#5 - Computing and Computing at Scale

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

5min Recap

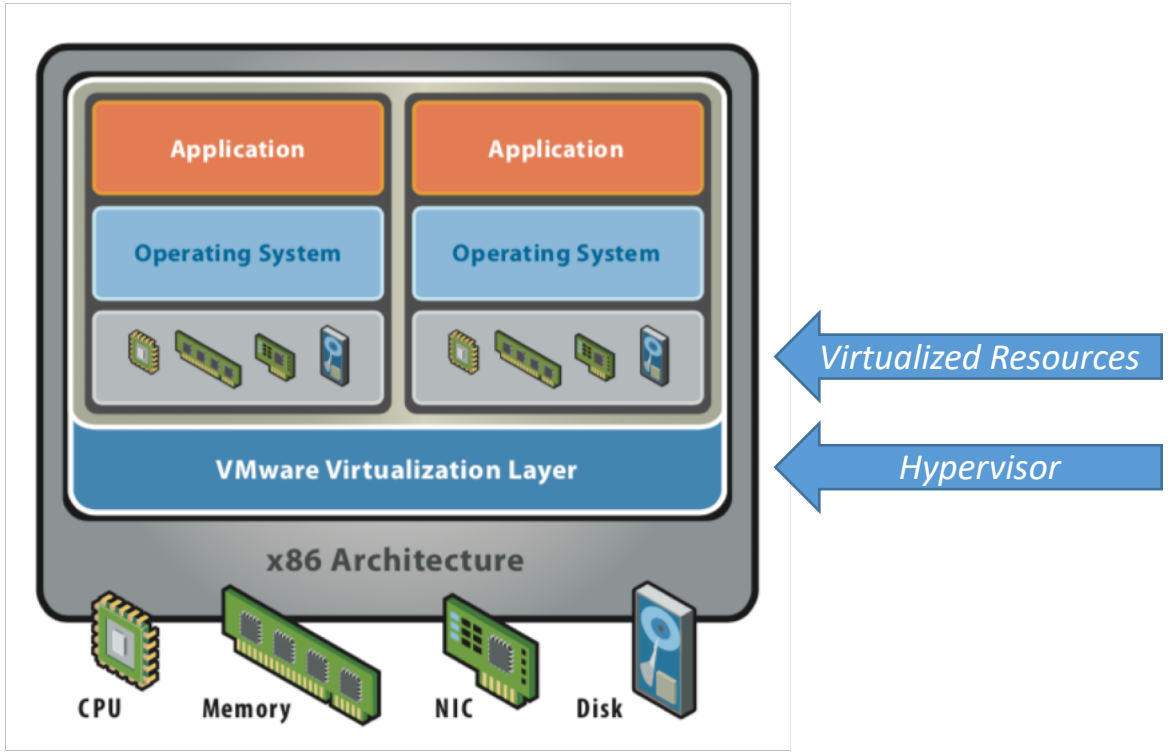
EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Virtual Machines

Virtual Machine → OS on top of **virtualized**
CPU, RAM… (via Hypervisor)

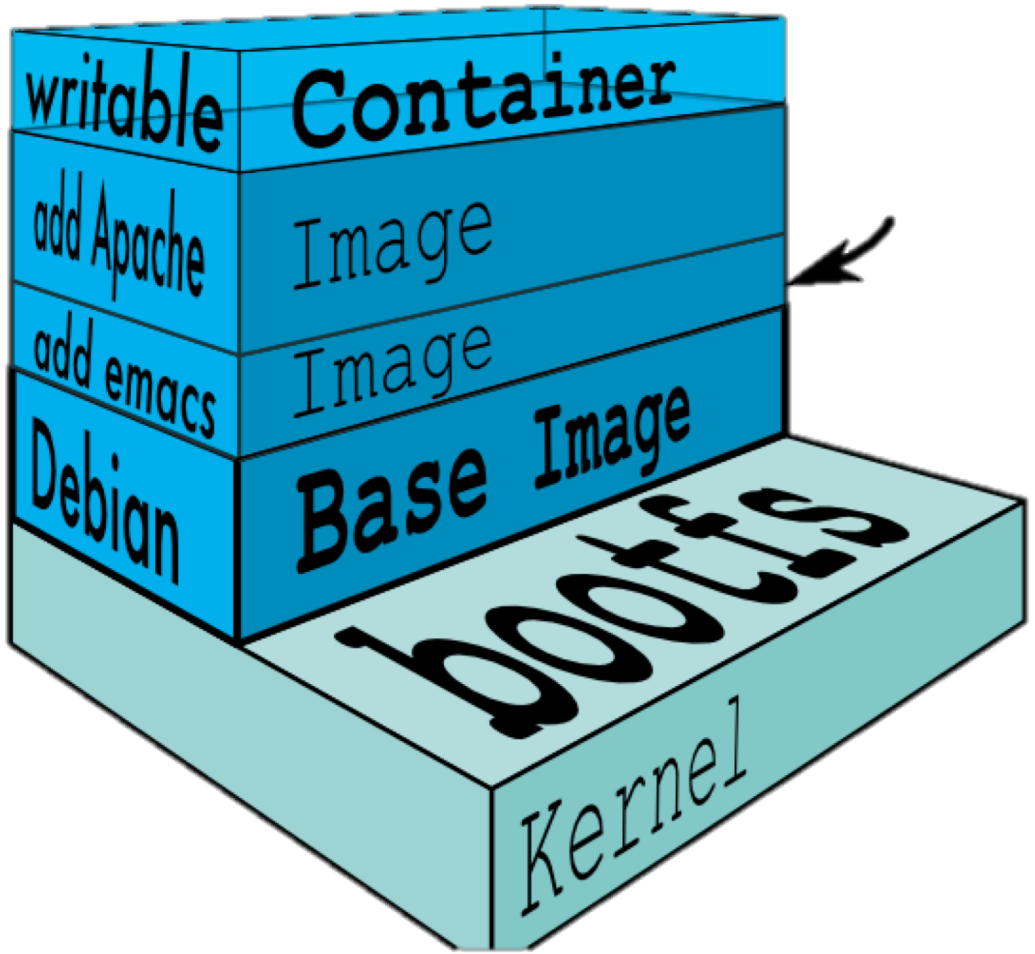
== Software/Hardware/Firmware.

- VMs run on a Hypervisor
- Hypervisor runs on the Guest Machine
- Hypervisor provides VM with resources (CPU, RAM, Disk, Net, GPU)
- Hypervisor (Hosted vs. Bare-Metal)



Source: VMWare

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE



Docker Container

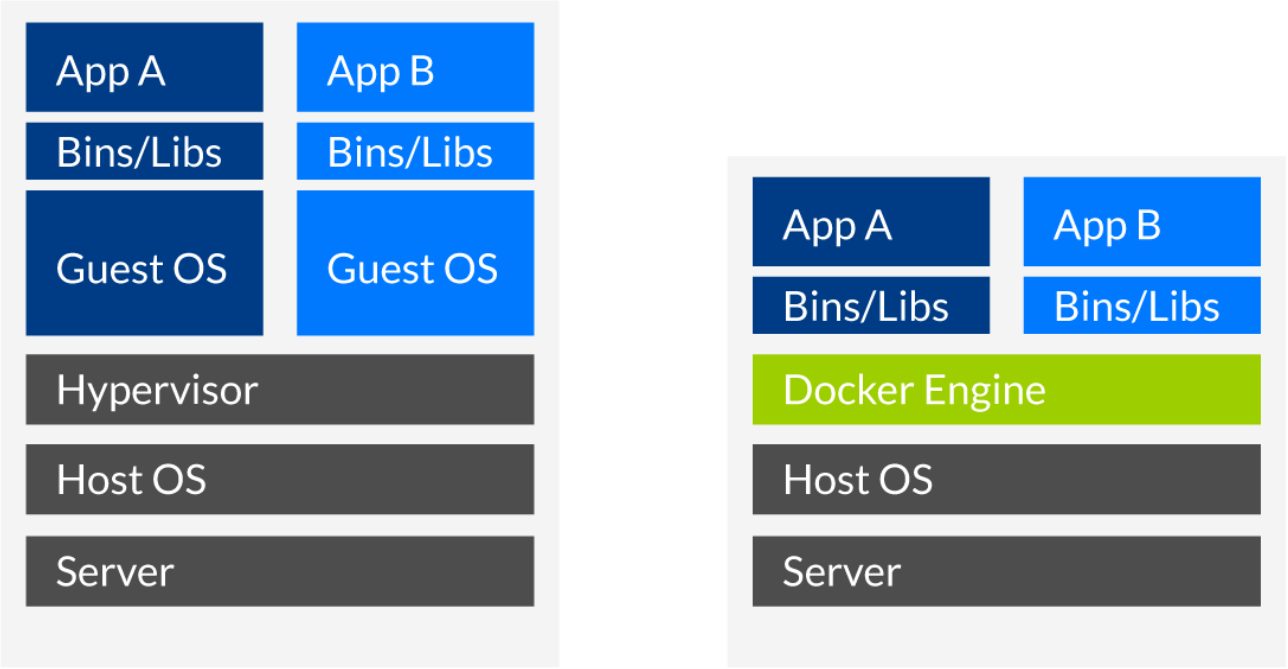
- Wraps an application's software into an invisible box with everything the application needs to run.
- Includes the OS, application code, runtime, system tools, system libraries, and etc.
- Docker containers are built off Docker images.
- Since images are read-only, Docker adds a read-write file system over the read-only file system of the image to create a container.

VM vs. Container

- Provides OS level virtualization by abstracting the “user space”.
- Looks like a VM w/ private IP, network, storage…
- Containers share the host system's kernel with other containers

EMORY UNIVERSITY SCHOOL OF MEDICINE

VMs vs Docker



EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Kubernetes

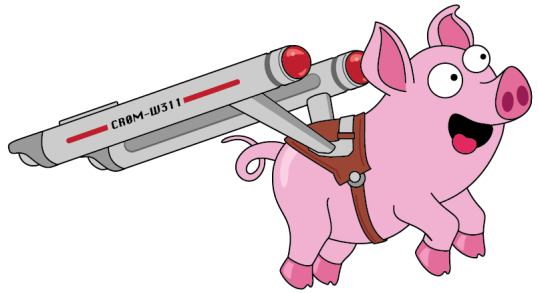
Orchestrating and Managing Workflows

https://cloud.google.com/kubernetes-engine/kubernetes-comic/

Meet WDL + Cromwell

- Workflow language that humans can read/write
  - Methods developers and biomedical scientists at large
  - https://software.broadinstitute.org/wdl/

- Execution engine that can
  - Run on any platform (on-prem *and* on Cloud)
  - Scale elastically based on workflow needs
  - https://github.com/broadinstitute/cromwell



Ref: Cromwell & WDL – Jeff Gentry, Broad Institute

Databases

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Flavors

- Relational
- NoSQL
  - Document
  - Key-Value
  - Columnar
  - Graph
- Managed Databases
  - Google BigQuery
  - AWS RedShift
  - ...

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

So what is NoSQL

== Not Only SQL

Simple Ops.

- Key/Value (KV) Lookups;
- Simple Rd/Wr;
- Aggregations
- Simple relationships

Sharding

- Pick a key
- Split data on that key
- Scale (grow) horizontally

Scaling:

- Horizontal → Distribute data + load over multiple servers
- Vertical → Server uses multiple CPUs

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

NoSQL Data Model

SQL → Rows and Columns
NoSQL??

- Agile (Schema less, or schema can change)
  - Schema == attributes
- Tuple
  - *Think rows in a RDBMS*
- Document
  - JSON
  - Nested values; extensible (agile)

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Key Value Stores

- Examples: Redis, Riak
- More like a filesystem than a database
  - Hard to capture relationships
- Higher performance
  - Simple lookup
  - No data model → No Hierarchical data
- Indexing:
  - Only primary index.
  - No secondary index

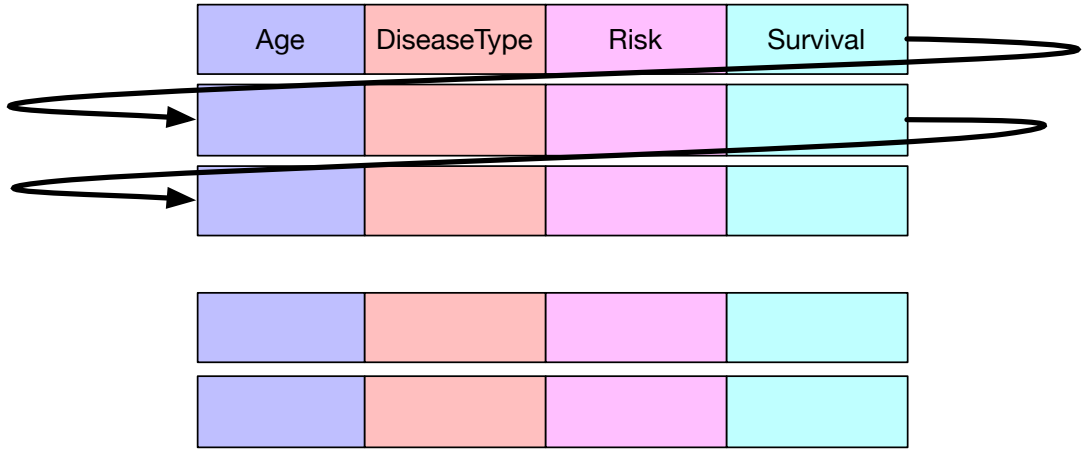
EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

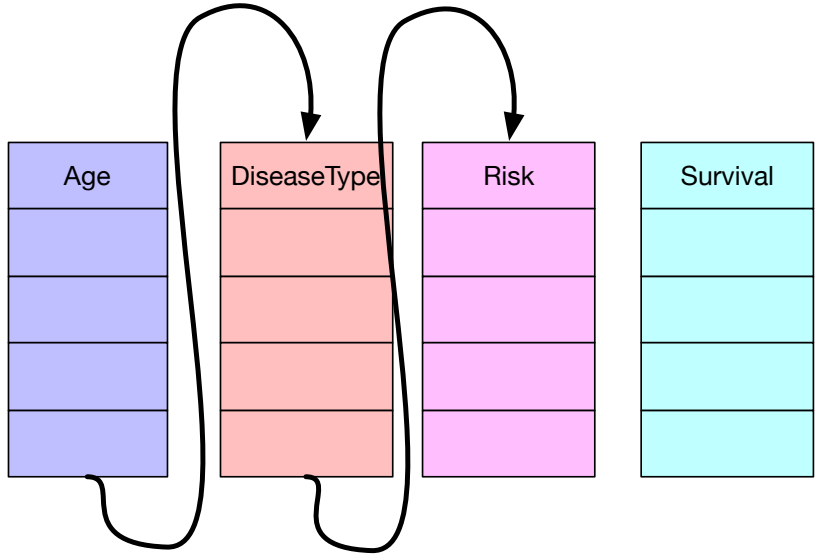
Document Store

- Document → Pointerless Object
- JSON
  - Maybe nested
  - May have a schema
- Supports Secondary indexes
- Eg: MongoDB; CouchDB
- Scalable
  - Sharding
  - Replication

Column Store

- Exploits the fact that queries only need to look at one or a few attributes and can safely ignore the rest of the data
- Excellent for aggregations, counts etc.
- Row-Based
  - Easy to add
  - Wasteful reads
- Column-Based
  - Writes are hard
  - Reads are very fast





Application 3

- Online Auction House
- Relationships between customer; purchase history; bids etc.
- Customer info not updated
  - Bid information frequently updated
- RDBMS

Application 1

- You have a web-application that displays a users info
- Data is added infrequently
- Read and Writes happen from the same interface

- Key-Value Store (Store customer name/id as Key. The rest == value)

Application 2

- DMV License Renewal
  - Lookup by multiple fields
- Frequent updates

- Document Store

Serverless Computing – FaaS

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

What is serverless?

Or Function-As-A-Service

- Server-side logic is written by you
- Is Stateless
  - Triggered by events
  - No persistent storage
- Thus can run on compute containers
- Triggered by events
- Fully managed by a third-party

FaaS (Serverless) Providers

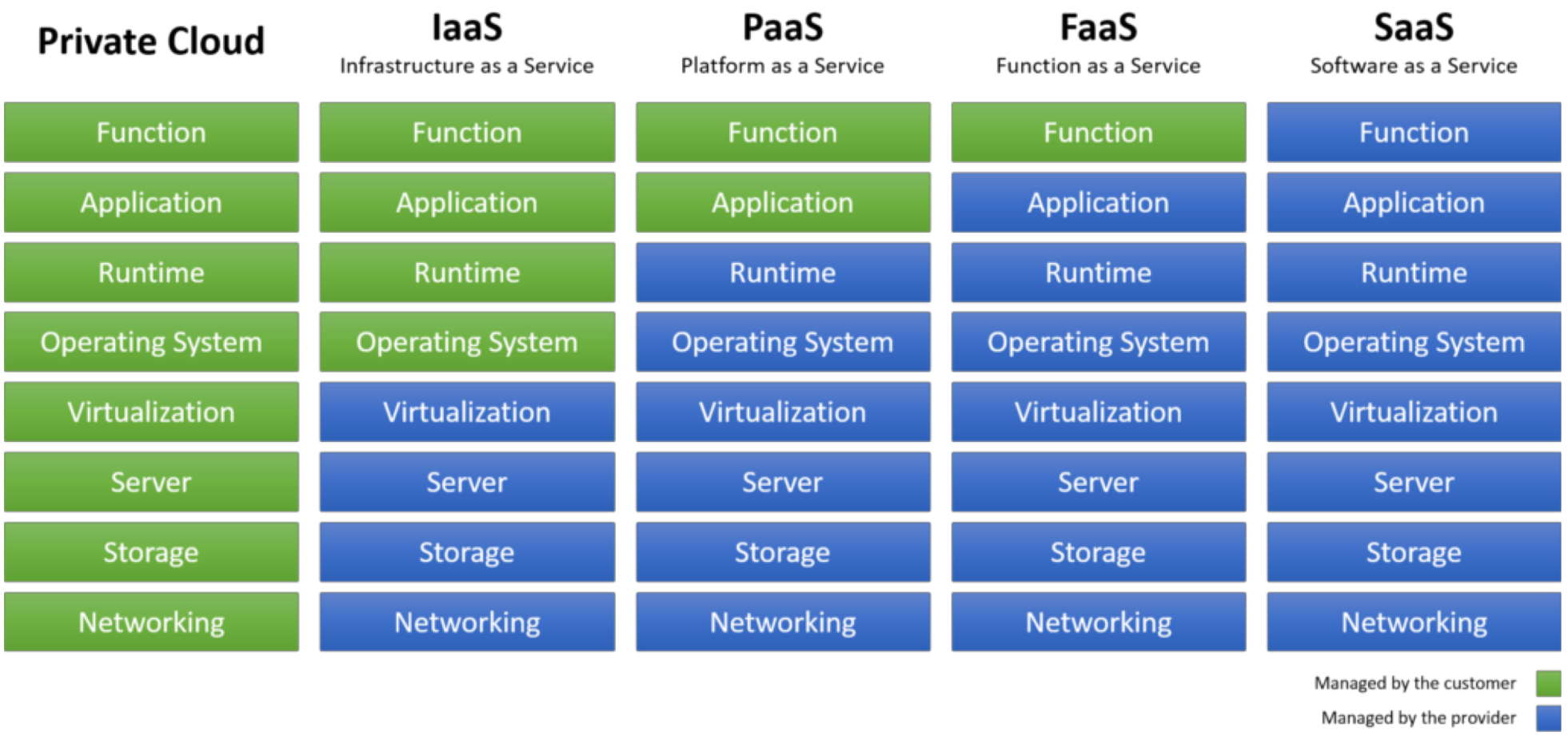
AWS Lambda

Google Cloud Functions

Azure Functions

IBM CloudFunctions Apache OpenWhisk

Serverless or Function-As-A-Service

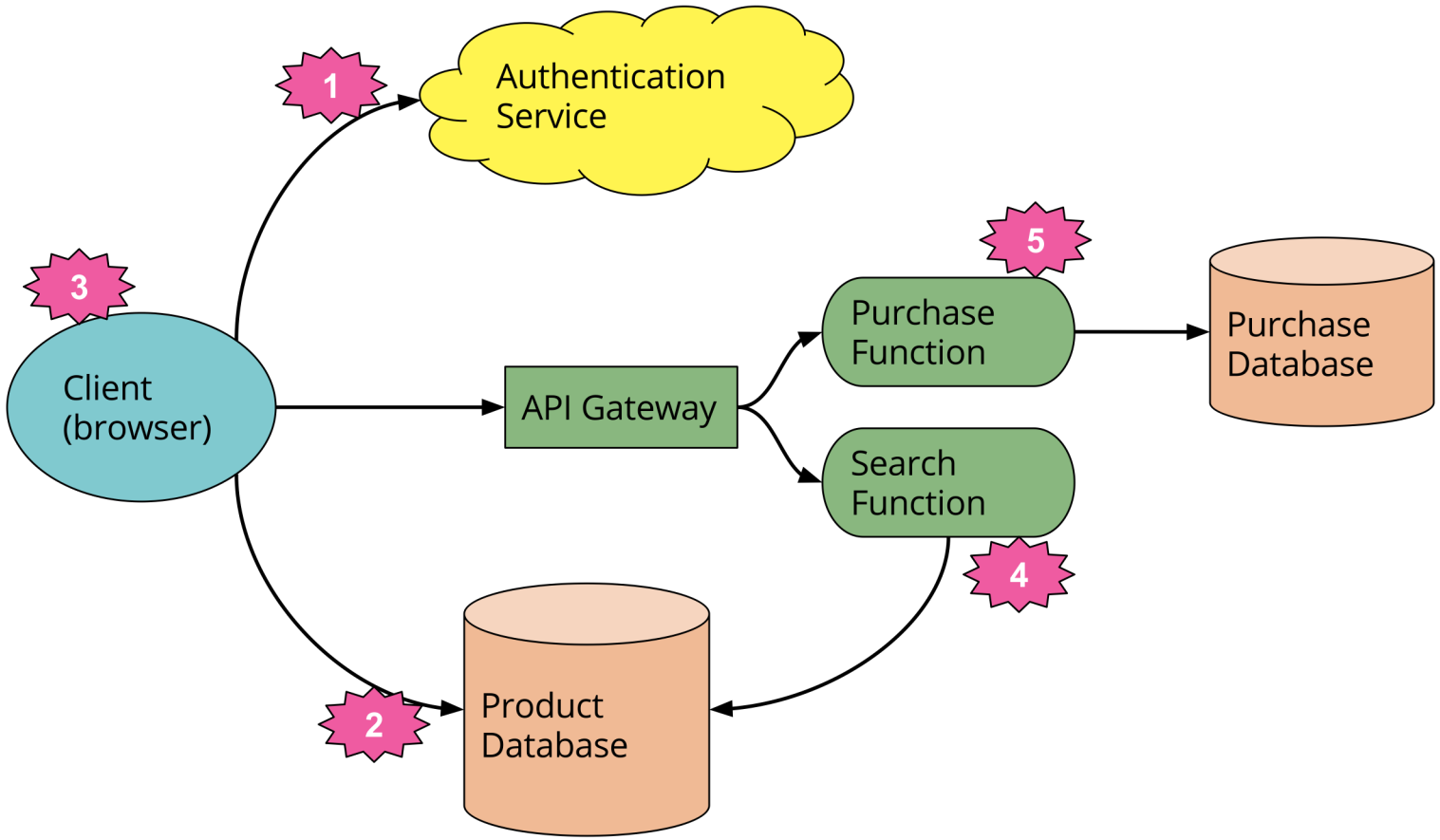


Source: Nuno Costa, "**From servers to functions, the serverless story**", Medium

Simple Example (Client-Server webapp)

Serverless Architectures by Mike Roberts. Martinfowler.com

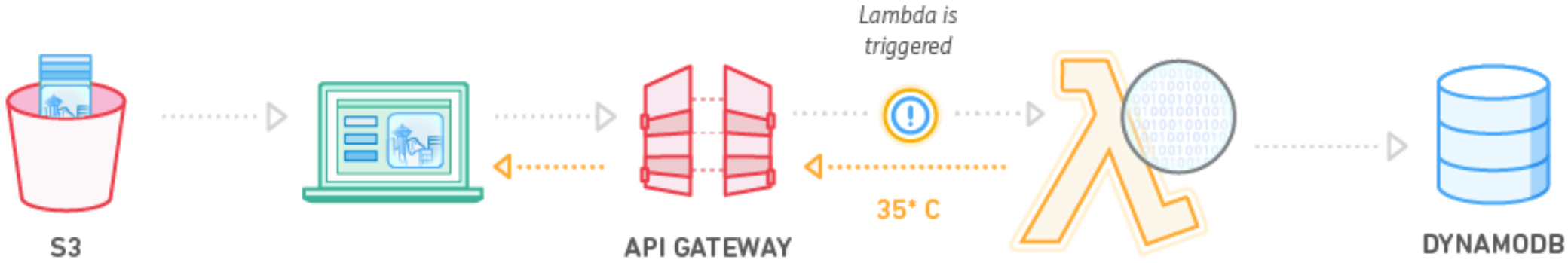
Serverless Redesign



Serverless Architectures by Mike Roberts. Martinfowler.com

EMORY UNIVERSITY SCHOOL OF MEDICINE

Example: *Weather Application*

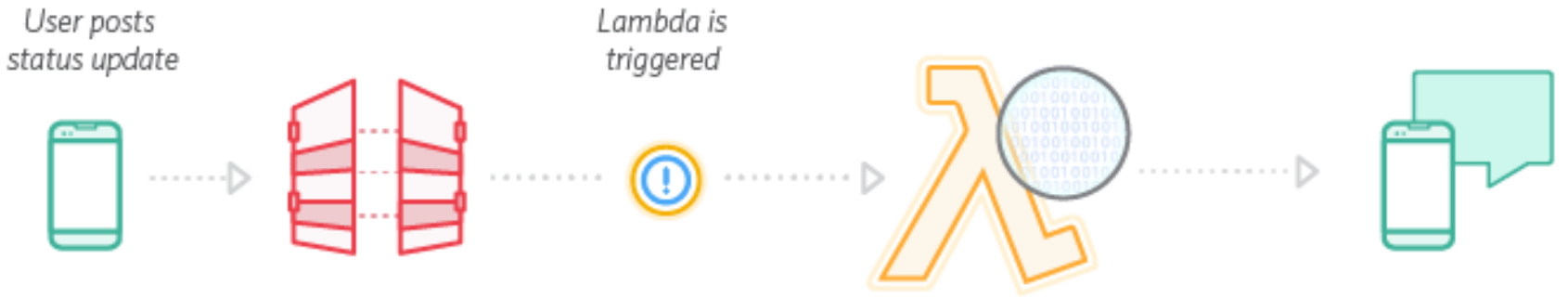


Ref: https://aws.amazon.com/serverless/

Simple Web-App → Client-Server (no Server)

EMORY UNIVERSITY SCHOOL OF MEDICINE

Example: *Mobile Backend for Social Media App*



Ref: https://aws.amazon.com/serverless/

Mobile App — Status Updates

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Example: *Image Thumbnail Creation*

Ref: https://aws.amazon.com/serverless/

On-Demand Service: Create Thumbnail / OCR on bank check…

Example: *Analysis of Streaming Social Media Data*

Ref: https://aws.amazon.com/serverless/

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

FaaS explored

https://martinfowler.com/articles/serverless.html

1. Run backend code without managing your own server systems or your own long-lived server applications.
2. FaaS replaces the click-processing server with something that doesn't need a provisioned server
3. We upload the code for our function to the FaaS provider, and the provider does everything else

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

FaaS explored

4. Horizontal scaling is completely automatic, elastic, and managed by the provider.
5. Triggered by event types defined by the provider
6. Most providers also allow functions to be triggered as a response to inbound HTTP requests

pywren

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

The most important primitive:

`map(function, data)`

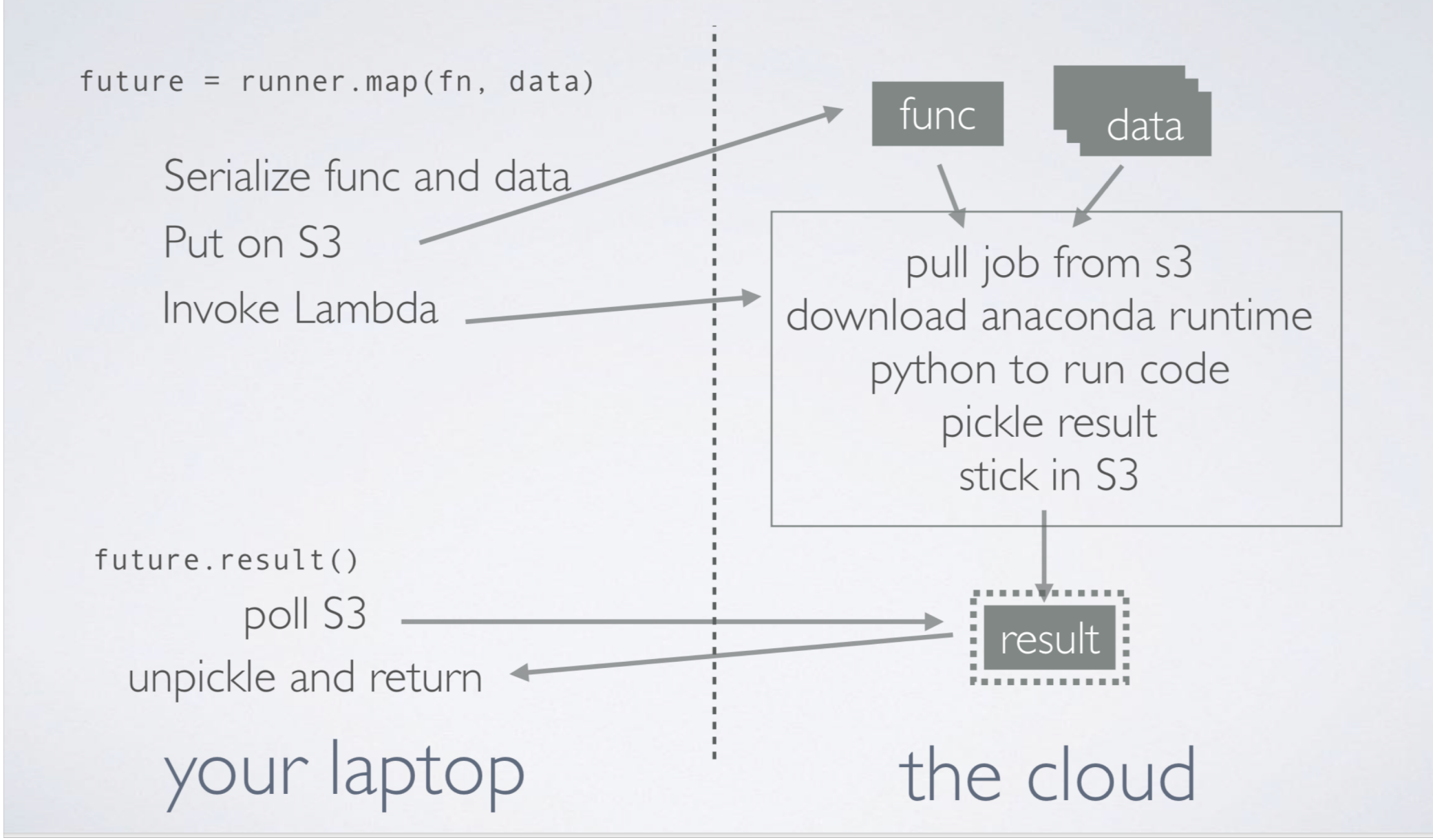
and… that's mostly it

```
import pywren
import numpy as np

def addone(x):
    return x + 1

wrenexec = pywren.default_executor()
xlist = np.arange(10)
futures = wrenexec.map(addone, xlist)

print [f.result() for f in futures]
```

The output is as expected:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Slides: From an introduction to pywren given by the pywren developer(s)



Slides: From an introduction to pywren given by the pywren developer(s)

Pywren – Lab/Exercise

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Data Transformation

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

DataFlow

Apache Beam
Apache NiFi

1. What is DataFlow
2. Writing data pipelines
3. Input/Output Run

HW/Exercise – Serverless Computing

- https://googlecoursera.qwiklabs.com/focuses/15711?locale=en

  - As a homework try to complete the Data Wrangling in Python using DataFlow and your own python code.

    OR

  - Try Data Wrangling using pywren

    OR

  - Pick out your own problem and run it on pywren

    OR

  - Complete the DataFlow exercises in the Coursera course.