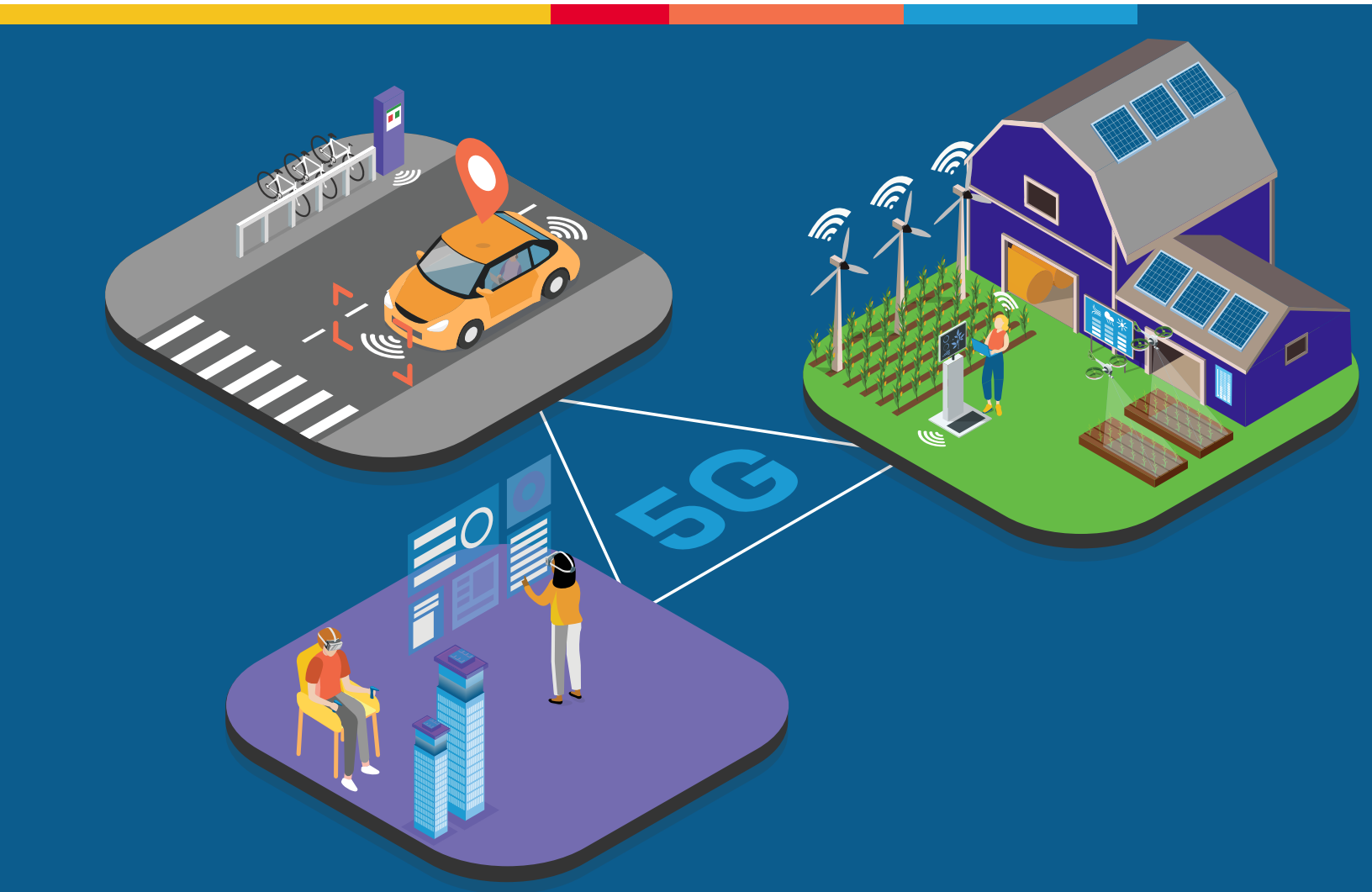
WHITE PAPER

F5 Telco Gateway for Red Hat OpenShift

Creating a scalable, resilient, and secure 5G architecture.



Introduction

Kubernetes is designed to provide services for web-oriented workloads, but its native constructs fall short of meeting the needs of Telco cloud-native network functions (CNFs) in some applications, such as:

- 5G core (5GC)
- Multi-access edge computing (MEC)

To meet the performance, access, and granular security requirements of these and similarly complex use cases, Telco-specific L4-L7 protocol support is needed to handle unsolicited traffic sent from addresses in the public Internet to private networks using various communications and network protocols, including:

- General packet radio service tunneling protocol-control (GTP-C)
- Stream control transmission protocol (SCTP)
- Diameter protocol
- User datagram protocol (UDP)
- Serial interface protocol (SIP)
- Advanced hypertext transfer protocol (HTTP/2) message routing
- Domain name system (DNS)/network address translation from IPv4 to IPv6 (NAT46)

To provide the consistent IP addressing needed for granular security risk management, these controls need to be applied on a per-namespace basis. Telcos also require these capabilities be delivered with low latency at wire speed.

F5 BIG-IP Next Service Proxy for Kubernetes (F5 SPK) is a containerized ingress and egress gateway and load balancer that delivers these functionalities. With Red Hat OpenShift's open virtual network (OVN) and Kubernetes carrier network infrastructure (CNI), they can be applied on a per-namespace basis to support use cases that provide:

- Signal traffic management and security
- Cluster level topology that hides ingress and egress
- CNF scalability, statistics, visibility, and analytics
- Transition to 5G.

Ultimately providing a CNF vendor agnostic solution. Thus, allowing multi-vendor CNFs in the same cluster rather than having dedicated Kubernetes clusters for each CNF-vendor.

Next it is described how this solution has been deployed in multi-node bare metal clusters for a major telco operator.

Overall Architecture

DATA PLANE ARCHITECTURE

Figure 1 shows a general view of F5 SPK's data plane. Its main features are:

- **OpenShift multiple external gateways.** This capability is an OVN-Kubernetes feature that uses external gateways for ingress and egress traffic for all the pods in an annotated namespace. This allows a namespace, or a group of namespaces, to be associated with a unique external gateway for ingress and egress traffic to and from outside the cluster.
- **Support for multiple external virtual LANs (VLANs) and virtual routing and forwarding (VRF) systems.** This solution applies these techniques to accomplish this:
  - Border gateway protocol (BGP) routing
  - Per-cloud native network function (CNF) secure network address translation (SNAT) pools
  - SNATs on a per VLAN basis
- **External BGP equal-cost multi-path (ECMP) ingress and egress scaling.** It doesn't require an outside solution to distribute the load across the Kubernetes nodes. F5 SPK is a single-tier solution that directly peers with the external routers.
- **Multi-node scale-out.** Each F5 SPK deployment can be dynamically scaled from one to multiple nodes, where the only limiting factor is the maximum number of ECMP paths supported by the BGP peerings. F5's SPK egress maximum ECMP paths is 64 paths.
- **Single-node scale-up.** Each node where F5 SPK is deployed can be configured to use 1-24 cores for data plane processing.
- **Single-root input/output virtualization (SR-IOV) network interfaces.** These interfaces allow low latency, wire-speed throughput and can be used for either external (to the upstream routers) or internal (to the pods) cluster connections. External and internal interfaces are configured using separate SR-IOV virtual functions (VFs). These interfaces are managed in OpenShift using the OpenShift SR-IOV Operator.
- **Advanced computing resource management.** To achieve high performance networking, F5 SPK uses the OpenShift Performance Add-on Operator to allocate CPU resources and memory for huge pages, and maintains non-uniform memory access (NUMA) alignment using the SR-IOV network interfaces.

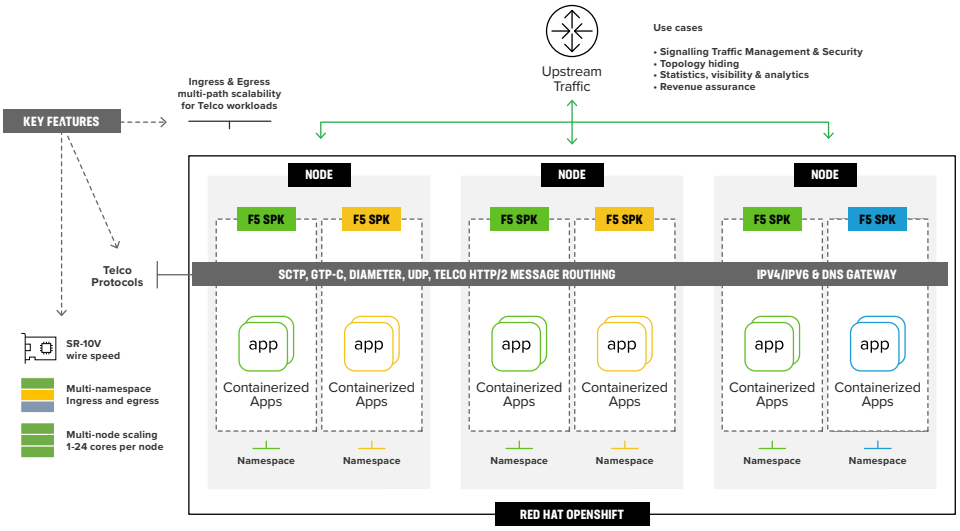


Figure 1. General data plane architecture

F5 SPK's core makes use of the widely trusted F5 BIG-IP's Traffic Management Microkernel (TMM) data plane. This allows for a high performance, dependable product from the start.

Figure 1 shows a high-level view of the solution's traffic handling. Figure 2 details traffic flows, based on the setup used in the reference customer.

Some notes about Figure 2:

- F5 SPK pods are instantiated in specific OpenShift nodes. This achieves a high level of resource isolation and guarantees NUMA alignment between the F5 SPK's used cores and the SR-IOV VFs.
- All the F5 SPK pods in a deployment run in active-active mode, no matter the number of replicas deployed.
- Symmetric ECMP traffic flows are used for stateful services. Stateful services require that each component handling the traffic manages both ingress and egress packets for a specific connections. To accomplish symmetric traffic flows, the following techniques are used:
- F5 SPK is connection-oriented and uses the auto-last hop[1] feature to achieve symmetric path returns to the external routers.
- OpenShift's OVN-Kubernetes CNI provides an ECMP symmetric flow between the nodes and the F5 SPK external gateways.
- BGP peerings between the F5 SPK external gateways and the upstream routers support bidirectional forwarding detection (BFD) for maximum Telco-grade reliability.
- F5 SPK gives full pod visibility by integrating OpenShift's OVN with Kubernetes CNI. It communicates with pods through the OVN-Kubernetes router without any address translation in the process.

Figure 2. Data plane flow

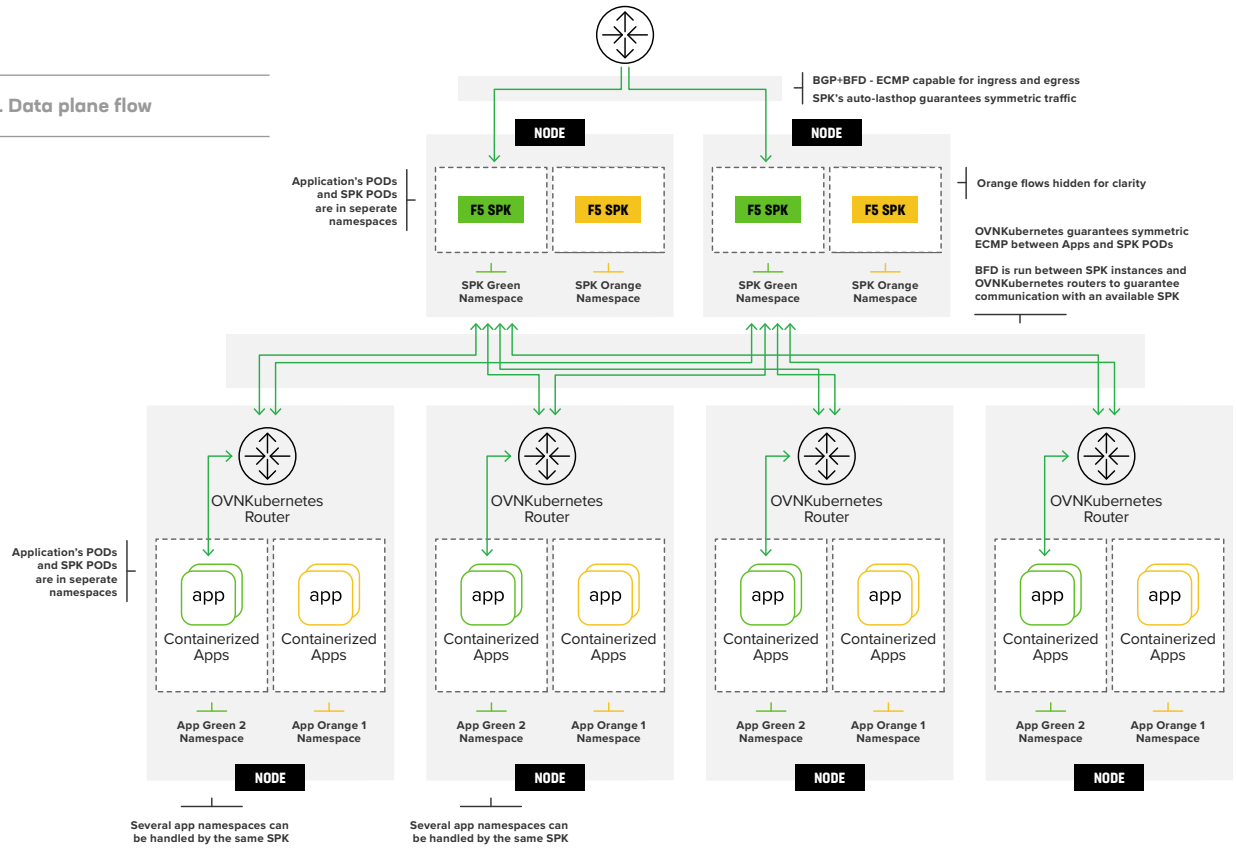
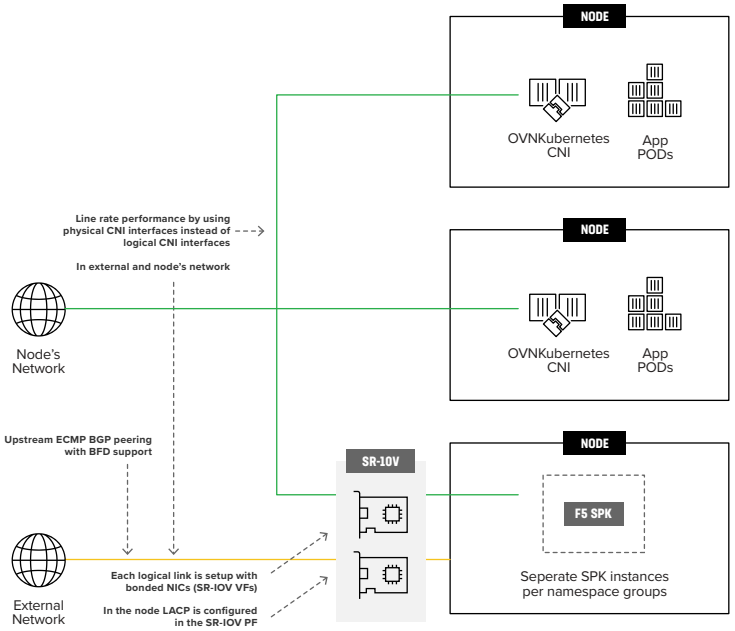


Figure 3 shows the physical layer architecture used for the nodes.

Figure 3. Physical architecture with dedicated interfaces



Traditional external gateway solutions use CNI interfaces for communication with the pods. In these arrangements, the external gateway shares the node's bandwidth with all other pods in the node. A potential problem is that sufficient throughput and latency needed to serve all pods in the cluster is not guaranteed.

SPK supports the traditional external gateway arrangement and, to assure low latency and sufficient throughput, it uses SR-IOV VF interfaces coupled with OpenShift's Multiple External Gateway capability. By bypassing software virtualization layers, this combination streamlines latency and boosts throughput, while maintaining networking isolation.

To achieve high resiliency in the SR-IOV interfaces, the F5 SPK uses link aggregation. When several F5 SPKs use the same physical network interface controller (NIC), OpenShift's nodes configuration provides link aggregation control protocol (LACP) functionality. This is done at the SR-IOV packet filter (PF) level.

CONTROL PLANE AND ANALYTICS ARCHITECTURE

F5 SPK data plane configuration

An F5 SPK controller is in charge of watching the Kubernetes API for any changes in workload pods (such as pods added or removed). The controller also watches F5 SPK's Custom Resources (CRs) for service configuration updates and applies them when necessary.

Persistent session state

A resilient and scaled-out architecture for a control plane requires session state sharing across all participant nodes. A session state includes:

- Pod persistence for Diameter protocol
- Load balancing
- NAT/NAT46 mappings

The F5 distributed Session State Manager (dSSM) component does this using Redis data stores backed by Kubernetes Persistent Volumes.

Analytics and log management

Fluentd and fluentbit, open source log and metrics tools commonly used by cloud-native applications, are used to handle the logs from the data plane, the dSSM database (DB) and the F5 SPK controller.

F5 SPK and OpenShift Integration

F5 SPK supports OpenShift 4.6 and later. This solution leverages these OpenShift features:

- **CPU Manager:** Using the CPU Manager with policy set to `static`[2] allows F5 SPK pods to be granted increased CPU affinity and exclusivity on the node.
- **Topology Manager:**[3] This component collects hints from the CPU Manager, Device Manager, and other Hint Providers to align pod resources, such as CPU, SR-IOV VFs, and other device resources, for all Quality of Service (QoS) classes on the same NUMA node. To select CPU cores that are in the same NUMA domain as the SR-IOV VFs, configure F5 SPK with a `single-numa-node` policy.
- **HugePages:** For maximum process efficiency, nodes running F5 SPK should be configured with Huge Pages allocated.
- **OVN-Kubernetes:**[4] To establish F5 SPK as the gateway for a given namespace, the following annotations are added in F5.

  SPK's TMM (data plane) pods:

  `8s.ovn.org/routing-namespaces`
  Defines the namespaces for which SPK is in charge.

  `k8s.ovn.org/routing-network`
  Defines the internal F5 SPK VLAN used as the gateway. When using multiple SPK pods, symmetric ECMP paths are set up between the OVN-Kubernetes router and the F5 SPK pods.

  `k8s.ovn.org/bfd-enabled`
  When defined, this annotation indicates that BFD is configured between F5 SPK and OpenShift nodes.

Installation and Configuration

Helm charts are the easiest and most cost-effective way to install Kubernetes containerized applications. installation uses two Helm charts:

`5-dssm`
Used to install the Distributed Session State Manager. It is shared among all the F5 SPK deployments, thus it is installed in a separate namespace.

`f5ingress`
Used to install the F5 SPK components that handle the traffic, the remaining control plane components, and the logging and analytics components. In an OpenShift cluster sharing a DSSM database, several instances of this are typical.

The following example shows an installation with an DSSM helm chart installed in the namespace `spk-utilities` and two F5 SPKs handling different namespaces. In the example, these two are deployed in namespaces `spk-data` and `spk-dns46` namespaces.

Before installing these Helm charts, perform the following steps in preparation:

```
$ helm ls-A | grep spk
spk-data        spk-data        1     2021-11-05 04:25:55.331022339-0500 CDT deployed    f5ingress-2.0.12    v2.0.12
spk-dns46       spk-dns46       1     2021-11-05 04:24:26.272528847-0500 CDT deployed    f5ingress-2.0.12    v2.0.12
spk-utilities   spk-utilities   1     2021-11-05 04:15:41.471003547-0500 CDT deployed    f5-dssm-0.11.0      v0.11.0
$
```

OpenShift platform

- Set up the CPU and Topology manager.
- Configure Node's HugePage[5] allocation.
- Allocate SR-IOV VFs for the internal and external interfaces.
- Label the nodes which have the required hardware resources to facilitate Kubernetes scheduling.

IP infrastructure

- Set up IP pre-allocation addresses for F5 SPK. These are internal IP addresses towards the cluster and external IP addresses towards the upstream BGP routers. Each F5 SPK deployment instantiates new pods (typically in different nodes) for scaling out. Each new pod needs, at minimum, one dedicated internal and one dedicated external address. If SNAT pools are required, these are also set up per F5 SPK pod.
- Set up BGP and BFD configuration for the upstream routers. The upstream routers have pre-configured BGP peerings, which place peering in an idle state until the F5 SPKs become active when scaled-out.

Helm installation

- Upload the images to a registry repository.
- Generate Google remote procedure calls (gRPC) to set up secure sockets layer (SSL) or transport layer security (TLS) keys and certificates, which will be used to secure communications between F5 SPK components.

Once an F5 SPK is deployed in a namespace, the remaining configuration is done dynamically through CRs, as shown next.

Using F5 SPK

NETWORK AND SERVICES CONFIGURATION

After Helm chart installation, F5 SPK is not yet connected to cluster networking. To set this up, use the following network management control registers (CRs).

- F5SPKVlan
  Configures TMM interface: VLANs, its own IP addresses, maximum transmission unit (MTU) size, and so on.
- F5SPKStaticRoute
  Manages the TMM static routing table.

Next, you deploy services and either expose them or provide them with egress using these CRs.

- F5SPKIngressTCP
  Manages ingress layer 4 TCP application traffic.
- F5SPKIngressUDP
  Manages ingress layer 4 UDP application traffic.
- F5SPKIngressDiameter
  Manages Diameter traffic unifying ingress and egress traffic using either TCP or SCTP and keeps sessions persistent using the SESSION-ID attribute value pair (AVP) by default.
- F5SPKIngressNGAP
  Balances ingress datagram loads for SCTP or NG application protocol (NGAP) signaling.
- F5SPKEgress
  Enables egress traffic for pods using SNAT or DNS/NAT46. DNS cache and rate limiting parameters can be configured.
- F5SPKSnatpool
  Allocates IP addresses for egress pod connections.
- **F5SPKDNSCache**
  Provides high-performance, transparent DNS resolution and caching for the F5SPKEgress resources.
- **F5SPKPortList and F5SPKAddressList**
  Creates sets of ports and addresses, respectively, to make creating and updating services easier.

These CRs are available with SPK v1.4. These features will be expanded, so check the latest documentation in https://clouddocs.f5.com/service-proxy/latest/.

An F5 SPK controller component watches for Kubernetes events such as changes in the setup of these CRs and configures the F5 SPK accordingly.

Application hairpinning

The F5 SPK application hairpinning feature is used to differentiate between internal and external clients. A selected set of internal clients accesses an F5 SPK Service with the same domain name or IP address as that of another F5 SPK Service, which is used by external clients using different configurations. The key difference between the two types of connections is that internal clients are connected using SNAT and external clients are not. This is done by installing two F5 SPK CRs of the same type, for example F5SPKIngressTCP, with each CR enabled on a selected VLAN or VLAN list.

IPV6 SUPPORT

F5 SPK fully supports IPv4/IPv6 dual-stack networking as implemented in Kubernetes v1.21 or later. F5 SPK's DNS46/NAT46 feature, however, does not rely on Kubernetes IPv4/IPv6 dual-stack and therefore, it can be used with earlier versions of Kubernetes.

DNS46/NAT46 TRANSLATION

The adoption of IPv6 in new 5G deployments has created a need to interact with older IPv4 single stack components and services. F5 SPK's DNS46/NAT46 provides this interoperability, easing the adoption and transition between IPv4 and IPv6 services. This solution allows IPv4 applications to access any IPv6 application on demand, without requiring reconfiguration.

Here's how it works: Applications use DNS to locate services. F5 SPK, acting as an egress gateway, monitors these requests and detects DNS replies for which only IPv6 AAAA records return. When this happens, F5 SPK dynamically appends an A record for IPv4 applications to use.

The dynamically created IPv4 A record is stored in the F5 SPK dSMM in-memory database with the IPv6 AAAA record mapping. This information is used by the F5 SPK NAT46 functionality to automatically create a NAT mapping between the actual service's IPv6 address and the dynamically allocated IPv4 address.

Figure 4 shows how the solution is configured for a selection of namespaces.

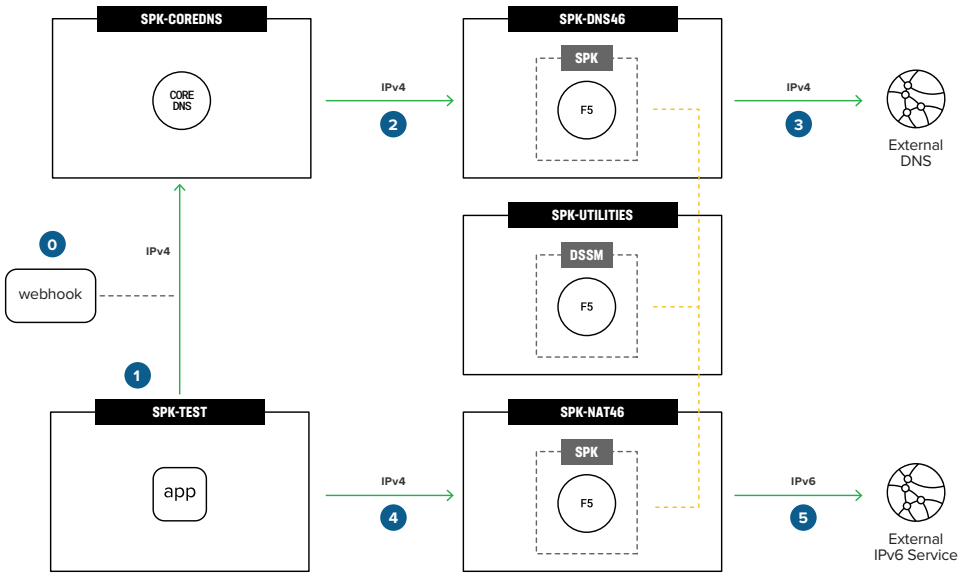


Figure 4. This solution applied to a selection of namespaces

To apply the DNS46/NAT46 functionality to a subset of namespaces, a dedicated instance of CoreDNS is deployed, using the CoreDNS image bundled with OpenShift. This instance is deployed into the spk-coredns namespace, as shown in Figure 4. In this arrangement, only DNS requests originating from an F5 SPK-deployed coreDNS are handled by F5 SPK's DNS46/NAT46 functionality.

For an application's pods to automatically use the spk-coredns, the pod's DNS resolver configuration (`/etc/resolv.conf`) must be updated with the spk-coredns service address. This is then handled transparently by a webhook when an application's namespace has the label `dns-spk=enabled` applied.

The steps an IPv4 service follows to access an IPv6 service are:

1. When a pod is created in a namespace labeled `dns-spk=enabled`, the resolver configuration is updated to point to a separate DNS service, which goes through an F5 SPK gateway with the DNS46/NAT46 functionality enabled.
2. The pod makes a DNS IPv4 request that is routed to the spk-coredns service.
3. When the reply to the DNS request is not found in the spk-coredns cache, the request is forwarded to the external DNS resolver, which goes through the F5 SPK with DNS46/NAT46 enabled.

4. When the DNS reply includes only IPv6 AAAA records, F5 SPK dynamically appends an A record for IPv4 applications to use. The DNS response with the attached IPv4 A record is returned that backtracks through the previous steps. The new A record is stored in the dSMM database and will be used in future DNS46/NAT46 translations. At this point, the pod has an IPv4 address of an IPv6 service. Also, the F5 SPK is configured as the external gateway for the pod's namespace and has the DNS46/NAT46 feature enabled.
5. The client then connects to the service—as with any IPv4 native service—and OVN routes the connection to the F5 SPK external gateway. Figure 4 shows an example in the spk-nat46 namespace.
6. F5 SPK applies the DNS46/NAT46 translation indicated in the dSMM database.

This completes the end-to-end flow.

Conclusion

This white paper introduces a scalable and dependable high performance gateway solution that delivers the granular ingress and egress controls in Kubernetes-based deployments that Telcos need. It builds on the unique potential of OpenShift external gateways by making full use of OpenShift capabilities—an industry first. Use cases that particularly benefit include 5GC and MEC. Plus, the F5 SPK solution can dynamically translate IPv4 to IPv6 network addresses, which solves the problem of mixed IPv4 and IPv6 deployments. The result is a gateway solution flexible enough to adapt to new and evolving Telco needs while offering interoperability with pre-5G services.

[1] F5, Overview of the Auto Last Hop setting, found at https://support.f5.com/csp/article/K13876

[2] Kubernetes, CPU Management Policies Configuration, found at https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/#configuration

[3] Kubernetes, How Topology Manager Works, found at https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/#how-topology-manager-works

[4] Red Hat OpenShift, About the OVN-Kubernetes default Container Network Interface (CNI) network provider, found at https://docs.openshift.com/container-platform/4.9/networking/ovn_kubernetes_network_provider/about-ovn-kubernetes.html

[5] Kubernetes, Manage HugePages, found at https://kubernetes.io/docs/tasks/manage-hugepages/scheduling-hugepages/