploaded, you can click on the hostname to view you the heuristics
- (d) Note the Diagnostics. Go to **Diagnostics >> Critical** on the side-bar.
- (a) Example: **The configuration contains user accounts with insecure passwords** is because we are using default passwords
- (b) Select **Network >> Virtual Servers**, then click on the small white triangles to expand the view or go to **Pools**, then **Pool Members** to continue to expand the view.
- (c) This is a little more detailed than **Local Traffic >> Network Map**
- (a) If you want to see some interesting CLI commands, go to **Commands >> Standard** and expand **tmsh** then **LTM** and click on **show /ltm virtual** toward the bottom
- (b) Under **Files >> config** you can view the **bigip.conf** file and see the command lines you used for you build
 - i. All of the **log** files are here too
- (c) Feel free to just poke around

Troubleshoot using TCPDump or Curl.

1. Go to your **www_vs (10.1.10.100)** virtual server and set **Source Address Translation** to **None**.
 - (a) Now browse the web site. You will not be able to access it even though the status of the virtual is available.
 - i. Because the BIG-IP is not the server's default gateway of the servers their response goes around the BIG-IP.
 - (b) The web administrator tells you everything is fine as far as he can see and thinks the issue is with the BIG-IP, because they ALWAYS think the issue is with the BIG-IP
 - (c) You begin by debugging the client connections to the web servers through the BIG-IP using TCPDump
2. SSH to the management port of your BIG-IP.

Attention: Remember the BIG-IP is a full proxy. You will need two dumps and therefore two SSH windows for the client-side connection and the server-side connection.

- (a) First let's see if we are hitting the virtual server. At the Linux CLI prompt:
- (b) **tcpdump -i <client vlan name> host -X -s128 10.1.10.100 and port 80**
 - i. This is a little overkill, but a good example of syntax. We will only look at traffic headed for the virtual server. We will see the first 128 bytes (-s128) in ASCII (-X).
- (c) Go to your browser and attempt to access the virtual server. You should see something like this:


```
17:38:40.051122 IP 10.1.10.1.43932 > 10.1.10.245.http: S 522853636:522853636(0)
win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
0x0000: 0ffe 0800 4500 0034 0a40 4000 4006 0699 ...E..4.@@.@...
```

```

0x0010: 0a80 0a01 0a80 0aeb ab9c 0050 1f2a 1d04 .....P*..
0x0020: 0000 0000 8002 2000 3d10 0000 0204 05b4 .....=.....
0x0030: 0103 0302 0101 0402 .....
17:38:40.051169 IP 10.1.10.245.http > 10.1.10.1.43932: S 245310500:245310500(0)
ack 522853637 win 4380 <mss 1460,sackOK,eol>
0x0000: 0ffe 0800 4500 0030 27ef 4000 ff06 29ed ....E..0'.@...).
0x0010: 0a80 0aeb 0a80 0a01 0050 ab9c 0e9f 2424 ..... P...$$
0x0020: 1f2a 1d05 7012 111c 2a0e 0000 0204 05b4 *.p.*.....
0x0030: 0402 0000 ....
17:38:40.053644 IP 10.1.10.1.43932 > 10.1.10.244.http: . ack 1 win 64240
0x0000: 0ffe 0800 4500 0028 0a41 4000 4006 06a4 ....E..(A@.@...
0x0010: 0a80 0a01 0a80 0aeb ab9c 0050 1f2a 1d05 .....P*..
0x0020: 0e9f 2425 5010 faf0 7018 0000 ..$%P...p...
17:38:40.053648 IP 10.1.10.1.43932 > 10.1.10.245.http: P 1:416(415) ack 1 win 64240
0x0000: 0ffe 0800 4500 01c7 0a42 4000 4006 0504 ....E....B@.@...
0x0010: 0a80 0a01 0a80 0aeb ab9c 0050 1f2a 1d05 .....P*..
0x0020: 0e9f 2425 5018 faf0 43c5 0000 4745 5420 ..$%P...C...GET.
0x0030: 2f20 4854 5450 2f31 2e31 0d0a 486f 7374 /.HTTP/1.1..Host
0x0040: 3a20 3130 2e31 3238 2e31 302e 3233 350d :.10.1.10.245.

```

(d) Well you are hitting the virtual server so let's look a little deeper and expand our dump. Your original client IP is in the first line of the dump 16:44:58.801250 IP **10.1.10.1.41536** > 10.128.10.245.https

3. In the second SSH window we will do an expanded **tcpdump** for the sake of interest

(a) **tcpdump -i <server vlan name> -X -s128 host <client IP>**

(b) Hit your virtual server again. As you can see, we are sending packers to the pool members. They just aren't responding so we can reasonably suspect it's a server issue.

4. It could be a port issue. Let's check to see if the server is responding on port 80. On the BIG-IP, in an SSH window:

(a) Do a **<ctrl-c>** to escape out of **tcpdump**, if you are still in it, and use **curl** to test the server

(b) **curl -i <server ip>**

(c) "-i" to dump the HTTP header information also

```

[root@bigip249:Active:Standalone] config # curl -i 10.1.20.11
HTTP/1.1 200 OK
Date: Sat, 26 Jul 2014 19:25:28 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.4.9-4ubuntu2.2
Vary: Accept-Encoding
Content-Length: 3819

```

```

Connection: close
Content-Type: text/html
<html>
<head>
<TITLE>Using virtual server 10.1.20.11 and pool member 10.1.20.11 (Node #1)</TI-
TLE>
<meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />

```

(d) The server is responding to the BIG-IP when directly connected, but not through the virtual server. Sounds like the server is routing around the BIG-IP, which means the BIG-IP is **not** the default gateway.

5. Turn **SNAT Automap** back on the **www_vs** virtual server

3.2.8 Lab 8: Device Service Clusters (DSC)

We want to familiarize you with the concept of Device and Traffic Groups as well as the building of Active-Standby, Active-Active BIG-IP pairs. While there is a wizard, for this lab, configuration will be done manually. The wizard will only build A/S HA groups. In order to build Active-Active and beyond a pair you will need to know the four steps to add a device object to a cluster.

Base Networking and HA VLAN

You will be creating a high availability cluster using the second BIG-IP (**bigip2**) in your lab, so let's prep our current BIG-IP and create a high availability VLAN.

1. On **BigIpA.f5agility.com** archive your configuration in case you need to revert
2. Go to **System >> Archives** and create a new archive.
3. You will be using your third interface (**1.3**) for Network Failover and ConfigSync. This requires certain ports to be open on the Self IP; TCP port 4353 for ConfigSync, TCP port 1026 for Network Failover and TCP port 6699 for the Master Control Program.
 - (a) Build a new untagged VLAN **ha_vlan** on interface **1.3**
 - (b) Add a self-IP address to the VLAN, **192.168.20.1** net mask **255.255.255.0**.
 - (c) Under **Port Lockdown**, select **Allow Default**, to open ports required for HA communications.
4. Go to <https://10.1.1.246> which is **BigIpB.f5agility.com** and login.
5. Bigip102 has already been licensed and provision. You will need to set up the base networking.

Interface	Untagged VLAN	Self IP	Netmask
1.1	client_vlan	10.1.10.246	255.255.255.0
1.2	server_vlan	10.1.20.246	255.255.255.0
1.3	ha_vlan	192.168.20.2	255.255.255.0

6. Set **Port Lockdown** to **Allow Default**
7. Build the default gateway destination **0.0.0.0**, mask **0.0.0.0**, gateway ip address **10.1.10.1**
8. What is the status your BIG-IPs? Check the upper left-hand corner next to the F5 ball.

Configure HA

1. **On each BIG-IP**, prior to building the Device Trust, it is recommended to renew the BIG-IP self-signed certificate with valid information and regenerate the local Device Trust certificate
2. Under **System >> Certificate Management >> Device Certificate Management >> Device Certificate**, select the **Renew...** button
 - (a) **Common Name:** <the Hostname of the BIG-IP in the upper left corner>
 - (b) **Country:** United States (or your country of preference)
 - (c) **Lifetime:** 3650
3. Lifetime is important. If your cert expires your HA setup will fail
4. Select **Finished**. Your browser will ask to exchange certs with the BIG-IP again
5. Under **Device Management >> Device Trust >> Local Domain** select **Reset Device Trust...**
6. In the **Certificate Signing Authority** select **Generate New Self-Signed Authority** and hit **Update**
7. **On each BIG-IP** configure the device object failover parameters the BIG-IP will send to other BIG-IPs that want to be a part of a sync-only or sync-failover group
 - (a) Under **Device Management >> Devices**, select the local BIG-IP. It will have the **(Self)** suffix.
 - (b) Under **Device Connectivity** on the top bar select:
 - (c) **ConfigSync**
 - (d) Use the Self IP address of the HA VLAN for your **Local Address**.
 - (e) **Failover Network**
8. In the **Failover Unicast Configuration** section select the **Add** button
9. Use the Self IP address the HA VLAN for your **Address**
10. Leave the **Port** at the default setting of 1026
11. **Note:** Multicast is for Viprion chassis' only
12. **Mirroring**
13. **Primary Local Mirror Address:** use the Self IP address of the HA VLAN for your
14. **Secondary Local Mirror Address:** None
15. On **bigip01.f5agility.com** build the Device Trust.
 - (a) Under **Device Management >> Device Trust >> Device Trust Members** and select **Add** to add other BIG-IP(s) you will trust.
 - (b) **Device IP Address:** <management IP address of the BIG-IP to add>
16. You could use any Self IP if the out-of-band management interface is not configured.
 - (a) Enter the Administrator Username and Password of the BIG-IP you are trusting
 - (b) Select **Retrieve Device Information**
17. The certificate information and name from the other BIG-IP should appear
 - (a) Select **Device Certificate Matches** to proceed
 - (b) Select **Add Device**

18. On each BIG-IP check the other BIG-IP in the **Device Trust Members** list. **Is all the information there?**

Peer Authority Devices				Add...
<input checked="" type="checkbox"/>	Name	Hostname	Serial Number	MAC Address
<input type="checkbox"/>	bigip248.f5demo.com	bigip248.f5demo.com	564dc0a6-7b3a-44b9-fb437ae340a6	0:c:29:e3:40:a6

19. If some information is missing delete the trust and try again

Peer Authority Devices				Add...
<input checked="" type="checkbox"/>	Name	Hostname	Serial Number	MAC Address
<input type="checkbox"/>	bigip249.f5demo.com			

20. What are the statuses of your BIG-IPs now?
21. They should be **In Sync**. But wait! Although they show in sync, only a **Sync-Only** group was created. We now need to create a **Sync-Failover** group to facilitate failover.
22. On BigIpA.f5agility.com create a new **Sync-Failover** device group
- (a) **Under Device Management >> Device Group** create a new device group
 - (b) **Name:** my-device-group
 - (c) **Group Type:** Sync-Failover
 - (d) Add the members of the group to the **Includes** box
 - (e) Check the **Network Failover** setting for the group
23. Check **Device Groups** on each BIG-IP
24. Did you have to create the Device Group on the other BIG-IP?
25. Is the full configuration synchronized yet? (No! Only the Device Group is sync'd)
26. What is your sync status?
27. It should be **Awaiting Initial Sync**
- (a) Click on the sync status or go to **Device Management >> Overview** (or click on **Awaiting Initial Sync**) of the BIG-IP with the **good/current** configuration
 - (b) Click the device with the configuration you want to synchronize. **Sync Options** should appear.
 - (c) **Synchronize to Group**. It could take up to 30 seconds for synchronization to complete.

Note: During the **Awaiting Initial Sync** phase either BIG-IP can perform the synchronization and the other BIG-IP will be overwritten.

1. What are the statuses of your BIG-IPs? Do you have an active-standby pair?
2. Are the configurations the same?
3. Now that you have created your HA environment, HA selections will show up for SNAT addresses (not tied to your base network), persistence profiles and connection mirroring on virtual servers.
 - (a) Go to your **Active** BIG-IP
 - (b) Go to your persistence profile **my-src-persistence** and check the **Mirror Persistence** box
 - (c) Go to your **www_vs** virtual server and set the **Default Persistence Profile** to **my-src-persistence**

- (d) Synchronize your changes. Did the changes sync?
 - (e) On each BIG-IP go to **Module Statistics > Local Traffic** and bring up the persistence record statistics
 - (f) Go to the home page of your www_vs web service (<http://10.1.10.100>). Refresh a few times.
 - (g) Check the persistence records on each of your BIG-IPs, you should see the records are mirrored on each device
4. Go to **Device Management >> Traffic Groups**. As you can see the default traffic group “**traffic-group-1**” already exists.
- (a) Select **traffic-group-1**. Check out the page information and then select **Force to Standby**.
 - (b) What are the statuses of your BIG-IPs? Go to your web page. What is the client IP?
 - (c) Go to your self-IP addresses. What traffic group are they in? What does it mean?
 - (d) Archive your work.

3.2.9 Bonus Lab – Traffic groups, iApps and Active-Active

If you have time, this is a bonus lab. Here you will create a new traffic group. You will use iApps to create a new HTTP application that reside in that address group and you will create a floating IP address that will be used as the default gateway that also resides in that traffic group.

Building a new traffic group and floating IP.

1. On your **Active** BIG-IP, go to **Device Management >> Traffic Groups** and hit **Create**
 - (a) Use the f5.http template, which was designed for general web services
 - i. **Name:** iapp_tg
 - ii. Take the defaults for the rest.
2. Add a floating Self-IP to the **server_vlan**. Go to **Network >> Self IP**
 - (a) **Name:** server_gateway
 - (b) **IP Address:** 10.1.20.240
 - (c) **Netmask:** 255.255.255.0
 - (d) **VLAN/Tunnel:** server_vlan
 - (e) **Traffic Group:** iapp_tg (floating)

Building an HTTP application using an iApp template.

1. Go to **iApp >> Application Services** and hit **Create**
 - (a) Use the f5.http template, which was designed for general web services
 - i. Set the **Template Selection** to **Advanced**
 - ii. **Name:** my_new_iapp
 - iii. **Traffic Group:** iapp_tg (floating)
 - A. You will have to uncheck the **Inherit traffic group from current partition / path**.

- iv. Under **Template Options**
 - A. Select the **Advanced – Configure advanced options** for the configuration mode
 - v. Under the **Network** tab
 - A. **How have you configured routing on your web servers?** Servers have a route to the clients through the BIG-IP system
 - B. In other words, the BIG-IP is the default gateway for the servers
 - C. Otherwise the template would use SNAT by default
 - vi. Under **Virtual Server and Pools**
 - A. Your virtual server IP is **10.1.10.110**
 - B. Your hostname will be *www.f5agility.com* <*http://www.f5agility.com*> because you have to put one in.
 - C. Create a new pool with the members **10.1.20.14:80** and **10.1.20.15:80**
 - D. **If you hit add after the last pool member and have a new row, you will need to delete the row prior to finishing**
 - vii. Hit **Finished** at the bottom of the page
2. Go to **iApp >> Application Services** and select the new application you created
 - (a) Select **Components** from the top bar
 - i. Here you will see all the configuration items created by the iApp
 - ii. Do you see anything created that you weren't asked about?
 3. Remember the concept of strictness? Let's test that out
 - (a) Go to **Local Traffic >> Pools >> Pool List**
 - i. Select the pool created by your iApp: **my_new_iapp_pool**
 - ii. Attempt to add **10.15.11.13:80** to your **my_new_iapp_pool**
 - A. Did it fail?
 - (b) Go to your iApp and select **Reconfigure** from the top bar
 - i. Now attempt to add your new pool member
 - ii. You can check the Components tab to verify your success

SYNCHRONIZE YOUR CHANGES

Active-Active Setup

1. Now, let's make our sync-failover group active-active. On the **Active** BIG-IP:
 - (a) Go to **Device Management >> Traffic Groups**
 - i. Go to you **iapp_tg** traffic group.
 - ii. Under **Advanced Setup Options**
 - A. You are going to set up **iapp_tg** to prefer to run on **bigip02.f5agility.com** and auto failback to **bigip02** if **bigip02** should go down and come back up later.
 - B. Is this normally a good idea?

- iii. **Failover Method:** HA Order
 - iv. **Auto Failback:** <checked>
 - v. **Failover Order:** **bigip102.f5agility.com** then **bigip01.f5agility.com**
 - vi. Ensure you synchronized the change to the other BIG-IP
2. If the traffic group is active on the wrong BIG-IP initially you will have to do a Force to Standby on the traffic group to make it active on the BIG-IP you want it on by default
- (a) What is the ONLINE status of each of your BIG-IPs?
 - (b) Reboot the BIG-IP with your second traffic group on it. Watch to see if the application becomes active on the other BIG-IP during the reboot and if it falls back to the Default Device once the BIG-IP has come back up.
 - (c) You can verify this by checking your traffic groups or going to the web server and looking at the client IP

Class 4: Troubleshoot with tcpdump and Wireshark

Welcome to the troubleshoot with tcpdump and Wireshark documentation.

4.1 F5 tcpdump and Wireshark

This class covers the following topics:

- tcpdump switches and filters
- F5 specific tcpdump commands
- F5 Wireshark Plugin
- Using the F5 Wireshark Plugin
- SSL decrypt packet capture

We will be using a jumpbox to connect to the lab environment. Click on the lab link given out during class and select the RDP option to connect to the lab box.

The credentials will be the following:

user: f5student

password: f5DEMOs4u

4.1.1 tcpdump Switches

The tcpdump command has several switches with different purposes. The following are some of the most commonly used.

You can run these commands from the Jumpbox to see the output in our lab environment or you can just read through the information, it is up to you. To launch the SSH connection to the BIG-IP double click on the Putty shortcut on the desktop. Then connect to the BIGIP01 instance. The credentials are:

user: root password: default

1. **tcpdump -D**

To list the available interfaces for packet capture use tcpdump -D

The 'any' interface will be taken by TMM and made into the interface '0.0'

```
[root@bigipA:Active:Standalone] config # tcpdump -D
1.eth0
2.tmm
3.tmm_bp
4.internal
5.external
6.eth1
7.lorax_portal_~1
8.any (Pseudo-device that captures on all interfaces)
```

2. tcpdump -i

To capture traffic on a specific interface use tcpdump -i <interface name>. i.e. 'tcpdump -i 0.0'

When using 0.0 for the interface on a capture make sure to use a capture filter or you will get too much information and may impact performance on the F5.

```
[root@bigipA:Active:Standalone] config # tcpdump -i external
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on external, link-type EN10MB (Ethernet), capture size 65535 bytes
```

3. tcpdump -n

Use tcpdump -n to disable name resolution of host names

4. tcpdump -nn

Use tcpdump -nn to disable name resolution of both host names and port names

5. tcpdump -X

Use tcpdump -X to show output including ASCII and hex. This will making reading screen output easier.

```
[root@bigipA:Active:Standalone] config # tcpdump -X -nni external host 10.1.10.55
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on external, link-type EN10MB (Ethernet), capture size 65535 bytes

09:30:32.085972 IP 10.1.10.199.49234 > 10.1.10.55.443: Flags [S], seq 2106152363, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sack0K], length 0 i
n slot1/tmm1 lis=
  0x0000:  4500 0034 0256 4000 8006 cf6e 0a01 0ac7  E..4.W@....n....
  0x0010:  0a01 0a37 c052 01bb 7d89 55ab 0000 0000  ...7.R..)U.....
  0x0020:  8002 2000 90d4 0000 0204 05b4 0103 0302  .....
  0x0030:  0101 0402 0105 0101 0001 00  .....
09:30:32.086122 IP 10.1.10.199.49235 > 10.1.10.55.443: Flags [S], seq 1181596894, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sack0K], length 0 i
n slot1/tmm0 lis=
  0x0000:  4500 0034 0257 4000 8006 cf6d 0a01 0ac7  E..4.W@....m....
  0x0010:  0a01 0a37 c053 01bb 466d bcde 0000 0000  ...7.S..Fm.....
  0x0020:  8002 2000 60bc 0000 0204 05b4 0103 0302  .....
  0x0030:  0101 0402 0105 0101 0000 00  .....
09:30:32.086248 IP 10.1.10.55.443 > 10.1.10.199.49235: Flags [S.], seq 1497446718, ack 1181596895, win 4380, options [mss 1460,sack0K,eol], length 0
ut slot1/tmm0 lis=/Common/lorax_portal_access_vs
  0x0000:  4500 0030 ede8 4000 f706 64df 0a01 0a37  E..0..@...d....7
  0x0010:  0a01 0ac7 01bb c053 5941 393e 466d bcdf  .....SYA9>Fm..
  0x0020:  7012 111c f219 0000 0204 05b4 0402 0000  p.....
  0x0030:  0123 0100 0000 1e2f 436f 6d6d 6f6e 2f6c  .#...../Common/l
  0x0040:  6f72 6178 5f70 6f72 7461 6c5f 6163 6365  orax_portal_acce
  0x0050:  7373 5f76 73  ss vs
```

6. tcpdump -w

Use tcpdump -w to write the packet capture to a capture file that is readable in an application such as Wireshark.

7. tcpdump -s

Use 'tcpdump -s0' to capture the full data packet. The number following the 's' indicates the number of bits to capture of each packet. 0 indicates all.

4.1.2 tcpdump Filters

This section is for informative value and nothing will be done in the lab environment.

When running tcpdump capture from the F5 you should always use a filter to limit the volume of traffic you will gather.

1. Host Filters

tcpdump host 192.168.2.5 This will filter the packet capture to only gather packets going to or coming from the host 192.168.2.5.

tcpdump src host 192.168.2.5 This will filter the packet capture to only gather packets coming from 192.168.2.5.

tcpdump dst host 192.168.2.5 This will filter the packet capture to only gather packets going to 192.168.2.5.

2. Port Filters

tcpdump port 443 This will filter the packet capture to only gather packets with a source or destination of port 443.

tcpdump src port 1055 This will capture traffic being sourced from port 1055.

tcpdump dst port 443 This will capture traffic destined for port 443.

4.1.3 F5 Specific tcpdump Switches

This section is for informative value and nothing will be done in the lab environment.

F5 has added some F5 specific switches to the tcpdump utility on the F5. These switches give additional information on your packet captures. These switches are places after the interface option in the command line as follows:

1. :n gives low details

(a) for example: **tcpdump -nni 0.0:n -s0 -w/var/tmp/capture.pcap**

(b) This will give basic information such as whether the captured traffic is ingress or egress to the F5. It will also give the TMM instance the traffic is on as well as the Chassis slot processing the traffic. This option also lists the virtual server name that processes the traffic.

Stream	Delta	Time	VLAN	L2 Src	Source
2766	133	0.000295	14:00:55.635257	10	52:54:00:28:6d:a5 10.1.10.199
2767	133	0.000004	14:00:55.635261	10	52:54:00:28:6d:a5 10.1.10.199
2768	133	0.000010	14:00:55.635271	10	52:54:00:65:f1:6c 10.1.10.35
2769	133	0.000013	14:00:55.635283	10	52:54:00:65:f1:6c 10.1.10.35

▼ F5 Ethernet trailer

- ▼ F5 Low Details
 - Ingress: True (IN)
 - Slot (1-based): 1
 - TMM (0-based): 1
 - VIP: /Common/dvwa_virtual_35

2. :nn gives medium details

(a) for example: **tcpdump -nni 0.0:nn -s0 -w/var/tmp/capture.pcap**

(b) This option will give all the low detail information plus the following:

- Flow ID number

- Peer Flow ID number
- TCP RST cause
- Flow type
- HA unit
- Ingress Slot
- Ingress port
- Priority

No.	Stream	Delta	Time	VLAN	L2 Src	Source
2773	136	-0.000001	14:01:16.879800	10	52:54:00:28:6d:a5	10.1.10.
2774	136	0.000161	14:01:16.879961	10	52:54:00:65:f1:6c	10.1.10.
2775	135	-0.000003	14:01:16.879958	10	52:54:00:65:f1:6c	10.1.10.
2776	137	0.167601	14:01:17.047610	10	52:54:00:28:6d:a5	10.1.10.

```

▼ F5 Ethernet trailer
  ► F5 Low Details
  ▼ F5 Medium Details
    Flow ID: 0x00002ae866bf6640
    Peer ID: 0x0000000000000000
    Connflow Flags High Bits: 0x00000000
    Connflow Flags: 0x04000020
    Flow Type: 0x46
    HA Unit: 0x01
    Ingress Slot: 0
    Ingress Port: 0
    Priority: 0
  ► RST cause: [22710cf:2046] No local listener

```

3. :nnn gives high details

- for example: **tcpdump -nni 0.0:nnn -s0 -w/var/tmp/capture.pcap**
- This option will give all the low and medium details plus the following
 - Peer IP Protocol
 - Peer VLAN
 - Peer Remote address
 - Peer local address
 - Peer remote port
 - Peer local port

No.	Stream	Delta	Time	VLAN	L2 Src	Source	Src Port	Dst port	Destination	L2 Dst	Protocol
72	2	0.000007	16:05:26.249479	10	52:54:00:65:f1:6c	10.1.10.35	443	49177	10.1.10.199	52:54:00:28:6d:a5	TCP
73	2	0.000016	16:05:26.249495	10	52:54:00:65:f1:6c	10.1.10.35	443	49177	10.1.10.199	52:54:00:28:6d:a5	TCP
74	3	0.000007	16:05:26.249502	20	52:54:00:65:f1:6c	10.1.20.240	38416	80	10.1.20.17	52:54:00:e8:f7:b4	TCP

▶ Frame 74: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits)
 ▶ Ethernet II, Src: RealtekU_65:f1:6c (52:54:00:65:f1:6c), Dst: RealtekU_e8:f7:b4 (52:54:00:e8:f7:b4)
 ▶ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 20
 ▶ Internet Protocol Version 4, Src: 10.1.20.240, Dst: 10.1.20.17
 ▶ Transmission Control Protocol, Src Port: 38416, Dst Port: 80, Seq: 518, Ack: 989, Len: 0
 ▼ F5 Ethernet trailer
 ▼ F5 Low Details
 Ingress: False (OUT)
 Slot (1-based): 1
 TMM (0-based): 0
 VIP: /Common/dvwa_virtual_35
 ▶ F5 Medium Details
 ▼ F5 High Details
 Peer IP Protocol: 6
 Peer VLAN: 10
 Peer remote address: 10.1.10.199
 Peer remote address: ::ffff:10.1.10.199
 Peer local address: 10.1.10.35
 Peer local address: ::ffff:10.1.10.35
 Peer remote port: 49177
 Peer local port: 443

4.1.4 Install the F5 Wireshark Plugin

Wireshark version 2.2.14 is installed on the jumpbox.

You can download the F5 Wireshark plugin from devcentral.f5.com here: <https://devcentral.f5.com/d/wireshark-plugin?lc=1>. In the lab the plugin is already downloaded to /home/f5student/Downloads/wireshark/.

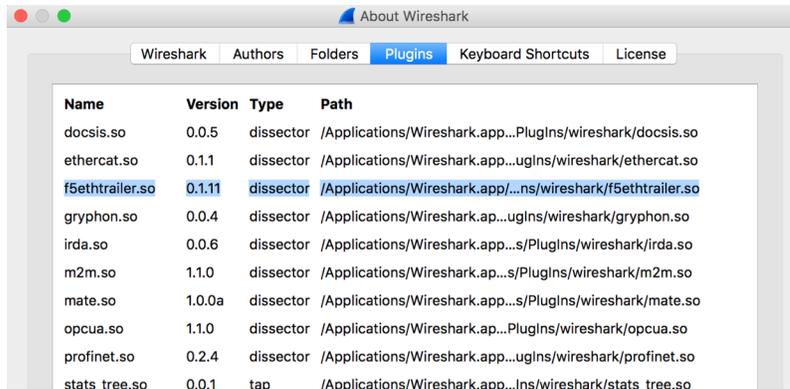
1. Start Wireshark by double clicking the shortcut on the desktop.
2. Click on Help and then About Wireshark.
3. Click on the plugins tab and check to see what directory the plugins are installed to.
4. Open the plugin directory in file explorer.
5. Copy the F5 wireshark plugin that has been copied to /home/f5student/Downloads/wireshark/
 - (a) The jumpbox client is Ubuntu 64 bit.
 - (b) Open the Linux64-2.2.0 folder and copy the f5ethtrailer.so to the plugin directory determined in step 3.
 - (c) In order to move the file you will need to elevate permissions. The easiest way to do this is from the command line terminal. Use the command:

```

1 sudo cp ./Downloads/wireshark/Linux64-2.2.0/f5ethtrailer.so /usr/lib/x86_64-
  ↪linux-gnu/wireshark/plugins/2.2.6/

```

6. Depending on your OS and Wireshark version, you will need the correct plugin files from the correct folder.
7. Shutdwon Wireshark and restart it.
8. Click on Help and then About Wireshark.
9. Check the plugins tab again and make sure the F5 plugin is installed.



4.1.5 Taking a Capture from the F5

Let's take the information we have gathered so far and take a packet capture from the F5.

1. Start Putty and launch the bigip01 SSH session.
2. Login as root user. Password is 'default'.
3. List the destination address of the virtual on the F5 using the following command:

```
tmsh list ltm virtual hackazon.f5demo.com.app/hackazon.f5demo.com_vs destination
```

4. Now take the destination address and compose a tcpdump command to track the traffic coming to this virtual server:

```
tcpdump -nni 0.0:nnn -s0 -w/var/tmp/hackazon.pcap host 10.1.10.201
```

5. After starting the capture, start Chrome and click on the Hackazon bookmark. Browse around the site following a couple links. Next go to the address bar and type in: "<https://hackazon.f5demo.com:8080>". Then stop the capture in the putty session by using 'Ctrl+c'.
6. Open a terminal screen on the Ubuntu jumpbox. Change Directory to the Downloads folder.
7. Run the following command:

```
sudo scp root@10.1.1.245:/var/tmp/hackazon.pcap ./
```

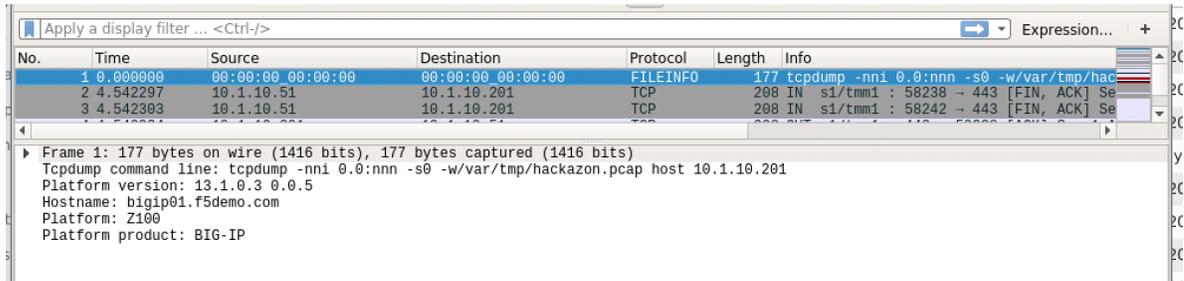
8. You will be prompted to save the SSH keys. Type yes and hit enter, then enter the root password for the F5.
9. Now open Wireshark and open the hackazon.pcap file you just copied from the F5.
10. If you run into issues copying the hackazon.pcap file to the jumpbox you can use the already created file in the Downloads folder hackazon2.pcap.

4.1.6 Configuring/Using Wireshark F5 Plugin

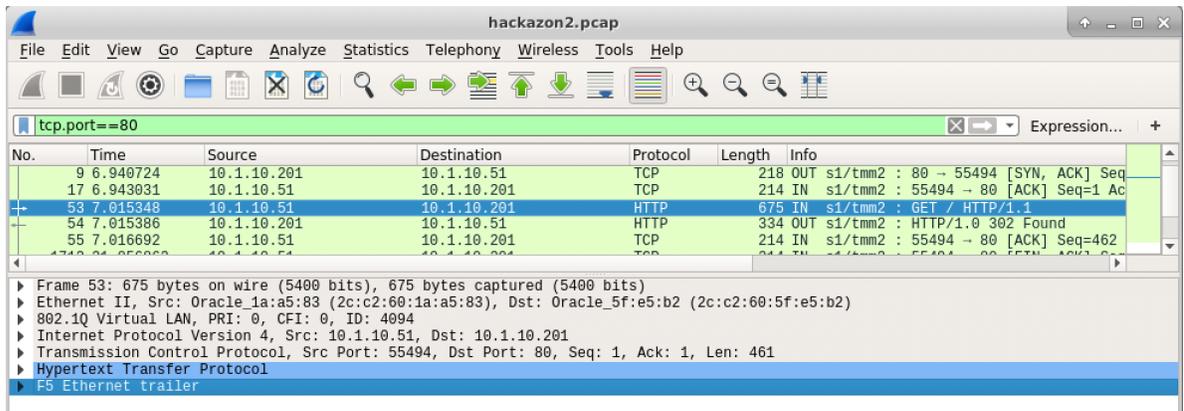
We will start with what kind of unique information is gathered through the plugin and using tcpdump on the F5.

1. Start by selecting packet 1 in Wireshark.

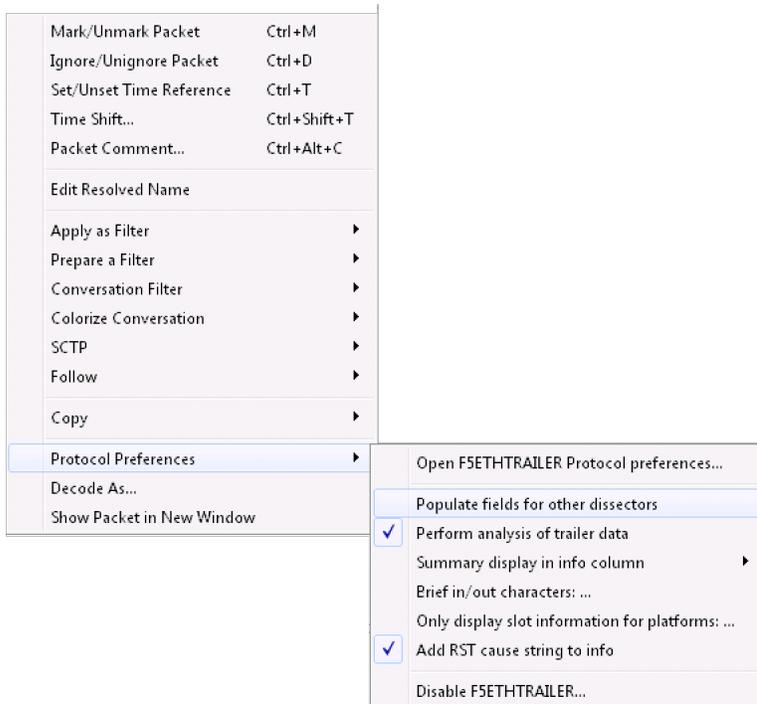
- Notice in the middle section of Wireshark you will see the tcpdump command being run. You will also see the version of the F5 code, the F5 hostname, and the Platform ID number (in this case Z100 for Virtual Edition).



- Now we will use a Wireshark display filter to see a specific request. Add 'tcp.port == 80' in the display filter field and hit enter.

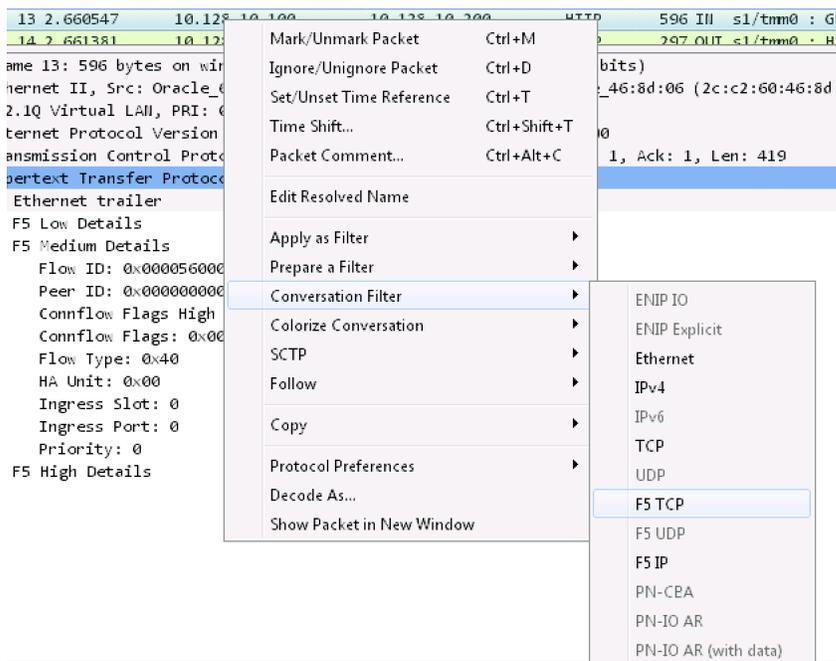


- In the capture above packet 53 shows the GET requests to the website. In your capture it will be a different packet number but you can see in the Info area that it is a GET request.
- Right click on the GET request and go to protocol preferences, and then populate fields for other dissectors. This makes it so when applying a display filter it applies to both the client and server sides of the F5 connection. We will cover this in the F5 High Details section.



4.1.7 Follow F5 Conversation

1. Clear the display filter and hit enter. Right Click on one of the packets in the capture and select Conversation Filter and then F5 TCP. This will automatically develop a display filter to show the client and server sides of a conversation going through the BIG-IP device.



4.1.8 F5 Low Details

1. Apply a new display filter of 'tcp.port == 80' to the capture.
2. Select the packet with the GET request in the info field.
3. In the middle section Expand F5 Ethernet Trailer.
4. Then expand F5 Low Details.

No.	Time	Source	Destination	Protocol	Length	Info
6	4.751554	10.128.10.100	10.128.10.200	TCP	177	50151 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	4.752109	10.128.10.100	10.128.10.200	HTTP	626	III s1/tmm3 : GET / HTTP/1.1
8	4.752349	10.128.10.200	10.128.10.100	HTTP	297	OUT s1/tmm3 : HTTP/1.0 302 Found
47	4.979067	10.128.10.100	10.128.10.200	TCP	177	50151 → 80 [ACK] Seq=450 Ack=121 Win=64120 Len=0

▶ Frame 7: 626 bytes on wire (5008 bits), 626 bytes captured (5008 bits)

▶ Ethernet II, Src: Oracle_6e:41:20 (2c:c2:60:6e:41:20), Dst: Oracle_46:8d:06 (2c:c2:60:46:8d:06)

▶ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 4094

▶ Internet Protocol Version 4, Src: 10.128.10.100, Dst: 10.128.10.200

▶ Transmission Control Protocol, Src Port: 50151, Dst Port: 80, Seq: 1, Ack: 1, Len: 449

▶ **Hypertext Transfer Protocol**

▶ F5 Ethernet trailer

- ▶ F5 Low Details
 - Ingress: True (IN)
 - Slot (1-based): 1
 - TMM (0-based): 3
 - VIP: /Common/hackazon.f5demo.com_http_vs
- ▶ F5 Medium Details
- ▶ F5 High Details

5. Notice the Ingress value is True (IN). This is from the perspective of the F5. The traffic is inbound to the F5.
6. The low details also gives the Slot value (always be 1 for an appliance). The TMM number in the image is 3.
7. The most important value here is the VIP. In this case it is the /Common/hackazon.f5demo.com_http_vs. Notice this is the port 80 VIP for this particular destination IP. The VIP is configured with a redirect to SSL.
8. The next packet in the capture is a HTTP 302 redirect to the SSL vip.

4.1.9 F5 Medium Details

1. Now expand the F5 Medium Details.

No.	Time	Source	Destination	Protocol	Length	Info
5	4.748757	10.128.10.200	10.128.10.100	TCP	185	OUT s1/tmm3 : 80 → 50151 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 SACK_PERM=1
6	4.751554	10.128.10.100	10.128.10.200	TCP	177	50151 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	4.752109	10.128.10.100	10.128.10.200	HTTP	626	III s1/tmm3 : GET / HTTP/1.1
8	4.752349	10.128.10.200	10.128.10.100	HTTP	297	OUT s1/tmm3 : HTTP/1.0 302 Found
9	4.763068	10.128.10.100	10.128.10.200	TCP	154	III s1/tmm2 : 50153 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1

▶ Frame 7: 626 bytes on wire (5008 bits), 626 bytes captured (5008 bits)

▶ Ethernet II, Src: Oracle_6e:41:20 (2c:c2:60:6e:41:20), Dst: Oracle_46:8d:06 (2c:c2:60:46:8d:06)

▶ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 4094

▶ Internet Protocol Version 4, Src: 10.128.10.100, Dst: 10.128.10.200

▶ Transmission Control Protocol, Src Port: 50151, Dst Port: 80, Seq: 1, Ack: 1, Len: 449

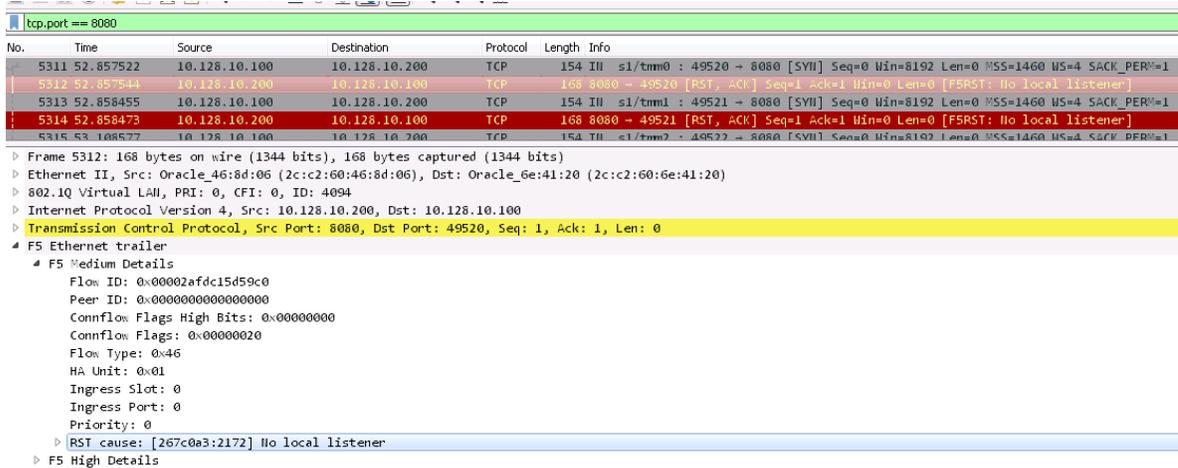
▶ **Hypertext Transfer Protocol**

▶ F5 Ethernet trailer

- ▶ F5 Low Details
- ▶ **F5 Medium Details**
 - Flow ID: 0x00005601d6849800
 - Peer ID: 0x0000000000000000
 - Connflow Flags High Bits: 0x00000000
 - Connflow Flags: 0x00000024
 - Flow Type: 0x40
 - HA Unit: 0x00
 - Ingress Slot: 0
 - Ingress Port: 0
 - Priority: 0
- ▶ F5 High Details

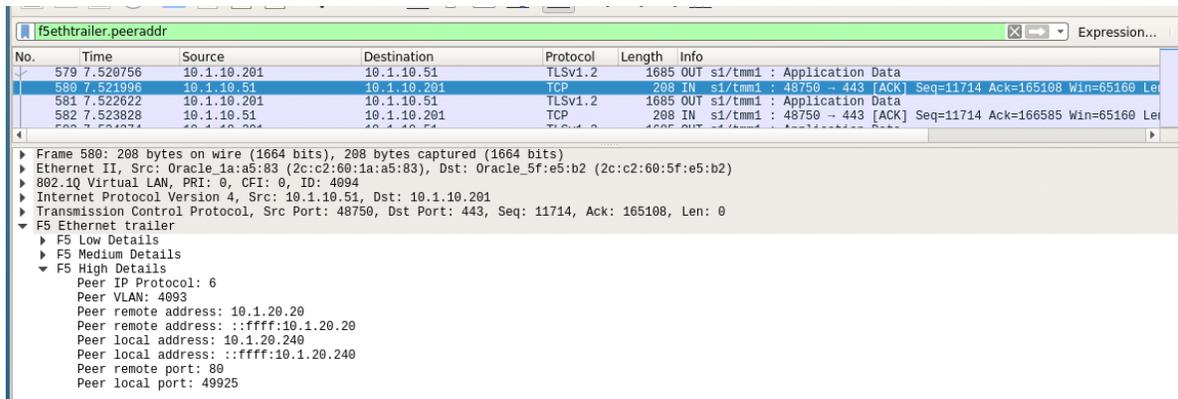
2. A majority of the information in the Medium Details will be used by F5 support in analyzing packet flow. The important parts to notice are the Flow ID, Peer ID, and RST Cause.

- The Flow ID and Peer ID. These are unique identifiers for the two flows of the connection. They are only unique within a specific slot and tmm combination. A flow ID may be reused, so it is possible to see unrelated packets with the same flow ID in a capture before or after the connection you are interested in.
- In the display filter type 'tcp.port == 8080'. Now Select one of the RST packets and expand the F5 Medium Details. Notice there is a RST Cause value. In this case the RST Cause is "No local listener". This indicates that there is no VIP on the F5 that matches the connection request. There are other RST causes. In the F5 Medium Details you will only see RST Cause for packets that were RST by the F5.



4.1.10 F5 High Details

- Now Expand the F5 High Details on one of the SSL sessions data packets. (You can type in a display filter of 'f5ethtrailer.peeraddr' to retrieve the session we are looking for). Make sure you select a packet that has Application Data in the Info field.



- The F5 High Details contains a lot of valuable information. In the example above we see that the Peer IP Protocol is 6 or TCP.
- The Peer VLAN is going to be the VLAN that the traffic enters the F5 from. In the example this is VLAN 4093 which is the external VLAN.
- The Peer remote address for an Outbound packet will be the Application server IP address, if it were Inbound the address would be the Client Address. The Peer Remote address will always be an

address that is not on the BIG-IP directly, in this case 10.1.20.20. The second Peer remote address is the IPv6 version of the address.

5. The Peer local address is the SNAT address for this particular Virtual Server, in this case 10.1.20.240. Once again the IPv6 version is below that.
6. Next we see the Peer remote port. The remote port is the port that the traffic is coming from on the remote side. In this case the application server is responding from port 80.
7. Next we have the Peer local port. This is the port the F5 is talking with the remote side on. In this case it is port 49925. If this were a request coming from the client, the local port would be 443 as the Virtual Server is configured with port 443 and the remote port would be a port > 1024.
8. The F5 High Details provide great information on what pool member is selected on a particular conversation that you are tracking down. This makes it possible to determine which server may be having issues in an application problem.

4.1.11 High Details and Other Field Dissectors

1. In order to see how the other field dissectors becomes useful, enter a display filter of 'ip.addr==10.1.20.240'.
2. Notice that the Source and Destination fields in the results do not contain the ip address of 10.128.20.210 as shown in the image. In the lab you won't see the IP 10.1.20.240 in the source or destination fields.

The screenshot shows a Wireshark interface with a display filter 'ip.addr==10.128.20.210'. The packet list shows three packets (173, 174, 175) of type TLSv1.2. The packet details pane is expanded to show 'F5 High Details' for packet 174. The details include:

No.	Time	Source	Destination	Protocol	Length	Info
173	3.820632	10.128.10.200	10.128.10.100	TLSv1.2	1667	OUT s1/tmm0 : Application Data
174	3.820672	10.128.10.200	10.128.10.100	TLSv1.2	1667	OUT s1/tmm0 : Application Data
175	3.820863	10.128.10.200	10.128.10.100	TLSv1.2	1667	OUT s1/tmm0 : Application Data

Packet details for F5 High Details:

- Peer IP Protocol: 6
- Peer VLAN: 4093
- Peer remote address: 10.128.20.210
- Peer remote address: ::ffff:10.128.20.210
- Peer local address: 10.128.20.11
- Peer local address: ::ffff:10.128.20.11
- Peer remote port: 80
- Peer local port: 58664

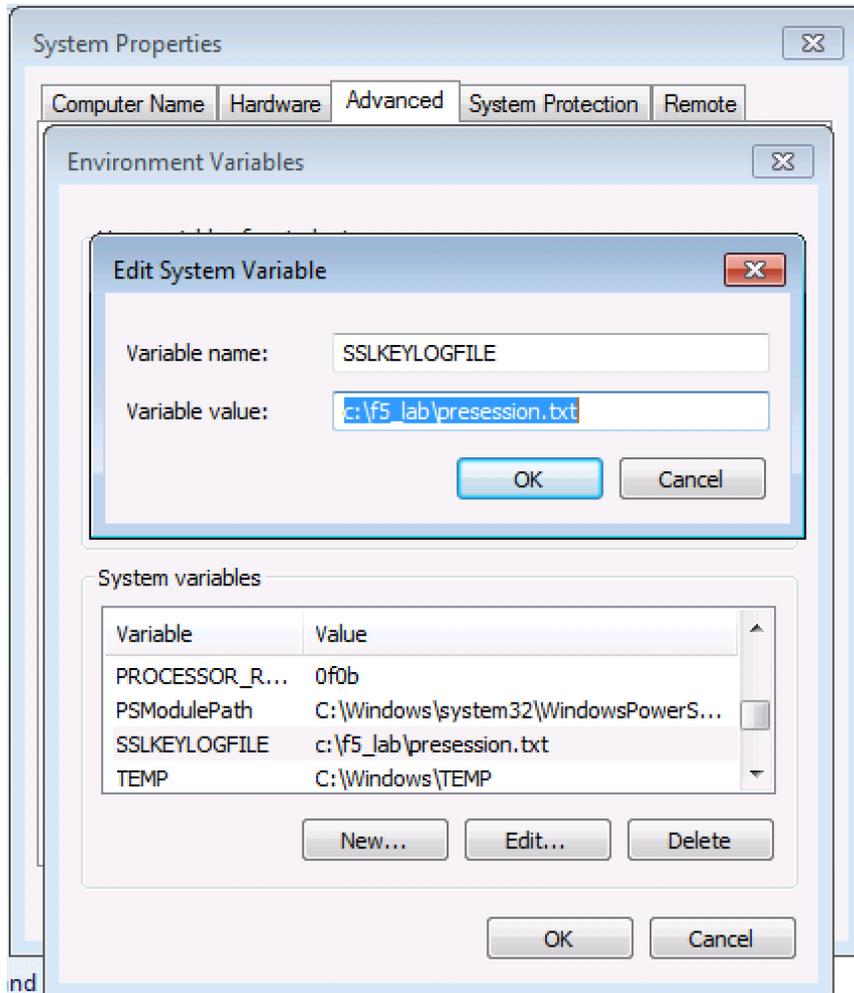
3. Is the display filter broken?
4. No it is not. Notice in the F5 High Details that the Peer remote address is 10.128.20.210. This is the pool member selected for the pool in the virtual server.
5. With the other field dissectors value turned on in Wireshark, you can set a display filter of the pool member you may be troubleshooting and see which virtual is being selected on the traffic. Or you can set a display filter of the client and see all the traffic to the pool members even with a SNAT pool selected.

4.1.12 SSL Decryption

There are two ways to decrypt the SSL traffic. Both ways require that you perform one of the following tasks before you take the TCP Capture.

SSL Decrypt from Windows Client

1. To use the client to decrypt you must add a System Variable to log the session key data for decryption. On a windows client you would go into the Environment Variables and add a SSLKEYLOGFILE value to a text file on the machine as in the following image.



2. Once the system variable has been put in place you can then launch a web browser and start the traffic that you want to analyze.
3. Once the traffic has been captured you will import the capture file into Wireshark and configure the SSL options to use the pre-master key file defined in the system variables.

SSL Decrypt from Linux Client

1. On the Ubuntu client open a terminal window and run the following commands:

```

1 touch session-key.log
2 export SSLKEYLOGFILE=/home/f5student/session-key.log
3 chromium-browser

```

2. This will launch the Chrome browser and once you close the browser it will stop logging the SSL Session key data.
3. Once the traffic has been captured you will import the capture file into Wireshark and configure the SSL options to use the session-key.log file.

SSL Decrypt from F5

More often you will not have access to modify the client in order to capture the SSL session data. The other option is to get the pre-master session data from the F5 itself by doing the following.

1. Configure a new iRule as follows:

```

1 when CLIENTSSL_HANDSHAKE {
2     log local0. "RSA Session-ID:[SSL::sessionid] Master-Key:[SSL::sessionsecret]"
3 }
4
5 when SERVERSSL_HANDSHAKE {
6     log local0. "RSA Session-ID:[SSL::sessionid] Master-Key:[SSL::sessionsecret]"
7 }

```

2. Apply this new iRule to the virtual server. In our lab environment the iRule has already been created and applied to the Virtual Server.
3. You can now start a tcpdump and surf the website.
4. After you have stopped the tcpdump, you will now need to SSH to the F5 and run the following command:

```
grep Session-ID /var/log/ltm | sed 's/.*\ (RSA.*) /\1/' > /var/tmp/session.pms
```

5. Now the session.pms file can be pulled from the F5 and put into Wireshark.

4.1.13 Decrypting SSL in Wireshark

Now you need to have your pre-master key file and your capture moved to your local box. To do this do the following:

1. Open Terminal on the Ubuntu Jump Box.
2. Change directory to Documents by typing: 'cd Documents'.
3. Run the following commands:

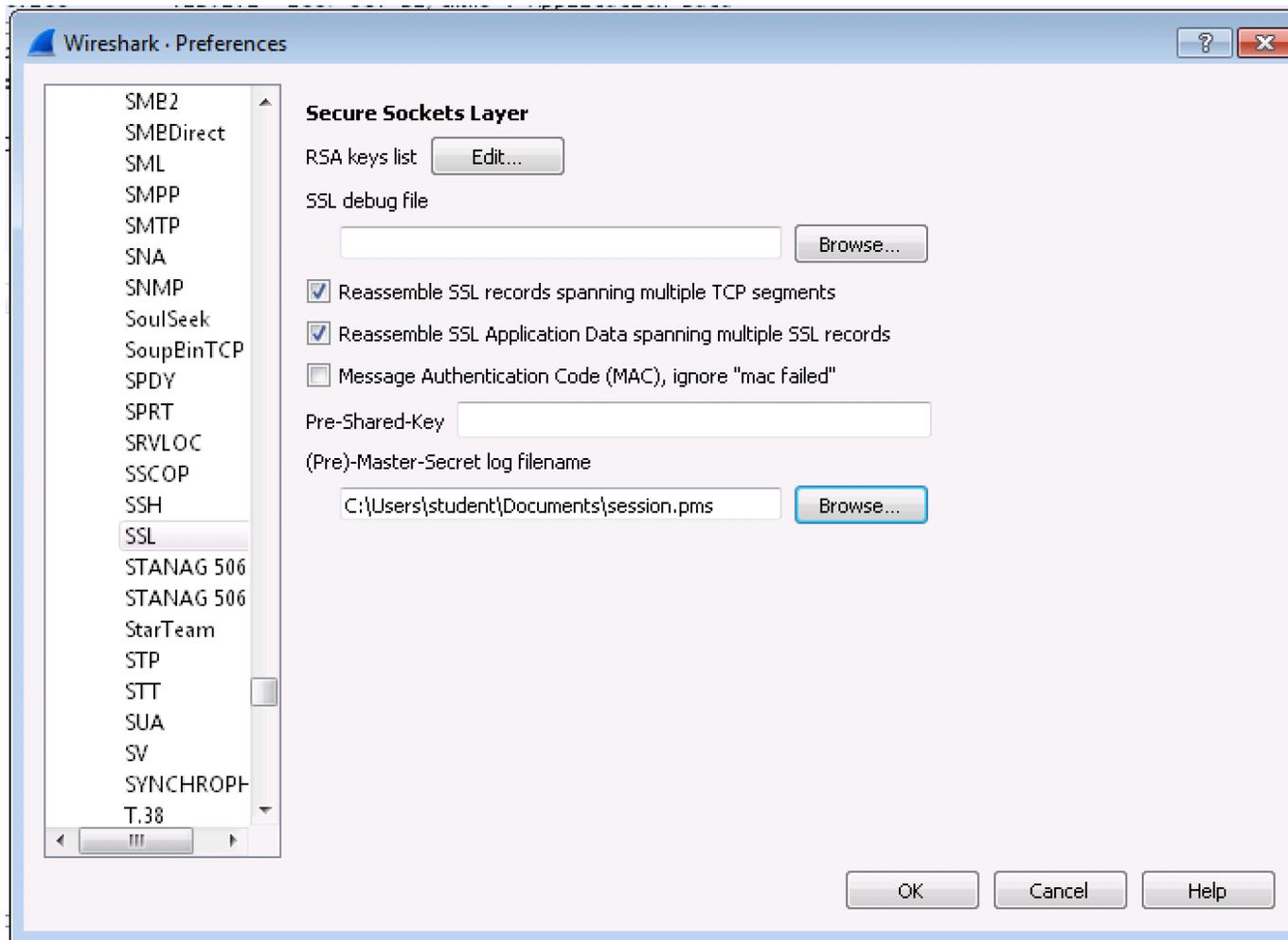
```

1 sudo scp root@10.1.1.245:/var/tmp/session.pms ./
2 sudo scp root@10.1.1.245:/var/tmp/hackazon.pcap ./
3
4 After each of these commands you will be prompted to accept the SSH keys. Type 
  ↪ yes to continue. Then you will be prompted for the F5 root password. Type that 
  ↪ in as well.

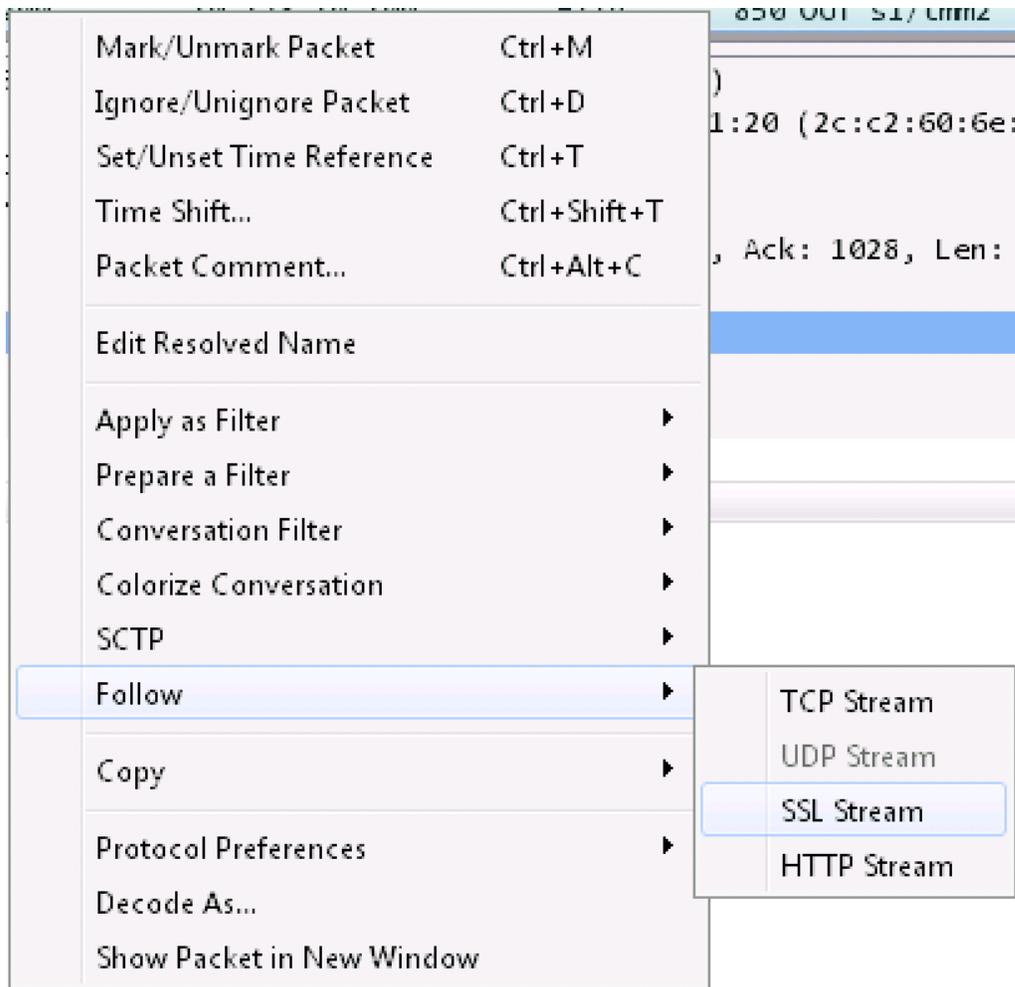
```

4. Now open Wireshark.
5. Once Wireshark is open go to Edit/Preferences.

- Expand on the left side, Protocols, then select SSL.



- Browse to the pre-master session key file and click on save.
- Open in Wireshark the pcap file you pulled down from the F5 BIG-IP.
- Right click on one of the SSL packets and select Follow, SSL Stream.



10. You will now see unencrypted SSL data in the capture as follows:

```
GET /css/bootstrapValidator.css HTTP/1.1
Host: hackazon.f5demo.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
Accept: text/css,*/*;q=0.1
Referer: https://hackazon.f5demo.com/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: visited_products=%2C81%2C1%2C16%2C; JSESSIONID=pddeqa6fsnlpha9k

HTTP/1.1 200 OK
Date: Fri, 04 May 2018 15:23:44 GMT
Server: Apache/2.4.7 (Ubuntu) PHP/5.5.9-1ubuntu4.12 OpenSSL/1.0.1f
Last-Modified: Thu, 17 Sep 2015 11:23:30 GMT
ETag: "1d8-51fefa5169ae1-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 308
Connection: close
Content-Type: text/css
```

Resilient Data Center Architectures with F5 BIG-IP

This class covers covering common High Availability, BGP configurations on the BIG-IP.

Expected time to complete: **3 hours**

5.1 Getting Started

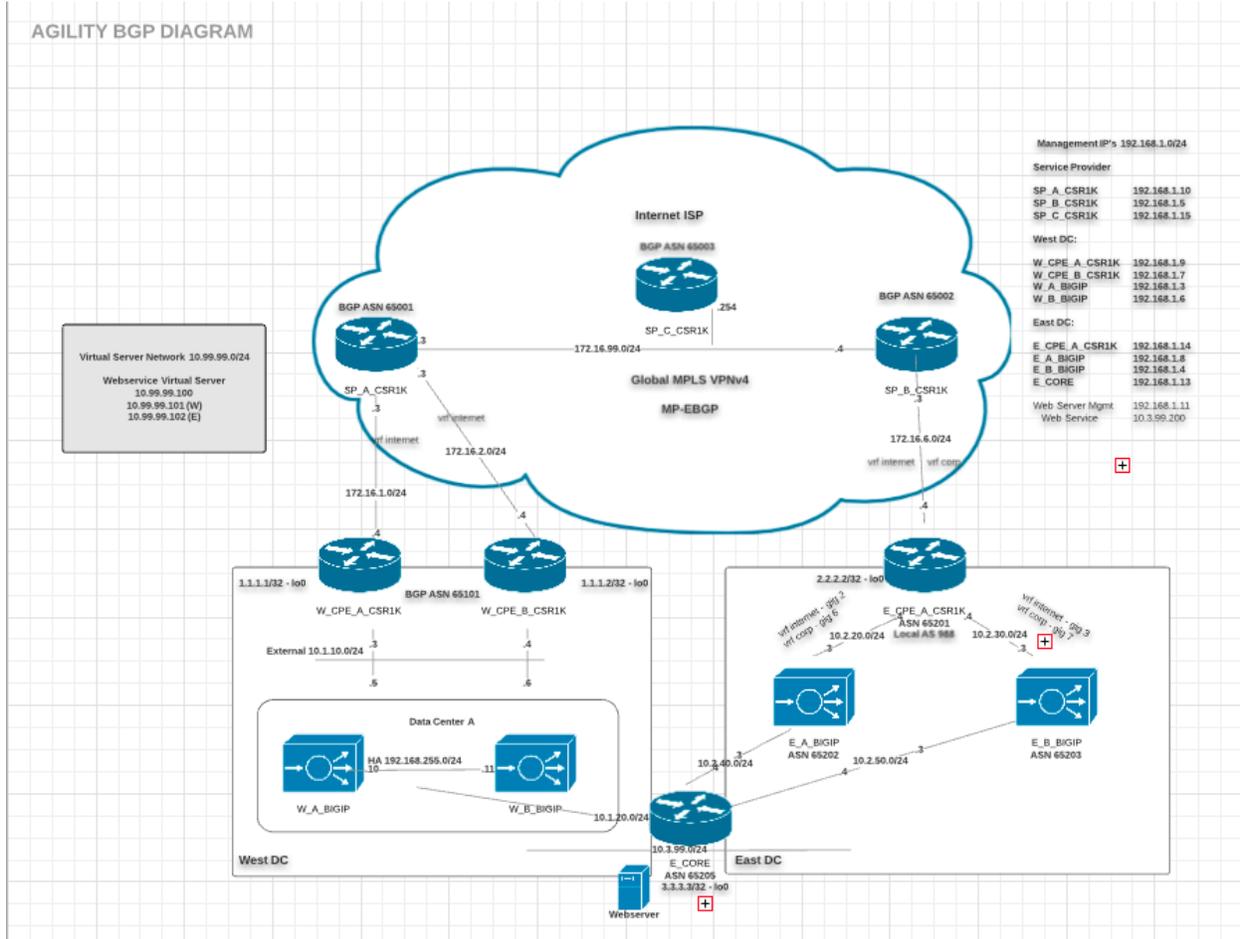
You will be using an ssh client to access all components of this lab. All configuration will be done via the CLI.

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 Linux Jumphost via ssh.

No installation or interaction with your local system is required. If you do not have an ssh client you can install the
<https://putty.org>

5.1.1 Lab Topology



The following components have been included in your lab environment:

- 4 x F5 BIG-IP VE (v13.1)
- 7 x CSR 1000v Virtual Routers
- 1 x Linux Webserver (xubuntu 14.04)
- 1 x Linux Jumphost

Lab Components

The following table lists the management IP Addresses and Credentials for all components:

Table 5.1: Management interfaces. All other interfaces are on diagram

Host	Management IP	Credentials
Linux Jump host	<Public IP on work-sheet>	ubuntu:Agility2018f5
W_A_BIGIP	192.168.1.3	root:default
W_B_BIGIP	192.168.1.6	root:default
W_CPE_A_CSR1K	192.168.1.9	root:default
W_CPE_B_CSR1K	192.168.1.7	root:default
E_A_BIGIP	192.168.1.8	root:default
E_B_BIGIP	192.168.1.4	root:default
E_CPE_A_CSR1K	192.168.1.14	root:default
SP_A_CSR1K	192.168.1.10	root:default
SP_B_CSR1K	192.168.1.5	root:default
SP_C_CSR1K	192.168.1.15	root:default
Web Service	10.3.99.200	

Note: Additional Credentials - enable password or privileged mode on the CSR1000v routers: **san-fran**
Privileged mode allows you to run commands that might be disruptive to network traffic or enter configuration mode.

5.1.2 BIG-IP CLI Primer

bash shell

Default linux shell when you log in to the BIG-IP as root

Common commands you can run from here:

- ping
- traceroute
- tcpdump
- imish (used to access the ZebOS router shell)
- tmsh (used to access the traffic management shell for BIG-IP configuration)

tm shell (tmsh)

From bash any command you know the syntax of can be entered directly by preceding it with the 'tmsh' command

```
[admin@BIGIP131:Active:In Sync] ~ # tmsh run sys failover standby
```

For context based help and tab completion you can enter the 'tmsh' command and go directly to the prompt.

```
[admin@BIGIP131:Active:In Sync] ~ # tmsh
admin@(BIGIP131) (cfg-sync In Sync) (Active) (/Common) (tmsh) #
```

Using the '?' for context aware help

```
admin@(BIGIP131) (cfg-sync In Sync) (Active) (/Common) (tmsh)# create ltm ?
Modules:
  auth          Virtual server authentication configuration
  cipher        Cipher Rule and Group configuration
  classification Traffic Classification
  data-group    Data group configuration
  dns           DNS configuration
  html-rule     Generalized HTML rule-based patcher
  message-routing Message routing framework configuration
  monitor       LTM monitor templates
  persistence   Virtual server persistence configuration
  profile       Virtual server profile configuration
  tacdb         TACDB configuration.
Components:
  eviction-policy Defines an eviction policy, used to select which flows to evict
↳when approaching limits.
  ifile           iFile Configuration
  nat            Network address translation configuration
  node           Node specific pool member configuration
  policy         Centralized Policy Matching configuration
  policy-strategy Centralized Policy Matching rule selection strategy
  pool          Load balancing pool configuration
  rule          iRules configuration
  rule-profiler
  snat          Secure network address translation (SNAT) configuration
  snat-translation SNAT translation address configuration
  snatpool      Collections of SNAT translation addresses
  traffic-class Traffic Class Configuration
  virtual       Virtual server configuration
  virtual-address Virtual server IP address configuration
```

Using 'tab' to offer possible completions

```
admin@(BIGIP131) (cfg-sync In Sync) (Active) (/Common) (tmsh)# create ltm
Modules:
  auth          classification  dns          message-routing
↳persistence   tacdb
  cipher        data-group    html-rule    monitor      profile
Components:
  eviction-policy nat          policy       pool         rule-
↳profiler      snat-translation traffic-class virtual-address
  ifile         node         policy-strategy rule         snat
↳
  snatpool     virtual
```

By partially typing a keyword you can use 'tab' to either auto-complete or give you the list of options.

```
admin@(BIGIP131) (cfg-sync In Sync) (Active) (/Common) (tmsh)# create ltm po
Components:
  policy          policy-strategy  pool
```

To exit the tmsh shell just type 'quit':

```
admin@(BIGIP131) (cfg-sync In Sync) (Active) (/Common) (tmsh)# quit
[admin@BIGIP131:Active:In Sync] ~ #
```

ZebOS router shell (imish)

From the BIG-IP bash cli you can enter the ZebOS router shell by typing the command 'imish'. If you have multiple route domains, you can specify which by adding the '-r #'

Route domain '0' is default.

```
[admin@BIGIP131:Active:In Sync] ~ # imish -r 0
```

From there you can enter privileged mode which will allow you to run administrative level commands and also enter in to the configuration mode.

Note how the prompt changes from '>' to '#'. This is how you can determine which mode you are in. If a command fails, it may be because you do not have sufficient privileges. There is no enable password on the ZebOS instances for this lab.

```
[root@E_A_BIGIP-13:Active:Standalone] log # imish
E_A_BIGIP-13.local[0]>enable
E_A_BIGIP-13.local[0]#
```

From privileged mode, you can type 'configure terminal' to enter configuration mode. You can shorten this as shown below with the command 'conf t' as you will see often in this lab.

```
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

Note: Cisco devices in this lab behave similar to the ZebOS cli with the exception of requiring a secondary enable password to access privileged mode.

5.2 Module 1 – Resilient Data Center Architectures

In this module you will learn how to configure BGP on the BIG-IP and understand how High Availability configurations work with ZebOS and external network devices.

On the West Data Center you will configure a standard HA Active/Standby Pair of BIG-IP devices and peer with your CPE devices connecting to the Internet. It is designed to help you understand how ZebOS interacts with the BIG-IP and any BGP peers.

On the East Data Center you will configure two redundant BIG-IP devices using BGP as the failover mechanism itself. You will also configure and test controlling failover between the two data centers.

Lastly in Lab 3 you will go through some basic troubleshooting steps.

5.2.1 Lab 1: West Data Center Configuration

In this lab you will be configuring a BIG-IP HA cluster and setting it to advertise available virtual servers through the BGP protocol.

Configure the BIG-IP HA Cluster in the West Data Center

From the jumhost ssh to the W_A_BIGIP-13

```
root@jumphost:~# ssh -l root 192.168.1.3
```

```
tmsh modify sys ntp servers add { pool.ntp.org }
tmsh mv cm device bigip1 W_A_BIGIP-13.local
tmsh modify cm device W_A_BIGIP-13.local configsync-ip 192.168.255.10
tmsh modify cm device W_A_BIGIP-13.local unicast-address { { ip 192.168.255.10 } }
```

From the jumphost ssh to the W_B_BIGIP-13

```
root@jumphost:~# ssh -l root 192.168.1.6
```

```
tmsh modify sys ntp servers add { pool.ntp.org }
tmsh mv cm device bigip1 W_B_BIGIP-13.local
tmsh modify cm device W_B_BIGIP-13.local configsync-ip 192.168.255.11
tmsh modify cm device W_B_BIGIP-13.local unicast-address { { ip 192.168.255.11 } }
```

Create the trust domain and sync-failover group from one of the two BIGIP's:

From W_B_BIGIP-13

```
tmsh modify cm trust-domain /Common/Root ca-devices add { 192.168.255.10 } name W_A_
↪BIGIP-13.local username admin password admin

tmsh create cm device-group sync-failover type sync-failover auto-sync enabled save-
↪on-auto-sync true devices add { W_A_BIGIP-13.local W_B_BIGIP-13.local }
```

Run the initial configuration sync from one of the two BIGIP's:

```
tmsh run cm config-sync to-group sync-failover

[root@W_B_BIGIP-13:Active:Changes Pending] config #
[root@W_B_BIGIP-13:Active:Awaiting Initial Sync] config #
[root@W_B_BIGIP-13:Active:Awaiting Initial Sync] config #
[root@W_B_BIGIP-13:Active:In Sync] config #
[root@W_B_BIGIP-13:Active:In Sync] config #
```

Save the configuration on **both** BIG-IP devices

```
tmsh save sys config partitions all
```

Create an application configuration for a virtual server and a pool member

Create the following virtual server and pool member

```
tmsh create ltm pool pool1 members add { 10.3.99.200:80 } monitor tcp_half_open
tmsh create ltm virtual vip1 destination 10.99.99.100:80 source-address-translation {
↪type automap } pool pool1 profiles add { tcp http }
```

Note: The configuration should now be auto-sync'd to the other device and auto-saved.

Attention: The virtual server is not available.

```
[root@W_A_BIGIP-13:Standby:In Sync] config # tmsh show ltm virtual
-----
Ltm::Virtual Server: vipl
-----
Status
  Availability      : offline
  State             : enabled
  Reason            : The children pool member(s) are down
  CMP               : enabled
  CMP Mode          : all-cpus
  Destination       : 10.99.99.100:80
```

The pool members are unavailable.

```
[root@W_A_BIGIP-13:Standby:In Sync] config # tmsh show ltm pool
-----
Ltm::Pool: pool1
-----
Status
  Availability      : offline
  State             : enabled
  Reason            : The children pool member(s) are down
  Monitor           : tcp_half_open
  Minimum Active Members : 0
  Current Active Members : 0
  Available Members  : 0
  Total Members     : 1
  Total Requests    : 0
  Current Sessions  : 0
```

You need to create a static route to the web server located in the core network on both BIG-IPs.

```
tmsh create net route webservers network 10.3.99.0/24 gw 10.1.20.254
```

Note: This only needs to be configured on one device as seen below

```
[root@W_B_BIGIP-13:Active:In Sync] config # tmsh create net route webservers network_
↪10.3.99.0/24 gw 10.1.20.254
[root@W_B_BIGIP-13:Active:Not All Devices Synced] config #
[root@W_B_BIGIP-13:Active:In Sync] config #
```

Your virtual server should now show available.

```
tmsh show ltm virtual
-----
Ltm::Virtual Server: vipl
-----
Status
  Availability      : available
  State             : enabled
  Reason            : The virtual server is available
  CMP               : enabled
  CMP Mode          : all-cpus
  Destination       : 10.99.99.100:80
```

Note: From the jump host you can now try to reach the website

Either open a web browser and browse to <http://10.99.99.100> or from the jump host CLI, type:

```
curl http://10.99.99.100
```

Why is it not available?

Configure the route advertisement on the BIG-IP cluster

On both W_A_BIGIP and W_B_BIGIP configure the following:

```
tmsh modify net route-domain 0 routing-protocol add { BGP }
```

Note: The following just needs to be applied on just one device

```
tmsh modify ltm virtual-address 10.99.99.100 route-advertisement all
```

Note:

route-advertisement Specifies the route advertisement setting for the virtual address. The default value is disabled.

enabled or selective Route is advertised when virtual-address is available.

disabled Route is not advertised.

always Route is advertised regardless of the availability status.

any Route is advertised when any of the contributing virtual server is available.

all Route is advertised when all of the contributing virtual server is available.

Because the previous commands are also base configuration you should save the configuration on both devices.

```
tmsh save sys config
```

Now you can begin to configure the routing protocol in ZebOS. To reach the ZebOS CLI, from the BIG-IP BASH shell.

```
[root@W_B_BIGIP-13:Active:In Sync] config # imish
[root@W_B_BIGIP-13:Active:In Sync] config # enable
W_B_BIGIP-13.local[0]#show ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default
C       10.1.10.0/24 is directly connected, external
C       10.1.20.0/24 is directly connected, internal
K       10.3.99.0/24 [0/0] via 10.1.20.254, internal
```

```
K      10.99.99.100/32 [0/0] is directly connected, tmm
C      127.0.0.1/32 is directly connected, lo
C      127.1.1.254/32 is directly connected, tmm
C      192.168.255.0/24 is directly connected, ha
```

Gateway of last resort is not set

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
[root@W_A_BIGIP-13:Active:In Sync] config # enable
W_A_BIGIP-13.local[0]>sh ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default
C      10.1.10.0/24 is directly connected, external
C      10.1.20.0/24 is directly connected, internal
K      10.3.99.0/24 [0/0] via 10.1.20.254, internal
C      127.0.0.1/32 is directly connected, lo
C      127.1.1.254/32 is directly connected, tmm
C      192.168.255.0/24 is directly connected, ha
```

Gateway of last resort is not set

Note: The two kernel routes designated with a 'K'. These are routes provided to ZebOS from tmm. The first route you may recall is 10.3.99.0/24. This is the static route you configured via tmsh. The second route, 10.99.99.100/32 is the virtual server.

Note: Why does the kernel route for 10.99.99.0/32 only show up in one of the devices?

Force a failover event to occur on the 'Active' device and validate.

```
W_B_BIGIP-13.local[0]#quit
```

```
tmsh run sys failover standby
```

Note: This needs to be run from whichever device in your HA Pair is currently active. Once you 'quit' imish, it can be determined from the bash shell prompt.

```
[root@W_A_BIGIP-13:Active:In Sync] [root@W_B_BIGIP-13:Active:In Sync]
```

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
[root@W_B_BIGIP-13:Active:In Sync] config # enable
W_B_BIGIP-13.local[0]#sh ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default
```

```
C    10.1.10.0/24 is directly connected, external
C    10.1.20.0/24 is directly connected, internal
K    10.3.99.0/24 [0/0] via 10.1.20.254, internal
C    127.0.0.1/32 is directly connected, lo
C    127.1.1.254/32 is directly connected, tmm
C    192.168.255.0/24 is directly connected, ha
```

Gateway of last resort is not set

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
[root@W_B_BIGIP-13:Active:In Sync] config # enable
W_A_BIGIP-13.local[0]#show ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default
```

```
C    10.1.10.0/24 is directly connected, external
C    10.1.20.0/24 is directly connected, internal
K    10.3.99.0/24 [0/0] via 10.1.20.254, internal
K    10.99.99.100/32 [0/0] is directly connected, tmm
C    127.0.0.1/32 is directly connected, lo
C    127.1.1.254/32 is directly connected, tmm
C    192.168.255.0/24 is directly connected, ha
```

Gateway of last resort is not set

Configure the iBGP session with the West CPE devices W_CPE_A_CSR1k and W_CPE_B_CSR1k. The CPE configuration is already done for you so you only need to configure the two BIG-IP devices.

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
W_A_BIGIP-13.local[0]>enable
W_A_BIGIP-13.local[0]#config t
W_A_BIGIP-13.local[0] (config)#router bgp 65101
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 remote-as 65101
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 description W_CPE_A
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 remote-as 65101
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 description W_CPE_B
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.6 remote-as 65101
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.6 description W_B_BIGIP-13
W_A_BIGIP-13.local[0] (config-router)# redistribute kernel
W_A_BIGIP-13.local[0] (config)#end
W_A_BIGIP-13.local[0]#wr
Building configuration...
[OK]
W_A_BIGIP-13.local[0]#
```

```
[root@W_B_BIGIP-13:Active:In Sync] config # imish
W_B_BIGIP-13.local[0]>enable
W_B_BIGIP-13.local[0]#config t
W_B_BIGIP-13.local[0] (config)#router bgp 65101
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 remote-as 65101
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 description W_CPE_A
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 remote-as 65101
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 description W_CPE_B
```

```

W_B_BIGIP-13.local[0](config-router)# neighbor 10.1.10.5 remote-as 65101
W_B_BIGIP-13.local[0](config-router)# neighbor 10.1.10.5 description W_A_BIGIP-13
W_B_BIGIP-13.local[0](config-router)# redistribute kernel
W_B_BIGIP-13.local[0](config-router)#end
W_B_BIGIP-13.local[0]#wr
Building configuration...
[OK]
W_B_BIGIP-13.local[0]#

```

Note: BGP has not come up between the BIG-IP's. Why is that?

```

W_A_BIGIP-13.local[0]#sh ip bgp summary
BGP router identifier 192.168.255.10, local AS number 65101
BGP table version is 3
4 BGP AS-PATH entries
0 BGP community entries

```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.1.10.3	4	65101	41	32	3	0	0	00:00:26	6
10.1.10.4	4	65101	40	30	3	0	0	00:00:26	6
10.1.10.6	4	65101	7	7	0	0	0	00:01:04	Active

Configure port lockdown on BIG-IP's to allow BGP via external interface on both BIG-IP:

```

tmsh list net self external allow-service
net self external {
    allow-service none
}

```

```

tmsh modify net self external allow-service add { tcp:179 }

```

```

tmsh list net self external allow-service
net self external {
    allow-service {
        tcp:bgp
    }
}

```

Validate BGP is now established between both BIGIP's

```

W_B_BIGIP-13.local[0]#sh ip bgp summary
BGP router identifier 192.168.255.11, local AS number 65101
BGP table version is 5
4 BGP AS-PATH entries
0 BGP community entries

```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.1.10.3	4	65101	32	23	5	0	0	00:03:16	6
10.1.10.4	4	65101	32	23	5	0	0	00:03:16	6
10.1.10.5	4	65101	12	11	5	0	0	00:00:07	2

Total number of neighbors 3

Note: iBGP requires that you configure a full mesh. But in this simplified scenario you do not need to

configure a full mesh for iBGP because there are only four routers and you've got the BIG-IP HA Cluster and your outbound gateway. Your topology may vary and you may want to configure a full mesh if there are other devices. Additionally in a larger topology you may want to configure eBGP between the CPE and the BIG-IP clusters. This is explored in more depth when you configure the East data center.

Note: The redistribute kernel command is the secret sauce that allows tmm routes from the BIG-IP to be advertised to BGP peers.

Validate that the website is accessible from the jumphost

```
root@jumphost:~# curl 10.99.99.100
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Validate your configuration on both of the CPE routers

You can telnet to either of the CPE devices

```
csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 2 subnets
C       1.1.1.1 is directly connected, Loopback100
O       1.1.1.2 [110/2] via 10.1.10.4, 02:49:46, GigabitEthernet4
    3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [20/0] via 172.16.1.3, 00:05:44
    10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.1.10.0/24 is directly connected, GigabitEthernet4
L       10.1.10.3/32 is directly connected, GigabitEthernet4
B       10.99.99.100/32 [200/0] via 10.1.10.5, 00:00:07
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.1.0/24 is directly connected, GigabitEthernet2
L       172.16.1.4/32 is directly connected, GigabitEthernet2
B       172.16.2.0/24 [20/0] via 172.16.1.3, 00:35:33
C       172.16.3.0/24 is directly connected, GigabitEthernet3
L       172.16.3.4/32 is directly connected, GigabitEthernet3
B       172.16.6.0/24 [20/0] via 172.16.1.3, 00:49:06
```

Note: The BGP route should be pointing to the 'Active' device in the HA cluster.

Initiate a failover event to determine how the route update happens.

From whichever BIG-IP device is active:

```
tmsm run sys failover standby
```

```
csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PFR

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 2 subnets
C       1.1.1.1 is directly connected, Loopback100
O       1.1.1.2 [110/2] via 10.1.10.4, 02:51:26, GigabitEthernet4
    3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [20/0] via 172.16.1.3, 00:07:24
    10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.1.10.0/24 is directly connected, GigabitEthernet4
L       10.1.10.3/32 is directly connected, GigabitEthernet4
B       10.99.99.100/32 [200/0] via 10.1.10.6, 00:00:02
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.1.0/24 is directly connected, GigabitEthernet2
L       172.16.1.4/32 is directly connected, GigabitEthernet2
B       172.16.2.0/24 [20/0] via 172.16.1.3, 00:37:13
C       172.16.3.0/24 is directly connected, GigabitEthernet3
L       172.16.3.4/32 is directly connected, GigabitEthernet3
B       172.16.6.0/24 [20/0] via 172.16.1.3, 00:50:46
```

Note: If you were watching the update time you'll notice it takes a bit for the next-hop address to change. This can be modified with timers, or you can use a floating self-ip between an HA pair of BIG-IP's to minimize the update time because the Next-hop entry for the floating address never needs to be updated.

Validate that the website is accessible from the jumphost

```
root@jumphost:~# curl 10.99.99.100
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Next we will configure a floating self-ip address to minimize the need for the next-hop to change. On one of the West Data Center BIG-IP devices. Combined with mac-masquerading and connection mirroring this is the least disruptive failover configuration.

```
tmsm create net self float-bgp address 10.1.10.7/24 vlan external traffic-group_
↵traffic-group-1
```

You can either wait until the route update occurs or you can reset the BGP neighbors with the clear ip bgp * command

```
[root@W_B_BIGIP-13:Active:In Sync] config # imish
W_B_BIGIP-13.local[0]>enable
W_B_BIGIP-13.local[0]#clear ip bgp *
```

```
csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 2 subnets
C       1.1.1.1 is directly connected, Loopback100
O       1.1.1.2 [110/2] via 10.1.10.4, 00:31:23, GigabitEthernet4
    2.0.0.0/32 is subnetted, 1 subnets
B       2.2.2.2 [20/0] via 172.16.1.3, 00:30:43
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C       10.1.10.0/24 is directly connected, GigabitEthernet4
L       10.1.10.3/32 is directly connected, GigabitEthernet4
B       10.99.99.100/32 [200/0] via 10.1.10.7, 00:03:06
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.1.0/24 is directly connected, GigabitEthernet2
L       172.16.1.4/32 is directly connected, GigabitEthernet2
B       172.16.2.0/24 [20/0] via 172.16.1.3, 00:30:44
C       172.16.3.0/24 is directly connected, GigabitEthernet3
L       172.16.3.4/32 is directly connected, GigabitEthernet3
B       172.16.6.0/24 [20/0] via 172.16.1.3, 00:30:44
```

On the CPE devices look at the result of the show ip bgp command

```
csr1000v-W_CPE_A#sh ip bgp
BGP table version is 10, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop          Metric LocPrf Weight Path
*>  1.1.1.1/32      0.0.0.0           0           32768  i
r>i  1.1.1.2/32      1.1.1.2           0           100    0  i
* i  2.2.2.2/32      172.16.2.3        0           100    0 65001 65002 65201  i
*>                                172.16.1.3        0           100    0 65001 65002 65201  i
* i  10.3.99.0/24    10.1.20.254       0           100    0  ?
*>i  10.99.99.100/32  10.1.10.7         0           100    0  ?
r i  172.16.1.0/24    172.16.2.3        0           100    0 65001  ?
r>                                172.16.1.3        0           100    0 65001  ?
* i  172.16.2.0/24    172.16.2.3        0           100    0 65001  ?
*>                                172.16.1.3        0           100    0 65001  ?
* i  172.16.6.0/24    172.16.2.3        0           100    0 65001 65002  ?
*>                                172.16.1.3        0           100    0 65001 65002  ?
```

```
csr1000v-W_CPE_A#
```

Note: Compare the show ip route output in the previous step to the show ip bgp output. What looks out of place here? Why is it there? Why does it not show up in the routing table?

Create an aggregate route since many service providers will not accept anything less than a /24

On both of the West BIG-IPs configure ZebOS to only advertise an aggregate route for the virtual server network and filter out the core network advertisement.

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
W_A_BIGIP-13.local[0]>enable
W_A_BIGIP-13.local[0]#config t
W_A_BIGIP-13.local[0] (config)#ip prefix-list PFX_ALLOW_VIPS seq 5 permit 10.99.99.0/24
W_A_BIGIP-13.local[0] (config)#route-map RESTRICT_ADVERTISE permit 10
W_A_BIGIP-13.local[0] (config-route-map)# match ip address prefix-list PFX_ALLOW_VIPS
W_A_BIGIP-13.local[0] (config)#router bgp 65101
W_A_BIGIP-13.local[0] (config-router)# aggregate-address 10.99.99.0/24
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 route-map RESTRICT_ADVERTISE_
↪out
W_A_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 route-map RESTRICT_ADVERTISE_
↪out
W_A_BIGIP-13.local[0] (config-router)#end
W_A_BIGIP-13.local[0]#wr
Building configuration...
[OK]
W_A_BIGIP-13.local[0]#
```

```
[root@W_A_BIGIP-13:Active:In Sync] config # imish
W_A_BIGIP-13.local[0]>enable
W_B_BIGIP-13.local[0]# config t
W_B_BIGIP-13.local[0] (config)#ip prefix-list PFX_ALLOW_VIPS seq 5 permit 10.99.99.0/24
W_B_BIGIP-13.local[0] (config)#route-map RESTRICT_ADVERTISE permit 10
W_B_BIGIP-13.local[0] (config-route-map)# match ip address prefix-list PFX_ALLOW_VIPS
W_B_BIGIP-13.local[0] (config)#router bgp 65101
W_B_BIGIP-13.local[0] (config-router)# aggregate-address 10.99.99.0/24
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.3 route-map RESTRICT_ADVERTISE_
↪out
W_B_BIGIP-13.local[0] (config-router)# neighbor 10.1.10.4 route-map RESTRICT_ADVERTISE_
↪out
W_B_BIGIP-13.local[0] (config-router)#end
W_B_BIGIP-13.local[0]#wr
Building configuration...
[OK]
W_B_BIGIP-13.local[0]#
```

Validate on either of the CPE routers. You can either wait until the route update occurs or you can reset the BGP neighbors with the clear ip bgp * command

Clear ip bgp * on the active BIG-IP such that /32 is no longer advertised.

```
[root@W_B_BIGIP-13:Active:In Sync] config # imish
W_B_BIGIP-13.local[0]>enable
W_B_BIGIP-13.local[0]#clear ip bgp *
```

```

csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

```

Gateway of last resort is not set

```

       1.0.0.0/32 is subnetted, 2 subnets
C       1.1.1.1 is directly connected, Loopback100
O       1.1.1.2 [110/2] via 10.1.10.4, 03:04:37, GigabitEthernet4
       2.0.0.0/32 is subnetted, 1 subnets
B       2.2.2.2 [20/0] via 172.16.1.3, 00:06:50
       10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C       10.1.10.0/24 is directly connected, GigabitEthernet4
L       10.1.10.3/32 is directly connected, GigabitEthernet4
B       10.99.99.0/24 [200/0] via 10.1.10.7, 00:06:50
       172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.1.0/24 is directly connected, GigabitEthernet2
L       172.16.1.4/32 is directly connected, GigabitEthernet2
B       172.16.2.0/24 [20/0] via 172.16.1.3, 00:06:50
C       172.16.3.0/24 is directly connected, GigabitEthernet3
L       172.16.3.4/32 is directly connected, GigabitEthernet3
B       172.16.6.0/24 [20/0] via 172.16.1.3, 00:06:50

```

Validate that the website is accessible from the jumphost.

```

root@jumphost:~# curl 10.99.99.100
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

```

Add another virtual server and test the behavior of failover of the aggregate route.

Add another virtual server within the VIP subnet.

```

tmsh create ltm pool pool2 members add { 10.3.99.200:8081 } monitor tcp_half_open
tmsh create ltm virtual vip2 destination 10.99.99.101:80 source-address-translation {
↳ type automap } pool pool2 profiles add { tcp http }

```

Validate that the website is accessible from the jumphost

```

root@jumphost:~# curl 10.99.99.101
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

```

Disable the first pool you created.

```

tmsh modify ltm pool pool1 members modify { 10.3.99.200:http { state user-down } }

tmsh show ltm virtual vip1
-----
Ltm::Virtual Server: vip1
-----
Status
    Availability      : offline
    State              : enabled
    Reason             : The children pool member(s) are down
    CMP                : enabled
    CMP Mode           : all-cpus
    Destination        : 10.99.99.100:80

```

Validate that the website is still accessible from the jumphost

```

root@jumphost:~# curl 10.99.99.101
curl: (7) Failed to connect to 10.99.99.101 port 80: No route to host

```

Now look at the routing table on the CPE virtual server.

```

csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 2 subnets
C       1.1.1.1 is directly connected, Loopback100
O       1.1.1.2 [110/2] via 10.1.10.4, 00:57:32, GigabitEthernet4
    2.0.0.0/32 is subnetted, 1 subnets
B       2.2.2.2 [20/0] via 172.16.1.3, 00:15:52
    10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.1.10.0/24 is directly connected, GigabitEthernet4
L       10.1.10.3/32 is directly connected, GigabitEthernet4
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       172.16.1.0/24 is directly connected, GigabitEthernet2
L       172.16.1.4/32 is directly connected, GigabitEthernet2
B       172.16.2.0/24 [20/0] via 172.16.1.3, 00:56:56
C       172.16.3.0/24 is directly connected, GigabitEthernet3
L       172.16.3.4/32 is directly connected, GigabitEthernet3
B       172.16.6.0/24 [20/0] via 172.16.1.3, 00:56:56

```

On the active BIG-IP look at the routing table.

```

W_B_BIGIP-13.local[0]#sh ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2

```

```

i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default

B    1.1.1.1/32 [200/0] via 10.1.10.3, external, 00:57:35
B    1.1.1.2/32 [200/0] via 10.1.10.4, external, 00:57:35
C    10.1.10.0/24 is directly connected, external
C    10.1.20.0/24 is directly connected, internal
K    10.3.99.0/24 [0/0] via 10.1.20.254, internal
C    127.0.0.1/32 is directly connected, lo
C    127.1.1.254/32 is directly connected, tmm
C    192.168.255.0/24 is directly connected, ha

```

Note: There are no kernel routes from tmm. e.g. 10.99.99.0/24, 10.99.99.100, or 10.99.99.101.

Why?

Note: You need to make sure either all virtual-addresses have advertisement-enabled or a healthy majority of them enabled or the aggregate route will be withdrawn. BGP by default needs to have a route from another routing protocol available in order to advertise a route to a BGP neighbor. In this case because only one of the virtual-addresses has it enabled and it goes down, all other addresses within the /24 are considered unreachable and so the route is withdrawn.

Re-enable the pool and then review the BIG-IP and CPE route tables.

```
tmsh modify ltm pool pool1 members modify { 10.3.99.200:http { state user-up } }
```

```

W_B_BIGIP-13.local[0]#sh ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default

B    1.1.1.1/32 [200/0] via 10.1.10.3, external, 00:58:09
B    1.1.1.2/32 [200/0] via 10.1.10.4, external, 00:58:09
C    10.1.10.0/24 is directly connected, external
C    10.1.20.0/24 is directly connected, internal
K    10.3.99.0/24 [0/0] via 10.1.20.254, internal
B    10.99.99.0/24 [200/0] is a summary, Null, 00:00:04
K    10.99.99.100/32 [0/0] is directly connected, tmm
C    127.0.0.1/32 is directly connected, lo
C    127.1.1.254/32 is directly connected, tmm
C    192.168.255.0/24 is directly connected, ha

Gateway of last resort is not set

```

```

csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route

```

```
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override, p - overrides from PfR
```

Gateway of last resort is not set

```
1.0.0.0/32 is subnetted, 2 subnets
C    1.1.1.1 is directly connected, Loopback100
O    1.1.1.2 [110/2] via 10.1.10.4, 01:10:54, GigabitEthernet4
2.0.0.0/32 is subnetted, 1 subnets
B    2.2.2.2 [20/0] via 172.16.1.3, 00:29:14
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C    10.1.10.0/24 is directly connected, GigabitEthernet4
L    10.1.10.3/32 is directly connected, GigabitEthernet4
B    10.99.99.0/24 [200/0] via 10.1.10.7, 00:09:42
172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C    172.16.1.0/24 is directly connected, GigabitEthernet2
L    172.16.1.4/32 is directly connected, GigabitEthernet2
B    172.16.2.0/24 [20/0] via 172.16.1.3, 01:10:18
C    172.16.3.0/24 is directly connected, GigabitEthernet3
L    172.16.3.4/32 is directly connected, GigabitEthernet3
B    172.16.6.0/24 [20/0] via 172.16.1.3, 01:10:18
```

Validate that the websites are accessible from the jumphost

```
root@jumphost:~# curl 10.99.99.100
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

```
root@jumphost:~# curl 10.99.99.101
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Modify the virtual-address for the second virtual server to allow route-advertisement and repeat disabling.

```
tmsh modify ltm virtual-address 10.99.99.101 route-advertisement selective
tmsh modify ltm pool pool1 members modify { 10.3.99.200:http { state user-down } }
```

```
csr1000v-W_CPE_A#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override, p - overrides from PfR
```

Gateway of last resort is not set

```
1.0.0.0/32 is subnetted, 2 subnets
C    1.1.1.1 is directly connected, Loopback100
```

```

O    1.1.1.2 [110/2] via 10.1.10.4, 01:21:27, GigabitEthernet4
    2.0.0.0/32 is subnetted, 1 subnets
B    2.2.2.2 [20/0] via 172.16.1.3, 00:39:47
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C    10.1.10.0/24 is directly connected, GigabitEthernet4
L    10.1.10.3/32 is directly connected, GigabitEthernet4
B    10.99.99.0/24 [200/0] via 10.1.10.7, 00:20:15
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C    172.16.1.0/24 is directly connected, GigabitEthernet2
L    172.16.1.4/32 is directly connected, GigabitEthernet2
B    172.16.2.0/24 [20/0] via 172.16.1.3, 01:20:51
C    172.16.3.0/24 is directly connected, GigabitEthernet3
L    172.16.3.4/32 is directly connected, GigabitEthernet3
B    172.16.6.0/24 [20/0] via 172.16.1.3, 01:20:51

```

The route now remained when one of the virtual servers went down.

Validate that .101 website remained up while the .100 is not.

```

root@jumphost:~# curl 10.99.99.100
curl: (56) Recv failure: Connection reset by peer

```

```

root@jumphost:~# curl 10.99.99.101
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

```

You can re-enable the .100 website before moving on to Module 2.

```

tmsh modify ltm pool pool1 members modify { 10.3.99.200:http { state user-up } }

```

Validate that the websites are accessible from the jumphost

```

root@jumphost:~# curl 10.99.99.101
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

root@jumphost:~# curl 10.99.99.100
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

```

Sample configuration

```

Sample complete configuration from one BIG-IP:
!
W_B_BIGIP-13.local[0]#sh run
!
no service password-encryption
!
router bgp 65101
  aggregate-address 10.99.99.0/24

```

```
    redistribute kernel
    neighbor 10.1.10.3 remote-as 65101
    neighbor 10.1.10.3 route-map RESTRICT_ADVERTISE out
    neighbor 10.1.10.4 remote-as 65101
    neighbor 10.1.10.4 route-map RESTRICT_ADVERTISE out
    neighbor 10.1.10.5 remote-as 65101
    !
    ip prefix-list PFX_ALLOW_VIPS seq 5 permit 10.99.99.0/24
    !
    route-map RESTRICT_ADVERTISE permit 10
        match ip address prefix-list PFX_ALLOW_VIPS
    !
    line con 0
        login
    line vty 0 39
        login
    !
end
```

Note: This completes Lab 1

5.2.2 Lab 2: East Data Center Configuration

In this lab you will be configuring two BIG-IP standalone devices and setting it up to advertise available virtual servers through the BGP protocol.

Configure the BIG-IP East Data Center

From the jumphost ssh to the E_A_BIGIP-13

```
root@jumphost:~# ssh -l root 192.168.1.8
```

```
tmsh modify sys ntp servers add { pool.ntp.org }
```

From the jumphost ssh to the E_B_BIGIP-13

```
root@jumphost:~# ssh -l root 192.168.1.4
```

```
tmsh modify sys ntp servers add { pool.ntp.org }
```

Save the configuration on both devices

```
tmsh save sys config partitions all
```

Configure the eBGP session between E_A_BIGIP-13 and the East Core via csr1000v-CORE.

Note: The CORE Router configuration is already done for you so you only need to configure the two EAST BIG-IP devices.

E_A_BIGIP-13: On E_A_BIGIP configure the following to enable BGP for route-domain 0.

```
tmsh modify net route-domain 0 routing-protocol add { BGP }  
  
tmsh save sys config partitions all
```

E_A_BIGIP-13: Bring up BGP with the East Core and configure the keepalive interval to 3 seconds and hold time to 9 seconds via neighbor statement instead of using the default values of 30 seconds for keepalive and 180 seconds for hold time:

```
[root@E_A_BIGIP-13:Active:In Sync] config # imish -r 0  
E_A_BIGIP-13.local[0]>enable  
E_A_BIGIP-13.local[0]#config t  
E_A_BIGIP-13.local[0](config)#router bgp 65202  
E_A_BIGIP-13.local[0](config-router)# neighbor 10.2.40.4 remote-as 65205  
E_A_BIGIP-13.local[0](config-router)# neighbor 10.2.40.4 description E_CORE  
E_A_BIGIP-13.local[0](config-router)# neighbor 10.2.40.4 timers 3 9  
E_A_BIGIP-13.local[0](config-router)# end  
E_A_BIGIP-13.local[0]#write mem
```

E_A_BIGIP-13: Verify eBGP adjacencies are up between E_A_BIGIP-13 and the East Core router - csr1000v-CORE.

```
E_A_BIGIP-13.local[0]#sh ip bgp sum  
BGP router identifier 10.2.40.3, local AS number 65202  
BGP table version is 2  
1 BGP AS-PATH entries  
0 BGP community entries  
  
Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ  OutQ Up/Down  State/PfxRcd  
10.2.40.4     4 65205      5       2        2    0    0 00:00:22      2
```

E_A_BIGIP-13: Verify route for the webservice via 10.3.99.0/24 is installed in routing table after eBGP is established between E_A_BIGIP-13 and the East Core router - csr1000v-CORE.

```
E_A_BIGIP-13.local[0]#sh ip route | i 10.2.40.4  
B          3.3.3.3/32 [20/0] via 10.2.40.4, internal, 00:01:54  
B          10.3.99.0/24 [20/0] via 10.2.40.4, internal, 00:01:54  
E_A_BIGIP-13.local[0]
```

Configure the eBGP session between E_B_BIGIP-13 and the East Core via csr1000v-CORE.

Note: The CORE Router configuration is already done for you so you only need to configure the two EAST BIG-IP devices.

E_B_BIGIP-13: On E_B_BIGIP configure the following to enable BGP for route-domain 0.

```
tmsh modify net route-domain 0 routing-protocol add { BGP }  
  
tmsh save sys config partitions all
```

E_B_BIGIP-13: Bring up BGP with the East Core and configure the keepalive interval to 3 seconds and hold time to 9 seconds via neighbor statement instead of using the default values of 30 seconds for keepalive and 180 seconds for hold time:

```
[root@E_B_BIGIP-13:Active:In Sync] config # imish -r 0
E_B_BIGIP-13.local[0]>enable
E_B_BIGIP-13.local[0]#config t
E_B_BIGIP-13.local[0] (config)#router bgp 65203
E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.50.4 remote-as 65205
E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.50.4 description E_CORE
E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.50.4 timers 3 9
E_A_BIGIP-13.local[0] (config-router)# end
E_A_BIGIP-13.local[0]#write mem
```

E_B_BIGIP-13: Verify eBGP adjacencies are up between E_B_BIGIP-13 and the East Core router - csr1000v-CORE.

```
E_B_BIGIP-13.local[0]#sh ip bgp sum
BGP router identifier 10.2.50.3, local AS number 65203
BGP table version is 2
1 BGP AS-PATH entries
0 BGP community entries

Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ  OutQ Up/Down  State/PfxRcd
10.2.50.4      4 65205      4       2        1    0    0 00:00:16      2

Total number of neighbors 1
E_B_BIGIP-13.local[0]#
```

E_B_BIGIP-13: Verify route for the webservice via 10.3.99.0/24 is installed in routing table after eBGP is established between E_B_BIGIP-13 and the East Core router - csr1000v-CORE.

```
E_B_BIGIP-13.local[0]#sh ip route | i 10.2.50.4
B          3.3.3.3/32 [20/0] via 10.2.50.4, internal, 00:06:52
B          10.3.99.0/24 [20/0] via 10.2.50.4, internal, 00:06:52
E_B_BIGIP-13.local[0]#
```

Validate Webserver Connectivity via Core Network:

Verify that you can reach the webserver on the core network with icmp ping and curl from both BIG-IPs.

E_A_BIGIP-13: Ping the webserver @ 10.3.99.200 via the core network from E_A_BIGIP-13. Note the ping may not work immediately and could take a few seconds. Also Note the escape sequence for icmp ping on the BIGIP is "CTRL + C"

```
[root@E_A_BIGIP-13:Active:Standalone] config # ping 10.3.99.200
PING 10.3.99.200 (10.3.99.200) 56(84) bytes of data.
64 bytes from 10.3.99.200: icmp_seq=1 ttl=63 time=8.51 ms
64 bytes from 10.3.99.200: icmp_seq=2 ttl=63 time=8.12 ms
^C
--- 10.3.99.200 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1823ms
rtt min/avg/max/mdev = 8.121/8.318/8.516/0.217 ms
```

E_A_BIGIP-13: Curl the webserver @ 10.3.99.200 via the core network from E_A_BIGIP-13.

```
[root@E_A_BIGIP-13:Active:Standalone] config # curl 10.3.99.200
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

E_B_BIGIP-13: Ping the webserver @ 10.3.99.200 via the core network from E_B_BIGIP-13. Note the ping may not work immediately and could take a few seconds. Also Note the escape sequence for icmp ping on the BIGIP is “CTRL + C”

```
[root@E_B_BIGIP-13:Active:Standalone] config # ping 10.3.99.200
PING 10.3.99.200 (10.3.99.200) 56(84) bytes of data.
64 bytes from 10.3.99.200: icmp_seq=1 ttl=63 time=6.06 ms
64 bytes from 10.3.99.200: icmp_seq=2 ttl=63 time=9.31 ms
^C
--- 10.3.99.200 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1726ms
rtt min/avg/max/mdev = 6.068/7.692/9.317/1.626 ms
```

E_B_BIGIP-13: Curl the webserver @ 10.3.99.200 via the core network from E_B_BIGIP-13.

```
[root@E_B_BIGIP-13:Active:Standalone] config # curl 10.3.99.200
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
[root@E_B_BIGIP-13:Active:Standalone] config #
```

Create an application configuration for a virtual server and a pool member on E_A_BIGIP-13:

E_A_BIGIP-13: Create the following virtual server and pool member on E_A_BIGIP-13:

```
tmsh create ltm pool pool1 members add { 10.3.99.200:80 } monitor tcp_half_open
tmsh create ltm virtual vip1 destination 10.99.99.102:80 source-address-translation {
↳type automap } pool pool1 profiles add { tcp http }

tmsh save sys config partitions all
```

E_A_BIGIP-13: Your virtual server should now show available on E_A_BIGIP-13:

```
[root@E_A_BIGIP-13:Active:Standalone] config # tmsh show ltm virtual

-----
Ltm::Virtual Server: vip1
-----

Status
  Availability      : available
  State             : enabled
  Reason            : The virtual server is available
  CMP               : enabled
  CMP Mode          : all-cpus
  Destination       : 10.99.99.102:80
```

Configure the route advertisement on the E_A_BIGIP-13:

E_A_BIGIP-13: Configure the eBGP session on E_A_BIGIP to East CPE_A. The CPE configuration is already done for you so you only need to configure the BIGIP side of session.

```
[root@E_A_BIGIP-13:Active:In Sync] config # imish -r 0
E_A_BIGIP-13.local[0]>enable
E_A_BIGIP-13.local[0]#config t
E_A_BIGIP-13.local[0] (config)#router bgp 65202
E_A_BIGIP-13.local[0] (config-router)# neighbor 10.2.20.4 remote-as 65201
E_A_BIGIP-13.local[0] (config-router)# neighbor 10.2.20.4 description E_CPE_A
E_A_BIGIP-13.local[0] (config-router)# neighbor 10.2.20.4 timers 3 9
E_A_BIGIP-13.local[0] (config-router)# end
E_A_BIGIP-13.local[0]#write mem
```

E_A_BIGIP-13: Verify eBGP adjacencies are up between E_A_BIGIP_13 and the East CPE_A.

```
E_A_BIGIP-13.local[0]#sh ip bgp sum
BGP router identifier 10.2.40.3, local AS number 65202
BGP table version is 3
5 BGP AS-PATH entries
0 BGP community entries

Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ  OutQ Up/Down  State/PfxRcd
10.2.20.4     4 65201     27     21       3    0    0 00:00:53      8
10.2.40.4     4 65205    157    141       3    0    0 01:07:25      2
```

E_A_BIGIP-13: On E_A_BIGIP configure the following network statement for 10.99.99.0/24 such that prefix is originated locally:

```
[root@E_A_BIGIP-13:Active:Standalone] config # imish -r 0
E_A_BIGIP-13.local[0]>en
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
E_A_BIGIP-13.local[0] (config)#router bgp 65202
E_A_BIGIP-13.local[0] (config-router)#network 10.99.99.0/24
E_A_BIGIP-13.local[0] (config-router)#end
E_A_BIGIP-13.local[0]#
```

E_A_BIGIP-13: On E_A_BIGIP verify 10.99.99.0/24 is being locally originated which can be seen with "Local":

```
E_A_BIGIP-13.local[0]#sh ip bgp 10.99.99.0/24 | b Local

...skipping
Local
0.0.0.0 from 0.0.0.0 (10.2.40.3)
  Origin IGP, localpref 100, weight 32768, valid, sourced, local, best
  Last update: Mon Jul 16 18:07:54 2018
```

E_A_BIGIP-13: On E_A_BIGIP verify 10.99.99.0/24 is being advertised outbound to East CPE device via E_CPE_A_CSR1k:

```
E_A_BIGIP-13.local[0]#sh ip bgp neighbor 10.2.20.4 advertised-routes | i 10.99.99.0/24
*> 10.99.99.0/24    10.2.20.3                100      32768 i
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is learning the 10.99.99.0/24 inbound from E_A_BIGIP:

Note: You can telnet to the CPE devices using the BGP neighbor IP address from Zebos using root/default for user/pass:

Example telnet from E_A_BIGIP-13 to East CPE Device @ 10.2.20.4:

```
E_A_BIGIP-13.local[0]#telnet 10.2.20.4
Trying 10.2.20.4...
Connected to 10.2.20.4.
Escape character is '^]'.

User Access Verification

Username: root
Password:
csr1000v-E_CPE_A>
```

E_CPE_A_CSR1k: Continued... Verify that E_CPE_A_CSR1k is learning the 10.99.99.0/24 inbound from E_A_BIGIP:

```
csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet neighbors 10.2.20.3 routes
BGP table version is 26, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop           Metric LocPrf Weight Path
Route Distinguisher: 65201:1000 (default for vrf internet)
*>  3.3.3.3/32      10.2.20.3         4294967295           0 65202 65205 i
*>  10.99.99.0/24   10.2.20.3         4294967295           0 65202 i

Total number of prefixes 2
csr1000v-E_CPE_A>
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is installing 10.99.99.0/24 from E_A_BIGIP:

```
csr1000v-E_CPE_A>show ip route vrf internet

Routing Table: internet
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

 1.0.0.0/32 is subnetted, 2 subnets
B       1.1.1.1 [20/4294967294] via 172.16.6.3, 22:12:30
B       1.1.1.2 [20/4294967294] via 172.16.6.3, 22:12:32
 3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [20/4294967294] via 10.2.20.3, 01:27:19
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.2.20.0/24 is directly connected, GigabitEthernet2
L       10.2.20.4/32 is directly connected, GigabitEthernet2
C       10.2.30.0/24 is directly connected, GigabitEthernet3
```

```

L      10.2.30.4/32 is directly connected, GigabitEthernet3
B      10.99.99.0/24 [20/4294967294] via 10.2.20.3, 01:22:58
      99.0.0.0/24 is subnetted, 1 subnets
B      99.99.99.0 [20/4294967294] via 172.16.6.3, 22:13:59
      172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
B      172.16.1.0/24 [20/4294967294] via 172.16.6.3, 22:12:40
B      172.16.2.0/24 [20/4294967294] via 172.16.6.3, 22:12:40
C      172.16.6.0/24 is directly connected, GigabitEthernet5
L      172.16.6.4/32 is directly connected, GigabitEthernet5
B      172.16.99.0/24 [20/0] via 172.16.6.3, 22:13:59
csr1000v-E_CPE_A>

```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is installing specific 10.99.99.0/24 from E_A_BIGIP using specific ip route command:

```

csr1000v-E_CPE_A#sh ip route vrf internet 10.99.99.0 255.255.255.0

Routing Table: internet
Routing entry for 10.99.99.0/24
  Known via "bgp 65201", distance 20, metric 4294967294
  Tag 65202, type external
  Last update from 10.2.20.3 00:00:02 ago
  Routing Descriptor Blocks:
  * 10.2.20.3, from 10.2.20.3, 00:00:02 ago
    Route metric is 4294967294, traffic share count is 1
    AS Hops 1
    Route tag 65202
    MPLS label: none

```

csr1000v-SP_C: Verify that csr1000v-SP_C is installing 10.99.99.0/24 via East DC because Origin attribute is IGP versus incomplete via for West DC:

You can telnet to csr1000v-SP_C from the jumpbox @ 192.168.1.15 with root/default user/pass:

```

ubuntu@jumphost:~$ telnet 192.168.1.15
Trying 192.168.1.15...
Connected to 192.168.1.15.
Escape character is '^]'.

User Access Verification

Username: root
Password:
csr1000v-SP_C>

```

csr1000v-SP_C: Verify that csr1000v-SP_C is installing 10.99.99.0/24 in BGP table via East DC. Note the Best path is via AS 65002 988.

```

csr1000v-SP_C>sh ip bgp 10.99.99.0/24
BGP routing table entry for 10.99.99.0/24, version 16
Paths: (2 available, best #1, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65002 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external, best

```

```
rx pathid: 0, tx pathid: 0x0
Refresh Epoch 1
65001 65101, (aggregated by 65101 192.168.255.10)
172.16.99.3 from 172.16.99.3 (172.1.1.1)
Origin incomplete, localpref 100, valid, external, atomic-aggregate
rx pathid: 0, tx pathid: 0
```

Note: The BGP peering between E_CPE_A and SP_B leverages AS 988 as seen below. The following command replaces the local private AS Path with 988 for prefixes originated from East DC to the SP Cloud.

```
csr1000v-E_CPE_A#sh run | i 988
neighbor 172.16.6.3 local-as 988 no-prepend replace-as
csr1000v-E_CPE_A#
```

csr1000v-SP_C: Verify that csr1000v-SP_C is installing 10.99.99.0/24 in the ip routing table:

```
csr1000v-SP_C>sh ip route 10.99.99.0 255.255.255.0
Routing entry for 10.99.99.0/24
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:18:45 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:18:45 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
```

Note: From the jump host you can now try to reach the website via E_A_BIGIP and validate the path is installed via EAST DC.

Either open a web browser and browse to <http://10.99.99.102> or from the jumpbox CLI, type:

```
curl http://10.99.99.102
```

Jumpbox: Curl from the jumphost to the virtual server.

```
root@jumphost:~# curl 10.99.99.102
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.
↪</p>
</body></html>
```

Jumpbox: Traceroute from the jumphost to the virtual server to verify the path it is taking.

```
root@jumphost:~# traceroute 10.99.99.102

traceroute to 10.99.99.102 (10.99.99.102), 30 hops max, 60 byte packets
 1 192.168.1.15 (192.168.1.15)  7.202 ms  8.251 ms  8.049 ms
 2 172.16.99.4 (172.16.99.4)  22.485 ms  23.834 ms  36.059 ms
 3 172.16.6.4 (172.16.6.4)  40.575 ms  40.425 ms  62.741 ms
 4 10.99.99.102 (10.99.99.102)  64.284 ms  64.026 ms  91.206 ms
root@jumphost:~#
```

E_CPE_A_CSR1k: You can also validate from the CPE with telnet to 10.99.99.102 on port 80. Note that you can clear the telnet session by executing “clear line vty 0” on the console of the CPE:

```
csr1000v-E_CPE_A>telnet 10.99.99.102 80 /vrf internet
Trying 10.99.99.102, 80 ... Open
```

csr1000v-SP_C: You can also validate via traceroute to 10.99.99.102 on SP_C:

```
csr1000v-SP_C>traceroute 10.99.99.102
Type escape sequence to abort.
Tracing the route to 10.99.99.102
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.99.4 [AS 65001] 7 msec 5 msec 9 msec
  2 172.16.6.4 [AS 65002] 10 msec 10 msec 14 msec
  3 10.99.99.102 [AS 988] 13 msec 13 msec 15 msec
csr1000v-SP_C>
```

Note: Now lets move on and configure BGP on E_B_BIGIP....

Create an application configuration for a virtual server and a pool member on E_B_BIGIP-13:

E_B_BIGIP-13: Create the following virtual server and pool member on E_B_BIGIP-13:

```
tmsh create ltm pool pool1 members add { 10.3.99.200:80 } monitor tcp_half_open
tmsh create ltm virtual vip1 destination 10.99.99.102:80 source-address-translation {
↵type automap } pool pool1 profiles add { tcp http }

tmsh save sys config partitions all
```

E_B_BIGIP-13: Your virtual server should now show available on E_B_BIGIP-13:

```
[root@E_B_BIGIP-13:Active:Standalone] config # tmsh show ltm virtual

-----
Ltm::Virtual Server: vip1
-----

Status
Availability      : available
State             : enabled
Reason            : The virtual server is available
CMP               : enabled
CMP Mode          : all-cpus
Destination       : 10.99.99.102:80
```

Configure the route advertisement on the E_B_BIGIP-13:

E_B_BIGIP-13: Configure the eBGP session on E_B_BIGIP-13 to East CPE_A. The CPE configuration is already done for you so you only need to configure the BIG-IP side of session.

```
[root@E_B_BIGIP-13:Active:In Sync] config # imish -r 0
E_B_BIGIP-13.local[0]>enable
E_B_BIGIP-13.local[0]#config t
E_B_BIGIP-13.local[0] (config)#router bgp 65203
```

```

E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.30.4 remote-as 65201
E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.30.4 description E_CPE_A
E_B_BIGIP-13.local[0] (config-router)# neighbor 10.2.30.4 timers 3 9
E_A_BIGIP-13.local[0] (config-router)# end
E_A_BIGIP-13.local[0]#write mem

```

E_B_BIGIP-13: Verify eBGP adjacencies are up between E_B_BIGIP-13 and the East CPE router - E_CPE_A_CSR1k.

```

E_B_BIGIP-13.local[0]#sh ip bgp sum
BGP router identifier 10.2.50.3, local AS number 65203
BGP table version is 6
12 BGP AS-PATH entries
0 BGP community entries

Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ  OutQ Up/Down  State/PfxRcd
10.2.30.4      4 65201    17     14       5    0    0 00:00:18         9
10.2.50.4      4 65205   385    350       6    0    0 02:52:20        10

Total number of neighbors 2
E_B_BIGIP-13.local[0]#

```

E_B_BIGIP-13: On E_B_BIGIP configure the following network statement for 10.99.99.0/24 such that prefix is originated locally:

```

[root@E_B_BIGIP-13:Active:Standalone] config # imish -r 0
E_B_BIGIP-13.local[0]>en
E_B_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
E_B_BIGIP-13.local[0] (config)#router bgp 65203
E_B_BIGIP-13.local[0] (config-router)#network 10.99.99.0/24
E_B_BIGIP-13.local[0] (config-router)#end
E_B_BIGIP-13.local[0]#wr
Building configuration...
[OK]
E_B_BIGIP-13.local[0]#

```

E_B_BIGIP-13: On E_B_BIGIP verify 10.99.99.0/24 is being locally originated which can be seen with "Local":

```

E_B_BIGIP-13.local[0]#sh ip bgp 10.99.99.0/24 | b Local

...skipping
Local
0.0.0.0 from 0.0.0.0 (10.2.50.3)
  Origin IGP, localpref 100, weight 32768, valid, sourced, local, best
  Last update: Mon Jul 16 19:29:34 2018

65201 65202
10.2.30.4 from 10.2.30.4 (2.2.2.2)
  Origin IGP metric 0, localpref 100, valid, external
  Last update: Mon Jul 16 19:02:03 2018

65205 65202
10.2.50.4 from 10.2.50.4 (3.3.3.3)
  Origin IGP metric 0, localpref 100, valid, external
  Last update: Mon Jul 16 19:02:03 2018

```

E_B_BIGIP-13: On E_B_BIGIP verify 10.99.99.0/24 is being advertised outbound to East CPE device via E_CPE_A_CSR1k:

```
E_B_BIGIP-13.local[0]#sh ip bgp nei 10.2.30.4 advertised-routes | i 10.99.99.0/24
*> 10.99.99.0/24      10.2.30.3                100          32768 i
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is learning the 10.99.99.0/24 inbound from E_B_BIGIP:

Note: You can telnet to the CPE devices using the BGP neighbor IP address from Zebos using root/default for user/pass:

Example telnet from E_B_BIGIP-13 to East CPE Device @ 10.2.30.4:

```
E_B_BIGIP-13.local[0]#telnet 10.2.30.4
Trying 10.2.30.4...
Connected to 10.2.30.4.
Escape character is '^]'.

User Access Verification

Username: root
Password:
csr1000v-E_CPE_A>
```

E_CPE_A_CSR1k: Continued... Verify that E_CPE_A_CSR1k is learning the 10.99.99.0/24 inbound from E_B_BIGIP:

```
csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet neighbors 10.2.30.3 routes
BGP table version is 28, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop           Metric LocPrf Weight Path
Route Distinguisher: 65201:1000 (default for vrf internet)
*m  3.3.3.3/32         10.2.30.3          4294967295           0 65203 65205 i
*m  10.99.99.0/24      10.2.30.3          4294967295           0 65203 i

Total number of prefixes 2
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is installing 10.99.99.0/24 from E_B_BIGIP using ip route command. Notice the next hop of E_A_BIGIP @ 10.2.20.3 & E_B_BIGIP @ 10.2.30.3:

```
csr1000v-E_CPE_A>sh ip route vrf internet

Routing Table: internet
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from Pfr
```

```
Gateway of last resort is not set
```

```
1.0.0.0/32 is subnetted, 2 subnets
B    1.1.1.1 [20/4294967294] via 172.16.6.3, 23:39:09
B    1.1.1.2 [20/4294967294] via 172.16.6.3, 23:39:11
3.0.0.0/32 is subnetted, 1 subnets
B    3.3.3.3 [20/4294967294] via 10.2.30.3, 00:37:06
      [20/4294967294] via 10.2.20.3, 00:37:06
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C    10.2.20.0/24 is directly connected, GigabitEthernet2
L    10.2.20.4/32 is directly connected, GigabitEthernet2
C    10.2.30.0/24 is directly connected, GigabitEthernet3
L    10.2.30.4/32 is directly connected, GigabitEthernet3
B    10.99.99.0/24 [20/4294967294] via 10.2.30.3, 00:34:33
      [20/4294967294] via 10.2.20.3, 00:34:33
99.0.0.0/24 is subnetted, 1 subnets
B    99.99.99.0 [20/4294967294] via 172.16.6.3, 23:40:38
172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
B    172.16.1.0/24 [20/4294967294] via 172.16.6.3, 23:39:19
B    172.16.2.0/24 [20/4294967294] via 172.16.6.3, 23:39:19
C    172.16.6.0/24 is directly connected, GigabitEthernet5
L    172.16.6.4/32 is directly connected, GigabitEthernet5
B    172.16.99.0/24 [20/0] via 172.16.6.3, 23:40:38
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is now installing specific 10.99.99.0/24 from E_B_BIGIP using specific ip route command. Notice the next hop of E_A_BIGIP @ 10.2.20.3 & E_B_BIGIP @ 10.2.30.3:

```
csr1000v-E_CPE_A>sh ip route vrf internet 10.99.99.0 255.255.255.0

Routing Table: internet
Routing entry for 10.99.99.0/24
Known via "bgp 65201", distance 20, metric 4294967294
Tag 65202, type external
Last update from 10.2.20.3 00:39:40 ago
Routing Descriptor Blocks:
* 10.2.30.3, from 10.2.30.3, 00:39:40 ago
  Route metric is 4294967294, traffic share count is 1
  AS Hops 1
  Route tag 65202
  MPLS label: none
  10.2.20.3, from 10.2.20.3, 00:39:40 ago
  Route metric is 4294967294, traffic share count is 1
  AS Hops 1
  Route tag 65202
  MPLS label: none
csr1000v-E_CPE_A>
```

Note: Congratulations! You now have eBGP Multipath Loadsharing working within the East DC! As seen above, this will trigger ECMP for 10.99.99.0/24 on E_CPE_A_CSR1k towards E_A_BIGIP and E_B_BIGIP. Note that normally the weight, local preference, AS path length, origin, med, etc. would need to be the same for the parallel routes to be installed in the routing table.

It is worth noting that this behavior varies from version to version of IOS. In this lab, we are using IOS-XE Version 16.3.6. With this version, the entire AS path needs to be the same for multipath condition to be met.

How did we work around this? The following hidden command is used on the CPE to ignore the different AS Path and install the route as multipath if all other conditions are met:

```
csr1000v-E_CPE_A#sh run | i as-path
bgp bestpath as-path multipath-relax
csr1000v-E_CPE_A#
```

csr1000v-SP_C: Verify that nothing changed on csr1000v-SP_C and it is still installing 10.99.99.0/24 via East DC because Origin attribute is IGP versus incomplete for West DC:

csr1000v-SP_C: Verify that csr1000v-SP_C is installing 10.99.99.0/24 in BGP table via East DC. Recall the Best path leveraging EAST DC is via AS 65002 988.

```
csr1000v-SP_C>sh ip bgp 10.99.99.0/24
BGP routing table entry for 10.99.99.0/24, version 16
Paths: (2 available, best #1, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65002 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external, best
    rx pathid: 0, tx pathid: 0x0
  Refresh Epoch 1
  65001 65101, (aggregated by 65101 192.168.255.10)
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin incomplete, localpref 100, valid, external, atomic-aggregate
    rx pathid: 0, tx pathid: 0
```

csr1000v-SP_C: Verify that csr1000v-SP_C is installing 10.99.99.0/24 in the ip routing table:

```
csr1000v-SP_C>sh ip route 10.99.99.0 255.255.255.0
Routing entry for 10.99.99.0/24
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:18:45 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:18:45 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
```

Note: As seen above, all traffic from the Jumpbox via SP_C destined to 10.99.99.0/24 is currently via the East DC. This is because EAST DC wins the tiebreaker as the Origin attribute is IGP versus incomplete for West DC

At this moment, you can only curl to 10.99.99.102 VIP in the EAST DC via the jumpbox.

Place E_A_BIGIP into maintenance mode within the East DC by using BGP AS Path Prepending:

E_A_BIGIP-13: Create AS-Path-Prepend-OUT route-map on E_A_BIGIP for 10.99.99.0/24 to insert 1 AS Path prepend into the prefix:

```

[root@E_A_BIGIP-13:Active:Standalone] config # imish -r 0
E_A_BIGIP-13.local[0]>en
E_A_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
E_A_BIGIP-13.local[0] (config)#
E_A_BIGIP-13.local[0] (config)#ip prefix-list as-path-prepend-prefix seq 10 permit 10.
↪99.99.0/24
E_A_BIGIP-13.local[0] (config)#
E_A_BIGIP-13.local[0] (config)#route-map AS-Path-Prepend-OUT permit 100
E_A_BIGIP-13.local[0] (config-route-map)# match ip address prefix-list as-path-prepend-
↪prefix
E_A_BIGIP-13.local[0] (config-route-map)# set as-path prepend 988
E_A_BIGIP-13.local[0] (config-route-map)#!
E_A_BIGIP-13.local[0] (config-route-map)#route-map AS-Path-Prepend-OUT permit 200
E_A_BIGIP-13.local[0] (config-route-map)#!
E_A_BIGIP-13.local[0] (config-route-map)#router bgp 65202
E_A_BIGIP-13.local[0] (config-router)#nei 10.2.20.4 route-map AS-Path-Prepend-OUT out
E_A_BIGIP-13.local[0] (config-router)#end
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#clear ip bgp *

```

E_A_BIGIP-13: Verify AS-Path-Prepend-OUT has inserted 1 AS Path prepend into the prefix towards CPE @ 10.2.20.4:

```

E_A_BIGIP-13.local[0]#sh ip bgp nei 10.2.20.4 advertised-routes | i 10.99.99.0
*> 10.99.99.0/24      10.2.20.3              100      32768 988 i

```

E_CPE_A_CSR1k: Verify AS-Path-Prepending inbound on E_CPE_A for 10.99.99.0/24 from E_A_BIGIP:

```

csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet neighbors 10.2.20.3 routes
BGP table version is 56, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop           Metric LocPrf Weight Path
Route Distinguisher: 65201:1000 (default for vrf internet)
*>   3.3.3.3/32        10.2.20.3         4294967295           0 65202 65205 i
*    10.99.99.0/24    10.2.20.3         4294967295           0 65202 988 i

```

E_CPE_A_CSR1k: Verify that E_A_BIGIP is no longer an installed route or preferred in BGP RIB for 10.99.99.0/24 on E_CPE_A. You will Note that the next hop for 10.99.99.0/24 is E_B_BIGIP @ 10.2.30.3 and not E_A_BIGIP @ 10.2.20.3.

```

csr1000v-E_CPE_A>sh ip route vrf internet

Routing Table: internet
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route

```

```
+ - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

  1.0.0.0/32 is subnetted, 2 subnets
B       1.1.1.1 [20/4294967294] via 172.16.6.3, 1d01h
B       1.1.1.2 [20/4294967294] via 172.16.6.3, 1d01h
  3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [20/4294967294] via 10.2.30.3, 00:05:07
        [20/4294967294] via 10.2.20.3, 00:05:07
 10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.2.20.0/24 is directly connected, GigabitEthernet2
L       10.2.20.4/32 is directly connected, GigabitEthernet2
C       10.2.30.0/24 is directly connected, GigabitEthernet3
L       10.2.30.4/32 is directly connected, GigabitEthernet3
B       10.99.99.0/24 [20/4294967294] via 10.2.30.3, 00:05:11
 99.0.0.0/24 is subnetted, 1 subnets
B       99.99.99.0 [20/4294967294] via 172.16.6.3, 1d01h
172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
B       172.16.1.0/24 [20/4294967294] via 172.16.6.3, 1d01h
B       172.16.2.0/24 [20/4294967294] via 172.16.6.3, 1d01h
C       172.16.6.0/24 is directly connected, GigabitEthernet5
L       172.16.6.4/32 is directly connected, GigabitEthernet5
B       172.16.99.0/24 [20/0] via 172.16.6.3, 1d01h
csr1000v-E_CPE_A>
```

Note: Congratulations! E_A_BIGIP has successfully been placed in maintenance mode within the East DC and is no longer taking any traffic. This was achieved by inserting the additional AS Path prepend in the previous step eliminating this as a candidate for BGP multipath selection on E_CPE_A. Let's continue with additional validation.

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is now installing specific 10.99.99.0/24 from E_B_BIGIP using specific ip route command. Notice the next hop of E_B_BIGIP @ 10.2.30.3:

```
csr1000v-E_CPE_A>sh ip route vrf internet 10.99.99.0 255.255.255.0

Routing Table: internet
Routing entry for 10.99.99.0/24
  Known via "bgp 65201", distance 20, metric 4294967294
  Tag 65203, type external
  Last update from 10.2.30.3 00:00:24 ago
  Routing Descriptor Blocks:
  * 10.2.30.3, from 10.2.30.3, 00:00:24 ago
    Route metric is 4294967294, traffic share count is 1
    AS Hops 1
    Route tag 65203
    MPLS label: none
```

E_CPE_A_CSR1k: Verify that E_CPE_A_CSR1k is now installing specific 10.99.99.0/24 from E_B_BIGIP using specific ip bgp command. Notice the best path is via E_B_BIGIP @ 10.2.30.3 due to AS Path length:

```
csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet 10.99.99.0

BGP routing table entry for 65201:1000:10.99.99.0/24, version 53
BGP Bestpath: deterministic-med: aigp-ignore: med
Paths: (2 available, best #1, table internet)
```

```

Multipath: eiBGP
  Advertised to update-groups:
    3          4
  Refresh Epoch 1
  65203
  10.2.30.3 (via vrf internet) from 10.2.30.3 (10.2.50.3)
    Origin IGP, metric 4294967295, localpref 100, valid, external, best
    Extended Community: RT:65201:1000
    rx pathid: 0, tx pathid: 0x0
  Refresh Epoch 1
  65202 988
  10.2.20.3 (via vrf internet) from 10.2.20.3 (10.2.40.3)
    Origin IGP, metric 4294967295, localpref 100, valid, external
    Extended Community: RT:65201:1000
    rx pathid: 0, tx pathid: 0

```

csr1000v-SP_C: Verify path via Virtual Server 10.99.99.102 is still up via East DC @ E_B_BIGIP now that E_A_BIGIP is in maintenance mode within East DC:

```

csr1000v-SP_C>traceroute 10.99.99.102
Type escape sequence to abort.
Tracing the route to 10.99.99.102
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.99.4 [AS 65001] 7 msec 7 msec 8 msec
  2 172.16.6.4 [AS 65002] 12 msec 14 msec 14 msec
  3 10.99.99.102 [AS 988] 18 msec 12 msec 13 msec
csr1000v-SP_C>

```

Reminder: You can telnet to csr1000v-SP_C from the jumpbox @ 192.168.1.15 with root/default user/pass:

```

ubuntu@jumphost:~$ telnet 192.168.1.15
Trying 192.168.1.15...
Connected to 192.168.1.15.
Escape character is '^]'.

User Access Verification

Username: root
Password:
csr1000v-SP_C>

```

Jumpbox: Verify curl to Virtual Server 10.99.99.102 is still up via East DC @ E_B_BIGIP:

```

root@jumphost:~# curl 10.99.99.102
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>

```

Jumpbox: Verify traceroute to Virtual Server 10.99.99.102 is still up via East DC @ E_B_BIGIP:

```

root@jumphost:~# traceroute 10.99.99.102
traceroute to 10.99.99.102 (10.99.99.102), 30 hops max, 60 byte packets
 1 192.168.1.15 (192.168.1.15) 13.403 ms 13.047 ms 12.418 ms
 2 172.16.99.4 (172.16.99.4) 12.830 ms 12.649 ms 12.351 ms
 3 172.16.6.4 (172.16.6.4) 31.121 ms 44.958 ms 44.866 ms

```

```
4 10.99.99.102 (10.99.99.102) 44.458 ms 45.634 ms 60.454 ms
root@jumpphost:~#
```

Note: Now that E_A_BIGIP is in maintenance mode we only have E_B_BIGIP taking all the traffic within the East DC for Virtual Servers on 10.99.99.0/24 via SP_C.

Let's also match AS Path Prepending on E_B_BIGIP such that both East BIG IP's have been added to maintenance mode and are no longer taking any traffic via SP_C.

This is because the AS Path will be longer via East DC as compared to West DC after we make the next set of changes.

Create AS-Path-Prepend-OUT route-map on E_B_BIGIP for 10.99.99.0/24 to insert 1 AS Path prepend into the prefix:

E_B_BIGIP-13: Create AS-Path-Prepend-OUT route-map on E_B_BIGIP for 10.99.99.0/24 to insert 1 AS Path prepend into the prefix:

```
[root@E_B_BIGIP-13:Active:Standalone] config # imish -r 0
E_B_BIGIP-13.local[0]>en
E_B_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line. End with CNTL/Z.
E_B_BIGIP-13.local[0] (config)#ip prefix-list as-path-prepend-prefix seq 10 permit 10.
↪99.99.0/24
E_B_BIGIP-13.local[0] (config)#
E_B_BIGIP-13.local[0] (config)#route-map AS-Path-Prepend-OUT permit 100
E_B_BIGIP-13.local[0] (config-route-map)# match ip address prefix-list as-path-prepend-
↪prefix
E_B_BIGIP-13.local[0] (config-route-map)# set as-path prepend 988
E_B_BIGIP-13.local[0] (config-route-map)#route-map AS-Path-Prepend-OUT permit 200
E_B_BIGIP-13.local[0] (config-route-map)#router bgp 65203
E_B_BIGIP-13.local[0] (config-router)#neighbor 10.2.30.4 route-map AS-Path-Prepend-OUT_
↪out
E_B_BIGIP-13.local[0] (config-router)#end
E_B_BIGIP-13.local[0]#wr
E_B_BIGIP-13.local[0]#clear ip bgp *
```

E_B_BIGIP-13: Verify AS-Path-Prepend-OUT has inserted 1 AS Path prepend into the prefix towards CPE @ 10.2.30.4:

```
E_B_BIGIP-13.local[0]#sh ip bgp nei 10.2.30.4 advertised-routes | i 10.99.99.0/24
*> 10.99.99.0/24 10.2.30.3 100 32768 988 i
```

E_CPE_A_CSR1k: Verify AS-Path-Prepending inbound on E_CPE_A for 10.99.99.0/24 from E_B_BIGIP.

```
csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet neighbors 10.2.30.3 routes
BGP table version is 71, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

Network          Next Hop          Metric LocPrf Weight Path
```

```

Route Distinguisher: 65201:1000 (default for vrf internet)
 *m  3.3.3.3/32      10.2.30.3      4294967295      0 65203 65205 i
 *m  10.99.99.0/24   10.2.30.3      4294967295      0 65203 988 i

Total number of prefixes 2

```

Note: You will also notice the 'm' notation has been restored above. This means the prefixes are selected for multipath since we have equalized the previous AS Path prepend configured on E_A_BIGIP.

Let's move along...

E_CPE_A_CSR1k: Verify that both E_A_BIGIP & E_B_BIGIP is now valid again for 10.99.99.0/24 on E_CPE_A. You will Note that the next hop for 10.99.99.0/24 is both E_A_BIGIP @ 10.2.20.3 and E_B_BIGIP @ 10.2.30.3

```

csr1000v-E_CPE_A>sh ip route vrf internet

Routing Table: internet
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

 1.0.0.0/32 is subnetted, 2 subnets
B       1.1.1.1 [20/4294967294] via 172.16.6.3, 1d01h
B       1.1.1.2 [20/4294967294] via 172.16.6.3, 1d01h
 3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [20/4294967294] via 10.2.30.3, 00:08:00
        [20/4294967294] via 10.2.20.3, 00:08:00
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.2.20.0/24 is directly connected, GigabitEthernet2
L       10.2.20.4/32 is directly connected, GigabitEthernet2
C       10.2.30.0/24 is directly connected, GigabitEthernet3
L       10.2.30.4/32 is directly connected, GigabitEthernet3
B       10.99.99.0/24 [20/4294967294] via 10.2.30.3, 00:07:57
        [20/4294967294] via 10.2.20.3, 00:07:57
 99.0.0.0/24 is subnetted, 1 subnets
B       99.99.99.0 [20/4294967294] via 172.16.6.3, 1d01h
172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
B       172.16.1.0/24 [20/4294967294] via 172.16.6.3, 1d01h
B       172.16.2.0/24 [20/4294967294] via 172.16.6.3, 1d01h
C       172.16.6.0/24 is directly connected, GigabitEthernet5
L       172.16.6.4/32 is directly connected, GigabitEthernet5
B       172.16.99.0/24 [20/0] via 172.16.6.3, 1d01h
csr1000v-E_CPE_A>

```

Note: Congratulations! E_B_BIGIP has successfully been placed in maintenance mode within the East DC and is no longer taking any traffic. This was achieved by inserting the additional AS Path prepend in the

previous step eliminating this as a candidate for BGP multipath selection on E_CPE_A.

Now the entire East DC is in maintenance mode as both E_A_BIGIP and E_B_BIGIP are no longer taking traffic.

That is, 10.99.99.0/24 is preferred via the West DC from the jumpbox when leveraging SP_C. Let's continue with additional validation.

E_CPE_A_CSR1k: We can observe that prepending is happening for 10.99.99.0/24 on E_CPE_A for both E_A_BIGIP & E_B_BIGIP:

```
csr1000v-E_CPE_A>show ip bgp vpnv4 vrf internet | i 988
*m  10.99.99.0/24    10.2.30.3          4294967295          0 65203 988 i
*> 10.2.20.3        4294967295          0 65202 988 i
```

csr1000v-SP_C: Verify 10.99.99.0/24 is available on SP_C BGP RIB table via East DC. You will notice the best path is via West DC via AS 65101.

Reminder: You can telnet to csr1000v-SP_C from the jumpbox @ 192.168.1.15 with root/default user/pass:

```
csr1000v-SP_C>sh ip bgp 10.99.99.100/24
BGP routing table entry for 10.99.99.0/24, version 25
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65002 988 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65001 65101, (aggregated by 65101 192.168.255.10)
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin incomplete, localpref 100, valid, external, atomic-aggregate, best
    rx pathid: 0, tx pathid: 0x0
```

csr1000v-SP_C: Verify 10.99.99.0/24 is no longer installed on SP_C IP routing table via East DC. You will notice the route installed is via West DC via AS 65001.

```
csr1000v-SP_C>sh ip route 10.99.99.100
Routing entry for 10.99.99.0/24
  Known via "bgp 65003", distance 20, metric 0
  Tag 65001, type external
  Last update from 172.16.99.3 00:07:29 ago
  Routing Descriptor Blocks:
  * 172.16.99.3, from 172.16.99.3, 00:07:29 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65001
    MPLS label: none
```

Note: This prefix is no longer installed in the routing table via East DC because the AS Path length is larger than that of West DC. At this point traffic is now via West DC for 10.99.99.0/24 from SP_C point-of-view.

csr1000v-SP_C: Verify path via Virtual Server 10.99.99.101 is now via West DC - AS 65101. Note that below output may not be an exact match as this can be via either 172.16.1.4 or 172.16.2.4 leveraging AS 65101 via West_CPE_A or West_CPE_B.

```

csr1000v-SP_C>traceroute 10.99.99.101
Type escape sequence to abort.
Tracing the route to 10.99.99.101
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.99.3 [AS 65001]  8 msec 8 msec 7 msec
  2 172.16.1.4 [AS 65001] 11 msec 8 msec 10 msec
  3 10.99.99.101 [AS 65101] 14 msec 13 msec 15 msec
csr1000v-SP_C>

```

Jumpbox: Verify curl to Virtual Server 10.99.99.101 is up via West DC:

```

root@jumphost:~# curl 10.99.99.101
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
root@jumphost:~#

```

Jumpbox: Verify traceroute to Virtual Server 10.99.99.101 is West DC. Note that below output may not be an exact match as this can be via either 172.16.1.4 or 172.16.2.4 leveraging AS 65101 via West_CPE_A or West_CPE_B.

```

root@jumphost:~# traceroute 10.99.99.101
traceroute to 10.99.99.101 (10.99.99.101), 30 hops max, 60 byte packets
 1 192.168.1.15 (192.168.1.15)  2.504 ms  15.470 ms  15.277 ms
 2 172.16.99.3 (172.16.99.3)    18.709 ms 19.369 ms 19.762 ms
 3 172.16.2.4 (172.16.2.4)    25.569 ms 28.738 ms 44.922 ms
 4 10.99.99.101 (10.99.99.101) 44.591 ms 47.980 ms 51.598 ms
root@jumphost:~#

```

Anycast DC Failover section - Swing Traffic back to East DC by adding 2 x /25 specific routes which comprise of the overall 10.99.99.0 /24

Note: In previous section we verified that 10.99.99.0/24 is only installed in the IP Routing table of SP_C via West DC. However, East DC is available as backup path in BGP RIB.

In this section we will swing traffic back to East DC by utilizing 2 x /25's.

E_A_BIGIP-13: Configure BGP on E_A_BIGIP-13 to originate 10.99.99.0 /25 and 10.99.99.128 /25:

```

[root@E_A_BIGIP-13:Active:Standalone] config # imish -r 0
E_A_BIGIP-13.local[0]>en
E_A_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line. End with CNTL/Z.
E_A_BIGIP-13.local[0] (config)#router bgp 65202
E_A_BIGIP-13.local[0] (config-router)#network 10.99.99.0/25
E_A_BIGIP-13.local[0] (config-router)#network 10.99.99.128/25
E_A_BIGIP-13.local[0] (config)#end
E_A_BIGIP-13.local[0]#clear ip bgp *
E_A_BIGIP-13.local[0]#wr
Building configuration...

```

E_A_BIGIP-13: Verify 10.99.99.0/24, 10.99.99.0/25, and 10.99.99.128/25 are advertised via E_A_BIGIP to E_CPE_A @ 10.2.20.4:

```
E_A_BIGIP-13.local[0]#sh ip bgp nei 10.2.20.4 ad
BGP table version is 2, local router ID is 10.2.40.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 3.3.3.3/32	10.2.20.3			0	65205 i
*> 10.3.99.0/24	10.2.20.3			0	65205 i
*> 10.99.99.0/24	10.2.20.3		100	32768	988 i
*> 10.99.99.0/25	10.2.20.3		100	32768	i
*> 10.99.99.128/25	10.2.20.3		100	32768	i

Total number of prefixes 5

csr1000v-SP_C: Verify 10.99.99.0/25 is available on SP_C BGP RIB table via East DC leveraging 10.99.99.0 /25. You will notice the best path is via East DC via AS 65002 988.

Reminder: You can telnet to csr1000v-SP_C from the jumpbox @ 192.168.1.15 with root/default user/pass.

```
csr1000v-SP_C>sh ip bgp 10.99.99.102
BGP routing table entry for 10.99.99.0/25, version 30
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
    65001 65002 988
      172.16.99.3 from 172.16.99.3 (172.1.1.1)
        Origin IGP, localpref 100, valid, external
        rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
    65002 988
      172.16.99.4 from 172.16.99.4 (172.1.1.2)
        Origin IGP, localpref 100, valid, external, best
        rx pathid: 0, tx pathid: 0x0
```

csr1000v-SP_C: Let's also verify 10.99.99.0/25

```
csr1000v-SP_C>sh ip bgp 10.99.99.0/25
BGP routing table entry for 10.99.99.0/25, version 26
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
    65001 65002 988
      172.16.99.3 from 172.16.99.3 (172.1.1.1)
        Origin IGP, localpref 100, valid, external
        rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
    65002 988
      172.16.99.4 from 172.16.99.4 (172.1.1.2)
        Origin IGP, localpref 100, valid, external, best
        rx pathid: 0, tx pathid: 0x0
```

csr1000v-SP_C: Verify 10.99.99.102 is installed on SP_C IP routing table via East DC leveraging 10.99.99.0/25. You will notice the route installed is via East DC via AS 65002.

```
csr1000v-SP_C>sh ip route 10.99.99.102
Routing entry for 10.99.99.0/25
```

```
Known via "bgp 65003", distance 20, metric 0
Tag 65002, type external
Last update from 172.16.99.4 00:03:10 ago
Routing Descriptor Blocks:
* 172.16.99.4, from 172.16.99.4, 00:03:10 ago
  Route metric is 0, traffic share count is 1
  AS Hops 2
  Route tag 65002
  MPLS label: none
```

Note: You will observe that the IP Routing table of SP_C will prefer the path via East DC for the 10.99.99.102 Virtual Server due longest match of 10.99.99.0/25 even though 10.99.99.0/24 is via West DC.

We can expect the same behavior with 10.99.99.128 /25. Let's validate.

csr1000v-SP_C: Verify 10.99.99.128 /25 is available on SP_C IP BGP RIB table via East DC. You will notice the best path is via East DC via AS 65002 988.

```
csr1000v-SP_C>sh ip bgp 10.99.99.128/25
BGP routing table entry for 10.99.99.128/25, version 39
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65001 65002 988
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65002 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external, best
    rx pathid: 0, tx pathid: 0x0
```

csr1000v-SP_C: Verify 10.99.99.128 /25 is installed on SP_C IP routing table via East DC leveraging 10.99.99.128 /25. You will notice the route installed is via East DC via AS 65002.

```
csr1000v-SP_C>sh ip route 10.99.99.128 255.255.255.128
Routing entry for 10.99.99.128/25
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:03:16 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:03:16 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
```

Note: What observations are made with 10.99.99.0/24? You will notice this remains the same with West DC preferred via AS Path length for 10.99.99.0/24.

csr1000v-SP_C: Verify 10.99.99.0 /24 is available on SP_C BGP RIB table via West DC. You will notice the

best path is via West DC via AS 65101.

```
csr1000v-SP_C>sh ip bgp 10.99.99.0/24
BGP routing table entry for 10.99.99.0/24, version 25
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65002 988 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65001 65101, (aggregated by 65101 192.168.255.10)
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin incomplete, localpref 100, valid, external, atomic-aggregate, best
    rx pathid: 0, tx pathid: 0x0
```

csr1000v-SP_C: Verify 10.99.99.0/24 is available on SP_C IP routing table via West DC. You will notice the best path is via West DC via AS 65001.

```
csr1000v-SP_C>sh ip route 10.99.99.0 255.255.255.0
Routing entry for 10.99.99.0/24
  Known via "bgp 65003", distance 20, metric 0
  Tag 65001, type external
  Last update from 172.16.99.3 01:00:50 ago
  Routing Descriptor Blocks:
  * 172.16.99.3, from 172.16.99.3, 01:00:50 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65001
    MPLS label: none
```

csr1000v-SP_C: Verify path via Virtual Server 10.99.99.102 is now via East DC due to the /25's being originated from East DC via E_A_BIGIP-13 .

```
csr1000v-SP_C>traceroute 10.99.99.102
Type escape sequence to abort.
Tracing the route to 10.99.99.102
VRF info: (vrf in name/id, vrf out name/id)
 1 172.16.99.4 [AS 65001] 7 msec 6 msec 8 msec
 2 172.16.6.4 [AS 65002] 9 msec 12 msec 10 msec
 3 10.99.99.102 [AS 988] 17 msec 21 msec 16 msec
csr1000v-SP_C>
```

Jumpbox: Verify curl to Virtual Server 10.99.99.102 is up via East DC:

```
root@jumphost:~# curl 10.99.99.102
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
root@jumphost:~#
```

Jumpbox: Verify traceroute to Virtual Server 10.99.99.102 is East DC.

```
root@jumphost:~# traceroute 10.99.99.102
traceroute to 10.99.99.102 (10.99.99.102), 30 hops max, 60 byte packets
```

```

1 192.168.1.15 (192.168.1.15) 14.200 ms 14.073 ms 13.654 ms
2 172.16.99.4 (172.16.99.4) 21.305 ms 27.179 ms 27.071 ms
3 172.16.6.4 (172.16.6.4) 26.755 ms 26.447 ms 26.119 ms
4 10.99.99.102 (10.99.99.102) 36.549 ms 48.875 ms 48.795 ms
root@jumpghost:~#

```

Note: Congratulations! You have successfully swing traffic back to the East DC even though 10.99.99.0/24 is currently preferred via West DC from SP_C. This was accomplished by introducing specific /25's from the East DC via E_A_BIGIP-13.

Let's finish this off as you are in the home stretch! We will finish the East DC by originating the same /25's on E_B_BIGIP-13 for consistency with E_A_BIGIP-13.

Re-introduce E_B_BIGIP-13 in the East DC via the /25's:

In this section we will finish the configuration on the East DC to originate the 2 x /25's on E_B_BIGIP-13. This will match the origination of 2 x /25's previously completed on E_A_BIGIP-13.

E_B_BIGIP-13: Configure BGP on E_B_BIGIP-13 to originate 10.99.99.0 /25 and 10.99.99.128 /25:

```

[root@E_B_BIGIP-13:Active:Standalone] config # imish -r 0
E_B_BIGIP-13.local[0]>enE_B_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line. End with CNTL/Z.
E_B_BIGIP-13.local[0](config)#router bgp 65203
E_B_BIGIP-13.local[0](config-router)#network 10.99.99.0/25
E_B_BIGIP-13.local[0](config-router)#network 10.99.99.128/25
E_B_BIGIP-13.local[0](config)#end
E_B_BIGIP-13.local[0]#clear ip bgp *
E_B_BIGIP-13.local[0]#wr
Building configuration...

```

E_B_BIGIP-13: Verify 10.99.99.0/24, 10.99.99.0/25, and 10.99.99.128/25 are advertised via E_B_BIGIP to E_CPE_A @ 10.2.30.4:

```

E_B_BIGIP-13.local[0]#sh ip bgp nei 10.2.30.4 advertised-routes
BGP table version is 9, local router ID is 10.2.50.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric      LocPrf      Weight Path
*> 3.3.3.3/32       10.2.30.3                0 65205 i
*> 10.3.99.0/24     10.2.30.3                0 65205 i
*> 10.99.99.0/24    10.2.30.3              100      32768 988 i
*> 10.99.99.0/25    10.2.30.3              100      32768 i
*> 10.99.99.128/25 10.2.30.3              100      32768 i

```

E_CPE_A_CSR1k: Verify that 10.99.99.0/24, 10.99.99.0/24, and 10.99.99.128/25 are learned via the routing table on E_CPE_A from both E_A_BIGIP @ 10.2.20.3 and E_B_BIGIP @ 10.2.30.3

```

csr1000v-E_CPE_A>sh ip route vrf internet

Routing Table: internet
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

```

```

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override, p - overrides from PFR

```

Gateway of last resort is not set

```

1.0.0.0/32 is subnetted, 2 subnets
B 1.1.1.1 [20/4294967294] via 172.16.6.3, 1d02h
B 1.1.1.2 [20/4294967294] via 172.16.6.3, 1d02h
3.0.0.0/32 is subnetted, 1 subnets
B 3.3.3.3 [20/4294967294] via 10.2.30.3, 00:04:59
[20/4294967294] via 10.2.20.3, 00:04:59
10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
C 10.2.20.0/24 is directly connected, GigabitEthernet2
L 10.2.20.4/32 is directly connected, GigabitEthernet2
C 10.2.30.0/24 is directly connected, GigabitEthernet3
L 10.2.30.4/32 is directly connected, GigabitEthernet3
B 10.99.99.0/24 [20/4294967294] via 10.2.30.3, 00:05:20
[20/4294967294] via 10.2.20.3, 00:05:20
B 10.99.99.0/25 [20/4294967294] via 10.2.30.3, 00:05:28
[20/4294967294] via 10.2.20.3, 00:05:28
B 10.99.99.128/25 [20/4294967294] via 10.2.30.3, 00:05:28
[20/4294967294] via 10.2.20.3, 00:05:28
99.0.0.0/24 is subnetted, 1 subnets
B 99.99.99.0 [20/4294967294] via 172.16.6.3, 1d03h
172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
B 172.16.1.0/24 [20/4294967294] via 172.16.6.3, 1d02h
B 172.16.2.0/24 [20/4294967294] via 172.16.6.3, 1d02h
C 172.16.6.0/24 is directly connected, GigabitEthernet5
L 172.16.6.4/32 is directly connected, GigabitEthernet5
B 172.16.99.0/24 [20/0] via 172.16.6.3, 1d03h
csr1000v-E_CPE_A>

```

E_CPE_A_CSR1k: As an example, let's take a closer look the bgp table for 10.99.99.128/25 on E_CPE_A:

```

csr1000v-E_CPE_A>sh ip bgp vpnv4 vrf internet 10.99.99.128/25
BGP routing table entry for 65201:1000:10.99.99.128/25, version 98
BGP Bestpath: deterministic-med: aigp-ignore: med
Paths: (2 available, best #1, table internet)
Multipath: eiBGP
  Advertised to update-groups:
    3          4
  Refresh Epoch 1
  65202
  10.2.20.3 (via vrf internet) from 10.2.20.3 (10.2.40.3)
    Origin IGP, metric 4294967295, localpref 100, valid, external, multipath, best
    Extended Community: RT:65201:1000
    rx pathid: 0, tx pathid: 0x0
  Refresh Epoch 1
  65203
  10.2.30.3 (via vrf internet) from 10.2.30.3 (10.2.50.3)
    Origin IGP, metric 4294967295, localpref 100, valid, external,
↳multipath(oldest)
    Extended Community: RT:65201:1000

```

```
rx pathid: 0, tx pathid: 0
```

csr1000v-SP_C: Verify nothing changed w.r.t. 10.99.99.0/25 and 10.99.99.128/25 and are still in the IP Routing table of SP_C via East DC after adding the /25's on E_B_BIGIP-13.

First let's take a look at the BGP table and confirm nothing changed since we previously added the 2 x /25's on E_A_BIGIP-13 and now completed E_B_BIGIP-13.

First will be 10.99.99.0 /25:

```
csr1000v-SP_C>sh ip bgp 10.99.99.0 255.255.255.128
BGP routing table entry for 10.99.99.0/25, version 38
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65001 65002 988
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65002 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external, best
    rx pathid: 0, tx pathid: 0x0
```

Second will be 10.99.99.128 /25:

```
csr1000v-SP_C>sh ip bgp 10.99.99.128 255.255.255.128
BGP routing table entry for 10.99.99.128/25, version 39
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65001 65002 988
  172.16.99.3 from 172.16.99.3 (172.1.1.1)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65002 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external, best
    rx pathid: 0, tx pathid: 0x0
```

Last will be 10.99.99.0 /24:

```
csr1000v-SP_C>sh ip bgp 10.99.99.0 255.255.255.0
BGP routing table entry for 10.99.99.0/24, version 25
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    1
  Refresh Epoch 1
  65002 988 988
  172.16.99.4 from 172.16.99.4 (172.1.1.2)
    Origin IGP, localpref 100, valid, external
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65001 65101, (aggregated by 65101 192.168.255.10)
```

```
172.16.99.3 from 172.16.99.3 (172.1.1.1)
  Origin incomplete, localpref 100, valid, external, atomic-aggregate, best
  rx pathid: 0, tx pathid: 0x0
```

Note: You will also observe that the IP Routing table on SP_C will still prefer West DC for the 10.99.99.0/24 due to previous AS Path Prepending exercise inserted from the East DC towards SP.

csr1000v-SP_C: Now let's take a look at the ip routing table and confirm nothing has changed since we previously added the 2 x /25's on E_A_BIGIP-13 and now completed E_B_BIGIP-13.

First will be 10.99.99.0 /25

```
csr1000v-SP_C>sh ip route 10.99.99.0 255.255.255.128
Routing entry for 10.99.99.0/25
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:19:27 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:19:27 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
```

Second will be 10.99.99.128 /25

```
csr1000v-SP_C>sh ip route 10.99.99.128 255.255.255.128
Routing entry for 10.99.99.128/25
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:19:34 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:19:34 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
```

Last will be 10.99.99.0 /24

```
csr1000v-SP_C>sh ip route 10.99.99.0 255.255.255.0
Routing entry for 10.99.99.0/24
  Known via "bgp 65003", distance 20, metric 0
  Tag 65001, type external
  Last update from 172.16.99.3 01:00:50 ago
  Routing Descriptor Blocks:
  * 172.16.99.3, from 172.16.99.3, 01:00:50 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65001
    MPLS label: none
```

csr1000v-SP_C: Verify path via Virtual Server 10.99.99.102 is still via East DC with introduction of adding the /25's from East DC via both E_A_BIGIP-13 & E_B_BIGIP-13 .

```
csr1000v-SP_C>traceroute 10.99.99.102
Type escape sequence to abort.
Tracing the route to 10.99.99.102
VRF info: (vrf in name/id, vrf out name/id)
  1 172.16.99.4 [AS 65001] 7 msec 7 msec 8 msec
  2 172.16.6.4 [AS 65002] 23 msec 11 msec 11 msec
  3 10.99.99.102 [AS 988] 14 msec 16 msec 14 msec
csr1000v-SP_C>
```

Jumpbox: Verify curl to Virtual Server 10.99.99.102 is up via East DC:

```
root@jumphost:~# curl 10.99.99.102
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Jumpbox: Verify traceroute to Virtual Server 10.99.99.102 is East DC.

```
root@jumphost:~# traceroute 10.99.99.102
traceroute to 10.99.99.102 (10.99.99.102), 30 hops max, 60 byte packets
 1 192.168.1.15 (192.168.1.15) 23.599 ms 21.587 ms 20.725 ms
 2 172.16.99.4 (172.16.99.4) 25.015 ms 24.031 ms 23.148 ms
 3 172.16.6.4 (172.16.6.4) 34.033 ms 33.082 ms 38.138 ms
 4 10.99.99.102 (10.99.99.102) 37.173 ms 36.389 ms 53.688 ms
```

Note: Congratulations! This section is now complete and everything checks out as expected. We are now going to move on to the final step and validate with a Virtual Server on the upper /25 – 10.99.99.128 /25.

Create an application configuration for a virtual server and a pool member on E_A_BIGIP-13 and E_A_BIGIP-13 to validate reachability via 10.99.99.128/25:

Create the following virtual server and pool member on both E_A_BIGIP-13 and E_B_BIGIP-13

```
tmsh create ltm virtual vip3 destination 10.99.99.129:80 source-address-translation {
→type automap } pool pool1 profiles add { tcp http }

tmsh save sys config partitions all
```

E_A_BIGIP-13: Your virtual server should now show available on E_A_BIGIP-13

```
root@E_A_BIGIP-13:Active:Standalone] config # tmsh show ltm virtual vip3

-----
Ltm::Virtual Server: vip3
-----

Status
  Availability      : available
  State             : enabled
  Reason            : The virtual server is available
  CMP               : enabled
  CMP Mode          : all-cpus
  Destination       : 10.99.99.129:80
```

E_B_BIGIP-13: Your virtual server should now show available on E_B_BIGIP-13

```
root@E_B_BIGIP-13:Active:Standalone] config # tmsh show ltm virtual vip3
-----
Ltm::Virtual Server: vip3
-----
Status
  Availability      : available
  State             : enabled
  Reason           : The virtual server is available
  CMP              : enabled
  CMP Mode         : all-cpus
  Destination      : 10.99.99.129:80
```

Verify path via Virtual Server 10.99.99.129 which falls on 10.99.99.128/25 is via East DC with introduction of adding the /25's from East DC.

csr1000v-SP_C: Verify path via Virtual Server 10.99.99.129 which falls on 10.99.99.128/25 is via East DC with introduction of adding the /25's from East DC. You can validate via traceroute to 10.99.99.129 on SP_C:

```
csr1000v-SP_C>traceroute 10.99.99.129
Type escape sequence to abort.
Tracing the route to 10.99.99.129
VRF info: (vrf in name/id, vrf out name/id)
 1 172.16.99.4 [AS 65001] 4 msec 6 msec 5 msec
 2 172.16.6.4 [AS 65002] 7 msec 8 msec 9 msec
 3 10.99.99.129 [AS 988] 7 msec 8 msec 7 msec
csr1000v-SP_C>
```

csr1000v-SP_C: You can validate via show ip route for 10.99.99.129 on SP_C:

```
csr1000v-SP_C>sh ip route 10.99.99.129
Routing entry for 10.99.99.128/25
  Known via "bgp 65003", distance 20, metric 0
  Tag 65002, type external
  Last update from 172.16.99.4 00:24:58 ago
  Routing Descriptor Blocks:
  * 172.16.99.4, from 172.16.99.4, 00:24:58 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
    Route tag 65002
    MPLS label: none
csr1000v-SP_C>
```

Jumpbox: Verify curl to Virtual Server 10.99.99.129 is up via East DC:

```
root@jumphost:~# curl 10.99.99.129
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
root@jumphost:~#
```

Jumpbox: Verify traceroute to Virtual Server 10.99.99.129 is East DC.

```
root@jumphost:~# traceroute 10.99.99.129
traceroute to 10.99.99.129 (10.99.99.129), 30 hops max, 60 byte packets
 1 192.168.1.15 (192.168.1.15) 2.569 ms 9.153 ms 9.097 ms
```

```
2 172.16.99.4 (172.16.99.4) 23.348 ms 22.639 ms 22.585 ms
3 172.16.6.4 (172.16.6.4) 25.018 ms 24.391 ms 23.766 ms
4 10.99.99.129 (10.99.99.129) 30.824 ms 30.220 ms 39.918 ms
root@jumphost:~#
```

Note: Congratulations! You have successfully completed lab 2.

5.2.3 Lab 3: Troubleshooting

Location and content of dynamic routing log files:

For each dynamic routing protocol, the BIG-IP system logs messages to a file that pertains to the route domain in which the protocol is running. An example of the path name to a dynamic routing log file is `/var/log/zebos/rd1/zebos.log` file, where `rd1` is the route domain of the protocol instance.

On `E_A_BIGIP` configure the following logging statement for BGP:

```
[root@E_A_BIGIP-13:Active:Standalone] log # imish
E_A_BIGIP-13.local[0]>enable
E_A_BIGIP-13.local[0]#conf t
Enter configuration commands, one per line. End with CNTL/Z.
E_A_BIGIP-13.local[0](config)#log file /var/log/zebos/rd0/zebos.log
E_A_BIGIP-13.local[0]#wr
Building configuration...
[OK]
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#
E_A_BIGIP-13.local[0]#exit
```

On `E_A_BIGIP` configure BGP debugging:

```
E_A_BIGIP-13.local[0]#deb bgp all
```

```
E_A_BIGIP-13.local[0]#sh debug bgp
BGP debugging status:
  BGP debugging is on
  BGP nht debugging is on
  BGP nsm debugging is on
  BGP events debugging is on
  BGP keepalives debugging is on
  BGP updates debugging is on
  BGP fsm debugging is on
  BGP filter debugging is on
  BGP Route Flap Dampening debugging is on
  BGP Bidirectional Forwarding Detection is on
```

```
E_A_BIGIP-13.local[0]#wr
Building configuration...
[OK]
```

On `E_A_BIGIP` view BGP log messages:

```

root@(E_A_BIGIP-13) (cfg-sync Standalone) (Active) (/Common) (tmos) # quit
[root@(E_A_BIGIP-13:Active:Standalone)] config # cd /var/log
[root@(E_A_BIGIP-13:Active:Standalone)] log # ls | grep zebos
zebos
[root@(E_A_BIGIP-13:Active:Standalone)] log # cd zebos
[root@(E_A_BIGIP-13:Active:Standalone)] zebos # ls
rd0
[root@(E_A_BIGIP-13:Active:Standalone)] zebos # cd rd0
[root@(E_A_BIGIP-13:Active:Standalone)] rd0 # ls
zebos.log
[root@(E_A_BIGIP-13:Active:Standalone)] rd0 # more zebos.log
2018/07/09 11:49:14 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.20.4 (FD=11)
2018/07/09 11:49:14 informational: BGP : [NETWORK] Accept Thread: 10.2.20.4 - No such
↳Peer configured
2018/07/09 11:49:16 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.40.4 (FD=11)
2018/07/09 11:49:16 informational: BGP : [NETWORK] Accept Thread: 10.2.40.4 - No such
↳Peer configured
2018/07/09 11:49:21 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.20.4 (FD=11)
2018/07/09 11:49:21 informational: BGP : [NETWORK] Accept Thread: 10.2.20.4 - No such
↳Peer configured
2018/07/09 11:49:26 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.40.4 (FD=11)
2018/07/09 11:49:26 informational: BGP : [NETWORK] Accept Thread: 10.2.40.4 - No such
↳Peer configured
2018/07/09 11:49:28 informational: BGP : [RIB] Scanning BGP Network Routes...
2018/07/09 11:49:30 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.20.4 (FD=11)
2018/07/09 11:49:30 informational: BGP : [NETWORK] Accept Thread: 10.2.20.4 - No such
↳Peer configured
2018/07/09 11:49:33 informational: BGP : [INIT] peer: 10.2.20.4 GR enabled for IPv4
↳(1) .
2018/07/09 11:49:33 informational: BGP : 10.2.20.4-Outgoing [FSM] State: Idle Event:
↳35 op_state: MASTER
2018/07/09 11:49:33 informational: BGP : 10.2.20.4-Outgoing [FSM] State: Idle Event:
↳1 op_state: MASTER
2018/07/09 11:49:36 informational: BGP : [NETWORK] Accept Thread: Incoming conn from
↳host 10.2.40.4 (FD=11)
2018/07/09 11:49:36 informational: BGP : [NETWORK] Accept Thread: 10.2.40.4 - No such
↳Peer configured

```

On E_A_BIGIP disable BGP debugging:

```

E_A_BIGIP-13.local[0] # undebg bgp all

E_A_BIGIP-13.local[0] # wr
Building configuration...
[OK]

```

Note: This completes Lab 3

