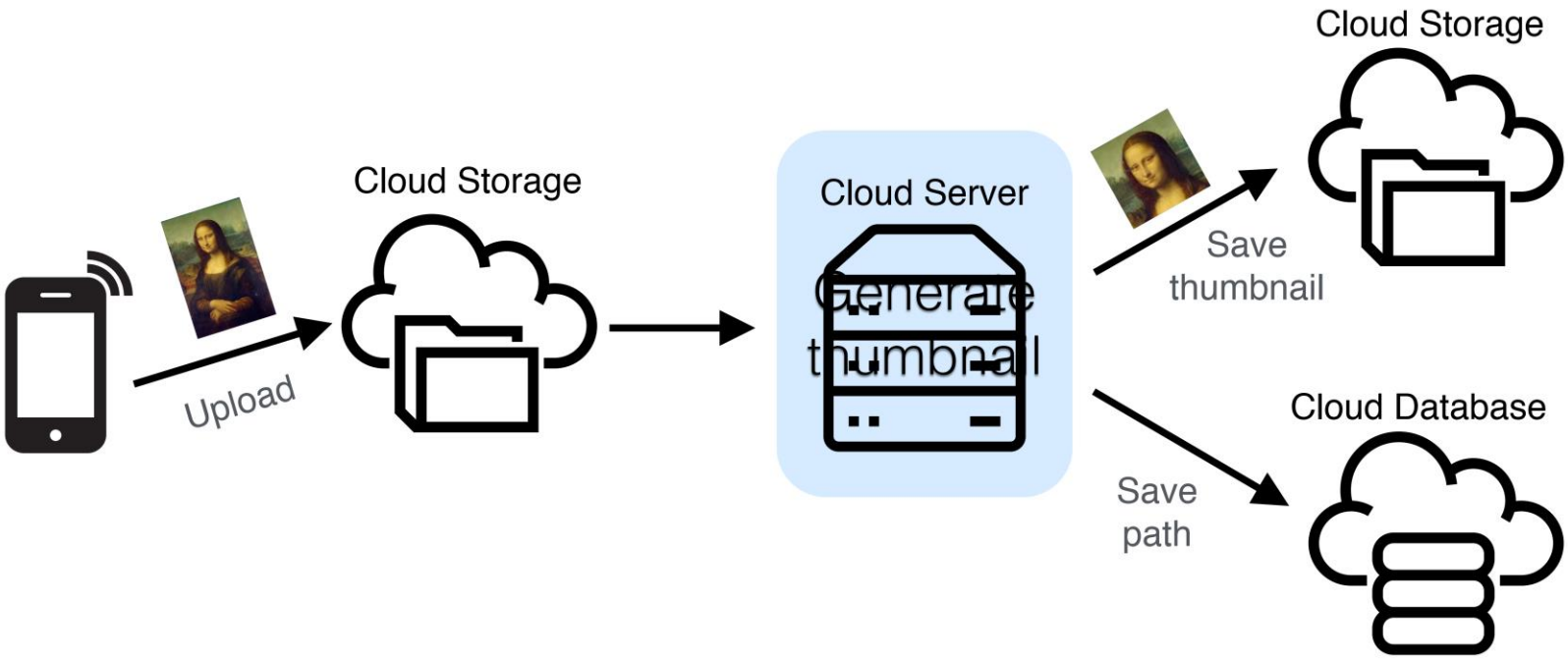
Serverless Computing

ENGR689 (Sprint)

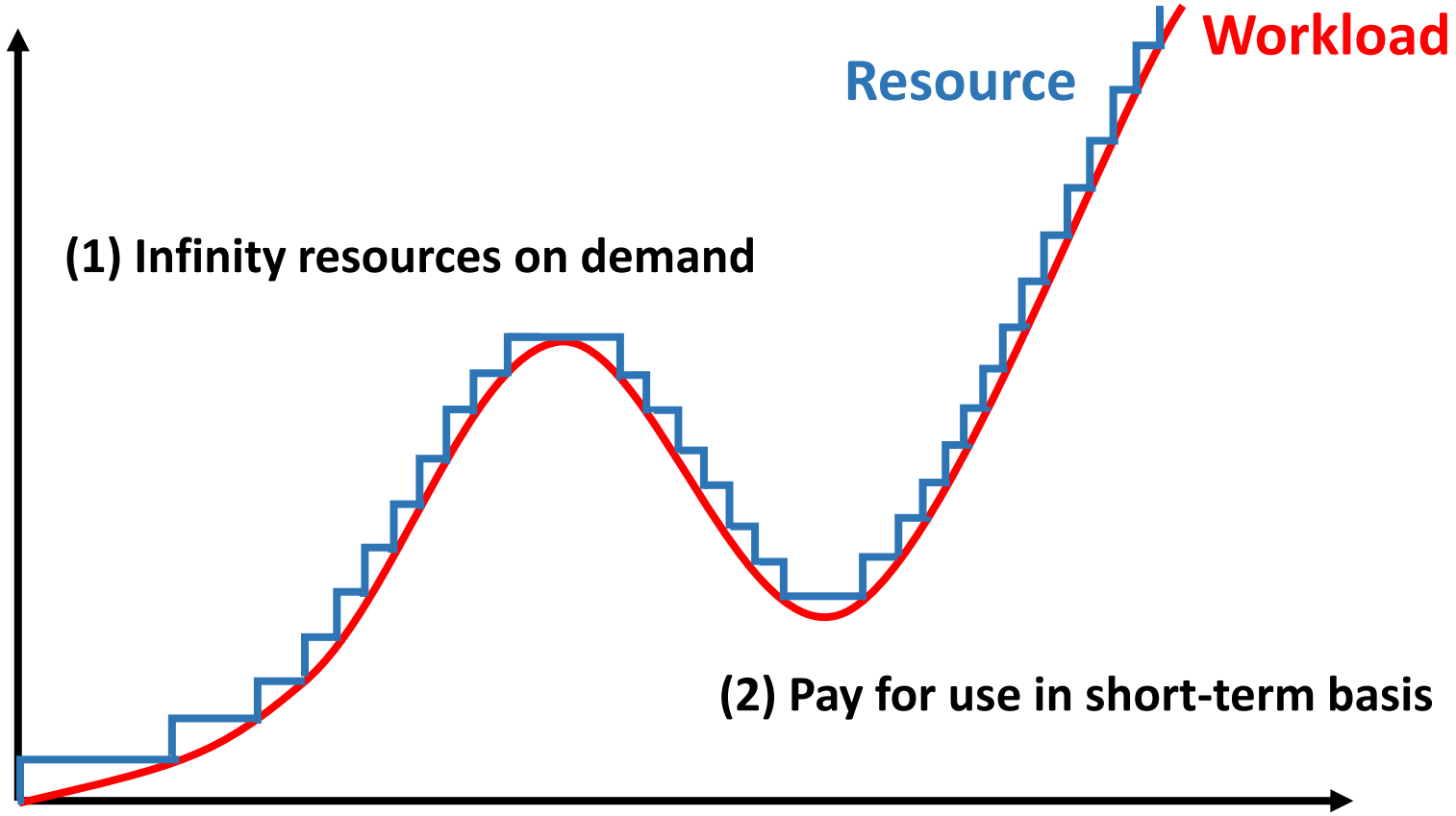
Cloud Prediction 10 Years Ago

Prediction	Delivery on promise 10 years later	
Appearance of infinite resources	★★ ★	
Elimination of up-front costs	★★ ★	
Pay-per-use	★★	Mostly “pay per reservation”
Economics of scale accessible to all	★★	Cloud providers capture most of the economic benefits of scale
Simplifying operations, increasing utilization due to virtualization	★★	Still requires substantial “virtual operations”
Higher hardware utilization via cross-organization multiplexing	★	Coarse-grained multiplexing only, mostly batch jobs

A Simple Example



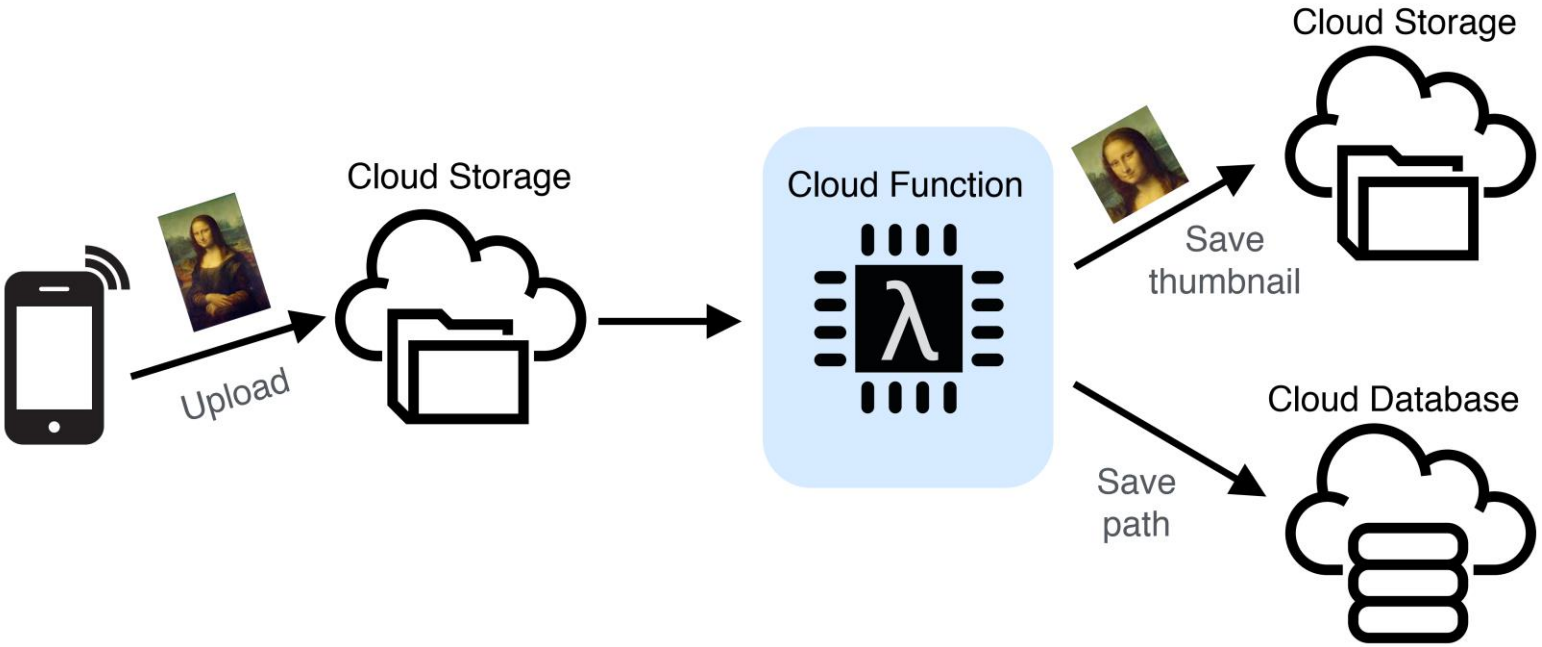
Utility Computing in Cloud



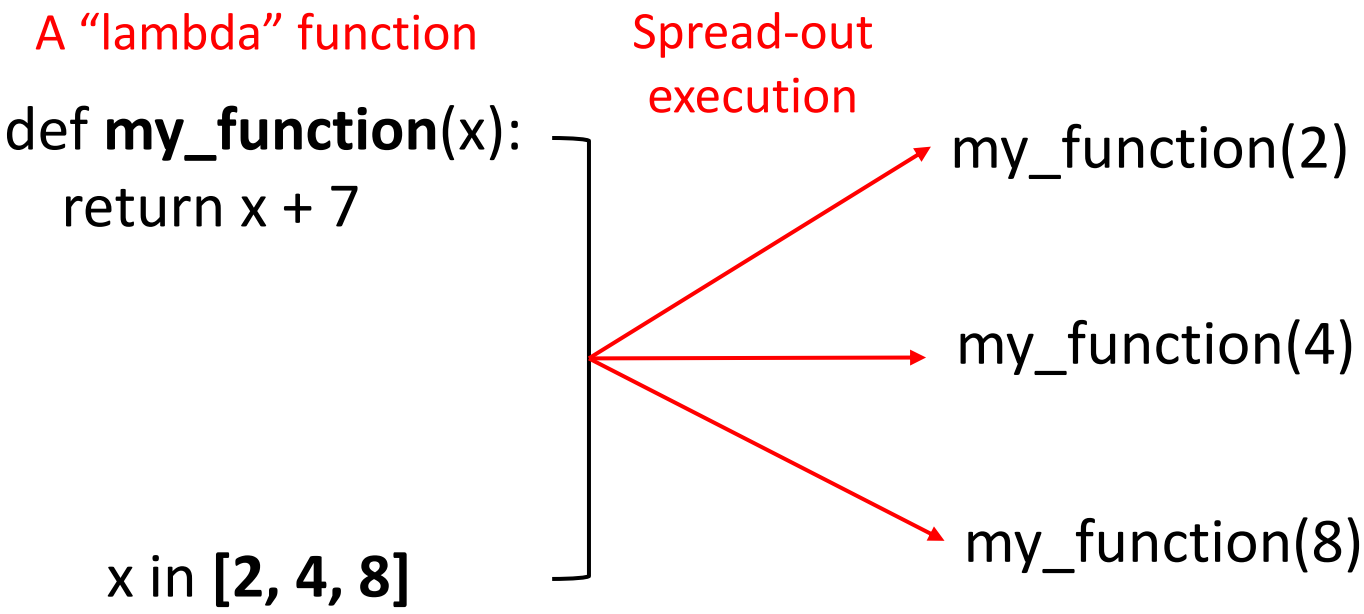
Challenges for Utility Computing

- **For users:**
  - Deciding the scaling units (CPUs, RAM, storage, etc)
  - Designing software that can adapt to scaling
- **For providers:**
  - Redundancy for availability and crash tolerance
  - VM creation and migration
  - System maintenance and upgrade

A Simple Example

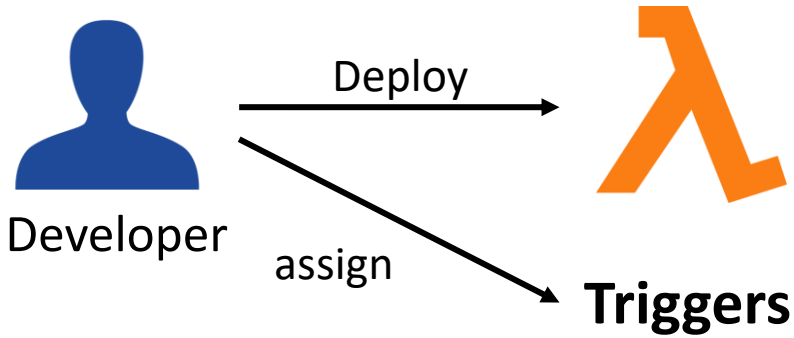


Function-as-a-Service: AWS Lambda (1/4)

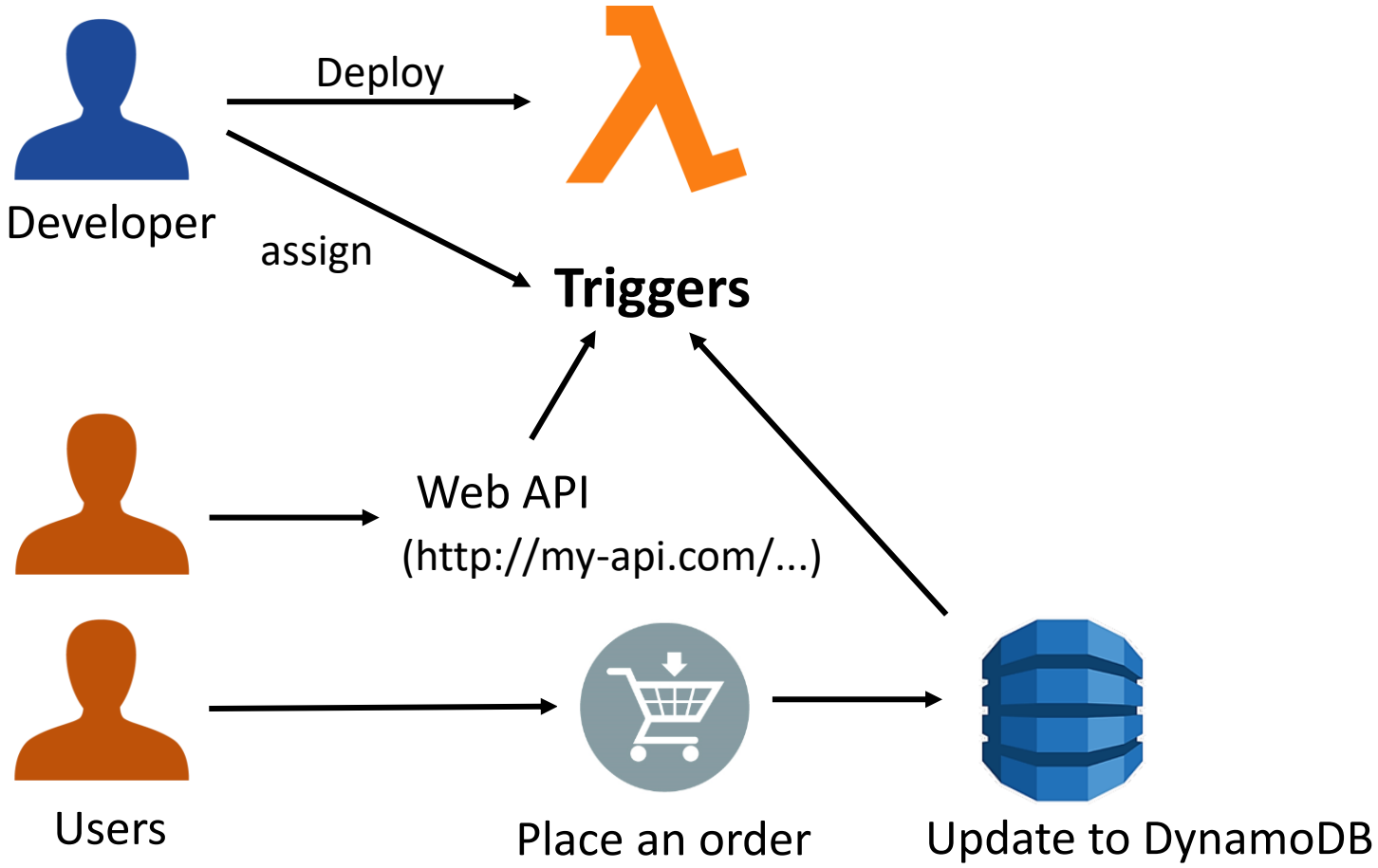


Lambda functions may or may not be scheduled to the same host, and the users cannot make any assumption.

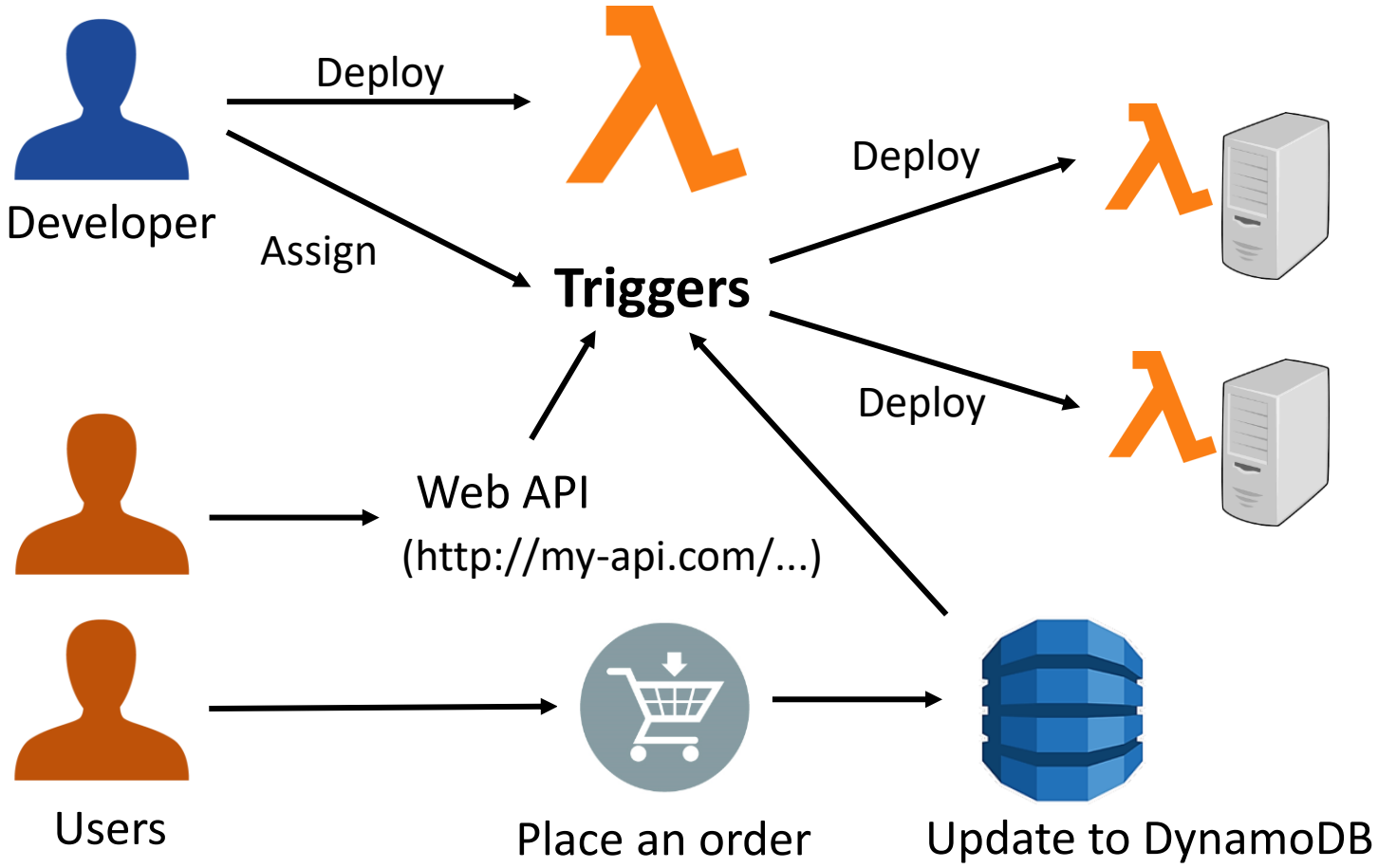
Function-as-a-Service: AWS Lambda (2/4)



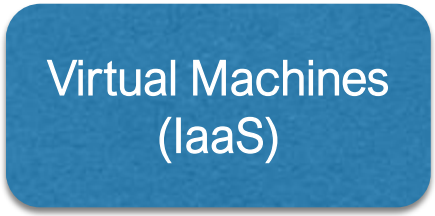
Function-as-a-Service: AWS Lambda (3/4)



Function-as-a-Service: AWS Lambda (4/4)



Two Paths



Amazon EC2

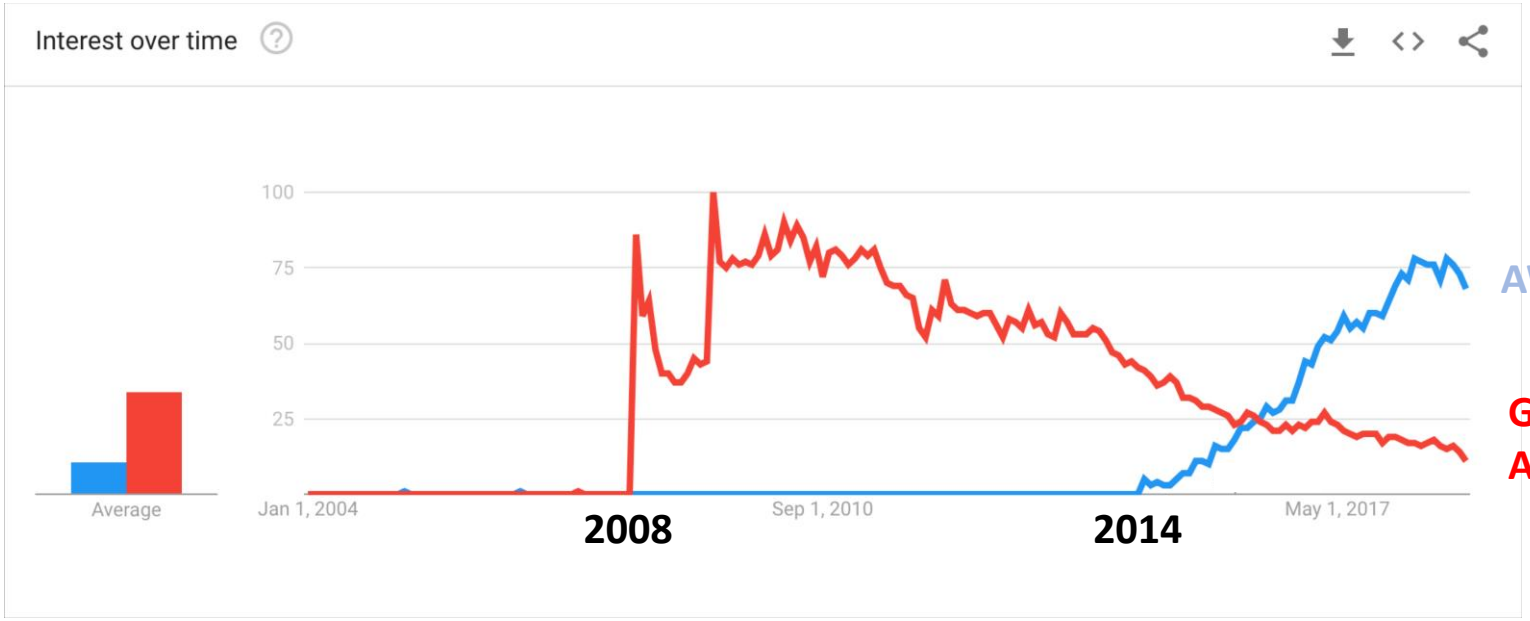
- Standard applications now in cloud
- Traditional administrative tasks

Google App Engine

- Customized application framework
- Simplified operations – takes care of redundancy, geographic distribution, autoscaling, monitoring, logging, system upgrades, migration, request routing / load balancing

Google App Engine vs AWS Lambda

Google Trend Search Term Popularity



Similarity Between PaaS and FaaS

- Using high-level language such as Python or JavaScript
- Impose limitations on resources and run time
- Bind function to events
- Pay bill based on code runtime

Why FaaS Took Off?

- Autoscaling done right
  - Highly elastic - adapts quickly
  - Scales down to zero
  - Fine-grained 100 ms billing increment
  - Cloud provider shares risk and responsibility for utilization
- Benefits from the scale of cloud platforms & ecosystem of APIs

AWS Lambda (FaaS) vs EC2 (IaaS)

	AWS Lambda	EC2 (On Demand)
How program is run	Triggered by events	Continues running
Language choices	JS, Python, Java, Go, etc (Can also run executable)	Any language
Program states	Stateless (kept in storage)	Stateful or stateless
Network connection	Only outbound	Inbound and outbound
Max memory size	0.125 - 3 GB	0.5 - 1952 GB
Max local storage size	0.5 GB	0 - 3600 GB
Max run time	15 minutes	Infinity
Min pricing units	0.1 seconds	60 seconds
Price per pricing units	$0.0000002 (0.125 GB)	$0.0000867 - $0.4080000
OSes and libraries, autoscaling, deployment, fault tolerance, etc	Maintained by cloud provider	Maintained by cloud users

Characteristics of Serverless

Same as PaaS (App Engine):

- Event-driven
- Language preferences
- cheap per-unit cost
- Stateless
- Resource restriction
- No inbound network

Different from PaaS:

- Run time limitations
- Strong isolation
- All-around ecosystems
- Faster scaling
  (10,000 lambdas in < 1 sec)

Characteristics of Serverless

Attractive to users

Same as PaaS (App Engine):

- Event-driven
- Language preferences
- cheap per-unit cost
- Stateless
- Resource restriction
- No inbound network

Different from PaaS:

- Run time limitations
- Strong isolation
- All-around ecosystems
- Faster scaling (10,000 lambdas in < 1 sec)

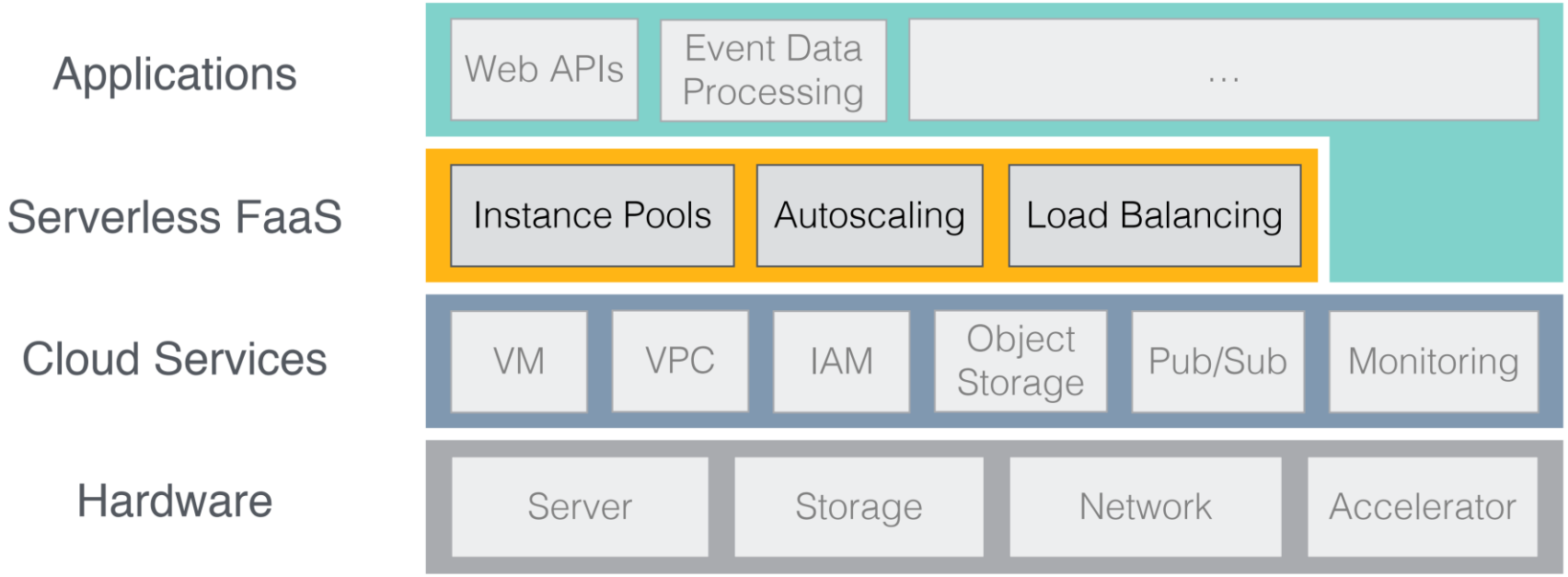
Valuable to cloud providers

Easier replication, instance placement, and resource allocation

Why is Serverless (FaaS) Popular?

- **For users: better pricing or convenience**
  - Web and API serving (32%) ➔ **burst-out execution**
  - Easy-to-program data processing (21%)
  - Service integration (17%)
  - Internal tools, chat bots, internet of things
- **For providers: vendor lock-in**
  - 24% of lambda users are new to cloud
  - 30% of existing cloud users also use lambdas

Today's FaaS Architecture

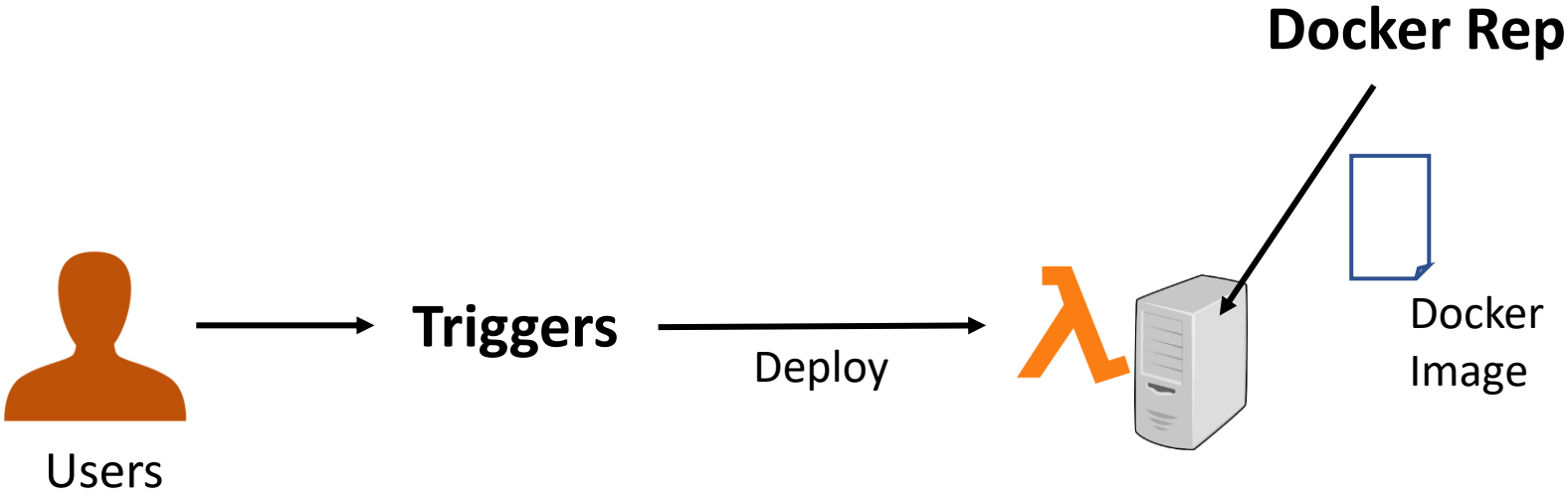


Security Concerns

- Most concerns (responsibility) have not changed but are shifted to cloud provider
- Cloud provider ensures monitoring, updating, and defending of the systems
- Needs **strong isolation** for lambda functions on a platform

Lambdas vs Containers

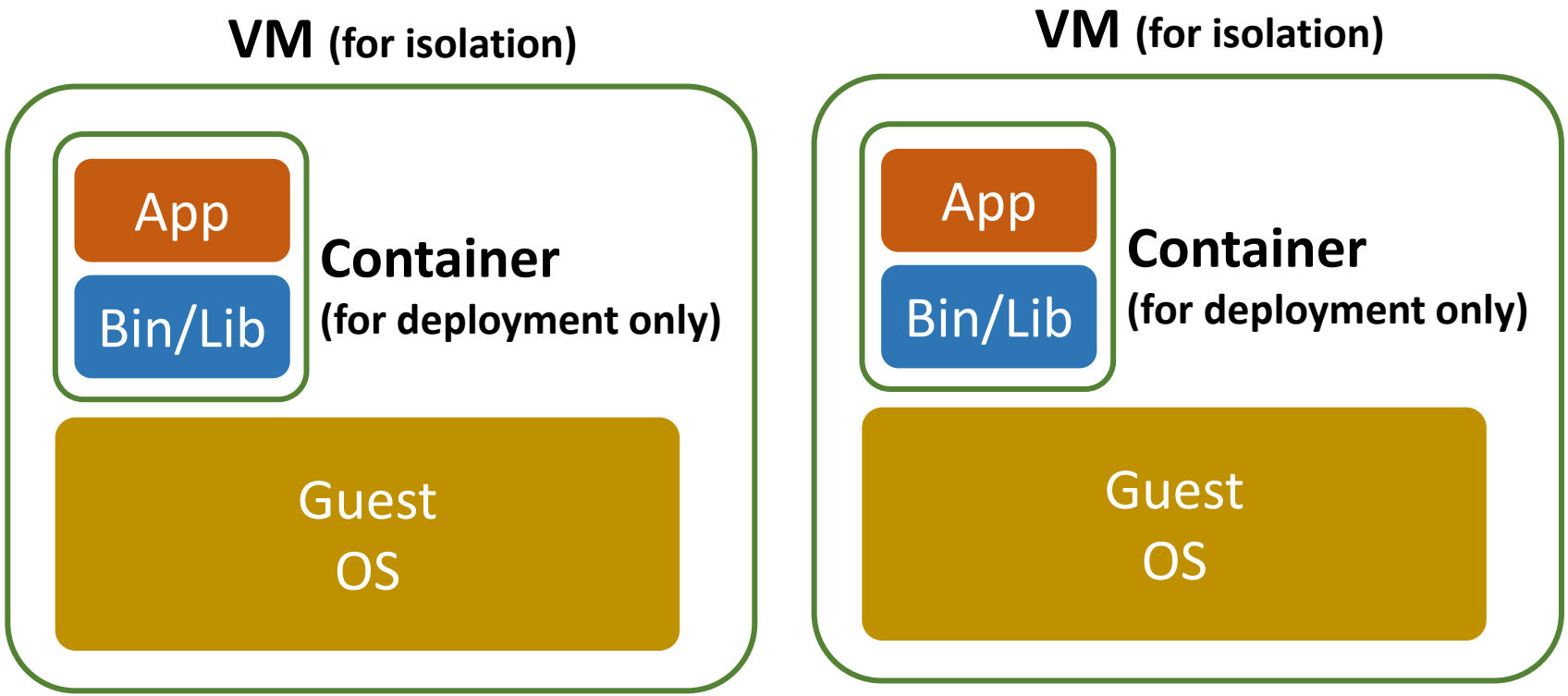
- Containers provide the customizable execution environment for lambdas



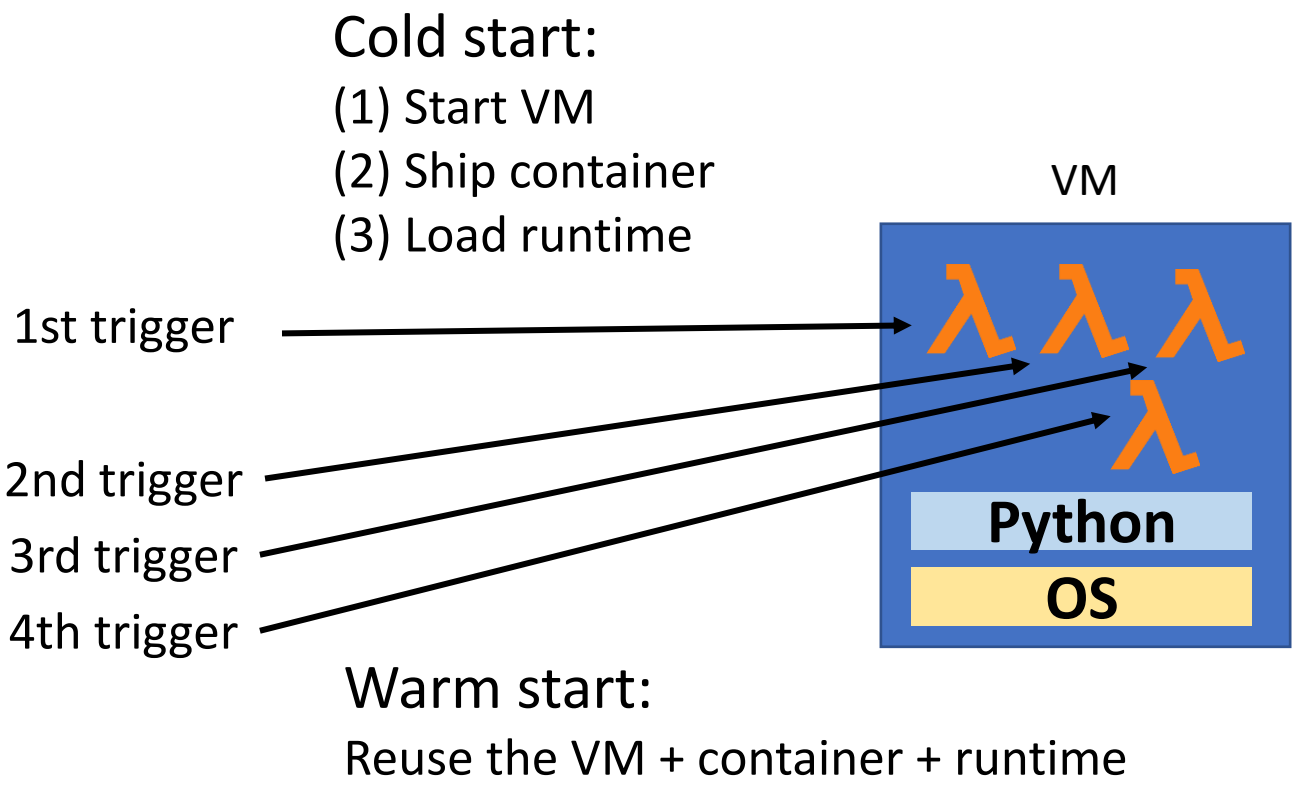
Need dependencies
(1) Runtime (e.g., Python)
(2) Libraries
(3) Packages/Modules

Combining Containers & VMs

- Containers are not secure enough for isolating malicious lambdas

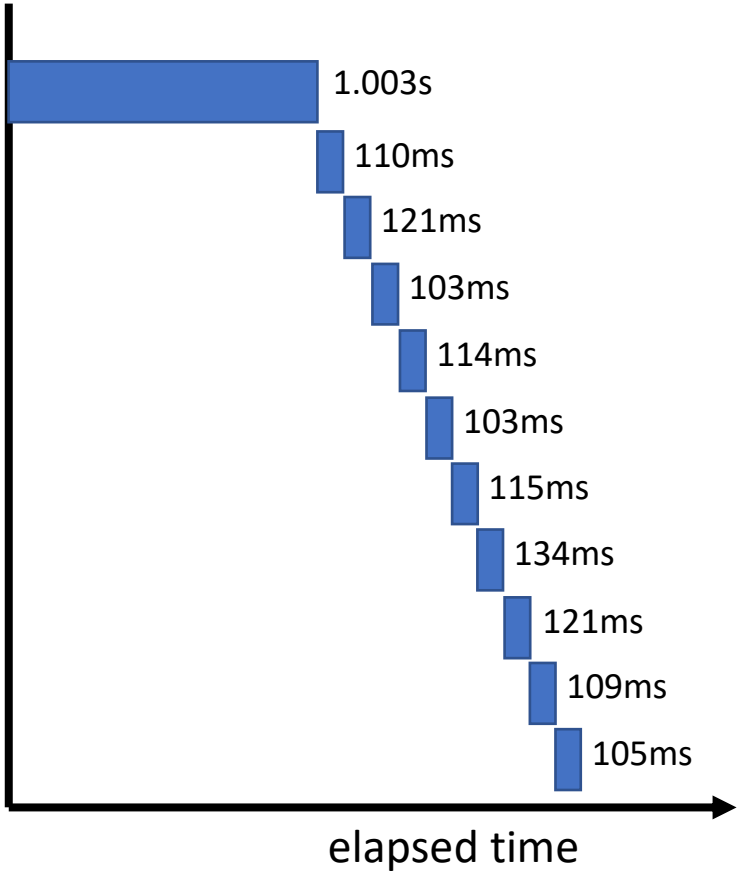


Cold Start vs Warm Start (1/2)

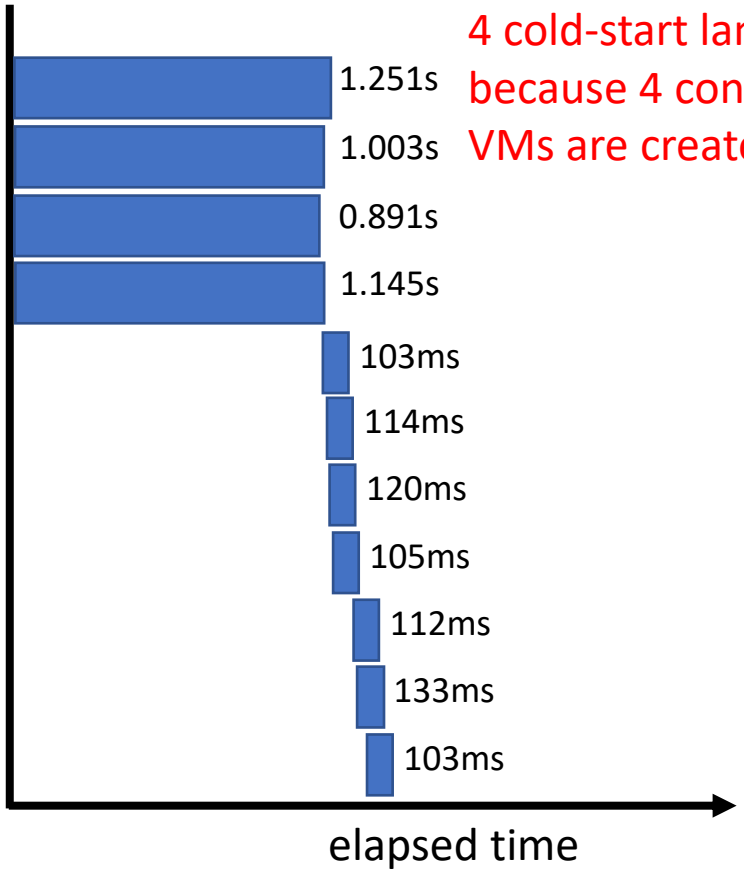


Cold Start vs Warm Start (2/2)

One trigger at a time:



4 triggers at a time:



4 cold-start lambdas because 4 concurrent VMs are created.