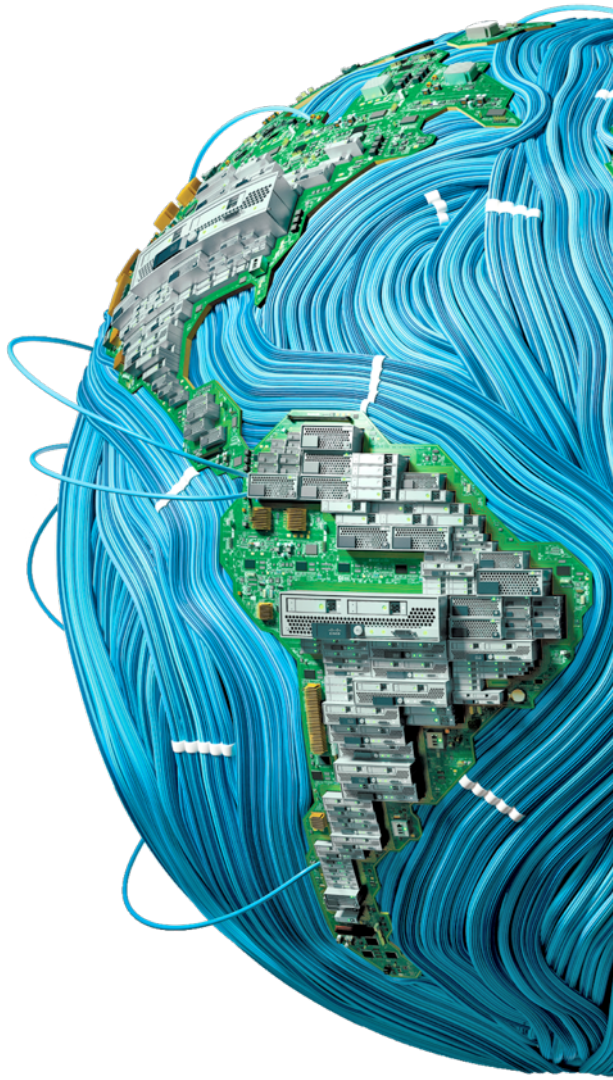
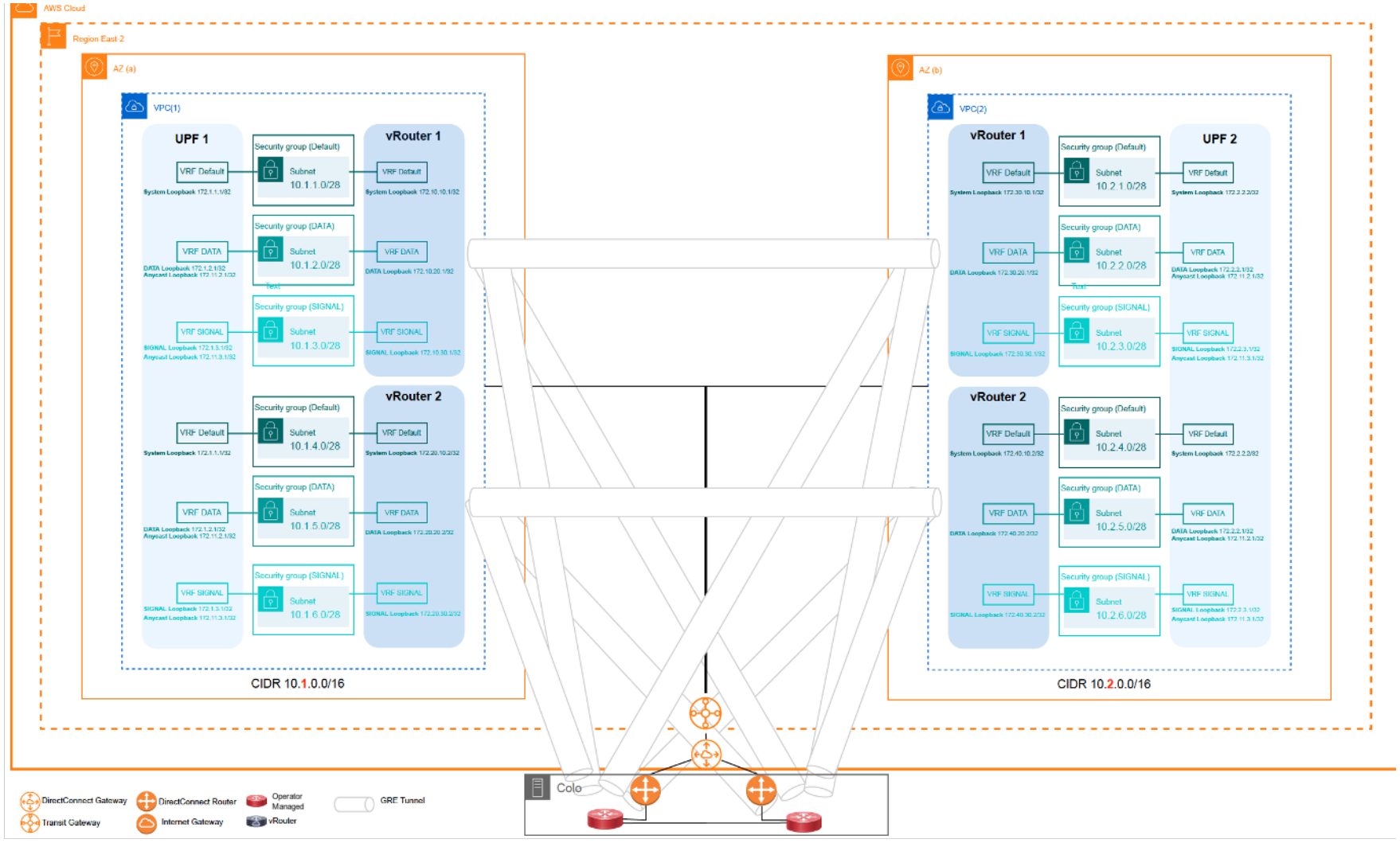
XRv9k on AWS
High Availability (HA) Solution

Akshat Sharma, TME

May 2021

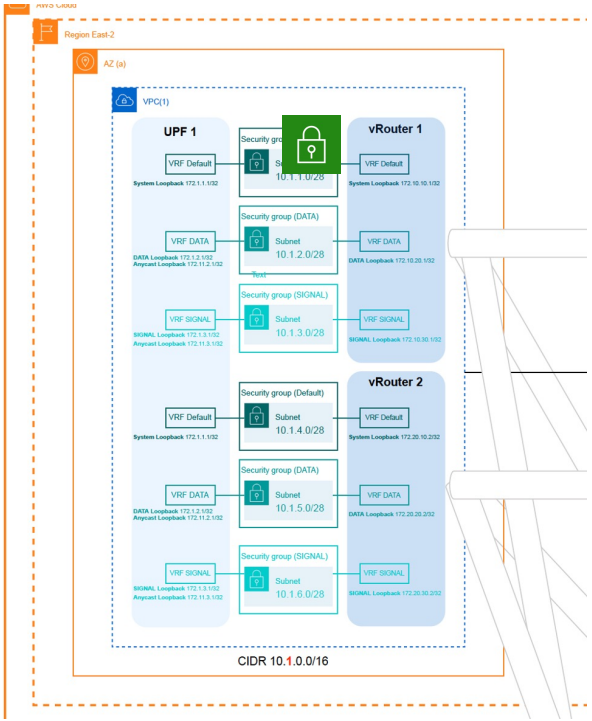


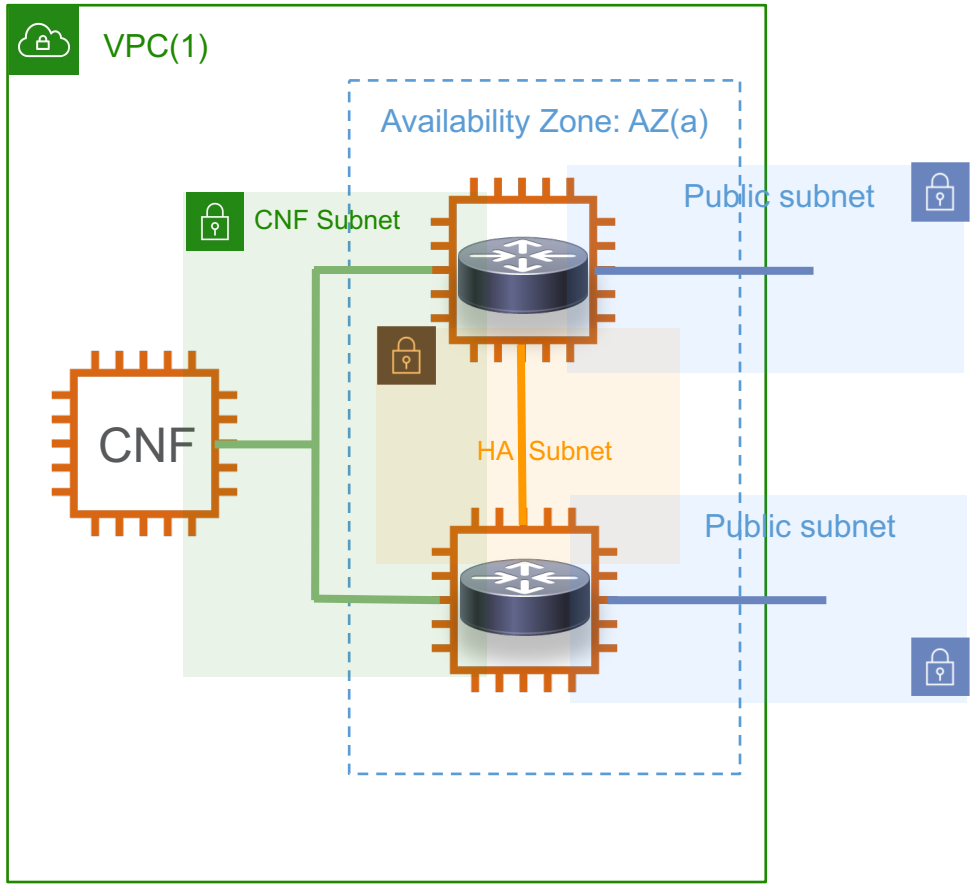
AWS Underlay 5G Deployment with vRouters


AWS Cloud
Region East 2
AZ (a)
VPC(1)
UPF 1
VRF Default
System Loopback 172.1.1.1/32
VRF DATA
DATA Loopback 172.1.2.1/32
Anycast Loopback 172.11.2.1/32
VRF SIGNAL
SIGNAL Loopback 172.1.3.1/32
Anycast Loopback 172.11.3.1/32
Security group (Default)
Subnet
10.1.1.0/28
Security group (DATA)
Subnet
10.1.2.0/28
Security group (SIGNAL)
Subnet
10.1.3.0/28
vRouter 1
VRF Default
System Loopback 172.10.10.1/32
VRF DATA
DATA Loopback 172.10.20.1/32
VRF SIGNAL
SIGNAL Loopback 172.10.30.1/32
VRF Default
System Loopback 172.1.1.1/32
VRF DATA
DATA Loopback 172.1.2.1/32
Anycast Loopback 172.11.2.1/32
VRF SIGNAL
SIGNAL Loopback 172.1.3.1/32
Anycast Loopback 172.11.3.1/32
Security group (Default)
Subnet
10.1.4.0/28
Security group (DATA)
Subnet
10.1.5.0/28
Security group (SIGNAL)
Subnet
10.1.6.0/28
vRouter 2
VRF Default
System Loopback 172.20.10.2/32
VRF DATA
DATA Loopback 172.20.20.2/32
VRF SIGNAL
SIGNAL Loopback 172.20.30.2/32
CIDR 10.1.0.0/16
AZ (b)
VPC(2)
vRouter 1
VRF Default
System Loopback 172.30.10.1/32
VRF DATA
DATA Loopback 172.30.20.1/32
VRF SIGNAL
SIGNAL Loopback 172.30.30.1/32
Security group (Default)
Subnet
10.2.1.0/28
Security group (DATA)
Subnet
10.2.2.0/28
Security group (SIGNAL)
Subnet
10.2.3.0/28
UPF 2
VRF Default
System Loopback 172.2.2.2/32
VRF DATA
DATA Loopback 172.2.2.1/32
Anycast Loopback 172.11.2.1/32
VRF SIGNAL
SIGNAL Loopback 172.2.3.1/32
Anycast Loopback 172.11.3.1/32
vRouter 2
VRF Default
System Loopback 172.40.10.2/32
VRF DATA
DATA Loopback 172.40.20.2/32
VRF SIGNAL
SIGNAL Loopback 172.40.30.2/32
Security group (Default)
Subnet
10.2.4.0/28
Security group (DATA)
Subnet
10.2.5.0/28
Security group (SIGNAL)
Subnet
10.2.6.0/28
VRF Default
System Loopback 172.2.2.2/32
VRF DATA
DATA Loopback 172.2.2.1/32
Anycast Loopback 172.11.2.1/32
VRF SIGNAL
SIGNAL Loopback 172.2.3.1/32
Anycast Loopback 172.11.3.1/32
CIDR 10.2.0.0/16
Colo
DirectConnect Gateway
Transit Gateway
DirectConnect Router
Internet Gateway
Operator Managed
vRouter
GRE Tunnel

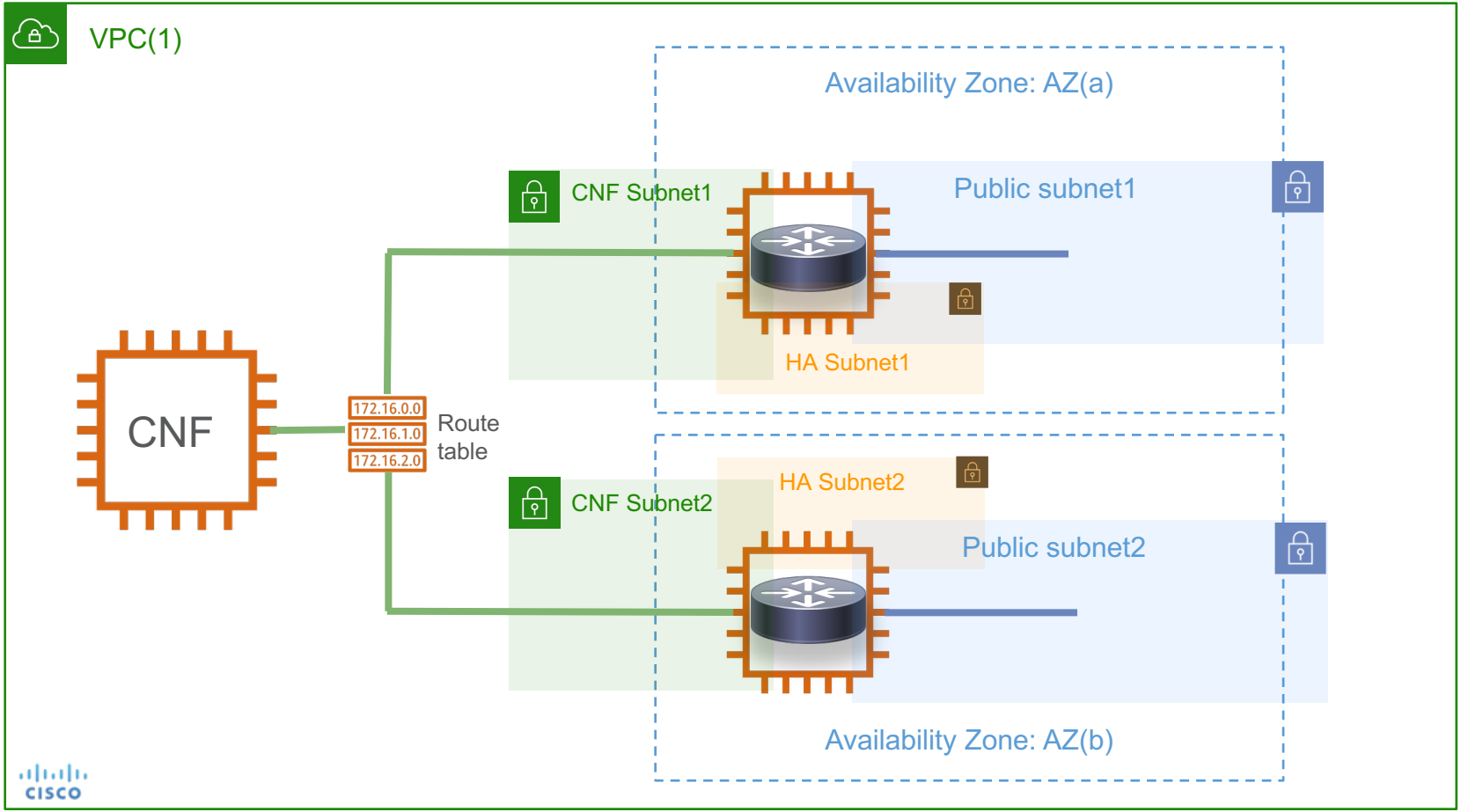
vRouter HA deployments

vRouter HA In Single AZ





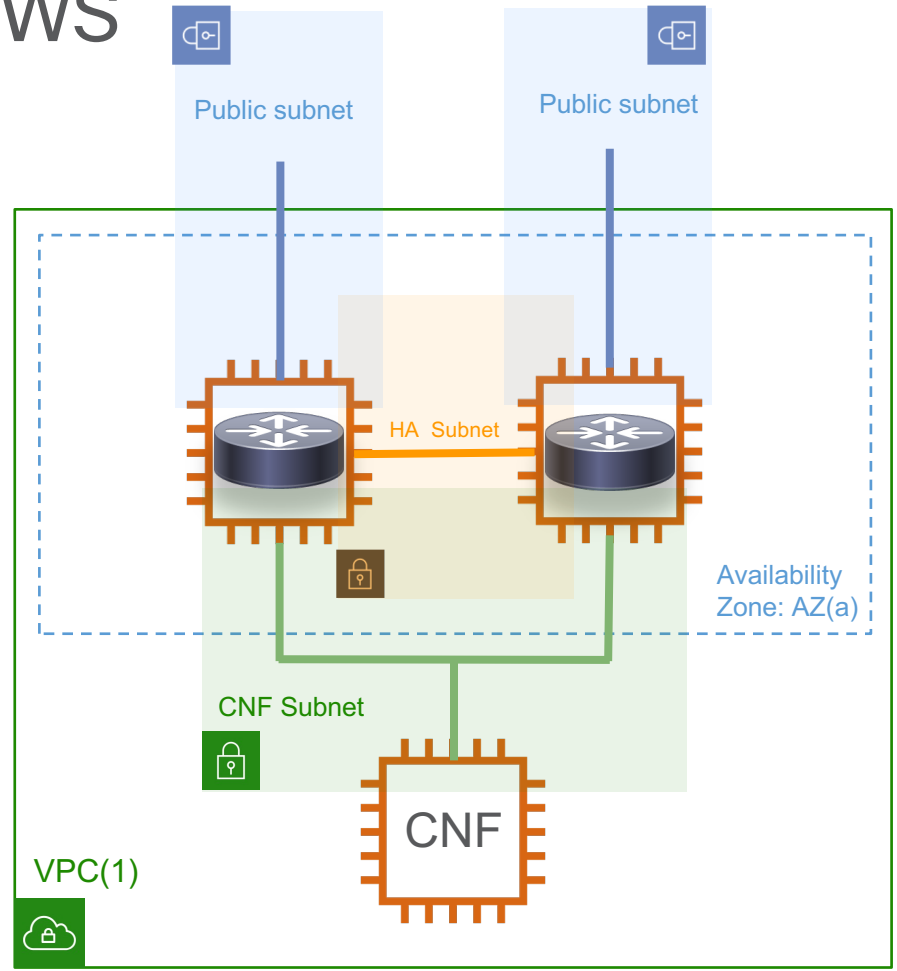
vRouter HA Across Two AZ



Why don’t we just use HSRP ?

Problems using HSRP on AWS

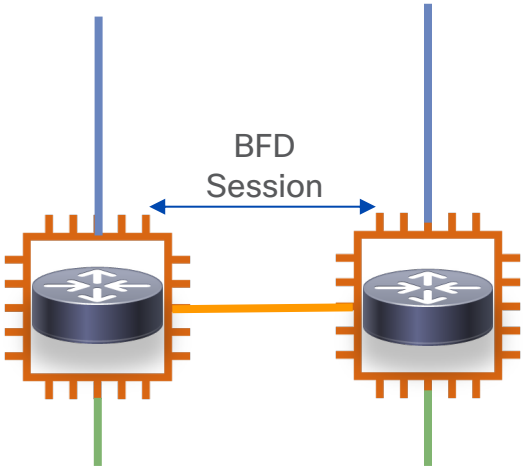
- AWS as an underlay imposes some limitations on the redundancy design
- AWS underlay blocks multicast/broadcast packets
- HSRP/VRRP hellos are sent to multicast addresses – blocked
- Further HSRP depends on the use of gratuitous ARP packets that are dropped as well.
- Active/Standby Selection and Failover Notifications therefore require some tricks on the vRouter and AWS level to accomplish sub-second failovers



AWS vRouter HA Redundancy Design Principles

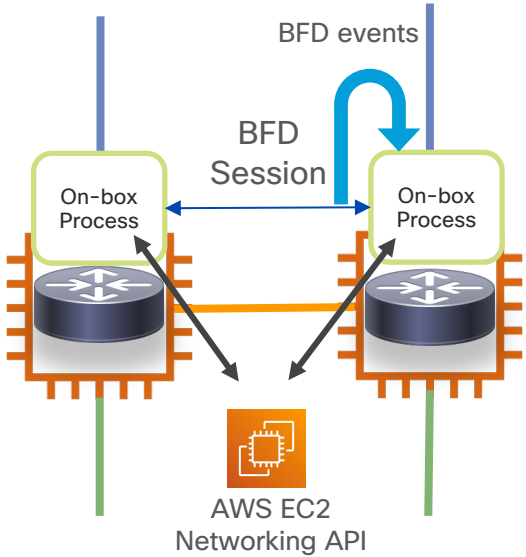
Components of the Redundancy Design

1 Peering Connection and protocol to Select Active/Standby



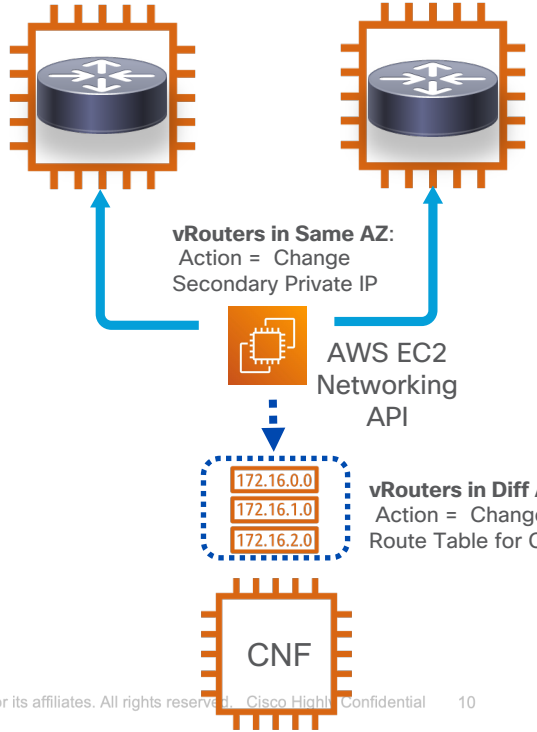
1) Regular BFD session (Single-hop BFD for single AZ or Multi-Hop for Diff AZ with local routing in VPC)
2) BFDoGRE session (Single-Hop over GRE for both AZ deployments)

2 Process on vRouter to Listen to BFD events, and trigger actions



1) On-box Process listens to BFD events
2) Maintains Active/Standby State in relation to the Peer
3) Calls AWS API to relevant action

3 AWS API Interface to affect failover at the underlay



1

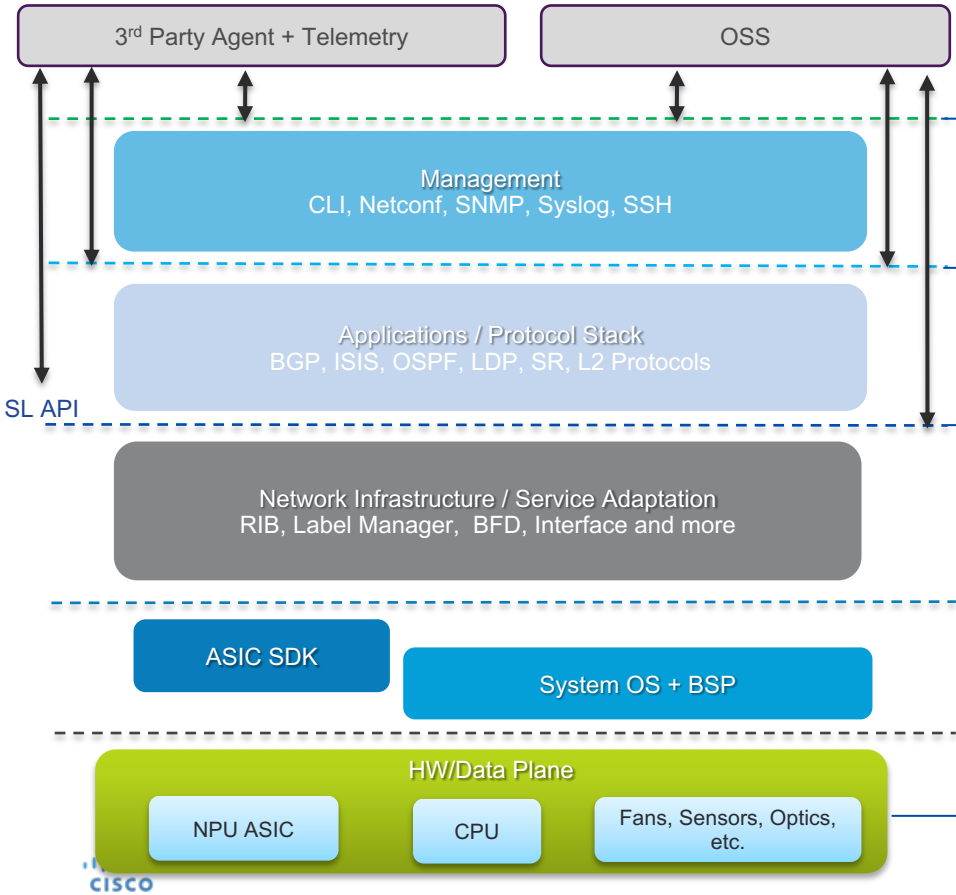
Peering Connection and protocol to Select Active/Standby

2

Process on vRouter to Listen to BFD events, and trigger actions

IOS-XR SL-API to Setup BFD and Monitor it

IOS-XR: API-Driven, Layered SW Architecture



Management/Presentation Layer

Provides access to configure and manage the stack through Network config/oper DB: Yang Models, CLI.

Application/Protocol Layer

Provides APIs into the Routing Protocols (BGP, IGP, SR, etc.)

Network Infrastructure Layer / Service Adaptation Layer

- Acts as the bridge between the Application Layer and the HW
- Presents abstractions to the Application/Protocol Layer

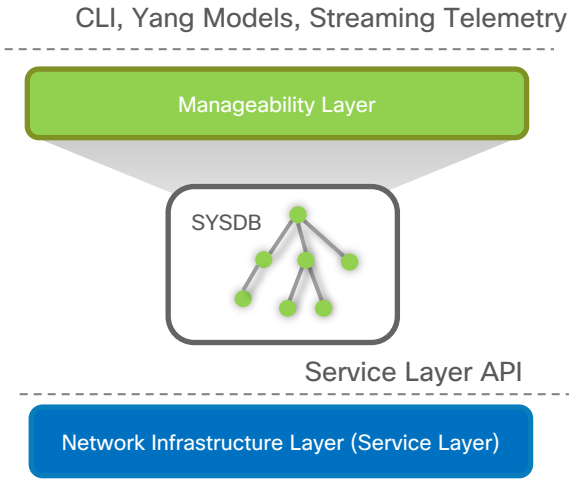
System OS - Linux Kernel
BSP(Board Support Package) - Boot Loader, Device Drivers, etc.

ASIC SDK and drivers for the SDK

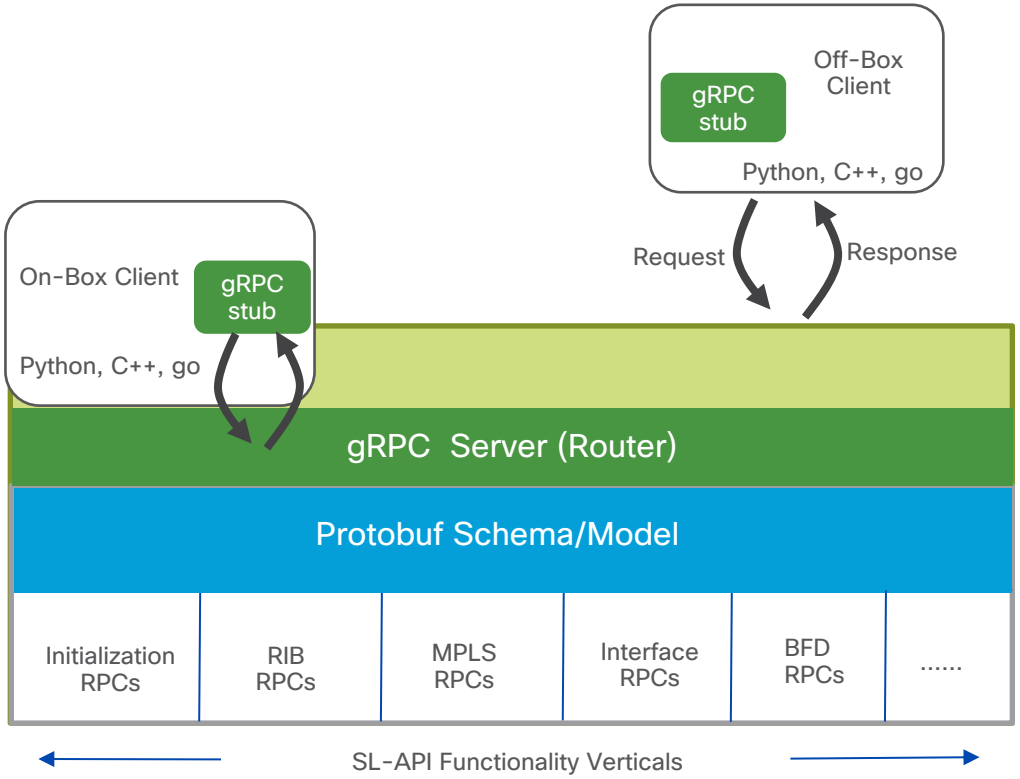
Hardware
Consists of ASIC/Chipset from HW vendors + CPU, Fans, Sensors

12

Service Layer API Architecture



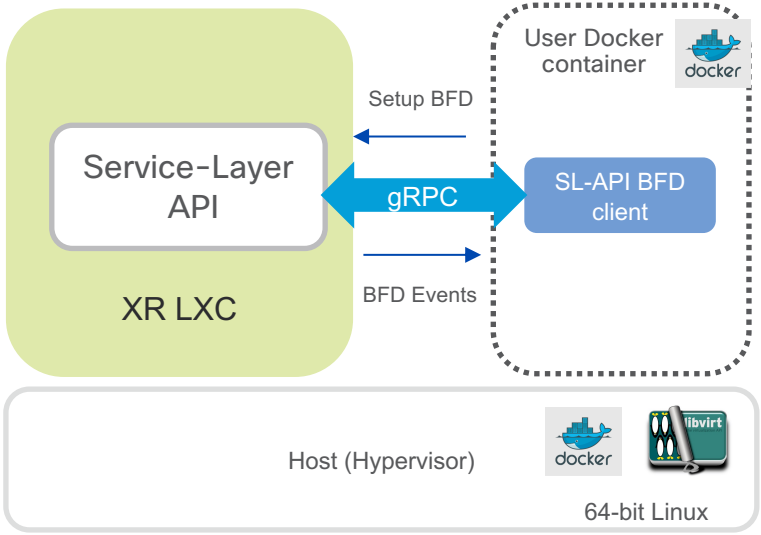
Service-Layer APIs bypass SysDB giving higher performance



More info on SL API : https://xrdocs.io/cisco-service-layer/

13

Max Performance with OnBox SL-API BFD-HA App



1) Using Xrv9000 Application-hosting capabilities, an SL-API BFD client will be spun up in a Docker container on the vRouter itself.

2) BFD sessions are normally associated with a Routing Protocol “Client” like BGP, OSPF, ISIS etc.

3) With Service-layer API- “Service-Layer” acts as client for BFD allowing us to add/modify BFD sessions directly WITHOUT ANY IOS-XR CONFIG. BFD session state is transient/ephemeral – tied to the client.

4) Real-time BFD events are then received over gRPC, allowing the SLAPI client to react to Peer-router unreachable/down events

XRv9000 Container Based AppHosting

1

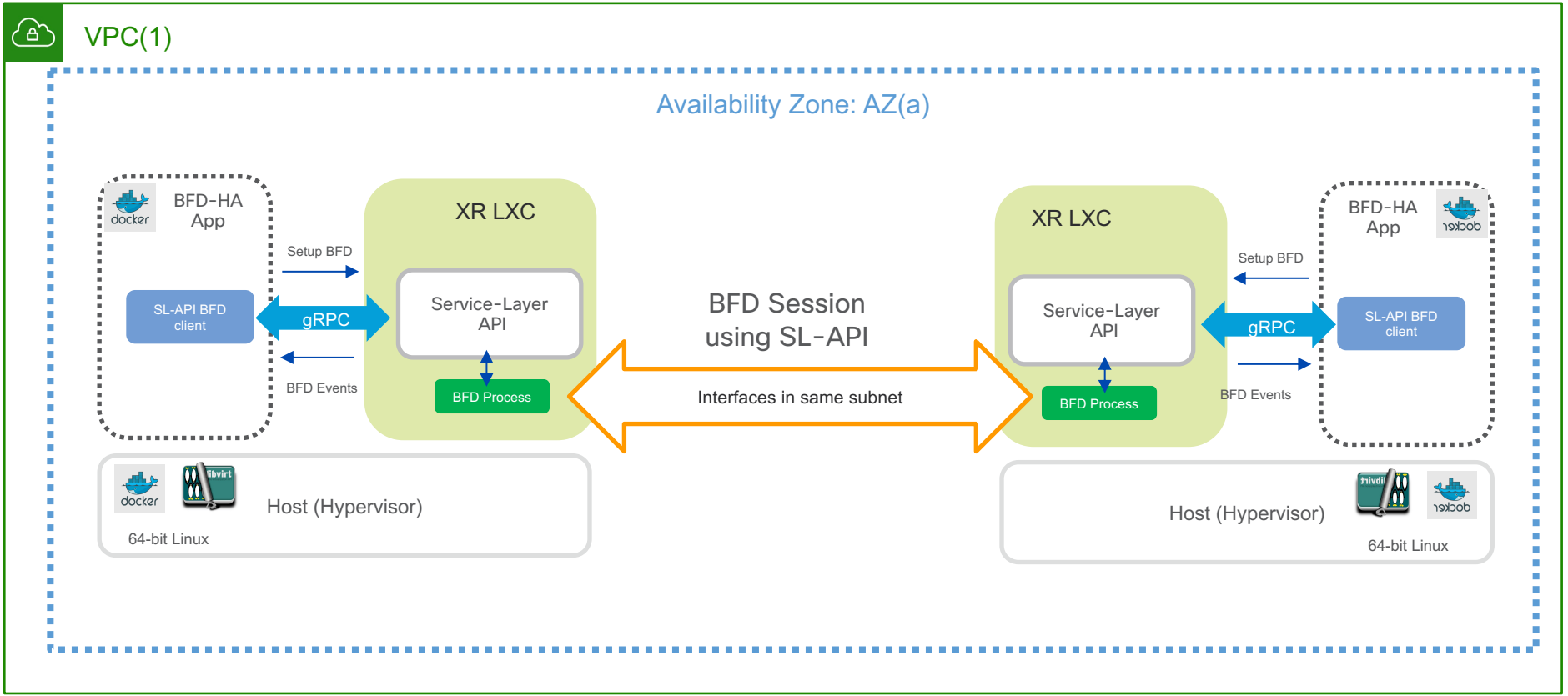
Peering Connection and protocol to Select Active/Standby

2

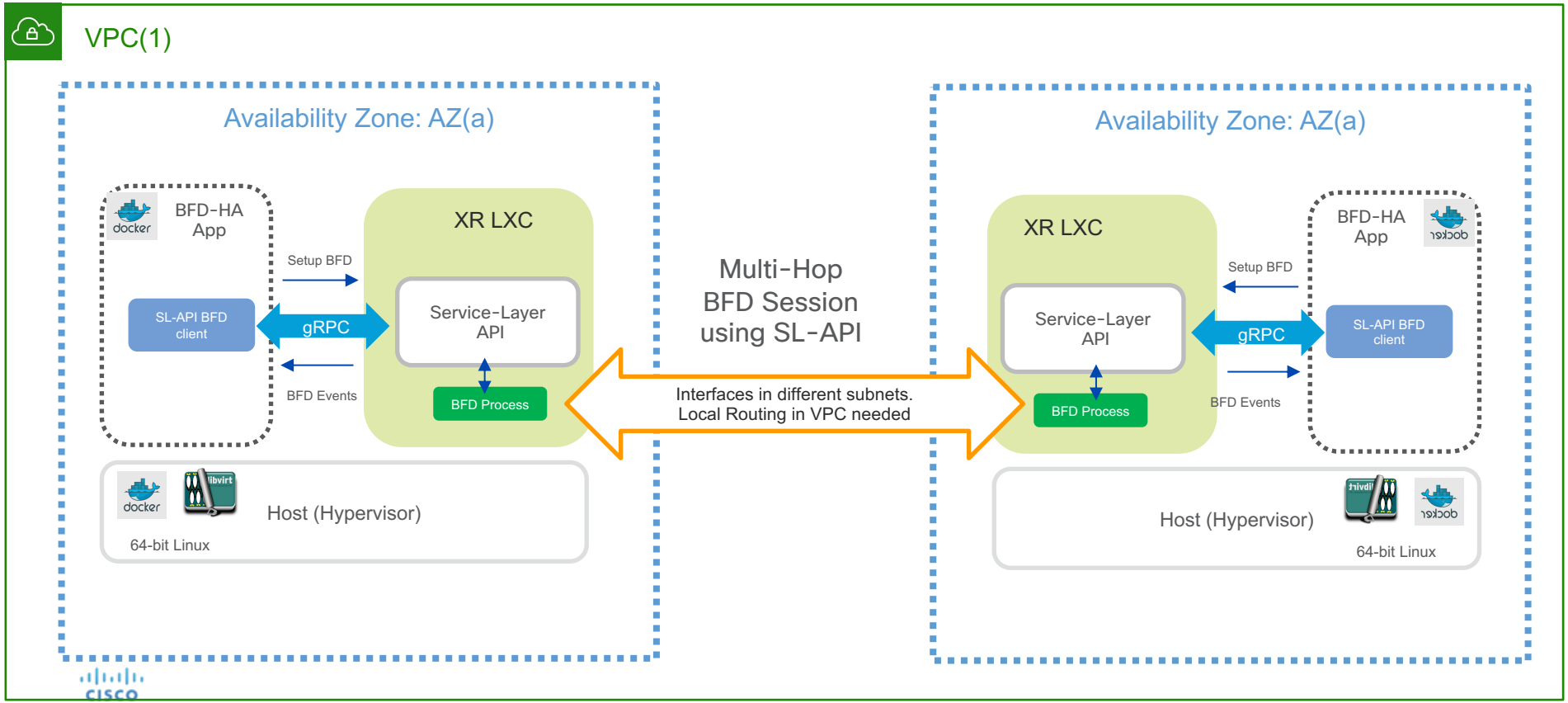
Process on vRouter to Listen to BFD events, and trigger actions

Deploying the OnBox BFD-HA App

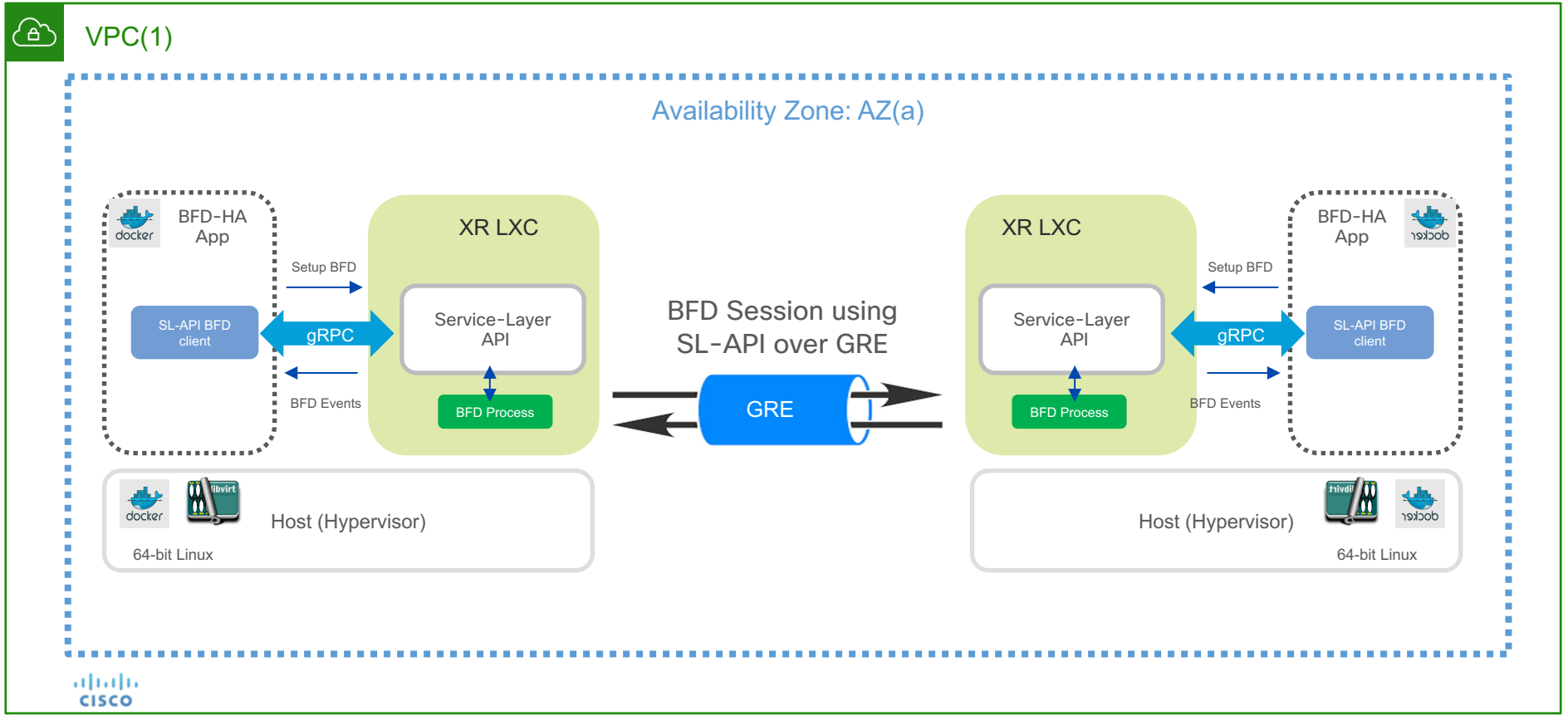
BFD-HA App Deployment Model: Single AZ, Single-Hop BFD



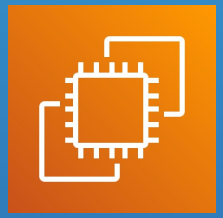
BFD-HA App Deployment Model: Different AZ



BFD-HA App Deployment Model: using BFDoGRE (Future)

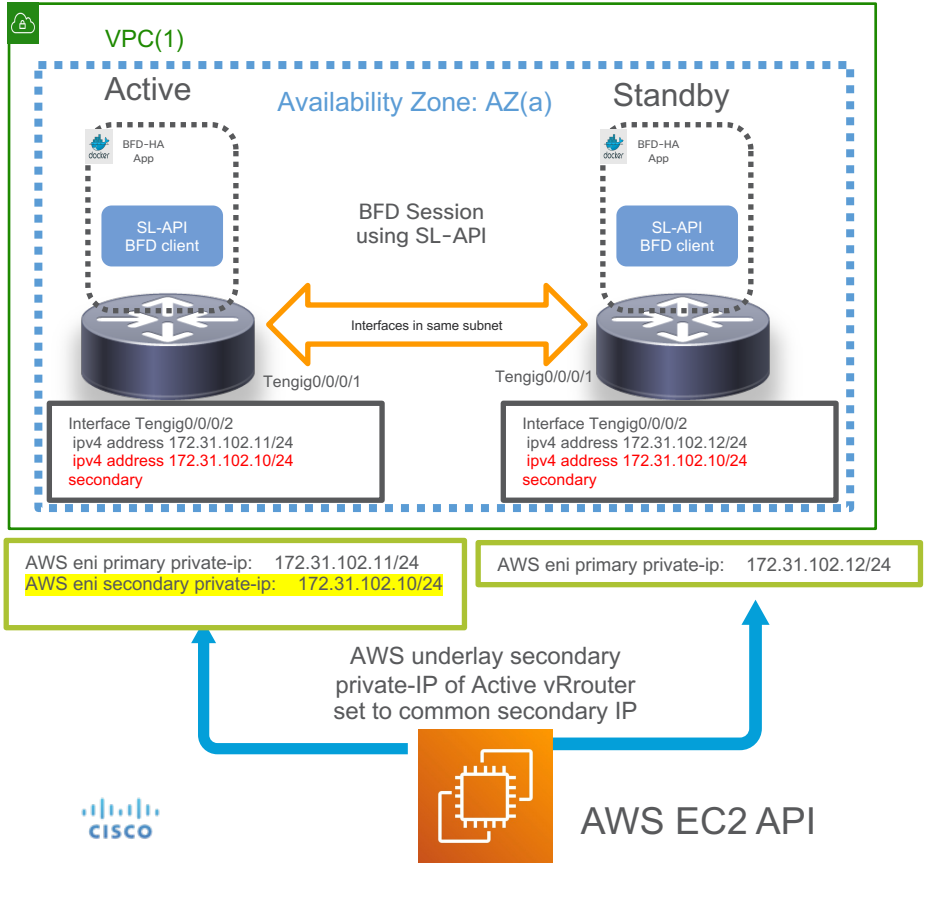


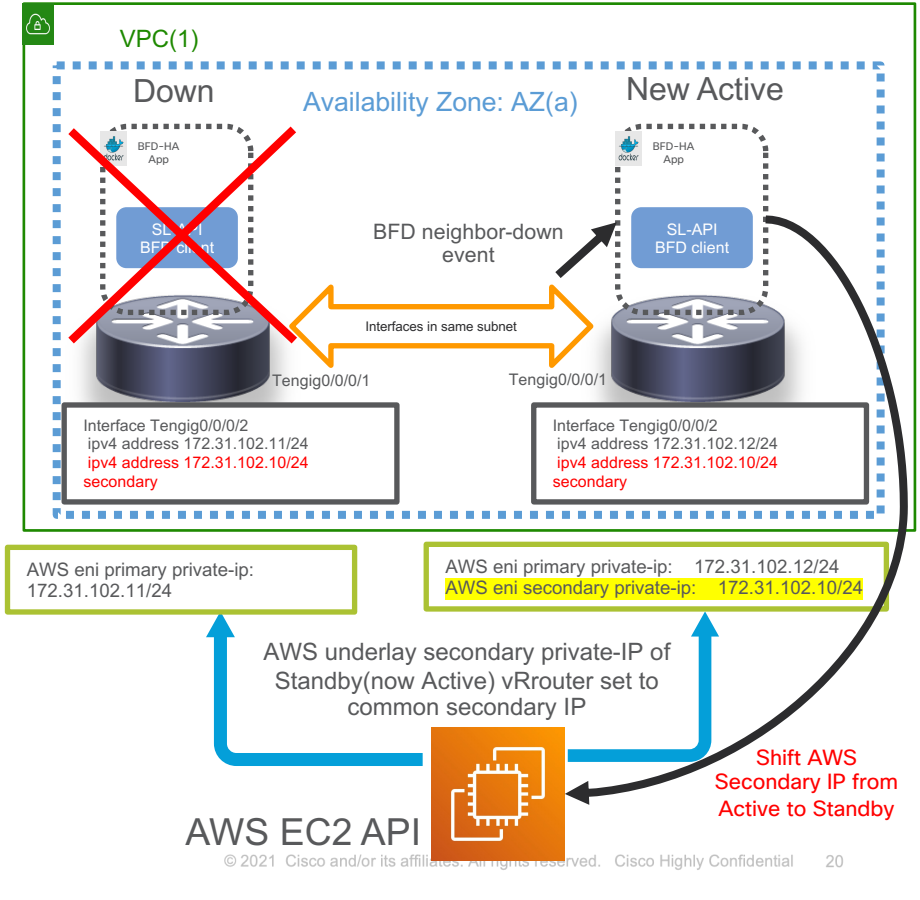
3 AWS EC2 Networking API to affect failover at the underlay



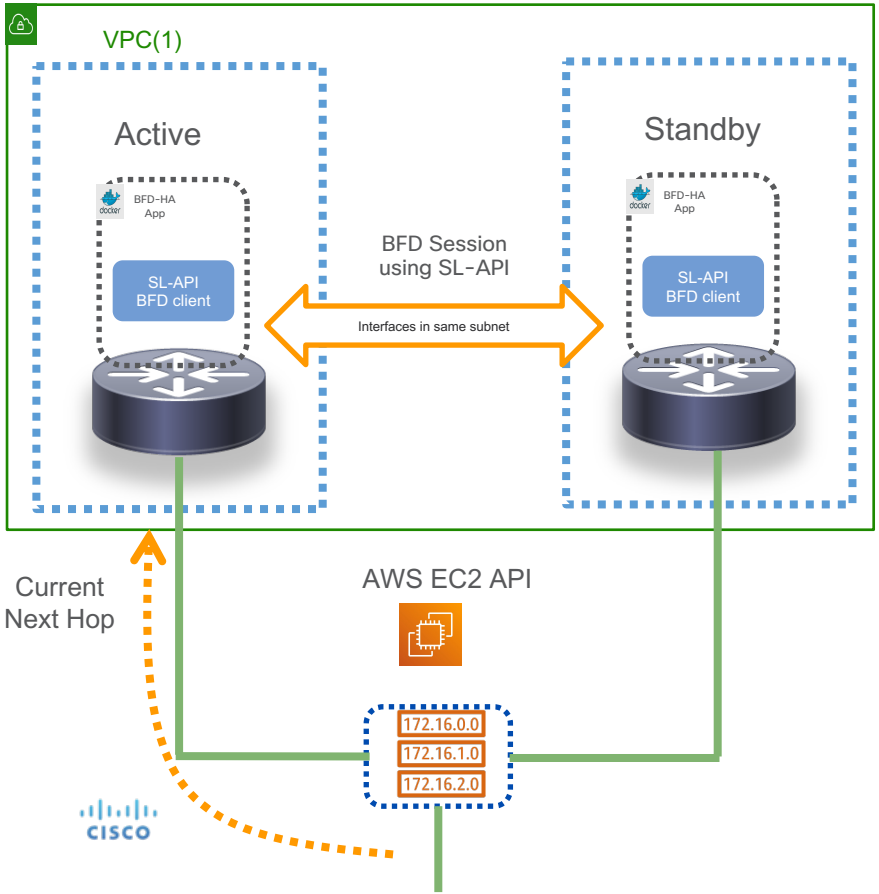
Triggering Failover

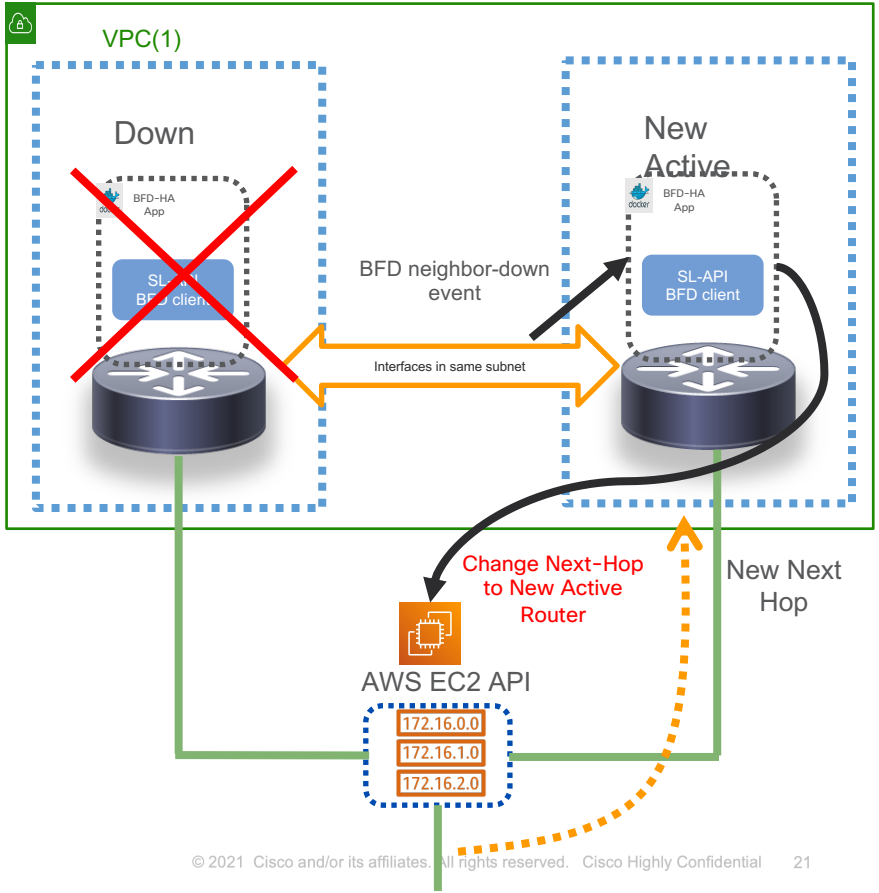
Single AZ: BFD HA App detects BFD event and triggers Secondary IP reassignment



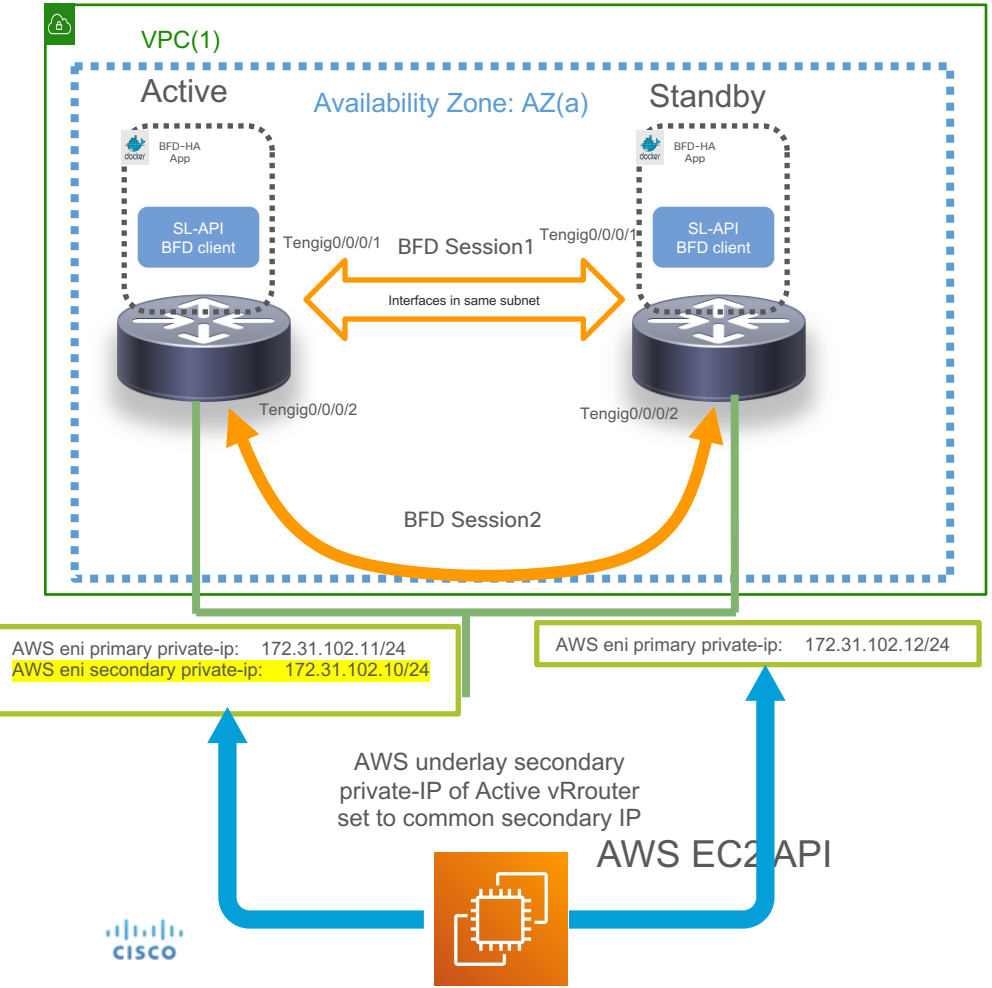


Different AZ: BFD HA App detects BFD event and triggers Route-Table change





BFD Sessions – where do they run?



BFD session 1:

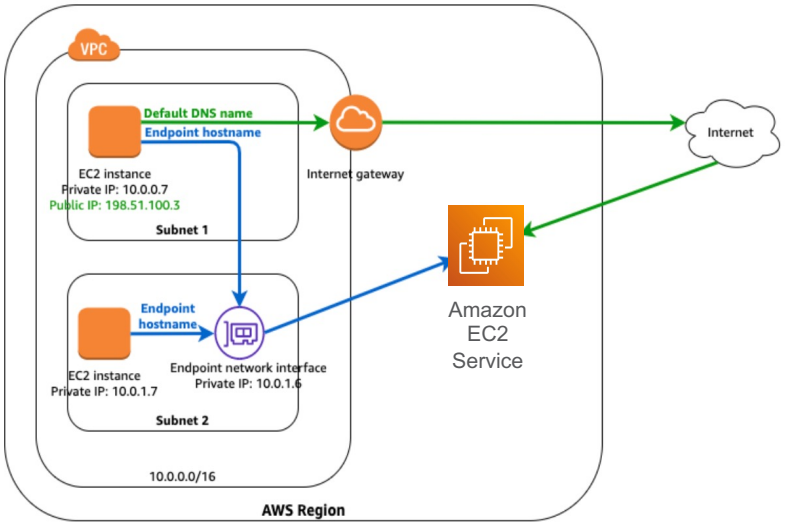
- Running over the Private-HA subnet
- Useful to detect complete failure of the active router
- Cannot be used to detect flap/failure of the Ingress interface on Active (without router failure)
- Any failover measurements with this session (using intf shut/no-shut) =
  Time taken for AWS underlay to switch traffic when secondary IP is updated.

BFD session 2:

- Running over the Ingress Subnet (where secondary-IPs are assigned)
- Useful to detect complete failure of the active router
- Also useful to detect ingress interface failure.
- Any failover measurements with this session (using intf shut/no-shut) =
  Time taken for BFD Event to be received by App
  +
  Time taken for App to invoke AWS API via endpoint service
  +
  Time taken for AWS underlay to switch traffic when secondary IP is updated.

Accessing AWS API from onBox HA App

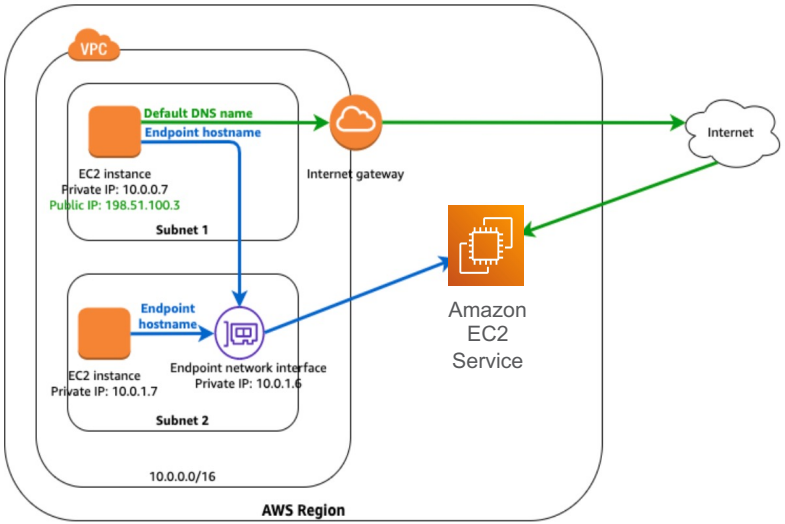
Using AWS private-link (Interface Endpoint Service)



- Private Interface Endpoint Service is created so that routers can access EC2 API without internet access
- The endpoint URL utilized by the router HA App is obtained from the interface endpoint Service configuration
- Sample boto3 resource creation for interface endpoint service:

```
resource = boto3.resource(
    service_name='ec2',
    endpoint_url='https://vpce-0786bfdf8aad8840c-
vacmlroq-us-west-2a.ec2.us-west-2.vpce.amazonaws.com',
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,
    aws_session_token=SESSION_TOKEN,
    region_name=REGION_NAME
    )
```

Using AWS private-link (Interface Endpoint Service)



- **Authentication:**
  - IAM role attached to router instances
  - App downloads the temporary credentials using metadata URLs

```
headers = {
    'X-aws-ec2-metadata-token': response.text,
}
```

```
response = requests.get('http://169.254.169.254/latest/meta-
data/iam/security-credentials/ec2access', headers=headers)
```

```
ACCESS_KEY=response.json()["AccessKeyId"]
SECRET_KEY=response.json()["SecretAccessKey"]
SESSION_TOKEN=response.json()["Token"]
```

Additional Considerations

- The BFD-HA app will accept input parameters associated with the API call (for eg. instance-id for secondary IP change in AWS OR route-table ID for next-hop change)
- The BFD-HA App will also accept configuration (in JSON format) to specify the priority of each router in the HA setup
- Pre-emption might be undesirable – so the system will not automatically revert to prevent additional data-loss during such a revert
- An API will be provided for the BFD-HA app to trigger a revert if needed.

Initial Test Results using Python HA App

Test Results:

Iteration	HA trigger Interface (shutdown)	Packet Size (bytes)	Python SLAPI-AWS client (Draft)		Packet Loss (Packets)	Ping Interval (ms)	Failover Time (ms)
			BFD event (BFD configured with interval=50ms and multiplier=3) (ms)	AWS EC2 call completion (ms)			
1	Private HA Interface TenGigE0/0/0/1	64	152	597	75	10	750
		800	99	549	8	10	80
2	Private HA Interface TenGigE0/0/0/1	64	147	519	14	10	140
		800	130	700	22	10	220
3	Private HA Interface TenGigE0/0/0/1	64	152	545	58	10	580
		800	177	587	40	10	400
4	Private HA Interface TenGigE0/0/0/1	64	128	571	45	10	450
		800	137	620	34	10	340
5	Private HA Interface TenGigE0/0/0/1	64	146	667	14	10	140
		800	174	788	50	10	500
6	Private HA Interface TenGigE0/0/0/1	64	119	550	16	10	160
		800	117	598	60	10	600
		Average -->	139.8333333	607.5833333	36.33333333		363.3333333
7	Downstream (standby IP associated) TenGigE0/0/0/2	64	171	627	98	10	980
		800	105	653	98	10	980
8	Downstream (standby IP associated) TenGigE0/0/0/2	64	143	633	117	10	1170
		800	109	620	132	10	1320
9	Downstream (standby IP associated) TenGigE0/0/0/2	64	120	544	117	10	1170
		800	156	674	89	10	890
10	Downstream (standby IP associated) TenGigE0/0/0/2	64	168	634	73	10	730
		800	143	632	72	10	720
11	Downstream (standby IP associated) TenGig0/0/0/2	64	161	572	113	10	1130
		800	125	491	60	10	600
12	Downstream (standby IP associated) TenGig0/0/0/2	64	118	594	64	10	640
		800	118	605	96	10	960
		Average -->	136.4166667	606.5833333	94.08333333		940.8333333

BFD Events are detected by App in about 139ms (TimeA)

Call to AWS API via private endpoint service takes about 600ms (TimeB)

Time taken by AWS underlay to failover traffic (post API call): Approx 350ms (TimeC) – very volatile.

Average HA failover time(Measured via PING traffic drop: average 940ms

HA failover time comes out to be approximately TimeA+TimeB+TimeC


CISCO