

# Software Architecture



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



Mubarak

1. Arch Requirements & Doc
2. Arch Design & Doc
3. Arch Eval



# **Architecture vs Design**

Quality	Measure	Approach
# Security	# response time	# cache
# Cost	# through put	# Chunking
# Performance	# tps	# eager loading
# Availability	# latency	# lazy loading
# Reliability	# jitter	# parallel
# Robustness	...	# ...
# ...		

## Quality

## Measure/Metric

## Approach

# performance

# response time  
# latency  
# tps  
# cost

# caching  
# parallel  
# pooling  
# Lazy loading  
# virtual  
# ...

# scalability

# ?

# Maintainability

# Man month  
# Coupling  
# Complexity  
# test coverage  
# ...

# Pluggable  
# versioning  
# low coupling  
# backward compatibility  
# log  
# exception management  
# ...

<b>Quality</b>	<b>Metrics</b>	<b>Approach</b>
<ul style="list-style-type: none"><li>• Performance</li><li>• Scalability</li><li>• Availability</li><li>• Maintainability</li><li>• Security (Trust)</li><li>• Reliability (Trust)</li><li>• Robustness</li><li>• Portability</li><li>• Interoperability</li><li>• Cost</li></ul>	<ul style="list-style-type: none"><li>• Response time</li><li>• Throughput</li><li>• tps</li><li>• Latency</li><li>• Resource usage</li></ul>	<ul style="list-style-type: none"><li>• Caching</li><li>• NIO</li><li>• Compression</li><li>• JIT (Lazy loading)</li><li>• Extensibility</li><li>• Testability</li><li>• KISS</li><li>• Doc</li><li>• Observability</li><li>• ACID</li><li>• </li></ul>

**Which Quality &  
How Much ?  
(Architectural  
Requirements)**

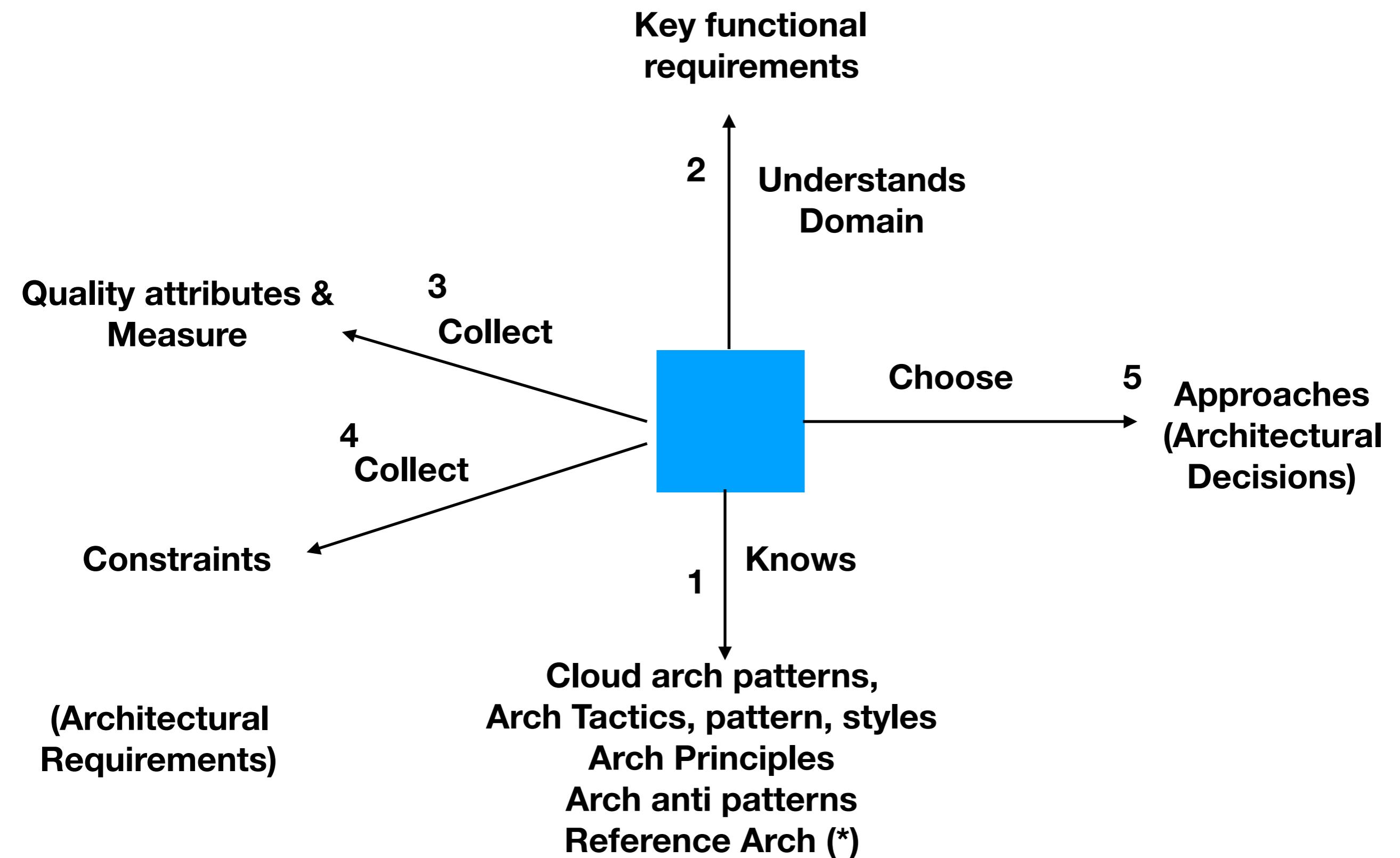
**Collect**



**Decide**

**Approaches  
(Architectural  
Decisions)**

## 6 Communicate



**Pre**

# **Engineering vs Tuning**

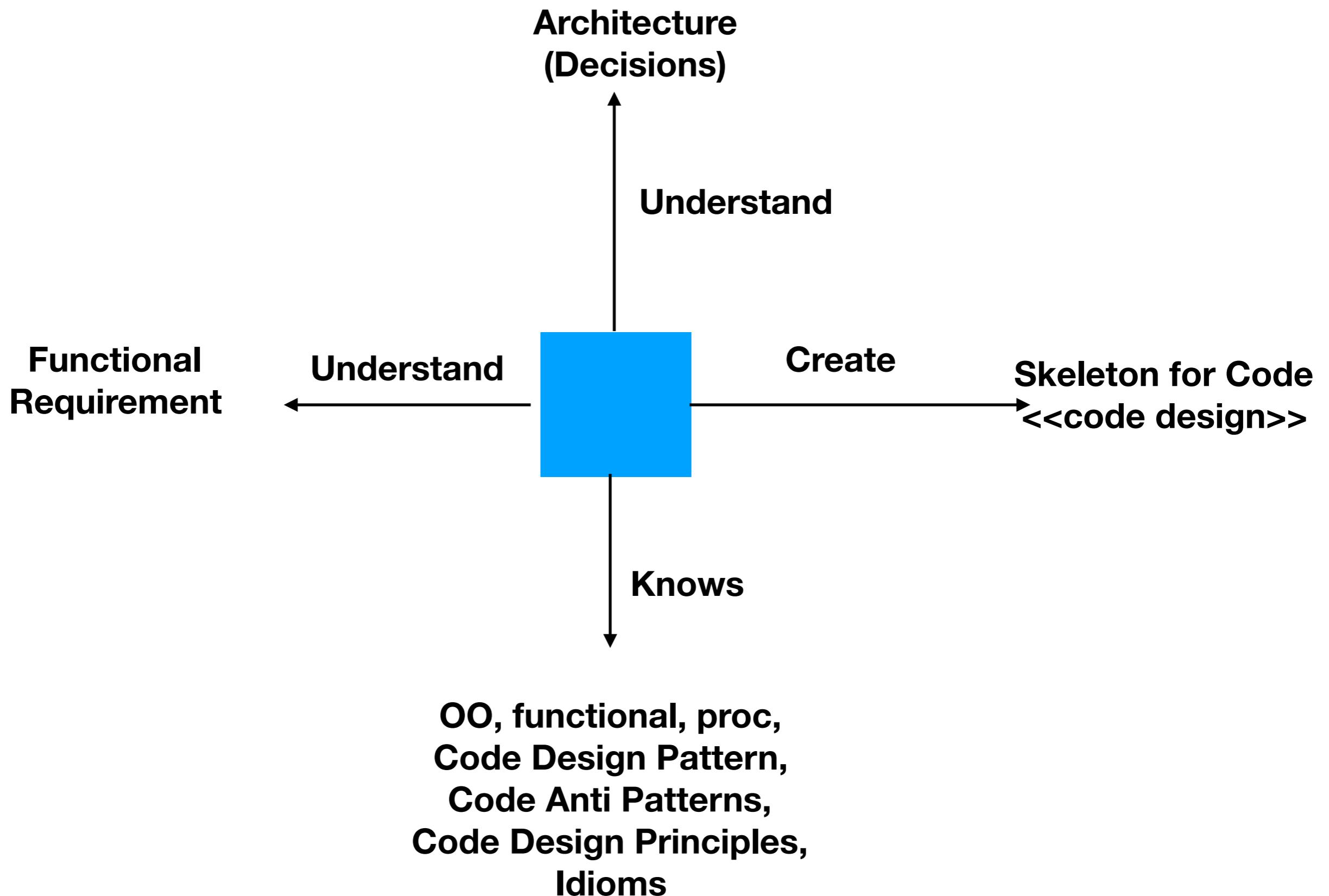
**Post**

**Performance engineering**

**Performance Tuning**

**Threat Modeling**

**Pen Testing/ Ethical Hacking**



# **Architecture vs Design**

**System quality**                            **Code Maintainability**

**Architecture Design**

**High Level Design**

**Blue Print**

**System Design**

**...**

**Detail Design**

**Module Design**

**Class Design**

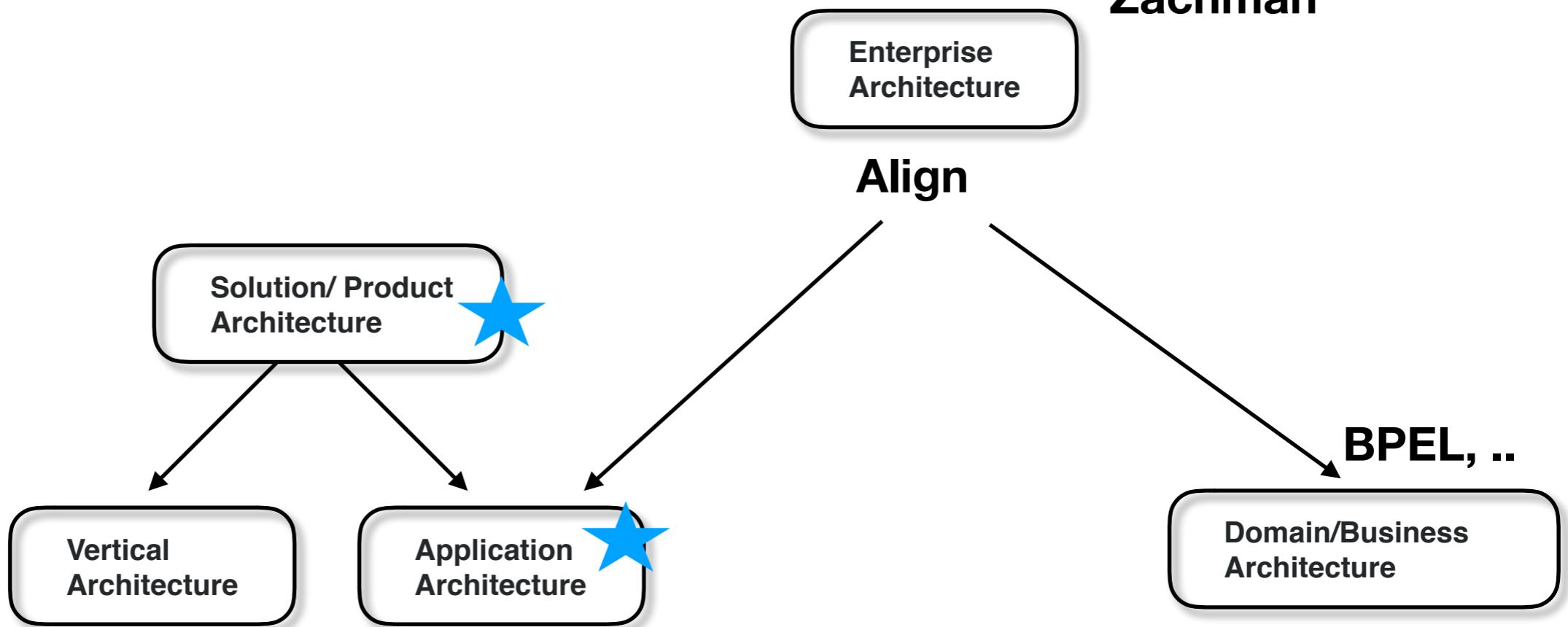
**Low Level Design**

**Code design <—**

**Implementation Design**

**...**

**TOGAF,  
DODAF,  
Zachman**



# Security  
# Infra  
# SEO  
# Data  
# Cloud  
# Technology  
# ui  
# network  
...

**Address  
quality of  
application**

**Address  
quality of  
process**

**Arch  
Requirements**

**Arch  
Design**

**Arch  
Justification**

# **Drivers      Design      Eval**

**Which Quality ?**

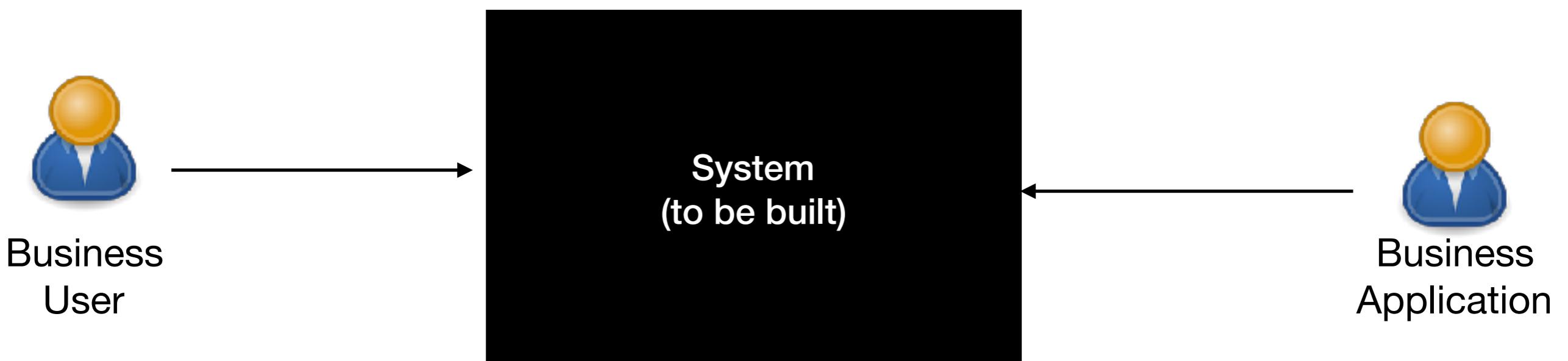
**Which Approach ?**

**Will it work ?**

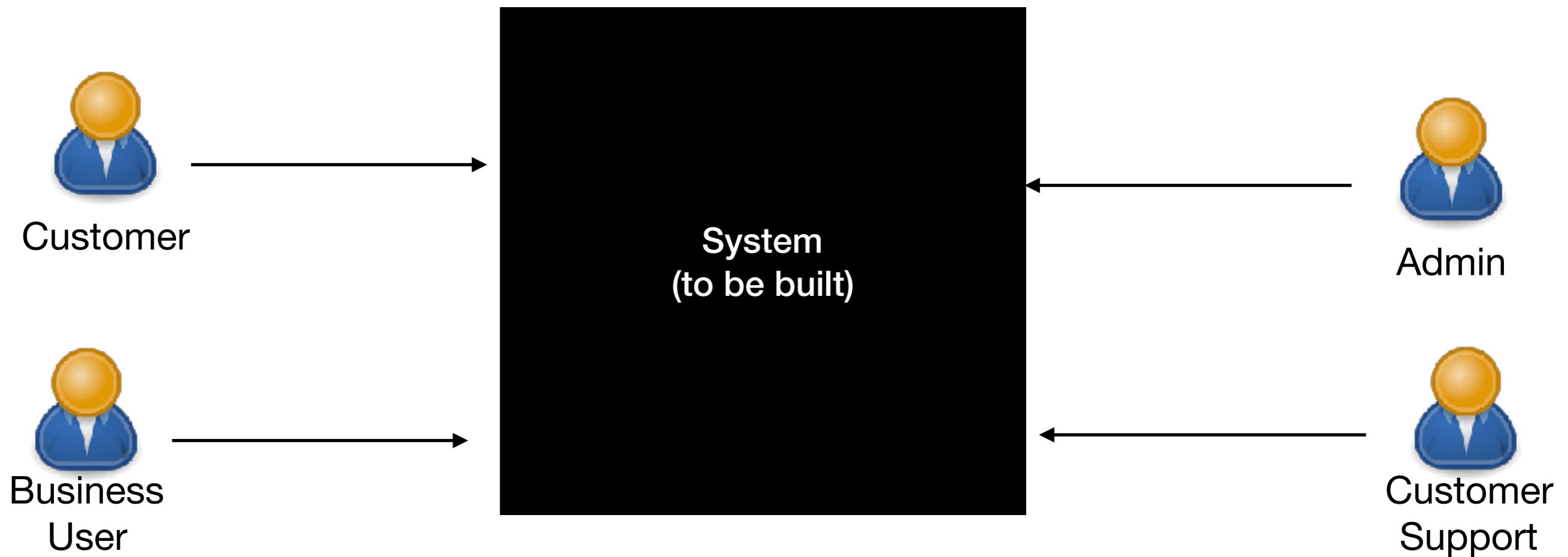
# **Architectural Requirements**

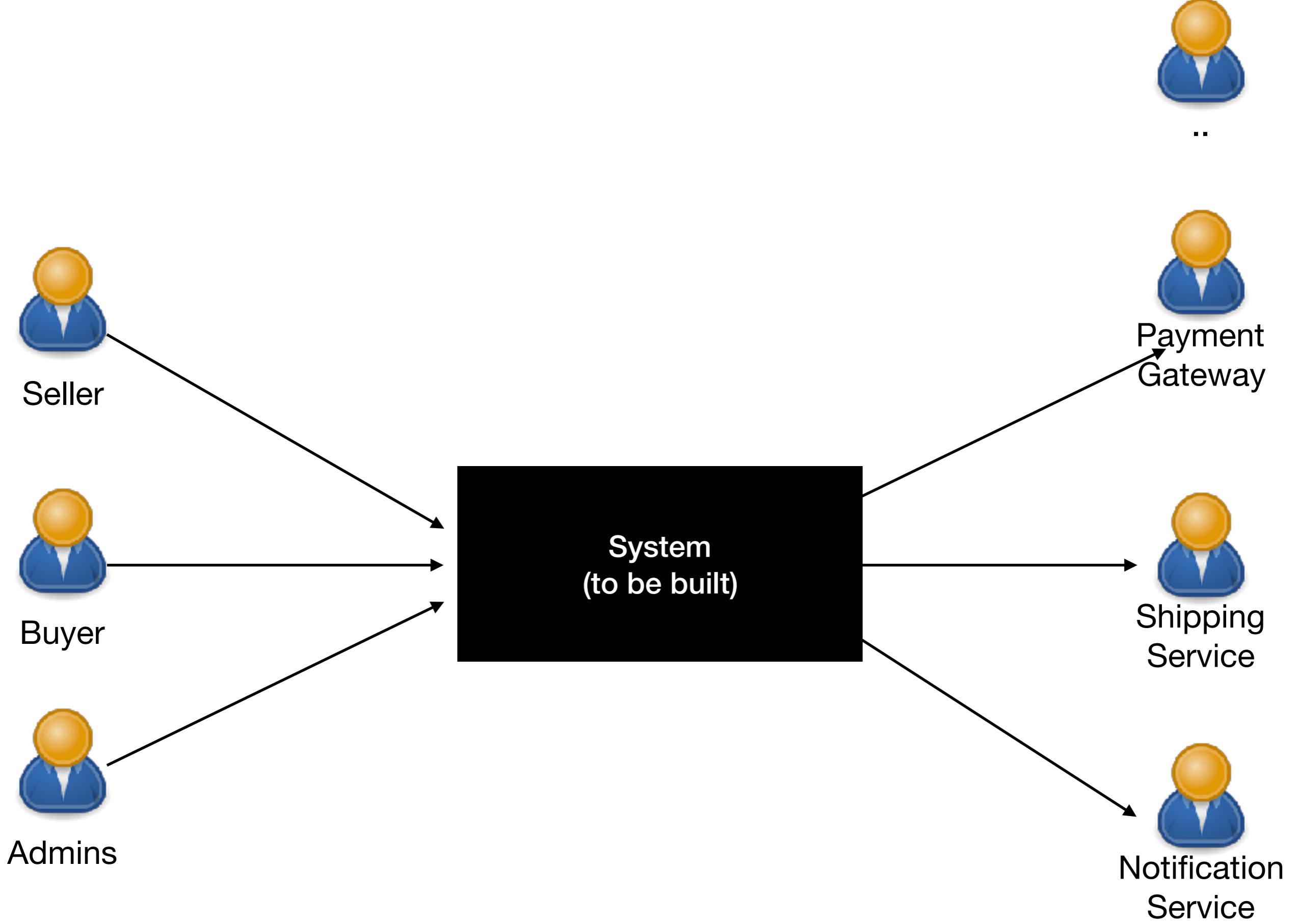
- # Context View**
- # Functional View**
- # Constraints**
- # Quality View \***
- # Assumptions**

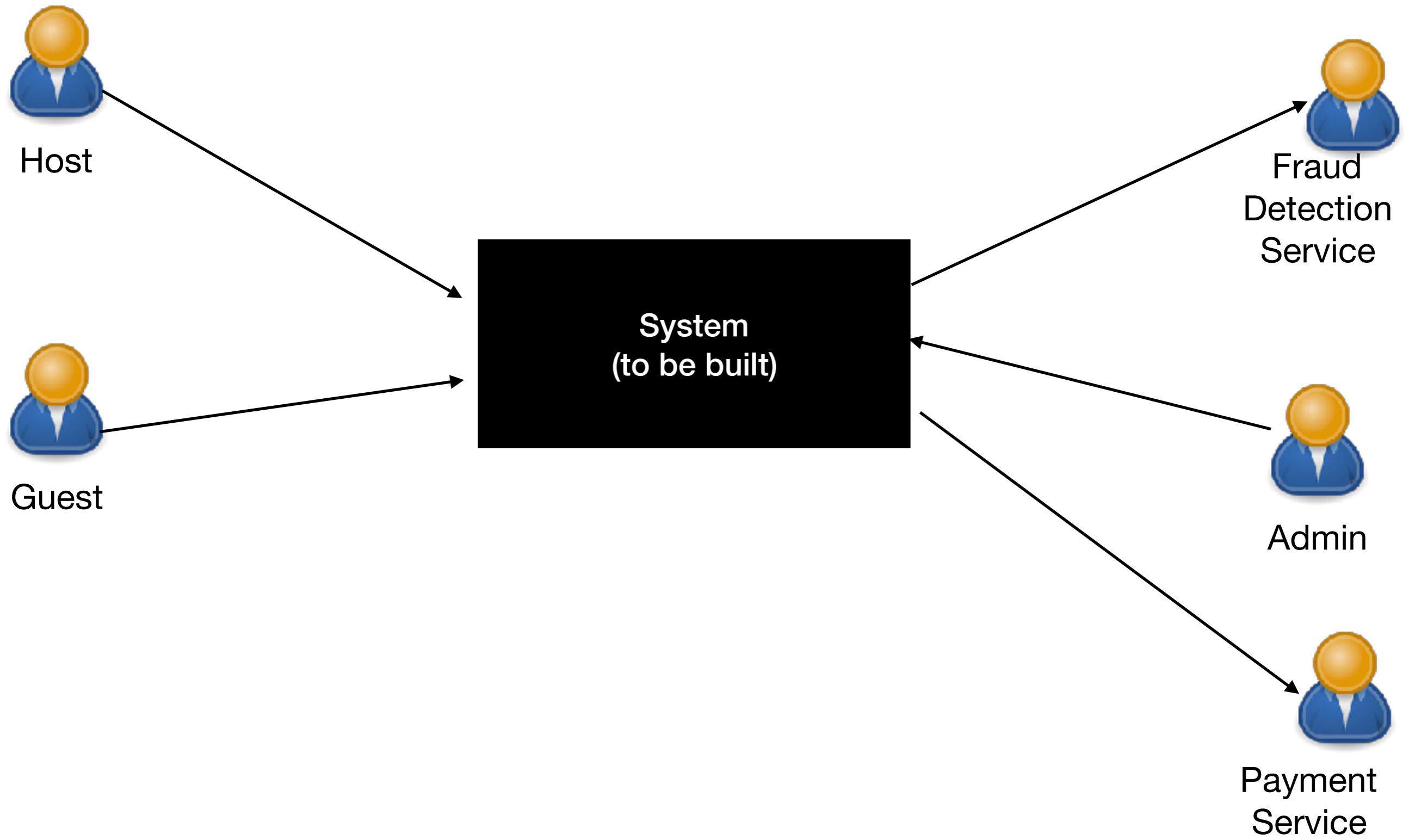
# Context View

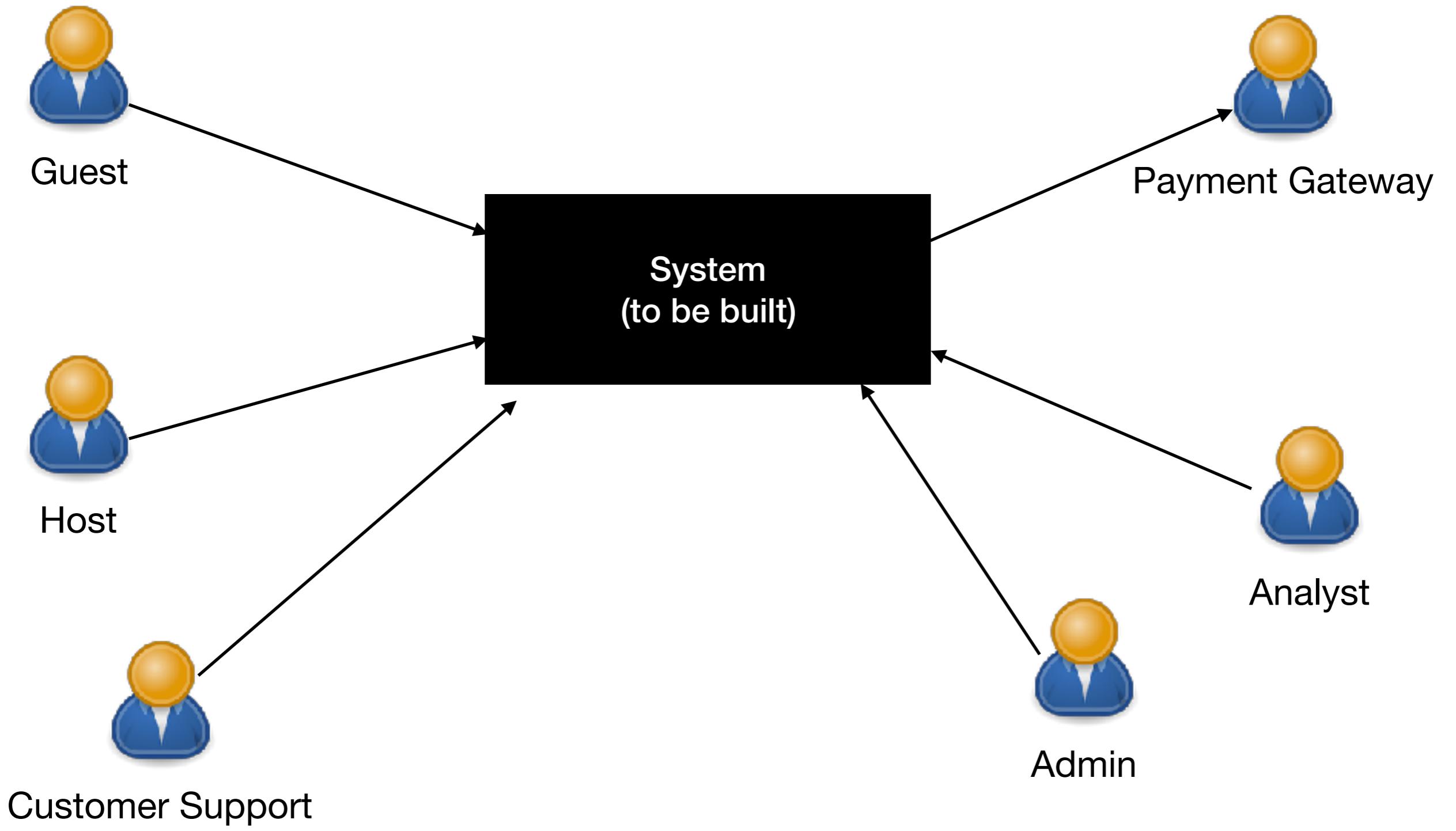


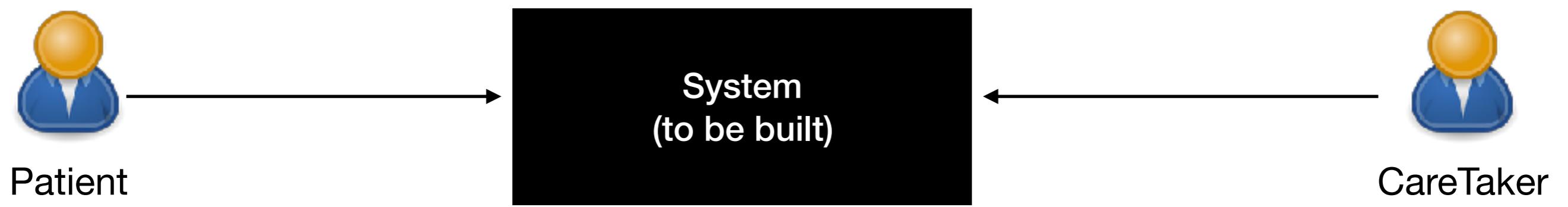
# Self Driving Car Rental

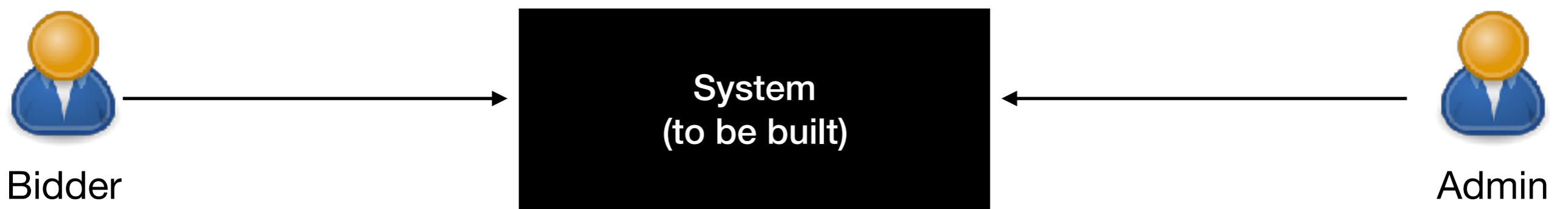


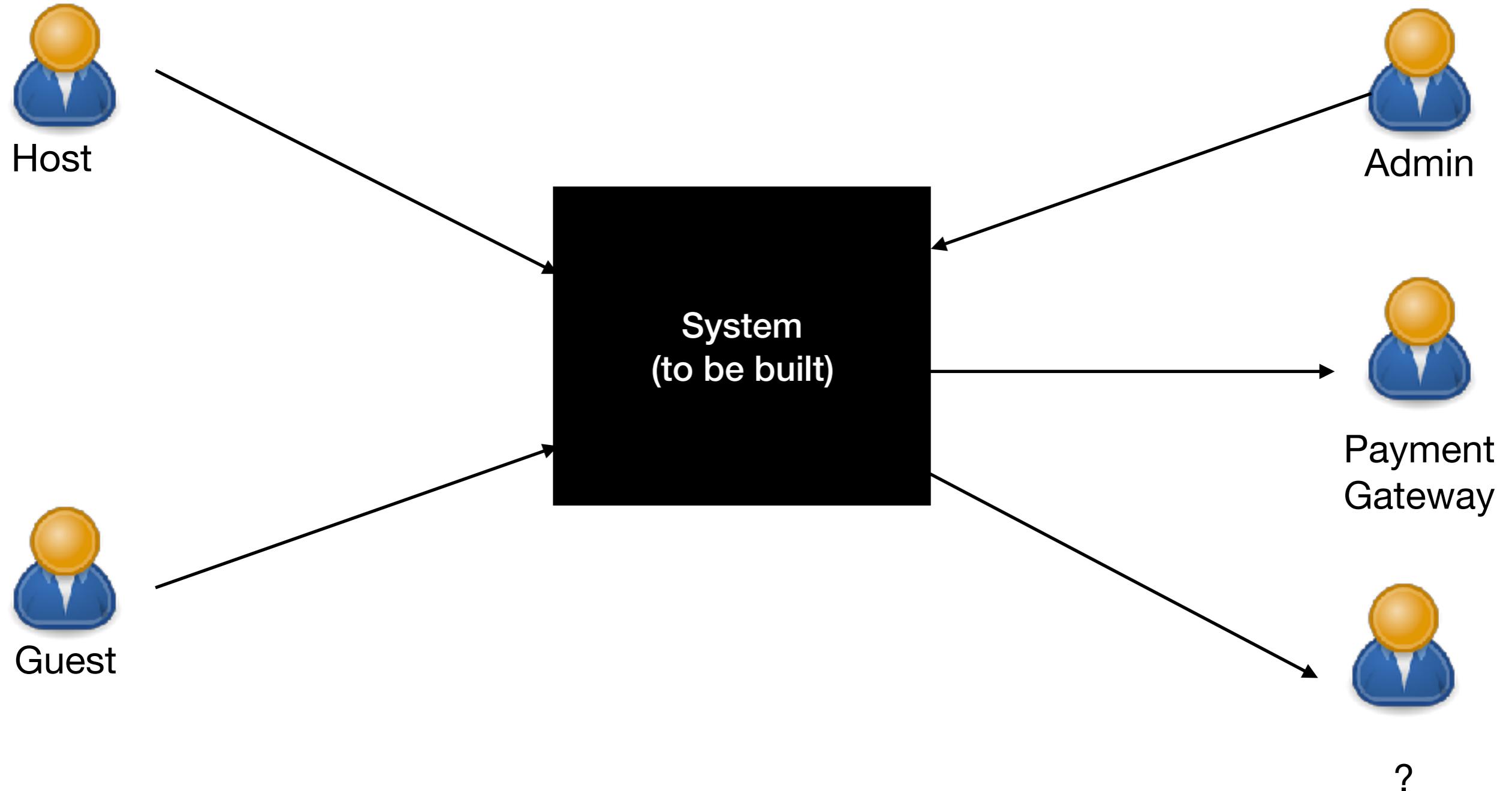




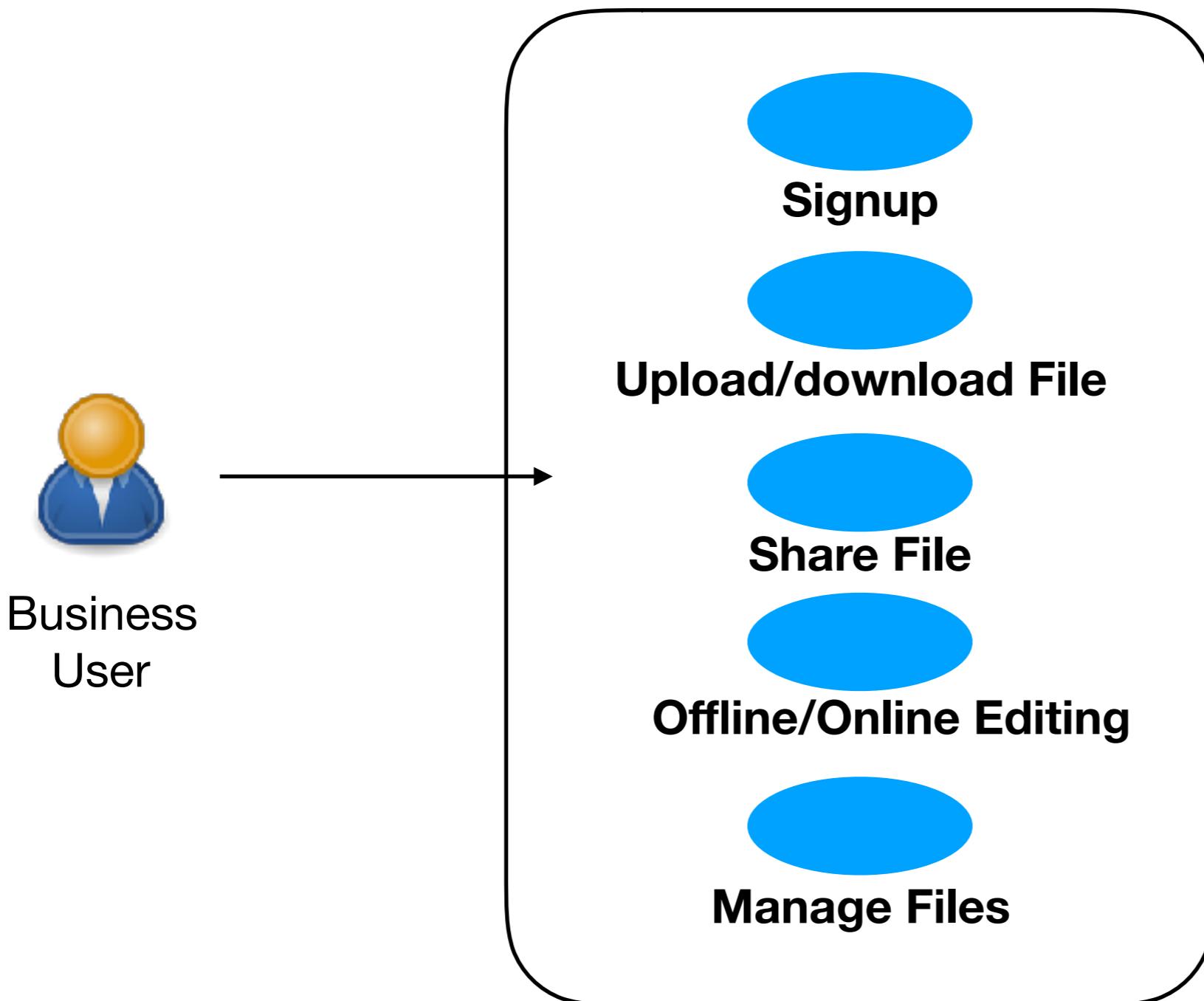




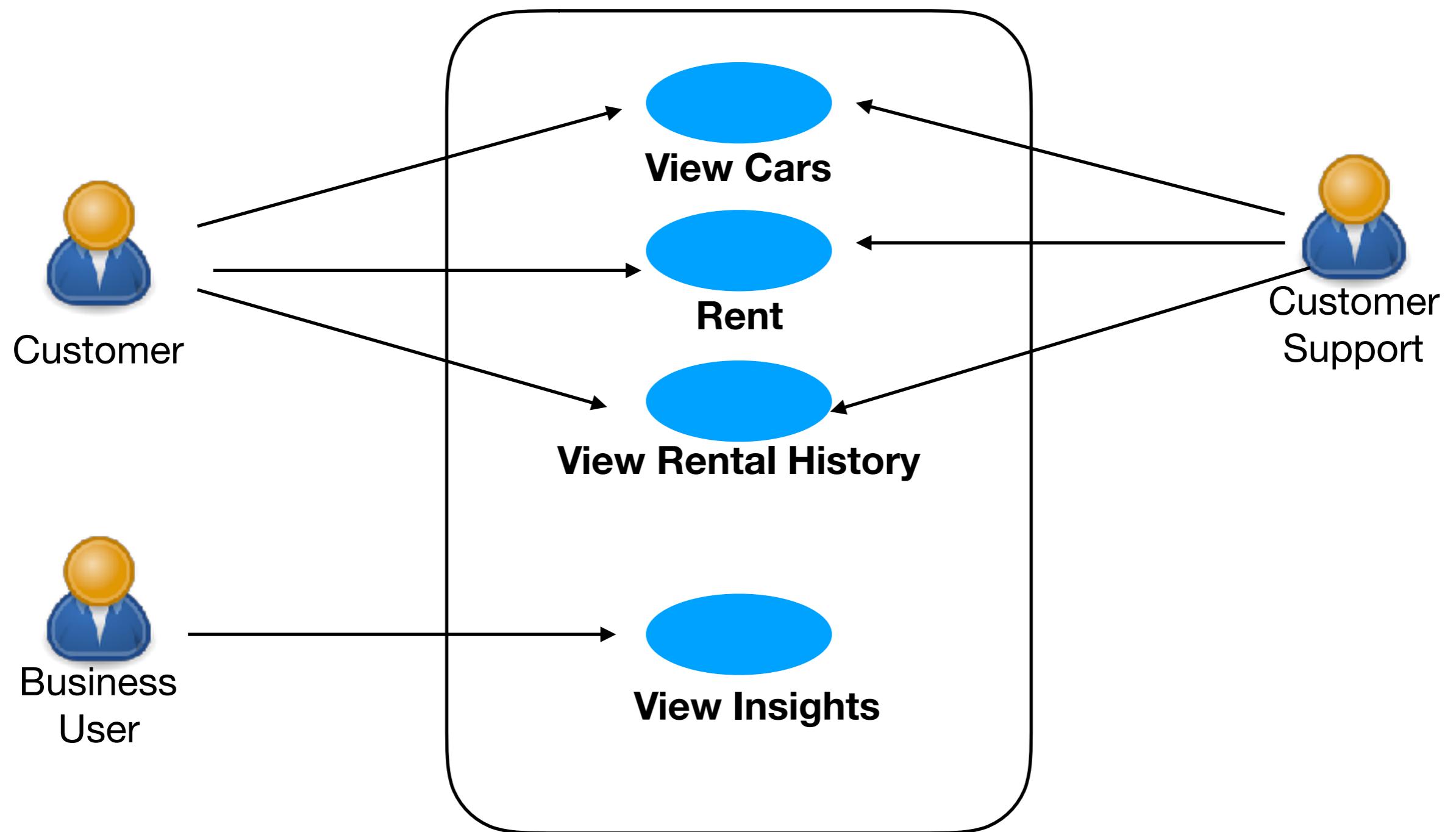


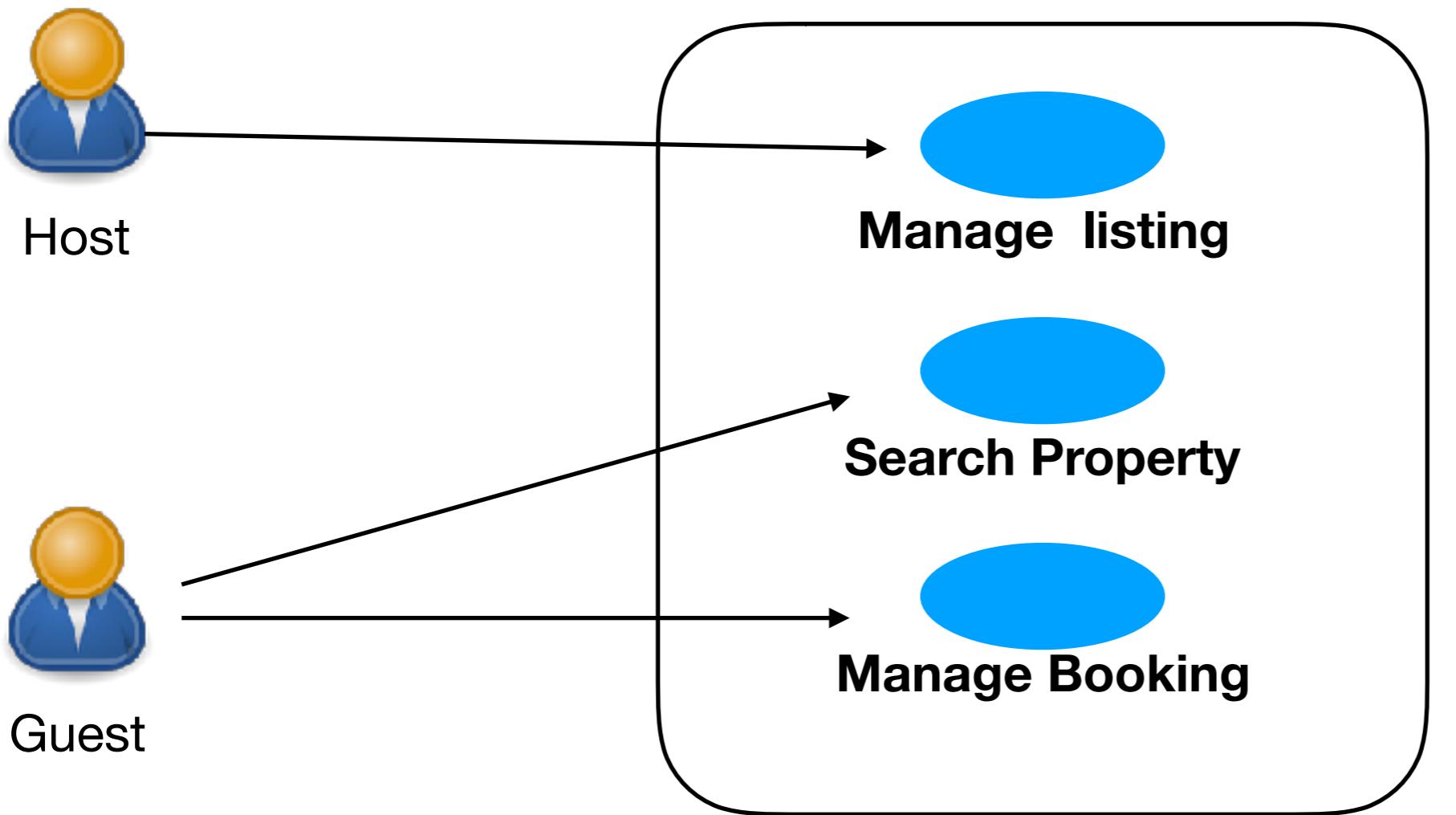


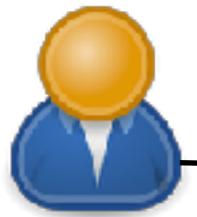
# Functional view



# Self Driving Car Rental



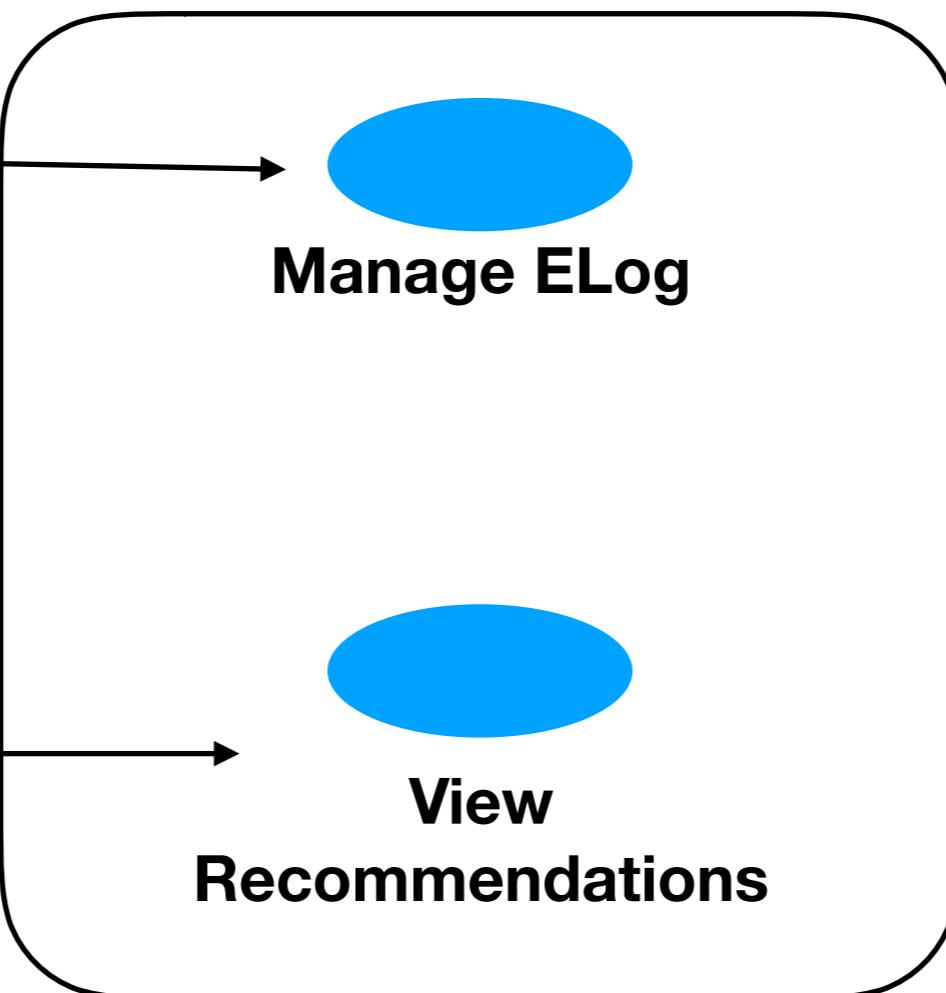


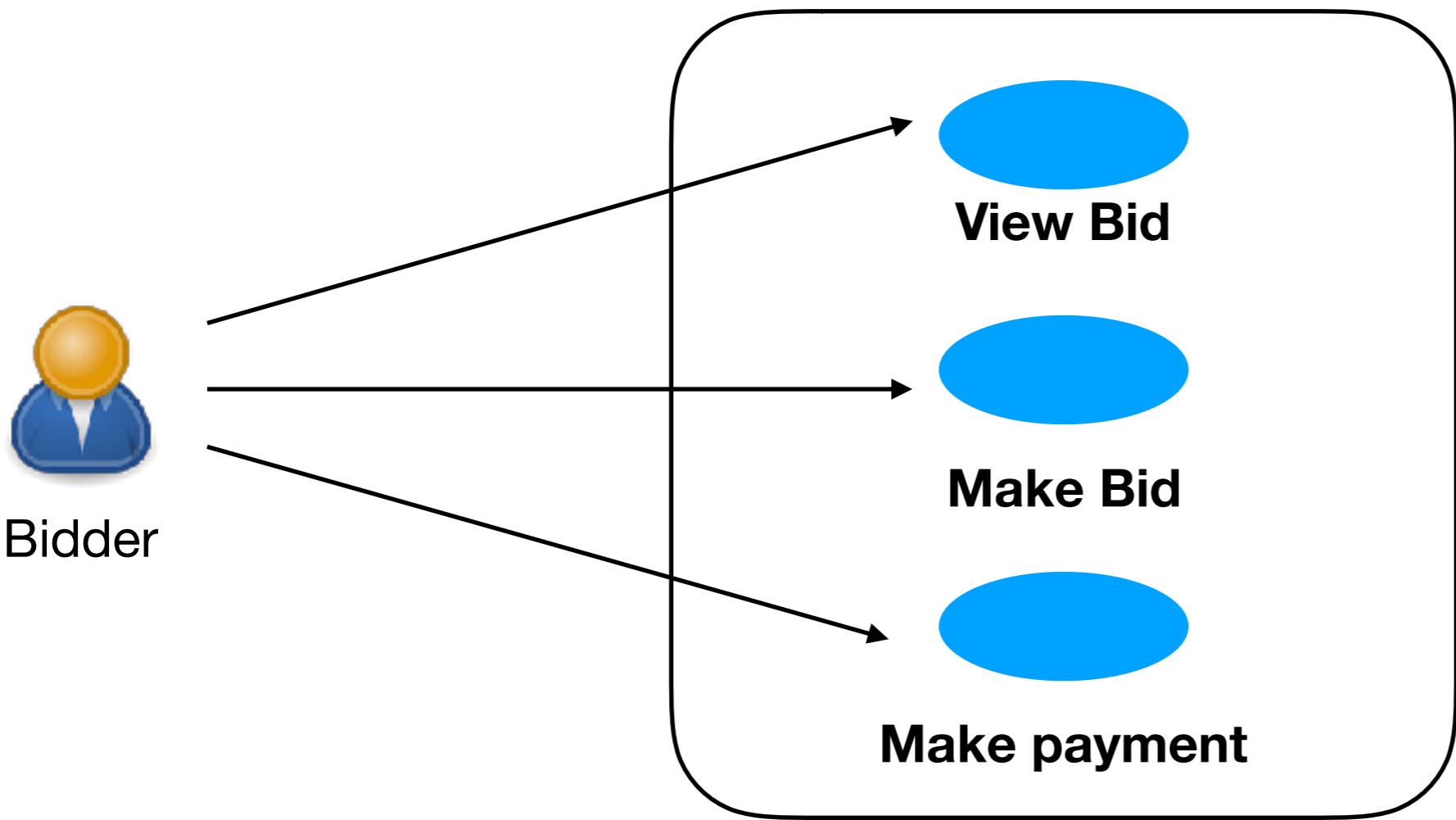


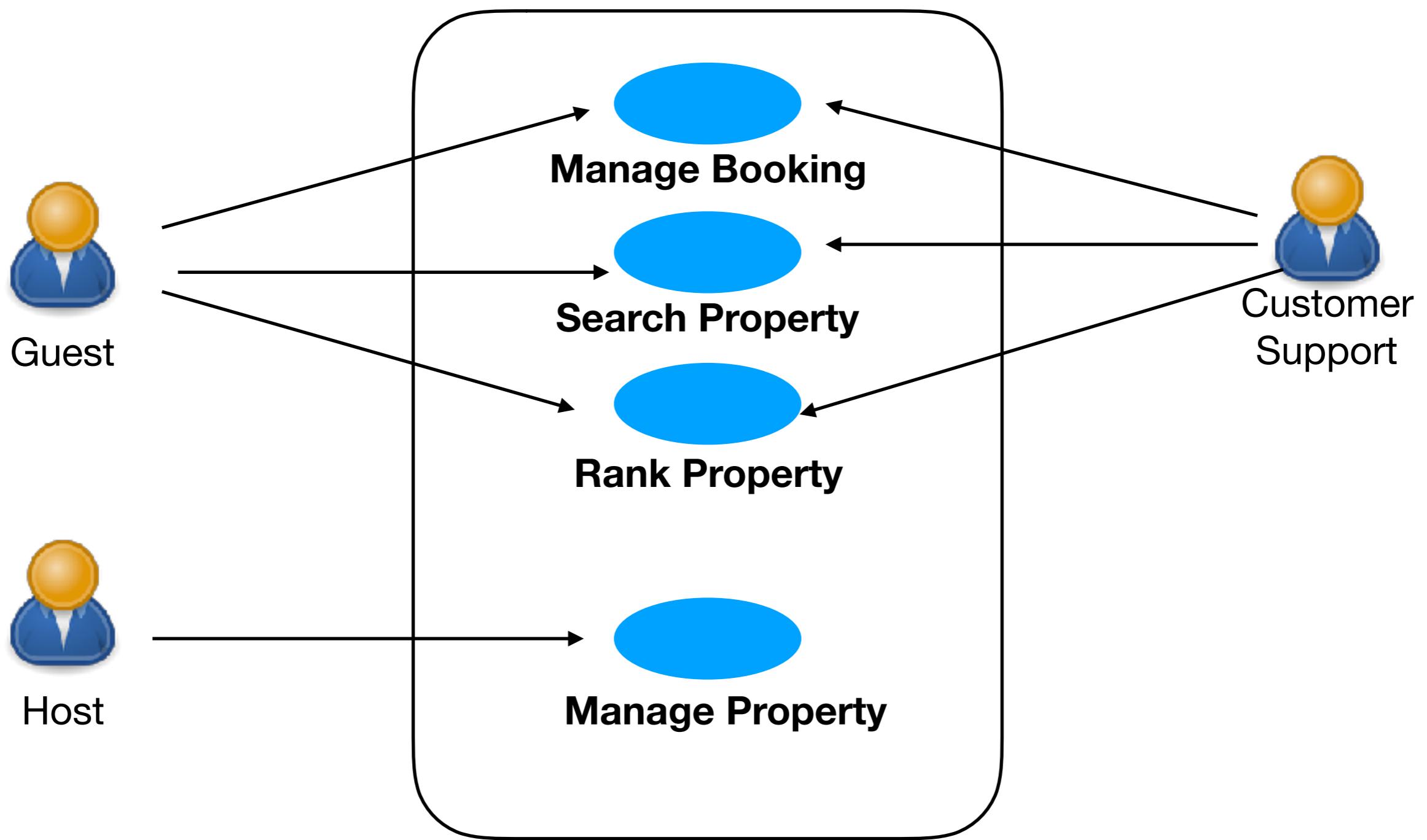
Patient



Care Taker







# Constraints view

Must support popular Desktop OS and Mobile OS

---

Should work on a low bandwidth network

---

Use Opensource only

---

Use Postgres database version xyz

---

Use GCP cloud

---

Yearly Cloud Billing should not exceed 1 million dollars - cost

---

Should be in production on April 1st 2023 - (next financial year)

---

Data should be kept for 8 years

# Self Driving Car Rental

Should work on Azure Cloud

---

GDPR Compliance

---

Should support Azure Cloud and AWS Cloud

---

GDPR Compliance

---

# Healthcare App

HIPPA Compliant

---

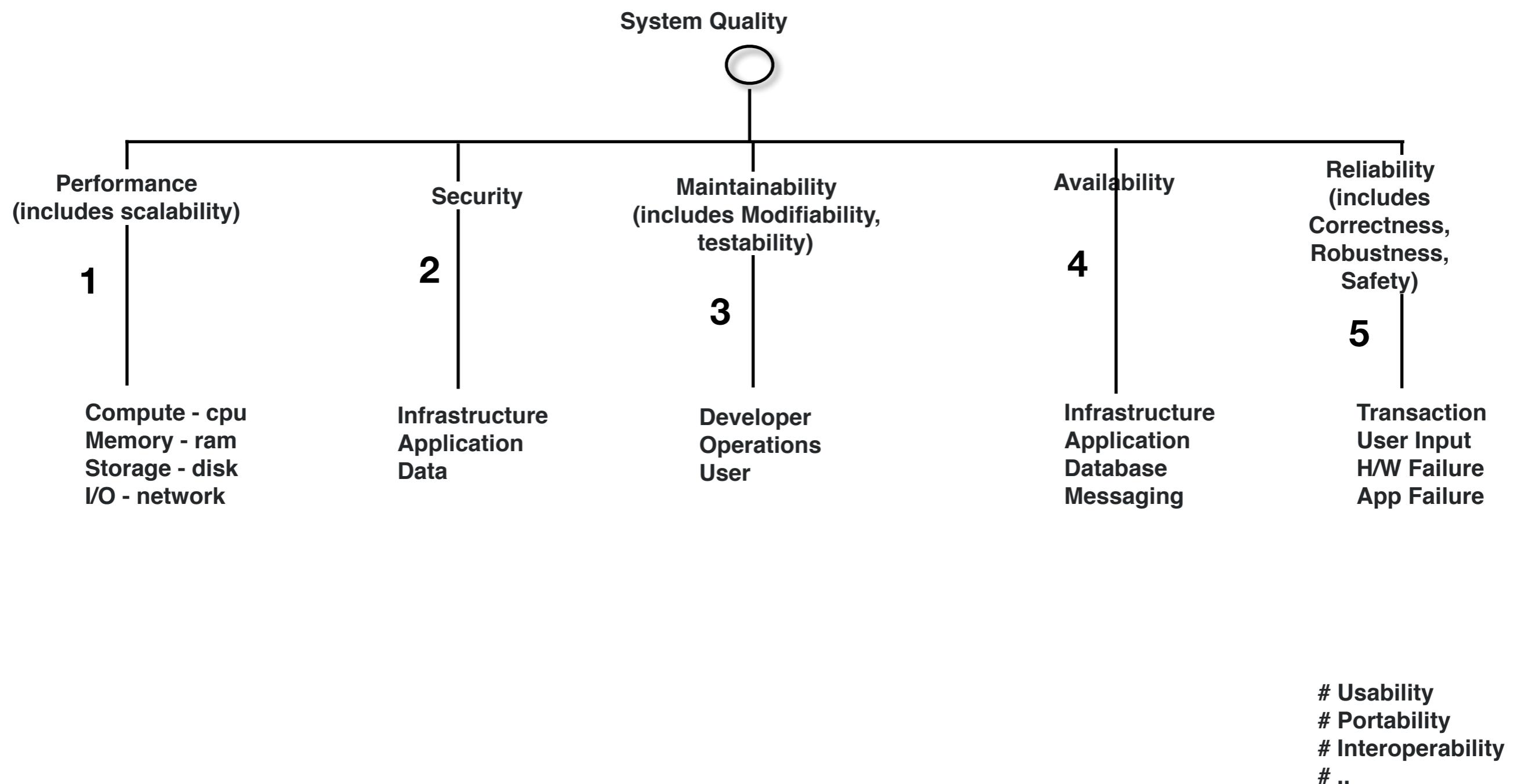
---

Should support IE 11

---

---

# Quality View



# Seed Quality Scenarios

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a user	I want to upload a 2 MB file	From Mobile Client	During Peak Load	The File is uploaded	In < 3 secs
Maintainability	Developer	Want to replace the storage	In the cloud API	During maintenance	The storage is replaced	In < 20 man days
Security	unknown identity	requests to download a file	In the portal	During Normal load.	block access to the data and record the access attempts	100% probability of detecting the attack, 100% probability of denying access
Availability	The Database	Failed	In the Data Centre	During Operational Hours	Secondary is made Primary	In < 2 minutes

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Reliability	Consumer Web site	sent a purchase order request	to the XYZ Application	duplicate request	consumer is not double-charged, data remains in a consistent state	

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Availability	Processor	Stops working	Central System	Peak hours	Notify Provide degraded mode	< 5 min degraded mode
Reliability	Consumer WebSite	Send purchase order request	To Xyz App	Duplicate Request	the consumer is not double-charged and one POS is placed.	99% of the time
Maintainability	New Display System	Is Installed	In the Bus Stop	Off Peak hours	The Display system starts showing bus arrival information	within 30 minutes

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a Host	I should be able to register a property	In the portal	During business hours	The property is registered	In < 2 sec
	As a guest	I want to find a property	In the portal	During peak load	The listing is shown to the user	In < 1 sec

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a Bidder	I should be able to see the Bids placed by other bidders	In the portal	When there are 100,000 bidders bidding	The Highest bid is shown to the bidder	In < 1 sec

# Assumptions View

Will use open source Technologies only

---

For offline scenarios in native app or any of services going down in backend last available timeline will be visible.

---

System will be eventually consistent.

---

# QAW

- Stake holders, dev, ops, QA, BA, ...
- half day
- Prepare Seed scenarios(lots example scenarios \*)
- Collect scenarios
-

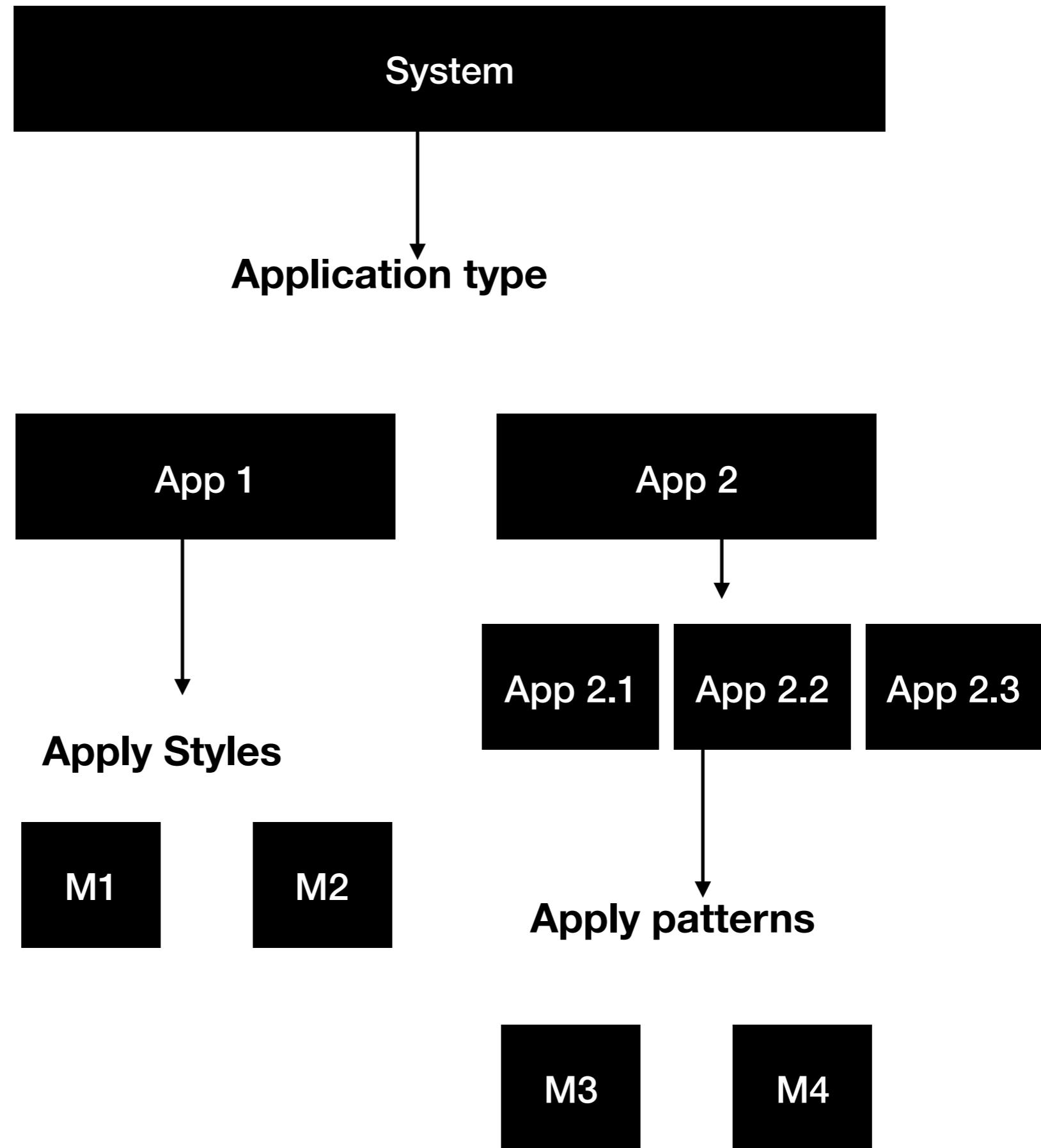
# **Drivers      Design      Eval**

**# Context View**  
**# Functional View**  
**# Constraints**  
**# Quality View \***

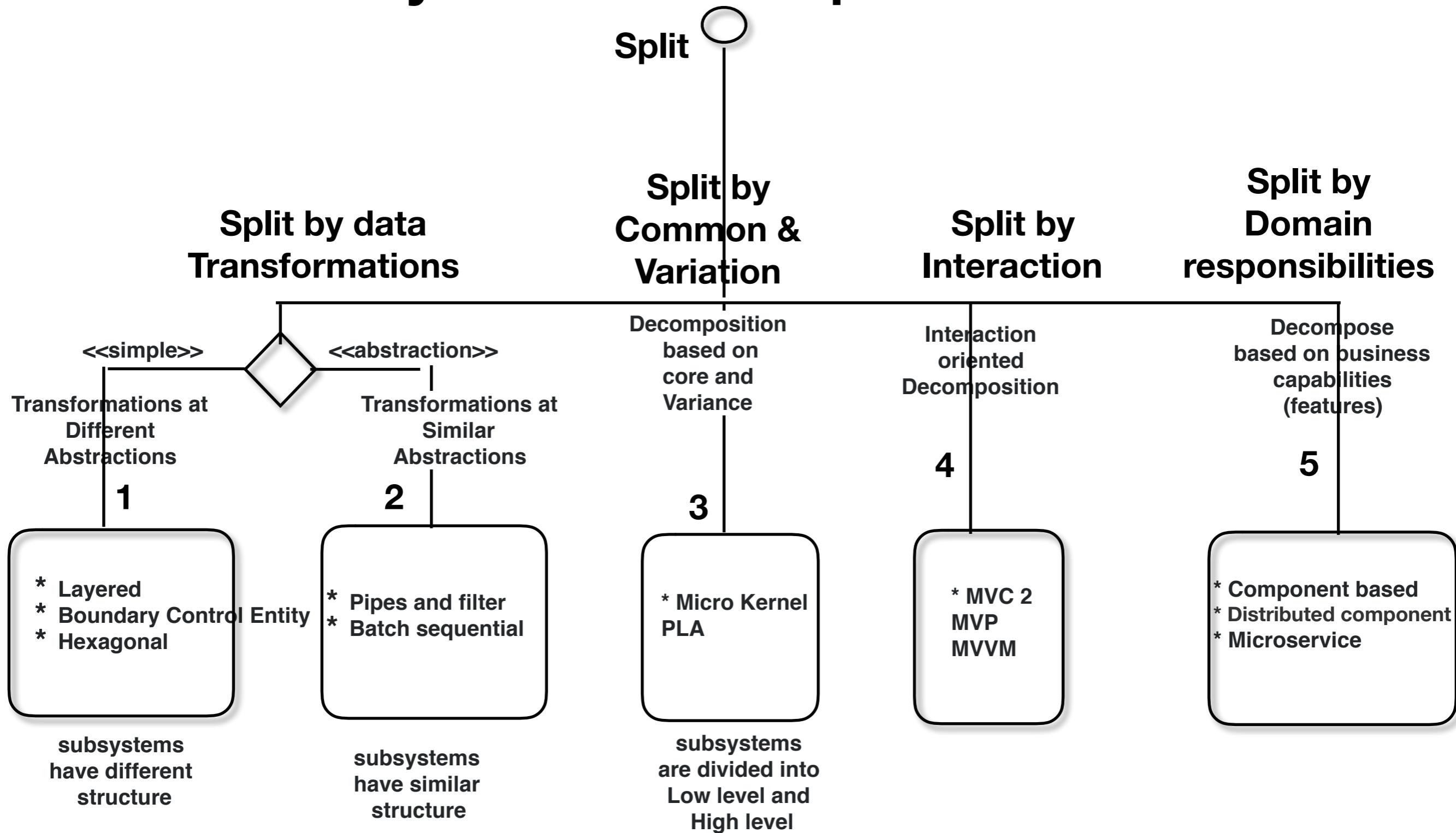
**# Logical View**  
**# Security View**  
**# Deployment View**

**# Justification**

# **Logical View**



# Choose System Decomposition Patterns





**Image**

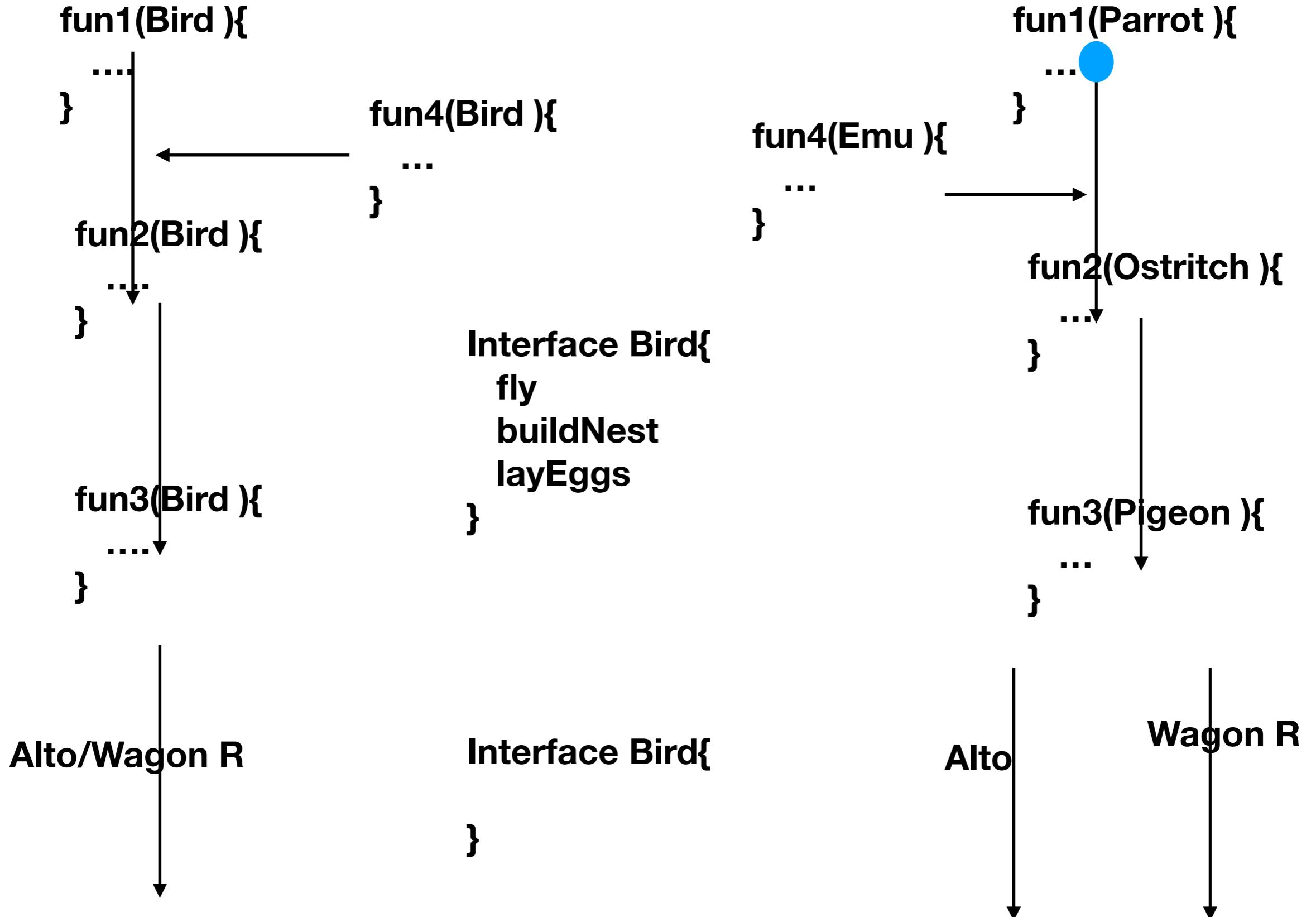
**Remove noise**

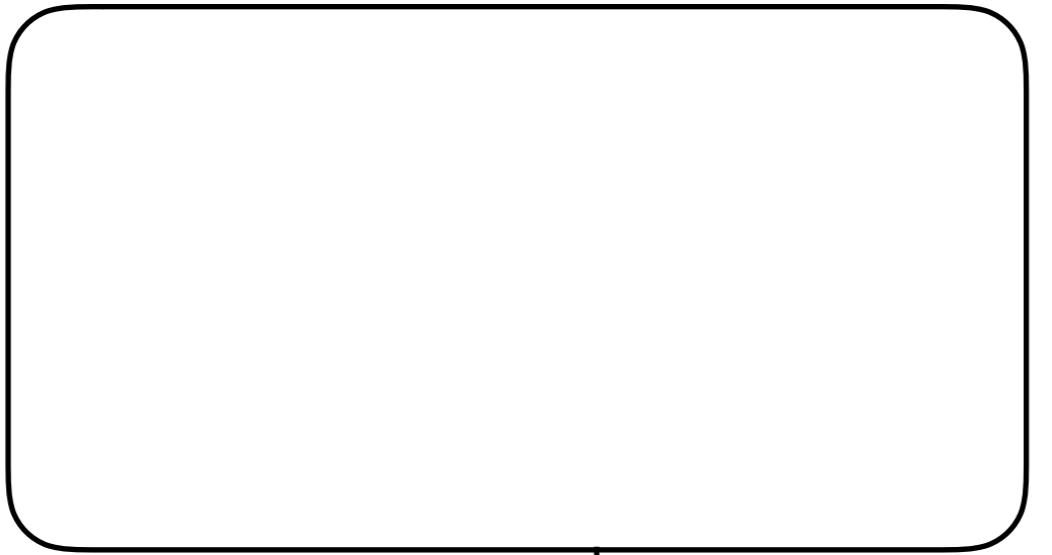
**Get contours**

...

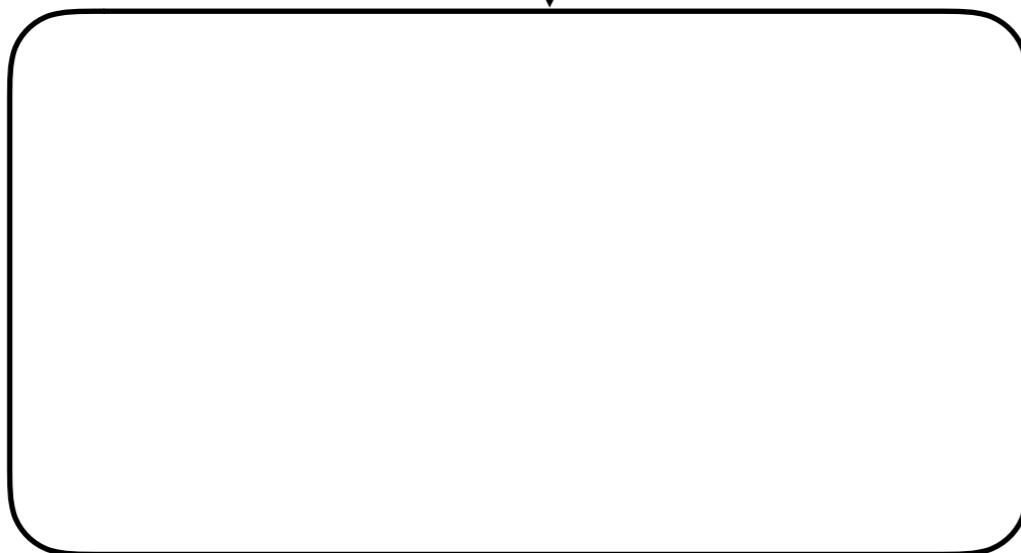
**Get Text**

## Bird vs Parrot





**Change data**



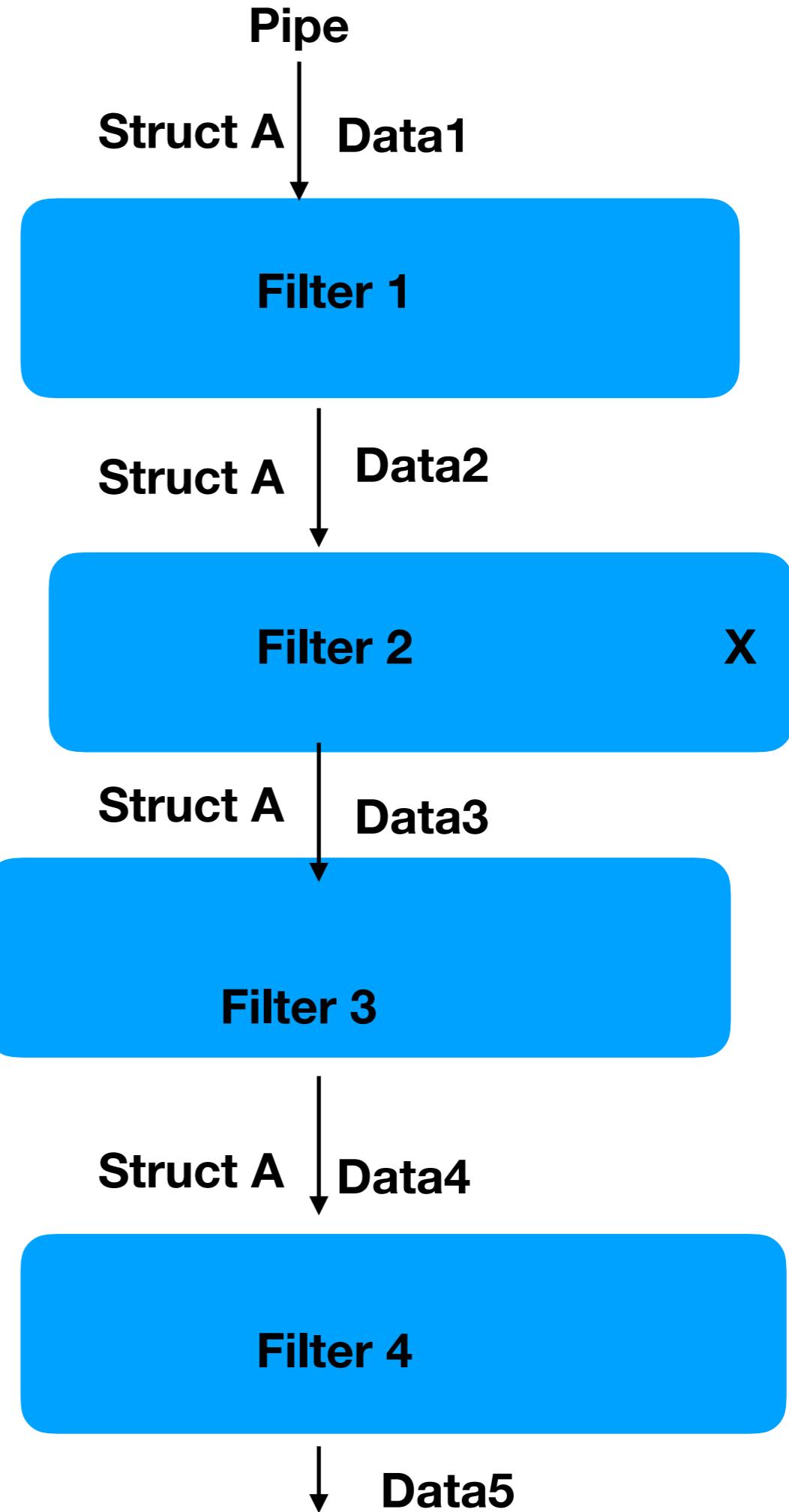
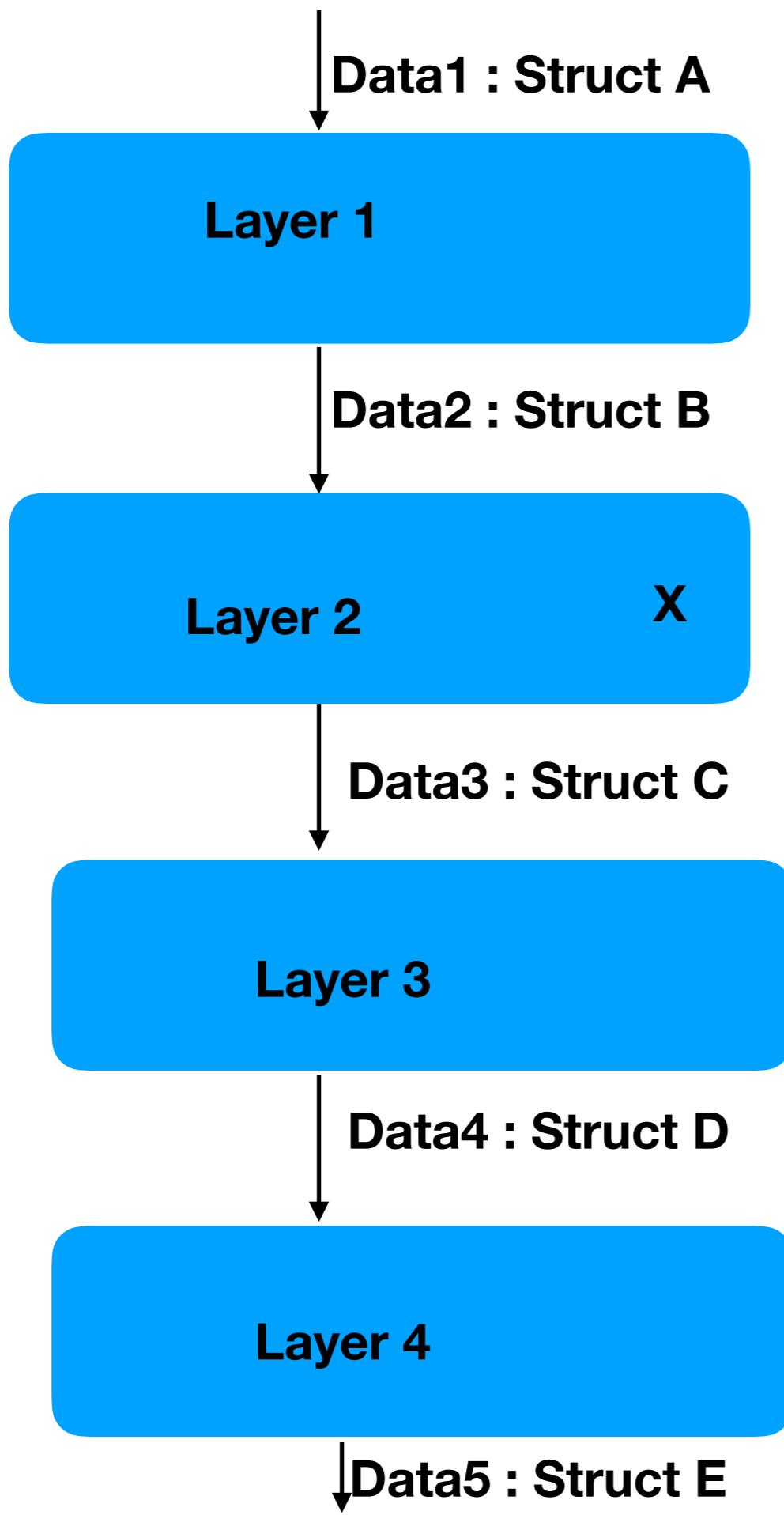
**Change data**



```
class Parrot implement FlyingBird{
    Fly
    Speak
    Eat
}
```

```
Interface LlvingBeing{
    Eat
    Sleep
}
Interface Bird extends LlvingBeing{
    ?
}
```

```
Interface FlyingBird extend Bird{
    Fly
}
```

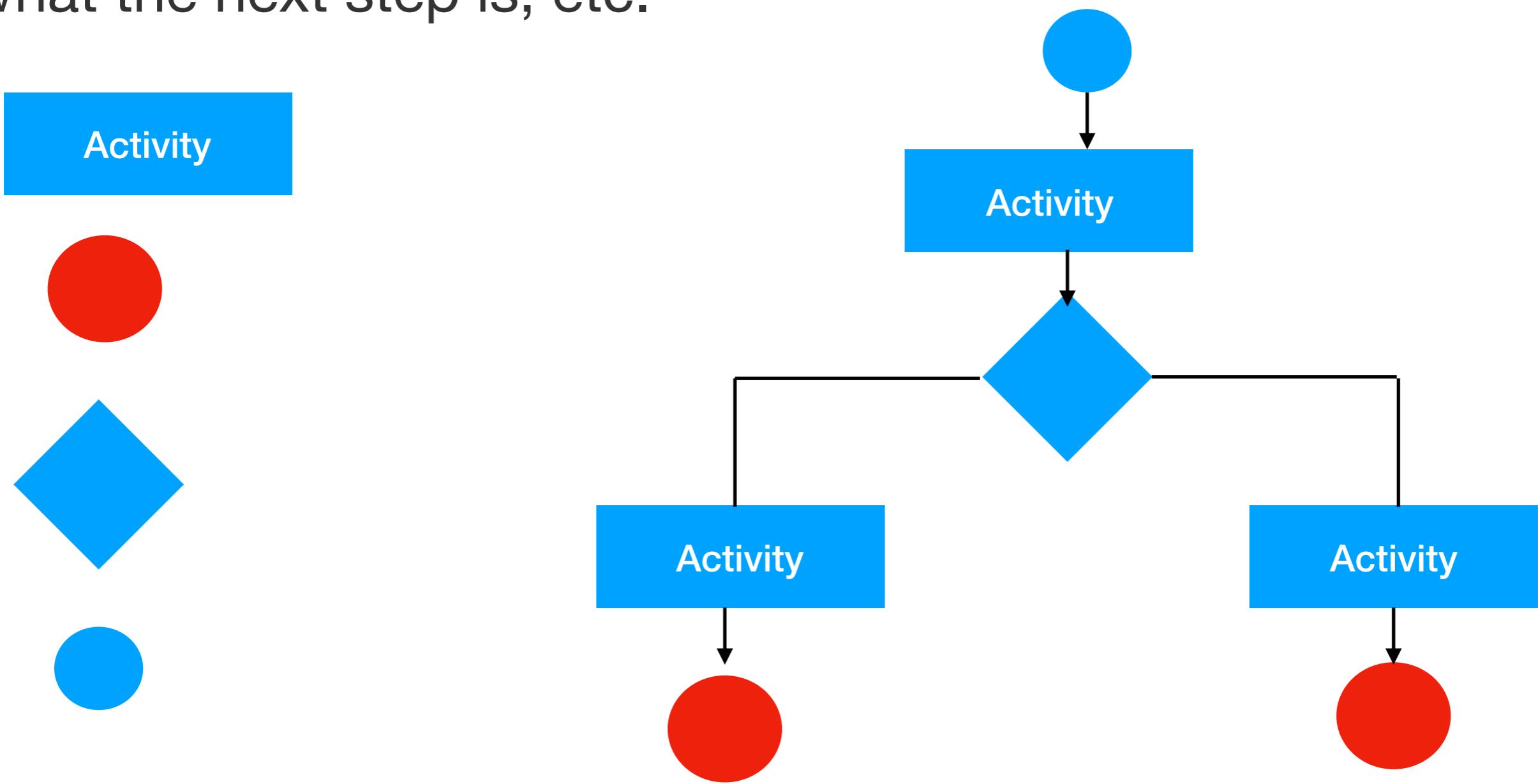


Eclipse IDE. Downloading the basic Eclipse product provides you little more than a fancy editor. However, once you start adding plug-ins, it becomes a highly customizable and useful product.



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.

A workflow implementation. The implementation of a workflow contains concepts like the order of the different steps, evaluating the results of steps, deciding what the next step is, etc.



A task scheduler. A scheduler contains all the logic for scheduling and triggering tasks

Internet browsers plug-ins add additional capabilities that are not otherwise found in the basic browser

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.

Presentation Layer

Component

Component

Component

Business Layer

Component

Component

Component

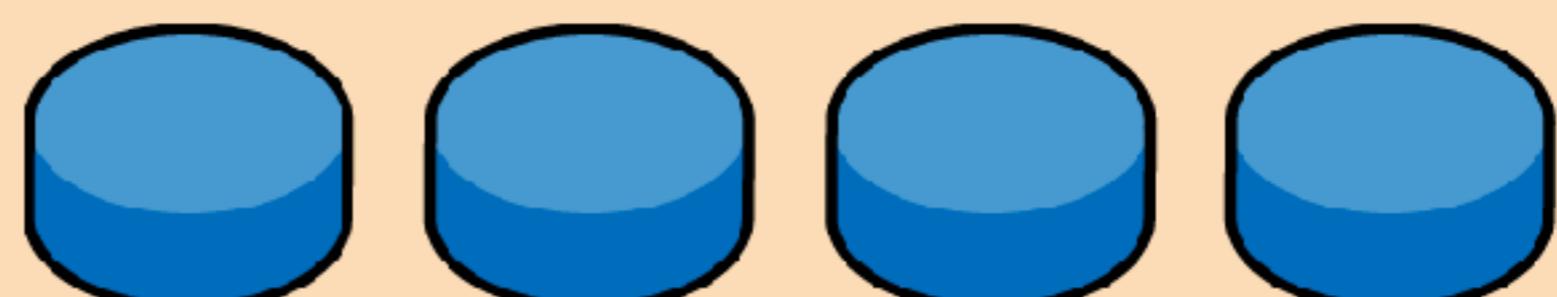
Persistence Layer

Component

Component

Component

Database Layer





**Api Layer**

**Domain Layer**

**Data Layer**

**Api Layer**

**Domain Layer**

**Invoice**

**Account**

**...**

**Data Layer**

### **Invoice**

**Api Layer**

**Domain Layer**

**Data Layer**

### **Account**

**Api Layer**

**Domain Layer**

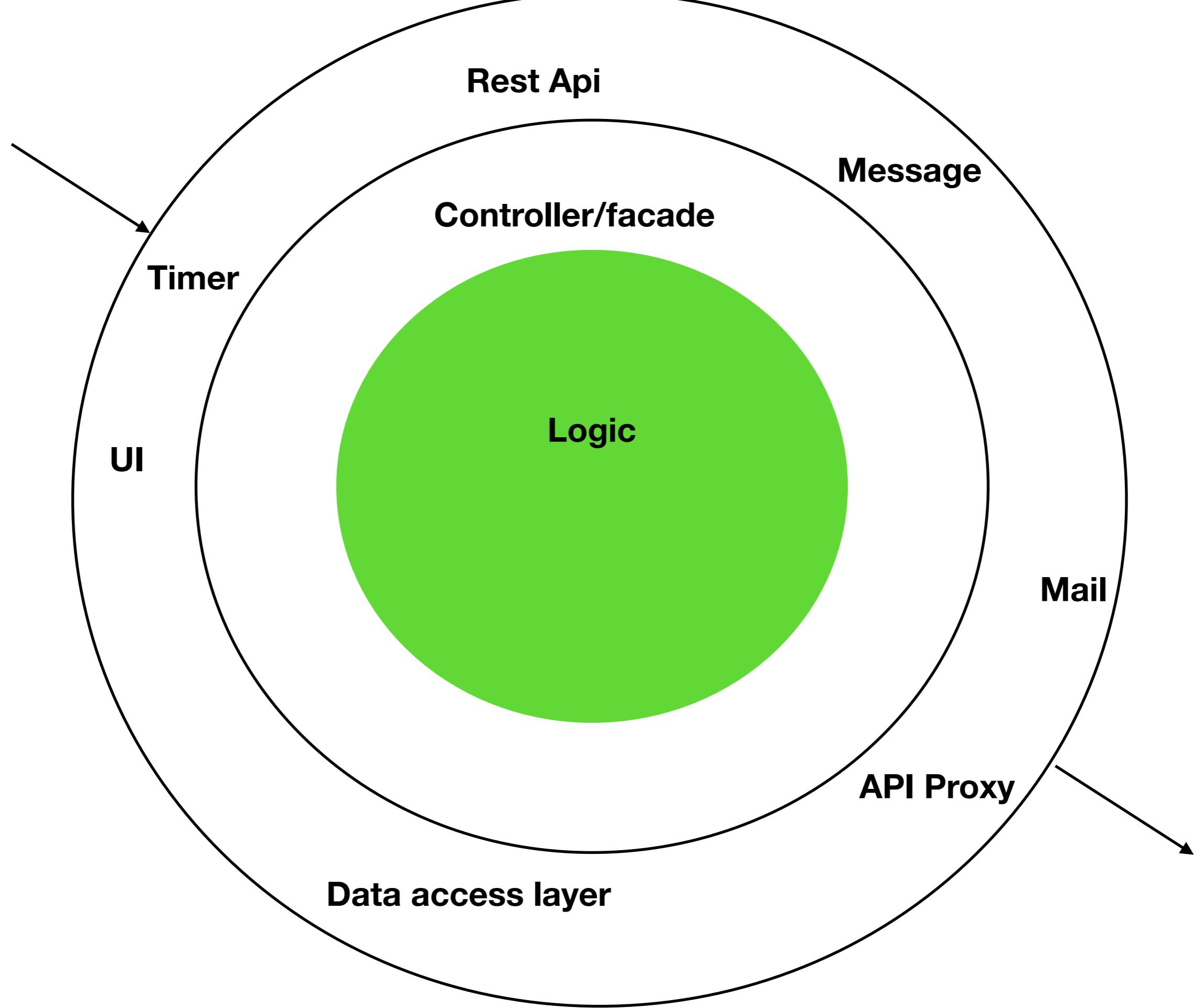
**Data Layer**

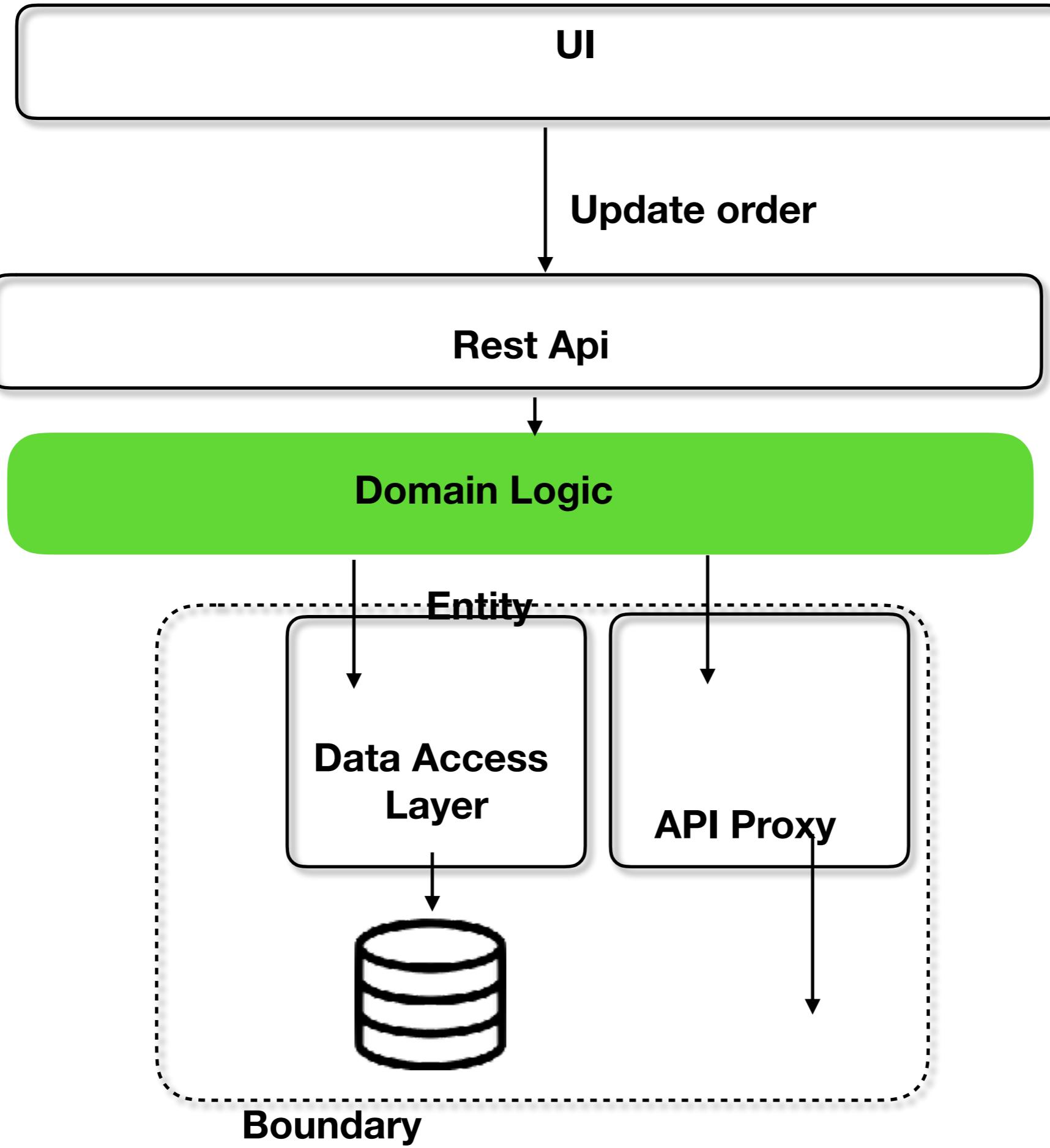
**...**

**Api Layer**

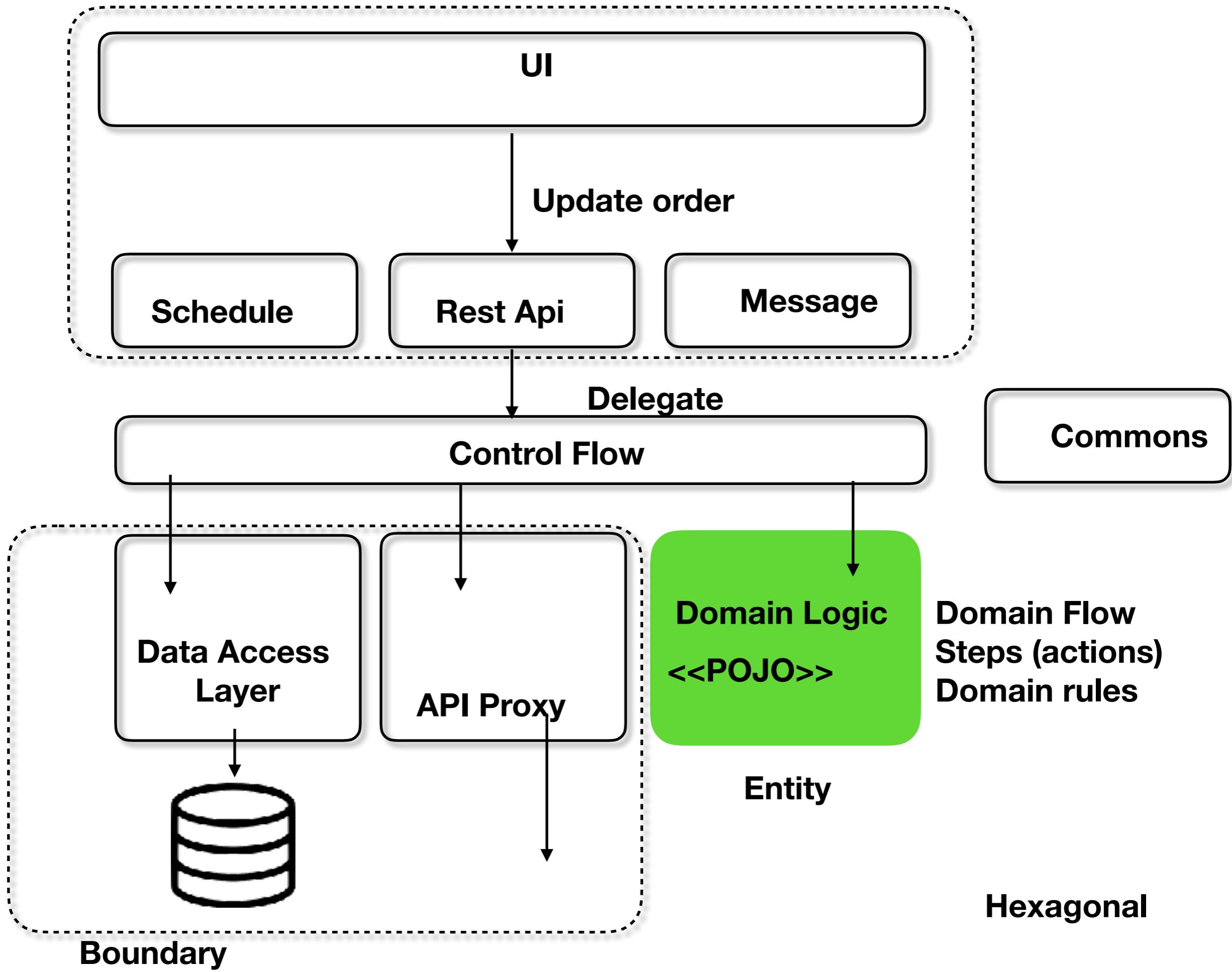
**Domain Layer**

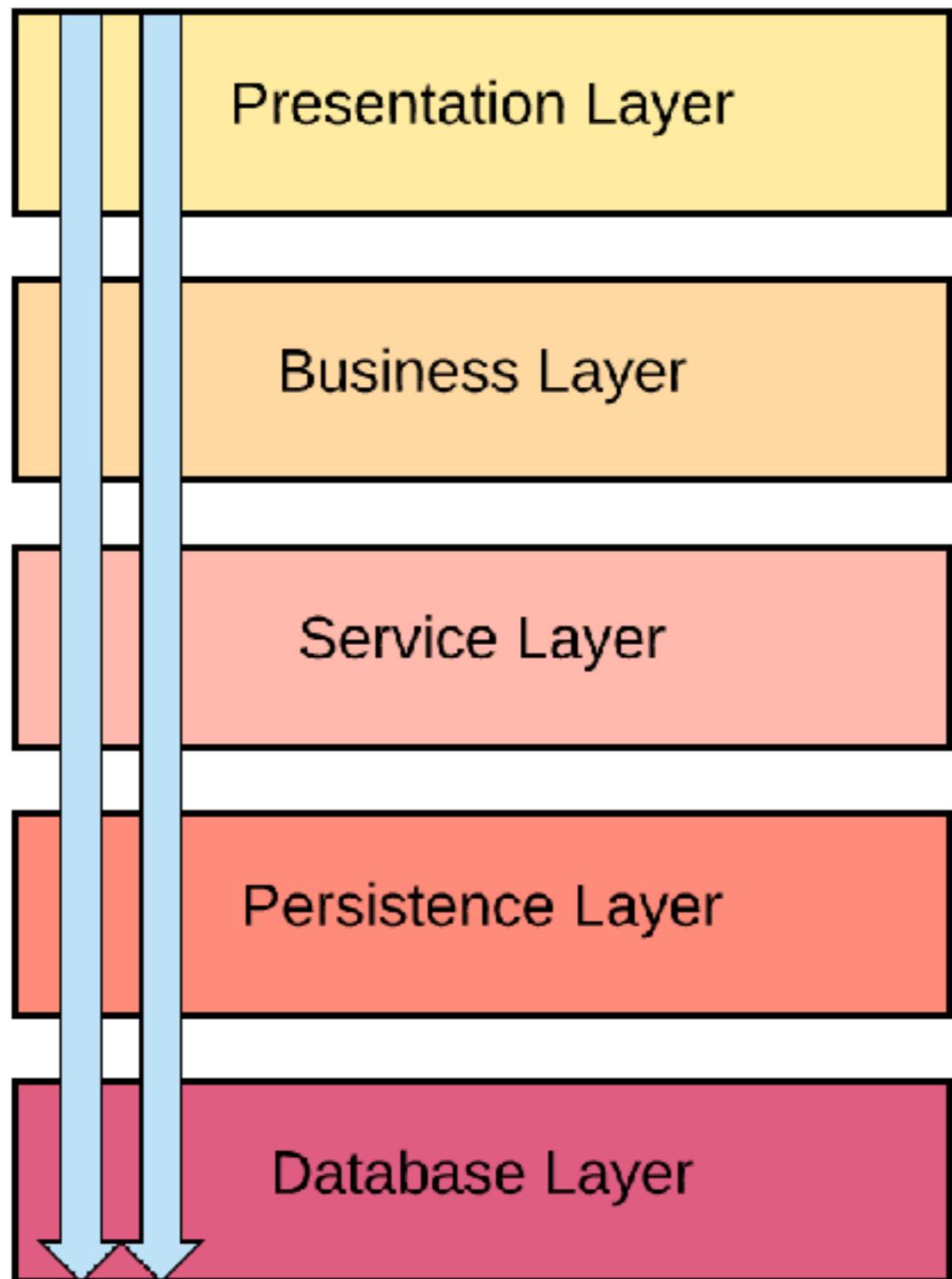
**Data Layer**



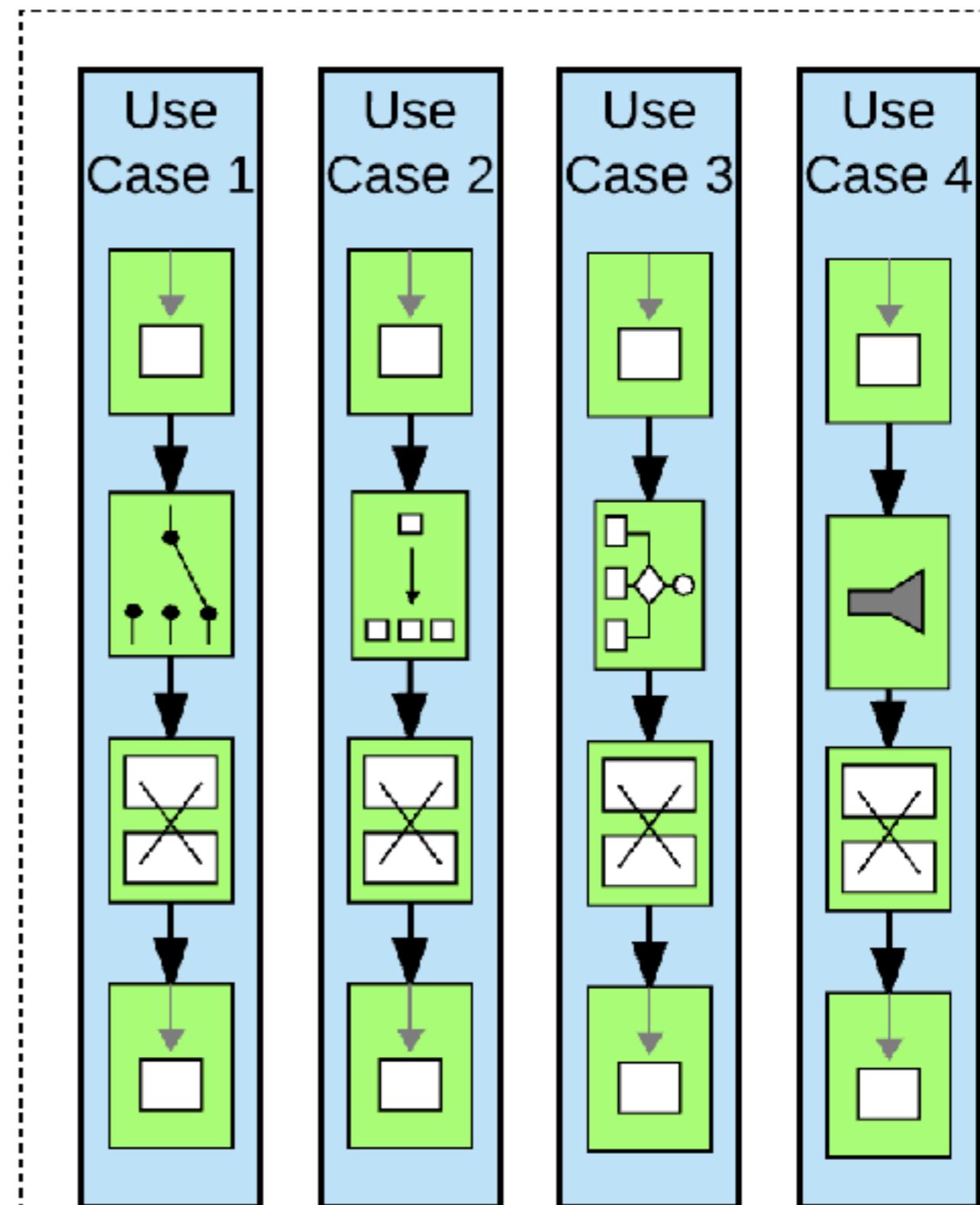


# Boundary



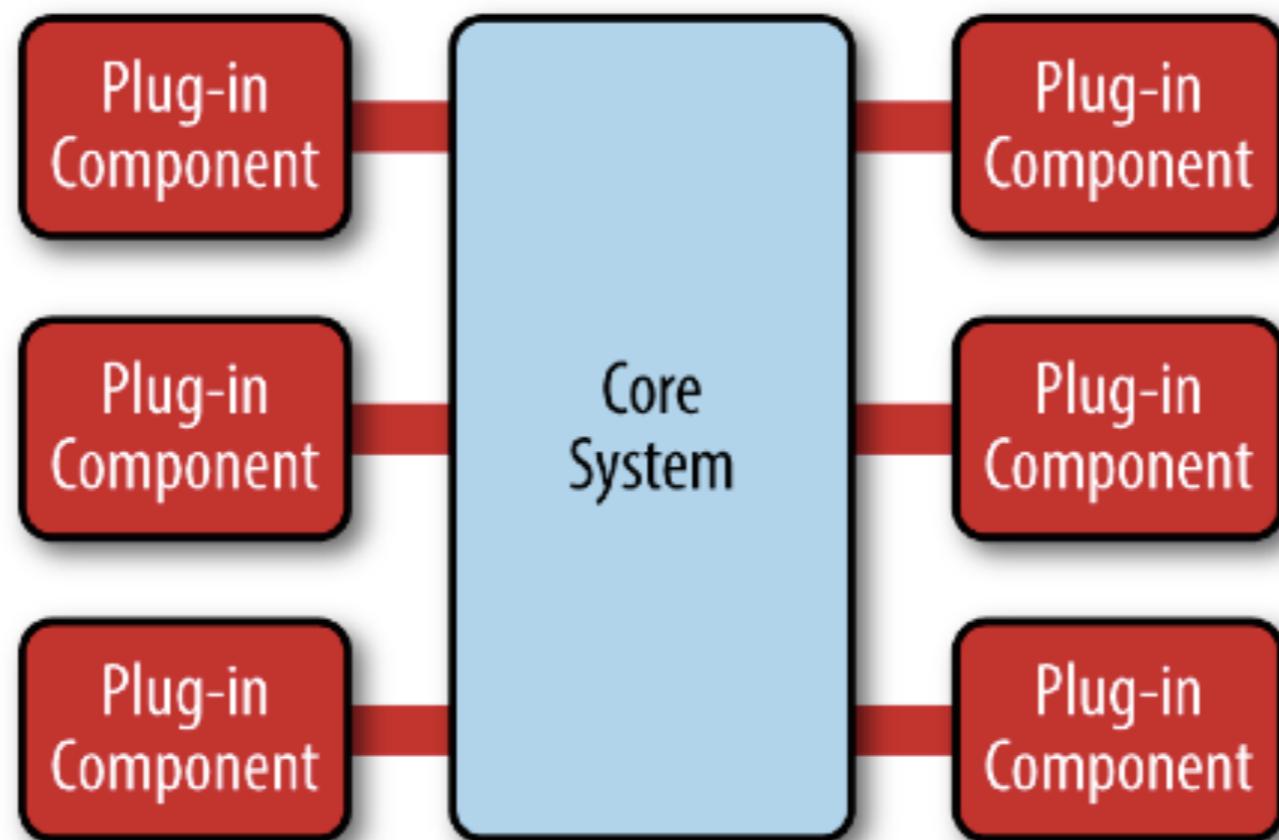


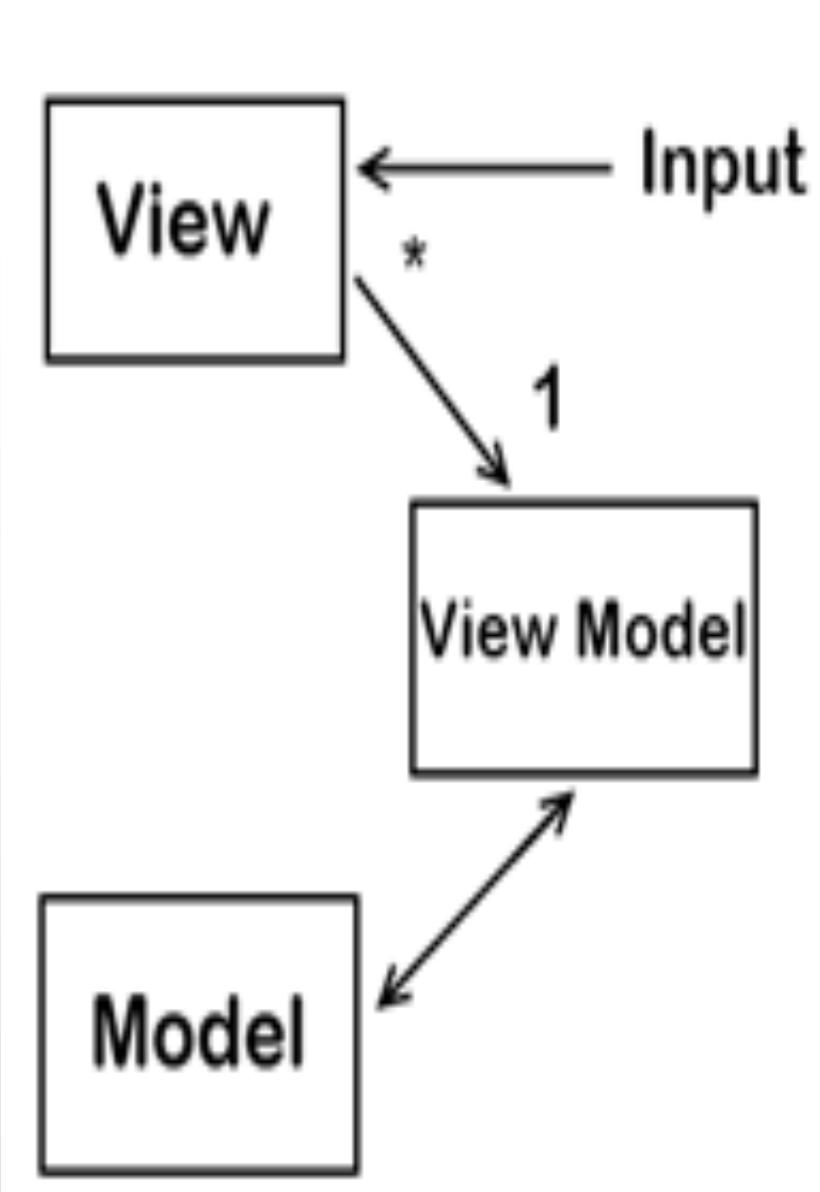
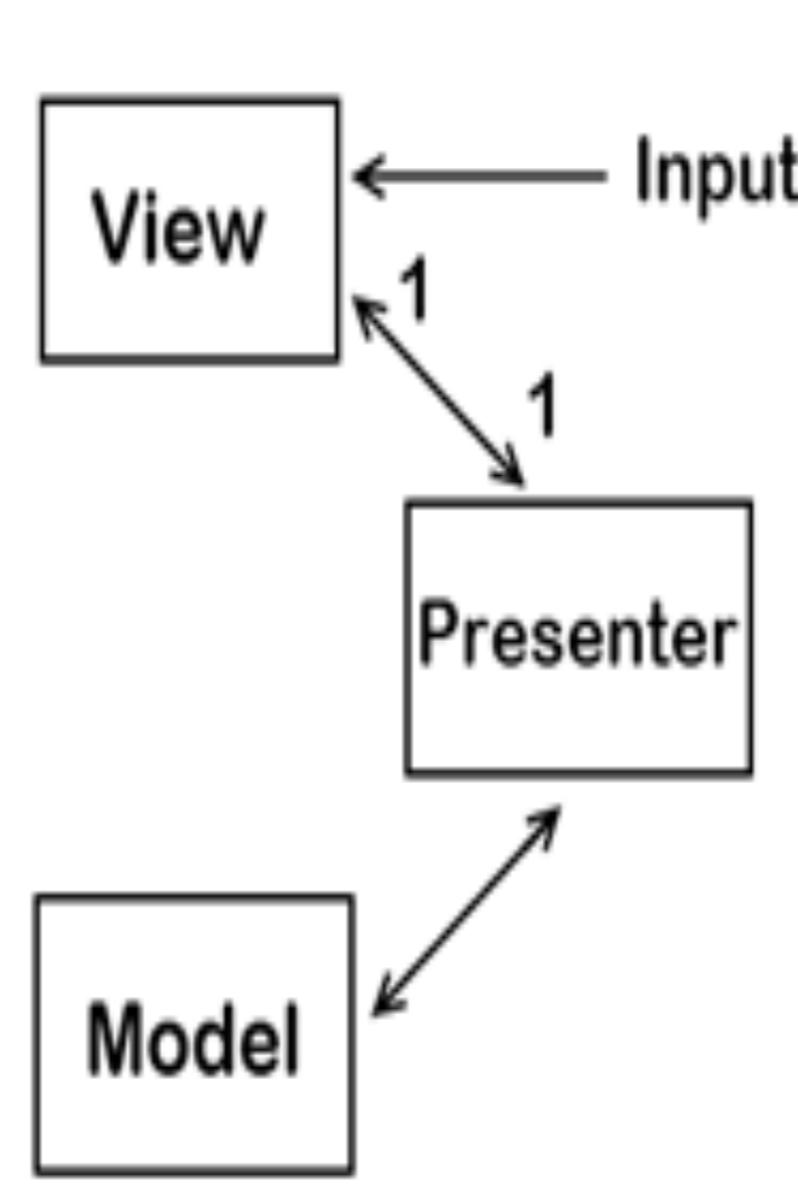
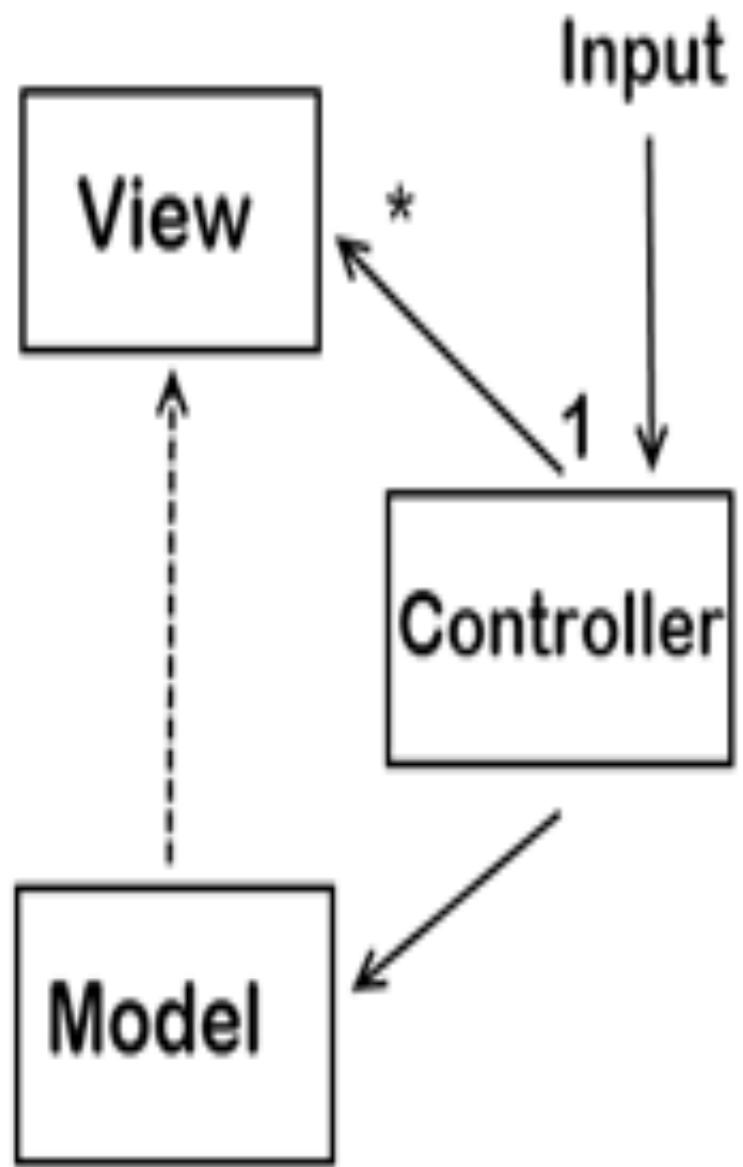
**Layered Architecture**



**Pipes and Filters**

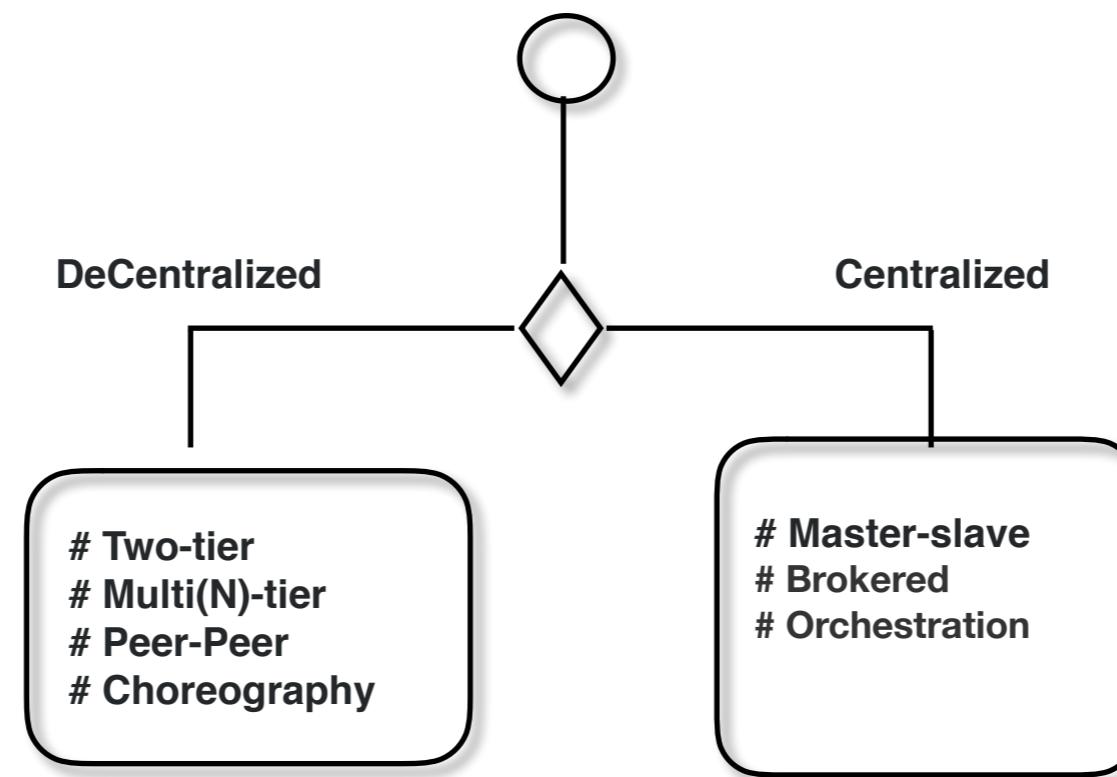
# Microkernel Architecture

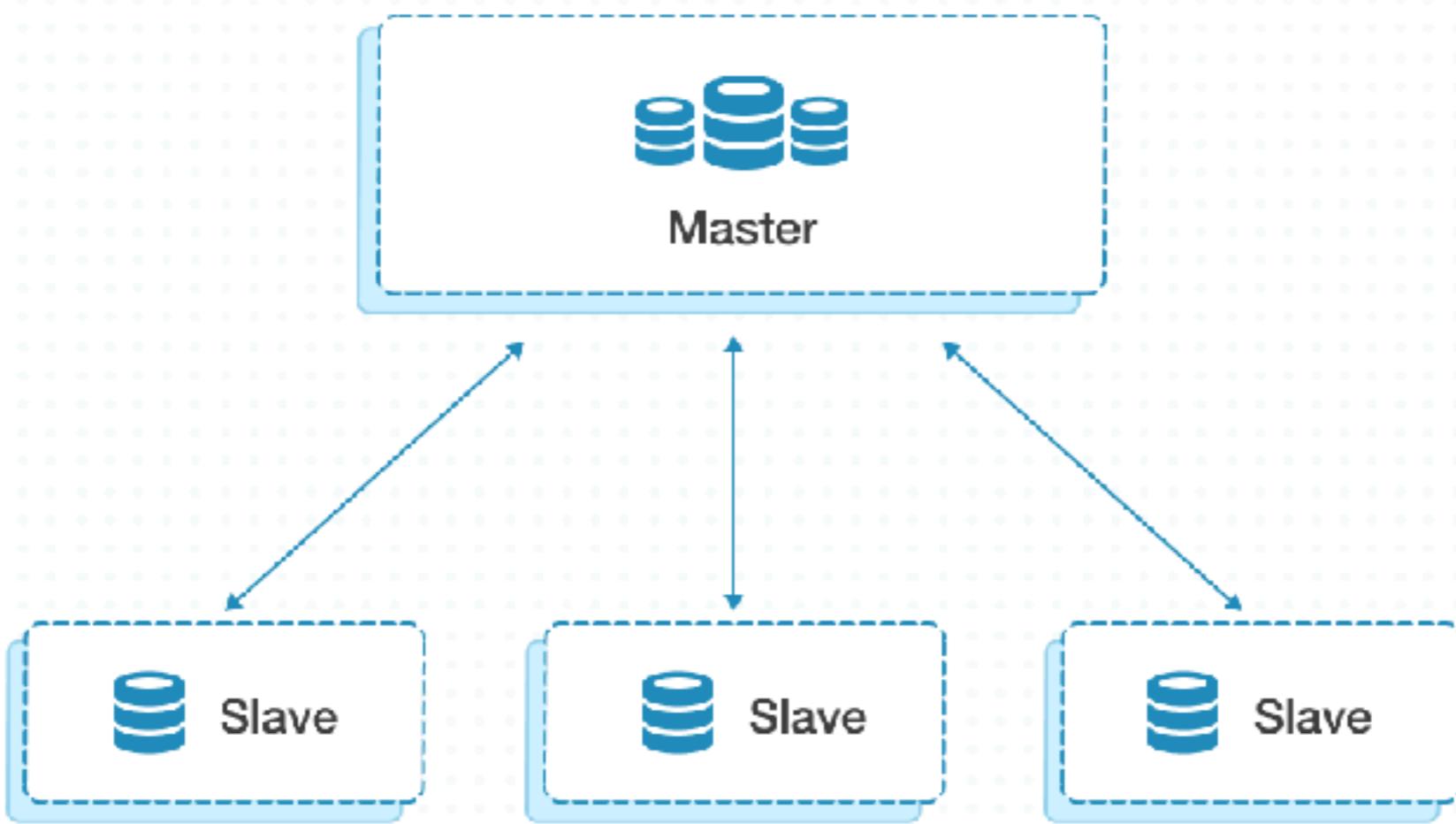




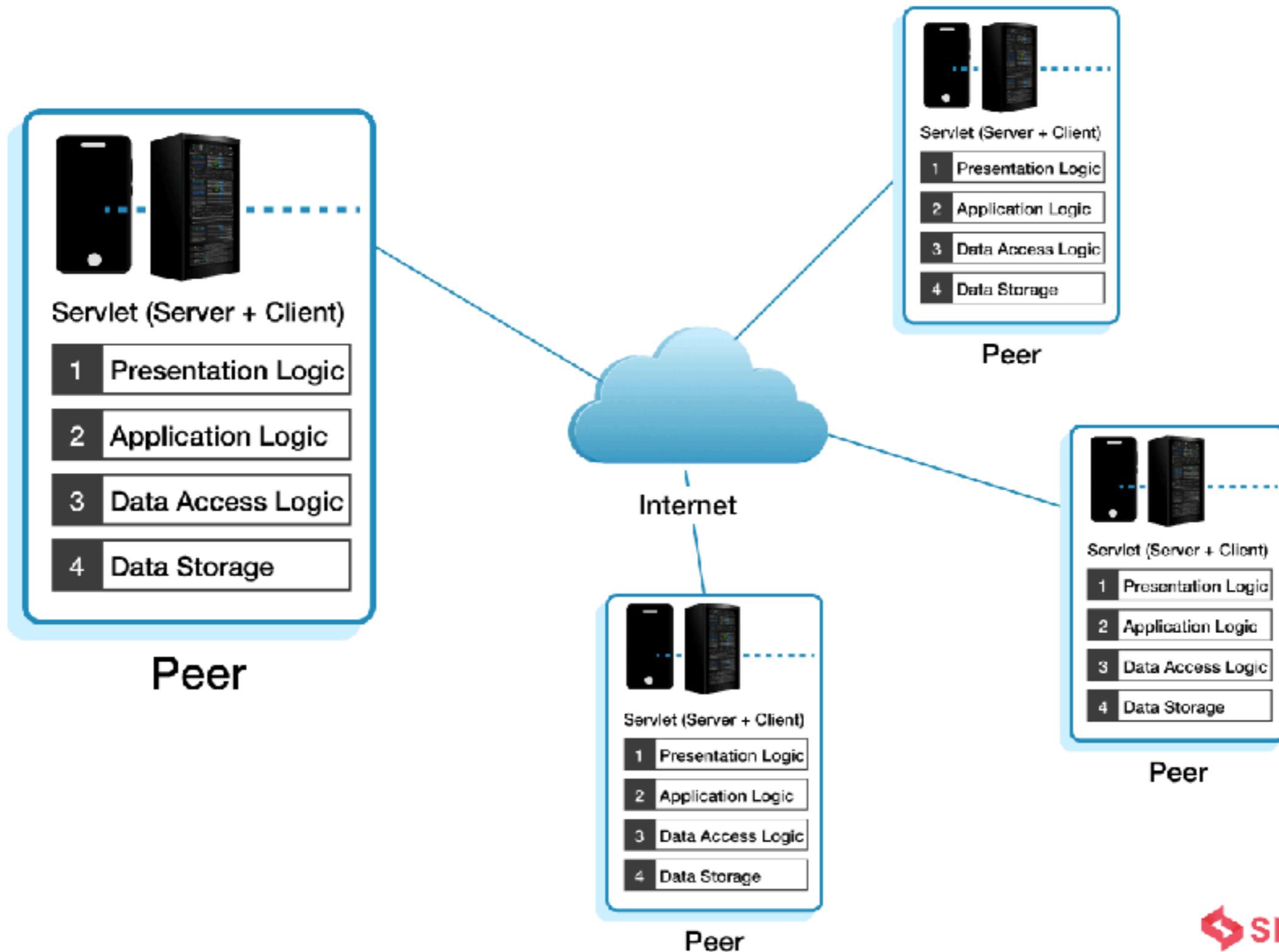
	Layered	Pipes and Filter	Micro Kernel
SOC	layer	Filter	common/Variance
Low Coupling	--	+	++
Does not need Abstraction	++	-	--
Ease of Design	++	-	--
Agility ( Shuffle, Rearrange, Add)	--	+	++
Integration Testability	--	+	++
Needs less domain knowledge	++	-	--

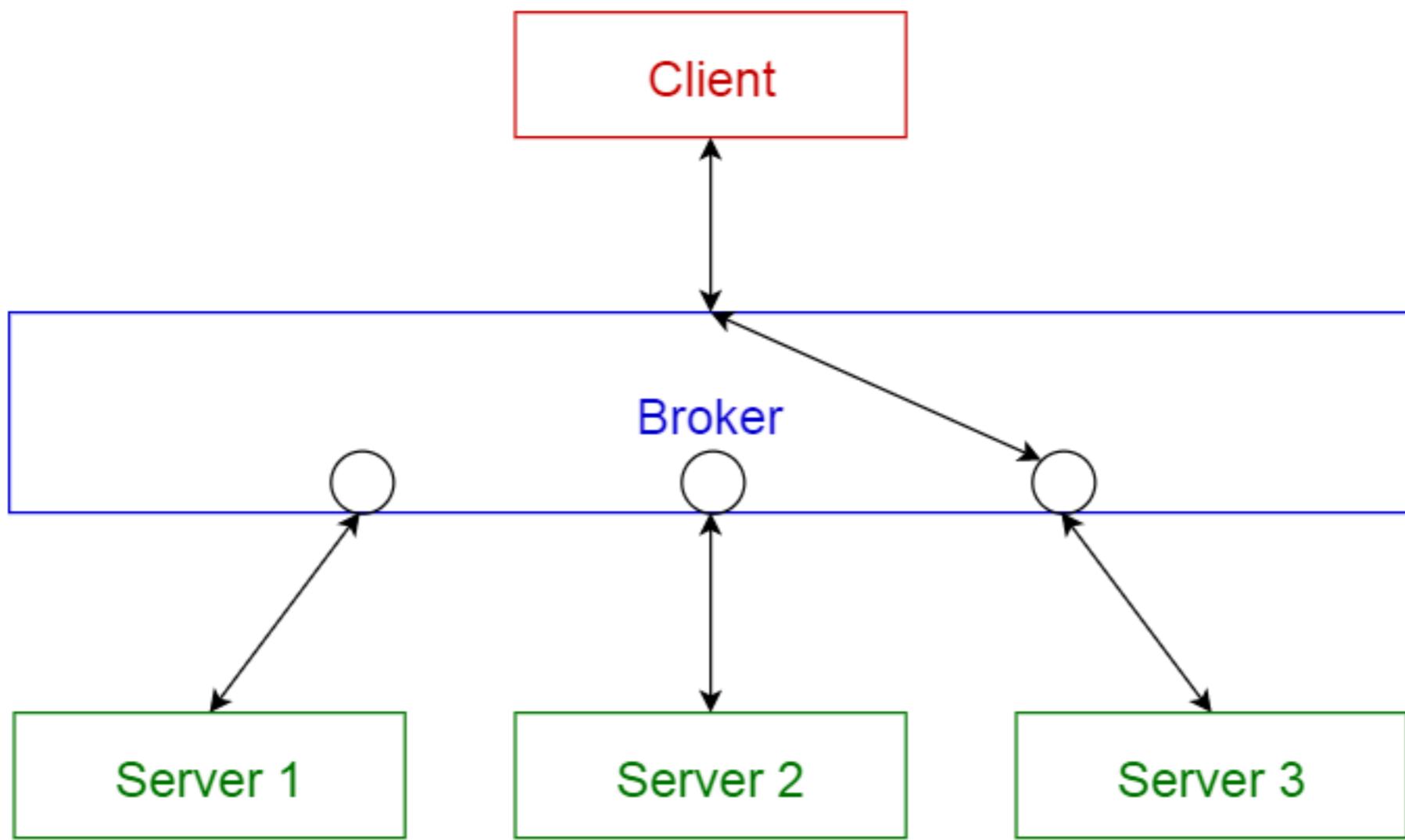
# Distributed Communication Patterns

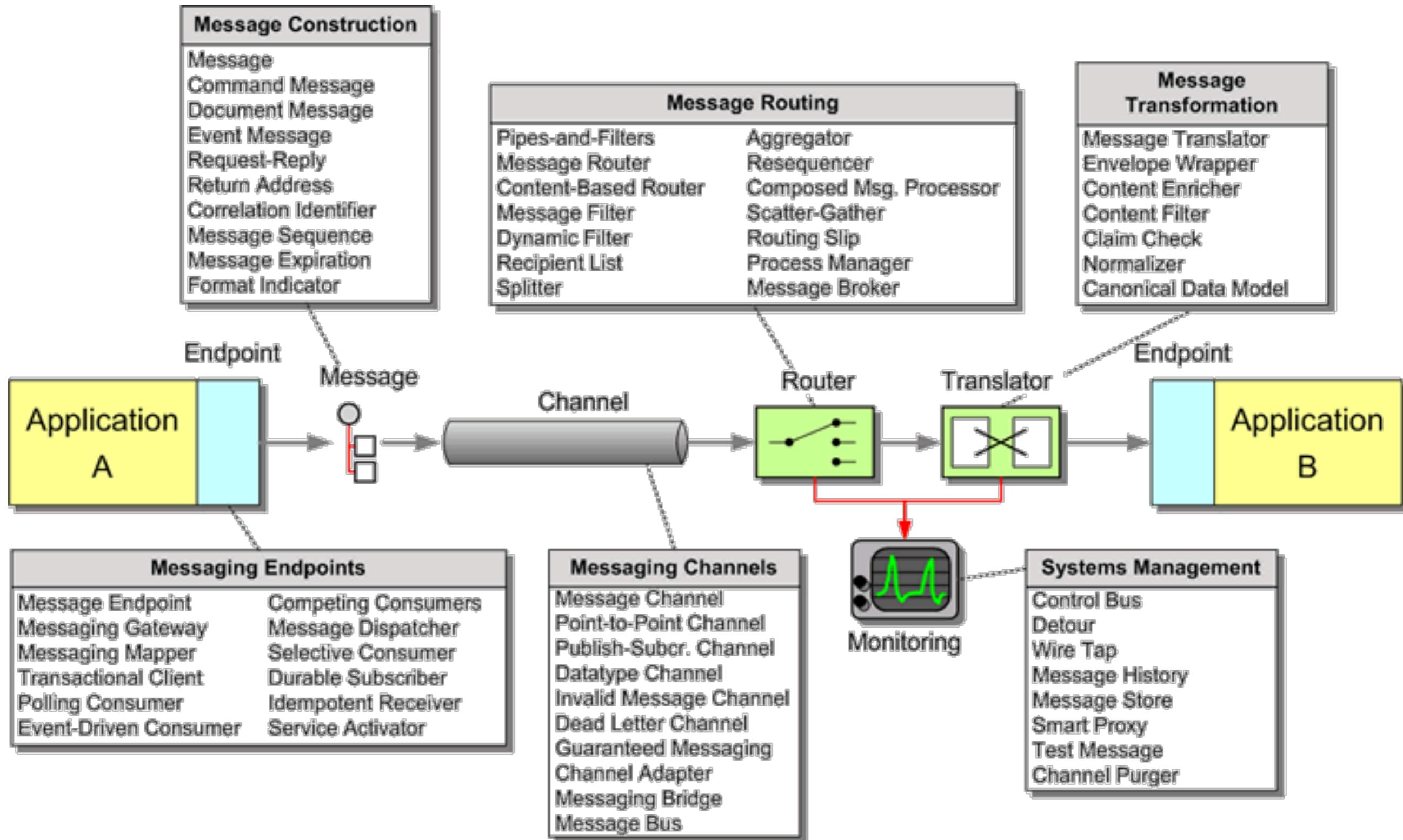


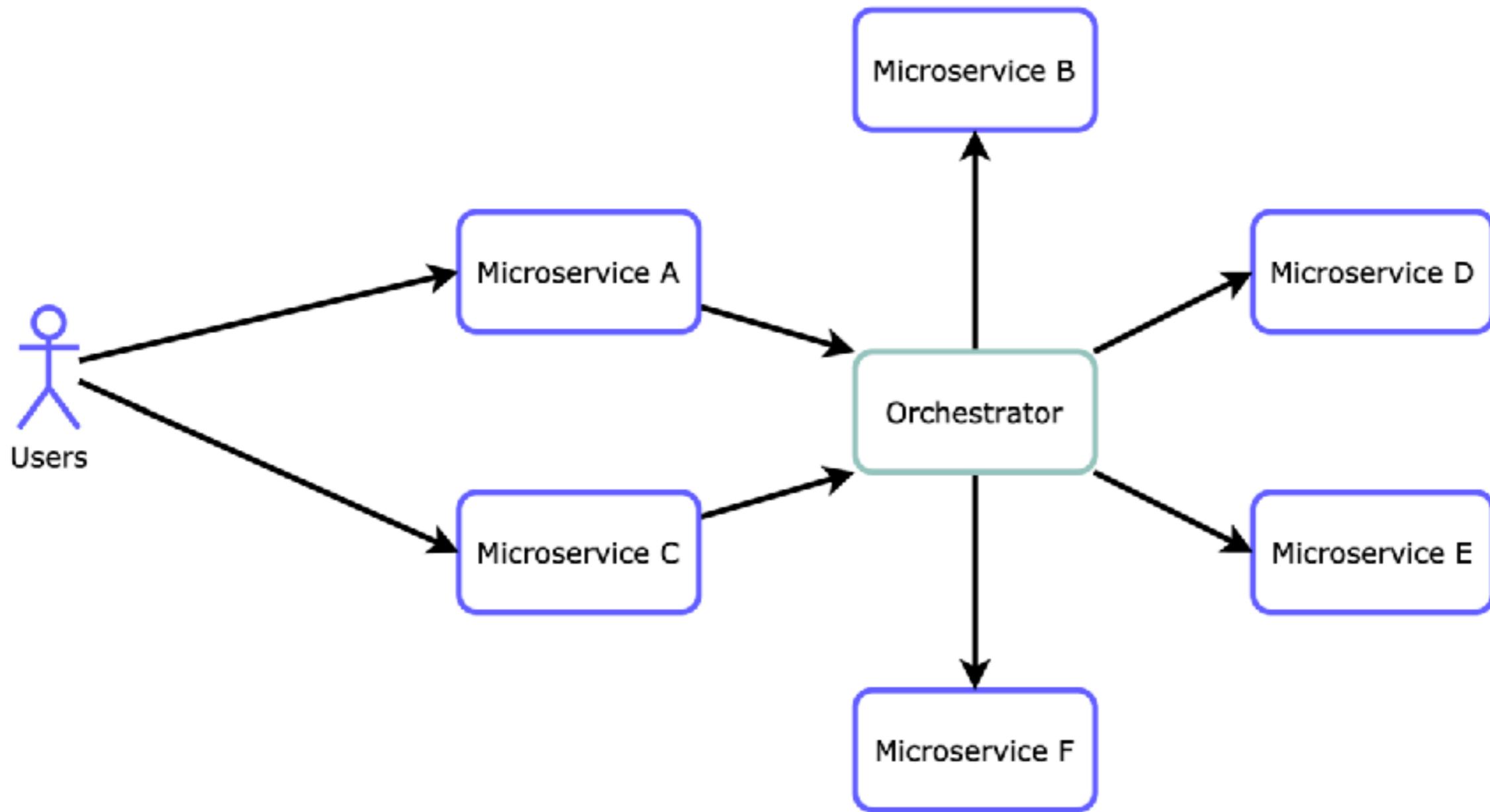


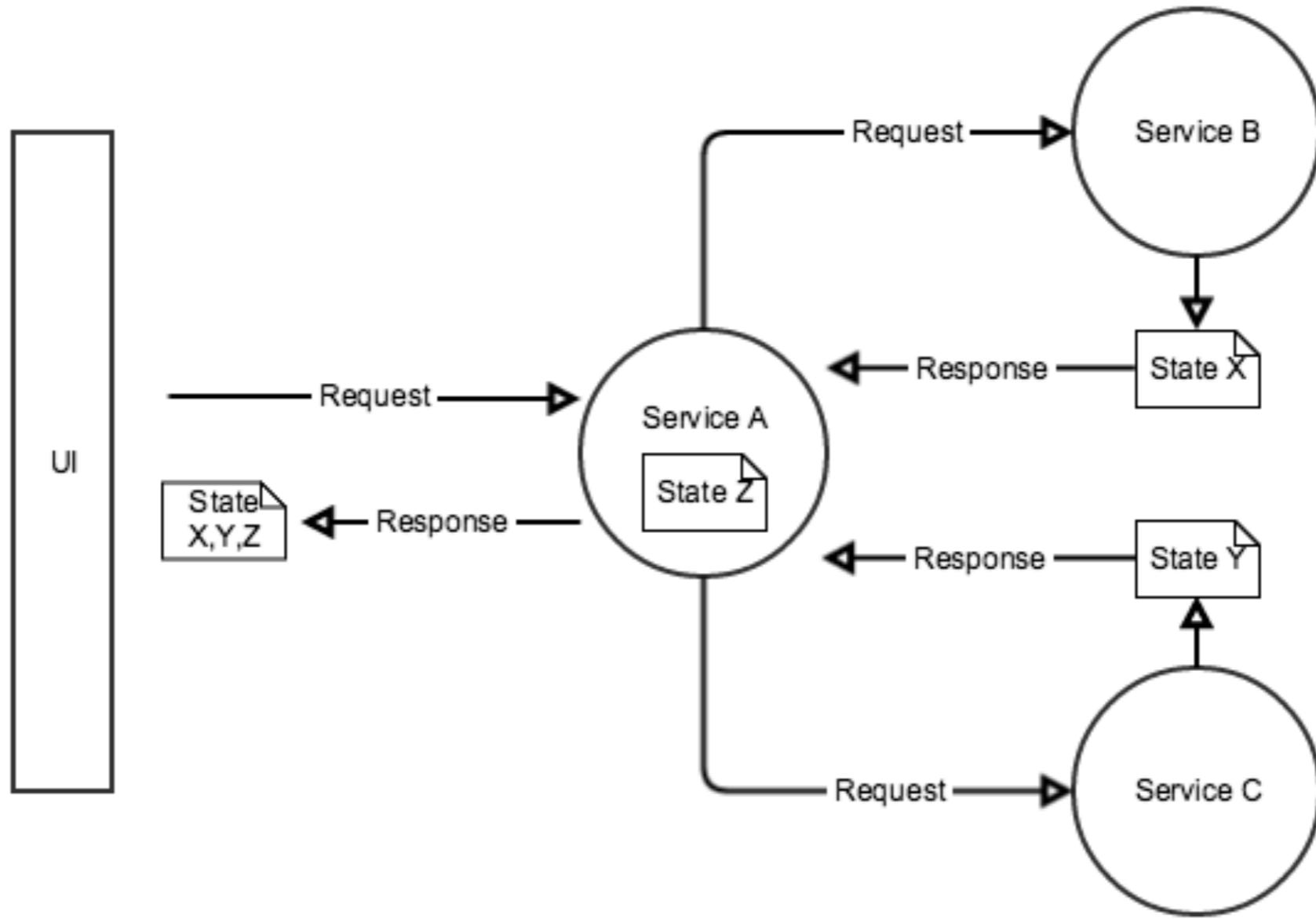
# Peer-to-Peer architecture

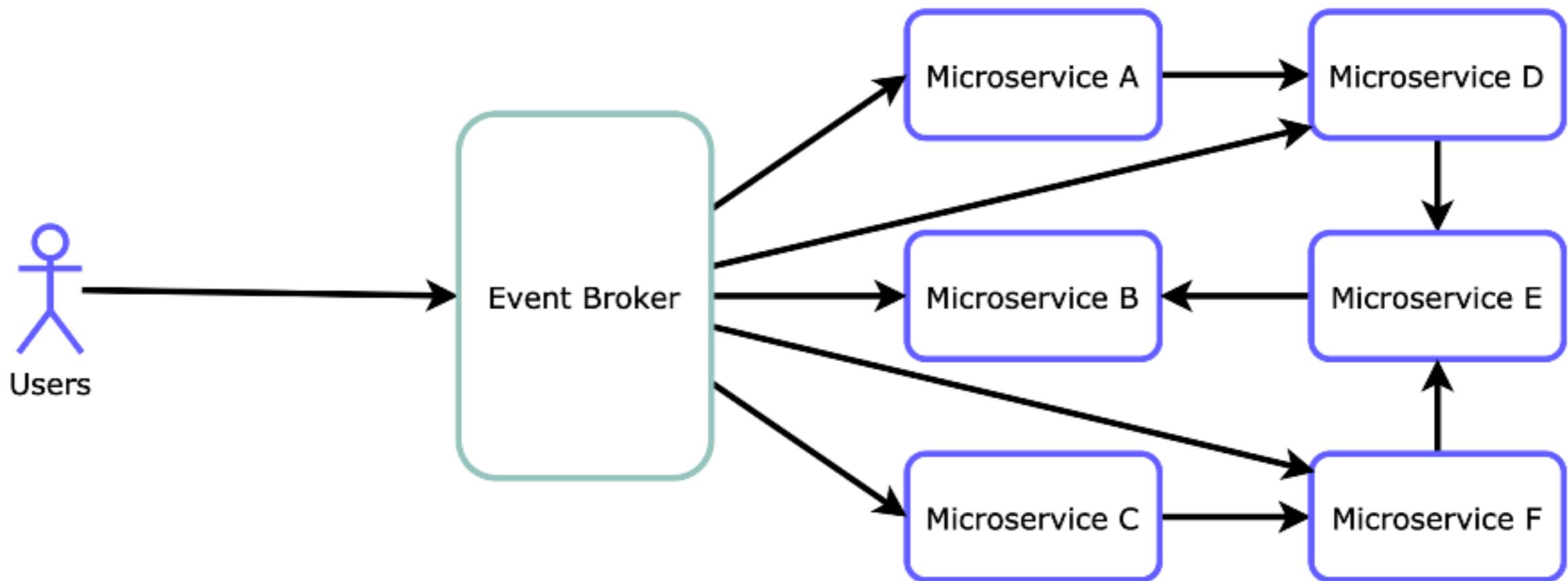




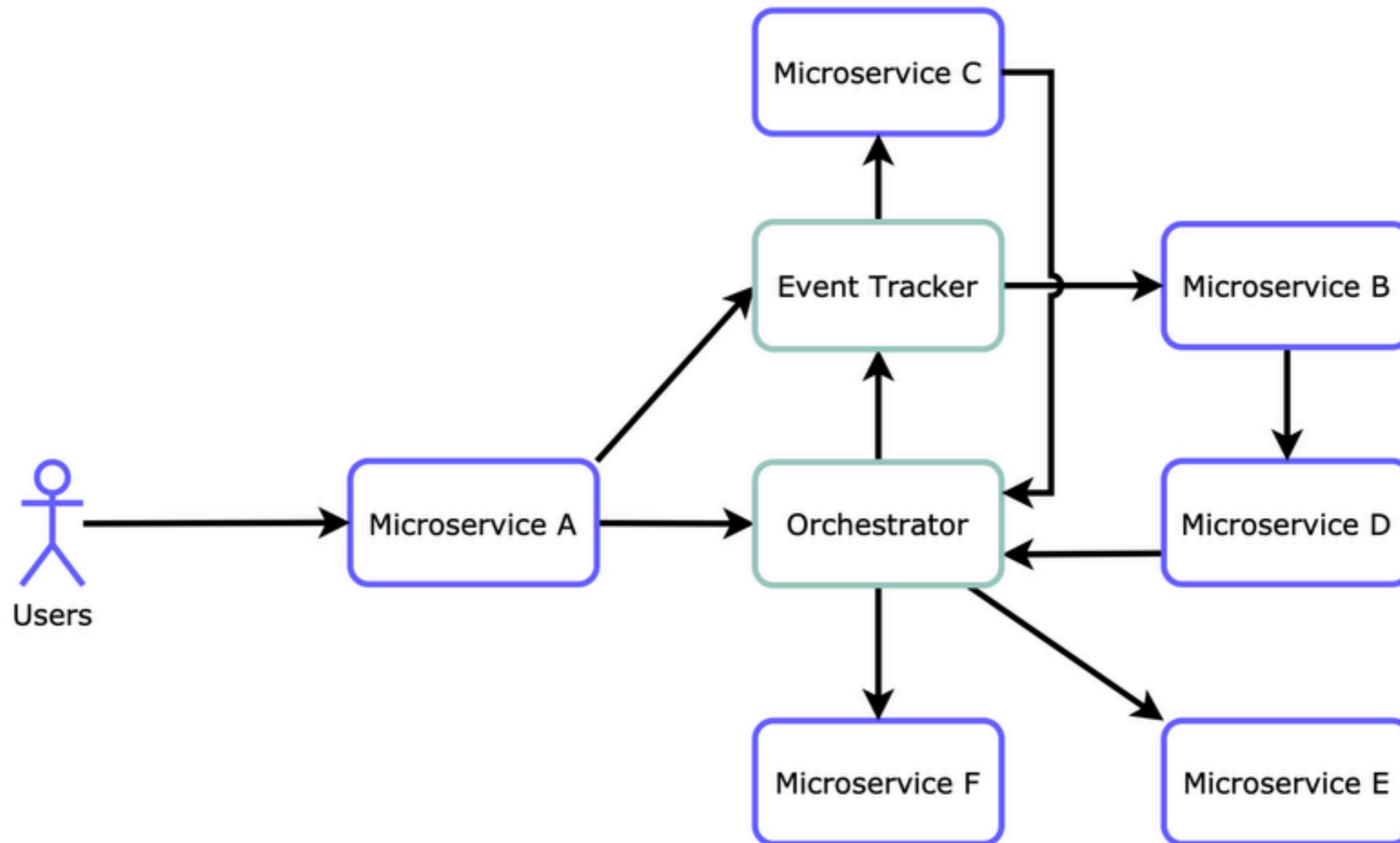


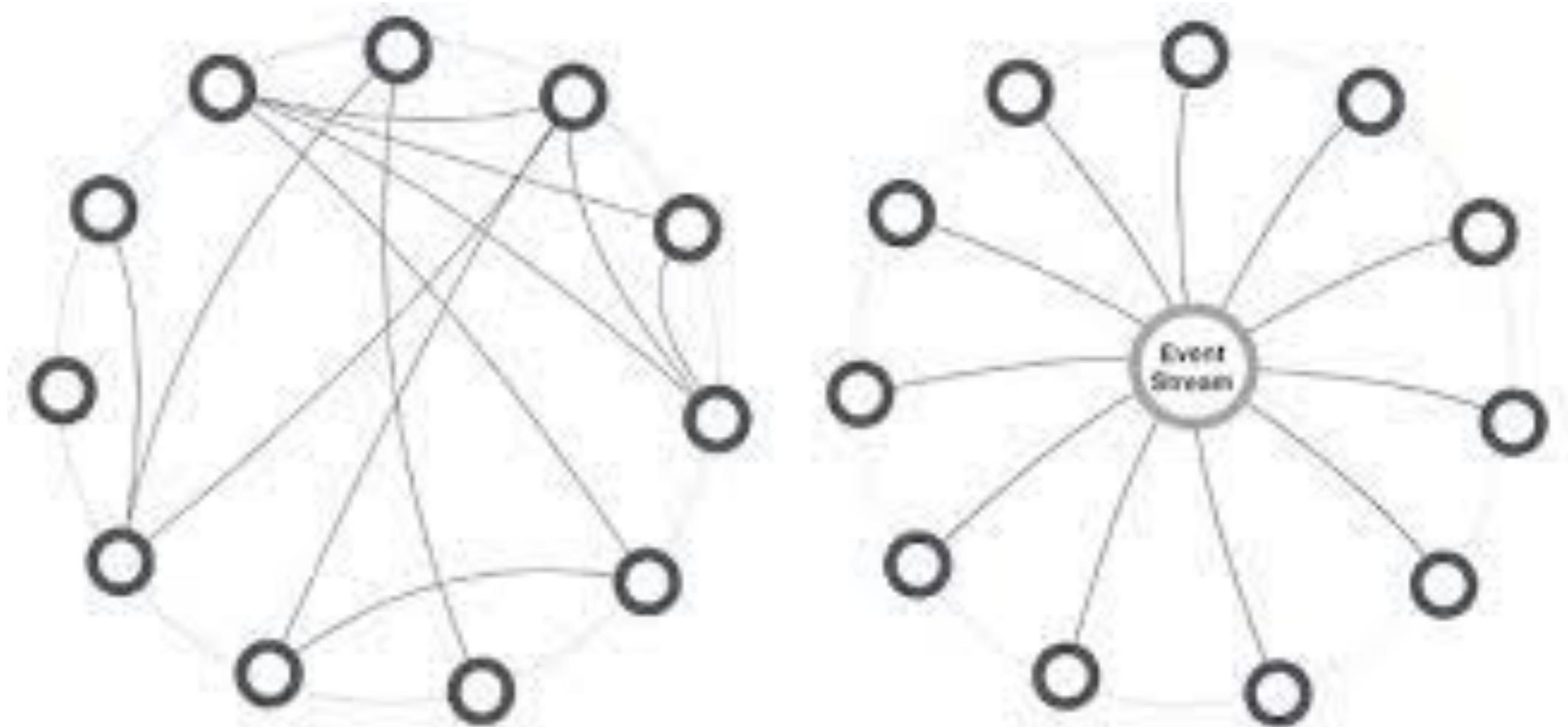




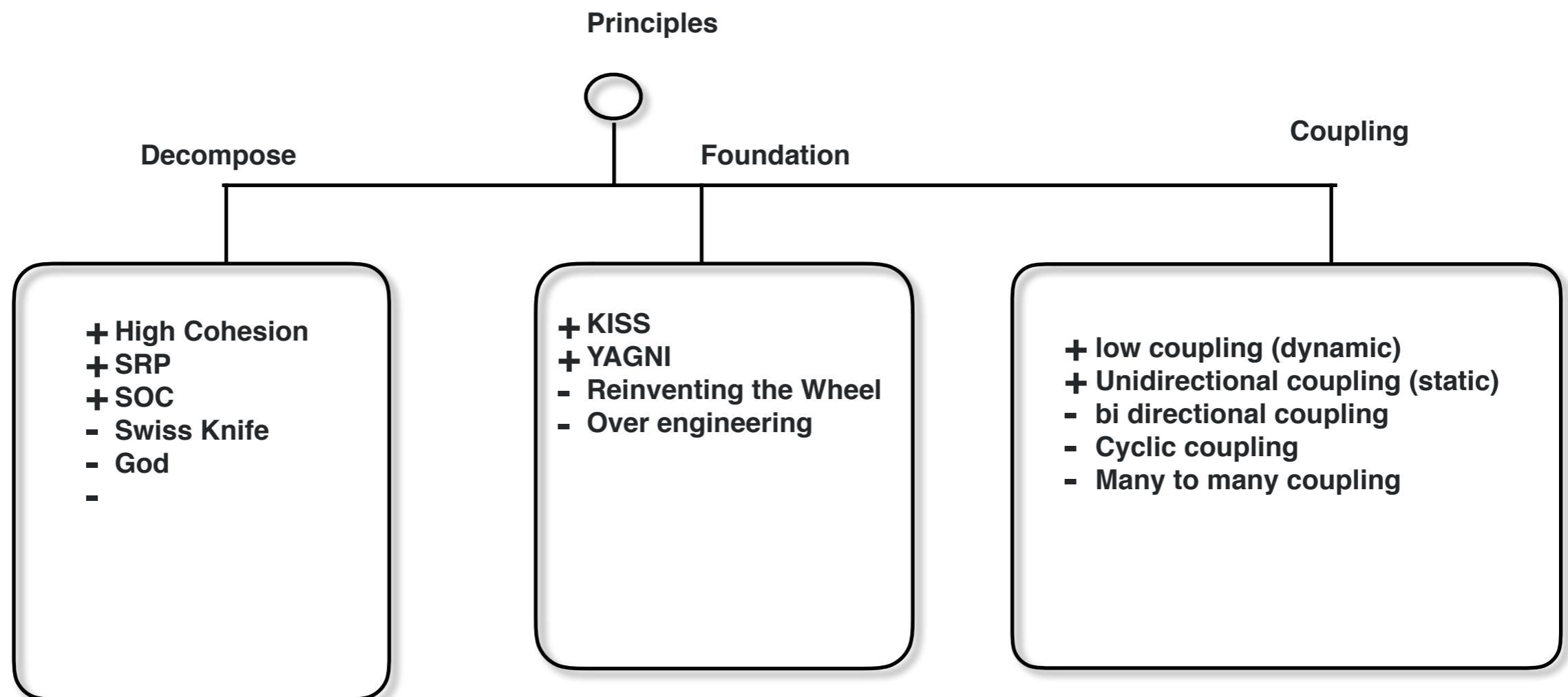


# Hybrid

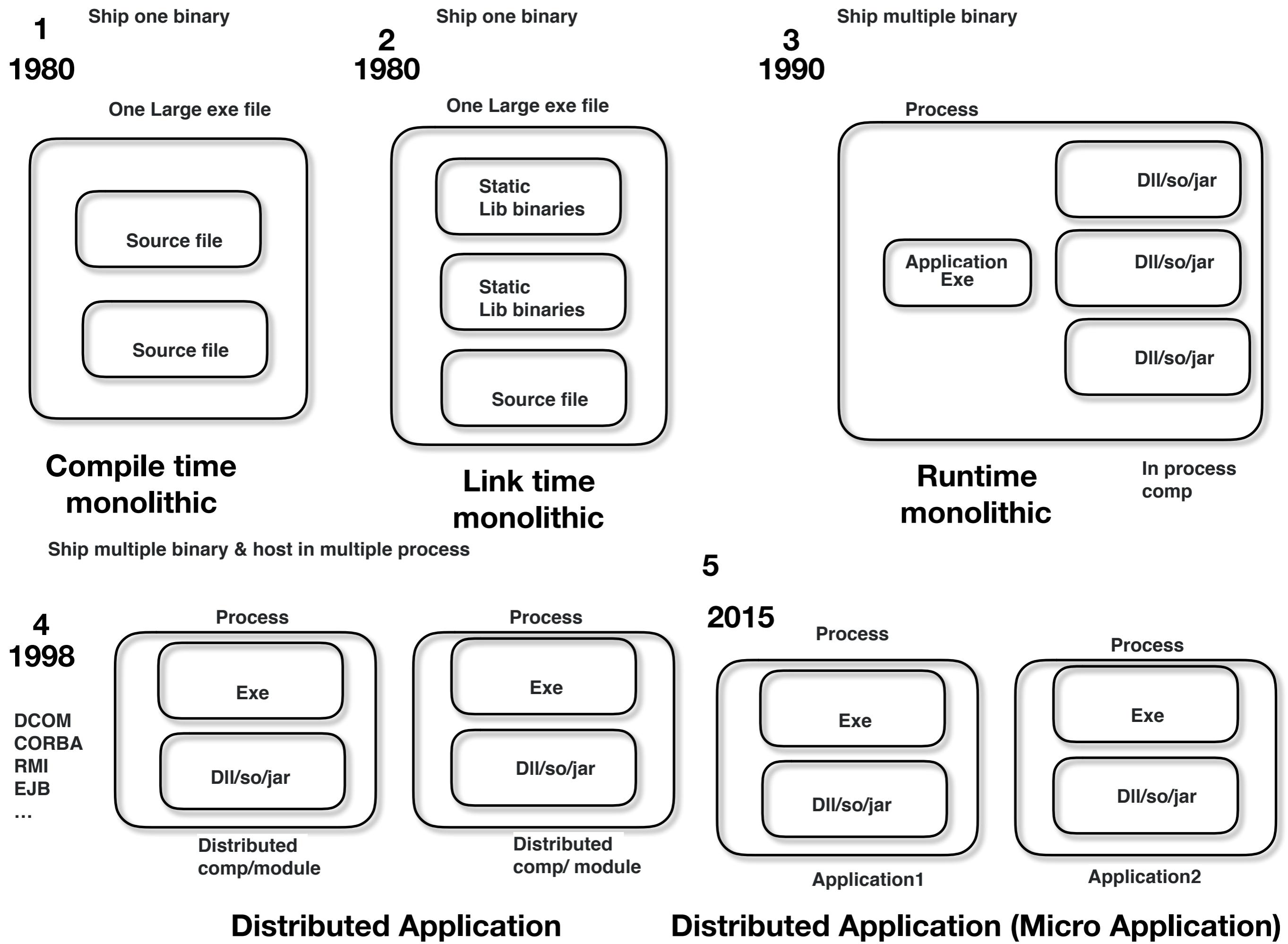




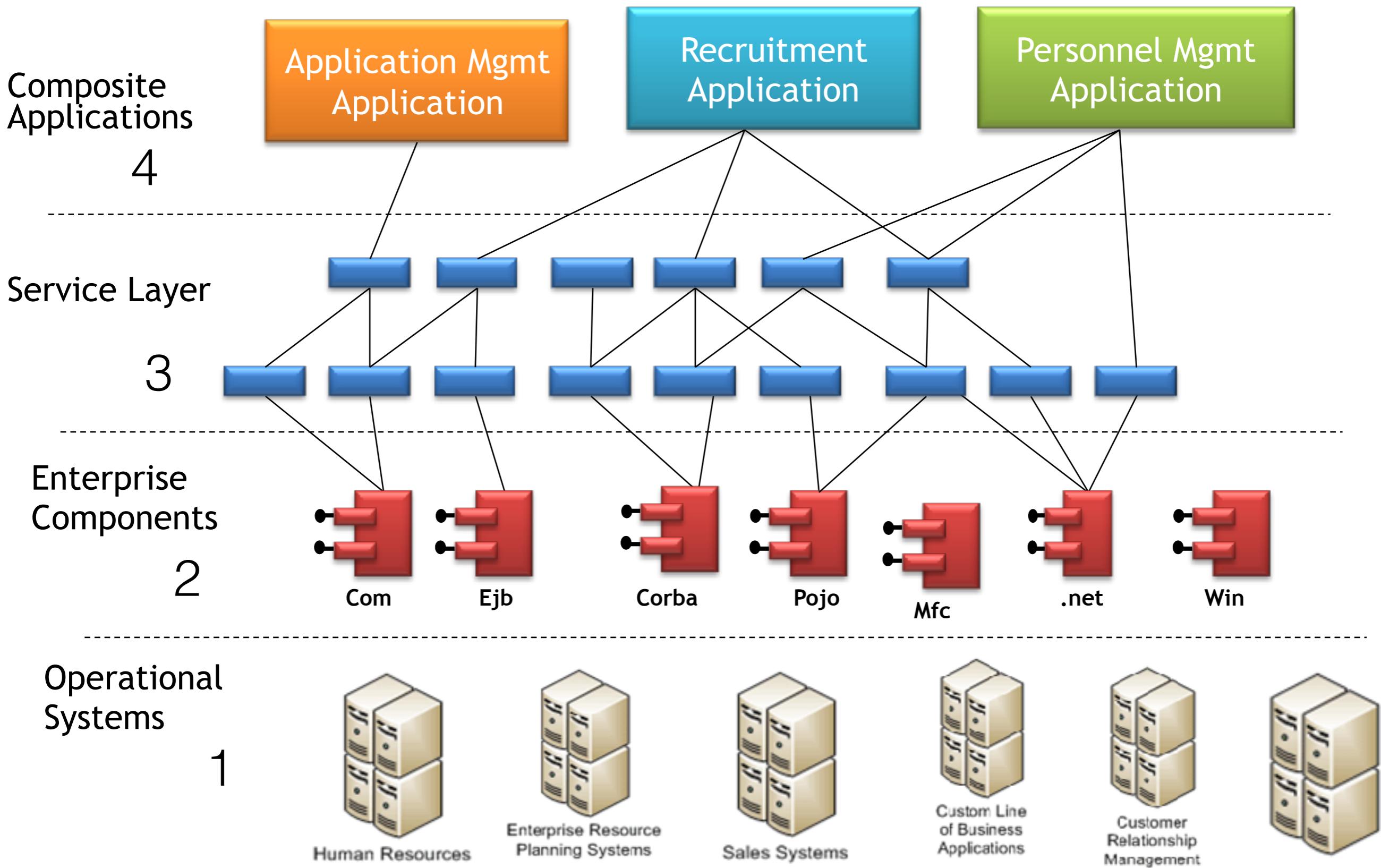
# Apply Architecture Principles



# Microservice



# Service Oriented Architecture



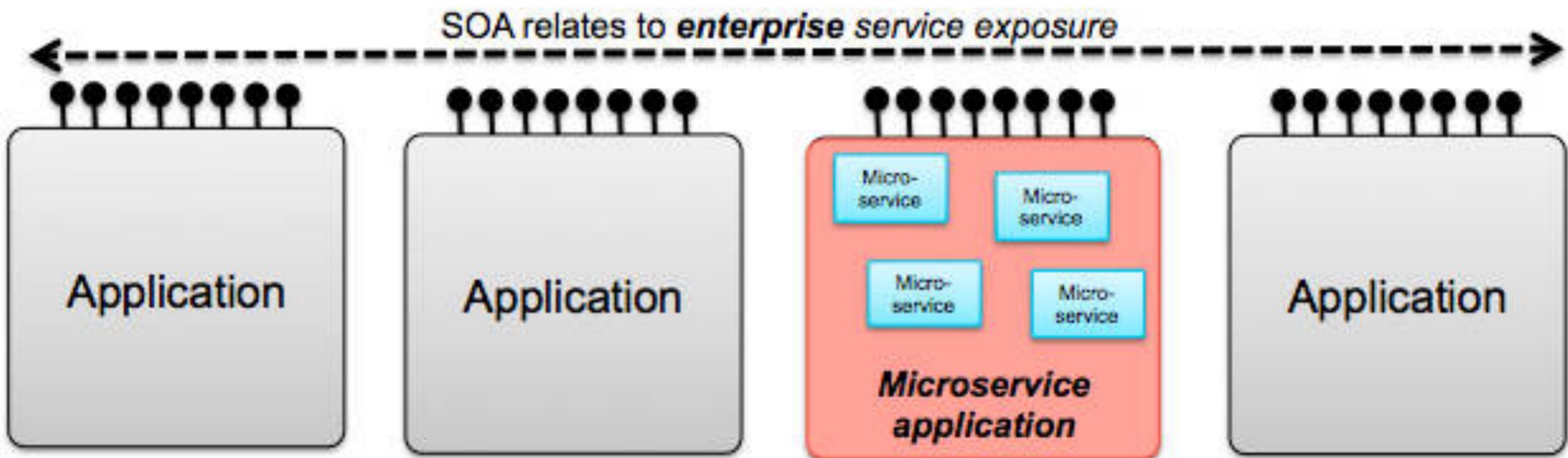
ToDo Portal  
(Single Page Application)

ToDo API Service  
(Api Application)

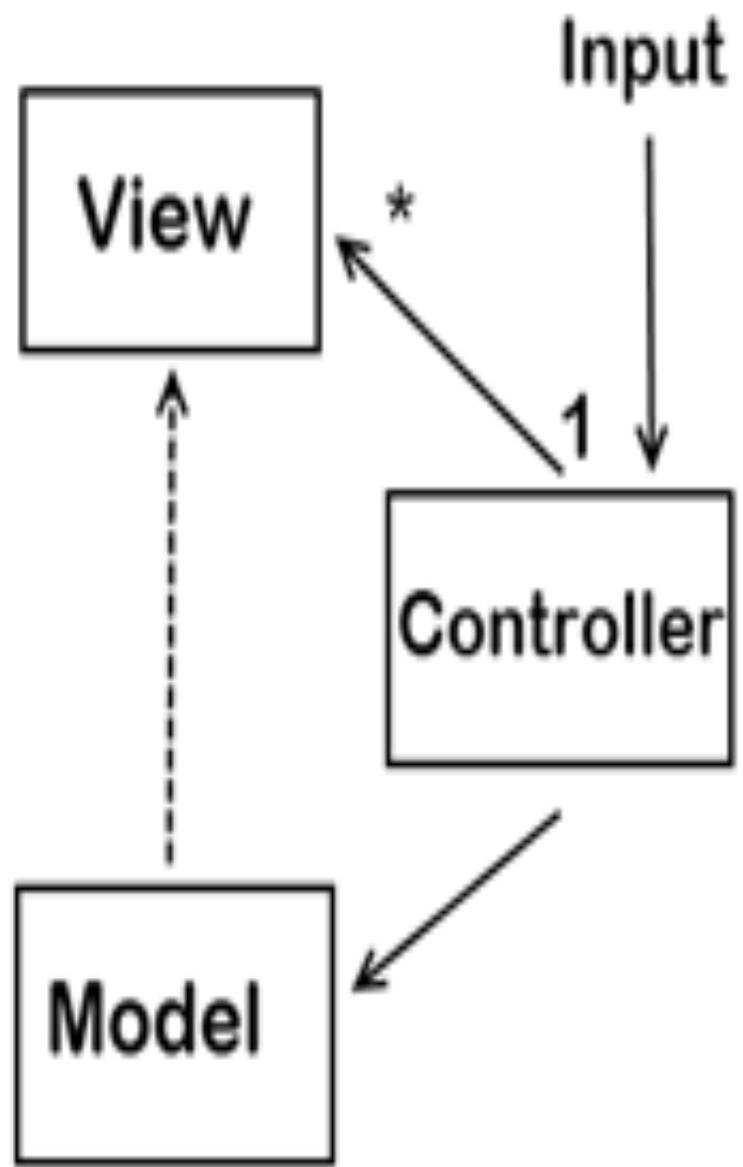


ToDo Calendar Service  
(Background Application)

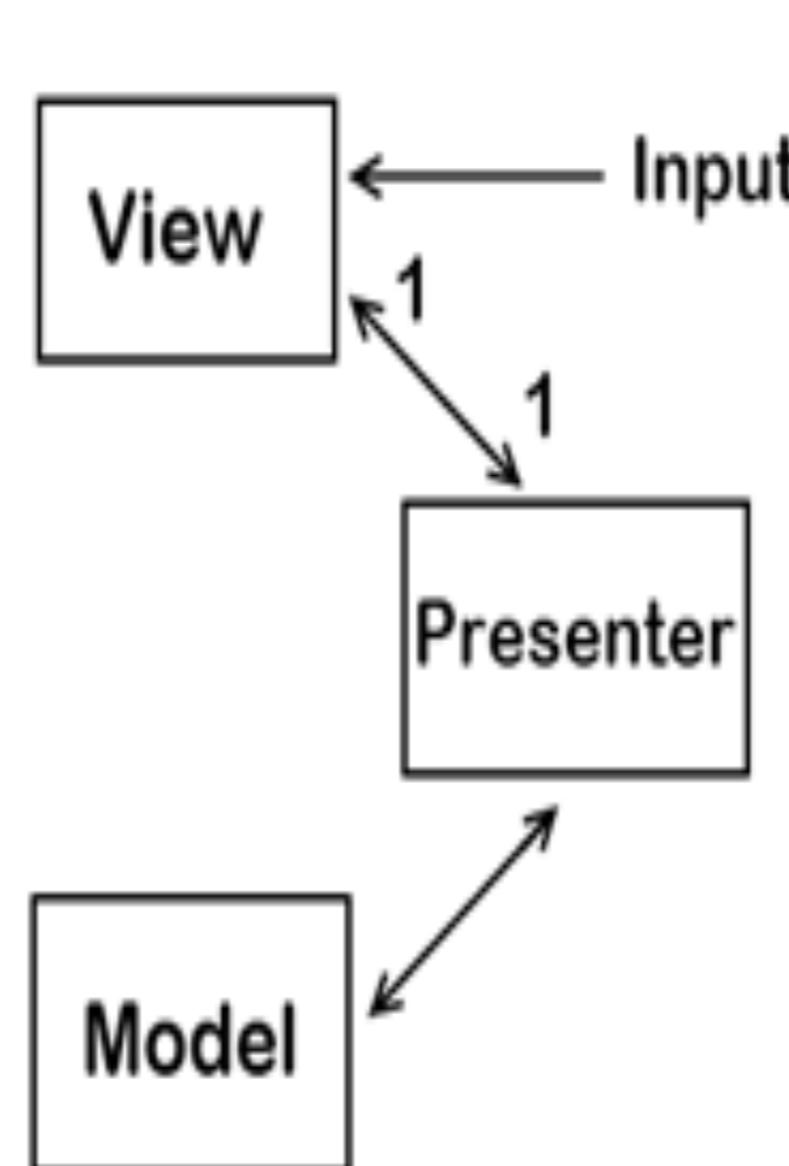
ToDo Prediction Service  
(Background Application)



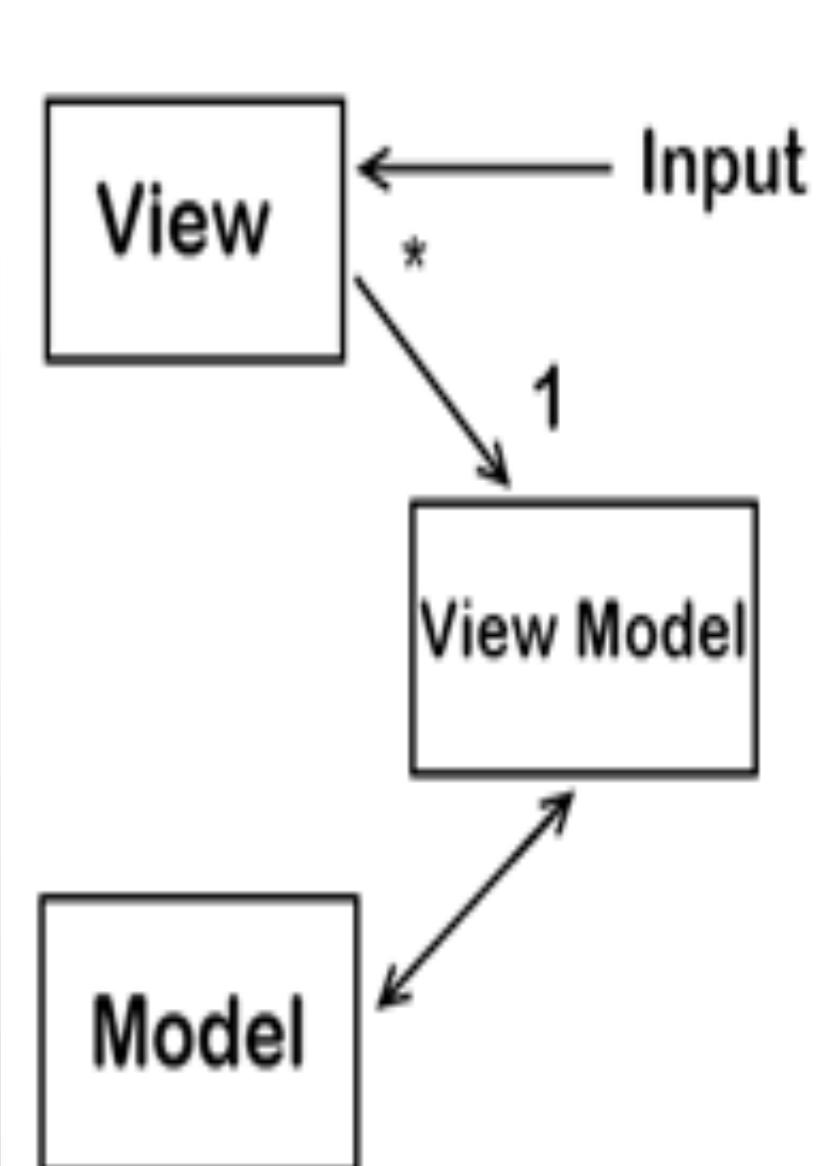
Microservices relate to  
**application** architecture



MVC



MVP



MVVM

# 2 distributed binaries

	<b>2 Modules</b>	<b>2 Applications</b>	<b>W</b>	<b>Score</b>
Share Database / Storage	Yes	No *	2	0
Share same Virtual Infra	Yes	No	3	1
Share Source Control	Yes	No	2	1
Share CI/CD (Build pipeline)	Yes	No	3	1
Fun Requirements	Application level	It owns	1	1
SCRUM Team / Sprint	Application Scope	Its owns	1	1
Acceptance Test Cases	Application Scope	Its owns	1	1
Architecture	Application Scope	Its owns	1	1
Technology Stack / Fwks	Application Scope	Its owns	1	1

Application

Modules

Modules

Modules

**3 or 4**

Application

Small App

Small App

Small App

Modules

Modules

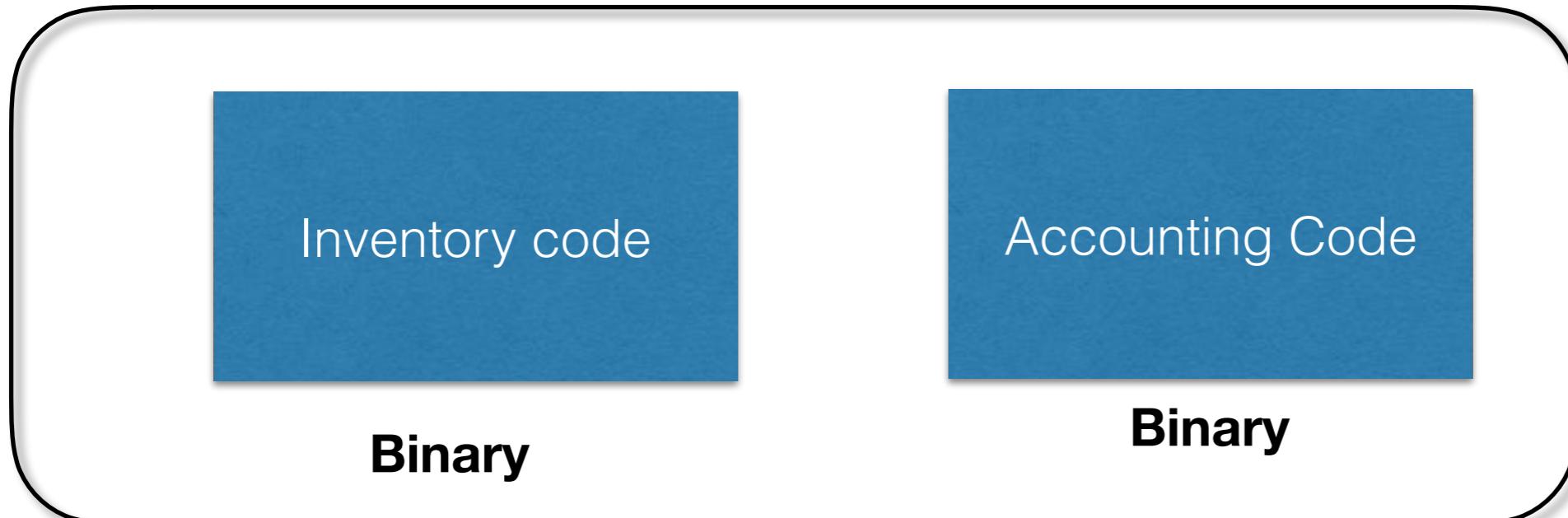
Modules

Modules

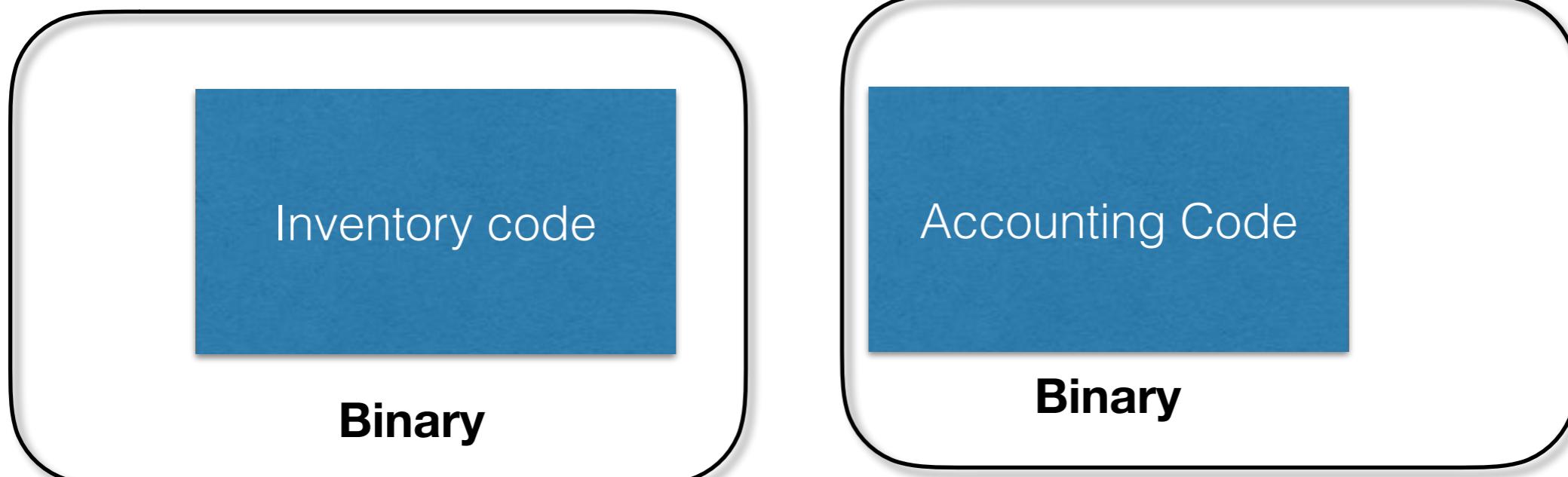
Modules

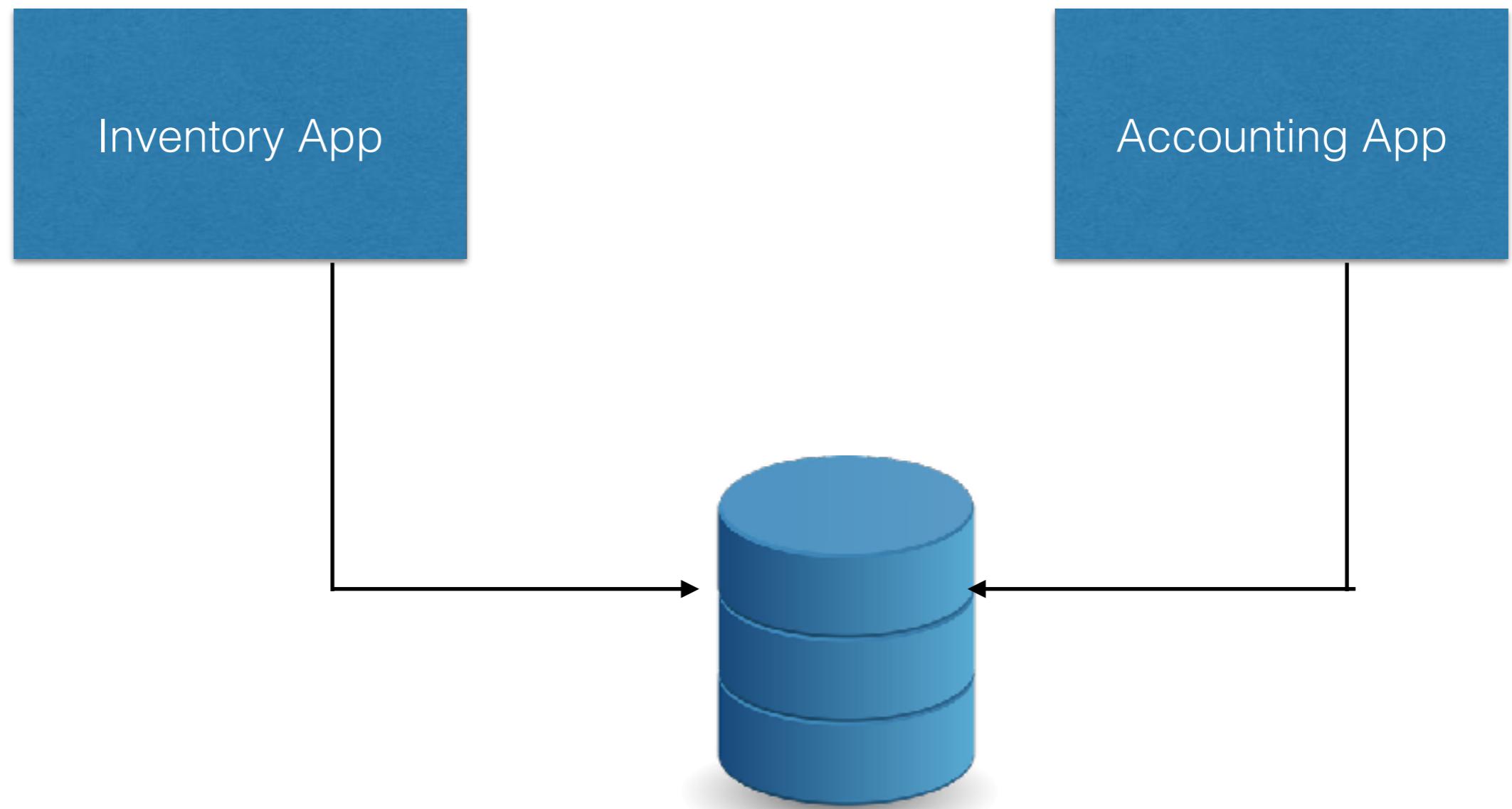
Modules

**App**



**App**





**module/ app**

Inventory  
Code

**module/ app**

Accounting  
Code



Data loss



Data Correction



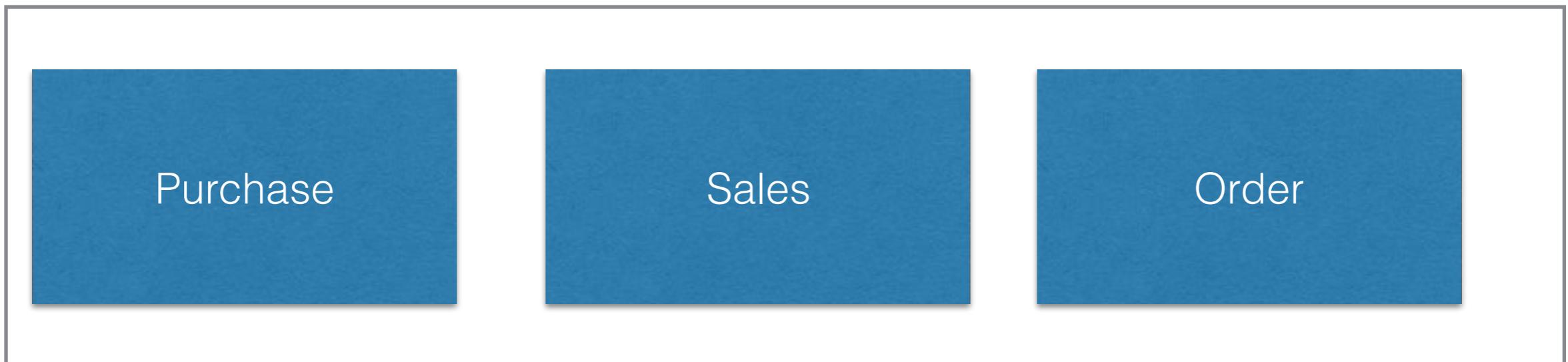
Data ...

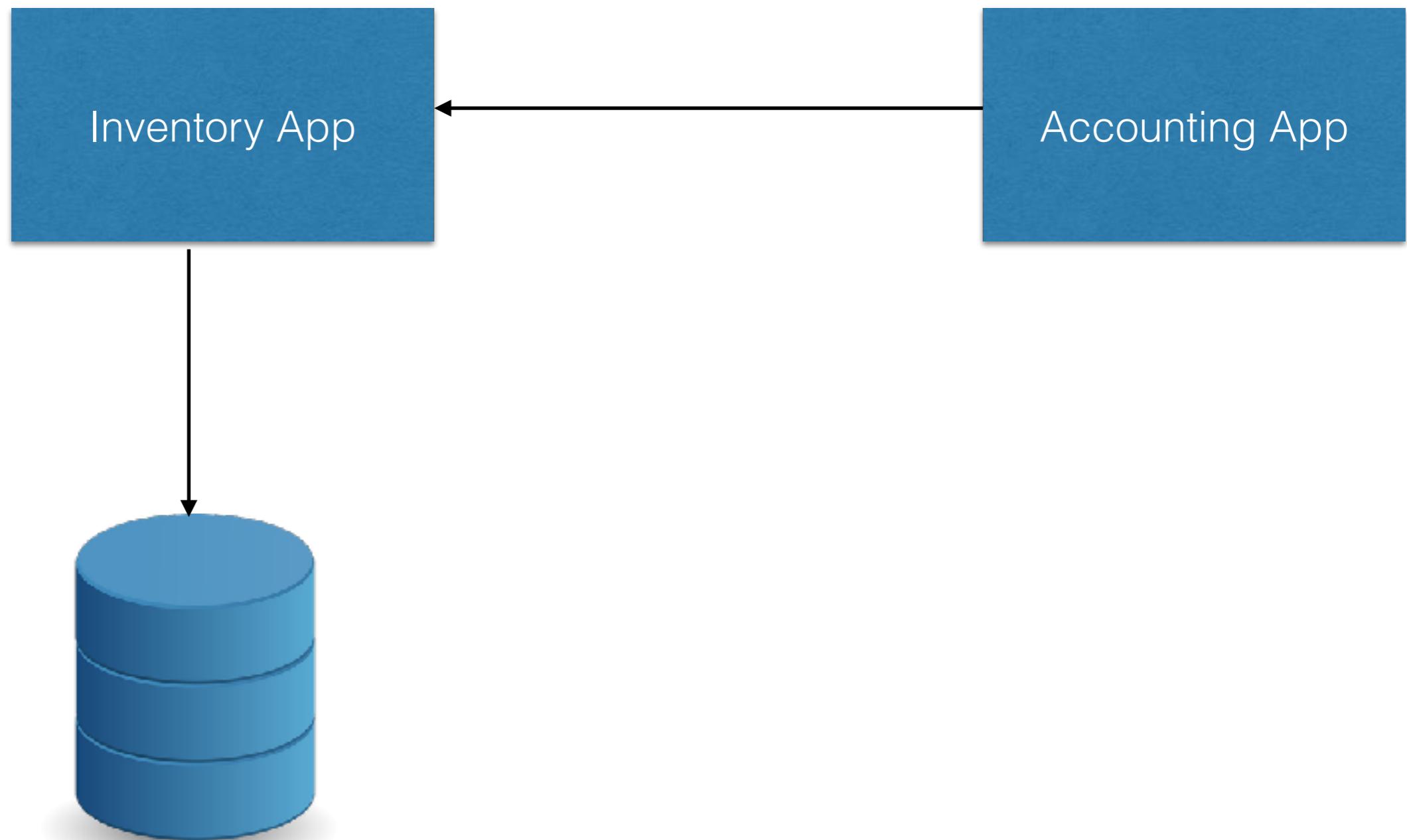


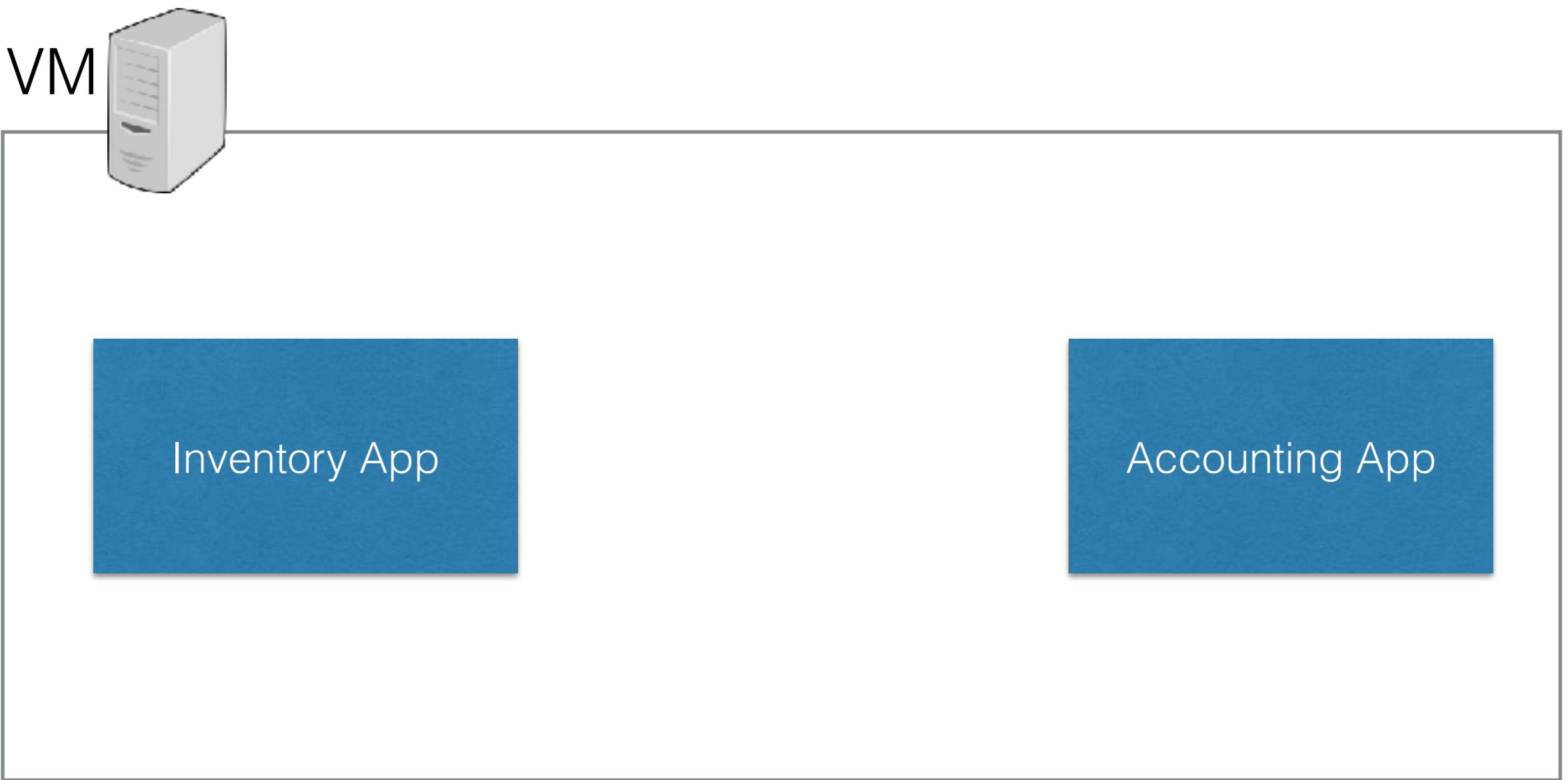
**Event data**



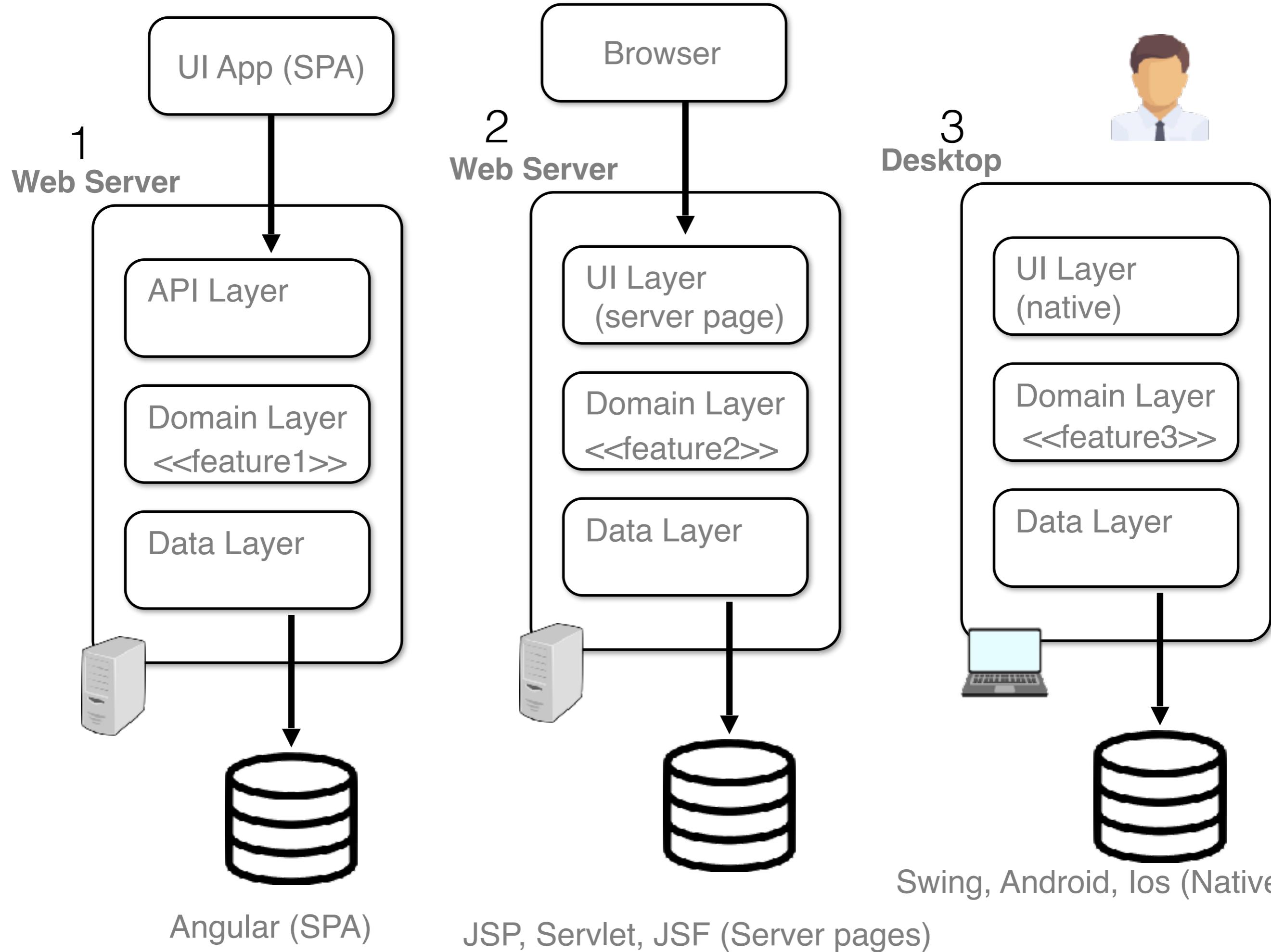
# Inventory Application

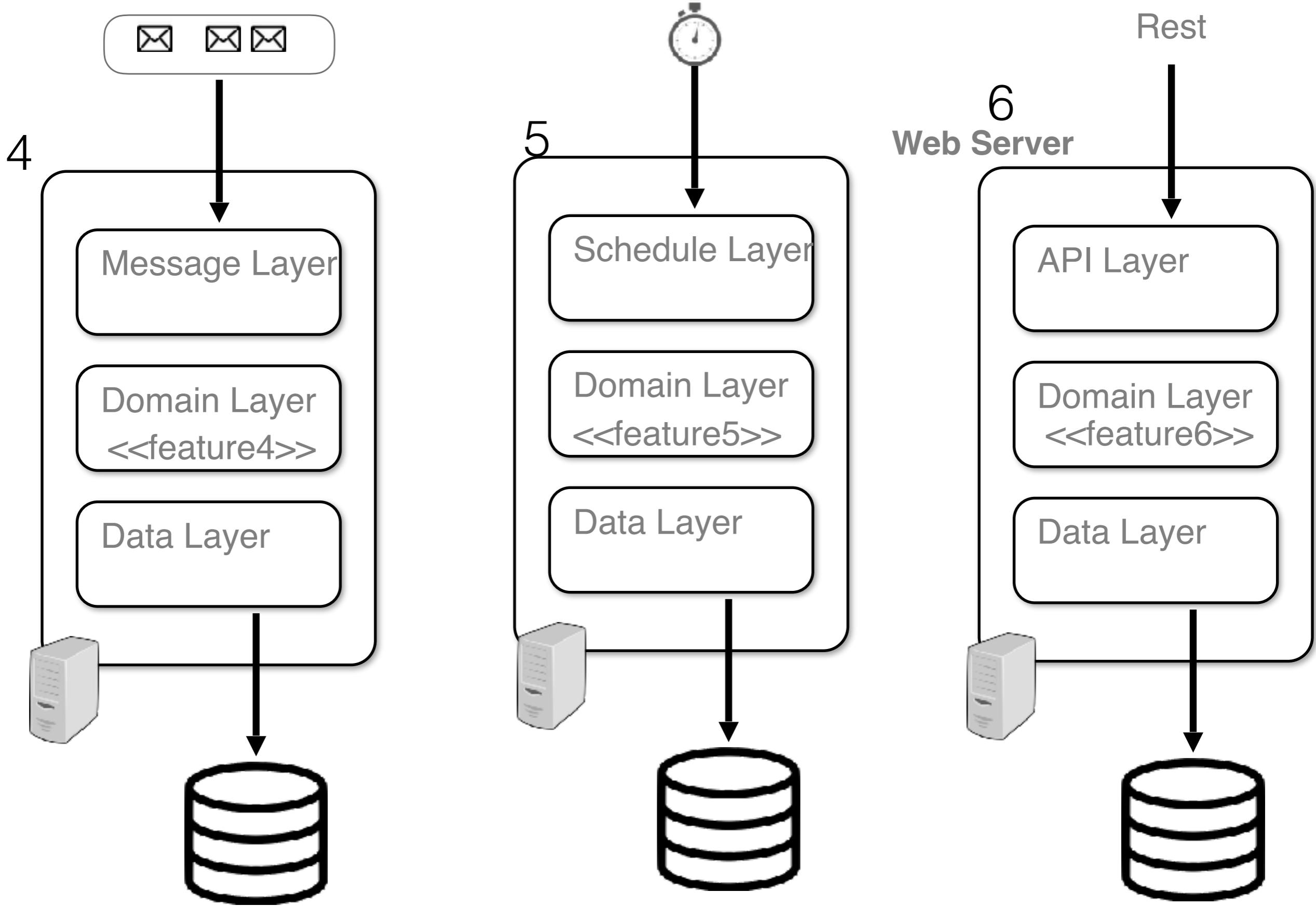


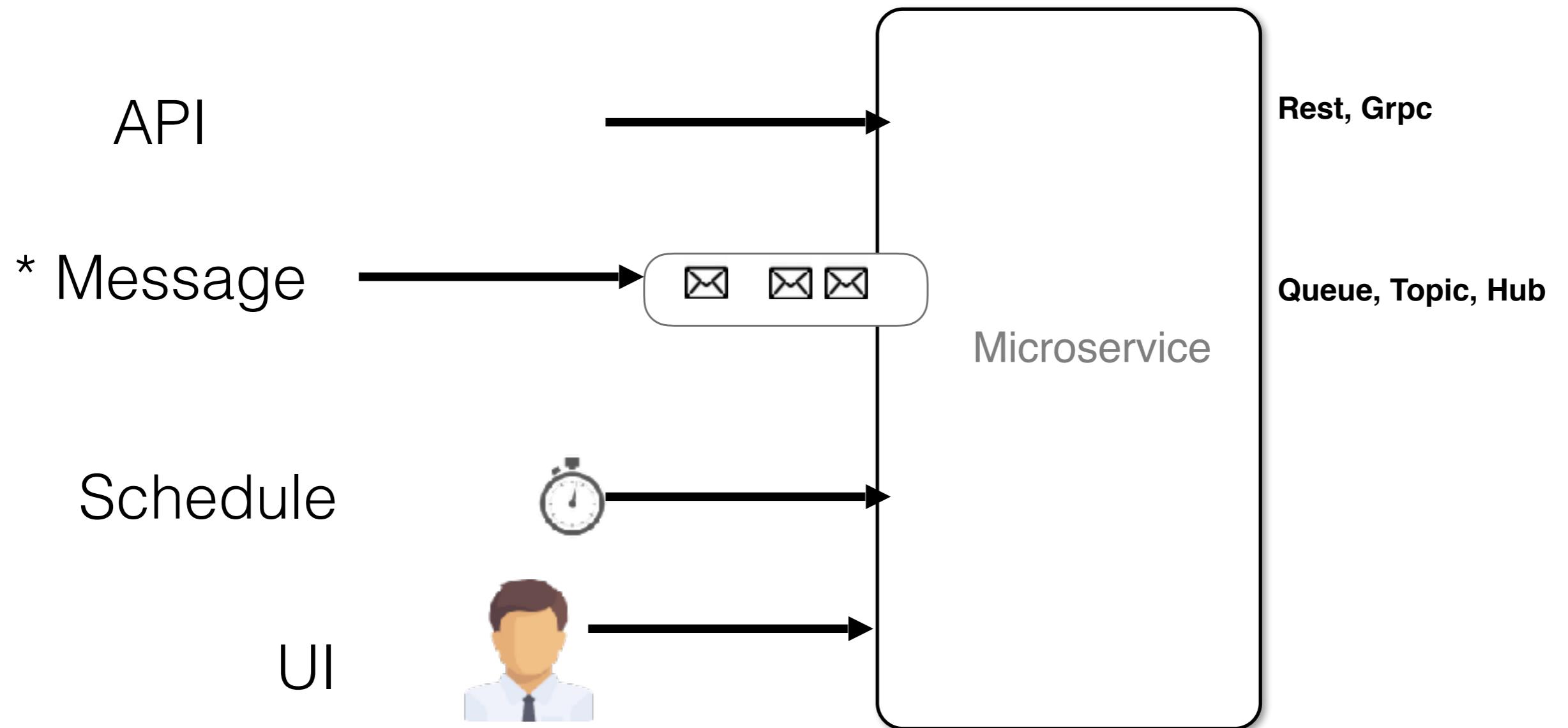




# Types of Microservice

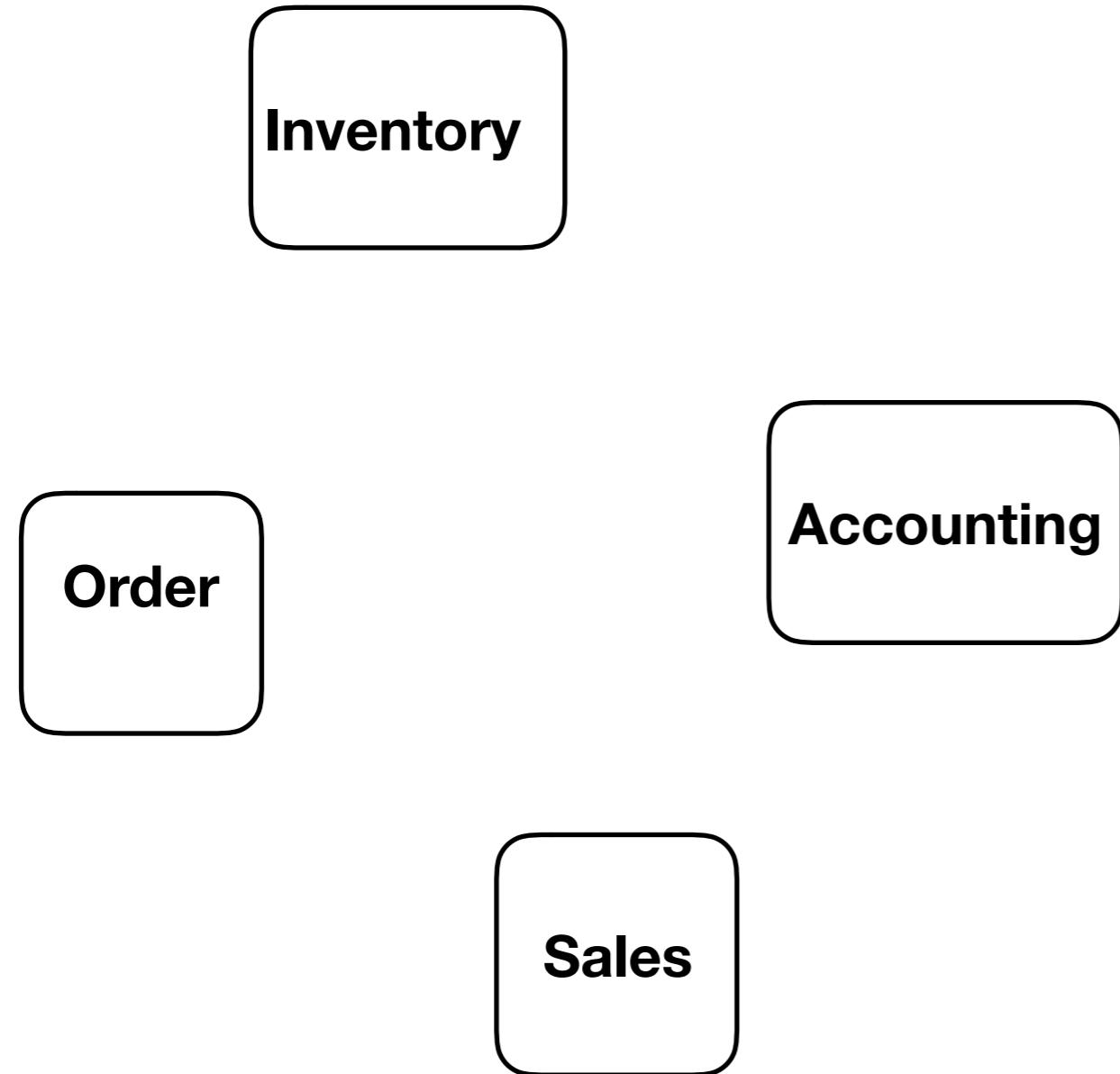






	Pros/	Solution
Development time	--	
Micro-service practices Learning Curve	--	
Resource Performance (CPU, Memory, I/O)	--	Grpc, protobuf, ..
Db Transaction Management	--	SAGA
Views / Report / Dash board/ join	--	Materialized View
Infra Cost	--	Containerization
First time Deployment effort	--	Automation
Debugging, Error Handling (End to End)	--	Centralize - Distributed Tracing (Jaeger)
Integration Test		Pyramid
Log Mgmt (debug/ error )	--	Centralize EFK, ELK, Splunk
Config Mgmt	--	Centralize Consul, Zookeeper, ..
Authentication (who)	--	Centralize (Oauth2, OpenID connect, SAML, ...)
Authorization (what can they do)	--	Centralize (Claims)
Audit Log mgmt (who accessed what)	--	Centralize (Event Store, ...)
Monitoring / Alerting	--	Centralize
Data Security and Privacy (transit, storage)	--	
Build Pipeline (CI/CD)	--	Automation
Agile Architecture (Agility to change)	+++	fwk, technology, platform, ...
Feature Shipping (Agility to ship)/ CD	+++	Timing of when to goto production
Scalability (volume - request, data, compute)	+	
Availability	+	
Ability to do Polygot	+	
Fault Isolation	+	

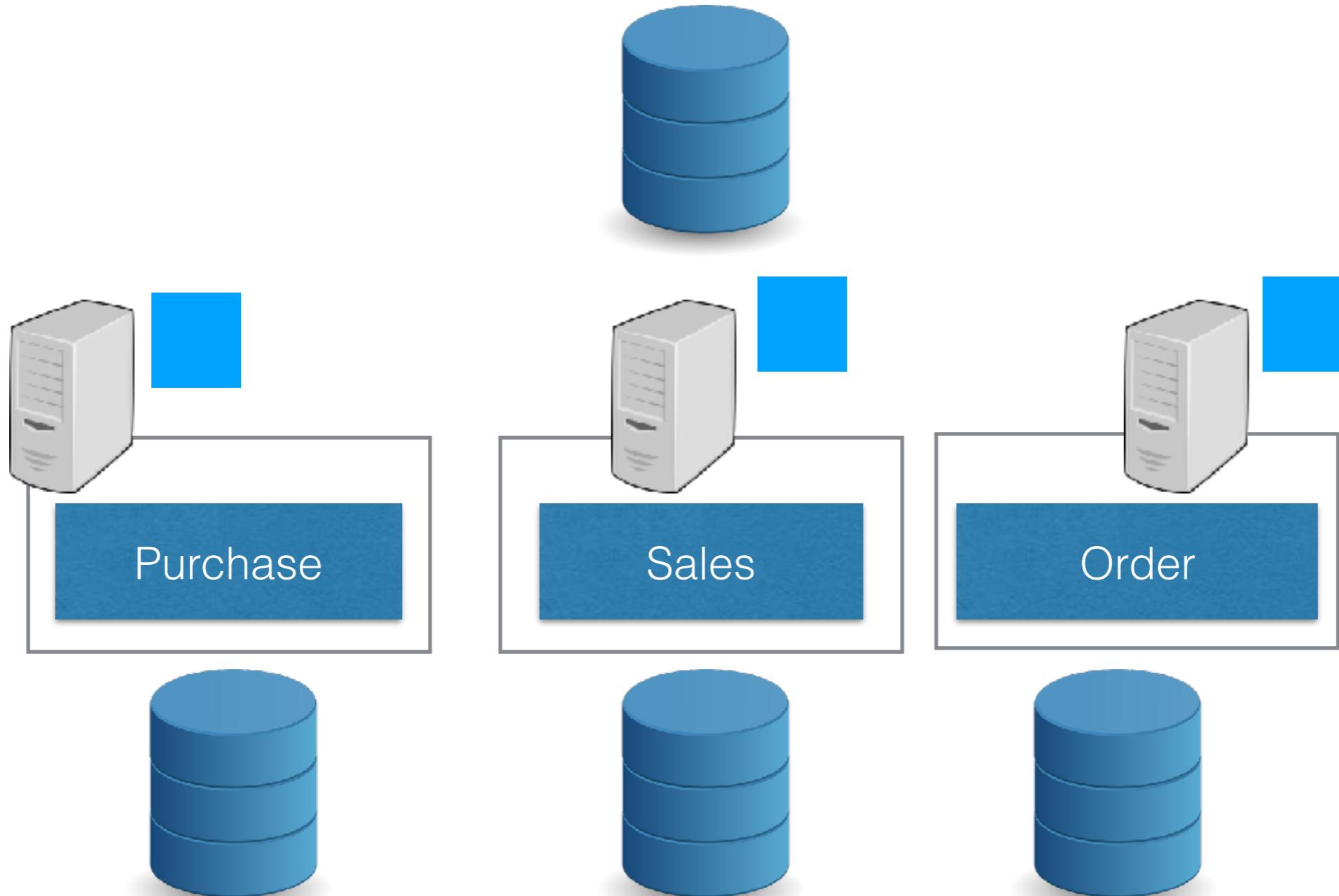
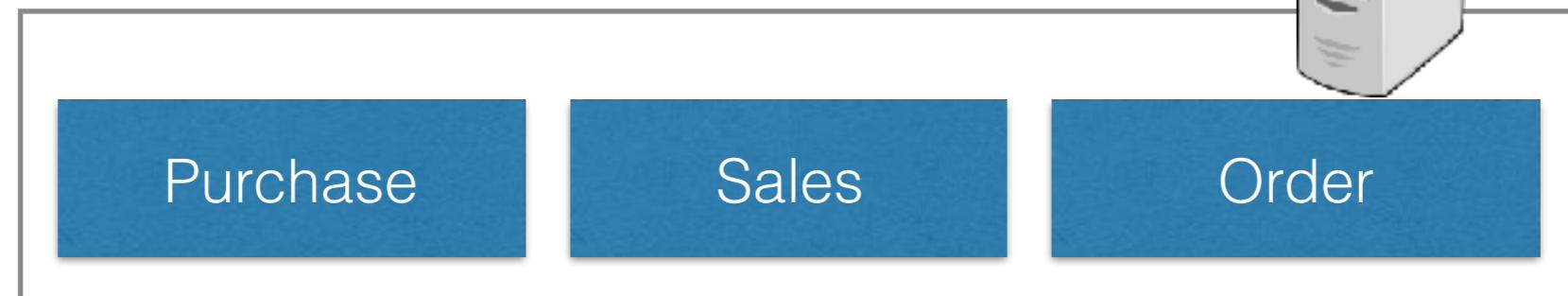
## De centralize domain



## centralize Common Concerns

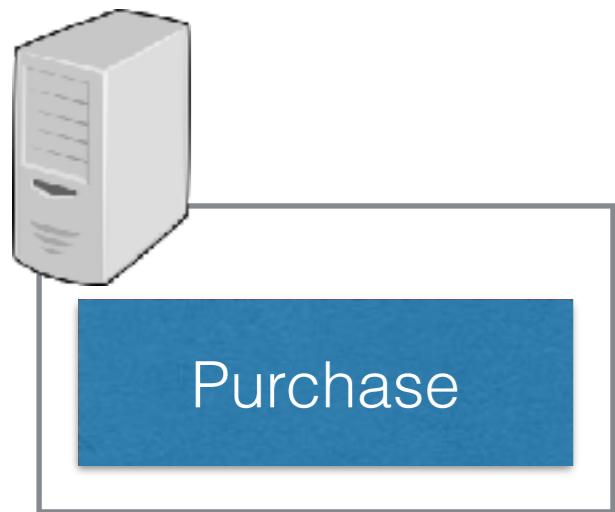
# Authentication  
# Authorization  
# Auditing  
# config  
# log  
# tracing  
# monitoring

# Inventory Application



## 1 phase commit

```
db.Begin()  
Cmd  
Cmd  
Cmd  
db.Commit()
```



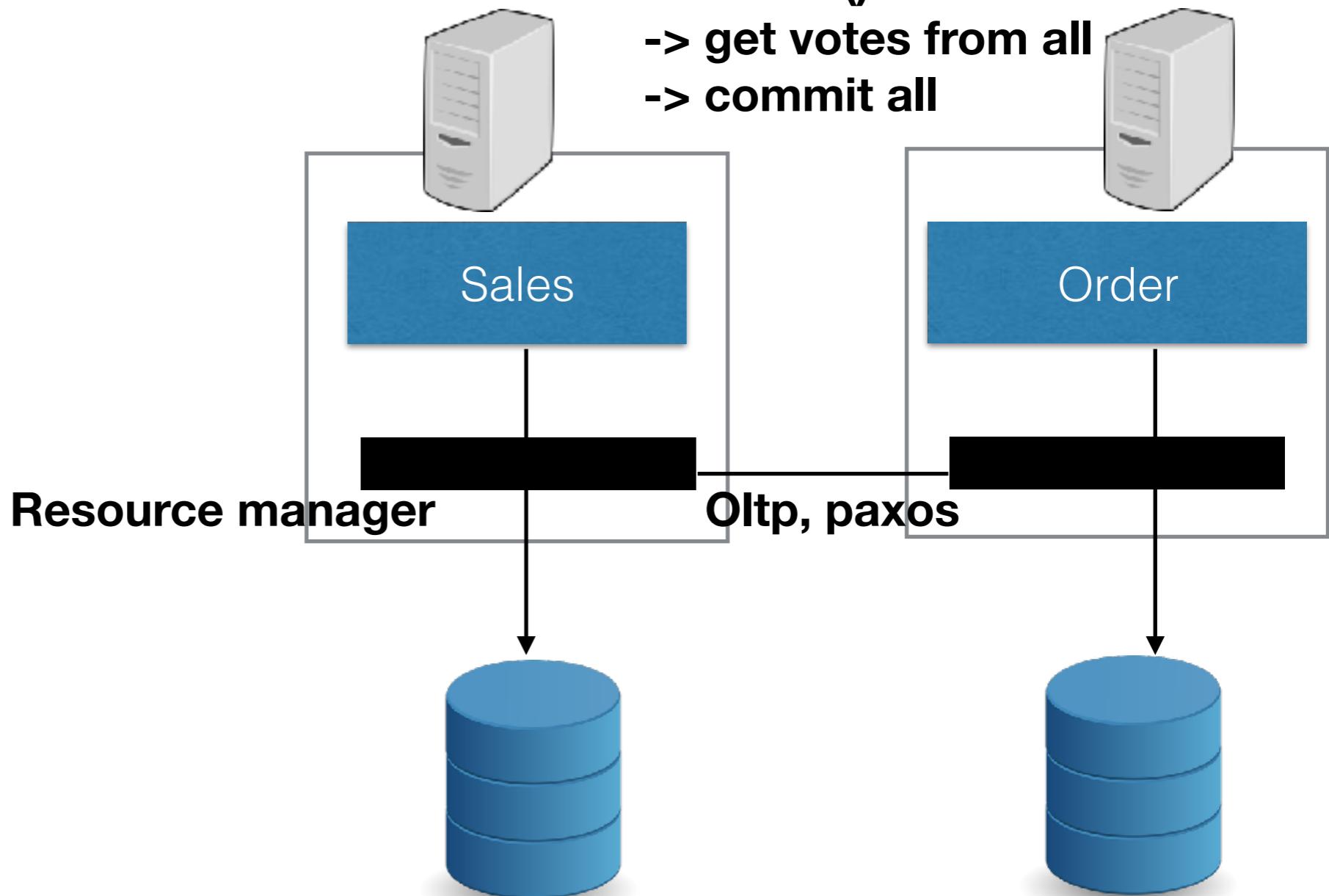
## 2 phase commit

```
rm.Begin()  
Cmd on db1  
Cmd on db2  
Cmd on db 1
```

...

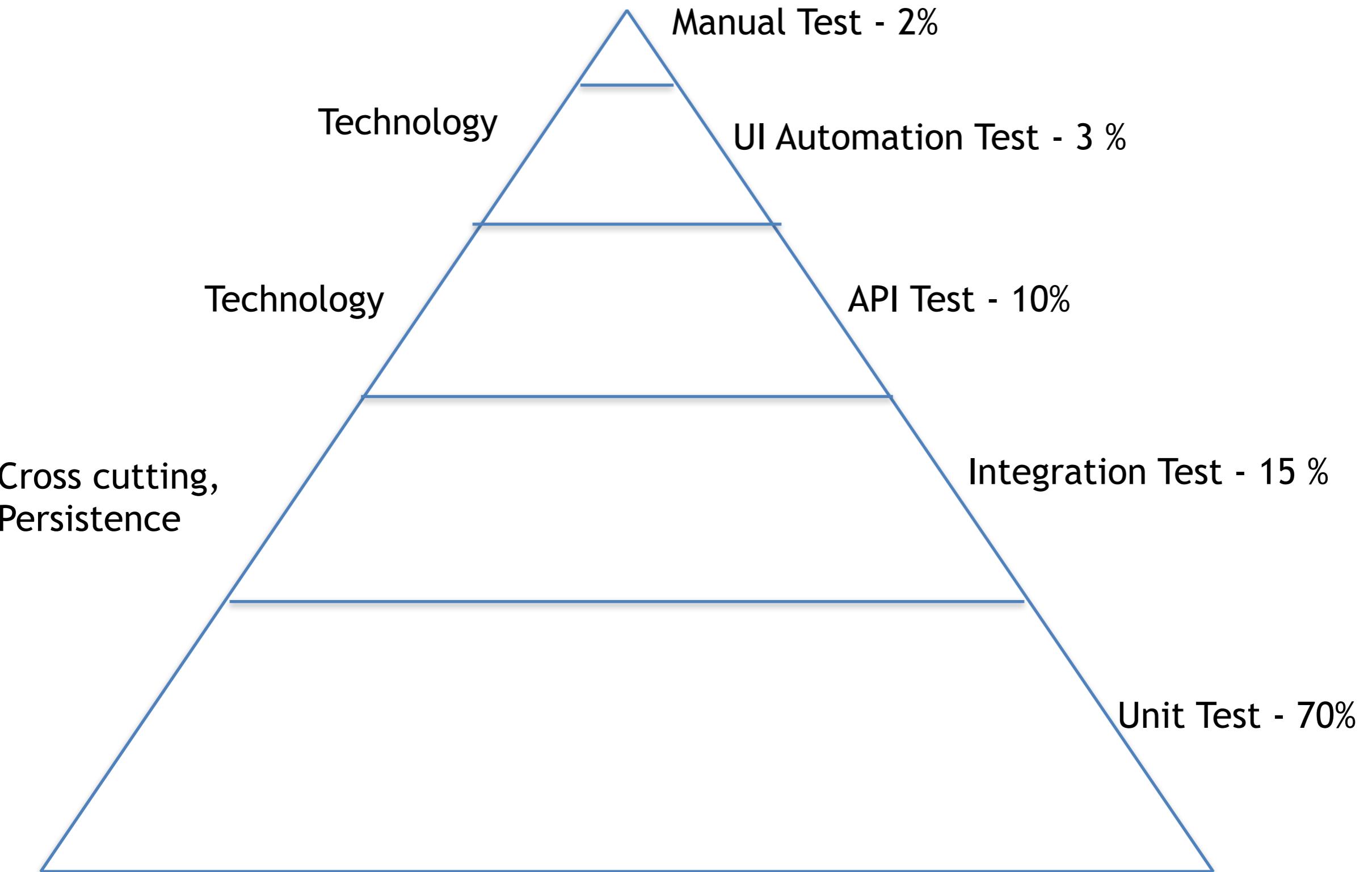
```
rm.Commit()
```

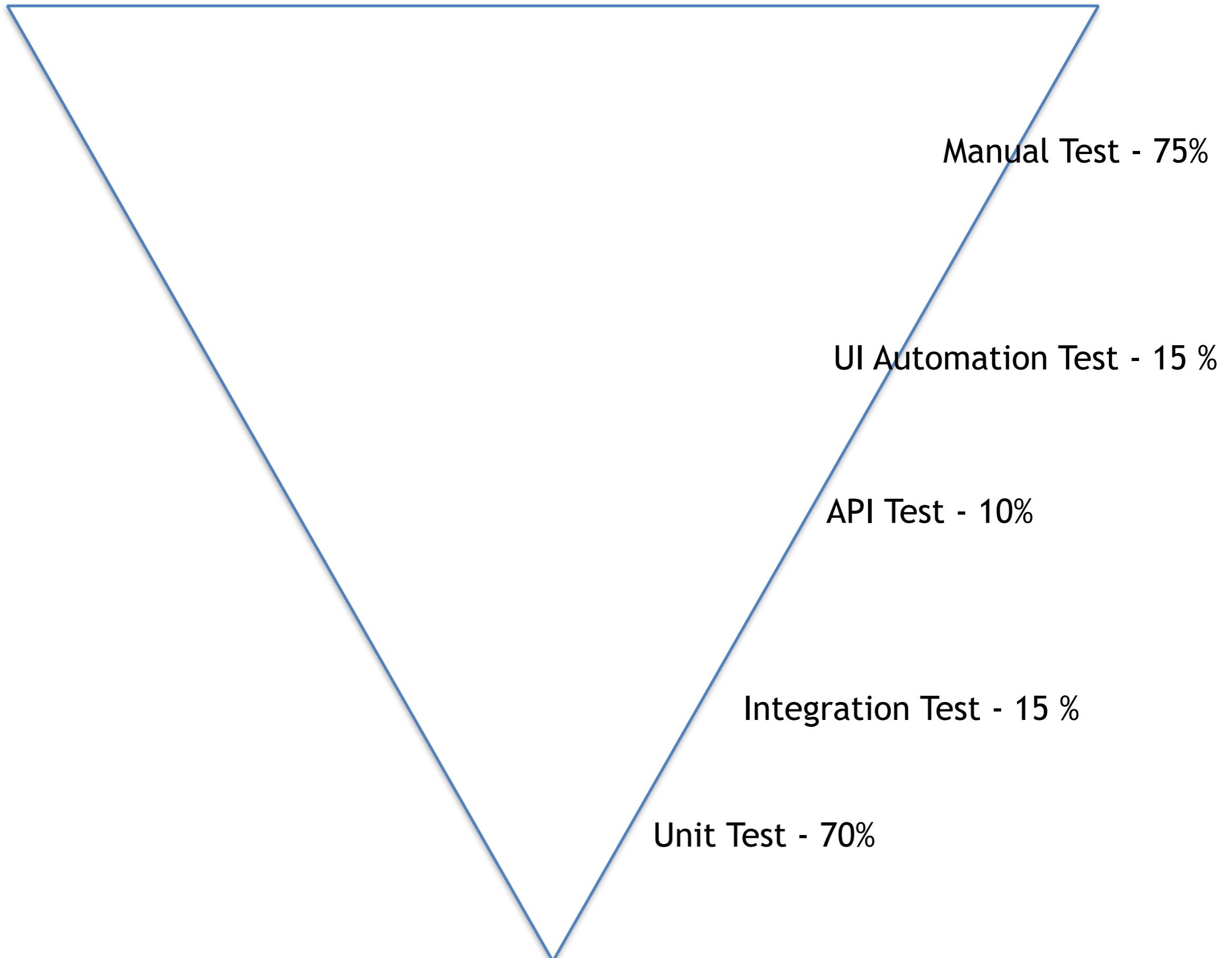
- > get votes from all
- > commit all



Decentralize Domain  
Centralize Tech

# Pyramid test

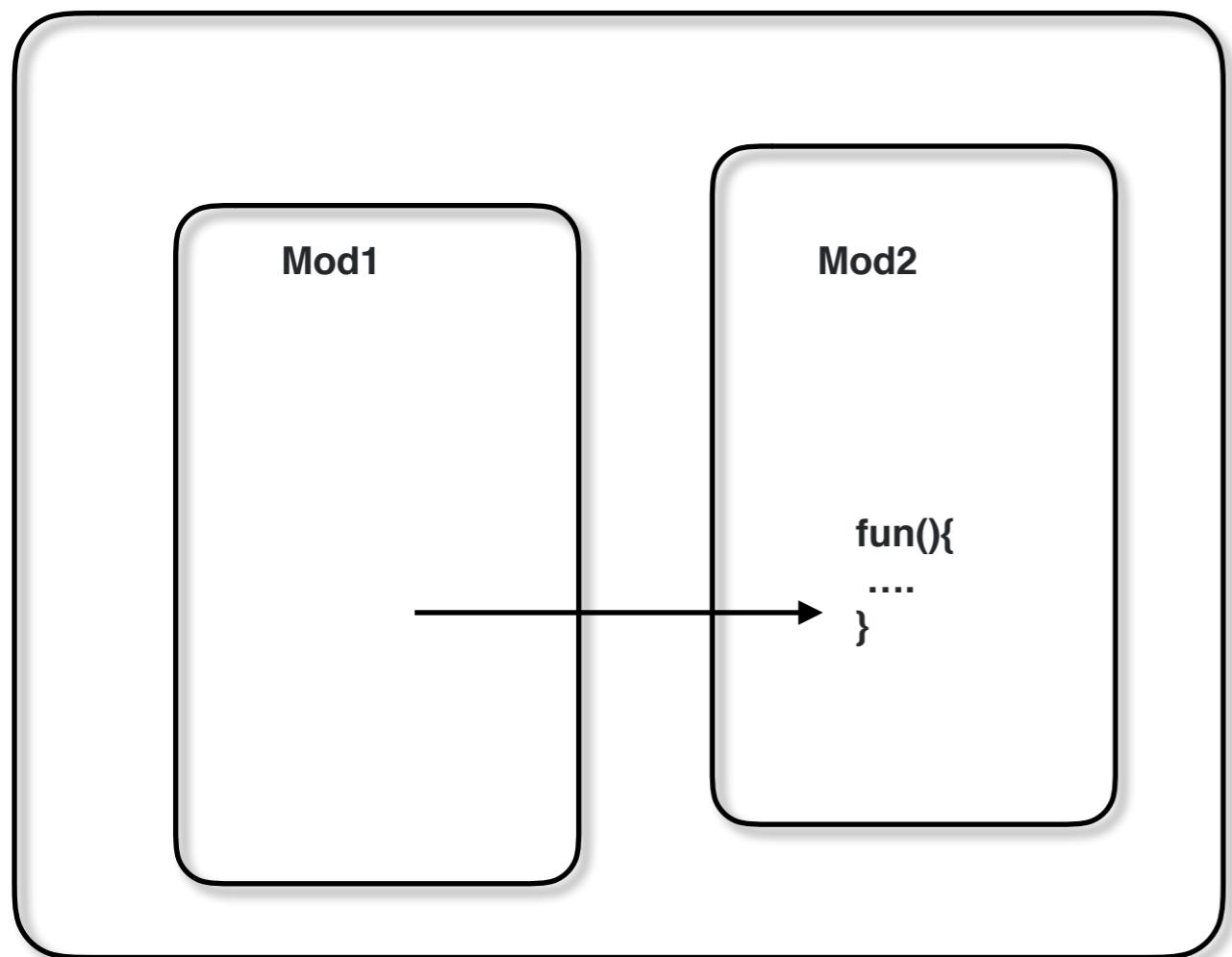




# 1. Performance

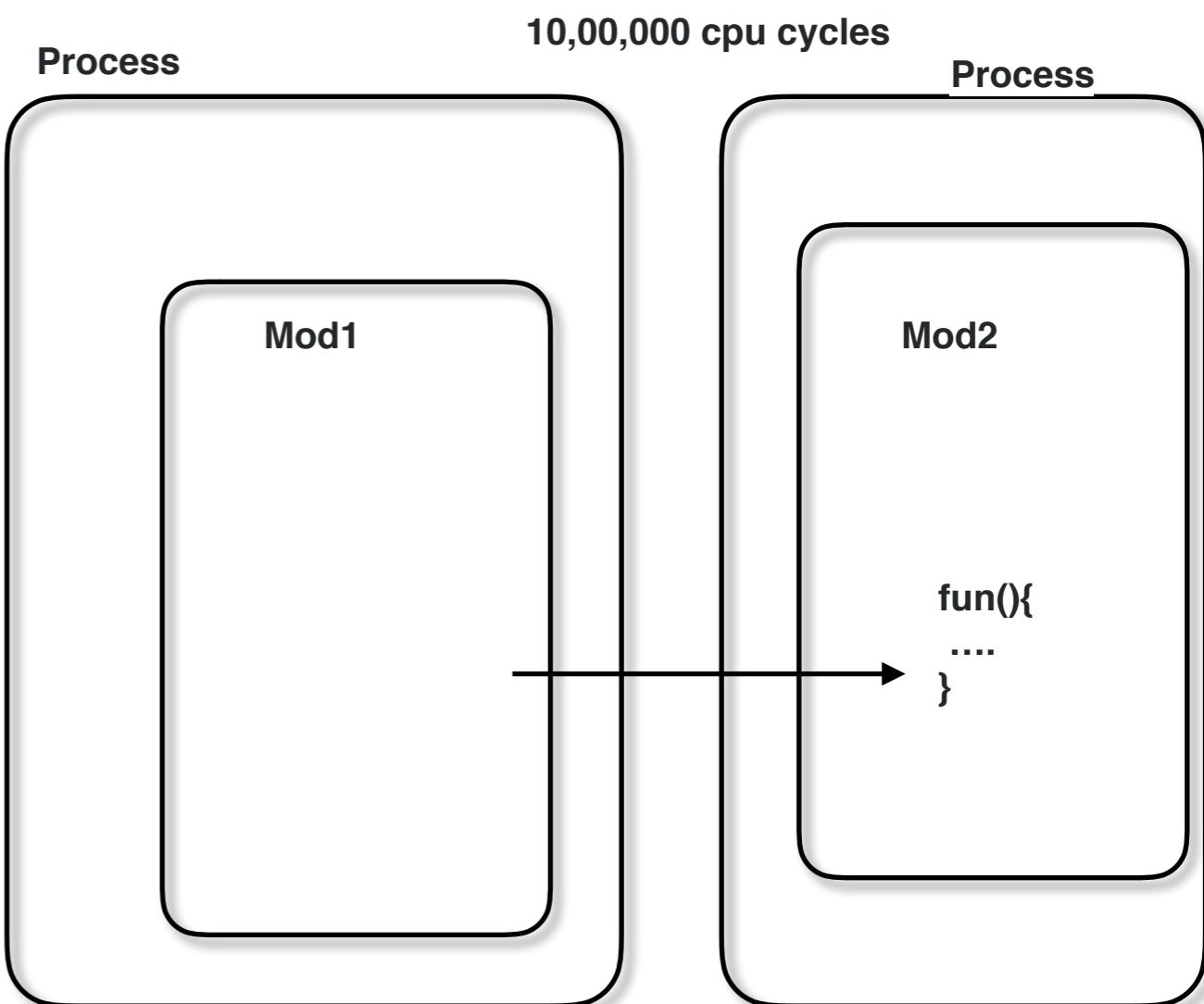
Operation	Cpu Cycles
• 10 + 12	3
• Calling a in-memory Method	10 ~
• Create Thread	2,00,000
• Destroy Thread	1,00,000
• Database Call	40,00,000
• Distributed Fun Call	20,00,000
Write to disk	10,00,000

### Process



10 cpu cycles

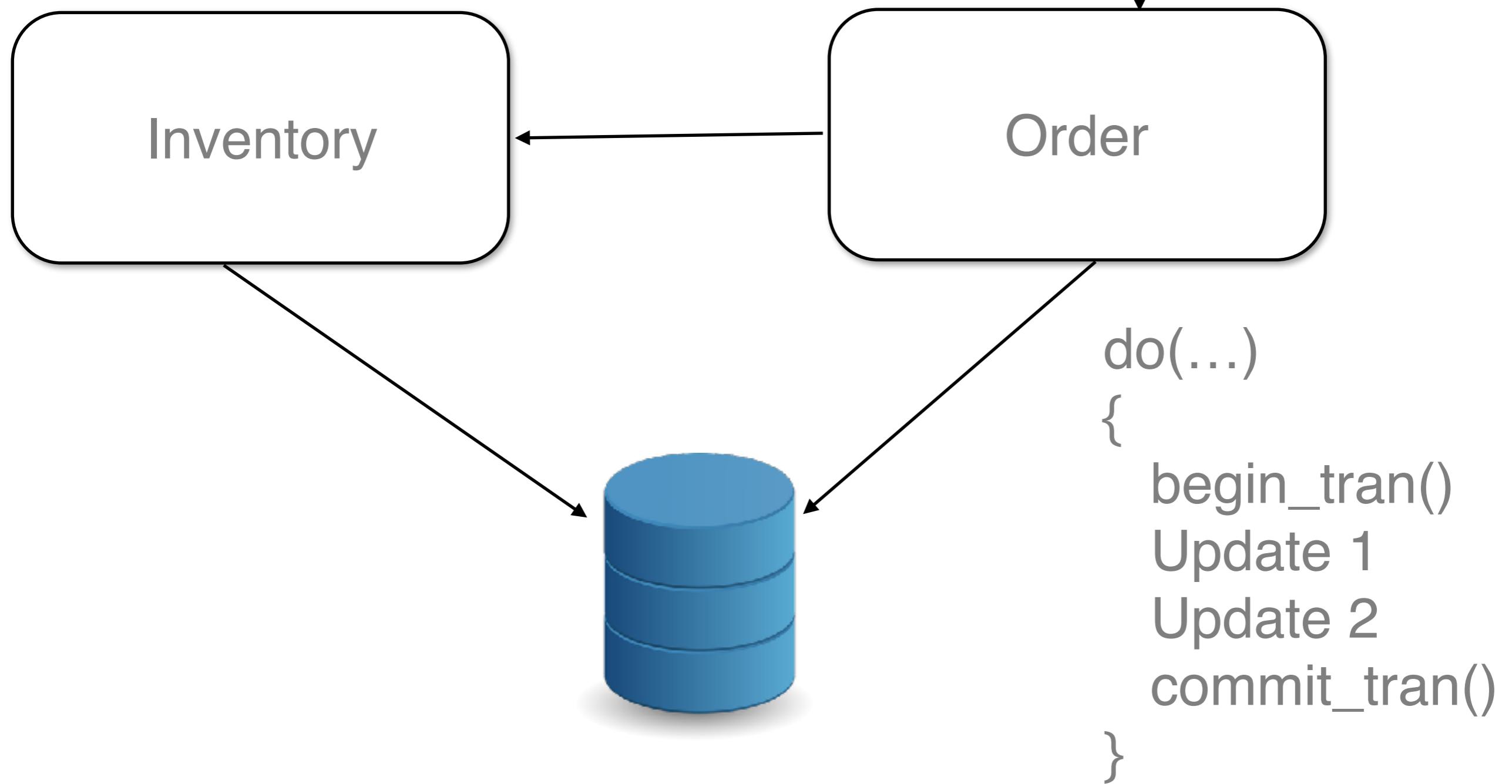
### Process



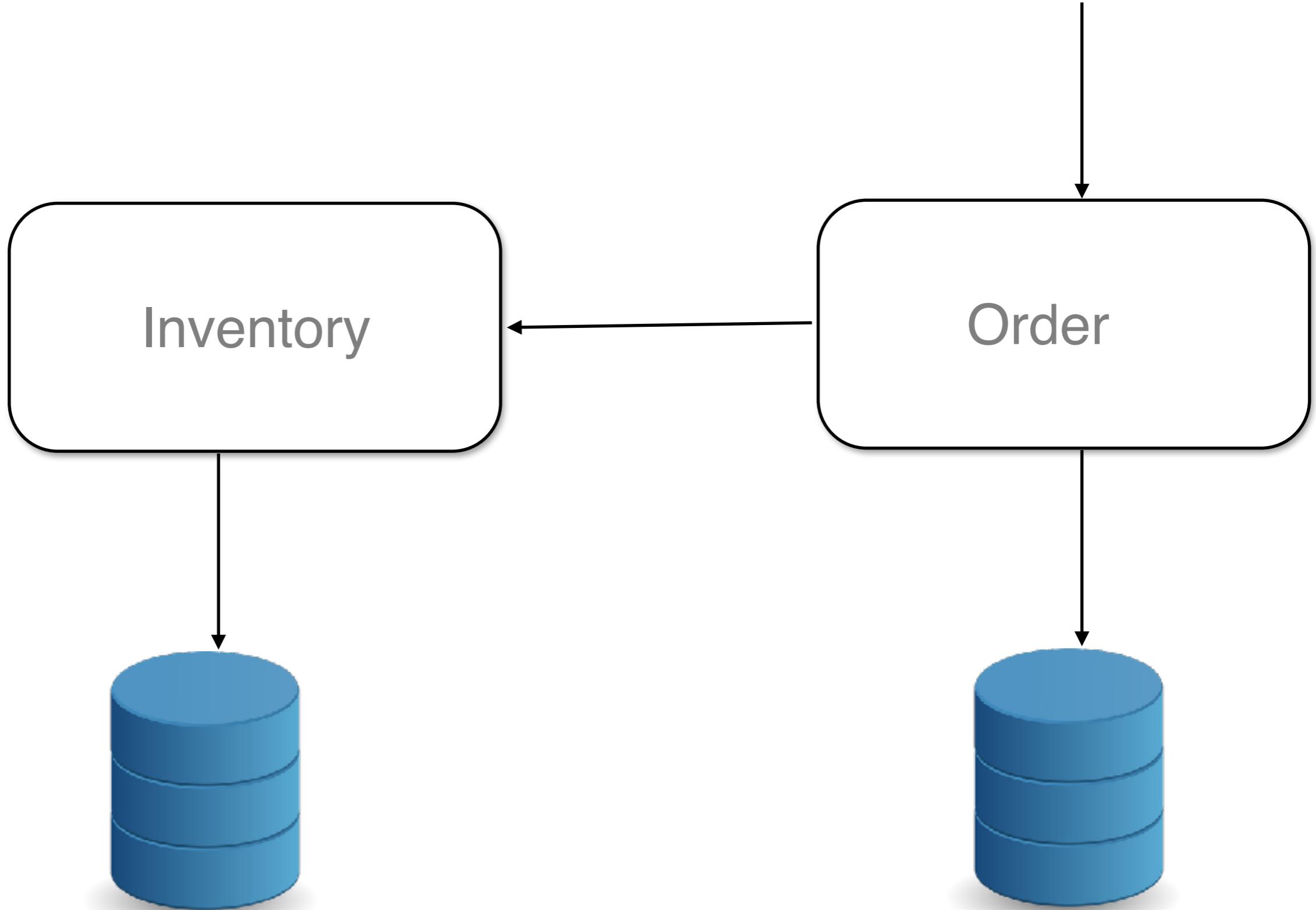
10,00,000 cpu cycles

### Process

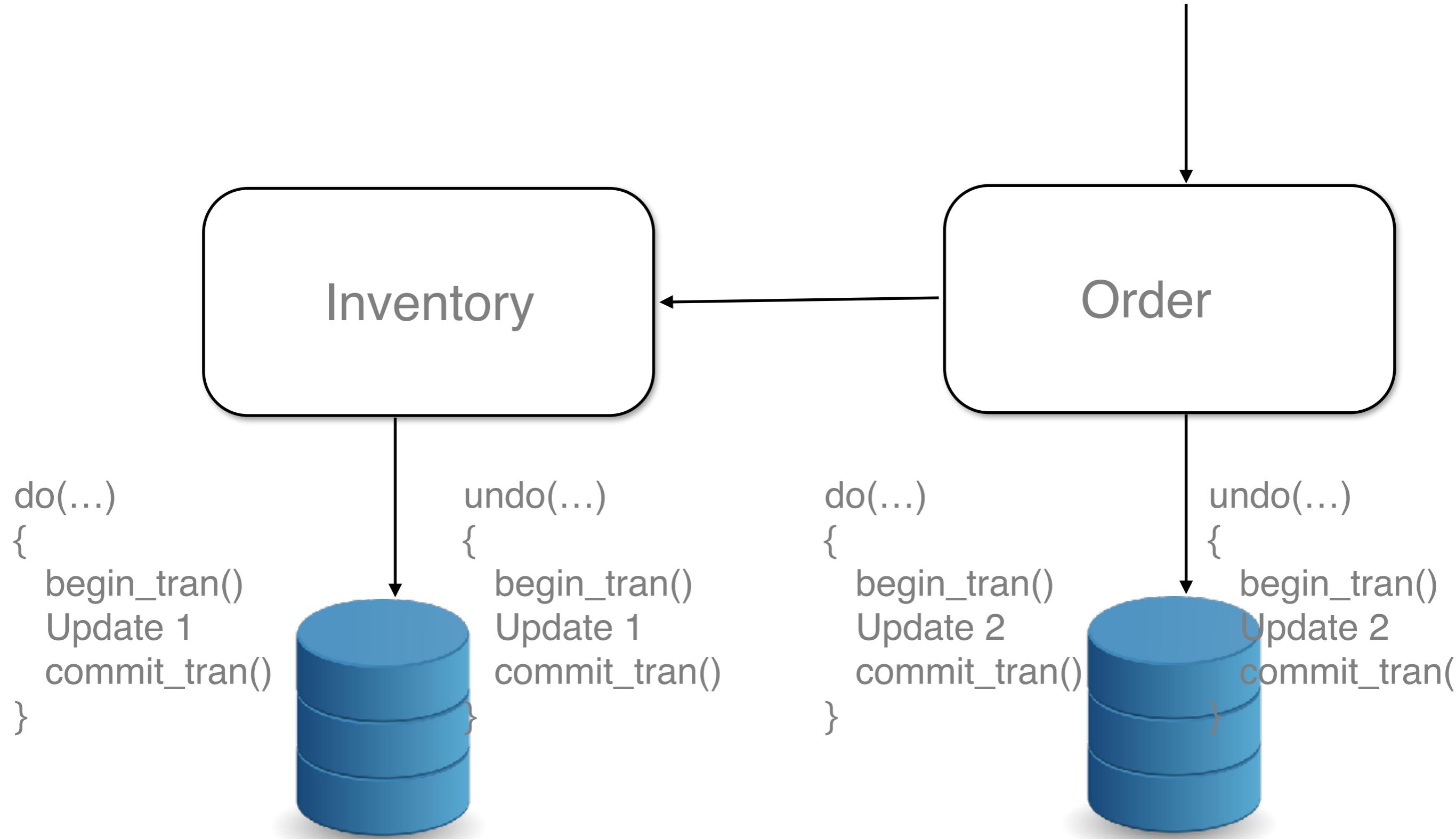
## 2. Db transaction

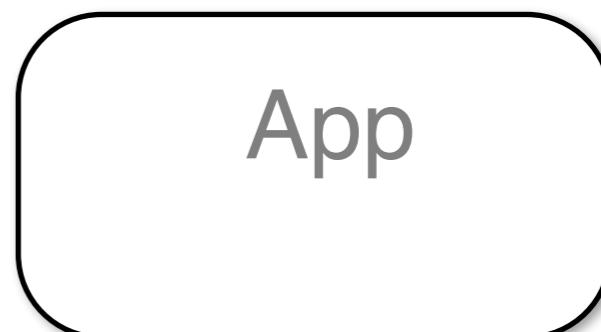
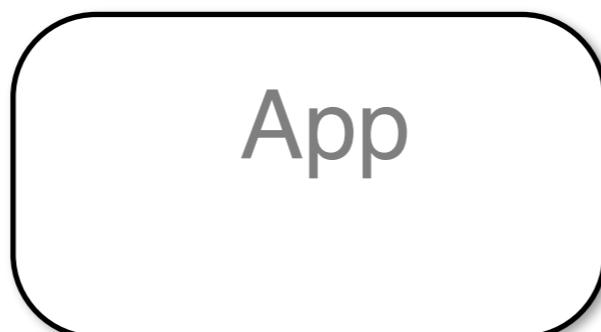
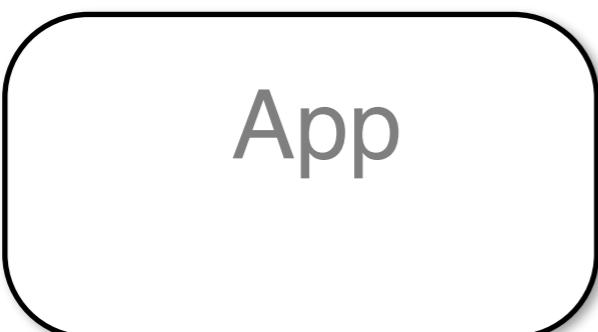
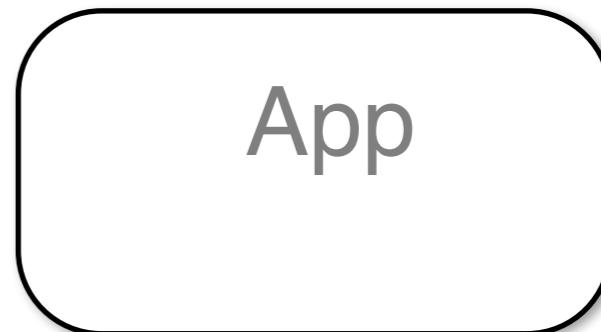
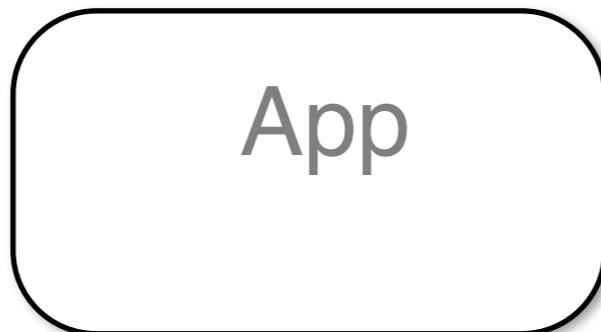
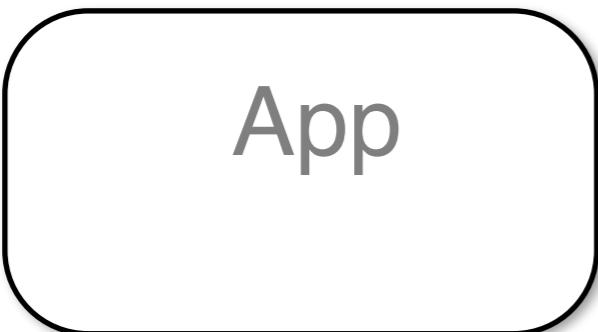


## 2. Db transaction



## 2. Db transaction





App

Mod

Mod

Mod



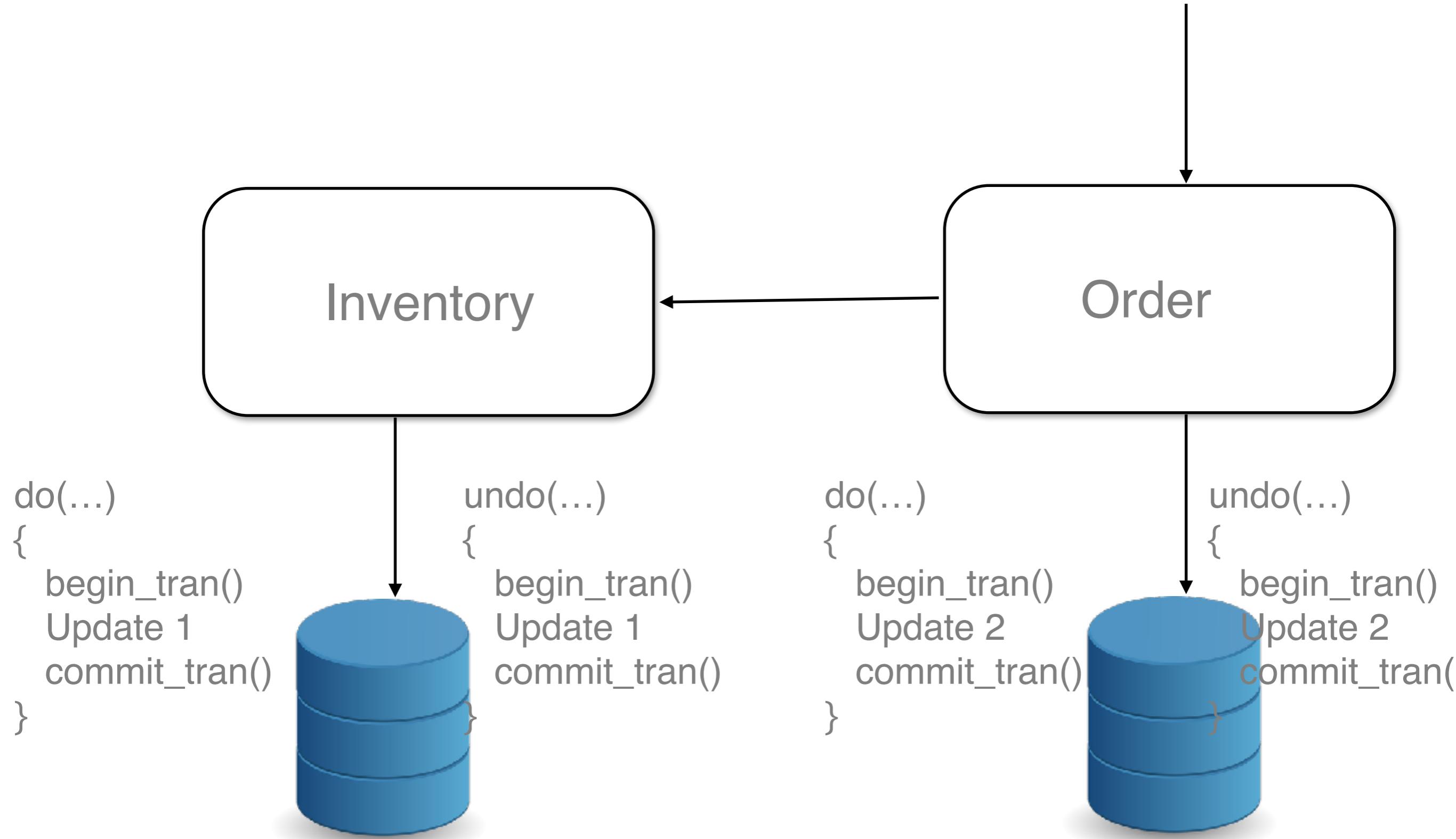
App

App

App

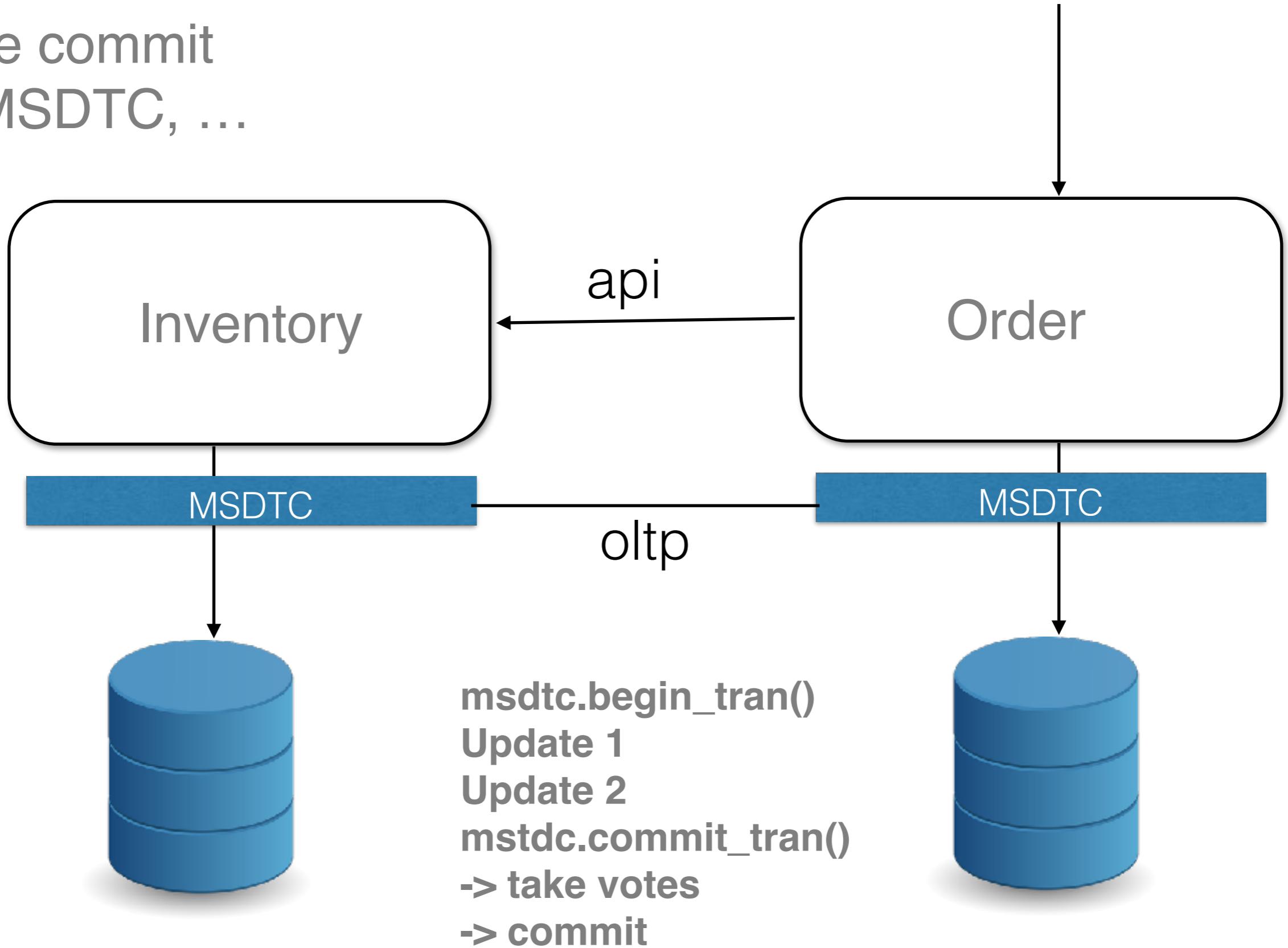


## 2. Db transaction

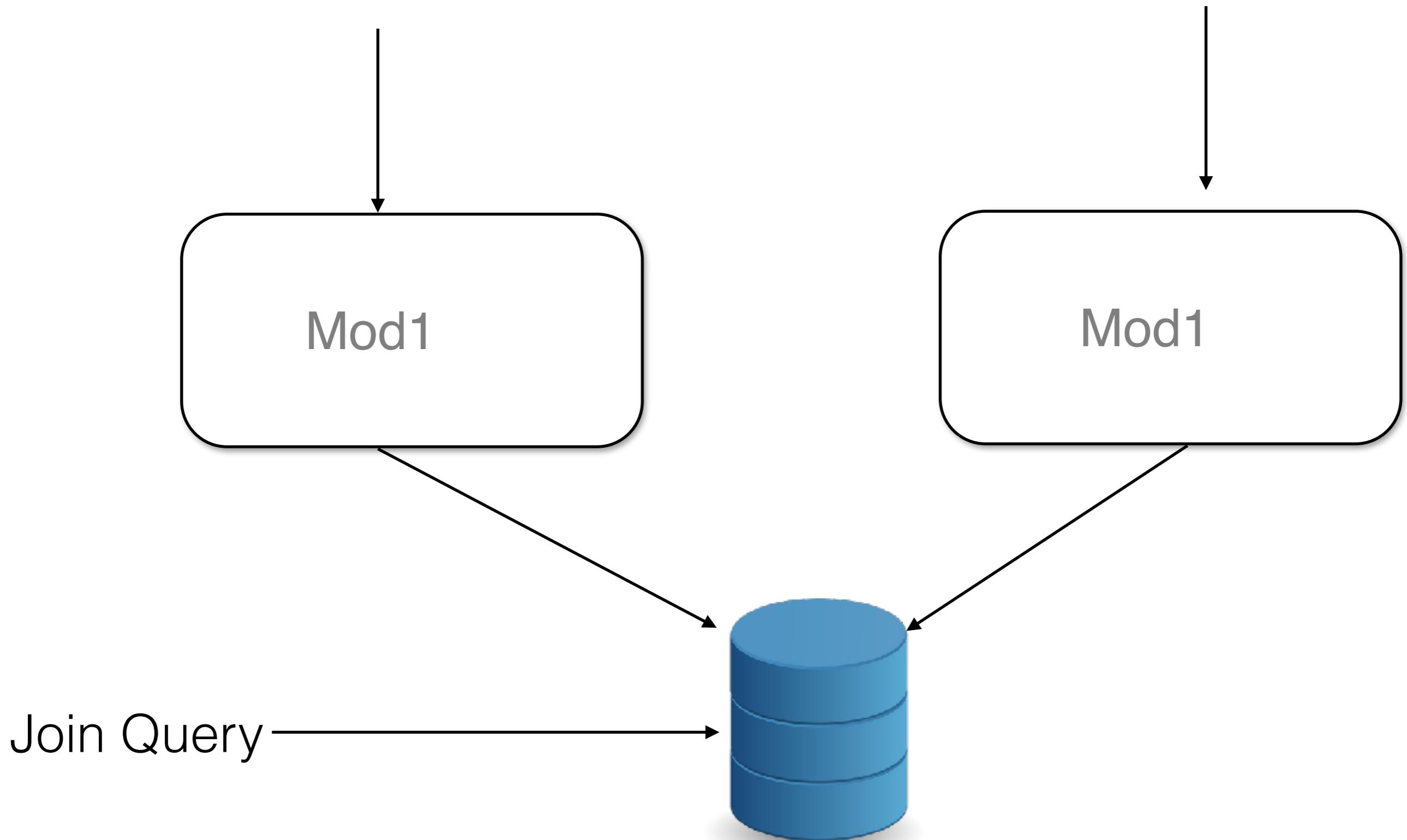


## 2. Db transaction

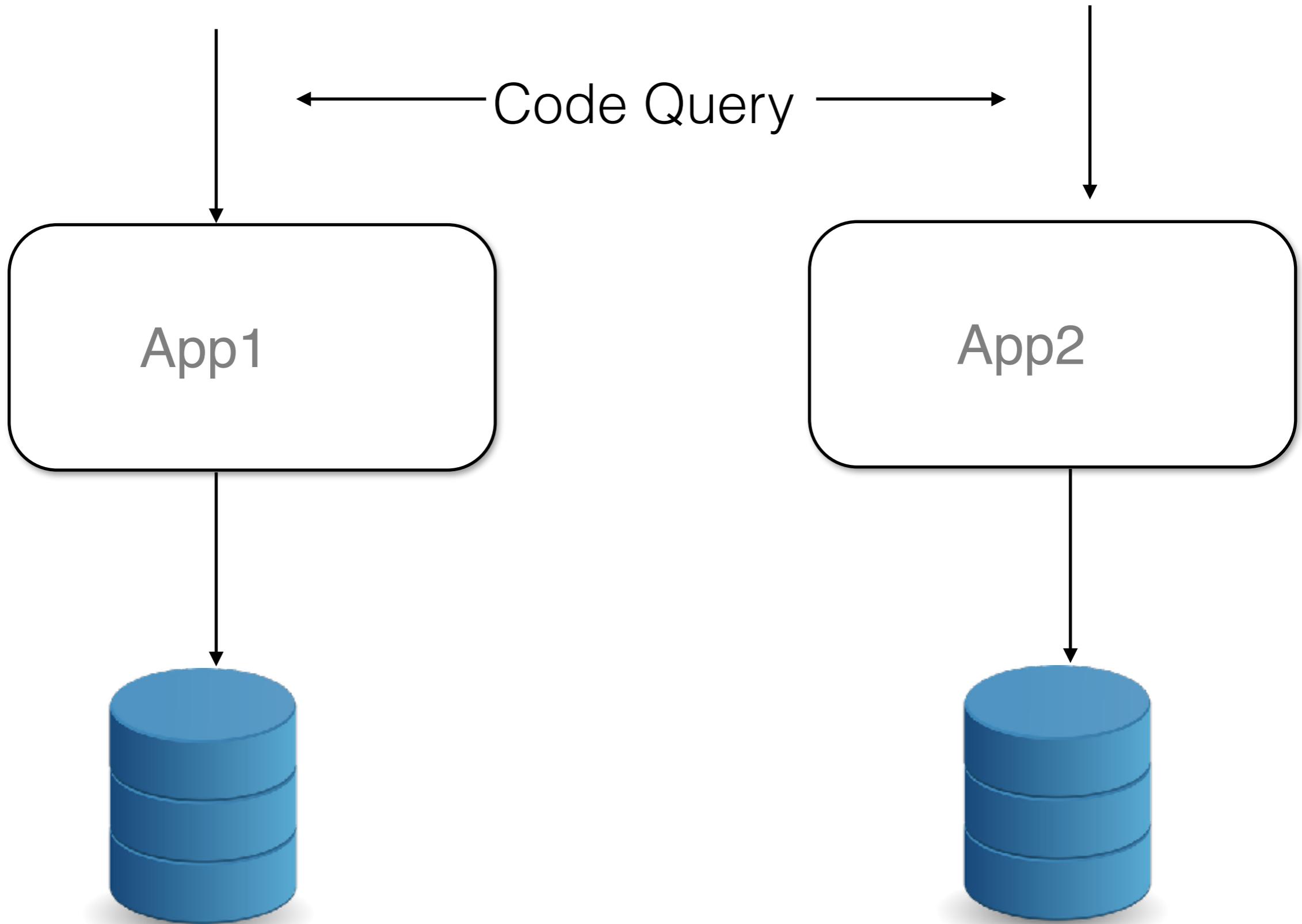
2 phase commit  
JTX , MSDTC, ...



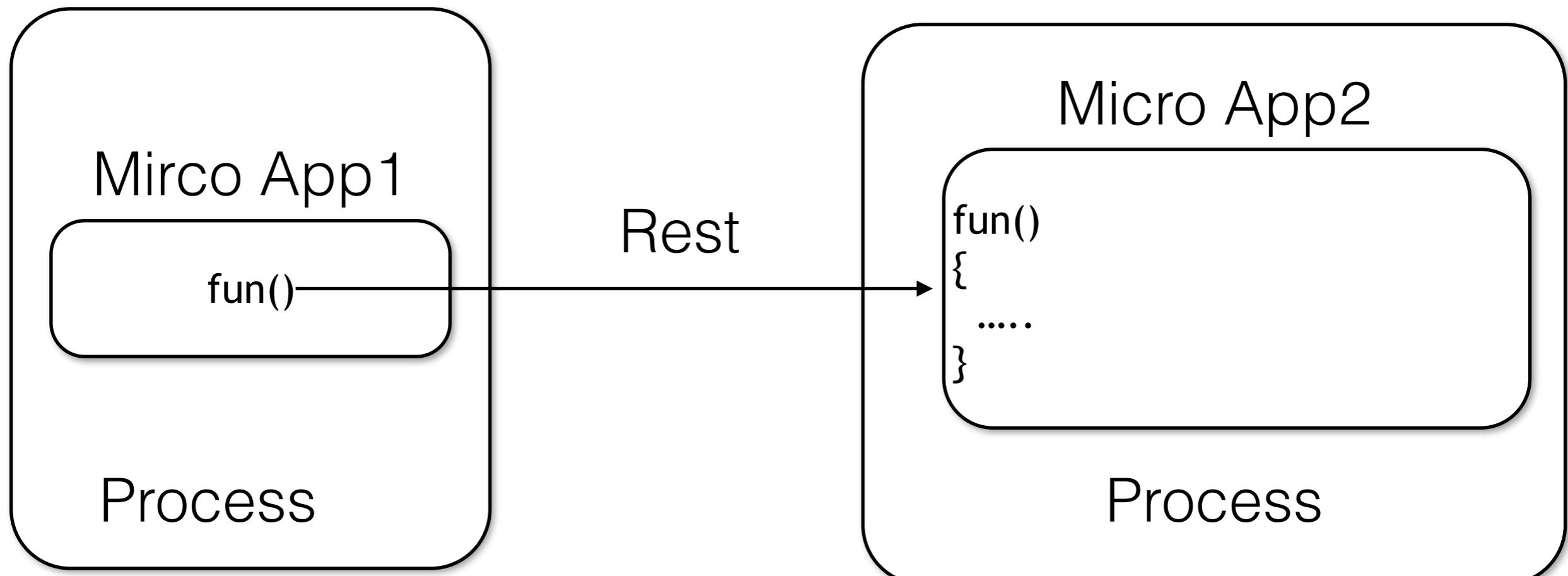
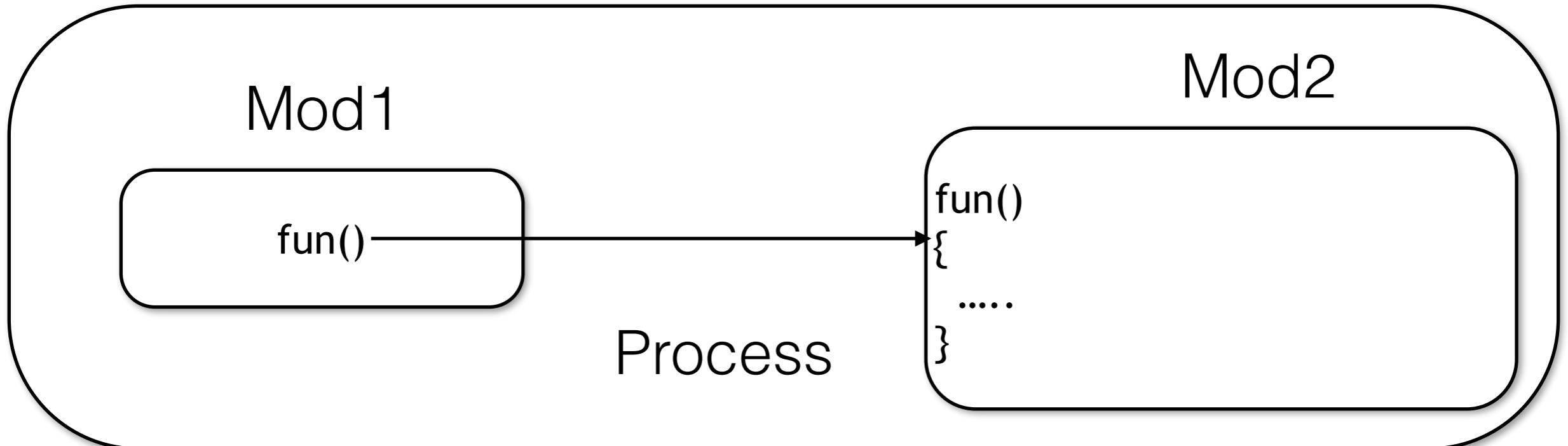
### 3. Db query



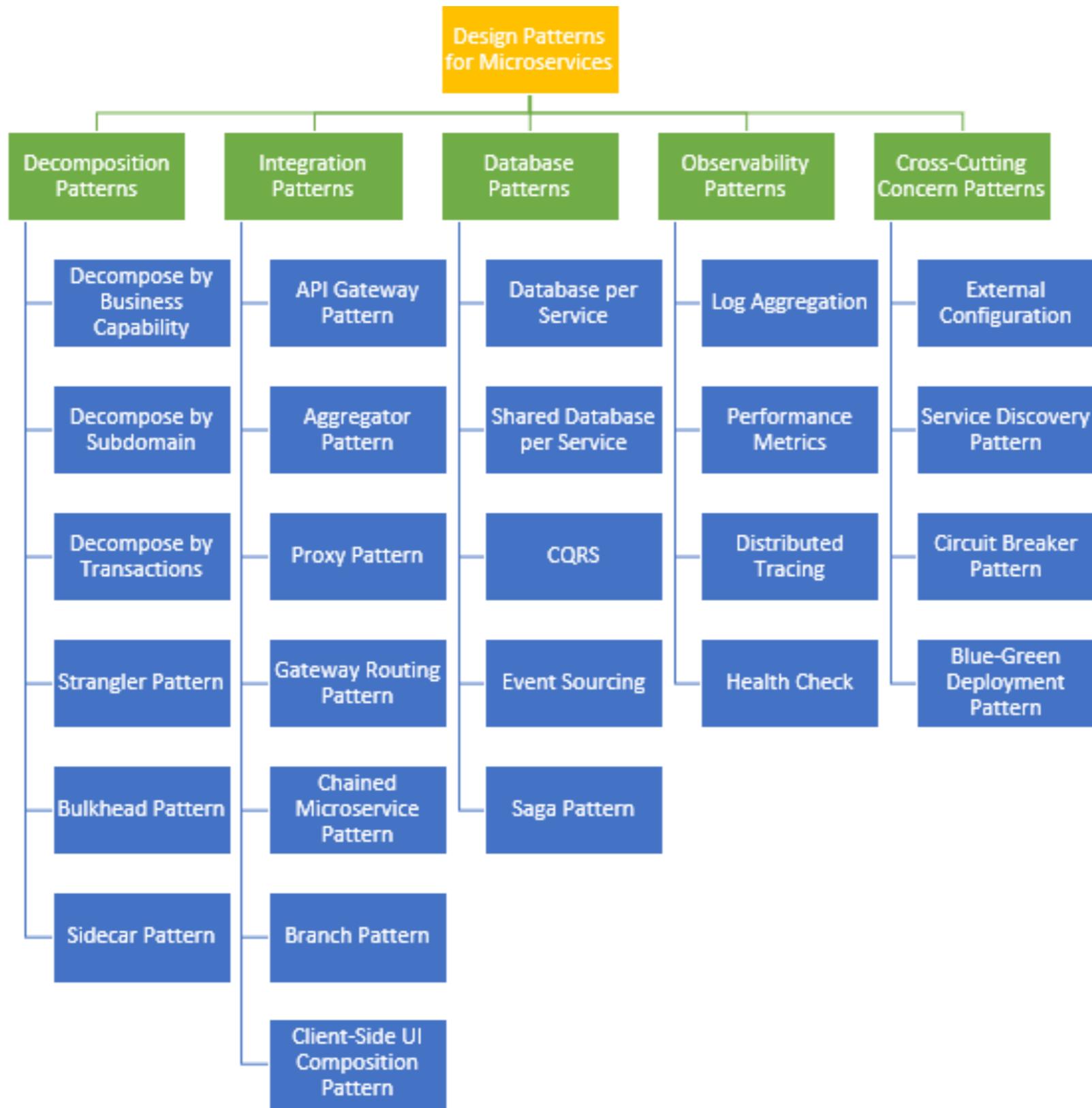
# 3. Query



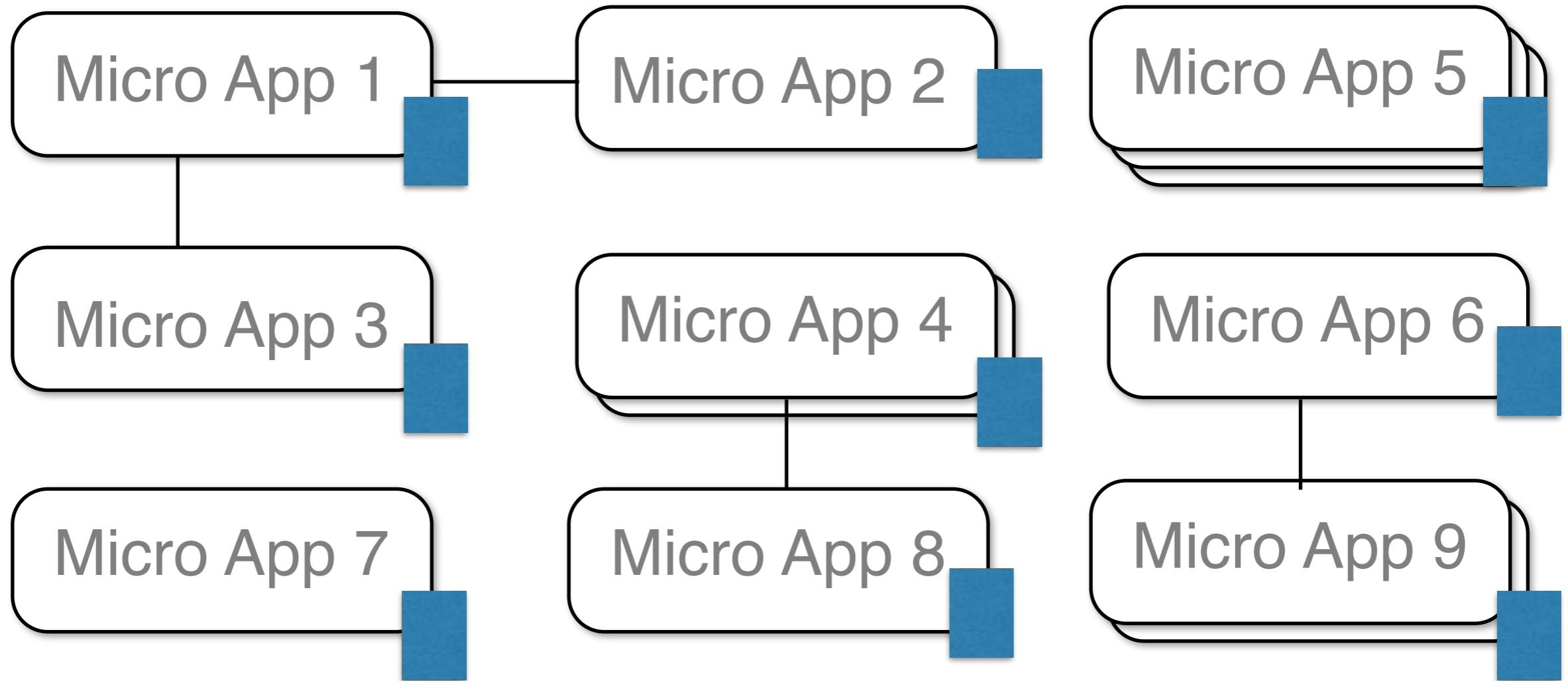
# 4. Development time



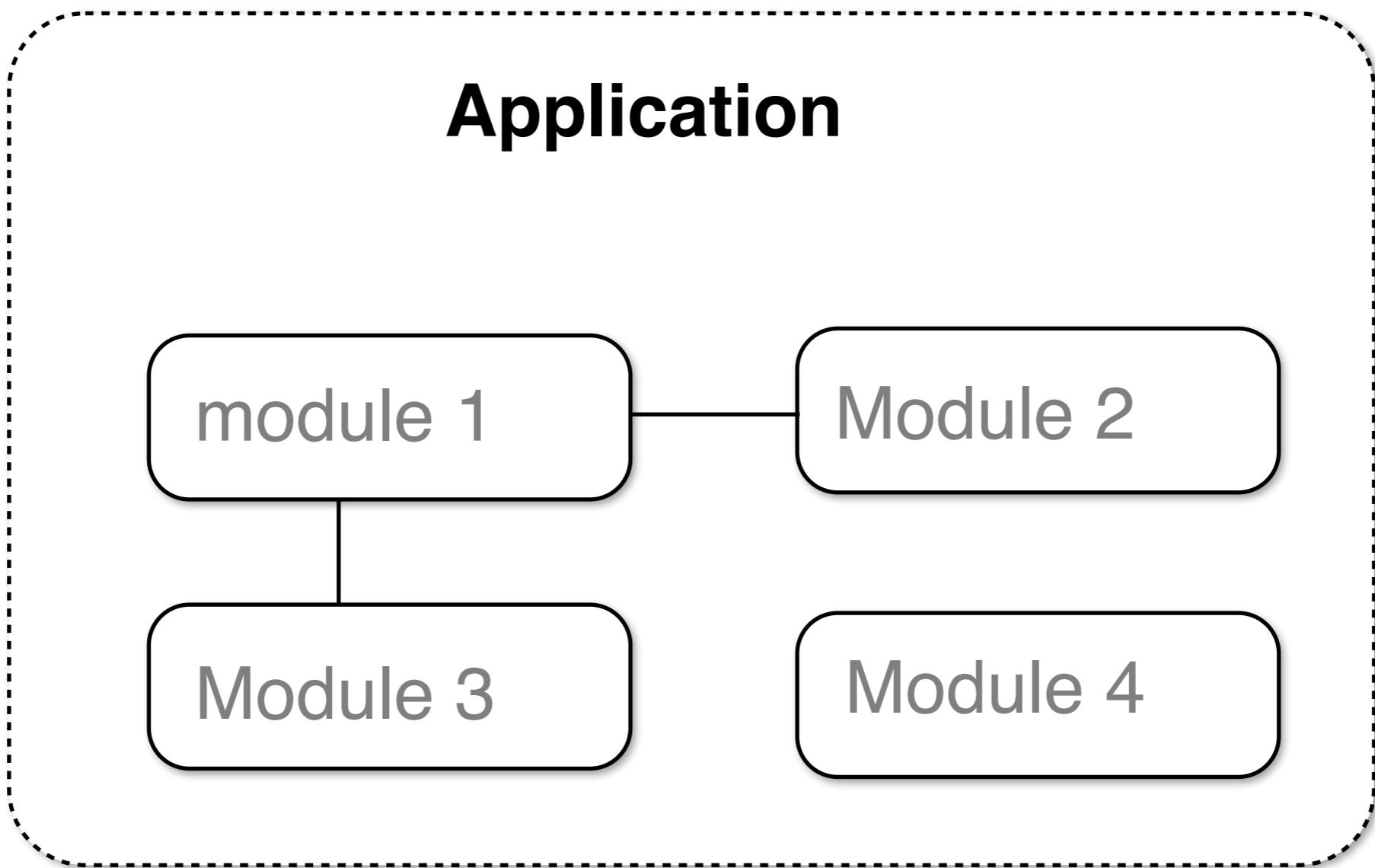
# 5. Learning Curve



# 6. Infra Cost



# 7. Debugging



## Dev & Ops Teams



### Log Data

Web Logs  
App Logs  
Database Logs  
Container Logs

### Metrics Data

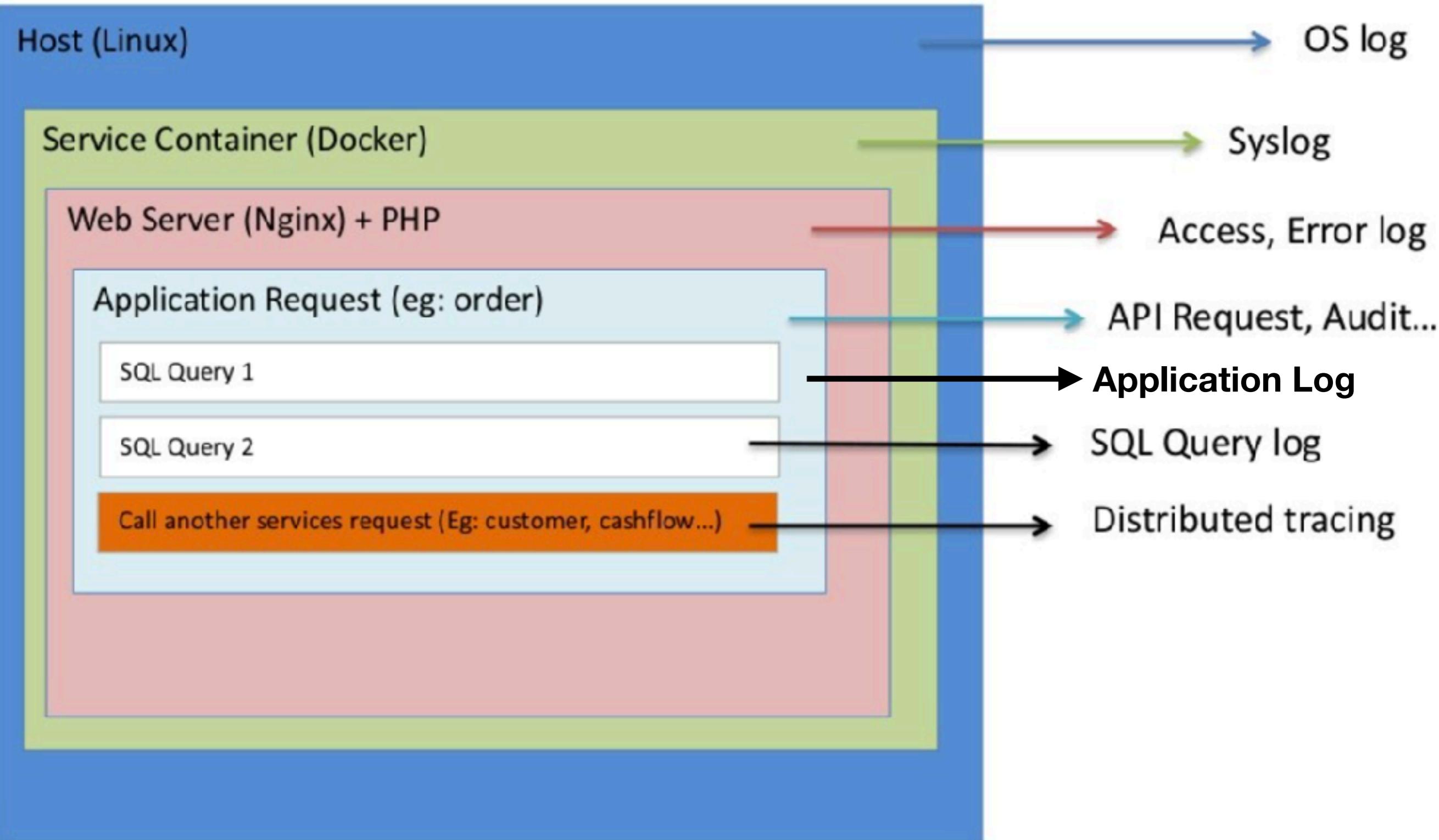
Container Metrics  
Host Metrics  
Database Metrics  
Network Metrics  
Storage Metrics

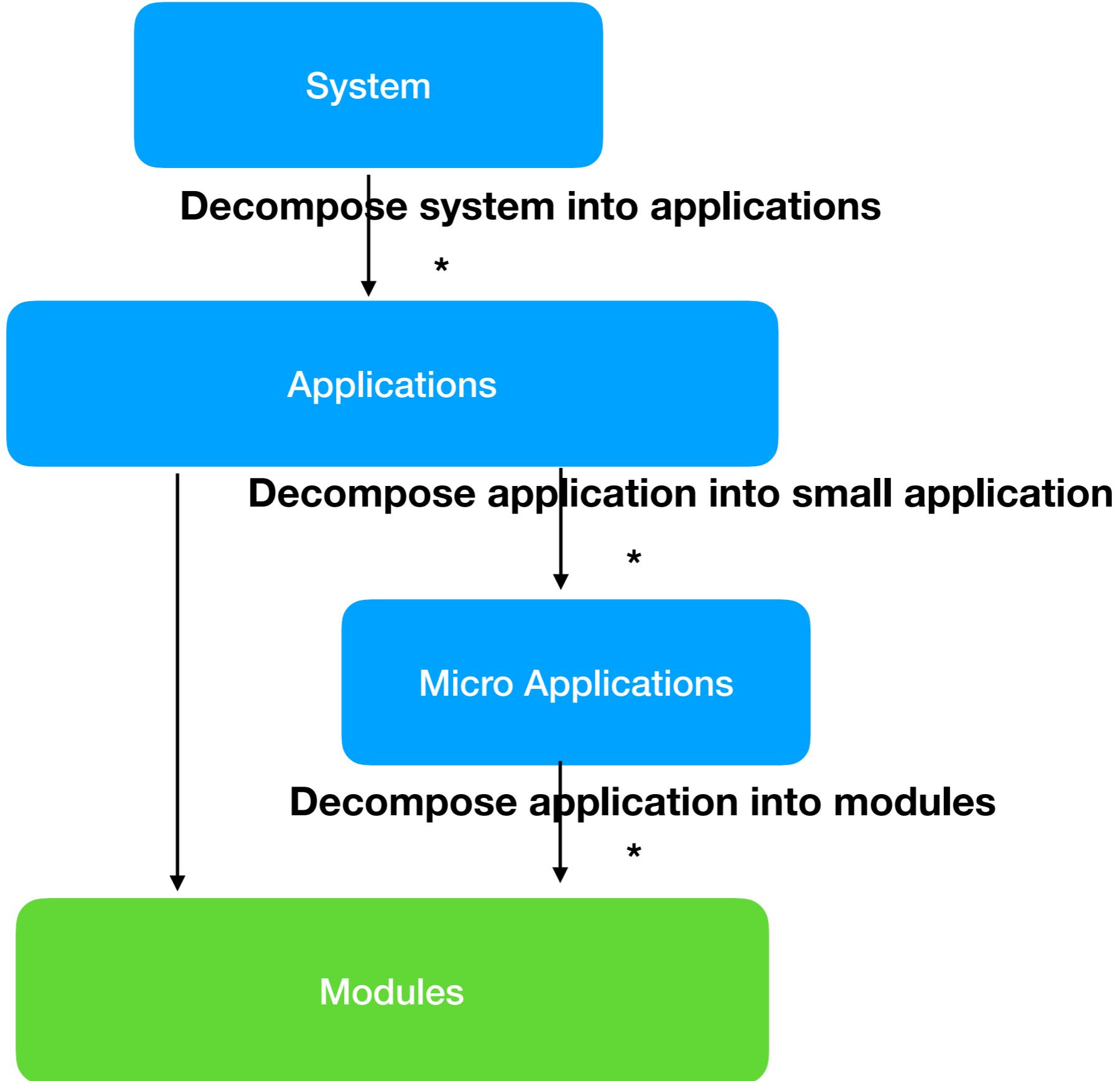
### APM Data

Real User Monitoring  
Txn Perf Monitoring  
Distributed Tracing

### Uptime Data

Uptime  
Response Time





# Service Oriented Architecture

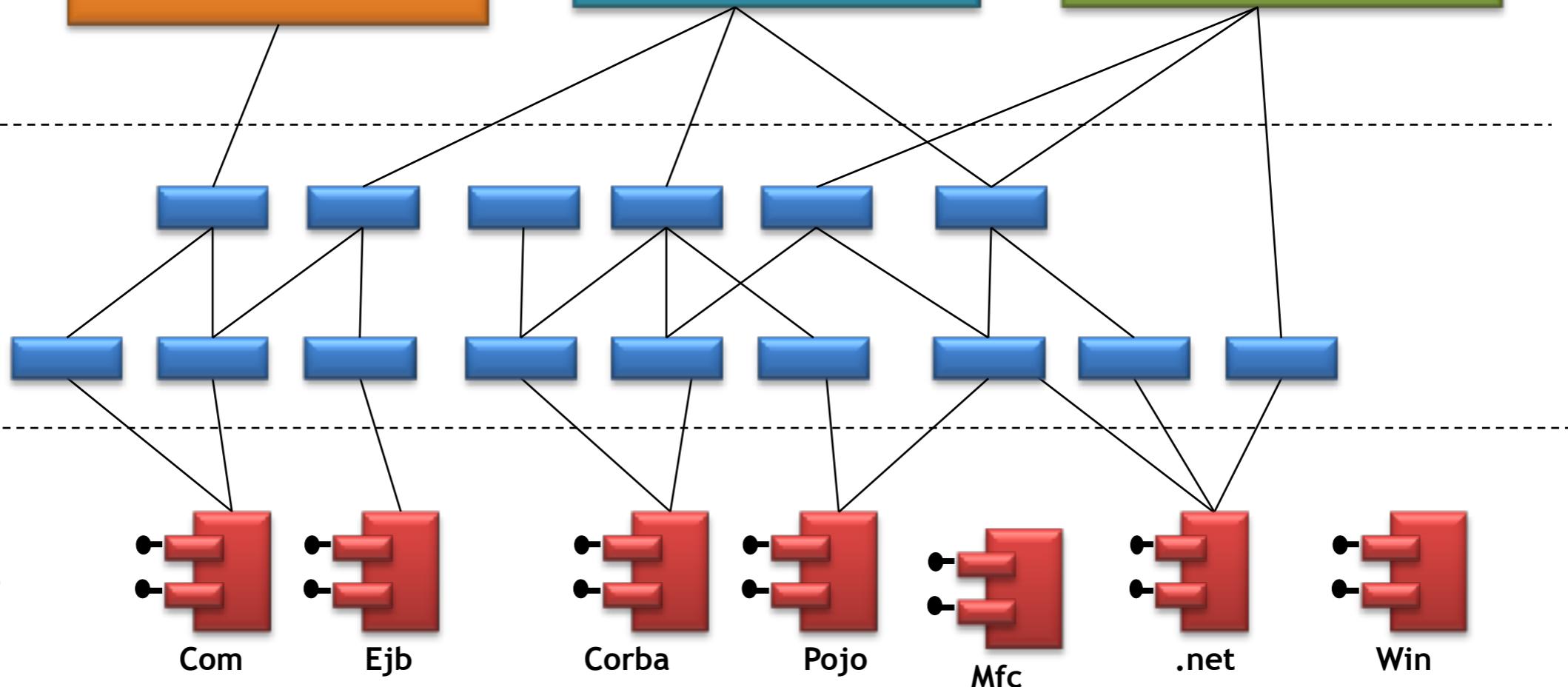
Composite Applications

Application Mgmt Application

Recruitment Application

Personnel Mgmt Application

Service Layer



Operational Systems



Human Resources



Enterprise Resource Planning Systems



Sales Systems



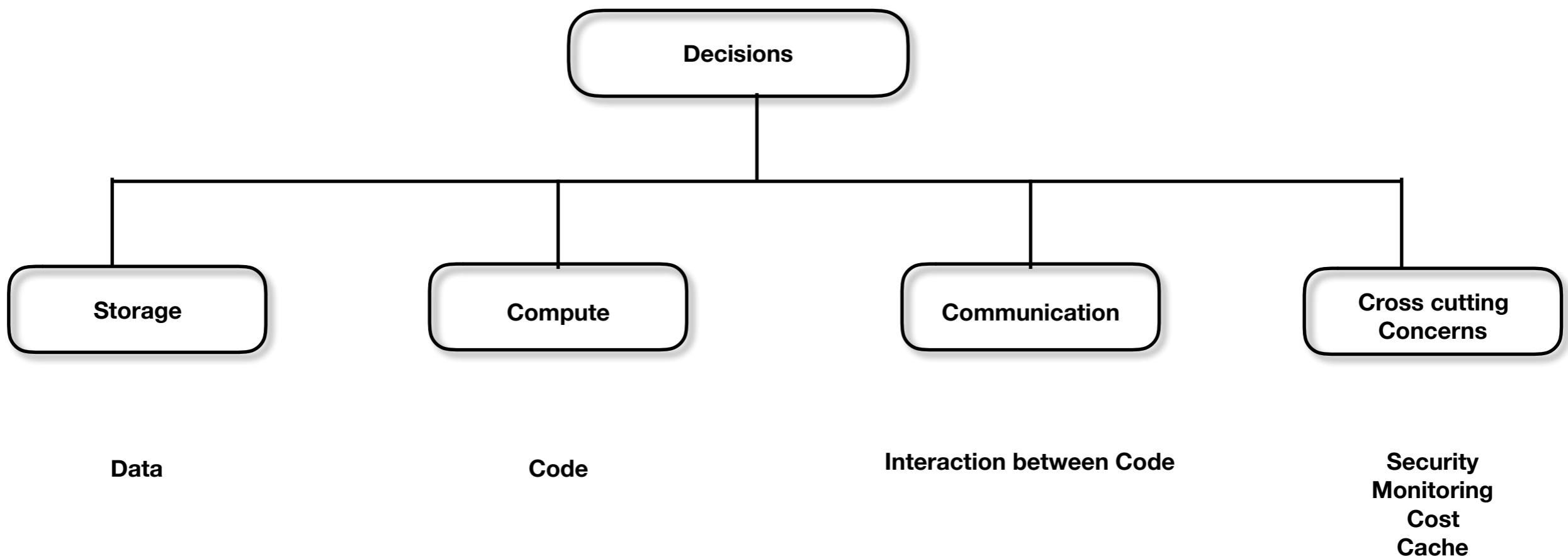
Custom Line of Business Applications



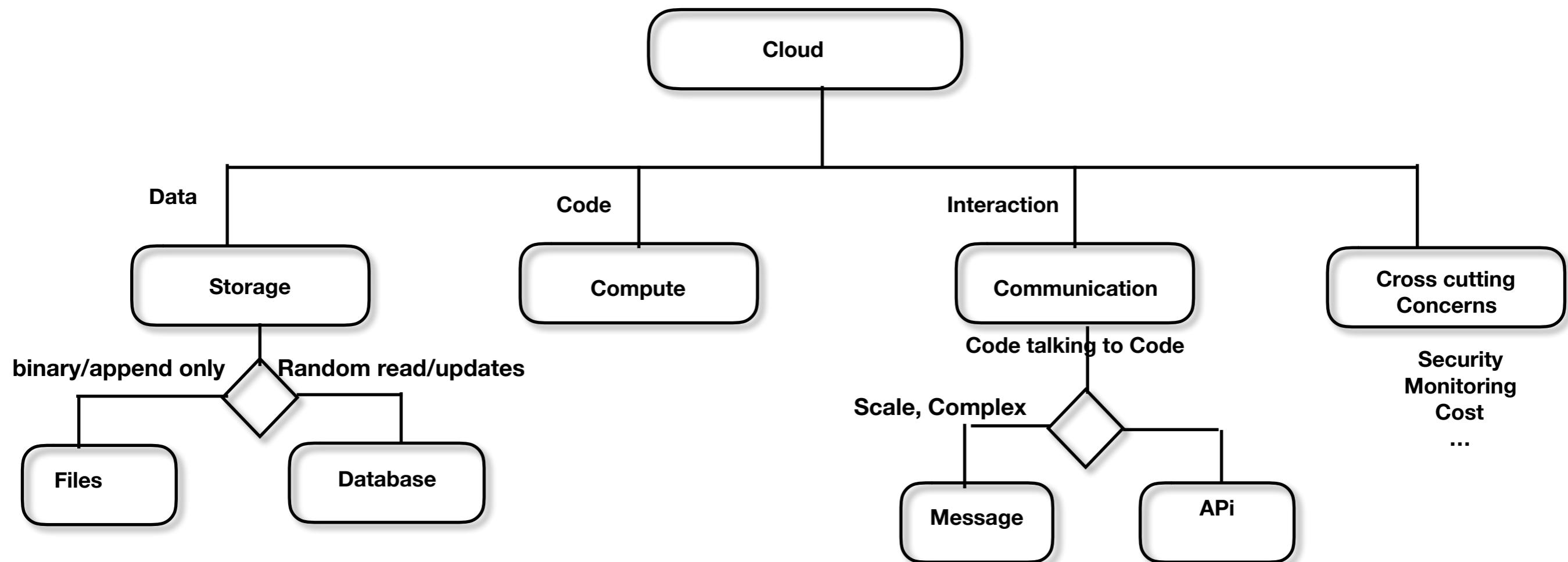
Customer Relationship Management

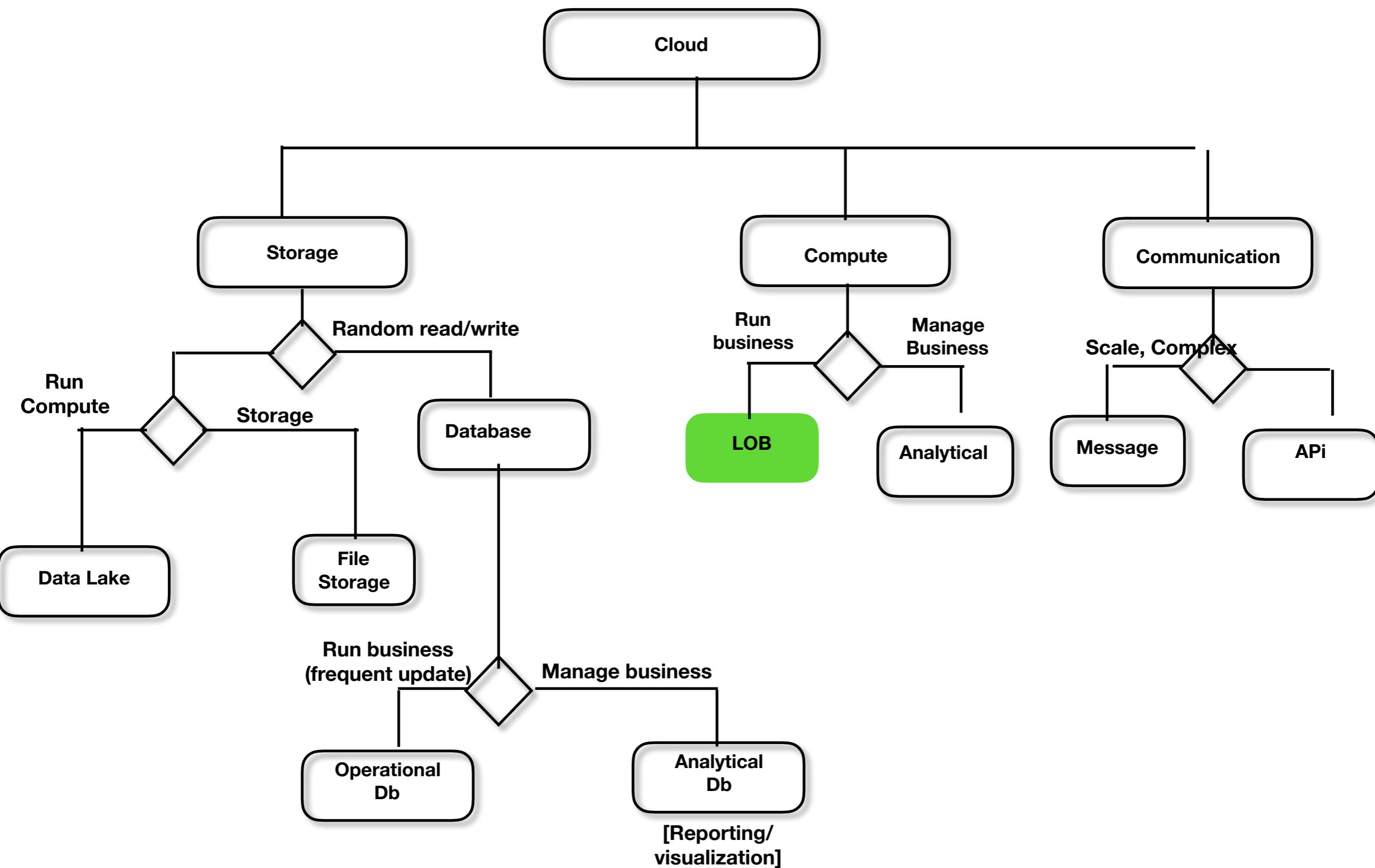


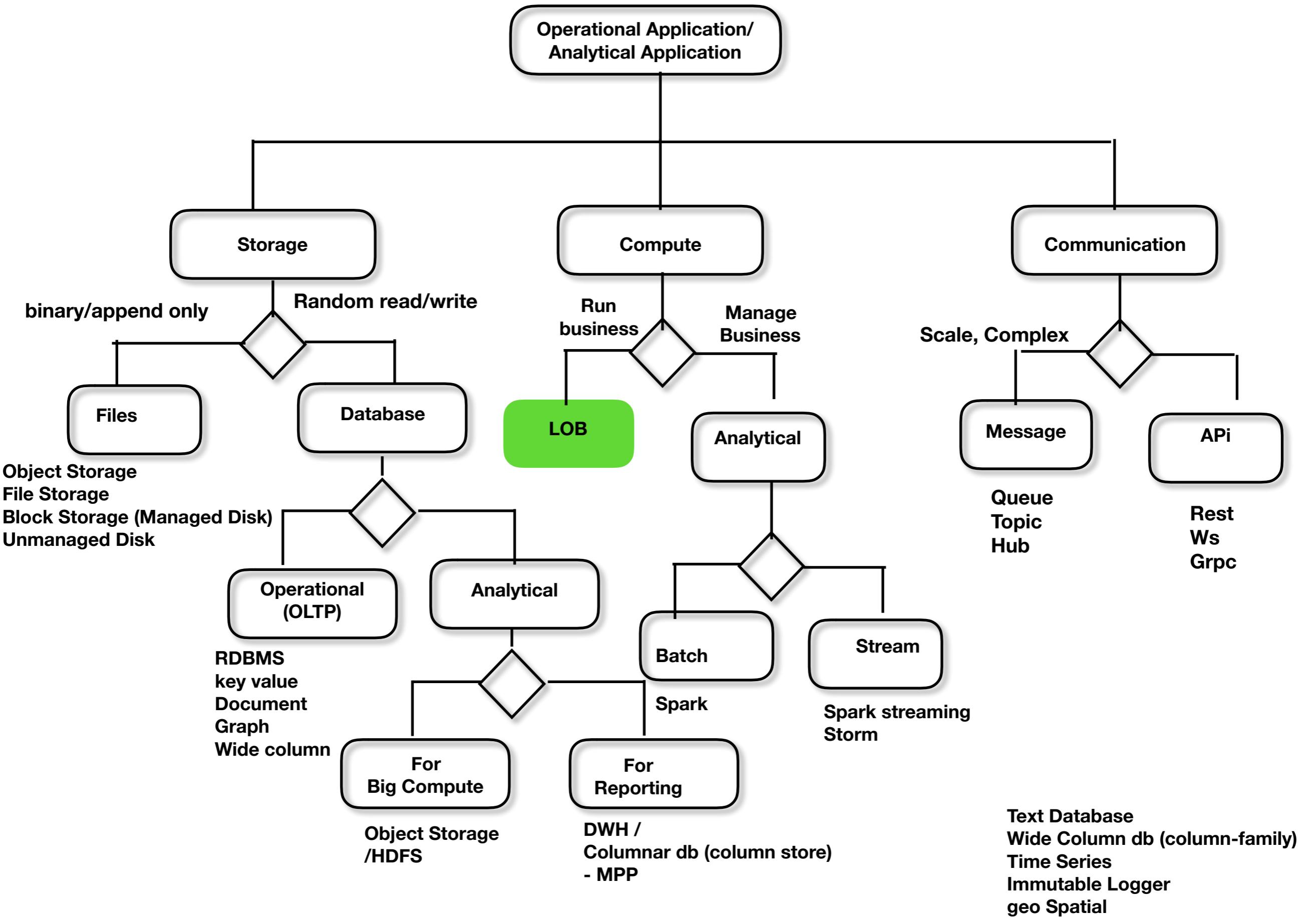
# Make decisions ?



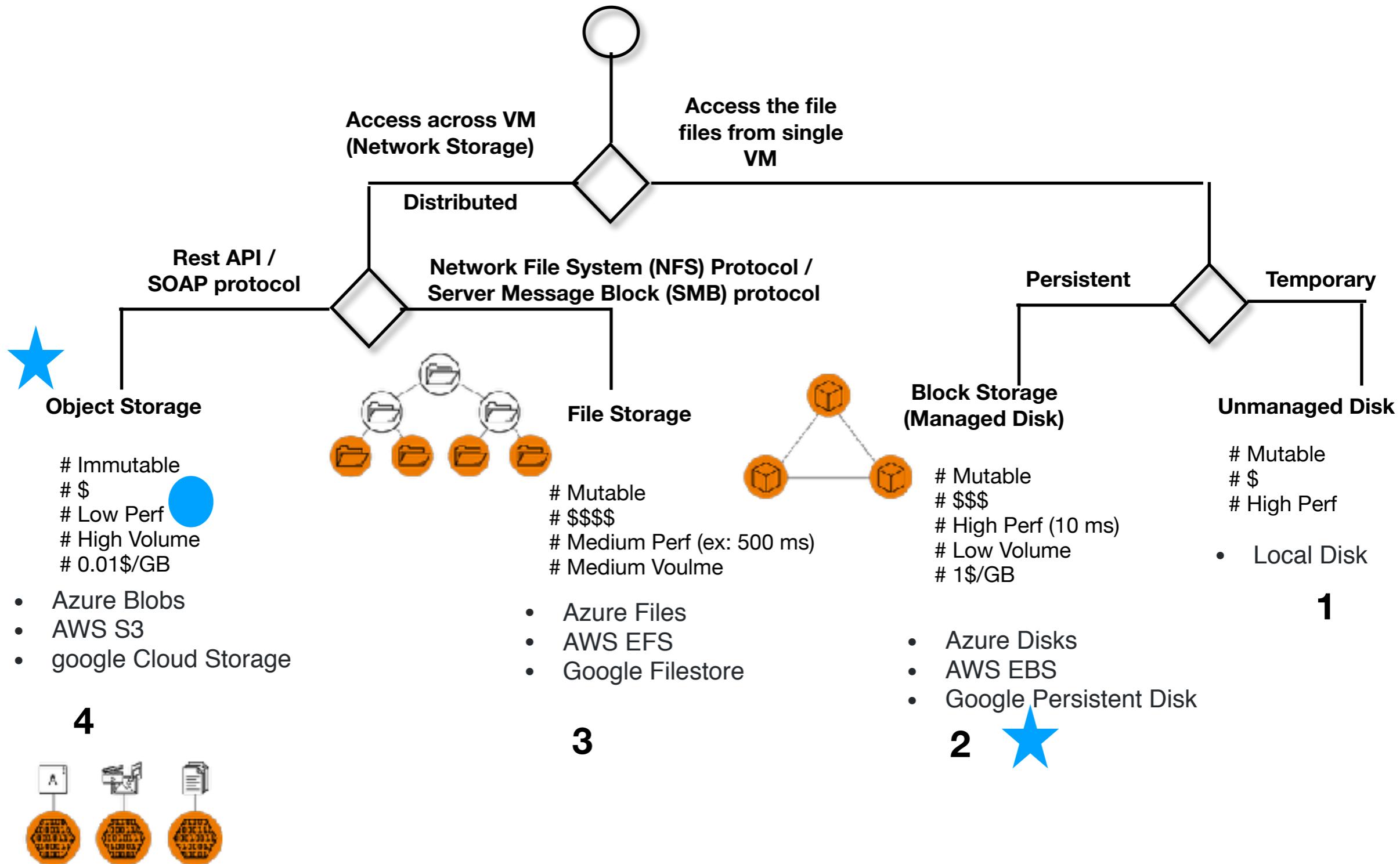
# Choosing Technology ?



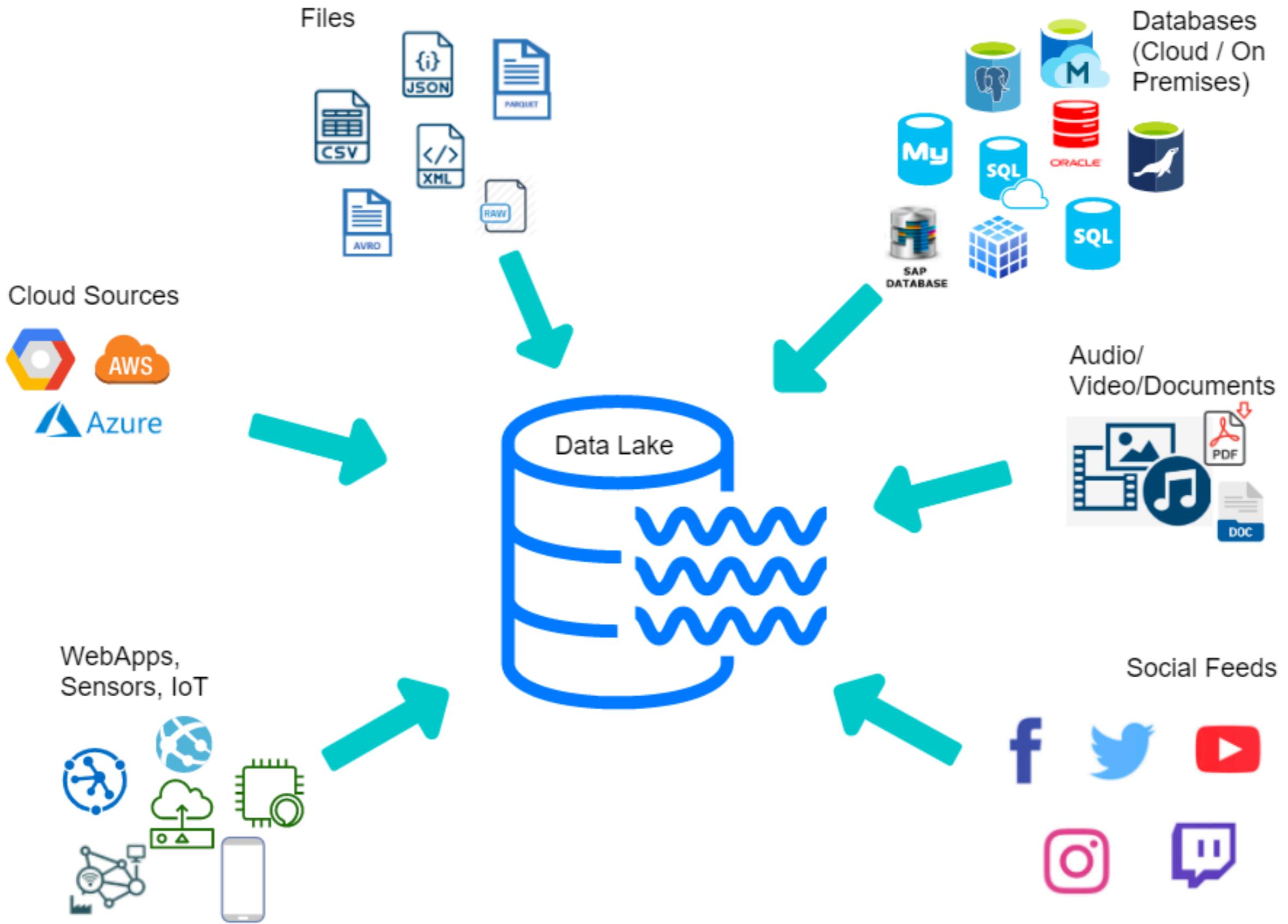


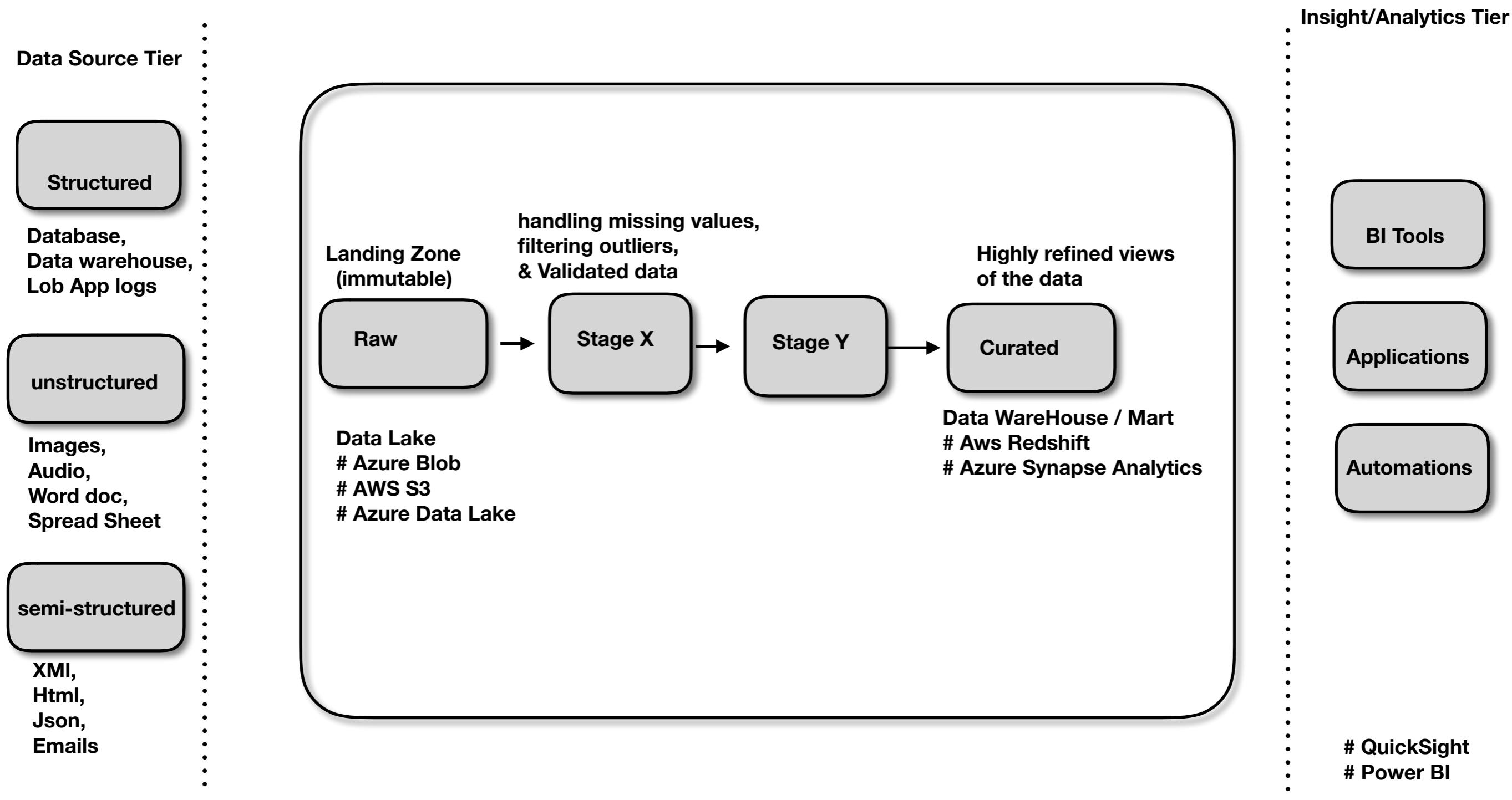


# File Storage



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

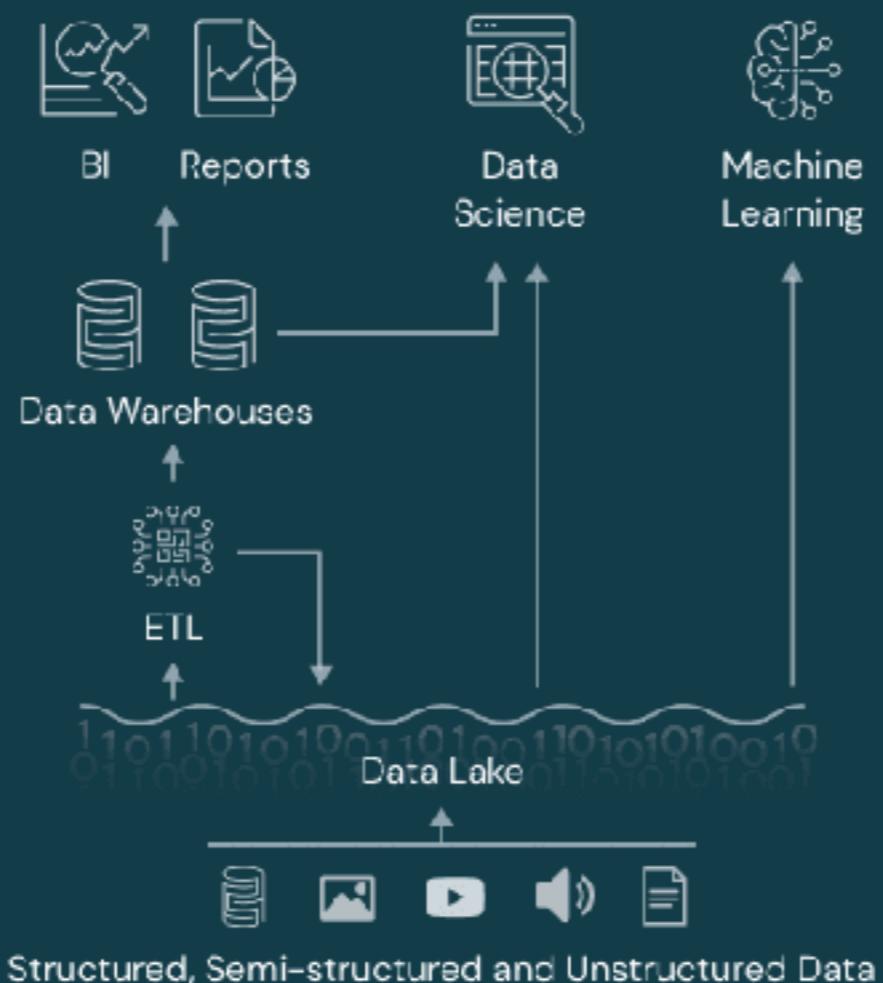




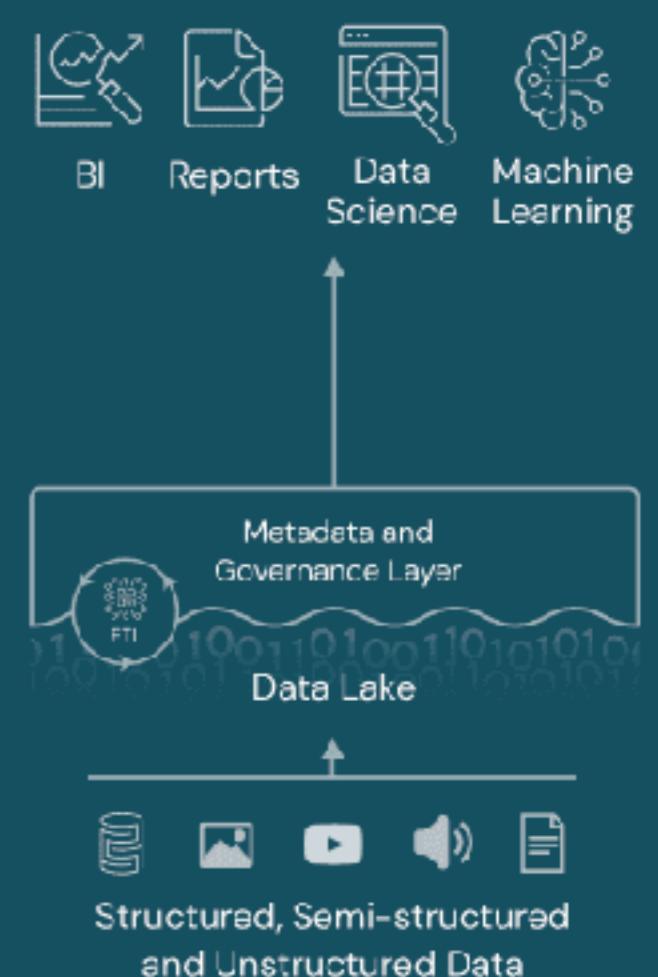
## Data Warehouse

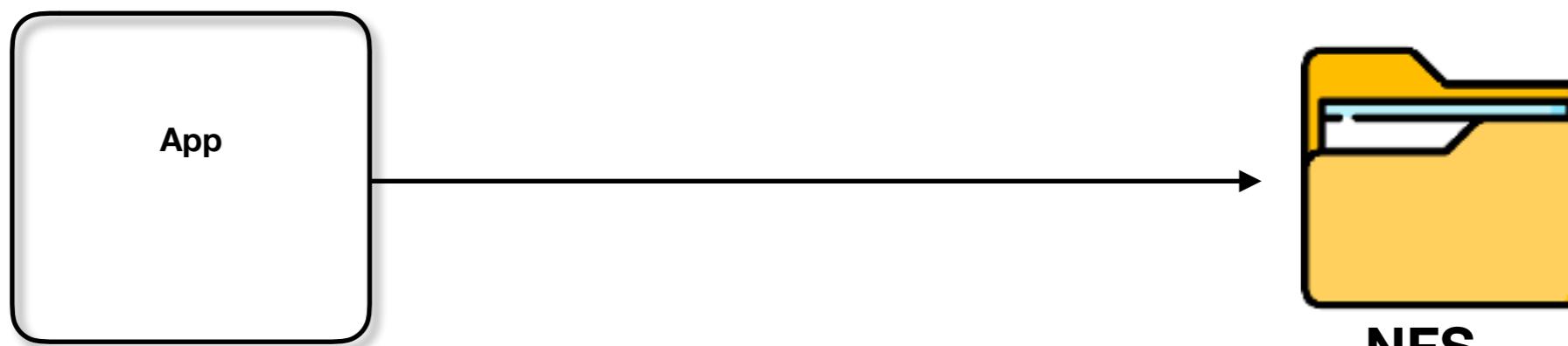


## Data Lake



## Data Lakehouse

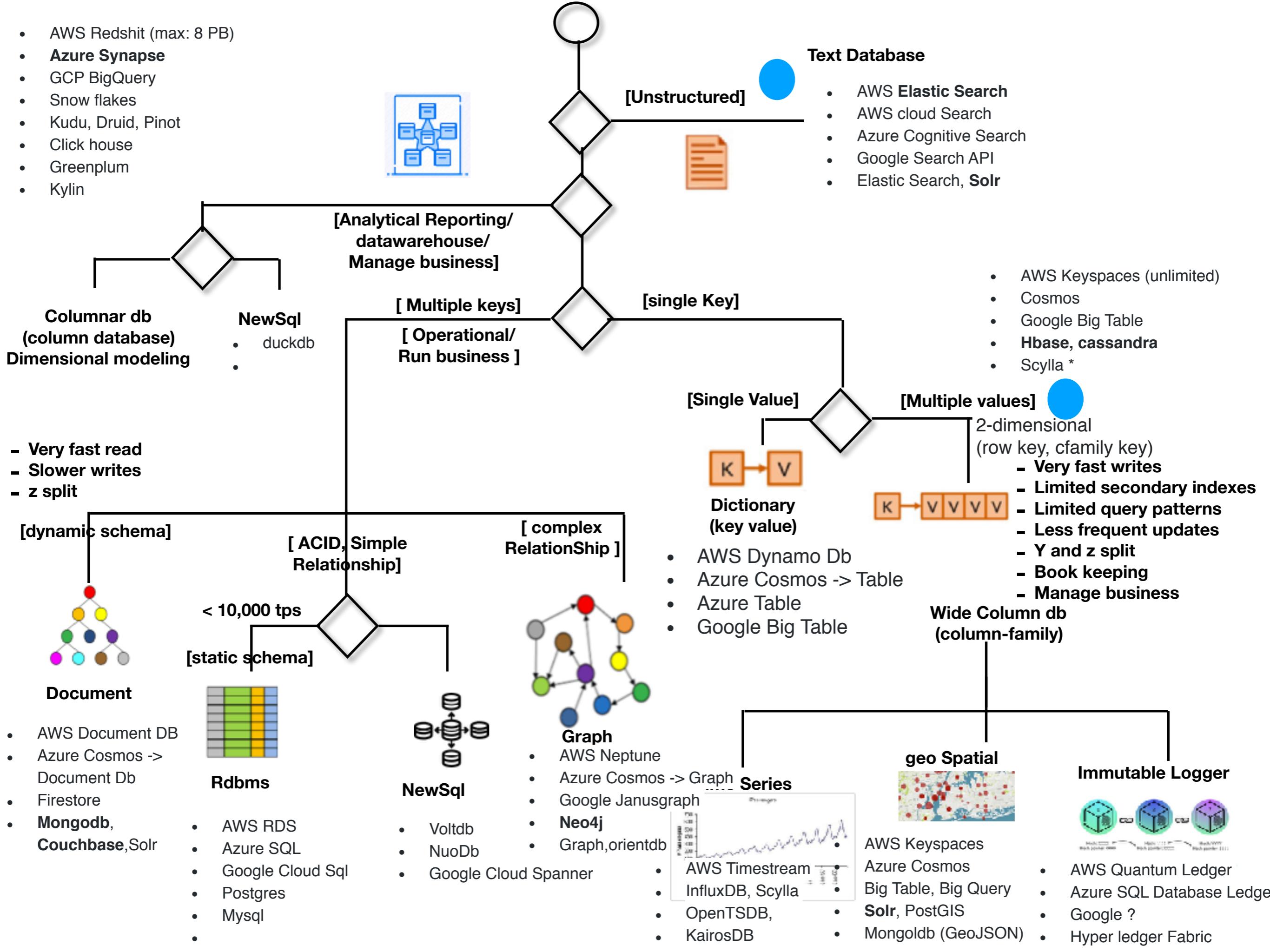




**system.io**

**f:\\\\dir\\\\file**

- AWS Redshift (max: 8 PB)
- Azure Synapse
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house
- Greenplum
- Kylin



	Document	Wide Column	Rdbms
Fast writes	--	++	--
Fast reads	++	--	++
Secondary indexes	++	--	++
ACID	-	--	++
Y (Split)	--	++	--
Z (Shard)	++	++	?
Dynamic Schema	++	++	--
Rich Query Support	+	--	++
Rich Object Model	++	--	+
CAP	CP	AP	CA

**Consider a NoSQL datastore when:**

You have high volume workloads that require predictable latency at large scale (e.g. latency measured in milliseconds while performing millions of transactions per second)

Your data is dynamic and frequently changes

Relationships can be de-normalized data models

Data retrieval is simple and expressed without table joins

Data is typically replicated across geographies and requires finer control over consistency, availability, and performance

Your application will be deployed to commodity hardware, such as with public clouds

**Consider a relational database when:**

Your workload volume generally fits within thousands of transactions per second

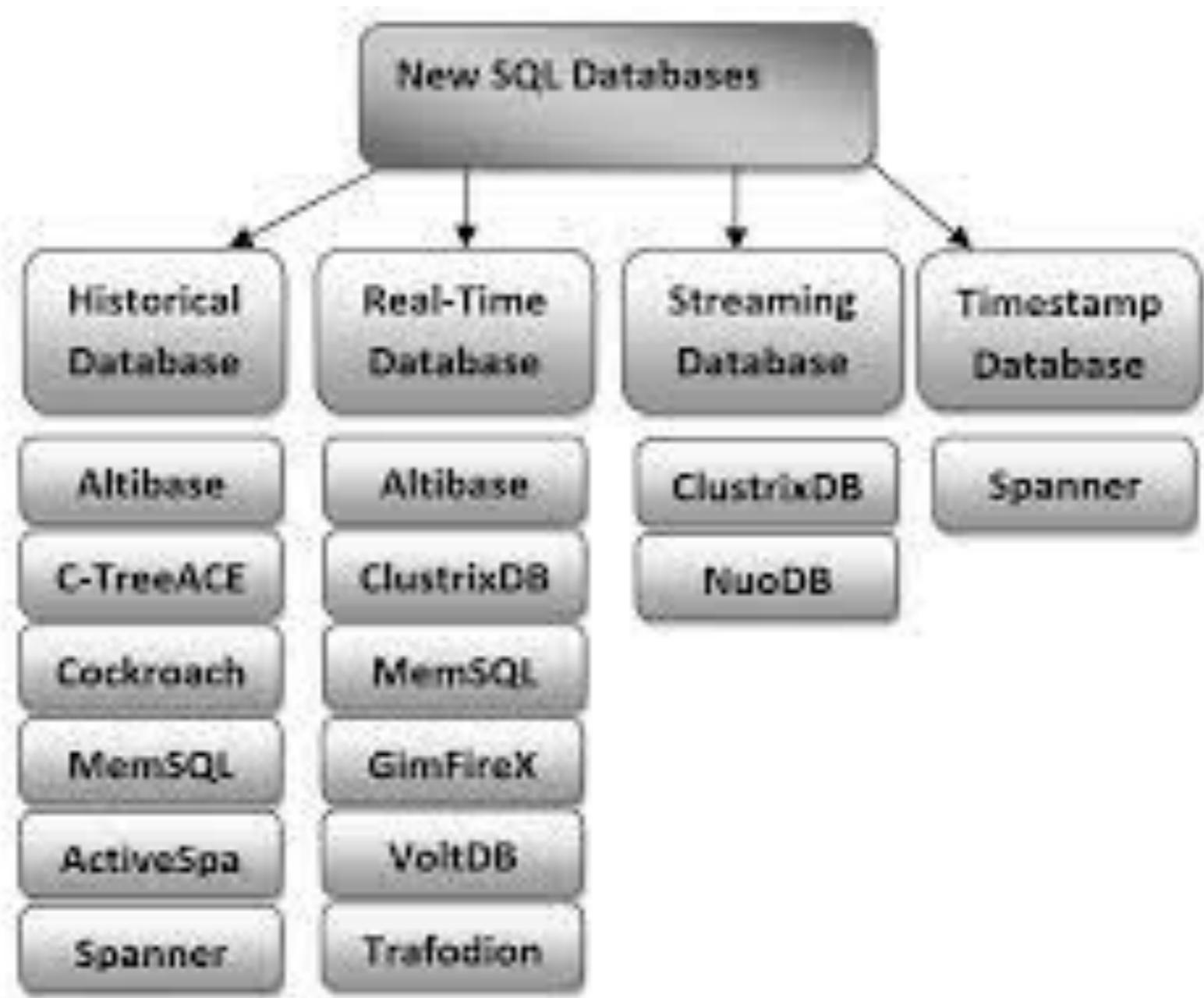
Your data is highly structured and requires referential integrity

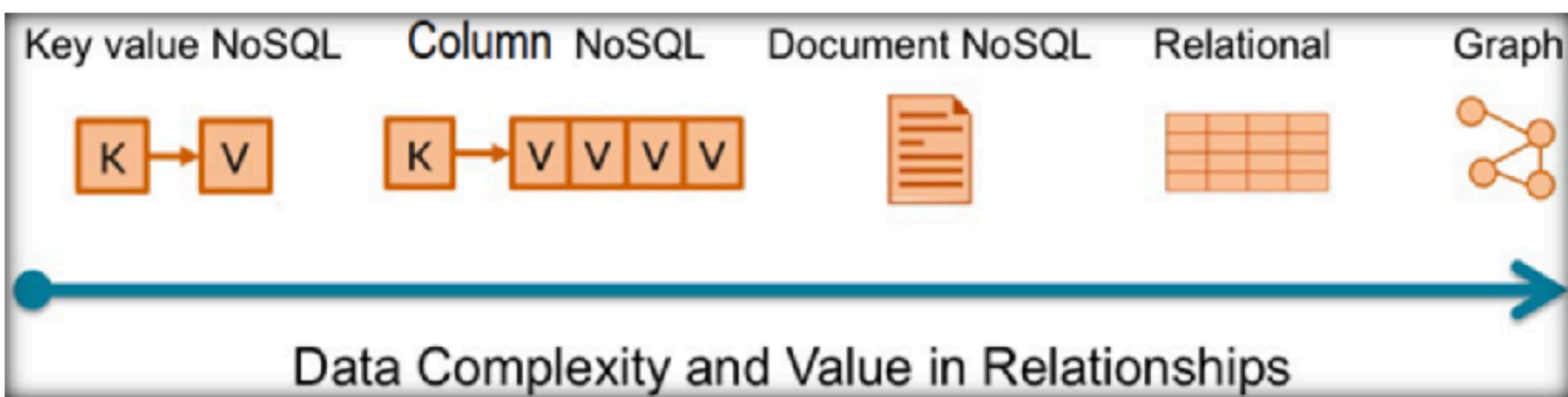
Relationships are expressed through table joins on normalized data models

You work with complex queries and reports

Data is typically centralized, or can be replicated regions asynchronously

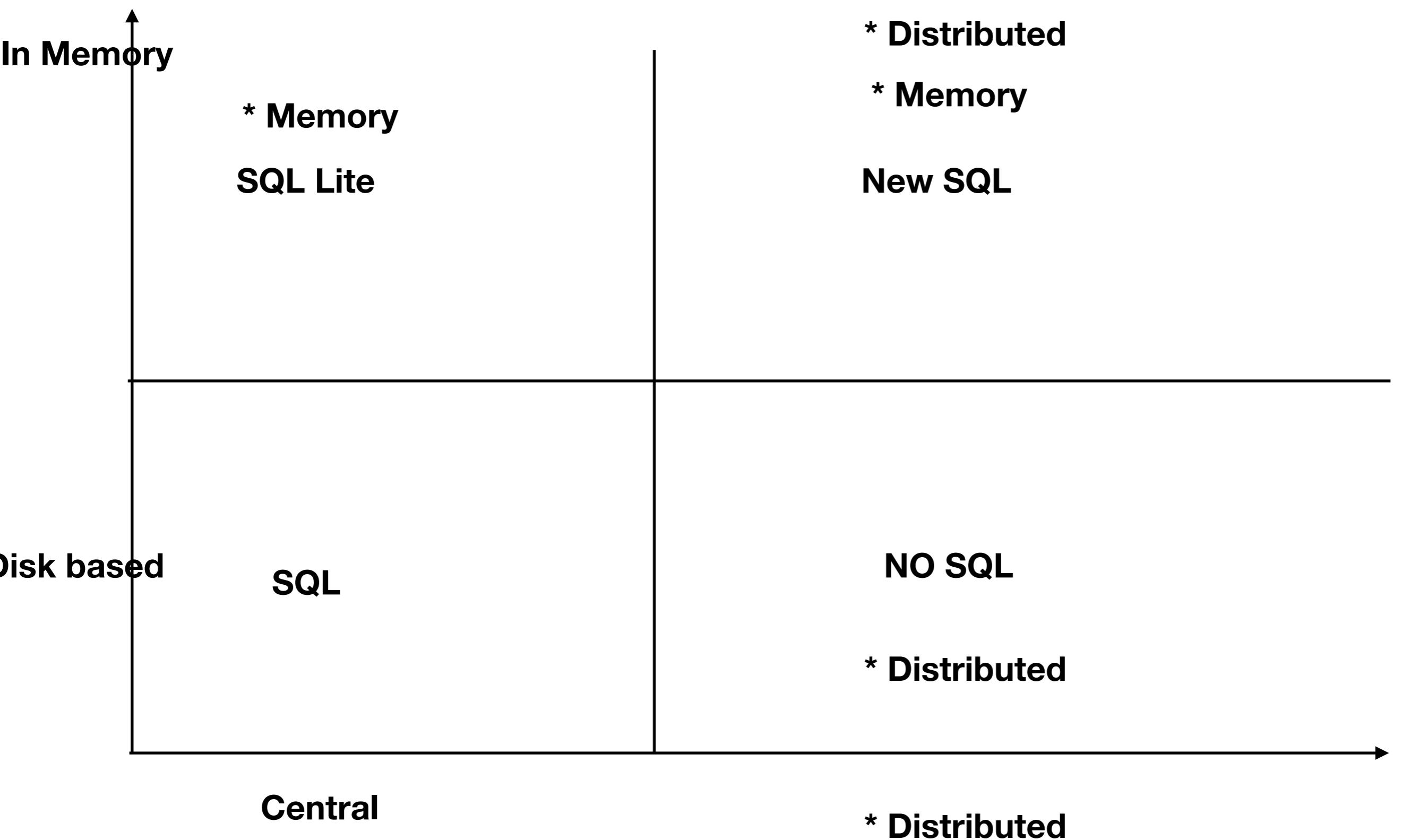
Your application will be deployed to large, high-end hardware

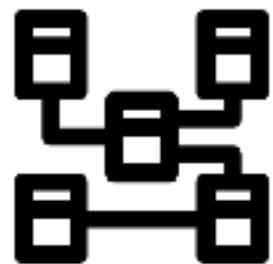




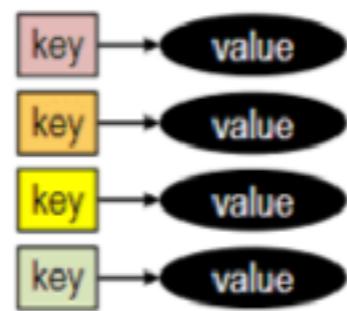
In PostgreSQL, the approach to scaling depends on whether you are talking about writing or reading data. For writes, it is based on a scale-up architecture, in which a single primary machine running PostgreSQL must be made as powerful as possible in order to scale. For reads, it is possible to scale-out PostgreSQL by creating replicas, but each replica must contain a full copy of the database.

Partitioning in SQL is for spreading data across storage devices. It's still one active server.





**Relational**



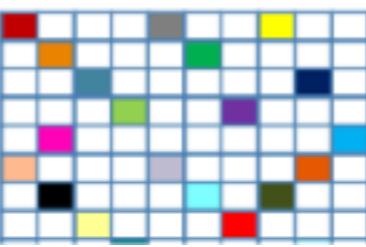
**Shard**

**Key Value**



**Shard**

**Document**



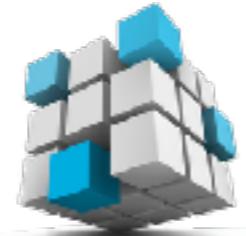
**Wide Column**

**Split,  
Shard**



**Shard**

**Graph**



**Columnar**



**Text**



**Time Series**



**DFS**



**Geo Spatial**



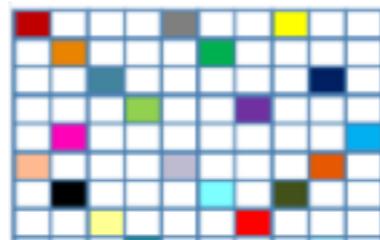
**Immutable Logger**

**New SQL**

**Lake House**

**Skan.**

© SKAN.AI 2019 CONFIDE



**Wide Column**



**Time Series**



**Geo Spatial**

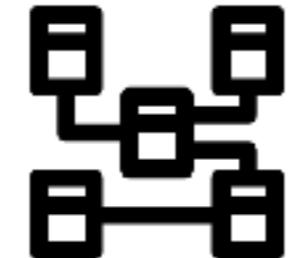


**Immutable Logger**

**Skan.**

© SKAN.AI 2019 CONFIDE

**Central  
Strict Schema (dense)  
Table + join  
Transaction**



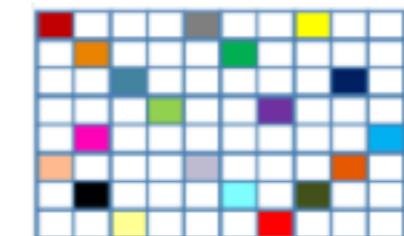
**Relational**

**Distributed  
Flexible Schema  
Hierarchy**

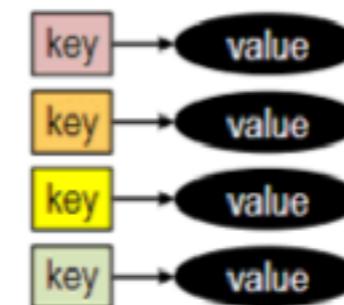


**Document**

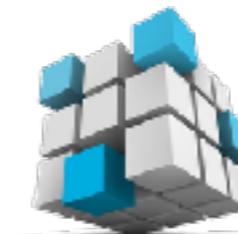
**Distributed  
Flexible Schema  
Table**



**Wide Column**



**Key Value**



**Columnar**

**Star schema  
Cube**

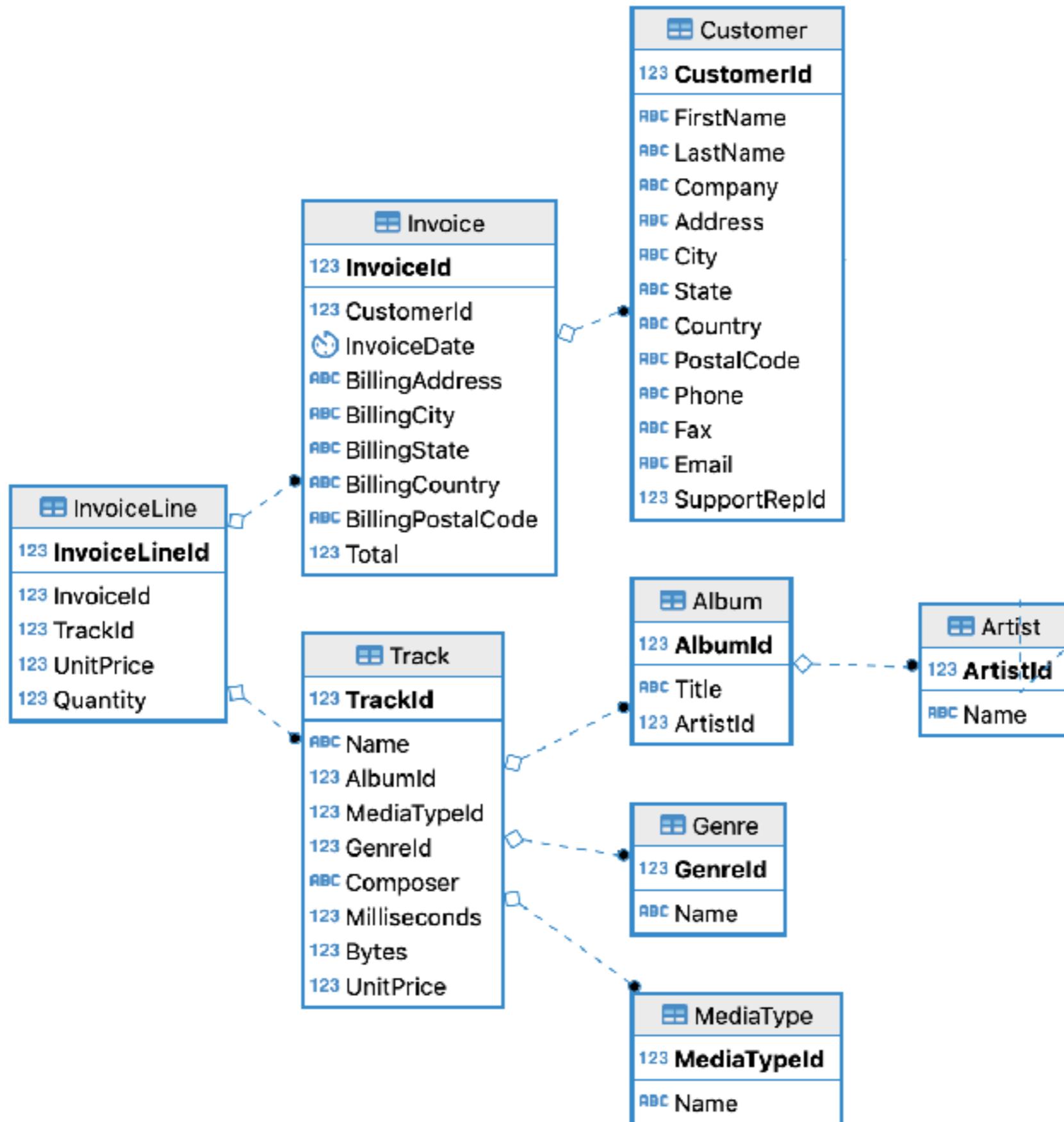


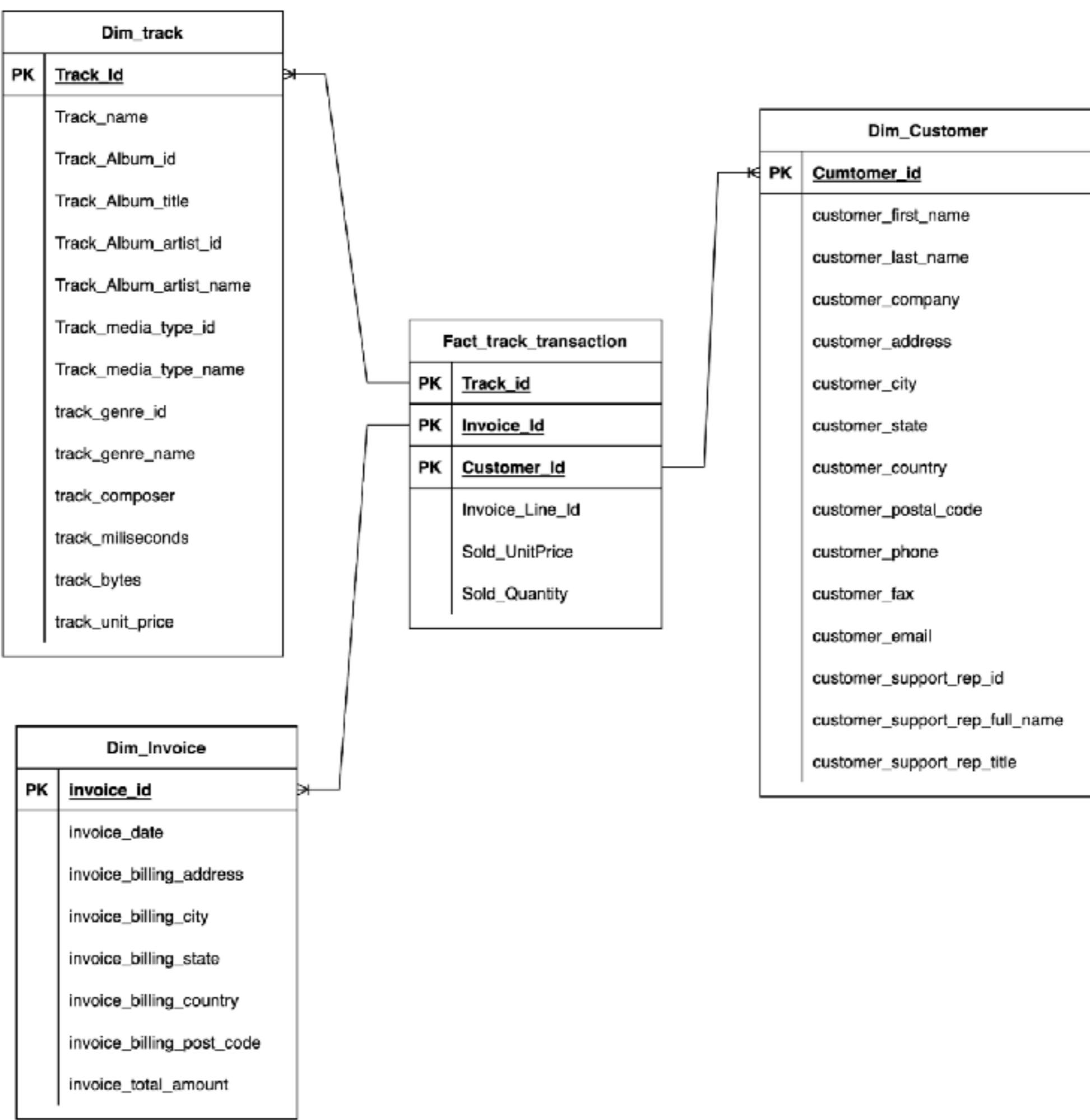
**Graph**

**Skan.**

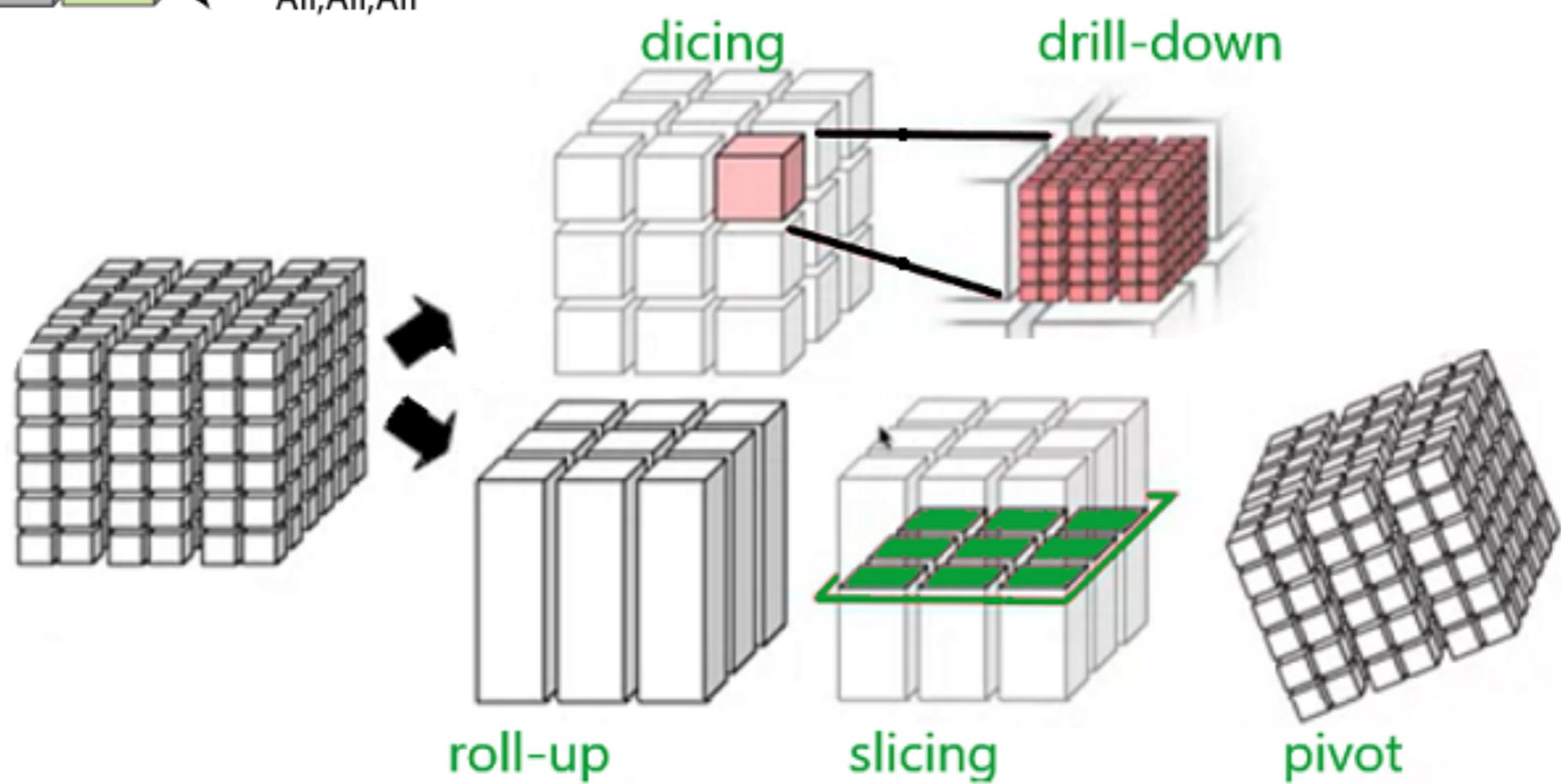
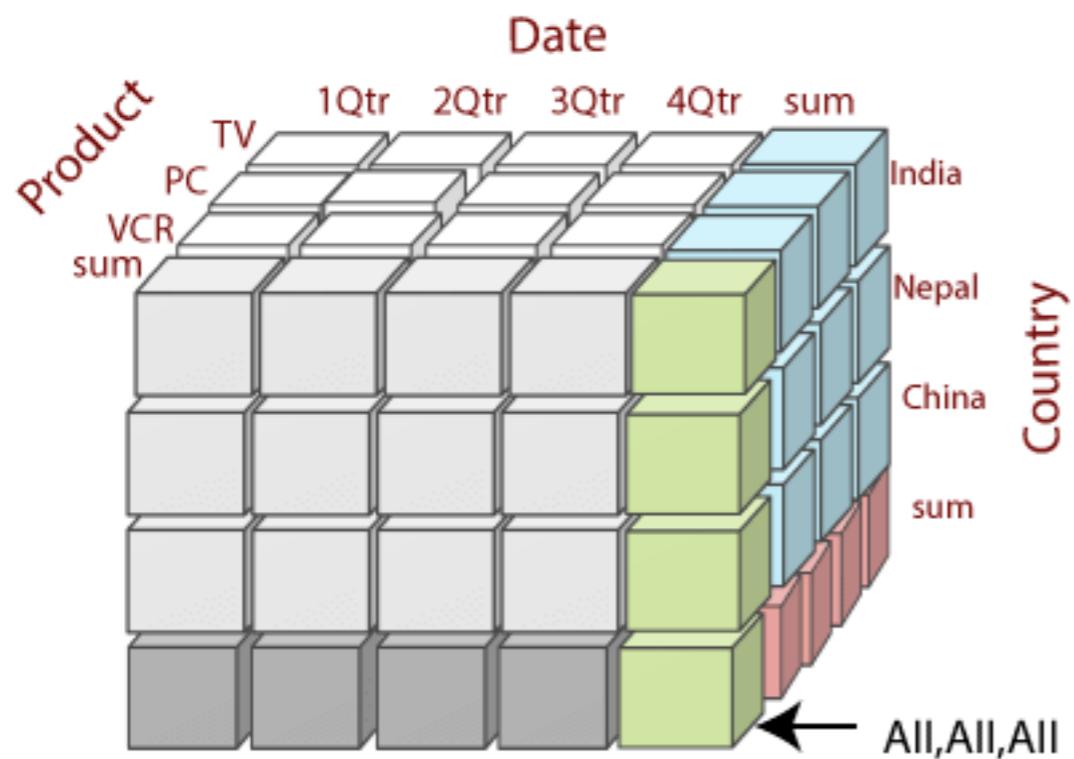
© SKAN.AI 2019 CONFIDE

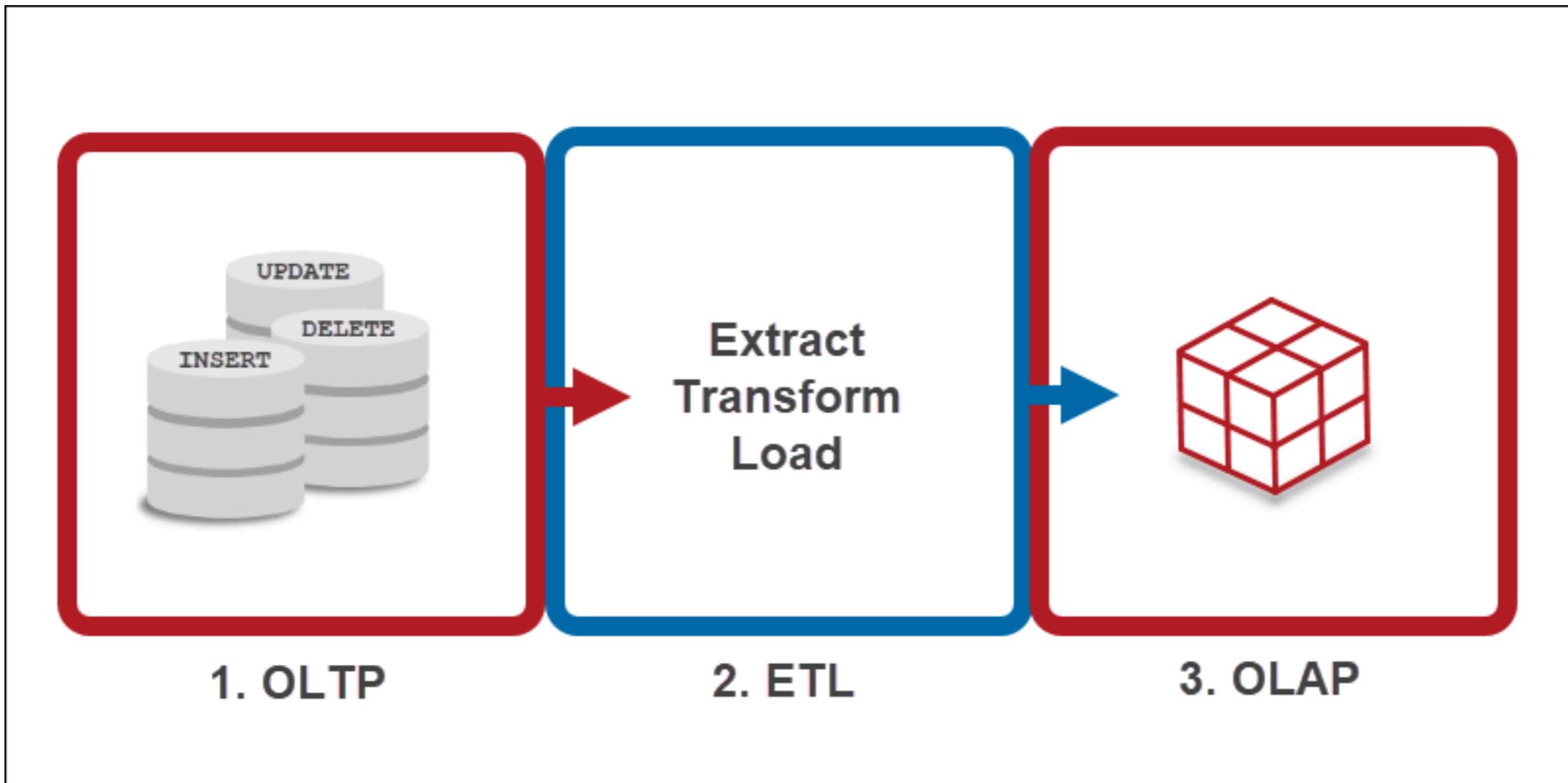
- Key-based read patterns.
- Data that will be written once and never deleted or updated.
- Log-based data sets that may grow over time.
- Read patterns of many, small queries.
- Time series.
-





## Data Cube





A fact table has two types of columns:  
those that include fact (purely numerical facts) and those that are foreign keys to the dimension table.

**Dimension Table**

time
time_key
day
day_of_the_week
month
Quarter
Year

**Dimension Table**

item
item_key
item_name
brand
type
supplier_type

**Sales Fact Table**

time_key
item_key
branch_key
location_key
unit_sold
dollars_sold

**Dimension Table**

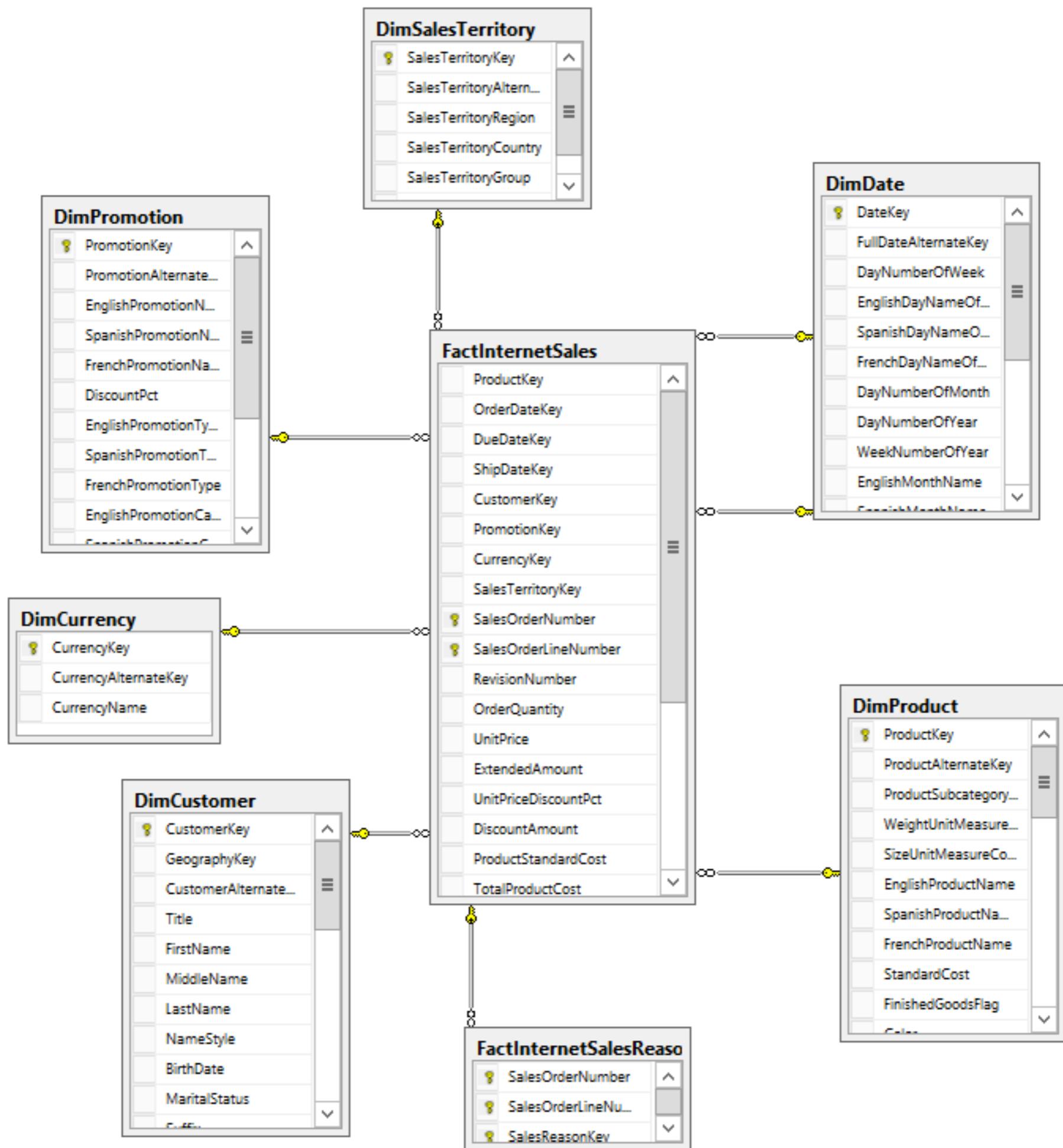
branch
branch_key
branch_name
branch_type

**Measures**

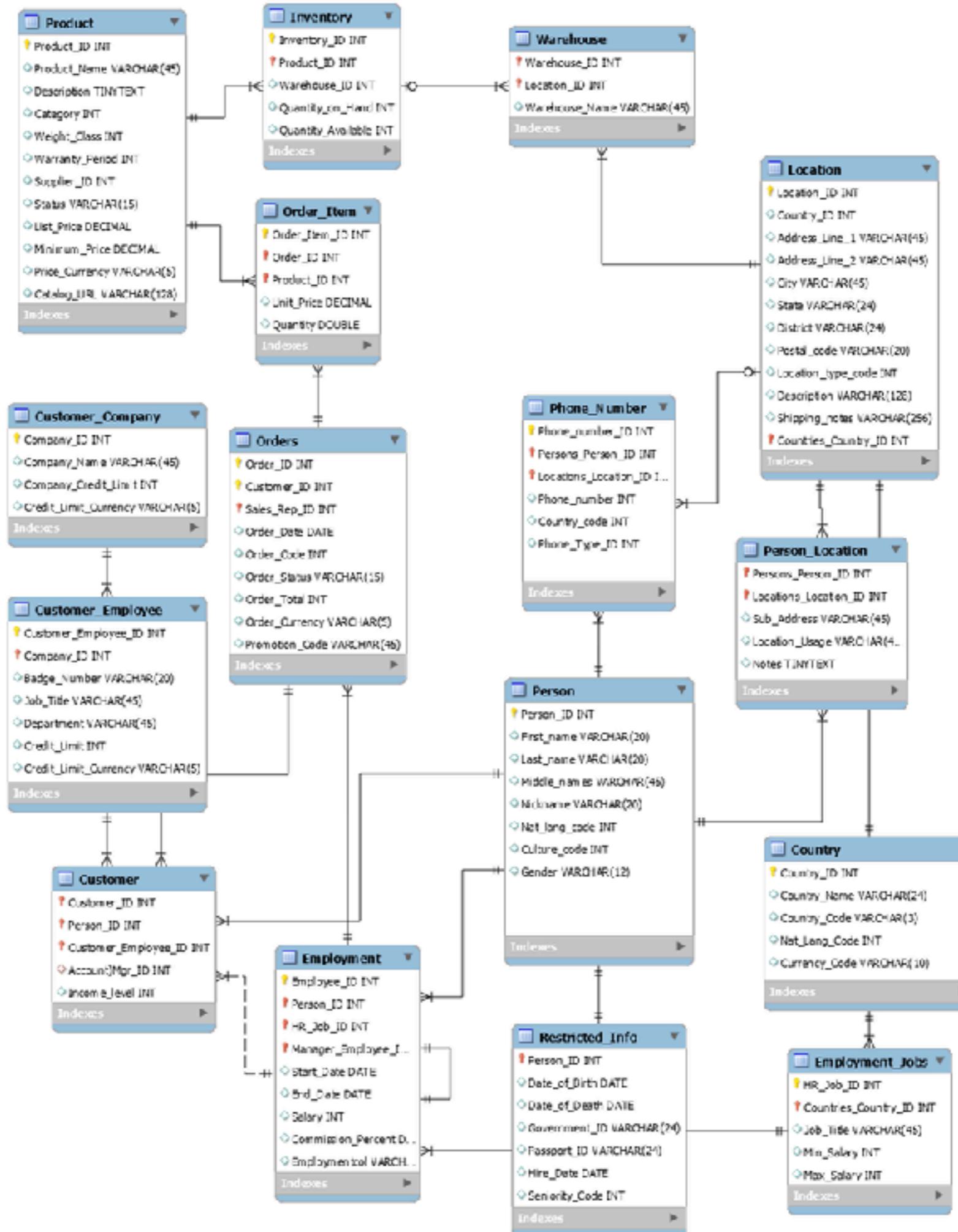
**Dimension Table**

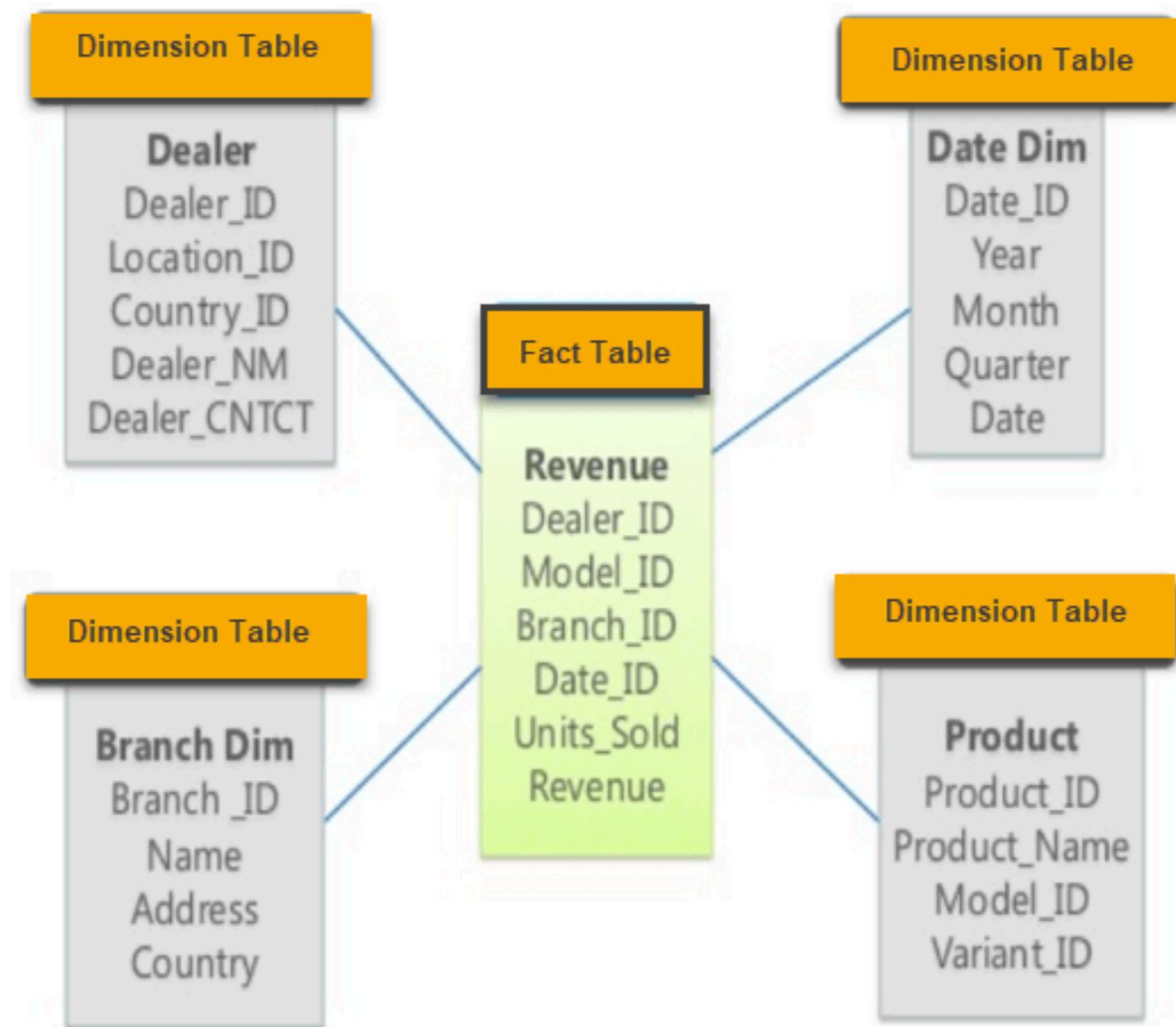
location
location_key
street
city
state_or_province
country

**Dimension tables** store the descriptive attributes of entities across the business enterprise and function as common lookup tables



# Relational

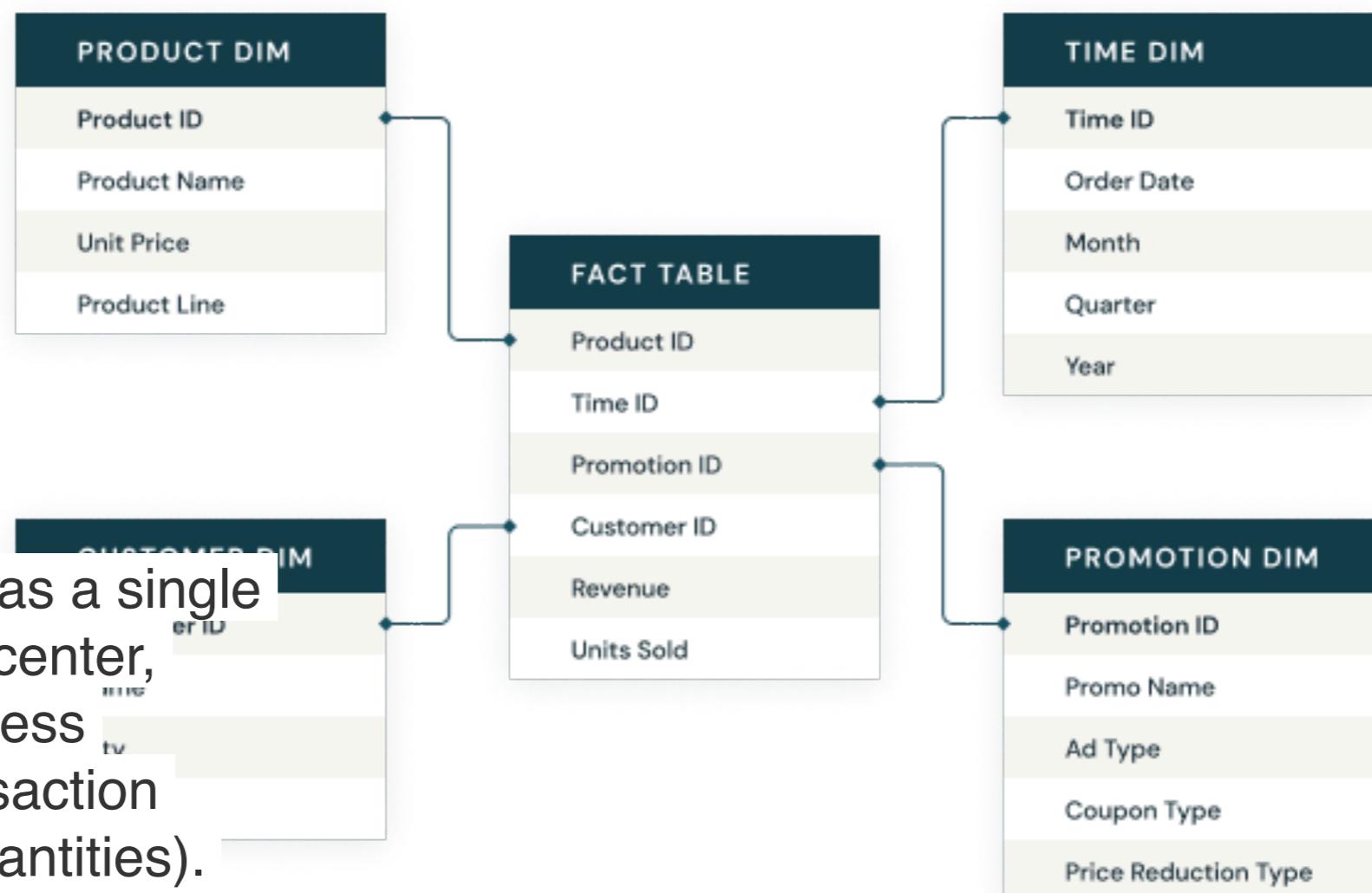




Skan.

© SKAN.AI 2019 CONFIDE

# Star schema



A star schema has a single fact table in the center, containing business "facts" (like transaction amounts and quantities).

The fact table connects to multiple other dimension tables along "dimensions" like time, or product. Star schemas enable users to slice and dice the data

Skan.

© SKAN.AI 2019 CONFIDE

there are two basic models which are used in dimensional modeling

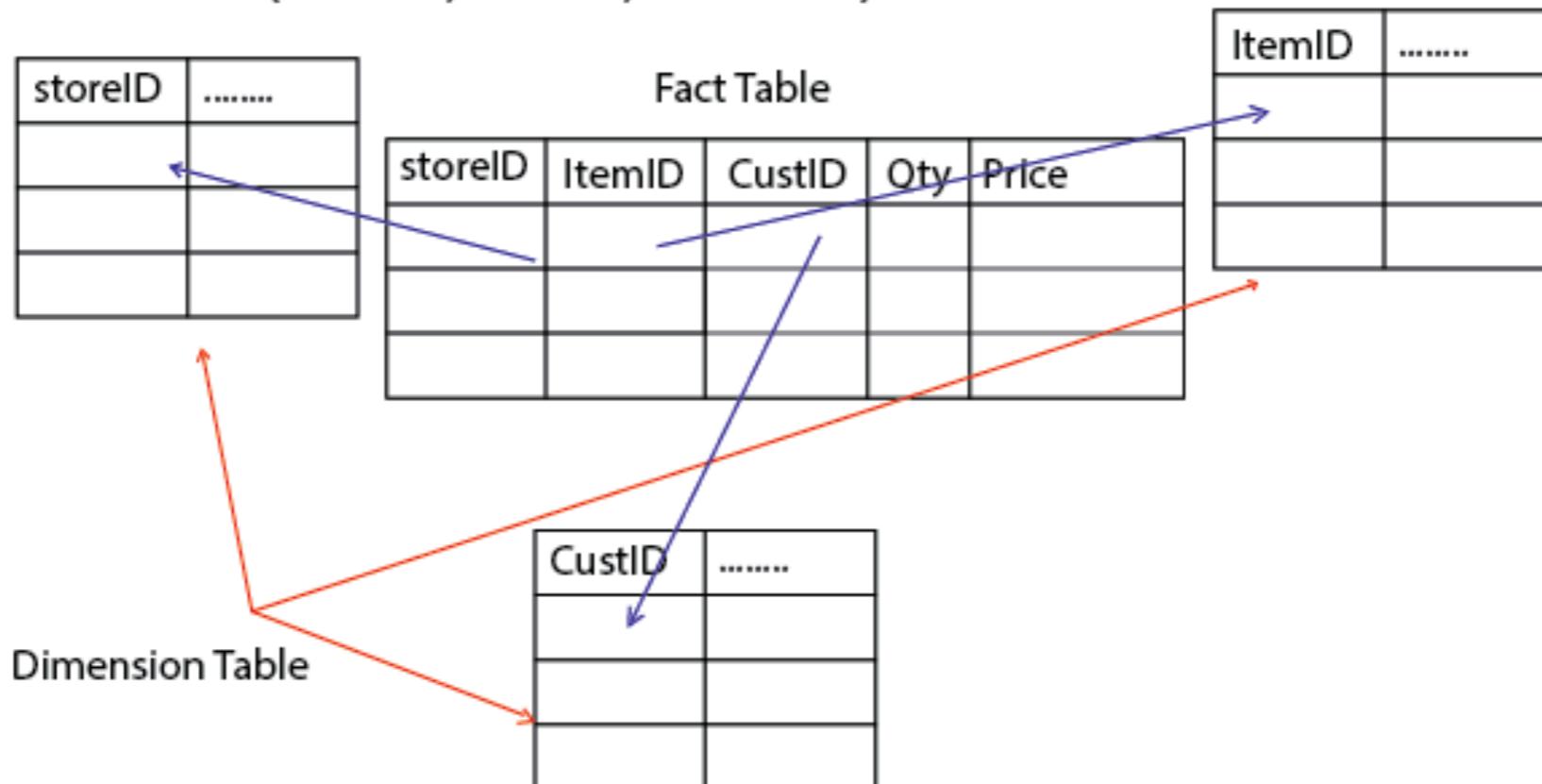
- Star Model
- Snowflake Model

Sales (StoreID, ItemID, CustID, qty, price)

StoreID (storeid, city, state)

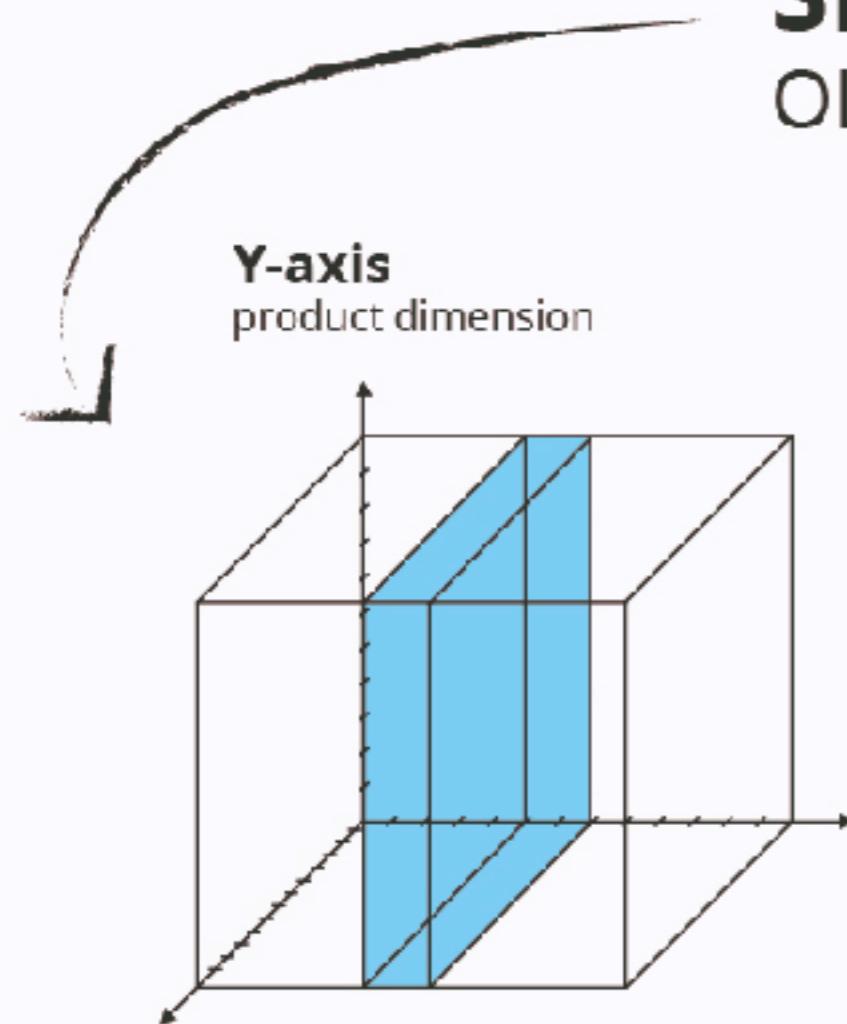
ItemID (itemid, category, brand, color, size)

CustID (custid, name, address)

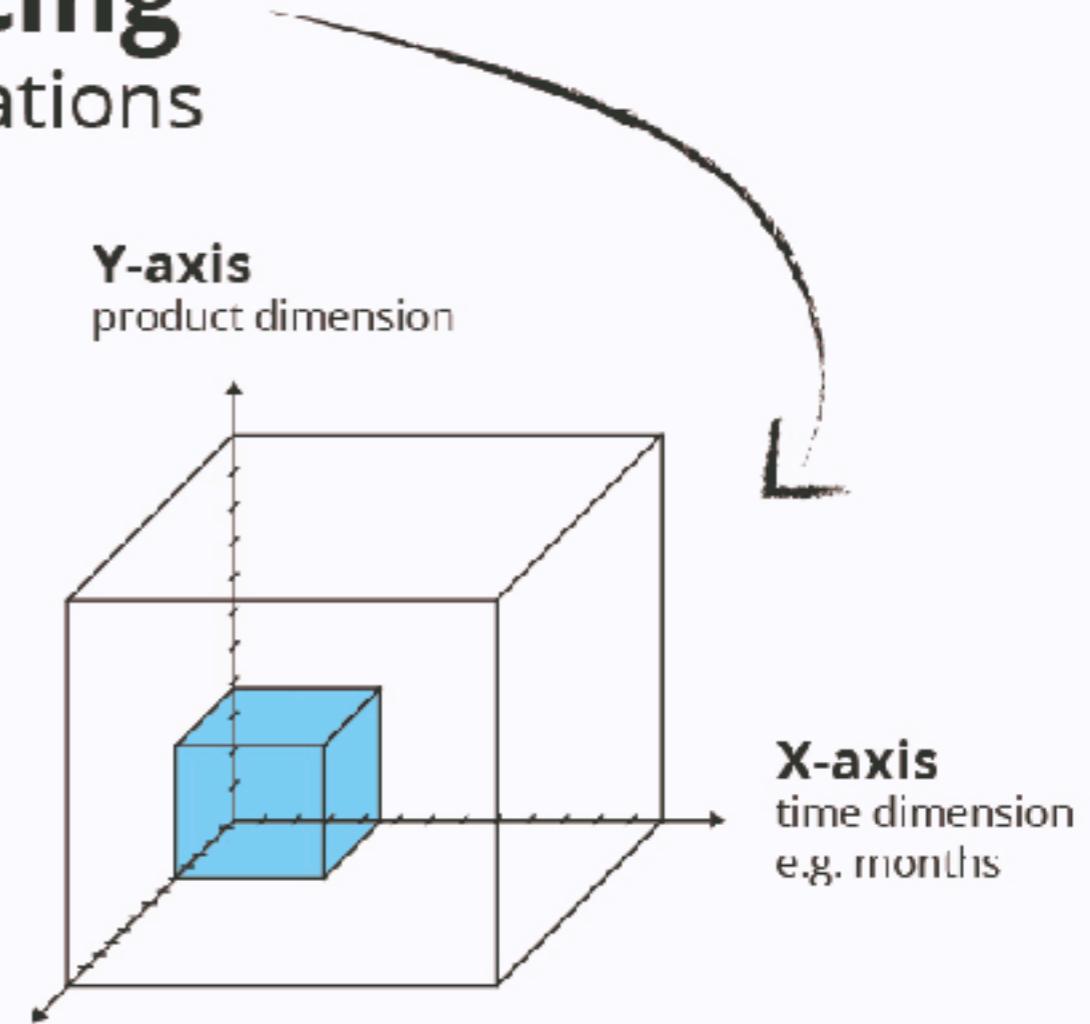


# Slicing & Dicing

OLAP cube operations



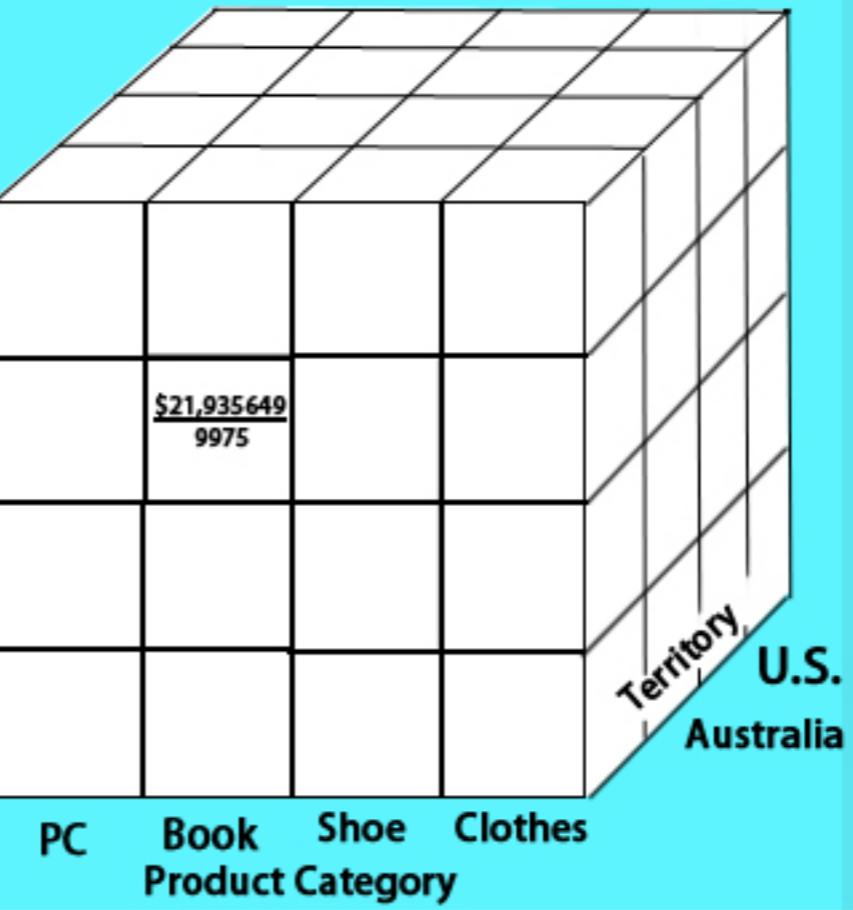
**Z-axis**  
customer dimension

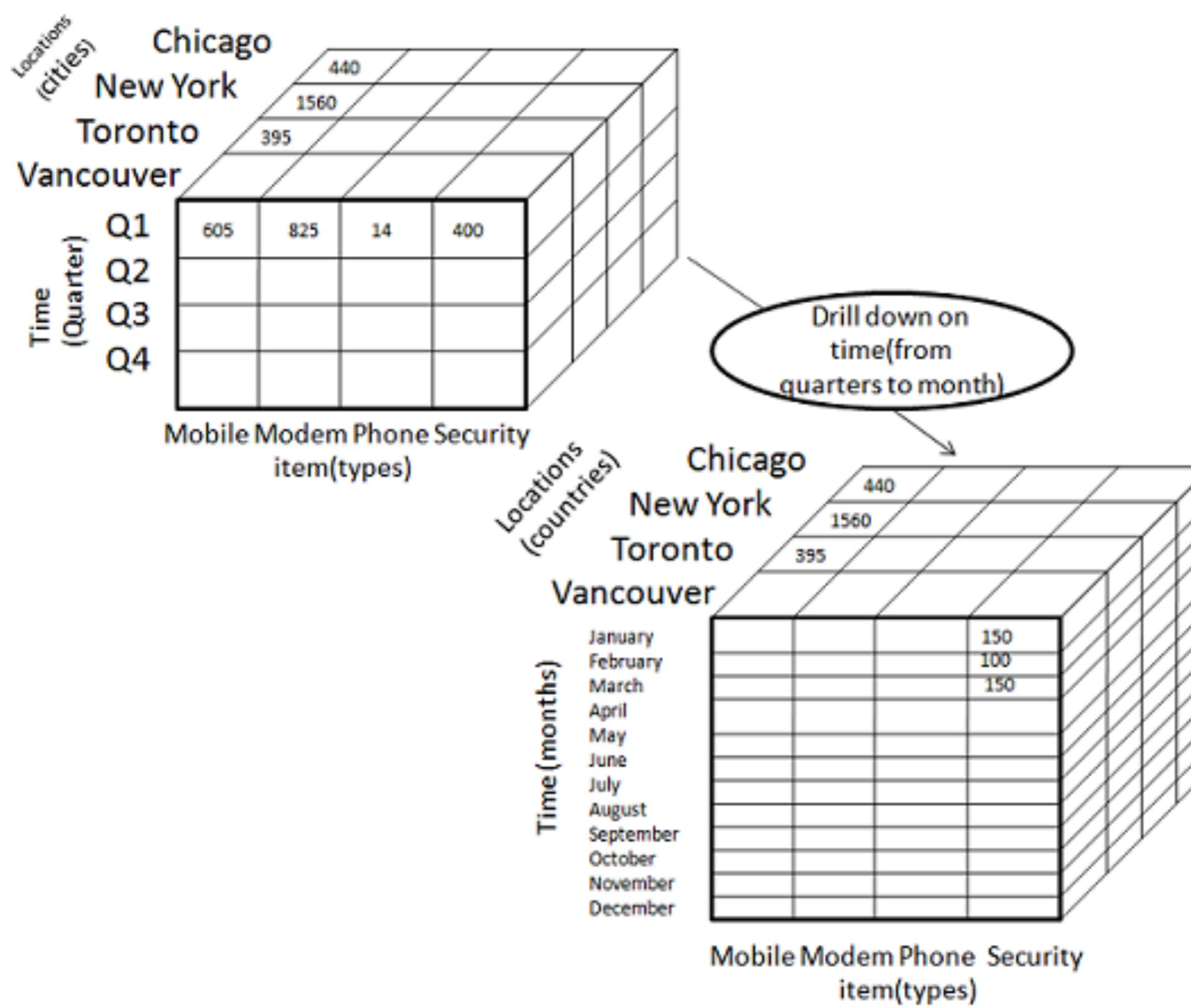


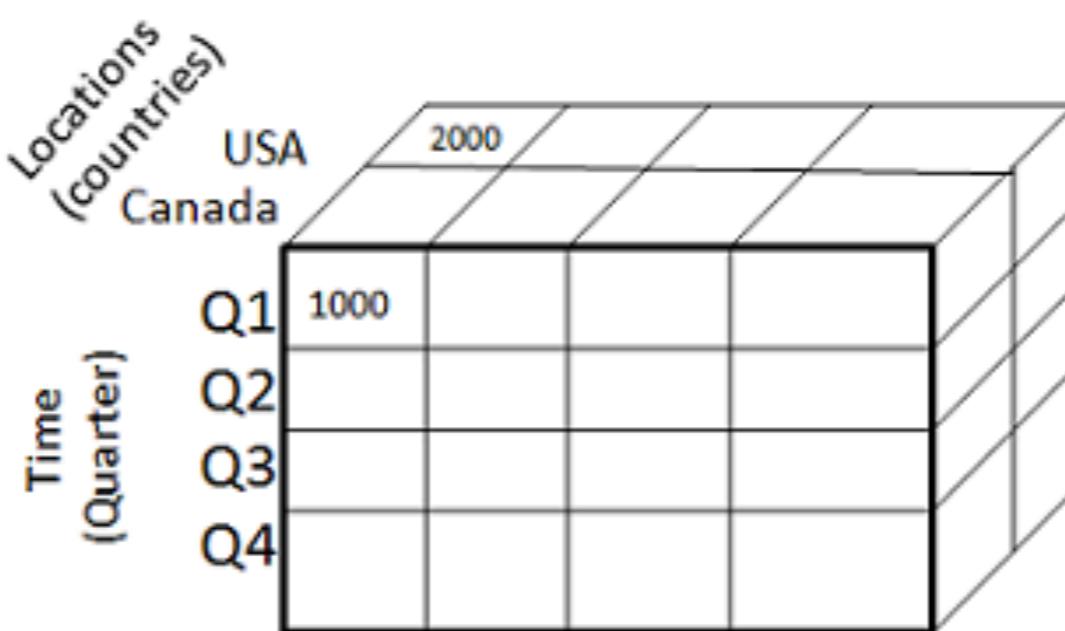
**Z-axis**  
customer dimension

## OLAP CUBE

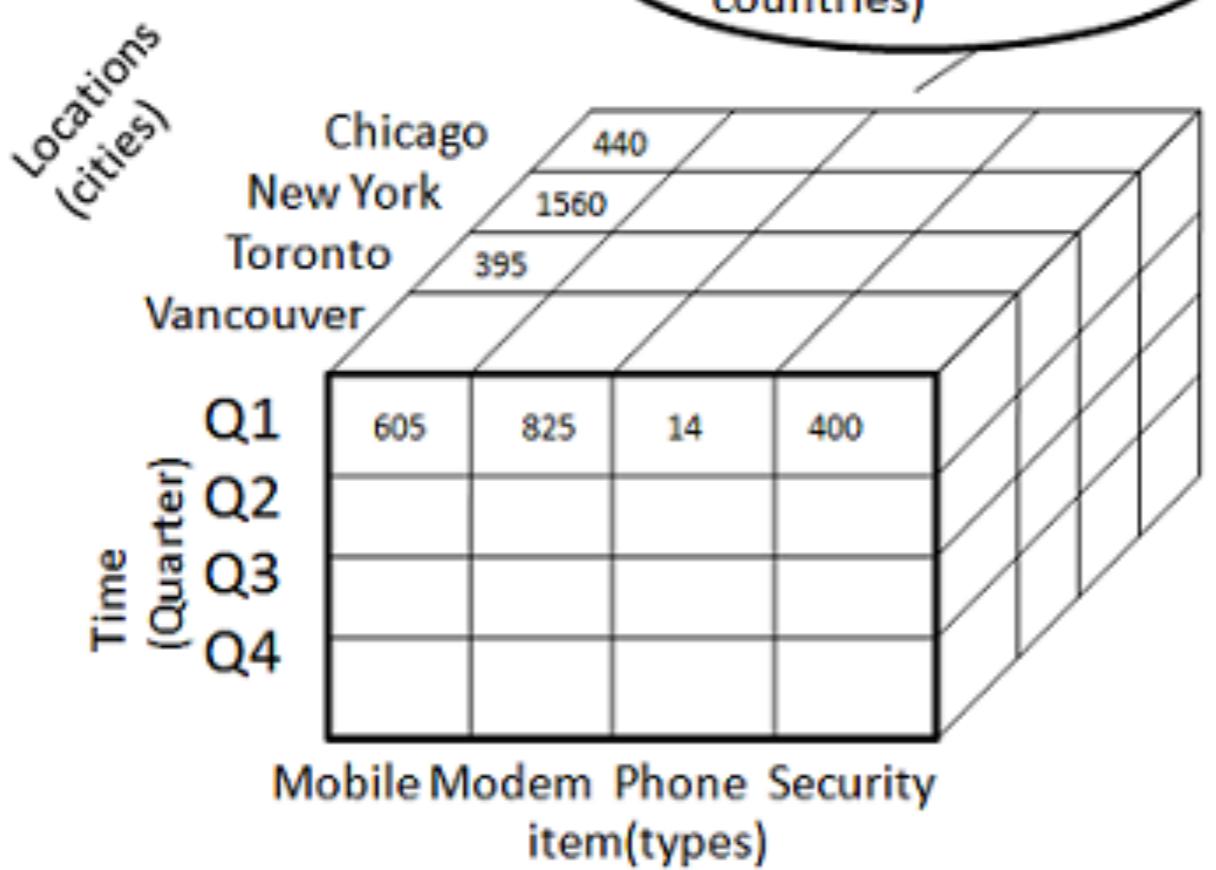
Quarters  
Q1  
Q2  
Q3  
Q4





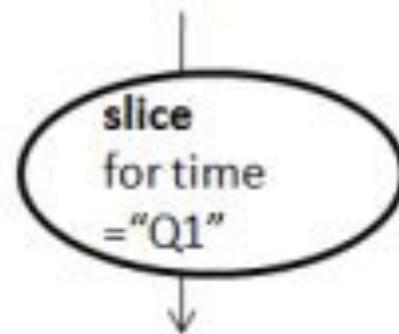


**roll-up** on location  
(from cities to  
countries)



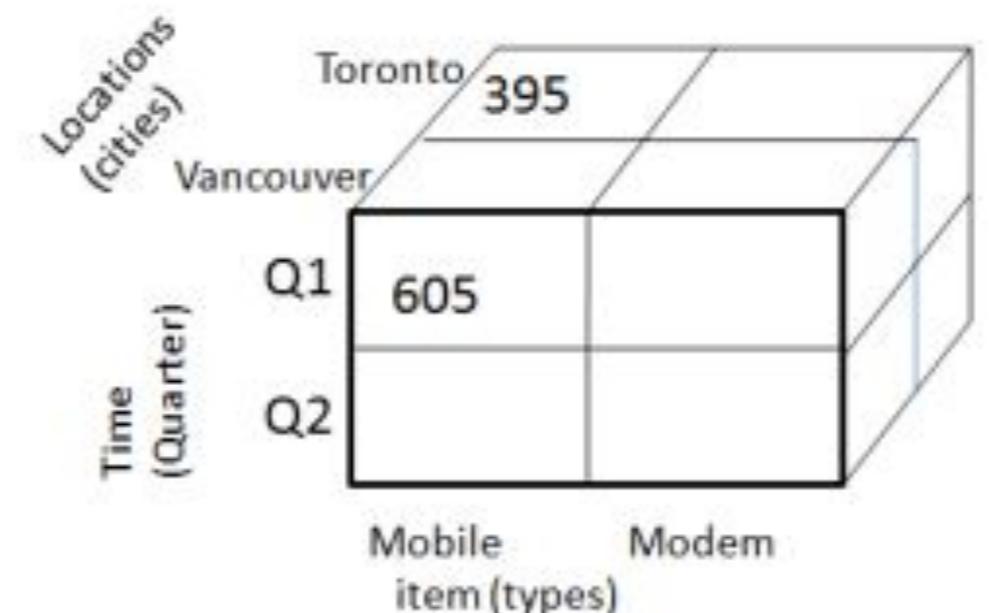
		Mobile Modem Phone Security				
		item(types)		item(types)		
Locations (cities)	Time (Quarter)	Chicago		New York		Toronto
		440		1560		395
		605		825		14
		Q1		400		
		Q2				

Mobile Modem Phone Security  
item(types)



		Mobile Modem Phone Security				
		item(types)		item(types)		
Locations (cities)	Time (Quarter)	Chicago		New York		Toronto
		440		1560		395
		605		825		14
		Q1		400		
		Q2				

Mobile Modem Phone Security  
item(types)



Dice for (location = "Toronto" or "Vancouver")  
and (time = "Q1" or "Q2") and  
(item = "Mobile" or "Modem")



The diagram illustrates the transpose operation on a matrix. The original matrix has "Locations (cities)" as rows (Chicago, New York, Toronto, Vancouver) and "Item (types)" as columns (Mobile, Modem, Phone, Security). The transpose operation swaps these, resulting in a new matrix where "Item (types)" are rows and "Location (cities)" are columns. A central oval labeled "Pivot" indicates the point of transformation.

605	825	14	400

Mobile Modem Phone Security  
item(types)

↓  
Pivot  
↓

			605
			825
			14
			400

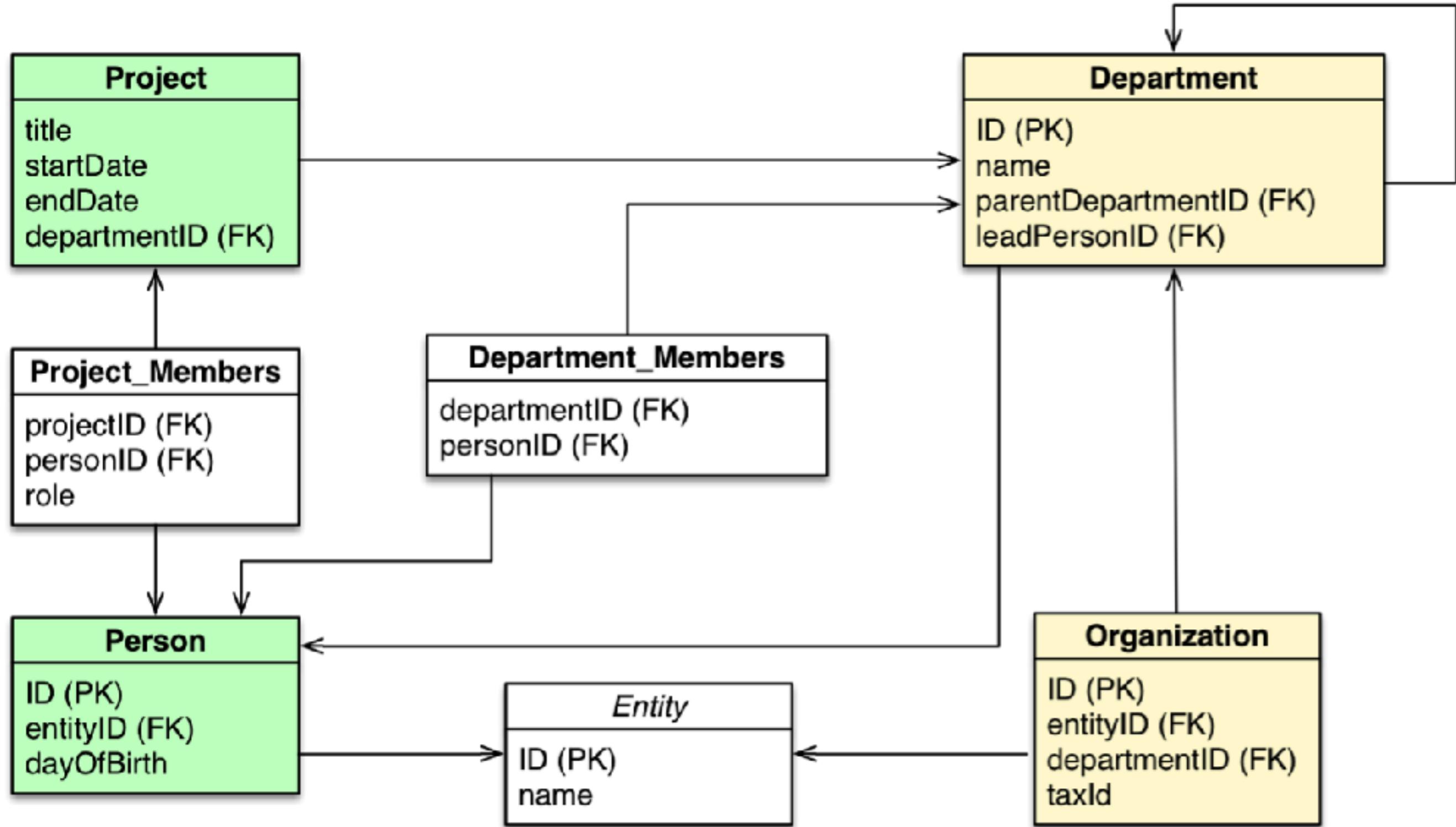
Mobile  
Modem  
Phone  
Security

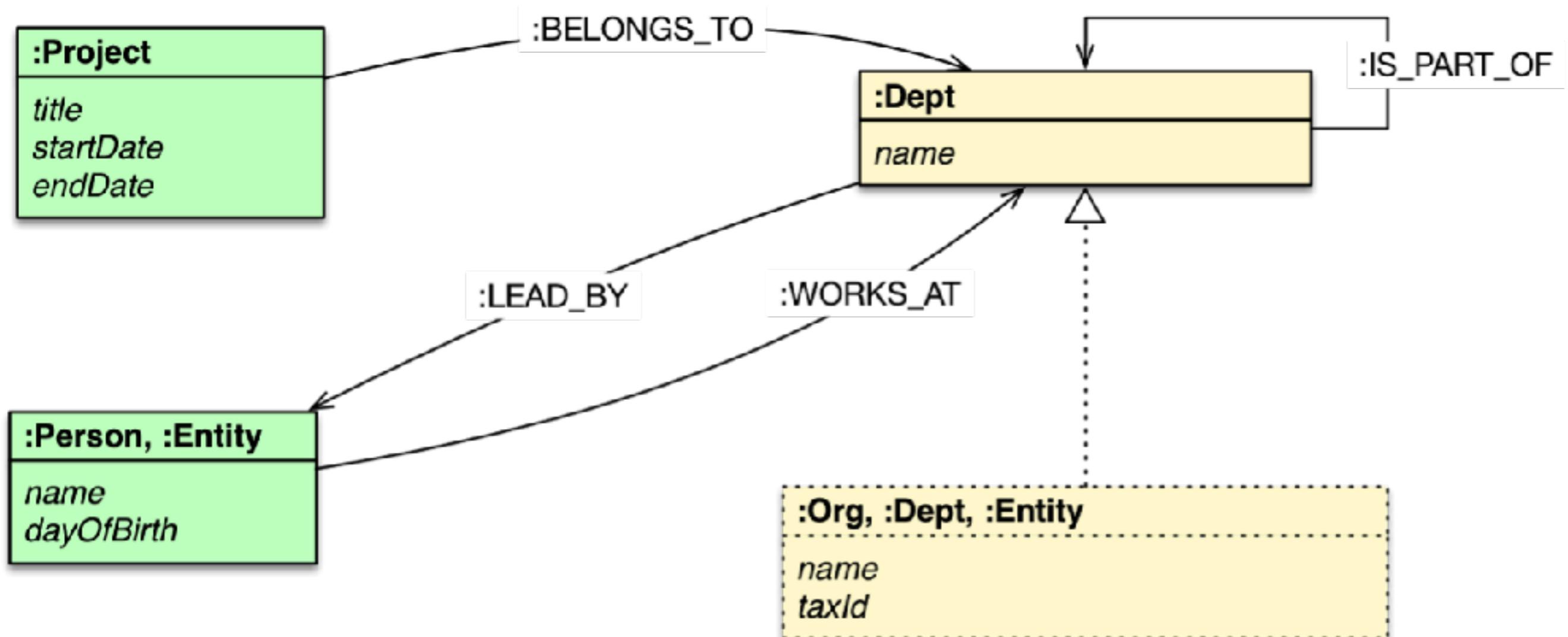
Chicago New York Toronto Vancouver  
Location (Cities)

	<b>User</b>	
One-to-One	<pre> User {   "_id": "ObjectId('AAA')",   "name": "Joe Karlsson",   "company": "MongoDB" } </pre>	
One-to-None	<pre> User {   "_id": "ObjectId('AAA')",   "name": "Joe Karlsson",   "company": "MongoDB",   "addresses": [     { "street": "45 Sesame St", "city": "Los Angeles" },     { "street": "123 Avenue Q", "city": "New York" }   ] } </pre>	
One-to-Many	<pre> Product {   "name": "left-handed smoke shifter",   "manufacturer": "Acme Corp",   "catalog_number": "1234",   "parts": [     "ObjectId('AAAA')",     "ObjectId('BBBB')",     "ObjectId('CCCC')"   ] } </pre>	
One-to-Many	<pre> Part {   "_id": "ObjectId('AAAAA')",   "partno": "123-aff-456",   "name": "V4 gizmojet",   "qty": 94,   "cost": "0.94",   "price": "3.99" } </pre>	
One-to-Many	<pre> Host {   "_id": ObjectId("AAAB"),   "name": "goofy.example.com",   "ipaddr": "127.66.66.66" } </pre>	
One-to-Many	<pre> Host {   "_id": ObjectId("AAAB"),   "name": "goofy.example.com",   "ipaddr": "127.66.66.66" } </pre>	
One-to-Many	<pre> Log {   "time": ISODate("2014-03-28T09:42:41.88Z"),   "message": "cpu is on fire!",   "host": ObjectId("AAAB") } </pre>	
Many-to-Many	<pre> User {   "_id": ObjectId("AAF1"),   "name": "Kate Monster",   "tasks": [     ObjectId("ADF9"),     ObjectId("AE02"),     ObjectId("AE73")   ] } </pre>	
Many-to-Many	<pre> Task {   "_id": ObjectId("ADF9"),   "description": "Write blog post about schema design",   "due_date": ISODate("2014-04-01"),   "owners": [ObjectId("AAF1"), ObjectId("BGGG")] } </pre>	

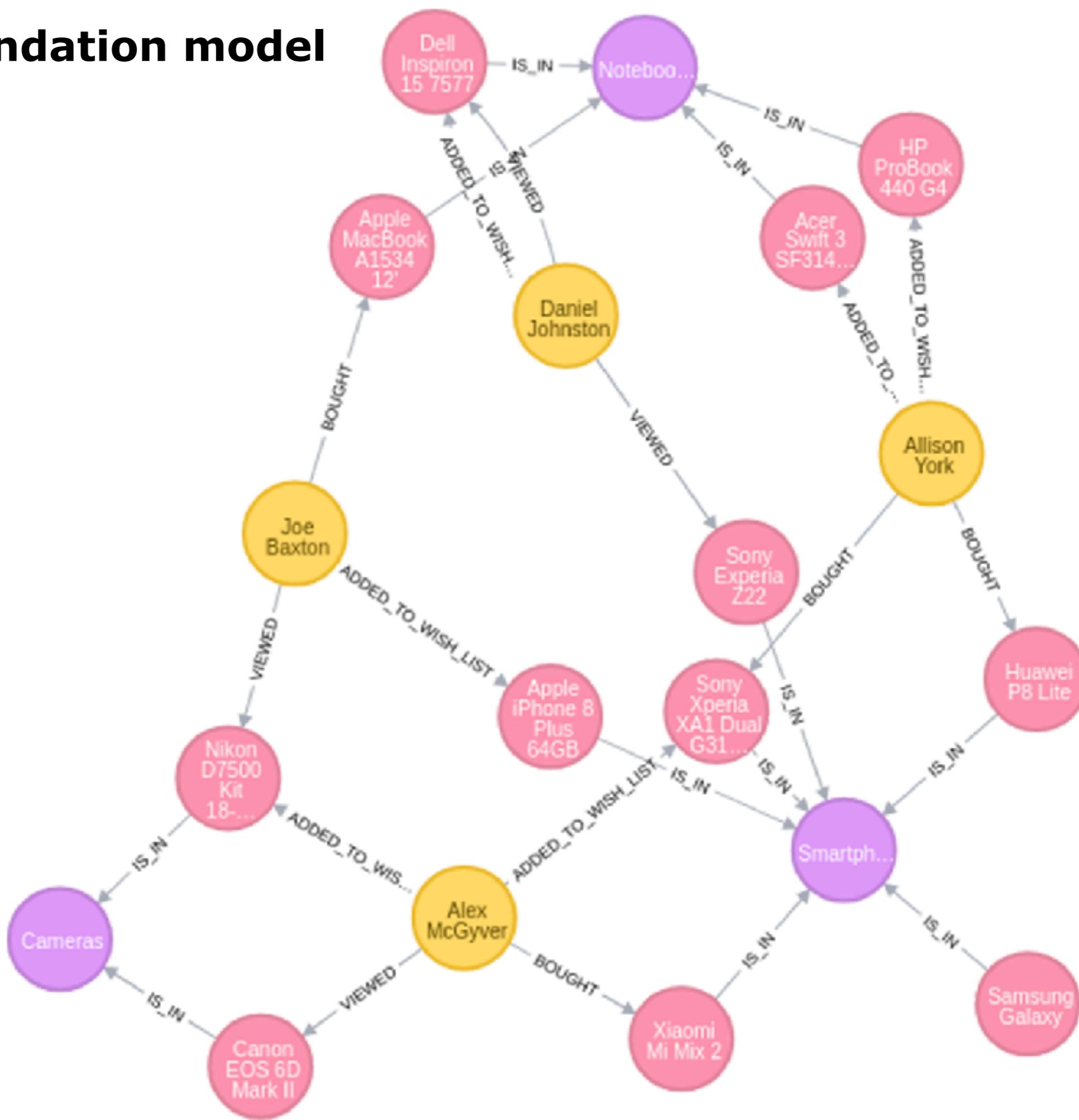
# Product Catalog

```
{  
    type: "Audio Album",  
    title: "A Love Supreme",  
    description: "by John Coltrane",  
  
    shipping: {  
        weight: 6,  
        dimensions: {  
            width: 10,  
            height: 10,  
            depth: 1  
        },  
    },  
  
    pricing: {  
        list: 1200,  
        retail: 1100,  
        savings: 100,  
        pct_savings: 8  
    },  
  
    details: {  
        title: "A Love Supreme [Original Recording Reissued]",  
        artist: "John Coltrane",  
        genre: [ "Jazz", "General" ],  
        tracks: [  
            "A Love Supreme Part I: Acknowledgement",  
            "A Love Supreme Part II - Resolution",  
            "A Love Supreme, Part III: Pursuance",  
            "A Love Supreme, Part IV-Psalm"  
        ],  
    },  
}
```





# Recommendation model



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

geojson.io

Not secure | geojson.io/#map=2/20.1/-0.2

Open Save New Share Meta unsaved

JSON Table Help anon | login

North Atlantic Ocean

South Atlantic Ocean

3000 km  
2000 mi

Mapbox Satellite OSM OSM

```

1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {},
7       "geometry": {
8         "type": "LineString",
9         "coordinates": [
10           [
11             -31.9921875,
12             23.241346102386135
13           ],
14           [
15             -4.21875,
16             39.90973623453719
17           ]
18         ]
19       }
20     }
21   ]
22 }
```

```
[  
  {  
    "name": "device.status",  
    "tags: {  
      "deviceId" : "aliyun-tablestore",  
      "version" : "2.0",  
      "model" : "unify"  
    },  
    "location": {  
      "lat" : 30.1276799186,  
      "lon": 120.0848161441  
    },  
    "timestamp" : 1531918377012, → When  
    "values" : {  
      "status" : "active",  
      "temperature" : 30,  
      "cpu" : 50  
    }  
  }  
]
```

Who At what time What Where Related to what?

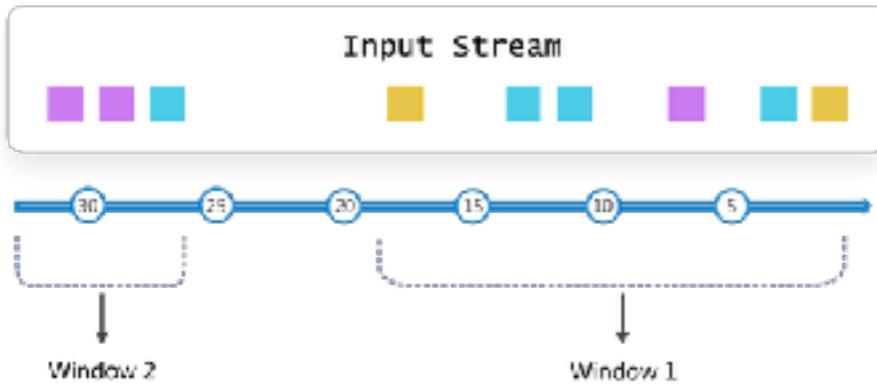
WHO

WHEN

WHERE

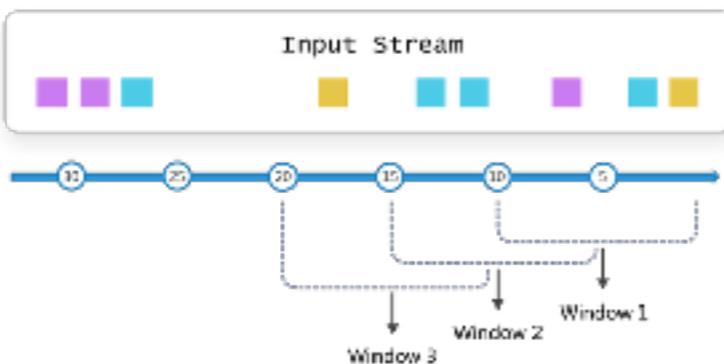
WHAT

## Session Window



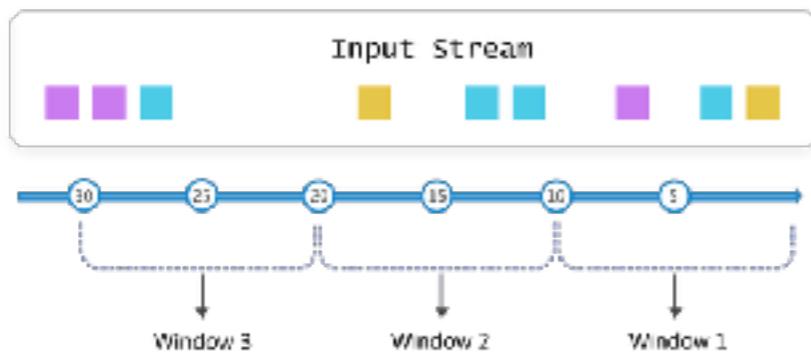
Session window boundaries are inactivity periods when there is no incoming data.

## Sliding Window



if we have a window size of 10 seconds with a sliding offset of 5 seconds, windows will slide every 5 seconds to create a new window of 10 seconds

## Tumbling Window



fixed-sized and non-overlapping window where each element is bound to a single window.

metric	timestamp	cluster	hostname	metric value
cpu	2015-04-28T17:50:00Z	Cluster-A	host-a	10
cpu	2015-04-28T17:50:10Z	Cluster-A	host-b	20
cpu	2015-04-28T17:50:20Z	Cluster-A	host-a	5
iops	2015-04-28T17:50:00Z	Cluster-A	host-a	10
iops	2015-04-28T17:50:10Z	Cluster-A	host-b	30
iops	2015-04-28T17:50:20Z	Cluster-A	host-a	8

Metrics

Timestamp

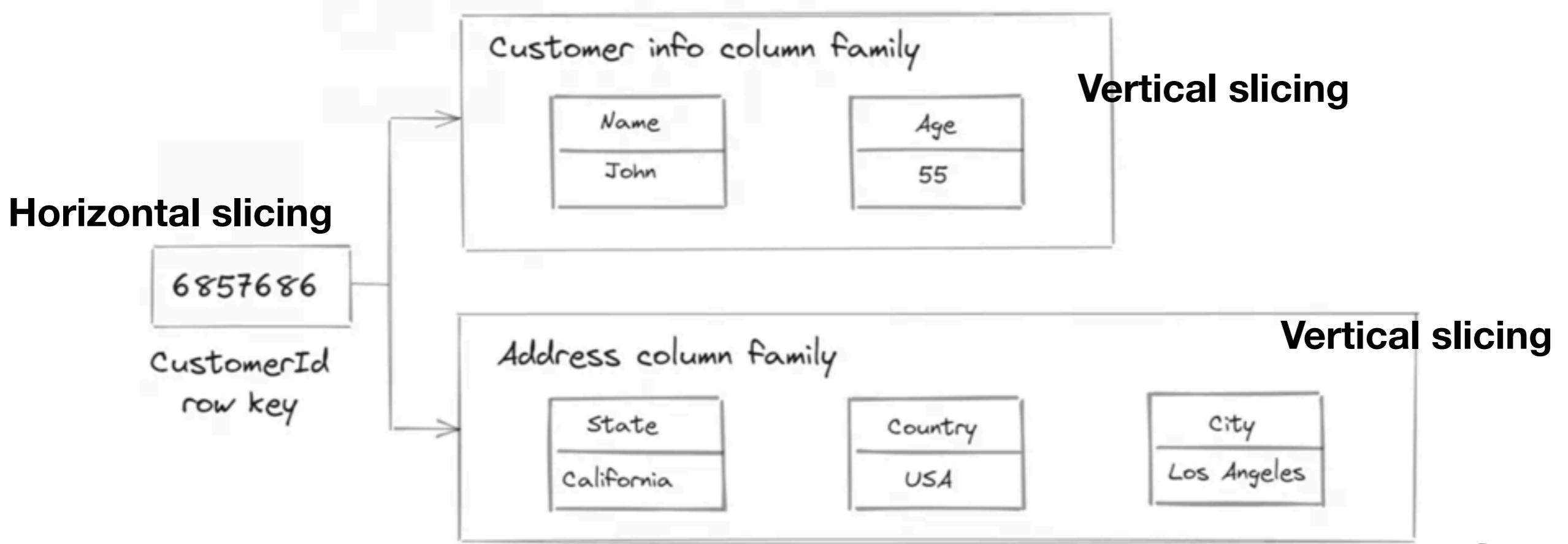
Subject dimension

Measurements

super column family

companies				
row key 1	address		website	
	city	San Francisco	subdomain	www
	state	California	domain	grio.com
	street	Kearny St.	protocol	http

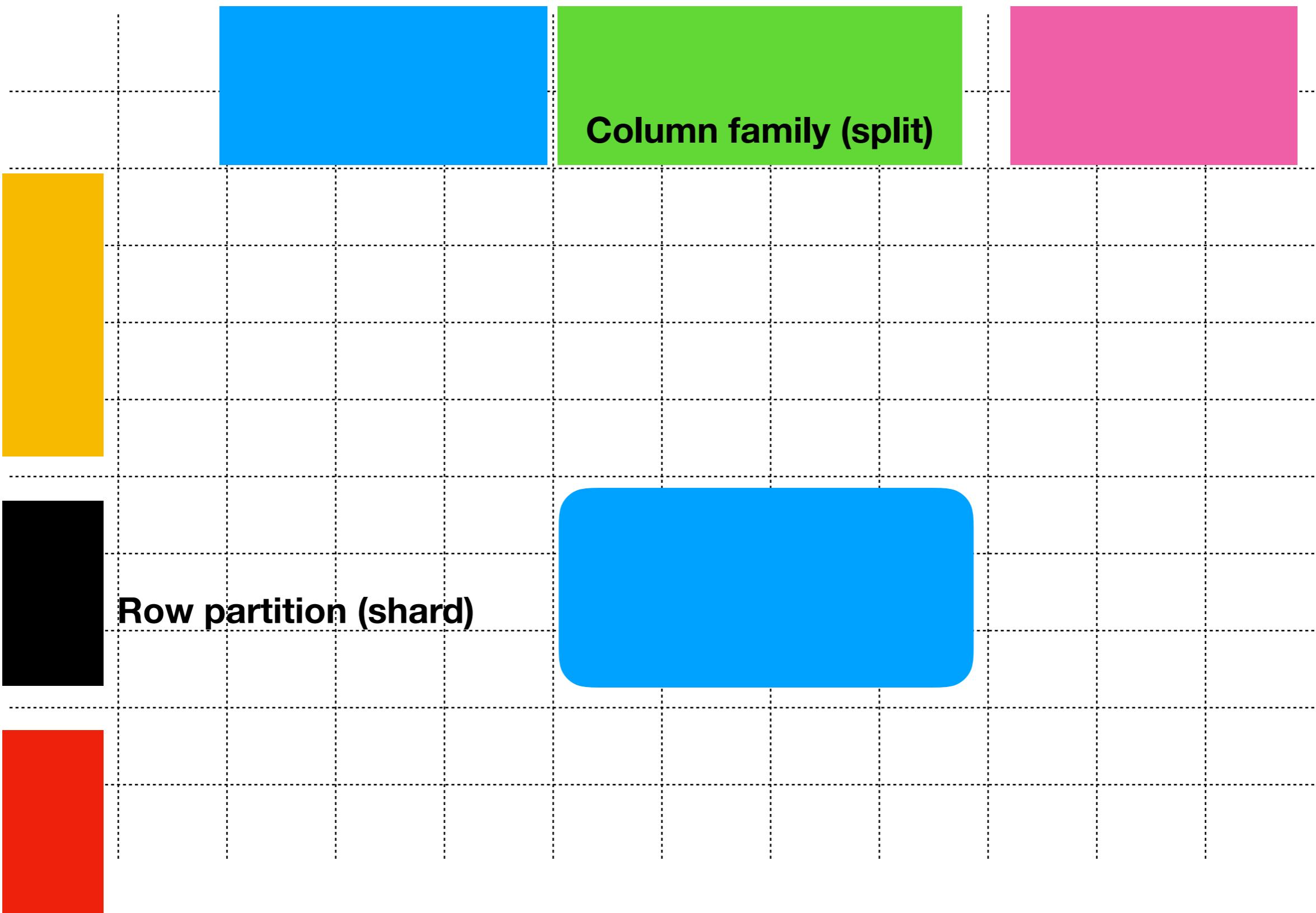
column family      column



Skan.

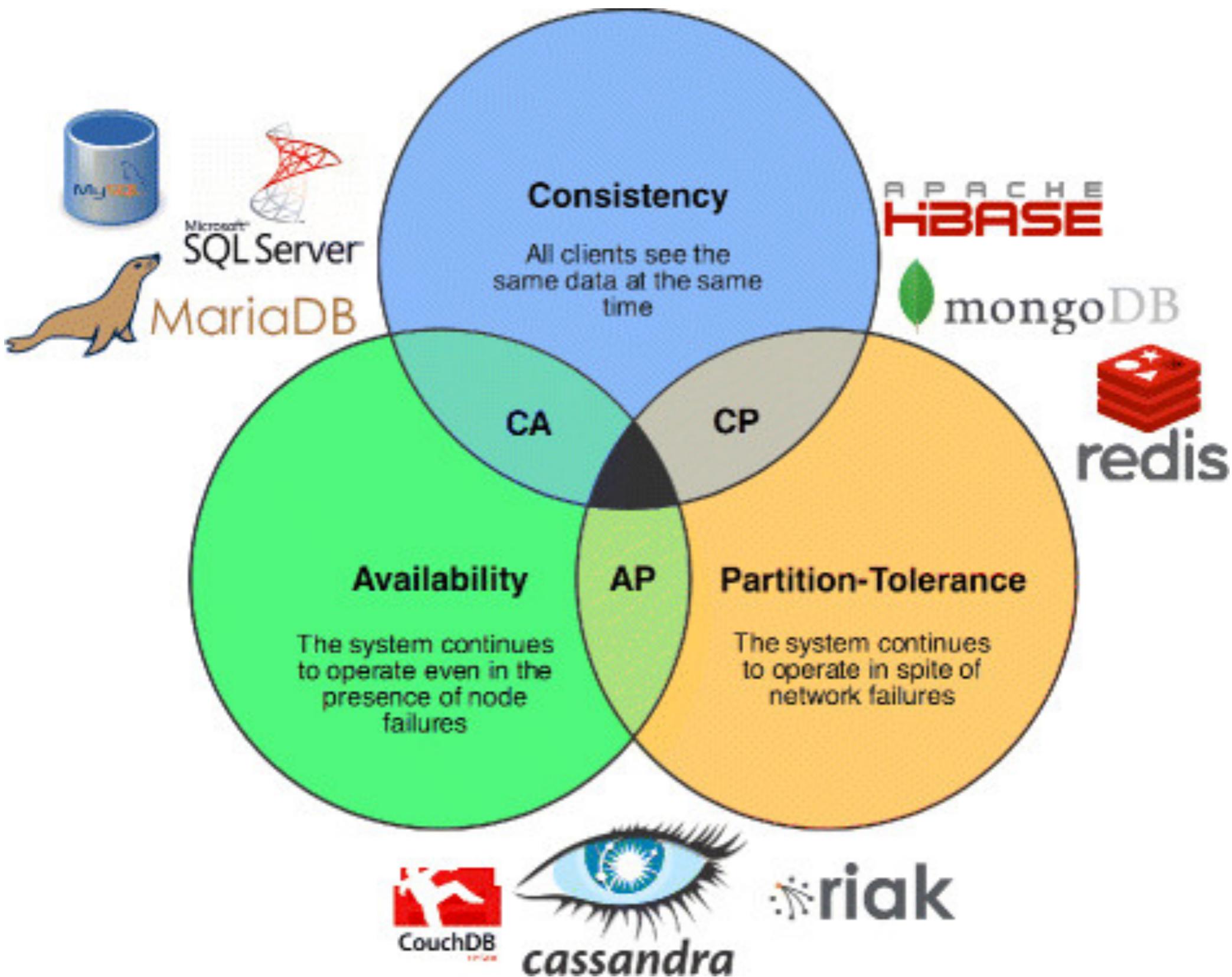
© SKAN.AI 2019 CONFIDE

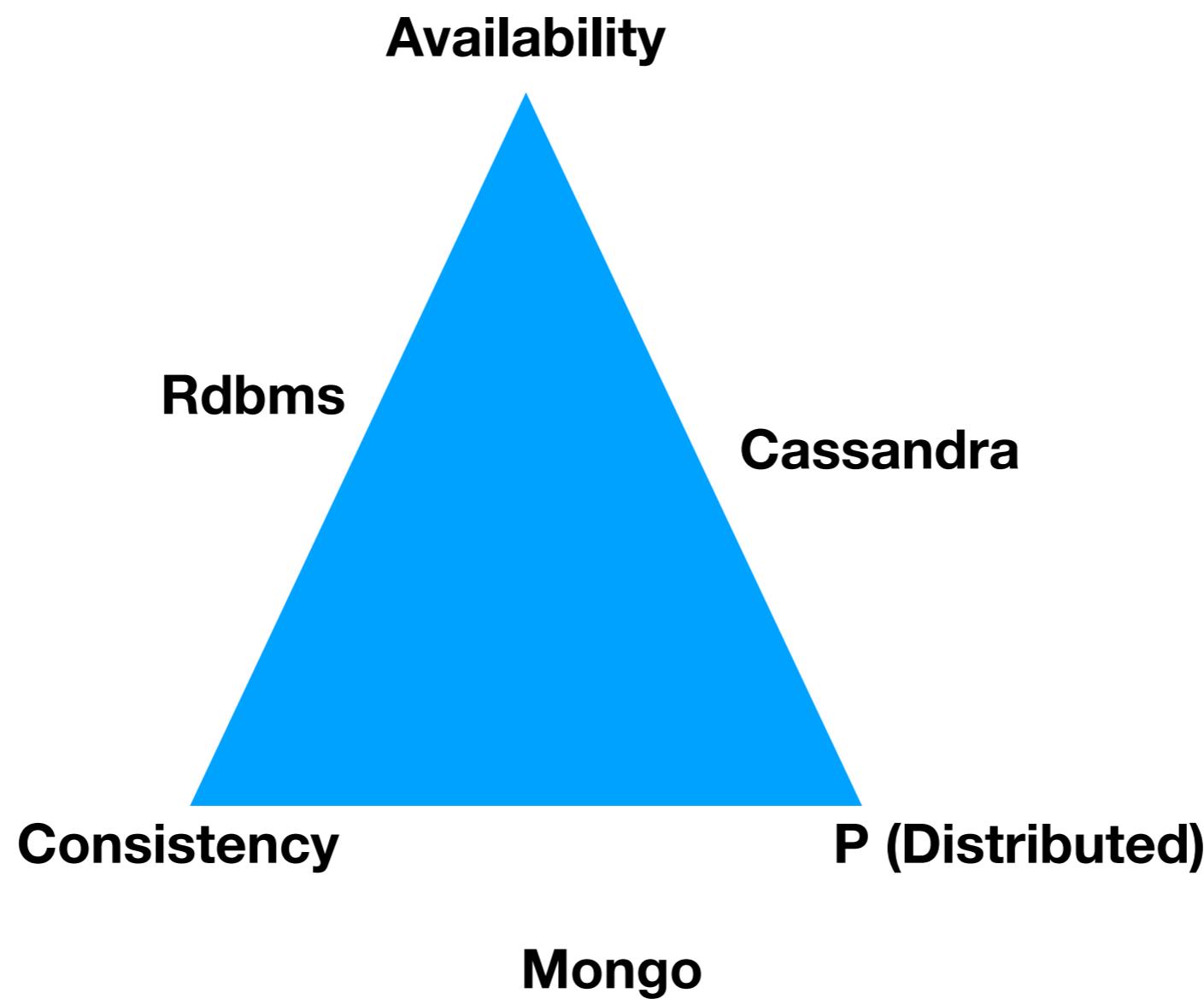
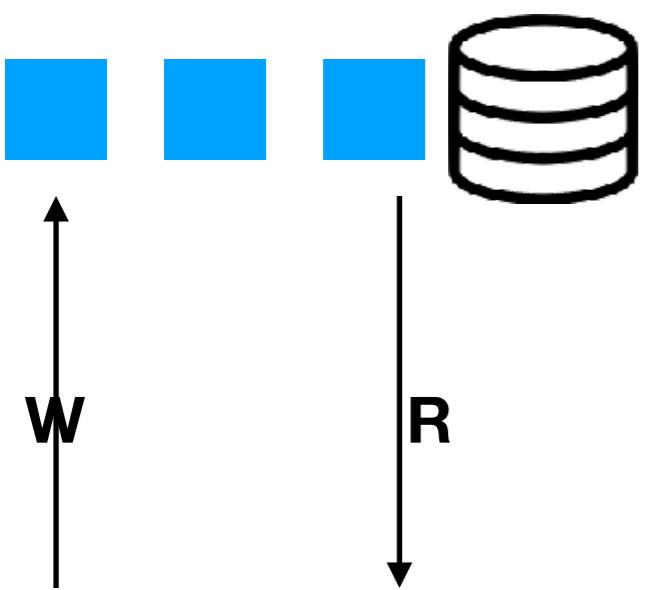
**Shard id, row id, column family id, column**



## Key Factors that Drive the Decision

	Mongo	Cassandra	HBase
	Document	Wide column	Wide column
Rich Object Model	++		
Secondary Indexes	++		
Replication	Master slave	Multi master	Master slave
High Availability		++	
Write Scalability		++	
Rich Query Language		++	
Schema	++		
CAP	CP	AP	CP
Performance			
Aggregation	++		





	MongoDB	Cassandra
Expressive object model	Yes	No
Secondary indexes	Yes	No
No downtime on node failure	No	Yes
High write throughput	No	Yes

<https://benchant.com/blog/mongodb-vs-cassandra>

If you need query language support, Cassandra is the better fit for you.

### Custom Reports



### Data LakeHouse



### Data warehouse



GreenPlum



Catalogue

Access Control

### Query Engine



Hive

Flink

### Table Format



### Data Format

Apache Avro, Apache ORC

CSV

Json



# on prem (   
 # On cloud k8s (aws,azure)   
 # SAAS

Azure/AWS vm's

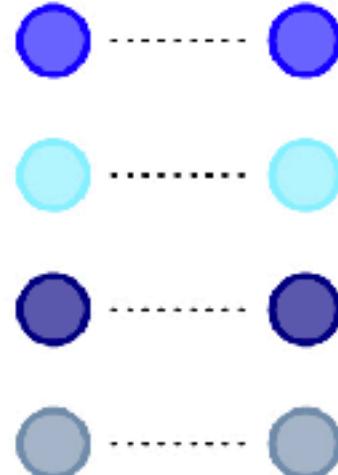
### Storage



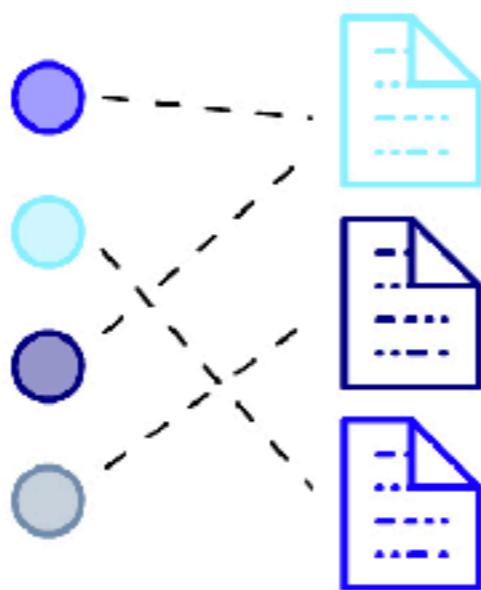
Skan.

© SKAN.AI 2019 CONFIDE

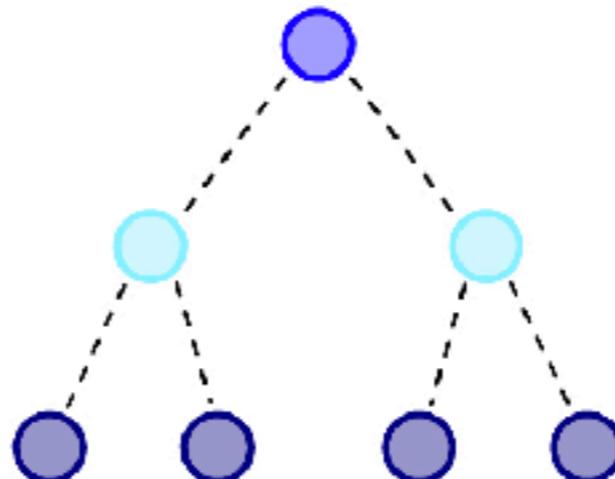
Key-Value

{  
}

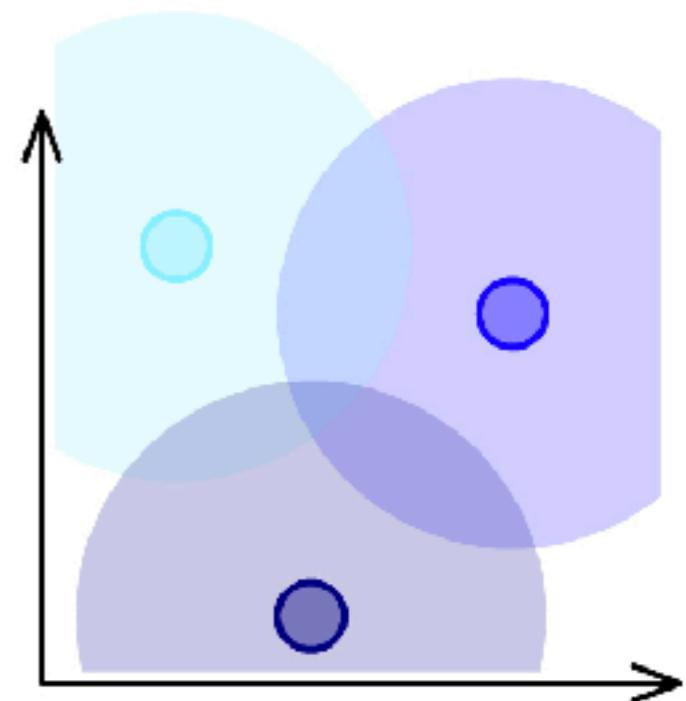
Documents



Graph



Vector Database



**Pinecone**

# Use Cases

# Content Management System

## Our Confusing Economy

Joe Weisenthal | Nov. 30, 2012, 9:59 AM | 693 | 3

Recommend 1 Share 1 Tweet 27 Email 0 More

Within every Chicago PMI (.pdf) report there are anecdotal comments from the survey's panel of manufacturing companies.

This month we were amused by these two back-to-back characterizations of the economy.

One guy sees things collapsing. Another says his manufacturing company is "officially swamped."



Patrick Denker via flickr

- Business sales been slowing throughout the year, and continue to slow, but now at an increasing rate, becoming very alarming.
- We are officially swamped and doing everything we can to get our machines out this year....some will bleed into 2013, but we will make up a large portion of our yearly revenue in Q4.

So yeah. If the economic coverage seems bipolar at times, that's how it is on the ground, too.

Recommended For You

Headline

Byline, Date, Comments

Copy

Image

Related Stories

# Fraud Detection

- We extract entities from a large database of business activities, make relationships between them.
- We can then use a technique called “Entity Link Analysis” to identify suspicious links between types of entities that may indicate fraudulent behaviour.
- We will use entities from people such as names and dates of birth, as well as contextual entities such as IP addresses, device identifiers and access times.
- We can then analyse the links between these entities and mark up those that have previously been marked as fraudulent. When entities that are marked as fraudulent start to intersect those that are not marked as fraudulent, suspicions begin to arise. For example, use of multiple bank accounts on a single device that has been previously used to access fraudulent accounts.

# Product Catalogue

New Arrivals  
Top Rated  
The Trend Spot  
Luxury  
Sandals Under \$40

Special Lists

Shop by Category

Sandals  
Gladiators  
Wedges  
Flat Sandals  
Dress Sandals  
Evening Sandals  
Casual Sandals  
Flip Flops & Beach  
Comfort Sandals  
Wide Width Sandals

Browse by category

Boots  
Pumps & Heels  
Flats  
Oxfords & Lace-Ups  
Loafers & Slip-Ons  
Evening & Wedding  
Comfort  
Wide Width  
Sneakers  
Athletic  
Work & Safety  
Slippers  
Clearance

Filter by: Brand Size Color Heel Height Sort by: All

Size: 9.5

Heel Height: High

Sort by: All



Steve Madden Nala Platform Pump  
**\$59.95**  
Compare at \$99.00  
More Colors



Zigi Soho Khloe Striped Sandal  
**\$79.95**  
Compare at \$100.00  
More Colors



Patrizia by Spring Step Extravagant Wedge Sandal  
**\$39.95**  
Compare at \$60.00  
More Colors



Levity Austin Platform Sandal  
**\$59.95**  
Compare at \$89.00  
More Colors



Franco Sarto Sassy Wedge Sandal  
**\$59.95**  
Compare at \$85.00  
More Colors



Diba Rosey Wedge Sandal  
**\$39.95**  
Compare at \$59.00  
More Colors



CL by Laundry Nolita Wedge Pump  
**\$39.95**  
Compare at \$54.00  
More Colors

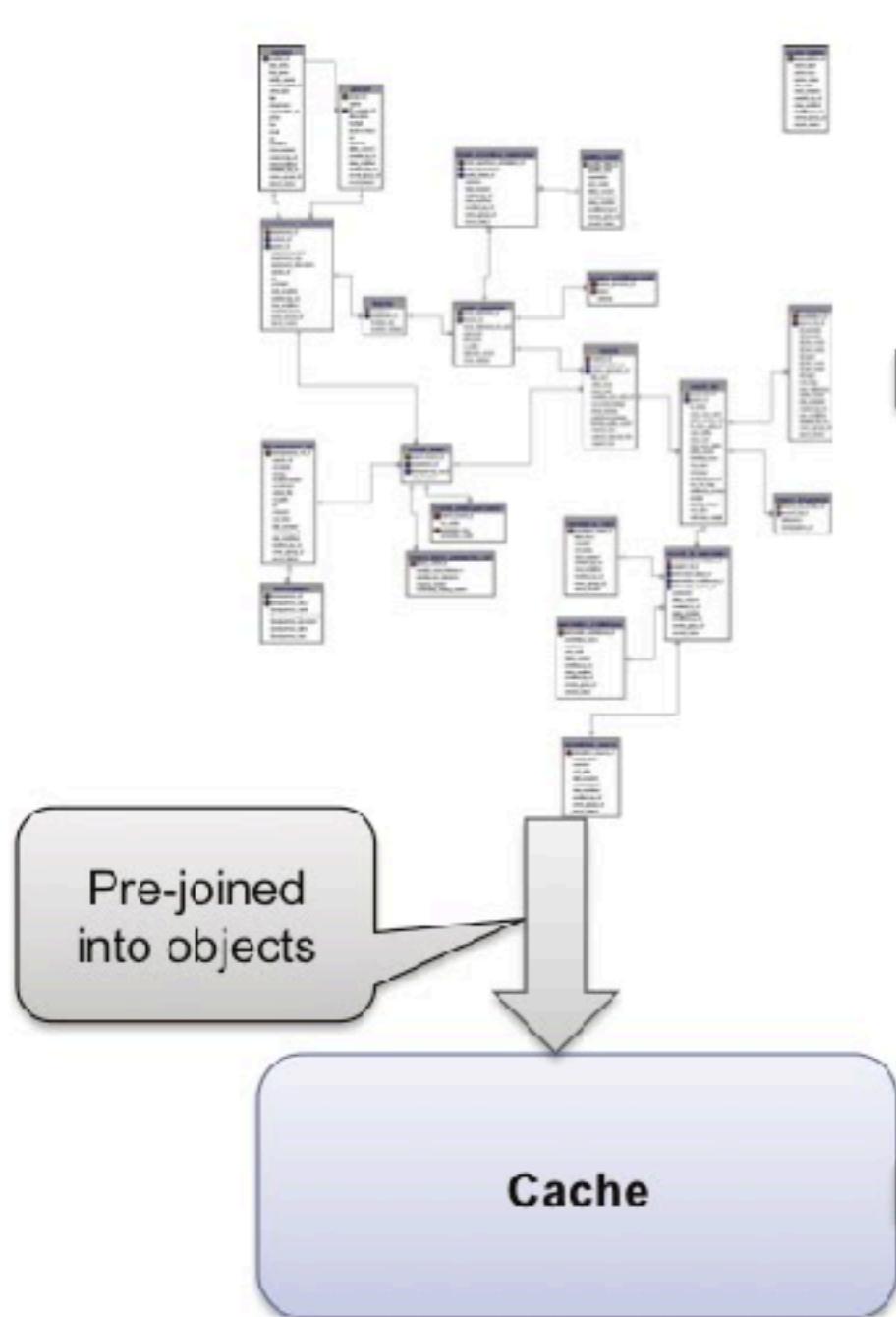


Lulu Townsend Shania Platform Pump  
**\$59.95**  
Compare at \$79.00  
More Colors

Filter by attributes

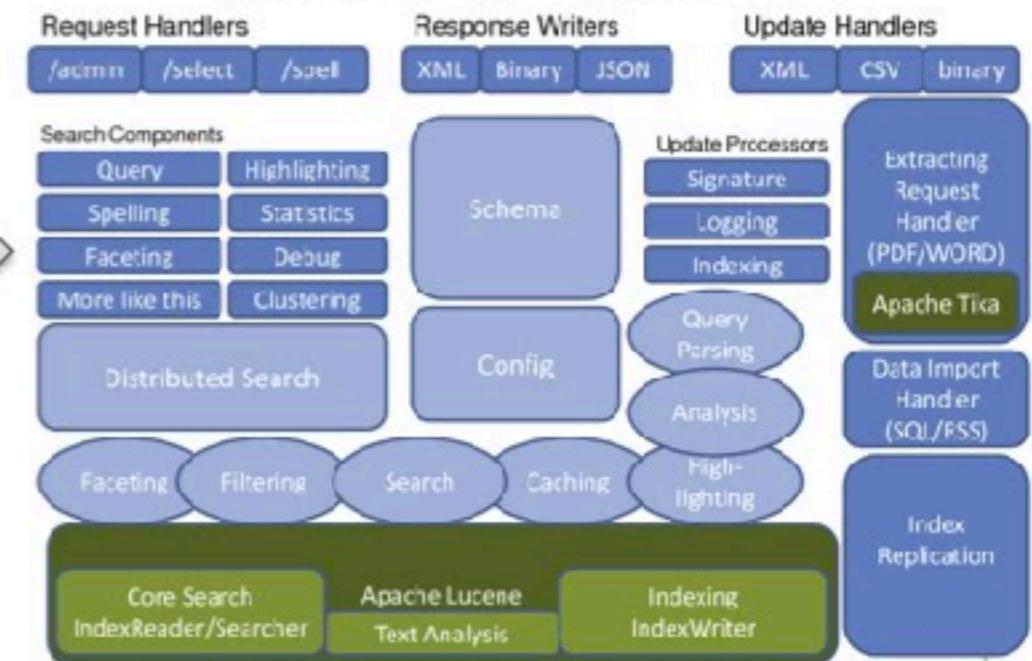
Lists hundreds of item summaries

# Product Data Store



# Product Search

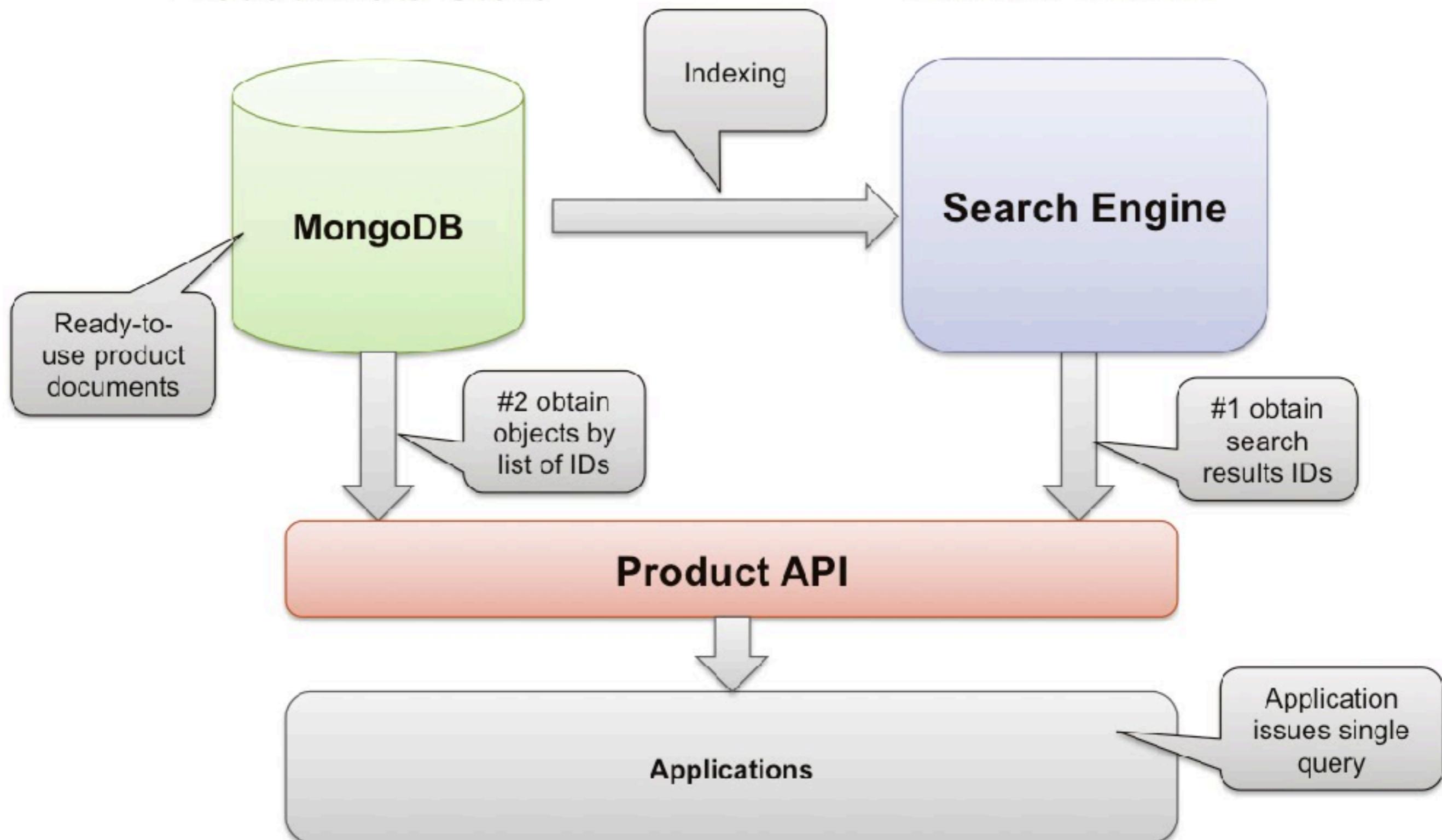
## Lucene/Solr Architecture



- 3 different systems to maintain: RDBMS, Search engine, Caching layer
- RDBMS schema is complex and static
- Applications needs to talk many languages

## Product Data Store

## Product Search



- With the API layer, the application just issues a single call, thus reducing complexity and latency
- No need for a caching layer since MongoDB's read performance is close to Memcache and the likes
- The schema is dynamic and maps well to the API calls

## Product (Book)

```
key:  
products::books::chasm_city  
  
document:  
{  
  "type": "book",  
  "title": "Chasm City",  
  "author": "Alastair Reynolds",  
  "skus": [  
    {  
      "sku": 123456,  
      "isbn-13": "9780441010646",  
      "format": "paperback",  
      "pages": 704  
      "published": 2003  
      "price": 9.99},  
    {  
      "sku": 234567,  
      "isbn-13": "9780575068773",  
      "format": "hardcover",  
      "pages": 528  
      "published": 2001  
      "price": 19.99}]  
    "reviews": {  
      "average": 4.4,  
      "count": 26},  
    "categories": [  
      "Science Fiction and Fantasy",  
      "Science Fiction",  
      "Space Opera"]  
  }
```

## Product (Movie)

```
key:  
products::movies::2001_a_space_odyssey  
  
document:  
{  
  "type": "movie",  
  "title": "2001: A Space Odyssey",  
  "director": "Stanley Kubrick",  
  "released": 1968  
  "skus": [  
    {  
      "sku": 789012,  
      "format": "blu-ray",  
      "duration": 149,  
      "price": 9.99,  
      "reviews": {  
        "average": 4.7,  
        "count": 150}},  
    {  
      "sku": 890123,  
      "format": "laserdisc",  
      "duration": 149,  
      "price": 124.99,  
      "reviews": {  
        "average": 4.5,  
        "count": 130}}]  
    "categories": [  
      "Science Fiction"]  
  ]
```



An inventory system that might want full ACID. I was very unhappy when I bought a product and they said later they were out of stock. I did not want a compensated transaction. I wanted my item!

# Asset Management

Tool & Asset Manager 2.0 --- Trial period : 29 days remaining

General Configuration Security Reports

Assets Personnel Reminders Booking Completed services Notes Parts inventory Orders Technical support Buy now

Global management General

**Assets**

Check in Check out Locations Asset type

Search

Picture	Barcode	Status	Asset type	Description	Manufacturer
	AST1000000	Checked out	Electronics	Laptop computer	Dell
	AST1000001	Checked out	Machinery	Scissor lift	Platform
	AST1000002	Checked out	Electric tools	Circular saw	DeWalt
	AST1000003	Out for repair	Electric tools	Oscillating tool	DeWalt
	AST1000004	In storage	Other tools	Shovel	
	AST1000005	In storage	Electronics	Printer	Hewlett pack...

Active  Inactive

**General info.** User-defined fields (simple) User-defined fields (with date)

Description: Laptop computer  
Manufacturer: Dell  
Model: XPS 14  
Serial number: XV1Z203405220  
Condition: Good  
Location: Unit A 101  
Barcode: AST1000000  
Asset type: Electronics

Status: Checked out  
Due return date:  
Checked out to:

Barcode: PRR100004  
Name: ANDERSON, Louise  
Phone #: 959-555-6542

Notes:

Transactions Log Booking Service schedule Completed services Parts used Incidents Notes Attached files Reminders

Reminder type Detail Requested by

Service due	Detail	Requested by
2015-11-09	General audit	
2015-11-08	General cleanup	

Show the source of this reminder Modify source of the reminder

# Network Mapping

- when mapping relationships between connected physical/virtual hardware and the services that they support. An enterprise will use CMDBs (configuration management databases) and/or service catalogs to keep inventory of their systems. They are used to keep track of components, their purpose, software versions and the interdependencies between them.
- relationships between infrastructure components not only enables interactive visualisations of the network estate but also network tracing algorithms to walk the graph. For example, algorithms for:

**Dependency Management:** Identify single points of failure and simulate the impact of their failure on services, to identify cascading failures before they happen.

**Bottleneck Identification:** Find weak links in network routing that could cause bottlenecks at times of high network utilisation.

**Latency Evaluation:** Estimate latency across paths in the network, and the impact on services accessed from various geographic regions.



- Real-time recommendations and advertising **can quickly access and present** new recommendations or ads as a web visitor moves throughout a site.

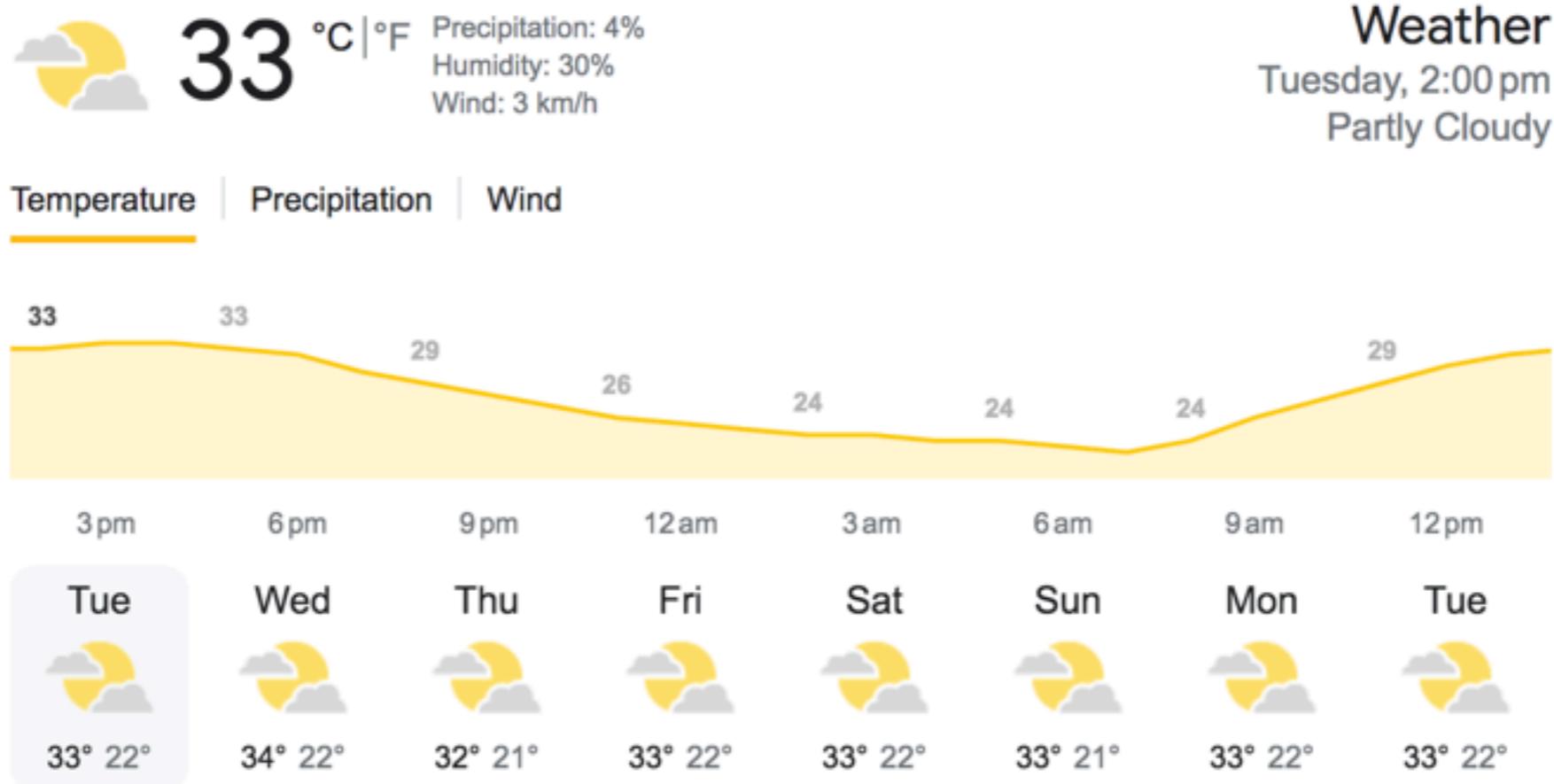
- build social relationship to enhance content personalization





- Store user session details and preference. All the information lend themselves to fast reads and writes.

# Any db vs Timeseries



## Weather tracking

- Transaction logging: Purchases, test scores, movies watched and movie latest location.
- Storing time series data (as long as you do your own aggregates).
- Tracking pretty much anything including order status, packages etc.
- Storing health tracker data.
- Weather service history.
- Internet of things status and event history.
- Telematics: IOT for cars and trucks.
- Email envelopes—not the contents.

transactions logging in Banking and Finance or events logging in web analytics or messaging systems, time-series data, tracking of inventory, IoT (Internet of Things) data, weather tracking,

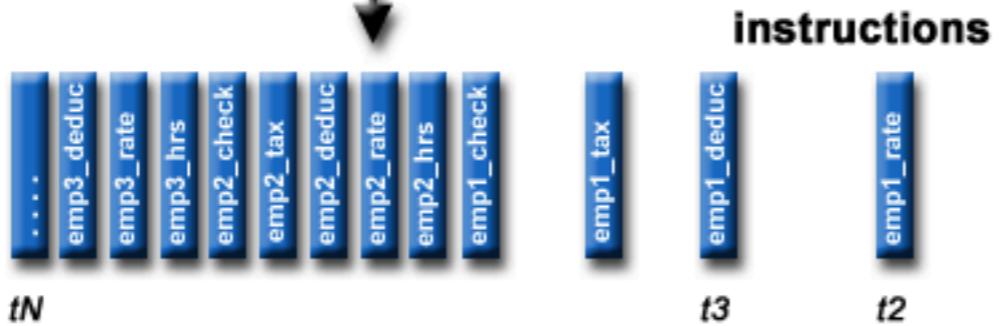
# Wide Column anti pattern

- Tables have multiple access paths. Example: lots of secondary indexes.
- The application depends on identifying rows with sequential values. MySQL autoincrement or [Oracle](#) sequences.
- Cassandra does not do ACID. LSD, Sulphuric or any other kind. If you think you need it go elsewhere. Many times people think they do need it when they don't.
- Aggregates: Cassandra does not support aggregates, if you need to do a lot of them, think another database.
- Joins: You may be able to data model yourself out of this one, but take care.
- Locks: Honestly, Cassandra does not support locking. There is a good reason for this. Don't try to implement them yourself. I have seen the end result of people trying to do locks using Cassandra and the results were not pretty.
- Updates: Cassandra is very good at writes, okay with reads. Updates and deletes are implemented as special cases of writes and that has consequences that are not immediately obvious.
- Transactions: CQL has no begin/commit transaction syntax. If you think you need it then Cassandra is a poor choice for you. Don't try to simulate it. The results won't be pretty.

# Wide column use case

- Distributed: Runs on more than one server node.
- Scale linearly: By adding nodes, not more hardware on existing nodes.
- Work globally: A cluster may be geographically distributed.
- Favor writes over reads: Writes are an order of magnitude faster than reads.
- Democratic peer to peer architecture: No master/slave.
- Favor partition tolerance and availability over consistency: Eventually consistent (see the CAP theorem: [https://en.wikipedia.org/wiki/CAP\\_theorem](https://en.wikipedia.org/wiki/CAP_theorem).)
- Support fast targeted reads by primary key: Focus on primary key reads alternative paths are very sub-optimal.
- Support data with a defined lifetime: All data in a Cassandra database has a defined lifetime no need to delete it after the lifetime expires the data goes away.

`do_payroll()`



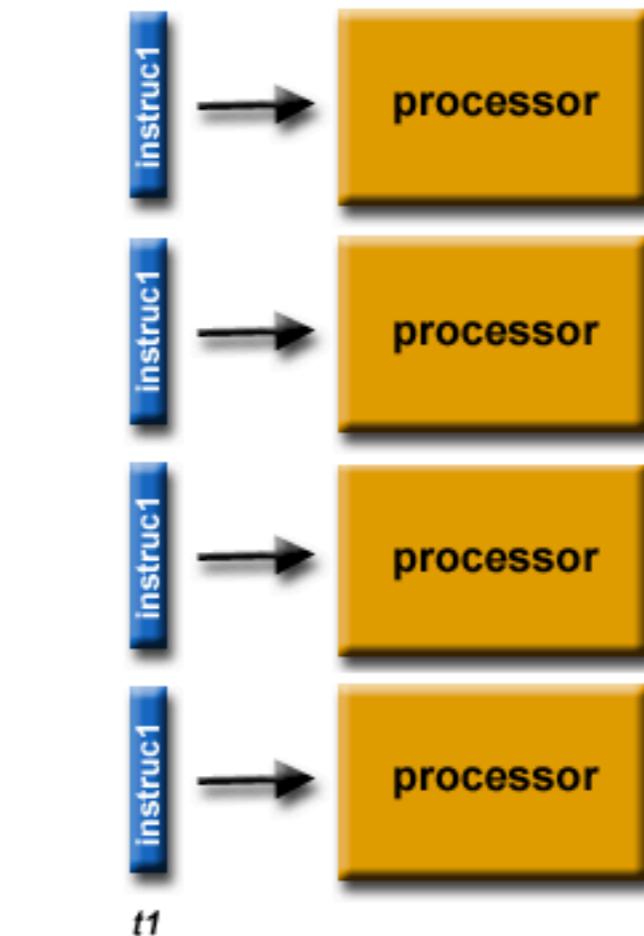
## Non-MPP

**problem**

- `do_payroll(emp1)`
- `do_payroll(emp2)`
- `do_payroll(emp3)`
- `do_payroll(empN)`



## MPP



## Row-Based Disk Storage

	Name	Industry	Age	Spines
	Percy	Construction	3	2,483
	Dylan	Construction	5	1,902
	Mary	Healthcare	4	2,190
	Dennis	Healthcare	8	1,828
	Megan	Space	5	1,231

## Column-Based Disk Storage

	#1	#2	#3	#4	#5
					
Name	Percy	Dylan	Mary	Dennis	Megan
Industry	Cons...	Cons...	Heal...	Heal...	Space
Age	3	5	4	8	5
Spines	2,483	1,902	2,190	1,828	1,231

Direction of Storage Scan

Direction of Storage Scan

You operate a dating app and need to change Employer in a John Doe's row.

## Row-Based Disk Storage

	Name	Industry	Age	Spines
	Percy	Construction	3	2,483
	Dylan	Construction	5	1,902
	Mary	Healthcare	4	2,190
	Dennis	Healthcare	8	1,828 +5 ms
	Megan	Space	5	1,231

+10 ms

10ms + 5ms = 15ms query\*

## Column-Based Disk Storage

	#1	#2	#3	#4	#5
+10 ms					
	Name	Percy	Dylan	Mary	Dennis +5 ms
+10 ms	Industry	Cons...	Cons...	Heal...	Heal... +5 ms
+10 ms	Age	3	5	4	8 +5 ms
+10 ms	Spines	2,483	1,902	2,190	1,828 +5 ms
					1,231

40ms + 20ms = 60ms query\*

access John Doe's row (including his other attributes), alter the Employer value and write.

load every Employer value for every entry, go to John Doe's index, alter it, and write the entire "column" back into data.

*you operate a financial transaction startup and need to calculate the average transaction price across billions of entries.*

## Row-Based Disk Storage

Name	Industry	Age	Spines
Percy	Construction	3	2,483
Dylan	Construction	5	1,902
Mary	Healthcare	4	2,190
Dennis	Healthcare	8	1,828
Megan	Space	5	1,231

+10 ms

+10 ms

+10 ms

+10 ms

+10 ms

## Column-Based Disk Storage

#1	#2	#3	#4	#5
----	----	----	----	----



Name	Percy	Dylan	Mary	Dennis	Megan
------	-------	-------	------	--------	-------

Industry	Cons...	Cons...	Heal...	Heal...	Space
----------	---------	---------	---------	---------	-------

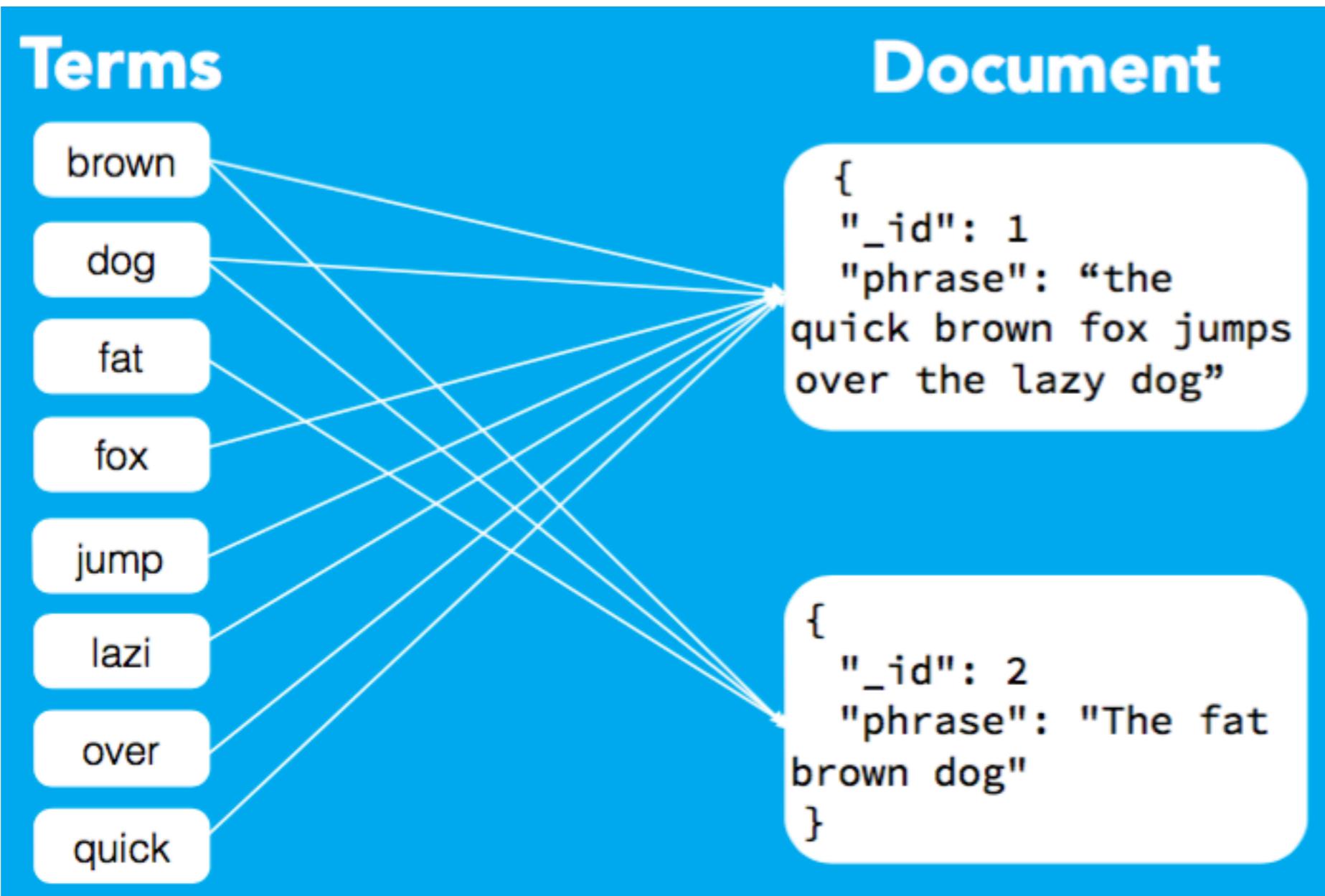
Age	3	5	4	8	+5 ms
-----	---	---	---	---	-------

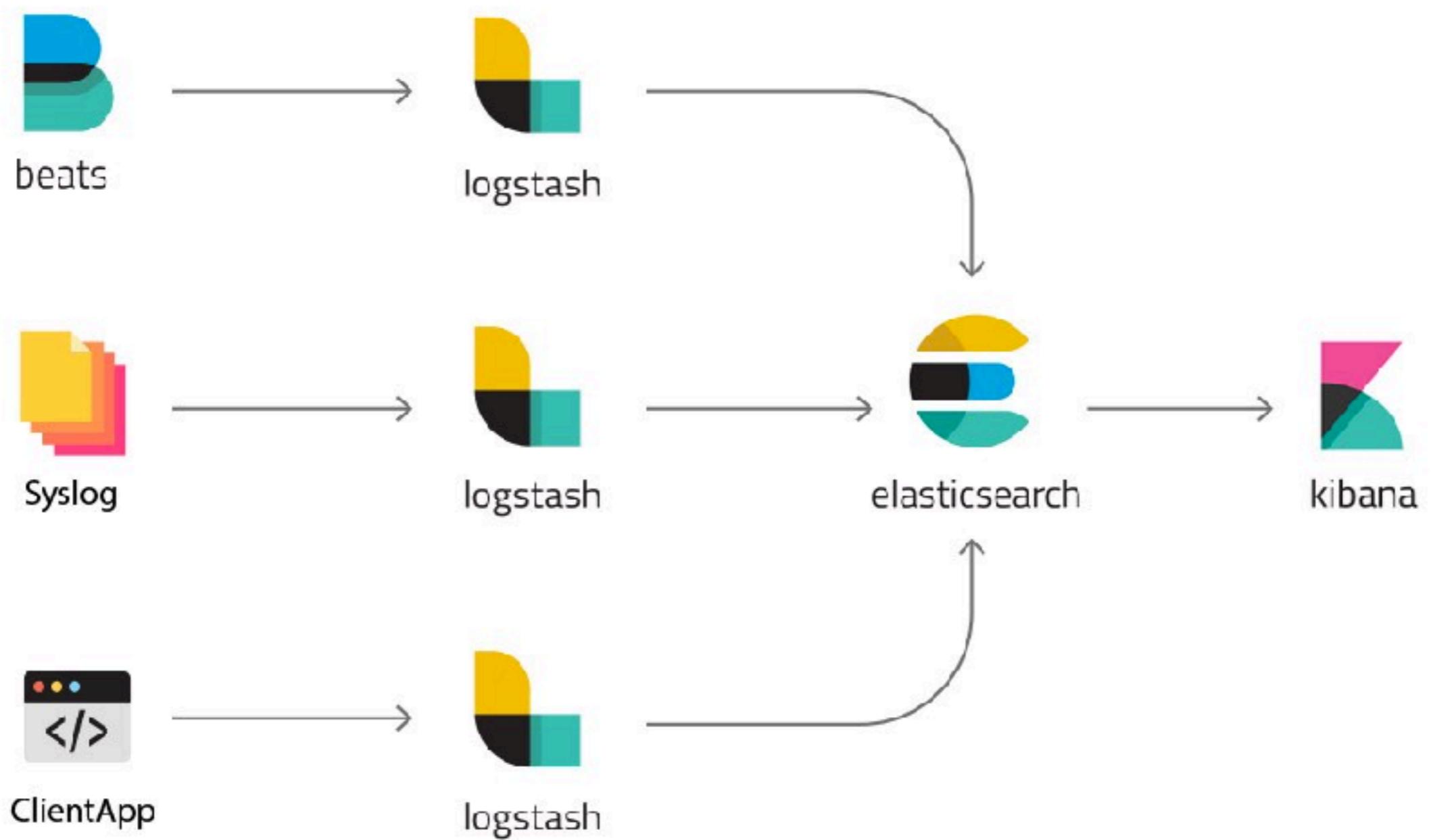
Spines	2,483	1,902	2,190	1,828	1,231
--------	-------	-------	-------	-------	-------

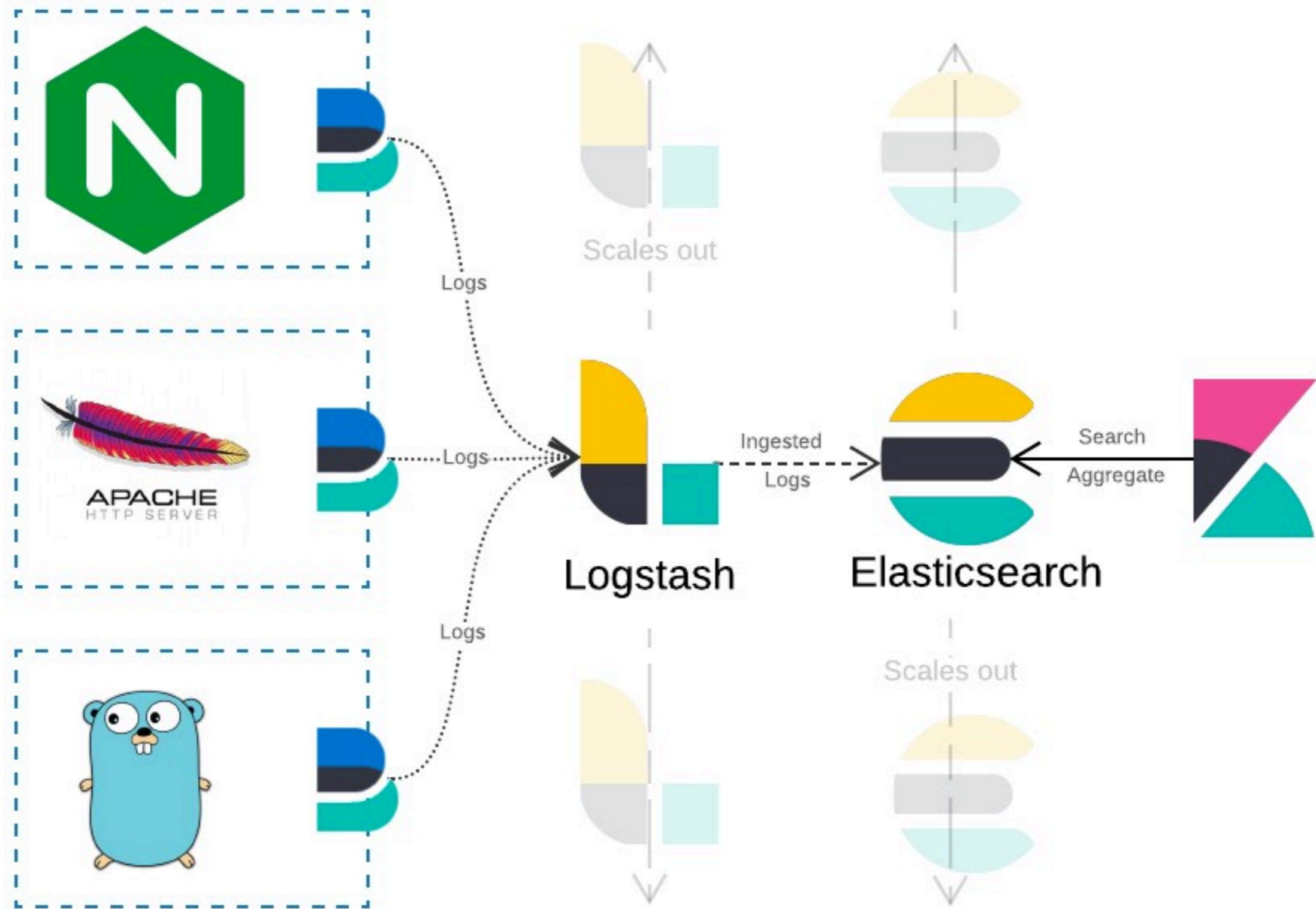
50ms + 25ms = 75ms query\*

10ms + 5ms = 15ms query\*

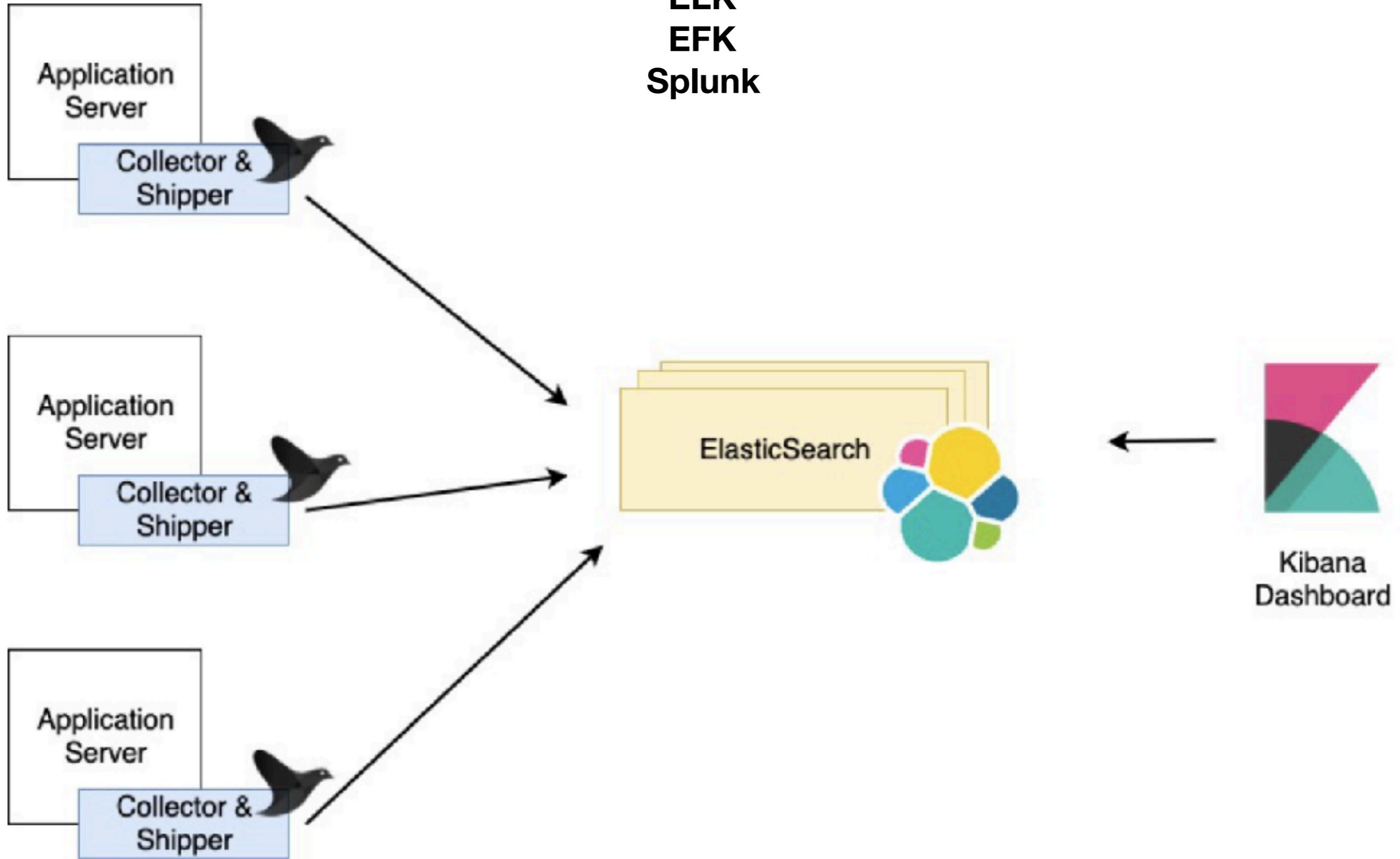
Postgres would need to incrementally retrieve every entry, grab the transaction price, add it to a running total, and return the value. ClickHouse meanwhile could calculate this, without any additional caches or optimized engines, with a single read.





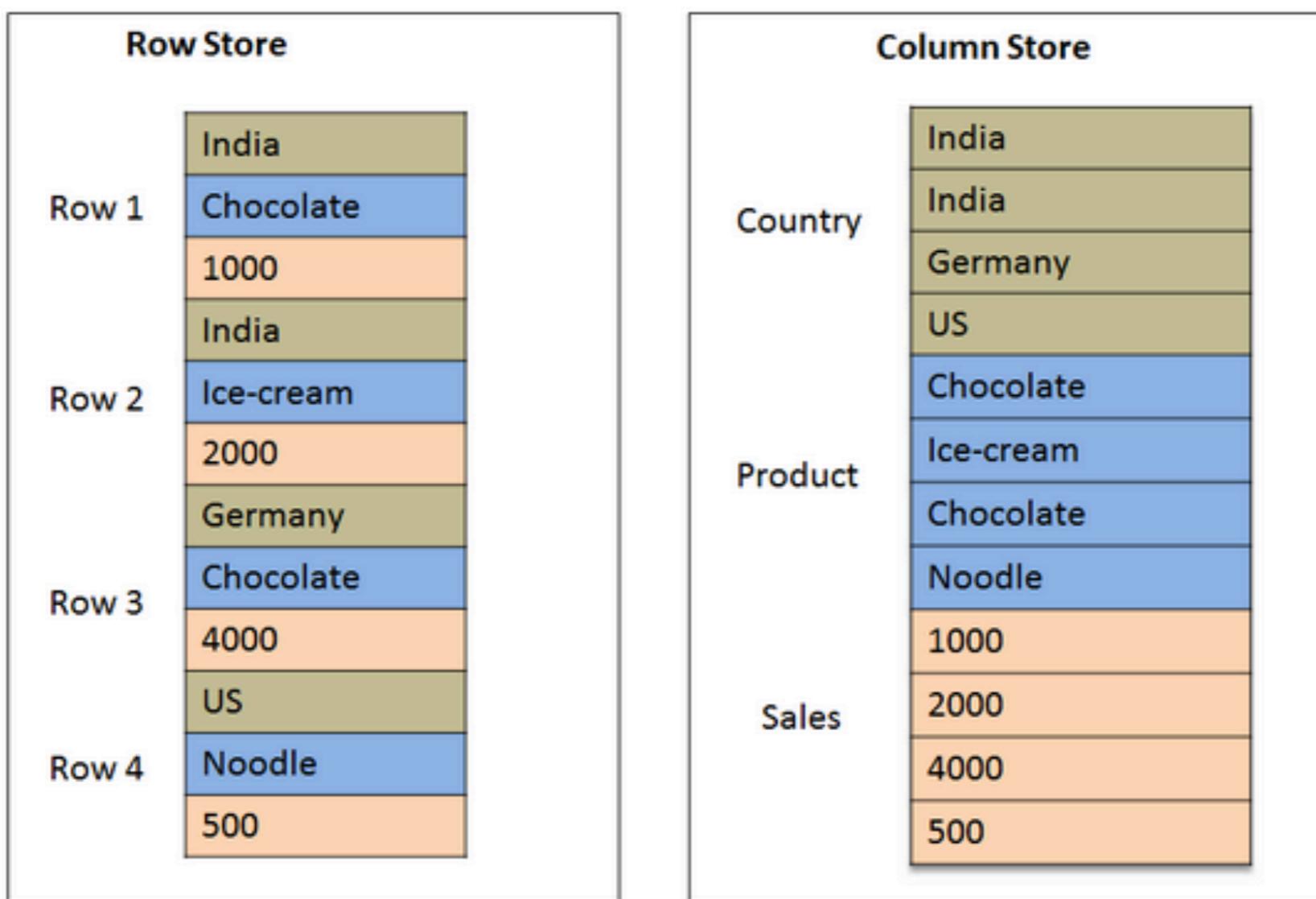


# ELK EFK Splunk



**Table**

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500



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

**Vertical slicing**

**Horizontal slicing**

rowkey1	column family (CF11)				column family (CF12)						
	column111		column112		column113		column121		column122		
rowkey1	version1111	value1111	version1121	value1121	version1121	value1131	version1211	value1211	version1221	value1221	
	version1112	value1112	version1122	value1122						version1222	value1222
			version1123	value1123							
			version1124	value1124							

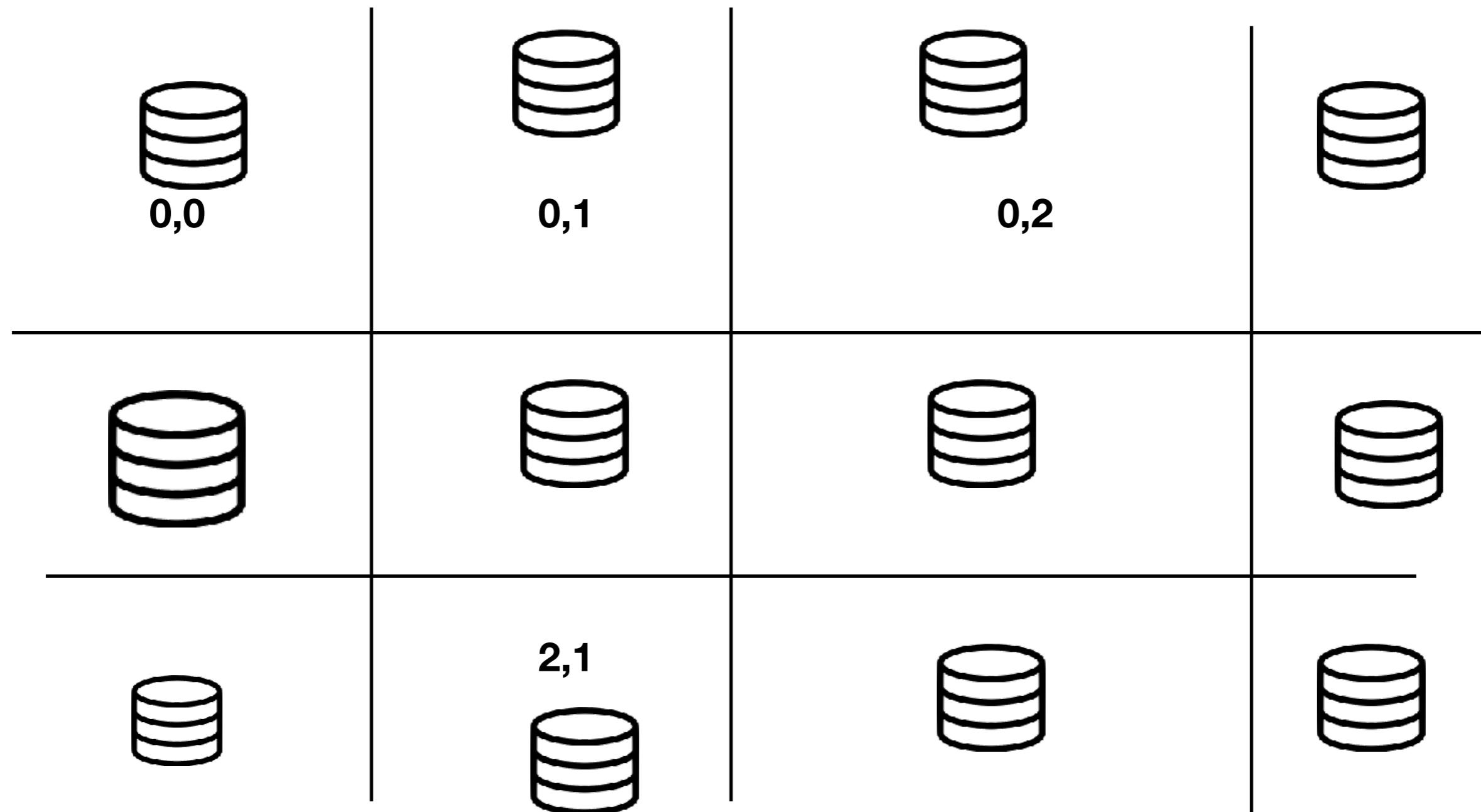
{Ordered, cust name, amount} , {itemcode, qty, price}

(Row partition id (shard key), column family id, row id)

# multidimensional map (map of maps)

```
{  
  "rowkey1": {"cf11": {"column111": {"version1111": value1111,  
                            "version1112": value1112},  
              "column112": {"version1121": value1121,  
                            "version1122": value1122,  
                            "version1123": value1123,  
                            "version1124": value1124},  
              "column113": {"version1131": value1131}  
            },  
  "cf12": {"column121": {"version1211": value1211},  
            "column122": {"version1221": value1221},  
            "version1222": value1222}  
          },  
  "rowkey2": {"cf11": {"column111": {"version2111": value2111,  
                            "version2112": value2112},  
              "column112": {"version2121": value2121,  
                            "version2122": value2122,  
                            "version2123": value2123,  
                            "version2124": value2124}  
            },  
  "cf12": {"column121": {"version2211": value2211},  
            "column122": {"version2221": value2221}  
          }  
}
```

**Row partition id,  
Col partition id**



		Column Family 1		Column Family 2		
		cf1:col-A	cf1:col-B	cf2:col-Foo	cf2:col-XYZ	cf2:foobar
Region 1	row-1					
	row-10					
	row-18	A18 - v1 ▼	B18 - v3 ▼	Foo18 - v1 ▼	XYZ18 - v2 ▼	foobar18 - v1 ▼
Region 2	row-2					
	row-5					
	row-6					
Region 3	row-7					
	row-8					

Physical Coordinates for a Cell: *Region Directory → Column Family Directory  
→ Row Key → Column Family Name → Column Qualifier → Version*

	CF1:colA	CF1:colB	CF1:colC
Row1	<p>@time7: value3</p>		
Row10	<p>@time2: value1</p>	<p>@time2: value1</p>	
Row11	<p>@time6: value2</p>		
Row2	<p>@time4: value1</p>		<p>@time4: value1</p>

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

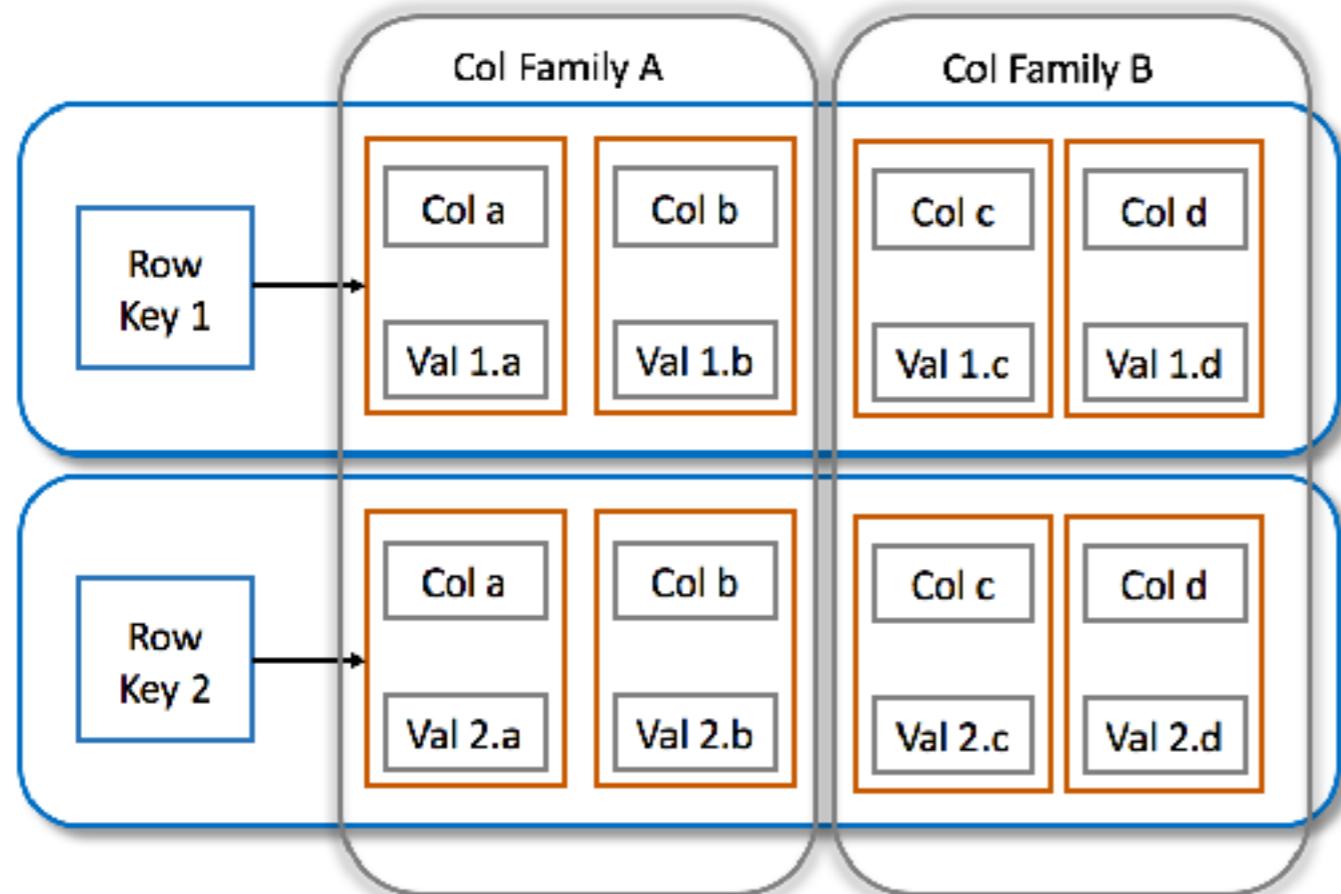
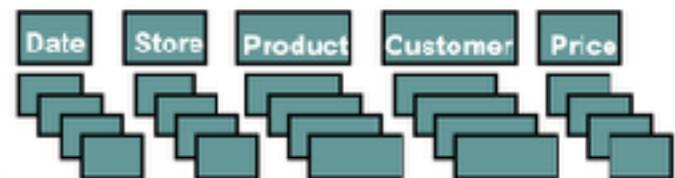
Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

row-store



column-store



column-family

Columnar

sparse data model

multi-dimensional map

column-families are not independently accessible.

every column is stored separately

NoSQL

SQL interface

Reads that use the partition key are incredibly fast

optimized for read-mostly analytical workloads

high throughput writes

very slow writes

data warehouses

	MongoDB	Cassandra
Expressive object model	Yes	No
Secondary indexes	Yes	No
No downtime on node failure	No	Yes
High write throughput	No	Yes

- Cassandra is optimized for really high throughput on writes. **If your use case is read-heavy (like cache) then Cassandra might not be an ideal choice.**
- It does not support complete transaction management across the tables. Not ACID compliant system.
- Secondary Index not supported. Have to rely on Elastic search /Solr for Secondary index and the custom sync component has to be written.

not possible to aggregate data in a query. Sums and averages like information have to be done by the client-side application.

Use Cassandra as the primary data store for capturing information as it arrives into the system; but then build “query-optimised views” of subsets of that data in other databases specialised to the kinds of access users require. Use an indexing engine such as SOLR to provide full-text search across records, or a graph database such as Neo4j to store metadata in a way that supports “traversal-heavy” queries that join together many different kinds of entity. Use an RDBMS when you need the full power and flexibility of SQL to express ad hoc queries that explore the data in multiple dimensions.

### Custom Reports



### Data LakeHouse



### Data warehouse



GreenPlum



Catalogue

Access Control

### Query Engine



Hive

Flink

### Table Format



### Data Format

Apache Avro, Apache ORC

CSV

Json



# on prem (   
 # On cloud k8s (aws,azure)   
 # SAAS

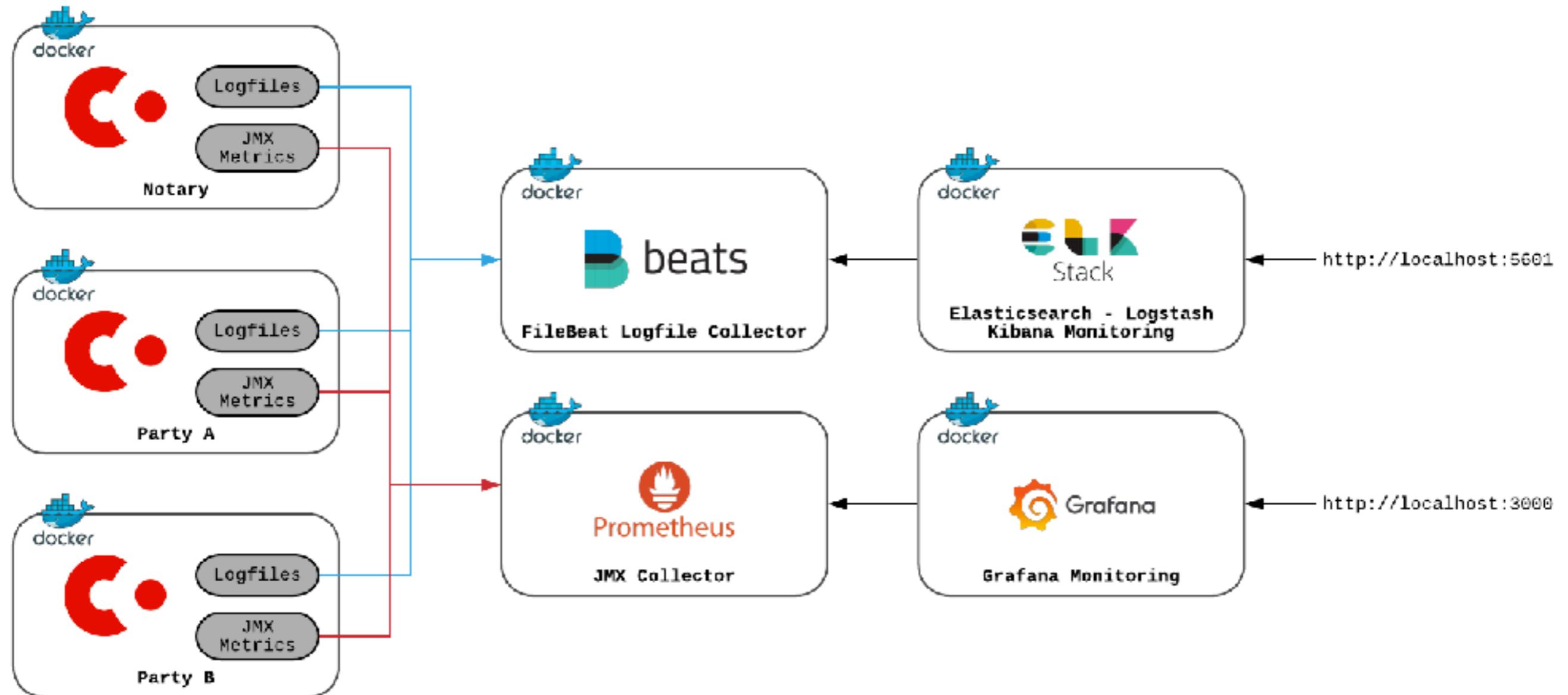
Azure/AWS vm's

### Storage



Skan.

© SKAN.AI 2019 CONFIDE



## 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"/>

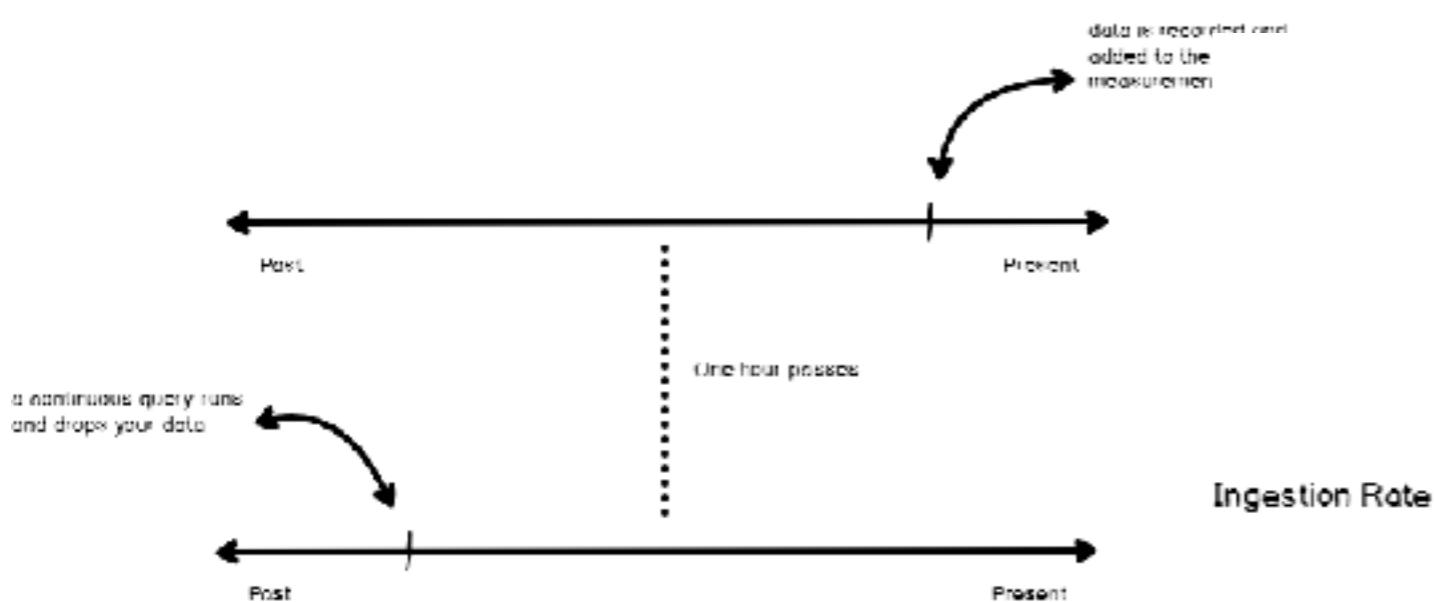
Data are multidimensional

## Time Series Databases

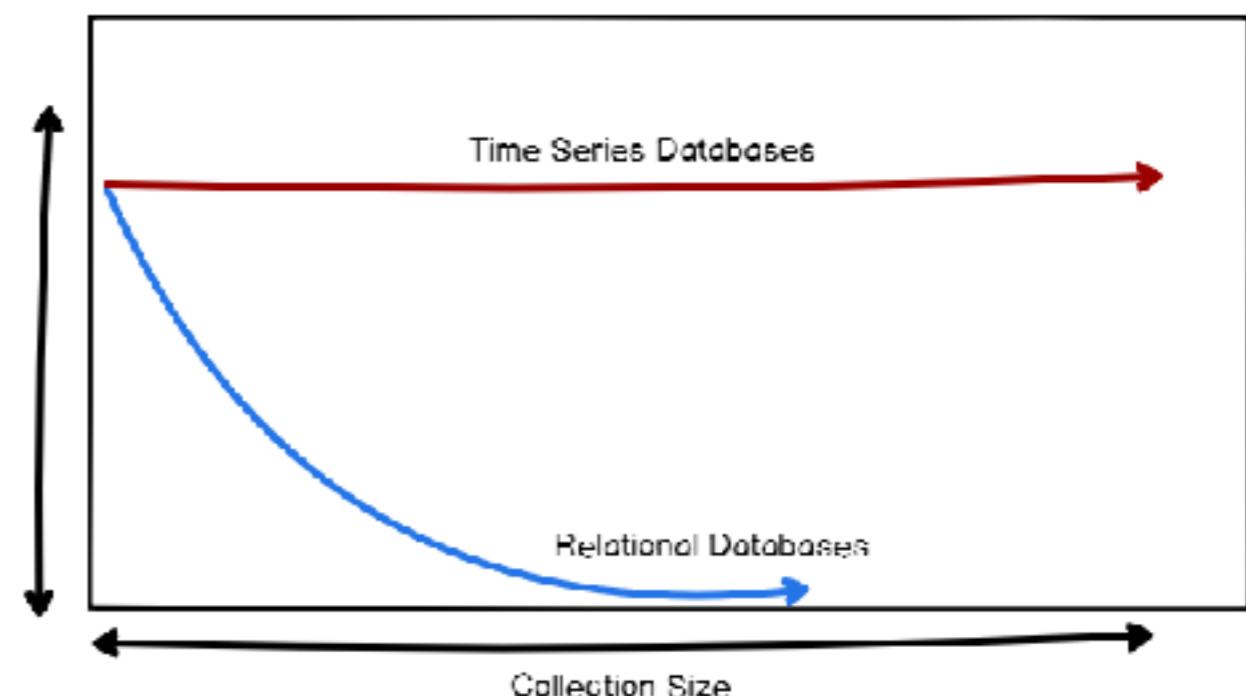
Sensor Temperature	Time
39.6	12/01/19 @ 11:12
11.2	12/01/19 @ 11:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

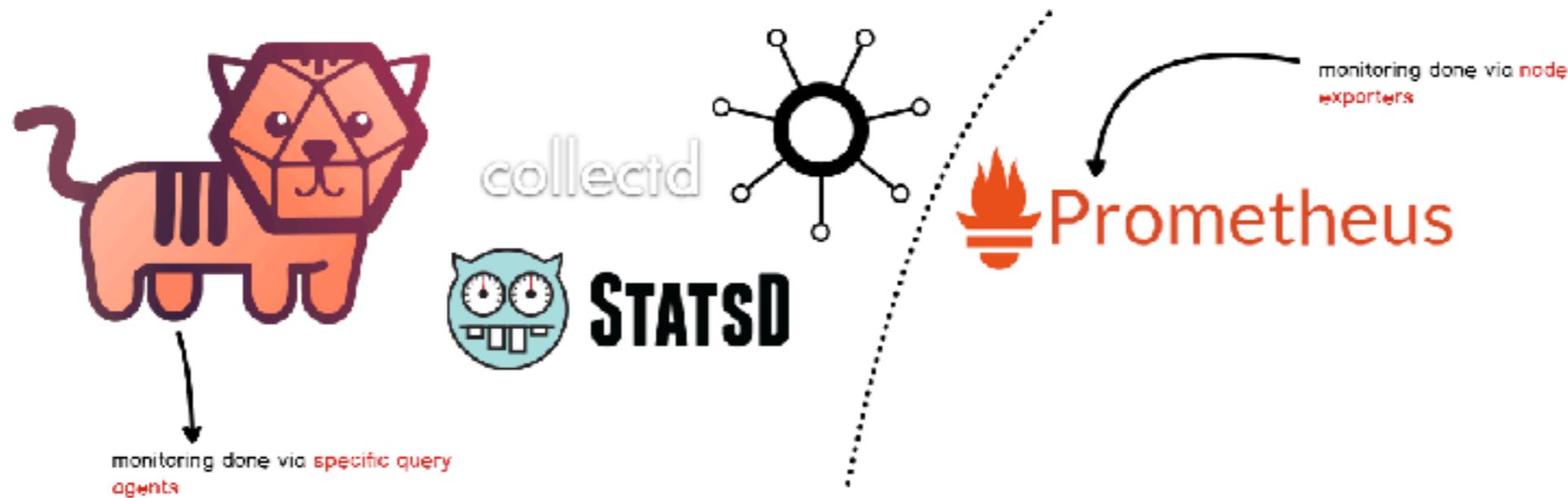
Case : retention policy = 1 hour



## DBMS & TSDB Difference



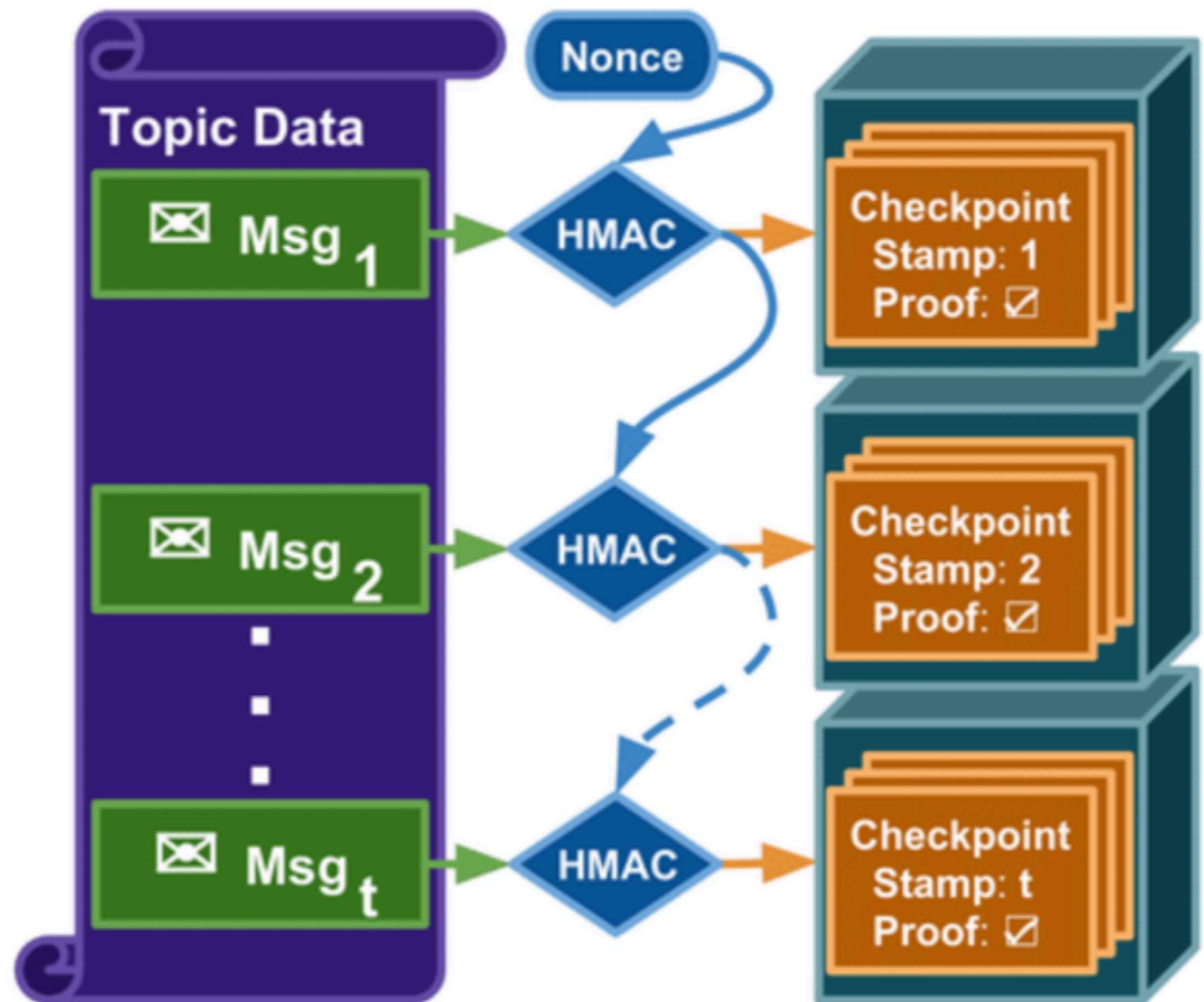
## Tools that 'produce' TSDB data



## Tools that 'consume' TSDB



\*Non-exhaustive list

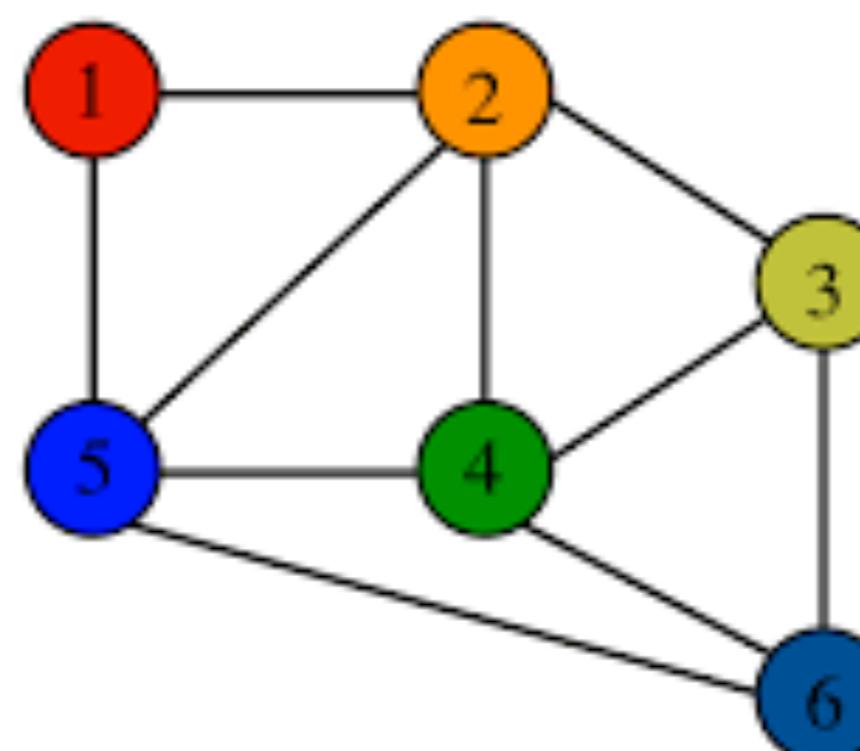
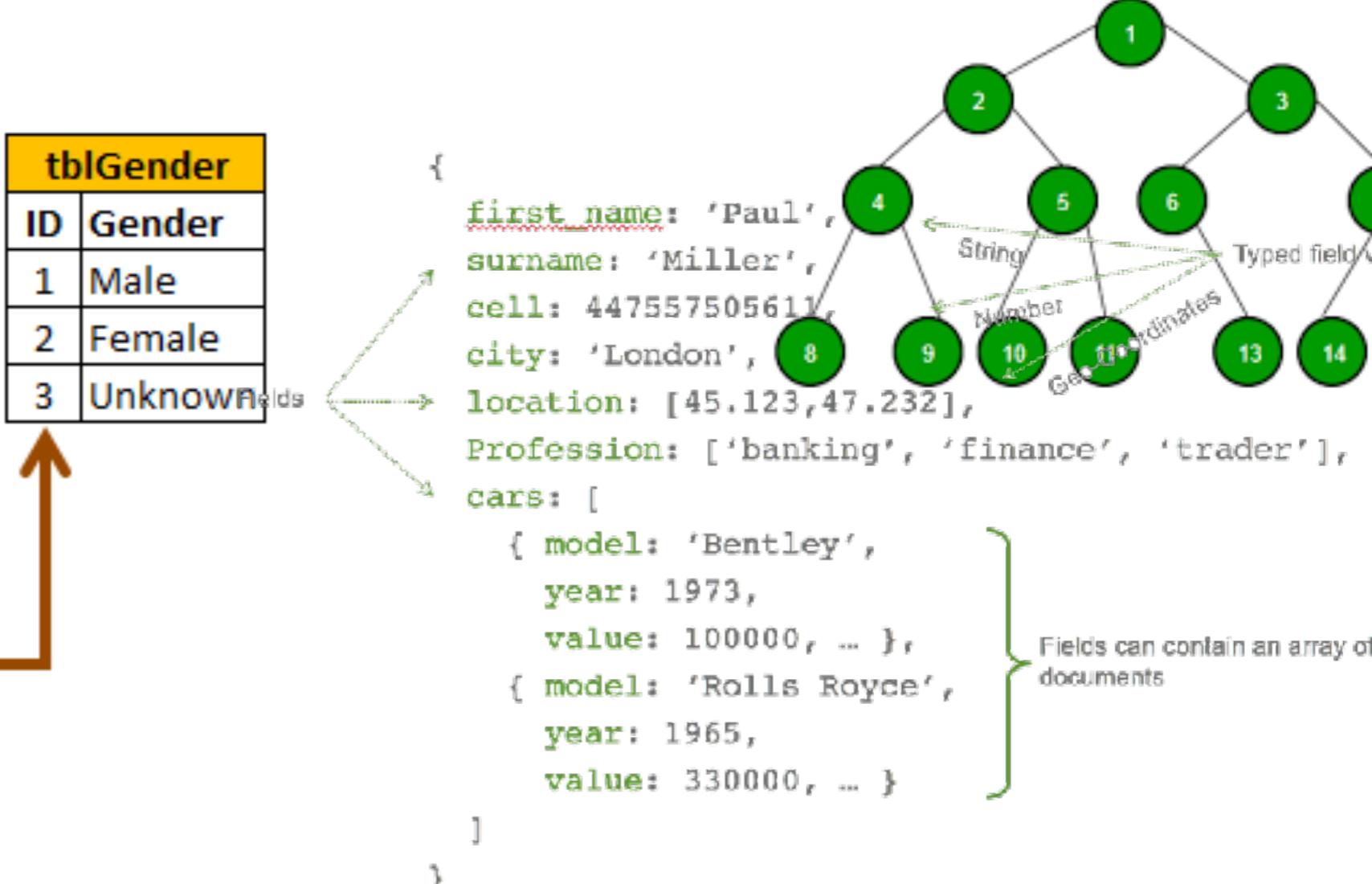
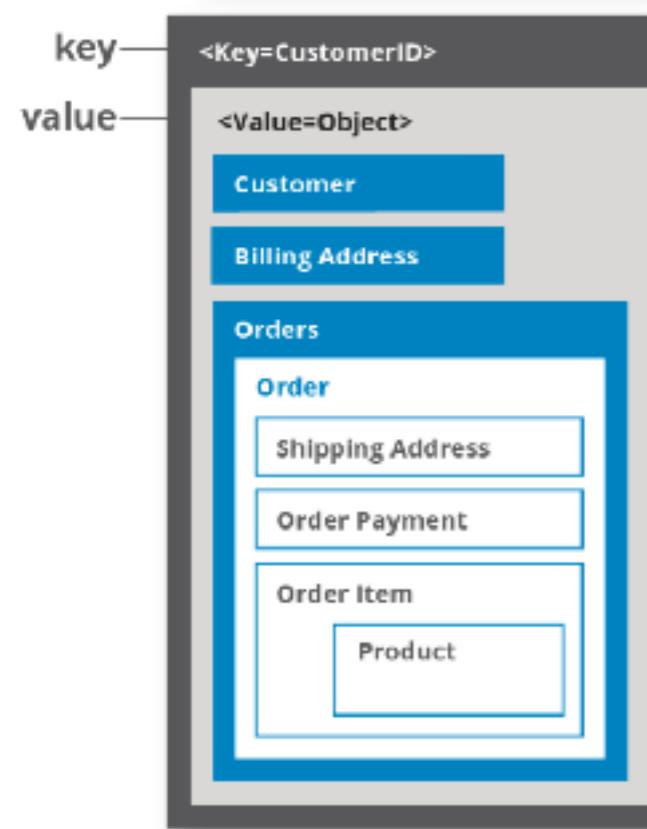


# Rdbms ( Referential Integrity)

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

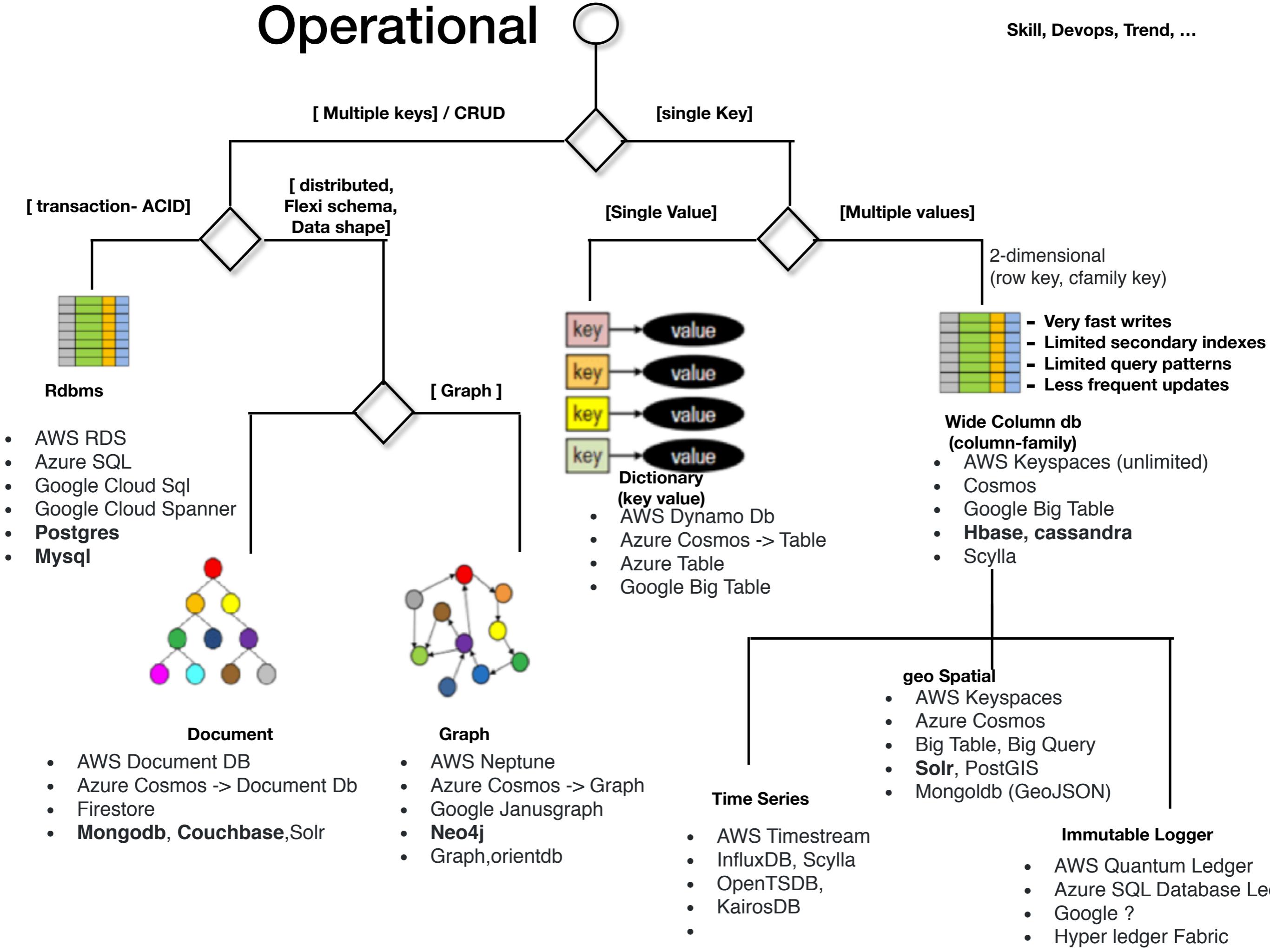


## Wide Column

- 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

# Operational

Skill, Devops, Trend, ...



# Analytical



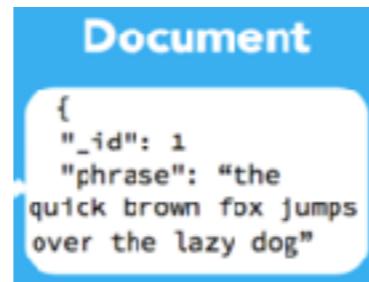
rowkey1	column family (CF11)				column family (CF12)			
	column111	column112	column113	column121	column122			
	version1111	value1111	version1121	value1121	version1121	value1131	version1211	value1211
	version1112	value1112	version1122	value1122				
			version1123	value1123				
			version1124	value1124				

Read  
Columnar db  
(column database)  
Dimensional modeling  
Cloud DWH (OLAP)

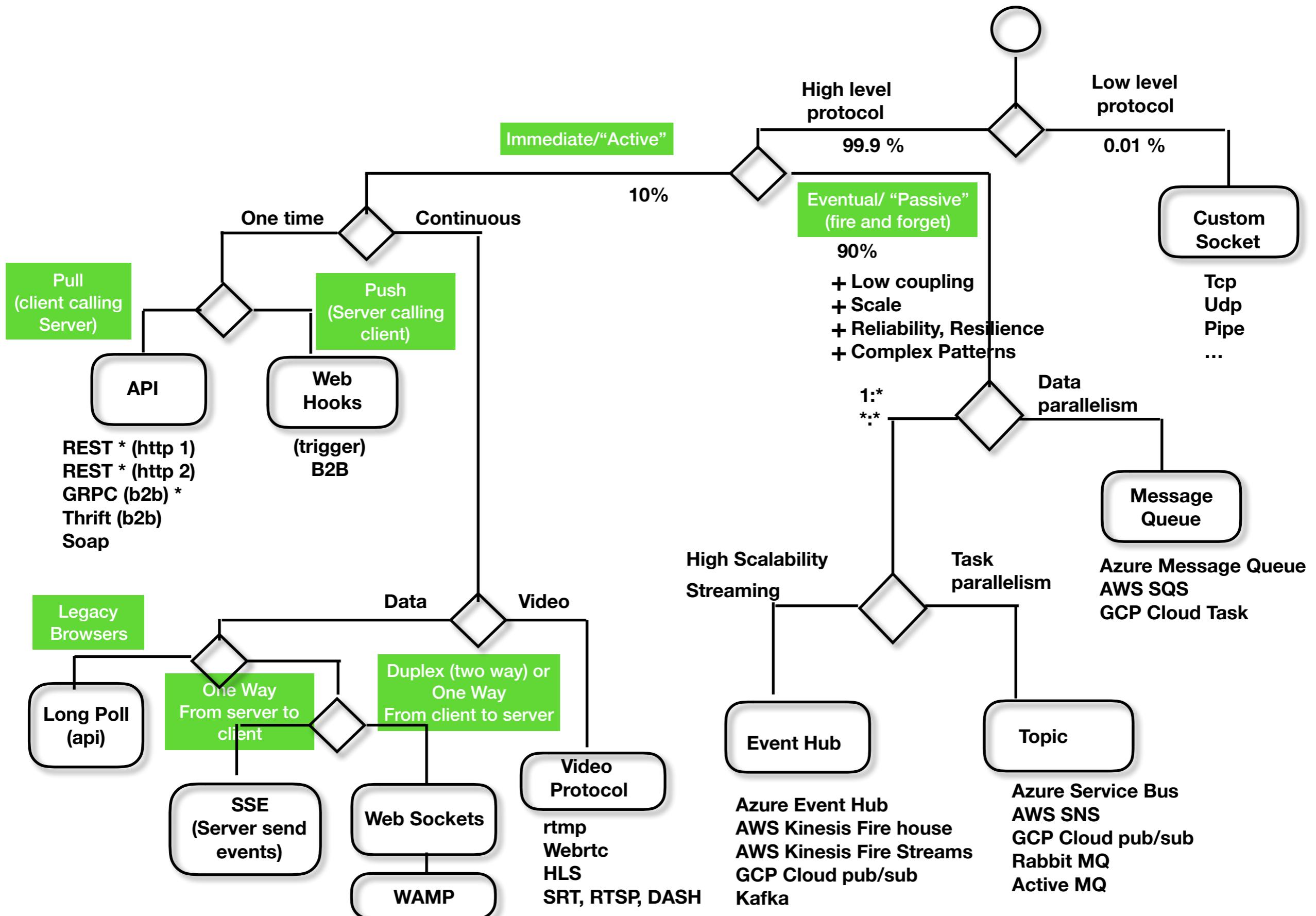
- AWS Redshift (max: 8 PB)
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house
- Greenplum
- Kylin

Read  
Text Database

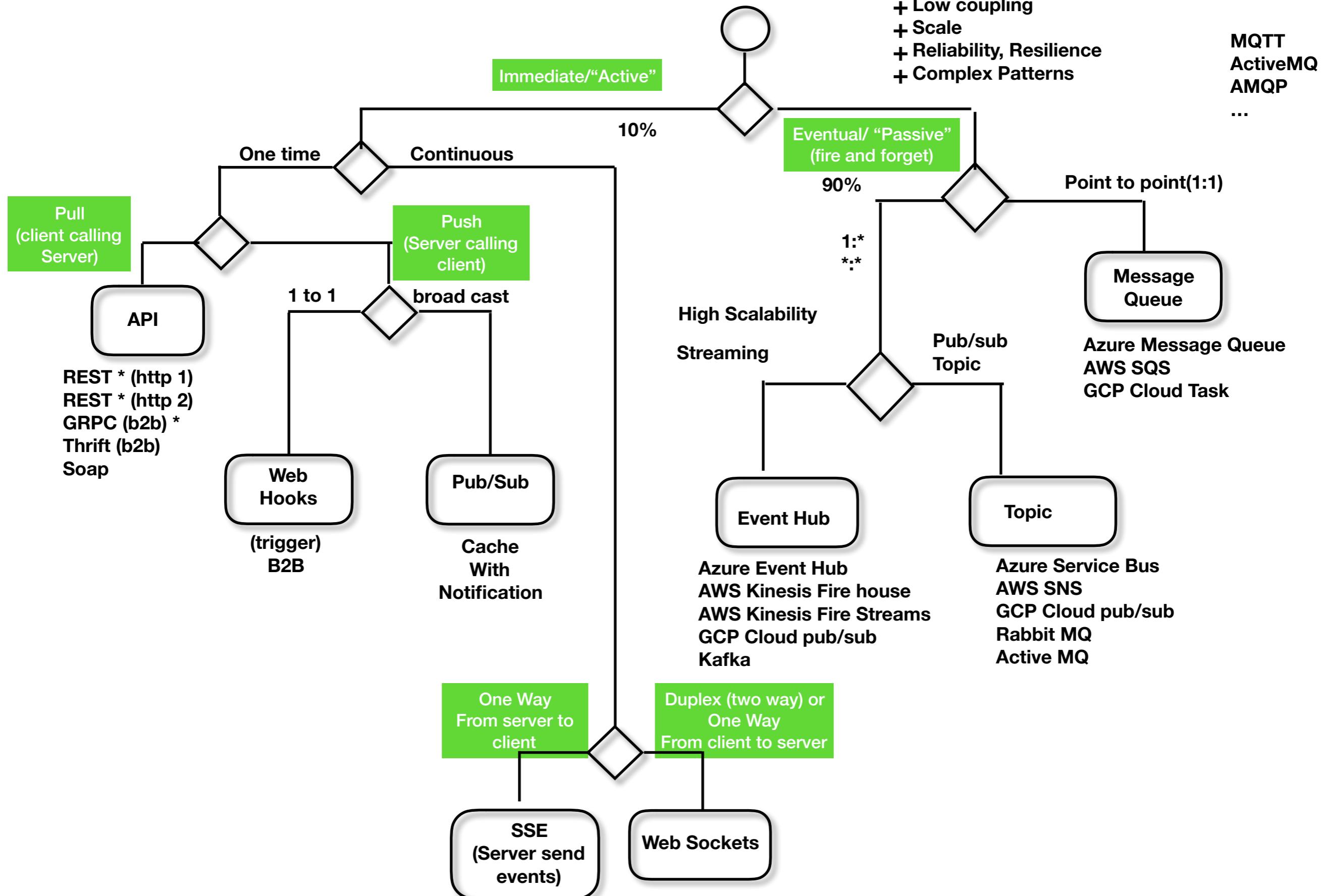
- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, **Solr**



# Choose Communication



# Choose Communication protocol



## Webhooks



"New data!"



## APIs



"Send me data"

"Here you go!"



**zapier**

# Backfilling & reconsuming

*re-producing messages instead of re-consuming messages.*

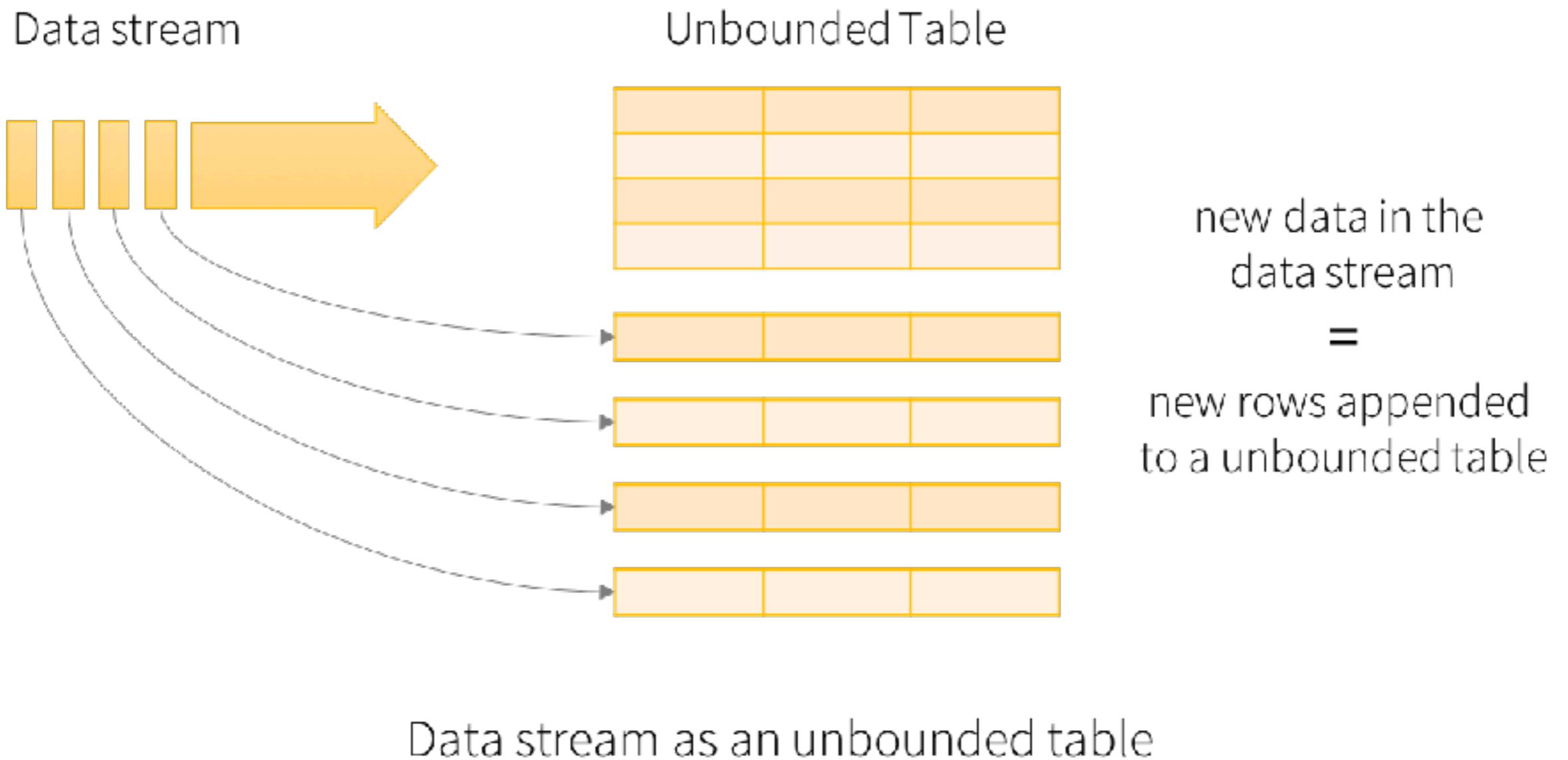
Backfilling is a good fit when only a limited number of entities need updating.

## Different approaches to reconsuming

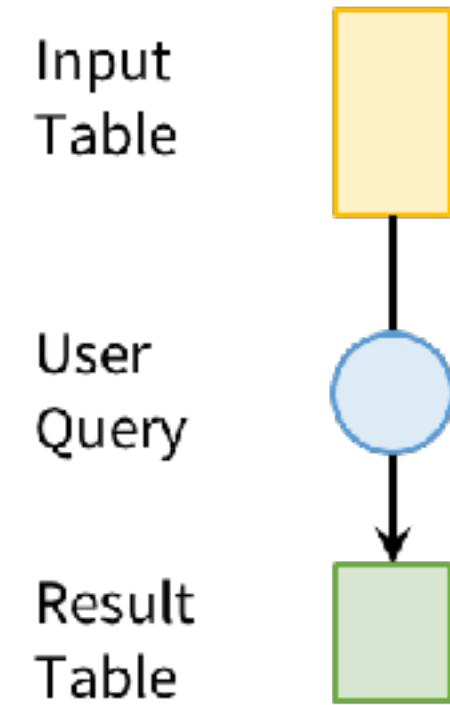
- Letting the existing consumer group reconsume the historical messages by setting its offset into the past
- Parallel processing by existing and temporary consumer groups until the temporary consumer has caught up
- Dynamic processing with a handover from the temporary consumer group to the existing consumer group

# Streaming

Streaming represents a stream of data as a table that is unbounded in depth,

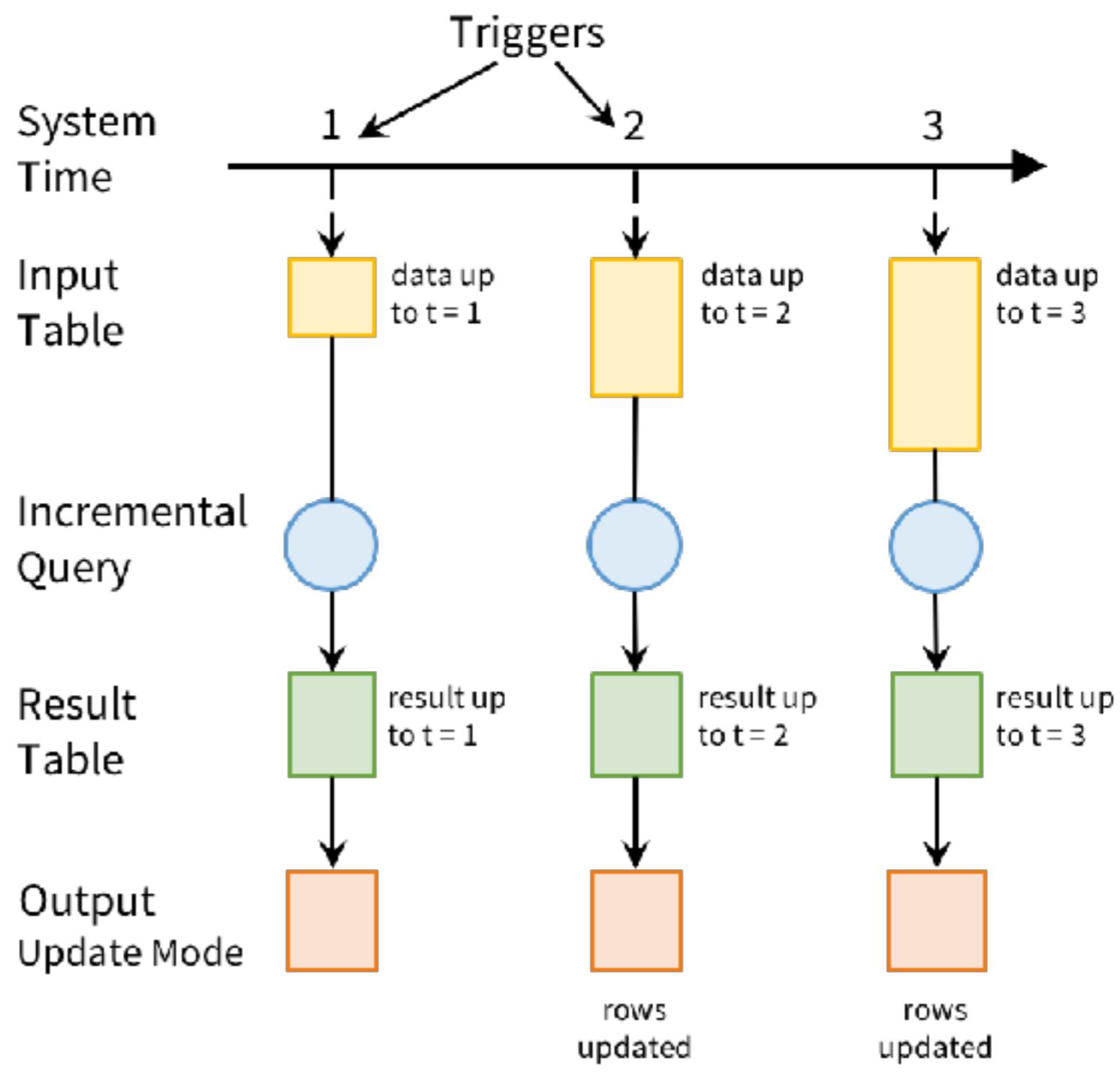


- Consider the input data stream as the “Input Table”. Every data item that is arriving on the stream is like a new row being appended to the Input Table.



Spark SQL  
Planner

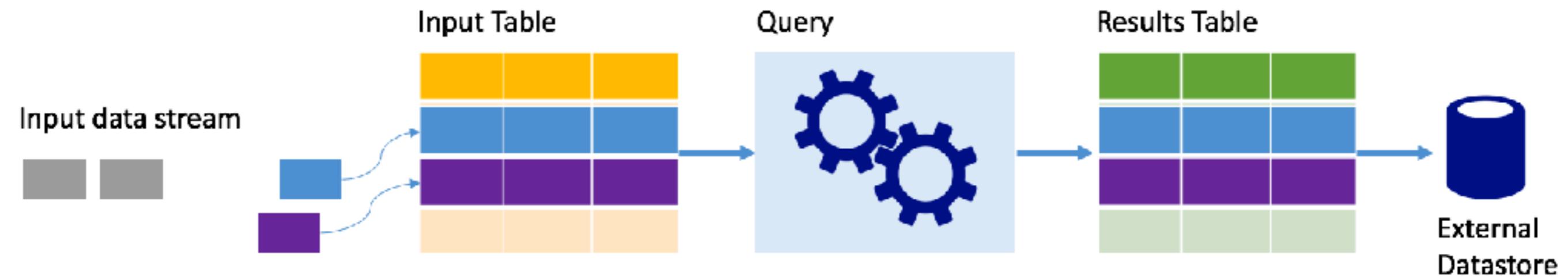
User's batch-like  
query on input table



Incremental execution on streaming data

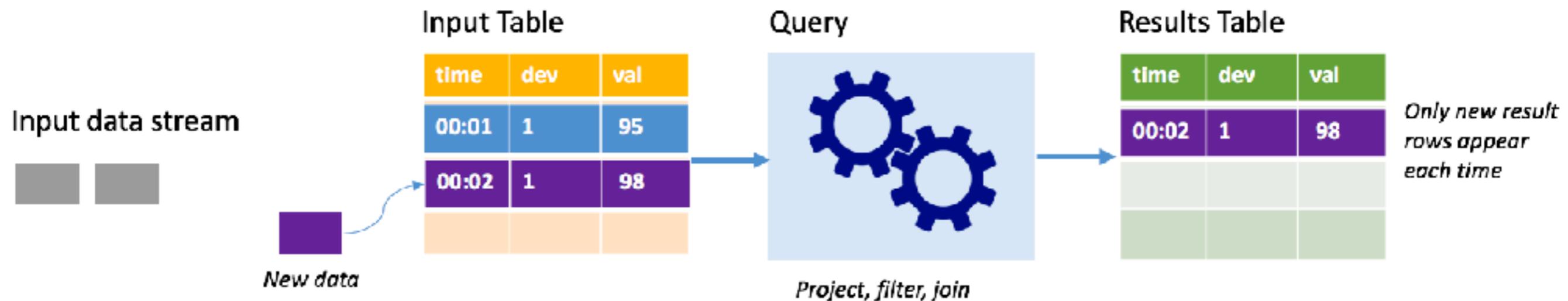
## Structured Streaming Processing Model

Users express queries using a batch API; Spark incrementally runs them on streams

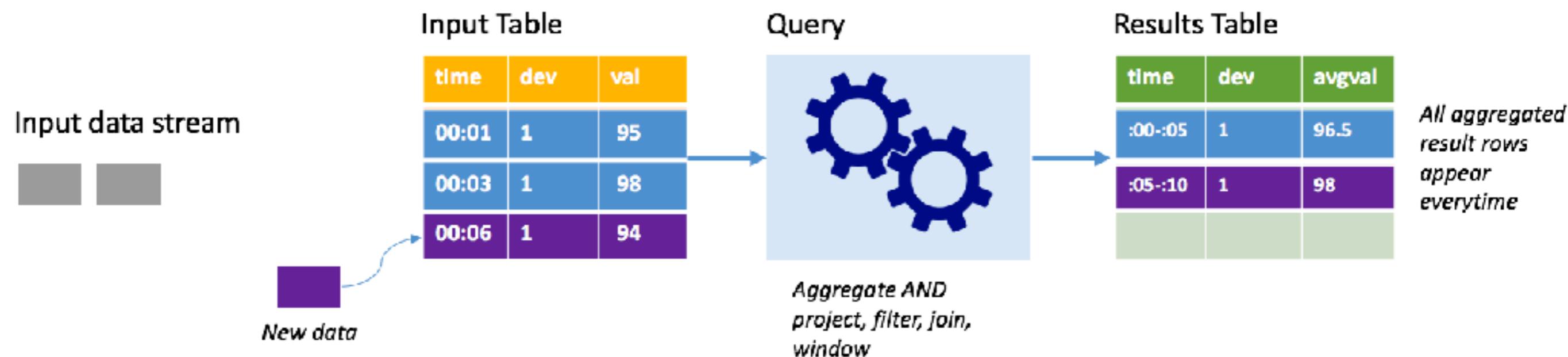


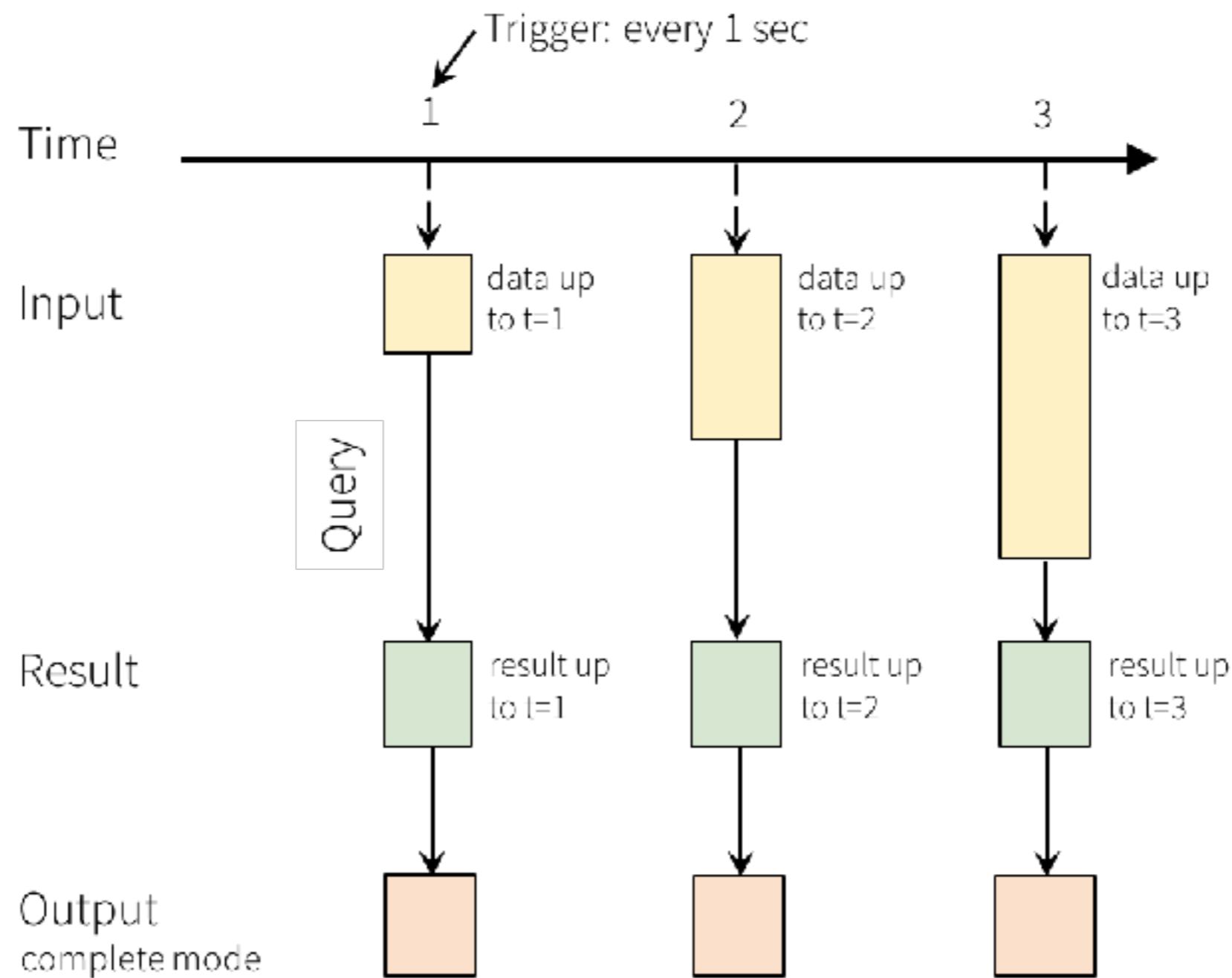
In Structured Streaming, data arrives at the system and is immediately ingested into an input table. You write queries (using the DataFrame and Dataset APIs) that perform operations against this input table. The query output yields another table, the *results table*. The timing of when data is processed from the input table is controlled by the *trigger interval*. By default, the trigger interval is zero, so Structured Streaming tries to process the data as soon as it arrives. In practice, this means that as soon as Structured Streaming is done processing the run of the previous query, it starts another processing run against any newly received data. You can configure the trigger to run at an interval, so that the streaming data is processed in time-based batches.

## Append mode



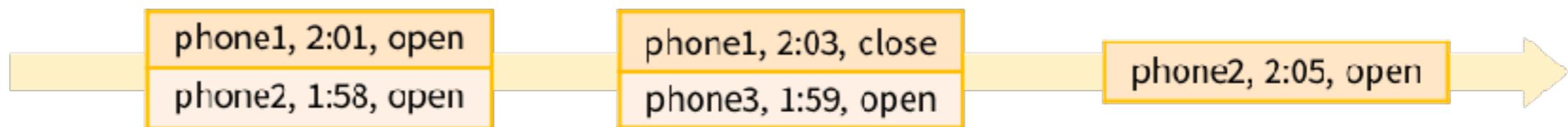
## Complete mode





Programming Model for Structured Streaming

## Arriving Records



## System Time

2:02

2:04

2:06

## Input Table

phone1, 2:01, open
phone2, 1:58, open

data up  
to 2:02

phone1, 2:01, open
phone2, 1:58, open
phone1, 2:03, close
phone3, 1:59, open

data up  
to 2:04

phone1, 2:01, open
phone2, 1:58, open
phone1, 2:03, close
phone3, 1:59, open
phone2, 2:05, close

data up  
to 2:06

## Query



state



state



## Result Table

open	2:00	1
open	1:00	1

result up  
to 2:02

open	2:00	1
open	1:00	2
close	2:00	1

result up  
to 2:04

open	2:00	2
open	1:00	2
close	2:00	1

result up  
to 2:06

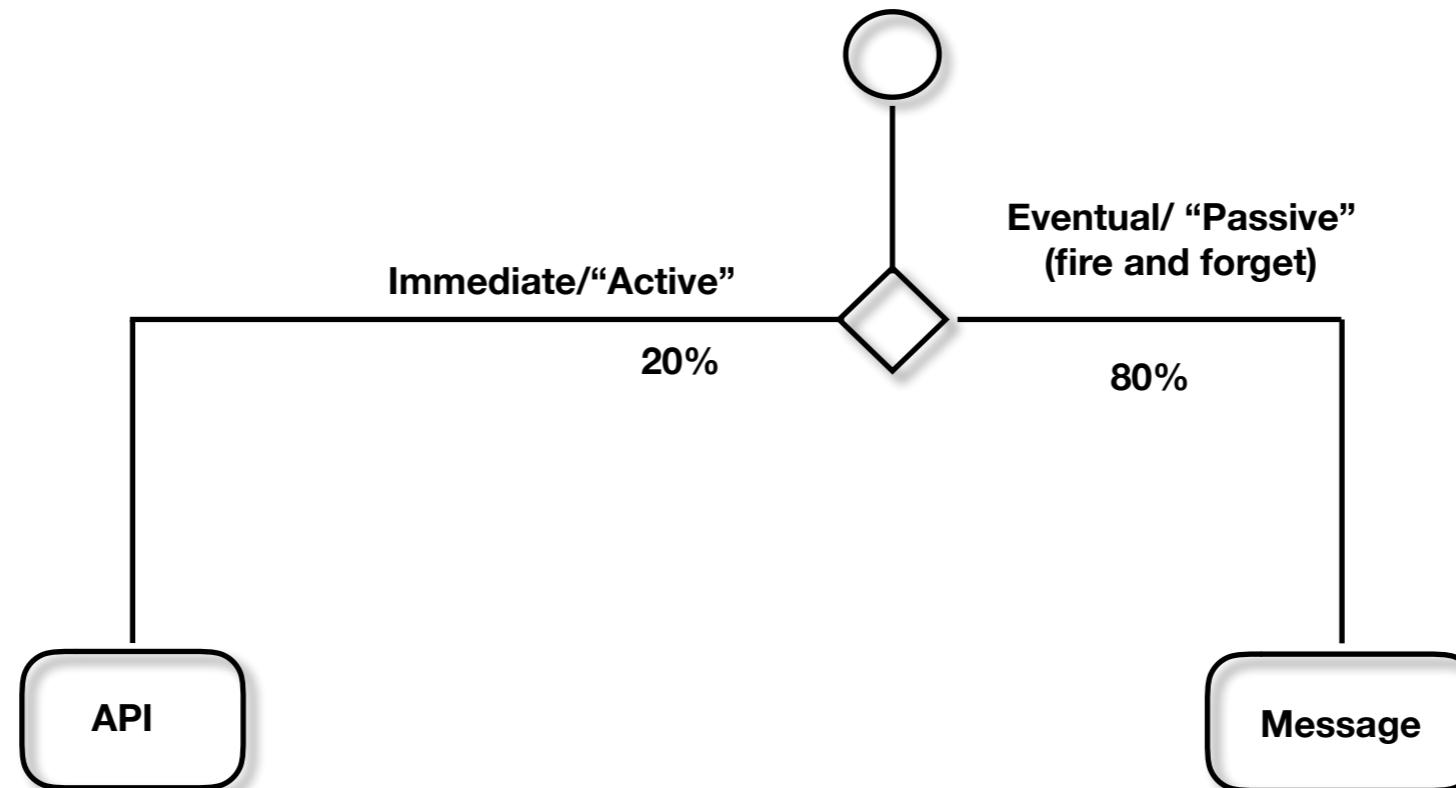
## Output Update Mode

open	2:00	1
open	1:00	1

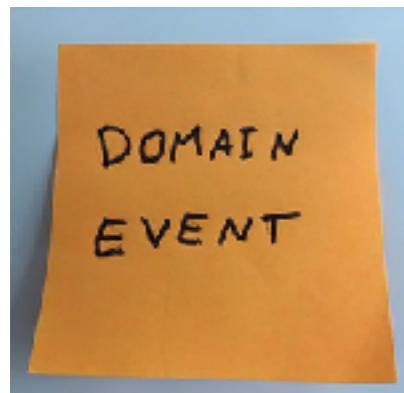
open	1:00	2
close	2:00	1

open	2:00	2
------	------	---

# Choose Communication protocol



# Event storming



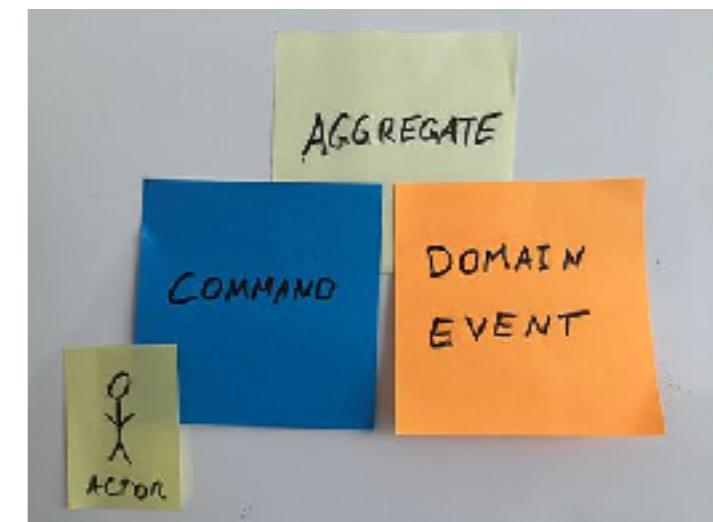
Step 1: Create domain events



Step 2: Add the commands that caused the domain event



Step 2b: Add the actor that executes the command

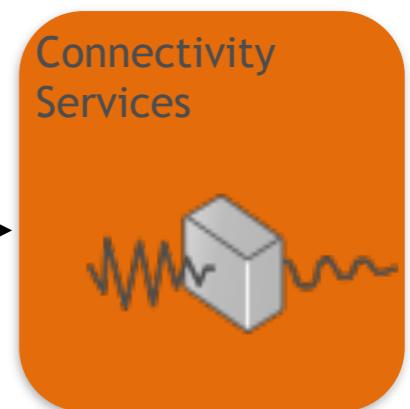


Step 3: Add corresponding aggregate



Communication  
Services

Connected protocol  
(REST, WS, GRPC , Thrift, ...)

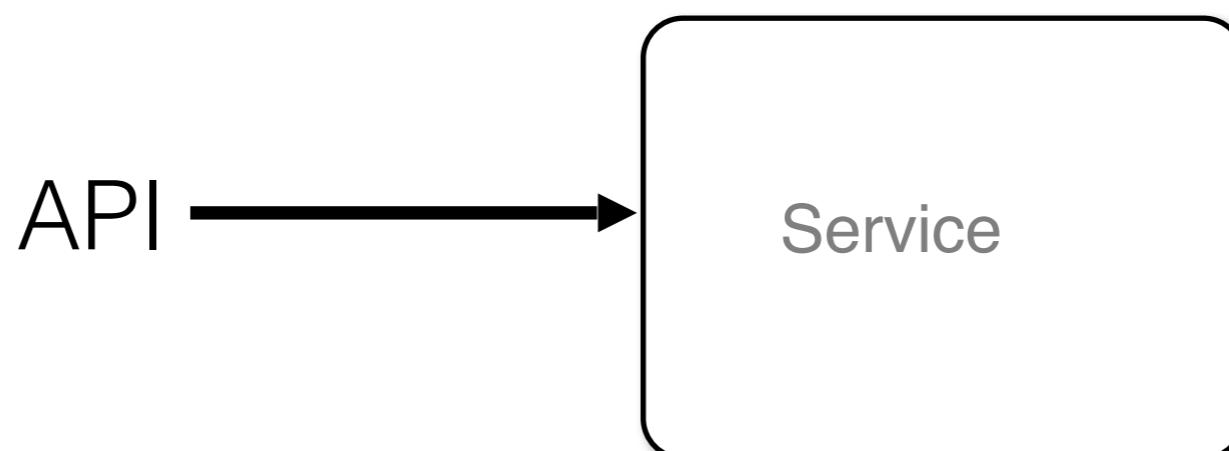


Connectivity  
Services

Message protocol  
(AMQP, MQTT, ...)

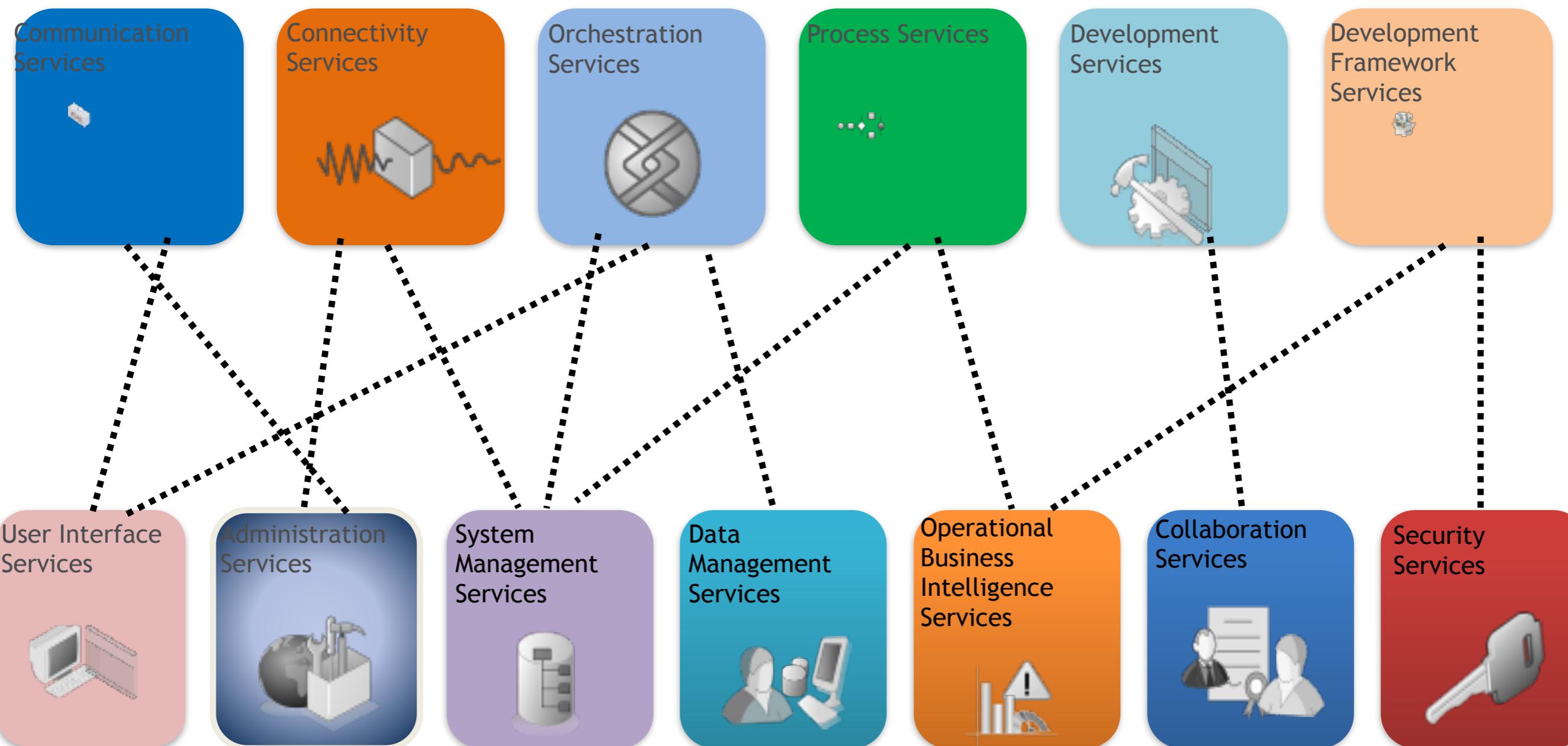


	Connected	Msg database (ACID)
Scale	--	++
Coupling	--	++
Reliability	--	++ (retry)
Developer effort		
Consistency	Immediate	Eventual - - (SAGA)
Debugging	+	-
Delivery Order	Ordered ++	Unordered * - -
Duplicate	--	--
Dev Ops Effort		
Exception Handling	++ (error response)	- - ?
Result	Response	? (one way)
Infra structure cost	--	++



The diagram illustrates the relationship between an API and a Service, with the Service containing a large empty area for notes.

	<b>REST</b>	<b>WSS</b>	<b>GRPC</b>
<b>Browser Support</b>	Y	Y	N
<b>Format</b>	TEXT (HTTP 1.x)	TEXT (HTTP 1.x)	BINARY, HTTP2
<b>Serialization</b>	JSON	JSON / XML / Any	Proto Buf
<b>Performance</b>	--	+	++
<b>Duplex communication</b>	Pull	Pull & Push	Pull & Push
<b>Use case</b>	North South	Streaming & North South	East West



Communication Services



Connectivity Services



Orchestration Services



Process Services



Development Services



Development Framework Services



User Interface Services



Administration Services



System Management Services



Data Management Services



Operational Business Intelligence Services

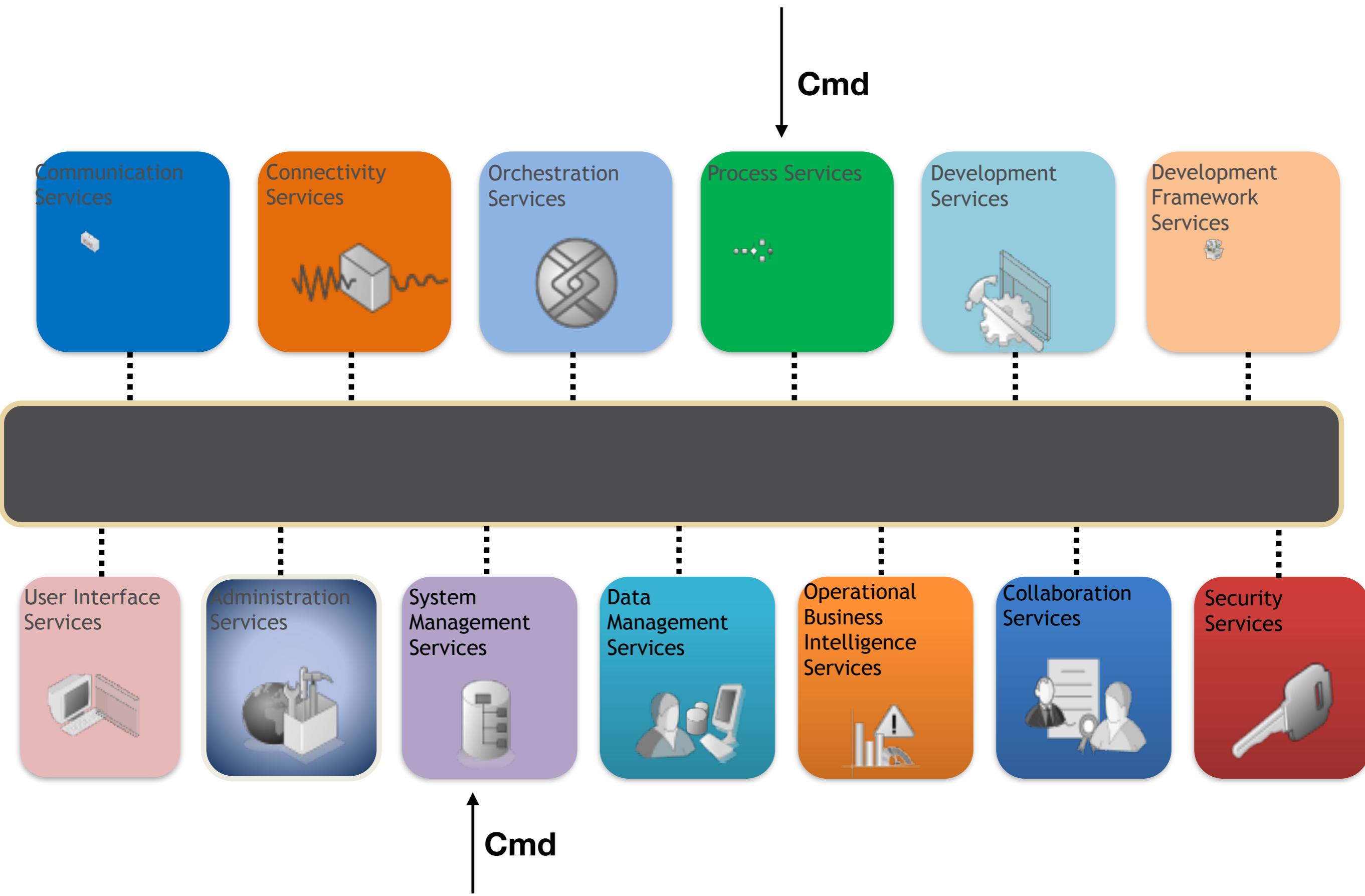


Collaboration Services

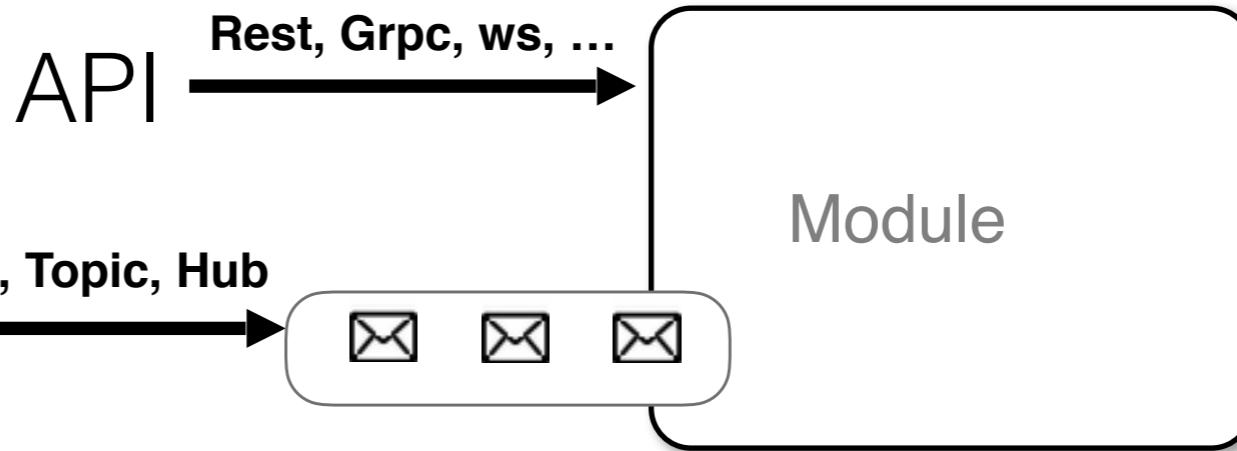


Security Services



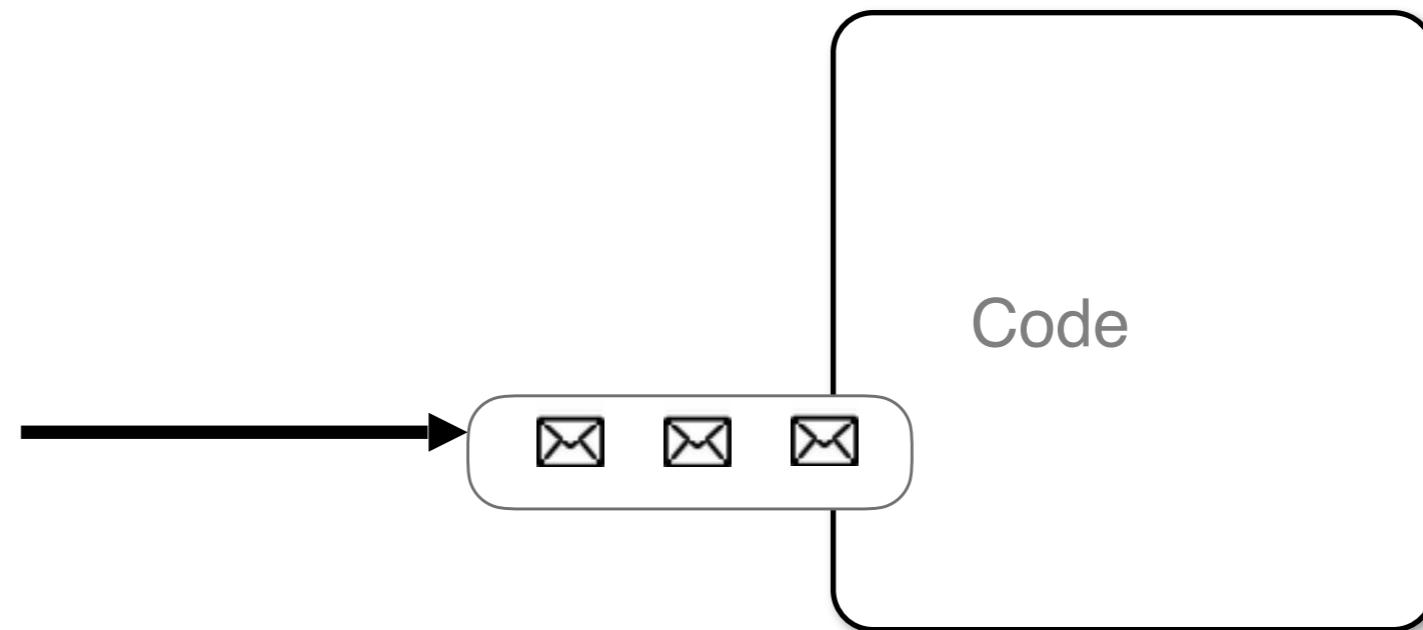


\* Message

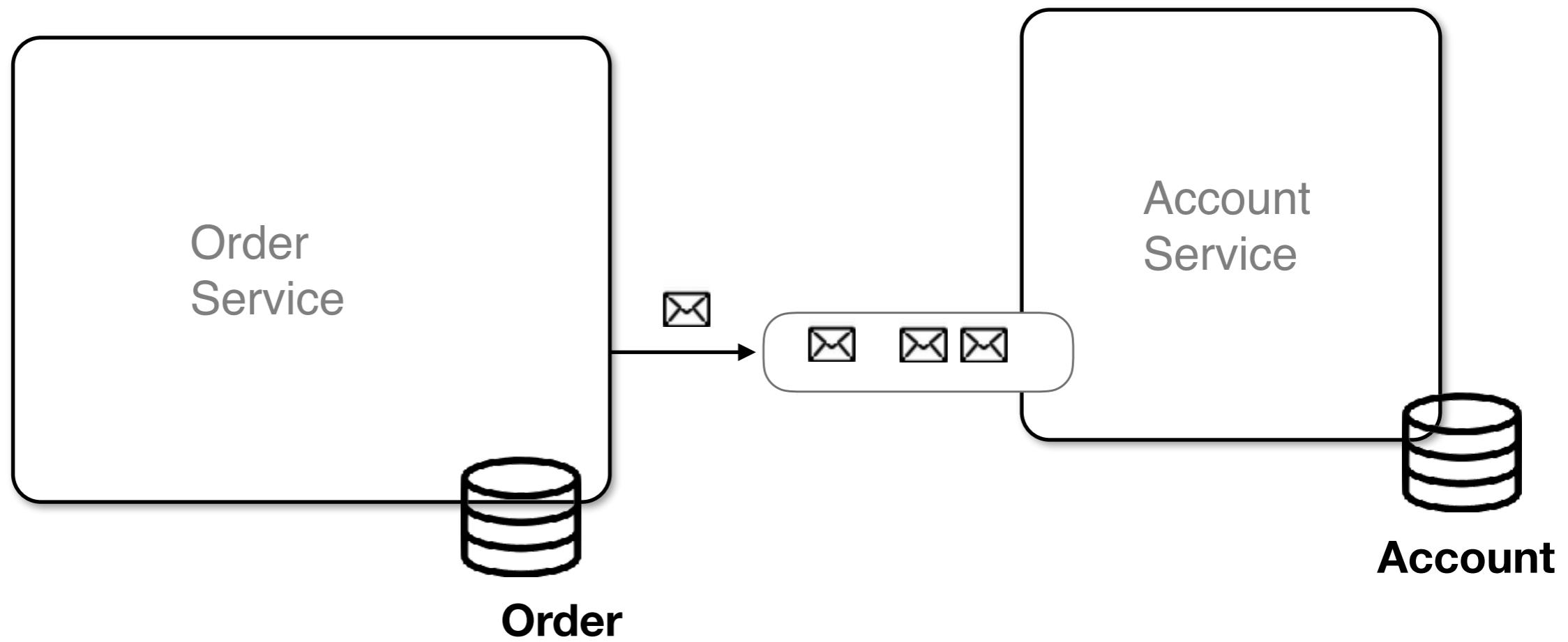
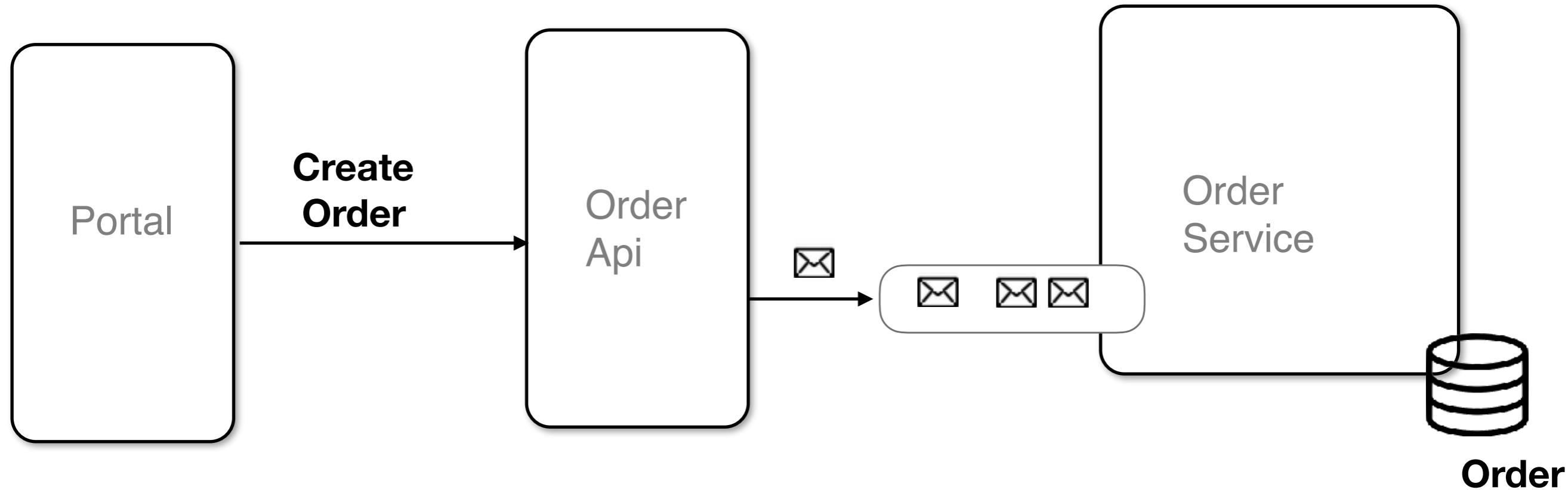


	<< API >>	<< Message >>	Solution
Ordering	Ordered (+)	Unordered(-)	?
Duplicate	Yes (-)	Yes(-)	Idempotency
Protocol	2 way (+)	One way (-)	
Consistency	Immediate (++)	Eventual (- -)	
Dev effort	++	--	
Operational effort	++	--	
Resilience (recover)	No (-) retry logic	Yes (+)	
Connection	Always connected	Occousionaly connected	
Load Leveling	--	++	Scalability
Low Coupling (maintainability)	--	++	
Distributed Comm Patterns	--	++	SAGA, Materialized View, ...

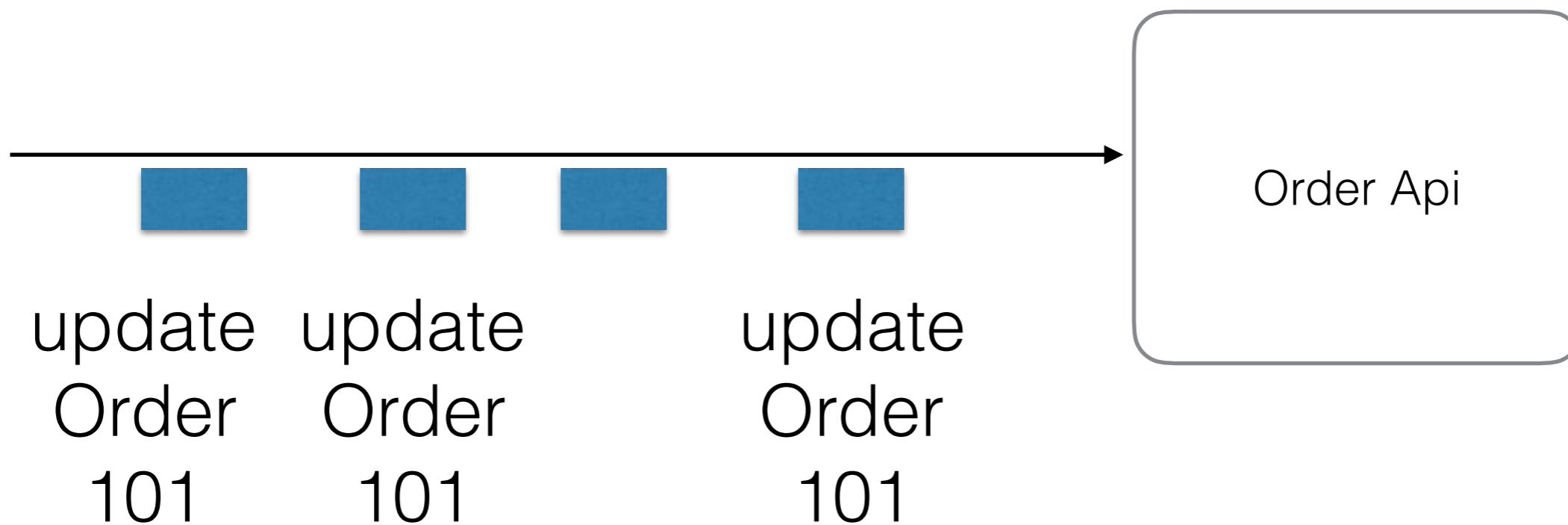
\* Message

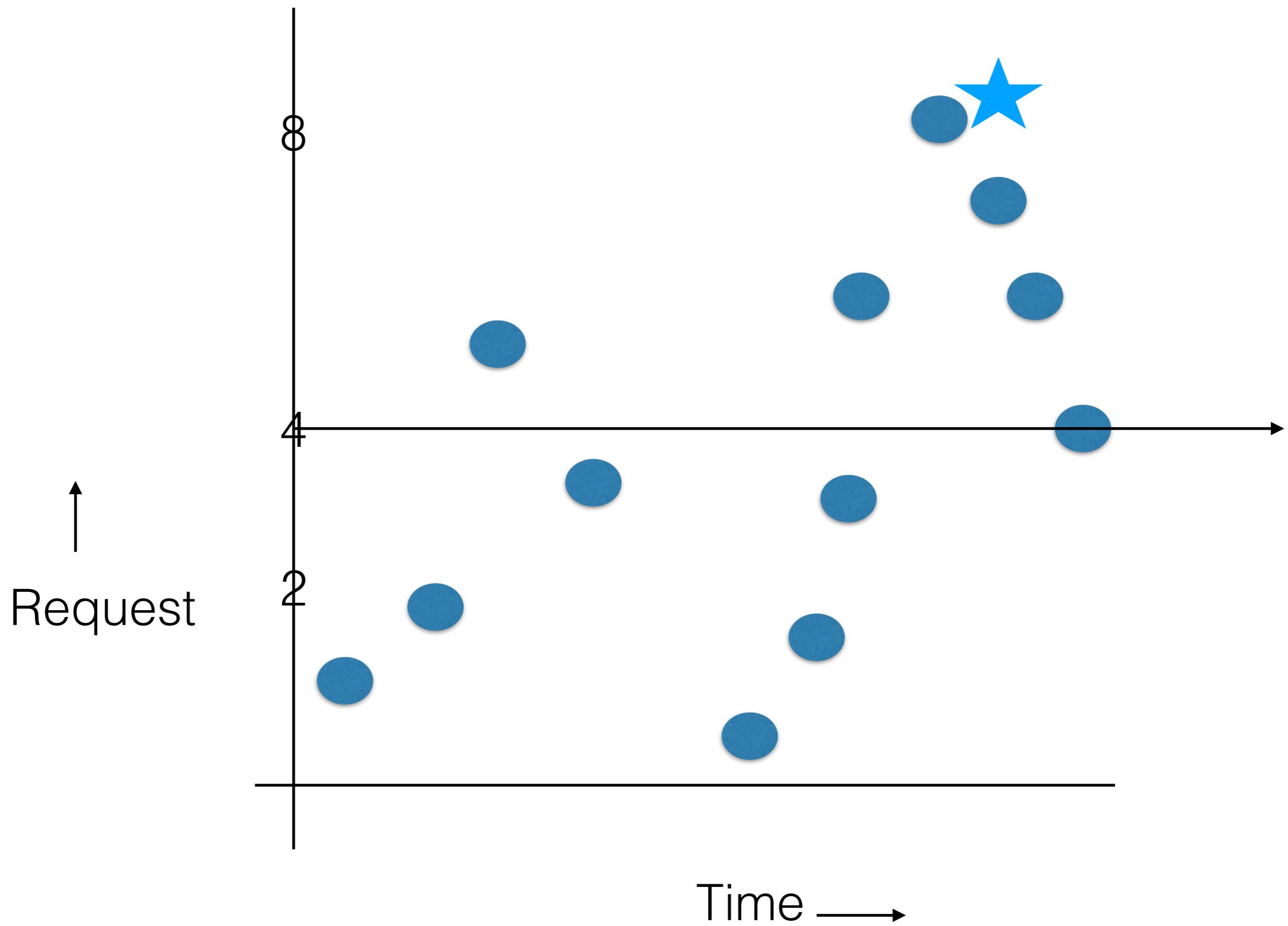


```
fun(){  
    while(true){  
        Msg m = GetMessage("sub1");  
        Run domain Logic  
        m.ack();  
    }  
}
```

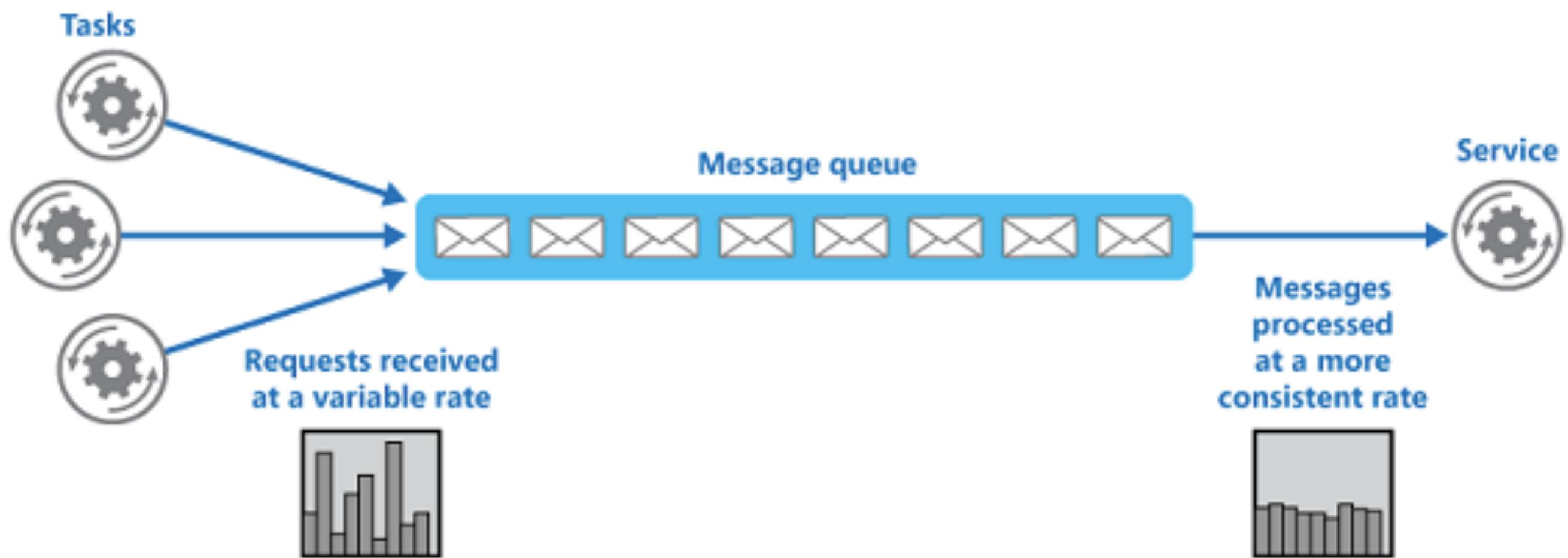


Idempotency

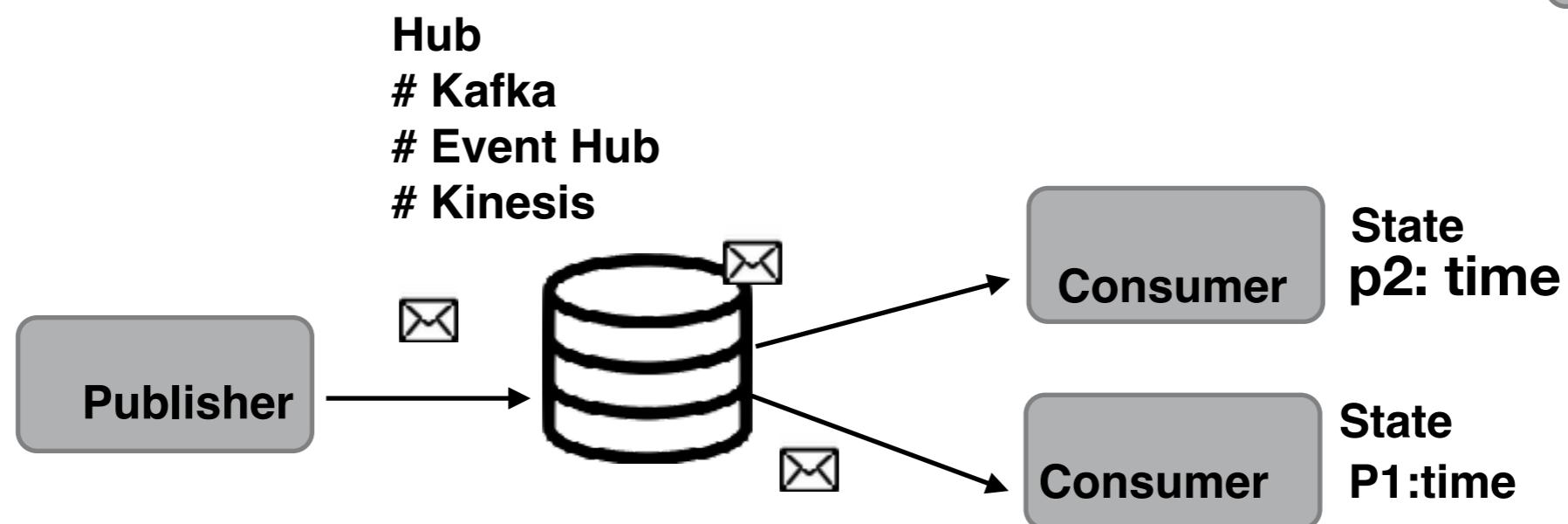
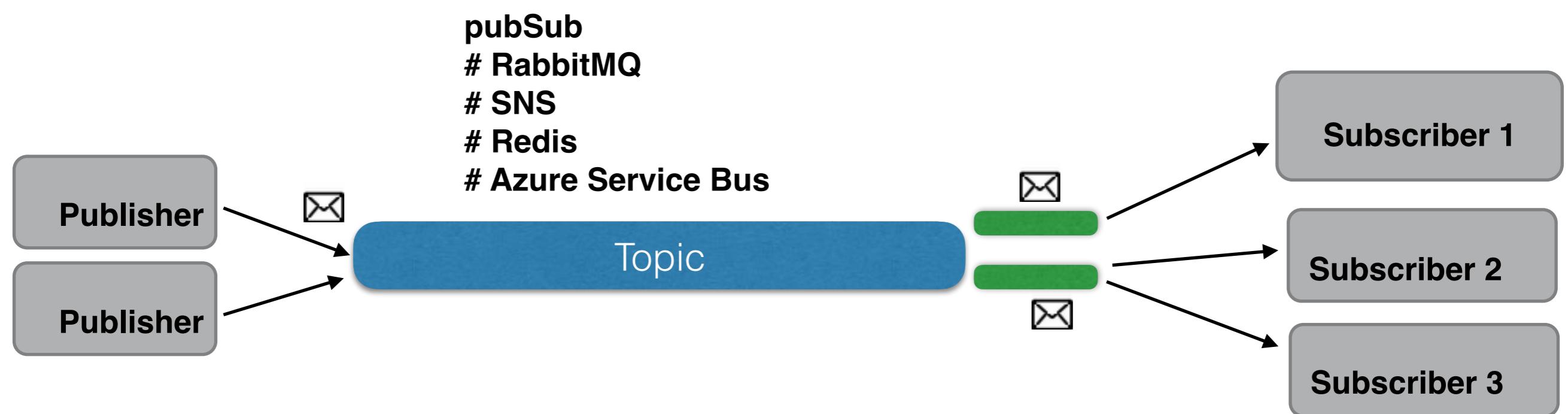
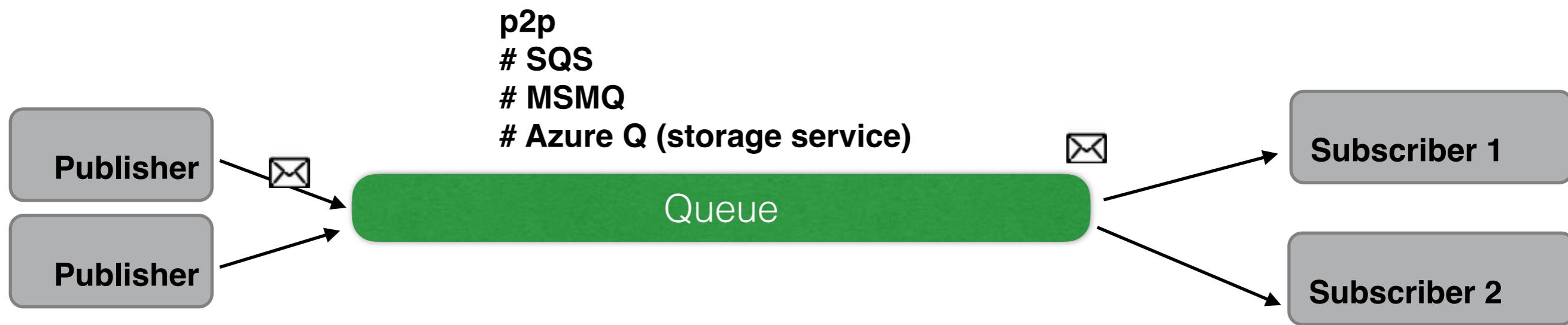


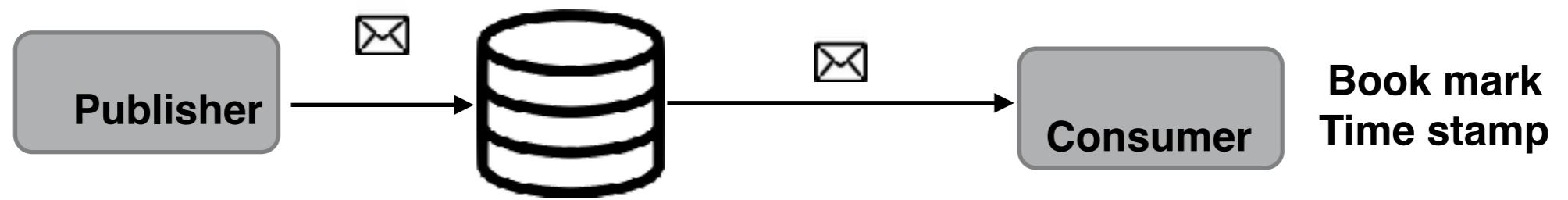


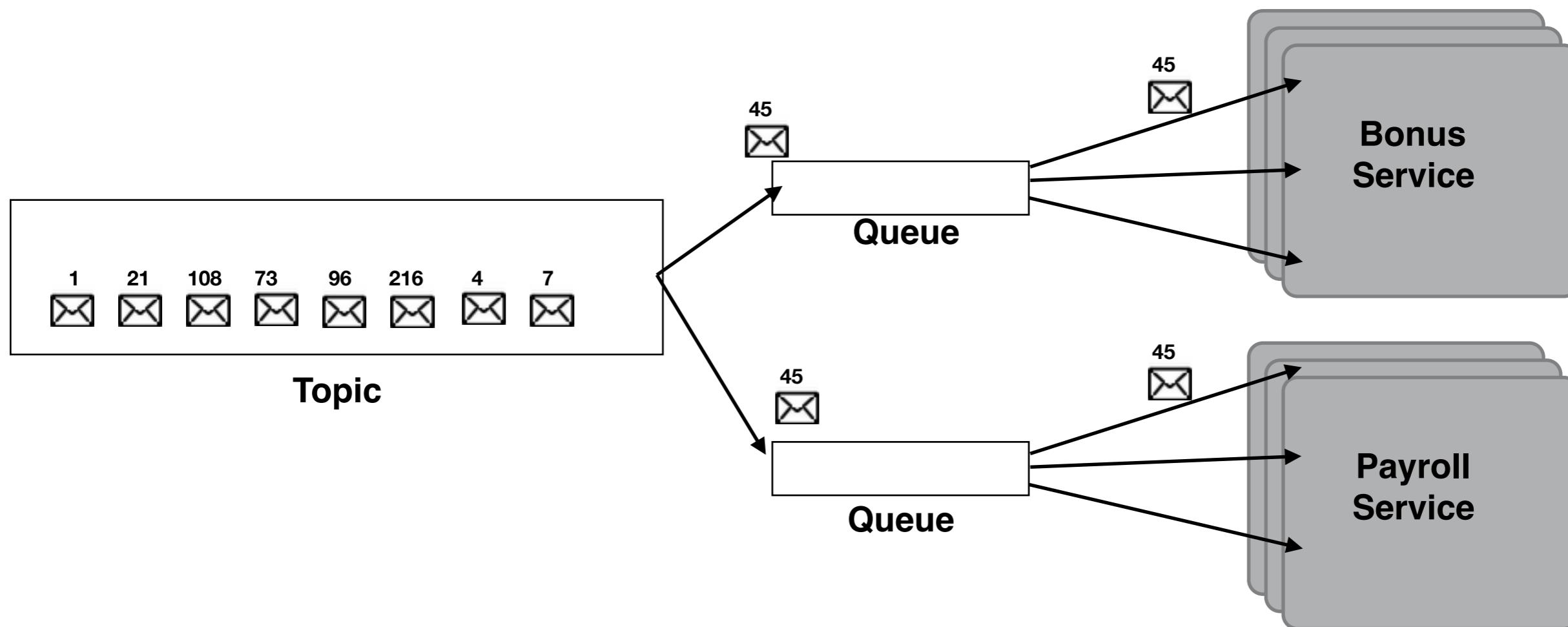
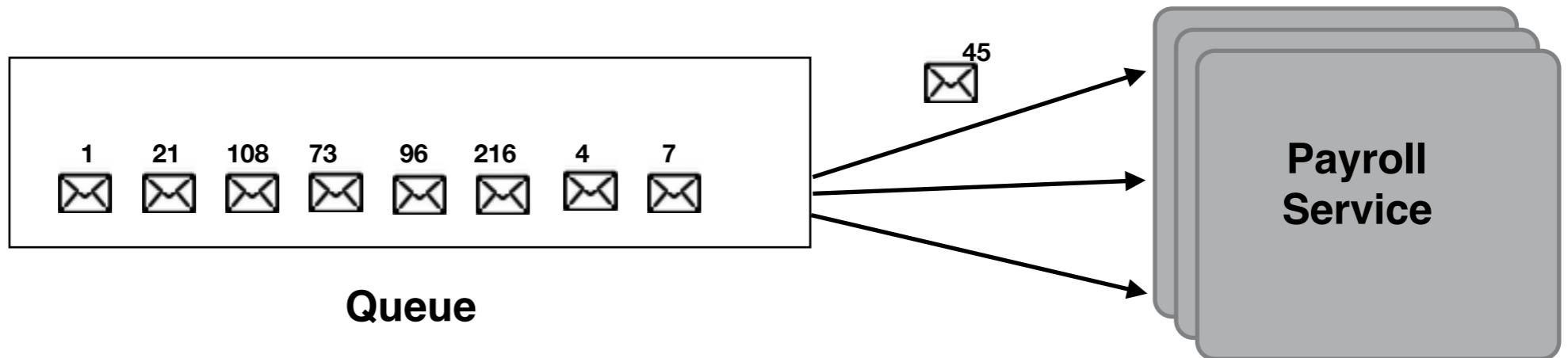
# Queue-based Load Levelling

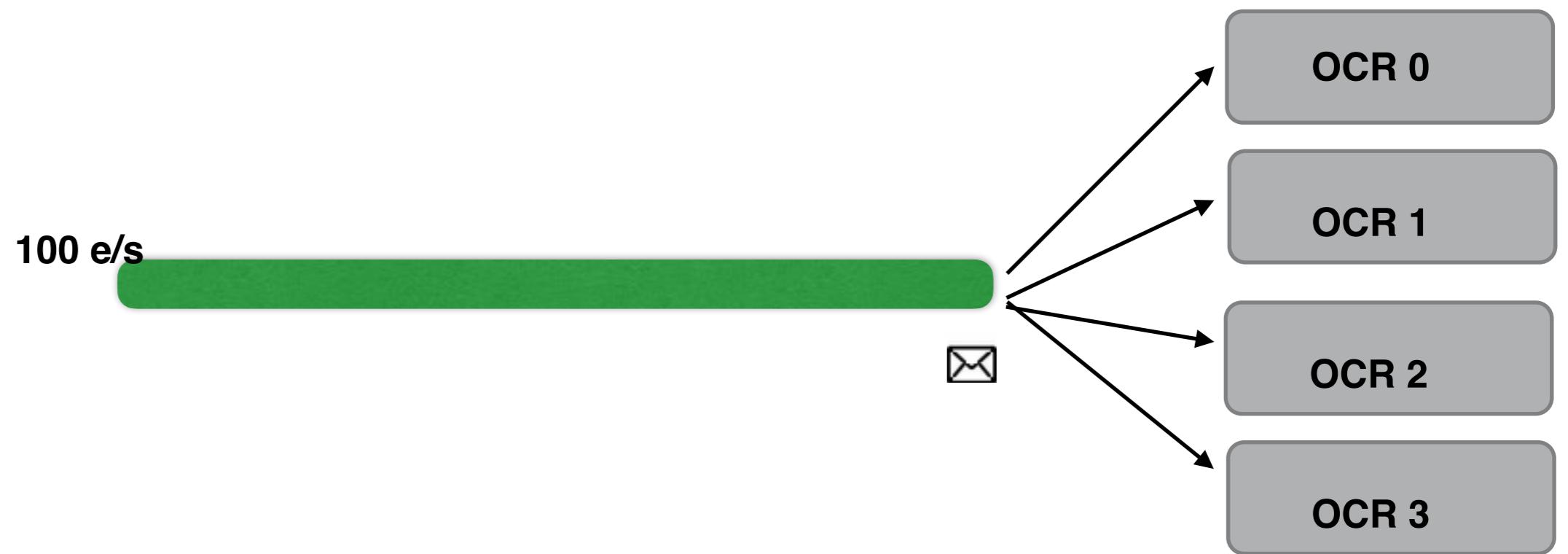


source:msdn

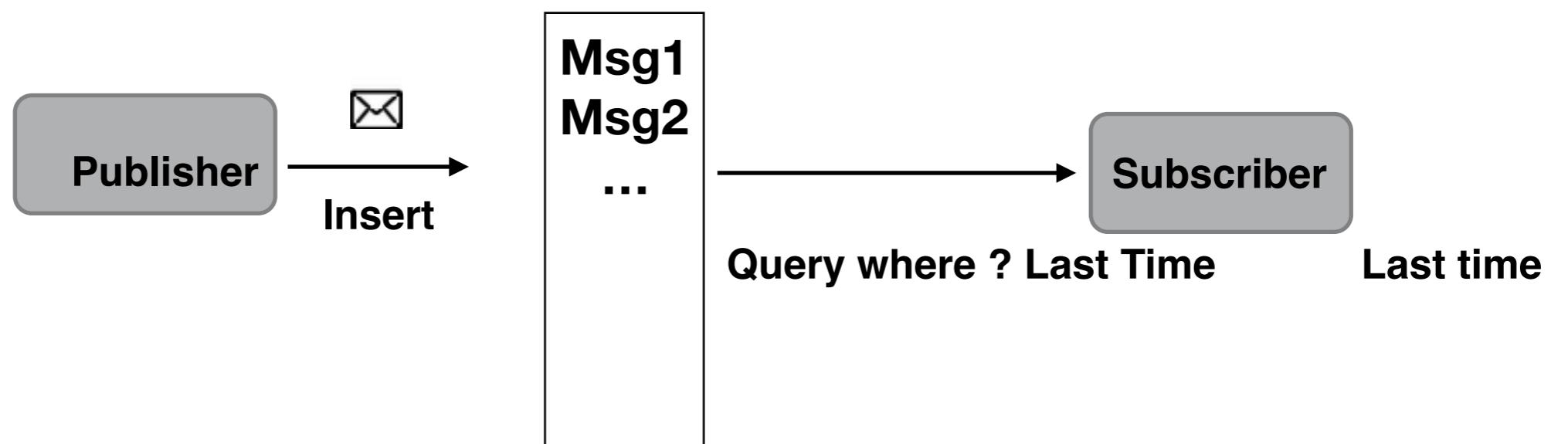


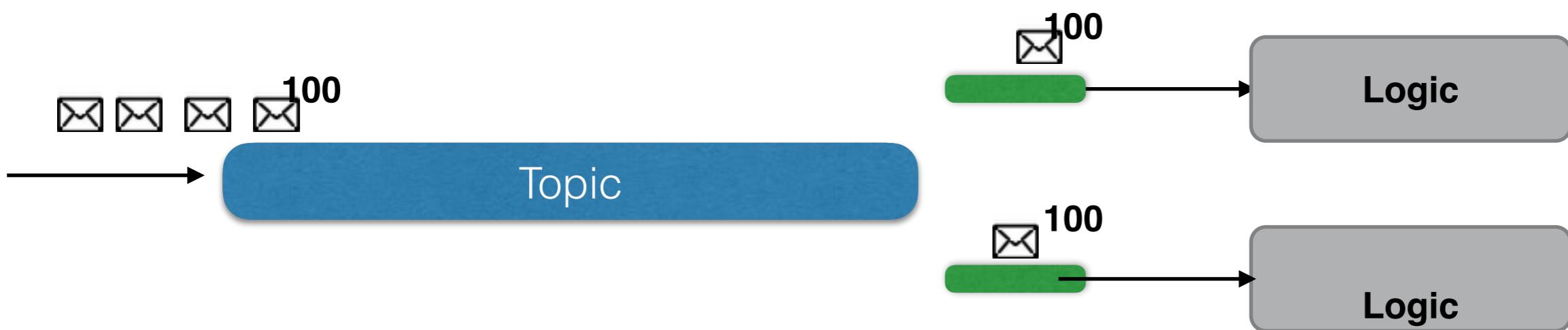






<https://www.learningjournal.guru/courses/kafka/kafka-foundation-training/offset-management/>

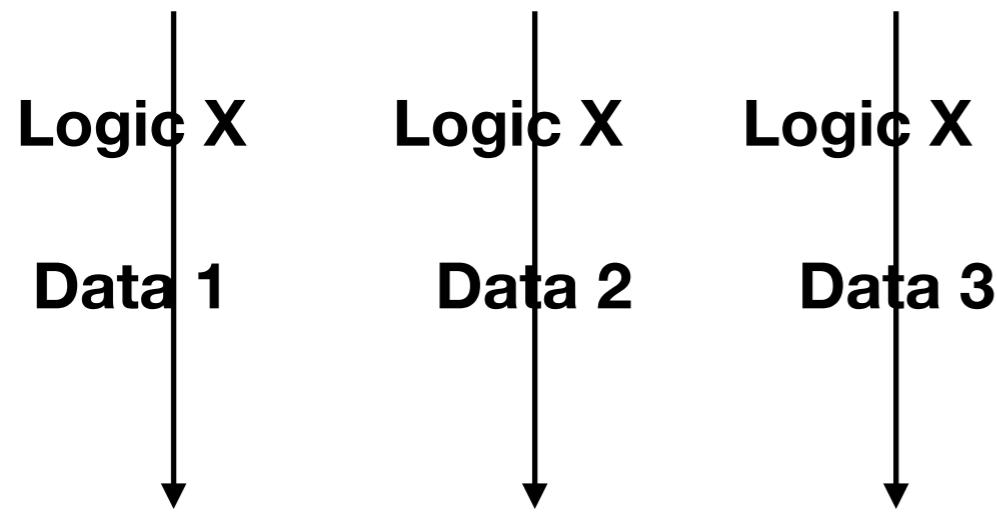




## Data Parallelism

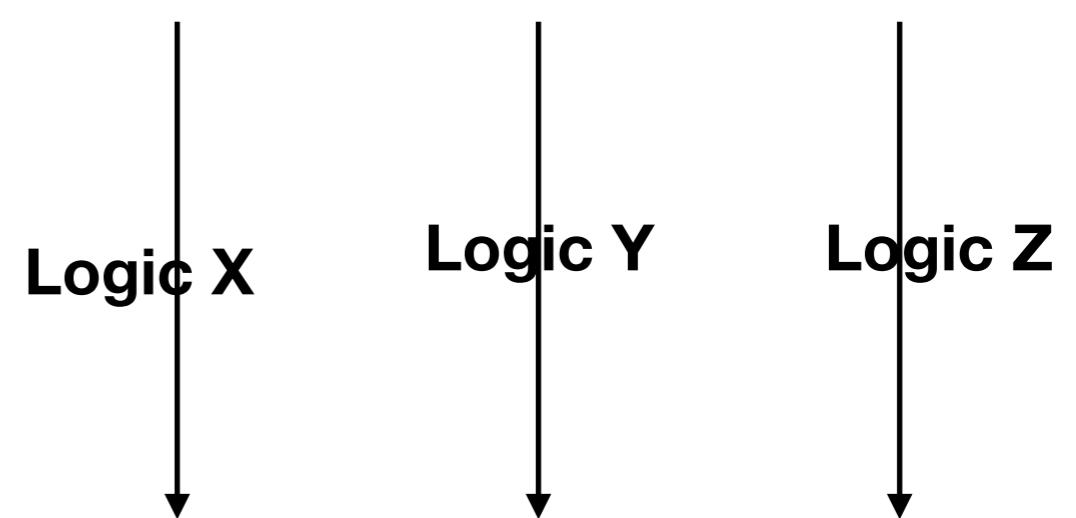
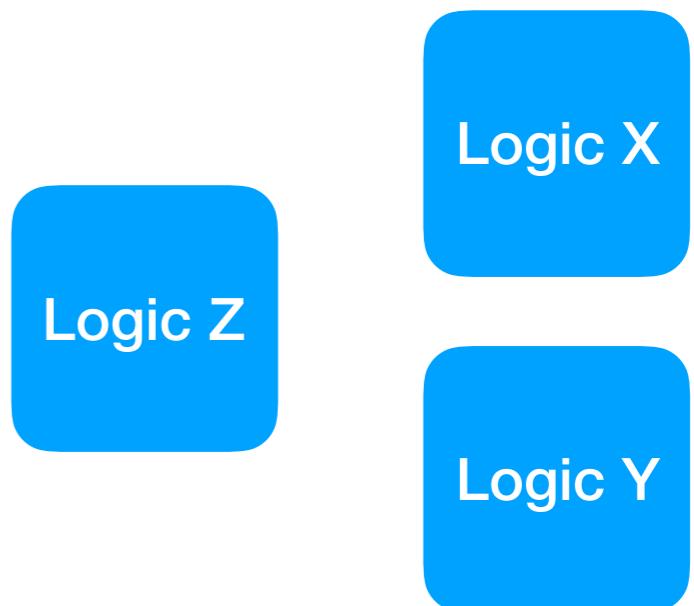


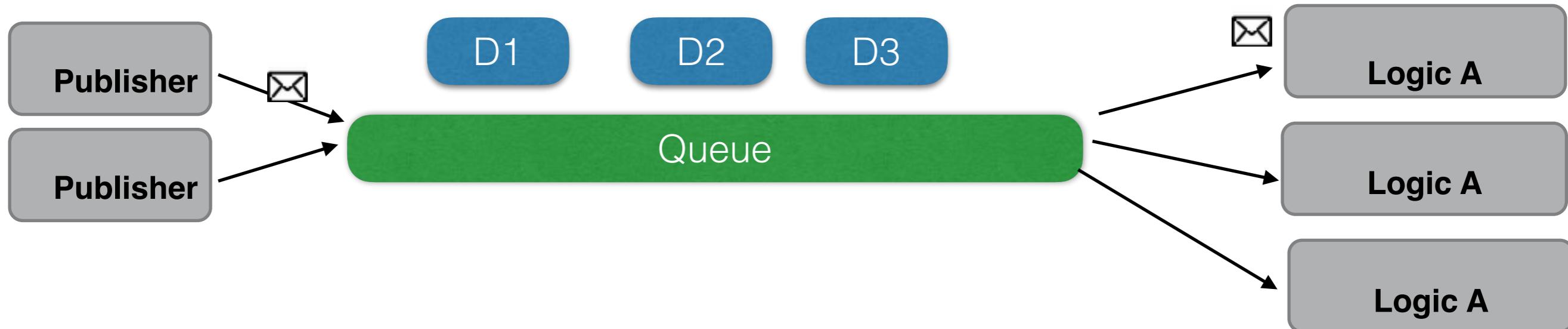
data 1  
data 2  
data 3  
data 4  
...  
data n



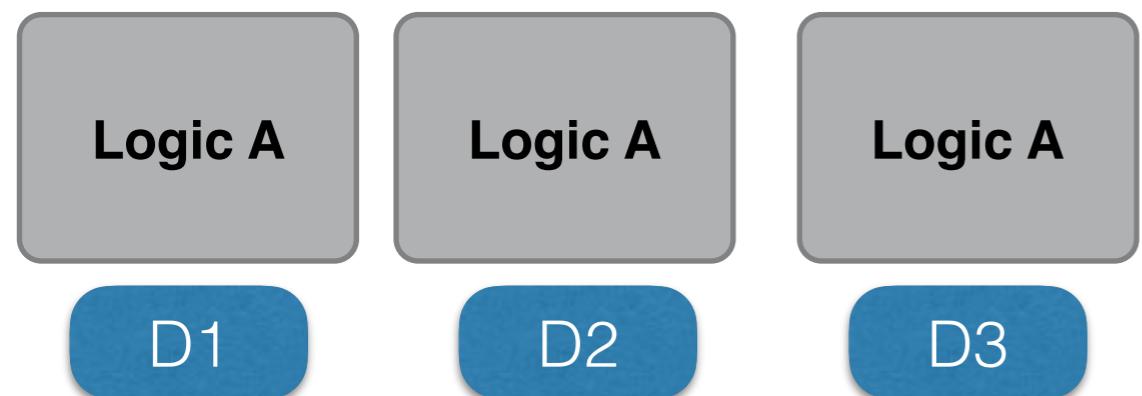
## Parallel

## Task Parallelism

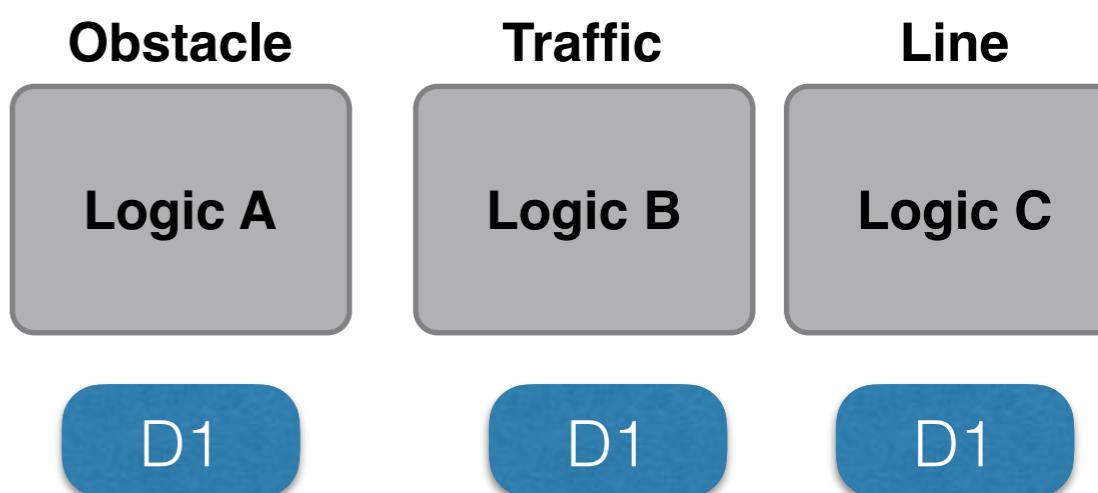




**Data Concurrency**



**Task Concurrency**

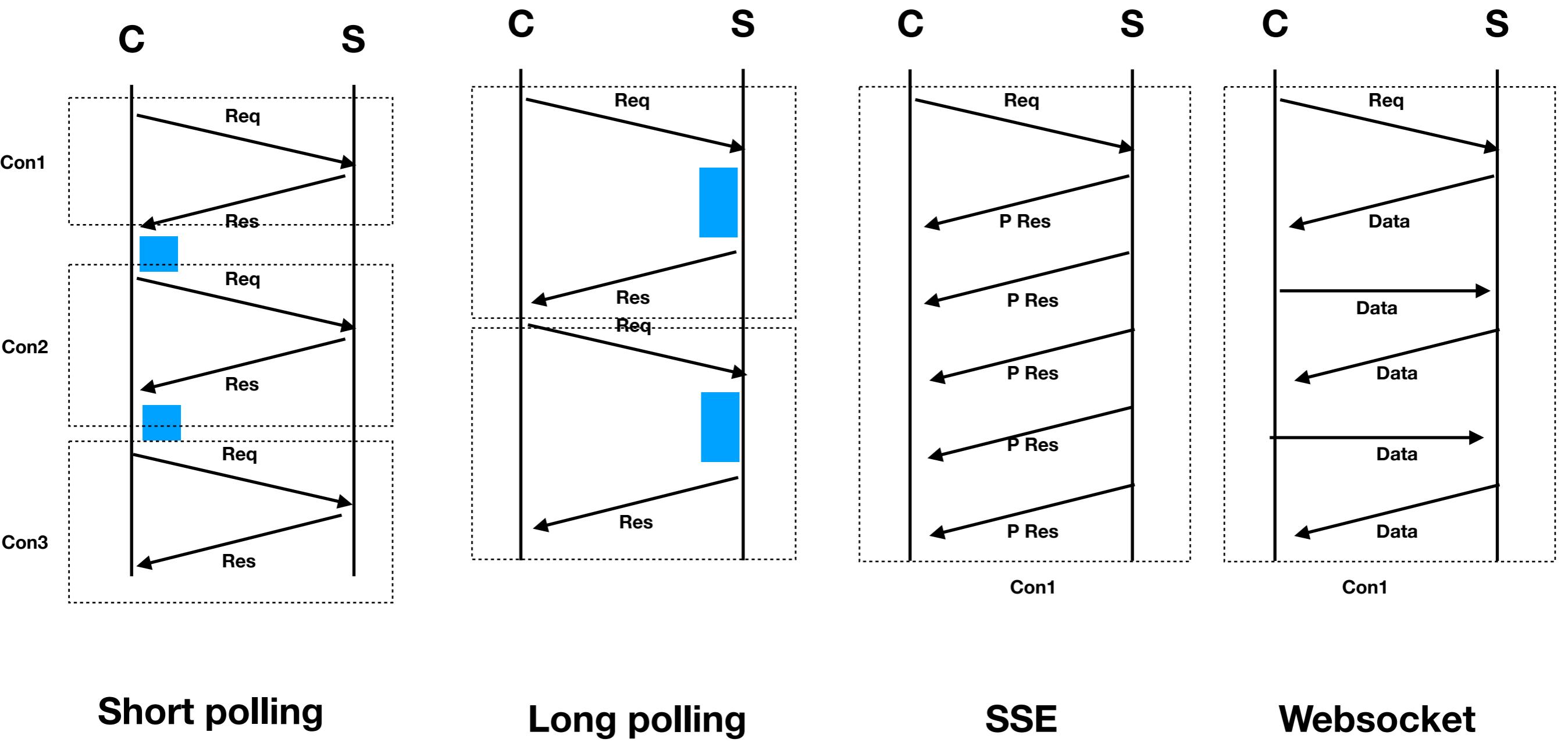


	Kafka	Pulsar	RabbitMQ (Mirrored)
<b>Peak Throughput (MB/s)</b>	605 MB/s	305 MB/s	38 MB/s
<b>p99 Latency (ms)</b>	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
<b>Number of Programming Languages Supported</b>	17	6	22
<b>Stack Overflow Questions</b>	21,233	134	11,430
<b>Pub/sub</b>	Yes	Yes	Yes
<b>Message routing</b>	Medium	Medium	High
<b>Queueing</b>	Low	Medium	High
<b>Event Streaming</b>	High	Medium	No
<b>Mission-critical</b>	High	Low	High
<b>Slack Community Size</b>	23,057	2,332	7,492
<b>Meetups</b>	486	1	1
<b>Message replay, time travel</b>	+++	+++	-
<b>Exactly-once processing</b>	+++	+	-
<b>consumption</b>	Pull	Push	Push
<b>Permanent storage</b>	Yes	Partial	No

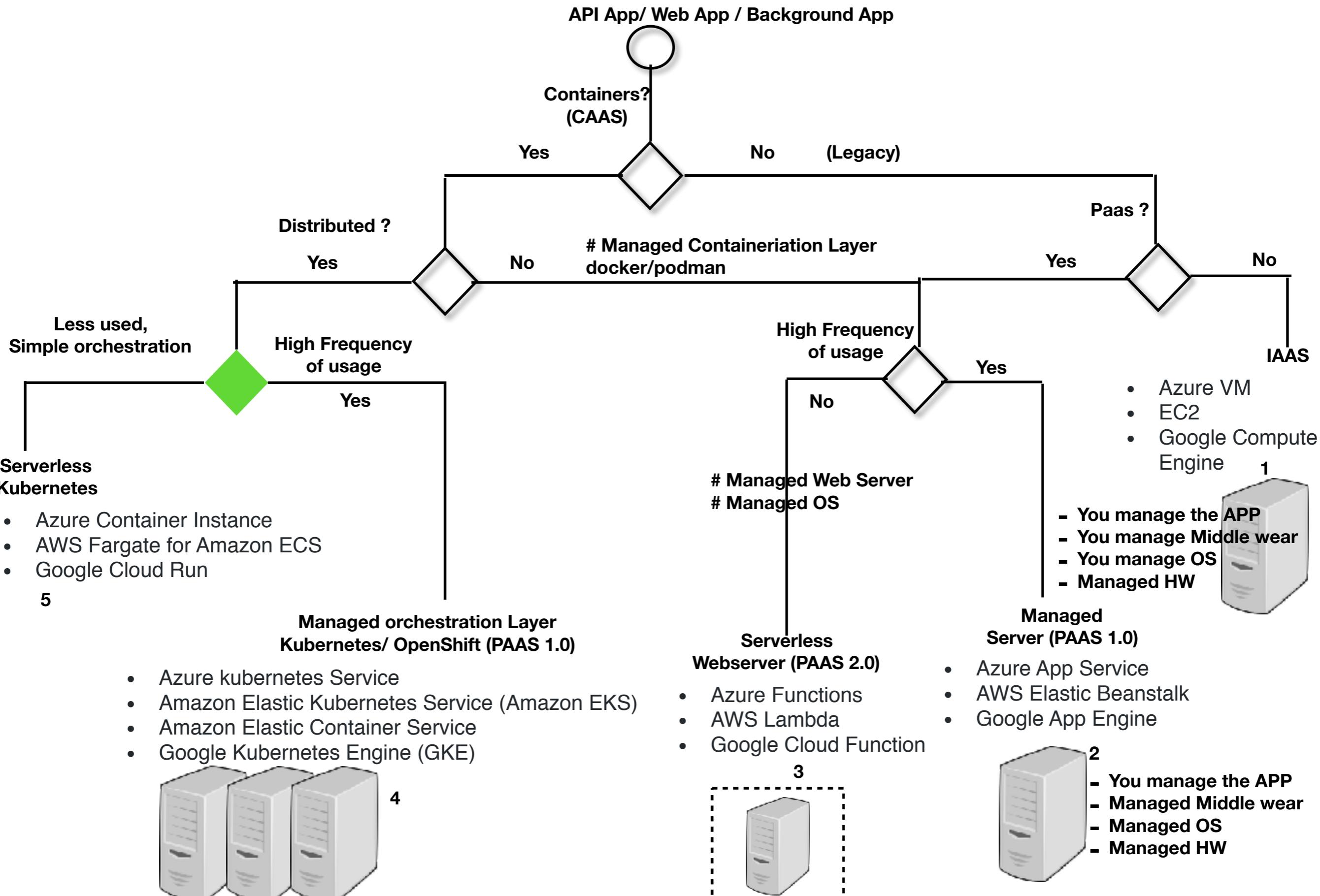
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent	yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, 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	

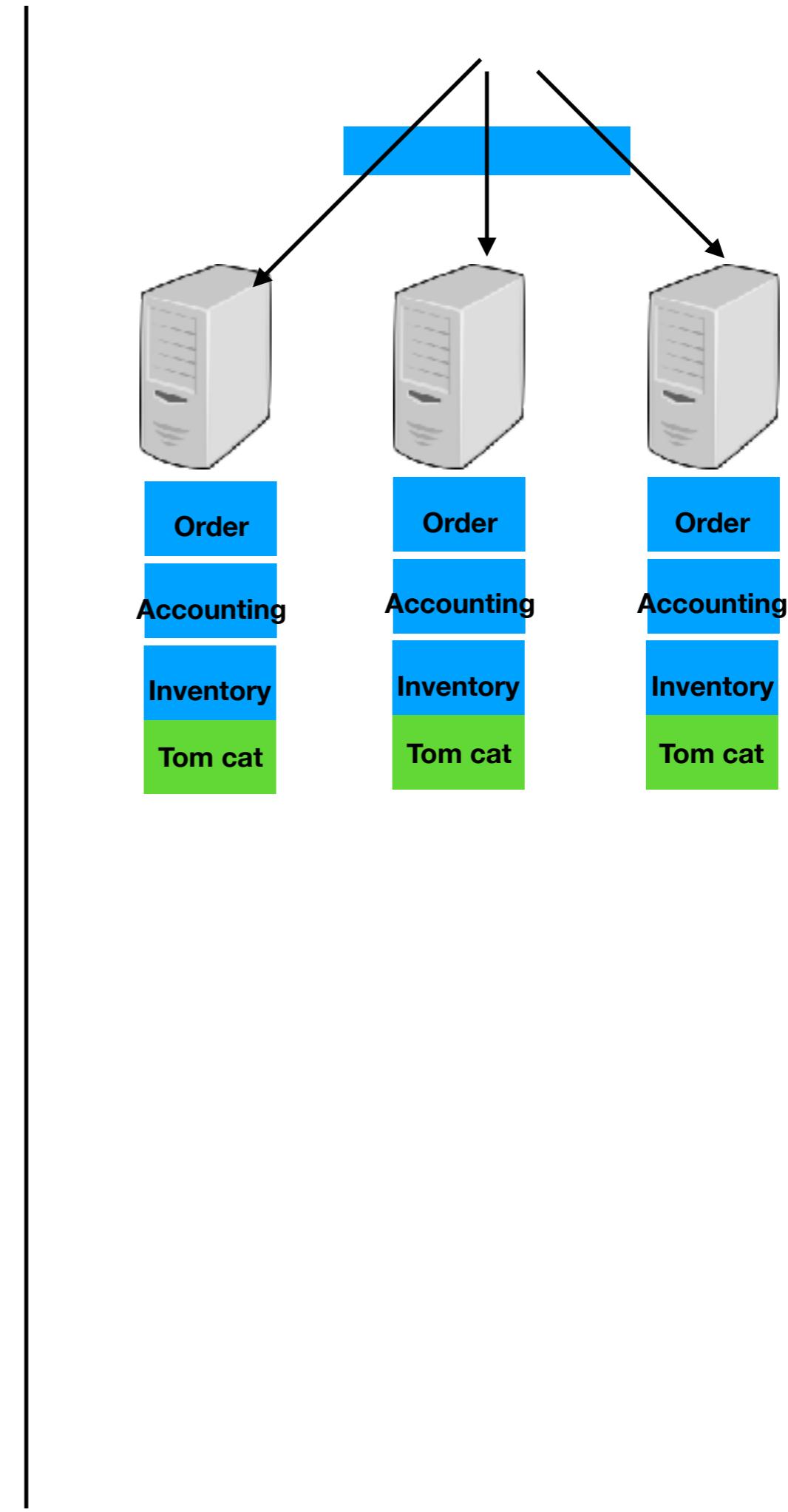
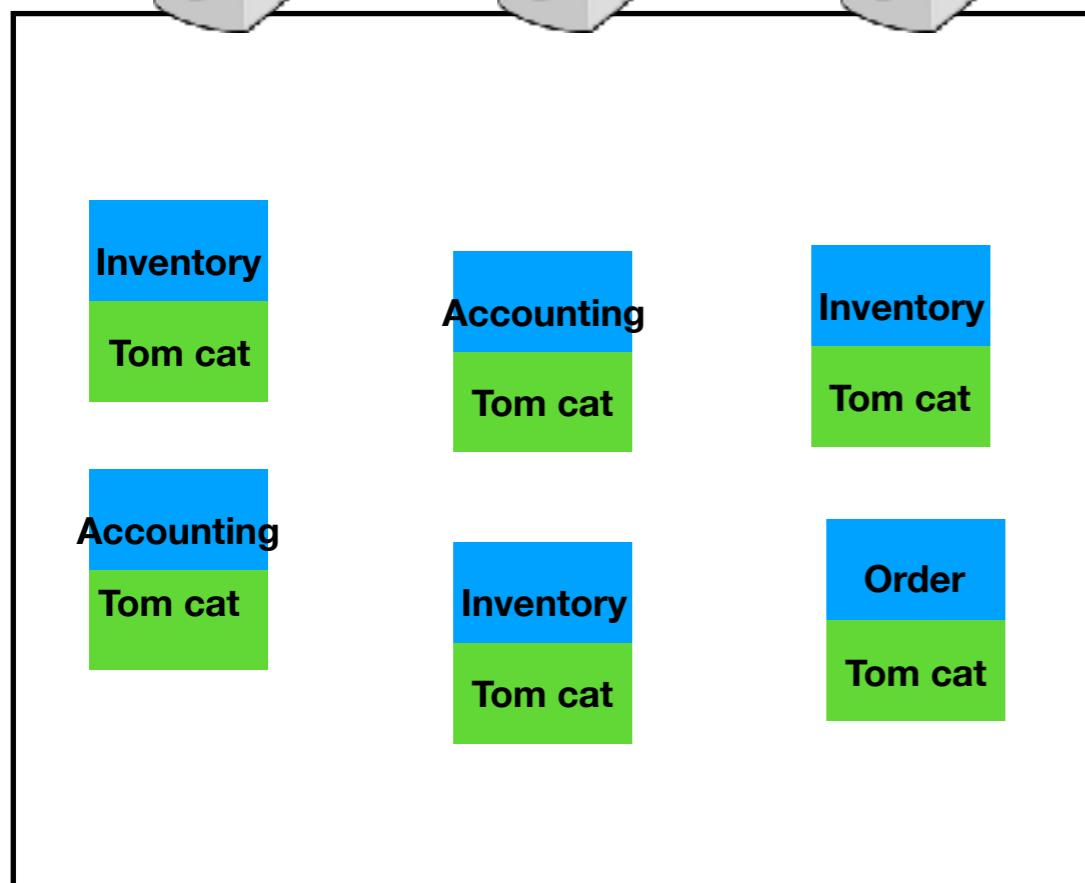
Table 1

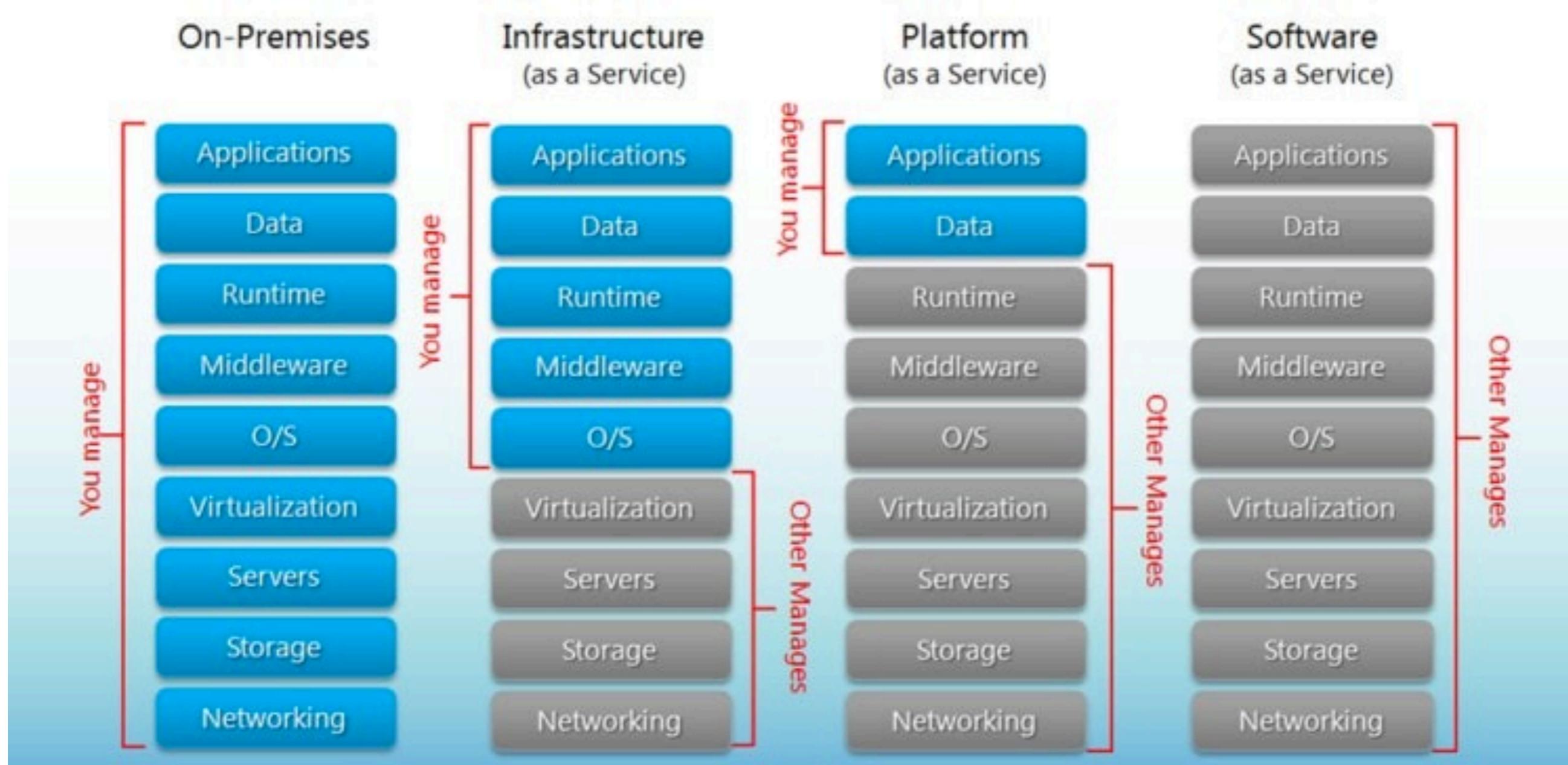
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .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	
Potential data loss	Yes	Yes	
Community and vendor support	Good	Good	Good
Building an event store system (used as a store)	No	Yes	No
Ordering	not guaranteed	Guaranteed	
Replay events	No	Yes	No
Transactions	No	Yes	No



# Choose Operational Compute







Isolated Env.

My app 1

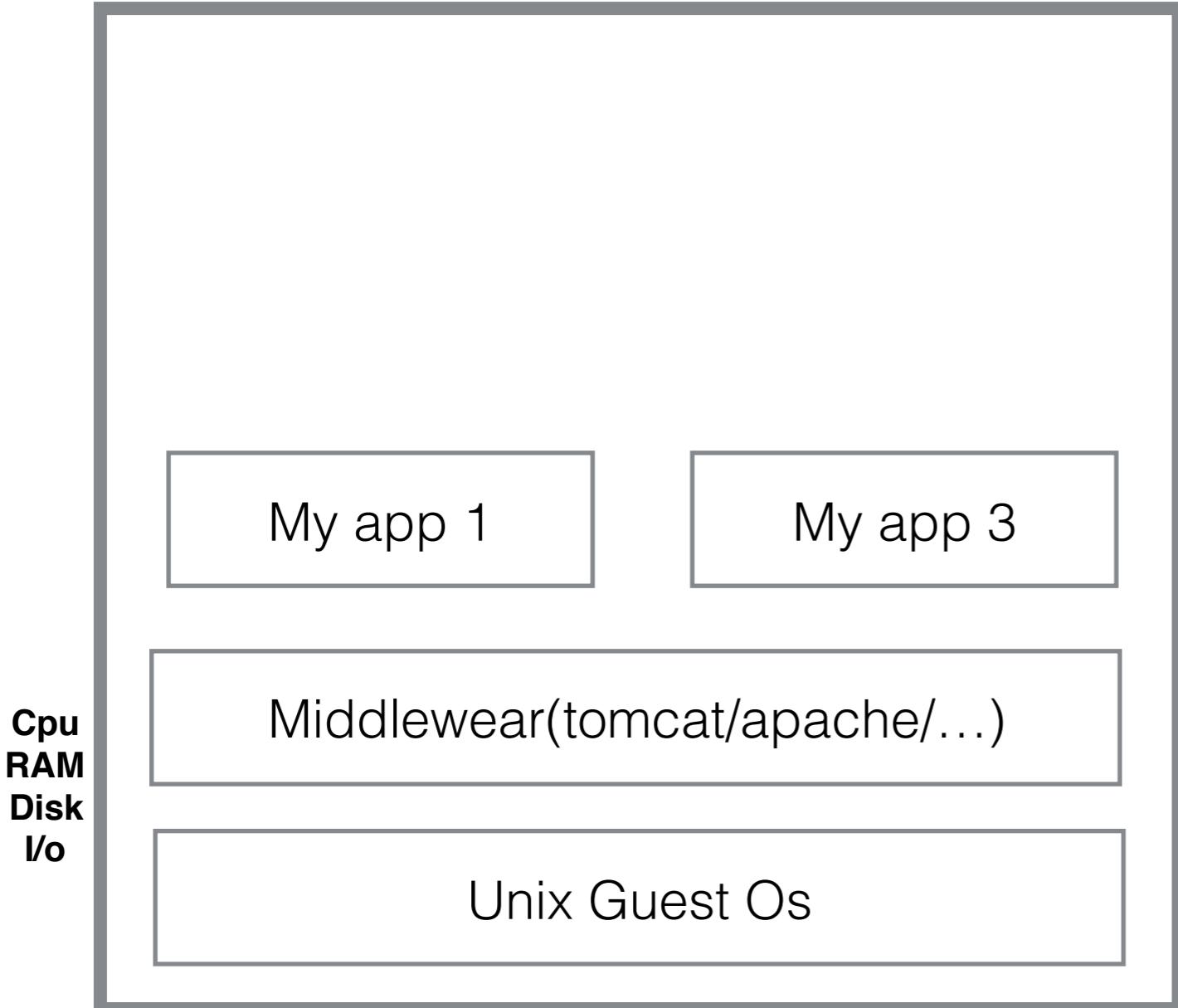
Middlewear (tomcat/ Apache/ IIS / Nodejs...)

Os

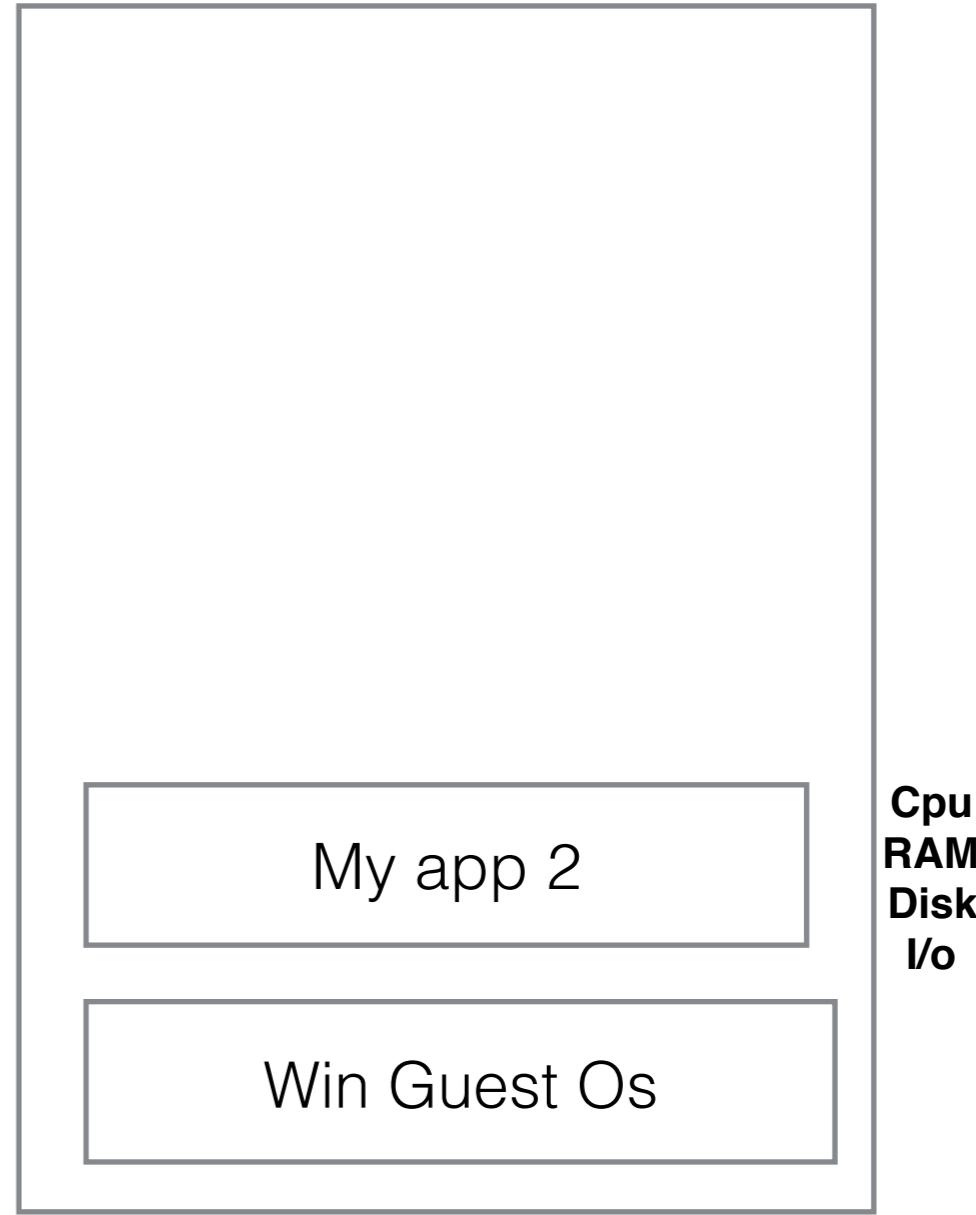
Cpu  
RAM  
Disk  
I/o

**Machine**

VM1



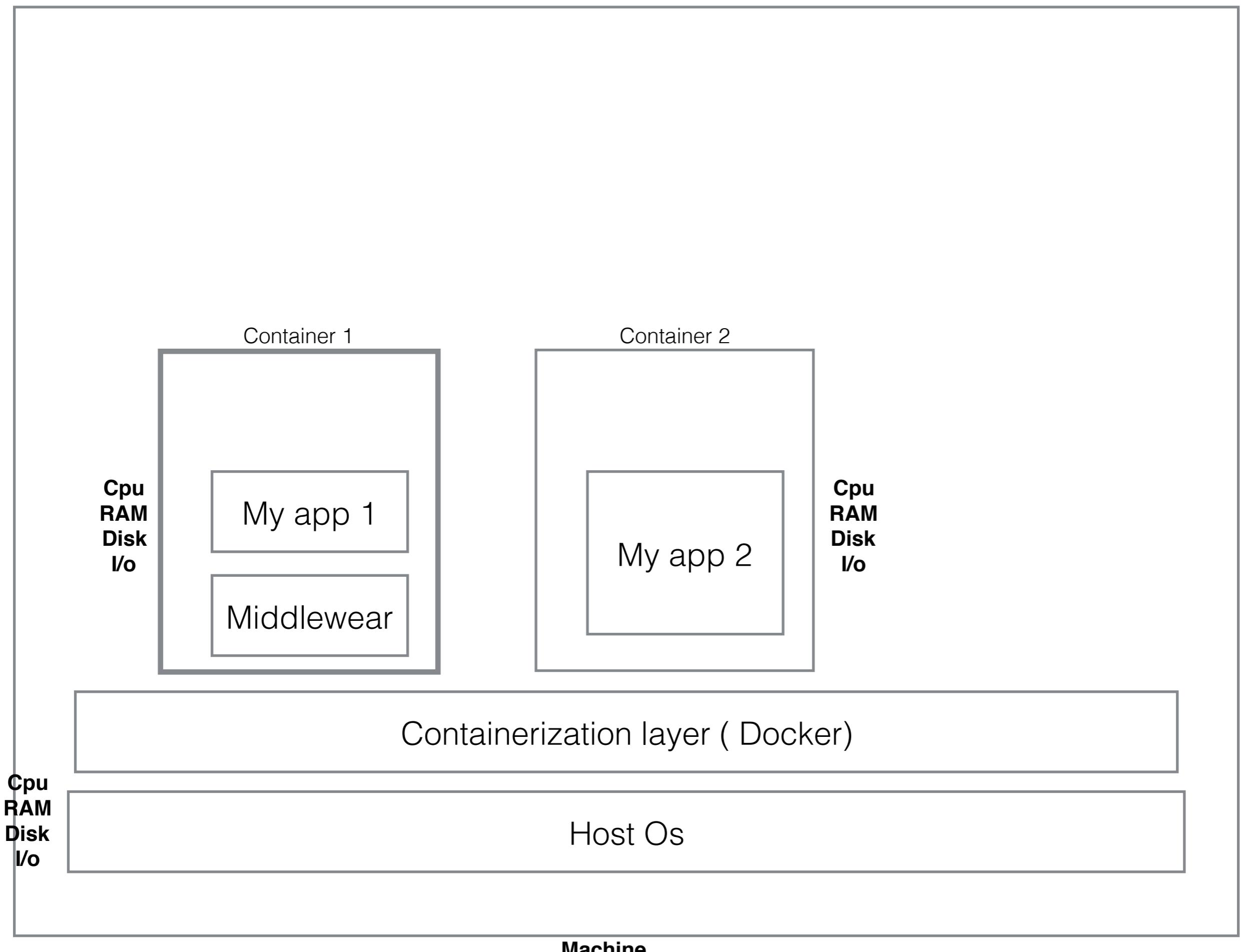
VM2



**Machine**

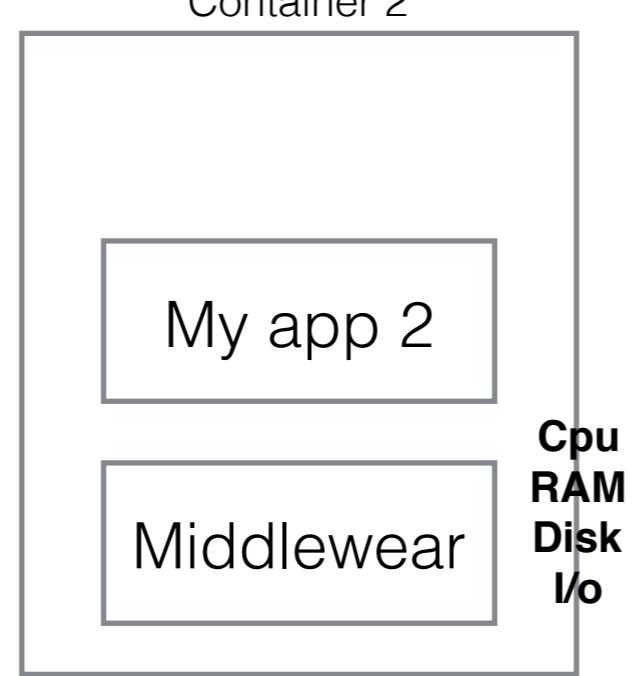
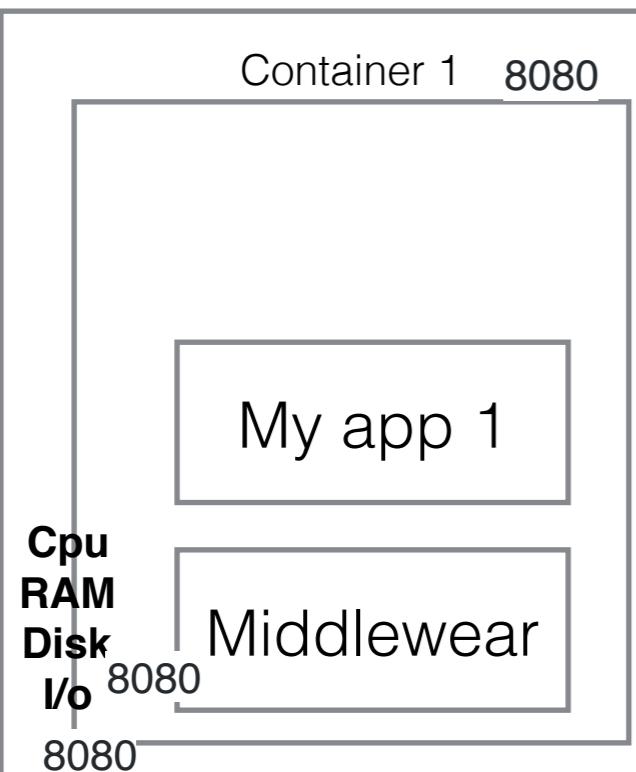
**Cpu  
RAM  
Disk  
I/o**

**Cpu  
RAM  
Disk  
I/o**



VM1

VM2



Containerization layer ( Docker)

Cpu  
RAM  
Disk  
I/o

Unix Guest Os

Win Guest Os

Cpu  
RAM  
Disk  
I/o

Virtualisation layer (Hyperviser)

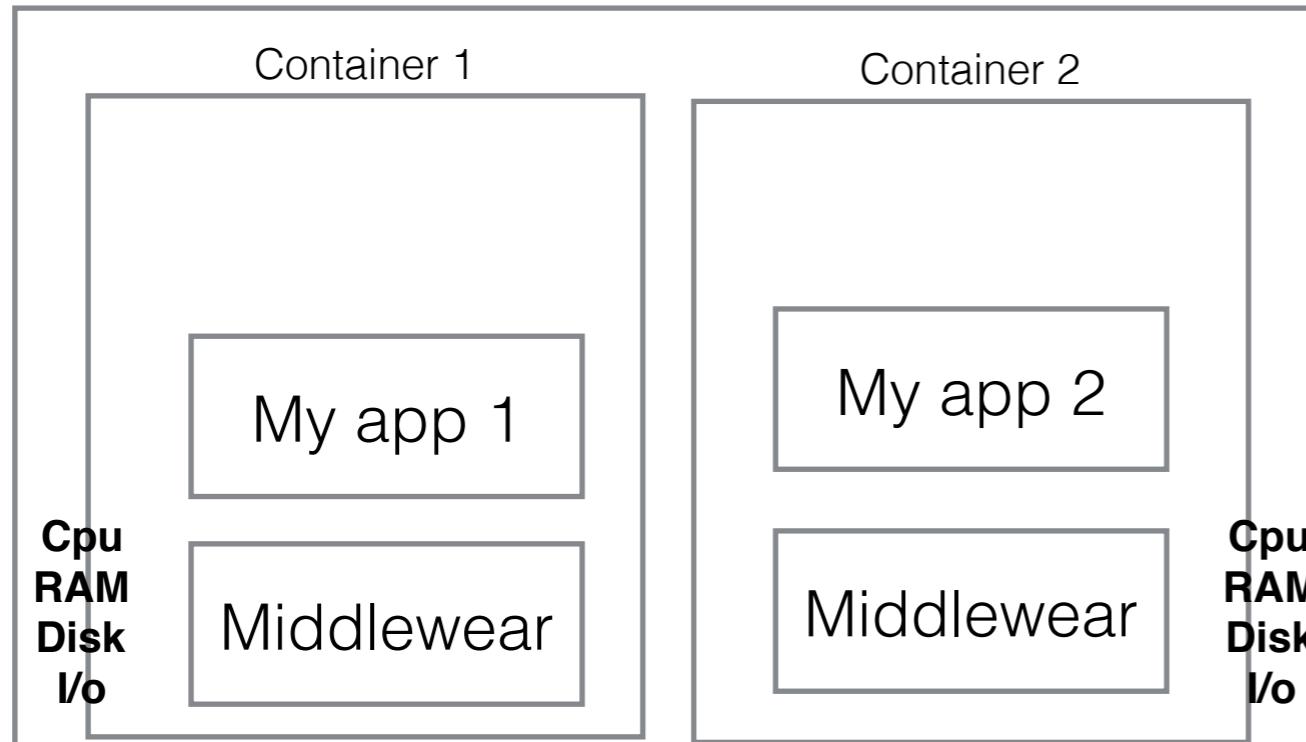
Cpu  
RAM  
Disk  
I/o

Host Os (azure)

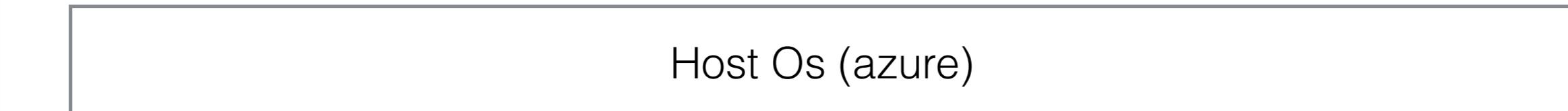
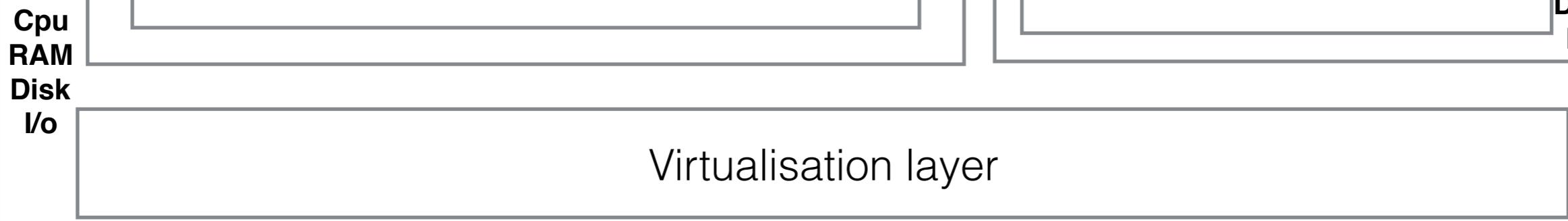
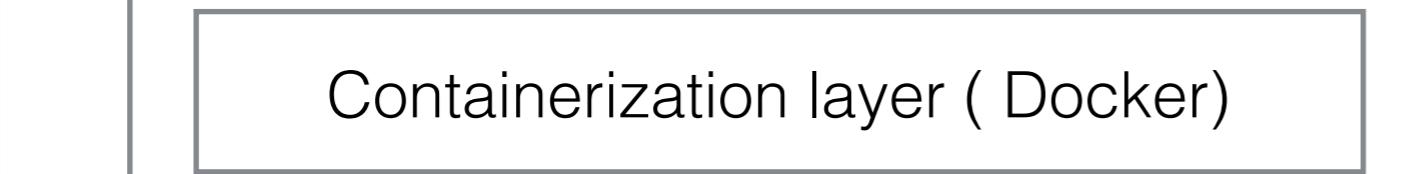
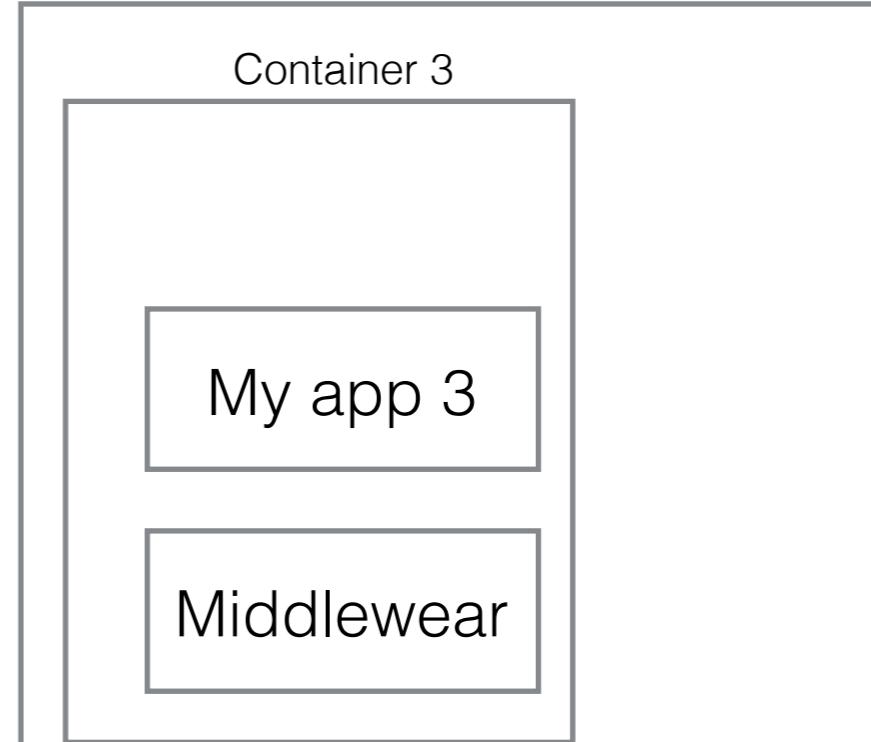
**Machine**

# Managing Container Life cycle

VM1



VM2



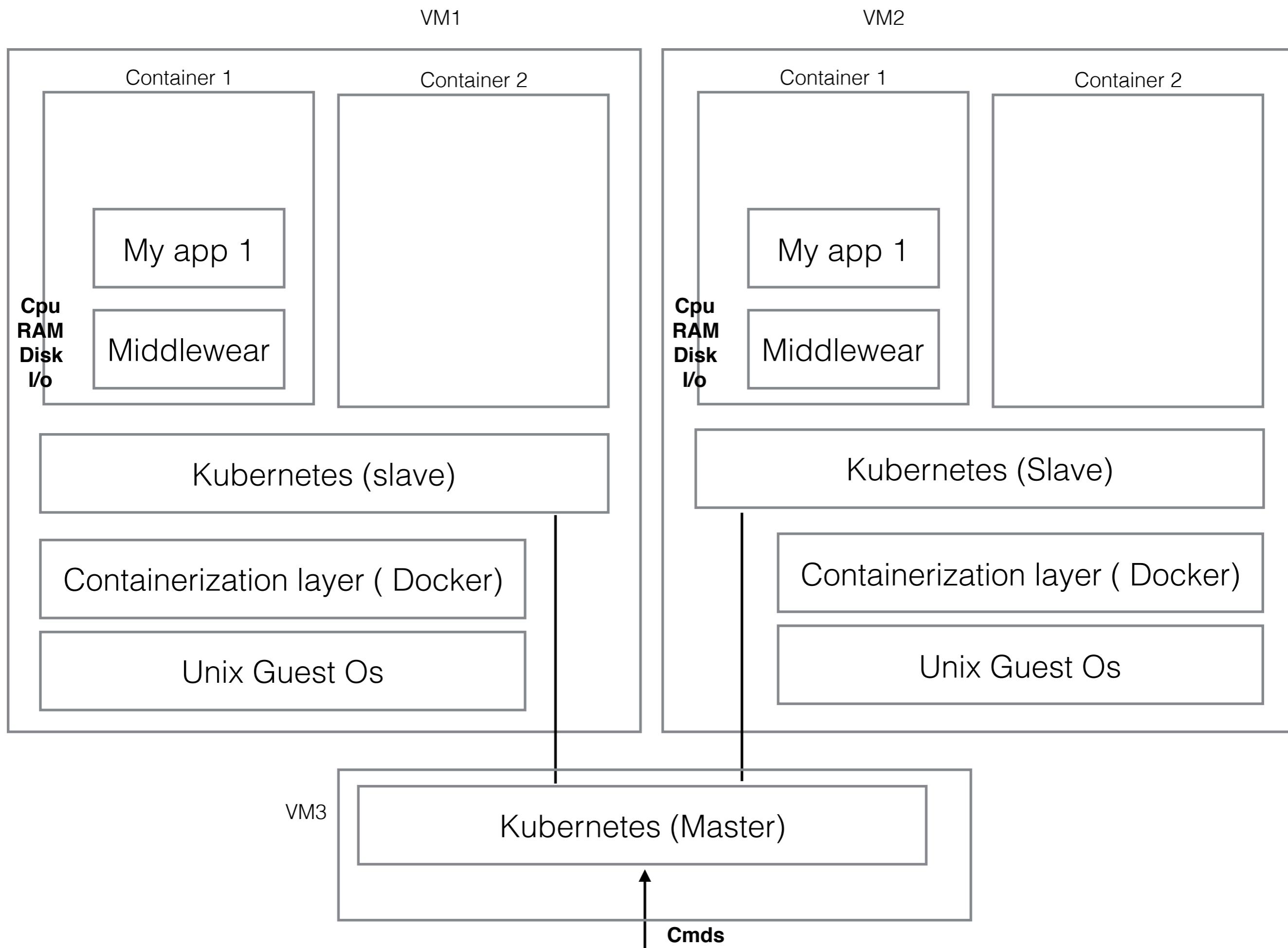
**Machine**

Kubernetes (slave)

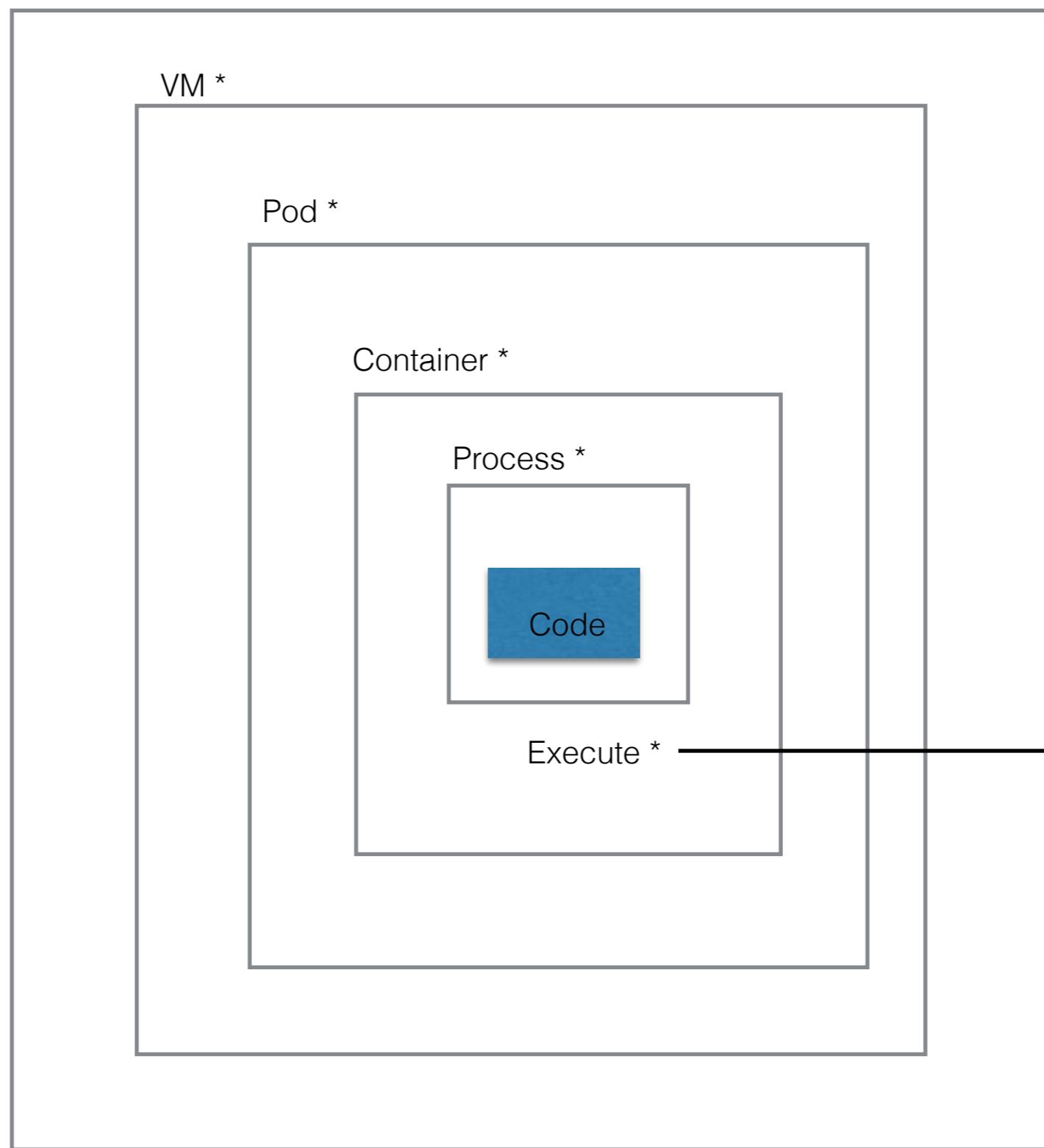
Kubernetes (Slave)

Kubernetes (Master)

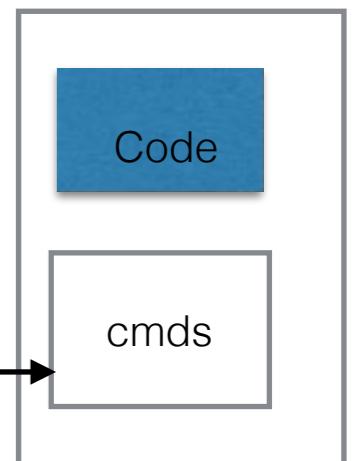




Cluster

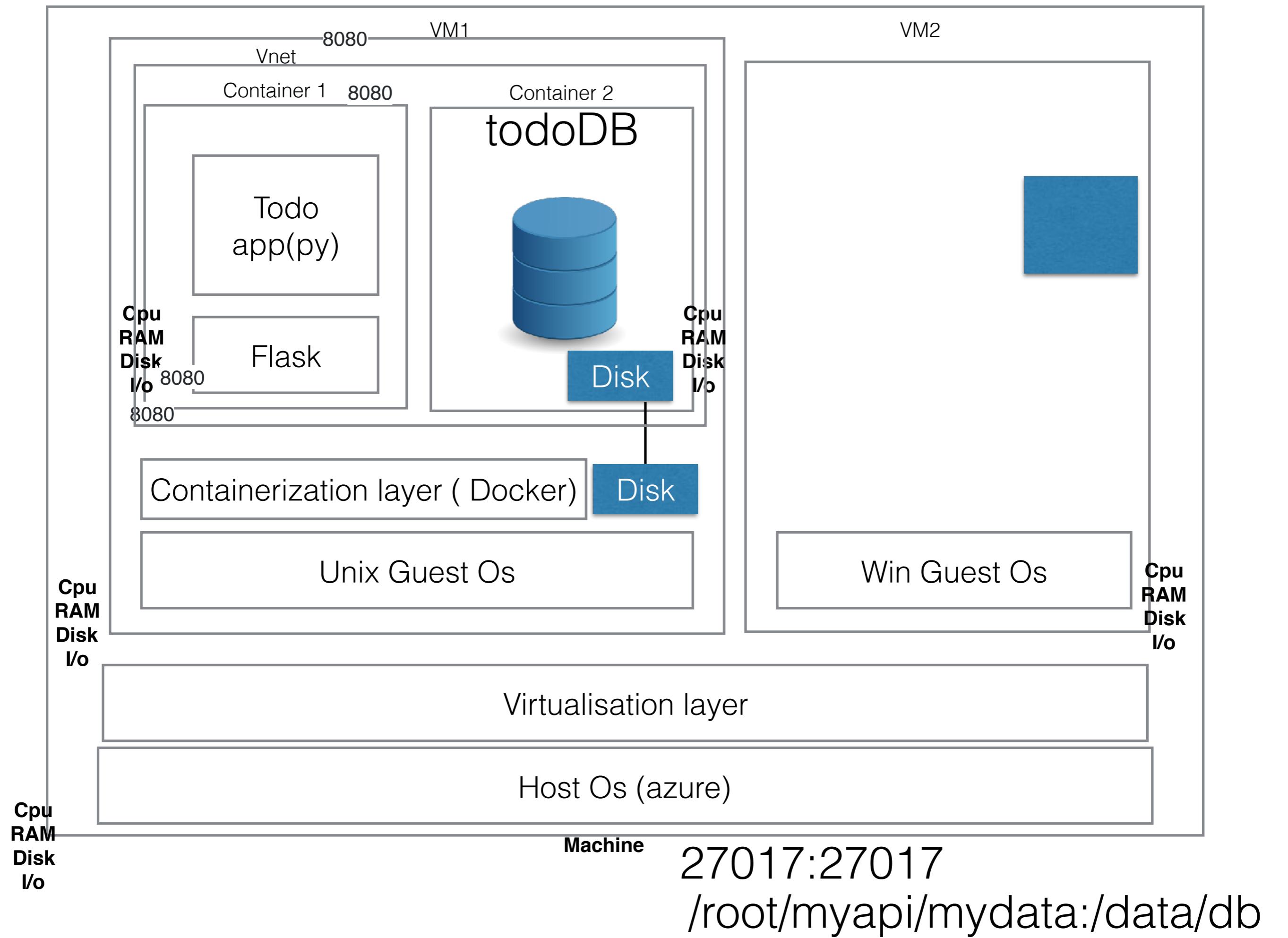


Docker Image

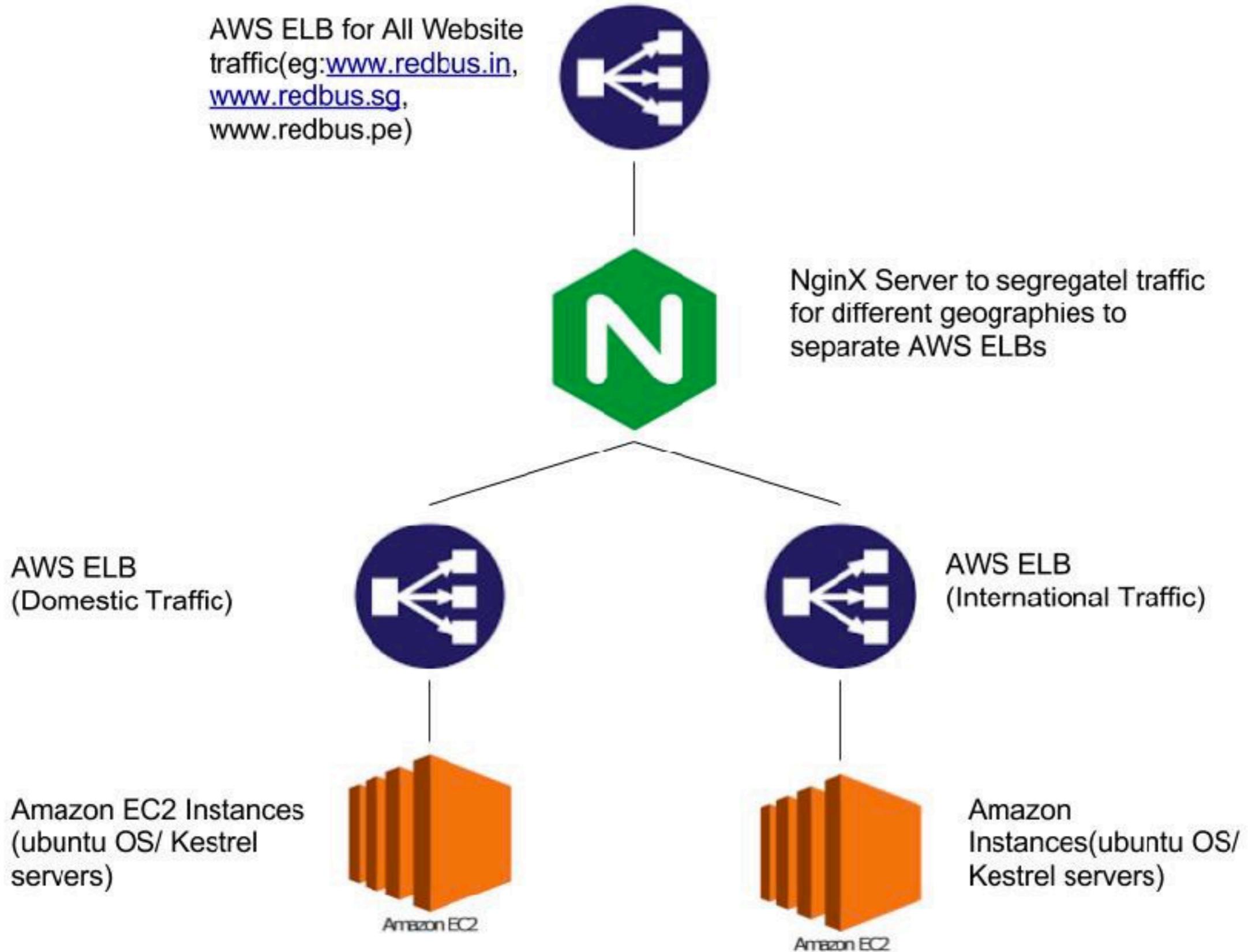


Docker /k8s

API management platform : API gateway plus a whole lot more features to create an entire API marketplace

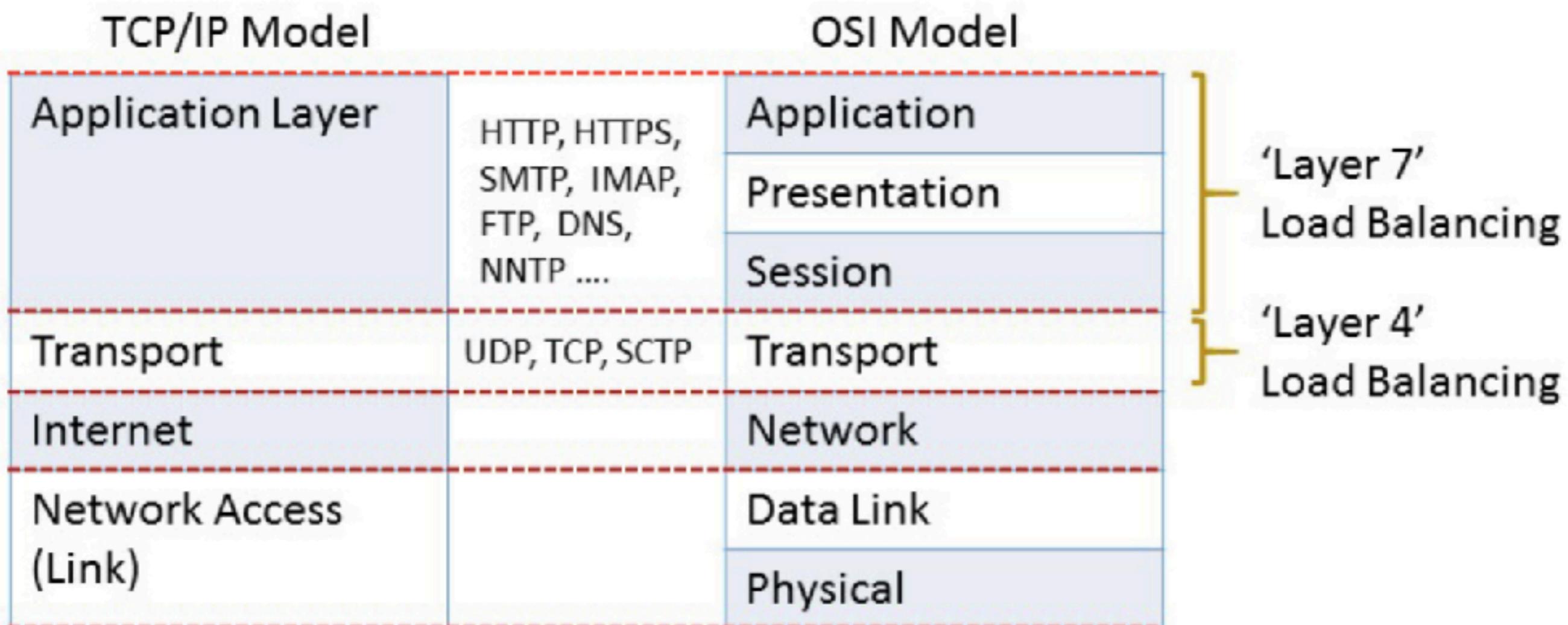


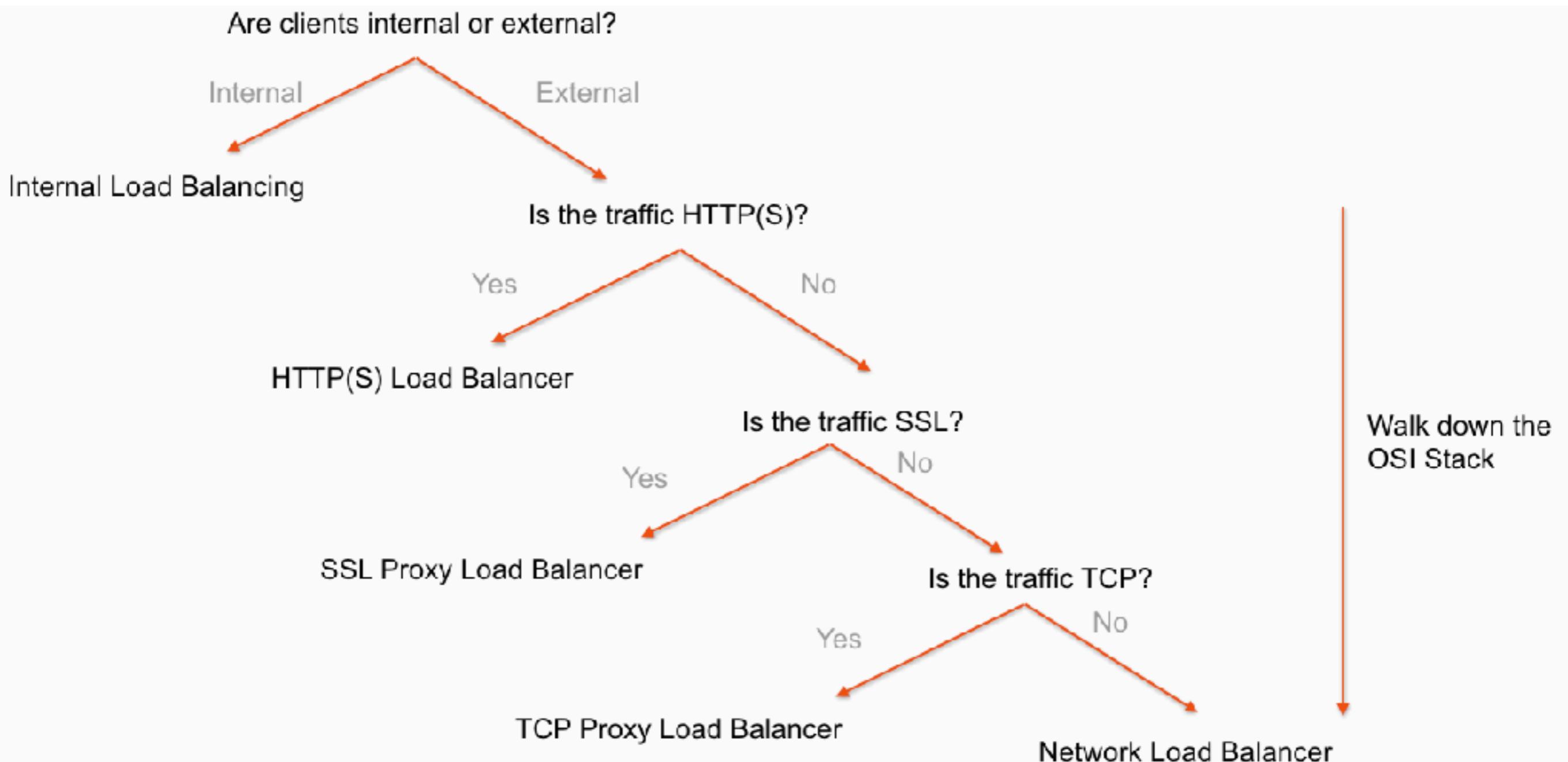
# INFRASTRUCTURE SETUP



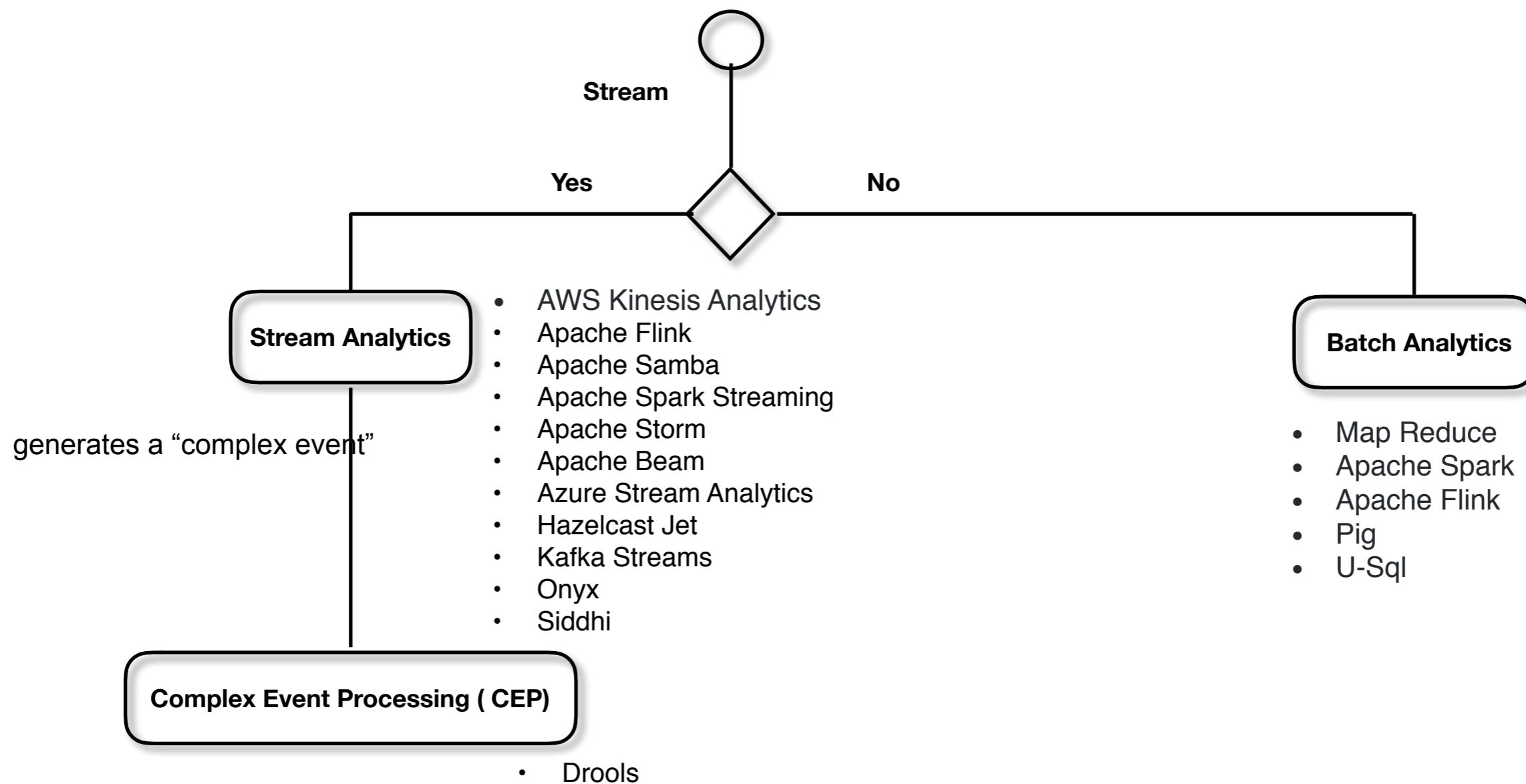
.NET framework  
MVC 4  
IIS  
windows  
[\$0.192/hr for Windows OS]  
<http://XXXX.redbus.pe/>, Throughput  
was 60.5/Sec.

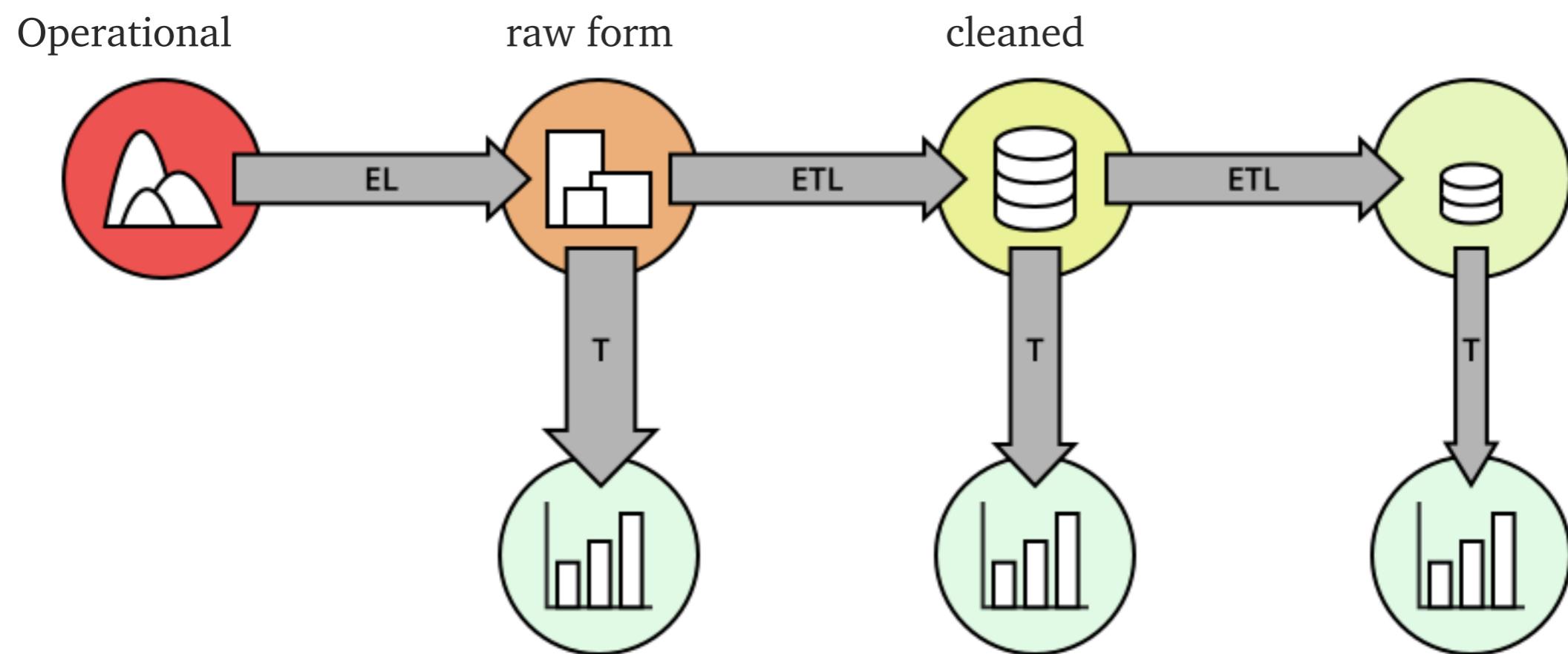
2–6 EC2s of C4 family  
depending on the traffic.  
.NET CORE  
[\$0.1/hr for Linux] 45% of cost  
<http://XXXX.redbus.pe/> ,  
Throughput was 142.9/Sec.





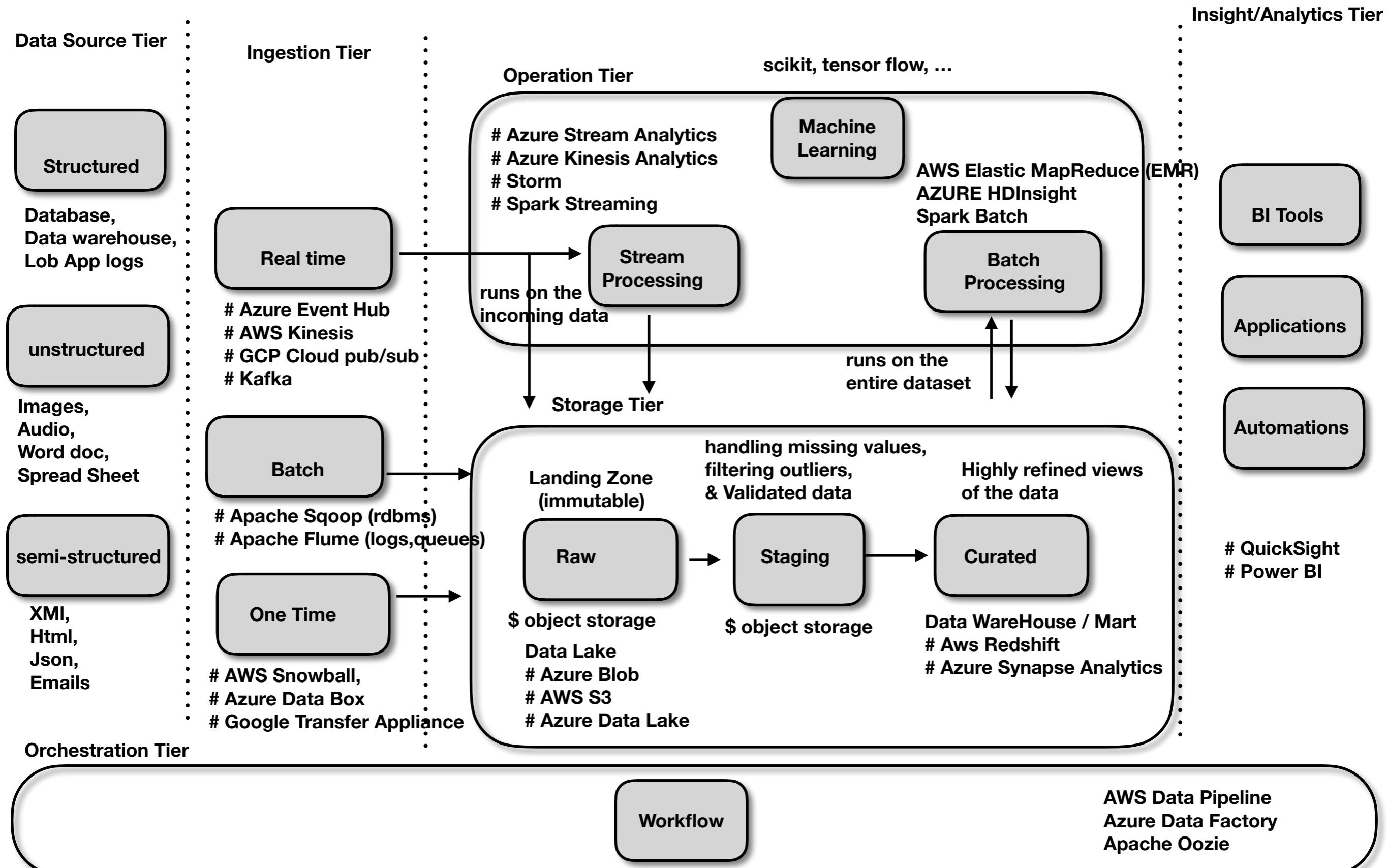
# Choose Analytical Compute



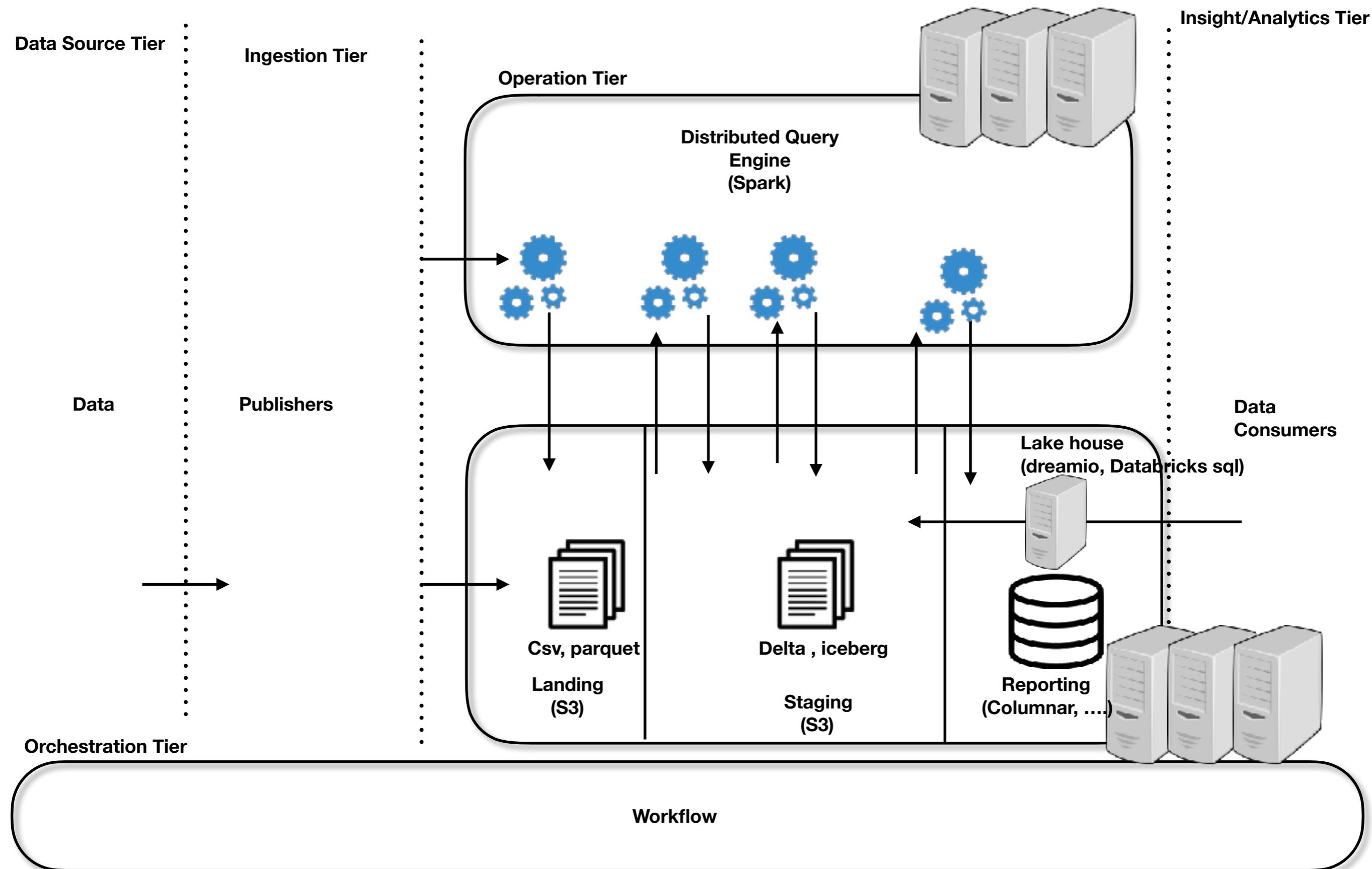


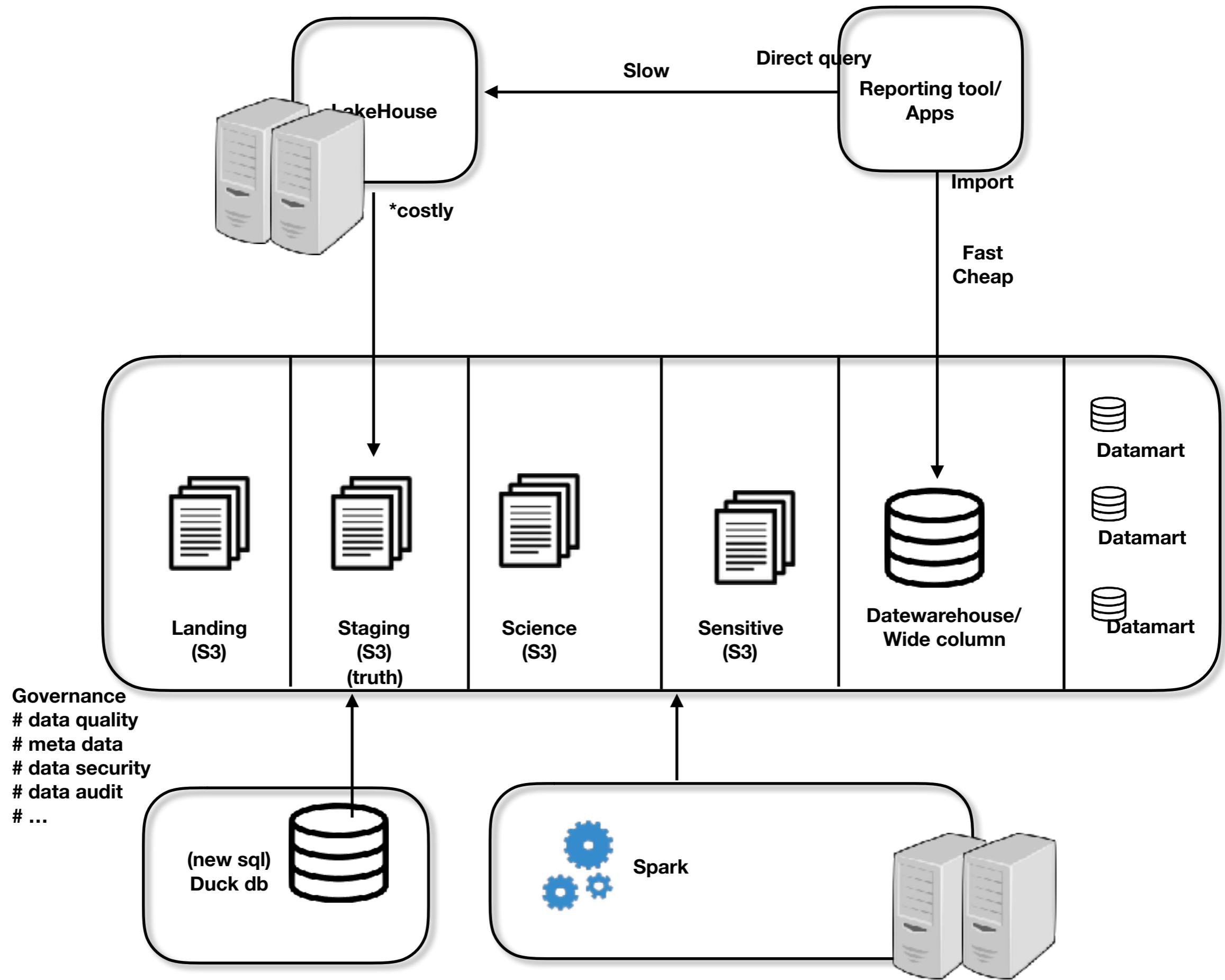
# Reference Architecture

## Analytical Application

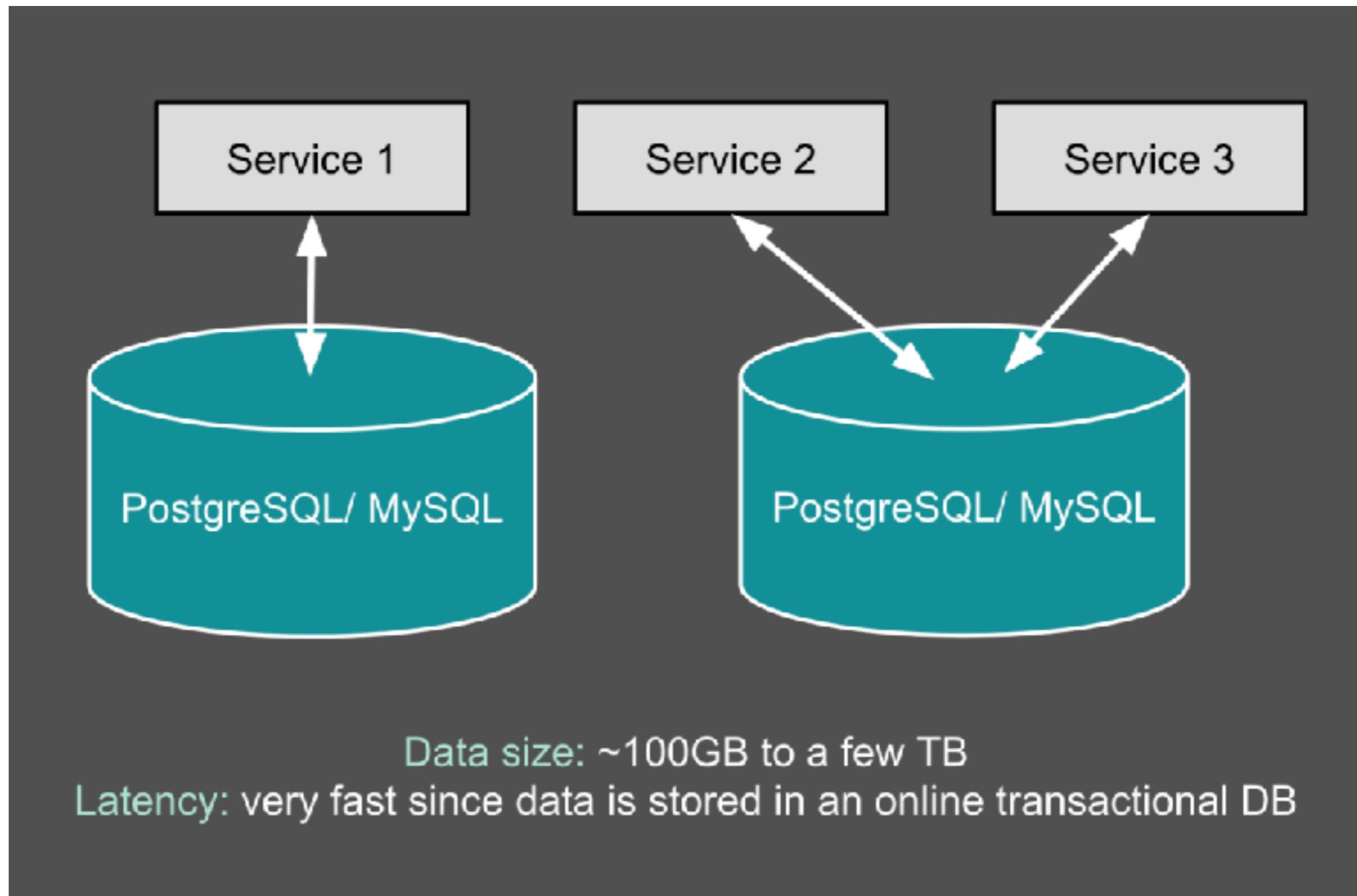


# Reference Architecture Analytical Application



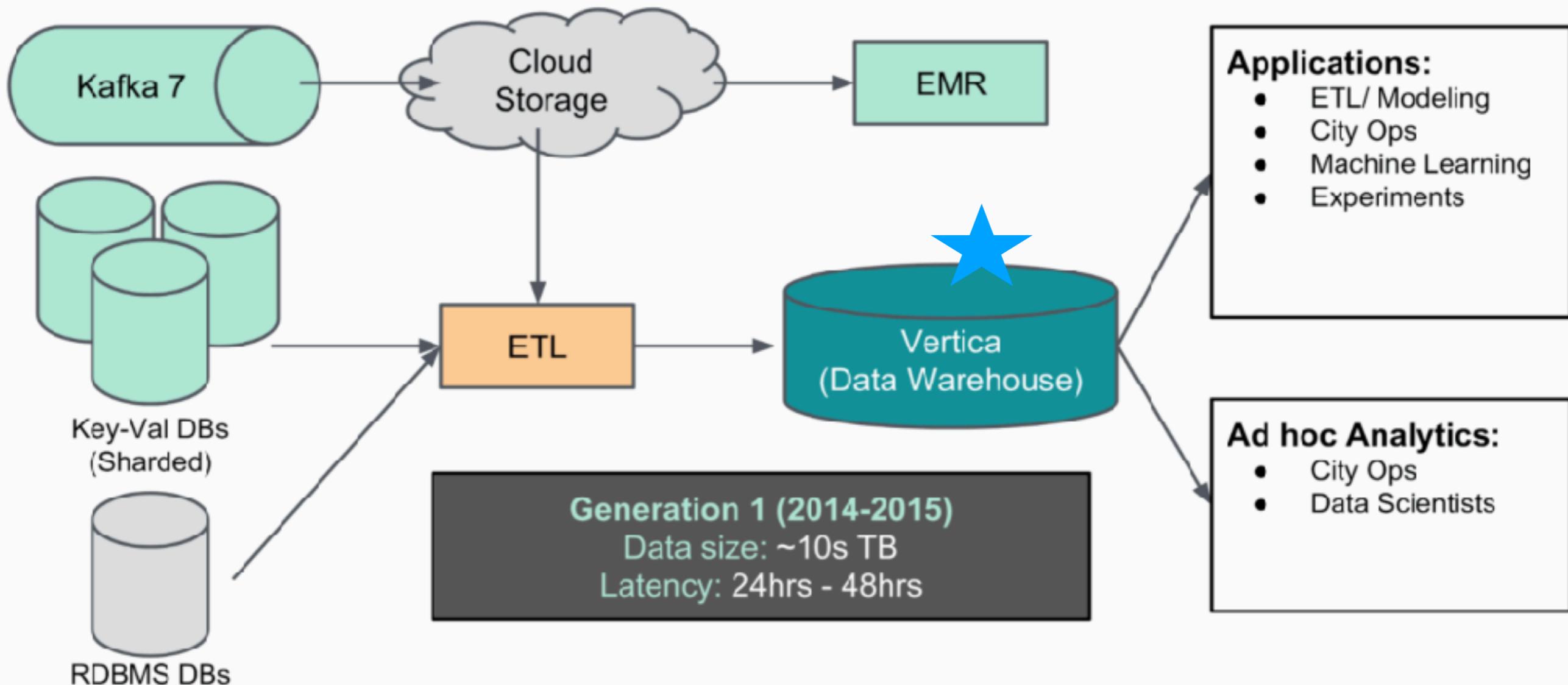


# Generation 1: The beginning of Big Data at Uber (Before 2014)



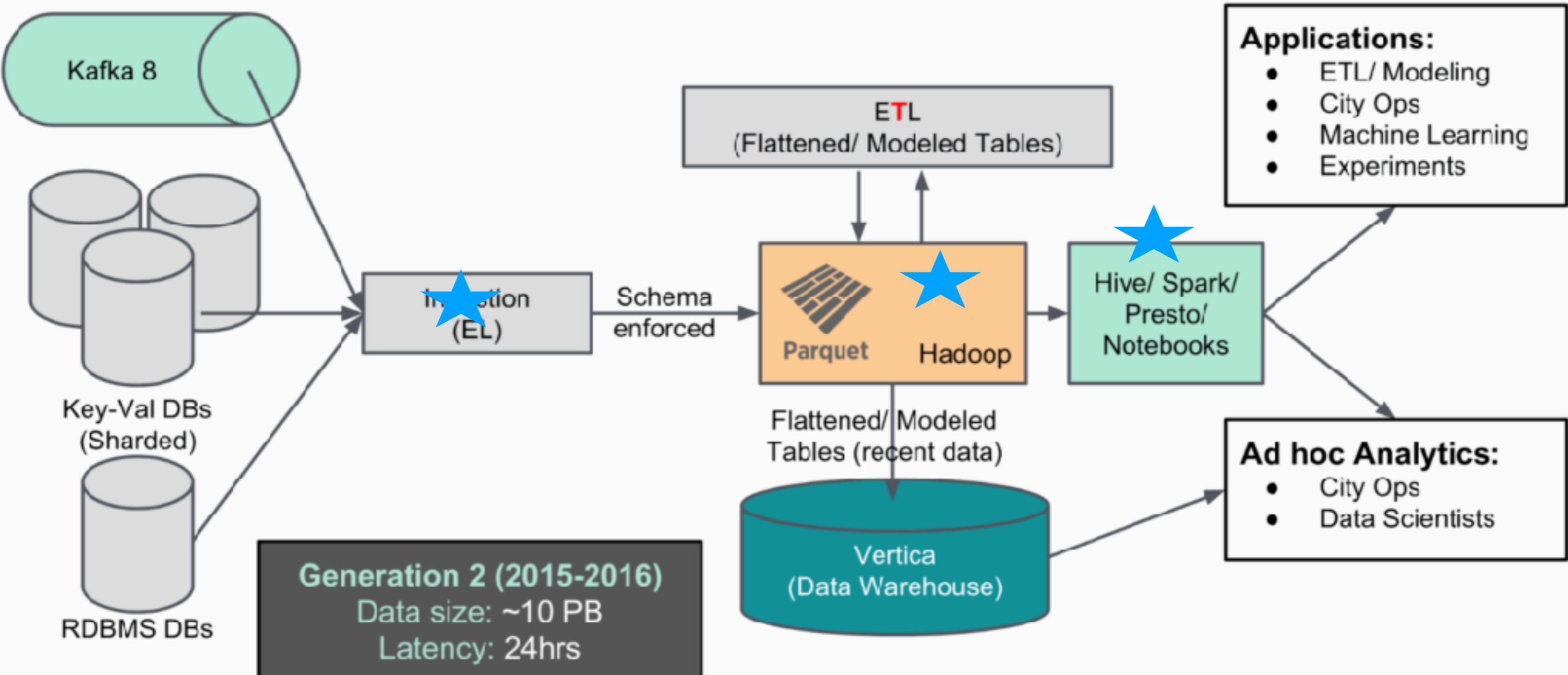
Data was scattered across different OLTP databases.

## Generation 1 (2014-2015) - The beginning of Big Data at Uber



Aggregating all of Uber's data in one place. Developed multiple ad hoc ETL jobs that copied data from different sources (i.e. AWS S3, OLTP databases, service logs, etc.) into Vertica.

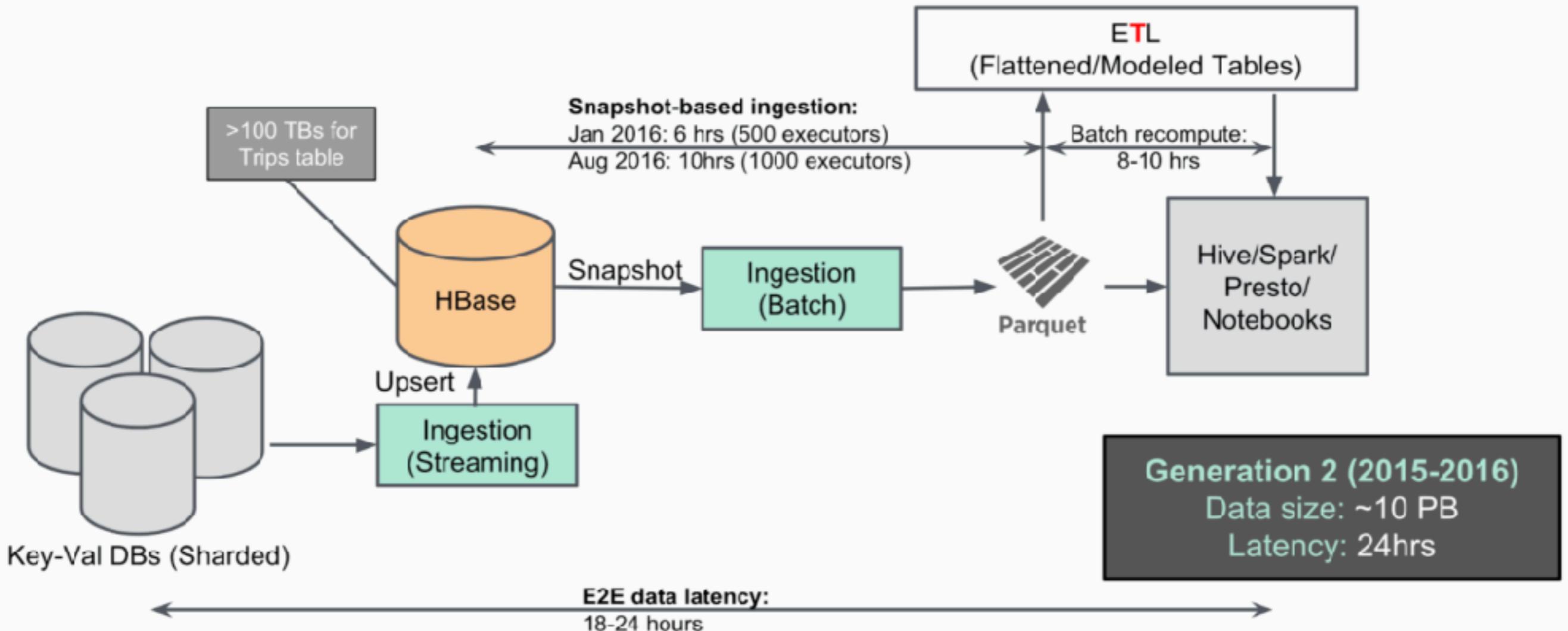
## Generation 2 (2015-2016) - The arrival of Hadoop



Data was ingested with no transformation during ingestion.

## Generation 2 (2015-2016) - The arrival of Hadoop

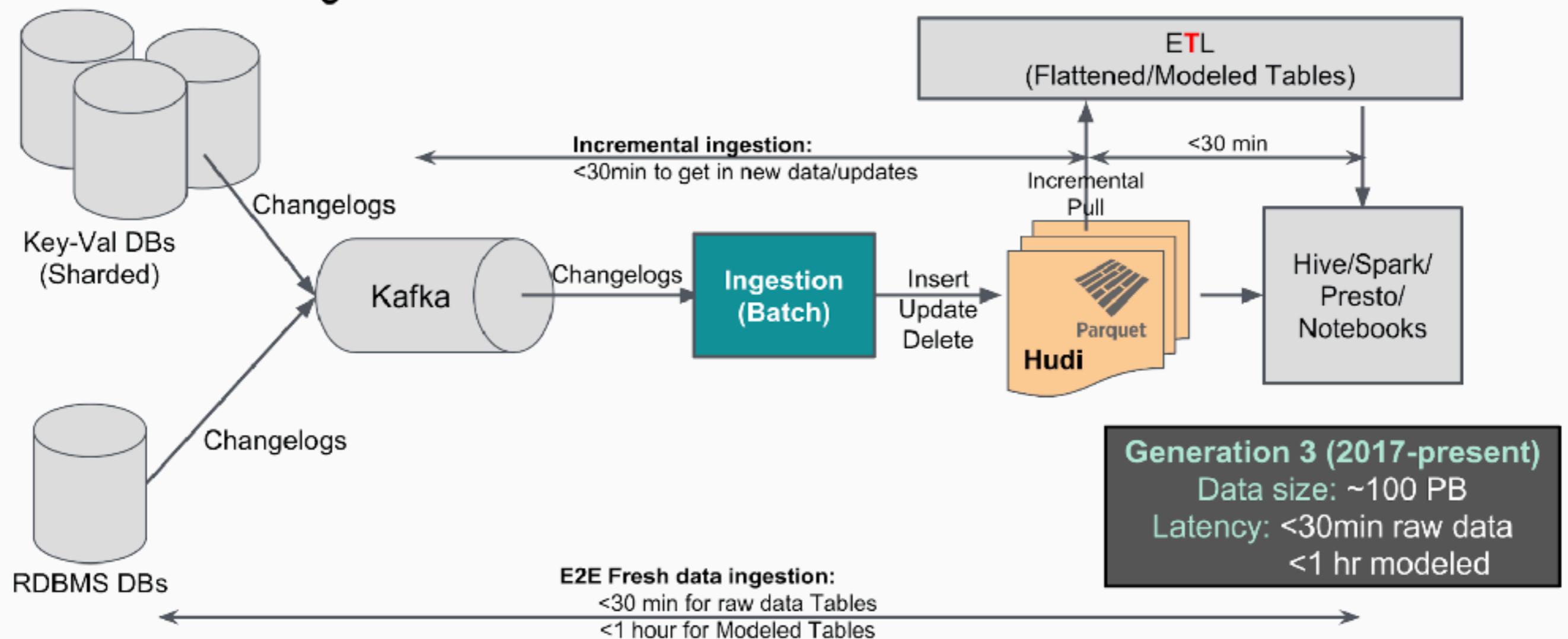
Why does data latency remain at 24 hours?



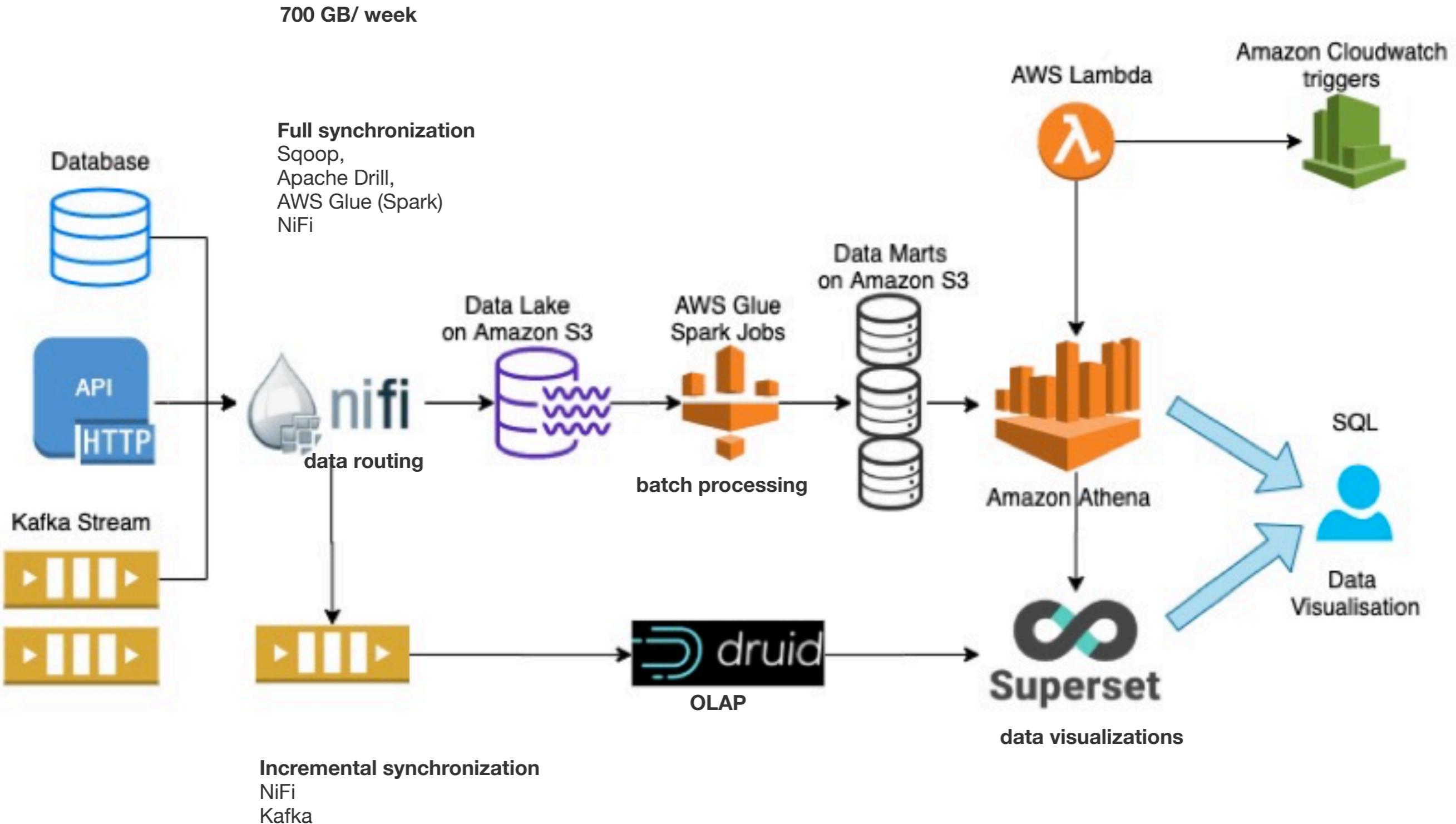
only over 100 gb of new data was added every day. Since HDFS and Parquet do not support data updates, each run of the ingestion job had to convert the entire, over 100 tb dataset for that specific table.

## Generation 3 (2017-present) - Let's rebuild for long term

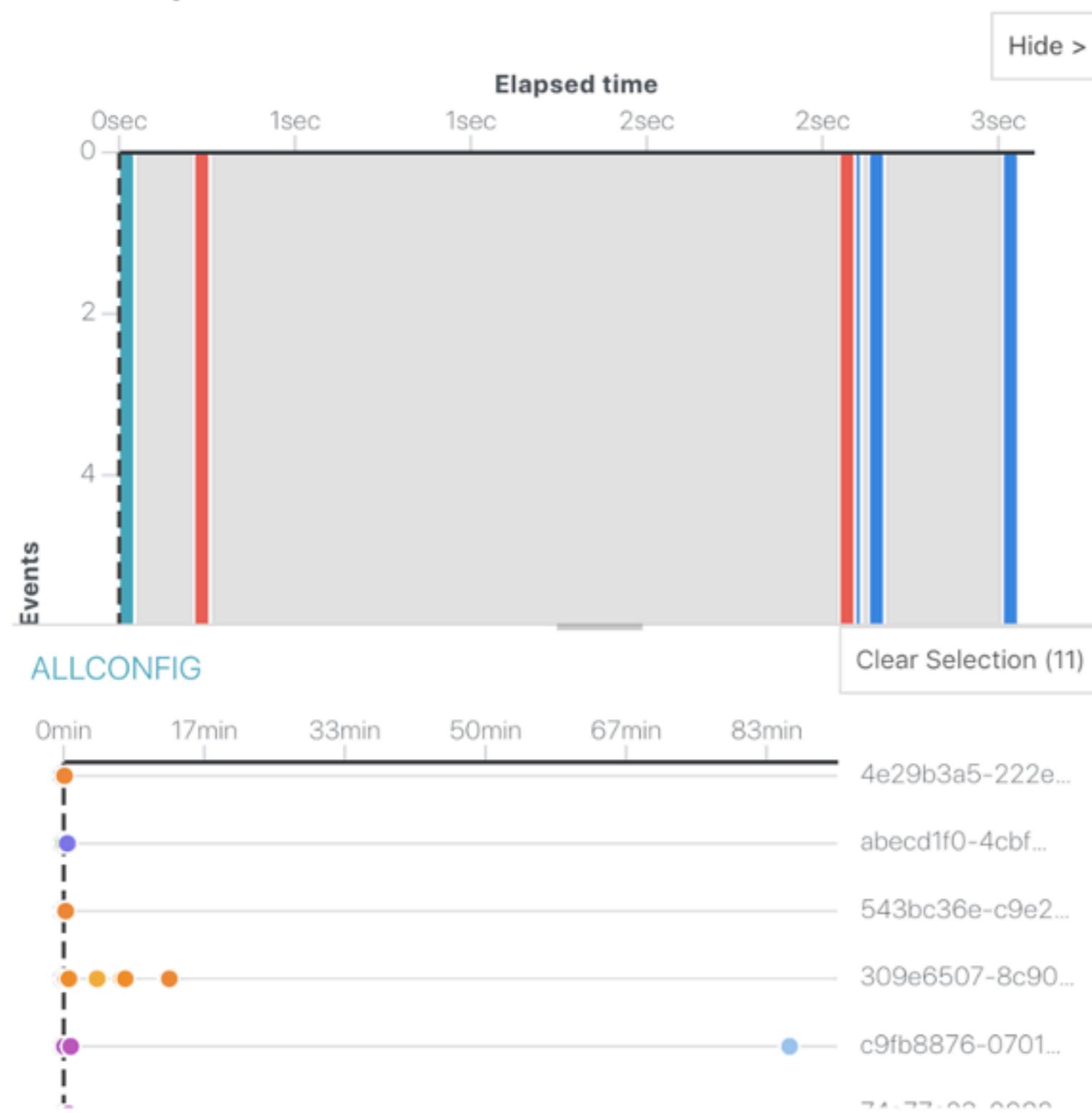
### Incremental ingestion:



# Redbus



## Event Sequence

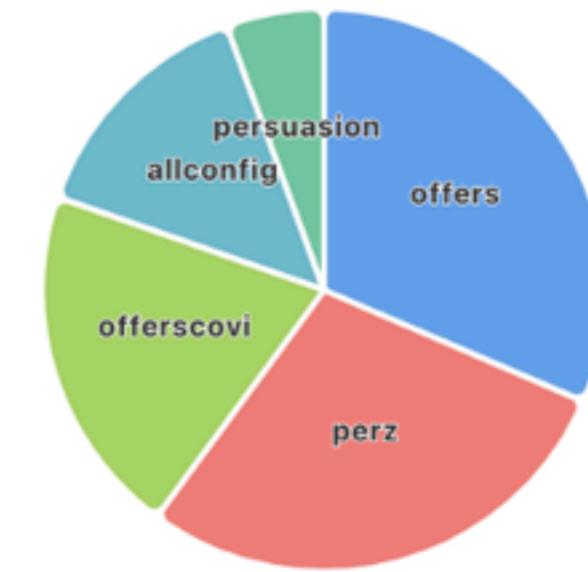


### Align sequences by

1st - + event

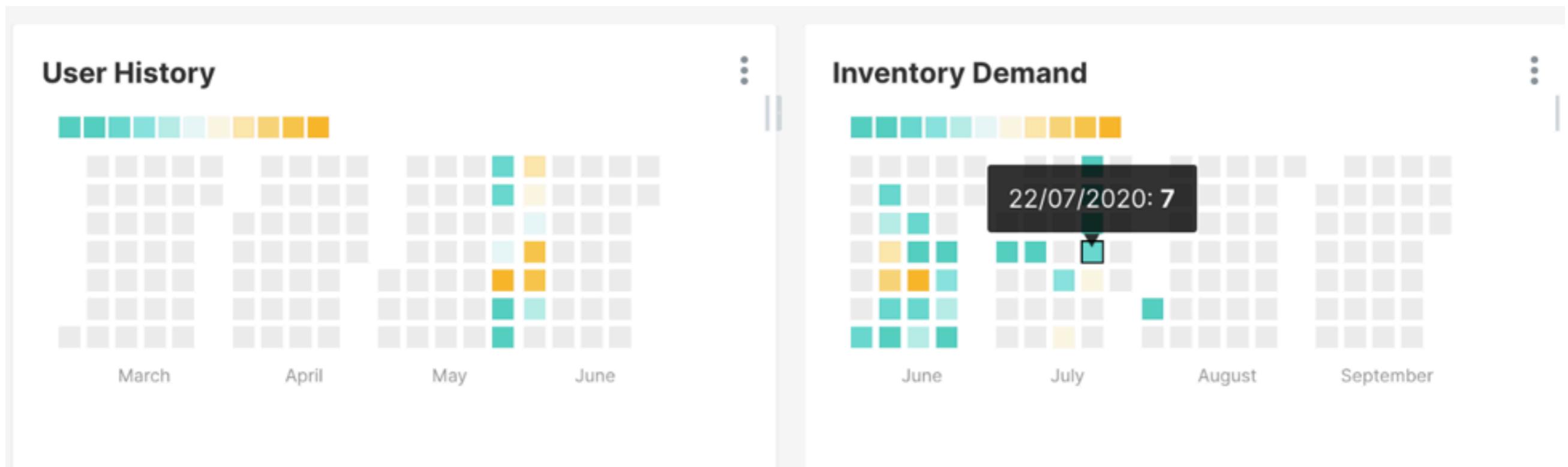
### Event type summary

78 events (427 hidden [84.6%])

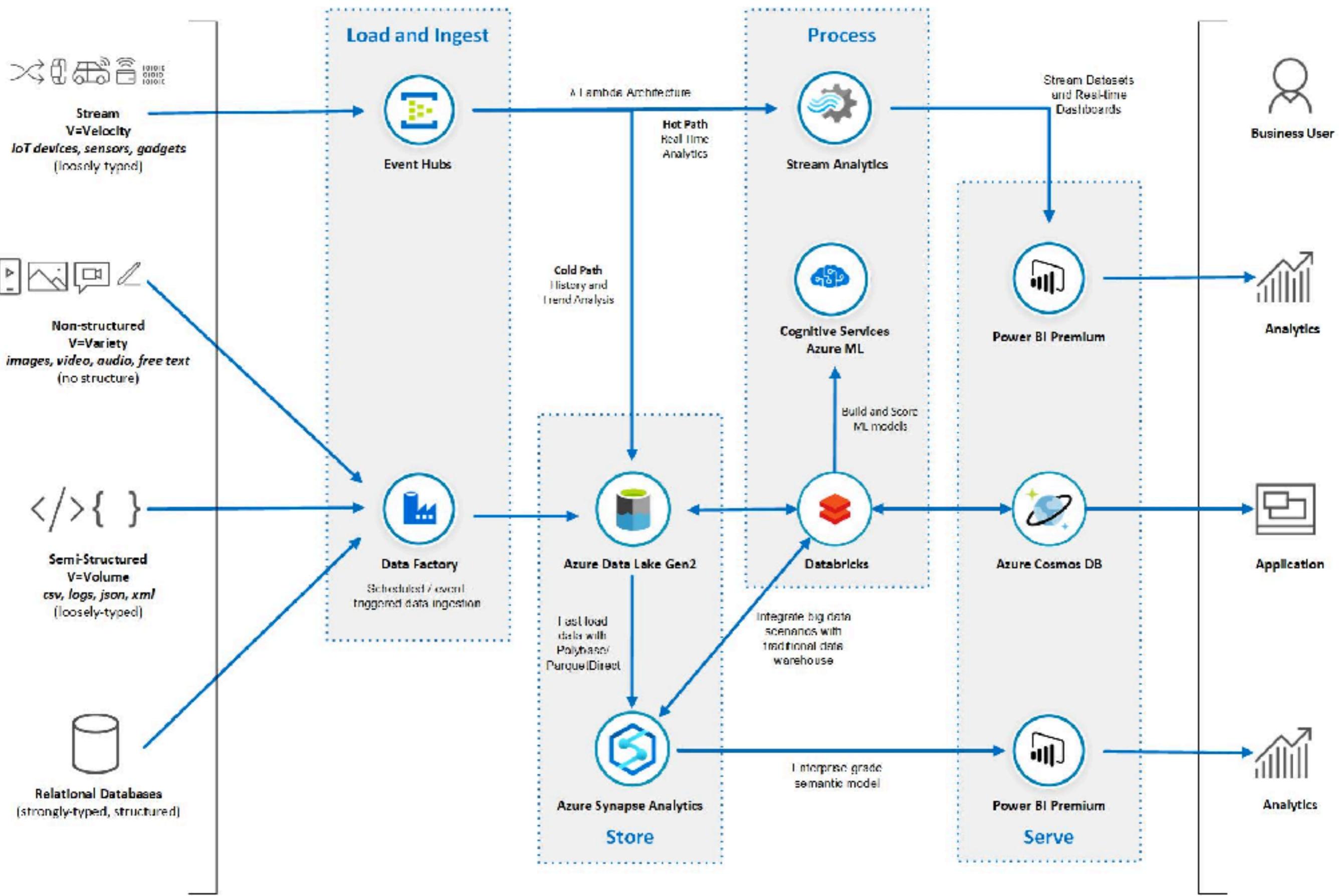


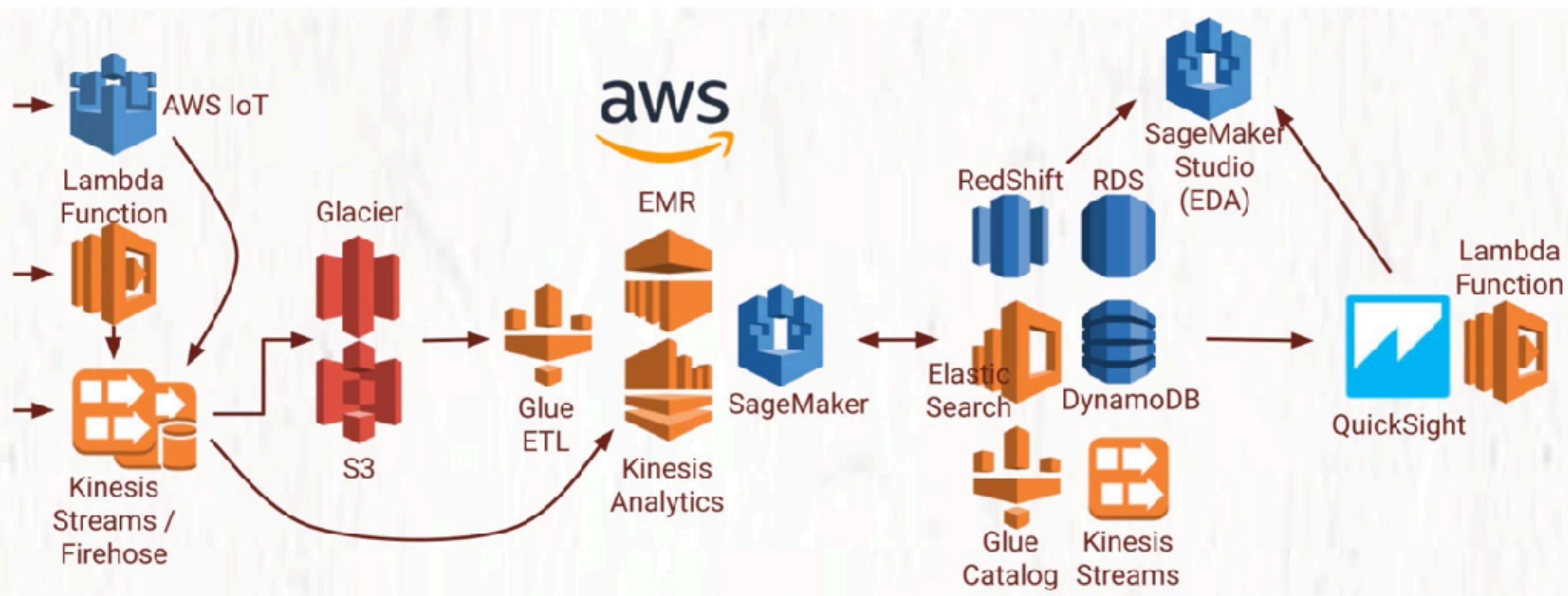
- allconfig (14.1%)
- busdetails
- busroutes
- filters
- offers (35.9%)
- offerscovid (23.1%)
- persuasion
- perz (26.9%)
- rides
- seatlayout
- signin

## Calendar heat-map

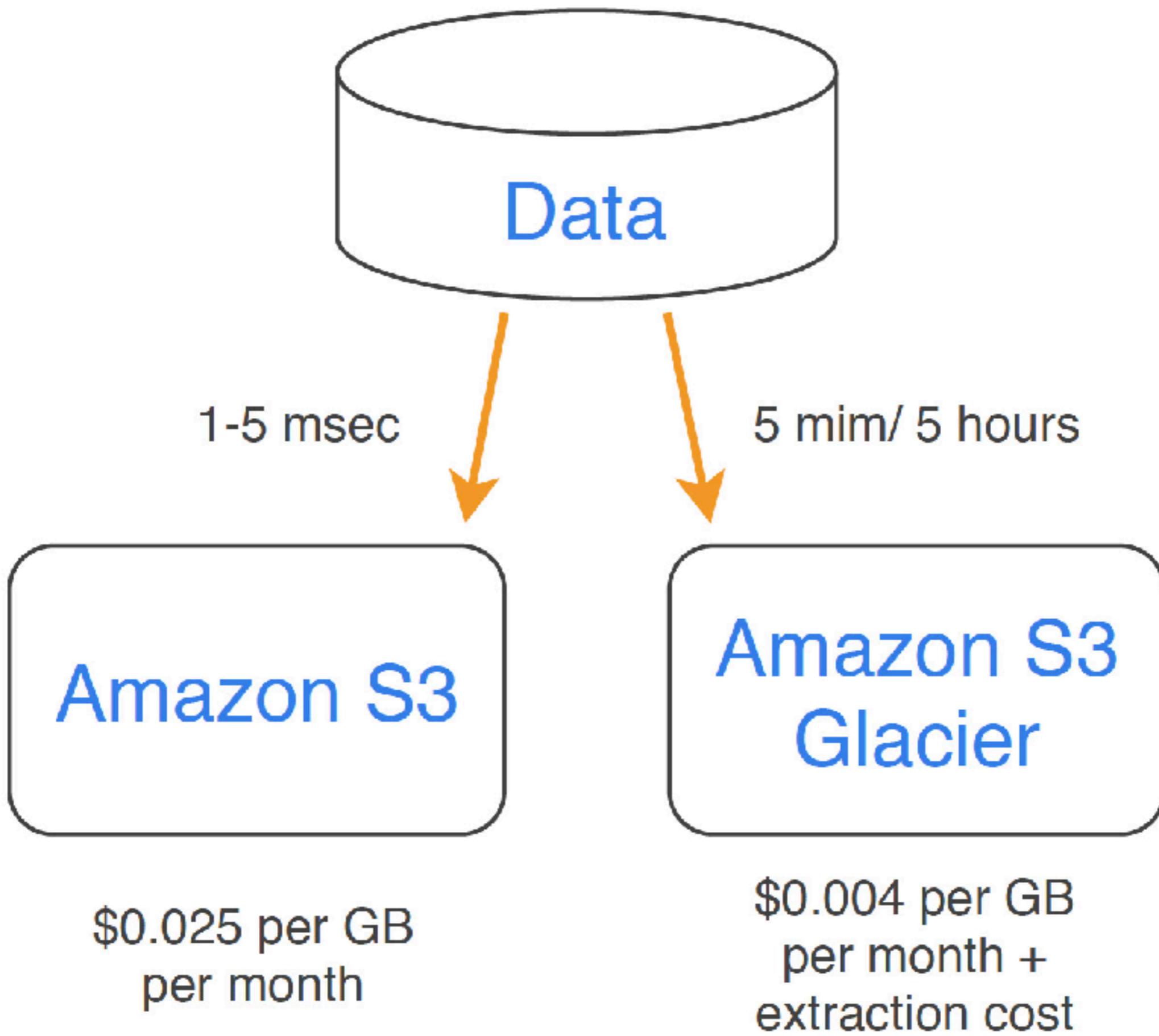


## Modern Data Platform Reference Architecture





		Kinesis Data Streams	Kinesis Firehose
Purpose	Low latency streaming service for ingest at scale	Data transfer service to load streaming data into Amazon S3, Redshift, Elasticsearch & Splunk	
Provisioning	Managed service but needs configuration for shards.	Fully managed service, no administration.	
Processing	Real Time (~200ms latency for classic, ~70ms for enhanced fan out)	Near real time (depends on buffer size OR buffer time min. 60 secs)	
Scaling	Must manage scaling (configure shards)	Automated Scaling – as per the demand	
Data Storage	Configurable from 1 to 7 days	Does not provide data storage	
Replay capability	Supports replay capability	Does not support replay capability	
Producers	Need to write code for producer. Supports SDK, Kinesis Agent, KPL, CloudWatch, IoT	Need to write code for producer. Supports KPL, Kinesis Agent, Data Streams, CloudWatch, IoT	
Consumers	Open ended. Supports multiple consumers and destinations. Supports KCL and Spark.	Closed ended. Handled by Firehose. Does not support KCL or Spark.	



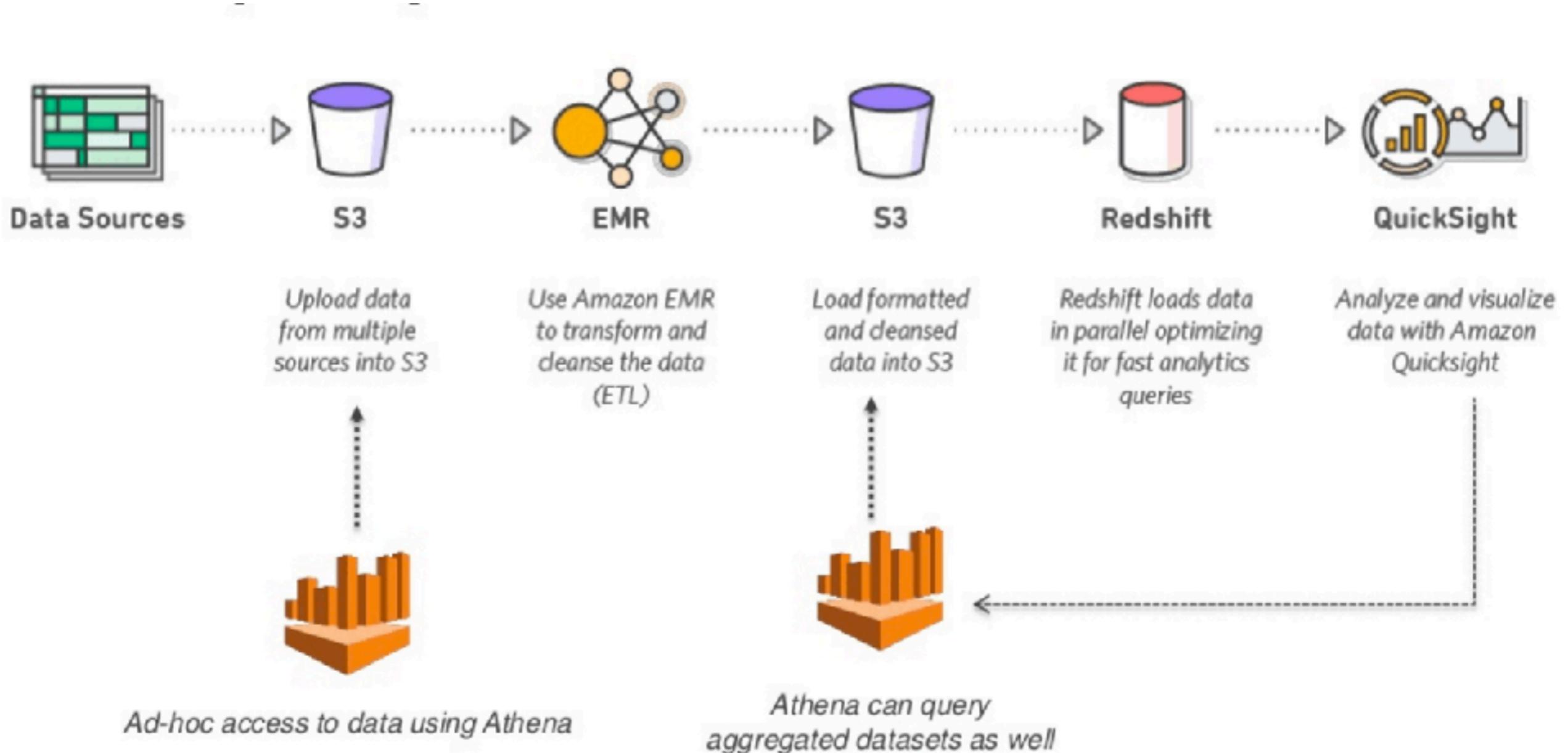
	<b>S3</b>	<b>S3 Glacier</b>
Access speed	Hi	Low
API availability	Yes	Yes
Data storage cost	Relatively high	Very low
Object size	Up to 5 Tb	40 Tb
Accommodation in regions	Many regions	Average
Static Web Content	Yes	No
Supporting Versioning	Yes	No

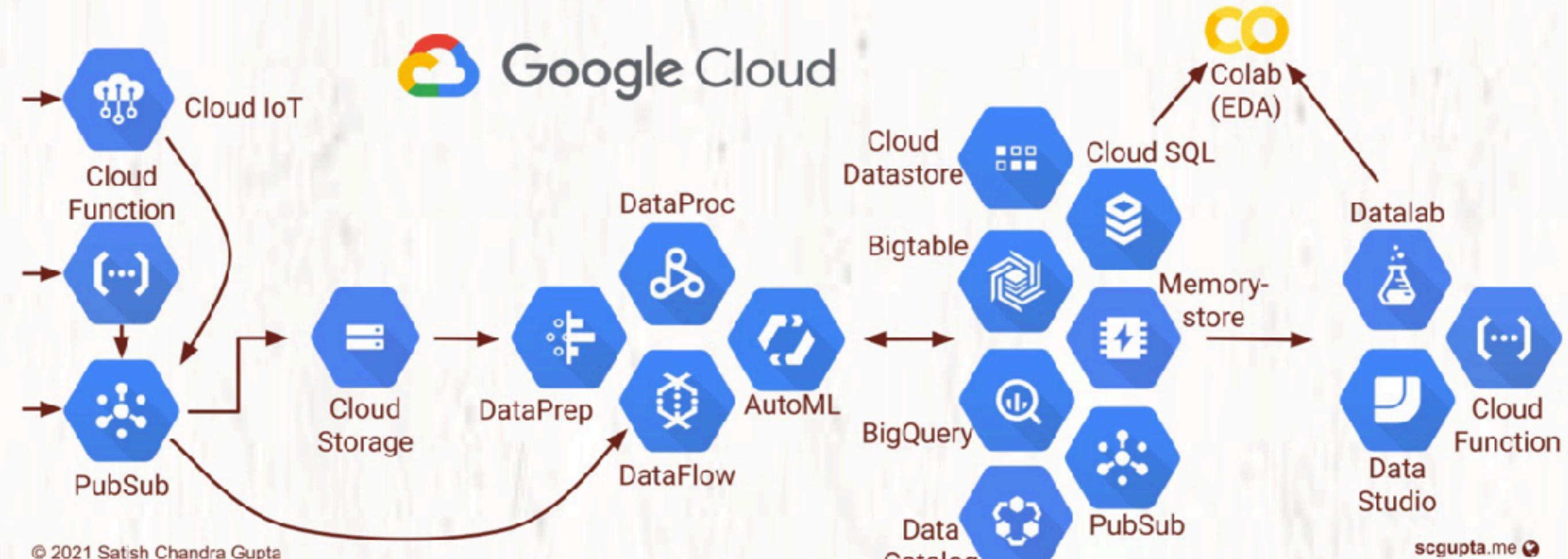
	<b>Data Streams</b>	<b>Data Firehose</b>	<b>Data Analytics</b>	<b>Video Streams</b>
<b>Short definition</b>	Scalable and durable real-time data streaming service.	Capture, transform, and deliver streaming data into data lakes, data stores, and analytics services.	Transform and analyze streaming data in real time with Apache Flink.	Stream video from connected devices to AWS for analytics, machine learning, playback, and other processing.
<b>Data sources</b>	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Amazon MSK, Amazon Kinesis Data Streams, servers, mobile devices, IoT devices, etc.	Any streaming device that supports Kinesis Video Streams SDK.
<b>Data consumers</b>	Kinesis Data Analytics, Amazon EMR, Amazon EC2, AWS Lambda	Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, generic HTTP endpoints, Datadog, New Relic, MongoDB, and Splunk	Analysis results can be sent to another Kinesis stream, a Kinesis Data Firehose delivery stream, or a Lambda function	Amazon Rekognition, Amazon SageMaker, MxNet, TensorFlow, HLS-based media playback, custom media processing application
<b>Use cases</b>	<ul style="list-style-type: none"> <li>– Log and event data collection</li> <li>– Real-time analytics</li> <li>– Mobile data capture</li> <li>– Gaming data feed</li> </ul>	<ul style="list-style-type: none"> <li>– IoT Analytics</li> <li>– Clickstream Analytics</li> <li>– Log Analytics</li> <li>– Security monitoring</li> </ul>	<ul style="list-style-type: none"> <li>– Streaming ETL</li> <li>– Real-time analytics</li> <li>– Stateful event processing</li> </ul>	<ul style="list-style-type: none"> <li>– Smart technologies</li> <li>– Video-related AI/ML</li> <li>– Video processing</li> </ul>

S. Amazon DynamoDB  
No.

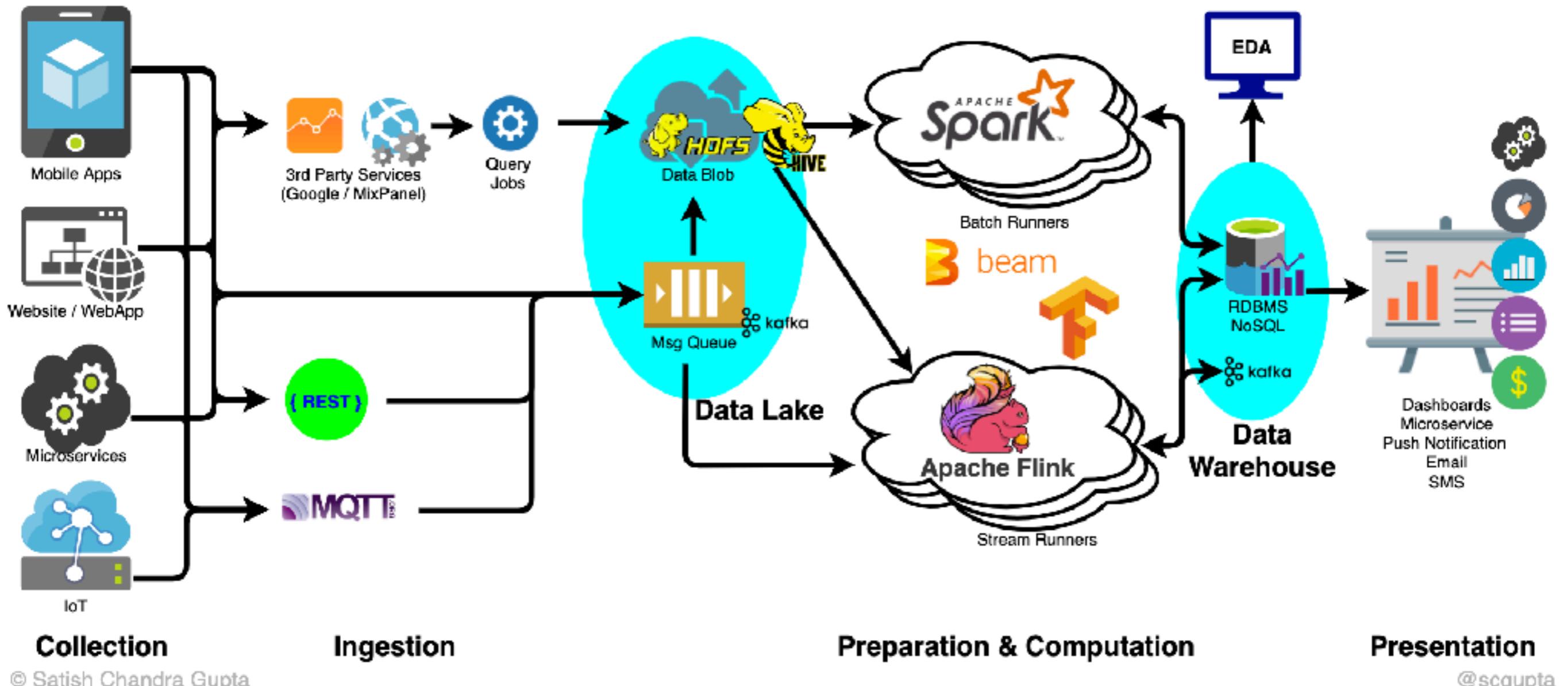
Amazon Redshift

1 It was developed by Amazon in 2012.	It was developed by Amazon in 2012.
2 It is hosted, scalable database service by Amazon with data stored in Amazon cloud.	It is large scale data warehouse service for use with business intelligence tools.
3 It does not support SQL query language.	It supports SQL query language. But it does not fully support an SQL-standard.
4 It does not provide concept of Referential Integrity. Hence, no Foreign Keys.	It provides concept of Referential Integrity. Hence, there are Foreign Keys.
5 Its Primary database models are Document store and Key-value store.	Its primary database model is Relational DBMS.
6 It does not support Server-side scripting.	It supports user-defined functions for Server-side scripting in python.
7 Eventual Consistency and Immediate Consistency are used to ensure consistency in distributed system.	Immediate Consistency is used to ensure consistency in distributed system.
8 It does not offer API for user-defined Map/Reduce methods. But maybe implemented via Amazon Elastic MapReduce.	It does not offer API for user-defined Map/Reduce methods.
9 It supports secondary indexes.	It supports restricted secondary indexes.

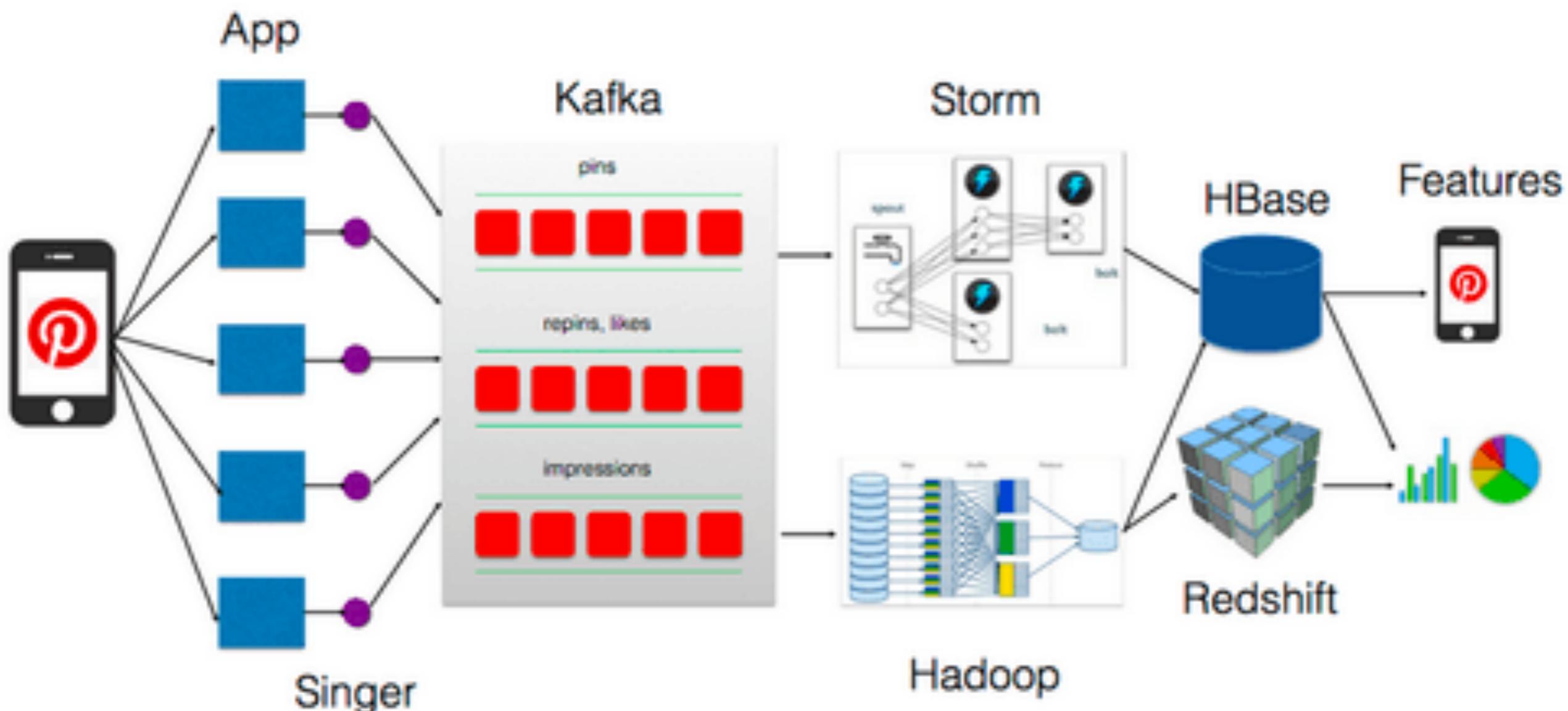




# Open Source Data analytics Architecture

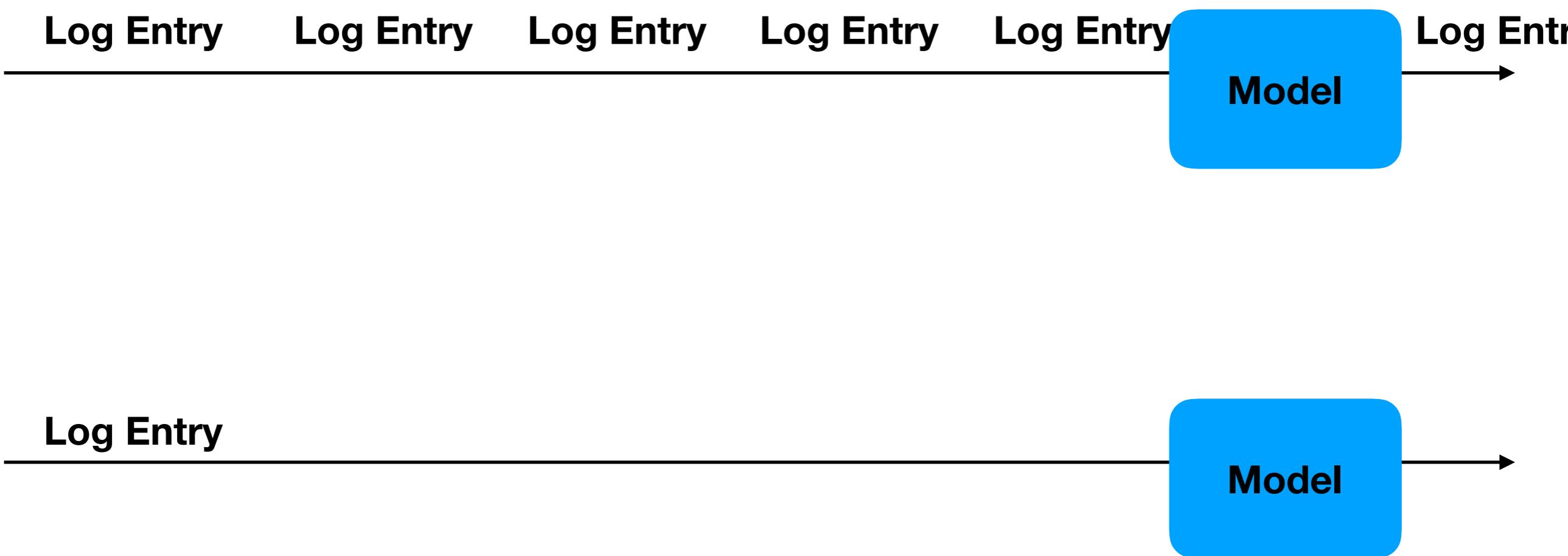


# Data analytics Architecture adopted by Pinterest

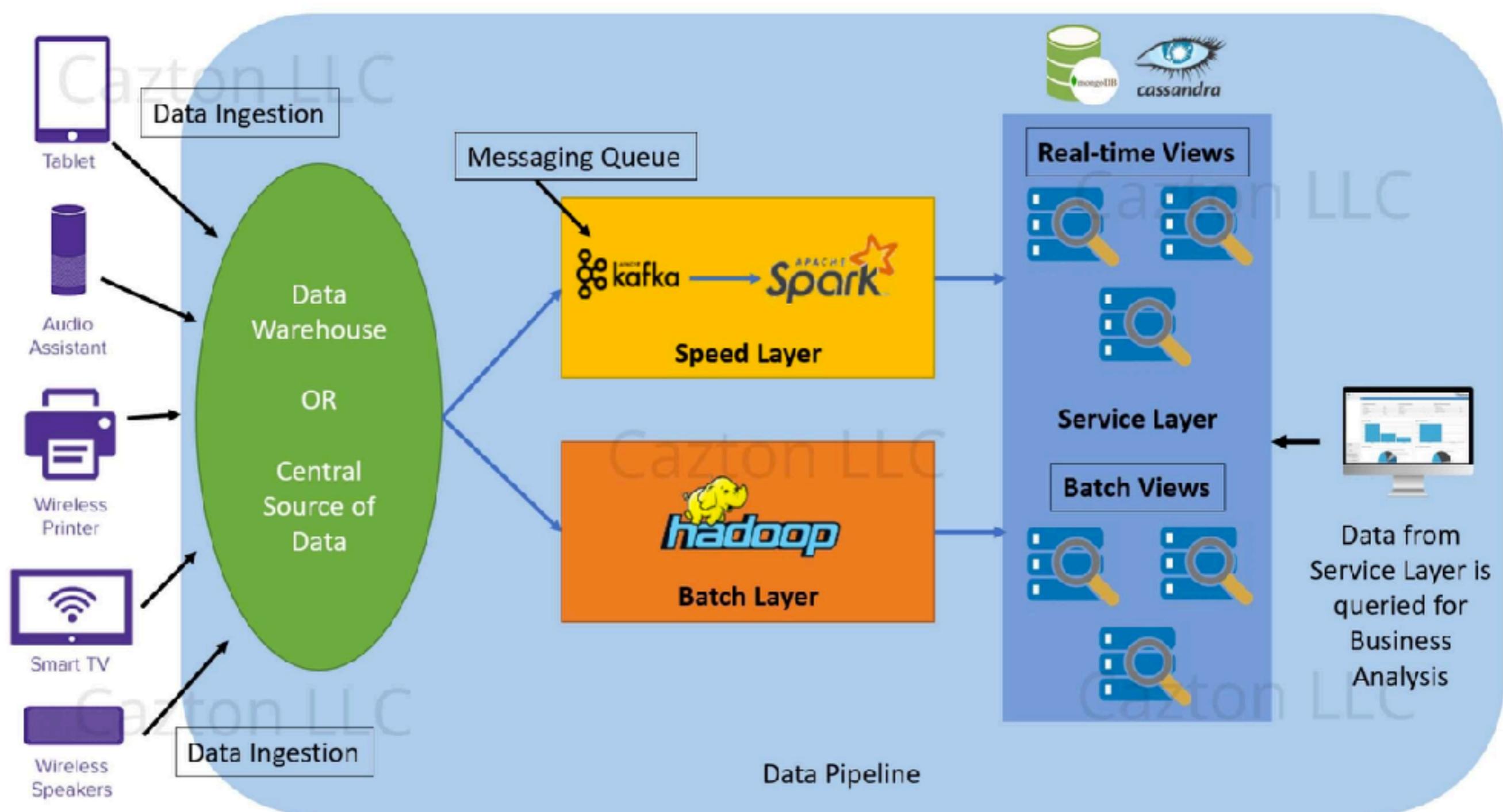


*Data Architecture overview*

# Welldoc

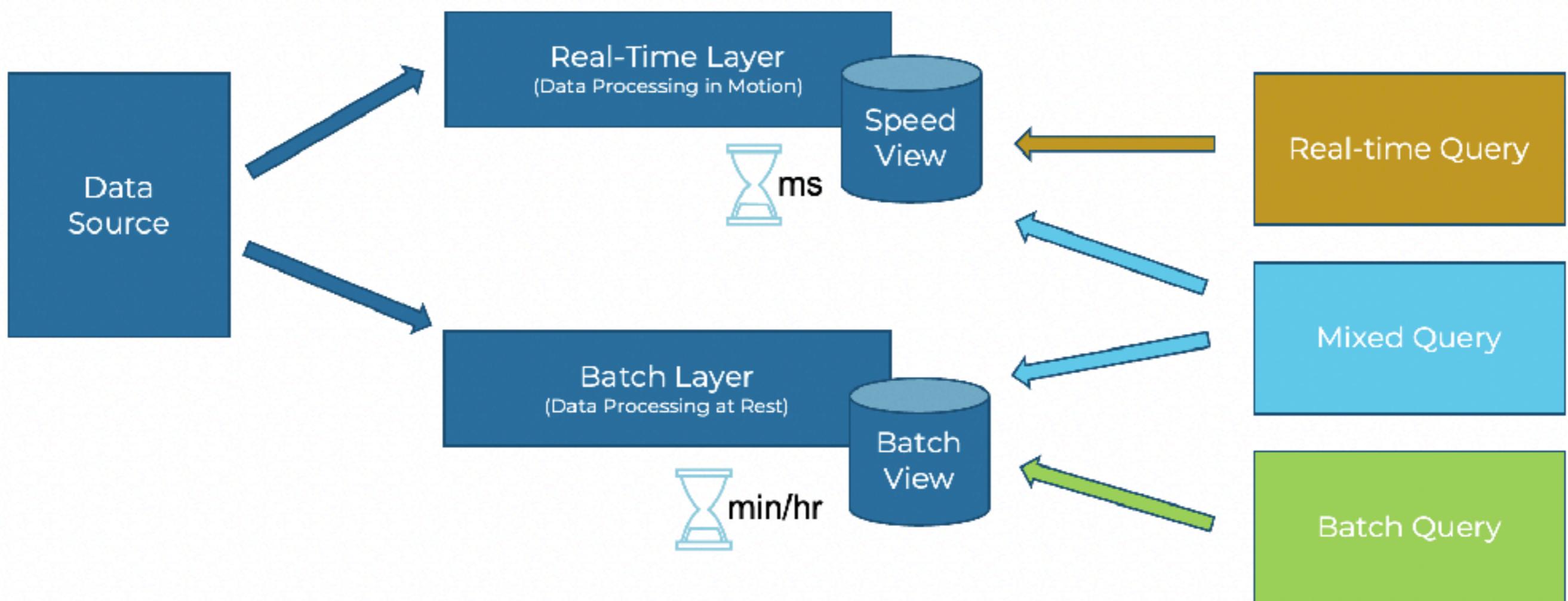


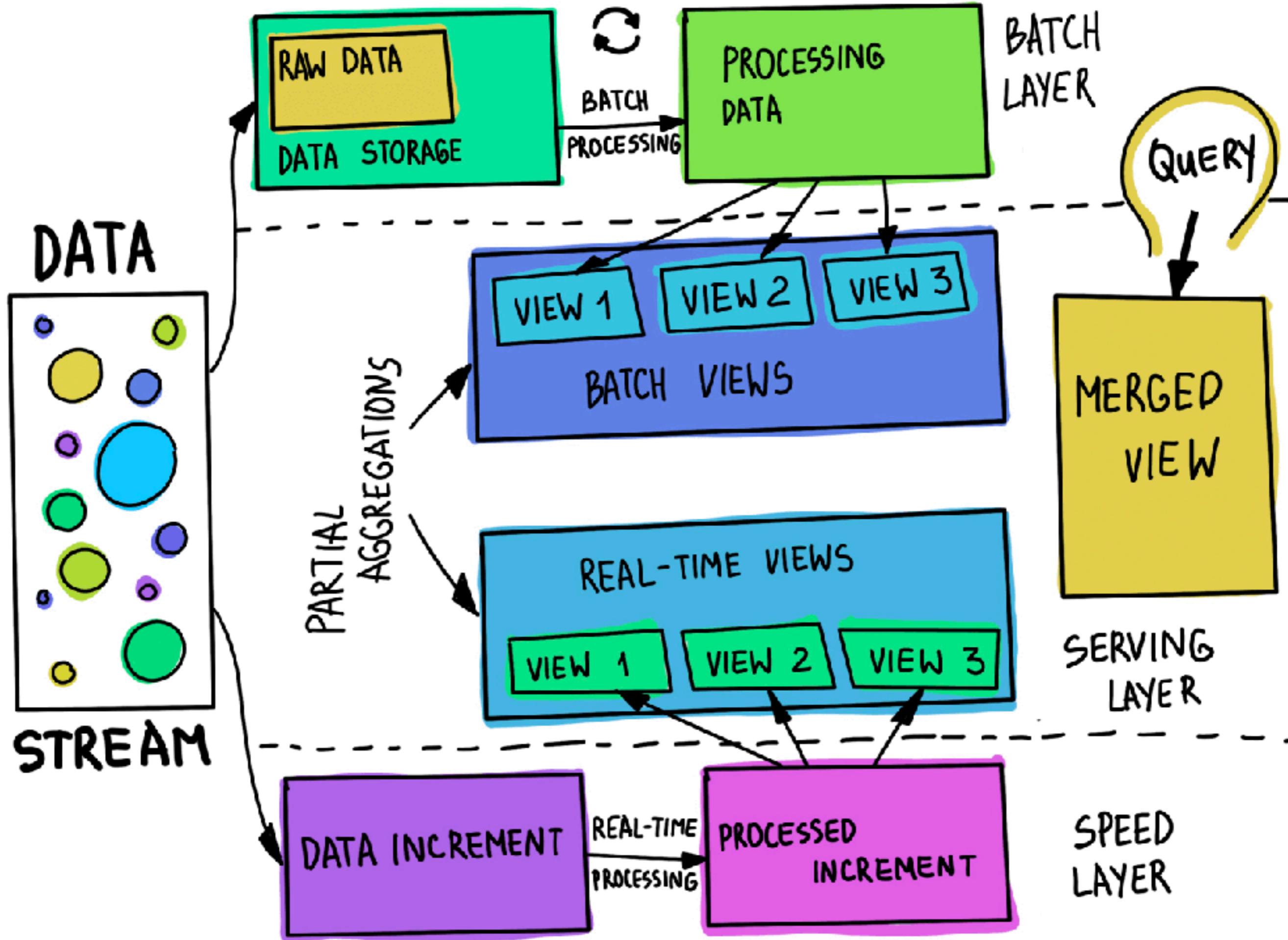
# Lambda Architecture



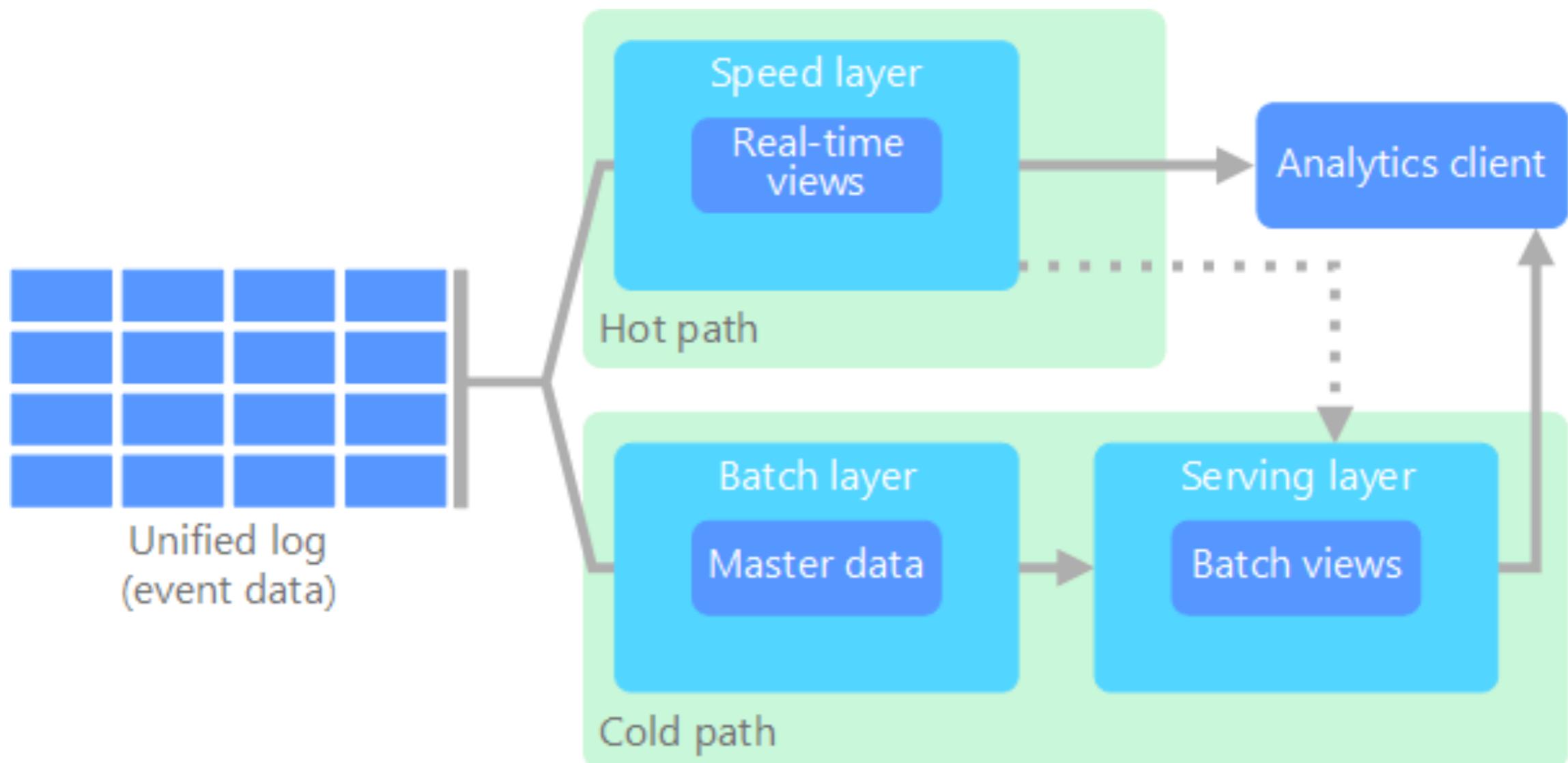
# Lambda Architecture

## Option 2: Separate serving layers



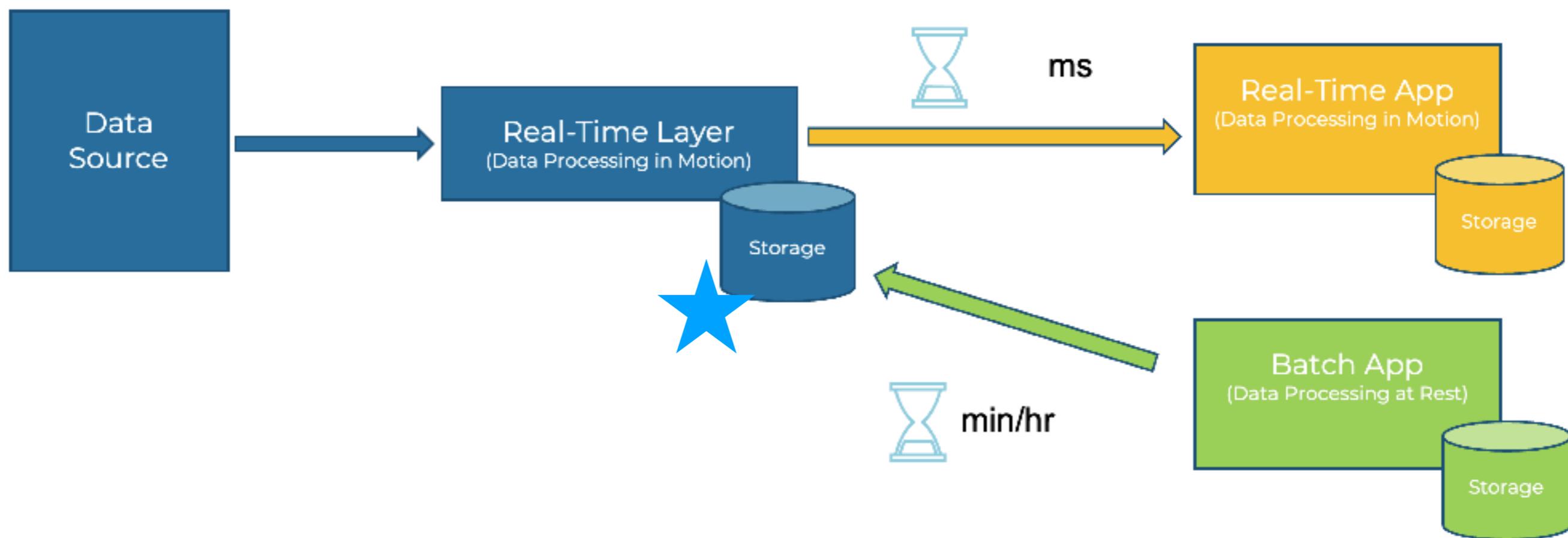


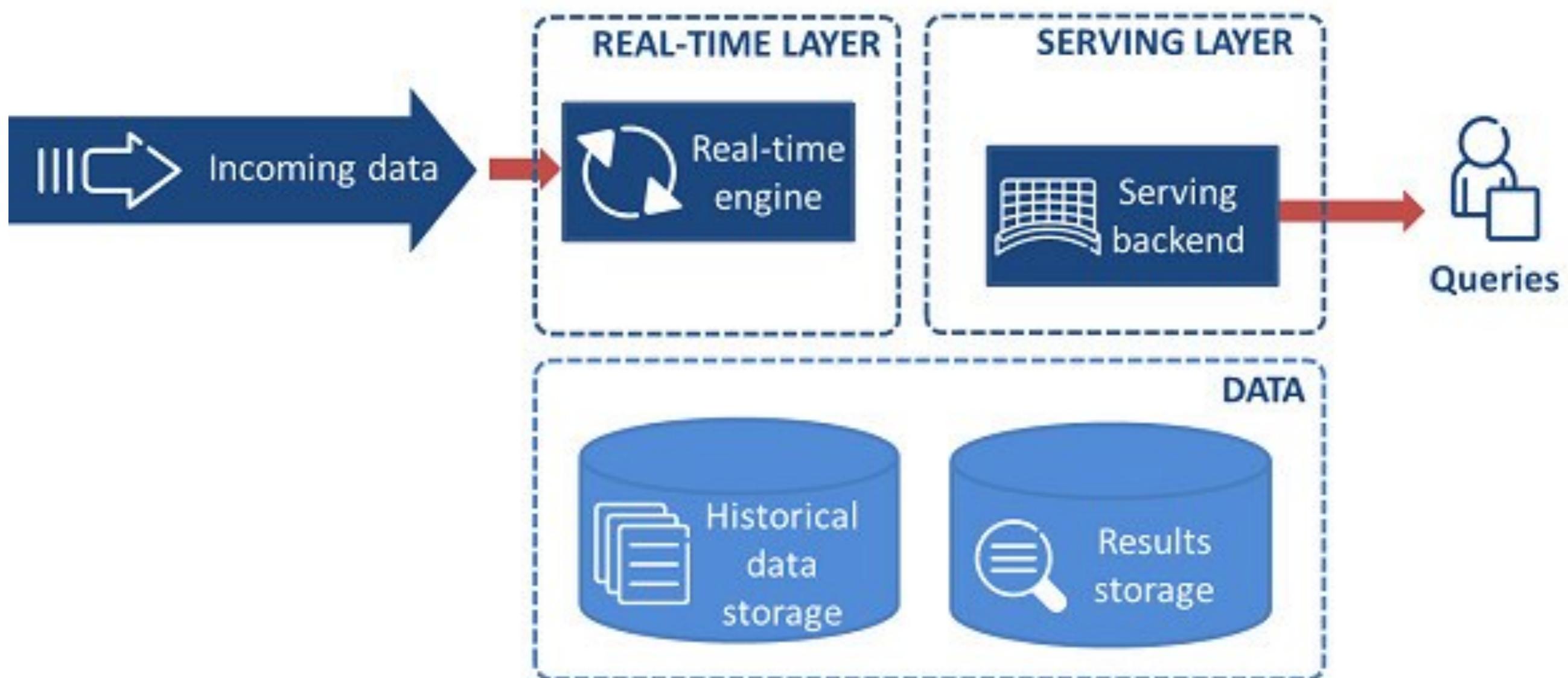
# Lambda Architecture



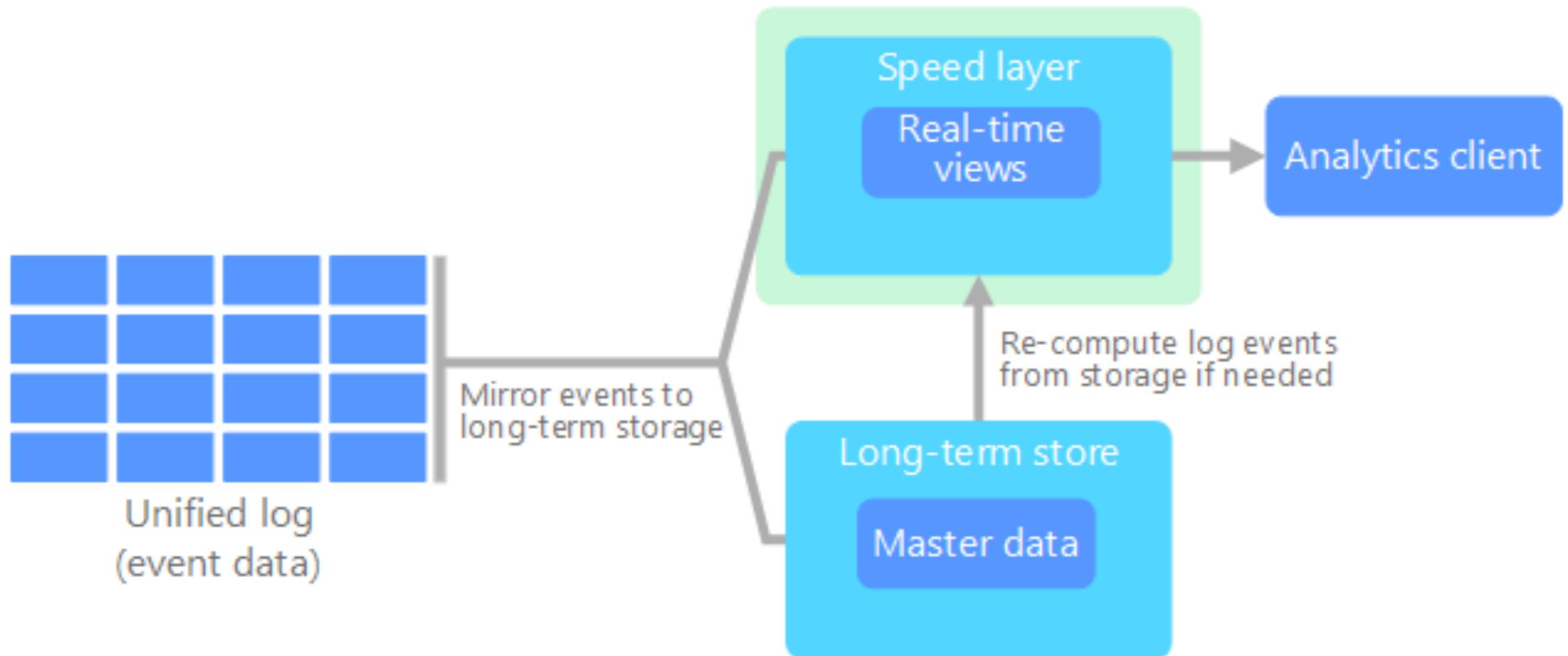
# Kappa Architecture

One pipeline for real-time and batch consumers

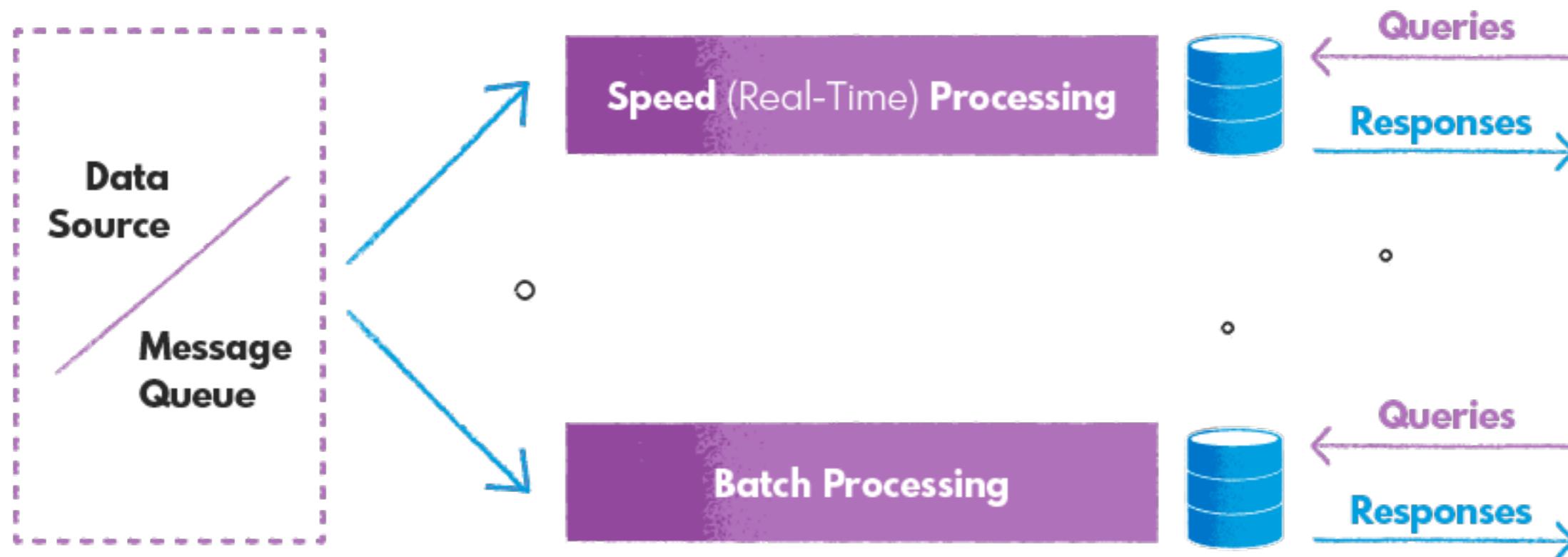




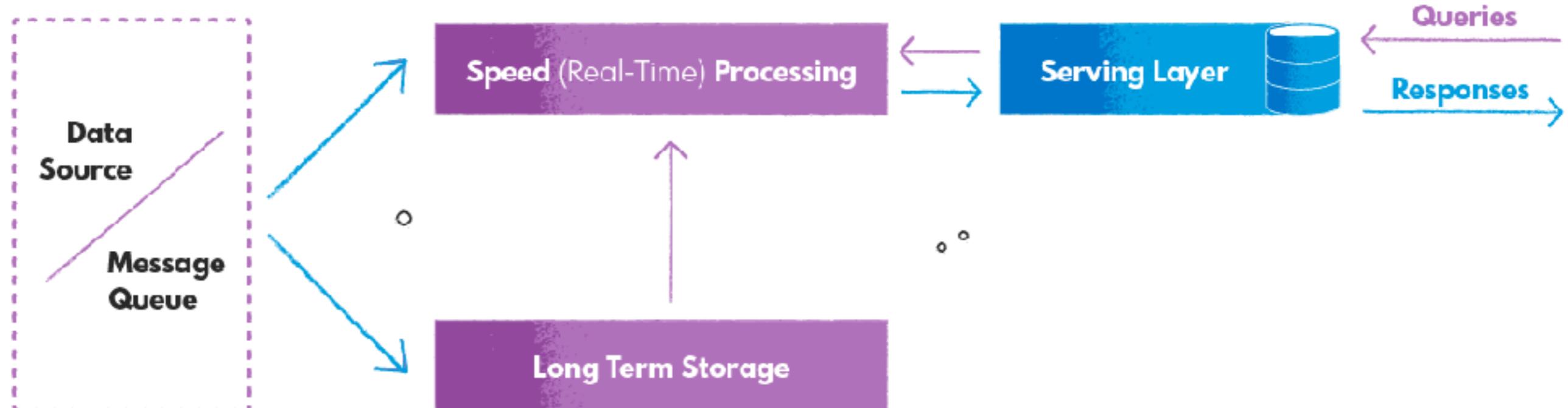
# Kappa Architecture

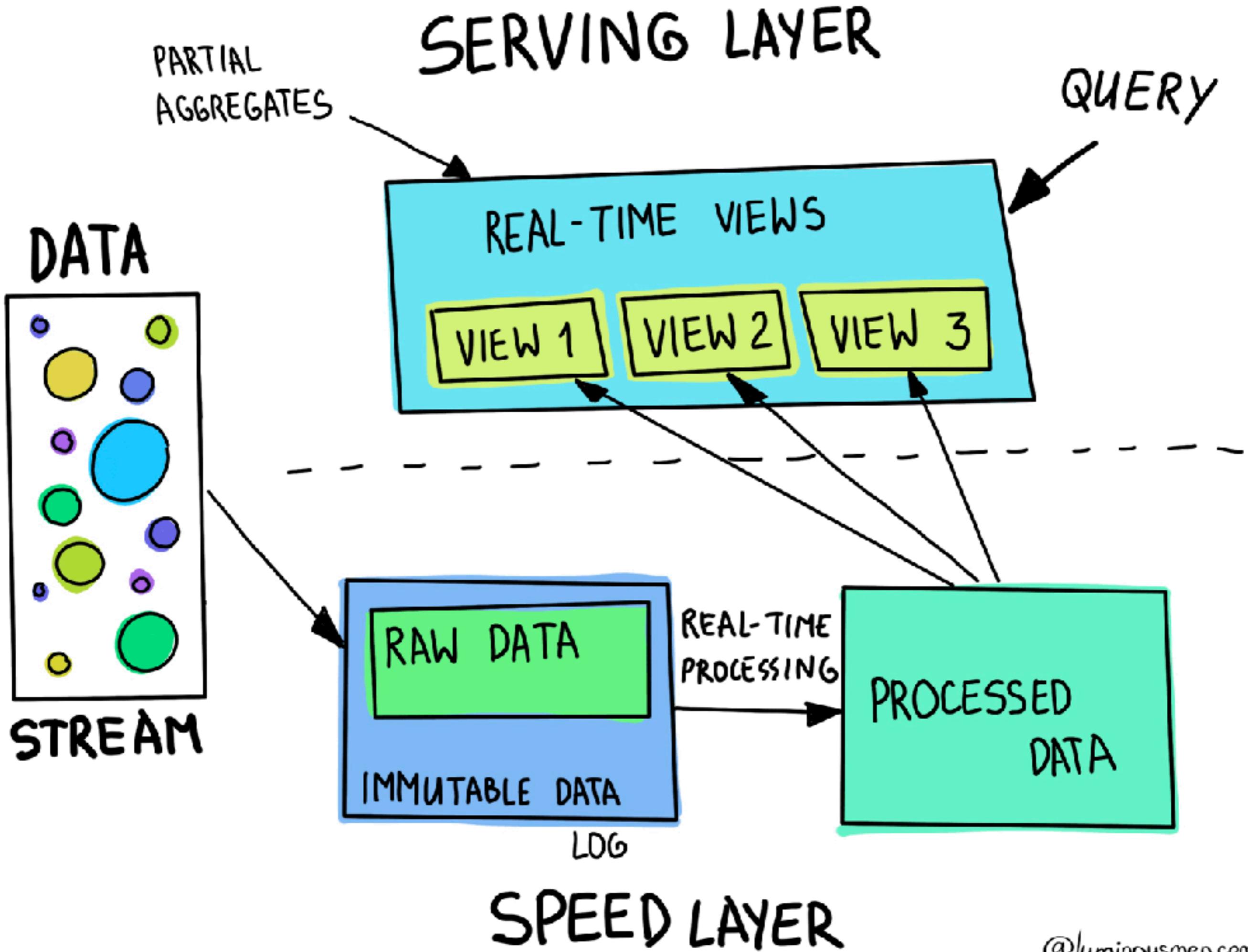


## LAMDA



## KAPPA





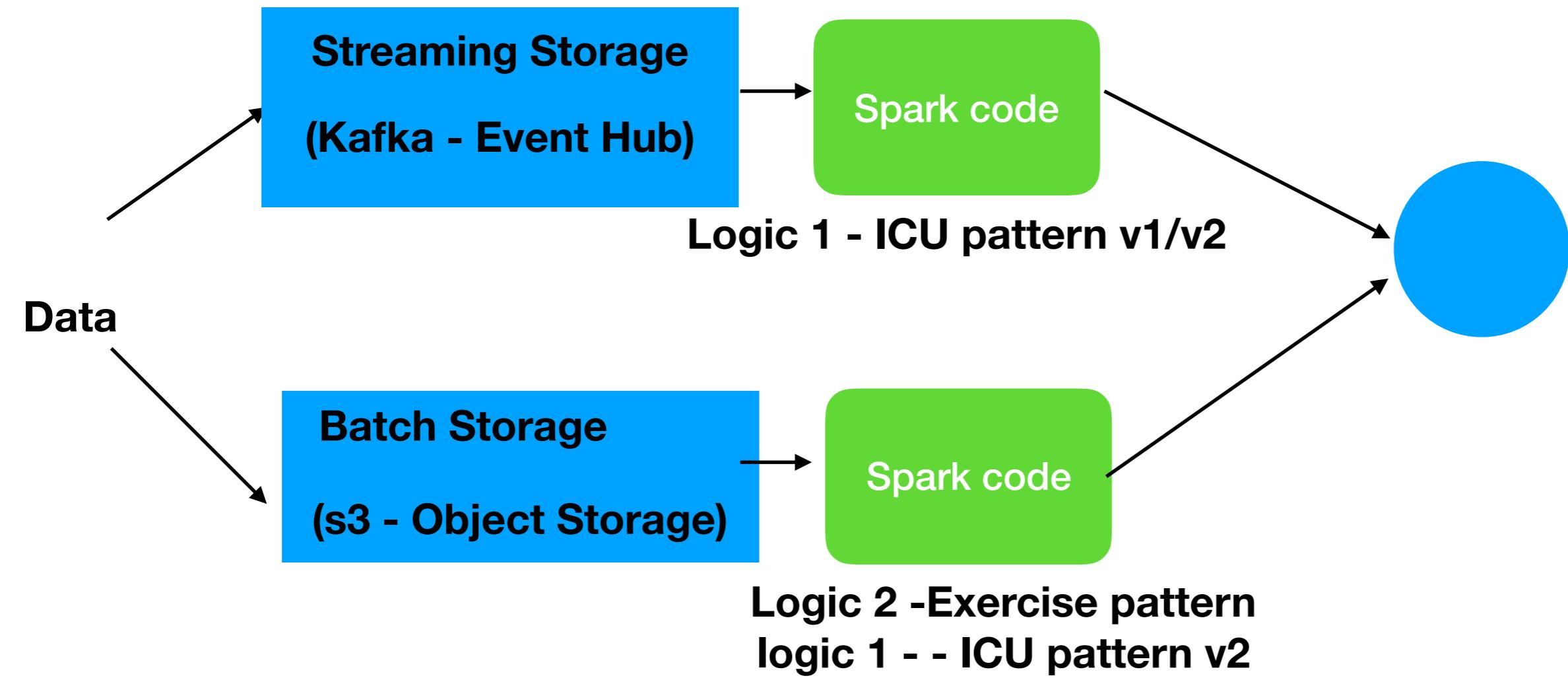
# What Architecture to Choose?

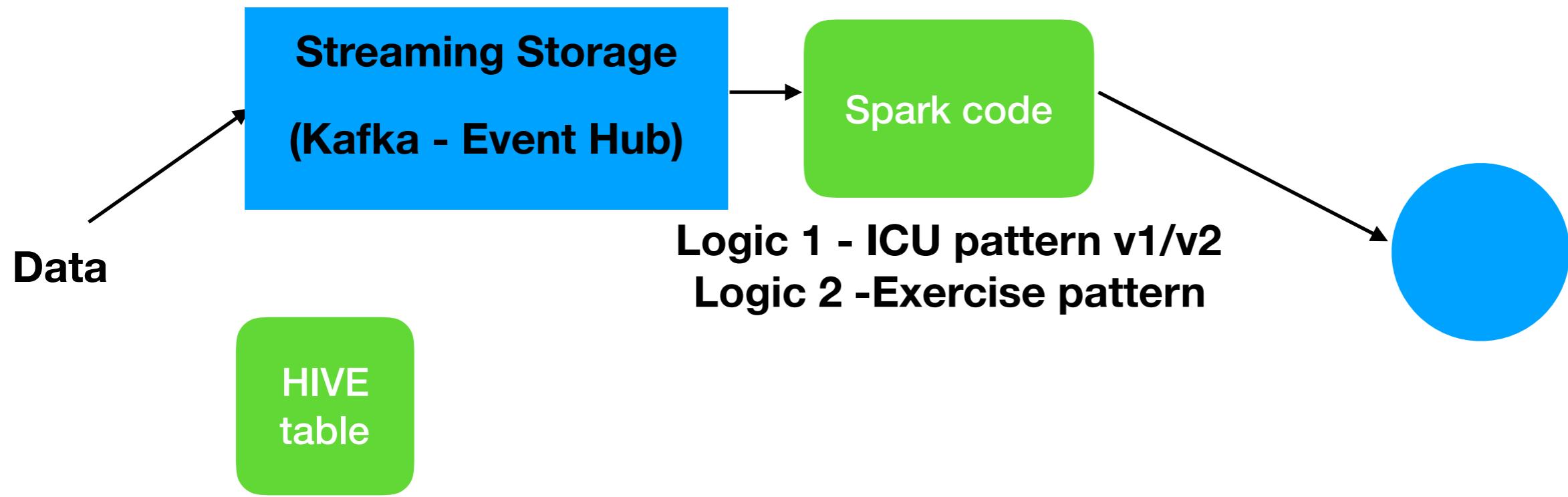


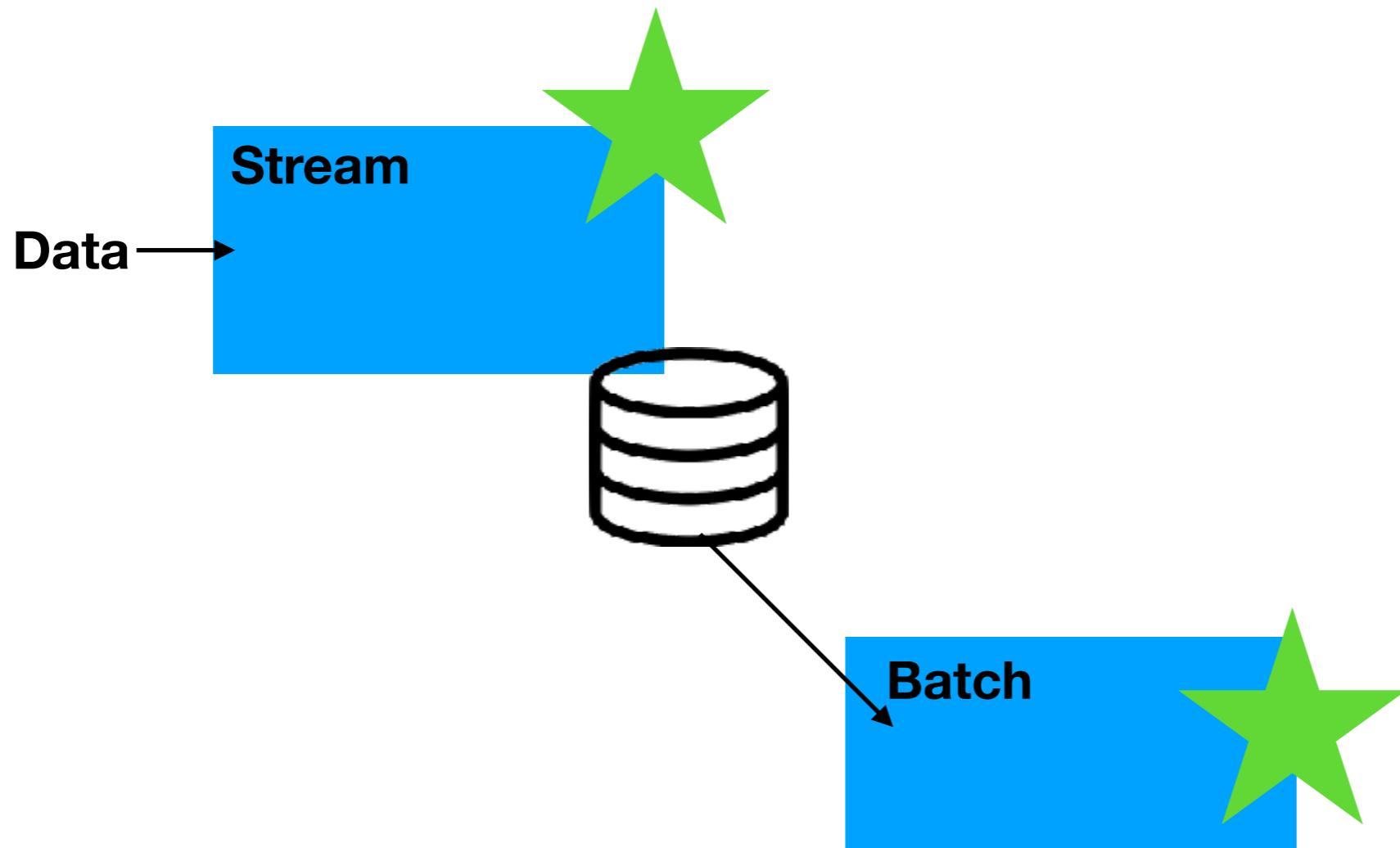
Kappa  
Real-Time  
Data in Motion

VS

Lambda  
Batch  
Data at Rest









Tx Tx Tx Tx Tx

**2 tx, same employee, in < 5 minutes on same account  
Tx by an employee on an acc from a different regions**



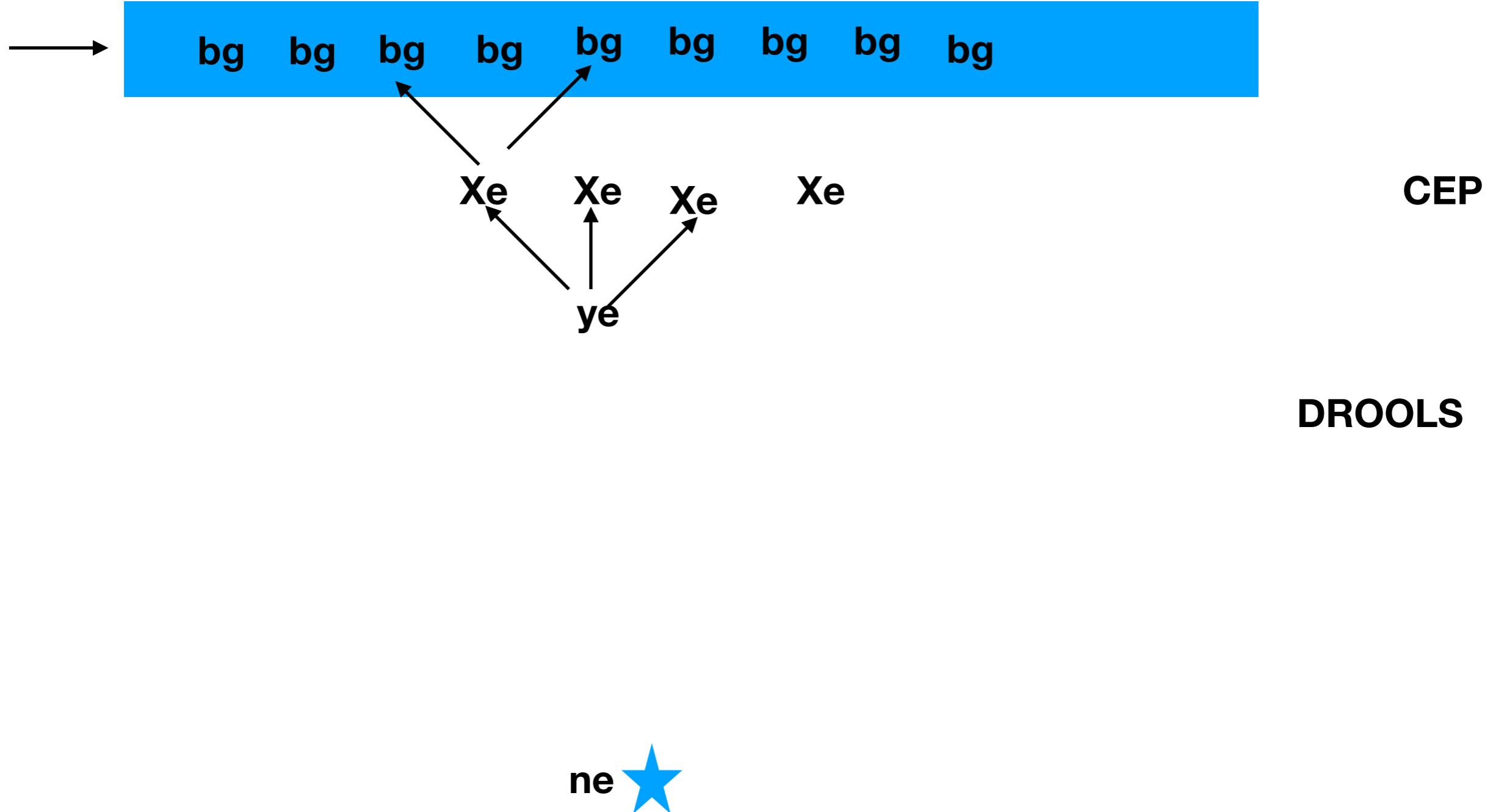
bg bg bg bg bg bg

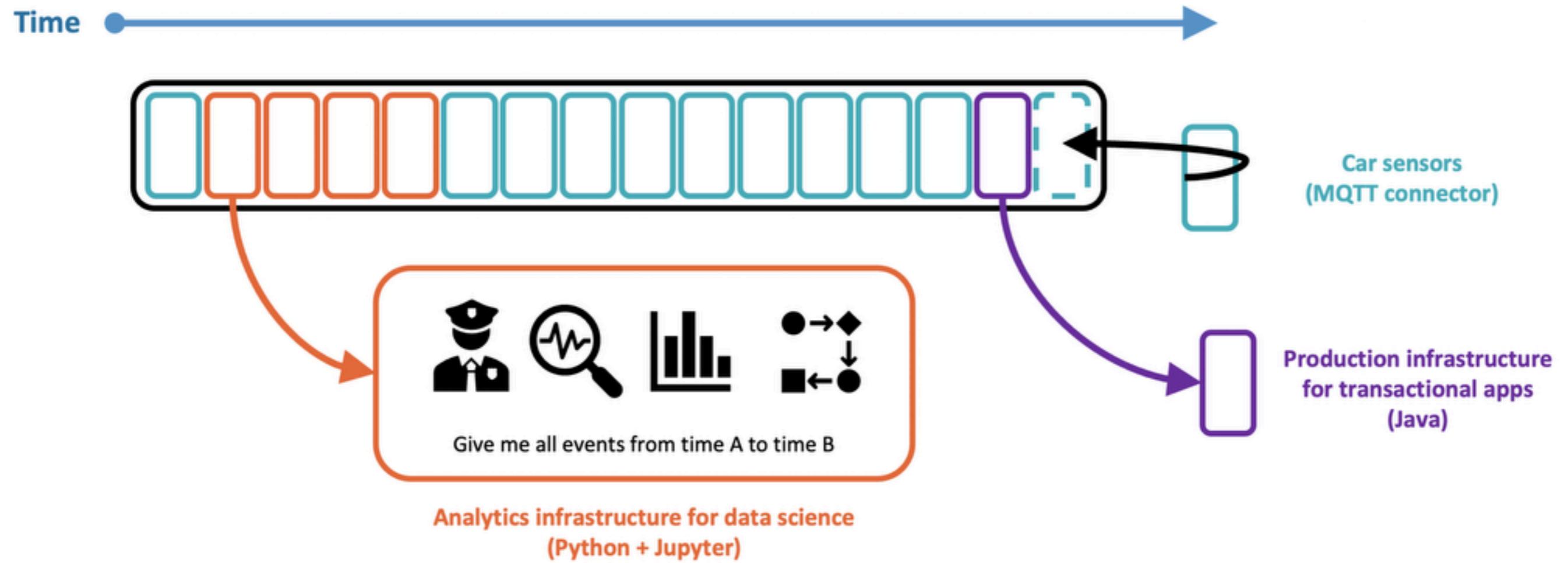
bg

bg



**Welldoc**





Kappa architectures enable transactional workloads in addition to analytical workloads.

A single pipeline for everything. No need for a Lambda architecture! Kappa enables transactional and analytical workloads.

**Stream**

**LOB**

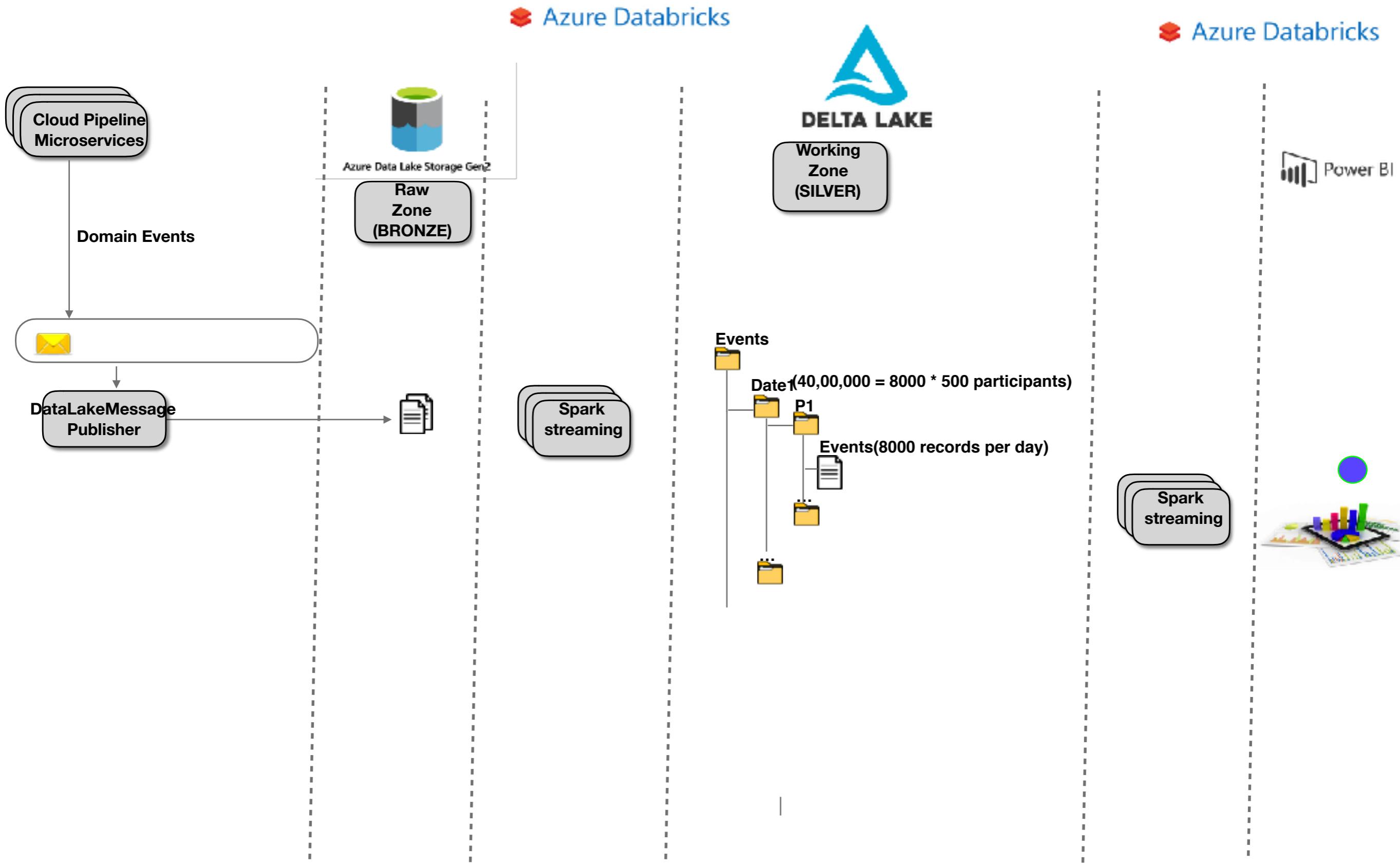
**Analytics**

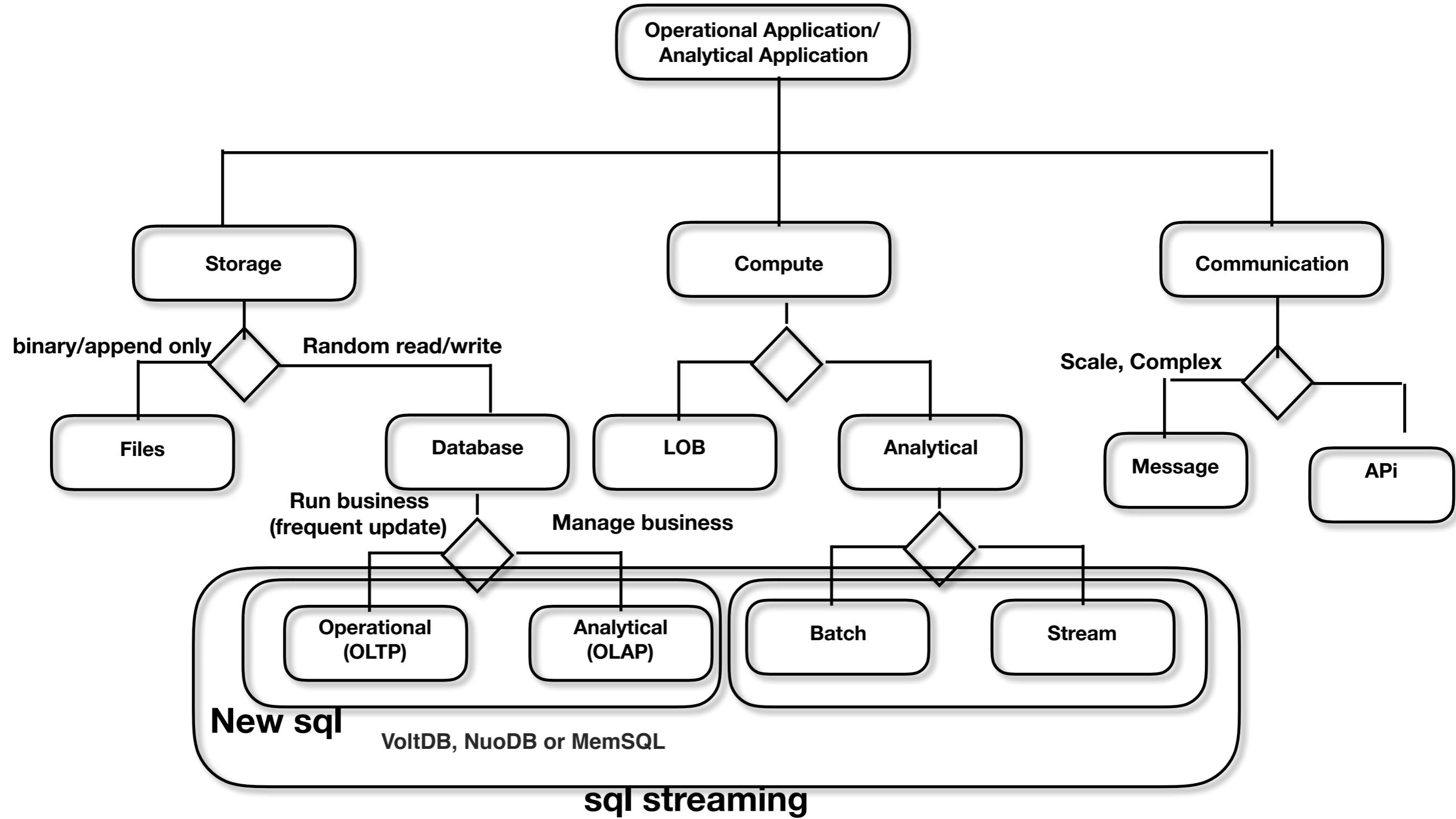
**Stream**

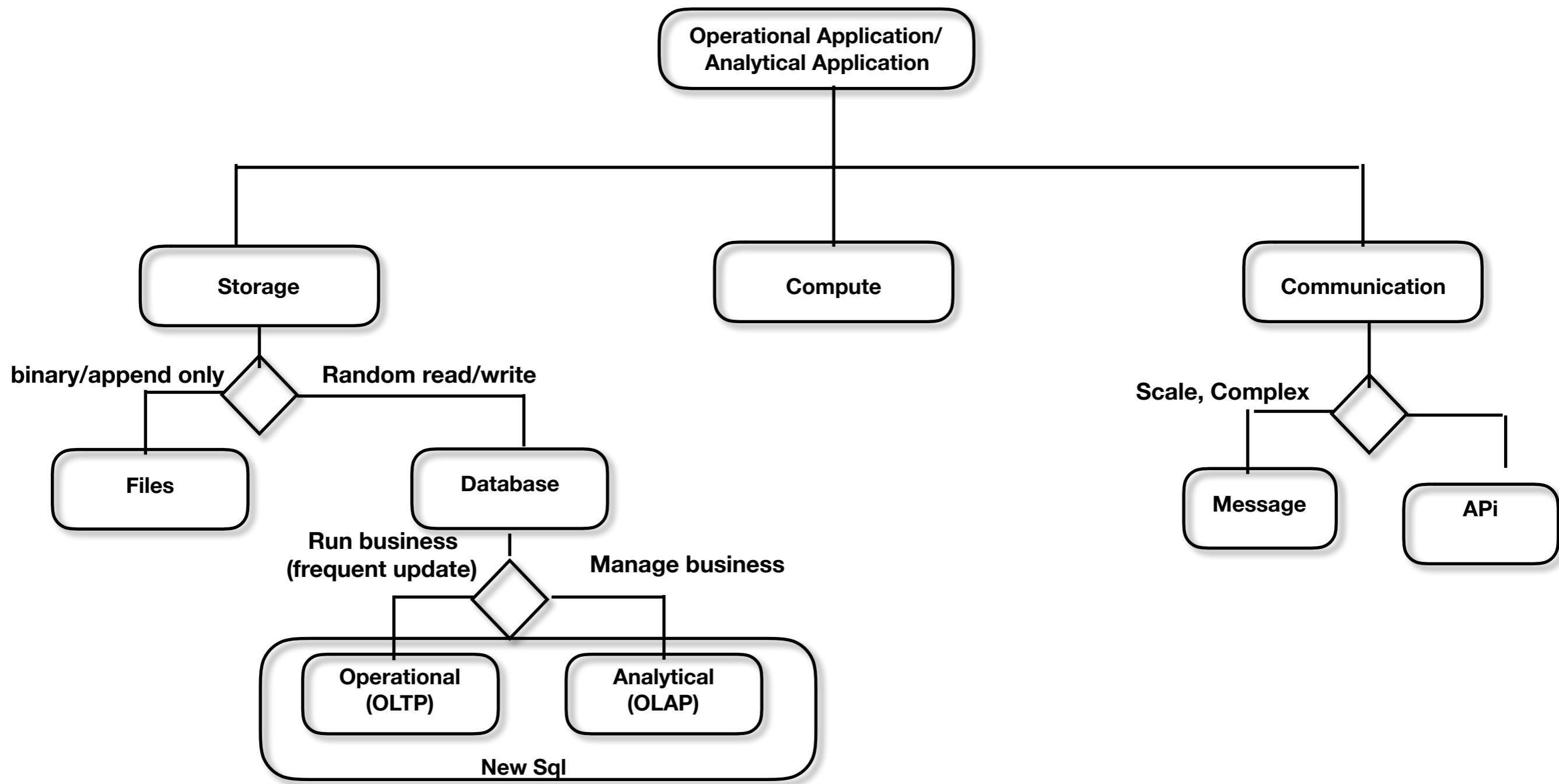
**Batch**

Spark itself doesn't manage these machines. It needs a cluster manager (also sometimes called *scheduler*)

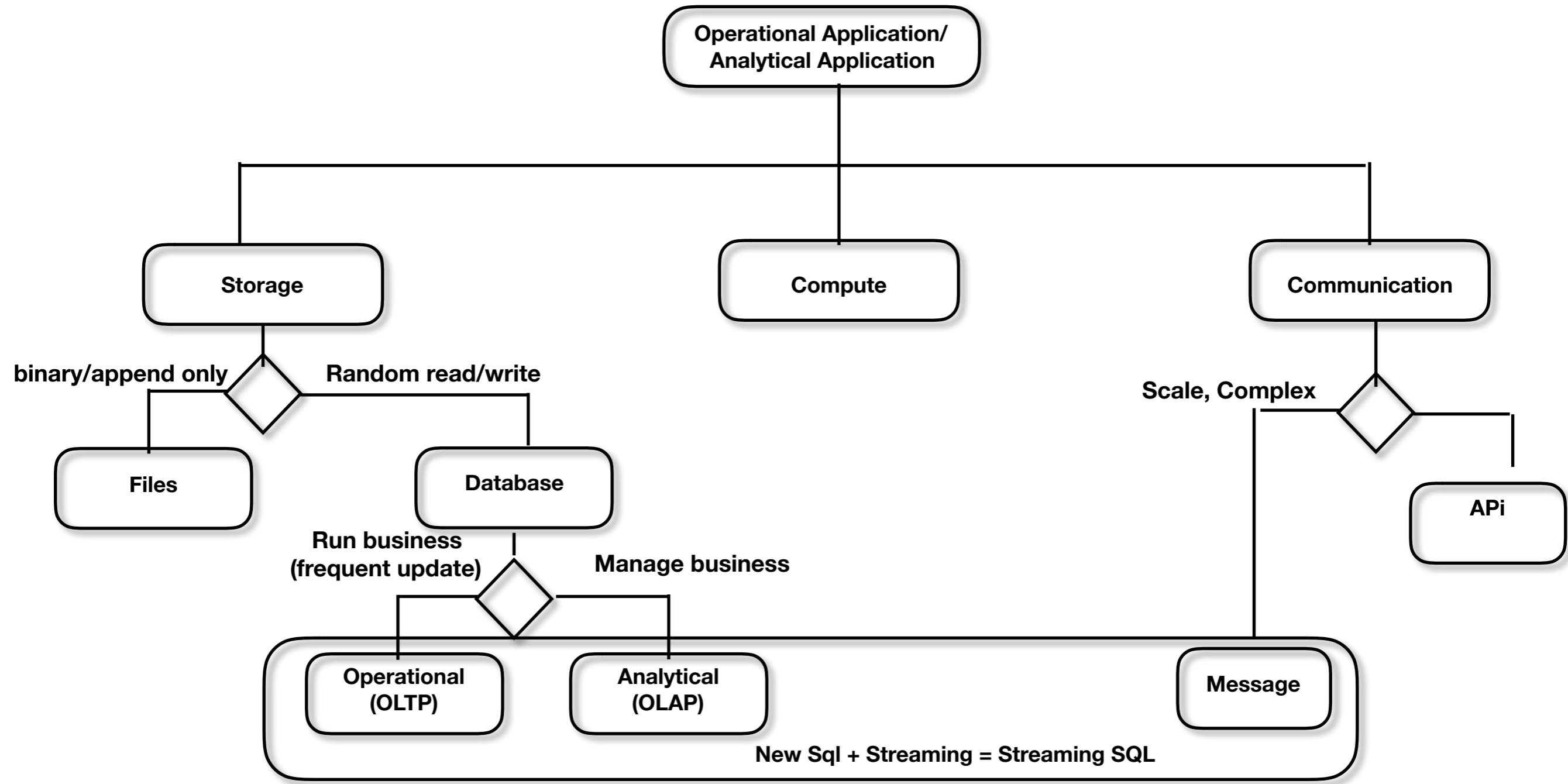
- Standalone: Simple cluster-manager, limited in features, incorporated with Spark.
- Apache Mesos.
- Hadoop YARN
- Kubernetes

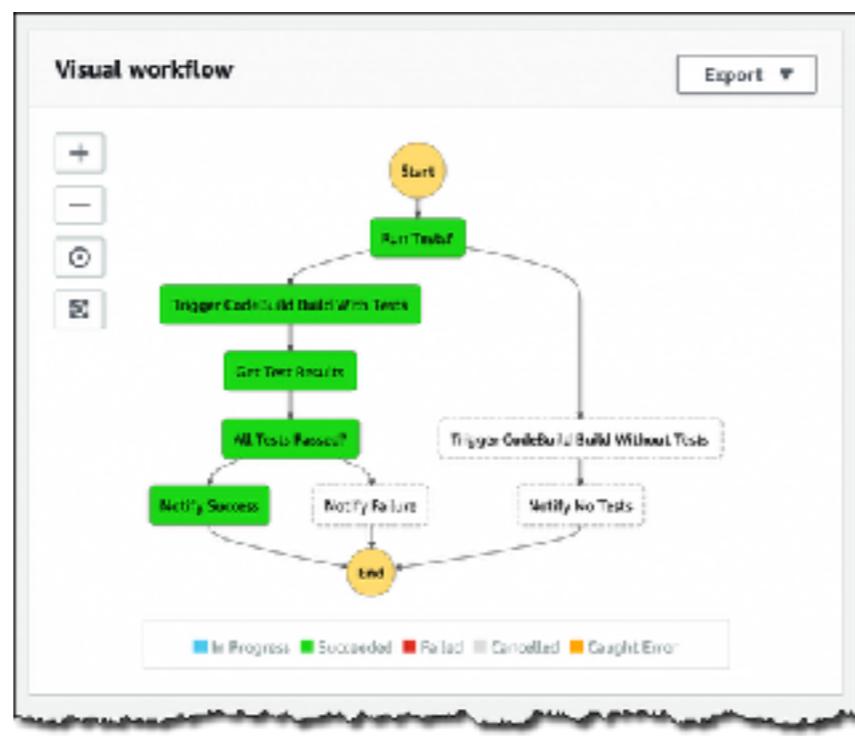






VoltDB, NuoDB or MemSQL





# AWS Step

# Azure Logic App

# Biztalk

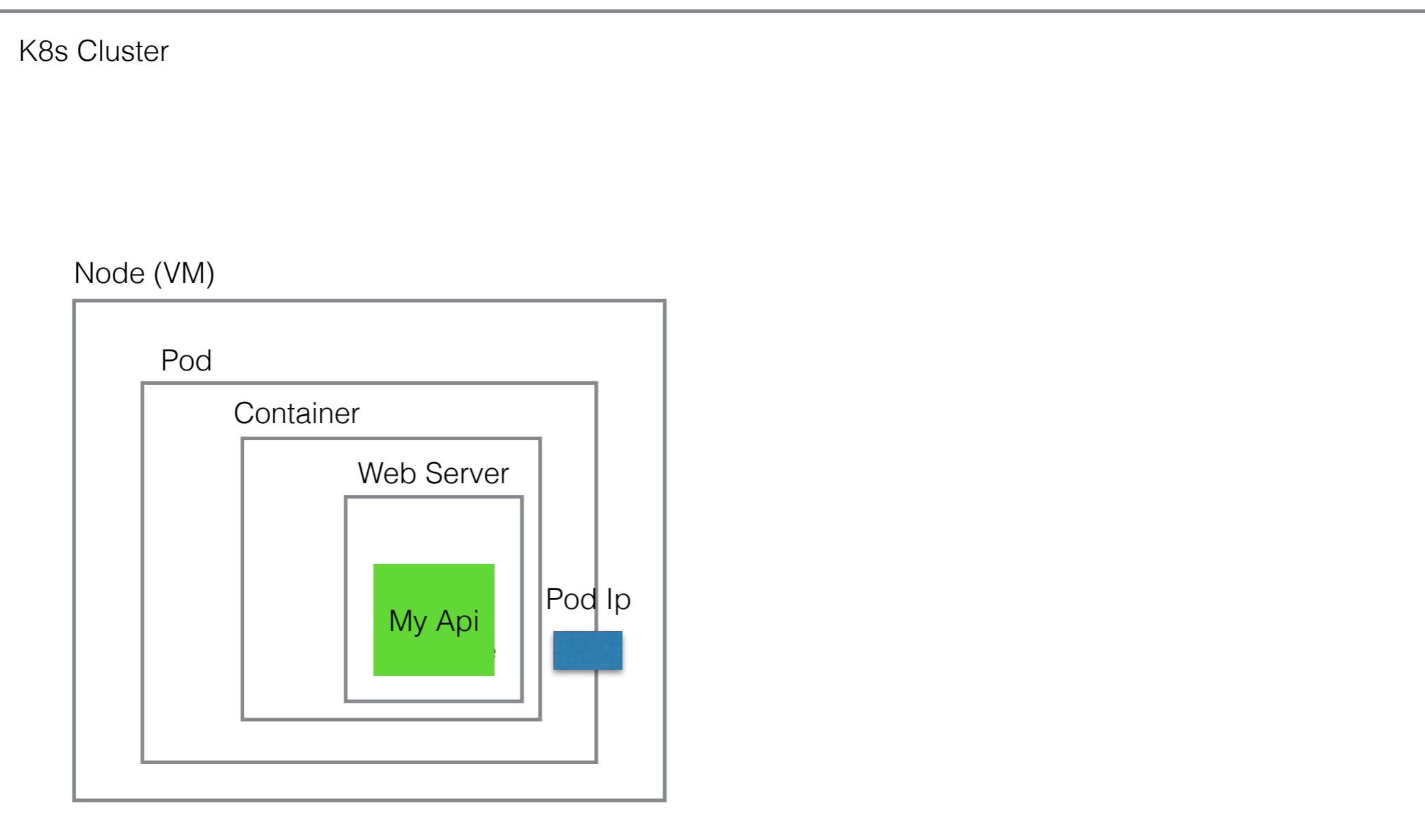
# Webmethods

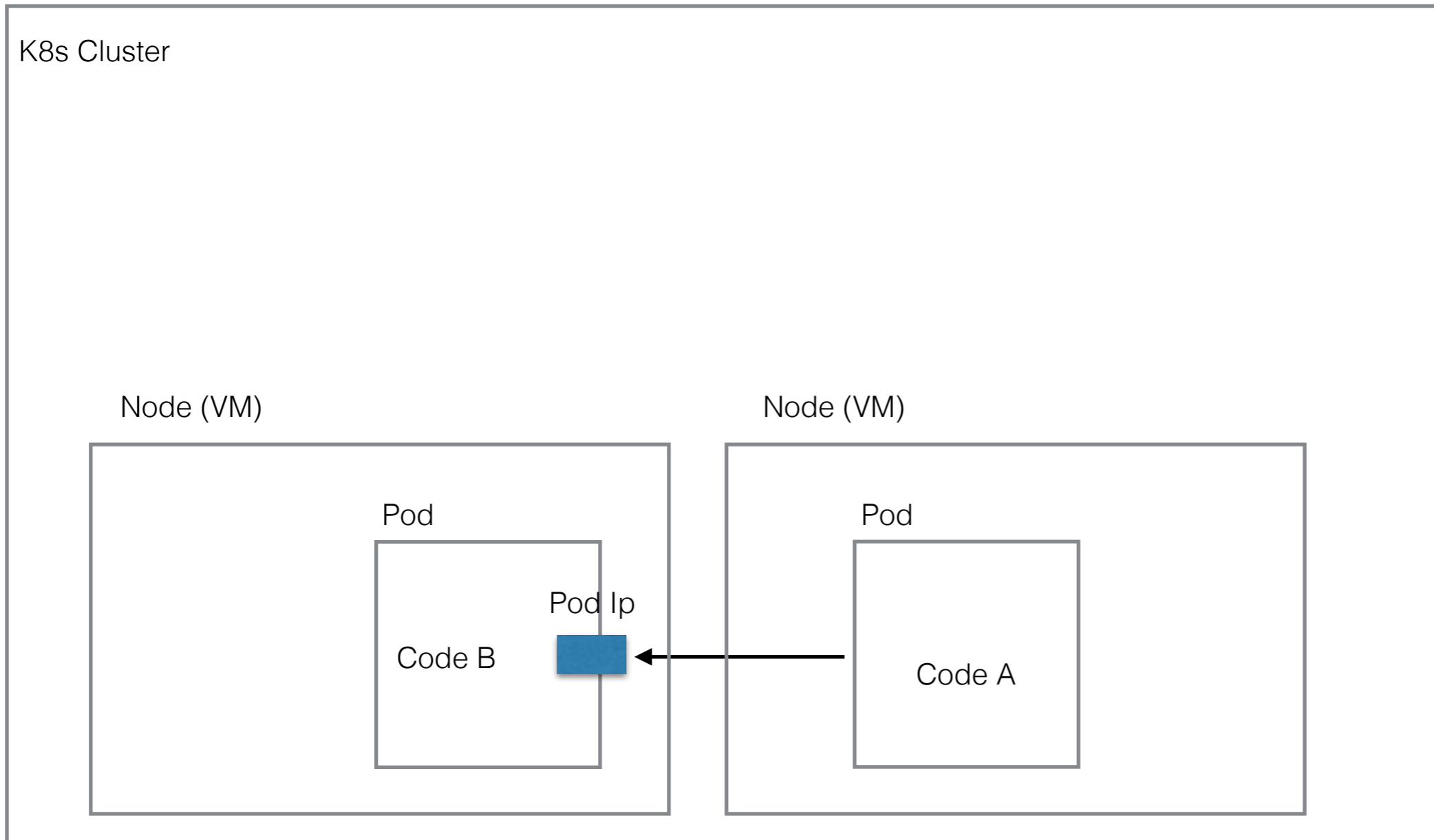
# Mule

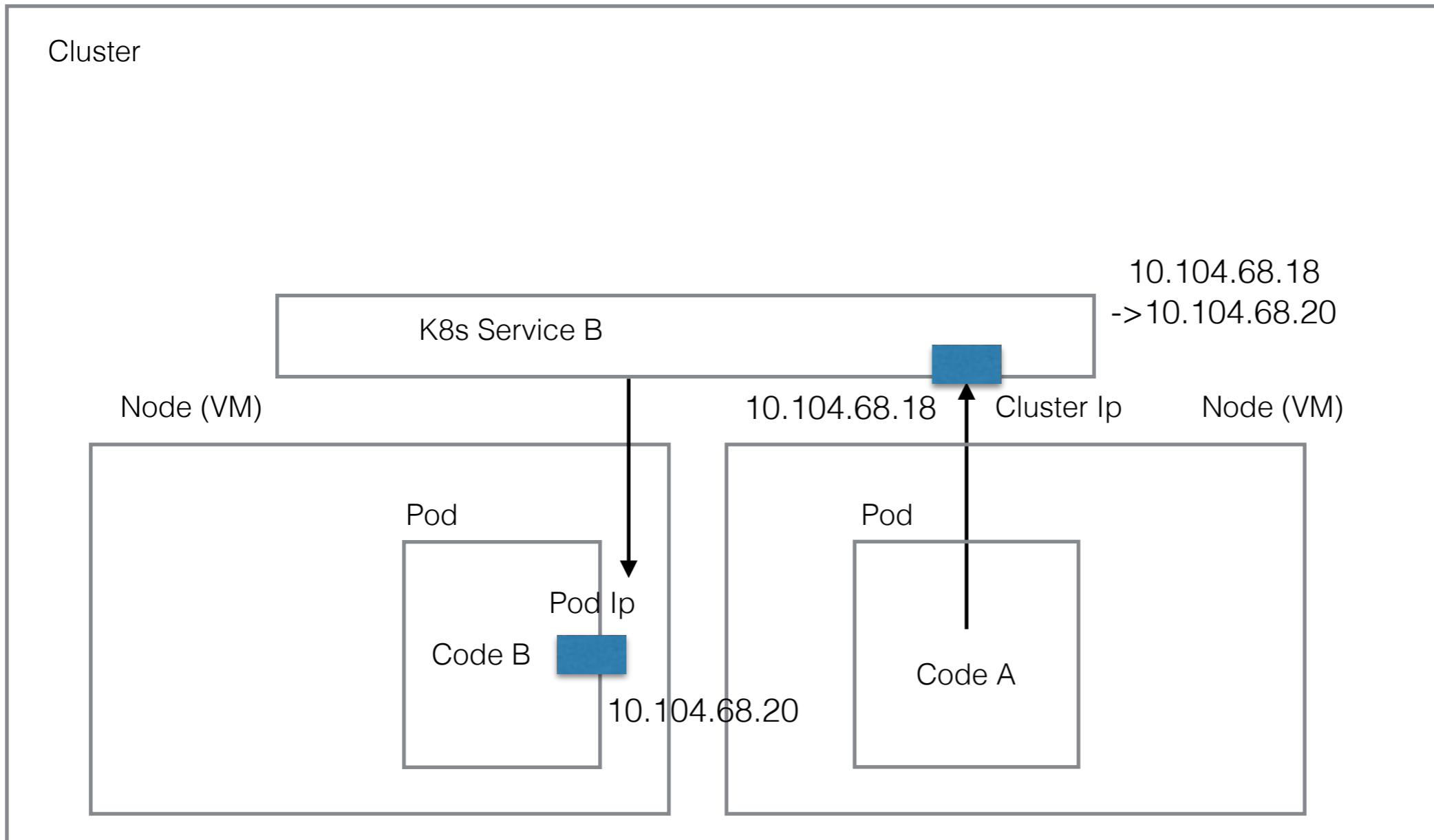
# ServiceNow

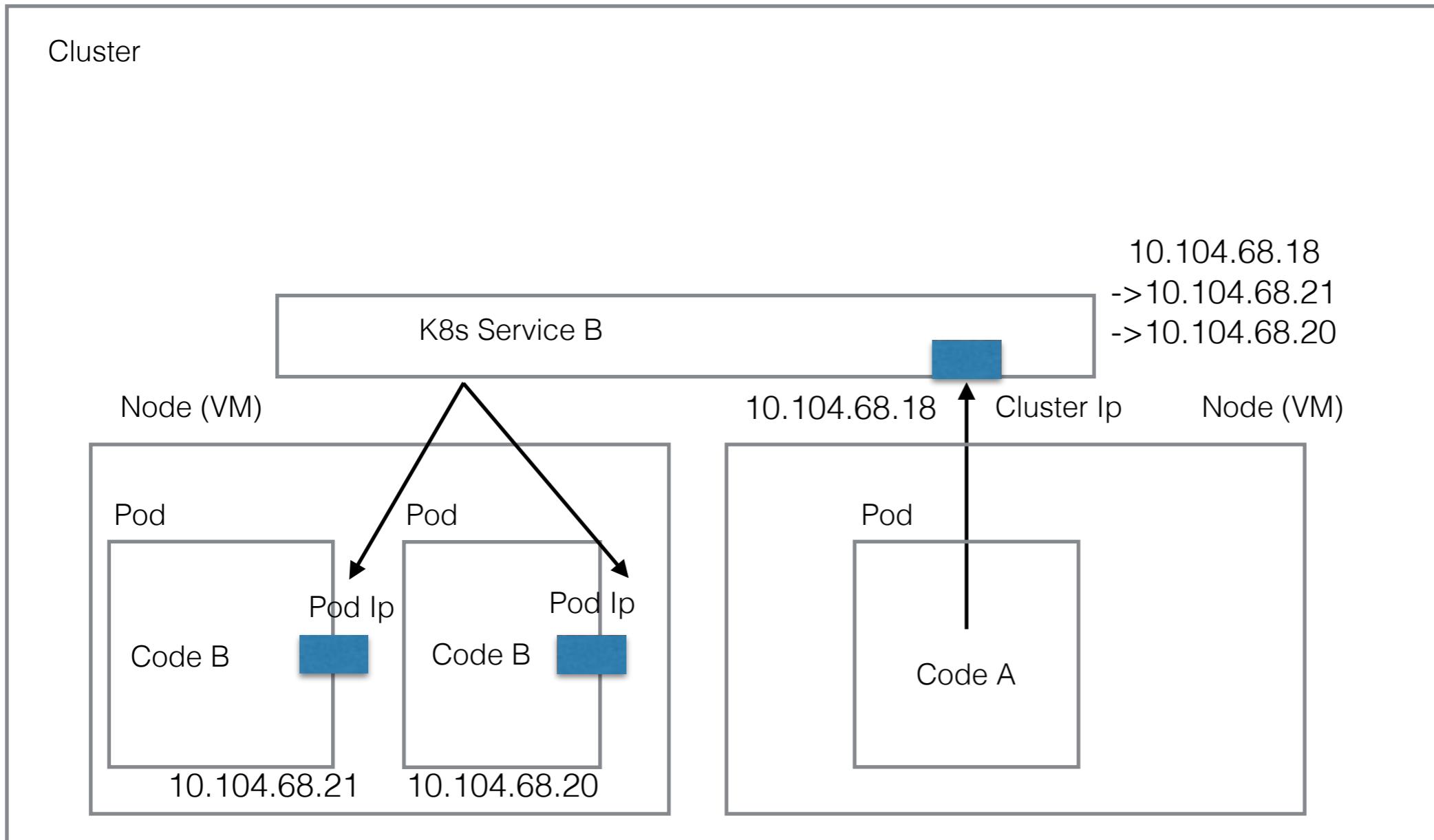
#

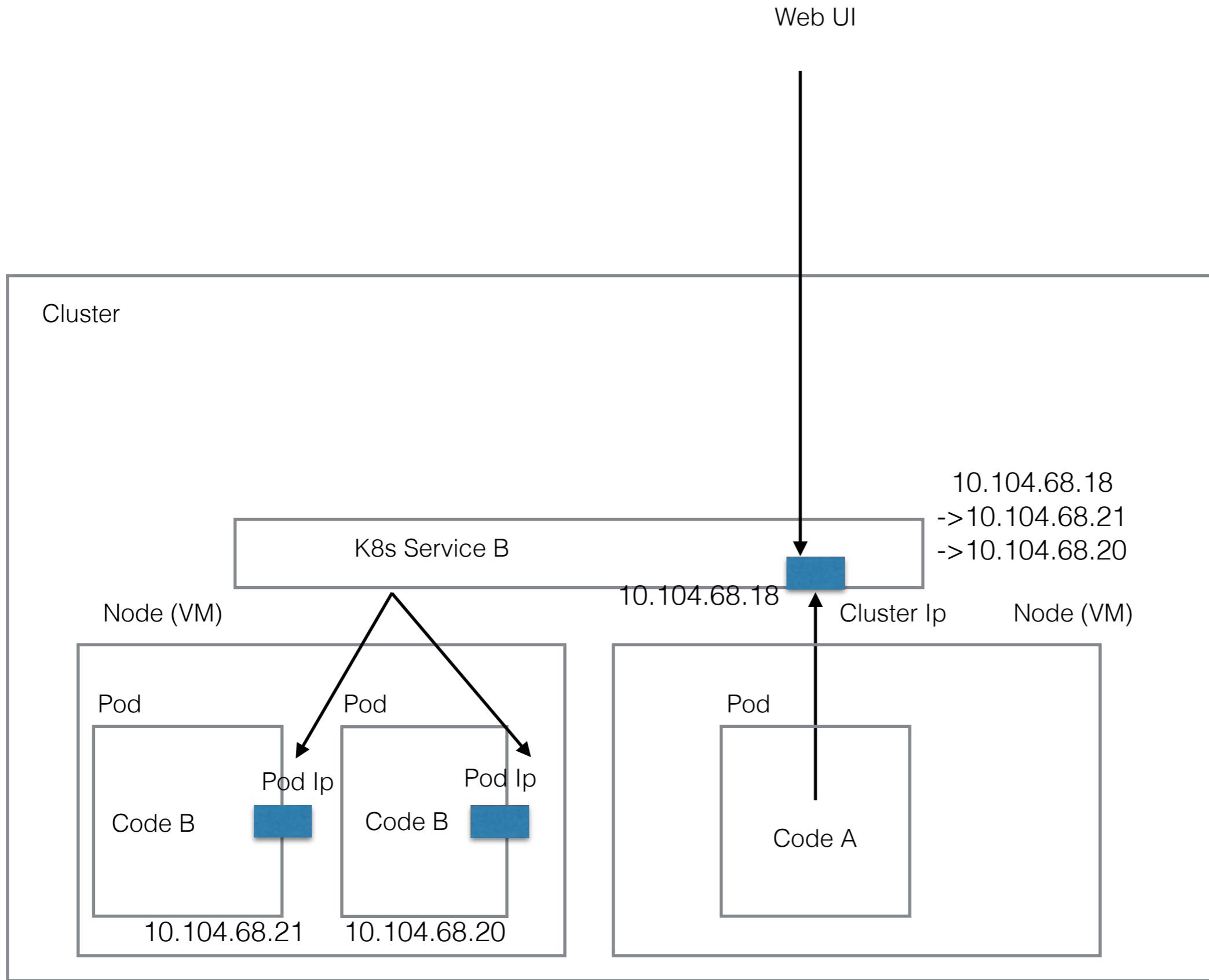
# **Deployment View**



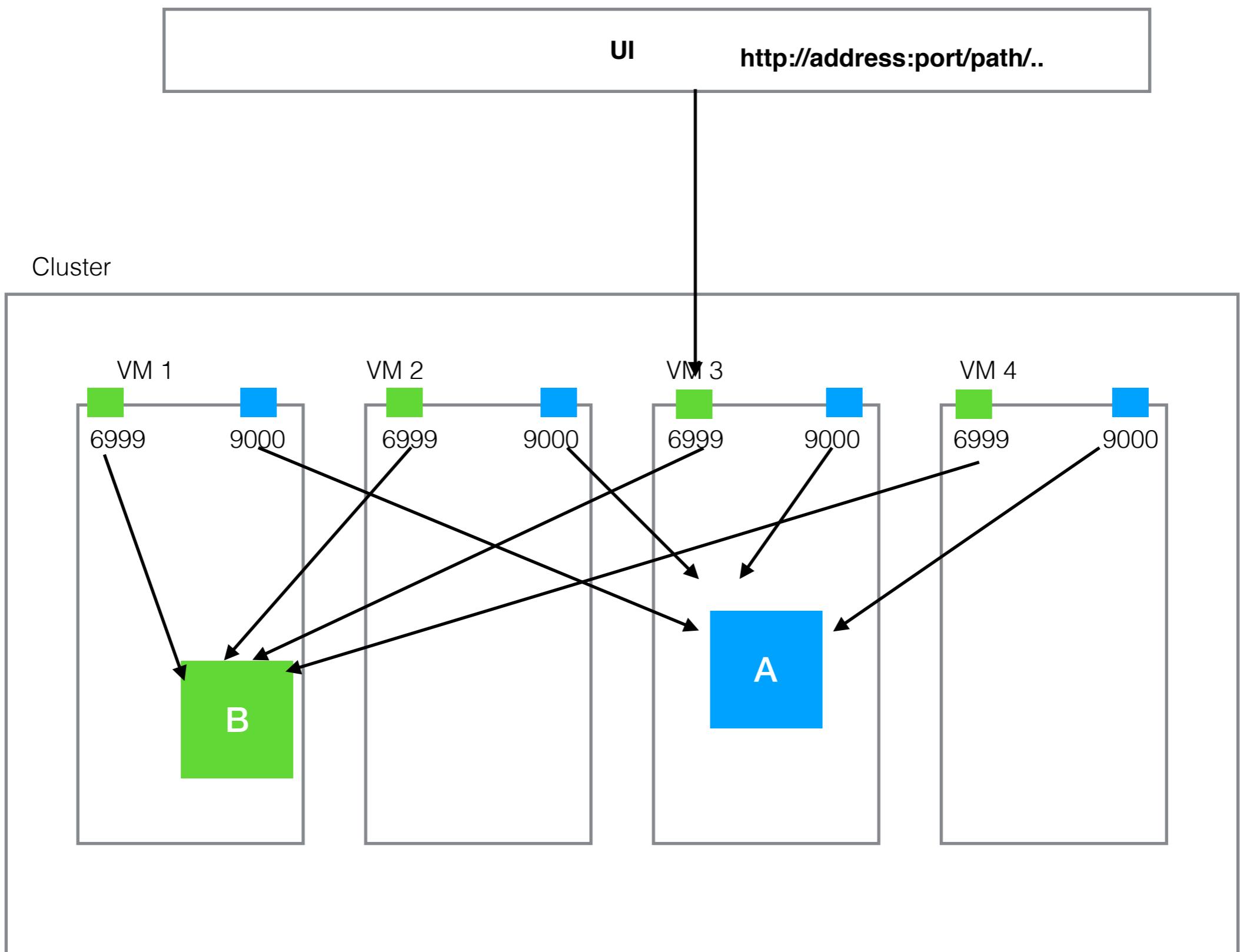




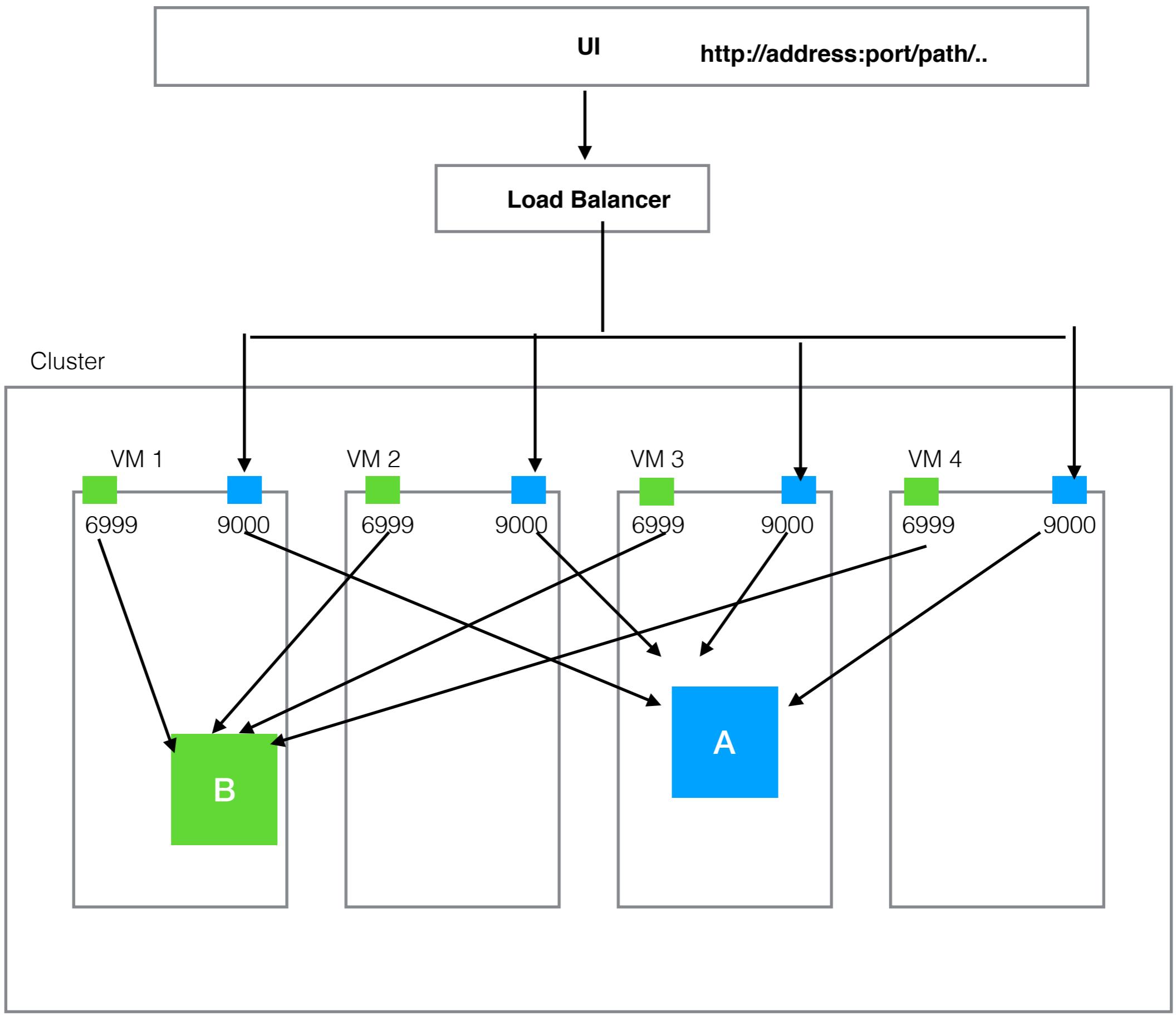




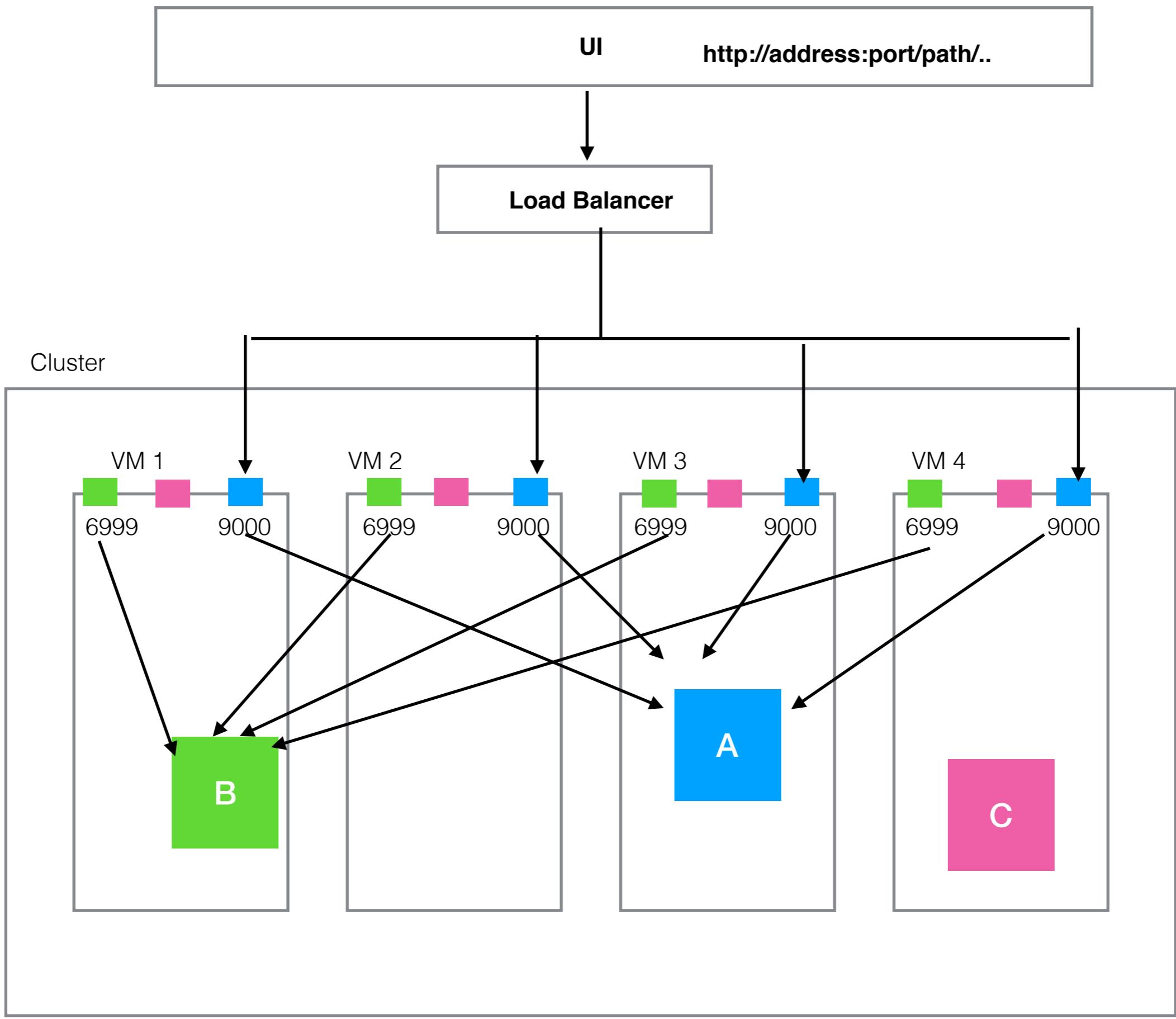
A -> any VM ip address: 9000  
B -> any VM ip address: 6999



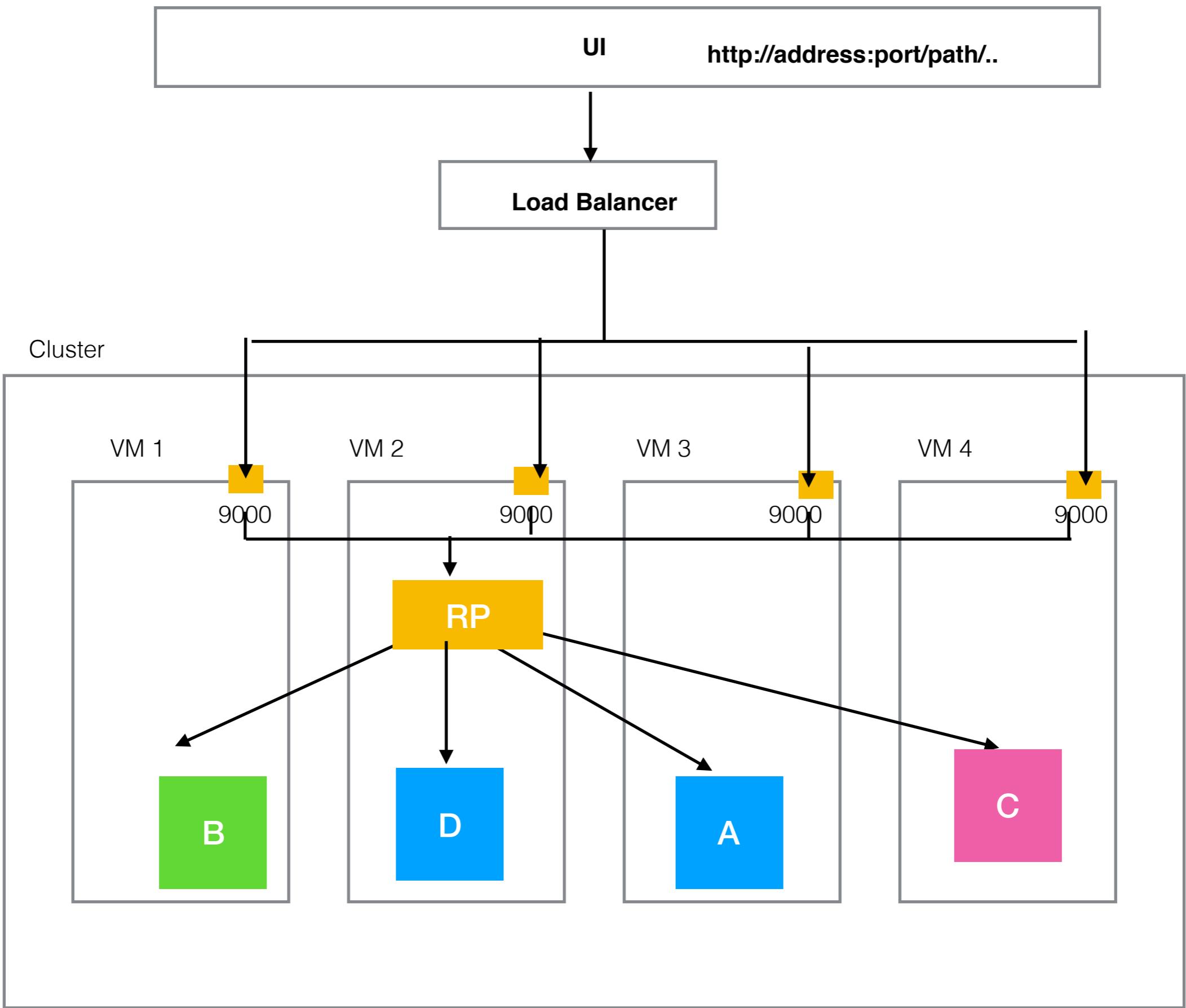
A -> any VM ip address: 9000  
B -> any VM ip address: 6999



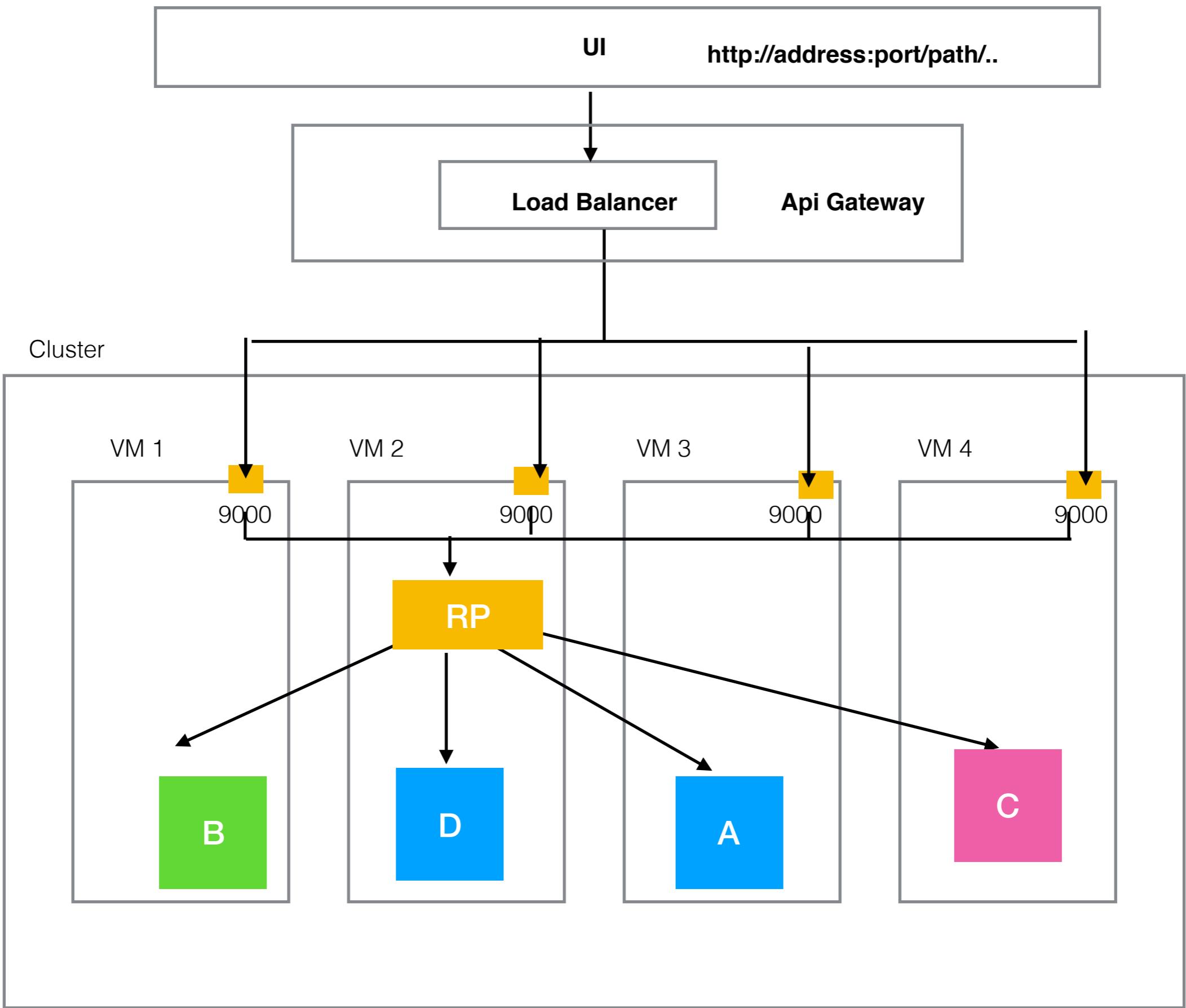
A -> any VM ip address: 9000  
B -> any VM ip address: 6999



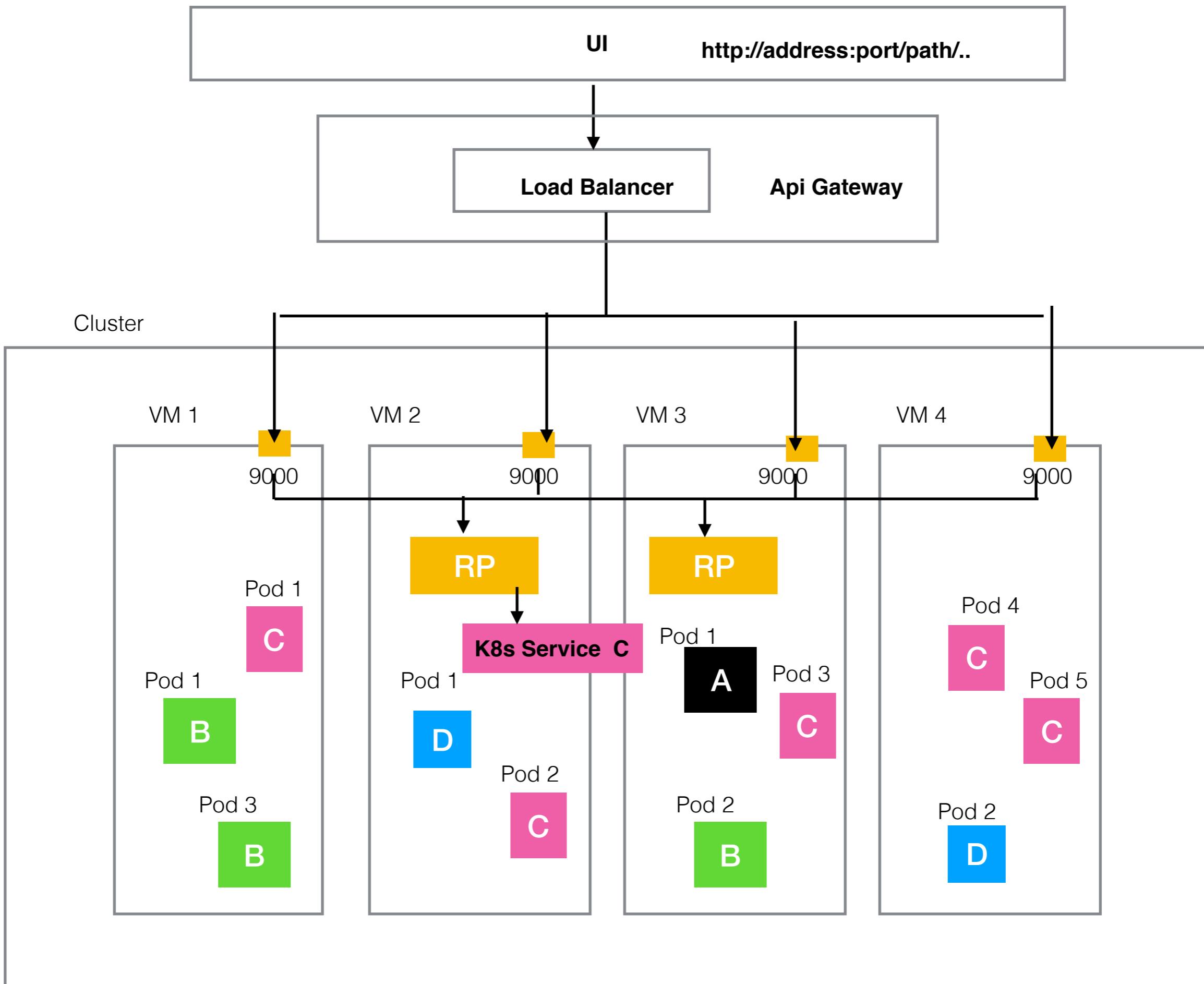
A -> any VM ip address: 9000  
B -> any VM ip address: 6999

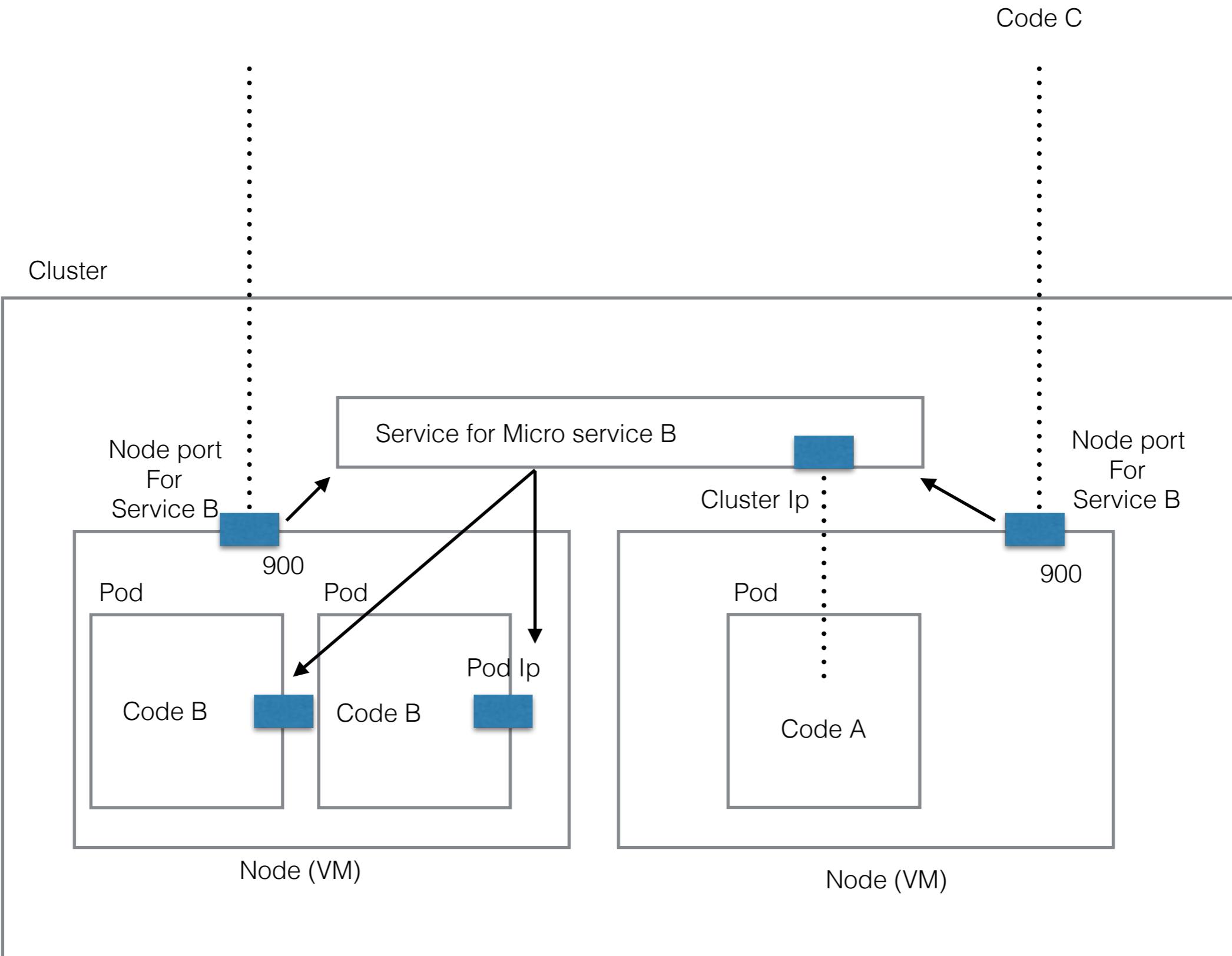


A -> any VM ip address: 9000  
B -> any VM ip address: 6999



A -> any VM ip address: 9000  
B -> any VM ip address: 6999



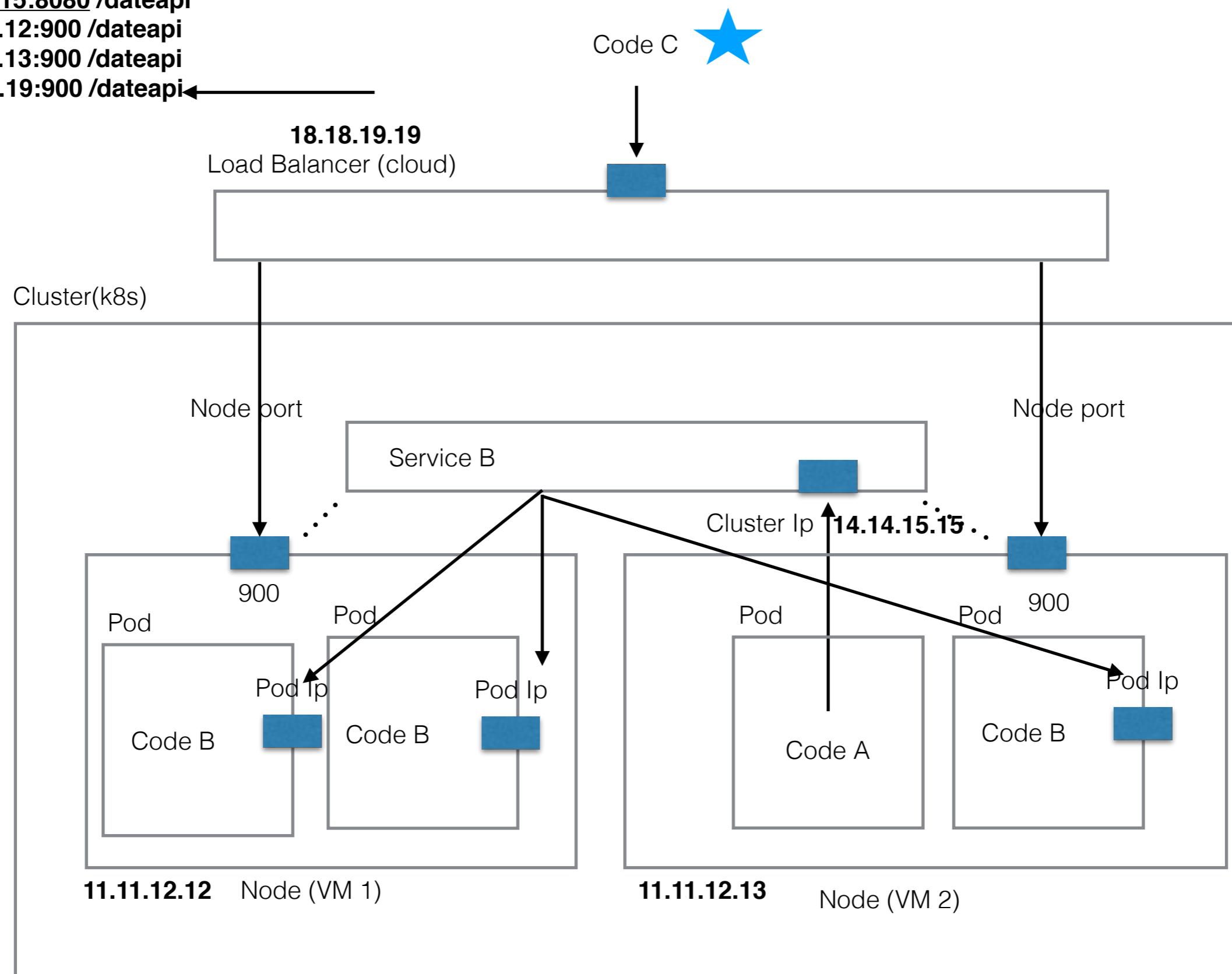


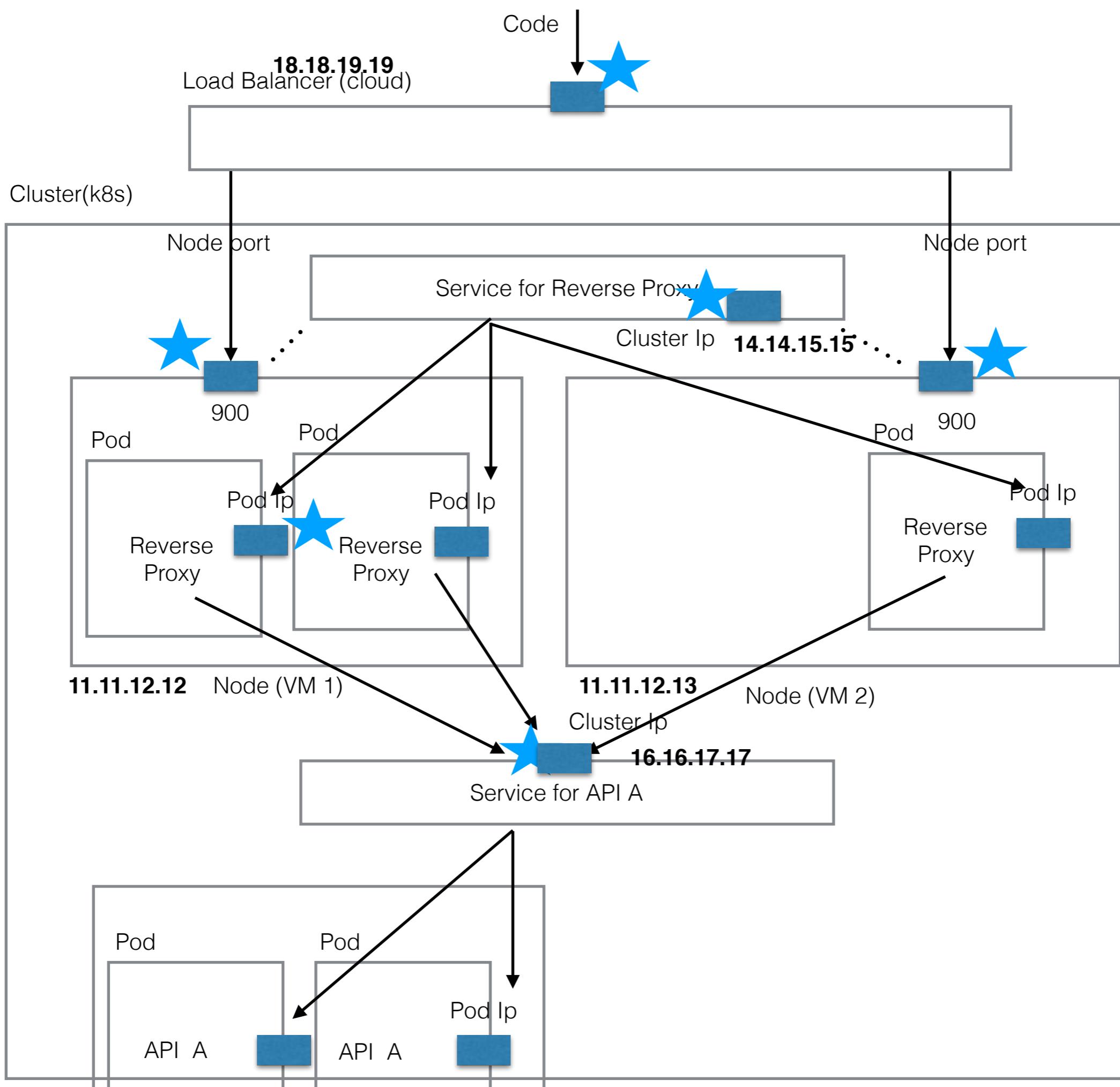
[http://14.14.15.15:8080 /dateapi](http://14.14.15.15:8080/dateapi)

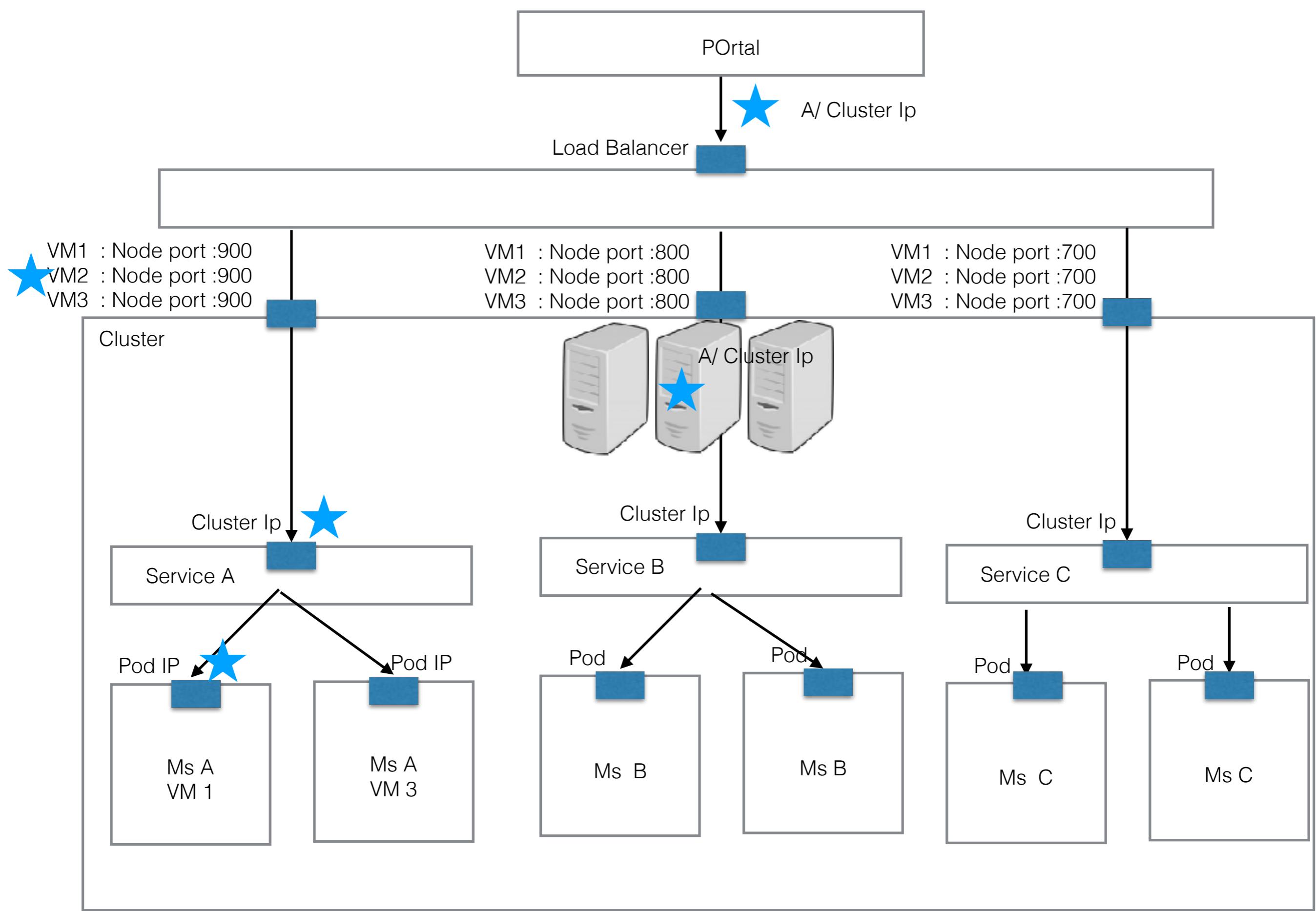
[http://11.11.12.12:900 /dateapi](http://11.11.12.12:900/dateapi)

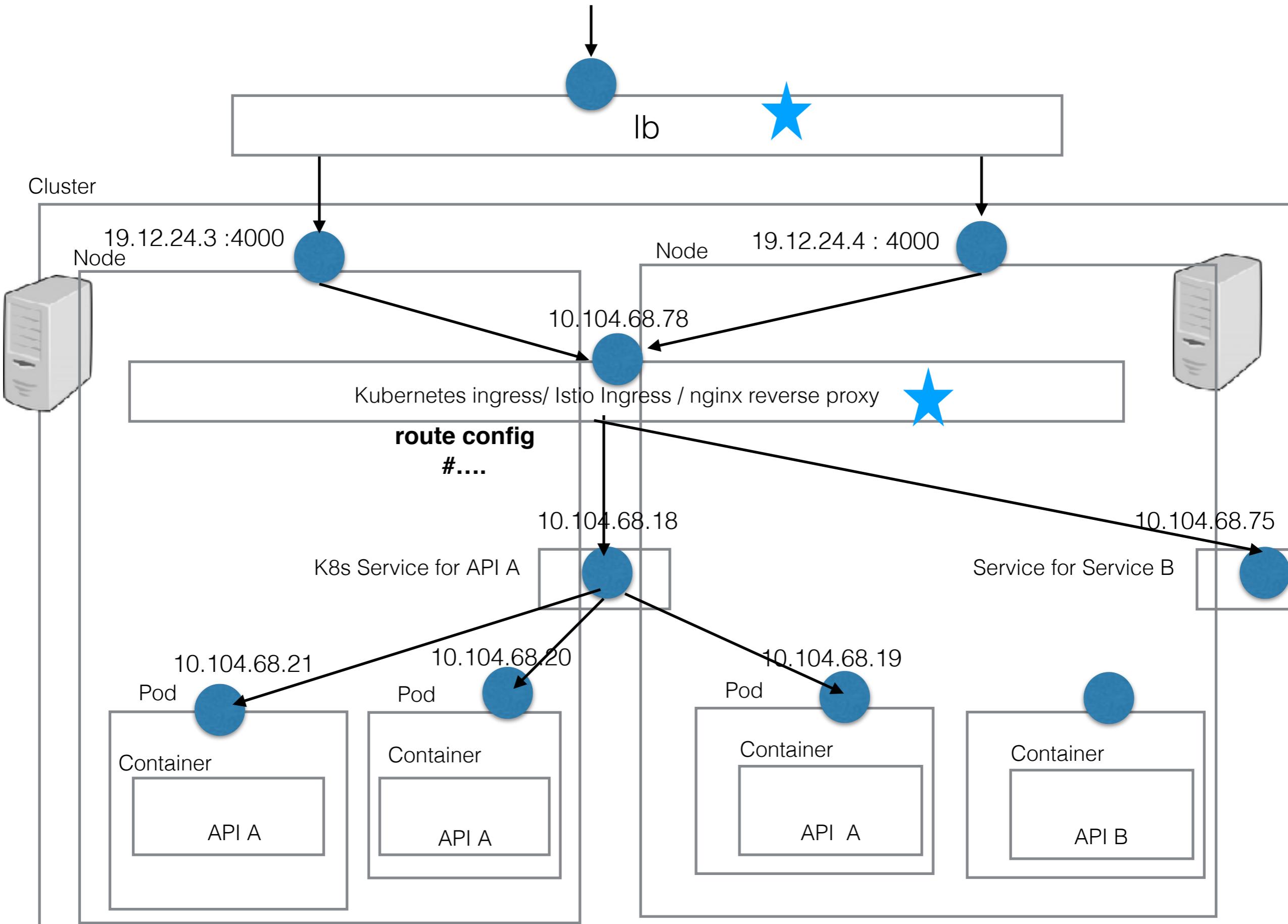
[http://11.11.12.13:900 /dateapi](http://11.11.12.13:900/dateapi)

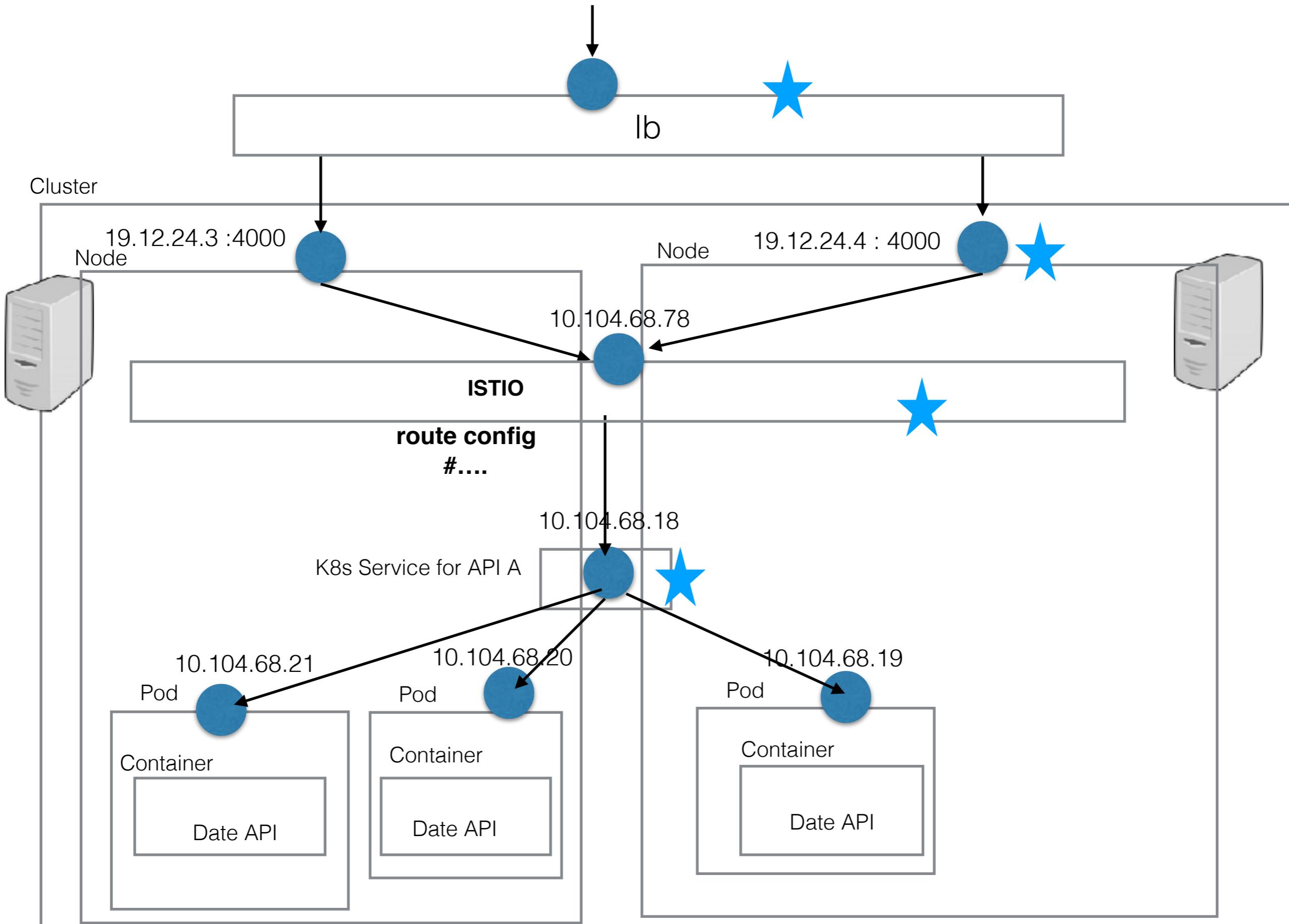
[http://18.18.19.19:900 /dateapi](http://18.18.19.19:900/dateapi)

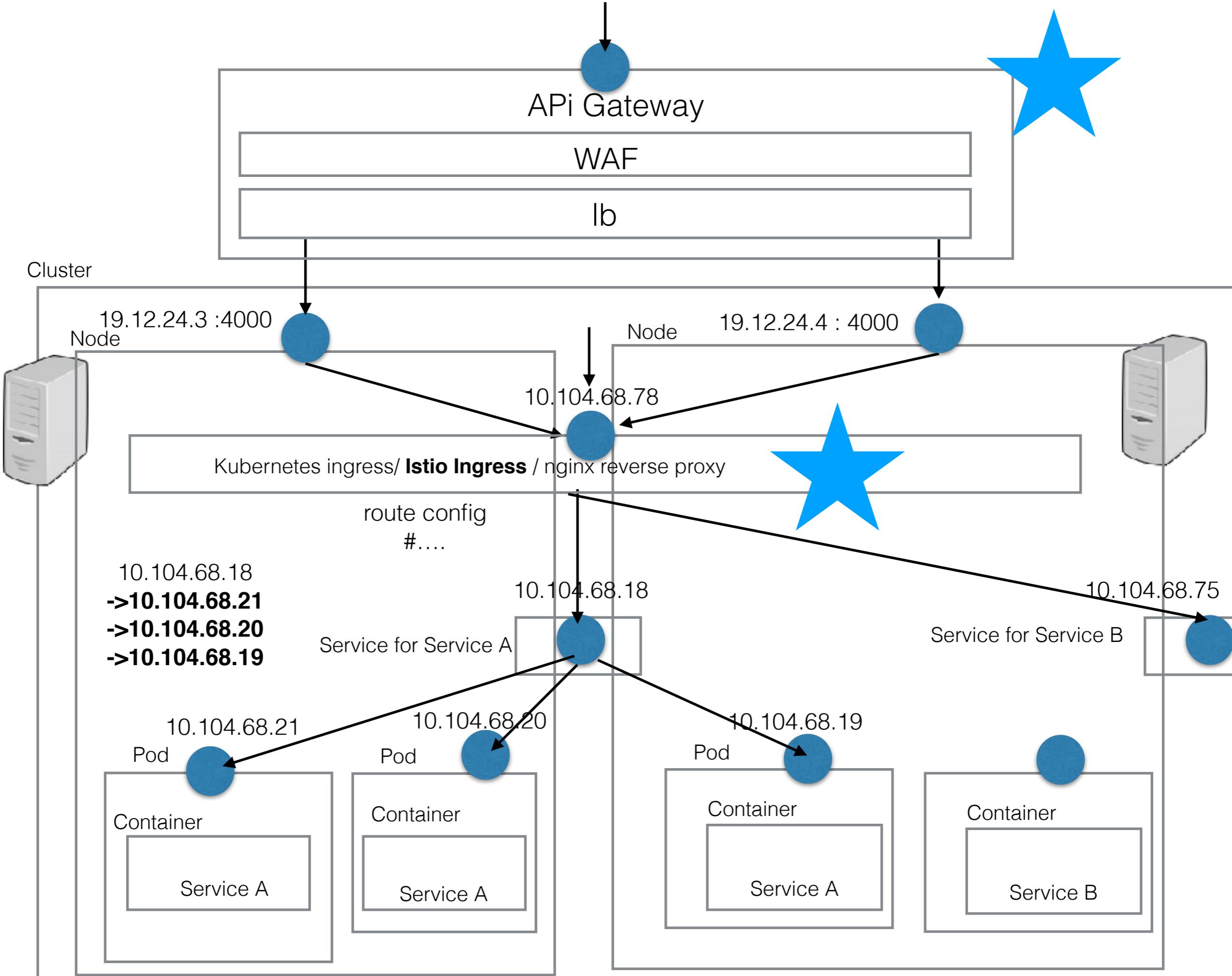


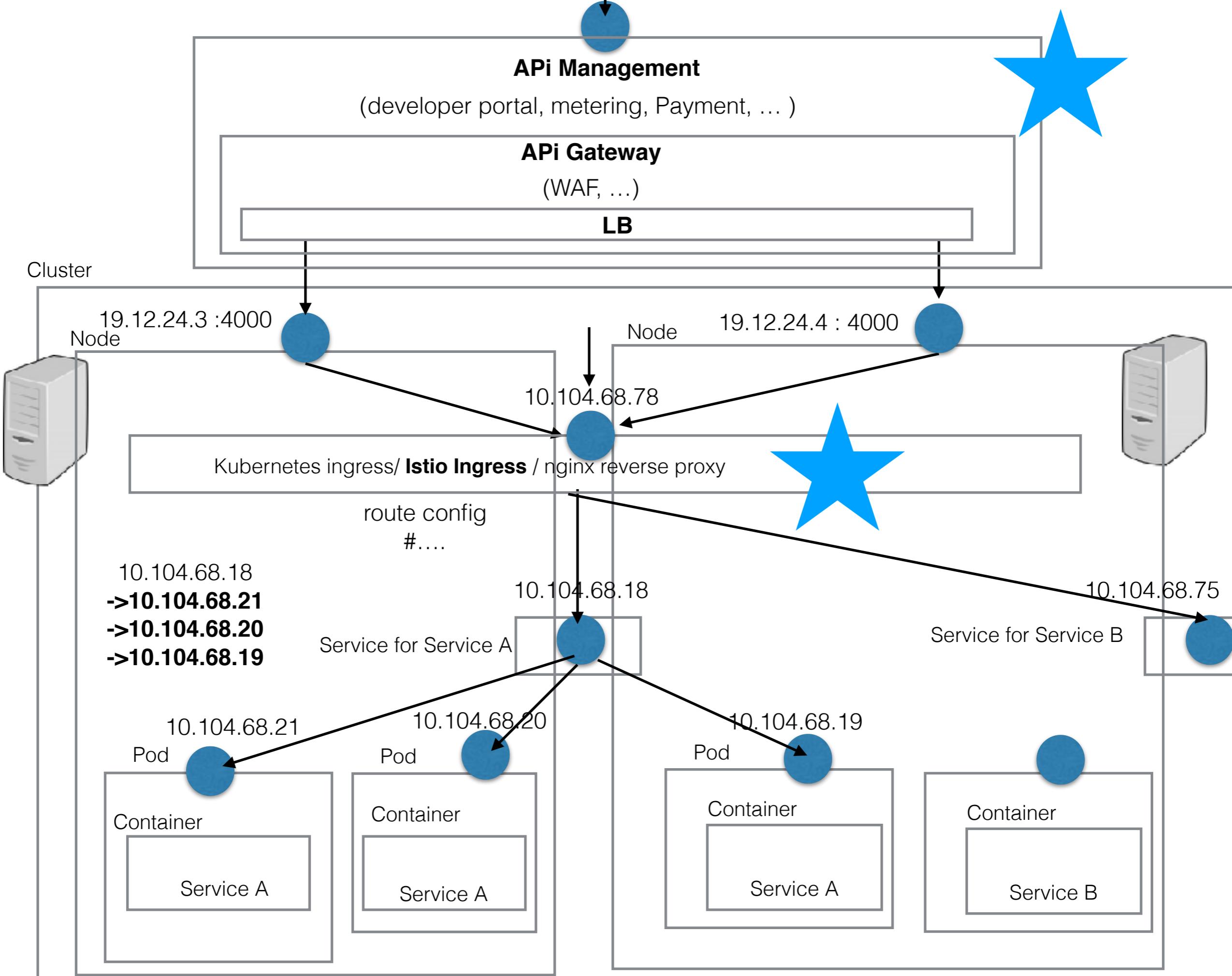


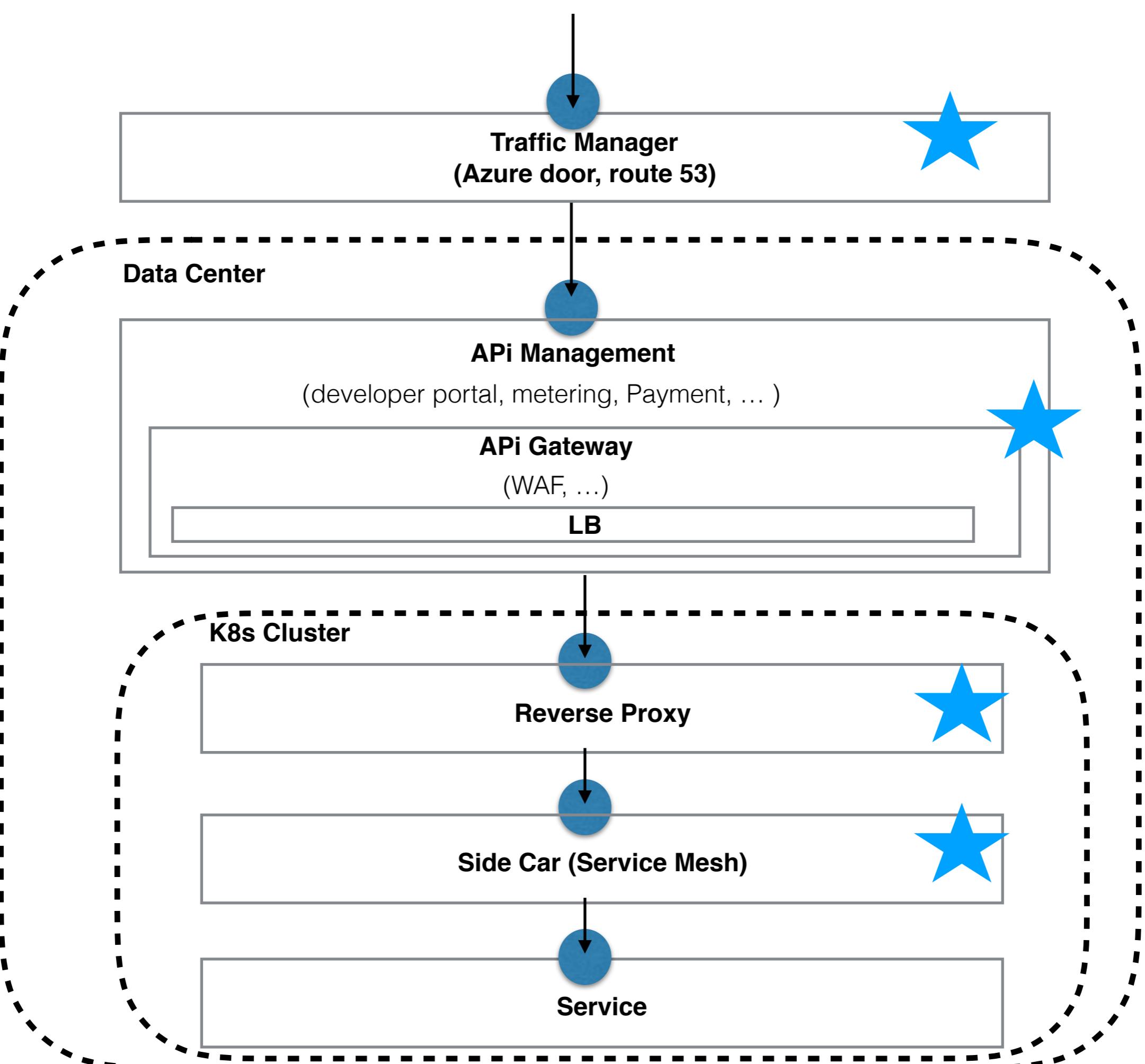




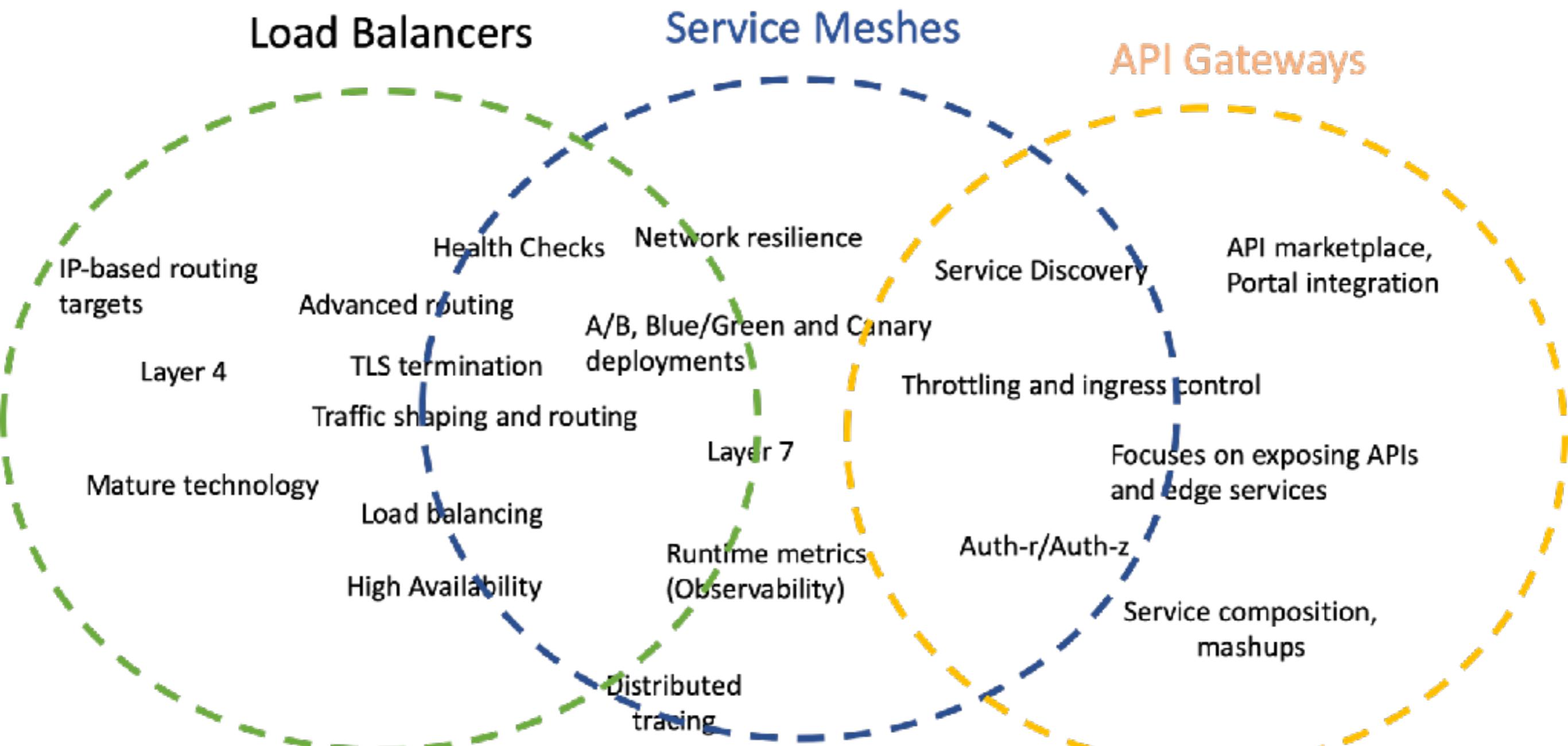


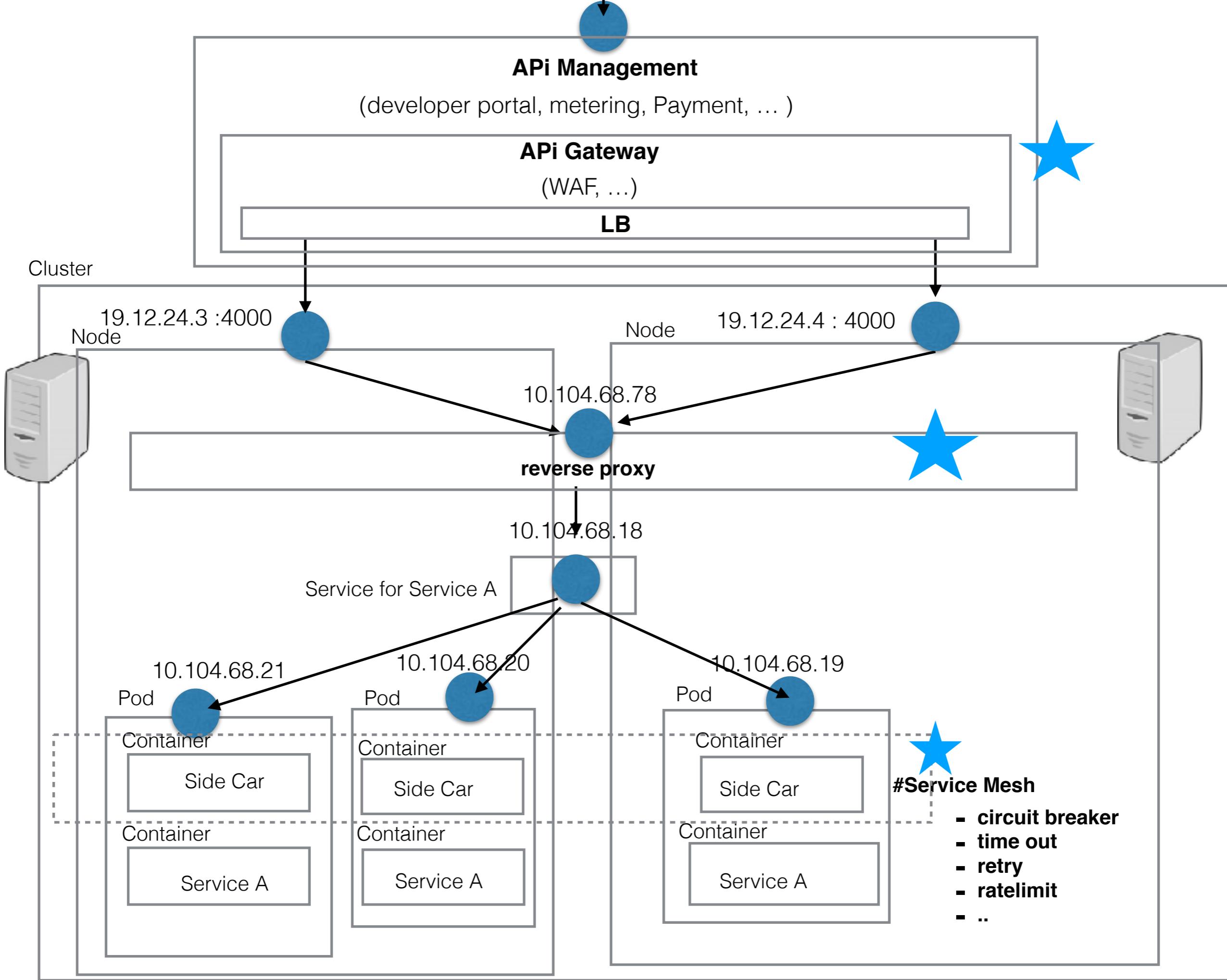


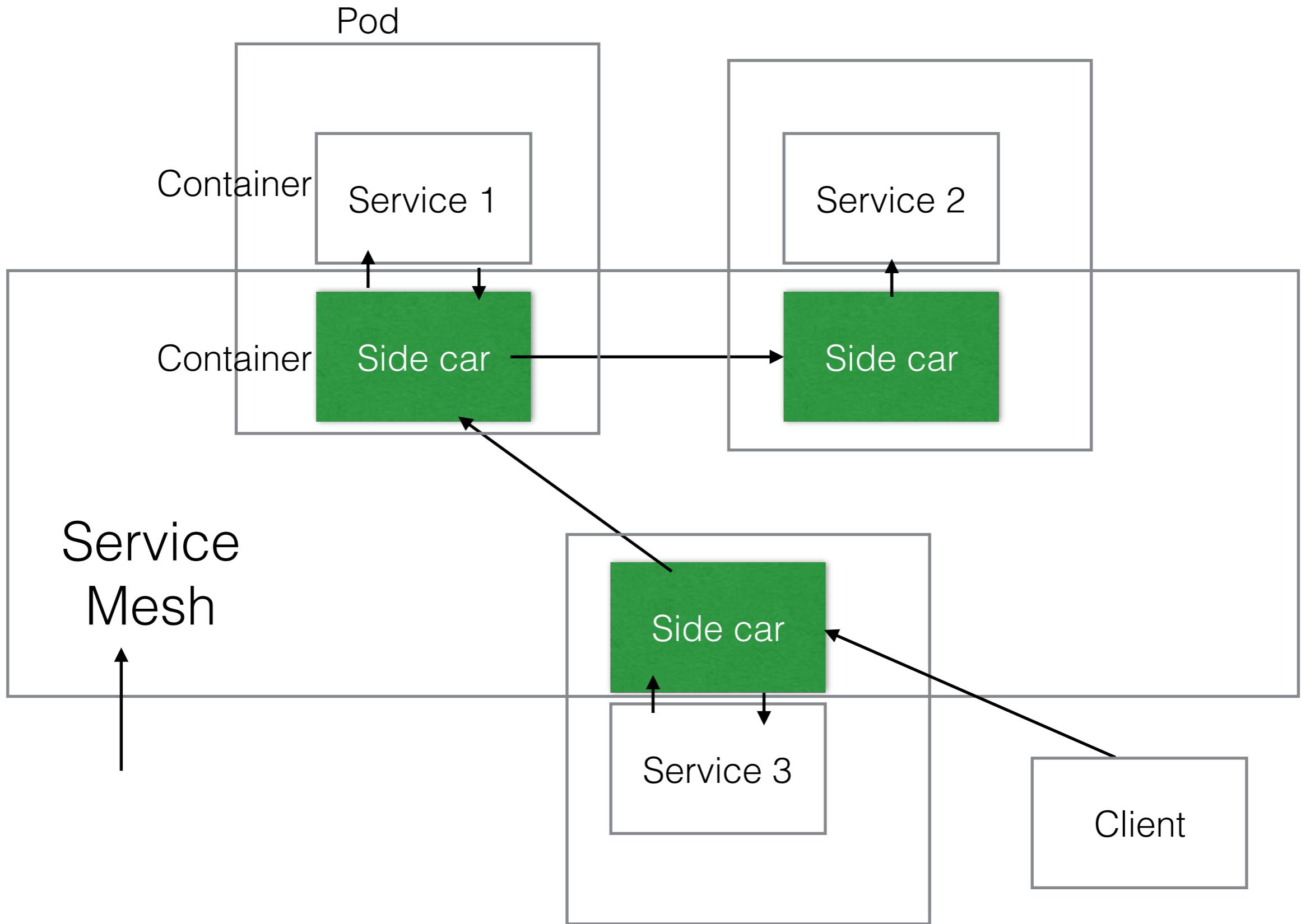




# Reverse Proxy/



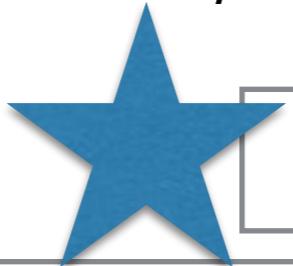




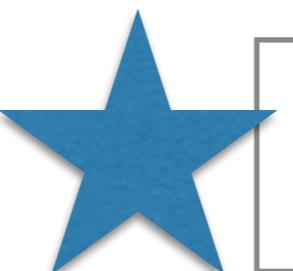
http://172.17.0.22:80/date

North- South

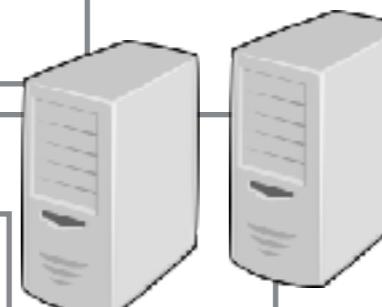
Cluster



LoadBalancer : L4 (tcp)



Reverse Proxy : L7  
(http)



Microservice A

http://10.97.33.128:8080/date

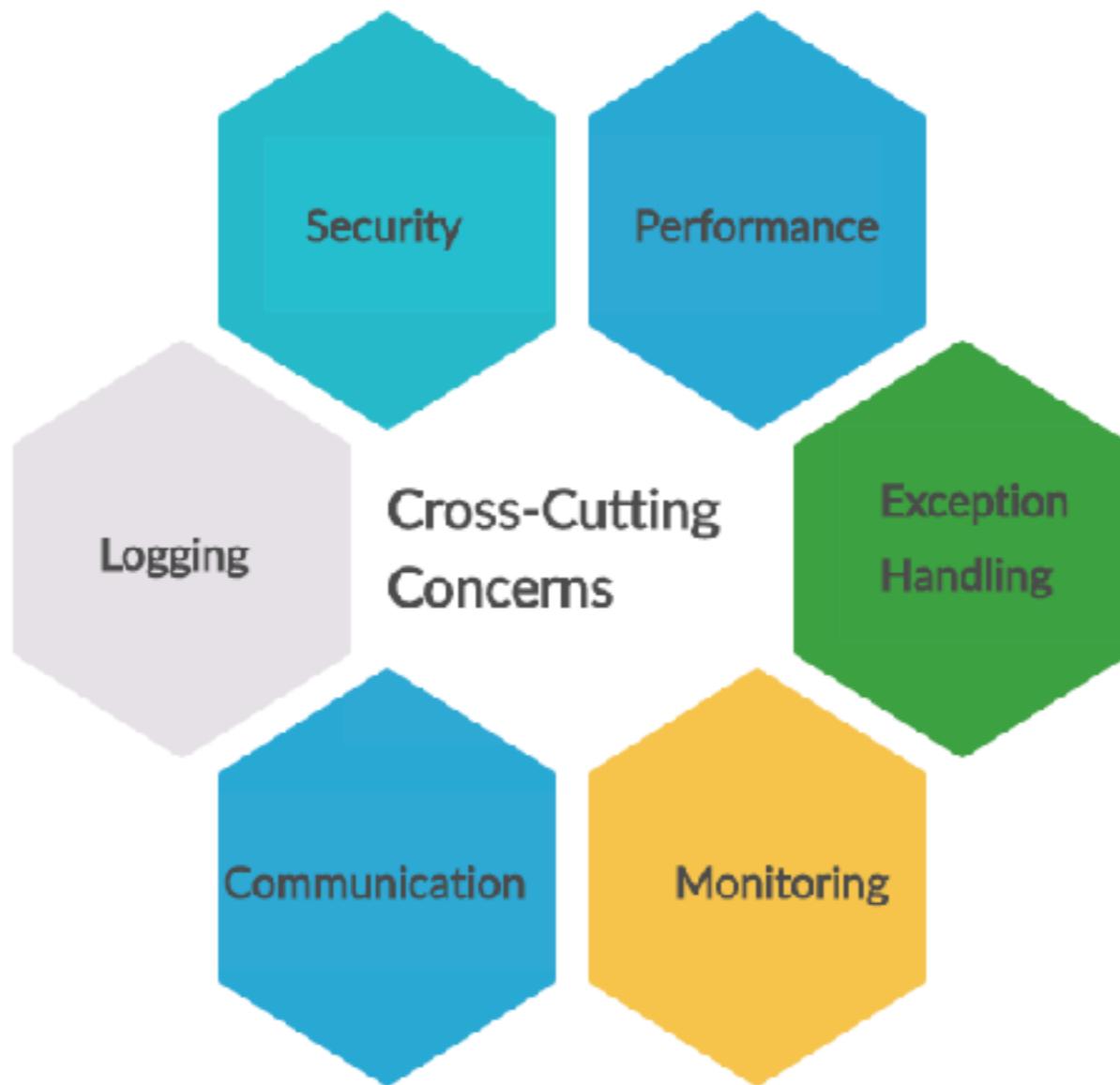
East - west

Service

Side car

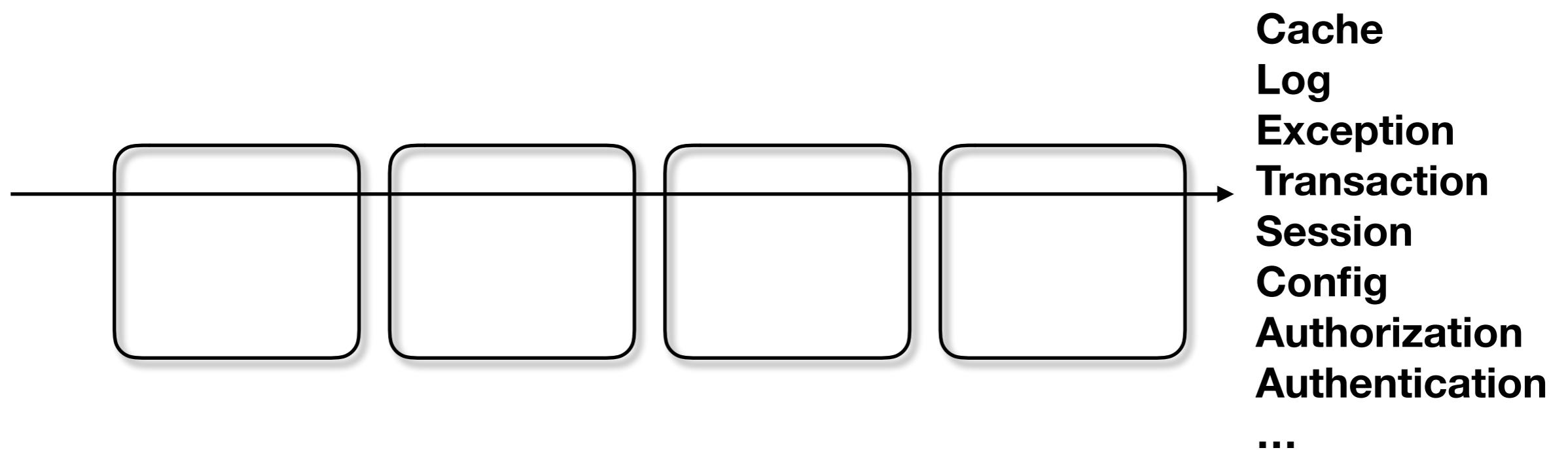
Microservice B

# Cross cutting concerns

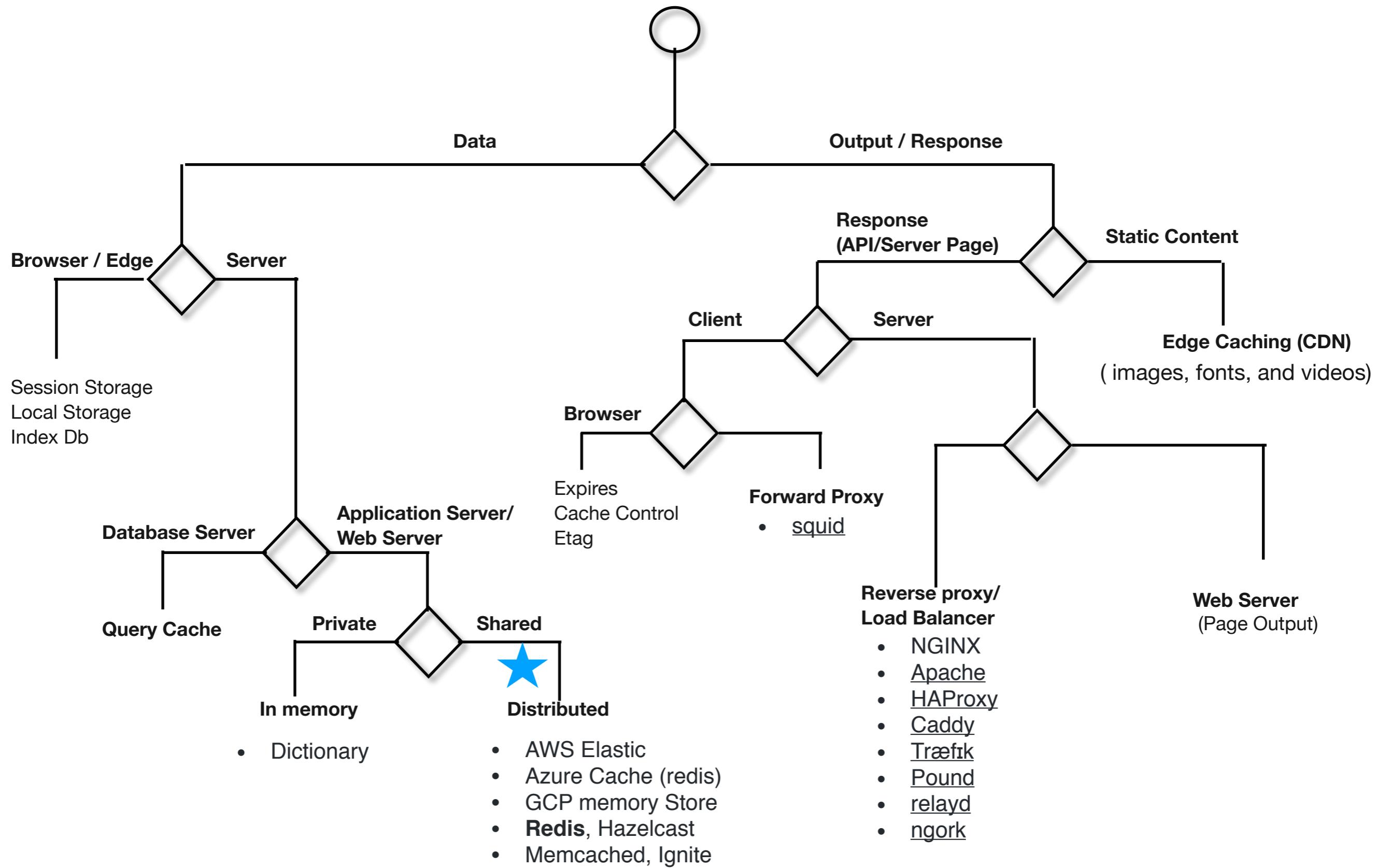


The **crosscutting concern** is a concern which is needed in almost every module of an application.

**Service**

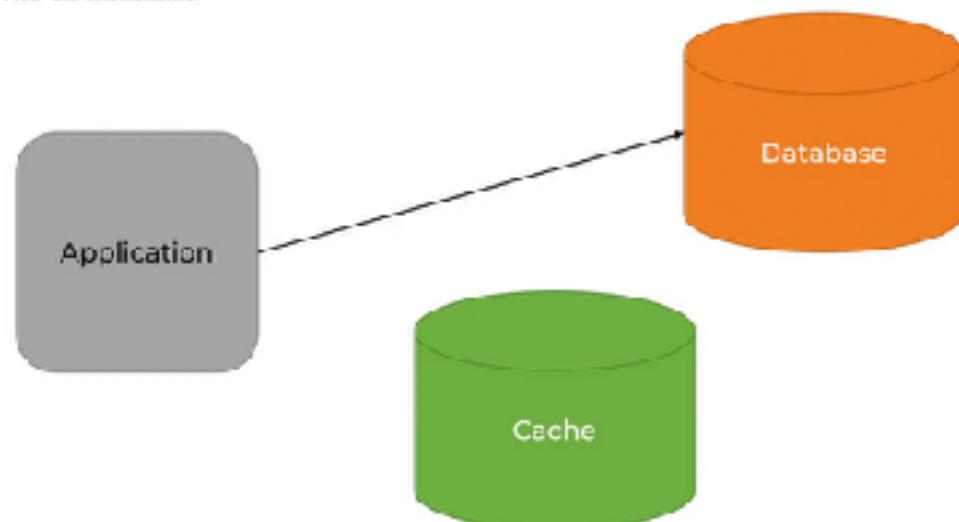


# Choose Cache



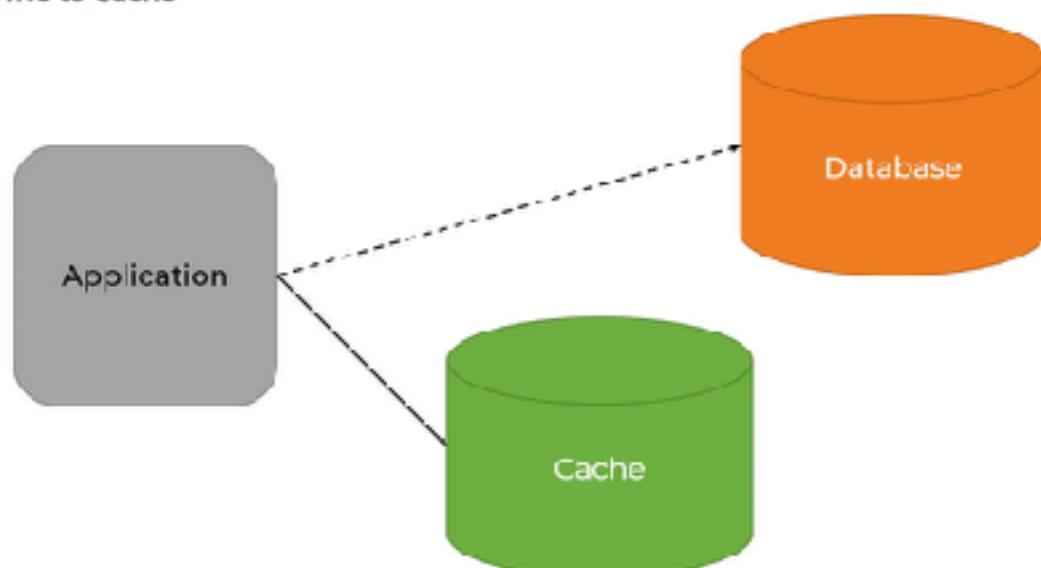
## Write-Through

Step 1: Write to Database



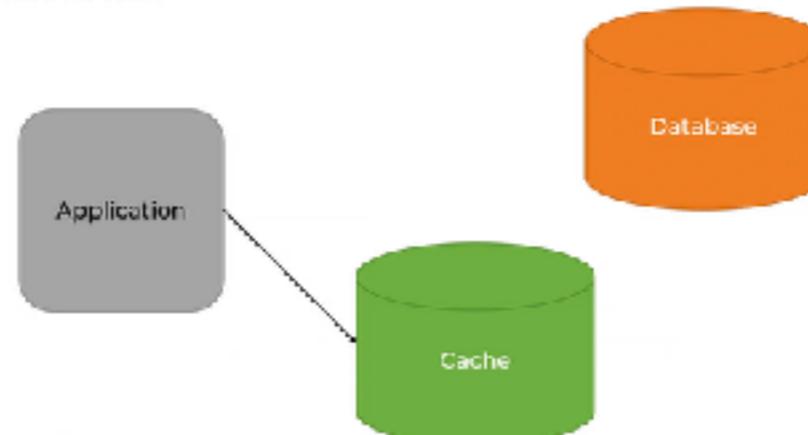
## Write-Through

Step 2: Write to Cache



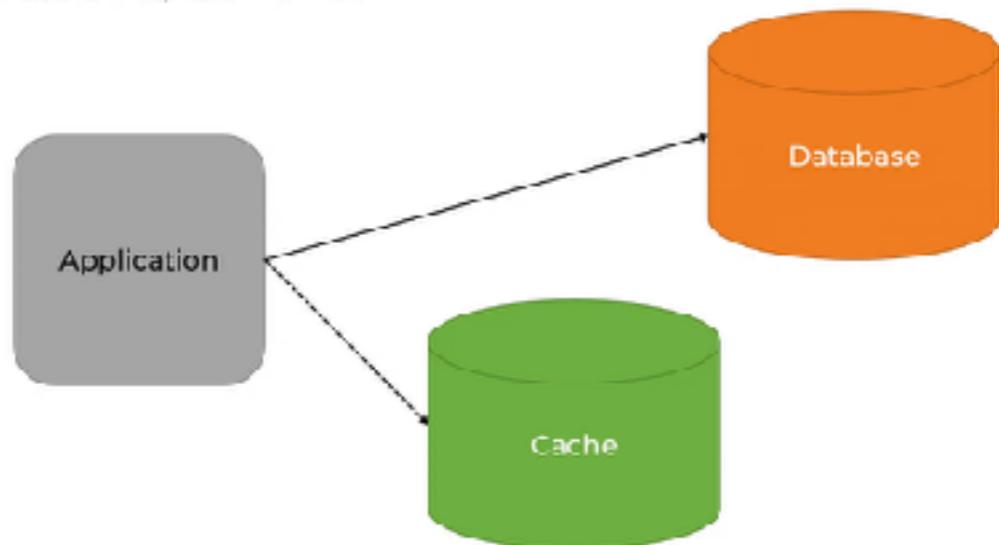
## Cache Aside (Lazy Loading)

Step 1: Read from Cache



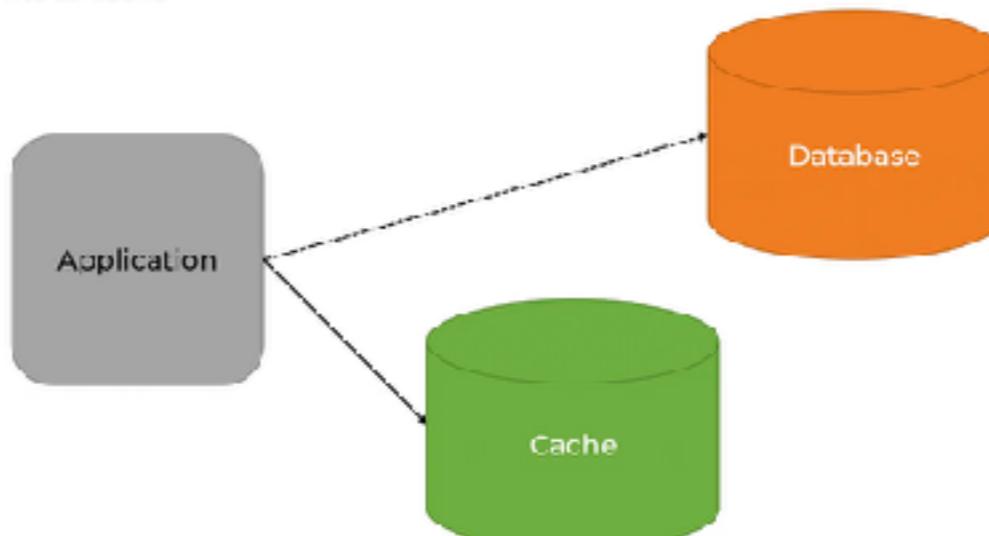
## Cache Aside (Lazy Loading)

Step 2: On Cache Miss, Fetch from DB

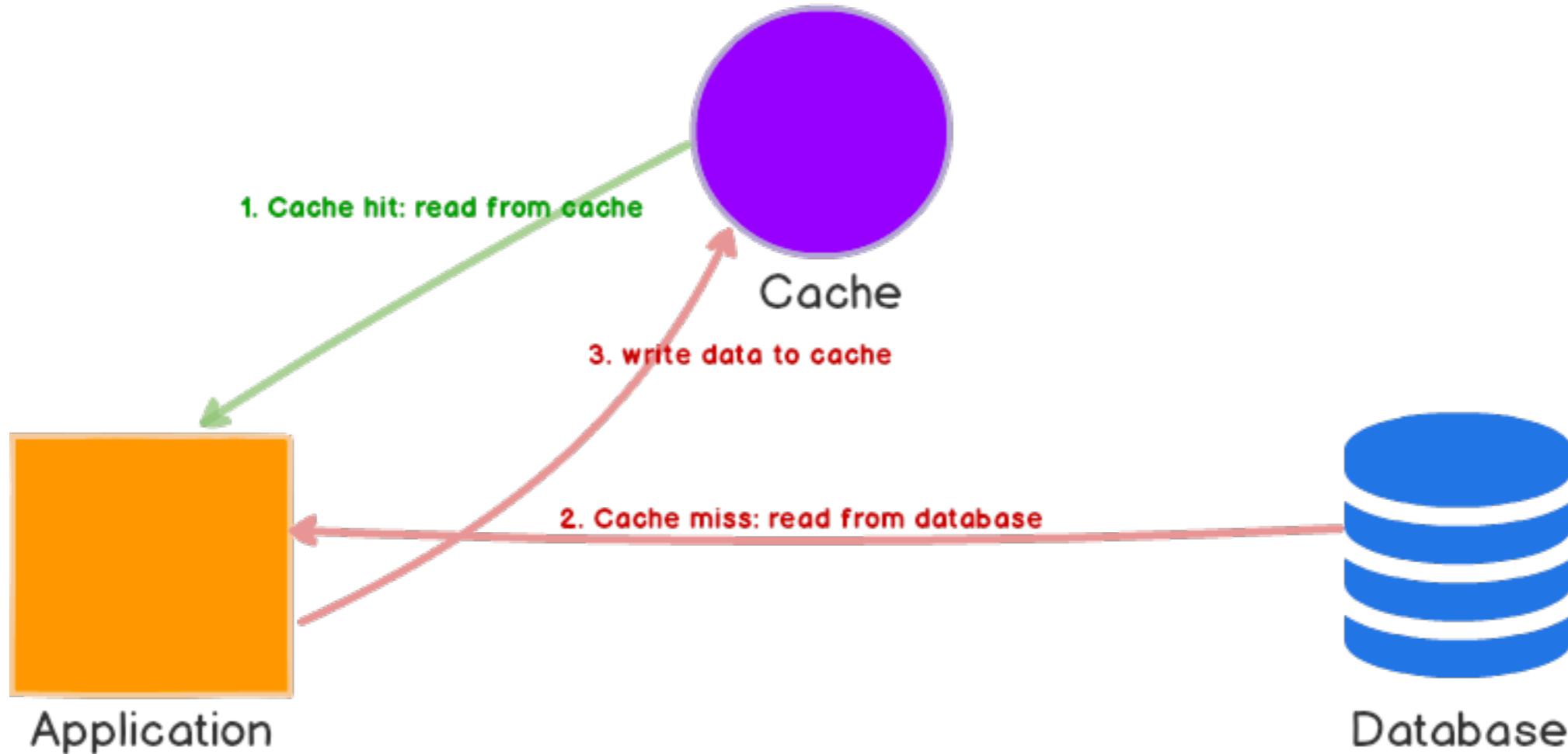


## Cache Aside (Lazy Loading)

Step 3: Write to Cache

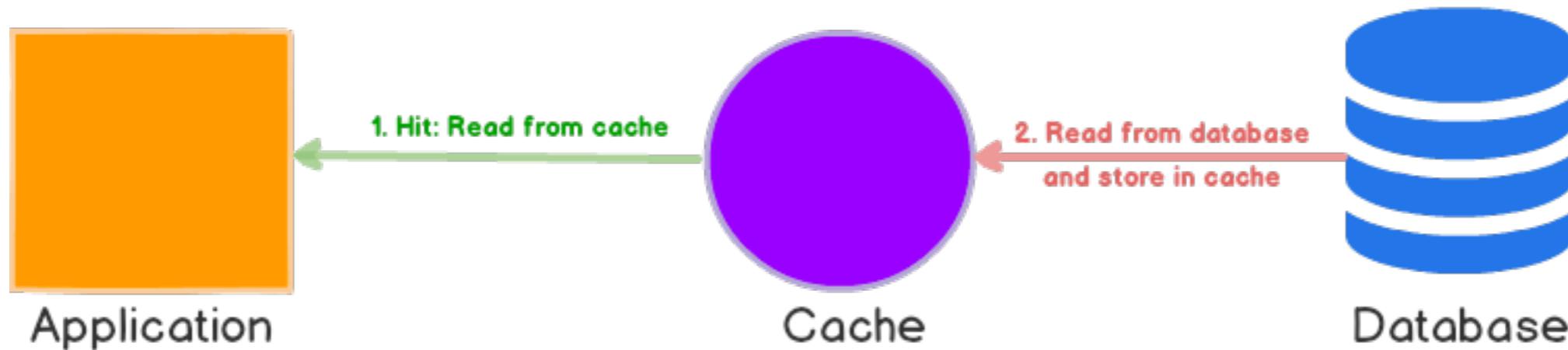


# Cache-Aside

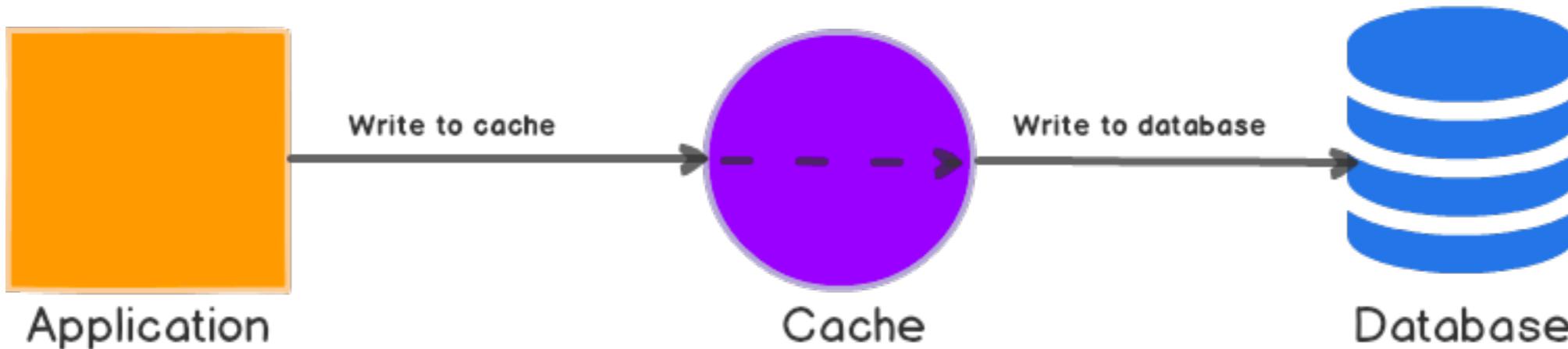


1. In cache-aside, the application is responsible for fetching data from the database and populating the cache. In read-through, this logic is usually supported by the library or stand-alone cache provider.
2. Unlike cache-aside, the data model in read-through cache cannot be different than that of the database.

## Read-Through

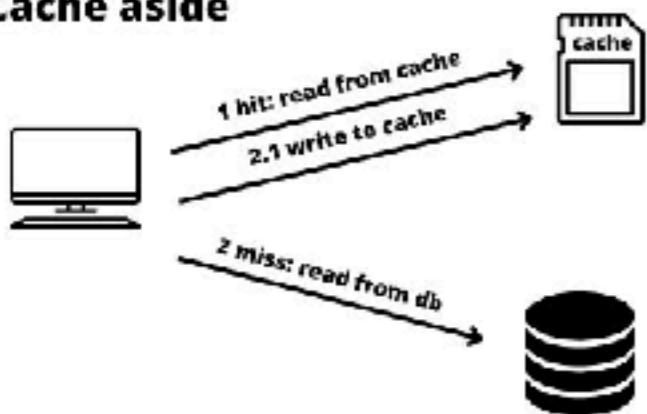


## Write-Through

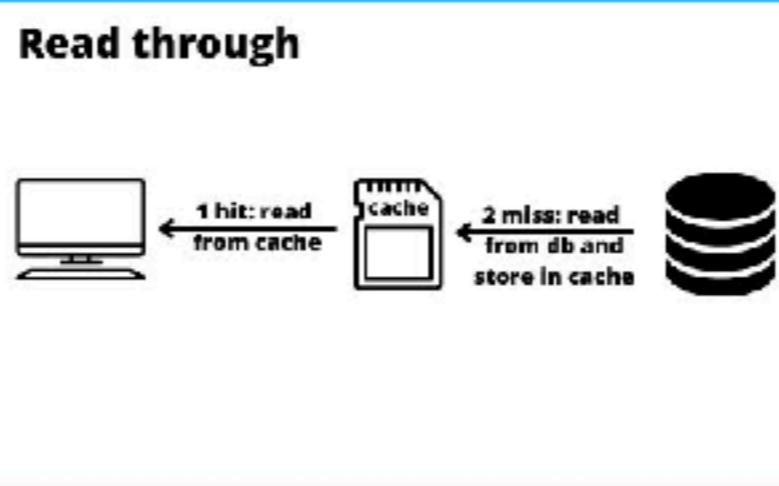


# Caching strategies

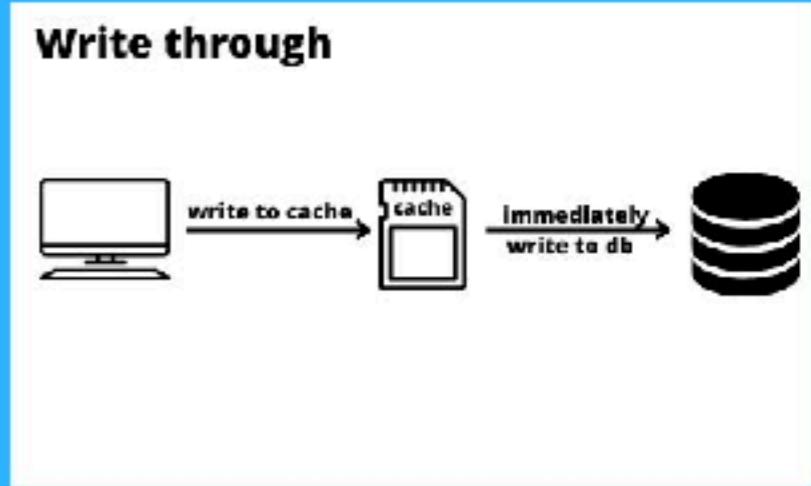
**Cache aside**



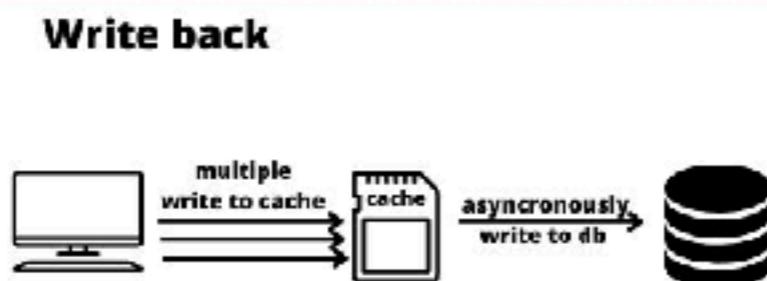
**Read through**



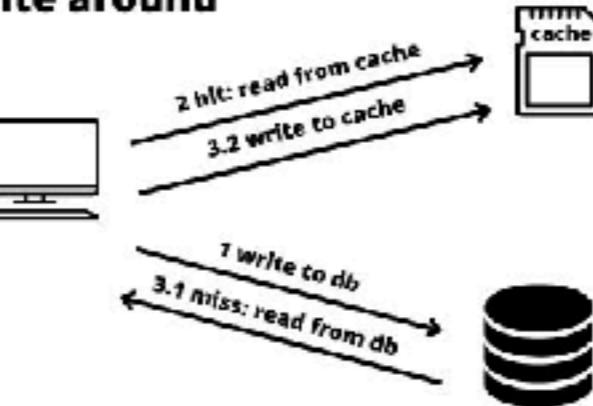
**Write through**



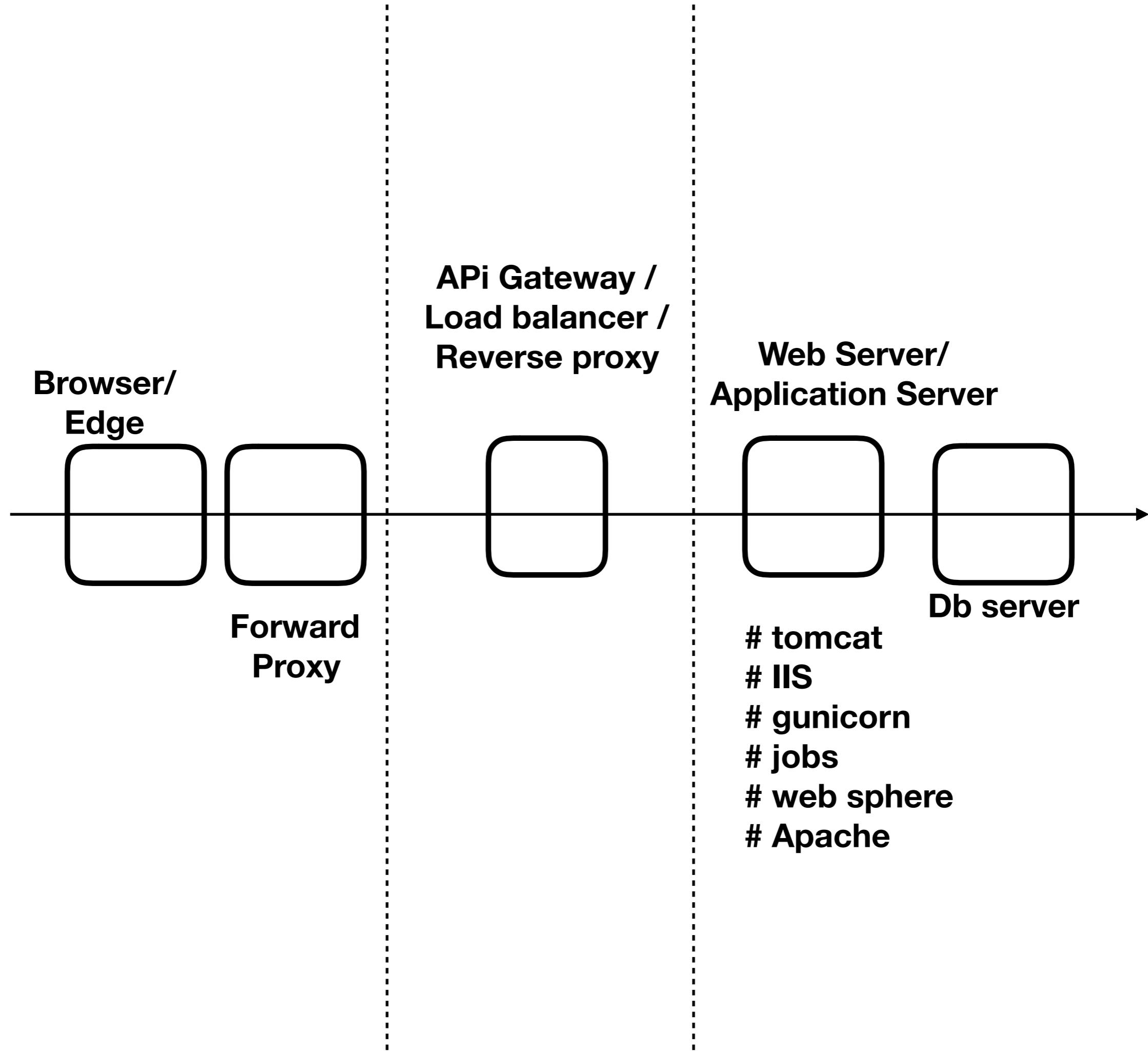
**Write back**



**Write around**



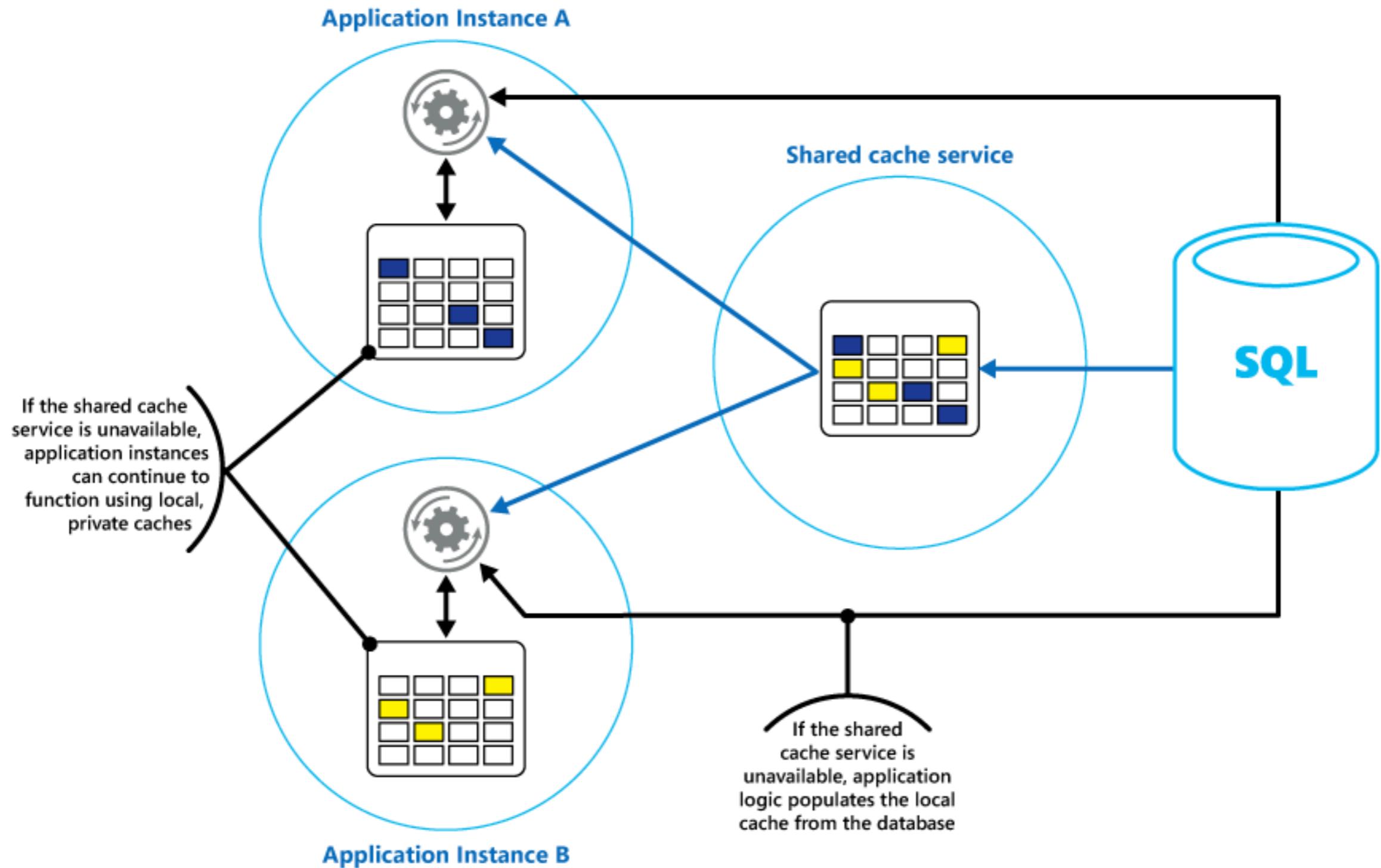
Franco Fernando  
@franc0fernando0



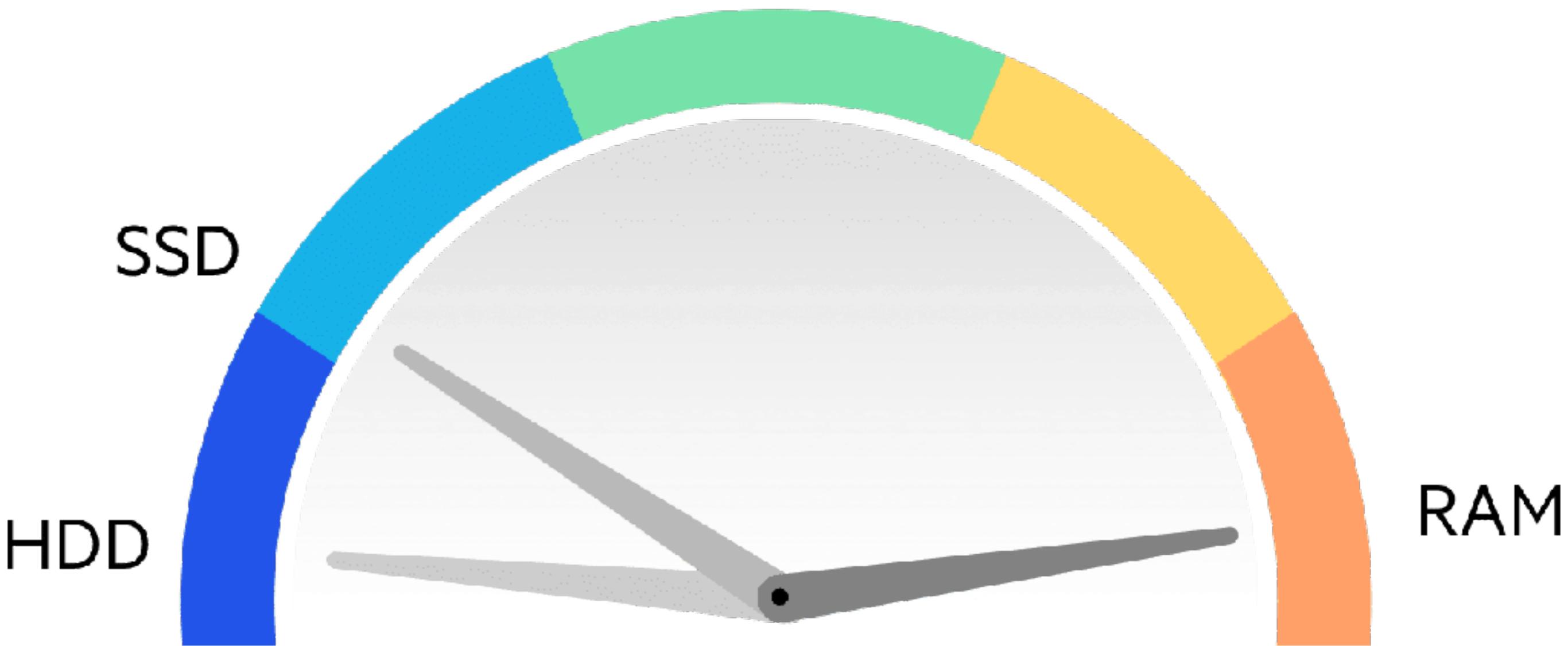
## Web acceleration

- SSL/TLS Processing
- Compression
- Caching and Prefetching

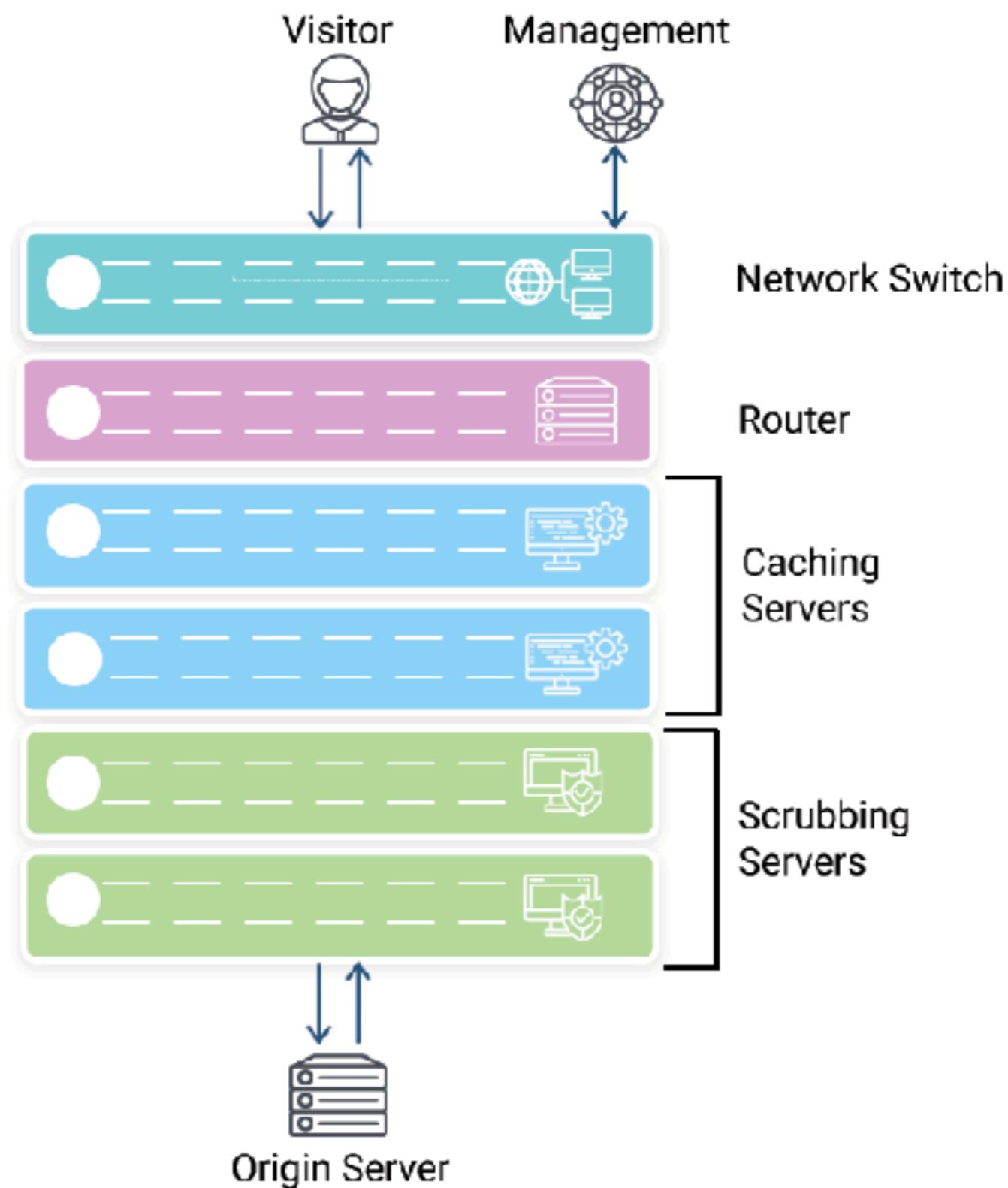
# *Using a local private cache with a shared cache.*



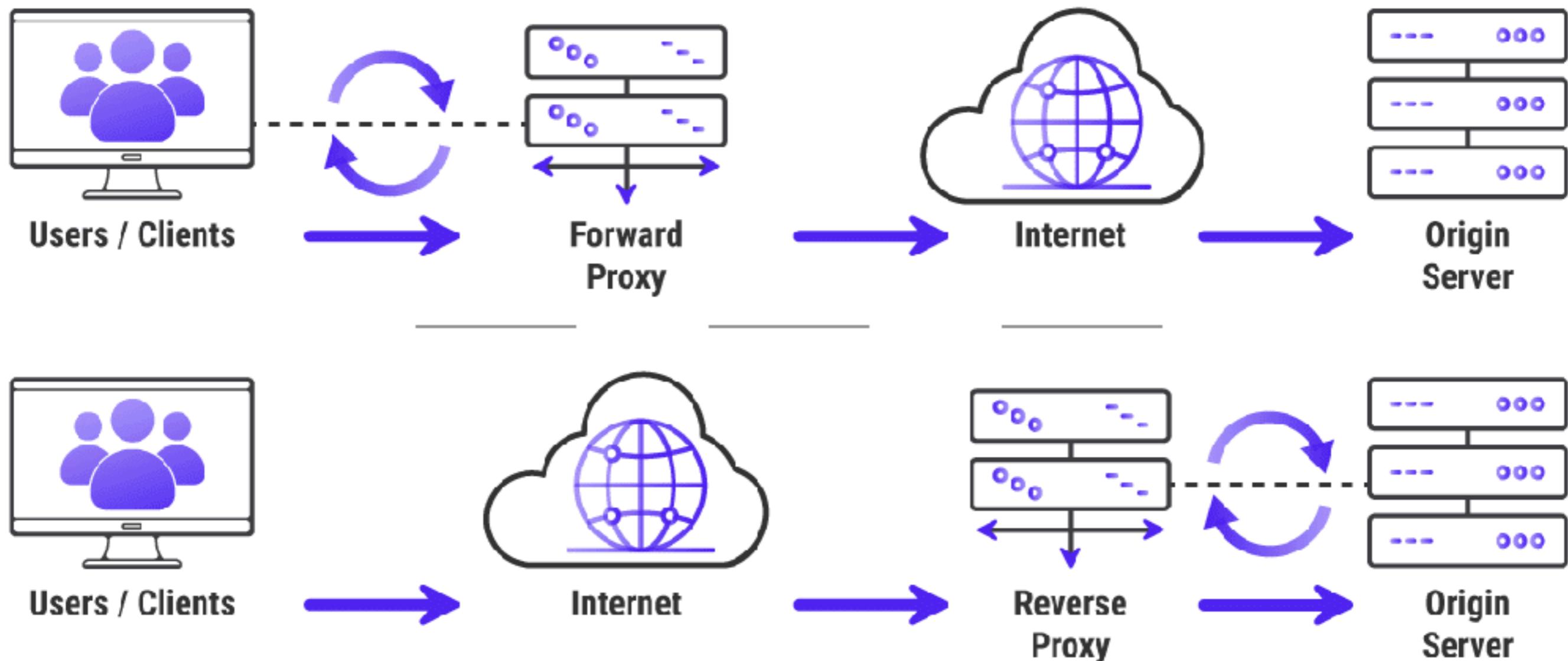
# Speed Comparison



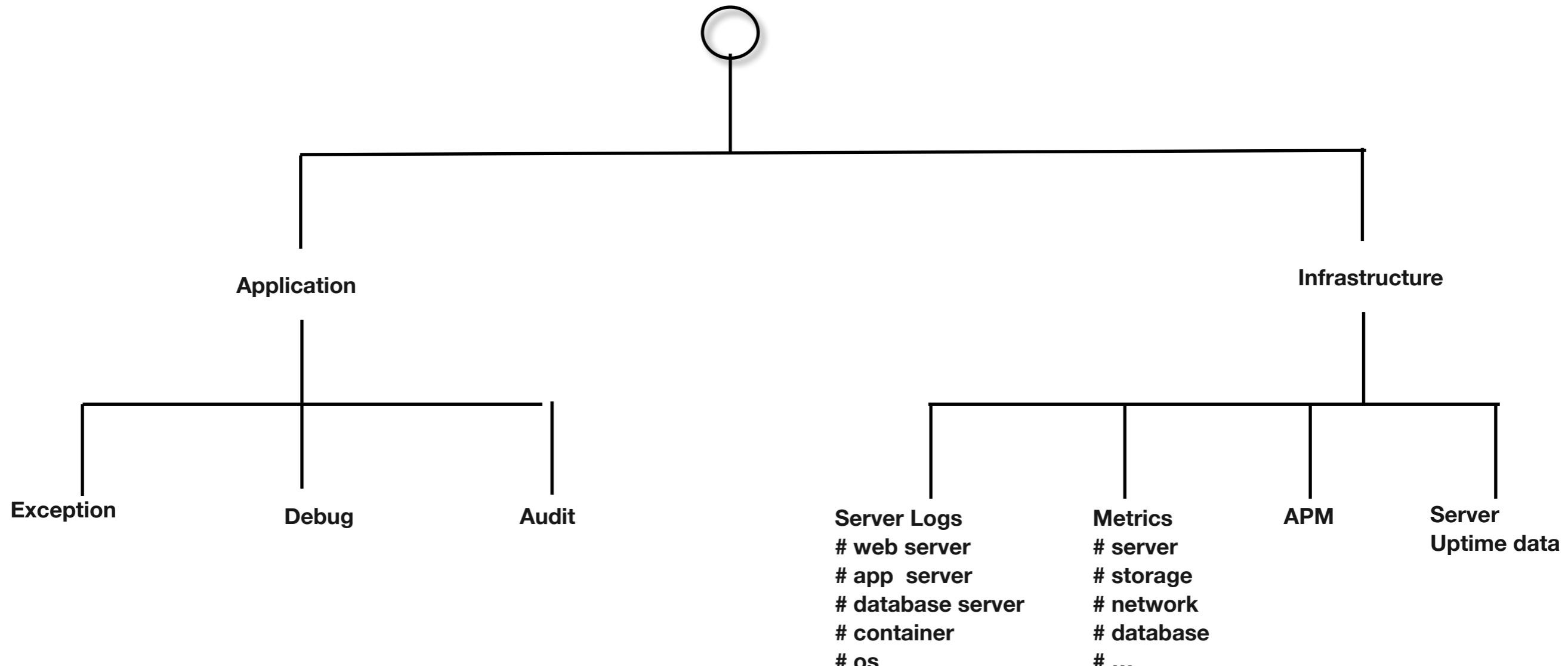
# HOW CONTENT DELIVERY NETWORK FUNCTIONS



# Forward Proxy vs Reverse Proxy

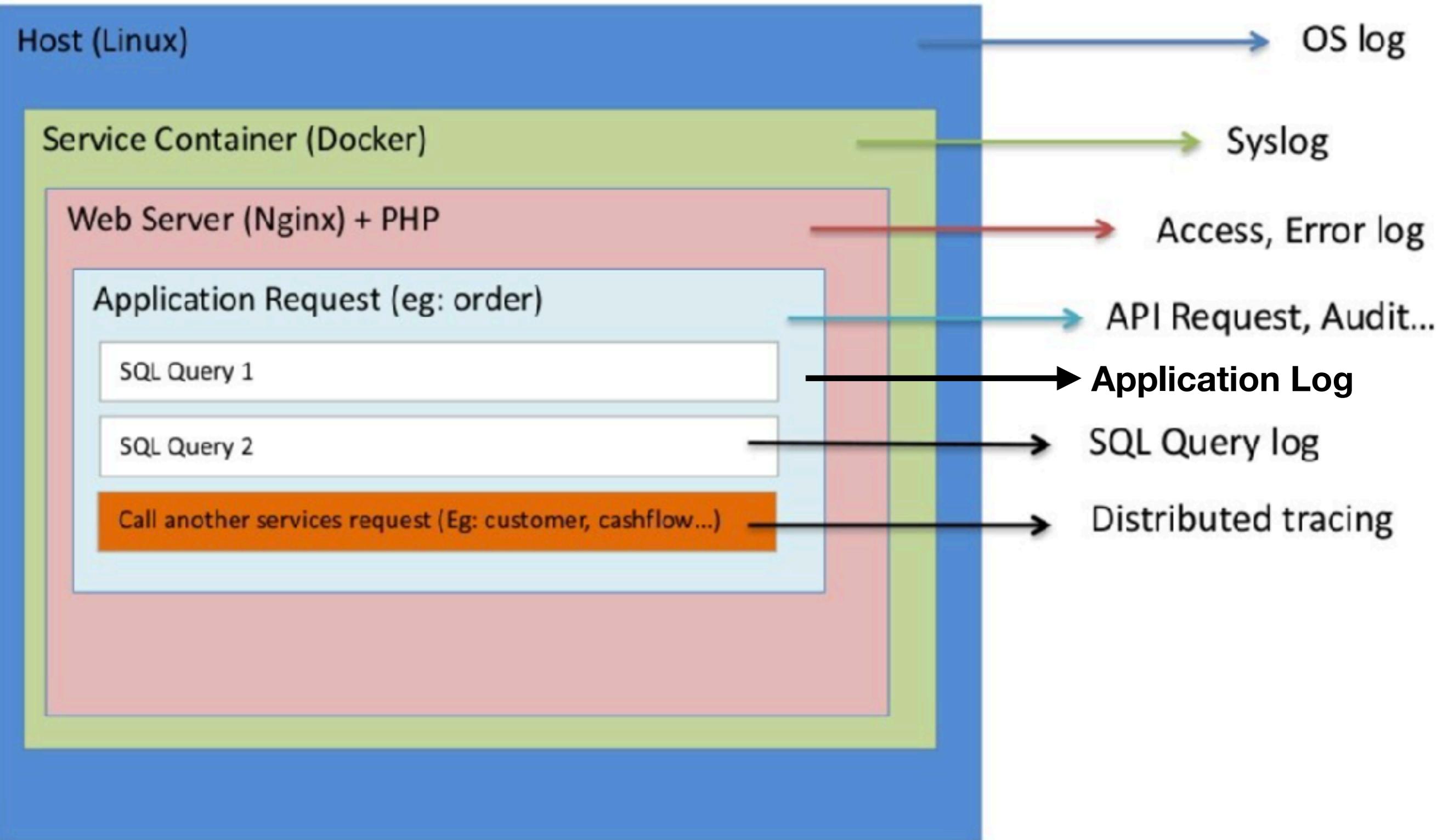


# Logs



Streaming Logs

Jaeger  
Elastic APM



## Dev & Ops Teams



### Log Data

Web Logs  
App Logs  
Database Logs  
Container Logs

### Metrics Data

Container Metrics  
Host Metrics  
Database Metrics  
Network Metrics  
Storage Metrics

### APM Data

Real User Monitoring  
Txn Perf Monitoring  
Distributed Tracing

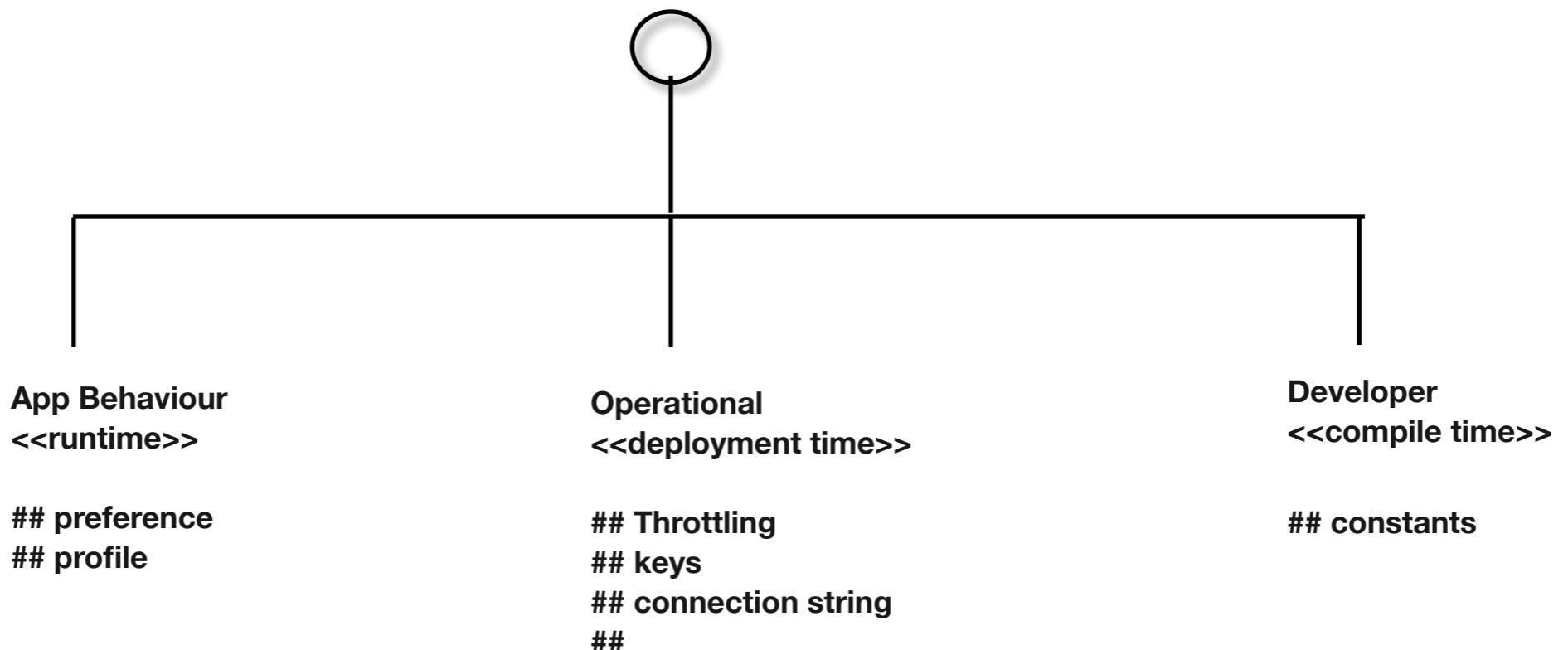
### Uptime Data

Uptime  
Response Time

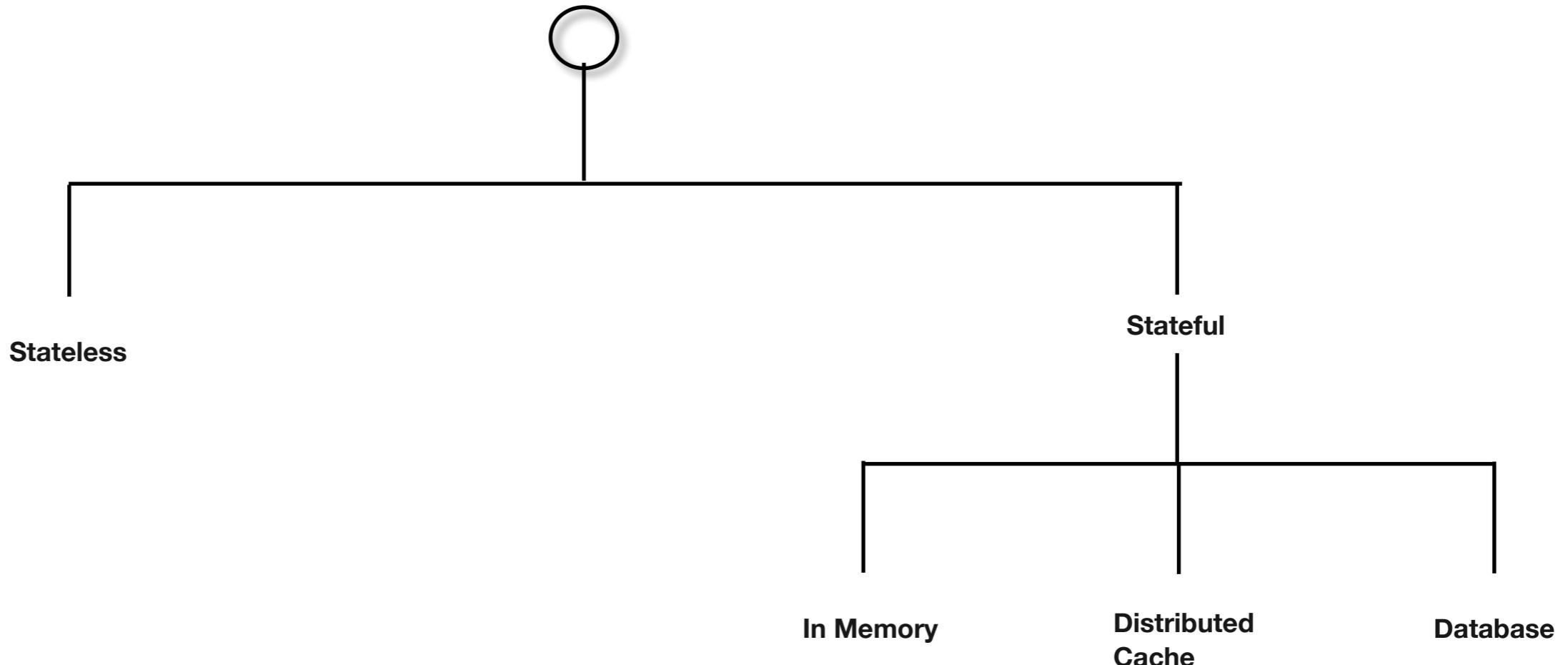
**Activity**

**Resource**

# Config



# State Management



# Concurrency View

# Engineering for Performance

## Performance Objectives

(Response Time, Throughput, Resource Utilization, Workload)

## Performance Modeling

(Scenarios, Objectives, Workloads, Requirements, Budgets, Metrics)

## Architecture and Design Guidelines

(Principles, Practices and Patterns)

## Performance and Scalability Frame

Coupling and Cohesion  
Communication  
Concurrency

Resource Management  
Caching, State Management  
Data Structures / Algorithms

## Measuring, Testing, Tuning

**Measuring**  
Response Time  
Throughput  
Resource Utilization  
Workload

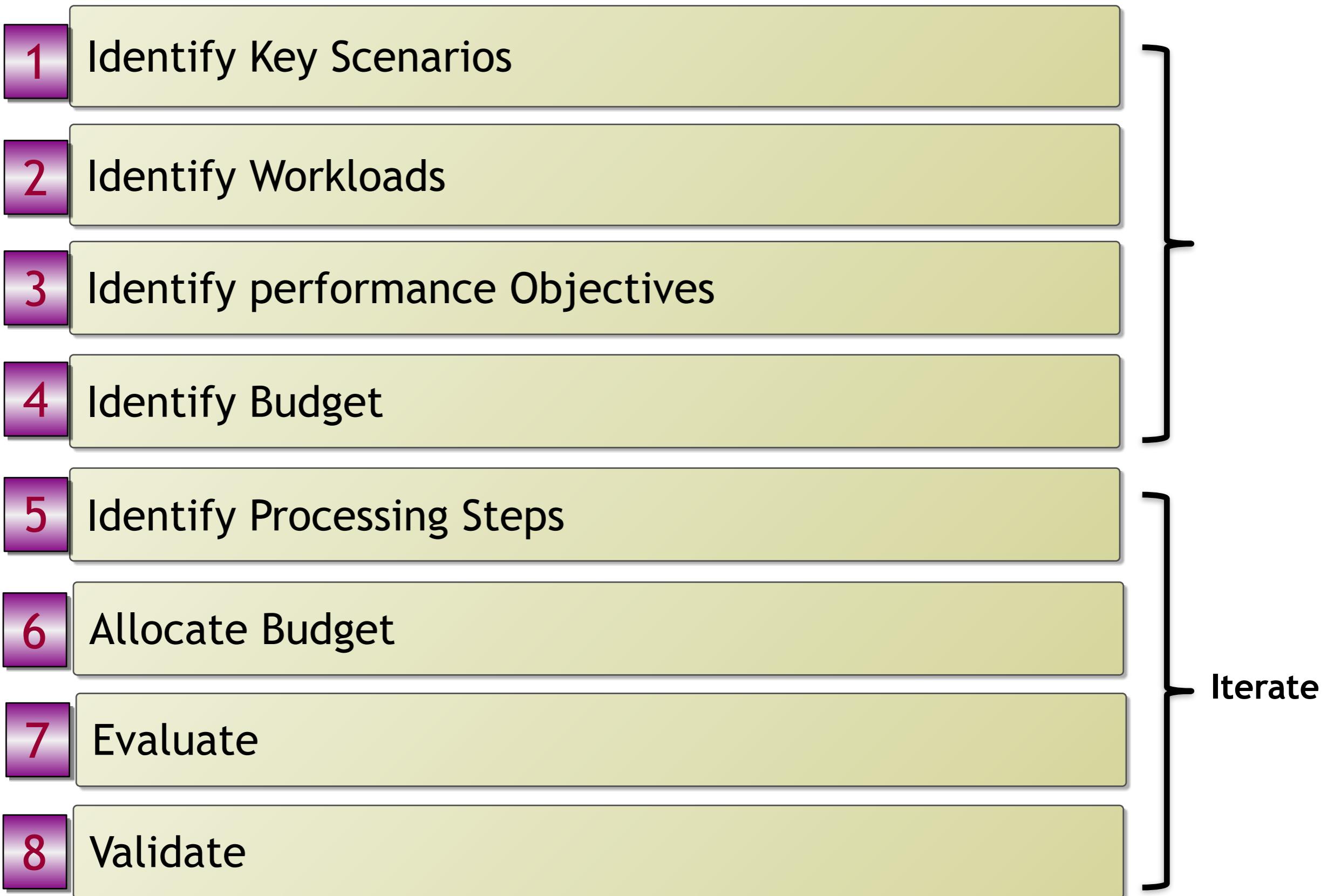
**Testing**  
Load Testing  
Stress Testing  
Capacity Testing

**Tuning**  
Network  
System  
Platform  
Application

**Roles**  
(Architects, Developers, Testers, Administrators)

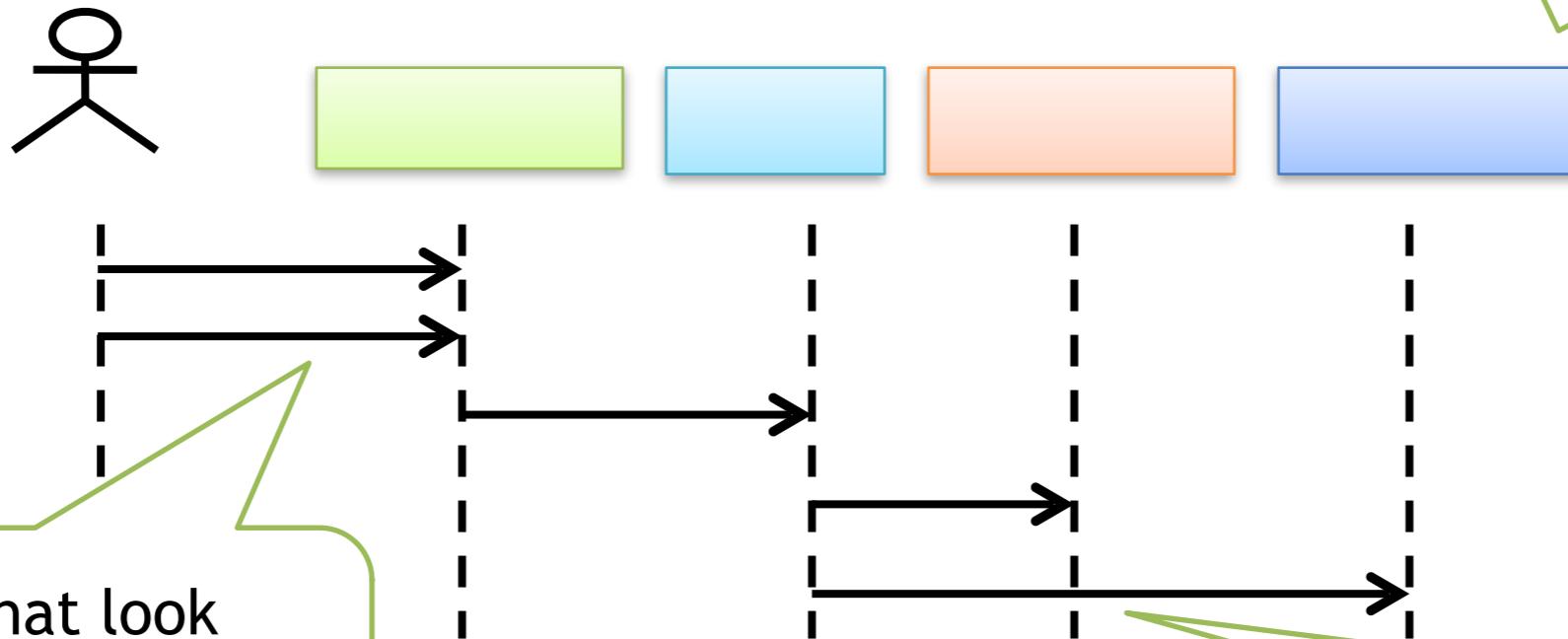
**Life Cycle**  
(Requirements, Design, Develop, Test, Deploy, Maintain)

# Performance Modeling



# Allocate Budget

Know the cost of your materials.

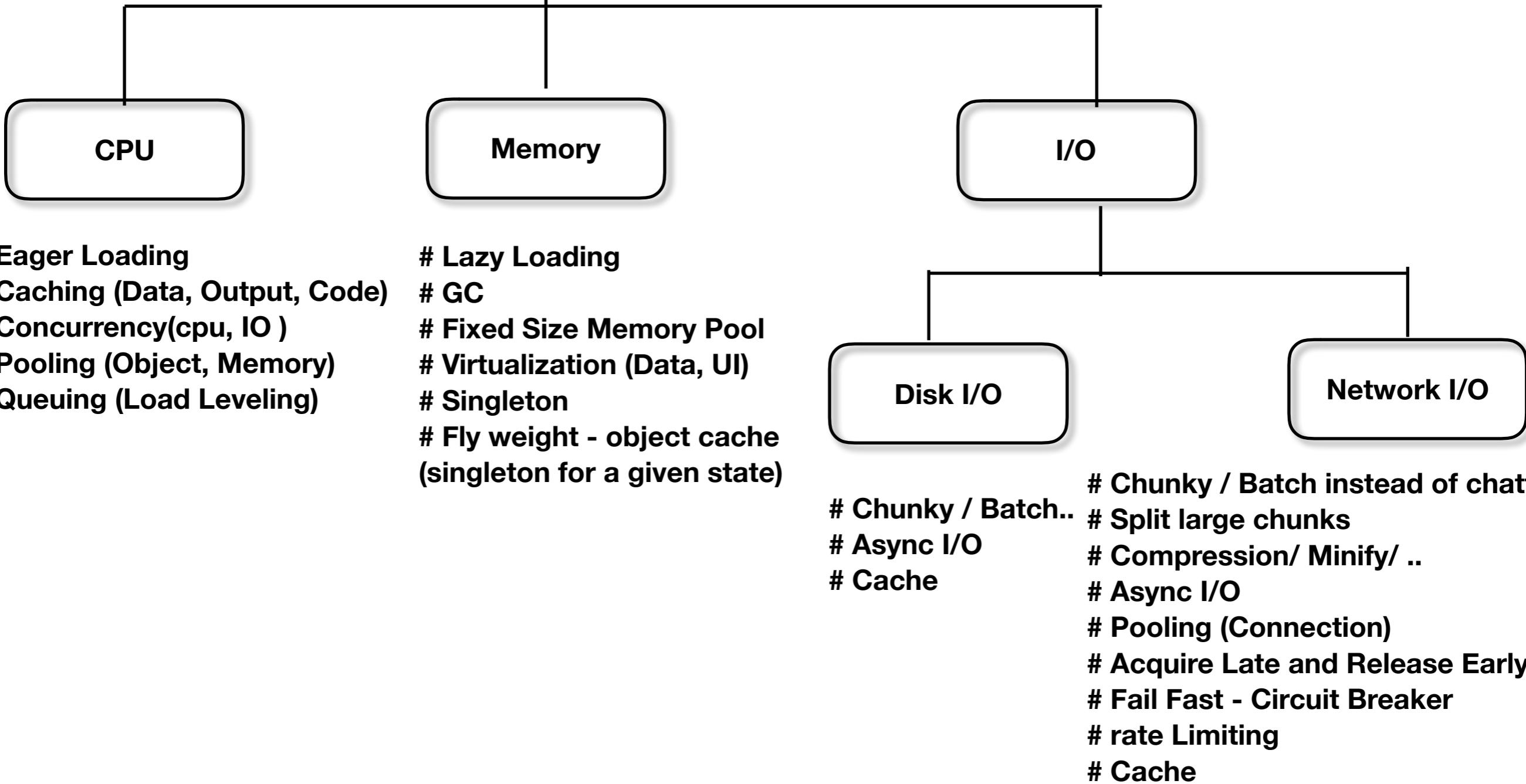
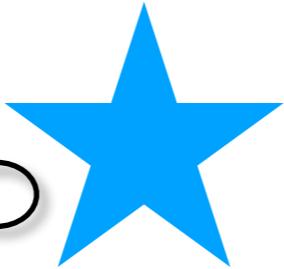


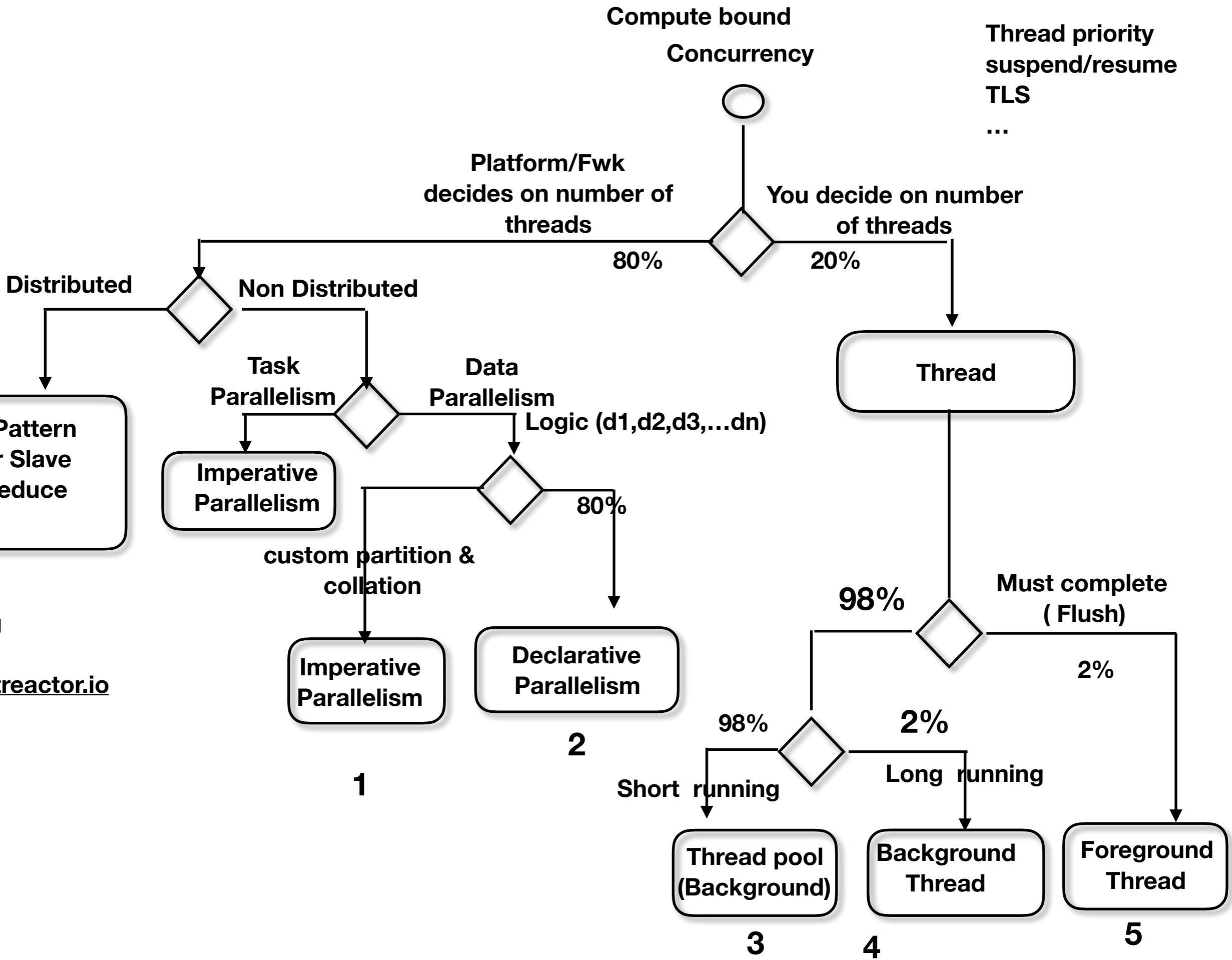
For the ones that look  
risky, conduct some  
experiments (prototypes)

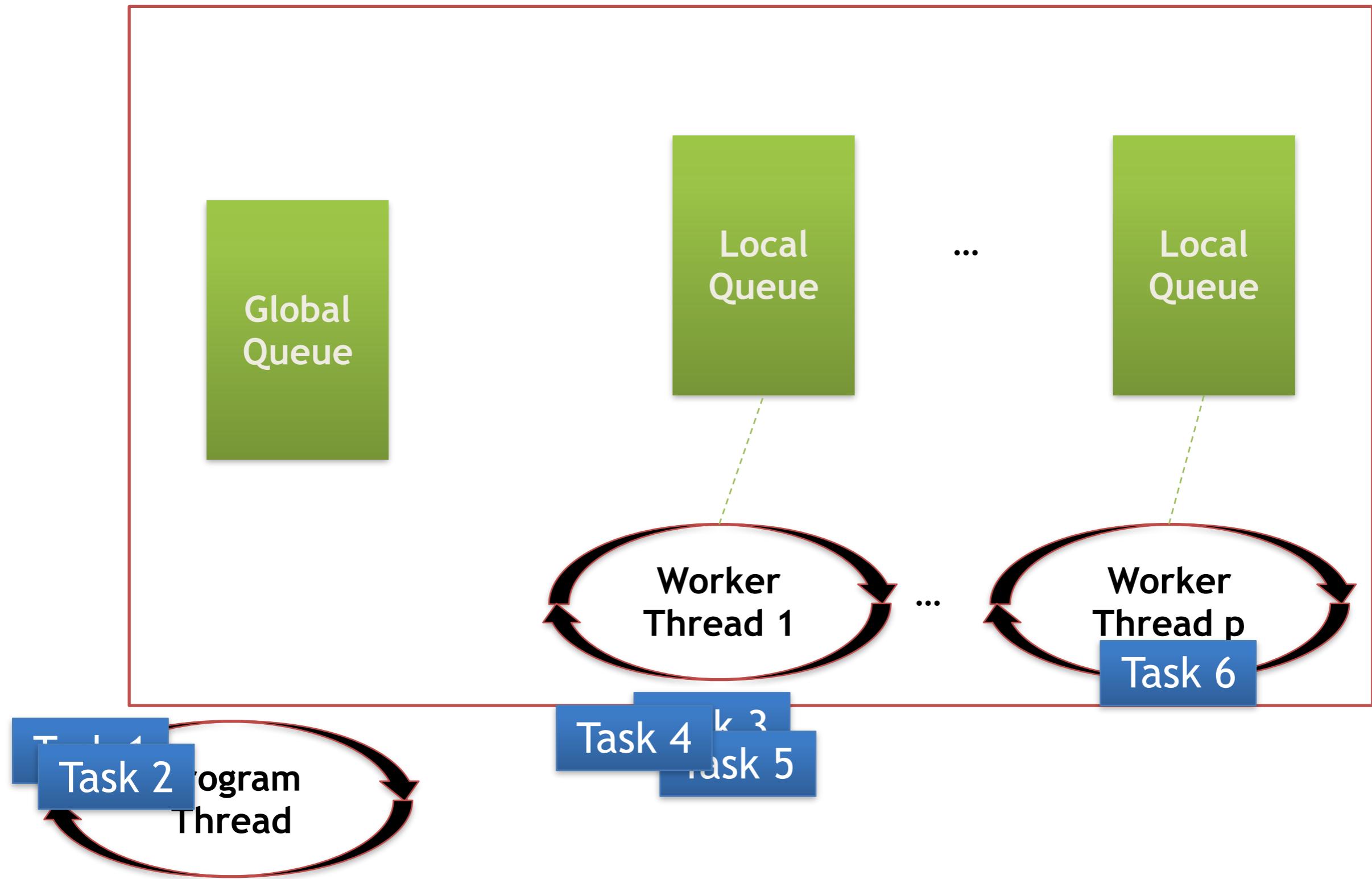
if you do not know how  
much time to assign, simply  
divide the total time  
equally between the steps.

Spread your budget across your processing steps

## Performance Patterns







```
foreach(var item in items)
{
    ThreadPool.Do(() => do(item));
}
```

v/s

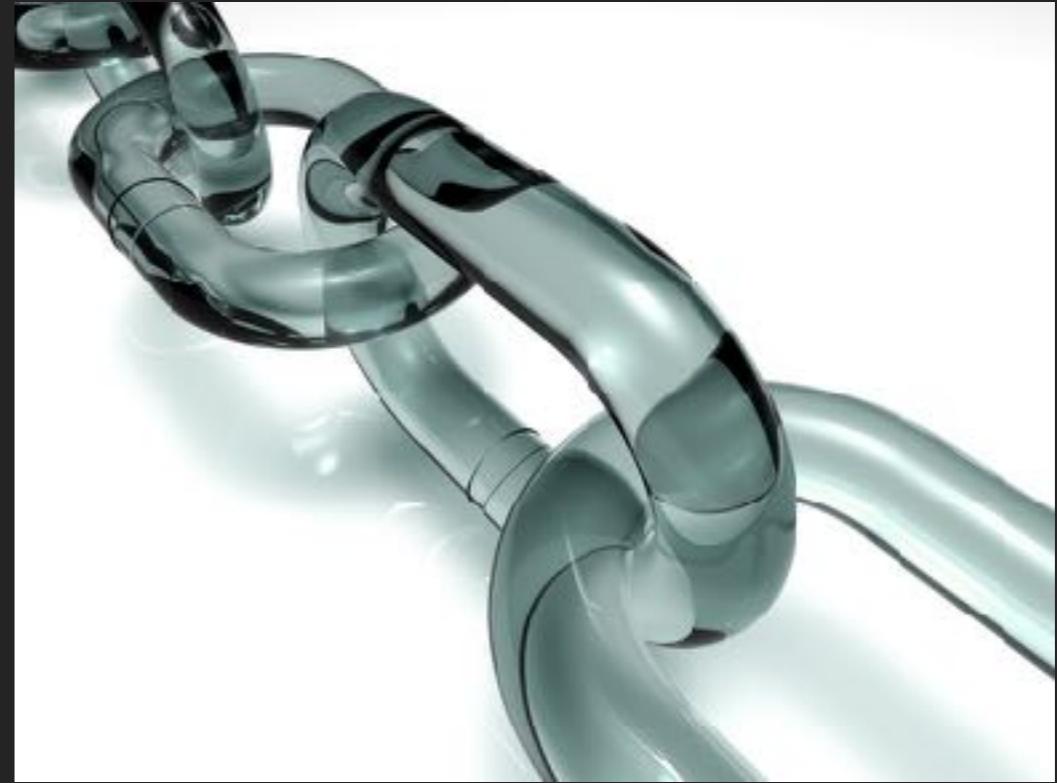
```
foreach(var item in items)
{
    Task.Factory.StartNew(() => do(item));
}
```

```
Parallel.ForEach<Item>(items, item  
=>do(item));
```

v/s

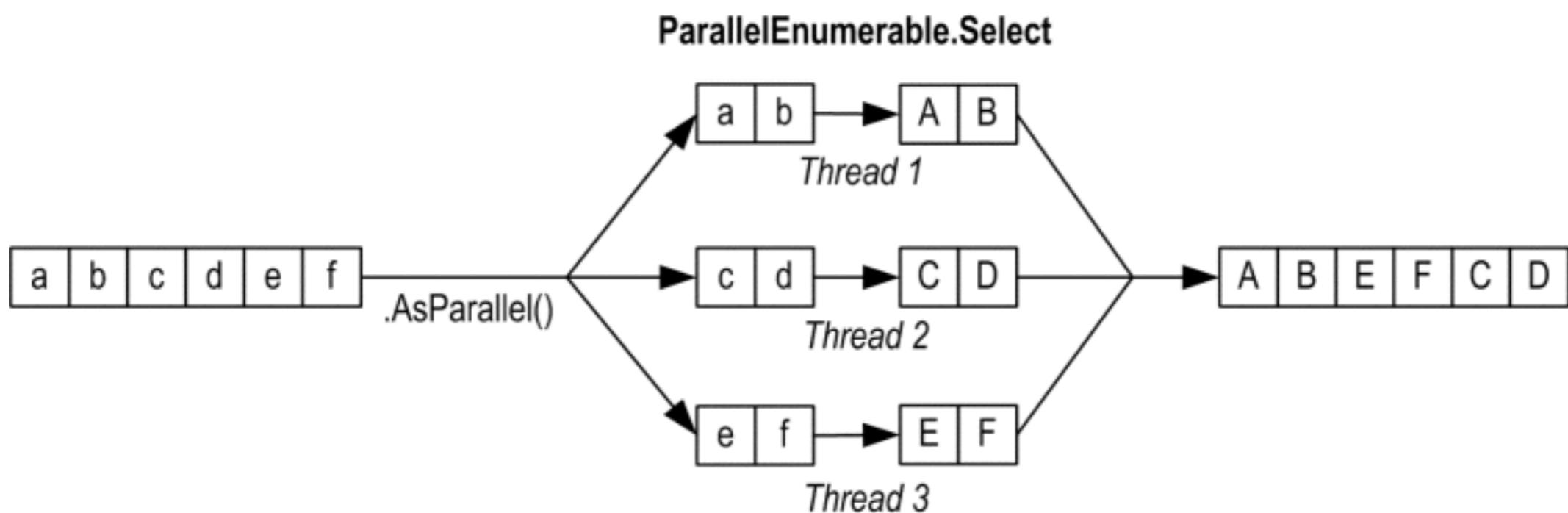
```
foreach(var item in items)  
{  
    Task.Factory.StartNew(() => do(item));  
}
```

# System.Threading



## PLinq

```
var q = from p in people.AsParallel()
        where p.Name == queryInfo.Name &&
              p.State == queryInfo.State &&
              p.Year >= yearStart &&
              p.Year <= yearEnd
        orderby p.Year ascending
        select p;
```



```
"abcdef".AsParallel().Select (c => char.ToUpper(c)).ToArray()
```

# Partitioning Algorithms in PLINQ

- Several partitioning schemes built-in

- Chunk

- Works with any `IEnumerable<T>`
    - Single enumerator shared; chunks handed out on-demand



- Range

- Works only with `IList<T>`
    - Input divided into contiguous regions, one per partition



- Stripe

- Works only with `IList<T>`
    - Elements handed out round-robin to each partition



- Hash

- Works with any `IEnumerable<T>`
    - Elements assigned to partition based on hash code

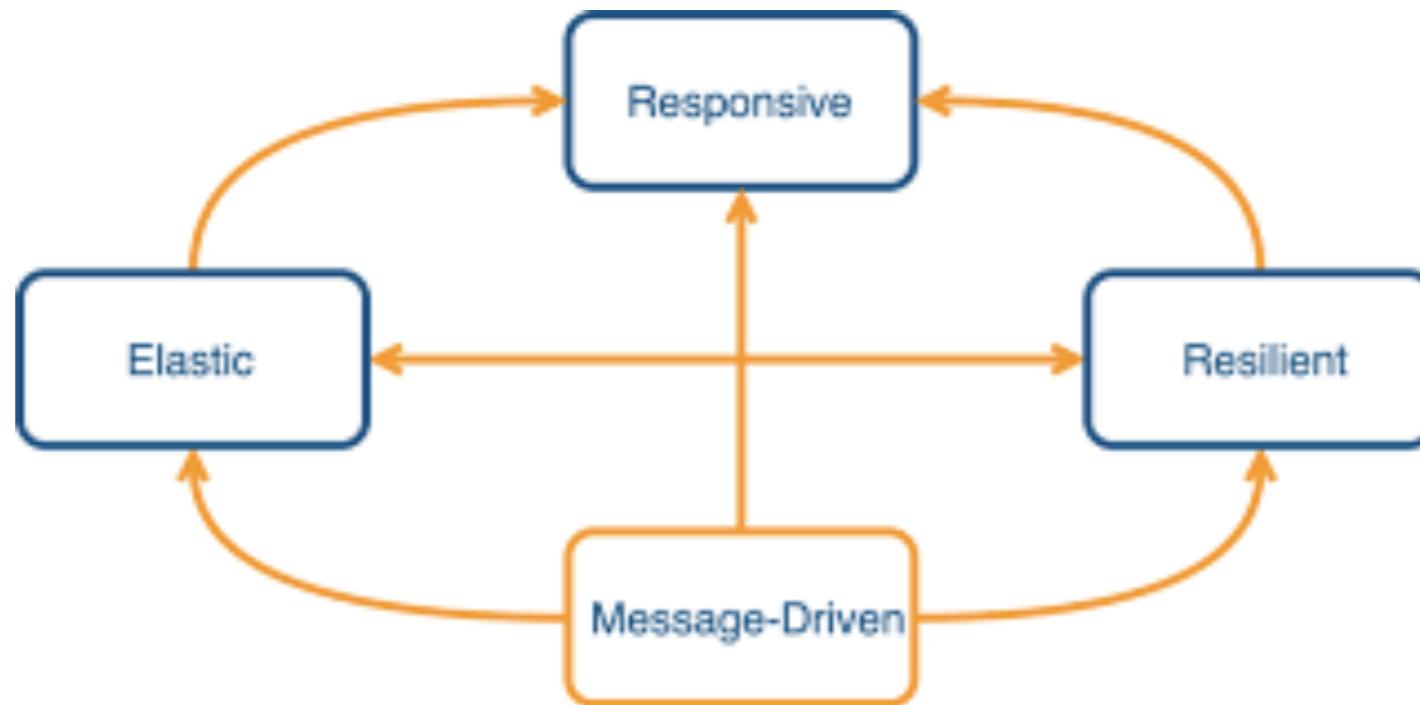


- Custom partitioning available through `Partitioner<T>`

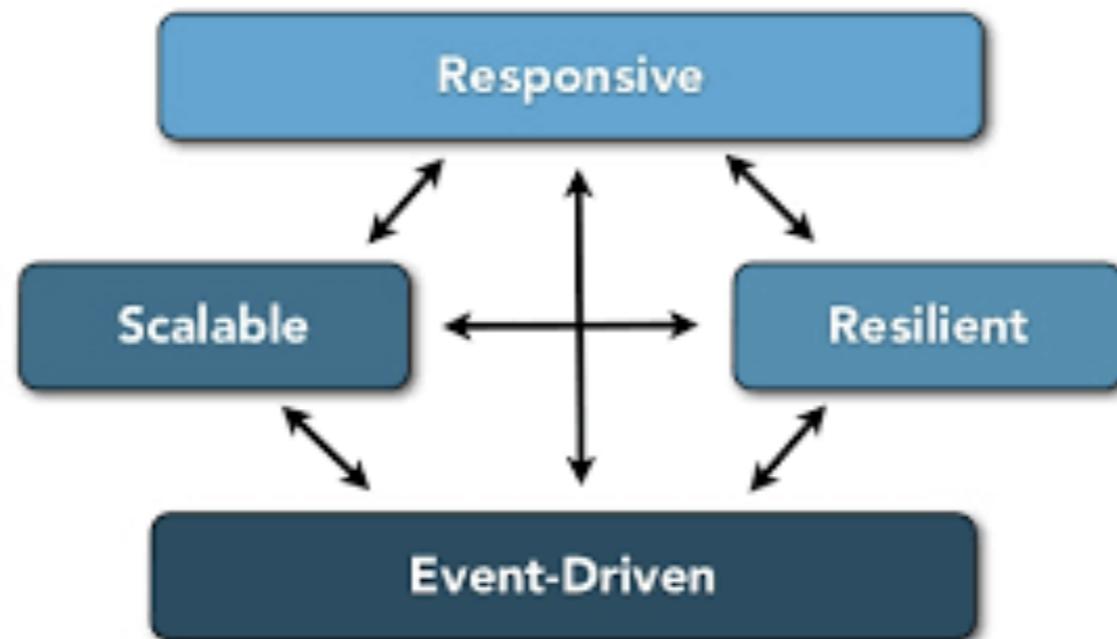
- `Partitioner.Create` available for tighter control over built-in partitioning schemes

**Reactive Programming != Reactive System**

# The Reactive Manifesto



The Reactive Manifesto (<http://www.reactivemanifesto.org/>) is a document defining the four reactive principles

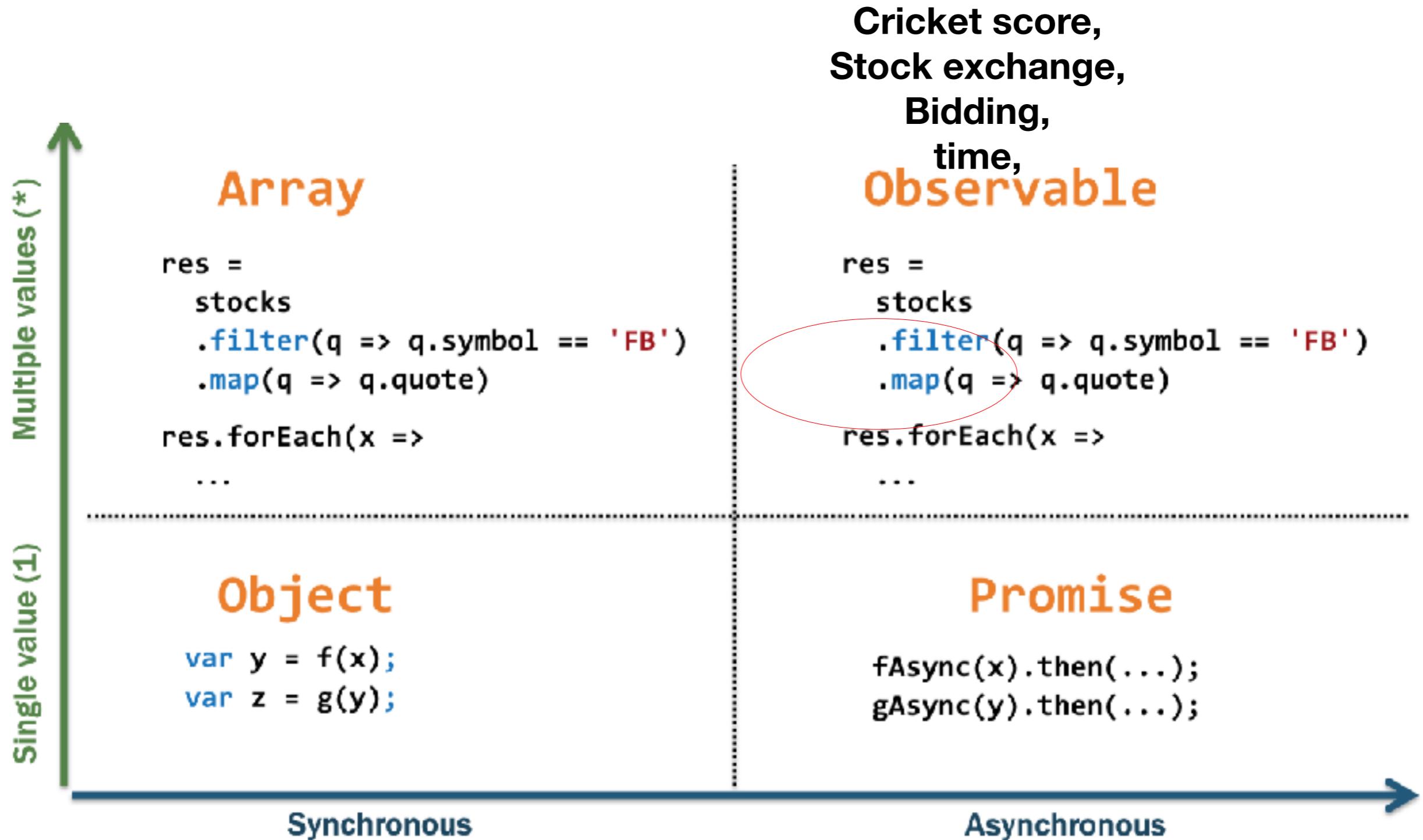


# reactive manifesto

Using **reactive programming does not build a reactive system**. Reactive systems, as defined in the [reactive manifesto](#), are an architectural style to *build responsive distributed systems*. Reactive Systems could be seen as distributed systems done right. A reactive system is characterized by four properties:

- **Responsive**: a reactive system needs to handle requests in a reasonable time (I let you define reasonable).
- **Resilient**: a reactive system must stay responsive in the face of failures (crash, timeout, 500 errors... ), so it must be designed for failures and deal with them appropriately.
- **Elastic**: a reactive system must stay responsive under various loads. Consequently, it must scale up and down, and be able to handle the load with minimal resources.
- **Message driven**: components from a reactive system interacts using asynchronous message passing.

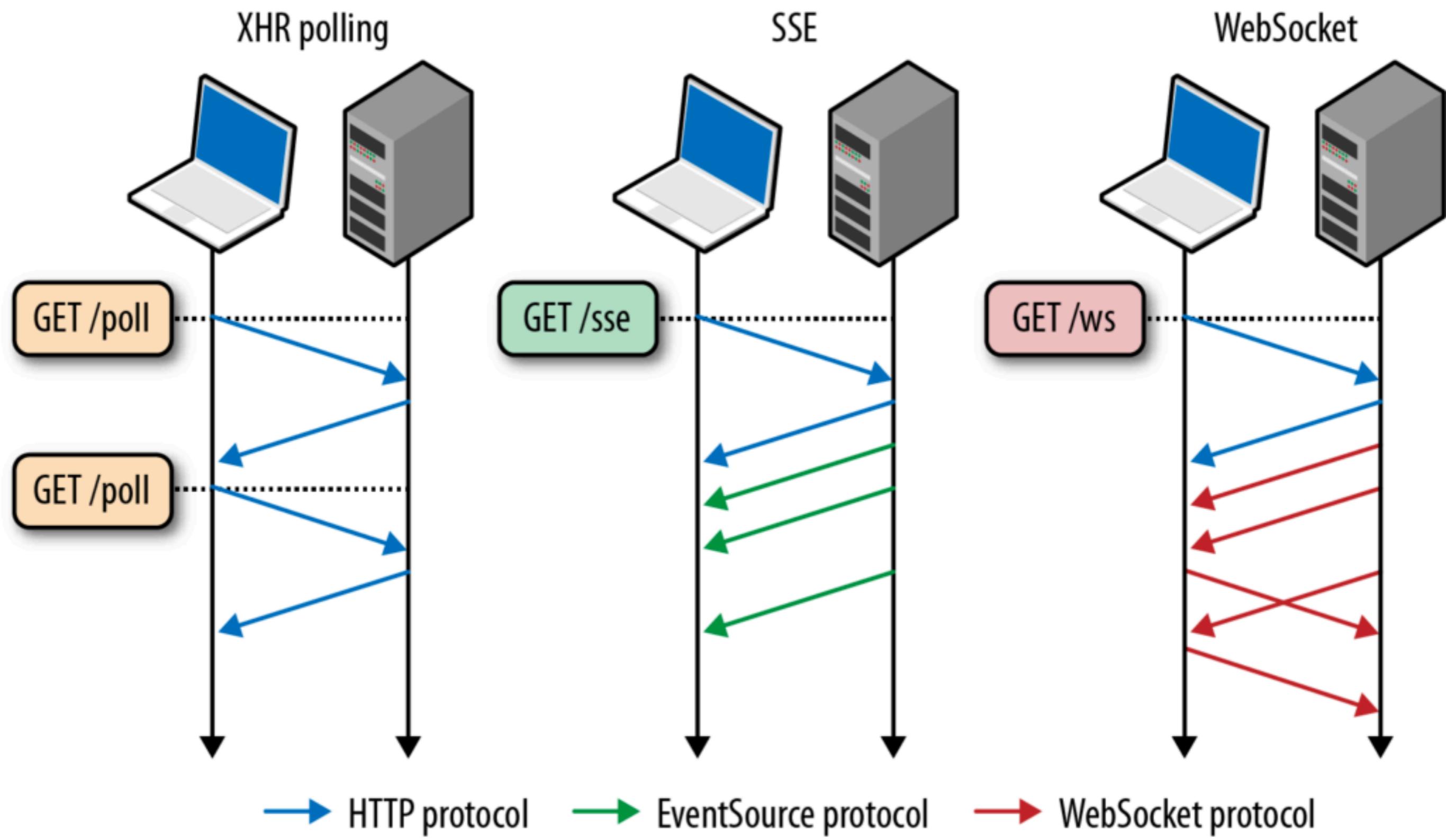
Despite the simplicity of these fundamental principles of reactive systems, building one of them is tricky. Typically, each node needs to embrace an asynchronous non-blocking development model, a task-based concurrency model and uses non-blocking I/O.

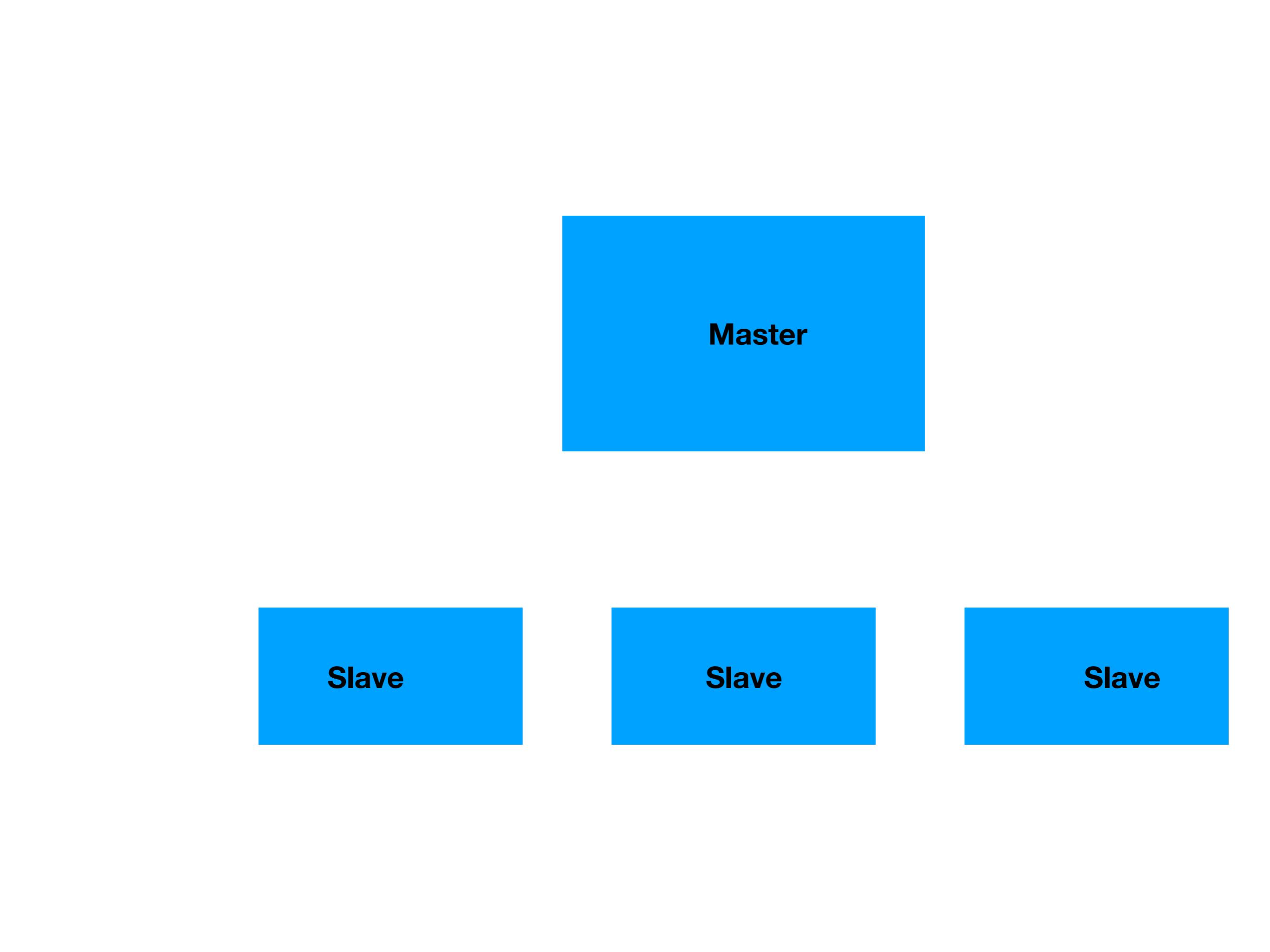


- Asynchronous data streams
- **Everything** is a stream

|

|





**Master**

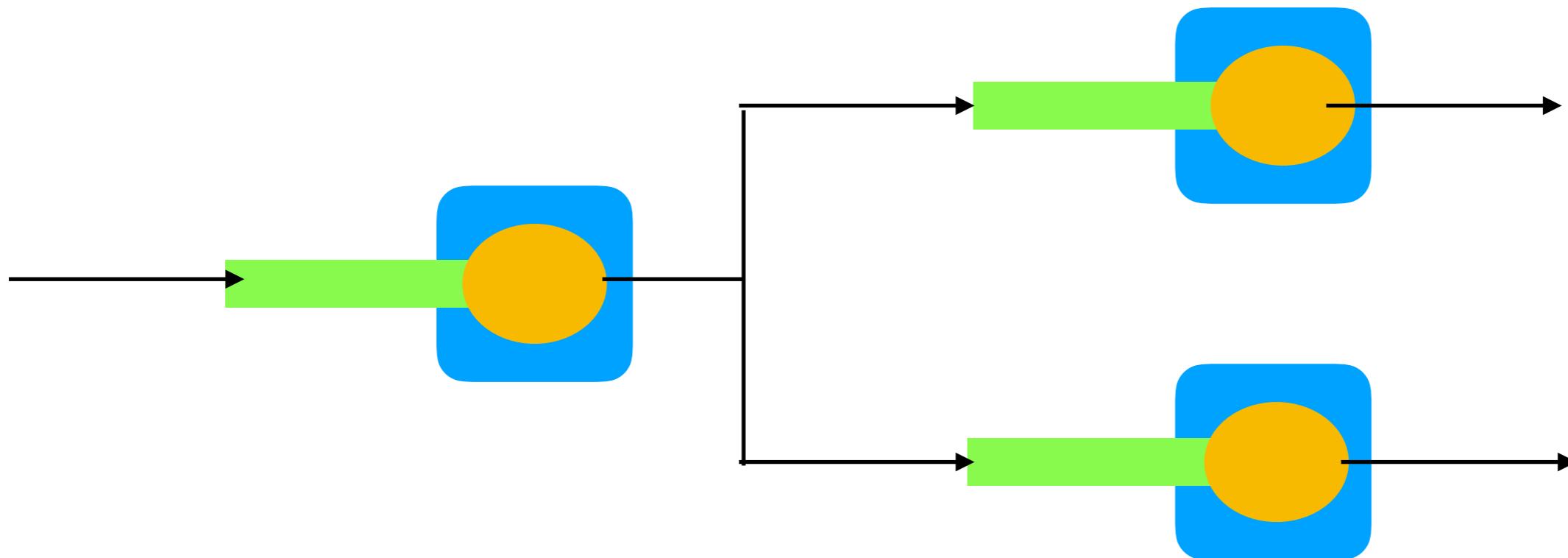
**Slave**

**Slave**

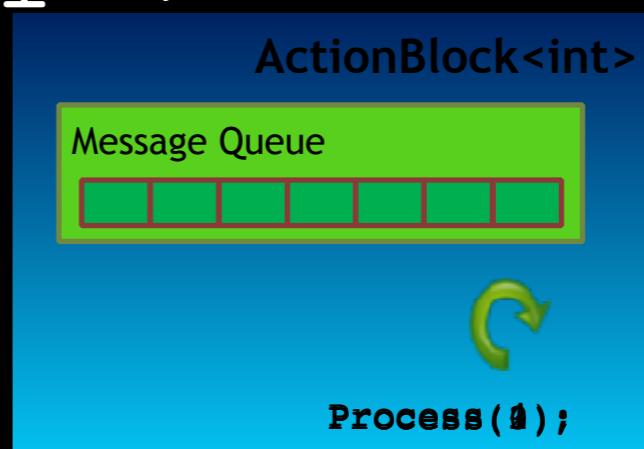
**Slave**

**dict[Key,value]**

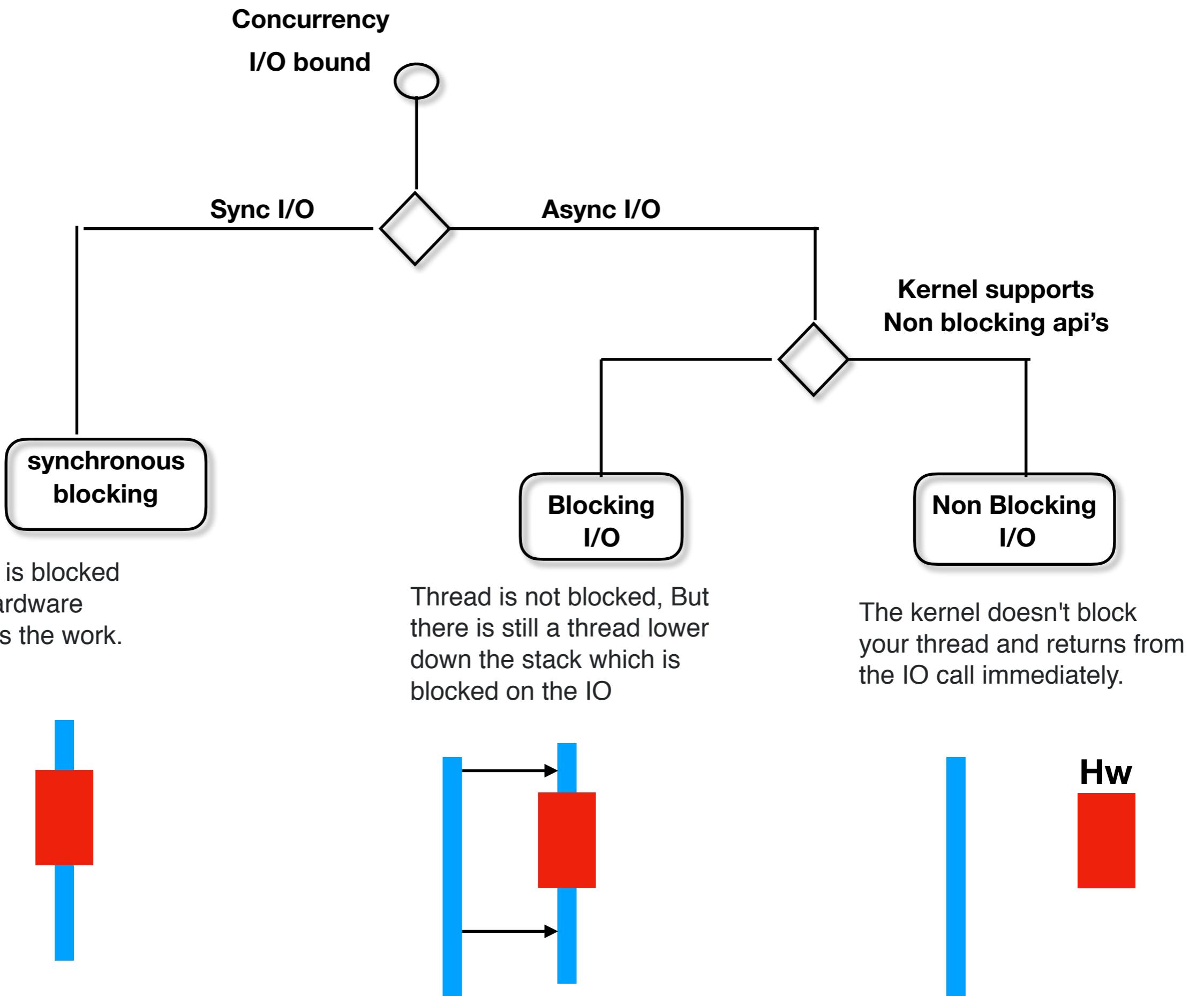
# Actor Pattern



```
var c = new ActionBlock<int>(i =>
{
    Process(i);
}) ;
```



```
for(int i = 0; i < 5; i++)
{
    c.Post(i);
}
```



# How Windows does Synchronous I/O

.NET

```
FileStream fs = new FileStream(...);  
Int32 bytesRead = fs.Read(...);
```

Win32  
User-Mode

```
ReadFile(...);
```

Windows  
Kernel-Mode

```
(Windows I/O Subsystem Dispatcher code)
```

①

⑨

③

⑧

④

⑦

Your thread blocks here!  
Hardware does I/O;  
No threads involved!

⑥

NTFS Driver

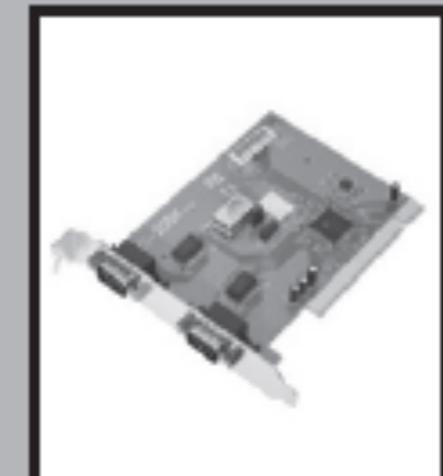
Network



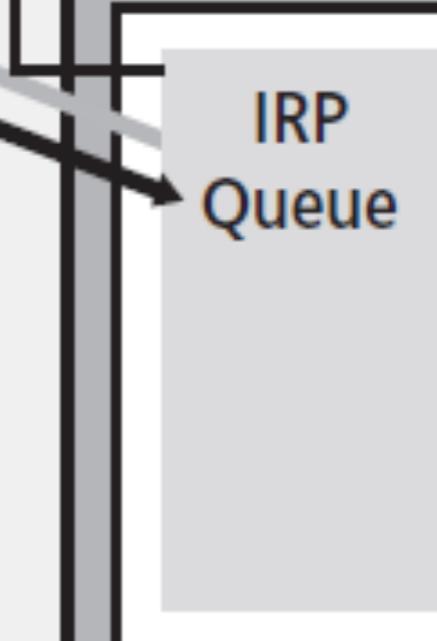
DVD-ROM



RS-232



IRP  
Queue



⑤

## How Windows does Asynchronous I/O

.NET

```
FileStream fs = new FileStream(..., FileMode.Open);
fs.BeginRead(..., AsyncCallback, ...);
```

①

⑦

```
void AsyncCallback(...) { ... }
```

Win32  
User-Mode

```
ReadFile(...);
```

③

IRP

⑥

Windows  
Kernel-  
Mode

```
(Windows I/O Subsystem Dispatcher code)
```

④

⑤

Your thread doesn't  
block here; it  
keeps running! ⑤

The CLR's Thread Pool

Threads can extract  
completed IRP's  
from here

NTFS Driver

IRP  
Queue



⑧

b

c

# Synchronization

No Lock  
Update

Nonblocking  
synchronization  
constructs

- volatile
- Atomic.

Wait

Simple blocking  
methods

- Sleep (bad)
- Join
- Spin

Resource  
Lock

Locking  
constructs

- *Exclusive* locking
- nonexclusive locking
  - Semaphore
  - reader/writer locks.

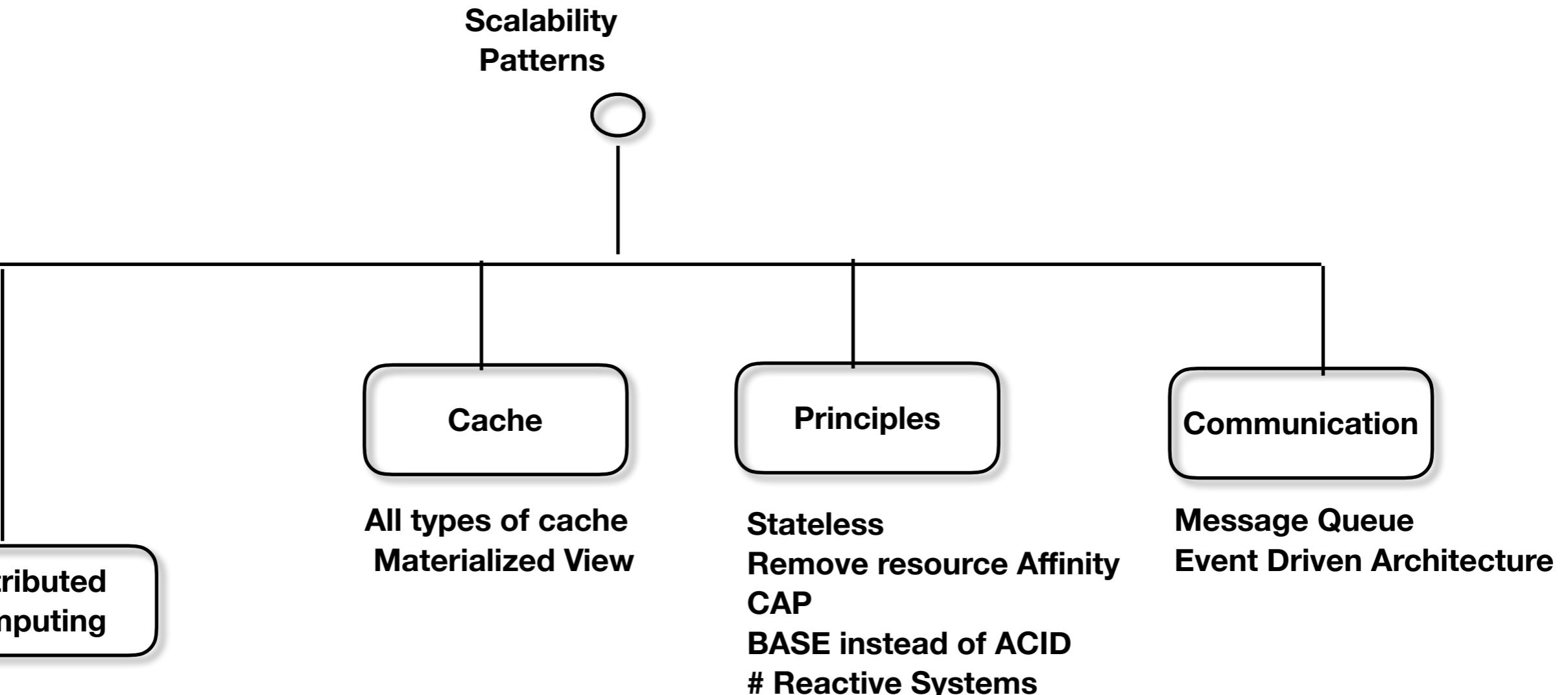
Notify

Signaling  
constructs

- event wait
- Latch
- Barrier



- Sleep (bad)
- Suspend
- Abort
- SetPriority
- Resume



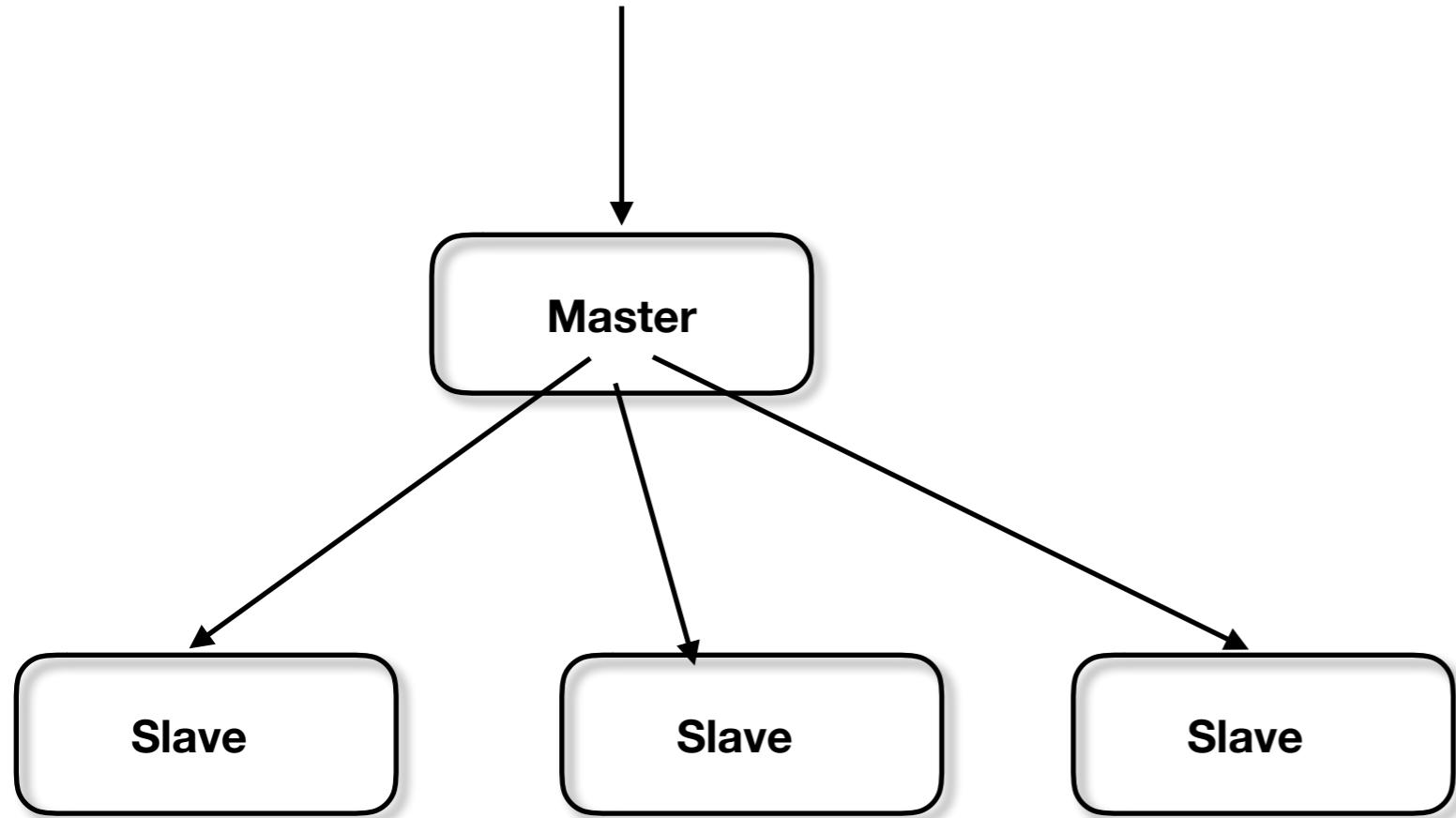
**Scale Up (bigger box)**

**CQRS**

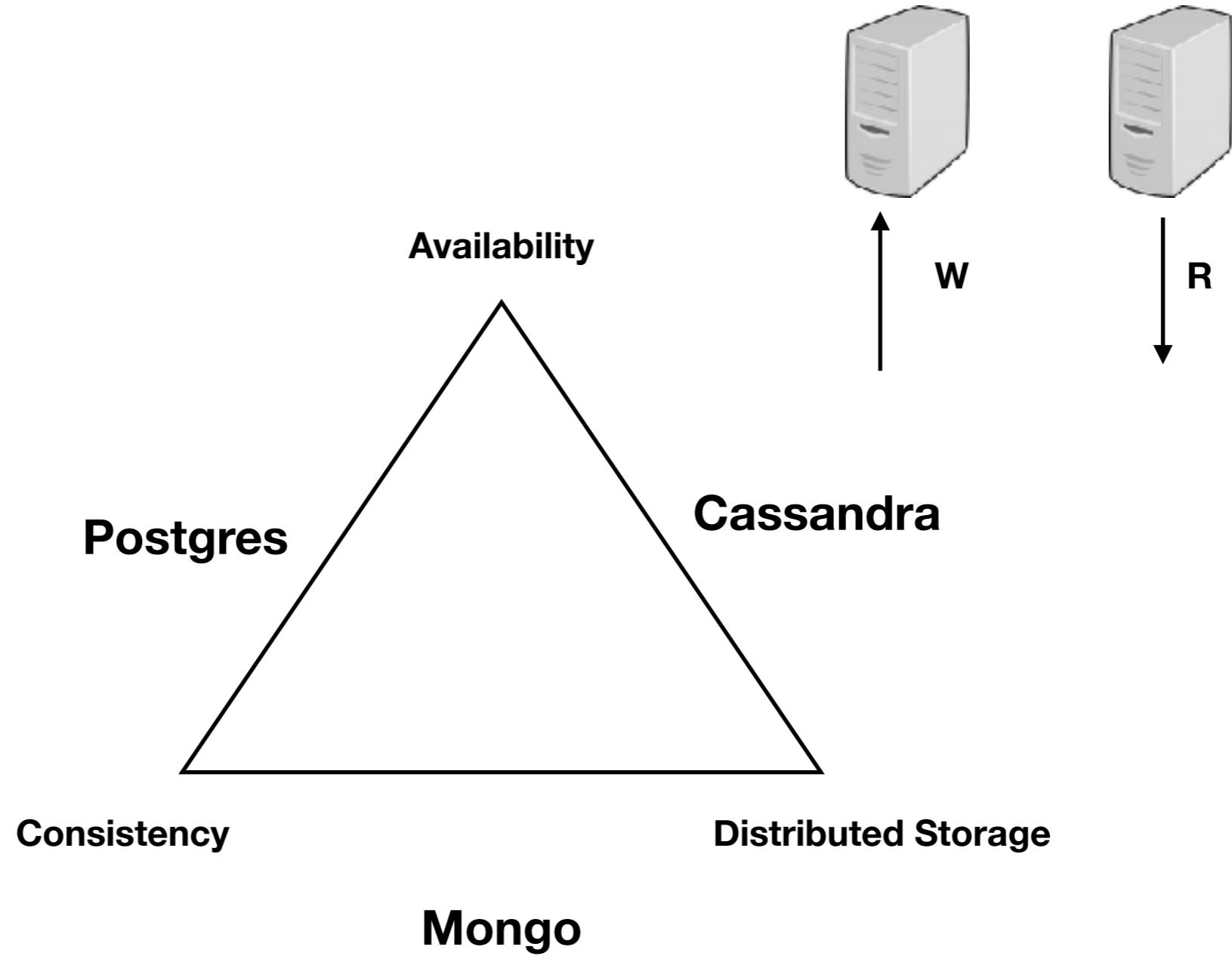
**# Master Slave/ Actor pattern/ Map Reduce/ MPP**

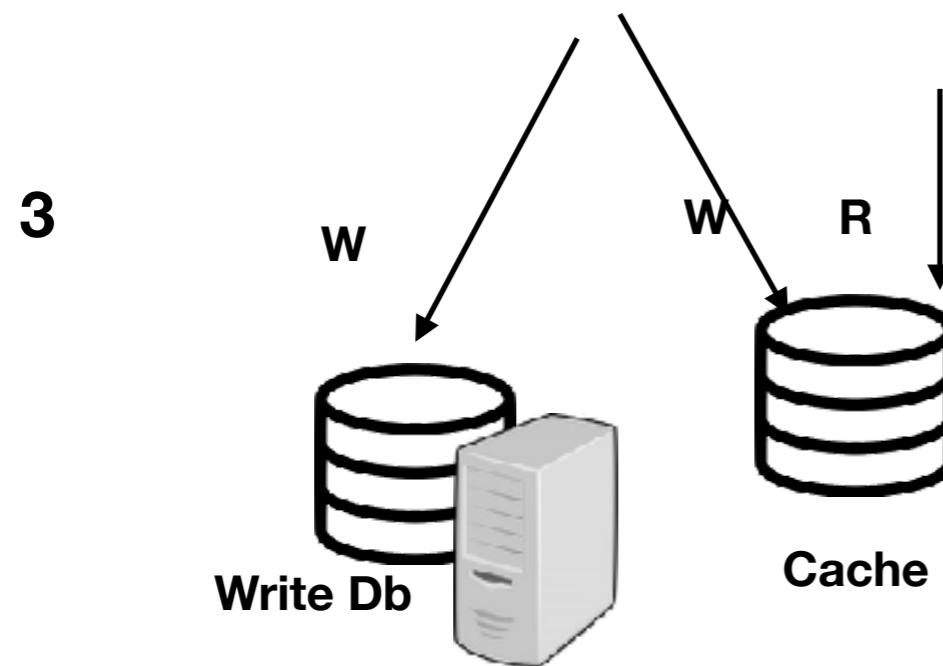
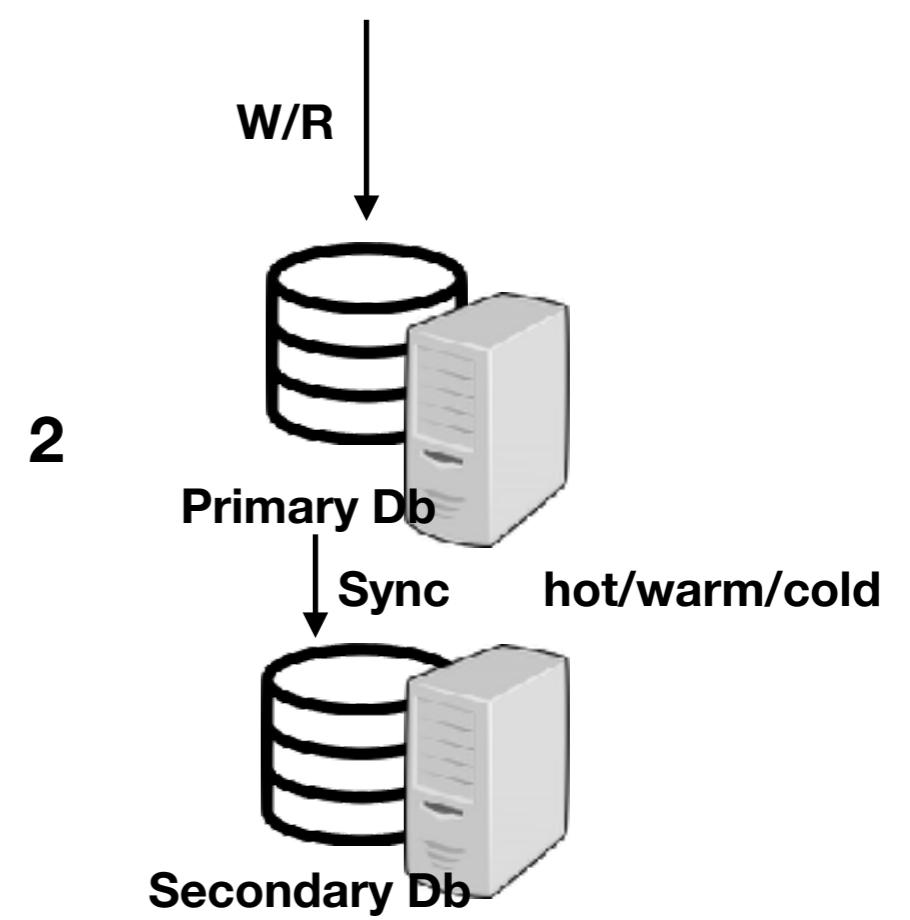
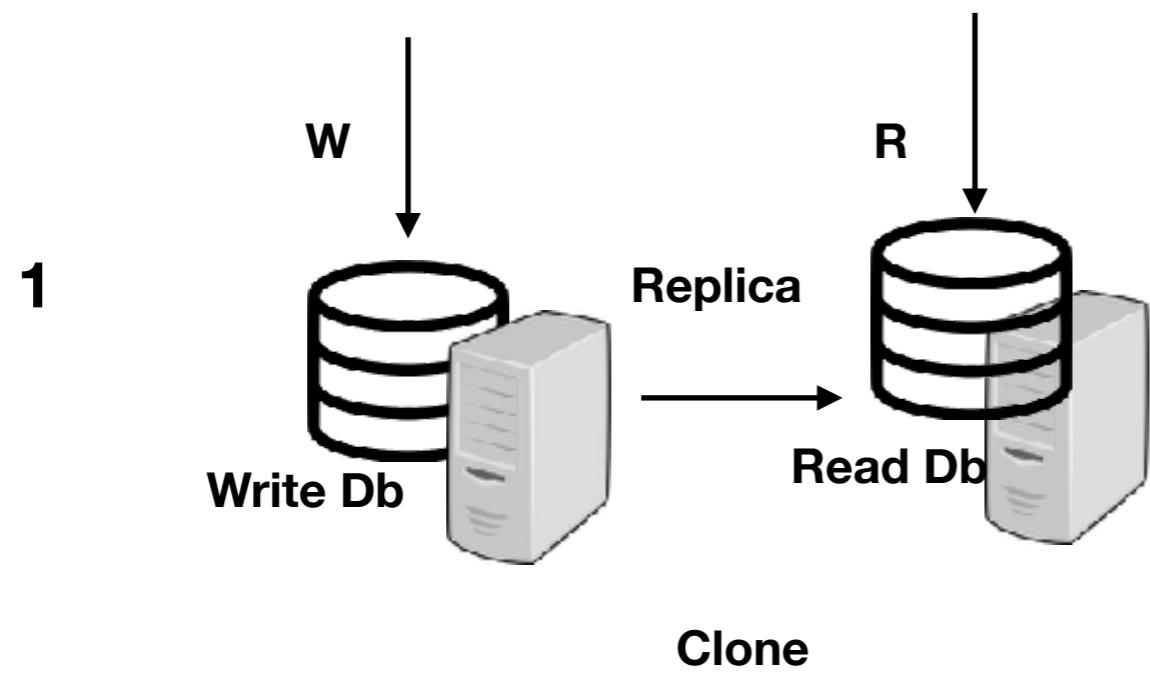
**Scalability Cube**

- Cloning
- Splitting y axis (vertical) column
- Sharding z axis (horizontal) rows



**Actor pattern (Akka)**







**Follower**



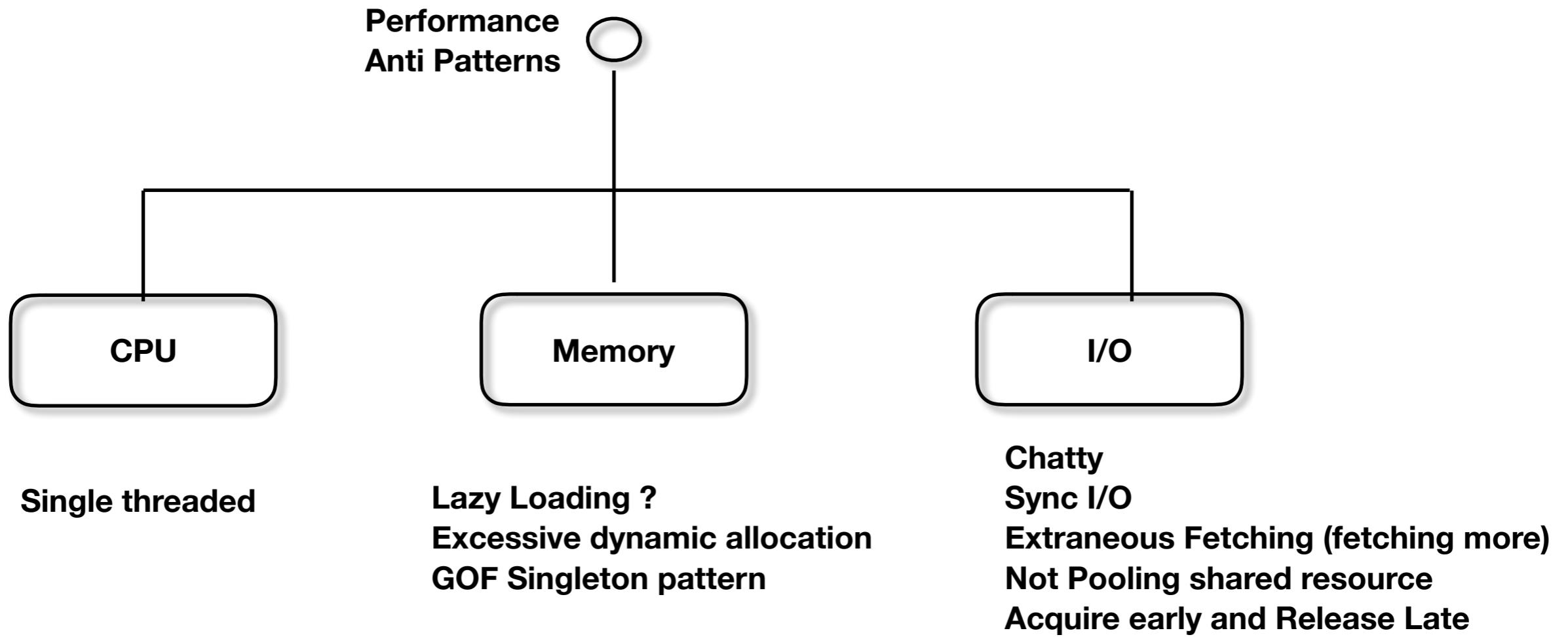
**Leader**



**Follower**



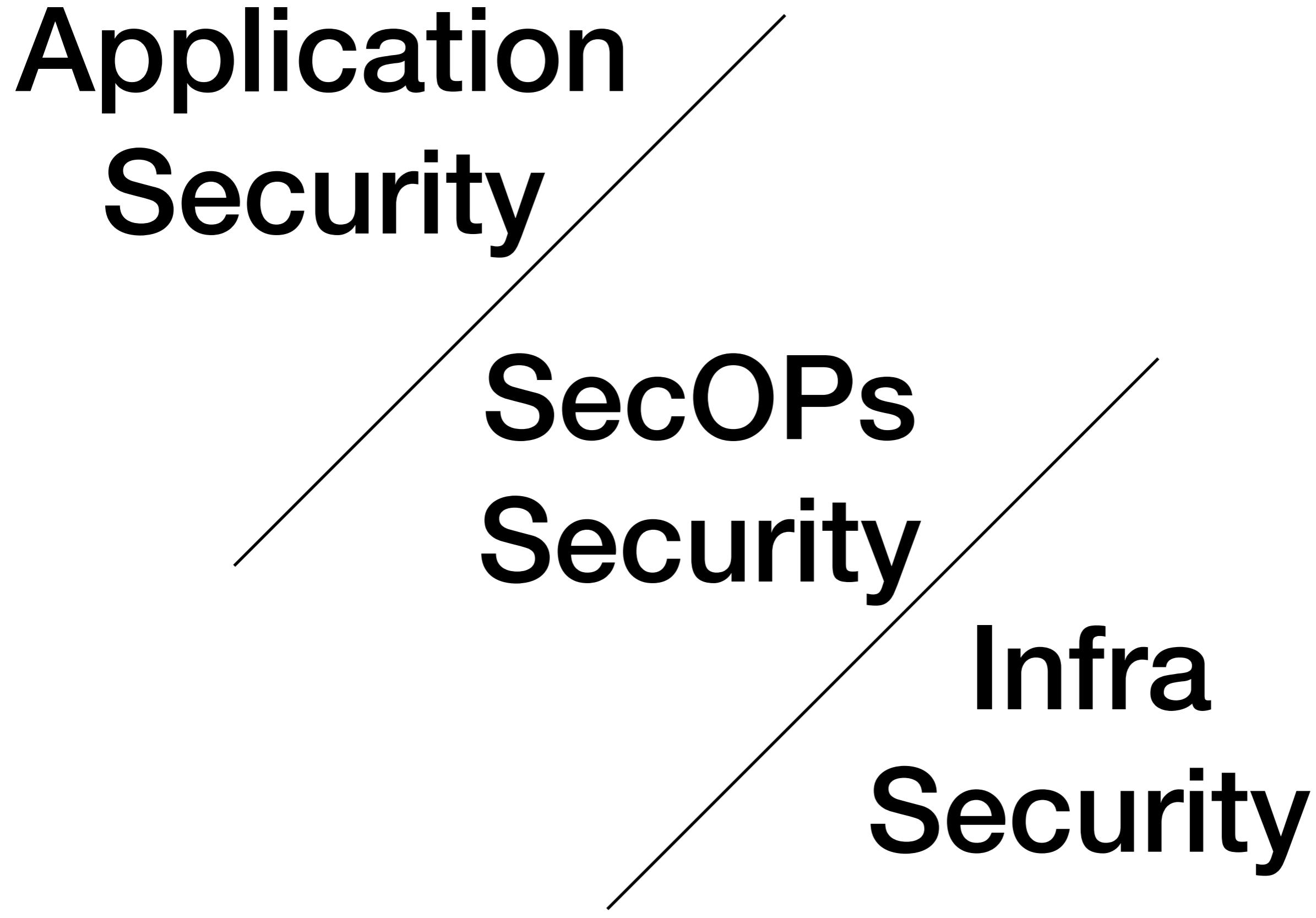
**Follower**



# Application Security View



# **Application Security**



## **SecOPS Security**

## **Infra Security**

## **AAA pattern**

- Authentication
- Authorization
- Audit

- Threat Model (STRIDE)
- S -> spoofing (server, user) {Authentication}
- T -> tampering (wire, rest) {Asset Handling - transit, rest}
- R -> repudation {Audit}
- I -> Information disclosure {Authorization, Asset Handling, exception Handling, session handling }
- D -> denial of service { input validation, throttling }
- E -> elevation of privilege { Authorization }

# **AAA**

**Authentication - who are you ?**

**Authorization - what can you do ?**

**Audit - what did you do ?**

Authentication (who are you)	By what you know : OAuth 2 + OPenIDConnect By What you have By Where you are	<b>&lt;&lt;centralize&gt;&gt;</b>
Authorization (who can do what)	Role Based Claim Based	<b>&lt;&lt;centralize&gt;&gt;</b>
Audit (who did , what)	Event Sourcing	<b>&lt;&lt;centralize&gt;&gt;</b>
Input validation	WAF + framework + code	
Asset Handling In rest (storage)	Code (Hash, Sym Encryption, Asym Encryption) Cloud storage encryption	
Asset Handling In Transit (wire)	TLS 1.3	
Exception Handling (Code)	Framework (Spring)	
Session Handling (code)	Framework (Spring)	
Key Management		
Throttling Limits		

**# Something you know - pwd(oauth2) , security question <– 1**  
**# Something you Have - otp, cert, rsa, email, <– 2**  
**# Where you are - country, office, time, <– 3**  
**# Something you are - voice, face, dna, finger print, ..**  
**# Something you do - behaviour**

SOMETHING YOU  
KNOW  
(PASSWORD)

SOMETHING YOU  
HAVE  
(OTP TOKEN)

2FA

2FA

SOMETHING YOU  
ARE  
(FINGERPRINT)

MFA

- User name , pwd <—
  - Email <—
  - SMS <—
  - Secret question <—
  - RSA token <—
  - Certificates
  - Bio metric (face, voice, finger print, dna, ...) <—
  - Location
  - Time
- By what you know \*
  - By what you have \*
  - By what you are
  - By where you are \*
  - By your behaviour

Authentication (who are you)

Externalize

Authorization (who can do what)

Externalize

Audit (who did , what)

Externalize

Input validation

50% Externalize + 50% in service

Asset Handling In rest (storage)

In service

Asset Handling In Transit (wire)

Externalize

Exception Handling (Code)

App Framework

Session Handling (code)

App Framework

Key Management

Externalize

Authentication (who are you)

Oauth2, OpenID Connect, STS

Authorization (who can do what)

RBAC, CBAC

Audit (who did , what)

Event Sourcing

Input validation

WAF + 50% in code

Asset Handling In rest (storage)

HASH, Sym or Aysm encryption,  
DISK encryption

Asset Handling In Transit (wire)

TLS 1.3

Exception Handling (Code)

App Framework

Session Handling (code)

App Framework

Key Management

Key Vault

Authentication (who are you)

OAuth2 + OIDC

Authorization (who can do what)

claim(attribute) based \*

Audit (who did , what)

Event Sourcing,

Input validation (WAF, ...)

WAF, code level

Asset Handling In rest (storage)

NA

Asset Handling In Transit (wire)

SSL / TLS

Exception Handling (Code)

Spring FWK

Session Handling (code)

Spring Fwk

Key Management

Key vault

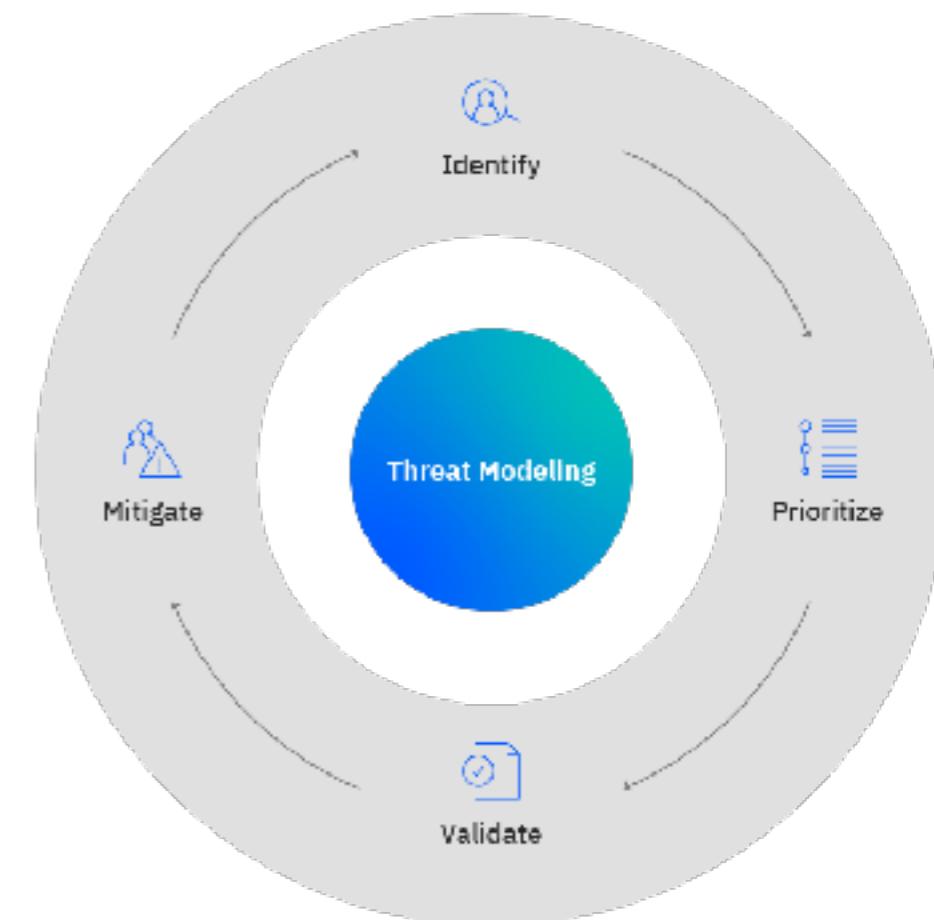
Authentication (who are you)	By What you Know (pwd, pin, dob, secret, By What you Have (otp, RSA , email, certificates, ...) By What you Are (face, retina, finger print, DNA, ...) By Where you Are (location, time, IP ...) By Behaviour (...)
Authorization (who do what)	RBAC, ABAC / CBAC,
Audit (who did , what)	Log, Event sourcing
Input validation (WAF, ...)	FWK , WAF, captcha
Asset Handling In rest	Encryption, Hashing
Asset Handling In Transit	Https , wss, TLS, ...
Exception Handling	Fwk
Session Handling	Fwk
Key Management	Vault

# STRIDE

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

# Threat Modeling

Threat Modeling Method	Features
STRIDE	<ul style="list-style-type: none"> <li>Helps identify relevant mitigating techniques</li> <li>Is the most mature</li> <li>Is easy to use but is time consuming</li> </ul>
PASTA	<ul style="list-style-type: none"> <li>Helps identify relevant mitigating techniques</li> <li>Directly contributes to risk management</li> <li>Encourages collaboration among stakeholders</li> <li>Contains built-in prioritization of threat mitigation</li> <li>Is laborious but has rich documentation</li> </ul>
LINDDUN	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Contains built-in prioritization of threat mitigation</li> <li>Can be labor intensive and time consuming</li> </ul>
CVSS	<ul style="list-style-type: none"> <li>Contains built-in prioritization of threat mitigation</li> <li>Has consistent results when repeated</li> <li>Has automated components</li> <li>Has score calculations that are not transparent</li> </ul>
Attack Trees	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Has consistent results when repeated</li> <li>Is easy to use if you already have a thorough understanding of the system</li> </ul>
Persona non Grata	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Directly contributes to risk management</li> <li>Has consistent results when repeated</li> <li>Tends to detect only some subsets of threats</li> </ul>
Security Cards	<ul style="list-style-type: none"> <li>Encourages collaboration among stakeholders</li> <li>Targets out-of-the-ordinary threats</li> <li>Leads to many false positives</li> </ul>
hTMM	<ul style="list-style-type: none"> <li>Contains built-in prioritization of threat mitigation</li> <li>Encourages collaboration among stakeholders</li> <li>Has consistent results when repeated</li> </ul>
Quantitative TMM	<ul style="list-style-type: none"> <li>Contains built-in prioritization of threat mitigation</li> <li>Has automated components</li> <li>Has consistent results when repeated</li> </ul>
Trike	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Directly contributes to risk management</li> <li>Contains built-in prioritization of threat mitigation</li> <li>Encourages collaboration among stakeholders</li> <li>Has automated components</li> <li>Has vague, insufficient documentation</li> </ul>
VAST Modeling	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Directly contributes to risk management</li> <li>Contains built-in prioritization of threat mitigation</li> <li>Encourages collaboration among stakeholders</li> <li>Has consistent results when repeated</li> <li>Has automated components</li> <li>Is explicitly designed to be scalable</li> <li>Has little publicly available documentation</li> </ul>
OCTAVE	<ul style="list-style-type: none"> <li>Helps identify relevant mitigation techniques</li> <li>Directly contributes to risk management</li> <li>Contains built-in prioritization of threat mitigation</li> <li>Encourages collaboration among stakeholders</li> <li>Has consistent results when repeated</li> <li>Is explicitly designed to be scalable</li> <li>Is time consuming and has vague documentation</li> </ul>



# PASTA

## STAGE I - Definition of the Objectives (DO)

- DO 1.1 - Document the business requirements
- DO 1.2 – Define the security/compliance requirements
- DO 1.3 – Define the business impact
- DO 1.4 – Determine the risk profile

## Stage II - Definition of the Technical Scope (DTS)

- DTS 2.1 – Enumerate Software components
- DTS 2.2 – Identify Actors & Data Sinks/Source
- DTS 2.3 – Enumerate System-Level services
- DTS 2.4 – Enumerate 3rd Party infrastructure.
- DTS 2.5 – Assert completeness of secure design.

## Stage III - Application Decomposition and Analysis (ADA)

- ADA 3.1 – Enumerate all application use cases
- ADA 3.2 – Document Data Flow Diagrams (DFDs)
- ADA 3.3 – Security functional analysis & the use of trust boundaries

## Stage IV - Threat Analysis (TA)

- TA 4.1 – Analyze the overall threat scenario
- TA 4.2 – Gather threat information from internal threat sources
- TA 4.3 – Gather threat information from External threat sources
- TA 4.4 – Update the threat libraries
- TA 4.5 – Threat agents to assets mapping.
- TA 4.6 – Assignment of the probabilistic values for identified threats

## Stage V - Weakness and Vulnerability Analysis (WVA)

- WVA 5.1 – Review/correlate existing vulnerabilities
- WVA 5.2 – Identify weak design patterns in the architecture
- WVA 5.3 – Map threats to vulnerabilities
- WVA 5.4 – Provide Context risk Analysis based upon Threat-Vulnerability
- WVA 5.5 – Conduct targeted vulnerability testing

## Stage VI - Attack Modeling & Simulation (AMS)

- AMS 6.1 – Analyze the attack scenarios
- AMS 6.2 – Update the attack library/vectors and the control framework
- AMS 6.3 – Identify the attack surface and enumerate the attack vectors
- AMS 6.4 – Assess the probability and impact of each attack scenario.
- AMS 6.5 – Derive a set of cases to test existing countermeasures.
- AMS 6.6 – Conduct attack driven security tests and simulations

## STAGE VII - Risk Analysis & Management (RAM)

- RAM 7.1 – Calculate the risk of each threat
- RAM 7.2 – Identify countermeasures and risk mitigations measures
- RAM 7.3 – Calculate the residual risks
- RAM 7.4 – Recommend strategies to manage risks

- Identify all assets (SSN, PII, Credit card, ...)
- Identify entry point (web server, app server, db server, ...)
- Identify exit point (web server->App sever, App server->db server)
- Identify all data flow (flow1,flow2, flow 3,,,)
- Identify vulnerabilities (input validation, data transit,. ....)
- Identify Threat (xss, sql injection, ...)
- Estimate Risk

# STRIDE

**Spoofing**

**Authentication**

**Elevation of Privilege**

**Session Handling**

**Tampering**

**Input Validation**

**Repudation**

**Audit**

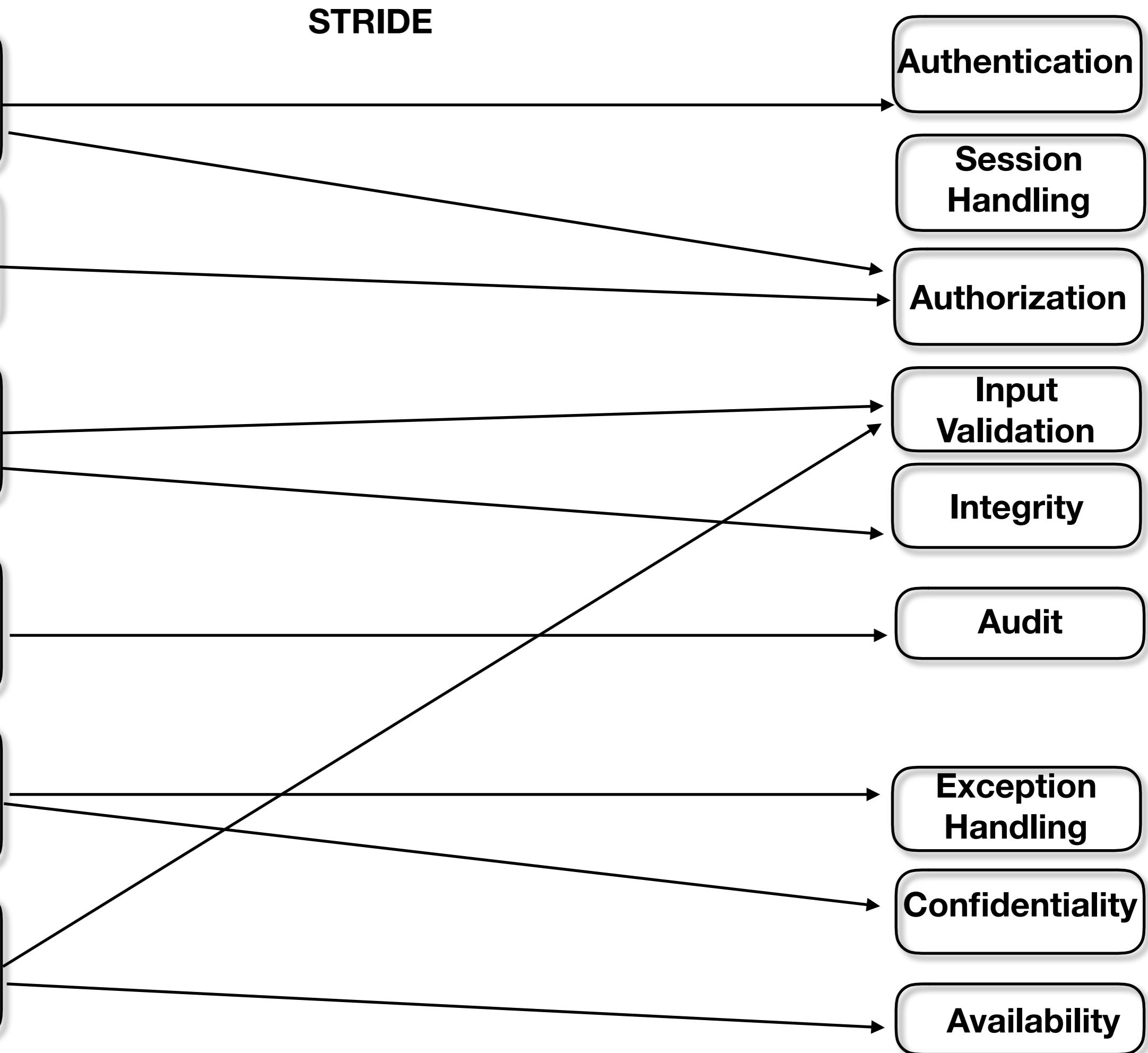
**Information Disclosure**

**Exception Handling**

**Denial of Service**

**Confidentiality**

**Availability**



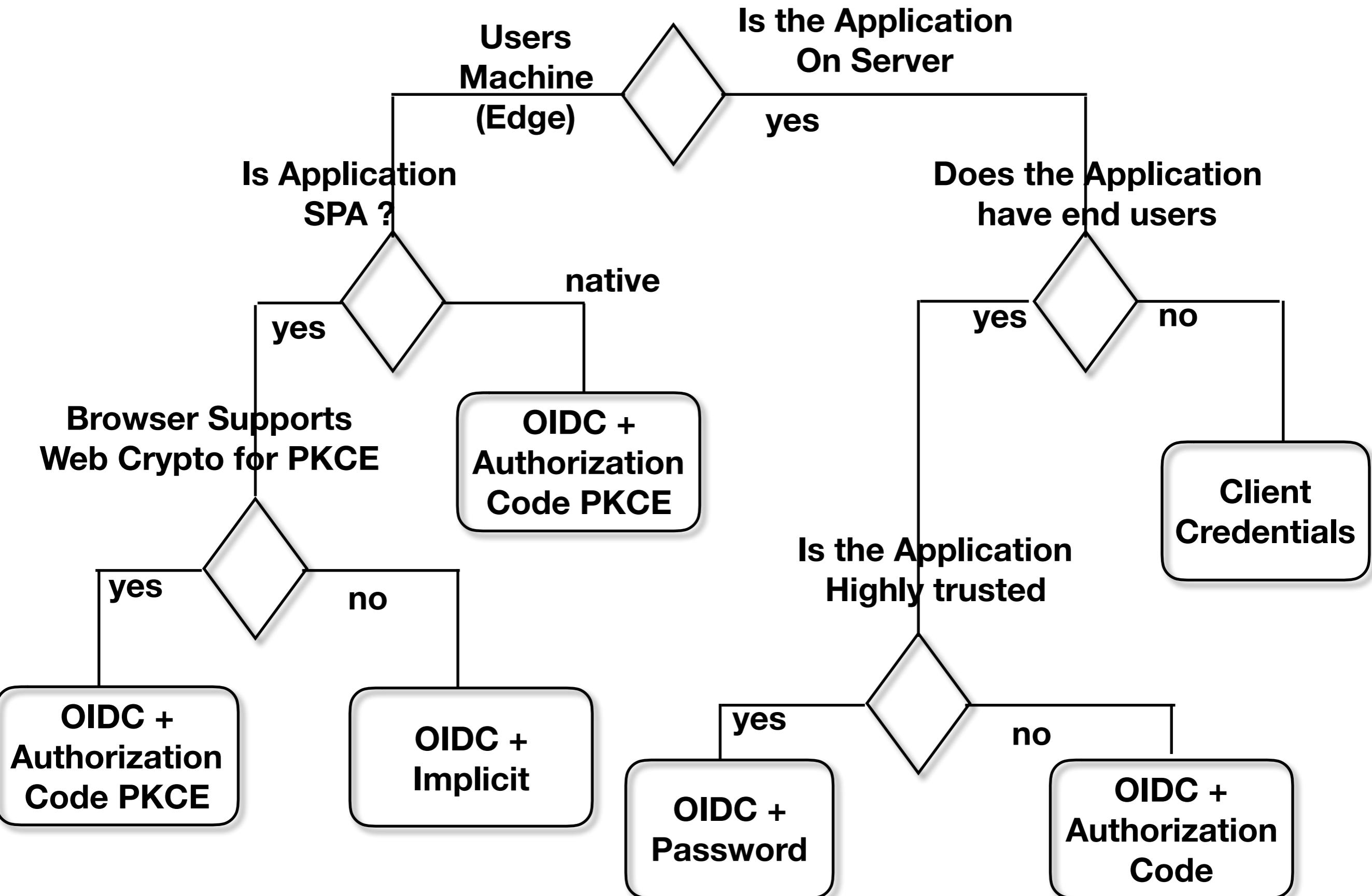
## **App Concerns**

- 3A
  - Authentication (who)
  - Authorization (what)
  - Audit
- Asset
  - Confidentiality (on wire, in rest)
  - Integrity (on wire, in rest)
- Input Validation (70%)
- Exception Handling
- Session Management
- Key Management

## **Infra Concerns**

- Updates/ patches
- Throttling
- vnet/ subnet
- L4 FW
- L7 FW (WAF)
- VPN

# Step 1. Authentication





Customer

Token

UI

Service1

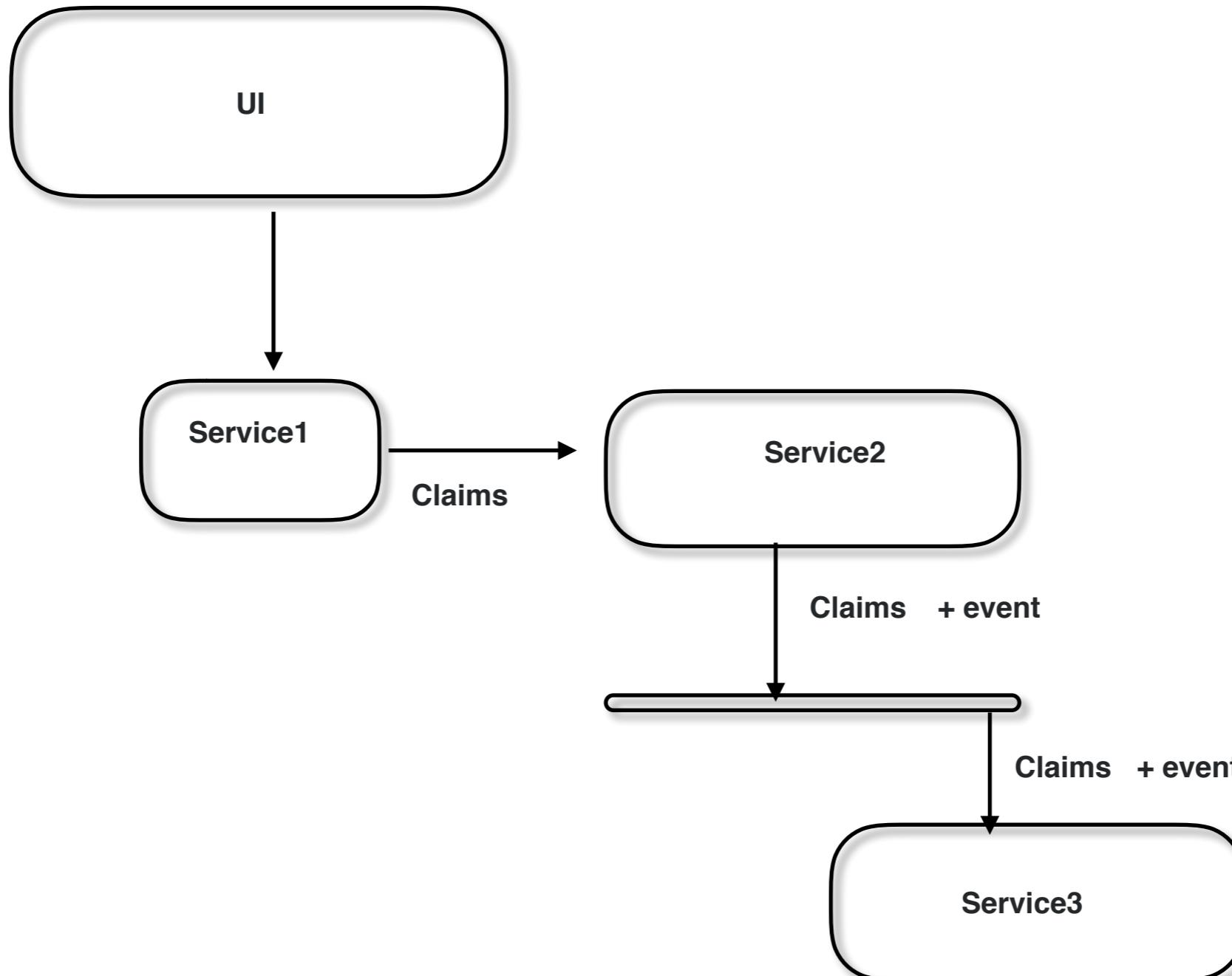
Service2

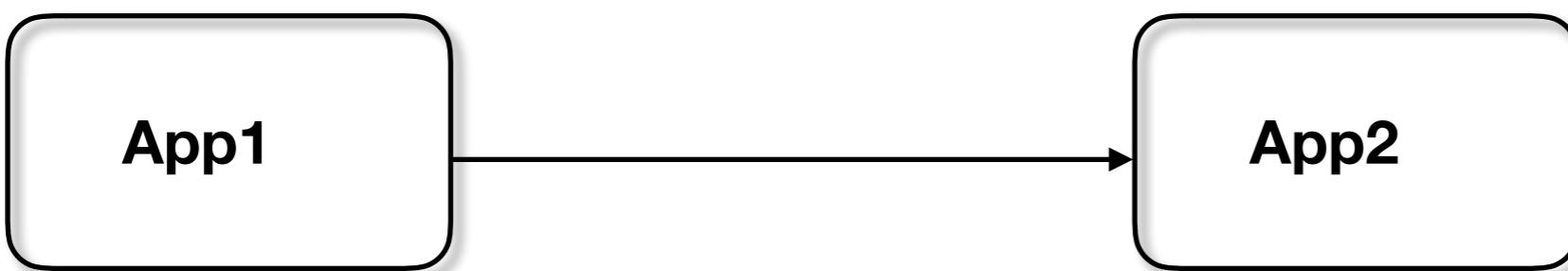
Claims

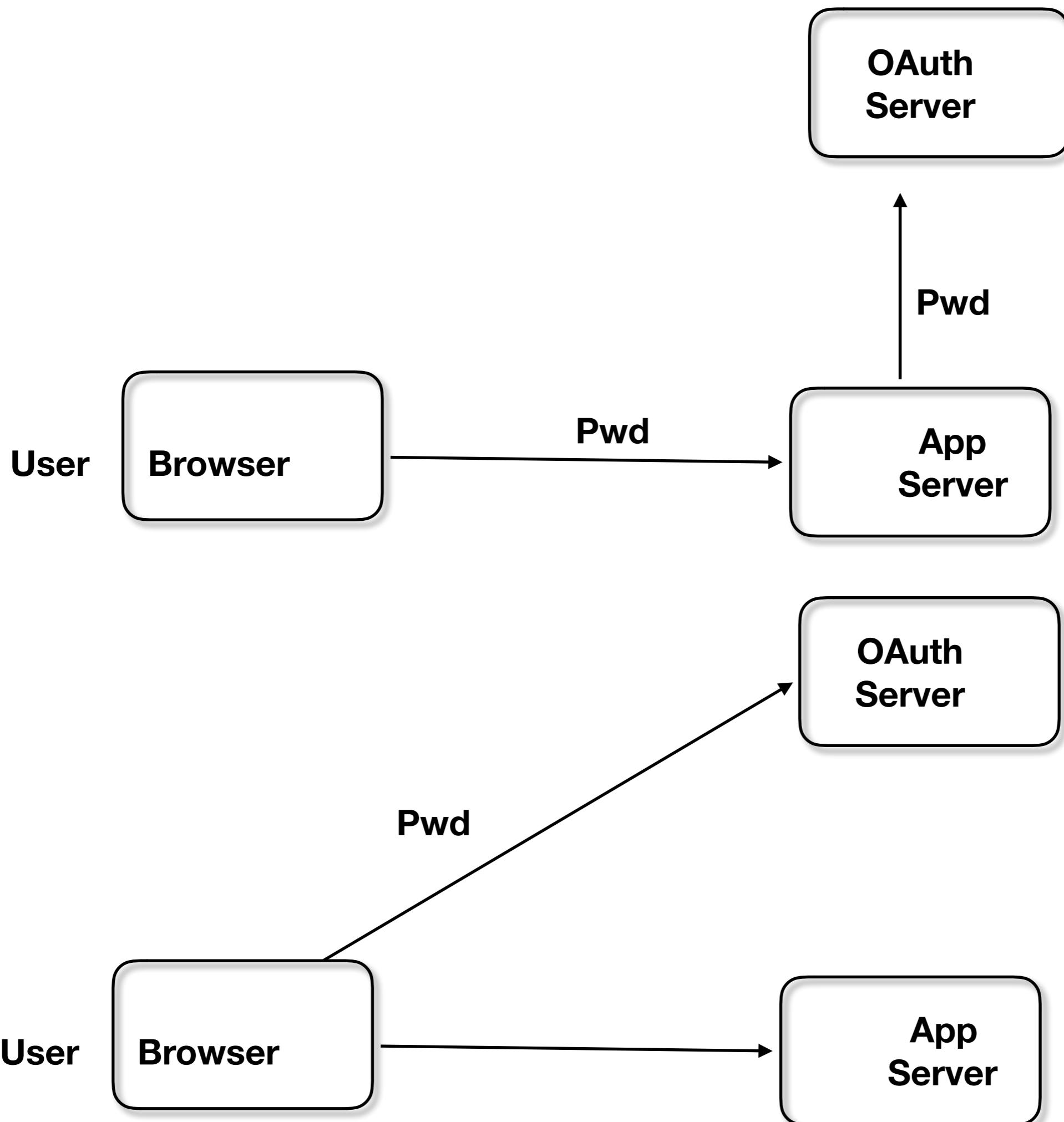
Claims + event

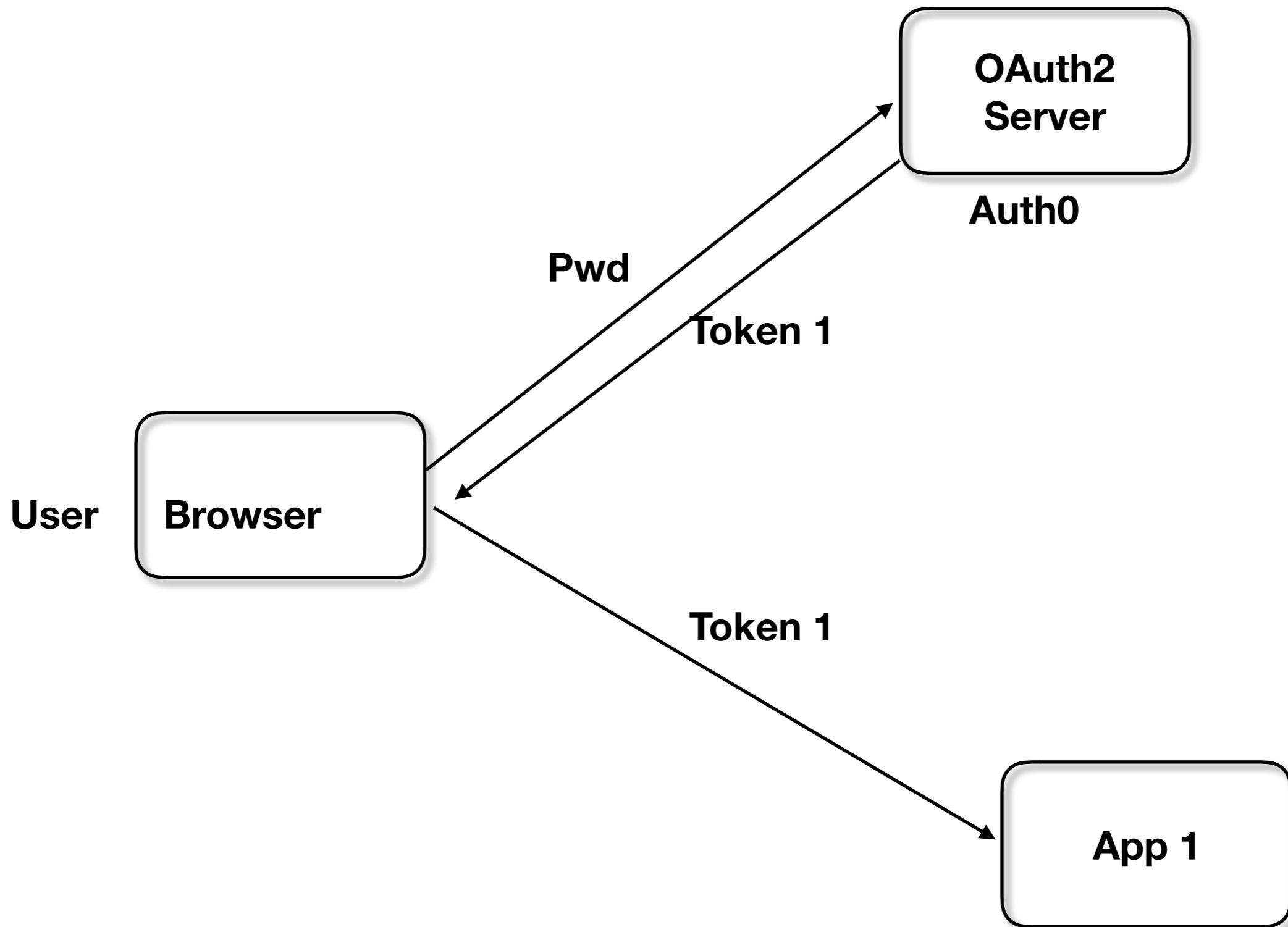
Claims + event

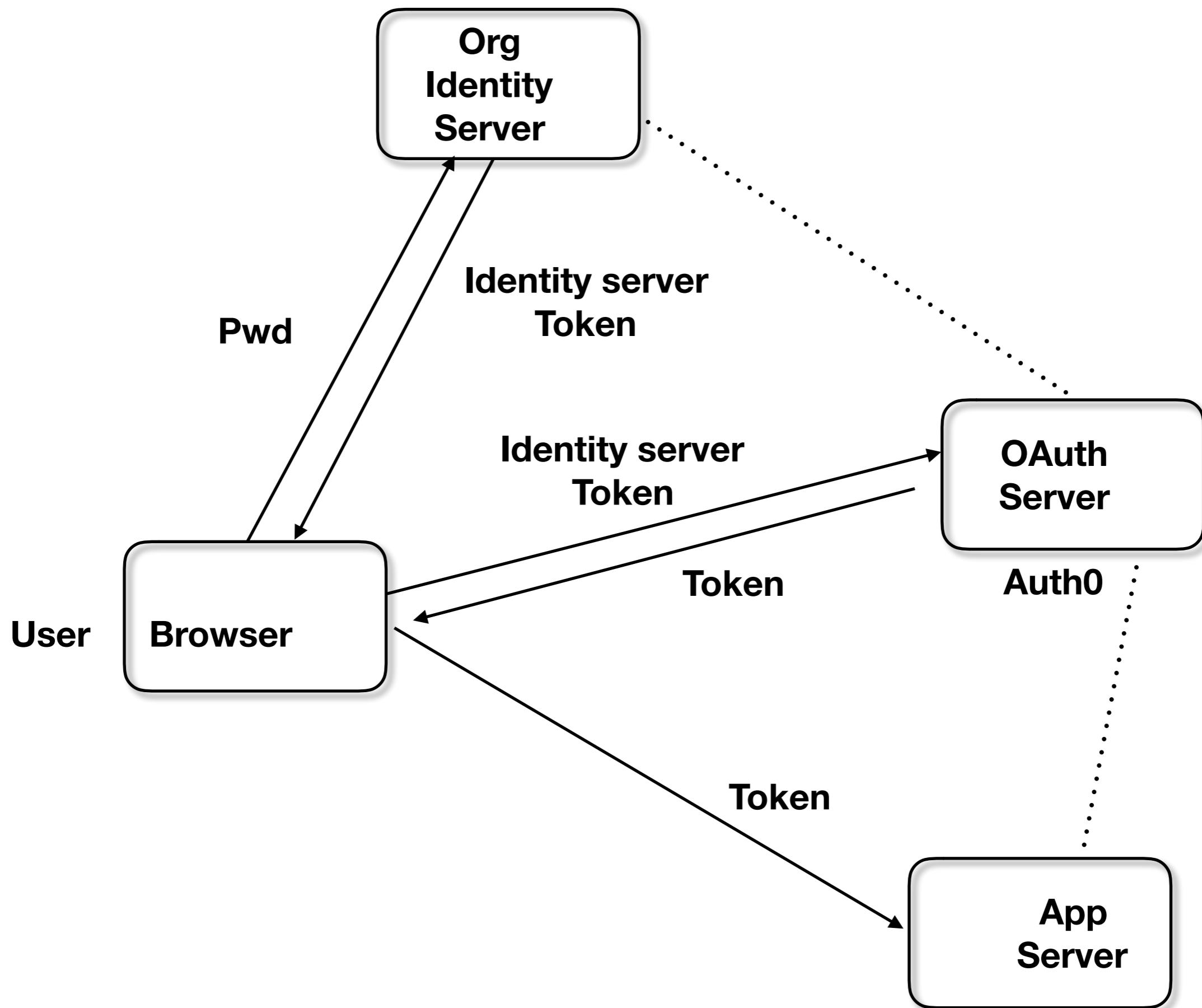
Service3



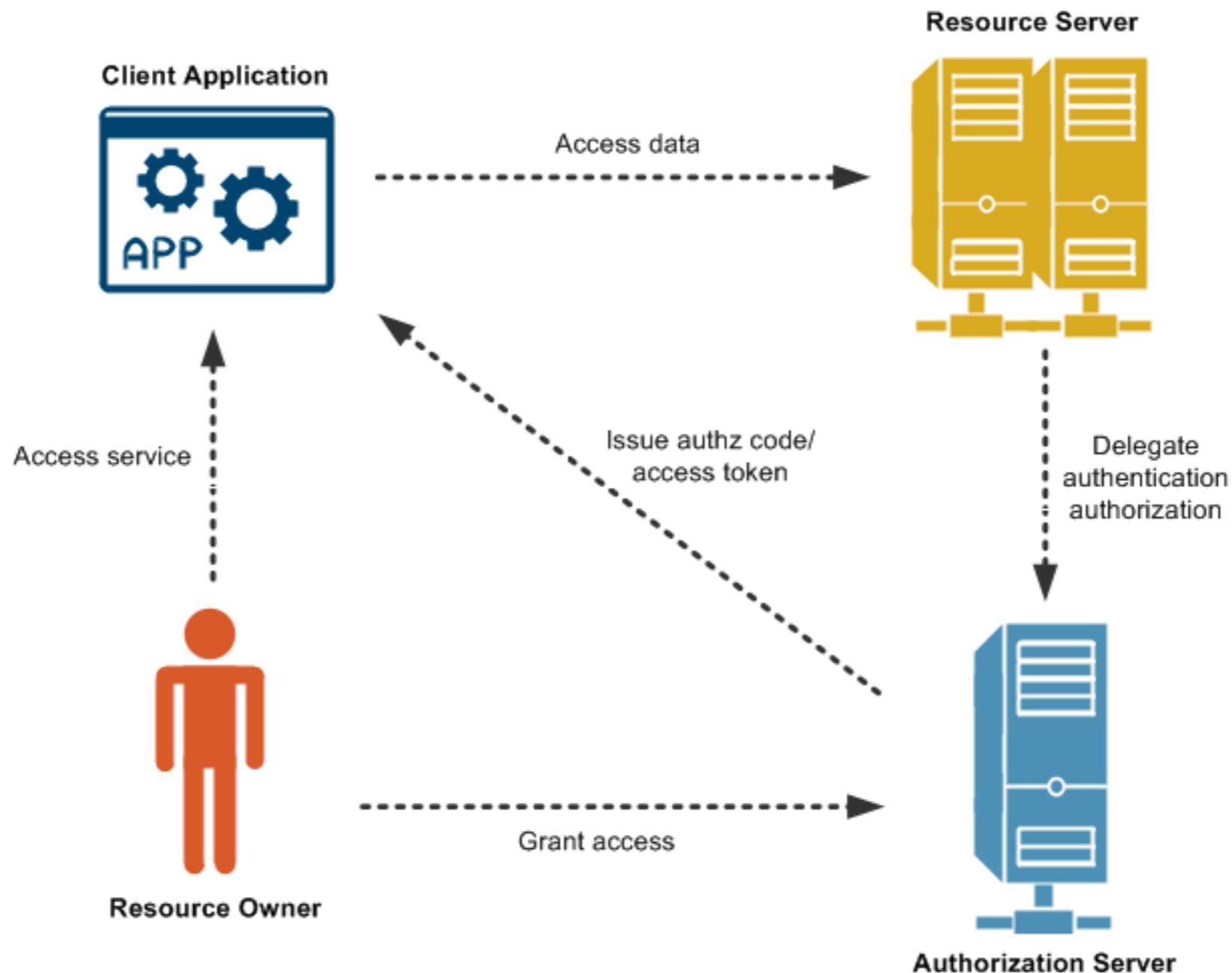




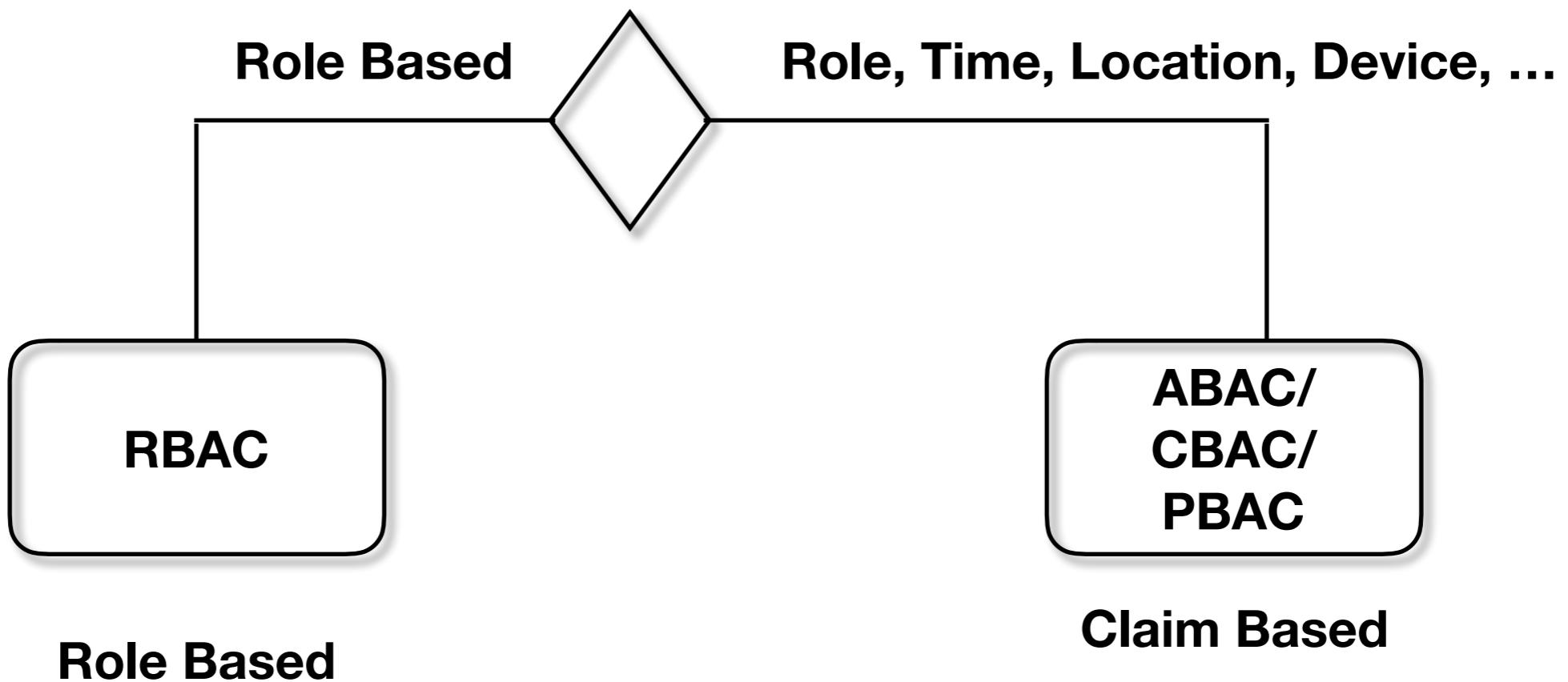




# Centralize



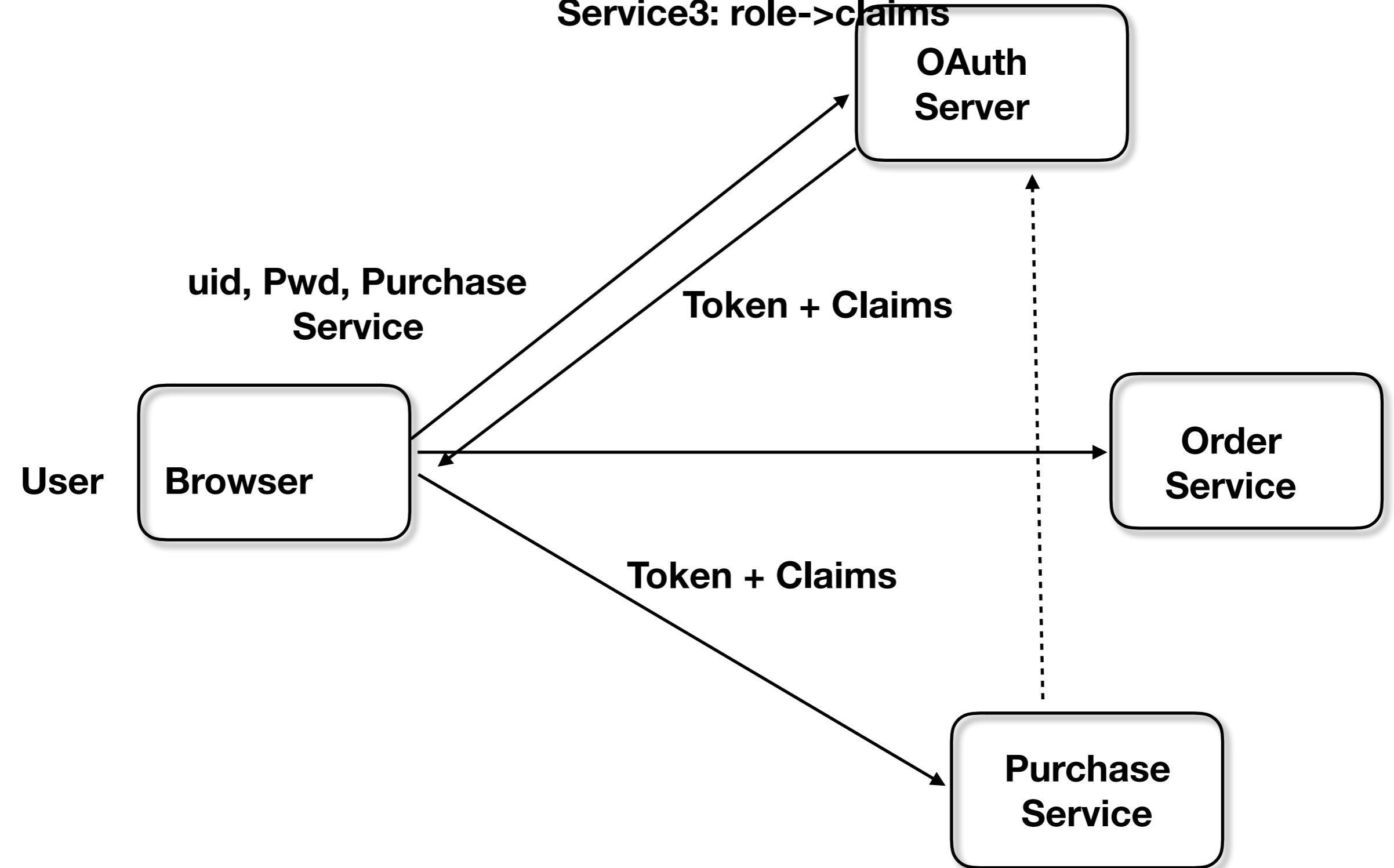
# Step 2. Authorization



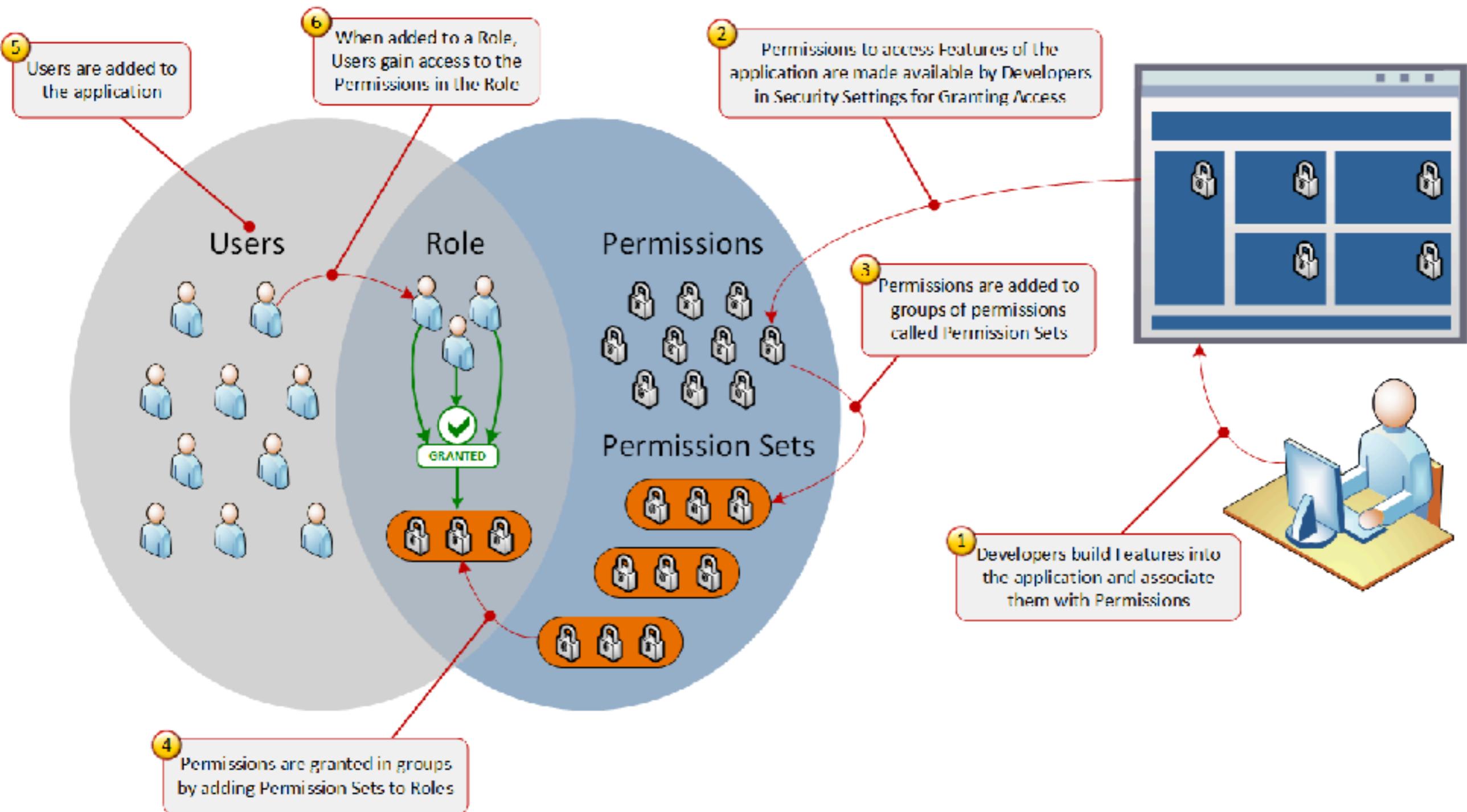
**Purchase Service: location:ny ->create new  
location:fe ->delete PO**

**Service2: role->claims  
Service3: role->claims**

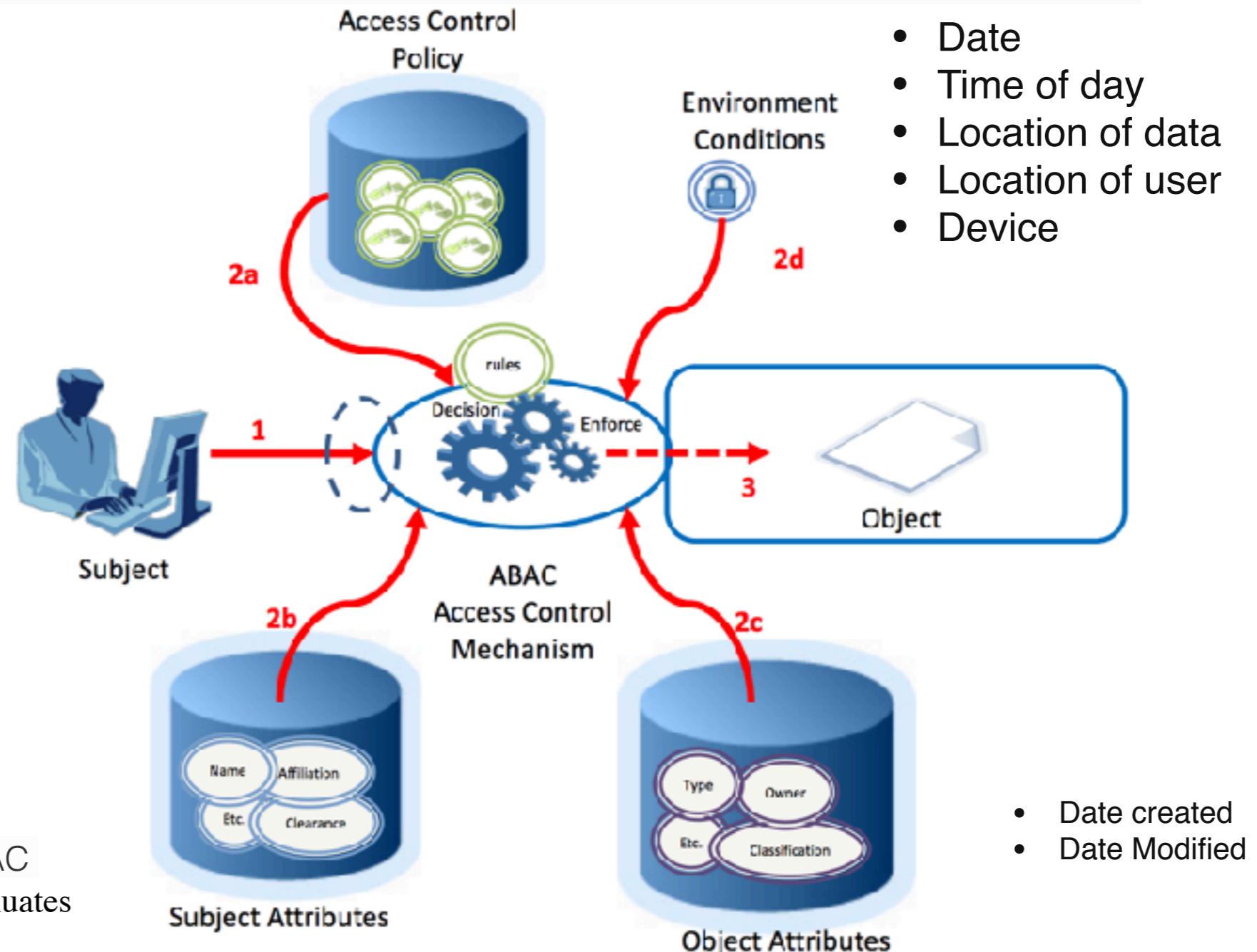
**OAuth  
Server**



# Role Based Access Control (RBAC)



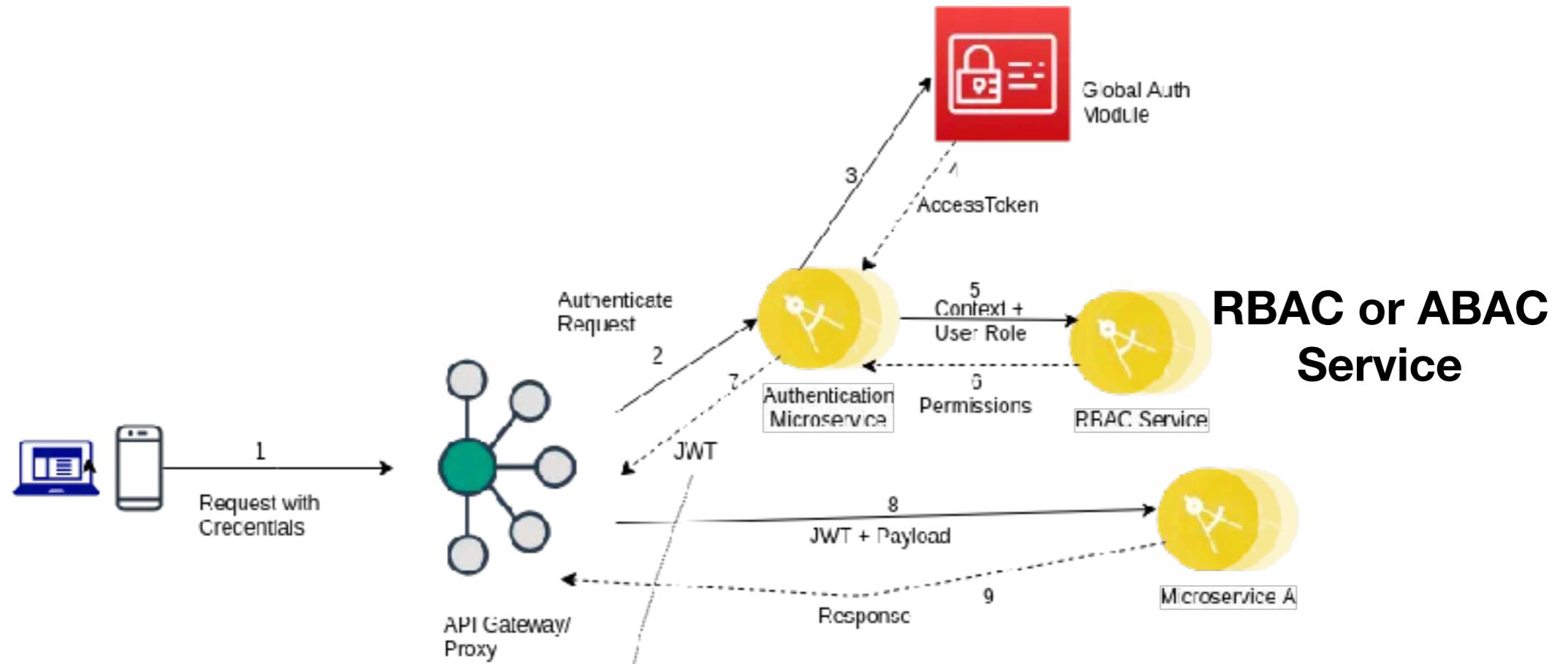
# Attribute Based Access Control (ABAC) / Claim Based Access Control (CBAC)



ABAC is an extension of RBAC  
Access Control Mechanism evaluates

- Rules (RBAC),
- Subject Attributes,
- Object Attributes
- Environment Conditions

to compute a decision



```
{
  "UserData": {
    "userId": "#{USER_ID}",
    "email": "#{USER_EMAIL}",
    "first_name": "#{USER_FIRST_NAME}",
    "last_name": "#{USER_LAST_NAME}",
    "name": "#{USER_NAME}",
    "roles": [
      "#{ROLE_NAME}": "#{ROLE_ID}"
    ],
    "permissions": [
      "#{PERMISSION_NAME}": "#{PERMISSION_ID}"
    ]
  },
  "exp": 148761231
}
```

# Step 3. Audit



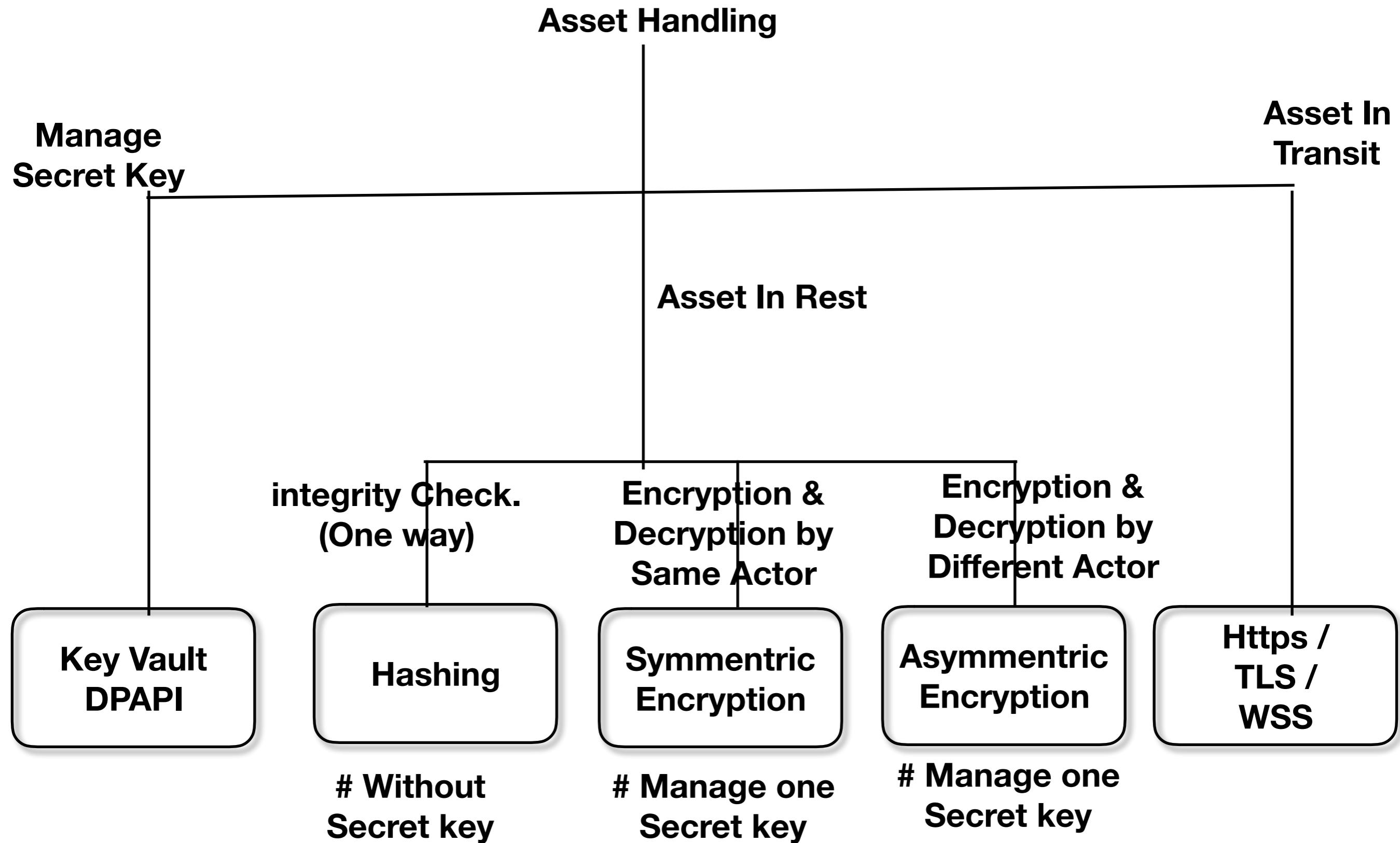
**who - Identity**

**what - Action**

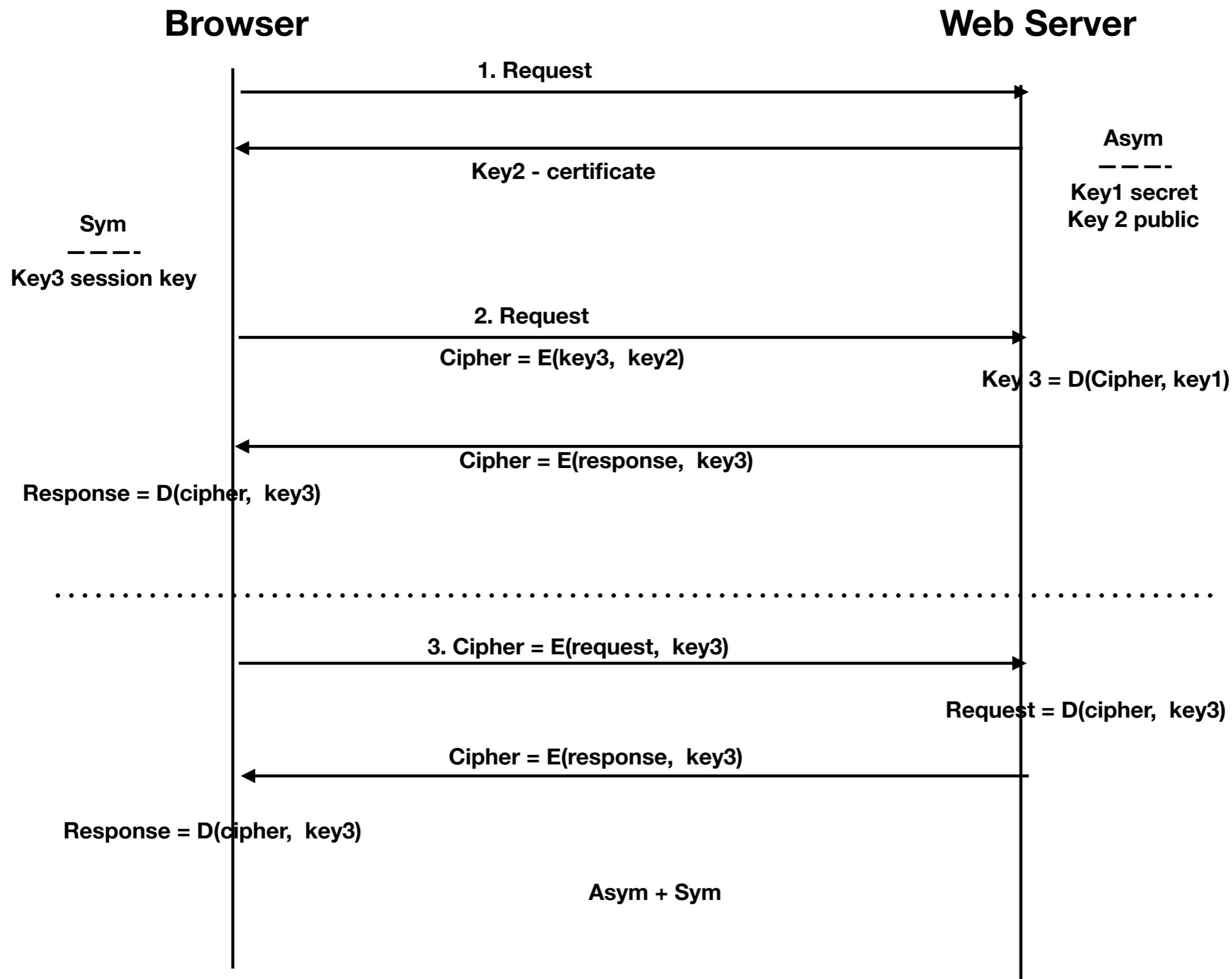
**where - Component/Service/Object/Method**

**when - Timestamp:**

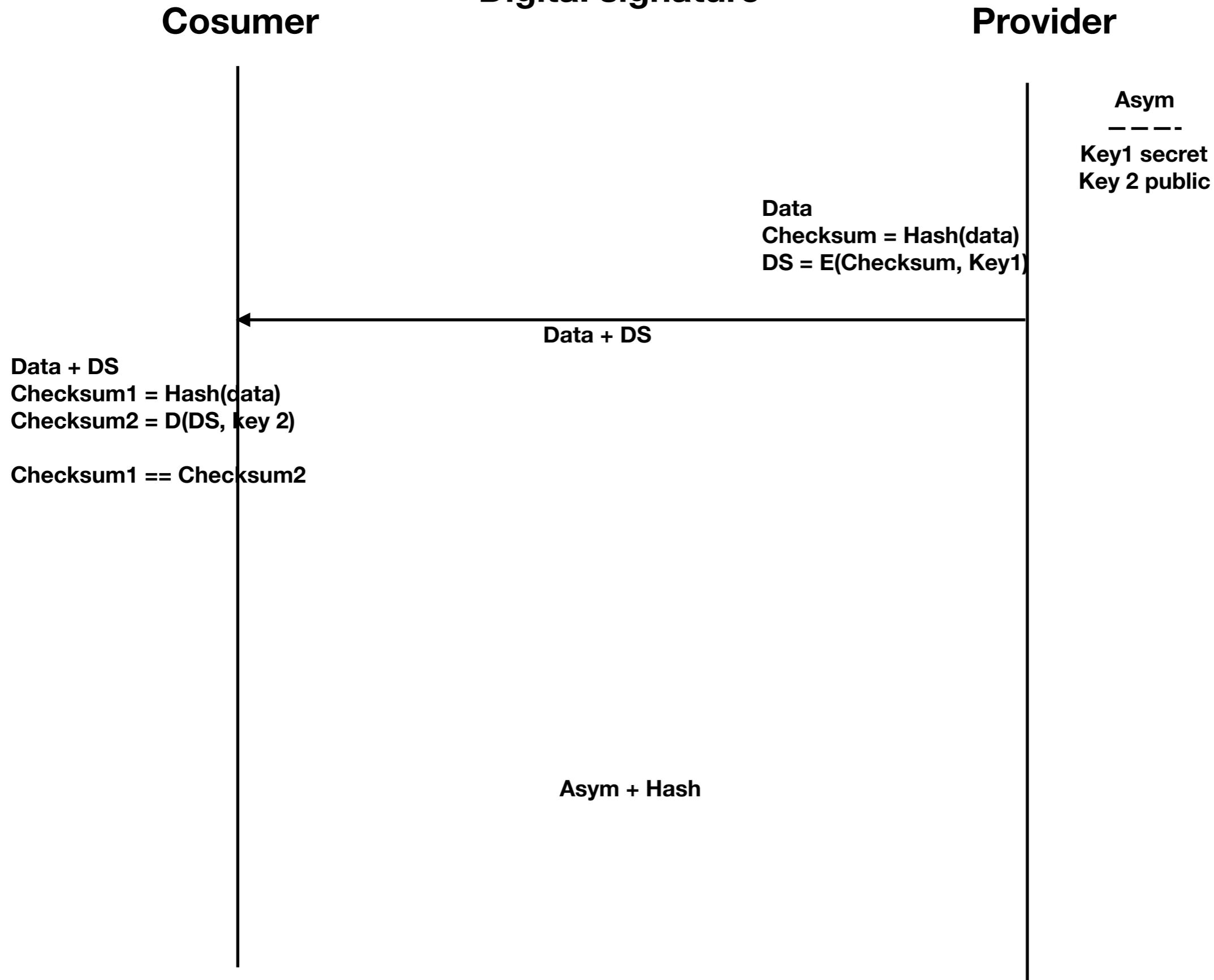
# Step 4. Privacy and Confidentiality

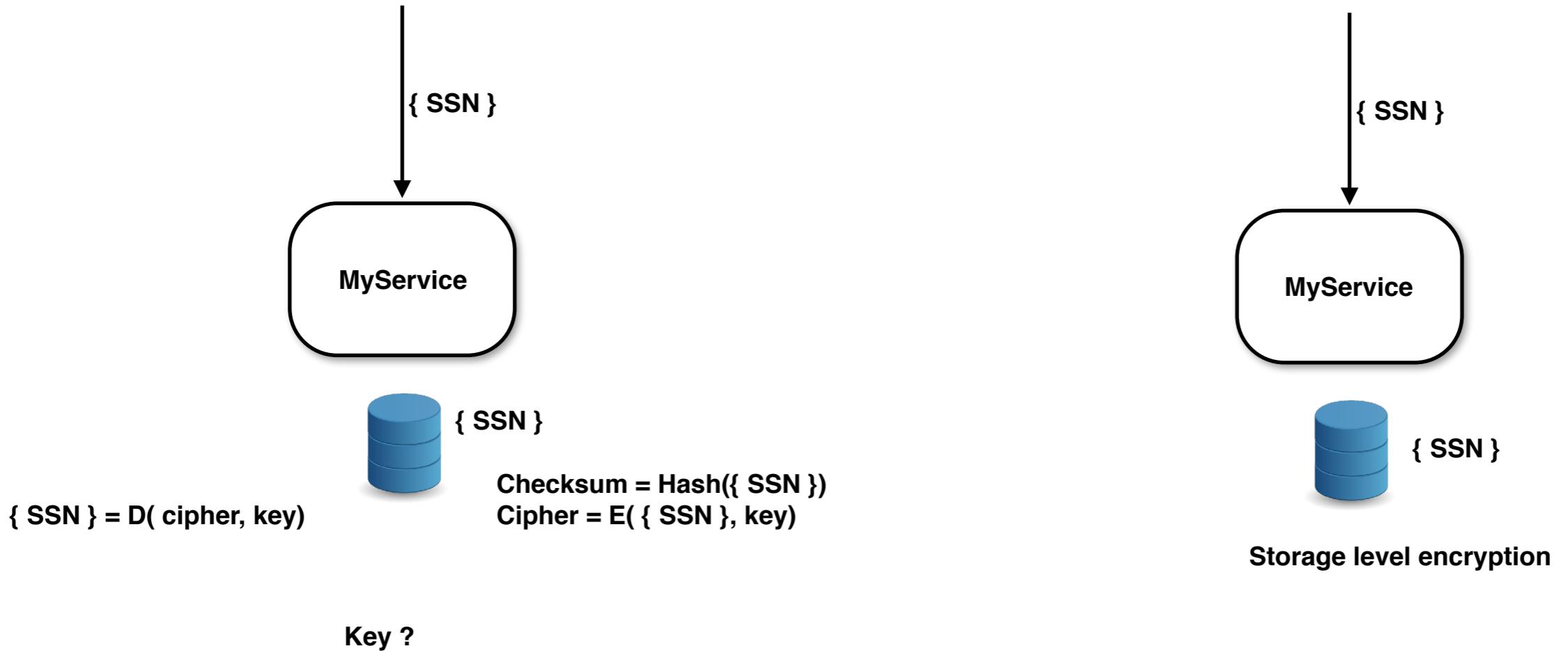


# SSL

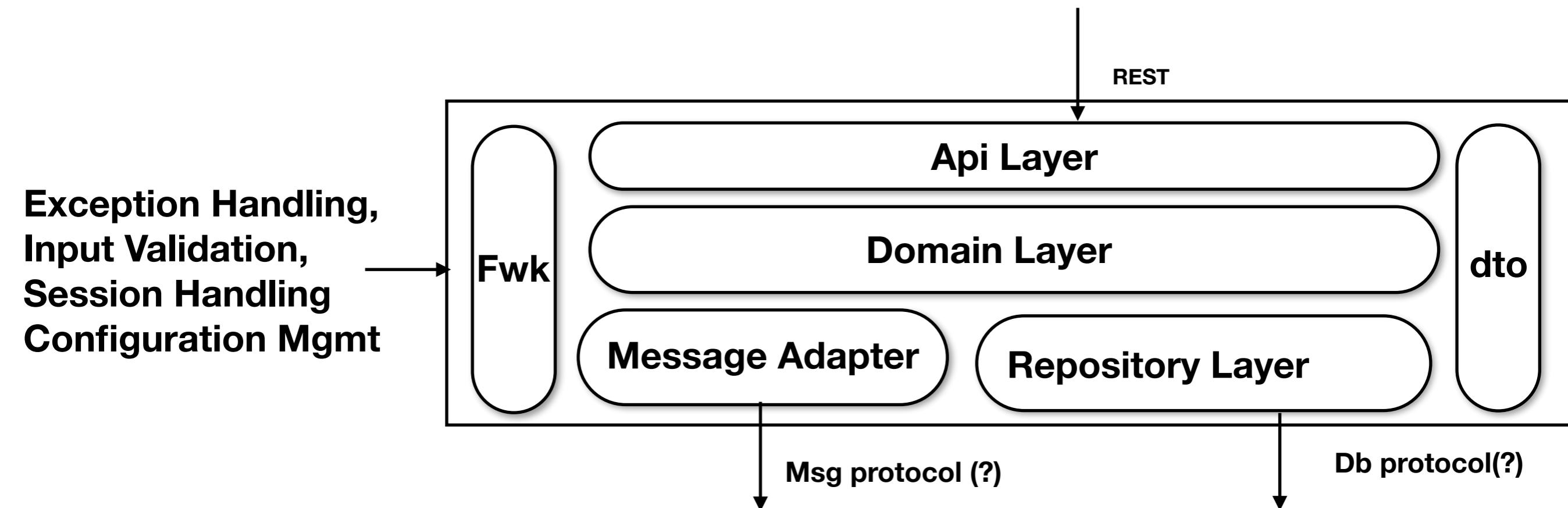


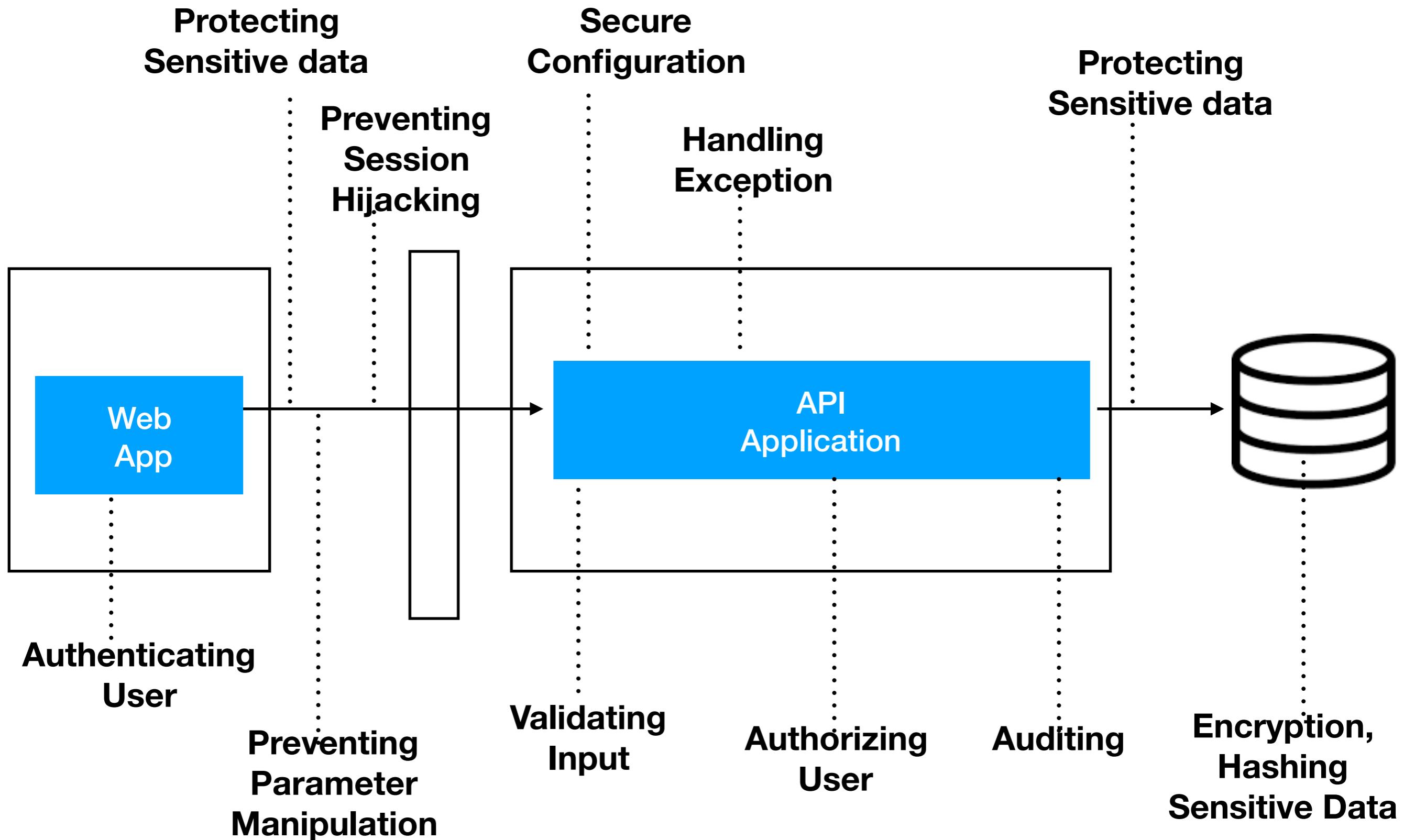
## Digital signature





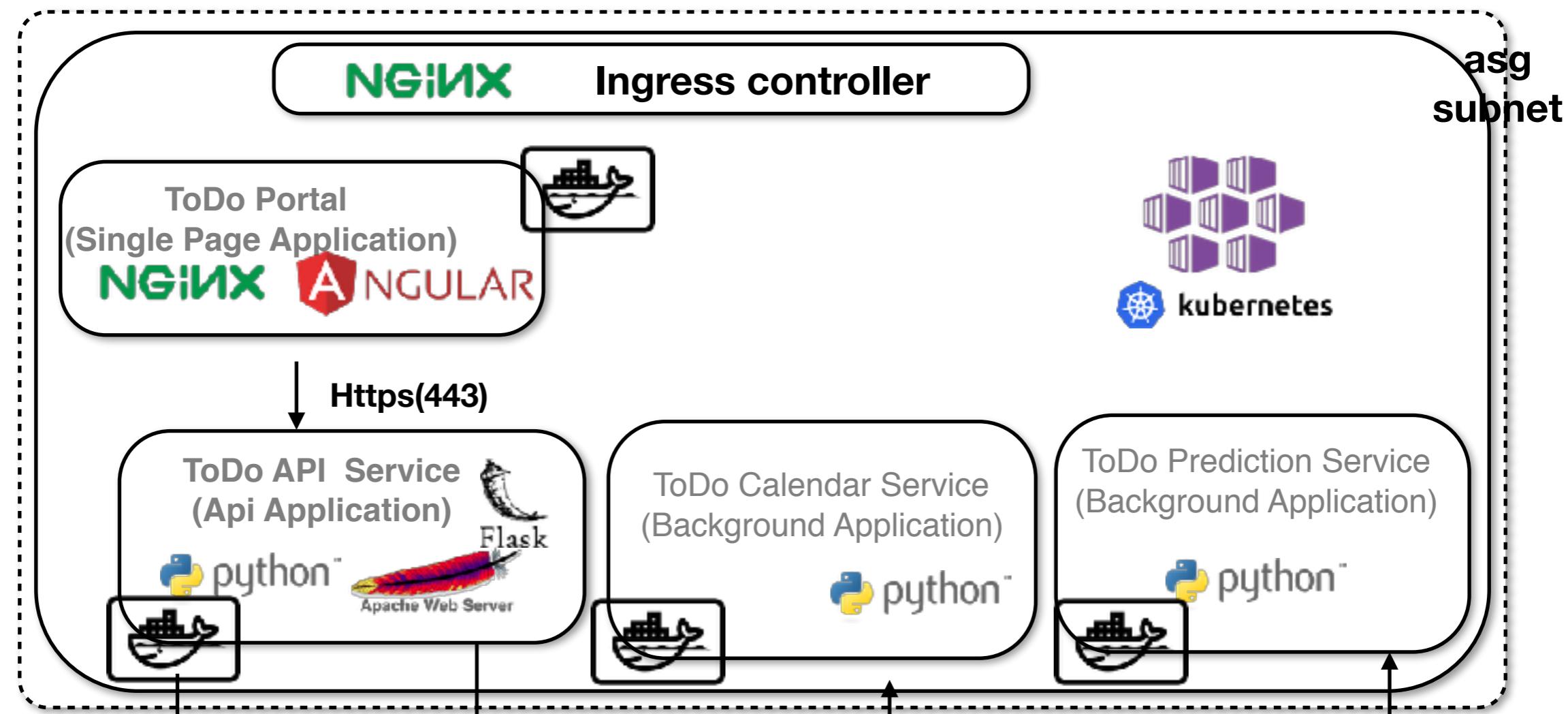
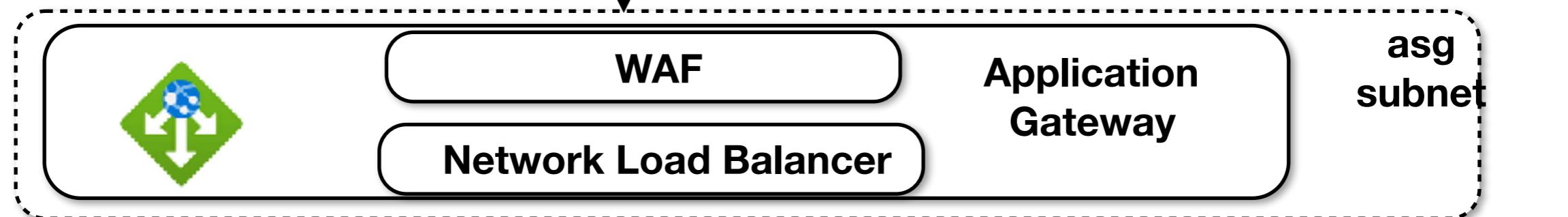
# Step 5. Cross cutting concerns



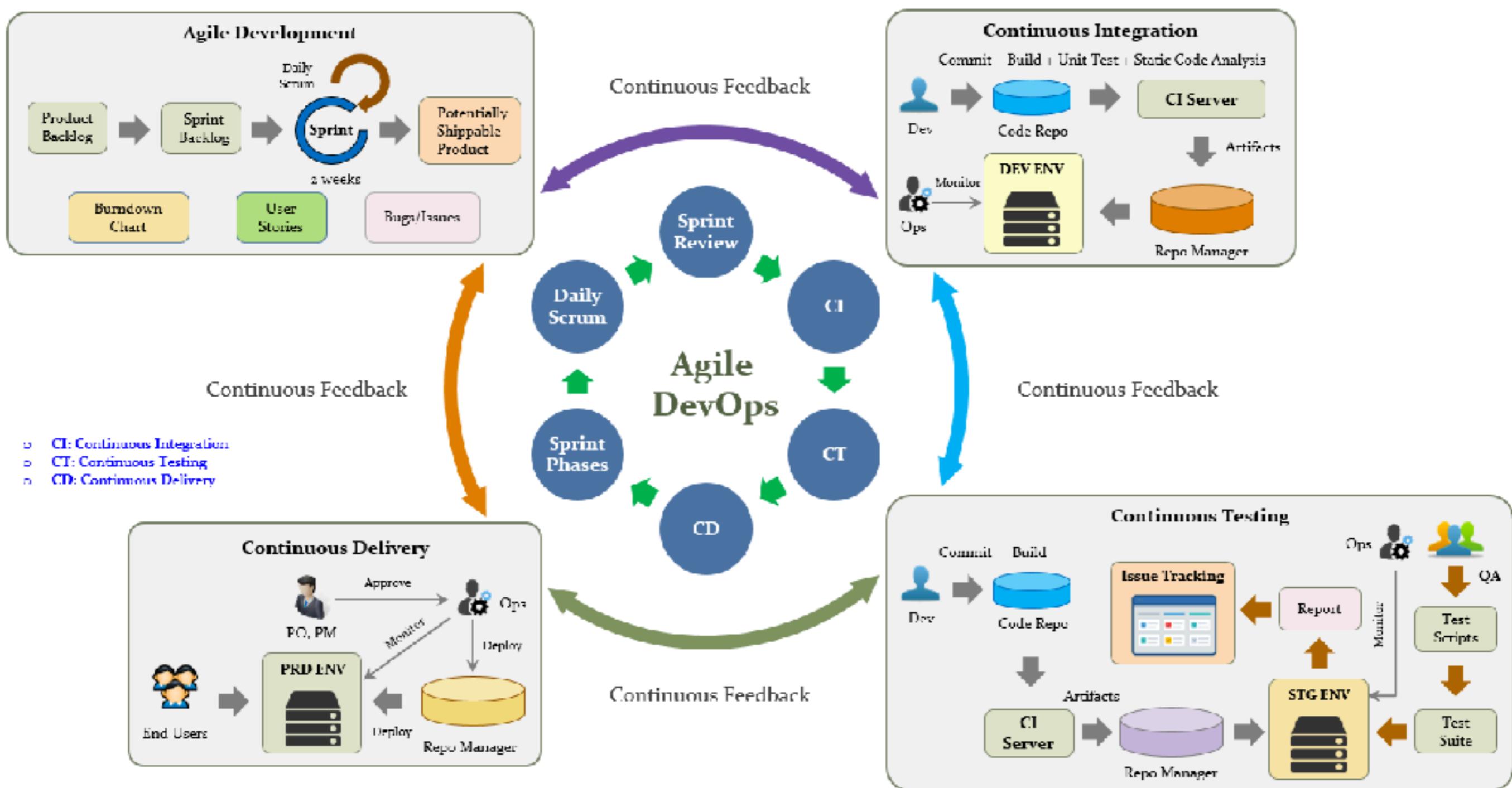


vnet

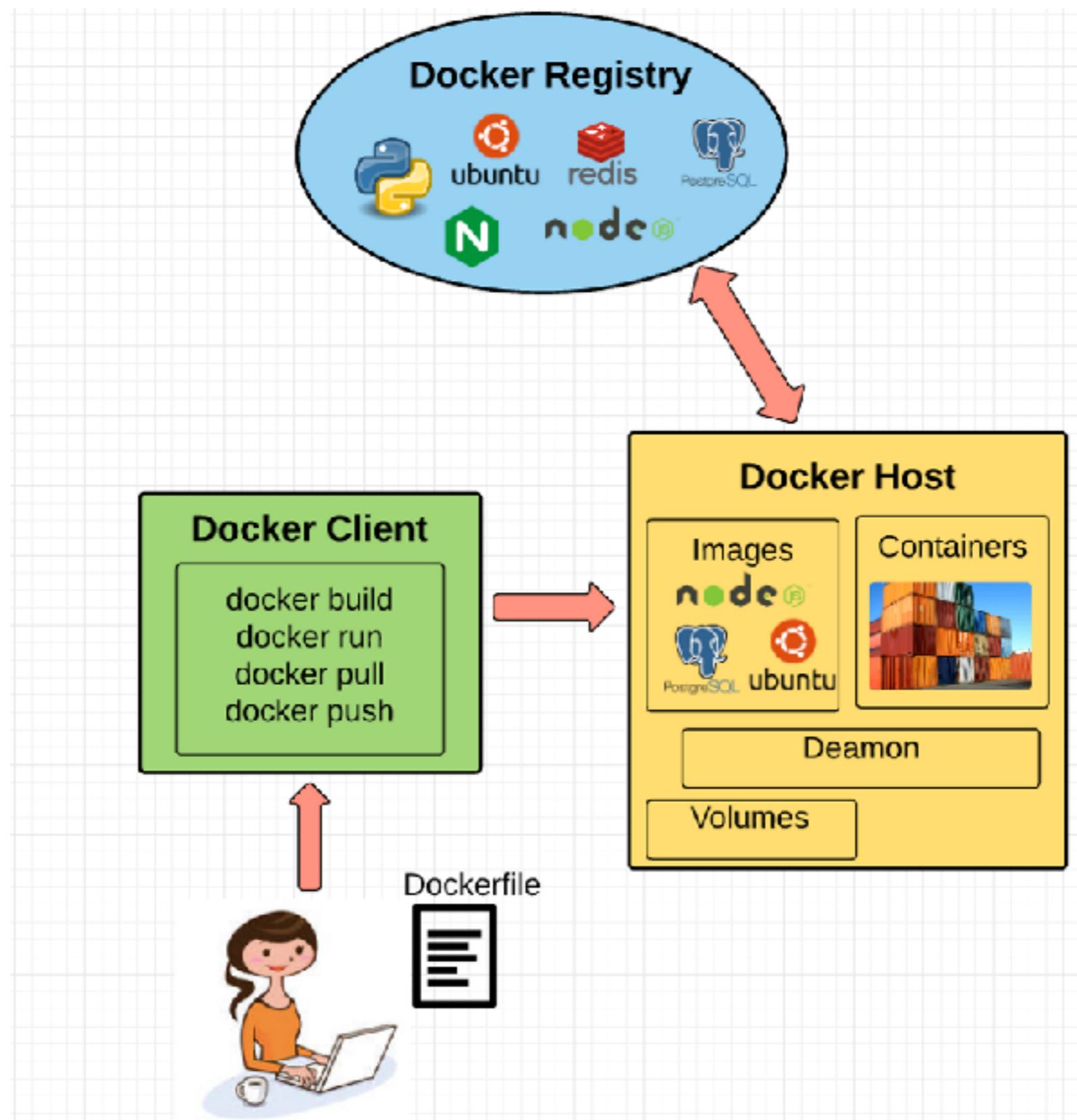
Public IP address



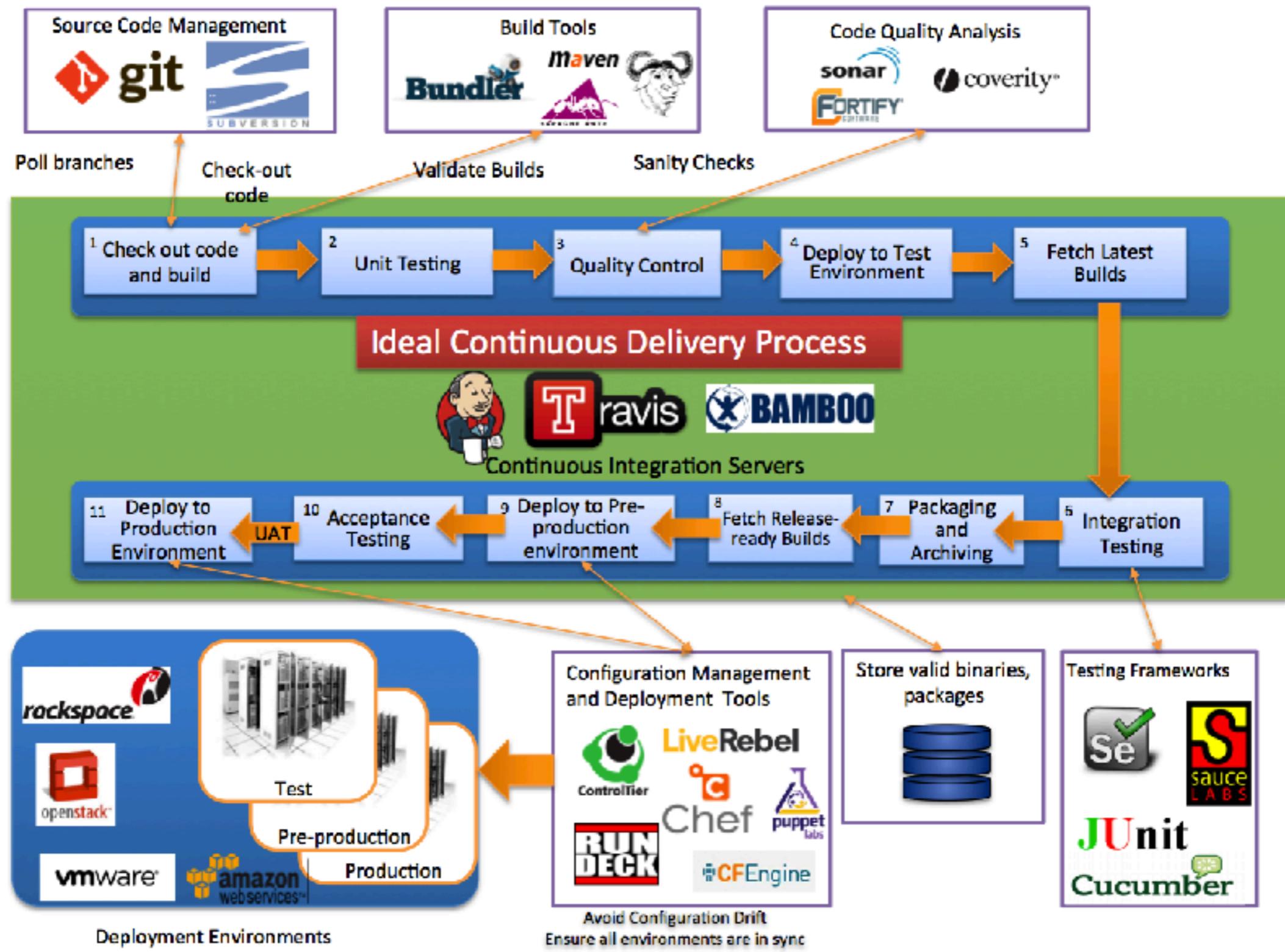
# DevOps View



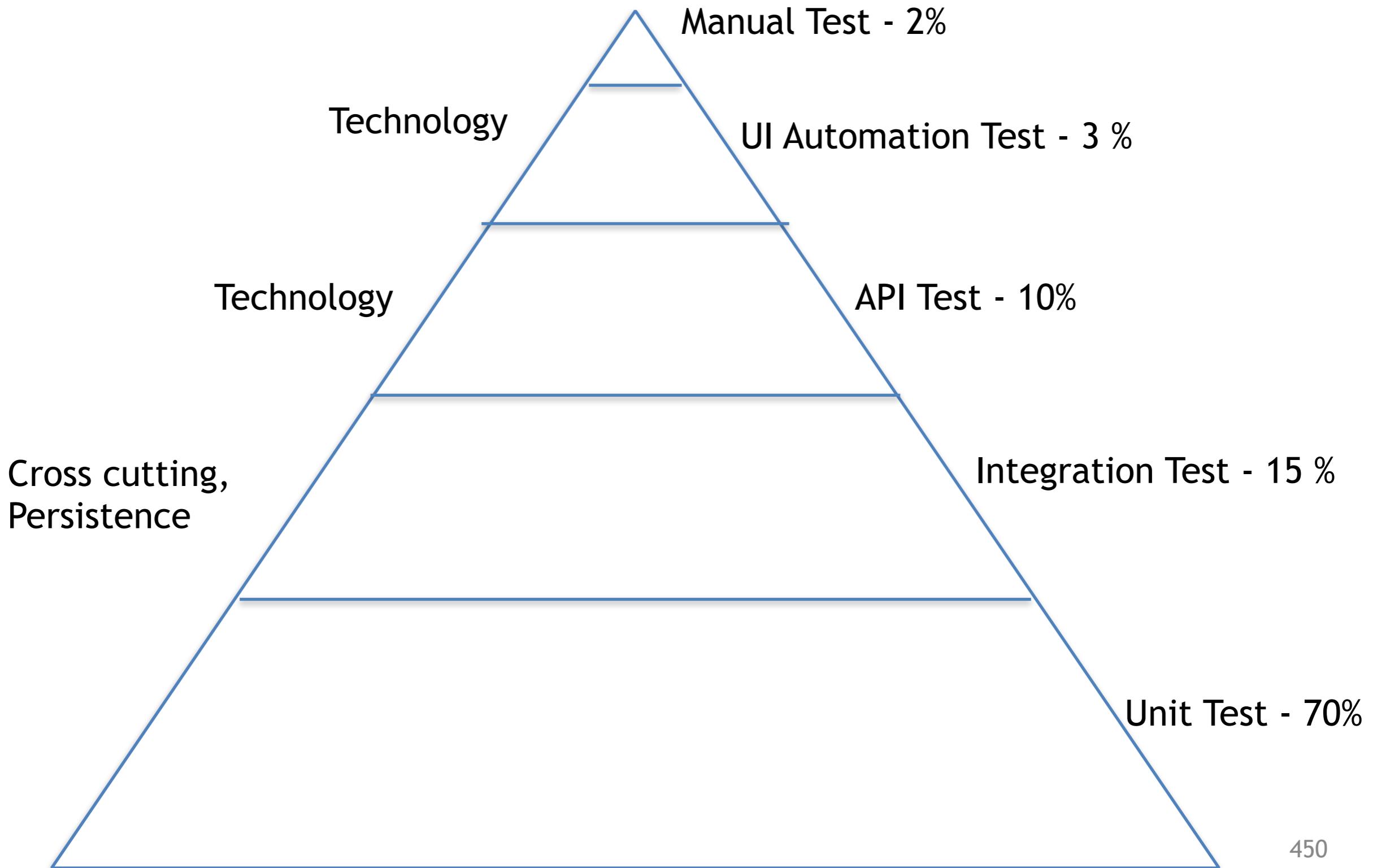
# Step 1. Containerizing



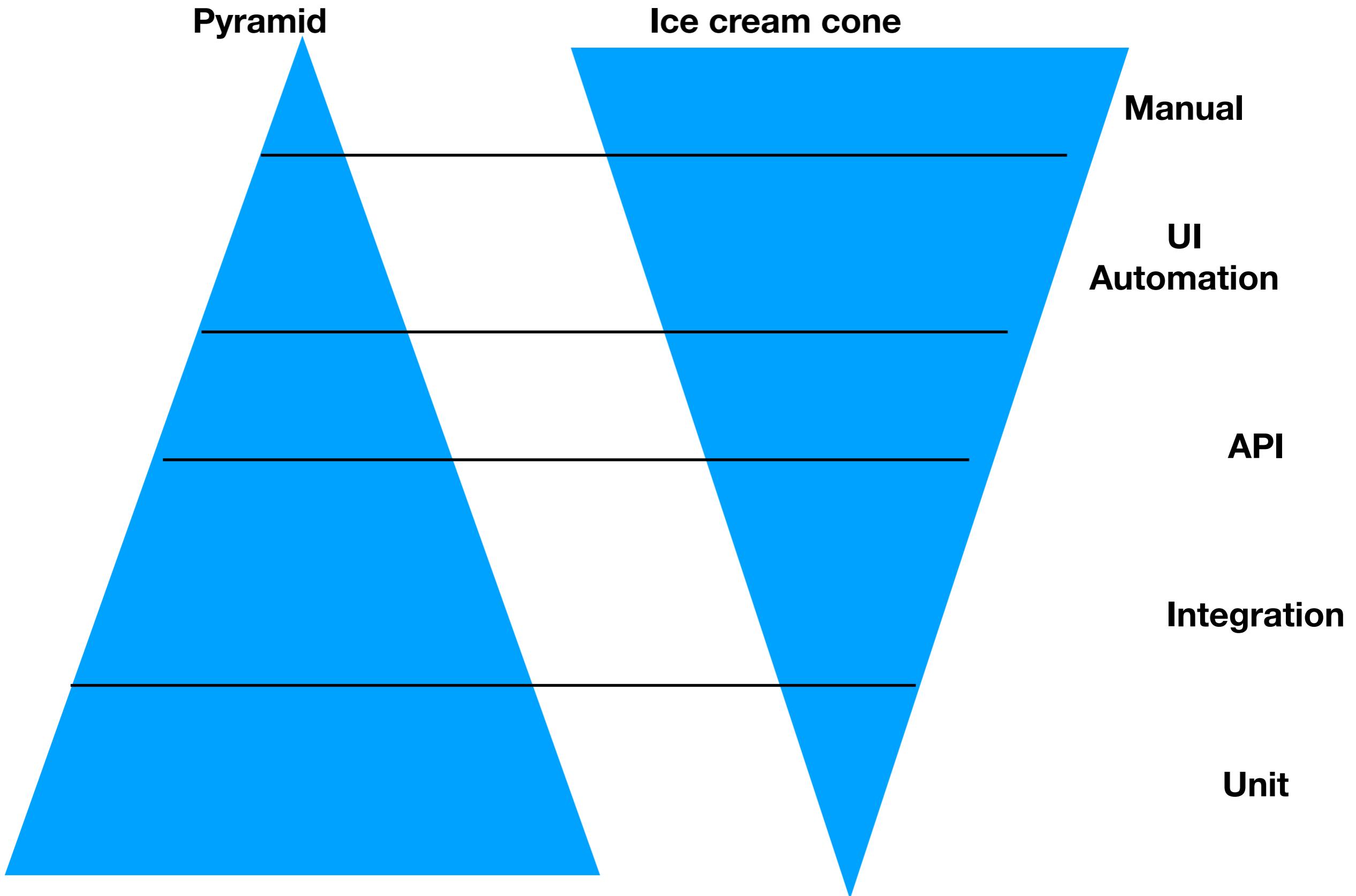
# Step 2. Integrating infrastructure automation with CI/CD

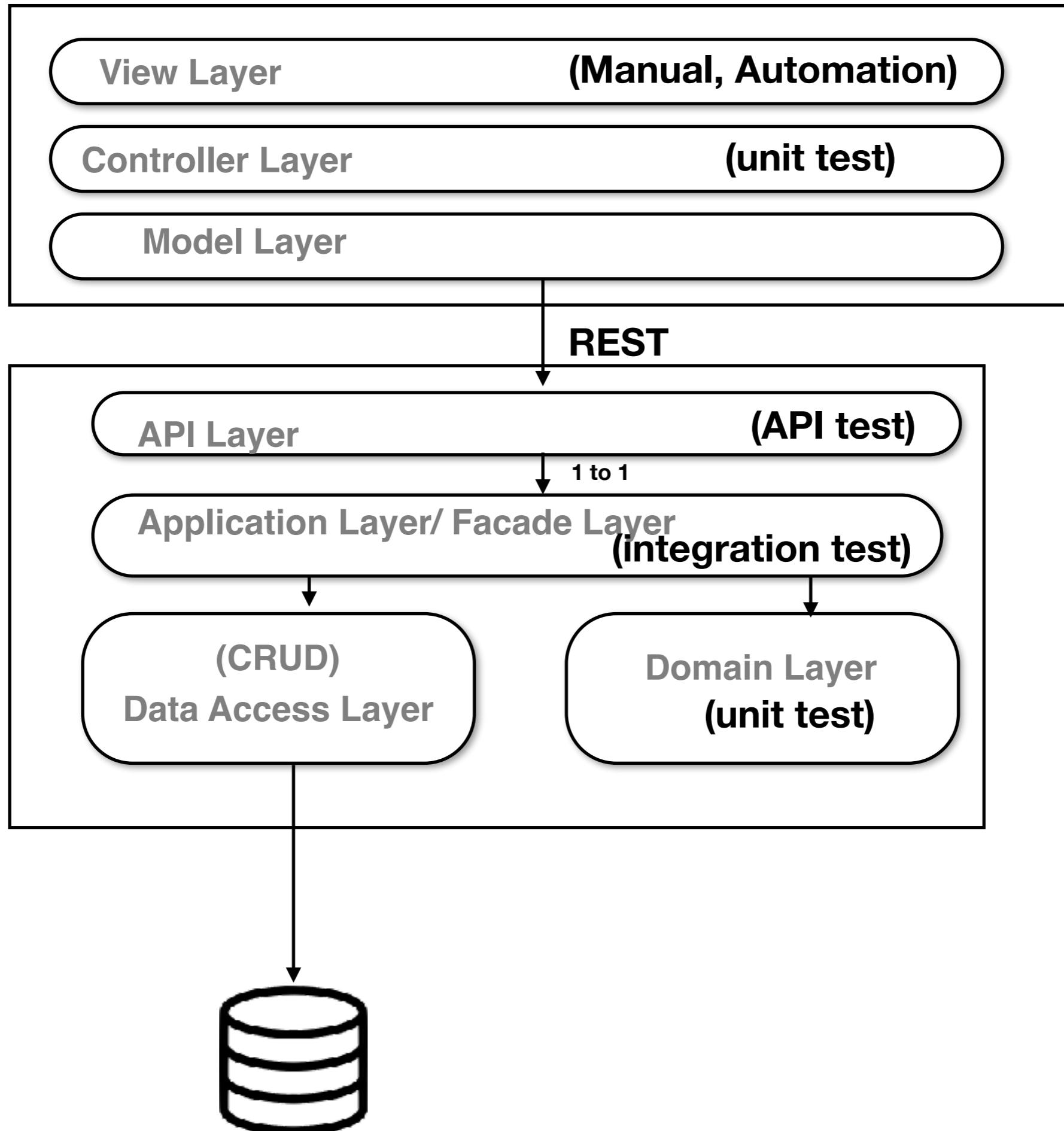


## Step 3. test automation and aligning QA with development

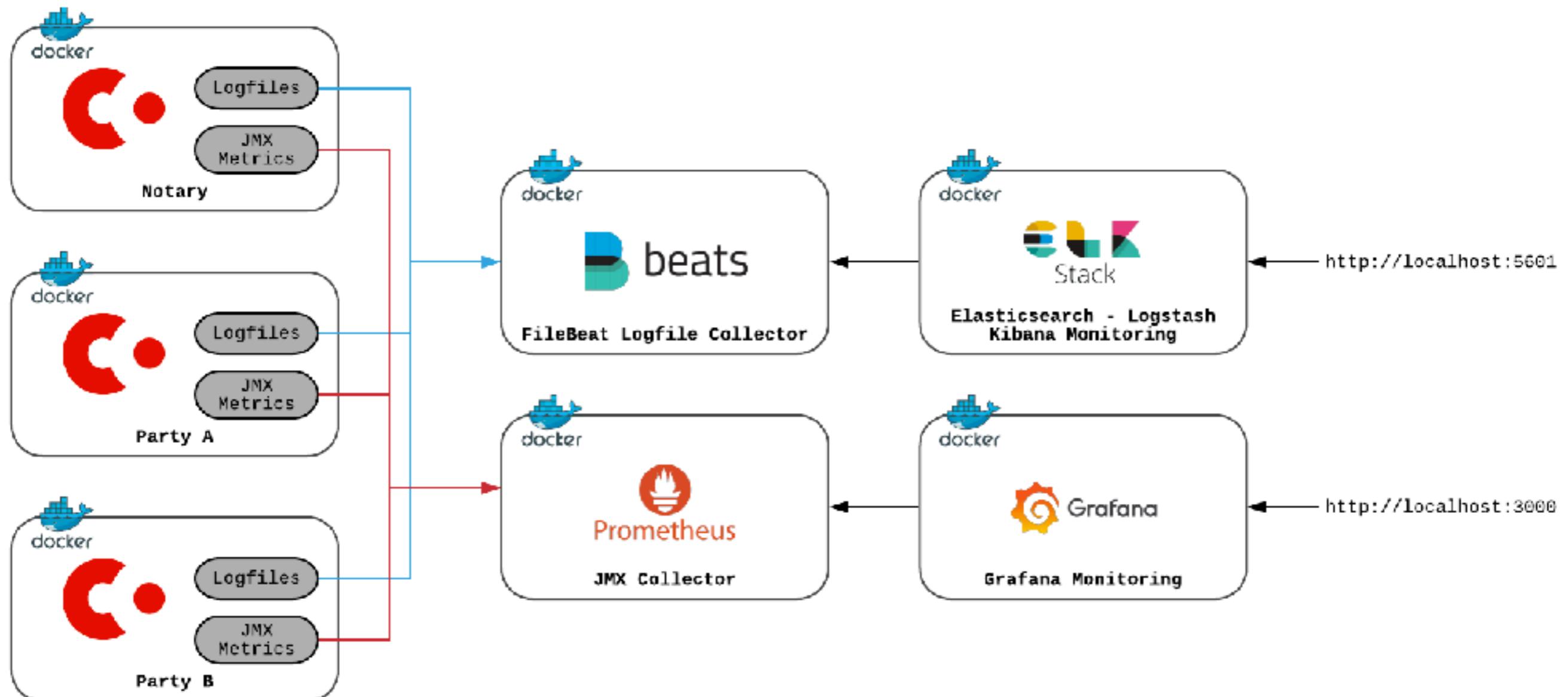


# Pyramid Test

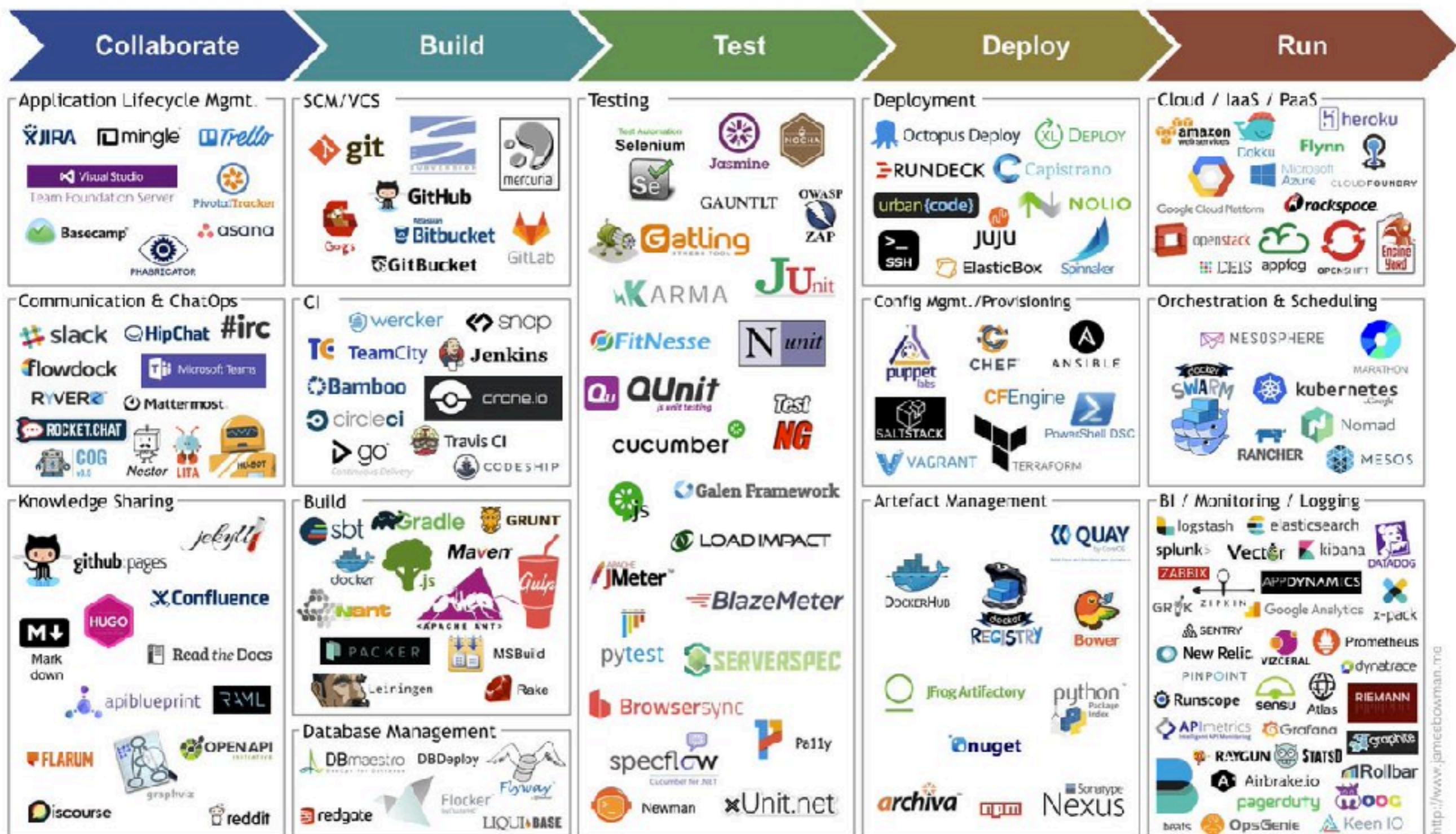




# Step 4. Setup Monitoring

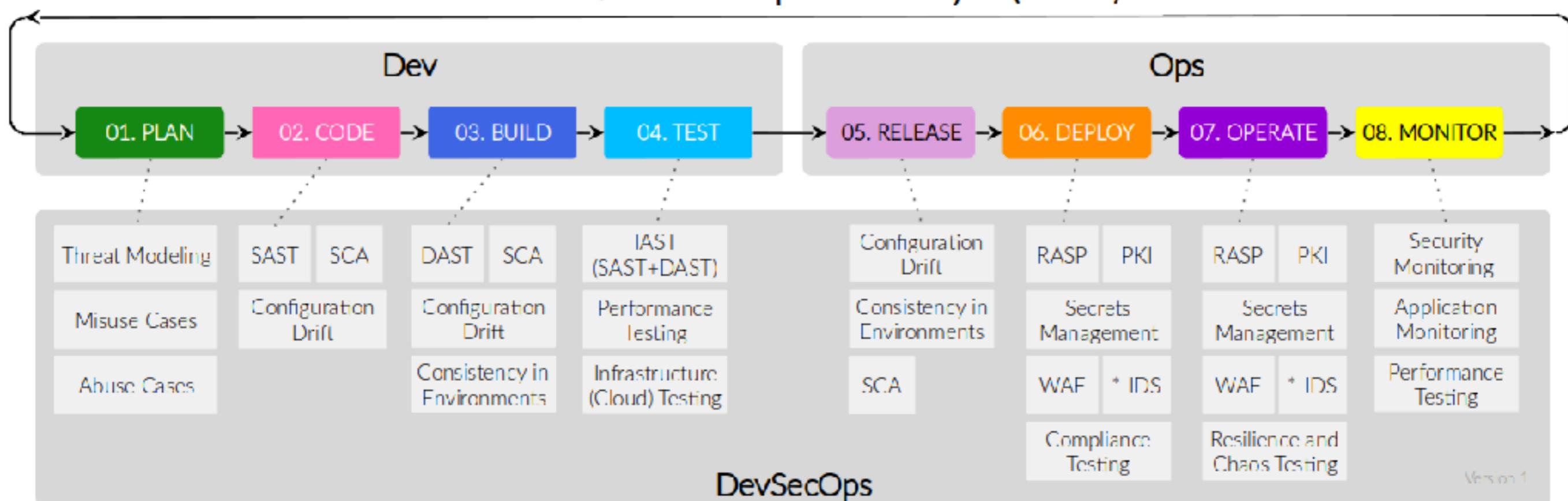


# Reference



# Step 5. SecOps

Secure Software Development Life Cycle (SSDLC)



# Arch Risk

Performance	As a user	I want to book a home	From Web Portal	During Peak Load	The booking is completed	In < 2 secs
-------------	-----------	--------------------------	--------------------	---------------------	-----------------------------	-------------

**# Compression**  
**# Concurrency**  
**# Cache**  
**# ...**

- ATAM \*
- SAAM
- ARID
- HASARD
- ...

**QAW**

# Evaluate Architecture (ATAM)

# identify all Architectural approaches  
(design)

# AD1 (CQRS)

# AD2(Sharding)

# AD3 (Caching)

...

# identify all quality requirements  
(requirements)

# s1 (< 5 sec)

# s2 (99.99%)

# s3 (...)

# ...

Utility Tree

# analyse Scenario -> Approach

S1 -> A1, A2

S2 -> A6,A8, A9

S3-> ?

S4 -> A6, ?

# brainstorm for scenarios

# s8

# s9

# s10

# ...

# analyse Scenario -> Approach

S8 -> A1, A2

S9 -> A6,A8, A9

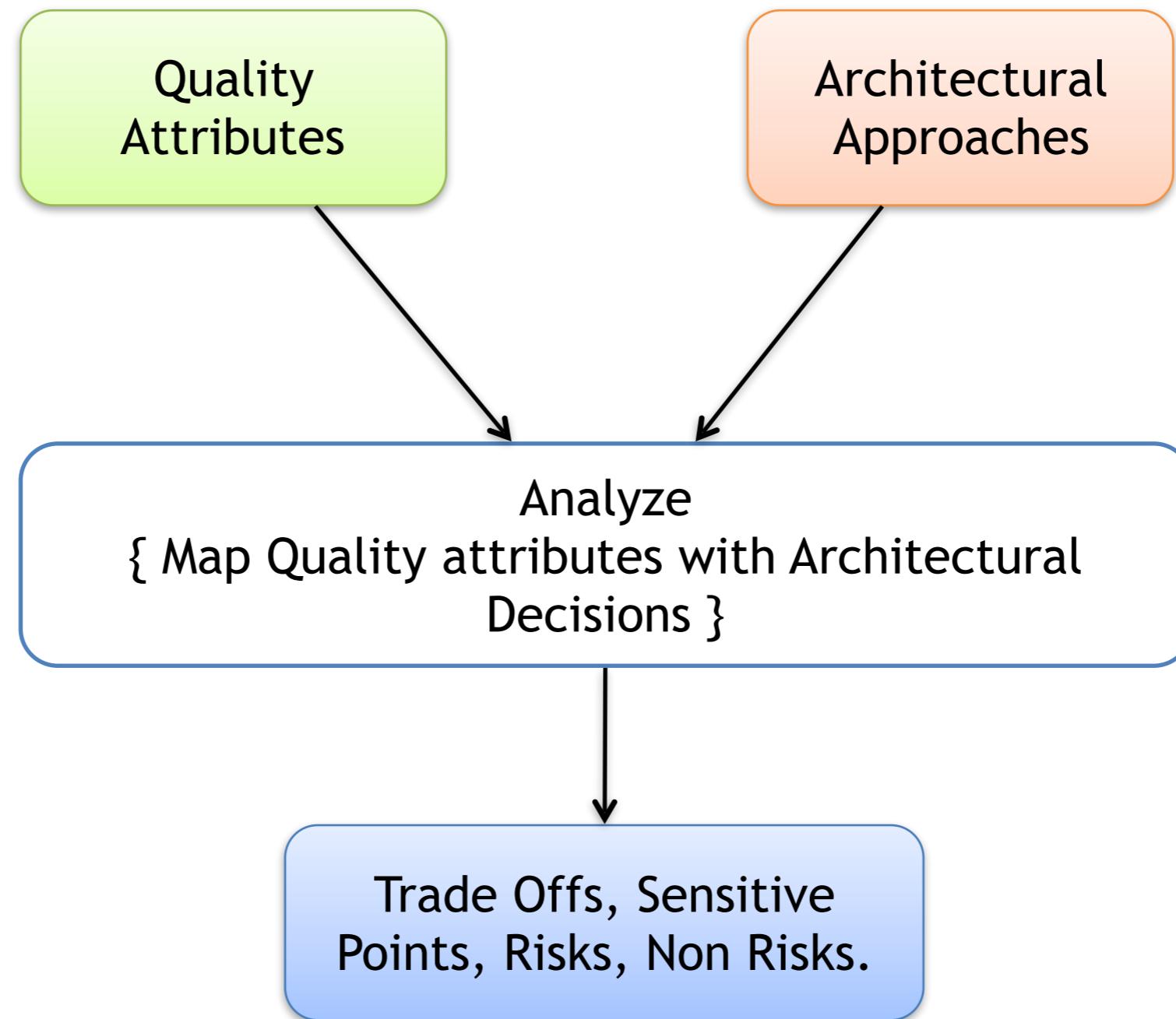
S10-> ?

=> Risk & trade off's

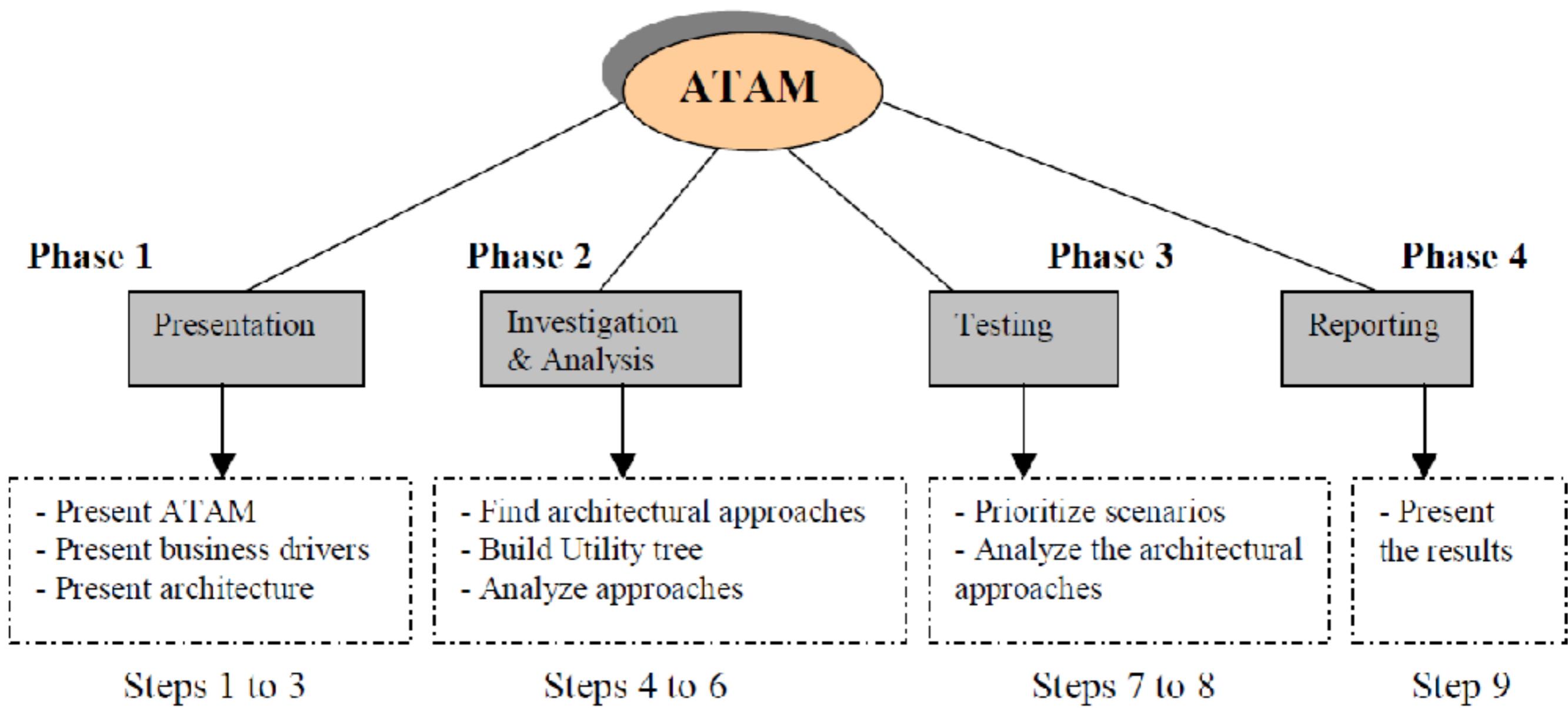
=> Risk & trade off's

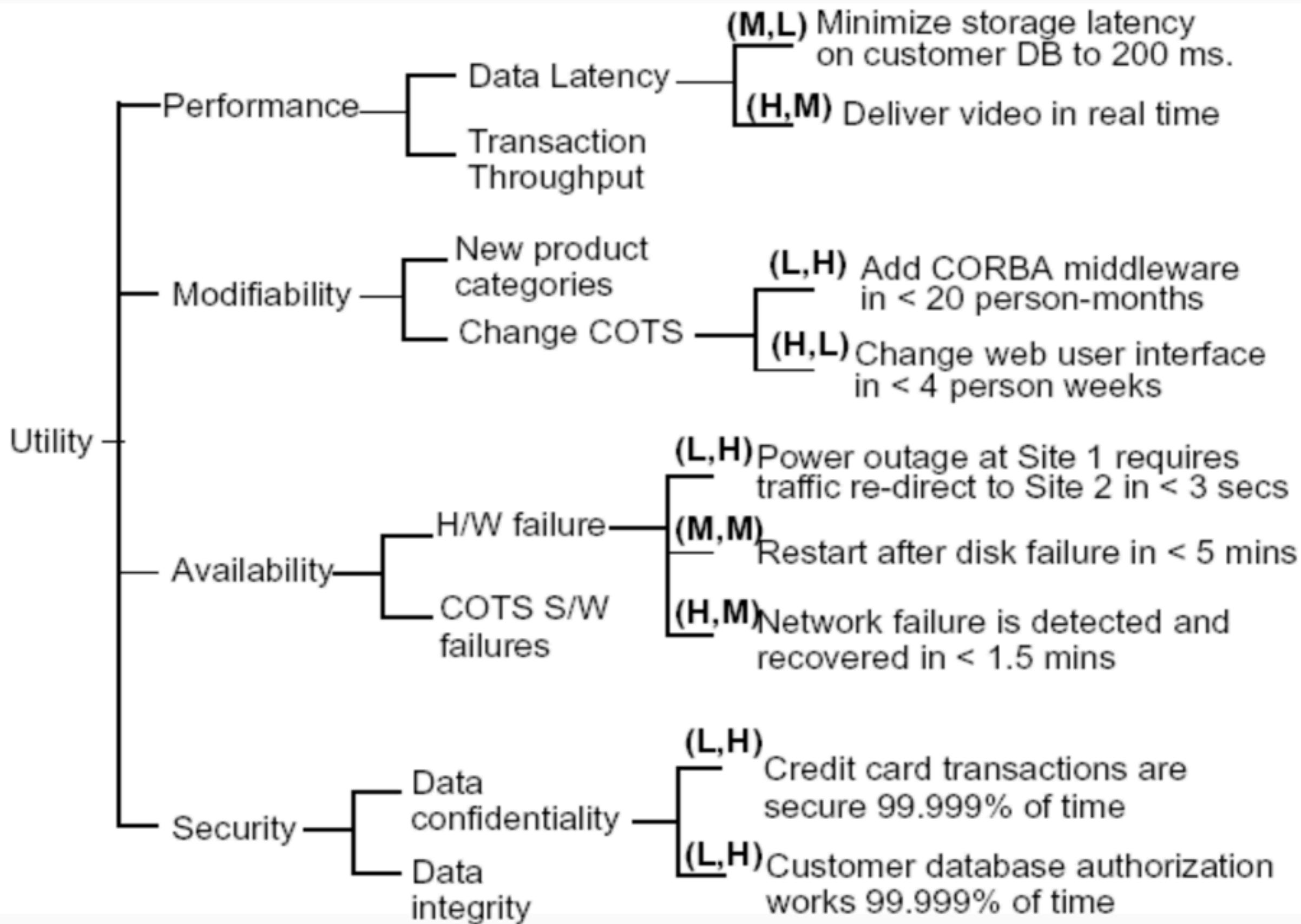


Evaluation Team



Offering mitigation strategies is *not an integral part* of the ATAM.  
ATAM is about locating architectural risks.





# Scenario Utility Tree

- **Utility**
  - **Performance**
    - Data latency
      - Minimize storage latency on customer DB to 200 ms
      - Deliver video in real time
    - Transaction throughput
      - Maximize average throughput to authentication server
  - **Modifiability**
    - New Product Categories
    - Change COTS
      - change web user interface in < 4 person weeks
  - **Availability**
    - Hardware Failure
      - power output at site 1 requires traffic redirect to site 3 in < 3 s
      - network failure is detected and recovered in < 1,5 min
  - **Security**
    - Data confidentiality
      - customer database authorisation works 99,999% of time

# Scenario Brain Storming

Sc#	Description	Quality Att.	Votes
4	Dynamically replan a dispatched mission within 10 minutes.	Performance	28
27	Split the management of a set of vehicles across multiple control sites.	Performance, Modifiability, Availability	26
10	Change vendor analysis tools after mission has commenced without restarting system.	Integrability	23
12	Retarget a collection of diverse vehicles to handle an emergency situation in less than 10 seconds after commands are issued.	Performance	13
14	Change the data distribution mechanism from CORBA to a new emerging standard with less than six person-months' effort.	Modifiability	12

## Architect Centric



Concentrates on eliciting and analyzing architectural information.

## Business Drivers

### Phase - 1

### Phase - 2

## Stakeholder Centric



Elicits points of view from a more diverse group of stakeholders, and verifies the results of the first phase

Scenario:	E-connector loses connection with SAP
Attribute:	Availability
Stimulus:	Temporary network fault
Response:	The system has an overall availability of 99,25% (max 2 hour down/month)

Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
DCTM content server runs on a clustered environment with 2 nodes	Common mode failure can not be handled			Probability of common mode failure is low
Integration relationship between SAP and Documentum is 'data consistency' and is not protected	Human user must report malfunction		From complaint to resolution > 2 hours	

Scenario:

Invoice poster needs e-document for data entry in SAP/R3

Attribute:	Perfomance-Latency
Stimulus:	Document request to Documentum
Response:	Document is available for processing in less than 10 s

Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
E-documents are scanned in color at 200 dpi	Size of document is sensitive to quality of scanning	Usability vs Performance	Document too large for roundtrip in 10 s.	
E-documents are not cached	Every document must be fetched from DMTM	Development cost vs bandwidth cost	Document roundtrip time exceeds 10 s.	

## # Gather Architectural Requirements

- Context View (overview)
- Functional View (overview)
- Constraints (c1,c2,c3,...)
- Quality View (s1,s2, s3,...)

## # Make Architectural Decision

- Logical view
  - Decompose
  - Technology (storage, compute, communication)
  - Cross cutting
- Security View
- Deployment View

## # Evaluate Architectural Decision

- List all Architectural decisions (a1,a2,a2,.....)
- List all Quality Scenarios (s1,s2,s3,....)
- Analyse scenario Approach
  - \*s1->(a1,a2)
  - \*s2->(a3)
  - \*s3->(?)
  - \*s4->(a2, ?)
  -

## **# Evaluate Architectural Decision**

Used a Event Bus to loosely couple the system

---

Used a GraphDb to search trains efficiently

---

Used a Cache db for frequent search

---

Decomposed the system in Microservices

---

Microservices are deployed in K8s cluster

---

...

<p>As a user I want to search for a train from Mobile Client . The search result is shown to user In &lt; 3 secs</p>	<p>Used a GraphDb to search trains efficiently</p>	<p>Support for Graphdb ?</p>
<p>The database System Crashes In the Data Centre During Operational Hours Secondary is made Primary In &lt; 2 minutes</p>	<p>Used a Cache db for frequent search</p>	<p>How to handle the expiry of cache</p>
<p>The database System Crashes In the Data Centre During Operational Hours Secondary is made Primary In &lt; 2 minutes</p>	<p>Shard based on Zones (4)</p>	
<p>As a user I want to use the portal when 100000 users( peak load) are active</p>	<p>Add 2 Secondary nodes for DB</p>	
<p>As a user I want to use the portal when 100000 users( peak load) are active</p>	<p>K8s Autoscale of pods</p>	
<p>As a user I want to use the portal when 100000 users( peak load) are active</p>	<p>Microservice</p>	
	<p>Used a Cache db for frequent search</p>	