CS 4530 Software Engineering

Module 13: Principles and Patterns of Cloud Infrastructure

Khoury College of Computer Sciences

Learning objectives for this lesson

By the end of this lesson, you should be able to…

- Describe what “cloud” computing is
- Understand the role of virtual machines and containers in cloud computing
- Deploy a web app to the cloud

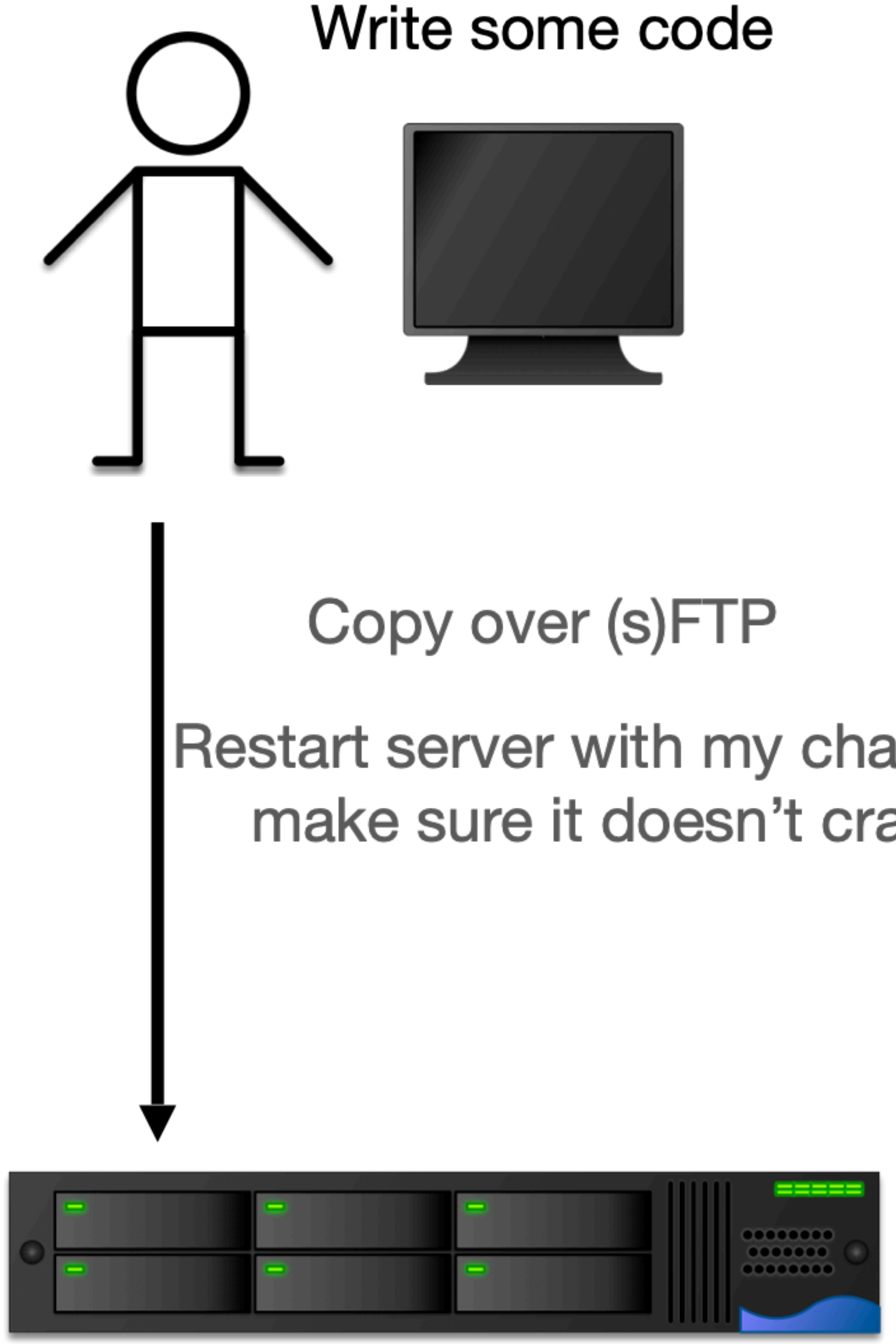
How to deploy web apps?

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?



Write some code

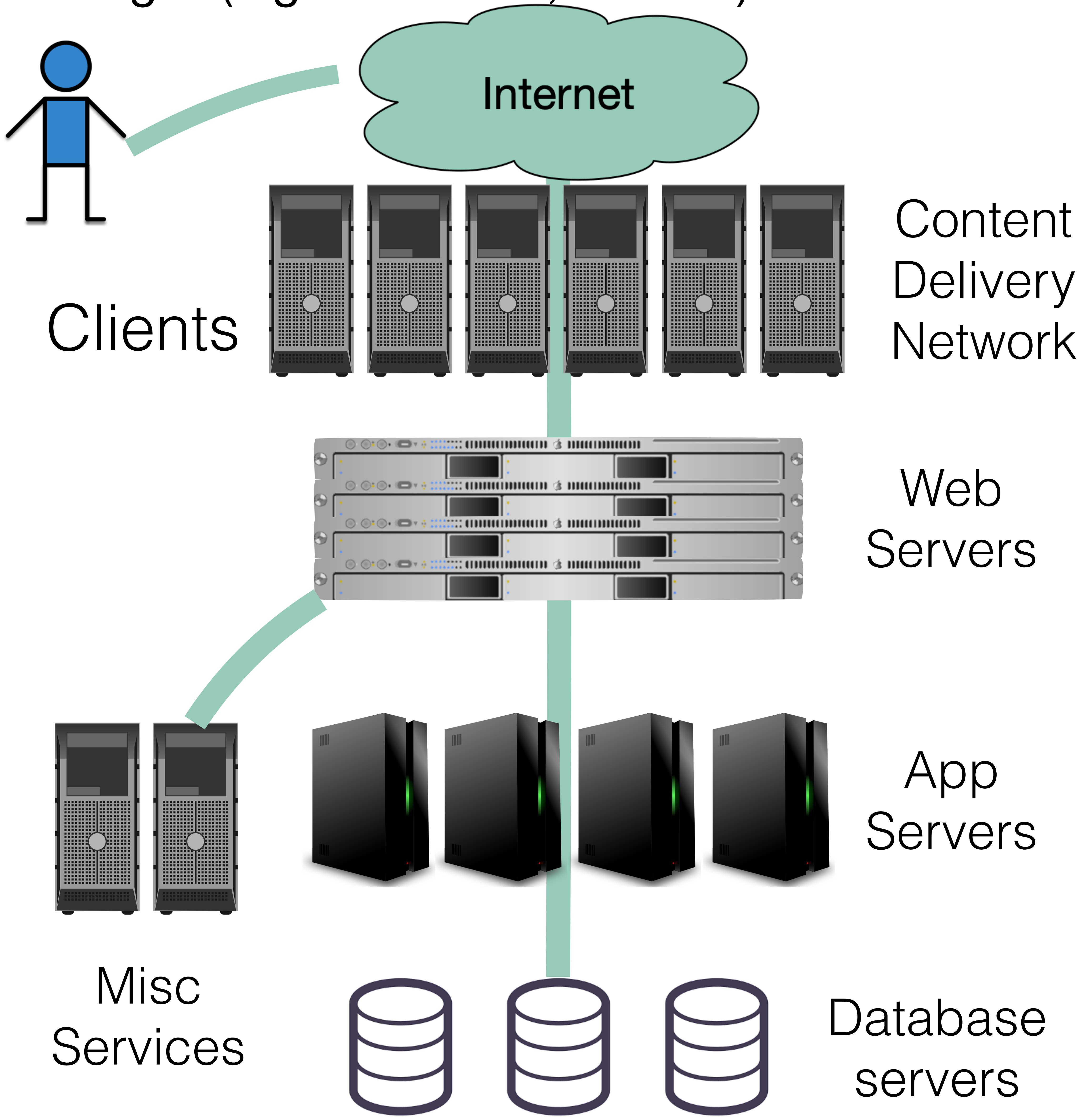
Copy over (s)FTP

Restart server with my changes,
make sure it doesn’t crash

Class Server, in CS Department
Data Center

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



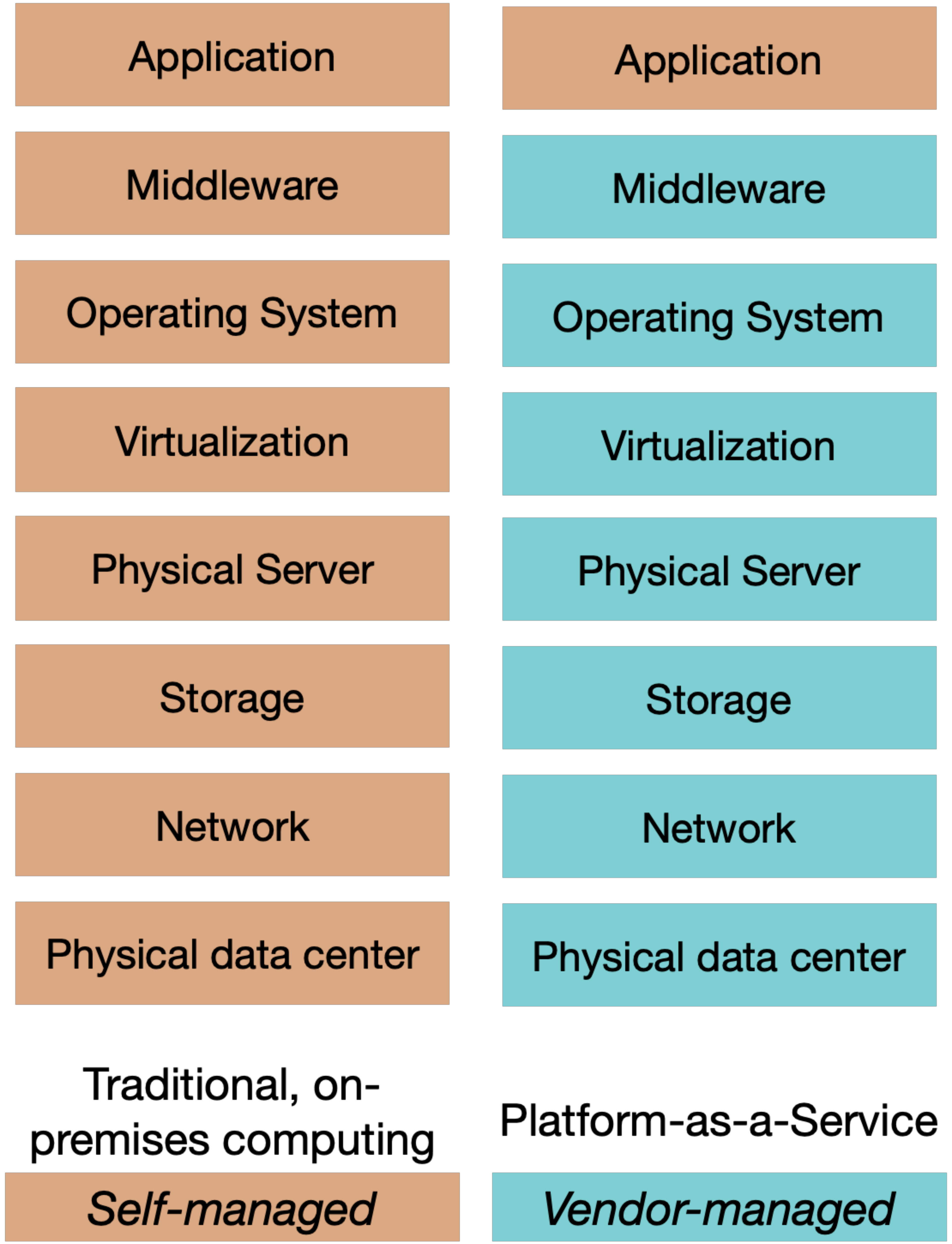
What is a cloud infrastructure?

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”



Traditional, on-premises computing	Platform-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

🟧 *Self-managed*

🟦 *Vendor-managed*

Cloud infrastructure creates economies of scale

At the physical level:

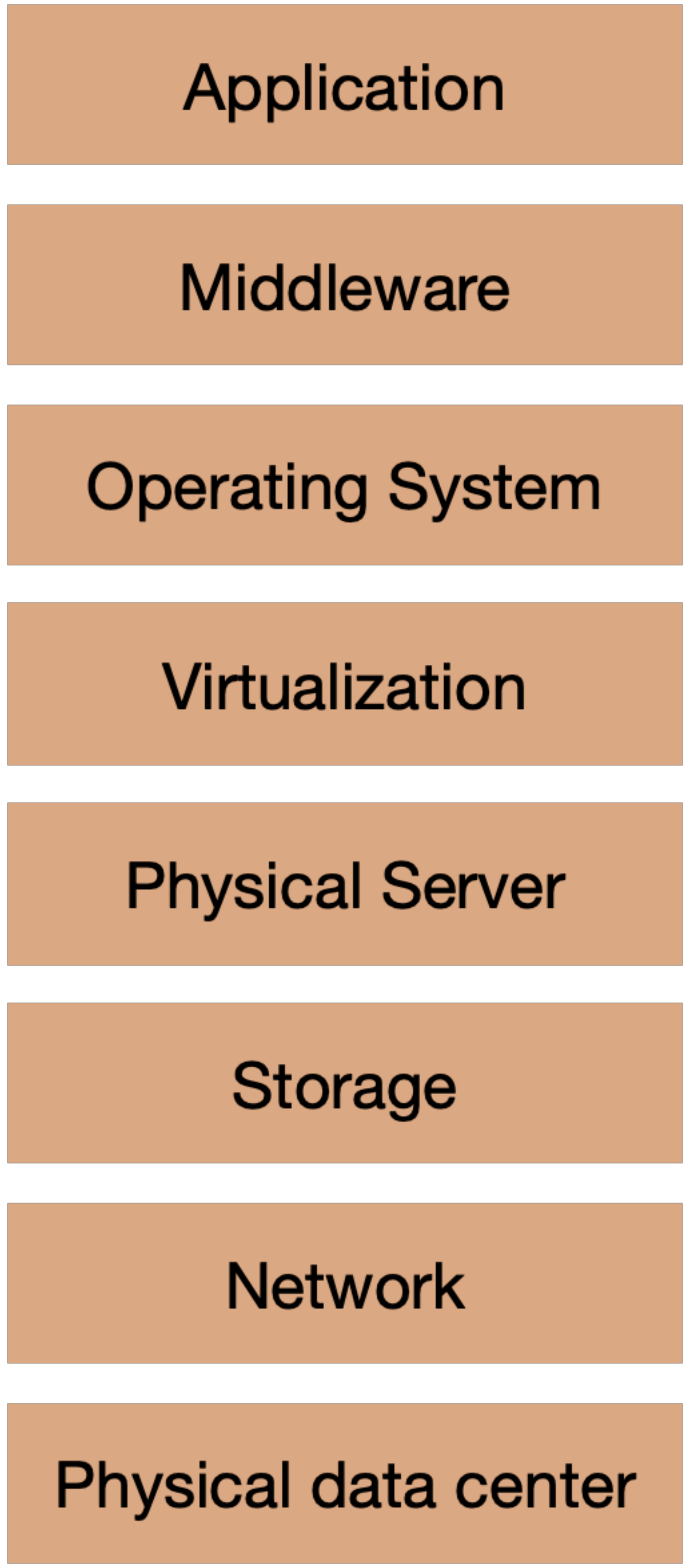
- Multiple customers’ physical machines in the same data center
- Save on physical costs (centralize power, cooling, security, maintenance)

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

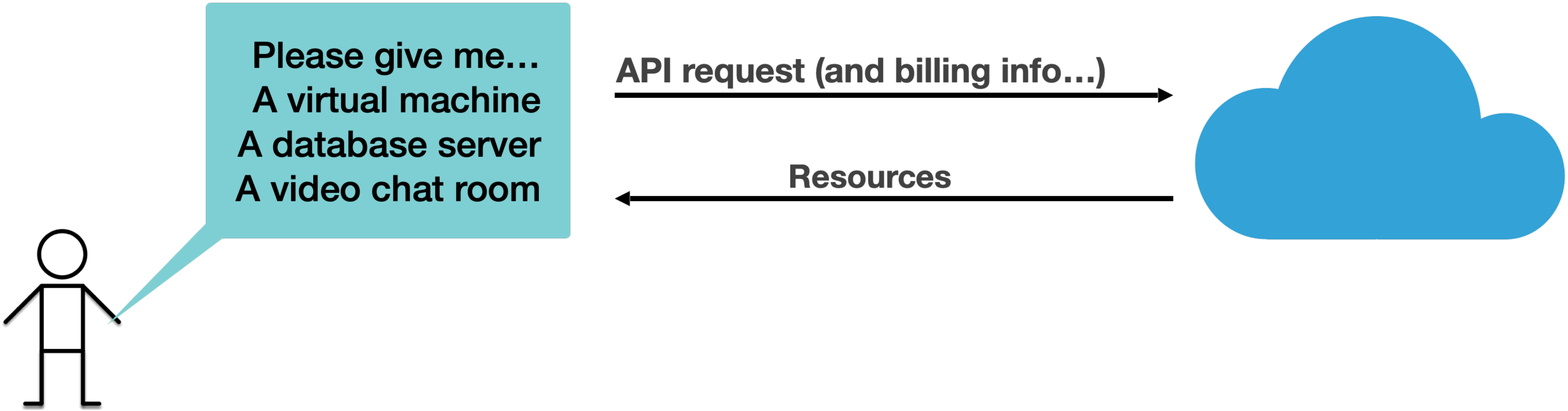
Cloud computing scales elastically:

- “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 Infra is on-demand access to resources

Vendor provides a service catalog of “X as a service” abstractions

API allows us to provision resources on-demand

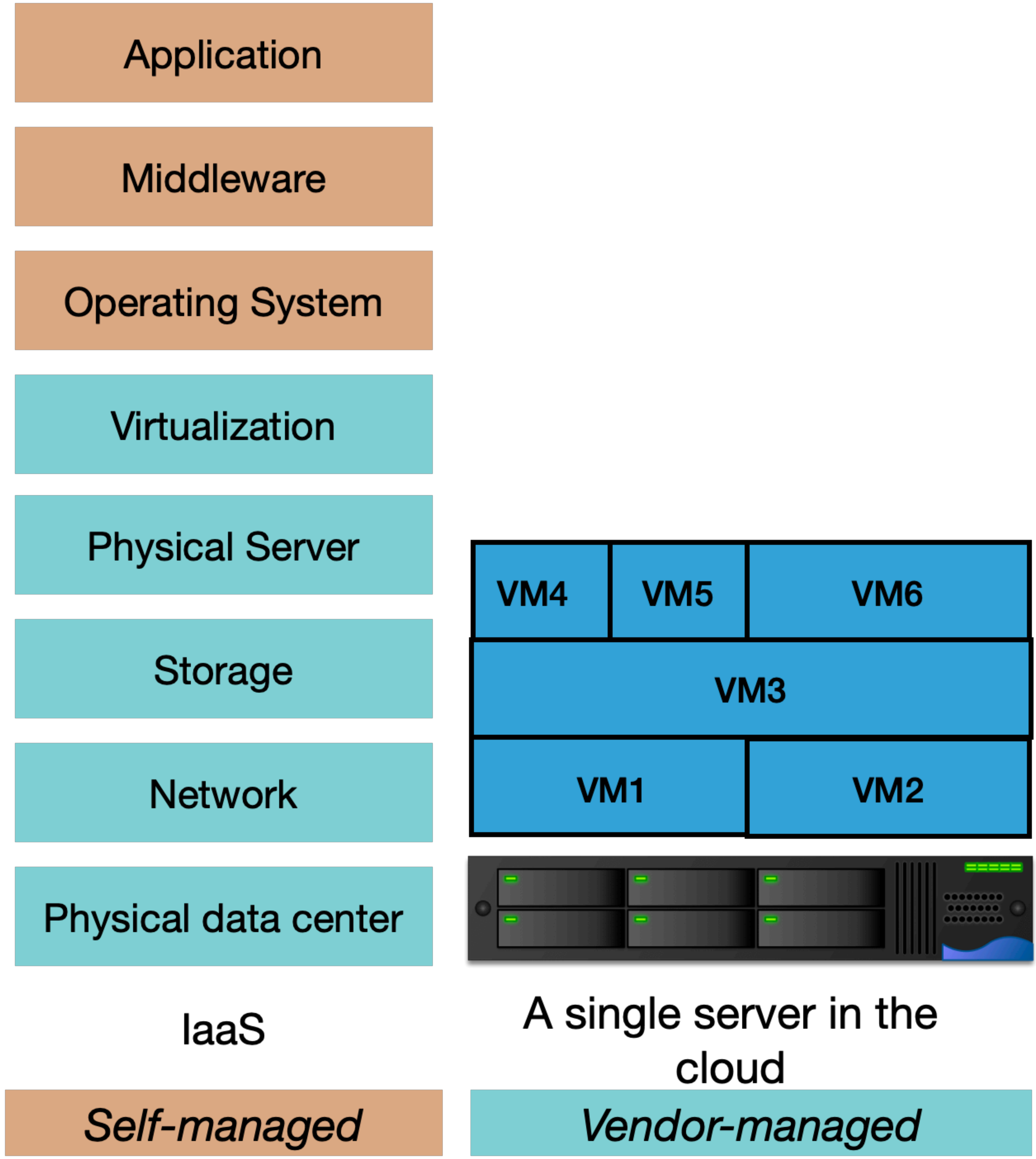


Please give me…
A virtual machine
A database server
A video chat room

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



Application

Middleware

Operating System

Virtualization

Physical Server

Storage

Network

Physical data center

IaaS

VM4 VM5 VM6

VM3

VM1 VM2

A single server in the cloud

Self-managed

Vendor-managed

Virtual Machines are a core abstraction

Multi-Tenancy

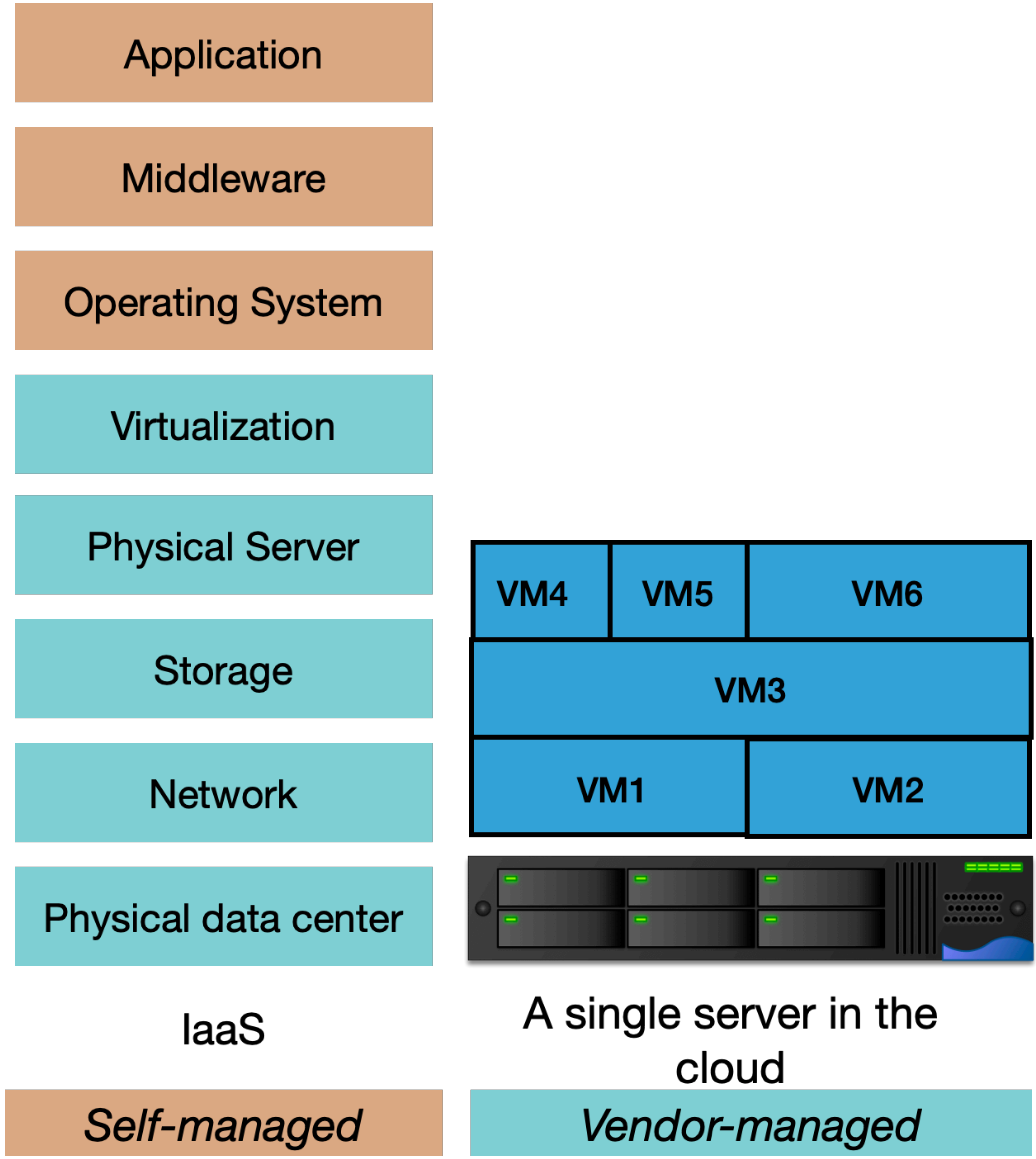
- Multiple customers sharing same physical machine, oblivious to each other

Decouples application from hardware

- virtualization service can provide “live migration”

Faster to provision and release

- VM v. physical machines == ~mins v. ~hours



Application

Middleware

Operating System

Virtualization

Physical Server

Storage

Network

Physical data center

IaaS

VM4 VM5 VM6

VM3

VM1 VM2

A single server in the cloud

Self-managed

Vendor-managed

Virtual Machines to Containers

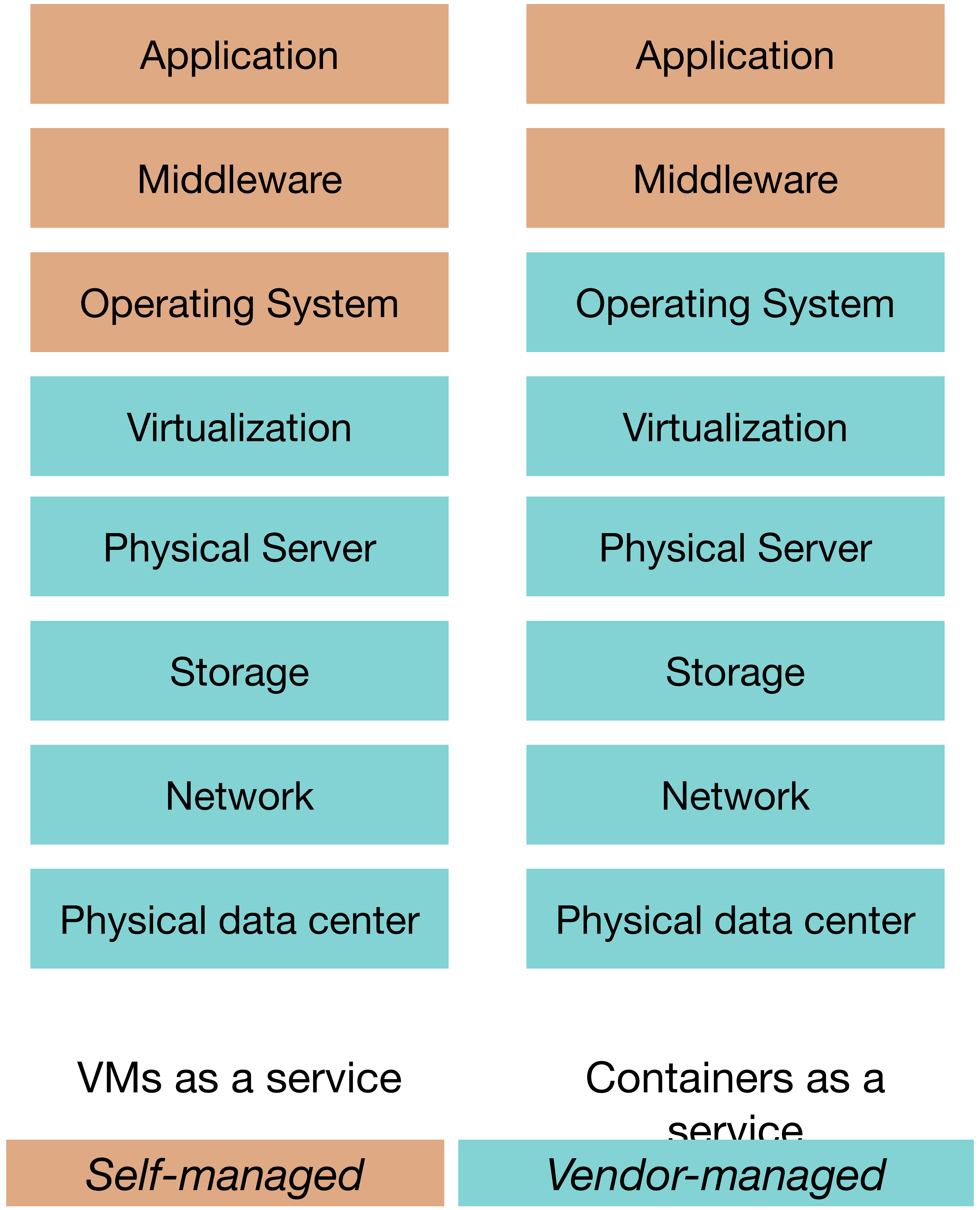
Each VM contains a full operating system

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



VMs as a service	Containers 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

🟧 *Self-managed* 🟦 *Vendor-managed*

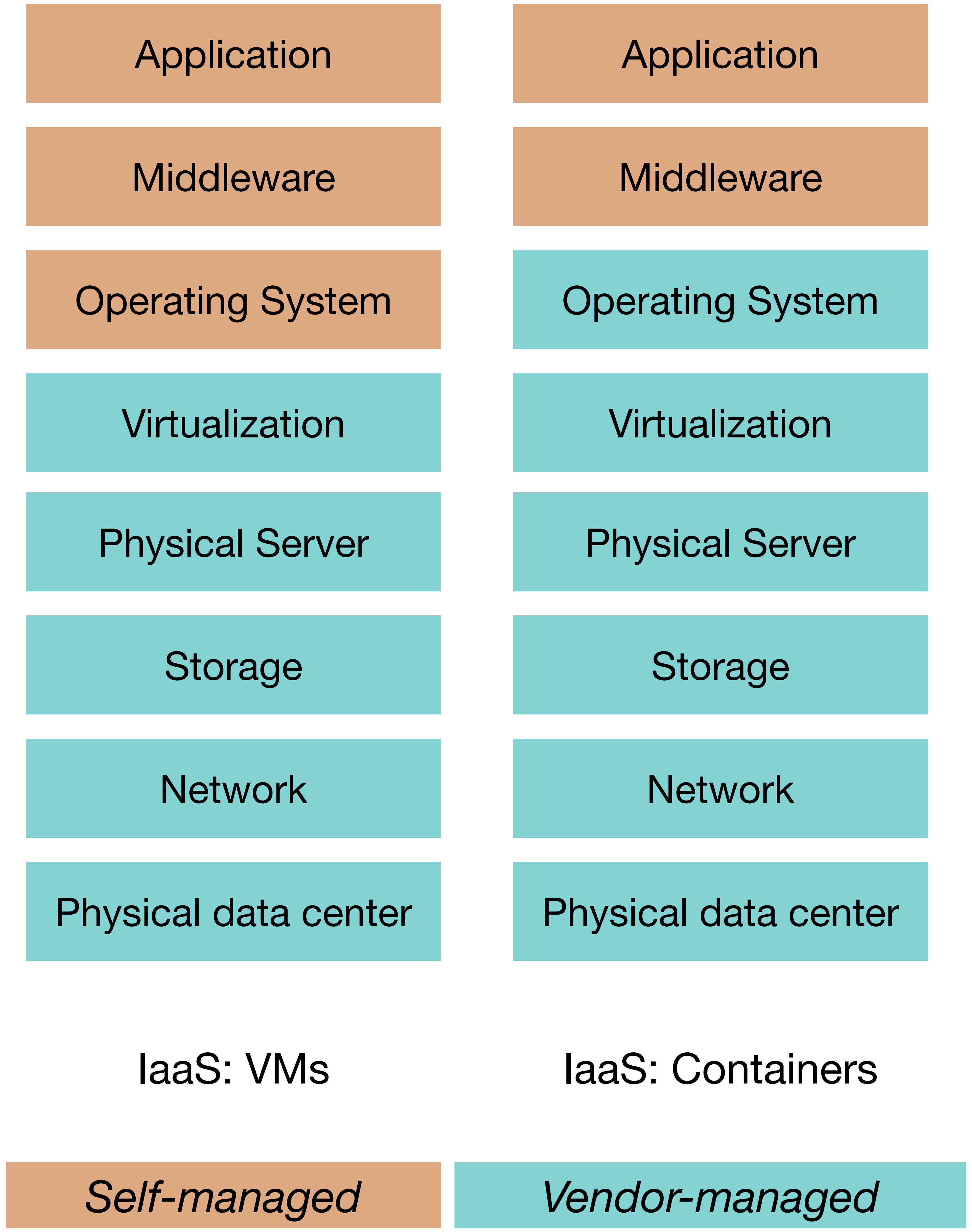
Infrastructure as a Service: Containers

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)



IaaS: VMs	IaaS: Containers
Application	Application
Middleware	Middleware
Operating System	Operating System
Virtualization	Virtualization
Physical Server	Physical Server
Storage	Storage
Network	Network
Physical data center	Physical data center

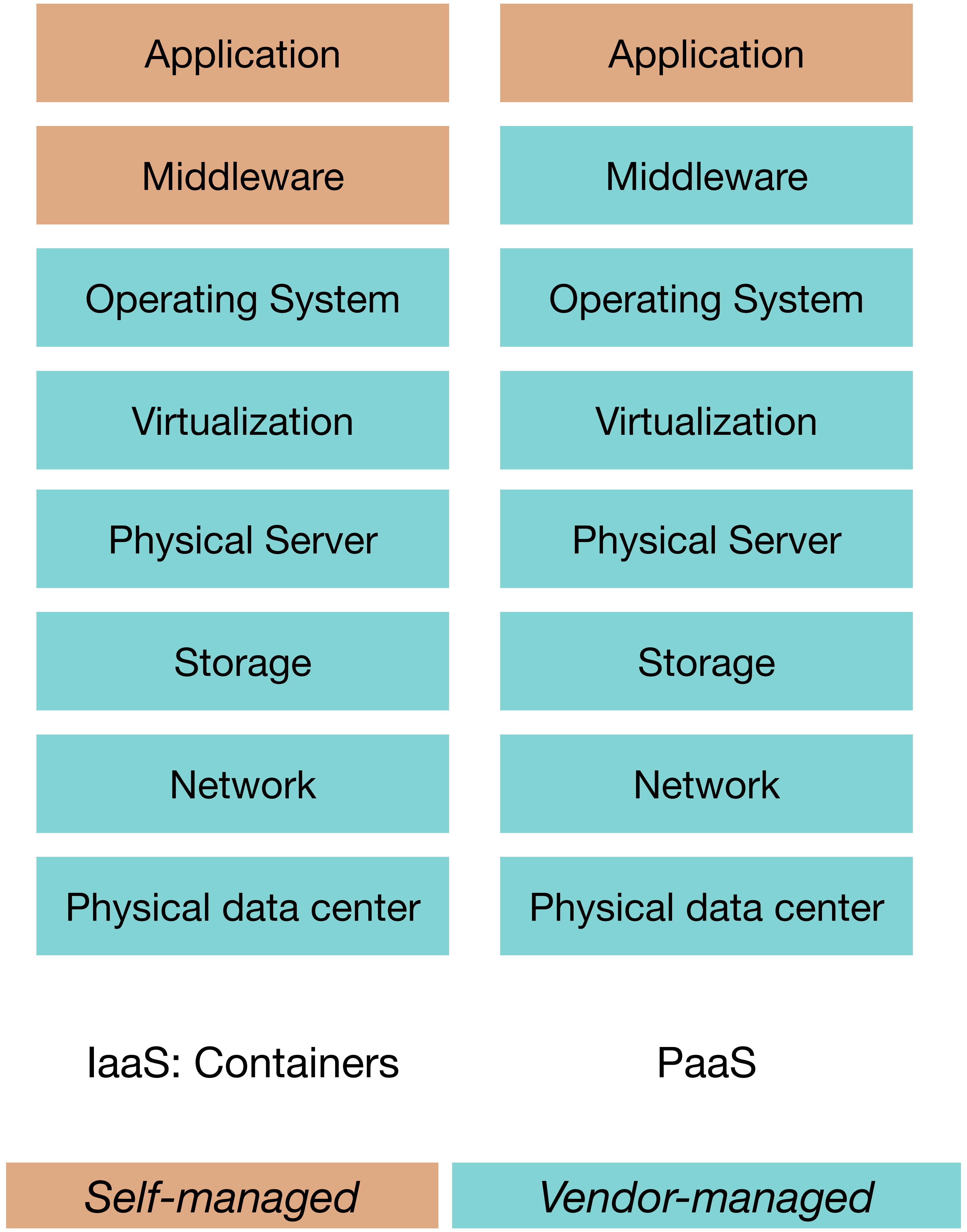
Self-managed *Vendor-managed*

Many apps rely on common 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"



IaaS: Containers	PaaS
🟧 Application	🟧 Application
🟧 Middleware	🟦 Middleware
🟦 Operating System	🟦 Operating System
🟦 Virtualization	🟦 Virtualization
🟦 Physical Server	🟦 Physical Server
🟦 Storage	🟦 Storage
🟦 Network	🟦 Network
🟦 Physical data center	🟦 Physical data center

🟧 *Self-managed* 🟦 *Vendor-managed*

PaaS, simplest choice for app deployment

Platform-as-a-Service provides components most apps need, fully managed by the vendor: load balancer, monitoring, application server

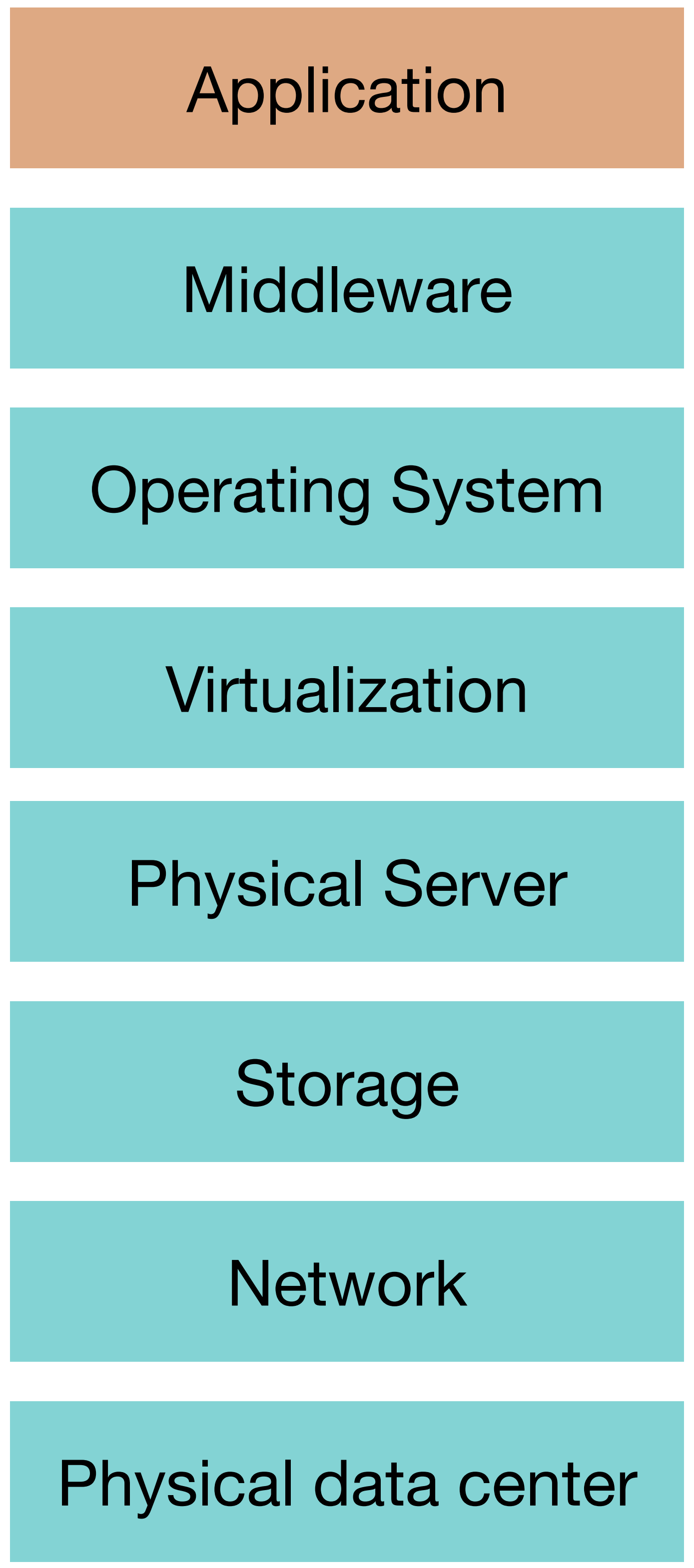
- Heroku, AWS Elastic Beanstalk, Google App Engine

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



Application

Middleware

Operating System

Virtualization

Physical Server

Storage

Network

Physical data center

PaaS

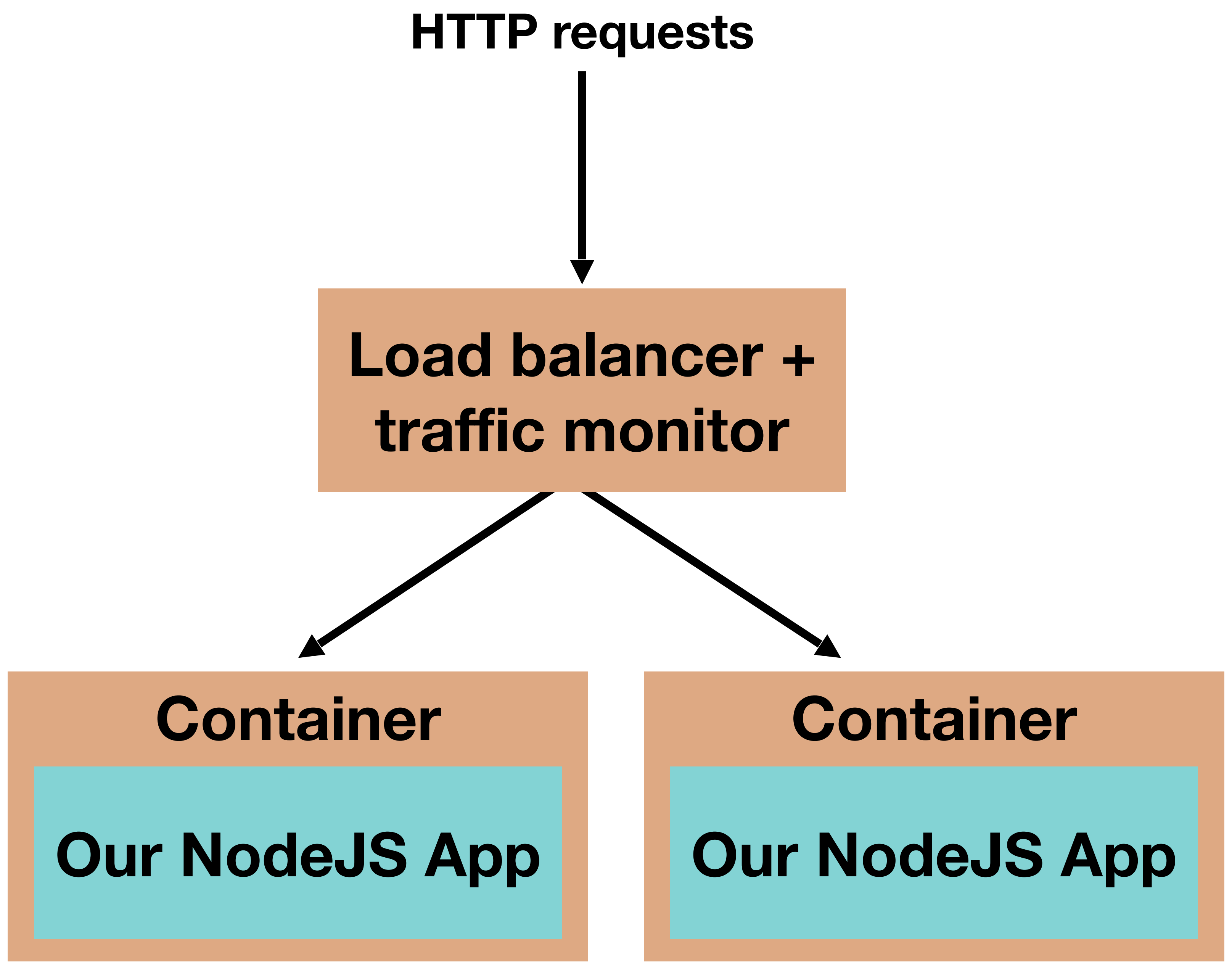
Heroku’s PaaS

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



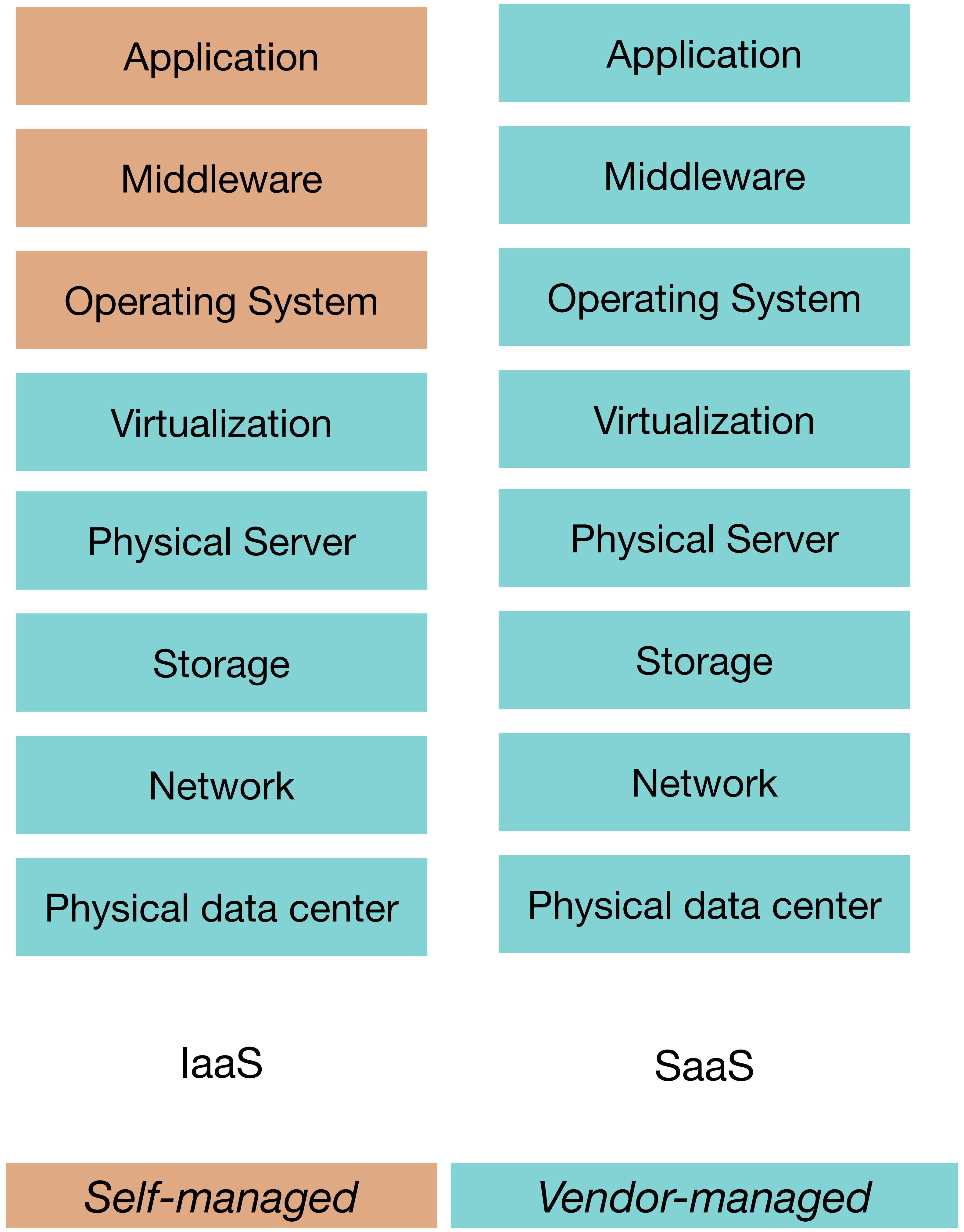
Software as a Service is fully managed

Many apps require same software, cloud providers can operate it for us

Providers also develop custom software offered only as a service

Examples:

- PostgreSQL (open source)
- Twilio Programmable Video (proprietary chat)



IaaS	SaaS
🟧 Application	🟦 Application
🟧 Middleware	🟦 Middleware
🟧 Operating System	🟦 Operating System
🟦 Virtualization	🟦 Virtualization
🟦 Physical Server	🟦 Physical Server
🟦 Storage	🟦 Storage
🟦 Network	🟦 Network
🟦 Physical data center	🟦 Physical data center

🟧 *Self-managed* 🟦 *Vendor-managed*

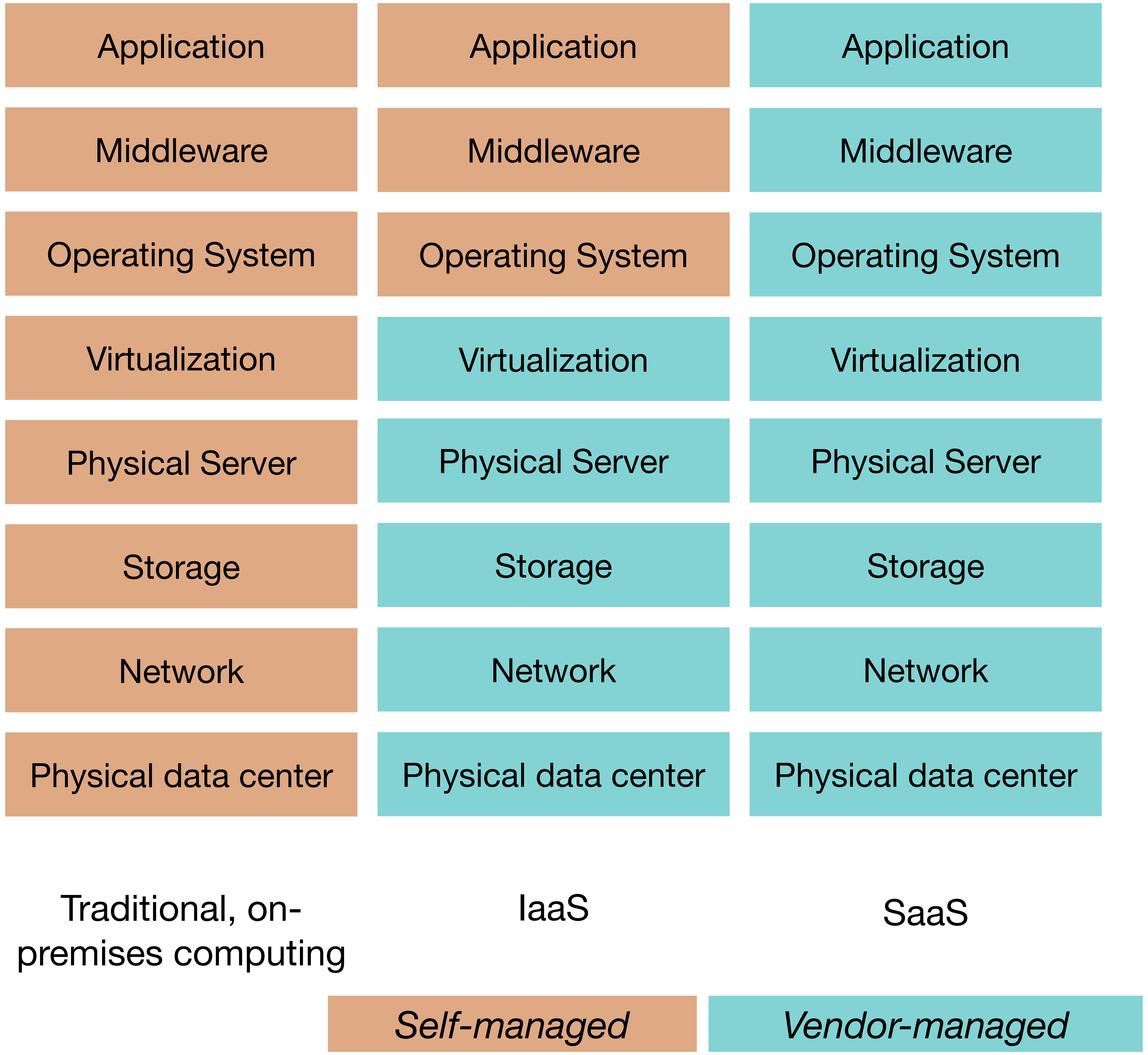
Self-managed vs Vendor-managed Infrastructure

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



Traditional, on-premises computing	IaaS	SaaS
🟧 Application	🟧 Application	🟦 Application
🟧 Middleware	🟧 Middleware	🟦 Middleware
🟧 Operating System	🟧 Operating System	🟦 Operating System
🟧 Virtualization	🟦 Virtualization	🟦 Virtualization
🟧 Physical Server	🟦 Physical Server	🟦 Physical Server
🟧 Storage	🟦 Storage	🟦 Storage
🟧 Network	🟦 Network	🟦 Network
🟧 Physical data center	🟦 Physical data center	🟦 Physical data center

🟧 *Self-managed* 🟦 *Vendor-managed*

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

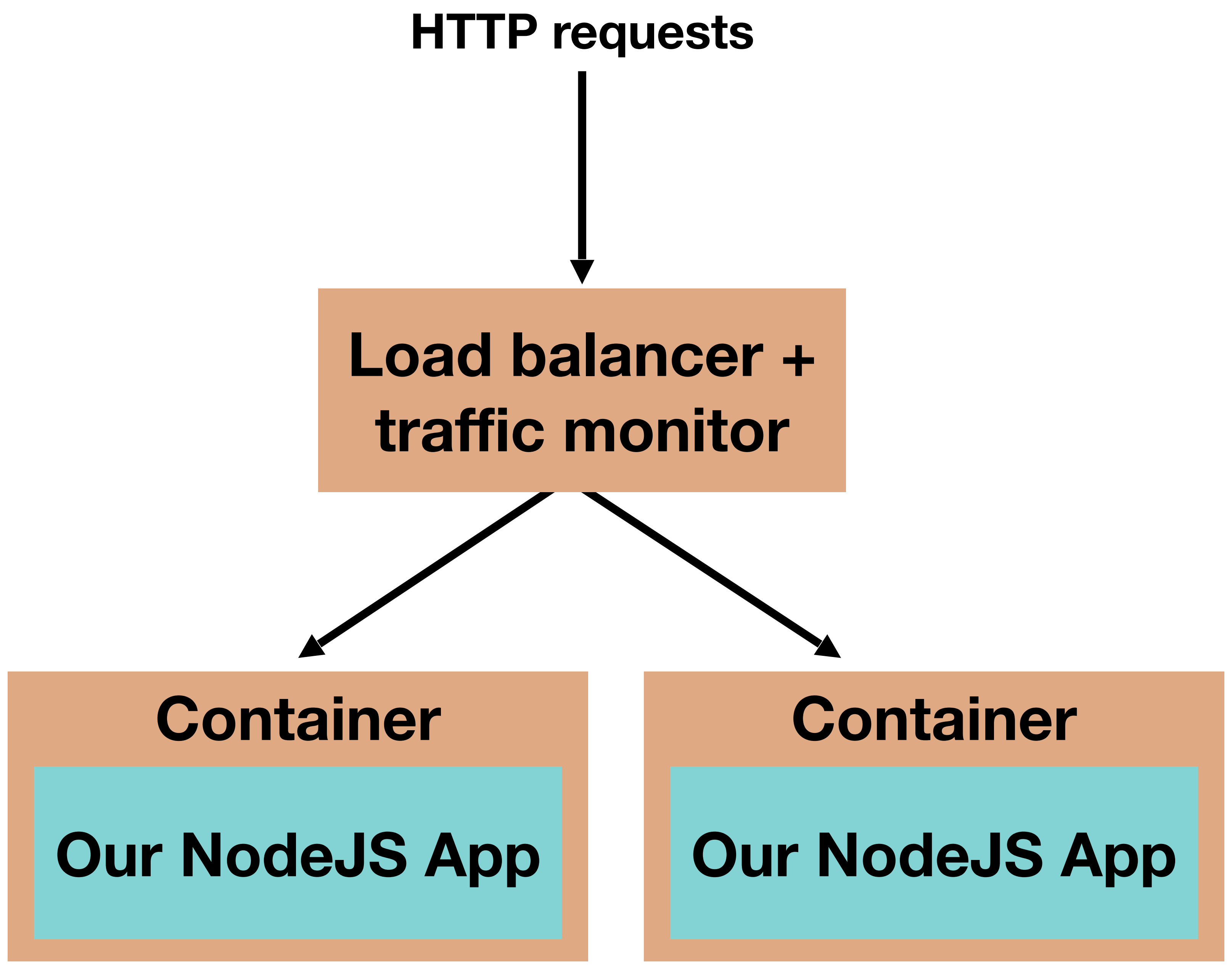
- Effective approach to “burst” capacity from private cloud to public cloud

Activity: Transcript server on Heroku

Heroku is one of the easiest platforms to use to host web apps

Objective:

- deploy our (completed) transcript server to Heroku
- Handout has detailed README



Review

You should now be able to…

- Describe what “cloud” computing is
- Understand the role of virtual machines and containers in cloud computing
- Deploy a web app to the cloud