4.6 TOOLS

Introduction

- In computing, **tools** are **software programs or platforms** that help developers, administrators, or users perform specific tasks more easily and efficiently.
- They are not the main application itself but **support the process of building, testing, deploying, or running software and systems**.
- Think of them like "helpers" that make work faster, more reliable, and automated.

The various virtualization tools are

1. VirtualBox,
2. KVM
3. VMware
4. Xen
5. Hyper-V
6. Docker

We can see some of the tools one by one

1. **Docker** - Containerization tool
2. **KVM** -Virtualization tool (used in Linux-based server/cloud)
3. **VirtualBox** - Virtualization tool (used in cross-platform, desktop)

4.6.1. Docker:

- **Docker is a containerization tool**.
- It allows you to package an application and all its dependencies (libraries, config files, runtime, etc.) into a single **container**.
- This container can run on **any system** without worrying about compatibility issues.

- In simple words: Docker makes sure "**it runs on my machine**" also means "**it runs everywhere**."

Use:

- Normally, software works on one computer but may fail on another due to missing libraries or different OS settings.
- Docker **solves this problem** by providing a consistent environment (container).
- It is lightweight (unlike Virtual Machines) because it shares the host OS kernel.

Features:

1. **Lightweight** – Uses fewer resources than Virtual Machines.
2. **Portable** – Works across Windows, Linux, Mac, and cloud.
3. **Fast Deployment** – Applications can be started in seconds.
4. **Scalable** – Easy to run multiple containers for load balancing.
5. **Version Control** – Can manage different versions of app environments.

Example Use Case

- Suppose you are building a web application using **Python + MySQL**.
- On your laptop, it runs fine, but on your friend's computer, it fails because of missing database drivers.
- With Docker, you create a container that includes:
    - Python runtime
    - Required libraries
    - MySQL database
- Now, anyone can run your app by just pulling and running your Docker image → no installation issues!

 Example: You can run a web app inside a Docker container with its own database, libraries, and runtime, without affecting your main system.

Common Docker Commands

```
# Pull an image (download from Docker Hub)
docker pull nginx

# Run a container
docker run -d -p 8080:80 nginx

# List running containers
docker ps

# Stop a container
docker stop <container_id>

# Remove a container
docker rm <container_id>

# Build a custom image from Dockerfile
docker build -t myapp .
```

Five type of Docker

a. Lazy Docker

Lazy Docker is a terminal UI for Docker and Docker Compose that makes managing containers a breeze. Instead of having to remember and type out long Docker commands, you get an interactive interface where everything is just a keypress away.

Some of the key features include:

- View container status, logs, and metrics at a glance
- Restart/remove/rebuild containers with a single keypress
- Monitor resource usage with ASCII graphs
- Attach to container shells easily
- Prune unused containers, images and volumes
- Full mouse support for clicking and scrolling

Getting started is super simple. On Mac you can install it with:

b. Sliplane

Sliplane is a hosting platform that makes deploying Docker containers super simple. While it's not exactly a "tool" in the traditional sense, it's become the go-to solution for many companies deploying Docker containers (full disclosure: I'm a co-founder!).

The main features that make it awesome for Docker deployments:

- Push-to-deploy from GitHub or Docker Hub
- Zero-downtime deployments
- Automatic health checks and notifications
- Built-in logging and monitoring
- Pay-per-server model (host unlimited containers on one server)

What I particularly like is that you don't need any DevOps knowledge - if your app works in a container locally, it'll work in production. Just connect your GitHub repo or point to a Docker Hub image, and you're good to go.

The pricing is also pretty straightforward - you pay for the server (starting at 7€/month) and can host as many containers as you want on it. Perfect for when you have multiple small projects or need development environments.

c. Dive

Dive is an incredible tool for exploring and analyzing Docker images layer by layer. It shows you exactly what files changed in each layer and helps identify ways to shrink your images by highlighting duplicated files and wasted space. The interactive UI lets you browse the complete filesystem tree while indicating what was added, modified or removed in each layer. Getting started is super simple - just run:

d. Orbstack

Orbstack is a Docker desktop alternative that I've been using for a while now. In my opinion the main selling point is that it's a native app for macOS and doesn't rely on Docker Machine to create VMs, meaning that its a lot faster and less resource intensive. It is still very new, so there are some rough edges, but I think it's worth a shot. I think its only supported on macOS at the moment, so that might be a problem for some of you 😉

e. Watchtower

I've written about Watchtower a few times already, and it's still one of my favorite tools. It's a simple tool that will watch your Docker images and automatically update them to the latest version. It's super easy to set up and just works. This is very useful for your homelab or so, not so much for production.

4.6.2. KVM (Kernel-based Virtual Machine)

- KVM is an **open-source virtualization technology** built into the Linux kernel.
- It allows Linux to act as a **hypervisor**, meaning it can run **multiple virtual machines (VMs)** on a single physical server.
- Each VM behaves like a **separate computer** with its own CPU, memory, storage, and network.

 KVM was introduced in **2006** and integrated into the Linux kernel in **2007**.

 Many enterprise virtualization solutions, like **Red Hat Virtualization**, rely on KVM. ie., Many open source virtualization technologies, including Red Hat's virtualization portfolio, depend on KVM as a component.

Hypervisor

A hypervisor is software that enables **virtualization** which lets you run an operating system (OS) within another OS. A hypervisor pools computing resources—like processing, memory, and storage—and reallocates them among VMs. A hypervisor can run multiple VMs at once, manage them, and support the creation of new ones. The physical hardware, when used as a hypervisor, is called the host, while the many VMs that use its resources are guests.

Hypervisors need operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more—to run VMs. KVM has all these components because it's part of the Linux kernel. Every VM is implemented as a regular Linux process, scheduled by the standard Linux scheduler, with dedicated virtual hardware like a network card, graphics adapter, CPU(s), memory, and disks.

Benefits of virtualization with KVM:

Virtualization makes it possible to quickly start and stop different operating system environments on a single piece of hardware, bringing a host of benefits to your IT ecosystem.

i) Flexibility

It is possible to run multiple VMs on one physical machine and allocate resources where they are needed. This way, you can maximize space, power consumption, and maintenance by hosting multiple VMs on a single piece of physical hardware.

ii) Speed

Because VM configurations are defined by software, VMs are quickly create, remove, clone, and migrated. VM are controlled remotely, and can automate the management of VMs.

iii) Compatibility

VMs can run software that was made for different OS versions. For example, you can launch an older operating system in a VM so you can keep existing software running on modern hardware.

iv) Stability and safety

VMs also have safety advantages because each VM runs separately. A VM can become unstable without affecting the host OS or other VMs on the same host. You can test a new configuration without risking a compromise to your entire system.

Features:

With KVM, a VM is a Linux process, scheduled and managed by the kernel. VMs running with KVM benefit from the performance features of Linux, and users can take advantage of the fine-grained control provided by the Linux scheduler. KVM also brings features related to security, storage, hardware support, and live migration.

i)Security boundaries with SELinux and sVirt

KVM uses a combination of Security-Enhanced Linux (SELinux) and sVirt for enhanced VM security and isolation. SELinux establishes security boundaries around VMs. sVirt extends SELinux's capabilities, allowing Mandatory Access Control (MAC) security to be applied to guest VMs and preventing manual labeling errors.

ii)Storage flexibility

KVM is able to use any storage supported by Linux, including some local disks and network-attached storage (NAS). KVM also supports shared file systems so VM images may be shared by multiple hosts.

iii)Support for multiple hardware architectures

KVM can run on a wide variety of hardware platforms. When used as part of Red Hat Enterprise Linux 9, KVM is supported with 64-bit AMD, Intel, and ARM architectures, as well as IBM z13 systems and later.

iv)Live migration

KVM supports live migration, which is the ability to move a running VM between physical hosts with no noticeable service interruption. The VM remains powered on, network connections remain active, and applications continue to run while the VM is relocated. KVM also saves a VM's current state so it can be stored and resumed later.

Manage VMs with KVM

When you're running multiple VMs, a virtualization management tool is helpful for keeping track of them. Some VM management tools run from the command line, others provide graphical user interfaces (GUIs), and others are designed to manage VMs across large enterprise environments. Here are a few common virtualization management solutions for KVM.

libvirt and virsh

The libvirt project provides an API for managing virtualization platforms. Within libvirt, virsh is a command-line utility for creating, starting, listing, and stopping VMs, as well as entering a virtualization shell.

Virtual Machine Manager

Virtual Machine Manager (known as VMM or virt-manager) provides a desktop interface for VMs, and is available for major Linux distributions.

Web consoles

VM administrators can choose to manage their VMs using web-based interfaces. For example, Cockpit offers a solution that lets users manage VMs from a web interface. Red Hat Enterprise Linux offers a web console plug-in for virtualization.

KubeVirt

KubeVirt is a solution for managing large numbers of VMs in a Kubernetes environment, where VMs can be managed alongside containerized applications. Kubevirt provides the foundation for Red Hat OpenShift® Virtualization.

Why migrate to a KVM-based virtualization platform?

One of the benefits of VMs is flexibility–and portability. That portability makes it possible to move a VM from one virtualization provider to another to take advantage of lower costs, expanded capabilities, more control, or other advantages.

Developers consider KVM a mature technology, and enterprises trust it to handle VMs efficiently at scale. A migration to a KVM-based platform, such as Red Hat's virtualization portfolio, is a way to preserve existing VM investments while choosing a solution with an open source foundation. Red Hat's technologies also make it possible to manage a unified environment for VMs, containers, and serverless apps.

Get an analyst perspective on VM migration

Why choose Red Hat for virtualization?

This video can't play due to privacy settings

To change your settings, select the "Cookie Preferences" link in the footer and opt in to "Advertising Cookies or try disabling adblockers."
Is OpenShift Virtualization right for your VMs? Video duration: 1:57

KVM is the hypervisor behind Red Hat's virtualization portfolio. Get the flexibility and benefits of an open source hypervisor like KVM with the assurance of Red Hat's enterprise support, security features, and stability.

Migrate your VMs and maintain your momentum

Red Hat's trusted products and partner ecosystem deliver comprehensive virtualization solutions. Migrate your virtual machines now to Red Hat® OpenShift® Virtualization, a modern app platform–based on KVM and KubeVirt–that integrates virtual and containerized workloads to provide flexibility without added complexity. Or, for a dedicated virtualization solution, explore Red Hat OpenShift Virtualization Engine, a streamlined, cost-effective offering to deploy, manage, and scale VMs exclusively. The included migration toolkit for virtualization provides the tools you need to start your migration in a few simple steps.

4.6.3. Virtual Box

Oracle Corporation developed a virtual box, and it is also known as VB. It acts like a hypervisor for X86 machines. Originally, it was created by Innotek GmbH, and they made it accessible to all in 2007. After that, it was bought by Sun Microsoft in 2008. Since then, it has been developed by Oracle, and people refer to it as Oracle VM Virtual Box. VirtualBox comes in a variety of flavors, depending on the operating system for which it is configured. VirtualBox Ubuntu is more common, however, VirtualBox for Windows is also popular. With the introduction of Android phones, VirtualBox for Android has emerged as the new face of virtual machines in smartphones.

Use of Virtual Box

In general, a Virtual Box is a software virtualization program that may be run as an application on any operating system. It's one of the numerous advantages of Virtual Box. It supports the installation of additional operating systems, known as Guest OS. It may then set up and administer free guest virtual machines, each with its own operating system and virtual environment. Virtual Box is supported by several operating systems, including Windows XP, Windows 7, Linux, Windows Vista, Mac OS X, Solaris, and Open Solaris. Windows, Linux, OS/2, BSD, Haiku, and other guest operating systems are supported in various versions and derivatives.

It can be used in following project

- **Software portability**
- **Application development**
- **System testing and debugging**
- **Network simulation**
- **General computing**

Advantages of Virtual Box

- **Isolation -** A virtual machine's isolated environment is suitable for testing software or running programmes that demand more resources than are accessible in other settings.
- **Virtualization-** VirtualBox allows users to run another OS on a single computer without purchasing a new device. It generates a virtual machine that functions just like a real computer, with its own processing cores, RAM, and hard disc space dedicated only to the virtual environment.
- **Cross-Platform Compatability-** VirtualBox can run Windows, Linux, Solaris, Open Solaris, and MacOS as its host operating system (OS). Users do not have to be concerned about compatibility difficulties while setting up virtual computers on numerous devices or platforms.

- **Easy Control Panel-** VirtualBox's simple control interface makes it easier to configure parameters like CPU cores and RAM. Users may begin working on their projects within a few moments of installing the software program on their PCs or laptops.
- **Multiple Modes-** Users have control over how they interact with their installations. Whether in full-screen mode, flawless window mode, scaled window mode, or 3D graphics acceleration. This allows users to customize their experience according to the kind of project they are working on.

DIsadvantages of Virtual Box

- VirtualBox, however, relies on the computer's hardware. Thus, the virtual machine will only be effective if the host is faster and more powerful. As a result, VirtualBox is dependent on its host computer.
- If the host computer has any defects and the OS only has one virtual machine, just that system will be affected; if there are several virtual machines operating on the same OS, all of them would be affected.
- Though these machines act like real machines, they are not genuine; hence, the host CPU must accept the request, resulting in delayed usability. So, when compared to real computers, these virtual machines are not as efficient.

Installation of Virtual Box

- Download it from its official website at Oracle VM Virtual Box site.
- Select your platform package.
- Click on the installer.exe.
- Follow all the Virtual Box installer.exe instruction.
- Start your Virtual Box and create your OS file.