
F5 DNS Automation Demo 12.1.x Documentation

Release

Eric Chen, Vladimir Bojkovic

Apr 04, 2017

Lab 1: Connecting to UDF 2.0

1	Connecting to UDF 2.0	3
2	Launching Deployment	5
3	About the Demo Environment	7
4	Servers in the Demo	9
5	Connecting with Windows RDP	11
6	Licensing/Resetting BIG-IP	13
7	Creating BIG-IP DNS Sync Group	15
8	Configuring BIG-IP DNS	17
8.1	Enabling DNS Sync	17
8.2	Add Data Center	17
8.3	Add Server	17
8.4	Syncing BIG-IP DNS	18
9	Optional Exercises	21
9.1	DNS Cache	21
9.2	DNS Profiles	22
9.3	DNS Listeners	24
9.4	LTM Configuration	26
9.5	DNS Topology	29
9.6	Verifying configuration	33
9.7	Testing Internal Connections	33
9.8	Testing External Connections	34
10	F5 Python SDK	35
11	Restoring the BIG-IP Configuration	37
12	Run Demo	39
13	Exploring the Demo	41
13.1	Application Services Integration iApp	41

13.2	Testing Connections	42
13.3	Testing External Connections	43
13.4	Testing Internal Connections	43
14	Optional Exercises	45
14.1	Changing the requirements	45
14.2	Automating the change	45
15	Lab Appendix	47
15.1	The Script	47
16	Welcome to the Lab 4 Lab Guide	51
17	Step 0 - Restoring the BIG-IP Configuration	53
18	Step 1 - login into Jenkins	55
19	Step 2 - deploying F5 config via automation scripts using Jenkins	57
20	Jenkins Pipeline	61
21	Restoring the BIG-IP Configuration	63
22	Creating a pipeline	65
23	Launching the pipeline	69
24	Failing Tests	71
25	Indices and tables	75

This is a Demo of using the F5 Python SDK to automate the process of deploying BIG-IP DNS.

CHAPTER 1

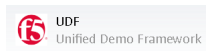
Connecting to UDF 2.0

This lab goes through the process of launching the demo environment, connecting via Windows RDP, and licensing/resetting the BIG-IP devices.

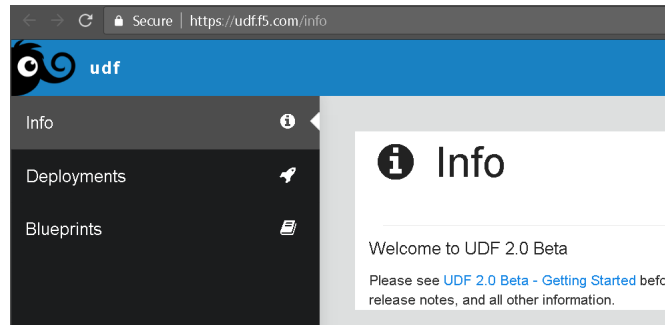
CHAPTER 2

Launching Deployment

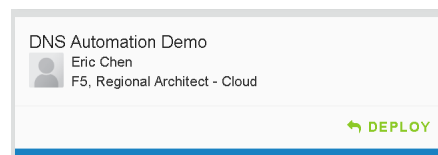
From <https://federate.f5.com> find the “UDF” link and click on it.



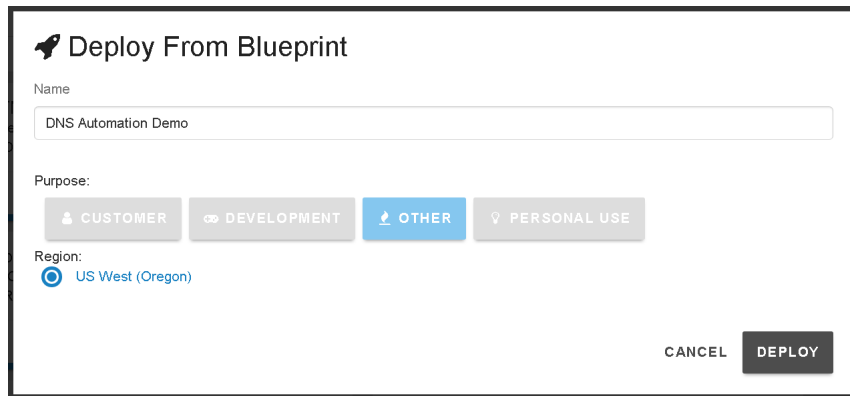
You should see the main UDF page. Click on the “Blueprints” link.



Find the “DNS Automation Demo” Blueprint and click on “Deploy”



You will see “Deploy From Blueprint” and click “Deploy” again.



Deploy From Blueprint

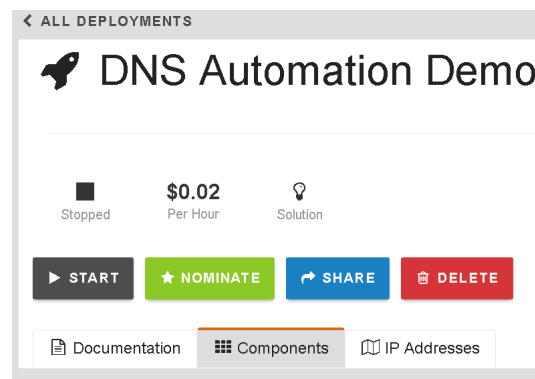
Name
DNS Automation Demo

Purpose:

Region:
☒ US West (Oregon)




You should now see the “DNS Automation Demo” screen that has the “Start” button listed.

Click on “Start”.

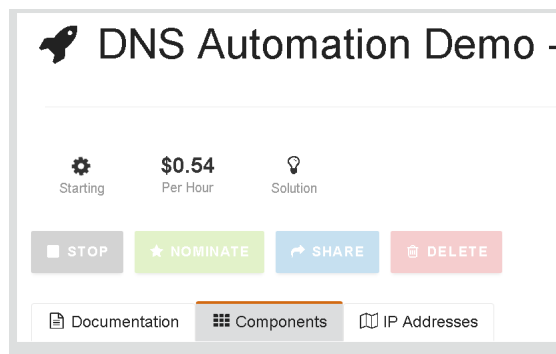


< ALL DEPLOYMENTS




DNS Automation Demo

 **Stopped**  **\$0.02**  **Solution**
Per Hour

The Deployment is now starting.



DNS Automation Demo -

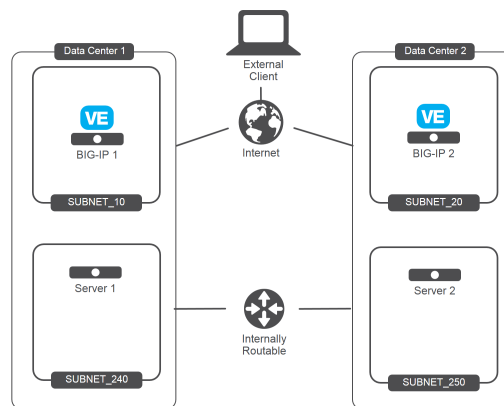
 **Starting**  **\$0.54**  **Solution**
Per Hour

CHAPTER 3

About the Demo Environment

This demo is designed to provide a solution with the following attributes.

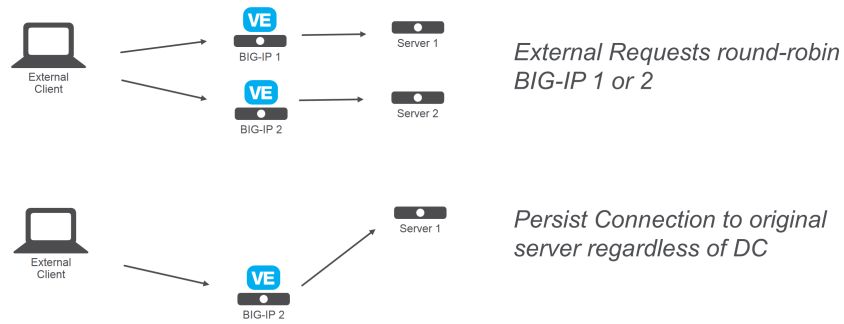
- Two BIG-IP devices in separate Data Centers (Regions, Availability Zone, etc...)
- Two backend servers in separate DC
- The two DC are routable to each other via L3
- Provide recursive DNS for internal clients



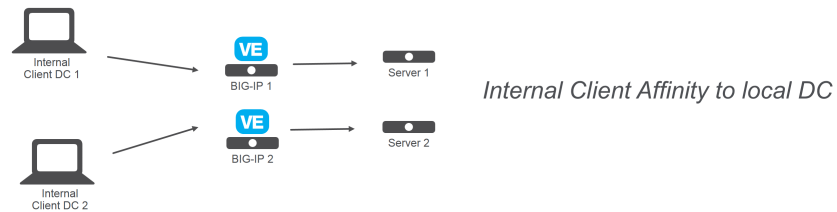
The desired behavior for requests

- External clients round-robin between backend servers
- Persist External client requests to original DC server if requests move between DC
- Internal client requests will have affinity to local DC server

DNS Automation Demo: External Clients



DNS Automation Demo: Internal Clients



CHAPTER 4

Servers in the Demo

Hostname	MGMT IP	Network IP	Login / Password
win2008-rdp	10.1.1.10	10.1.20.5	administrator / [see details page]
bigip1	10.1.1.7	10.1.10.240	admin / admin
bigip2	10.1.1.8	10.1.10.240	admin / admin
server1	10.1.1.4	10.1.240.10	centos or user / [ssh key]
server2	10.1.1.5	10.1.250.10	centos or user / [ssh key]
automation	10.1.1.6	10.1.20.8	centos or user / [ssh key]
jenkins	10.1.1.11		ubuntu / [ssh key]

CHAPTER 5

Connecting with Windows RDP

Once the Deployment is finished starting click on the “Components” tab to get a view like the following.

The screenshot shows the 'DNS Automation Demo' interface. At the top, it says 'ALL DEPLOYMENTS'. Below that, there's a rocket icon and the title 'DNS Automation Demo'. A status bar shows 'Running' with a play icon, '\$0.54 Per Hour', and 'Solution' with a lightbulb icon. Below this are four buttons: 'STOP' (black), 'NOMINATE' (green), 'SHARE' (blue), and 'DELETE' (red). A tab bar at the bottom of this section has 'Documentation', 'Components' (selected), and 'IP Addresses'. The main content area is divided into three columns: 'F5 Products', 'Subnets', and 'Systems'. Each column has a '+ ADD' button. Under 'F5 Products', there's a card for 'bigip2' with 'BIG-IP 12.1.1.1.0.196' and a 'Running' status. Under 'Subnets', there's a card for 'Management' with '10.1.1.0/24' and a 'Running' status. Under 'Systems', there's a card for 'server2' with 'CentOS 7' and a 'Running' status. Each card has 'ACCESS' and 'DETAILS' links at the bottom.

Find the win2008-rdp component and click on the “Access” button to display the “RDP” link.

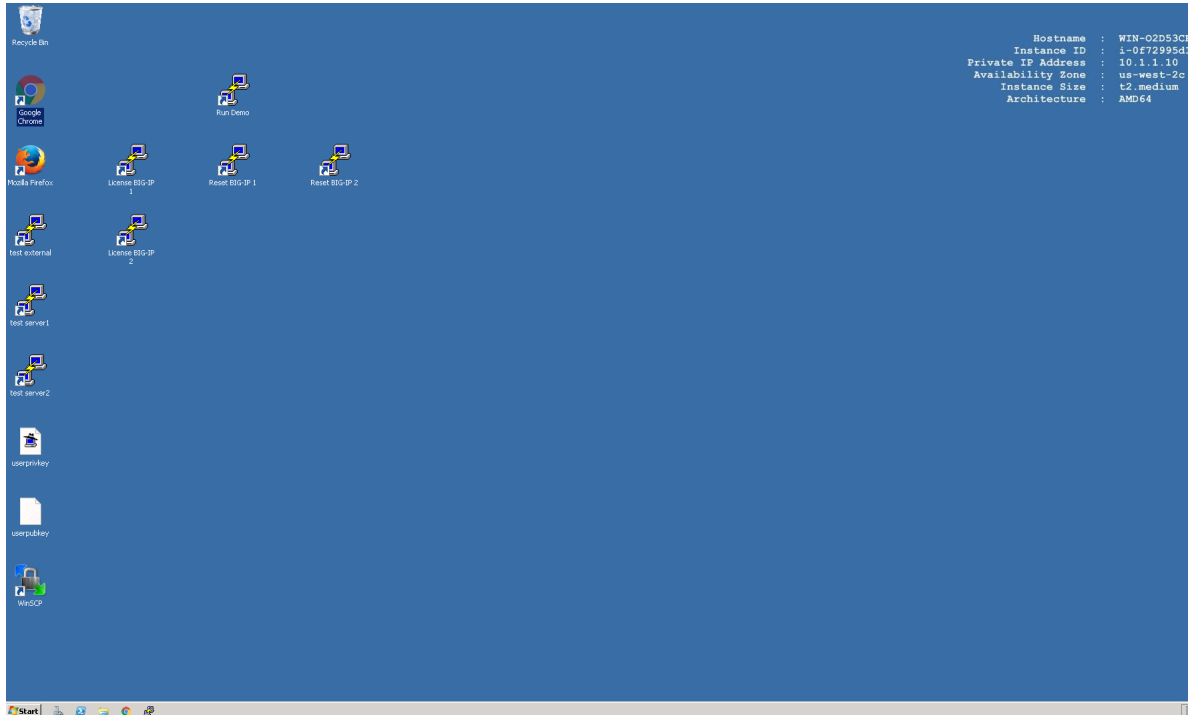
This screenshot shows a detailed view of the 'win2008-rdp' component, which is a 'Windows Server 2008 R2 Base'. It shows a 'Running' status with a play icon. At the bottom, there are 'ACCESS' and 'DETAILS' links. The 'ACCESS' link is highlighted, and a dropdown menu is open showing the 'RDP' option. The word 'automation' is visible at the very bottom of the interface.

Download/launch the RDP link.

Note that RDP will launch full screen

For HiDPI displays you may want to re-size your screen first

The username is “Administrator” and the password can be found under the “Details” page of the win2008-rdp component. Note: copy/paste of the password into the RDP session does not work. You can copy the password, open the RDP session settings in your RPD client and paste the password there. Save it and open the RDP session. You should see a desktop that looks like the following.



CHAPTER 6

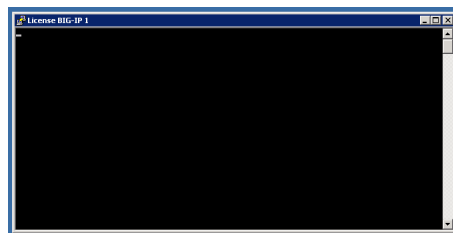
Licensing/Resetting BIG-IP

When a UDF Deployment is started you will need to re-license the device. There are links on the Desktop to expedite this process.

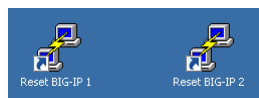
First find the “License” links.



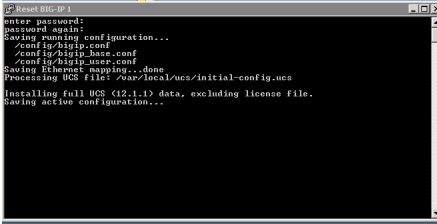
Double-click on both of these and you should see a window appear briefly like the following.



Next find the “Resetting” links.



Double-click on both of these and you should see a window appear briefly like the following.



```
Reset BIG-IP 1
enter password:
password echo:
Saving running configuration...
Copying /config/bigip.conf
Copying /config/bigip_base.conf
Copying /config/bigip_user.conf
Saving Ethernet mapping...done
Processing UCS file: /var/local/ucs/initial-config.ucs
Installing full UCS (12.1.1) data, excluding license file.
Saving active configuration...
```

The previous steps go through a scripted process of resetting the license and restoring the system to a known state via a UCS restore.

You are now ready to start the next Lab: *Creating BIG-IP DNS Sync Group*

Lab 1 will cover the process of connecting to UDF 2.0.

CHAPTER 7

Creating BIG-IP DNS Sync Group

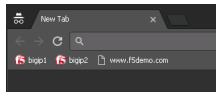
The following is adapted from the official F5 documentation:

<https://support.f5.com/kb/en-us/products/big-ip-dns/manuals/product/bigip-dns-implementations-12-1-0/3.html#conceptid>

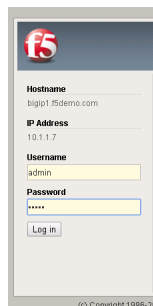
First start by launching Google Chrome.



In Chrome you should see links to BIG-IP 1 and 2.



Login to BIG-IP 1 *username: admin, password: admin*



Take a look at the current config. You should see BIG-IP DNS is provisioned.

Current Resource Allocation	
CPU	MGMT TMM(88%)
Disk (86GB)	MGMT
Memory (7.8GB)	MGMT TMM
Module Provisioning	
Management (MGMT)	Small N
Carrier Grade NAT (CGNAT)	Disabled S
Local Traffic (LTM)	<input checked="" type="checkbox"/> Nominal S
Application Security (ASM)	<input type="checkbox"/> None S
Fraud Protection Service (FPS)	<input type="checkbox"/> None N
Global Traffic (GTM)	<input checked="" type="checkbox"/> Nominal S

There is an existing self IP.

Network » Self IPs

Self IP List

<input checked="" type="checkbox"/>	Name	Application	IP Address	Netmask	VLAN / Tunnel
<input type="checkbox"/>	self_ip		10.1.10.240	255.255.255.0	external

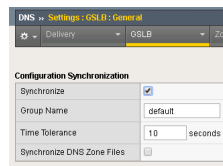
Configuring BIG-IP DNS

The following will go through the basic steps of setting up BIG-IP DNS.

Enabling DNS Sync

First go to DNS -> Settings -> GSLB -> General

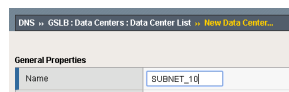
Find the “Synchronize” checkbox and click on it.



Add Data Center

Next go back to DNS -> GSLB -> Data Centers

Create a Data Center (DC) named SUBNET_10 and SUBNET_30.

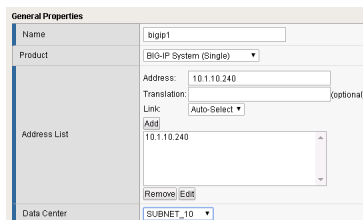


Add Server

Next go to DNS->GSLB->Servers

Create a server bigip1 associated with SUBNET_10 and bigip2 associated with SUBNET_30.

Name	Address	Data Center
bigip1	10.1.10.240	SUBNET_10
bigip2	10.1.30.240	SUBNET_30



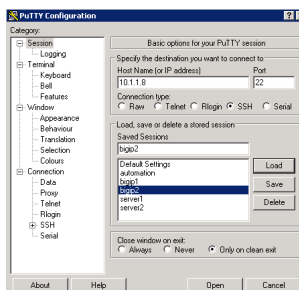
Syncing BIG-IP DNS

At this point BIG-IP 1 has the desired BIG-IP DNS configuration, but it needs to be synced with BIG-IP 2.

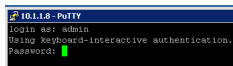
From the Desktop launch “Putty”.



Find the BIG-IP 2 login.

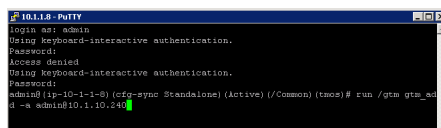


Login. *username: admin, password: admin*



Run the command:

```
run /gtm gtm_add -a admin@10.1.10.240
```



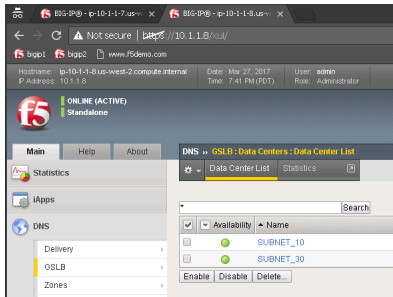
When prompted confirm/enter password.

```

10.1.1.8 - PUTTY
login as: admin
Using Keyboard-interactive authentication.
Password:
Access denied
Using Keyboard-interactive authentication.
Password:
admin@10-1-1-8 (cfg-sync Standalone) (Active) (/Common) (tmsh)# run /gtm gtm_ad
d -a admin@10.1.1.240
WARNING: Running this script will wipe out the current configuration
files (bigip_gtm.conf, named.conf and named zone files) on the BIG-IP GTM
Controller on which this script is run. The configuration will be
replaced with the configuration of the remote BIG-IP GTM Controller
in the specified sync group
The local BIG-IP GTM MUST already be added in the configuration of the
other GTM.
NOTE: The current master key of this BIG-IP will be changed to match the
master key of the remote BIG-IP GTM.
The BIG-IP config will be saved via:
tmsh save sys config
after the master key is changed.
Are you absolutely sure you want to do this? [y/n] y

```

In Chrome login to BIG-IP 2 and you should see that it is now synced.



You should have a pair of BIG-IP devices that are in a DNS Sync Group. The next lab will go through the process of scripting these actions.

You are now ready to start the next Lab: *F5 Python SDK*

CHAPTER 9

Optional Exercises

The following next lab, *F5 Python SDK*, will perform all of the following sections via an automation script. You can choose to perform all of these tasks to understand what the automation will be performing, but at the end of this lab you will delete the work that you have done.

DNS Cache

In the Demo environment we will use BIG-IP DNS as a DNS resolver. Create a DNS cache named “dns_cache”. This needs to be done separately on each BIG-IP device.

Under DNS -> Caches -> Cache List: Create a DNS cache profile “internal_cache” and accept default values.

DNS » Caches : Cache List » New...

General Properties	
Name	internal_cache
Resolver Type	Resolver ▼
Route Domain Name	0 ▼

DNS Cache	
Message Cache Size	1048576 bytes
Resource Record Cache Size	10485760 bytes
Name Server Cache Count	16536 entries
Answer Default Zones	<input type="checkbox"/> Enabled
RRSet Rotate	none ▼

DNS Resolver	
Use IPv4	<input checked="" type="checkbox"/> Enabled
Use IPv6	<input checked="" type="checkbox"/> Enabled
Use UDP	<input checked="" type="checkbox"/> Enabled
Use TCP	<input checked="" type="checkbox"/> Enabled
Max. Concurrent UDP Flows	8192
Max. Concurrent TCP Flows	20
Max. Concurrent Queries	1024
Unsolicited Reply Threshold	0
Allowed Query Time	200
Randomize Query Character Case	<input checked="" type="checkbox"/> Enabled
	IP Address: <input type="text"/>
	<input type="button" value="Add"/>
Root Hints (Optional) Leave	<input type="text"/>

DNS Profiles

Two DNS profiles are required. One for providing a resolving DNS server and one for external DNS requests (bad idea to have an open resolver on the internet). Now create them on both BIG-IP's.

External DNS Profile

Under DNS -> Delivery -> Profiles -> DNS: Create a profile named “external_dns” that only provides GSLB and disables fallback to BIND.

General Properties		
Name	external_dns	
Parent Profile	dns	
Denial of Service Protection Custom <input type="checkbox"/>		
Rapid Response Mode	Disabled	<input type="checkbox"/>
Rapid Response Last Action	Drop	<input type="checkbox"/>
Hardware Acceleration		
Protocol Validation	Disabled	<input type="checkbox"/>
Response Cache	Disabled	<input type="checkbox"/>
DNS Features		
DNSSEC	Disabled	<input checked="" type="checkbox"/>
GSLB	Enabled	<input type="checkbox"/>
DNS Express	Disabled	<input checked="" type="checkbox"/>
DNS Cache	Disabled	<input checked="" type="checkbox"/>
DNS Cache Name	Select...	<input checked="" type="checkbox"/>
DNS IPv6 to IPv4	Disabled	<input type="checkbox"/>
Unhandled Query Actions	Allow	<input type="checkbox"/>
Use BIND Server on BIG-IP	Disabled	<input checked="" type="checkbox"/>

Internal DNS Profile

Under DNS -> Delivery -> Profiles -> DNS: Create a profile named “internal_dns” that enables a DNS cache for resolving names.

DNS » Delivery : Profiles : DNS » New DNS Profile...		
General Properties		
Name	internal_dns	
Parent Profile	dns	
Denial of Service Protection Custom <input type="checkbox"/>		
Rapid Response Mode	Disabled	<input type="checkbox"/>
Rapid Response Last Action	Drop	<input type="checkbox"/>
Hardware Acceleration		
Protocol Validation	Disabled	<input type="checkbox"/>
Response Cache	Disabled	<input type="checkbox"/>
DNS Features		
DNSSEC	Enabled	<input type="checkbox"/>
GSLB	Enabled	<input type="checkbox"/>
DNS Express	Enabled	<input type="checkbox"/>
DNS Cache	Enabled	<input checked="" type="checkbox"/>
DNS Cache Name	internal_cache	<input checked="" type="checkbox"/>
DNS IPv6 to IPv4	Disabled	<input type="checkbox"/>
Unhandled Query Actions	Allow	<input type="checkbox"/>
Use BIND Server on BIG-IP	Enabled	<input type="checkbox"/>

DNS Listeners

For external DNS we have two listeners for each BIG-IP. One TCP and one UDP. First create on both BIG-IP's the external Listeners for TCP and UDP. Apply the external_dns profile to each. Use these IP addresses:

Name	Address	Port
bigip1	10.1.10.13	53
bigip2	10.1.30.13	53

External DNS Listener

DNS -> Delivery -> Listeners Here the external TCP listener

The screenshot shows the F5 configuration interface for a DNS listener. The breadcrumb trail is: DNS » Delivery: Listeners: Listener List » Properties: external_TCP_Listener. The interface has tabs for Properties, Load Balancing, iRules, and Statistics. The 'General' section includes fields for Name (external_TCP_Listener), Partition (Common), Description (empty), and State (Enabled). The 'Listener' section is set to 'Basic' and includes fields for Destination (Type: Host, Address: 10.1.10.13) and VLAN Traffic (All VLANs). The 'Service' section is also set to 'Basic' and includes fields for Protocol (TCP) and DNS Profile (external_dns). At the bottom are 'Update' and 'Delete...' buttons.

DNS » Delivery: Listeners: Listener List » Properties: external_TCP_Listener	
⚙️	Properties Load Balancing iRules Statistics
General	
Name	external_TCP_Listener
Partition	Common
Description	
State	Enabled
Listener: Basic	
Destination	Type: <input checked="" type="radio"/> Host <input type="radio"/> Network Address: 10.1.10.13
VLAN Traffic	All VLANs
Service: Basic	
Protocol	TCP
DNS Profile	external_dns
Update Delete...	

Here the external UDP listener

DNS » Delivery : Listeners : Listener List » Properties : external_UDP_Listener

⚙️ Properties Load Balancing iRules Statistics

General

Name	external_UDP_Listener
Partition	Common
Description	<input type="text"/>
State	Enabled ▼

Listener: Basic ▼

Destination	Type: <input checked="" type="radio"/> Host <input type="radio"/> Network Address: 10.1.10.13
VLAN Traffic	All VLANs ▼

Service: Basic ▼

Protocol	UDP ▼
DNS Profile	external_dns ▼

Update Delete...

Next go to LTM Virtual server menu. The external listeners will appear as virtual servers.

Local Traffic » Virtual Servers : Virtual Server List

⚙️ Virtual Server List Virtual Address List Statistics

Search Create...

<input checked="" type="checkbox"/>	Status	Name	Description	Application	Destination	Service Port	Type	Resources	Partition / Path
<input type="checkbox"/>		external_TCP_Listener			10.1.10.13	53	Standard	Edit...	Common
<input type="checkbox"/>		external_UDP_Listener			10.1.10.13	53	Standard	Edit...	Common

Enable Disable Delete...

Internal DNS Listener

Next create on each BIG-IP internal listeners via the LTM menu. The listener is a virtual server. Specify following source address range on each internal listener: 10.1.0.0/16 and apply the “internal_dns” DNS profile. Keep all other settings as default. Use these IP addresses:

Name	Address	Port
bigip1	10.1.10.13	53
bigip2	10.1.30.13	53

Create listeners for TCP and UDP Here is an example of the internal TCP Listener:

Local Traffic » Virtual Servers : Virtual Server List » internal_TCP_listener	
⚙️	Properties Resources Statistics
General Properties	
Name	internal_TCP_listener
Partition / Path	Common
Description	<input type="text"/>
Type	Standard ▼
Source Address	10.1.0.0/16
Destination Address/Mask	10.1.10.13
Service Port	53 Other: ▼
Notify Status to Virtual Address	<input checked="" type="checkbox"/>
Link	None
Availability	<input checked="" type="checkbox"/> Unknown (Enabled) - The children pool member(s) either don't have service checking enabled, or service check results are not available yet
Syncookie Status	Off
State	Enabled ▼
Configuration: Advanced ▼	
Protocol	TCP ▼
Protocol Profile (Client)	tcp ▼
Protocol Profile (Server)	(Use Client Profile) ▼

LTM Configuration

Now we have to configure the LTM section of both BIG-IP's. Since both BIG-IP's are standalone the configuration steps has to be applied to both BIG-IP's.

First create an http profile named "http-XFF" that inserts X-Forwarded-For headers Local Traffic -> Profiles -> Services -> HTTP

Local Traffic » Profiles : Services : HTTP » **New HTTP Profile...**

General Properties

Name	http-XFF
Proxy Mode	Reverse ▼
Parent Profile	http ▼

Settings Custom ☐

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"/>	<input type="checkbox"/>
Insert X-Forwarded-For	Enabled ▼	<input checked="" type="checkbox"/>

In the Demo LTM is configured to use cookie persistence, insert X-Forwarded-For headers, and use Priority Groups for delivering traffic.

Pools

Create a pool “serverpool” on each BIG-IP. Local Traffic -> Pools

Assign HTTP and TCP monitors Enable Priority Group Activation with “1 Available Member”

Device	Pool Member #1	Port	Priority Group	Pool Member #2	Port	Priority Group
bigip1	10.1.240.10	80	10	10.1.250.10	80	0
bigip2	10.1.250.10	80	10	10.1.240.10	80	0

Pool config example:

Local Traffic » Pools : Pool List » New Pool...

Configuration: Basic ▾

Name: serverpool

Description:

Health Monitors

Active: /Common, tcp, http

Available: https_443, https_head_f5, inband, tcp_half_open, udp

Resources

Load Balancing Method: Round Robin ▾

Priority Group Activation: Less than... ▾ 1 Available Member(s)

New Members

☒ New Node ☐ New FQDN Node

Node Name: 10.1.250.10 (Optional)

Address: 10.1.250.10

Service Port: 80 HTTP ▾

Priority: 0 (optional)

Add

R:1 P:10 C:0 10.1.240.10 10.1.240.10 :80
R:1 P:0 C:0 10.1.250.10 10.1.250.10 :80

Edit Delete

Cancel Repeat Finished

Virtual Servers

In the next step create two standard TCP virtual servers per BIG-IP. One external and one internal. Apply the http-XFF profile, SNAT Automap and the pool “serverpool”

Use following IP addresses

Device	Name	Address	Port
bigip1	external_vs	10.1.10.10	80
bigip1	internal_vs	10.1.10.100	80
bigip2	external_vs	10.1.30.10	80
bigip2	internal_vs	10.1.30.100	80

Here a configuration example:

Local Traffic » Virtual Servers : Virtual Server List » [New Virtual Server...](#)

General Properties

Name	external_vs
Description	
Type	Standard ▼
Source Address	
Destination Address/Mask	10.1.10.10
Service Port	80 HTTP ▼
Notify Status to Virtual Address	<input checked="" type="checkbox"/>
State	Enabled ▼

Configuration: Basic ▼

Protocol	TCP ▼
Protocol Profile (Client)	tcp ▼
Protocol Profile (Server)	(Use Client Profile) ▼
HTTP Profile	http-XFF ▼

Source Address Translation Auto Map ▼

Default Pool + serverpool ▼

DNS Topology

After BIG-IP DNS and BIG-IP LTM are configured and ready to run, it is time to create the logical geographical load balancing. BIG-IP DNS will receive DNS requests and respond based on the location of the requesting IP.

The demo will show two sections:

1. Split DNS: BIG-IP DNS will respond to internal clients with private IP addresses differently than for external clients with IP addresses that are not internal.
2. **regional Loadbalancing - based on the region of the client IP the response will be different.** Note: we will simulate the requin with a private IP address range, because this lab is not exposed to the Internet.

The steps to create geolocation based load balancing on BIG-IP DNS are:

1. add the virtual servers
2. create the pools
3. create wide-IPs
4. create regions
5. create records

All configurations have to be applied one BIG-IP DNS only. The config changes will be synced to the other BIG-IP DNS via a sync group that was created before.

Step 1: Virtual Servers

BIG-IP DNS has to be aware of the services that are provided by BIG-IP LTM. BIG-IP DNS sees BIG-IP LTM as a Server that is bound to a datacenter. For BIG-IP DNS the virtual servers on the BIG-IP LTM are virtual servers on the server in the datacenter.

Create the virtual servers on BIG-IP DNS under:

DNS -> GSLB -> Servers

Use following IP addresses for the virtual servers:

Device	Name	IP:PORT	Health Monitor
bigip1	external_vs	10.1.10.10:80	bigip
bigip1	internal_vs	10.1.10.100:80	bigip
bigip2	external_vs	10.1.30.10:80	bigip
bigip2	internal_vs	10.1.30.100:80	bigip

Here an example:

The screenshot shows the configuration page for a virtual server named 'external_vs' under the path 'DNS >> GSLB : Servers : Server List >> external_vs'. The 'General Properties' section includes fields for Name (external_vs), Address (10.1.10.10), Service Port (HTTP), Translation (10.1.10.10), Translation Service Port (80), Assigned Link, Link (Auto-Select), and Status (Enabled). The 'Configuration' section is set to 'Basic'. Under 'Health Monitors', there are two lists: 'Selected' containing '/Common bigip' and 'Available' containing '/Common gateway_icmp', 'gtp', 'http', and 'http_head_f5'.

Step 2: Pools

The next step is to configure the pool. under: DNS -> GSLB -> Pools

Configure two pools (internal_pool and external_pool) with following load balancing methods: * Preferred: Topology
* Alternate: Round-Robin * Fallback: none

use following settings:

Name	Type	Pool Members
external_pool	A	bigip1:external_vs, bigip2:external_vs
internal_pool	A	bigip1:internal_vs, bigip2:internal_vs

Here an example:

DNS » GSLB : Pools : Pool List » **New Pool...**

General Properties

Name	external_pool
Type	A
State	Enabled

Configuration

Health Monitors	<div> <div>Selected</div> <div>Available</div> <div> <div> <div><<</div> <div>>></div> </div> <div> <div>Up</div> <div>Down</div> </div> </div> <div> <div>/Common</div> <div>bigip</div> <div>bigip_link</div> <div>external</div> <div>firepass_gtm</div> </div> </div>
Availability Requirements	All Health Monitor(s)
Limit Settings	Bits: Disabled Packets: Disabled Current Connections: Disabled
Manual Resume	<input type="checkbox"/>
TTL	30
Dynamic Ratio	<input type="checkbox"/>
Maximum Answers Returned	1
Verify Member Availability	<input checked="" type="checkbox"/>

Members

Load Balancing Method	Preferred: Topology Alternate: Round Robin Fallback: Return to DNS
Fallback IP	0.0.0.0
Member List	Virtual Server: internal_vs (/Common/bigip1) - 10.1.10.100:80 Ratio: 1 Add external_vs (/Common/bigip1) - 10.1.10.10:80, Ratio(1) external_vs (/Common/bigip2) - 10.1.30.10:80, Ratio(1) Delete Up Down

Step 3: Wide IPs

The next step is to configure the Wide IP. under: DNS -> GSLB -> Wide IPs
use following settings and Topology as load balancing method

Name	Type	Pools	Last Resort pool
www.f5demo.com	A	external_pool, internal_pool	external_pool

Here an example:

DNS » GSLB : Wide IPs : Wide IP List » [New...](#)

General Properties: Basic ▾

Name	www.f5demo.com
Type	A ▾
Description	
State	Enabled ▾

iRules

iRule List	Selected	Available
	<div></div>	<div></div>
	<< >>	
	Up Down	

Pools

Load Balancing Method	Topology ▾
Persistence	Disabled ▾
Pool List	Pool Select... ▾
	Ratio 1
	Add
	<div> /Common external_pool(A) Ratio(1) internal_pool(A) Ratio(1) </div>
	Edit Delete Up Down
Last Resort Pool	None ▾

Step 4: Topology Regions

The next step is to define regions that will be used by topology records. under DNS -> GSLB -> Topology -> Regions

Use following IP addresses:

Name	Subnets
internal_network	10.1.240.0/20
region_1	10.1.10.0/24,10.1.240.0/24
region_2	10.1.30.0/24,10.1.250.0/24

Here an example:

DNS » GSLB : Topology : Regions » Properties : internal_network

Properties

General Properties

Name	internal_network
Partition / Path	Common

Region Members

Member List

Member Type: IP Subnet
 is
 IP Subnet: /
 Add
 IP Subnet is 10.1.240.0/20
 Delete

Step 5: Topology Records

The last step is to define the topology records, that BIG-IP DNS will use for load balancing decisions under DNS -> GSLB -> Topology -> Records

use following settings

Source	is/is not	Destination
region /Common/internal_network	is not	pool /Common/external_pool
region /Common/internal_network	is	pool /Common/internal_pool
region /Common/region_1	is	region /Common/region_1
region /Common/region_2	is	region /Common/region_2

Verifying configuration

Testing Internal Connections

Now it is time to test if your configuration works.

Now run the “Test Server1” link.

```

test server1
HOST: www.f5demo.com
CLIENT IP: 10.1.19.240
SERVER IP: 10.1.240.10
Data Center: US-EAST-1D

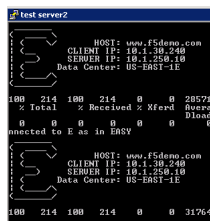
100 219 100 219 0 0 33242
% Total % Received % Xferd  Average
0 0 0 0 0 0 0
connected to 0 as in DOG

HOST: www.f5demo.com
CLIENT IP: 10.1.19.240
SERVER IP: 10.1.240.10
Data Center: US-EAST-1D

100 219 100 219 0 0 35362

```

and run the “Test server2” link.



The “test-server[1-2]” links are simulating requests from internal clients. Note that BIG-IP DNS is configured to prefer requests to the same Data Center.

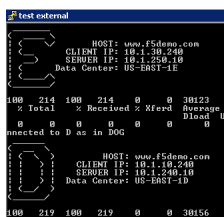
Question Can you explain how this is being done?

Testing External Connections

Find the “Test External” link.



Double-click on it and you should see:



The “Test External” link is simulating requests from an external client. BIG-IP DNS is configured to use round-robin load balancing between the two backend servers.

From Google Chrome find the link for “www.f5demo.com”. The Windows Desktop client is configured to act like an external client.

Question Using Google Chrome the requests will always go back to the same server, why?

Deploying without Automation. This will cover the basic steps of:

1. Adding BIG-IP servers to BIG-IP DNS
2. Creating BIG-IP DNS Cluster

This provides some context of what the automation will be performing.

Additional optional exercises.

1. Creating BIG-IP LTM Virtual Server and Pools
2. Creating BIG-IP DNS Virtual Server and Pools
3. Creating BIG-IP DNS Wide-IP

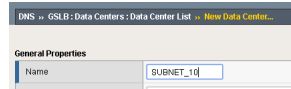
CHAPTER 10

F5 Python SDK

The F5 Python SDK provides an interface to the iControl REST interface.

This provides the ability to translate actions that you would have normally done via the GUI to actions that can be performed from Python.

Via GUI



Via Code

```
def add_datacenter(self, datacenter):  
    "add datacenter in BIG-IP DNS"  
    self.mgmt.tm.gtm.datacenters.datacenter.create(name=datacenter, partition=self.  
↪partition)
```

This Lab will combine the process of creating a BIG-IP DNS deployment in Lab 2 and automate the process using the F5 Python SDK. The Python code in this lab is run as a script to deploy BIG-IP configurations. Think of this script like a remote tmsh Command Line Interface (CLI) to the BIG-IP. Here's an example of adding a server to BIG-IP DNS using the script.

CLI example

```
python bigip_dns_helper.py --host=10.1.1.7 \  
--action add_datacenter --datacenter SUBNET_10
```

In this example we have created a Data Center (DC) named "SUBNET_10" using the F5 Python SDK example from before.

A full description of sample inputs (taken from this lab) can be found at the end of this lab in the Lab Appendix. *The Script*

For Python nerds; the pseudo code to invoke this from the CLI:

Python Pseudo Code

```
parser = OptionParser()
parser.add_option('--host')
parser.add_option('--datacenter')
parser.add_option('--action')

(options, args) = parser.parse_args()

dns_helper = DnsHelper(options.host)

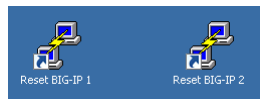
if options.action == 'add_datacenter':
    dns_helper.add_datacenter(options.datacenter)
```

Restoring the BIG-IP Configuration

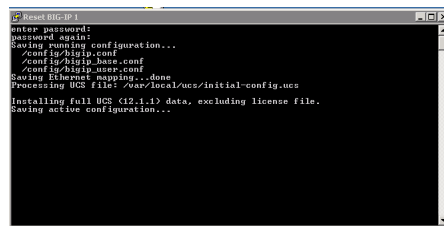
This step is to cleanup the BIG-IP config that was created in Lab 2. RDP into the Windows jump host.

Reset both BIG-IP to be the same state as after Lab 1.

Find the “Resetting” links on the Desktop.



Double-click on both of these and you should see a window appear briefly like the following.



```
Reset BIG-IP 1
enter password:
password again:
Saving running configuration...
/commit/bigip.conf
/commit/bigip_base.conf
/commit/bigip_user.conf
Saving Ethernet mapping...done
Processing UCS file: /var/local/ucs/initial-config.ucs
Installing full UCS (12.1.1) data, excluding license file.
Saving active configuration...
```

Verify that you no longer see the changes that were previously deployed.

Run Demo

This Lab is not intended to teach you how to write Python code, but instead demo how it can be leveraged to help automate a solution.

On the Desktop you will find the “Run Demo” link. Double-click the link.



The script is currently configured to output all the REST calls making for a verbose output.

[illegible]

An excerpt of the script that performs Lab 2 (create a DNS Sync Group):

```
python bigip_dns_helper.py --host=10.1.1.7 \
--action enable_sync
python bigip_dns_helper.py --host=10.1.1.7 \
--action add_datacenter --datacenter SUBNET_10
python bigip_dns_helper.py --host=10.1.1.7 \
--action add_datacenter --datacenter SUBNET_30

python bigip_dns_helper.py --host=10.1.1.7 \
--action add_server --datacenter SUBNET_10 --server_name bigip1 --server_ip=10.1.10.
↵240
python bigip_dns_helper.py --host=10.1.1.7 \
--action add_server --datacenter SUBNET_30 --server_name bigip2 --server_ip=10.1.30.
↵240
```

```
python bigip_dns_helper.py --host=10.1.1.7 \  
--action save_config  
sleep 3  
python bigip_dns_helper.py --host=10.1.1.8 \  
--action gtm_add --peer_host=10.1.1.7 --peer_selfip 10.1.10.240
```

There is the same number of steps involved, but one-click!

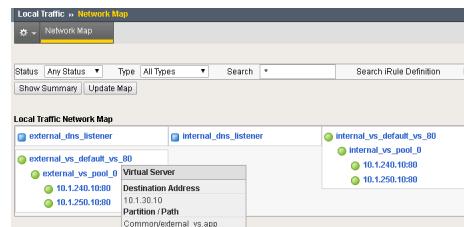
The full script can be found on [GitHub](#).

CHAPTER 13

Exploring the Demo

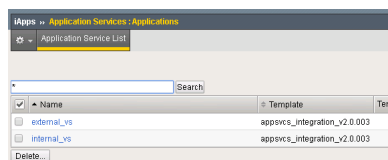
Take a look at what has been deployed. (Hint: Look at the Optional Exercise from Lab 2).

From the Network Map you can see that we have a set of external/internal DNS listeners and external/internal LTM Virtual Servers.



Application Services Integration iApp

The demo script utilizes the Application Services Integration iApp to deploy the LTM L4-L7 services.



iApp Scripts

The iApp is deployed using modified scripts from: <https://github.com/F5Networks/f5-application-services-integration-iApp/tree/master/scripts>

```
# import Application Services Integration iApp onto BIG-IP
python iapps/import_template_bigip.py --impl iapps/iapp.tcl --apl iapps/iapp.apl 10.
1.1.7 appsvcs_integration_v2.0.003
python iapps/import_template_bigip.py --impl iapps/iapp.tcl --apl iapps/iapp.apl 10.
1.1.8 appsvcs_integration_v2.0.003
```

```
# Create L4-L7 services
python iapps/deploy_iapp_bigip.py -r 10.1.1.7 iapps/sample_http.json --strings pool__
↪addr=10.1.10.10 \
  --pool_members=0:10.1.240.10:80:0:1:10:enabled:none,0:10.1.250.
↪10:80:0:1:0:enabled:none --iapp_name external_vs

python iapps/deploy_iapp_bigip.py -r 10.1.1.8 iapps/sample_http.json --strings pool__
↪addr=10.1.30.10 \
  --pool_members=0:10.1.250.10:80:0:1:10:enabled:none,0:10.1.240.
↪10:80:0:1:0:enabled:none --iapp_name external_vs

python iapps/deploy_iapp_bigip.py -r 10.1.1.7 iapps/sample_http.json --strings pool__
↪addr=10.1.10.100 \
  --pool_members=0:10.1.240.10:80:0:1:10:enabled:none,0:10.1.250.
↪10:80:0:1:0:enabled:none --iapp_name internal_vs

python iapps/deploy_iapp_bigip.py -r 10.1.1.8 iapps/sample_http.json --strings pool__
↪addr=10.1.30.100 \
  --pool_members=0:10.1.250.10:80:0:1:10:enabled:none,0:10.1.240.
↪10:80:0:1:0:enabled:none --iapp_name internal_vs
```

You can view the parameters used to configure the iApp under iApps -> Application Services.

The screenshot shows the F5 iApp configuration interface for 'external_vs'. The 'Basic' tab is selected, showing the 'Template Selection' section with 'external_vs' as the Name and 'iappves_integration_v2.0.003' as the Template. Below this is the 'F5 Application Services Integration iApp v2.0.003 (Community Edition)' section, which includes an 'Introduction' tab and a 'Please complete the following template' section. The 'iApp Options' section contains various settings: 'iApp: Strict Updates' (enabled), 'iApp: Statistics Handler Creation' (enabled), 'iApp: Mode' (auto), 'iApp: Log Level' (5), 'iApp: Route Domain' (auto), 'iApp: ASM: Deployment Mode' (preserve-bypass), and 'iApp: API: Deployment Mode' (preserve-bypass). The 'Virtual Server Listener & Pool Configuration' section includes fields for 'Virtual Server: Address' (10.1.30.10), 'Virtual Server: Mask' (255.255.255.255), 'Virtual Server: Port' (80), and 'Virtual Server: Default Pool Index' (0). It also shows a 'Pool: Pool Table' with 'Index' (0), 'Name' (external_vs), 'Description' (external_vs), 'LB Method' (round-robin), 'Monitor(s)' (0,1), and 'Adv Options' (min-active-member). The 'Pool: Member Default Port' is set to 80. The 'Pool: Members' section lists two members: 'Pool Id: 0' with 'IPNode Name' 10.1.250.10, 'Port' 80, 'Connection Limit' 0, 'Ratio' 1, 'Priority Group' 10, 'State' enabled, and 'Adv Options' none; and 'Pool Id: 1' with 'IPNode Name' 10.1.240.10, 'Port' 80, 'Connection Limit' 0, 'Ratio' 1, 'Priority Group' 0, 'State' enabled, and 'Adv Options' none.

Testing Connections

This demo is designed to provide a solution with the following attributes.

- Two BIG-IP devices in separate Data Centers (Regions, Availability Zone, etc...)
- Two backend servers in separate DC
- The two DC are routable to each other via L3
- Provide recursive DNS for internal clients

The desired behavior for requests

- External clients round-robin between backend servers

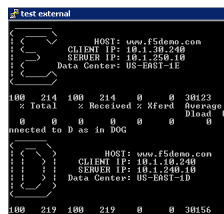
- Persist External client requests to original DC server if requests move between DC
- Internal client requests will have affinity to local DC server

Testing External Connections

Find the “Test External” link.



Double-click on it and you should see:



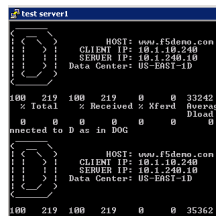
The “Test External” link is simulating requests from an external client. BIG-IP DNS is configured to use round-robin load balancing between the two backend servers.

From Google Chrome find the link for “www.f5demo.com”. The Windows Desktop client is configured to act like an external client.

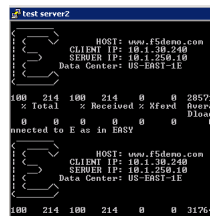
Question Using Google Chrome the requests will always go back to the same server, why? (Hint: Look at the Optional Exercise from Lab 2).

Testing Internal Connections

Now run the “Test Server1” link.



and run the “Test server2” link.



The “test-server[1-2]” links are simulating requests from internal clients. Note that BIG-IP DNS is configured to prefer requests to the same Data Center.

Question Can you explain how this is being done? (Hint: Look at the Optional Exercise from Lab 2).

Please proceed to Lab 4. [*Welcome to the Lab 4 Lab Guide*](#)

Changing the requirements

Can you change the behavior to the following:

1. External requests will not persist to the same backend server (still round-robin, Hint: one change to each external LTM Virtual Server)
2. Flip the affinity of the internal requests (could be done via either LTM/DNS)

Automating the change

The second change “flipping the affinity” can be done via changing the automation script to change how the LTM Virtual Servers are deployed. Reset the deployment and deploy with an updated deployment that implements that change.

CHAPTER 15

Lab Appendix

The Python Script that is used in this lab was created by Eric Chen. This is considered F5 contributed software. [K80012344](#) covers this.

From K80012344.

F5 contributed software

F5 employees create and test software contributed to GitHub as part of supporting the development community related to F5 products. However, if the software is not a specific tagged release listed as F5 Supported in this article, and if the software is not also available on the F5 Downloads site, F5 does not provide technical support for them, and may refer you back to GitHub for community support for the project in question. For more information and for community discussions, refer to the information available for that software in the GitHub repository.

The Script

Enable DNS Sync

```
python bigip_dns_helper.py --host=[MGMT IP] \  
                           --action enable_sync
```

This performs the GUI steps of *Enabling DNS Sync*

Add Data Center

```
python bigip_dns_helper.py --host=[MGMT IP] \  
                           --action add_datacenter --datacenter [Data Center Name]
```

This performs the GUI steps of *Add Data Center*

Save Config

```
python bigip_dns_helper.py --host=[MGMT IP] \  
                           --action save_config
```

This performs the TMSH equivalent of:

```
save /sys config  
save /sys config gtm-only
```

This ensures that the running configuration is saved to disk.

Syncing BIG-IP DNS

```
python bigip_dns_helper.py --host=[MGMT IP of BIG-IP that you will WIPE OUT DNS_  
→config] \  
                           --action gtm_add  
                           --peer_host=[MGMT IP of BIG-IP you want to copy from]  
                           --peer_selfip [SELF IP of BIG-IP you want to copy from]
```

This performs the steps of *Syncing BIG-IP DNS*

Create DNS Cache

```
python bigip_dns_helper.py --host=[MGMT IP] \  
                           --action create_dns_cache
```

This performs the steps of *DNS Cache*

Create External DNS Profile

```
python bigip_dns_helper.py --host=[MGMT IP] --action create_external_dns_profile
```

This performs the steps of *External DNS Profile*

Create Internal DNS Profile

```
python bigip_dns_helper.py --host=[MGMT IP] --action create_internal_dns_profile
```

This performs the steps of *Internal DNS Profile*

Create External DNS Listener

```
python bigip_dns_helper.py --host=[MGMT IP] \  
                           --action create_external_dns_listener  
                           --listener_ip [Listener IP]
```

This performs the steps of *External DNS Listener*

Create Internal DNS Listener

```
python bigip_dns_helper.py --host=[MGMT IP] \
                           --action create_internal_dns_listener
                           --listener_ip [Listener IP]
                           --internal_network [Network/CIDR]
```

This performs the steps of *Internal DNS Listener*

Import iApp Template

```
python iapps/import_template_bigip.py --impl [TCL file] \
                                       --apl [APL File] \
                                       [MGMT IP] \
                                       [Name of Template]
```

This will import the *Application Services Integration iApp*.

Deploy iApp

```
python iapps/deploy_iapp_bigip.py -r [MGMT IP] \
                                     [JSON Input] \
                                     --strings pool__addr=[Virtual Server IP] \
                                     --pool_members=0:[Member IP]:[Member_
↵Port]:0:1:[Priority Group]:enabled:none,\
                                     0:[Member IP]:[Member_
↵Port]:0:1:[Priority Group]:enabled:none
```

This will deploy an iApp. This is modified from the Application Services Integration iApp GitHub scripts to allow the specification of “strings” and “pool_members” from the CLI.

This performs the steps from *LTM Configuration*

Create DNS Virtual Server

```
python bigip_dns_helper.py --host=[MGMT IP] \
                           --action create_vs \
                           --vip [LTM VS IP]:[LTM VS Port] \
                           --vip_translate [LTM VS External IP]:[LTM VS External Port]
                           --vs_name [DNS VS Name]
                           --server_name [Server Name (BIG-IP Device)]
```

This performs the steps from *Step 1: Virtual Servers*

Create DNS Pools

```
python bigip_dns_helper.py --host=[MGMT IP] \
                           --action create_pool \
                           --name [Pool Name]
```

This performs the steps from *Step 2: Pools*. The pool is configured with a Topology LB method.

Create DNS Wide IP

```
python bigip_dns_helper.py --host=[MGMT IP] \
                           --action create_wideip \
                           --name [DNS Name] --pool [DNS Pool #1],[DNS Pool #2]
```

This performs the steps from *Step 3: Wide IPs*. The Wide IP is configured with a Topology LB method.

Create Topology Regions

```
python bigip_dns_helper.py --host [MGMT IP] \
                           --action create_region \
                           --name [Region Name] --internal_network [Subnet #1],
➔[Subnet #2]
```

This performs the steps from *Step 4: Topology Regions*.

Create Topology Records

```
python bigip_dns_helper.py --host [MGMT IP]
                           --action create_topology_record
                           --name [Topology Record]
```

This performs the steps from *Step 5: Topology Records*. An example of a topology record: “ldns: region /Common/internal_network server: pool /Common/internal_pool”

During Lab 3 we will utilize the F5 Python SDK to script the steps that were previously performed manually. The Application Services iApp will also be leveraged to provide Service Catalog of L4-L7 services.

CHAPTER 16

Welcome to the Lab 4 Lab Guide

This Lab will show how Jenkins as a build engine can help automate BIG-IP deployments. The automation will be done in UDF 2.0

This Lab will perform following tasks:

1. License two BIG-IP
2. Reset the config
3. deploy the udf.sh script from lab3 to provision BIG-IP DNS and LTM, create DNS clusters, join LTM and DNS, deploy iapps and deploy BIG-IP DNS configuration

This will be done without touching the BIG-IPs after the default config is loaded.

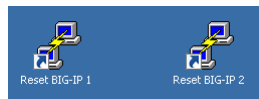
The goal of this lab is to demonstrate how complex automation tasks can be hidden behind a generic automation engine. In this case the deployment is in UDF 2.0. Running through the Lab demonstrates how easy it is to change the destination to AWS or Azure or private cloud environments with keeping the shim layer unchanged.

Step 0 - Restoring the BIG-IP Configuration

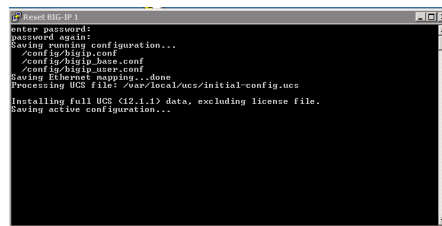
This step is to cleanup the BIG-IP config that was created in Lab 2 and 3. RDP into the Windows jump host.

Reset both BIG-IP to be the same state as after Lab 1.

Find the “Resetting” links on the Desktop.



Double-click on both of these and you should see a window appear briefly like the following.



```
Reset BIG-IP 1
enter password:
password again:
Saving running configuration...
/conf/bigip.conf
/conf/bigip_base.conf
/conf/bigip_user.conf
Saving Ethernet mapping...done
Processing UCS file: /var/local/ucs/initial-config.ucs
Installing full UCS (12.1.1) data, excluding license file.
Saving active configuration...
```

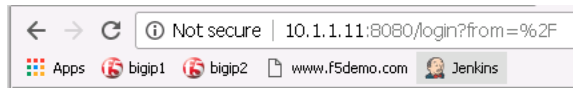
Verify that you no longer see the changes that were previously deployed.

CHAPTER 18

Step 1 - login into Jenkins

This step demonstrates how to login into Jenkins and find the deployment folder.

After both BIG-IP are active again open Chrome in the RDP Session and click on the Jenkins link.



Login to the jenkins server. The credentials are on the RDP Desktop in the “Jenkins credentials.txt” file.

After login to the Jenkins Web interface, please note the UDF-demo folder.

[add description](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Built On
		UDF-demo	N/A	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

The UDF-folder contains a collection of Projects/Jobs that can be started or build by Jenkins. Click the UDF-demo folder. In the UDF demo folder there are several Projects from job 0 to job 3.

UDF-demo

[add description](#)

All						
S	W	Name ↓	Last Success	Last Failure	Last Duration	Built On
		Job 0 - clone git repository	N/A	N/A	N/A	
		Job 1a - license bigip-1	N/A	N/A	N/A	
		Job 1b - license bigip-2	N/A	N/A	N/A	
		Job 2a - reset big-ip 1	N/A	N/A	N/A	
		Job 2b - reset big-ip 2	N/A	N/A	N/A	
		Job 3 - launch-udf.sh	N/A	N/A	N/A	

 Icon: [S](#) [M](#) [L](#)
[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

In the next steps these Jobs will be build and the result verified.

Step 2 - deploying F5 config via automation scripts using Jenkins

The config deployment is split into multiple steps.

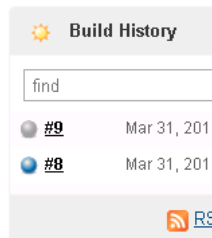
Job 0

The task of Job 0 is to download the repository for this lab from github to the Jenkins Servers local HDD. This enables customer to test and deploy services always from the latest stable or develop cycle.

In the Jenkins GUI click on “Job 0 - clone git repository”. In order to run the project click on the left side the “Build Now” link.

The screenshot shows the Jenkins web interface for a project named "Project Job 0 - clone git repository". On the left is a sidebar with navigation links: Up, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Move. The main area displays the project name, its full name "UDF-demo/Job 0 - clone git repository", and links to "Workspace" and "Recent Changes". Below this is a "Permalinks" section. At the bottom left, there is a "Build History" widget with a search bar containing the text "find", a "trend" link, and RSS feeds for "all" and "failures".

Note that at the left side the “Building History” list adds a new build.



Click at the new build number. In the next screen there is a link called “Console Output”

Console Output

```
Started by user f5admin at jenkins
[EnvInject] - Loading node environment variables.
Building in workspace /var/lib/jenkins/workspace/udf-demo
[udf-demo] $ /bin/sh -xe /tmp/hudson1274247318674520968.sh
+ pwd
/var/lib/jenkins/workspace/udf-demo
+ rm -rf f5-dns-automation-demo-12-1-x
+ git clone -b develop https://github.com/chen23/f5-dns-automation-demo-12-1-x
Cloning into 'f5-dns-automation-demo-12-1-x'...
Finished: SUCCESS
```

Click on “Console Output”

In the middle of the screen the console output is displayed.

Question How is the deployment status?

In the task list click on UDF-demo to return to the folder



Job 1a and 1b

The task of Job 1a and Job 1b is to license the BIG=IP instances.

there are multiple ways to build a project. In this case click on the “Build On” button at the right side of the screen for Job 1a and Job 1b.



Question What is the console status after the Job completed?

Job 2a and 2b

The task of Job 2a and 2b is to send the reset scripts from lab 2 and lab3 to the BIG-IP’s.

Build the jobs 2a and 2b. Check the console output for both Projects and their status responses for success.

Question What is the console status after the Job completed?

Job 3

In Job 1 and 2 the BIG-IP were prepared to receive the configuration. Job 3 deploys the udf.sh script from lab 3 to both BIG-IP’s. After this deployment the BIG-IP’s are ready to serve the service.

Click “Build On” and monito the console output.

Question What is the console status after the Job completed?

In Lab 3 we launched the automation scripts via ssh/shell scripts. During this lab we will utilize Jenkins to perform the automation steps.

Jenkins can provide a more standard way of deploying and monitoring the health of automated workflows.

The goal of this lab is to demonstrate how complex automation tasks can be hidden behind a generic automation engine. In this case the deployment is in UDF 2.0. Running through the Lab demonstrates how easy it is to change the destination to AWS or Azure or private cloud environments with keeping the shim layer unchanged.

CHAPTER 20

Jenkins Pipeline

During the prior lab *Welcome to the Lab 4 Lab Guide* Jenkins was used to provide a frontend tool to launch the automation scripts from *F5 Python SDK*.

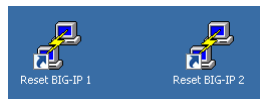
This lab will leverage Jenkins Pipeline to create a workflow that will validate the individual steps that are executing in a more declarative manner.

Restoring the BIG-IP Configuration

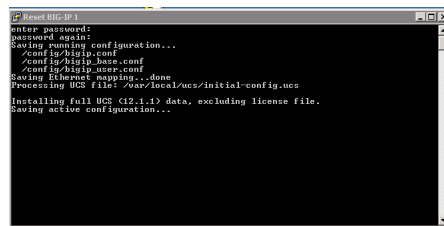
This step is to cleanup the BIG-IP config that was created in prior Labs. RDP into the Windows jump host.

Reset both BIG-IP to be the same state as after Lab 1.

Find the “Resetting” links on the Desktop.



Double-click on both of these and you should see a window appear briefly like the following.

A screenshot of a terminal window titled 'Reset BIG-IP 1'. The window has a black background with white text. The text shows a series of commands and their outputs: 'enter password:', 'password admin:', 'Saving running configuration...', 'config/bigip.conf', 'config/bigip_base.conf', 'config/bigip_user.conf', 'Saving Ethernet mapping...done', 'Processing UCS file: /var/local/ucs/initial-config.ucs', 'Installing full UCS (12.1.1) data, excluding license file.', and 'Saving active configuration...'.

```
Reset BIG-IP 1
enter password:
password admin:
Saving running configuration...
config/bigip.conf
config/bigip_base.conf
config/bigip_user.conf
Saving Ethernet mapping...done
Processing UCS file: /var/local/ucs/initial-config.ucs
Installing full UCS (12.1.1) data, excluding license file.
Saving active configuration...
```

Verify that you no longer see the changes that were previously deployed.

CHAPTER 22

Creating a pipeline

Jenkins allows you to put together a collection of actions, branch on conditions, and measure the results.

This can be put together to put together a series of stages.

- Stage 1: Enable DNS Sync
- Stage 2: Cluster DNS Devices
- Stage 3: Add additional DNS Configuration
- Stage 4: Import App Svcs iApp Template
- Stage 5: Deploy App Svcs iApp Template
- Stage 6: Add DNS Configuration
- Stage 7: Verify Configuration

To describe these stages we can create a “Jenkinsfile” that will instruct Jenkins to execute these stages.

The following will pull down a copy of the scripts used in this lab and execute the steps to create a DNS cluster.

```
stage('clone git repo') {
    node {
        git url: 'https://github.com/f5devcentral/f5-dns-automation-demo-12-1-x.git',
        ↪branch: 'master'
    }
}

stage('enable dns sync') {
    node {
        dir ('lib') {
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
            ↪action enable_sync'
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
            ↪action add_datacenter --datacenter ' + params.dc1 + ' '
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
            ↪action add_datacenter --datacenter ' + params.dc2 + ' '
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
            ↪action add_server --datacenter ' + params.dc1 + ' --server_name bigip1 --server_ip=
            ↪' + params.bigip1_selfip + ' '
```

```

        sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action add_server --datacenter ' + params.dc2 + ' --server_name bigip2 --server_ip=
↪' + params.bigip2_selfip + ''
        sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action save_config'
    }
}
stage('gtm add') {
    node {
        dir ('lib') {

            sh 'sleep 3'
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action gtm_add --peer_host=' + params.bigip1 + ' --peer_selfip ' + params.bigip1_
↪selfip + ''
            sh 'sleep 3'

        }
    }
}
stage('additional dns setup') {
    node {
        dir ('lib') {

            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action create_dns_cache'
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action create_dns_cache'
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action create_external_dns_profile'
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action create_external_dns_profile'
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action create_internal_dns_profile'
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action create_internal_dns_profile'
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action create_external_dns_listener --listener_ip ' + params.bigip1_dns_listener + '
↪'
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action create_external_dns_listener --listener_ip ' + params.bigip2_dns_listener + '
↪'

            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action create_internal_dns_listener --listener_ip ' + params.bigip1_dns_listener +
↪' --internal_network ' + params.internal_network + ''
            sh 'python bigip_dns_helper.py --host=' + params.bigip2 + ' --
↪action create_internal_dns_listener --listener_ip ' + params.bigip2_dns_listener +
↪' --internal_network ' + params.internal_network + ''
            sh 'python bigip_dns_helper.py --host=' + params.bigip1 + ' --
↪action save_config'
            sh 'sleep 3'

        }
    }
}

```

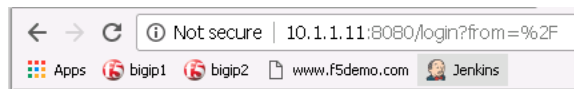
Note that in this code example that we are no longer using hardcoded IP addresses. These are now parameters that can be input when the pipeline is executed.

The full code can be found on [GitHub](#).

CHAPTER 23

Launching the pipeline

After both BIG-IP are active again open Chrome in the RDP Session and click on the Jenkins link.

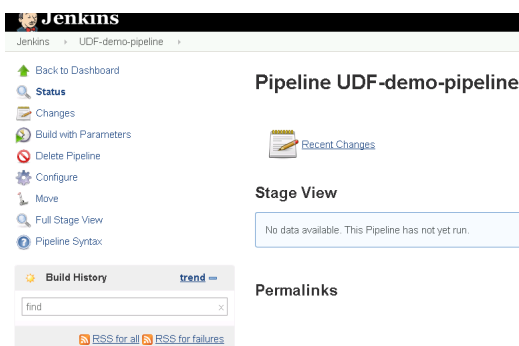


Login to the Jenkins server. The credentials are on the RDP Desktop in the “Jenkins credentials.txt” file.

After login to the Jenkins Web interface, please note the UDF-demo-pipeline folder.



Click on “UDF-demo-pipeline” and you should see.



In order to run the project click on the left side the “Build with Parameters” link.

lab5/udf-demo-pipeline-build-parameters.png

In previous labs all the input parameters were statically defined making the solution only usable with a specific network. By utilizing parameters in Jenkins we are creating a solution that can be deployed more dynamically.

Scroll to the bottom of the page and find the “Build” button and click on it.

lab5/udf-demo-pipeline-build-parameters-button.png

While the build is running you will see.

The screenshot shows the Jenkins interface for the 'UDF-demo-pipeline'. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build with Parameters, Delete Pipeline, Configure, Move, Full Stage View, and Pipeline Syntax. The main area displays the 'Pipeline UDF-demo-pipeline' with a 'Recent Changes' section showing a change on Apr 04 at 12:36. The 'Stage View' section shows a progress bar for the current build, with average stage times: 2s for 'clone git repo' and 11s for 'enable dns sync'. The 'Permalinks' section shows the last build (#12) completed 44 ms ago.

When the pipeline is complete you will see:

The screenshot shows the Jenkins interface for the 'UDF-demo-pipeline' after a successful build. The left sidebar is the same as the previous screenshot. The main area displays the 'Pipeline UDF-demo-pipeline' with a 'Recent Changes' section showing a change on Apr 04 at 12:36. The 'Stage View' section shows a progress bar for the current build, with average stage times: 2s for 'clone git repo', 12s for 'enable dns sync', 53s for 'gtm add', 12s for 'additional dns setup', 2s for 'Import App Services Template', 57s for 'Deploy App Services Template', 21s for 'DNS Configuration', and 1min 3s for 'Verify Configuration'. The 'Permalinks' section shows the last build (#12) completed 44 ms ago.

clone git repo	enable dns sync	gtm add	additional dns setup	Import App Services Template	Deploy App Services Template	DNS Configuration	Verify Configuration
2s	12s	53s	12s	2s	57s	21s	1min 3s

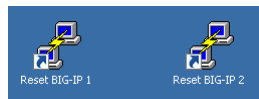
CHAPTER 24

Failing Tests

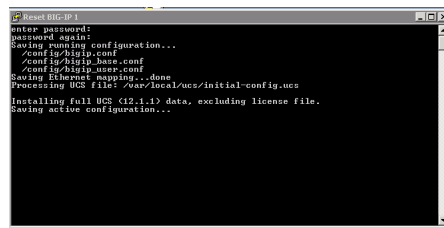
In the previous exercise we deployed a successful deployment. In this exercise we will purposely break the pipeline.

Reset both BIG-IP to be the same state as after Lab 1.

Find the “Resetting” links on the Desktop.

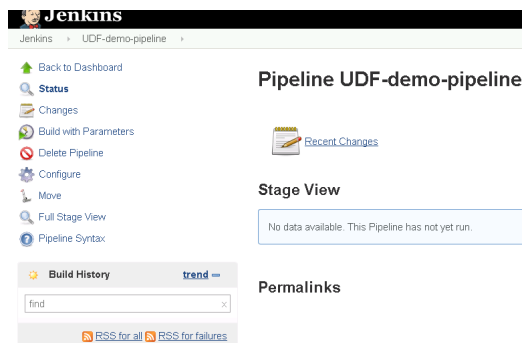


Double-click on both of these and you should see a window appear briefly like the following.



Verify that you no longer see the changes that were previously deployed.

Go back to the ‘UDF-demo-pipeline’ page.



This time click on the “Configure” link.

Find the “Pipeline” section.

The current pipeline is configured to pull down a copy of Jenkinsfile from the f5devcentral GitHub repository. We are going to download a local copy of this file and modify it to simulate a failure.

Select the pulldown for “Pipeline script from SCM” and change to “Pipeline Script”.

Go to [GitHub](#) and copy the text into your clipboard.

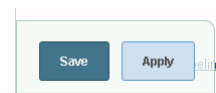
Paste the contents into the script text area.

Comment out the following lines (around line 100).

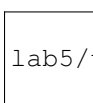
```
//sh 'python bigip_dns_helper.py --host ' + params.bigip1 + ' --action create_
↳ topology_record --name "ldns: region /Common/region_1 server: region /Common/region_
↳ 1"'
//sh 'python bigip_dns_helper.py --host ' + params.bigip1 + ' --action create_
↳ topology_record --name "ldns: region /Common/region_2 server: region /Common/region_
↳ 2"'
//sh 'python bigip_dns_helper.py --host ' + params.bigip2 + ' --action create_
↳ topology_record --name "ldns: region /Common/region_1 server: region /Common/region_
↳ 1"'
//sh 'python bigip_dns_helper.py --host ' + params.bigip2 + ' --action create_
↳ topology_record --name "ldns: region /Common/region_2 server: region /Common/region_
↳ 2"'
```

The result should look something like:

Now click on the Save Button.

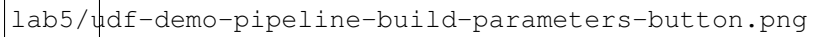


Back on the pipeline page find the “Build with Parameters” link.

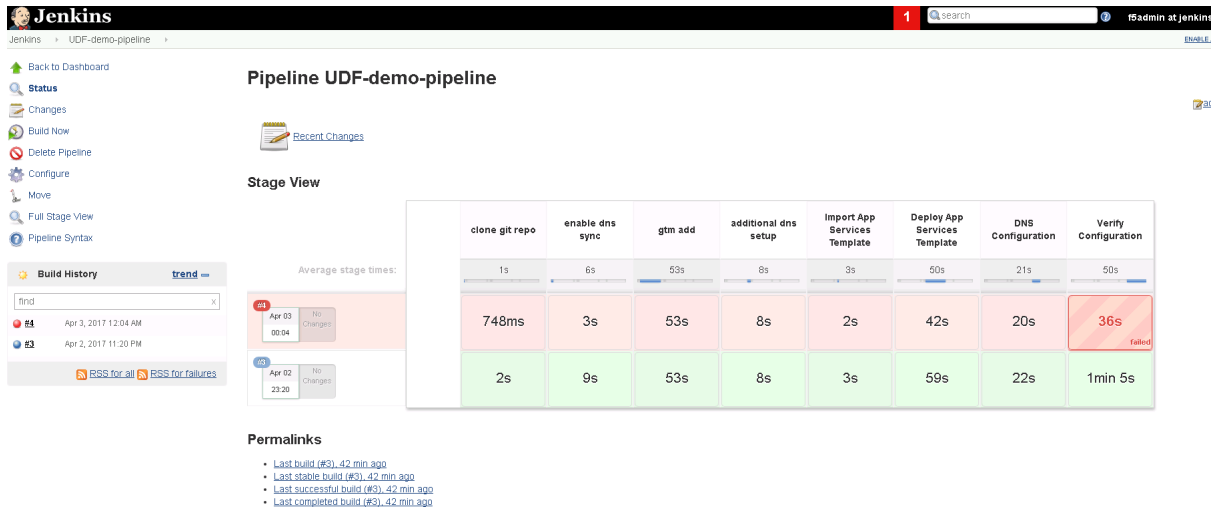


lab5/udf-demo-pipeline-build-parameters.png

Scroll to the bottom of the page and find the “Build” button and click on it.



Once the build completes you should see a failure.



Pipeline UDF-demo-pipeline

Recent Changes

Stage View

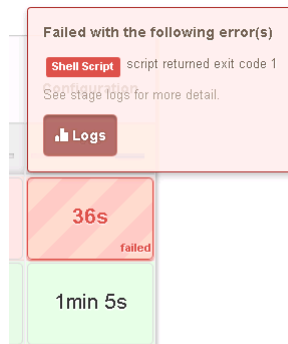
Average stage times:

	clone git repo	enable dns sync	gtm add	additional dns setup	Import App Services Template	Deploy App Services Template	DNS Configuration	Verify Configuration
1s	6s	53s	8s	3s	50s	21s	50s	
748ms	3s	53s	8s	2s	42s	20s	36s failed	
2s	9s	53s	8s	3s	59s	22s	1min 5s	

Permalinks

- Last build (#3), 42 min ago
- Last stable build (#3), 42 min ago
- Last successful build (#3), 42 min ago
- Last completed build (#3), 42 min ago

Hover your mouse over the failure.



Failed with the following error(s)

Shell Script script returned exit code 1

See stage logs for more detail.

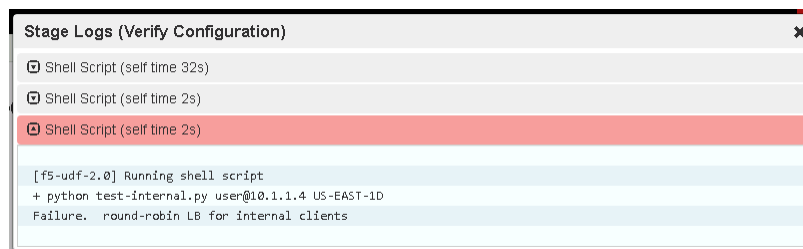
Logs

36s failed

1min 5s

Click on “Logs” to see the detail and expand the failing task.

Hover your mouse over the failure.



Stage Logs (Verify Configuration)

Shell Script (self time 32s)

Shell Script (self time 2s)

Shell Script (self time 2s)

[f5-udf-2.0] Running shell script
+ python test-internal.py user@10.1.1.4 US-EAST-1D
Failure. round-robin LB for internal clients

In this case the script “test-internal.py” exited with a non-zero exit code. This causes Jenkins to treat this as a failure. In this case the script was only expecting to see responses from a single Data Center and instead received responses from both Data Centers due to the lack of topology records.

CHAPTER 25

Indices and tables

- `genindex`
- `search`