Virtual Machines Containers (A brief overview)

ICS332

Operating Systems

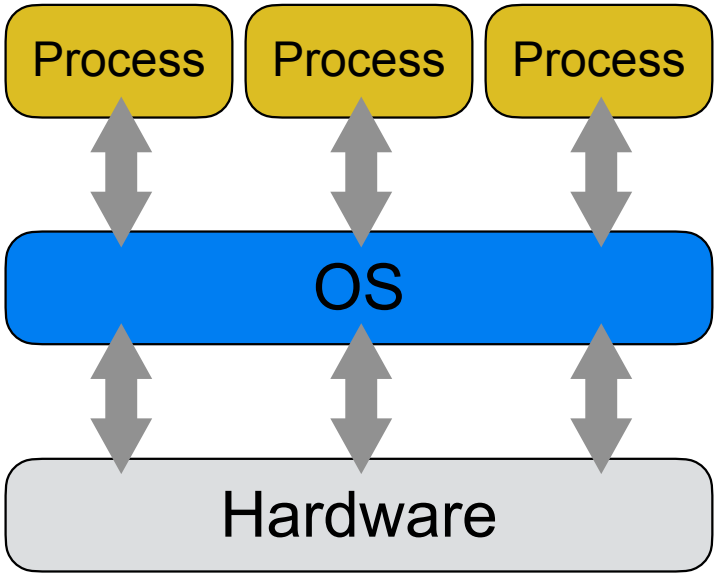
Henri Casanova (henric@hawaii.edu)

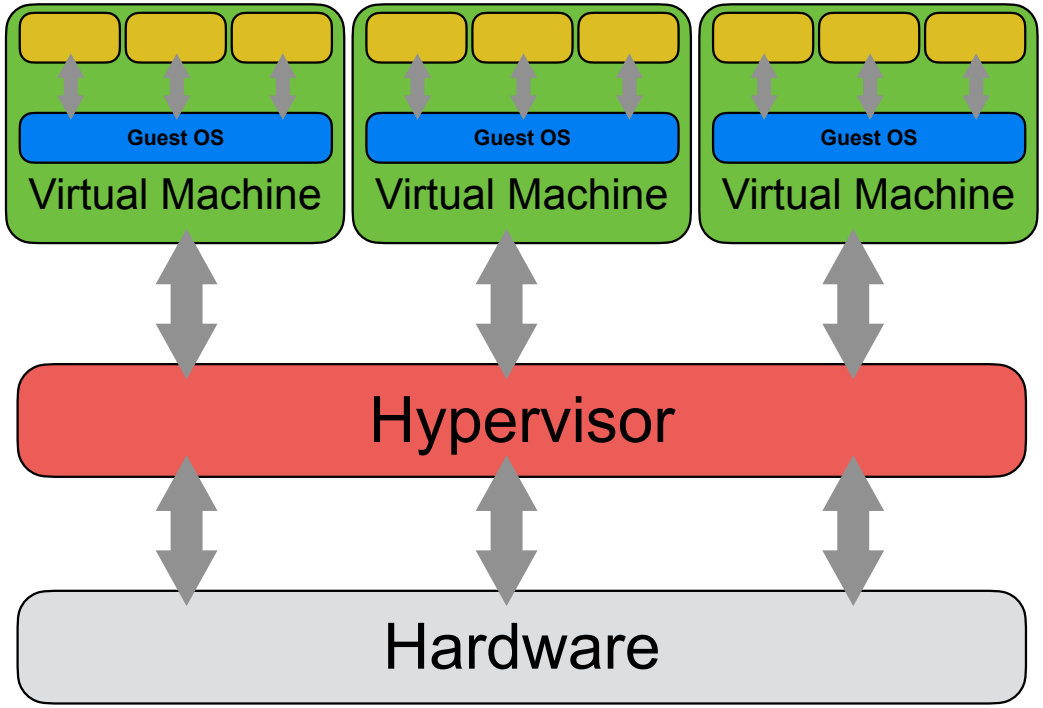
Objective

- Make sure you have a basic understanding of **Virtual Machines** (VMs)
- Make sure you have a basic understanding of **containers**
- They both have the same “run anywhere” goal: replicate the functionality and behavior of one system (the guest) on another system (the host)
  - A “system” is hardware and/or software
- And yet, there are different
  - Both useful in their own way
  - Often used together

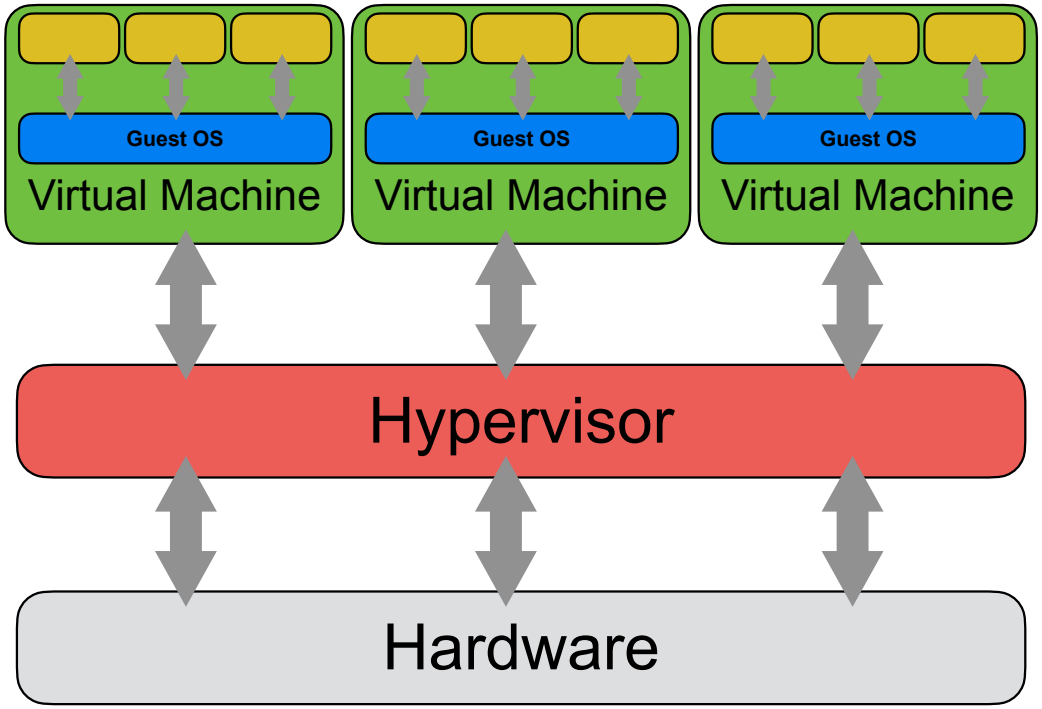
Virtual Machines (VMs)

- The software used to run guest VMs on a host is called a hypervisor or virtual machine monitor
  - It abstracts and allocates resources to VMs
- The hypervisor is to VMs what the OS is to processes:





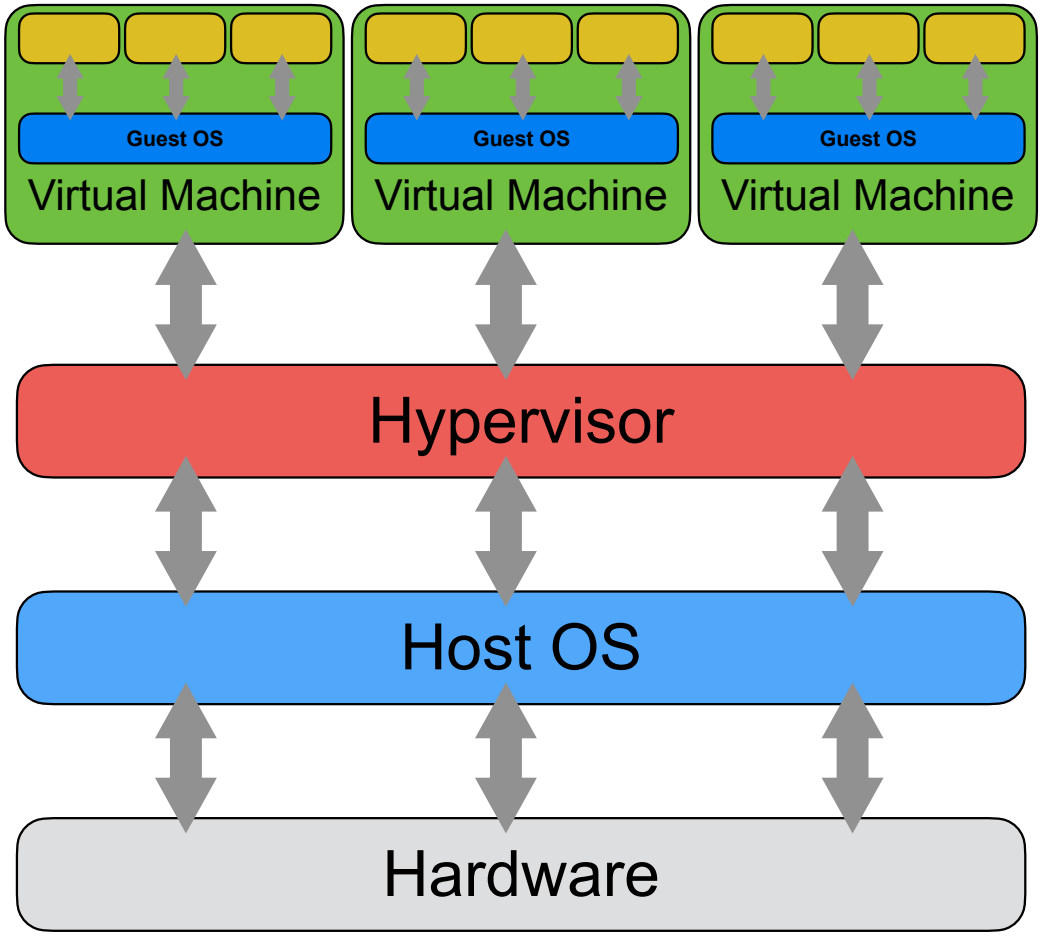
Hypervisor Type 1: Bare-Metal



This diagram showed the hypervisor running directly on the hardware

- Very efficient
- Used in enterprises, not on personal computers
- Examples: Hyper-V, Xen, VMWare ESXi

Hypervisor Type 2: Hosted



This diagram shows the hypervisor running on the host OS

- Less efficient, but easy to setup and convenient
- Used on personal computers
- Examples: VirtualBox, VMWare Workstation, Hyper-V, Parallels

Virtualization / Emulation

- Everything is “easily virtualized” when the guest is for the same computer architecture as the host
  - e.g., an x86 VM running on an x86 host
- If this is not the case, then the hypervisor must use emulation to “mimic the hardware”
  - e.g., using QEMU on my Mac, which emulates a full system and does automatic binary translation of machine instructions of the guest architecture to the machine instructions of the host architecture!
  - Completely transparent to the user, but much slower
  - You may have used emulators before (for game consoles?)
    - These are really simulators in software of the guest machine (they don’t do binary instruction translation)

Reasons to use VMs (1)

- **Isolation / Sandboxing**
  - Running untrusted code, having untrusted users
    - `rm -rf /` on the guest does not do anything harmful on the host
- **Resource allocation**
  - The hypervisor can partition hardware resources (CPU, RAM, etc.) among the VMs and limit each VMs resource allocation
    - With hardware support from the CPU (Intel VT-x, AMD-V, etc.)
  - This makes it possible to have better hardware resource utilization, e.g., in cloud platforms
  - A cloud can run 1,000 useful VMs on 200 physical hosts
    - Because not all VMs need the full power of a host
  - This avoids over provisioning the cloud with 1,000 physical hosts, which would leave most of them unused

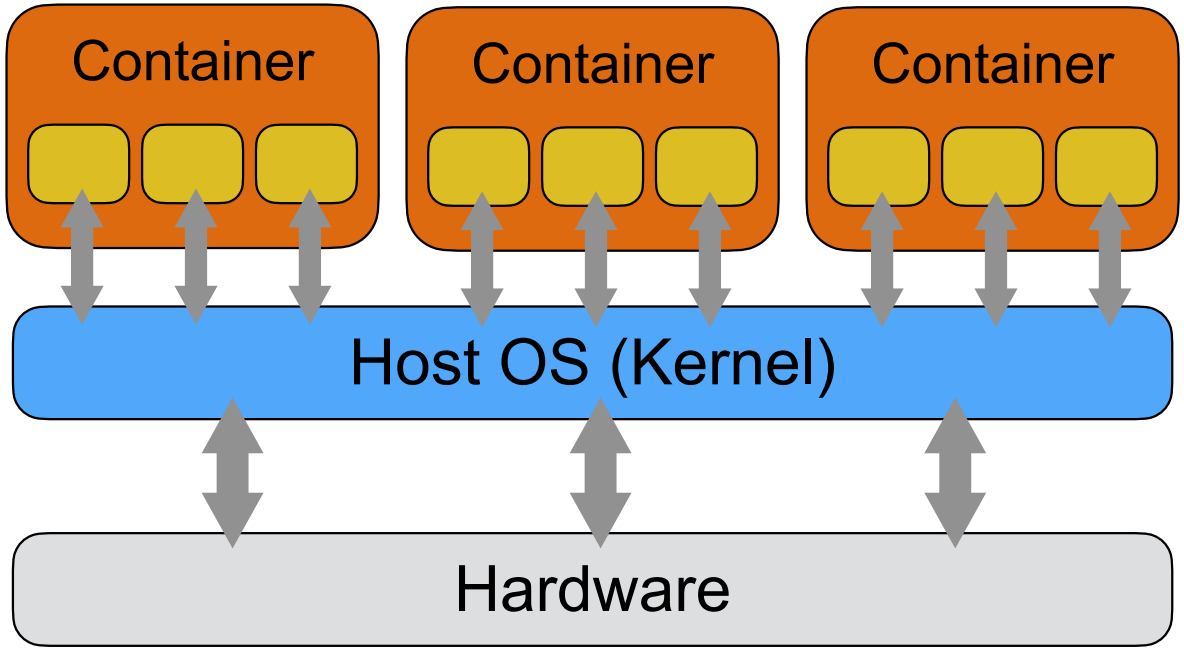
Reasons to use VMs (2)

- **Convenient to use**
  - Easy to suspend/save/restore/shutdown a VM without losing access to the host
- **Convenient to distribute software**
  - Easy to send somebody a VM image for them to run a specific system with all kinds of useful software pre-installed
    - Avoids the: “Oh, you want to use my software? First, you need to install a hundred dependencies…”
- **Cross-platform testing/development**
  - Makes it very easy to test and develop code on all kinds of system configurations
  - You can run multiple VMs with an emulated network to mimic a distributed system on a single host
  - Great for kernel experimentation and development

Containers

- Containers are useful for some of the same reasons, and at a very high level have the same goal: mimic a system on another system
  - Or the same system but with a bunch of useful software already installed!
- They are often said to be “lighter than VMs”
  - Faster to start / stop, less memory
  - Often pretty ephemeral / disposable
- The key difference: the container defines the OS to use but not the Kernel!
  - Instead, it uses the Kernel of the host’s OS
  - Therefore, *there is no need to boot a container*

Containers



Processes inside containers all use the same Kernel

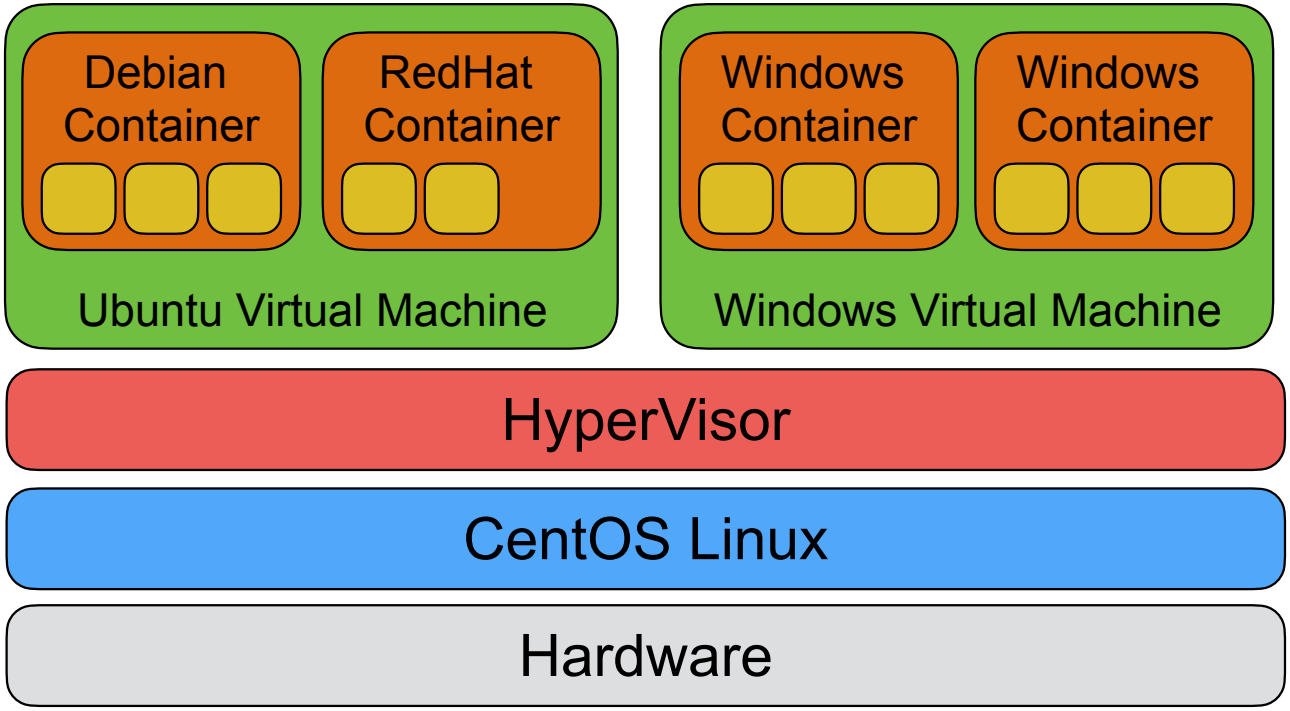
If the container is not compatible with the host (different OS, different architecture), then it transparently use emulation or a VM underneath!

Docker

- In this course many of you have used Docker
  - The first highly popular container system
- A Docker image is described in a so-called Dockerfile
  - Defines the CPU's architecture family
  - Defines the OS
    - Can be for Linux or Windows
  - Will run anywhere
    - But perhaps using emulation and/or a VM, which slows things down considerably (like for instance on my M1)
  - Can inherit from another Dockerfile
  - Specifies software installation, among other things
- Then Docker containers can be created for the image

Containers

- Containers allow to do some resource allocation and partitioning, but not as much as VMs
- They also provide less isolation than VMs
- Often the two are used in combination:

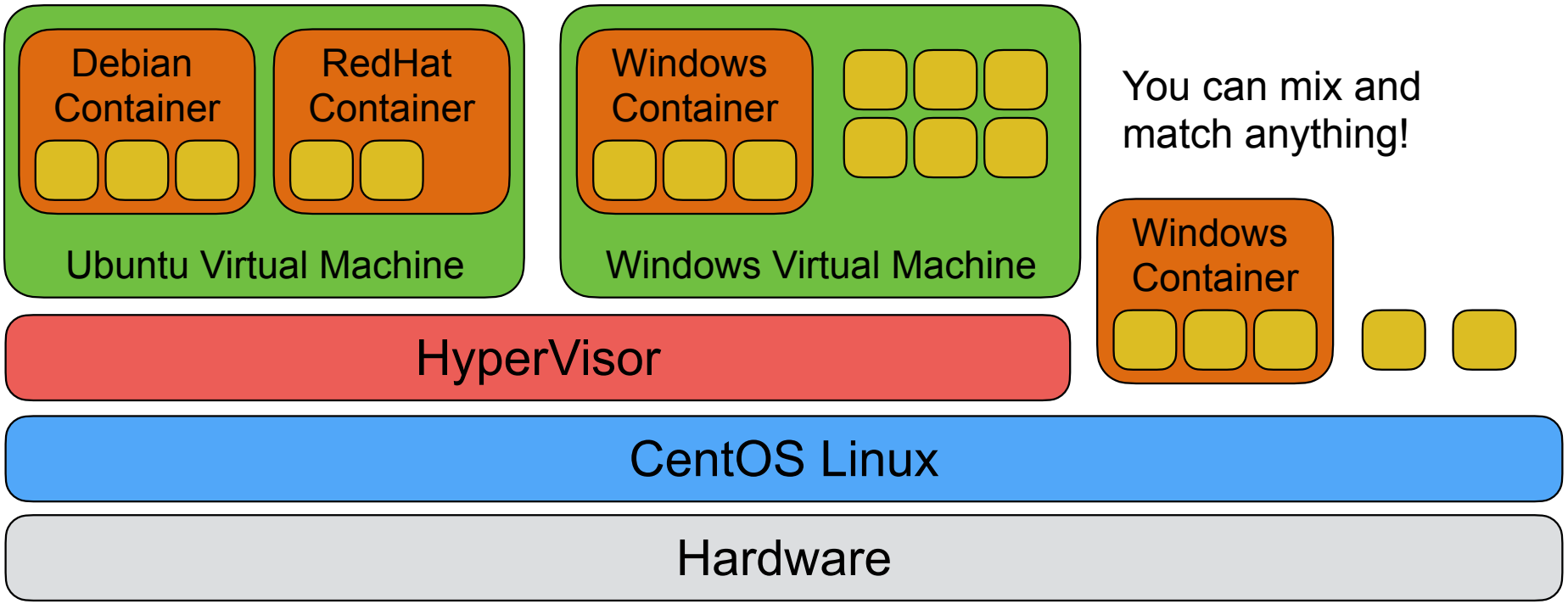


Virtual machines are used for 100% isolation and precise resource allocation

Containers run within each VM and come with all kinds of useful software

Containers

- Containers allow to do some resource allocation and partitioning, but not as much as VMs
- They also provide less isolation than VMs
- Often the two are used in combination:



Conclusion

- In almost all conceivable jobs you will have after graduation you will use VMs and containers
  - Most of you are probably doing it now anyway