**26.3 Virtualization Options**

When it comes to virtual machines, there are many different vendors and open source projects to choose between. Careful consideration should be taken before deciding on a particular VM technology since compatibility between them is difficult at best, and considerable effort is required to change between them. Virtual machines are typically deployed from an image which contains the guest OS, application programs, and whatever customizations the administrator has configured for system management. These images can be deployed somewhat automatically, but they may require some human intervention to configure network and security settings.

**VMware**

One of the oldest and most widely deployed platforms is **VMware**, owned by Dell Technologies. VMware develops and maintains a vast array of virtualization technologies as well as providing technical assistance for their customers. Their vSphere and ESXi hypervisor platforms are nearly ubiquitous across industries where customers value stability and support.

**KVM**

**KVM**, or Kernel-based Virtual Machine is an open source hypervisor that is built into Linux. It should not be confused with a KVM (keyboard, video, mouse) switch, a hardware device that allows multiple computers to share a single monitor. As an open source program, it has the advantage of being non-proprietary and lacking licensing fees. Its deployment is quite widespread because of this and it is generally considered more scalable than VMware. KVM is used extensively in the SUSE Linux Enterprise distribution.

**Xen**

The **Xen** project is another open source hypervisor. It is considered a Type 1 or bare metal system because it runs directly on hardware without a host OS. This, along with its microkernel design, allows for very high performance from a lightweight system. Although Xen is most frequently used with Linux, it’s not tied to any one OS. Xen is currently the only bare metal hypervisor offered as open source software and is a Linux Foundation collaborative project.

**Hyper-V**

**Hyper-V** is a hardware virtualization product offered by Microsoft. It’s popular in environments already running Windows servers and desktop machines. Although it provides the appearance of a Type 2, or hosted hypervisor, running on top of Windows Server, it is actually installed beneath any Windows or other OS software. It also allows supported Linux distributions to be installed in a Windows environment.

**Virtual Box**

**VirtualBox** is a popular open source product from Oracle that is relatively easy to install and configure. It is a Type 2 hypervisor that runs on Windows, Macintosh, and Linux hosts. It is billed as a “general-purpose full virtualizer for x86 hardware”. Although it will run in a server environment it’s more suited to desktop users that need to run different operating systems occasionally.

**Docker**

As noted previously, Linux containers are quickly becoming the industry standard model for networked application development. The two most dominant forces in the containerization movement are Docker and Kubernetes

**Docker** is the container engine that allows programmers and system engineers to create containerized applications. These applications are stand-alone components that do not rely on any host OS to perform their functions. Its runtime, containerd, facilitates packaging application code and dependencies into a container that can run consistently across different infrastructure, thus allowing applications to operate regardless of the underlying infrastructure. As a top-level open source project of the Cloud Native Computing Foundation, containerd is constantly being improved to meet industry needs.

**Kubernetes**

**Kubernetes**, originally developed by Google, was converted to an open source project in 2014. Kubernetes can be described as a platform that “provides a container-centric management environment” for containers, microservices, and cloud infrastructure. It provides labels and annotations to help users keep track of resources. It “orchestrates” computing, network, and storage resources for container applications and workflows. It is also highly integrated with Docker so both programs can work together.

Kubernetes is organized into clusters, nodes (formerly minions), pods, and containers. The master node is in charge of maintaining the desired state in the cluster. It provides communication to the APIs, starts and stops processes, and balances workloads.

