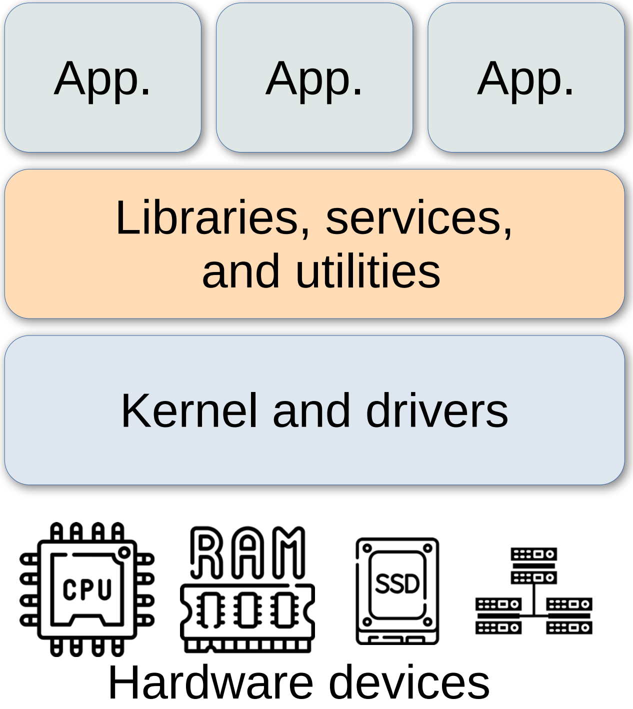
# Distributed Data Processing Environments

José Orlando Pereira

Departamento de Informática
Universidade do Minho

# Operating system stack

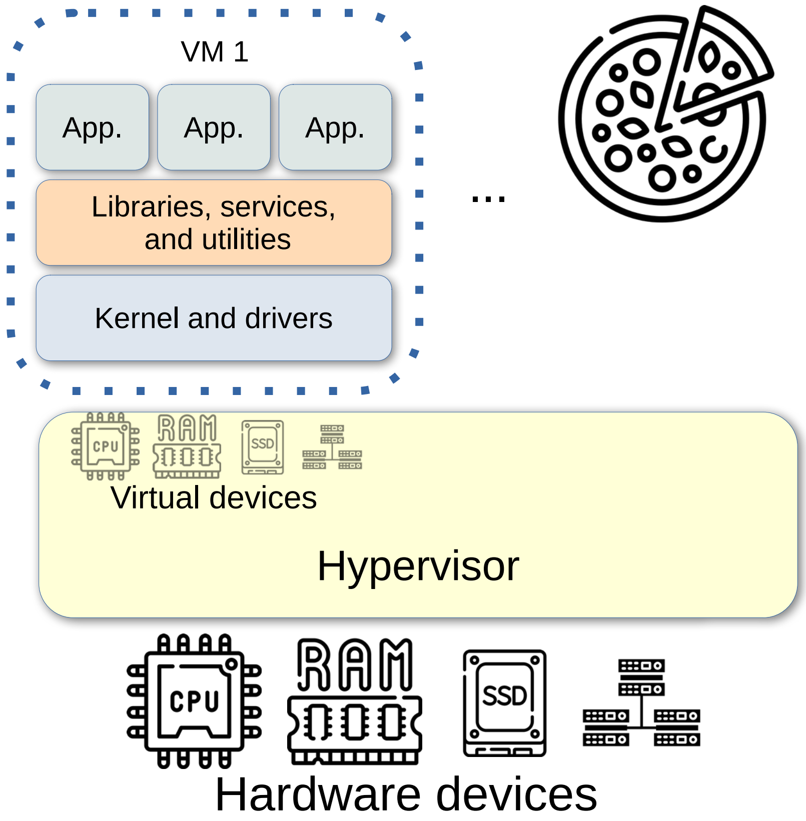
- Libraries, services, and utilities
  - e.g., user interface, ...
- Kernel and device drivers
  - Encapsulates hardware
  - Protects resources
  - e.g., scheduler, file systems, ...



Icons by Flaticon.com.

# Virtualization

- Slice hardware resources for different users / applications
- Each slice looks like an actual machine
  - Virtual machine
- Isolate slices from each other:
  - Security
  - Performance

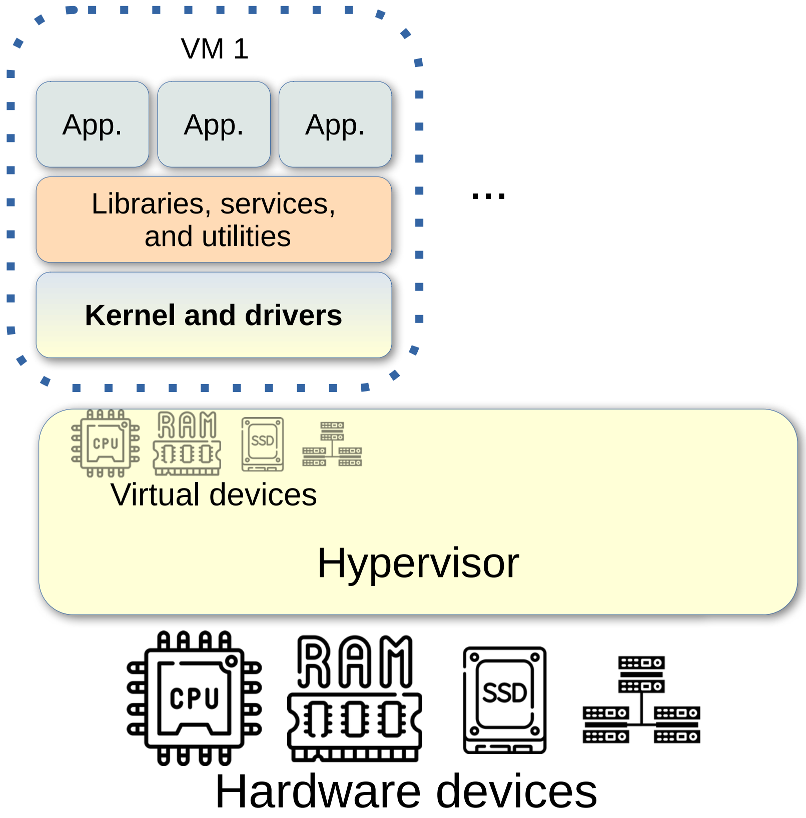


Icons by Flaticon.com.

# Paravirtualization

- How to trick the kernel into accepting virtual devices?

- Modify kernel and/or device driver code to directly use hypervisor services

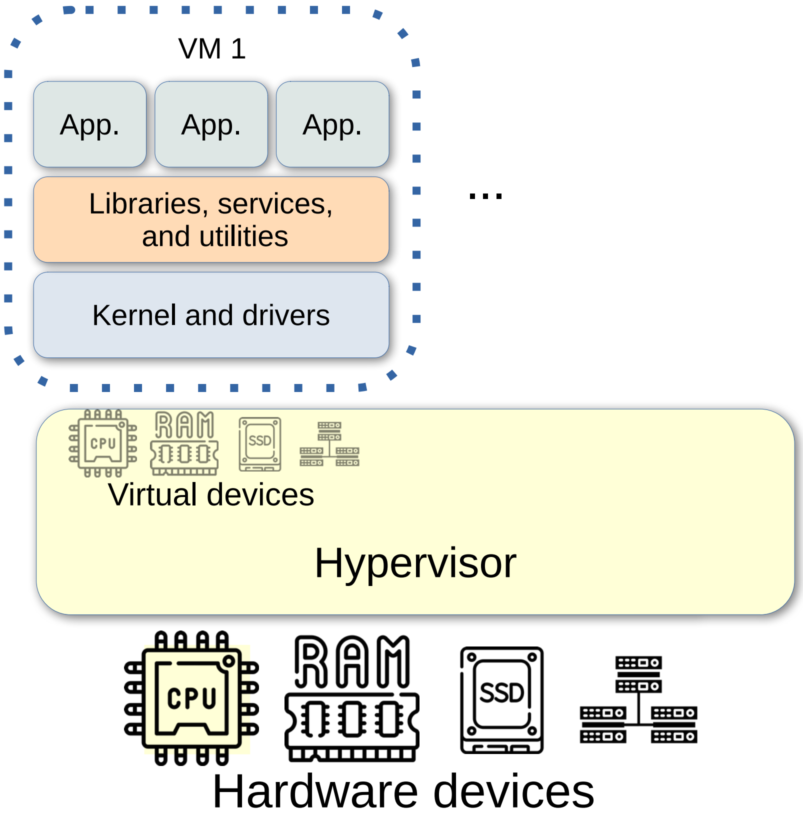


Icons by Flaticon.com.

# Full virtualization

- How to trick the kernel into accepting virtual devices?

- Modify CPU to route VM operations to hypervisor services
  - Priviledged operations

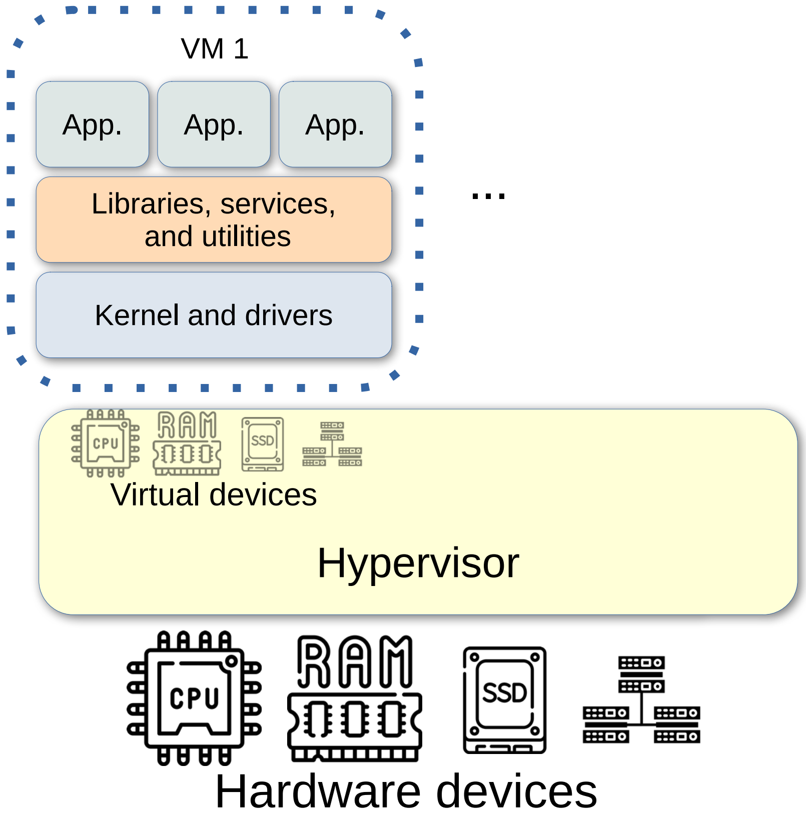


Icons by Flaticon.com.

# Bare metal

- How to bootstrap the hypervisor?

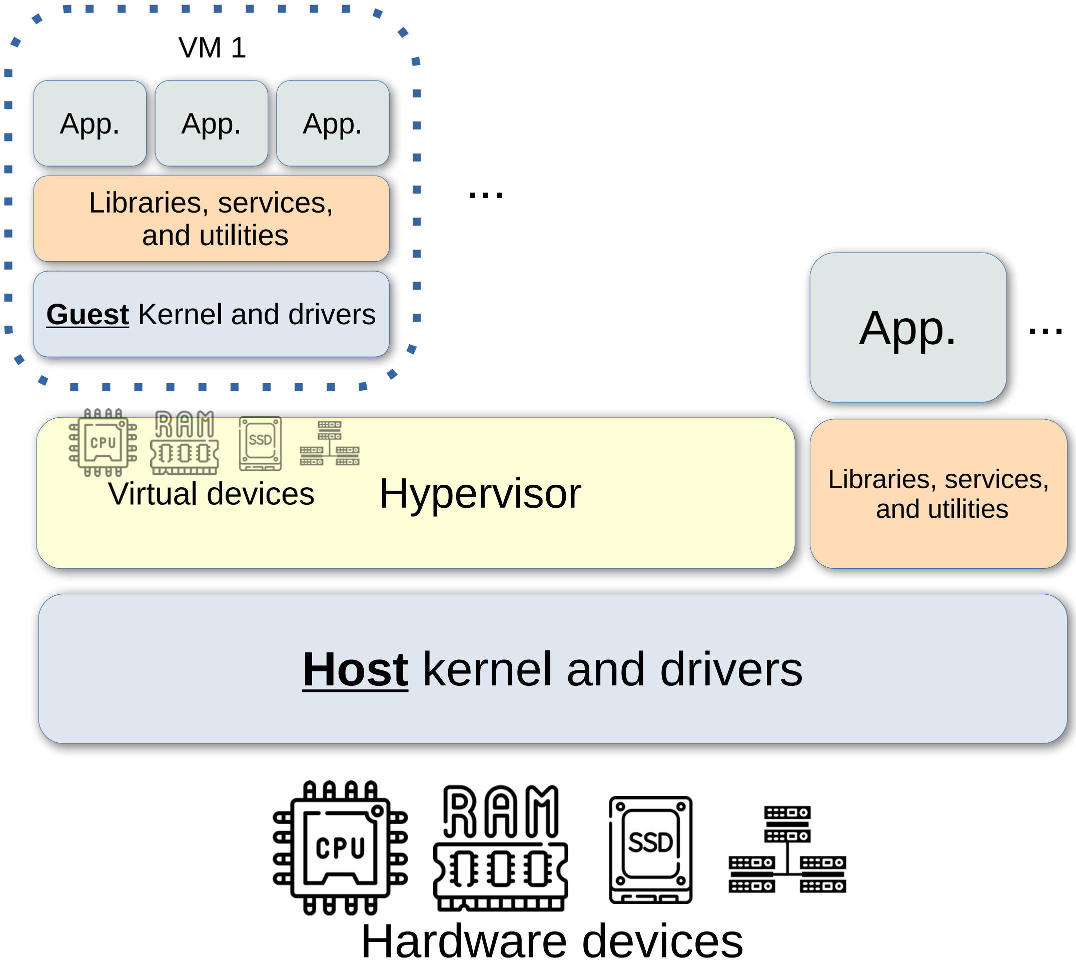
- The hypervisor is itself a small operating system kernel
  - Custom device drivers



Icons by Flaticon.com.

# Hosted

- How to bootstrap the hypervisor?
- The hypervisor is runs on top of a normal operating system kernel
  - Host kernel provides support


VM 1
App.
App.
App.
Libraries, services, and utilities
**Guest** Kernel and drivers
...
App.
...
CPU
RAM
SSD
Virtual devices
Hypervisor
Libraries, services, and utilities
**Host** kernel and drivers
CPU
RAM
SSD
Hardware devices

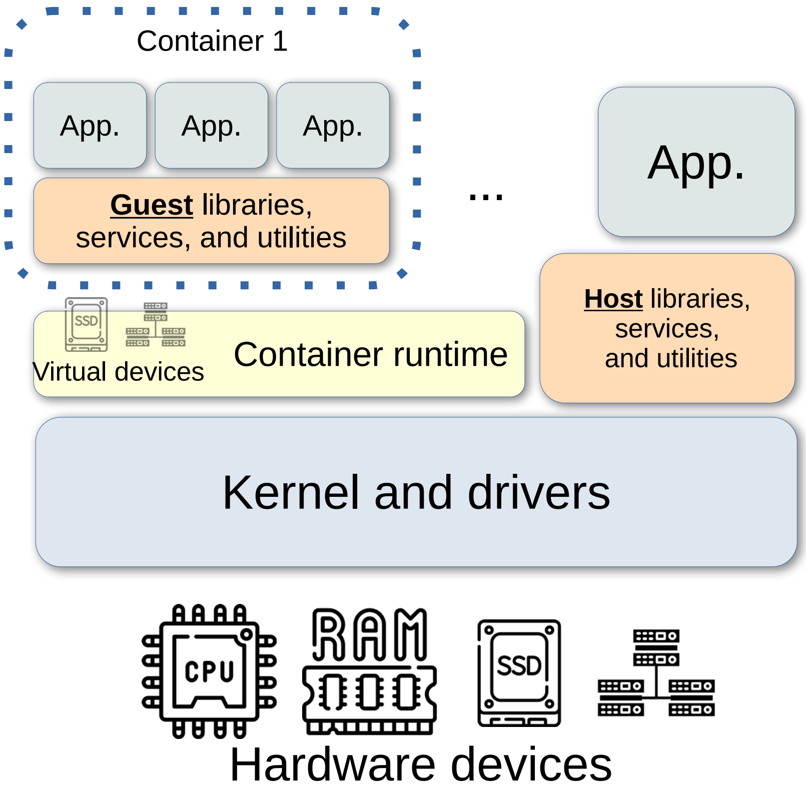
Icons by Flaticon.com.

# Examples

- Full / bare metal
  - Xen (modern), VMWare ESX
- Paravirtualization / bare metal
  - Xen (original)
- Full / hosted
  - Linux KVM, VirtualBox
- Paravirtualization / hosted
  - VirtualBox with “Guest Additions”

# Containerization

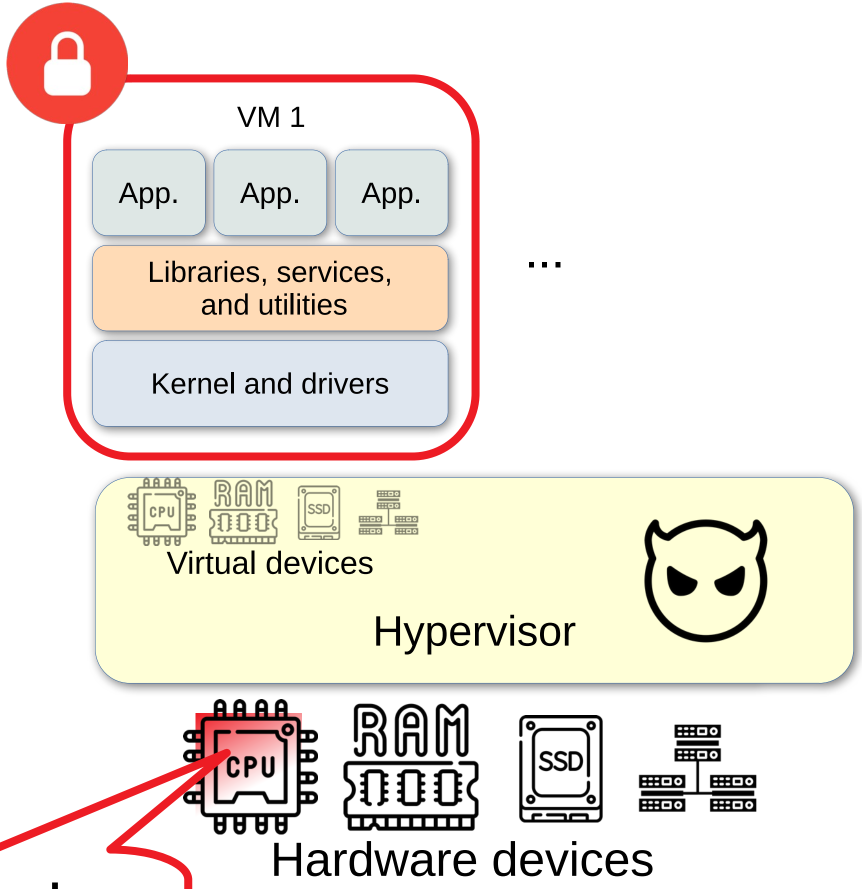
- Weaker isolation
- Lower performance overhead
  - Faster setup and tear down
- Examples:
  - Docker / Podman
  - Kubernetes (K8s)



Icons by Flaticon.com.

# Trusted execution

- Isolation from a malicious hypervisor
  - Data is signed and encrypted
- Relies on the CPU to enforce isolation
- Examples:
  - Intel SGX, ARM TrustZone

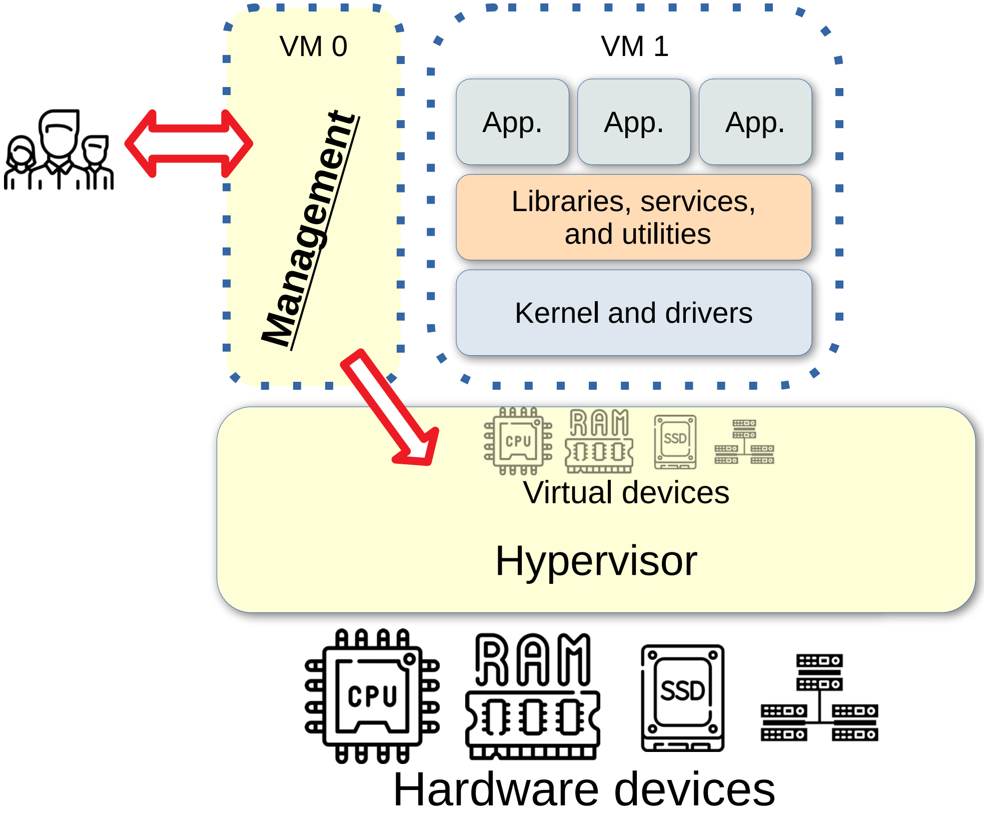


Support for “enclaves”

Icons by Flaticon.com.

# Remote management

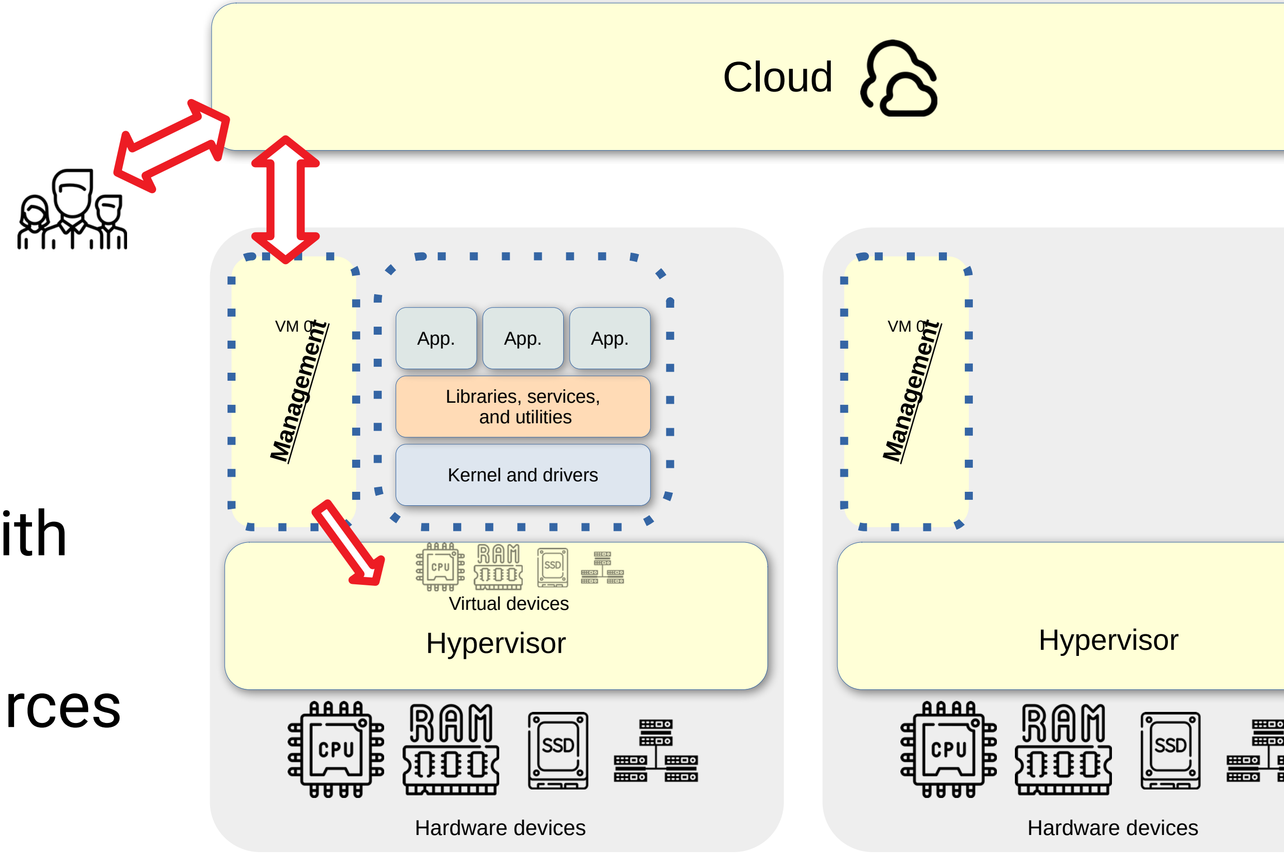
- Add a management service to the physical host:
  - As an additional VM
  - As a host process
- The management service allows
  - Creating, starting, stopping, destroying VMs
  - Allocating physical to virtual resources



Icons by Flaticon.com.

# Cloud services

- Do not directly contact hosts
- A central service:
  - Routes provisioning requests to hosts with available resources
  - Bills users for resources used



Icons by Flaticon.com.

# Elasticity

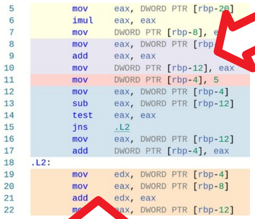
- The ability to dynamically add and remove capacity according to actual needs
  - Avoids expensive over-provisioning
- Elasticity can be managed:
  - Cloud services monitor usage of allocated resources and workload
  - More resources are automatically added when needed
  - Resources are automatically removed when idle

# Cloud services

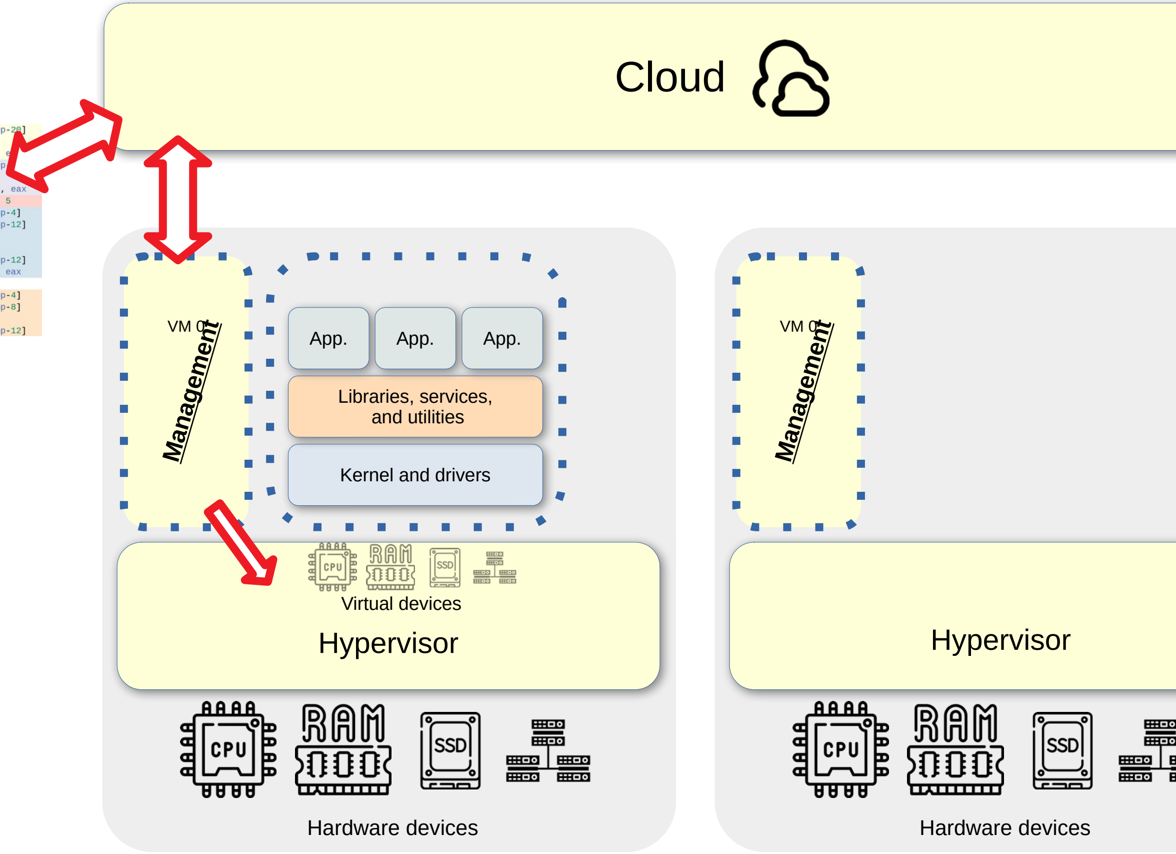
- **Infrastructure as a Service (IaaS)**
  - Raw resources provided by hypervisors
  - Optional managed elasticity (e.g., with K8s)
- **Platform as a Service (PaaS)**
  - Services used by application developers
  - Managed elasticity
  - Examples: Storage (S3), DBaaS (Aurora), FaaS
- **Software as a Service (SaaS)**
  - Services for end-users
  - Managed elasticity
  - Examples: GMail, ...

# Infrastructure as Code (IaC)

- The user does not directly interact with provisioning
- Instead, they write programs that control provisioning
- Provisioning code can be managed as usual (git, ...)



```
        mov     eax, DWORD PTR [rbp-20]
        imul    eax, eax
        mov     DWORD PTR [rbp-8], e
        mov     eax, DWORD PTR [rbp-
        add     eax, eax
        mov     DWORD PTR [rbp-12], eax
        mov     DWORD PTR [rbp-4], 5
        mov     eax, DWORD PTR [rbp-4]
        sub     eax, DWORD PTR [rbp-12]
        test    eax, eax
        jns     .L2
        mov     eax, DWORD PTR [rbp-12]
        add     DWORD PTR [rbp-4], eax
.L2:
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     edx, eax
        mo      eax, DWORD PTR [rbp-12]
```



Icons by Flaticon.com.

# Provisioning

- Step 1: Select VMs / containers and hardware resources
  - CPU, RAM, storage, networking
- Step 2: Install and configure software
  - Operating system base
  - Libraries and services
  - User application
  - Configuration parameters

**Key Issue:** Security and access!

# Provisioning styles

- Imperative: **How** to assemble needed infrastructure?
  - The user describes the steps to perform
  - The user validates that the system is in the correct state and reruns from scratch when needed
- Declarative: **What** infrastructure is needed?
  - The system decides the steps to perform
    - From scratch or from an existing running infrastructure
  - Validates that the system is in the correct state and takes corrective action

# Provisioning scope

- Single instance
  - Describes a single instance
  - Focuses on software installation and configuration
- Orchestration
  - Describes multiple instances
  - Focuses on relation between instances
    - Networking resources
    - Multi-instance constraints (e.g., “at least 3 instances of X”)

# Examples

- Imperative / single
  - Docker
- Imperative / single or orquestration
  - Vagrant
- Declarative / orchestration
  - Kubernetes (K8s)

# Example with Vagrant

- Install and run a Python / NumPy program
- Configuration:
  - VM with 2 cores and 1GB RAM
  - Ubuntu operating system
  - Install Python globally, with PIP and Virtualenv
- Application setup:
  - Download requirements with PIP
  - Use a configuration variable
  - Run on startup

# Example with Vagrant

```
Vagrant.configure("2") do |config|
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--graphicscontroller", "VBoxSVGA"]
    vb.memory = "1024"
    vb.cpus = "2"
  end

  config.vm.box = "cloud-image/ubuntu-24.04"

  config.vm.provision "shell", inline: <<-SHELL
    apt update
    apt install -y python3-pip python3-venv
    chsh -s /bin/bash vagrant
  SHELL

  config.vm.provision "file", source: "./hello.py", destination: "~vagrant/h    o/"
```

Select hardware

Select OS

Install and configure software

Copy local files

# Example with Vagrant

```
  config.vm.provision "shell", privileged: false, inline: <<-SHELL
    python3 -m venv hello/venv
    source hello/venv/bin/activate
    pip install numpy

    echo "export MYPARAM=10" >> ~vagrant/.profile
  SHELL

  config.vm.provision "shell", run: "always", privileged: false, inline: <<-SHELL
    source hello/venv/bin/activate
    python3 ./hello/hello.py ${MYPARAM} >> result.txt
  SHELL
end
```

Install user software

Set configuration variables

Run workload

# Summary

- Virtualization technologies
- Cloud services
- Provisioning tools