





- » Skan.ai - chief Architect
- » Ai.robots - chief Architect
- » Genpact - solution Architect
- » Welldoc - chief Architect
- » Microsoft
- » Mercedes
- » Siemens
- » Honeywell



Mubarak



what do
YOU?
expect.

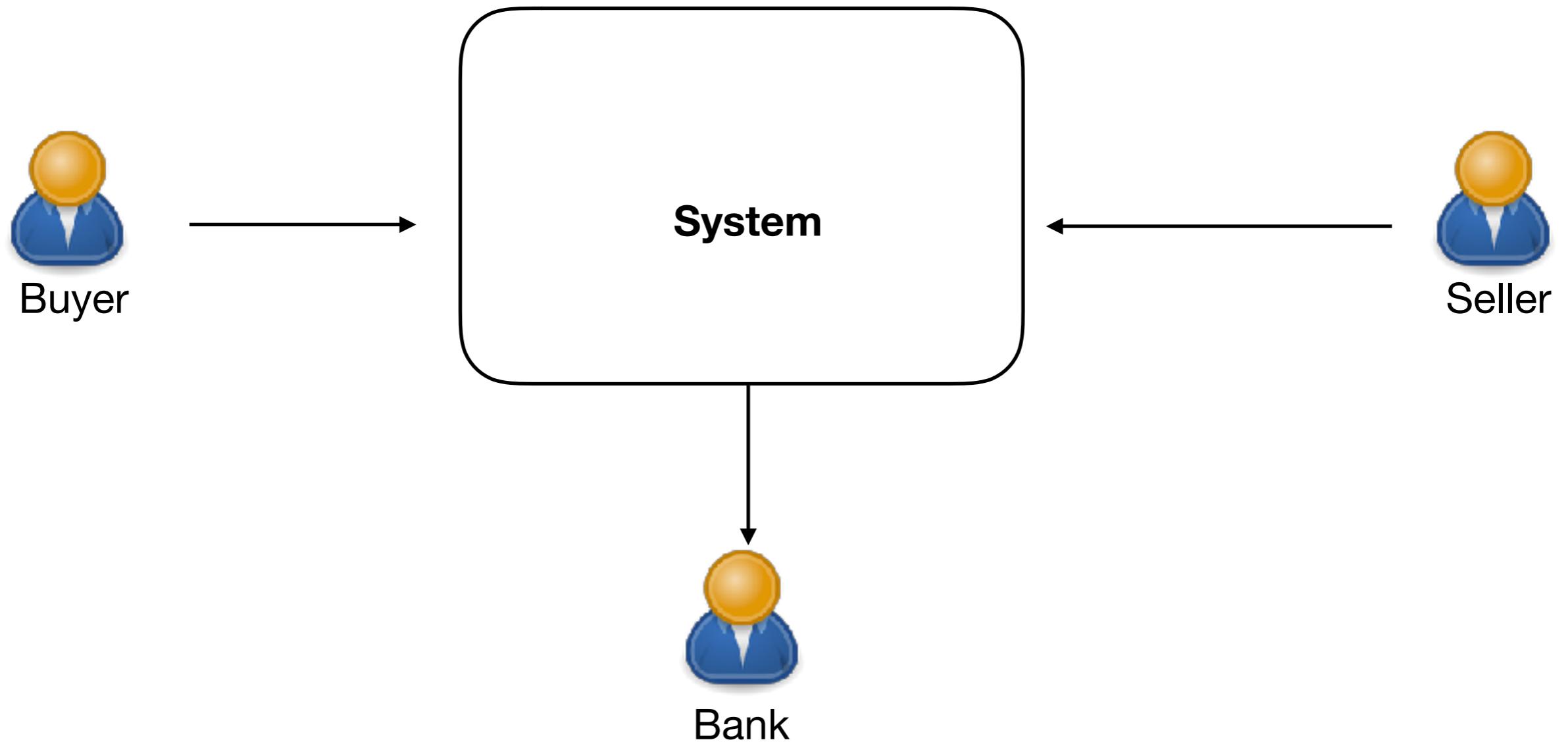
- Cloud Computing
- Micro services
- ML & Data Science
- DevSecOps

- Years of experience
- Role
- Expectations

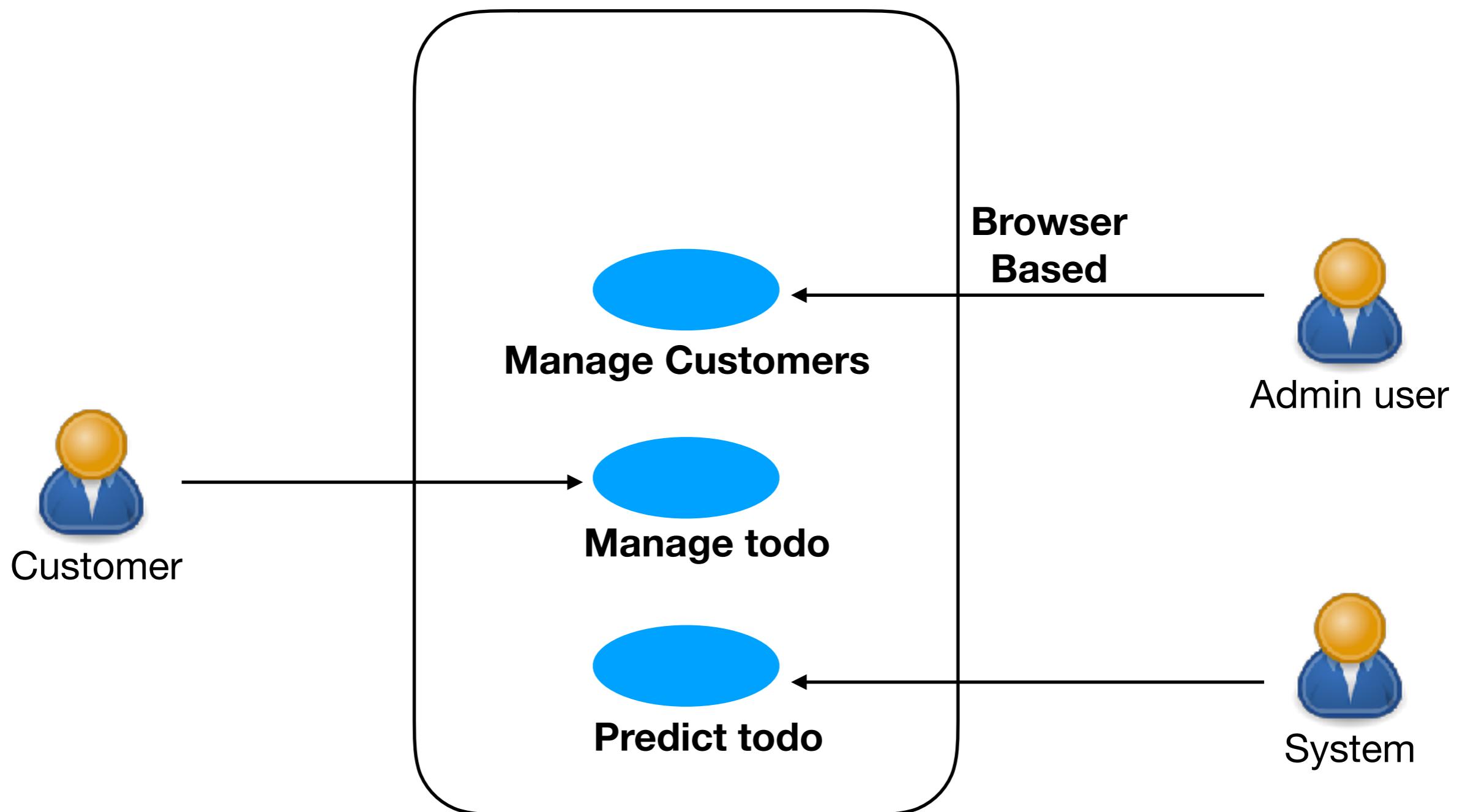
Context View



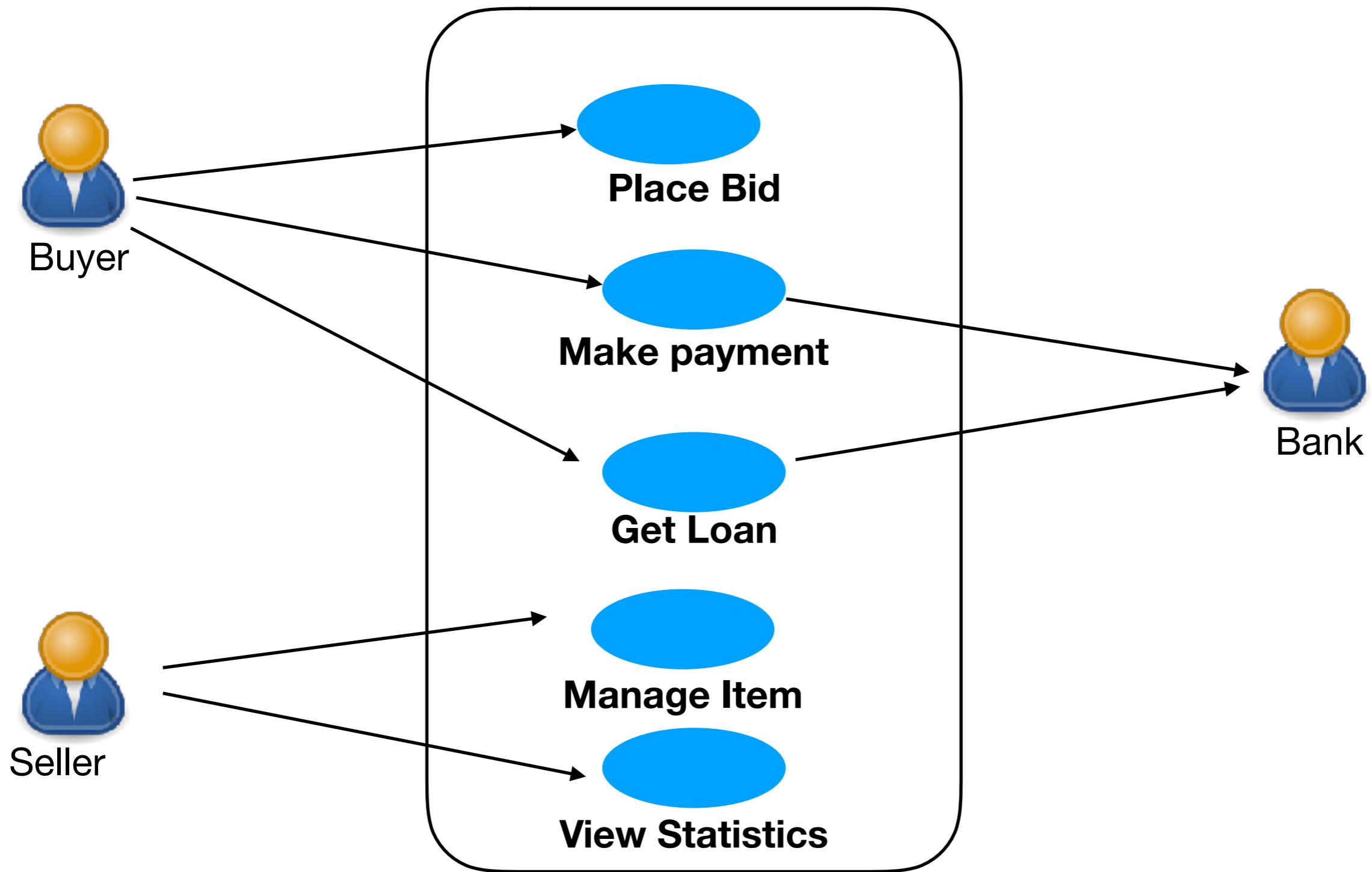
Context view



Functional view



Functional view



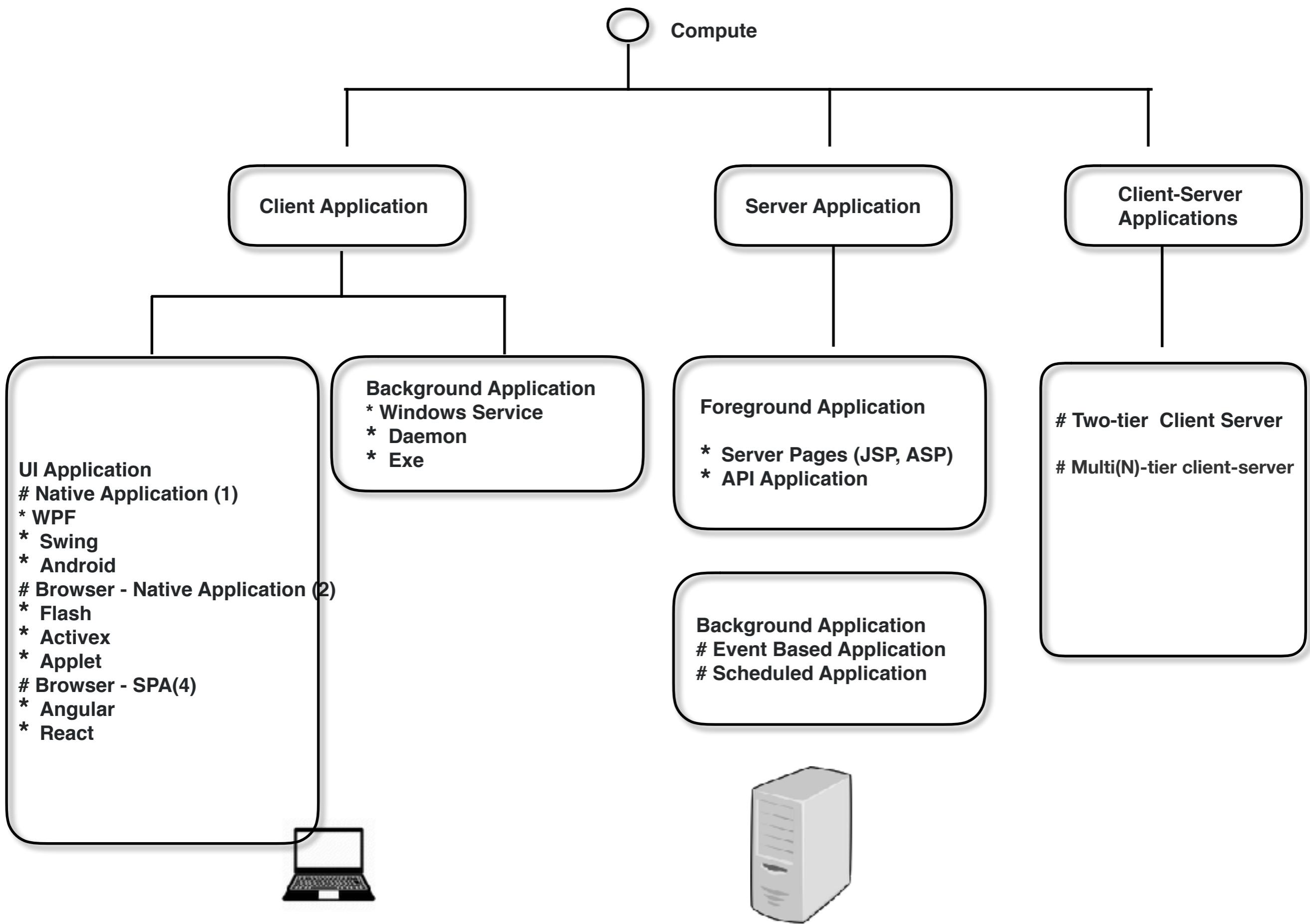
Compute

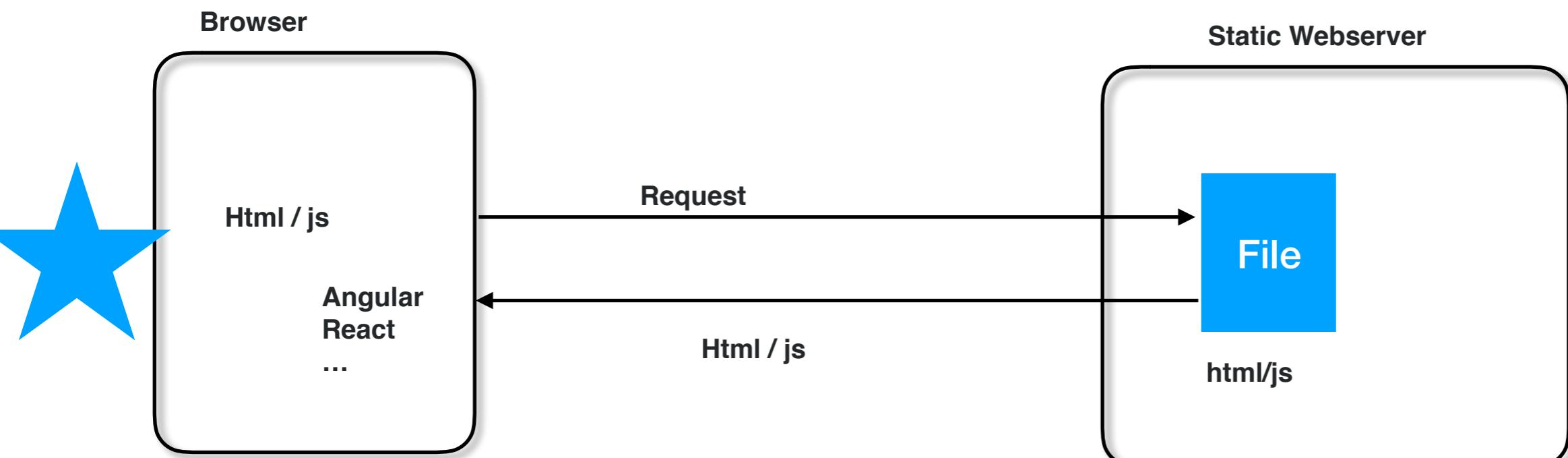
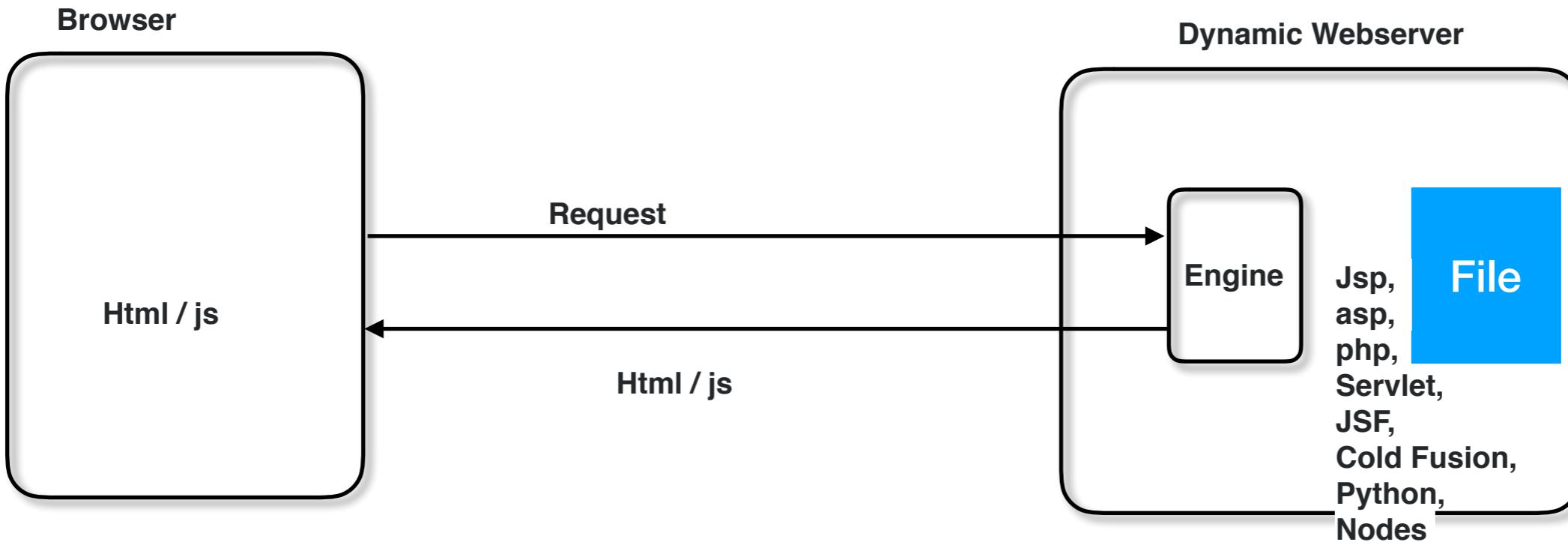
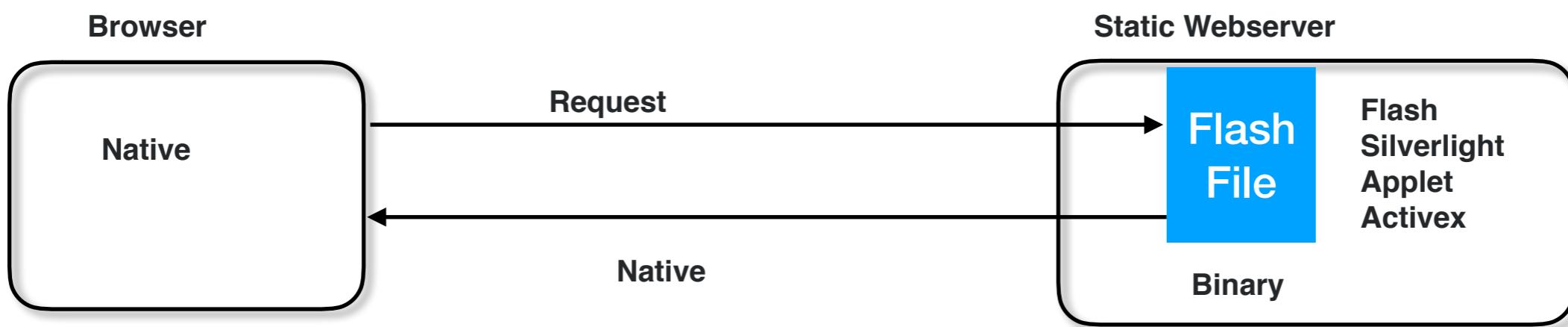
Storage

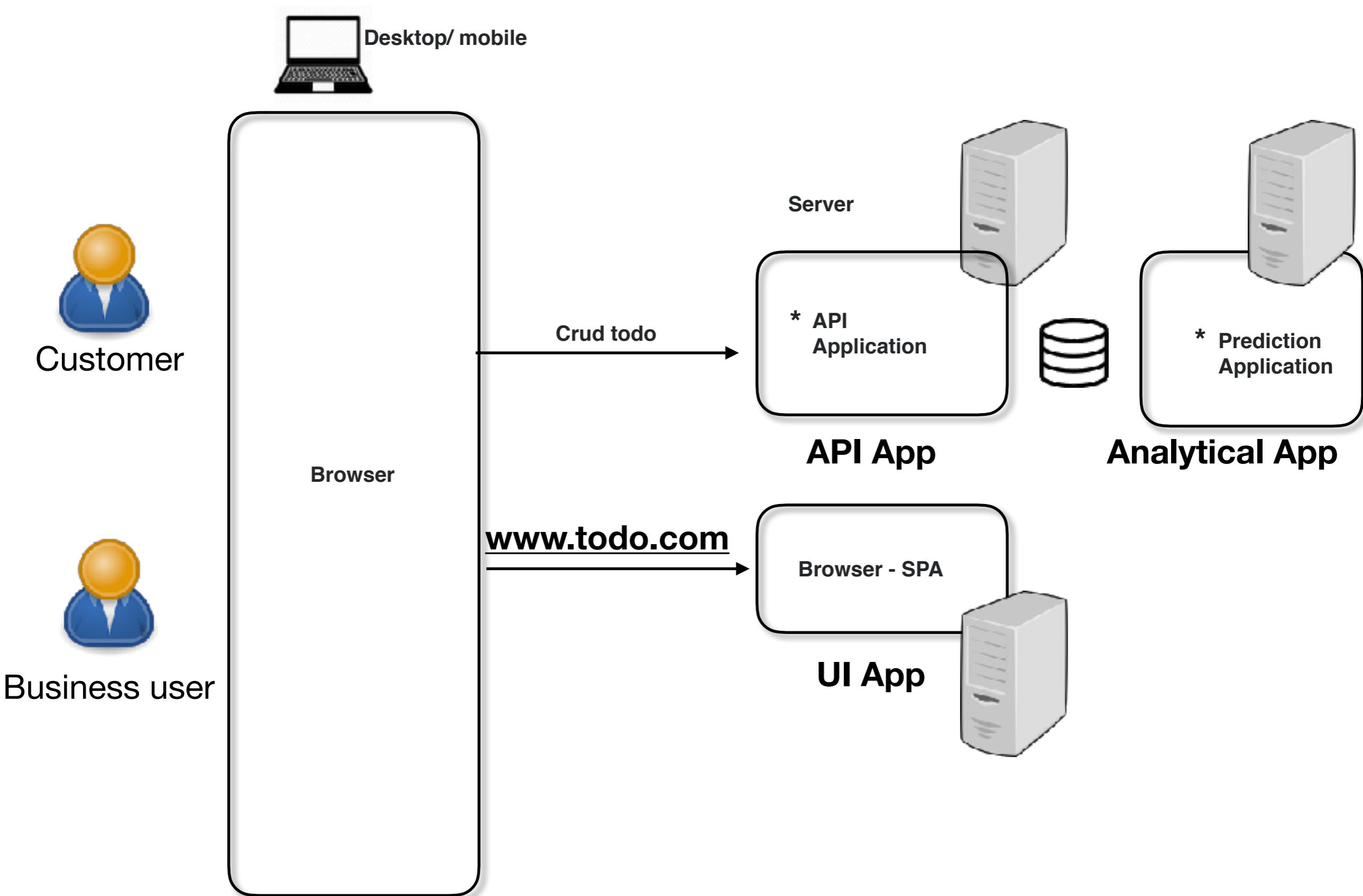
Message

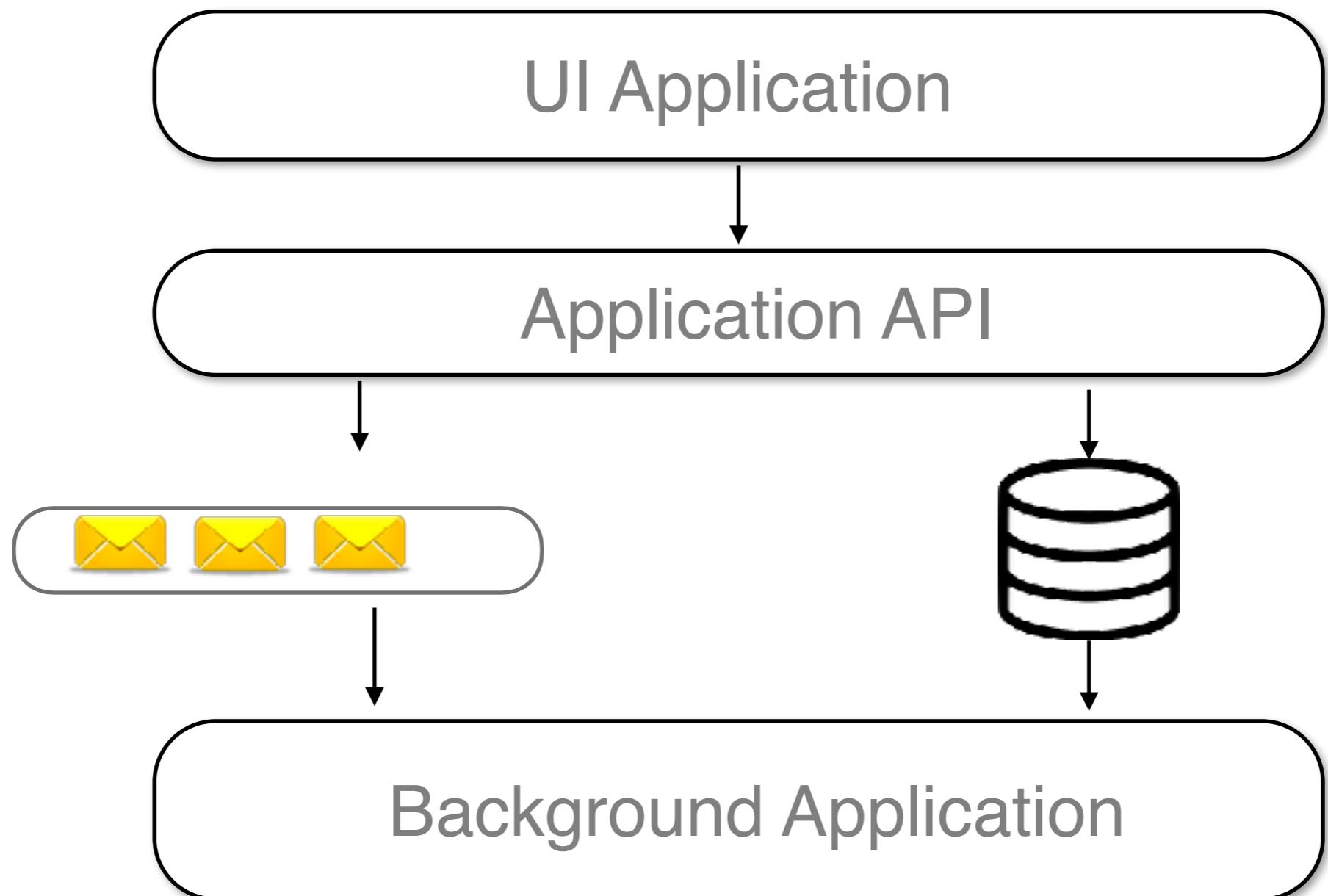
System Decomposition

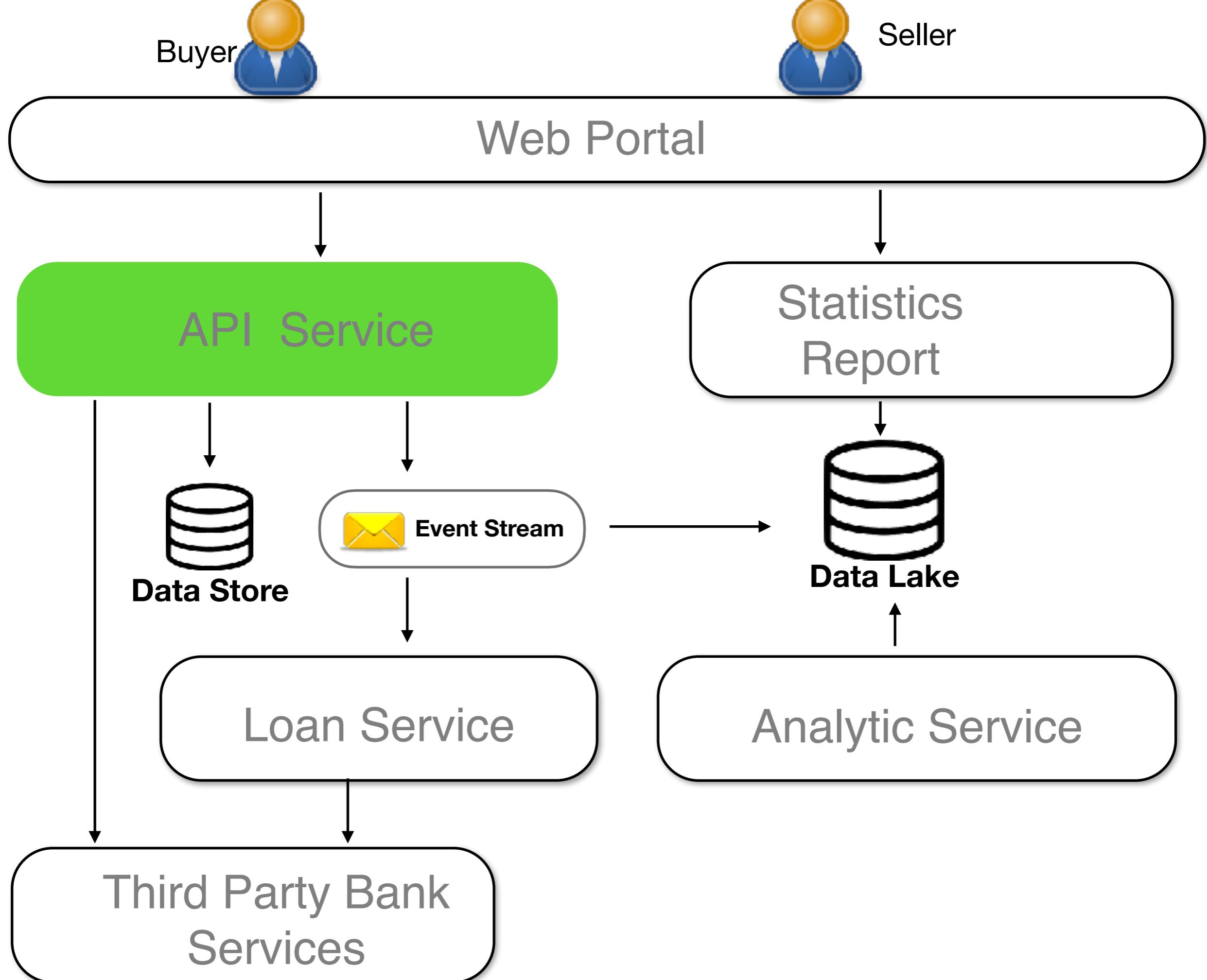




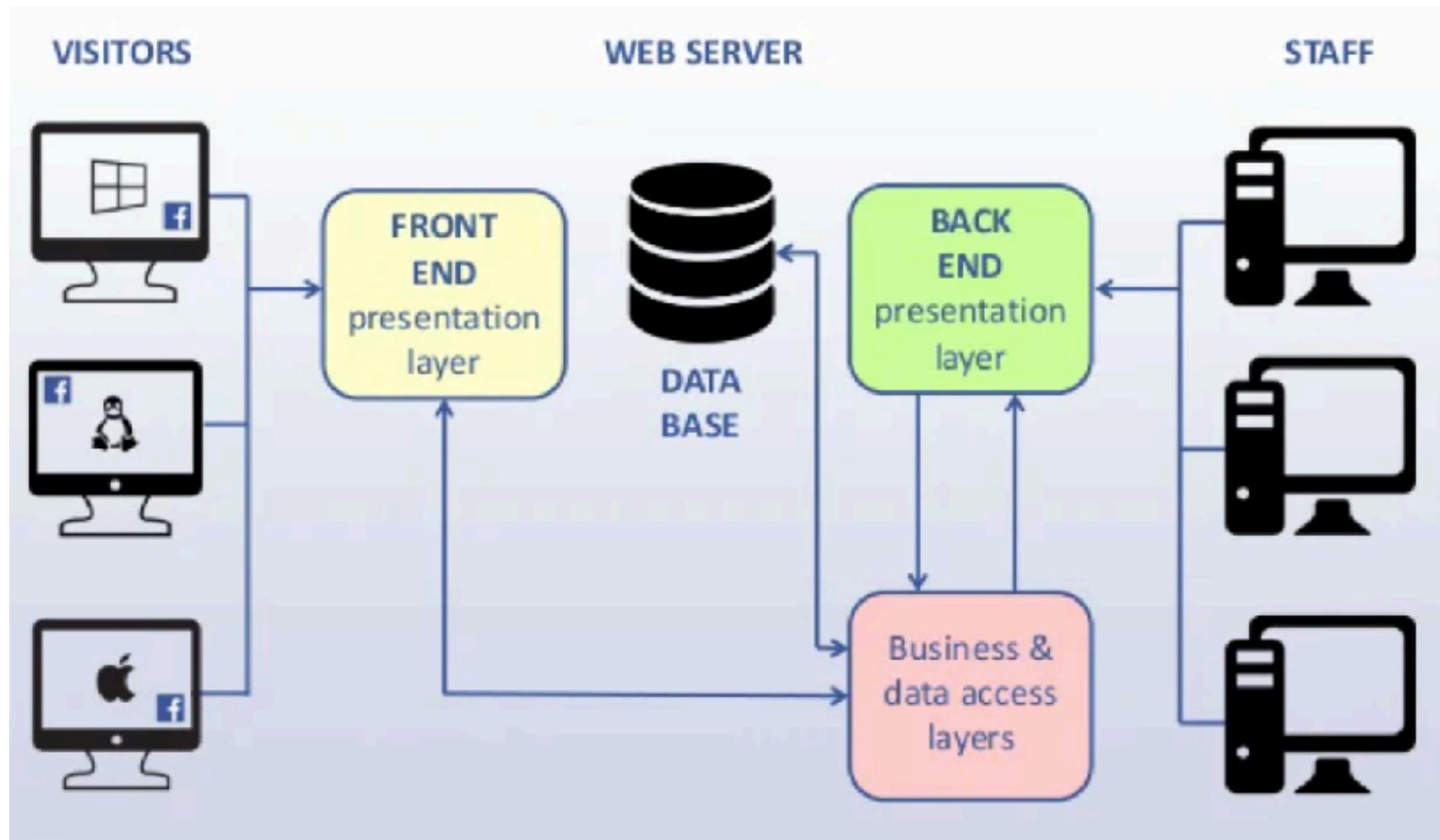


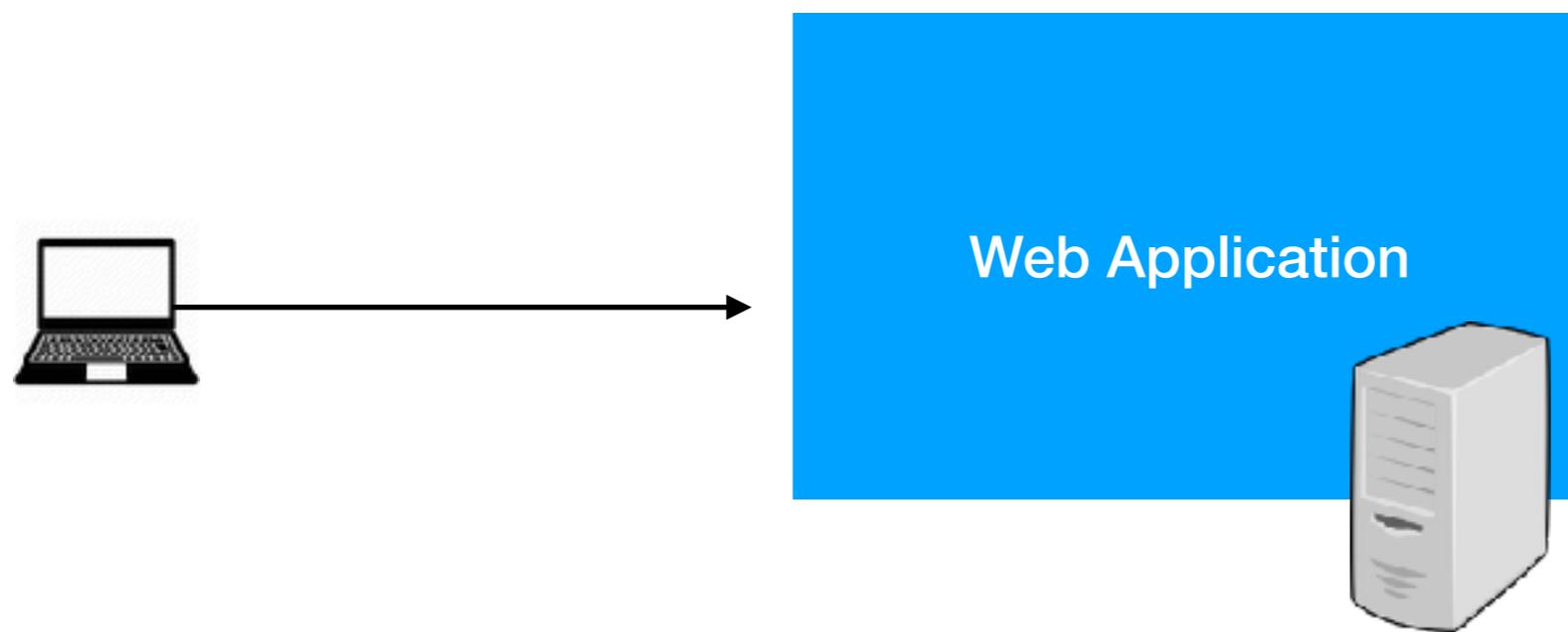




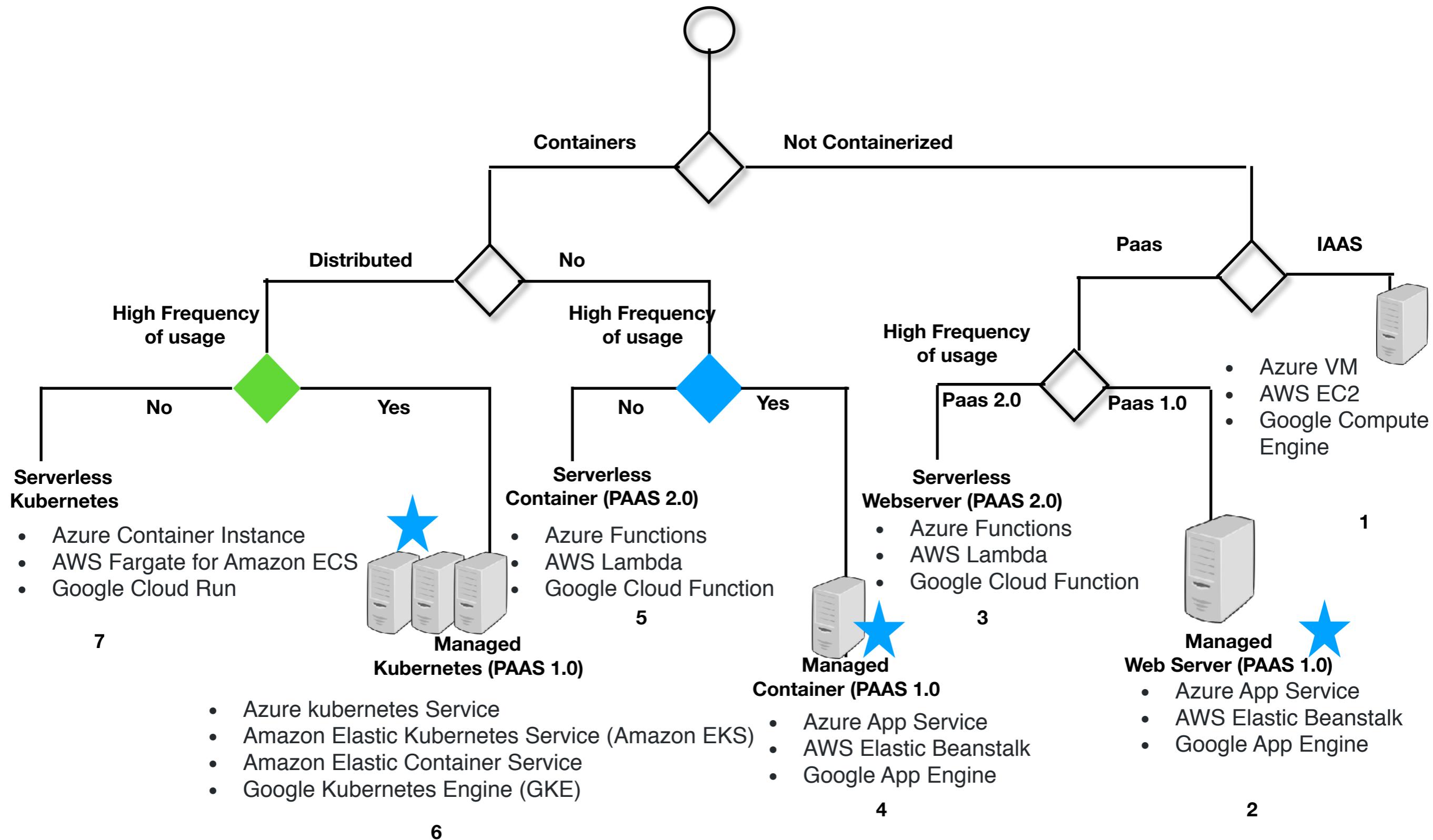


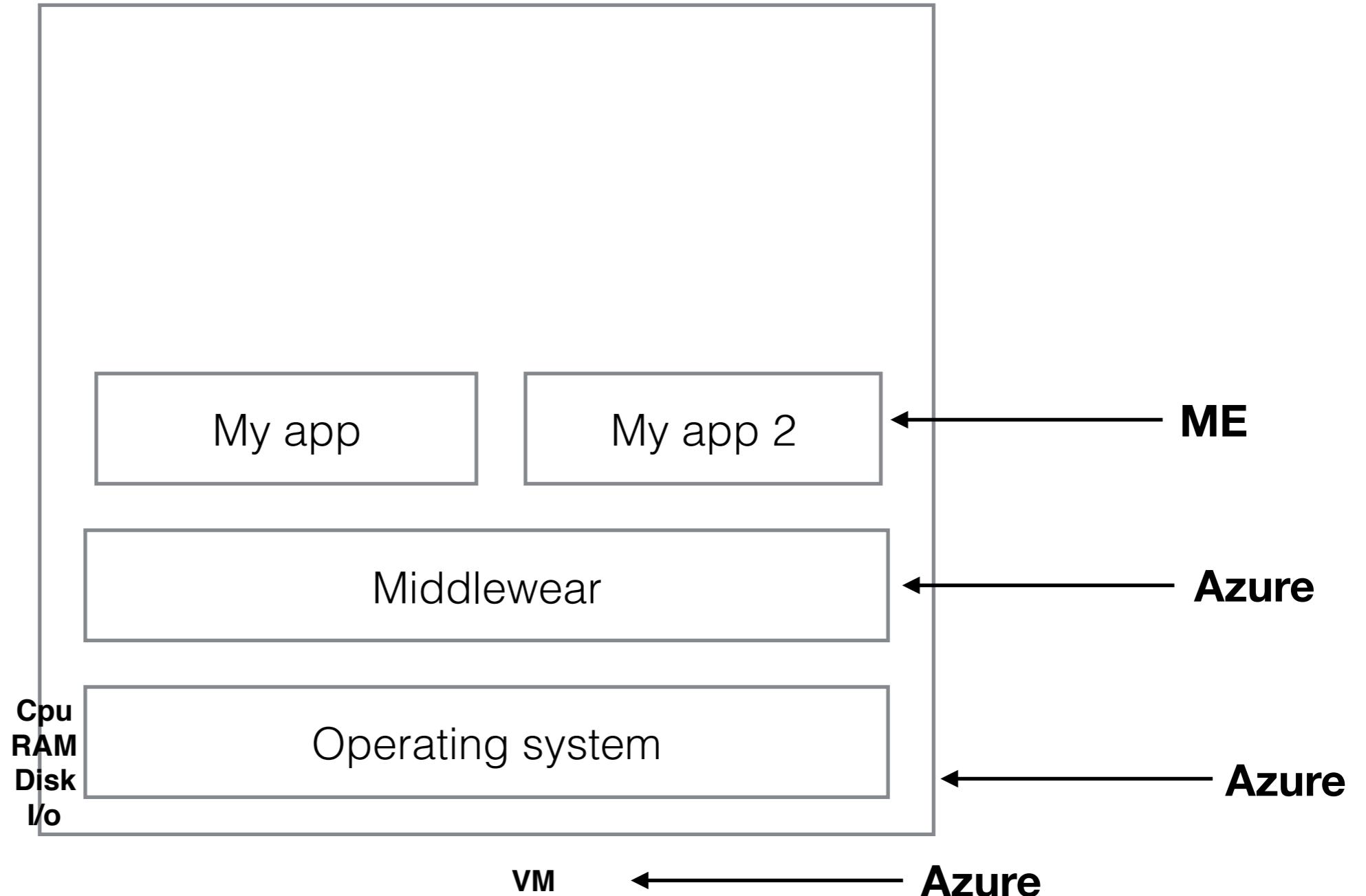
Facebook Reference Architecture

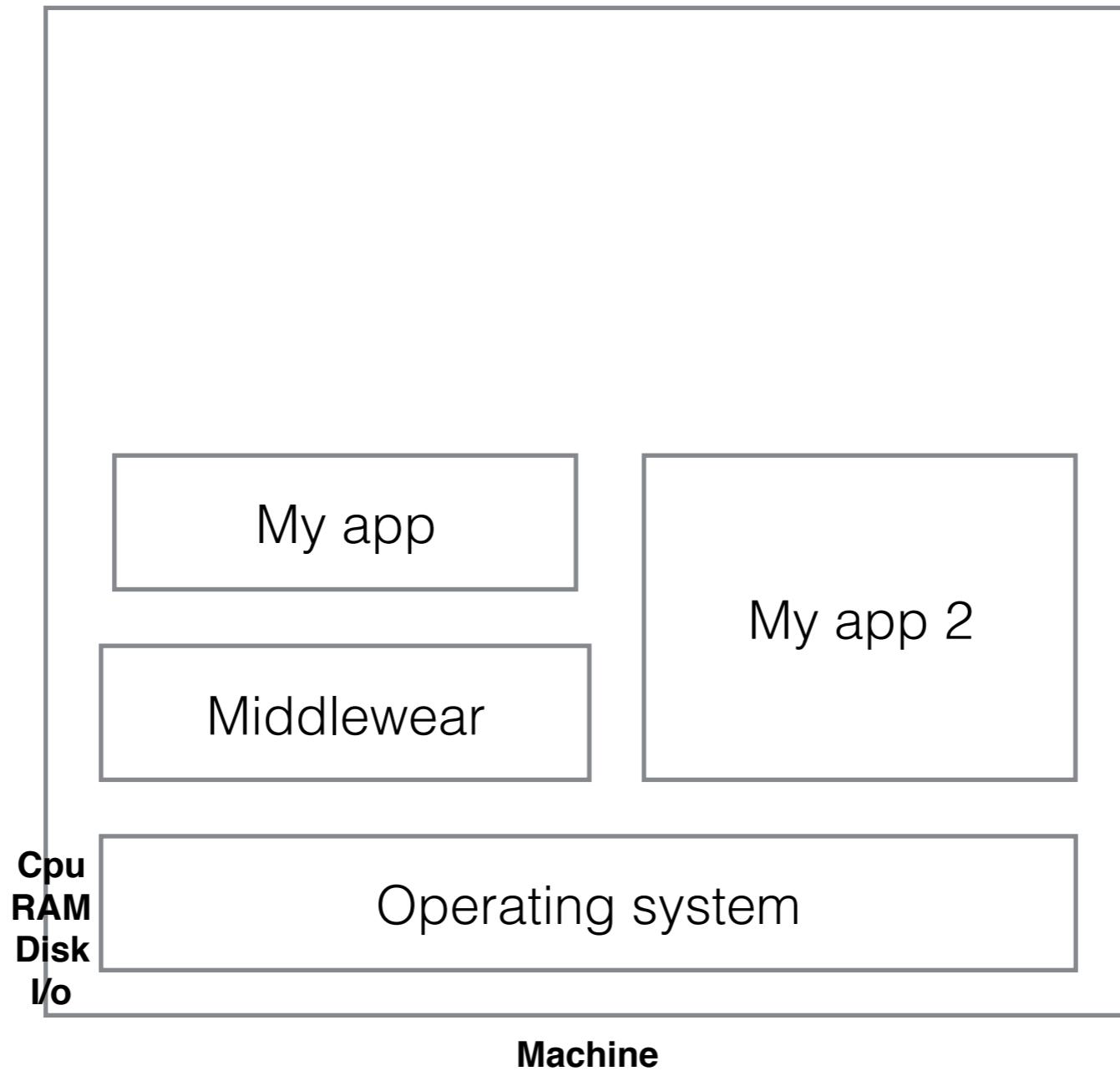


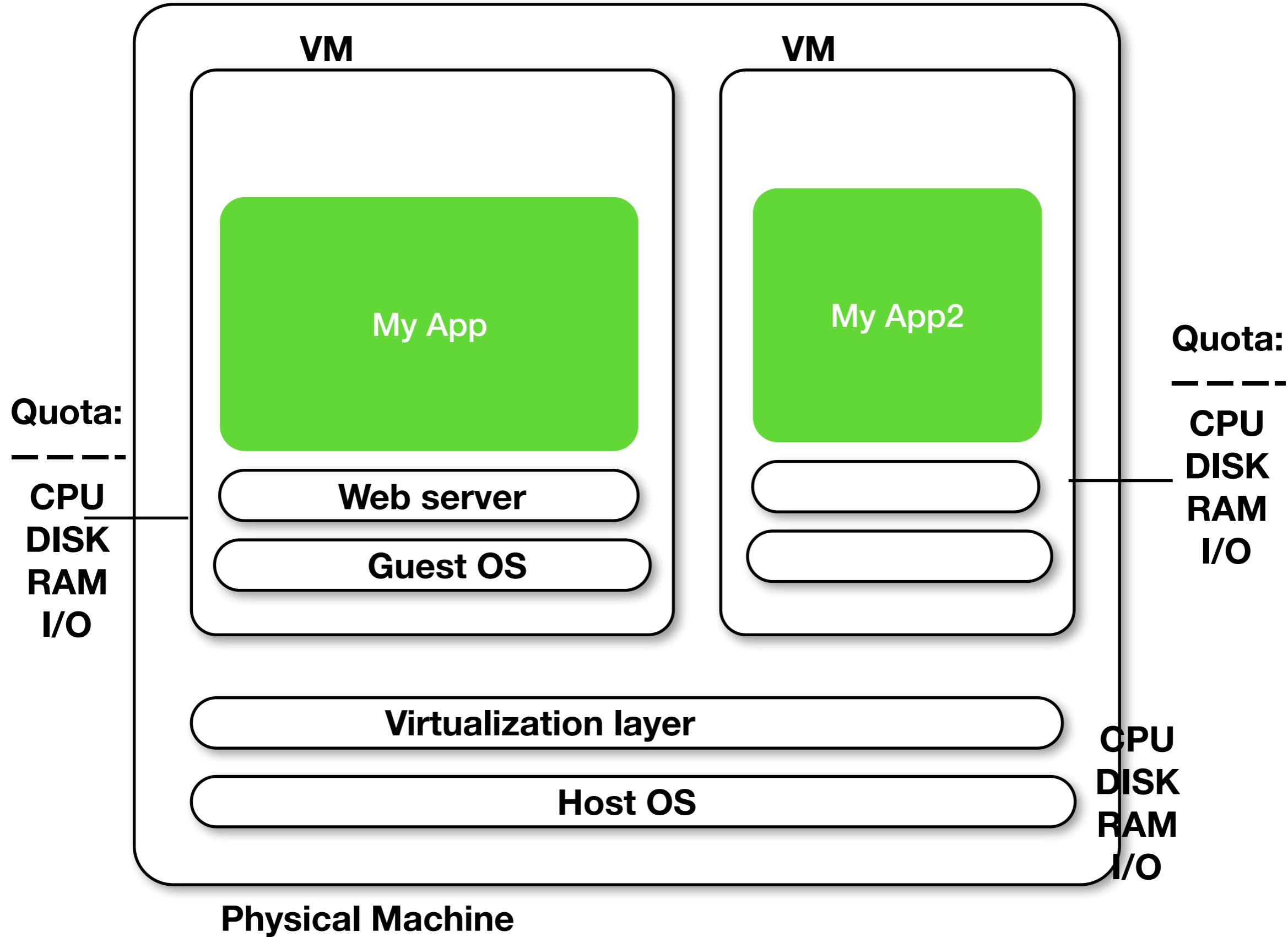


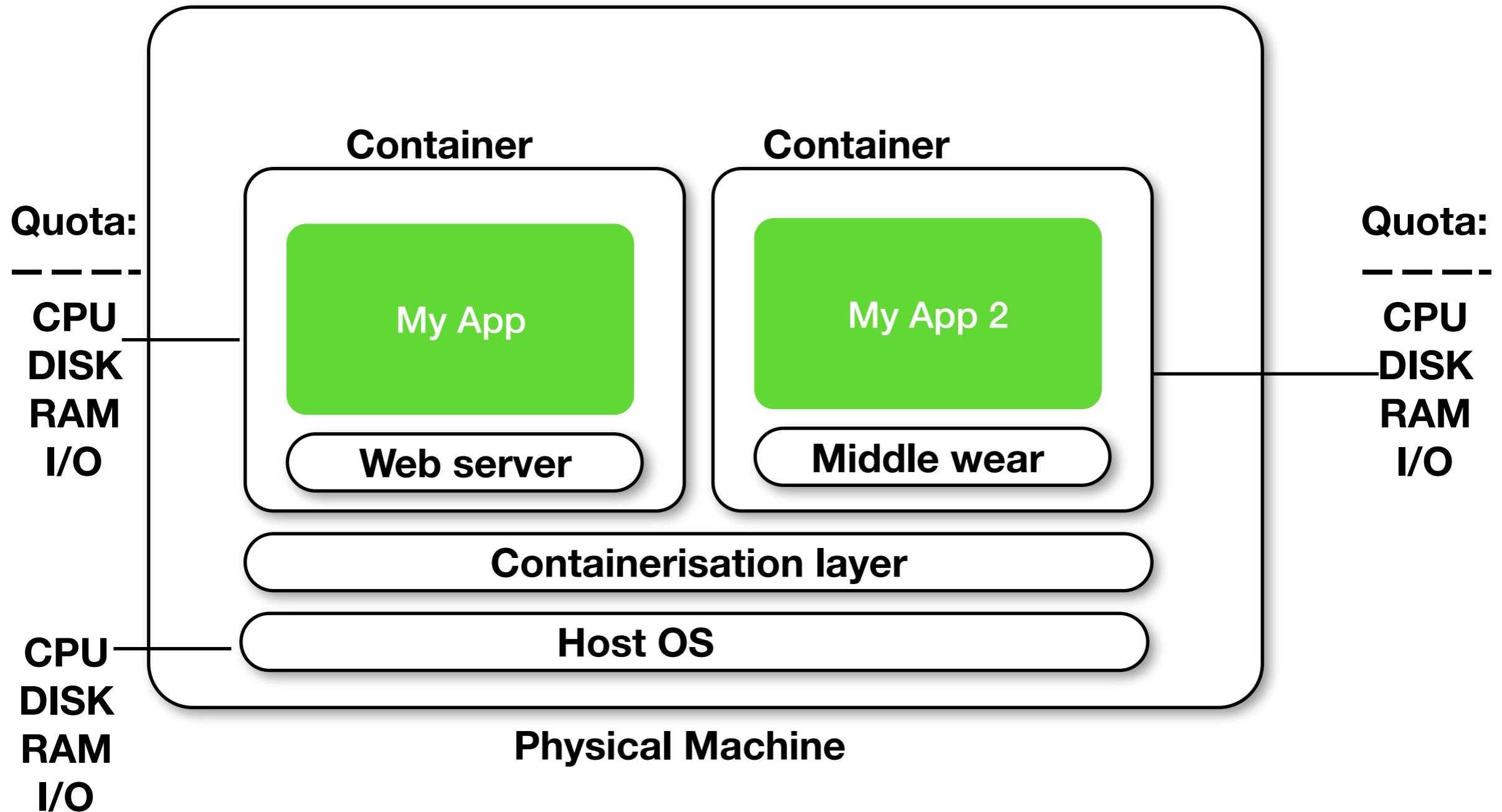
Deployment View



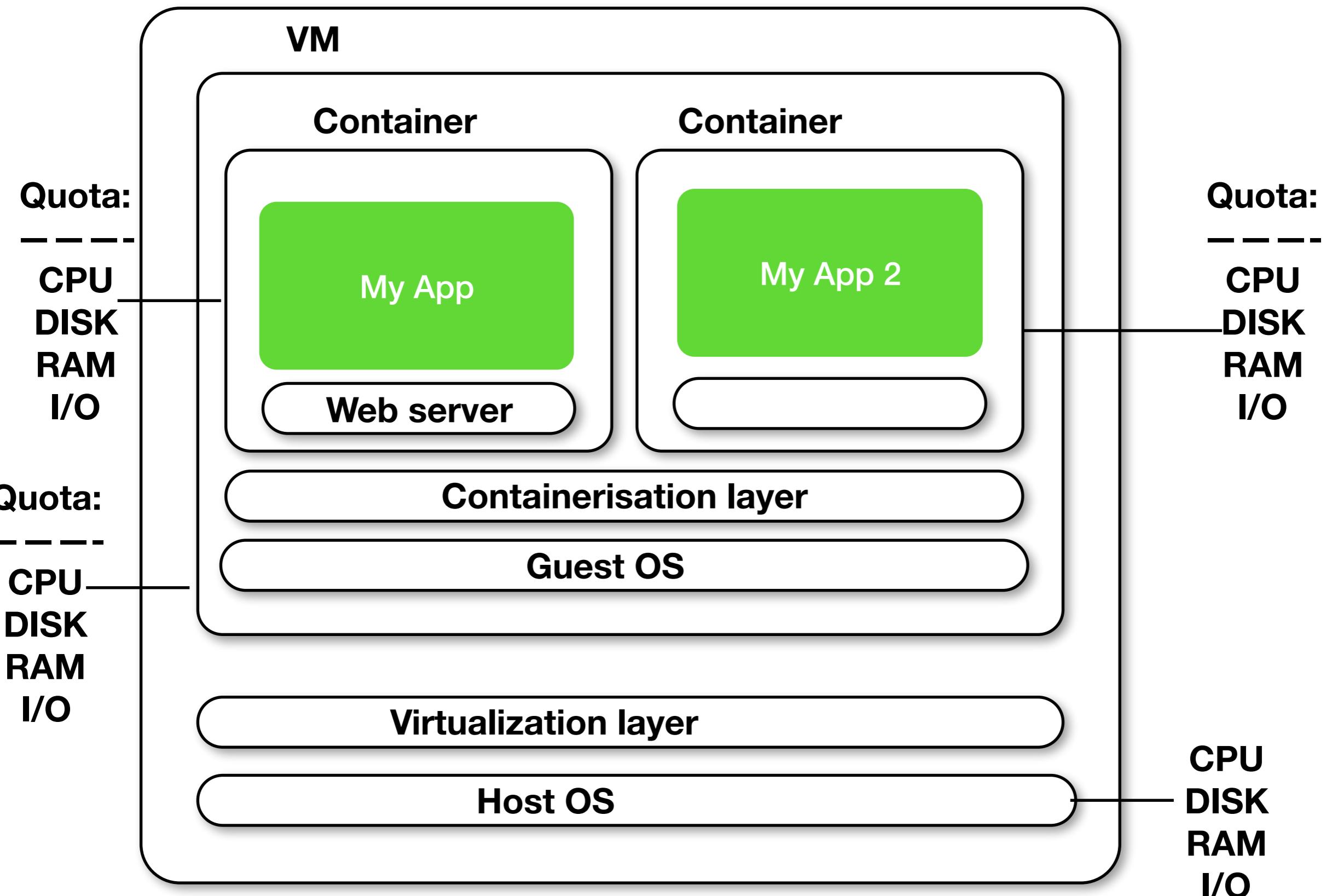








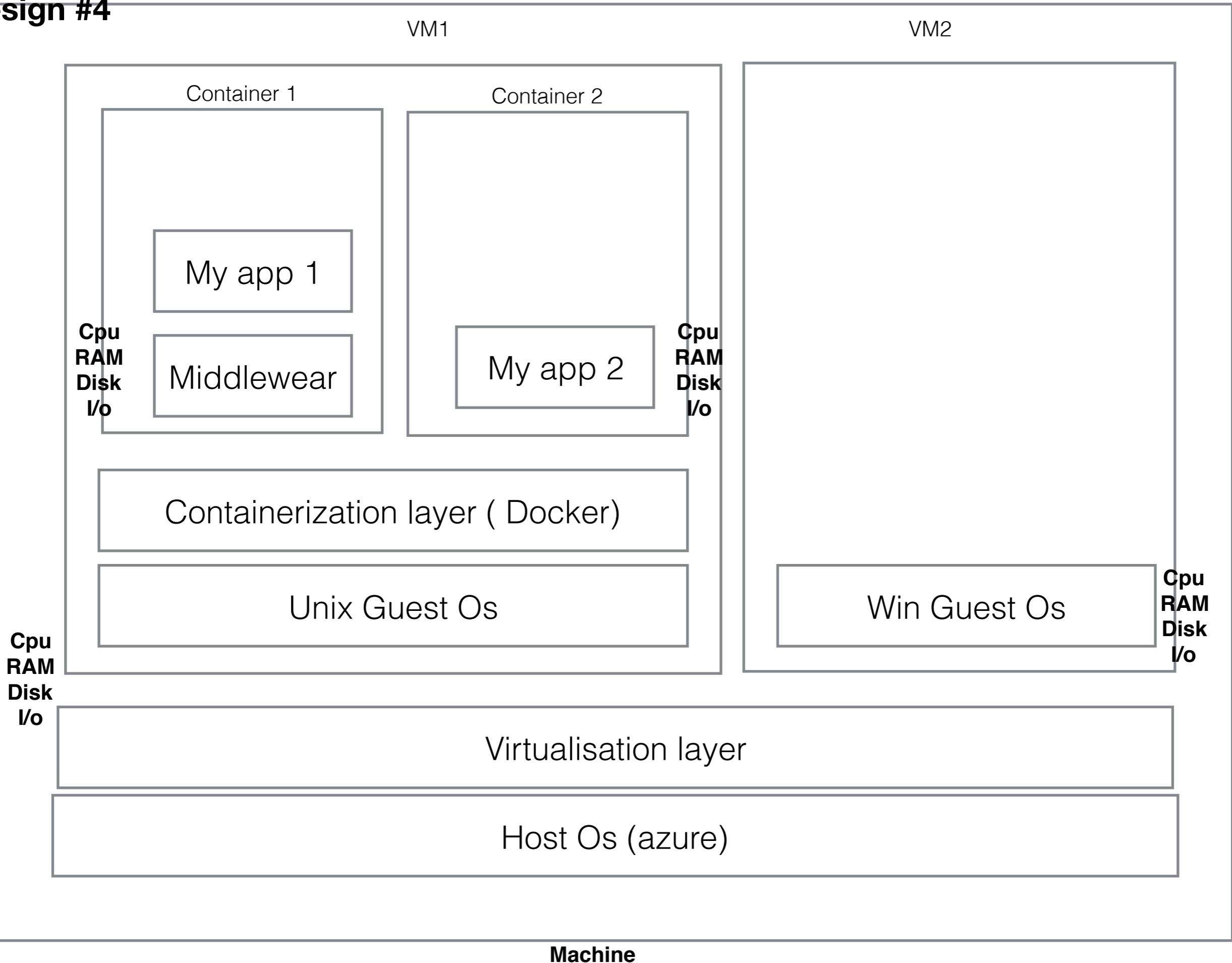
Physical Machine



Design #4

VM1

VM2



VM1

VM2

Container 1

Container 2

Cpu
RAM
Disk
I/o

My app 1

Middlewear

Cpu
RAM
Disk
I/o

My app 2

Containerization layer (Docker)

Cpu
RAM
Disk
I/o

Unix Guest Os

My app 3

Containerization layer

Cpu
RAM
Disk
I/o

Unix Guest Os

Virtualisation layer

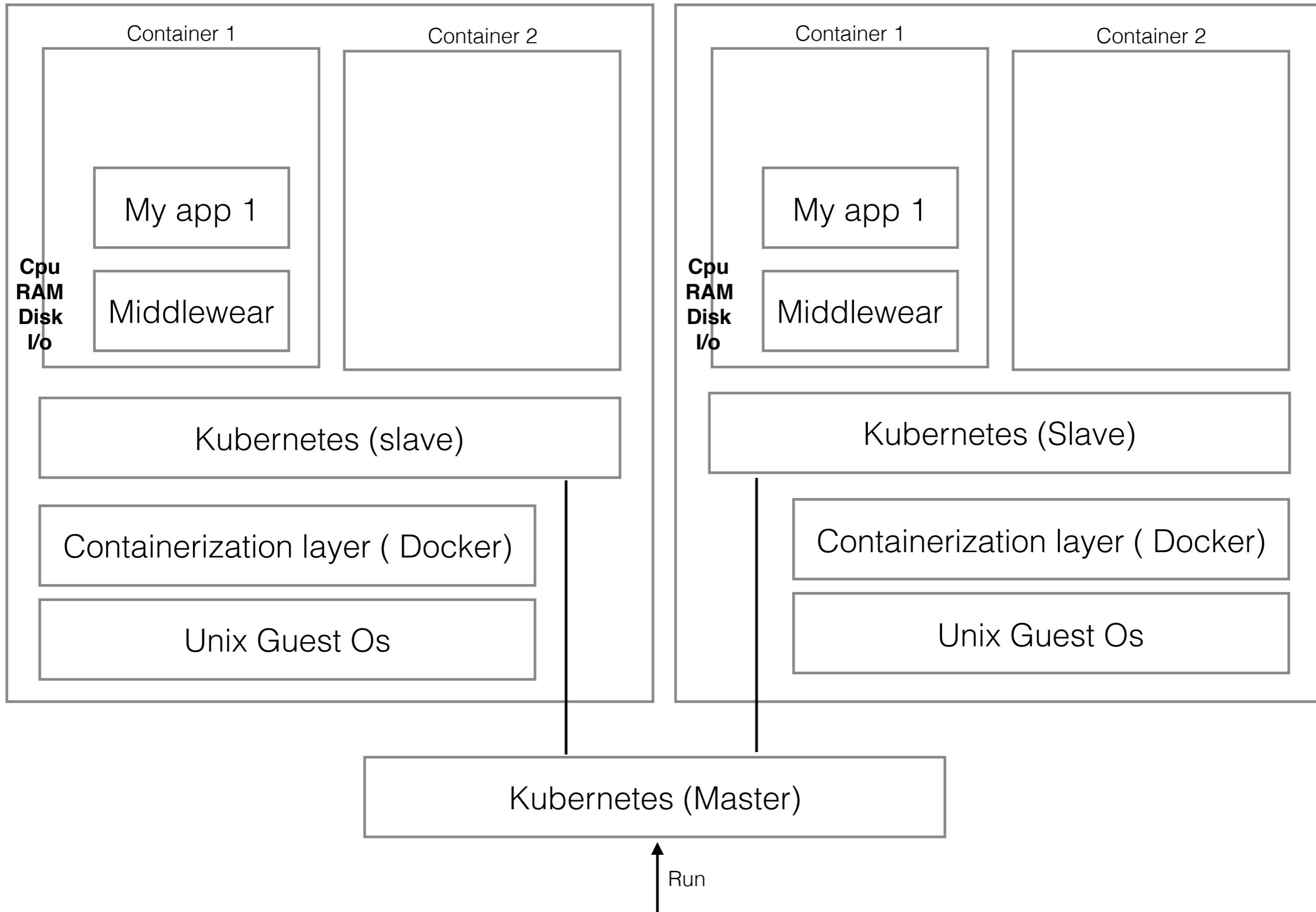
Host Os (azure)

Machine

Design #5

VM1

VM2



My app

OS

Machine

My app

Middlewear

OS

Machine

My app

Middlewear

Guest OS

VM

Machine

My app

Middlewear

Container

Guest OS

VM

Machine

Kubernetes

Cluster

My app

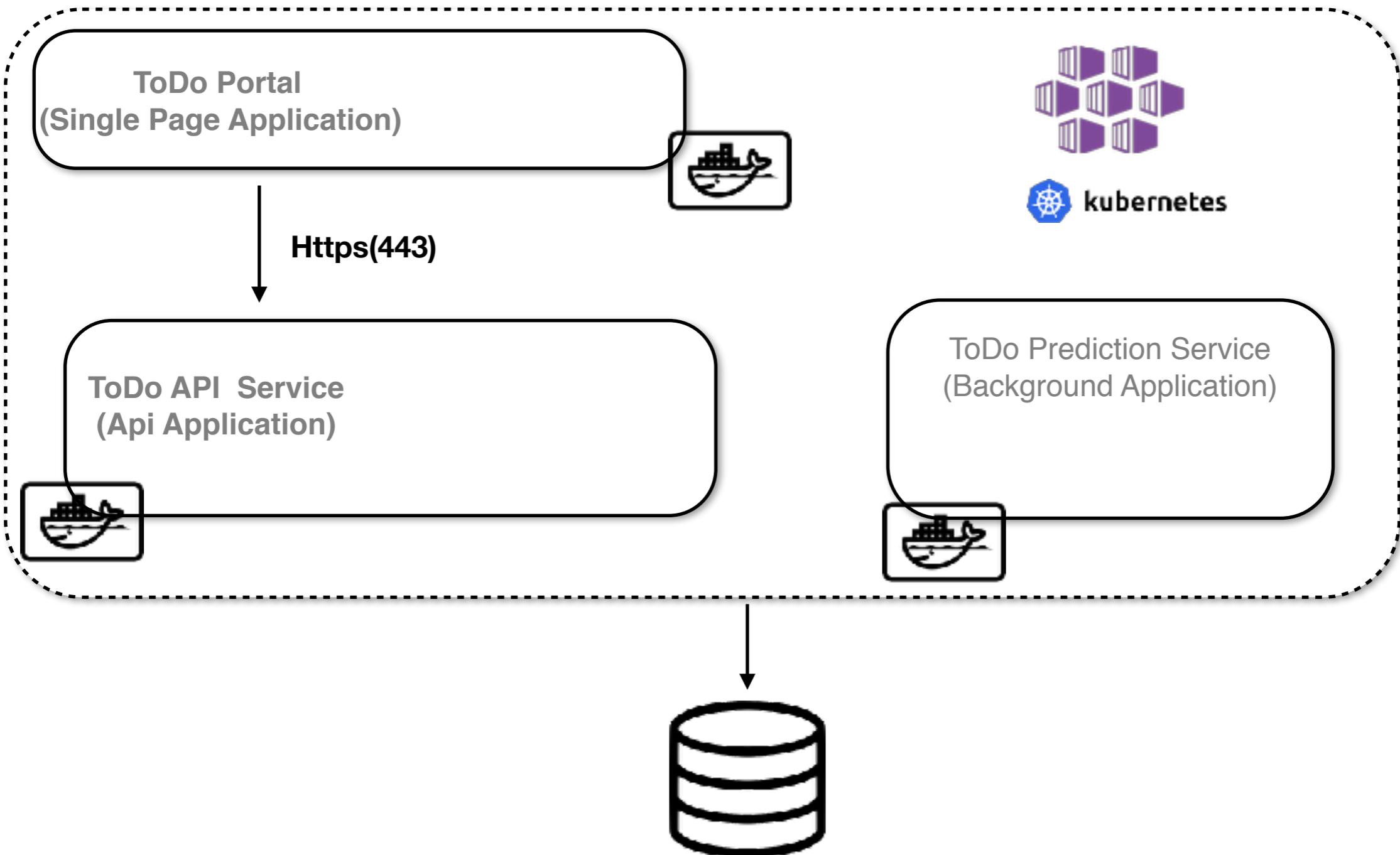
Middlewear

Container

Guest OS

VM

Machine



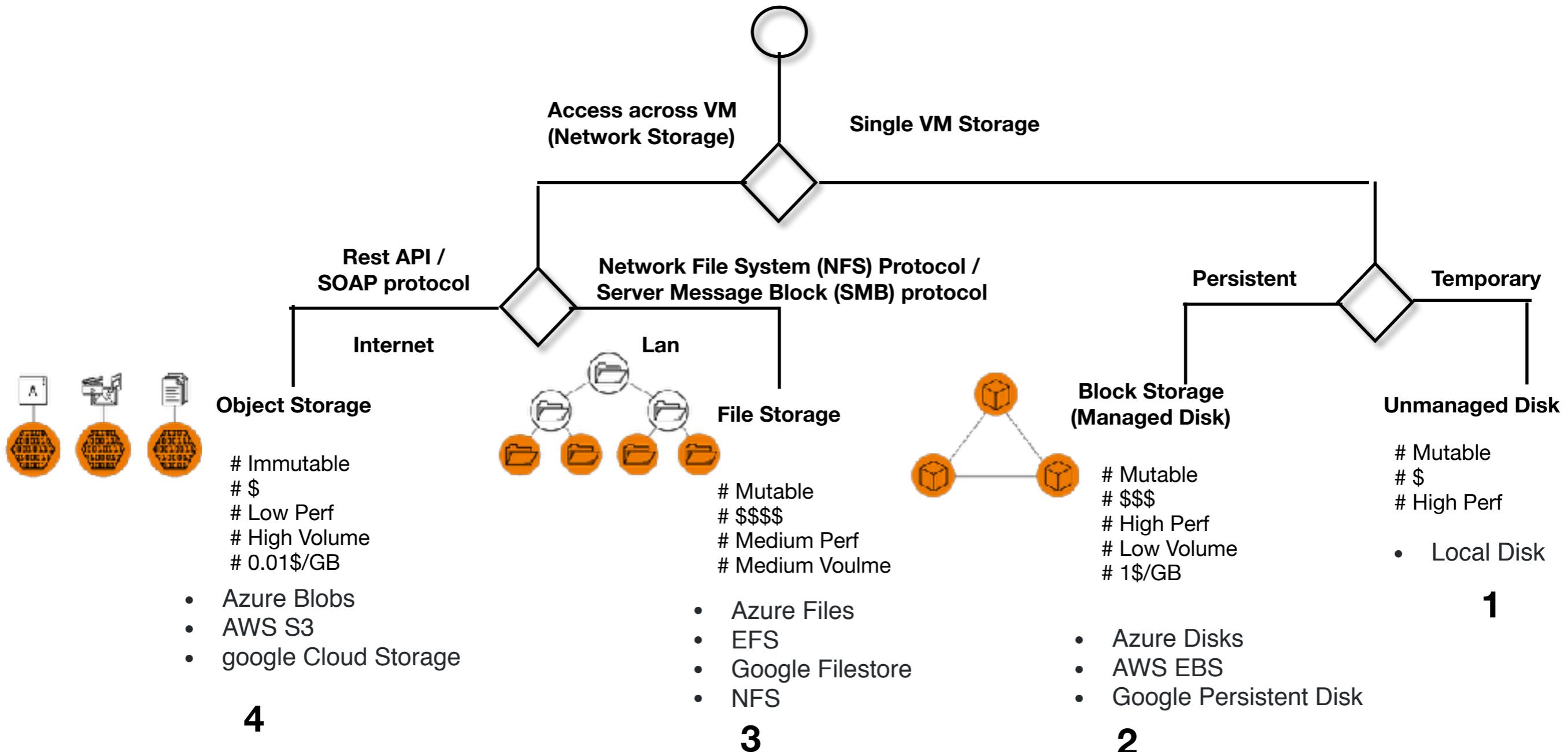
System Storage

Binary

Lob

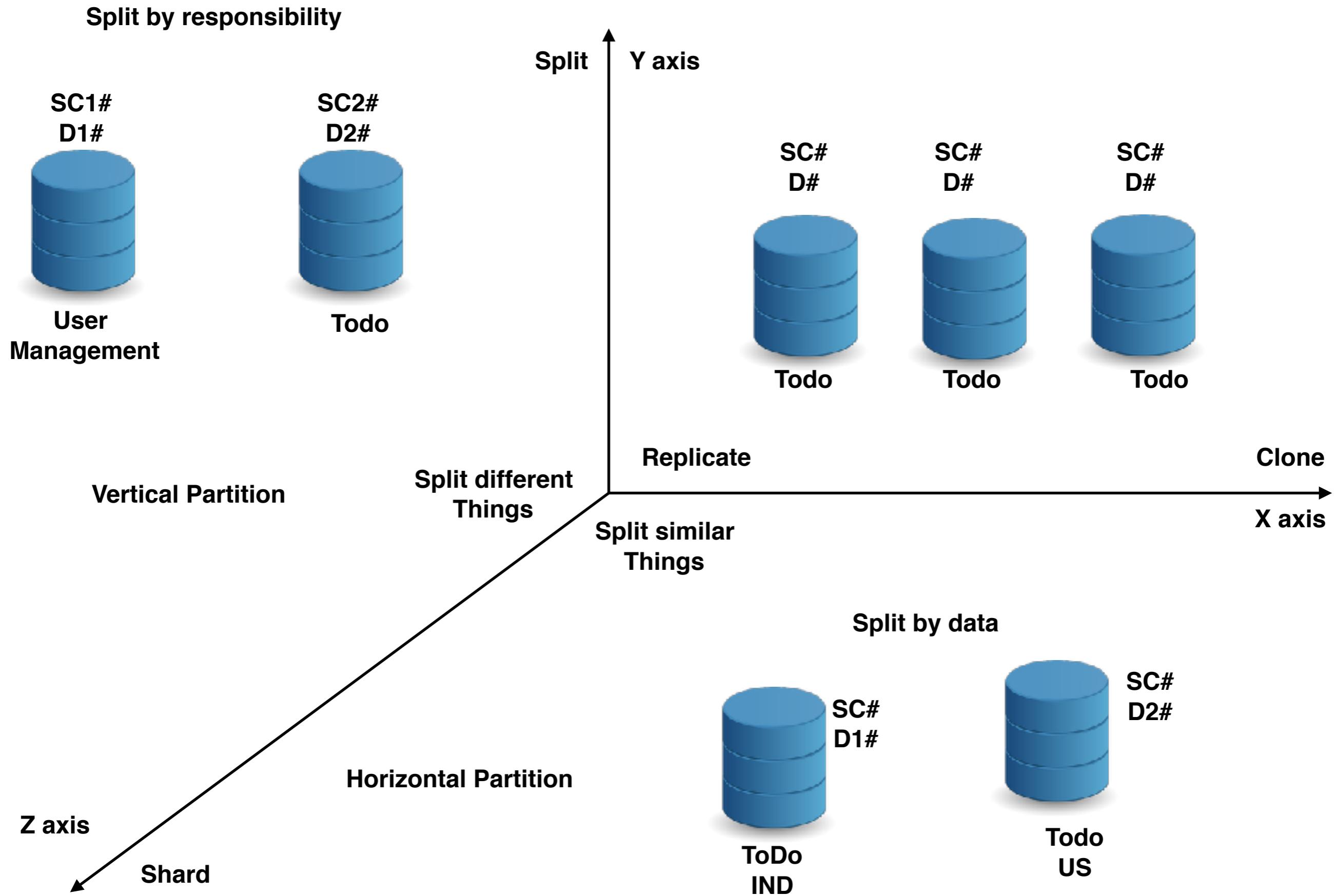
Analytics

Binary Storage

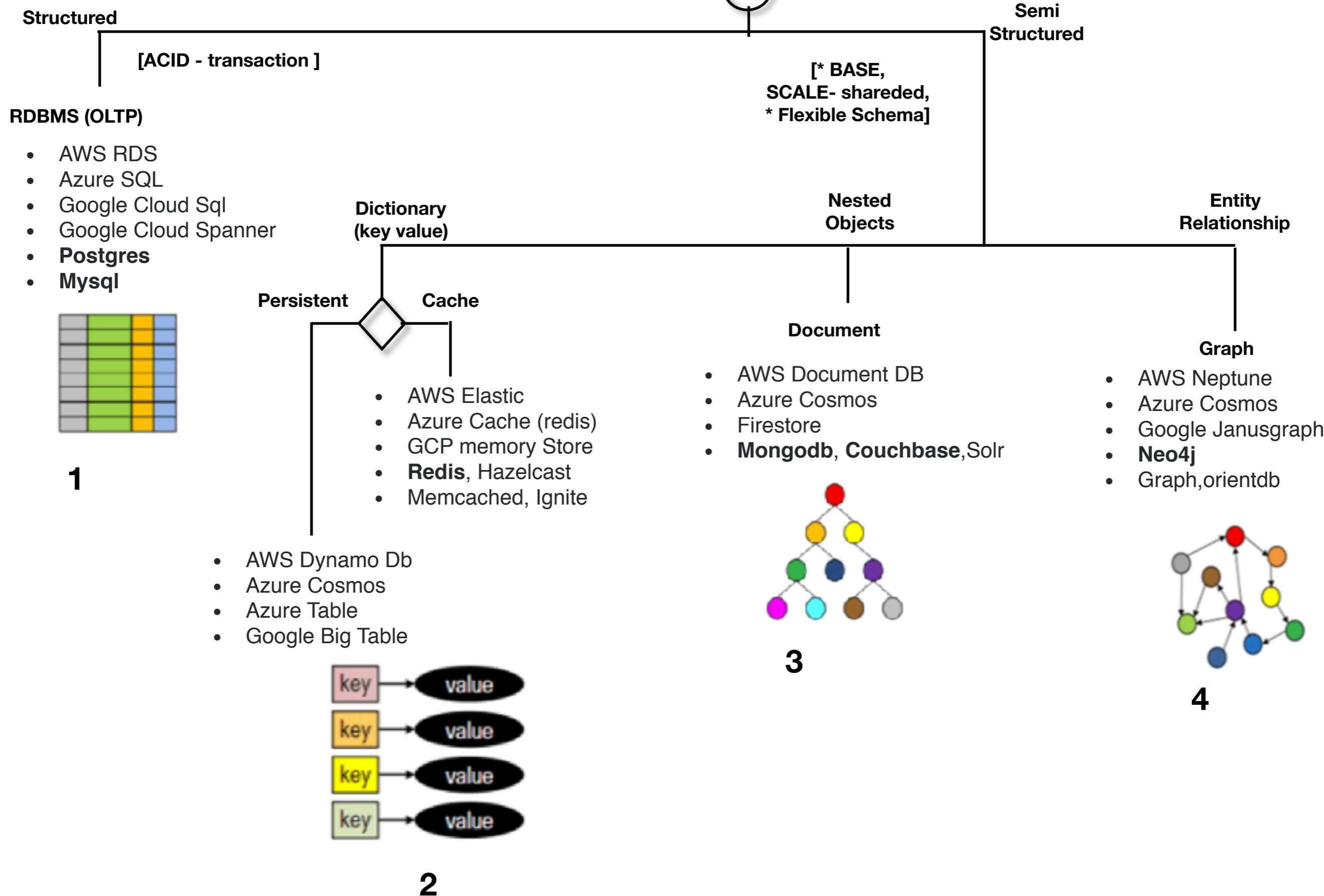


* AWS DataSync subscription is required to provide support for Server Message Block (SMB) protocol

Scalability Cube - 50 rules for high Scalability

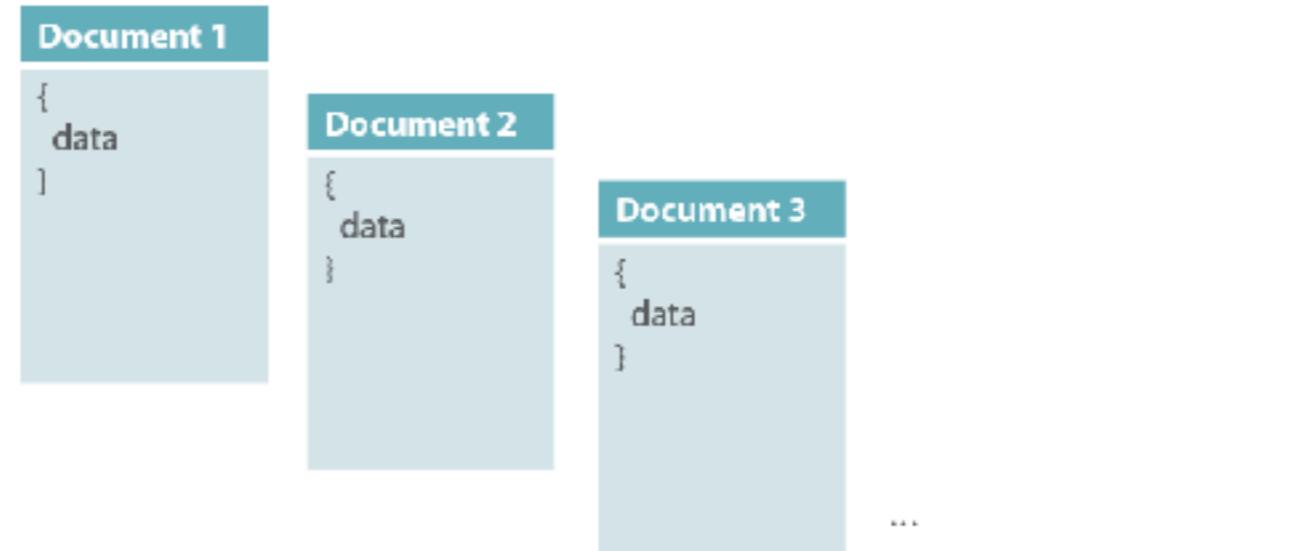


LOB



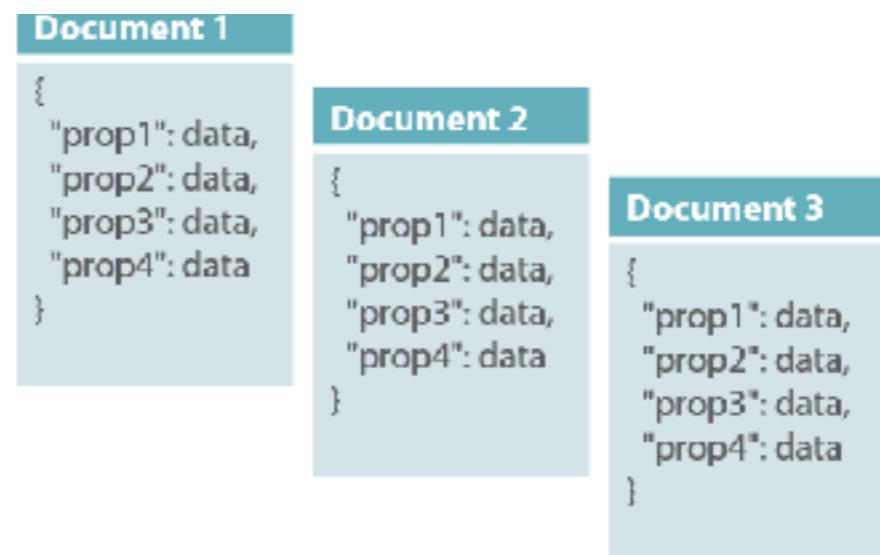
Rows vs. Documents

Table	
Row 1	Data
Row 2	Data
Row 3	Data
...	:



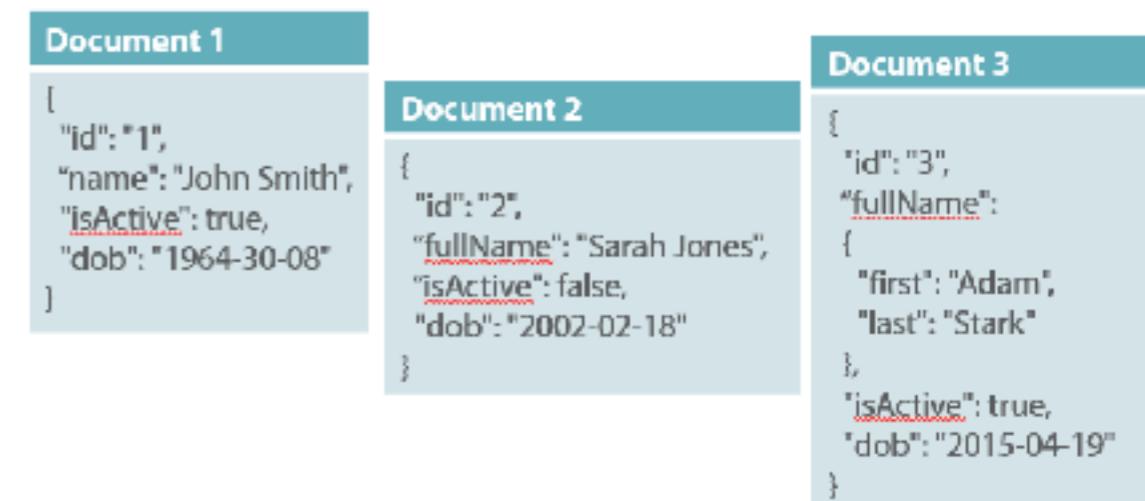
Columns vs. Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data



Schema vs. Schema-Free

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987



Normalized vs. Denormalized

User Table		
User ID	Name	Dob
1	John Smith	8/30/1964

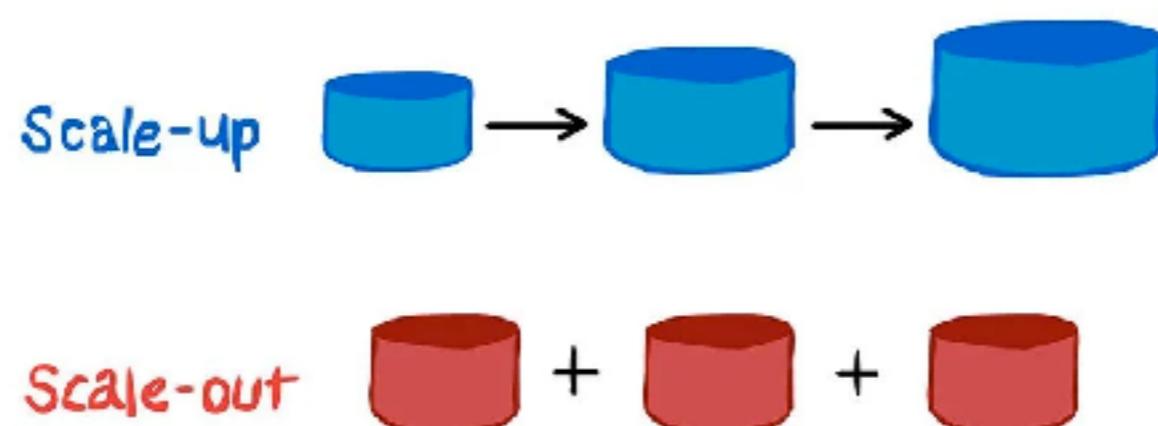
Holdings Table

StockID	User ID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

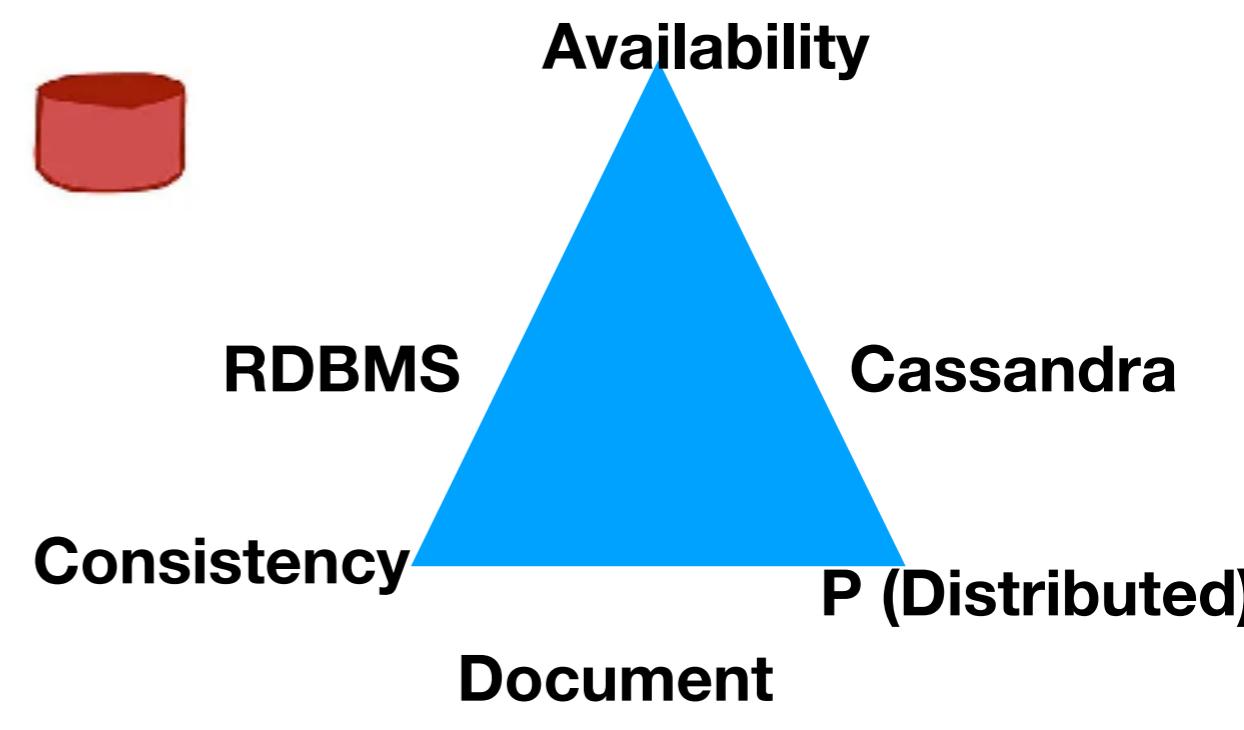
Document

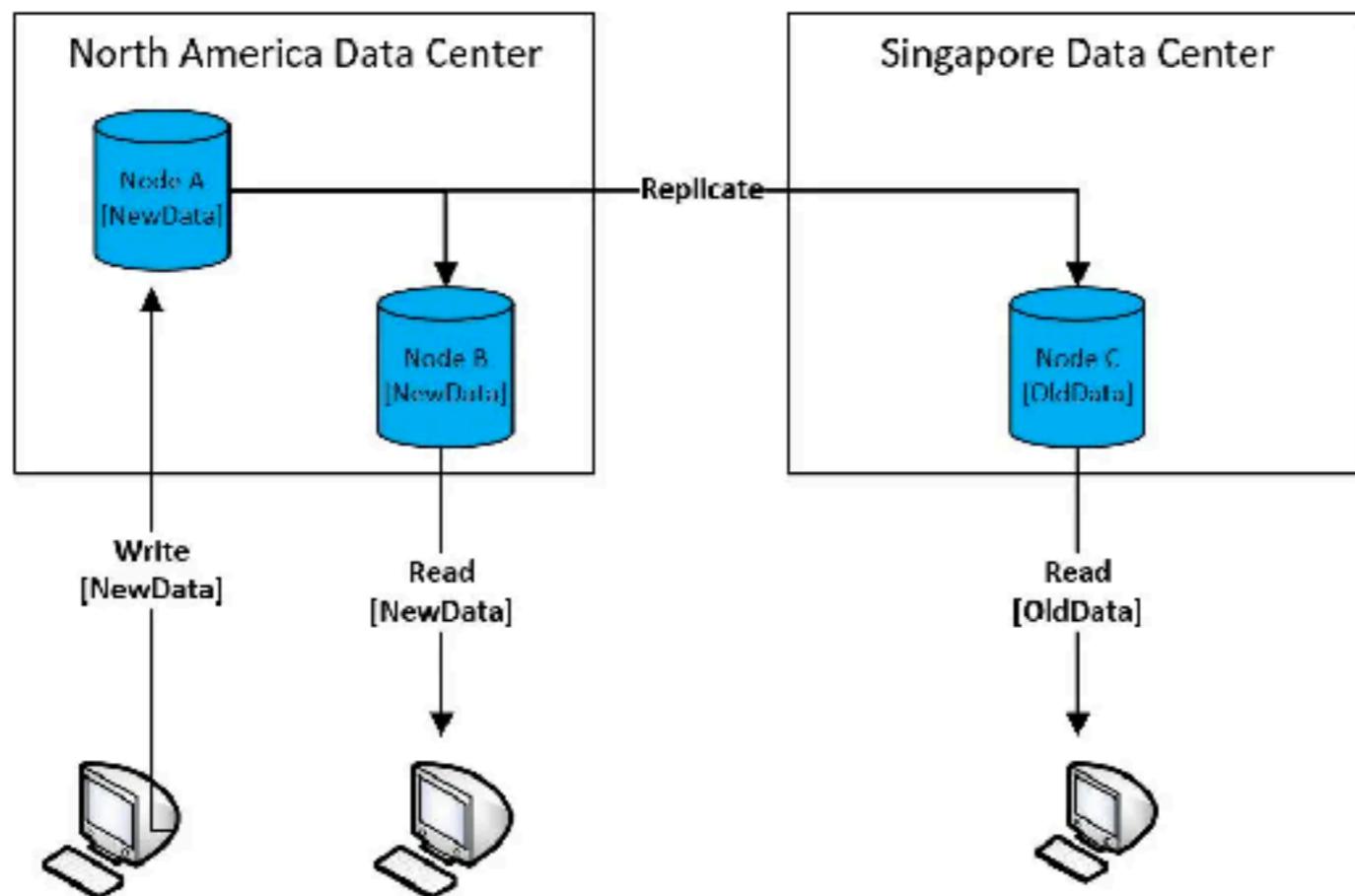
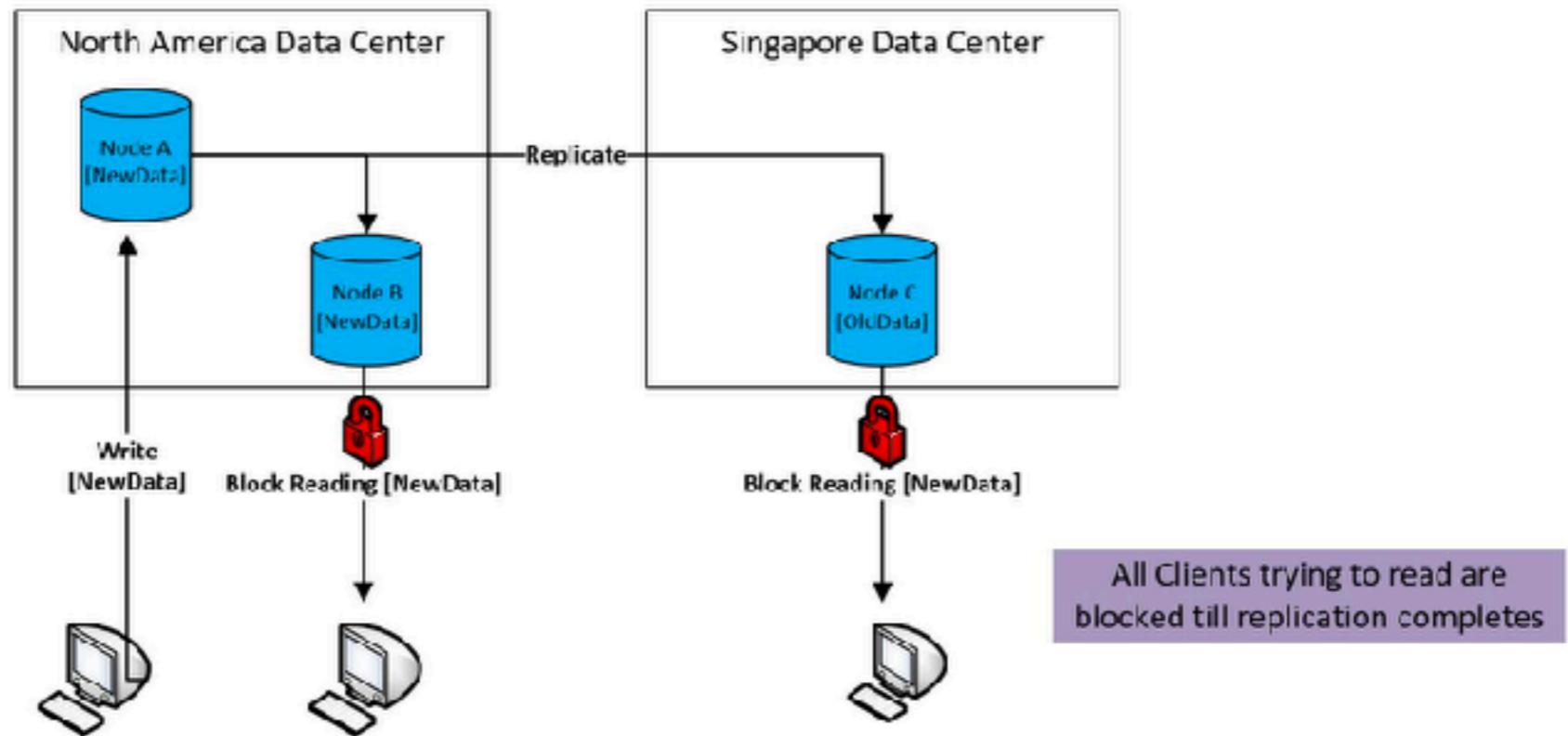
```
{  
  "id": "1",  
  "name": "John Smith",  
  "dob": "1964-30-08",  
  "holdings": [  
    { "qty": 100, "symbol": "MSFT" },  
    { "qty": 75, "symbol": "WMT" }  
  ]  
}
```

Scale-Up vs. Scale-Out



Strong Consistency vs. Eventual Consistency





COMMON COMPARISONS BETWEEN MySQL & NoSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or fields affect the design	No effect on the design with the insertion of new columns or fields

Facebook, Wikipedia,
Quora, Flickr

MySQL

Twitter

MySQL for tweets and users
their own special kind of graph database, FlockDB, built on top of MySQL
their own version of Memcached

LinkedIn

Oracle Database and Voldemort

YouTube

MySQL -> BigTable

Microsoft, Myspace

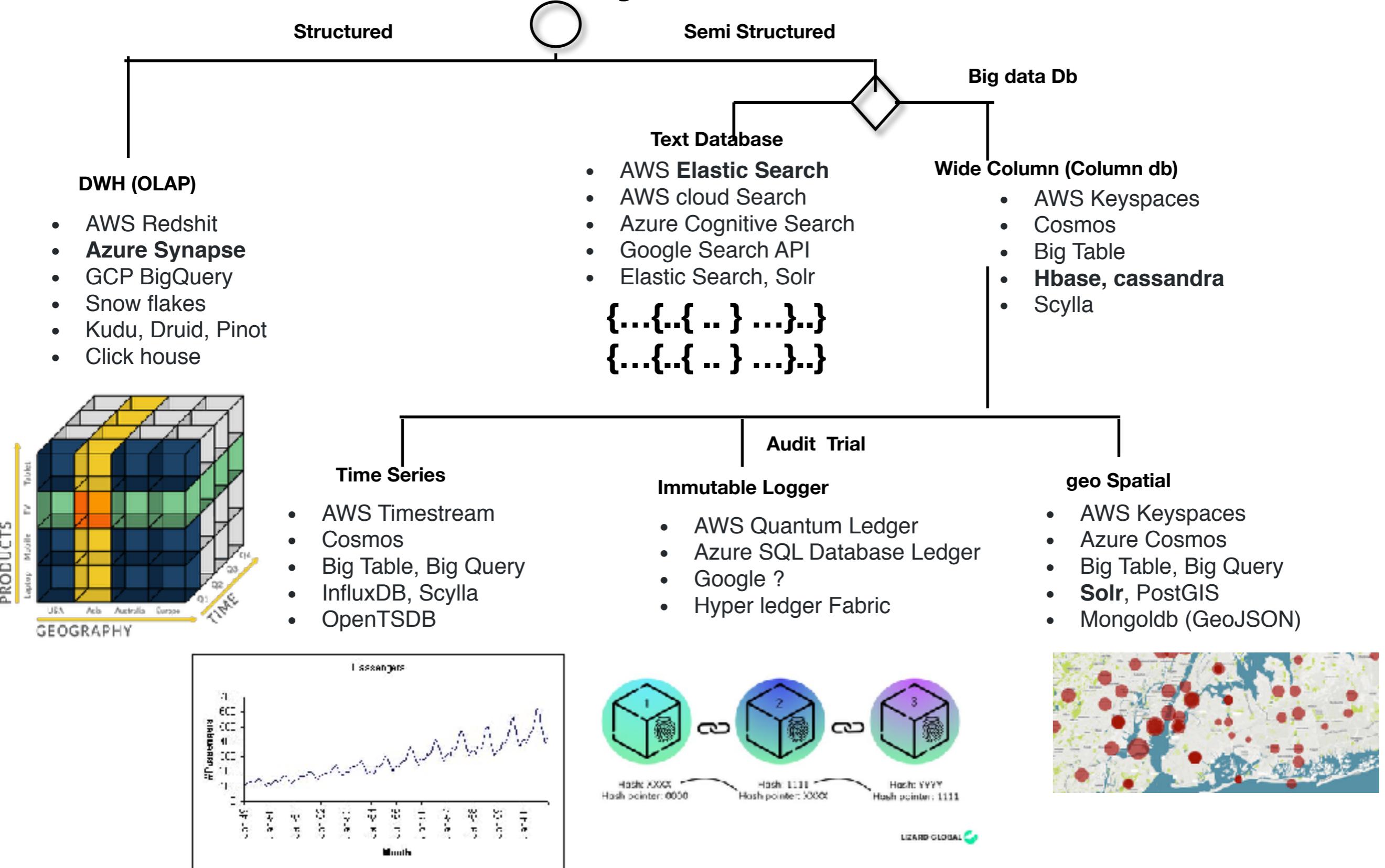
SQL Server

Yahoo

PostgreSQL

	<ul style="list-style-type: none">○ Twitter uses Redis to deliver your Twitter timeline
Key Value	<ul style="list-style-type: none">○ Pinterest uses Redis to store lists of users, followers, unfollowers, boards, and more○ Coinbase uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions
Graph	<ul style="list-style-type: none">○ Quora uses Memcached to cache results from slower, persistent databases○ Walmart uses Neo4j to provide customers personalized, relevant product recommendations and promotions
Document	<ul style="list-style-type: none">○ Medium uses Neo4j to build their social graph to enhance content personalization○ Cisco uses Neo4j to mine customer support cases to anticipate bugs
	<ul style="list-style-type: none">○ SEGA uses MongoDB for handling 11 million in-game accounts○ Cisco moved its VSRM (video session and research manager) platform to Couchbase to achieve greater scalability
	<ul style="list-style-type: none">○ Aer Lingus uses MongoDB with Studio 3T to handle ticketing and internal apps○ Built on MongoDB, The Weather Channel's iOS and Android apps deliver weather alerts to 40 million users in real-time

Analytical



Classic Relational Databases

Name	Age	Nickname	Employee
Gianfranco Quilizzoni Founder & CEO	40	Heldi	<input checked="" type="radio"/>
Marco Botton Tuttofare	38		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	41	Potato	<input type="checkbox"/>
Valerie Liberty Head Chef	16	Val	<input checked="" type="checkbox"/>

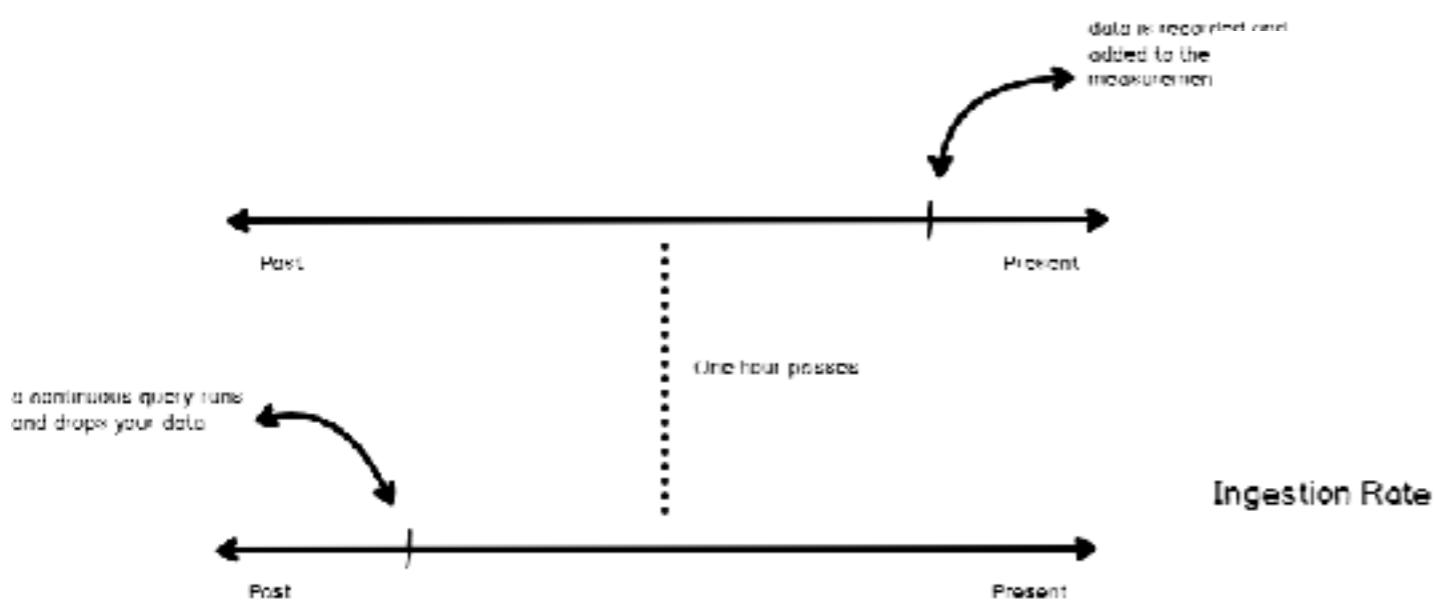
Data are multidimensional

Time Series Databases

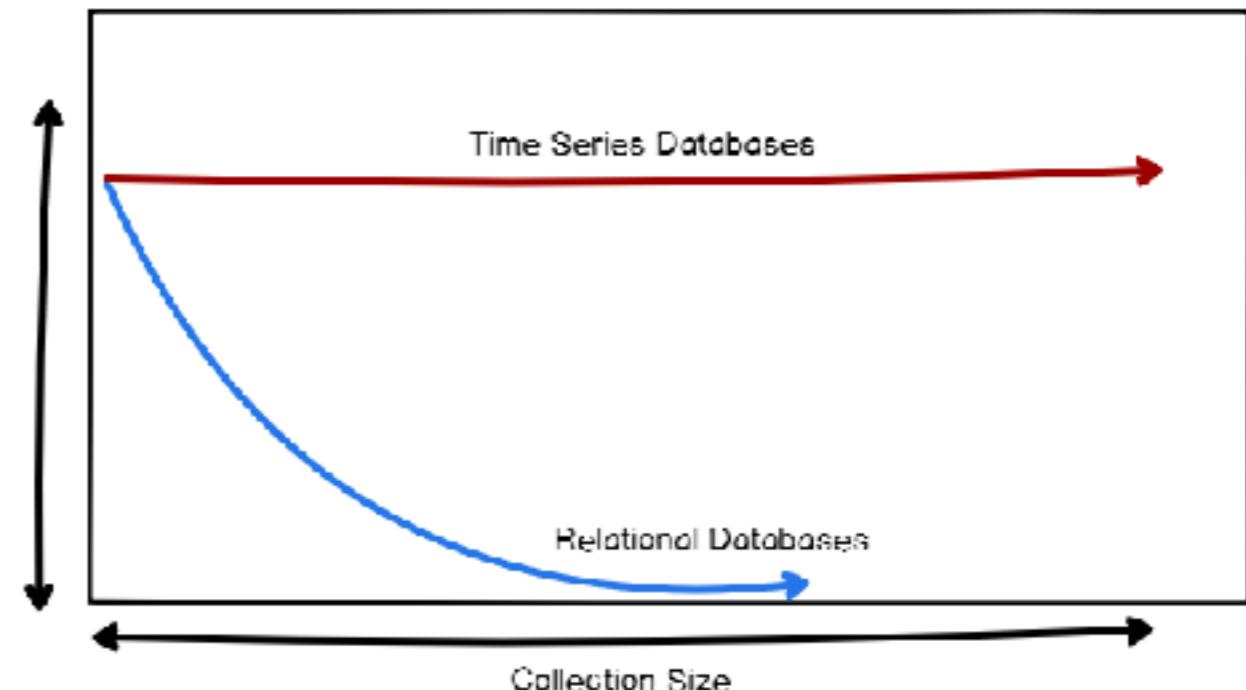
Sensor Temperature	Time
39.6	12/01/19 @ 11:12
11.2	12/01/19 @ 11:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

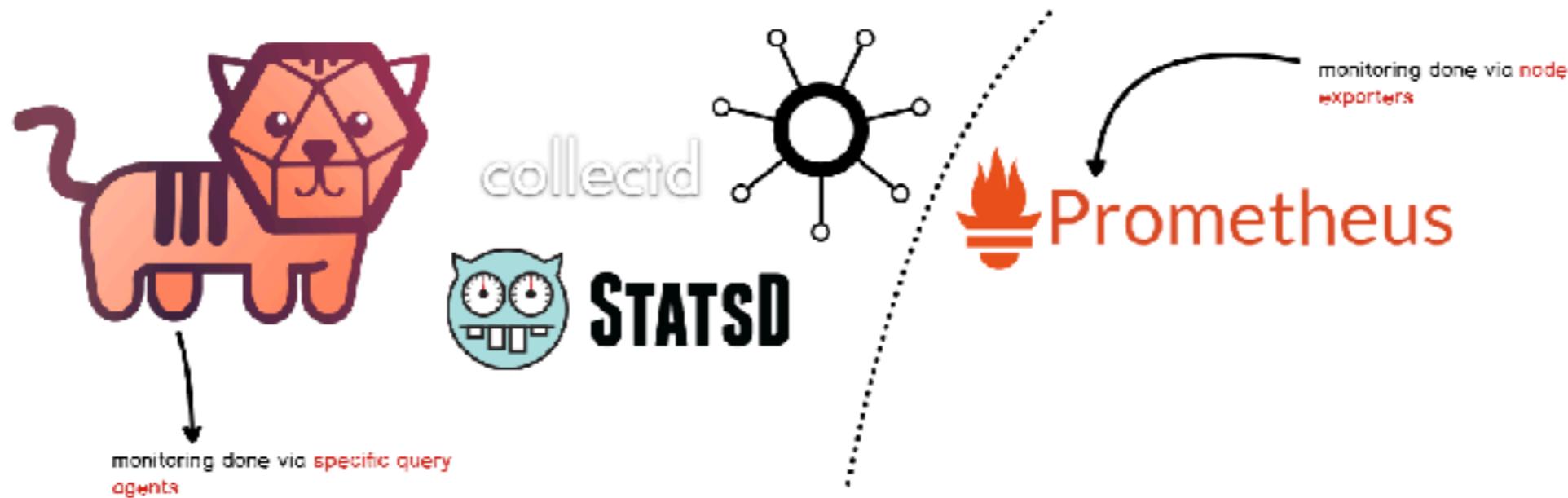
Case : retention policy = 1 hour



DBMS & TSDB Difference



Tools that 'produce' TSDB data

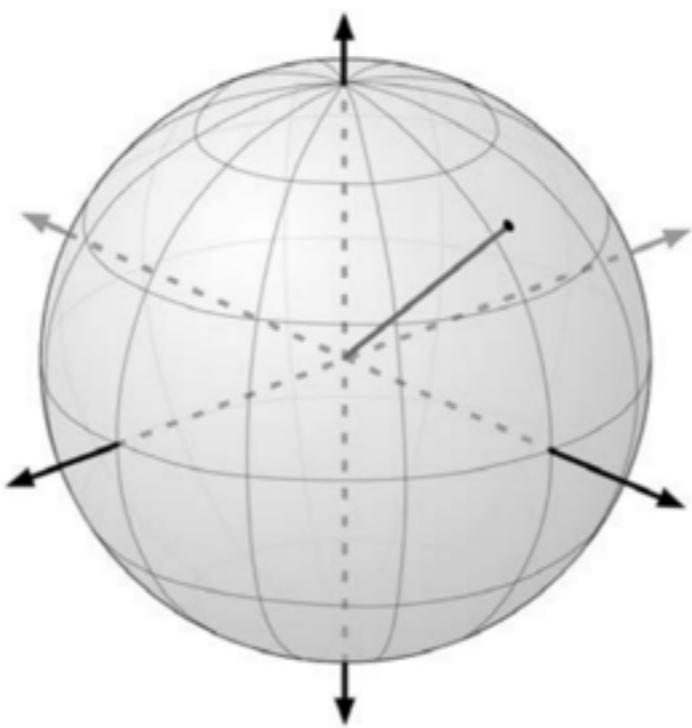


Tools that 'consume' TSDB



*Non-exhaustive list

Car ID	Departure			Arrival		
	Date-Time	Latitude	Longitude	Date-Time	Latitude	Longitude
...
...
...



Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

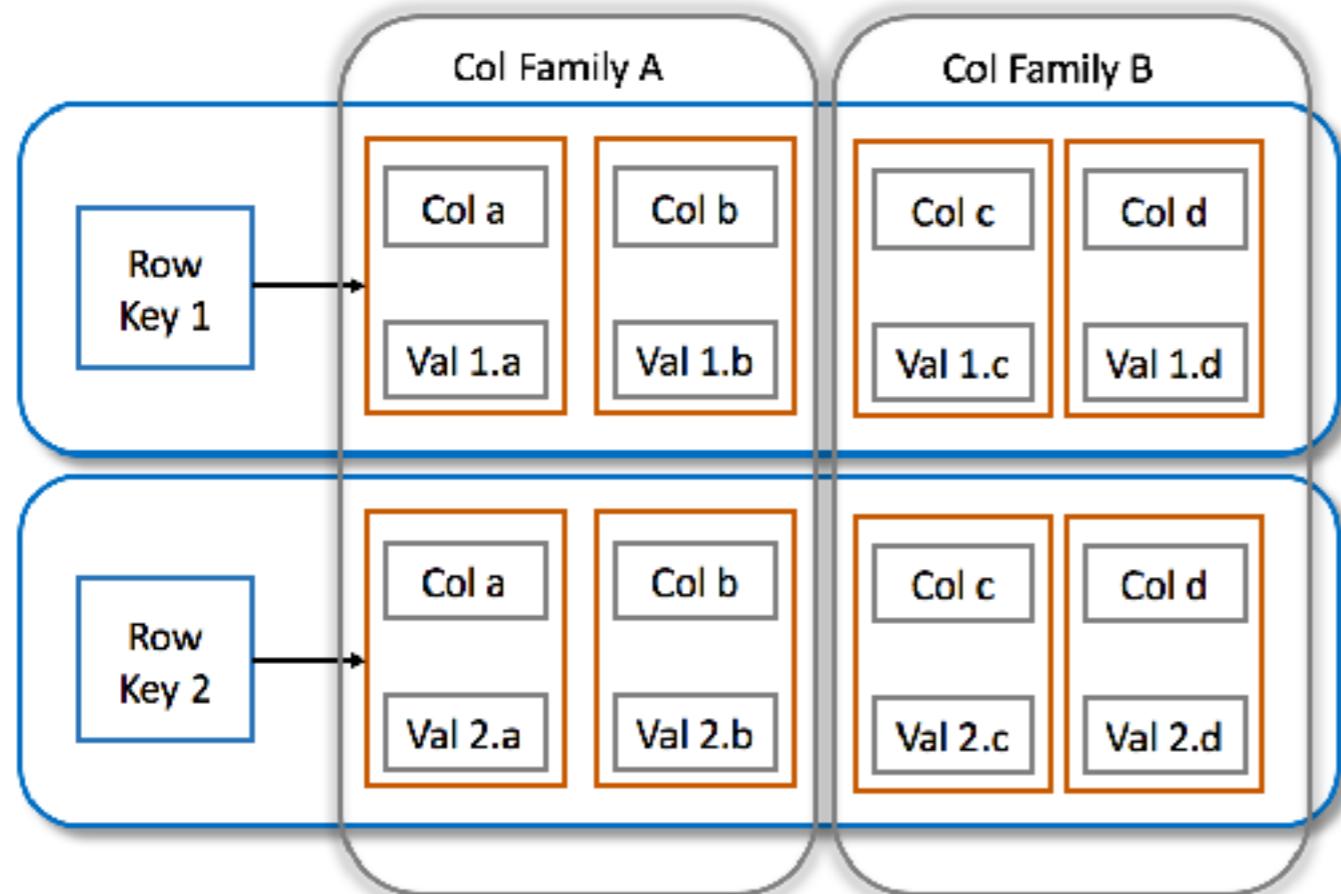
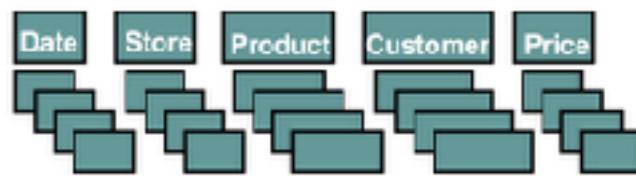
Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

row-store

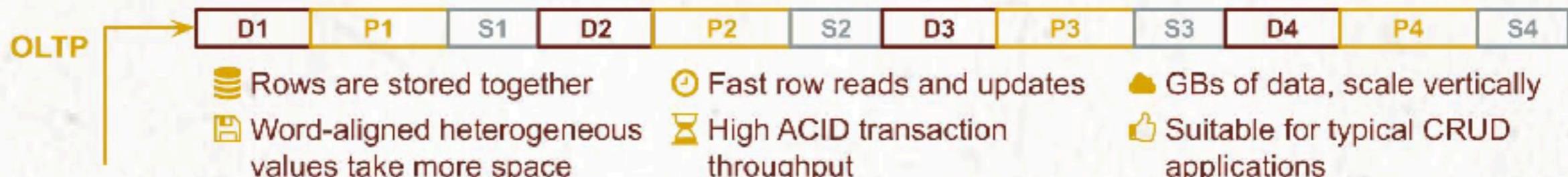


column-store



RDBMS vs. Columnar: OLTP vs. OLAP

scgupta.link/datastores 

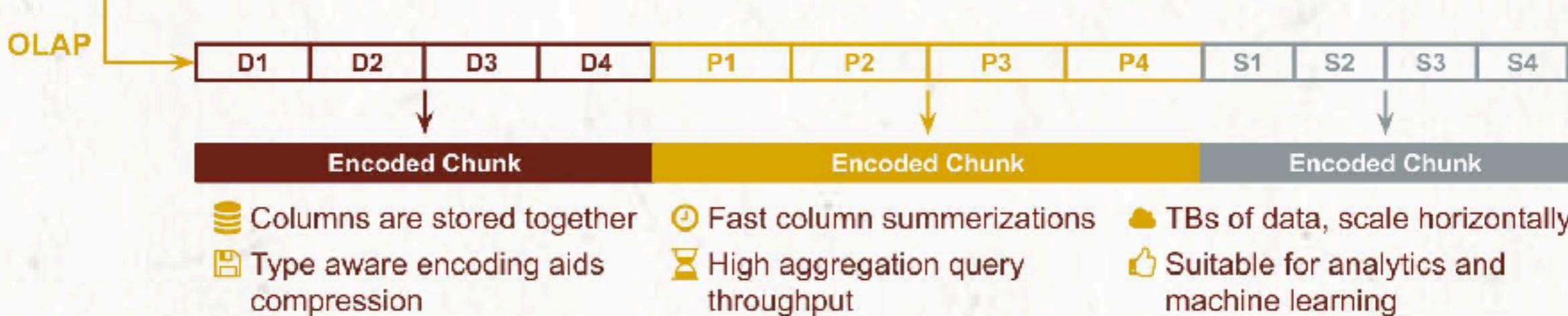


Logical Table

Date	Product	Sale
D1	P1	S1
D2	P2	S2
D3	P3	S3
D4	P4	S4

RDBMS: Row-Oriented Datastore

Columnar: Column-Oriented Datastore



DATABASE VERSUS DATA WAREHOUSE

DATABASE

An organized collection of related data which stores data in a tabular format

Contains detailed data

Uses Online Transactional Processing (OLTP)

Helps to perform fundamental operations of a business

Less fast and less accurate

Application oriented

Tables and joins are complex because they are normalized

Design is helped by entity relationship modelling

DATA WAREHOUSE

A central location which stores consolidated data from multiple databases

Contains summarized data

Uses Online Analytical Processing (OLAP)

Helps to analyze the business

Faster and accurate

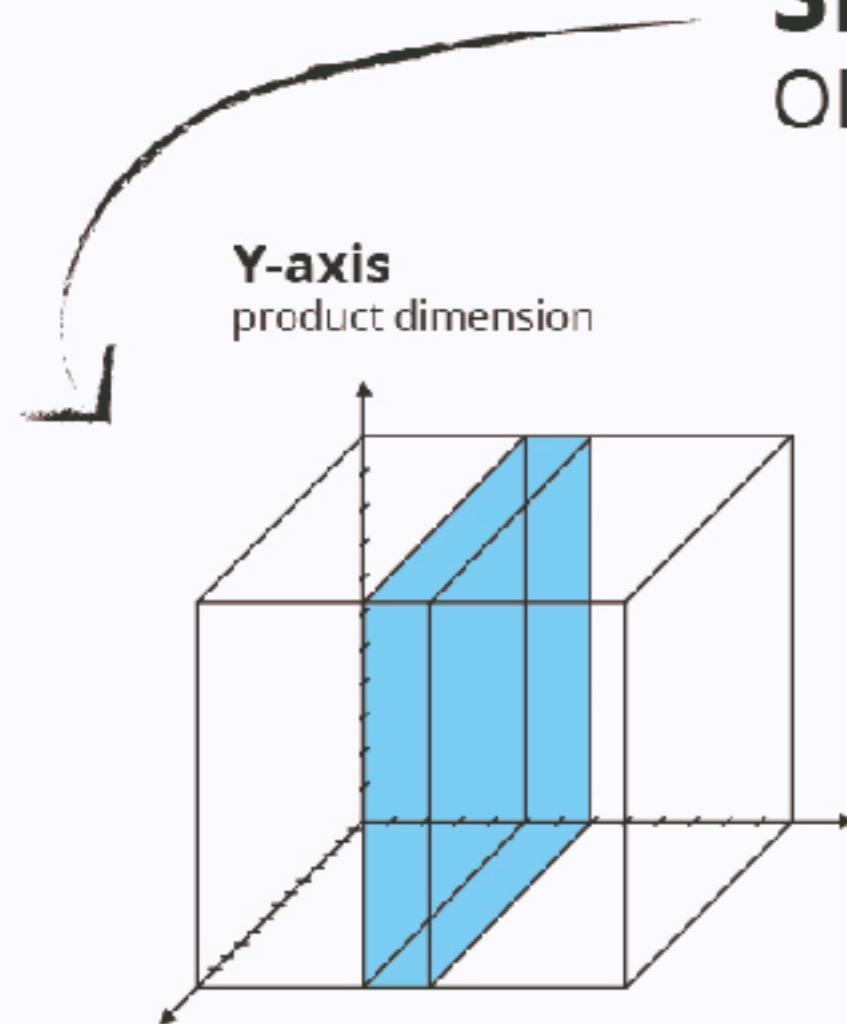
Subject oriented

Tables and joins are simple because they are denormalized

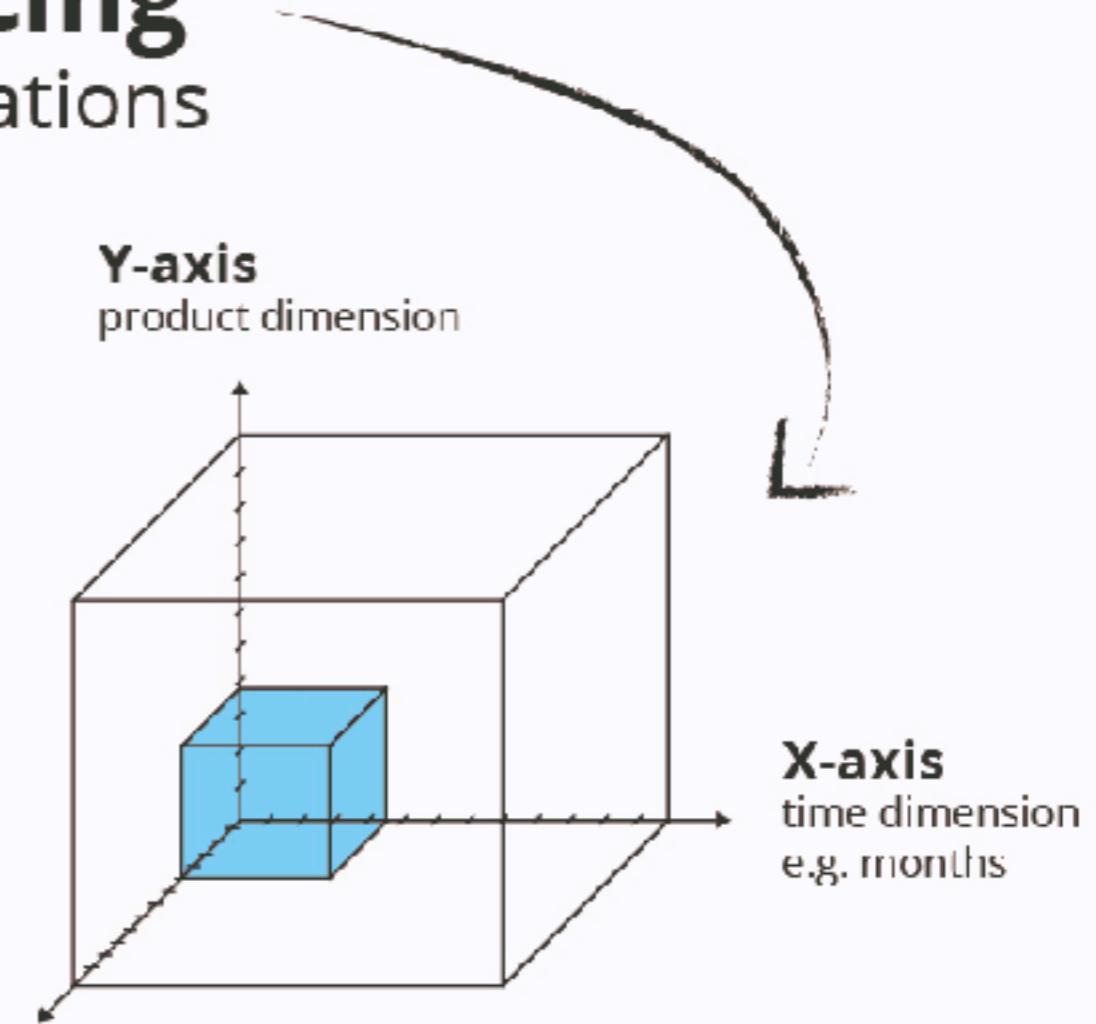
Design is helped by data modelling technique

Slicing & Dicing

OLAP cube operations



Z-axis
customer dimension



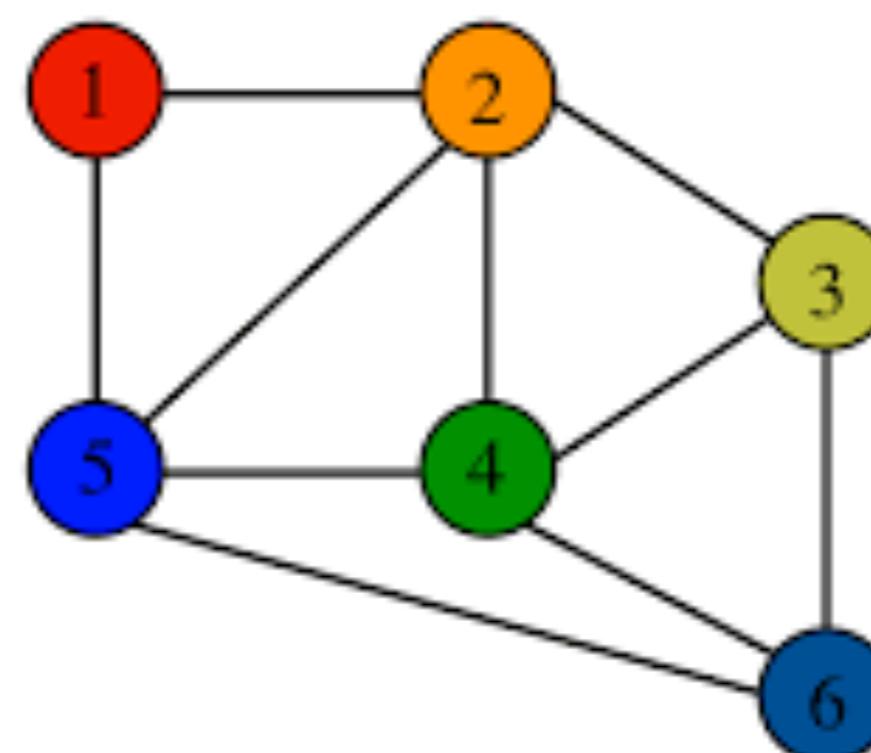
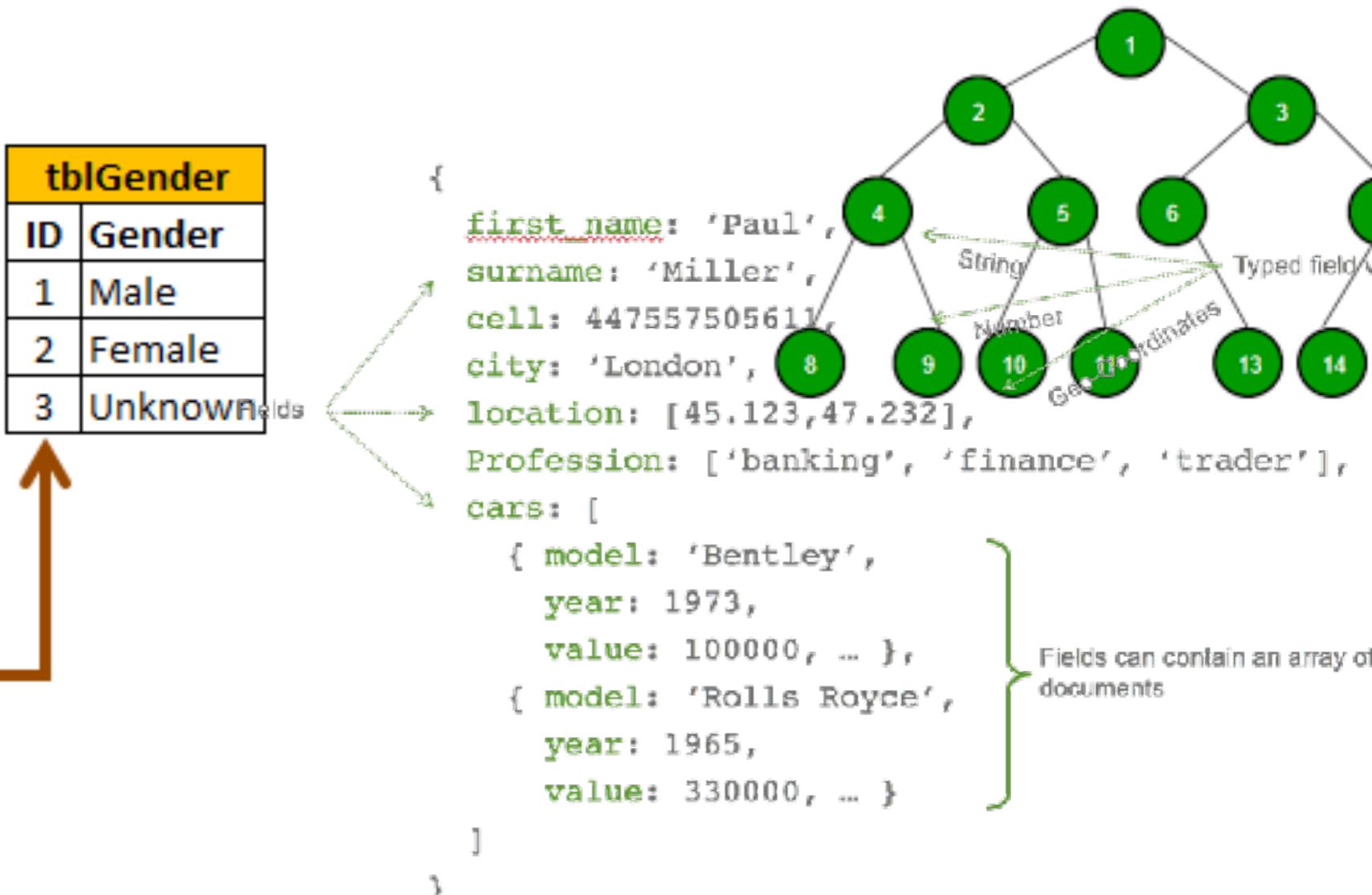
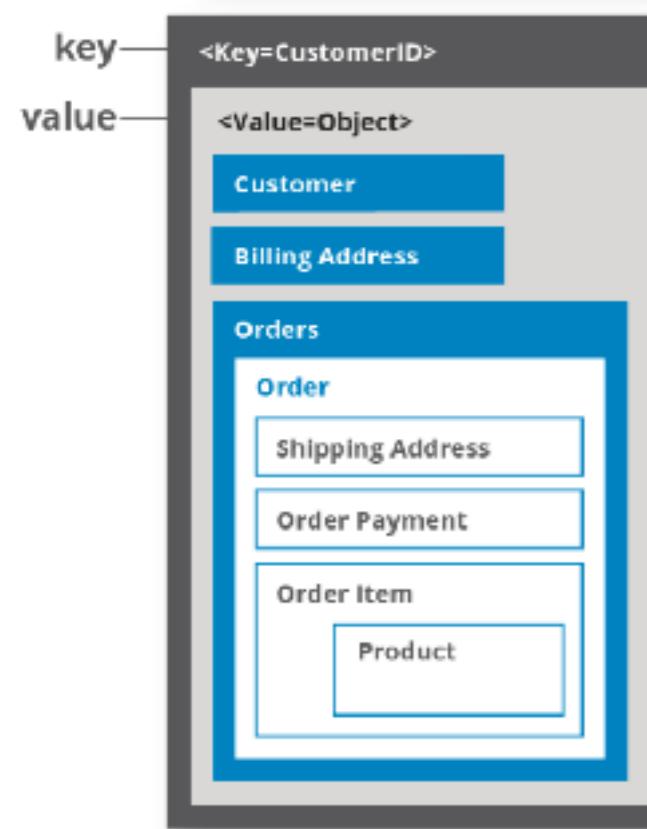
Z-axis
customer dimension

Rdbms (Referential Integrity)

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

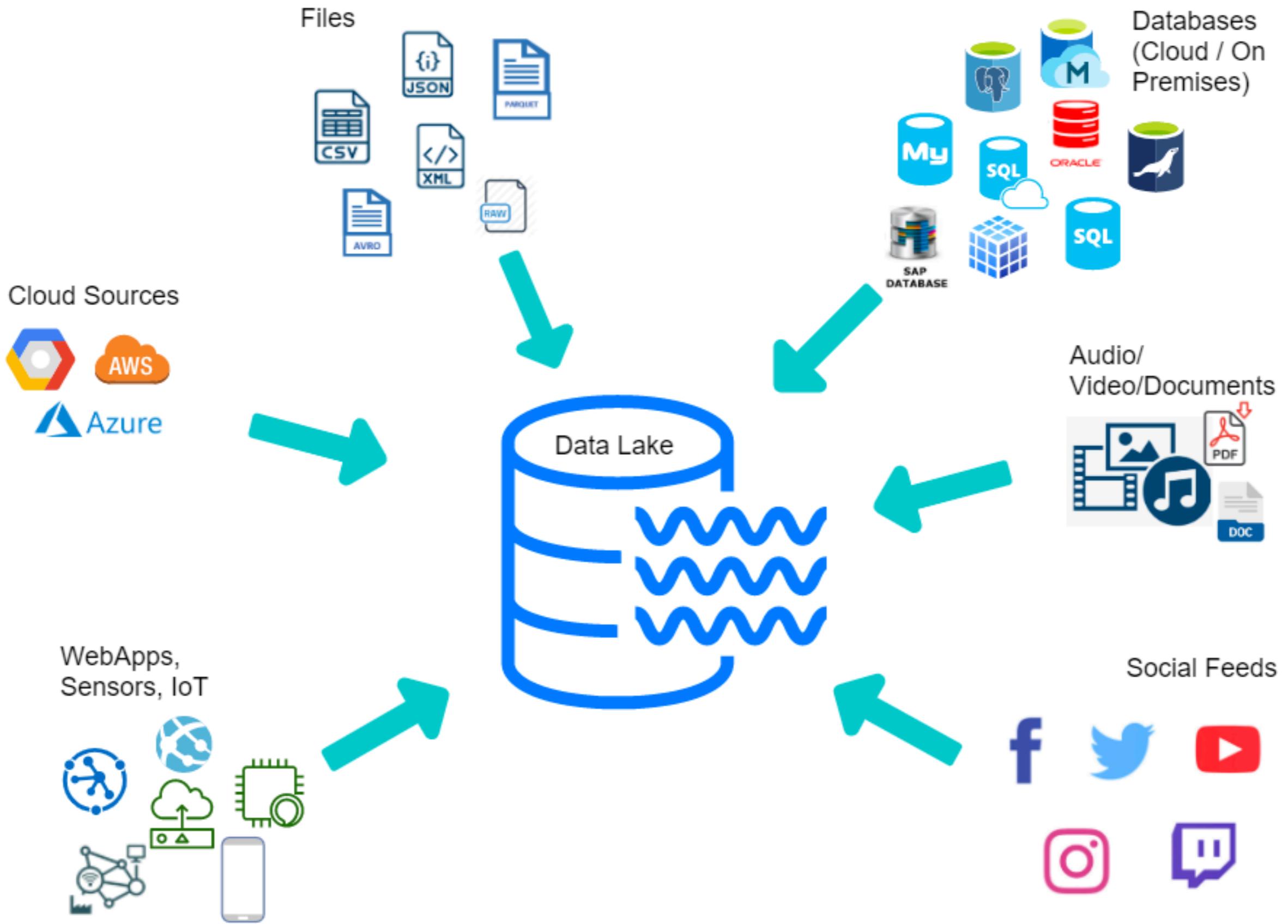
tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

key	value
123	123 Main St.
126	(805) 477-3900



Wide Column

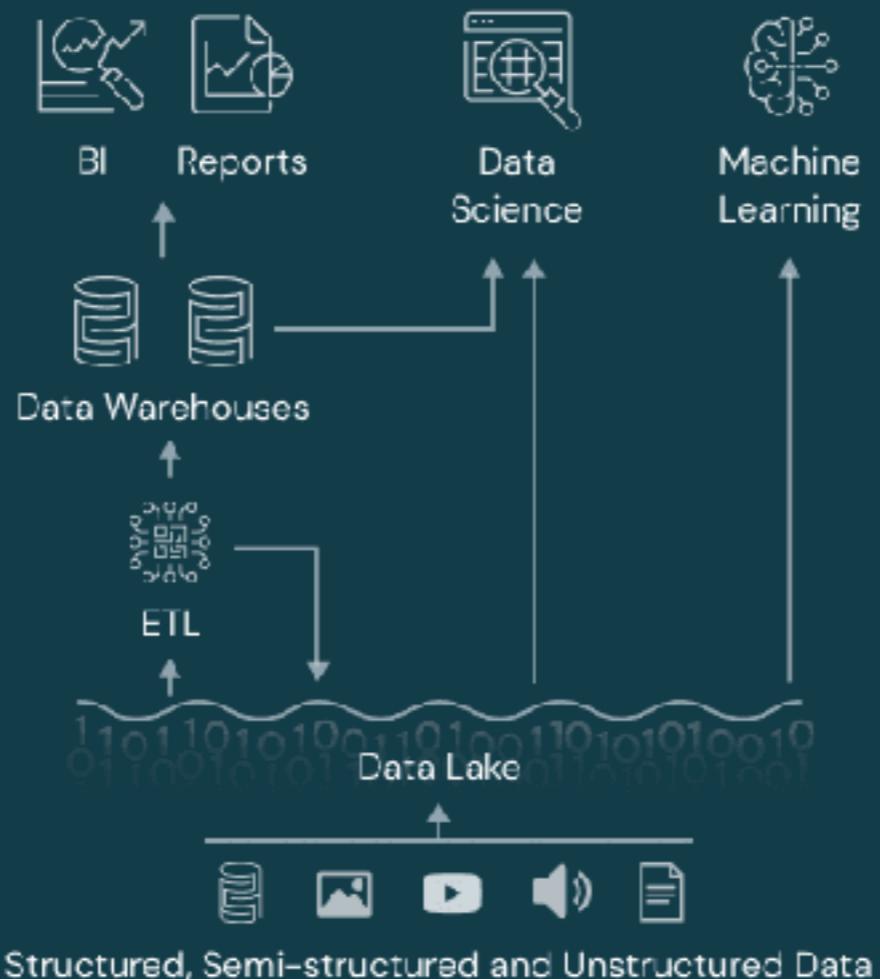
- [Spotify](#) uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- [Facebook](#) initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- [Outbrain](#) uses Cassandra to serve over 190 billion personalized content recommendations each month



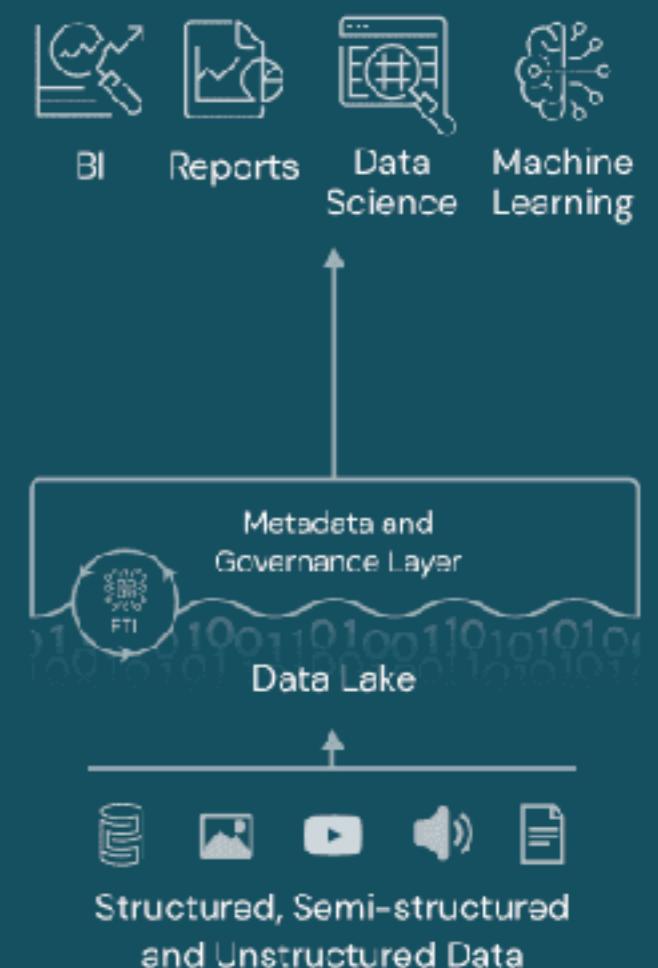
Data Warehouse

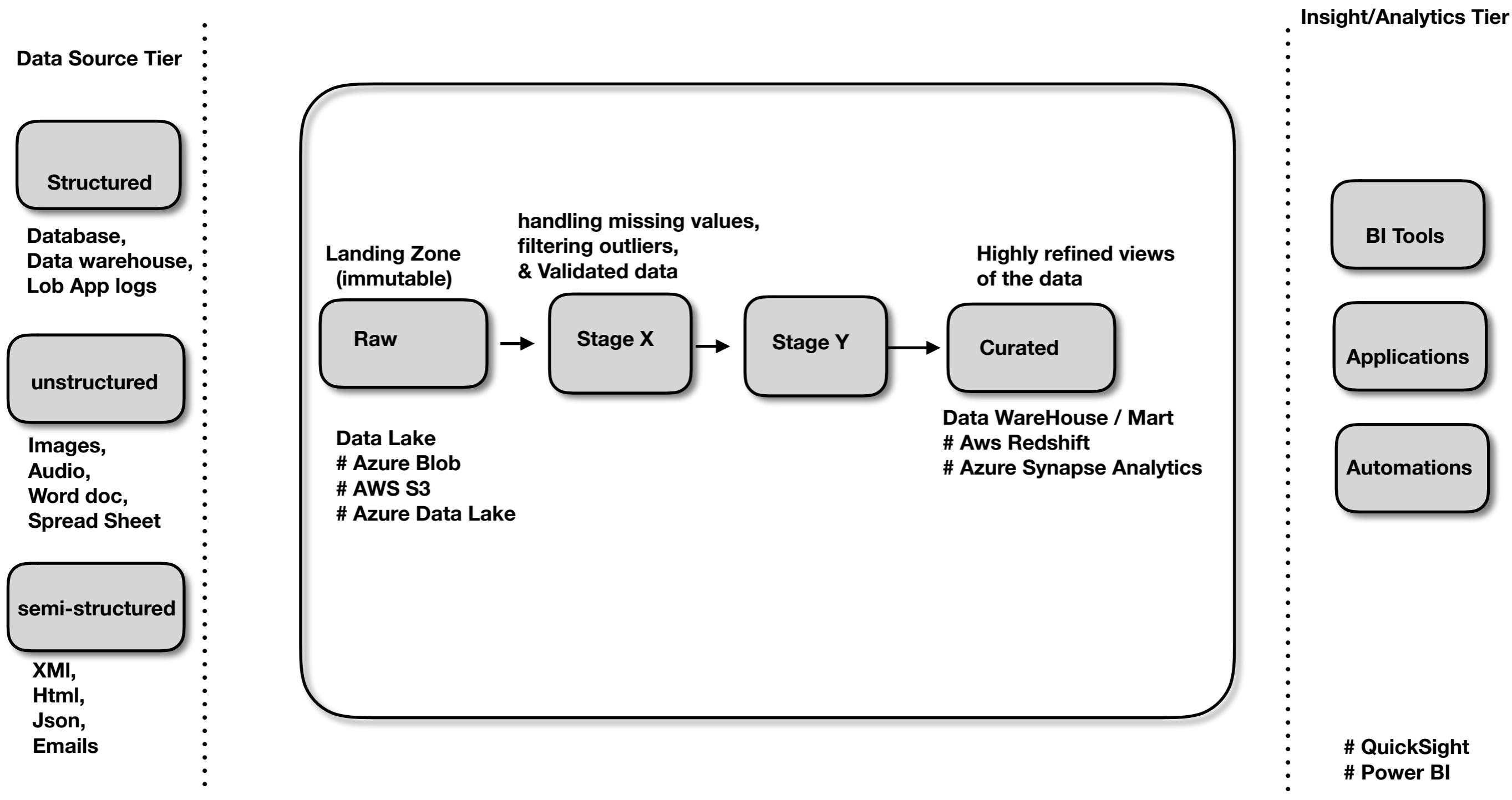


Data Lake

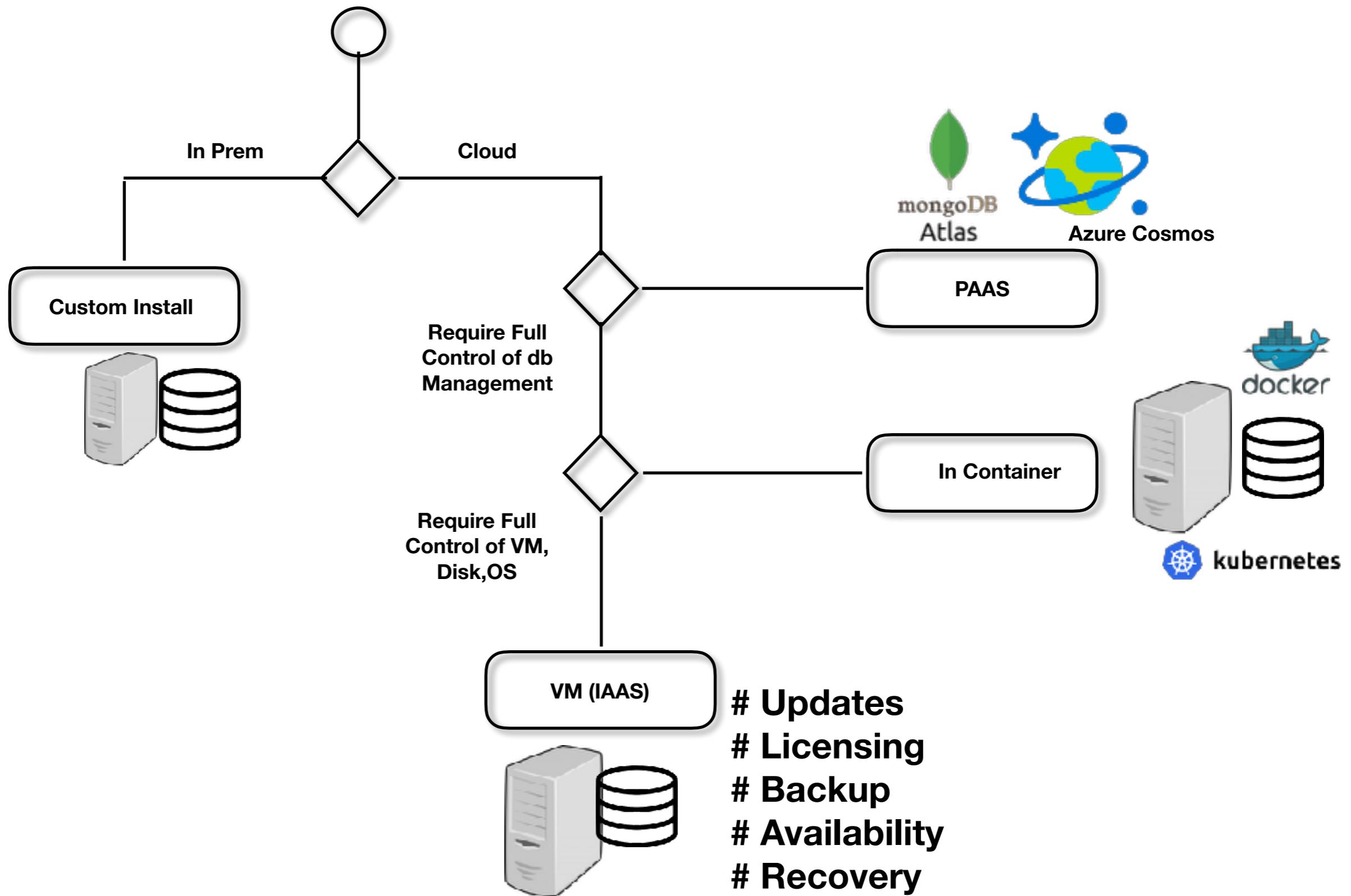


Data Lakehouse

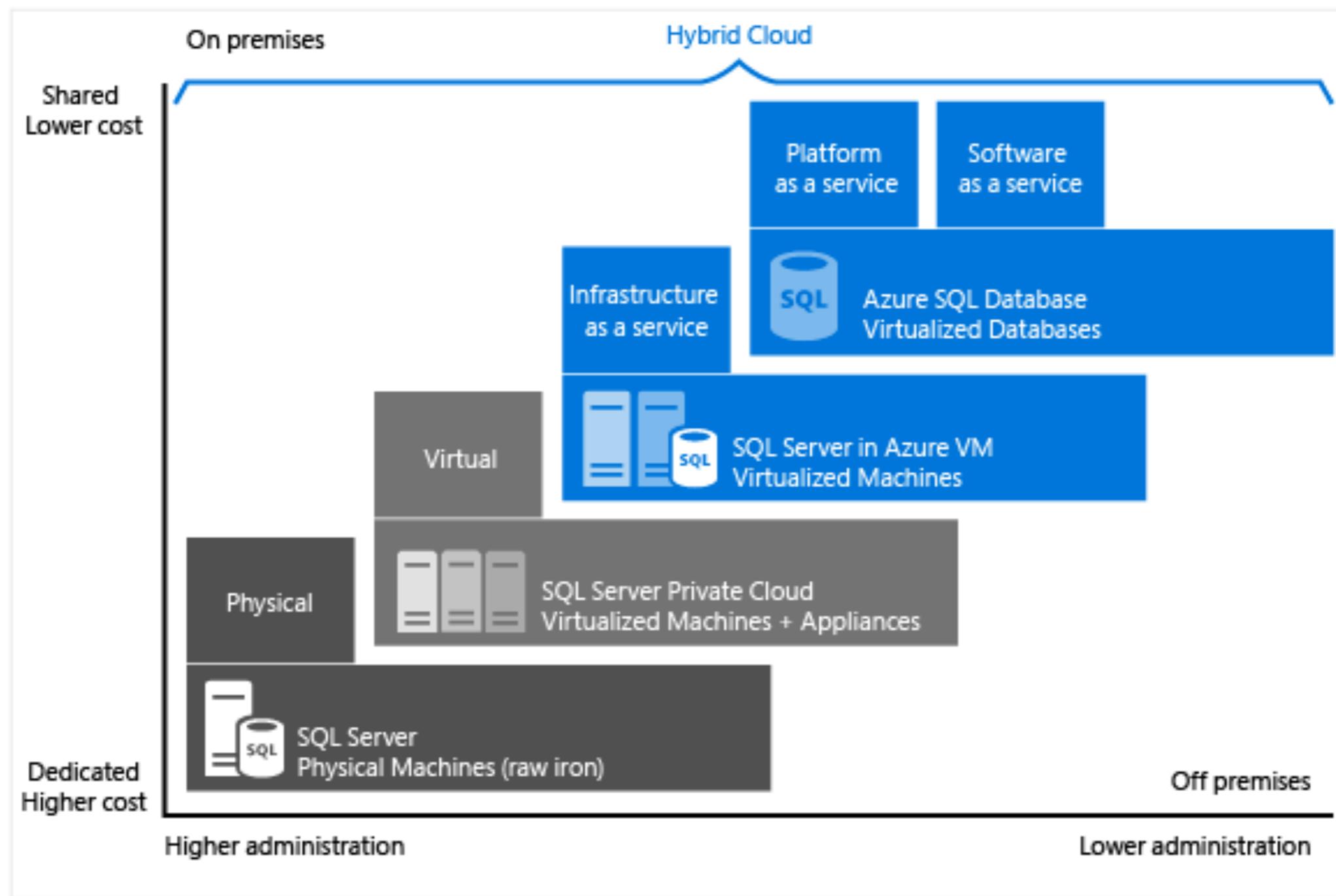


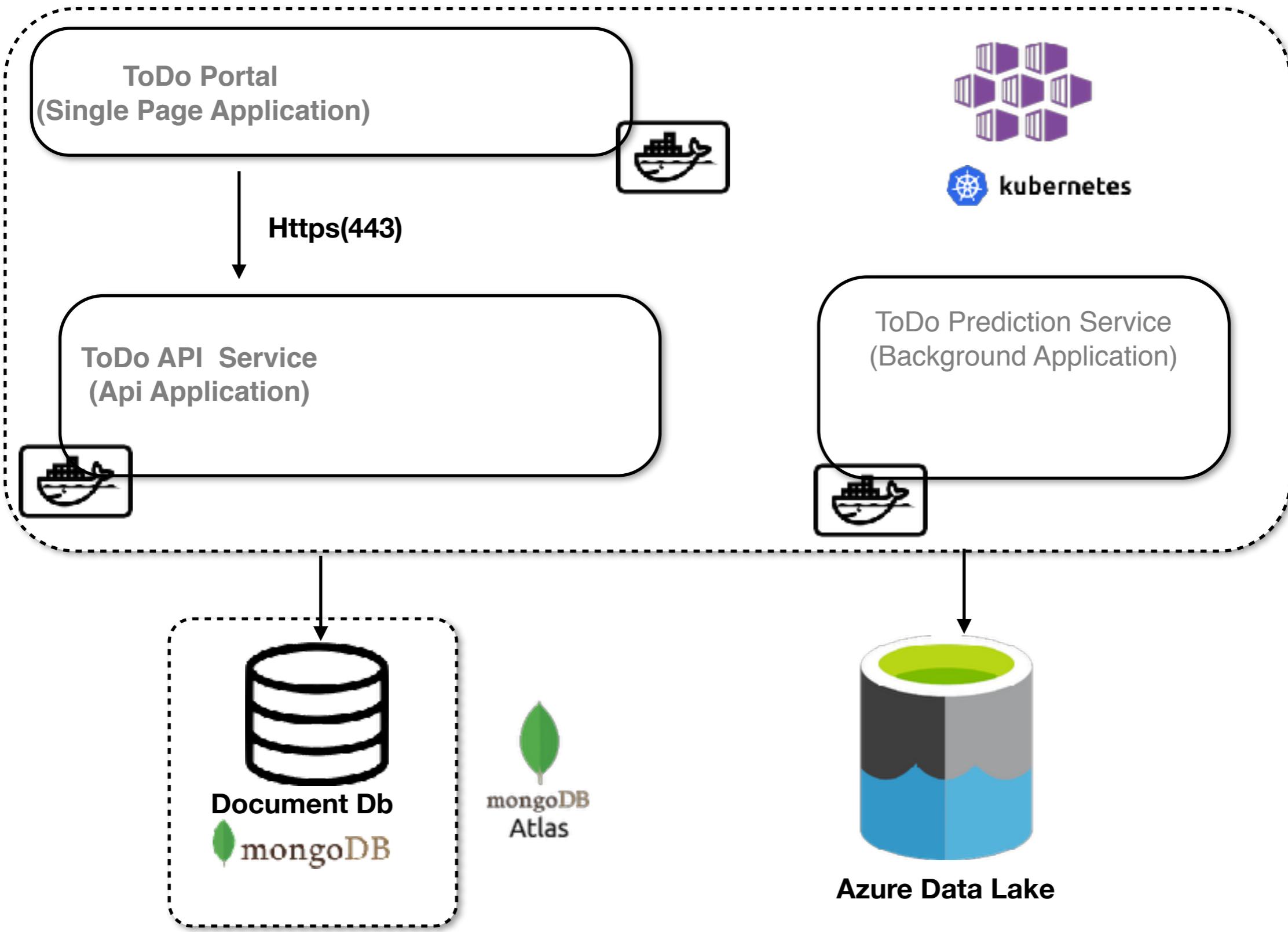


Deploying Data Tier

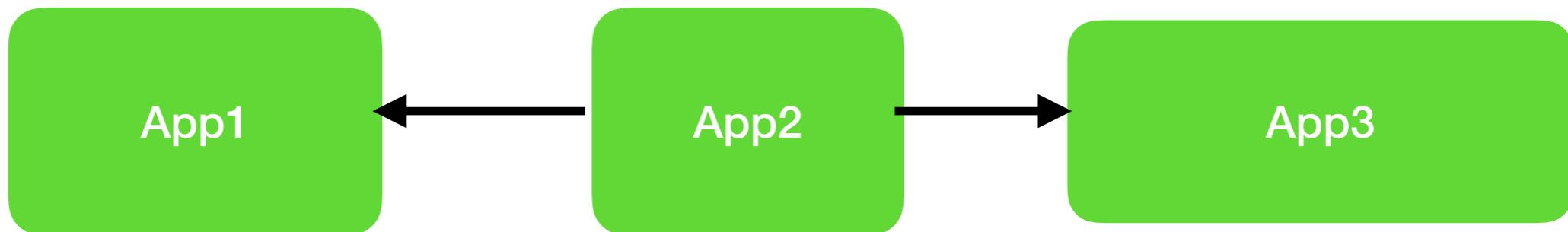


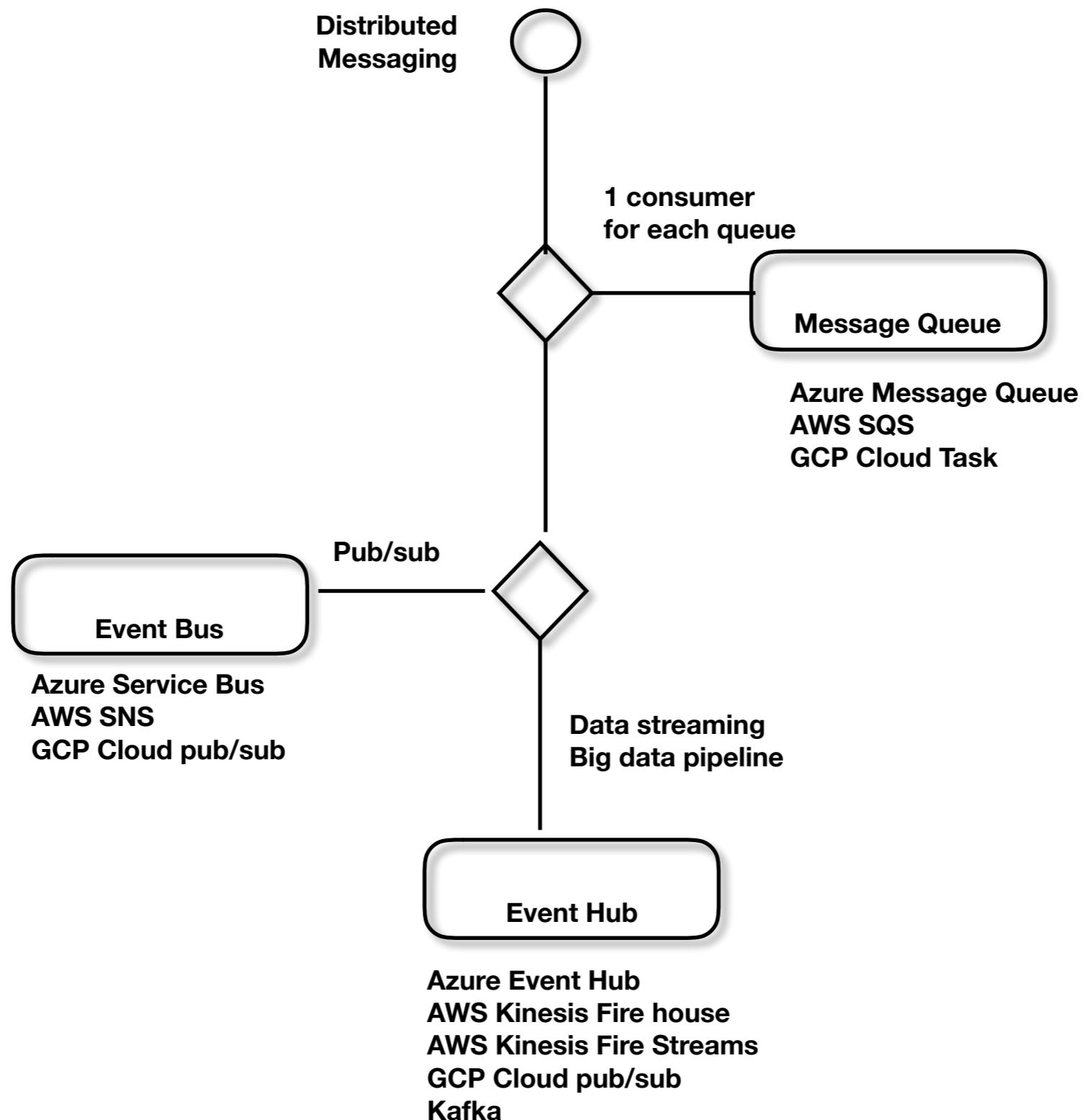
Deploying Data Tier

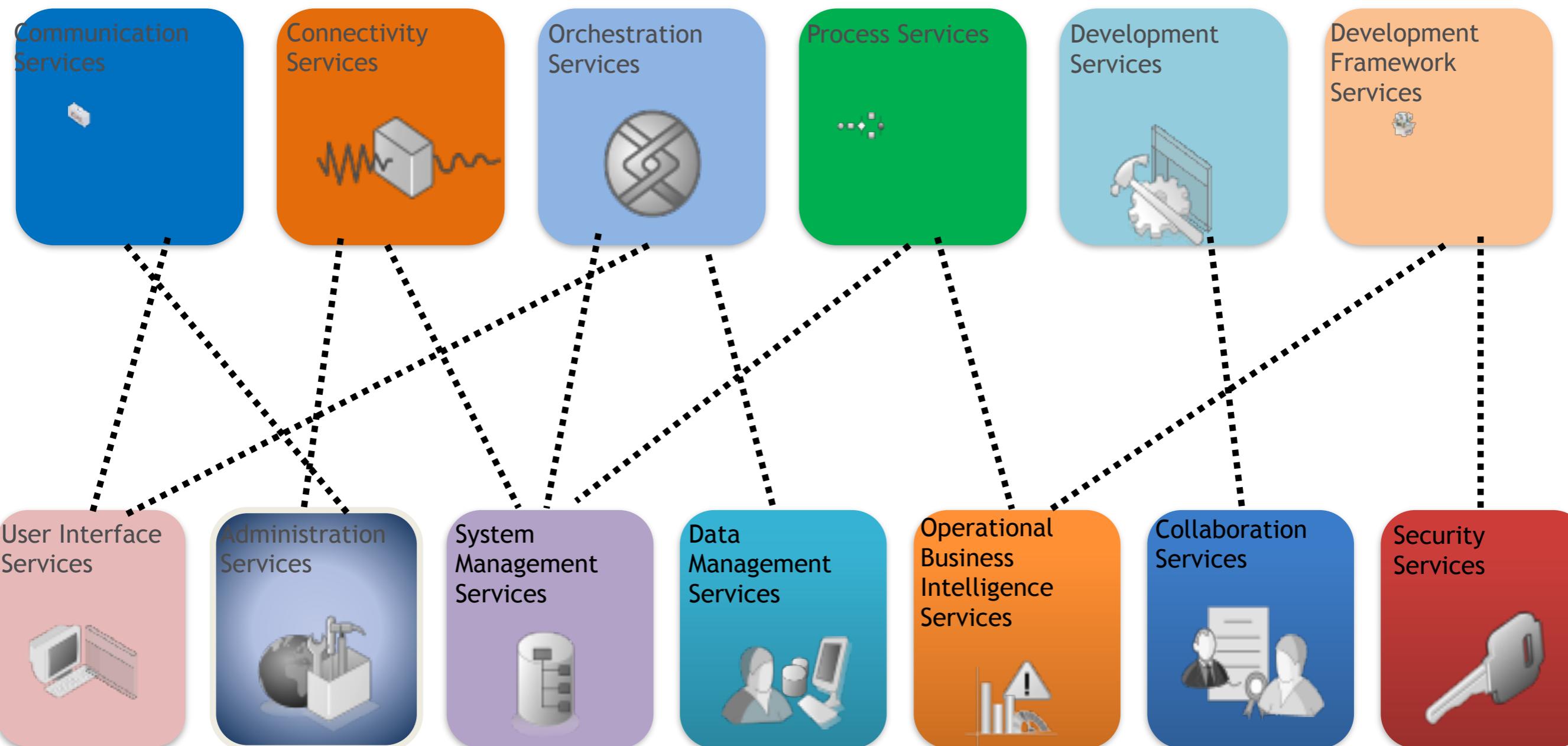


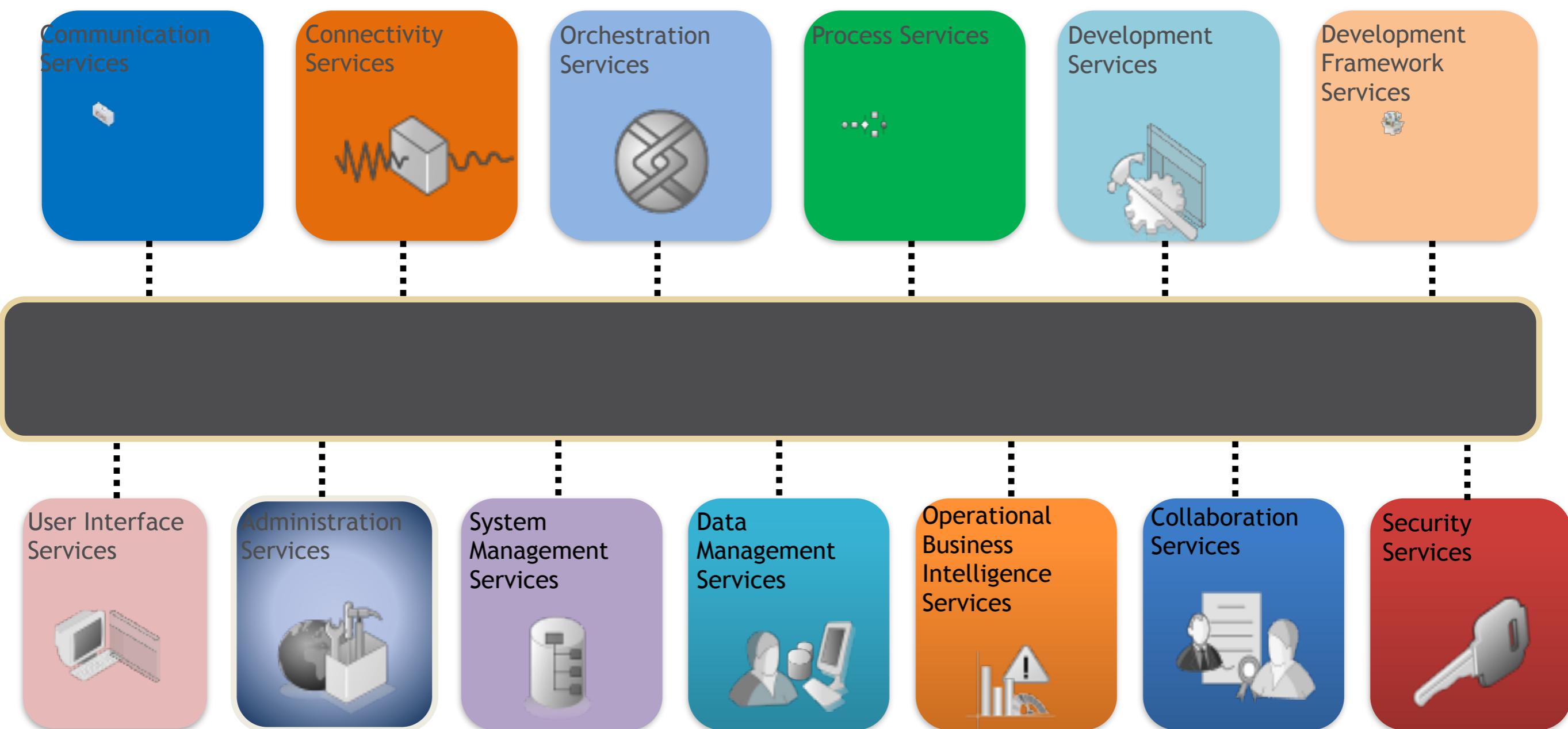


System Integration

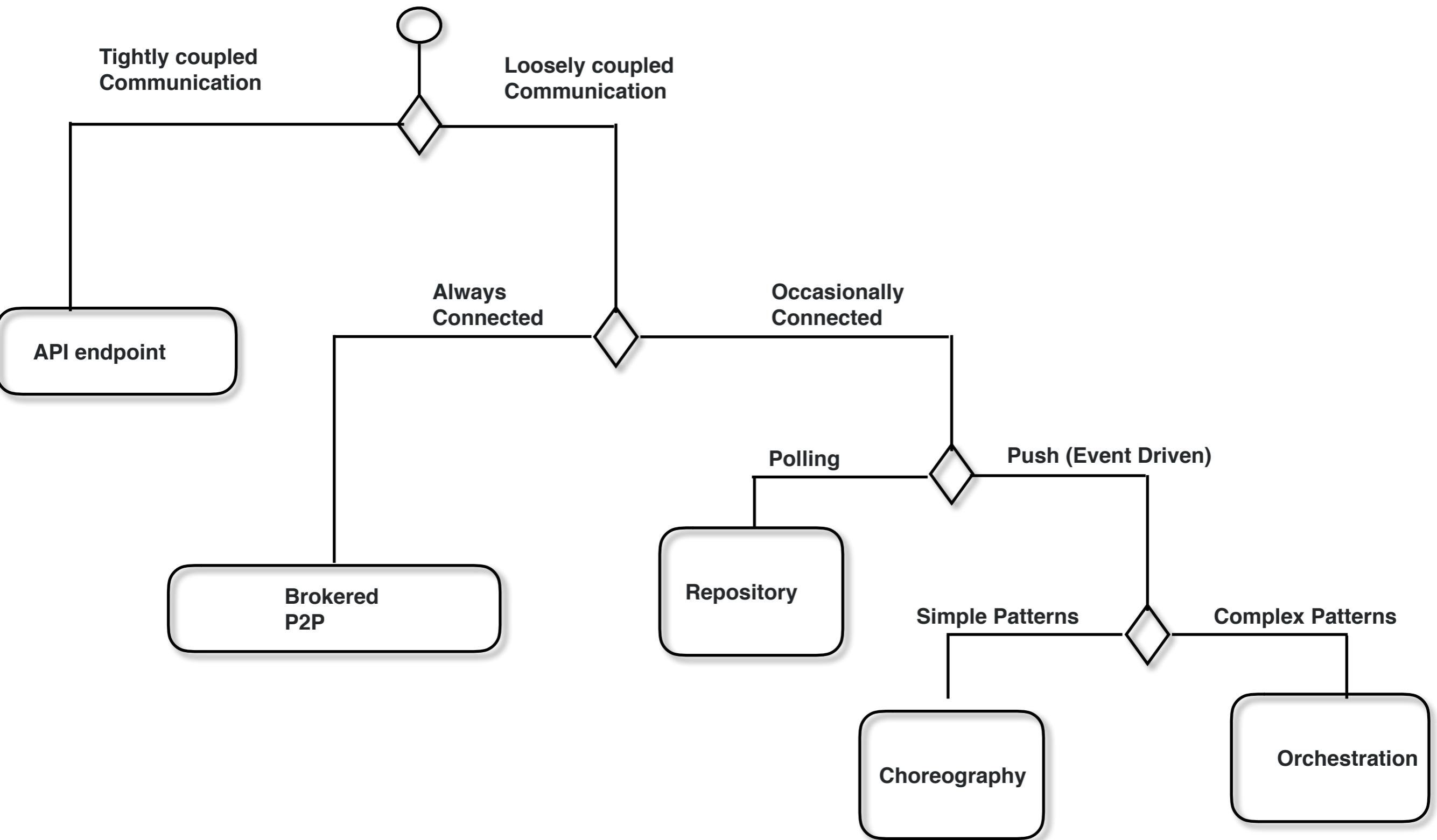




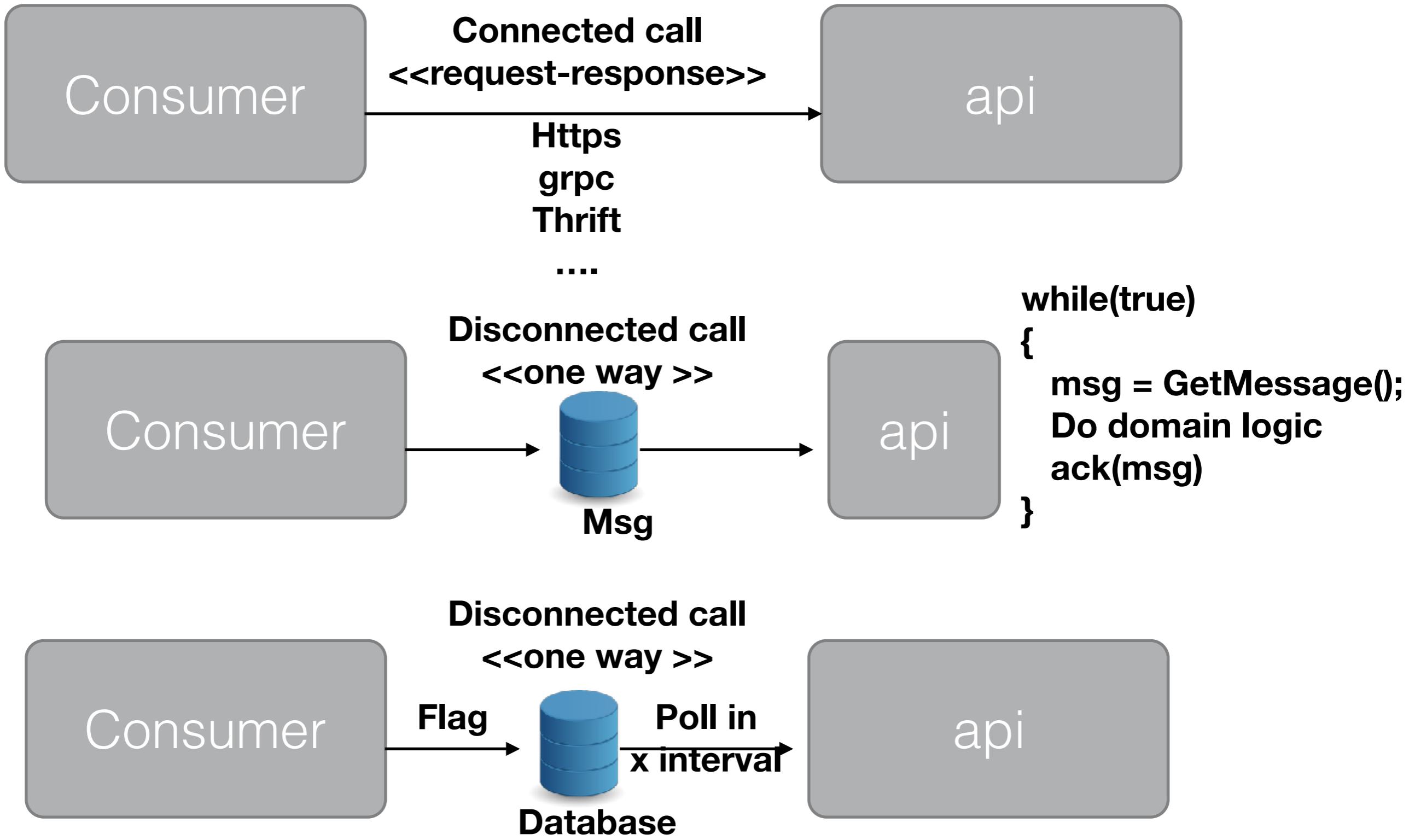




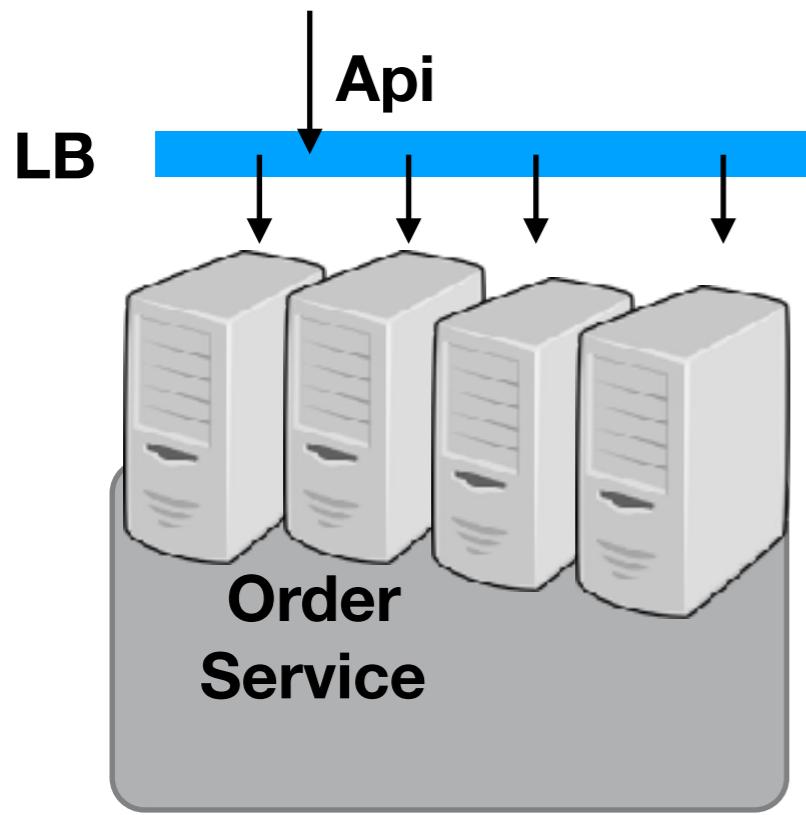
Choose Communication Mechanism



Logical view Communication



API call



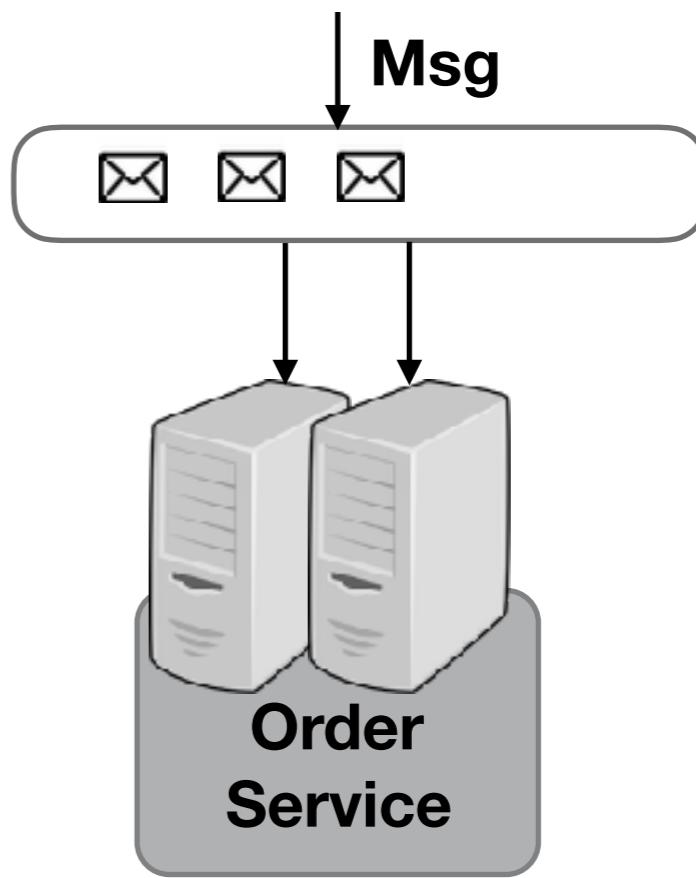
immediate Consistency

Less Scalable
no of server's depends on peak load

less reliable
Will not remember last execution context
After recovery during a crash

loss of data
Because of throttle

Messaging



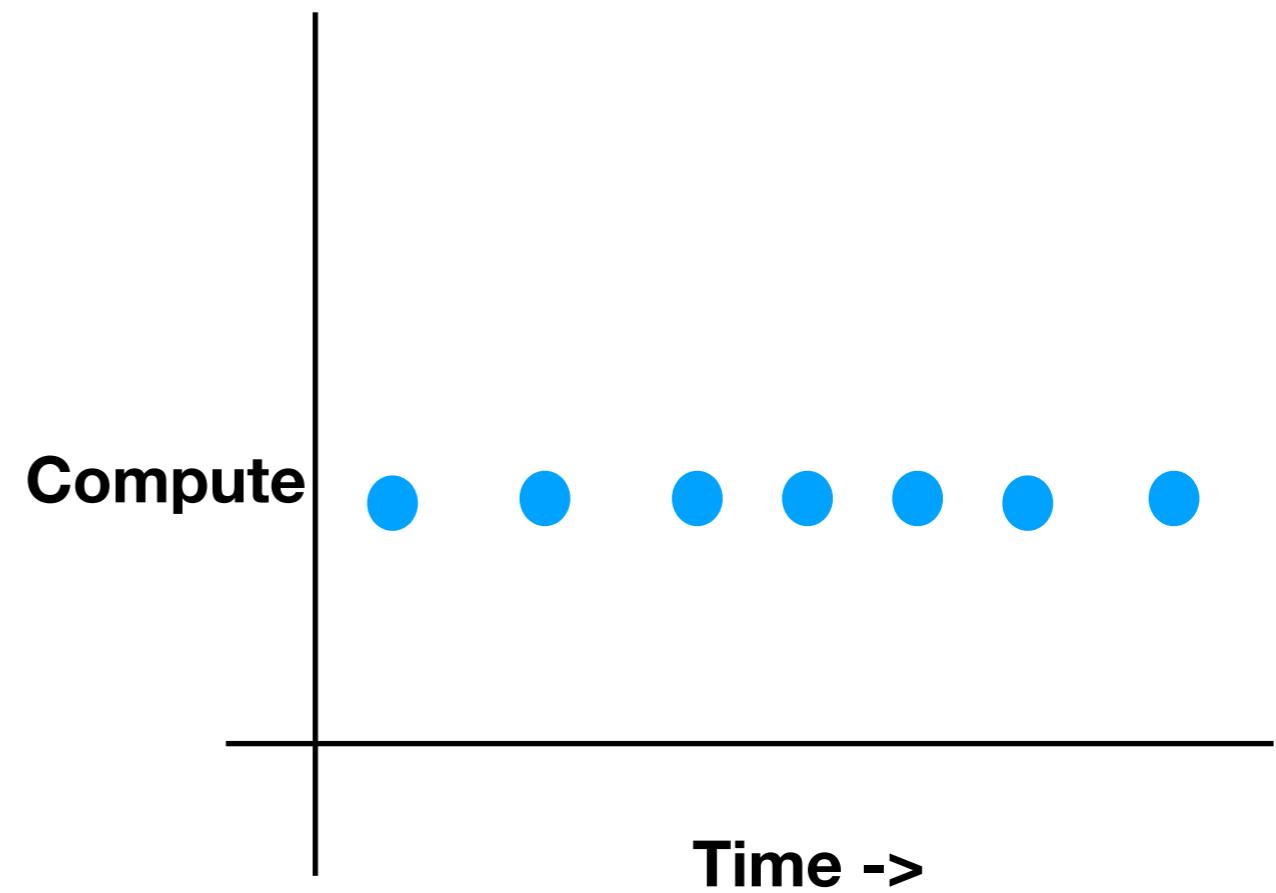
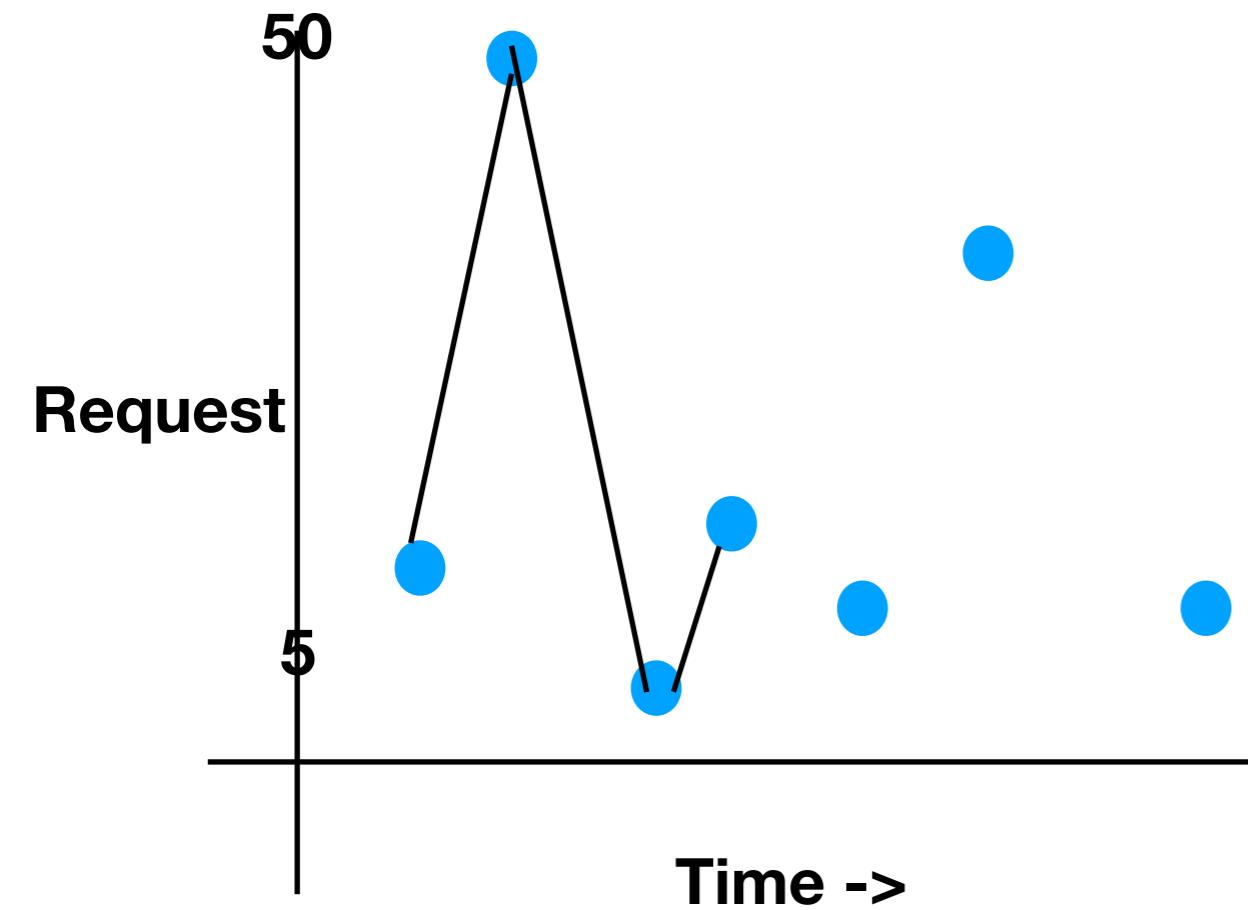
Eventual Consistency

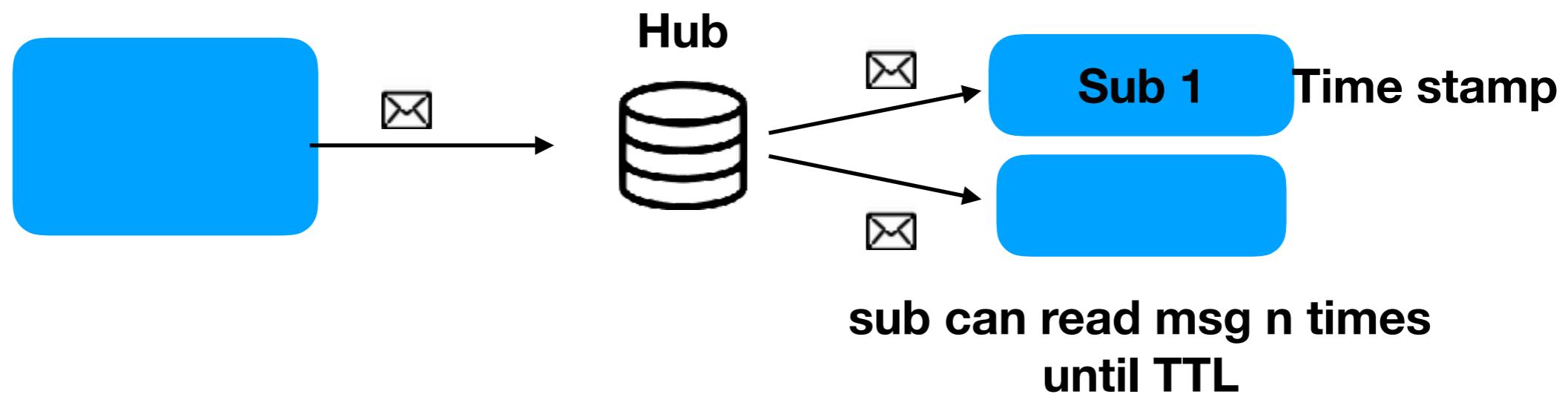
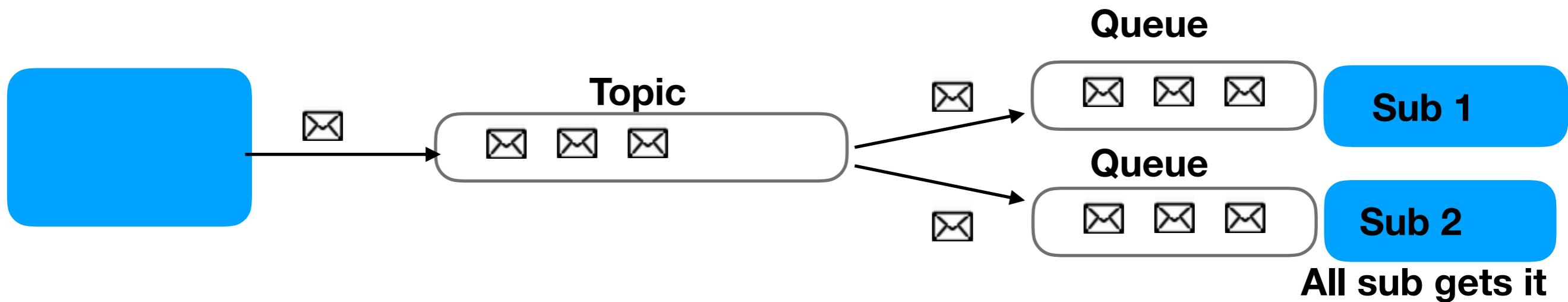
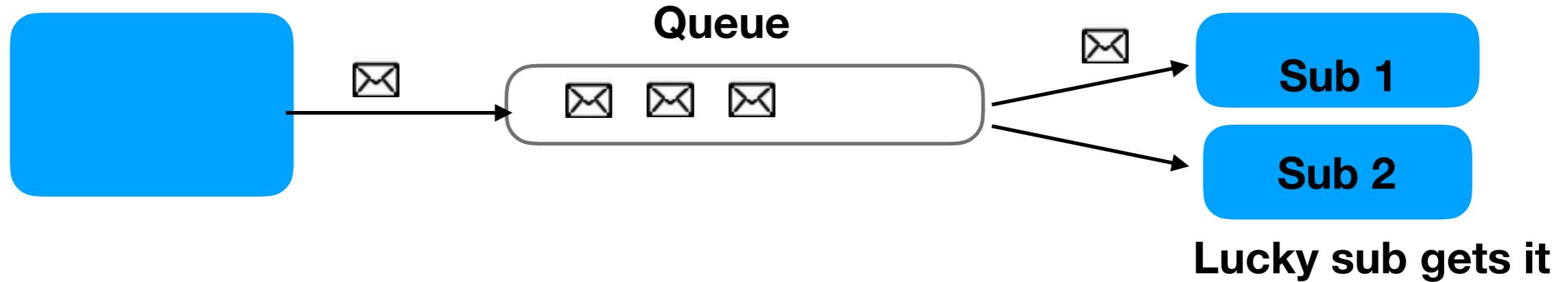
more scalable
no of server's depends on Avg load
(Load Leveling)

more reliable
If message are not ACK, messages reappear in Queue

no loss of data
Gets queued

Load Leveling





Ui portal

CRUD

**Inventory
Api**



rdbms

3rd normal

Pupil Table

Pupil ID	First Name	Last Name	ClassID
1	Bob	Jones	1
2	Bill	Jones	2
3	Fred	Jones	1

Class Table

Class ID	Class Name	Room
1	CompSci 101	S16
2	English 101	M42

write friendly
transaction friendly

Denormalized Form

Weather			
city	state	high	low
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flagstaff	Arizona	88	69
San Diego	California	77	60
Albuquerque	New Mexico	80	72

read friendly

Web Portal

CUD (rest)

R (rest)

Inventory



Normalised db

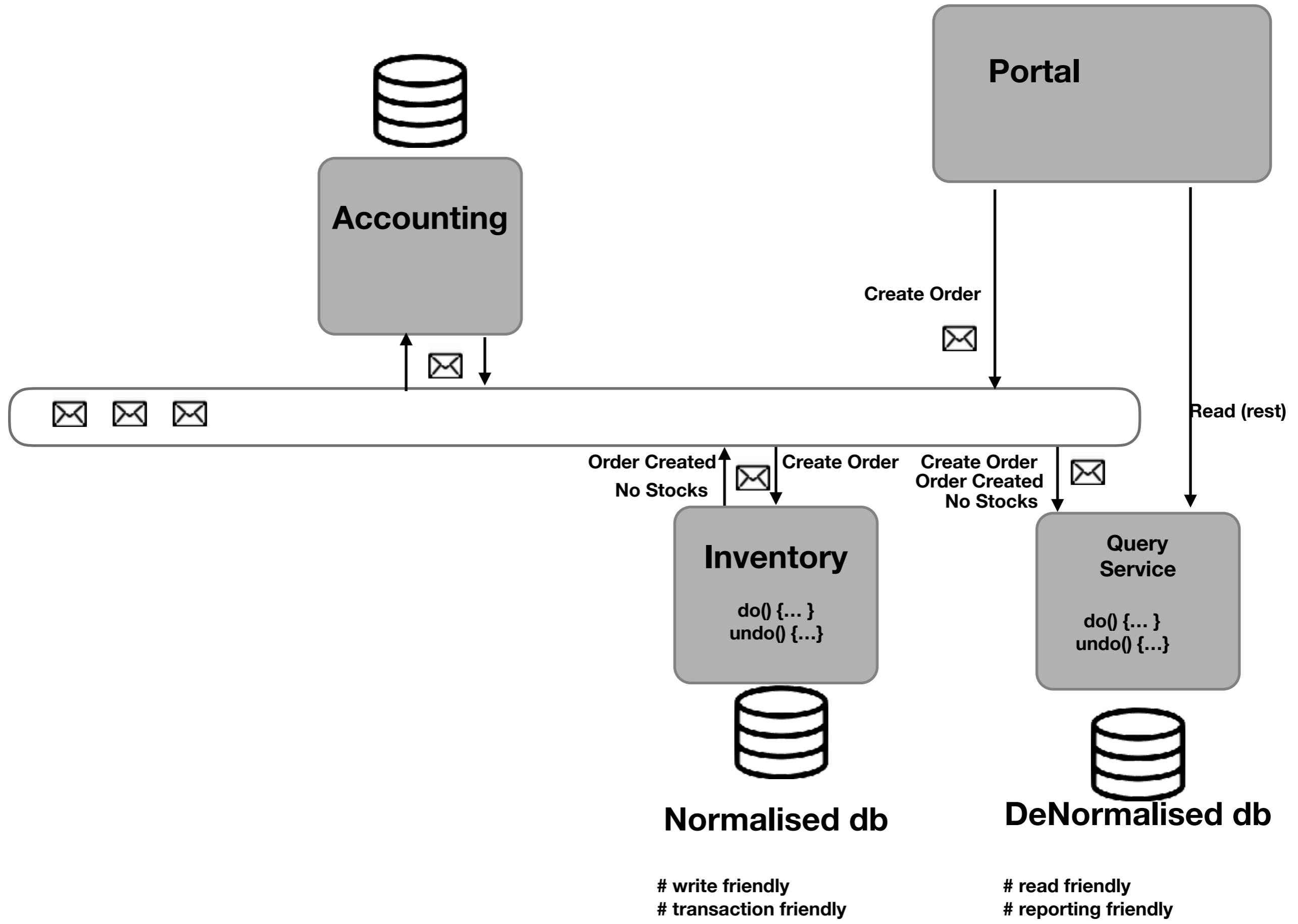
write friendly
transaction friendly

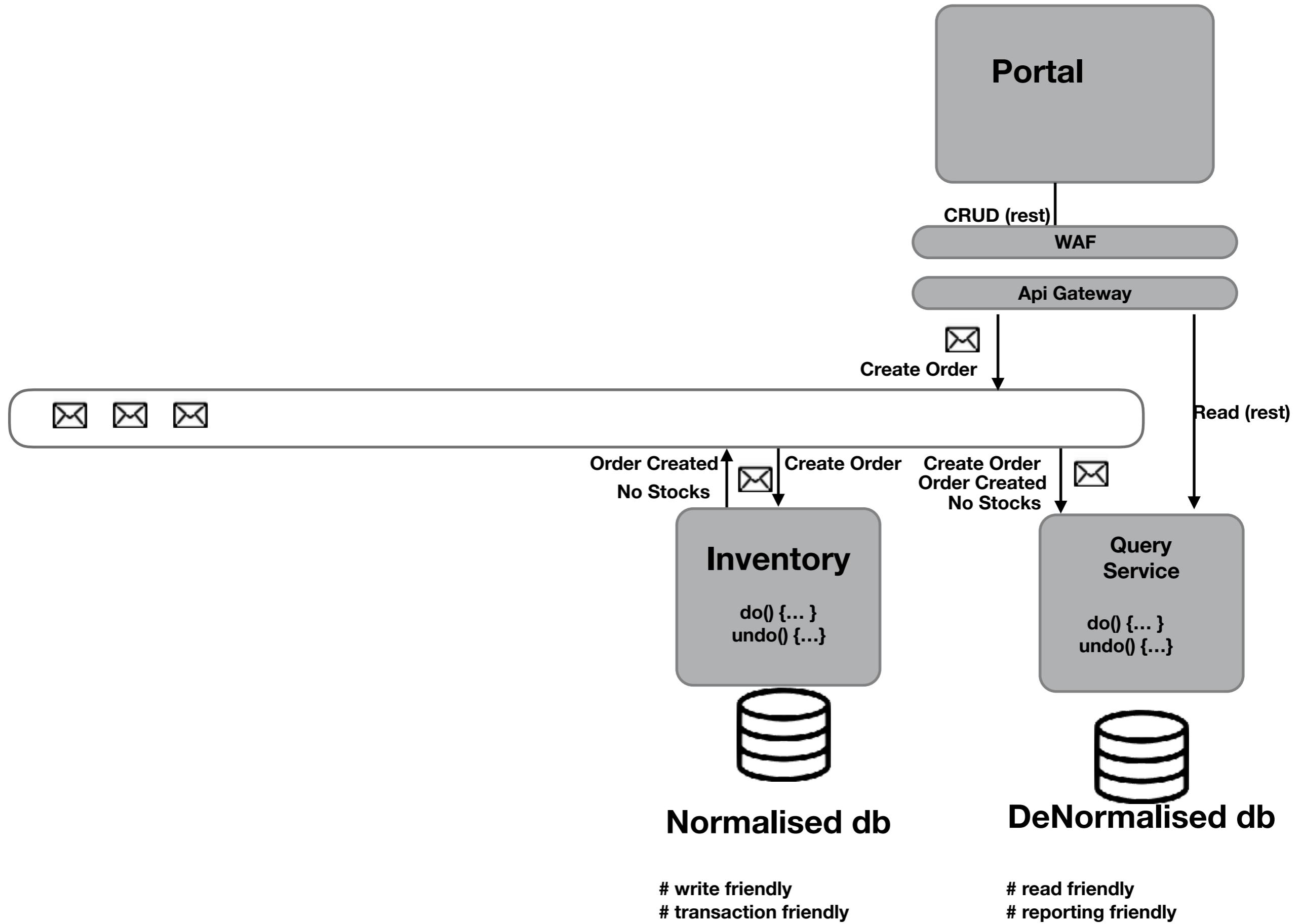
Query Service



DeNormalised db

read friendly
reporting friendly

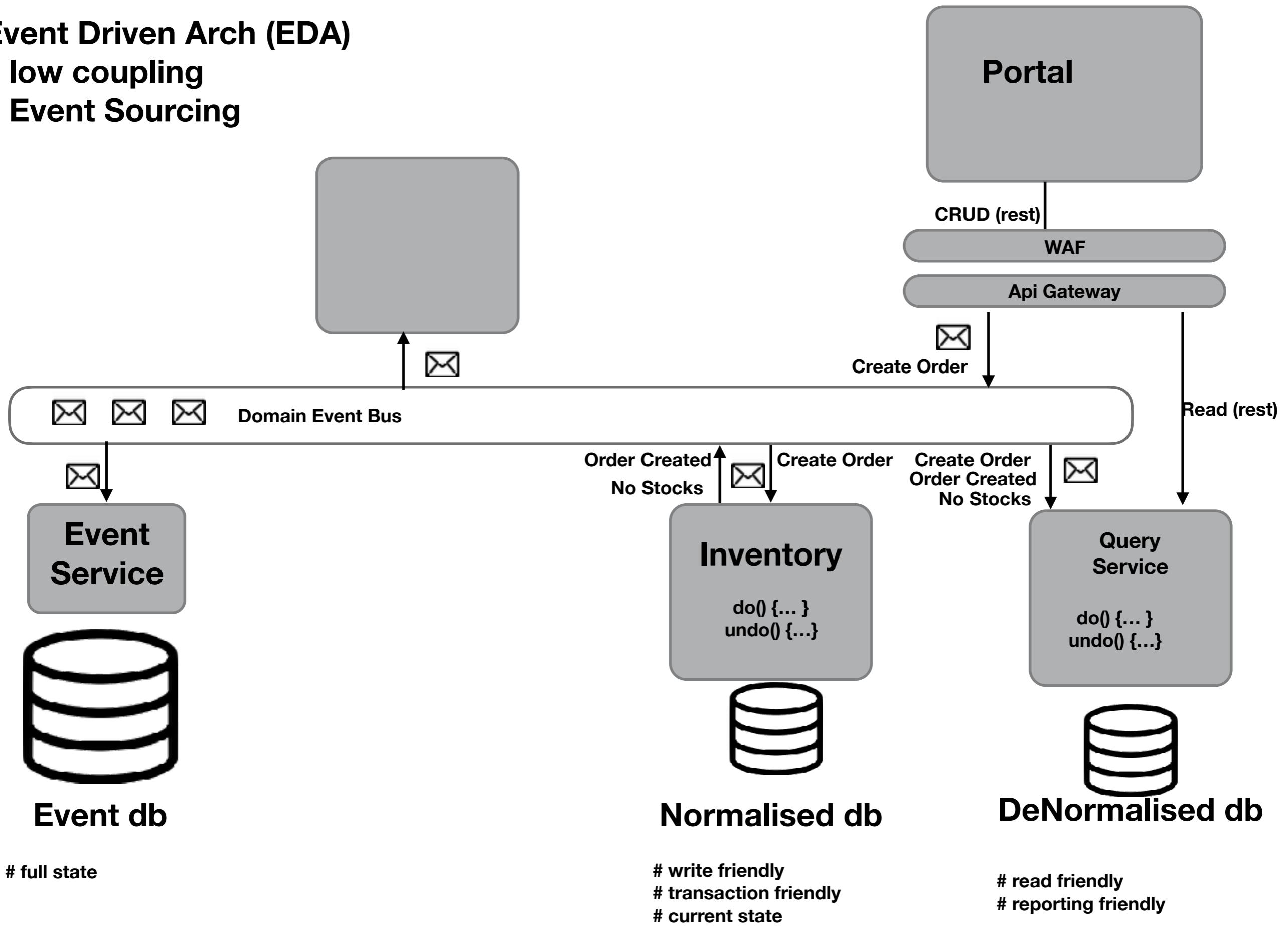


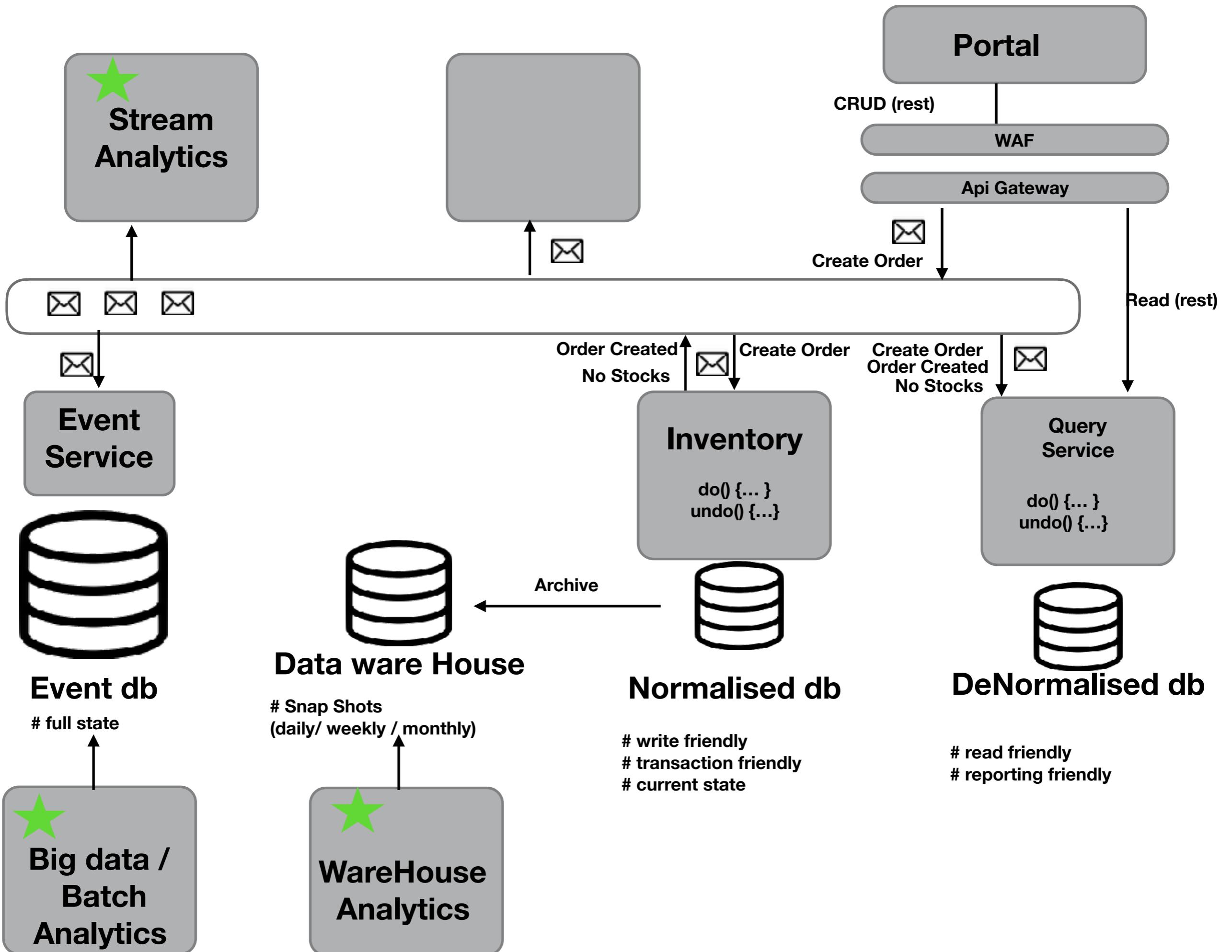


Event Driven Arch (EDA)

low coupling

Event Sourcing



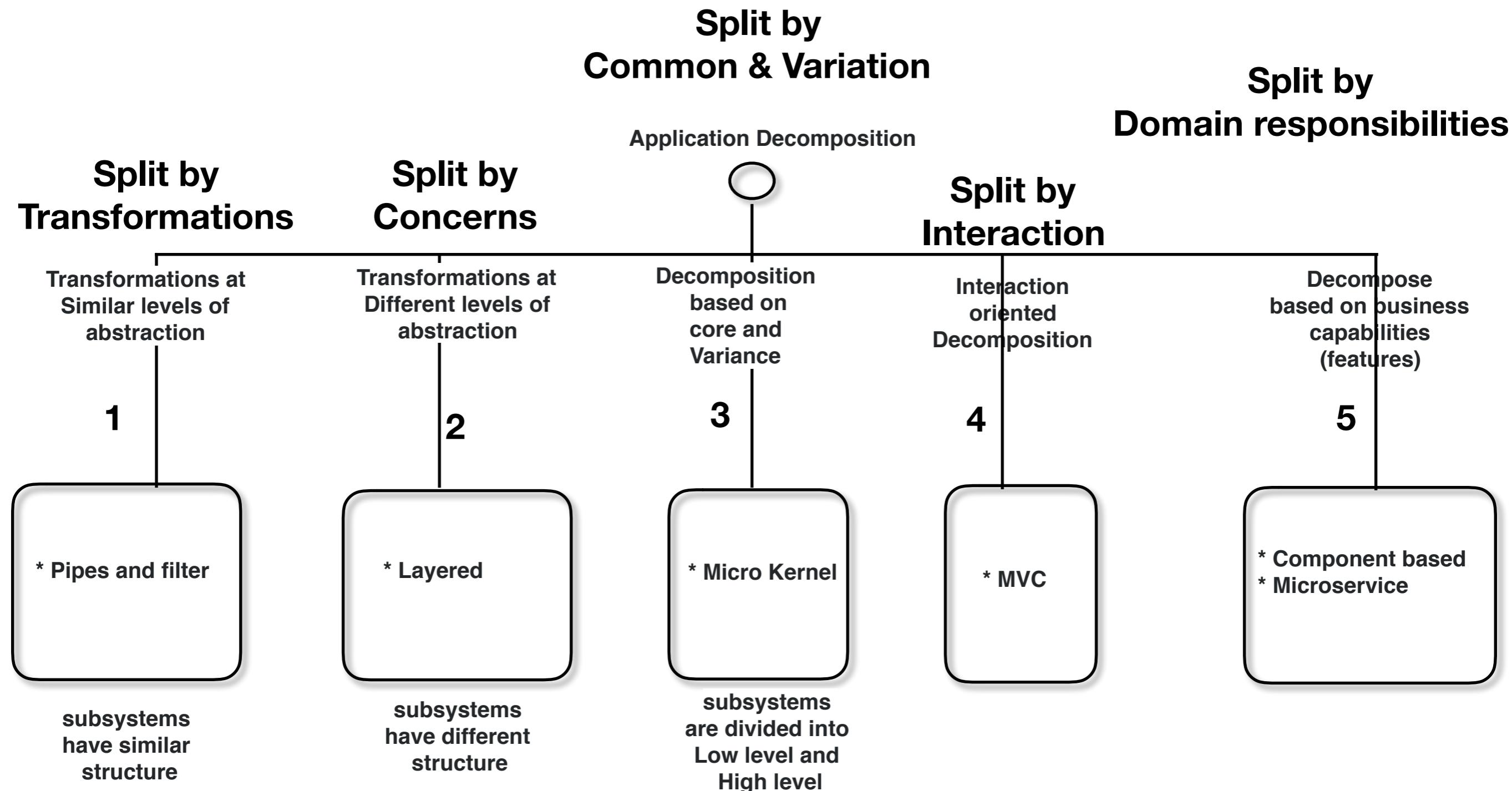


	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent	yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,...	Py, java, php, .net, js,...	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	

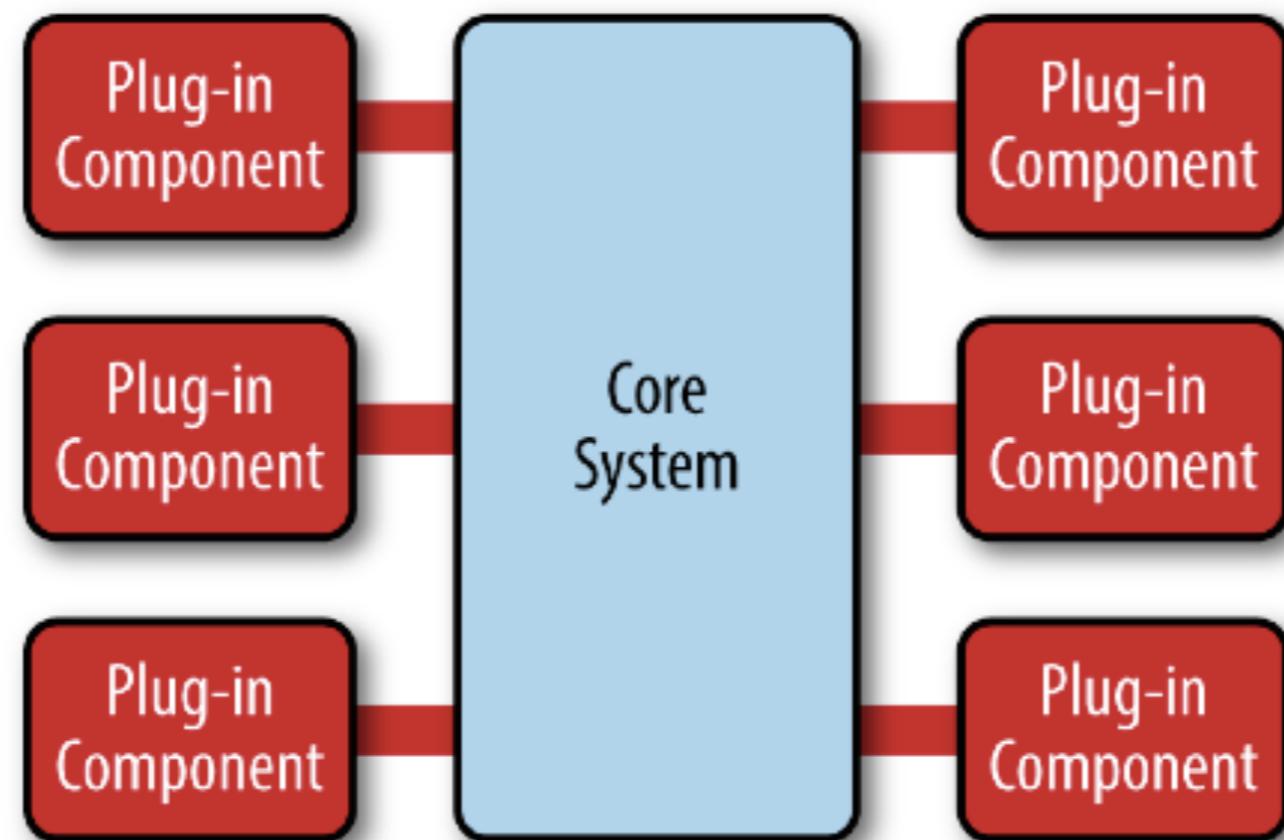
Application Decomposition



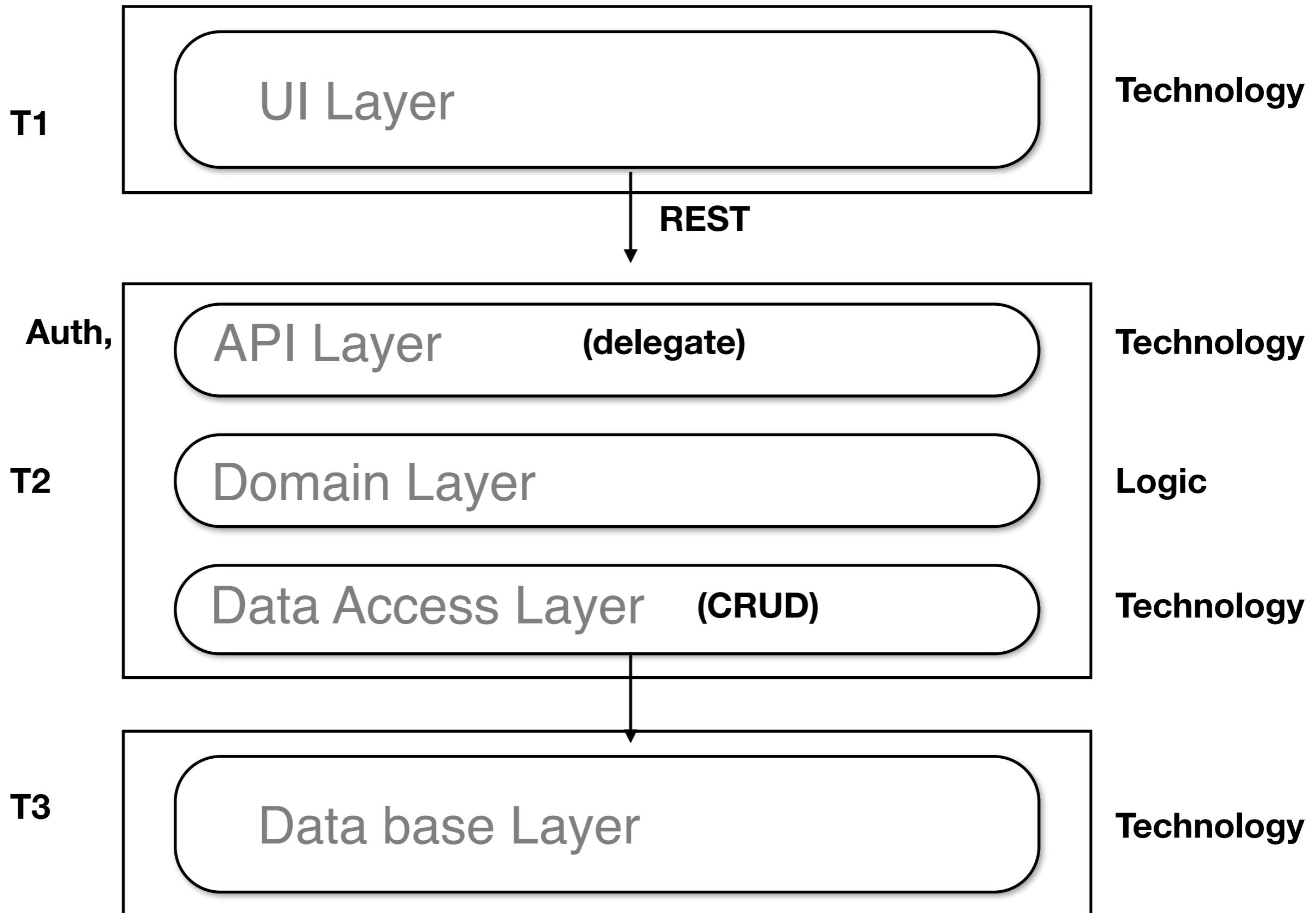
Decompose application



Microkernel Architecture



Logical View



Invoke logic()

Layer 1

f1(int) f2() f3(int,int)

Layer 2

f1(bool) f3(int,int)
f2(double) f4()

Layer 3

f1() f4()
f2(int) f3() f5(int)

Layer 4

f1(int) f2()

Process Data()

Filter 1

f1(int)

Filter 2

f2(int)

Filter 3

f3(int)

Filter 4

f4(int)

Data

Pipe

* Can swap filters

Break up the User Interface layout and Logic into separate components. This way, it's much easier to manage and make changes to either side without them interfering with each other.

Eclipse IDE. Downloading the basic Eclipse product provides you little more than a fancy editor. However, once you start adding plug-ins, it becomes a highly customizable and useful product.

An application is required to perform a variety of tasks of varying complexity on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, could change as business requirements are updated. Also, additional processing might be required in the future, or the order in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is decomposed into more manageable parts.

claims processing system contains the basic business logic required by the insurance company to process a claim, except without any custom processing. most insurance claims applications leverage large and complex rules engines to handle much of this complexity.

However, these rules engines can grow into a complex big ball of mud where changing one rule impacts other rules, or making a simple rule change requires an army of analysts, developers, and testers. State-specific rules and processing is separate from the claims system and can be added, removed, and changed with little or no effect on the rest of the system or other modules.

1

Application
Exe

2

Application
Exe

3

Process

In process
comp

**Compile time
monolithic**

**Link time
monolithic**

**Runtime
monolithic**

4

Process

Pipes
Sockets
http
Roc
...

Process

5

Process

Process

Distributed
comp

Distributed
comp

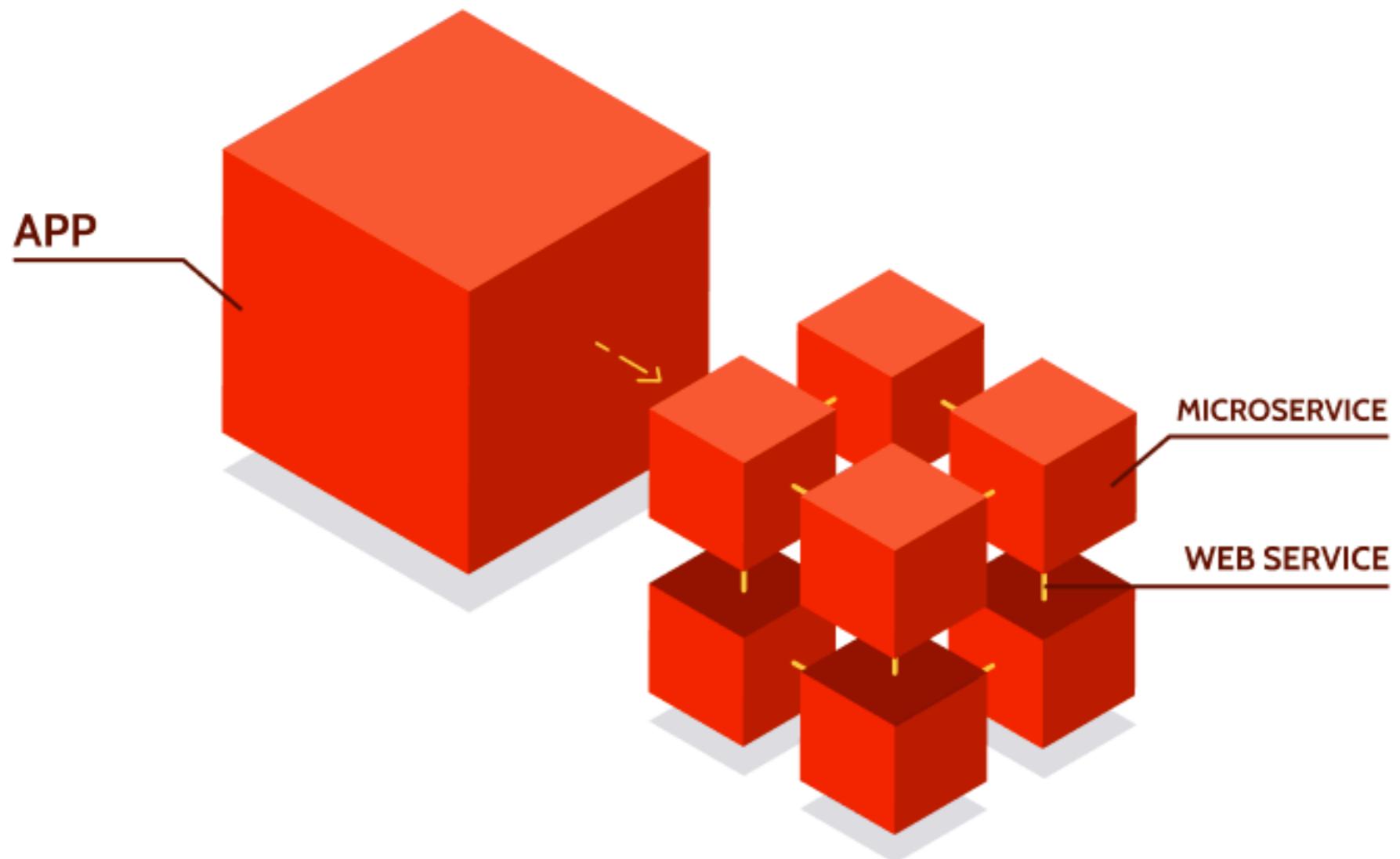
Application1

Application2

Distributed Application

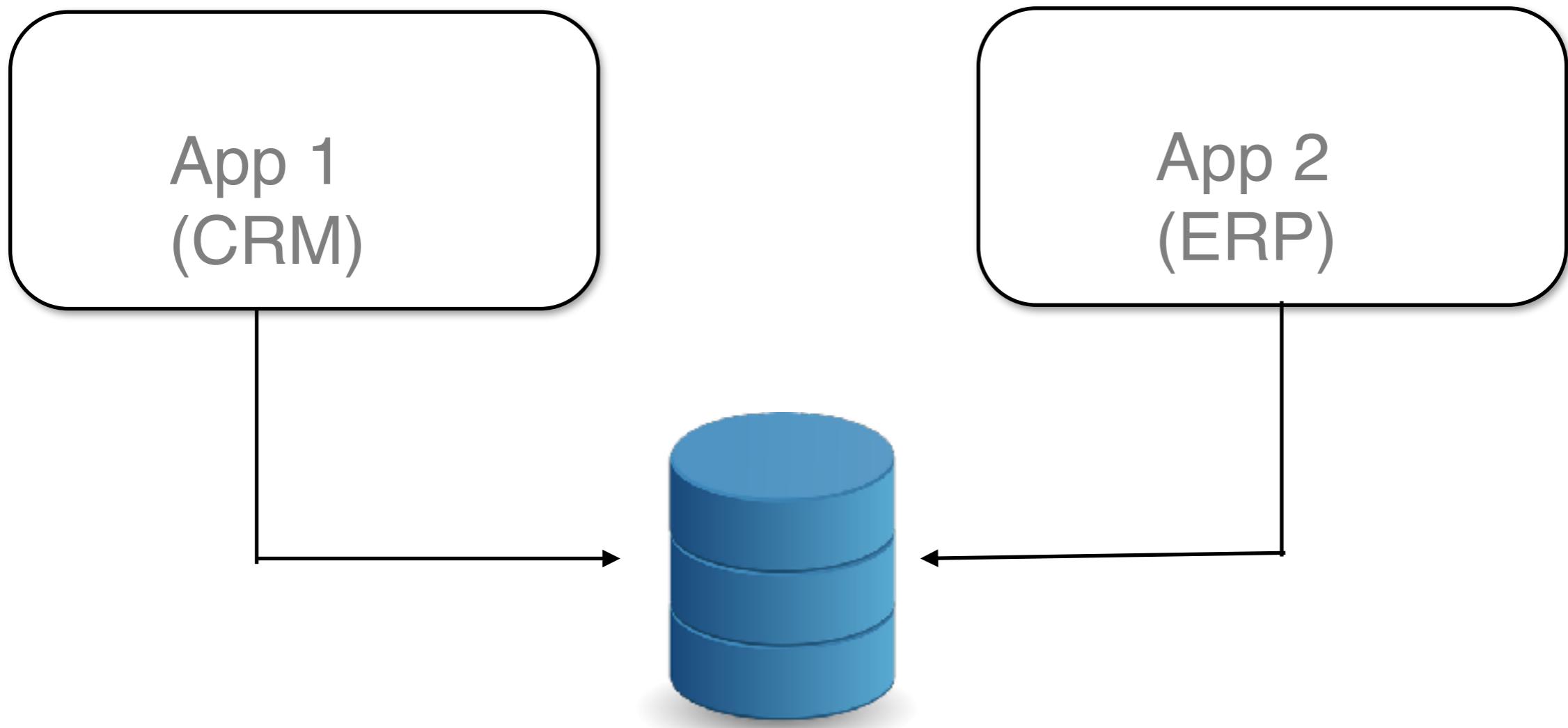
Micro Application

Microservice

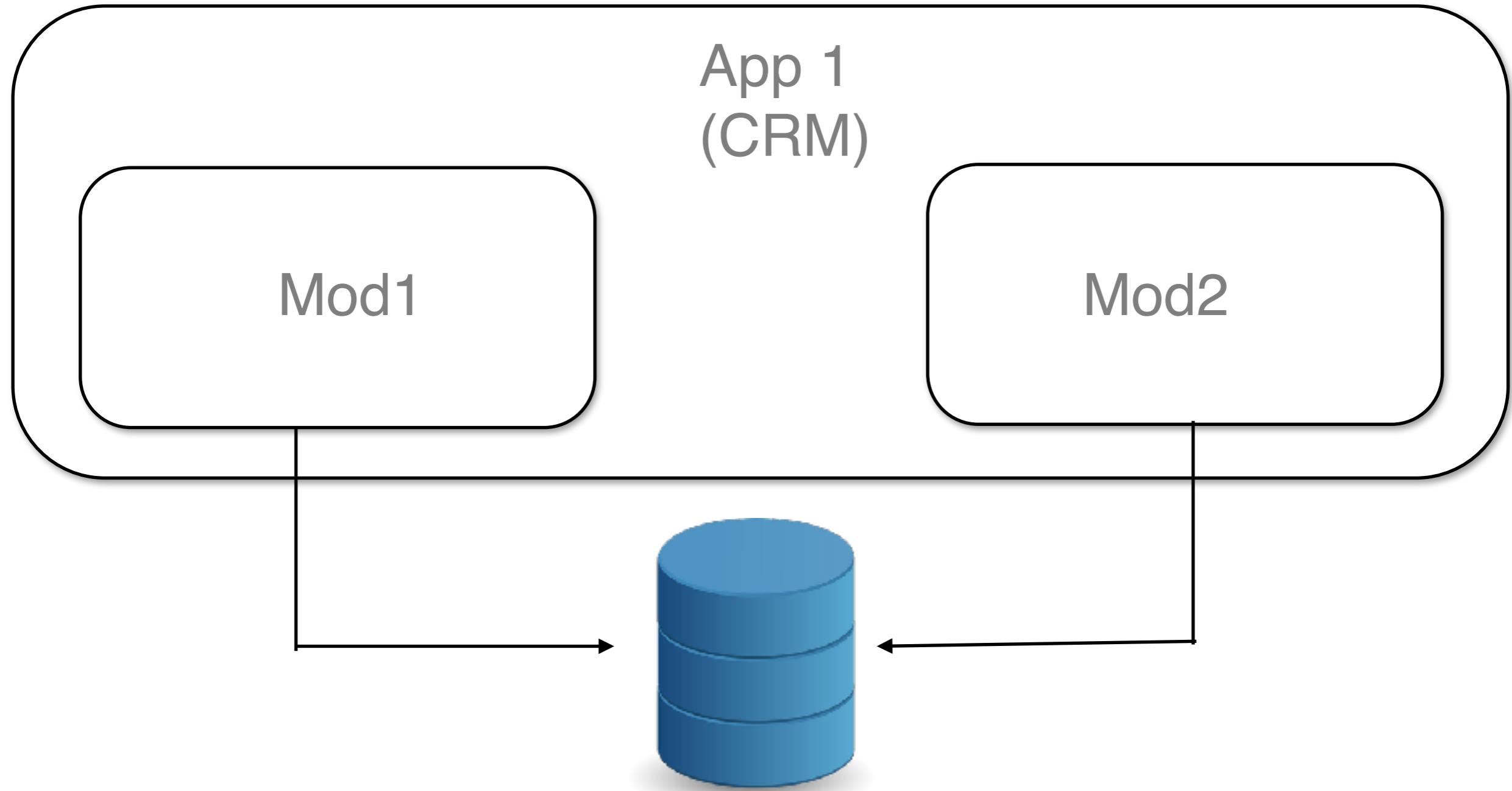


	2 modules	2 Applications	W	Score
Database / Storage	Shared	Not Shared	2	
Infra (Hosting)	Shared	Not Shared	3	
Sorce Control	Shared	Not Shared	2	
CI/CD (Build Server)	Shared	Not Shared	3	
Fun Requirements	Shared	Its Own	1	
SCRUM Team / Sprint	Shared	Its own	1	
Test Cases	Shared	Its own	1	
Architecture	Shared	Its own	1	
Technology Stack / Fwks	Shared	Its own	1	

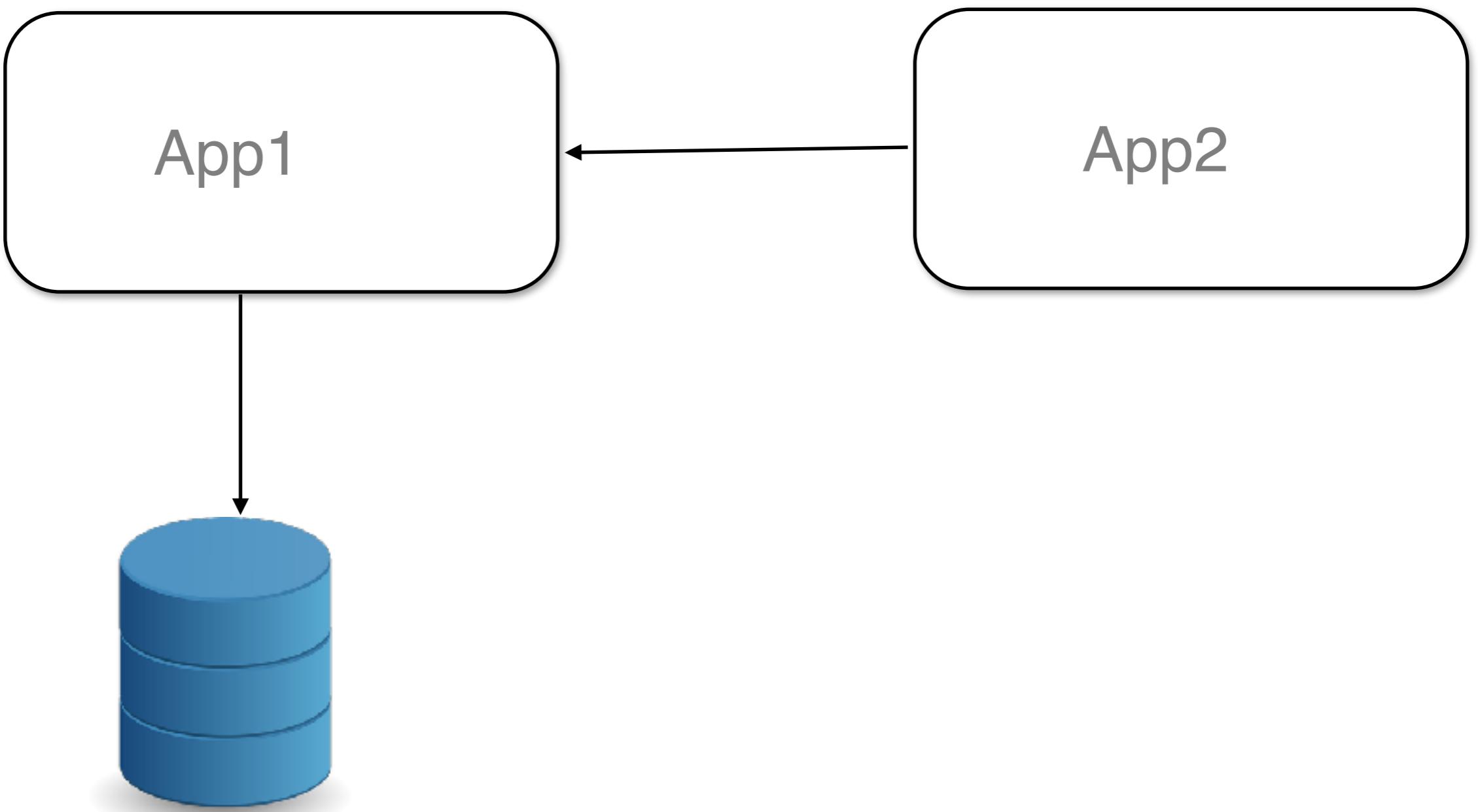
1. Database



1. Database



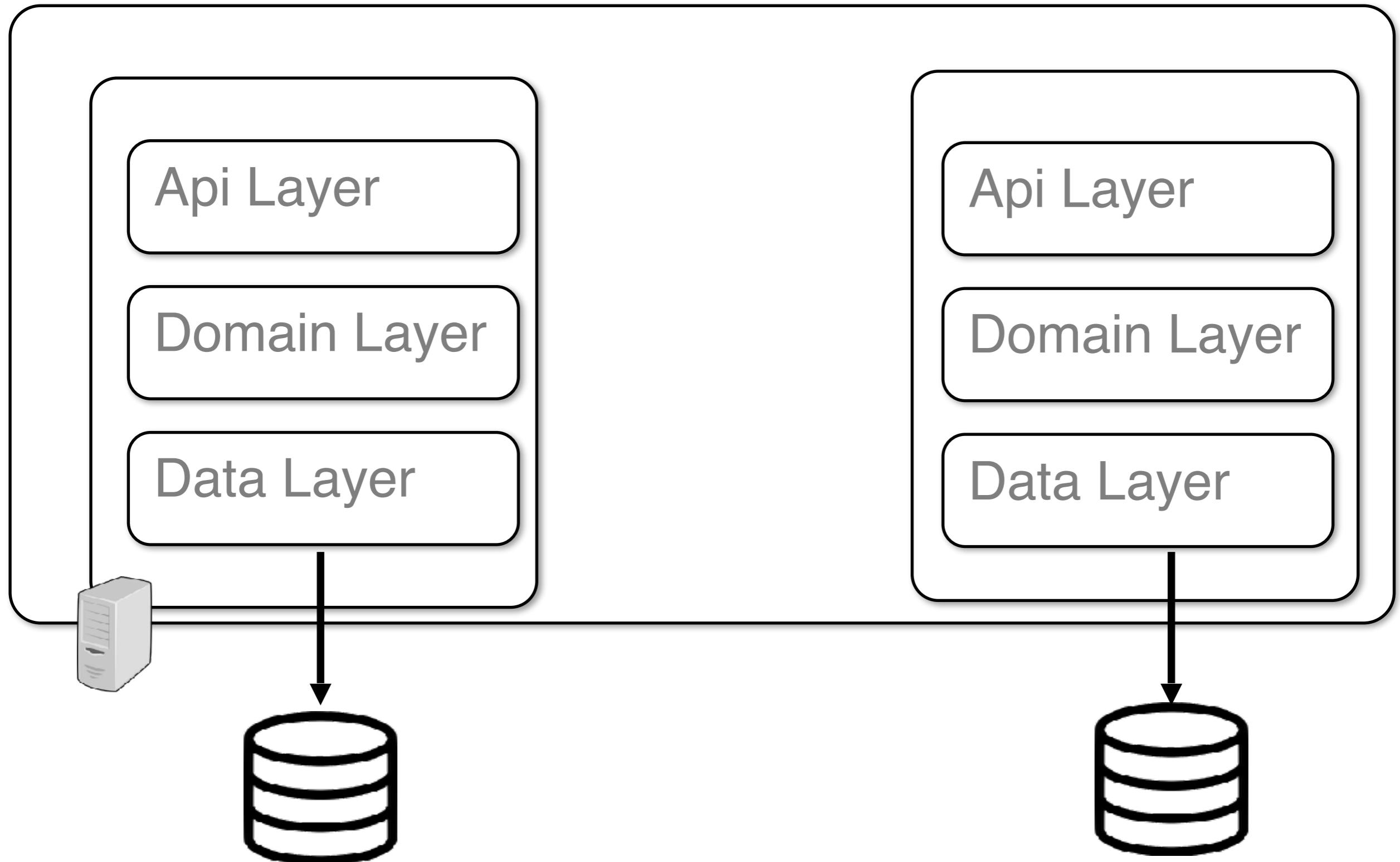
1. Database



2. Infrastructure

Web Server

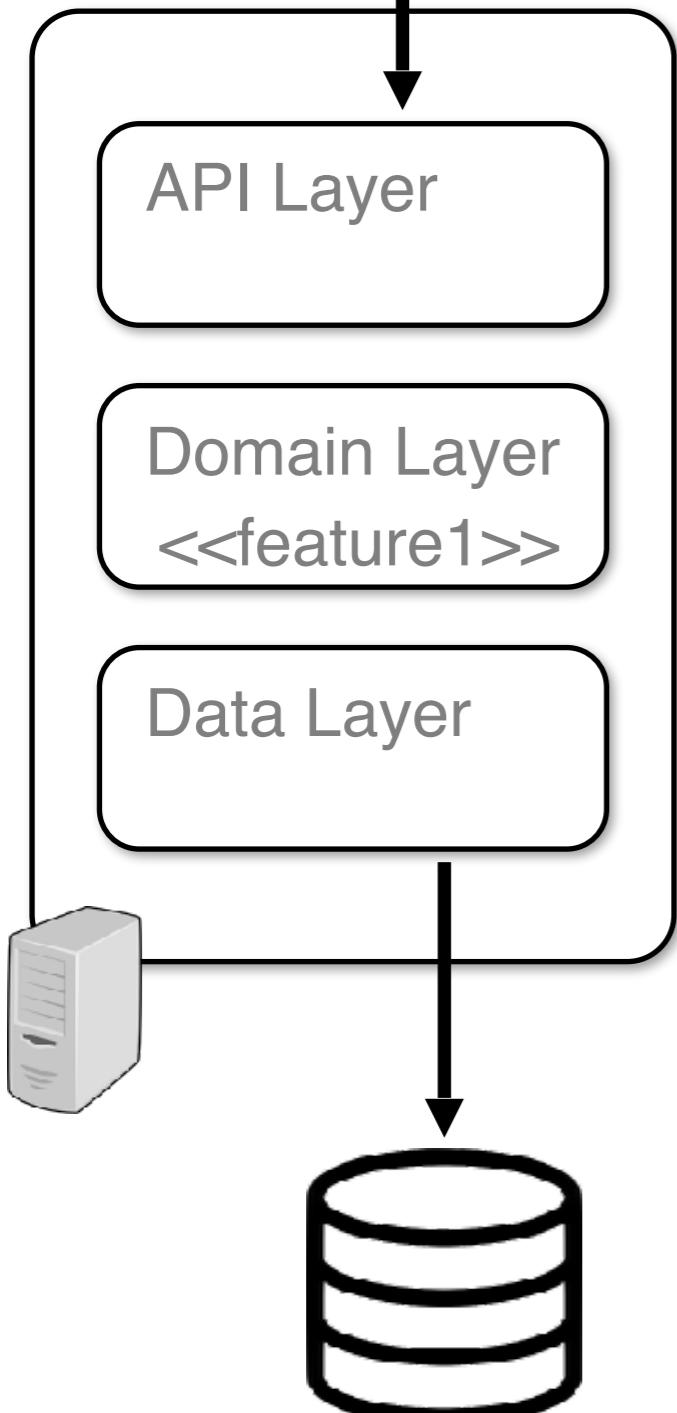
Comp1



1

UI App - SPA

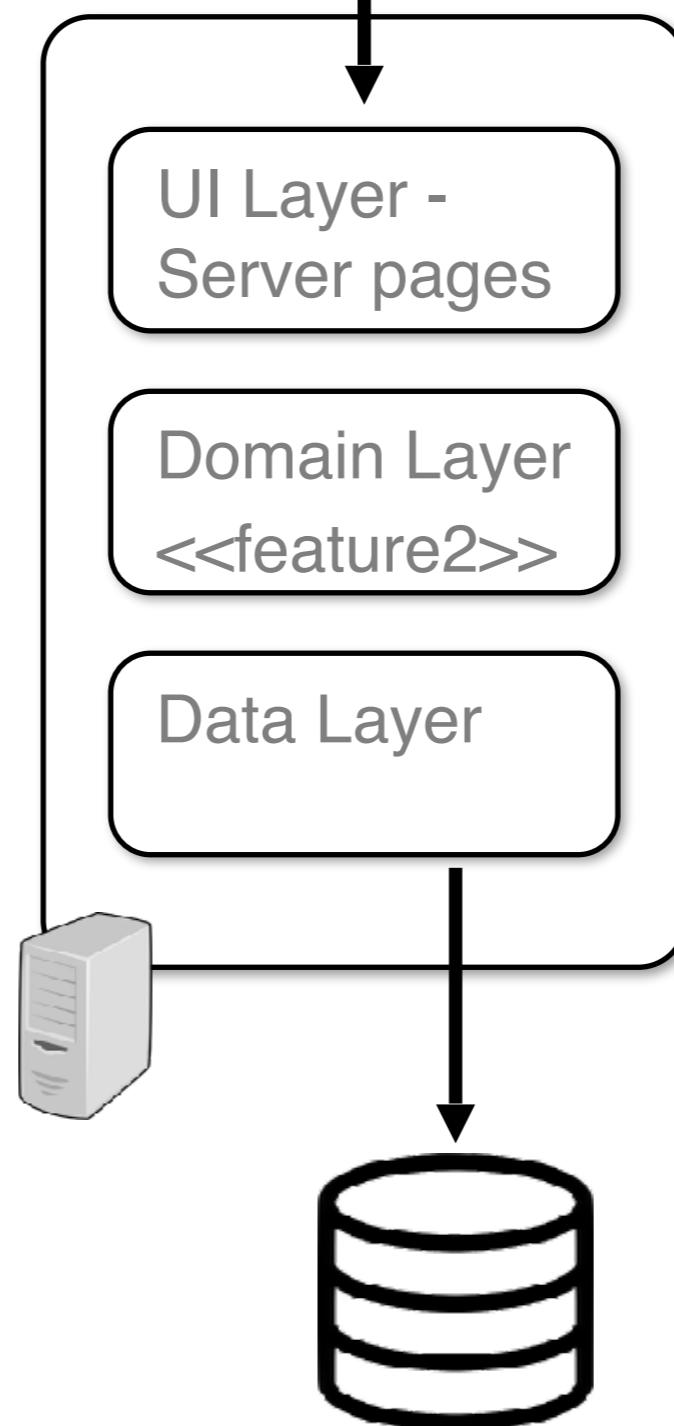
Web Server



2

Browser

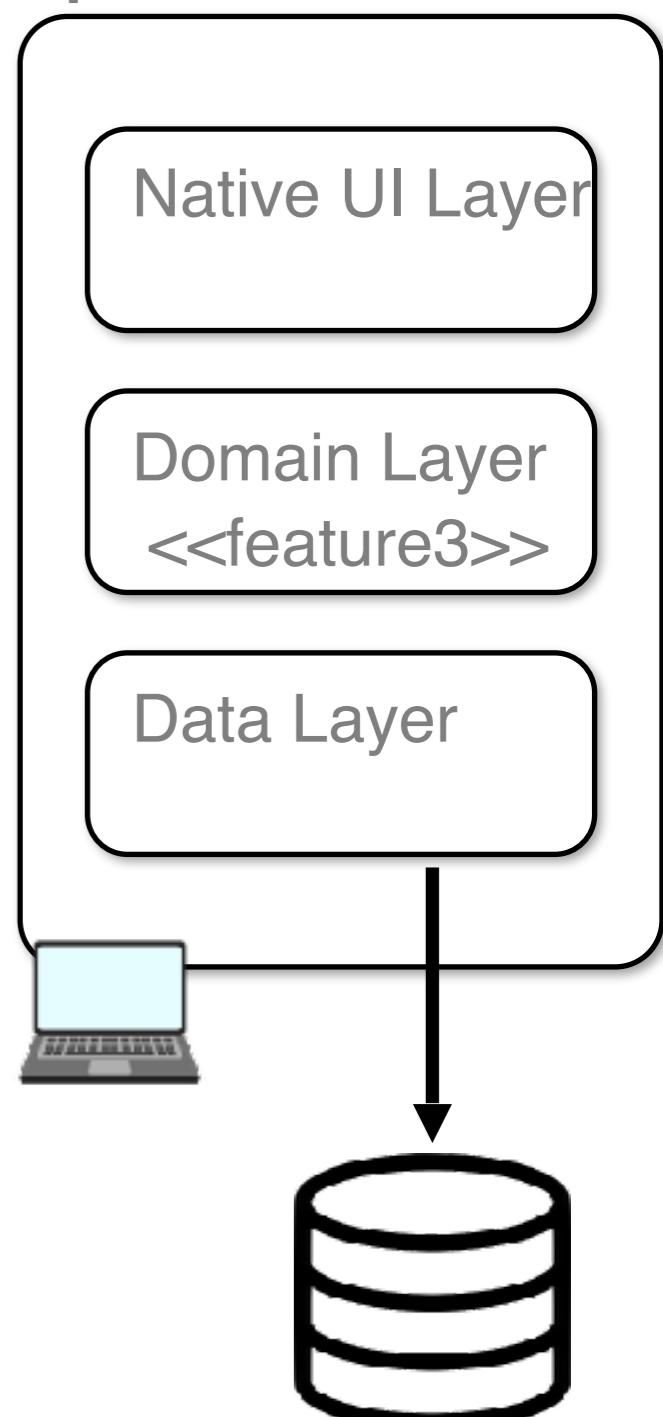
Web Server



3



Desktop

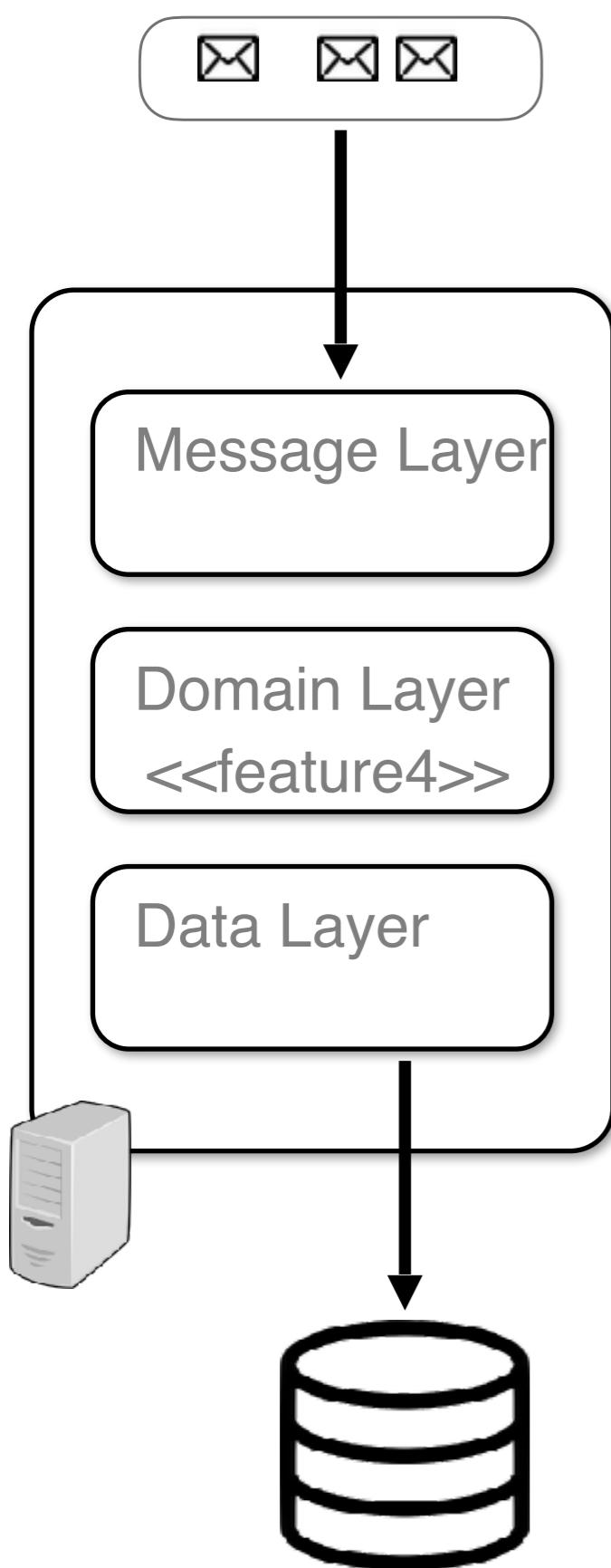


Angular (SPA)

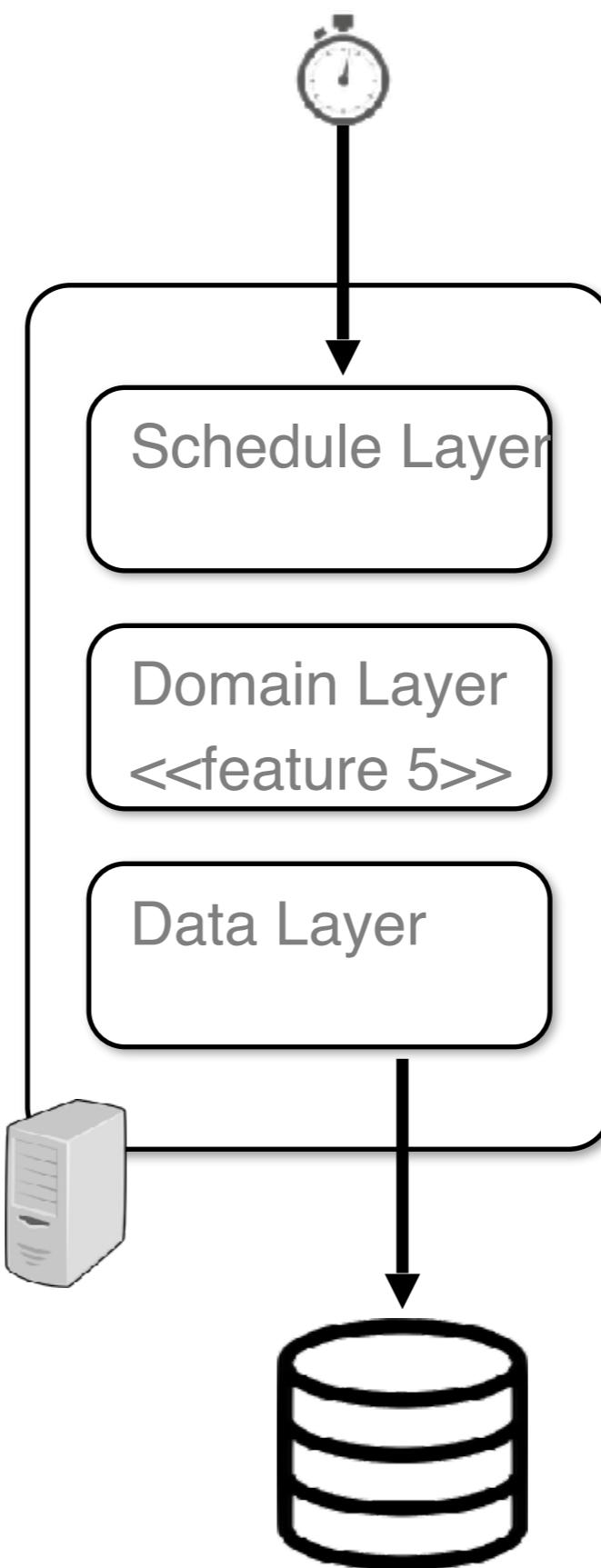
JSP, Servlet, JSF (Server pages)

Swing, Android, Ios (Native)

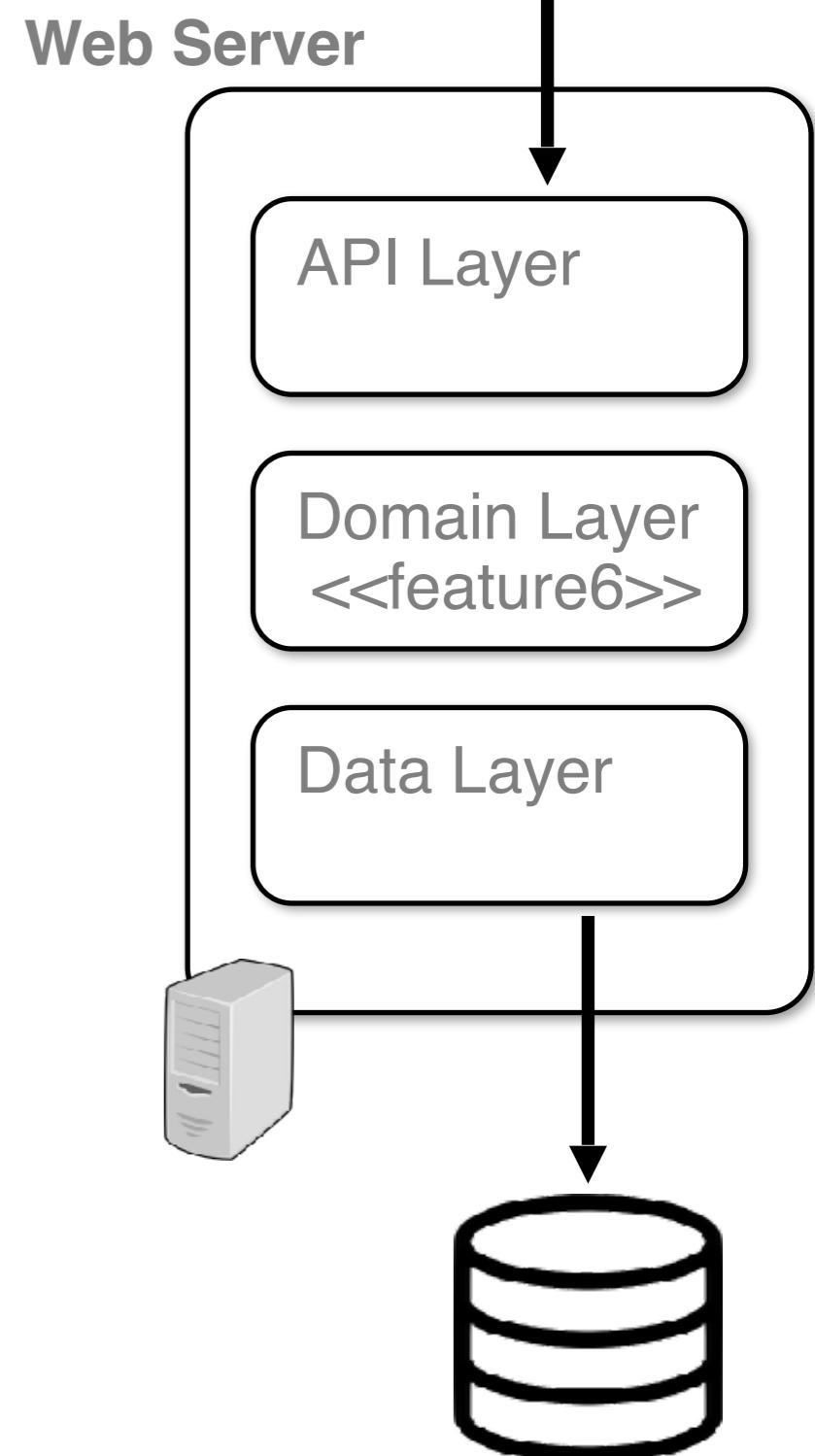
1



2



3



1

Application
Exe

2

Application
Exe

3

Process

In process
comp

**Compile time
monolithic**

**Link time
monolithic**

**Runtime
monolithic**

4

Process

Process

Exe

DII/so/jar

Distributed
comp

5

Process

Process

Exe

Exe

DII/so/jar

Application1

Application2

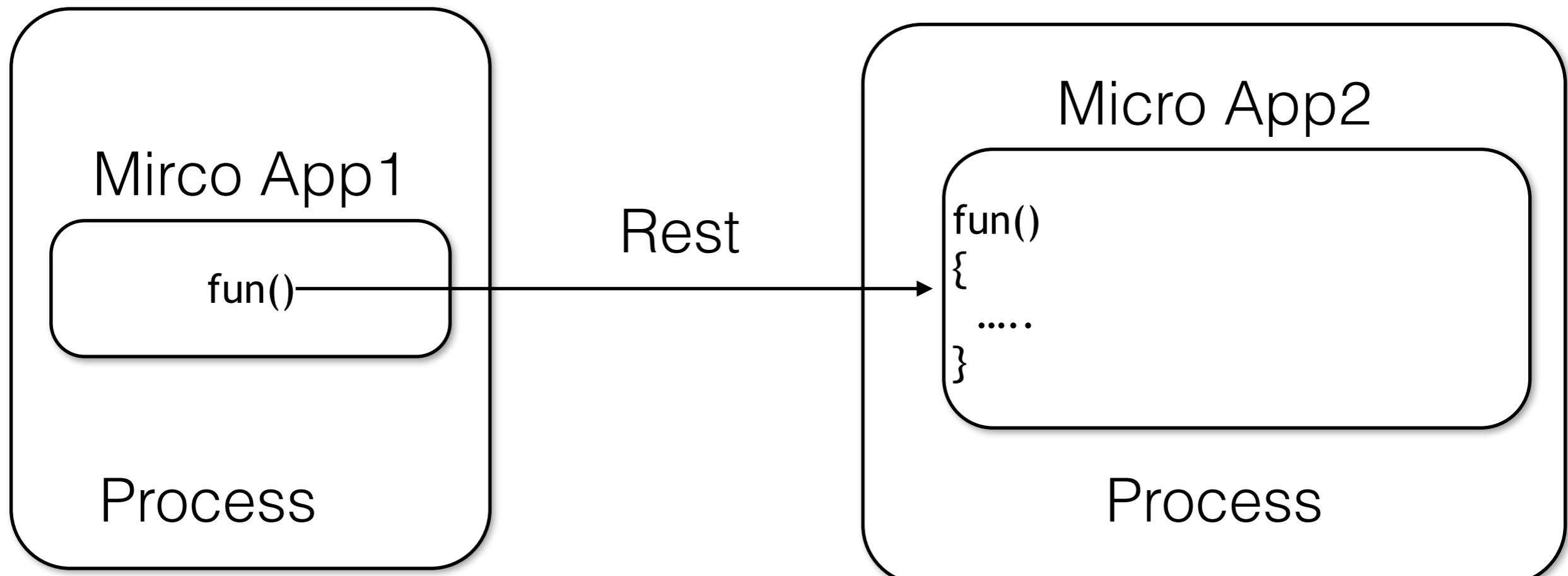
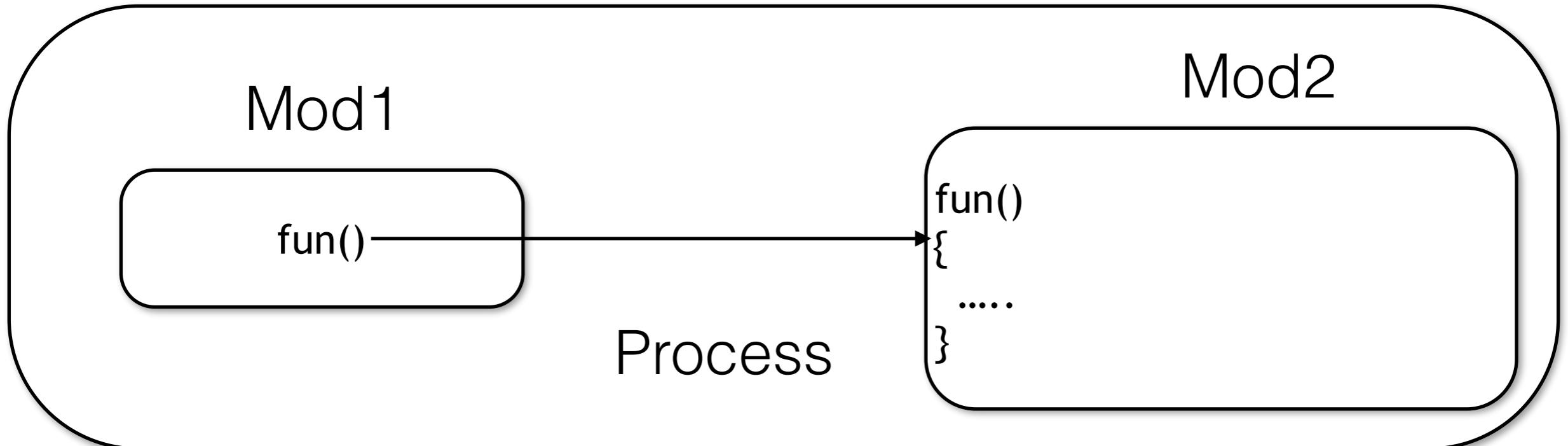
Distributed Application

Micro Application

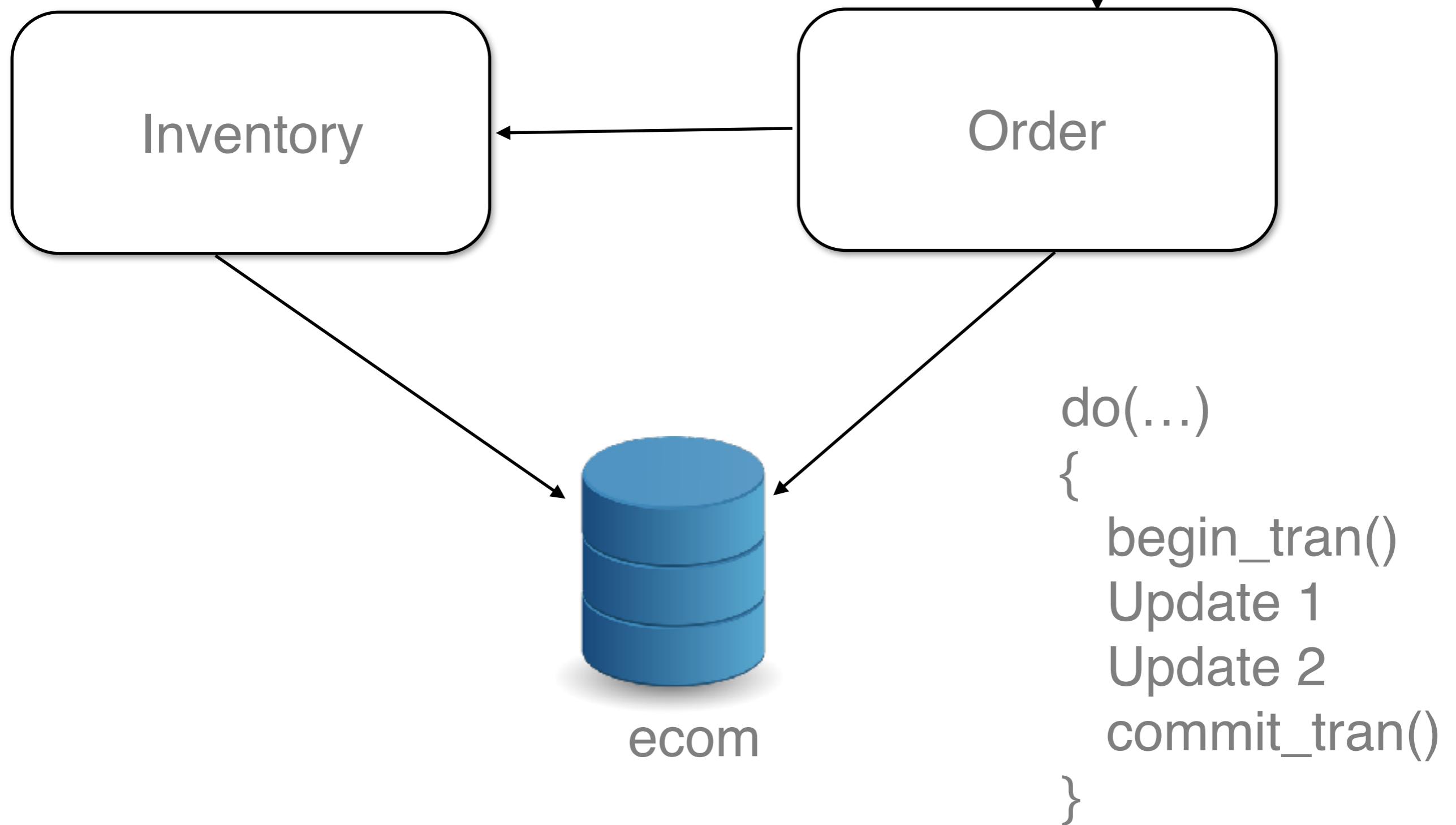
	Pros/ Cons	Solution
Development time		
Learning Curve		
Resource Performance (CPU, Memory, I/O)		
Db Transaction Management		
Views / Report / Dash board/ join		
Infra Cost		
Deployment effort		
Debugging, Error Handling,		
Integration Test		
debug/ error Log Mgmt		
Config Mgmt		
Authentication		
Authorization		
Audit Log mgmt		
Monitoring / Alerting		
Data Security and Privacy		
Build Pipeline (CI)		
Agile Architecture (Agility to change)		
Feature Shipping (Agility to ship)/ CD		
Scalability (volume - request, data,		
Resilience		
Availability		
Ability to do Polygot		
Maintainability (easy to change)		

	CPU Cycles
a + b	3 cpu
Fun call	10 cpu cycles
Create Thread	200,000 cpu cycles
Destroy thread	100,000 cpu cycles
Write file	10,00,000
API Call	20,00,000
Db call	45,00,000

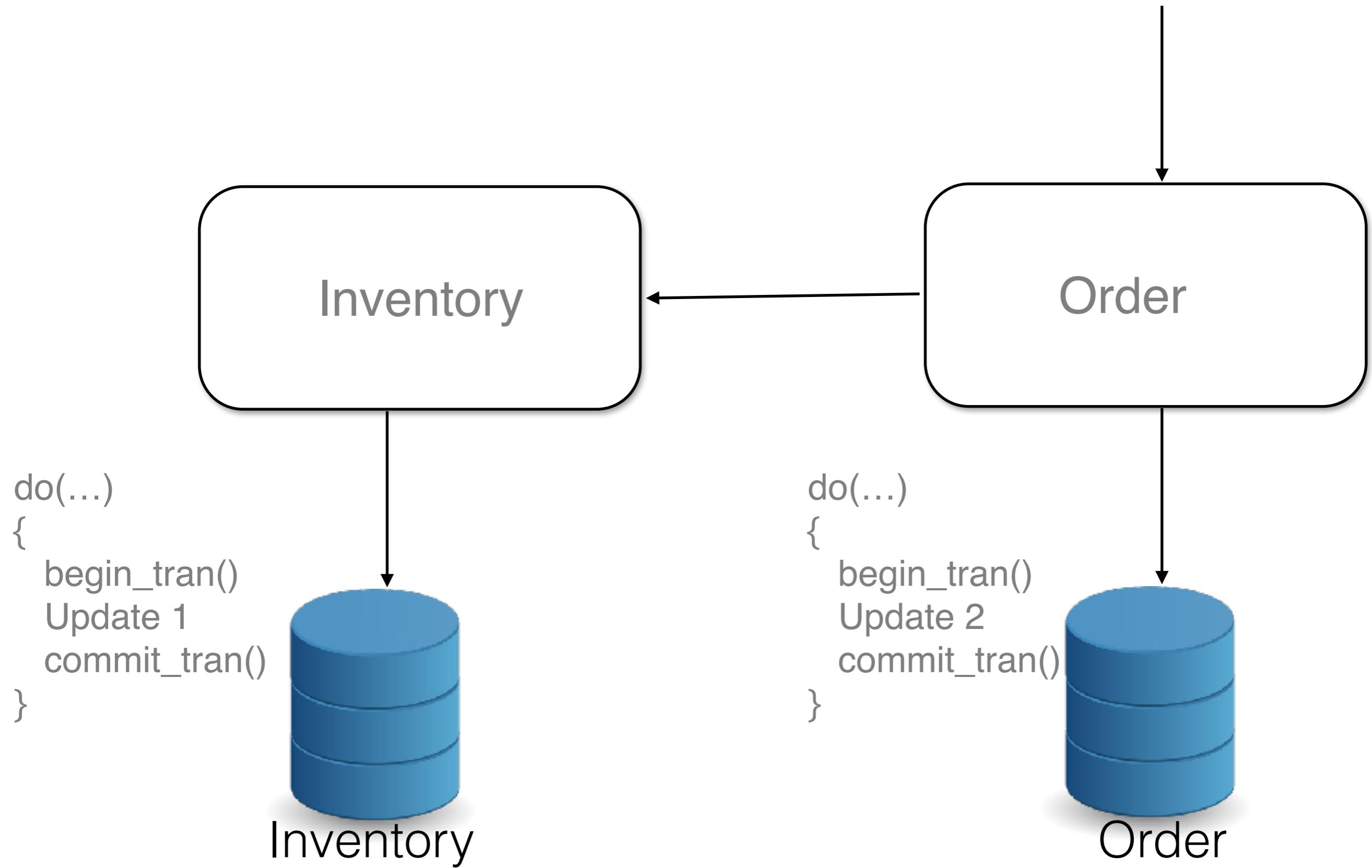
4. Development time



2. Db transaction

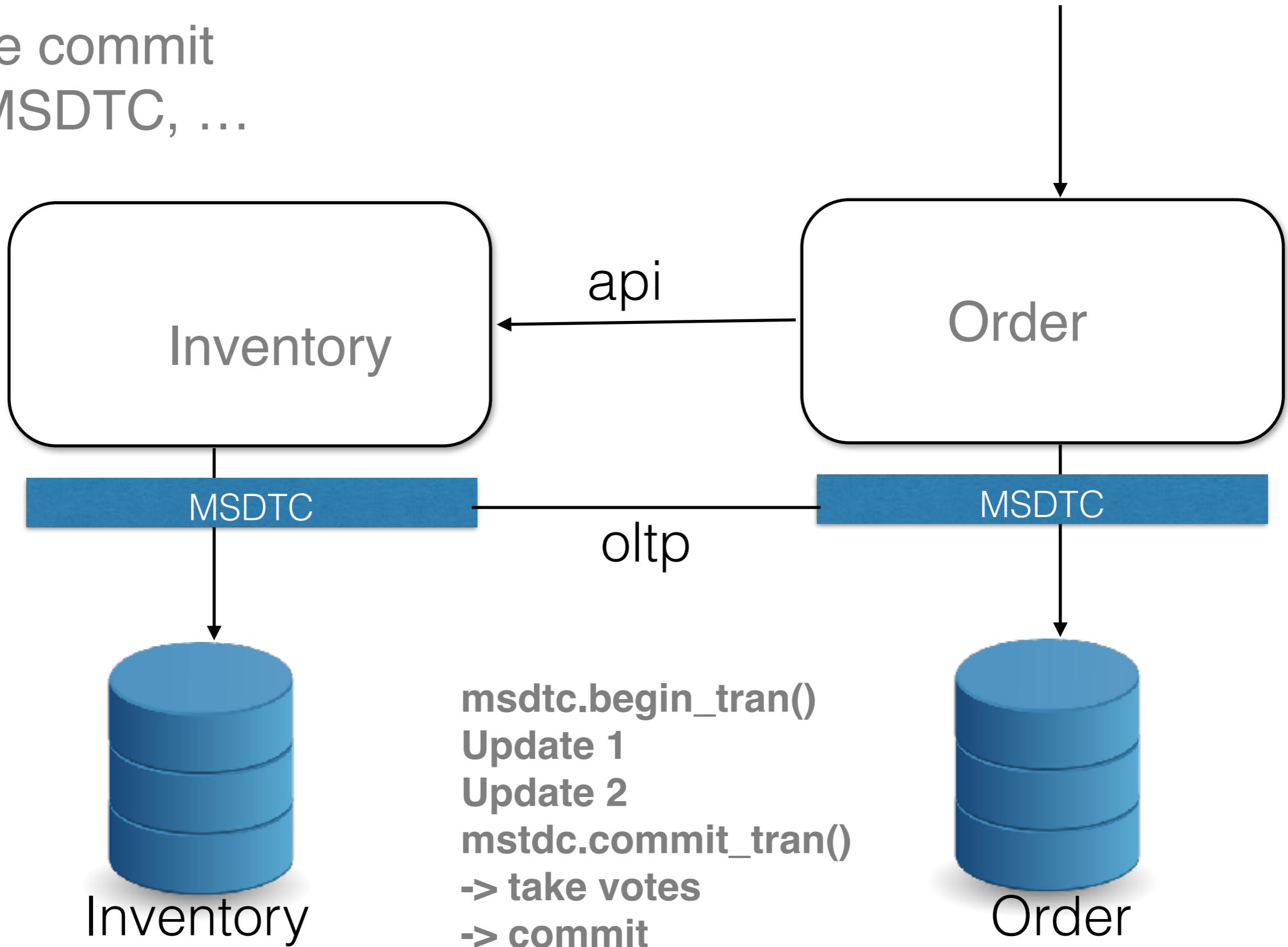


2. Db transaction

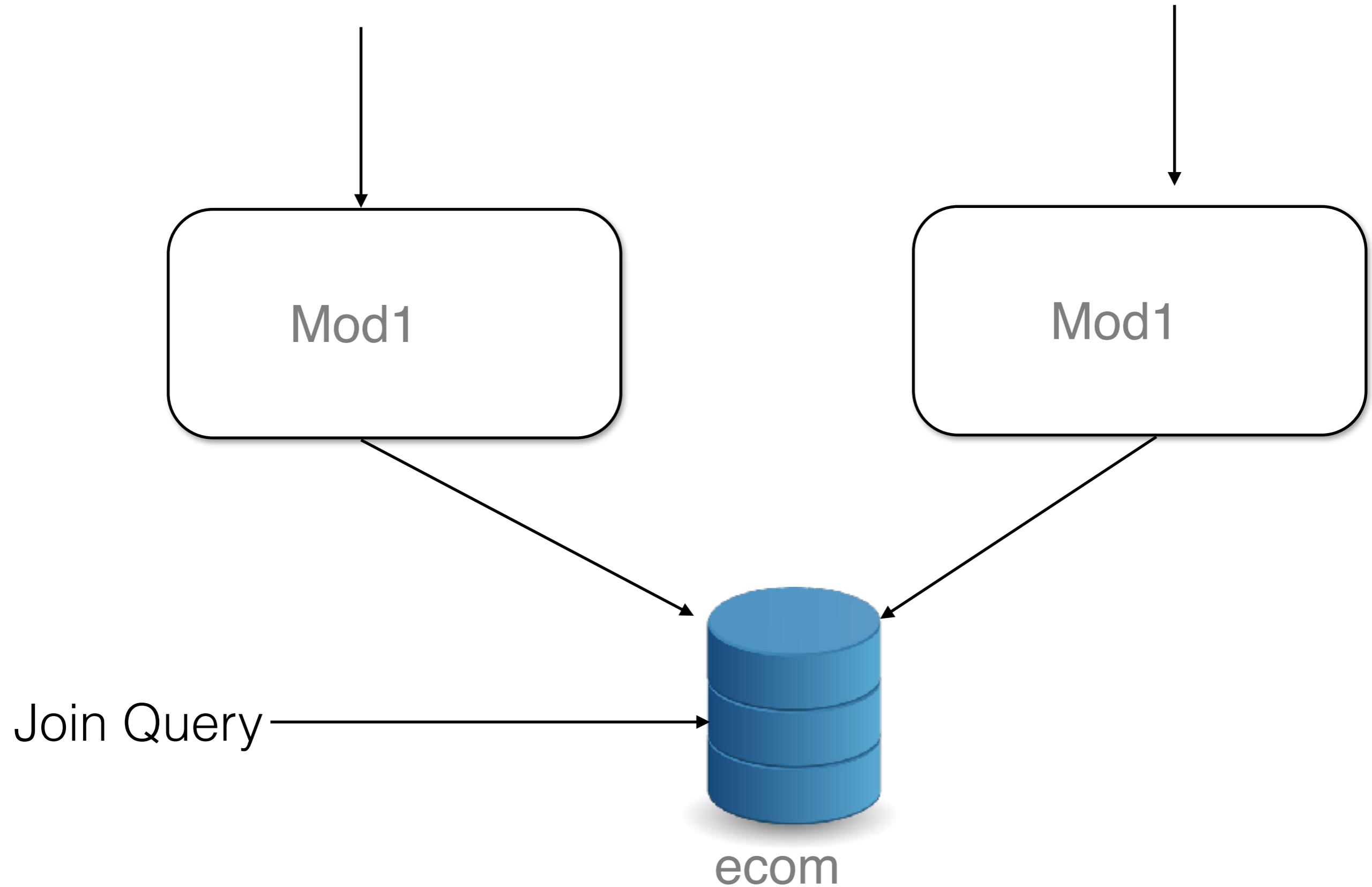


2. Db transaction

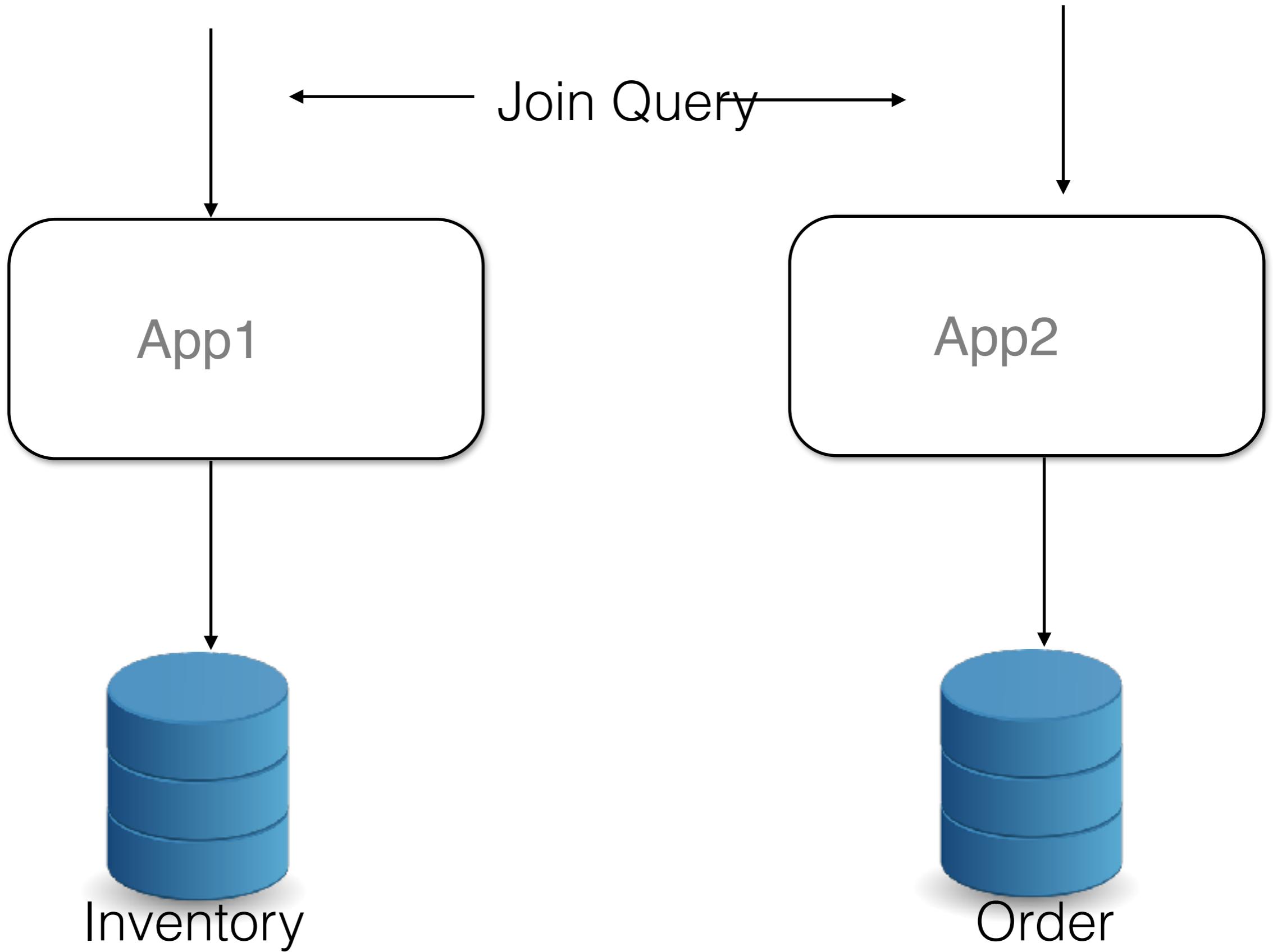
2 phase commit
JTX , MSDTC, ...



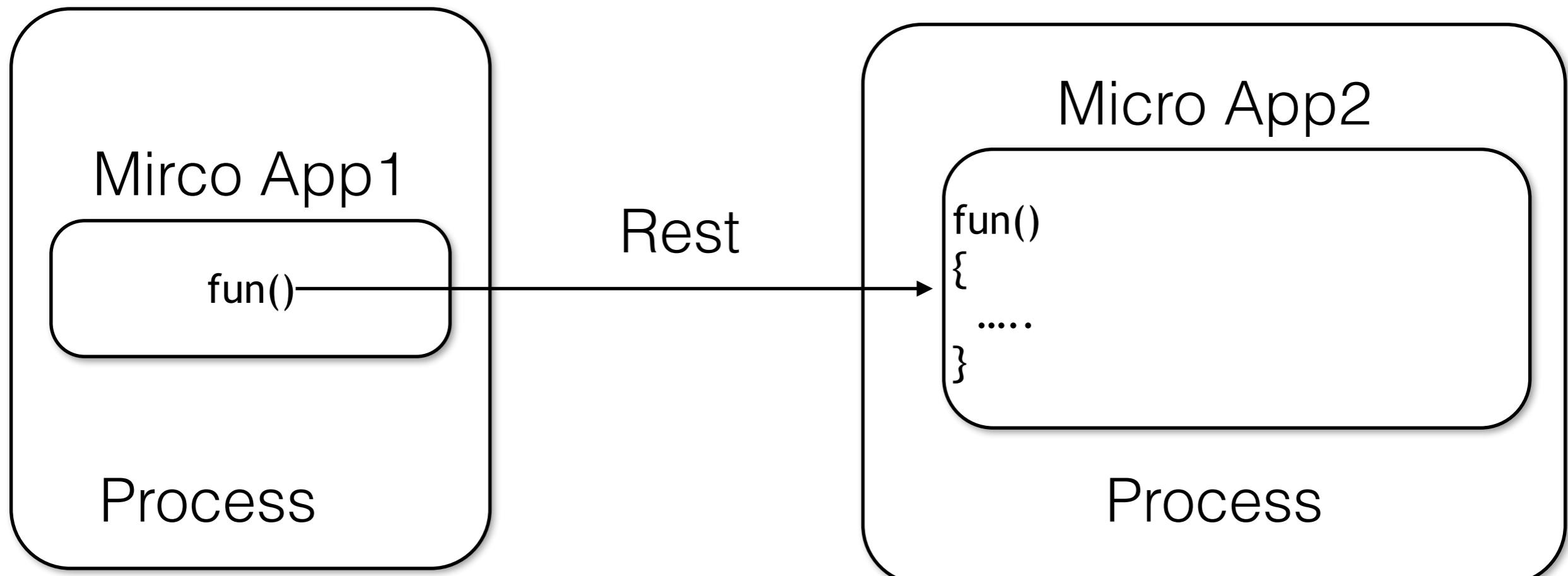
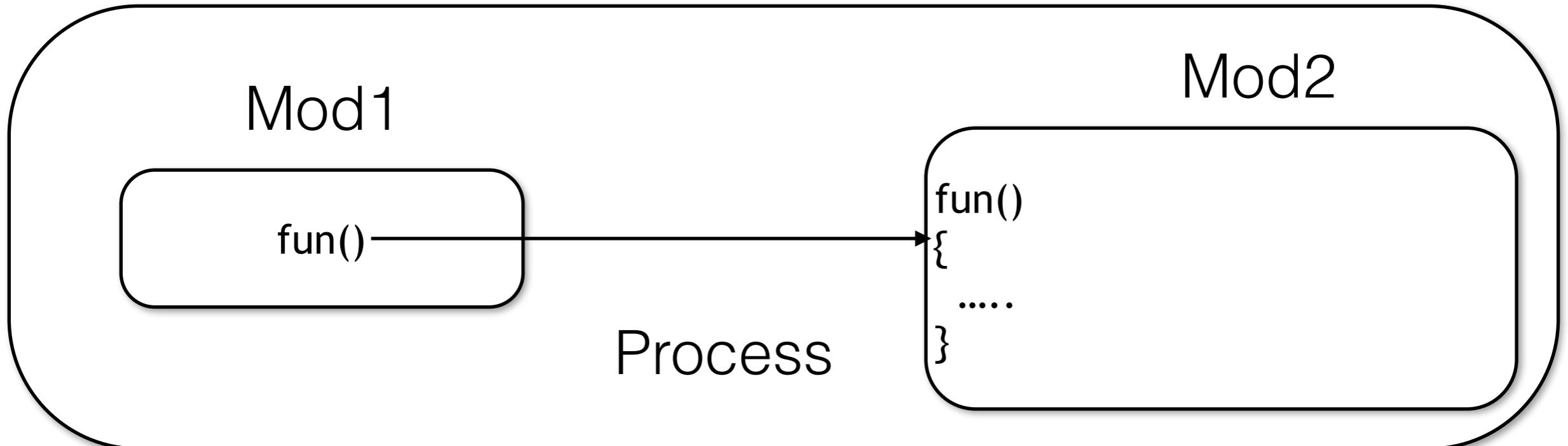
3. Db query

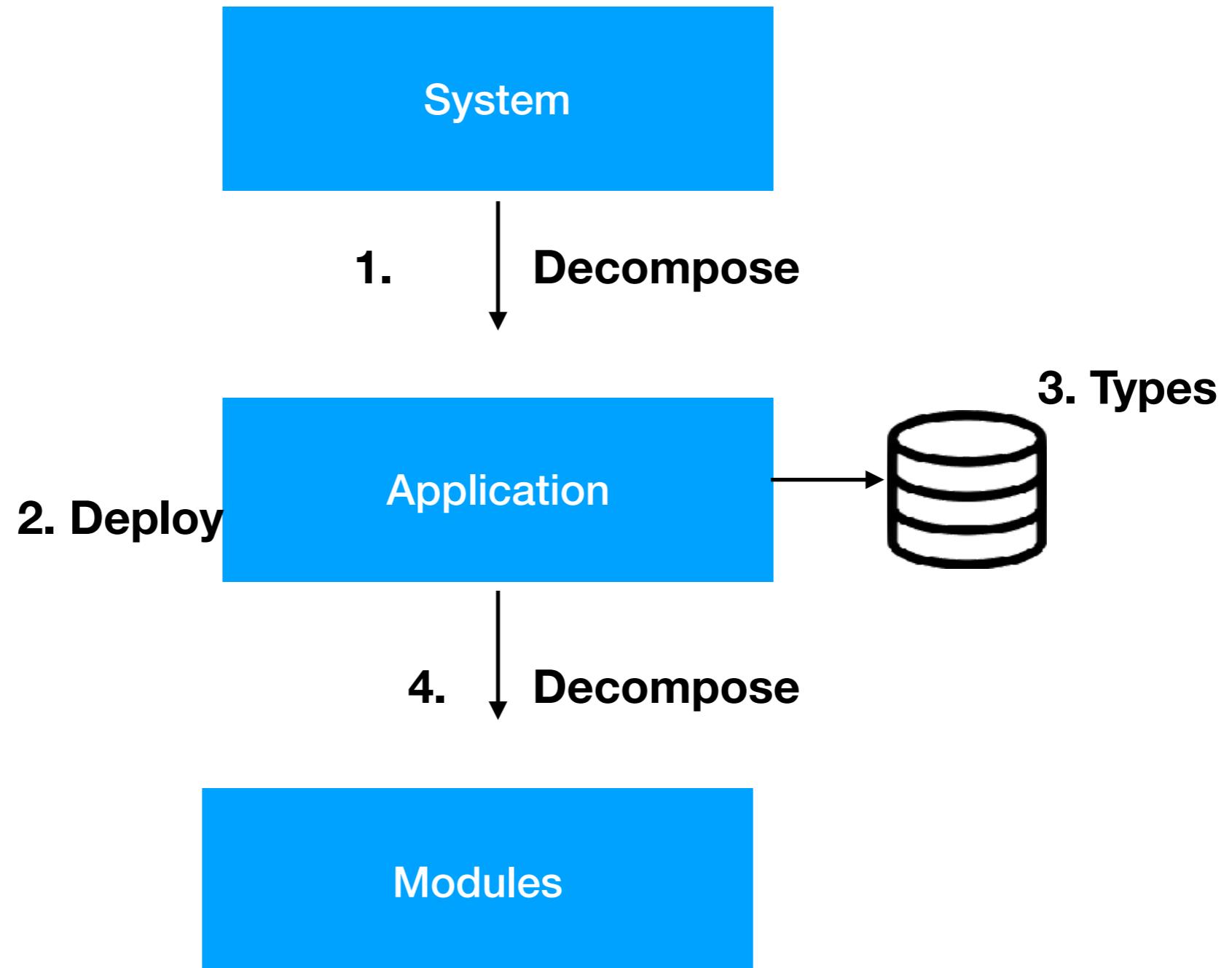


3. Query



4. Development time



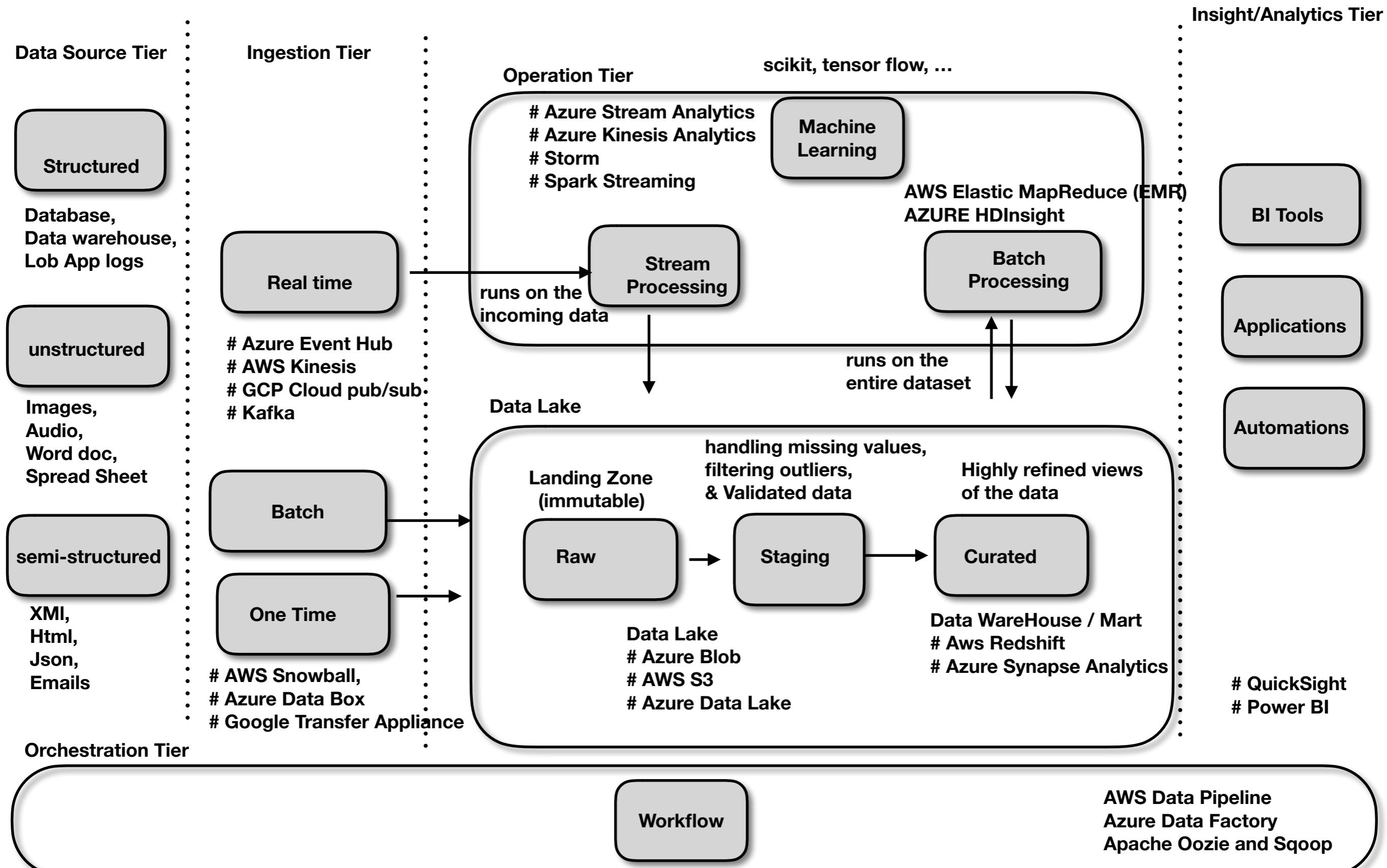


Analytics App



Reference Architecture

Analytical Application



Reference Architecture IOT Application

