# ECE566
# Enterprise Storage Architecture

# Spring 2025

## Cloud

Tyler Bletsch
Duke University

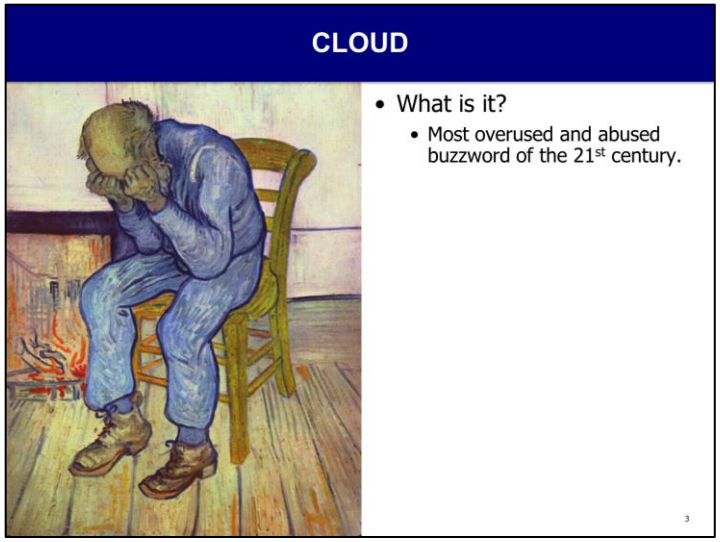
Includes material adapted from the course "Information Storage and Management v2" (module 13), published by EMC corporation.

**Meta-notes**

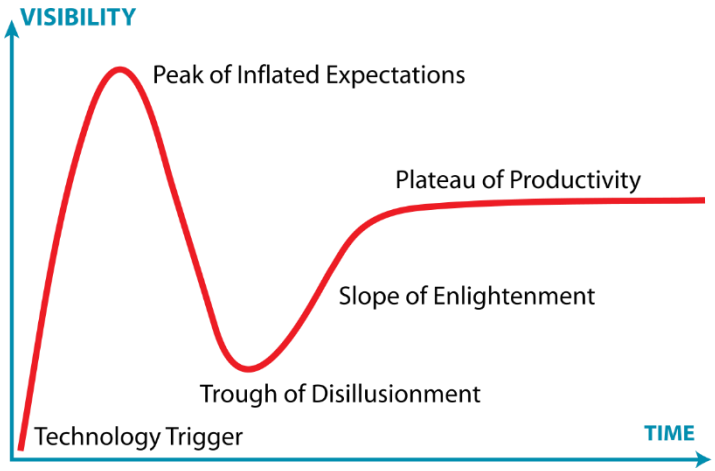
Notes I've added to the EMC stuff will appear in boxes like this one.

# Cloud: How we got here

- For most of the time I taught this course, cloud was the hottest topic ever (like "AI" is today)
- We finally calmed down, and use cloud services as a mundane tool
- This journey is illustrative of the industry in general

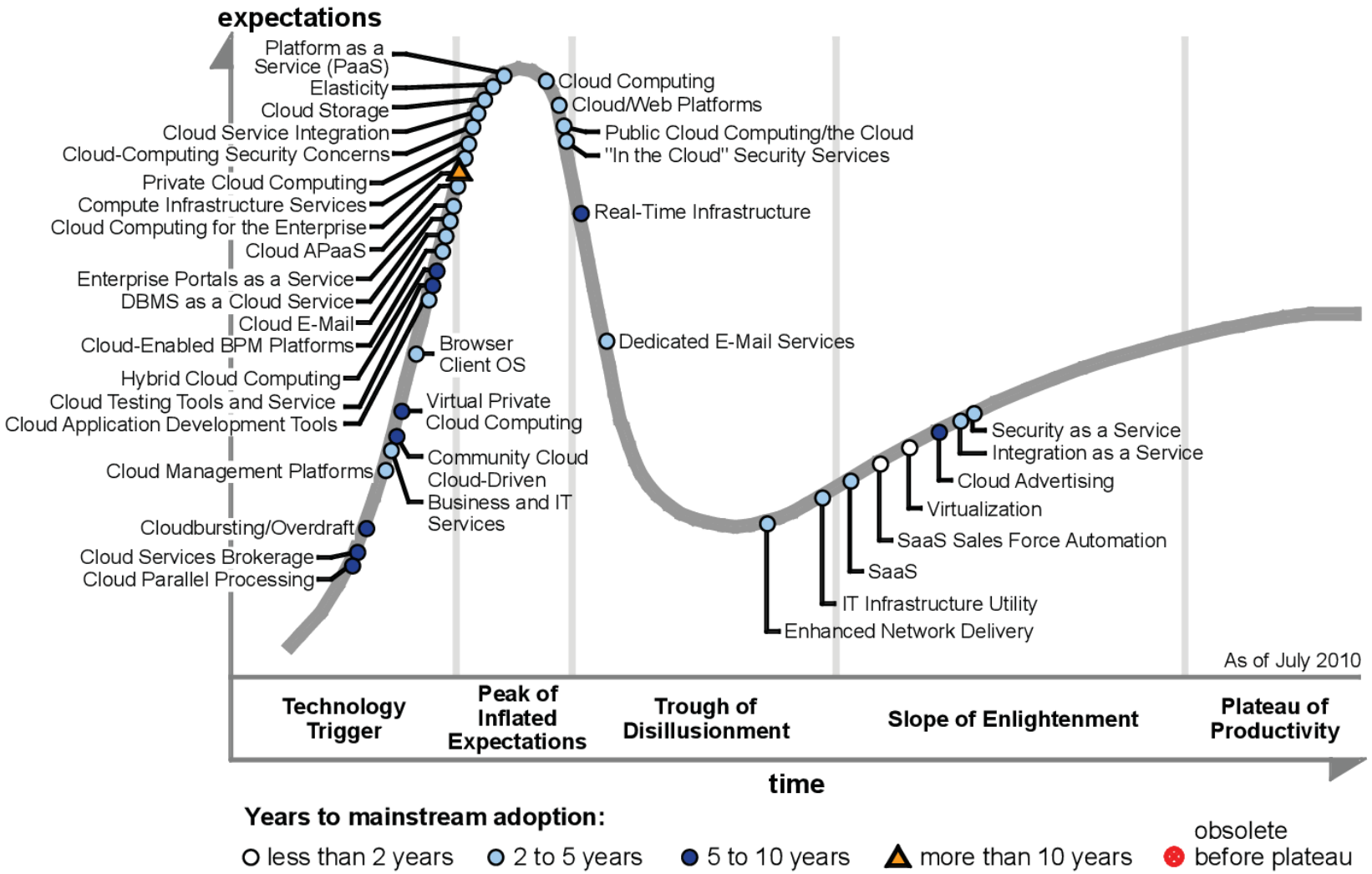



The old intro slide to this deck



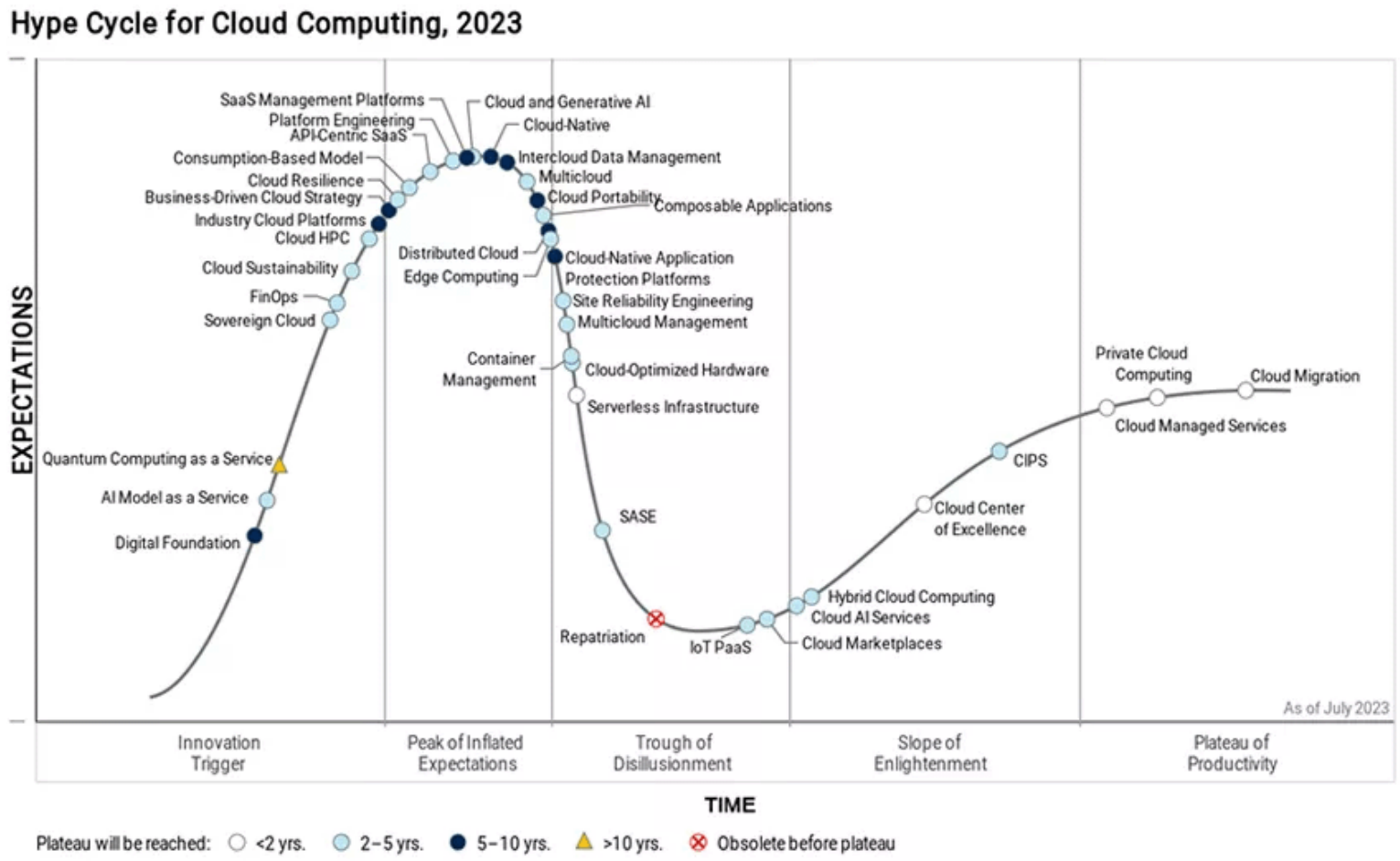

The Gartner "hype cycle" attempts to describe almost all technology rollouts

# 2010: Cloud Mania

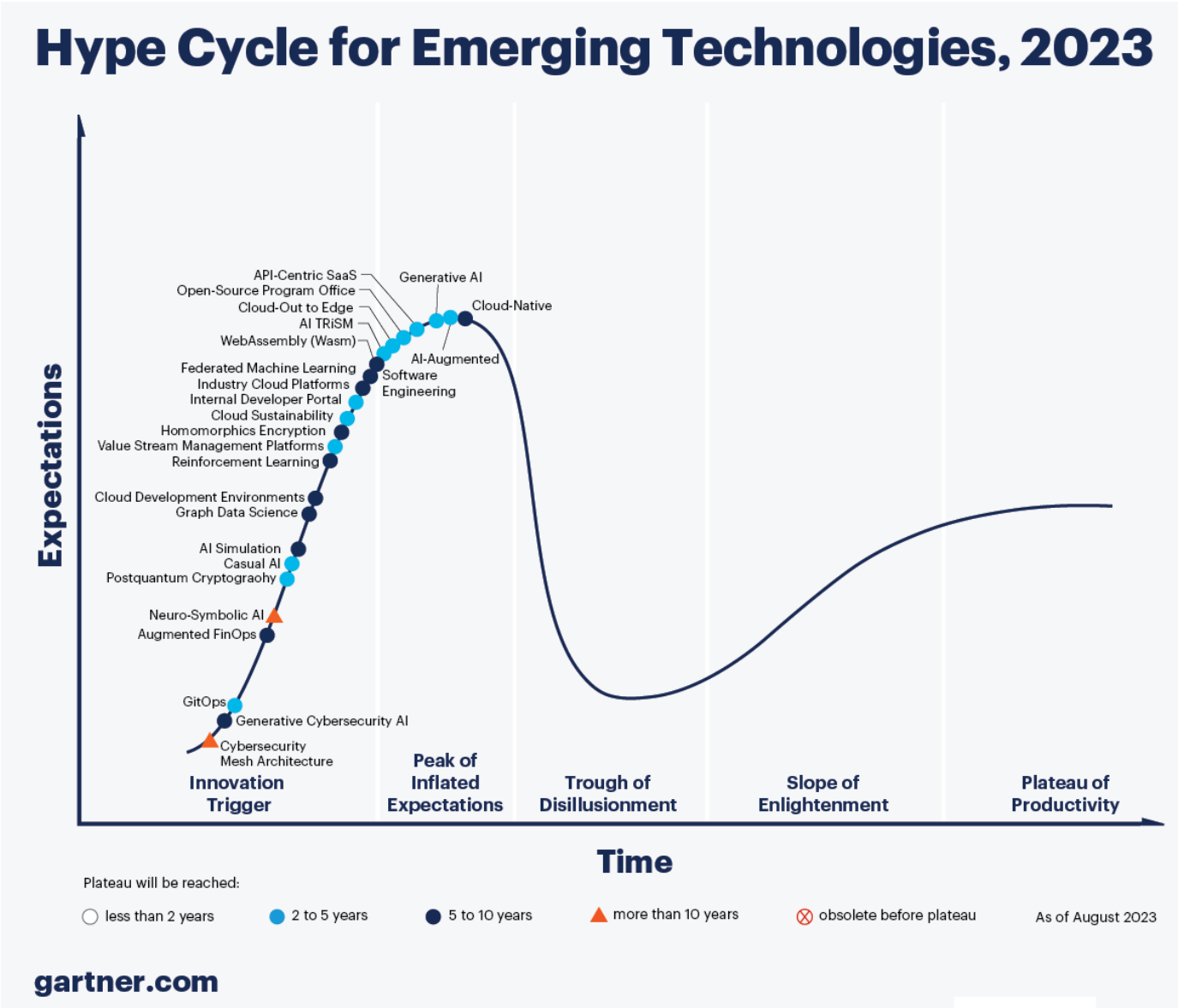



Source: Gartner (July 2010)

# 2023: Cloud has its own hype cycle



# 2023: General hype cycle



# Cloud

- What is it?
  - It's when you borrow a computer over a network.
  - That's all.

- Lots of ways to "borrow".
- Lots of kinds of "computer".
- Lots of kinds of "network".

- Marketing nonsense was so bad the National Institute of Standards and Technology (NIST) produced a definition which most people go by now

# Why do cloud?

- So you don't have to buy the stuff.
  - Save time and up-front costs.
- So you don't have to maintain the stuff.
  - Avoid recurring effort and unpredictable expenses (or rather, pay to make them someone else's problem)
- So you can rent more stuff only when you need to.
  - The "Christmas season" effect – only rent servers to handle 100M hits/day when you actually might get 100M hits in a day.

# Why care about infrastructure at all then?

- **Question**: If cloud hides all this "infrastructure" stuff, why study it? Why have this course?

- **Answer**: Because someone still has to do the work!
  - We just moved where most of it happens:
    individual companies → cloud providers

- Infrastructure skills from this course are in high demand for cloud providers!

# What is Cloud Computing?

**Cloud Computing**

A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., servers, storage, networks, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

– NIST

- Essential Cloud characteristics
  - On-demand self-service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured service

# On-demand Self-service


- Enables consumers to unilaterally provision computing capabilities (examples: server time and storage capacity) as needed automatically
- Consumers view service catalogue via a Web-based user interface and use it to request for a service

**Translation:**

Press button → Receive resource

# Broad Network Access


- Computing capabilities are available over the network
- Computing capabilities are accessed from a broad range of client platforms such as:
  - Desktop computer
  - Laptop
  - Tablet
  - Mobile device

**Translation:**

Leasing a physical server doesn't count.

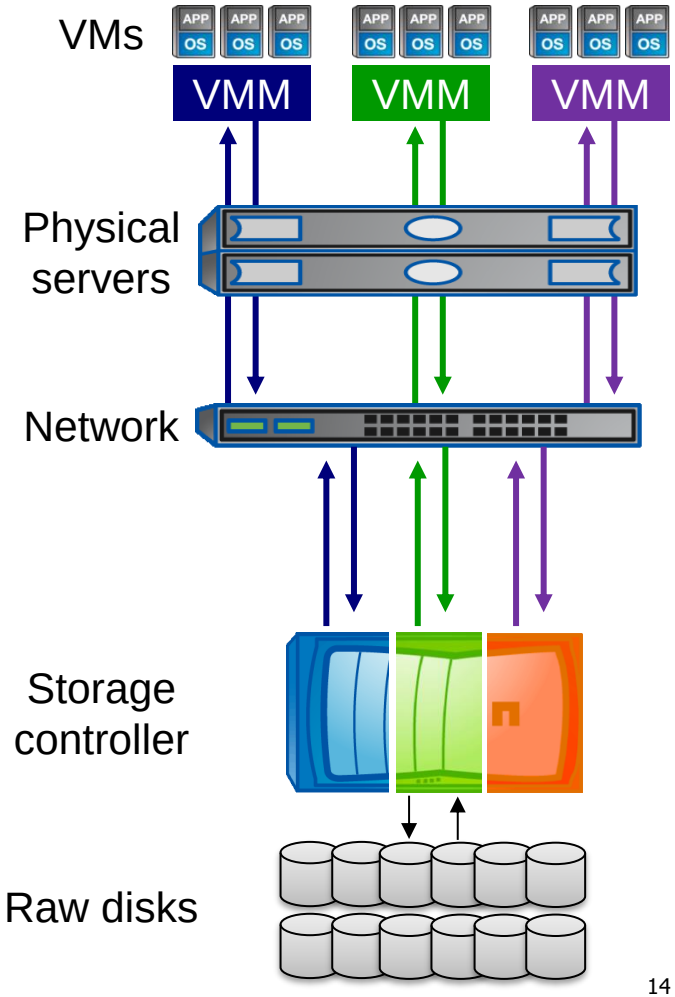
  

# Resource Pooling

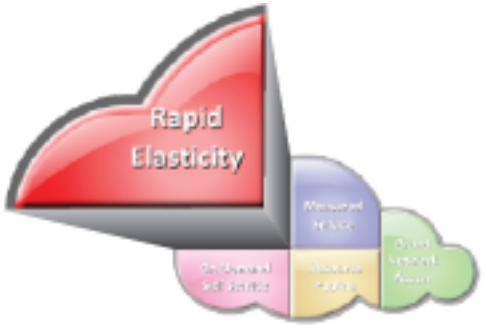

- Provider's computing resources are pooled to serve multiple consumers using a multitenant model
- Resources are assigned from the pool according to consumer demand
- Consumers have no control or knowledge over the exact location of the provided resources

# “Resource pooling”?

- What are some architectures/technologies that pool resources?
  - RAID array and LVM pools raw disks
  - NAS/SAN in general pools storage
  - Server virtualization pools compute
  - End-to-end virtual environment pools them all at once!



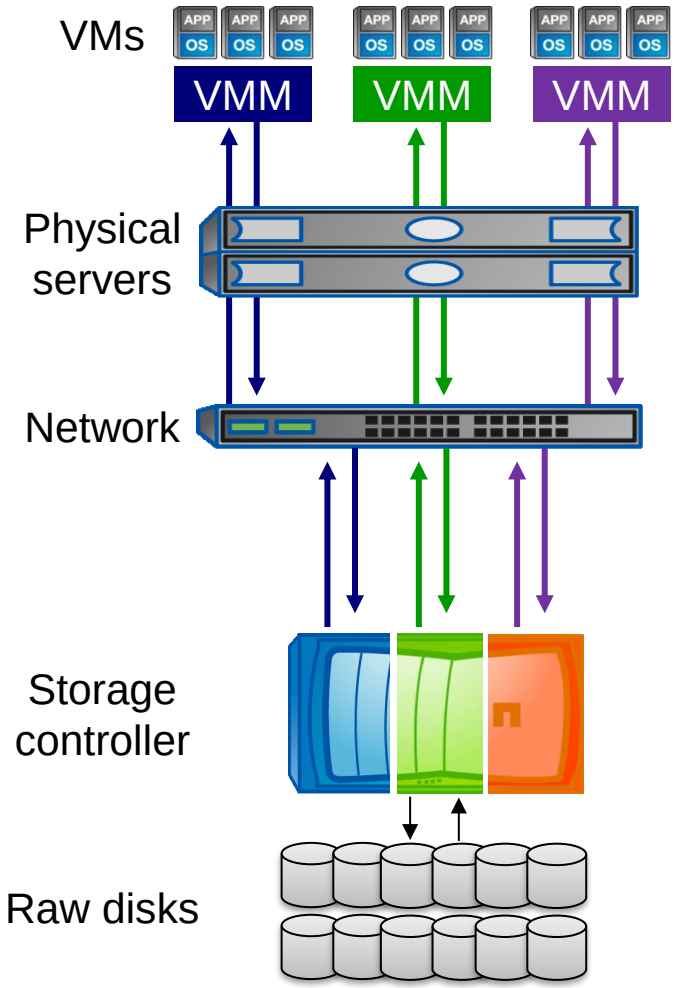
# Rapid Elasticity




- Computing capabilities can be elastically provisioned and released
- Computing capabilities are scaled rapidly, commensurate with consumer's demand
  - Provides a sense of unlimited scalability

# Rapid elasticity

- How can we scale each layer of this stack?



# Measured Service


- Cloud computing provides a metering system that continuously monitors resource consumption and generates reports
  - Helps to control and optimize resource use
  - Helps to generate billing and chargeback reports

**Translation:**

We watch what you use and, if this is a pay-for-use cloud, charge you for it.

# Benefits of Cloud Computing

| Benefits | Description |
|---|---|
| Reduced IT cost | • Reduces the up-front capital expenditure (CAPEX) |
| Business agility | • Provides the ability to deploy new resources quickly<br>• Enables businesses to reduce time-to-market |
| Flexible scaling | • Enables consumers to scale up, scale down, scale out, or scale in the demand for computing resources easily<br>• Consumers can unilaterally and automatically scale computing resources |
| High availability | • Ensures resource availability at varying levels, depending on consumer's policy and priority |

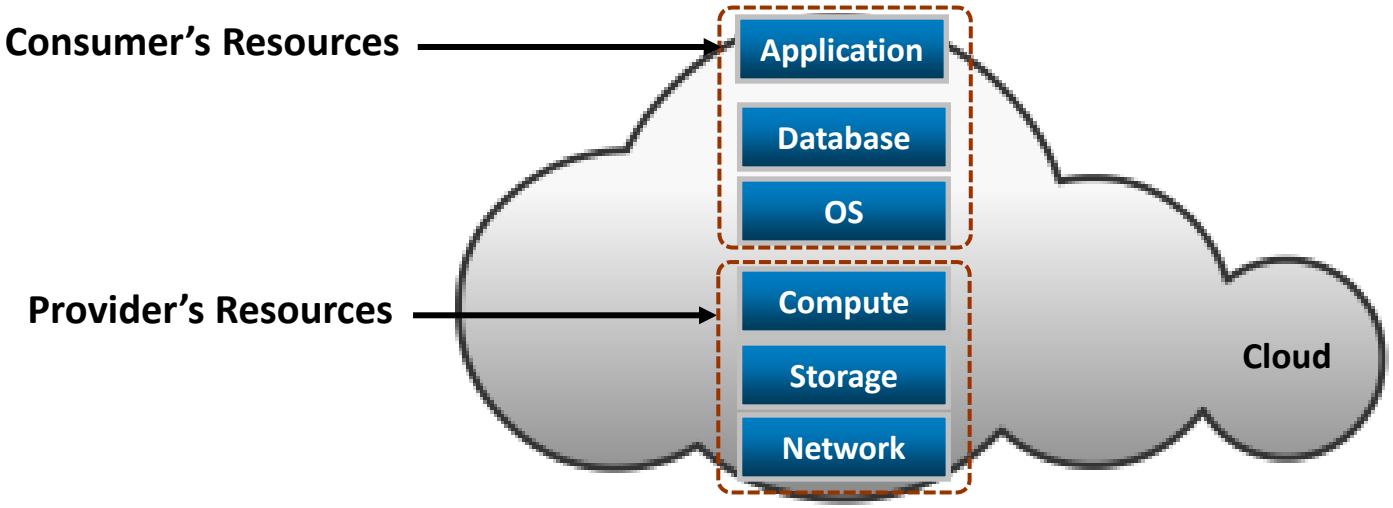
# Cloud Service Models

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

- Storage-as-a-Service (StaaS)
- Tons of other stuff -as-a-Service (XaaS)

# Infrastructure-as-a-Service

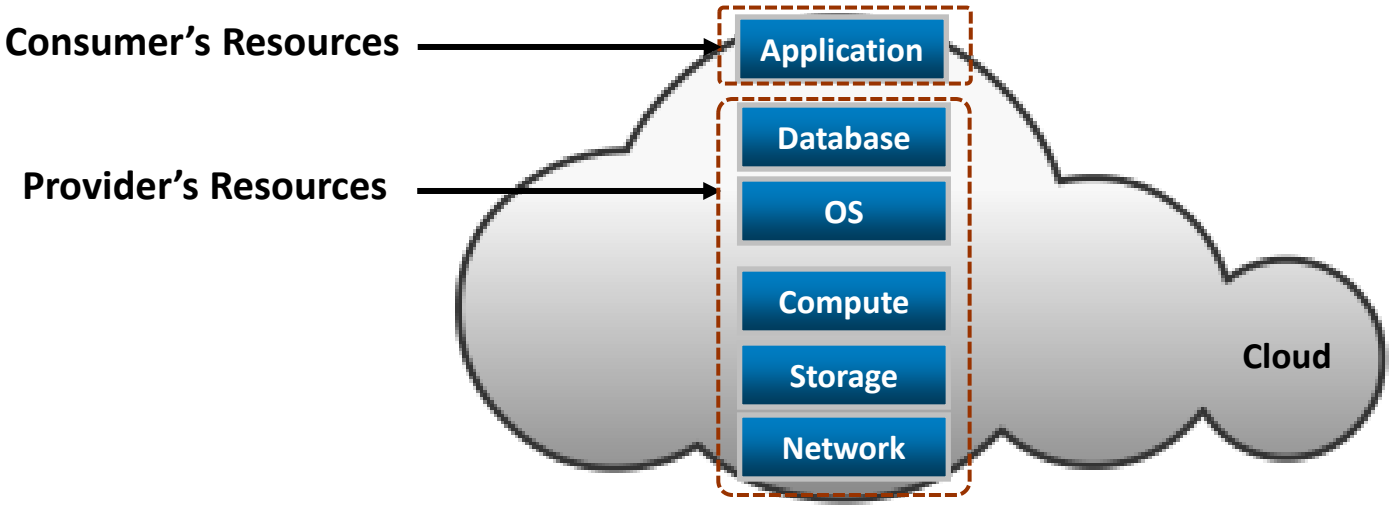
- Consumers deploy their software, including OS and application on provider's infrastructure
  - Computing resources such as processing power, memory, storage, and networking components are offered as service
  - Example: Amazon Elastic Compute Cloud
- Consumers have control over the OSs and deployed applications



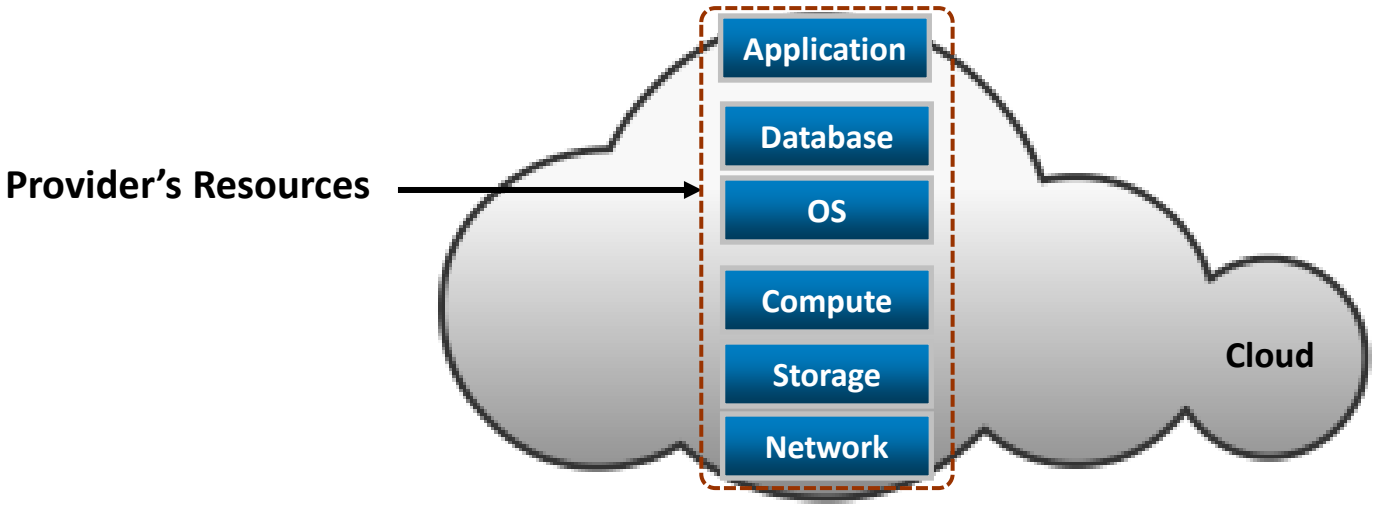
# Platform-as-a-Service

- Consumers deploy consumer-created or acquired applications onto provider's computing platform
  - Computing platform is offered as a service
  - Example: Google App Engine and Microsoft Windows Azure Platform
- Consumer has control over deployed applications



# Software-as-a-Service

- Consumers use provider's applications running on the cloud infrastructure
  - Applications are offered as a service
  - Examples: EMC Mozy and Salesforce.com
- Service providers exclusively manage computing infrastructure and software to support services



# Storage as a Service

- Several delivery models:

- "Comes with storage" model: the storage you get is the virtual disk attached to your VM (cheap)
- NAS-type: Can request file-oriented space.
  - Example: Amazon Elastic File System (which is just NFS for money)
- SAN-type: Can request block-oriented space.
  - Example: Amazon Elastic Block System (attaches to VM as virtual disk)
  - Example: Traditional SAN LUNs
- Object storage: A simplified storage interface
  - Example: Amazon S3
  - Need to zoom in...

# Object storage

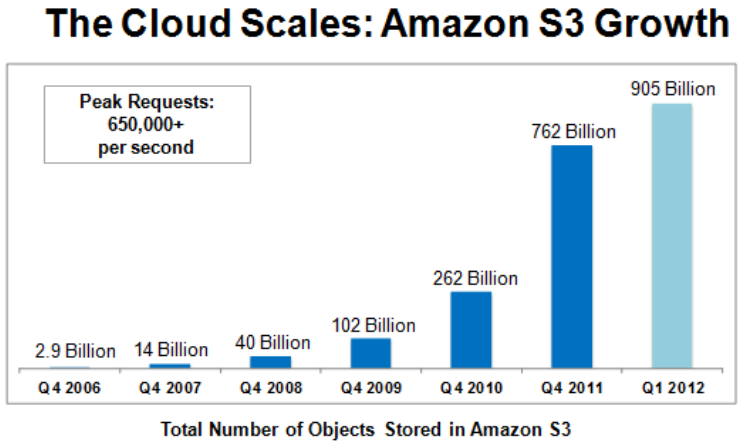
- Insight: if we drop traditional POSIX file interface (open/close/seek/read/write), can make a cheaper/faster/simpler file system

- Simpler verbs: GET and PUT
  - GET: Read the whole thing
  - PUT: Write the whole thing
- Intentionally omitted verbs: SEEK, MODIFY, etc.

- Example: Amazon S3
  - GET/PUT stuff to URLs – REST interface
  - All storage details behind that hidden from user
  - Cloud provider can migrate/replicate data and redesign back end
  - No changes means no consistency issues!

# Object storage has gotten huge! (Both in popularity ☺ and in size/costs ☹)

- Forces behind object storage
  - **Growth of the web**: S3 objects can be directly used on the web due to REST interface.
  - **Efficient distribution**: No fine grained file modifications means HTTP caching can be used – works like a Content Distribution Network (CDN) with multi-site storage. SAN/NAS can't do that!
  - **Ease of HA/DR**: Much simpler backup/replication semantics can be employed.




- Downsides
  - **Flexibility:** You may be tempted to use it like a filesystem, but you shouldn't: though "mount" tools exist, they have very bad performance for random IO, since the underlying protocol doesn't support it.
  - **Security**: difficult to get permissions set right – lots of data leaks have come from misconfigured S3 buckets

# Cloud Deployment Models

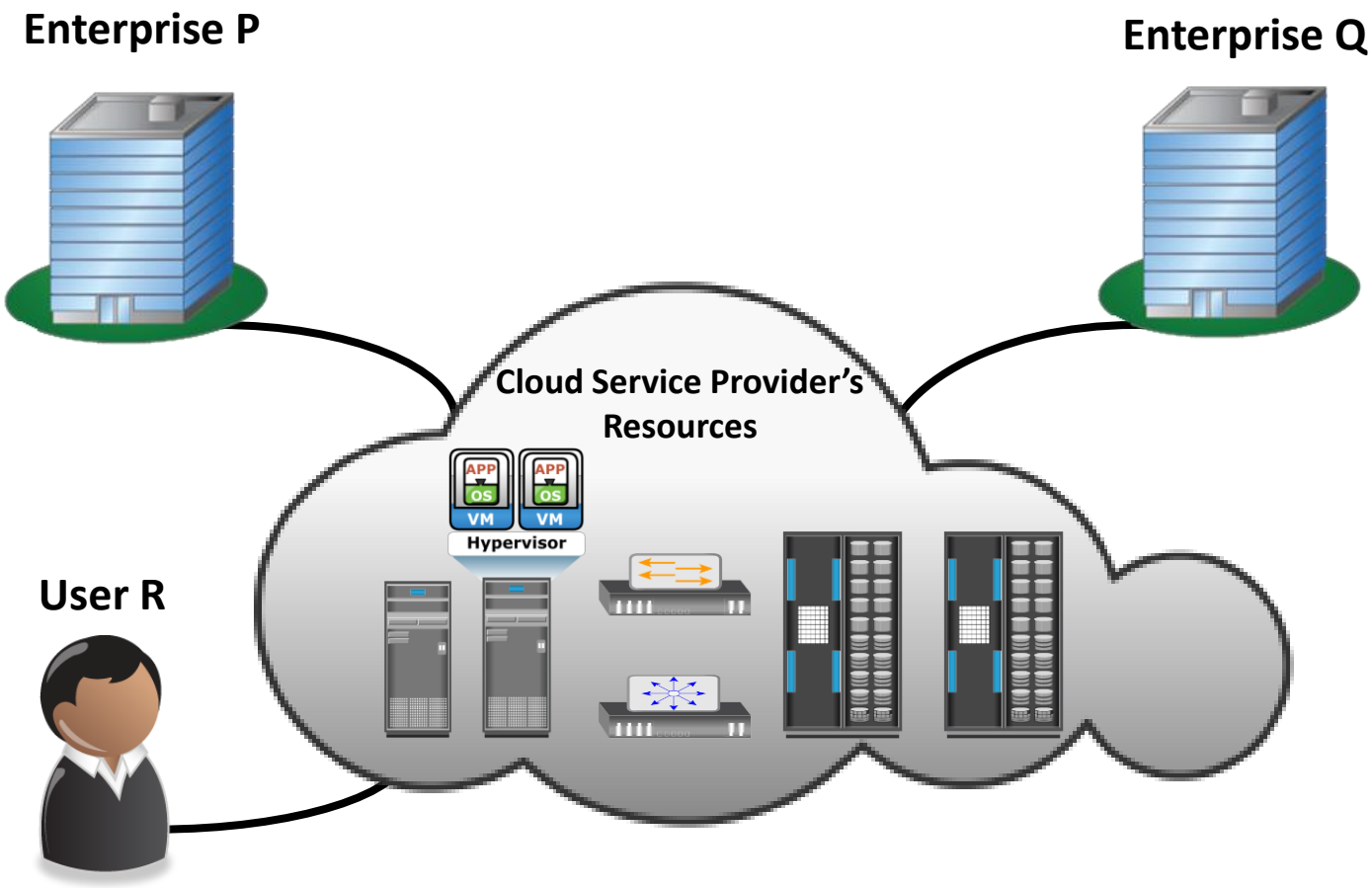
- Public
- Private
- Hybrid

# Public Cloud

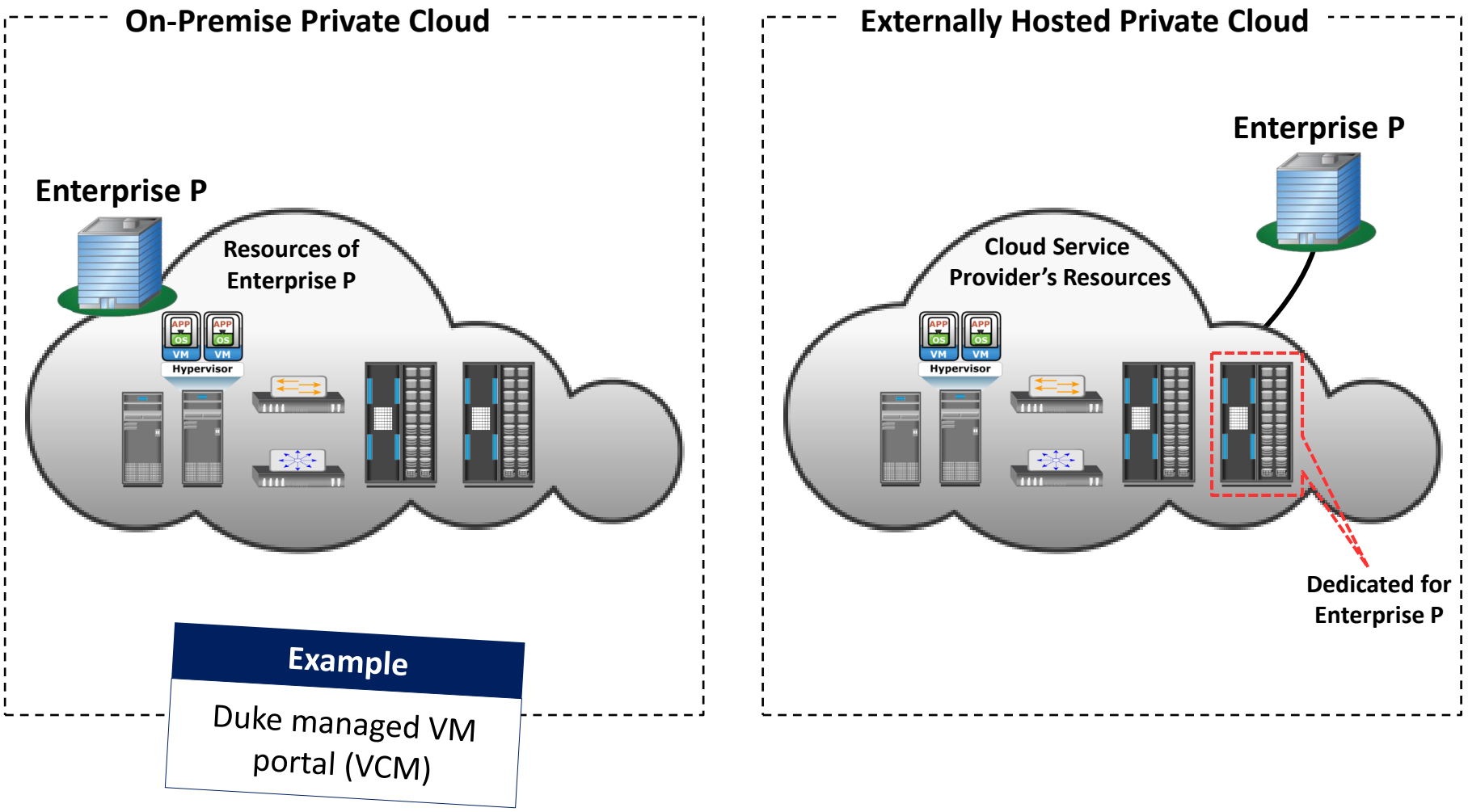
**Example**

Amazon AWS, Microsoft Azure, etc.



# Private Cloud

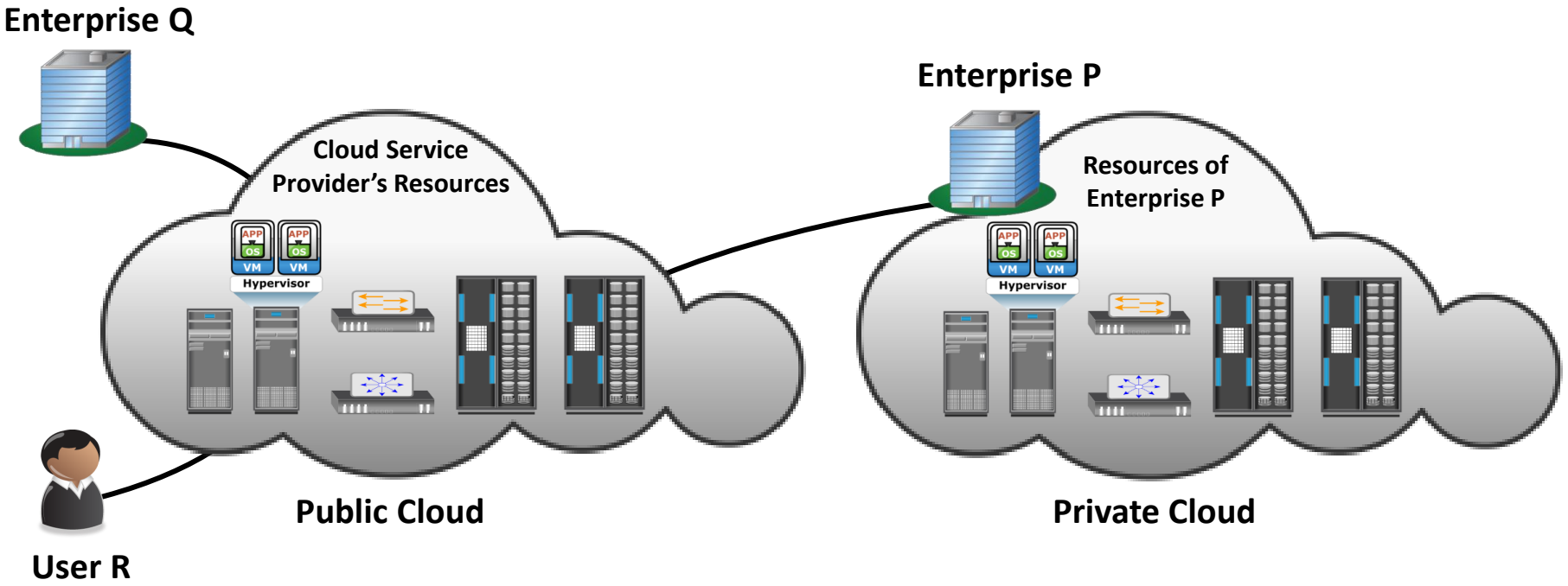


# Hybrid Cloud

**Example**

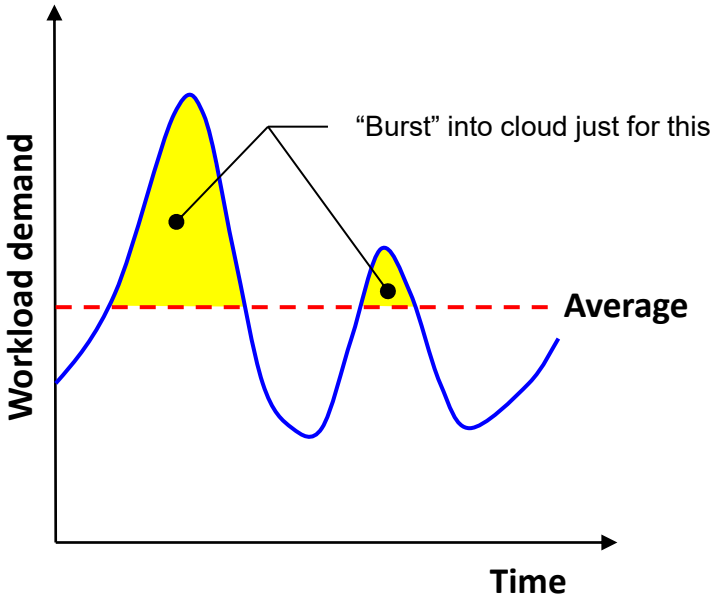
Amazon Direct Connect to a hosted data center (e.g. Equinix)



# Hybrid example with “bursting”

- Amazon has a program called Direct Connect
  - They have fast network lines at each Amazon AWS datacenter to nearby **colocation facilities**
    - **Colocation**: When you lease space for your server in someone else’s datacenter. (Colocation by itself isn’t considered ‘cloud’.)
  - You put up resources needed for your average workload
  - Run exclusively on your own gear most of the time
  - When demand grows past capacity, rent cloud services
  - Can start immediately; data comes from YOUR storage, not theirs
    - No migration



# Side benefit to direct-connect to cloud

- If your storage is directly connected to cloud,
  you still own your data, but can benefit from cloud's compute

- Important for regulations or concerns of liability/privacy
  - Financial/medical generally can't use any public cloud otherwise
  - Legal discovery: want to know if your data is being inspected/subpoenaed by authorities (or the NSA without a warrant)

# Cloud Challenges – Consumer's Perspective

- Security and regulation
  - Consumers are indecisive to transfer control of sensitive data
  - Regulation may prevent organizations to use cloud services
- Network latency
  - Real time applications may suffer due to network latency and limited bandwidth
- Supportability
  - Service provider might not support proprietary environments
  - Incompatible hypervisors could impact VM migration
- Vendor lock-in
  - Restricts consumers from changing their cloud service providers
  - Lack of standardization across cloud-based platforms

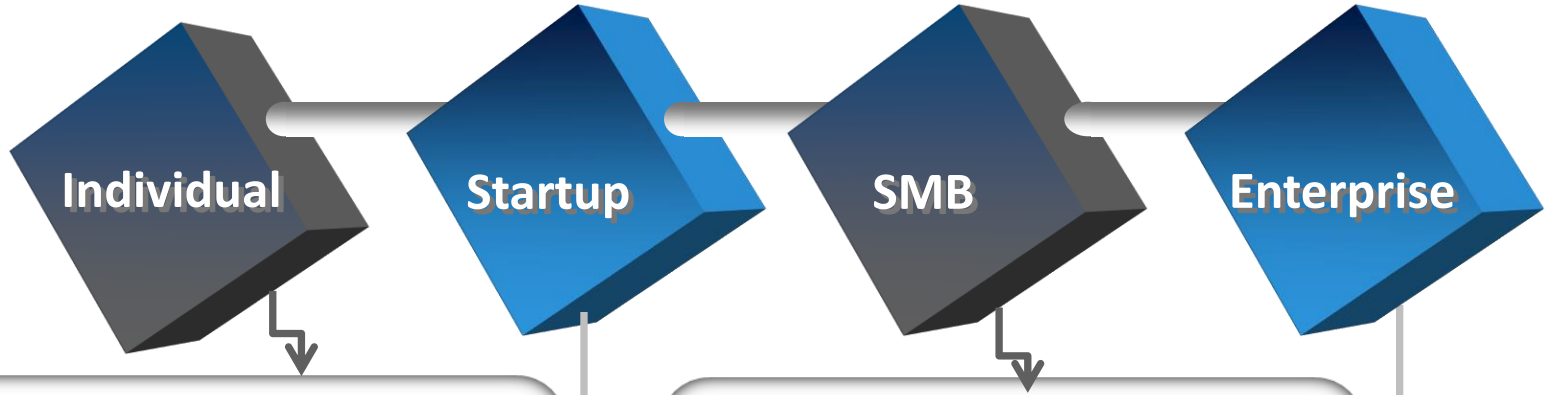
# Cloud Challenges – Provider's Perspective

- Service warranty and service cost
  - Resources must be kept ready to meet unpredictable demand
  - Hefty penalty, if SLAs are not fulfilled
- Complexity in deploying vendor software in the cloud
  - Many vendors do not provide cloud-ready software licenses
  - Higher cost of cloud-ready software licenses
- No standard cloud access interface
  - Cloud consumers want open APIs
  - Need agreement among cloud providers for standardization

**SLA = Service Level Agreement**

Contract that says what you'll get (and the penalty the provider pays if you don't get it)

# What Deployment Model Fits for You?




**Public cloud**

- Convenience outweighs risk
- Low cost or free
- Ex: Picasa, Google apps

**Hybrid cloud**

- Tier 1 apps: private cloud
- Tier 2-4 apps (backup, archive, testing): public cloud

**Public cloud**

- Convenience outweighs risk

**Private cloud**

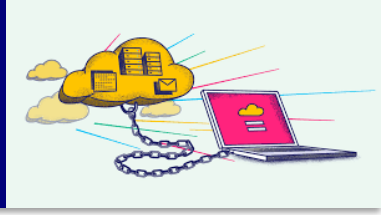
- Tier 2-4: private cloud
- Tier 1: may continue to run in a traditional data center environment

# Brief sideline: CAPEX and OPEX

- CAPEX = Capital Expenditure
  - Big investments
  - E.g., buying land, constructing buildings, ordering IT gear, etc.
- OPEX = Operating Expenditure
  - Money paid over time (financial obligations)
  - E.g., payroll, electricity, lease payments

- On **high-risk** projects, want ways to turn CAPEX into OPEX
  - "This project might not work, so I don't want to buy a rack of gear to support it; better to lease gear or use cloud."
- On **low-risk** projects, want to invest CAPEX to stop unending OPEX payments
  - "The project was a success, and we'll be running this app for at least 10 years, it makes sense to buy servers/storage for it and stop paying a premium to the cloud provider to host it."

# Danger: Becoming trapped



- Observation: It's much easier to migrate compute versus storage: programs are small, datasets are large.
- Common customer scenario:
  - Build new system on a cloud service
  - Low utilization means low OPEX payments, cool ☺
  - System is successful and grows (and data grows to many TB)
  - Large size means high OPEX payments ☹
  - Hmm, it makes sense to move this to non-cloud storage
  - **Problem**: have to transfer many TB
    - **Time** to transfer can be weeks *while* you're generating more data!
    - **Bandwidth** costs to just get the data out can be high
    - **Compatibility**: Tools/protocols may differ in on-premise system
  - **Result**: Customers become trapped on a cloud service with very unfavorable terms
  - **Solution**? Firms exist solely to help hide downtime and increase efficiency of cloud migration operations (but they cost too!)
- **Think ahead and watch out** when you architect cloud systems!

# QoS Considerations

- Consumers should check whether the QoS attributes meet their requirements
- SLA is a contract between the cloud service provider and consumers that defines QoS attributes
  - Attributes examples: throughput, uptime, and so on

# Quality of Service (QoS)

- As engineers, that previous slide is actually HUGE
- Want to set **performance guarantees and priorities**

- For CPU, prioritization/minimums are easy:
  - "If X is higher priority than Y and both X and Y are ready to run, run X"
  - "If X hasn't gotten it's minimum CPU time and X is ready to run, run X"
- For storage, it's hard; open field of research
- Challenges to storage QoS:
  - Allocation of cache?
  - Lots of implied reads/writes; how to prioritize?
    - Backup/replication IO
    - Journaling effects
  - Multiple metrics (IOPS for random, throughput for sequential)
    - If we guarantee a certain number of IOPS for a process, that could be most of our disk performance if we're doing random IO, or very little if we're doing sequential IO.

# Questions?