Deployment and Operations for Software Engineers 2nd Ed

Chapter 2—Virtualization

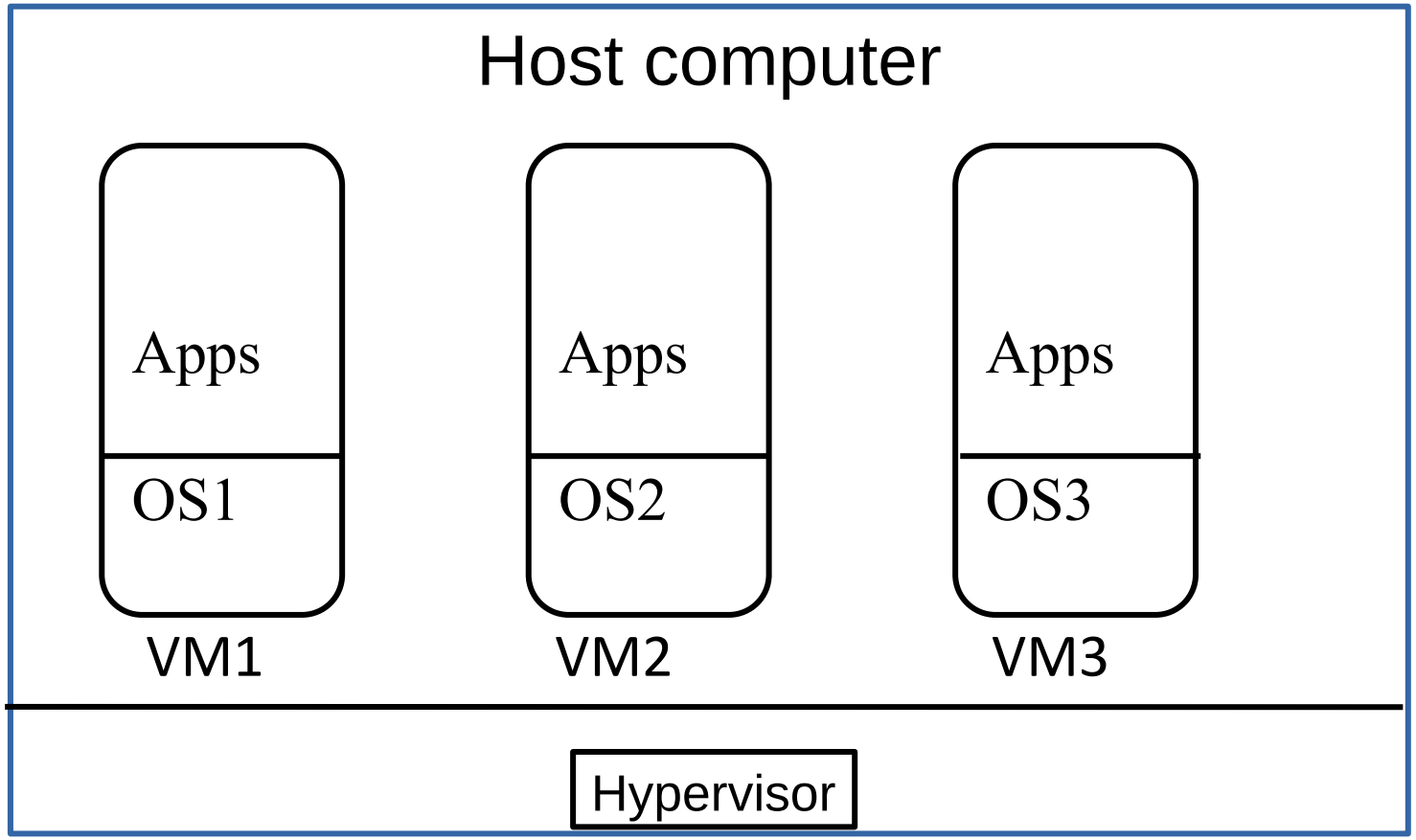
Outline

- **Virtual machines**
- Virtual machine images
- Containers
- Container repositories
- Serverless architecture

What is a Virtual Machine?

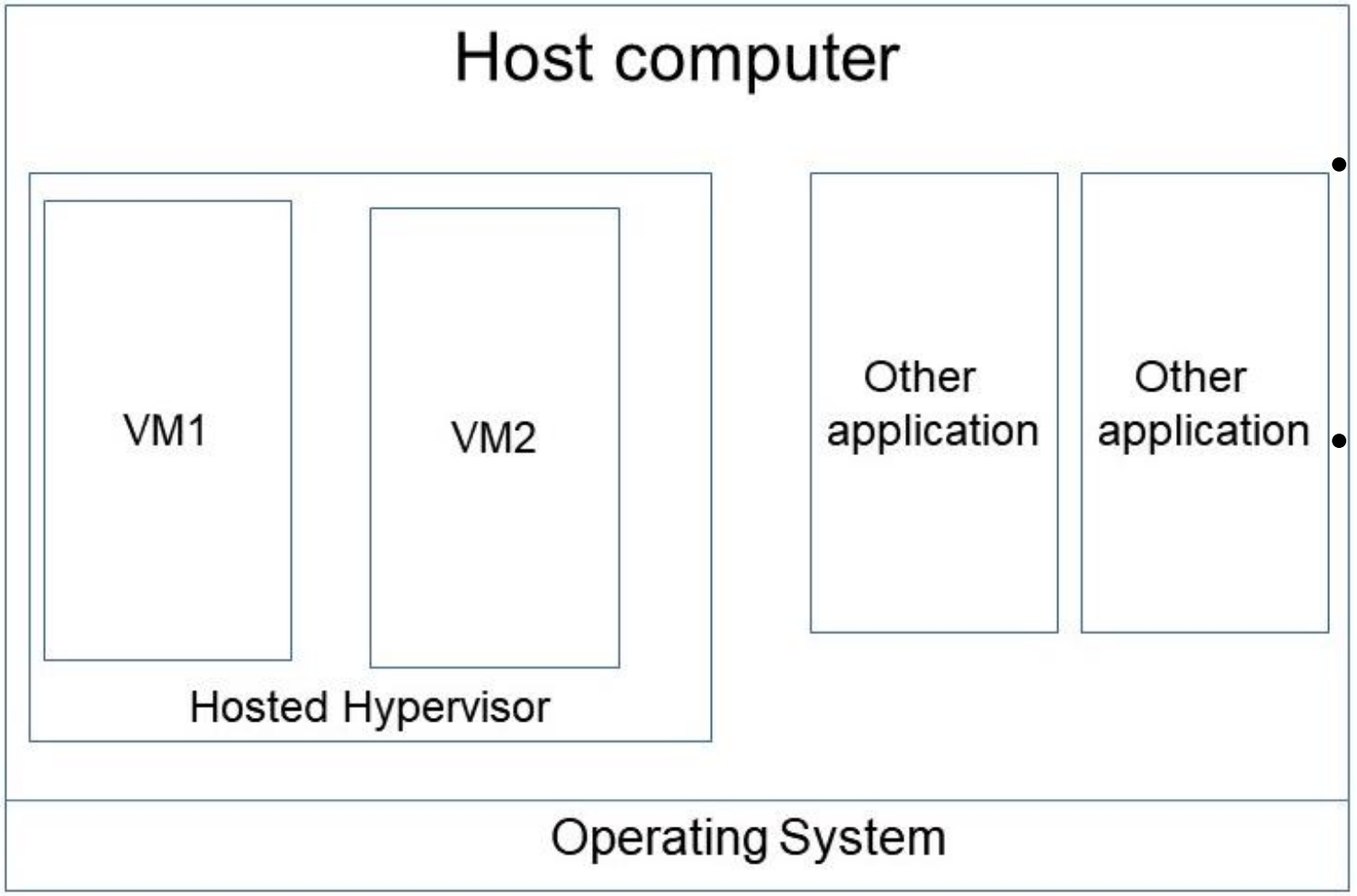
- A Virtual Machine (VM) is a software construct that exposes the resources of CPU, memory, disk, and network connection to loaded software making it appear to the loaded software that it is executing on a physical computer.
- A VM runs under the control of a specialized operating system called a hypervisor.

Type 1 Hypervisor



The hypervisor is the operating system for virtual machines
A type 1 hypervisor runs on bare metal.
C

Type 2 Hypervisor



- A type 2 hypervisor runs on top of your laptop's operating system.
- From the perspective of the laptop's OS, it is an application.

VM resources

- From the perspective of software executing inside a VM, it is the same as software executing on a bare metal computer.
- A VM has the same resources as a physical computer.
  - CPU
  - Memory
  - Disk
  - Network connection

Discussion questions

1. How are the actions of one VM kept separate from the actions of another VM on the same host?
   (hint: think one resource at a time)
2. How does a VM write to a disk?
3. How does a VM write to the network?
4. How is a VM created?
5. How are VMs booted?

Outline

- Virtual machines
- **Virtual machine images**
- Containers
- Container repositories
- Serverless architecture

VM Image

- A VM image is a set of bits on a disk (or from over a network) that contains all of the software, directories, and data from its created VM
- An image is created by
  - building a VM or physical computer with all of the desired software.
  - Writing the contents of memory to a disk file (or sending it over a network)
- A VM is created by reading an image.

Creating an image and using it to create another VM

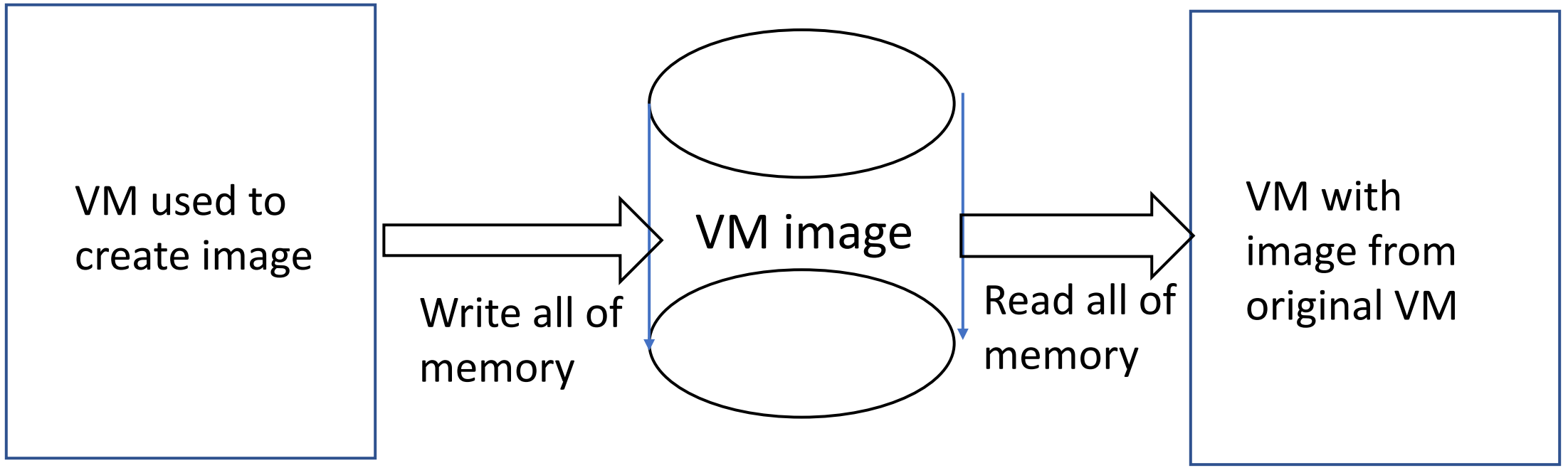


Image across network

- Suppose image is 8 Gb(yte).
- Reasonable network speed is 1 Gb(it)/sec (rated)
- 8 bits per byte
- Moving image, in theory, will take 64 seconds.
- In practice a 1Gb(it) network speed is .35 Gb(it)/sec
- Transferring image will take ~3 minutes.
- VM still must be booted to execute in new location.

Discussion questions

- Can a VM image created on AWS be moved to Google Cloud?

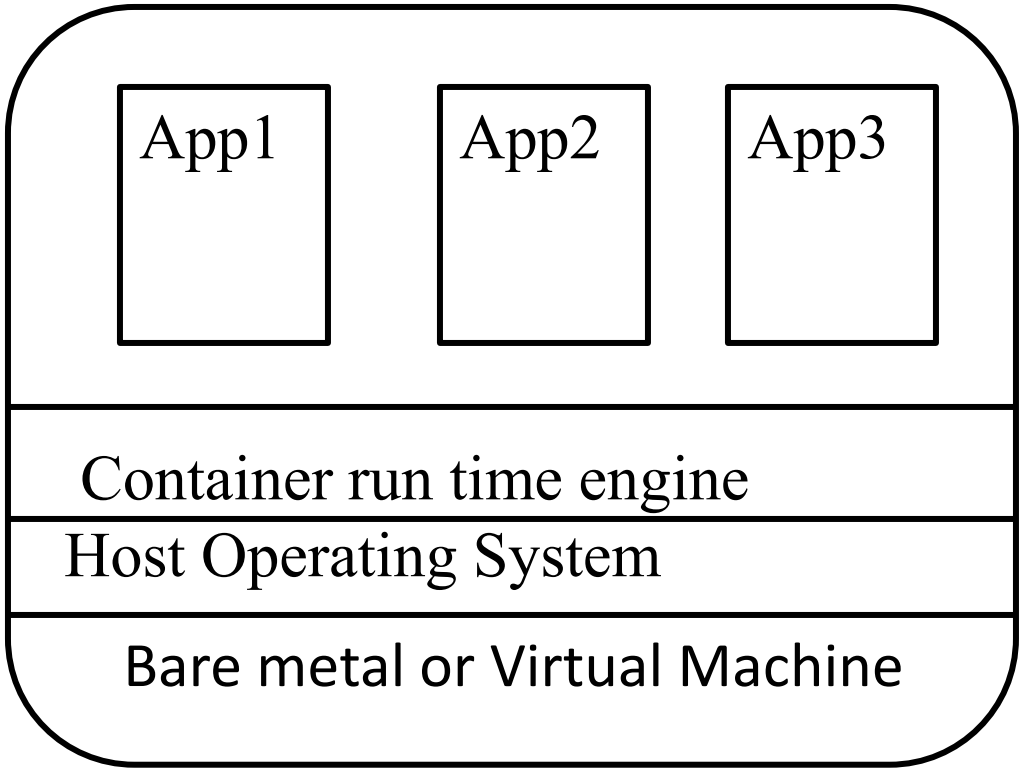
Outline

- Virtual machines
- Virtual machine images
- **Containers**
- Container repositories
- Serverless architecture

Container is lighter weight virtualization mechanism

- Containers are a mechanism to maintain the advantages of virtualization—isolation of its actions from other containers—while reducing image transfer time and startup time.
- A container is an executable image that packages a service and its dependent libraries. It runs under the control of a container engine.
- Containers are isolated from each other in terms of
  - address space
  - Disk usage
  - Processor usage
  - Network usage (with some exceptions).

Executing container



- Containers execute on top of a container run time engine
- This, in turn, executes on top of an operating system
- Means containers share a single operating system

Container layers

- Containers can be segmented into layers.
  - E.g. consider the LAMP stack – Linux, Apache, MySQL, PhP
  - Each item can be made a separate layer
- Suppose only the top layer has changed. Then when moving the container, only the top layer must be moved. E.g., a new version of PhP in our example.
- This makes moving a container much lighter weight than moving a VM. On the order of milliseconds rather than minutes.

Comparing VMs and containers

- VMs virtualize hardware
- Containers virtualize operating systems
- VMs take minutes to transfer over a network
- Containers can be transferred in milliseconds—depending on the mechanisms used
- VMs are monitored, started and stopped by the hypervisor
- Containers are monitored, started and stopped by the run time engine.

Discussion questions

1. How are the actions of one container kept separate from the actions of another container on the same host? On the same runtime engine?
2. How does a container write to a disk?
3. How does a VM container write to the network?
4. How is a container created?
5. How are container placed into execution?
6. How does the container management system know that only the top layer has been changed?

Outline

- Virtual machines
- Virtual machine images
- Containers
- **Container repositories**
- Serverless architecture

Container repositories

- Container images are typically stored in repositories
- Similar to version control systems
  - Accessible with permissions
  - Push/pull interface
  - Images can be tagged with version numbers

Docker Hub

- Docker Hub is a publicly available repository.
- Since it is open source, It is also used for private container repositories
- It is the most widely used container repository
- Private repositories can be access controlled

Integrating with development workflow

- Multiple team members may wish to share images
- Images can be in production, under development or under test
- Private container repository allows images to be stored and shared.
  - Any image can be “pulled” to any host
  - Tagging as “latest” allows updates to be propagated. Pull <image name>:latest gets the last image checked into repository with that name.

Discussion questions

1. What are container repositories other than Docker Hub? Why would you chose one over the other?

Outline

- Virtual machines
- Virtual machine images
- Containers
- Container repositories
- **Serverless architecture**

Serverless architecture

- A simple example using PhP.
- PhP is an interpreted language.
  - A file containing your program is fed to the interpreter to be executed.
- Cloud provider has pool of PhP servers.
  - You tell one of the pool the location of the file with your program.
  - Your program is executed immediately.

General definition

- A serverless architecture consists of
  - A pool of servers with container run time engines
  - A stateless container with no dependencies on your other code
- Since the container has no dependencies, it can be executed directly by run time engine.
- Container is deployed and activated when a message arrives for it.
- Loads in milliseconds depending on cloud provider's caching strategy.
- For AWS, only one request per instance.

More complicated example

- Suppose you wish to send an email welcome message to a new subscriber.
- Your container
  - Consists of code to prepare and end the email.
  - Parameters are email address and subscriber's name.
- Container is stateless. Necessary information comes from request parameters.
- Your full system sends request to the welcome message container
  - This triggers the cloud provider to load and activate it.
- Email message is sent.

Discussion questions

1. How does the cloud provider know that a container is intended to be serverless?
2. What are restrictions on serverless container other than it must be stateless?