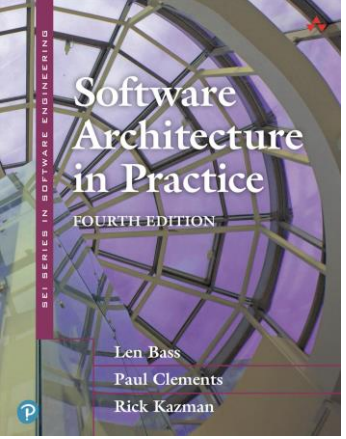
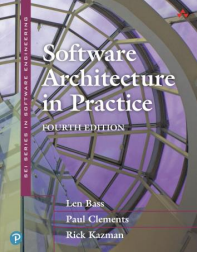


Chapter 16: Virtualization

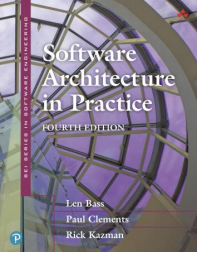
Virtual means never knowing where your next byte is coming from.

—Unknown



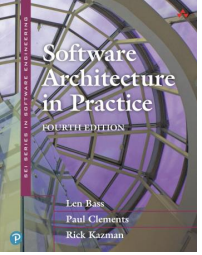
Chapter Outline

- Shared Resources
- Virtual Machines
- Containers and Pods
- Serverless Architecture
- Summary



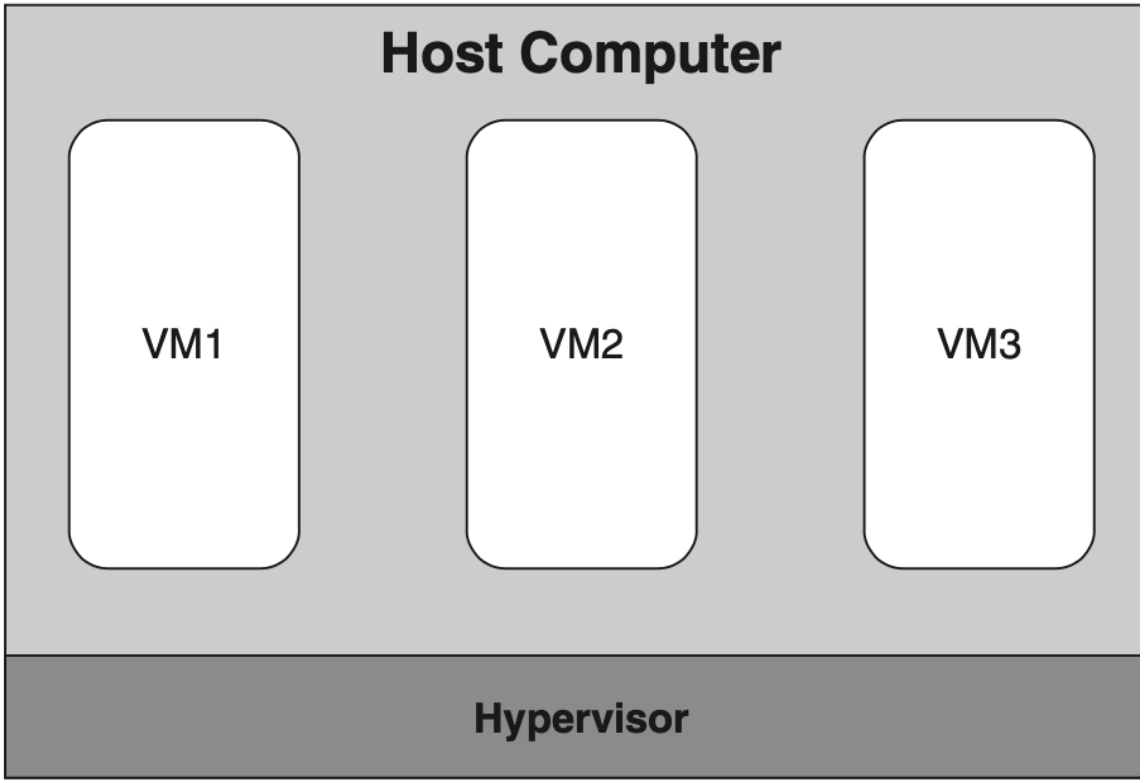
Shared Resources

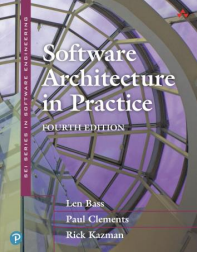
- The goal of virtual machines and containers is to isolate one application from another, while still sharing resources.
- Resource sharing can dramatically lower the costs of deploying a system.
- There are four resources that we typically care about sharing:
  - *Central processor units (CPUs)*
  - *Memory*
  - *Disk storage*
  - *Network connections*



Virtual Machines

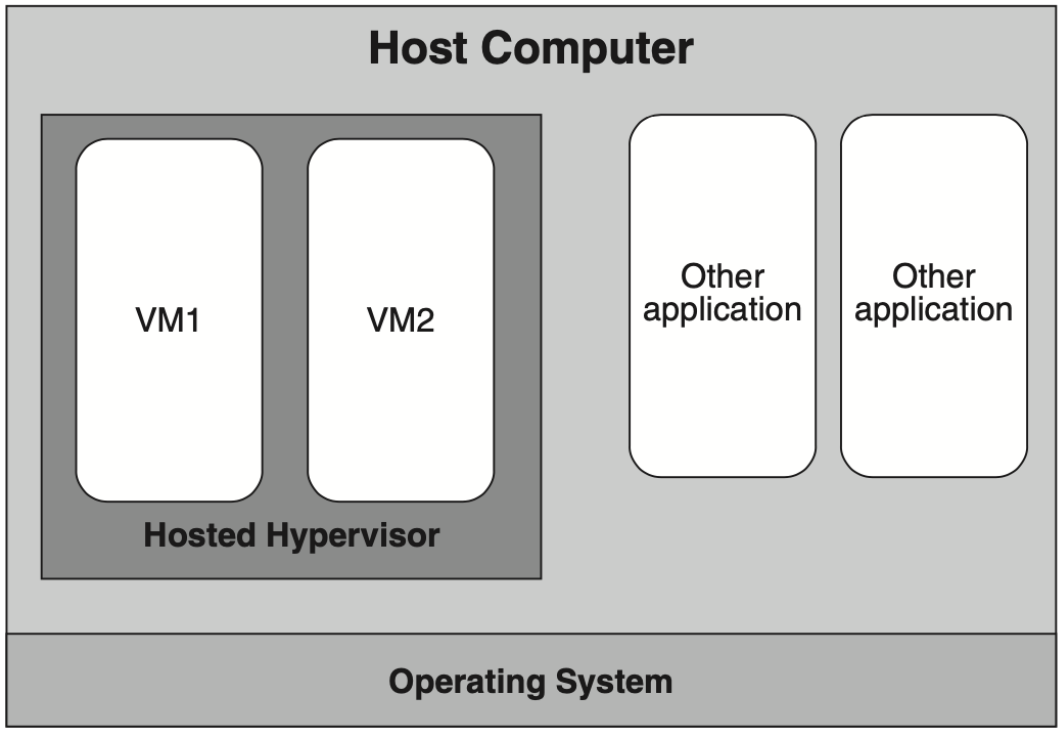
- The physical computer is called the “host computer”; the VMs are called “guest computers.” The *hypervisor* is an OS for the VMs. This hypervisor runs directly on the physical computer hardware and is often called a *bare-metal* or *Type 1* hypervisor
- The VMs that it hosts implement applications and services.

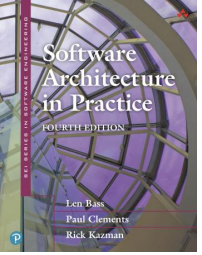




Virtual Machines

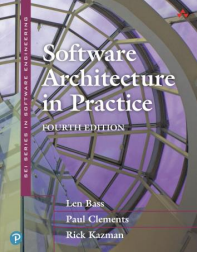
- Another type of hypervisor is a *hosted* or *Type 2* hypervisor which runs as a service on top of a host OS.
- The hypervisor in turn hosts one or more VMs. Hosted hypervisors are typically used on desktop or laptop computers.





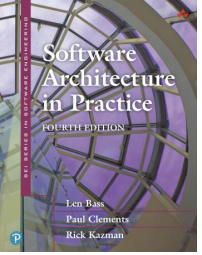
Virtual Machines

- A hypervisor performs two main functions:
  1. It manages the code running in each VM, and
  2. it manages the VMs themselves.
- Code that communicates outside the VM by accessing a virtualized disk or network interface is intercepted by the hypervisor and executed by the hypervisor on behalf of the VM.
- VMs must be managed, e.g. created and destroyed.
- From the perspective of the OS and services inside a VM, it appears as if the software is executing inside of a physical machine. The VM provides a CPU, memory, I/O devices, and a network connection.



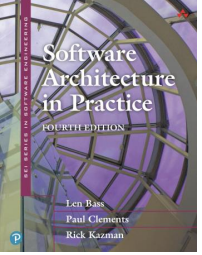
Virtual Machines

- The hypervisor is a complicated piece of software.
- One concern with VMs is the overhead introduced by the sharing and isolation needed for virtualization.
- Implications for the architect:
  - *Performance*. Virtualization incurs a performance cost.
  - *Separation of concerns*. Virtualization allows an architect to treat runtime resources as commodities, deferring provisioning and deployment decisions to another person or organization.



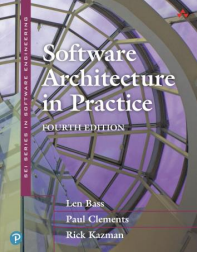
VM Images

- We call the contents of the disk storage from which we boot a VM a *VM image.*
- You can build a VM image on your development computer and then deploy it to the cloud.
- VM images are large, so transferring them over a network can be slow.
- It is customary to create images that contain only the operating system and other essential programs, and then add services to these images after the VM is booted, in a process called configuration.



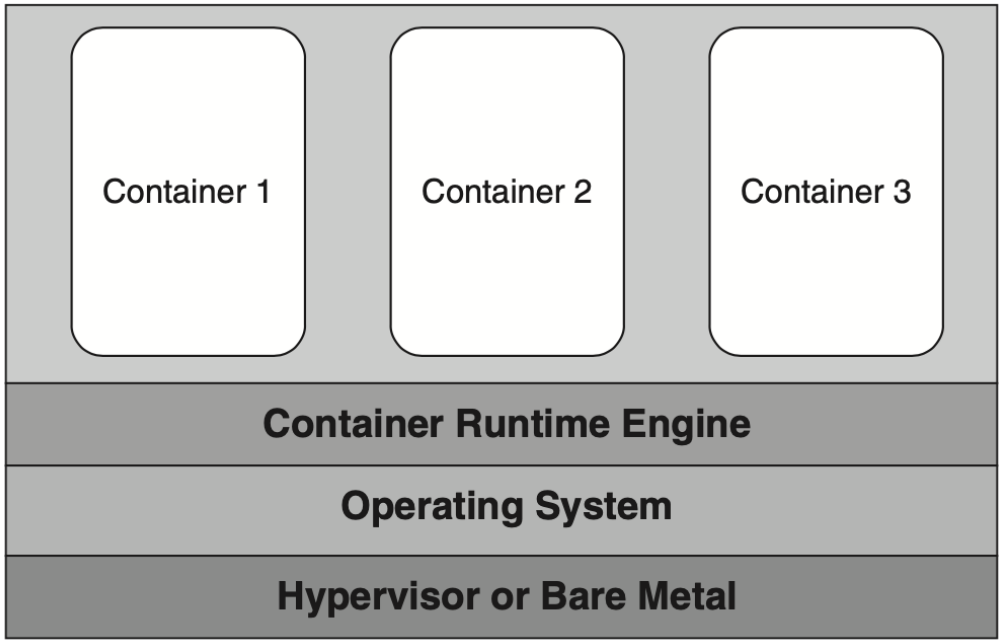
Containers

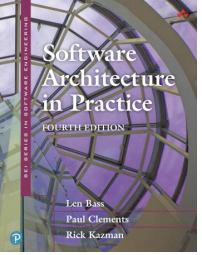
- VMs solve the problem of sharing resources and maintaining isolation. But VM images can be large, and transferring them is time-consuming.
- *Containers* maintain most of the advantages of virtualization while reducing the image transfer time and startup time.
- Like VMs and VM images, containers are packaged into executable container images for transfer.



Containers

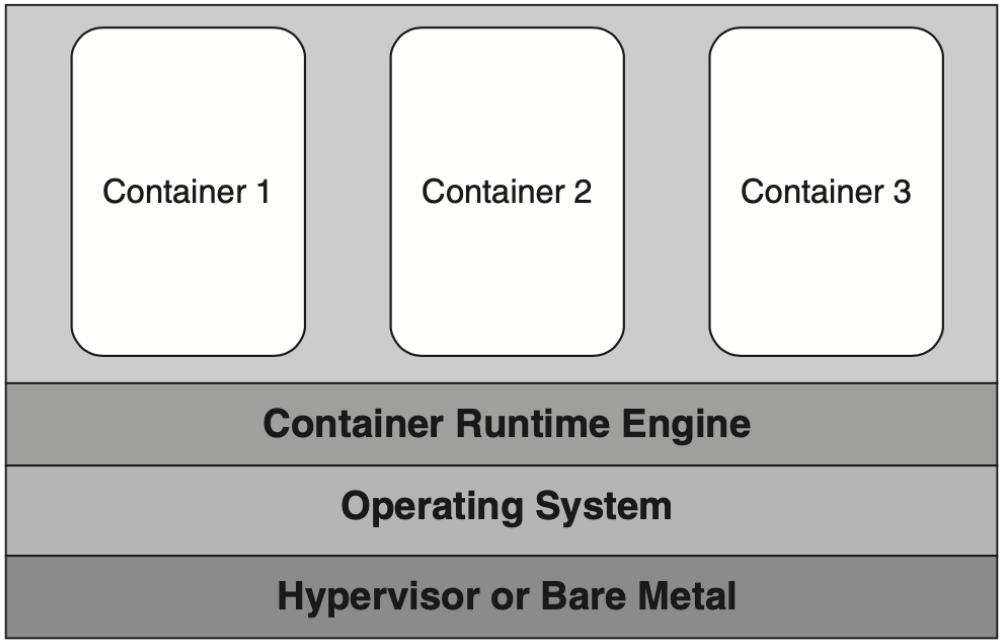
- Containers operate under the control of a *container runtime engine*, which runs on top of a fixed OS.
- The container runtime engine acts as a virtualized OS.
- The OS can be loaded either onto a bare-metal physical machine or a virtual machine.
- Containers are allocated by finding a container runtime engine that has sufficient unused resources to support an additional container.
- Containers impose some restrictions on port usage.

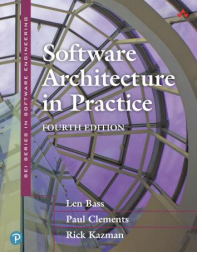




Containers

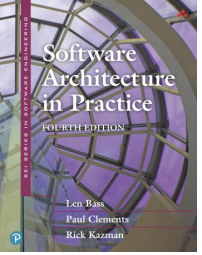
- This sharing of the OS is a source of performance improvement.
- As long as the target machine has a standard container runtime engine running on it, there is no need to transfer the OS as part of the container image.
- Another source of performance improvement is the use of “layers” in the container images.





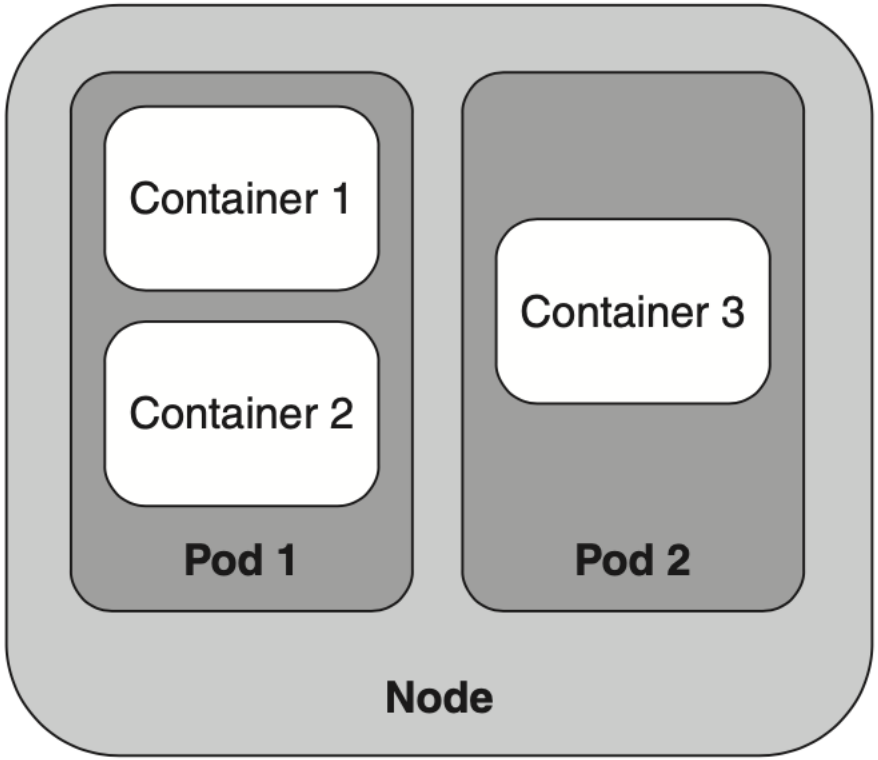
Containers versus VMs

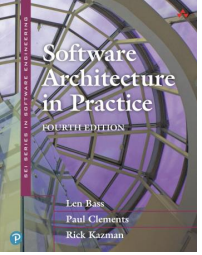
- What are the tradeoffs between delivering your service in a VM and delivering your service in a container?
- The software that you run on the VM includes an entire OS.
- This allows you to run multiple services in the same VM—a desirable outcome when the services are tightly coupled or share large data sets.
- Container instances share an OS.
- Containers generally run a single service so the size of the container image is small.
- Containers are portable.
- VMs persist beyond the termination of services running within them; containers do not.



Pods

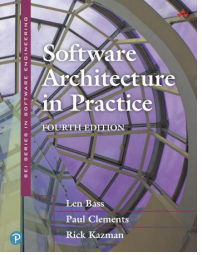
- Kubernetes is orchestration software for deploying, managing, and scaling containers.
- It has one more element in its hierarchy: Pods.
- A *Pod* is a group of related containers.
- In Kubernetes, nodes (hardware or VMs) contain Pods, and Pods contain containers.





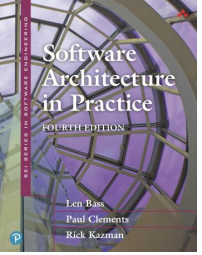
Serverless Architecture

- Serverless architectures are not, in fact, serverless.
- There are servers, which host container runtime engines.
- Containers are allocated dynamically with each service request. And they are cached, to improve performance (after initial load). These containers are typically stateless.



Summary

- Virtualization has been a boon for software and system architects, as it provides efficient, cost-effective allocation platforms for networked (typically web-based) services.
- Hardware virtualization allows for the creation of several virtual machines that share the same physical machine. It does this while enforcing isolation of the CPU, memory, disk storage, and network.
- Consequently, the resources of the physical machine can be shared among several VMs, while the number of physical machines that an organization must purchase or rent is minimized.



Summary

- A VM image is the set of bits that are loaded into a VM to enable its execution.
- Containers are a packaging mechanism that virtualizes the operating system. A container can be moved from one environment to another if a compatible container runtime engine is available.
- Placing several containers into a Pod means that they are all allocated together and any communication between the containers can be done quickly.
- Serverless architecture allows for containers to be rapidly instantiated and moves the responsibility for allocation and deallocation to the cloud provider.