Two decades ago Virtualization was the most fashionable keyword in IT as there was a revolution in the way we build servers. It was mainly about the adoption of Virtual machine instead of dedicated physical server in building applications   
Back in that time Virtual Machine was a clear winner (also valid today) when it comes to scaling, portability, capacity management, cost, ..,etc and you can find tones of comparison between the two approaches   
  
If Virtualization was the keyword that sums it up, the keywords of today are Cloud , SDN and Containers

Today, the heavily discussed comparison would be between VM and Containers promising a new way to build and scale applications.  
While many small organizations are thinking of containers as something too early to adopt, the simple fact that Google stated that “From Gmail to YouTube to Search, everything at Google runs in containers, we run two billion container a week” might give you a clue where the industry is heading

But what is Container and how is it comparable to VM?  
  
From technical prospect container is rooted from the Cgroups concept in Linux but the name might be inspired by the Actual metal cargo shipping containers that you see on ships

As both share the ideas of isolating contents , carrier independence , easy movement , …,etc   
  
Container is a logical packaging mechanism, you can think of it as Lightweight OS-level virtualization that run an application and its dependencies without the need to replicate an entire operating system as shown in the diagram. By doing so Application would be confined in a lightweight package that could be developed/tested individually then implemented and scaled much faster than the tradition VM as Developer just need to build/configure this light peace of software. currently most of the application has been containerized and publicly available. And finally there is no need to manage/support one OS per application

A screenshot of a cell phone

Description automatically generated

Many developers would call the Container Runtime “the Hypervisor of Containers” Although this is not a technically correct term, but it may be useful only to visualize the hierarchy.  
  
As in the VM technologies, the most common Hypervisors are VMware ESX/ESXi and KVM. In the Container technologies Rkt and Docker are the most known with Docker being the most widely deployed one.   
Before we give an overview of Docker let’s see some useful numbers in comparing VM vs Container

Juniper VSRX vs CSRX   
  
Currently most of common applications such as Redis, Ngnix, Mongo , MySQL, WordPress, Jenkins , Kibana and Perl have been containerized and offered publicly in https://hub.docker.com allowing developers to build/test application quickly

There are a lot of tests that compare performance and scaling for a giving application when it runs on Container vs VM. All of this comparison stress on the benefits of running your application in container.  
  
But what about network functions NFV such as firewalls , NAT , Routing , …,etc   
when it comes to VM based NFV, most of network vendors already implemented a Virtualized flavor of the hardware equipment that could be run on a hypervisor on standard x86 hardware  
VSRX is a Juniper SRX Series Services Gateway in a virtualized form factor built on the Junos operating system (Junos OS), and delivers networking and security features similar to   
those available for SRX  
  
when it comes to containerized based NFV, Not all vendors offer that   
CSRX is the industry first containerized firewall offering a Compact footprint, high-density firewall for virtualized and cloud environments   
  
In the comparison below between CSRX vs VSRX we can see the idea of CSRX being lightweight NFV but kindly be aware this comes at a cost, there are some features that won’t exist in containerized based NFV (VPN in the CSRX is one example) but is that a big drawbacks of the containerized based NFV?  
In fact not as deploying applications in a containerized style means micro-services technique where we split the application into different smaller services each part (container in that case) is doing a specific job. So there will be no need to have a NFV container that do everything in the same time.

|  |  |  |
| --- | --- | --- |
|  | **vSRX** | **cSRX** |
| Use Cases | Integrated routing, security, NAT, VPN, High Performance | L4-L7 Security, Low Footprint |
| Memory Requirement | 4GB Minimum | In MB’s |
| NAT | Yes | Yes |
| IPSec VPN | Yes | No |
| Boot-up Time | ~minutes | <1second |
| Image size | In GB’s | In MB’s |