**Virtual Networking**

**Software**

Vrnetlab, or VR Network Lab, is an open-source network emulator that runs virtual routers using KVM and Docker. Software developers and network engineers use vrnetlab, along with continuous-integration processes, for testing network provisioning changes in a virtual network. Researchers and engineers may also use the vrnetlab command line interface to create and modify network emulation labs in an interactive way. In this post, I review vrnetlab’s main features and show how to use it to create a simple network emulation scenario using open-source routers.

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

**cross-connect programs**

Vrnetlab has two programs for building connections between virtual routers: **vr-xcon** and **topo-machine**:

vr-xcon is a cross-connect program that adds point-to-point links between nodes. it is suitable for adding links one-by-one, or for building small topologies. recommend using vr-xcon if you want to be able to “disconnect” and “reconnect” individual links in the network.

topo-machine creates virtual network nodes and links between nodes, where the nodes and links are described in a json file.

**Virtual Connection**

Vrnetlab uses a cross-connect program named vr-xcon to define connections between node interfaces and to collect and transport data packets between those interfaces. All traffic between containers passes through the standard Docker0 management bridge, and the vr-xcon cross-connect program creates an overlay network of point-to-point TCP sessions on top of the management bridge. If the user stops the cross-connect script, the network connections between virtual nodes stop transporting packets.

The vr-xcon script runs in a Docker container and can take in a list of all point-to-point connections in the network and handle forwarding for all of them. If you set up your virtual network this way, then all connections will stop if you stop the script. You may also run many different instances of the script — each in its own Docker container — to create links one-by-one or in smaller groups. This way, you can “disconnect” and “reconnect” individual links by stopping and starting the container that runs the script for each link.

**VrNetLab and OpenWRT**

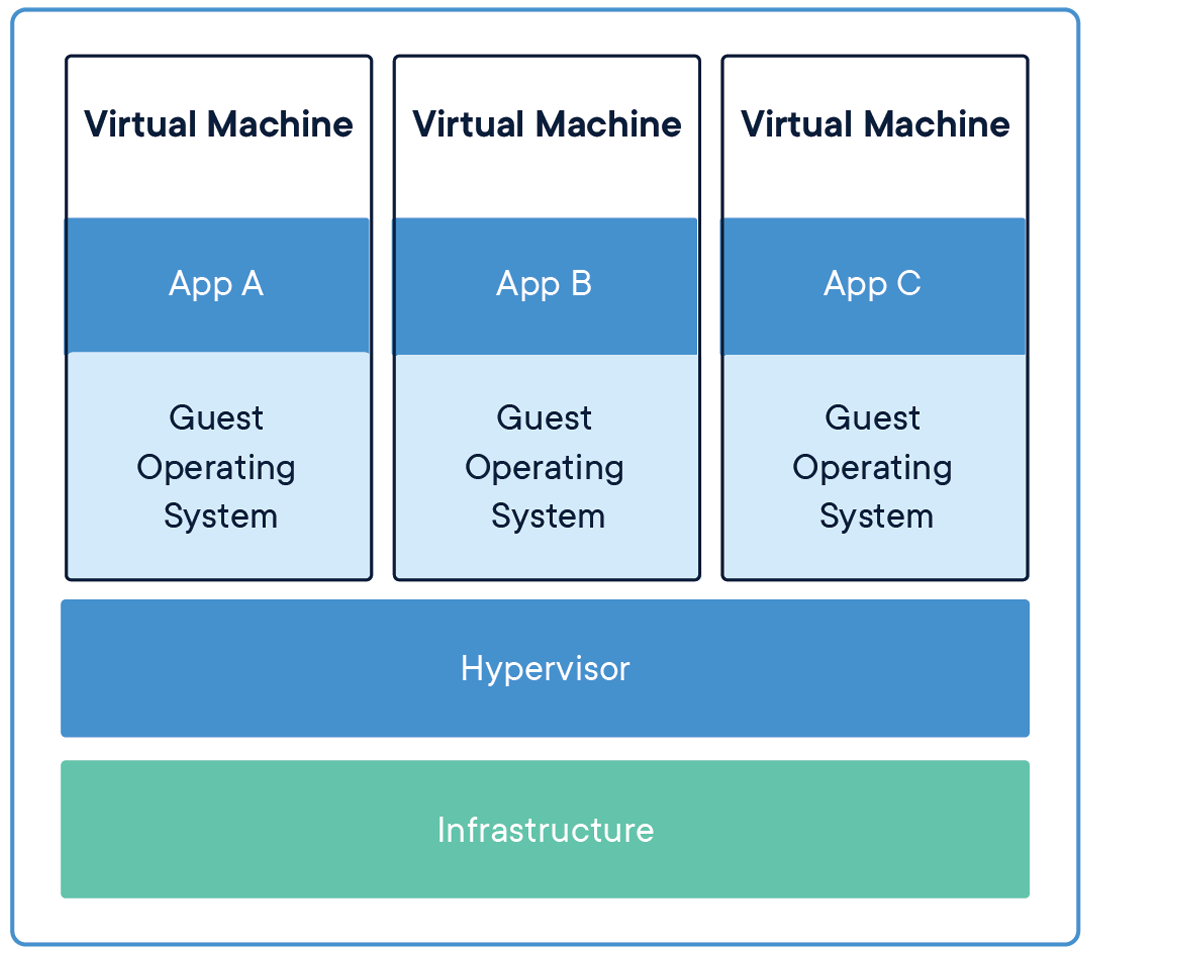
vrnetlab aims to make this process as simple and convenient as possible so that it may be used both by humans and automated systems to spin up virtual routers. In addition, there are scripts to help you generate topologies.

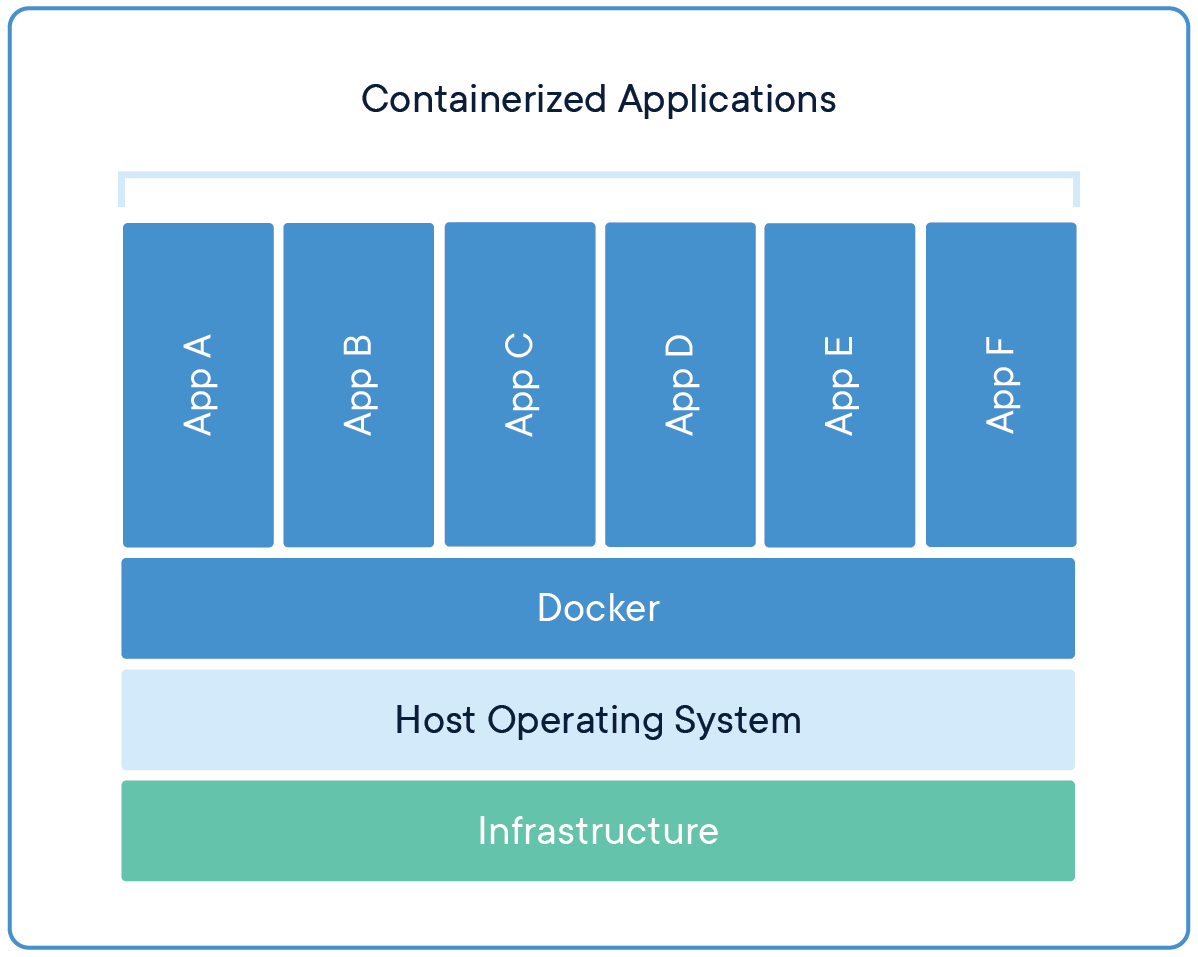
The virtual machines are packaged up in docker container. Since we need to start KVM the docker containers have to be run with --privileged which effectively defeats the security features of docker. Our use of docker is essentially reduced to being a packaging format but a rather good one at that. Also note that since we still rely on KVM the same amount of resources, if not sightly more, will be consumed by vrnetlab. A container is no thinner than a VM if the container contains a VM.

The assignment of a management IP address is handed over to docker, so you can use whatever docker IPAM plugin you want. Overall the network setup of the virtual routers are kind of shoe-horned into the world of docker networking.

**Docker Containers**

|  |  |
| --- | --- |
| **Docker** | **Virtual Machines** |





**Containers (Docker)**

This solution can run directly on an engineer’s machine, whether in a slim hypervisor for Mac or Windows or directly on the Linux kernel. If Docker is running solely on an engineer’s machine, resources are much more limited than the other options above, and therefore the fidelity isn’t as high as the other solutions. Where this option makes up for the lower fidelity, is that the level of effort to make changes to a simulated network is very low. These changes can be localized to an individual’s machine, so that it doesn’t affect any other team members. Highly configurable, but not as close to a production environment.

**Data persistence**

Vrnetlab VMs save changes made in the router configuration files or to data files on their disks. These changes will persist in the qemu disk images after the container is stopped. For example, when you want to work on something else, you may stop the containers in your network emulation scenario and turn off your server. Then, when you are ready to start work again, you can start your server and start all the containers associated with your network emulation scenario, including all vr-xcon containers. Your configuration changes will still exist on the network nodes.

**Links**

<https://ttl255.com/vrnetlab-run-virtual-routers-in-docker-containers/#few-words-on-doc>

<https://www.tecmint.com/sshpass-non-interactive-ssh-login-shell-script-ssh-password/>

<https://www.brianlinkletter.com/vrnetlab-emulate-networks-using-kvm-and-docker/>

<https://github.com/plajjan/vrnetlab#usagehttps://github.com/plajjan/vrnetlab#usage>

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<https://medium.com/tenable-techblog/simulating-enterprise-networks-in-development-using-the-docker-networking-stack-94bf547743c9>

<https://ttl255.com/vrnetlab-run-virtual-routers-in-docker-containers/https://ttl255.com/vrnetlab-run-virtual-routers-in-docker-containers/https://ttl255.com/vrnetlab-run-virtual-routers-in-docker-containers/>

<https://kubernetes.io/https://kubernetes.io/>

<https://networkop.co.uk/post/2019-01-k8s-vrnetlab/>