

CS 4530 Software Engineering

Module 13: Principles and Patterns of Cloud Infrastructure

Adeel Bhutta and Mitch Wand

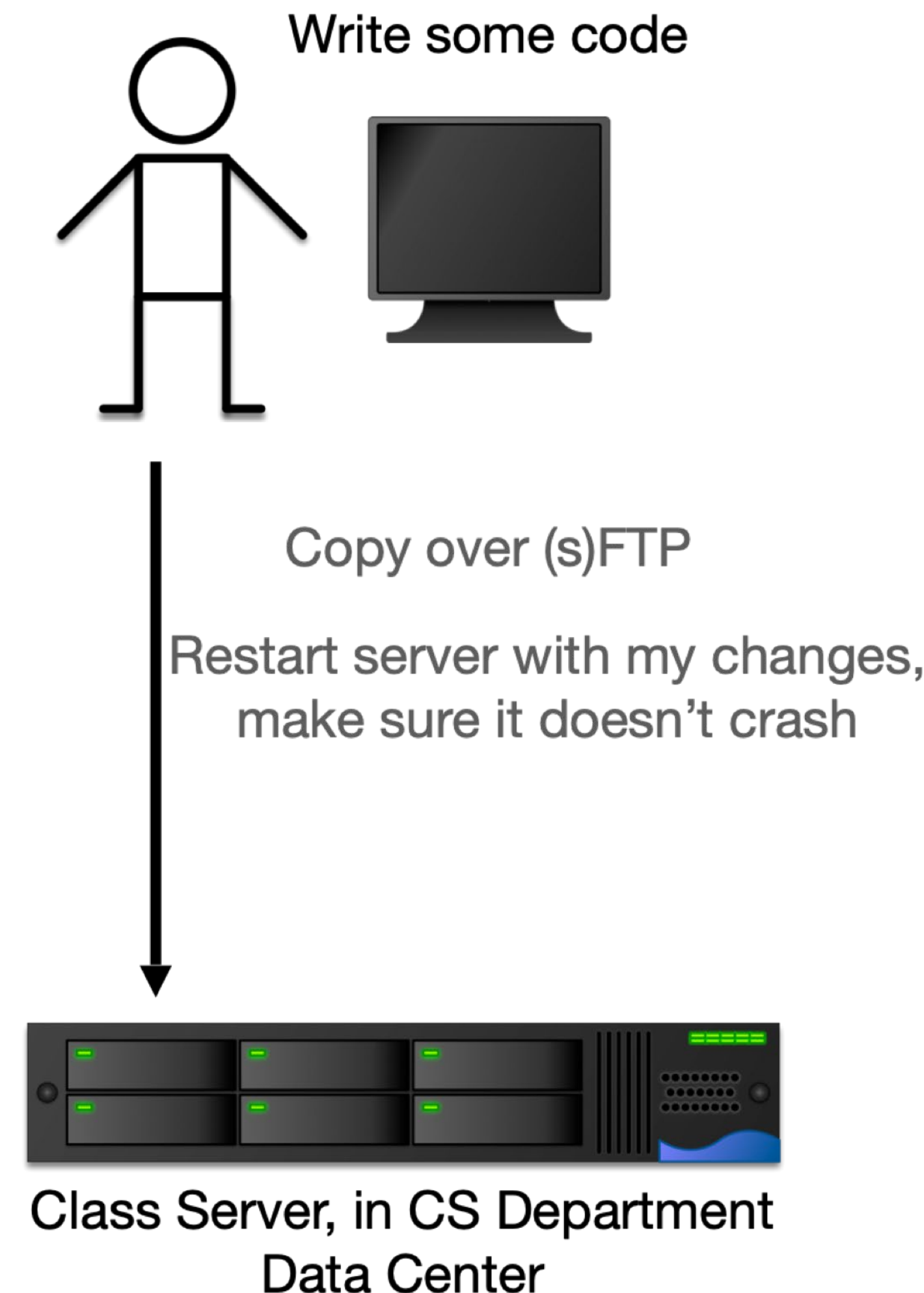
Khoury College of Computer Sciences

Learning objectives for this lesson

- By the end of this lesson, you should be able to...
 - Explain what “cloud” computing is and why it is important
 - Describe the difference between virtual machines and containers
 - Explain why virtual machines and containers are important in cloud computing

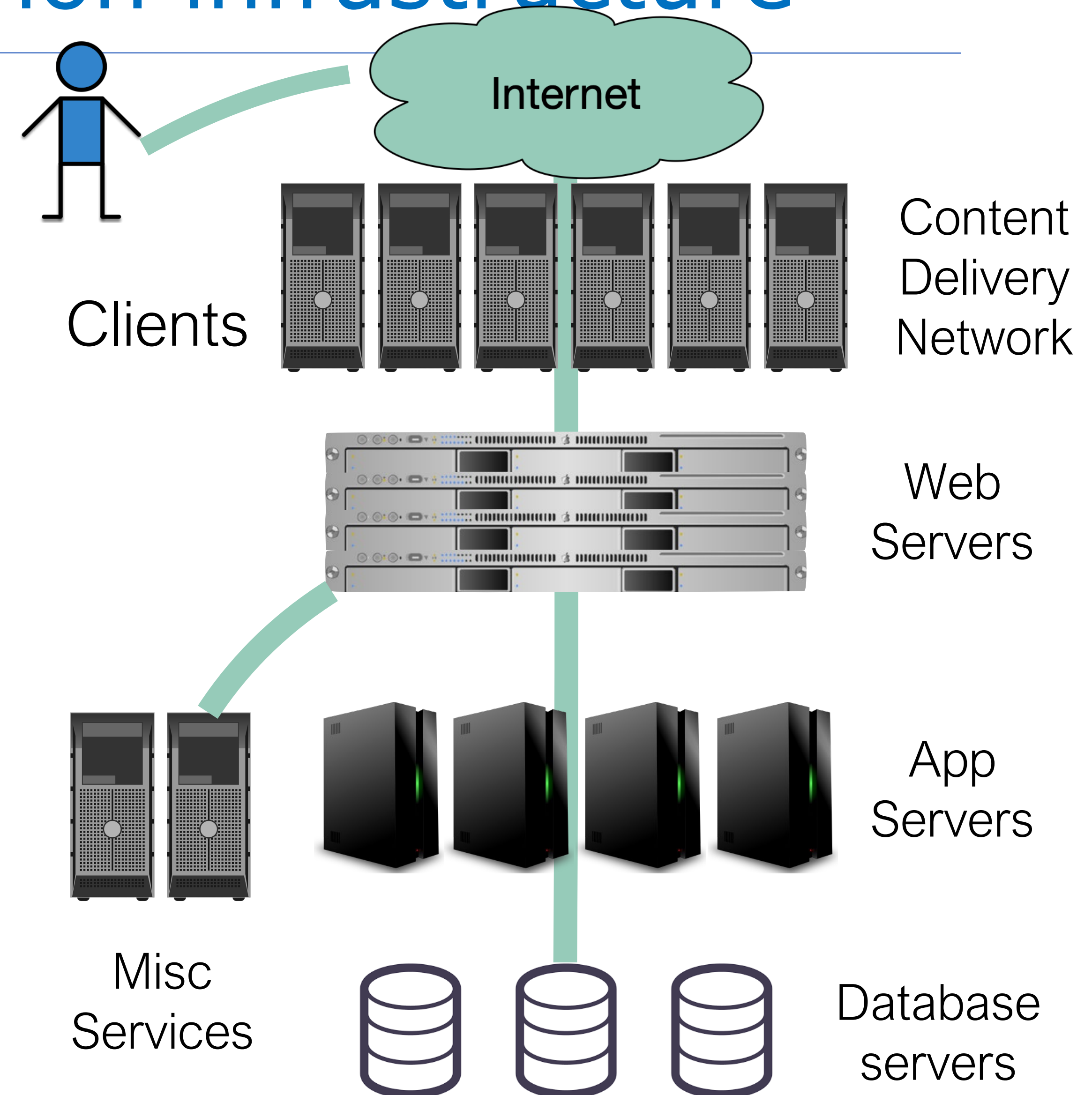
How to deploy web apps?

- What we need:
 - A server that can run our application
 - A network that is configured to route requests from an address to that server
- Questions to think about:
 - What software do we need to run besides our application code?
 - Where does this server come from?
 - Who else gets to use this server?
 - Who maintains the server and software?

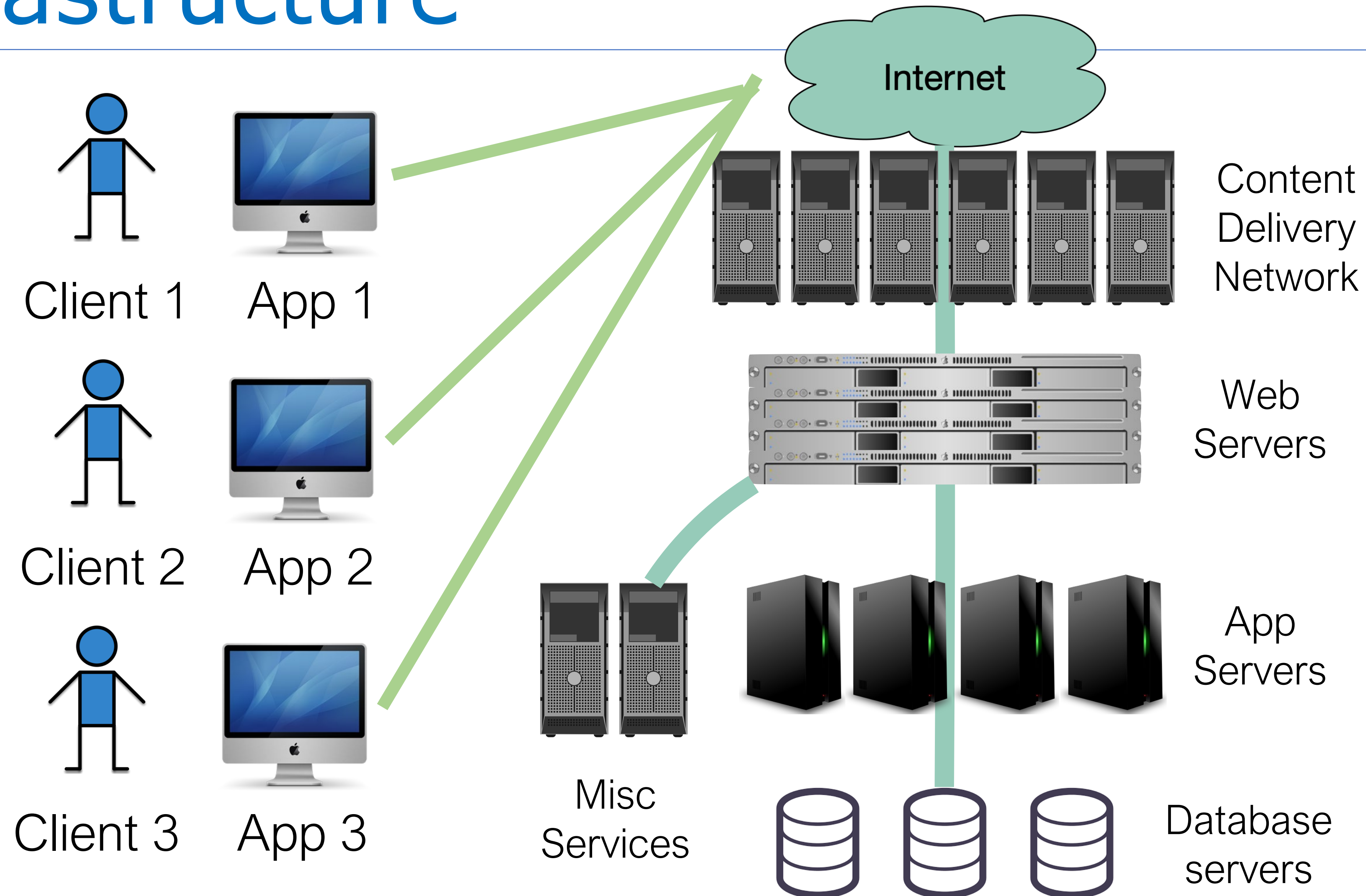


Many apps rely on common infrastructure

- Content delivery network: caches static content “at the edge” (e.g. cloudflare, Akamai)
- Web servers: Speak HTTP, serve static content, load balance between app servers (e.g. haproxy, traefik)
- App servers: Runs our application
- Misc services: Logging, monitoring, firewall
- Database servers: Persistent data



Many apps typically share the same infrastructure



What is the infrastructure that needs to be shared?

- Our apps run on a “tall stack” of dependencies
- Traditionally this full stack is self-managed
- Cloud providers offer products that manage parts of that stack for us:
 - “Infrastructure as a service”
 - “Platform as a service”
 - “Software as a Service”

Application	Application
Middleware	Middleware
Operating System	Operating System
Virtualization	Virtualization
Physical Server	Physical Server
Storage	Storage
Network	Network
Physical data center	Physical data center
Traditional, on-premises computing	Platform-as-a-Service
<i>Self-managed</i>	<i>Vendor-managed</i>

Cloud infrastructure creates economies of scale

- At the physical level:
 - Multiple customers' physical machines in the same data center
 - Save on physical costs (centralize power, cooling, security, maintenance)
- At the physical server level:
 - Multiple customers' virtual machines in the same physical machine
 - Save on resource costs (utilize marginal computing capacity)
- At the application level:
 - Multiple customer's applications hosted in same virtual machine
 - Save on resource overhead (eliminate redundant infrastructure like OS)

Application

Middleware

Operating System

Virtualization

Physical Server

Storage

Network

Physical data center

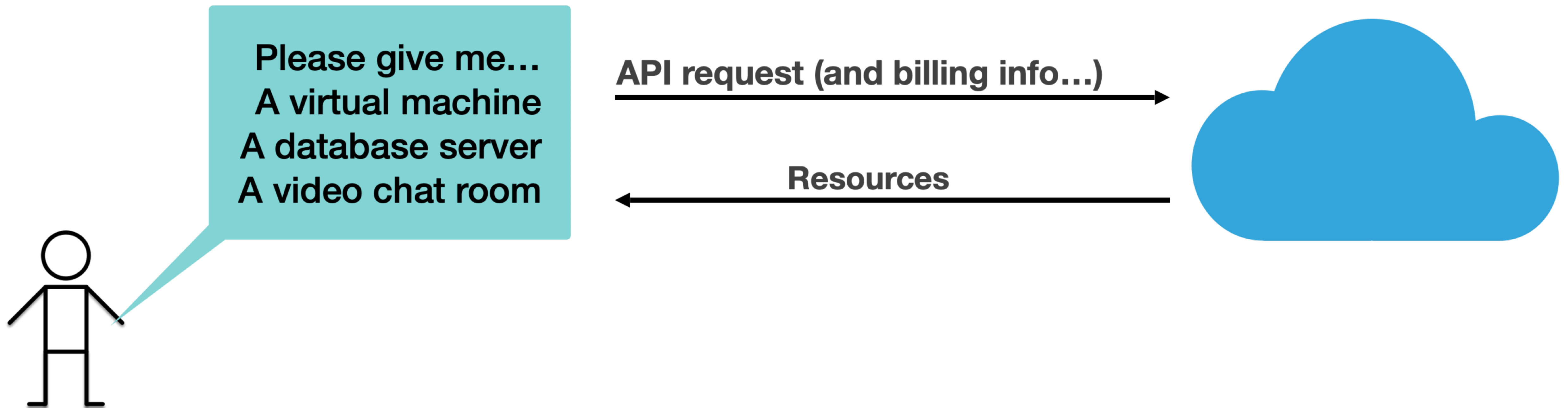
*Multiple customers
could share each of
these tiers*

Cloud infrastructure scales elastically

- “Traditional” computing infrastructure requires capital investment
 - “Scaling up” means buying more hardware, or maintaining excess capacity for when scale is needed
 - “Scaling down” means selling hardware, or powering it off
- Cloud computing scales elastically:
 - “Scaling up” means allocating more shared resources
 - “Scaling down” means releasing resources into a pool
 - Billed on consumption (usually per-second, per-minute or per-hour)

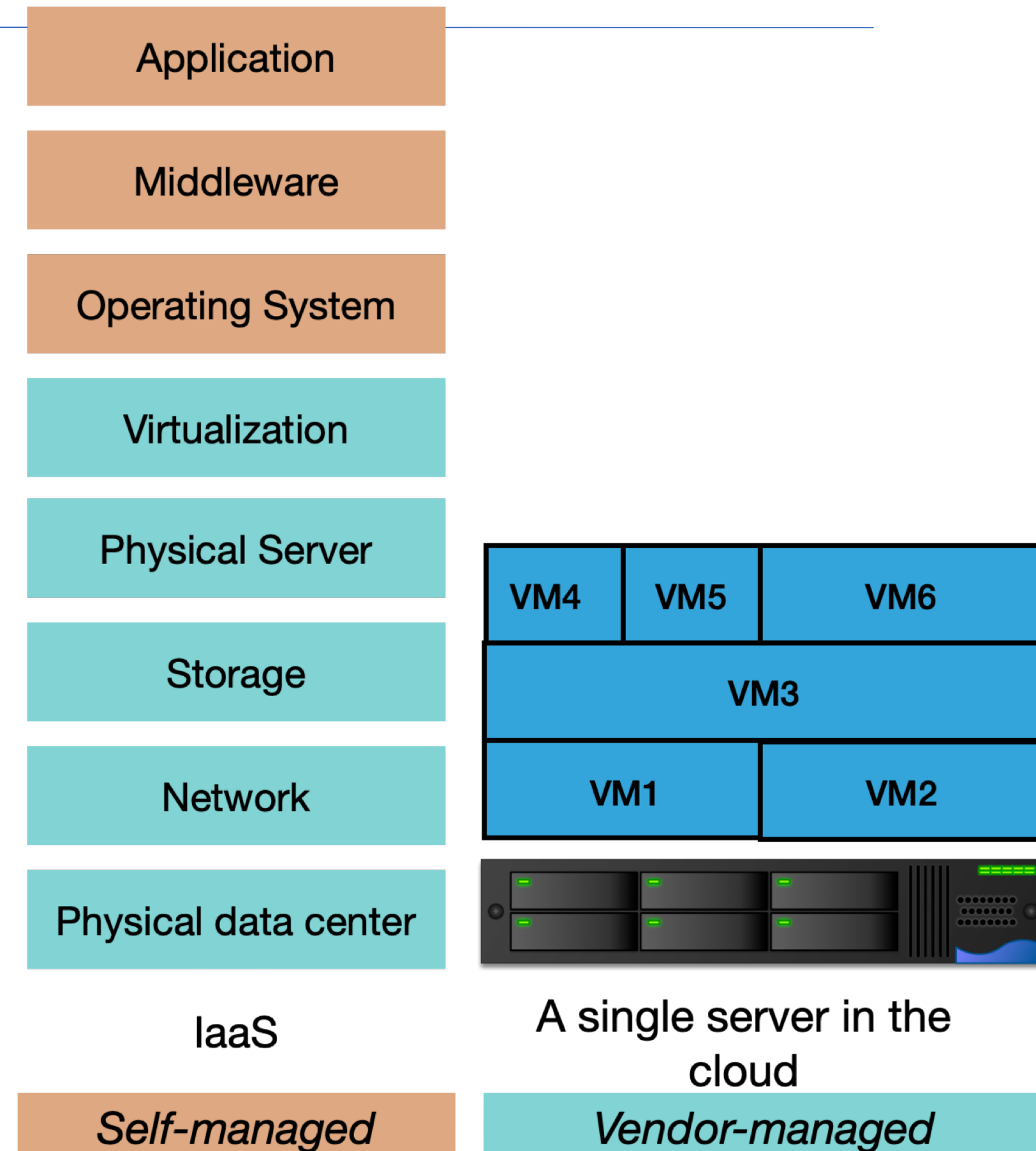
Cloud infrastructure gives on-demand access to resources

- Vendor provides a service catalog of “X as a service” abstractions
- API allows us to provision resources on-demand



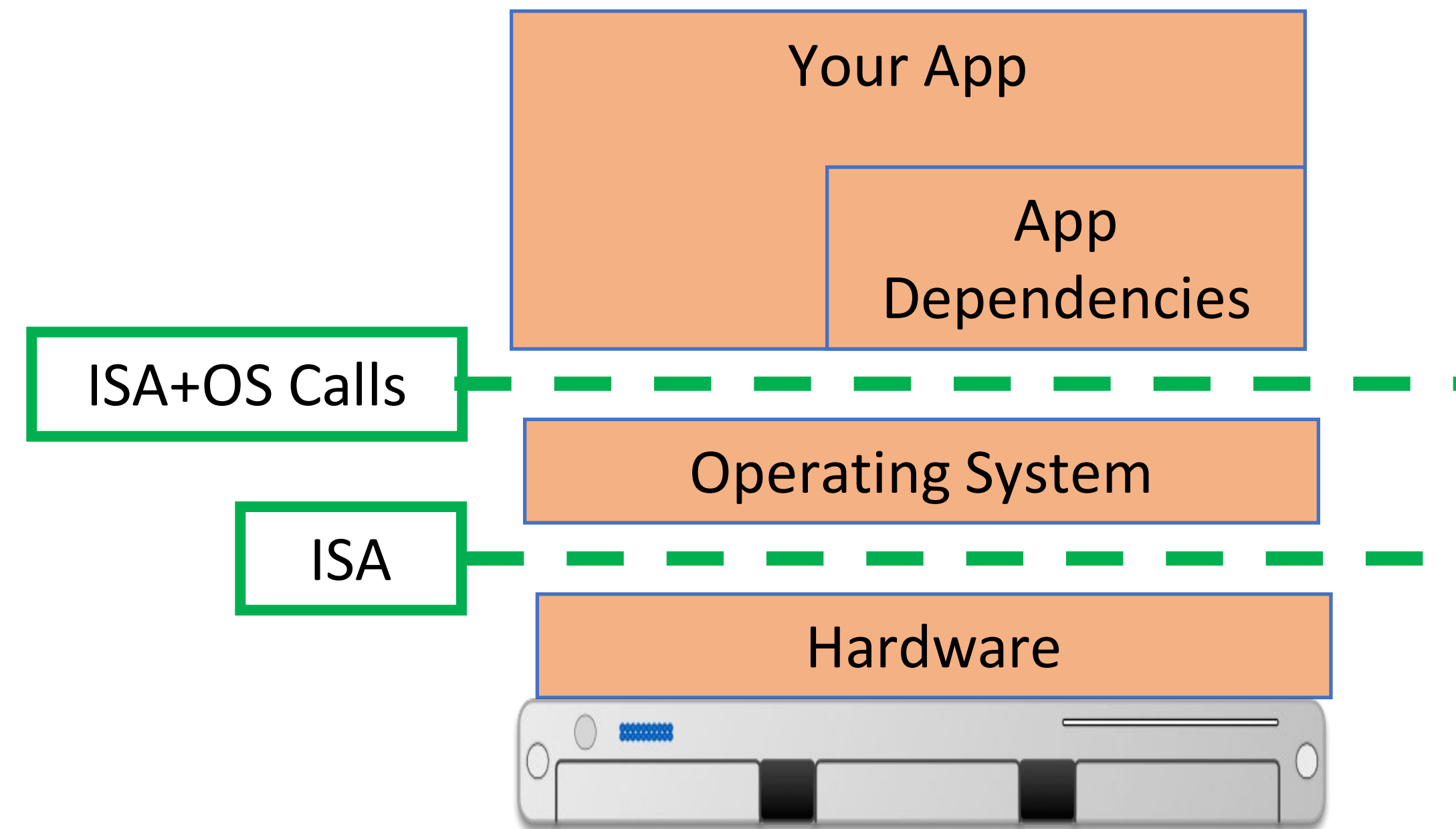
Infrastructure as a Service: Virtual Machines

- Virtual machines:
 - Virtualize a single large server into many smaller machines
 - OS limits resource usage and guarantees quality per-VM
 - Each VM in its own OS
 - Examples: Amazon EC2, Google Compute Engine, Azure

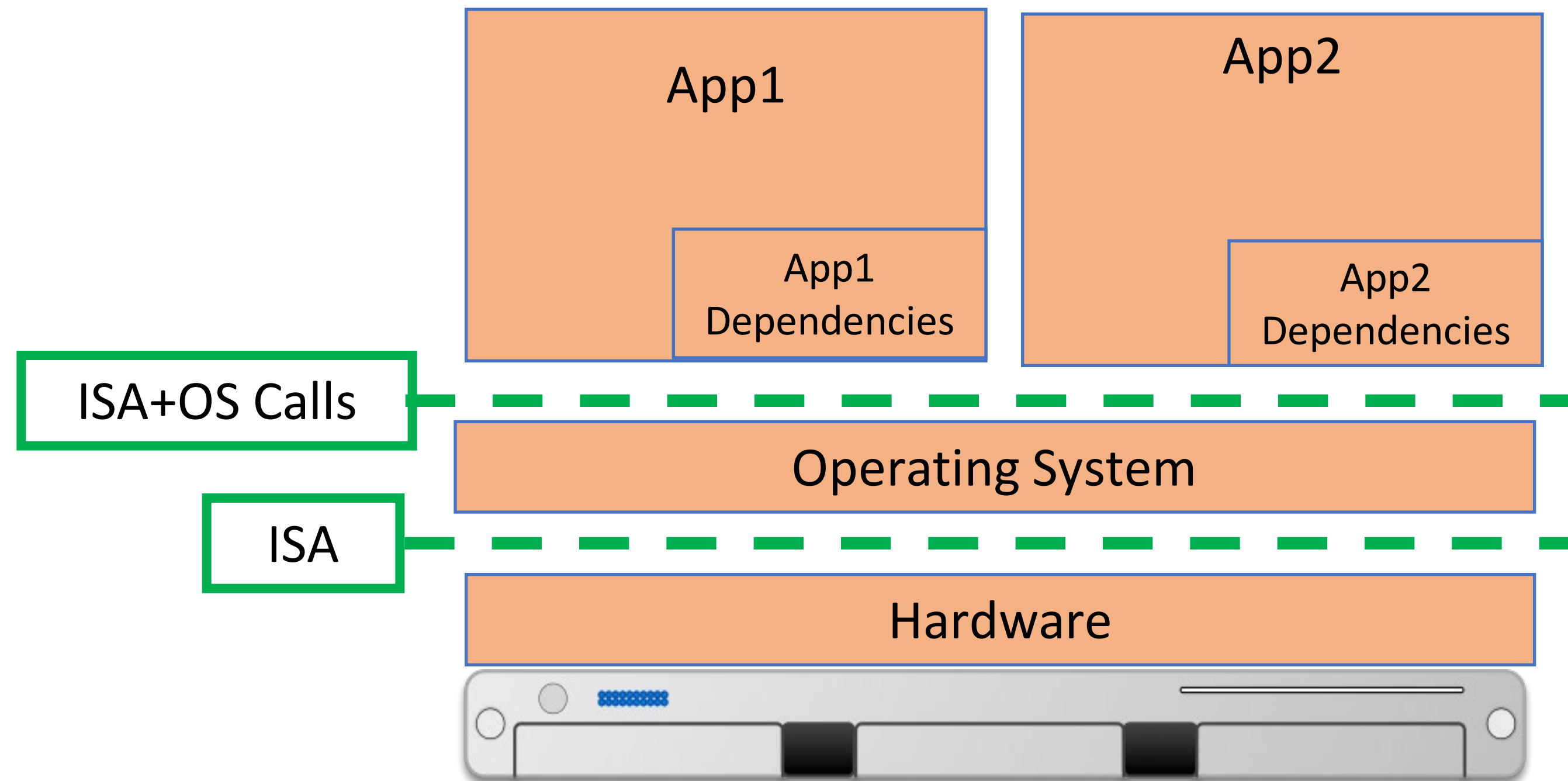


Let's look more closely at this software stack

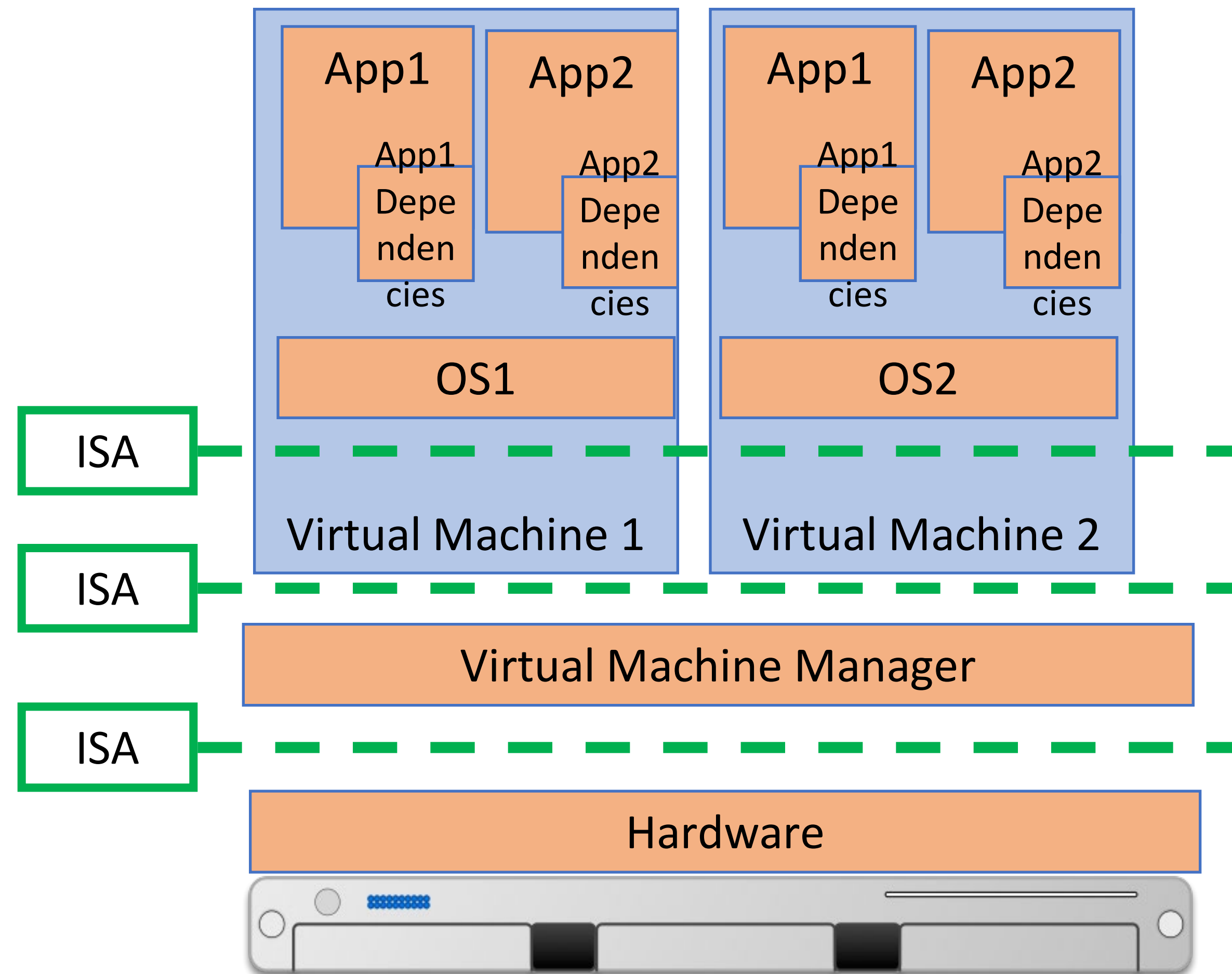
- The “instruction set” is an abstraction of the underlying hardware
- The operating system presents the same abstraction + OS calls.



The operating system allows several apps to share the underlying hardware



A virtual machine layer allows several different operating systems to share the same hardware



Virtual Machines facilitate multi-tenancy

- Multi-Tenancy
 - Multiple customers sharing same physical machine, oblivious to each other
- Decouples application from hardware
 - virtualization service can provide “live migration”
- Faster to provision and release
 - VM v. physical machines == ~mins v. ~hours

Virtual Machines to Containers

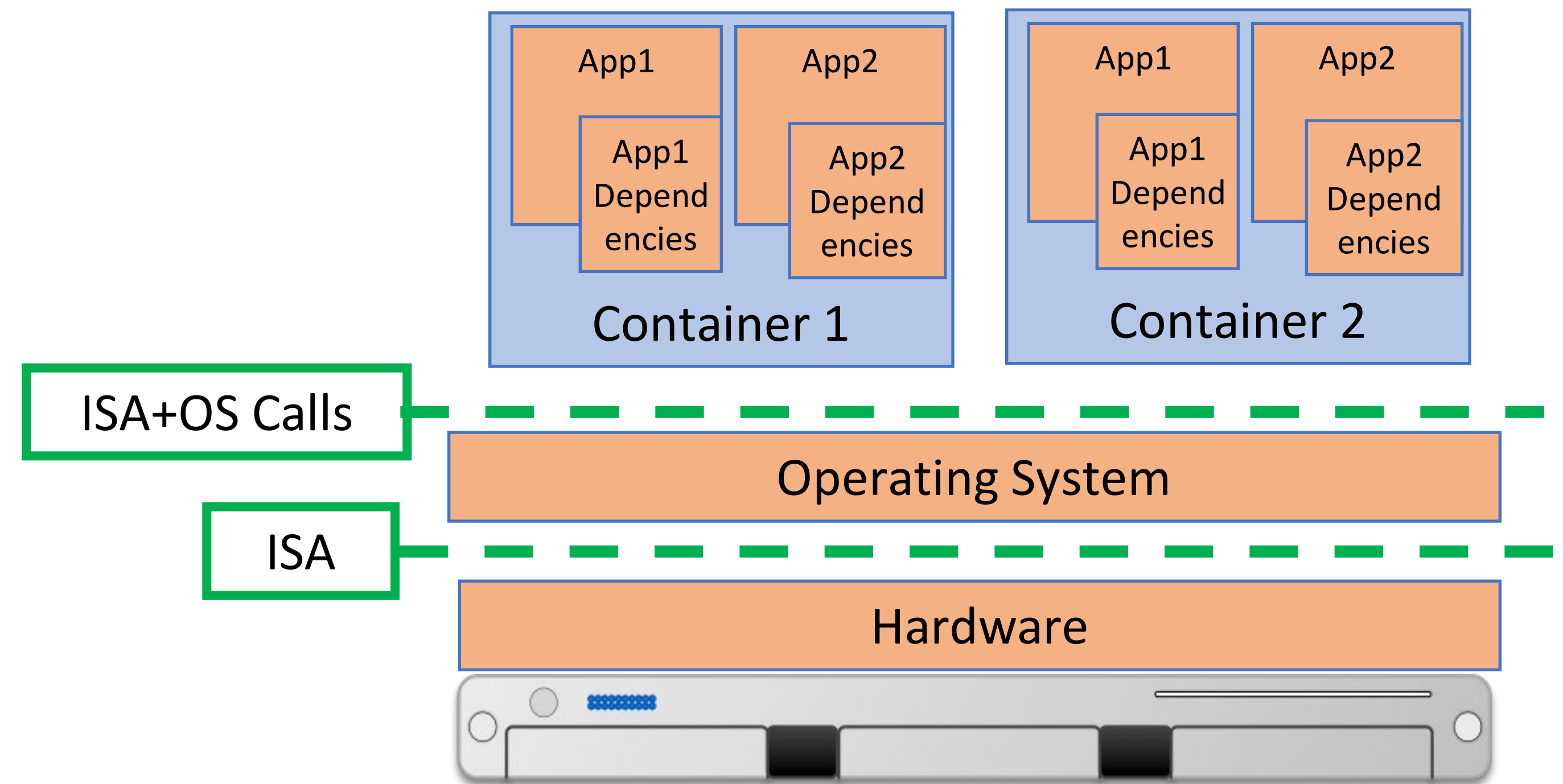
- Each VM contains a full operating system
- What if each application could run in the same (overall) operating system? Why have multiple copies?
- Advantages to smaller apps:
 - Faster to copy (and hence provision)
 - Consume less storage at rest

Infrastructure as a Service: Containers

- Each application is encapsulated in a “lightweight container,” includes:
 - System libraries (e.g. glibc)
 - External dependencies (e.g. nodejs)
- “Lightweight” in that container images are smaller than VM images - multi tenant containers run in the OS
- Cloud providers offer “containers as a service” (Amazon ECS Fargate, Azure Kubernetes, Google Kubernetes)

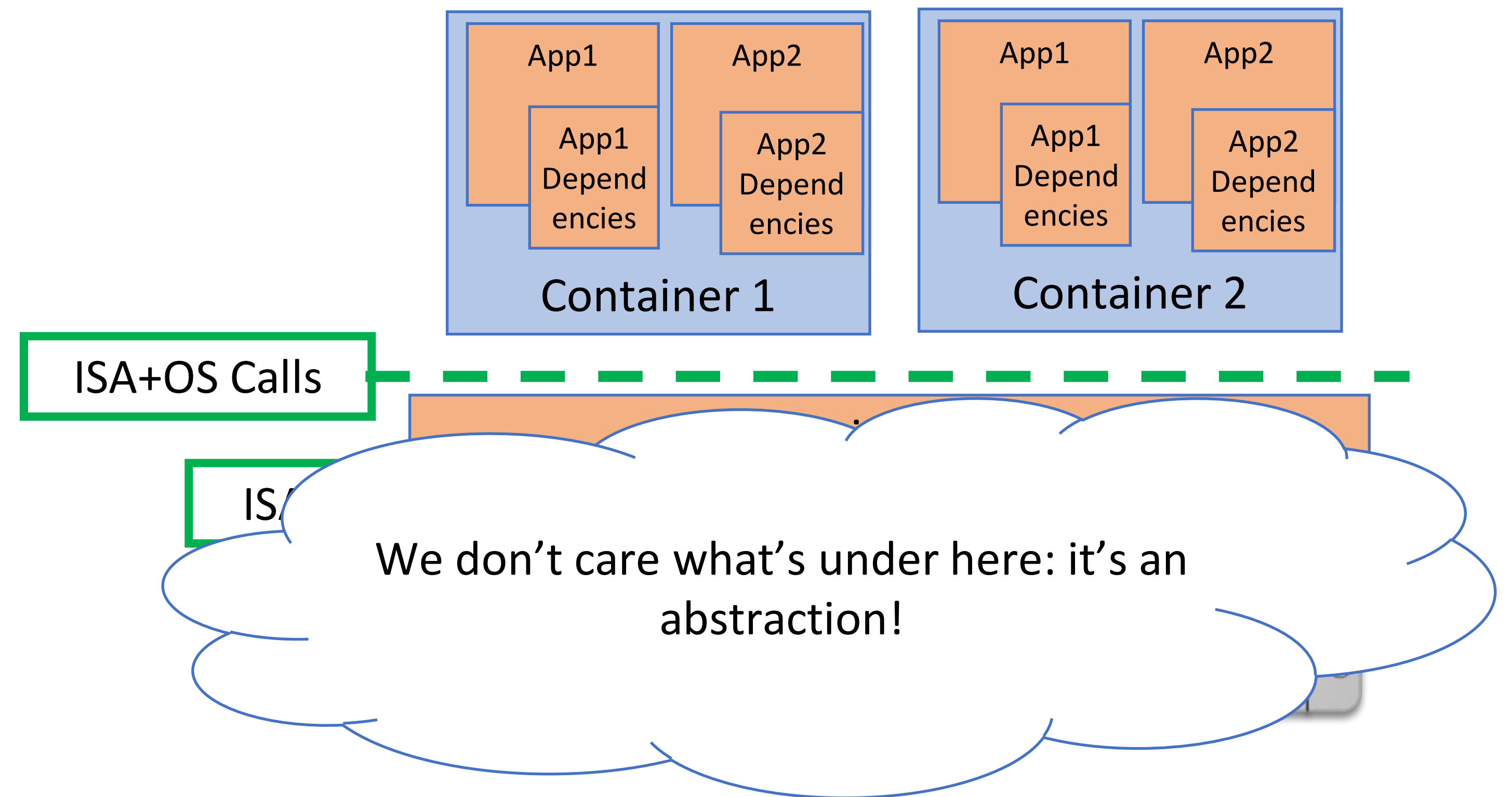
A container contains your apps and all their dependencies

- You might put several apps in a single container, together with their dependencies
- Might have only one copy of shared dependencies



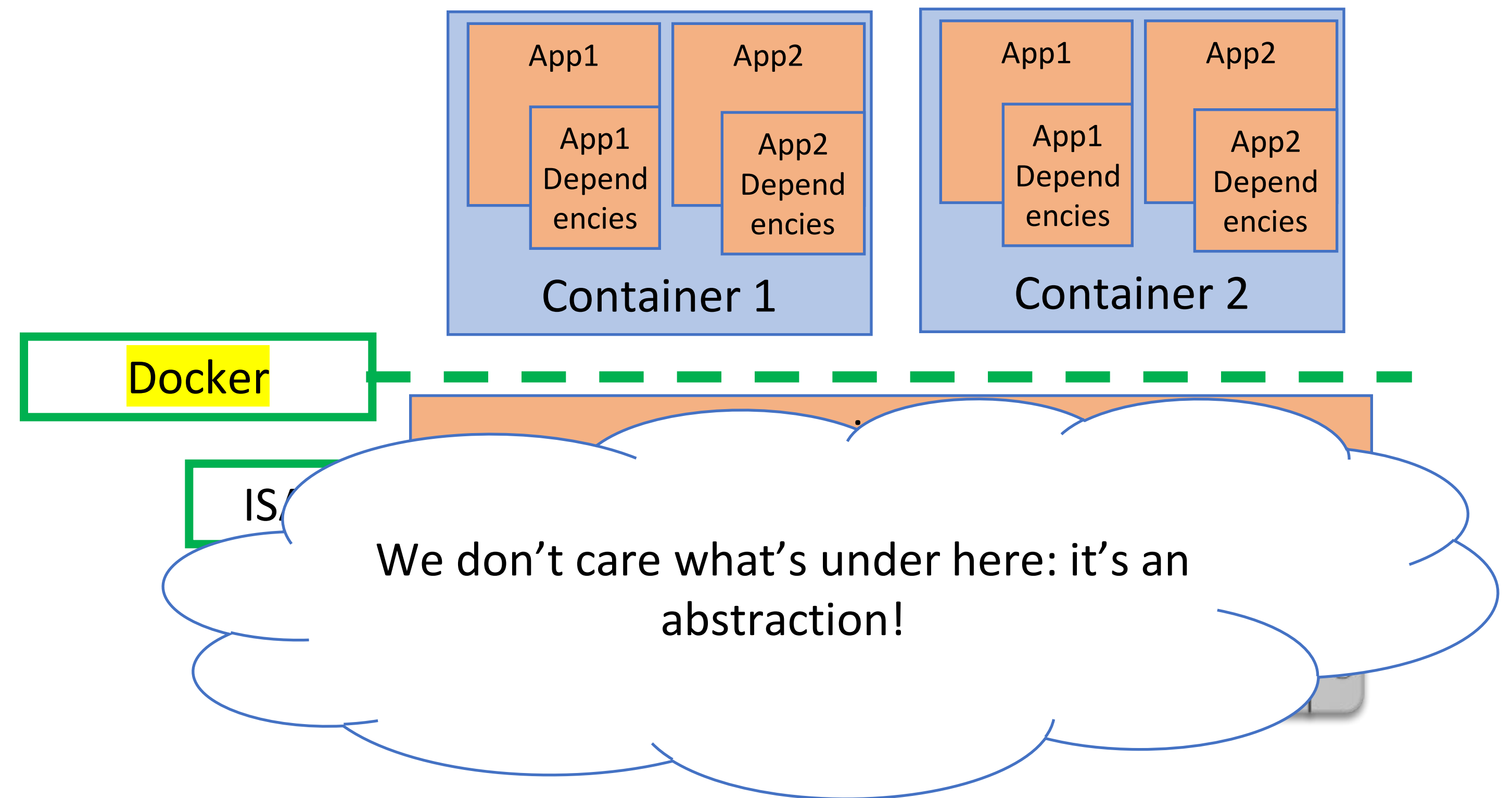
Infrastructure as a Service: with containers

- Vendor supplies an on-demand instance of an operating system
 - Eg: Linux version NN
- Vendor is free to implement that instance in a way that optimizes costs across many clients.



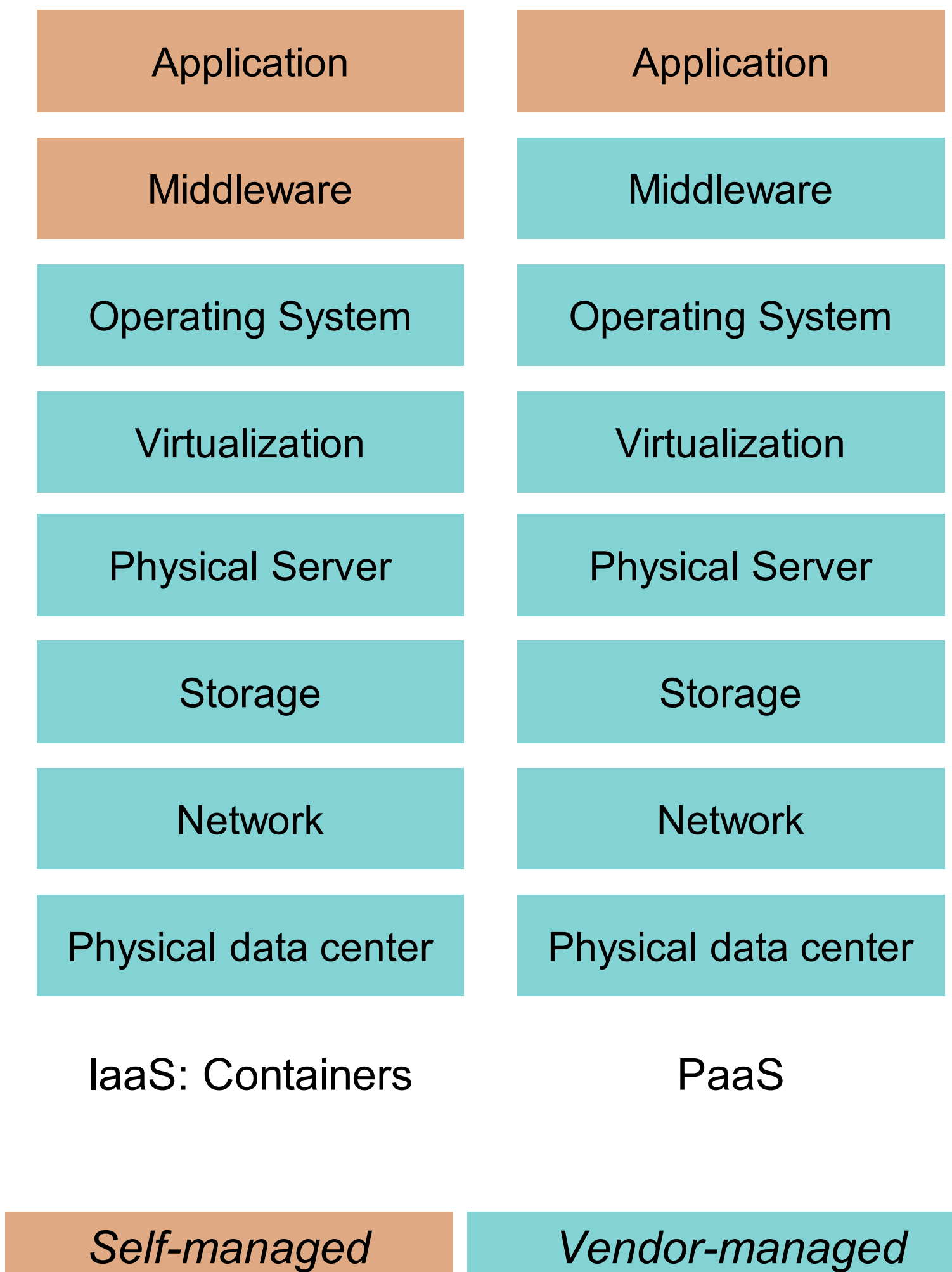
Infrastructure as a Service: Docker

- Docker provides a standardized interface for your container to use
- Many vendors will host your Docker container



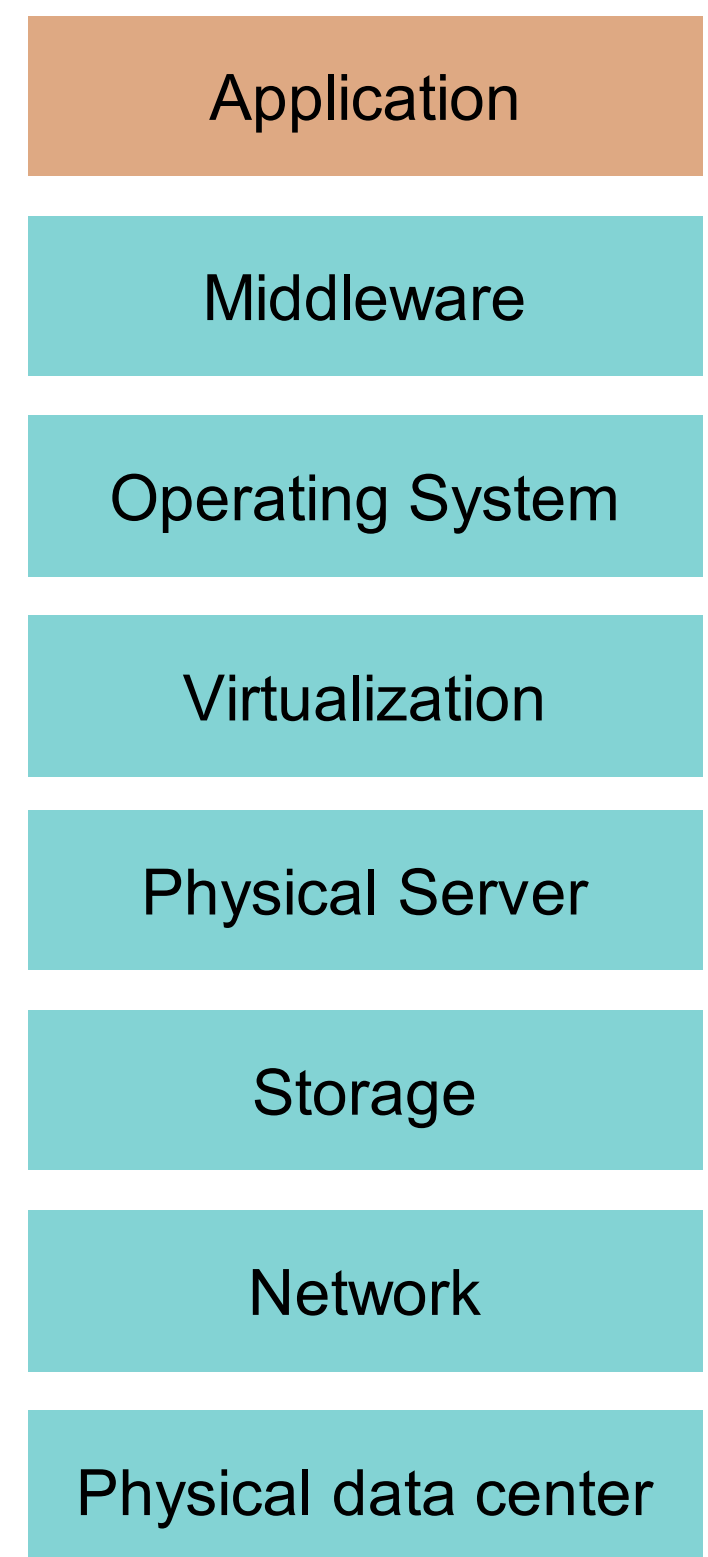
Platform-as-a-Service: vendor supplies OS + middleware

- Middleware is the stuff between our app and a user's requests:
 - Load balancer: route client requests to one of our app containers
 - Application server: run our handler functions in response to requests from load balancer
 - Monitoring/telemetry: log requests, response times and errors
- Cloud vendors provide managed middleware platforms too: "Platform as a Service"



PaaS is often the simplest choice for app deployment

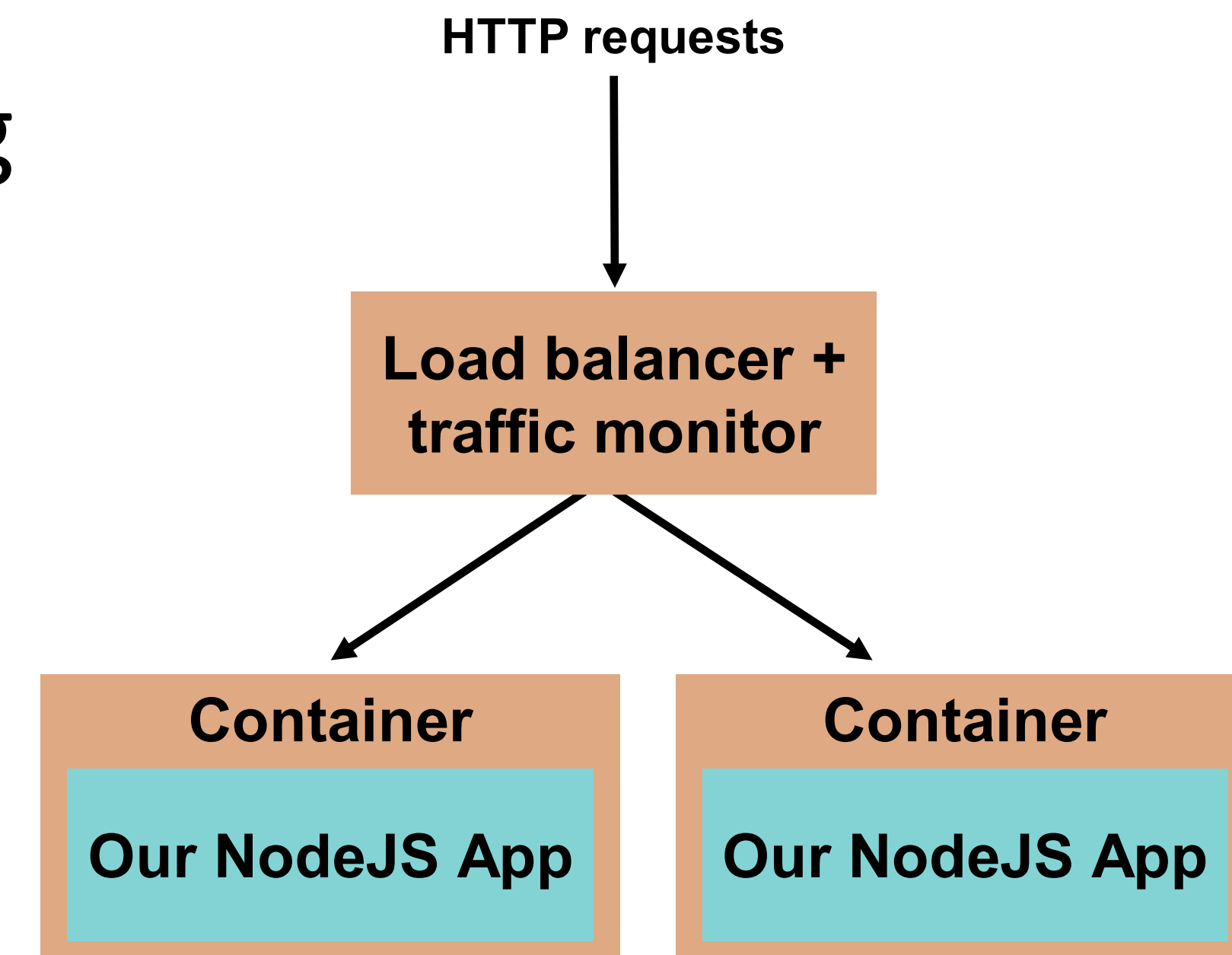
- **Platform-as-a-Service** provides components most apps need, fully managed by the vendor: load balancer, monitoring, application server
 - Heroku, AWS Elastic Beanstalk, Google App Engine
- Some PaaSs deploy apps as single functions invoked only when a web request is made
 - AWS Lambda, Google Cloud Functions, Azure Functions
- Some PaaSs provide databases and authentication
 - Google Firebase, Back4App



PaaS

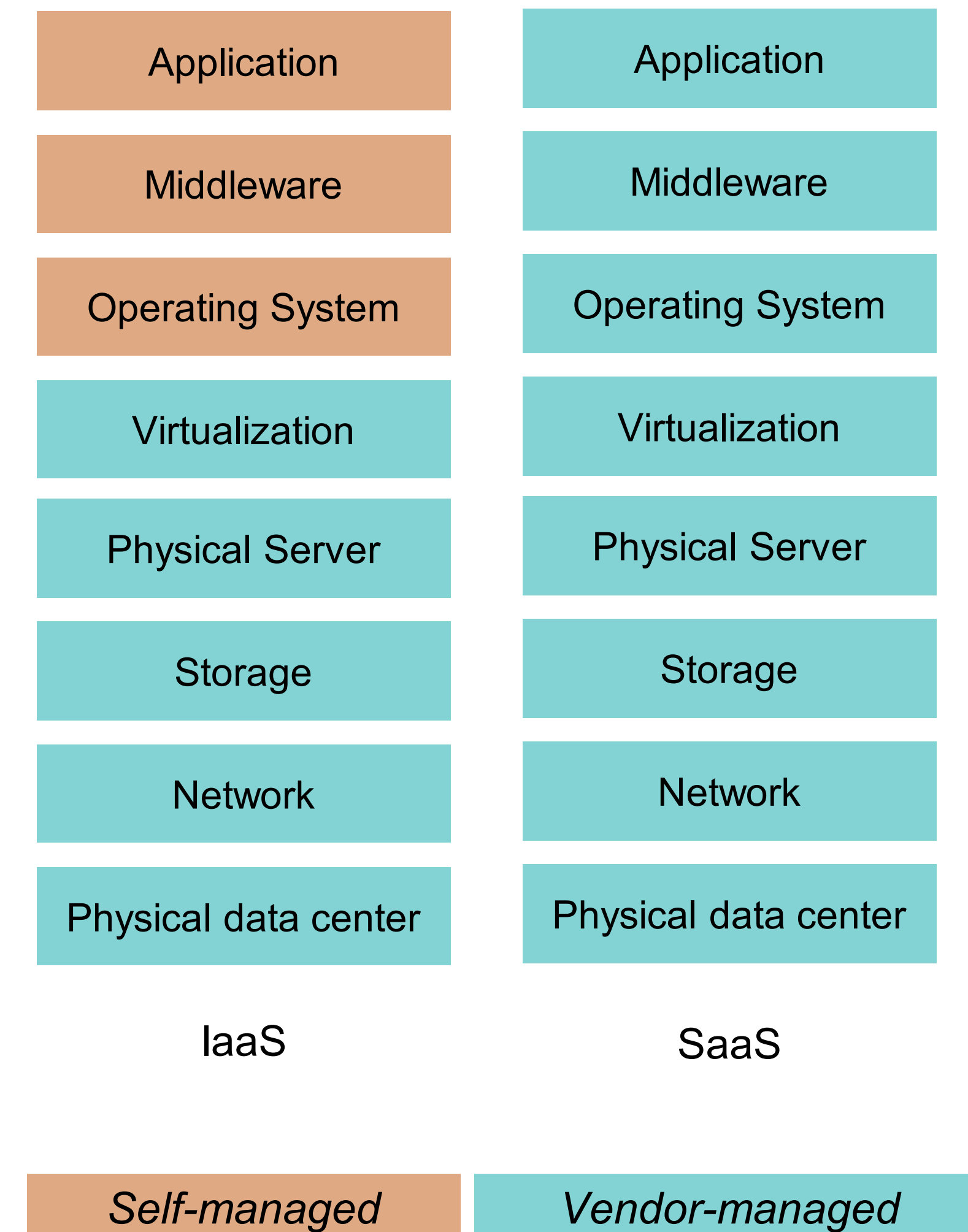
Heroku's PaaS

- Takes a web app as input
 - No container, only need entry point to code, e.g. “npm start”
- Hosts web app at chosen URL, can scale resources up/down on-demand
 - Load balancer fully managed by Heroku, scaling transparent
 - Auto-scale down to use no resources, spins up container on reception of a request
 - Dashboard for monitoring/reporting



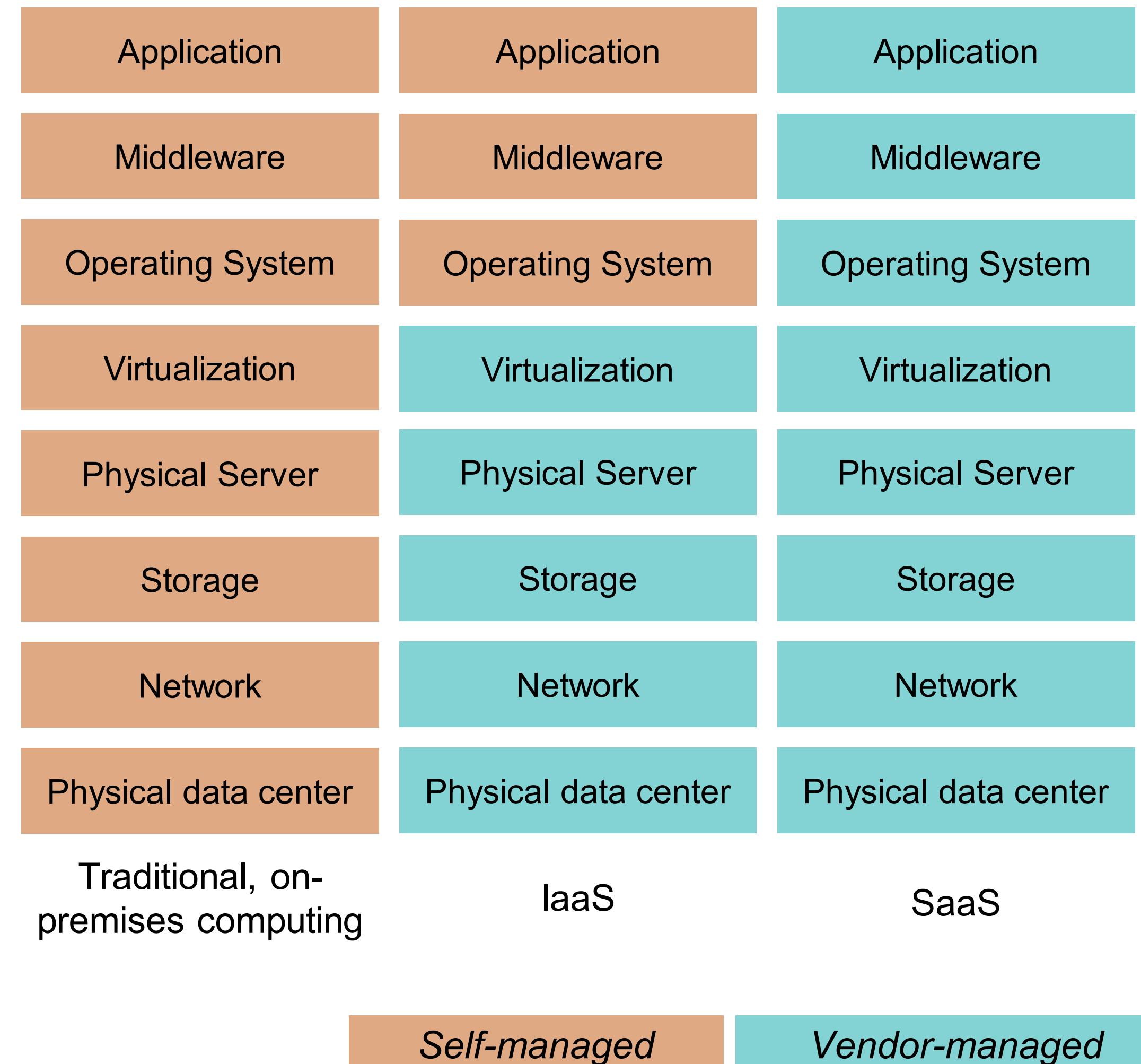
Software as a Service adds more vendor-managed apps

- Providers may also develop custom software offered only as a service
- Examples:
 - PostgreSQL (open source)
 - Twilio Programmable Video (proprietary chat)



Self-managed vs Vendor-managed Infrastructure

- Benefits to vendor-managed options:
 - More ways to reduce resource consumption, improve resource utilization
 - Less management burden
 - Less capital investment, greater operating expenses
- Benefits to self-managed options:
 - Greater flexibility and avoid vendor lock-in
 - More capital investment, less operating expenses



Cloud Infrastructure is best for variable workloads

- Consider:
 - Does your workload benefit from ability to scale up or down?
- Example:
 - need to run 300 VMs, each 4 vCPUs, 16GB RAM
- Private cloud:
 - Dell PowerEdge Pricing (AMD EPYC 64 core CPUs)
 - 7 servers, each 128 cores, 512GB RAM, 3 TB storage = \$162,104
- Public cloud:
 - Amazon EC2 Pricing (M5.xlarge instances, \$0.121/VM-hour)
 - 10 VMs for 1 year + 290 VMs for 1 month: \$36,215.30
 - 300 VMs for 1 year: \$317,988

Public clouds are not the only option

- “Public” clouds are connected to the internet and available for anyone to use
 - Examples: Amazon, Azure, Google Cloud, DigitalOcean
- “Private” clouds use cloud technologies with on-premises, self-managed hardware
 - Cost-effective when a large scale of baseline resources are needed
 - Example management software: OpenStack, VMWare, Proxmox, Kubernetes
- “Hybrid” clouds integrate private and public (or multiple public) clouds
 - Effective approach to “burst” capacity from private cloud to public cloud

Review

- You should now be able to...
 - Explain what “cloud” computing is and why it is important
 - Describe the difference between virtual machines and containers
 - Explain why virtual machines and containers are important in cloud computing