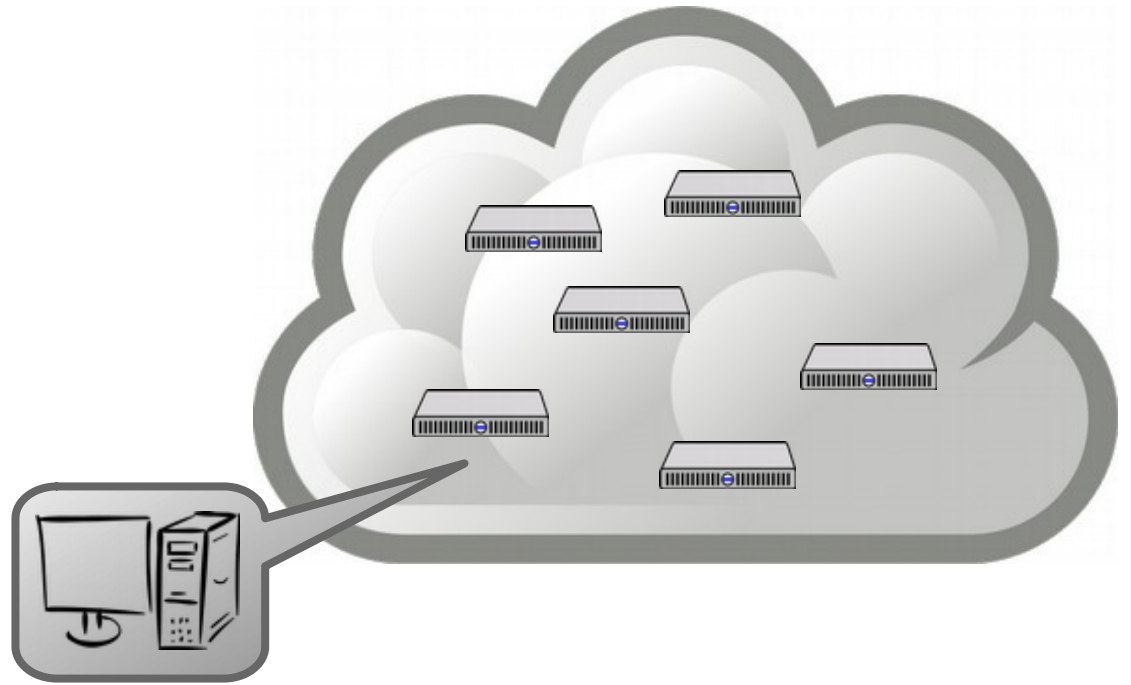


# CS 470 Spring 2025

Mike Lam, Professor



## Virtualization and Cloud Computing

Content taken from the following:

A. Silberschatz, P. B. Galvin, and G. Gagne. "*Operating System Concepts, 9<sup>th</sup> Edition*" (Chapter 16)

Various online sources; some images from wikipedia.org and openclipart.org

# Problem

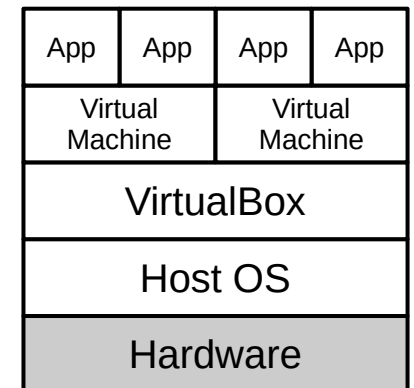
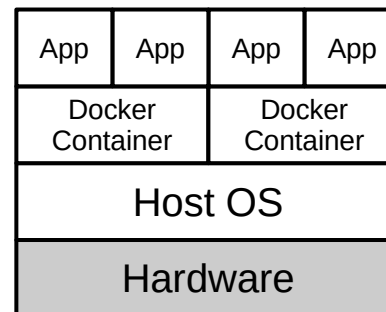
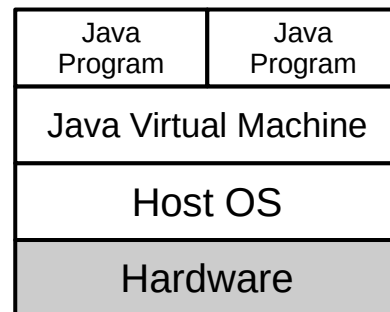
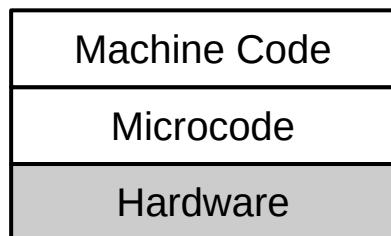
- Distributed systems are now ubiquitous
  - It's hard to provide any software service at a modern scale from a single server
    - (Although if you can, you SHOULD!)
  - Many companies don't need/want to manage hardware
    - High up-front costs, security vulnerabilities, etc.
  - Solution: abstraction!
    - In particular, abstracting away the hardware
      - Sometimes software too
    - Usually referred to as **virtualization**

# Virtualization

- **Virtual environment**: abstract machine (**guest**) implemented on top of another (sometimes physical) machine (**host**)
  - Requires some kind of interpretation layer
- Various goals:
  - **Emulation**: run programs designed for one architecture on another
  - **Isolation**: run programs in a sandbox
  - **Scalability**: spawn/destroy instances dynamically
  - **Automation**: reduce tedium and mistakes during deployment
  - **Reproducibility**: suspend/resume snapshots or configurations

# Virtualization

- Various levels
  - Circuits / CPU (**microcode** emulating machine code)
  - Storage (e.g., **RAID**)
  - Networks (e.g., **NAT** or **overlays**)
  - Runtime environment (e.g., **Java VM** or **Microsoft .NET**)
  - Operating system (e.g., **Docker**)
  - Full desktops (e.g., **QEMU**, **VMware** or **VirtualBox**)



# Hypervisors

- **Native** hypervisors (“type 1”)
  - Run directly on the host’s hardware in kernel mode
  - Sometimes as part of a general-purpose OS
  - Examples: [VMware ESX](#), [Microsoft Hyper-V](#), [Oracle VM Server](#), [Xen](#)
- **Hosted** hypervisors (“type 2”)
  - Runs as a process inside the host OS
  - Often hardware-accelerated (e.g., [Intel VT-x](#) or [AMD-V](#))
  - Examples: [VMware Workstation](#), [VirtualBox](#), [QEMU](#)
  - Sometimes called an **emulator** if it virtualizes a different architecture
    - Example: Project 4 in CS 261 is a Y86-64 emulator for x86-64

# Windows: 3.1, 95, and 10 on 8.1

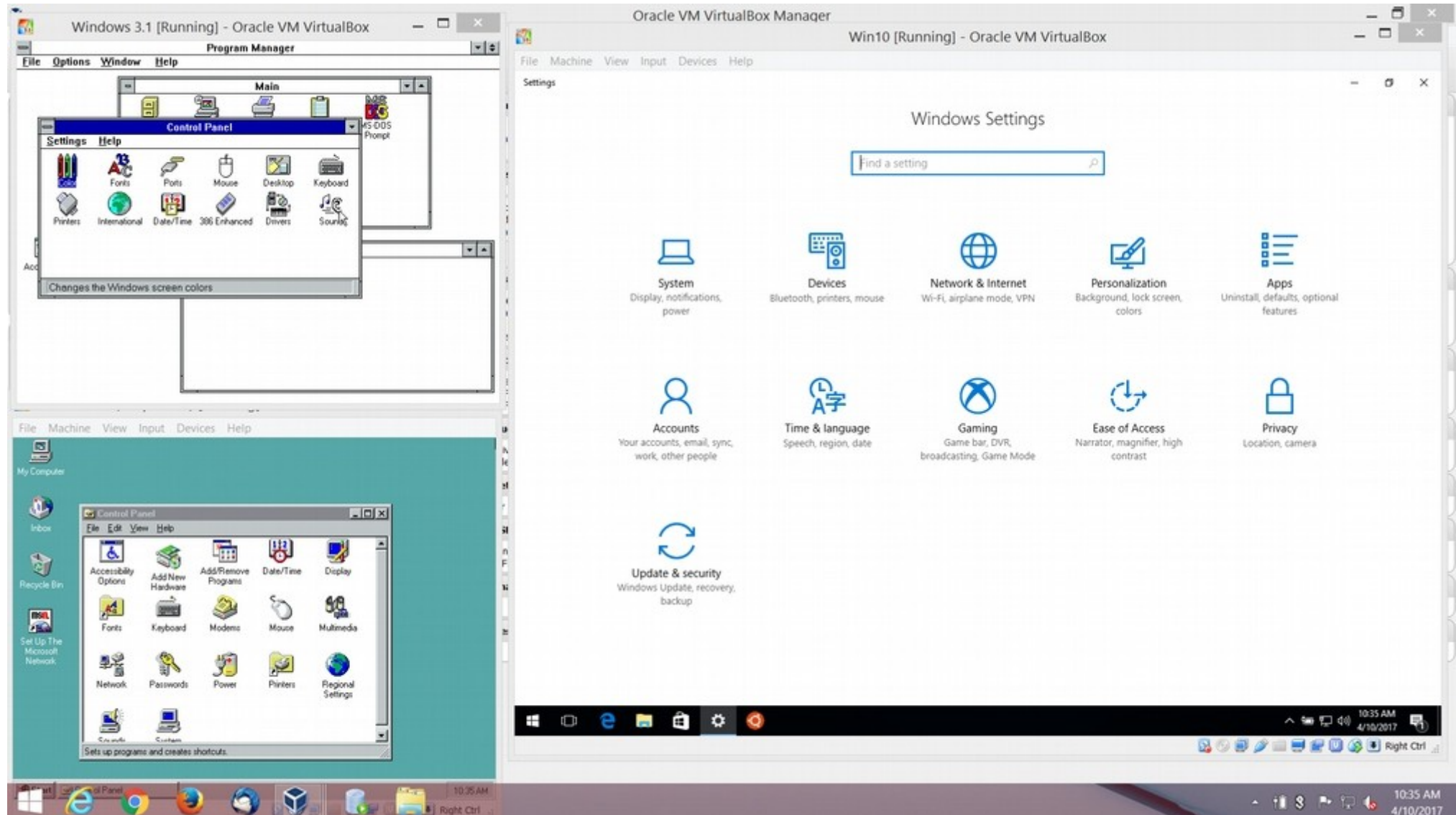


Image courtesy of Mike Ripley (JMU Infrastructure and Database Support)

# OS-level virtualization

- **Container**: isolated user space for a program and its dependencies
  - Multiple user spaces implemented at the kernel level
  - Alternative descriptions:
    - Virtual memory extended to files and libraries
    - Sandboxed, lightweight, app-specific VMs that run natively (no guest OS)
    - “Packages” for a single program's file system
  - **Performant**: minimal overhead vs. running natively
  - Examples: [chroot](#), [FreeBSD jail](#), [Docker](#), [Apptainer/Singularity](#)



# Cloud computing

- **Cloud computing**: technically, it's more nuanced than just “other people's computers”



<https://fsfe.org/contribute/spreadtheword#nocloud>



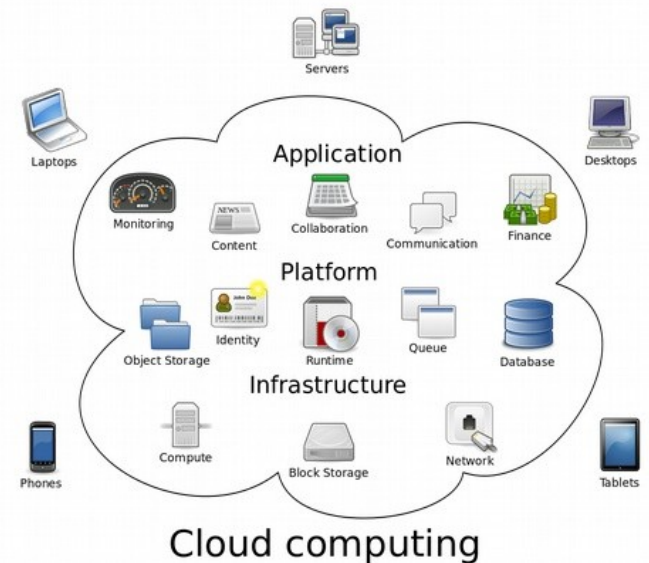
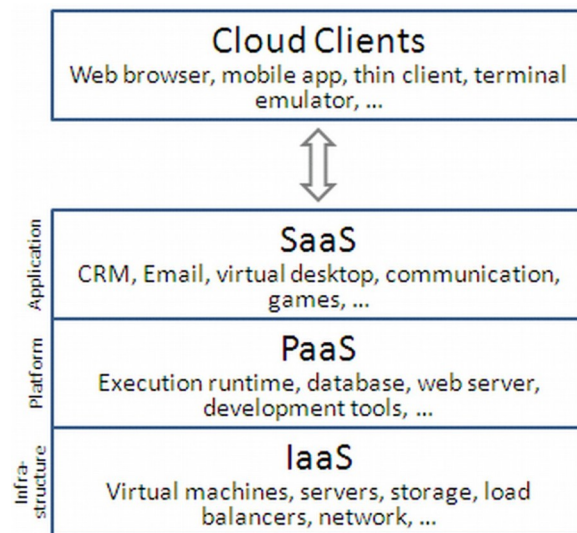
# Cloud computing

- Essential characteristics (from NIST definition\*)
  - **On-demand self-service** for provisioning
  - **Broad network access** for availability
  - **Resource pooling** for independence
  - **Rapid elasticity** for scaling
  - **Measured service** for transparency
  - Examples: [Amazon Web Services](#), [Google Cloud Platform](#), [Microsoft Azure](#), [Rackspace](#)



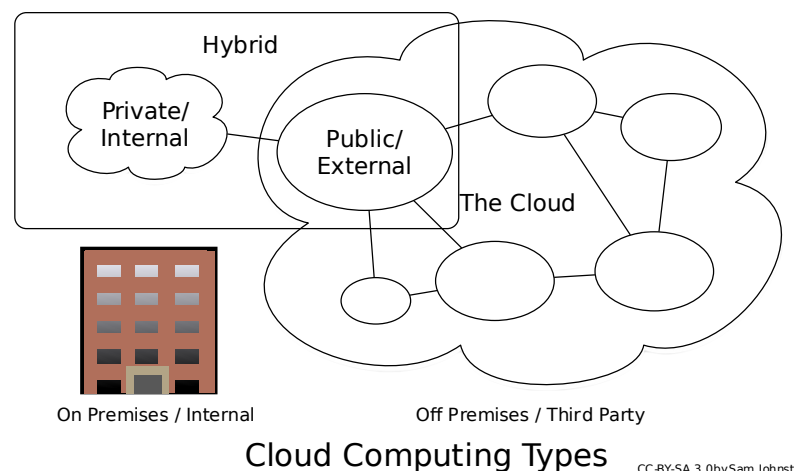
# Cloud computing

- Service models (from NIST definition\*)
  - Software as a Service (SaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (IaaS)



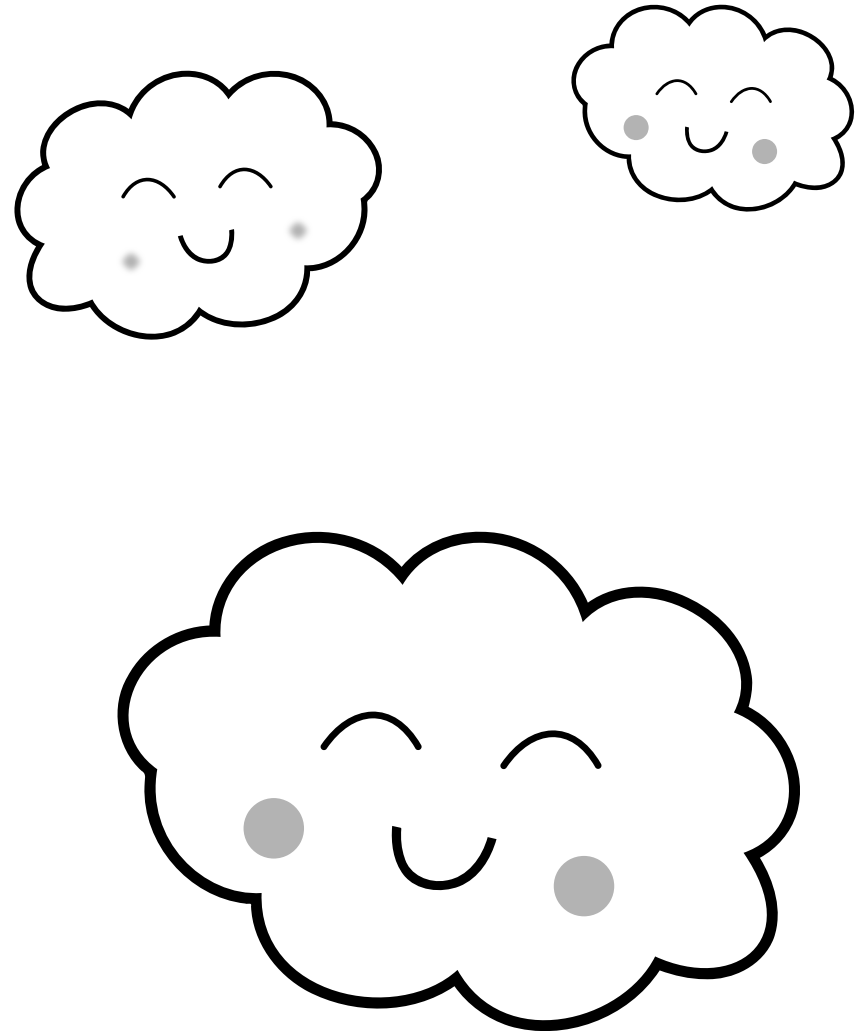
# Cloud computing

- Deployment models (from NIST definition\*)
  - Private (single organization)
  - Community (multiple organizations)
  - Public (open to general public)
  - Hybrid (combination of above)



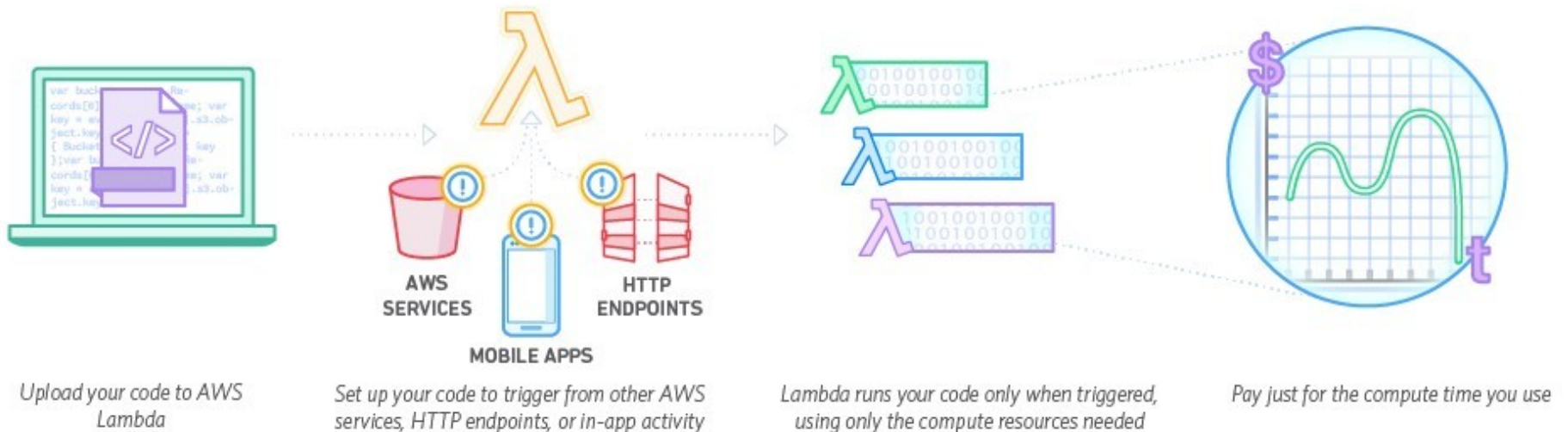
# Everything as a service (EaaS/XaaS)

- Analytics as a service (AaaS)
- Backend as a service (BaaS)
- Communication as a service (CaaS)
- Containers as a service (CaaS)
- Content as a service (CaaS)
- Data platform as a service (dPaaS)
- Desktop as a service (DaaS)
- Function as a service (FaaS)
- Games as a service (GaaS)
- Hardware as a service (HaaS)
- Integration platform as a service (iPaaS)
- IT as a service (ITaaS)
- ...
- Workspace as a service (WaaS)
- Hybris as a service (YaaS)
- Zenoss as a service (ZaaS)



# Cloud computing

- “Serverless” computing
  - FaaS: **Function as a Service** (another layer of abstraction!)
  - Pay for compute time, not a particular host or VM
  - There's still a server, but the user doesn't interact with it directly
  - Code must be written using a supported language
  - **Amazon Lambda**, **Google Functions**



# Cloud engineering

- Emerging/developing field
  - Combines computer system engineering (EE), software engineering (CS), and computer information systems (business)
  - Focus on IaaS/PaaS/SaaS/FaaS applications
    - Often with a “big data” focus
  - Goals: performance, scalability, security, reliability
  - Challenge: integrating multiple solutions and layers
  - First IEEE International Conference on Cloud Engineering (IC2E) in March 2013