VNET - Architecture

Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:40 PM PDT

Contents

Architecture

The VNet architecture enables to create virtual networks in Azure that are completely integrated with customers' existing networks: While enterprise customers wish to expand their existing corporate infrastructure (e.g., compute clusters, management servers, application server farms, disaster recovery centers, test labs) into Azure, simply running their own tenants in Azure fails to realize this key scenario.

Limitations

A few critical limitations of the existing Azure service model are:

1. VMs of a tenant are given Azure addresses and hence subject to address collision with the customers' other machines present in their on-premise networks;

2. addresses given to Azure VMs vary upon migration or re-provisioning, making it impossible for customers to describe and enforce their own networking policies (e.g., access control) using the VMs' addresses;

3. different tenants of the same customer, even located within the same Azure cluster, can communicate only indirectly through VIPs, suffering from poor scalability, performance, and even functional limitations (e.g., customer VMs cannot run their own IPsec because IPsec cannot traverse NAT). VNet addresses all these issues and allows customers to build their own networks in Azure, supporting the familiar network management models and abstractions for enterprise customers: IP subnets, VLANs, network ACLs and firewall rules. Conceptually, VNet enables the service model shown in following figure.

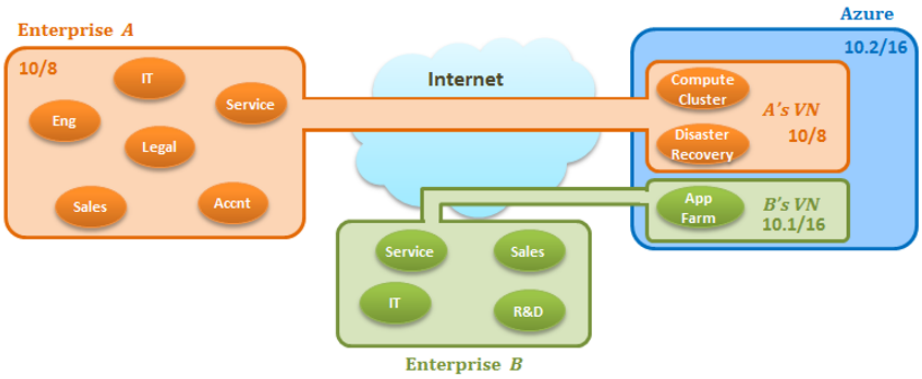

Figure 1. VNet allows customers to build their own virtual networks in Azure. In this example, two customers (Enterprise *A* and *B*), sharing the same Azure datacenter, build their own virtual networks (*A*'s VN and *B*'s VN). A virtual network (VN) can contain any number of virtual IP subnets (represented as small blobs in the customers'

VNs). A virtual subnet can contain any number of VMs located anywhere in Azure and can grow or shrink per customer's needs. All VMs in a virtual subnet share the same address prefix, and a customer can choose its own IP address space for its own subnet. A customer can describe and enforce access-control policies to the inter-subnet communication. Inter-subnet communication within a VN can be supported via DIPs, rather than only through VIPs.

- Specifically, VNet enables:
  - **BYOAS (Bring Your Own Address Space)**: VNet allows an Azure enterprise customer to extend its enterprise network into the Azure cloud, leveraging the Sydney Gateway for secure connectivity between customer and Azure sites. Azure customers can use their own address space in Azure, and apply their own management policies to manipulate resources in the cloud and enterprise network. VNet allows customers to control the assignment of IP addresses to their Azure VMs.
  - **BYO Networking Policy**: Once BYOAS is enabled, enterprise customers can build a new corporate site in Azure, as extension to their corporate networks. Customers then would want to apply their own corporate networking policies (access-control policies, most importantly) to the new site built in Azure. In the overwhelming majority of cases, these policies are tied to address space (subnets, VLANs, network ACLs, firewall rules). Accordingly, VNet enables further extensions of the "BYO" paradigm for the customer – for example to DNS and Active Directory.

- **Simpler, More Scalable Tenant Modification**: To ensure tenant isolation, Azure today uses NetFilter and host firewall. Unfortunately, because VMs' IP addresses within a tenant are not contiguous, the FC creates a large number of NetFilter rules written with individual VMs' IP addresses. The consequence of this design is inflexibility and poor scalability. Whenever a VM is relocated for maintenance or upgrade, or whenever a tenant grows or shrinks, the FC has to modify the NetFilter and firewall rules of not only the relocated VM (or the new VM), but all VMs in the tenant. Ideally, the FC should be able to complete a VM-relocation or a tenant-resizing task by modifying the configuration of the moving VMs or new VMs, but *not any other VMs'*. VNet ensures this core benefit by allowing VMs to retain their own IP addresses irrespective of their locations, even across data centers. As tenant sizes grow and inter-tenant communication arises, meeting this requirement will become paramount.

- **CVMs (Cheap VMs)**: VNet (with the "encap" option, described below, which we intend to implement in a future release) provides an architecturally sound, future-proof solution for the CVM scenario: allowing the Fabric to assign any IP addresses to any tenants in a completely location-independent fashion. With VNet, the Fabric can even assign the same addresses to different tenants and significantly reduce address consumption. Further, VNet allows the Fabric to populate any number of IP endpoints (VMs) per machine, rack, or VLAN. Hence, regardless of how many CVMs are created in a single node, VNet completely eliminates any network reconfiguration (address reassignment) to support that growth.

VNet creates one or more *virtual subnets* in each customer's virtual network (VN). A virtual subnet is a protected pool of physical or virtual machines (VMs) that share the same IP-prefix, with the members of the pool located anywhere in Azure data centers. The mechanism VNet employs to build virtual subnets is *ID-location separation*---VNet treats the IP addresses (*IDs*) of VMs as flat, customer-specific names that have no topological significance pertaining to the shared underlying physical networks. While VMs generate packets destined to the IP addresses of other VMs, network devices in the underlying physical network do not understand those IP addresses. Hence, to deliver packet correctly in the network, VNet additionally assigns a *locator* to each VM---locators are hierarchically assigned (hence, location-specific) IP addresses that network devices understand. When receiving packets from VMs, VNet then converts the VM addresses in the packets into corresponding locators at the network ingress point, and vice versa at the egress point. The specific conversion mechanism VNet employs is either encapsulation/decapsulation or address rewriting.

This design hides the VMs' identifiers from the underlying network devices (switches and routers) and allows customers general control over the choice of identifiers assigned to their VMs. To disambiguate customers which might share overlapping of the same address spaces, VNet introduces the notion of customer ID, used only in the control-plane, making data-plane backwards compatible with existing networking devices and protocols. Hence VNet is compatible with any physical layer-2 or layer-3 interconnection between host machines.

In the control plane, each VM's information (address, location, and customer ID) is determined at provisioning time by the Fabric Controller, and then replicated at Directory Servers. An Azure node hosting a source VM learns the locator for a given destination VM's identifier (IP address) by querying a Directory Server. The Directory Servers knows which machines belong to which customer and, hence, to which virtual subnet. Based on this knowledge, the Directory Servers enforce access-control policies dictating which VMs are allowed to communicate one another, enabling reachability isolation at customer, subnet and VM levels.

Internal Reference

Recording : VNET Service Endpoints

How good have you found this content?