Tungsten Fabric Architecture

Detailed Technical Description of the Virtual Networking and Security Platform

Introduction
- Use Cases
- Tungsten Fabric Components

Three Tungsten Fabric Worlds
- Tungsten Fabric Working with An Orchestrator
- Interaction With An Orchestrator

Architecture Details of vRouter
- Tungsten Fabric Implementation on top of a vRouter
- Packet Flow Between VMs in The Same Subnet
- Packet Flow Between VMs in Different Subnets

Service Chaining
- Basic Service Chain
- Analytics Service Chain
- Scale-out Service

Application-based Security Policies
- Creating an Application Policy
- Application Policy Enforcement
- Advanced Application Policies

Deployment Options for vRouter
- Kernel Mode vRouter
- DPDK vRouter
- SR-IOV (Single Root – Input/Output Virtualization)
- Smart NIC vRouter

Tungsten Fabric Collection and Analytics

Tungsten Fabric Deployment

Tungsten Fabric APIs
- Tungsten Fabric Native REST API
- Python Bindings
- Tungsten Fabric CLI

Orchestrators
- OpenStack with Tungsten Fabric
- Kubernetes Cluster with Tungsten Fabric
- Mesos Cluster with Tungsten Fabric
- Nested Kubernetes with OpenStack as a Cluster

Connecting to Physical Networks
- Physical Underlay
- Bare Metal
- Physical Gateways

Introduction

This document describes how Tungsten Fabric provides a scalable virtual networking platform that works with a variety of virtual machine and container orchestrators and can integrate with physical networking and compute infrastructure. Tungsten Fabric uses overlay networking and runs on top of any IP fabric underlay. It can also integrate with seamlessly connect workloads in different orchestration domains. E.g. Virtual machines managed by VMware vCenter and containers managed by Kubernetes.

Tungsten Fabric provides a full set of networking services and a full suite of cloud-ready security. Groups of network cloud services can be chained together to create virtual topologies that have been re-ordered and re-tuned. E.g. VMware vCD is networking multi-tenant workloads with firewalls, NAT, and load balancers. Tungsten Fabric provides a highly scalable virtual networking platform that is designed to support high-performance networks in the largest environments while supporting multiple orchestrators simultaneously.

Since there are very few datacenter deployments that are truly "greenfield", there are nearly always requirements to integrate workloads deployed on new infrastructure with legacy infrastructure and workloads that are not virtualized. For this reason, interoperability and of scenarios for deployments where there are cloud infrastructure with the datacenter, and where coexistence with existing infrastructure is also needed.

Use Cases

The following common use cases are covered in this document.

- Enable Platform as a Service and Software as a Service with high scalability and multi-tenancy for enterprise or service provider networks
- Virtual networking with Kubernetes container management clusters, including Kubernetes-as-a-service
- Bare metal or existing virtualized environment running VMware vCenter to use Tungsten Fabric virtual networking between virtual machines
- Integrate with existing bare metal or virtualized network overlay gateway routers with BGP signaling with networking overlays, and directly forward packets between underlay network.

These use cases can be deployed in any combination in address the specific requirements and priorities of deployment scenarios. The main factors around Tungsten Fabric are illustrated below.

Tungsten Fabric Architecture
Virtual Machines
Containers
Bare Metal Servers
Hypervisor
Linux
Orchestration
Management
Analytics
Control Nodes
Analytics Nodes
Database Nodes
Orchestrator Plugins
Analytics Collectors
Configuration Management
Analytics Management
Analytics Rules
Analytics Alarms
Analytics Topology
Analytics Traffic Statistics
Analytics Logs
Analytics Flows
Analytics Sessions
Analytics Packets
Analytics Events
Analytics Exchange
Analytics Database
Analytics Query Engine
Analytics Rules Engine
Analytics Alarm Generator
Analytics Topology Manager
Analytics Collector
Analytics Alarm Gen
Analytics Snmp Collector
Analytics Topology
Analytics Alarm Node Manager
Analytics Snmp Node Manager
Analytics Alarm Node Mgr
Analytics Snmp Node Mgr
Analytics Vrouter Agent
Analytics Vrouter Agent Manager
Analytics Vrouter Agent Mgr
Analytics Vrouter Agent Module
Analytics Vrouter Agent Module Manager
Analytics Vrouter Agent Module Mgr
Analytics Vrouter Agent Module Manager Mgr
Analytics Vrouter Agent Module Manager Manager
Analytics Vrouter Agent Module Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager
Analytics Vrouter Agent Module Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Manager Mgr
Analytics Vrouter Agent Module Manager
Analytics Vrouter Agent Module Manager Mgr
Analytics Vrouter Agent Module Manager
Analytics Vrouter Agent Module Manager Mgr
Analytics Vrouter Agent Module Manager
Analytics Vrouter Agent Module Manager Mgr
Analytics Vrouter Agent Module Manager

is secure policy. In the example in the diagram, traffic on ports 80 and 8080 flows to pices-service, traffic on port 9000 flows to pices-data, and traffic on port 443 is dropped. Instead of a firewall VM, vmware may have a different implementation for this.

Active-Standby Service Chains

In some use cases, it is desirable to run a service in active-standby mode. In a typical active-standby service chain, but if there are issues detected with that chain, then traffic should be switched to a backup service chain. In this case, where the standby service chain is hosted on a new kubernetes geographic location.

Active-standby configuration is achieved in two steps in Tungsten Fabric. First a route aggregation is used to ensure that traffic destined to the active service chain and the standby service chain uses the preferred active chain inspection. Secondly, a health check is configured on each chain that can fail service instances are inactivated, or that a destination on the other chain is reachable. When a service chain is inactive, its routes are not exported, so that traffic to some service chain is unbiased and traffic will flow through the standby.

Application-based Security Policies

Conventional firewall policies contain rules based on individual IP addresses or subnet addresses, ports and protocols. This is a very low level of abstraction and it is extremely difficult to manage when being created and difficult to understand when troubleshooting. This is because the IP address of server or VM doesn't relate to the application. Contrailnetwork policy, based on the ideas of Juniper Networks, uses a higher level of abstraction that has less data content and thereby is more fun application to development and troubleshooting.

It is a requirement in the enterprise that this layers of each instance of an application can only communicate with the next layer in the same instance. This requires a separate policy for each of the application instances, as shown. When troubleshooting an issue, the policy ensures that the issue can be isolated to a single application instance and when time is less important, a new firewall rule must be written.

Application Tags

The Tungsten Fabric provider supports security policies based on tags that can be applied to projects, networks, vRouters, VMs and interfaces. The tags propagate to the compute node where they are instantiated. Security policies are then created that are applied as a cover layer of the underlayment Security lists (ACLs) that are applied to the vRouter agent. Tags have a name and a value. A number of tag types are supplied as part of the Tungsten Fabric distribution. Typical uses for the tag types are shown in the table below.

Tag Name	Typical Use	Examples
application	Identifies a group of VMs that run a set of software instances of different types to support a single application, such as a financial services. Can correspond to a head-data	CAMP stack, Hadoop cluster, set of Docker hosts, Openstack Tungsten Fabric cluster
tier	A set of software instances of the same type that are deployed on different servers for fault or same location. The number of such instances is typically dynamic and managed by orchestration software	Apache servers, Cassandra clusters, Hadoop datanodes, OpenStack services, containers
deployment	Indicates the purpose of a set of VMs. Usually applies to all the VMs in a stack	development, test, production
site	Indicates the location of a stack, usually of the compute nodes. Can be a hierarchy	US East, London, Datacenter 1, Rack 10
custom	Open tags can be created as needed	Customer name
label	Multiple labels can be applied to objects. Can be used for a variety of data flows within and between stacks	customer sessions, Hadoop jobs, Cassandra clusters

As shown in the table, in addition to the tag types that are provided with Tungsten Fabric users can create their own custom tag types as needed, and there is a .net JavaTag which can be used to mirror these tyes deep flows.

Creating an Application Policy

Application policies contain rules based on tag values and service groups, which are sets of ports. The rules are applied to traffic between two endpoints, which can be "application". In the application stack, each subpoint is tag of type "application" and the component of the application. This is illustrated below.

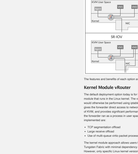
In this example, the application is tagged PricingAPI, and this tenant has tagged , web, app1 ,app2 ,db1 ,Service groups have been created for the traffic flows into the application components and between the tiers. Application policies have been created for "pricing-web", "pricing-app1", "pricing-app2" "pricing-db1" and allow just the required traffic. Three Application policy sets ar then associated with the application tag, ,PricingAPI-Prod and PricingAPI-Test and PricingAPI-Dev. The Application policy sets are associated with the tag, so the tag is touched and the tags applied to the various VMs in the Tungsten Fabric compute node. The vrouter the controller is calculate which routes need to be sent to each vRouter to enforce the policies and sends the routes to the vRouter agents. The vRouter agents then create the security underlayment, the routing tables in each vRouter and the ACLs to enforce the policies.

Field	RIP	Source	Destination	Ports	Route
S1	Apache	10.1.0.0/24 | 10.1.1.0/24 | 10.1.2.0/24	10.1.3.0/24	8080	Interface to VM1
			10.1.4.0/24	443	Interface to VM2
			10.1.5.0/24	443	Interface to VM3
S2	Apache	10.1.3.0/24 | 10.1.4.0/24 | 10.1.5.0/24	10.1.3.0/24	8080	Interface to VM1
			10.1.4.0/24	443	Interface to VM2
			10.1.5.0/24	443	Interface to VM3
S3	Apache	10.1.6.0/24 | 10.1.7.0/24 | 10.1.8.0/24	10.1.6.0/24	8080	Interface to VM1
			10.1.7.0/24	443	Interface to VM2
			10.1.8.0/24	443	Interface to VM3

The vrouters and VMs are named here for the ten Bear-key ion. In deploy-ing relationships between entity routes and their result ion usually be ac-complish. We can seen in the table, the routes enable traffic only ac-ce specified in the application policy, but the ACLs are applied to the vrouter-connected tins network underlay-based thread-cross that the vRouter is able to apply.

Controlling Flows Between Deployments

Having successfully created an application stack, tier and set of tags,here when another deployment of the stack is created, as shown below.



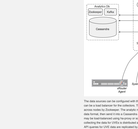
There is nothing in the original policy that prevents traffic flowing between a tier in one deployment into a tier in a different deployment. This behavior can be modified by tagging each component of each stack with a ,deployment tag, and by adding a ,match deployment, to the application policy. This ensures that the data only ships the deployment tags match. The updated policy is shown in below.

Now the traffic flows conform to the strict requirements that traffic only flows between components within the same stack.

Advanced Application Policies

Applying tags of different types allows the security policies to be applied in multiple dimensions. For example, the application tag can be used to segregate traffic, and policies can segment traffic within individual clusters based on site, but allow sharing of the database tier across a site.

It includes stacks are deployed within the same combination of sites and deployments, a single policy can be used to control the application and database tiers. Application tags can be used to create the required relationship, as seen in the diagram, below.



The application policy features in Tungsten Fabric provide a very powerful enforcement framework, while simultaneously enabling dramatic simplification of policies, and reduction in their number.

Deployment Options for vRouter

There are several deployment options for vRouter that offer different benefits and ease of use:

- **Kernel Module** - This is the default deployment mode.
- **DPDK** - Forwarding acceleration is accomplished using Intel library.
- **SR-IOV (Single Root - Input/Output Virtualization)** - Intel specification.
- **SmartNIC** - vRouter forwarder is implemented in a programmable NIC.

Three options are illustrated below.

The features and benefits of each option are described below.

Kernel Module vRouter

The default deployment option today is for the vRouter forwarder to be implemented in kernelspace mode in the Linux Kernel. The vRouter implements networking functionality that would otherwise be performed using iptables or Open vSwitch. Running in the kernel ensures that packets need not be moved to user space for most of the forwarding function. Use of a NIC jumo provides significant performance improvement over what can be achieved by forwarding packets in user space. Among the optimizations that have been implemented are:

- TCP segmentation offload
- Large receive offload
- Receive side scaling with flow control processing

The kernel module approach allows users to implement network virtualization using Tungsten Fabric with minimal dependency on underlying server and NIC hardware. However, only specific kernel versions are supported.

DPDK vRouter

Intel has developed a port to DPDK, from Intel, a set of libraries and drivers that allow data packet processing to take place in kernel-free spaces in a NIC offload going through the KVM network stack. A version of the vRouter forwarder is available that runs in user space and uses the DPDK library. The DPDK implementation provides higher throughput compared to the kernel module with unmodified VMs, and even better performance can be achieved if the guest VMs also have DPDK enabled.

The DPDK vRouter can be tuned to the Linux scheduler to provide dedicated cpu cores to packet processing. The DPDK vRouter can also be configured to use huge memory (HM). Use of HM, as the name implies, allows for faster memory access.

SR-IOV (Single Root - Input/Output Virtualization)

In the SR-IOV deployment, the vRouter data plane runs as a VM and can then access the hardware NIC directly using the Intel SR-IOV specification. This approach is ideal for running multiple clients as if each has sole access. Intel like a hypervisor does for CPU. A single PCI Express (PCIe) interface can be shared by multiple virtual machines (or networking stack, which leads to enhanced performance. SR-IOV is also used when the VM is performing a gateway function between a physical network and virtual network, but it requires that the underlay network fabric be aware of the MAC addresses being used in the Fabric virtual interfaces and don't participate in network protocols and network services.

Smart NIC vRouter

Some new NICs are becoming available which are programmable. The Tungsten Fabric vRouter forwarder functionality can be implemented on these new NICs, and the things that would normally be done by the server CPU can be offloaded to the network card. Tungsten Fabric virtual networks and don't participate in network protocols and network services.

Tungsten Fabric Collection and Analytics

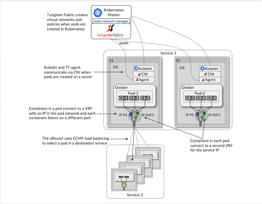
Tungsten Fabric collects information from the cloud infrastructure (compute, network and storage) and from the network. It uses the data to provide a real-time view of the cloud, troubleshooting and capacity planning. The data is collected in a variety of formats such as syslogs, structured messages (known as Sandesh), IpFix, Sflow and SNMP. Operators can also add their own data collectors and parsers. The collected data and the analytics it provides are modeled on Open Stack Ceilometer (OSP) and the attributes in a UVE may come from a variety of sources in different formats.

The architecture for analytics collection is shown in the figure below.

REST API for UVE protocol
Sandesh for control messages
Analytics
Analytics Node
Cassandra
Kafka
Zookeeper
Collector
Query Engine
Control Node
Control Node
Control Node
Compute Node
Compute Node
Compute Node
vRouter Agent
vRouter Agent
vRouter Agent
Control Node
Control Node
Control Node
Compute Node
Compute Node
Compute Node

Tungsten Fabric virtual networking can be integrated in a Kubernetes environment to provide a range of multi-tenant networking features in similar fashion as with OpenStack.

This configuration of Tungsten Fabric with Kubernetes is shown below.



The architecture for Tungsten Fabric with Kubernetes orchestration and Docker containers is similar to OpenStack and KVM/QEMU, with the vRouter running in the host Linux OS and containing VRFs with virtual network forwarding tables. All containers in a pod share a networking stack with a single IP address (IP-1, IP-2 in the diagram), but listen on different TCP or UDP ports, and the interface of each networking stack is connected to a VRF at the vRouter. A process called _kube-network-manager_ listens for network-related messages using the Kubernetes _kubs_ API and sends these into the Tungsten Fabric API. When a pod is created on a server, there is communication between the local _kubelet _and the vRouter agent via the Container Network Interface (CNI) to connect the new interfaces into the correct VRFs. Each pod in a service is allocated a unique IP addresses within a virtual network, and also a floating IP address which is the same for all the pods in a service. The service address is used to send traffic into the service from pods in other services, or from external clients or servers. When traffic is sent from a pod to a service IP, the vRouter attached to that pod performs ECMP load-balancing using the routes to the service IP address that resolve to the interfaces of the individual pods that form the destination service.

When traffic needs to sent to a service IP from outside the Kubernetes cluster, Tungsten Fabric can be configured to create a pair (for redundancy) of ha-proxy load balancers which can perform URL-based routing to Kubernetes services, preferably using floating IP addresses to avoid exposing the internal IP addresses of the cluster. These externally visible service addresses resolve to ECMP load balanced routes to pods of the service. Kubernetes proxy load-balancing is not needed when Tungsten Fabric virtual networking is used in a Kubernetes cluster.

Other alternatives for providing external access include: using a floating IP address which is associated with a load balancer object, or using a floating IP address associated with the service.

When services and pods are created or deleted in Kubernetes, the kube-network-manager process detects corresponding events in the k8s API, and it uses the Tungsten Fabric API to apply network policy according to the network mode that has been configured for the Kubernetes cluster. The various options are summarized in the following table.

Networking Mode	Network Policy	Effect
Kubernetes default	Any-to-any, no tenant isolation	Any container can talk to any other container or service
Namespace isolation	Kubernetes namespaces map to projects in Tungsten Fabric	Containers within a namespace can communicate with each other
Service isolation	Each pod is in its own virtual network and security policy is applied so that only the service IP address is accessible from outside the pod	Communication within a pod is enabled, but only the service IP address is accessible from outside a pod
Container isolation	Zero-trust between containers in the same pod.	Only specifically allowed communications between containers are enabled, even within a pod. Only specific pod to specific services may be enabled.

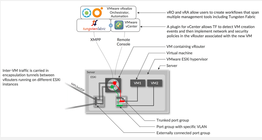
Tungsten Fabric brings many powerful networking features to the Kubernetes world, in the same way that it does for OpenStack, including:

- IP address management
- DHCP
- DNS
- Load balancing
- Network address translation (1:1 floating IPs and N:1 SNAT)
- Access control lists
- Application-based security

Tungsten Fabric and VMware vCenter

VMware vCenter is in widespread use as a virtualization platform, but requires manual configuration of a network gateway in order to achieve networking between virtual machines that are in different subnets and with destinations external to a vCenter cluster. Tungsten Fabric virtual networking can be deployed in an existing vCenter environment to provide all the networking features that were listed previously, while preserving the workflows that users may have come to rely on to create and managed virtual machines using the vCenter GUI and API. Additionally, support has been implemented for Tungsten Fabric in vRealize Orchestrator and vRealize Automation so that common tasks in Tungsten Fabric such as creation of virtual networks and network policies can be included in workflows implemented in those tools.

The architecture for Tungsten Fabric working with VMware vCenter is shown in the diagram, below.



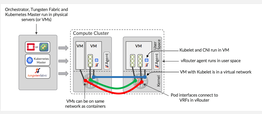
Virtual networks and policies are created in Tungsten Fabric, either directly, or using TF tasks in vRO/vRA workflows.

When a VM is created by vCenter, using its GUI or via vRO/vRA, the vCenter plugin for Tungsten Fabric will see a corresponding message on the vCenter message bus, and this is the trigger for Tungsten Fabric to configure the vRouter on the server that the VM will be created on. Each interface of each VM is connected to a port group that corresponds to the virtual network that the interface is in. The port group has a VLAN associated with it that is set by the Tungsten Fabric controller using the "VLAN override" option in vCenter, and all the VLANs for the port groups are sent through a trunked port group into the vRouter. The Tungsten Fabric controller maps between the VLAN of a interface to the VRF of the virtual network that contains that subnet. The VLAN tag is stripped, and route look up in the VRF is performed as described in the section XXX

Using Tungsten Fabric with vCenter gives users access to the full range of network and security services that Tungsten Fabric offers as described in earlier in this document, including zero-trust micro-segmentation, proxy DHCP, DNS and DHCP which avoids network flooding, easy service chaining, almost unlimited scale, and seamless interconnection with physical networks.

Nested Kubernetes with OpenStack or vCenter

In the previous section, the it was assumed that the KVM hosts where the containers run had been provisioned beforehand by some means. An alternative is to use OpenStack or vCenter to provision VMs in which the containers run, and for Tungsten Fabric to manage virtual networking between VMs created by OpenStack or vCenter and containers created by Kubernetes. This is illustrated below.



The orchestrator (OpenStack or vCenter), Kubernetes Master and Tungsten Fabric are running in a set of servers or VMs. The orchestrator is configured to manage the compute cluster with Tungsten Fabric, so there are vRouters on each server. VMs can be spun up and configured to run Kubelet and the CNI plugin for Tungsten Fabric. These VMs become available for the Kubernetes master to run containers in, with networking managed by Tungsten Fabric. Since the same Tungsten Fabric is managing the networks for both the orchestrator and Kubernetes, seamless networking is possible between VMs, between containers, and between VMs and containers.

In the nested configuration, Tungsten Fabric delivers the same levels of isolation as described previously, and it is possible for multiple Kubernetes Masters to co-exist and for multiple VMs running Kubelet to run on the same host. This allows a multi-tenant Kubernetes container service to be offered.

Connecting to Physical Networks

In any datacenter, there is a need for some VMs to access external IP addresses, and for users outside the datacenter to access some VMs via public IP addresses. Tungsten Fabric provides several ways to achieve this:

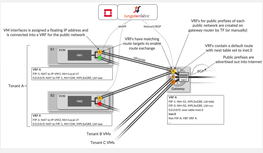
- VPN connection to a BGP-enabled gateway
- Source NAT in the vRouter
- Local gateway in the vRouter to the underlay fabric

Each of these is applicable in different use cases, and have varying dependencies on configuration of external devices and networks.

The methods of connection to external networks are described in the following sections.

BGP-Enabled Gateway

One way of achieving external connectivity is to create a virtual network using a range of externally routable IP addresses, and to extend the network to a gateway router. When the gateway router is a Juniper MX router, the configuration on the device can be done automatically by Tungsten Fabric. This is illustrated below.



Network A is defined in Tungsten Fabric, and contains a subnet of publicly addressable IP addresses. This public virtual network is configured in Tungsten Fabric to extend to the gateway router, which, when using Tungsten Fabric Device Manager results in automatic creation of a VRF on the gateway with route target matching that of the virtual network (e.g. VRF labeled A). Tungsten Fabric configures this VRF with a default route that causes route lookup for traffic arriving in the VRF from the Tungsten Fabric cluster to occur in the main inet.0 routing table (which will contain routes to public destinations in the Internet). A forwarding filter is installed, which causes traffic arriving at the gateway with destinations in the Network A to be looked up in the VRF that Tungsten Fabric created. The router advertises a default route via the VRF to the Tungsten Fabric controller.

Network A is configured to be a floating IP address pool in Tungsten Fabric, and when such an address is assigned to an existing VM interface, an additional VRF (e.g. for Network A) is created in the vRouter for the VM, and the interface is connected to the new, public VRF, in addition to being connected to the original VRF (green or red in Figure 6). VRFs for floating IP addresses perform 1:1 NAT between the floating IP address and the IP address configured on the VM. The VM is unaware of this additional connection and continues to send and receive traffic using the address for its original virtual network that it received via DHCP. The vRouter advertises a route to the floating IP address to the controller, and this route is sent to the gateway via BGP and it is installed in the public VRF (e.g. VRF A). The Tungsten Fabric controller sends the vRouter a default route via the VRF on the physical router and this is installed in vRouter's public VRF.

The result of these actions is that the public VRFs on vRouters contain a route to a floating IP address via a local interface of a VM, and a default route via a VRF on the router. The VRFs on the gateway have a default route (implemented using filter-based forwarding) via the inet.0 route table, and have host routes to each allocated floating IP address. The inet.0 route table has routes to each floating IP network via the corresponding VRF.

Multiple separate public subnets can be used as separate floating IP address pools with their own VRFs when tenants own their own public IP address ranges (as shown in the diagram), and conversely, one floating IP address pool can be shared among multiple tenants (not shown).

In cases where a non-Juniper device is used, or Tungsten Fabric is not permitted to make configuration changes on the gateway, a BGP session, public network prefix and static routes can be set up on the gateway manually, or by a configuration tool. This method is used when the router is combining a provider edge (PE) router role for enterprise VPNs with a datacenter gateway role. Generally, in this case, the VRFs will created by a VPN management system. A virtual network in the Tungsten Fabric cluster will be connected into an enterprise VPN when a matching route target is configured in the virtual network, and routes are exchanged between the controller and the gateway/PE.

Source NAT

Tungsten Fabric enables networks to be connected via a Source-based NAT service which allows multiple VMs or containers to share the same external IP address. Source NAT is implemented as a distributed service in each vRouter. The next hop for traffic being sent from a VM to the Internet will be the SNAT service in the same vRouter, and it will forward to the gateway of the underlay network without encapsulation, with source address modified to that of the vRouter host and setting source port to be specific to the sending VM. The vRouter uses the destination port in returning packets to map back to the originating VM.

This option is useful for providing internet access for workloads where the actual IP address of the source does not need to be known by the destination (which is usually the case).

Routing in Underlay

Tungsten Fabric allows networks to be created that use the underlay for connectivity. In the case that the underlay is a routed IP fabric, the Tungsten Fabric controller is configured to exchange routes with the underlay switches. This allows virtual workloads to connect to any destination reachable from the underlay network and provides a much simpler way than a physical gateway to connect virtual workloads to external networks. Care must be taken that overlapping IP address are not connected into the fabric, so this feature is more useful for enterprises connecting cloud to legacy resources rather than multi-tenant service providers.

Note that the traffic flowing to and from the underlay network is subject to network and security policy enforcement just as it is for traffic between workloads using virtual networks.

Copyright © 2018 Tungsten Fabric a Series of LF Projects, LLC For web site terms of use, trademark policy and general project policies please see https://lfprojects.org