@unterstein @dcos @bedcon #bedcon

Operating microservices with Apache Mesos and DC/OS

DC/OS

Johannes Unterstein

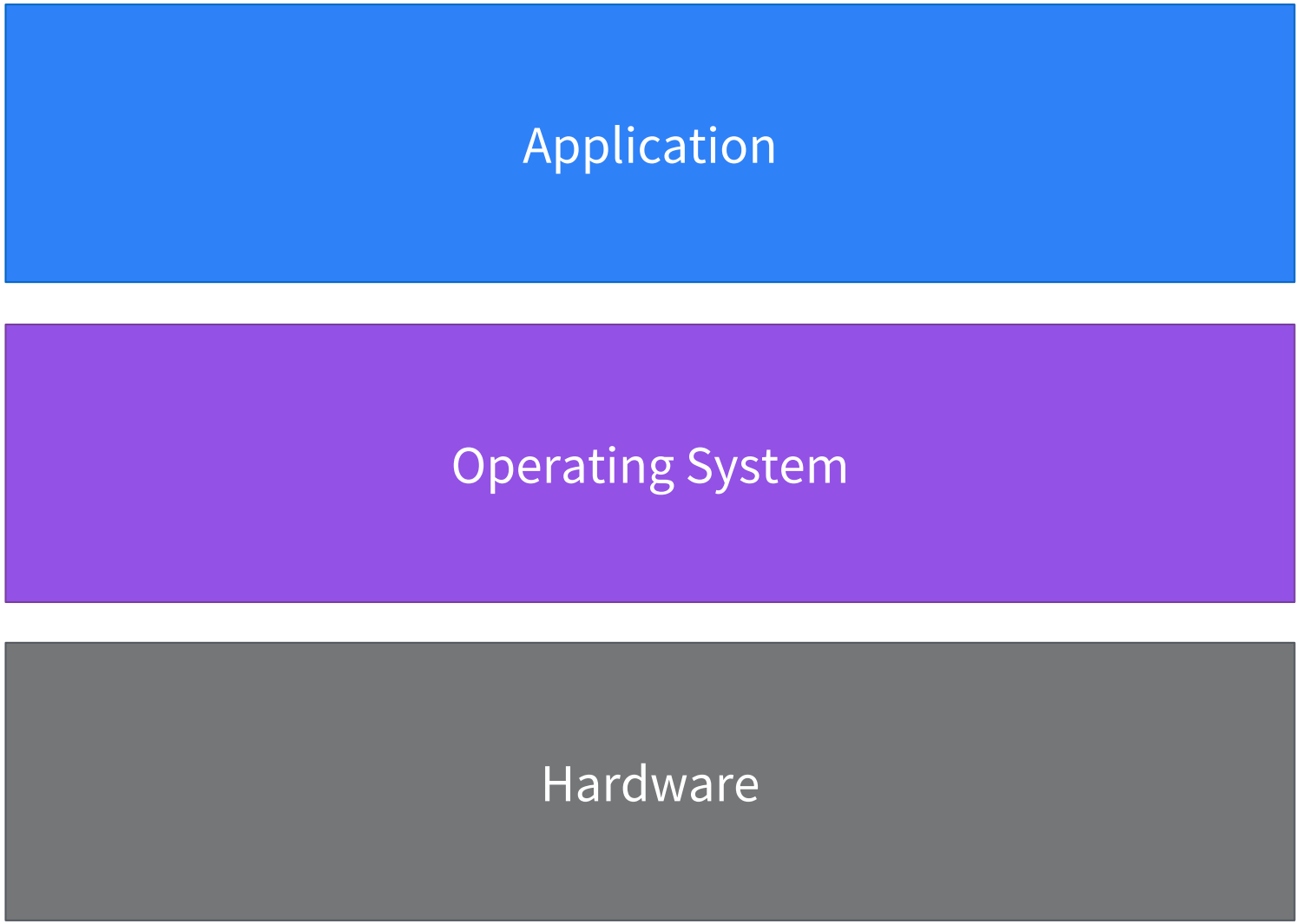
Software Engineer @Mesosphere

@unterstein

@unterstein.mesosphere

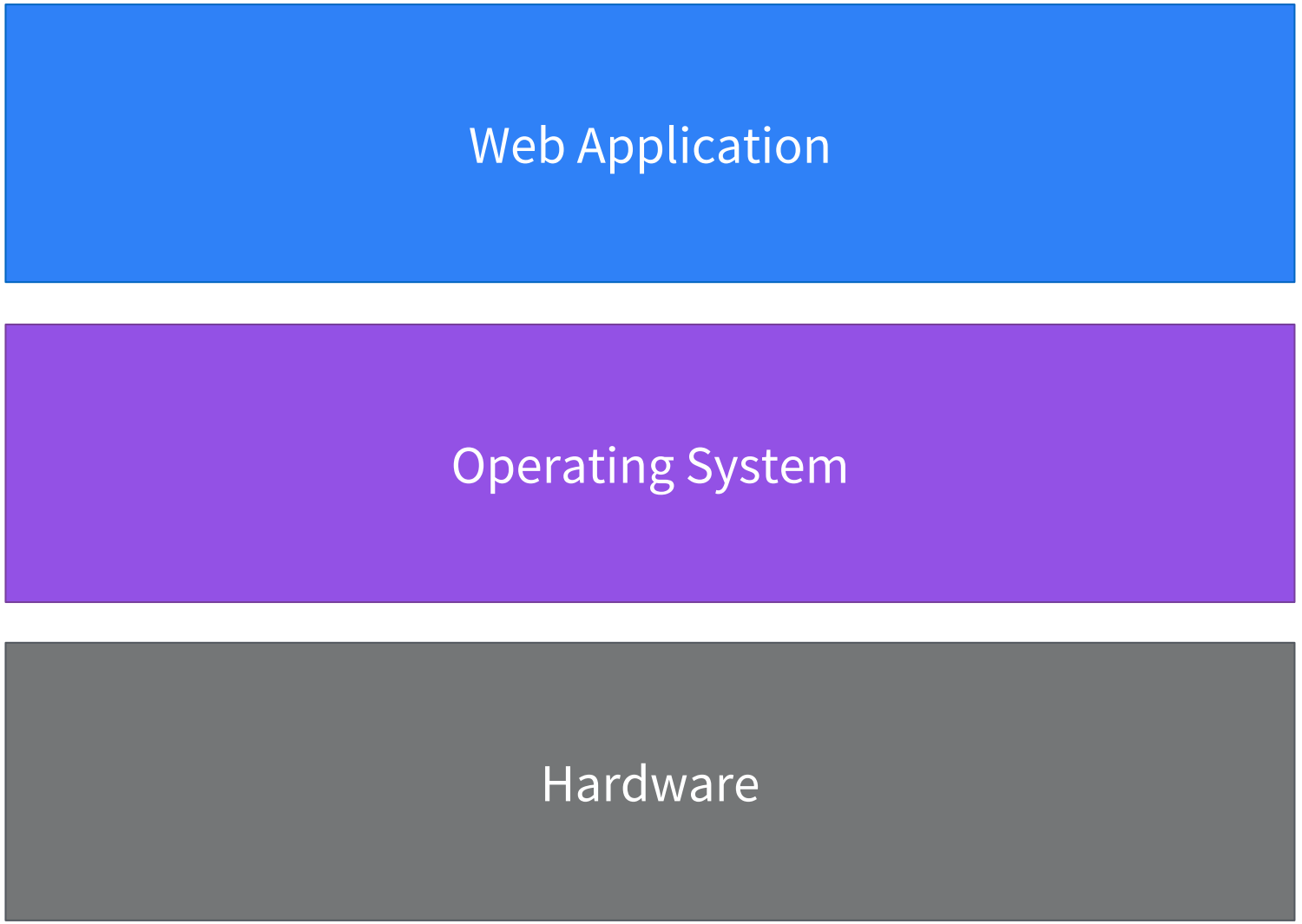

In the beginning
there was a big
Monolith

COMPUTERS



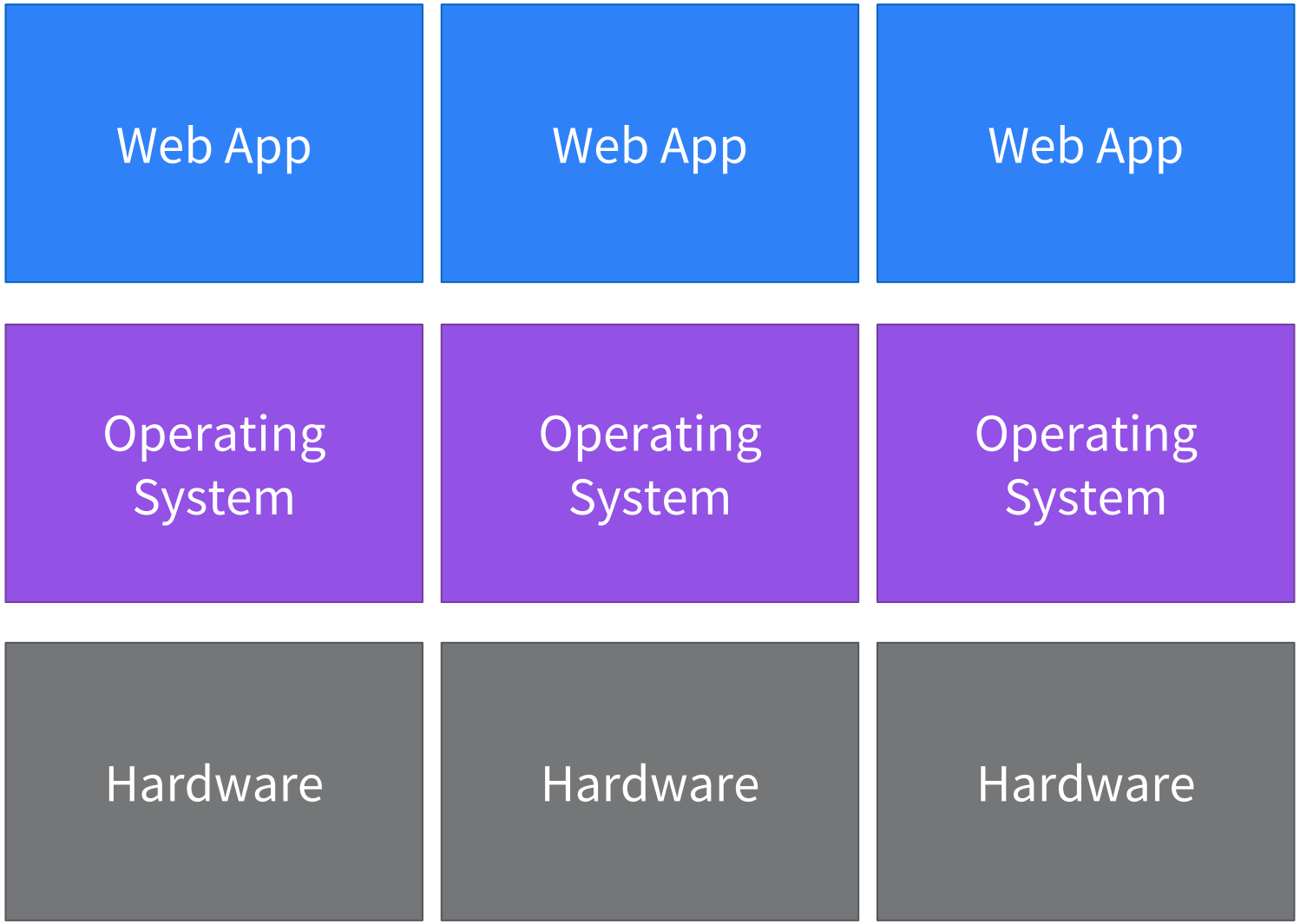
INTERNET

- Remote Users!



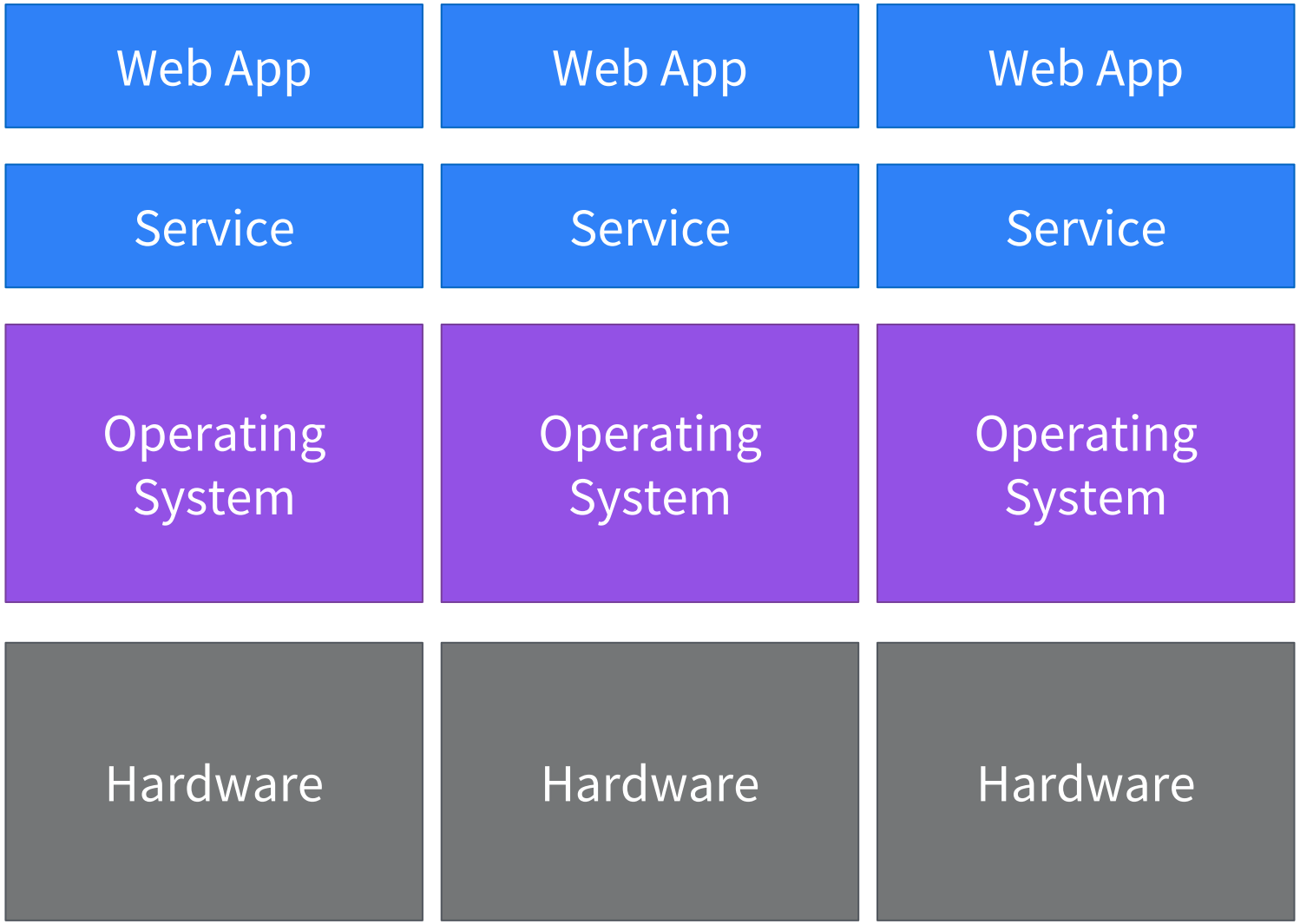
DISTRIBUTION

- Horizontal Scale
- Fault Tolerance
- Availability
- Load Balancing



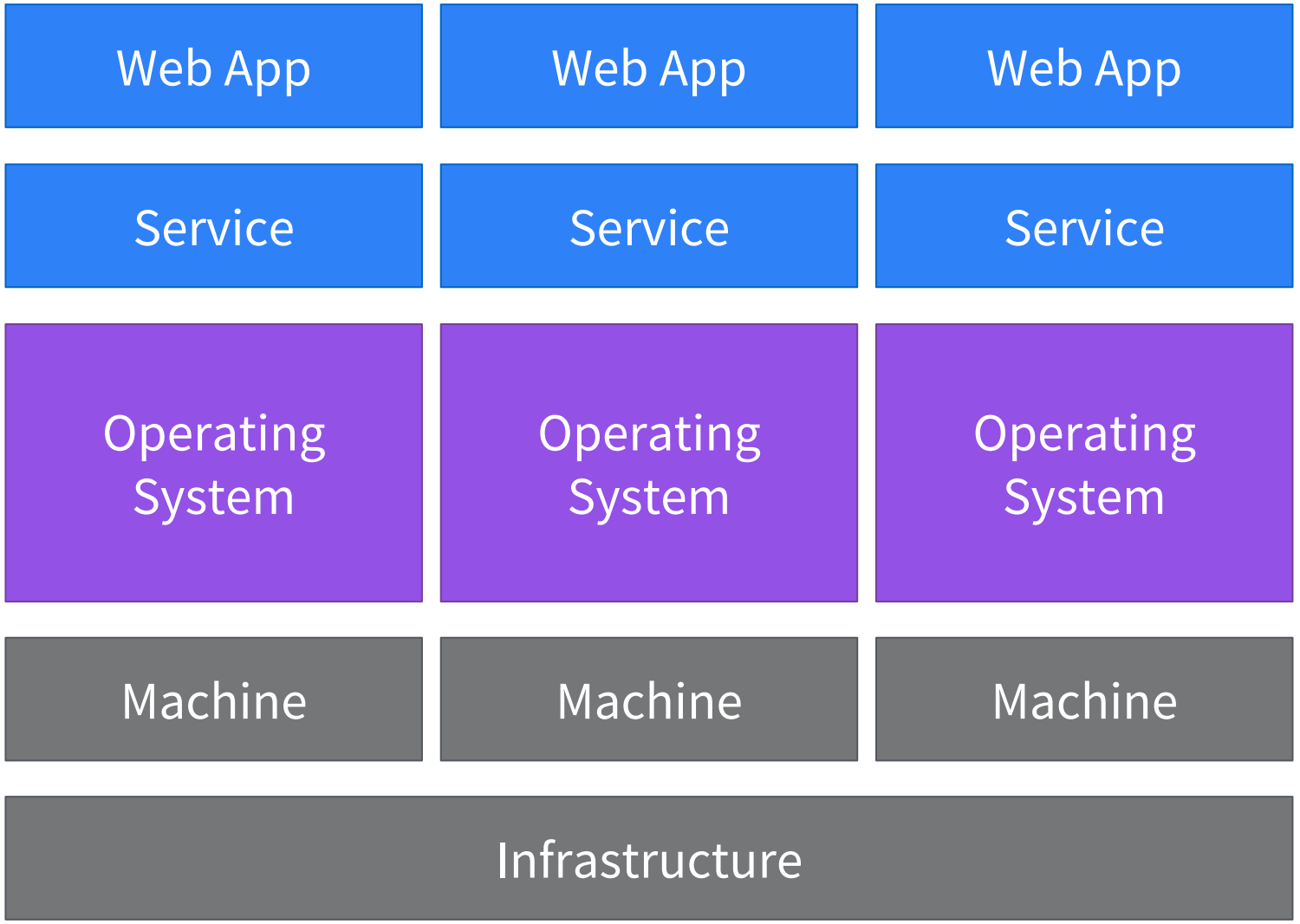
SERVICE-ORIENTED ARCHITECTURE

- Separation of concerns
- Optimization of bottlenecks
- Smaller teams
- API Contracts
- Data replication
- Complicated provisioning
- Dependency management



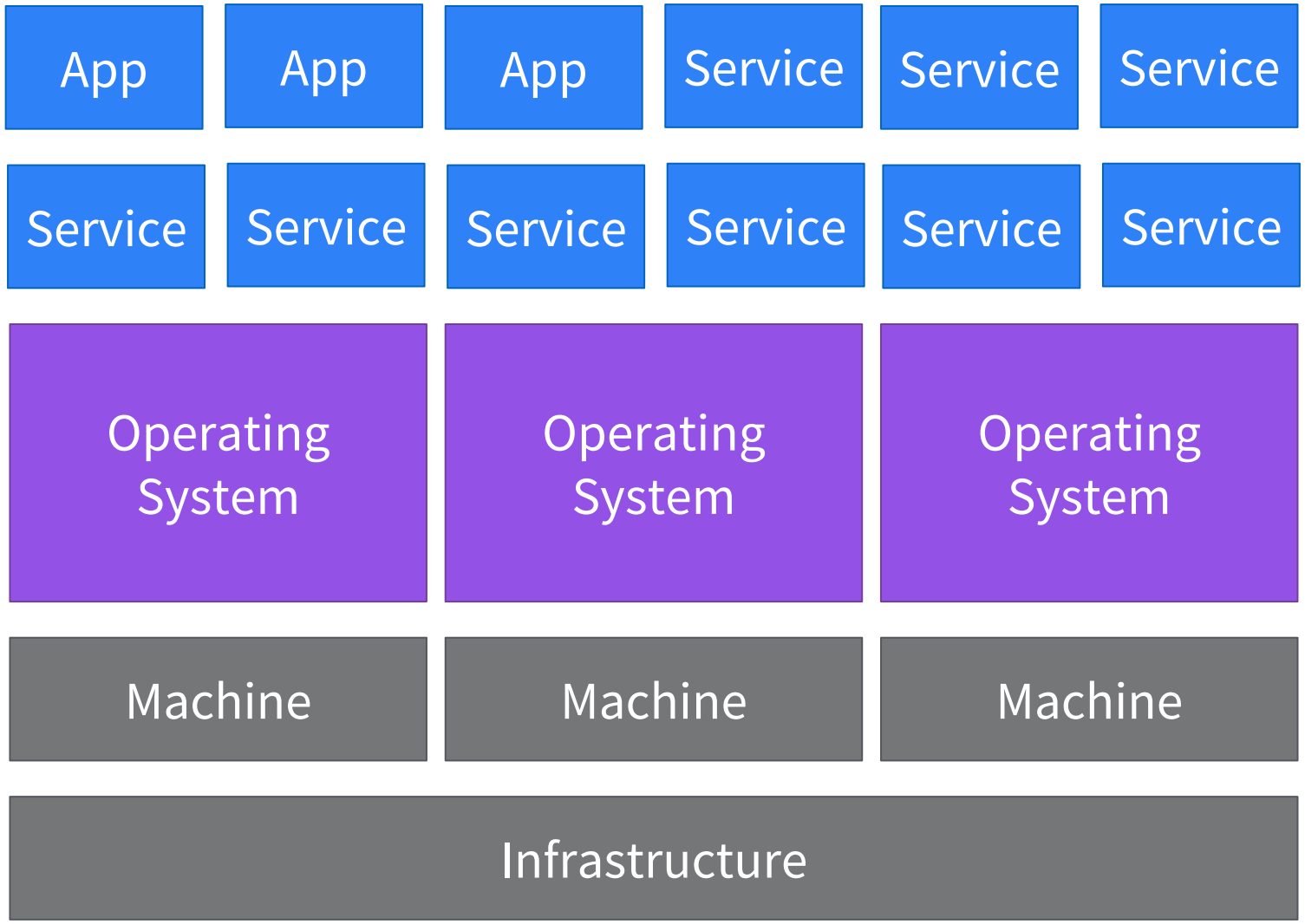
HARDWARE VIRTUALIZATION

- Fast provisioning
- Isolation
- Portability
- Utilization
- Configuration Management
- Virtual Networking
- Credential management



MICROSERVICES

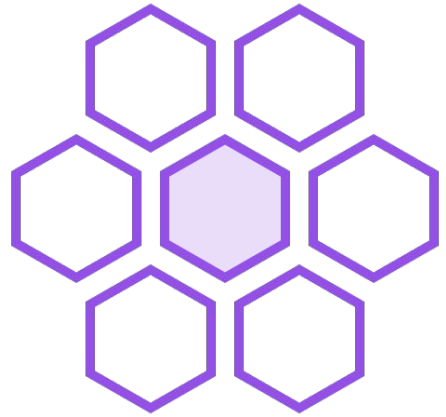
- Polyglot
- Single Responsibility
- Smaller Teams
- Utilization
- Machine types/groups
- Dependency hell



OVERVIEW

noun | *ˈmīkrō/ /ˈsərvəs/* :

an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other modules.*

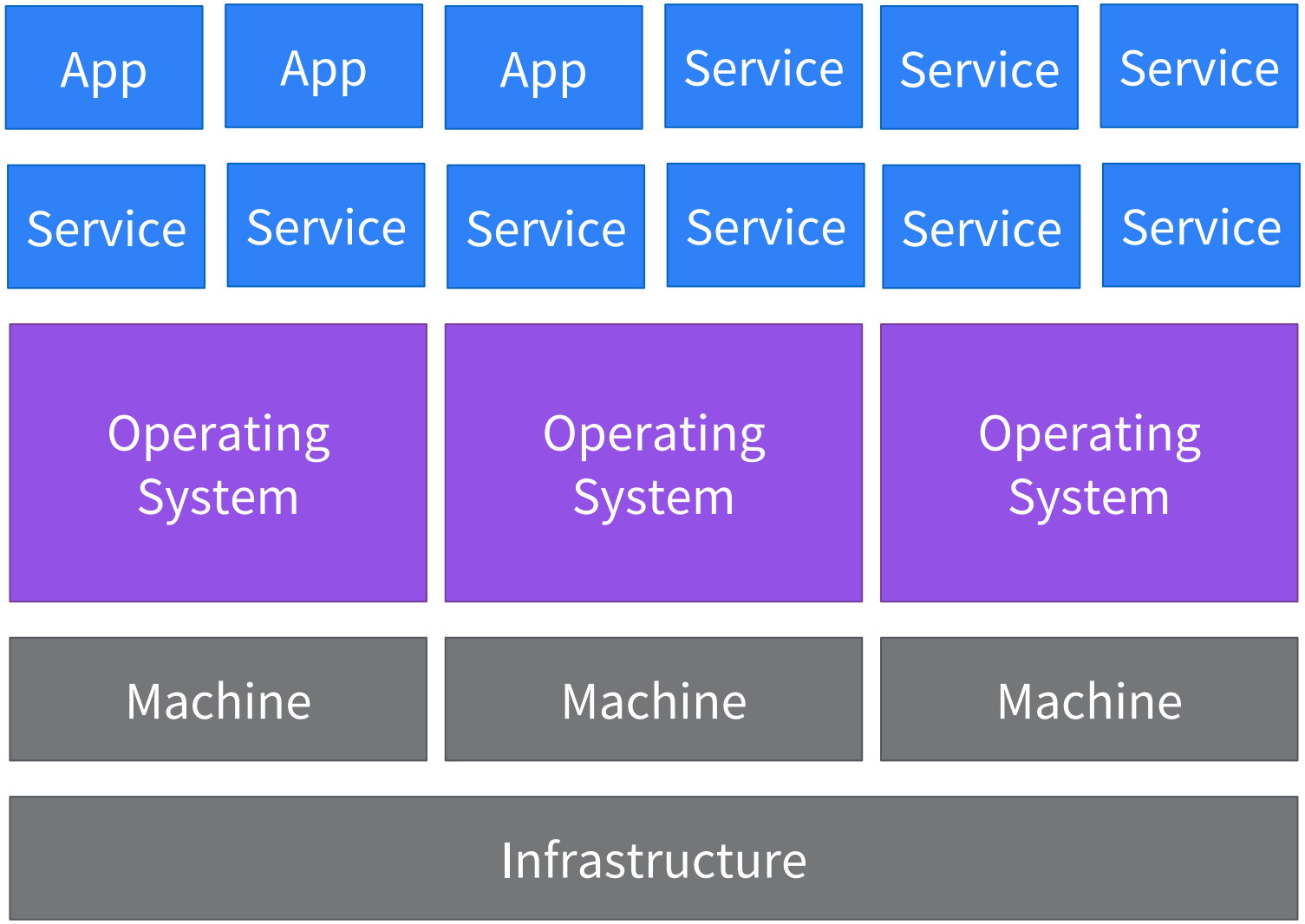


Microservices are designed to be **flexible**, **resilient**, **efficient, robust**, and **individually scalable**.

*From whatis.com

MICROSERVICES

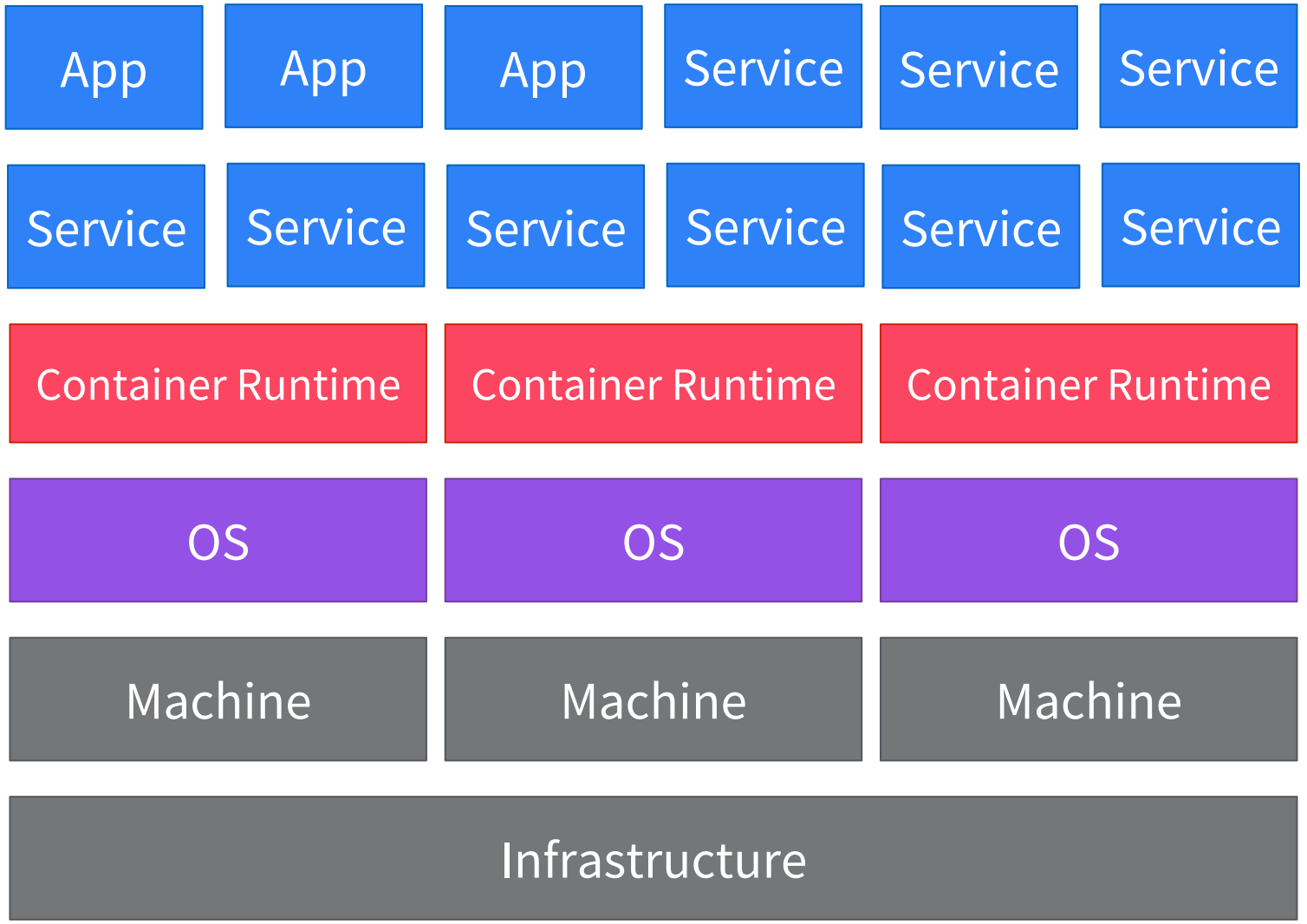
- Polyglot
- Single Responsibility
- Smaller Teams
- Utilization
- Machine types/groups
- Dependency hell



Run everything in containers!

CONTAINERS

- Rapid deployment
- Dependency vendoring
- Container image repositories
- Spreadsheet scheduling



CONTAINER SCHEDULING

RESOURCE MANAGEMENT

SERVICE MANAGEMENT

CONTAINER ORCHESTRATION

CONTAINER SCHEDULING

RESOURCE MANAGEMENT

SERVICE MANAGEMENT

CONTAINER ORCHESTRATION

CONTAINER SCHEDULING

- Placement
- Replication/Scaling
- Resurrection
- Rescheduling
- Rolling Deployment
- Upgrades
- Downgrades
- Collocation

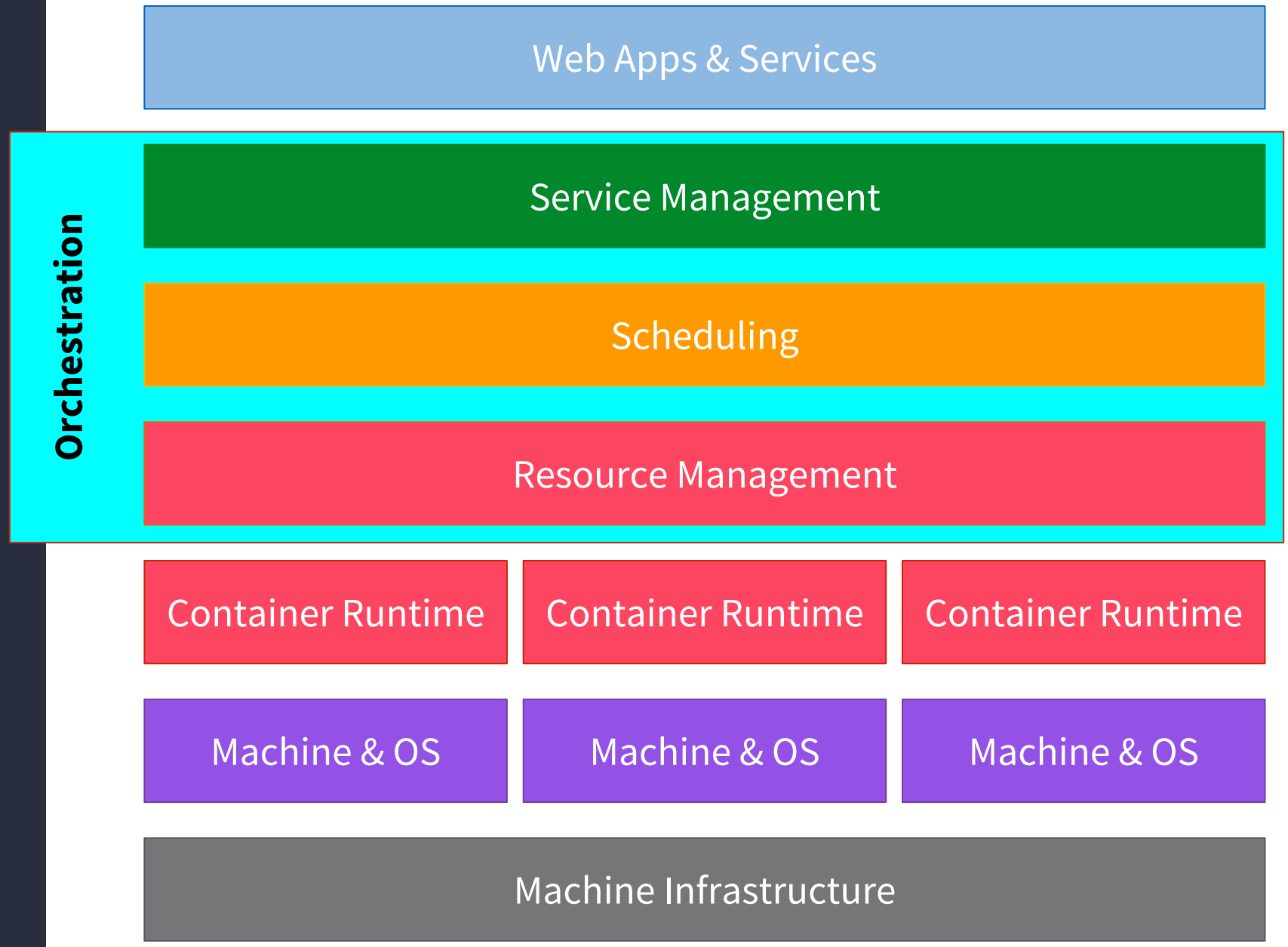
RESOURCE MANAGEMENT

- Memory
- CPU
- GPU
- Volumes
- Ports
- IPs
- Images/Artifacts

SERVICE MANAGEMENT

- Labels
- Groups/Namespaces
- Dependencies
- Load Balancing
- Readiness Checking

CONTAINER ORCHESTRATION




Meanwhile...
MapReduce is crunching Data

DATA PROCESSING AT HYPERSCALE

EVENTS

Ubiquitous data streams from connected devices

INGEST

Ingest millions of events per second

STORE

Distributed & highly scalable database

ANALYZE

Real-time and batch process data

ACT

Visualize data and build data driven applications

Sensors

Devices

Clients

Apache Kafka

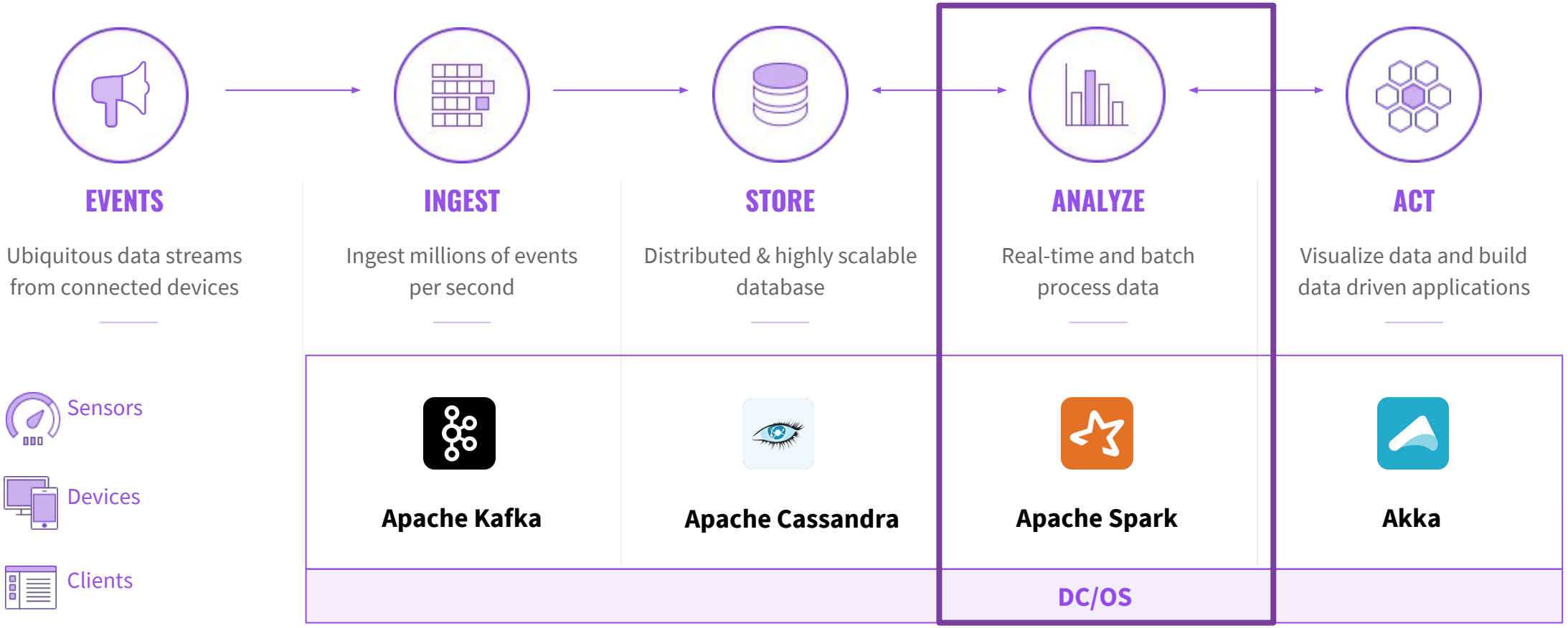
Apache Cassandra

Apache Spark

Akka

DC/OS

DATA PROCESSING AT HYPERSCALE



STREAM PROCESSING

- Apache Storm
- Apache Spark
- Apache Samza
- Apache Flink
- Apache Apex
- Concord
- cloud-only: AWS Kinesis, Google Cloud Dataflow

EXECUTION MODEL

Micro-Batching

Spark

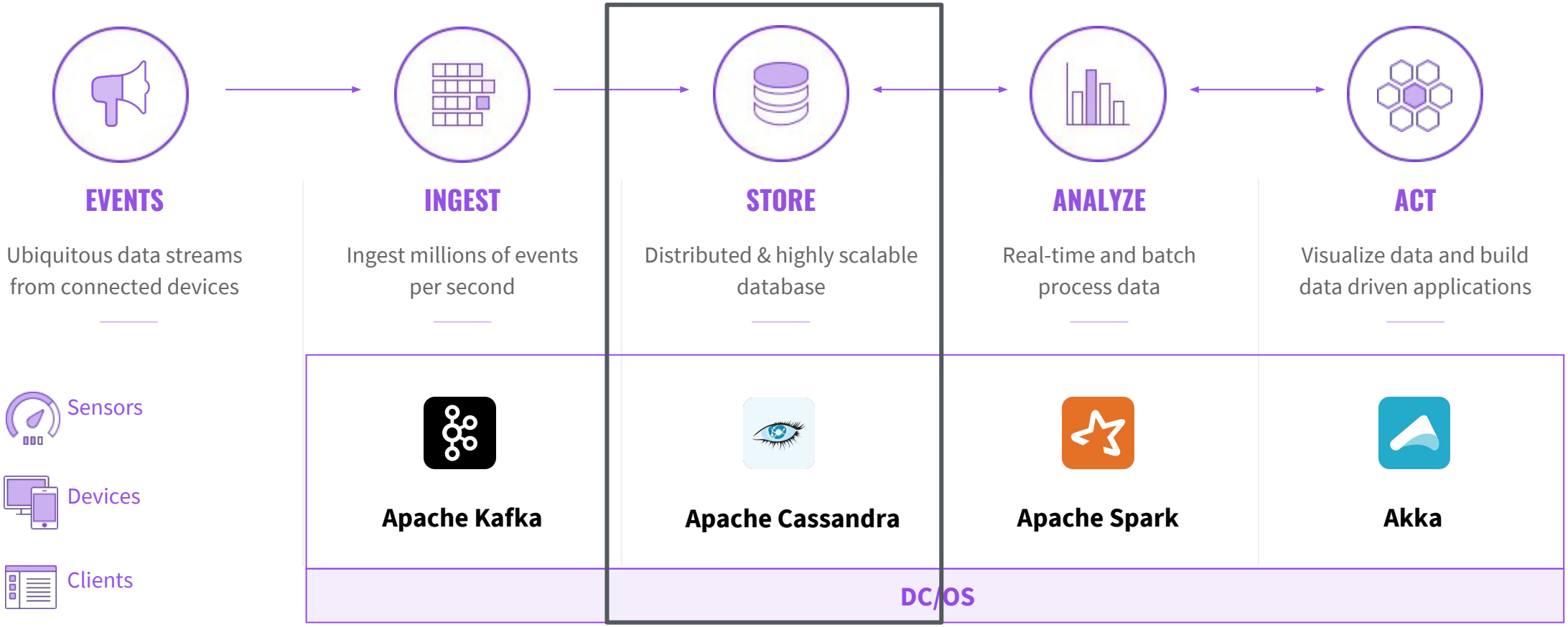
Native Streaming

APEX

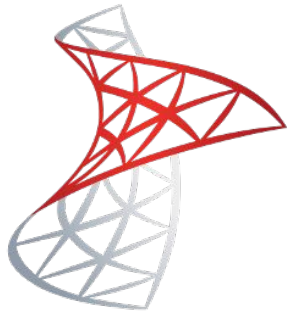
Flink

STORM

DATA PROCESSING AT HYPERSCALE



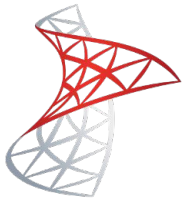
Datastores



Data Model

Relational

- Schema
- SQL
- Foreign Keys/Joins
- OLTP/OLAP



Key-Value

- Simple
- Scalable
- Cache

Graph

- Complex relations
- Social Graph
- Recommendation
- Fraud detections

Document

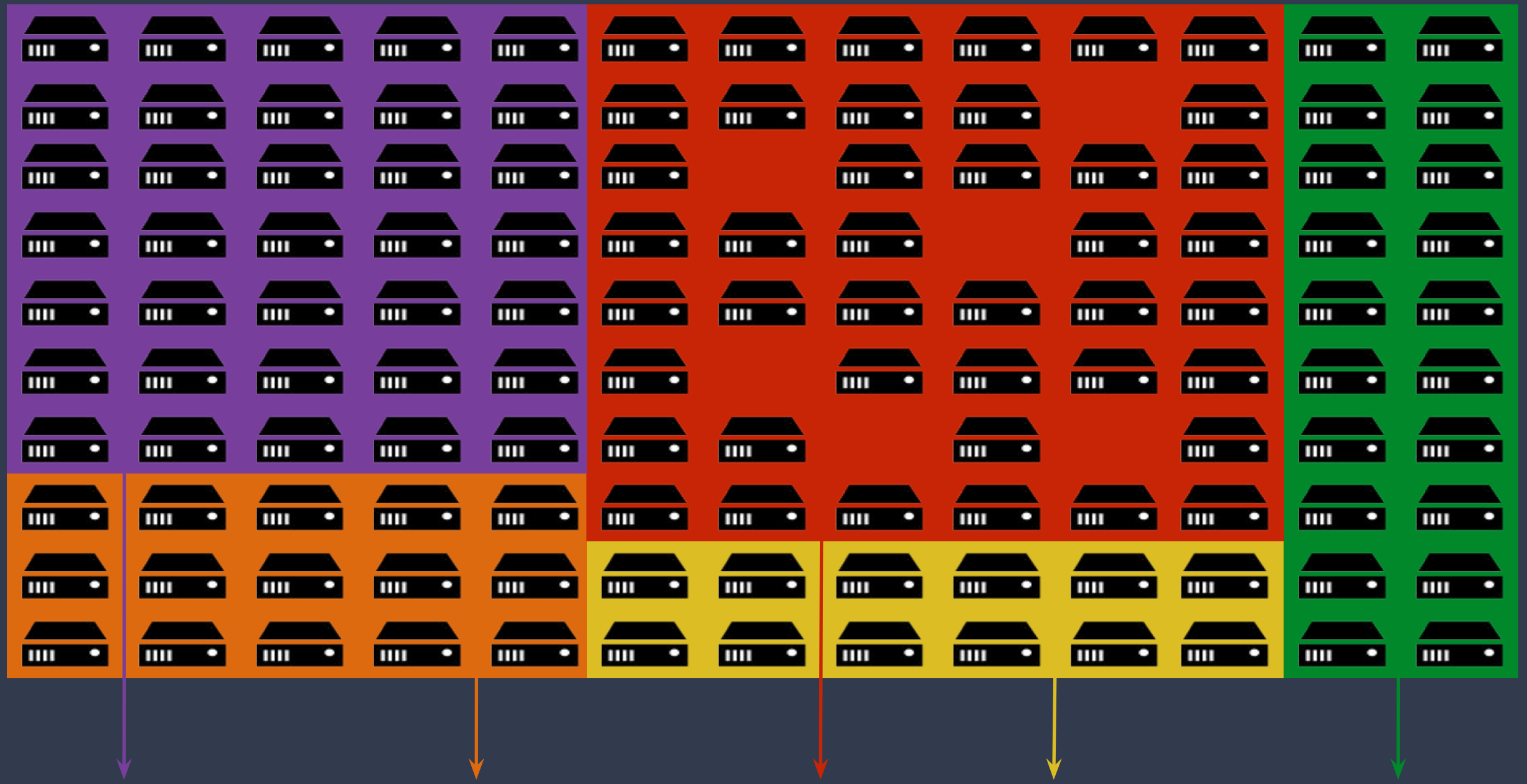
- Schema-Less
- Semi-structured queries
- Product catalogue
- Session data

Time-Series

Files


Modern datacenter
Google | google.com/datacenters

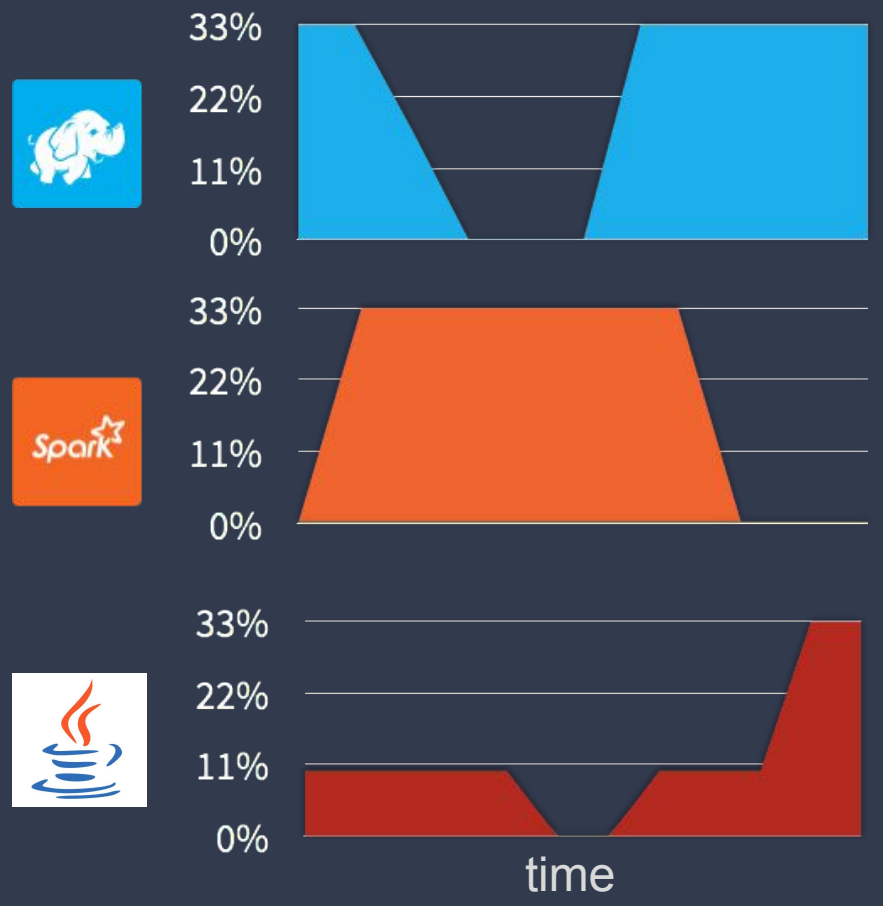

Flink
Cassandra
µServices
Spark
RDB


Flink
Cassandra
µServices
Spark
RDB

KEEP IT STATIC

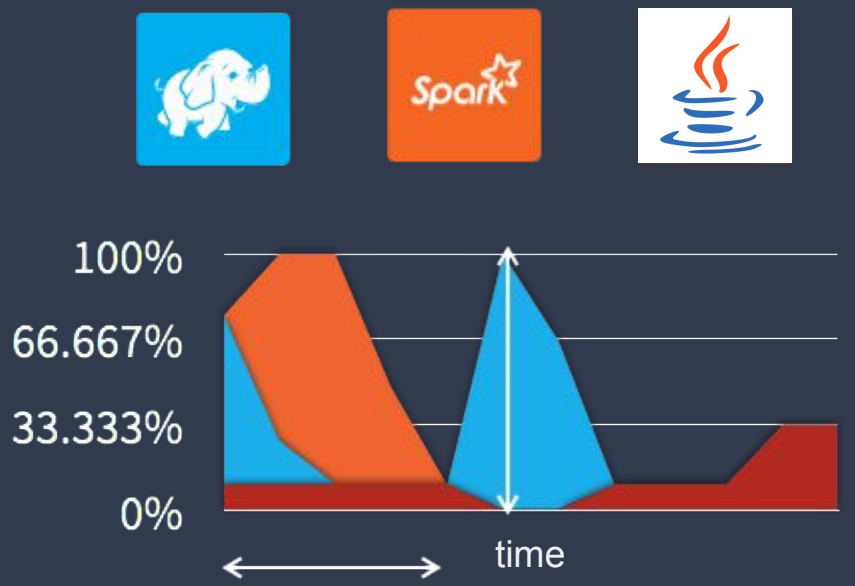
A naive approach to handling varied app requirements: **static partitioning**.

Maintaining sufficient headroom to handle peak workloads on all partitions leads to **poor utilization** overall.

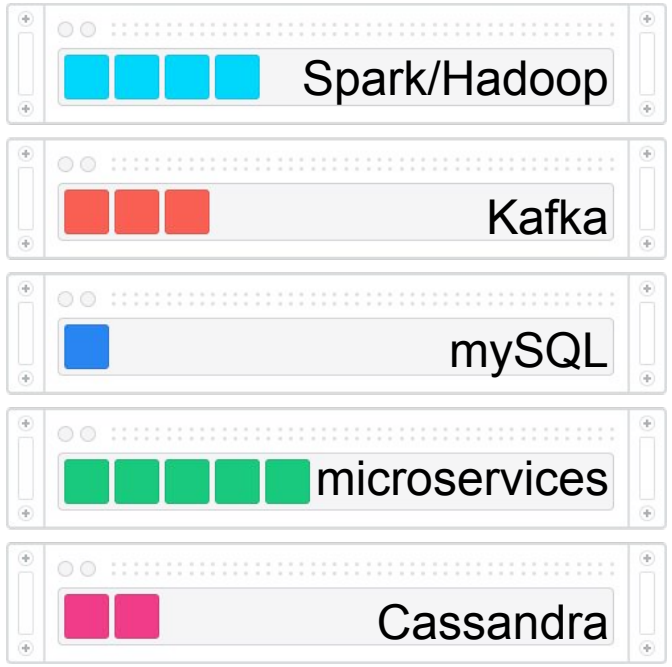


SHARED RESOURCES

Multiple frameworks can use the same cluster resources, with their share adjusting dynamically.

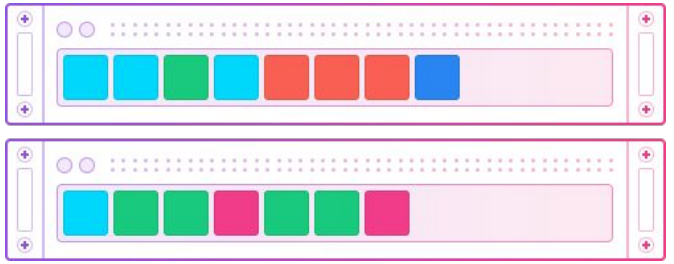


SILOS OF DATA, SERVICES, USERS, ENVIRONMENTS



Industry Average
12-15% utilization

Typical Datacenter
siloed, over-provisioned servers, low utilization



Multiplexing
30-40% utilization, more at some users

4X

Modern Datacenter
automated schedulers, workload multiplexing onto the same machines

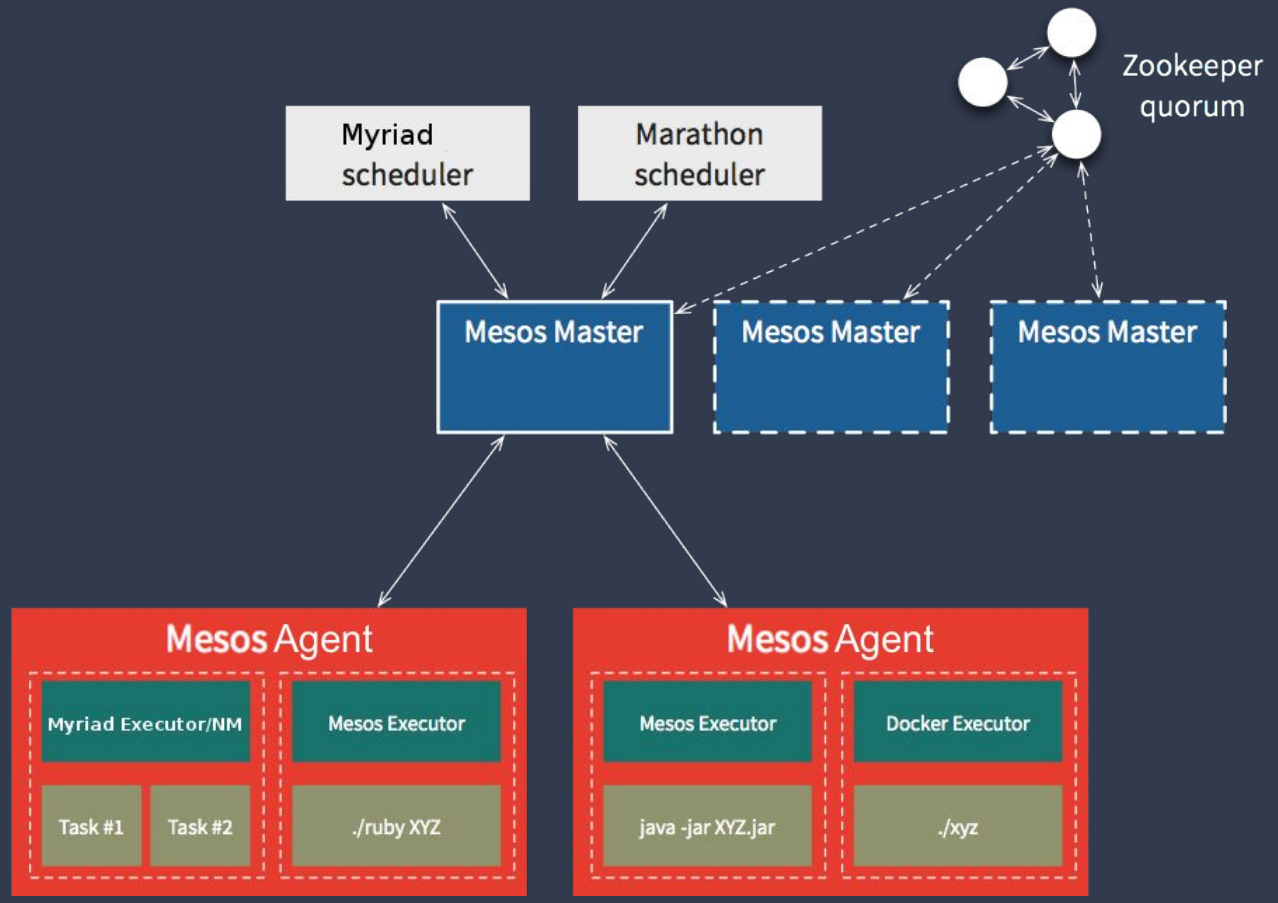
THE BIRTH OF MESOS

Apache Mesos

- A top-level Apache project
- A cluster resource negotiator
- Scalable to 10,000s of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps
- Native Docker support

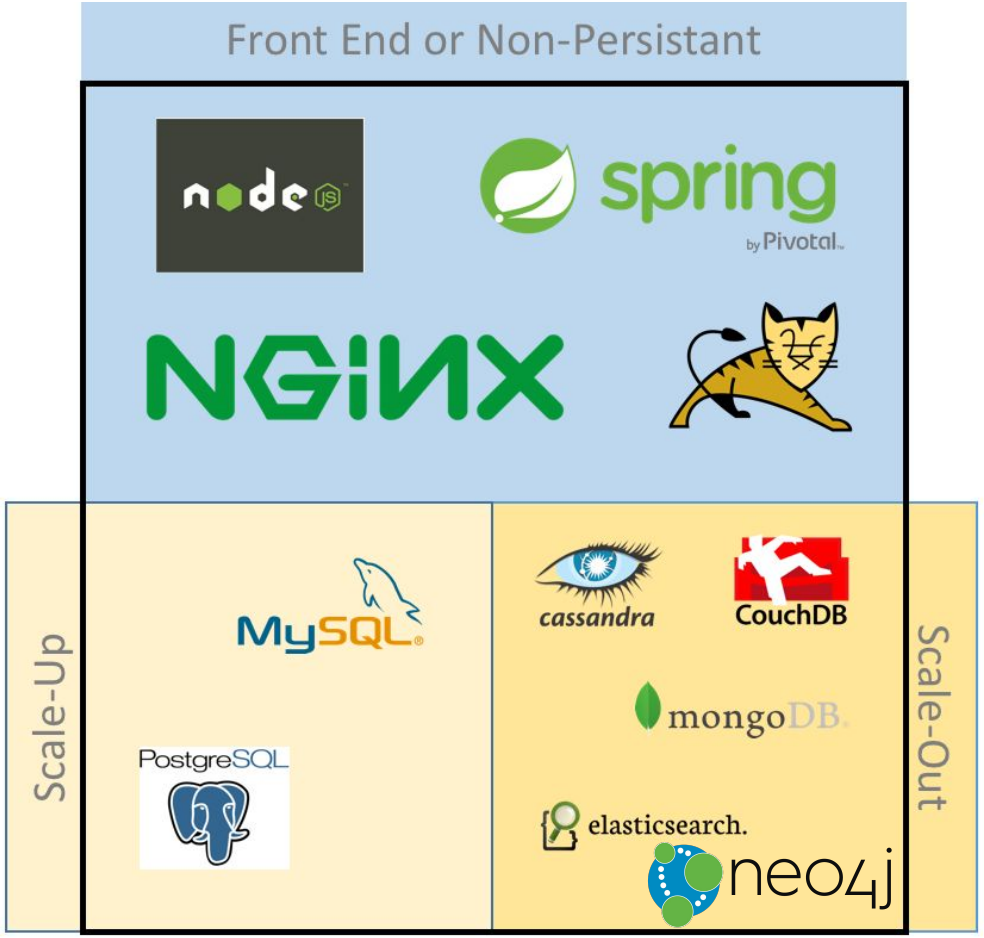
MESOS FUNDAMENTALS

ARCHITECTURE



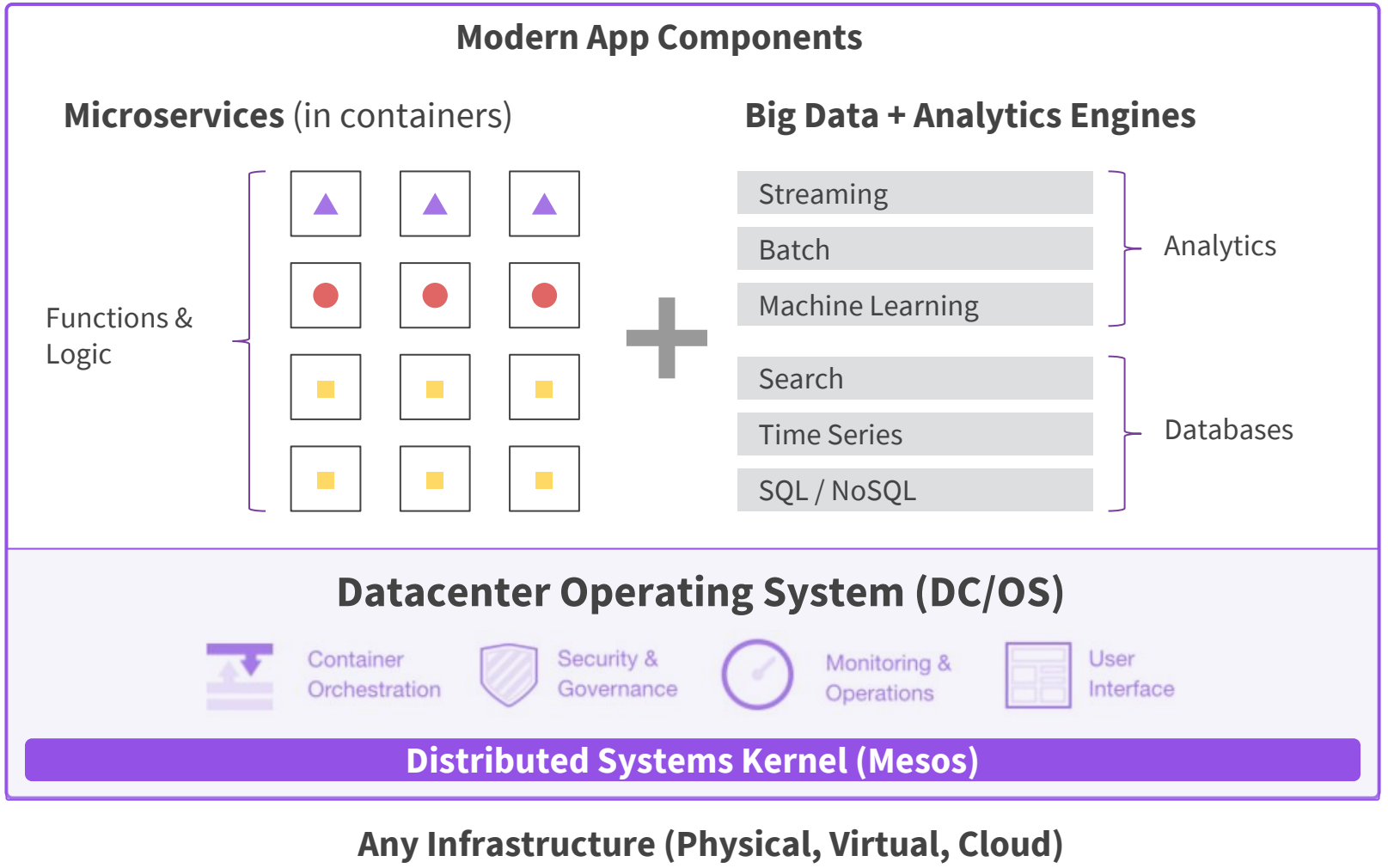
STORAGE OPTIONS

- Default Sandbox
  - ⟹ Simple to use, Task failures
- Persistent Volumes
  - ⟹ Task failures, (permanent) Node failures
- Distributed File System/External Storage
  - ⟹ Node failures, non-local writes



DC/OS

DC/OS ENABLES MODERN DISTRIBUTED APPS



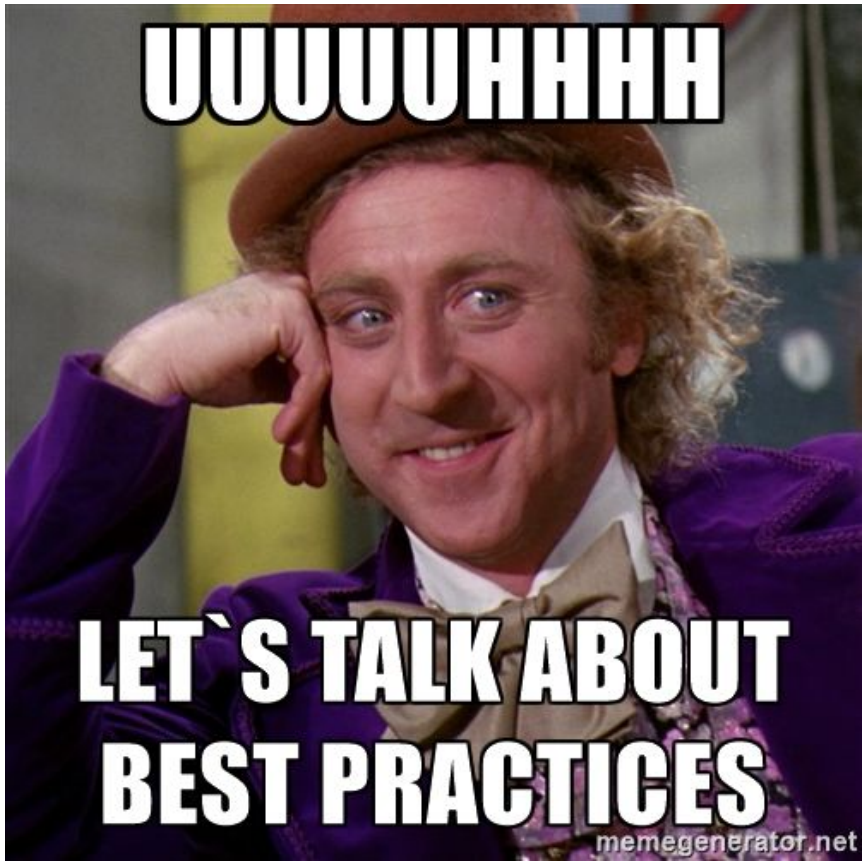
THE BASICS

DC/OS is ...

- 100% open source (ASL2.0)
  - A big, diverse community
- An umbrella for ~30 OSS projects
  - Roadmap and designs
  - Docs, tutorials, setup installations..
  - Check https://dcos.io
- Familiar, with more features
  - Networking, Security, CLI, UI, Service Discovery, Load Balancing, Packages, ...

Best Practices

GOING TO PRODUCTION



- Deployment
- Service Discovery
- Monitoring
- Logging

BEST PRACTICES

DEPLOYMENT

- Version configurations
- Private registries
- Resource limits
- Make it HA

BEST PRACTICES

SERVICE DISCOVERY

- Dynamic ports
- Virtual IPs
- DNS
- Overlay networks
- External tools

BEST PRACTICES

MONITORING & LOGGING

- Application metrics
- Health checks
- Alerting
- Aggregate logs
- Consistent service logs

Questions?

Code: https://git.io/v1DjV
https://git.io/vXUoy

@unterstein
github.com/unterstein

@mesosphere / mesosphere.com
@dcos / dcos.io / chat.dcos.io

DC/OS