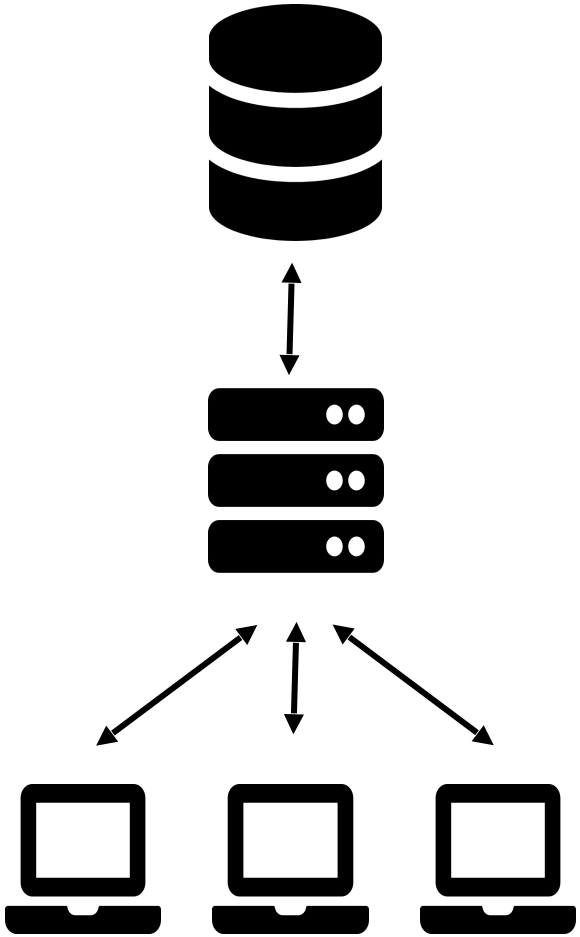
Microservices

(Slides prepared by Derek Brown)

1. A Short History of Applications at Scale

Motivation: The Early web



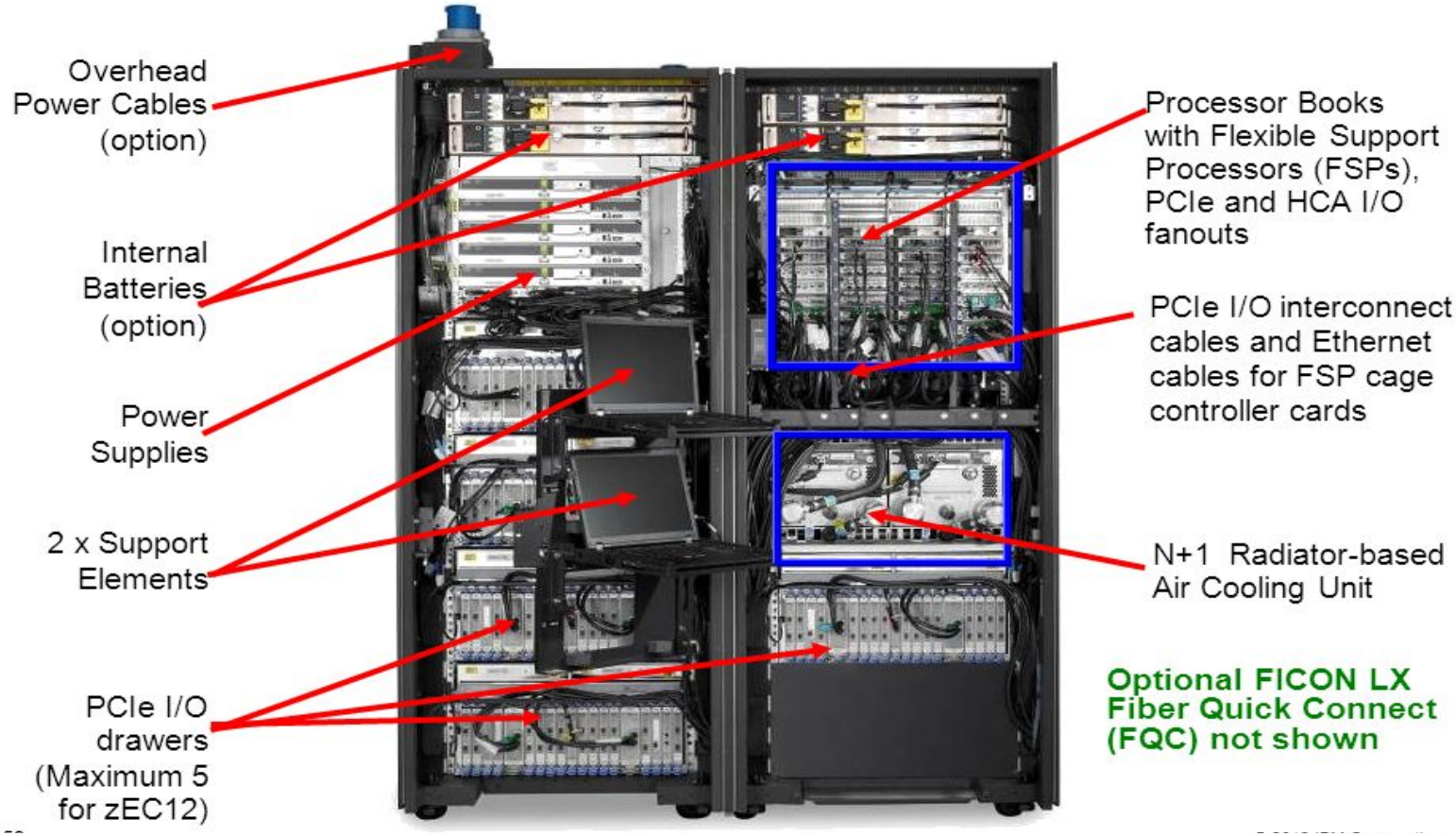
Database Process
(SQL)

Single, Monolithic App Server
(PHP, Rails, Java EE)

Limited Clients, Mostly Text Data

Q: All the sudden, we have a lot more clients. What do we do?

Q: All the sudden, we have a lot more clients. What do we do?



MAINFRAME!

Motivation: The Early Web

Advantages of the “Mainframe Model”:

- Simple deployment model
- Easy to manage
  - Single point of entry for security / network access.

Motivation: The Early Web

Disadvantages of the "Mainframe Model":

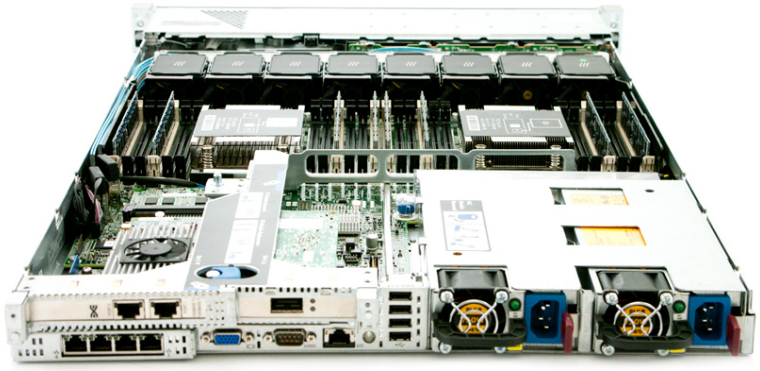
- Cost increases Exponentially with Scale
  - Increased power density
  - Requires smaller circuitry / higher operating frequencies
  - Complex

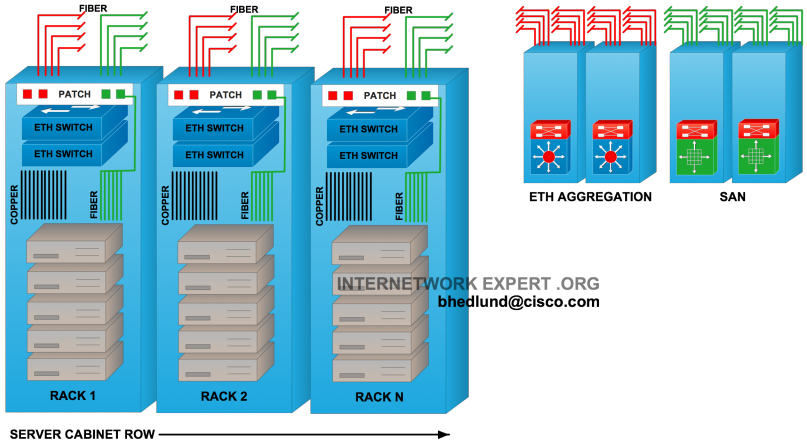
- Low Redundancy

- Resource Sharing Inefficiencies

Q: Eventually, single servers couldn’t be made any faster. So how do we continue to serve more clients?

Q: Eventually, single servers couldn't be made any faster. So how do we continue to serve more clients?

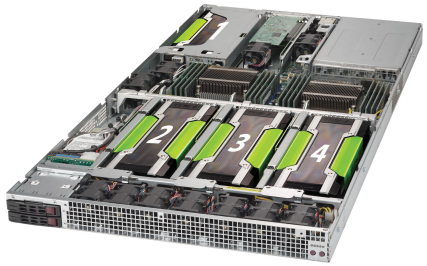




DATACENTER!

Aside: Not all servers are created equal

Block / Object Storage Servers
Store data for other applications like Databases.



GPU Servers
Perform parallel processing (ex. Machine Learning)

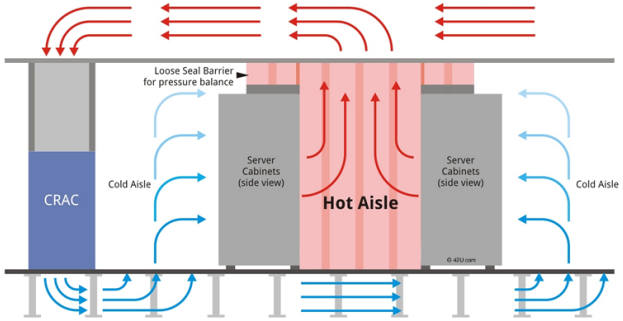
Blade Servers
Small dedicated instances for clients / applications.

General Compute
Good all-around servers

Appliances / Middleboxes
Servers with preloaded software to serve a dedicated purpose.
(ex. Firewall, SDN, VM Server)

Remote Access Servers
Small servers embedded in big servers for provisioning and control.

Motivation: Datacenters are complicated



- Complex network setup (Single/Multi Fiber, Network Switches, External Switching)
- Computer Room Air Cooling system (CRAC) to maintain operating temperature.
- Uninterruptible Power Supply (UPS); multiple power “grids”, multiple power “phases”, battery backups for faults, diesel generators.

Motivation: Datacenters

Advantages of the "Datacenter Model":

- Scaling is relatively easy; just buy more servers.

- Fairly Redundant
  - Multiple servers can perform the same task.
  - Disks and Power supplies are "hot swappable", allowing servers to run even while maintenance is being performed.

- Application residency; applications can be put on a specific set of servers based on resource requirements.

Motivation: Datacenters

Disadvantages of the "Datacenter Model":

- Applications must be distributed/scalable
  - Hard to distribute data across multiple instances.

- Management is difficult
  - Network topologies can be complex
  - Automatic installation of OSes/App is hard

- Must physically provision servers

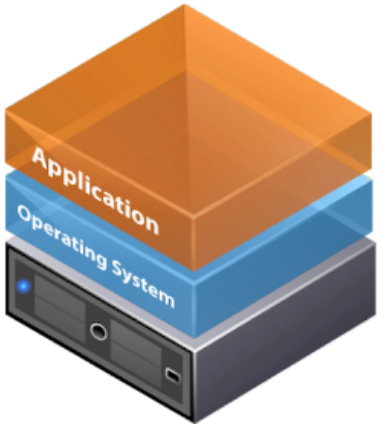
And… Servers can be Expensive

Item		Quantity	Price
	New PowerEdge R940 Rack Server Edit | Save for later | Remove	1	$393,350.00
			- $145,827.04
			$247,522.96

Q: WHAT IF I DON'T WANT A SEPARATE SERVER FOR EACH APP?

vmware®

Microsoft Hyper-V

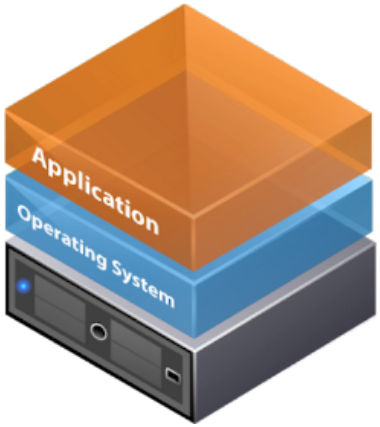


Xen™

Q: WHAT IF I DON'T WANT A SEPARATE SERVER FOR EACH APP?

vmware®

Microsoft Hyper-V



Xen™

VIRTUAL MACHINES!

Motivation: Virtual Machines

Advantages of Virtual Machines

- Can divide servers into separate smaller pieces
  - We don't need to devote a ton of resources to each app.
  - Means we can "over-schedule" servers.
  - Applications can run more securely as they are "sandboxed"
  - Allows for more complex virtual networking.

- Allows for hardware portability.

Motivation: Virtual Machines

Disadvantages of the Virtual Machines:

- Adds additional layers of abstraction
  - Reduces performance of system as emulation is slow.
  - Adds more complexity in terms of scheduling.

- More expensive (VMware ESXi ~$5k/server)

Q: I am a developer. I don't want to have to think about what physical server is hosting my app. How do I get just a generic server instance?

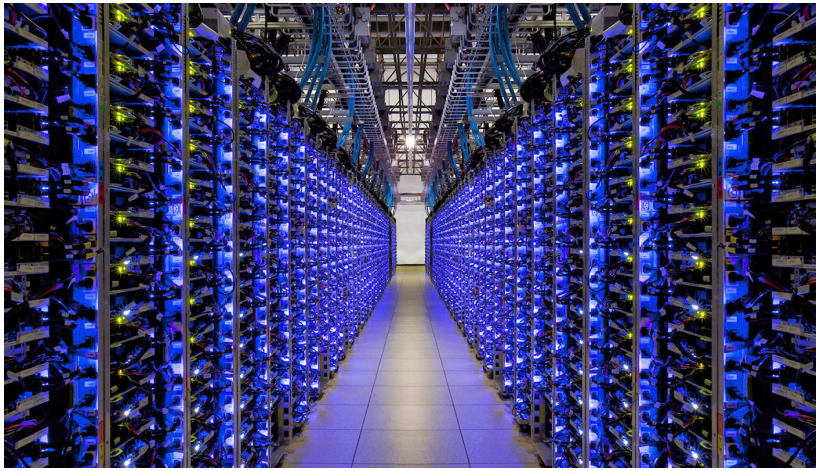
Google Cloud Platform

vmware®

openstack.

Azure

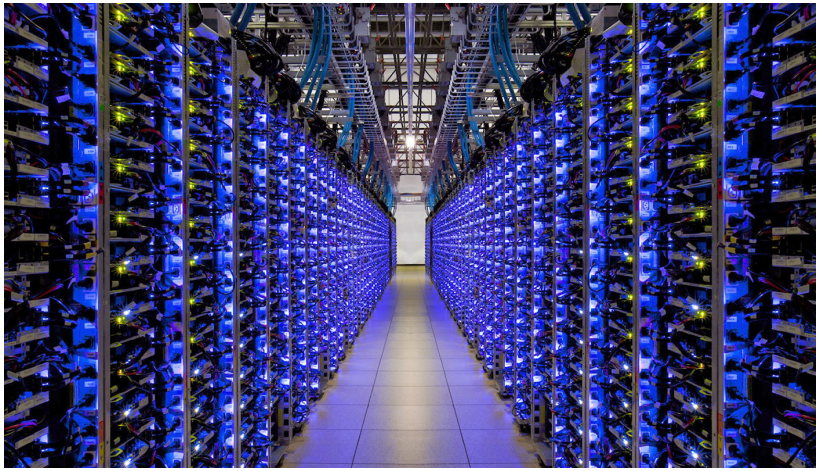
amazon web services



Q: I am a developer. I don't want to have to think about what physical server is hosting my app. How do I get just a generic server instance?

Google Cloud Platform

vmware



openstack

Azure

amazon web services

CLOUD!

Aside: A Tale of Two Clouds

Public Cloud

- Hardware managed by a 3rd Party
- Product is the server as a service

Private Cloud

- Hardware is managed internally.
- Product is the management platform

Motivation: Cloud

Advantages of the "Cloud":

- Really easy to provision an instance (as long as resources exist)

- Don't have to think about hardware
  - Can move "virtual machines" across systems
  - Can setup complex virtual networks
  - Can divide a server into smaller virtual instances
  - Automatic backup and replication

- Can over-schedule servers

Motivation: Cloud

Disadvantages of the “Cloud”:

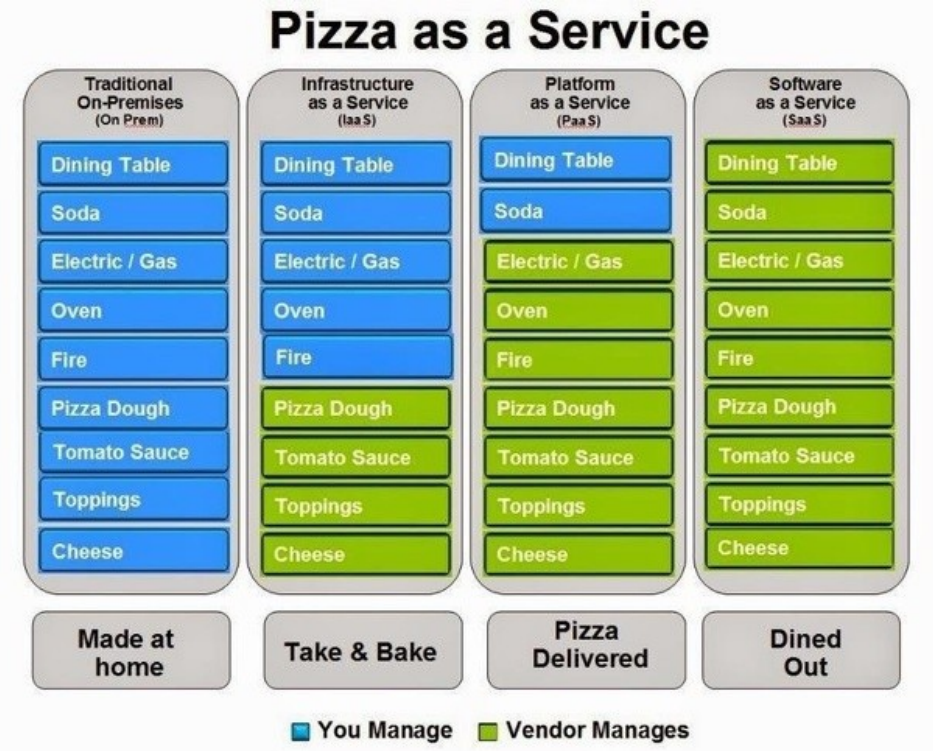
- Requires substantial overhead for VMs/management layer.
- Less control over data residency
- Other applications/customers can affect your application

Q: I am a developer. I don't want to have to think about servers at all. I just want to write code!

Q: I am a developer. I don't want to have to think about servers at all. I just want to write code!

PLATFORM AS A SERVICE!

Recap: By Analogy to Pizza



We are implicitly trading **control** for **ease-of-use**.

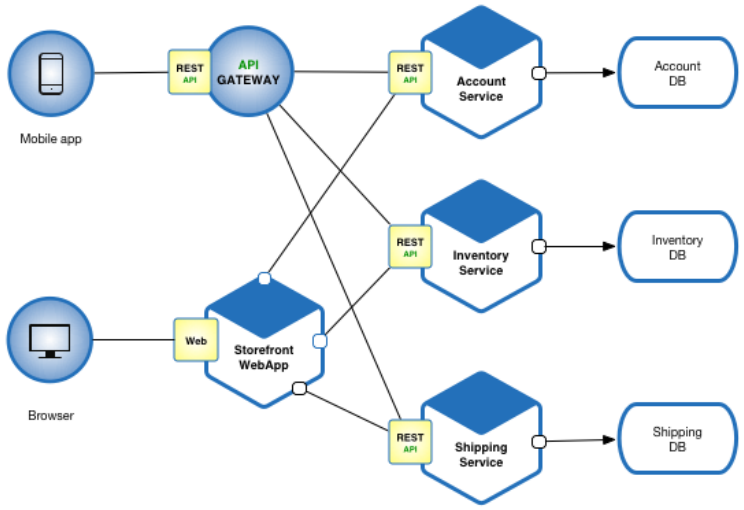
2. What are Microservices?

KEY IDEA:

The Microservice architecture is a method of developing distributed applications.

Microservice Philosophy

- Microservices are applications organized around a single capability.
- Small in size, modular, replaceable, independently deployable.
- Stateless- meaning they are scalable.



Aside - Stateless Programming

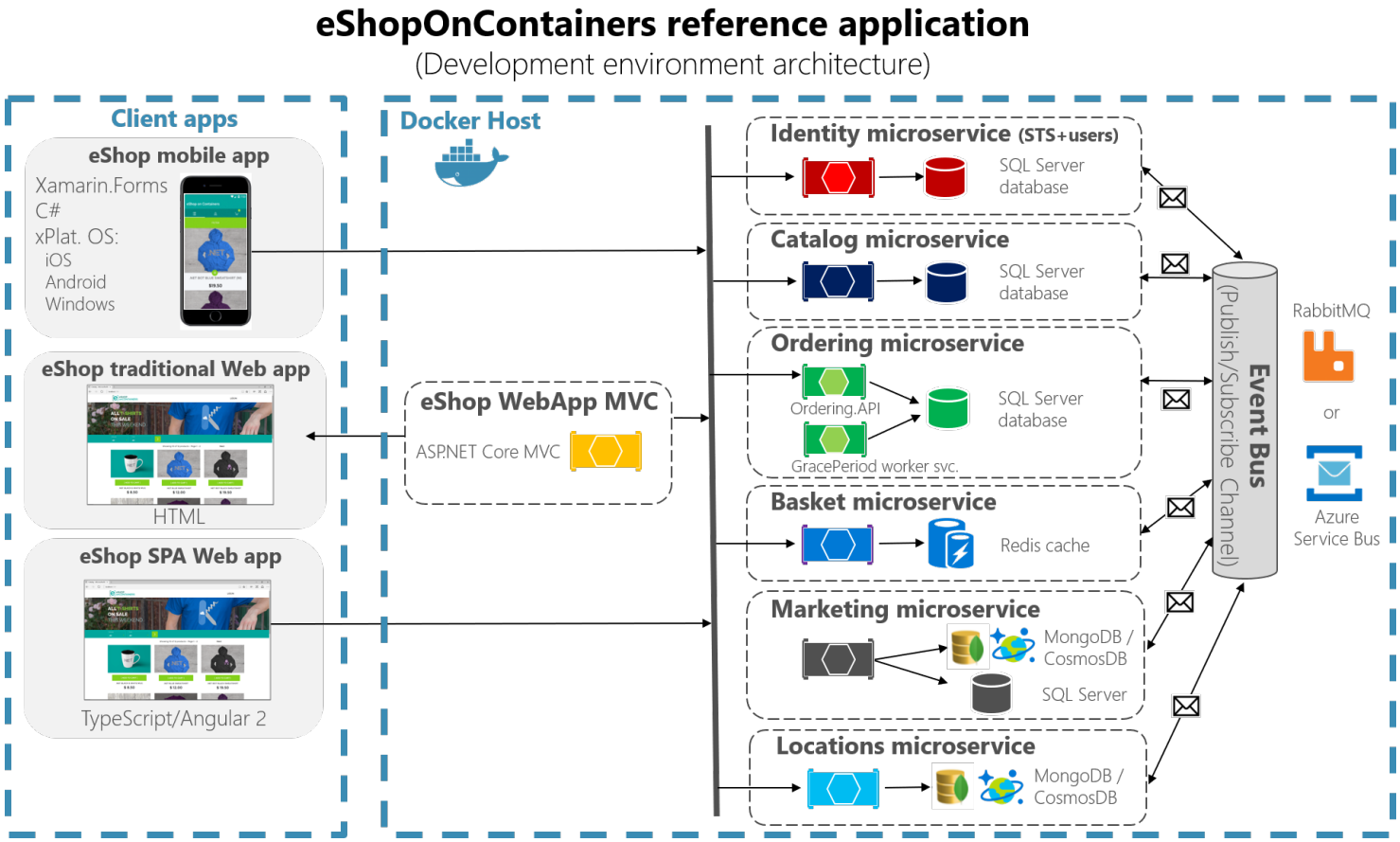
Stateless means there is no record of previous interactions and each interaction request has to be handled based entirely on information that comes with it.

SOURCE: http://whatis.techtarget.com/definition/stateless

Why?

- Microservices are continually created and destroyed. Data stored in an instance can be erased.
- Microservices are built to be scaled (in parallel). Storing stateful data can lead to race conditions.
- This offloads computation from the server to the client, speeding up the application.

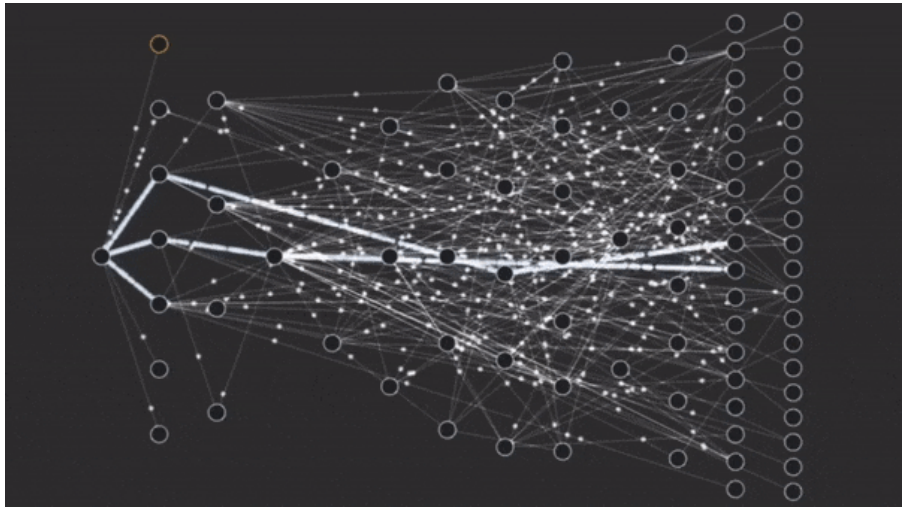
Example: Mobile Shop



SOURCE:https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/multi-container-microservice-net-applications/microservice-application-design

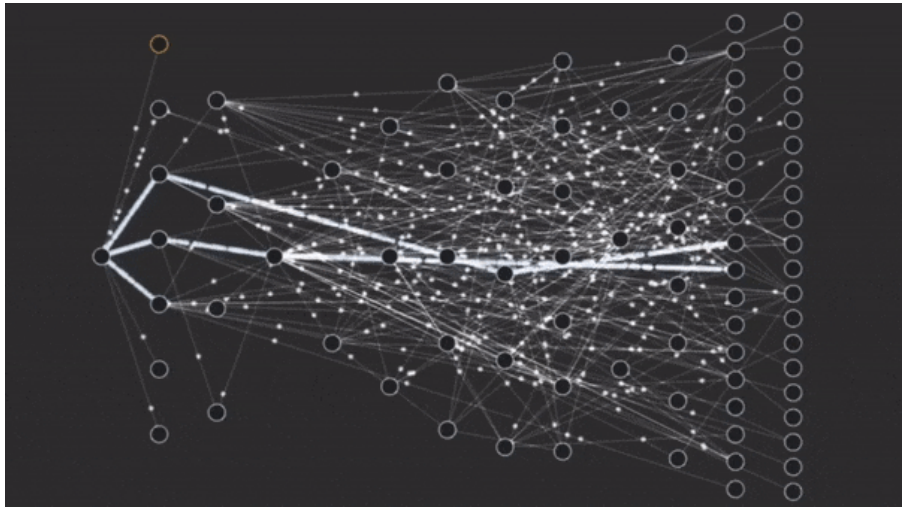
Class Exercise - Netflix

What types of microservices would you expect Netflix to run in production?



Class Exercise - Netflix

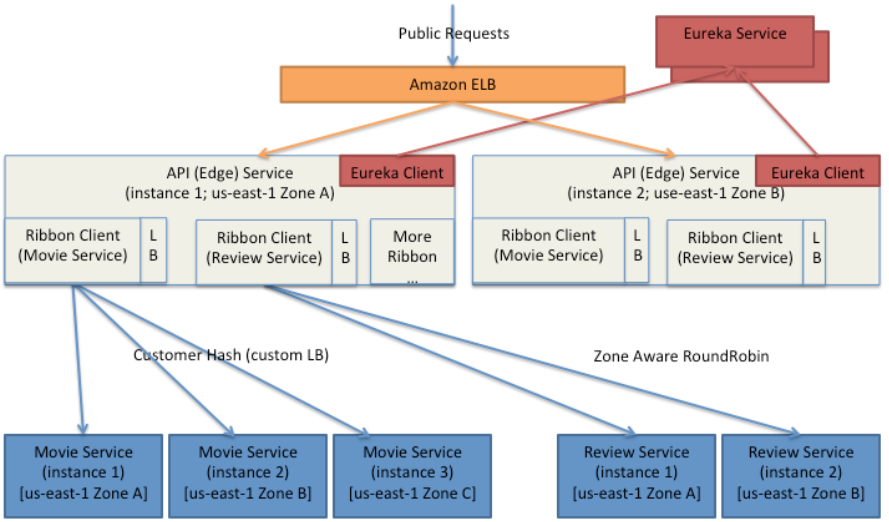
What types of microservices would you expect Netflix to run in production?



Class Exercise - Netflix

Services Netflix has:

- API Gateway Service
- Titles Service (for Metadata)
- Ratings Service
- Movies Service (for Data)
- Accounts Service
- A/B Testing Service
- Service Failure Service
- Rescheduling Service
- Log Collection Service
- Service Status Visualization Service
- Transaction Service
- ...



3. How does the Microservices Architecture affect software development practices?

Class Exercise

How would you expect microservices would affect:

1. Performance?
2. Error Tolerance / Recoverability?
3. Ability to find a bug in the codebase?
4. Team Size?
5. Length of development/release cycle?
6. Willingness to adopt new technologies?

Class Exercise

1. **Performance**
- Likely to increase as the application scales more flexibly.
- However, poor architecture (such as a bottleneck) reduces efficacy.
2. **Error Tolerance / Recoverability**
- Can be less fault tolerant, as a single failed service can cause entire application to fail.
- But this can be mitigated by redundant services and fallbacks.
3. **Ability to find a bug in the codebase**
- Much easier. Errors are segmented by service, reported directly to team.
4. **Team Size**
- Likely smaller, as less integration is required.
5. **Length of development/release cycle**
- Services are likely quick to develop, as they can be created on a blank canvas.
- Once created it can be difficult to refactor all the API, making change slow.
6. **Willingness to adopt new technologies**
- Likely more willing to adopt new technologies, as each service can operate on its own tech stack.