Cloud Computing

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 8a

Yue Cheng

Some material taken/derived from:

- Wisconsin CS 320 by Tyler Caraza-Harter.

Learning objectives

- Understand basic cloud pricing model
- Know IaaS and PaaS and their differences
    - The cloud offering that we've been using through this semester is IaaS (EC2).
    - PaaS cloud offerings are similar to the open-source data systems that we have been learning this semester.
- Get to know some basic ideas behind containerization and container orchestration

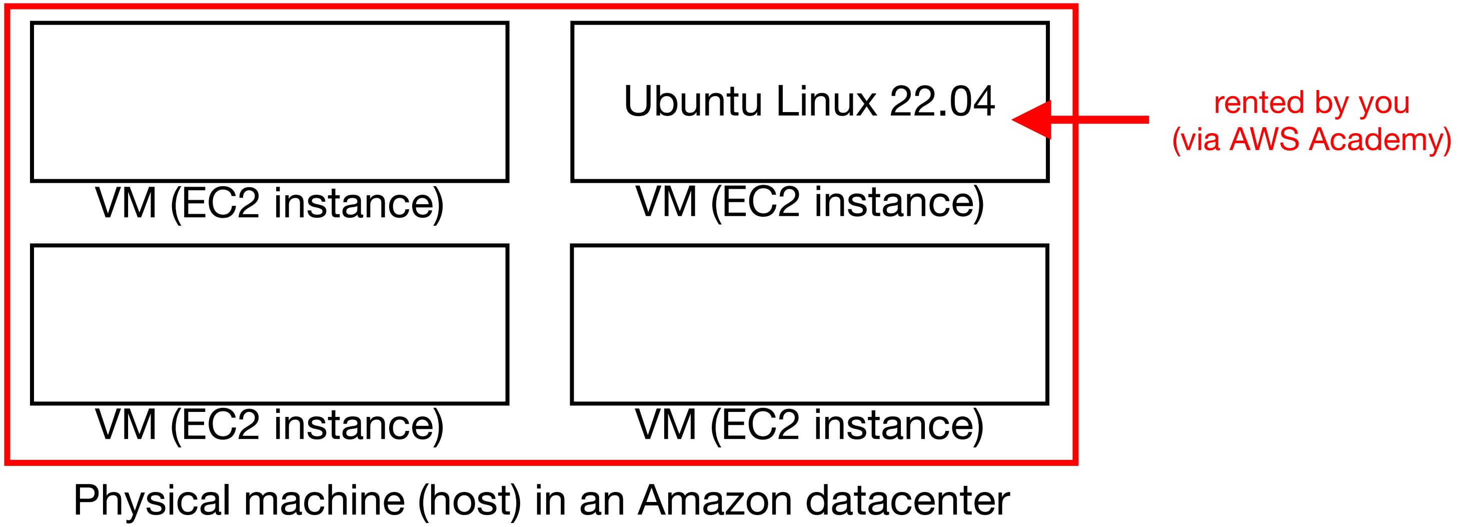
Background

The beginning

"Sometimes you need a lot of processing power; and sometimes you need just a little. Sometimes you need a lot, but you only need it for a limited amount of time."
-- Jeff Barr (https://aws.amazon.com/blogs/aws/amazon_ec2_beta/)

Amazon Web Services (AWS)

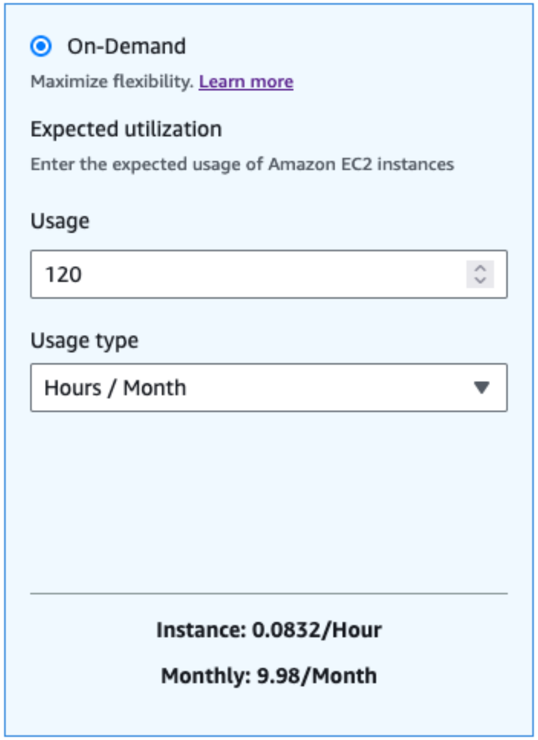
- Elastic Computing Cloud (EC2), rented VMs, launched in 2006
- "**Infrastructure as a Service**" (IaaS): rent infrastructure (compute, storage, network) instead of owning the hardware yourself



VM hours

Pricing summary

t3.large | Family: t3 | 2vCPU | 8 GiB Memory



● On-Demand

Maximize flexibility. Learn more

Expected utilization

Enter the expected usage of Amazon EC2 instances

Usage: 120

Usage type: Hours / Month

Instance: 0.0832/Hour

Monthly: 9.98/Month

Amazon EC2 On-Demand instances cost (Monthly): 9.98
Amazon Elastic Block Store (EBS) total cost (Monthly): 1.28

AWS pricing calculator: https://calculator.aws/#/

Pricing comparison

- one VM for a month: about $10
- about 120 hours a month (4*30)
- 120 VMs for an hour: about $10
- same computation + storage resources
- very different wait time

Be careful!!

- programmers previously optimized when things were too slow
- now we need to optimize when it is too expensive
- cost is not always obvious at the moment you're running a job (need to do "back of the envelope" estimates before you deploy the resources)

Other cloud services

- AWS now has > 200 services beyond EC2 (and growing)

- **IaaS** (Infrastructure as a Service)
  - EC2, other services that feel closer to raw hardware
  - Virtual disks, virtual network, some storage systems, etc.
  - Cheap + flexible – you can deploy & run anything on it (Spark, Ray, etc.)

- **PaaS** (Platform as a Service)
  - Cloud providers has deployed systems on the infrastructure; you pay to use the deployed system
  - Databases, application framework/platforms, ML training/deployment systems
  - Less flexible, easier to use
  - Often more expensive (though not necessarily more than doing it yourself due to efficiencies available to cloud provider but not you)

- Line between IaaS and PaaS distinction is a bit subjective.

Lock-in

- Customers (tenants) worry: what if the cloud provider increases the price? If it’s hard to move to a competing cloud, you’re “**locked in**”

- PaaS: services are often unique, and it would be hard to move to a different cloud providers

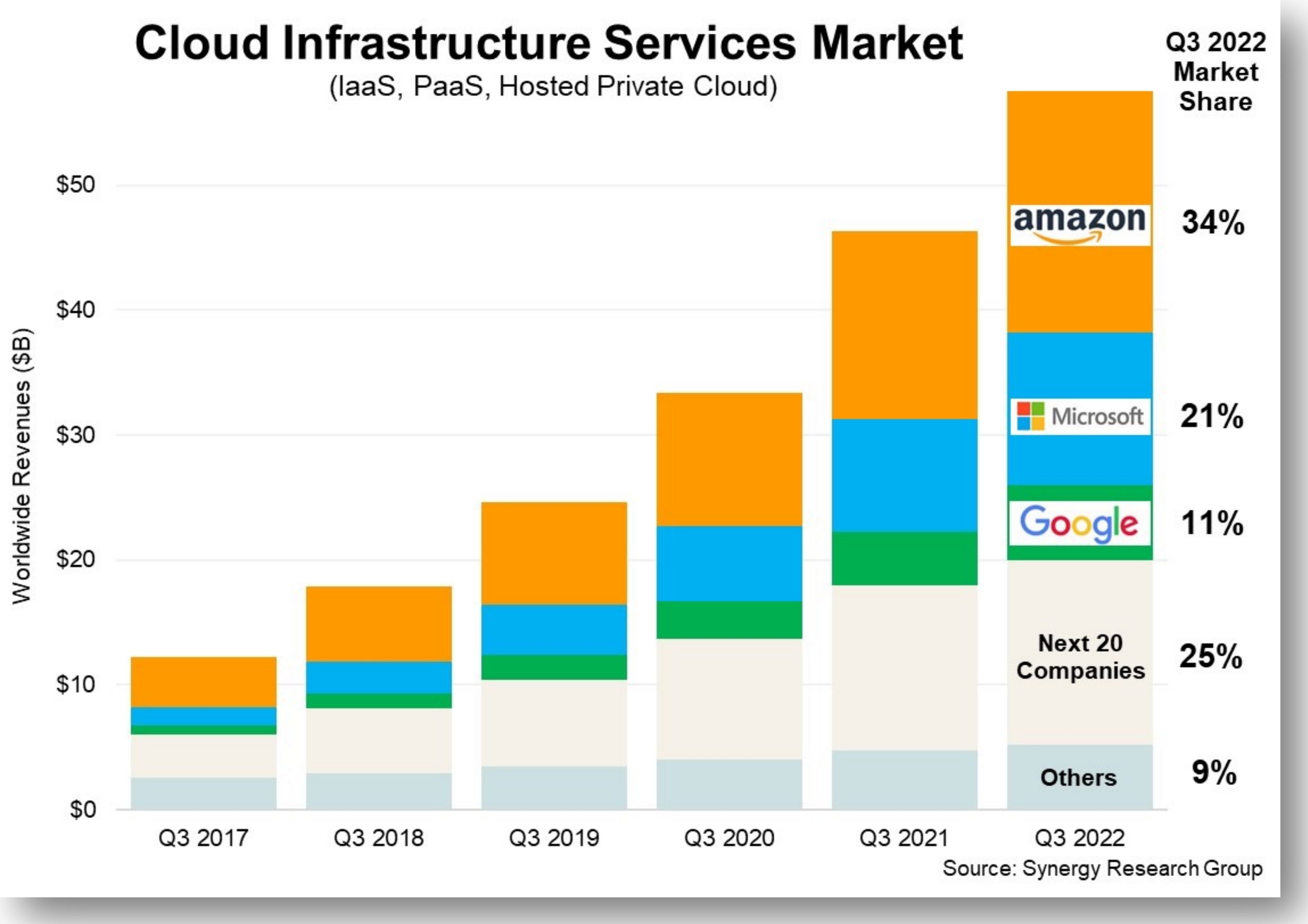
- IaaS: services like VMs are more uniform – it would be easier to switch to a different cloud to find the cheapest place to rent VMs

- **Data**: cloud providers often make it free to bring data into the cloud (ingress) but expensive to take it out (egress)

Case study: Dropbox

- A data sync startup founded back in 2008
- Became popular so quickly
  - Peak number of users: 500+ Million
  - Overall amount of data stored: 500 PB
- Initially stored all data on public clouds (AWS)
- Seriously considered to move data out of AWS
- Cloud vendor lock in
  - **Enormous** egress costs
- Now still parts of its data services sitting on AWS

Major cloud providers today



Source: Synergy Research Group

AWS

Azure

GCP

https://www.srgresearch.com/articles/q3-cloud-spending-up-over-11-billion-from-2021-despite-major-headwinds-google-increases-its-market-share

Cloud economics and billing models

Tenants: Pay-as-you-go?

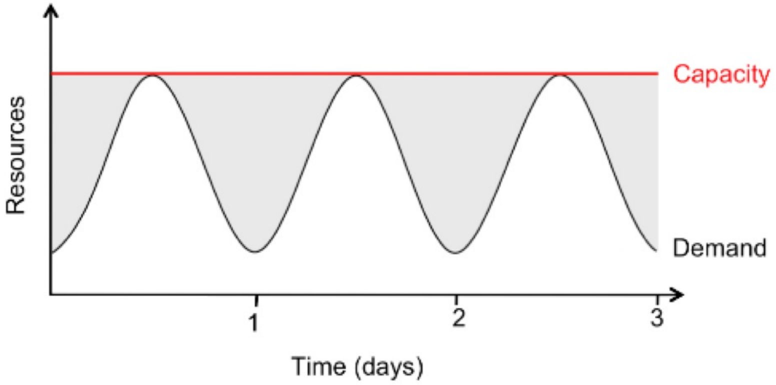
- (**Claimed**) pay-as-you-go pricing
  - Usage-based?
  - Most (compute) services charged per minute
  - Storage and network services charged per byte
  - No minimum or upfront fee

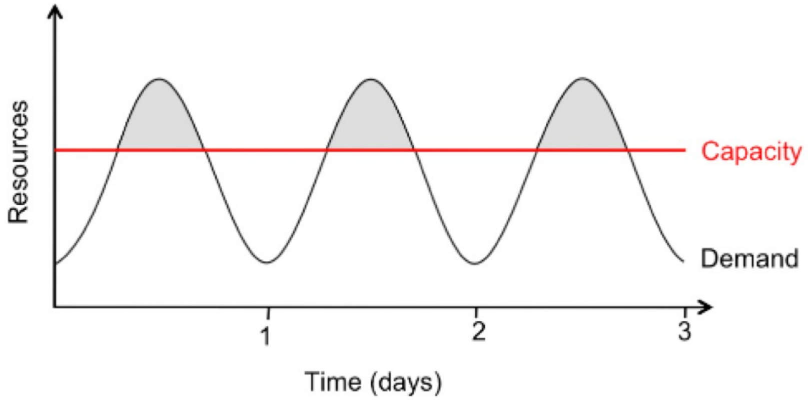
Tenants: Pay-as-you-go?

- (**Claimed**) pay-as-you-go pricing
  - Usage-based?
  - Most (compute) services charged per minute
  - Storage and network services charged per byte
  - No minimum or upfront fee

Q: Is the cloud pricing truly pay-as-you-go?

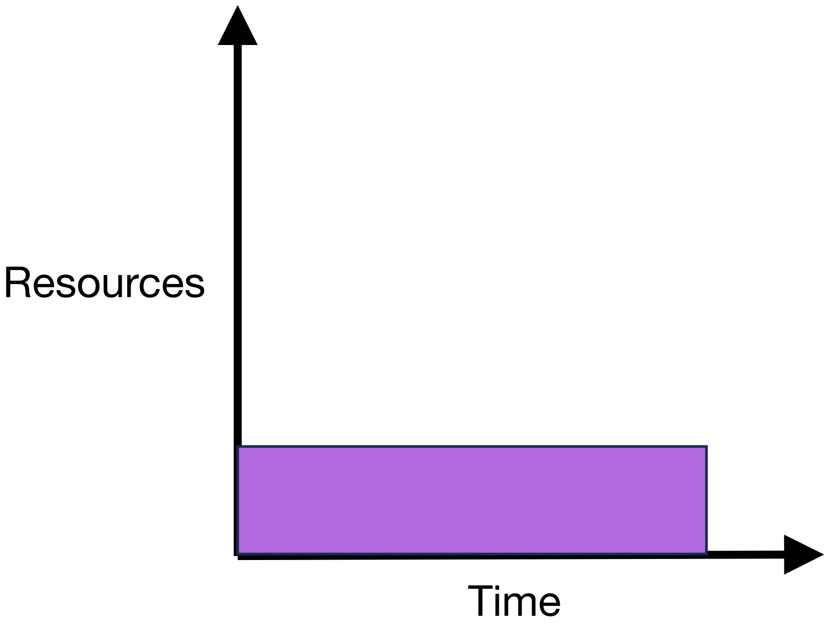
- **Problem**: How to perform strategic planning?

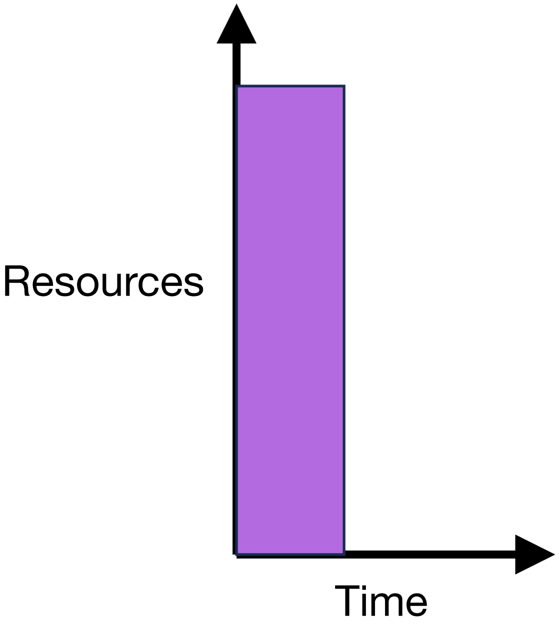




Tenants: Scalability gained?

- (**Ideally**) Linear scalability & perfect elasticity
  - Using 1000 servers for 1 hour costs the same as 1 server for 1000 hours
  - Same price to get a result faster

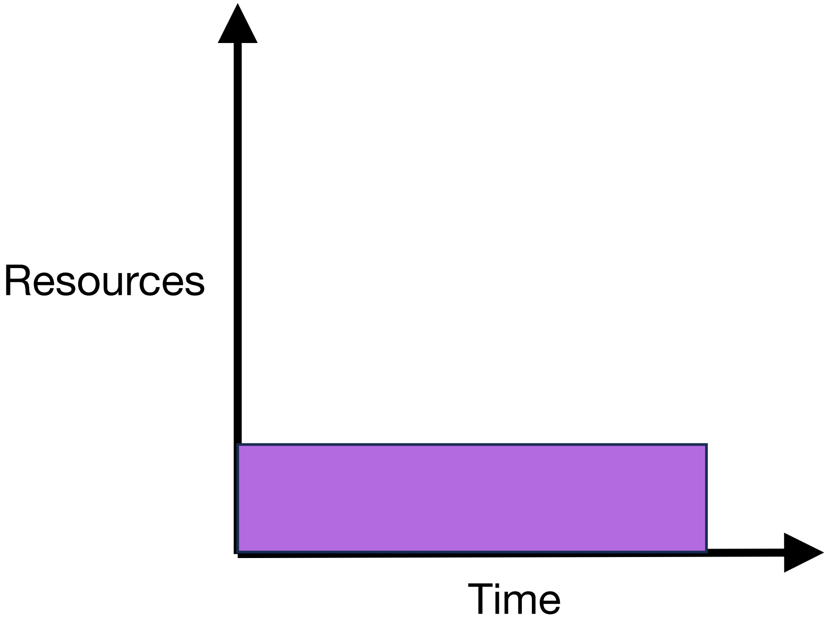


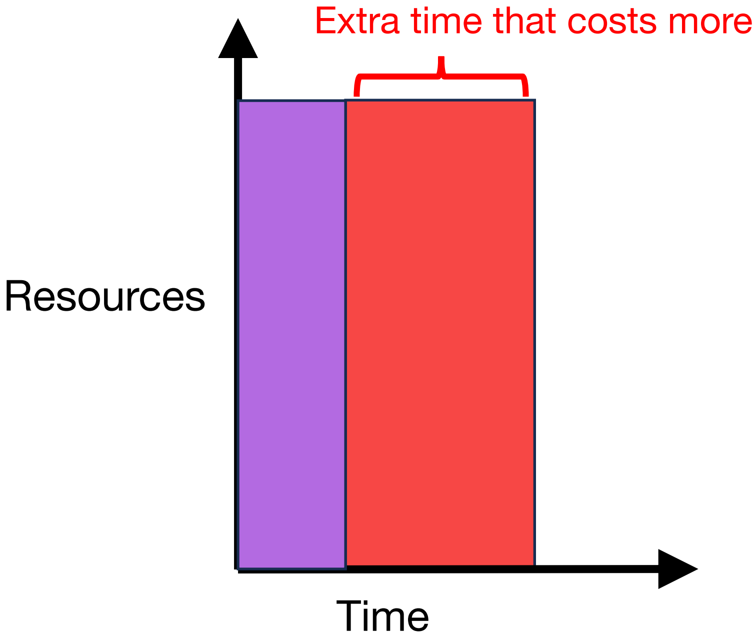


In practice, it really depends, case by case.

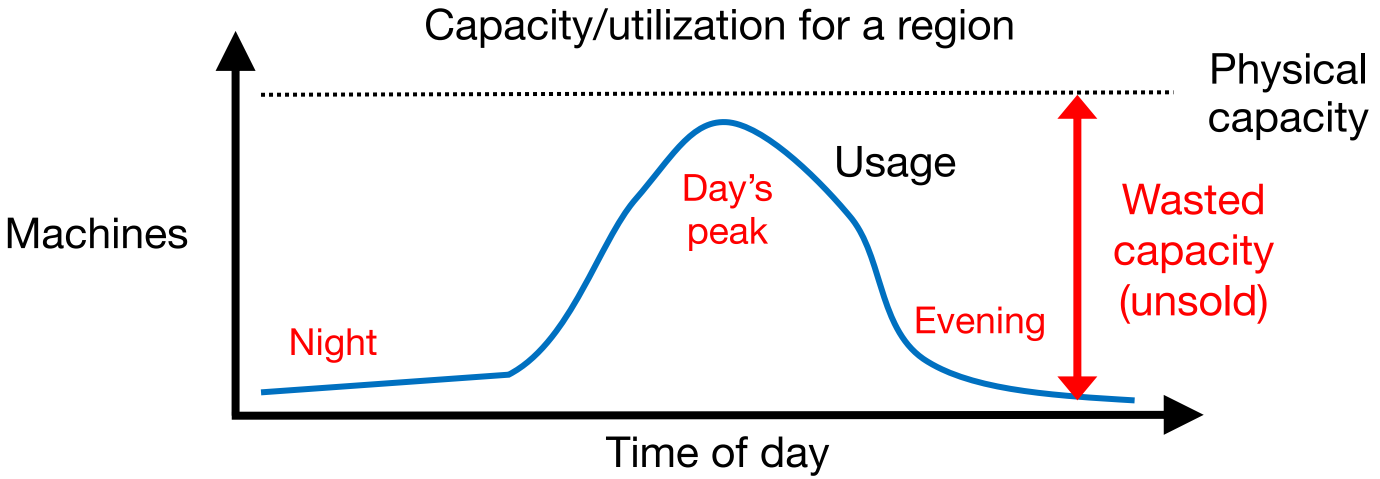
Likely the speedup of the computation is much lower than 1000X!

- (**In reality**) Scalability is sublinear and VM scaling is slow.
  - Using 1000 servers for 1+N hour costs N times more than 1 server for 1000 hours
  - Often higher price to get a result faster





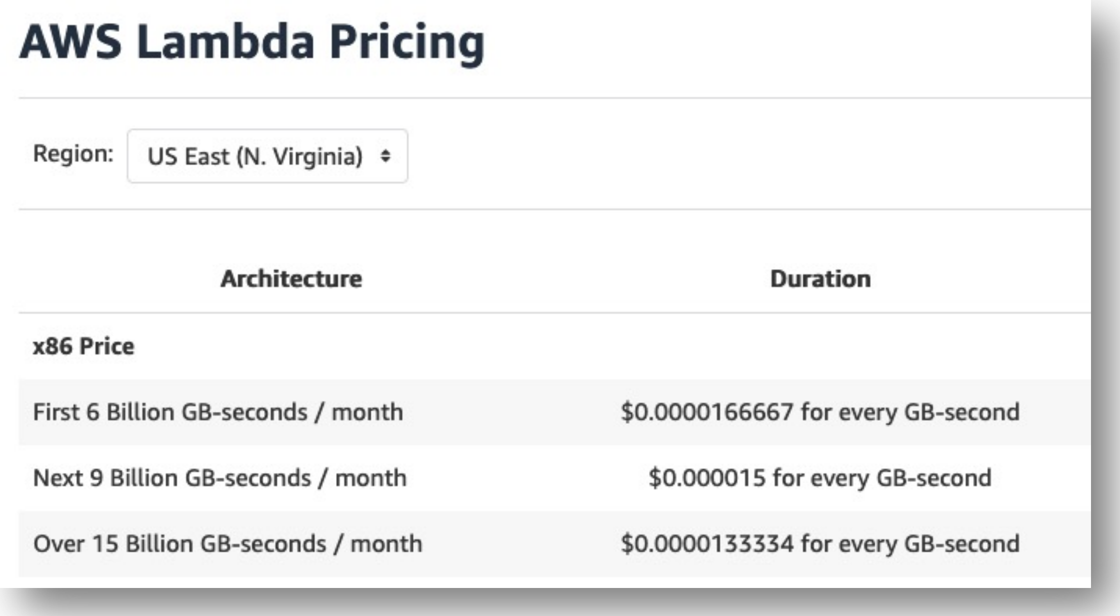
Providers: On-demand vs. spot instances



- How to create incentives for tenants?
  - Use less at peak time
  - Use more at low times

- Two VM deployment options
  - On-demand instances: Constant (high) price. Can generally get a VM. Won't be taken away from you arbitrarily. Used when capacity is needed at specific times.
  - Spot instances: Price varies throughput day. If you're not willing to pay enough, your computation waits for a cheaper price. VM might be interrupted ("preempted") once started. Excellent for once-a-day batch jobs.

Providers: Free tier, discounts at scale



AWS Lambda Pricing

Region: US East (N. Virginia) ⬍

Architecture	Duration
x86 Price	
First 6 Billion GB-seconds / month	$0.0000166667 for every GB-second
Next 9 Billion GB-seconds / month	$0.000015 for every GB-second
Over 15 Billion GB-seconds / month	$0.0000133334 for every GB-second

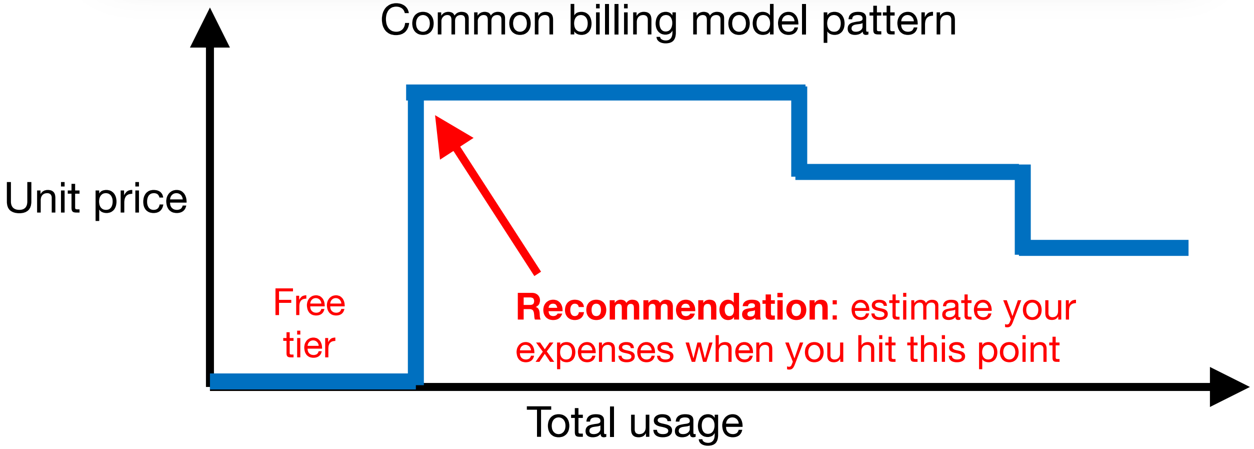
AWS Lambda example

"The AWS Lambda **free tier** includes one million free requests per month and 400,000 GB-seconds of compute time per month."

(https://aws.amazon.com/lambda/pricing/)

"Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 1 ms."

Common billing model pattern



Recommendation: estimate your expenses when you hit this point

Recommendation: check if you have a large number of small ops getting rounded up

Virtualization and container orchestration

Virtualization: Providing isolation

- We don’t want different applications running on the same hardware to interfere with each other – we want them to be isolated. Concerns:
  - Malicious programs, buggy programs, fairness

- Ways to interfere
  - **Directly**: Seeing/modifying data of another process
  - **Indirectly**: Inflicting bad performance on another process

- Some operating system isolation features with a long history:
  - Virtual memory: Can’t see another process’s data (namespace isolation)
  - Schedulers: Can’t hog the whole CPU (performance isolation)

Problem: CPU and memory are not the only resources

Goal: Both namespace AND performance isolation for EVERY kind of resource

Linux features: cgroups and namespaces

- cgroup types (performance isolation)
  - cpu, memory, cpuacct, cpuset, freezer, net_cls, blkio, perf_event, net_prio, hugetlb, pids, rdma
- namespace types (namespace isolation)
  - network, mount, time, user, cgroup, IPC, PID, UTS

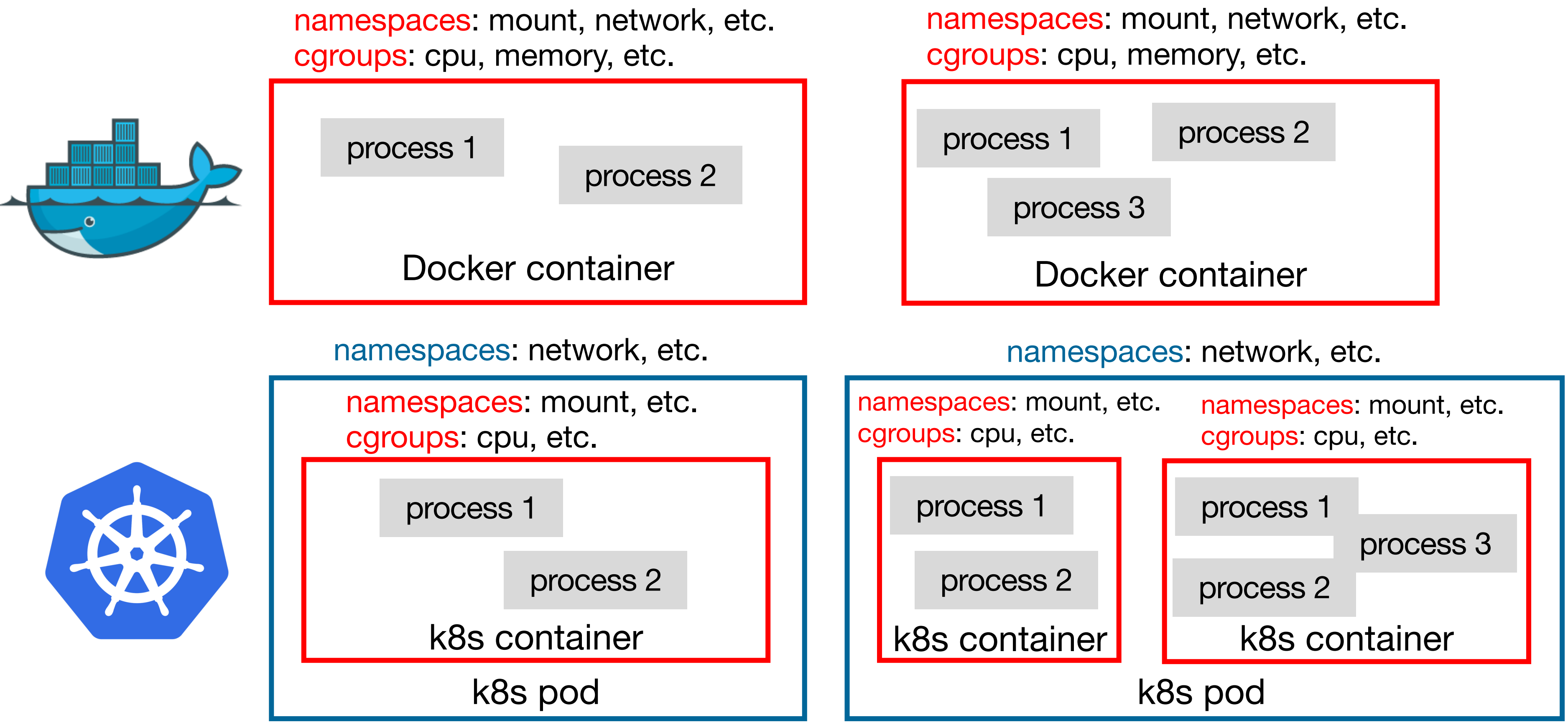
“mount” is for
file system

- Both cgroups and namespaces apply to sets of processes. Configuring all this by hand is VERY complicated.
- One reason Docker is popular: “docker run …” starts a process using all these features, each with reasonable configurations.
- “Containers” definition: Set of processes using a combination of cgroup / namespace / other features.

Kubernetes (k8s)

8 letters

- cgroups and namespaces are very flexible: Docker's approach is just ONE way to use them to build containers



namespaces: mount, network, etc.
cgroups: cpu, memory, etc.

process 1
process 2

Docker container

namespaces: mount, network, etc.
cgroups: cpu, memory, etc.

process 1
process 2
process 3

Docker container

namespaces: network, etc.

namespaces: mount, etc.
cgroups: cpu, etc.

process 1
process 2

k8s container

k8s pod

namespaces: network, etc.

namespaces: mount, etc.
cgroups: cpu, etc.

process 1
process 2

k8s container

namespaces: mount, etc.
cgroups: cpu, etc.

process 1
process 3
process 2

k8s container

k8s pod

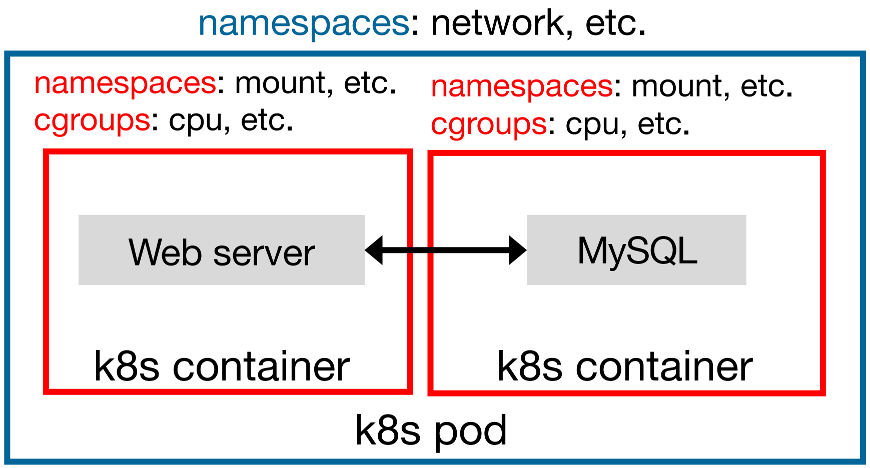
Kubernetes (k8s)

8 letters

- **Motivation**: We often want to deploy multiple applications that "work together"

- Shared between containers in same pod
  - Same VM, IP, port visibility

- Not shared
  - CPU/memory resources (etc.)
  - Files (great! Each can have their own Linux distro, packages versions, etc.)



Container orchestration

- Kubernetes currently is the most popular container orchestrator.
  - A container orchestrator can launch many containers in a cluster (of VMs or physical machines).

- Other orchestrators:
  - Docker compose: only launches containers on one node (so not necessarily an “orchestrator” depending on definition)
  - Docker swarm: built from compose to support multiple nodes
  - Nomad: simpler alternative to Kubernetes

Docker demo