



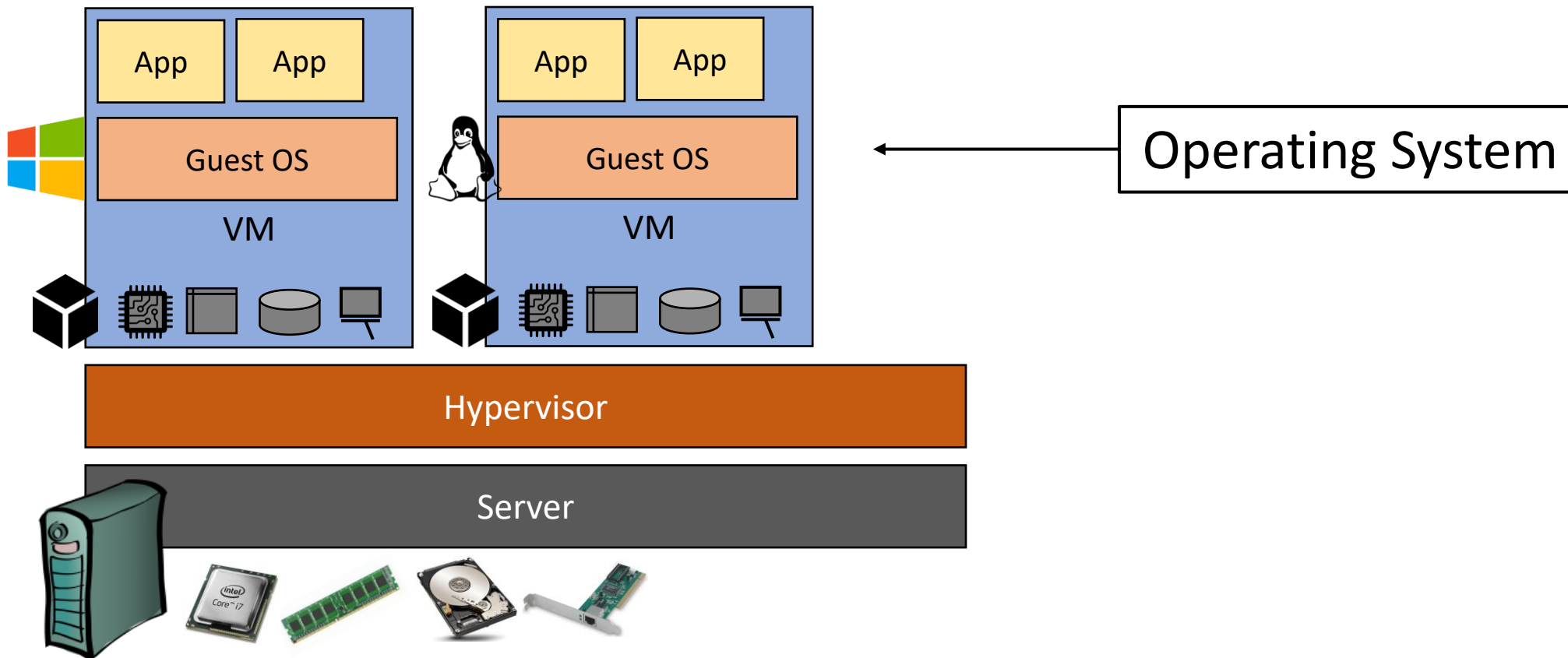
CompSci 401: Cloud Computing

Overhead of Virtual Machines

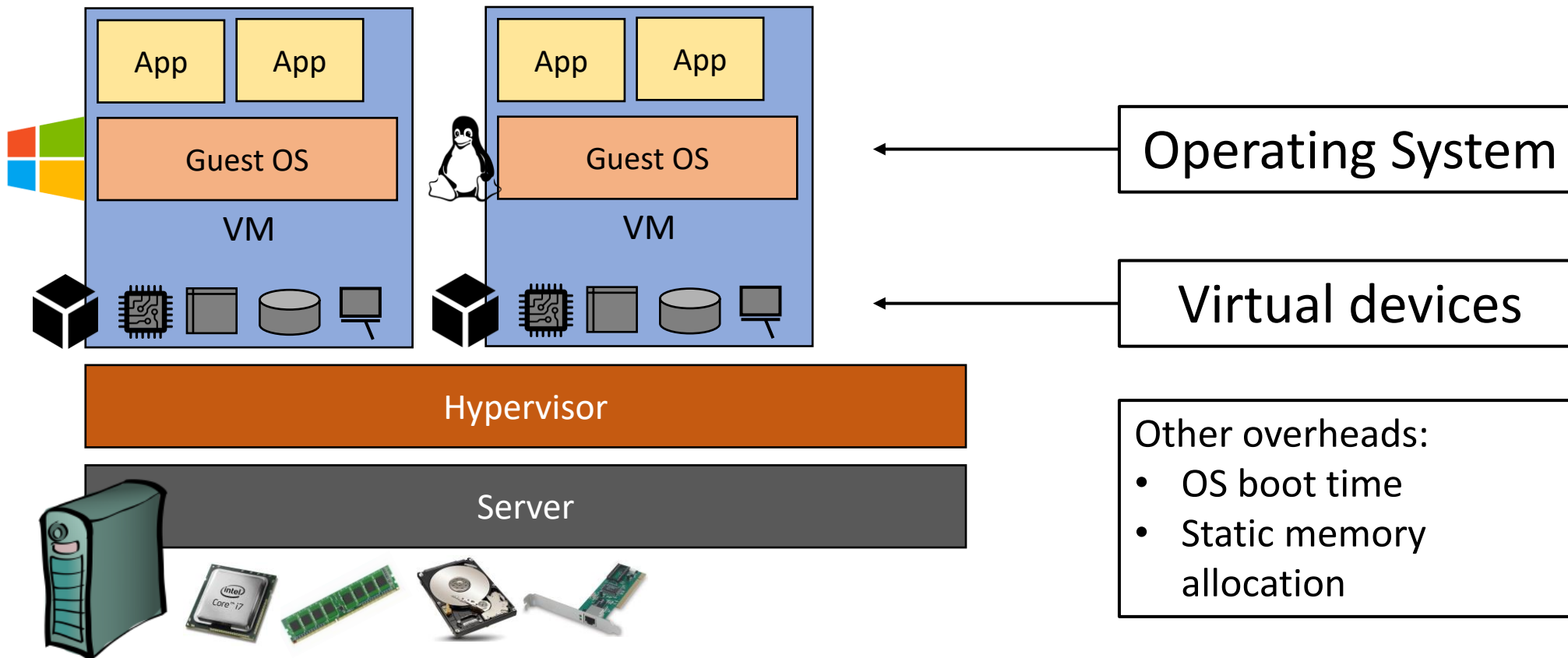
Prof. Ítalo Cunha



VMs provide complete flexibility



VMs provide complete flexibility, at a cost



Virtual machines and autoscaling

- VM overhead is amortized when applications run permanently
 - Pay overhead for booting the operating system once
- VM overhead is amortized when multiple applications run in the OS
 - Pay overhead of the operating system once

Virtual machines and autoscaling

- VM overhead is amortized when applications run permanently
 - Pay overhead for booting the operating system once
- VM overhead is amortized when multiple applications run in the OS
 - Pay overhead of the operating system once
- Autoscaling means applications do not run permanently
- Most deployments have a single application

Virtual machines and autoscaling

- VM overhead is amortized when applications run permanently
 - Pay overhead for booting the operating system once
- VM overhead is amortized when multiple applications run in the OS
 - Pay overhead of the operating system once
- Autoscaling means applications do not run permanently
- Most deployments have a single application

Virtual machines are not a great match for highly dynamic cloud applications



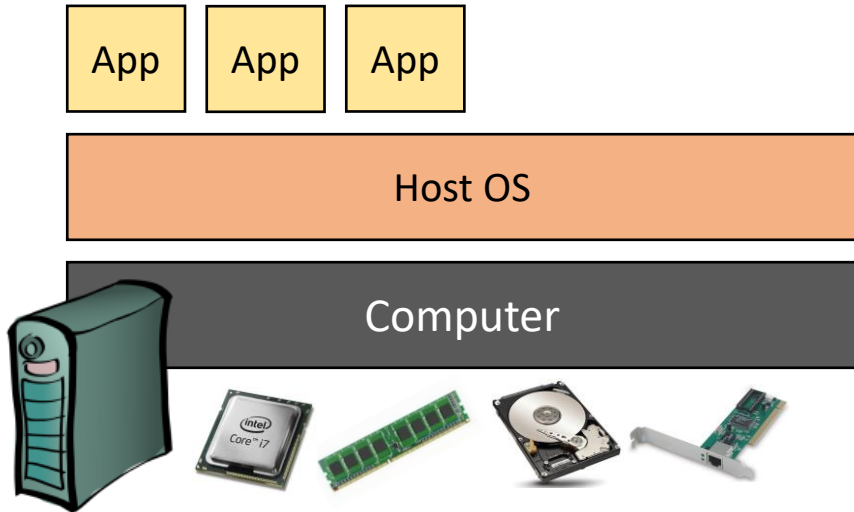
CompSci 401: Cloud Computing

Process Management

Prof. Ítalo Cunha



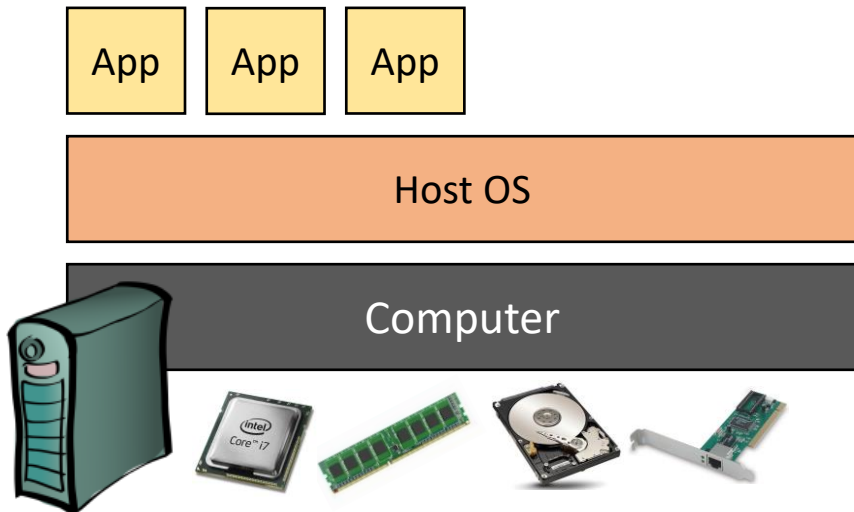
Operating systems manage applications



Operating systems manage applications

Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

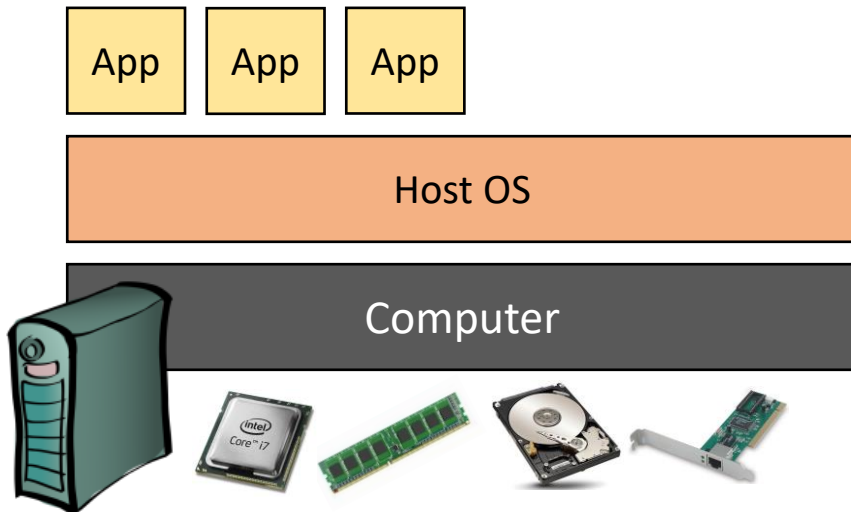


Operating systems manage applications

Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

OSes do not enforce complete isolation



Operating systems manage applications

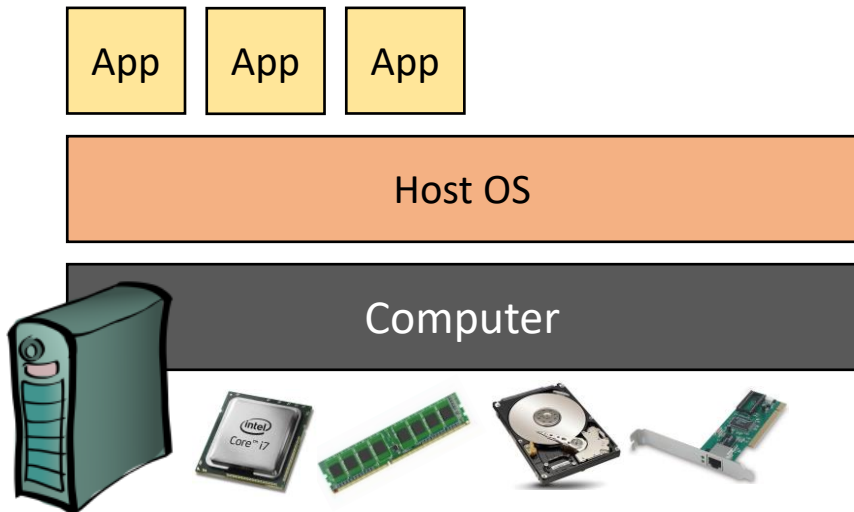
Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

OSes do not enforce complete isolation

Isolation facilities in operating systems:

- Virtual address spaces isolate memory
- User IDs and filesystem permissions isolate data



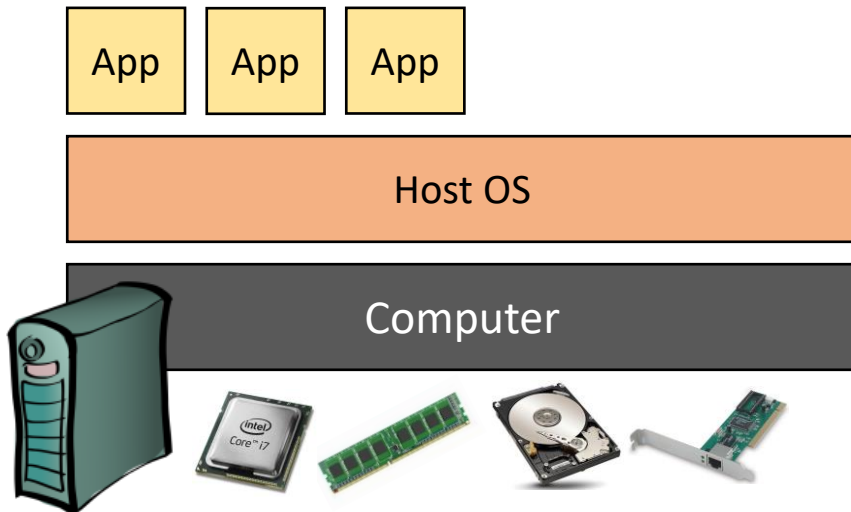
Operating systems manage applications

Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

OSes do not enforce complete isolation

- Shared access to the network (IP address, ports)
- Shared filesystem
- Shared metadata
 - What processes are running



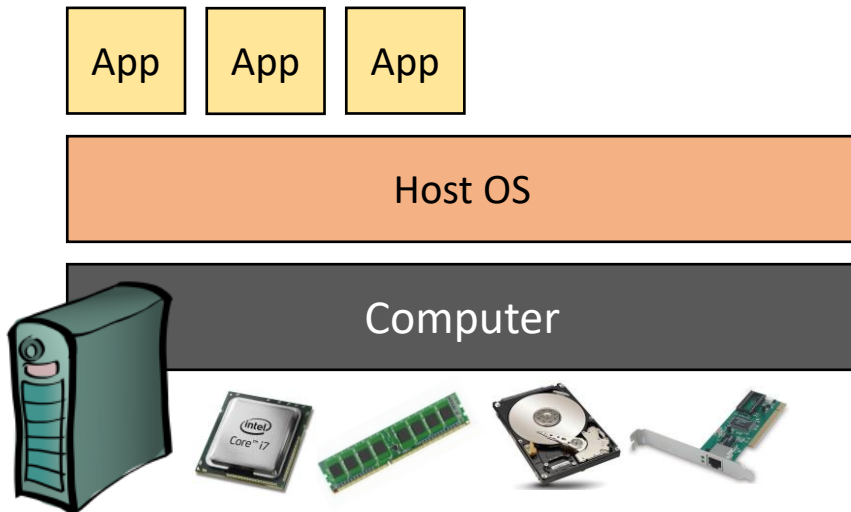
Operating systems manage applications

Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

OSes do not enforce complete isolation

- Shared access to the network (IP address, ports)
- Shared filesystem
- Shared metadata
 - What processes are running
- Free competition to access hardware (memory, CPU)



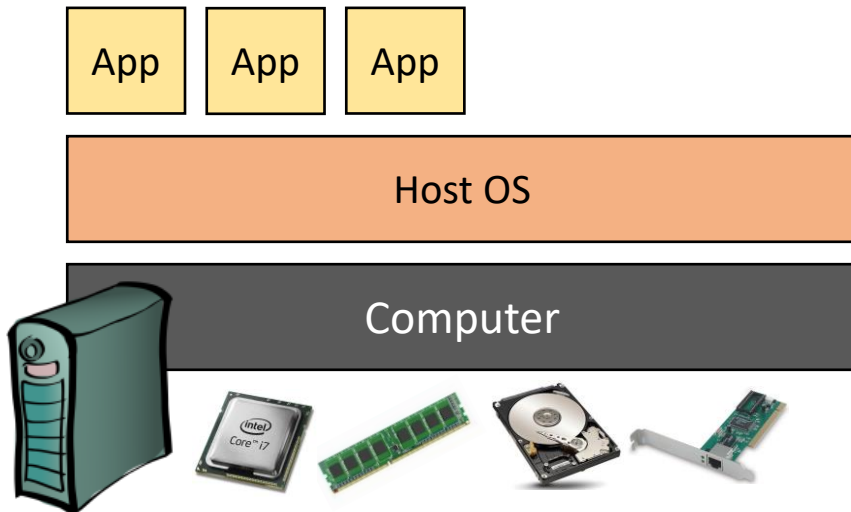
Operating systems manage applications

Applications in an operating system

- Start and terminate quickly
- More direct, efficient access to hardware
- Share memory effectively
- Can scale running instances on multiple servers

OSes do not enforce complete isolation

- Shared access to the network (IP address, ports)
- Shared filesystem
- Shared metadata
 - What processes are running
- Free competition to access hardware (memory, CPU)



Sharing leaks information from one application to another



CompSci 401: Cloud Computing

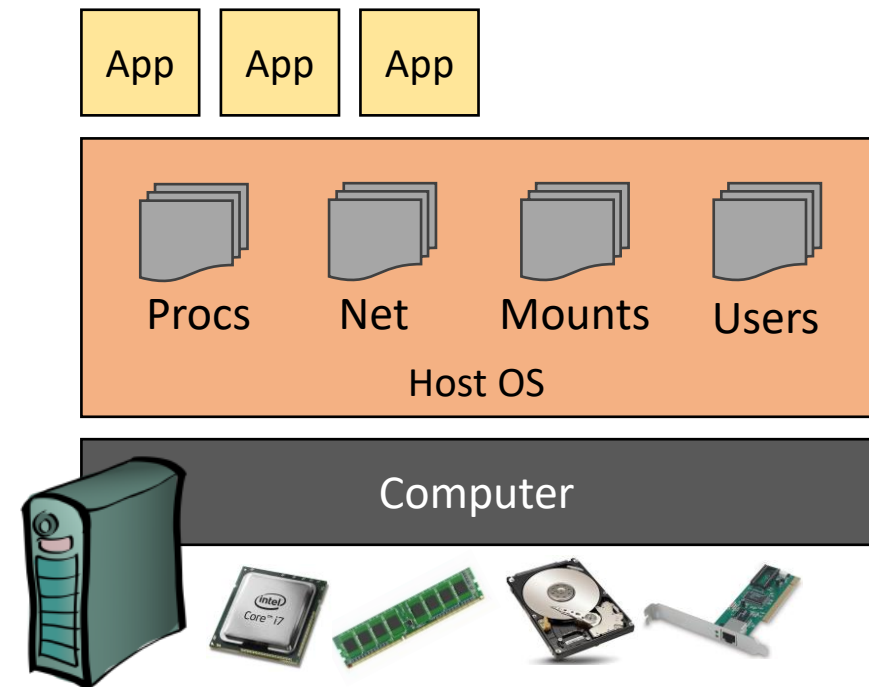
Containers

Prof. Ítalo Cunha



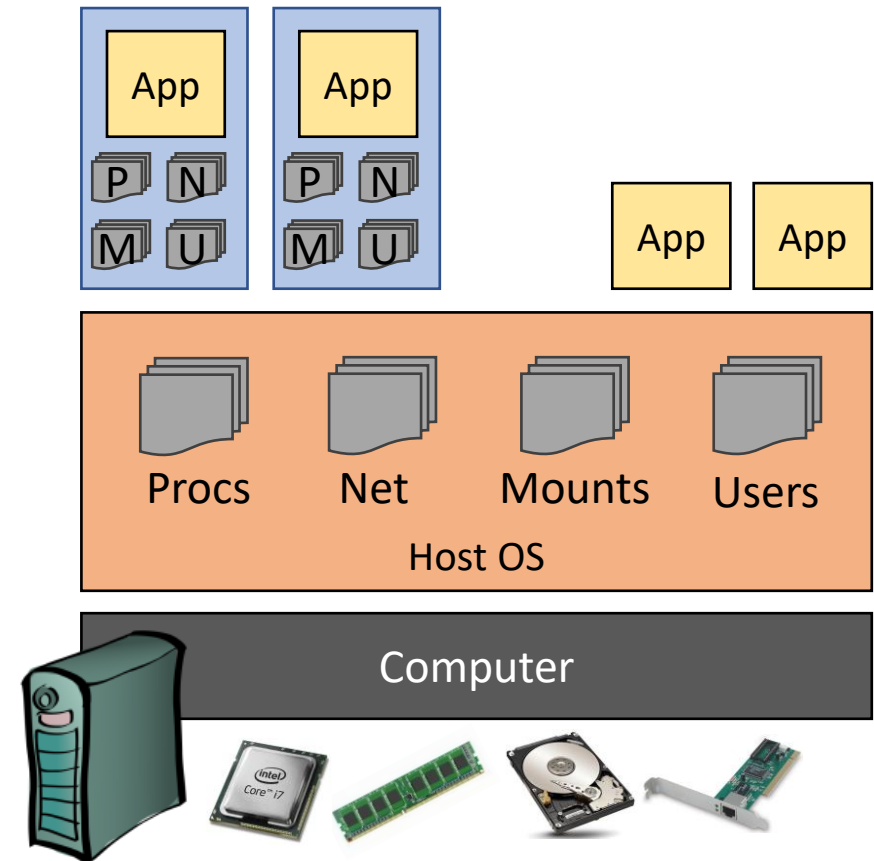
Shared resources within operating systems

- Operating systems manages resources that are shared across apps
- Protection and isolation mechanisms are not thorough



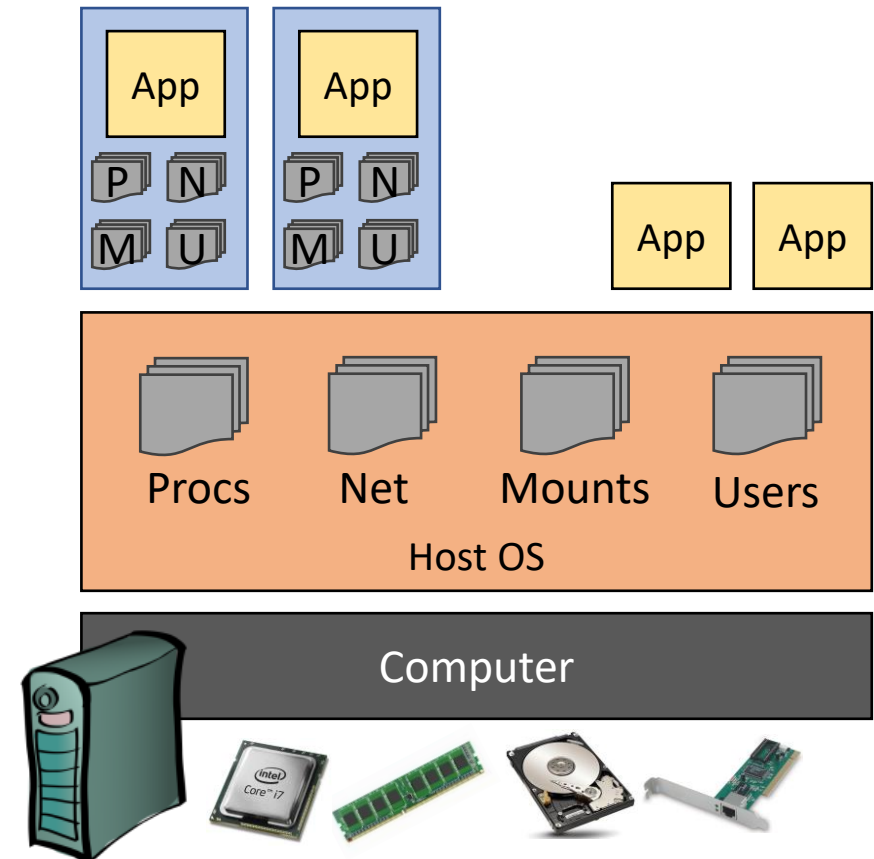
Containers have dedicated resources

- Containers do not share any resources with the host OS



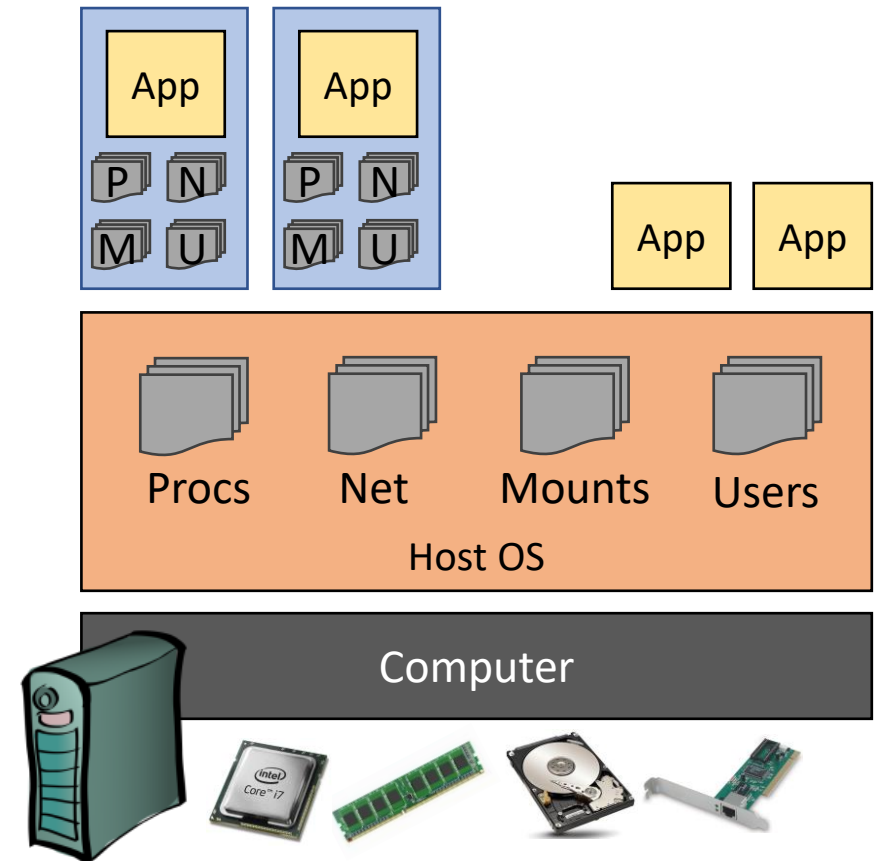
Containers have dedicated resources

- Containers do not share any resources with the host OS
- Linux namespaces:
 - Mount (filesystem)
 - Processes
 - Network
 - Interprocess communication
 - Users
 - Name resolution



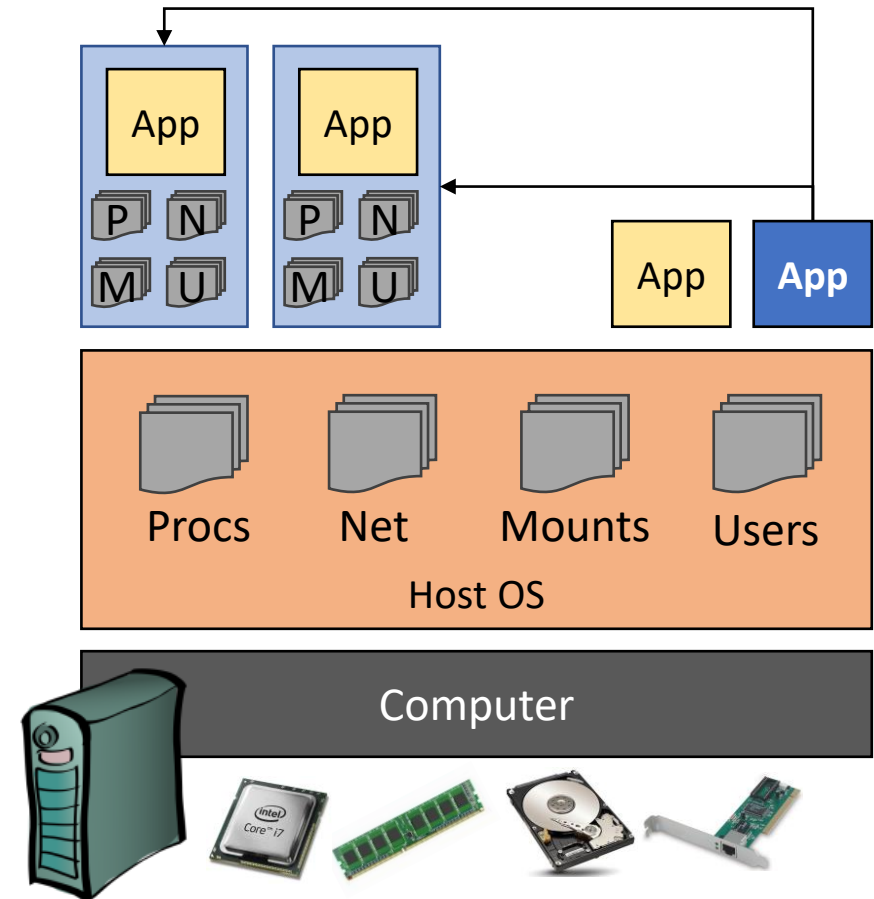
Containers have dedicated resources

- Containers do not share any resources with the host OS
- Linux namespaces:
 - Mount (filesystem)
 - Processes
 - Network
 - Interprocess communication
 - Users
 - Name resolution
- Resource allocation



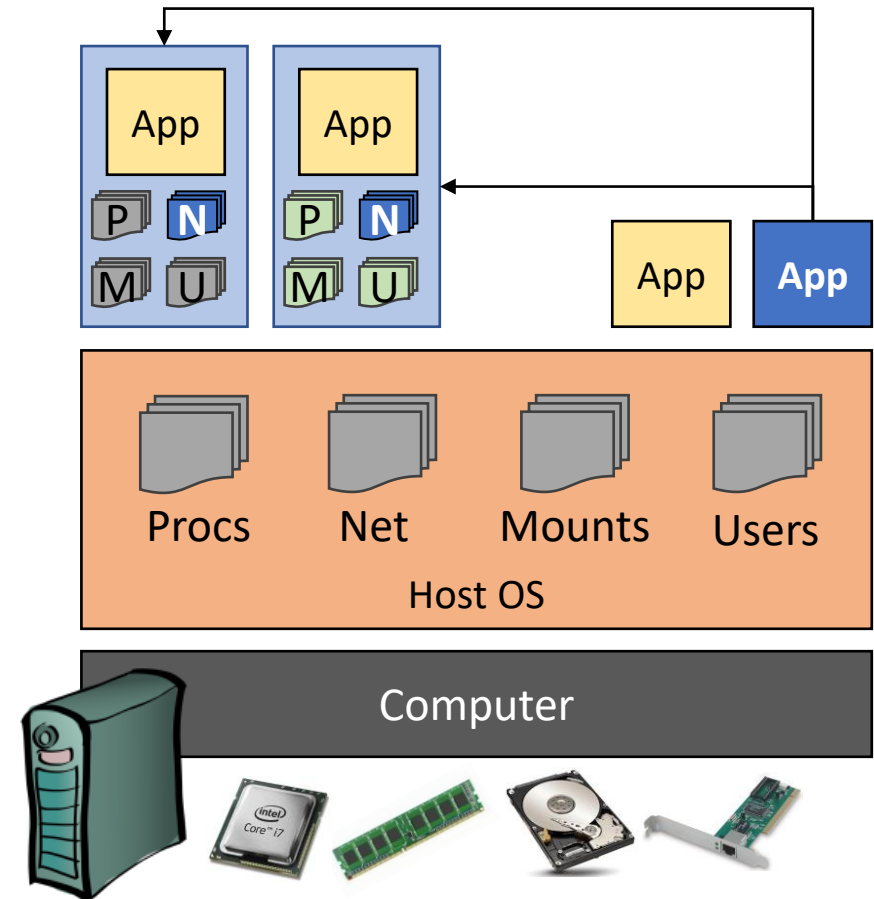
Controlled sharing of resources

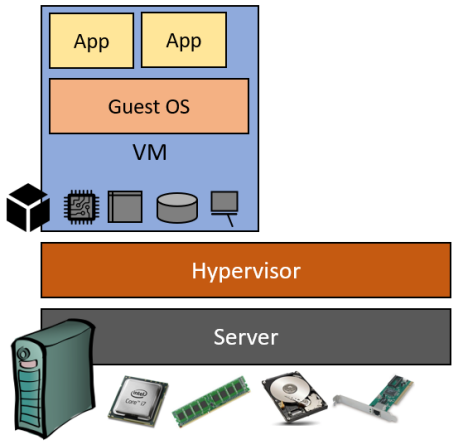
- **Privileged** applications on the host can manage and inspect containers
 - Usually limited to management software like Docker or LXC



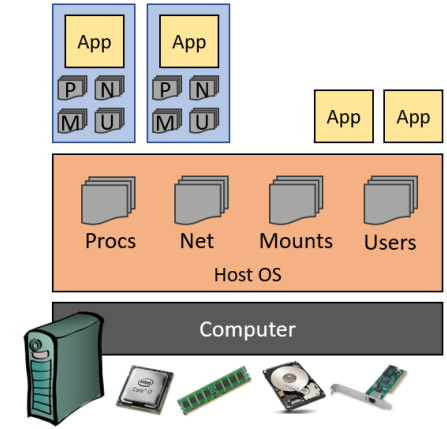
Controlled sharing of resources

- **Privileged** applications on the host can manage and inspect containers
 - Usually limited to management software like Docker or LXC
- Containers can be configured to **share namespaces**
 - Useful when building sets of containers that cooperate





VM vs containers



Virtual Machines

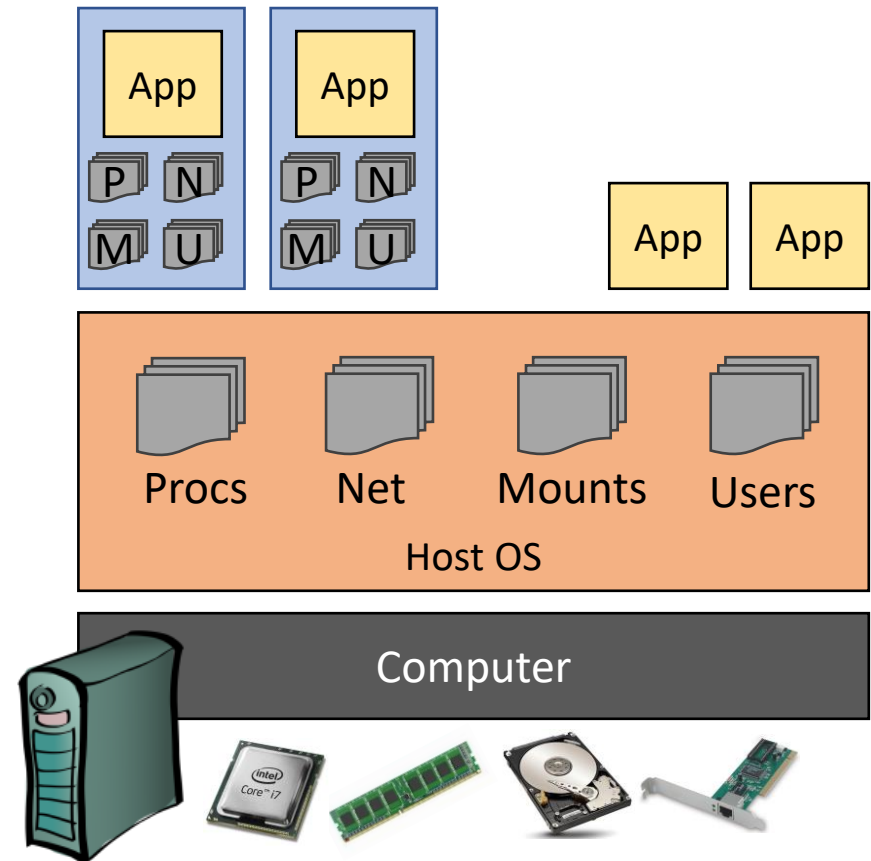
- Slow boot/shutdown sequences
- Operating system overhead
- Total isolation between VMs
- Complete flexibility

Containers

- Fast startup/teardown
- Small resource tracking overhead
- Fine-grained control of isolation
- Runs on host operating system

Containers run on the host OS

- Applications within containers are limited to the host OS
- Cannot run Windows applications on Linux containers
 - Need a virtual machine for that
- Applications cannot use features not supported by the OS kernel
 - May require kernel updates





CompSci 401: Cloud Computing

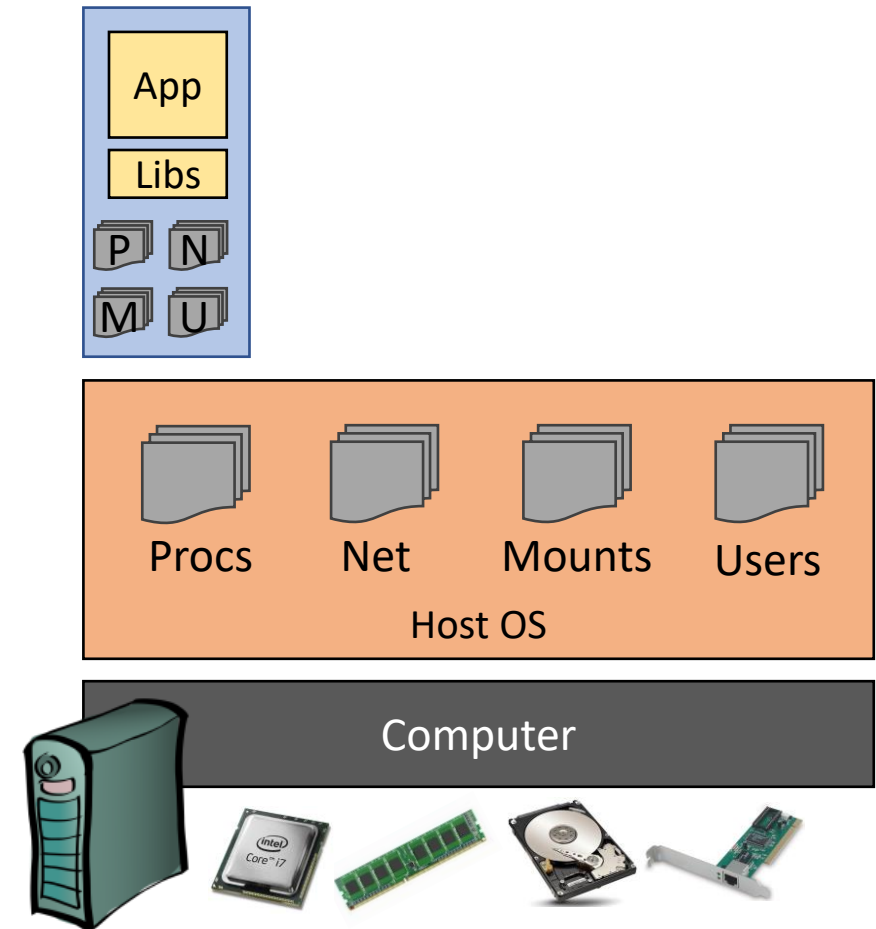
Docker

Prof. Ítalo Cunha



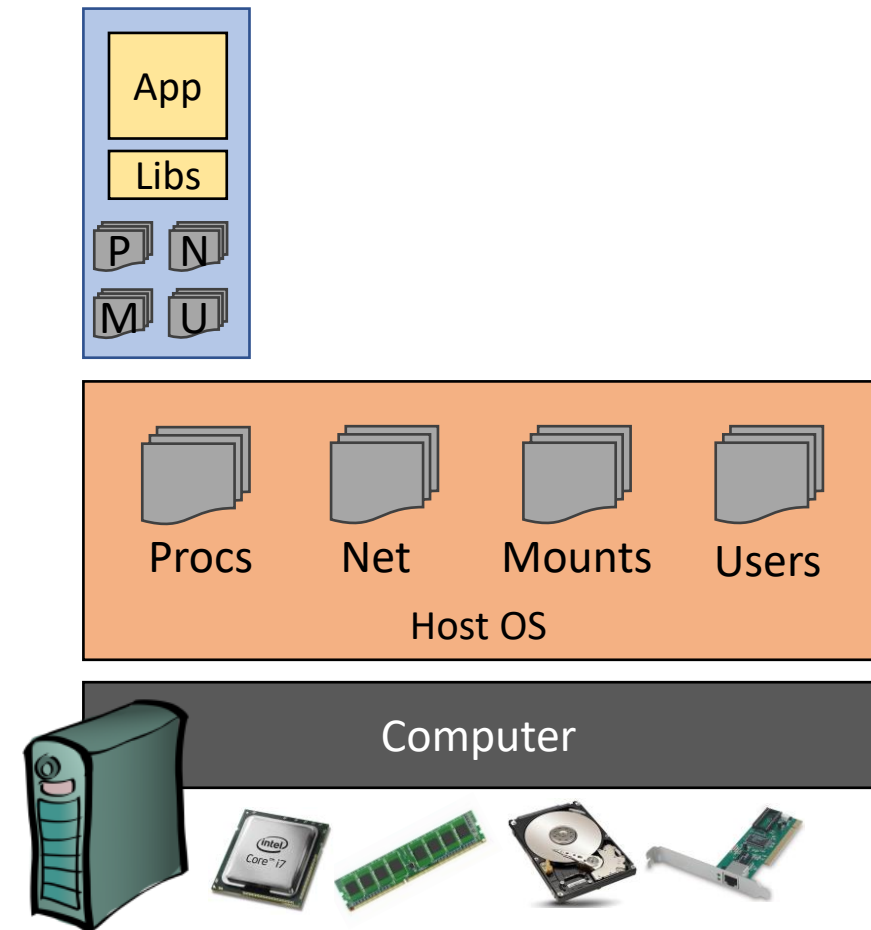
Docker containers

- Small containers running a single application
 - One container image per application
 - Small images speed up download
 - Single application speeds up startup time



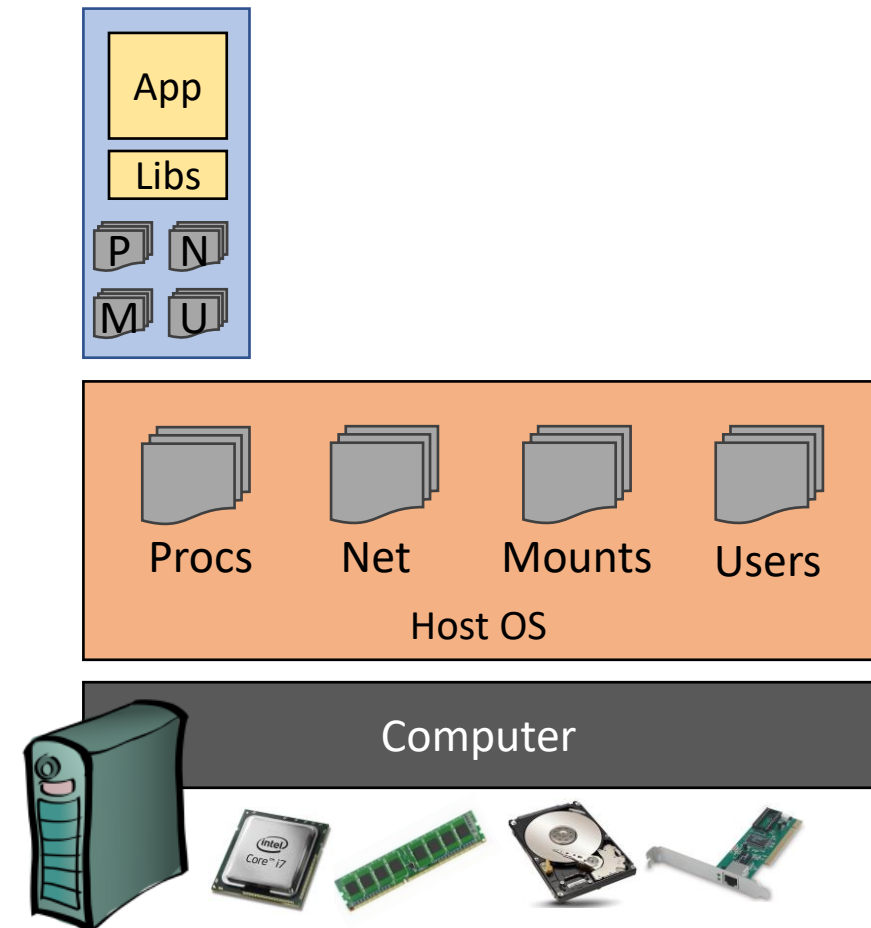
Docker containers

- Small containers running a single application
 - One container image per application
 - Small images speed up download
 - Single application speeds up startup time
- Libraries and dependencies bundled in image
 - No dependencies on host OS
 - Works on all OSes



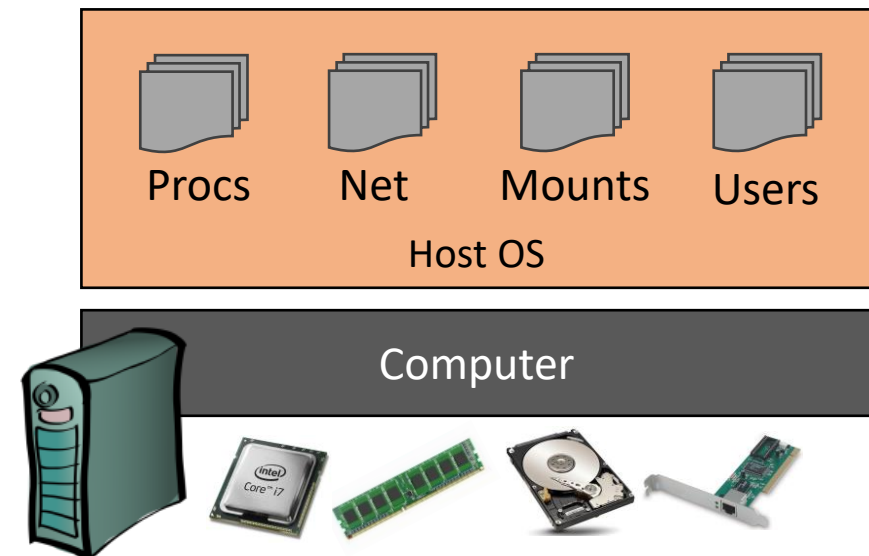
Docker containers

- Small containers running a single application
 - One container image per application
 - Small images speed up download
 - Single application speeds up startup time
- Libraries and dependencies bundled in image
 - No dependencies on host OS
 - Works on all OSes
- Read-only images
 - Changes to container to not persist upon restart
 - Reproducible execution
 - Container starts from a known state every time



Building Docker containers

- Tooling around the development and deployment of containers
 - [Dockerfile](#) set of instructions
- Extensive image registry (DockerHub)
 - Ready-to-use containers/applications
 - Building blocks for more complex containers



Dockerfile: a container recipe

Django uses Python

```
FROM python:3.7-slim-buster

# Install dependencies
RUN apt-get update
RUN apt-get install --no-install-recommends -y bird
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
RUN apt-get install --no-install-recommends -y procps

# Copy application to container
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app

# Install modules in requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt

ENTRYPOINT entrypoint.sh
```

Dockerfile: a container recipe

Django uses Python

We prefer Debian

```
FROM python:3.7-slim-buster

# Install dependencies
RUN apt-get update
RUN apt-get install --no-install-recommends -y bird
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
RUN apt-get install --no-install-recommends -y procps

# Copy application to container
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app

# Install modules in requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt

ENTRYPOINT entrypoint.sh
```


Dockerfile: a container recipe

Django uses Python

Install other applications and system components we need

We prefer Debian

```
FROM python:3.7-slim-buster

# Install dependencies
RUN apt-get update
RUN apt-get install --no-install-recommends -y bird
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
RUN apt-get install --no-install-recommends -y procps

# Copy application to container
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app

# Install modules in requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt

ENTRYPOINT entrypoint.sh
```

Dockerfile: a container recipe

We prefer Debian

Django uses Python

Install other applications and system components we need

Copy our application to the container

```
FROM python:3.7-slim-buster

# Install dependencies
RUN apt-get update
RUN apt-get install --no-install-recommends -y bird
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
RUN apt-get install --no-install-recommends -y procps

# Copy application to container
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app

# Install modules in requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt

ENTRYPOINT entrypoint.sh
```

Dockerfile: a container recipe

We prefer Debian

Django uses Python

```
FROM python:3.7-slim-buster
```

Install other applications and system components we need

```
# Install dependencies
```

```
RUN apt-get update
```

```
RUN apt-get install --no-install-recommends -y bird
```

```
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
```

```
RUN apt-get install --no-install-recommends -y procs
```

Copy our application to the container

```
# Copy application to container
```

```
RUN mkdir -p /usr/src/app
```

```
WORKDIR /usr/src/app
```

```
COPY . /usr/src/app
```

Install Python modules (including Django)

```
# Install modules in requirements.txt
```

```
RUN pip3 install --no-cache-dir -r requirements.txt
```

```
ENTRYPOINT entrypoint.sh
```

Dockerfile: a container recipe

We prefer Debian

Django uses Python

```
FROM python:3.7-slim-buster
```

Install other applications and system components we need

```
# Install dependencies
```

```
RUN apt-get update
```

```
RUN apt-get install --no-install-recommends -y bird
```

```
RUN apt-get install --no-install-recommends -y libgit2-27 libgit2-dev
```

```
RUN apt-get install --no-install-recommends -y procps
```

Copy our application to the container

```
# Copy application to container
```

```
RUN mkdir -p /usr/src/app
```

```
WORKDIR /usr/src/app
```

```
COPY . /usr/src/app
```

Install Python modules (including Django)

```
# Install modules in requirements.txt
```

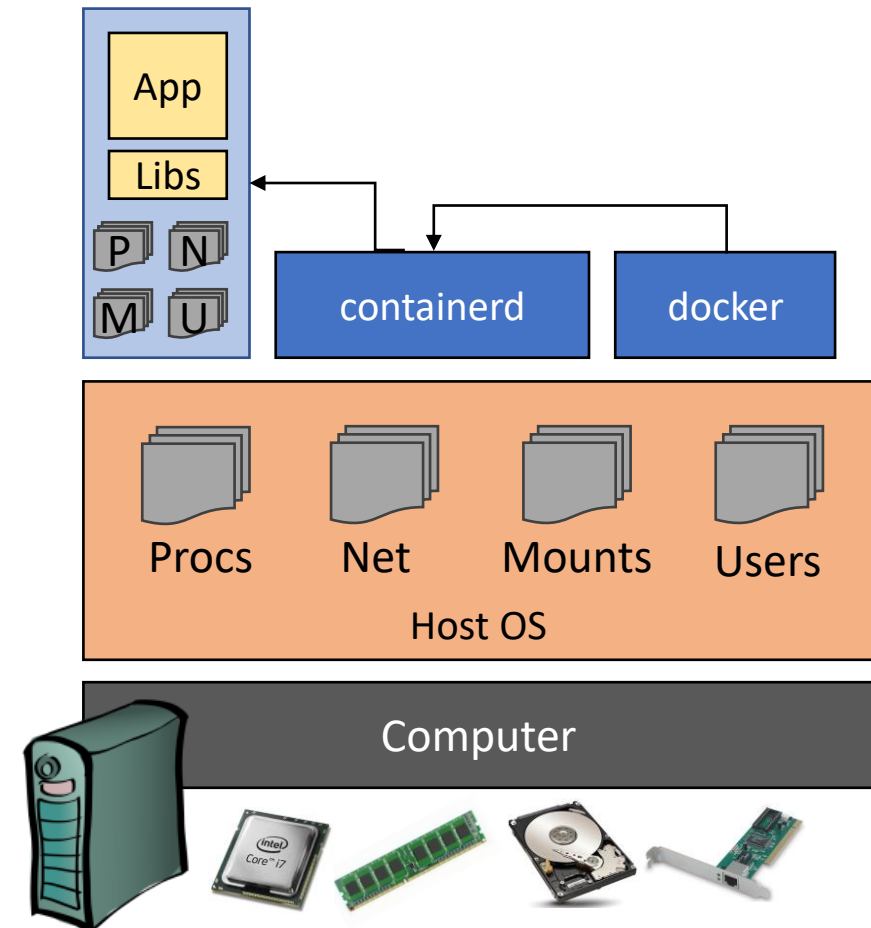
```
RUN pip3 install --no-cache-dir -r requirements.txt
```

Set the application to run inside the container

```
ENTRYPOINT entrypoint.sh
```

Docker container runtime

- Container runtime controls container
- User issues commands to the runtime using CLI tools or network APIs
- Runtime sets up required components in the operating system
 - Creates and destroys namespaces
 - Configures namespaces
 - For example, the network and volumes



Open source and standards

- Docker is a toolset
- The container runtime has been standardized and open-sourced
 - containerd.io
- Other controllers for the container runtime exist
 - Kubernetes
 - Red Hat Podman

Linux Containers (LXC)

- LXC is another tool to run containers on Linux
 - Relies on the same technologies (namespaces)
- Different approach from Docker
 - More similar to a virtual machine
 - Larger images with a complete user environment
 - Multiple applications running in the container
 - State persists between executions of the same container