Serverless Computing

CS 4740: Cloud Computing

Fall 2024

Lecture 14

Yue Cheng

University of Virginia

Some material taken/derived from:

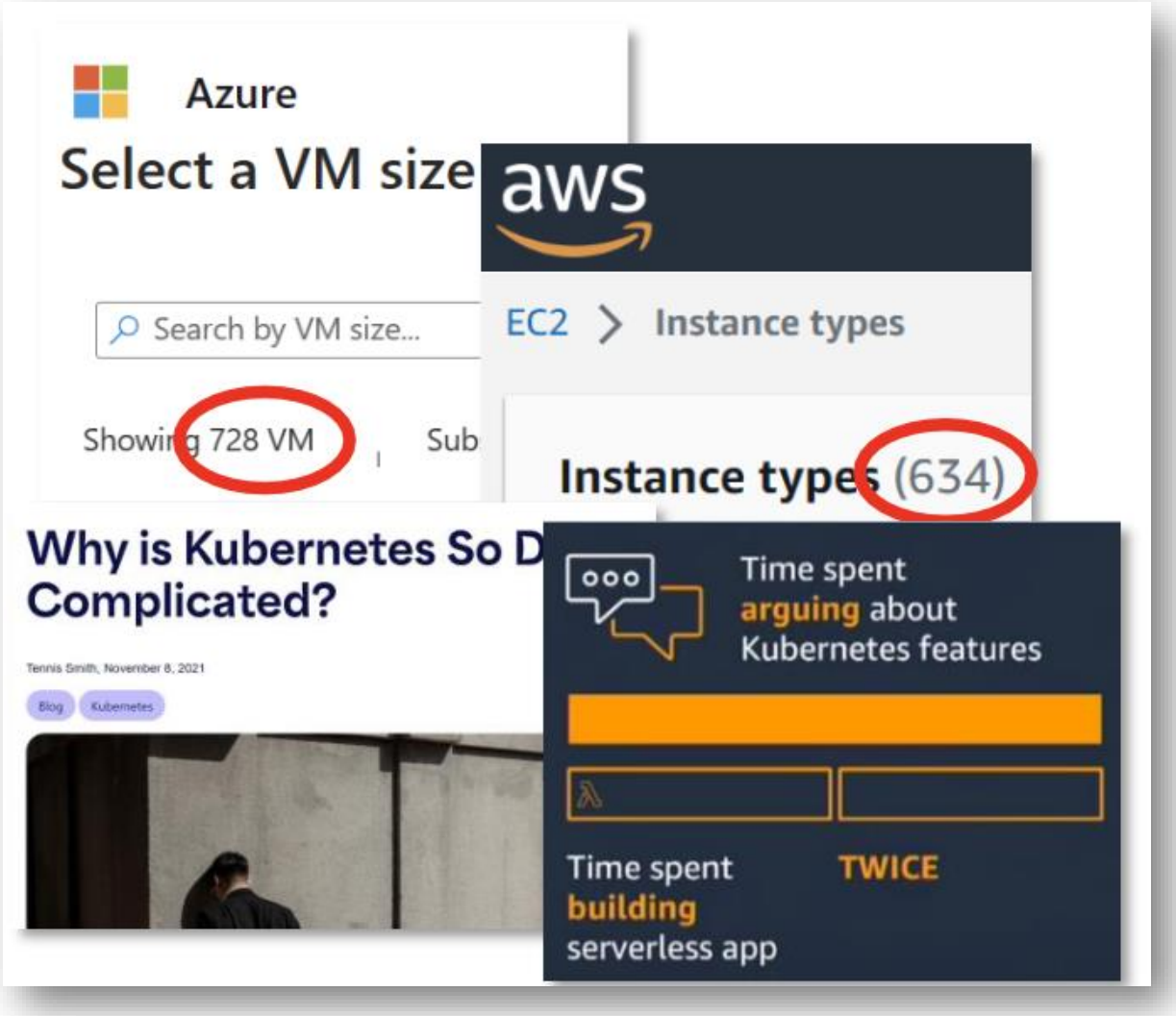
- Berkeley CS 262a (Spring '18) by Ali Ghodsi and Ion Stoica;
- Tyler Harter's HotCloud '18 OpenLambda talk;

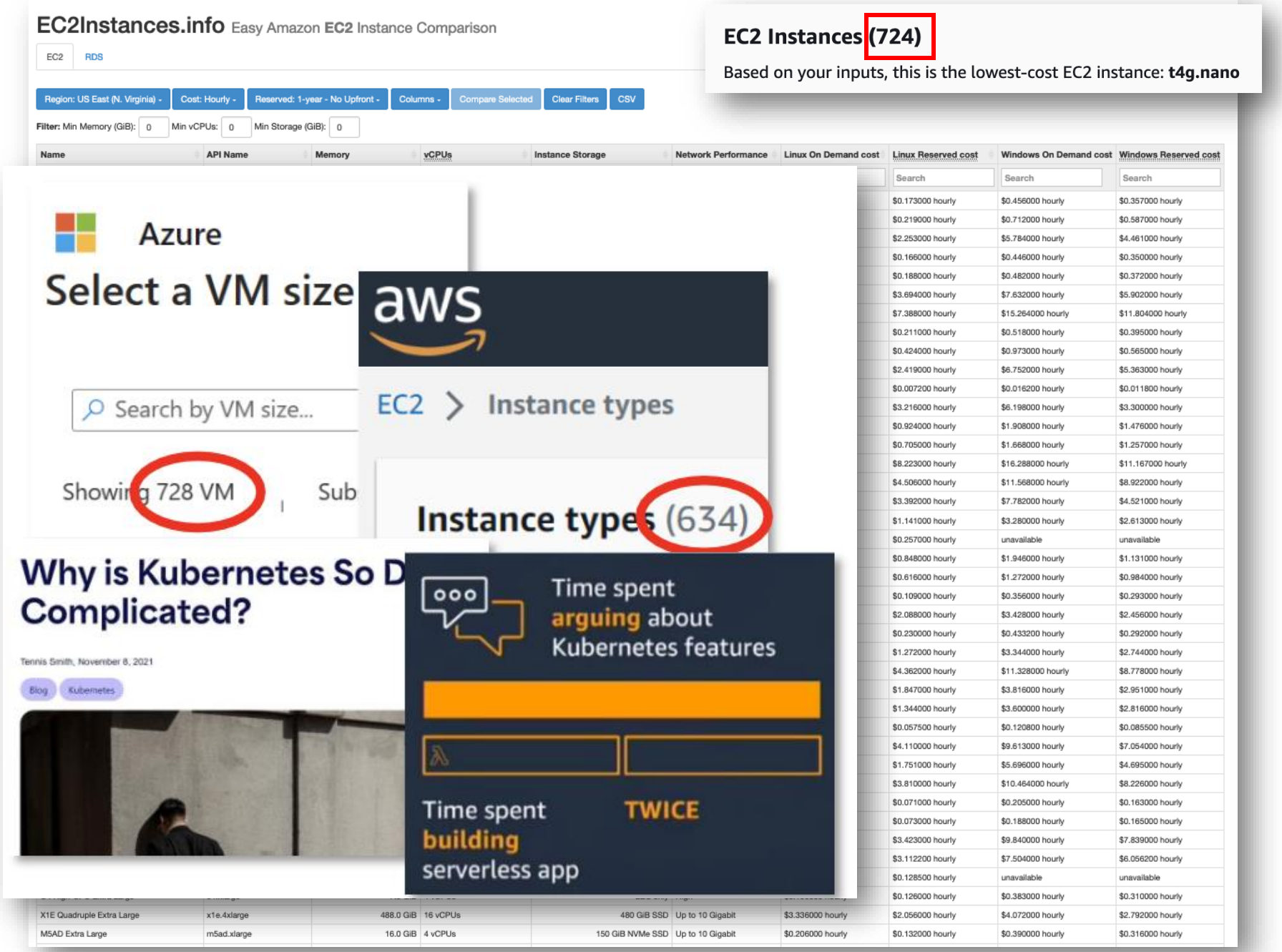
Motivation

When to use the cloud?

- Data
  - Large amounts of data – can't store locally
  - Shared data across users
  - Long-term storage

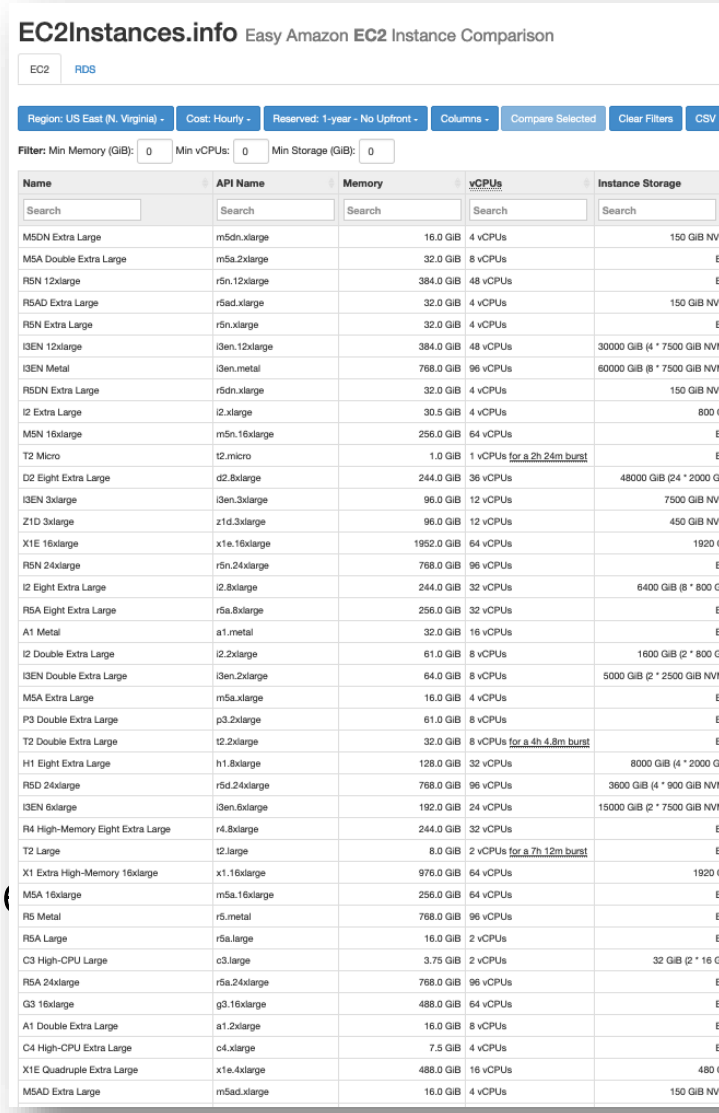
- Compute
  - Need lots of CPUs for data processing
  - Varying computing demands (resources)
  - No admin (for managing your local hardware)


Azure
Select a VM size
Search by VM size...
Showing 728 VM
Sub
aws
EC2 > Instance types
Instance types (634)
Why is Kubernetes So D
Complicated?
Tennis Smith, November 8, 2021
Blog
Kubernetes
Time spent
arguing about
Kubernetes features
Time spent
building
serverless app
TWICE


EC2Instances.info Easy Amazon EC2 Instance Comparison
EC2
RDS
Region: US East (N. Virginia)
Cost: Hourly
Reserved: 1-year - No Upfront
Columns
Compare Selected
Clear Filters
CSV
Filter: Min Memory (GiB): 0 Min vCPUs: 0 Min Storage (GiB): 0
EC2 Instances (724)
Based on your inputs, this is the lowest-cost EC2 instance: t4g.nano
Name
API Name
Memory
vCPUs
Instance Storage
Network Performance
Linux On Demand cost
Linux Reserved cost
Windows On Demand cost
Windows Reserved cost
Search
Search
Search
$0.173000 hourly $0.456000 hourly $0.357000 hourly
$0.219000 hourly $0.712000 hourly $0.587000 hourly
$2.253000 hourly $5.784000 hourly $4.461000 hourly
$0.166000 hourly $0.446000 hourly $0.350000 hourly
$0.188000 hourly $0.482000 hourly $0.372000 hourly
$3.694000 hourly $7.632000 hourly $5.902000 hourly
$7.388000 hourly $15.264000 hourly $11.804000 hourly
$0.211000 hourly $0.518000 hourly $0.395000 hourly
$0.424000 hourly $0.973000 hourly $0.565000 hourly
$2.419000 hourly $6.752000 hourly $5.363000 hourly
$0.007200 hourly $0.016200 hourly $0.011800 hourly
$3.216000 hourly $6.198000 hourly $3.300000 hourly
$0.924000 hourly $1.908000 hourly $1.476000 hourly
$0.705000 hourly $1.668000 hourly $1.257000 hourly
$8.223000 hourly $16.288000 hourly $11.167000 hourly
$4.506000 hourly $11.568000 hourly $8.922000 hourly
$3.392000 hourly $7.782000 hourly $4.521000 hourly
$1.141000 hourly $3.280000 hourly $2.613000 hourly
$0.257000 hourly unavailable unavailable
$0.848000 hourly $1.946000 hourly $1.131000 hourly
$0.616000 hourly $1.272000 hourly $0.984000 hourly
$0.109000 hourly $0.356000 hourly $0.293000 hourly
$2.088000 hourly $3.428000 hourly $2.456000 hourly
$0.230000 hourly $0.433200 hourly $0.292000 hourly
$1.272000 hourly $3.344000 hourly $2.744000 hourly
$4.362000 hourly $11.328000 hourly $8.778000 hourly
$1.847000 hourly $3.816000 hourly $2.951000 hourly
$1.344000 hourly $3.600000 hourly $2.816000 hourly
$0.057500 hourly $0.120800 hourly $0.085500 hourly
$4.110000 hourly $9.613000 hourly $7.054000 hourly
$1.751000 hourly $5.696000 hourly $4.695000 hourly
$3.810000 hourly $10.464000 hourly $8.226000 hourly
$0.071000 hourly $0.205000 hourly $0.163000 hourly
$0.073000 hourly $0.188000 hourly $0.165000 hourly
$3.423000 hourly $9.840000 hourly $7.839000 hourly
$3.112200 hourly $7.504000 hourly $6.056200 hourly
$0.128500 hourly unavailable unavailable
$0.126000 hourly $0.383000 hourly $0.310000 hourly
X1E Quadruple Extra Large x1e.4xlarge 488.0 GiB 16 vCPUs 480 GiB SSD Up to 10 Gigabit $3.336000 hourly $2.056000 hourly $4.072000 hourly $2.792000 hourly
M5AD Extra Large m5ad.xlarge 16.0 GiB 4 vCPUs 150 GiB NVMe SSD Up to 10 Gigabit $0.206000 hourly $0.132000 hourly $0.390000 hourly $0.316000 hourly
Azure
Select a VM size
Search by VM size...
Showing 728 VM
Sub
aws
EC2 > Instance types
Instance types (634)
Why is Kubernetes So D
Complicated?
Tennis Smith, November 8, 2021
Blog
Kubernetes
Time spent arguing about Kubernetes features
Time spent building serverless app
TWICE

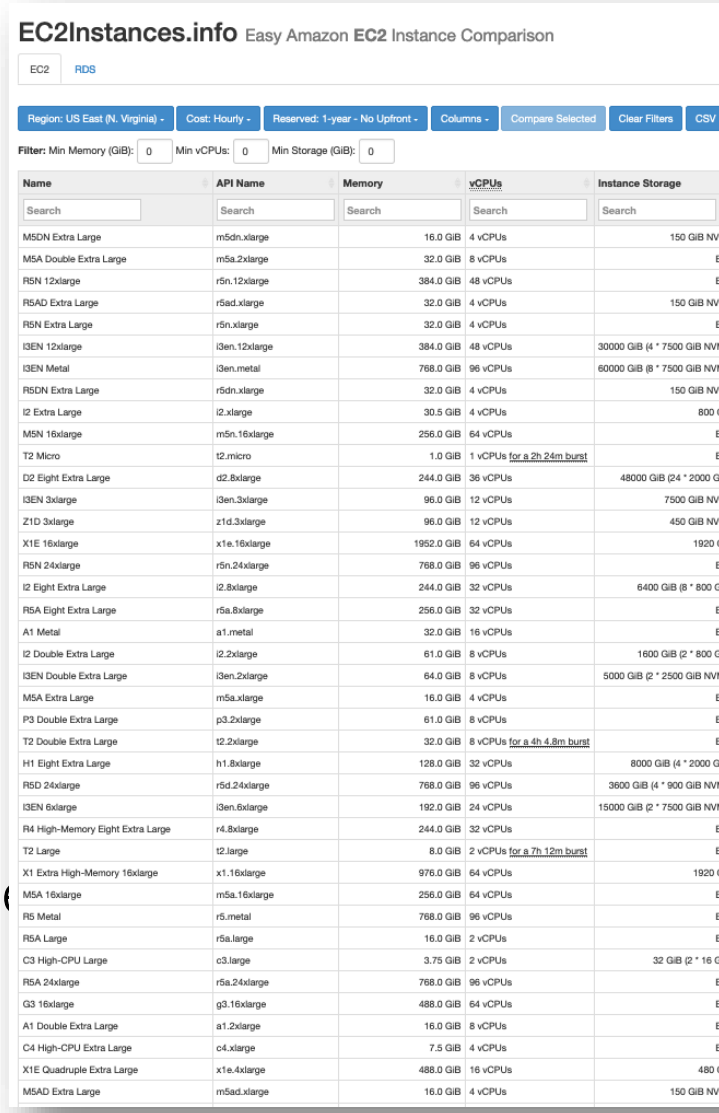
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



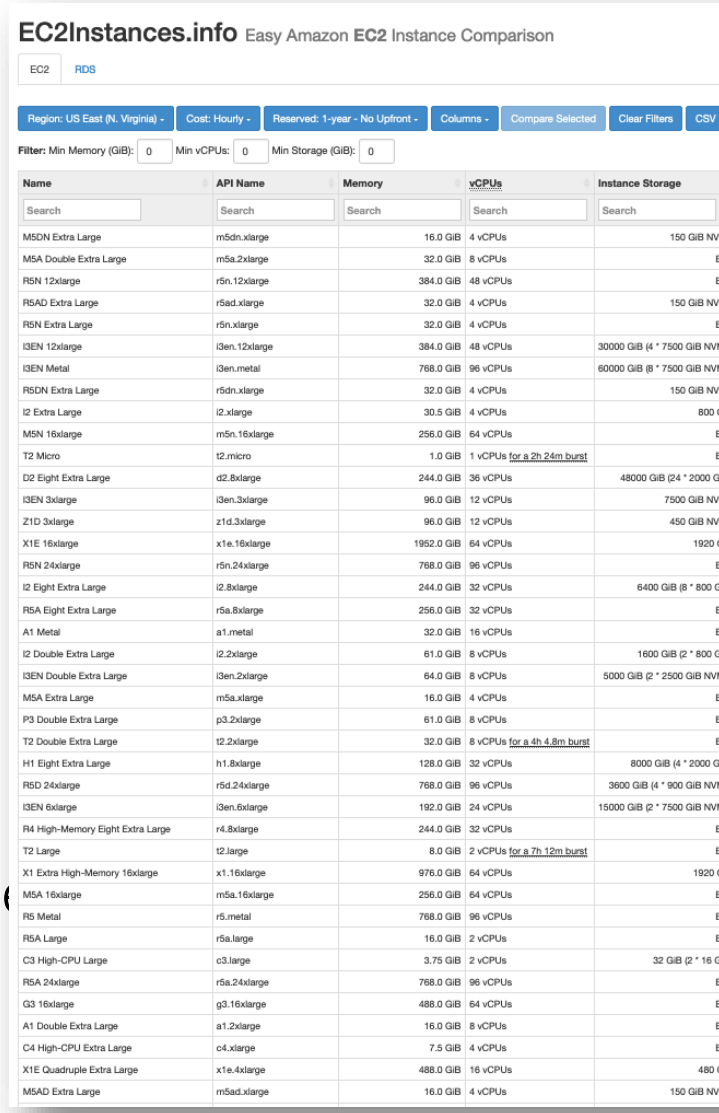
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



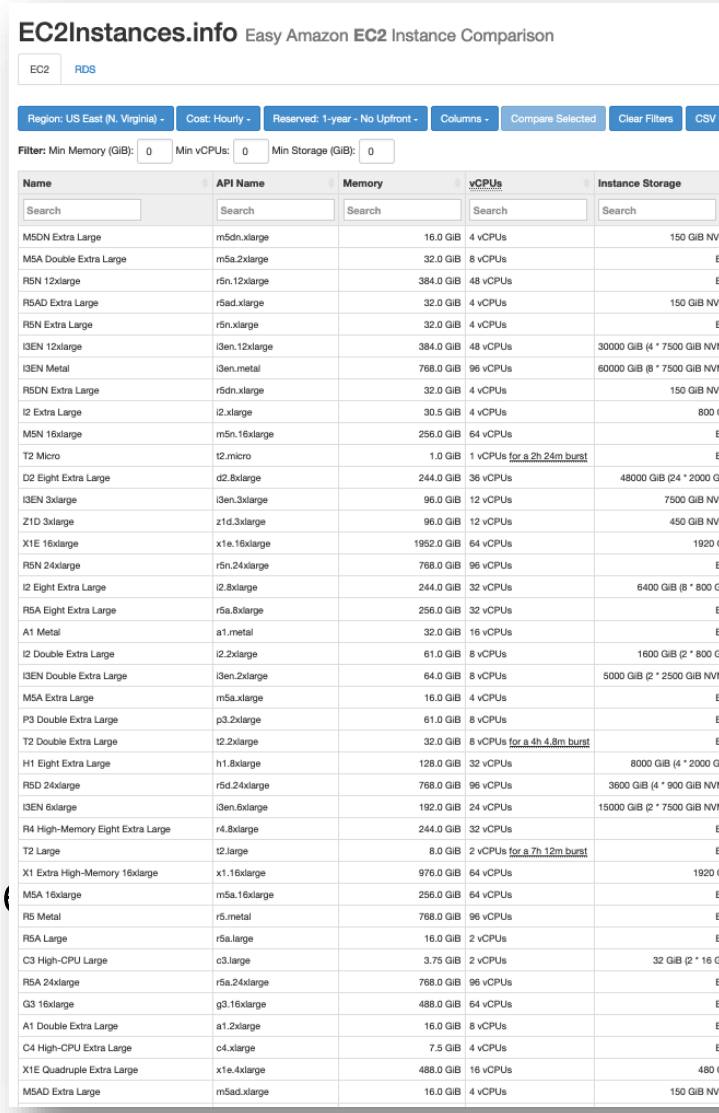
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



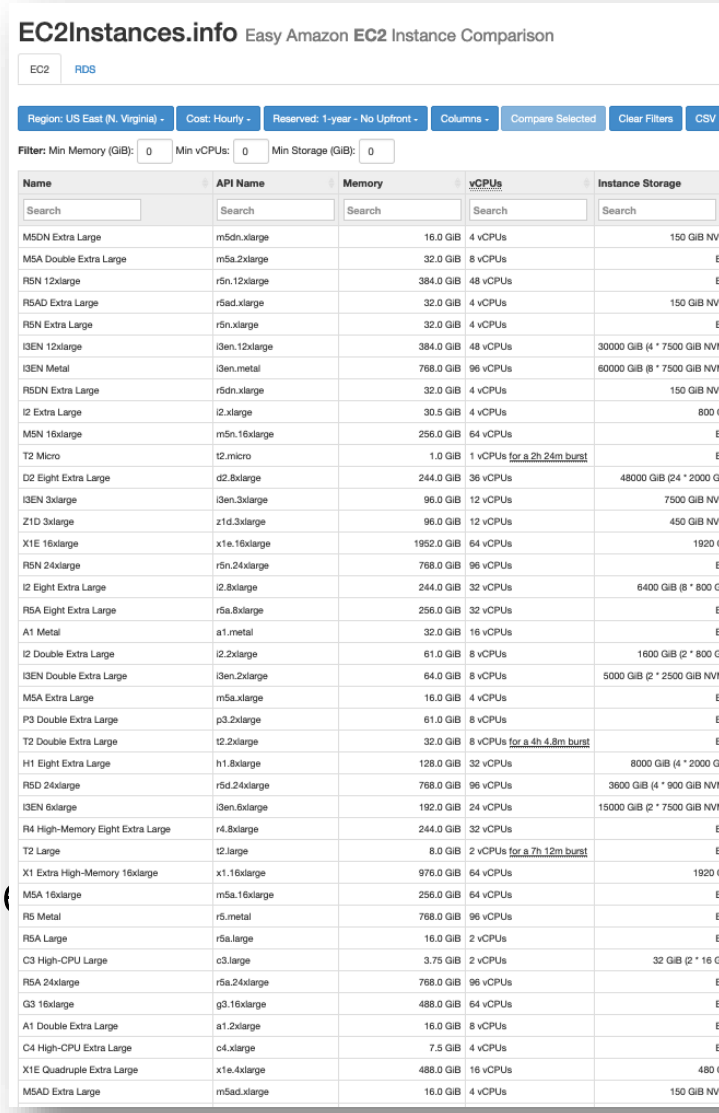
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



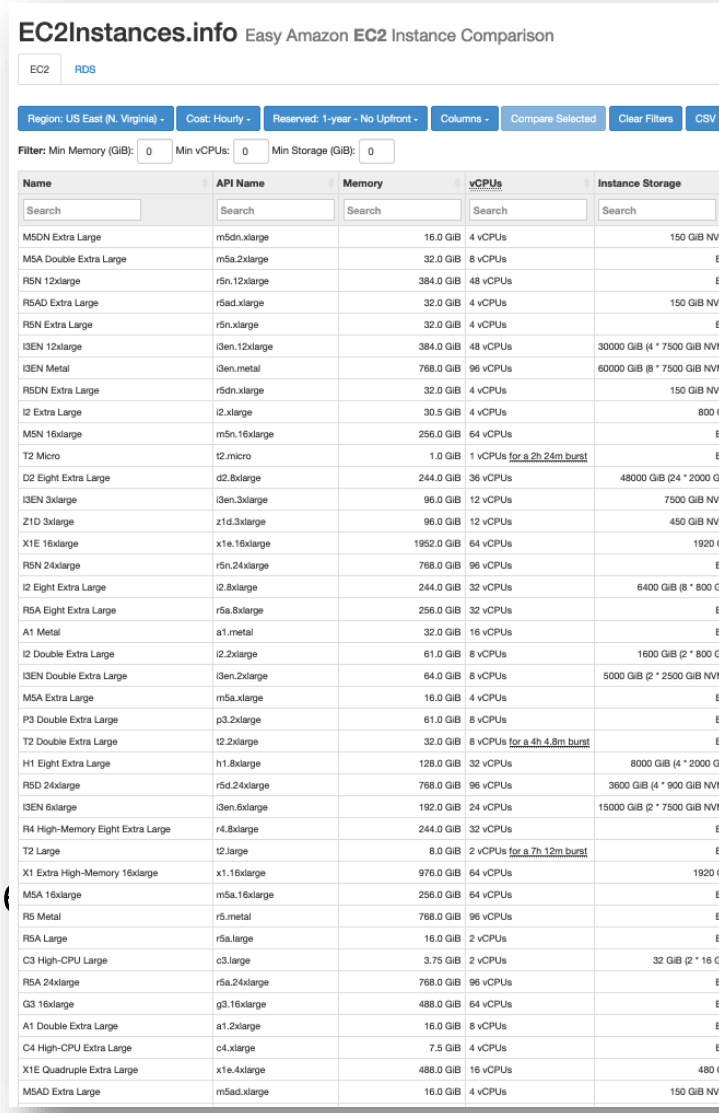
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



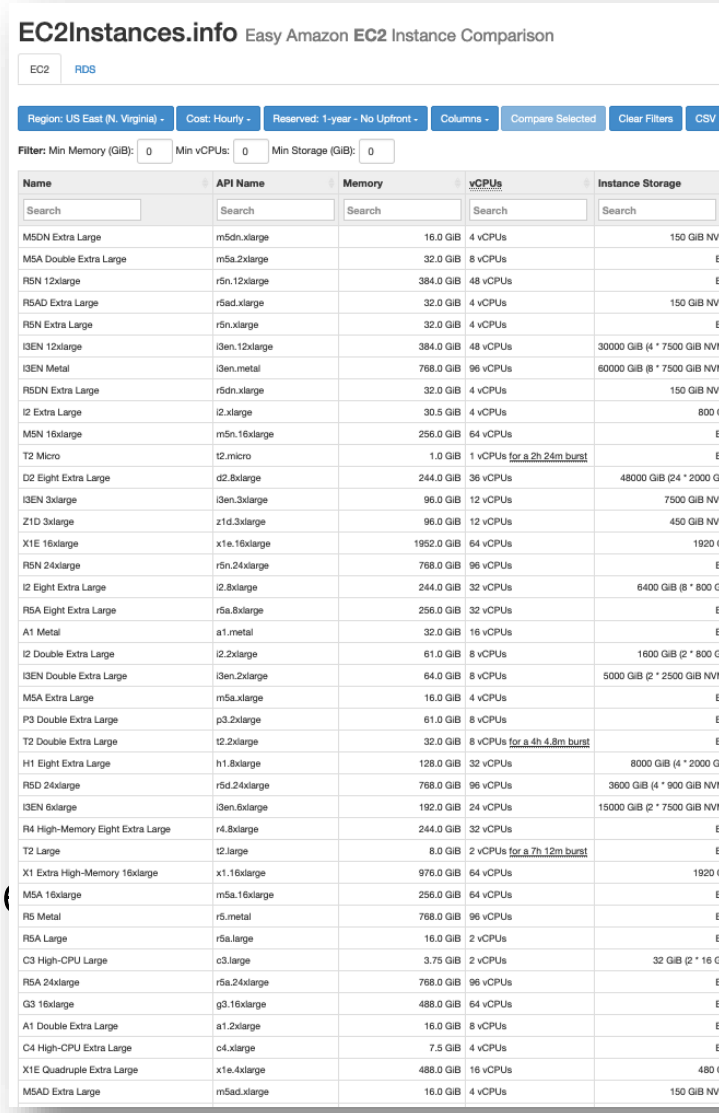
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



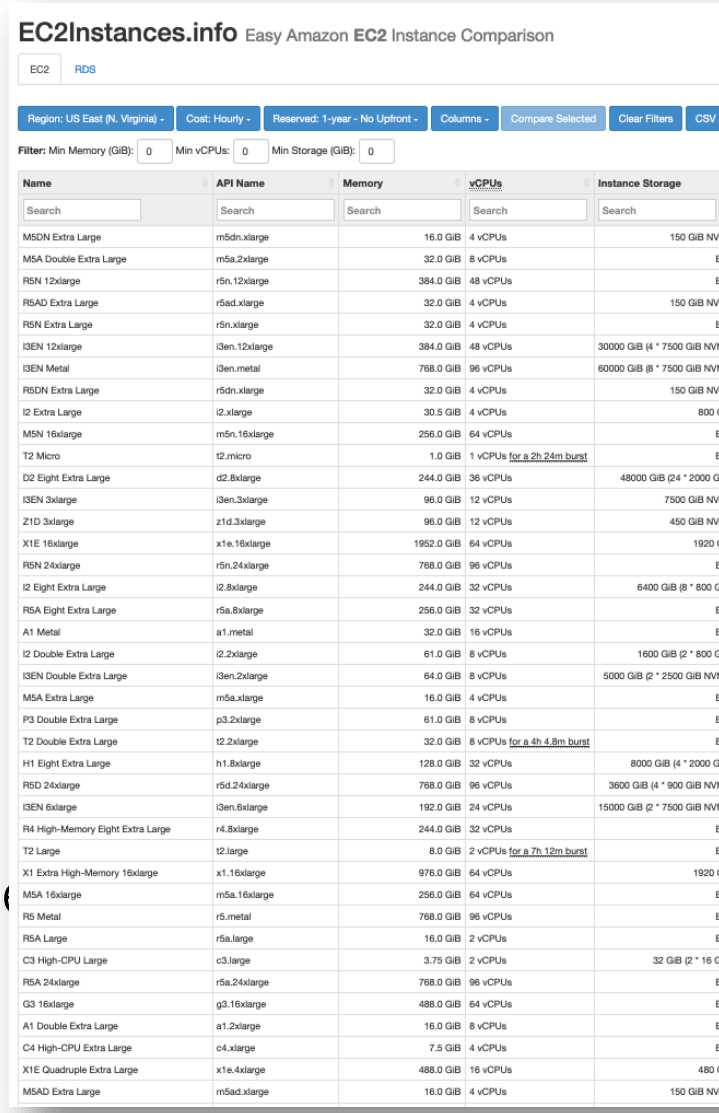
#thecloudistoodamnhard

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?
8. Go back to Step 1…



#thecloudistoodamnhard

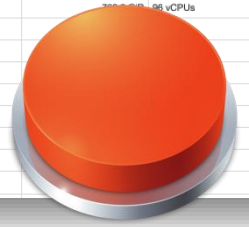
1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
   1. When to scale out?
   2. When to scale in?
   3. When to switch to different instance types?
8. Go back to Step 1…



#thecloudistoodamn

1. What type of instances?
2. How many to spin up?
3. What base image?
4. On-demand or spot?
5. What storage service to use?
6. And then wait to start…
7. Not the end of the horror story:
    1. When to scale out?
    2. When to scale in?
    3. When to switch to different instance types?

R5A Eight Extra Large	r5a.8xlarge	256.0 GiB	32 vCPUs	E
A1 Metal	a1.metal	32.0 GiB	16 vCPUs	E
I2 Double Extra Large	i2.2xlarge	61.0 GiB	8 vCPUs	1600 GiB (2 * 800 G
I3EN Double Extra Large	i3en.2xlarge	64.0 GiB	8 vCPUs	5000 GiB (2 * 2500 GiB NVI
M5A Extra Large	m5a.xlarge	16.0 GiB	4 vCPUs	E
P3 Double Extra Large	p3.2xlarge	61.0 GiB	8 vCPUs	E
T2 Double Extra Large	t2.2xlarge	32.0 GiB	8 vCPUs for a 4h 4.8m burst	E
H1 Eight Extra Large	h1.8xlarge	128.0 GiB	32 vCPUs	8000 GiB (4 * 2000 G
R5D 24xlarge	r5d.24xlarge	768.0 GiB	96 vCPUs	3600 GiB (4 * 900 GiB NVI
I3EN 6xlarge	i3en.6xlarge	192.0 GiB	24 vCPUs	15000 GiB (2 * 7500 GiB NVI
R4 High-Memory Eight Extra Large	r4.8xlarge	244.0 GiB	32 vCPUs	E
T2 Large	t2.large	8.0 GiB	2 vCPUs for a 7h 12m burst	E
X1 Extra High-Memory 16xlarge	x1.16xlarge	976.0 GiB	64 vCPUs	1920 G
M5A 16xlarge	m5a.16xlarge	256.0 GiB	64 vCPUs	E
R5 Metal	r5.metal		96 vCPUs	E
R5A Large	r5a.large			E
				32 GiB (2 * 16 G
				E
				E
				E
				E
				480 G
M5AD Extra Large	m5ad.xlarge			150 GiB NV



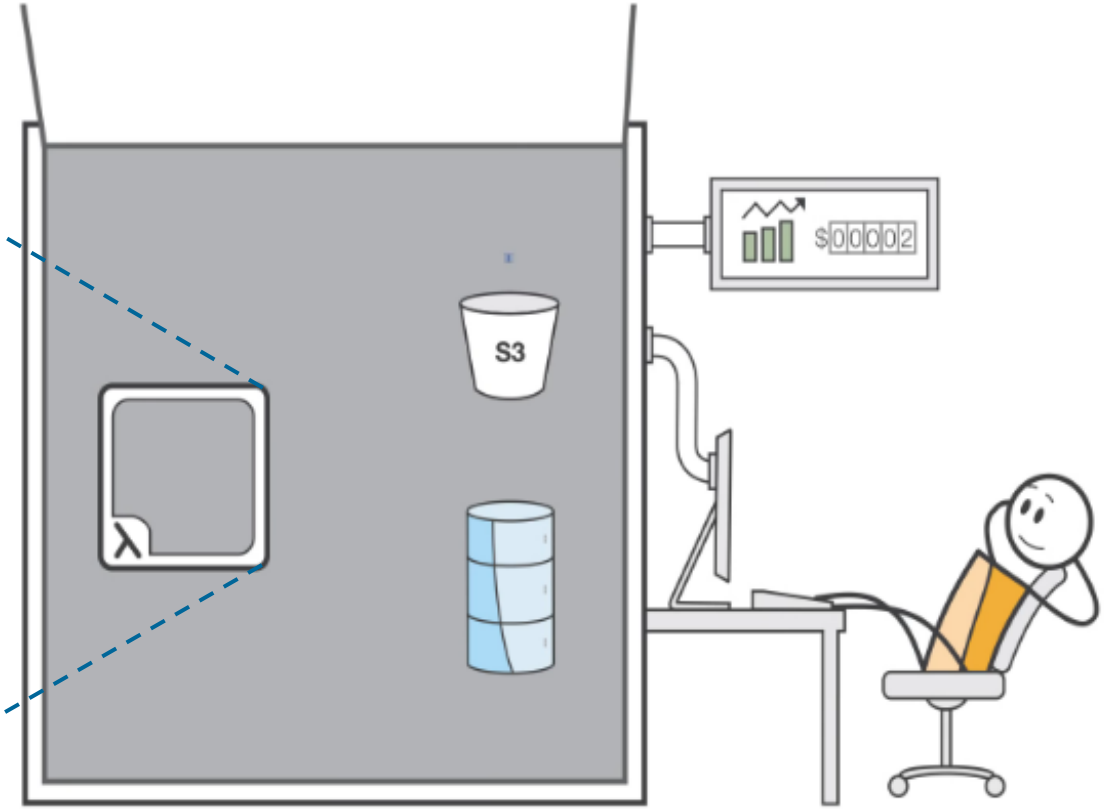
Why is there no “cloud button”?

~~Decision paralysis??~~
Go for Serverless!

Amazon
Lambda

Google Cloud Functions

Microsoft Azure Functions



What is serverless computing?

What is serverless computing?

Serverless computing (Function-as-a-Service, or FaaS) is a programming abstraction that enables users to upload programs, run them at (virtually) any scale, and pay only for the resources used

A car analogy

https://www.slideshare.net/loige/building-a-serverless-company-with-nodejs-react-and-the-serverless-framework-jsday-2017-verona

Concept of serverless is not new

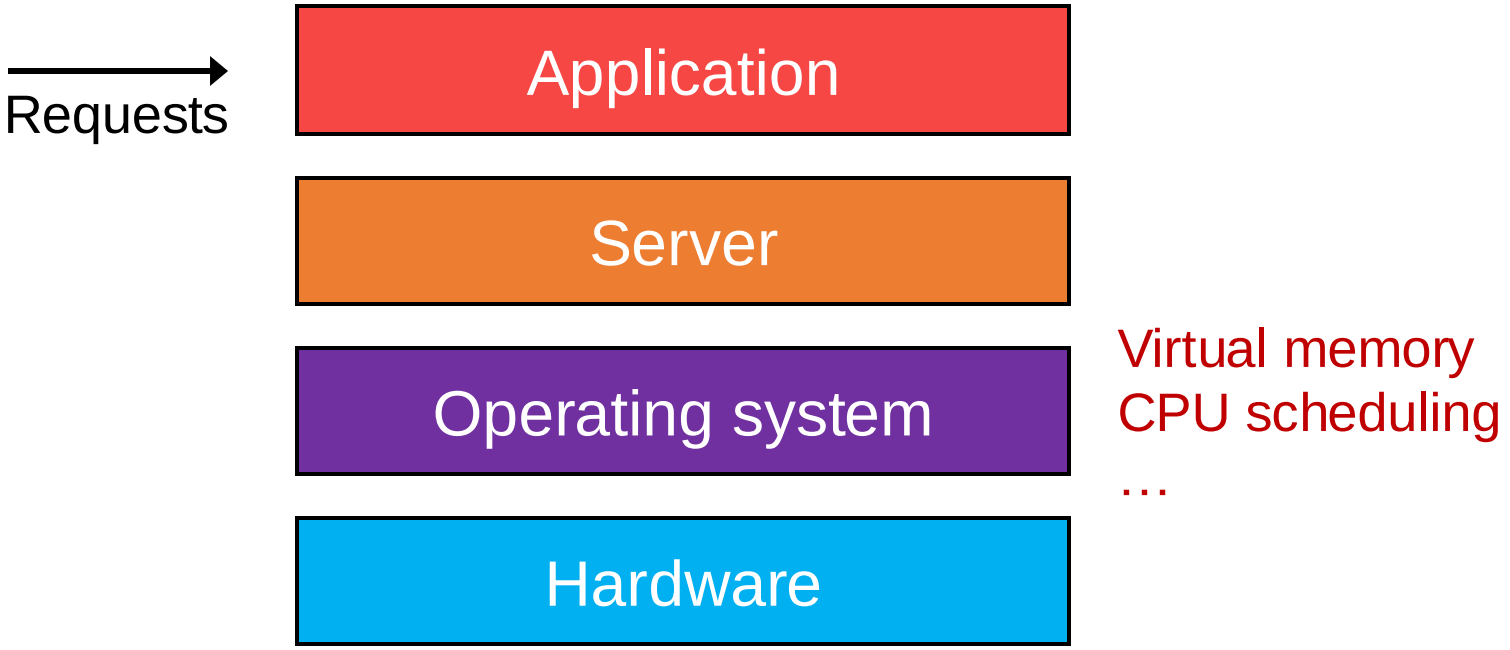
- Google App Engine
  - Fully managed platform as a service (PaaS) for developing and hosting web applications

- Google BigQuery
  - Fully managed data warehouse
  - “Arbitrarily” large data and queries
  - Pay per byte being processed
  - No concept of server or cluster

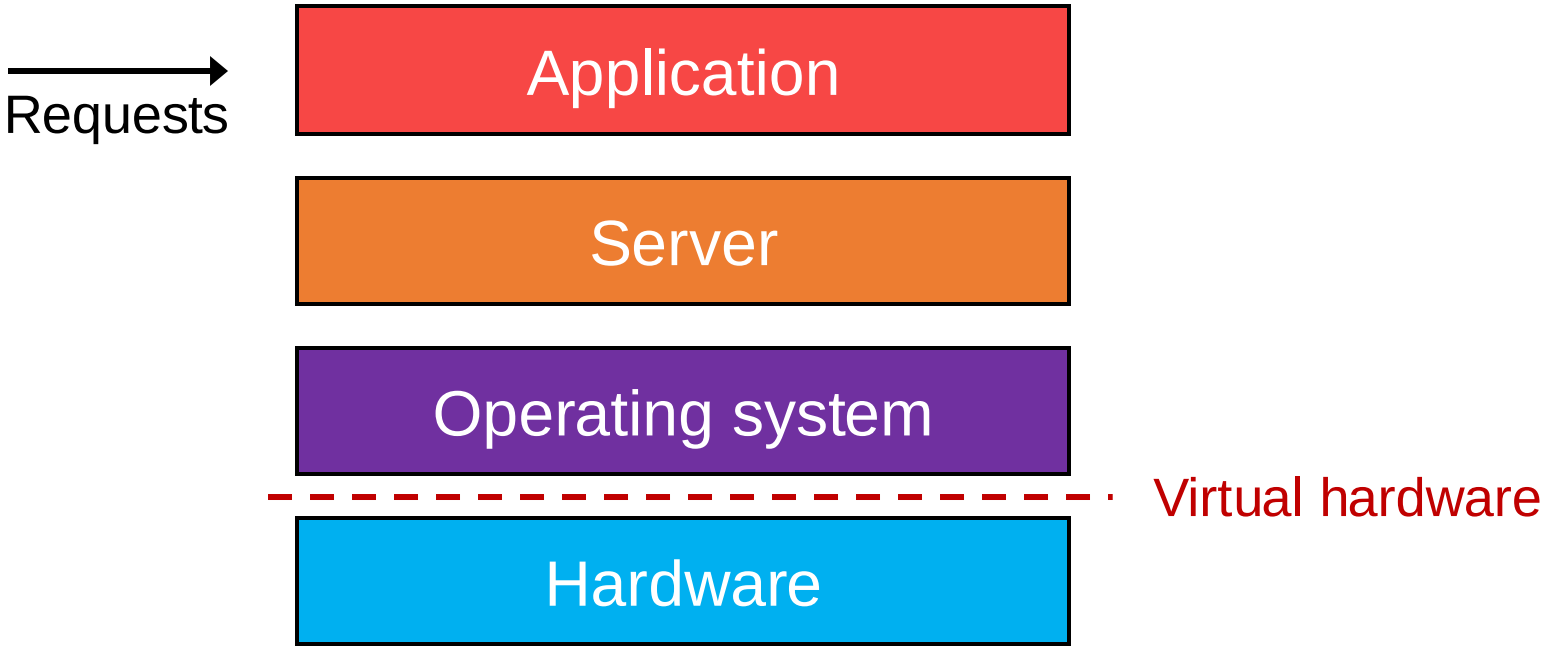
- AWS S3
  - Fully managed object storage service
  - Pay per byte being stored and written
  - No server maintenance or resource scaling

Recap: Cloud evolution history – A virtualization story

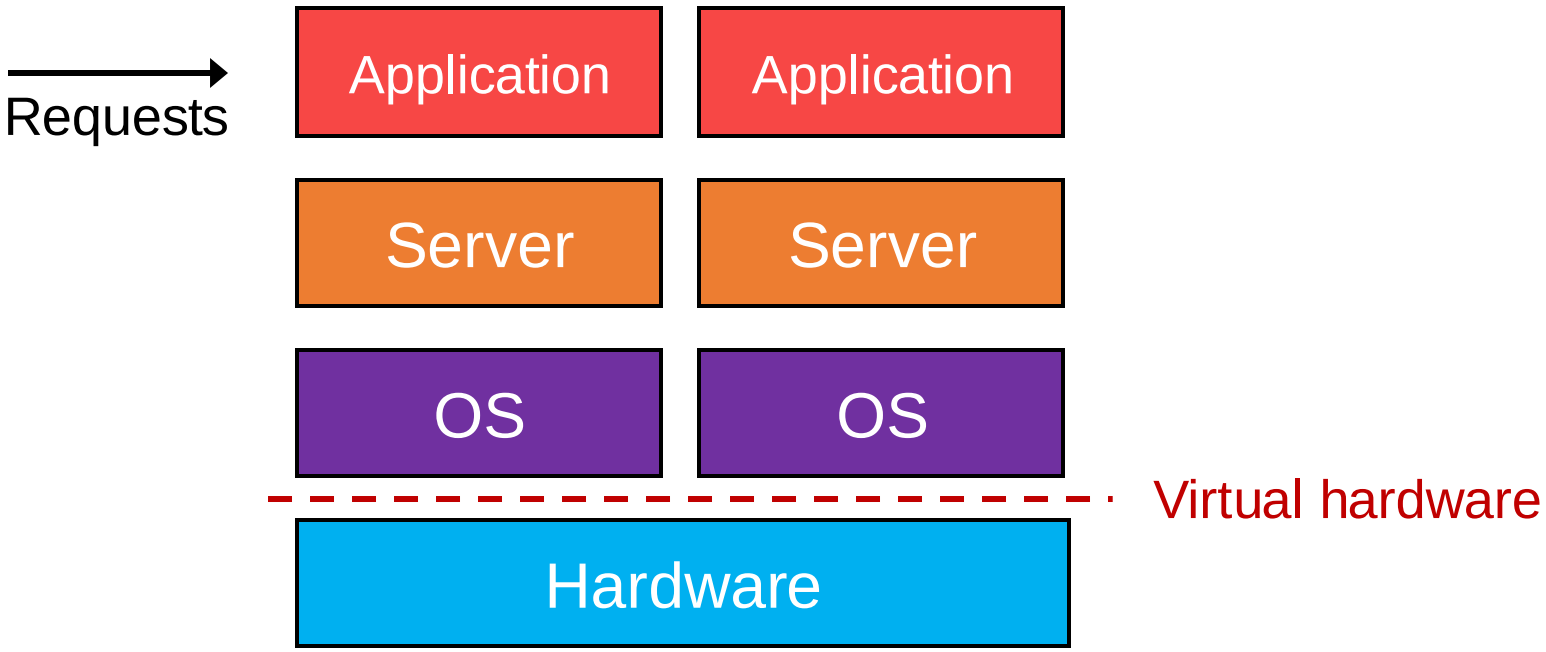
Classic cloud app stack



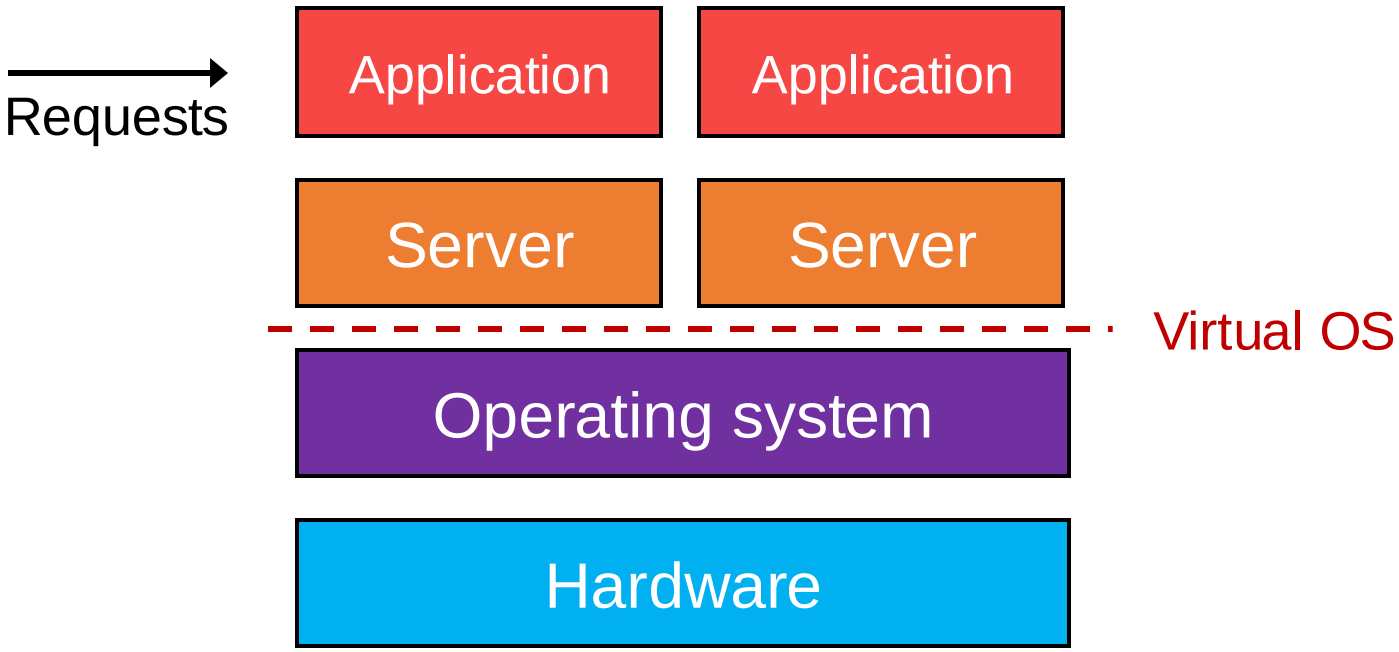
1st generation: virtual machine (VM)



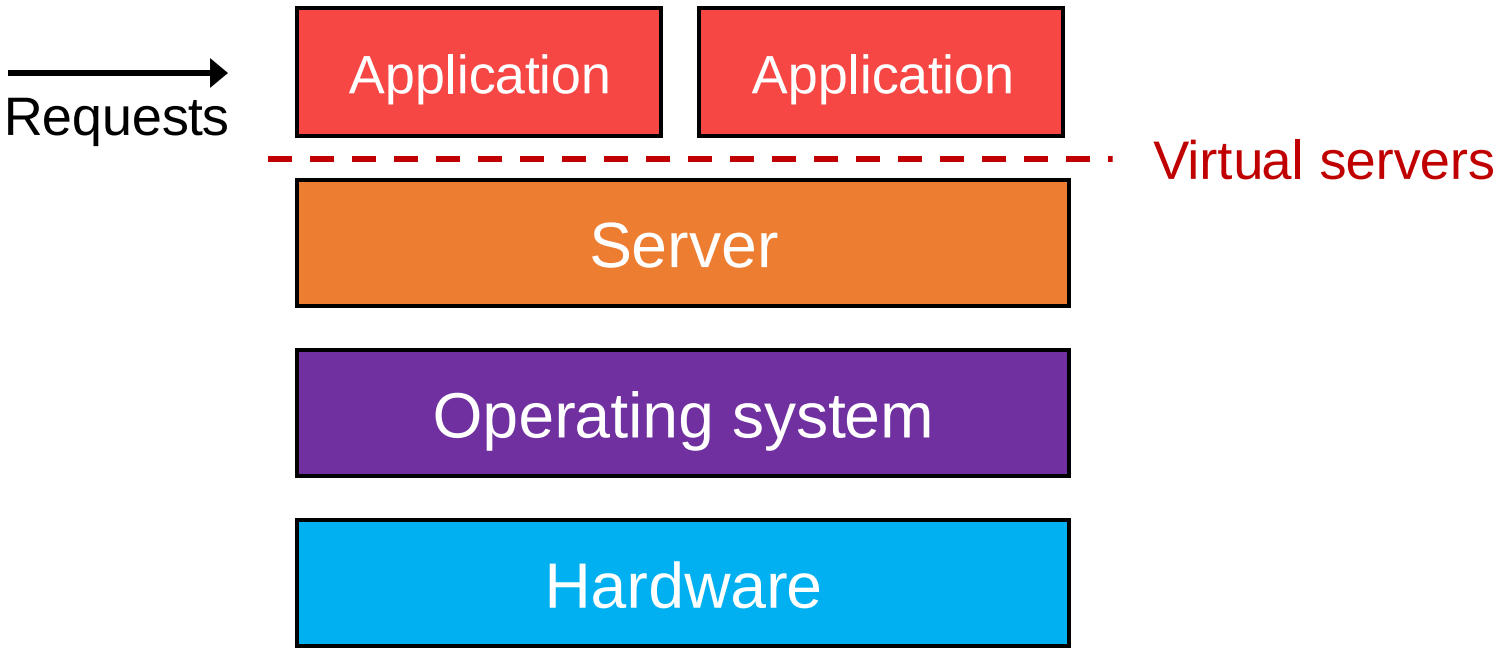
1st generation: virtual machine (VM)



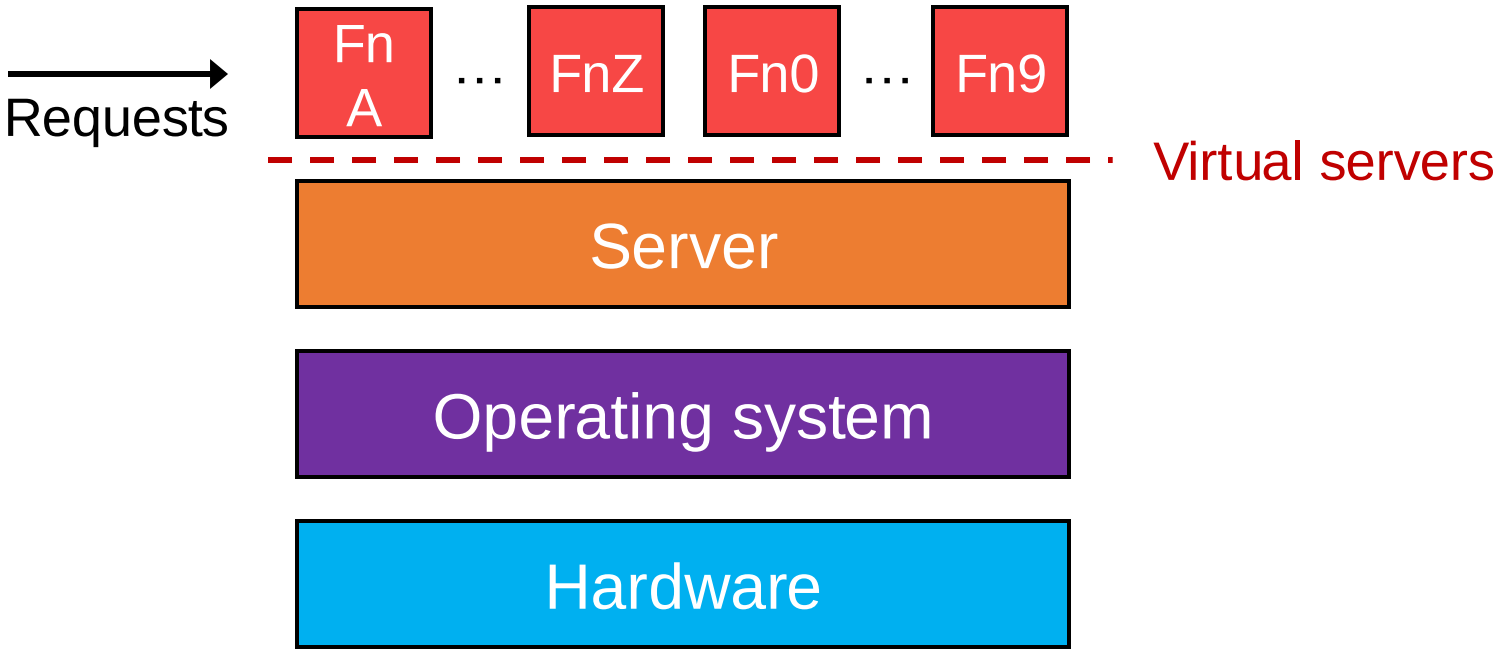
2nd generation: containers



3rd generation: serverless functions



3rd generation: serverless functions



Tradeoff discussion

Serverless functions (AWS Lambdas)

Containers

VMs

Isolation?

Flexibility?

Overhead?

Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions

Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source

Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source
3. Determine function(s) to which to dispatch the event

Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source
3. Determine function(s) to which to dispatch the event
4. Find an existing instance of function or create a new one

Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source
3. Determine function(s) to which to dispatch the event
4. Find an existing instance of function or create a new one
5. Send the event to the function instance

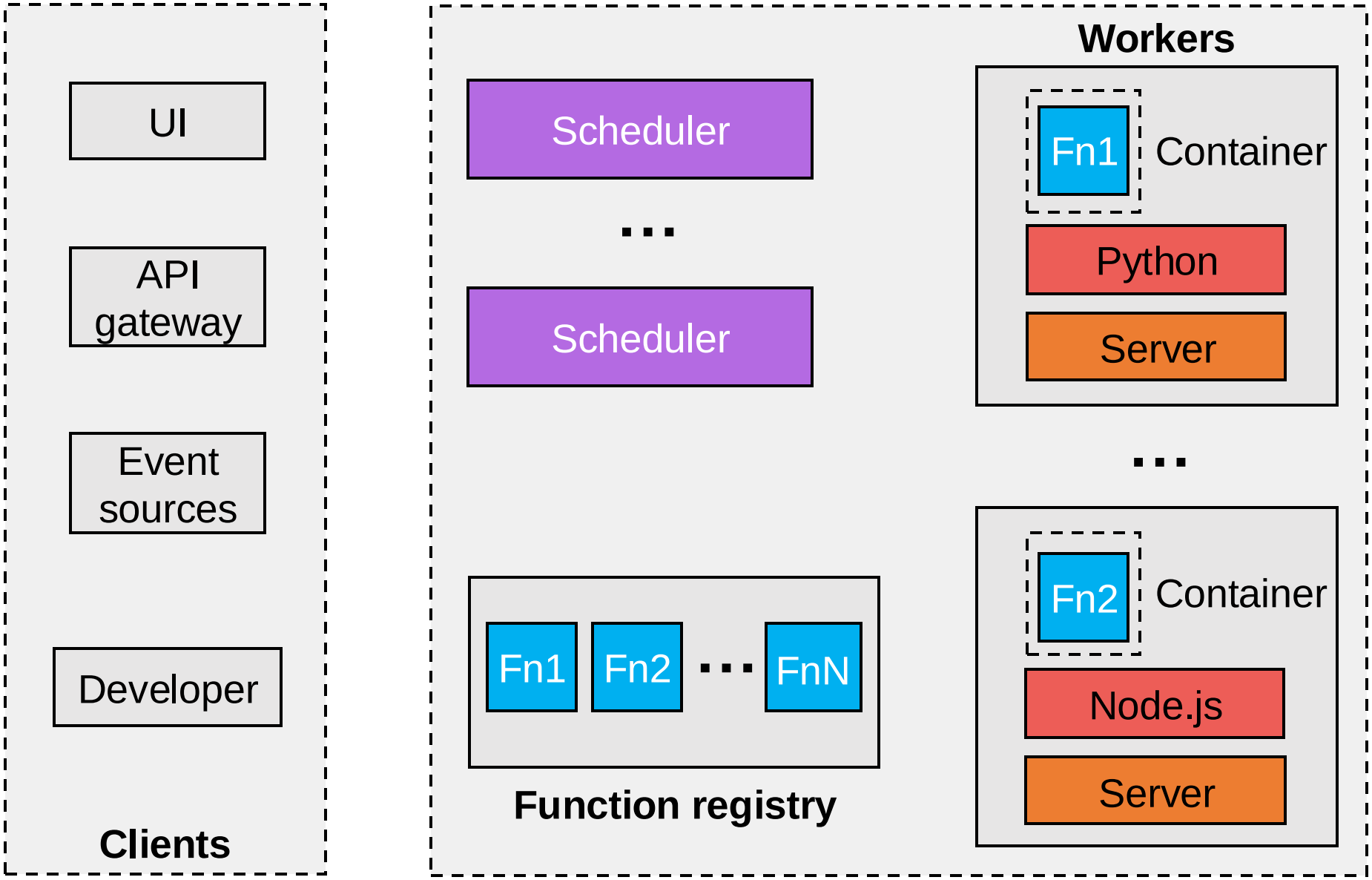
Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source
3. Determine function(s) to which to dispatch the event
4. Find an existing instance of function or create a new one
5. Send the event to the function instance
6. Wait for a response

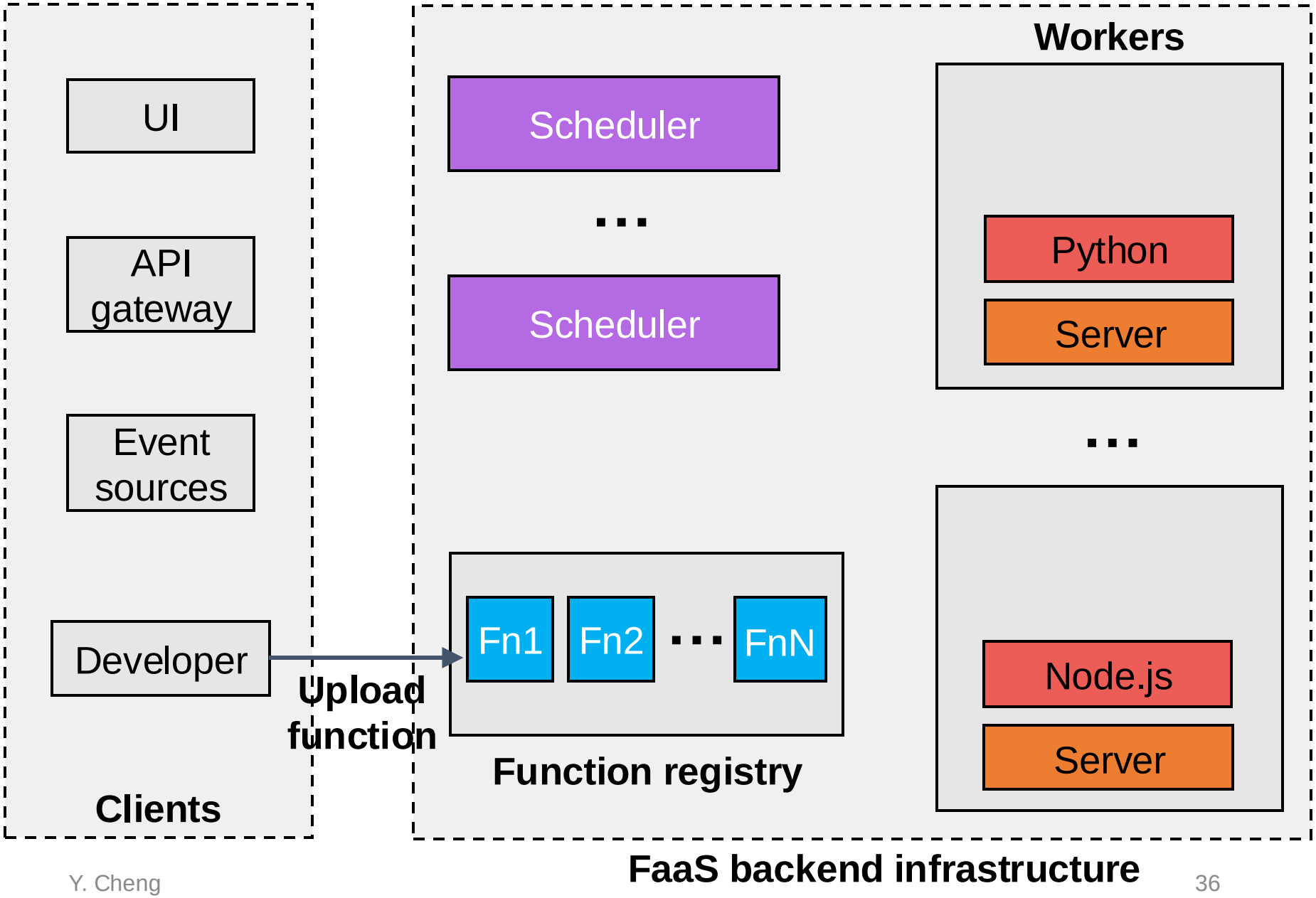
Above the surface: Core capability

1. (Provider) Manage a set of user-defined functions
2. Take an event sent over HTTP or received from an event source
3. Determine function(s) to which to dispatch the event
4. Find an existing instance of function or create a new one
5. Send the event to the function instance
6. Wait for a response
7. Gather execution logs
8. Make the response available to the user
9. Stop the function when the execution terminates

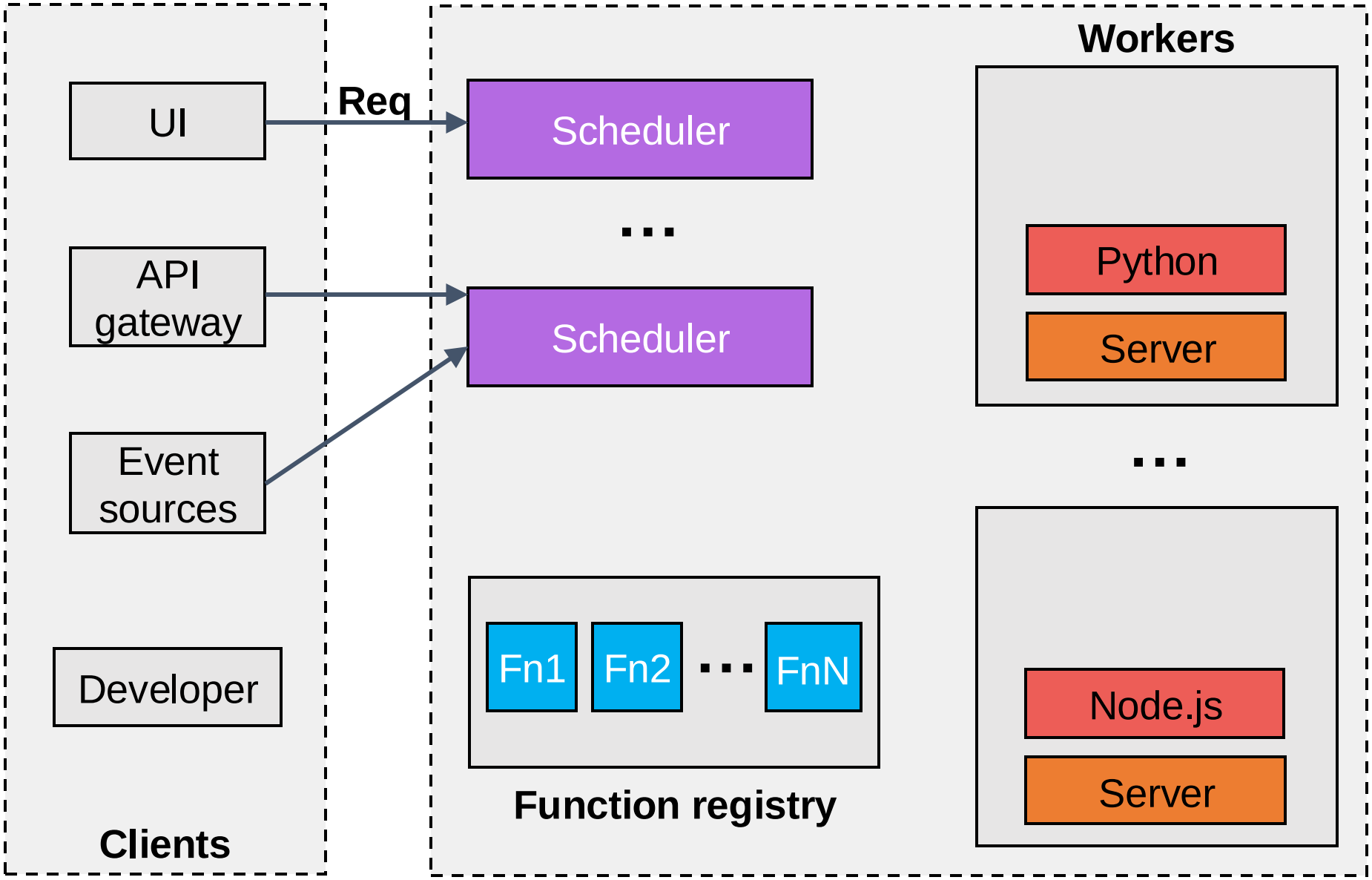
Under the hood: FaaS architecture



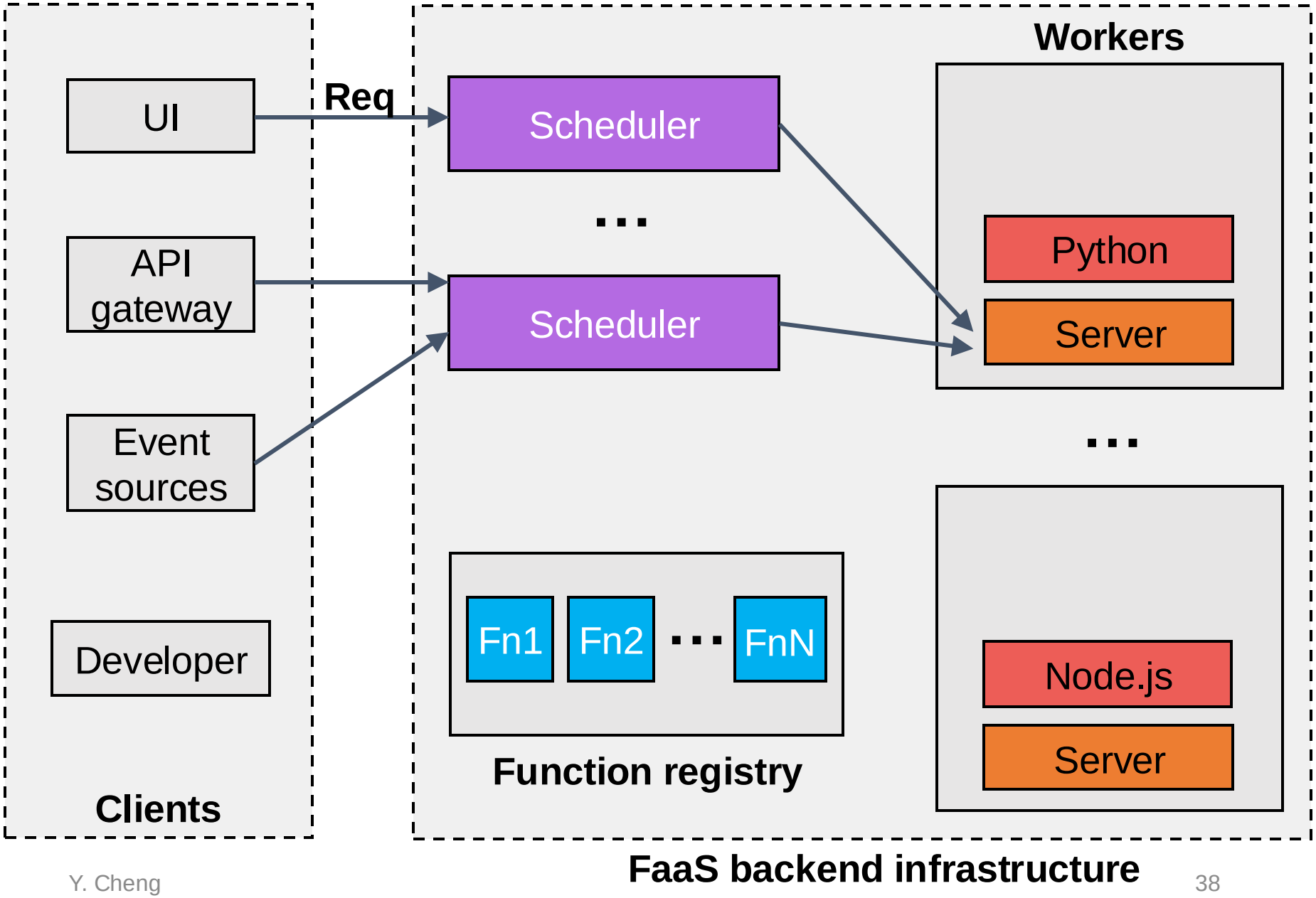
Under the hood: FaaS architecture



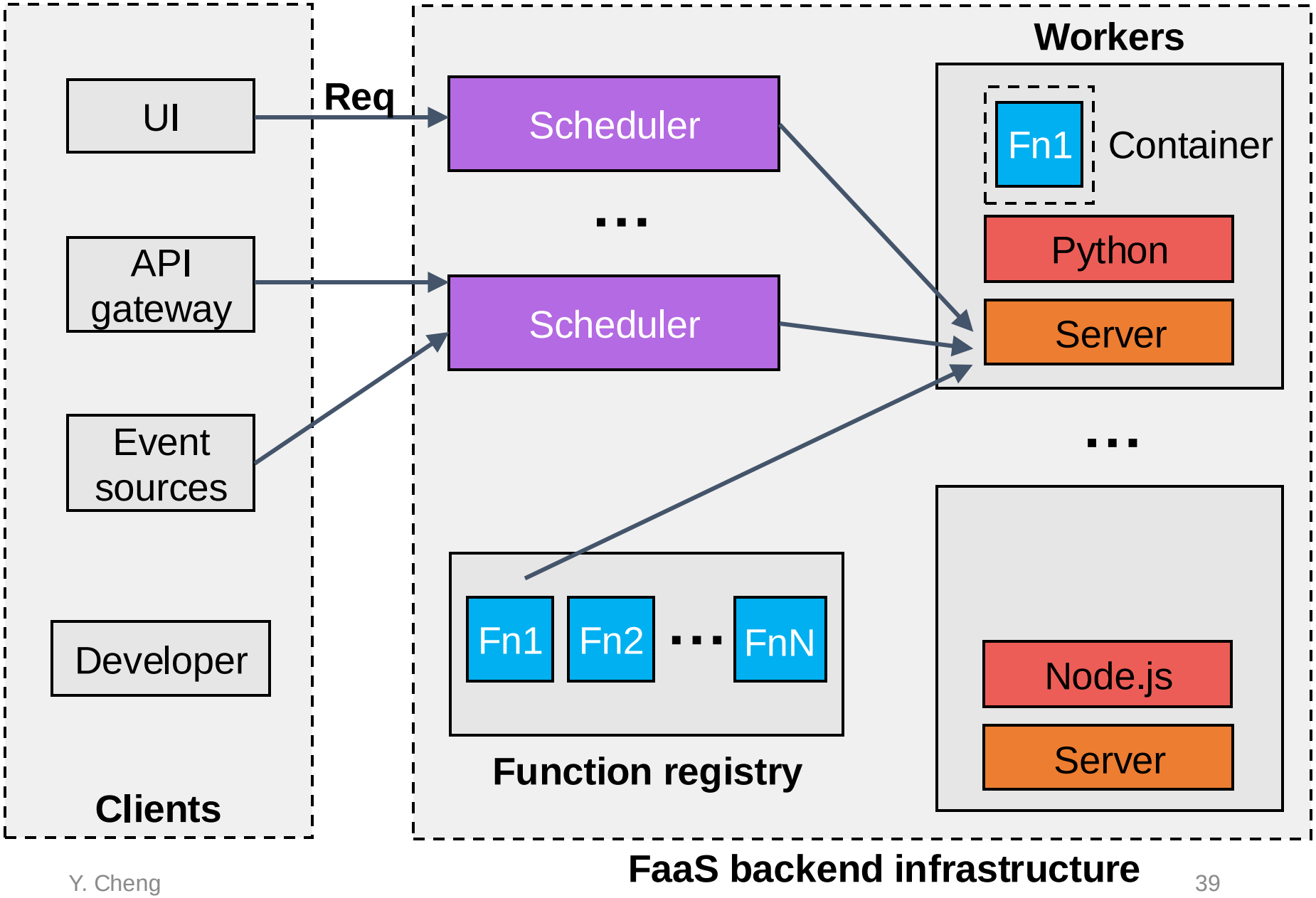
Under the hood: FaaS architecture



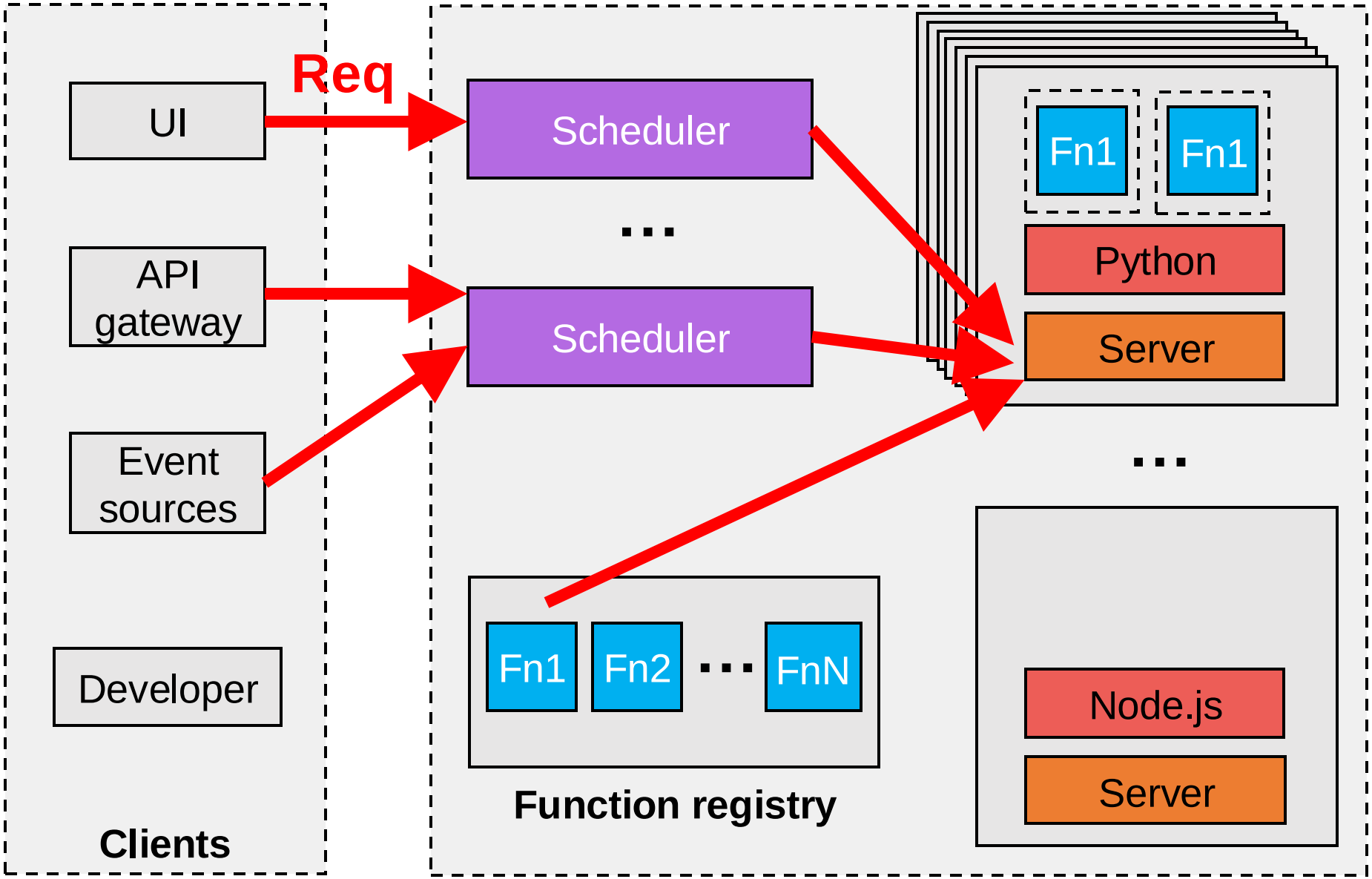
Under the hood: FaaS architecture



Under the hood: FaaS architecture



Under the hood: FaaS architecture



AWS Lambda

- Lambda capacity config keeps **evolving**:

1st gen:
- ~~300 seconds~~ 900 seconds (15 minutes)
- ~~single-core~~ ~~two-core~~ → up to 6 cores
- ~~1.5 GB~~ → 10 GB memory
- ~~512 MB~~ → up to 10GB of /tmp file system

Current offering

Python, Java, Node.js, Go, …

Pricing:

- Fine-grained billing: 1-millisecond billed duration
- $0.20 per 1M requests (invocations charge $)
- $0.0000166667 for every GB-second (compute time charges $$)
- 6,000 1 GB Lambda functions for one second: 10¢

Desirable properties

- (Near) zero administration overhead
  - “No-ops”: No need to handle server provisioning / infrastructure, failure, etc.
- Elastic auto-scaling
  - Spin up / tear down functions quickly based on load
- Pay-per-use
  - Only pay for the resources used (CPU-mem bundle)

Limitations

- **Banned** inbound network
- **No guaranteed** data availability
- Lambdas are **resource-constrained**
- Lambdas have **limited execution time**
- **High cold startup cost** and **invocation cost**