Achieving firewall high-availability in Azure with F5

Chris Zhang, 2018-15-06

Background

Due to the lack of Layer 2 functions (e.g. ARP) in public Cloud provider networks, certain firewall vendors recommend achieving Firewall (FW) high-availability (HA) through the use of load balancing.

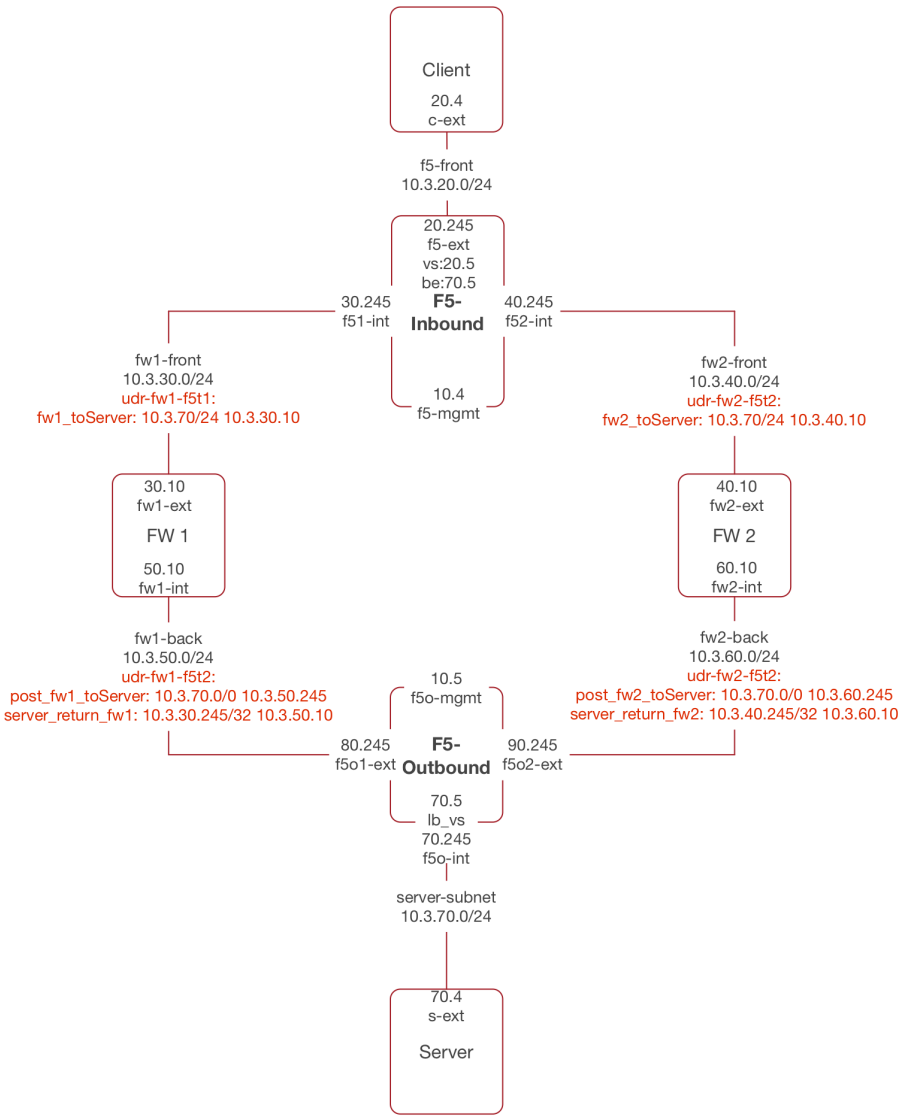
Take Palo Alto (PA) as an example, this article (High Availability Considerations on AWS and Azure | Palo Alto Networks) seems to suggest that load balancing is the only option, when deploying FW HA in Azure.

For inbound HTTPS traffic, it does not make sense to pass that traffic as is to the FW, as the FW can't see it. The way to protect that traffic is using AWAF and AFM.

This article focuses on load balancing FW's, achieving HA, and protect inbound non-HTTPS, as well as outbound traffic. - outbound being traffic originated from internal networks (e.g. user browsers) and destined to the Internet

Inbound

The following diagram depicts a two tier configuration.



The VPC is associated with 10.3.0.0/16 CIDR block.

For the ease of testing, Client is placed within the VPC, however, it represents users coming from the Internet. Traffic flow is shown as below,

1. Client (20.4) -> vs (20.5) Note: vs is the LB VIP fronting a second LB VIP configured on F5-Outbound

2. LTM sends traffic to the backend (70.5), which is a LB VIP configured on F5-Outbound, frontending resources in Server network

To get to the 70.0 subnet, we use a firewall pool, and the LTM selects a firewall from that pool.

```
azureuser@(F5-Inbound)(cfg-sync Standalone)(Active)(/Common)(tmos)# list net route
net route to_Server {
    network 10.3.70.0/24
    pool /Common/fw_pool
}

azureuser@(F5-Inbound)(cfg-sync Standalone)(Active)(/Common)(tmos)# list ltm pool fw_pool
ltm pool fw_pool {
    members {
        10.3.30.10:any {
            address 10.3.30.10
            session monitor-enabled
            state up
        }
        10.3.40.10:any {
            address 10.3.40.10
            session monitor-enabled
            state up
        }
    }
    monitor custom_PING_mon
}
```

3. For traffic to be routed to either FW (e.g. 30.10 or 40.10), we also need to leverage User Defined Route (UDR), an Azure construct. e.g. udr-fw1-inbound and udr-fw2-inbound shown in the diagram

4. Once traffic arrives at a FW, the FW forwards that traffic to the tier 2 LTM (e.g. F5-Outbound).

Again, UDR's must be used, for both outgoing and returning traffic, otherwise Azure will send traffic to the default subnet gateway (e.g. .1 address of each subnet)

5. When traffic (src: 30.245 or 40.245, dst: 70.5) arrives at F5-Outbound. The tier 2 LTM makes a load balancing decision and proxies that traffic to Server.

6. Traffic makes its way back.

Routes back to F5-Inbound must be created on the tier 2 LTM.

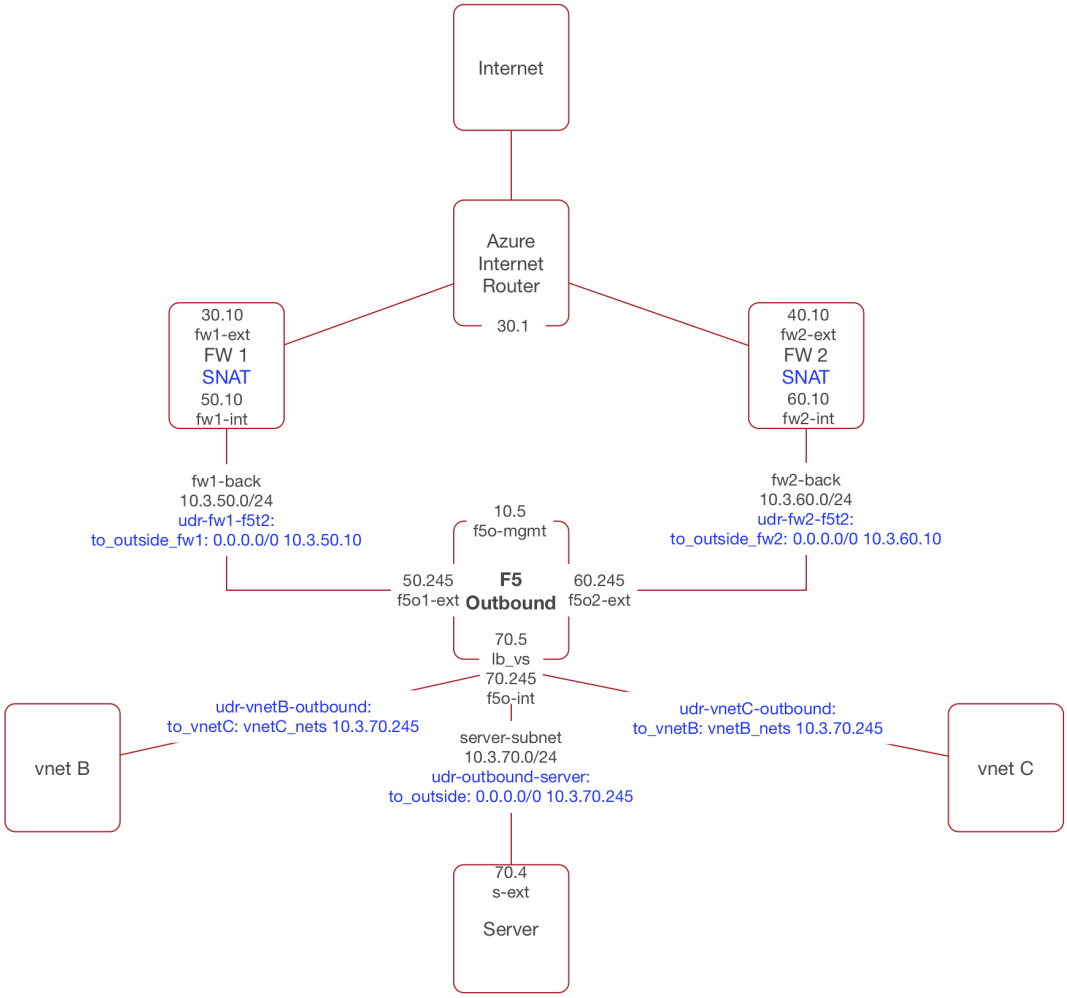
```
net route to_10.3.30.245 {
    gw 10.3.50.10
    network 10.3.30.245/32
}
net route to_10.3.40.245 {
```

```
    gw 10.3.60.10
    network 10.3.40.245/32
}
```

This concludes inbound.

Outbound

For outbound traffic, we will only leverage the tier 2 LTM (e.g. F5-Outbound). Once traffic arrives at the FW's, they get sent to the Internet via Azure default gateway. See diagram below.



Traffic flow is shown as below,

1. Traffic originates from Server (70.4) and destines to the Internet (e.g. 1.2.3.4)

2. UDR must be created to route traffic to F5-Outbound

3. Once traffic arrives at the F5, a FW is selected as the next hop. See F5-Outbound config below.

```
net route to_outside {
    network default
    pool /Common/fw_pool
}

ltm pool fw_pool {
    members {
        10.3.50.10:any {
            address 10.3.50.10
            session monitor enabled
```

```
            session monitor-enabled
            state up
        }
        10.3.60.10:any {
            address 10.3.60.10
            session monitor-enabled
            state up
        }
    }
    monitor gateway_icmp
}
```

Again, watch the UDR's (we will need to create a new route and add to the existing UDR route table, as only one UDR is permitted per subnet)

4. Once traffic arrives at the FW, the FW sends it off to the Internet. To do this, SNAT must be enabled on the FW.

For Linux, the following works.

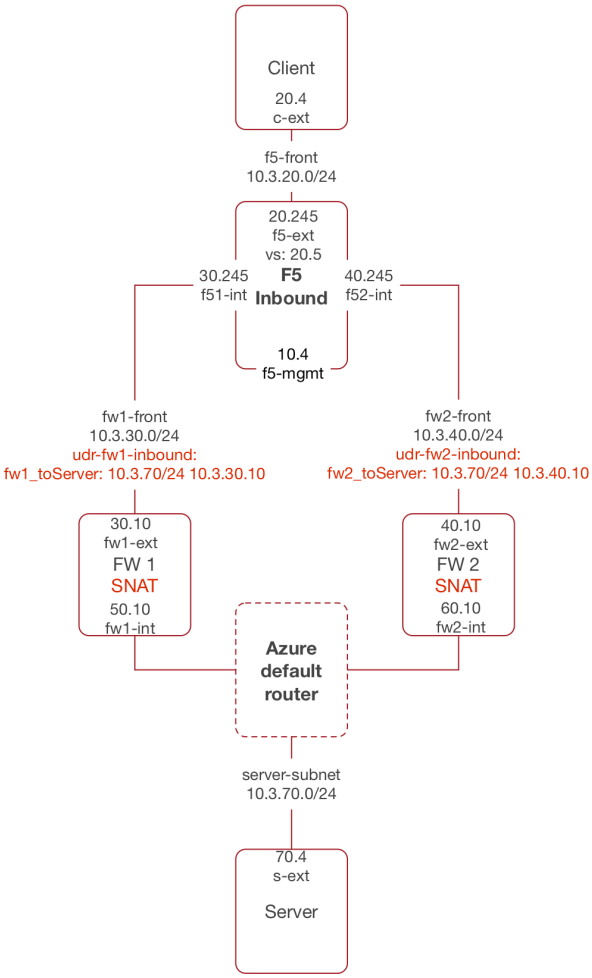
```
#FW1
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.3.30.10

#FW2
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.3.40.10
```

Internal traffic traversing different vnet's also go through the load balanced FW's. FW's routing table will send traffic to the destination, using Azure system default gateway. Notice that SNAT must be enabled.

Remarks

It's also possible to have Inbound working on a single F5. The FW's will need to SNAT inbound traffic in this case.

Diagrams is shown below.



Make sure to enable IP forwarding on all interfaces that forward traffic. The setting is under Interface, IP Configurations, within the Azure portal.

Azure CLI

The following CLI completed 95% of this build.

```
az network public-ip create --resource-group cz-lb-fw-in_out_bound --name Client-pub-ip
az network nic create --resource-group cz-lb-fw-in_out_bound --name c-ext --vnet-name cz-A1 --subnet

az network public-ip create --resource-group cz-lb-fw-in_out_bound --name F5-Inbound-mgmt-ip
az network nic create --resource-group cz-lb-fw-in_out_bound --name f5-mgmt --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name f5-ext --vnet-name cz-A1 --subnet

az network nic create --resource-group cz-lb-fw-in_out_bound --name f51-int --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name f52-int --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name fw1-ext --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name fw2-ext --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name fw1-int --vnet-name cz-A1 --subne

az network nic create --resource-group cz-lb-fw-in_out_bound --name fw2-int --vnet-name cz-A1 --subne

az network public-ip create --resource-group cz-lb-fw-in_out_bound --name F5-Outbound-mgmt-ip
az network nic create --resource-group cz-lb-fw-in_out_bound --name f5o-mgmt --vnet-name cz-A1 --subn
```

```
az network nic create --resource-group cz-lb-fw-in_out_bound --name f5o1-ext --vnet-name cz-A1 --subn

az network nic create --resource-group cz-lb-fw-in_out_bound --name f5o2-ext --vnet-name cz-A1 --subn

az network nic create --resource-group cz-lb-fw-in_out_bound --name f5o-int --vnet-name cz-A1 --subne

az network public-ip create --resource-group cz-lb-fw-in_out_bound --name Server-pub-ip
az network nic create --resource-group cz-lb-fw-in_out_bound --name s-ext --vnet-name cz-A1 --subnet

# Add nics (only use when missing interfaces)
az vm nic add --resource-group cz-lb-fw-in_out_bound --vm-name F5-Inbound --nics f51-int f52-int

# FW builds
az vm create --image ubuntults --resource-group cz-lb-fw-in_out_bound --name FW1 --admin-username azu

az vm create --image ubuntults --resource-group cz-lb-fw-in_out_bound --name FW2 --admin-username azu

#F5 builds

az vm image accept-terms --urn f5-networks:f5-big-ip-best:f5-bigip-virtual-edition-best-byol:latest

az vm create --image f5-networks:f5-big-ip-best:f5-bigip-virtual-edition-best-byol:latest --resource-

az vm create --image f5-networks:f5-big-ip-best:f5-bigip-virtual-edition-best-byol:latest --resource-

#Client and server builds
az vm create --image ubuntults --resource-group cz-lb-fw-in_out_bound --name Server --admin-username

az vm create --image ubuntults --resource-group cz-lb-fw-in_out_bound --name Client --admin-username

# Update nics to enable forwarding
az network nic update --resource-group cz-lb-fw-in_out_bound --name fw1-ext --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fw1-int --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fw2-ext --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fw2-int --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fwo1-ext --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fwo2-ext --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fwo1-int --ip-forwarding true
az network nic update --resource-group cz-lb-fw-in_out_bound --name fwo2-int --ip-forwarding true

# Create route-tables, associate to subnets and create routes
az network route-table create --name udr-fw1-inbound --resource-group cz-lb-fw-in_out_bound --locatio
az network vnet subnet update --resource-group cz-lb-fw-in_out_bound --vnet-name cz-A1 --name fw1-fro
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw1

az network route-table create --name udr-fw2-inbound --resource-group cz-lb-fw-in_out_bound --locatio
az network vnet subnet update --resource-group cz-lb-fw-in_out_bound --vnet-name cz-A1 --name fw2-fro
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw2

az network route-table create --name udr-fw1-f5t2 --resource-group cz-lb-fw-in_out_bound --location e
az network vnet subnet update --resource-group cz-lb-fw-in_out_bound --vnet-name cz-A1 --name fw1-bac
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw1
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw1
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw1

az network route-table create --name udr-fw2-f5t2 --resource-group cz-lb-fw-in_out_bound --location e
az network vnet subnet update --resource-group cz-lb-fw-in_out_bound --vnet-name cz-A1 --name fw2-bac
```

```
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw2
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw2
az network route-table route create --resource-group cz-lb-fw-in_out_bound --route-table-name udr-fw2
```

Updates:

Alternative approaches have also been developed, and can be used to support the following use cases:

1. vnet to vnet traffic be load balanced to FW's - a hub and spoke topology is used where the hub vnet hosts the services (F5's and FW's), and spoke vnets host consumer networks (e.g. DMZ, internal)
2. vnet to Internet traffic be load balanced to FW's - traffic originated from spoke vnets, destined to Internet, traverses the hub vnet, where FW's are load balanced
3. Inbound Intertnet to vnet's traffic be load balanced to FW's - e.g. RDP to internal machines, or application load balancing
4. SNAT is optional for vnet to vnet, as well as inbound Internet traffic - The F5 can retain client IP addresses and pass traffic as is to FW's (e.g. No address translation on the F5)

Due to variations in requirements and their associated complexities, it's not possible to share all the details here.

If you have a particular requirement, please reach out to your Sales team!

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com