Configuring ADVPN in FortiOS 5.4 - Redundant hubs (Expert)

This recipe is a followup to the ADVPN basic recipe. As we have seen in the base configuration, ADVPN provides the means for spokes to automatically establish VPN sessions in a peer-to-peer fashion without the hub being involved in data forwarding. This recipe explores how redundancy can be achieved for the hub functions within an ADVPN environment.

As ADVPN leverages BGP, a redundant hub configuration involves having a second hub device that will also act as a route reflector for the topology. That redundant hub device could be located in a geographically distinct site, in the same datacenter or even as an additional VDOM on the same physical device. Each ADVPN topology generally benefits from residing in its own VDOM or physical device.

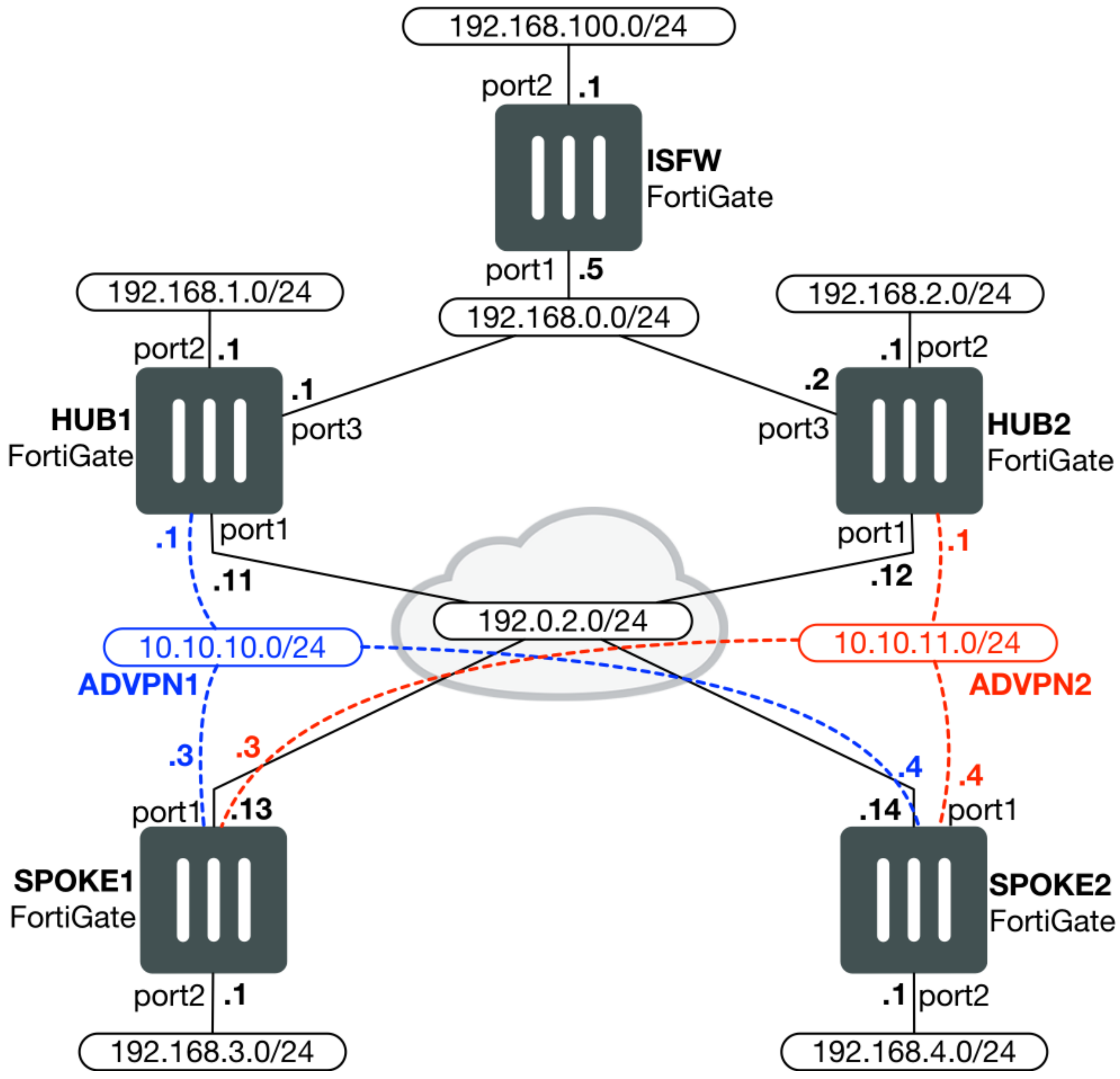
ADVPN topologies can be regrouped under 3 deployment scenarios. While each of these scenario can benefit from FGCP for physical device redundancy, only the dual hub scenario provides logical redundancy for a given group of spokes (which we will term a “region”):

- **Single hub**: this offers no logical redundancy. A single hub with a single ADVPN IPSec topology running in a single BGP autonomous system.
- **Dual hub**: this offers logical redundancy. Each additional hub role runs on a distinct VDOM or physical device. Each additional hub runs a topology within the same BGP autonomous system, however hubs (route reflectors) are unaware of each other.
- **Dual regions**: this offers no logical redundancy. This reflects a configuration in which we have distinct BGP autonomous systems, each with distinct hubs which are interconnected and have shortcut forwarding enabled between the hubs. Spokes are only connected to the hub of their own region. While this scenario can be combined with “dual hub” to effectively create logically redundant regions, it is not in itself a redundancy technique as spokes are not peered with more than a single BGP route reflector. It is redundant only in the sense that one hub being unavailable would result in only partial availability issues.

More information on dual region setups can be obtained here:
https://kb.fortinet.com/kb/microsites/search.do?cmd=displayKC&docType=kc&externalId=FD39360

When multiple ADVPN hubs are deployed, careful consideration should be given to ensure that traffic will use the same forward and reverse path due to the issues that can arise from symmetric routing in a firewall environment. While a layered model with a stateless ADVPN topology can be leveraged for ECMP load balancing, in this recipe we will explore a simpler version of redundancy in which one hub is designated as primary, with the secondary hub's topology being only leveraged upon failure of the first hub.

Our topology is the following:



When compared with our original ADVPN article, this topology sees the addition of a hub unit. Our example assumes WAN connectivity between the hub sites, which could be replaced with additional hub-to-hub VPN links. We are also adding an additional device to this scenario, termed “ISFW”, which we use to advertise a route in OSPF in order to test connectivity.

A few topology notes are in order before we go over the configuration:

- The topology makes use of a second BGP route reflector however neither route reflector is aware of the other, as this introduces complexity related to route advertisement.
- The topology prioritizes **ADVPN1**.
- The topology assumes that both hubs and spokes have a single ISP link. In the case of multiple ISPs at either hub or spoke, a decision must be made as to which ADVPN topology is used for each ISP and how fully meshed the overlay VPNs must be. A common scenario is to have dual ISPs for the spokes where one ISP is used for **ADVPN1** and the second is used for **ADVPN2**, in which case the configuration is no different from the topology we use in this article.

- This example however does maintain that **ADVPN2**'s hub direct attached subnets are being sent towards **ADVPN2** directly. All other hub-destined traffic will cross **ADVPN1**.

Things are going to get slightly more complex that a standard ADVPN configuration here, however this article will try to clarify the configurations made as they are listed below.

1. Configure HUB1

Configure phase 1 parameters

Note that we added some DPD settings to ensure DPD detects IPSec failures much faster than usual in order for convergence to occur.

```
config vpn ipsec phase1-interface
    edit "ADVPN"
        set type dynamic
        set interface "port1"
        set proposal aes128-sha1
        set add-route disable
        set dhgrp 14
        set auto-discovery-sender enable
        set psksecret somereallysecuresauuuuce
        set dpd-retrycount 3
        set dpd-retryinterval 2
        set dpd on-idle
    next
end
```

Configure phase2 parameters

```
config vpn ipsec phase2-interface
    edit "ADVPN-P2"
        set phase1name "ADVPN"
        set proposal aes128-sha1
    next
end
```

Configure the tunnel interface IP

```
config system interface
    edit "ADVPN"
        set vdom "root"
        set ip 10.10.10.1 255.255.255.255
        set type tunnel
        set remote-ip 10.10.10.254
        set interface "port1"
    next
end
```

Configure prefix-lists and route-maps

We use a first route-map to select which connected networks to advertise.

Our second route-map ensures to clear the weight of redistributed OSPF routes to 0 to ensure they do not become sticky and also assigns OSPF redistributed routes a local preference of 50.

```
config router prefix-list
    edit "PF_REDIS_CONNECTED"
        config rule
            edit 1
                set prefix 192.168.1.0 255.255.255.0
                unset ge
                unset le
            next
            edit 2
                set prefix 192.168.0.0 255.255.255.0
                unset ge
                unset le
            next
        end
    next
end
config router route-map
    edit "RM_REDIS_CONNECTED"
        config rule
            edit 1
                set match-ip-address "PF_REDIS_CONNECTED"
            next
            edit 2
                set action deny
            next
        end
    next
    edit "RM_OSPF_TO_BGP"
        config rule
            edit 1
                set set-local-preference 50
                set set-weight 0
            next
        end
    next
end
```

Note regarding BGP weight: *routes that are sourced by a given BGP instance (either by network statements or through redistribution) are always by default marked with a weight of 32768. When a route stops being learnt from a spoke over iBGP and instead is learnt through OSPF from the other hub, that route will be inserted with a high weight and will no longer be displaced when the spoke reestablishes its BGP adjacency. As this mechanism does not account for IGP backdoor connectivity as our topology does with OSPF, we have to break this mechanism in order to ensure convergence favours routes coming directly from the spokes.*

Configure iBGP and route-reflection

Timers for BGP are modified to accelerate convergence. Care should however be given to ensure that the combination of faster DPD and faster BGP timers does not result in issues when the topology deployed has a very large number of spokes.

We have route-map filtering configured for the redistribution of both connected networks and OSPF-learnt networks.

The admin distance of iBGP routes is also modified to be preferred over OSPF. While not a common configuration, it is however crucial in our usage as both HUB devices must always prefer their iBGP routes to anything learnt over OSPF. This is supplemental to our previous “route weight” route-map fix.

```
config router bgp
    set as 65000
    set router-id 192.168.1.1
    set distance-internal 100
    config neighbor-group
        edit "ADVPN"
            set advertisement-interval 10
            set next-hop-self enable
            set remote-as 65000
            set keep-alive-timer 2
            set holdtime-timer 6
            set route-reflector-client enable
        next
    end
    config neighbor-range
        edit 1
            set prefix 10.10.10.0 255.255.255.0
            set neighbor-group "ADVPN"
        next
    end
    config redistribute "connected"
        set status enable
        set route-map "RM_REDIS_CONNECTED"
    end
    config redistribute "ospf"
        set status enable
        set route-map "RM_OSPF_TO_BGP"
    end
end
```

Configure OSPF

This is a standard OSPF configuration with BGP redistribution. Note that we are bumping the redistribution metric for our second HUB as documented further along in this article.

```
config router ospf
    set router-id 192.168.0.1
    config area
        edit 0.0.0.0
        next
    end
    config network
        edit 1
            set prefix 192.168.0.0 255.255.0.0
        next
    end
    config redistribute "bgp"
        set status enable
    end
end
```

Configure firewall policies

```
config firewall policy
    edit 0
        set name "OUT ADVPN"
        set srcintf "port1" "port2" "port3" "ADVPN"
        set dstintf "port1" "port2" "port3" "ADVPN"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
end
```

2. Configure HUB2

Configure phase 1 parameters

This part is identical to HUB1.

```
config vpn ipsec phase1-interface
    edit "ADVPN"
        set type dynamic
        set interface "port1"
        set proposal aes128-sha1
        set add-route disable
        set dhgrp 14
        set auto-discovery-sender enable
        set psksecret somereallysecuresauuuuce
        set dpd-retrycount 3
        set dpd-retryinterval 2
        set dpd on-idle
    next
end
```

Configure the phase2 parameters

This part is identical to HUB1.

```
config vpn ipsec phase2-interface
    edit "ADVPN-P2"
        set phase1name "ADVPN"
        set proposal aes128-sha1
    next
end
```

Configure the tunnel interface IP

Our second topology's IP subnet is configured.

```
config system interface
    edit "ADVPN"
        set vdom "root"
        set ip 10.10.11.1 255.255.255.255
        set type tunnel
        set remote-ip 10.10.11.254
        set interface "port1"
    next
end
```

Configure prefix-lists and route-maps

Other than modifying the connected subnets advertised, we are using a local preference of 25 for OSPF routes redistributed by BGP on our second hub. This ensures HUB1 is prioritized for reaching those subnets.

```
config router prefix-list
    edit "PF_REDIS_CONNECTED"
        config rule
            edit 1
                set prefix 192.168.2.0 255.255.255.0
                unset ge
                unset le
            next
            edit 2
                set prefix 192.168.0.0 255.255.255.0
                unset ge
                unset le
            next
        end
    next
end
config router route-map
    edit "RM_REDIS_CONNECTED"
        config rule
            edit 1
                set match-ip-address "PF_REDIS_CONNECTED"
            next
            edit 2
                set action deny
            next
        end
    next
    edit "RM_OSPF_TO_BGP"
        config rule
            edit 1
                set set-local-preference 25
                set set-weight 0
            next
        end
    next
end
```

Configure iBGP and route-reflection

Our second topology uses the same AS number therefore most of the configuration is identical except for router-id and subnets.

```
config router bgp
    set as 65000
    set router-id 192.168.2.1
    set distance-internal 100
    config neighbor-group
        edit "ADVPN"
            set advertisement-interval 10
            set next-hop-self enable
            set remote-as 65000
            set keep-alive-timer 2
            set holdtime-timer 6
            set route-reflector-client enable
        next
    end
    config neighbor-range
        edit 1
            set prefix 10.10.11.0 255.255.255.0
            set neighbor-group "ADVPN"
        next
    end
    config redistribute "connected"
        set status enable
        set route-map "RM_REDIS_CONNECTED"
    end
    config redistribute "ospf"
        set status enable
        set route-map "RM_OSPF_TO_BGP"
    end
end
```

Configure OSPF

HUB2's OSPF configuration is identical to HUB1 except for a redistribution metric which is bumped to ensure HUB2 is not the preferred egress for the datacenter to use.

```
config router ospf
    set router-id 192.168.0.2
    config area
        edit 0.0.0.0
        next
    end
    config network
        edit 1
            set prefix 192.168.0.0 255.255.0.0
        next
    end
    config redistribute "bgp"
        set status enable
        set metric 5
    end
end
```

Configure firewall policies

```
config firewall policy
    edit 0
        set name "OUT ADVPN"
        set srcintf "port1" "port2" "port3" "ADVPN"
        set dstintf "port1" "port2" "port3" "ADVPN"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
end
```

3. Configure the Spoke FortiGates

Configure phase 1 parameters

Compared with a single hub setup, we add a second ADVPN connection towards our second hub.

Note that we are configuring only one of the spokes in this example – the parameters that need to change for each spoke are highlighted in red.

```
config vpn ipsec phase1-interface
 edit "ADVPN"
  set interface "port1"
  set proposal aes128-sha1
  set add-route disable
  set dhgrp 14
  set auto-discovery-receiver enable
  set remote-gw 192.0.2.11
  set psksecret ***
  set dpd-retrycount 3
  set dpd-retryinterval 2
  set dpd on-idle
 next
 edit "ADVPN2"
  set interface "port1"
  set proposal aes128-sha1
  set add-route disable
  set dhgrp 14
  set auto-discovery-receiver enable
  set remote-gw 192.0.2.12
  set psksecret ***
  set dpd-retrycount 3
  set dpd-retryinterval 2
  set dpd on-idle
 next
end
```

Configure the phase2 parameters

Spokes benefit from having auto-negotiate enabled in order to ensure tunnels come up given the usage of dynamic routing to learn actual protected traffic subnets.

```
config vpn ipsec phase2-interface
    edit "ADVPN-P2"
        set phase1name "ADVPN"
        set proposal aes128-sha1
       set auto-negotiate enable
    next
    edit "ADVPN2-P2"
        set phase1name "ADVPN2"
        set proposal aes128-sha1
       set auto-negotiate enable
    next
end
```

Configure the tunnel interface IPs

Each ADVPN topology needs a hardcoded IP configured.

```
config system interface
    edit "ADVPN"
        set vdom "root"
        set ip 10.10.10.3 255.255.255.255
        set type tunnel
        set remote-ip 10.10.10.1
        set interface "port1"
    next
    edit "ADVPN2"
        set vdom "root"
        set ip 10.10.11.3 255.255.255.255
        set type tunnel
        set remote-ip 10.10.11.1
        set interface "port1"
    next
end
```

Configure iBGP

To ensure our primary ADVPN topology is prioritized, we apply a local preference of 200 to inbound and outbound routes towards the primary ADVPN topology's hub.

```
config router route-map
    edit "RM_ADVPN1"
        config rule
            edit 1
                set set-local-preference 200
            next
        end
    next
end
config router bgp
    set as 65000
    set router-id 192.168.3.1
    config neighbor
        edit "10.10.10.1"
            set remote-as 65000
            set route-map-out "RM_ADVPN1"
        next
        edit "10.10.11.1"
            set remote-as 65000
        next
    end
    config network
        edit 1
            set prefix 192.168.3.0 255.255.255.0
        next
    end
end
```

Configure a static route for the tunnel IP subnet

Both ADVPN topologies require a summary route to be present on spokes in order for recursive routing to function when addressing other spokes.

For good measure however, we have two additional floating blackhole routes to ensure that if the tunnels are down, BGP does not attempt to establish connections with the hub over the default gateway nor install recursive routes for other spokes through the default gateway. This ensures faster convergence upon tunnel restoration.

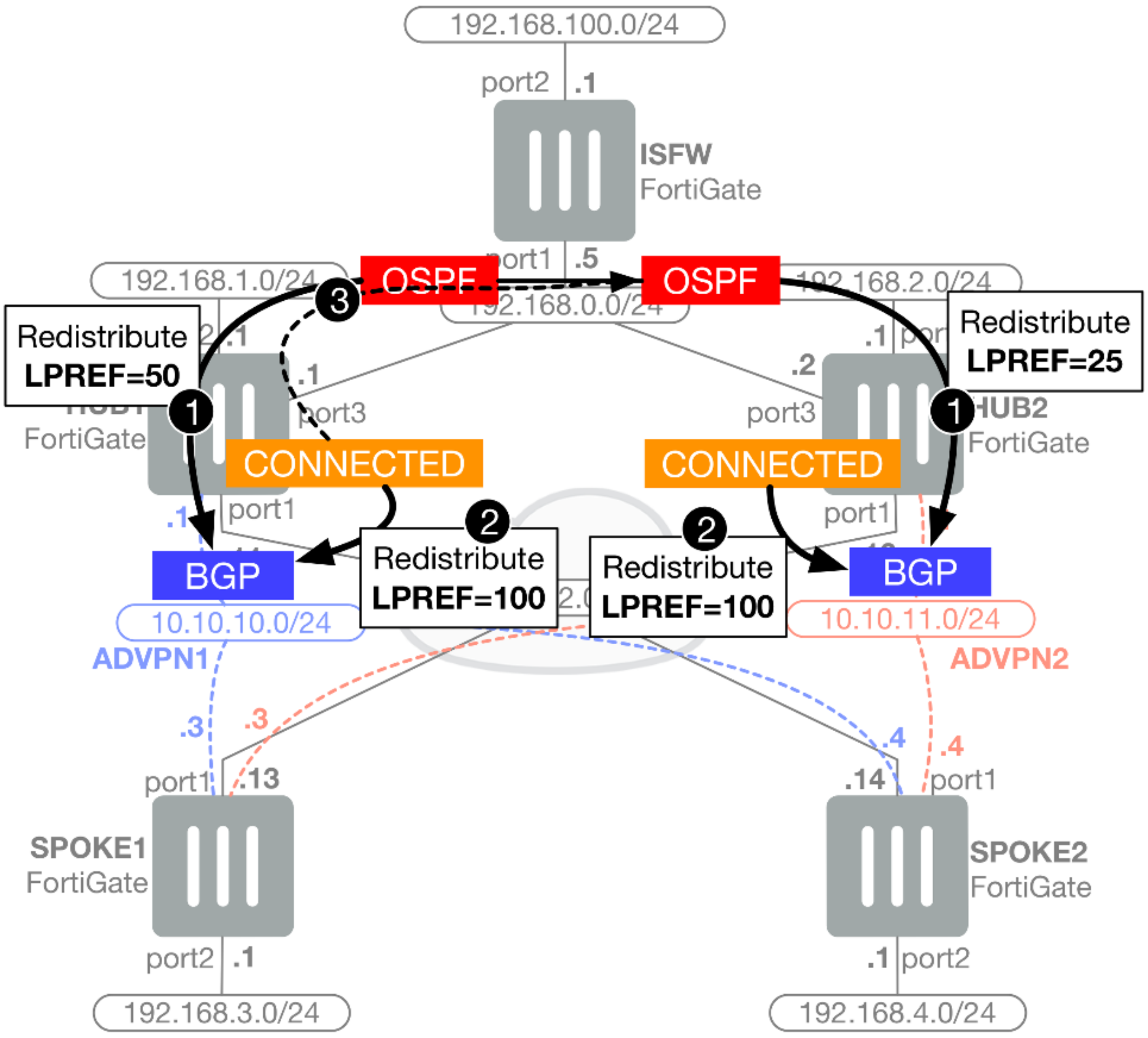
```
config router static
    edit 0
        set dst 10.10.10.0 255.255.255.0
        set device "ADVPN"
    next
    edit 0
        set dst 10.10.11.0 255.255.255.0
        set device "ADVPN2"
    next
    edit 0
        set dst 10.10.10.0 255.255.255.0
        set distance 254
        set blackhole enable
    next
    edit 0
        set dst 10.10.11.0 255.255.255.0
        set distance 254
        set blackhole enable
    next
end
```

Configure policies

```
config firewall policy
    edit 0
        set name "Allowed traffic"
        set srcintf "lan" "ADVPN" "ADVPN2"
        set dstintf "lan" "ADVPN" "ADVPN2"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
end
```

Results

The following schemas help understand the routing logic implemented by this article:

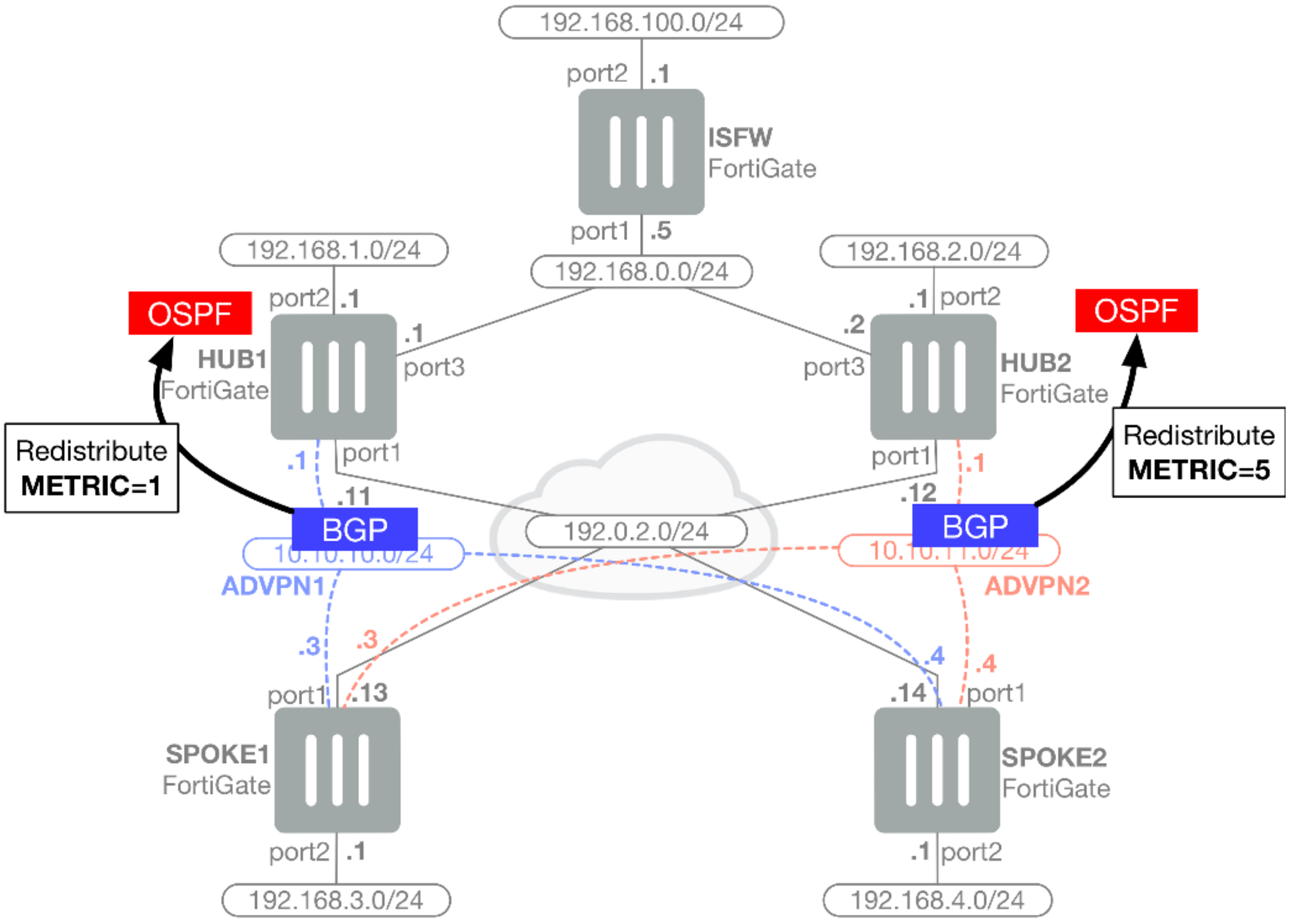


OSPF routes are redistributed into BGP using a **local preference** of **50** for **HUB1** and **25** for **HUB2** (#1 in diagram). Connected routes are redistributed with the **defaultlocal preference** of **100** (#2 in diagram) . As BGP prefers routes with higher local preferences, this logic ensures that a connected route advertised by a HUB to its spokes is preferred *over* that connected route being redistributed in OSPF and advertised by the other HUB (#3 in diagram). Take a look at this output from SPOKE1:

```
SPOKE1 # get router info bgp network
BGP table version is 2, local router ID is 192.168.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*>i192.168.0.0      10.10.10.1               0    100      0 ?
* i                 10.10.11.1               0    100      0 ?
* i192.168.1.010.10.11.1              2      25      0 ?
```

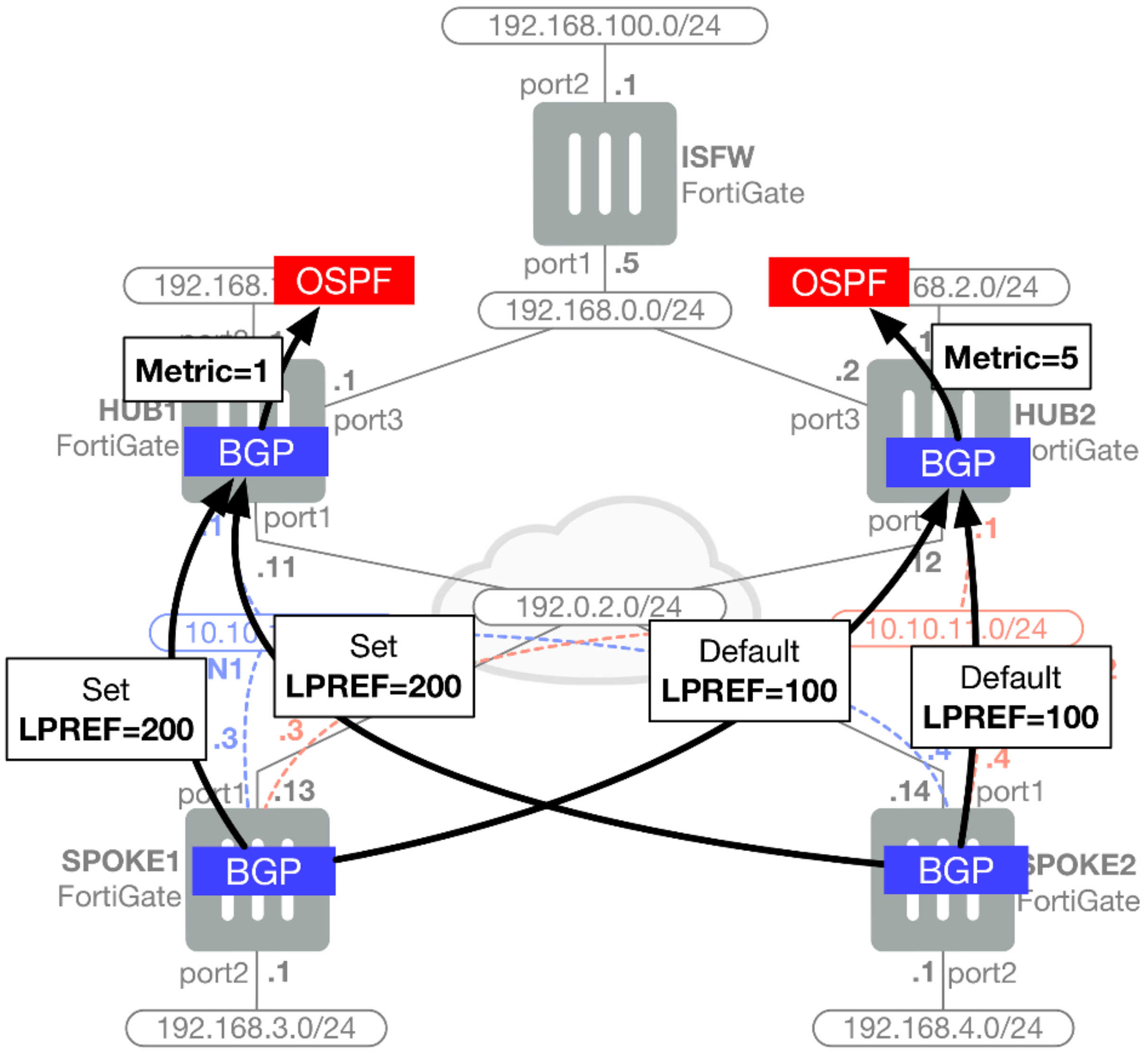
```
*>i               10.10.10.1                0    100       0 ?
* i192.168.2.010.10.10.1              2    50       0 ?
*>i               10.10.11.1                0    100       0 ?
*> 192.168.3.0      0.0.0.0                      100   32768 i
*>i192.168.4.0      10.10.10.4                0    200       0 i
* i               10.10.11.4                0    100       0 i
Total number of prefixes 5
SPOKE1 #
```

In this topology's use case, this is preferable as it ensures traffic going to 192.168.1.0/24 or 192.168.2.0/24 will, under normal conditions, use the VPN to the hub the route is directly connected to rather than circling around through the other hub. As the output illustrates, while SPOKE1 has alternate paths for both HUB networks but prefers using the direct path through each respective network's hub.

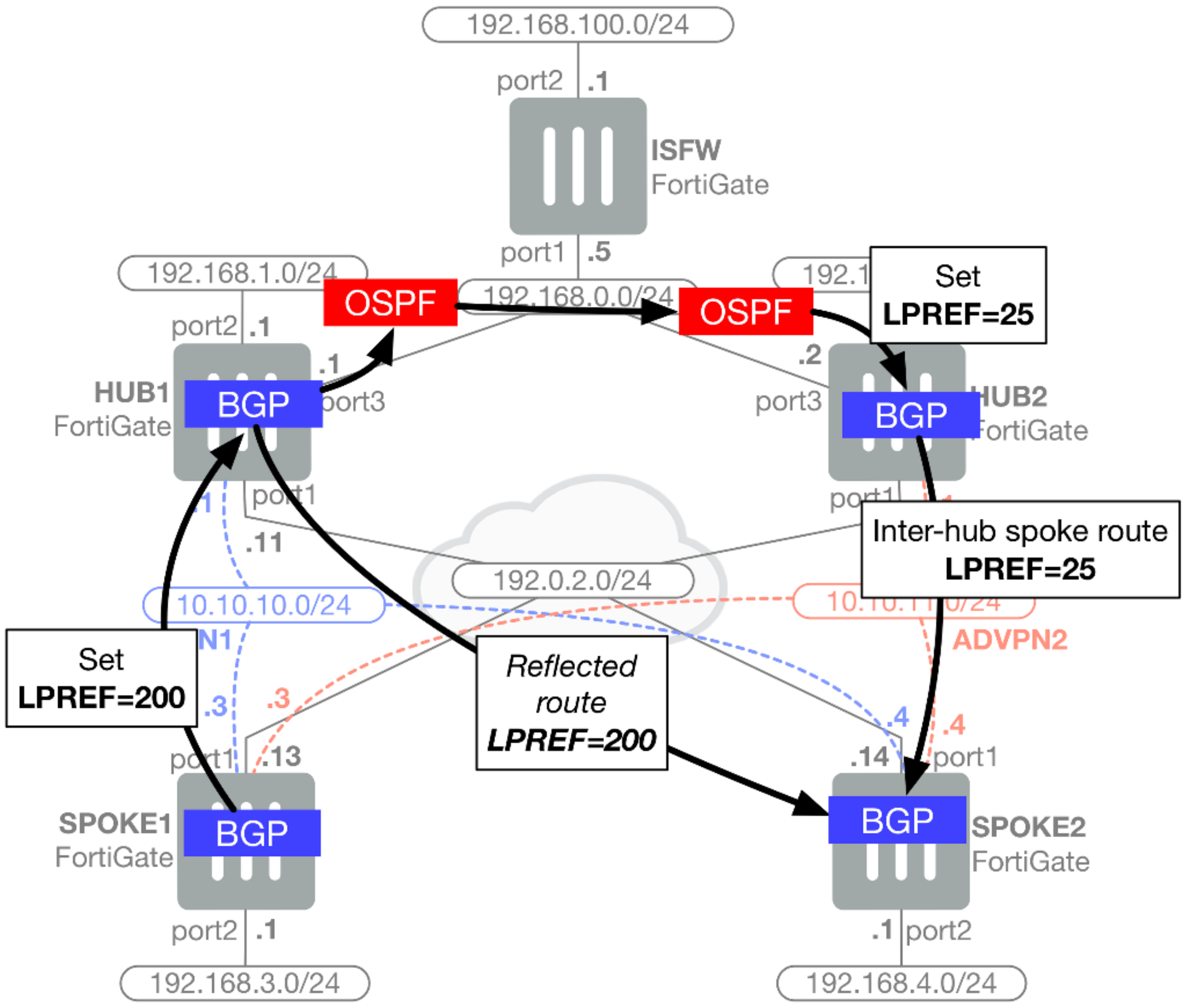


When receiving routes from spokes on either HUB, BGP routes are redistributed into OSPF. Metrics of 1 on the preferred path and 5 on the backup path ensure that traffic flowing in and out of the datacenter's shared subnets (e.g. 192.168.100.0/24) follows the same path. The following output from our ISFW device (which is a basic OSPF router for the intent of this topology) shows those metrics applied with preference for SPOKE1 prefix 192.168.3.0/24 taking the path through ADVPN1 (HUB1):

```
FGT-5 # get router info routing-table details 192.168.3.0
Routing entry for 192.168.3.0/24
  Known via "ospf", distance 110, metric 1, best
  Last update 00:03:21 ago
  * 192.168.0.1, via port1
FGT-5 # get router info ospf database external lsa 192.168.3.0
                AS External Link States
  LS age: 816
  Options: 0x2 (*|-|-|-|-|-|E|-)
  LS Type: AS-external-LSA
  Link State ID: 192.168.3.0 (External Network Number)
  Advertising Router: 192.168.0.1
  LS Seq Number: 80000001
  Checksum: 0x1adf
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 1
        Forward Address: 0.0.0.0
        External Route Tag: 1
  LS age: 133
  Options: 0x2 (*|-|-|-|-|-|E|-)
  LS Type: AS-external-LSA
  Link State ID: 192.168.3.0 (External Network Number)
  Advertising Router: 192.168.0.2
  LS Seq Number: 80000002
  Checksum: 0x28cc
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 5
        Forward Address: 0.0.0.0
        External Route Tag: 0
```



Spokes advertise their own routes by setting a **local preference of 200** on the **primary ADVPN** topology and leaving the **default local preference of 100** on the **secondary ADVPN** topology. This is consistent with the goals of this architecture by ensuring spoke-originated routes are systematically preferred using ADVPN1.



Both local preferences associated with spoke originated routes (**200 and 100**) are higher than those of OSPF redistribution (**50 and 25**) – again, this is to ensure we never favour spoke routes that have looped through OSPF for inter-spoke traffic. In the above example of routing, SPOKE1 has advertised its routes to HUB1 using local preference of 200. HUB1 reflects this route to SPOKE2 and we have reachability as shown from the below output from SPOKE2:

```
SPOKE2 # get router info bgp network 192.168.3.0
BGP routing table entry for 192.168.3.0/24
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Local
    10.10.10.3 from 10.10.10.1 (192.168.3.1)
      Origin IGP metric 0, localpref 200, valid, internal, best
      Originator: 192.168.3.1, Cluster list: 10.10.10.1
      Last update: Mon Jan  9 12:37:06 2017
  Local
```

```
    10.10.11.3 from 10.10.11.1 (192.168.3.1)
      Origin IGP metric 0, localpref 100, valid, internal
      Originator: 192.168.3.1, Cluster list: 10.10.11.1
      Last update: Mon Jan  9 12:37:06 2017
```

As expected, SPOKE2 prefers routes from SPOKE1 when advertised over ADVPN1 due to the local preference being higher. As the output confirms however, there is predictably no route originating from either HUB and injected from OSPF – both hubs having an administrative distance of 100 for internal BGP results in BGP routes always being preferred over OSPF routes and thus, “backdoor” routes going through OSPF are not normally inserted into BGP by either hub. However, if we simulate a failure between **SPOKE1** and **HUB2 (ADVPN2)**:

```
SPOKE1 # config sys int
SPOKE1 (interface) # edit ADVPN2
SPOKE1 (ADVPN2) # set status down
SPOKE1 (ADVPN2) # end
SPOKE1 #
SPOKE2 # get router info bgp network 192.168.3.0
BGP routing table entry for 192.168.3.0/24
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Local
    10.10.10.3 from 10.10.10.1 (192.168.3.1)
      Origin IGP metric 0, localpref 200, valid, internal, best
      Originator: 192.168.3.1, Cluster list: 10.10.10.1
      Last update: Mon Jan  9 12:37:06 2017
  Local
    10.10.11.1 from 10.10.11.1 (10.10.11.1)
      Origin incomplete metric 1, localpref 25, valid, internal
      Last update: Mon Jan  9 13:02:19 2017
```

… SPOKE2's route that was redistributed by HUB1 into OSPF now does indeed appear in the list as HUB2 no longer receives a better iBGP route from SPOKE1. That route is still not being used given that our primary ADVPN1 path is still quite functional. Should we however add another failure, this time between **SPOKE2** and **HUB1 (ADVPN1)**:

```
SPOKE2 # config sys int
SPOKE2 (interface) # edit ADVPN
SPOKE2 (ADVPN) # set status down
SPOKE2 (ADVPN) # end
SPOKE2 #
SPOKE2 # get router info bgp network 192.168.3.0
BGP routing table entry for 192.168.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Local
    10.10.11.1 from 10.10.11.1 (10.10.11.1)
      Origin incomplete metric 1, localpref 25, valid, internal, best
      Last update: Mon Jan  9 13:02:19 2017

SPOKE2 # exec ping-options source 192.168.4.1
SPOKE2 # exec ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: icmp_seq=0 ttl=253 time=6.5 ms
64 bytes from 192.168.3.1: icmp_seq=1 ttl=253 time=0.9 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=253 time=1.1 ms
64 bytes from 192.168.3.1: icmp_seq=3 ttl=253 time=1.1 ms
^C
--- 192.168.3.1 ping statistics ---
```

```
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.9/2.4/6.5 ms
SPOKE2 # get router info routing-table details 192.168.3.0
Routing entry for 192.168.3.0/24
  Known via "bgp", distance 200, metric 1, best
  Last update 00:09:42 ago
  * 10.10.11.1, via ADVPN2
```

… SPOKE2 now only has the path going through HUB2, then HUB1 in order to reach SPOKE1. Worthwhile to note that in this condition, no ADVPN SHORTCUT will establish as spoke devices are on segregated ADVPN topologies and do not share a common ADVPN hub. If we restore services:

```
SPOKE1 # config sys int
SPOKE1 (interface) # edit ADVPN2
SPOKE1 (ADVPN2) # set status up
SPOKE1 (ADVPN2) # end
SPOKE1 #
SPOKE2 # config sys int
SPOKE2 (interface) # edit ADVPN
SPOKE2 (ADVPN) # set status up
SPOKE2 (ADVPN) # end
SPOKE2 #
```

… we return to correct routing tables. If we then issue some traffic towards SPOKE1's subnet, we get the initiation of an ADVPN SHORTCUT directly to SPOKE1:

```
SPOKE2 # get router info routing-table details 192.168.3.0
Routing entry for 192.168.3.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 00:00:37 ago
  * 10.10.10.3 (recursive via 10.10.10.1)

SPOKE2 # exec ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: icmp_seq=0 ttl=254 time=1.8 ms
64 bytes from 192.168.3.1: icmp_seq=1 ttl=255 time=0.5 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 192.168.3.1: icmp_seq=3 ttl=255 time=0.5 ms
64 bytes from 192.168.3.1: icmp_seq=4 ttl=255 time=0.4 ms
--- 192.168.3.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.7/1.8 ms
SPOKE2 #
SPOKE2 # get router info routing-table details 192.168.3.0
Routing entry for 192.168.3.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 00:00:04 ago
  * 10.10.10.3, via ADVPN_0

SPOKE2 # get vpn ipsec tunnel summary
'ADVPN_0' 192.0.2.13:0  selectors(total,up): 1/1  rx(pkt,err): 4/0  tx(pkt,err): 4/0
'ADVPN' 192.0.2.11:0  selectors(total,up): 1/1  rx(pkt,err): 148/0  tx(pkt,err): 149/0
'ADVPN2' 192.0.2.12:0  selectors(total,up): 1/1  rx(pkt,err): 4102/0  tx(pkt,err): 4103/2
```

Configuring ADVPN in FortiOS 5.4 (Expert)

In this recipe, we will explore a new VPN feature introduced in FortiOS 5.4.0: ADVPN.

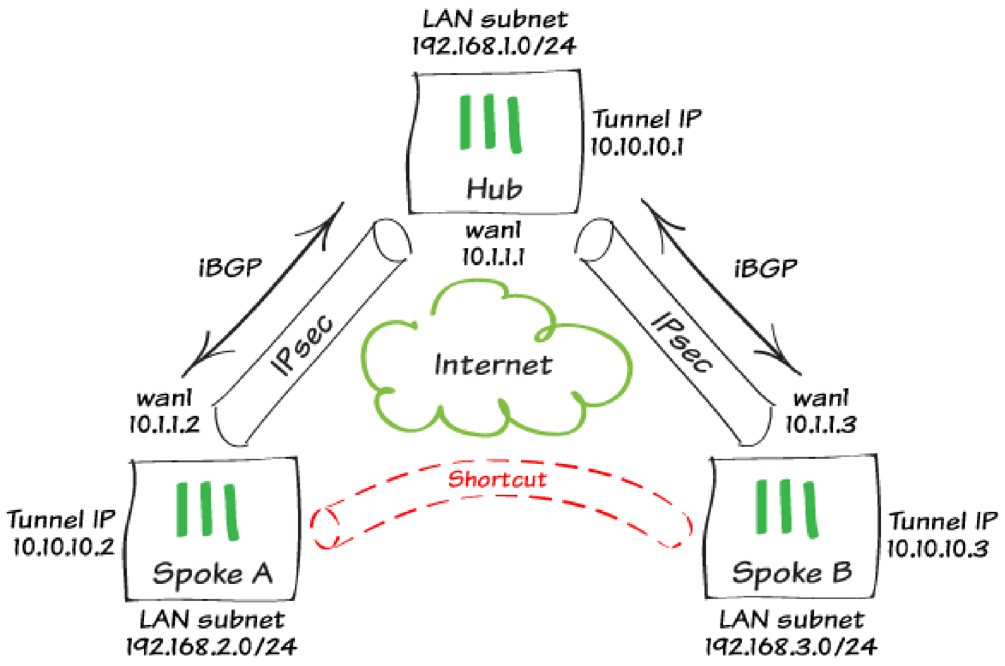
ADVPN (Auto Discovery VPN) is an IPsec technology based on an IETF RFC draft (https://tools.ietf.org/html/draft-sathyanarayan-ipsecme-advpn-03). In simple terms, ADVPN allows a traditional hub and spoke VPN's spokes to establish dynamic, on-demand direct tunnels between each other so as to avoid routing through the topology's hub device. ADVPN requires the use of dynamic routing in order to function and FortiOS 5.4 supports both BGP and RIP. This recipe will focus on using BGP and its route-reflector mechanism as the dynamic routing solution to use with ADVPN.

ADVPN's primary advantages is that it provides the full meshing capabilities to a standard hub and spoke topology, greatly reducing the provisioning effort required for full spoke to spoke low delay reachability and addressing the scalability issues associated with very large fully meshed VPN networks.

BGP (and specifically, iBGP) is a natural fit for ADVPN as its route reflector mechanism resides on the VPN hub device and mirrors routing information from each spoke peer to each other. Furthermore, dynamic group peers result in near zero-touch hub provisioning when a new spoke is introduced in the topology.

As pictured, while the static configuration will involve both spoke FortiGate units to connect to our circular hub FortiGate, Spoke A will be able to establish a dynamic on-demand shortcut IPSec tunnel to Spoke B (and vice versa) if a host behind either spoke attempts to reach a host behind the other spoke. We will complete the configuration below and our verification step below will include reachability from 192.168.2.1 (spoke A) to 192.168.3.1 (spoke B) over the dynamically created shortcut link.

This recipe is documented in CLI as configuration such as BGP and ADVPN are best done using the command line interface. We are assuming basic IP and default routing configuration has been completed on the devices.



1. Configure the Hub FortiGate

Using the CLI, configure phase 1 parameters.

The auto-discovery commands enable the sending and receiving of shortcut messages to spokes (the hub is responsible for lettings the spokes know that they should establish those tunnels).

Note: aggressive mode is **not** supported currently for ADVPN.

```
config vpn ipsec phase1-interface
    edit "ADVPN"
        set type dynamic
        set interface "wan1"
        set proposal aes128-sha1
        set dhgrp 2
        set auto-discovery-sender enable
        set add-route disable
        set psksecret fortinet
    next
end
```

Configure the phase2 parameters.

This is a standard phase 2 configuration.

```
config vpn ipsec phase2-interface
    edit "ADVPN-P2"
        set phase1name "ADVPN"
        set proposal aes128-sha1
    next
end
```

Configure the tunnel interface IP.

ADVPN requires that tunnel IPs be configured on each device connecting to the topology. Those IP addresses need to be unique for each peer. A particularity of the hub is that it needs to define a bogus remote-IP address (10.10.10.254 in our example). This address should be unused in the topology and it will not be actually considered as part of the configuration for the hub.

```
config system interface
    edit "ADVPN"
        set vdom "root"
        set ip 10.10.10.1 255.255.255.255
        set type tunnel
        set remote-ip 10.10.10.254
        set interface "wan1"
    next
end
```

Configure iBGP and route-reflection.

iBGP will be our overlay protocol of choice for enabling ADVPN communications. We are using an arbitrary private AS number (65000) in our example, and configuring a dynamic client group to reduce provisioning requirements.

While we are advertising our LAN network directly (“config network” command), route redistribution is a perfectly valid alternative.

```
config router bgp
    set as 65000
    set router-id 10.10.10.1
    config neighbor-group
        edit "ADVPN-PEERS"
            set remote-as 65000
            set route-reflector-client enable
            set next-hop-self enable
        next
    end
    config neighbor-range
        edit 0
            set prefix 10.10.10.0 255.255.255.0
            set neighbor-group "ADVPN-PEERS"
        next
    end
    config network
        edit 0
            set prefix 192.168.1.0 255.255.255.0
        next
    end
end
```

Configure basic policies to allow traffic to flow between the local network and the ADVPN VPN topology. As we generally desire traffic to be allowed between spokes in an ADVPN setup, we must remember to create a policy allowing spoke to spoke communications.

```
config firewall policy
    edit 0
        set name "OUT ADVPN"
        set srcintf "lan"
        set dstintf "ADVPN"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
    edit 0
        set name "IN ADVPN"
        set srcintf "ADVPN"
        set dstintf "lan"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
    edit 0
        set name "ADVPNtoADVPN"
        set srcintf "ADVPN"
        set dstintf "ADVPN"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
end
```

2. Configure the Spoke FortiGates

Using the CLI, configure phase 1 parameters.

Note that we are configuring only one of the spokes in this example – the parameters that need to change for each spoke are highlighted in red.

```
config vpn ipsec phase1-interface
 edit "ADVPN"
  set interface "wan1"
  set proposal aes128-sha1
  set dhgrp 2
  set auto-discovery-receiver enable
  set add-route disable
  set remote-gw 10.1.1.1
  set psksecret fortinet
 next
end
```

Configure the phase2 parameters.

```
config vpn ipsec phase2-interface
    edit "ADVPN-P2"
        set phase1name "ADVPN"
        set proposal aes128-sha1
       set auto-negotiate enable
    next
end
```

Configure the tunnel interface IP.

Notice that on the spokes, the remote IP is actually used and points to the IP defined on the hub.

```
config system interface
    edit "ADVPN"
        set vdom "root"
        set ip 10.10.10.2 255.255.255.255
        set type tunnel
        set remote-ip 10.10.10.1
        set interface "wan1"
    next
end
```

Config iBGP.

This is a static standard configuration and as stated for the hub, redistribution could be used instead of explicit route advertisement.

```
config router bgp
    set as 65000
    set router-id 10.10.10.2
    config neighbor
        edit "10.10.10.1"
            set soft-reconfiguration enable
            set remote-as 65000
            set next-hop-self enable
        next
    end
    config network
        edit 0
            set prefix 192.168.2.0 255.255.255.0
        next
    end
end
```

Configure a static route for the tunnel IP subnet.

This is an important special step for the spokes as they need a summary route that identifies all tunnel IP used over your topology to point towards the ADVPN interface. In our example, we use 10.10.10.0/24 (if our network planning expects less than 255 sites). Be sure to adequately plan this IP range as it needs to be hardcoded in the spokes.

```
config router static
    edit 0
        set dst 10.10.10.0 255.255.255.0
        set device "ADVPN"
    next
end
```

Configure policies.

```
config firewall policy
    edit 0
        set name "OUT ADVPN"
        set srcintf "lan"
        set dstintf "ADVPN"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
    edit 0
        set name "IN ADVPN"
        set srcintf "ADVPN"
        set dstintf "lan"
        set srcaddr "all"
        set dstaddr "all"
        set action accept
        set schedule "always"
        set service "ALL"
        set status enable
    next
end
```

Results

We can validate the behaviour of our configuration using a few commands. We are going to run these commands from SPOKE A.

get router info routing-table bgp will at a minimum display the learned routes from the topology. Note the recursive routing – a result of our spoke's required static route. In this case, there has not been any traffic between our local subnet (192.168.2.0/24) and the other spoke's subnet, as the routes are both going through the hub.

```
B       192.168.1.0/24 [200/0] via 10.0.0.1, ADVPN, 22:30:21
B       192.168.3.0/24 [200/0] via 10.0.0.3 (recursive via 10.0.0.1), 22:30:21
```

However once we initiate a ping between both spokes, we obtain a different display of routing information – routing now goes through a newly established dynamic tunnel directly through the remote spoke rather than through the hub. The ping hiccup that occurs is the tunnel rerouting through a newly negotiated tunnel to the other spoke.

Our routing information now displays the remote subnet as being available through the spoke directly, through interface ADVPN_0, a dynamically instantiated interface going to that spoke.

```
FG # exec ping-options source 192.168.2.1
FG # exec ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: icmp_seq=0 ttl=254 time=38.3 ms
64 bytes from 192.168.3.1: icmp_seq=1 ttl=254 time=32.6 ms
Warning: Got ICMP 3 (Destination Unreachable)
64 bytes from 192.168.3.1: icmp_seq=2 ttl=255 time=43.0 ms
64 bytes from 192.168.3.1: icmp_seq=3 ttl=255 time=31.7 ms
64 bytes from 192.168.3.1: icmp_seq=4 ttl=255 time=31.2 ms
--- 192.168.3.1 ping statistics ---
```

```
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 31.2/35.3/43.0 ms
FG # get router info routing-table bgp
B       192.168.1.0/24 [200/0] via 10.0.0.1, ADVPN, 22:34:13
B       192.168.3.0/24 [200/0] via 10.0.0.3, ADVPN_0, 00:02:28
```

Some additional data can be obtained using the very useful **diag vpn tunnel list** command. We are highlighting the aspects in the output which convey data specific to ADVPN, in this case the auto-discovery flag and the child-parent relationship of new instantiated dynamic tunnel interfaces.

```
FG # diag vpn tunnel list
list all ipsec tunnel in vd 0
------------------------------------------------------
name=ADVPN_0 ver=1 serial=a 10.1.1.2:0->10.1.1.3:0
bound_if=6 lgwy=static/1 tun=intf/0 mode=dial_inst/3 encap=none/0
 parent=ADVPN index=0
proxyid_num=1 child_num=0 refcnt=19 ilast=3 olast=604 auto-discovery=2
stat: rxp=7 txp=7 rxb=1064 txb=588
dpd: mode=on-demand on=1 idle=20000ms retry=3 count=0 seqno=0
natt: mode=none draft=0 interval=0 remote_port=0
proxyid=ADVPN-P2 proto=0 sa=1 ref=2 serial=1 auto-negotiate adr
  src: 0:0.0.0.0/0.0.0.0:0
  dst: 0:0.0.0.0/0.0.0.0:0
  SA: ref=3 options=2f type=00 soft=0 mtu=1438 expire=42680/0B replaywin=2048 seqno=8 esn=0
  life: type=01 bytes=0/0 timeout=43152/43200
  dec: spi=9a487db3 esp=aes key=16 55e53d9fbc8dbeaa6df1032fbc80c4f6
       ah=sha1 key=20 a1470452c6a444f26a070add087f0d970c18e3a7
  enc: spi=3c37fea7 esp=aes key=16 8fd62a6745a9ba4fda062d4504b76851
       ah=sha1 key=20 44c606f1ef1bf5739ba62f6572031aa956974d0a
  dec:pkts/bytes=7/588, enc:pkts/bytes=7/1064
------------------------------------------------------
name=ADVPN ver=1 serial=9 10.1.1.2:0->10.1.1.1:0
bound_if=6 lgwy=static/1 tun=intf/0 mode=auto/1 encap=none/0
proxyid_num=1 child_num=1 refcnt=22 ilast=8 olast=8 auto-discovery=2
stat: rxp=3120 txp=3120 rxb=399536 txb=191970
dpd: mode=on-demand on=1 idle=20000ms retry=3 count=0 seqno=12
natt: mode=none draft=0 interval=0 remote_port=0
proxyid=ADVPN-P2 proto=0 sa=1 ref=2 serial=1 auto-negotiate adr
  src: 0:0.0.0.0/0.0.0.0:0
  dst: 0:0.0.0.0/0.0.0.0:0
  SA: ref=3 options=2f type=00 soft=0 mtu=1438 expire=4833/0B replaywin=2048 seqno=5ba esn=0
  life: type=01 bytes=0/0 timeout=43148/43200
  dec: spi=9a487db2 esp=aes key=16 4f70d27edad656cfcacbae61b23d4b11
       ah=sha1 key=20 b19ea87c90dd92d1cab58cbf24ae8fe12ee927cb
  enc: spi=b3dde355 esp=aes key=16 efbb4440df75018610b4ba8f5756167d
       ah=sha1 key=20 81cc9cee3bee1c2dba0eb1e7ac66e9d34b67bde9
  dec:pkts/bytes=1465/90152, enc:pkts/bytes=1465/187560
------------------------------------------------------
```