University of CINCINNATI

Virtualization

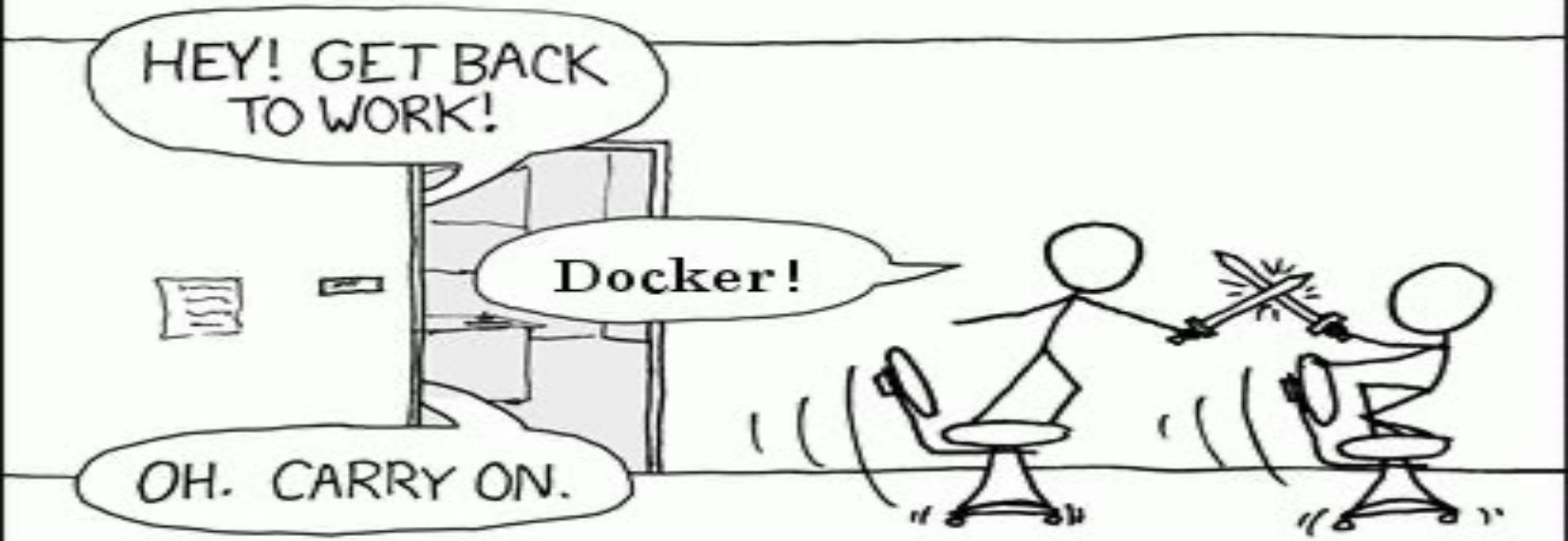
Concepts of Virtualization

(Mastering Cloud Computing: Chapter#3)

Rashmi Kansakar

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

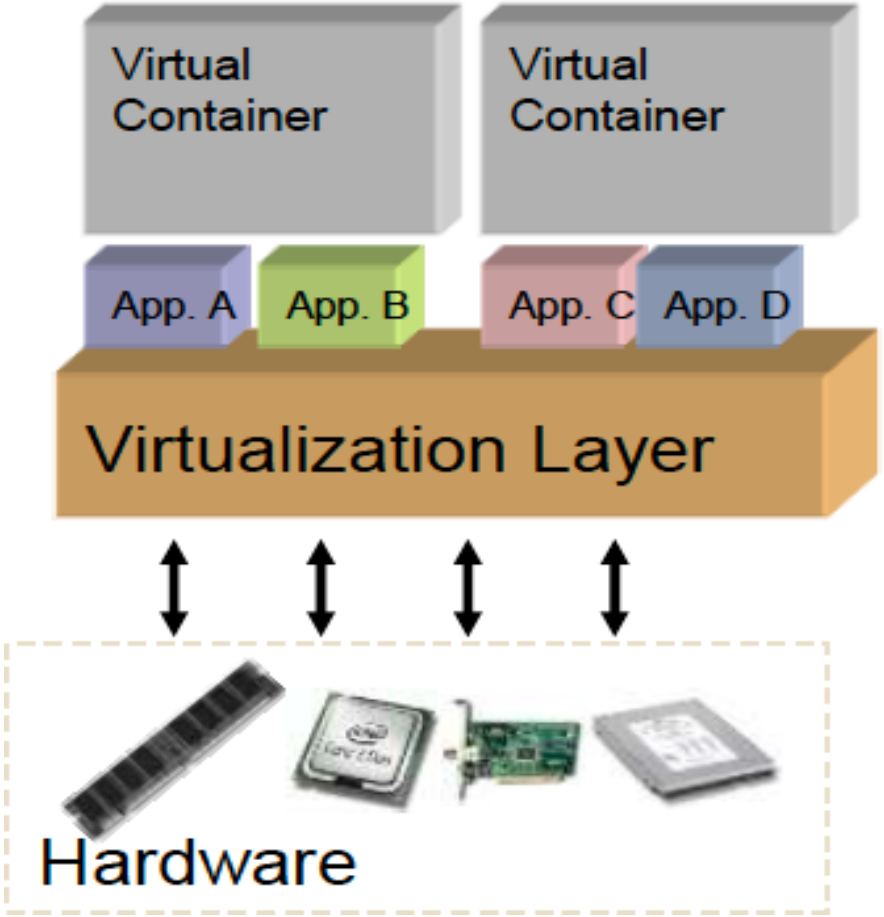
"My Docker containers are building!"



Definition…

University of CINCINNATI

- Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something…
  - … **virtual computer hardware platform, operating system (OS), storage device, network resources**
- Pooling of physical resource from multiple resources to appear as single device managed by single physical server & distribute them

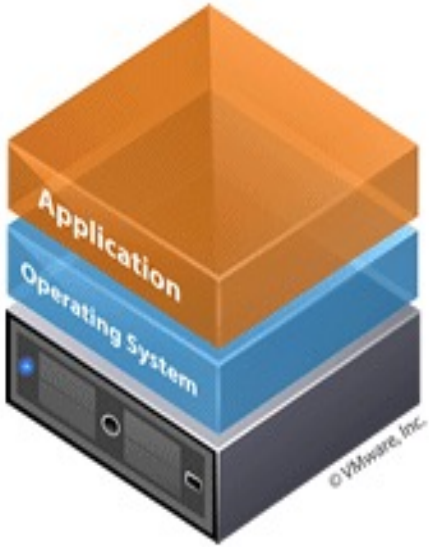


Simple definitions …

University of CINCINNATI

- Virtualization is a Technology that **transforms hardware into software**.
- Allows **multiple isolated version OS's and workloads** to run on the same **physical** hardware as virtual machines.

- Application virtualization allows you to **distribute multiple copies** of an **application** from a **single** physical server.
- Provide **direct access** to the hardware resources to give much **greater performance**



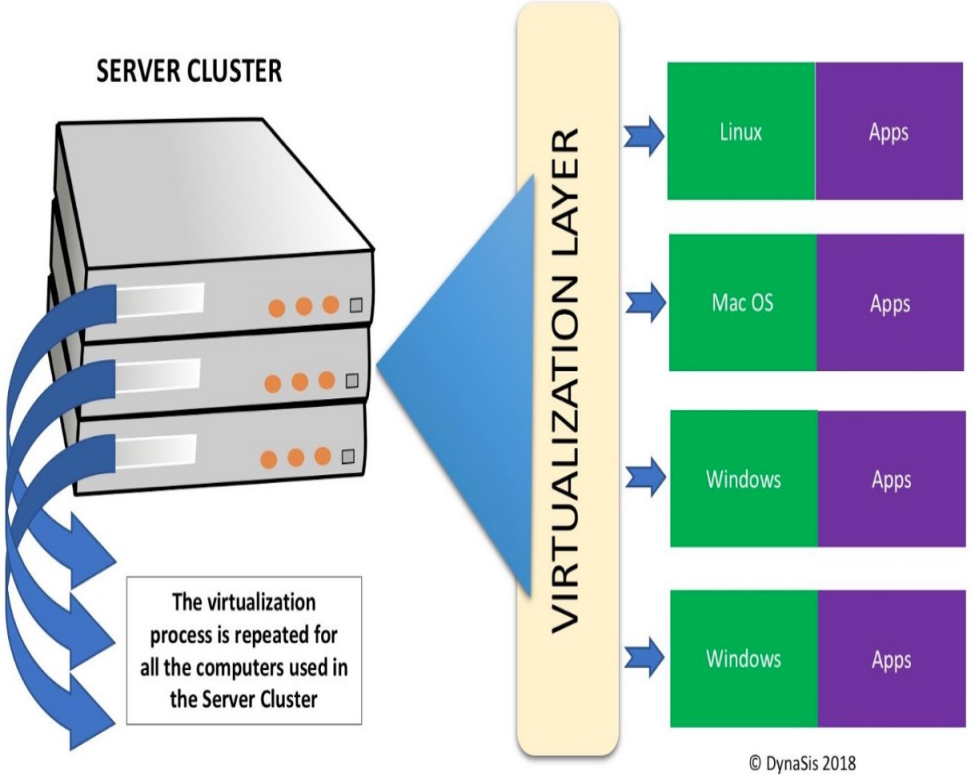
Traditional Architecture

Virtual Architecture

Organization Chart

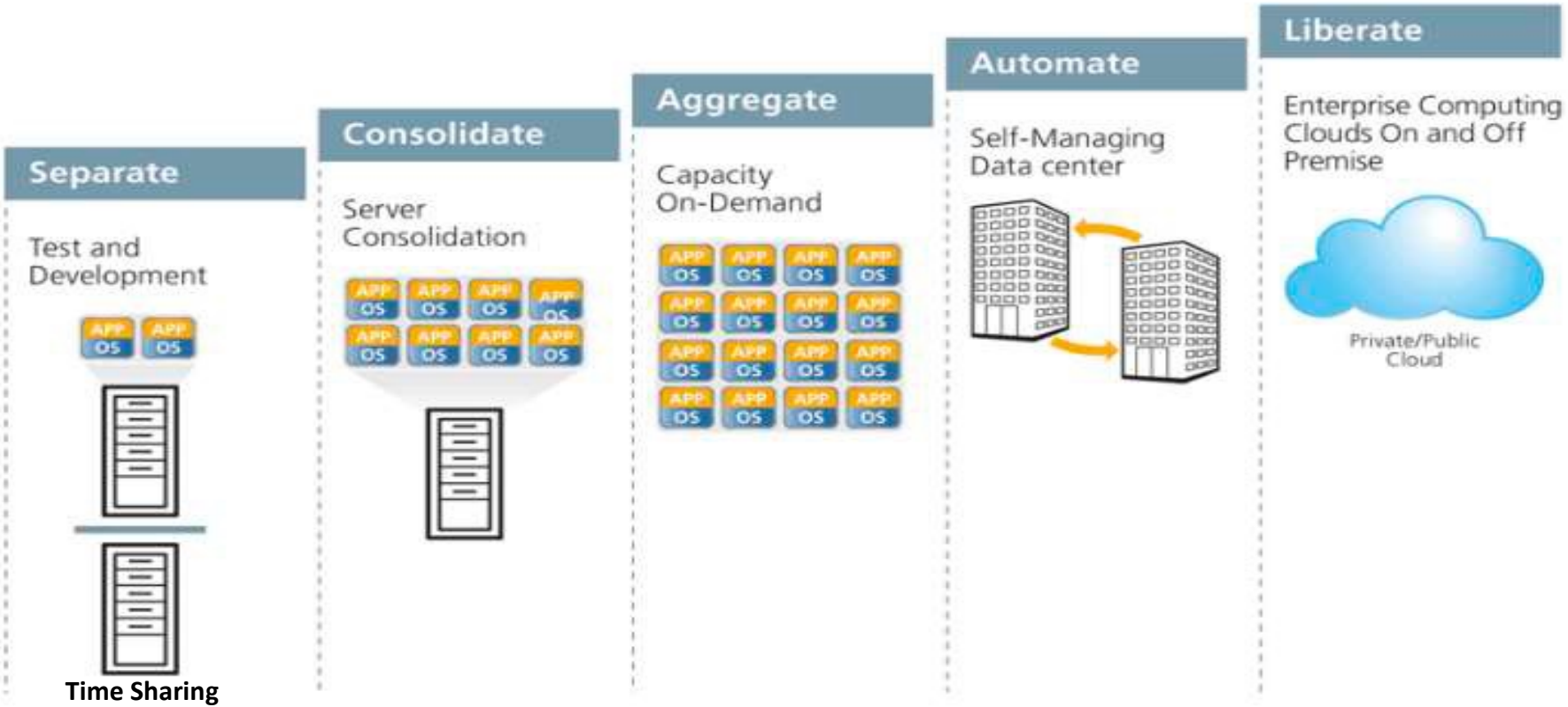
University of CINCINNATI

- **PARTITIONING**
  - Run multiple operating systems on one physical machine
  - Share physical resources between virtual machines
- **PORTABILITY**
  - Entire virtual machine is saved as a file, so…
  - Move, copy, or export as easily as a file
- **SECURITY**
  - Hardware is isolated from the operating system
  - Recovery as easily as restoring a file
- **AGNOSTIC**
  - Migrate a virtual machine between similar, or different, physical servers



Evolution of Virtualization towards Cloud

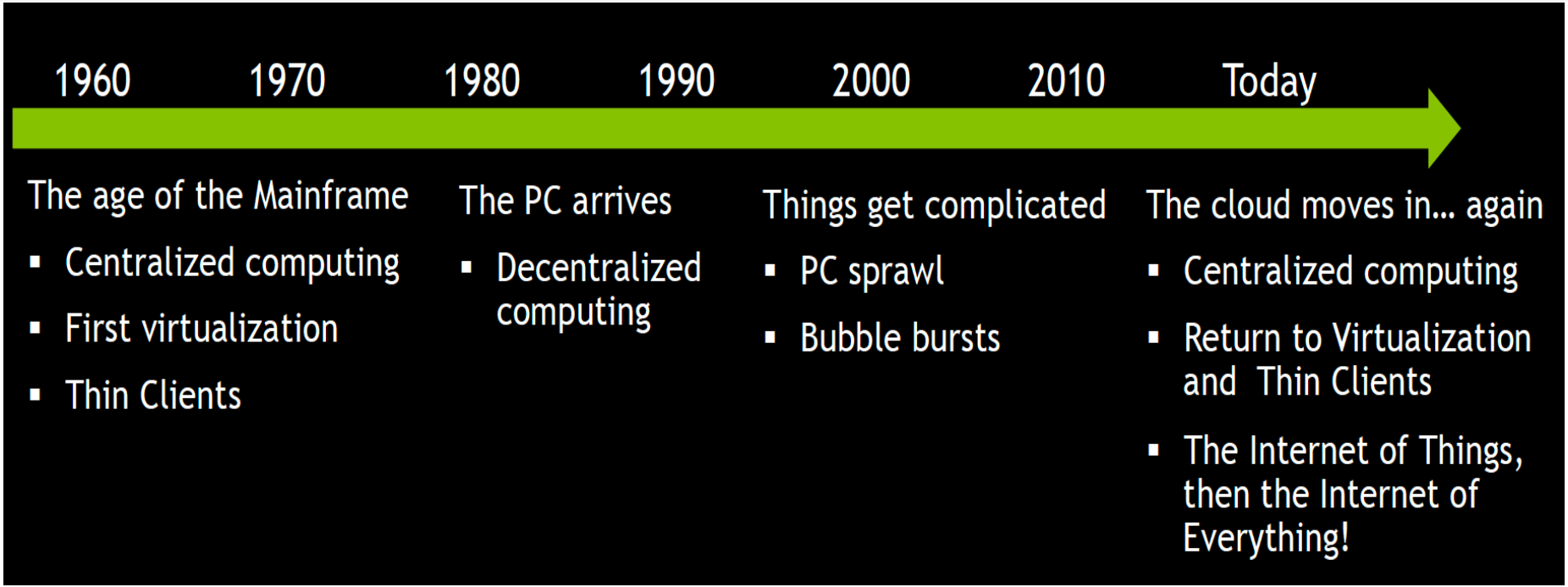
University of CINCINNATI



IBM Mainframes w/ Virtualization > Personal Computers > Virtualization to reduce Capex > **CLOUD**

Short History

University of CINCINNATI



- The idea behind VMs originates in the **concept of virtual memory and time sharing**
- The term "virtualization" traces its roots to 1960s mainframes
- Method of **logically dividing** the physical resources for different applications.

What Is A Virtual Machine ?

University of CINCINNATI

- **Isolated guest operating** system installation within a host operating system.
- Virtual machines posse's control of hardware virtually.
- Virtualization allows **transformation** of a server for **multiple** applications
- Uses virtualization software & hardware virtualization techniques
- From the user perspective virtual machine is **software platform like physical computer** that runs operating systems and apps.

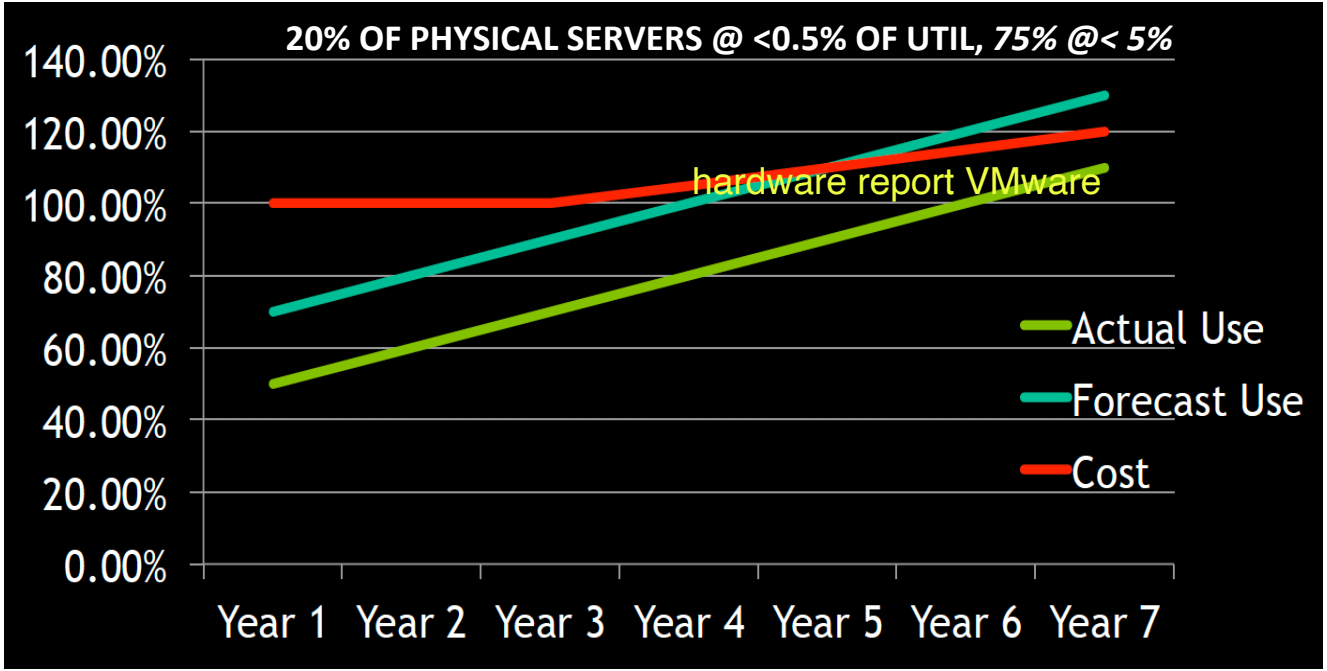
Virtual Server

Why Virtualization?

University of CINCINNATI

- Business demand: **Cost savings, Flexibility, Mobility, Speed**
- End users demand: **Frequent refresh, More Power, Mobility, BYOD, Graphics**
- Allows **multiple applications** to run in **isolation** within VM on the same physical machine
- Increased computing power
- Lack of space
- Greening initiatives
- Rise of administration costs
- Centrally managed
- Ease of support

RESOURCE OPTIMIZATION: Underutilized HW Resource



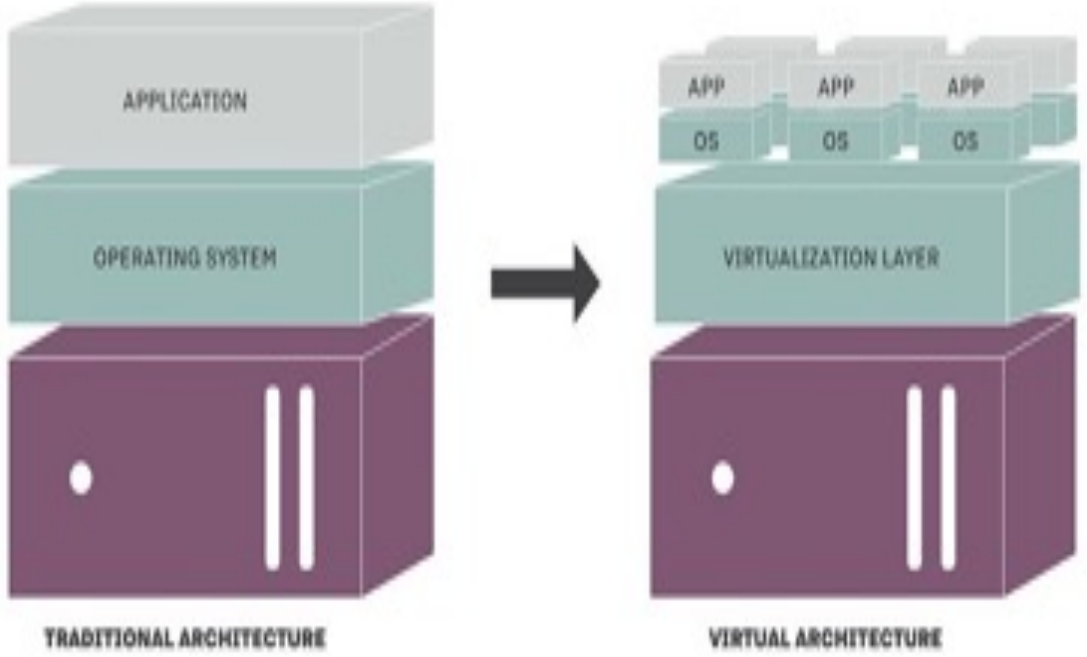
Physical Server vs Virtual Server

University of CINCINNATI

Virtual machines provide:

- Hardware independence
  - **Guest VM** sees the same hardware regardless of the host hardware
- Isolation
  - –VM's operating system is **isolated** (different) from the **host** operating System
- Encapsulation
  - Entire VM **encapsulated into a single file**

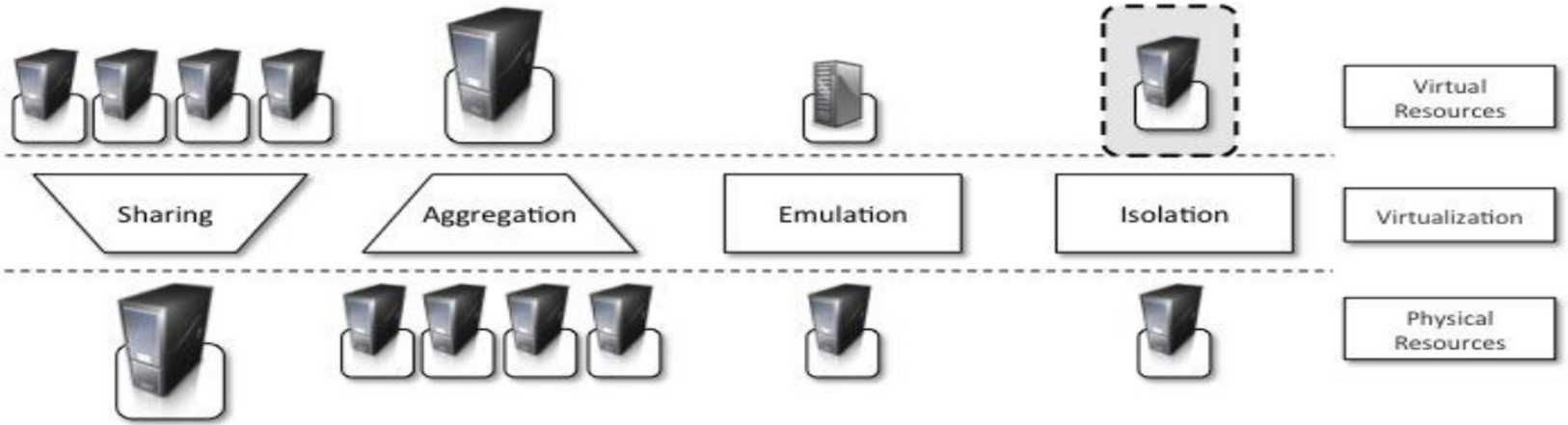
VIRTUALIZATION
= COMPUTING RESOURCES ABSTRACTION
+. PARTITIONING OF ONE PHYSICAL SYSTEM



Managed Execution

University of CINCINNATI

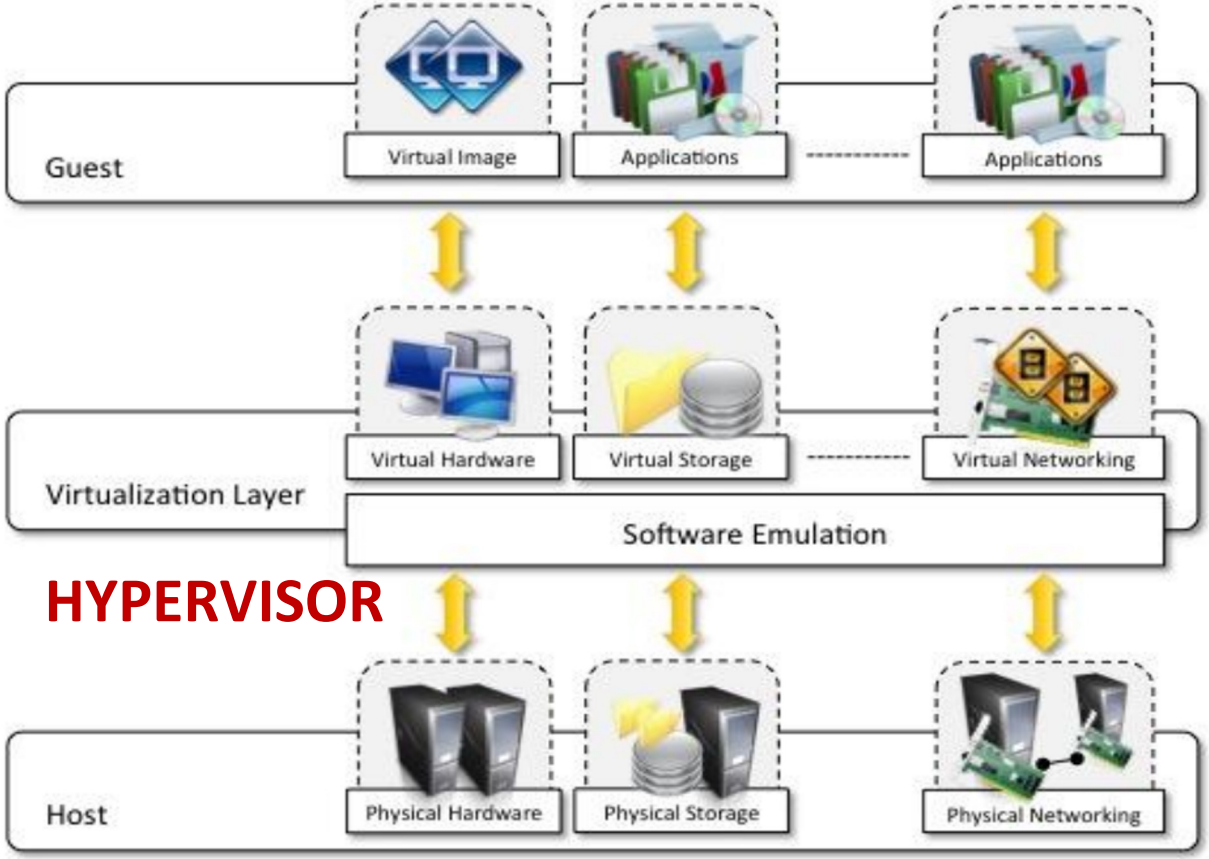
- Additional features:
  - Sharing - better utilization
  - Aggregation - many looks like 1
  - Emulation - provide different hardware to host
  - Isolation - security



Organization Chart

University of CINCINNATI

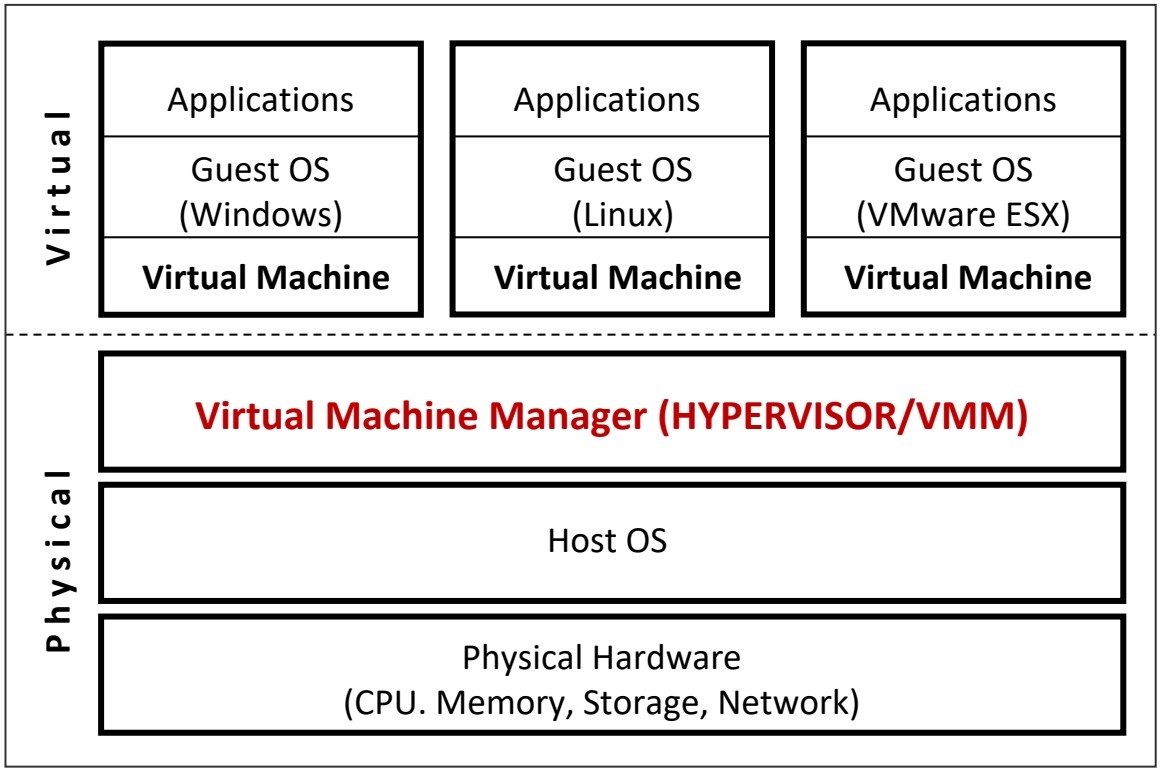
Software Environment →



Virtualization Architecture

University of CINCINNATI

- **HYPERVISOR a**llows multiple OS to interacts directly with the physical server's CPU and disk space, OS allocation, and make sure that the **Guest OS** (called virtual machines) do not disrupt each other
- Software environment (hardware VM)
- Storage
- Network (VPN)
- 3 Components:
  i. Guest
  ii. Virtualization layer
  iii. Host
- Multiple isolated environments



Increased Security… How?

University of CINCINNATI

- VM manager can control and filter activity of the guest.
- Hide information on host
- Sandbox environment (JVM & .Net)
- Adjacent virtualized hosts can't spy on each other :
  - An exploration of L2 cache covert channels in virtualized environments
  - A survey of security issues in hardware virtualization

Other Features

University of CINCINNATI

- Performance tuning
  - tune host for optimal performance
  - expose custom hardware to guest
- VM snapshots
  - pausing - saving - resuming
- VM migration
  - move a vm from one host to another,
  - sometimes while running
- Portability
  - move VM from host to host

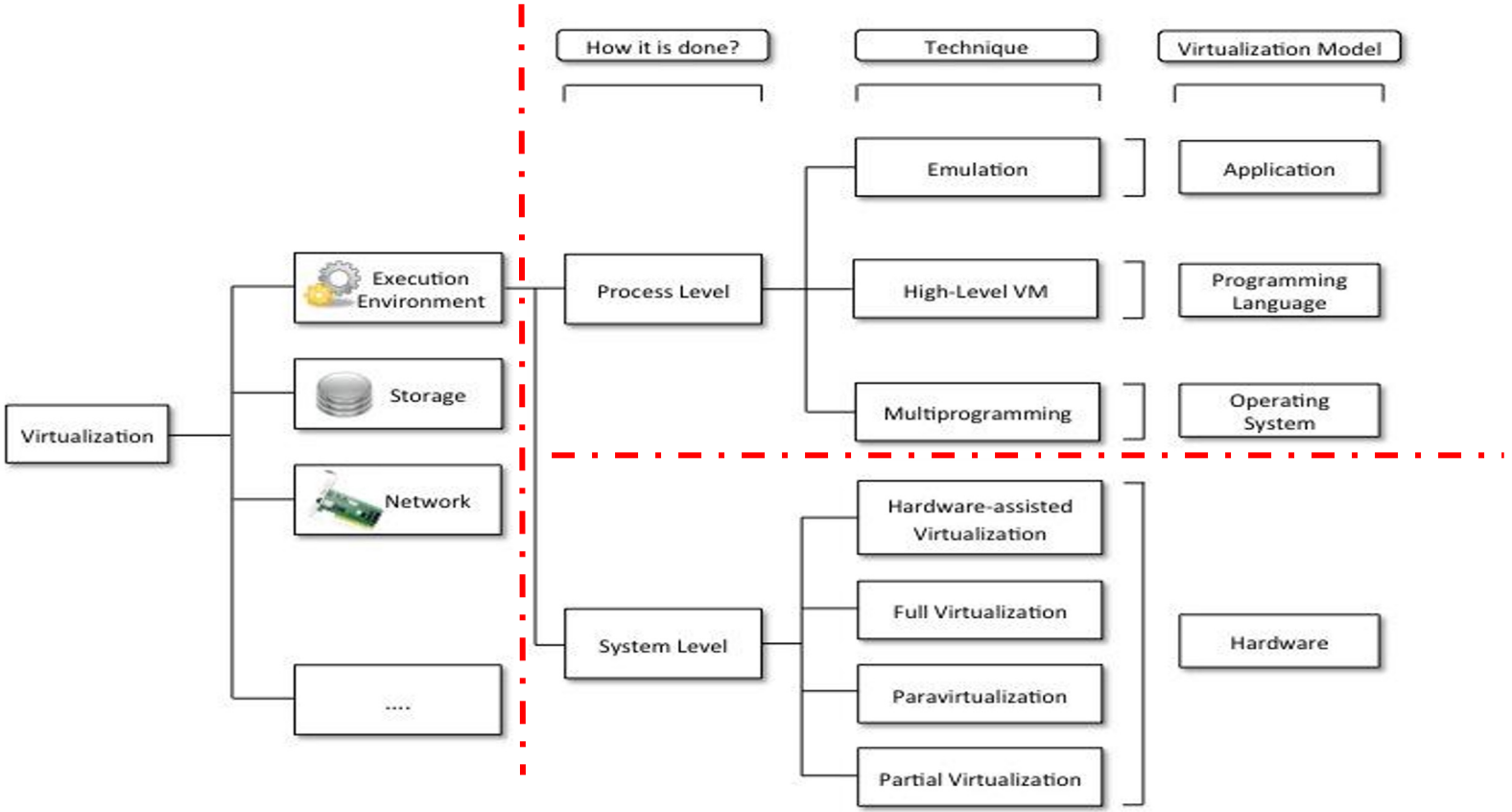
Virtualization Taxonomy (*Execution Virtualization*)

2 Main categories:

- Process-level
  - On top of an existing OS
- System-level
  - Directly on hardware or minimal OS support

Virtualization Taxonomy Flowchart

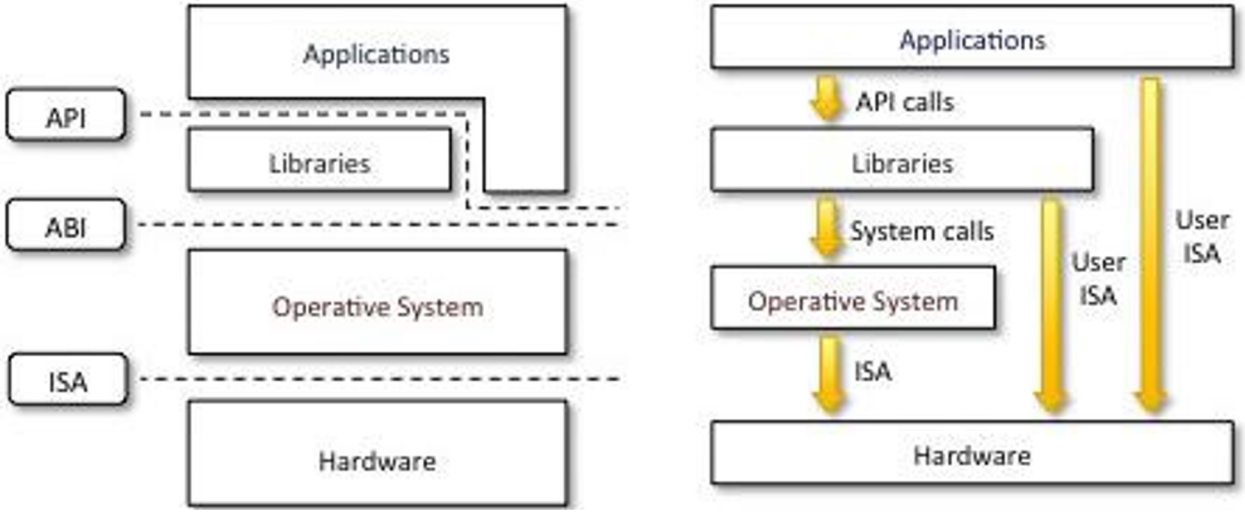
University of CINCINNATI



Machine Reference Model

University of CINCINNATI

- Defines layer of abstraction
  - ISA - instruction set architecture, registers, memory, interrupts
  - ABI - application binary interface, data-types, alignment, system-calls
  - API - application programming interface, libraries and underlying OS



Layered Approach helps with Security

University of CINCINNATI

- Hardware can allow different layers to execute different instructions.
  - Privileged
    - change shared resources -
    - IO and register changing instructions
  - Nonprivileged
    - safe - don’t access shared resources - math instructions
- From the user perspective, virtual machine is software platform like physical computer that runs operating systems and apps.

Hierarchy of Privileges

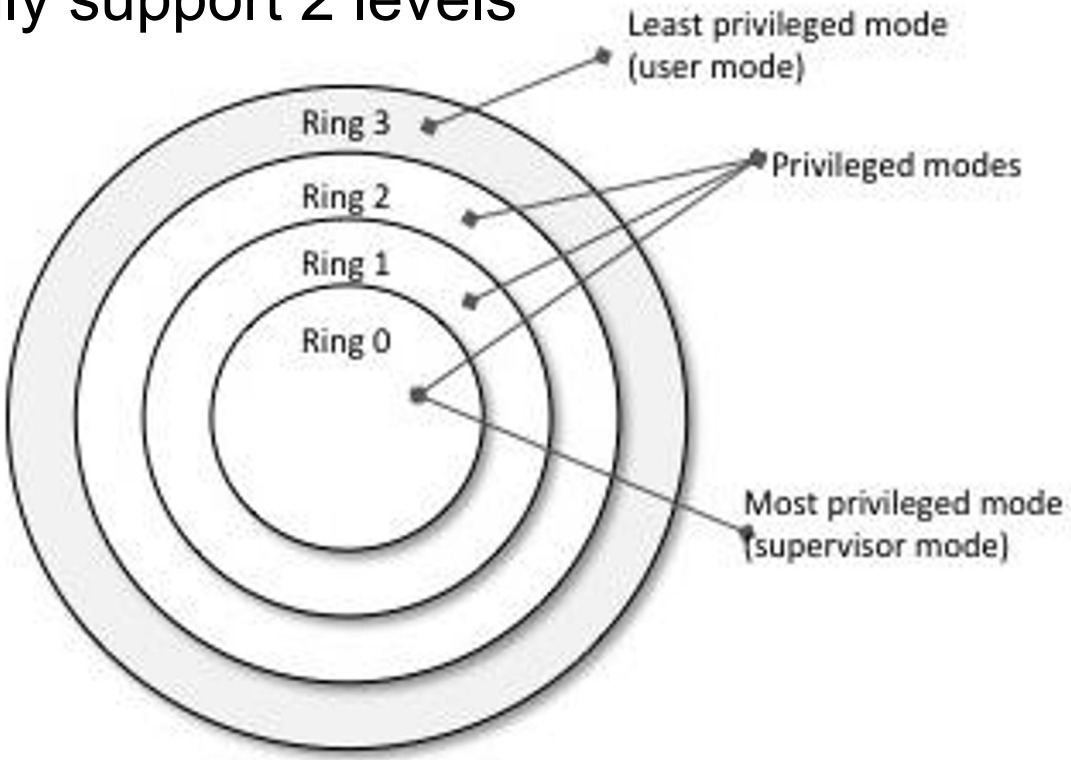
University of CINCINNATI

Most recent systems only support 2 levels

- Use only 2 key levels:
  - Ring 0 - supervisor mode (kernel)
  - Ring 3 - user mode
  - Sensitive instructions cause trap to kernel

- Virtualization adds a hypervisor over Ring 0
  - In reality they run at same level

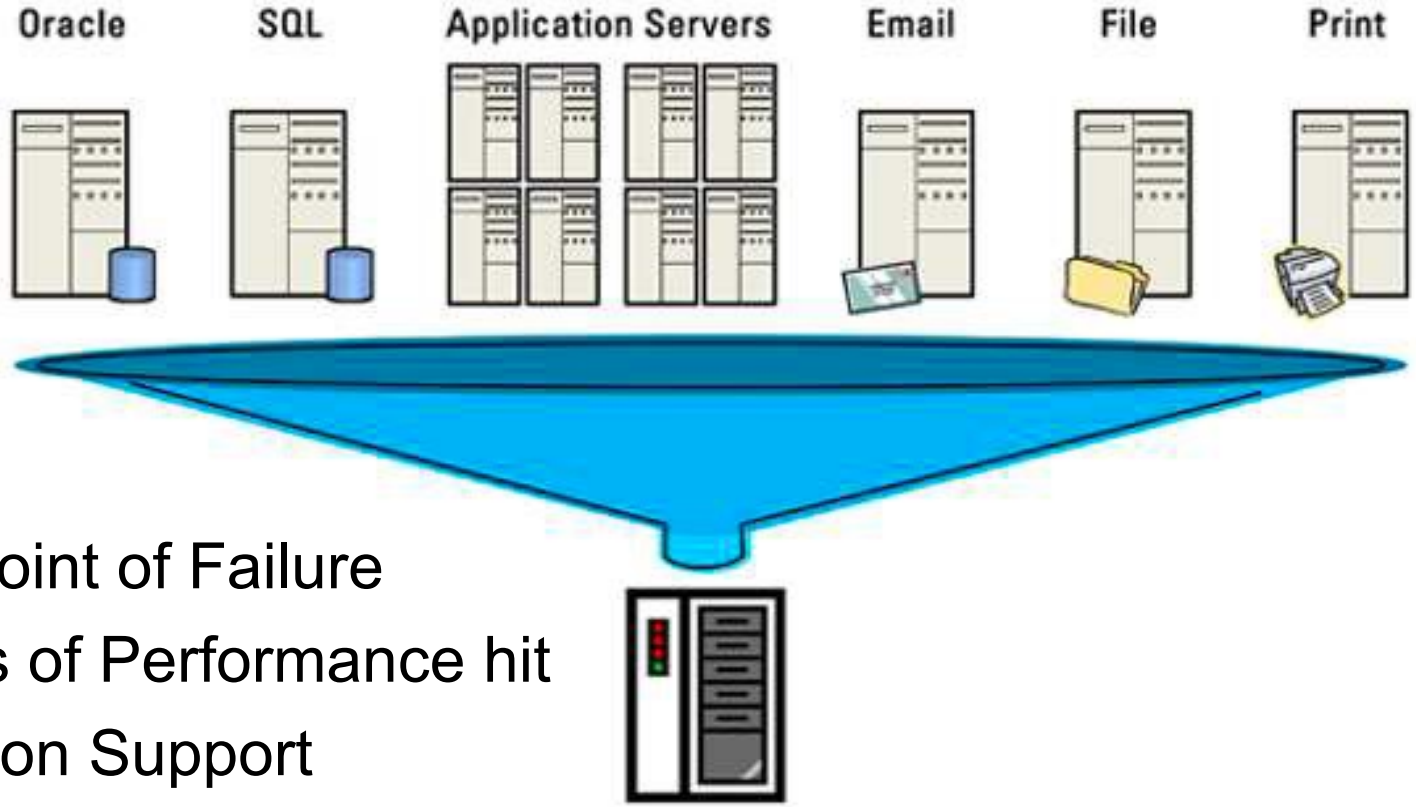
- Isolate different Oss as they all need access to privileged instructions



Level#1: Ring 0 for supervisor mode
Level#2: Ring 3 for user mode

Disadvantages of Virtualization

University of CINCINNATI

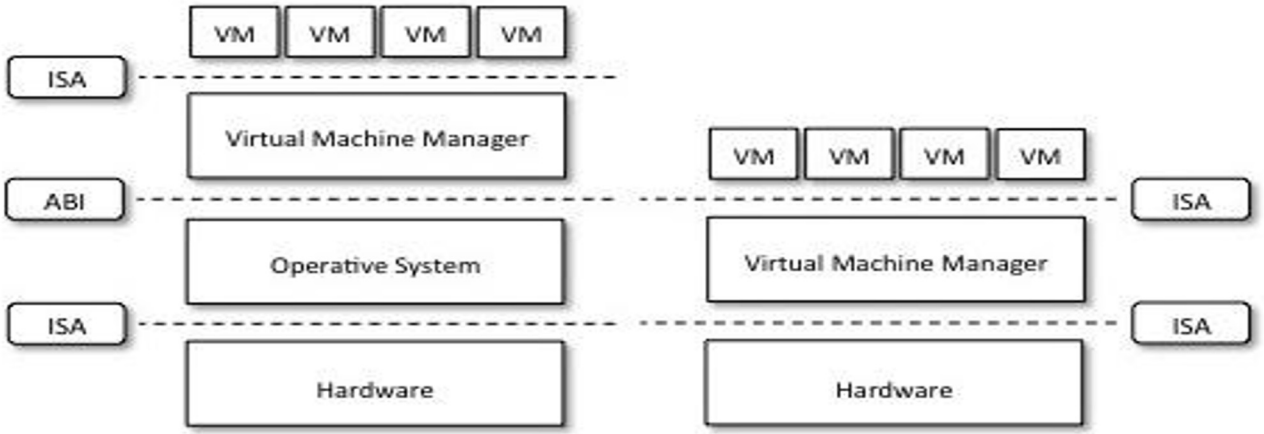


- Single Point of Failure
- Chances of Performance hit
- Application Support

Hardware-level Virtualization

University of CINCINNATI

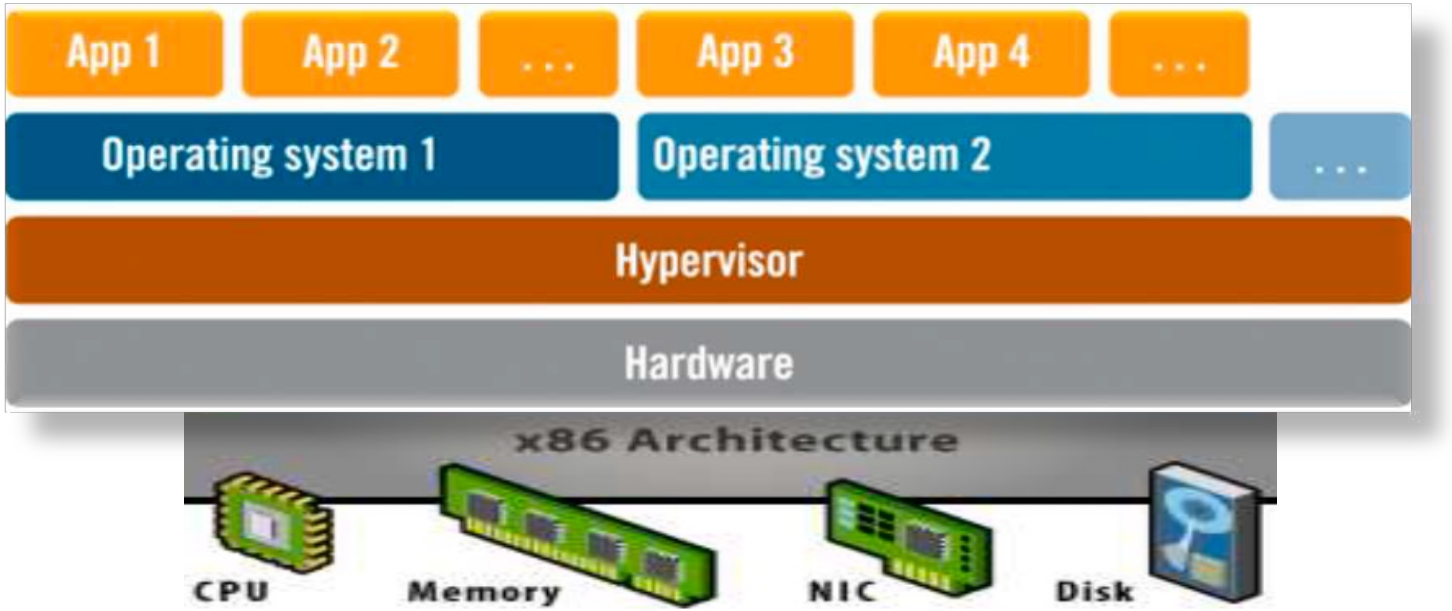
- Virtualizes ISA - system virtualization
- Hypervisors manage system - virtual machine manager (VMM)
  - Type I - Runs directly on hardware, native virtual machine (Bare metal)
  - Type II - Runs in an OS - hosted virtual machine



How Does Virtualization Work ?

University of CINCINNATI

- A Virtualization layer is installed.
  - Uses Bare-metal or Hosted Hypervisor architecture.
- A bare-metal hypervisor system does not require operating system.
  - Hypervisor is operating system.



Types Details

University of CINCINNATI

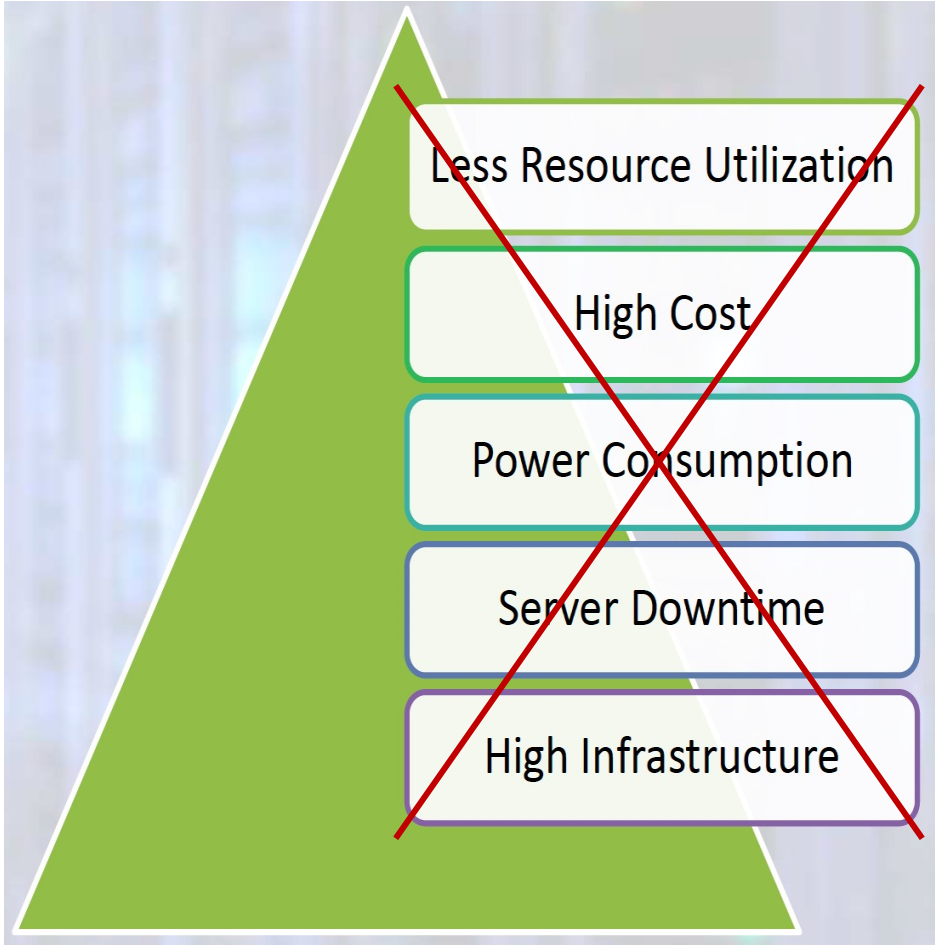
- Type I - Runs directly on hardware - native virtual machine
  - More resource efficient (no OS in the way)
  - Must reinstall ‘OS’
  - ESX/ESXi just mini version of Linux
  - Ex: VMWare ESX/ESXi, MS HyperV
- Type II - Runs in an OS - hosted virtual machine
  - Easier to use, just install the program
  - Ex: VMWare Workstation/Fusion, VirtualBox, Kernel-based Virtual Machine (KVM)

- Types are not definitive! KVM uses virtualization features in the kernel, but can do general purpose work.
- ESX(i) is linux as well and you can run normal programs.

Benefits of Virtualization

University of CINCINNATI

- Typically synonymous with **IaaS**.
- Server **consolidation, hardware cost and performance**.
- **Ease of Management**, Allowing users to have **concurrent** operating systems on **one** computer
- Efficient, Flexible and Scalable usage of storage disks
- **Stable**, **recoverable** and **highly** available solution for storage medium.
- Remove **hardwire** connection between storage hardware and the processor



[Microservices explained - the What, Why and How?]

Challenges with monolithic software

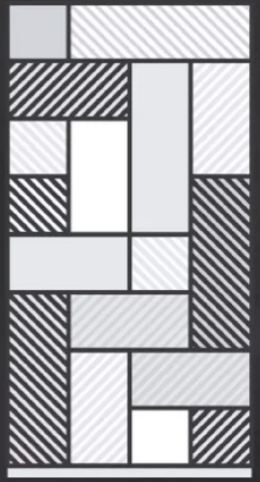
Difficult to scale	Architecture is hard to maintain and evolve	Lack of agility
Long Build/Test/Release Cycles (who broke the build?)	New releases take months	Lack of innovation
Operations is a nightmare (module X is failing, who's the owner?)	Long time to add new features	Frustrated customers

Monolith development lifecycle

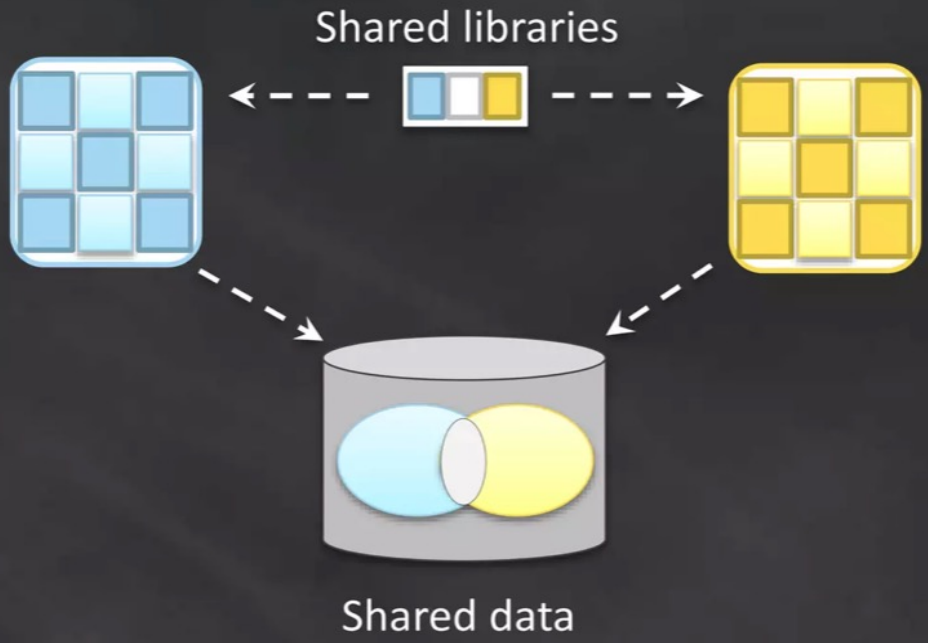
developers

delivery pipeline

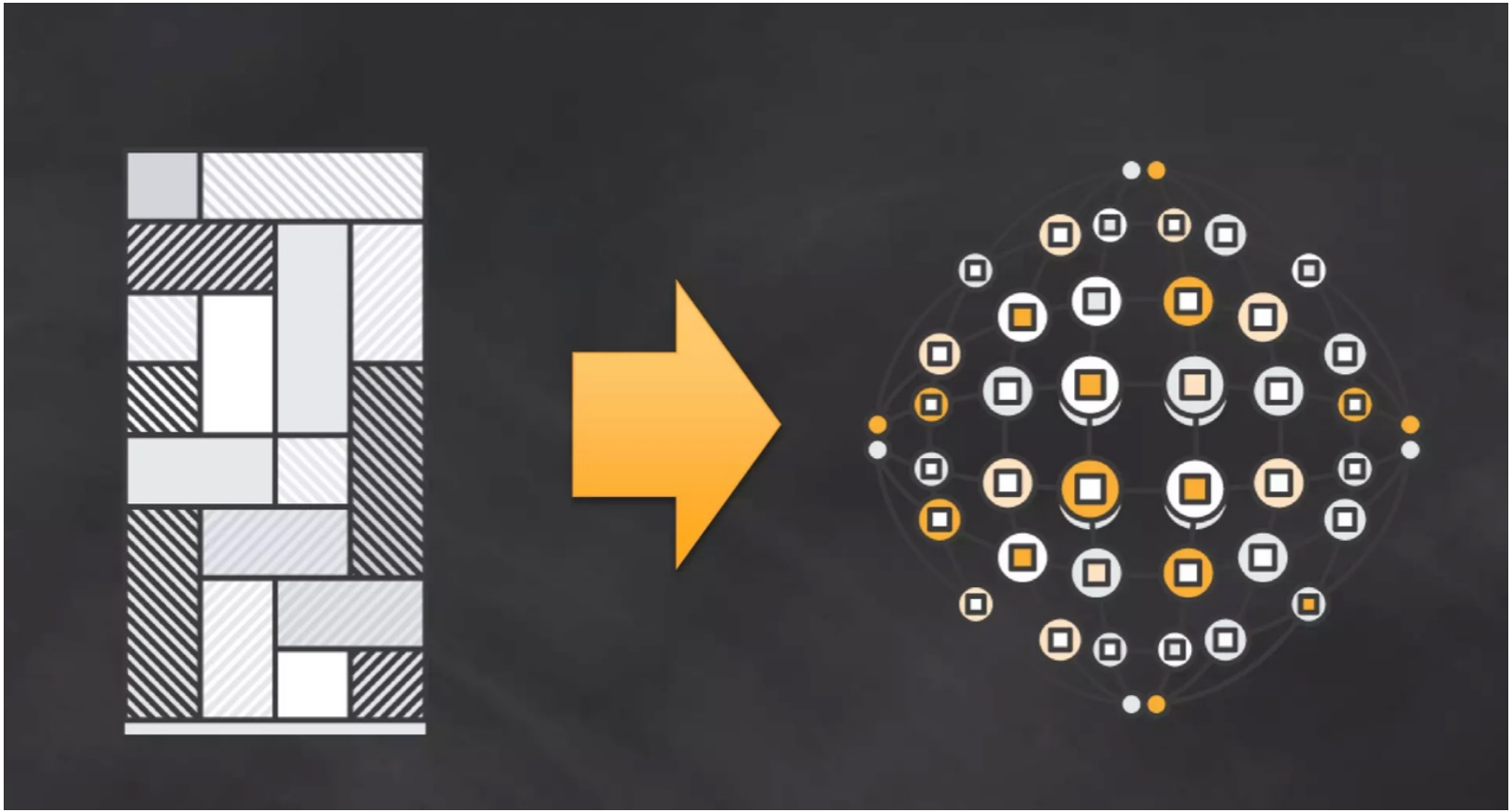
app
(aka the“monolith”)



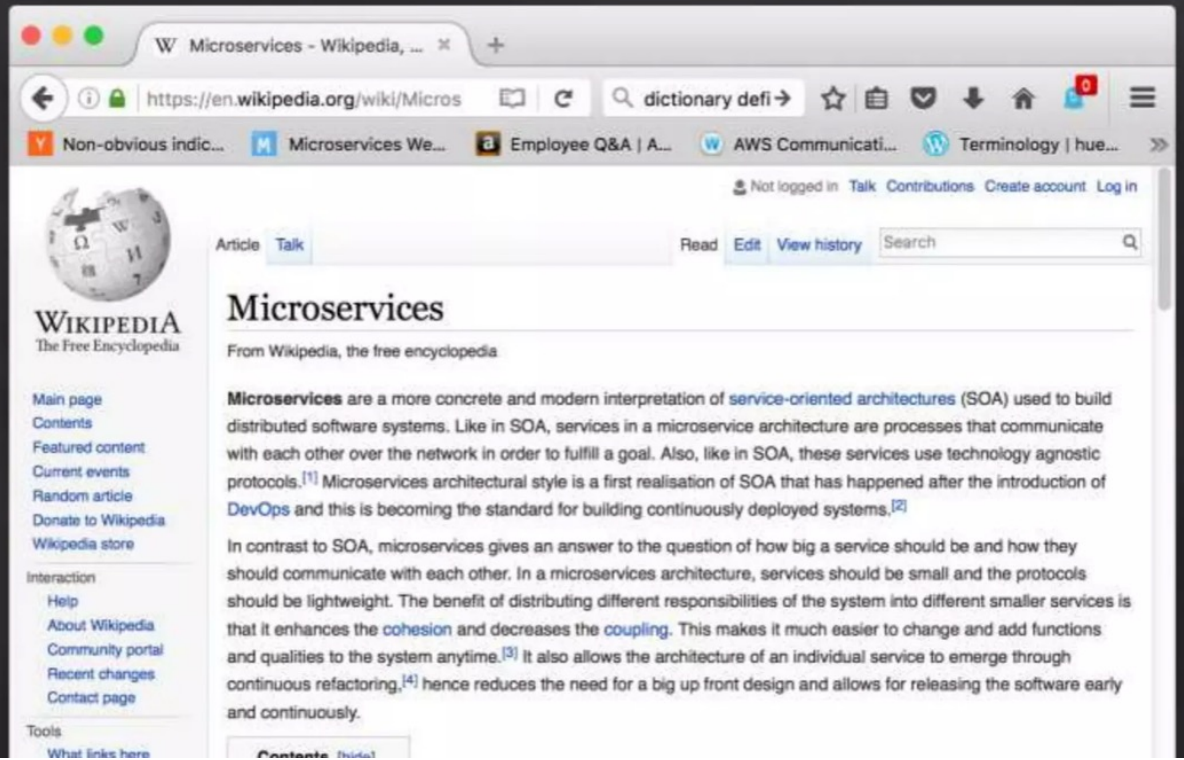
Too much software coupling




CHANGE
FOR THE BETTER



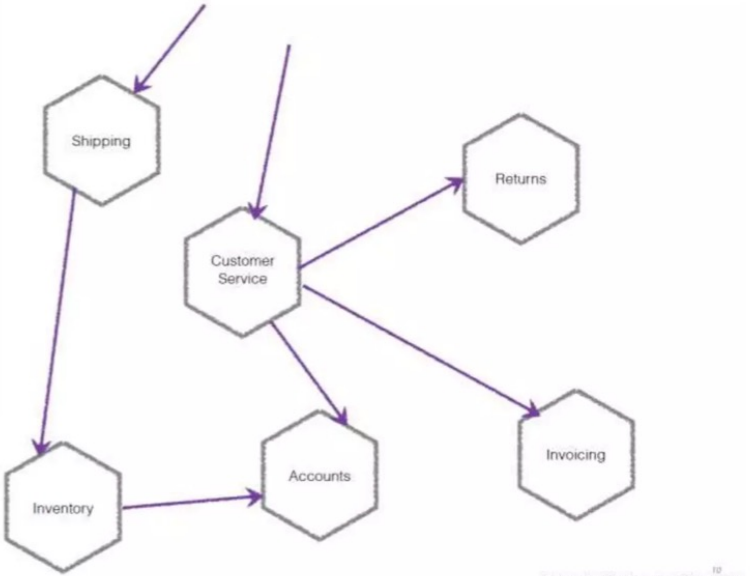
what are microservices



"service-oriented
architecture
composed of
loosely coupled
elements
that have
bounded contexts"

Adrian Cockcroft (former Cloud Architect at Netflix, now Technology Fellow at Battery Ventures)

*Small **independently deployable** services that **work together**, modelled around a **business domain***



SMALL

Size is not the actual point!

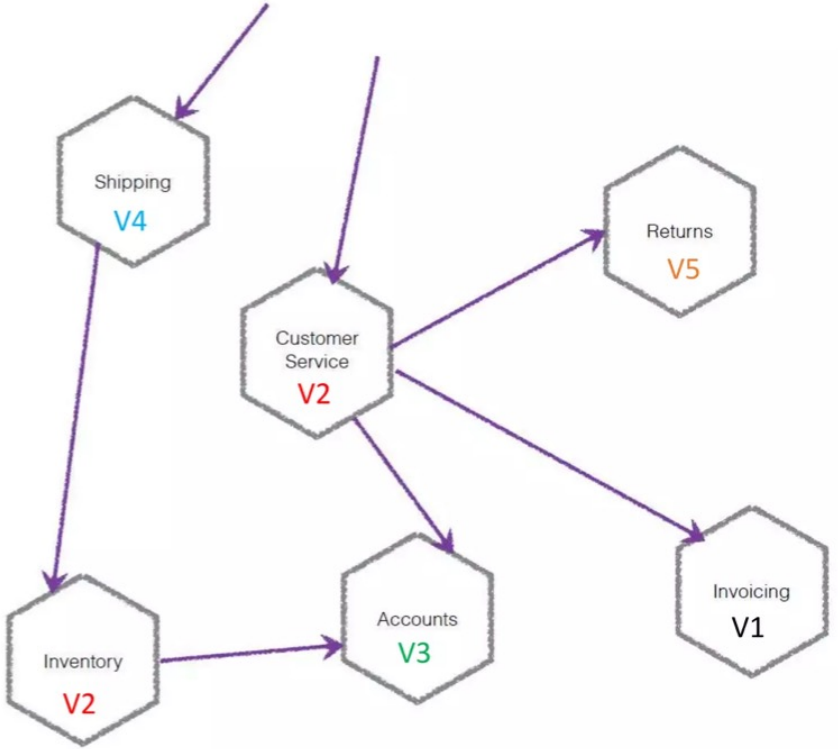
not as big as a server app that needs to be built and deployed as a single block

Manageable units of functionality and deployability

INDEPENDENTLY DEPLOYABLE

No lock-step *build and deployment*

Avoiding the "Distributed Monolith"



"service-oriented architecture composed of loosely coupled elements that have bounded contexts"

Services communicate with each other over the network

Adrian Cockcroft (former Cloud Architect at Netflix, now Technology Fellow at Battery Ventures)

"service-oriented architecture composed of loosely coupled elements that have bounded contexts"

Adrian Cockcroft (former Cloud Architect at Netflix, now Technology Fellow at Battery Ventures)

You can update the services independently; updating one service doesn't require changing any other services.

"service-oriented
architecture
composed of
loosely coupled
elements
that have
bounded contexts"

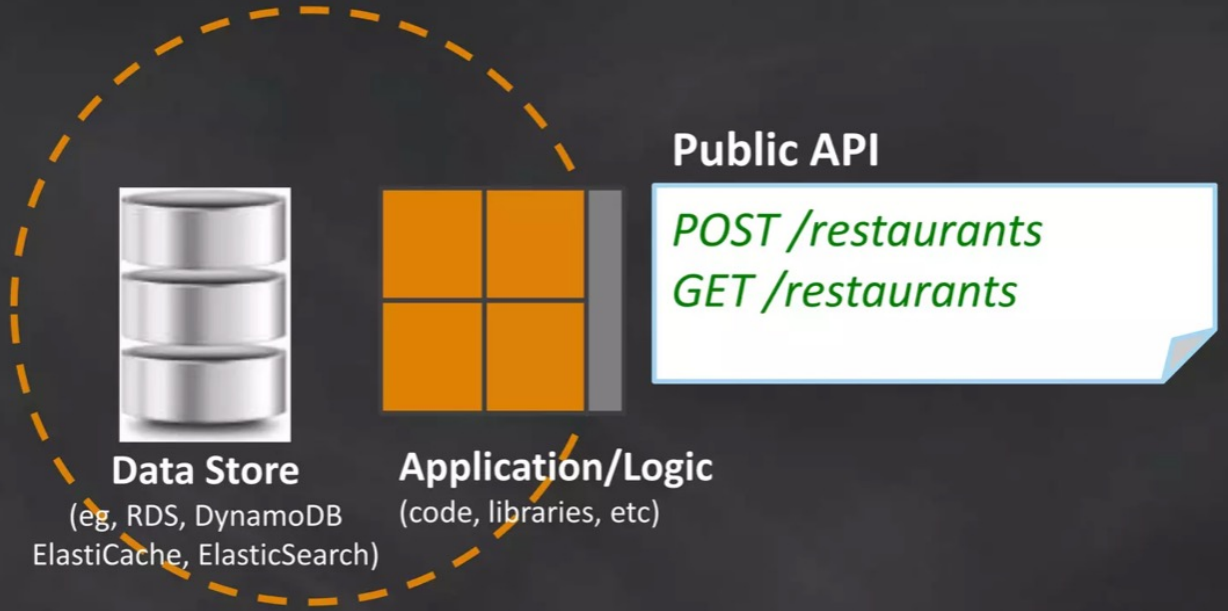
Adrian Cockcroft (former Cloud Architect at Netflix, now Technology Fellow at Battery Ventures)

Self-contained; you can update the code without knowing anything about the internals of other microservices

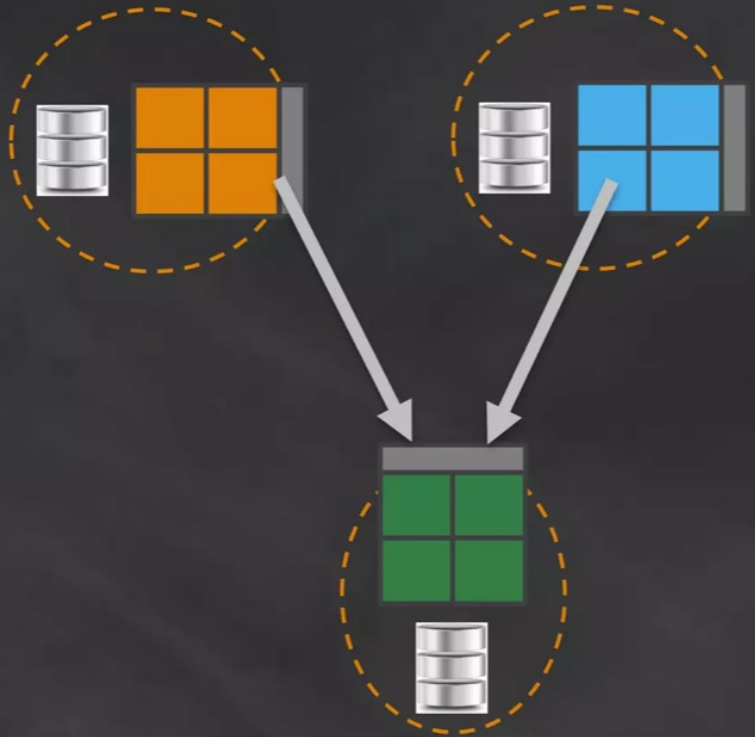
“Do one thing, and do it well”

“Do one thing, and do it well”

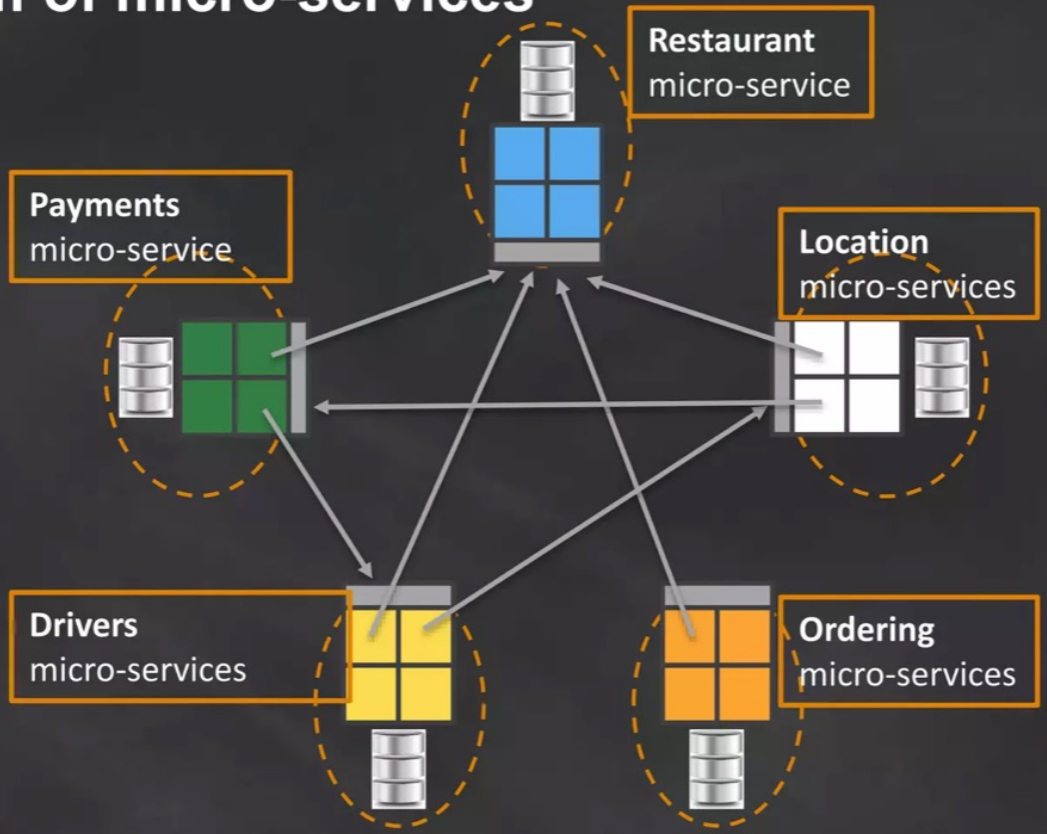
Anatomy of a Micro-service



Avoid Software Coupling



Ecosystem of micro-services



Thousands of teams

× Microservice architecture

× Continuous delivery

× Multiple environments

= 50 million deployments a year

(5708 per hour, or every 0.63 second)

AWS re:Invent

CONSUMER DIGITAL TECH

ARC 308

Nike's Journey to Microservices

Jason Robey, Consumer Digital Technology (CDT), Nike, Inc.

November 12, 2014 | Las Vegas, NV

Getting to production fast

- AMI
- AWS Console
- Conformity Monkey
- Asgard
- Ribbon
- Eureka
- Archaius
- Servo
- ASG

https://api.nike.com/plusv3/domain/context

API Gateway → Legacy Services (Other traffic)

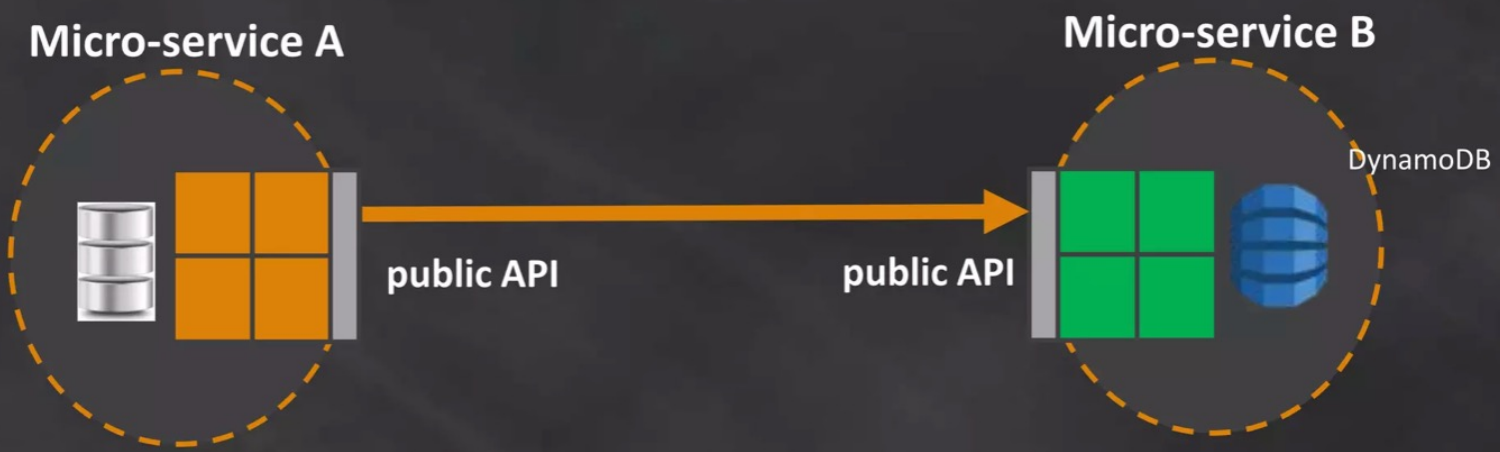
Wildcard Policy → Master Zuul

Three simple classes!

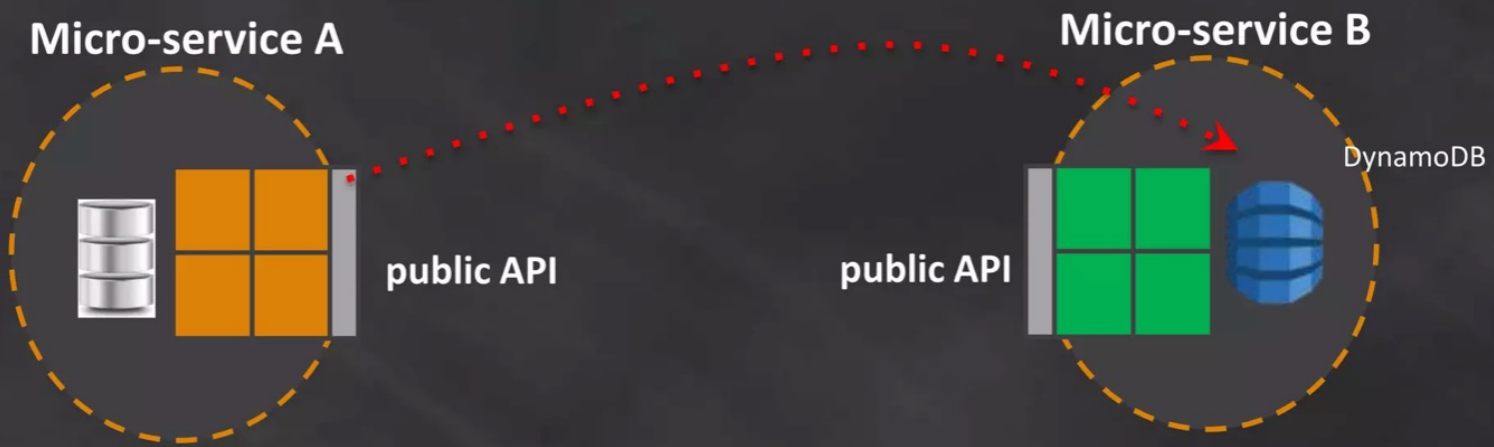
Eureka Lookup → Cloud DNS Space

API Domain Zuul → Context → Micro-services

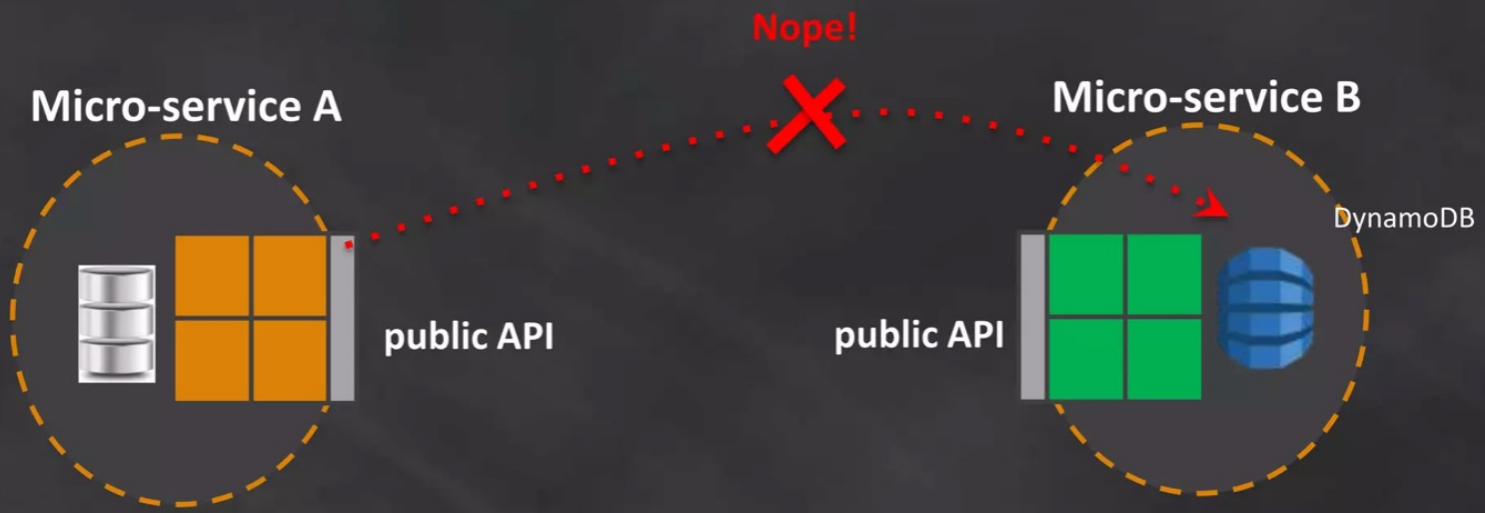
Principle 1: Microservices only rely on each other's public API

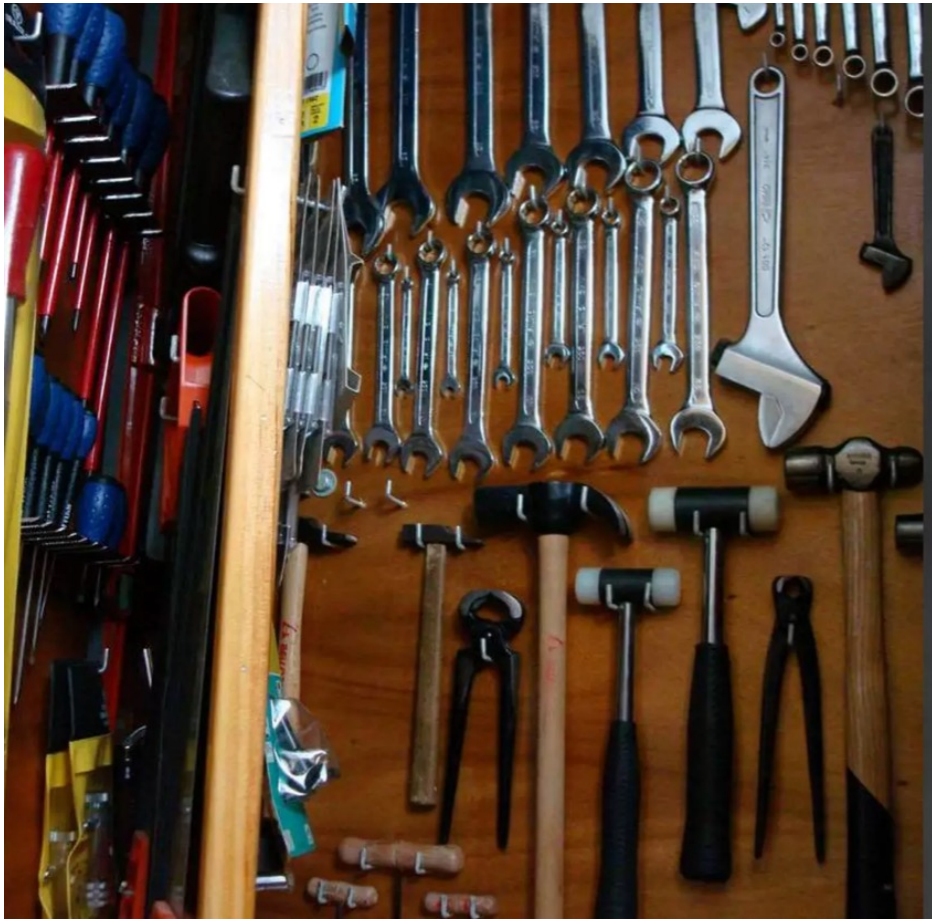


Principle 1: Microservices only rely on each other's public API
(Hide Your Data)



Principle 1: Microservices only rely on each other's public API (Hide Your Data)

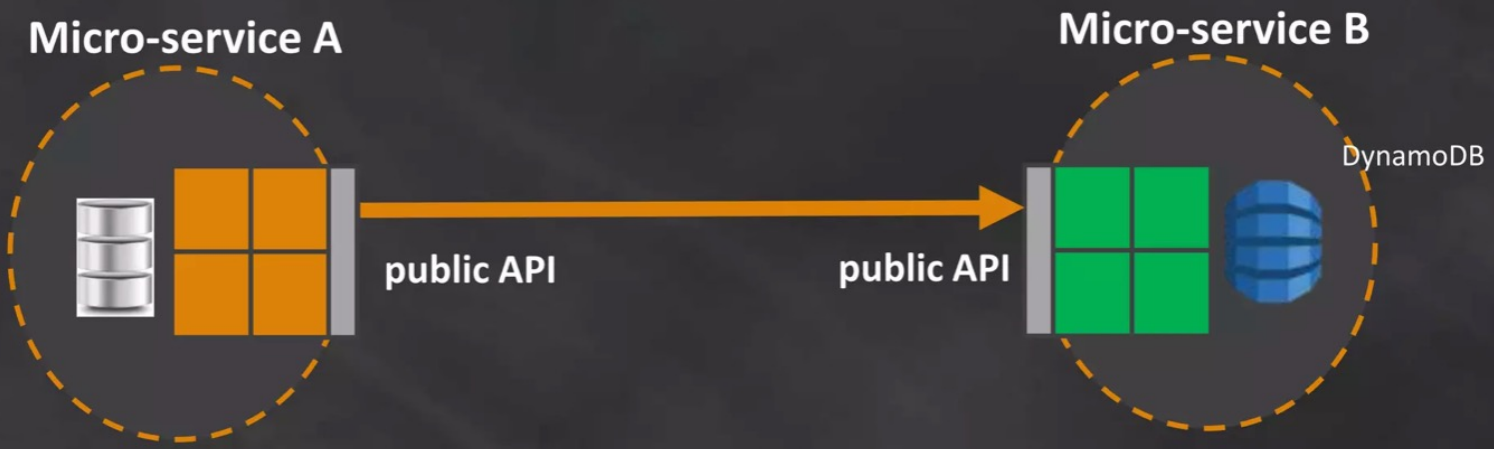




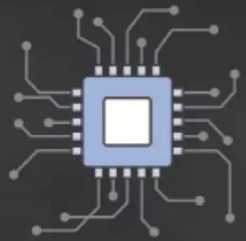
Principle 2

Use the right tool for the job

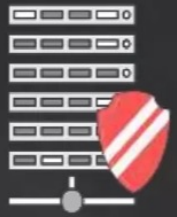
Principle 2: Use the right tool for the job
(Embrace polyglot persistence)



Principle 2: Use the right tool for the job
(Embrace polyglot programming frameworks)



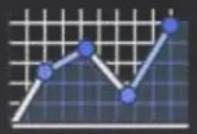
Create a unified API frontend for multiple micro-services



Authenticate and authorize requests

Handles DDoS protection and API throttling

…as well as monitoring, logging, rollbacks, client SDK generation…



Highly Scalable

- Inherently scalable

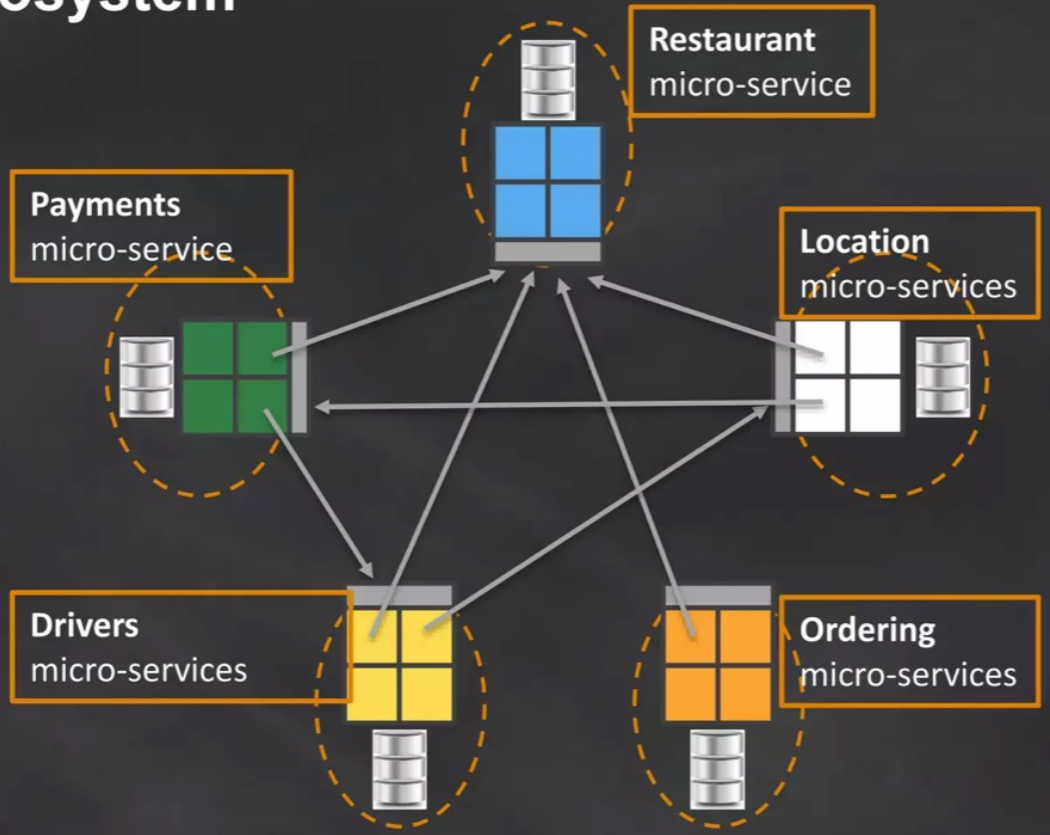
Secure

- API Gateway acts as “front door”
- Can add authN/authZ; or throttle API if needed
- S3 bucket policies
- IAM Roles for Lambda invocations

Cost-efficient

- Only pay for actual microservice usage

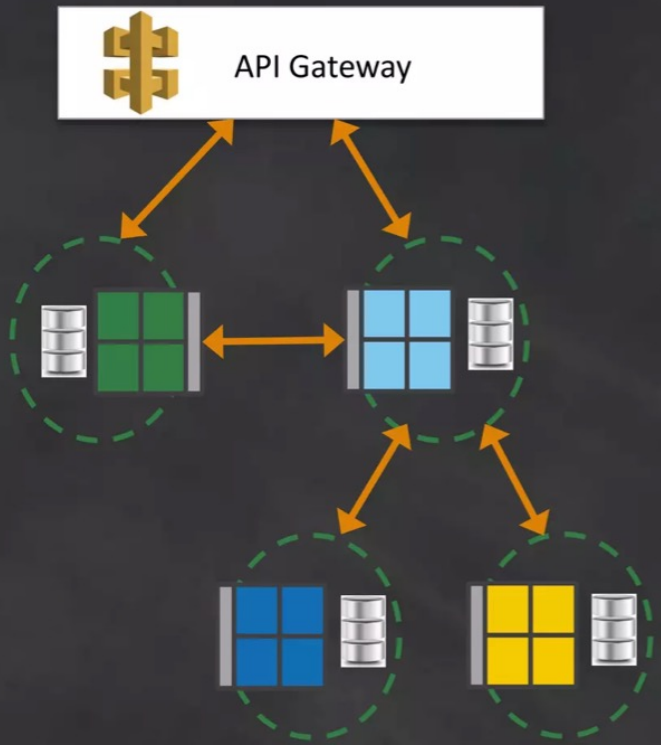
…to an ecosystem



Principle 3

Secure Your Services

Principle 3: Secure Your Services

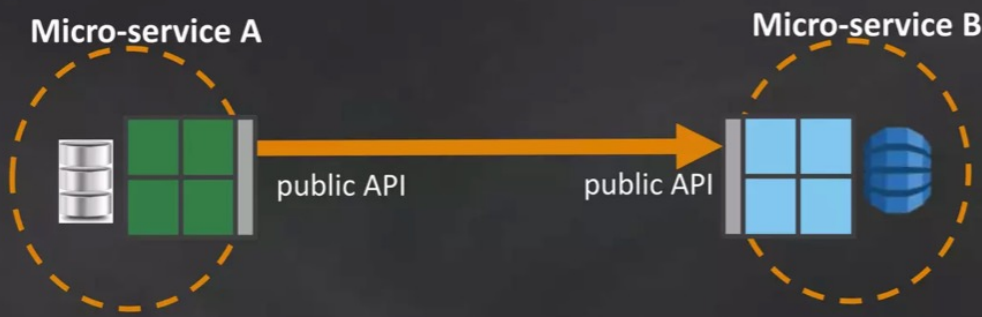


- **Defense-in-depth**
  - Network level (e.g. VPC, Security Groups, TLS)
  - Server/container-level
  - App-level
  - IAM policies
- **Gateway** (“Front door”)
- **API Throttling**
- **Authentication & Authorization**
  - Client-to-service, as well as service-to-service
  - API Gateway: custom Lambda authorizers
  - IAM-based Authentication
  - Token-based auth (JWT tokens, OAuth 2.0)
- **Secrets management**
  - S3 bucket policies + KMS + IAM
  - Open-source tools (e.g. Vault, Keywhiz)

Principle 4

Be a good citizen within the ecosystem

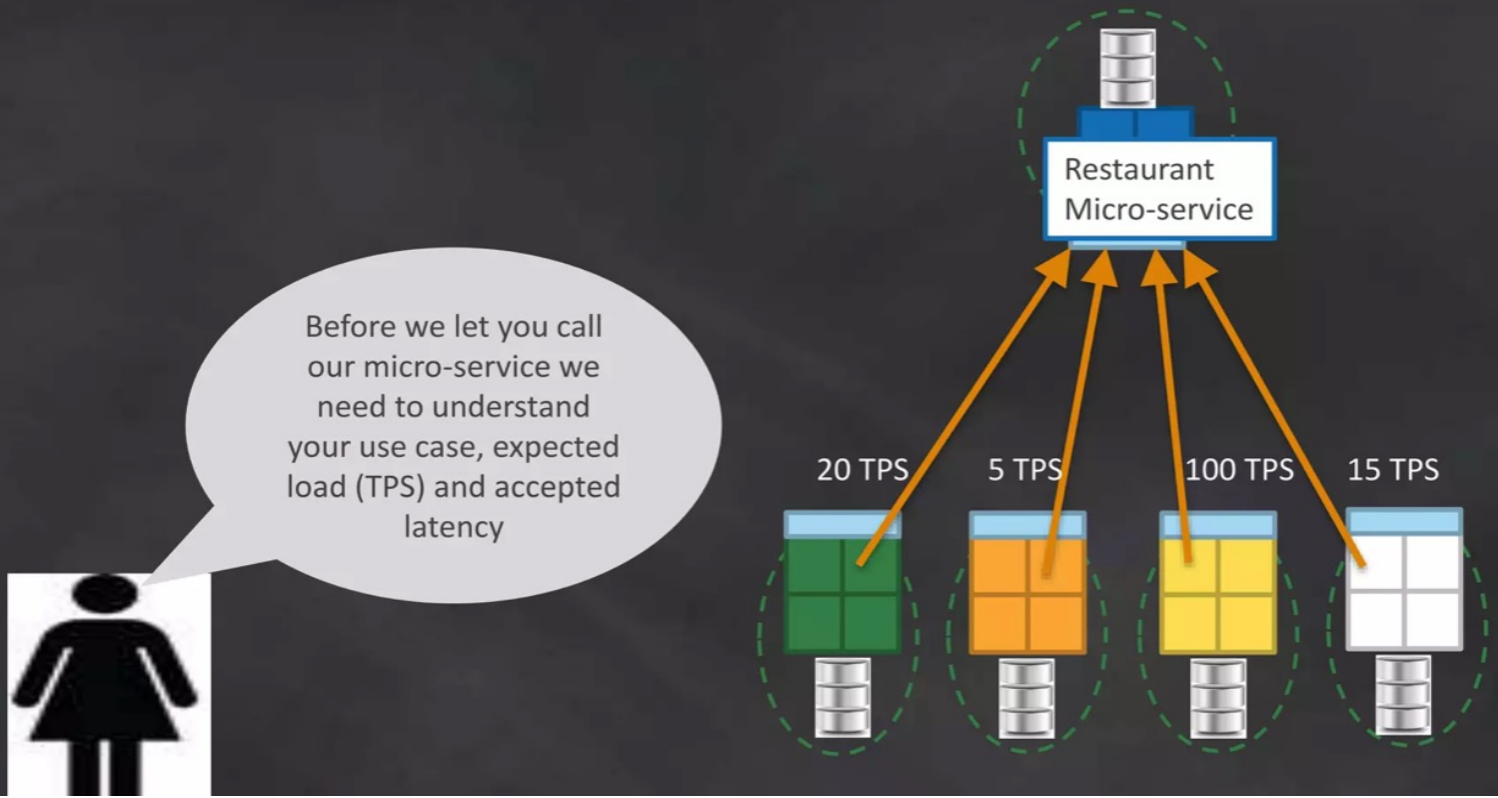
Principle 4: Be a good citizen within the ecosystem



Hey Sally, we need to call your micro-service to fetch restaurants details.

Sure Paul. Which APIs you need to call? Once I know better your use cases I'll give you permission to register your service as a client on our service's directory entry.

Principle 4: Be a good citizen within the ecosystem (Have clear SLAs)



Principle 4: Be a good citizen within the ecosystem
(Distributed monitoring, logging and tracing)

Distributed monitoring and tracing
- "Is the service meeting its SLA?"
- "Which services were involved in a request?"
- "How did downstream dependencies perform?"

Shared metrics
- e.g. request time, time to first byte

Distributed tracing
- e.g. Zipkin, OpenTracing

User-experience metrics

Principle 5

More than just technology transformation

Conway's Law

"Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure."

Melvin E. Conway, 1967

Decentralize governance and data management

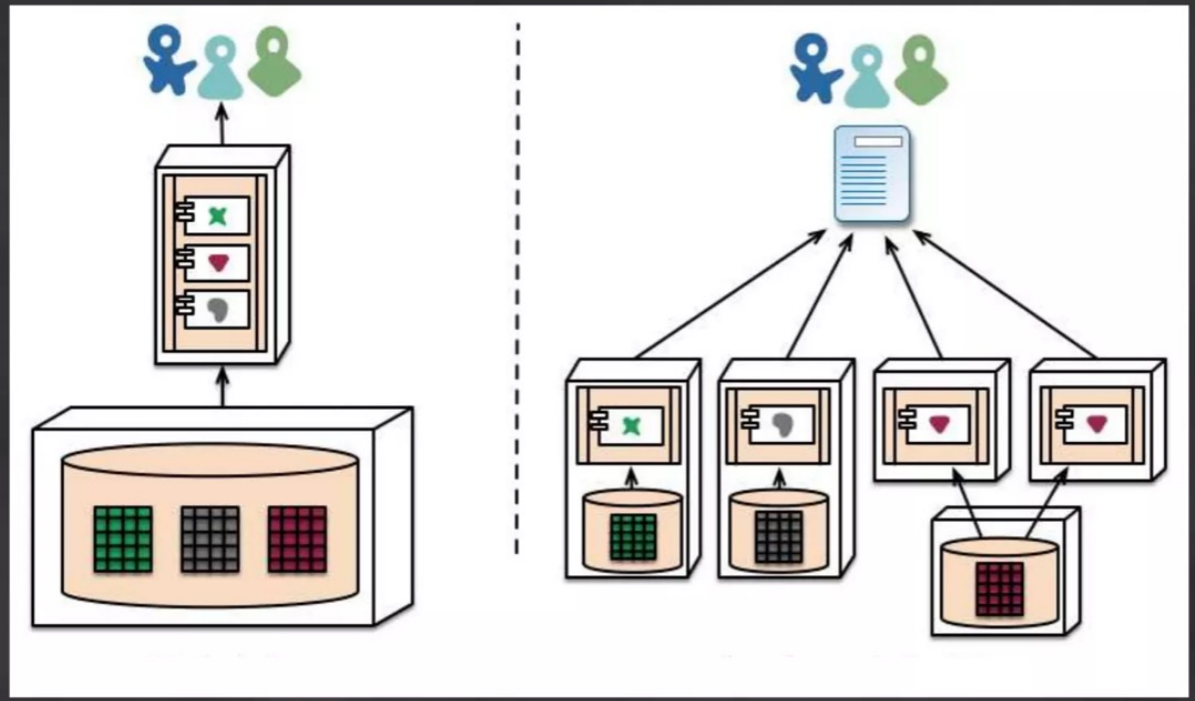
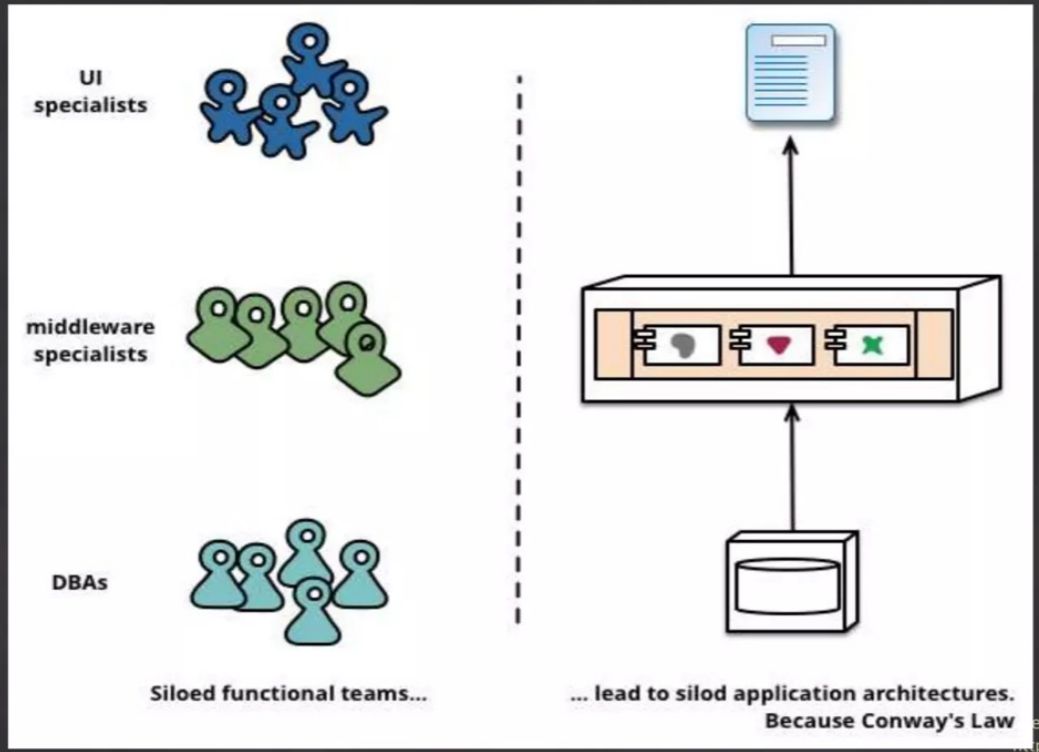


Image from Martin Fowler's article on microservices, at
http://martinfowler.com/articles/microservices.html
No alterations other than cropping.
Permission to reproduce: http://martinfowler.com/faq.html

Silo'd functional teams → silo'd application architectures



e from Martin Fowler's article on microservices, at
tp://martinfowler.com/articles/microservices.html
No alterations other than cropping.
Permission to reproduce: http://martinfowler.com/faq.html

Cross functional teams → self-contained services

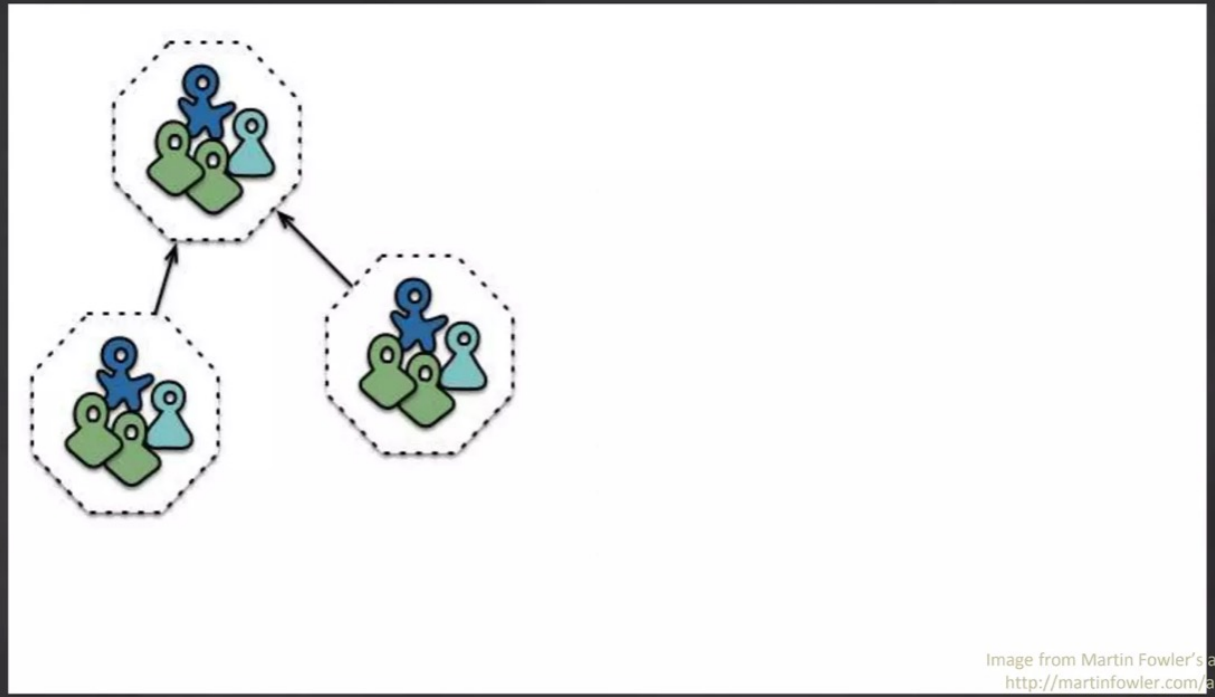


Image from Martin Fowler's article on microservices, at
http://martinfowler.com/articles/microservices.html
No alterations other than cropping.
Permission to reproduce: http://martinfowler.com/faq.html

Cross functional teams → self-contained services

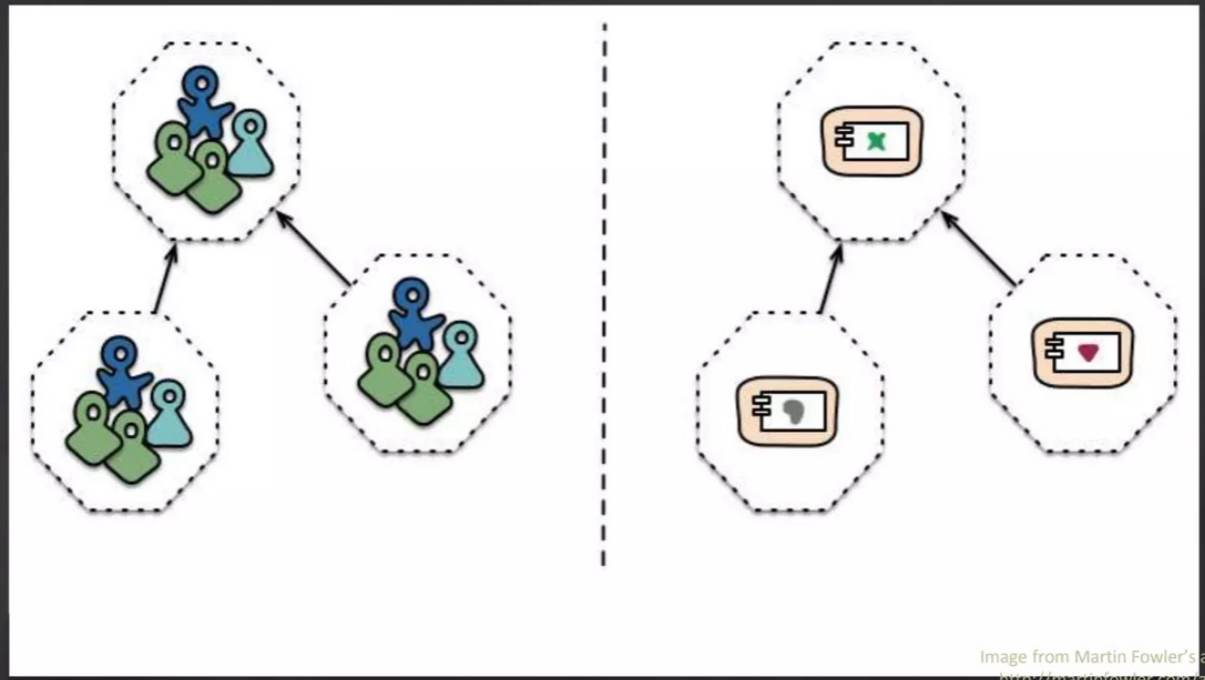


Image from Martin Fowler's article on microservices, at
http://martinfowler.com/articles/microservices.html
No alterations other than cropping.
Permission to reproduce: http://martinfowler.com/faq.html

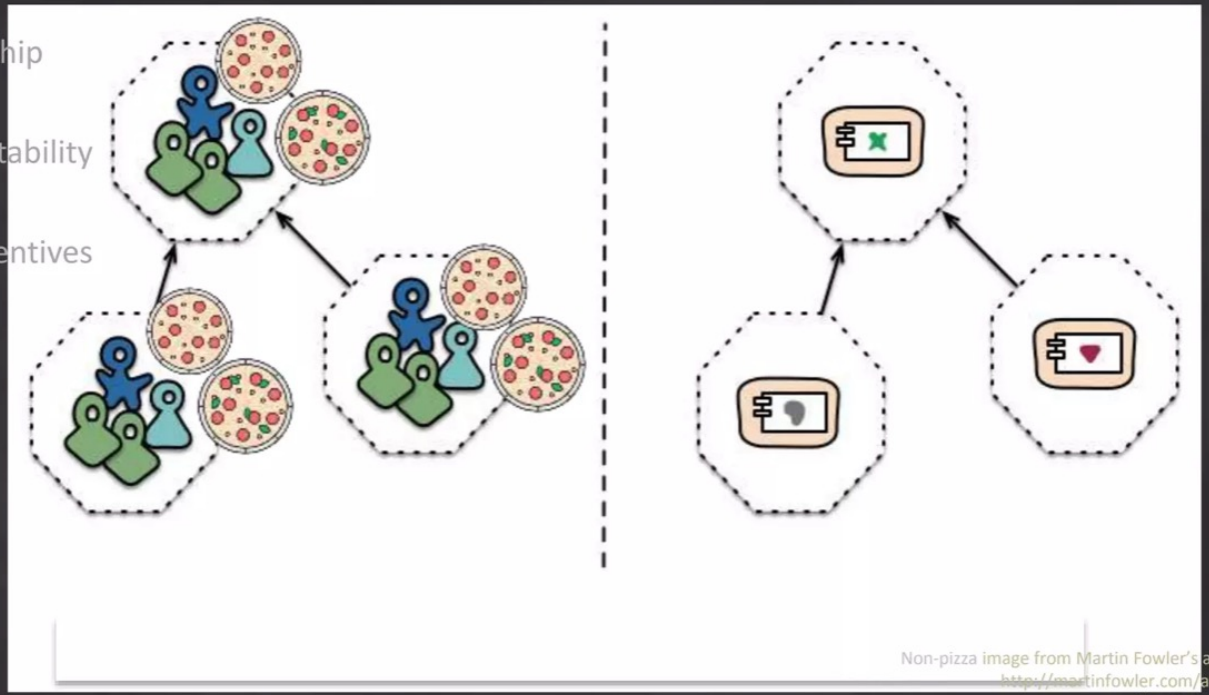
Cross functional teams → self-contained services
("Two-pizza teams" at Amazon)

Full ownership

Full accountability

Aligned incentives

"DevOps"

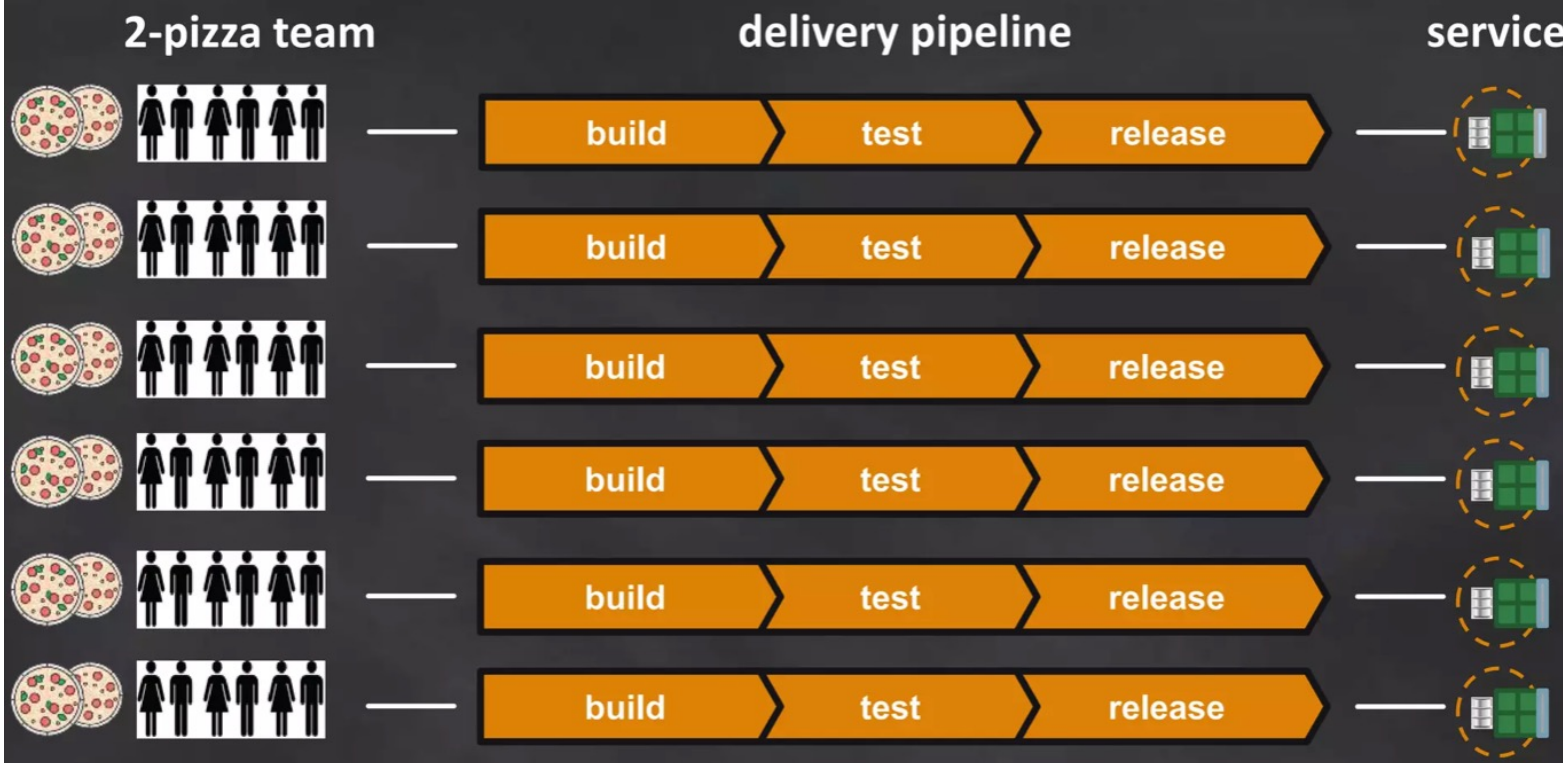


Non-pizza image from Martin Fowler's article on microservices, at http://martinfowler.com/articles/microservices.html
No alterations other than cropping.
Permission to reproduce: http://martinfowler.com/faq.html

Principle 6

Automate Everything

Focused agile teams



Principle 6: Automate everything

AWS
CodeCommit

AWS
CodePipeline

AWS
CodeDeploy

RDS

DynamoDB

ElastiCache

CloudWatch

Cloud Trail

API Gateway

SQS

SWF

EC2

ECS

Lambda

Auto
Scaling

ELB

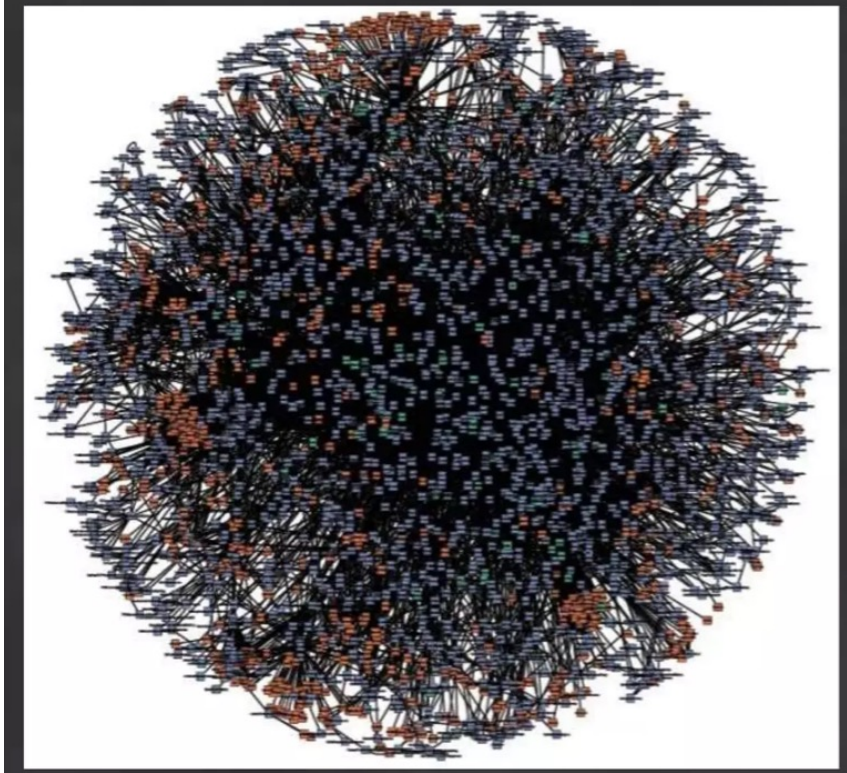
Elastic
Beanstalk

SES

SNS

Kinesis

It’s a journey…



Expect challenges along the way…

- Understanding of business domains
- Coordinating txns across multiple services
- Eventual Consistency
- Service discovery
- Lots of moving parts requires increased coordination
- Complexity of testing / deploying / operating a distributed system
- Cultural transformation

Principles of Microservices

1. Rely only on the public API
   - Hide your data
   - Document your APIs
   - Define a versioning strategy

2. Use the right tool for the job
   - Polygot persistence (data layer)
   - Polyglot frameworks (app layer)

3. Secure your services
   - Defense-in-depth
   - Authentication/authorization

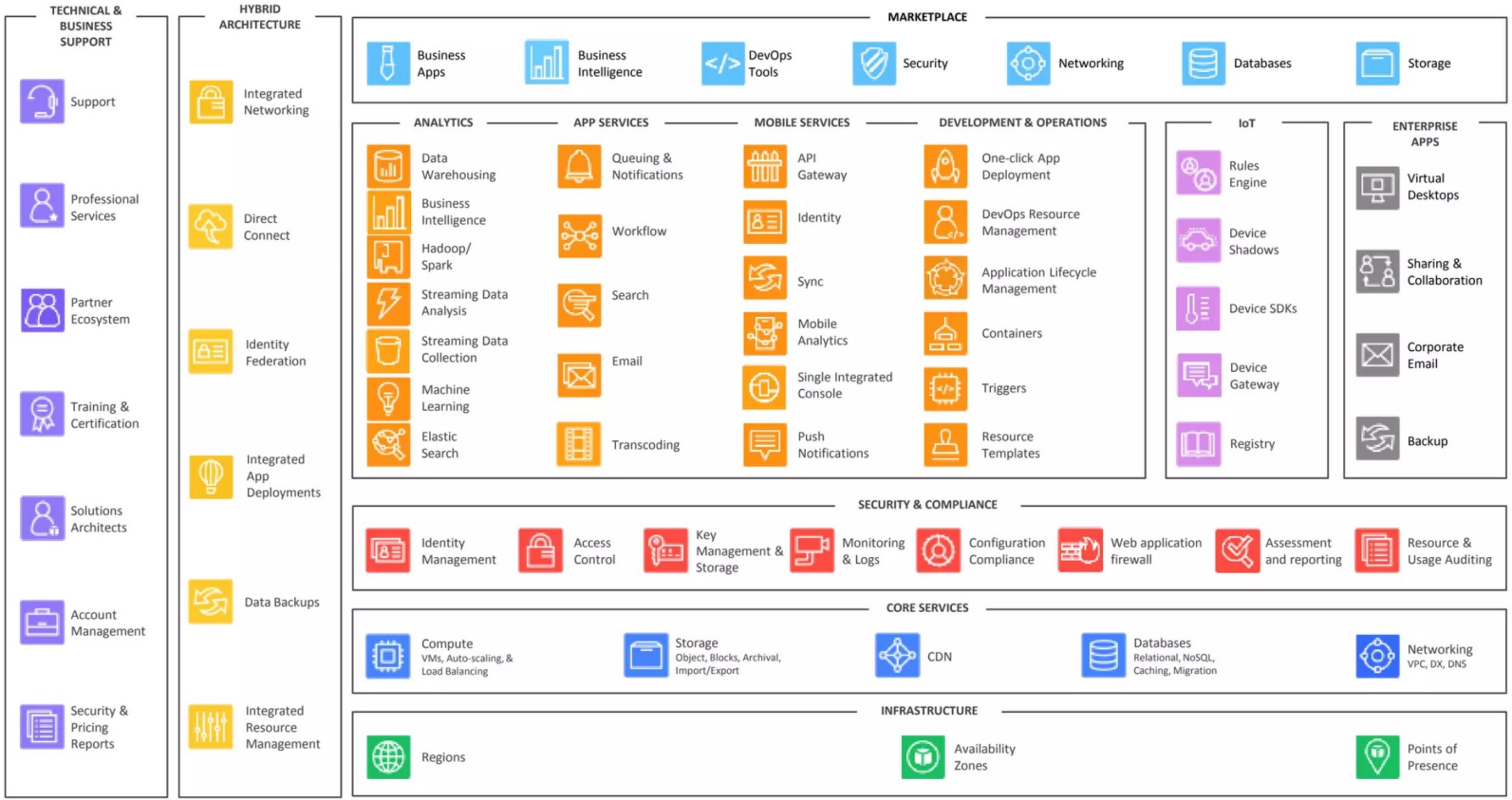
4. Be a good citizen within the ecosystem
   - Have SLAs
   - Distributed monitoring, logging, tracing

5. More than just technology transformation
   - Embrace organizational change
   - Favor small focused dev teams

6. Automate everything
   - Adopt DevOps

Benefits of microservices

- Easier to scale each individual micro-service
- Easier to maintain and evolve system
- Increased agility
- Rapid Build/Test/Release Cycles
- New releases take minutes
- Faster innovation
- Clear ownership and accountability
- Short time to add new features
- Delighted customers


TECHNICAL & BUSINESS SUPPORT
Support
Professional Services
Partner Ecosystem
Training & Certification
Solutions Architects
Account Management
Security & Pricing Reports
HYBRID ARCHITECTURE
Integrated Networking
Direct Connect
Identity Federation
Integrated App Deployments
Data Backups
Integrated Resource Management
MARKETPLACE
Business Apps
Business Intelligence
DevOps Tools
Security
Networking
Databases
Storage
ANALYTICS
Data Warehousing
Business Intelligence
Hadoop/ Spark
Streaming Data Analysis
Streaming Data Collection
Machine Learning
Elastic Search
APP SERVICES
Queuing & Notifications
Workflow
Search
Email
Transcoding
MOBILE SERVICES
API Gateway
Identity
Sync
Mobile Analytics
Single Integrated Console
Push Notifications
DEVELOPMENT & OPERATIONS
One-click App Deployment
DevOps Resource Management
Application Lifecycle Management
Containers
Triggers
Resource Templates
IoT
Rules Engine
Device Shadows
Device SDKs
Device Gateway
Registry
ENTERPRISE APPS
Virtual Desktops
Sharing & Collaboration
Corporate Email
Backup
SECURITY & COMPLIANCE
Identity Management
Access Control
Key Management & Storage
Monitoring & Logs
Configuration Compliance
Web application firewall
Assessment and reporting
Resource & Usage Auditing
CORE SERVICES
Compute
VMs, Auto-scaling, & Load Balancing
Storage
Object, Blocks, Archival, Import/Export
CDN
Databases
Relational, NoSQL, Caching, Migration
Networking
VPC, DX, DNS
INFRASTRUCTURE
Regions
Availability Zones
Points of Presence

Microservices explained - the What, Why and How?

TechWorld with Nana