

Software Architecture



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



Mubarak

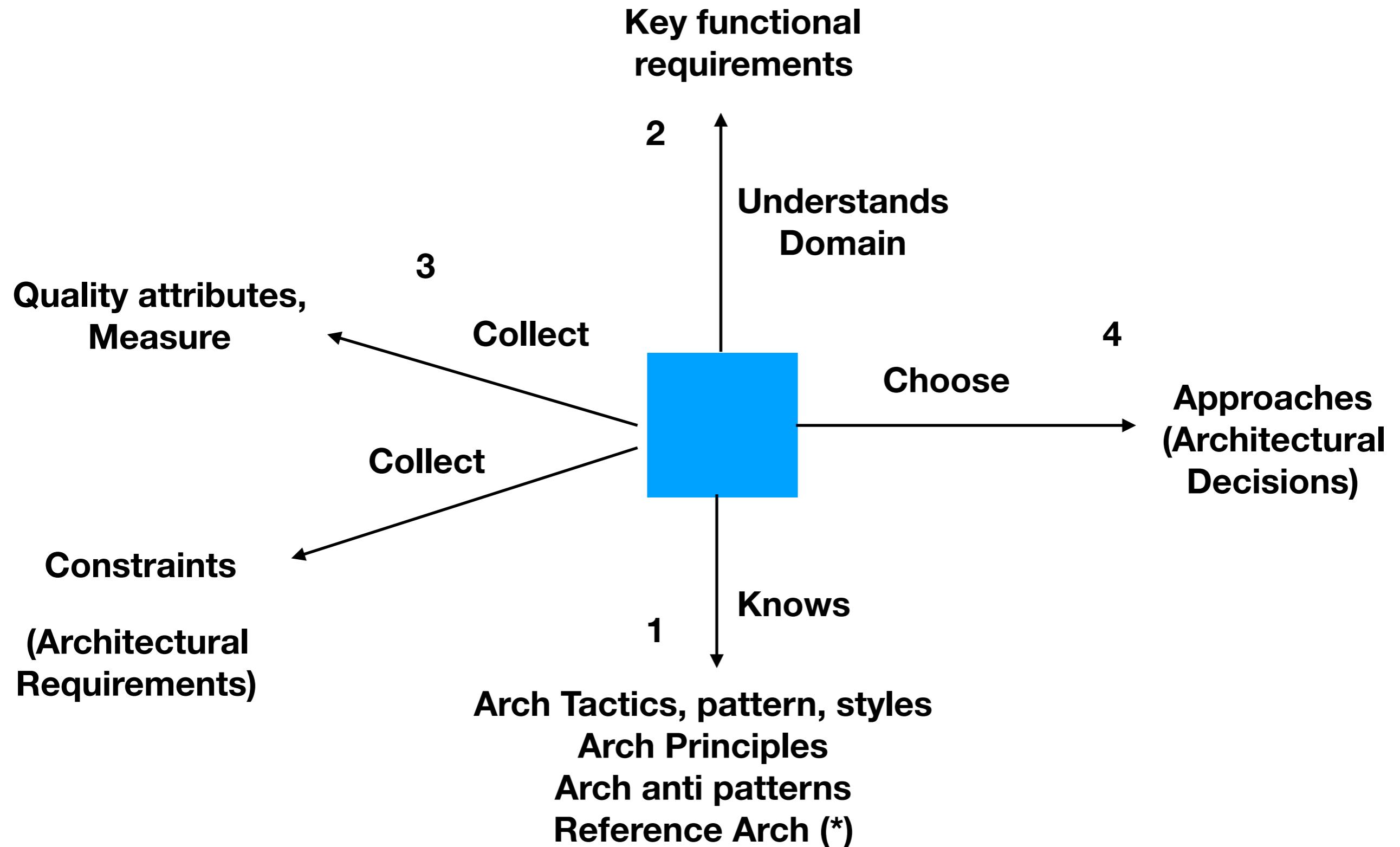


- Arch Requirements (5 views)
- Arch Design (3 ~4 views)
 # Low level detail of DWH
 # Large file processing
- Arch Doc
 Cloud neutral
 Multi tenant
- Arch Eval

Architecture vs Design

Attribute	Measure	Approach
• Performance <–		• Caching
• Maintainability <–	• Response time	• Parallel
• Security (Trust) <–	• Memory	• Pooling
• Scalability (Volume - I/O, CPU, Memory) <–	• CPU	• Lazy Loading
• Usability x	• I/O	• Compression
• Availability <–	• Latency	• Chunking
• Reliability (Trust) <–	• TPS	• Reusability/ Extensible
• Robustness (Rugud)	• ...	• Modular /Component
•		• Low Coupling
		• EDA
		• ACID - transaction
		• Input Validation

5 Communicate



Pre

Engineering vs Tuning

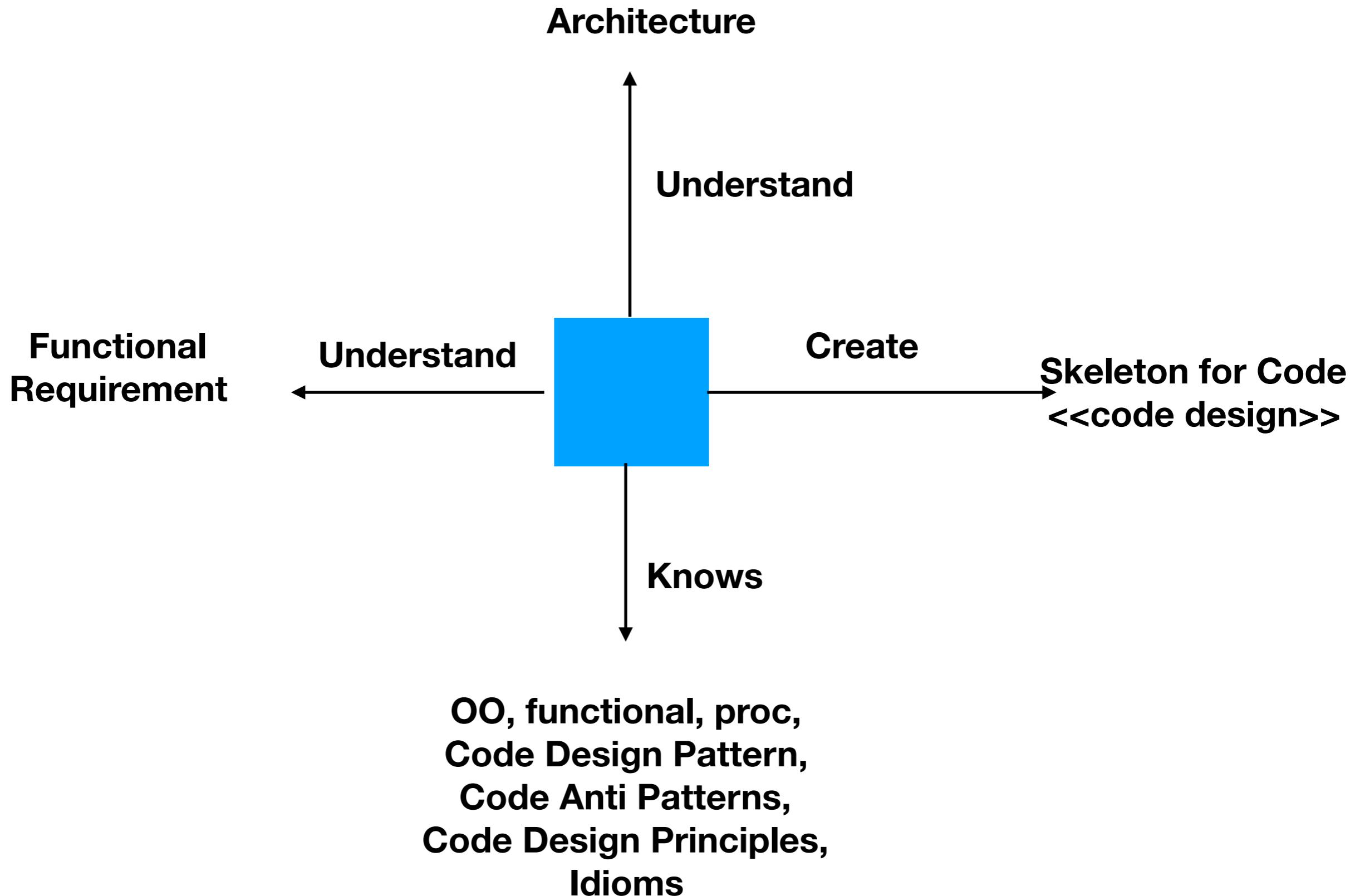
Post

Performance engineering

Performance Tuning

Threat Modeling

Pen Testing/ Ethical Hacking



Architecture vs Design

System quality

Code Maintainability

High Level Design

Blue Print

System Design

Detail Design

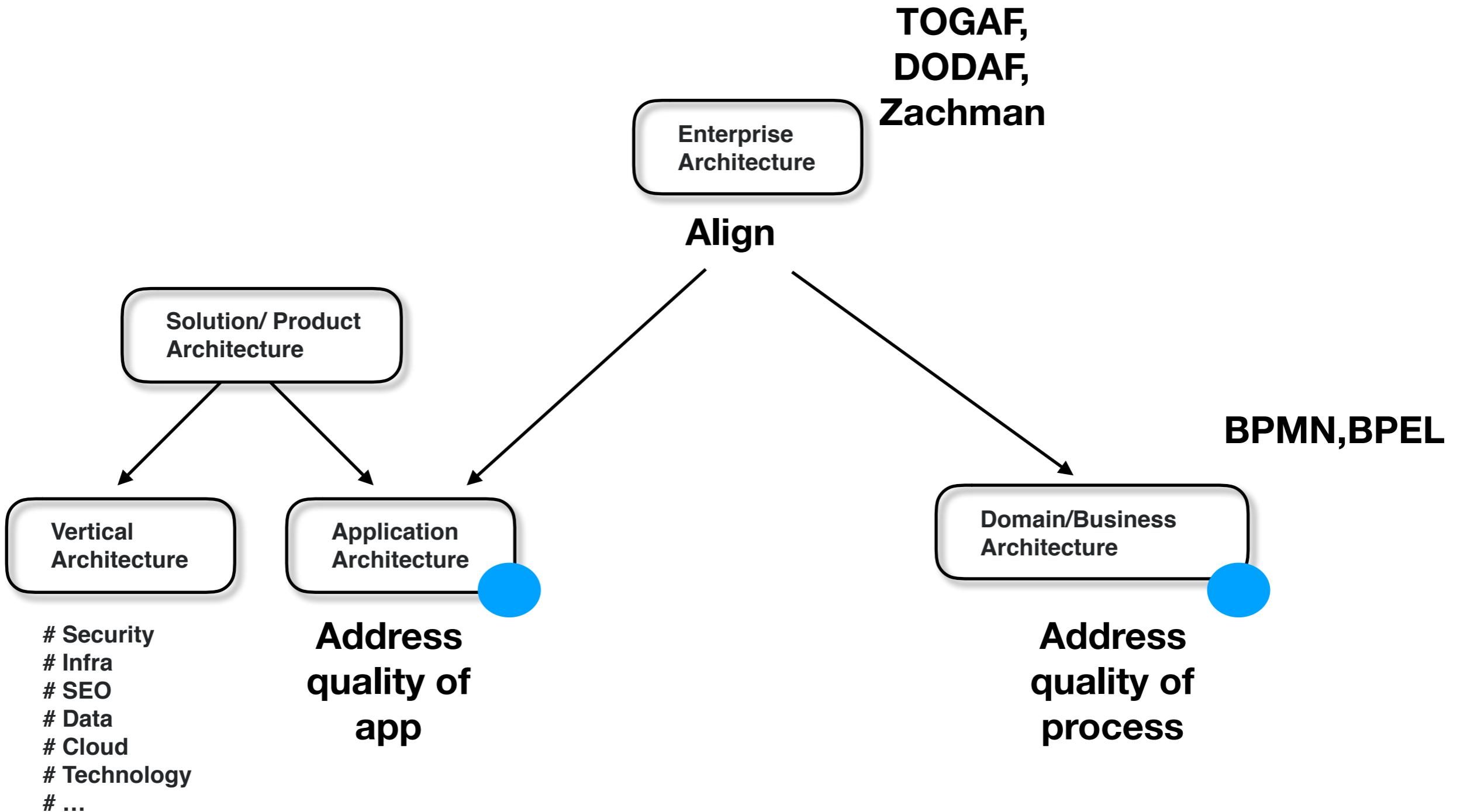
Module Design

Class Design

Low Level Design

Code design

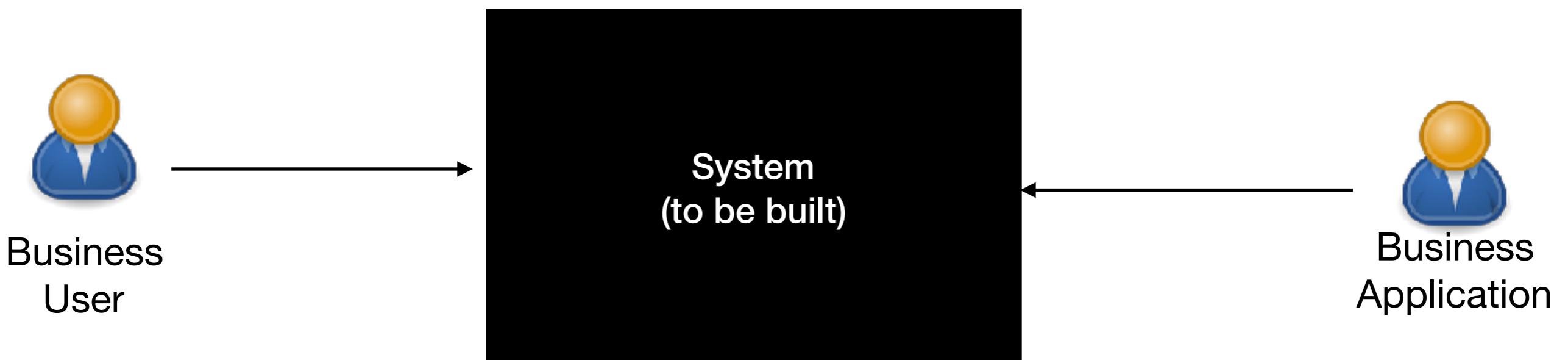
Implementation Design

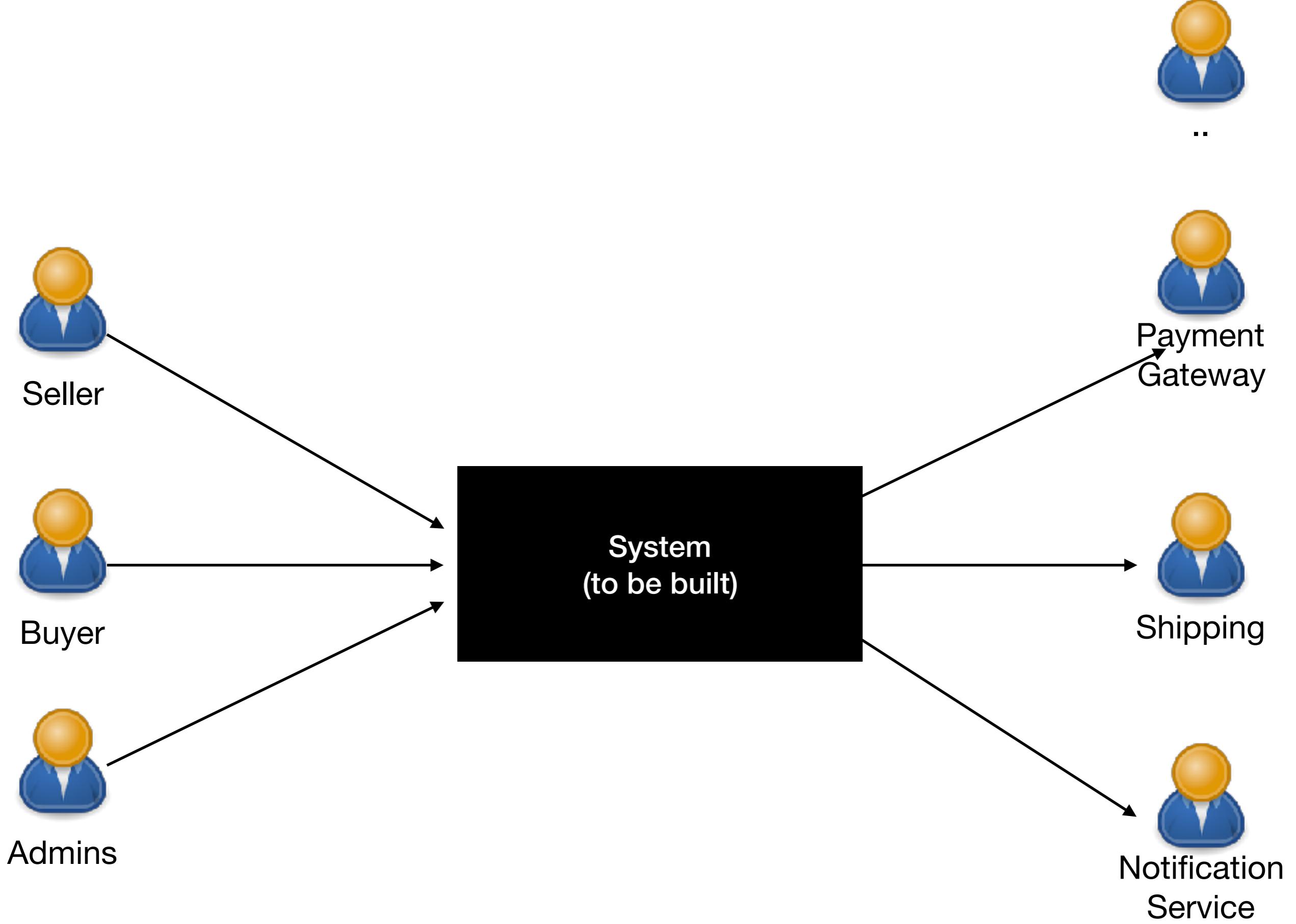


DropBox

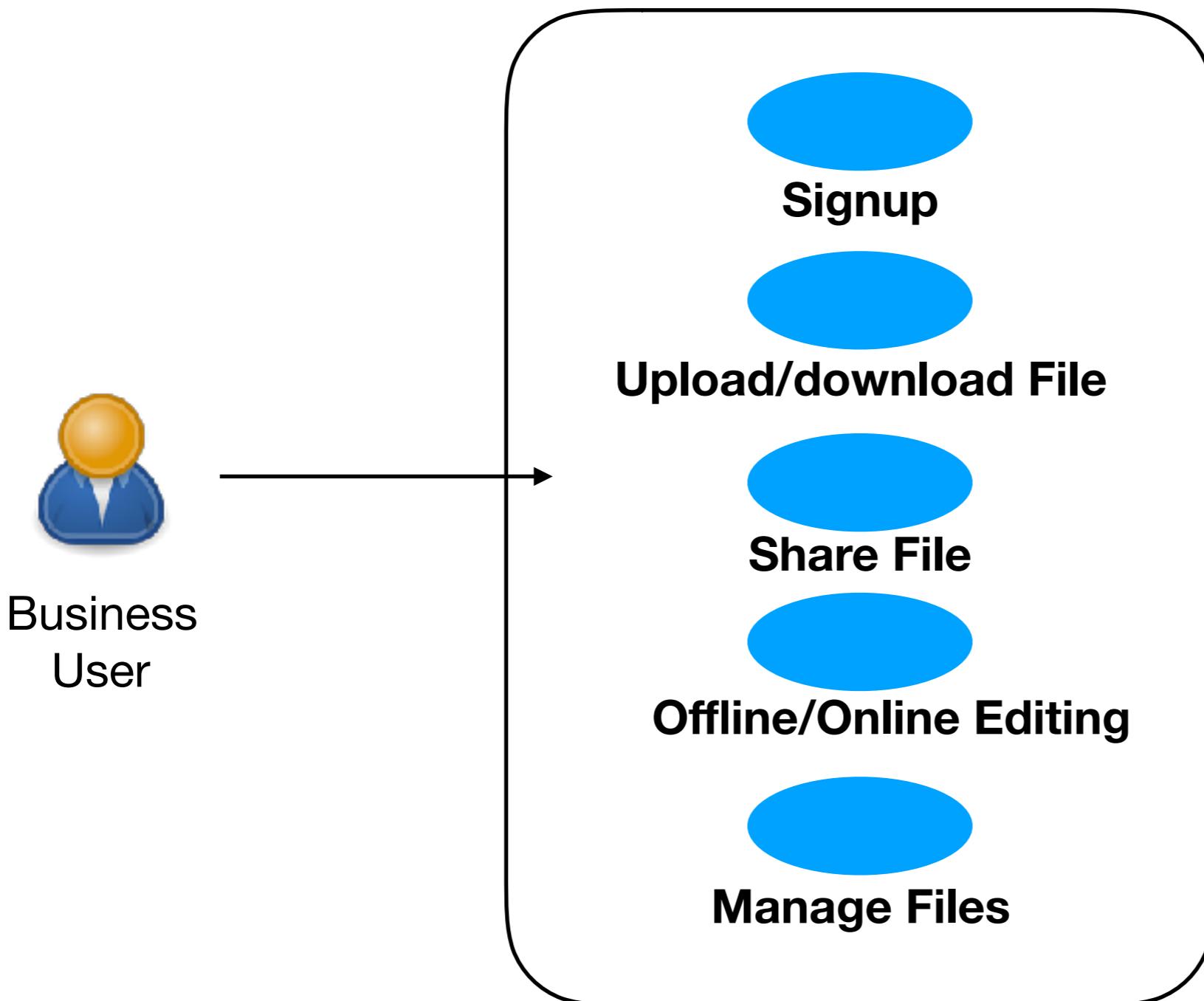
Architectural Requirements

Context View





Functional view

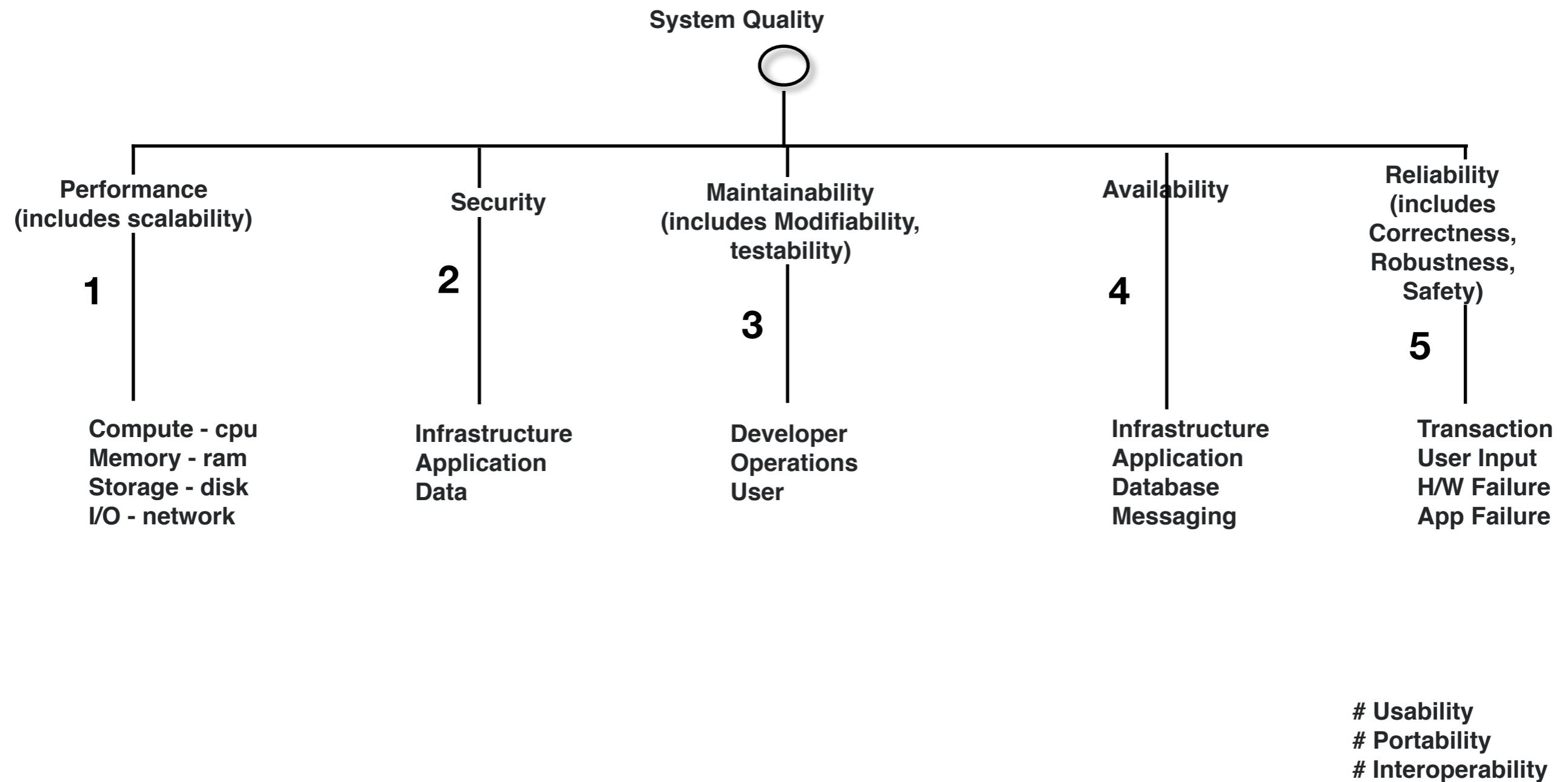


Constraints view

Must support popular Desktop OS and Mobile OS

Should work on a low bandwidth network

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

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
- Collect scenarios
- Pre: prepare lots example scenarios *
- prioritize

DropBox

Architectural Design

Logical View

System

↓
Apply Patterns

App 1

↓
Apply Patterns

M1

M2

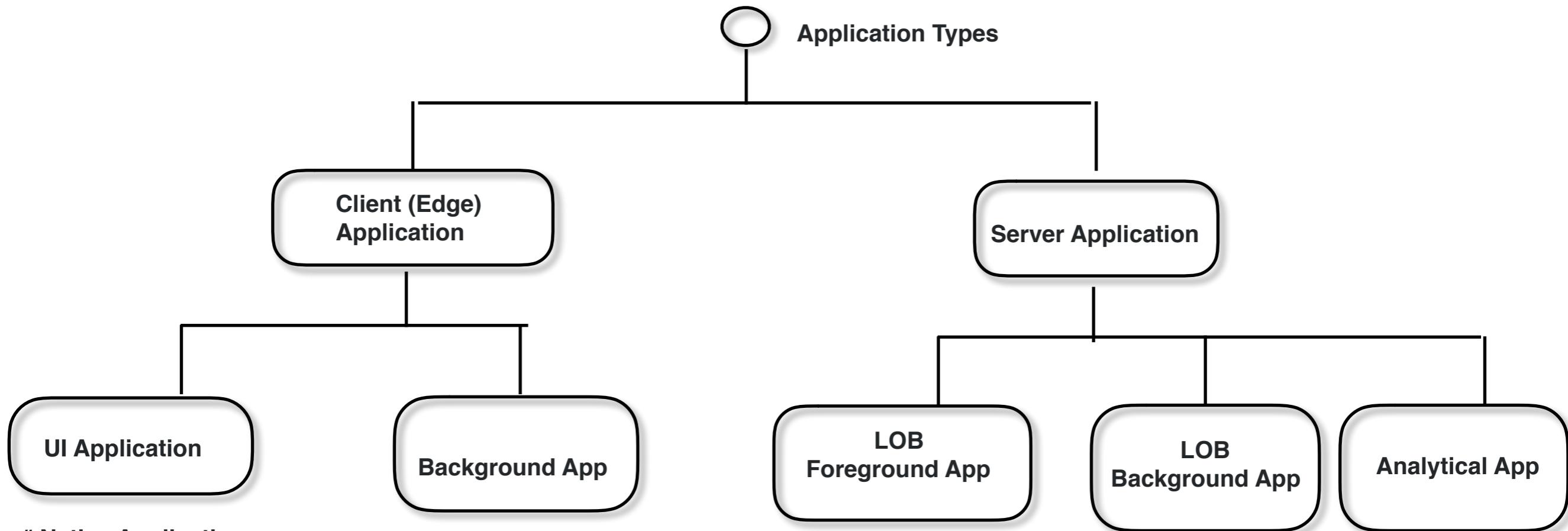
App 2

↓
App 2.1 App 2.2 App 2.3

↓
Apply Patterns

M3 M4

Decompose the system into Applications



Native Application

- * WPF
- * Swing
- * Android

Browser - Native Application

- * Flash
- * ActiveX
- * Applet

Browser - Server Pages

- * JSP
- * PHP

Browser - SPA

- * Angular
- * React

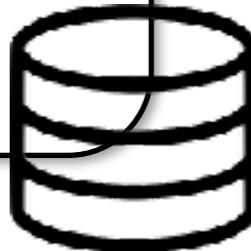
* Windows Service * Daemon

* API Application

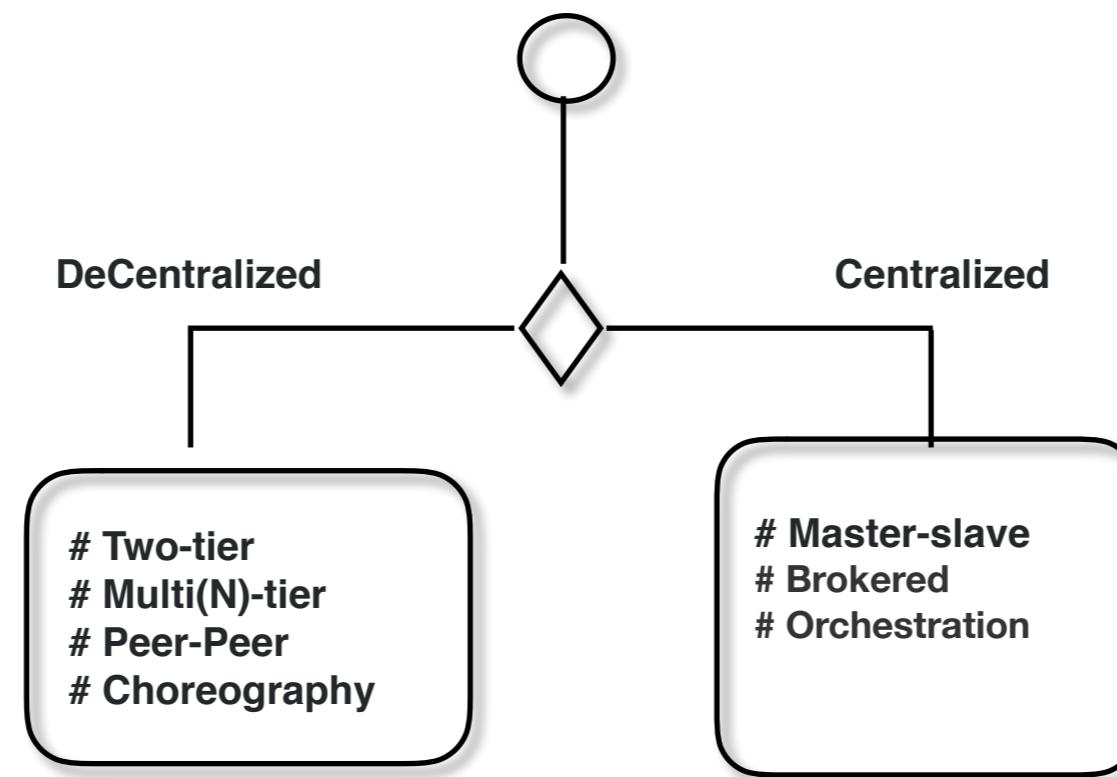
- # Event Based Application
- # Scheduled Application
- * Windows Service
- * Daemon

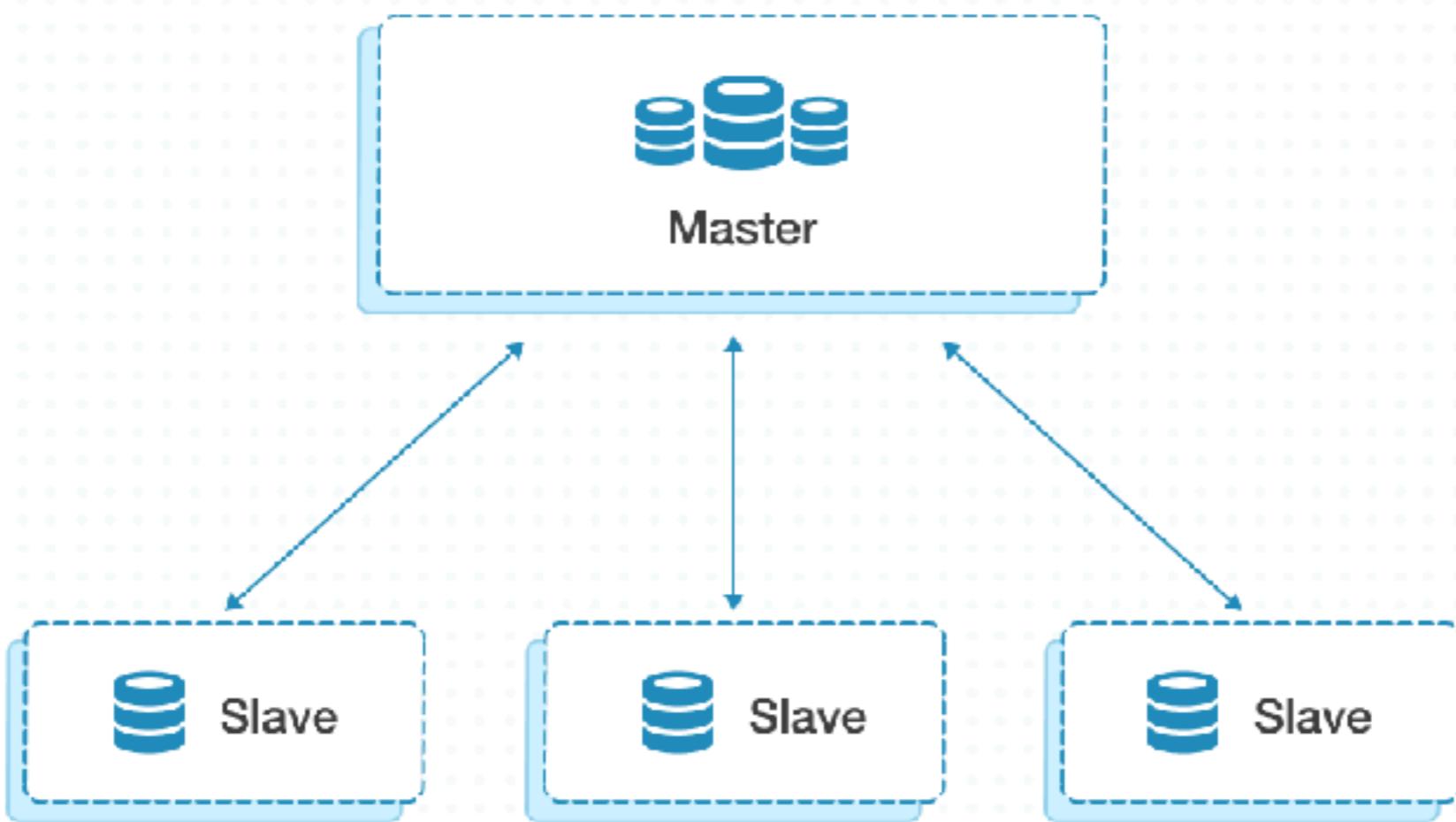
ToDo Portal Application

ToDo API Application

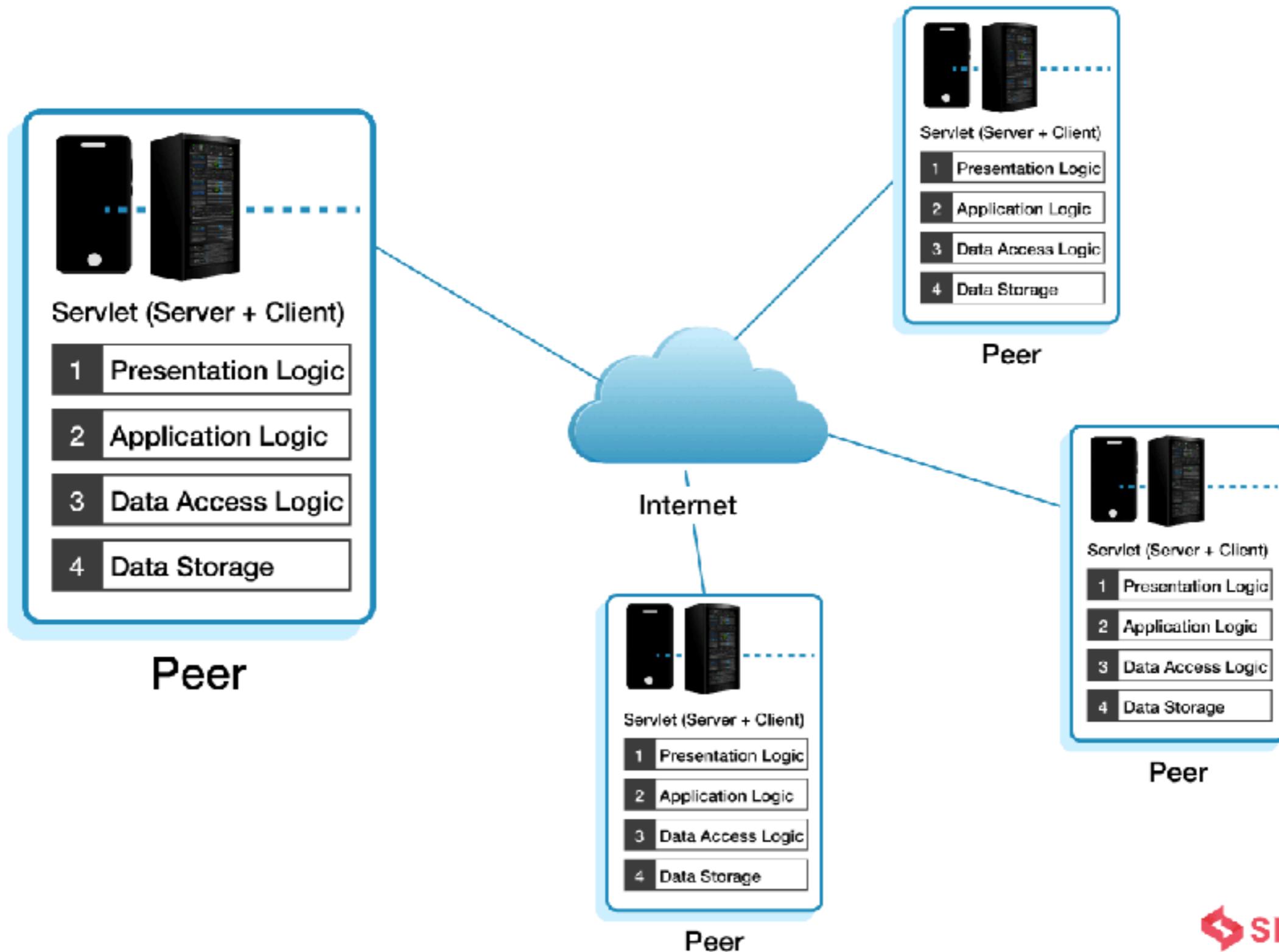


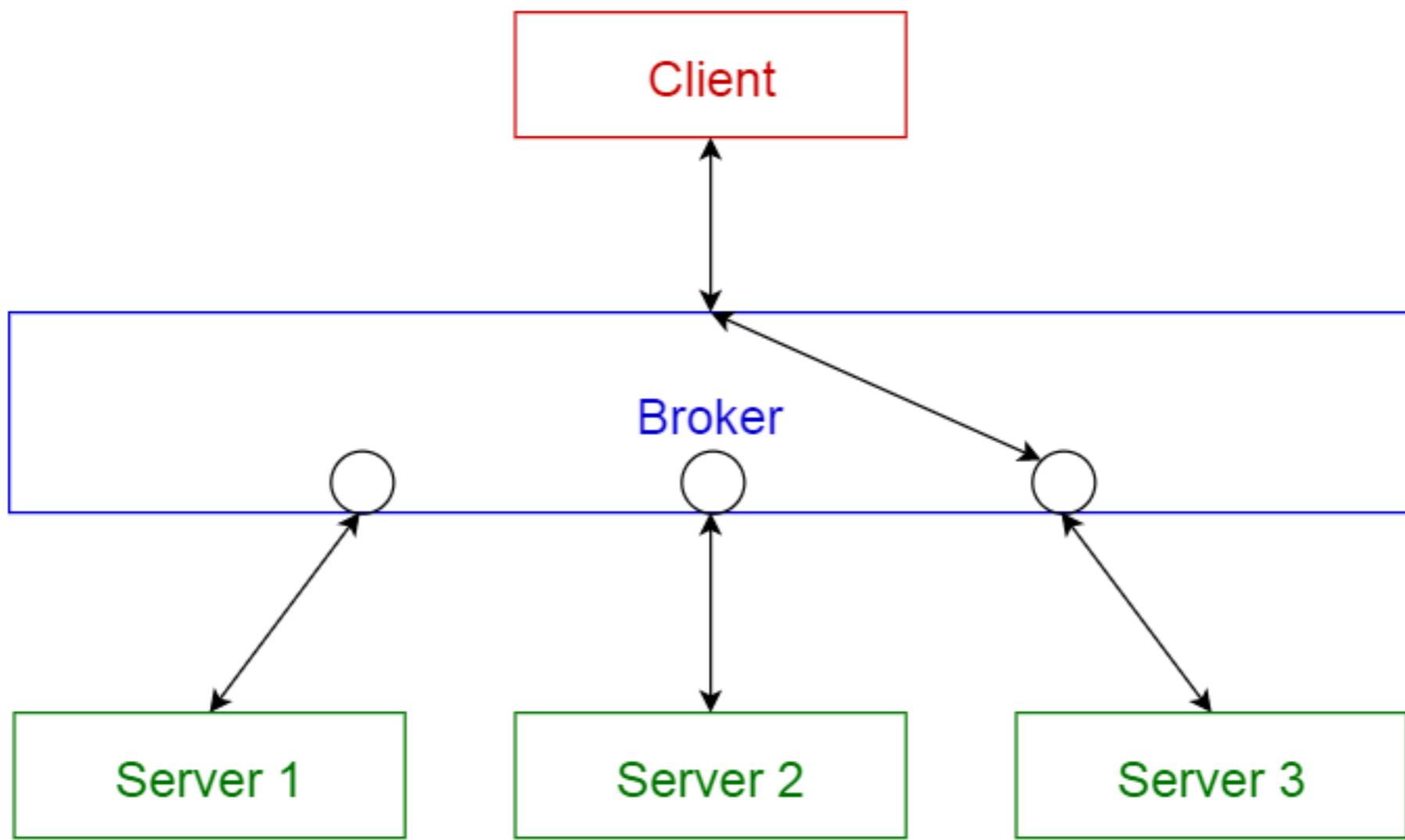
Distributed Communication Patterns

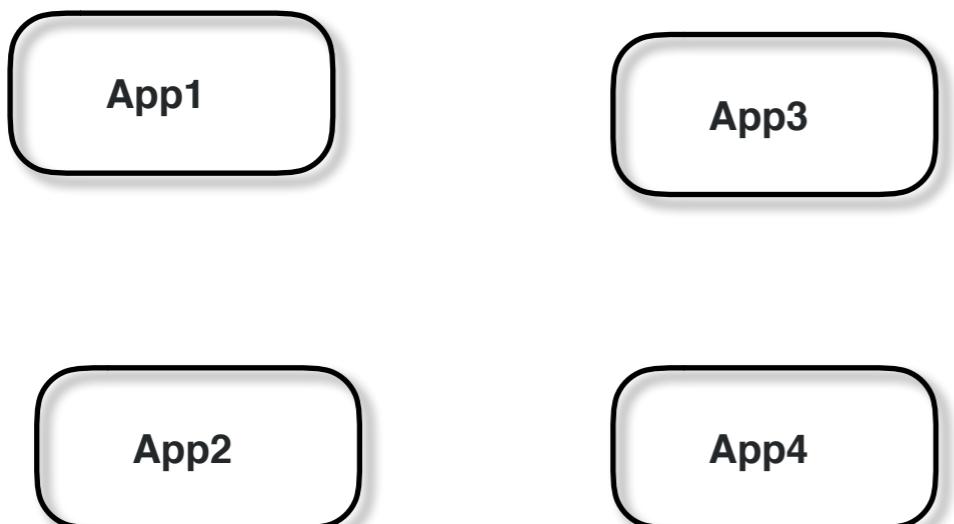
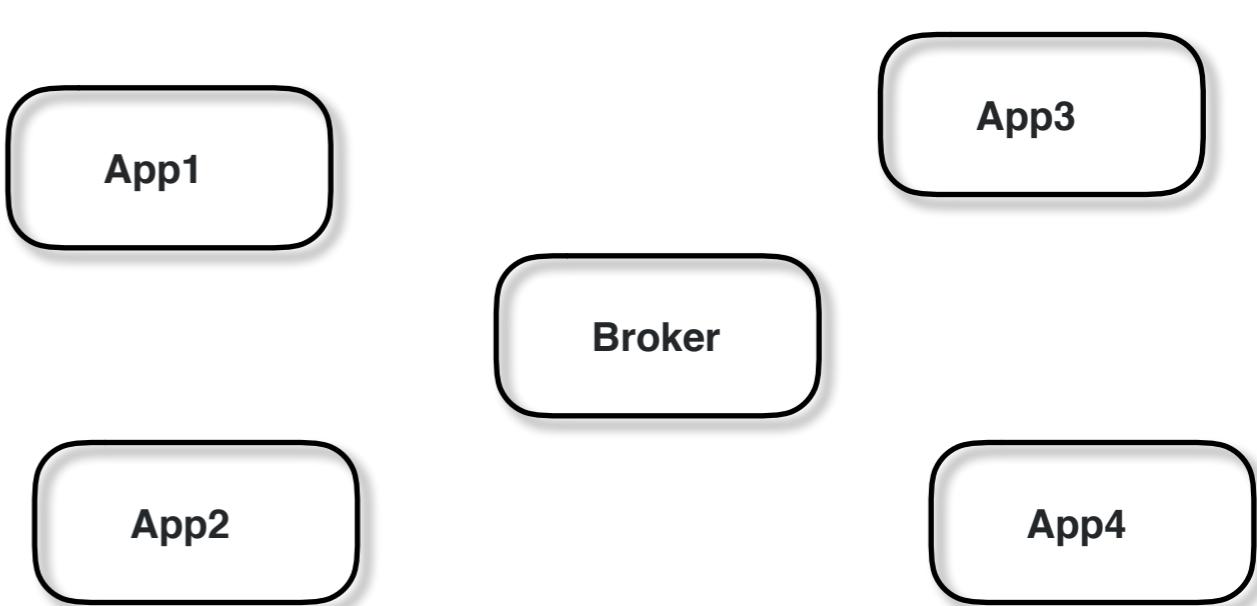
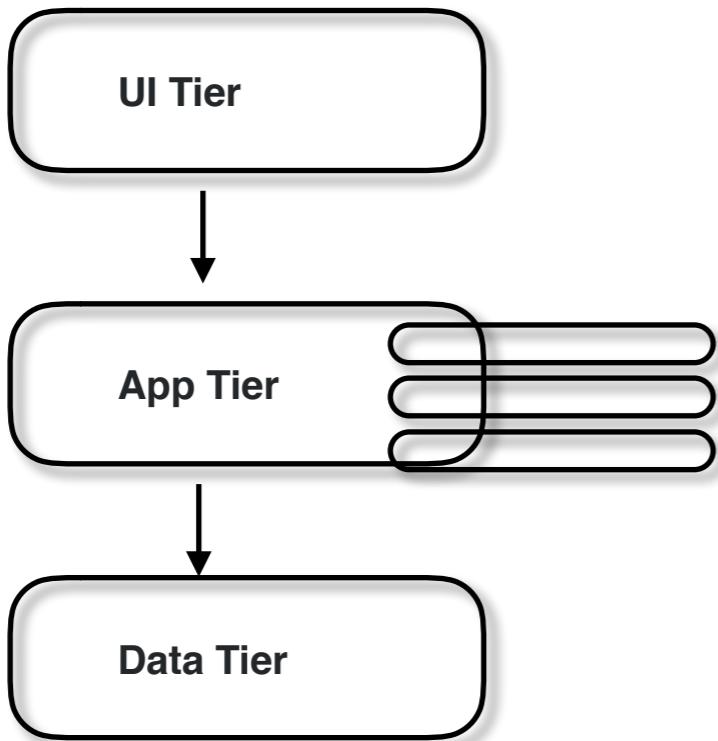




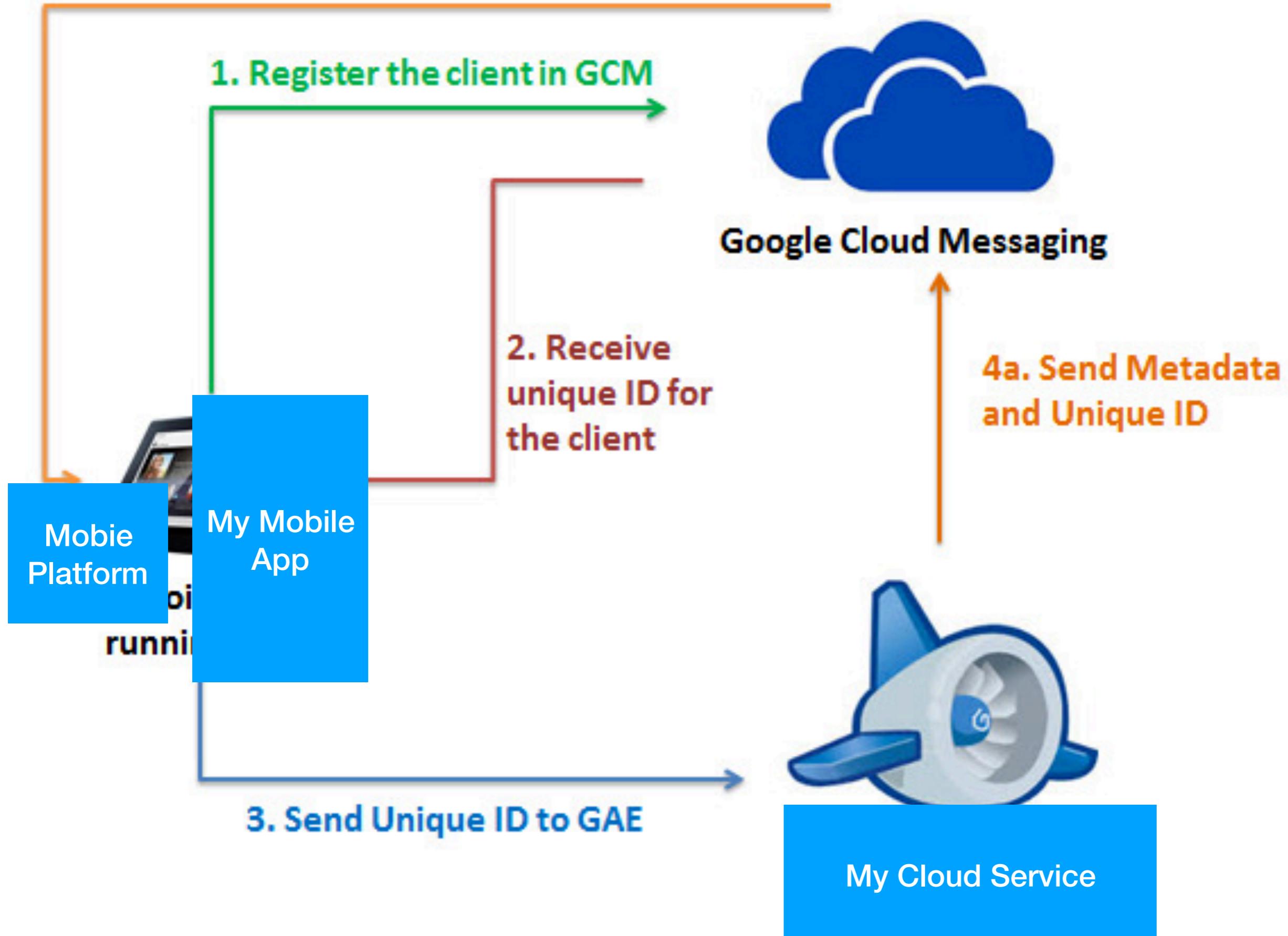
Peer-to-Peer architecture

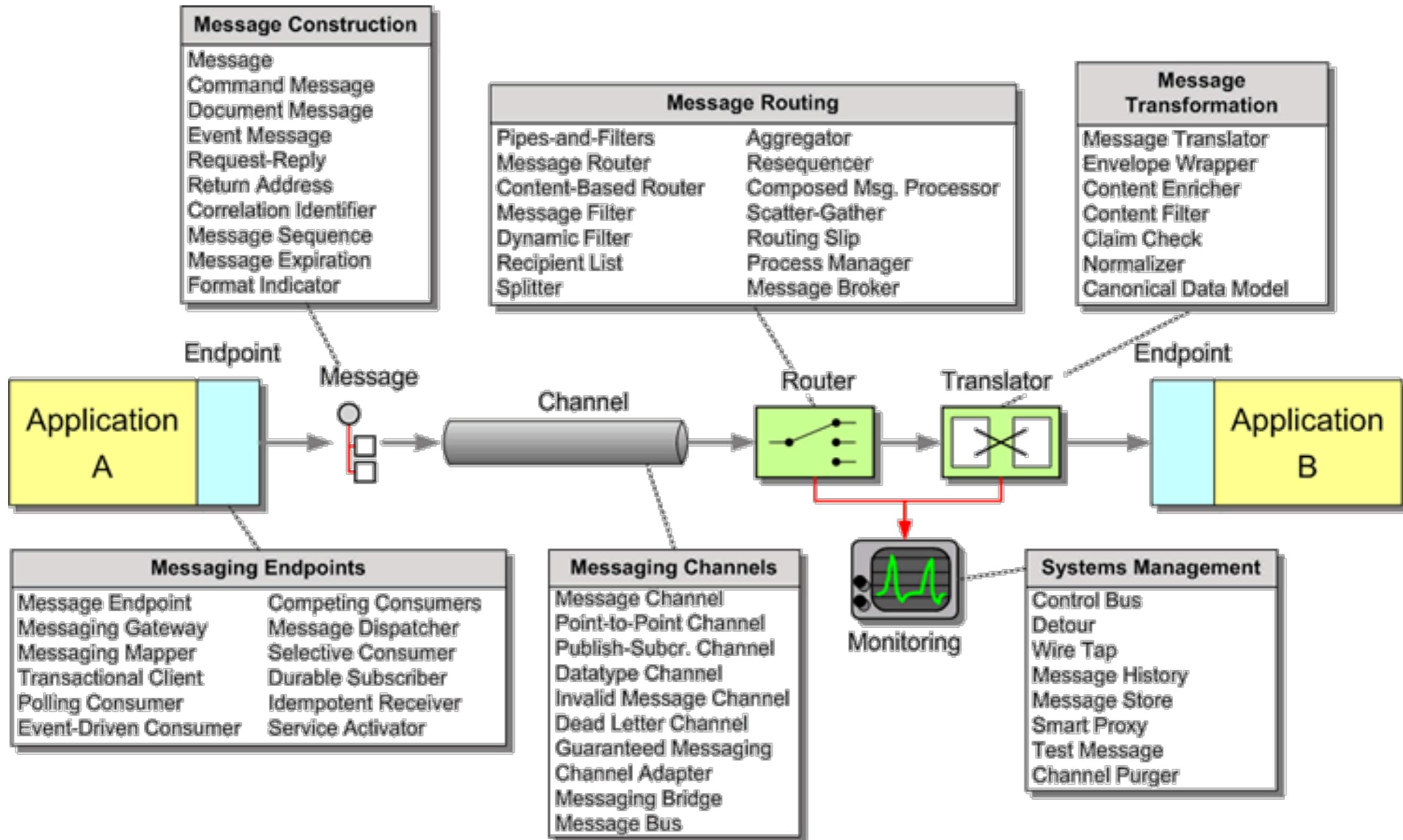


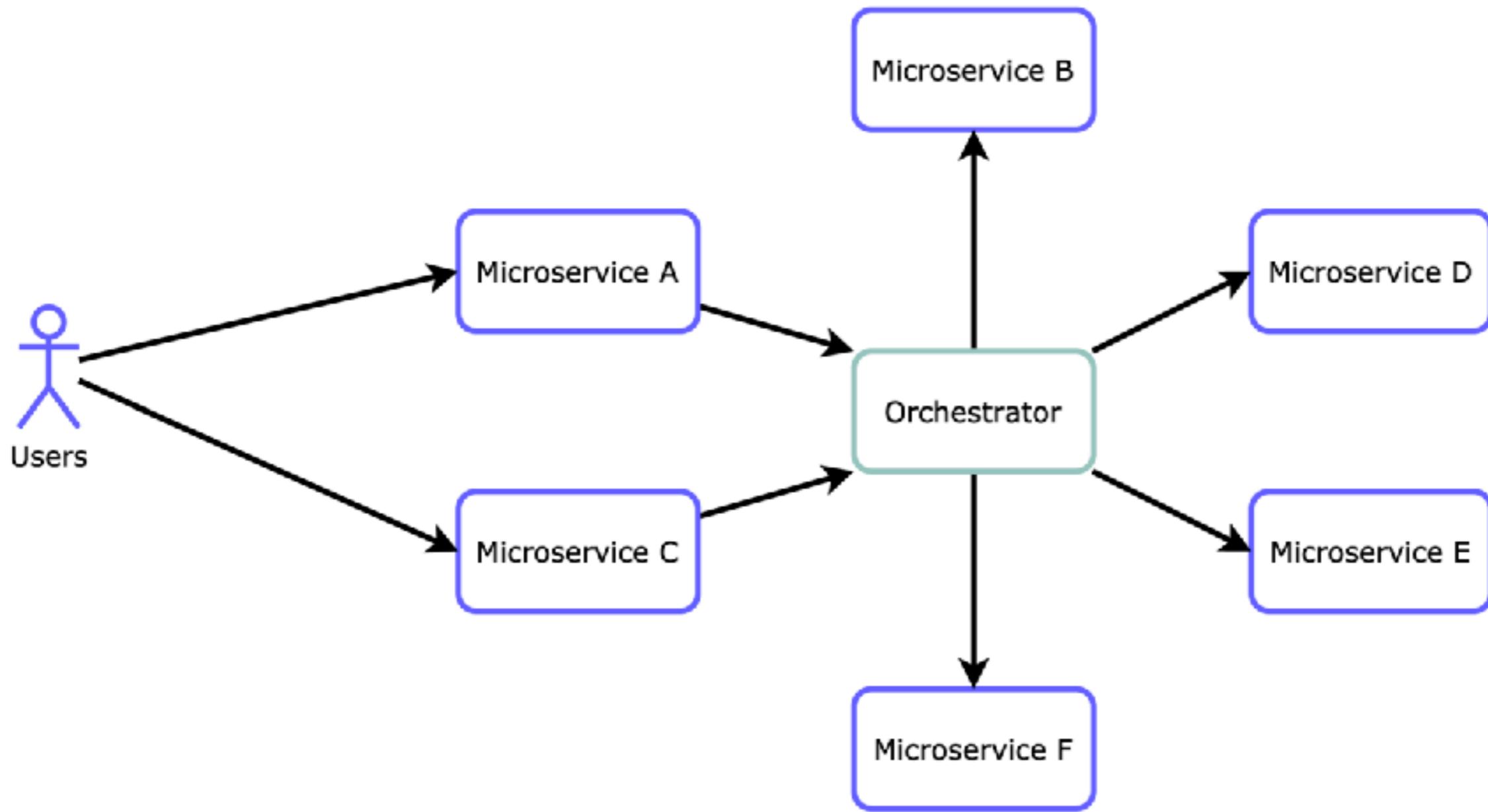


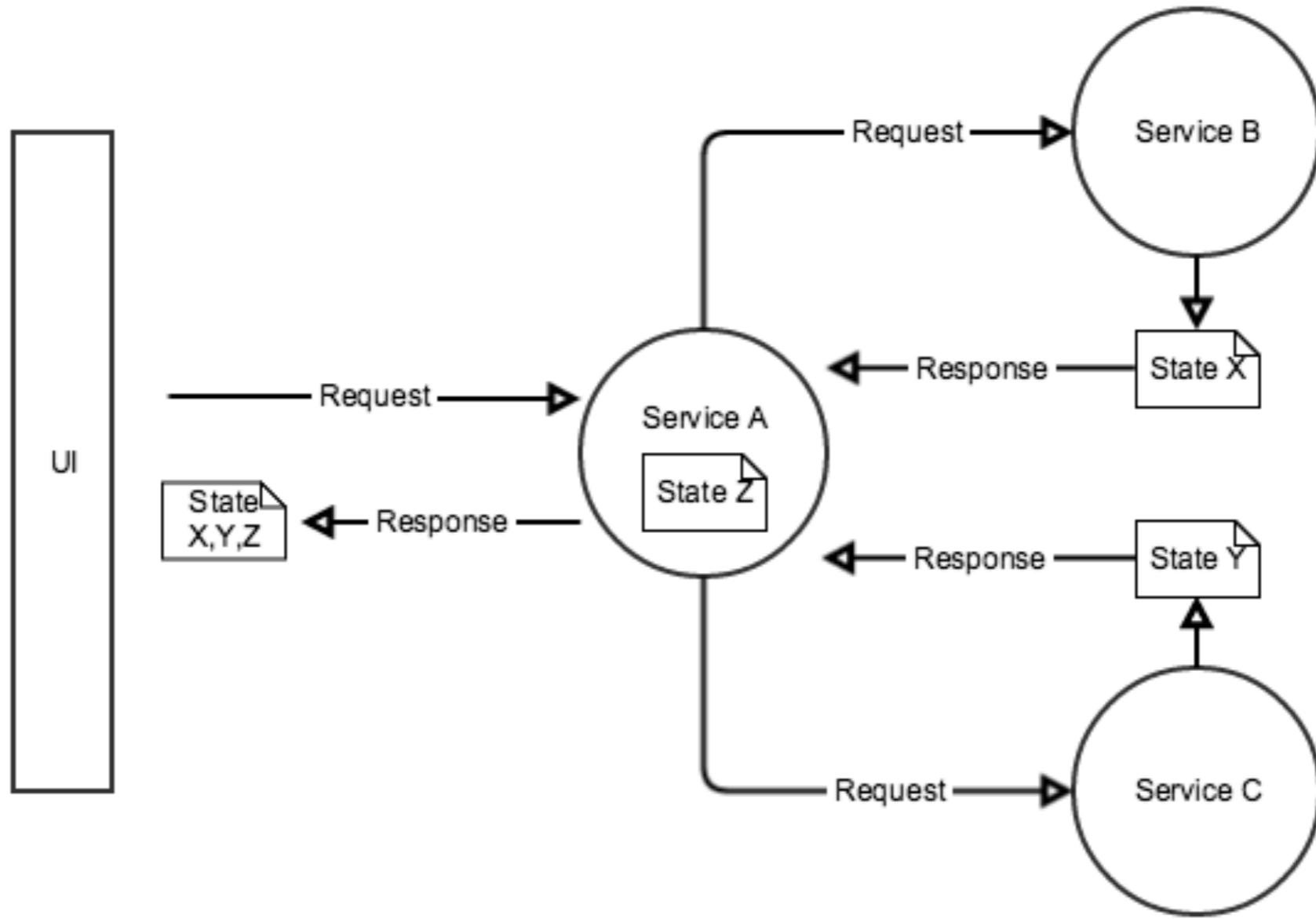


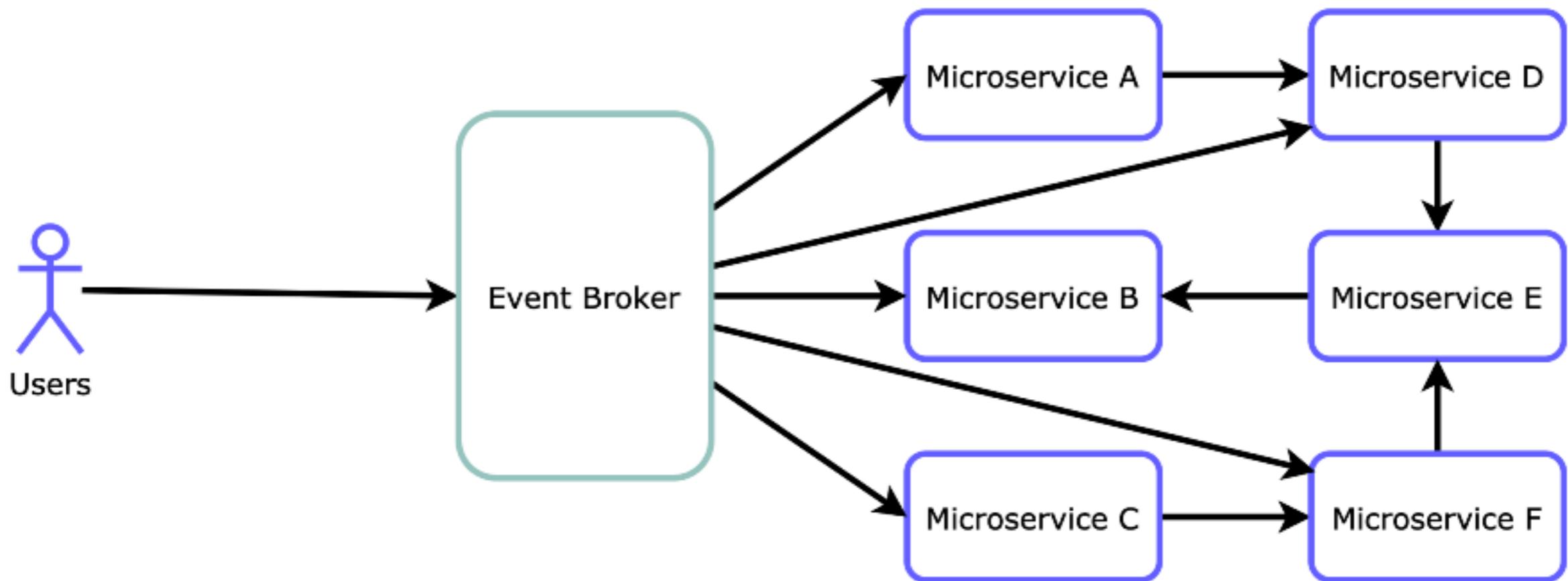
4b. Send Metadata to the client having the Unique ID



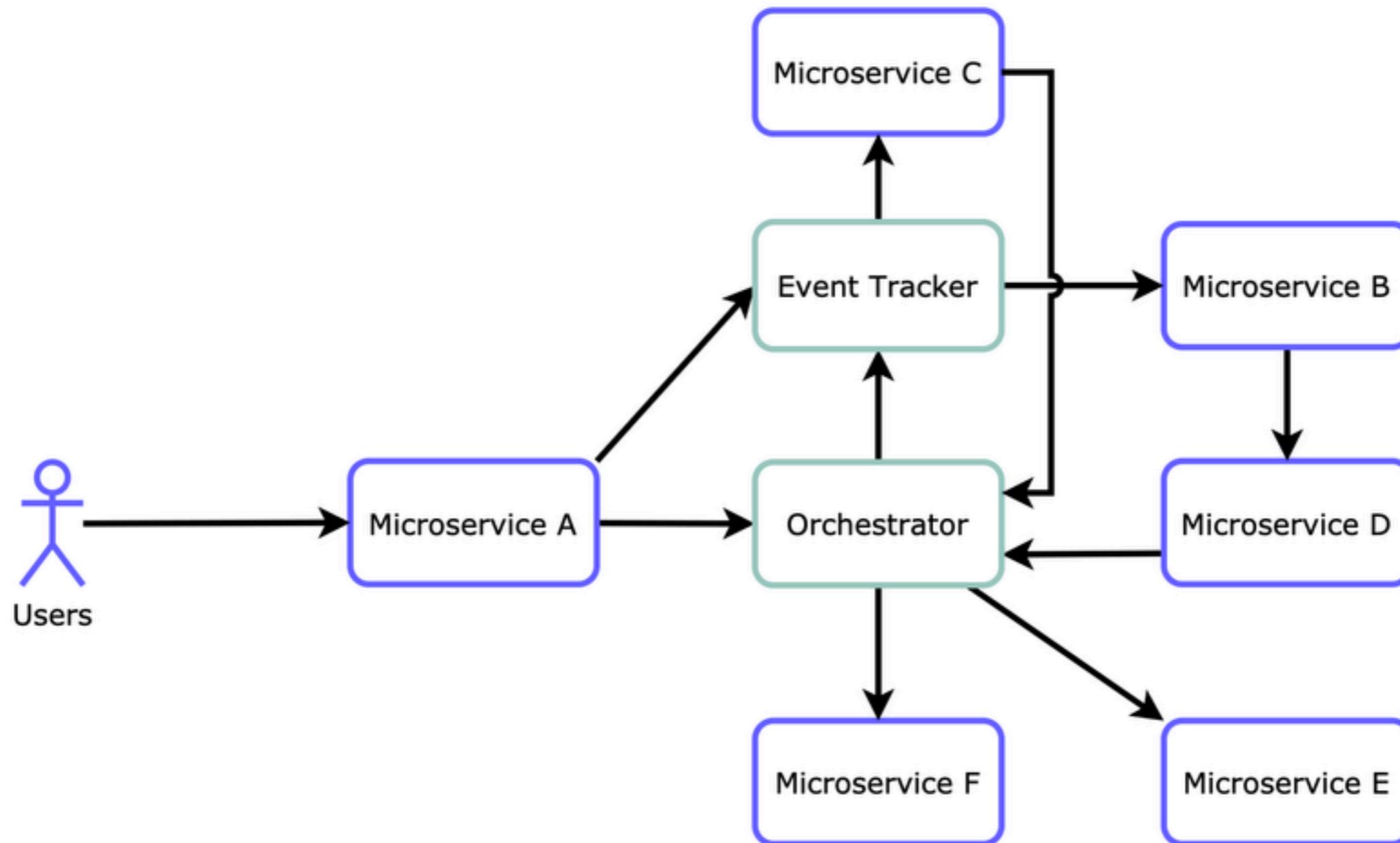


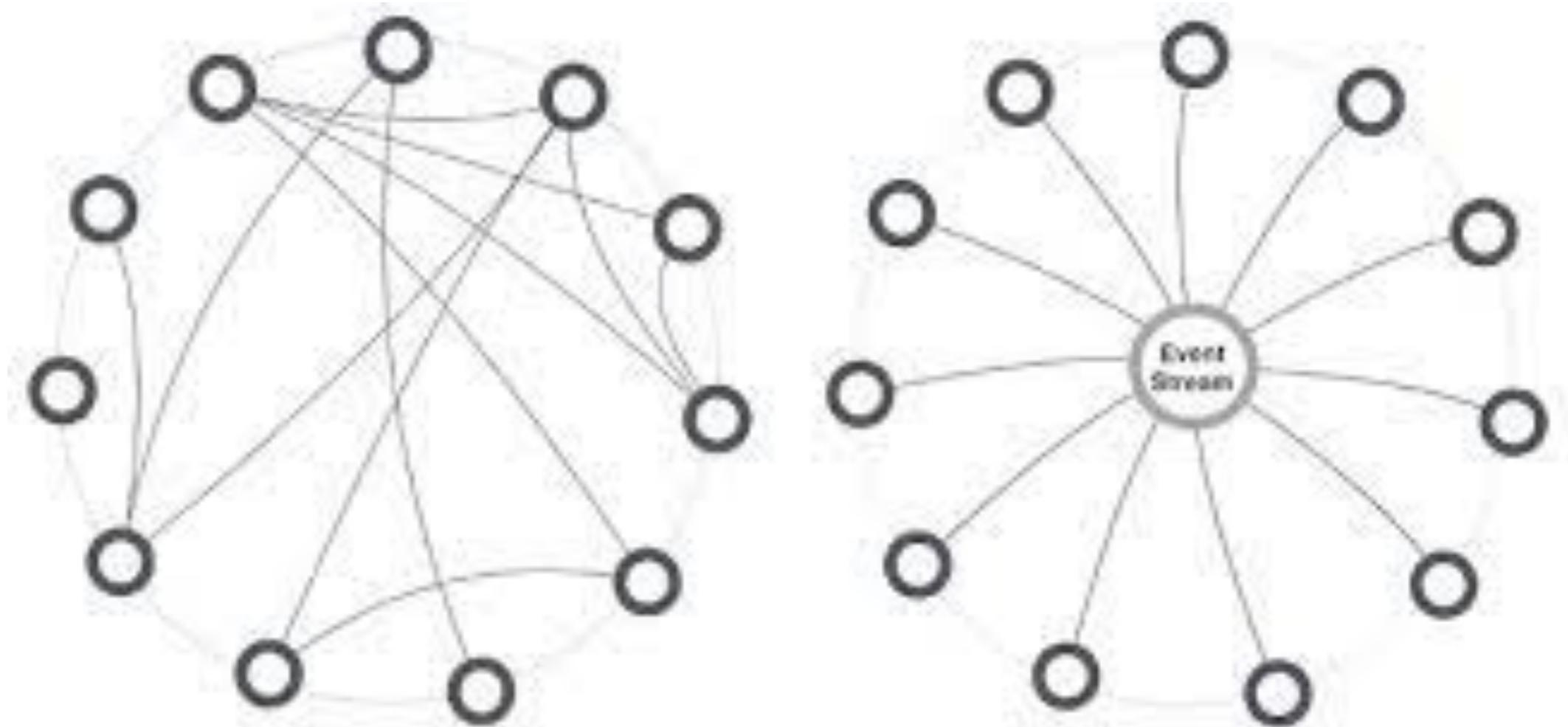




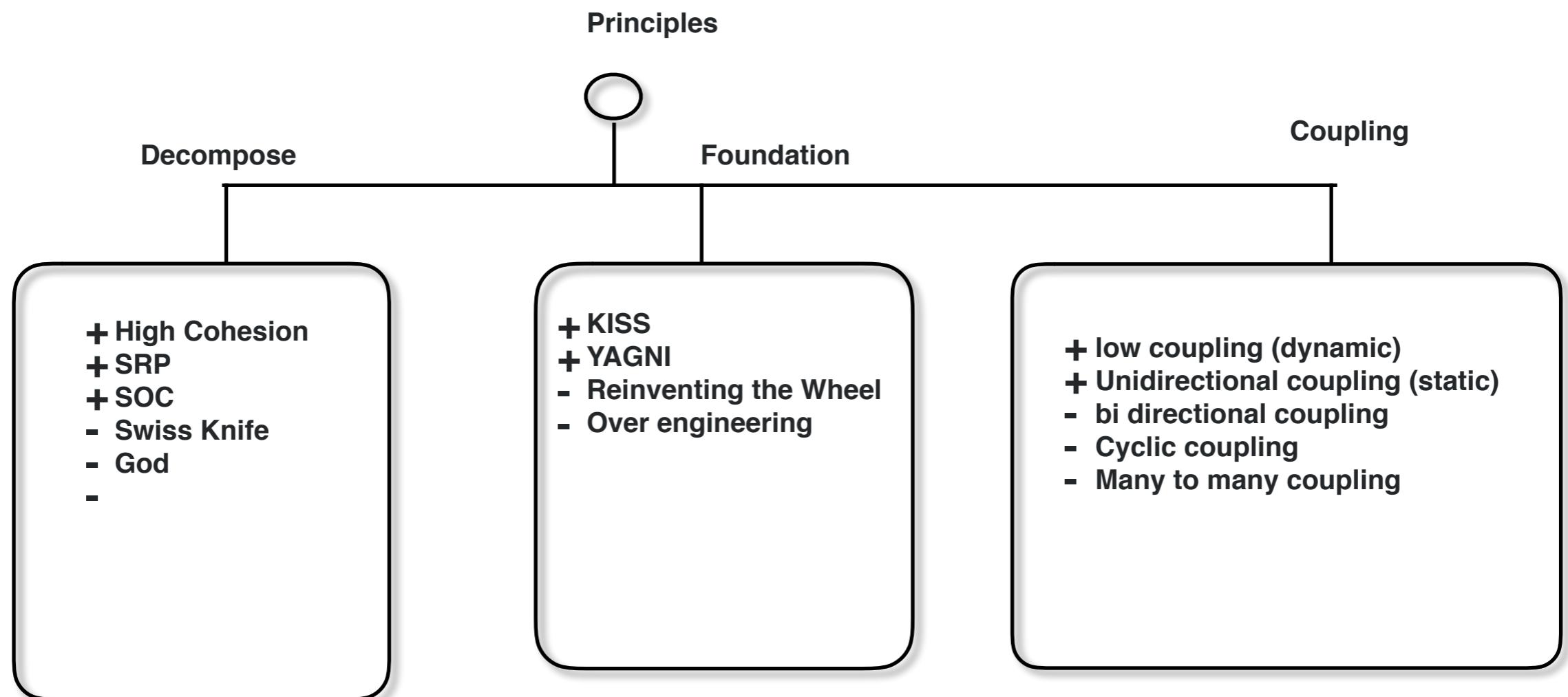


Hybrid

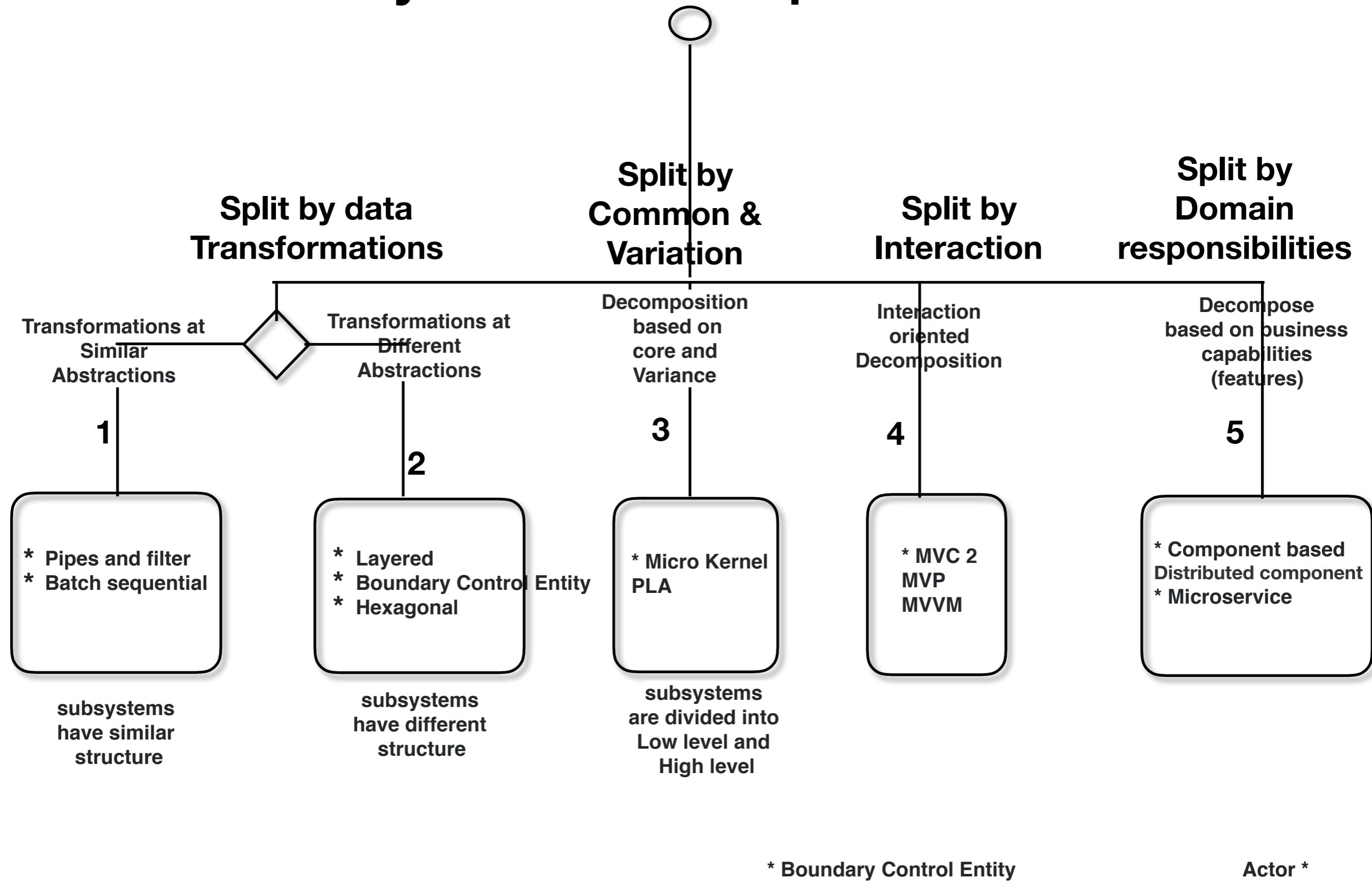




Apply Architecture Principles



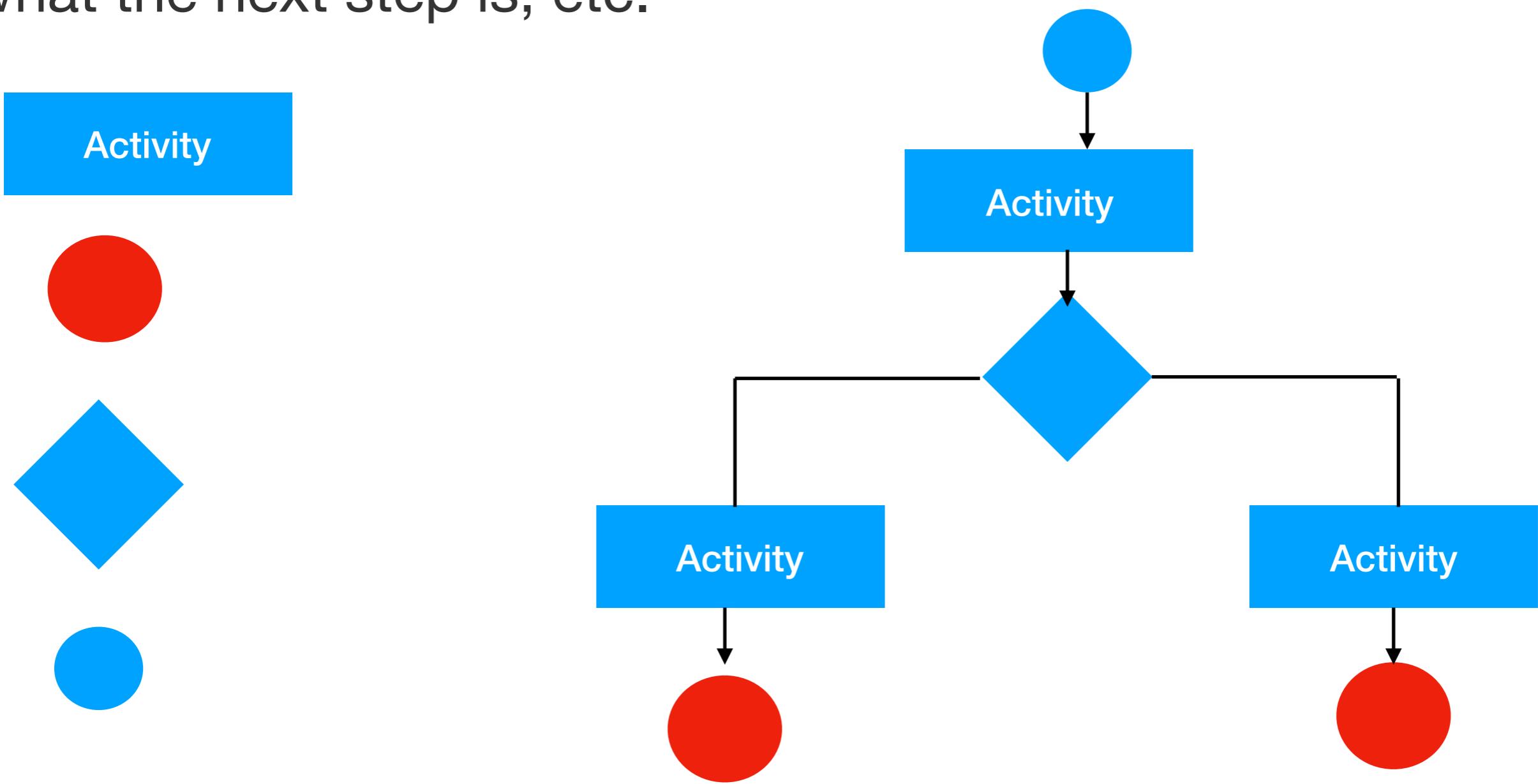
Choose System Decomposition Patterns



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.



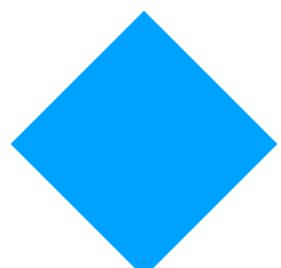
A task scheduler. A scheduler contains all the logic for scheduling and triggering tasks

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

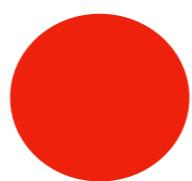
Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is decomposed into more manageable parts.

Core

Pipes and filter



Activity holder



Activity1

Activity2

Activity3

Presentation Layer

Component

Component

Component

Business Layer

Component

Component

Component

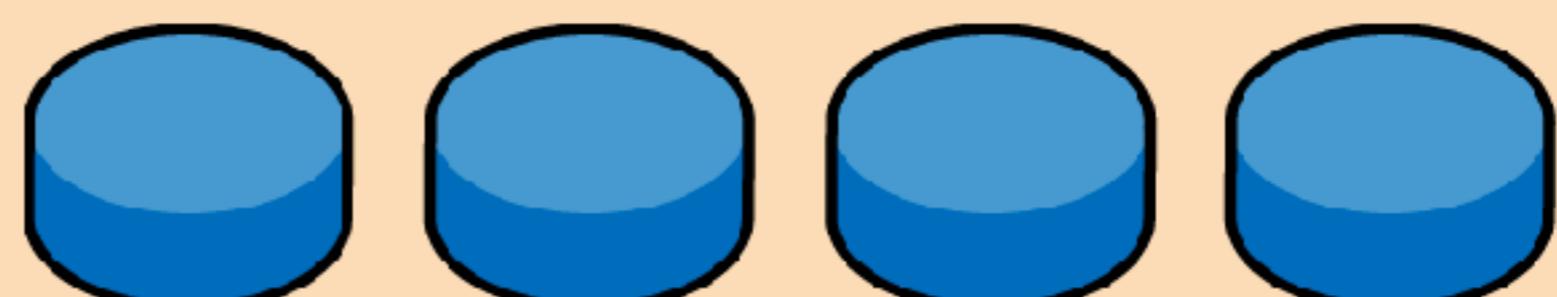
Persistence Layer

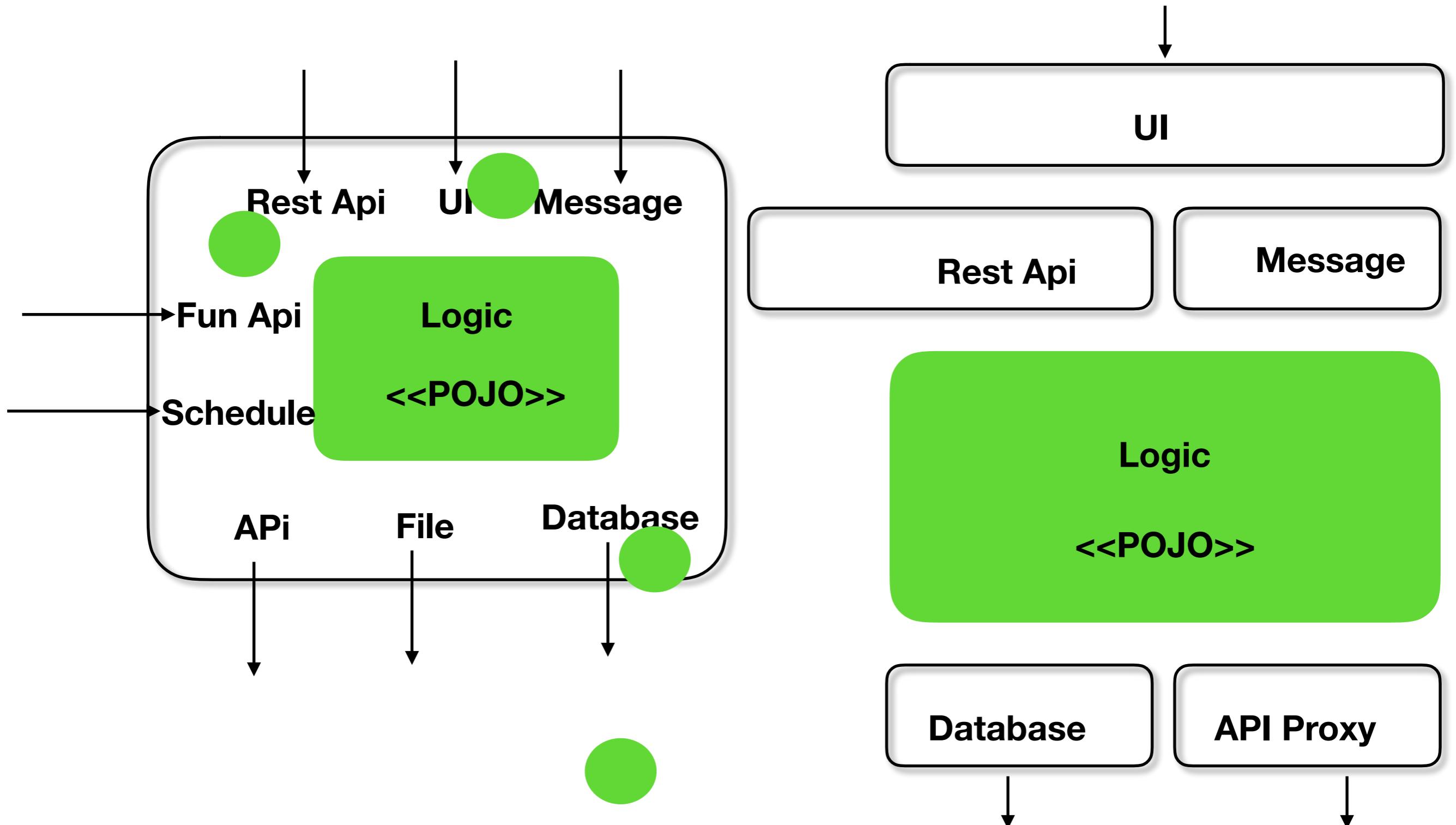
Component

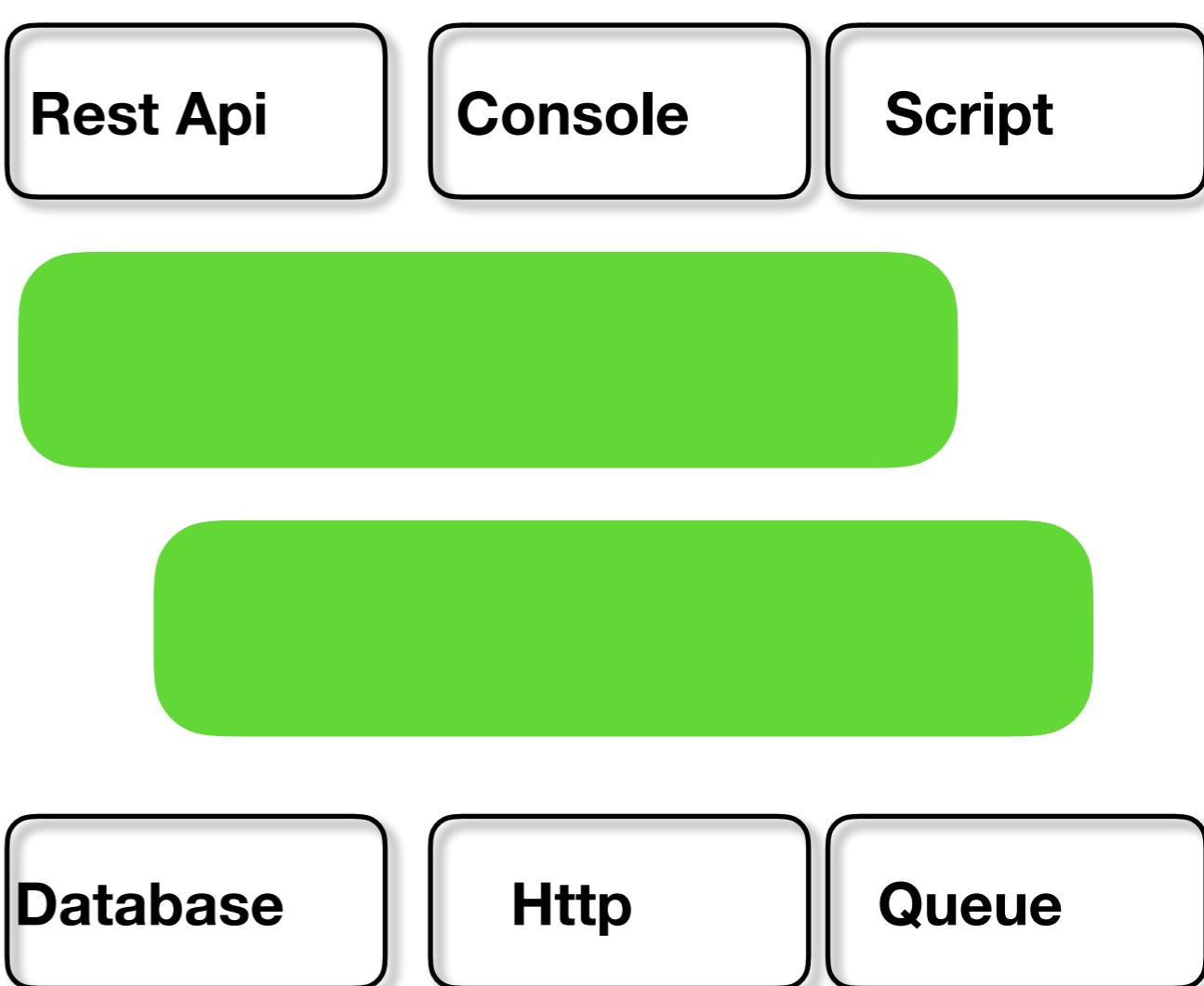
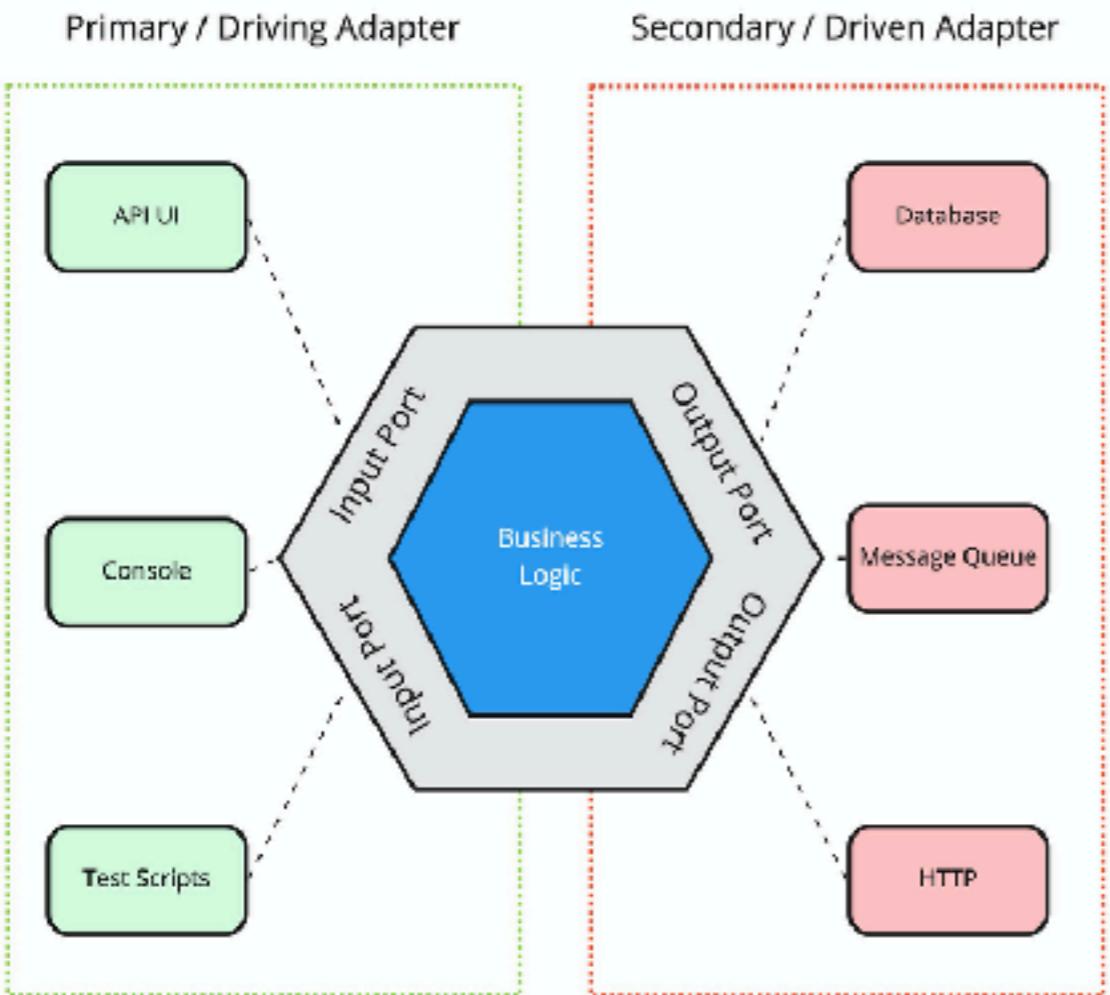
Component

Component

Database Layer







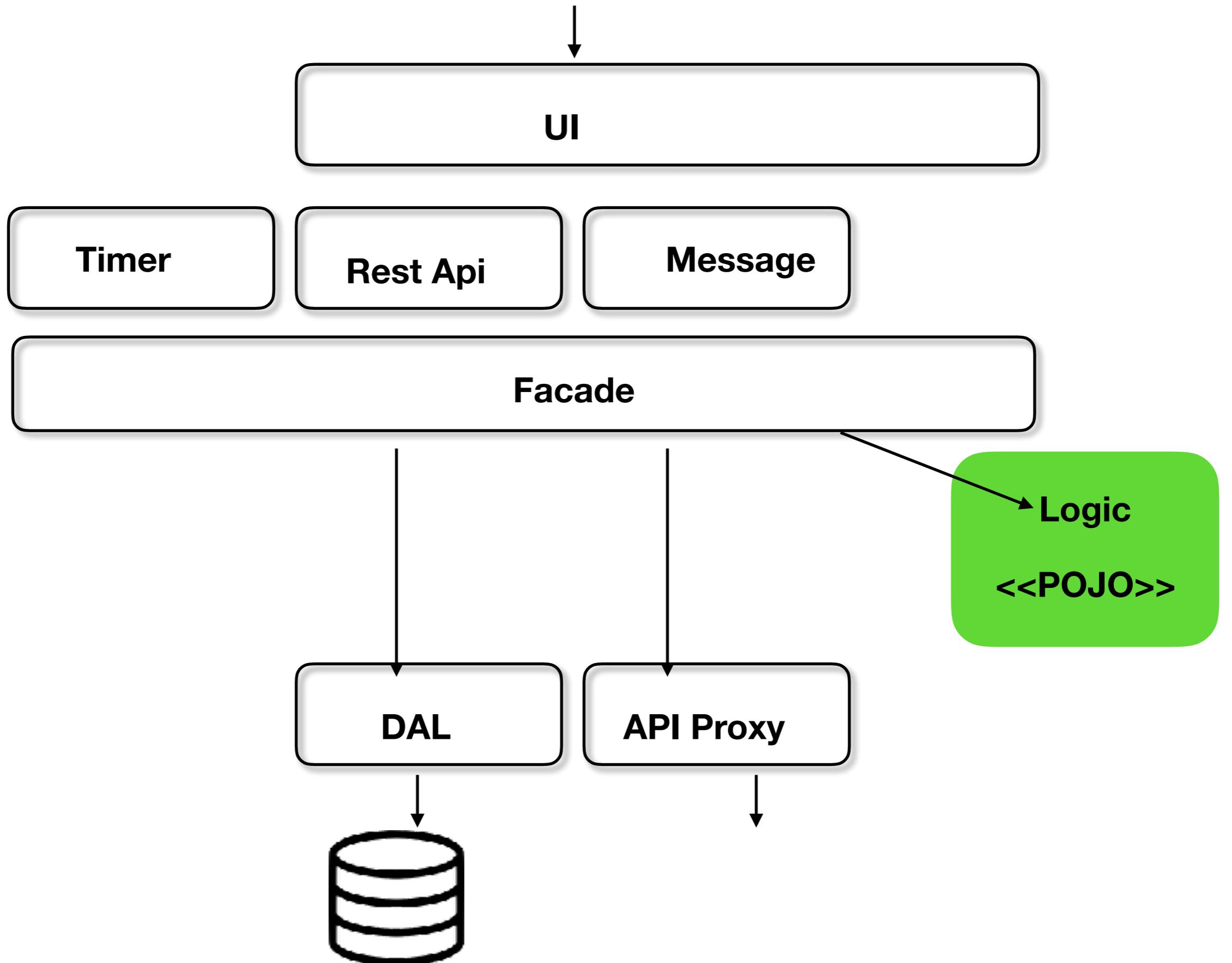
View

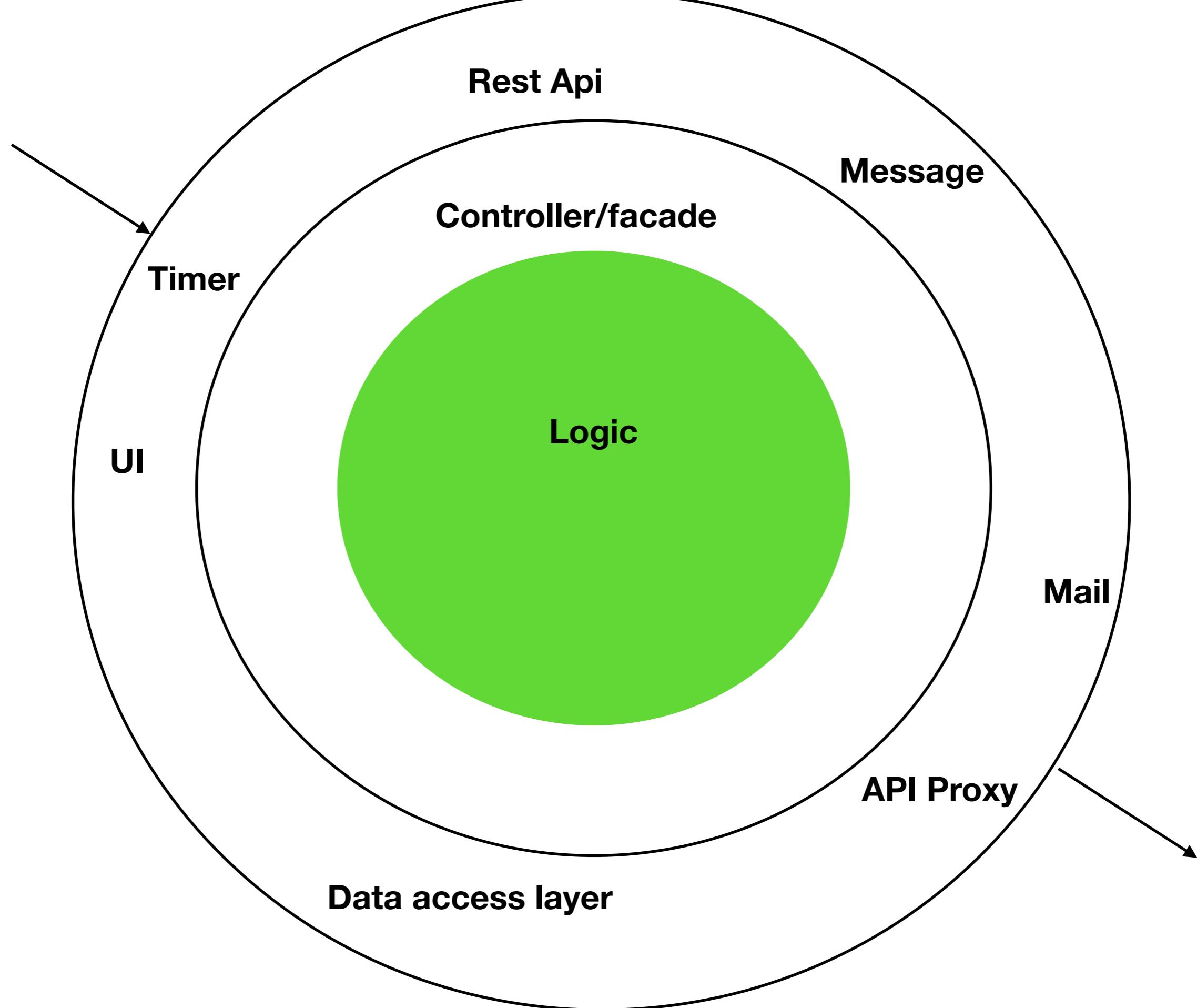
Control

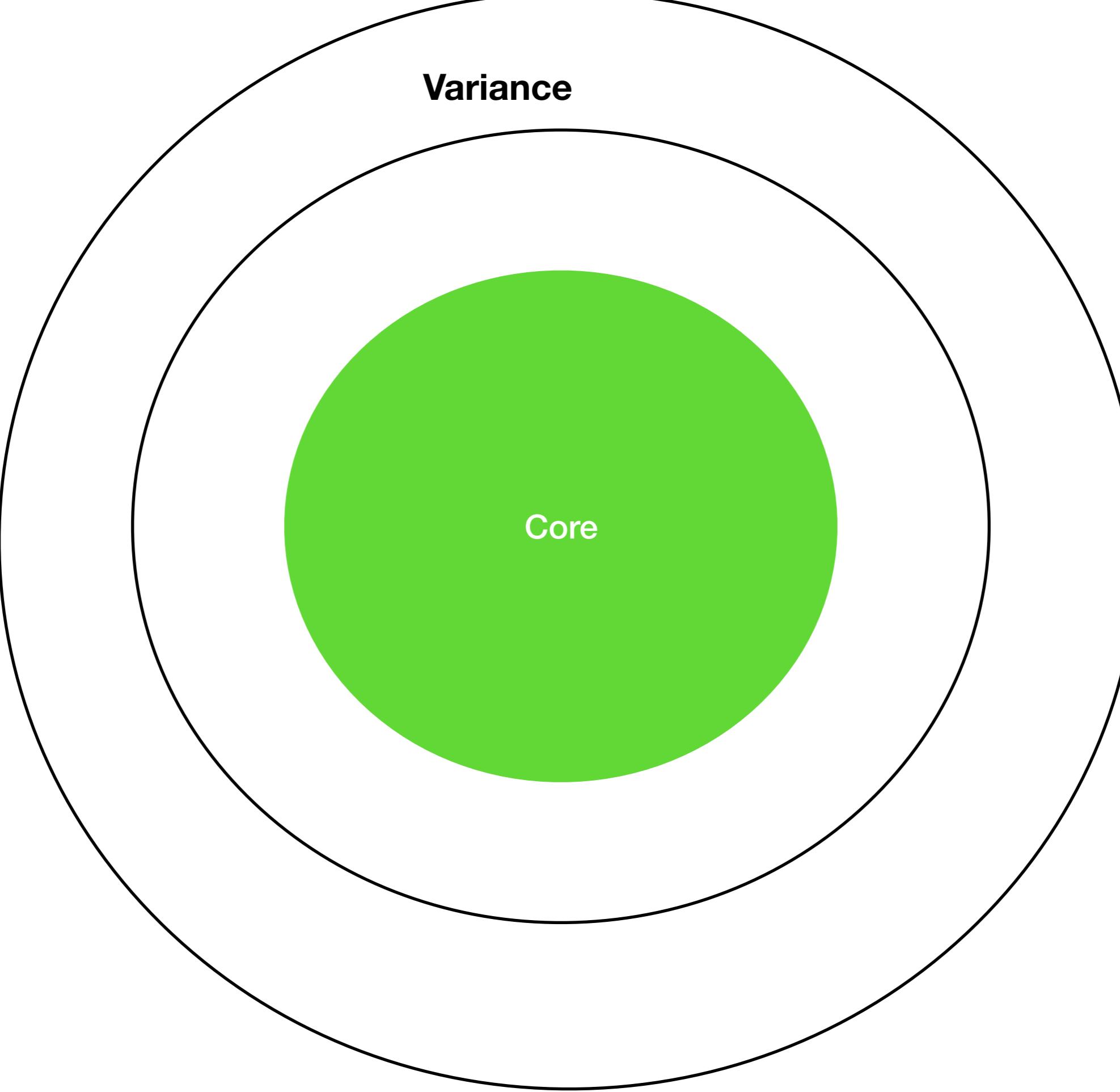
Model

UI Layer

Api

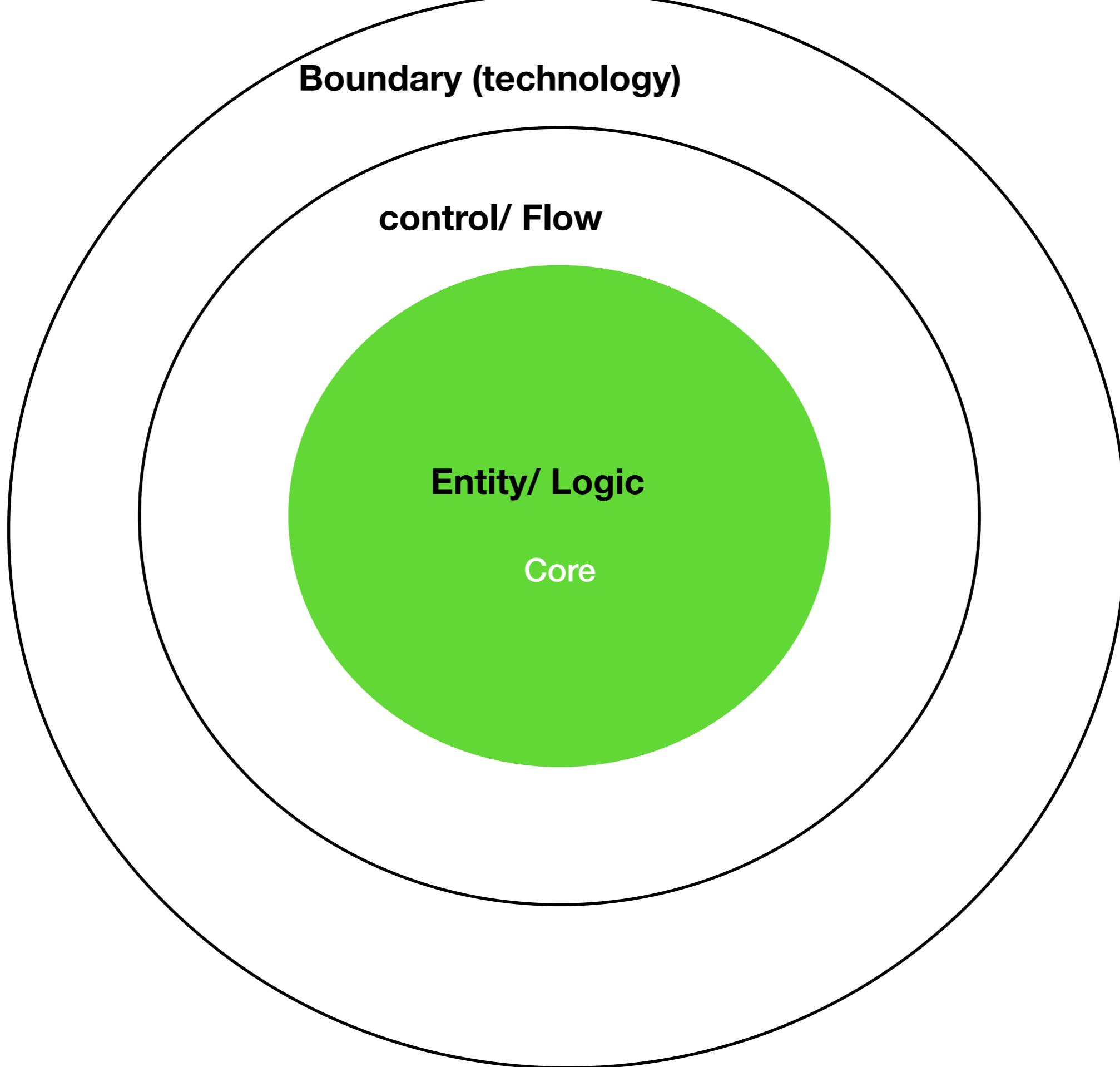


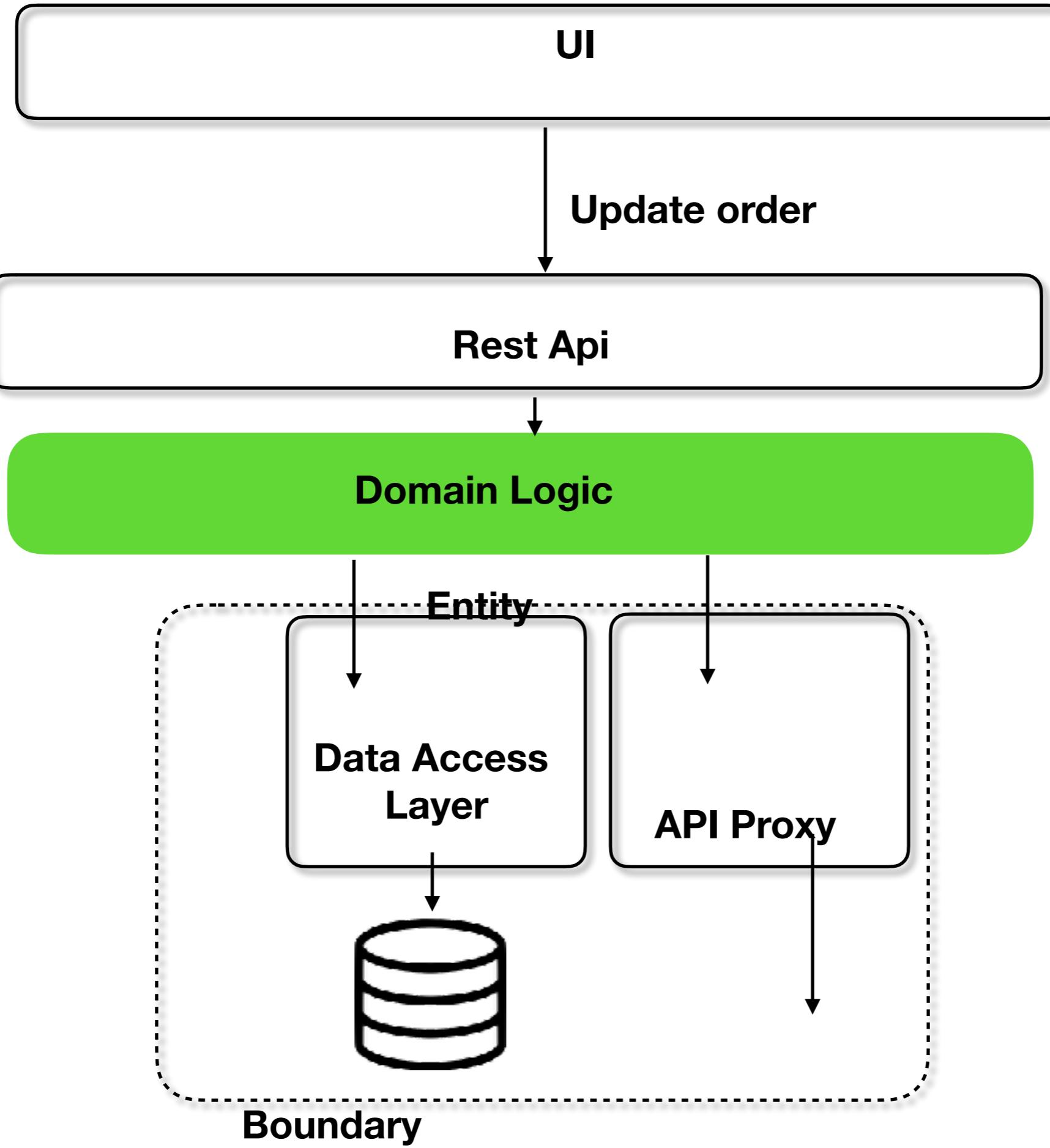




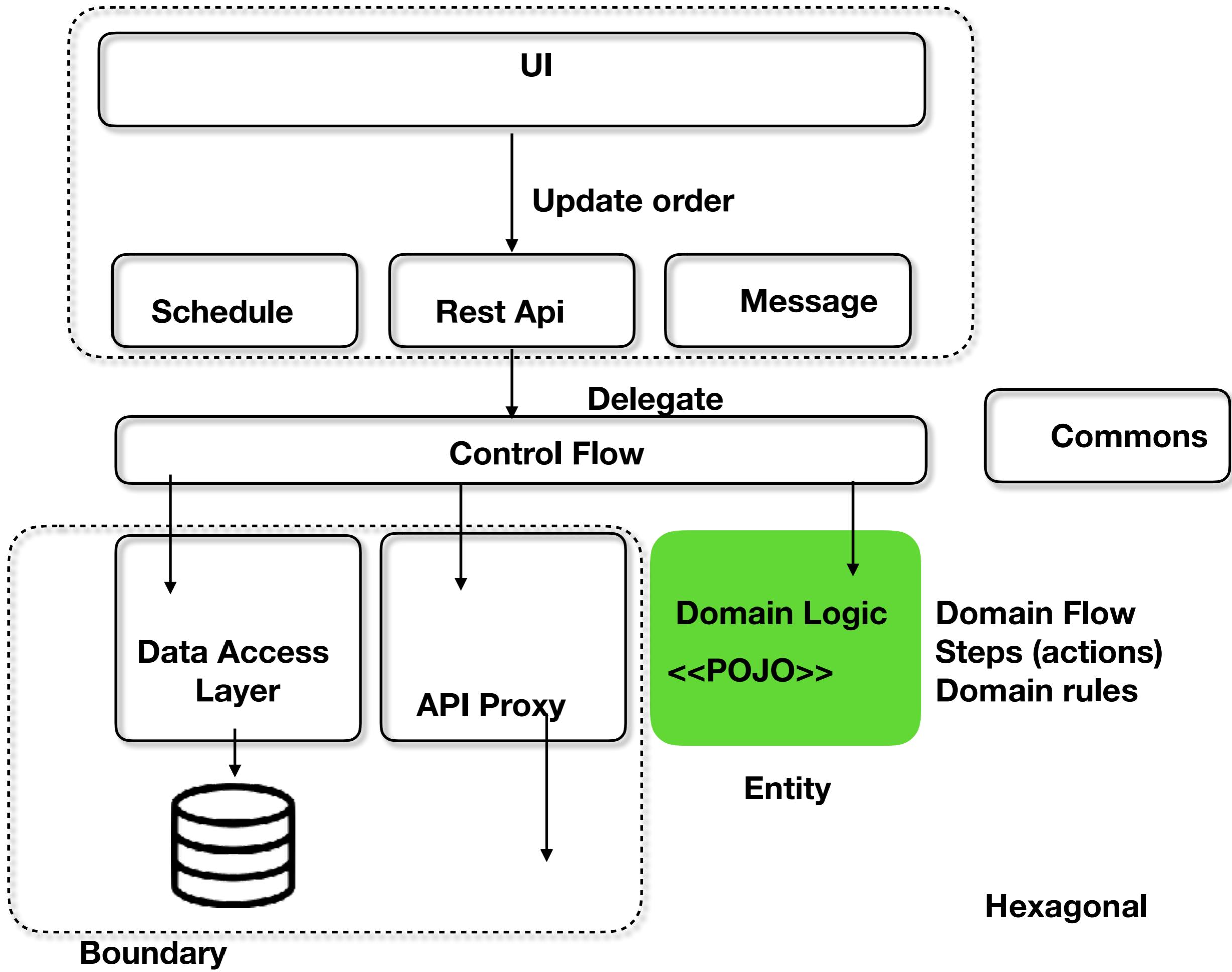
Variance

Core

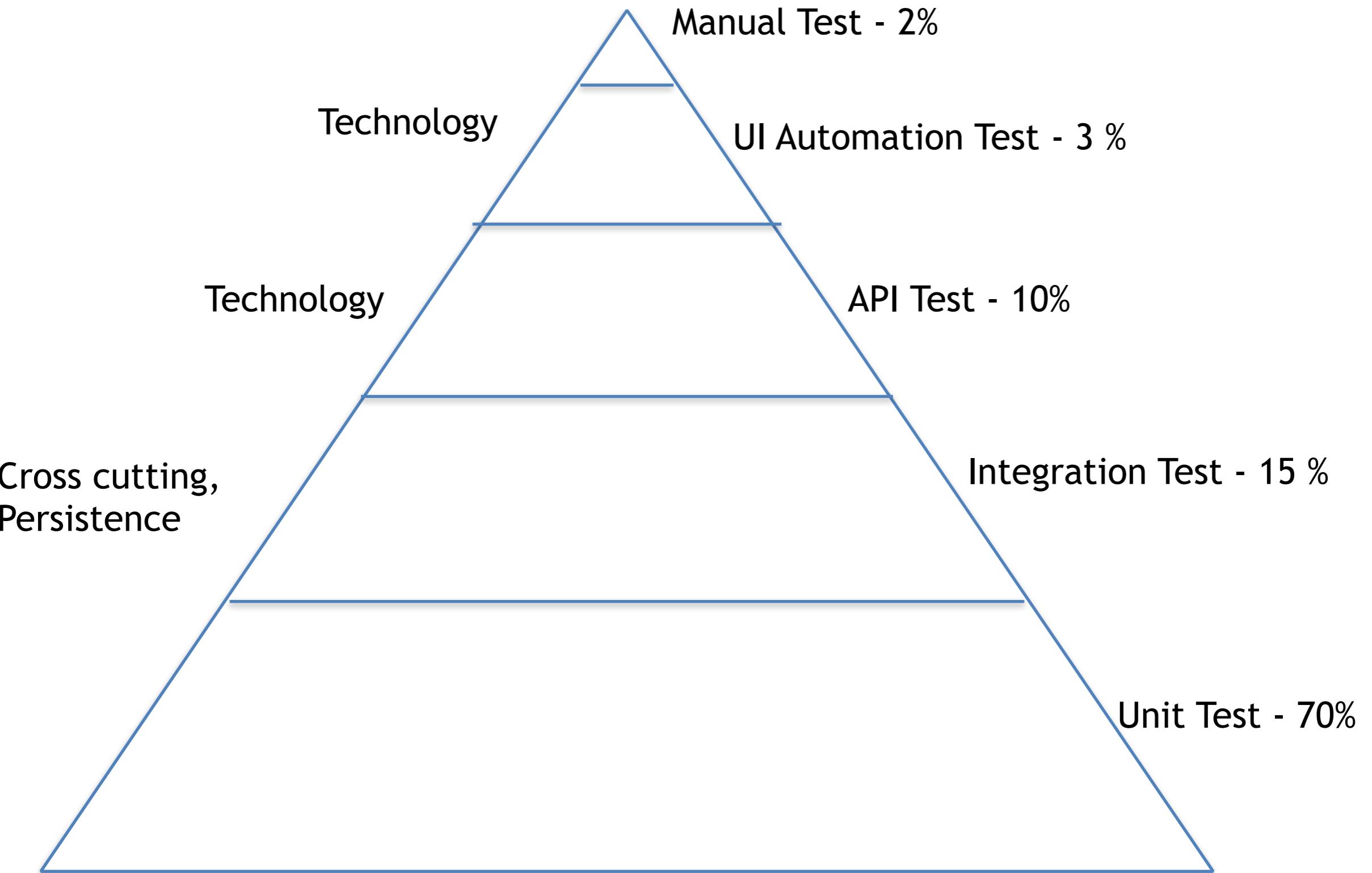




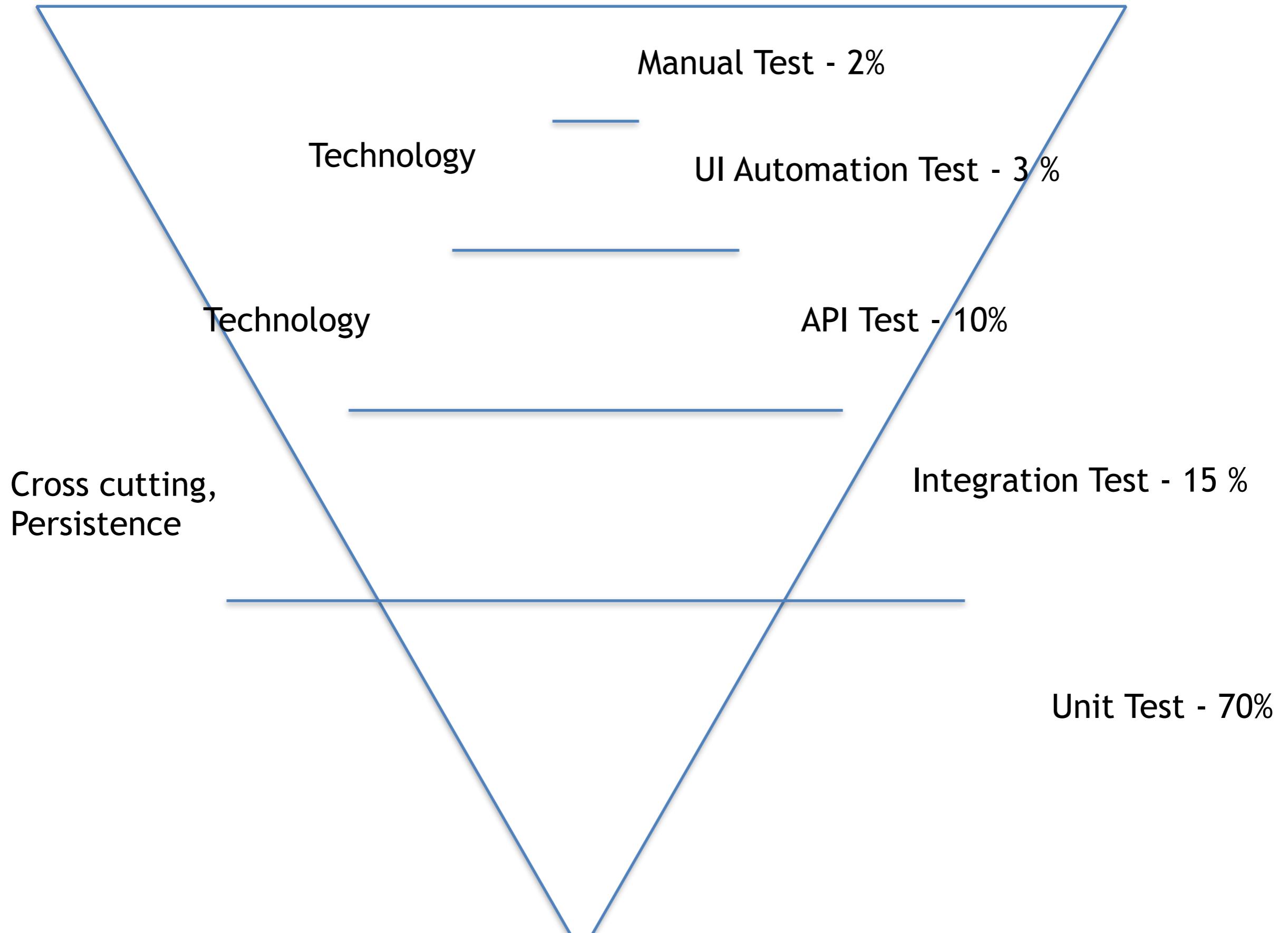
Boundary



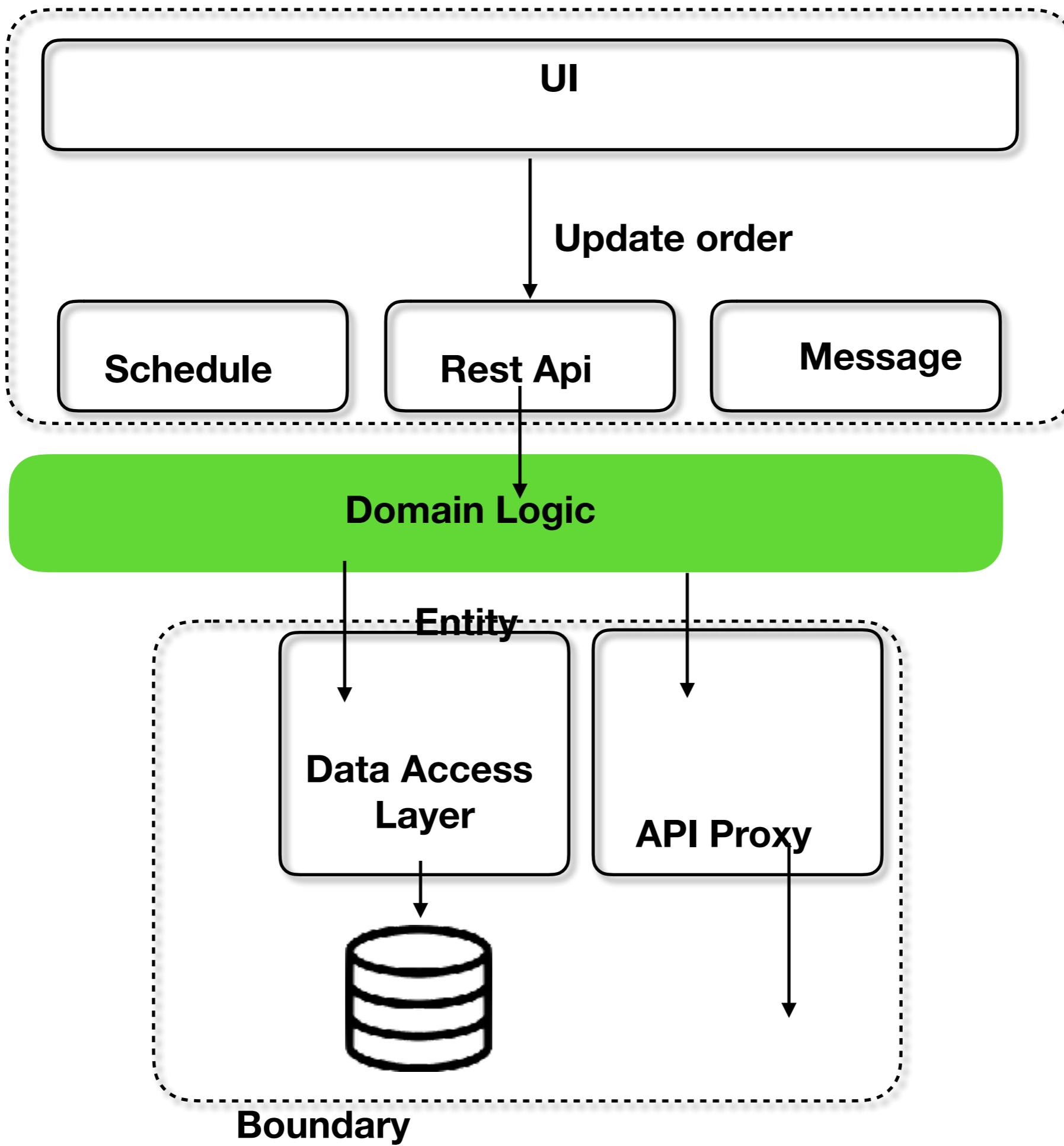
Pyramid test

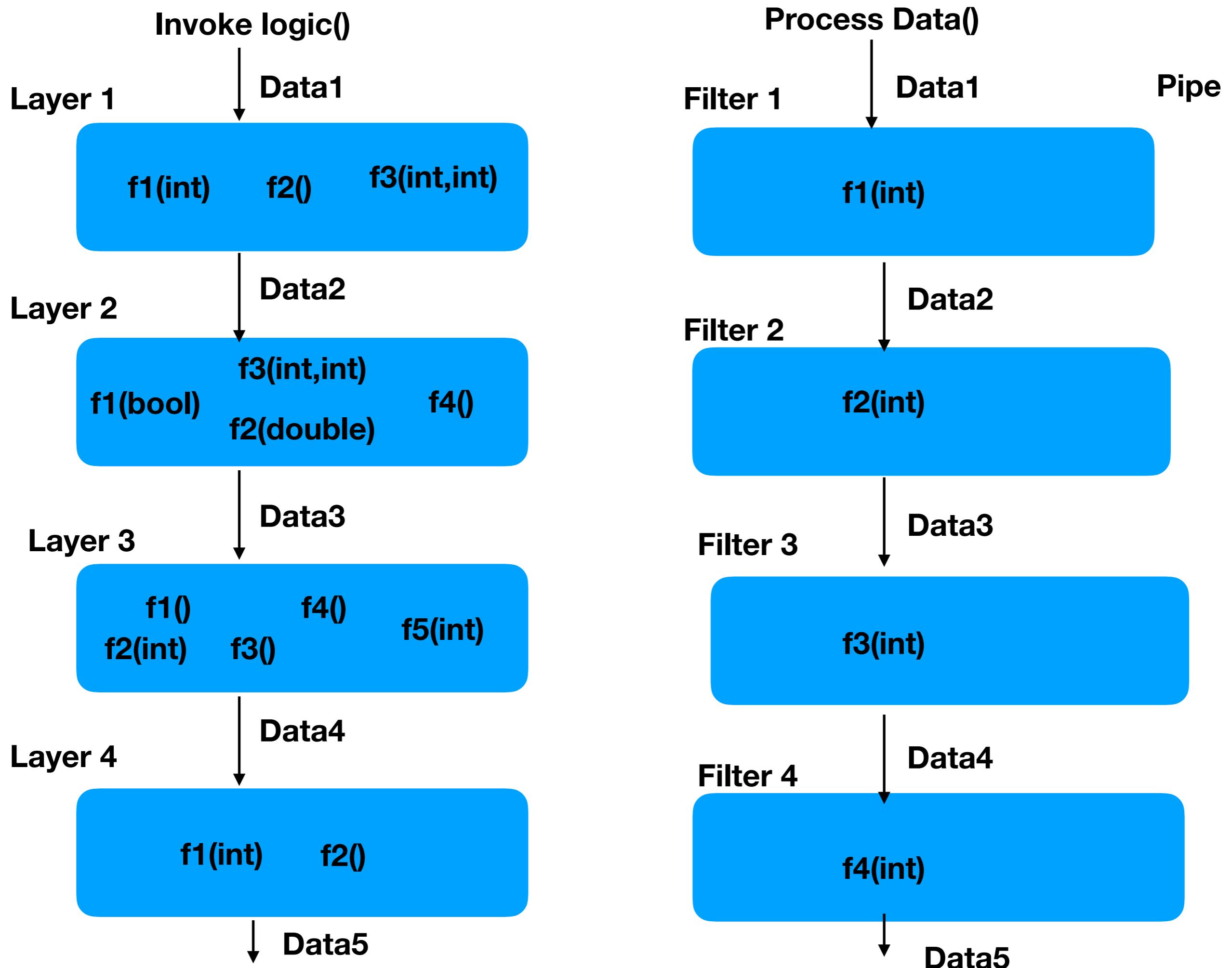


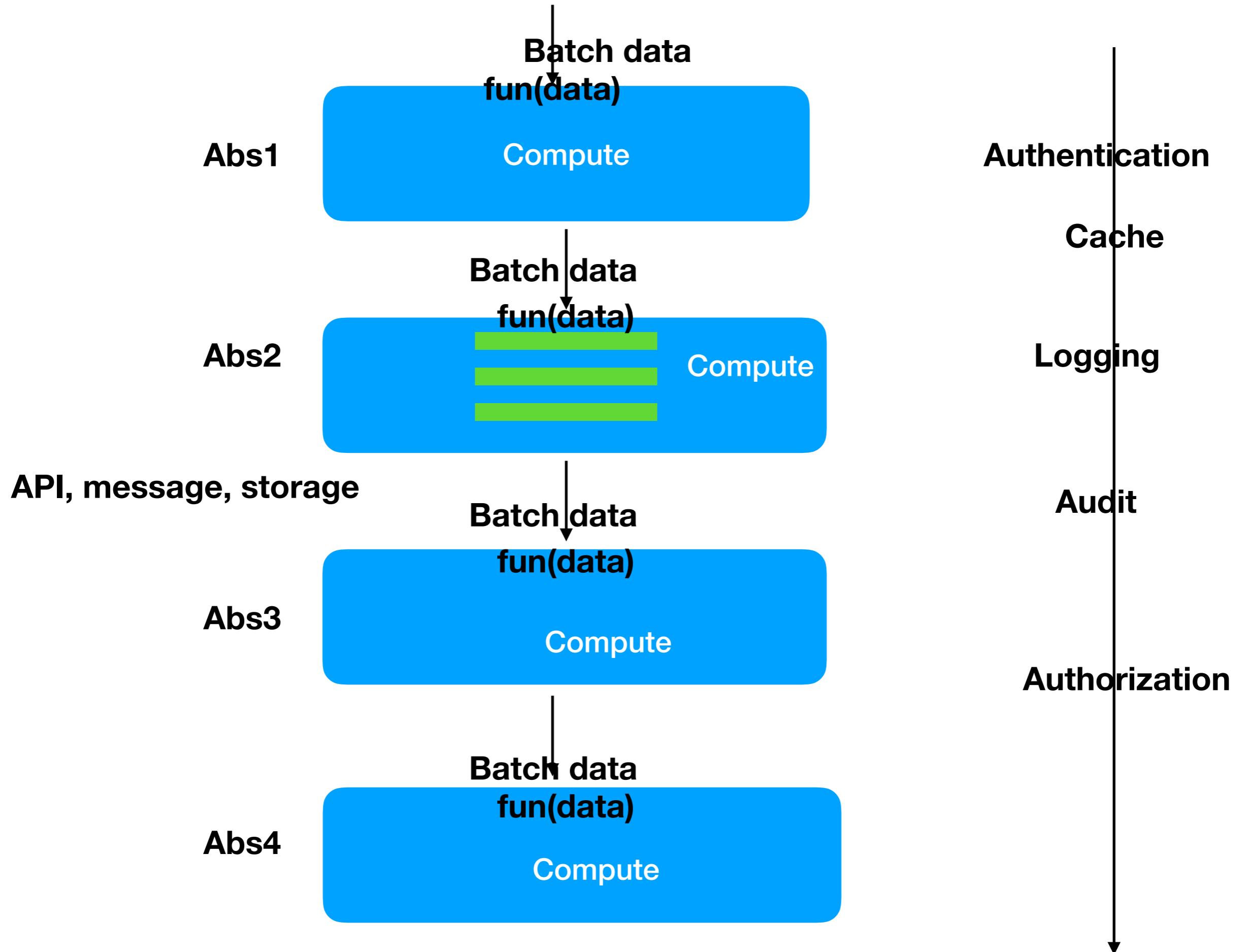
Cone test

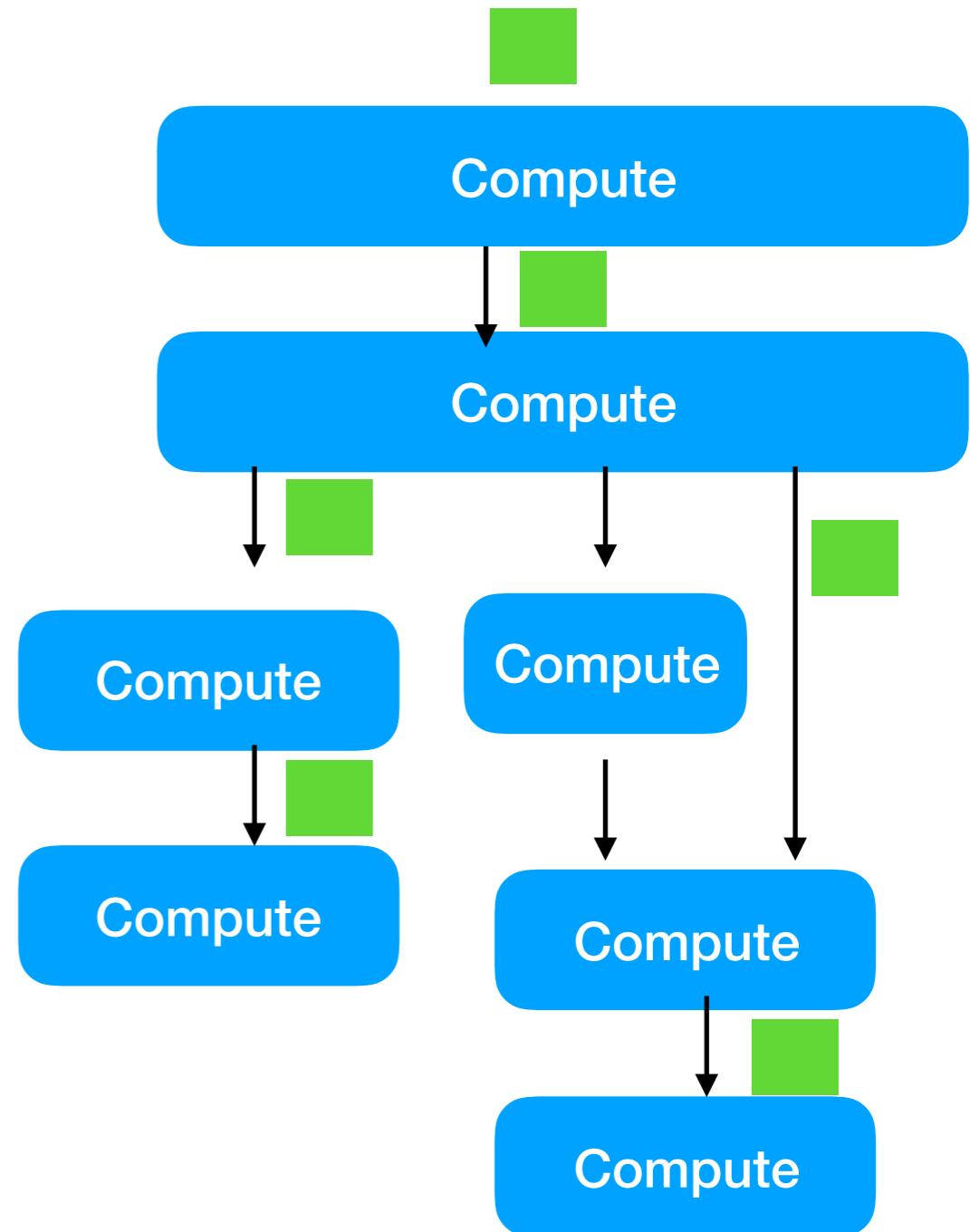
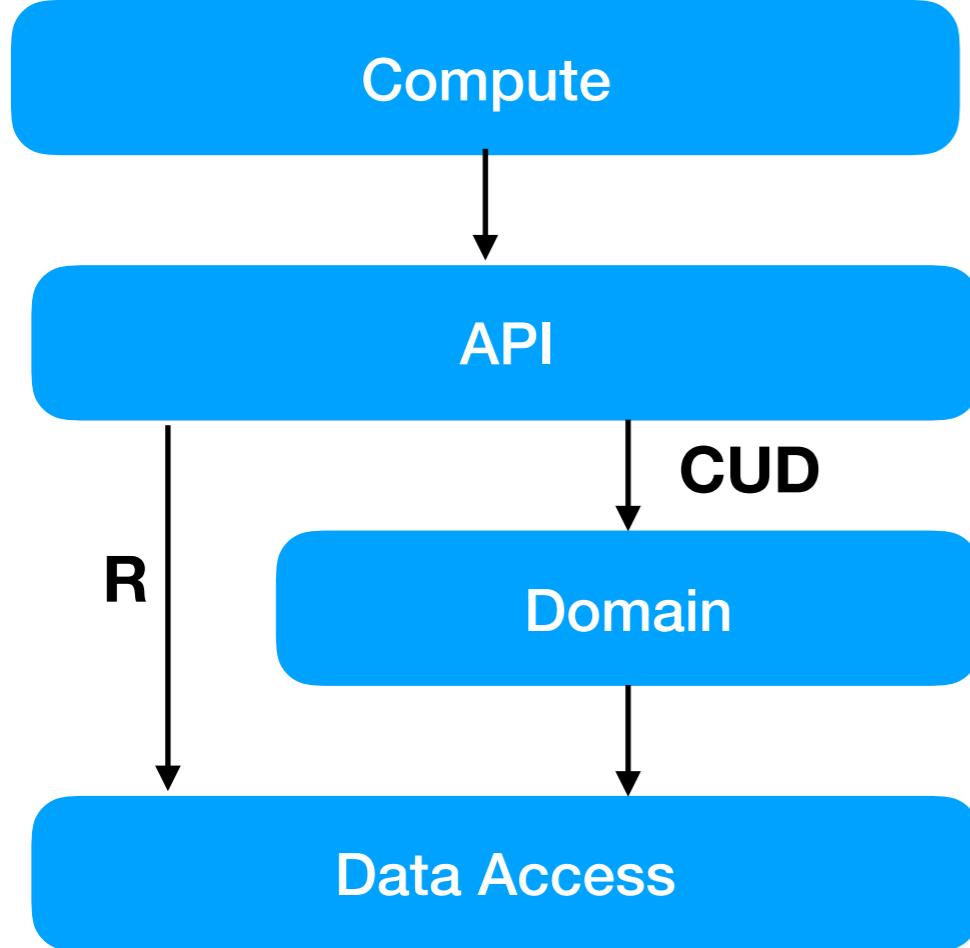


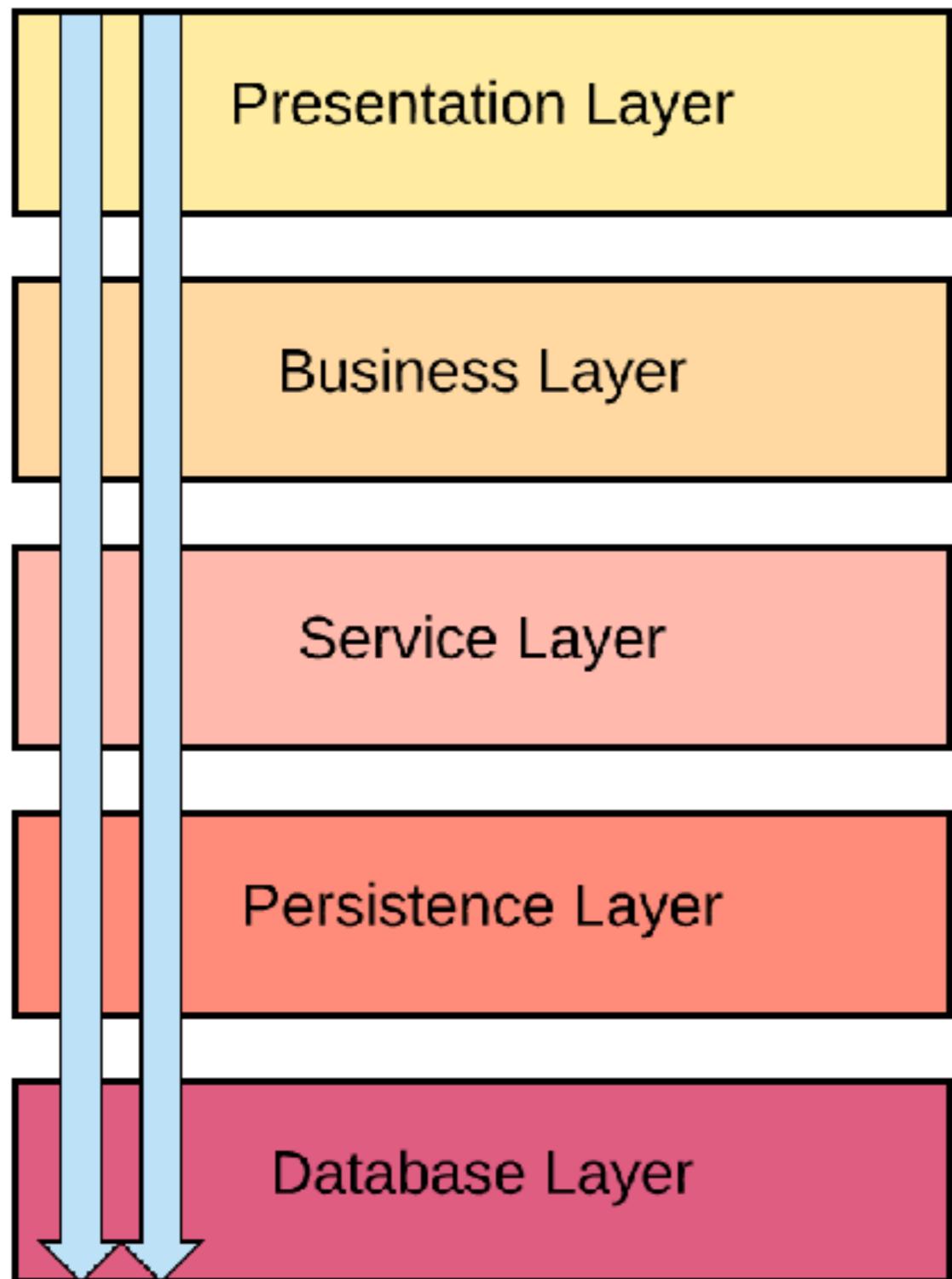
Boundary



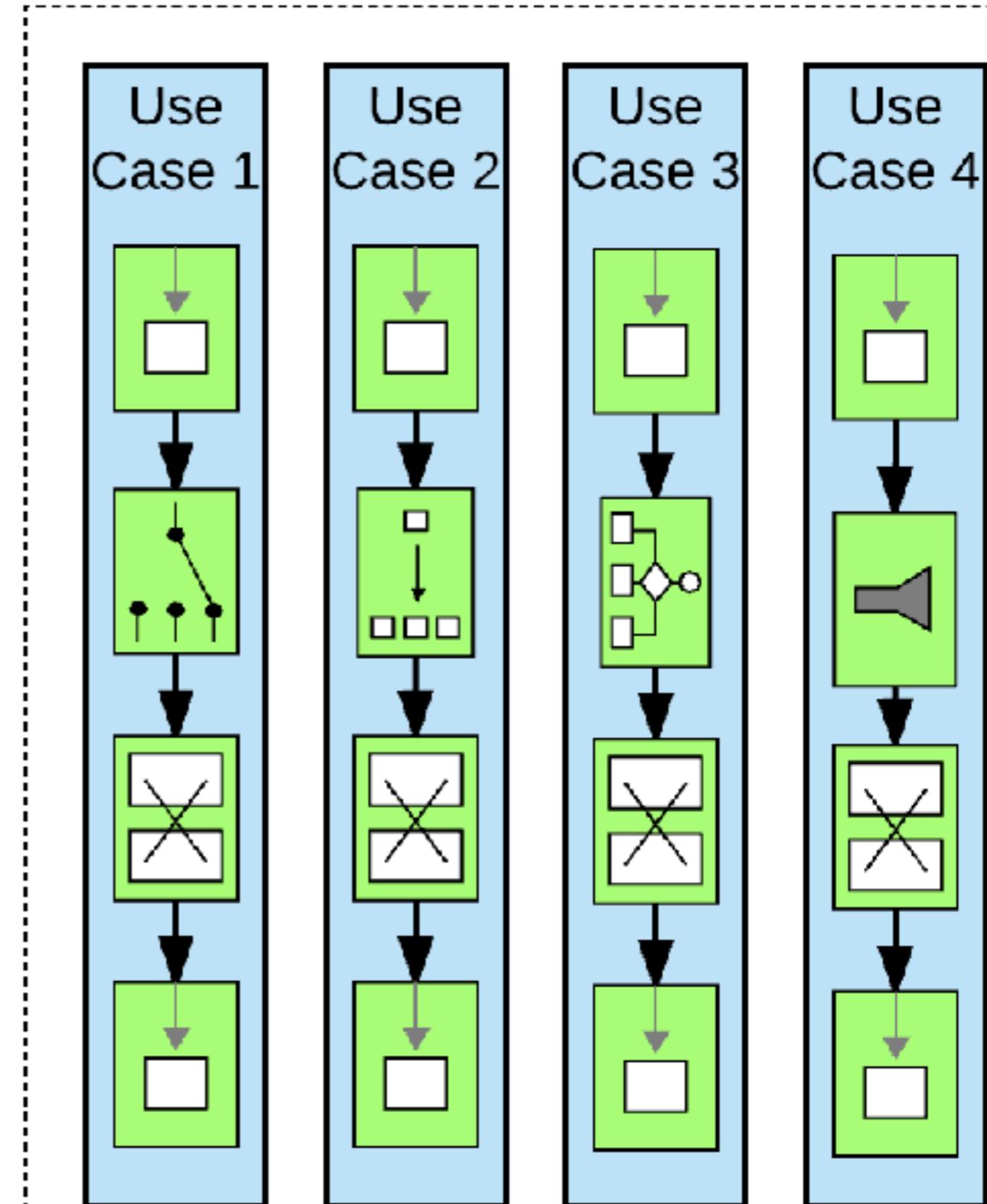








Layered Architecture



Pipes and Filters

Core

Layer

Layer

Filter

Filter

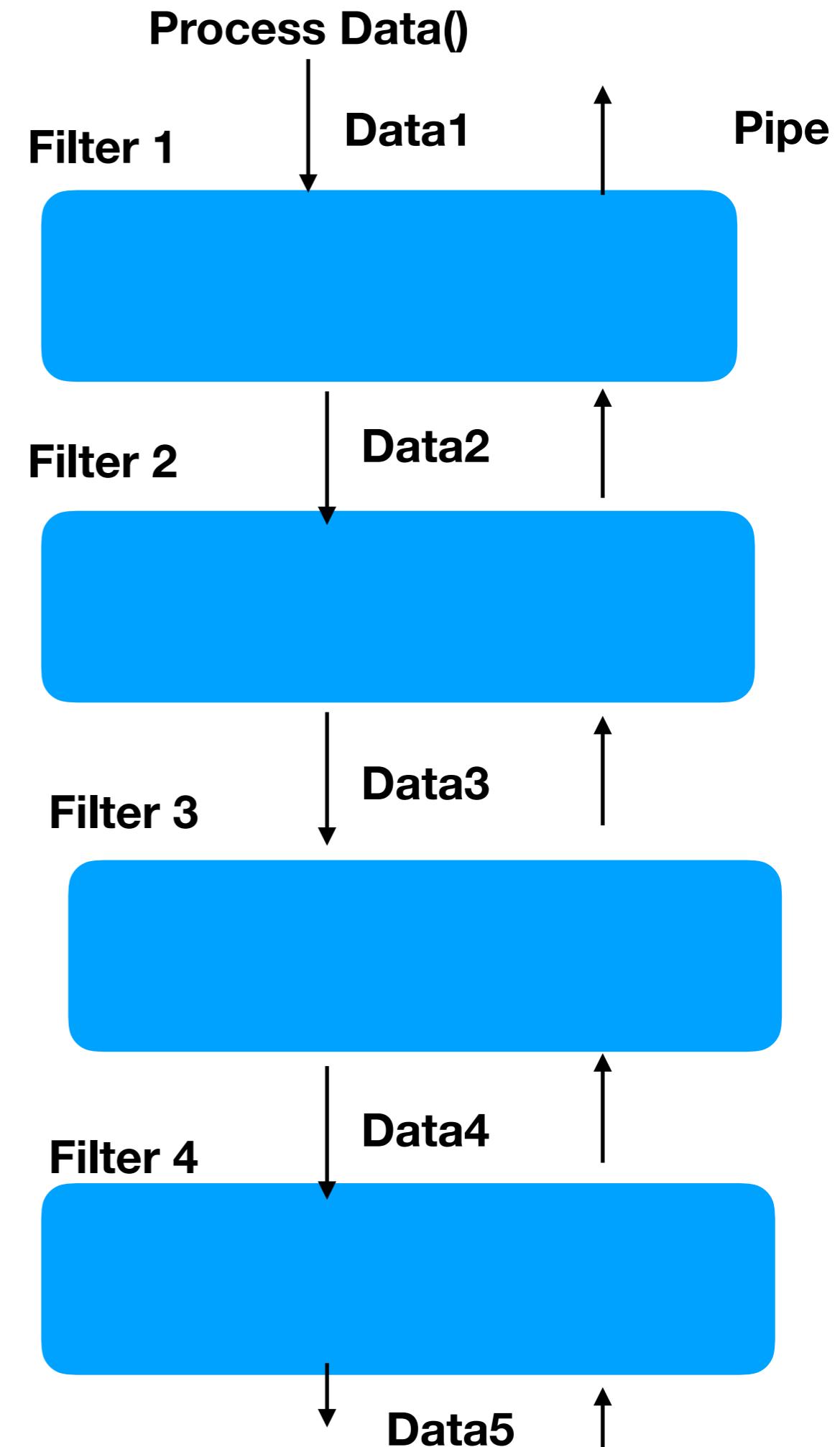
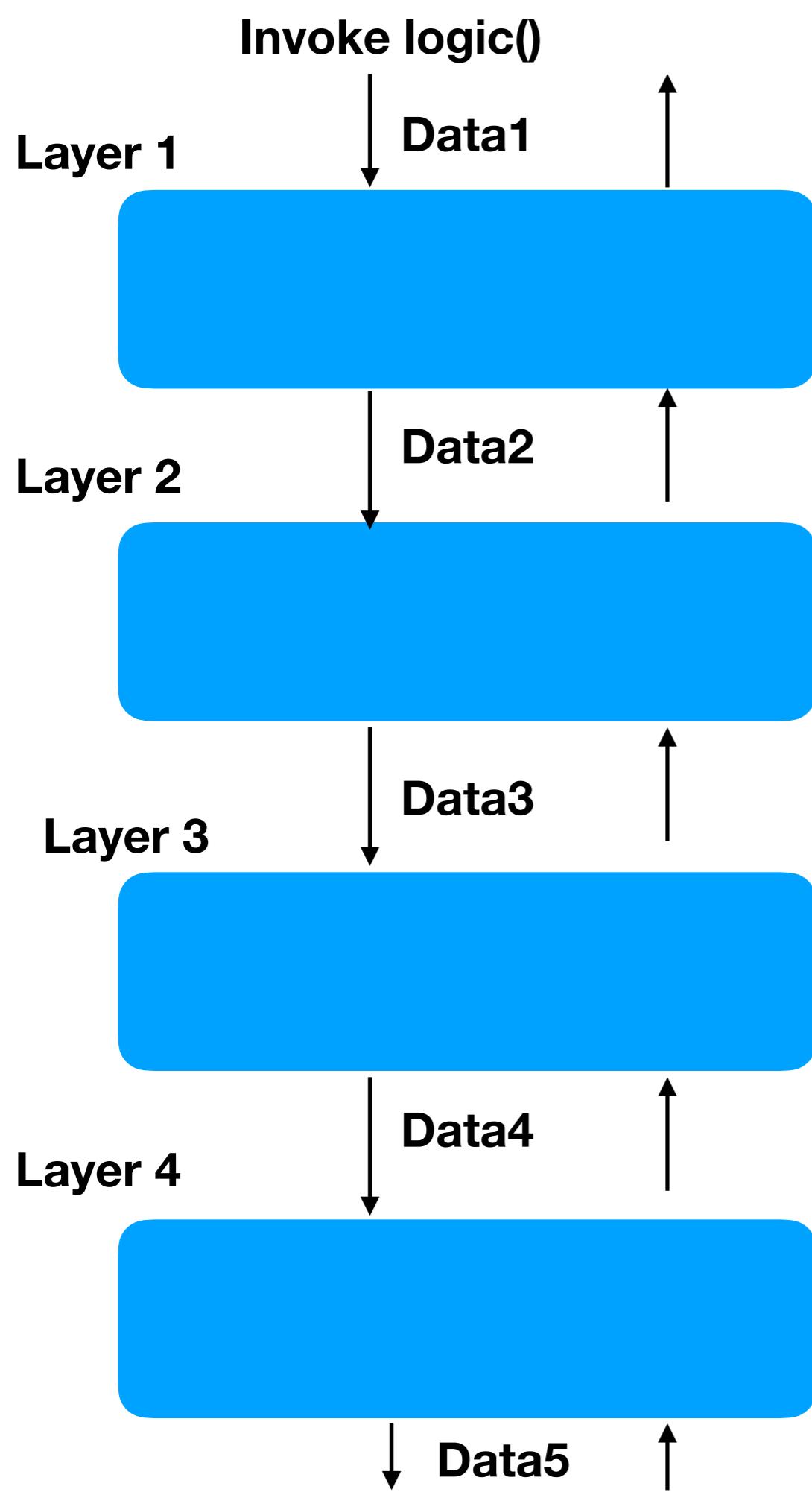
Filter

Layer

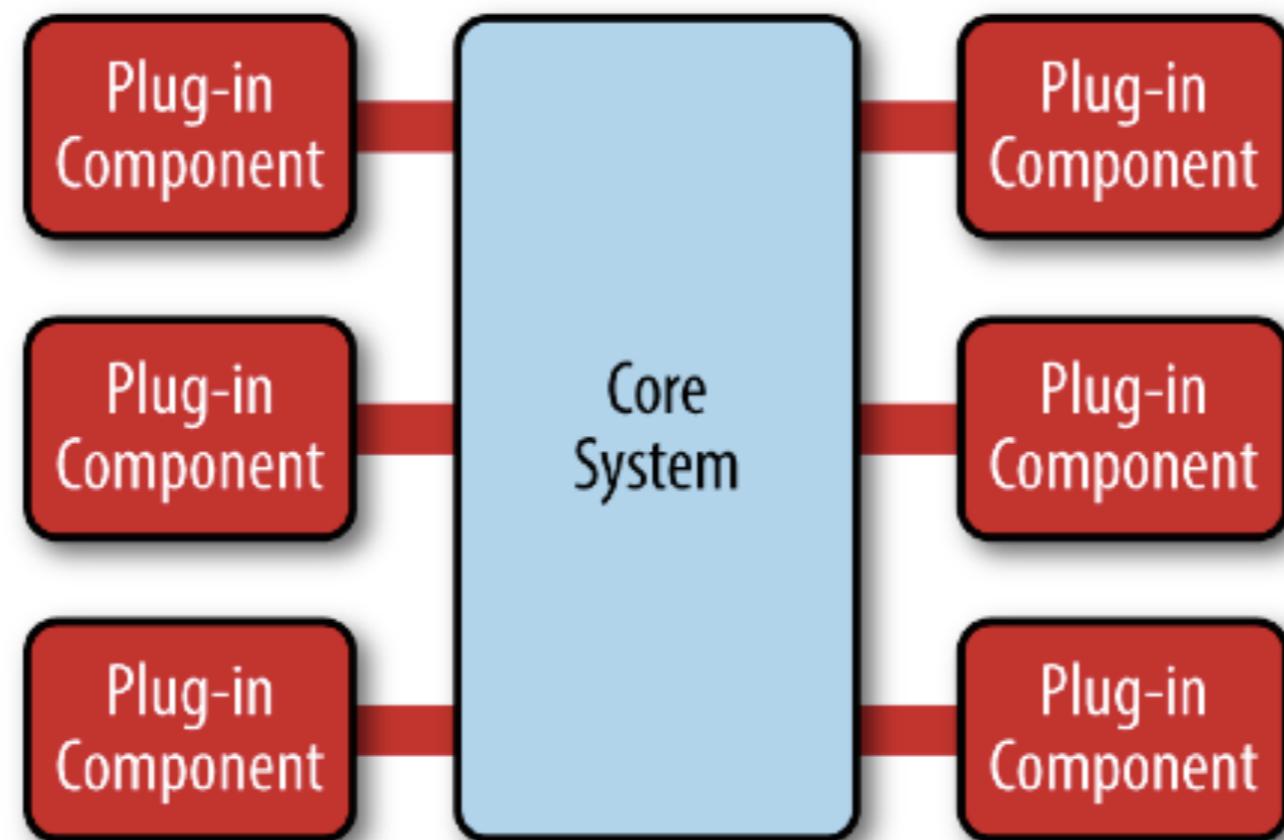
Plugins

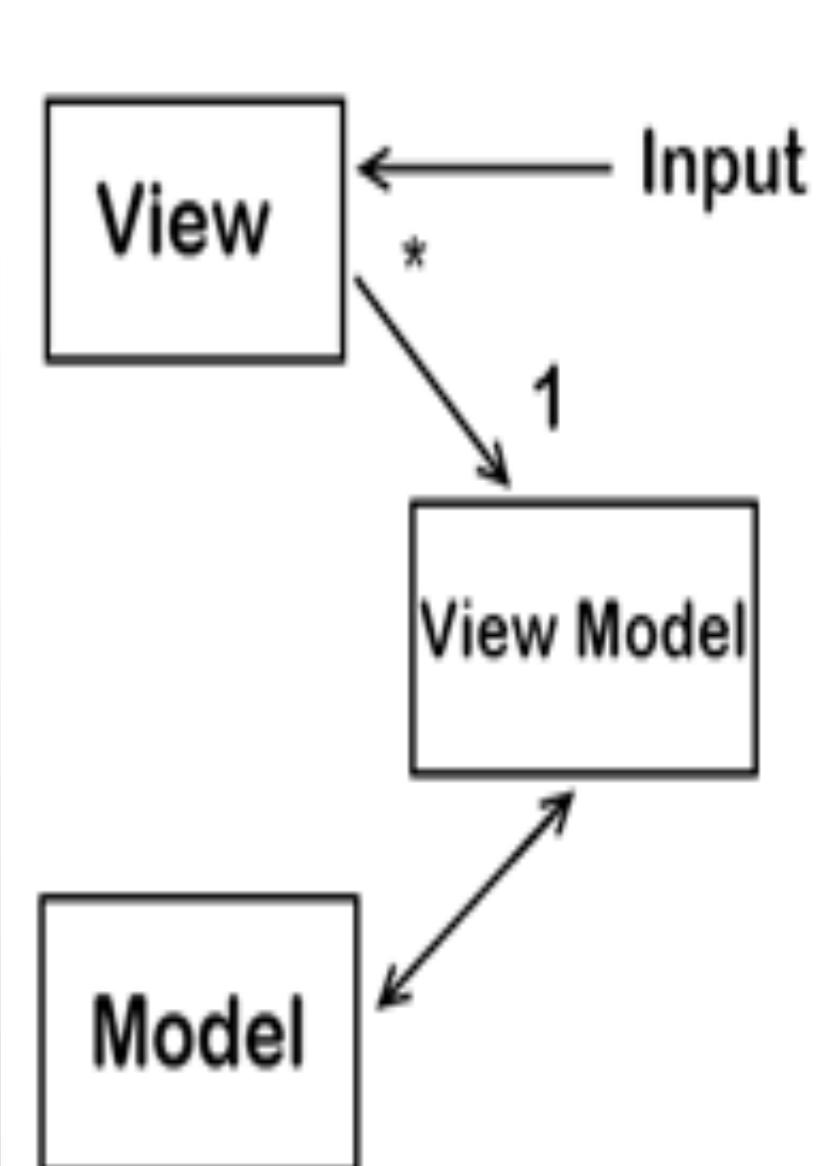
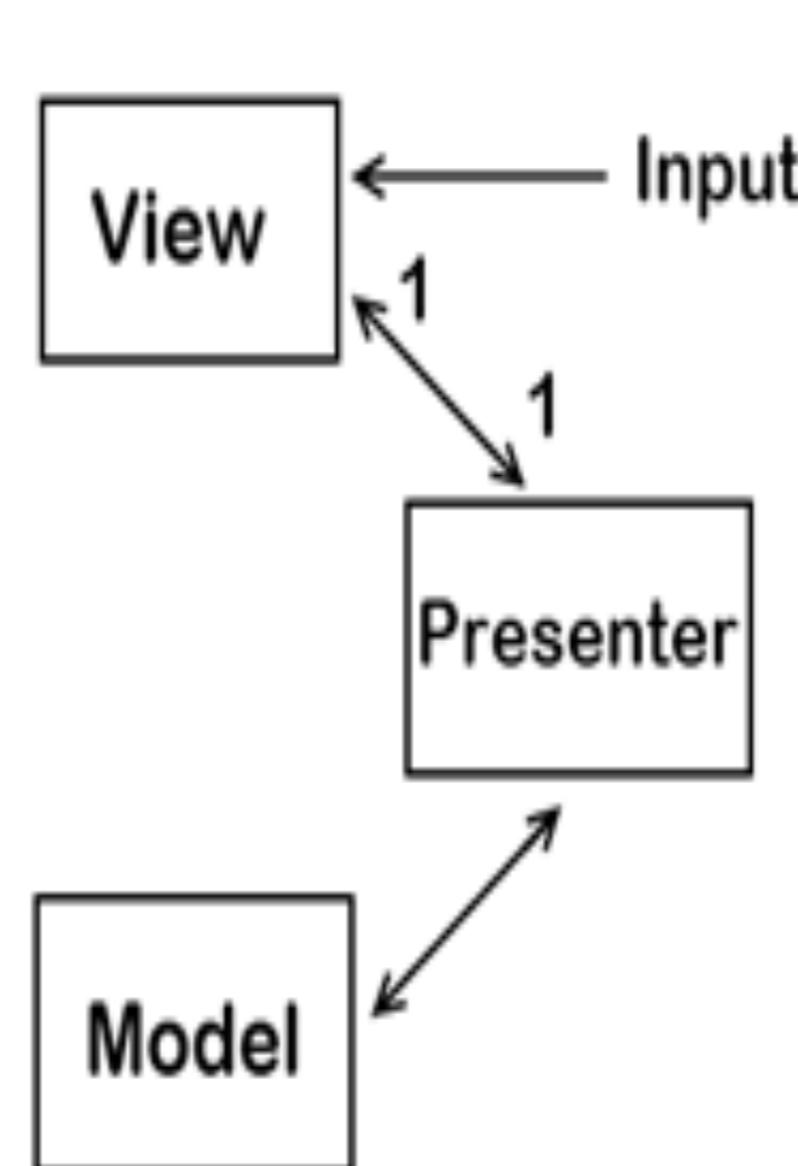
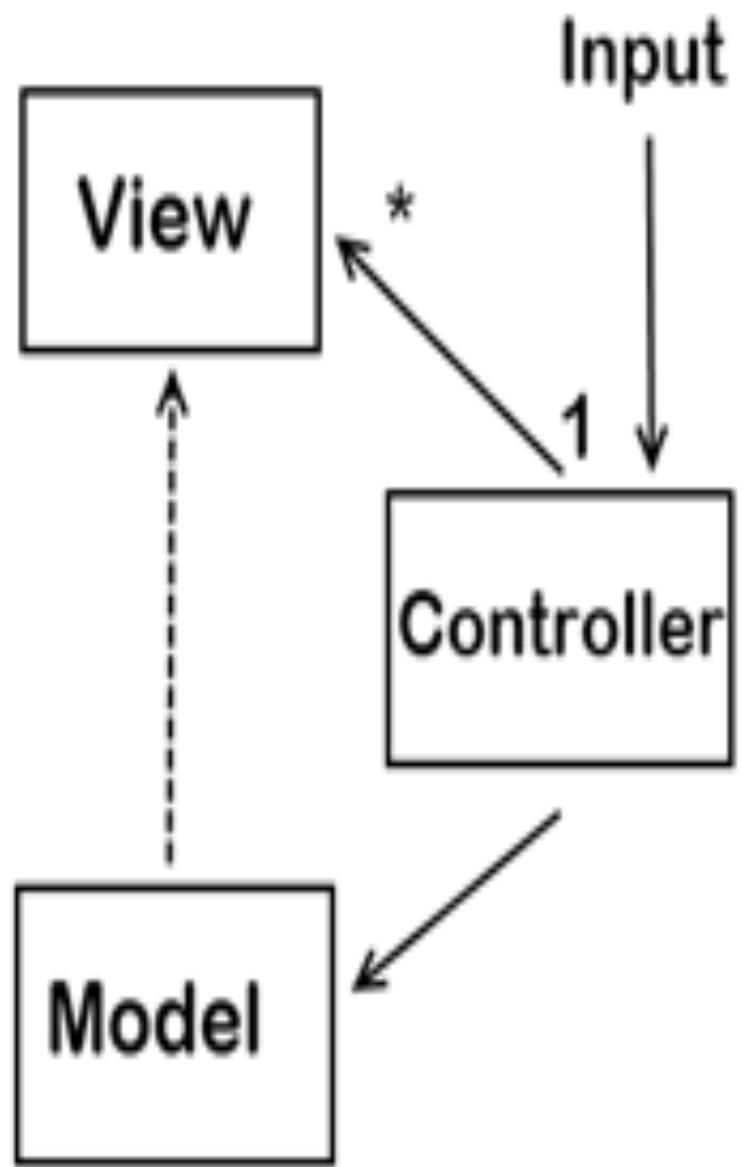
Plugins

Plugins



Microkernel Architecture

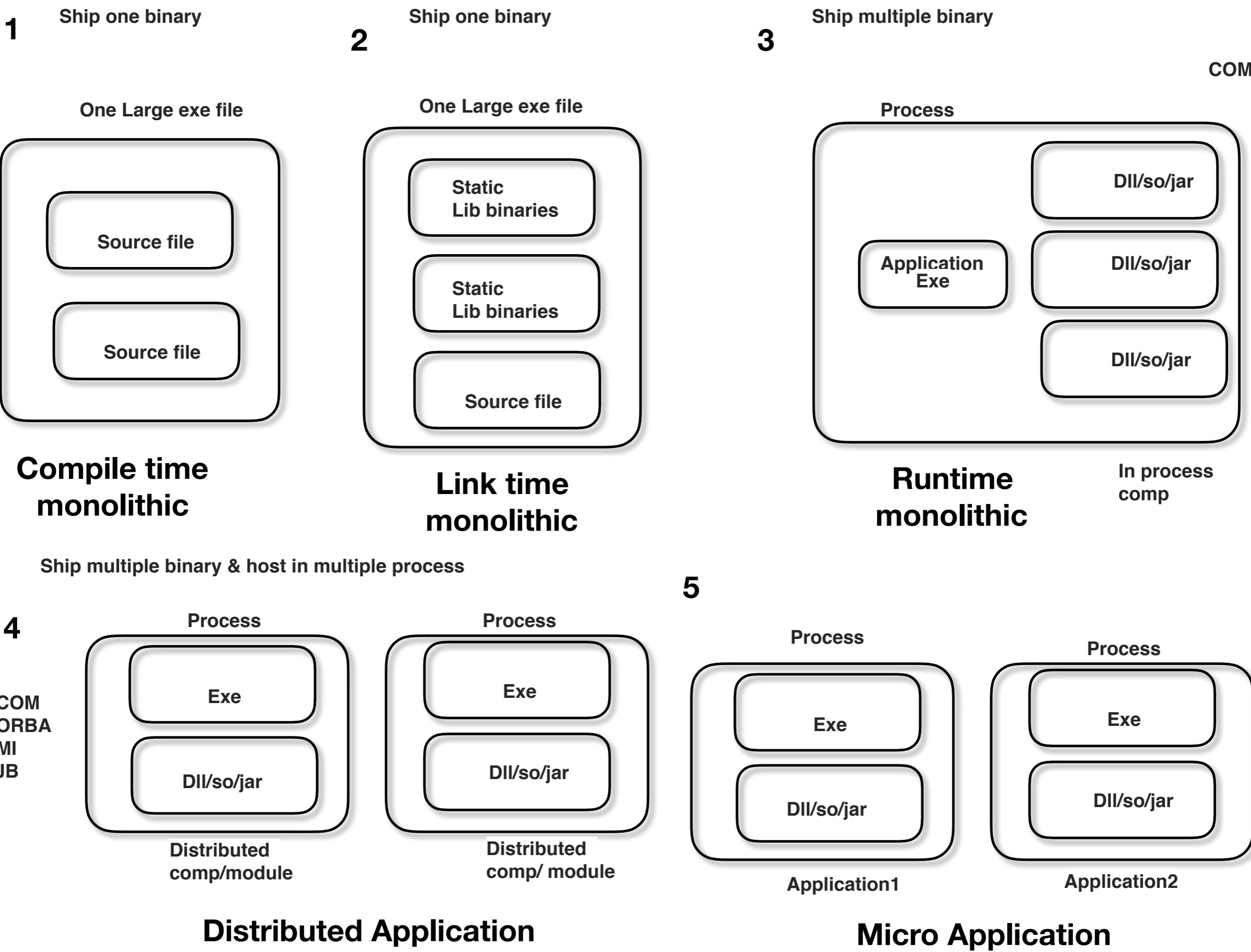




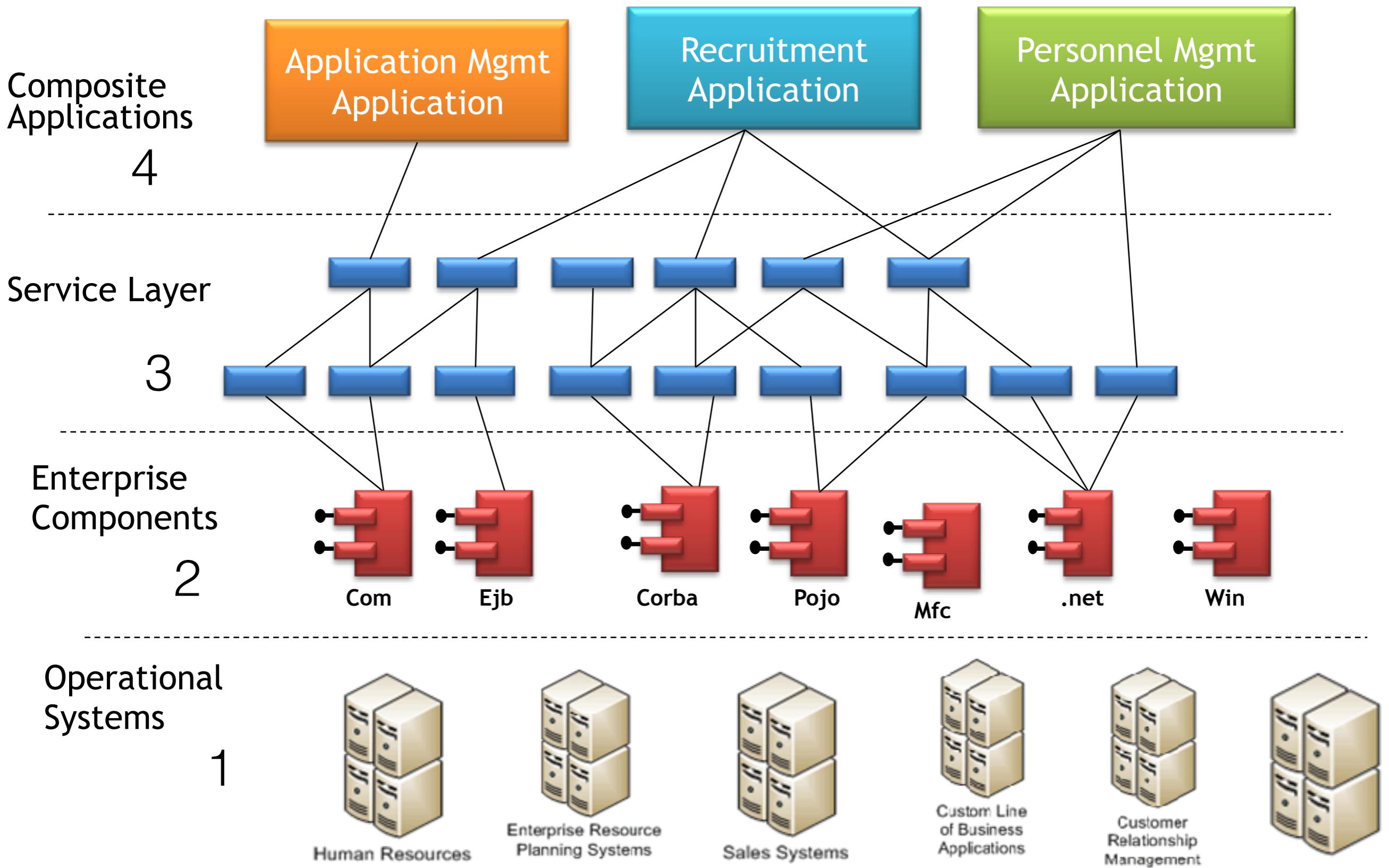
	Layered	Pipes and Filter	Micro Kernel
Low Coupling	--	++	++
Ease of Design	++	--	-
Agility (Shuffle, Rearrange, Add)	--	++	++
Unit Testability	--	++	++
Needs less domain knowledge	++	--	--

	Layered	Pipes and Filter	Micro Kernel
Low Coupling	--	++	++
Ease of Design	++	-	--
Agility (change)	--	+	++
Unit Testability	++	++	++
Needs less domain knowledge	++	-	-- (more)

Microservice



Service Oriented Architecture



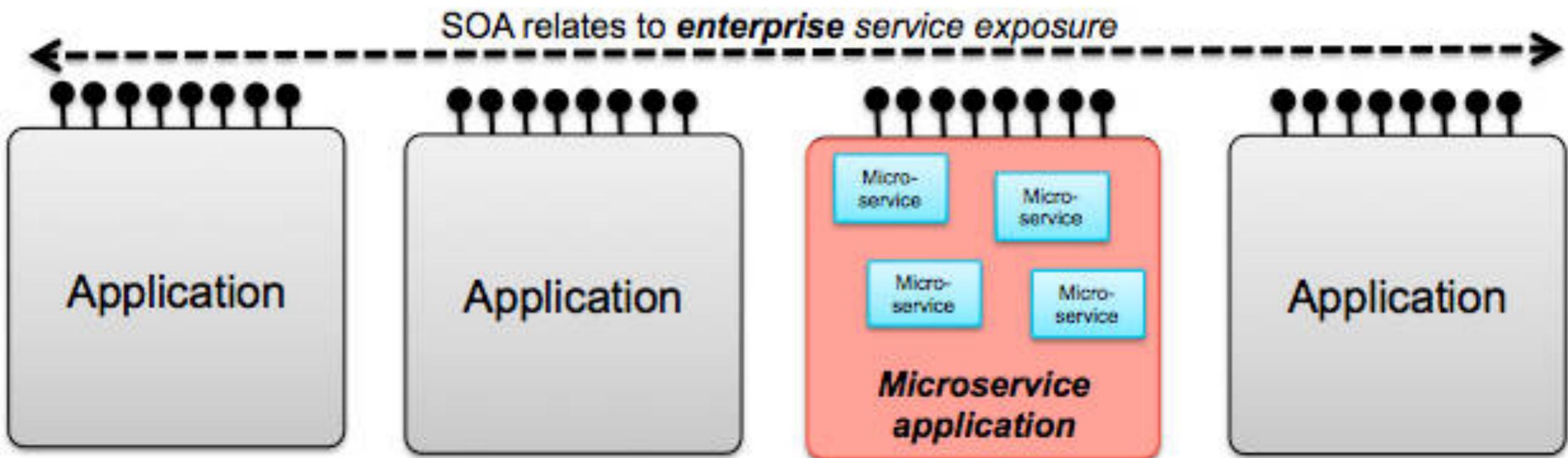
ToDo Portal
(Single Page Application)

ToDo API Service
(Api Application)

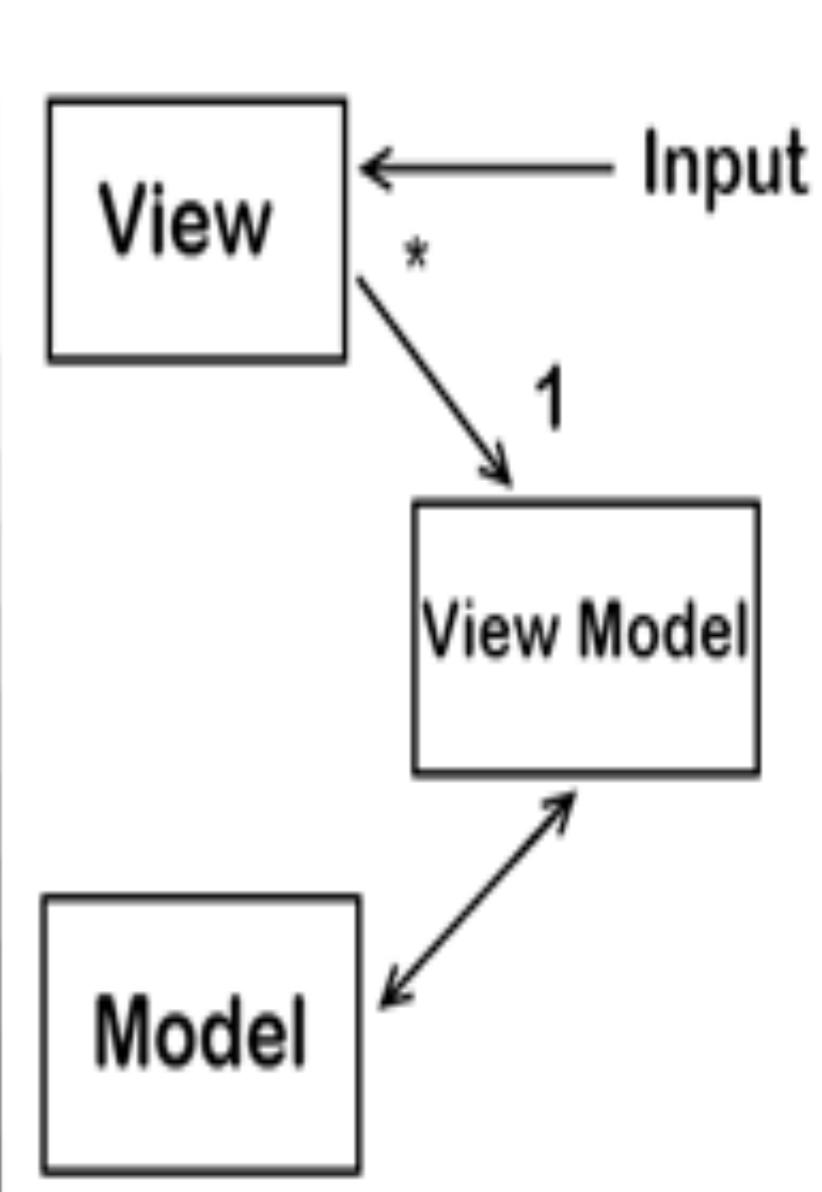
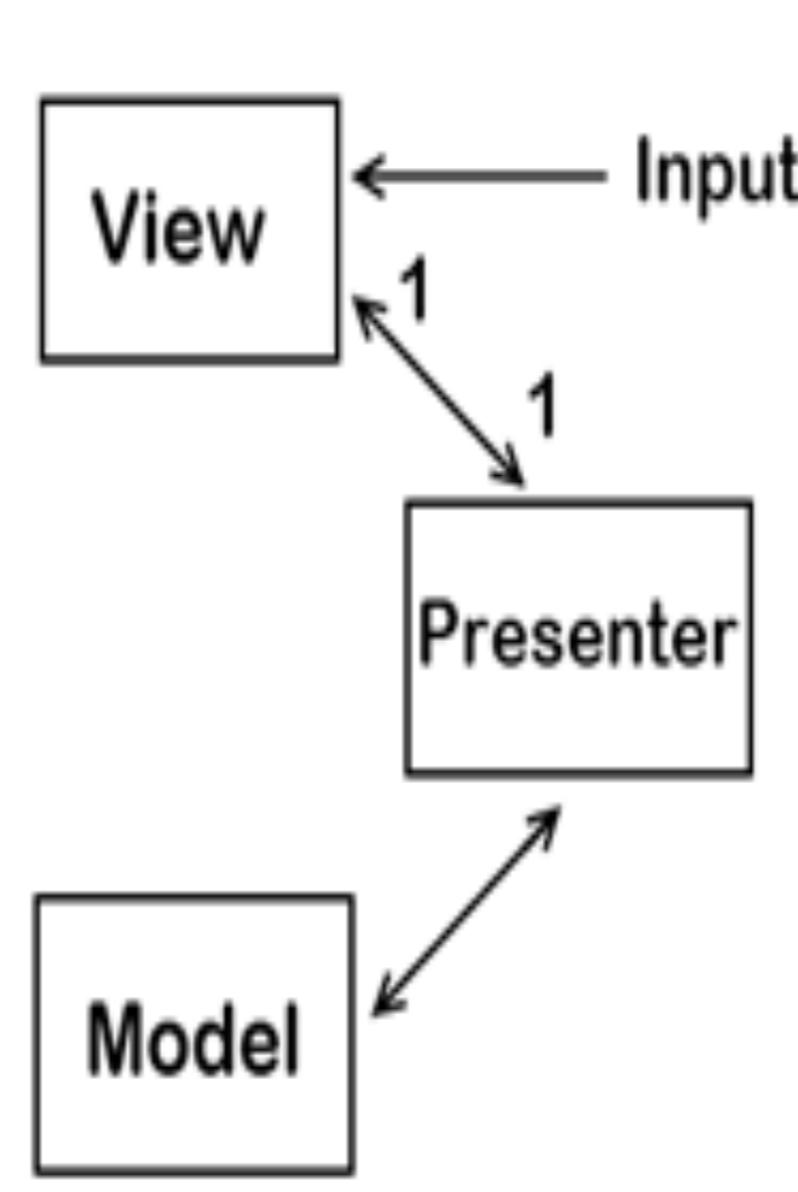
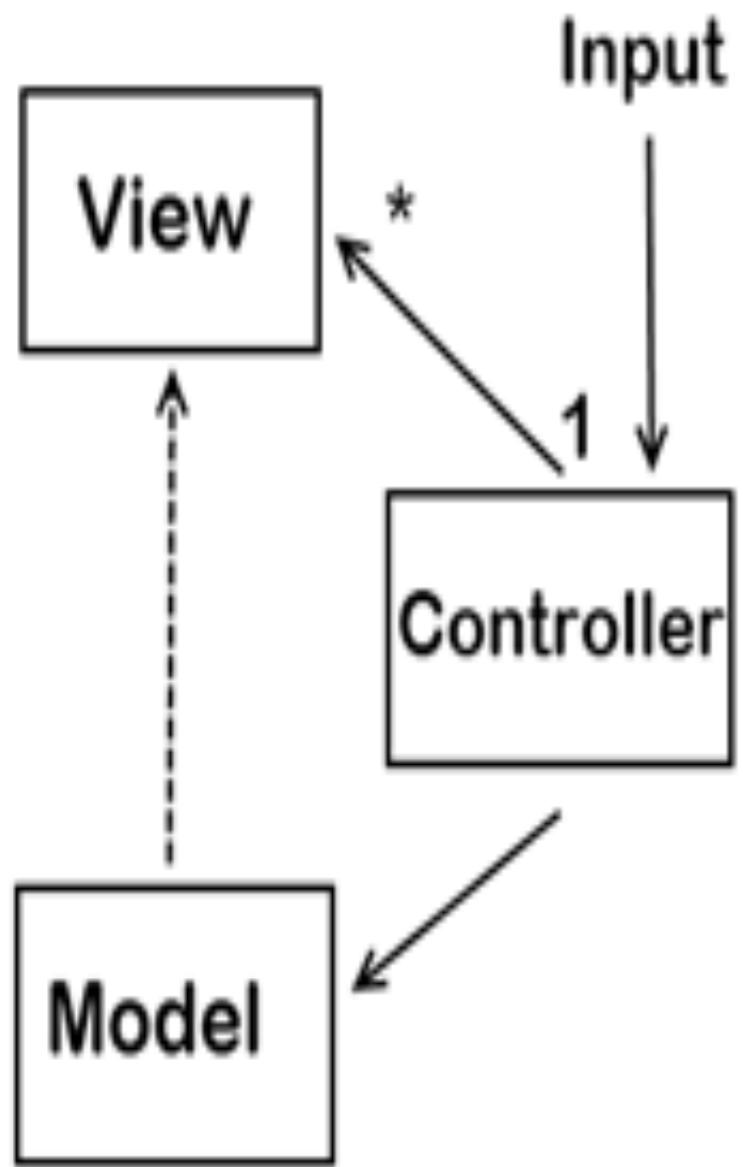


ToDo Calendar Service
(Background Application)

ToDo Prediction Service
(Background Application)



Microservices relate to
application architecture



Break an app into smaller manageable piece

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

Application

Modules

Modules

Modules

3 or 4

Application

Small App

Small App

Small App

Modules

Modules

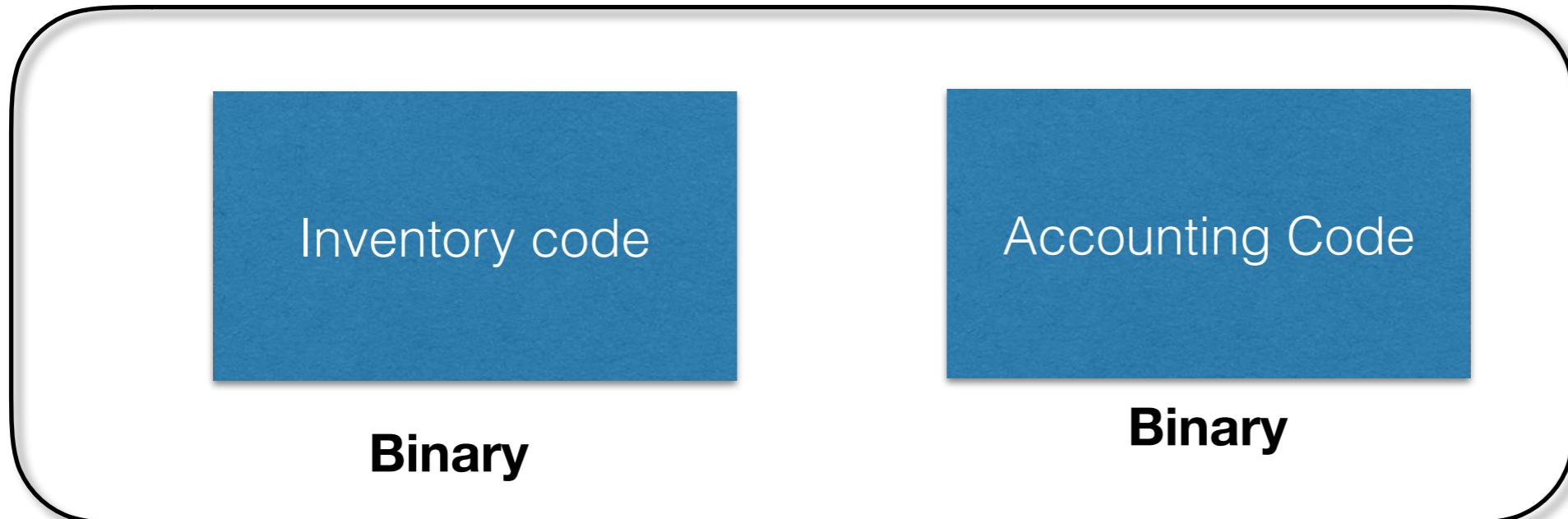
Modules

Modules

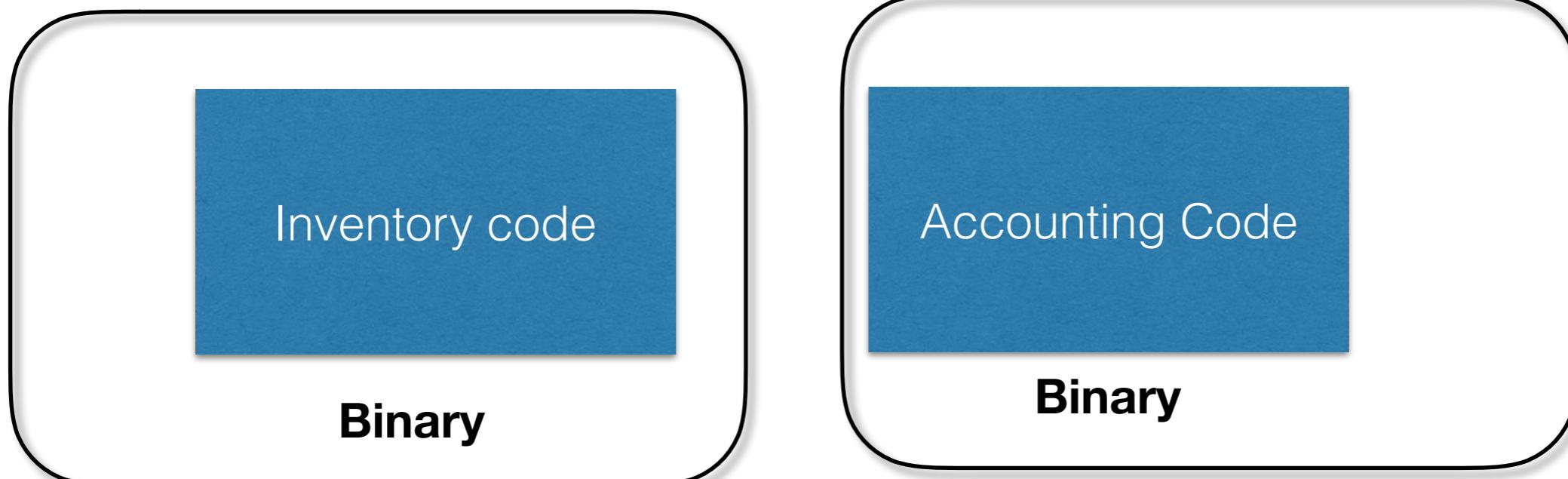
Modules

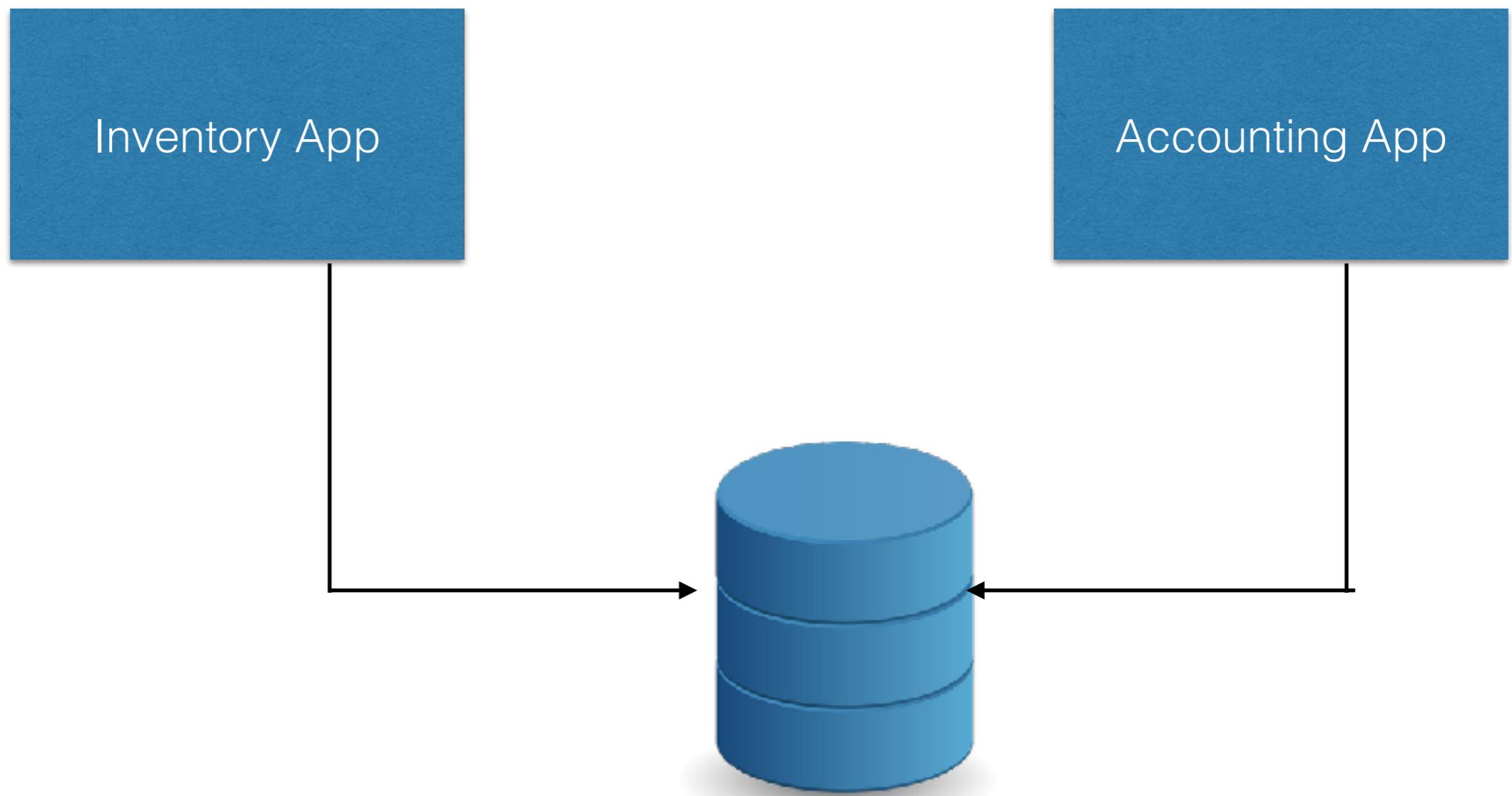
Modules

App

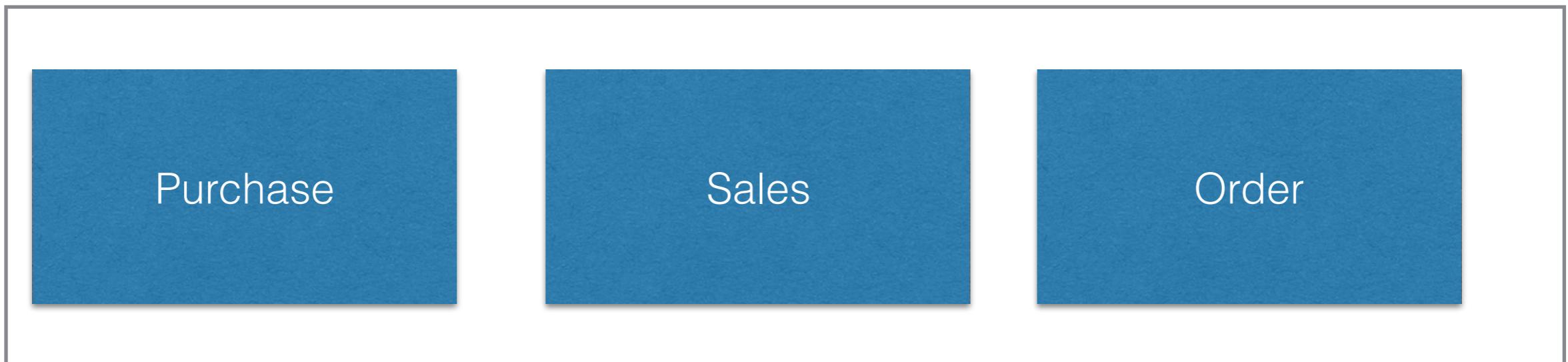


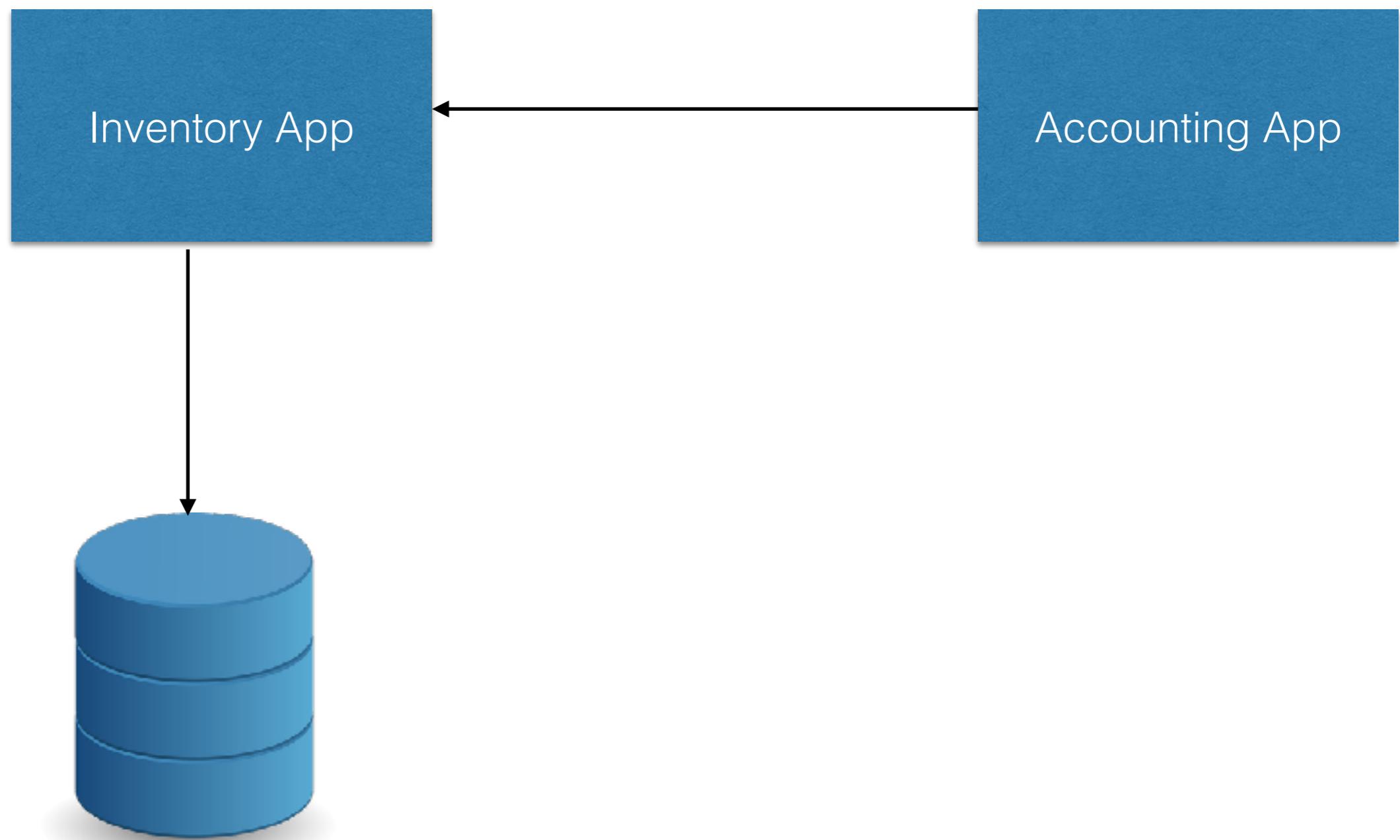
App

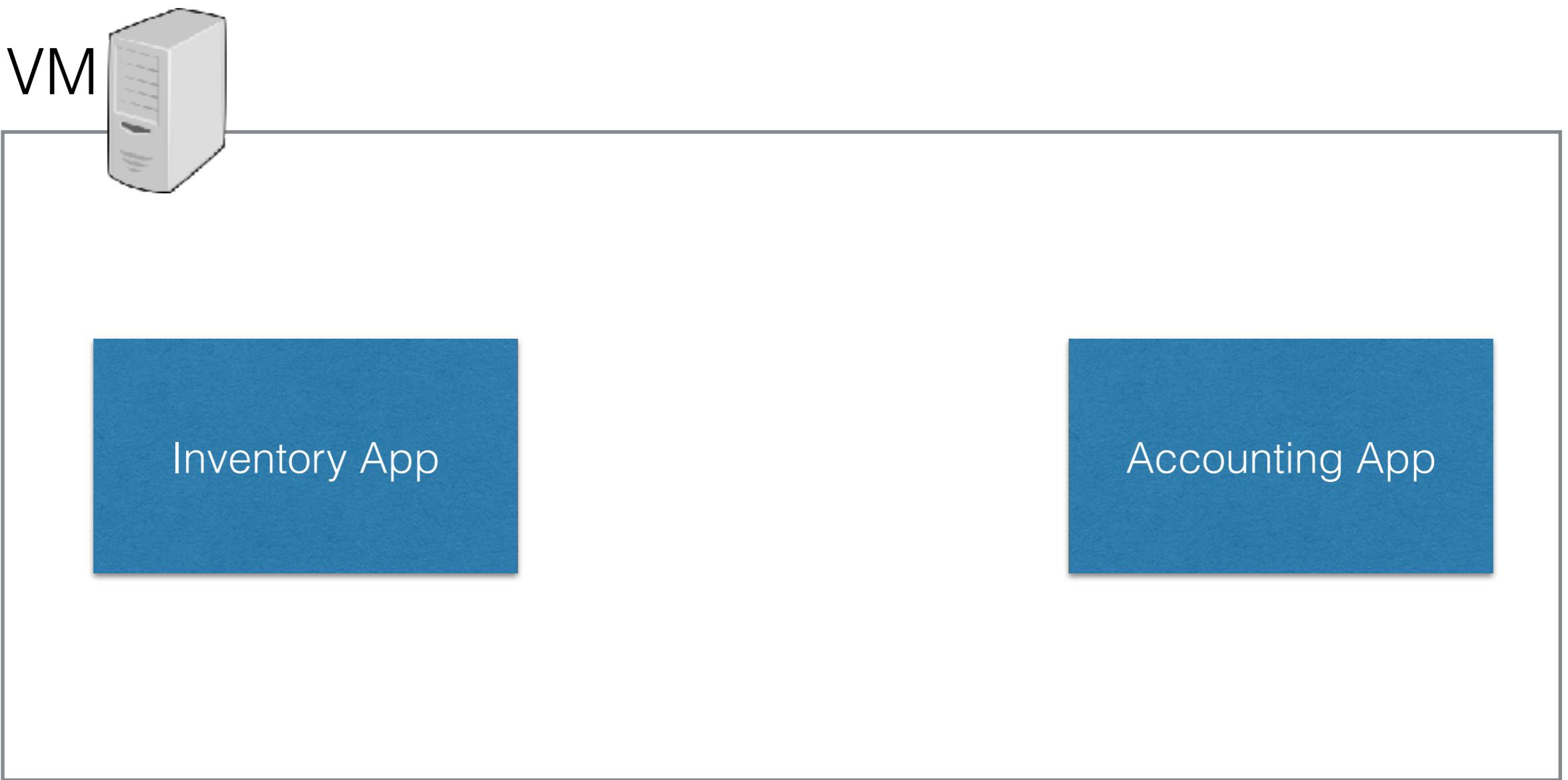


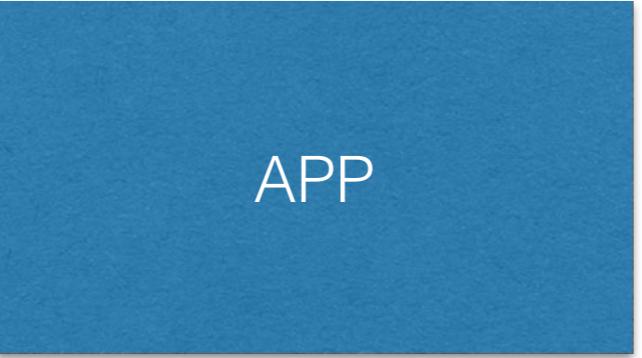


Inventory Application

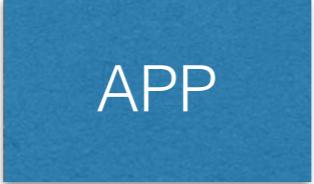




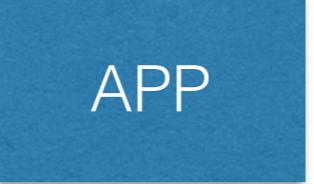




APP



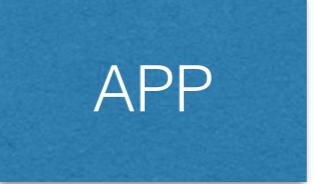
APP



APP



APP

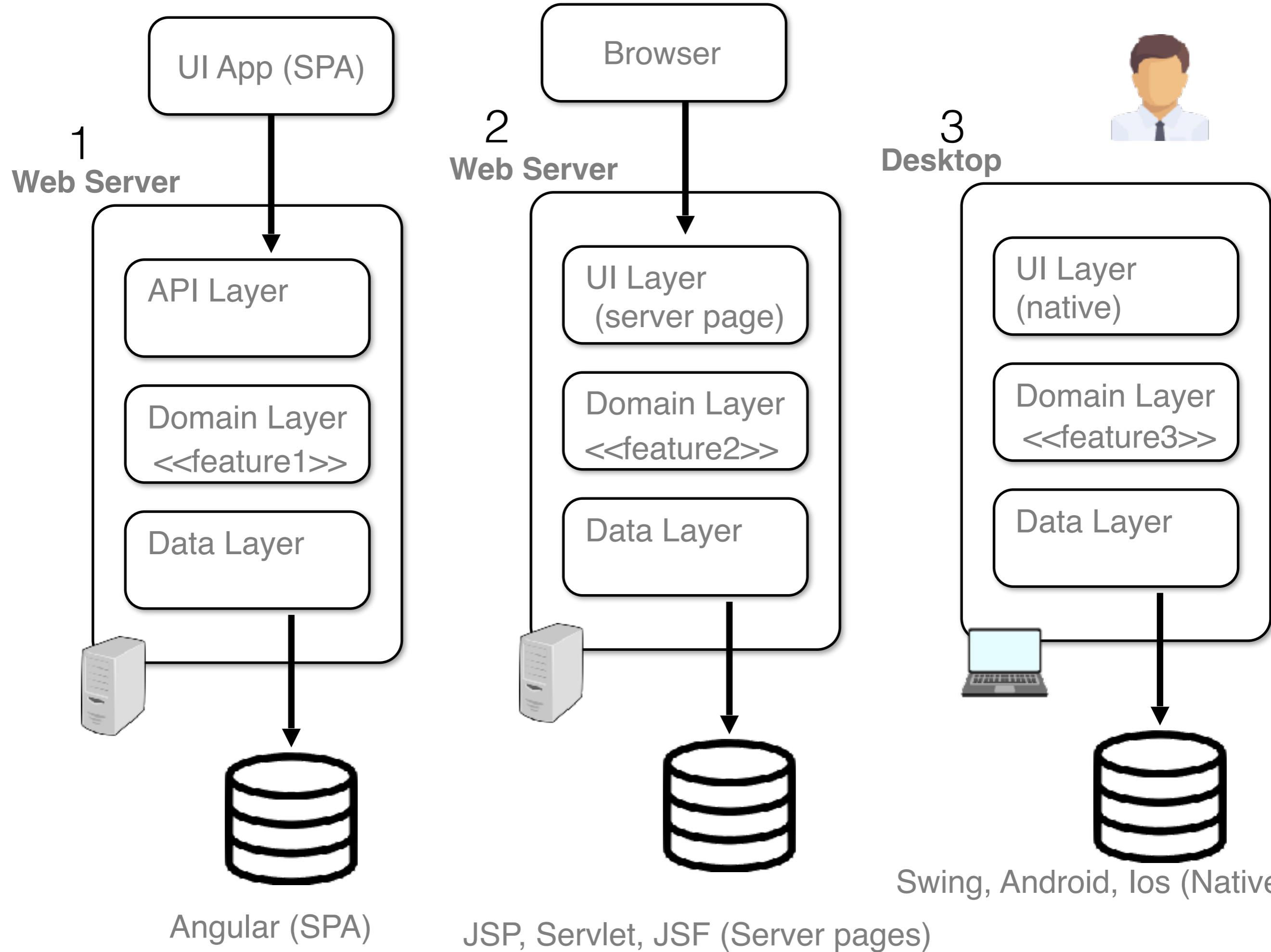


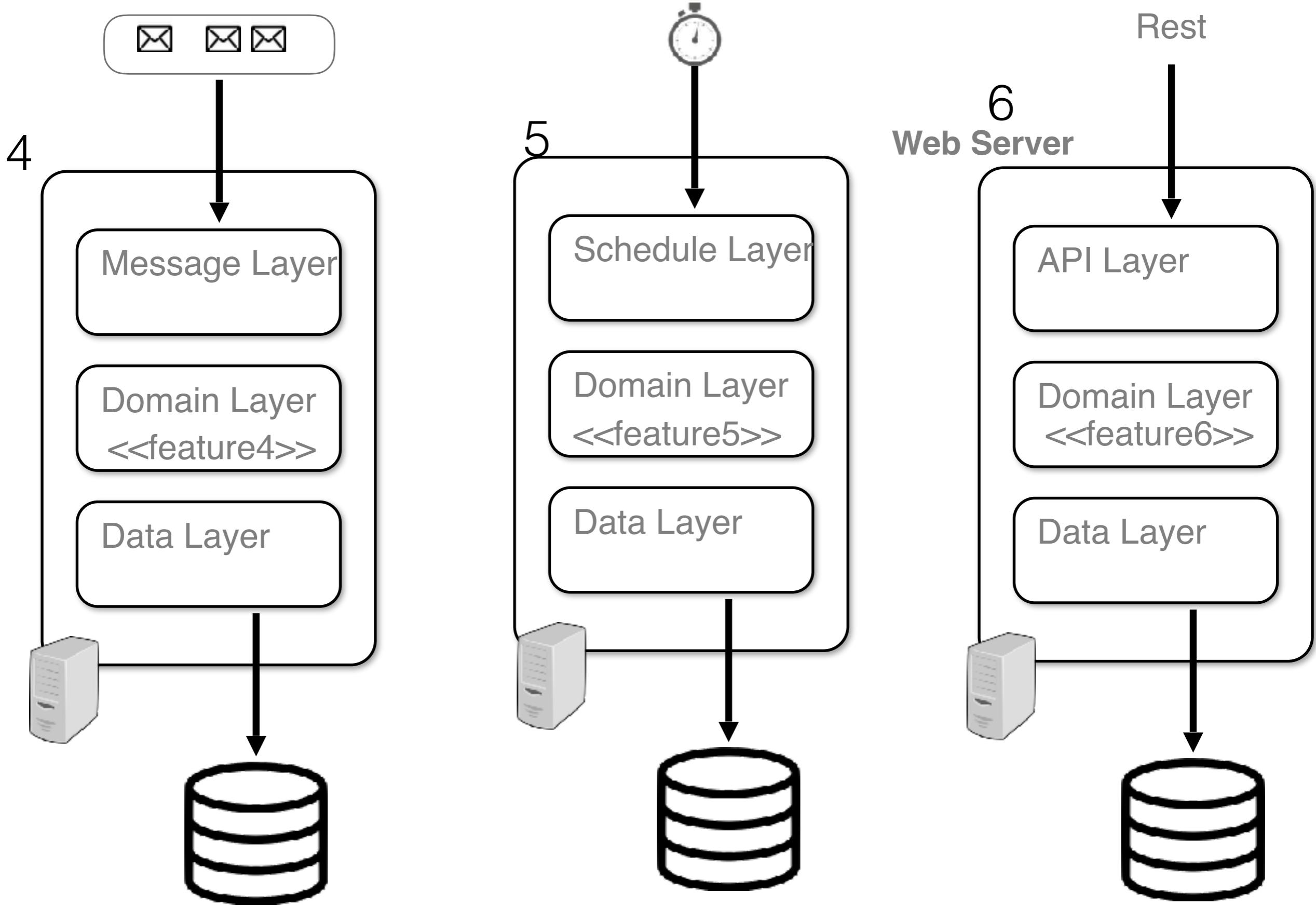
APP

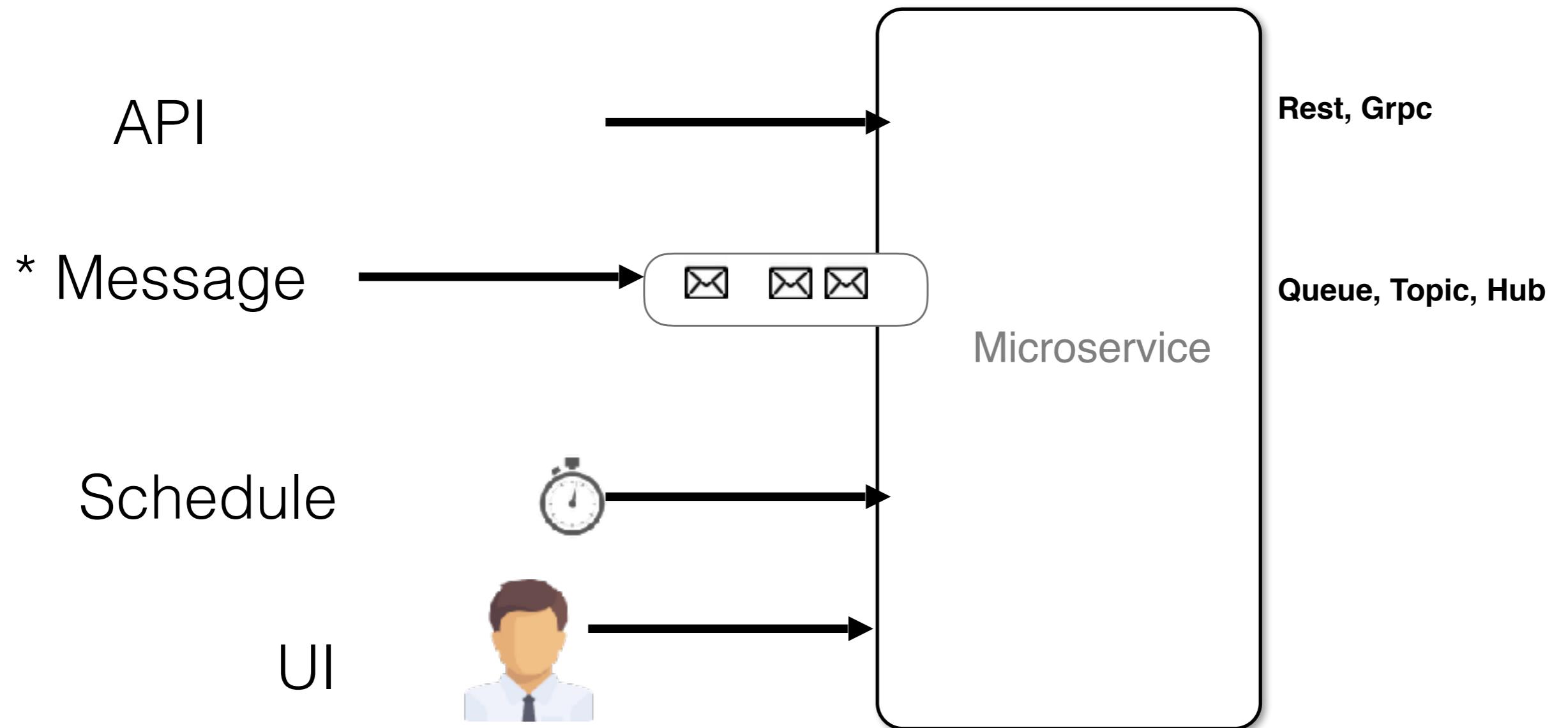
Types of Microservice

App

App



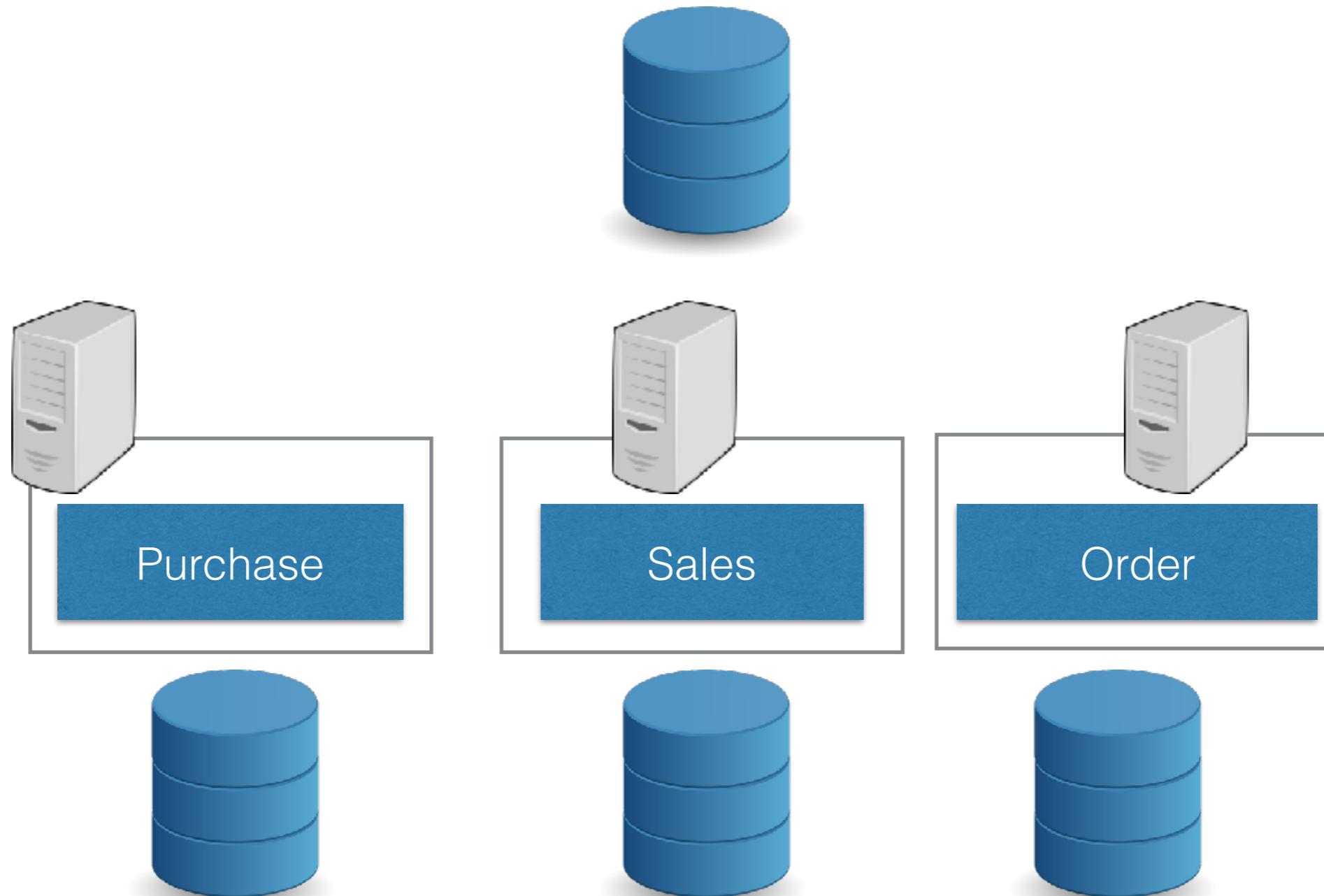
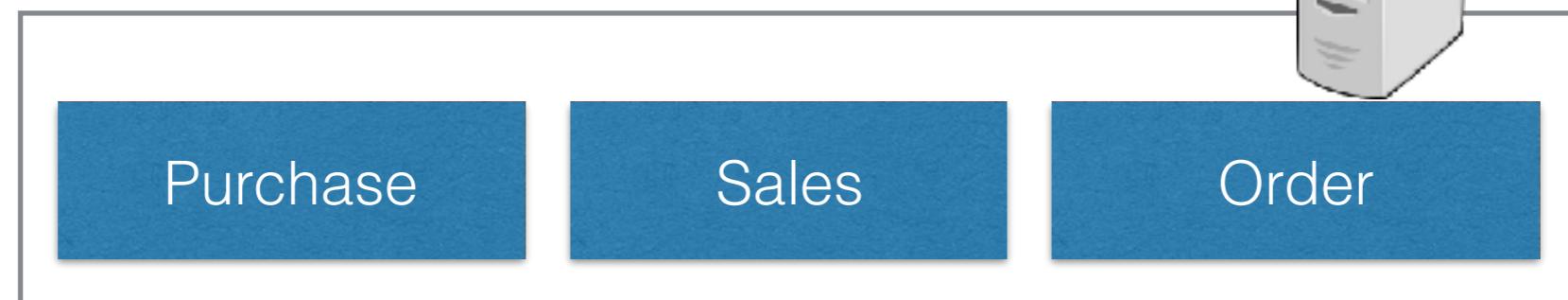




	Pros/	Solution
Development time	--	
Micro-service practices Learning Curve	--	
Resource Performance (CPU, Memory, I/O)	--	Grpc, protobuf, ...
Db Transaction Management	--	SAGA
Views / Report / Dash board/ join	--	Materialized View
Infra Cost	--	Containerization
First time Deployment effort	--	Automation
Debugging, Error Handling (End to End)	--	Distributed Tracing (Jaeger)
Integration Test		
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	+	

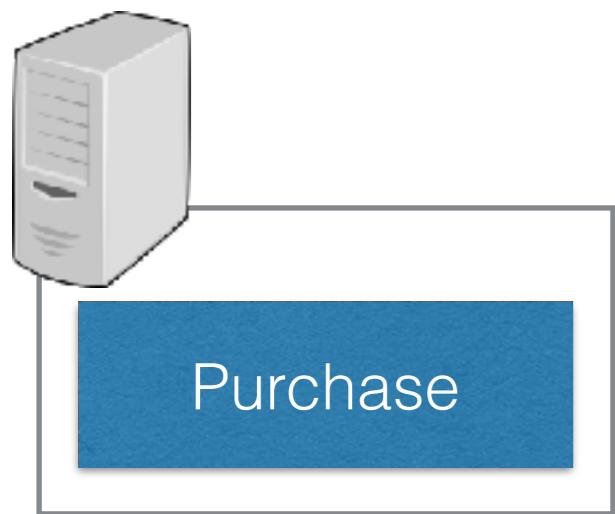
	Pros/ Cons	Solution
Development time	--	
Micro-service practices Learning Curve	--	
Resource Performance (CPU, Memory, I/O)	--	
Db Transaction Management	--	SAGA
Views / Report / Dash board/ join	--	Materialized View, CQRS
Infra Cost	--	Containerization
First time Deployment effort	--	Automation (chef, puppet)
Debugging, Error Handling (End to End)	--	Distributed Tracing (Jaeger, ...)
Integration Test	---	
Log Mgmt (debug/ error)	--	Centralize - ELK, EFK, Splunk
Config Mgmt	--	Centralize - Consul
Authentication (who)	--	Centralize - OAuth2, SSO, SAML, ...
Authorization (what can they do)	--	Centralize - ?
Audit Log mgmt (who accessed what)	--	Centralize - Event Sourcing
Monitoring / Alerting	--	Centralize
Data Security and Privacy (transit, storage)	--	
Build Pipeline (CI)	--	Automation
Agile Architecture (Agility to change)	+++	
Feature Shipping (Agility to ship)/ CD	++	
Scalability (volume - request, data,	+	
Availability	+	
Ability to do Polygot	+	
Fault Isolation	+	

Inventory Application



1 phase commit

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



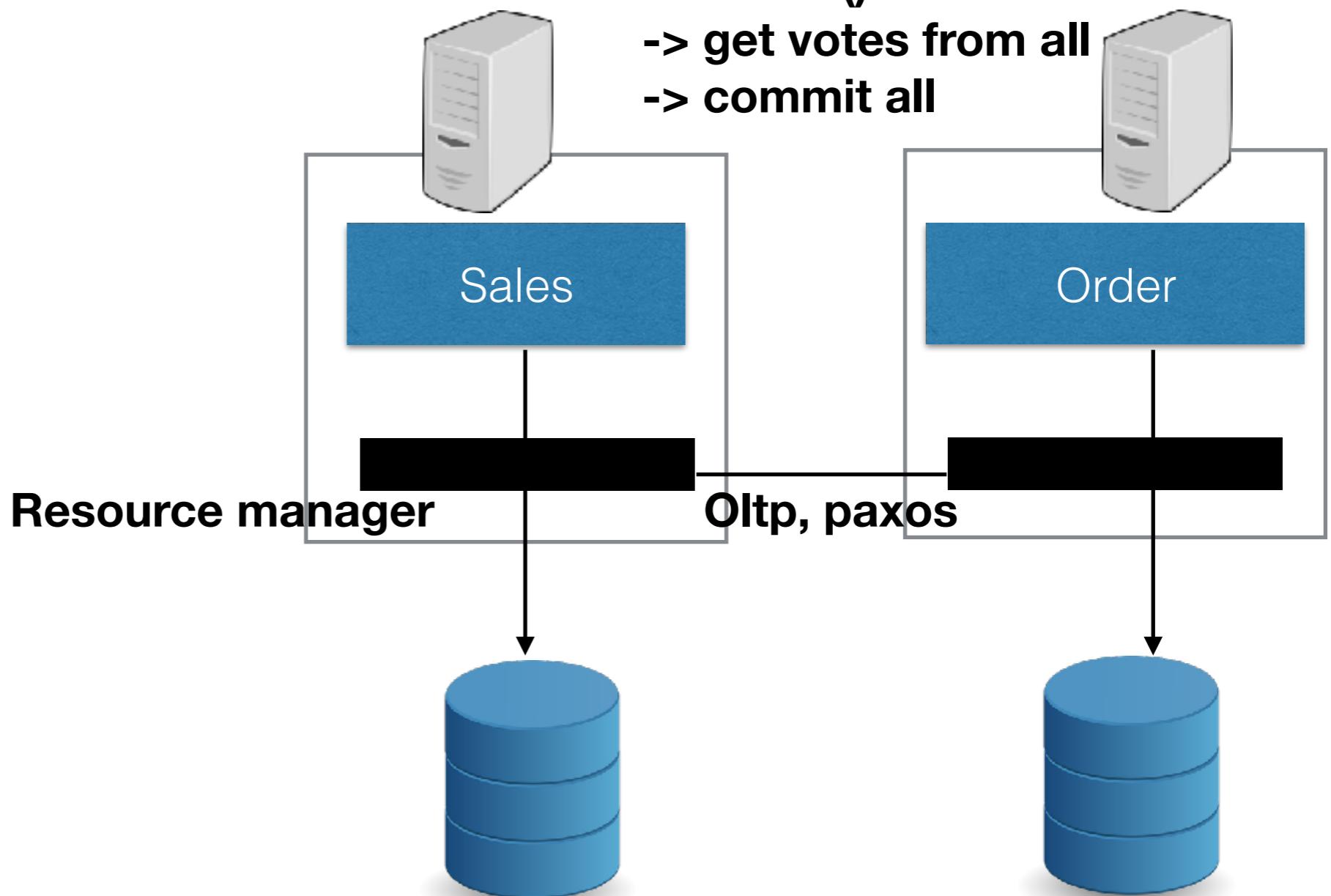
2 phase commit

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

...

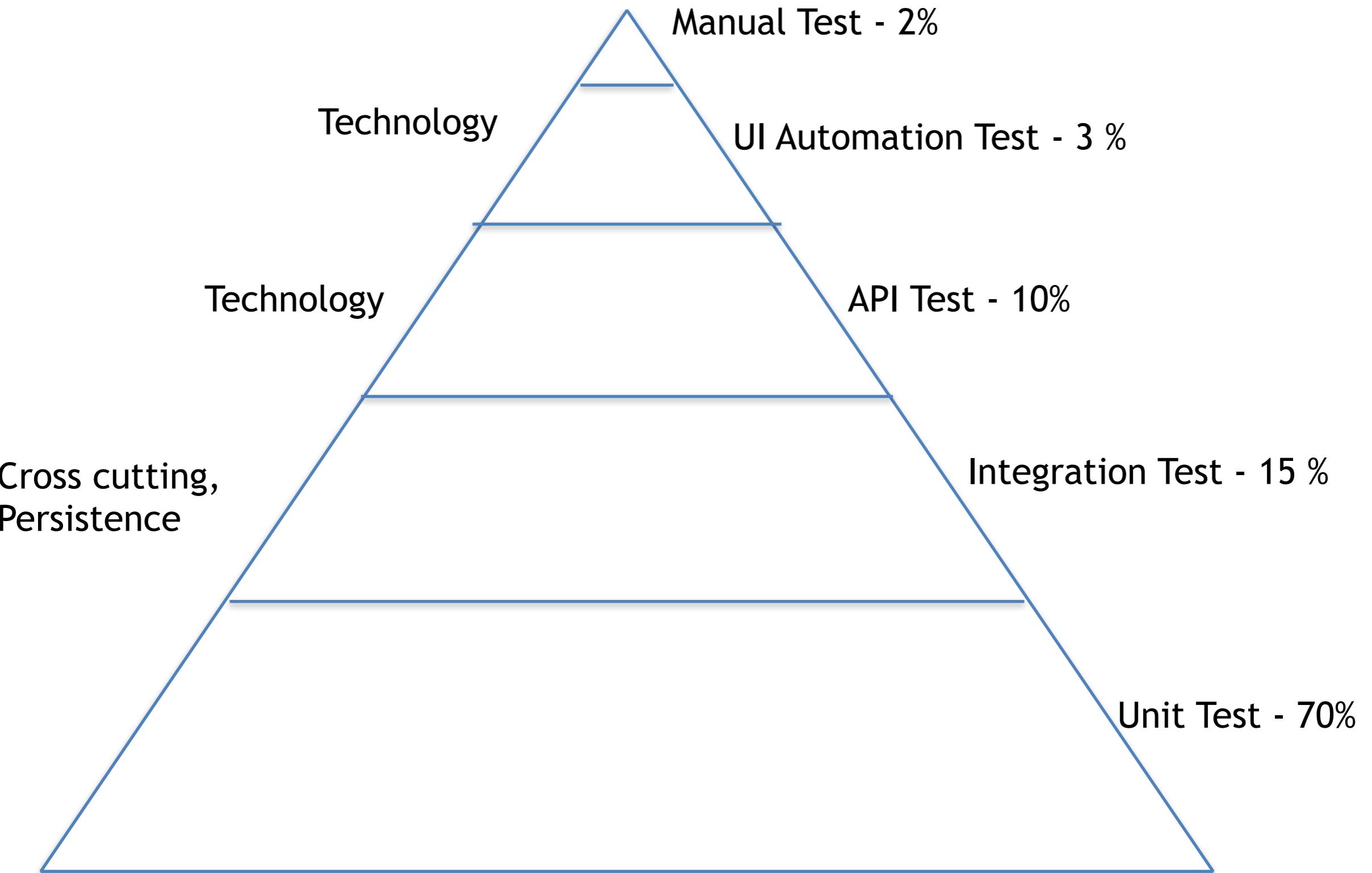
```
rm.Commit()
```

- > get votes from all
- > commit all



Decentralize Domain
Centralize Tech

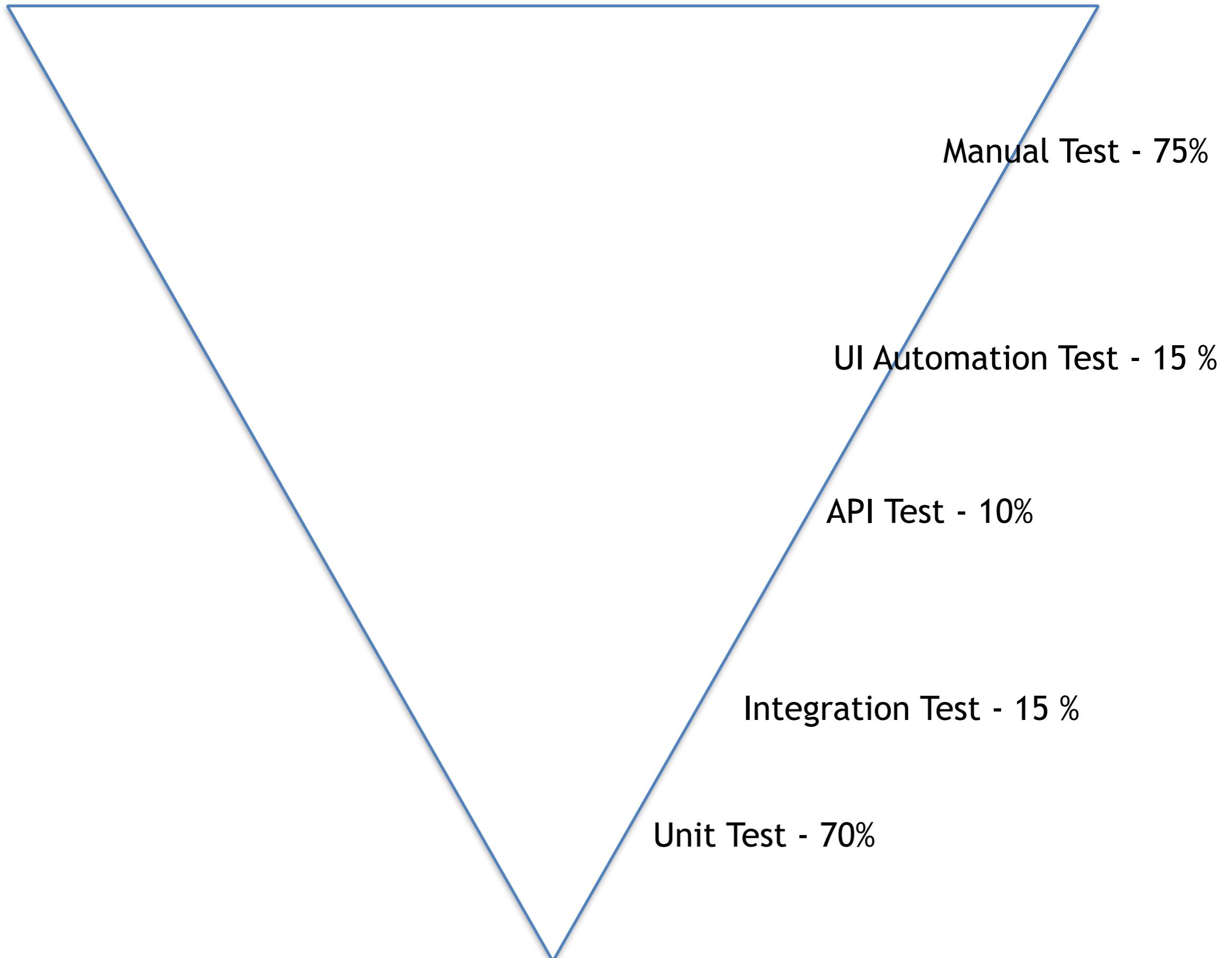
Pyramid test

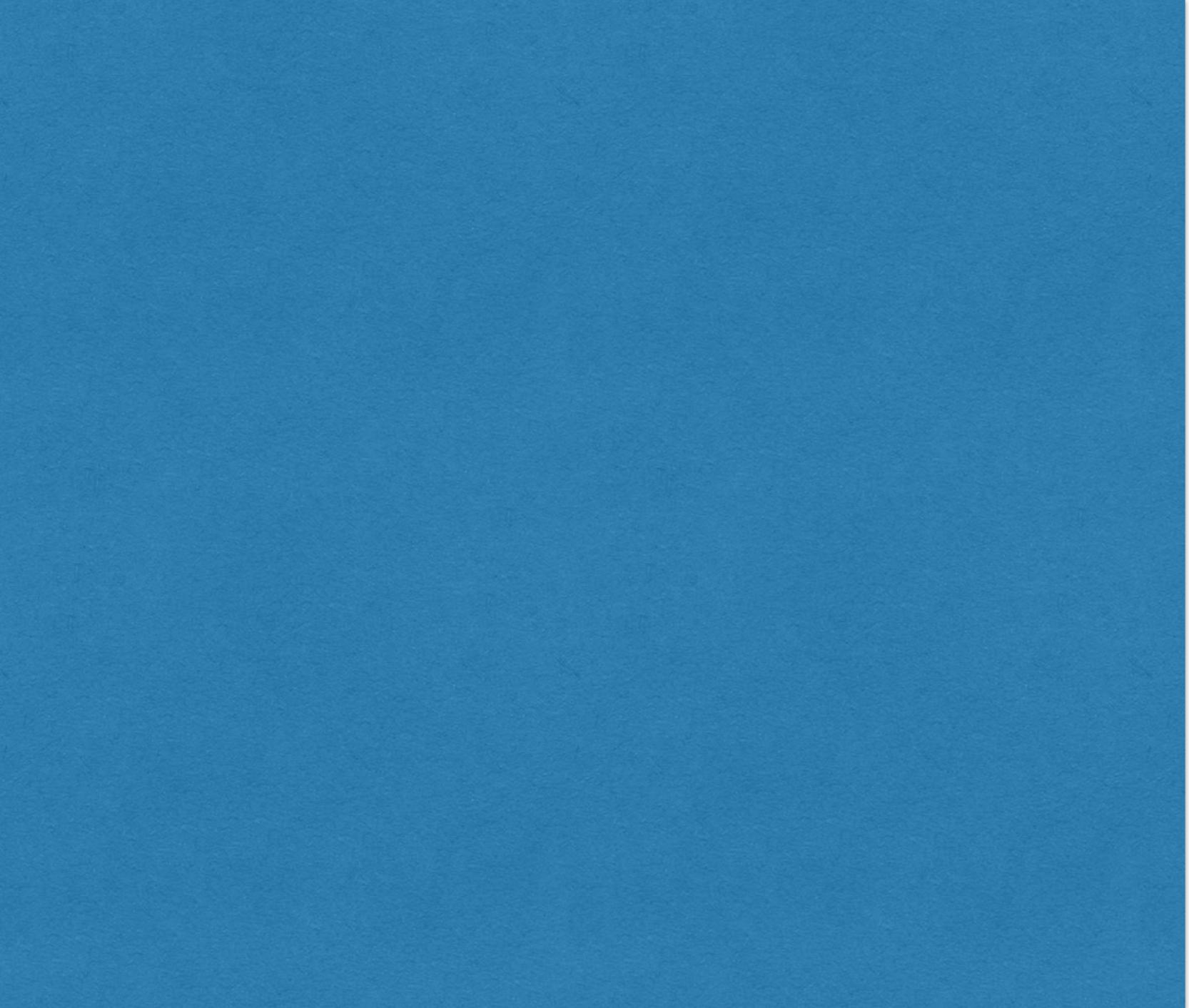


Unit Test

- Documentation for Code
- Design Code
- Regression
- Find Bugs

Its working don't touch it





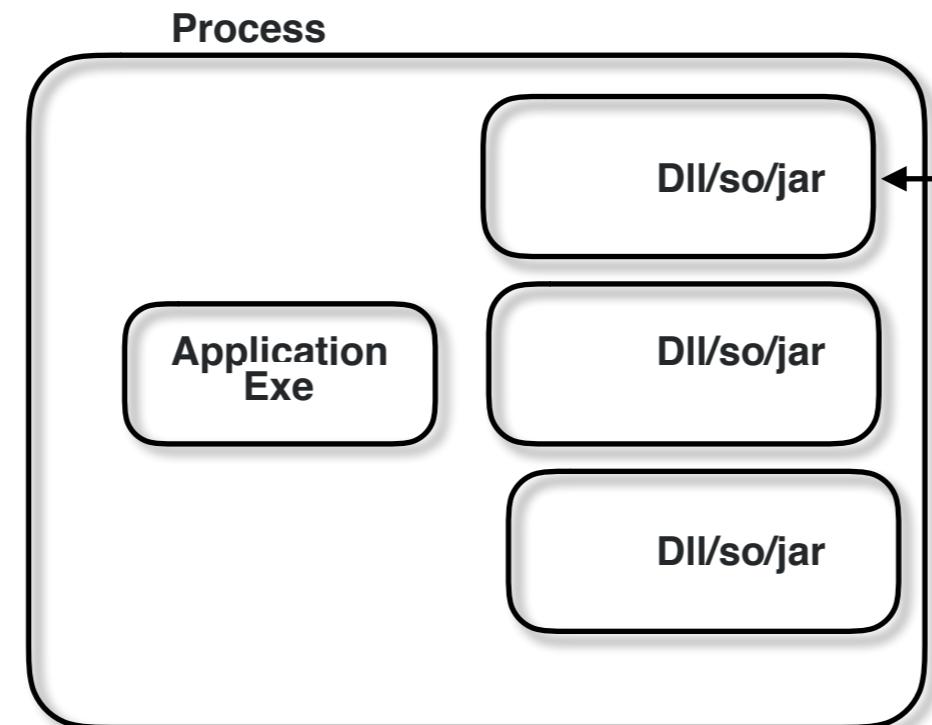
Manual Test - 2%

UI Automation Test - 3 %

API Test - 10%

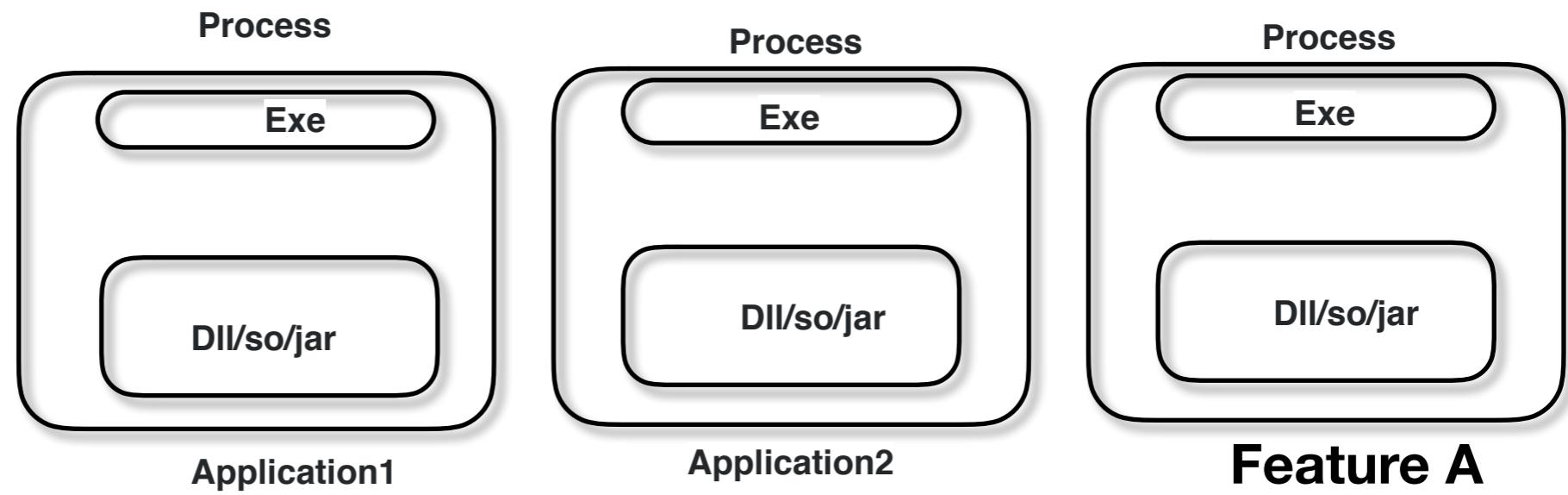
Integration Test - 15 %

Unit Test - 70%



**Runtime
monolithic**

In process
comp

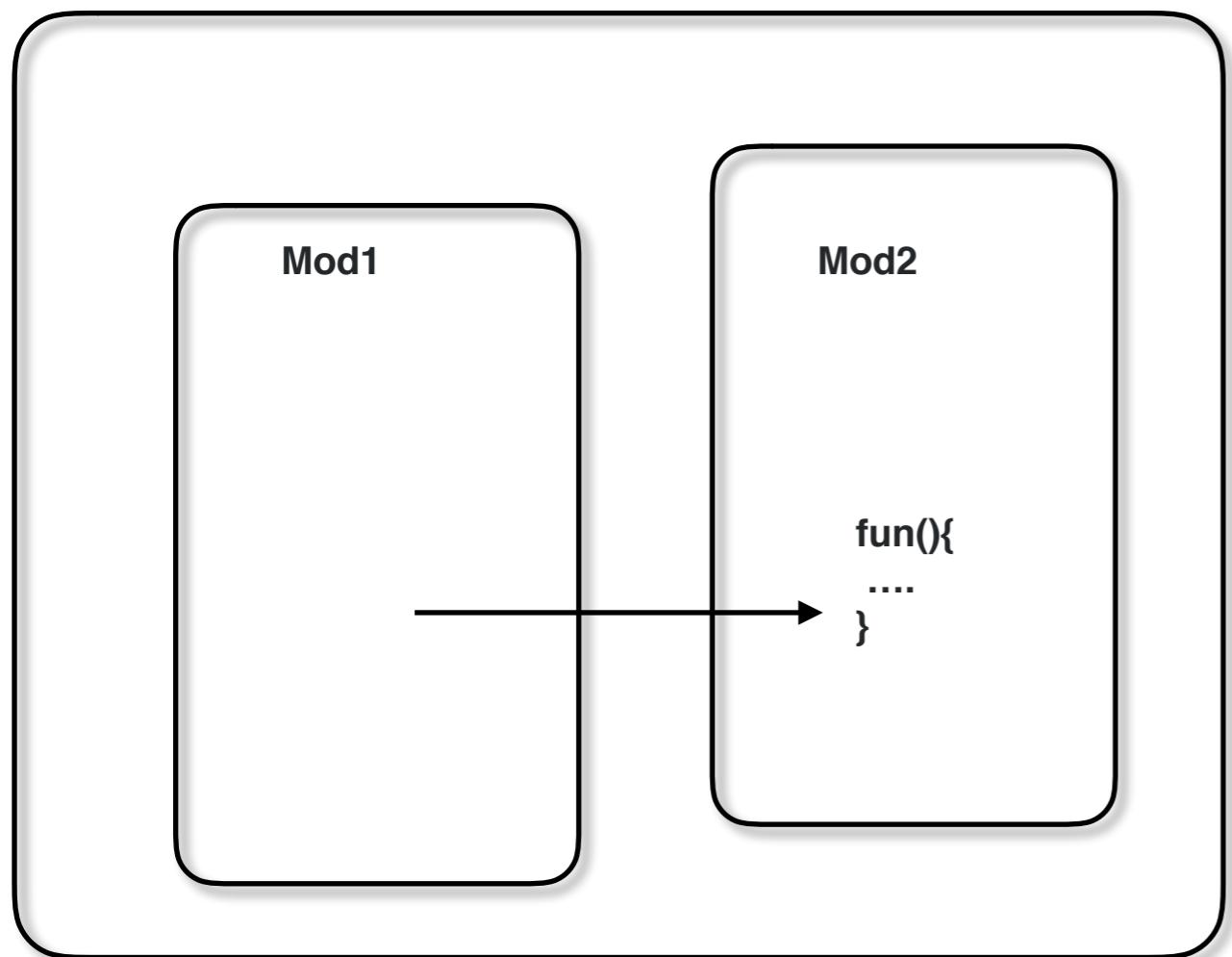


Micro Application

1. Performance

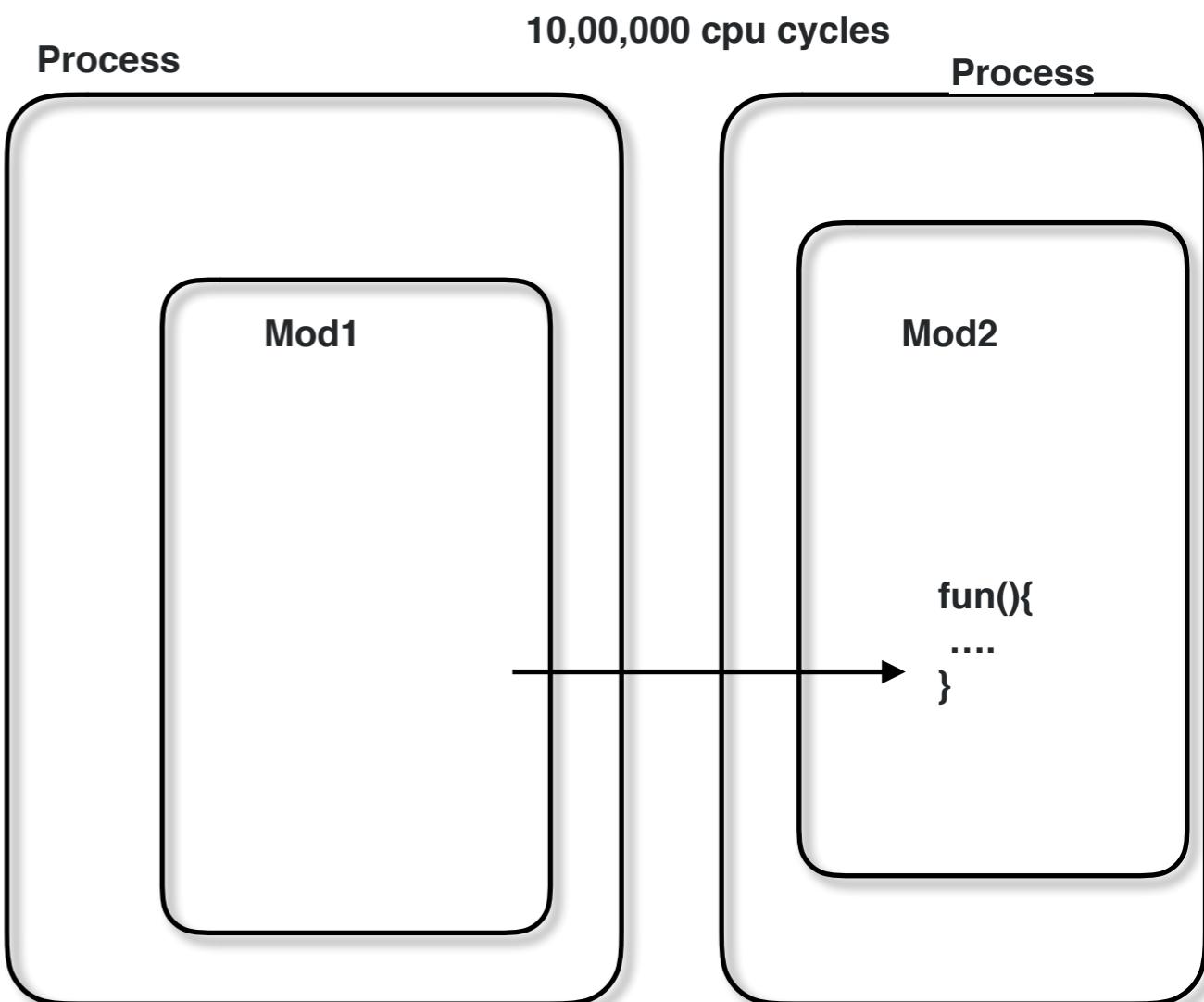
Operation	Cpu Cycles
• 10 + 12	3
• Calling a in-memory Method	10
• Create Thread	2,00,000
• Destroy Thread	1,00,000
• Database Call	40,00,000
• Distributed Fun Call	20,00,000
Write to disk	10,00,000

Process



10 cpu cycles

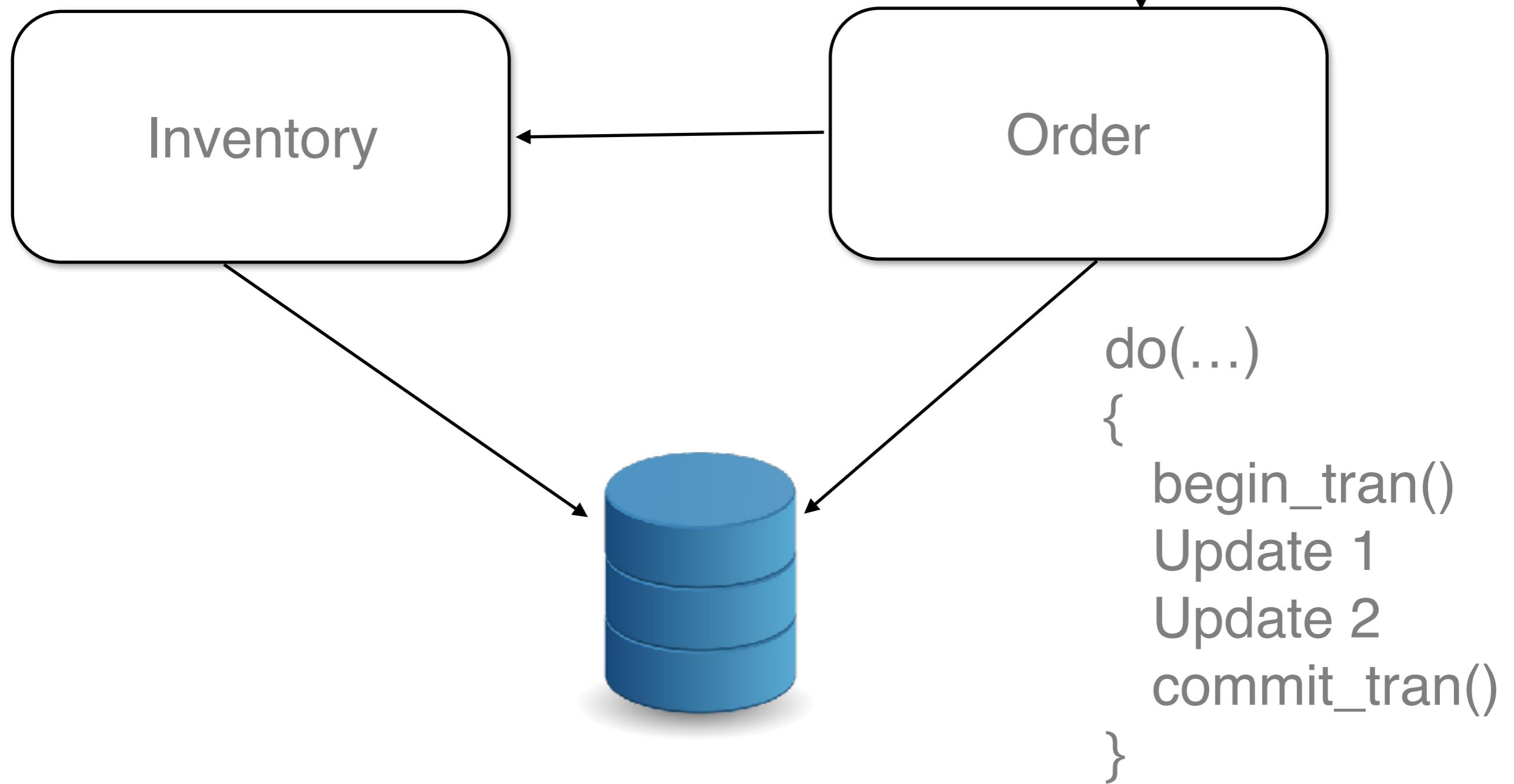
Process



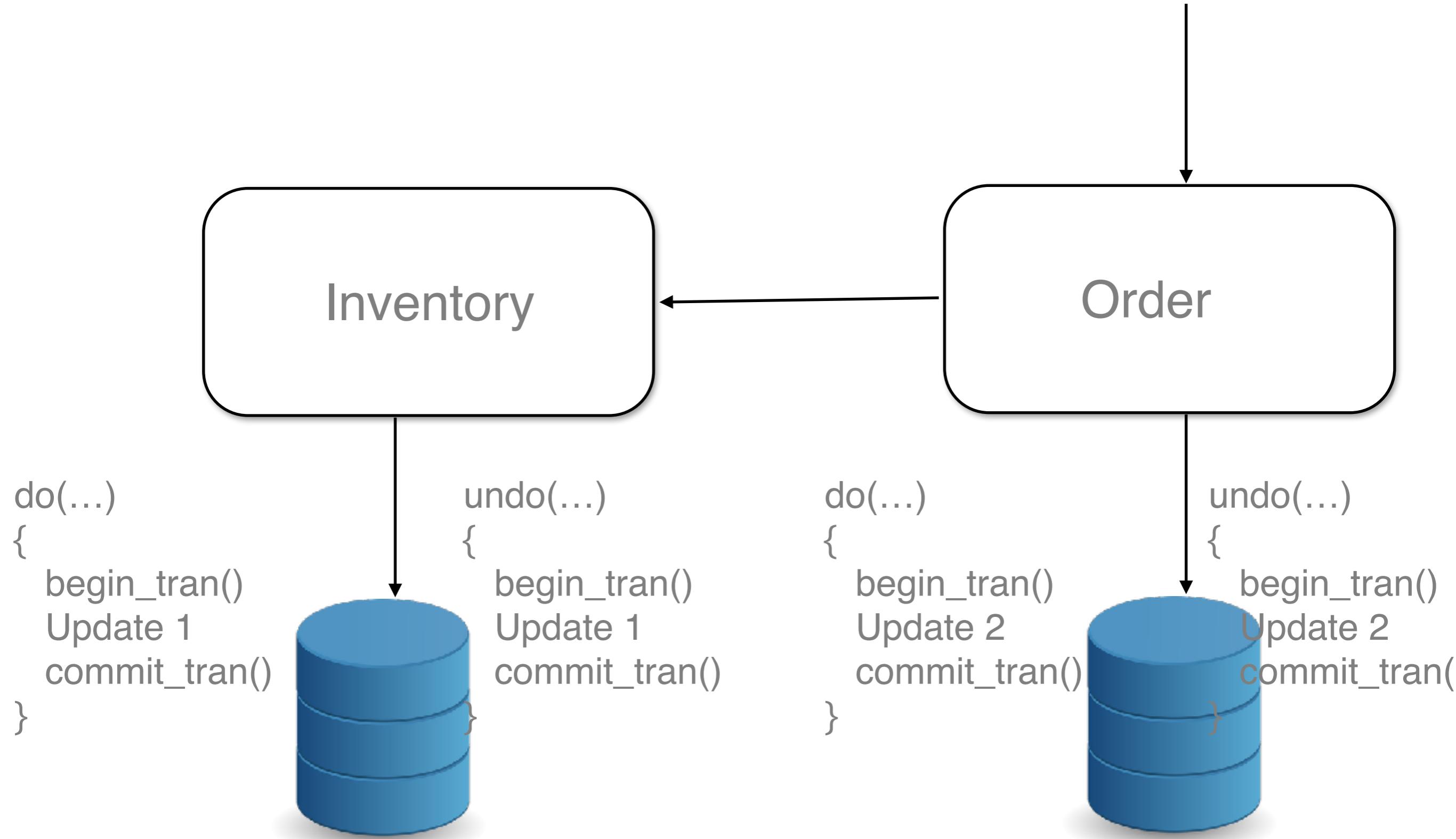
10,00,000 cpu cycles

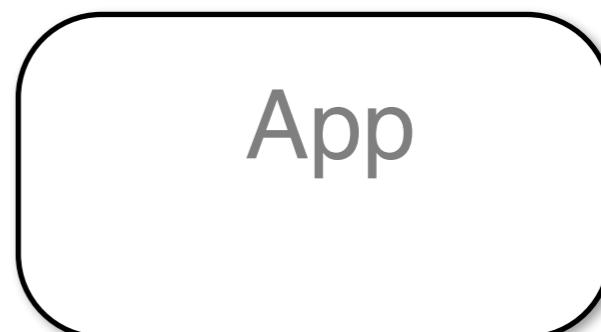
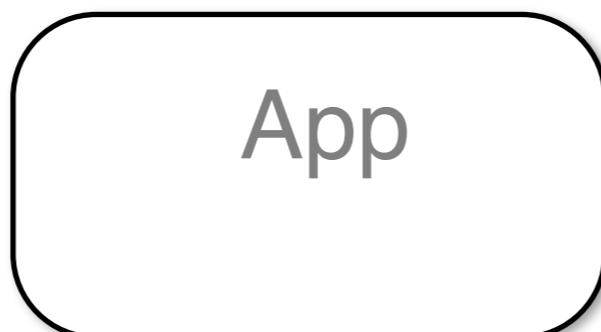
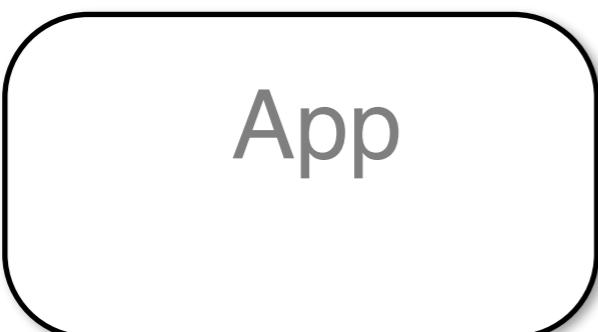
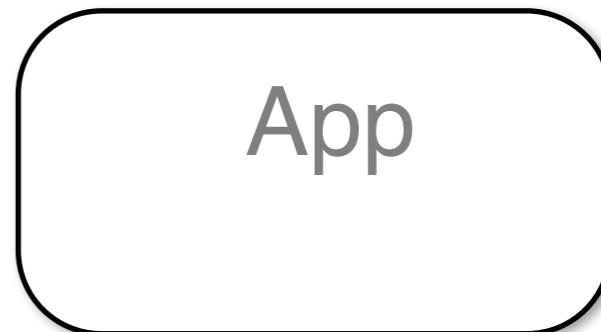
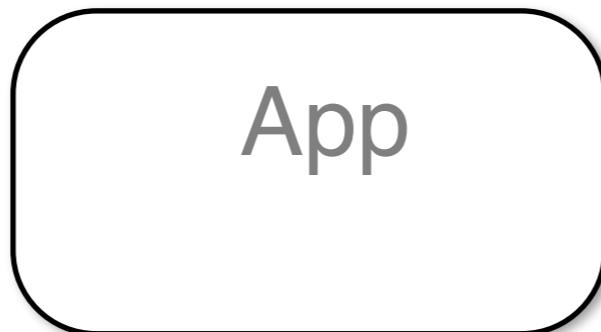
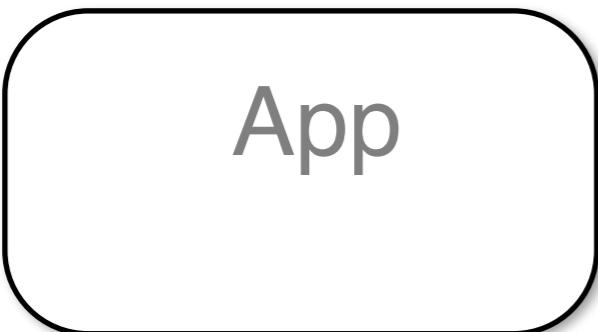
Process

2. Db transaction



2. Db transaction





App

Mod

Mod

Mod



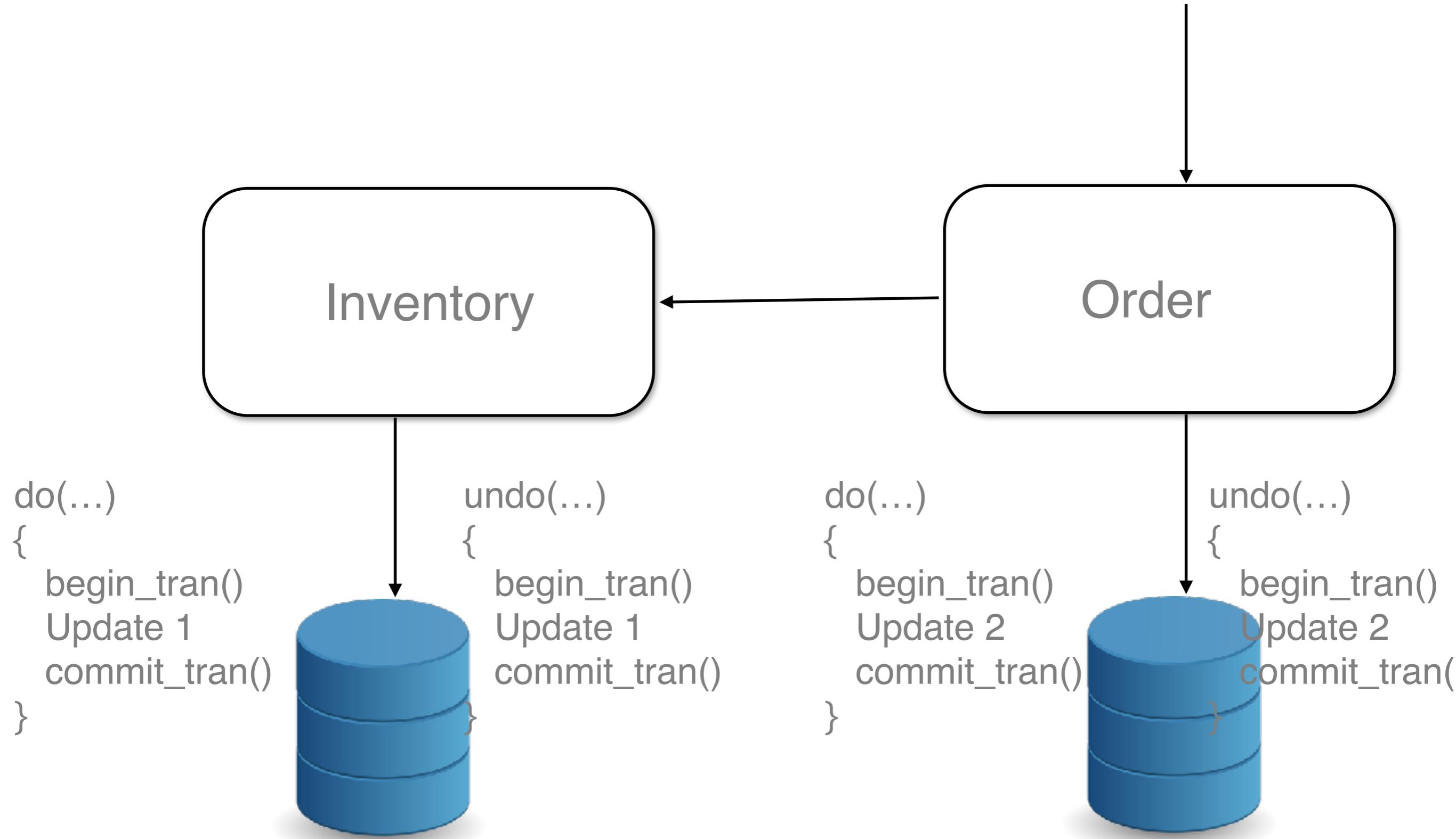
App

App

App

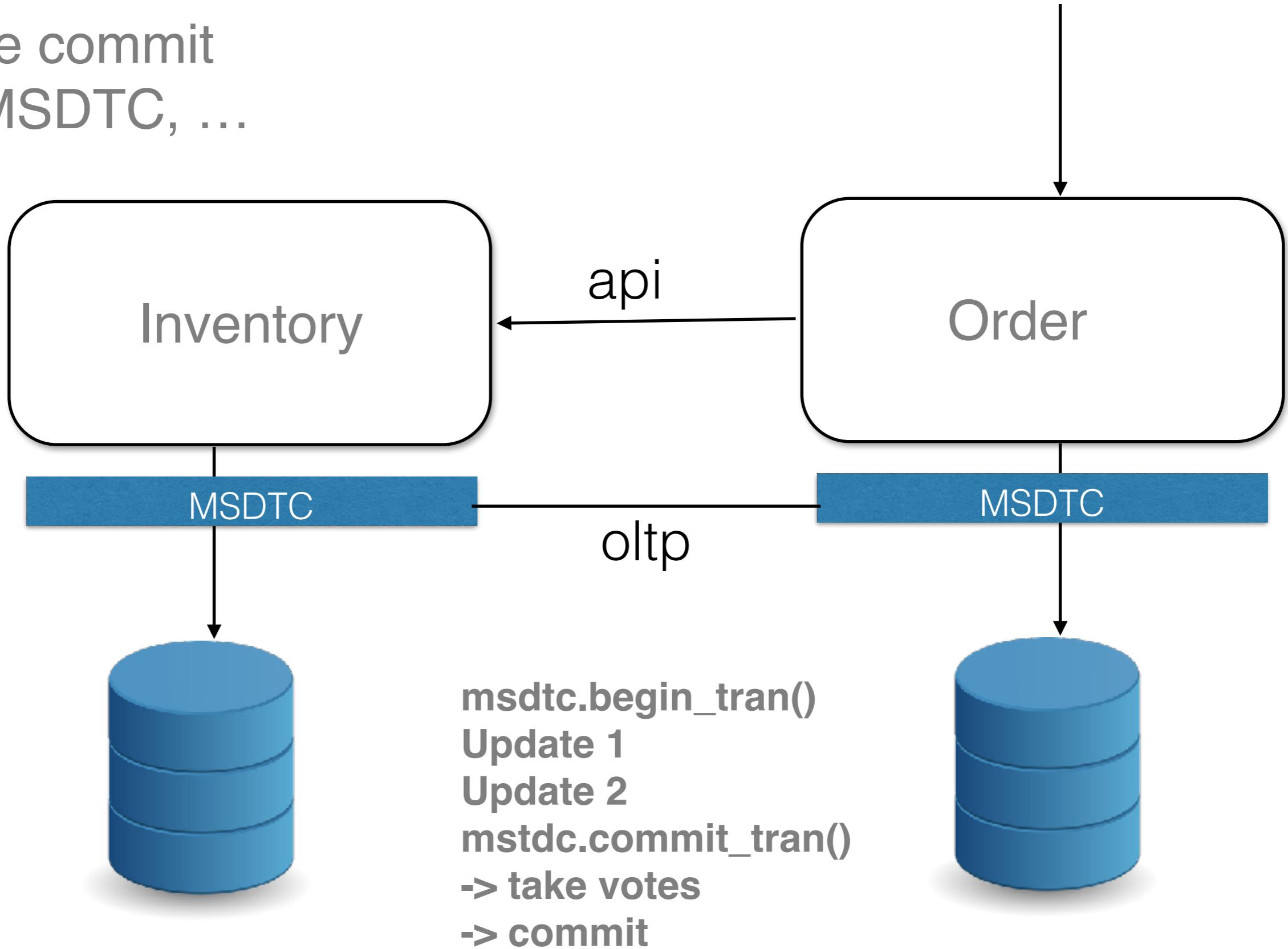


2. Db transaction

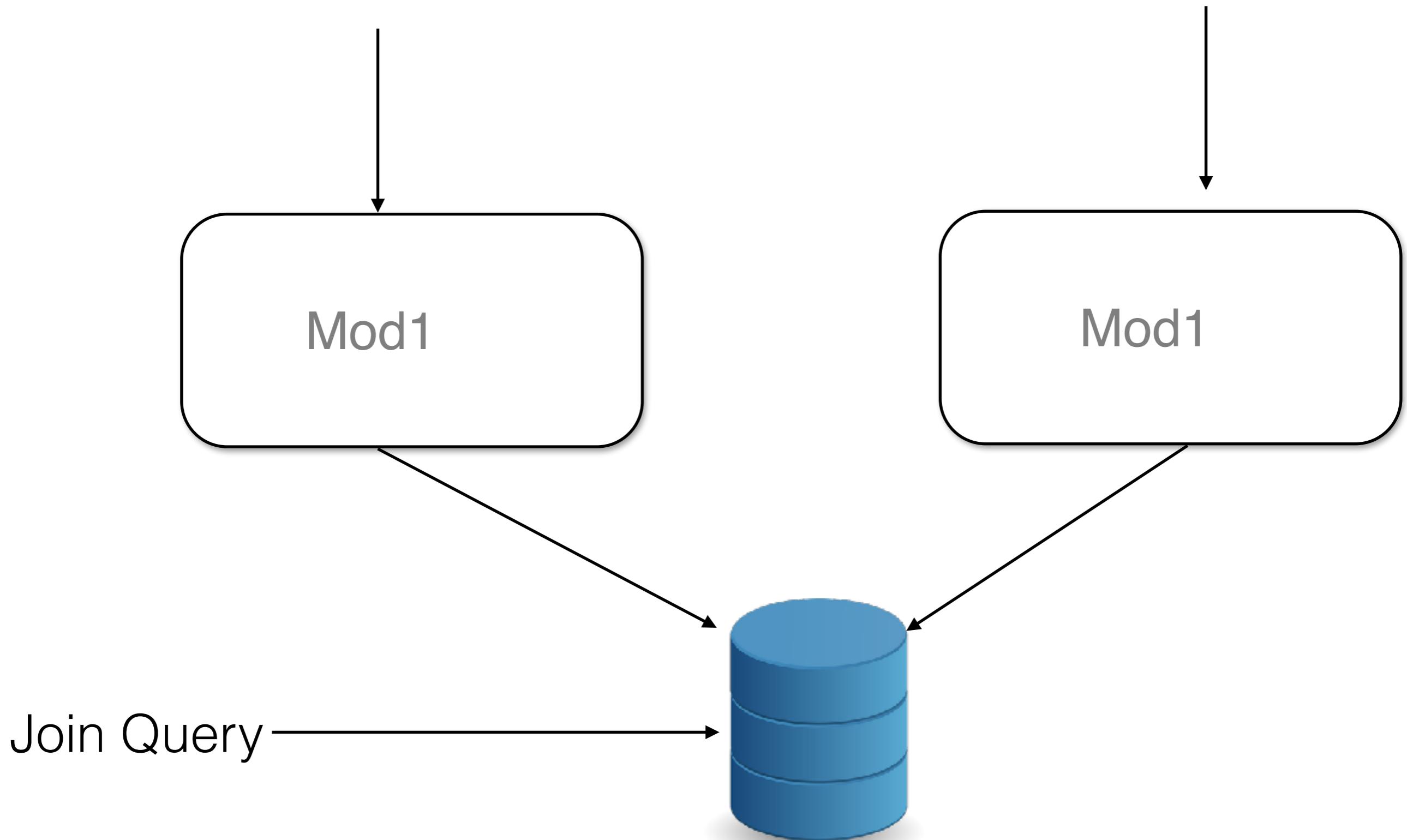


2. Db transaction

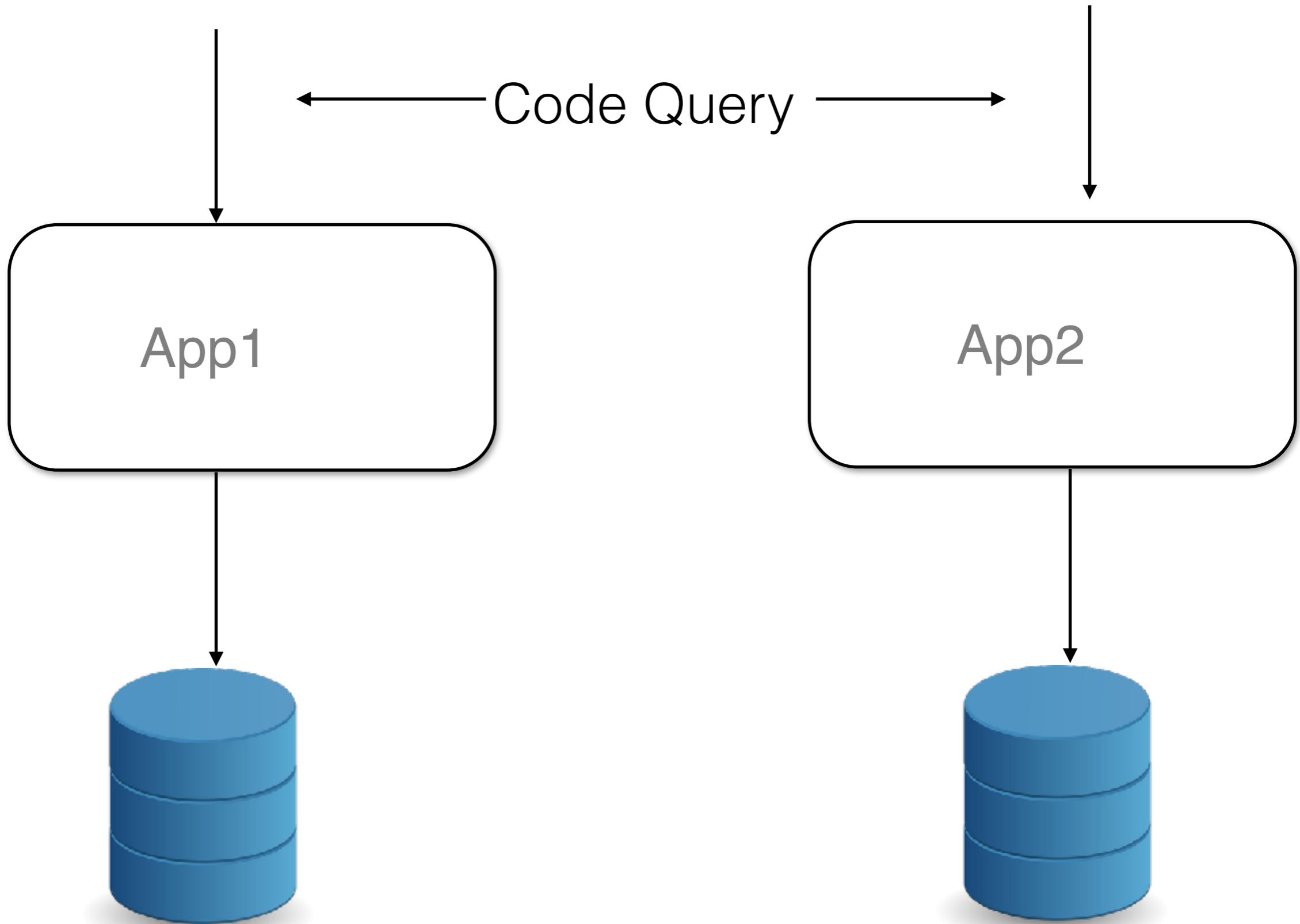
2 phase commit
JTX , MSDTC, ...



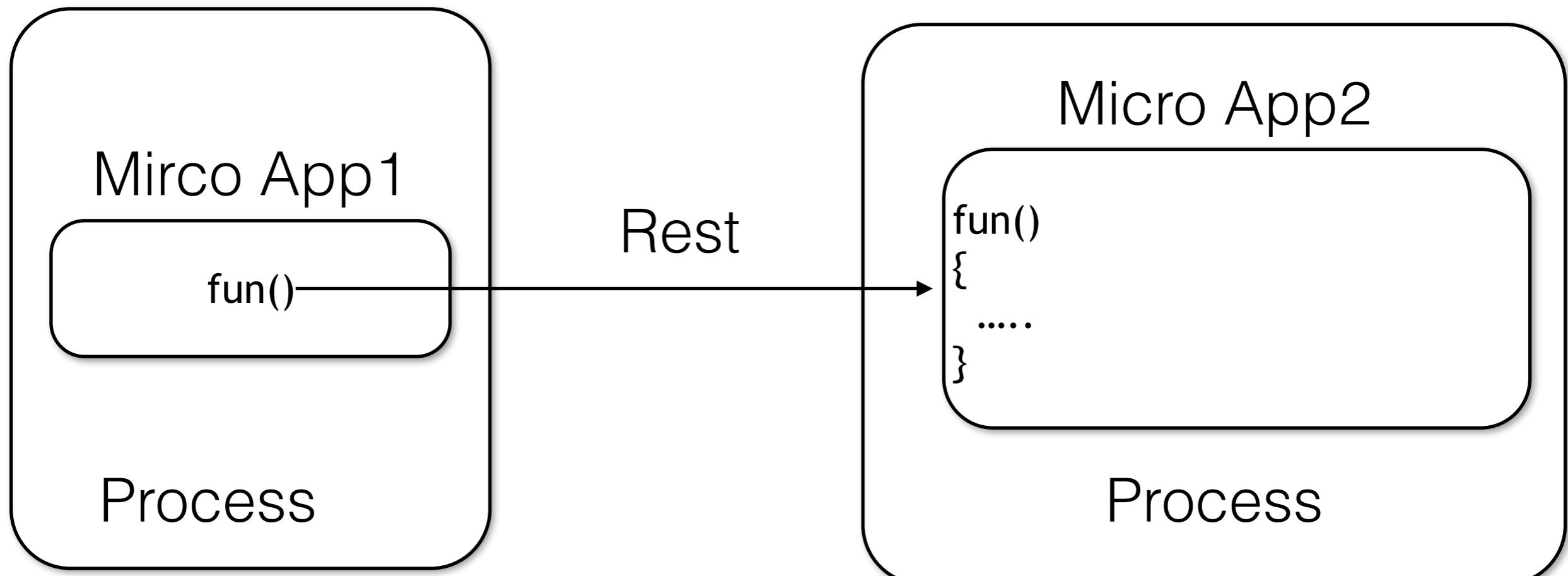
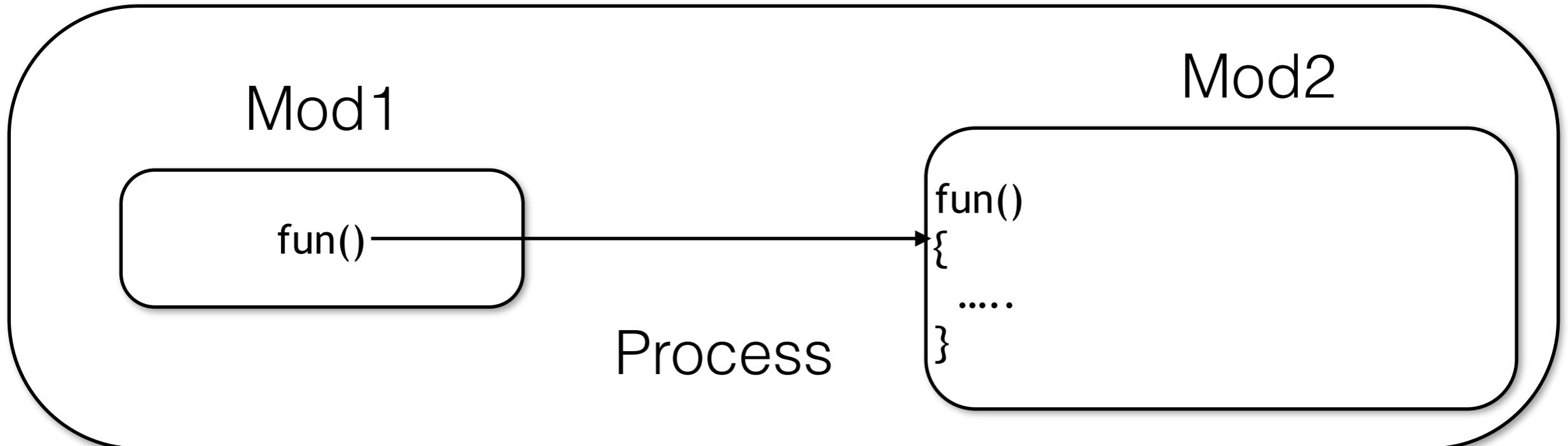
3. Db query



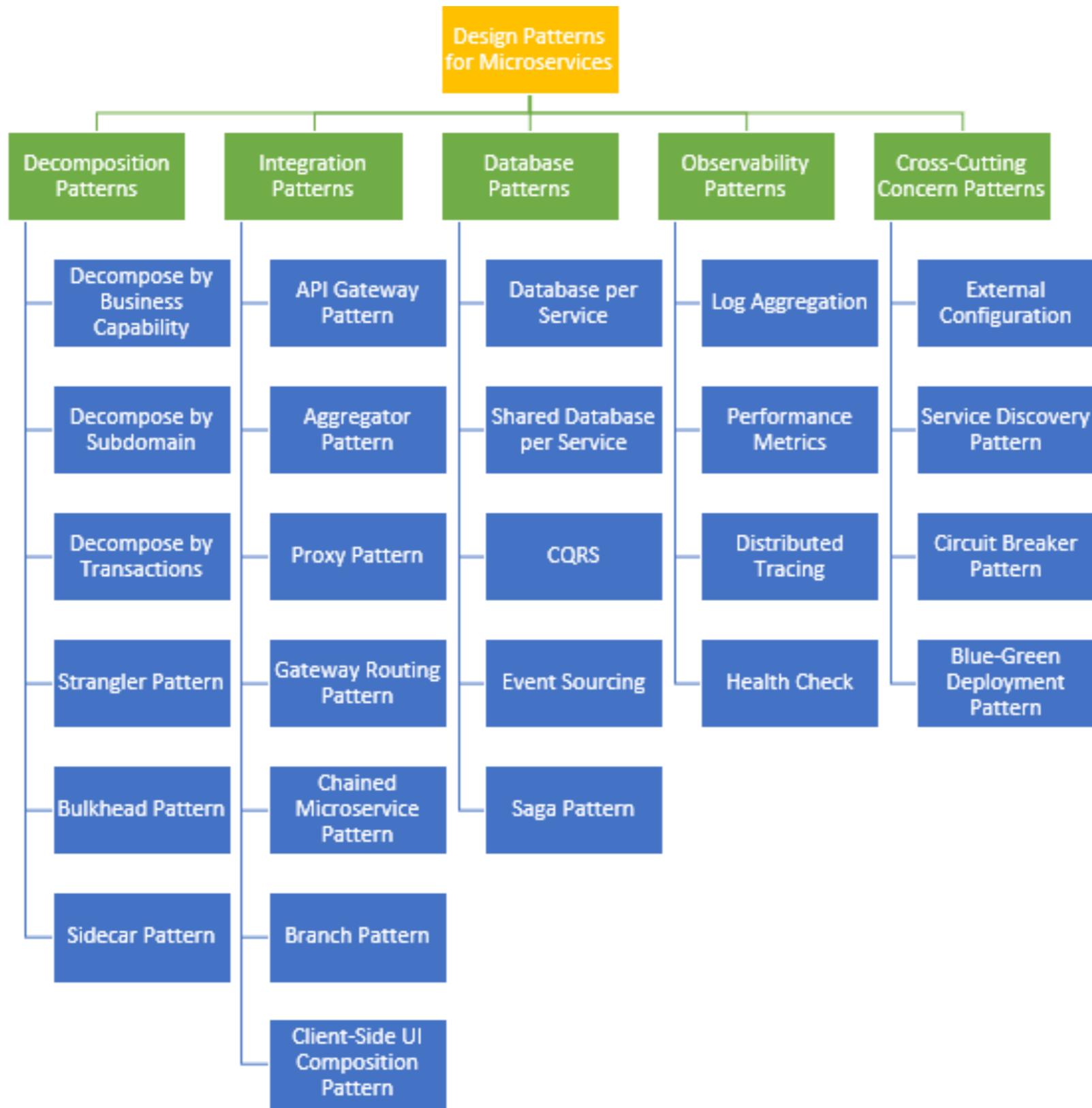
3. Query



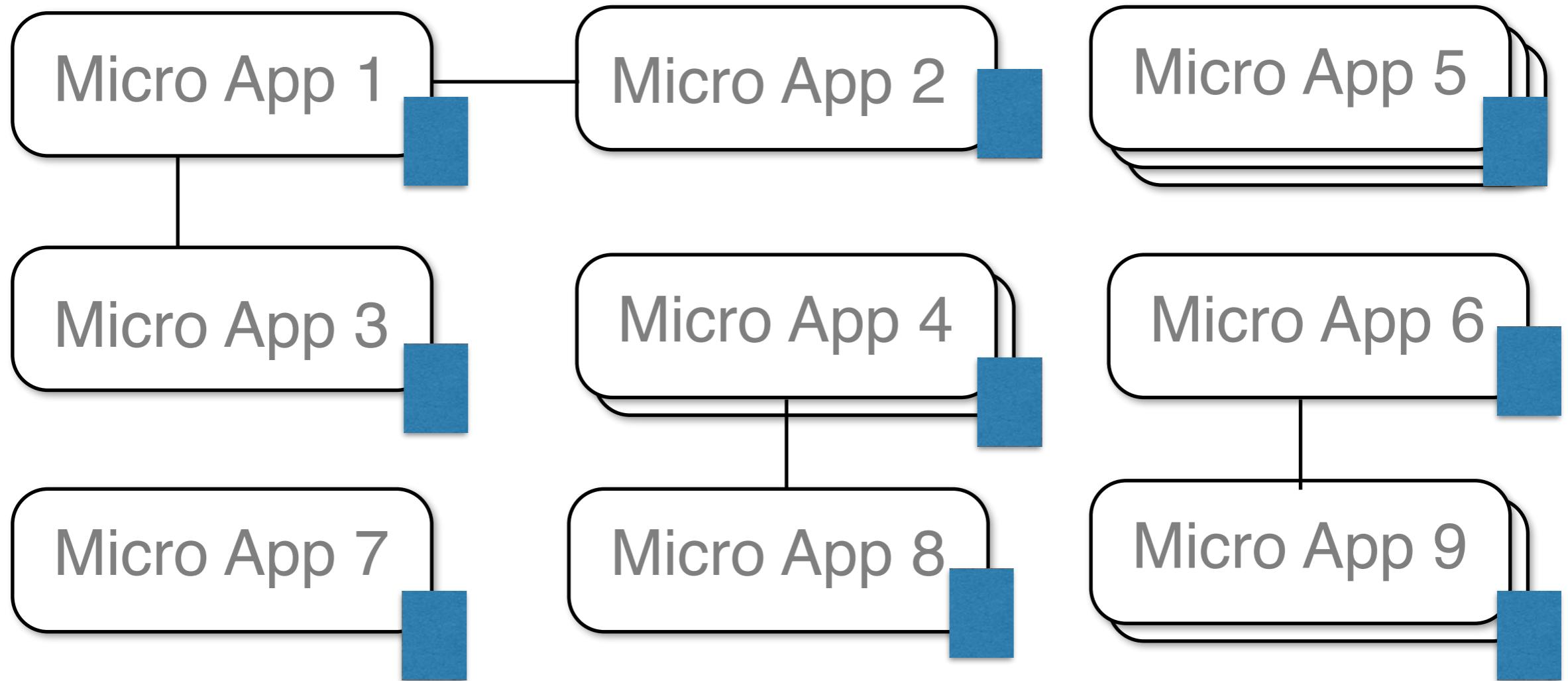
4. Development time



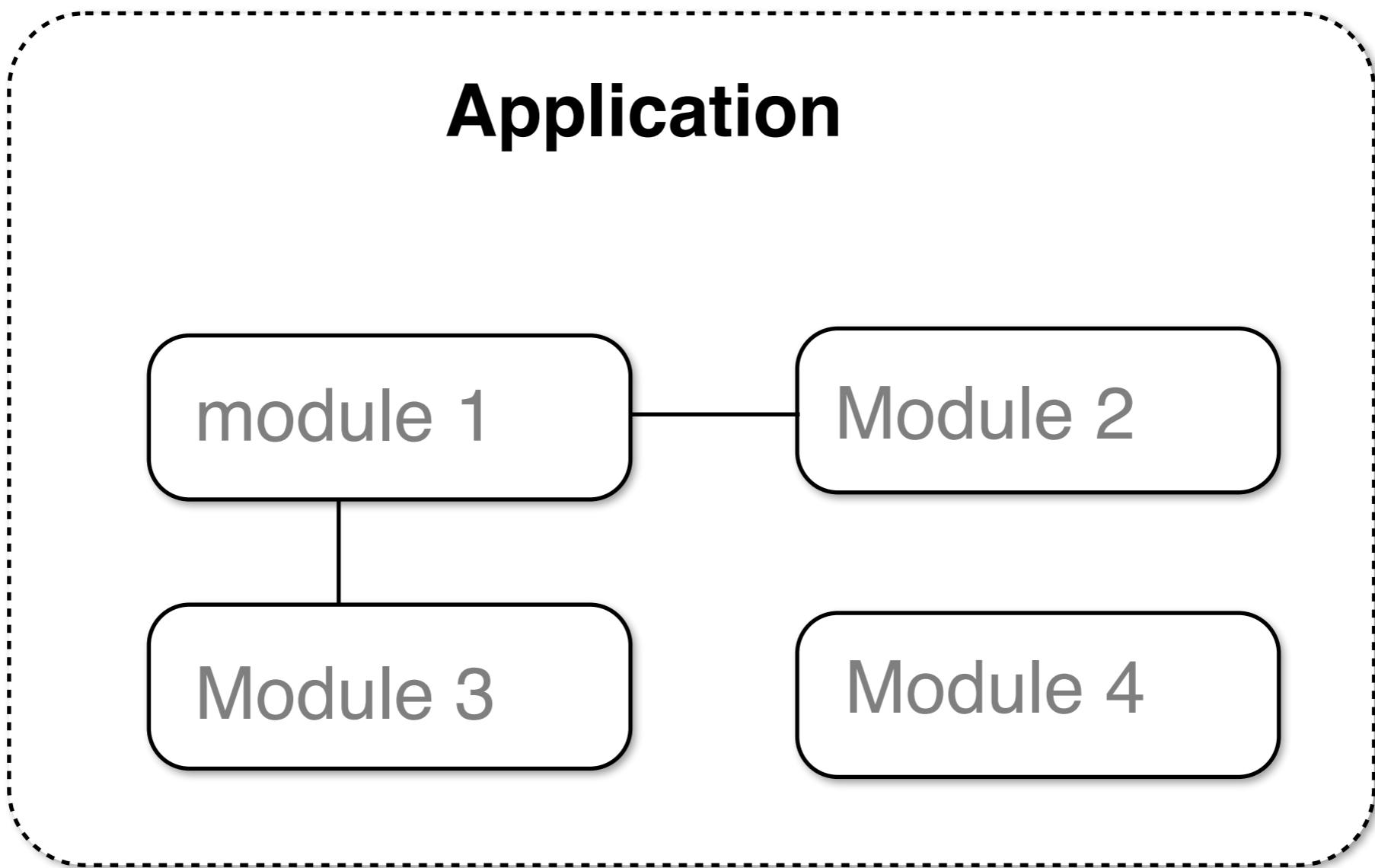
5. Learning Curve



6. Infra Cost



7. Debugging



Dev & Ops Teams



Log Data

Web Logs
App Logs
Database Logs
Container Logs

Metrics Data

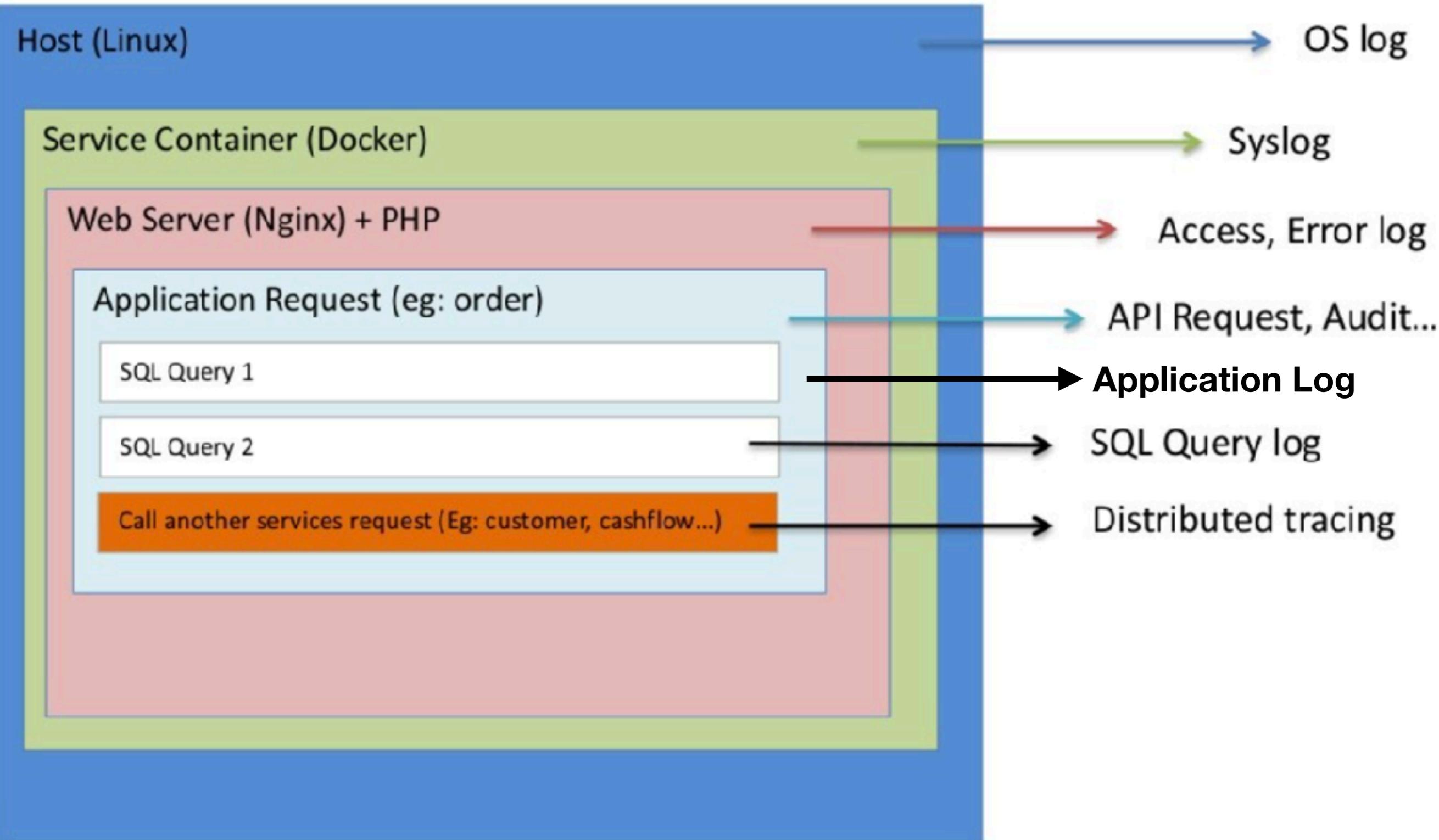
Container Metrics
Host Metrics
Database Metrics
Network Metrics
Storage Metrics

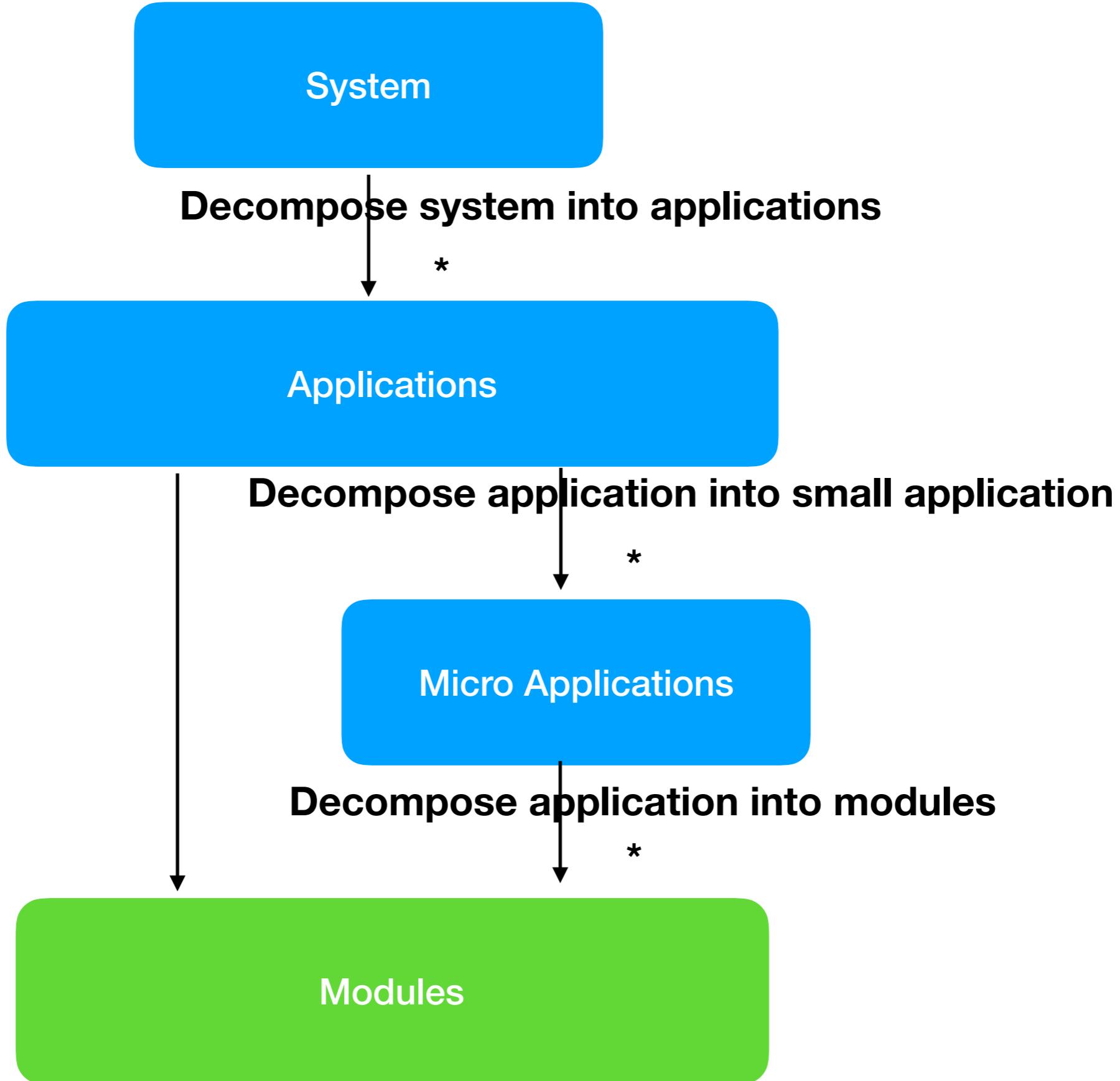
APM Data

Real User Monitoring
Txn Perf Monitoring
Distributed Tracing

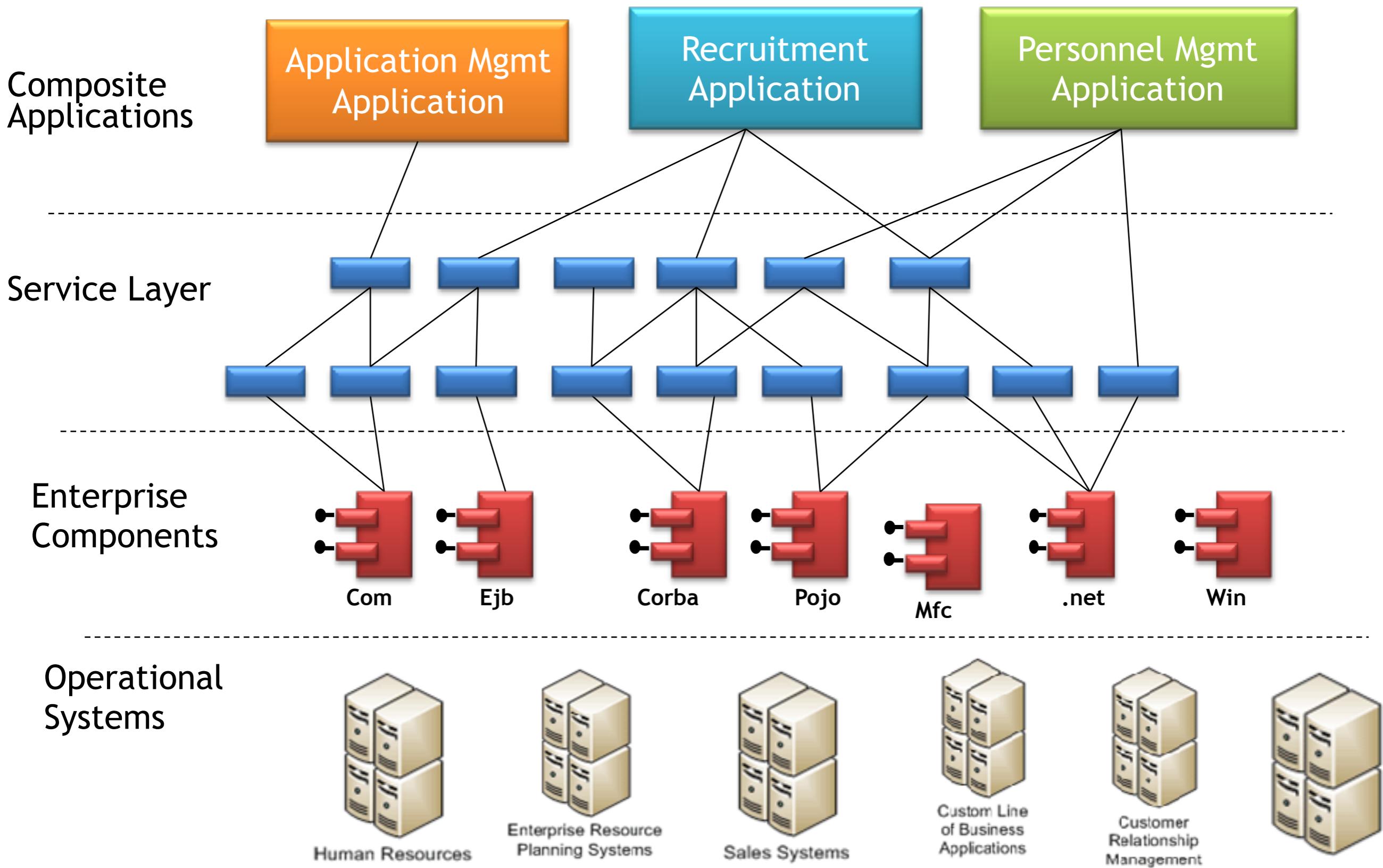
Uptime Data

Uptime
Response Time

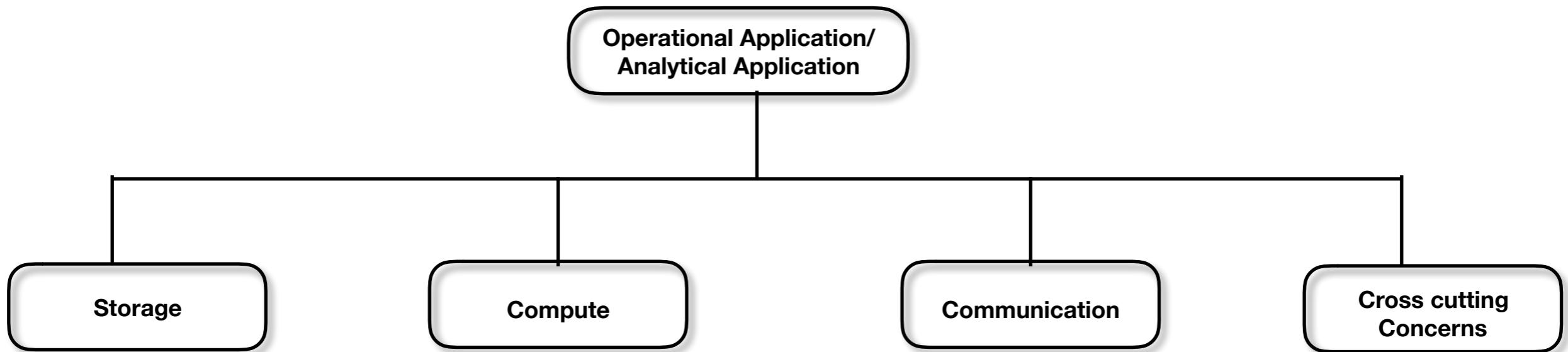


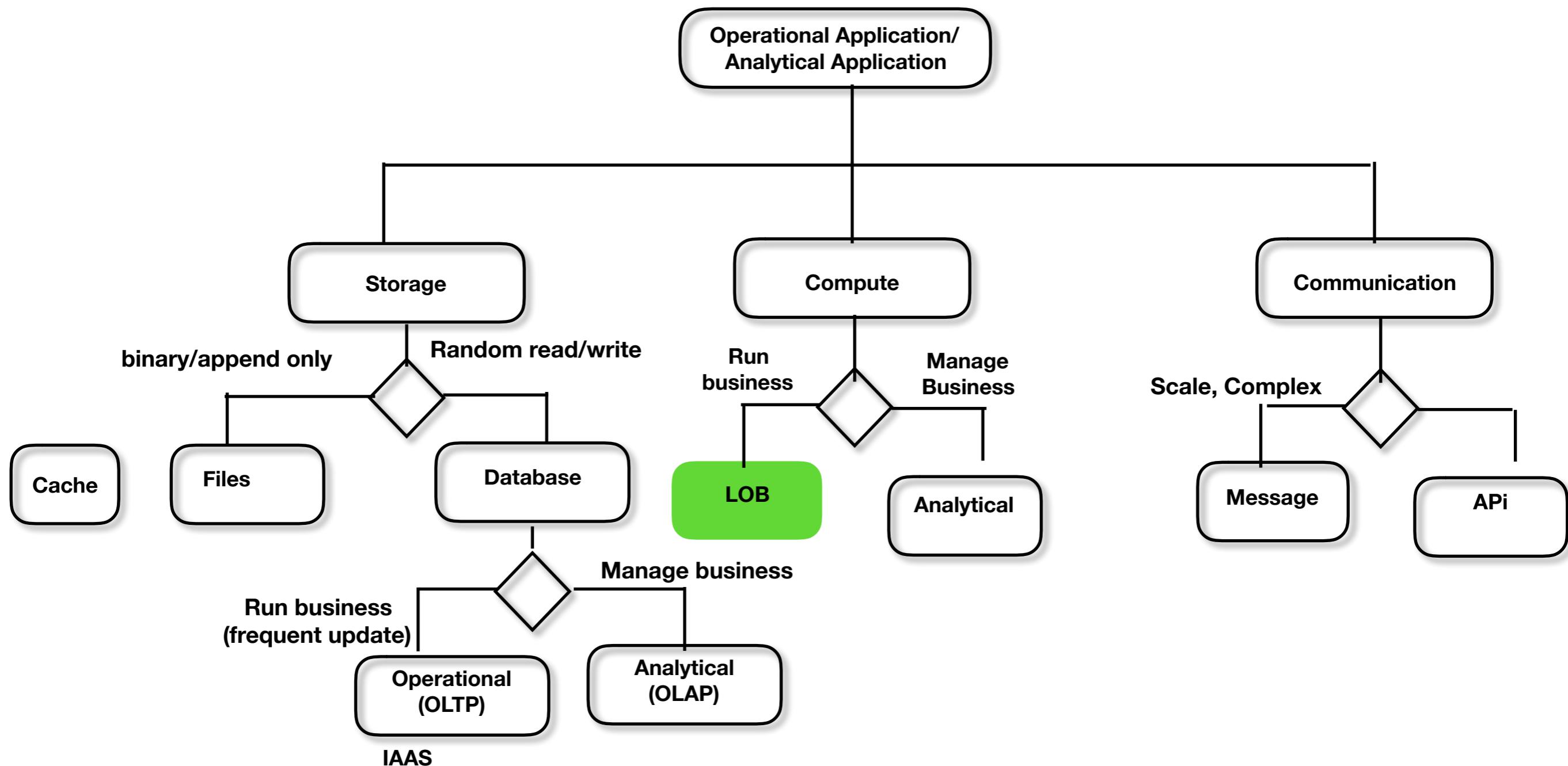


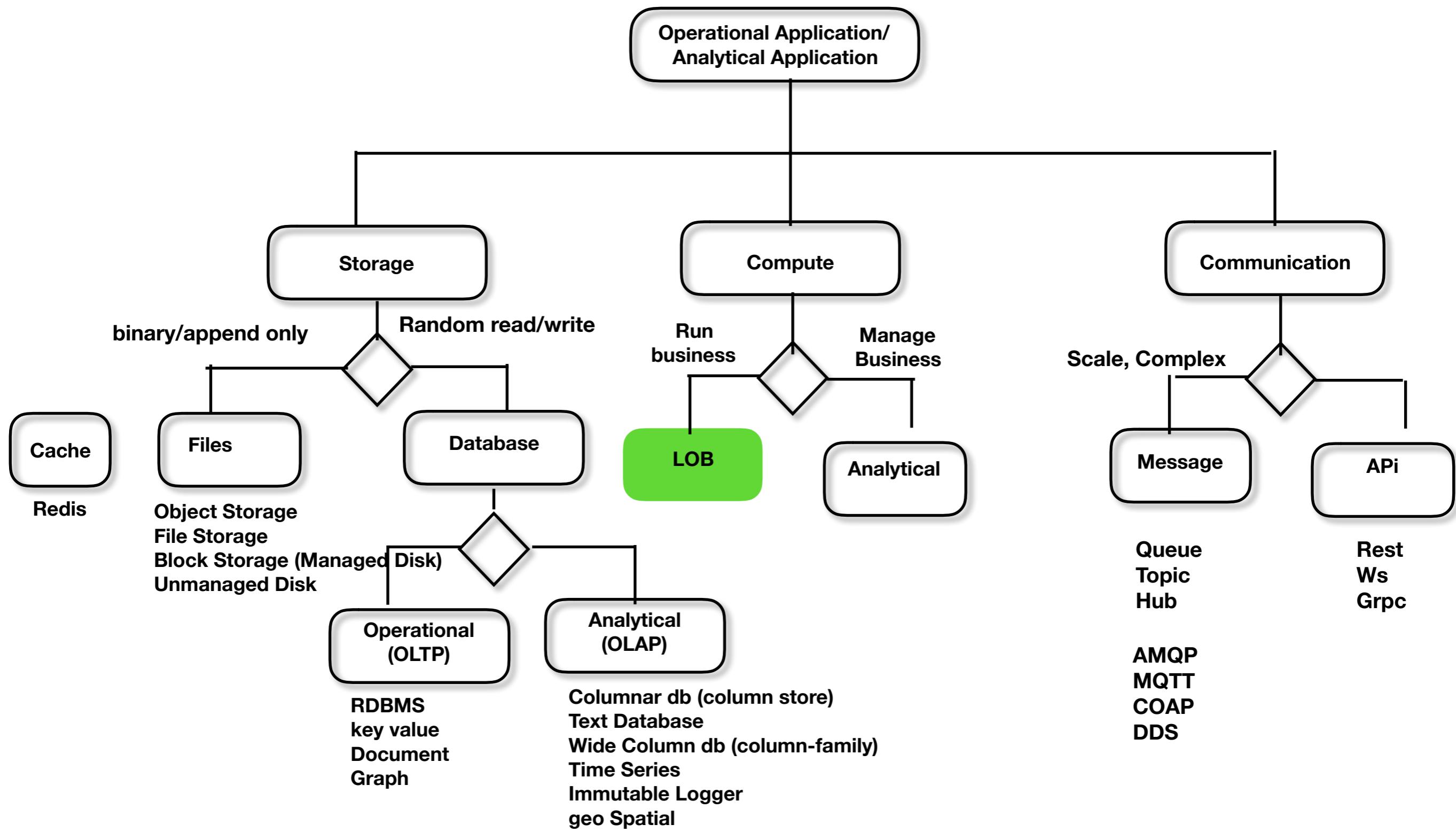
Service Oriented Architecture

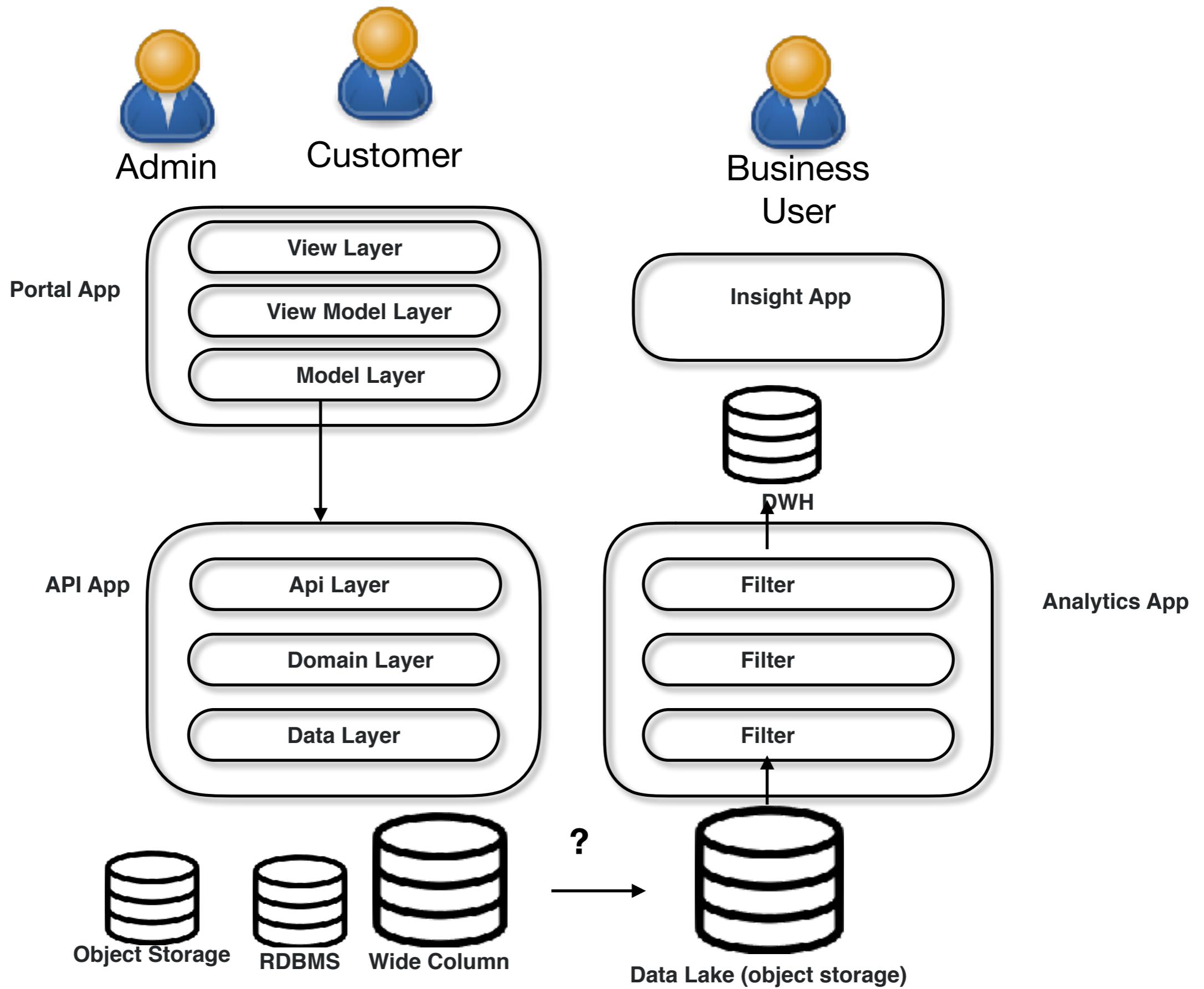


Choosing Cloud Technology ?

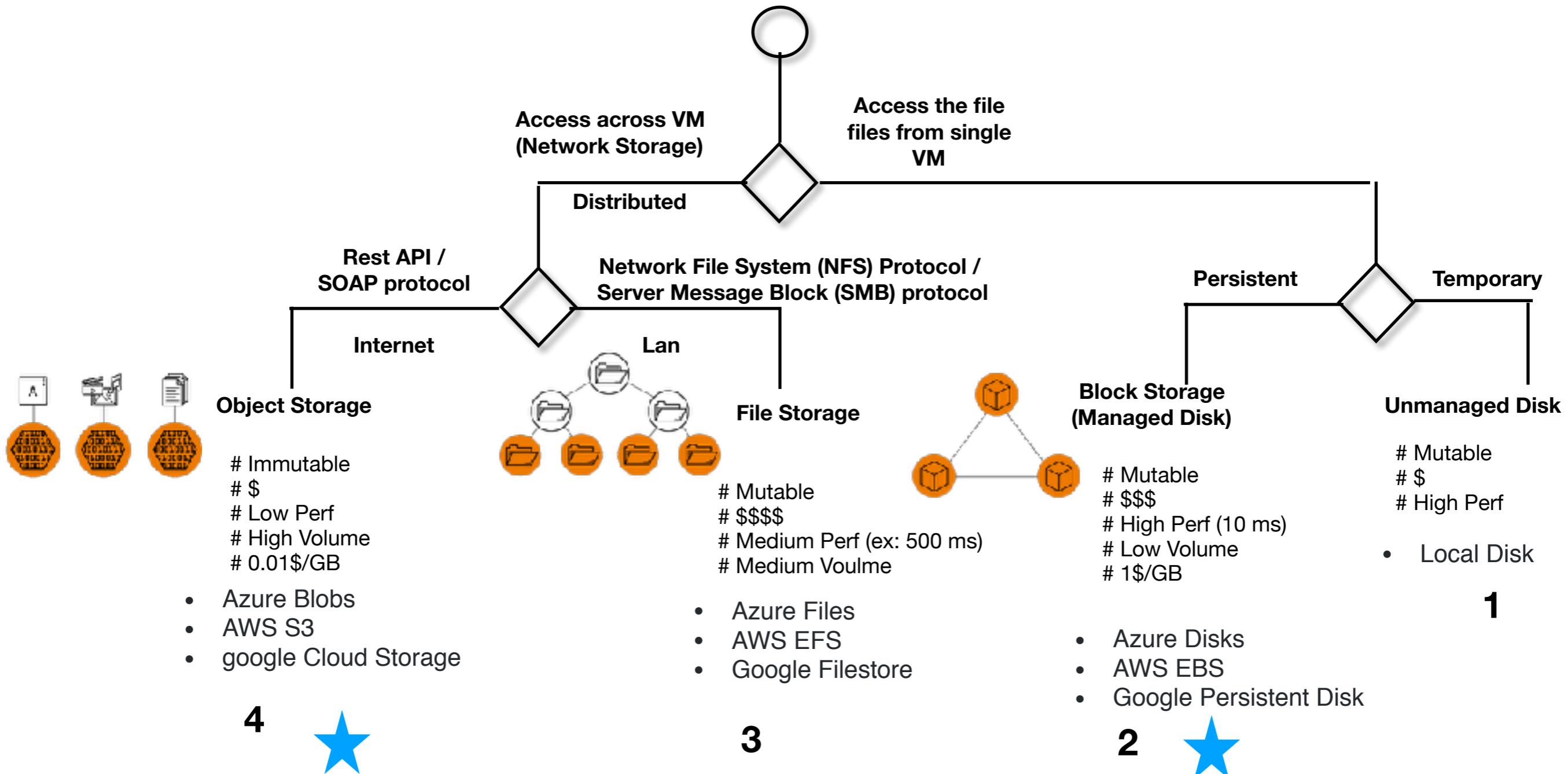




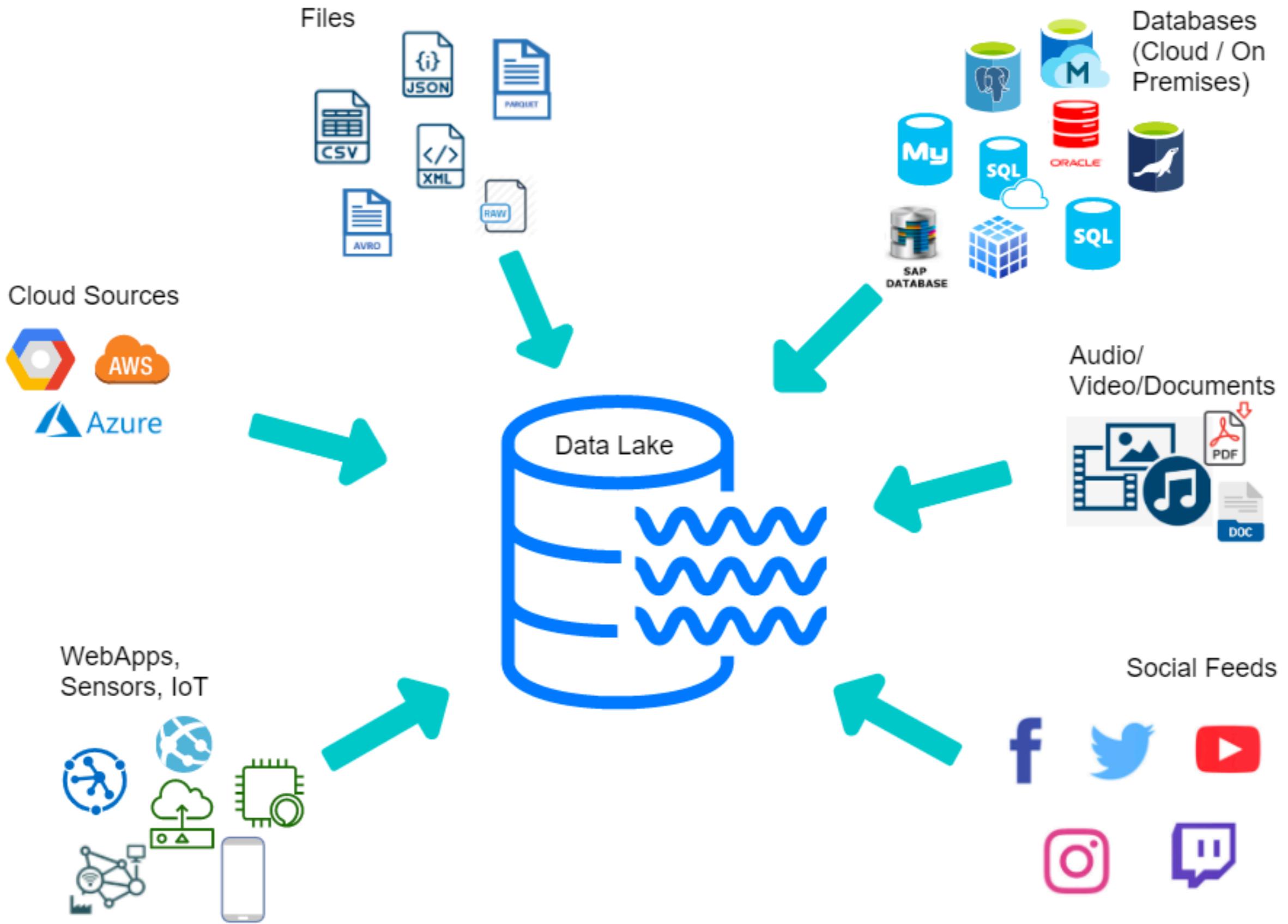


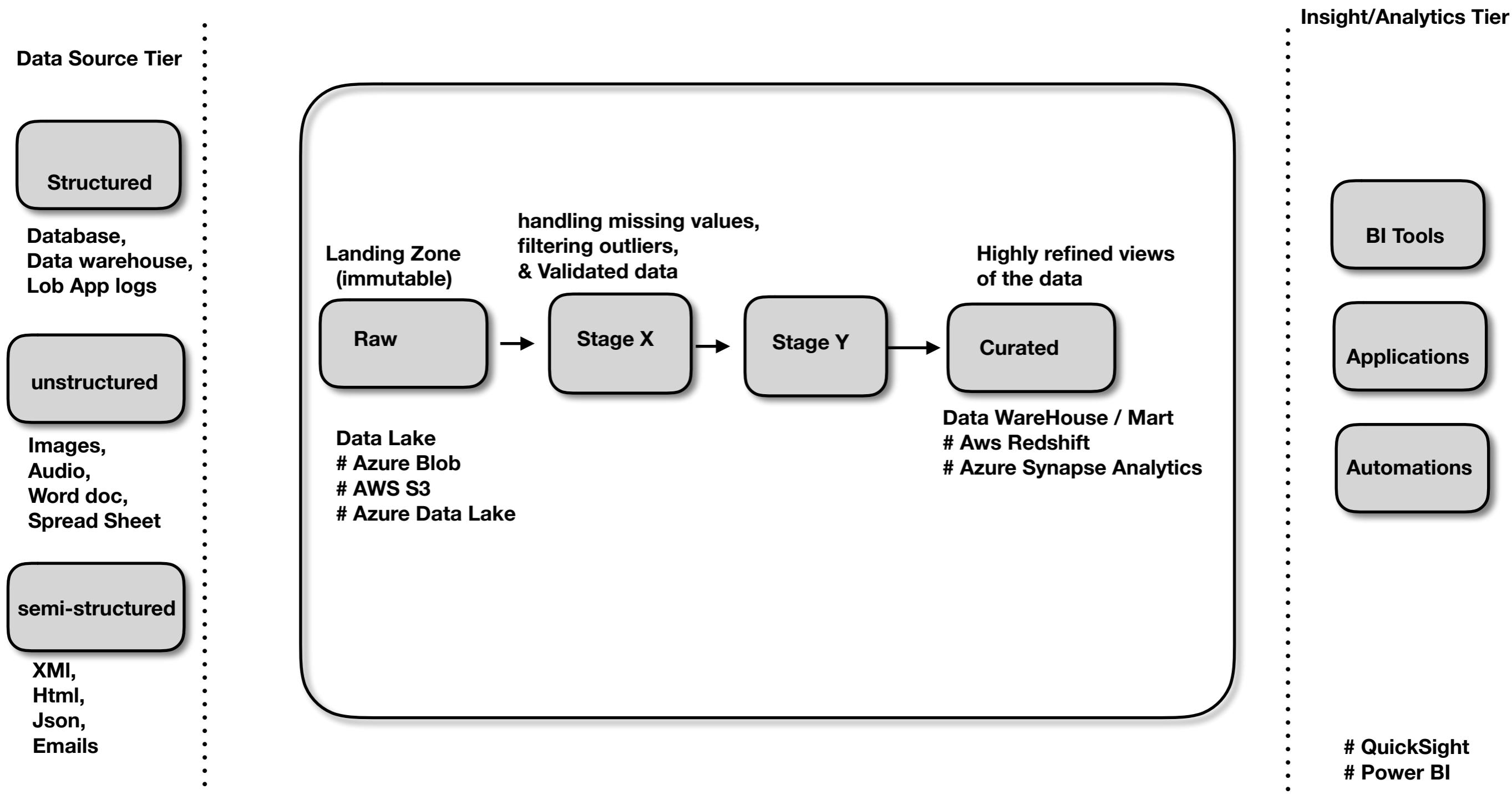


Binary Storage



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

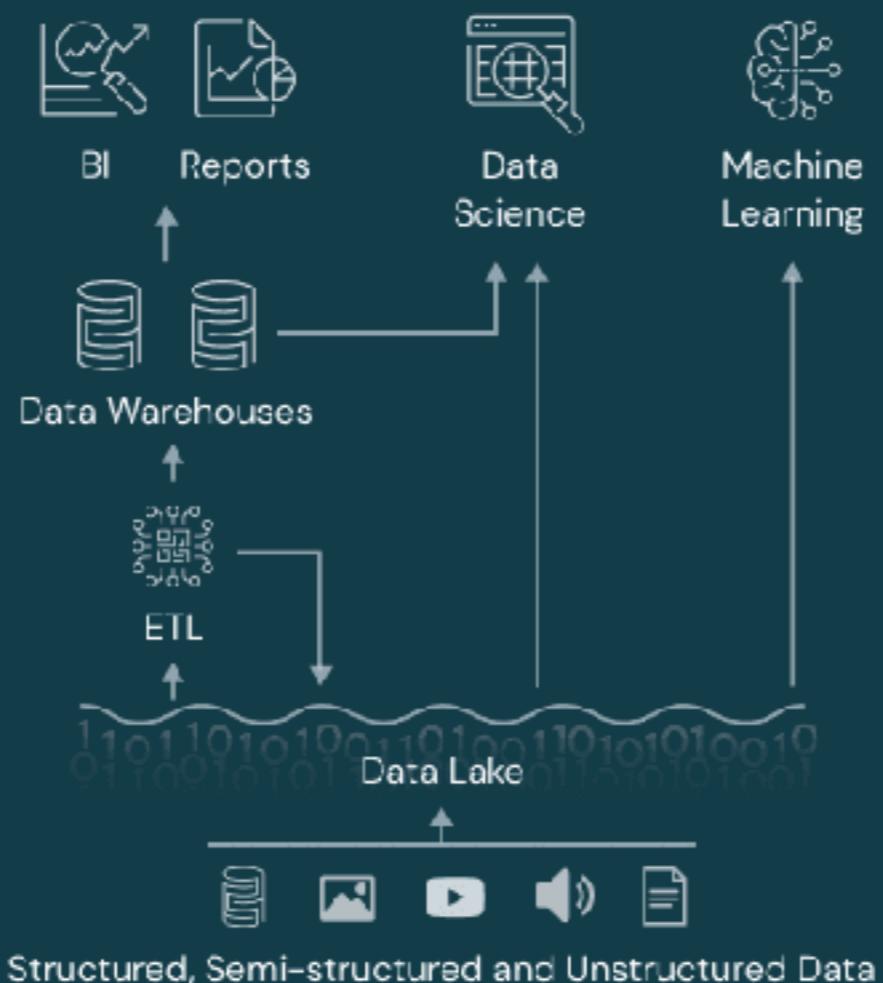




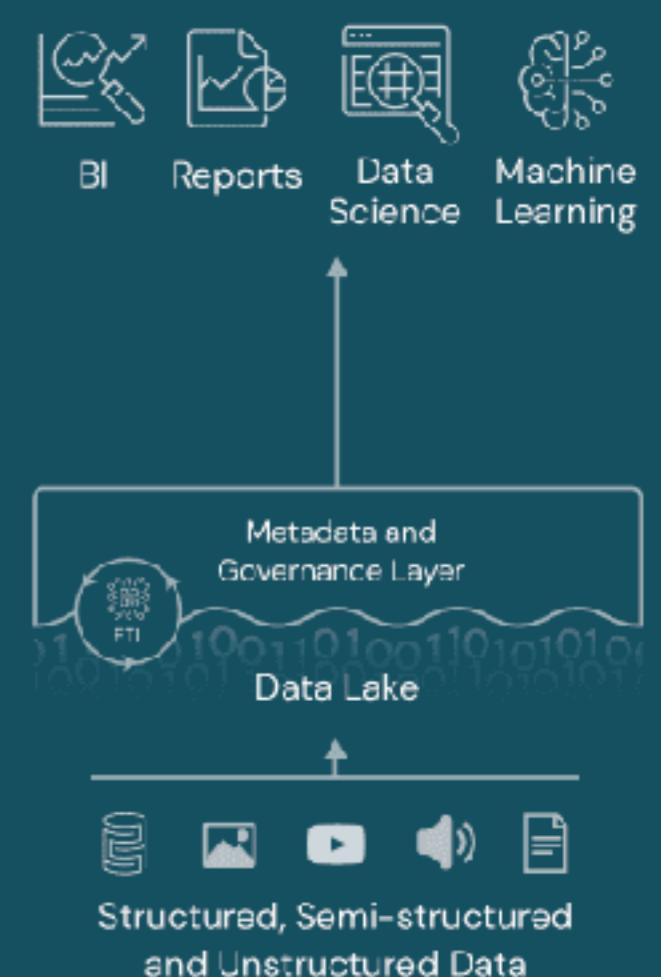
Data Warehouse



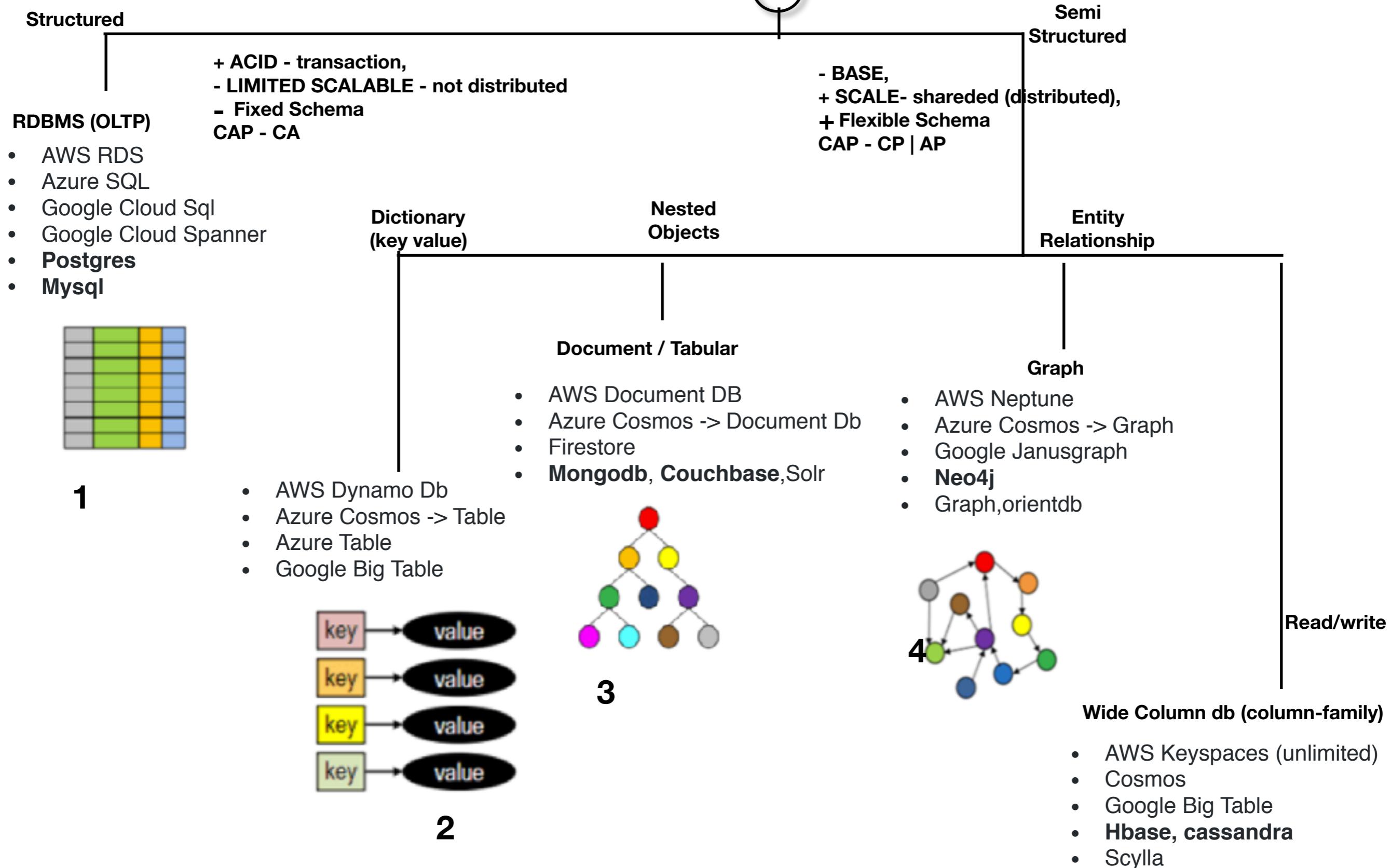
Data Lake



Data Lakehouse



Operational (OLTP)



Rows vs. Documents

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

Document 1

```
{
  data
}
```

Document 2

```
{
  data
}
```

Document 3

```
{
  data
}
```

...

Columns vs. Properties

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

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Schema vs. Schema-Free

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

Document 1

```
[
  {
    "id": "1",
    "name": "John Smith",
    "isActive": true,
    "dob": "1964-08-30"
  }
]
```

Document 2

```
[
  {
    "id": "2",
    "name": "Sarah Jones",
    "isActive": false,
    "dob": "2002-02-18"
  }
]
```

Document 3

```
[
  {
    "id": "3",
    "name": "Adam Stark",
    "isActive": true,
    "dob": "1987-07-13"
  }
]
```

Normalized vs. Denormalized

User Table

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

Holdings Table

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

Document

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

Counter

C

BP

Counter

C

BP

Counter

E

BP

Counter

?

BP

General

C

E

?

BP

BP

BP

BP

BP

BP

General

C

E

?

BP

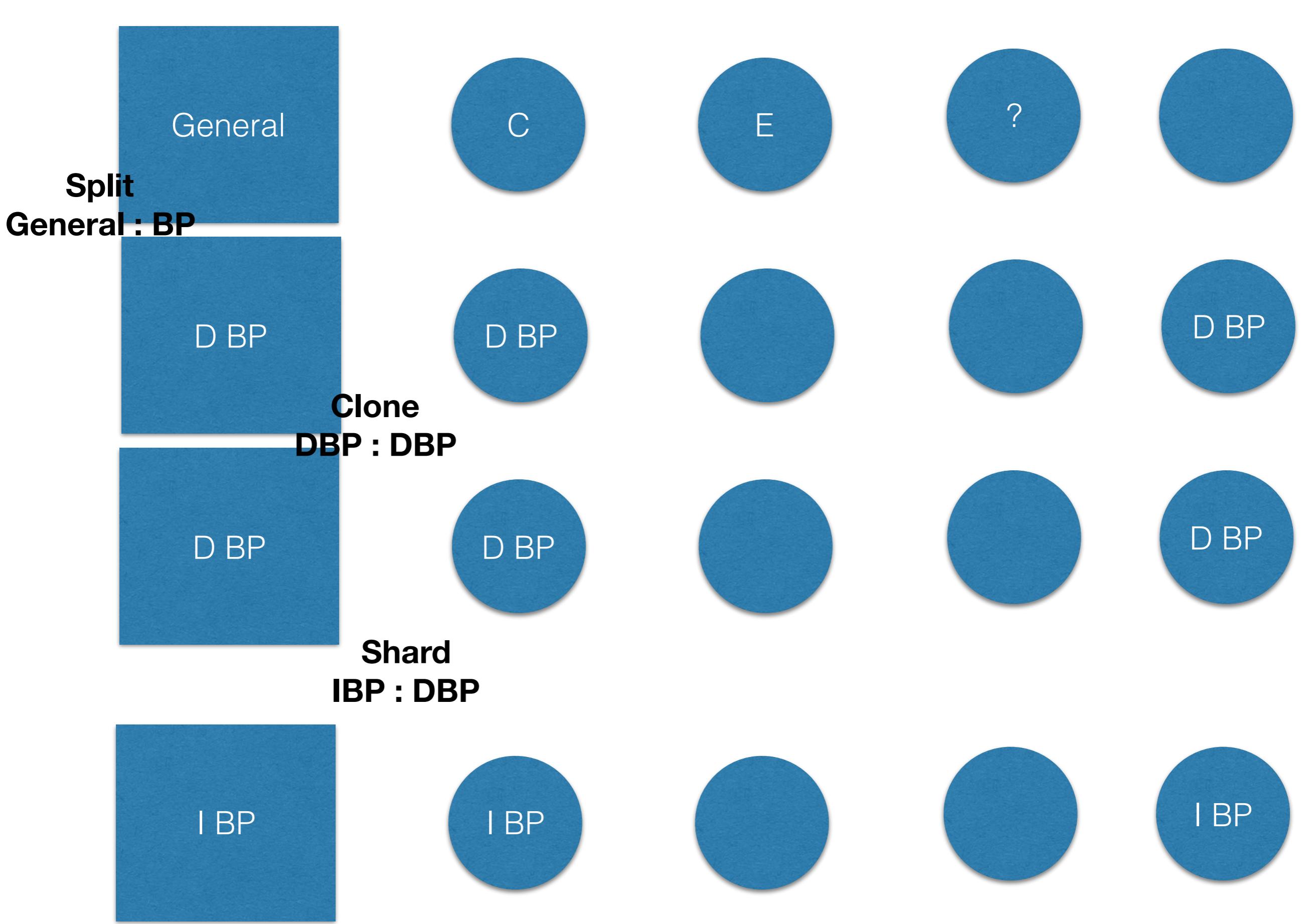
D BP

I BP

BP

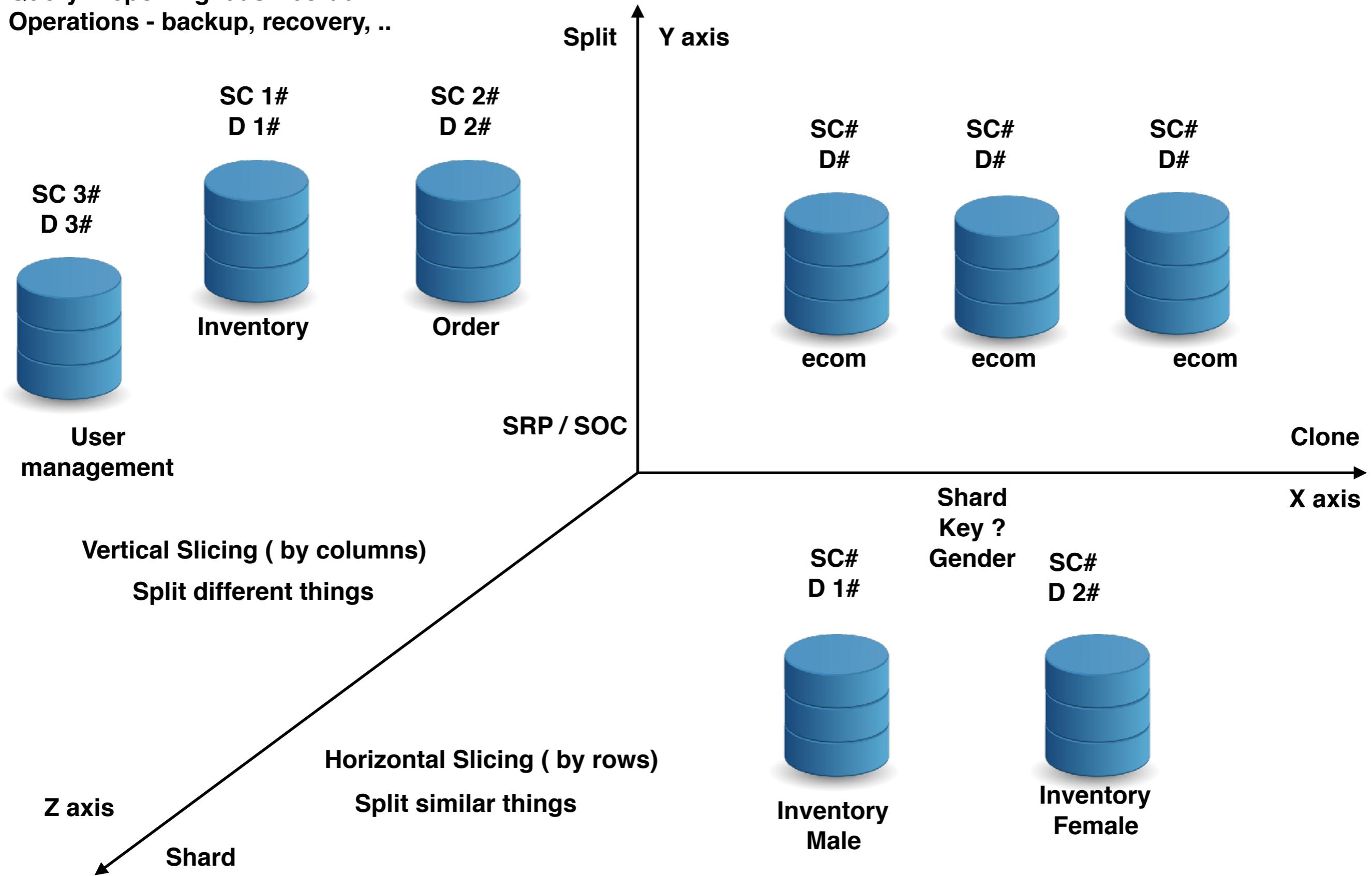
D BP

I BP

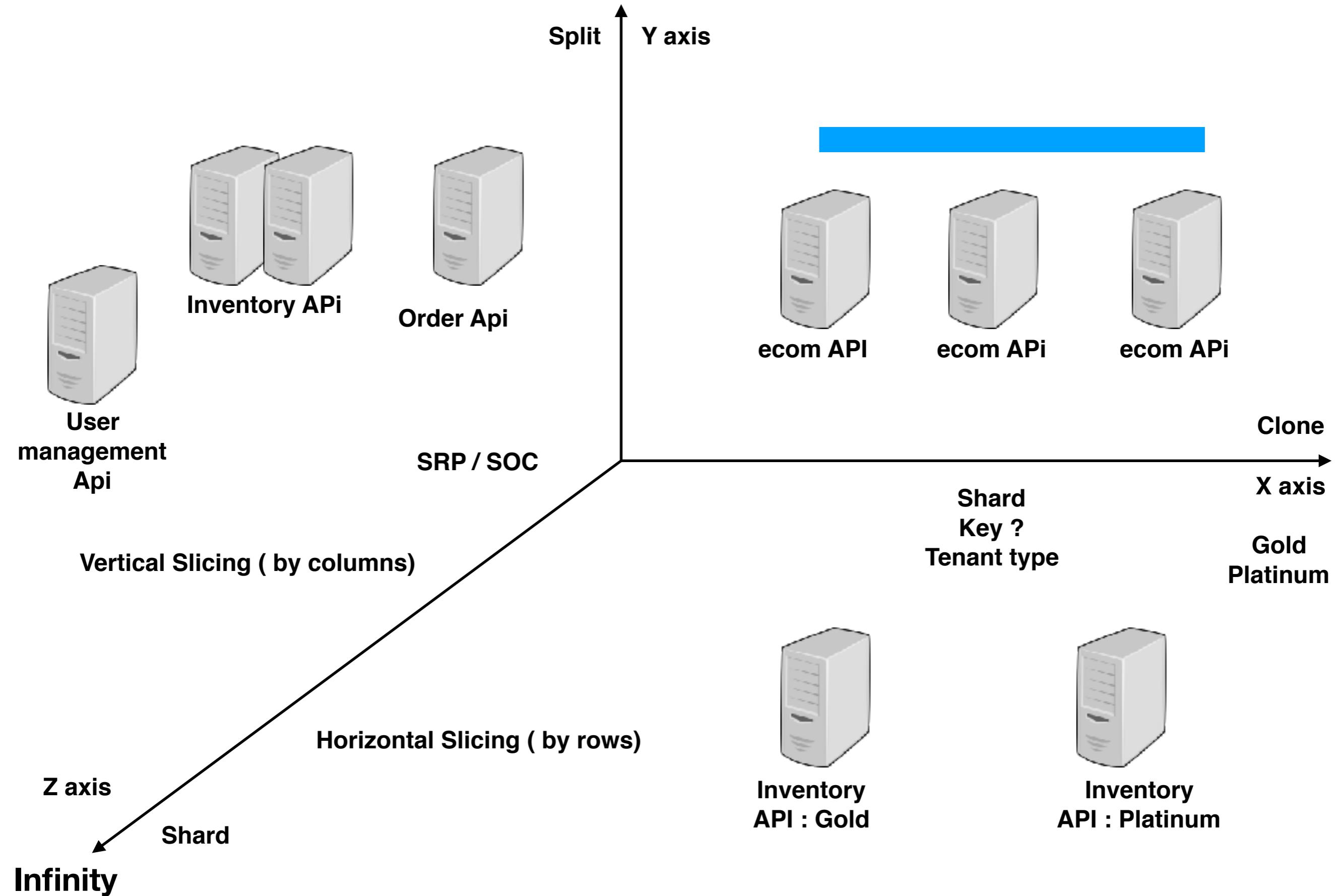


Scalability Cube - 50 rules for high Scalability

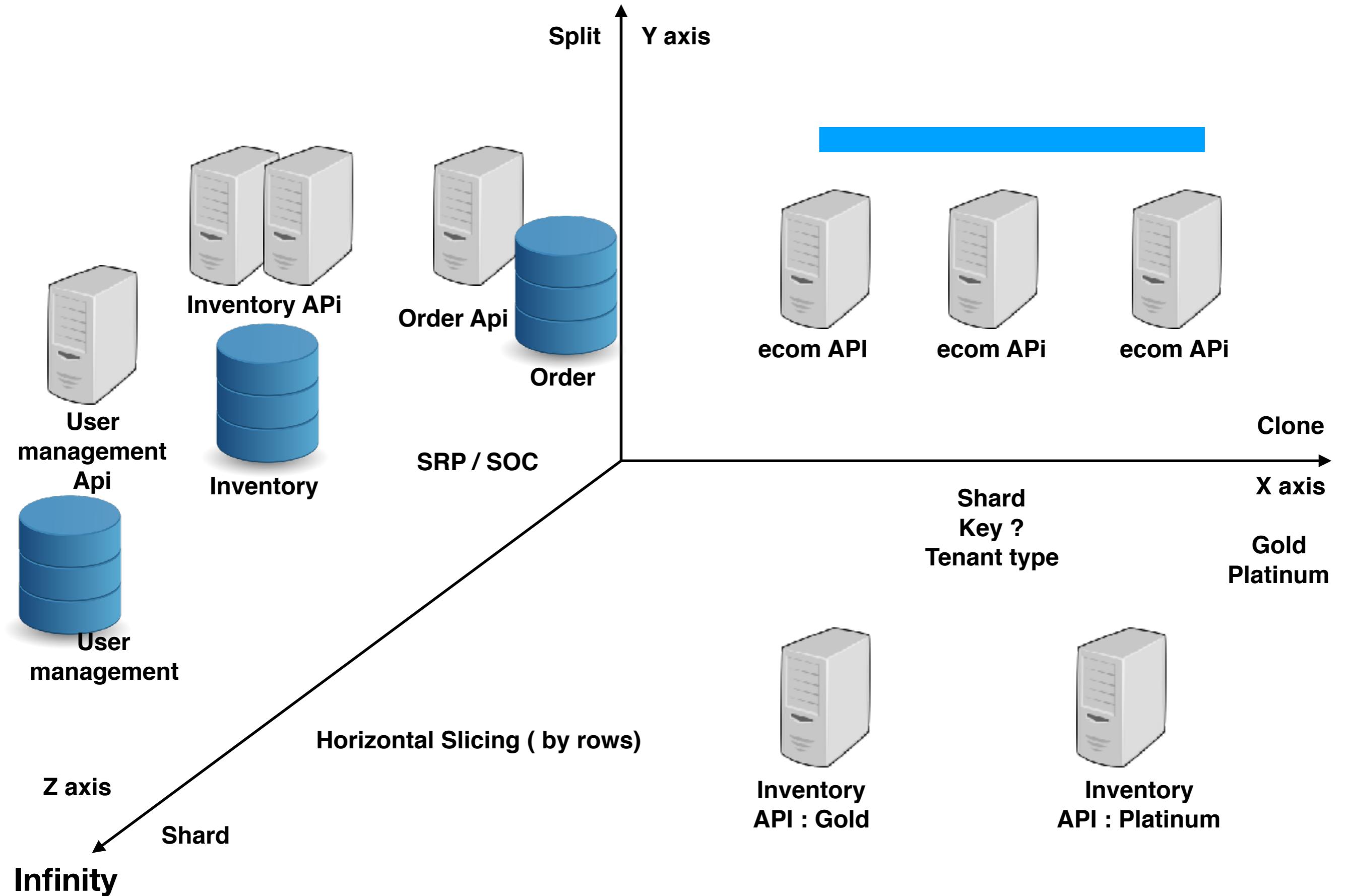
ACID - transaction
Query / reporting/ dash board
Operations - backup, recovery, ..



Scalability Cube - 50 rules for high Scalability



Scalability Cube - 50 rules for high Scalability





Order Api

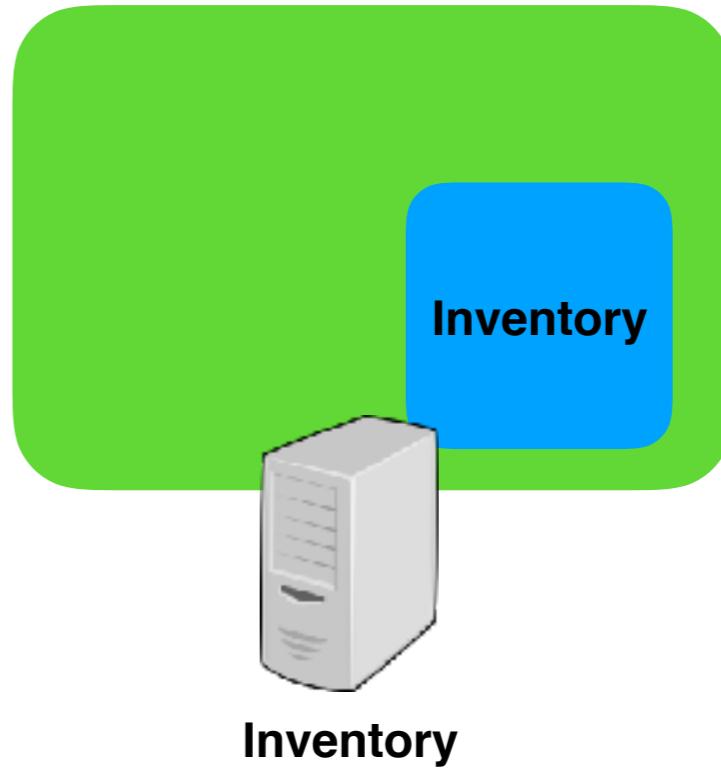
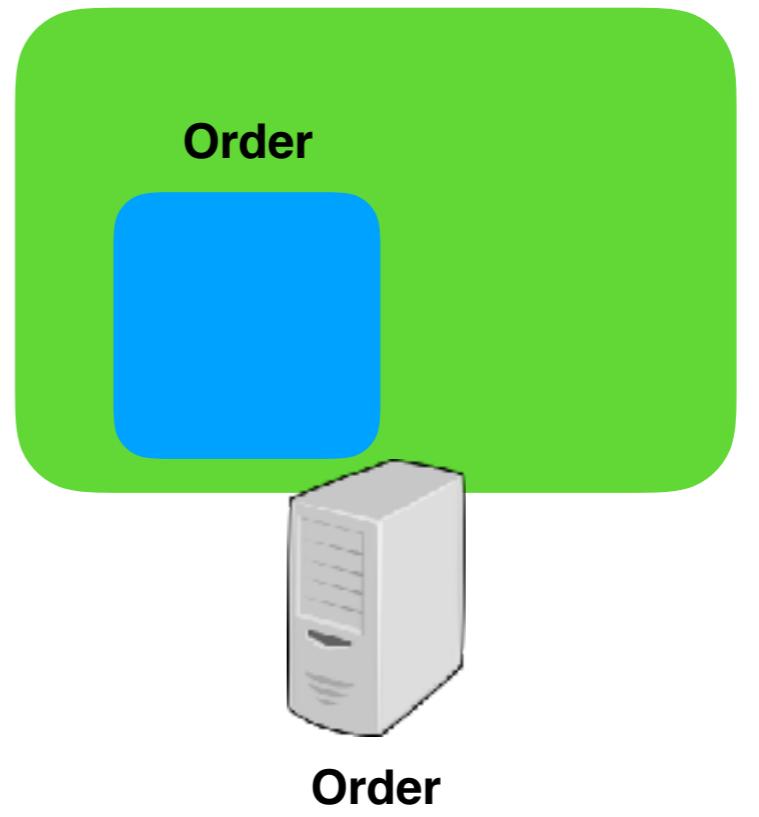
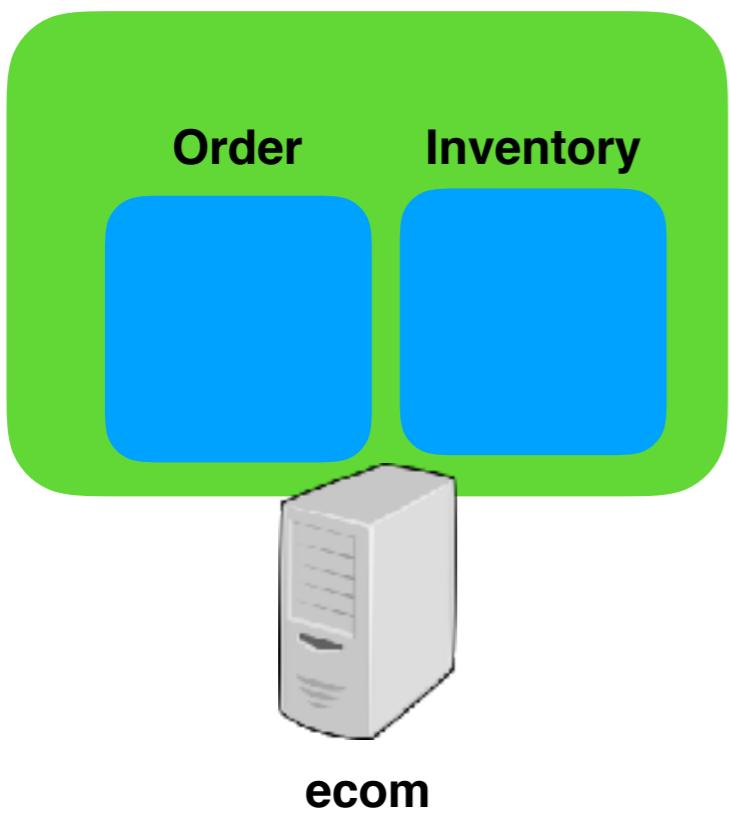


**Inventory
API : Gold**

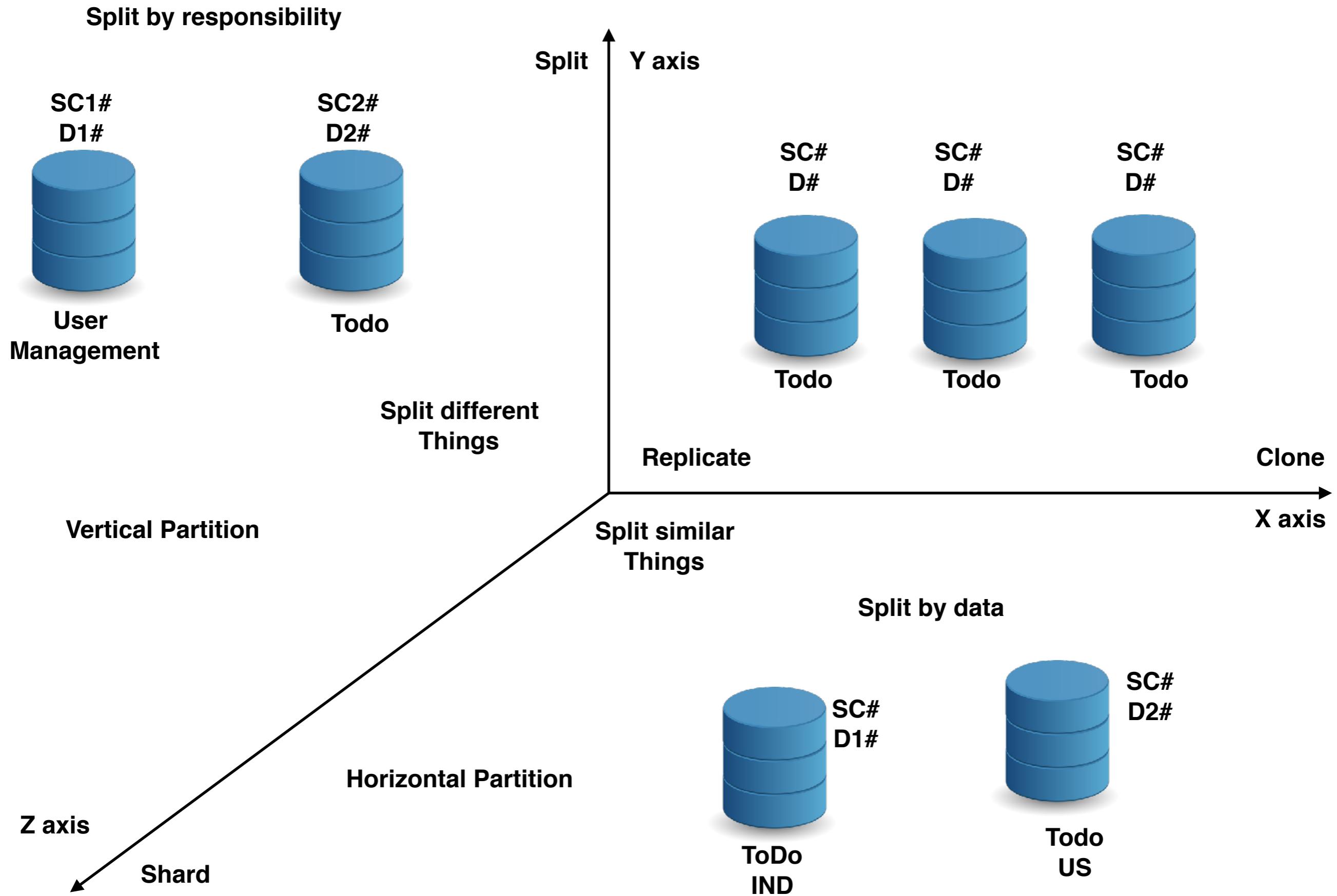


**Inventory
API : Platinum**





Scalability Cube - 50 rules for high Scalability



**Split different
Things**



Ecom

Vertical Partition



User



Inventory



Accounts



Order

Horizontal Partition



Shard



Inventory:
M



Order:
AUS



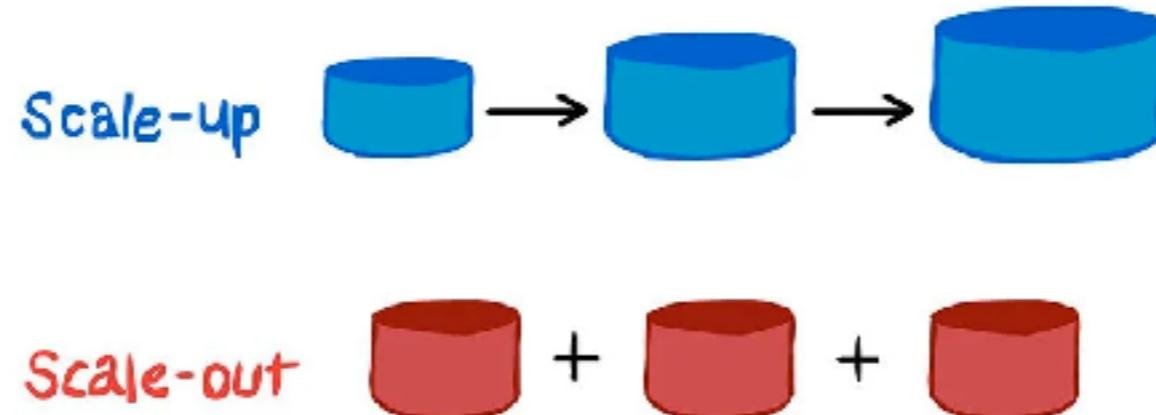
Order:
USD



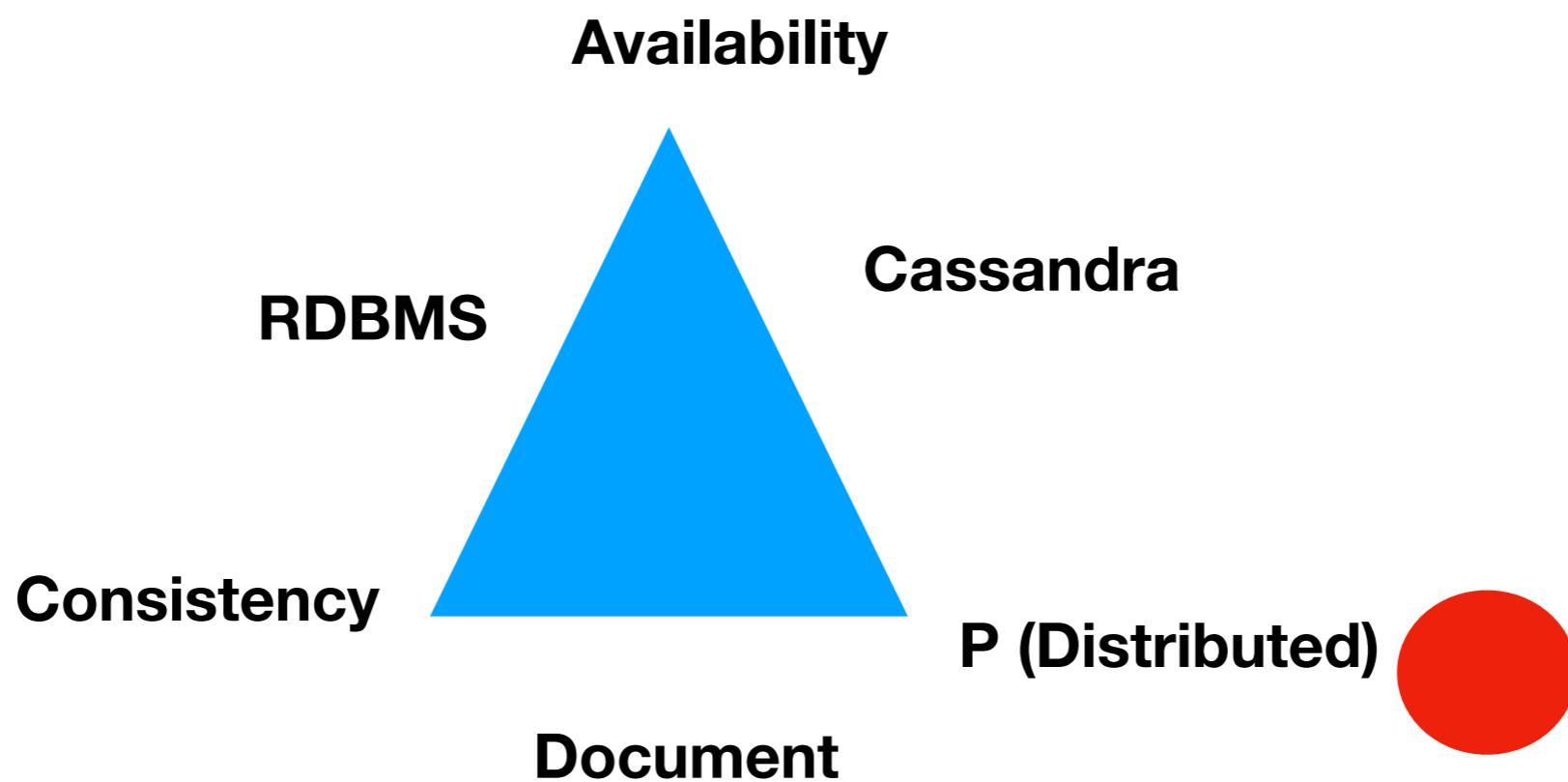
Inventory: **Inventory:**
M **M**

Clone

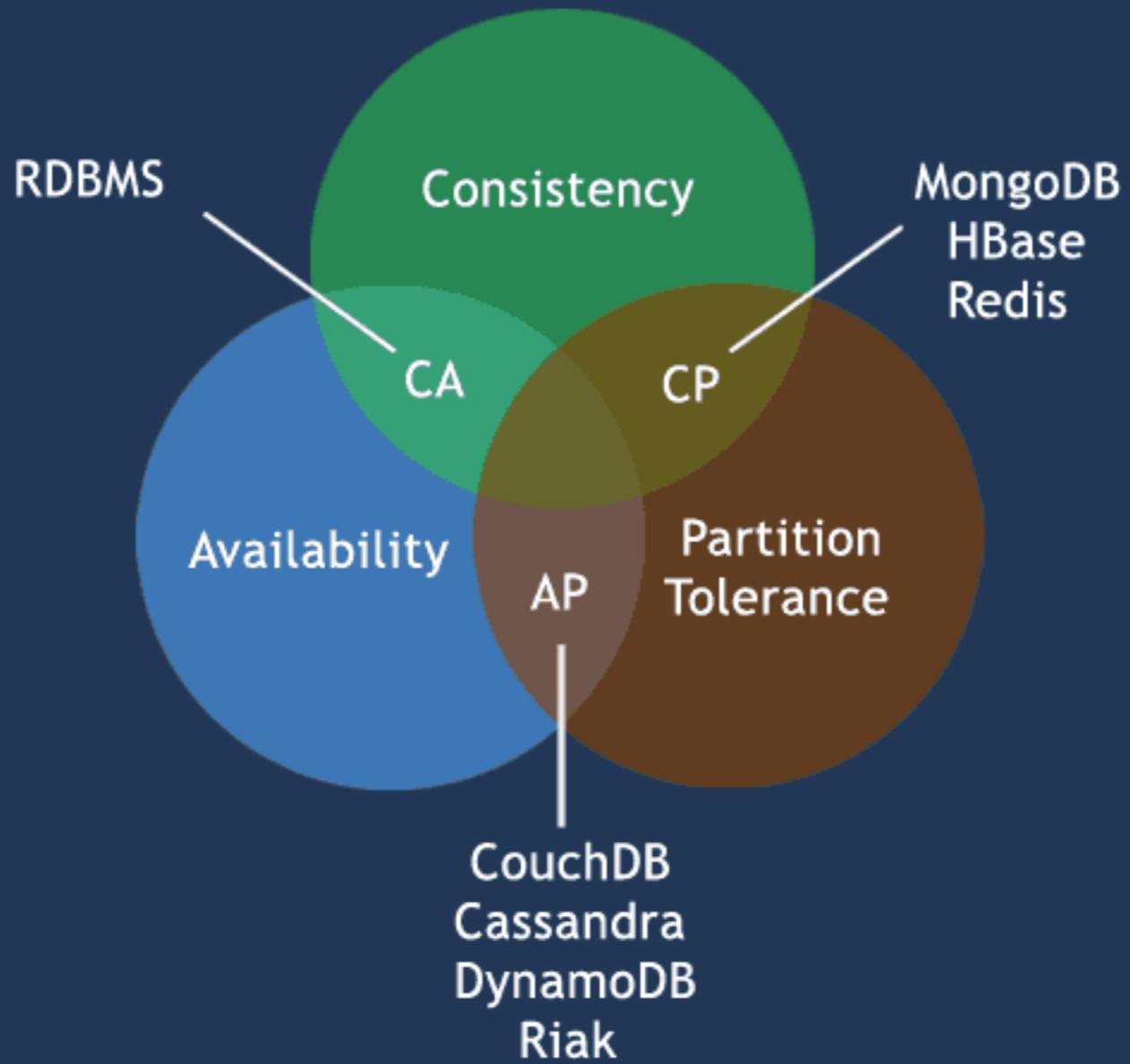
Scale-Up vs. Scale-Out

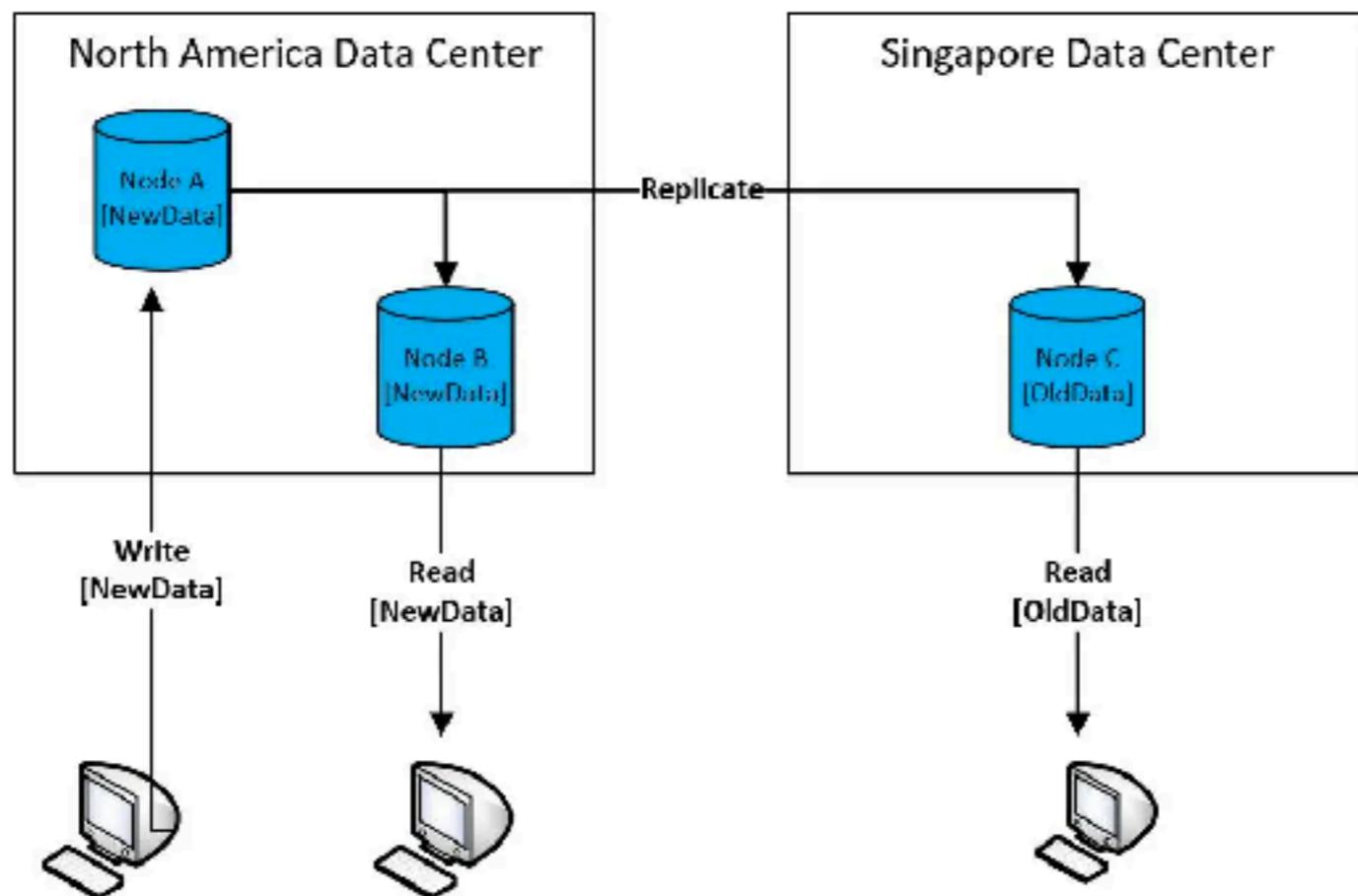
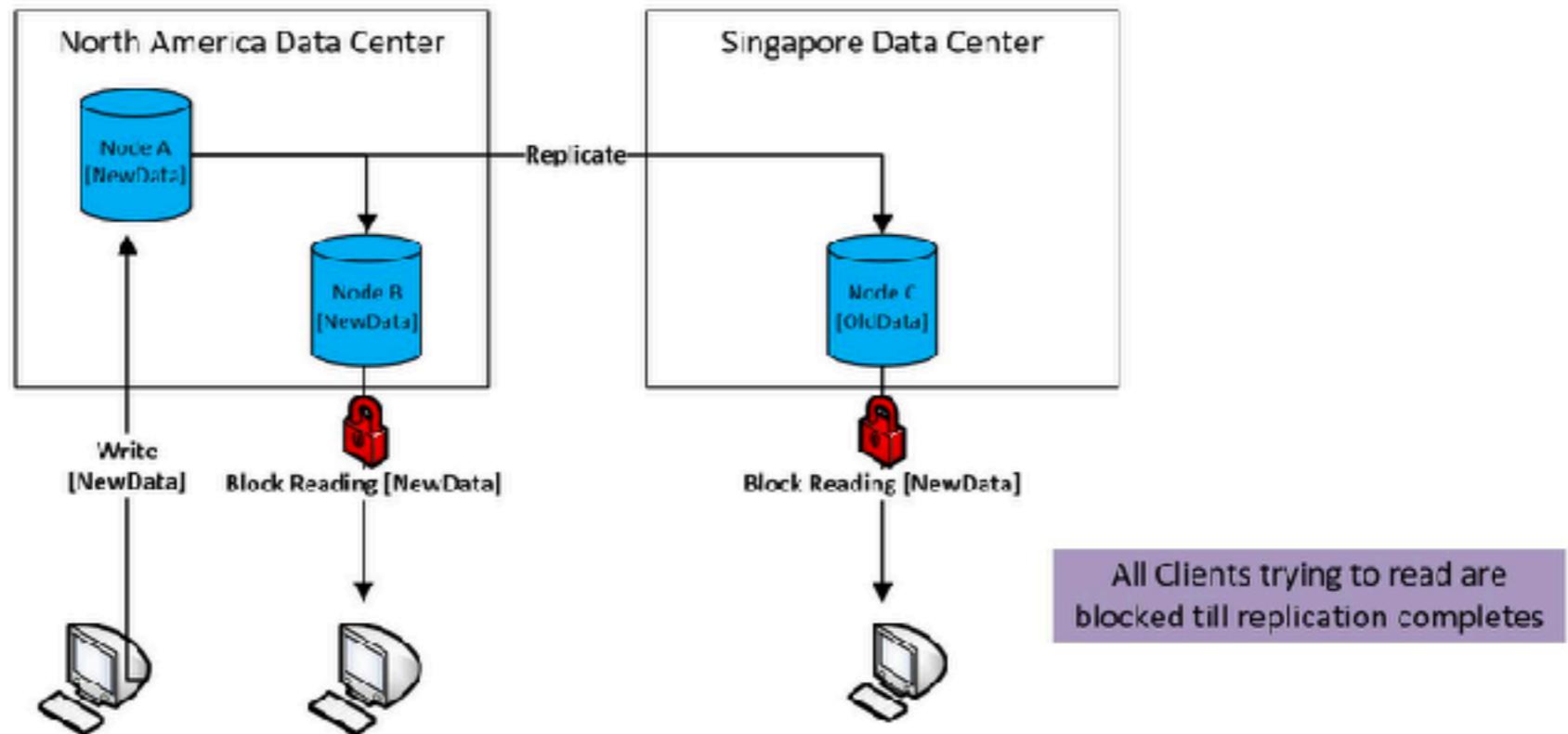


Strong Consistency vs. Eventual Consistency



CAP Theorem





COMMON COMPARISONS BETWEEN MySQL & NoSQL

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

Facebook, Wikipedia, Quora,
Flickr

MySQL

Twitter

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

LinkedIn

Oracle Database and Voldemort

YouTube

MySQL -> BigTable

Microsoft, Myspace

SQL Server

Yahoo

PostgreSQL

Key Value

- Twitter uses Redis to deliver **your Twitter timeline**
- Pinterest uses Redis to store lists of users, followers, unfollowers, boards, **and more**
- **Coinbase** uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions

Graph

- **Walmart** uses Neo4j to provide customers personalized, relevant product recommendations and promotions
- **Medium** uses Neo4j to build their social graph to enhance content personalization
- **Cisco** uses Neo4j to mine customer support cases to anticipate bugs

Document

- SEGA uses MongoDB for handling 11 million in-game accounts
- Cisco moved its VSRM (video session and research manager) platform to Couchbase to **achieve greater scalability**
- Aer Lingus uses MongoDB with **Studio 3T** to handle ticketing and internal apps
- Built on MongoDB, The Weather Channel's iOS and Android apps **deliver weather alerts** to 40 million users in real-time

Use Cases

Content Management System

Our Confusing Economy

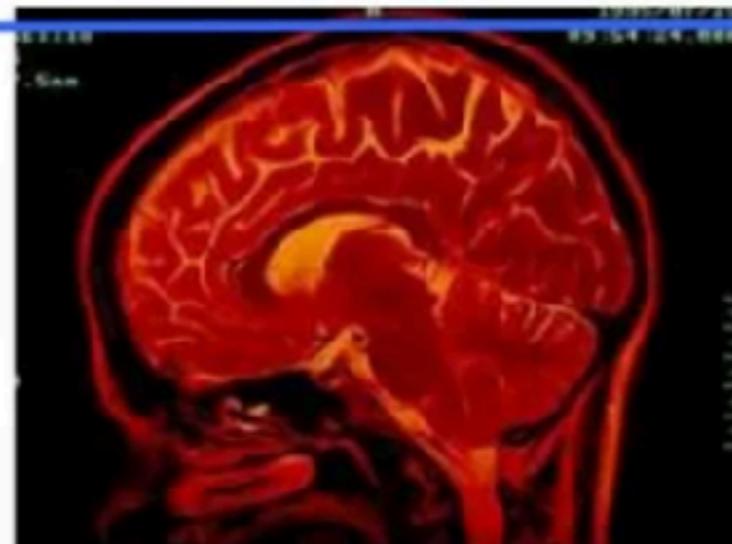
Joe Weisenthal | Nov. 30, 2012, 9:59 AM | 693 | 3

 Recommend 1  Share 1  Tweet 27  +1 0  Email 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

Pumps

umps & Heels

ats

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

High

Filter by
attributes



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



Leavy 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

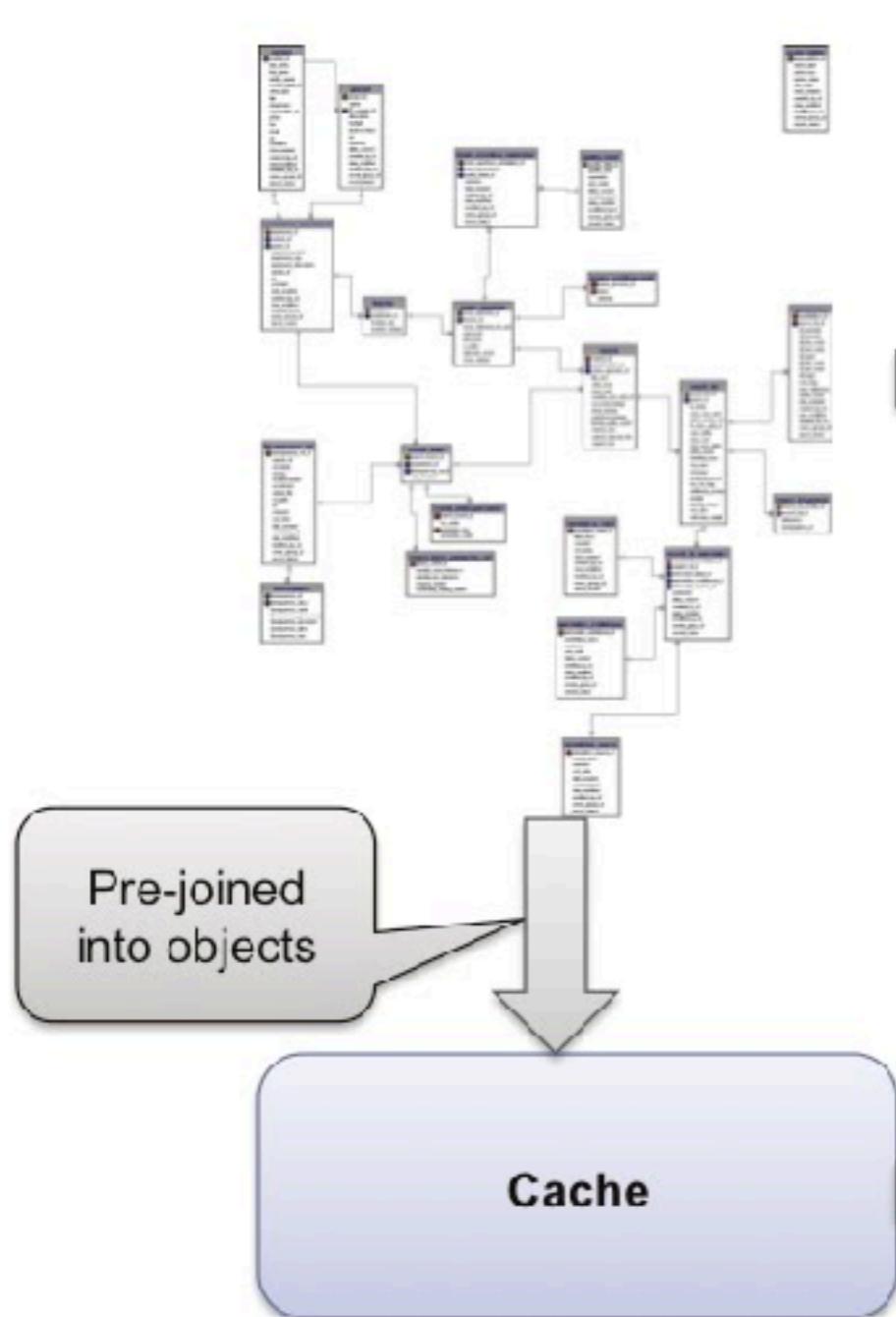
\$59.95

Compare at \$79.00

More Colors

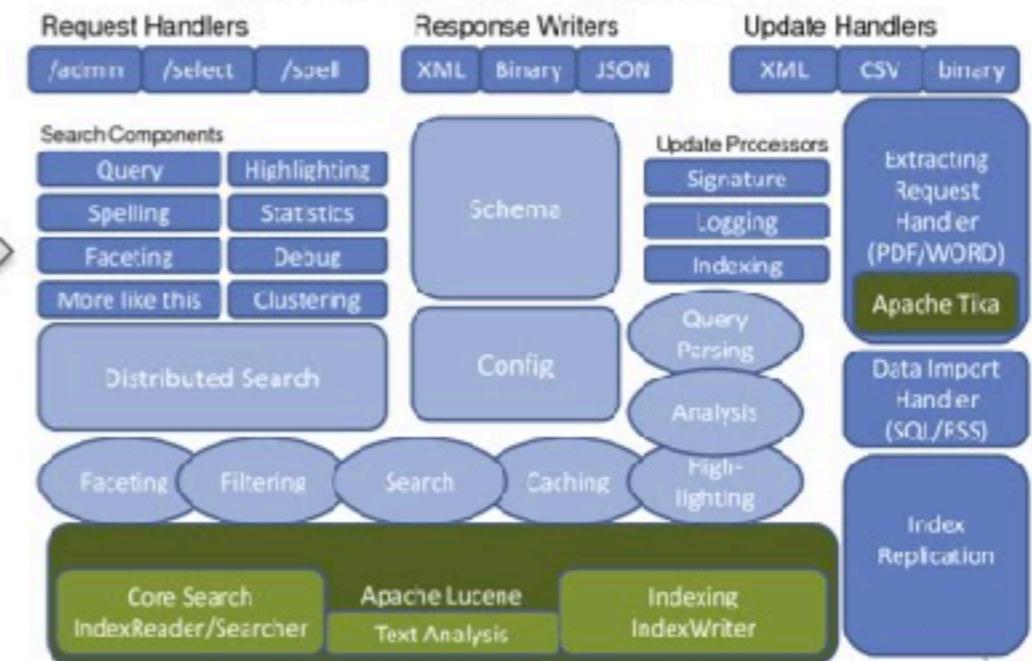
Lists hundreds
of item
summaries

Product Data Store



Product Search

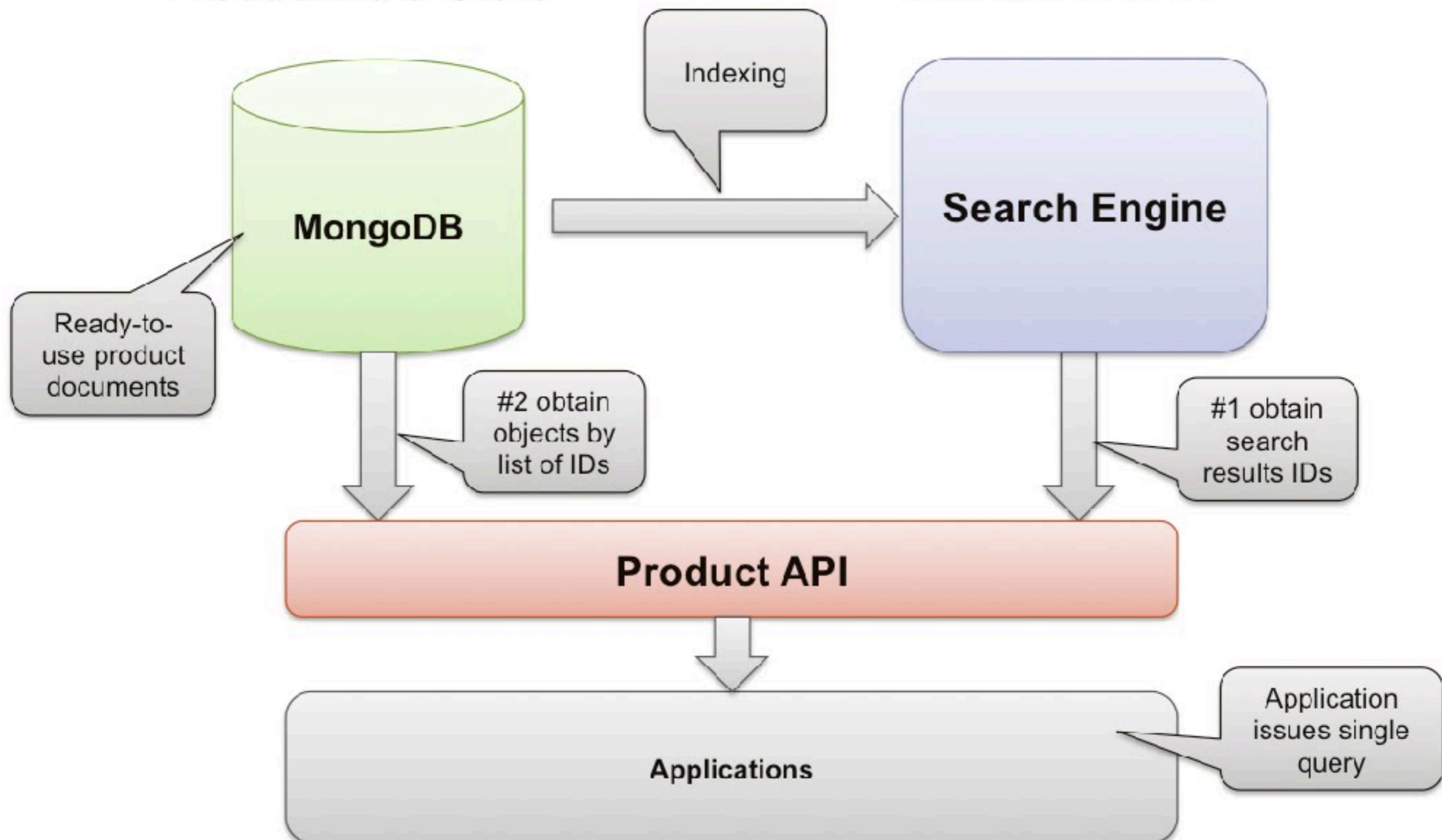
Lucene/Solr Architecture



- 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



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

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.

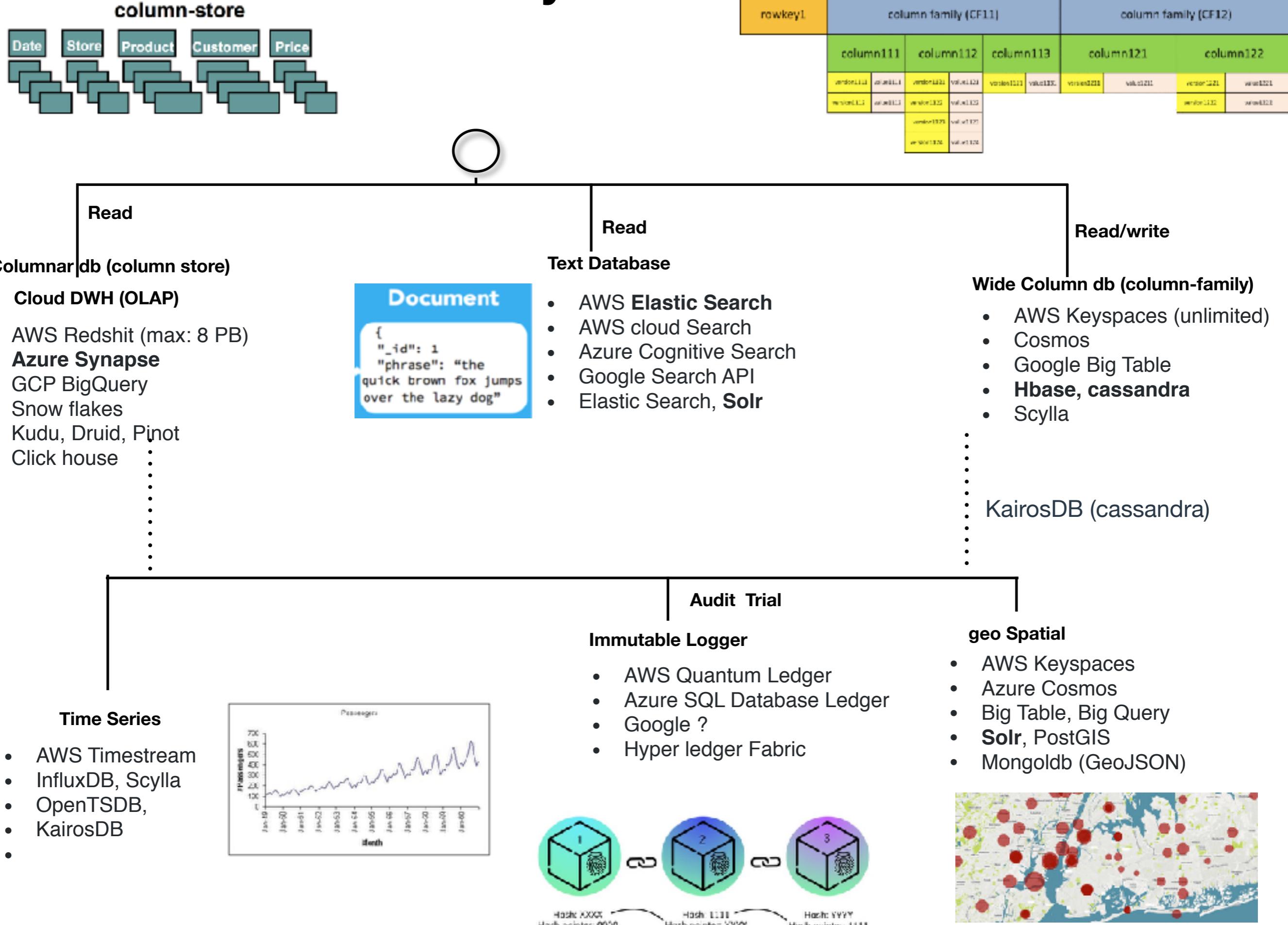
- build social relationship to enhance content personalization

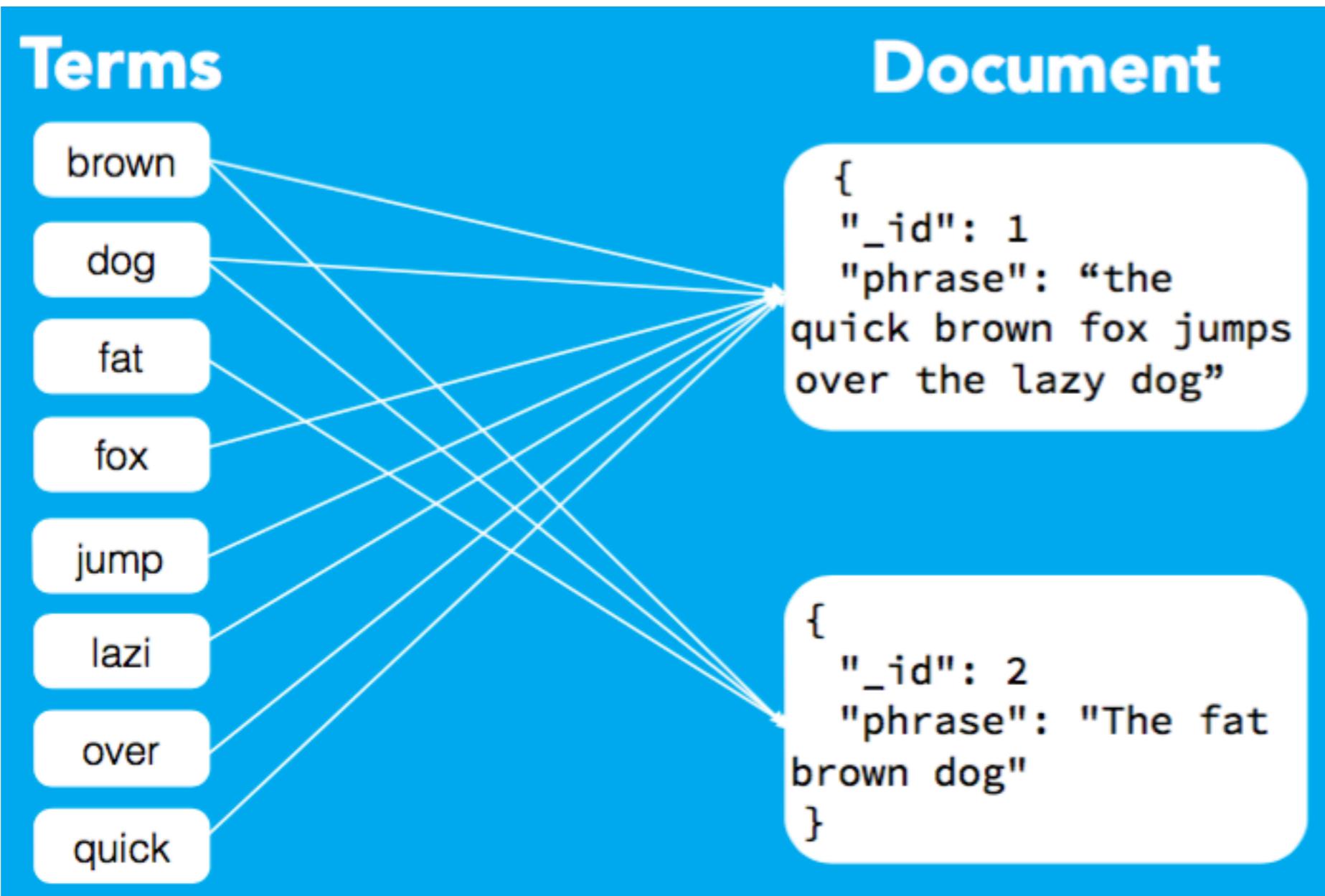


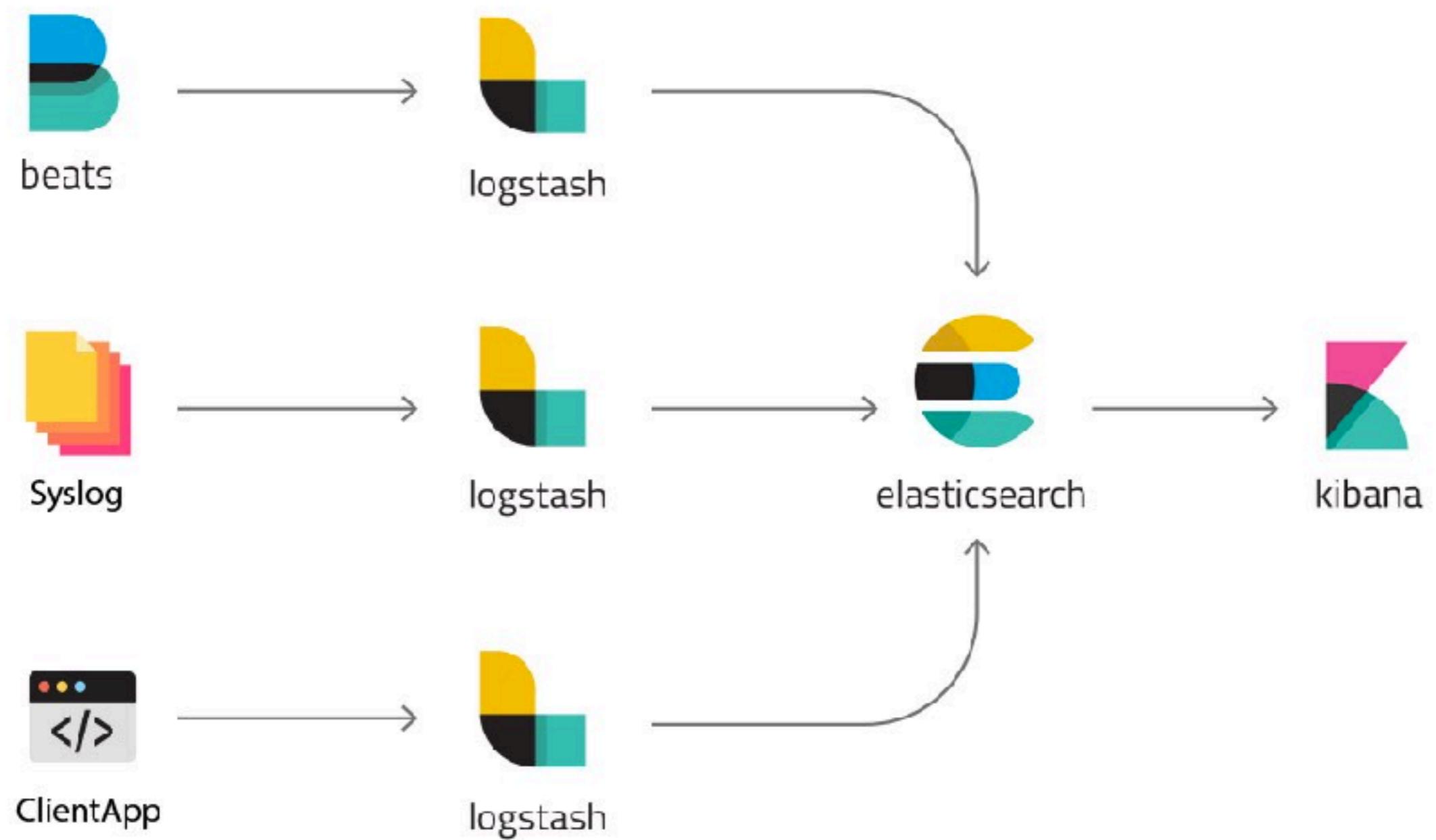


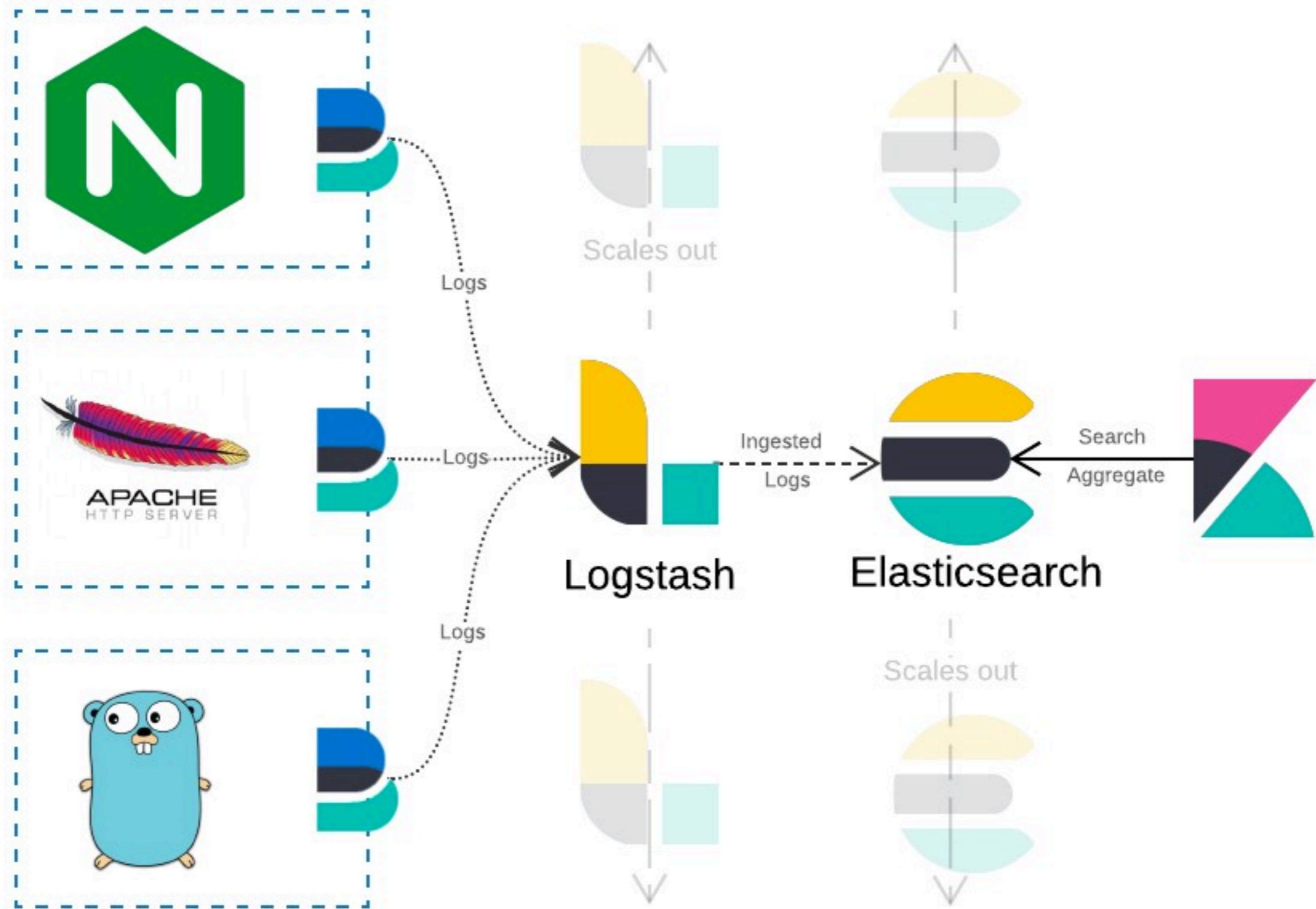
- Store user session details and preference. All the information lend themselves to fast reads and writes.

Analytical

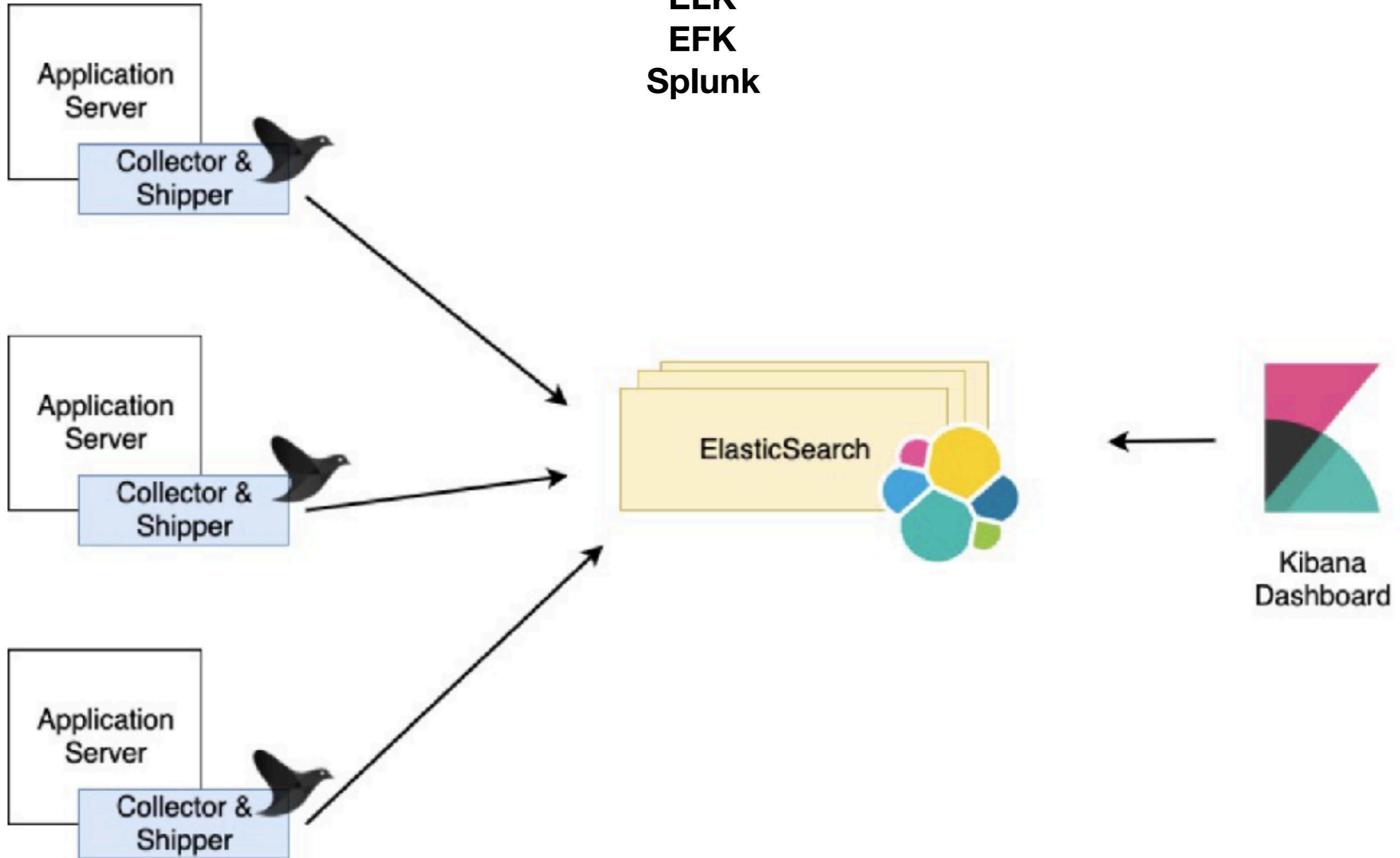






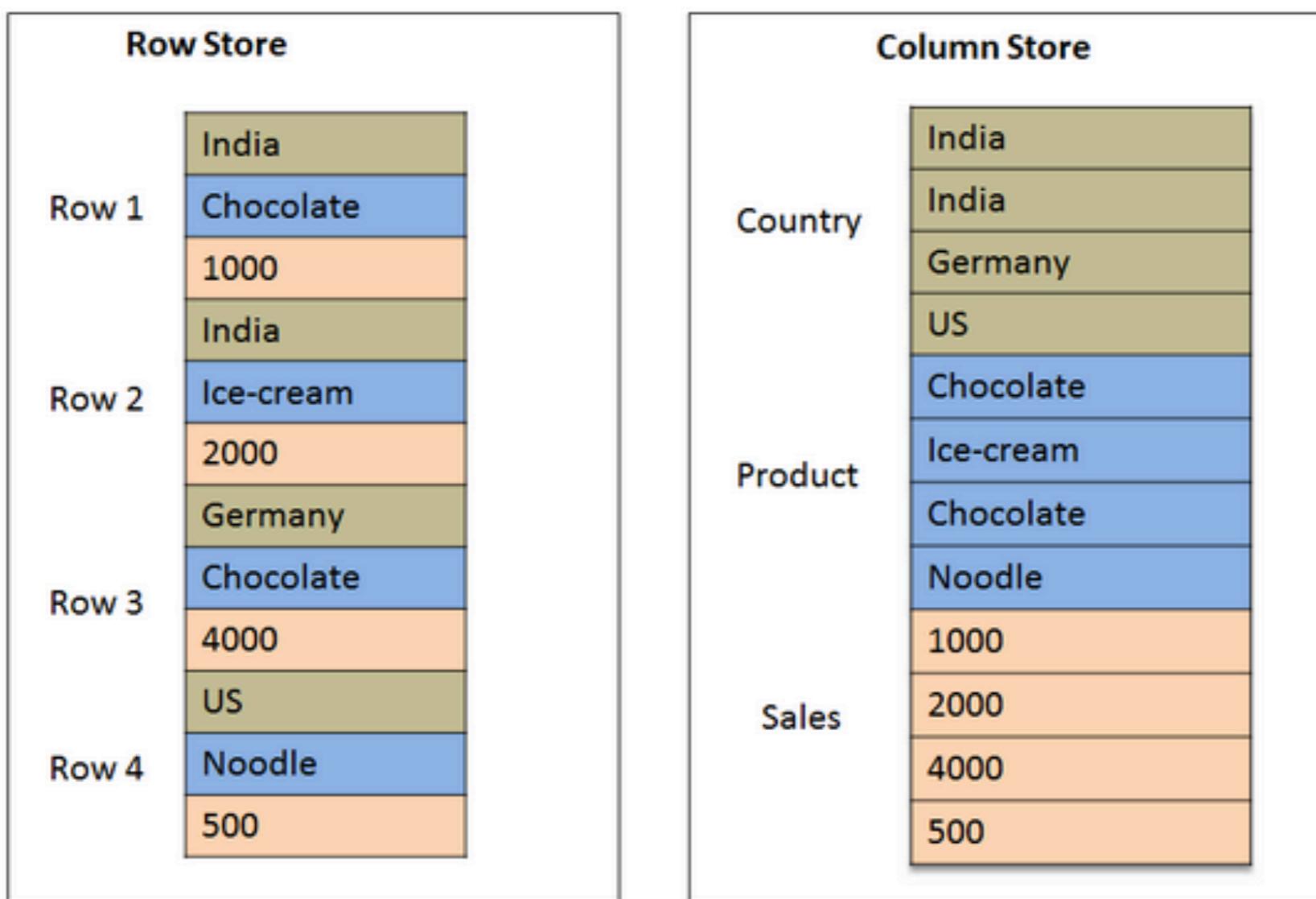


ELK EFK Splunk



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							

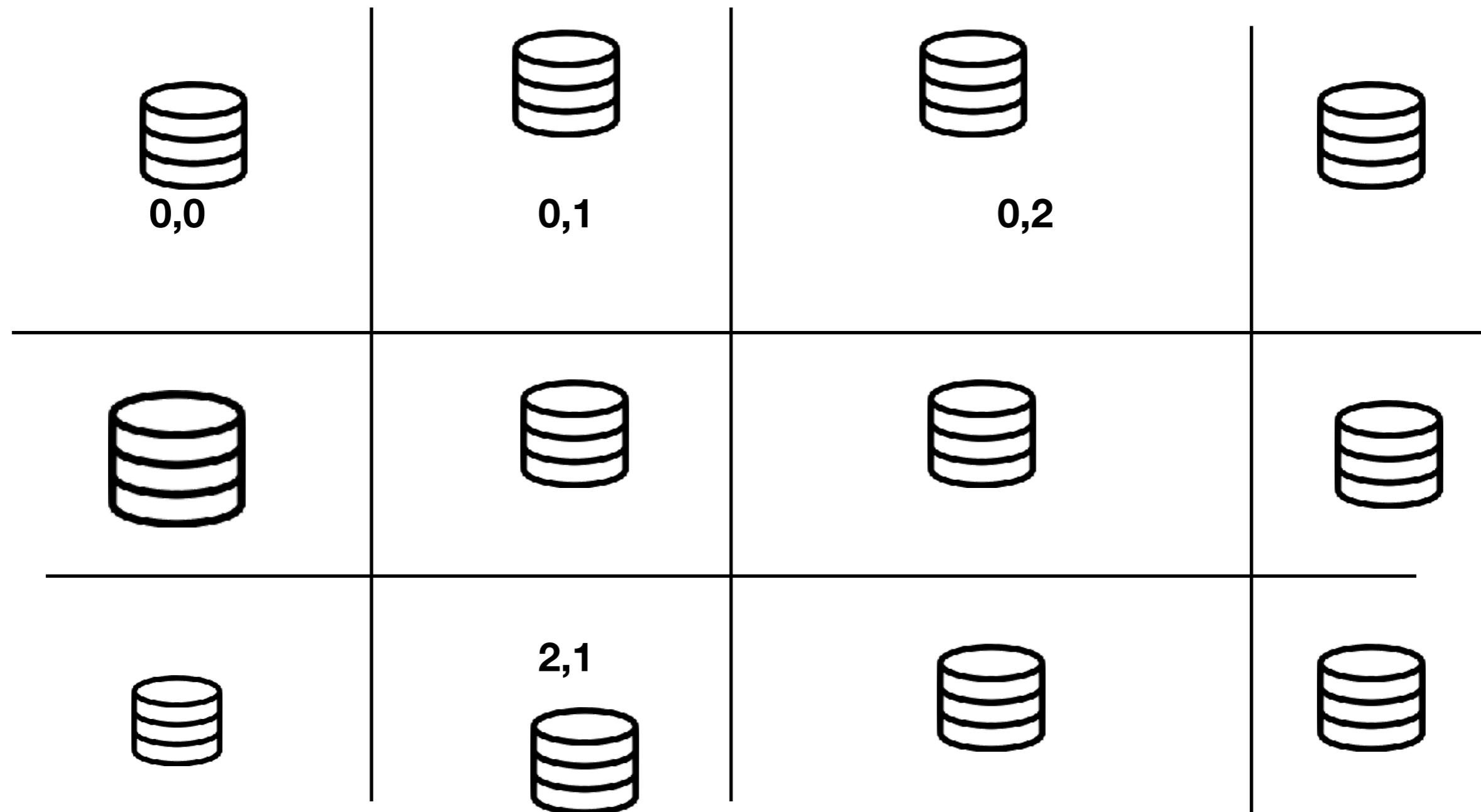
{Ordered, cust name, amount} , {itemcode, qty, price}

(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

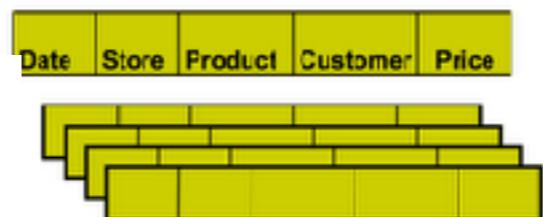
Column-oriented

Name	ID
John	001
Karen	002
Bill	003

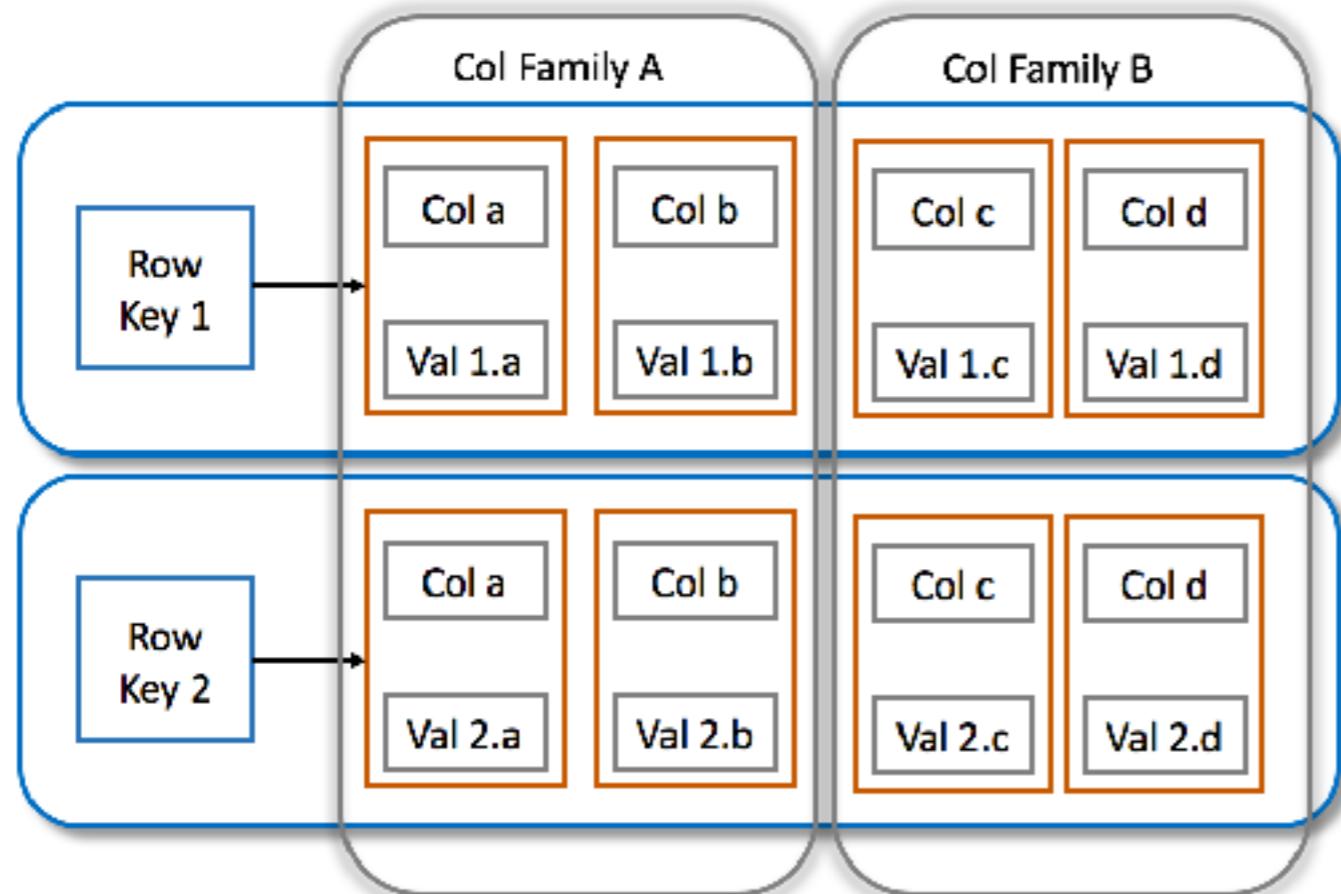
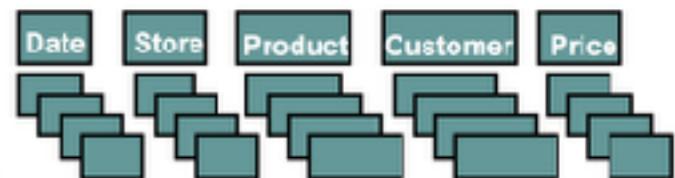
Grade	ID
Senior	001
Freshman	002
Junior	003

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

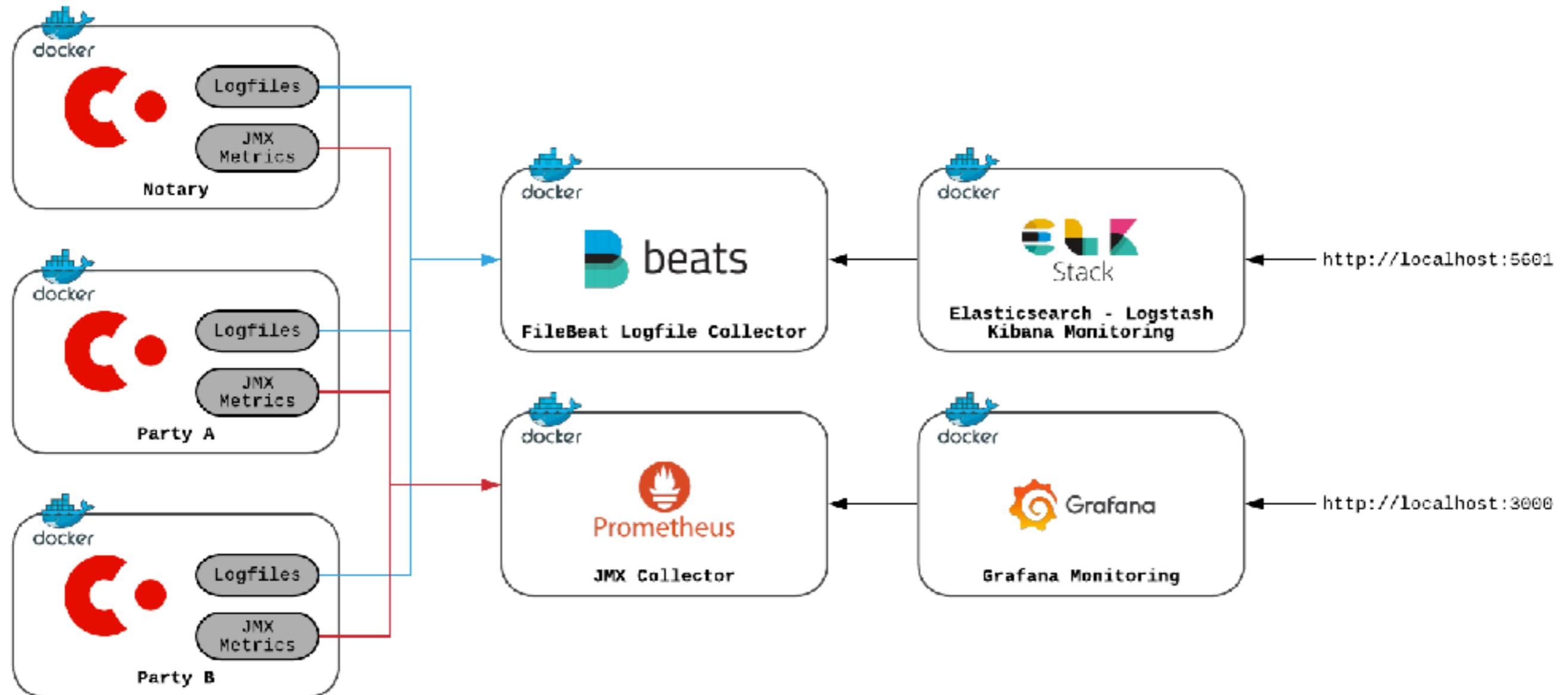
data warehouses

	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.



Classic Relational Databases

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

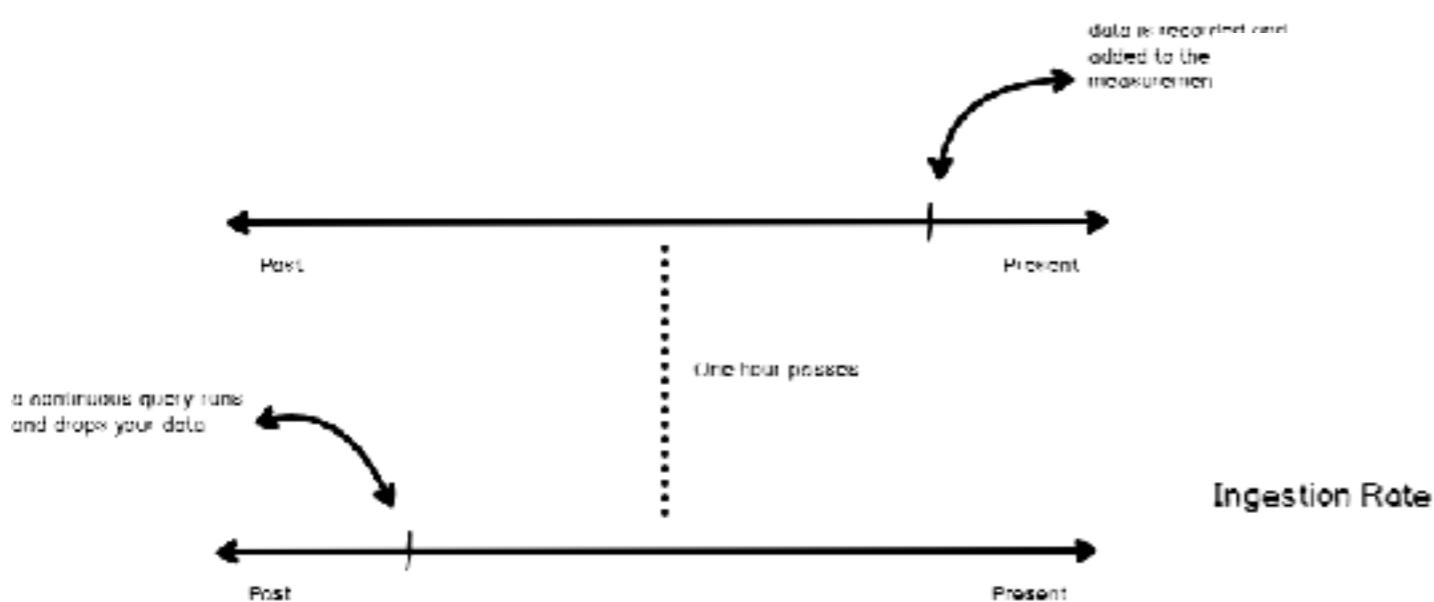
Data are multidimensional

Time Series Databases

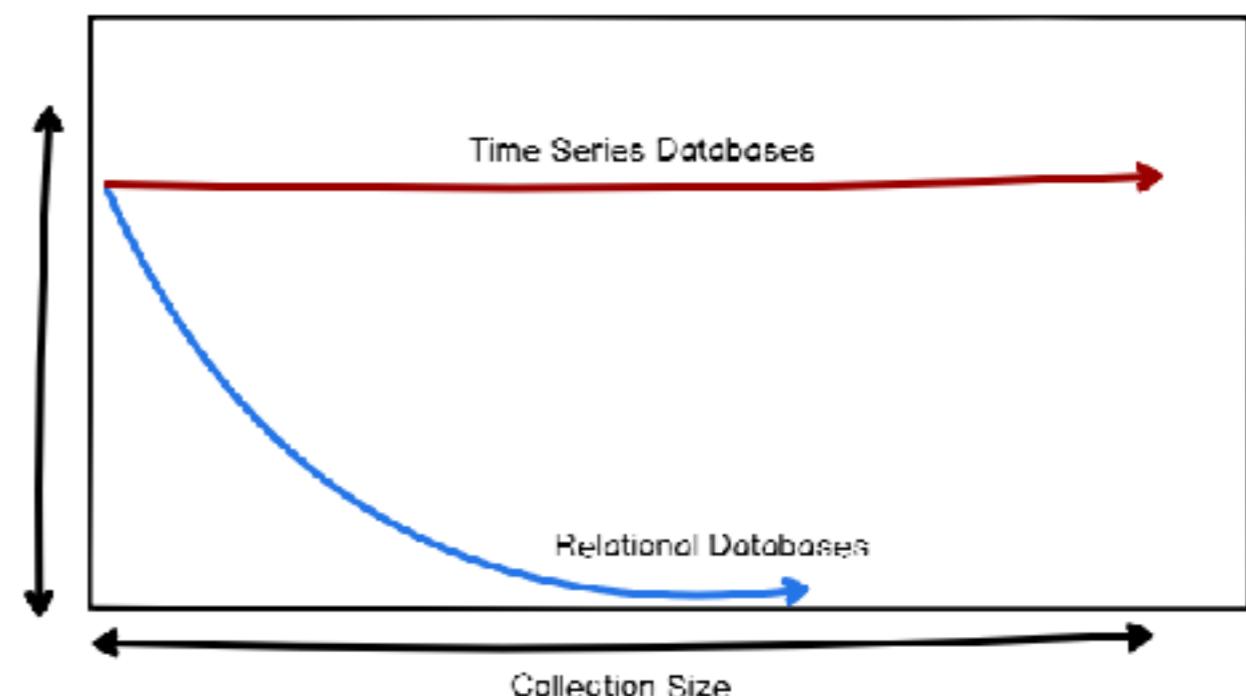
Sensor Temperature	Time
39.6	12/01/19 @ 11:12
11.2	12/01/19 @ 11:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

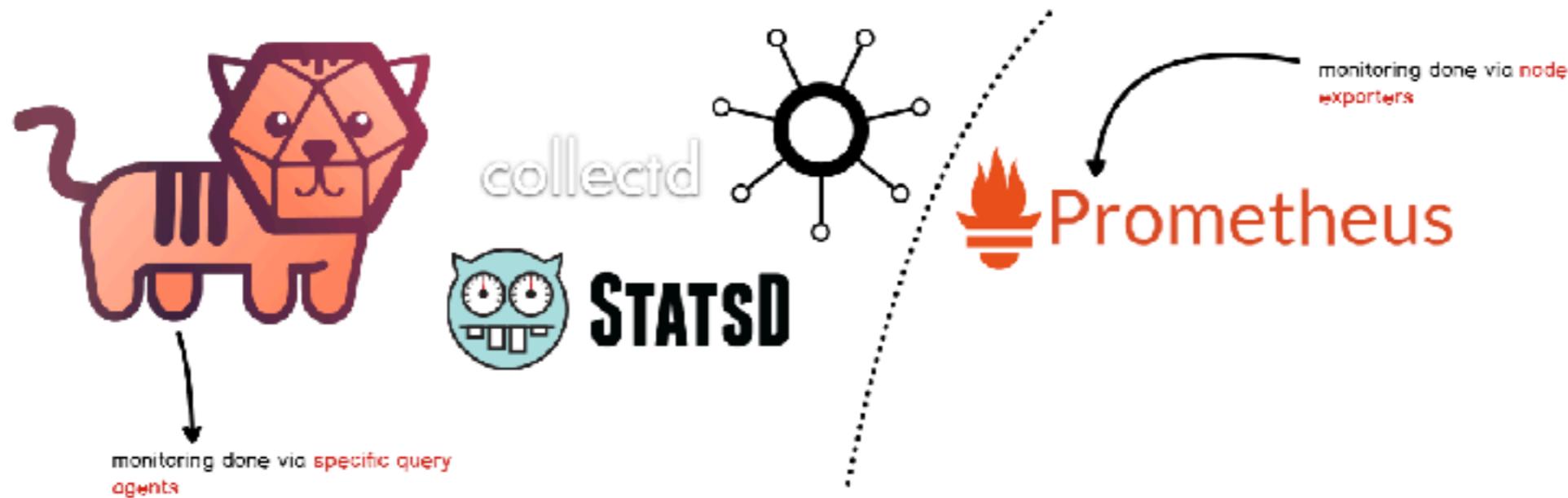
Case : retention policy = 1 hour



DBMS & TSDB Difference



Tools that 'produce' TSDB data



Tools that 'consume' TSDB



*Non-exhaustive list

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

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

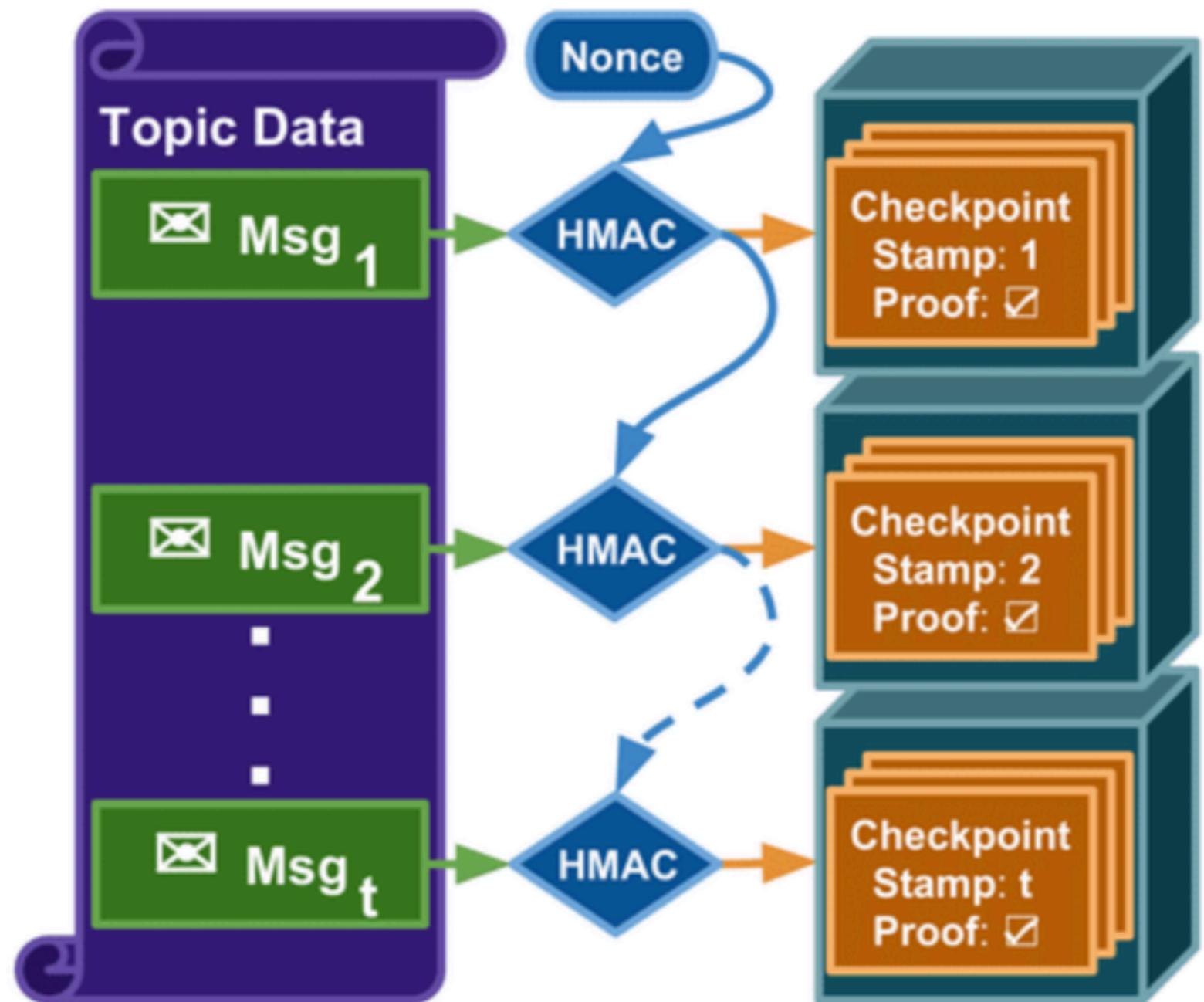
South Atlantic Ocean

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 }
```

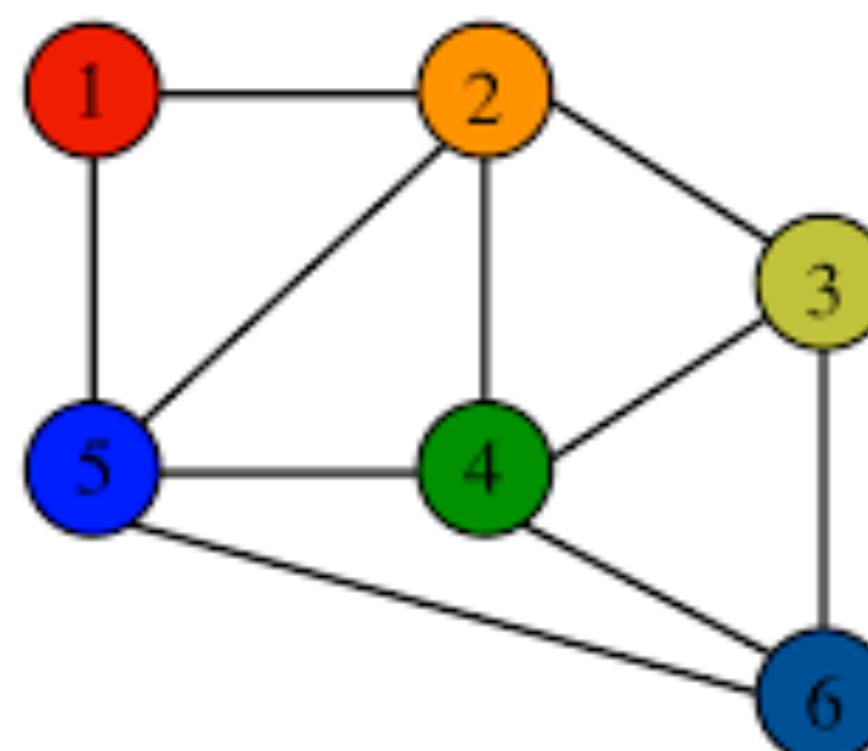
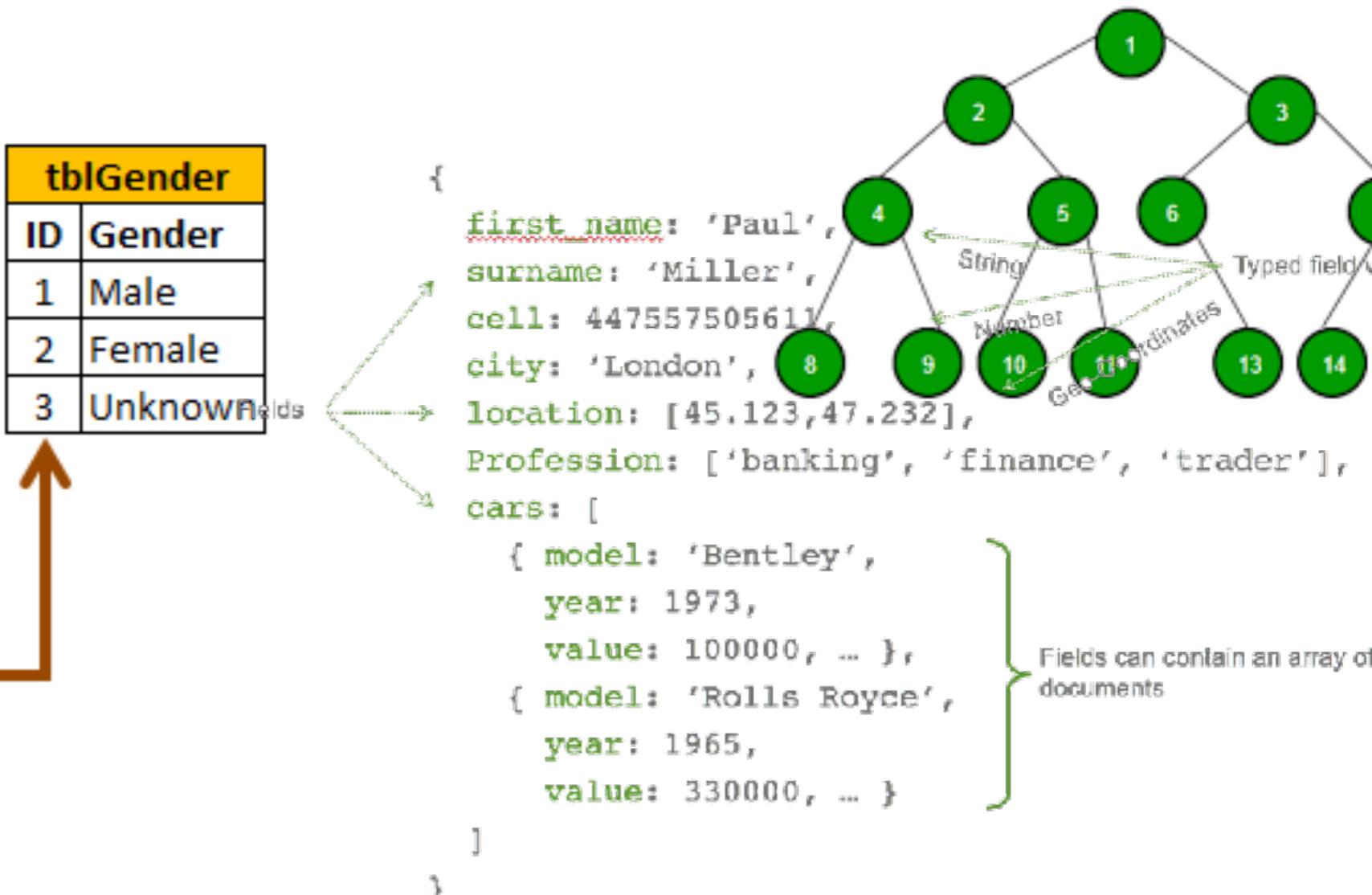
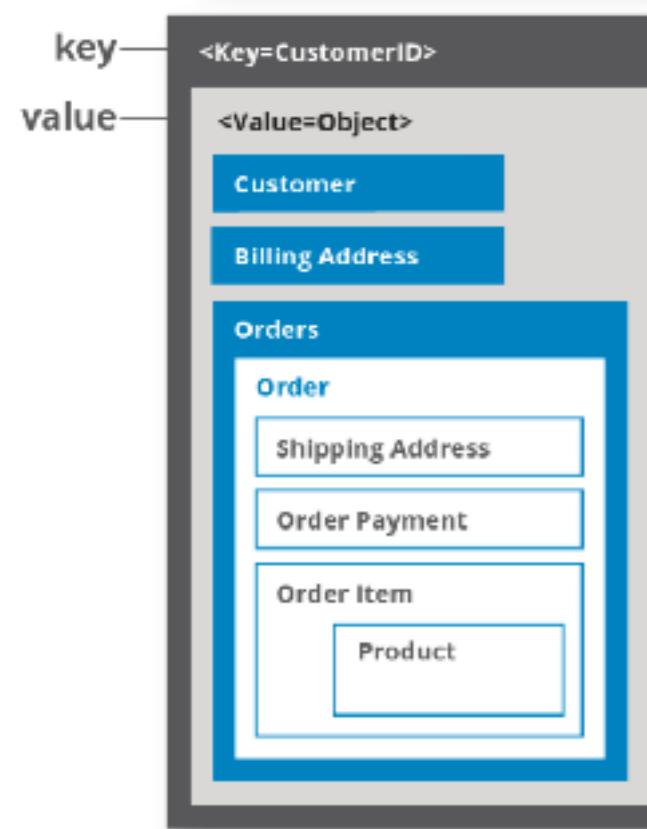


Rdbms (Referential Integrity)

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

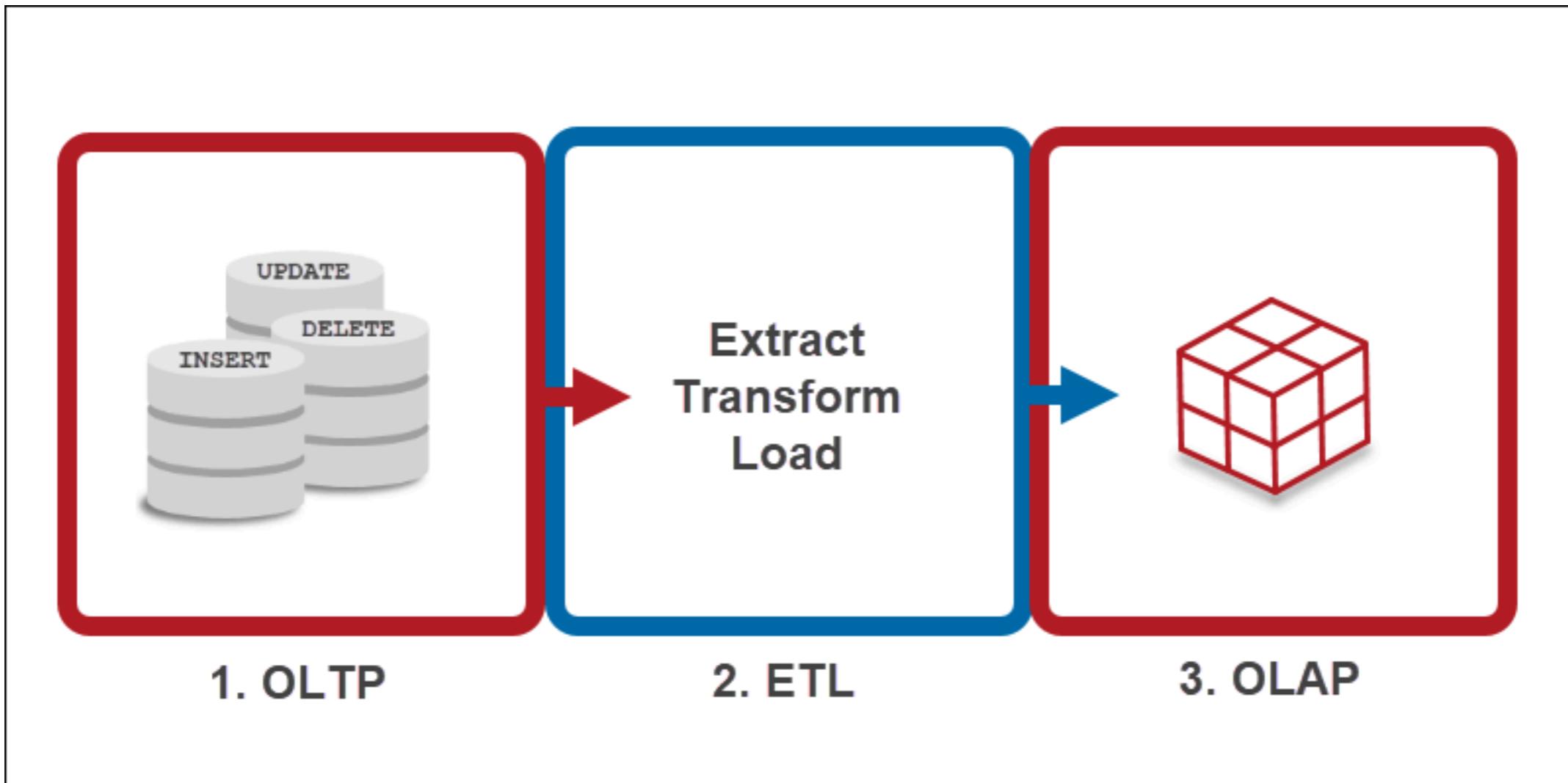
tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

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



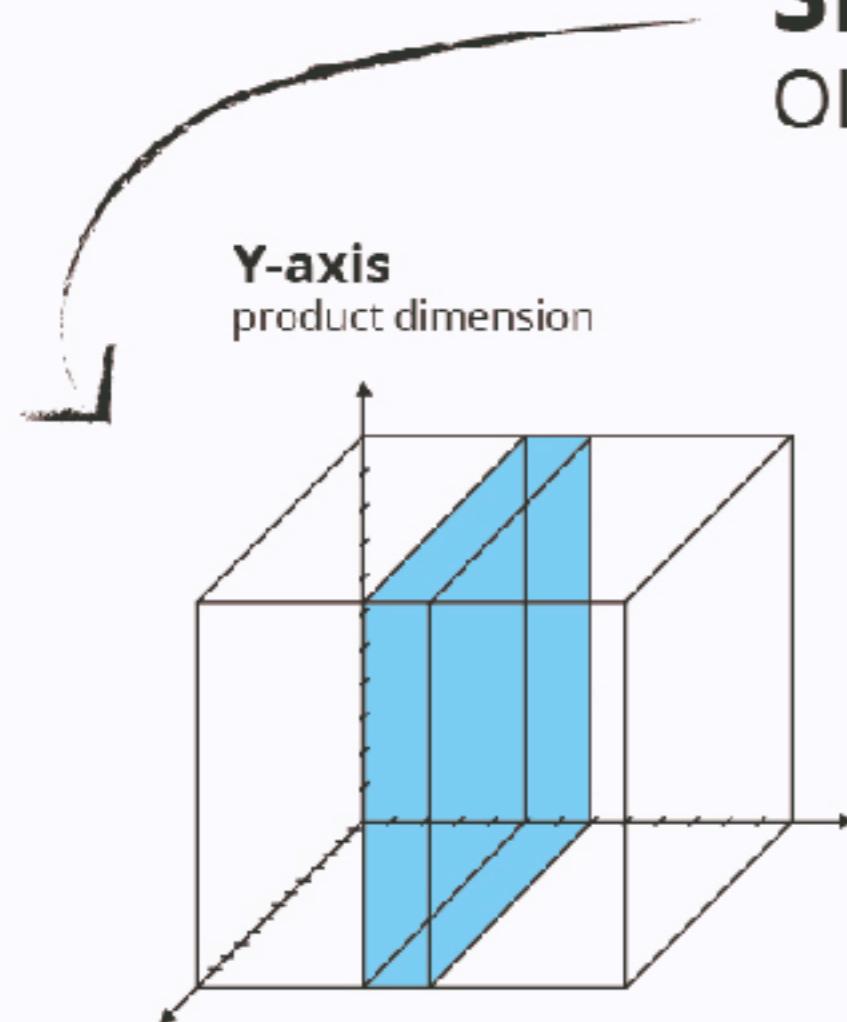
Wide Column

- 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

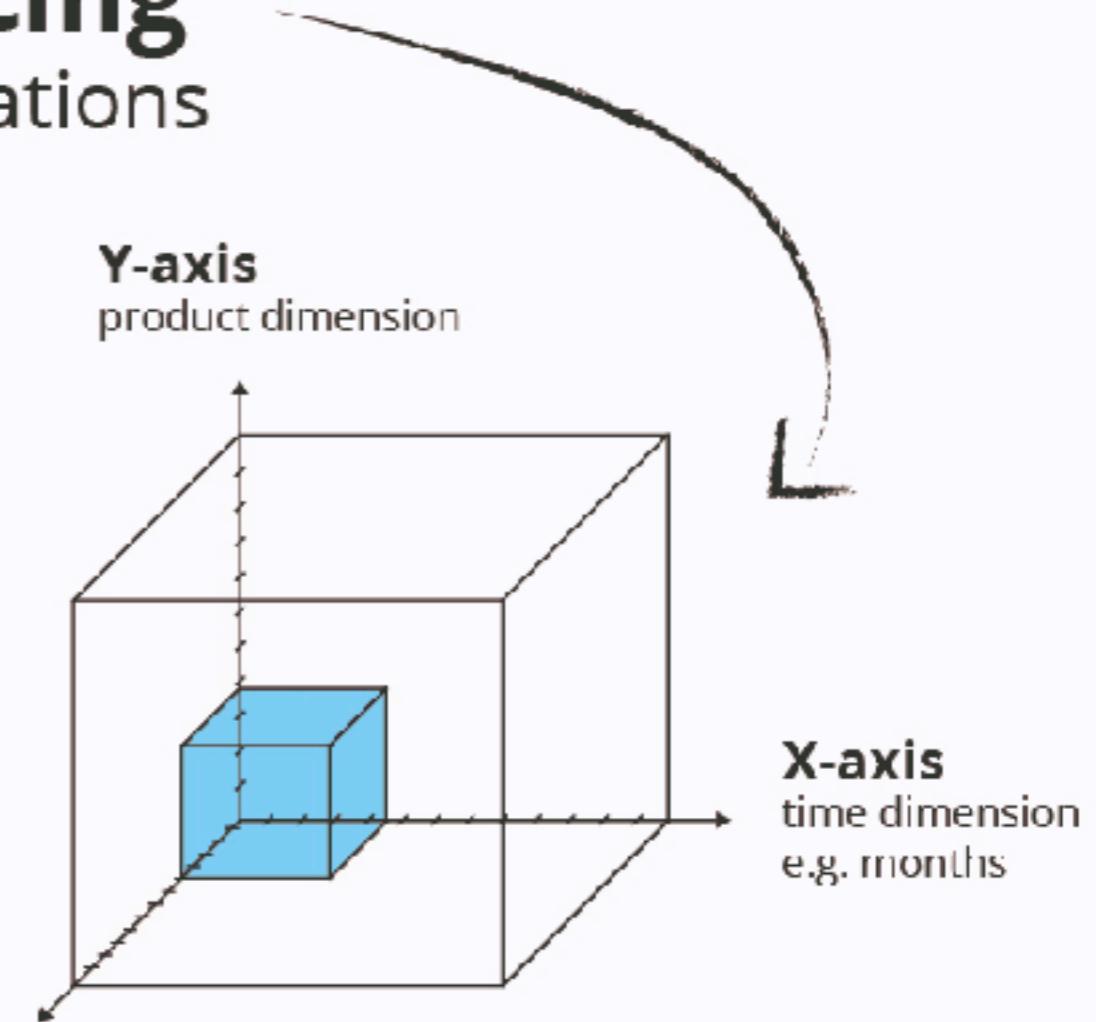


Slicing & Dicing

OLAP cube operations



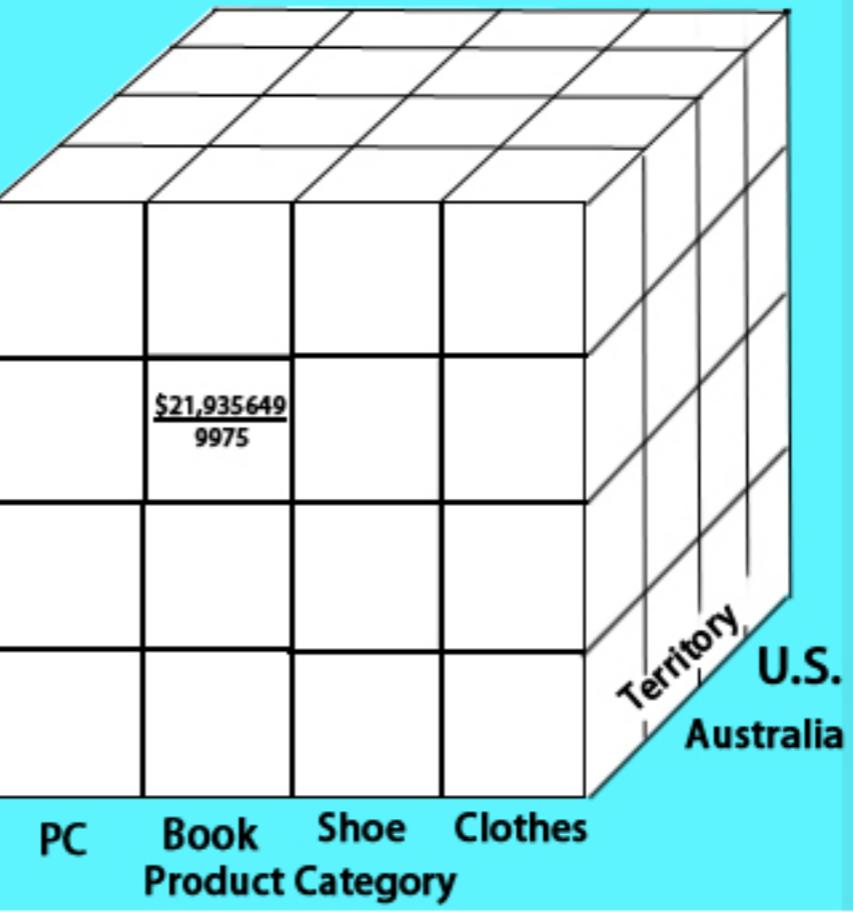
Z-axis
customer dimension

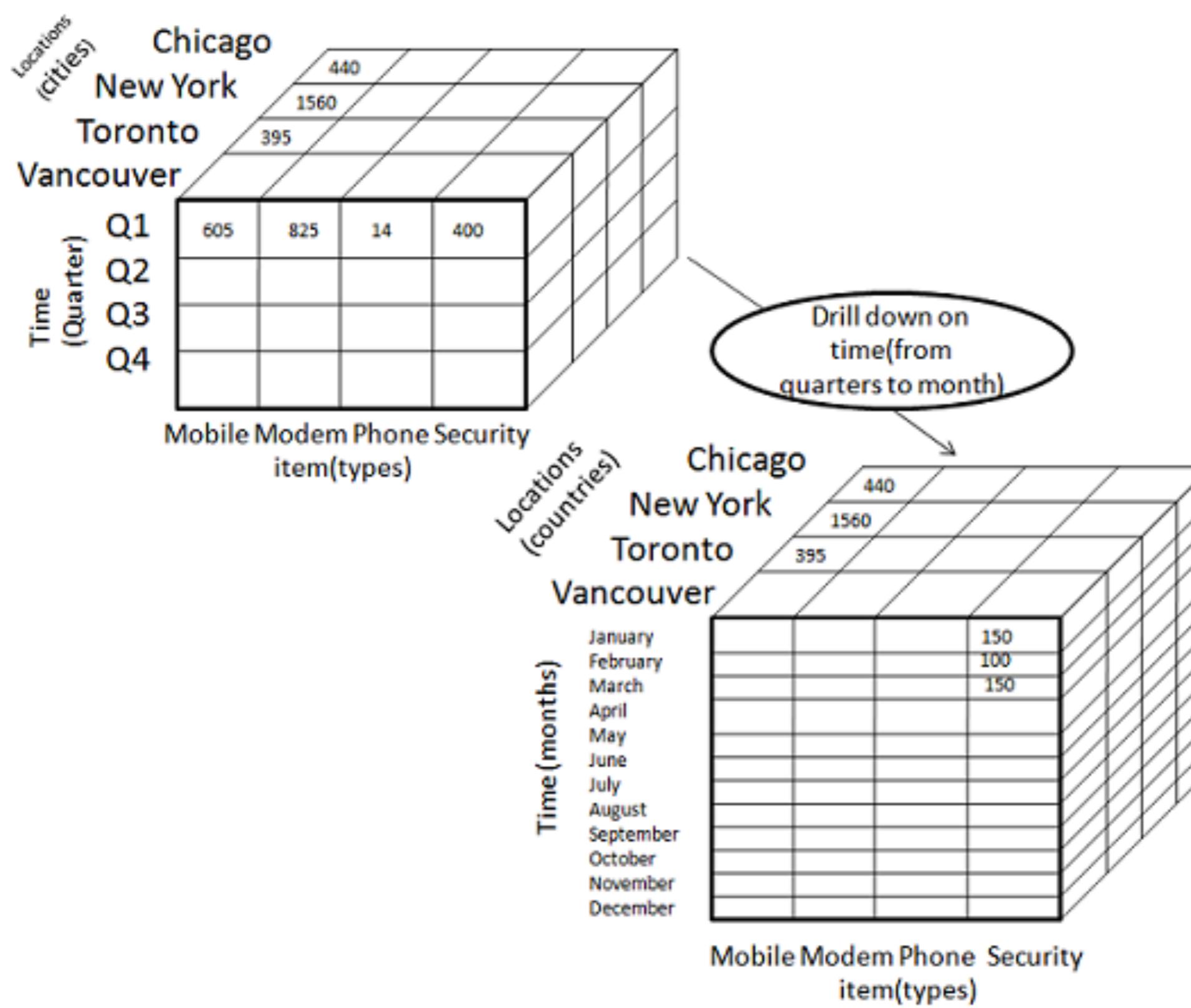


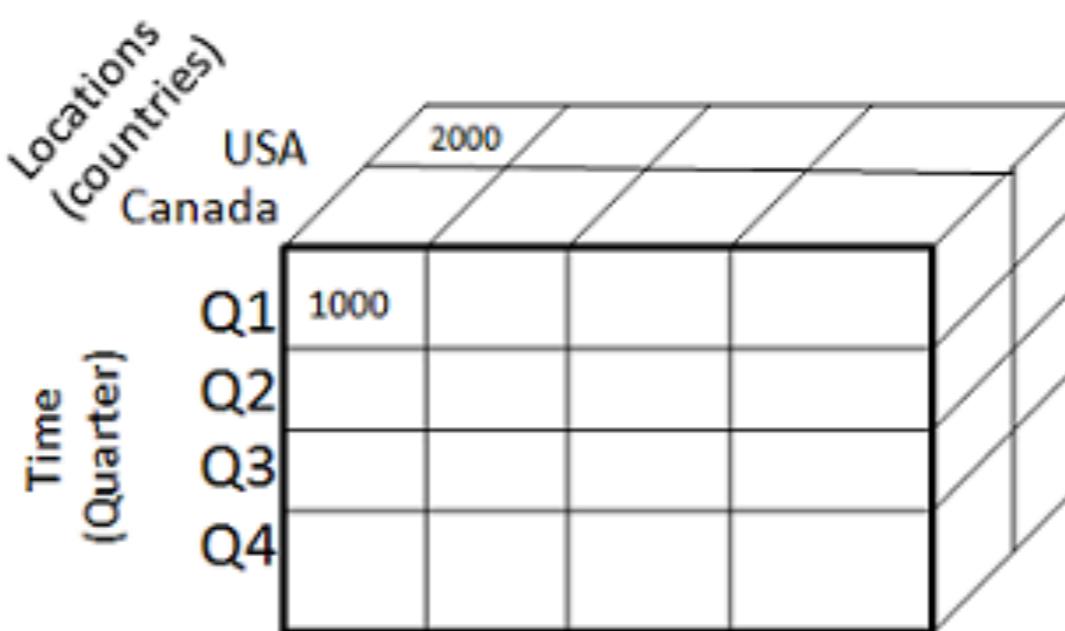
Z-axis
customer dimension

OLAP CUBE

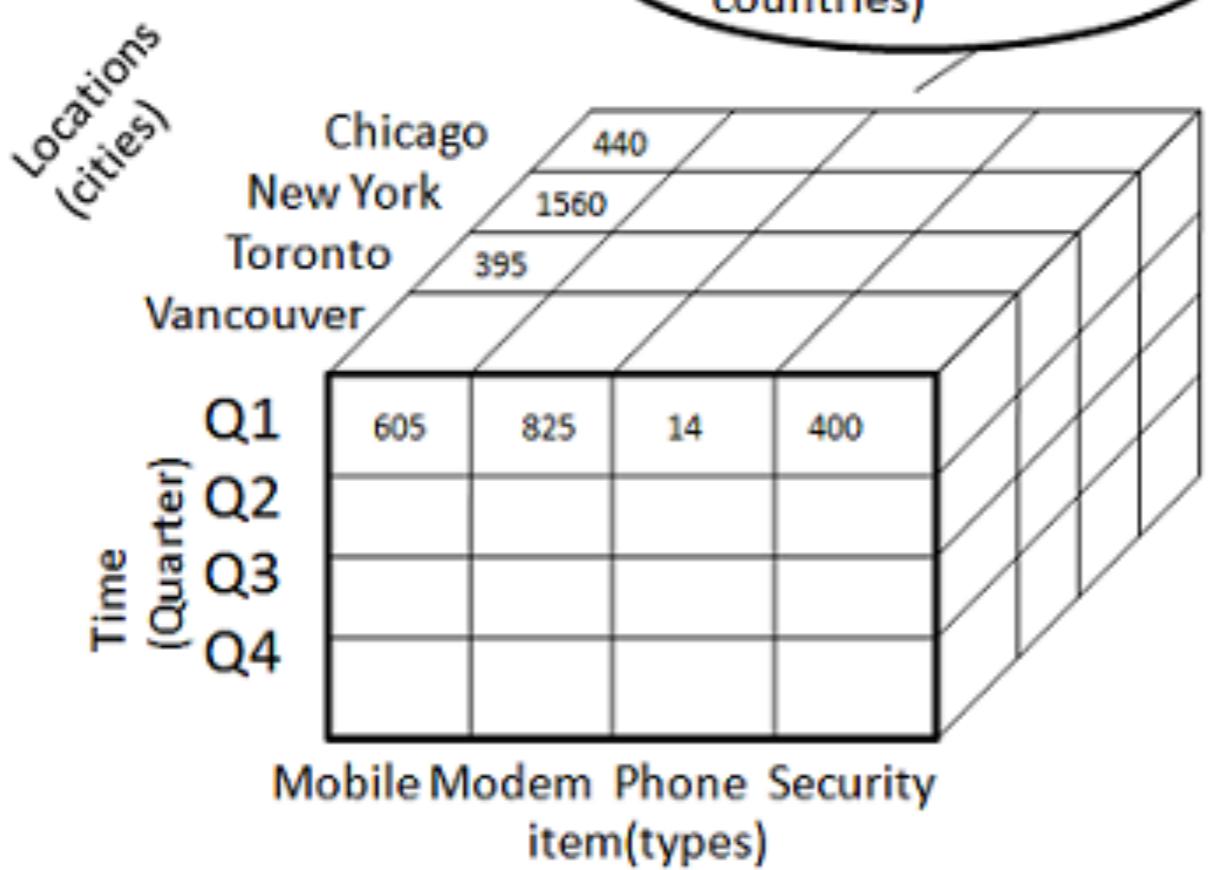
Quarters
Q1
Q2
Q3
Q4





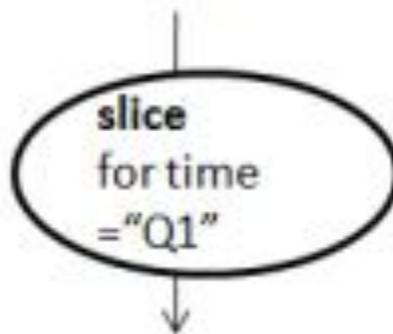


roll-up on location
(from cities to
countries)



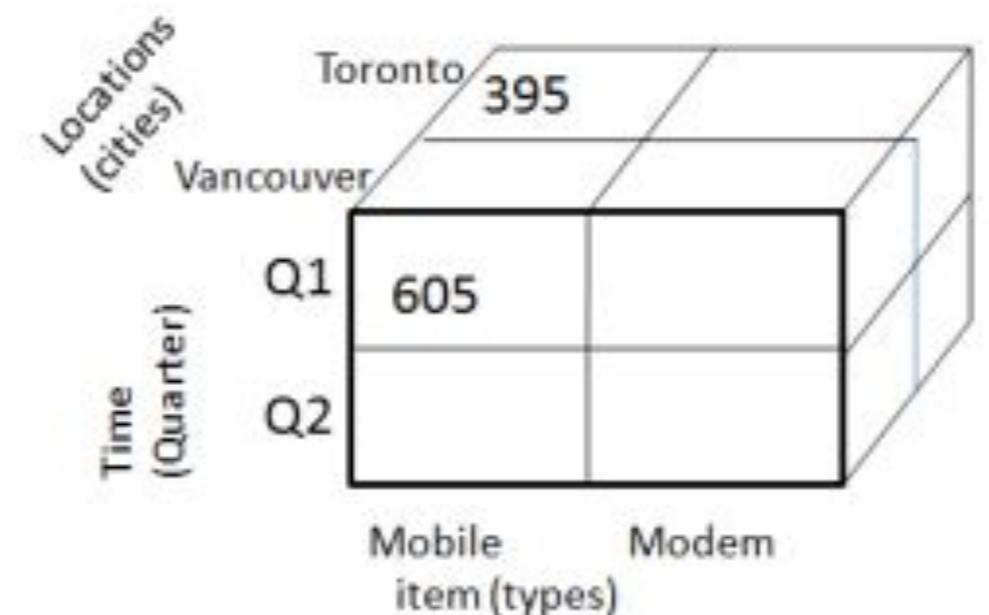
		Mobile Modem Phone Security				
		item(types)		item(types)		
Locations (cities)	Time (Quarter)	Chicago		New York		Toronto
		440		1560		395
		605		825		14
		Q1		400		
		Q2				

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)

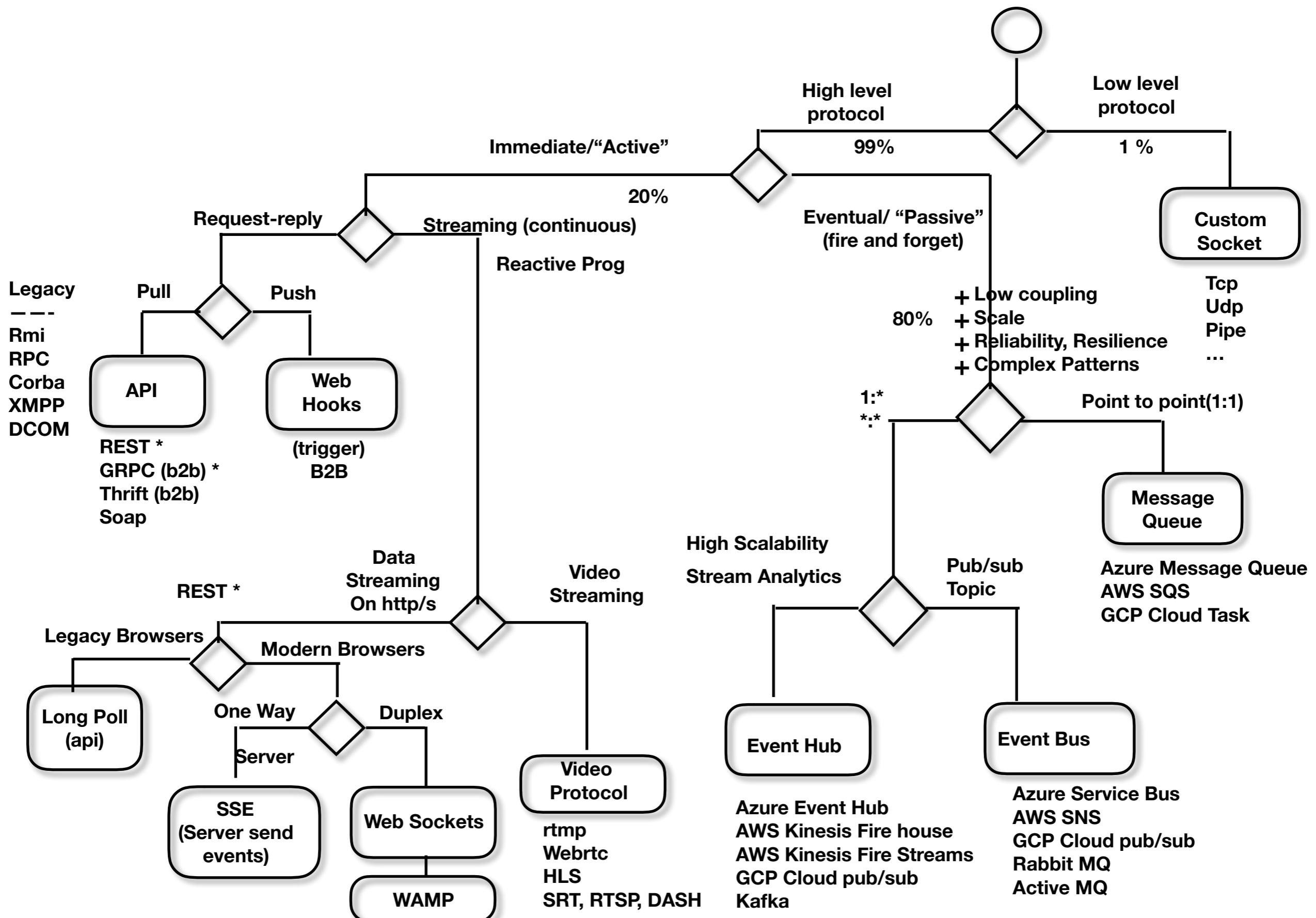
↓
Pivot
↓

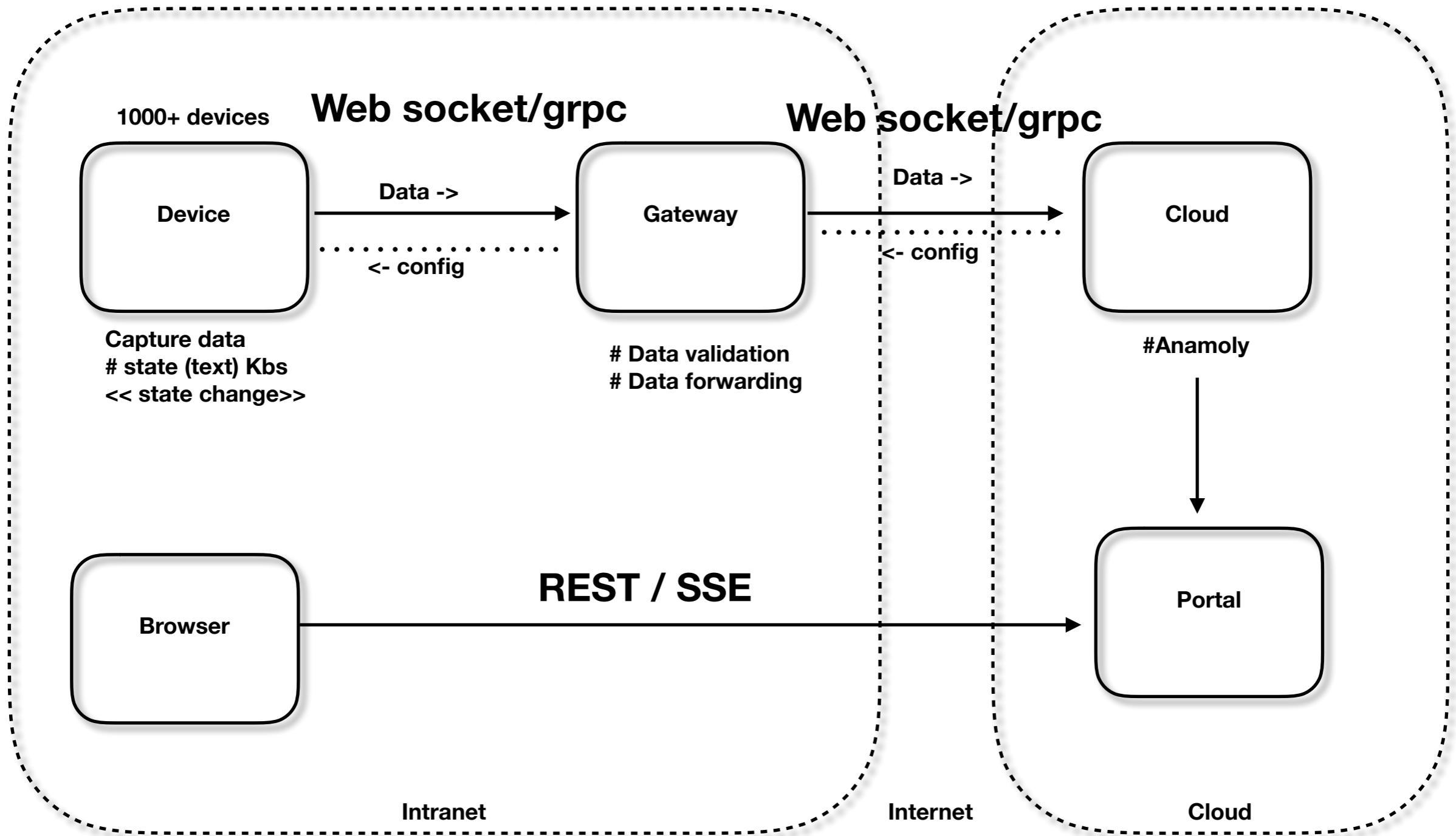
			605
			825
			14
			400

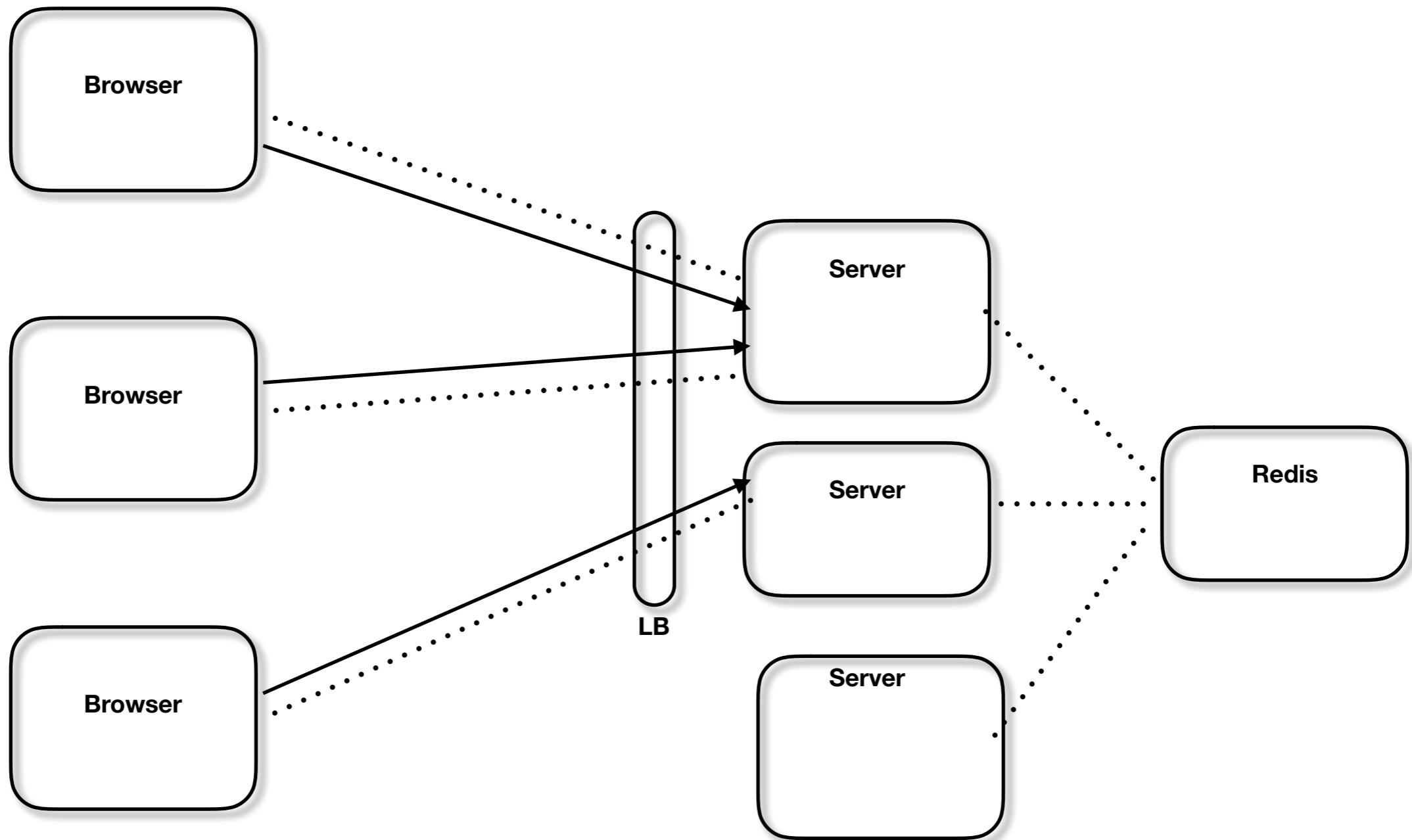
Mobile Modem Phone Security
Item (types)

Chicago New York Toronto Vancouver
Location (Cities)

Choose Communication protocol

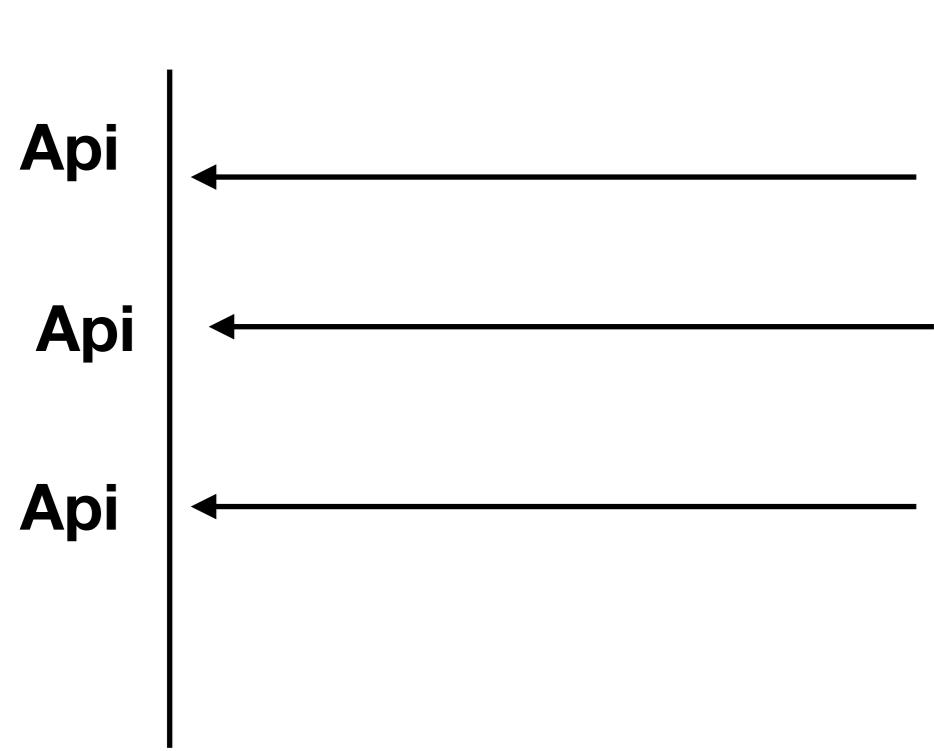
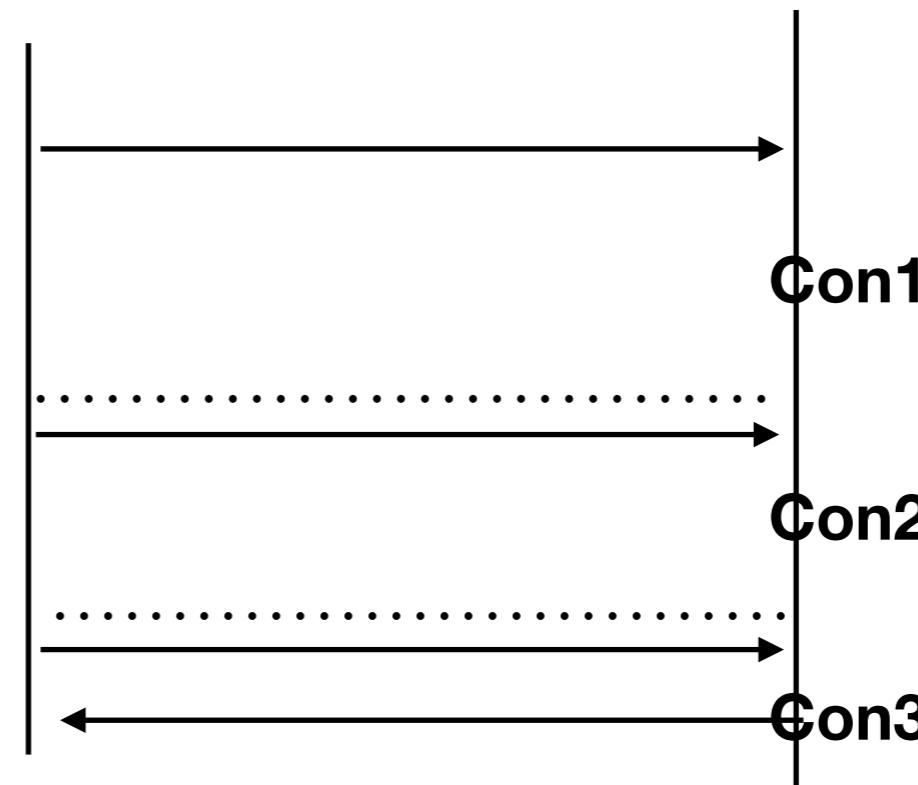
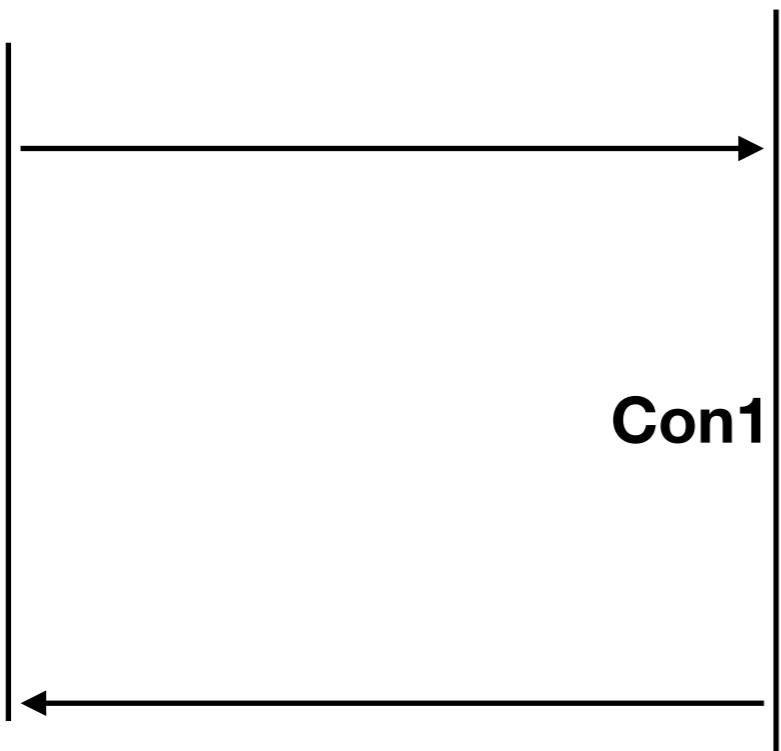








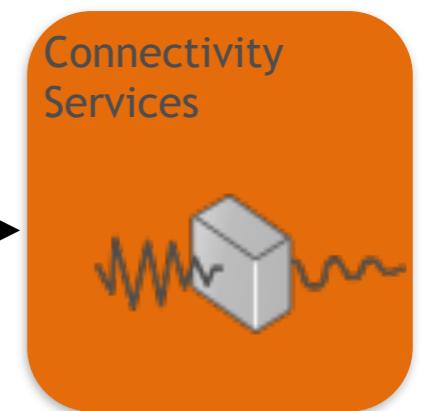
SSE





Communication
Services

Connected protocol
(REST, WS, GRPC , Thrift)

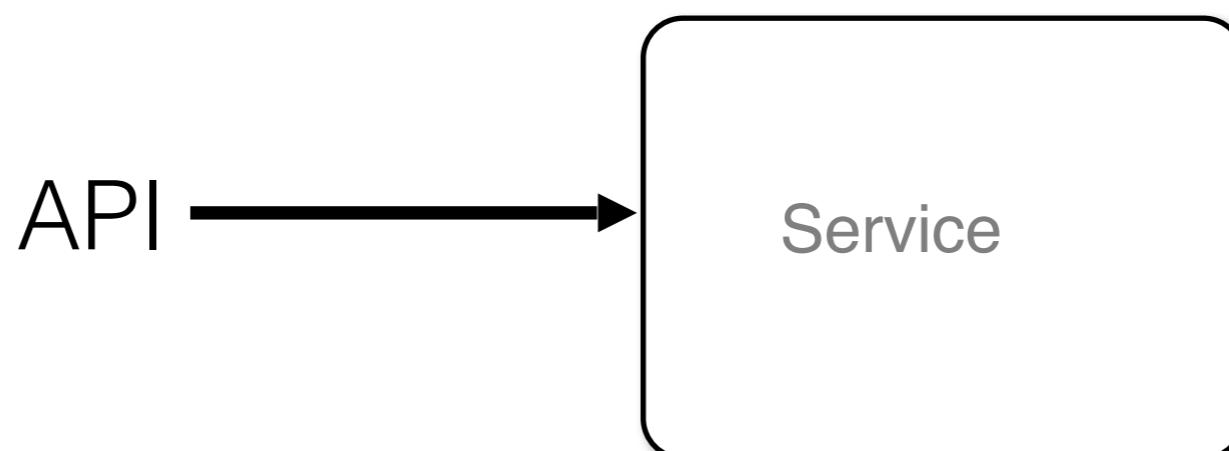


Connectivity
Services

Message protocol
(AMQP, MQTT, ...)

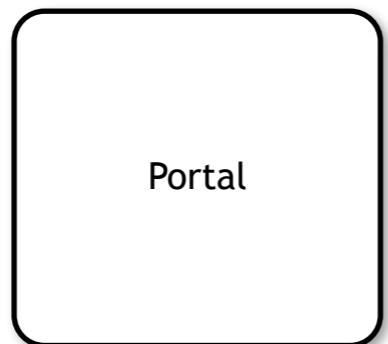


	Connected	Msg database (ACID)
Scale	--	++
Coupling	--	++
Reliability	--	++ (retry)
Developer effort		
Consistency	Immediate	Eventual - - (SAGA)
Debugging	+	-
Delivery Order	Ordered ++	Unordered * - -
Duplicate	--	--
Dev Ops Effort		
Exception Handling	++ (error response)	- - ?
Result	Response	? (one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



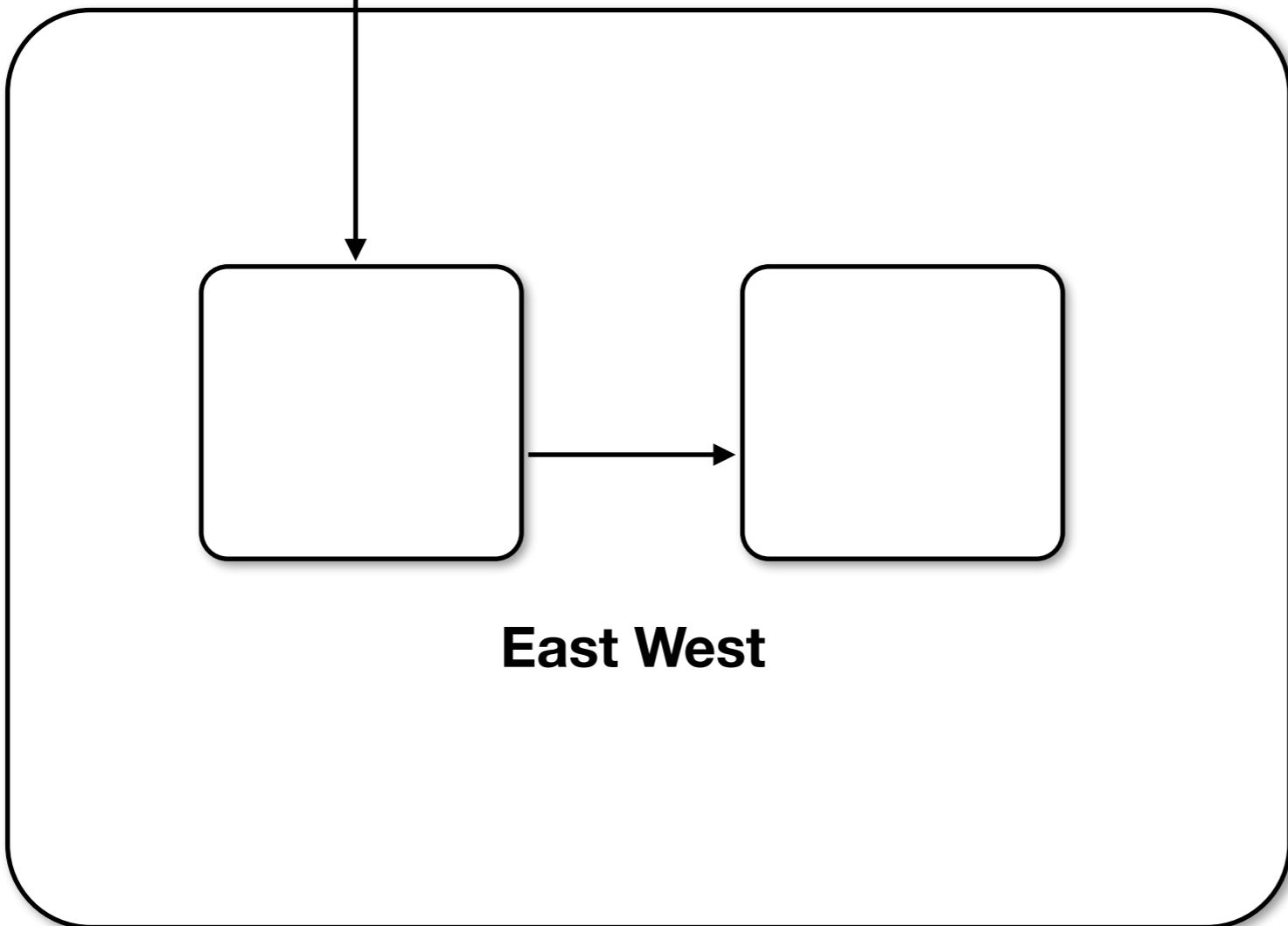
The diagram illustrates the relationship between an API and a Service, with the Service containing a large empty area for notes.

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

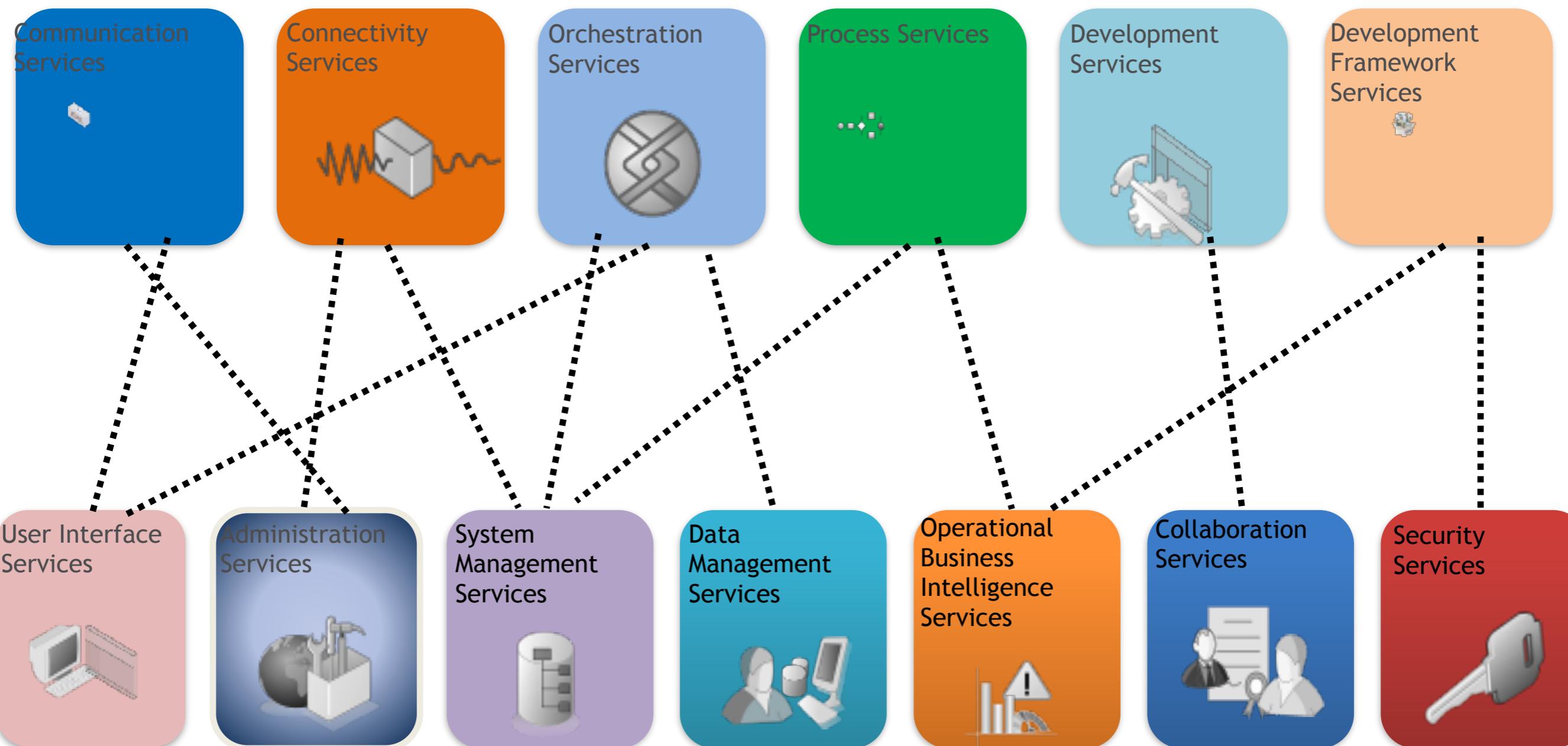


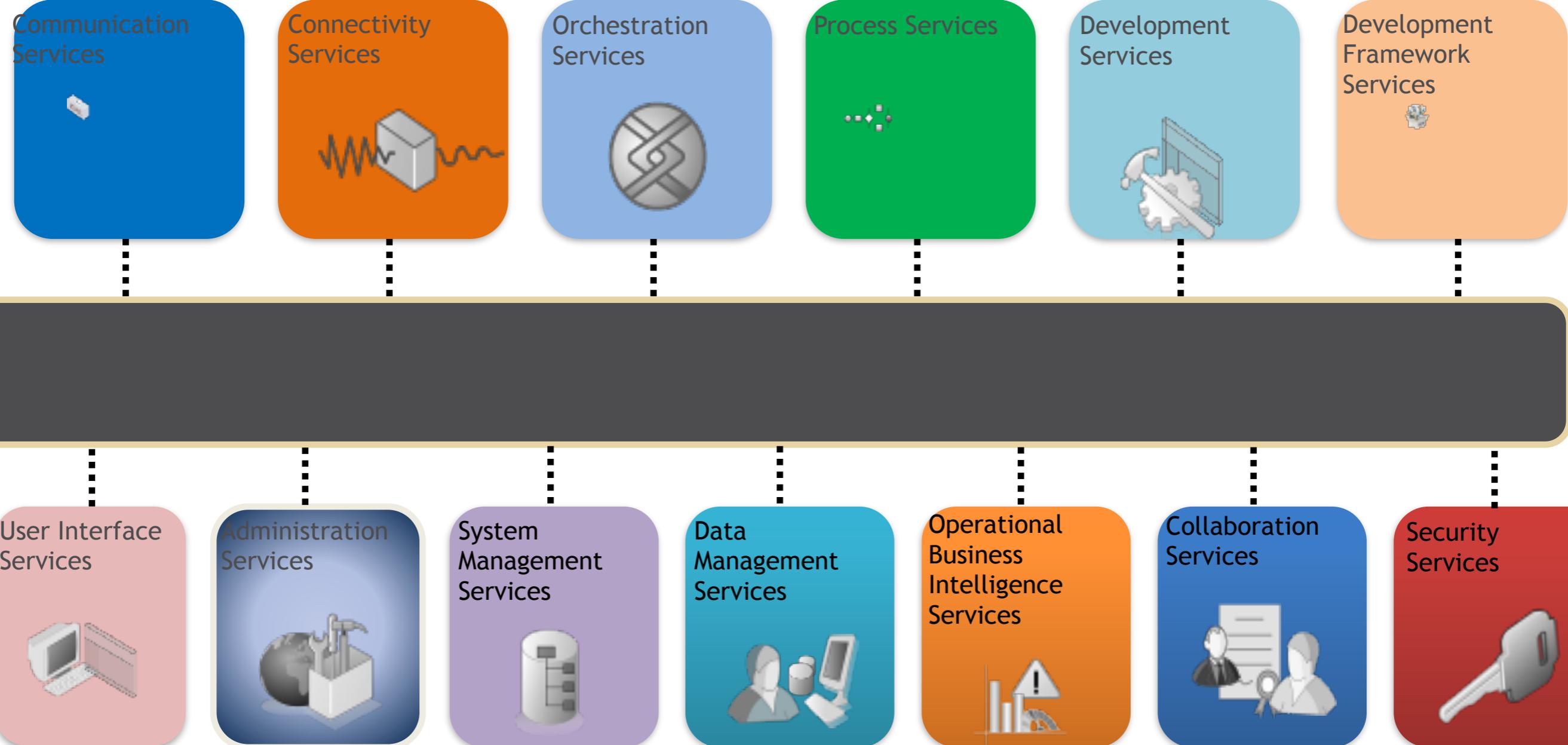
Portal

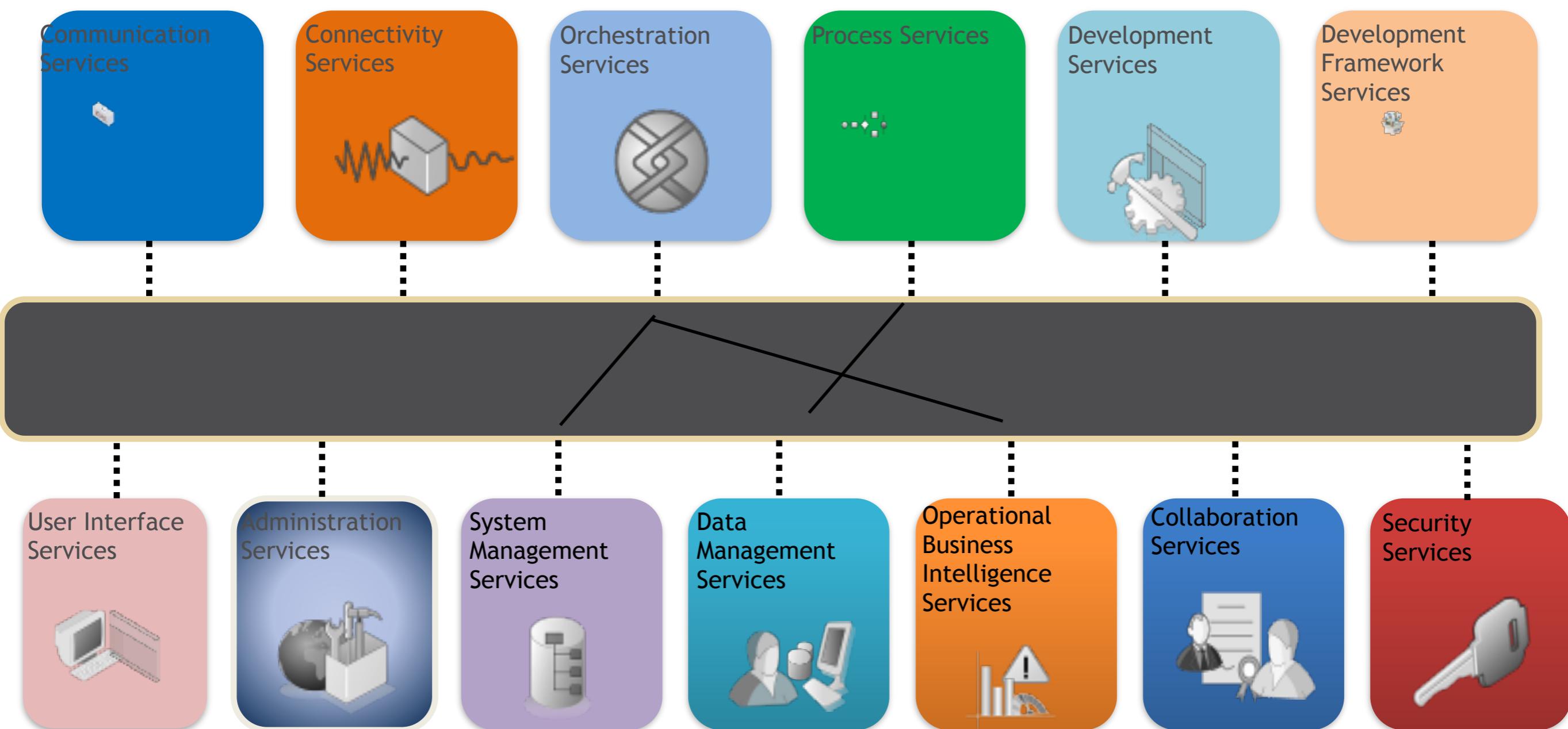
North south



East West







Event vs Command

Post

Pre

OrderCreated



CreateOrder

InvoiceCreated



OrderCanceled



OrderCreated



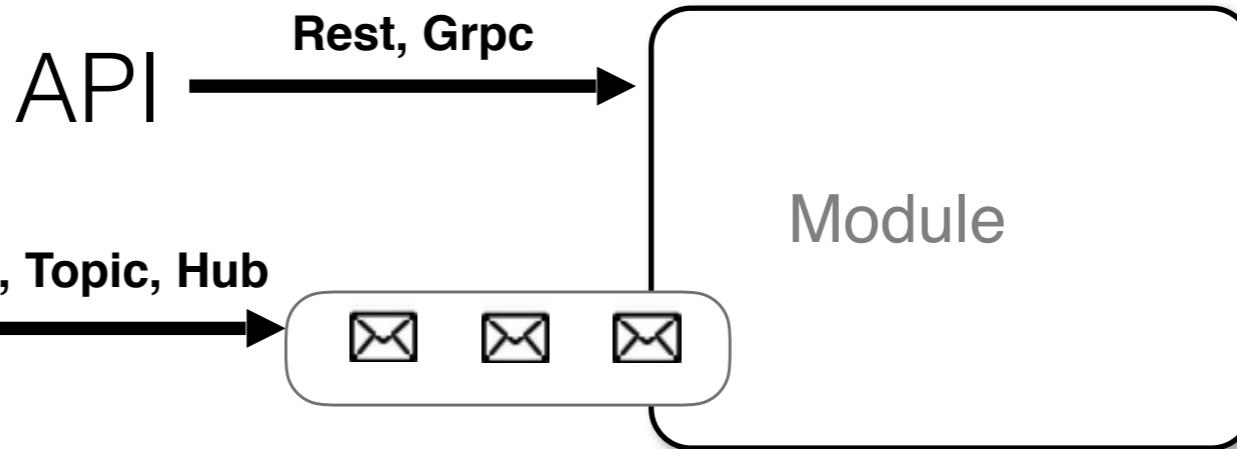
OderCanceled



OrderCreated

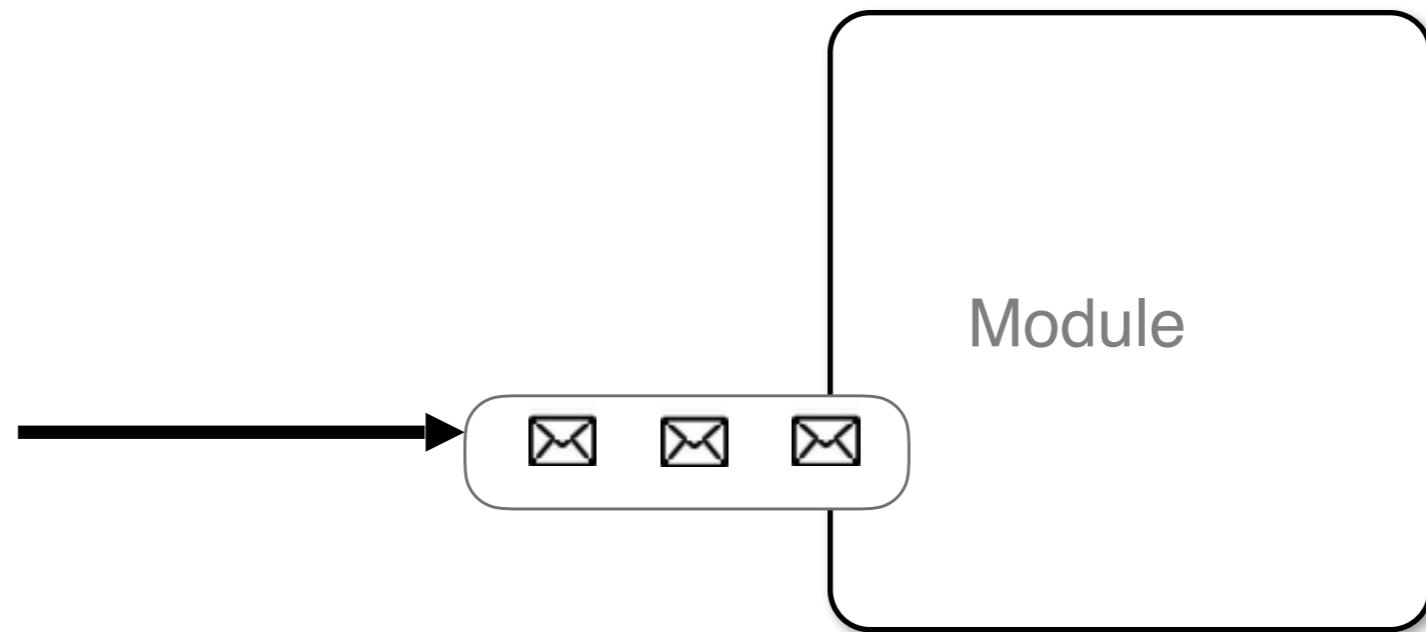


* Message

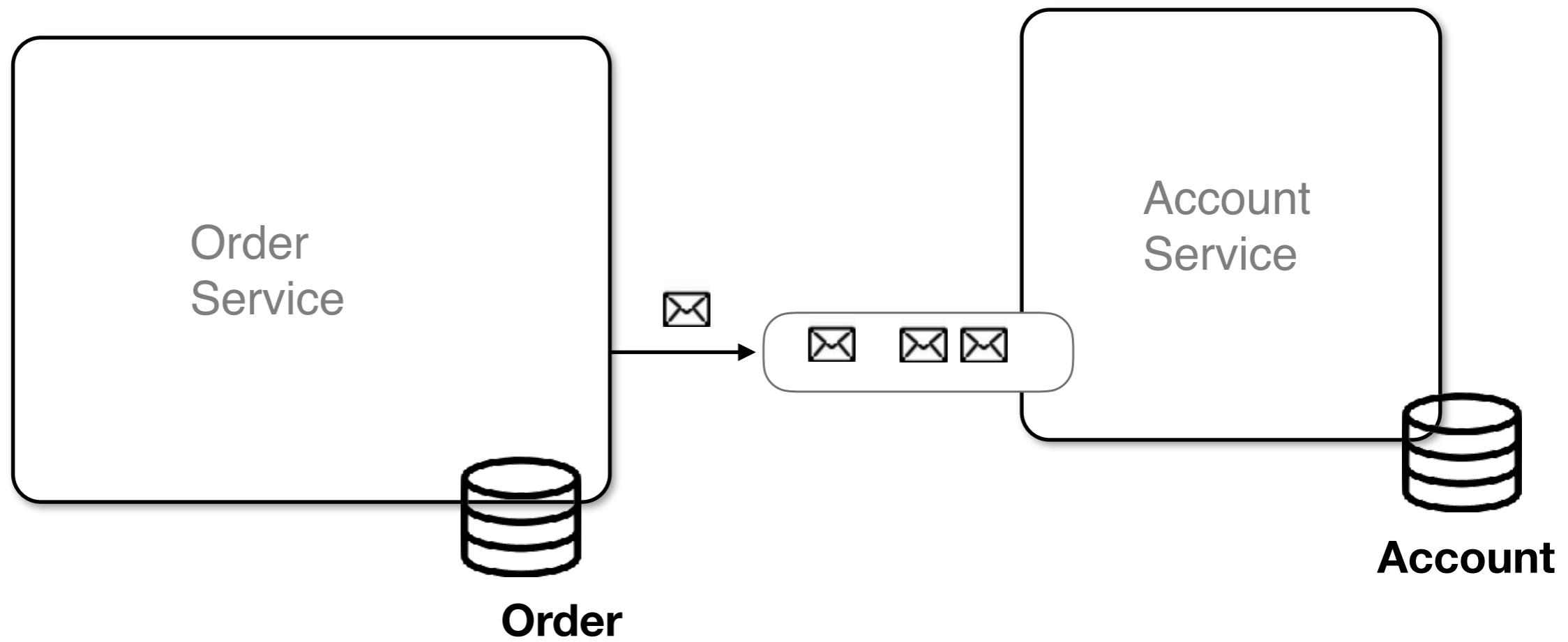
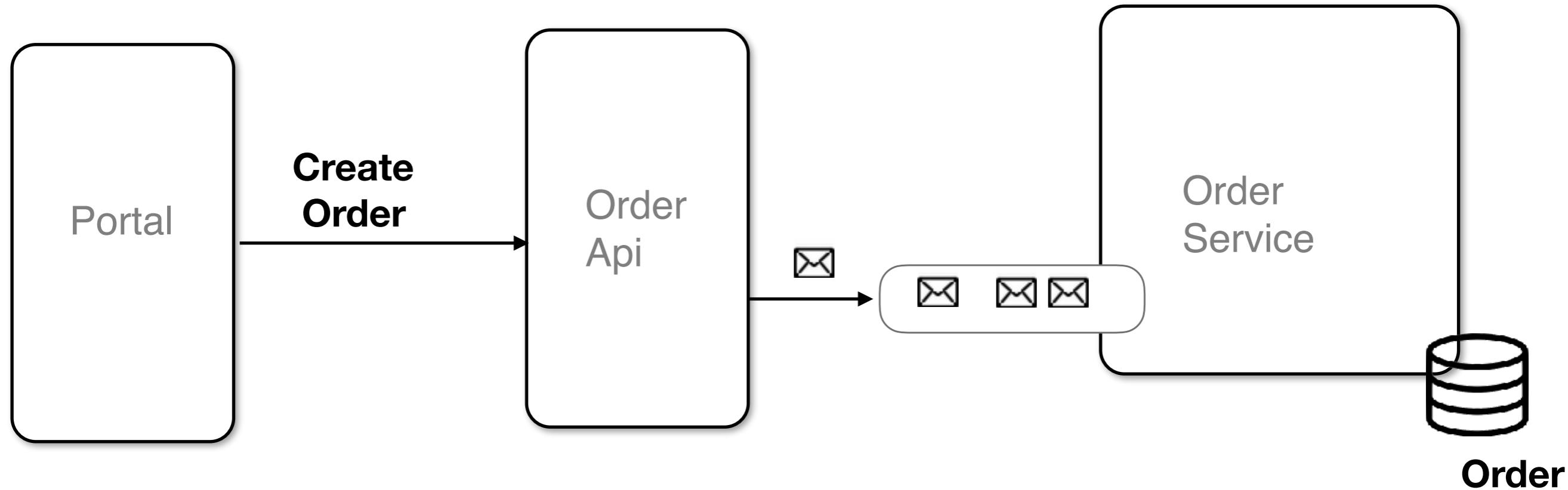


	<< API >>	<< Message >>	
Ordering	Ordered (+)	Unordered(-)	
Duplicate	Yes (-)	Yes(-)	Idempotency
Protocol	2 way (+)	One way (-)	
Resilience (recover)	No (-) retry logic	Yes (+)	
Connection	Always connected	Occousionaly connected	
Scalability	--	+++	
Consistency	Immediate (++)	Eventual (- -)	
Load Leveling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

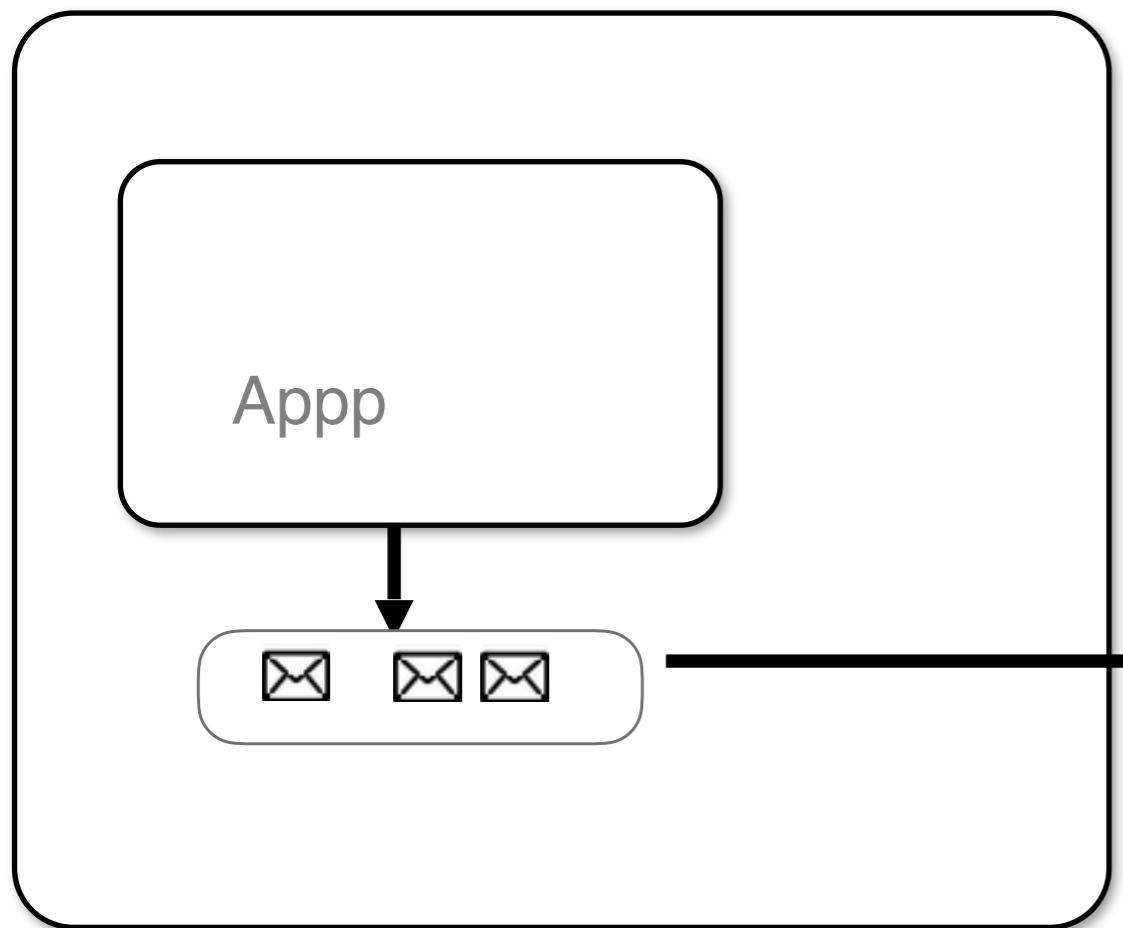
* Message



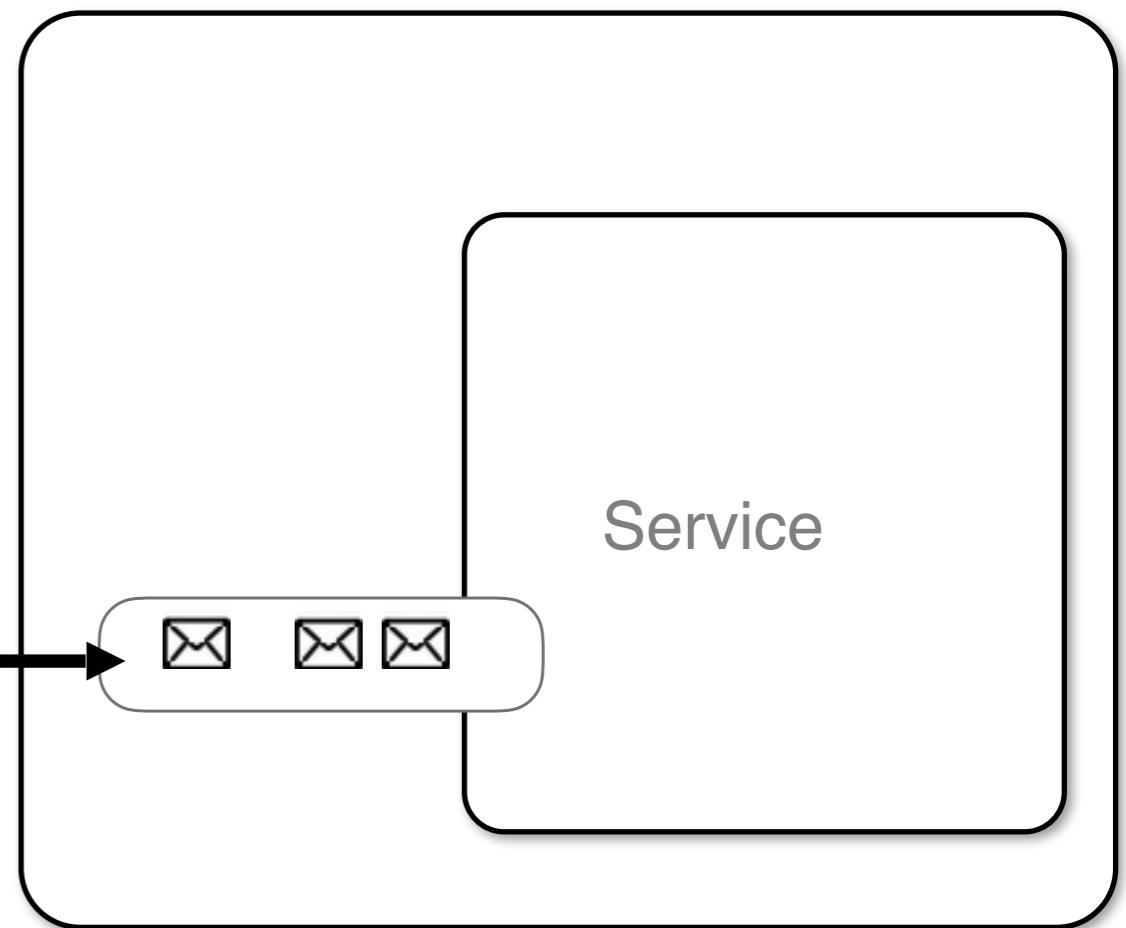
```
fun(){  
    while(true){  
        Msg m = GetMessage();  
        Run domain Logic  
        m.ack();  
    }  
}
```

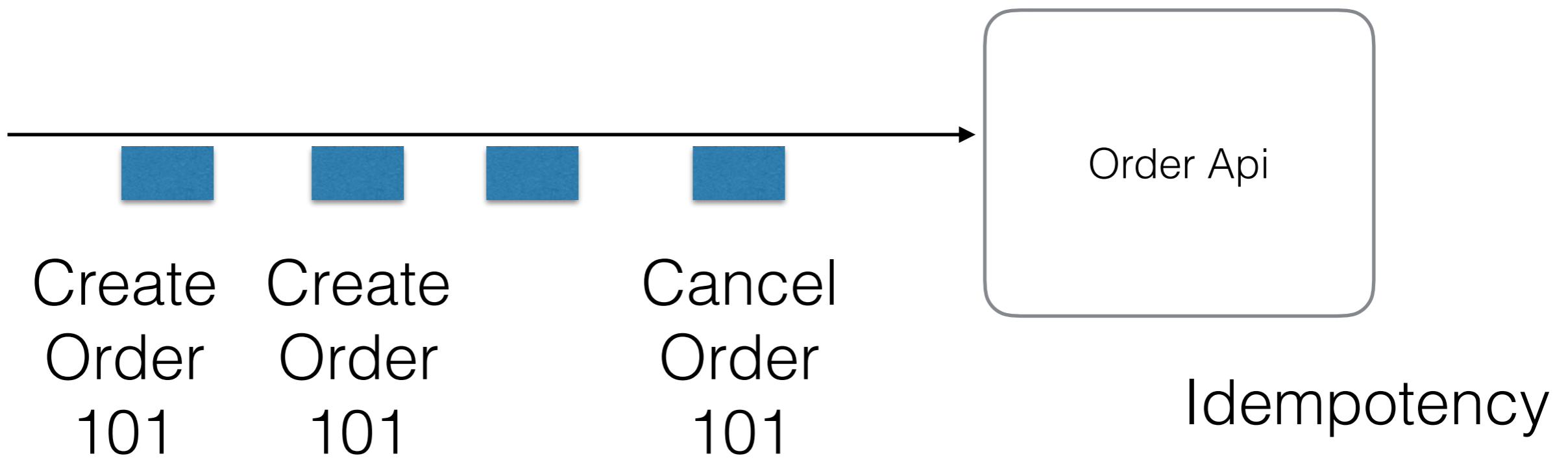



Client



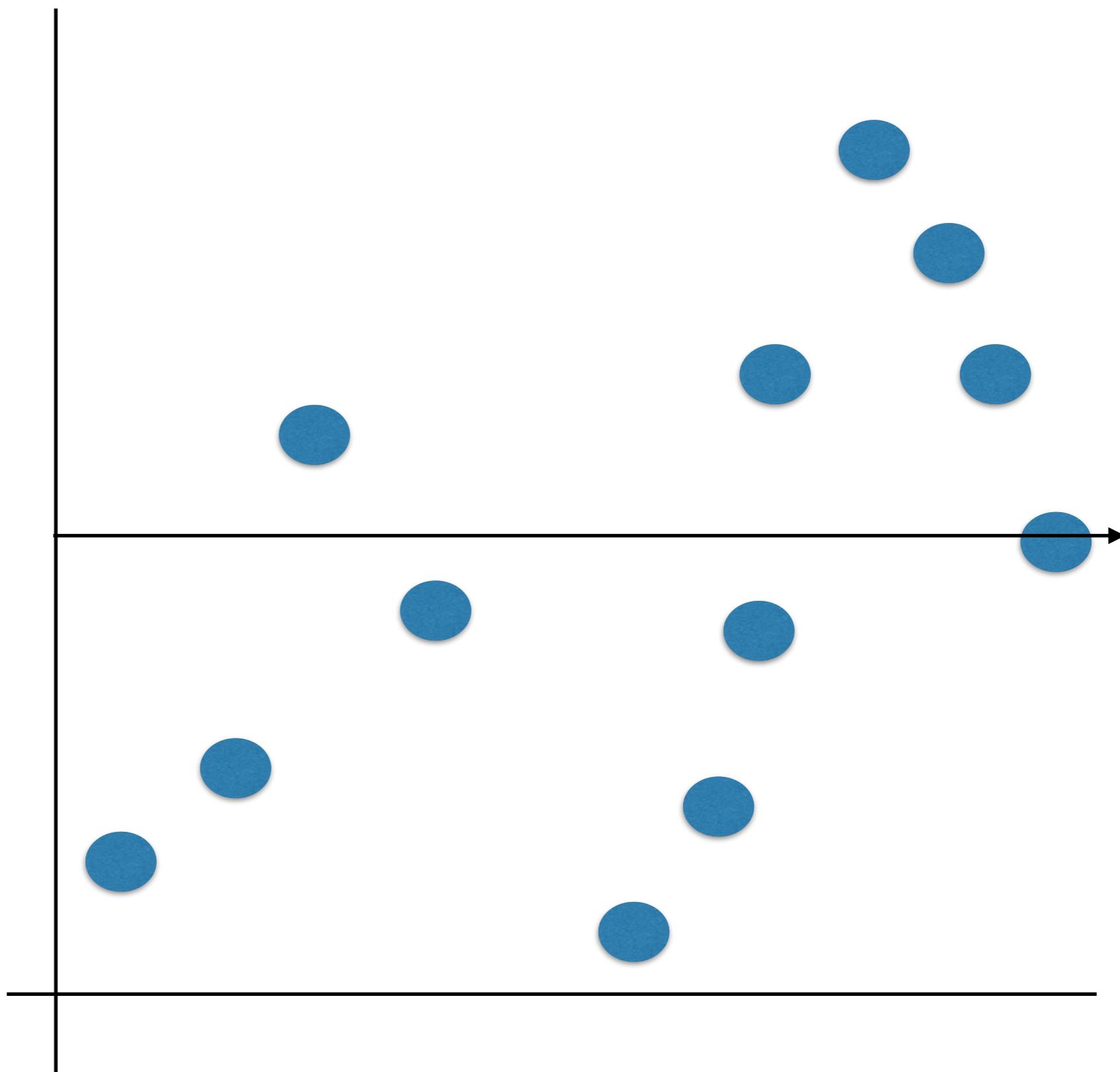
Server



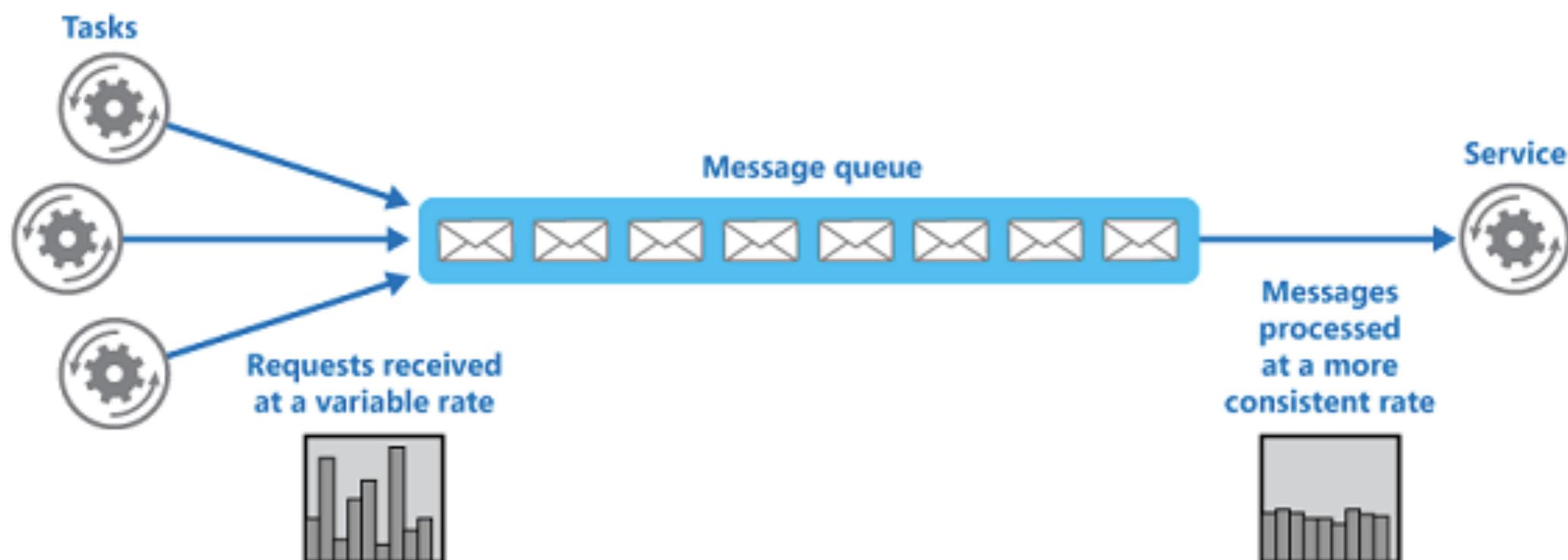


Request

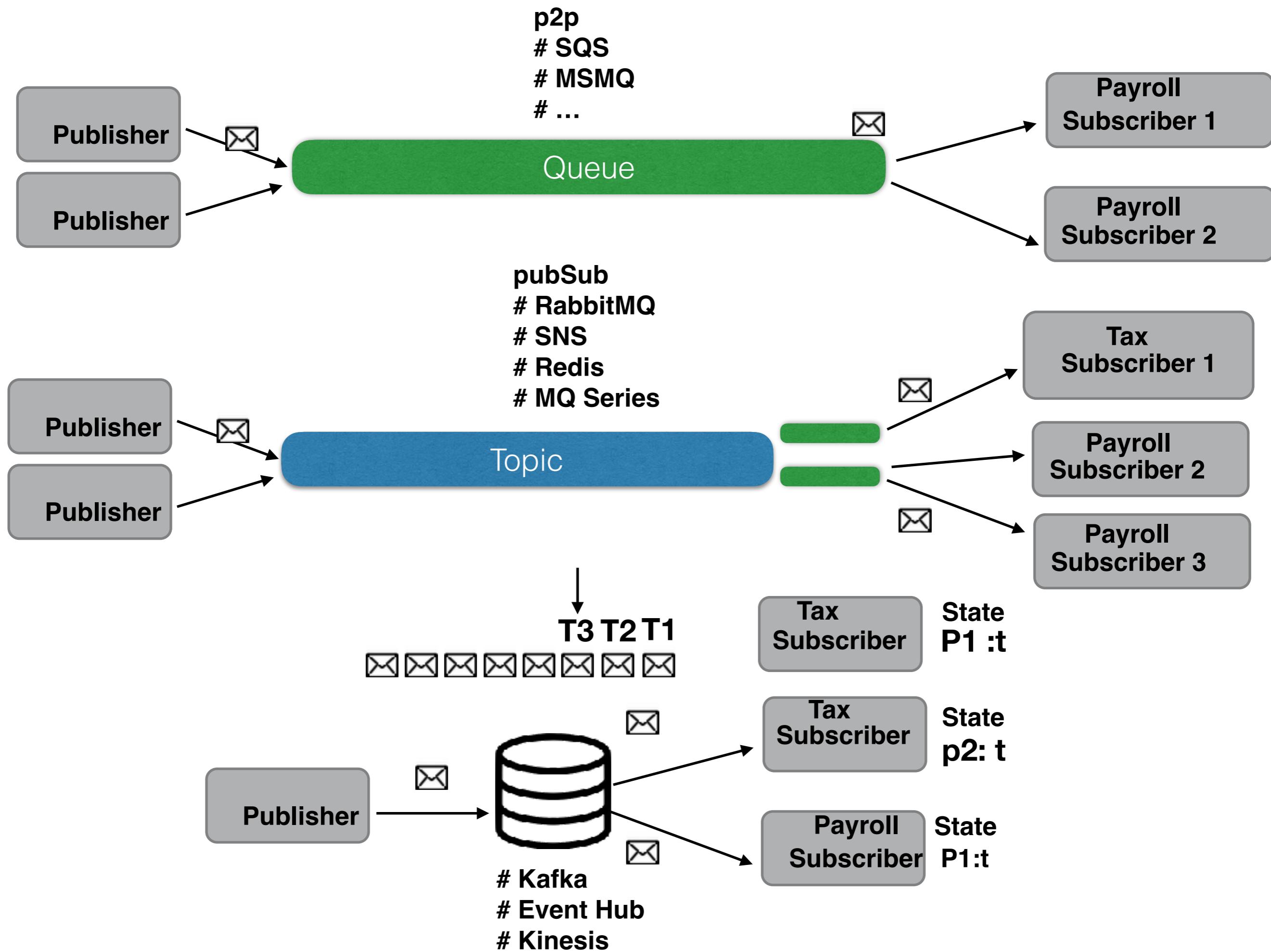
Time



Queue-based Load Levelling



source:msdn

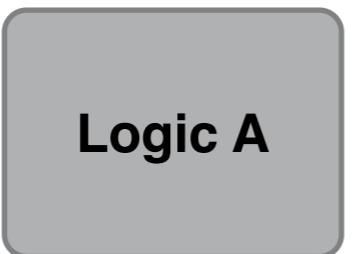
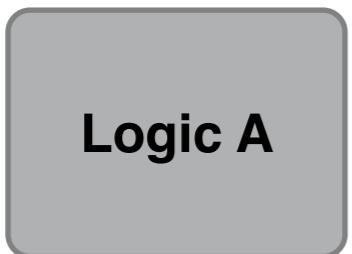
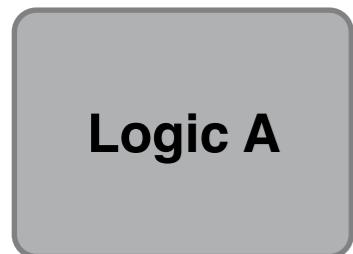


D1

D2

D3

Data Concurrent Processing

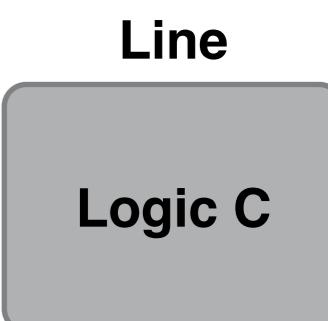
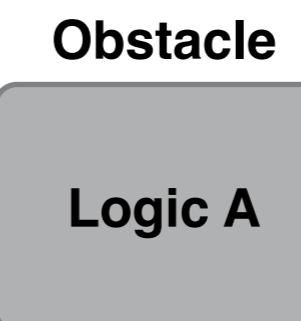


D1

D2

D3

Task Concurrent Processing



D1

D1

D1

D2

D2

D2

D3

D3

D3

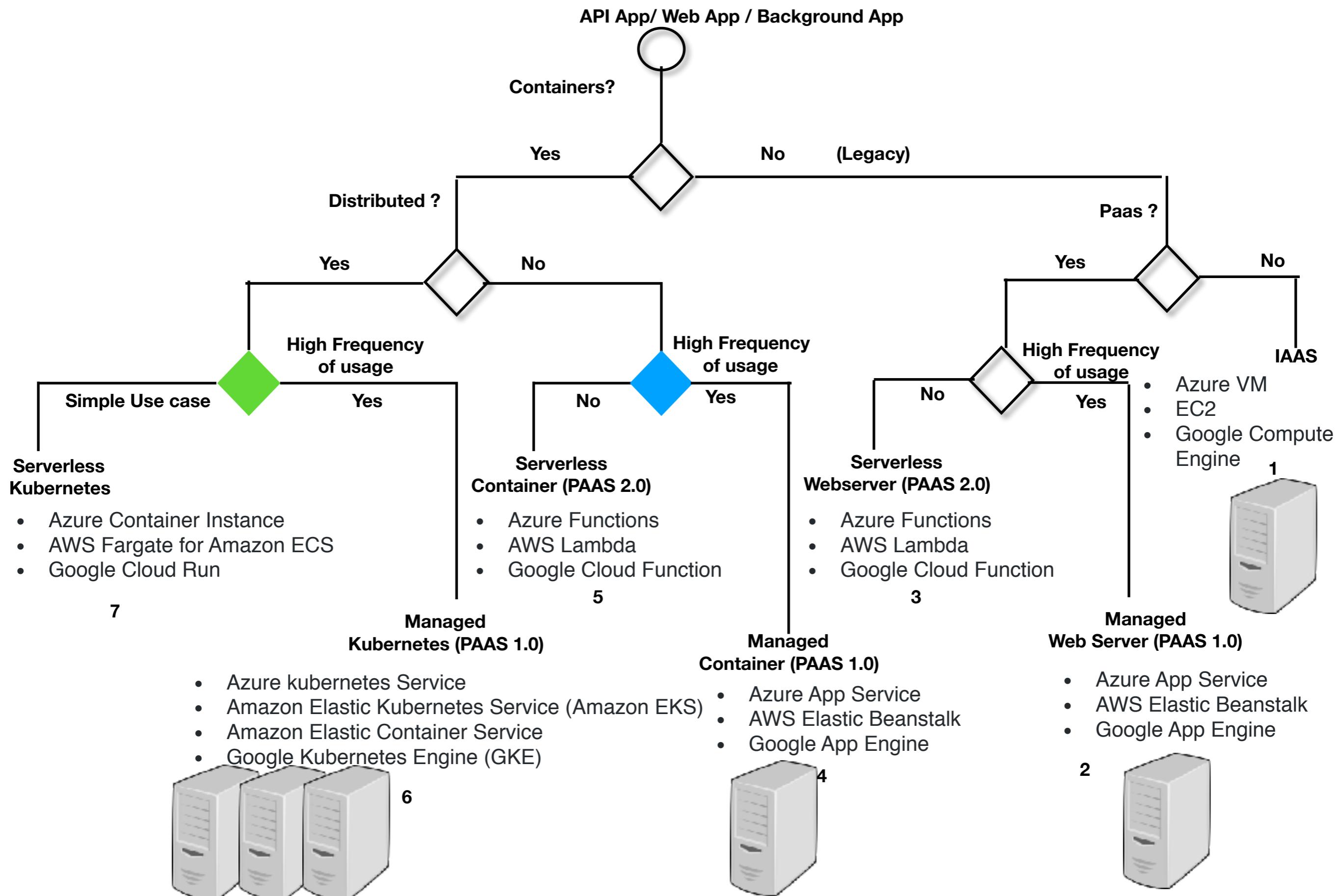
	Kafka	Pulsar	RabbitMQ (Mirrored)
Peak Throughput (MB/s)	605 MB/s	305 MB/s	38 MB/s
p99 Latency (ms)	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
Number of Programming Languages Supported	17	6	22
Stack Overflow Questions	21,233	134	11,430
Pub/sub	Yes	Yes	Yes
Message routing	Medium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

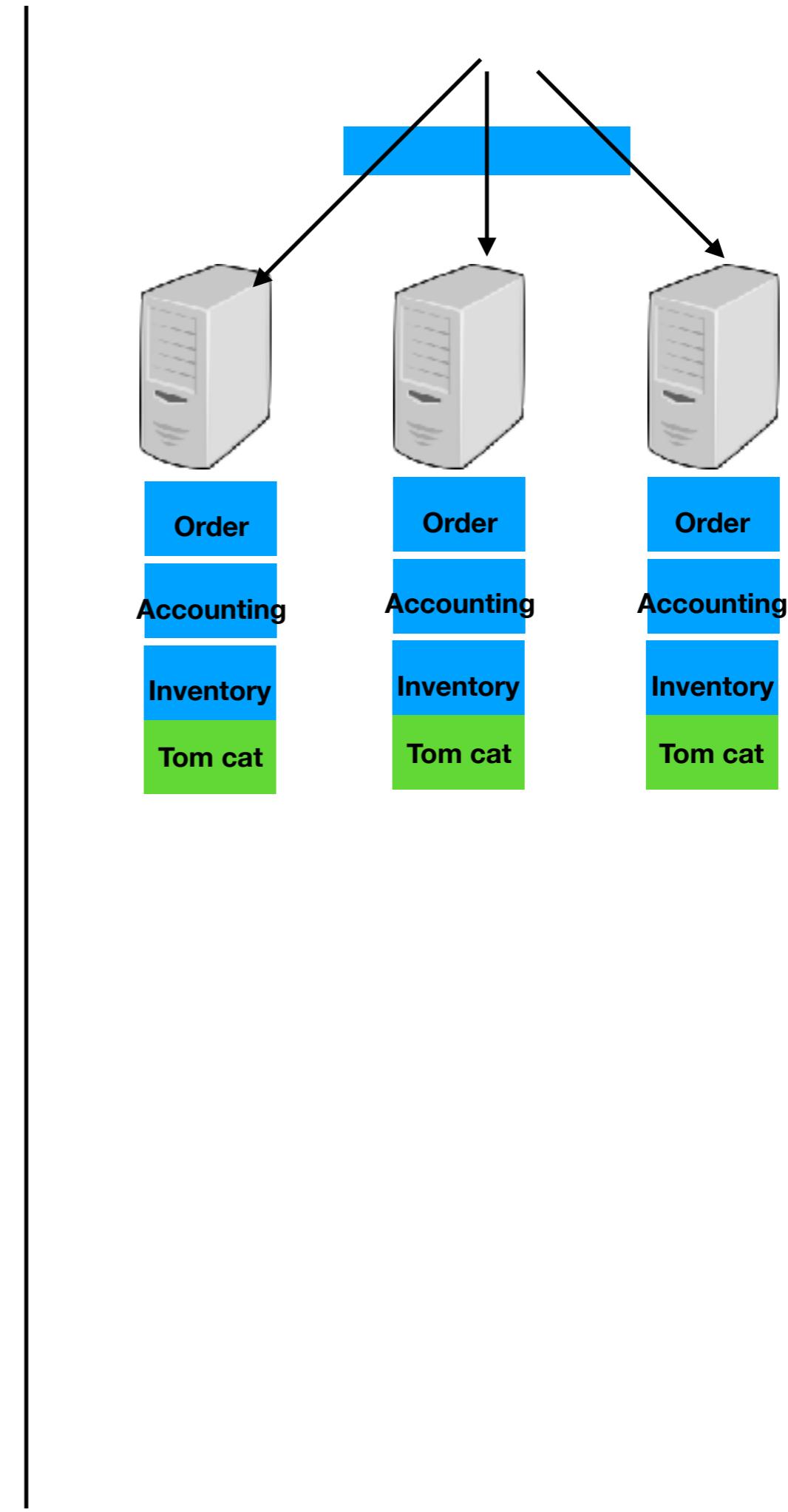
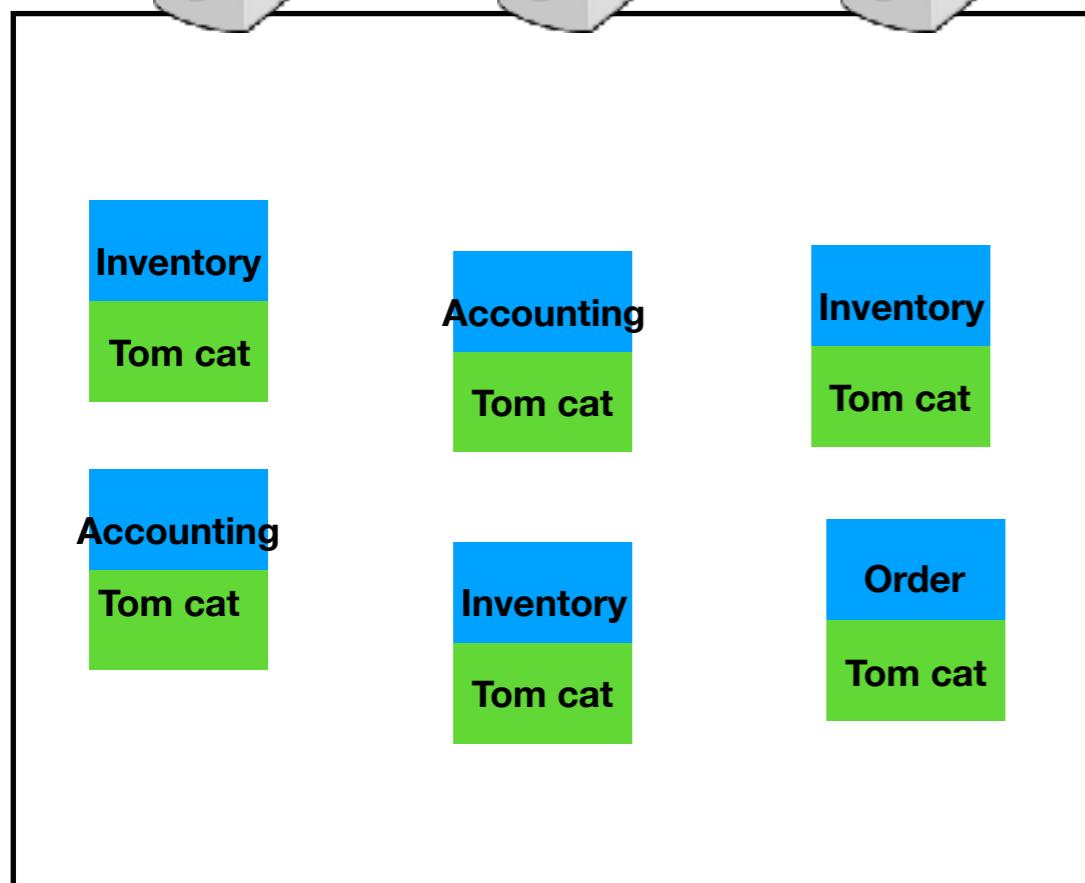
Table 1

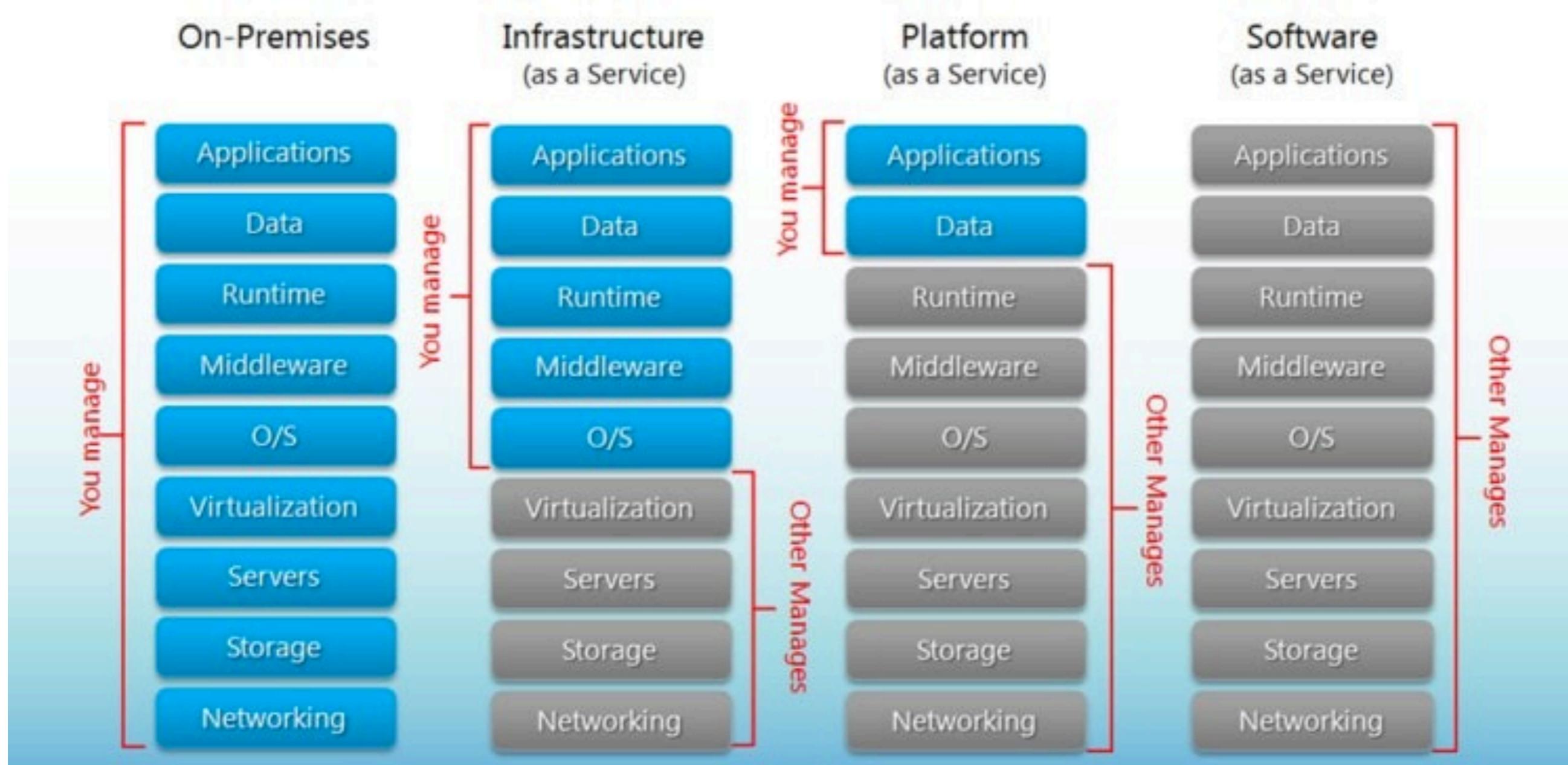
	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php,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

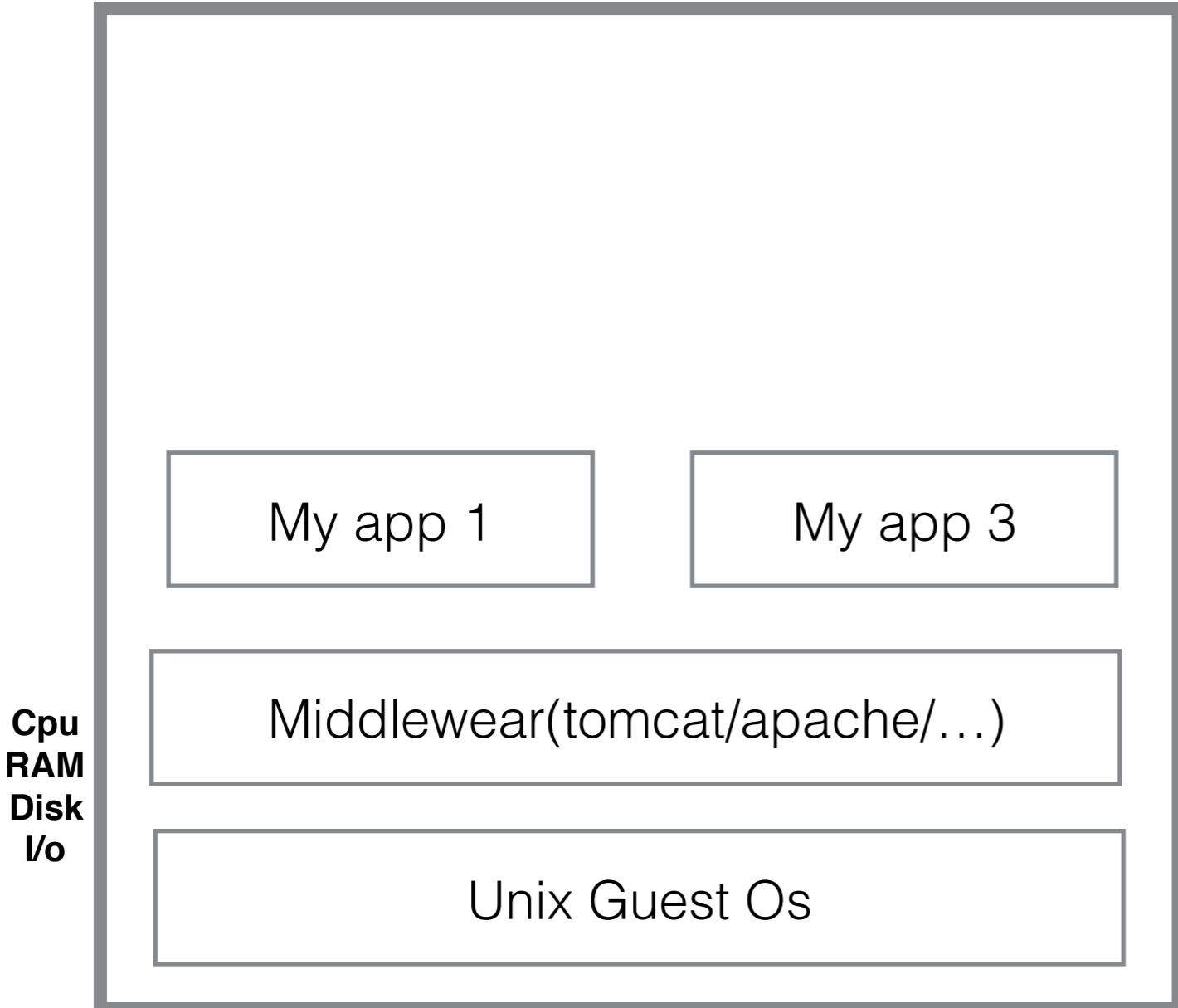
Middlewear (tomcat/ Apache/ IIS / Nodejs...)

Os

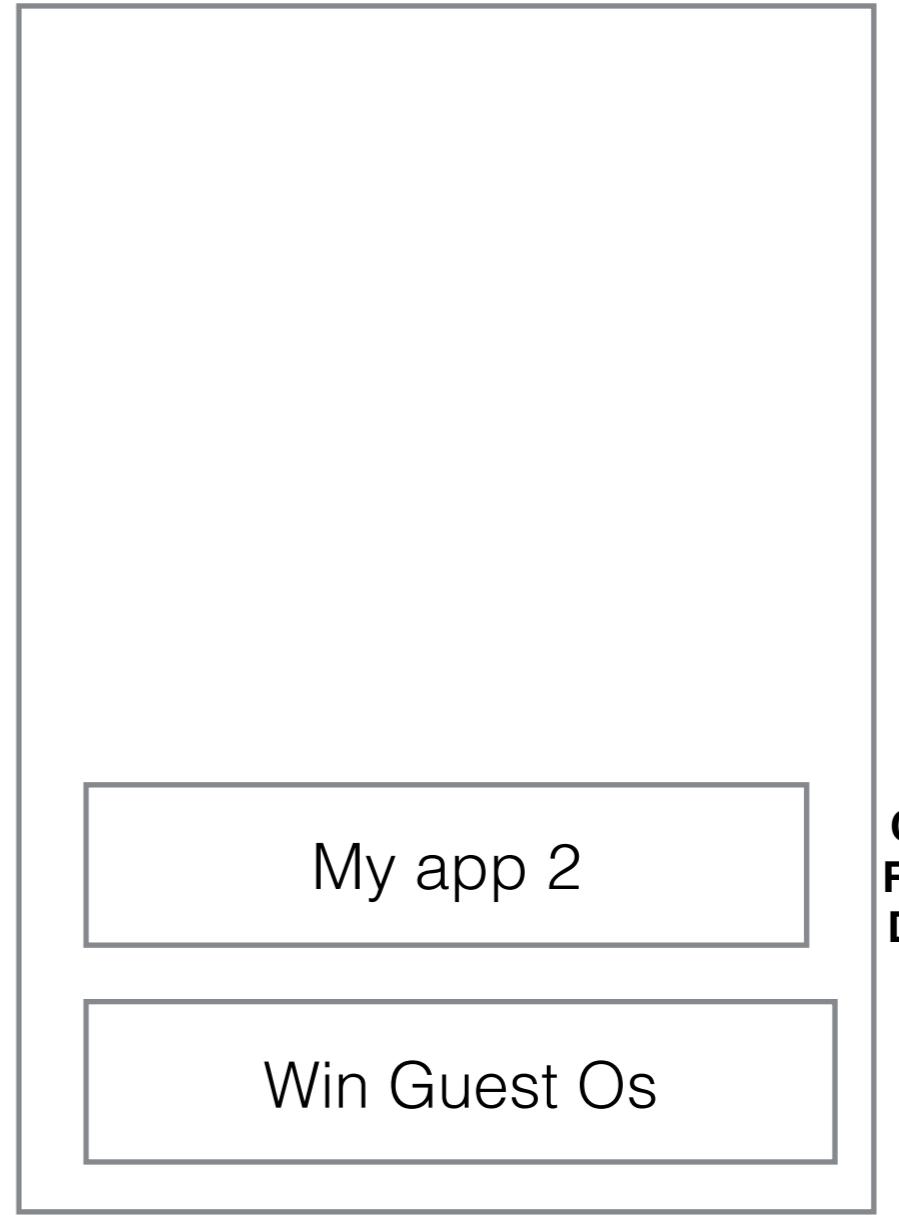
Cpu
RAM
Disk
I/o

Machine

VM1



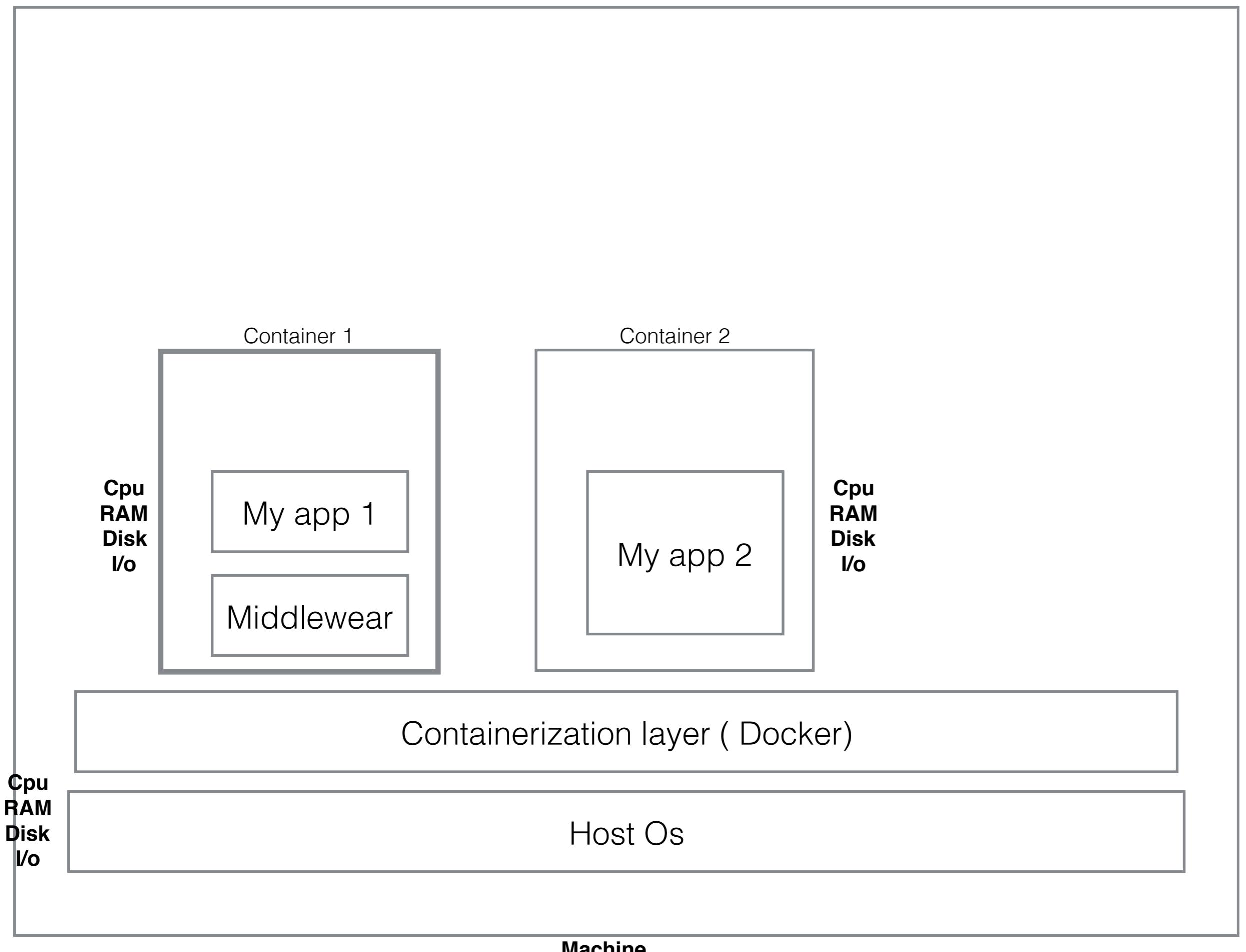
VM2



Machine

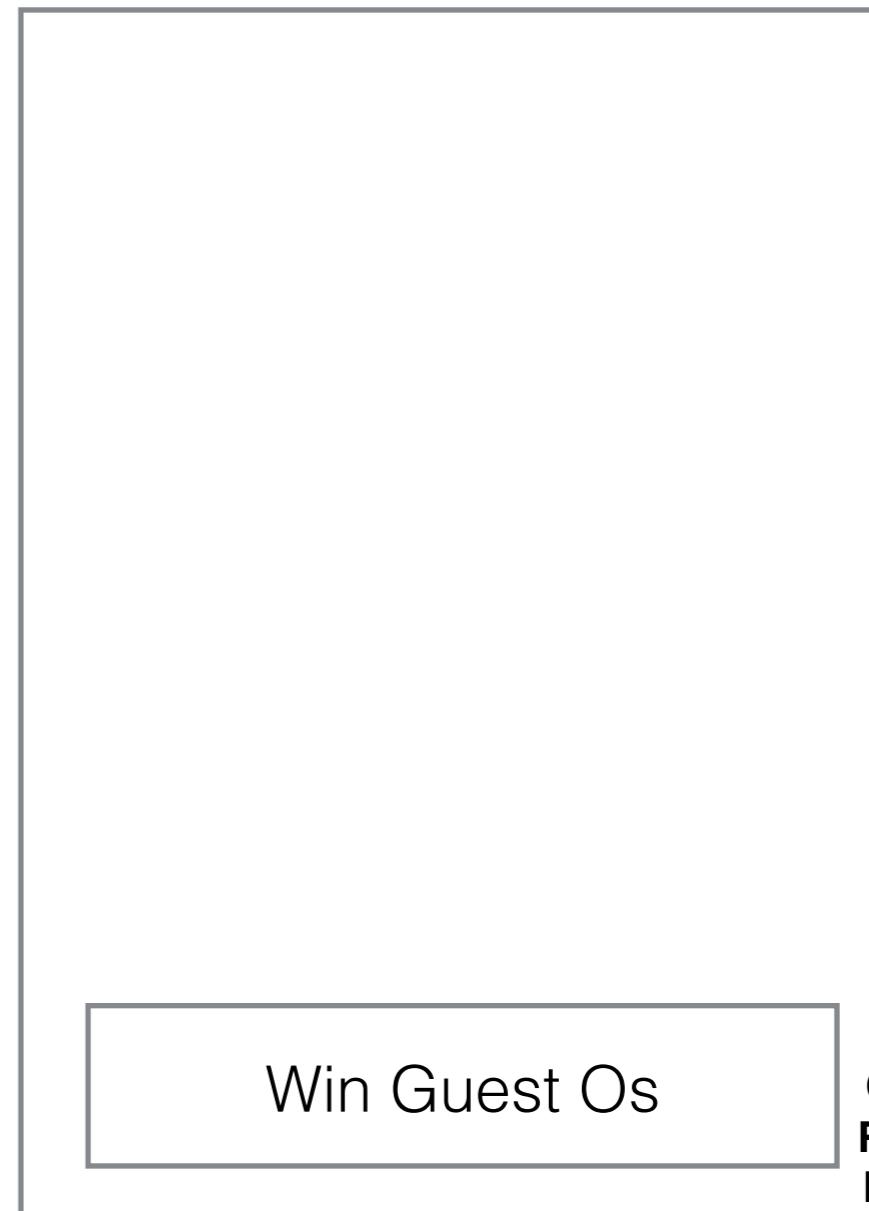
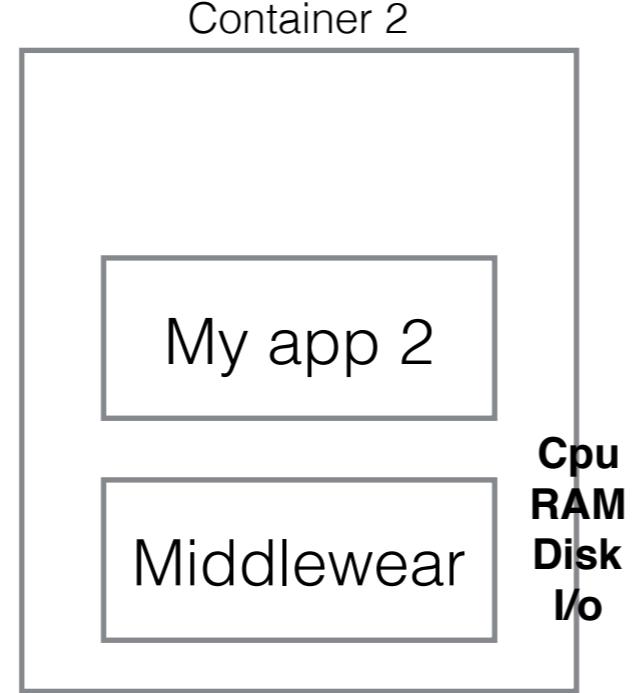
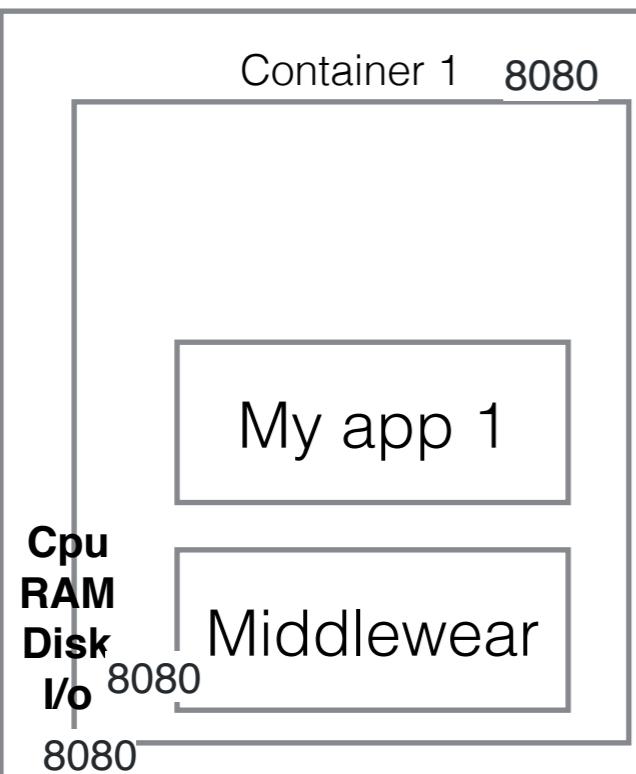
**Cpu
RAM
Disk
I/o**

**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)

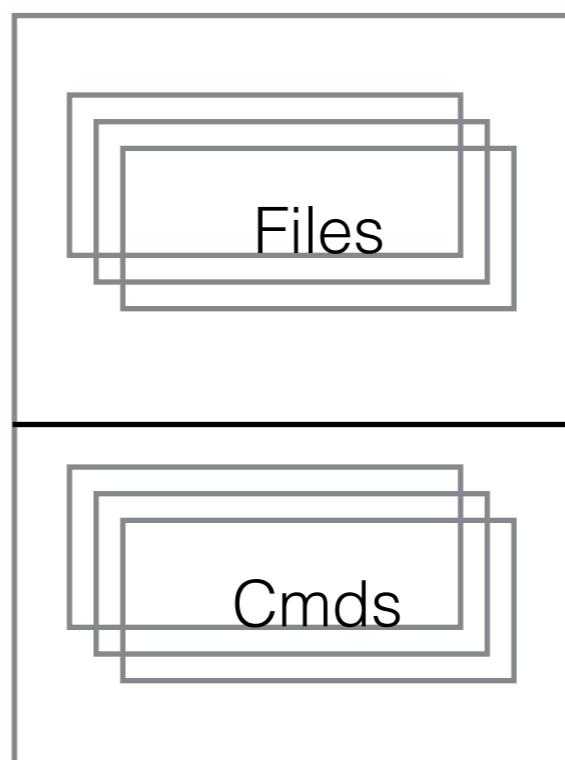
Cpu
RAM
Disk
I/o

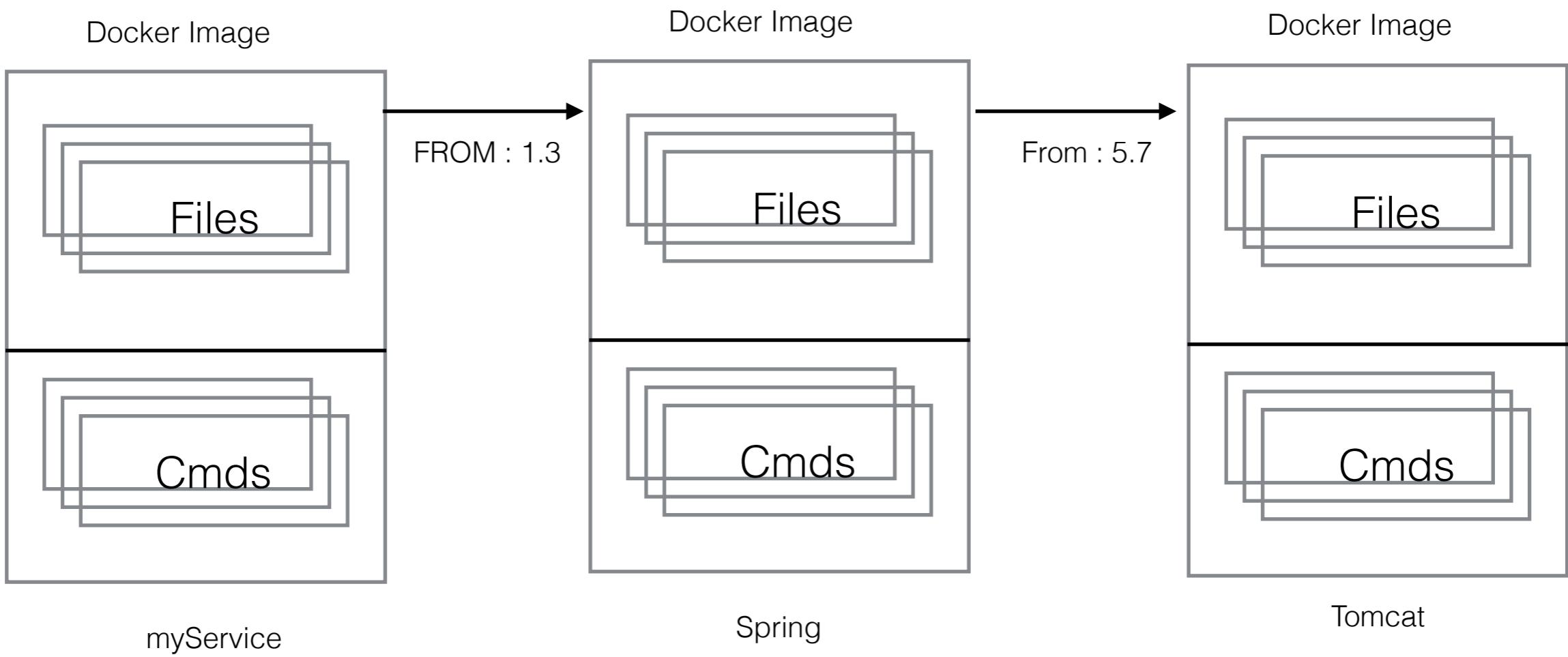
Host Os (azure)

Machine

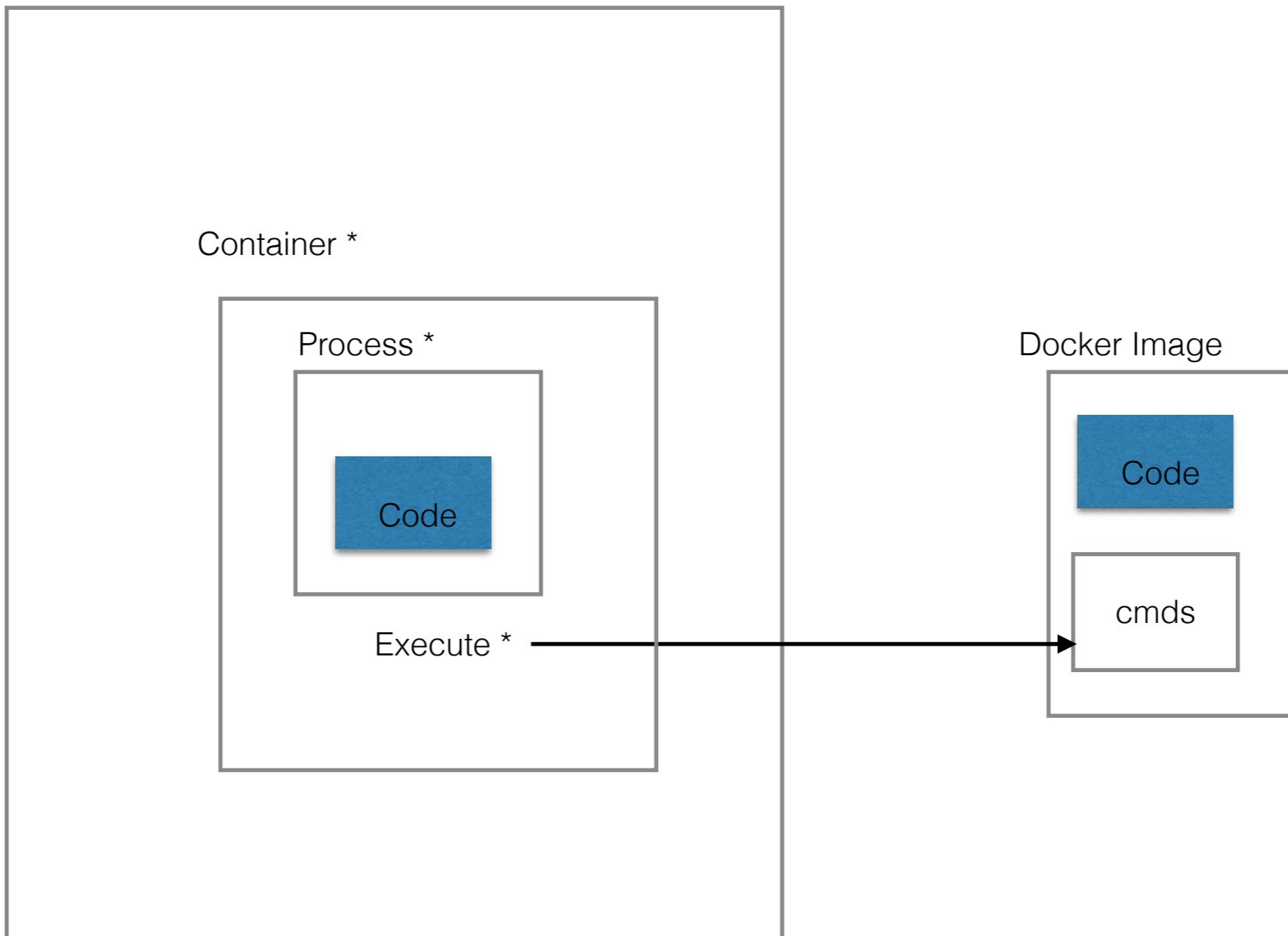
Reproducible Env.

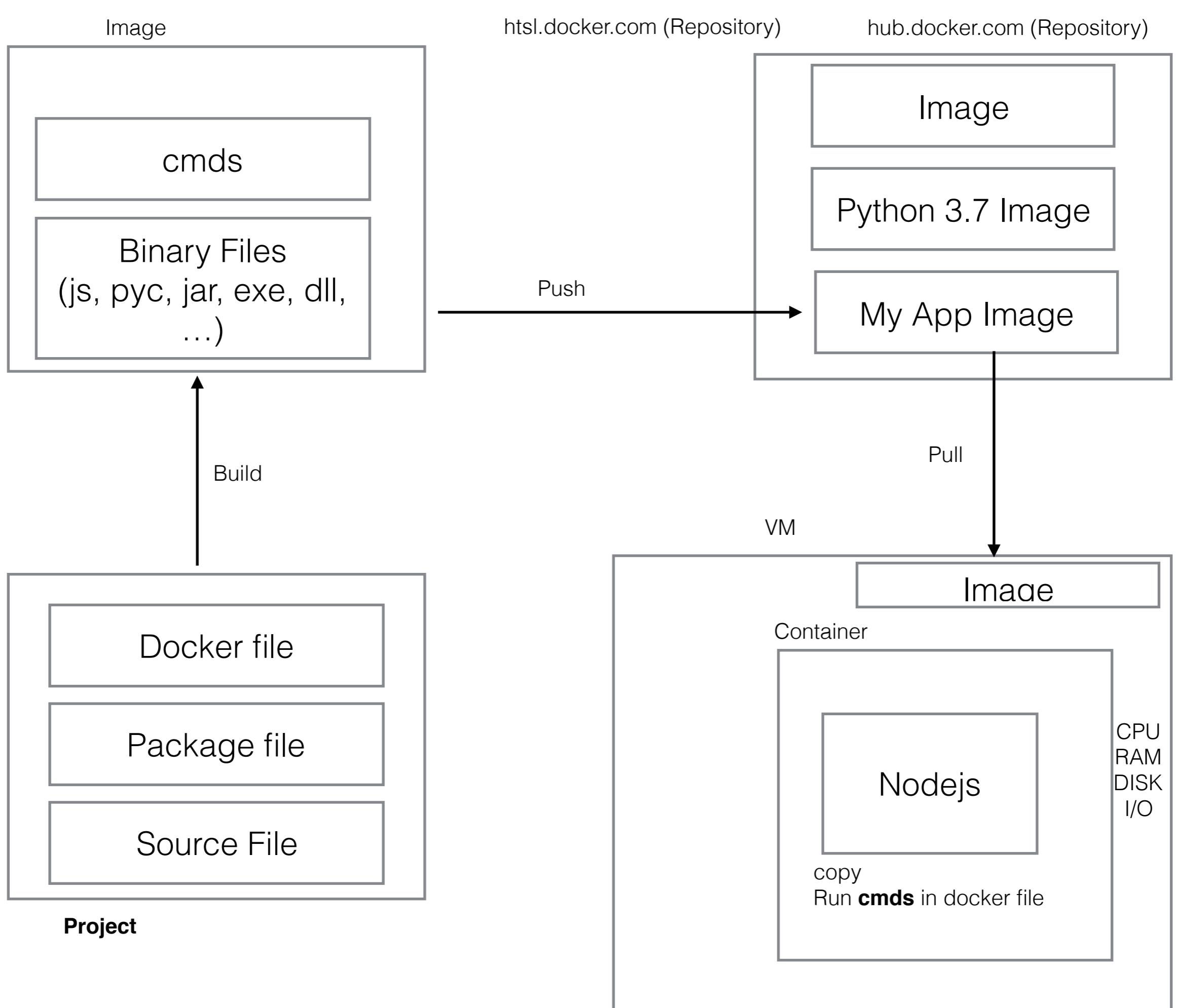
setup/msi/war/..





Virtual Machine/ Physical Machine

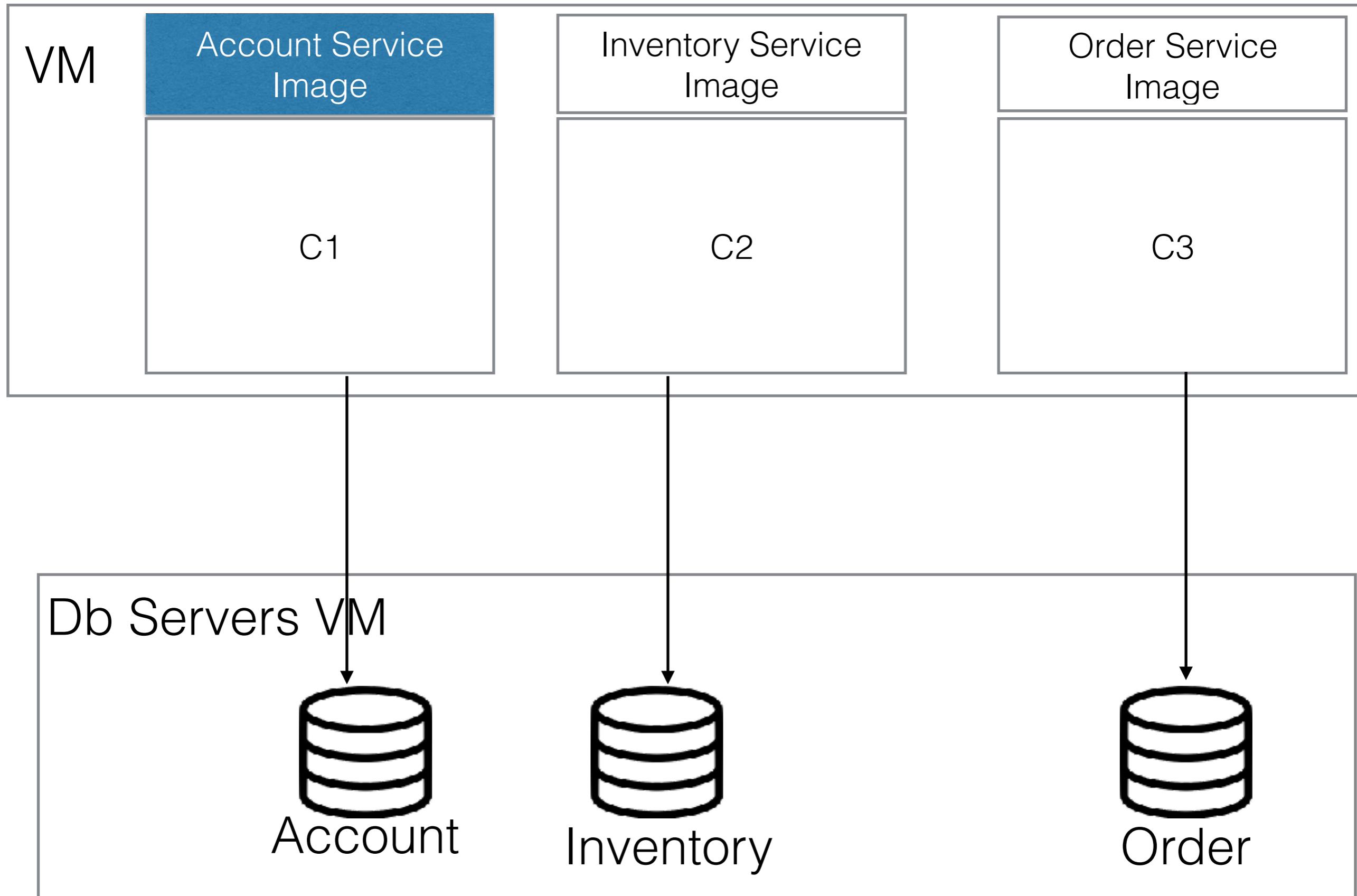




Account Service
Image

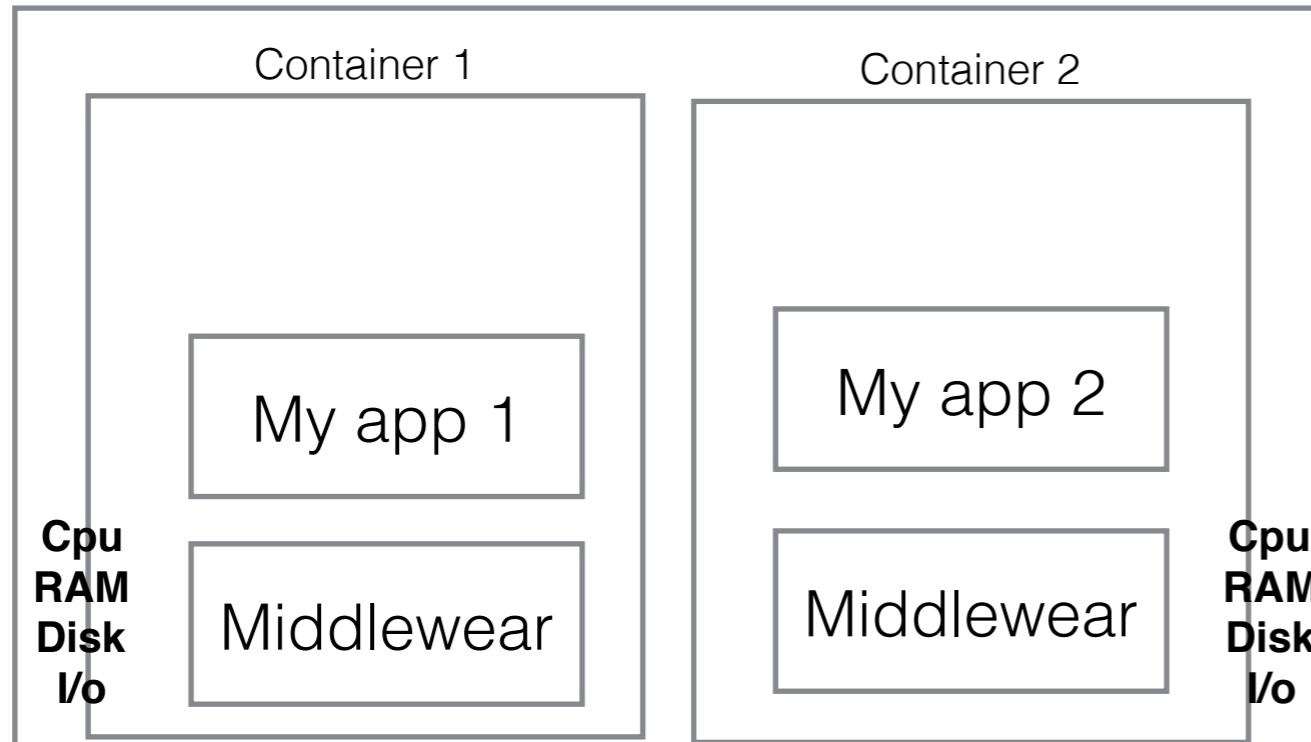
Inventory Service
Image

Order Service
Image

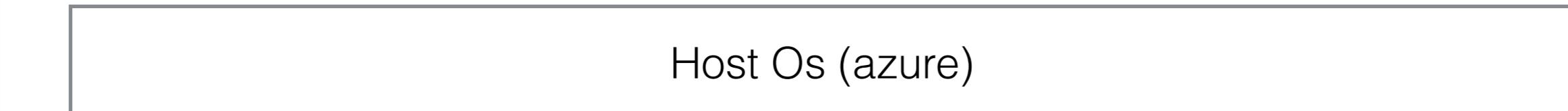
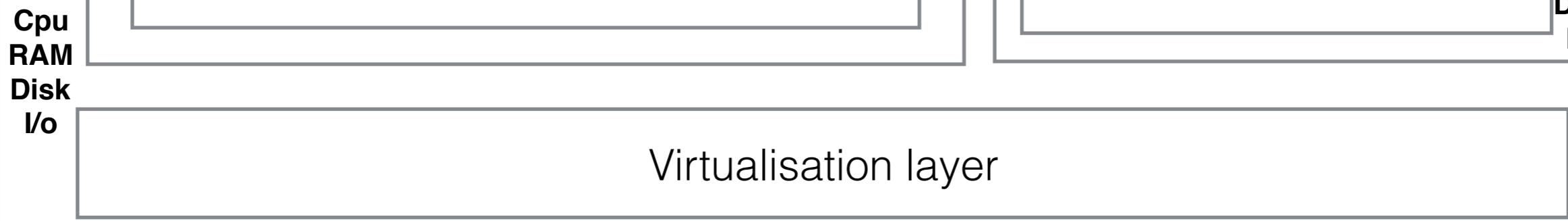
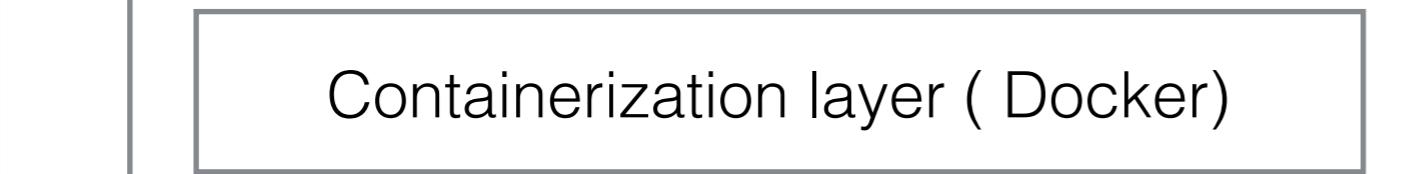
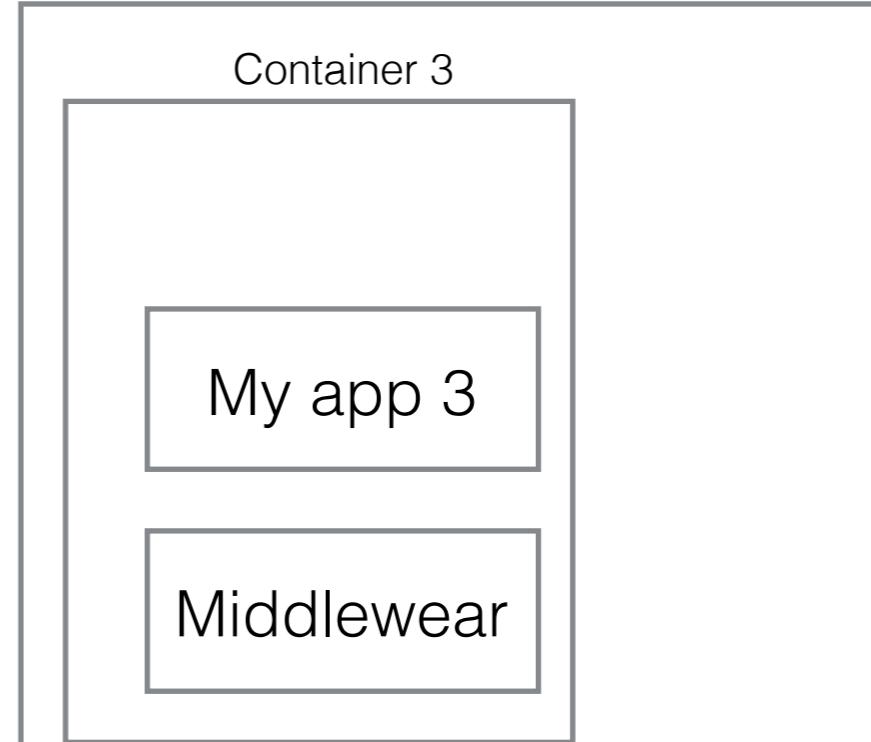


Managing Container Life cycle

VM1



VM2



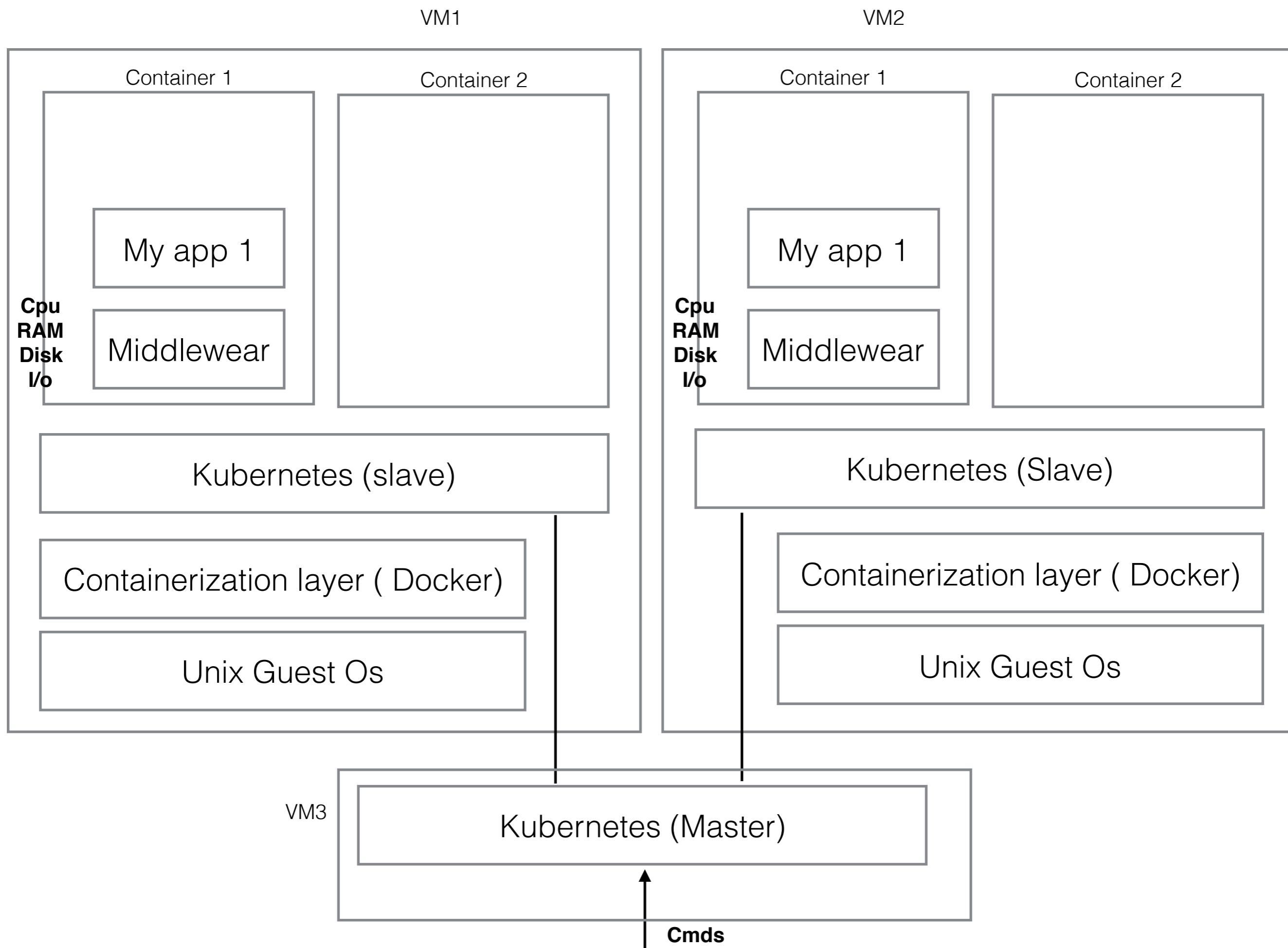
Machine

Kubernetes (slave)

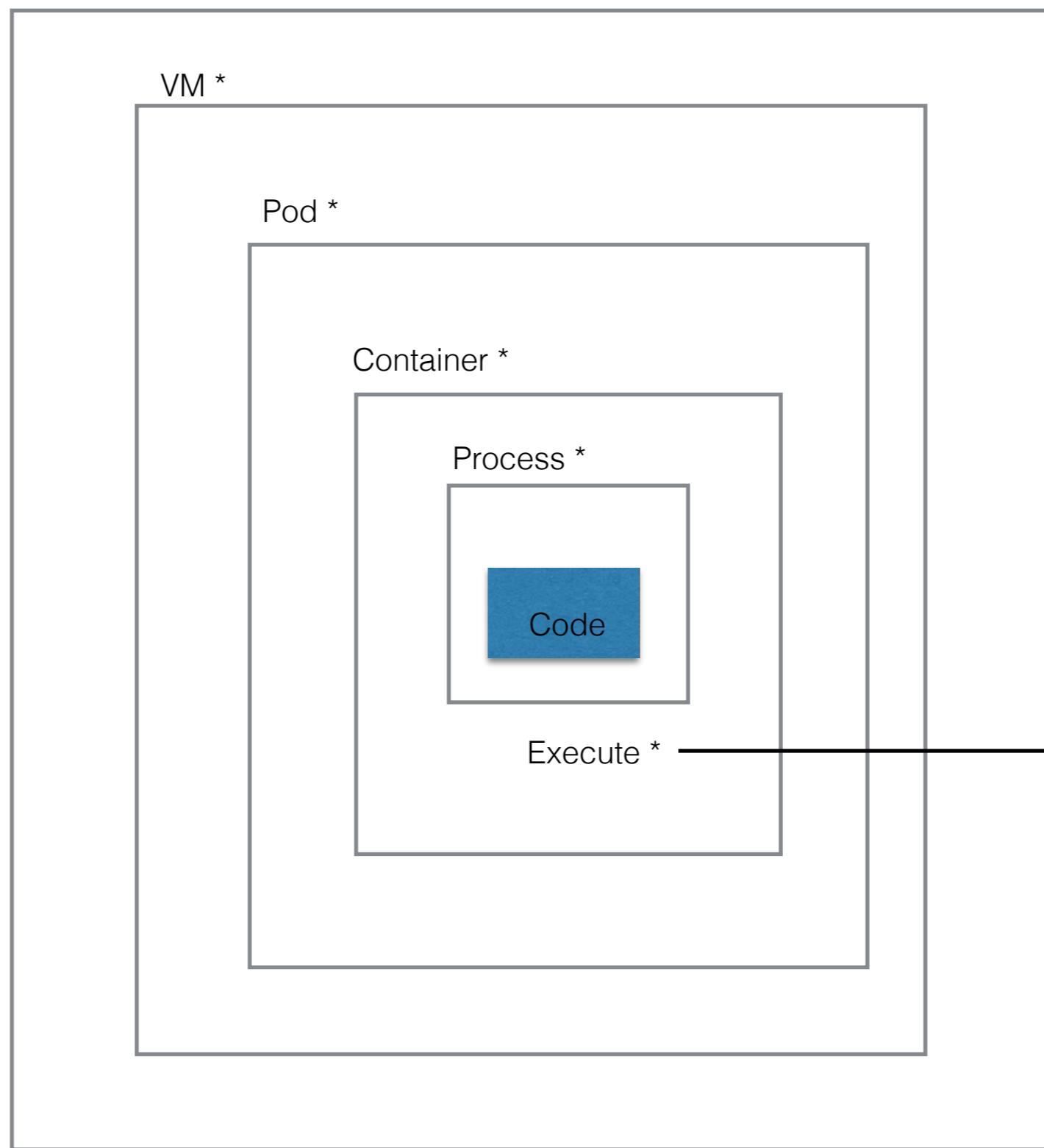
Kubernetes (Slave)

Kubernetes (Master)

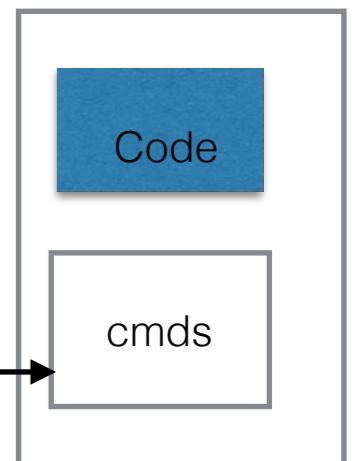




Cluster

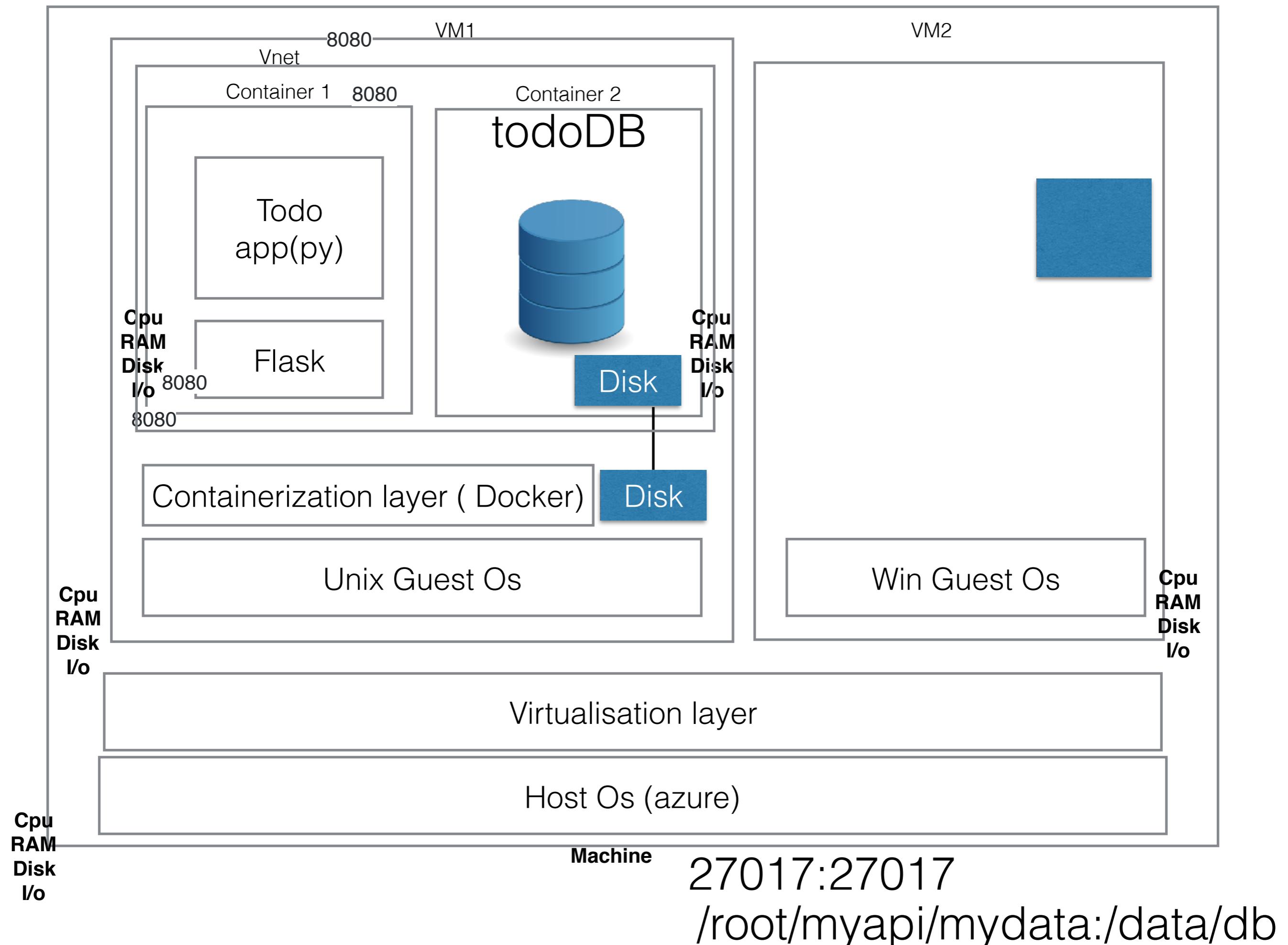


Docker Image

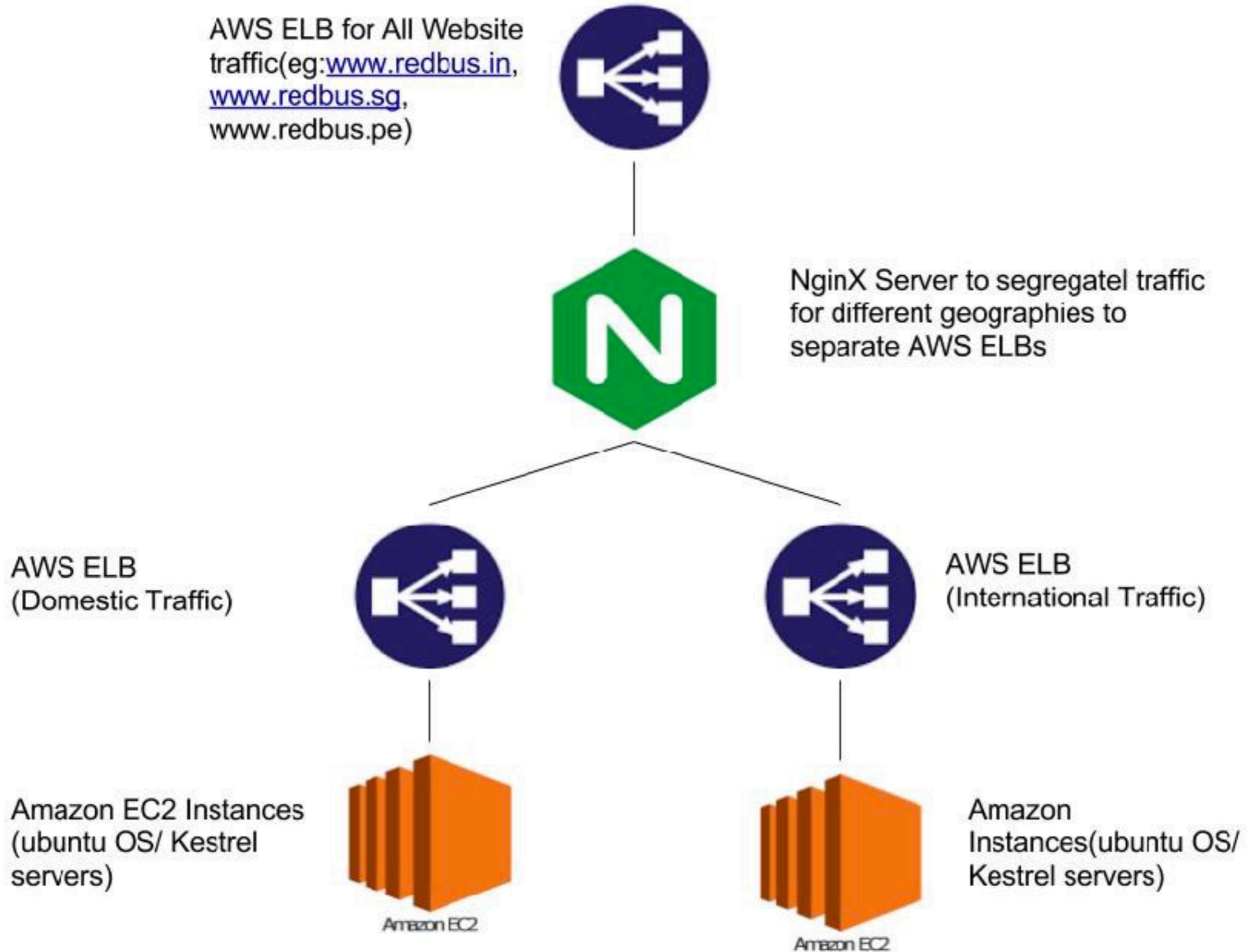


Docker /k8s

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

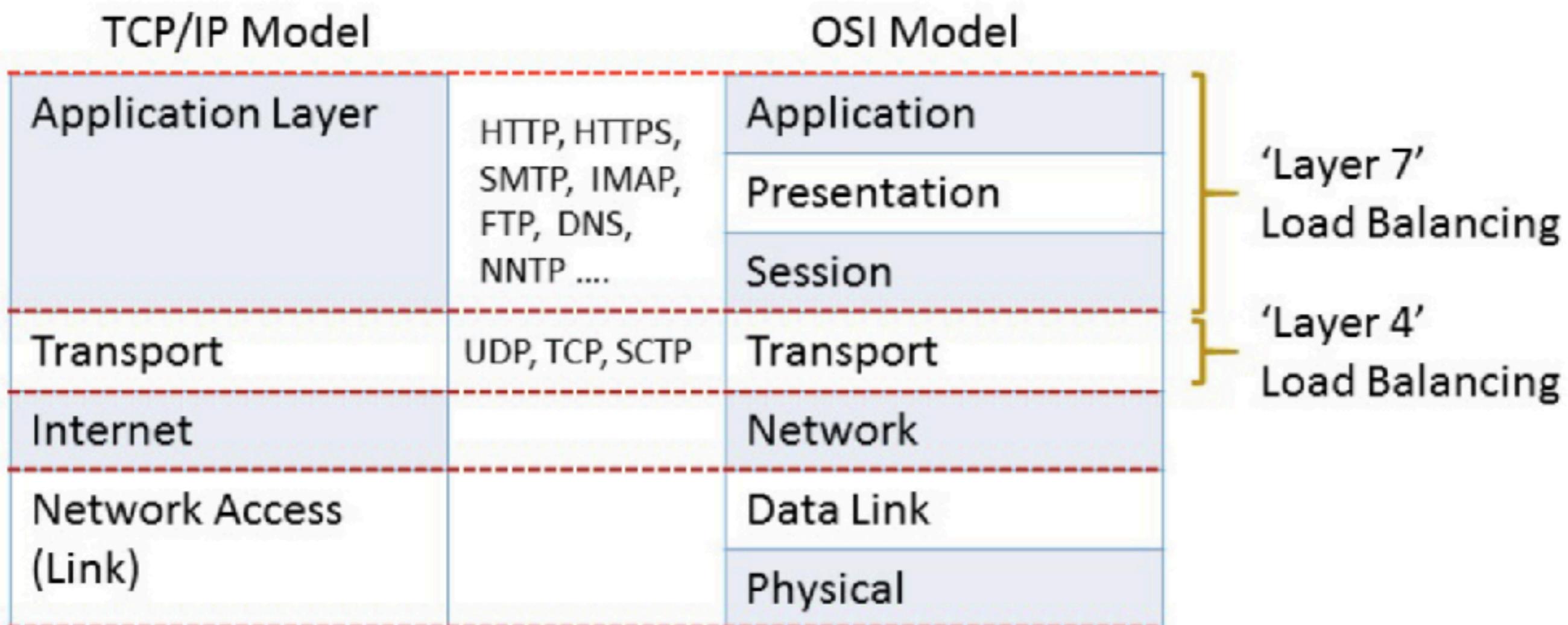


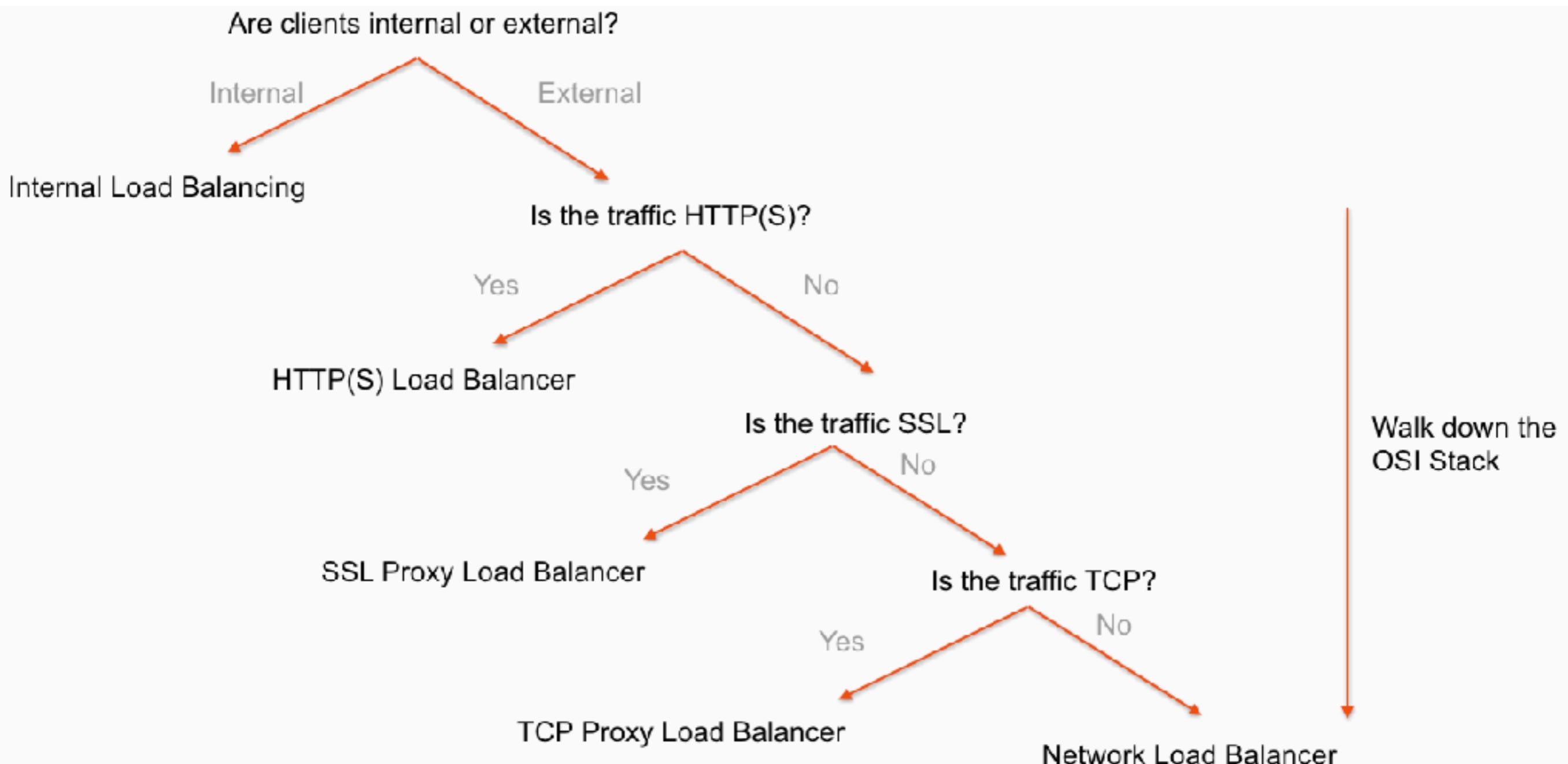
INFRASTRUCTURE SETUP

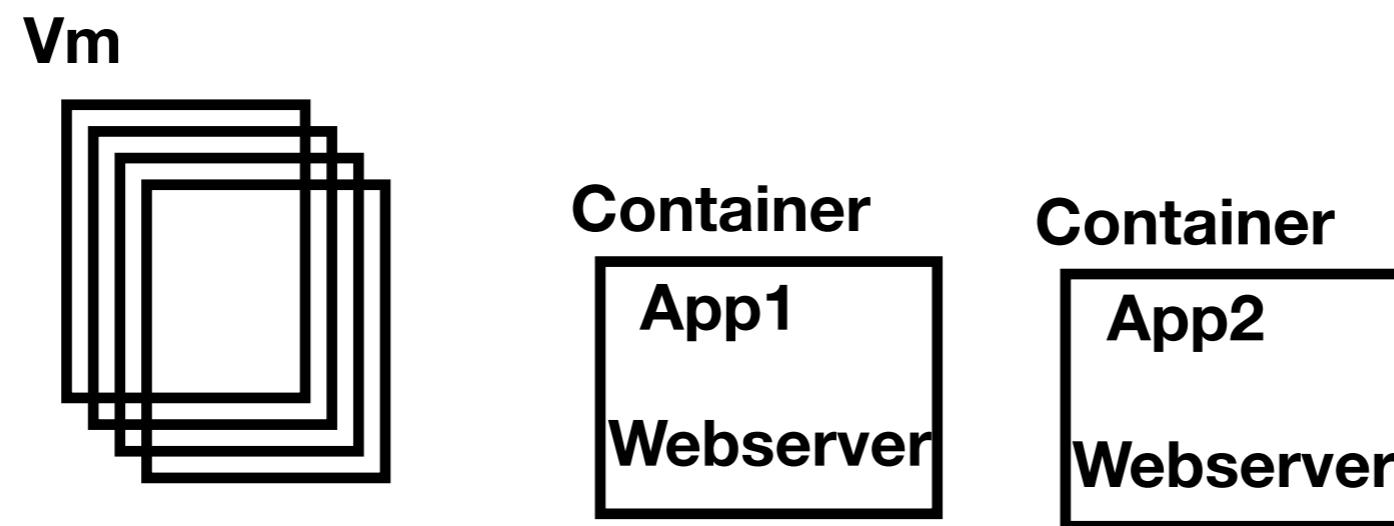
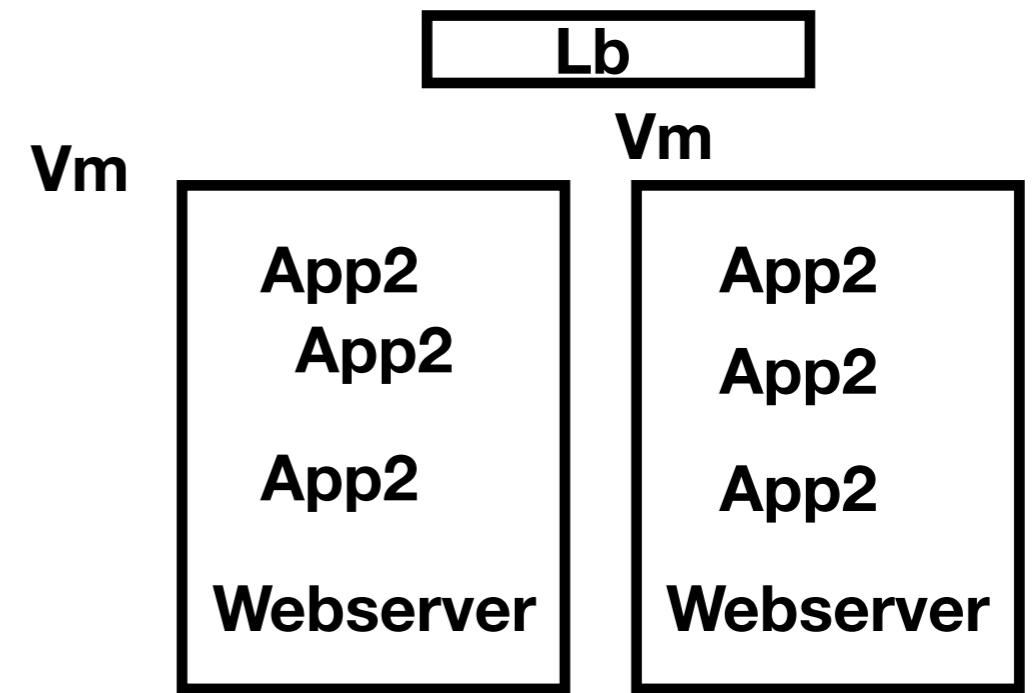
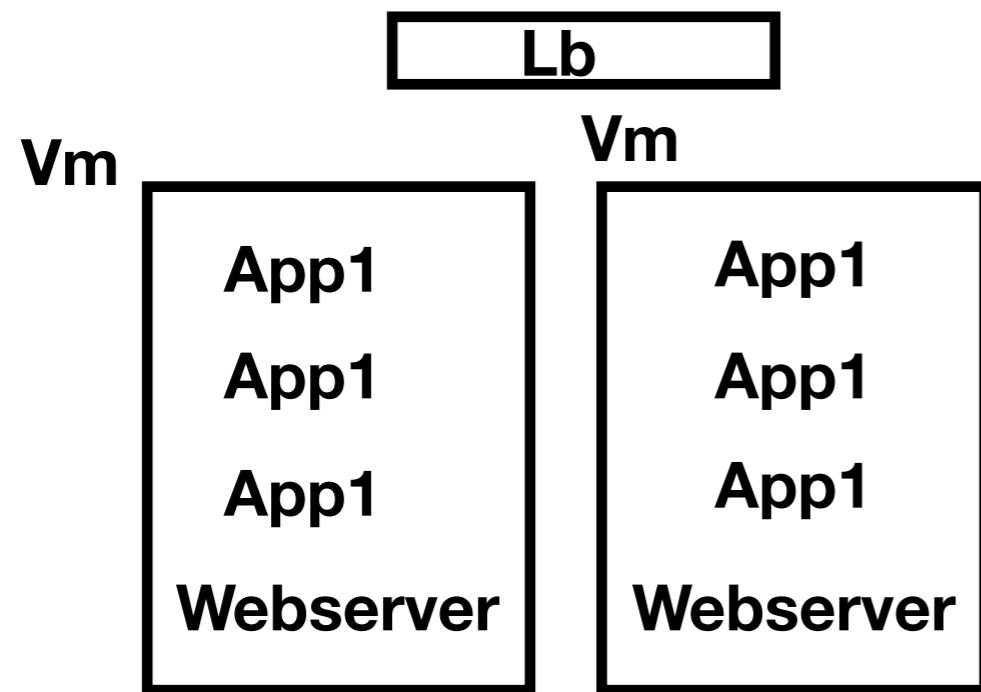


.NET framework
MVC 4
IIS
windows
[\$0.192/hr for Windows OS]
<http://XXXX.redbus.pe/>, Throughput
was 60.5/Sec.

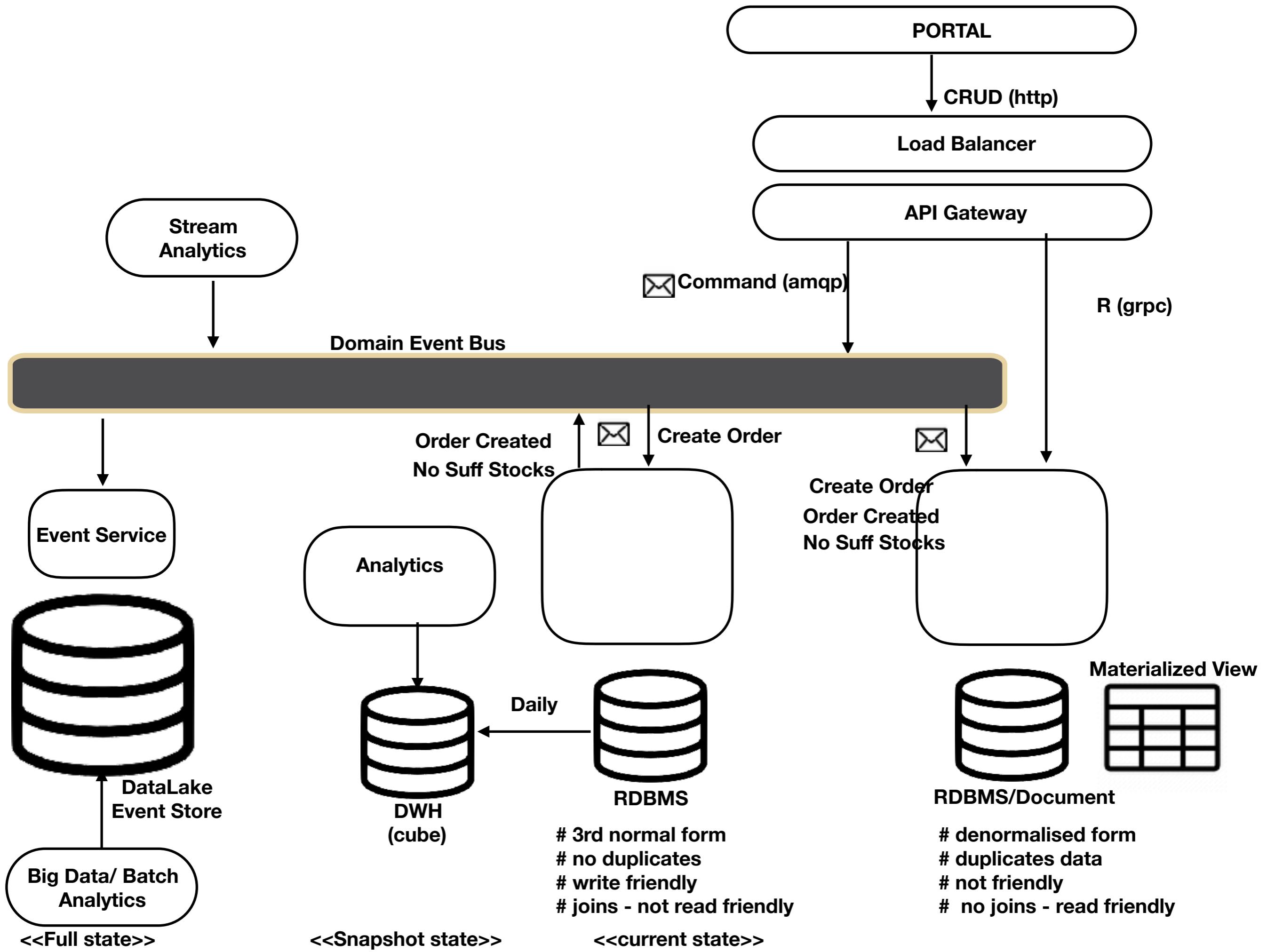
2–6 EC2s of C4 family
depending on the traffic.
.NET CORE
[\$0.1/hr for Linux] 45% of cost
<http://XXXX.redbus.pe/> ,
Throughput was 142.9/Sec.







Case Study



CQRS

Event vs Command

Event Sourcing

Materialized View

SAGA - Compensatable transaction

Eventual Consistency

Portal



CRUD - Rest

API Gateway / Reverse Proxy / Ingress Service / ...

Grpc

Stream Analytics

Account Command Service

Do Undo

Order Created

Order API Service

Create Order

Grpc R

Event Service



Big Data Analytics

Reporting

DWH Analytics

Order Command Service

Do Undo

Order Created
NoStocks

Create Order

Order Created
Create Order
NoStocks

Order Query Service

Do Undo

Fail
-> domain
-> tech (server, net)

Rdbms
3rd normal form
current state



Rdbms/Document
denormalised form

Wide Column db/

Data Lake

Event Store

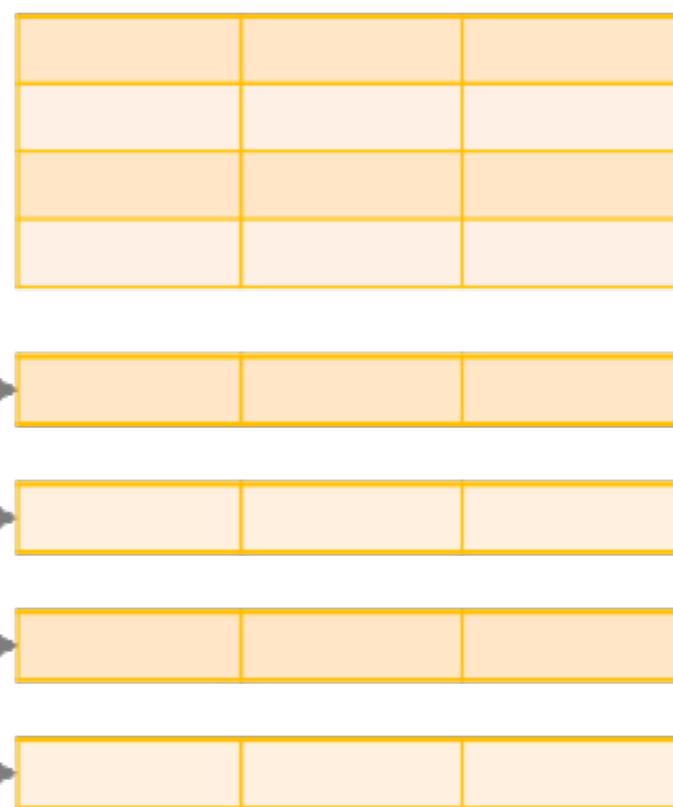
Full State

DWH Columnar db

Data stream



Unbounded Table

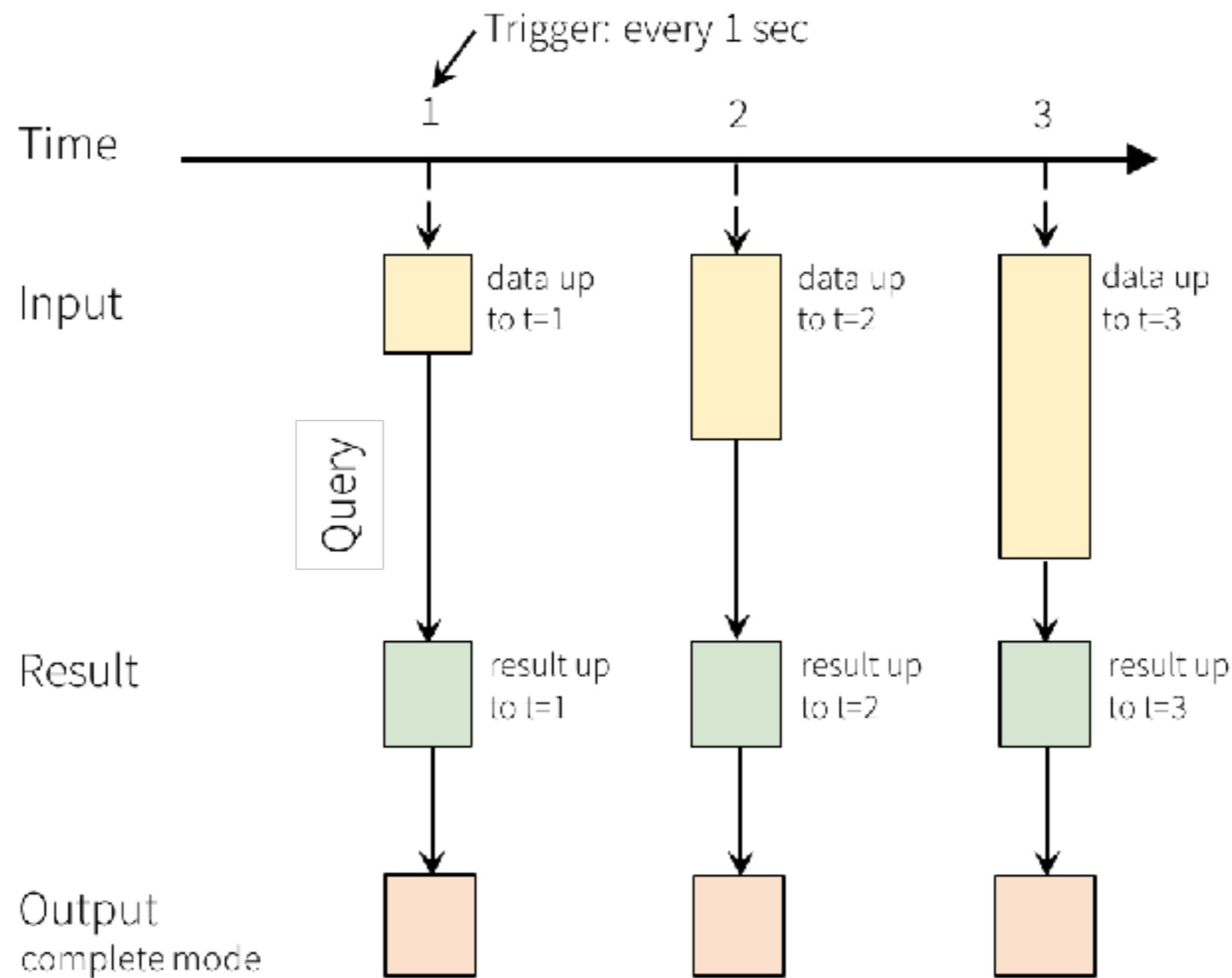


new data in the
data stream

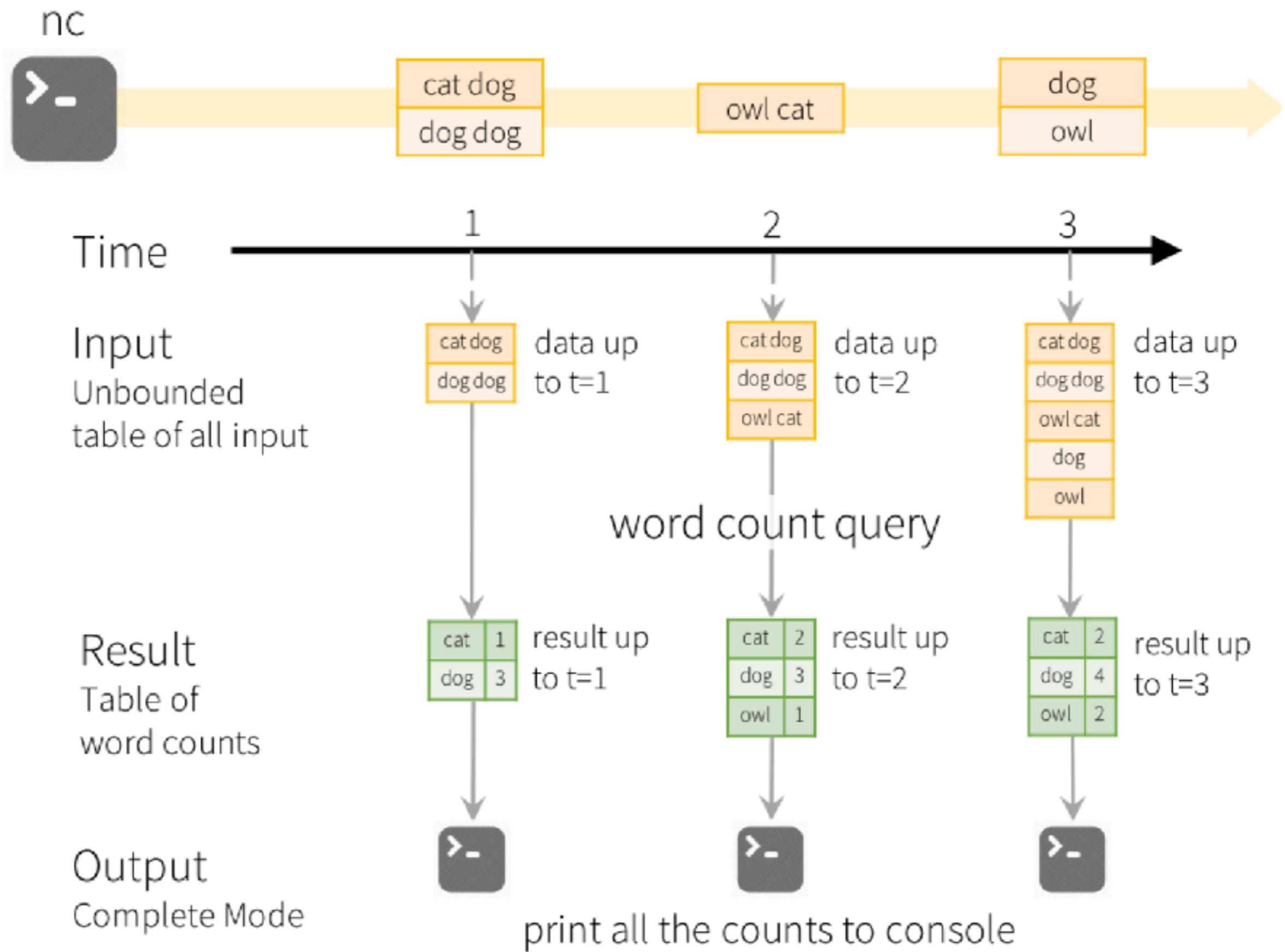
=

new rows appended
to a unbounded table

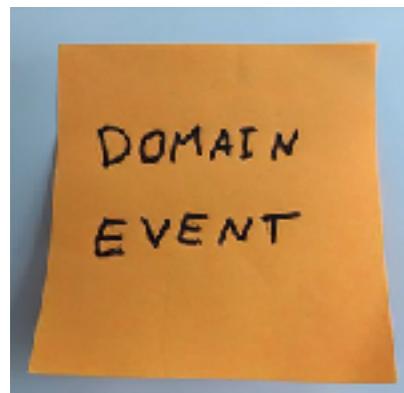
Data stream as an unbounded table



Programming Model for Structured Streaming



Event storming



Step 1: Create domain events



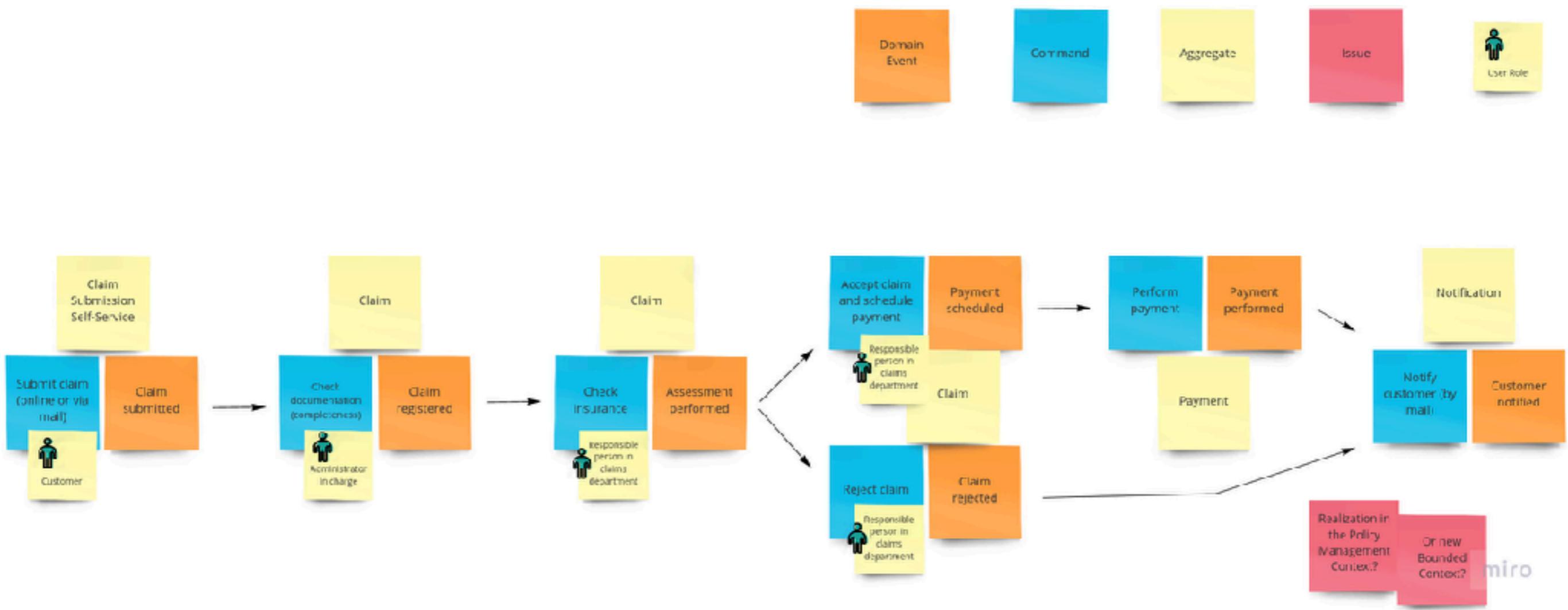
Step 2: Add the commands that caused the domain event



Step 2b: Add the actor that executes the command



Step 3: Add corresponding aggregate

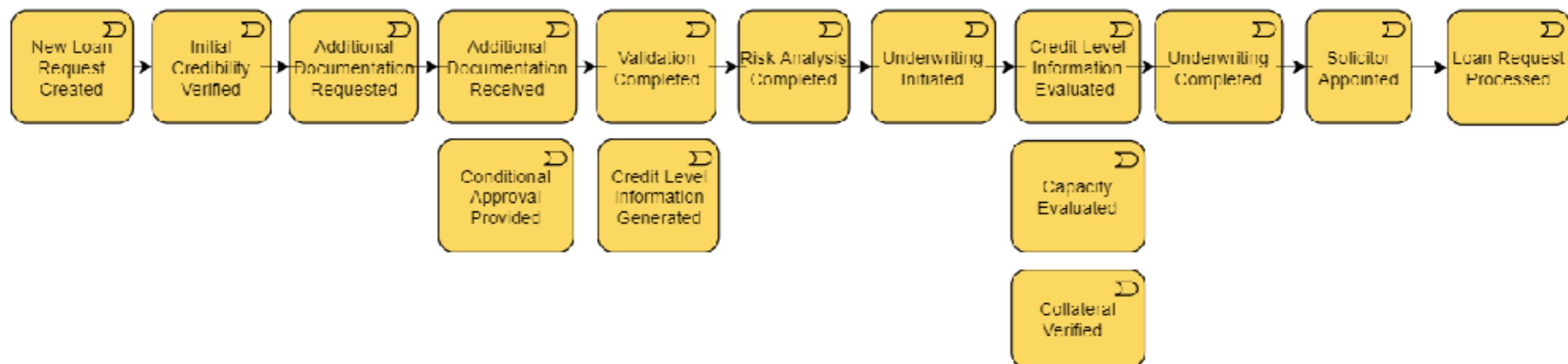


[https://github.com/getmubarak/Microservice/blob/master/
case%20study/Loan%20Approval%20Process.md](https://github.com/getmubarak/Microservice/blob/master/case%20study/Loan%20Approval%20Process.md)

Identification of Domain Events

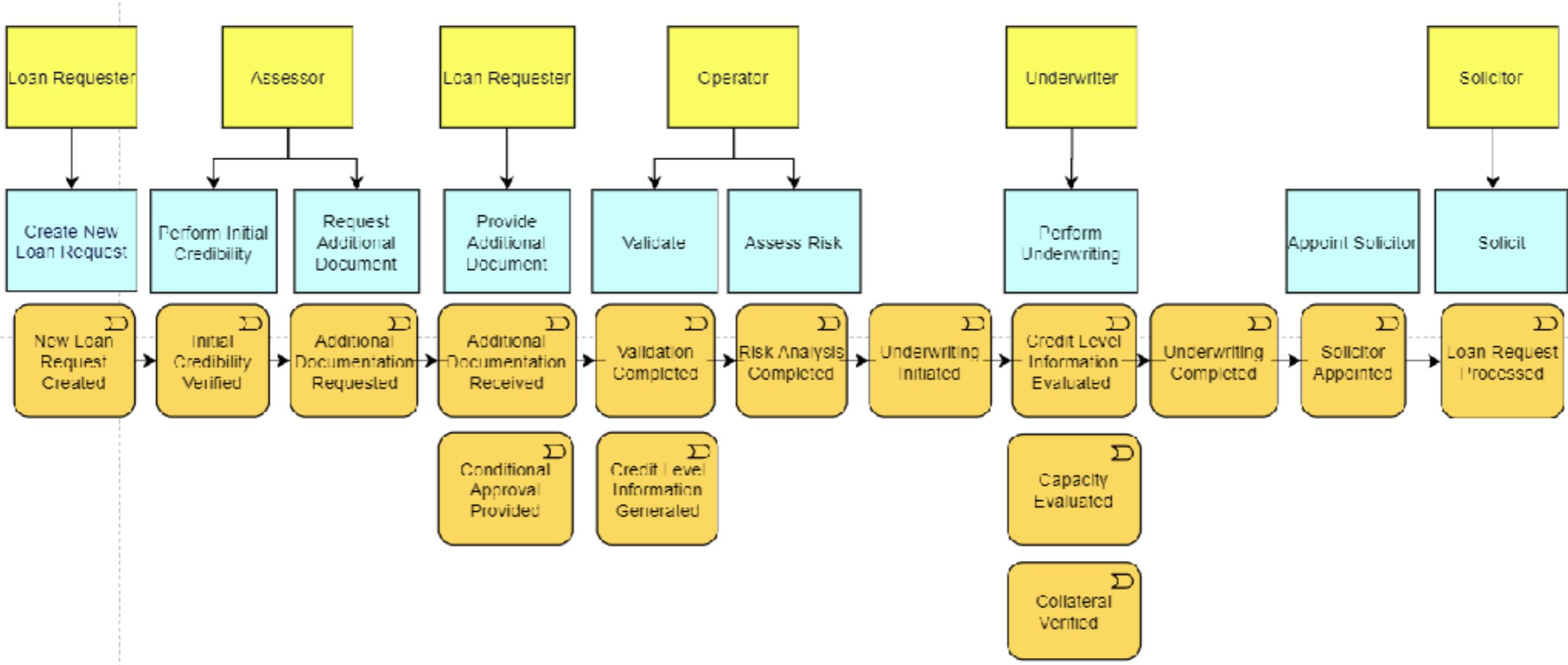


Time Sequencing of Domain Events



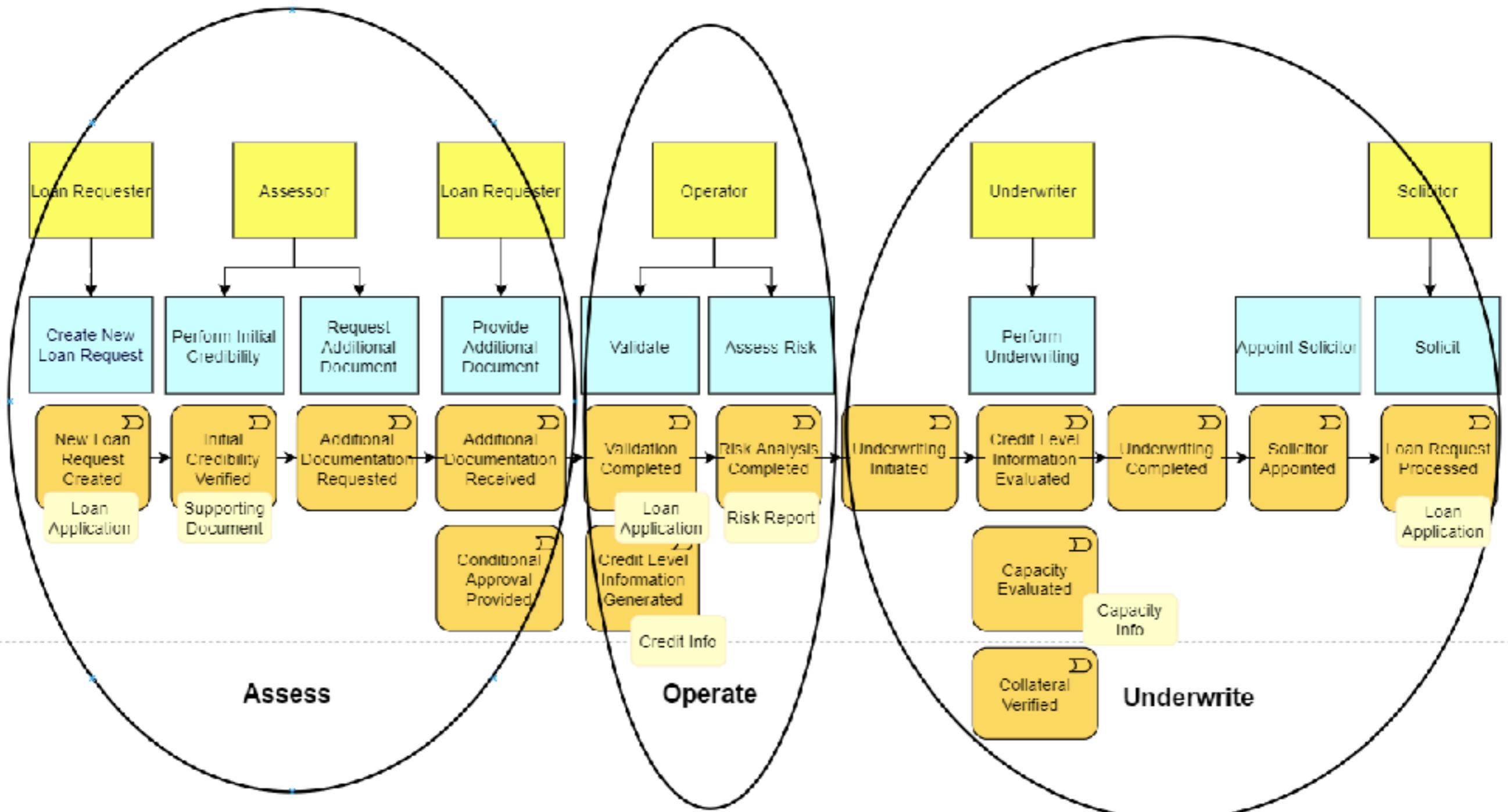
parallel have been stacked up vertically

Identification of Triggers/Commands

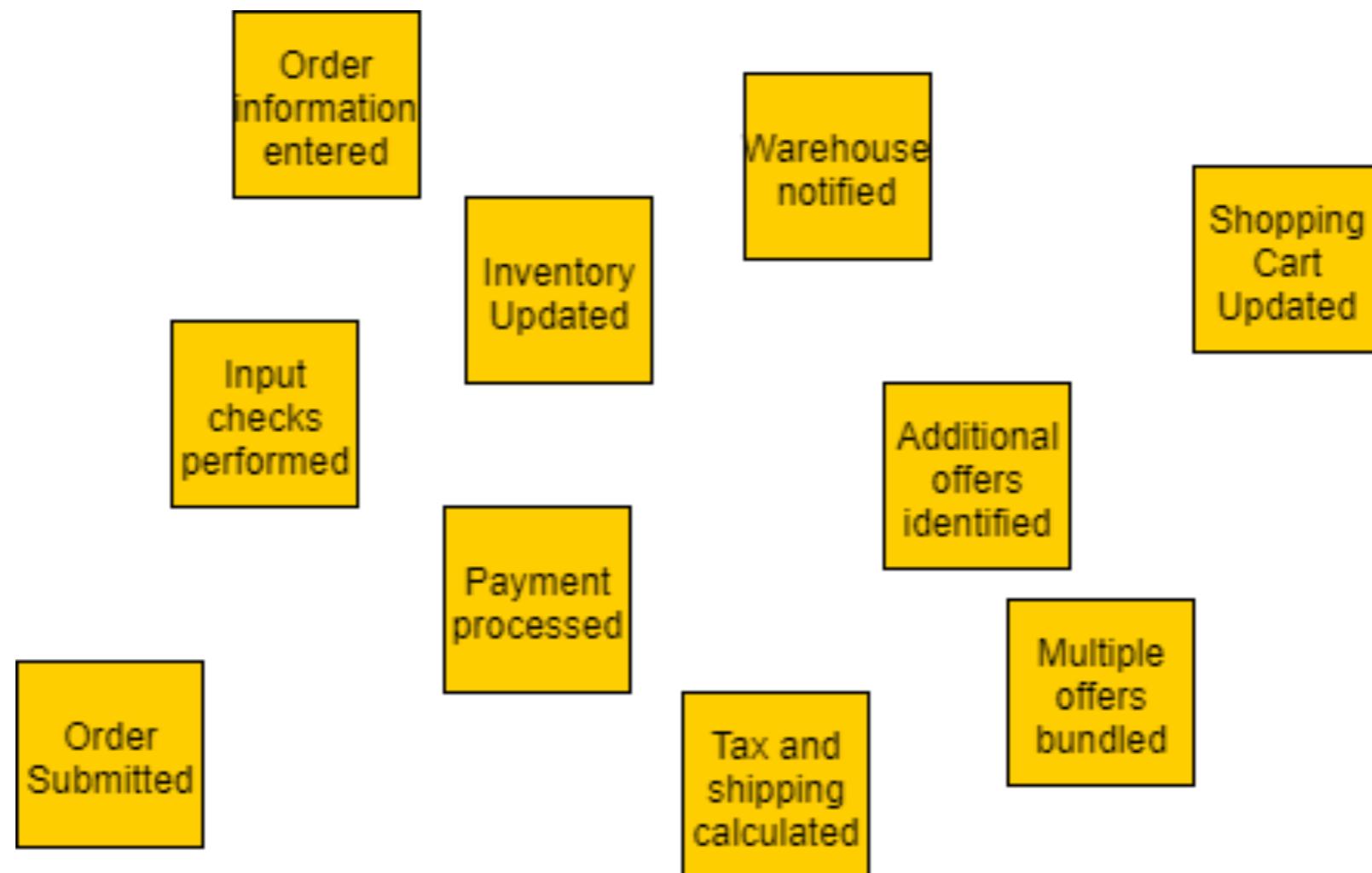


- Yellow represents the actors
- Blue represents the commands issued by the actors
- Orange are the events we started with

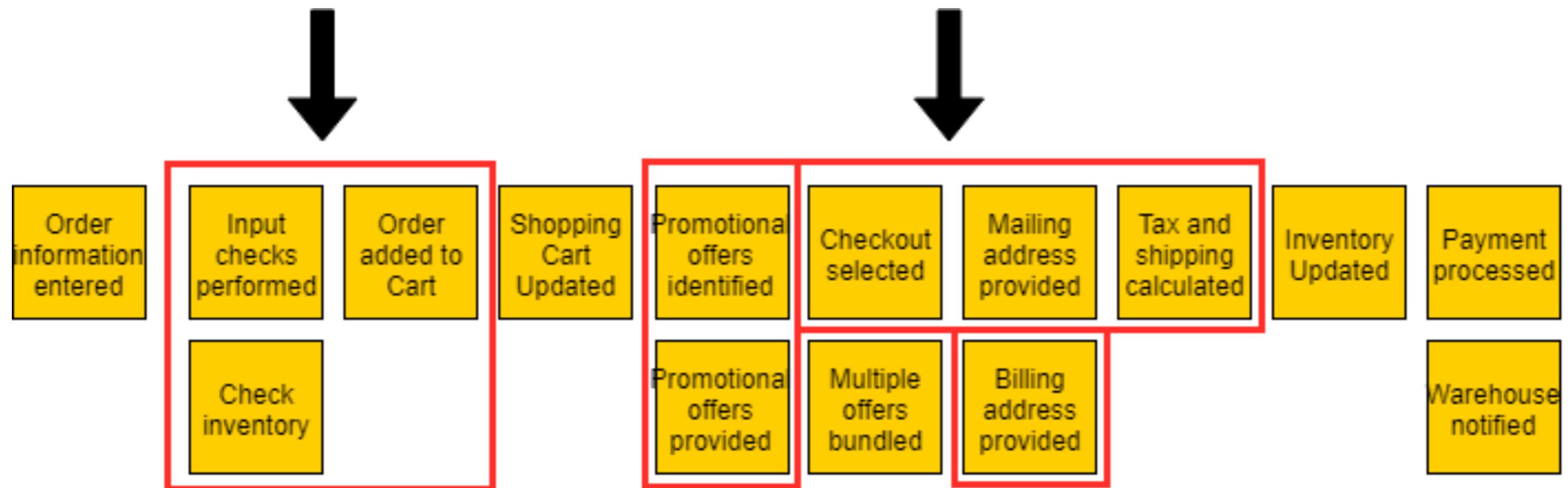
Identification of Aggregates



Step #1 – Event Discovery

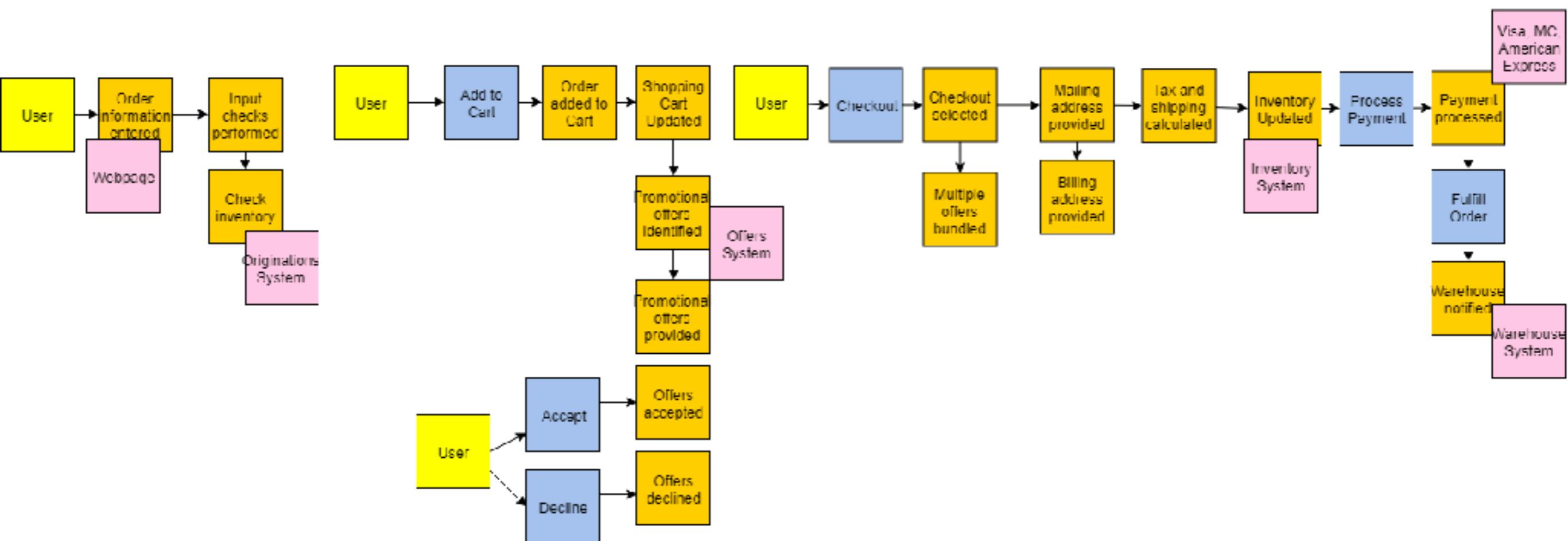


Step #2 – Placing the Events in Sequence

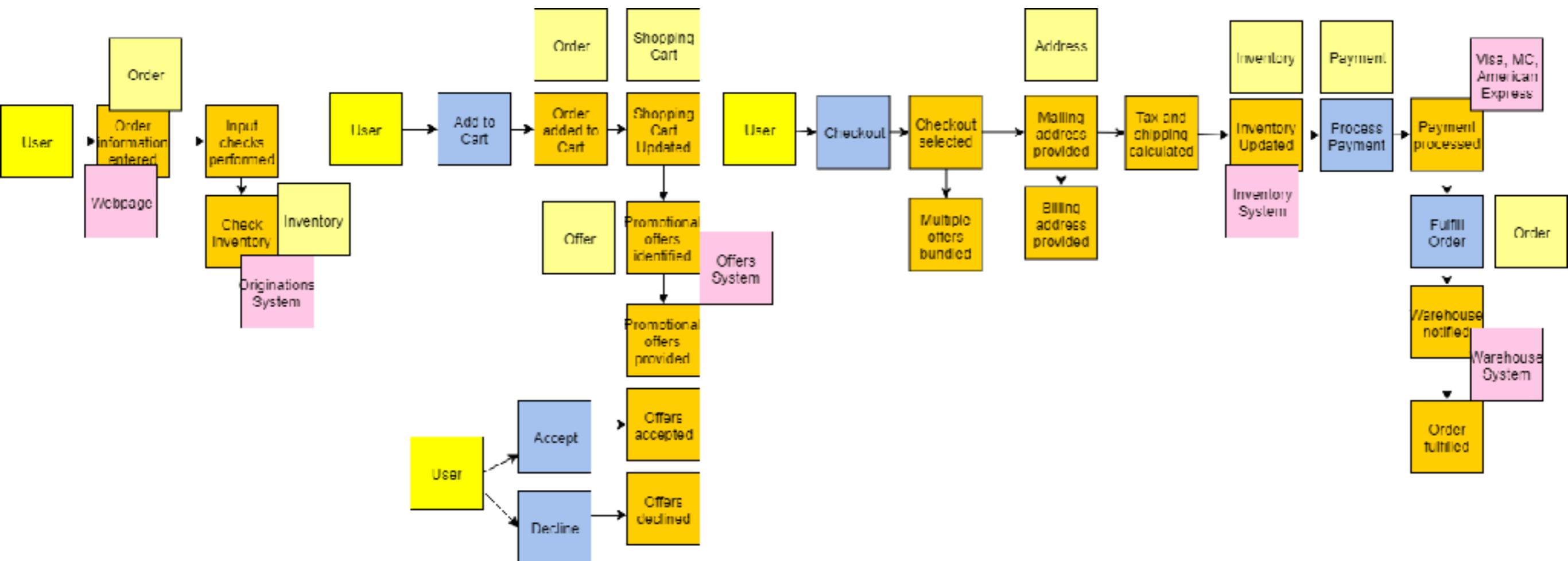


missing events (outlined in red)

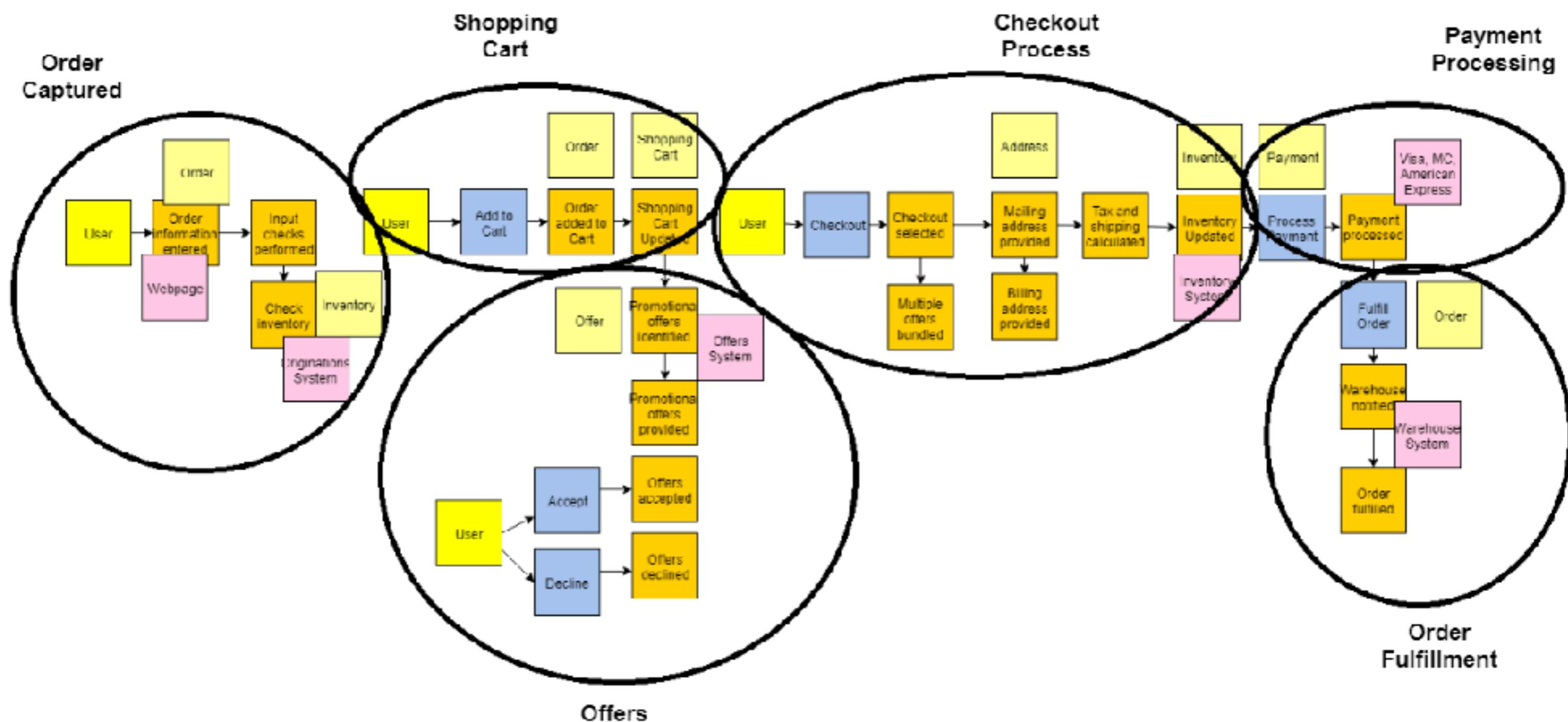
Step #3 – Modeling Out the Broader Ecosystem



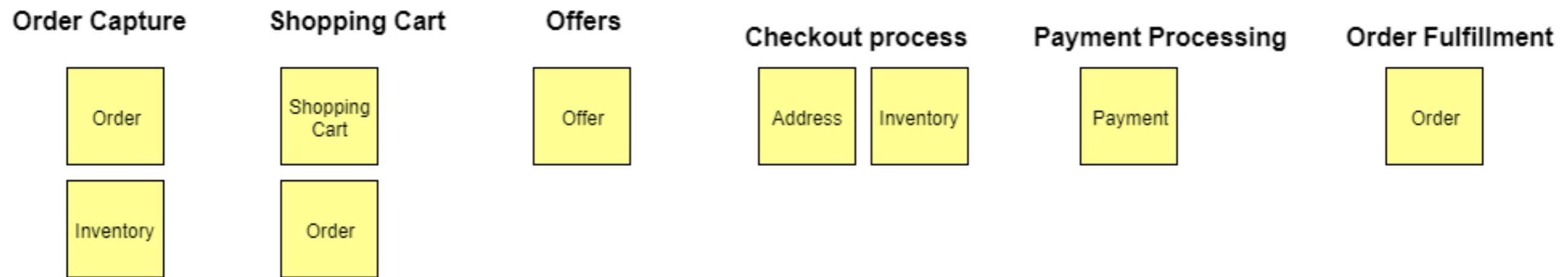
Step #4 – Identify Aggregates



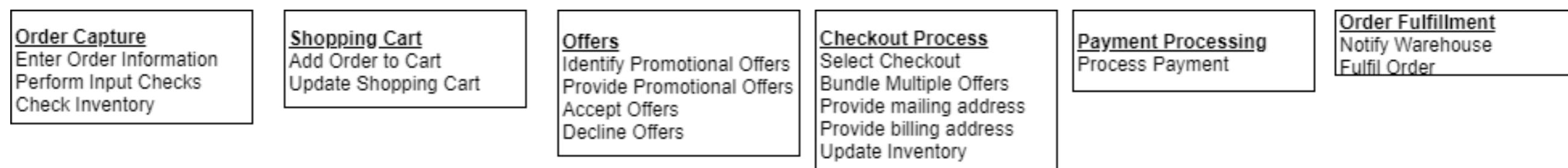
Step #5 – Bounded Context Categorization of Events

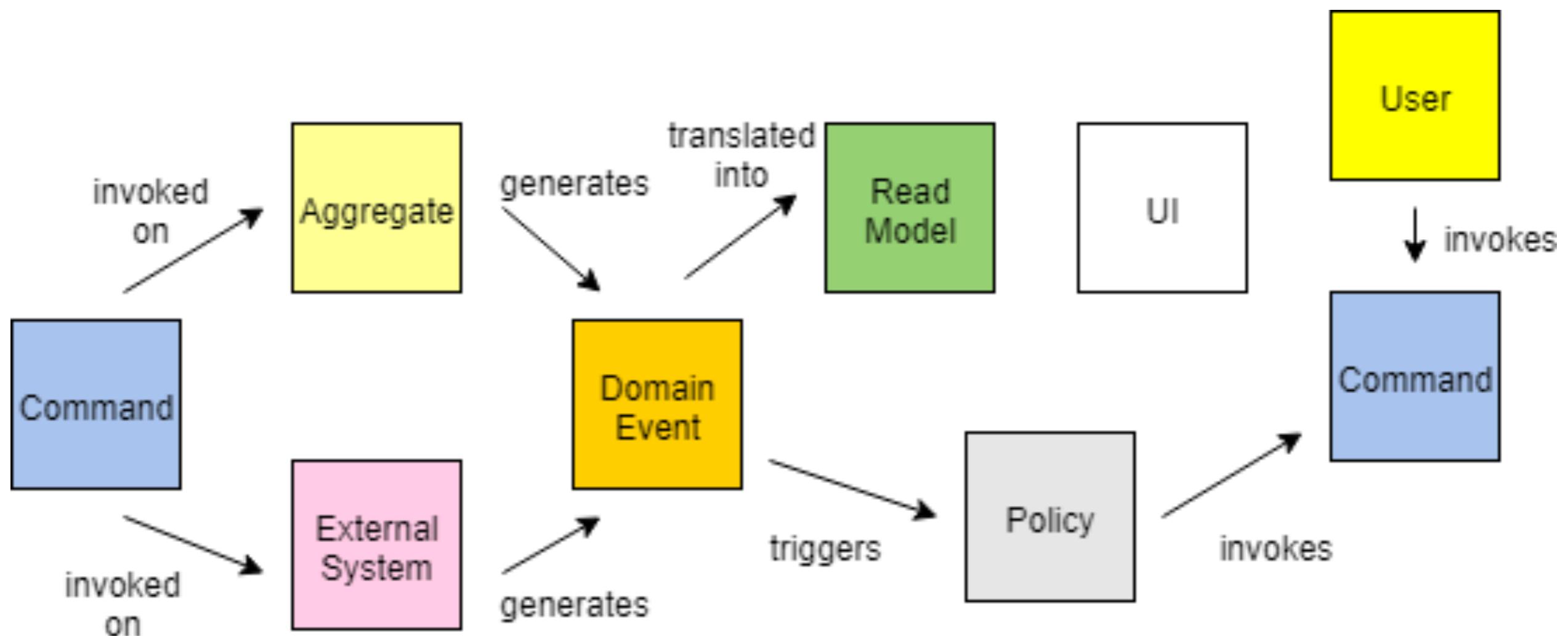


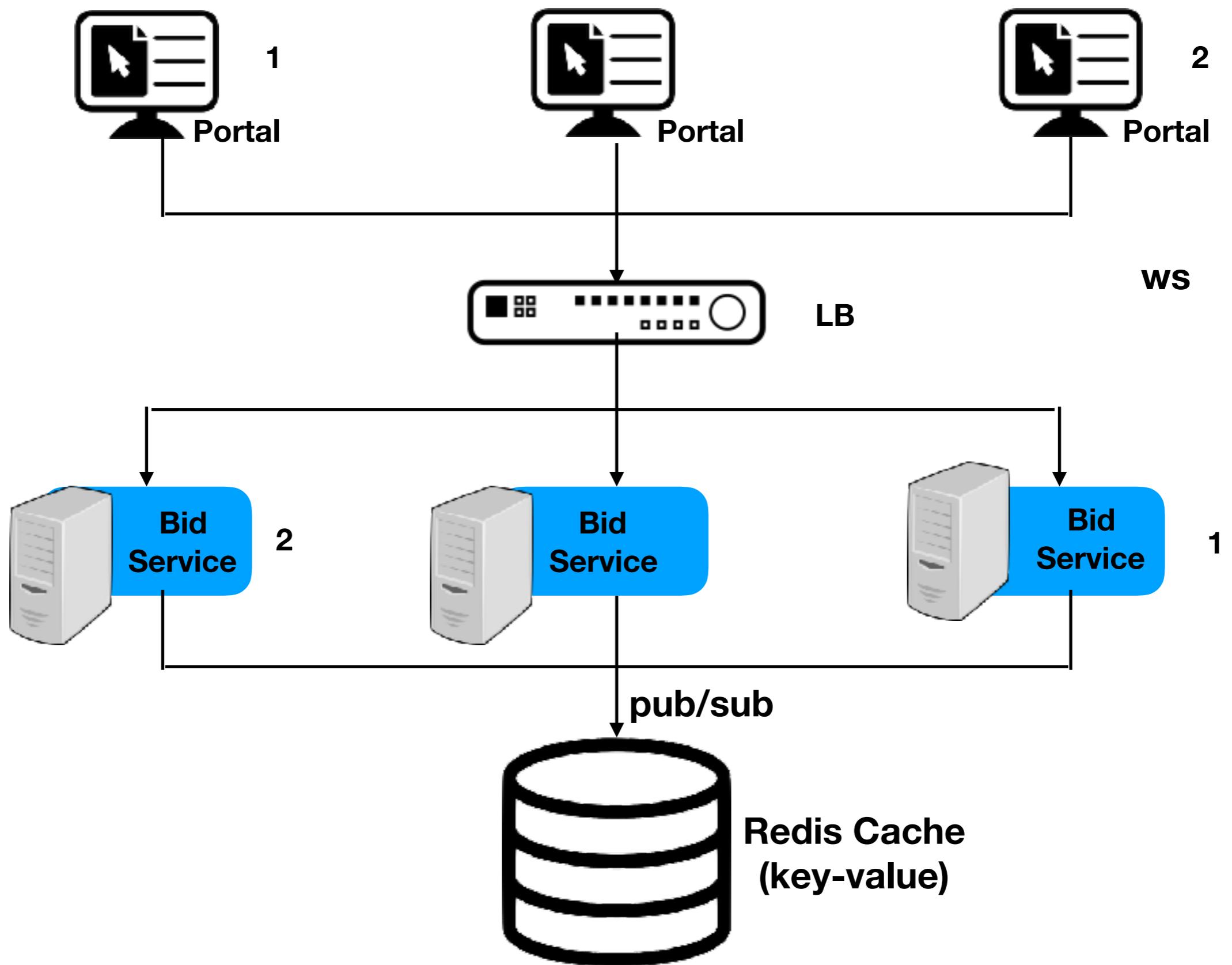
Step #6: Create Services



Step #7: Create Capabilities





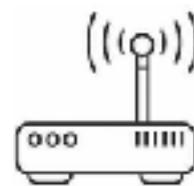


Device registration
Device communication
Protocol translation

Sensor



Connected vehicles,
Fleet management



Field
Gateway

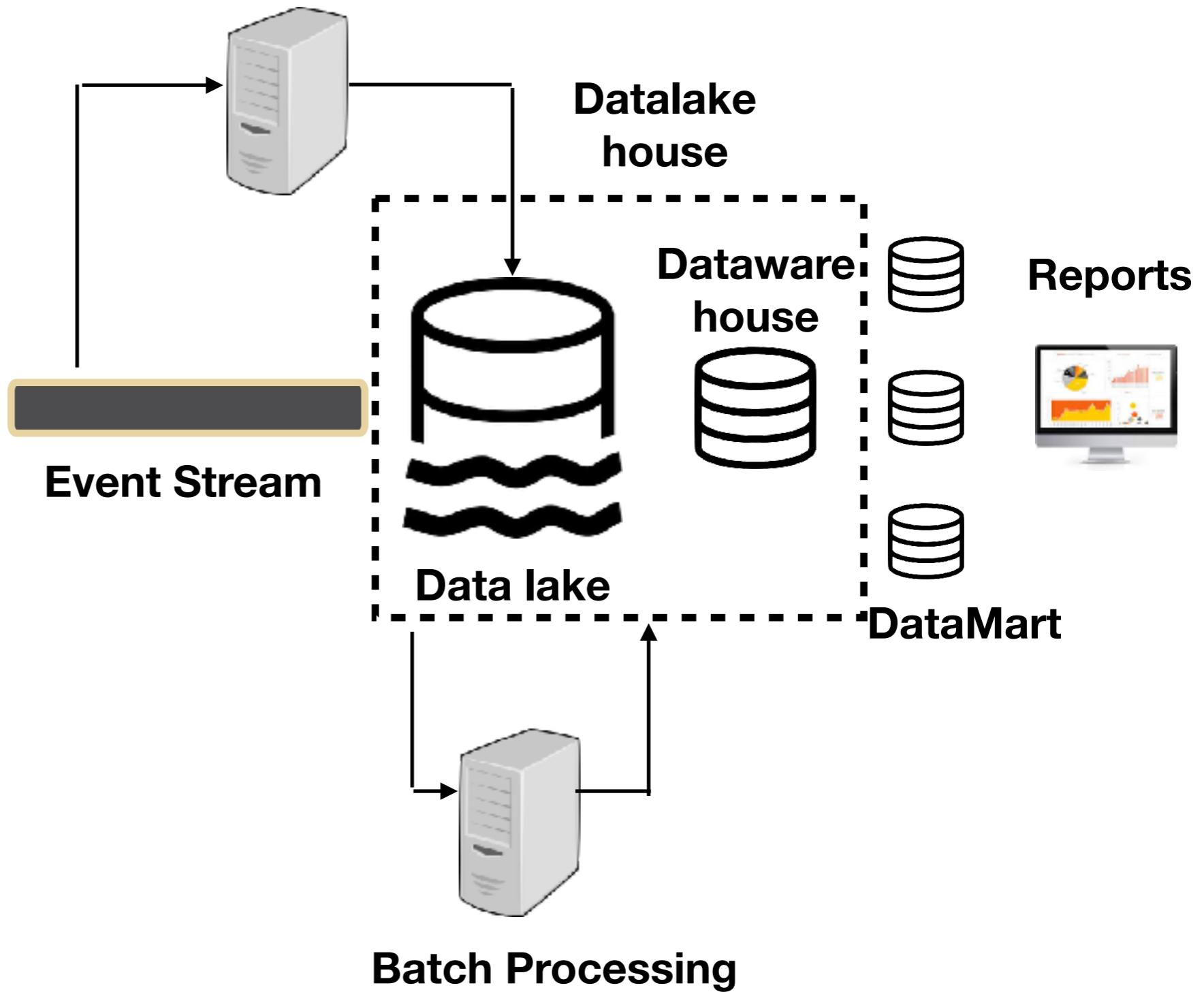
Cloud
Gateway



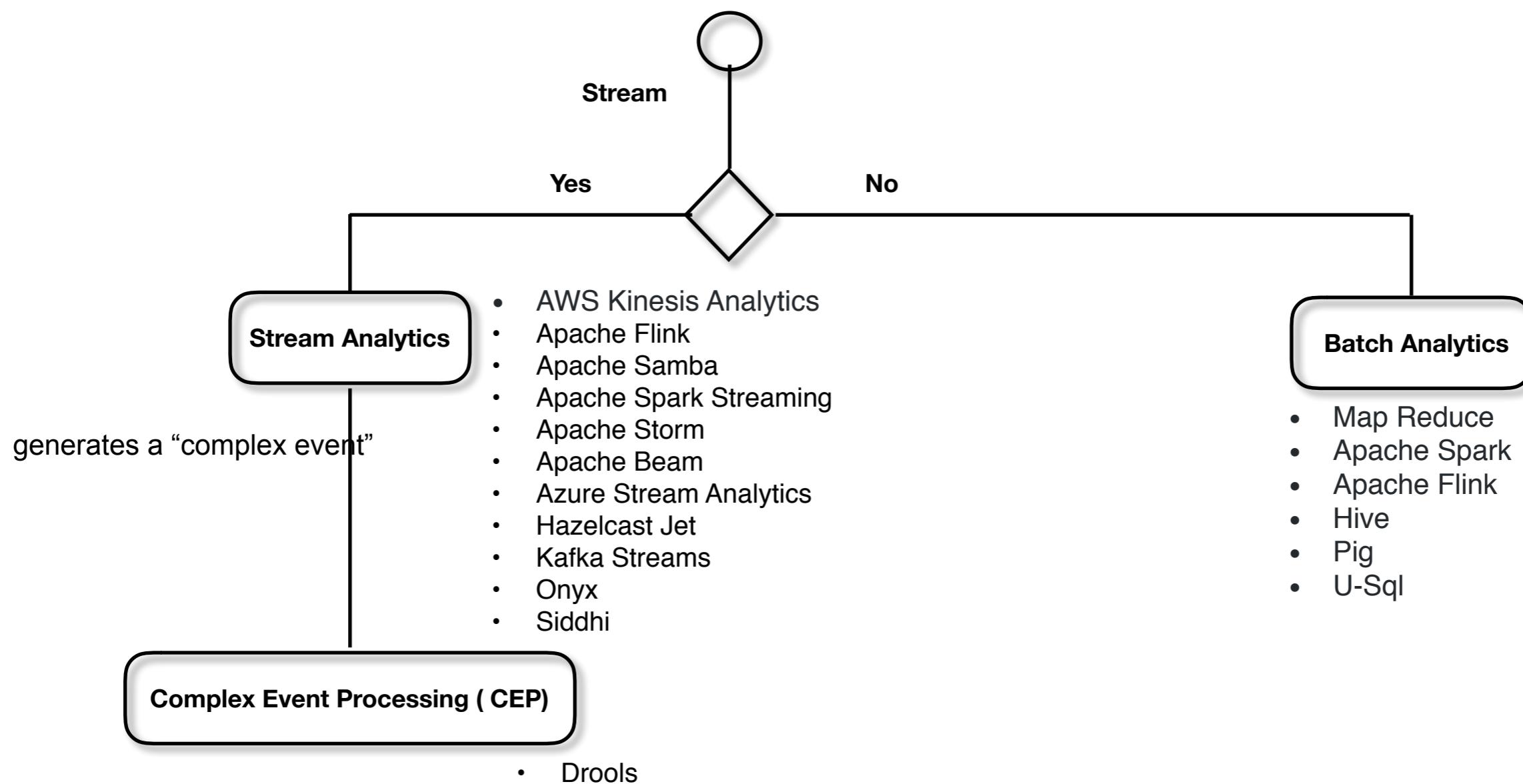
Connected
manufacturing

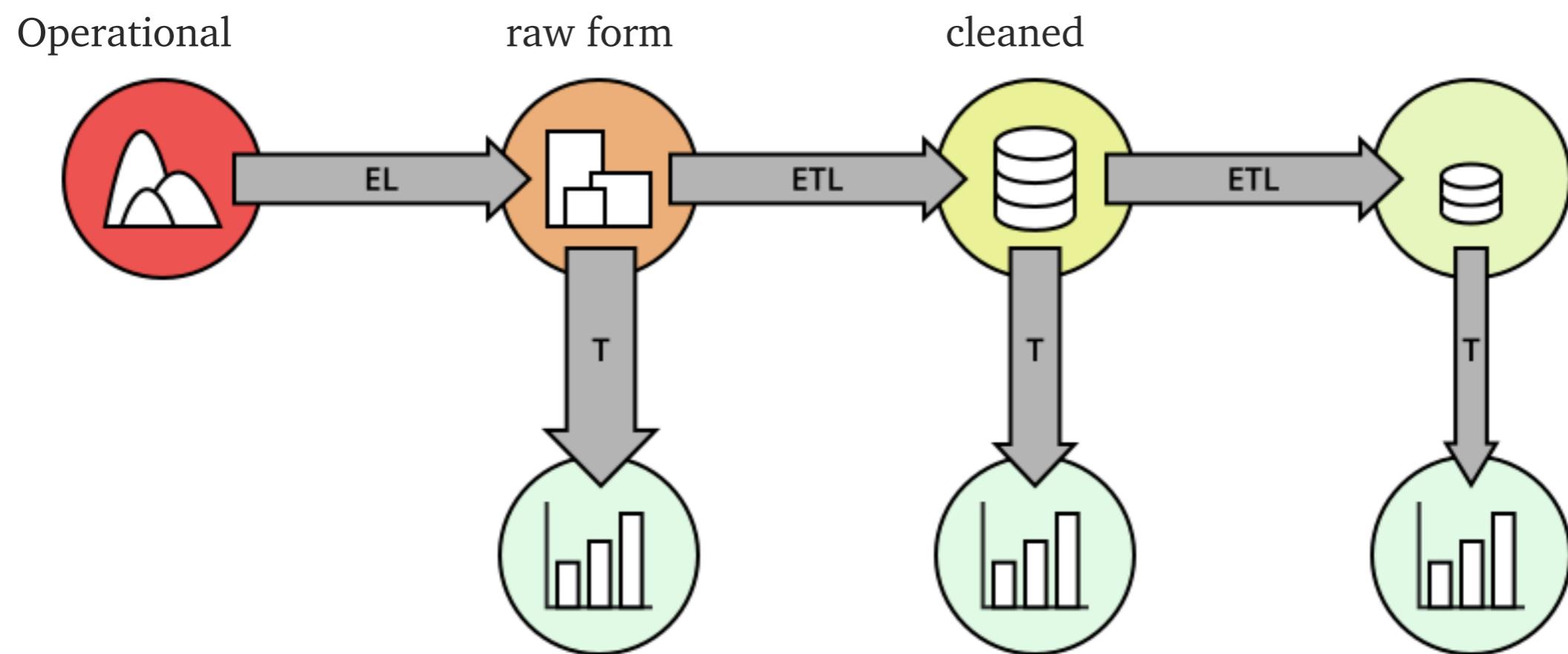
Facilities
management

Stream Processing



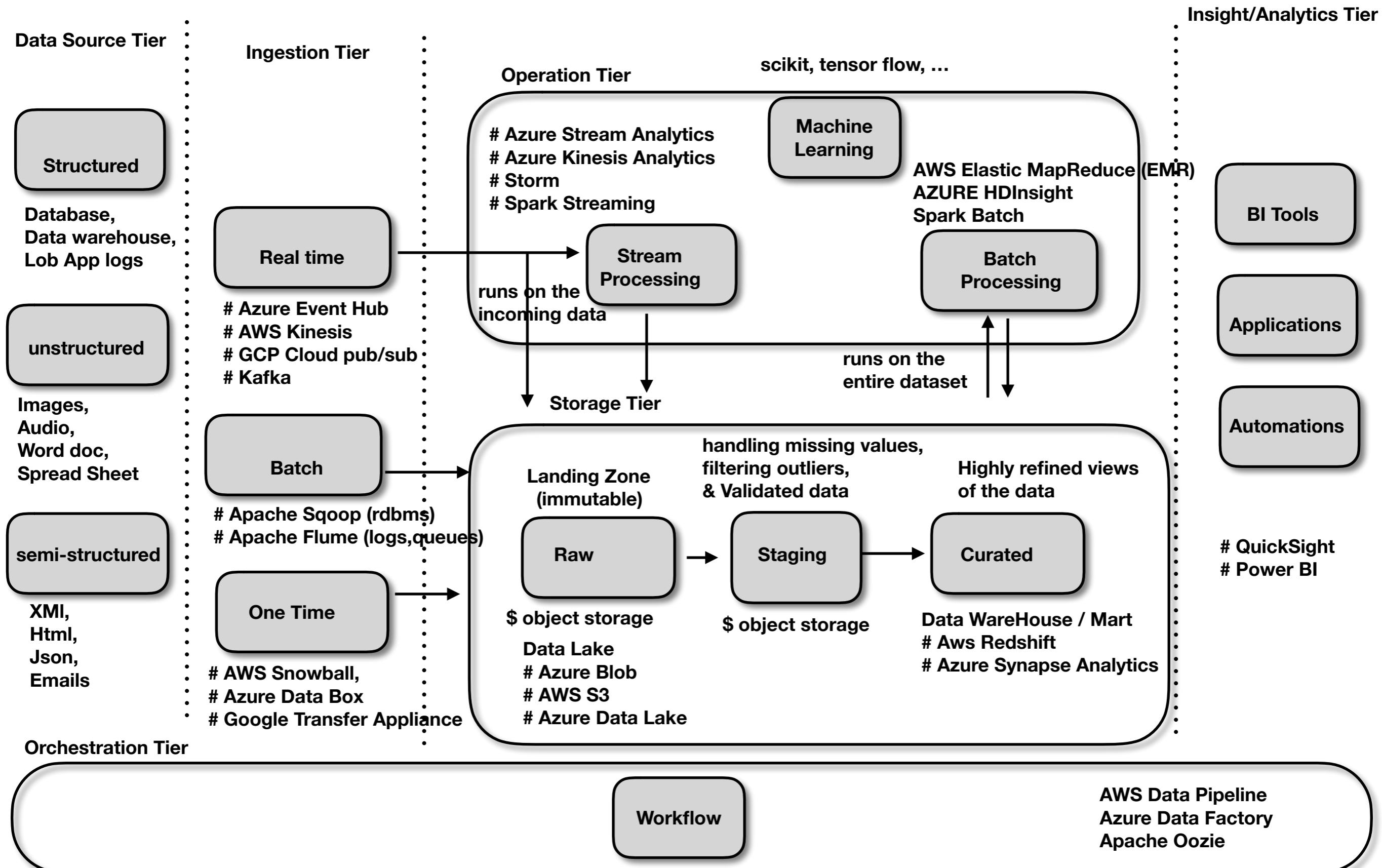
Choose Analytical Compute



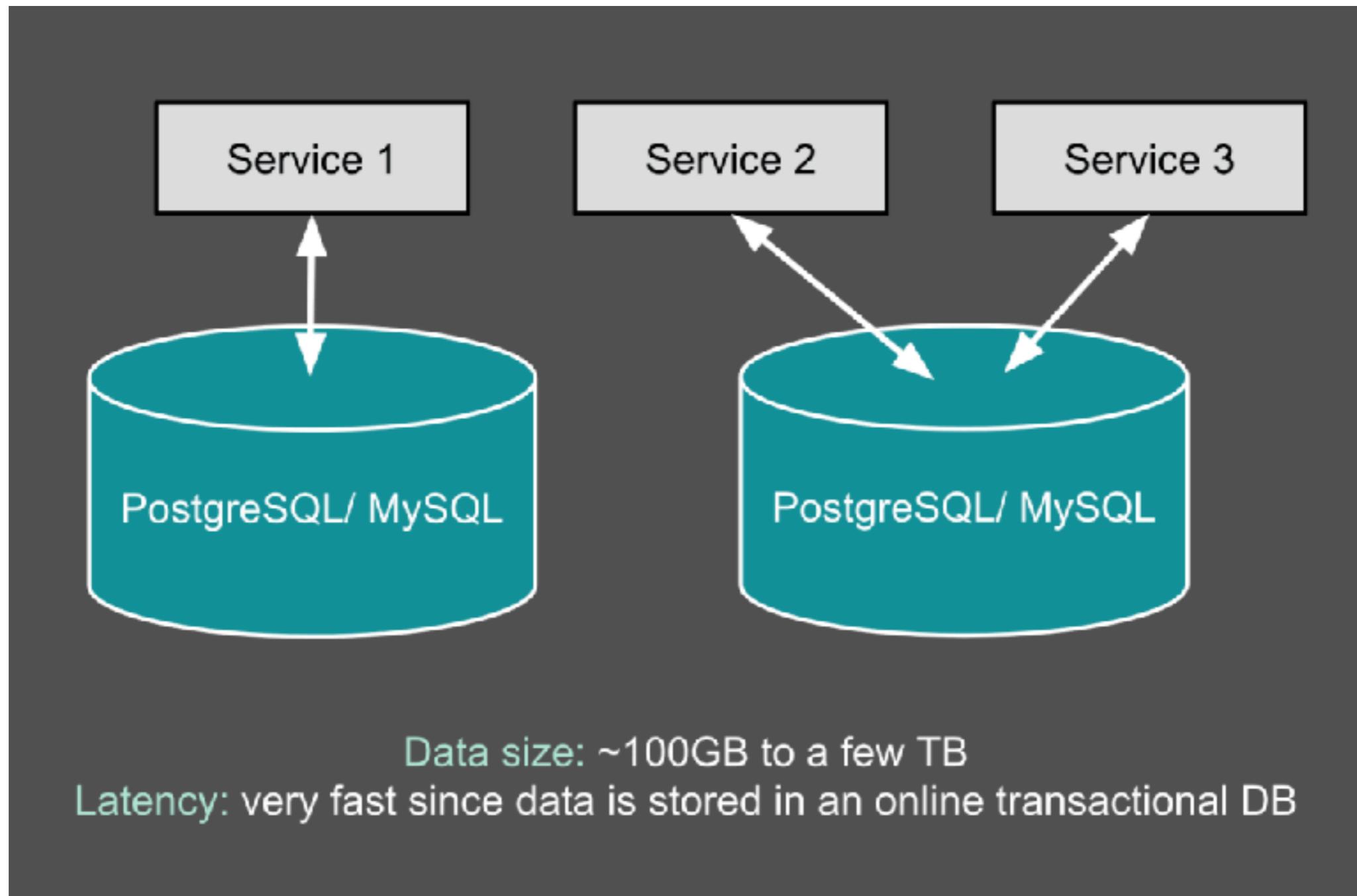


Reference Architecture

Analytical Application

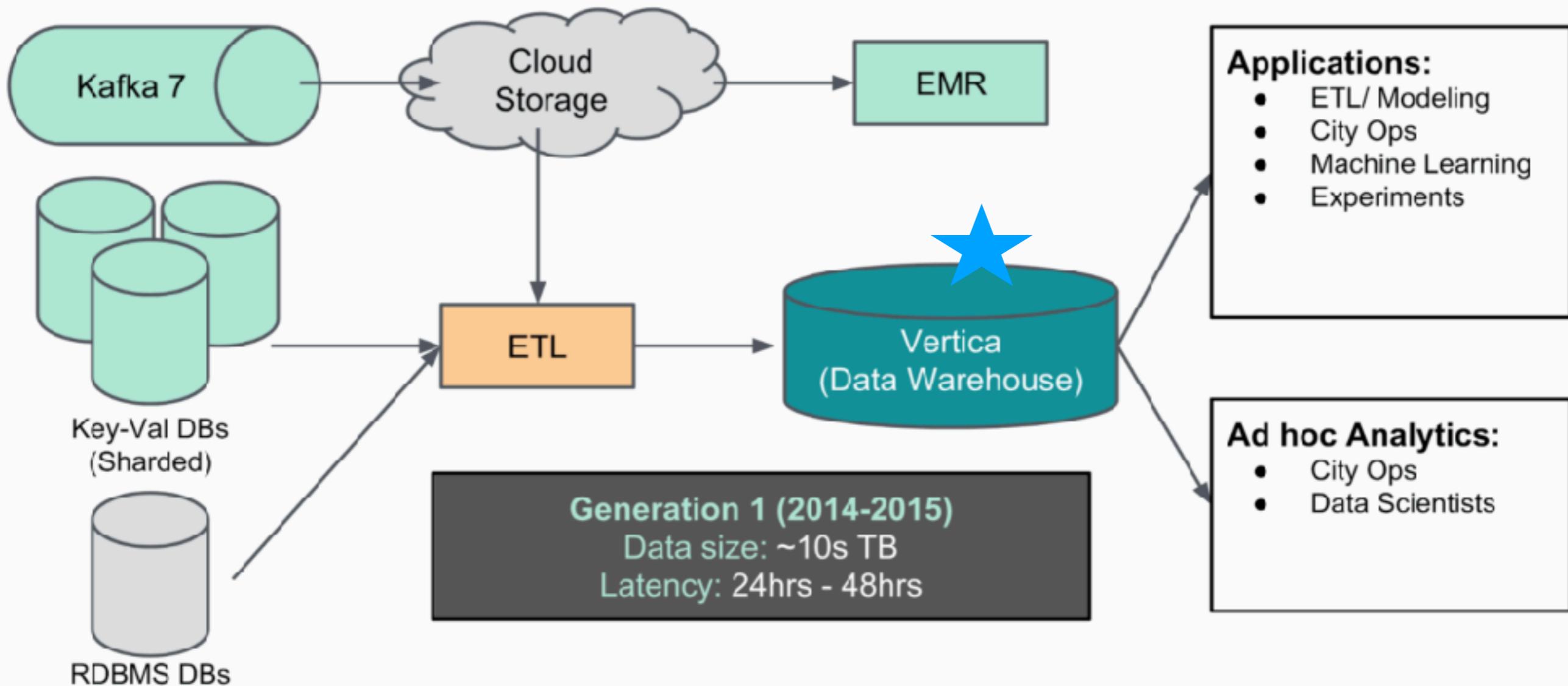


Generation 1: The beginning of Big Data at Uber (Before 2014)



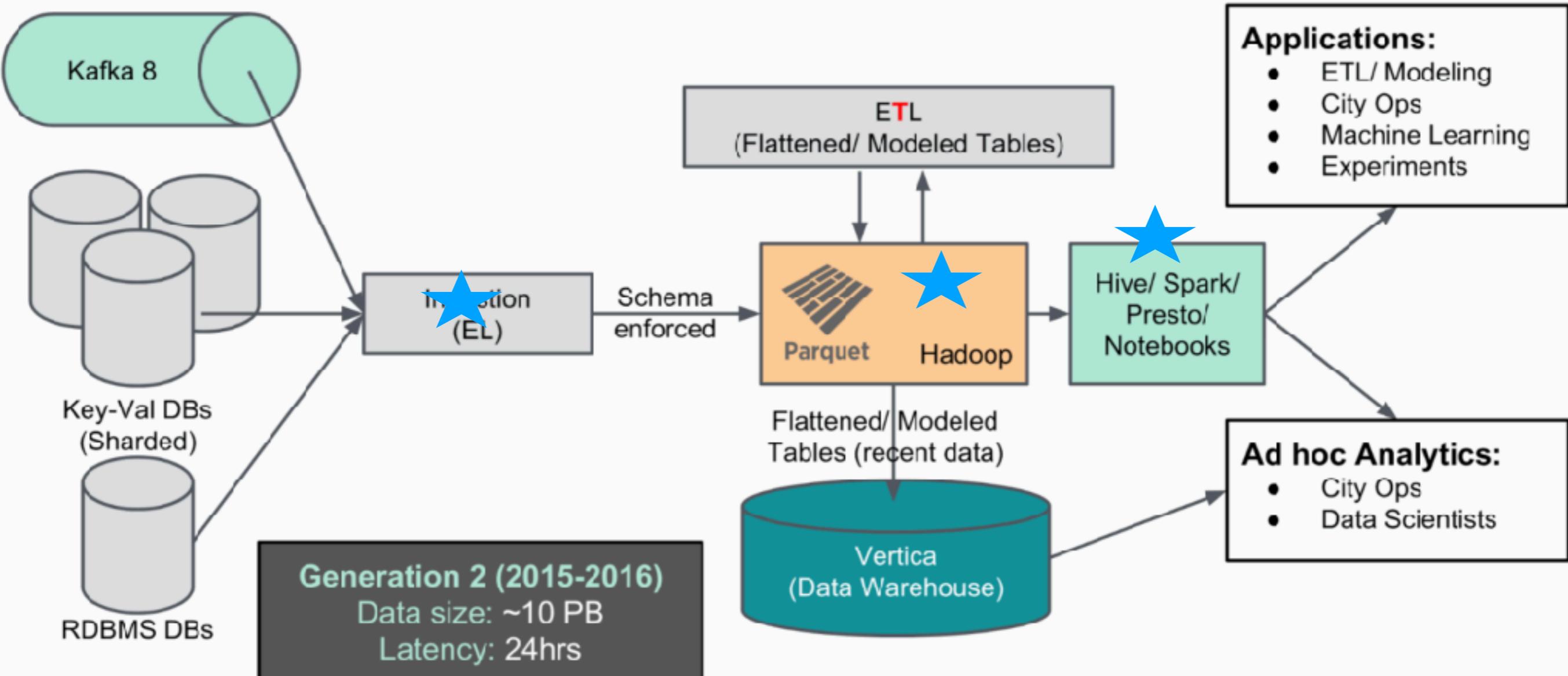
Data was scattered across different OLTP databases.

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.

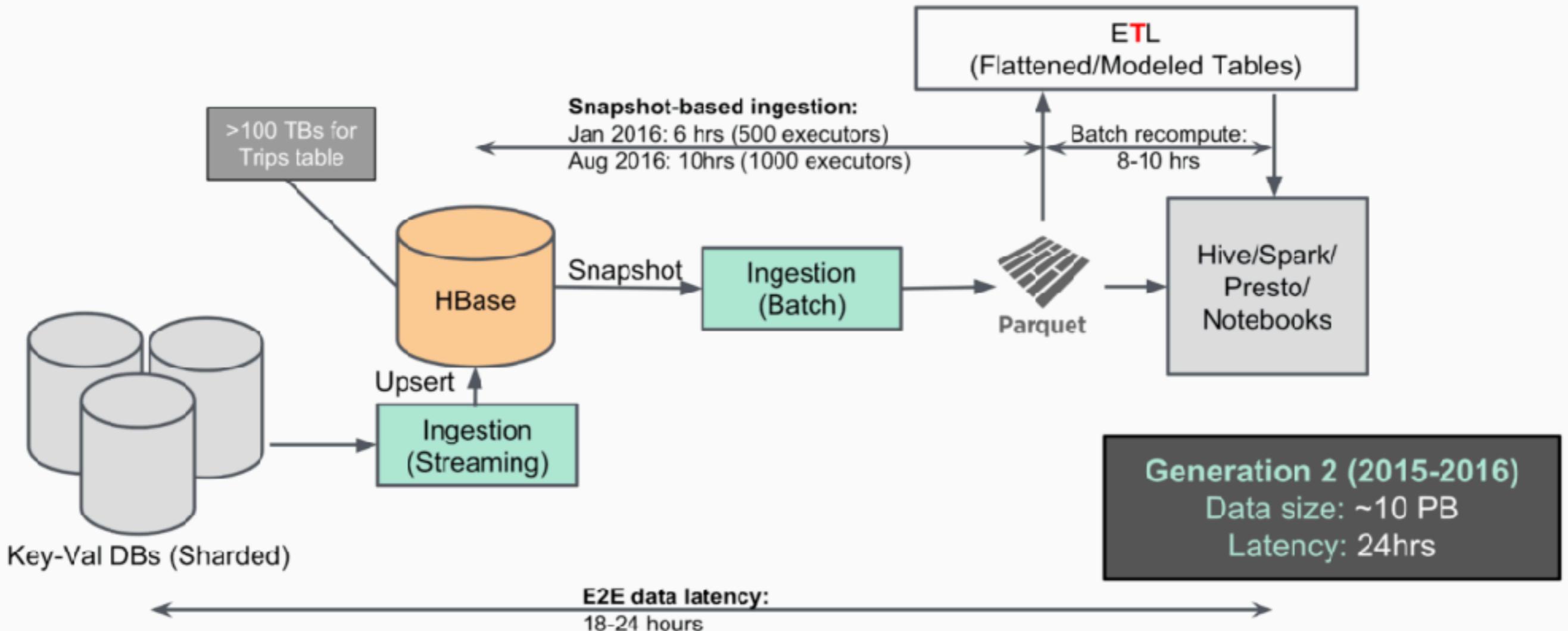
Generation 2 (2015-2016) - The arrival of Hadoop



Data was ingested with no transformation during ingestion.

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

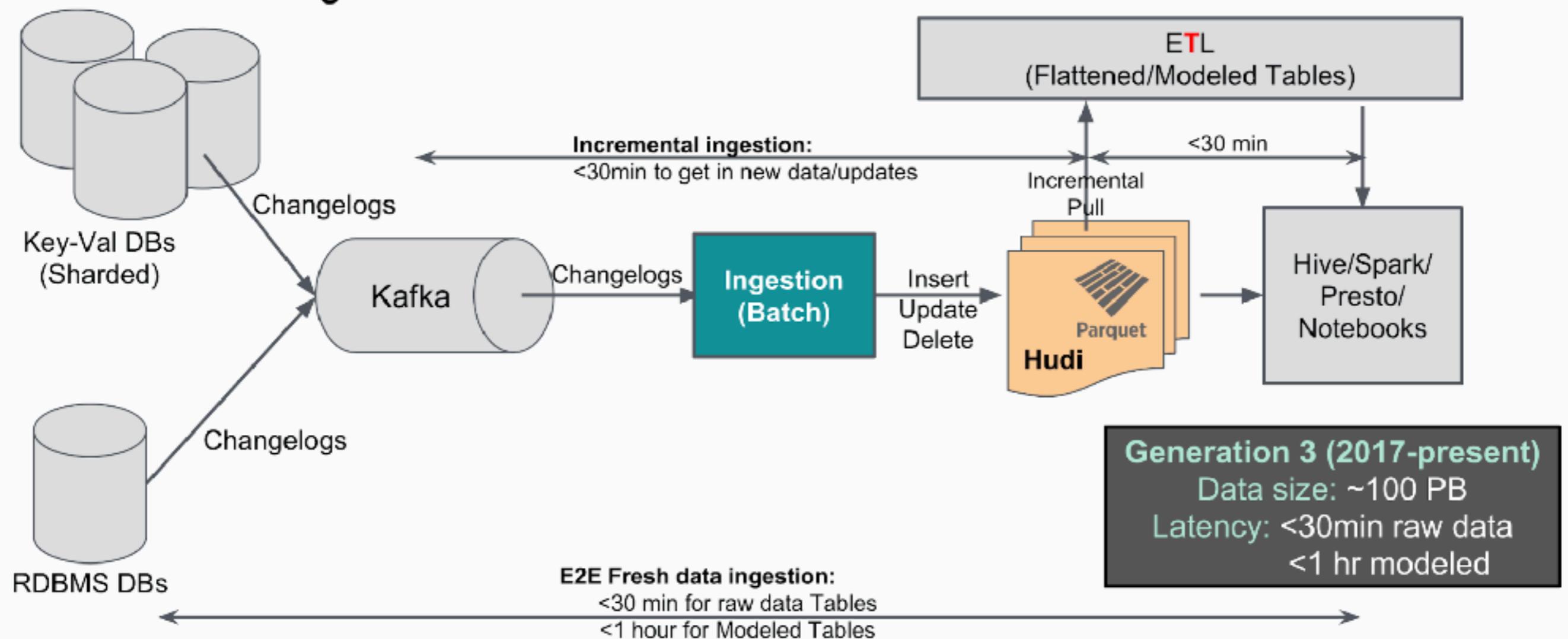
Why does data latency remain at 24 hours?



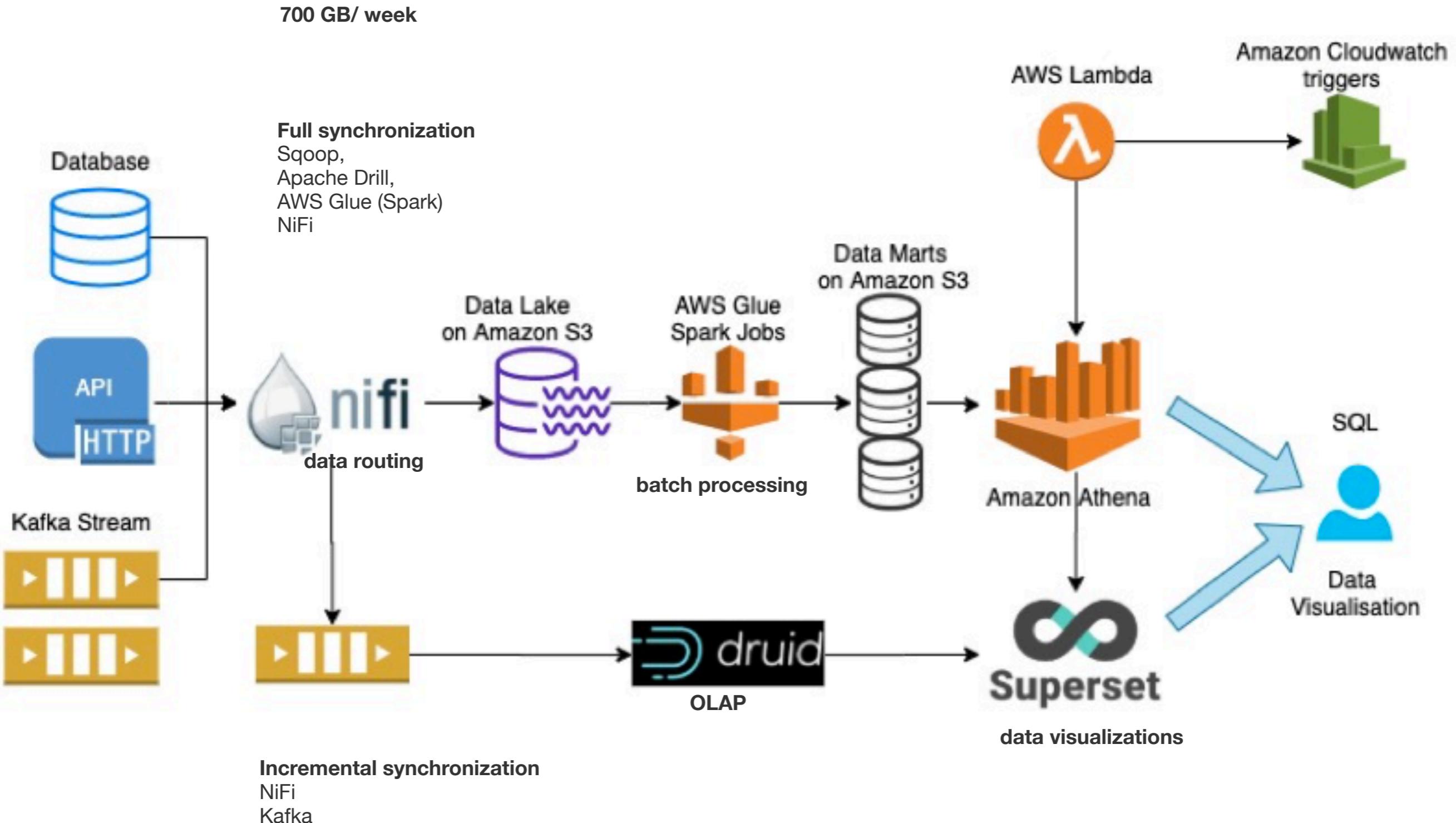
only over 100 gb of new data was added every day. Since HDFS and Parquet do not support data updates, each run of the ingestion job had to convert the entire, over 100 tb dataset for that specific table.

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

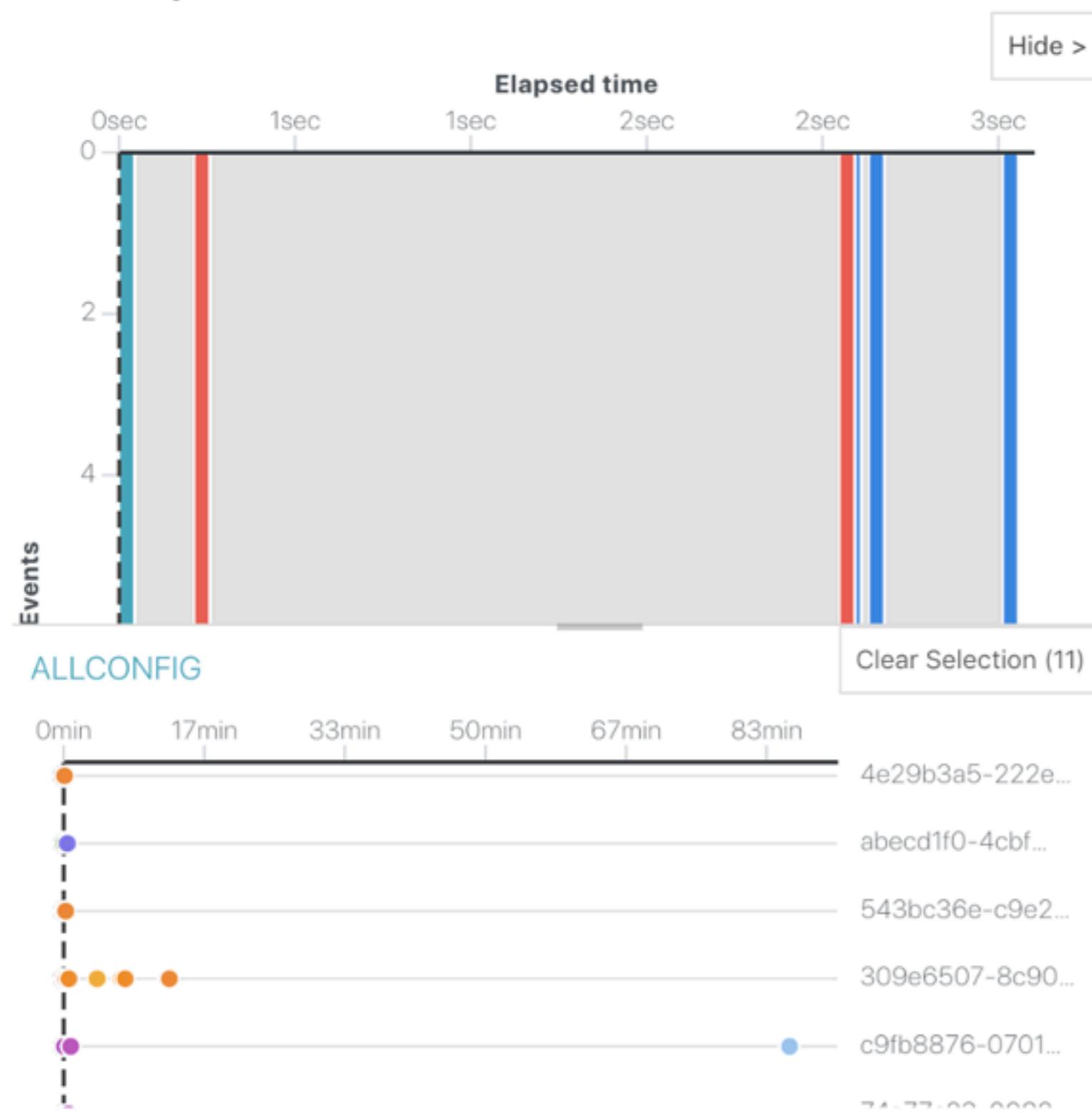
Incremental ingestion:



Redbus



Event Sequence

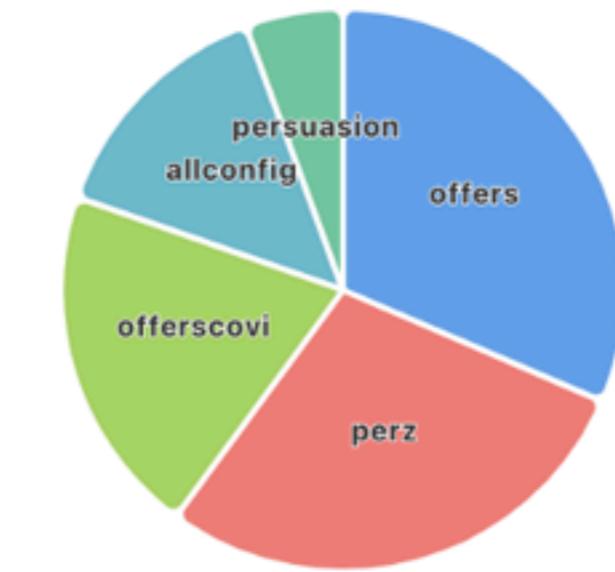


Align sequences by

1st - + event

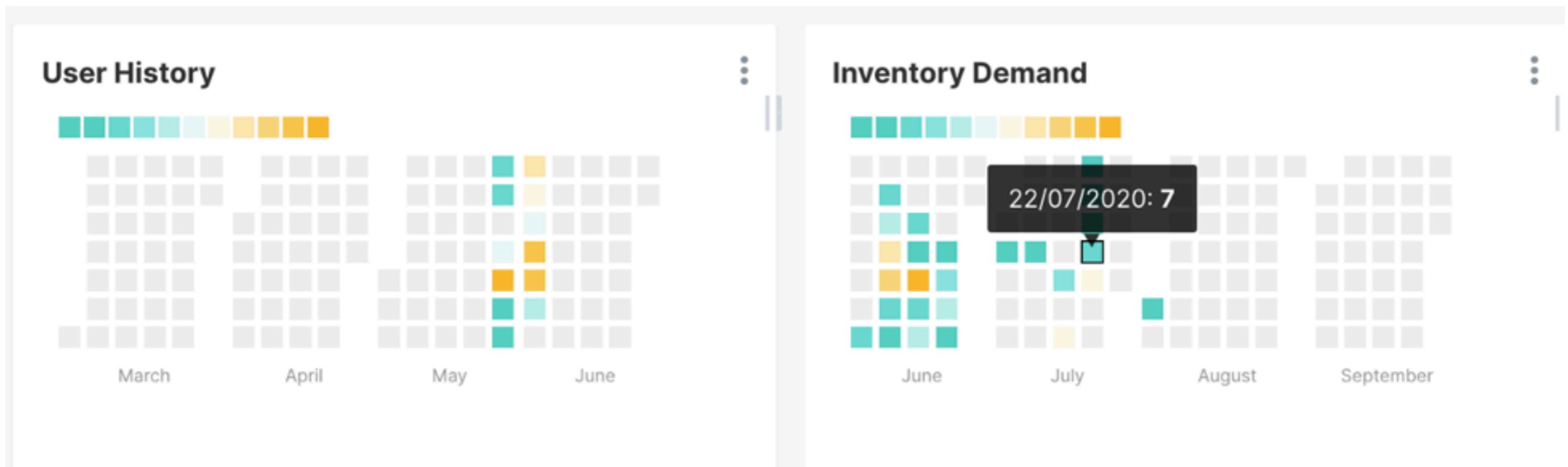
Event type summary

78 events (427 hidden [84.6%])

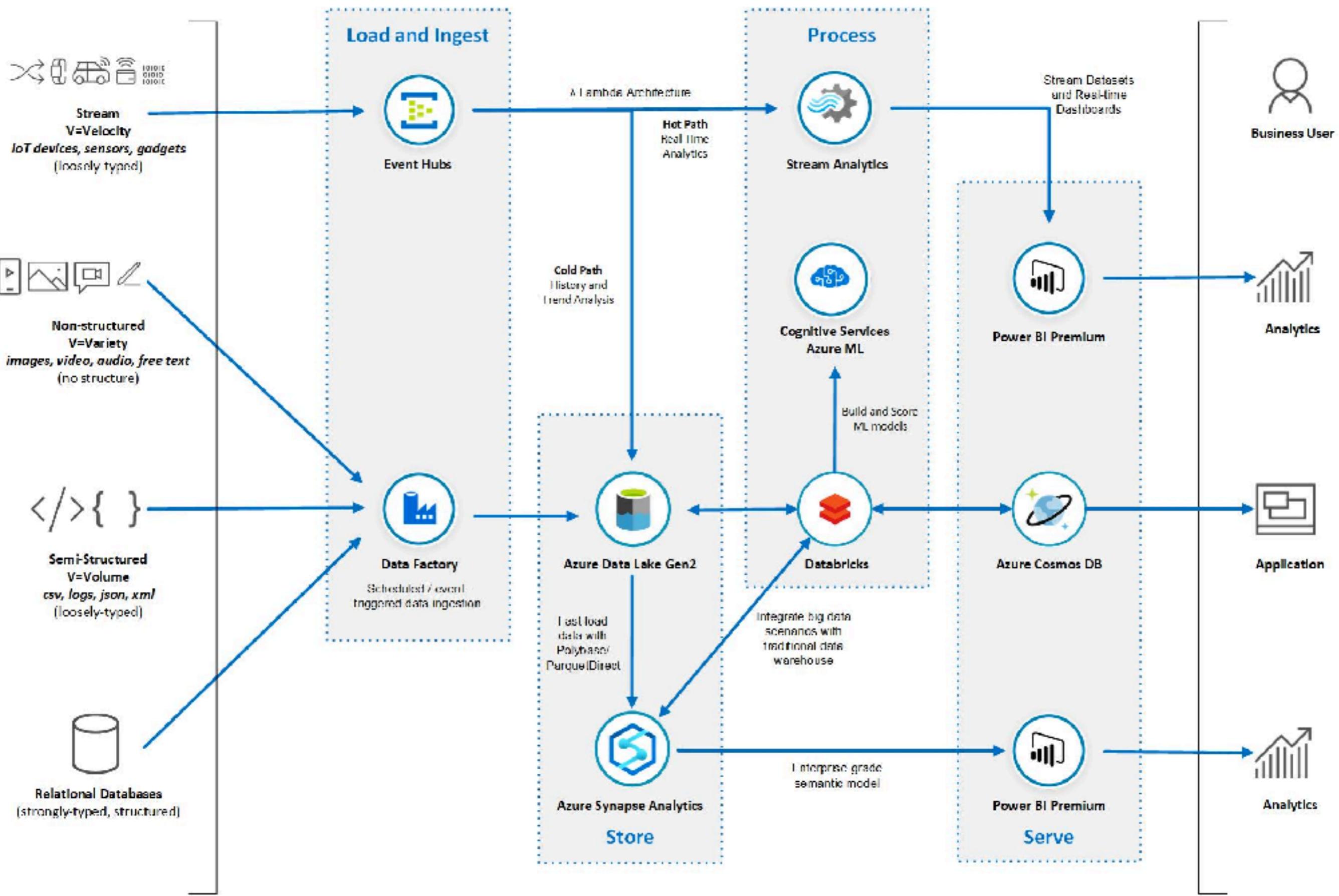


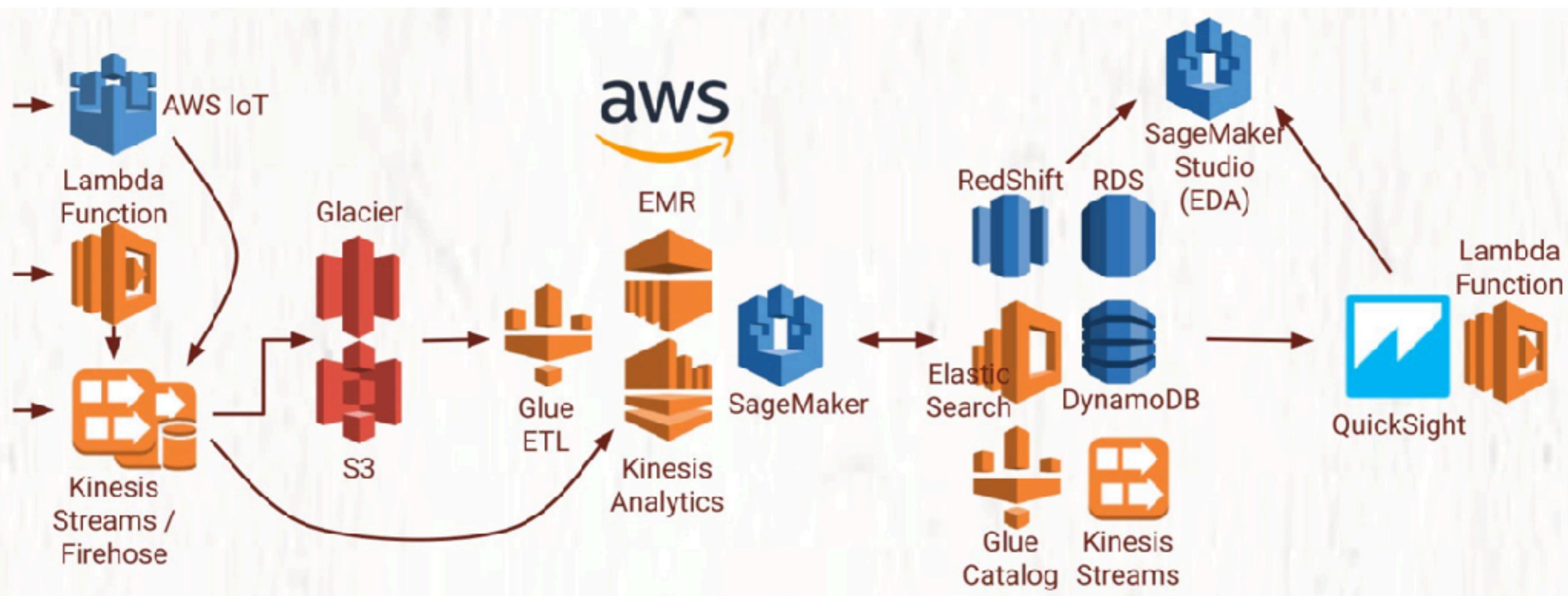
- allconfig (14.1%)
- busdetails
- busroutes
- filters
- offers (35.9%)
- offerscovid (23.1%)
- persuasion
- perz (26.9%)
- rides
- seatlayout
- signin

Calendar heat-map

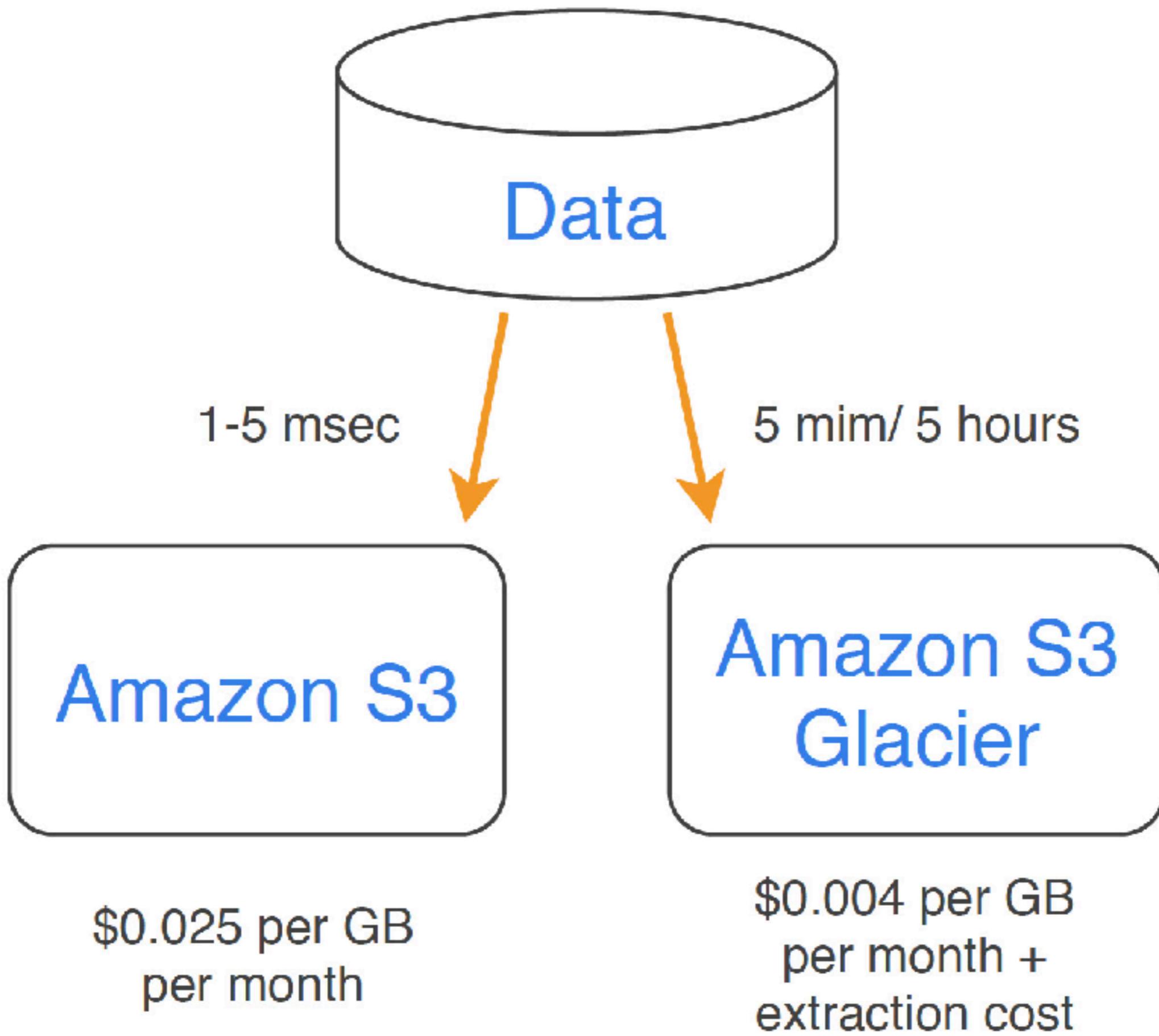


Modern Data Platform Reference Architecture





		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.	



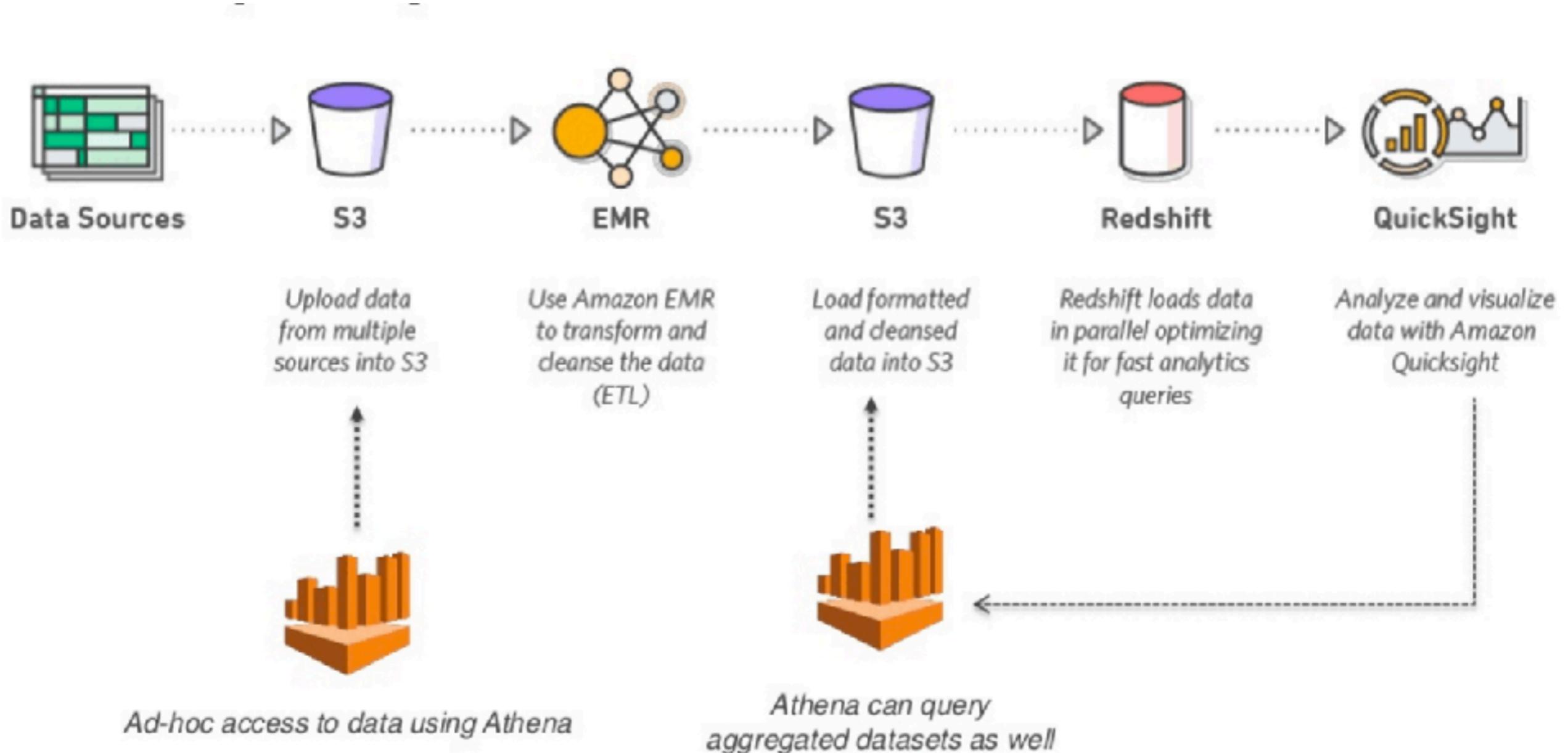
	S3	S3 Glacier
Access speed	Hi	Low
API availability	Yes	Yes
Data storage cost	Relatively high	Very low
Object size	Up to 5 Tb	40 Tb
Accommodation in regions	Many regions	Average
Static Web Content	Yes	No
Supporting Versioning	Yes	No

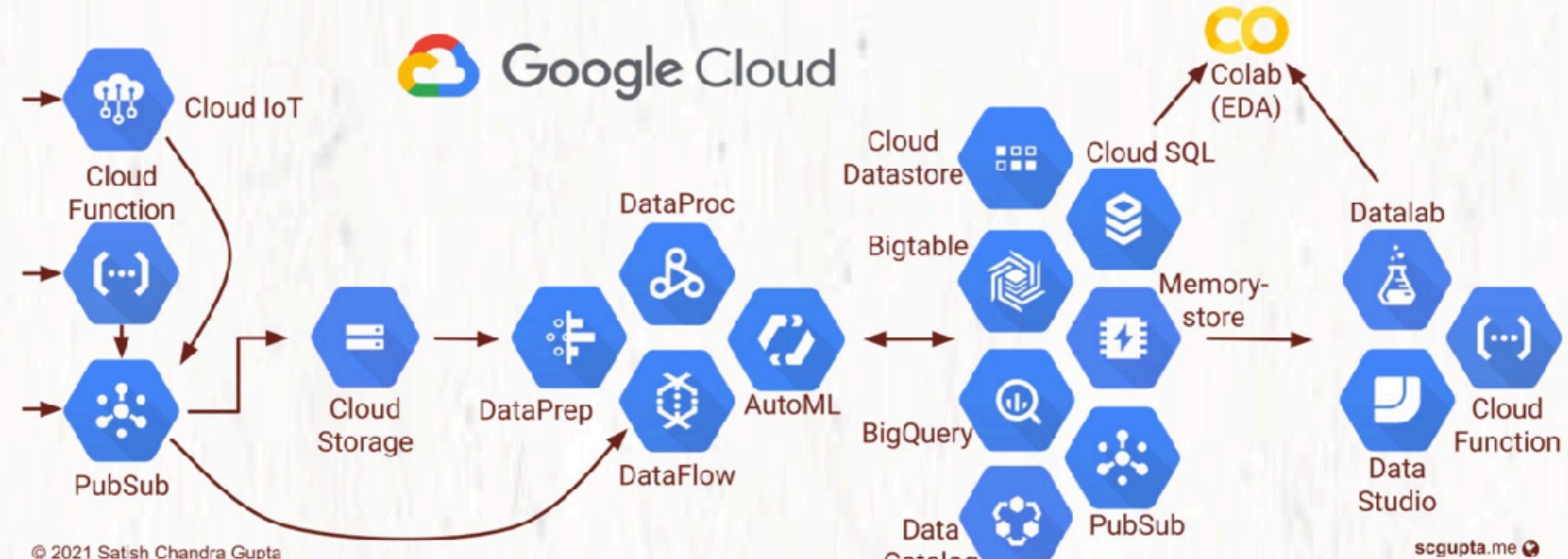
	Data Streams	Data Firehose	Data Analytics	Video Streams
Short definition	Scalable and durable real-time data streaming service.	Capture, transform, and deliver streaming data into data lakes, data stores, and analytics services.	Transform and analyze streaming data in real time with Apache Flink.	Stream video from connected devices to AWS for analytics, machine learning, playback, and other processing.
Data sources	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.
Data consumers	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
Use cases	<ul style="list-style-type: none"> – Log and event data collection – Real-time analytics – Mobile data capture – Gaming data feed 	<ul style="list-style-type: none"> – IoT Analytics – Clickstream Analytics – Log Analytics – Security monitoring 	<ul style="list-style-type: none"> – Streaming ETL – Real-time analytics – Stateful event processing 	<ul style="list-style-type: none"> – Smart technologies – Video-related AI/ML – Video processing

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. |



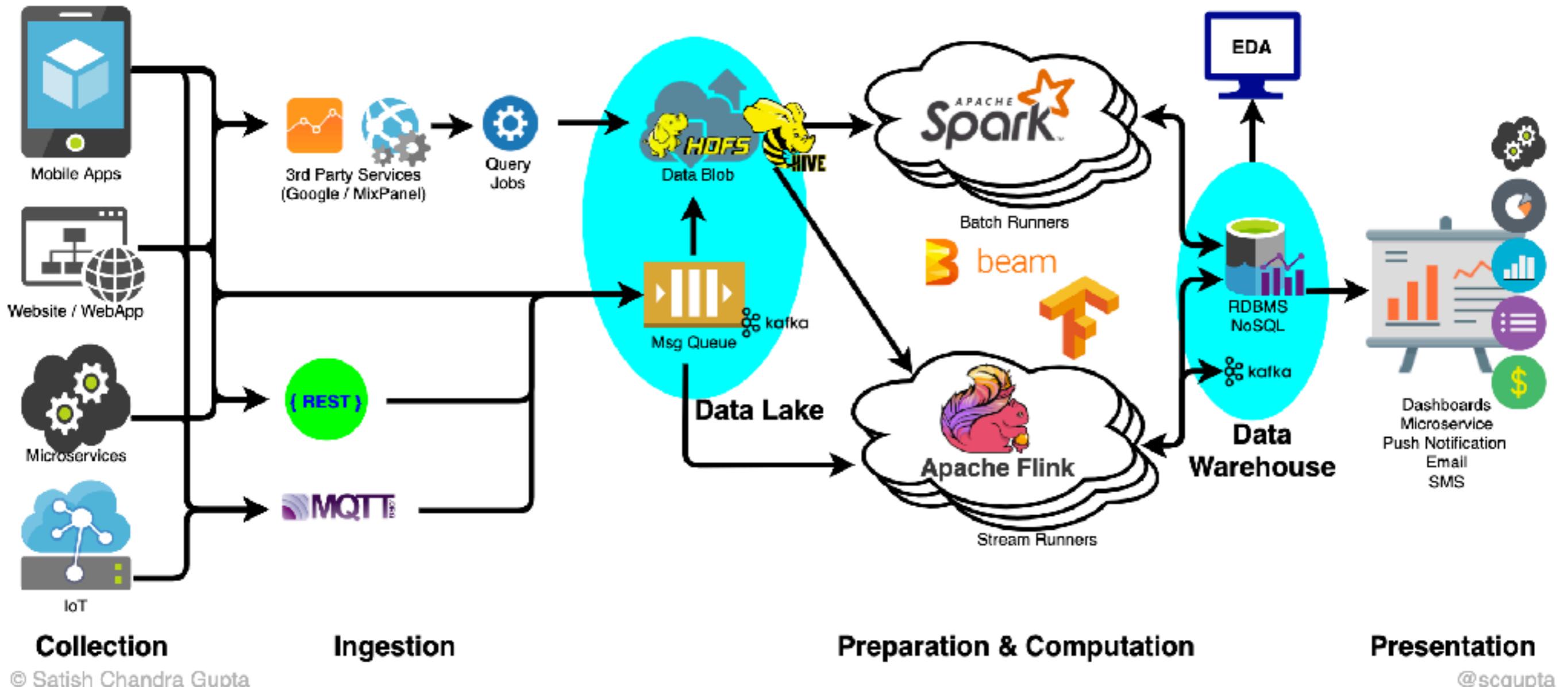


© 2021 Satish Chandra Gupta

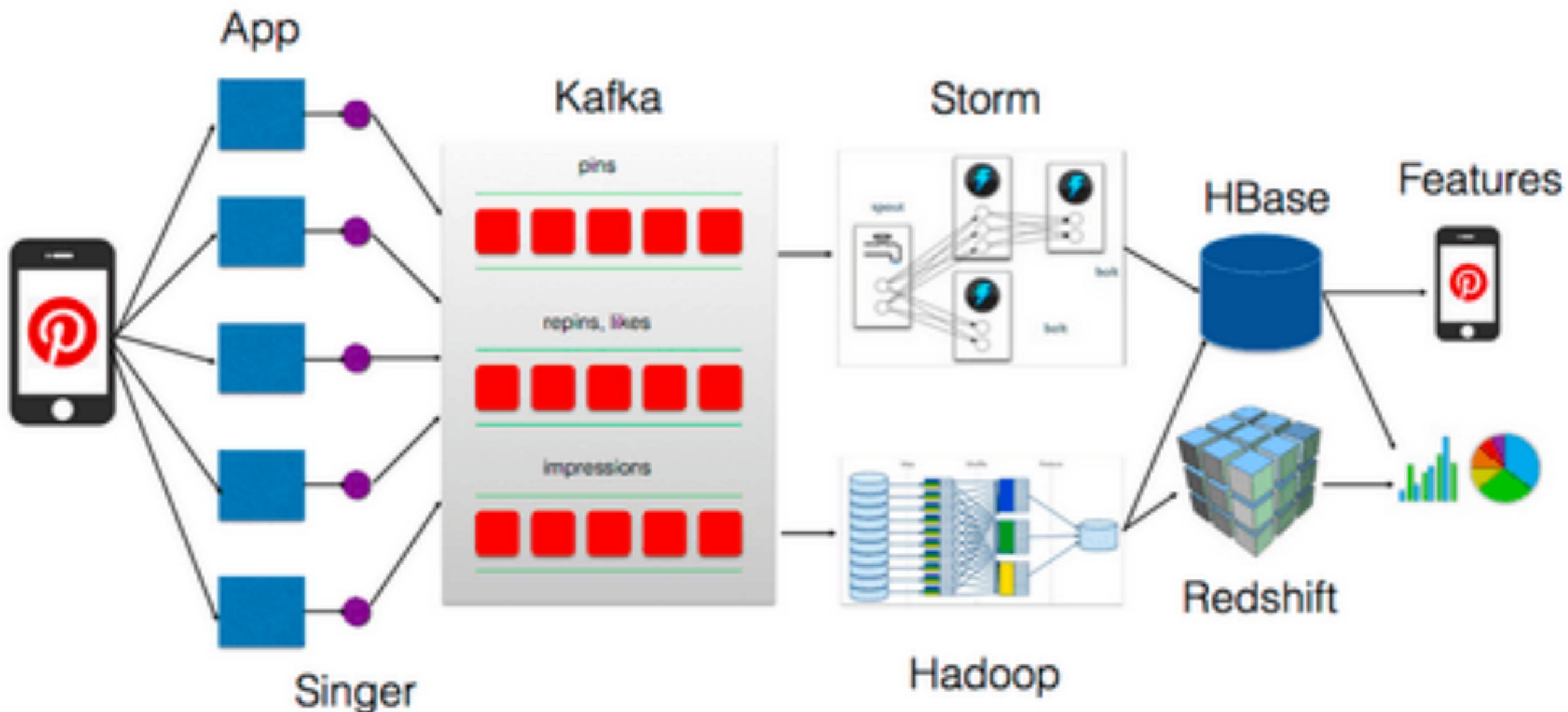
CC BY-NC-ND 4.0 International Licence
creativecommons.org/licenses/by-nc-nd/4.0/

scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

Open Source Data analytics Architecture

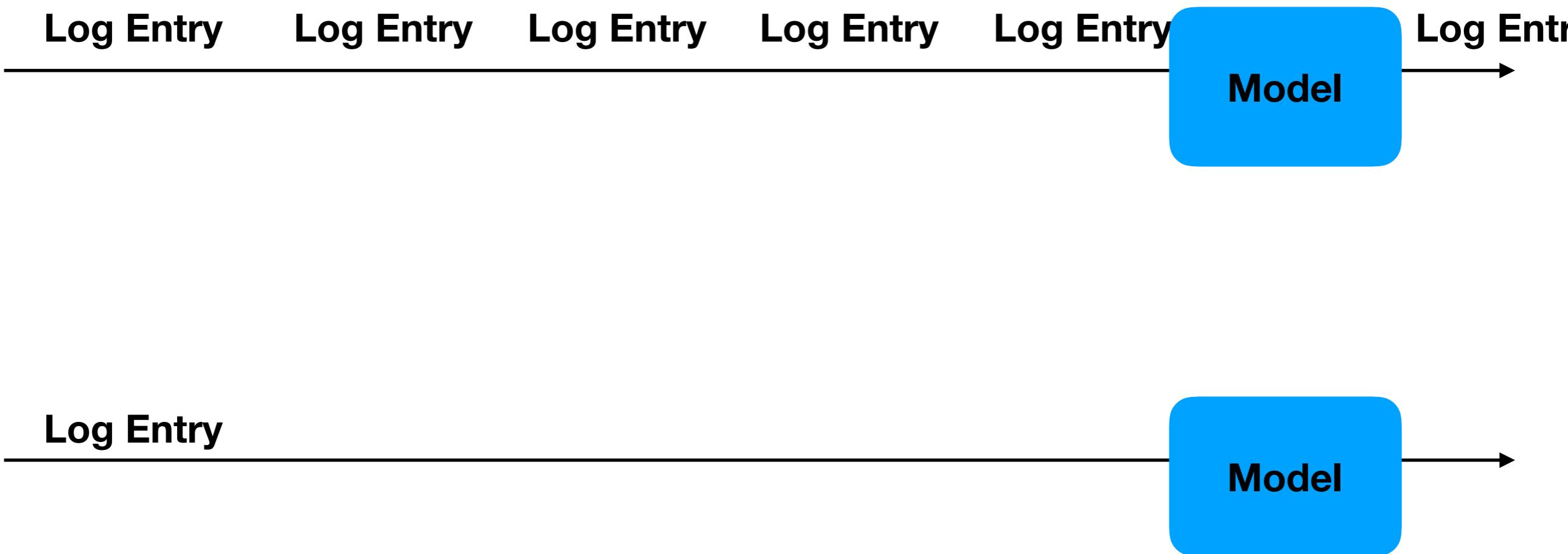


Data analytics Architecture adopted by Pinterest

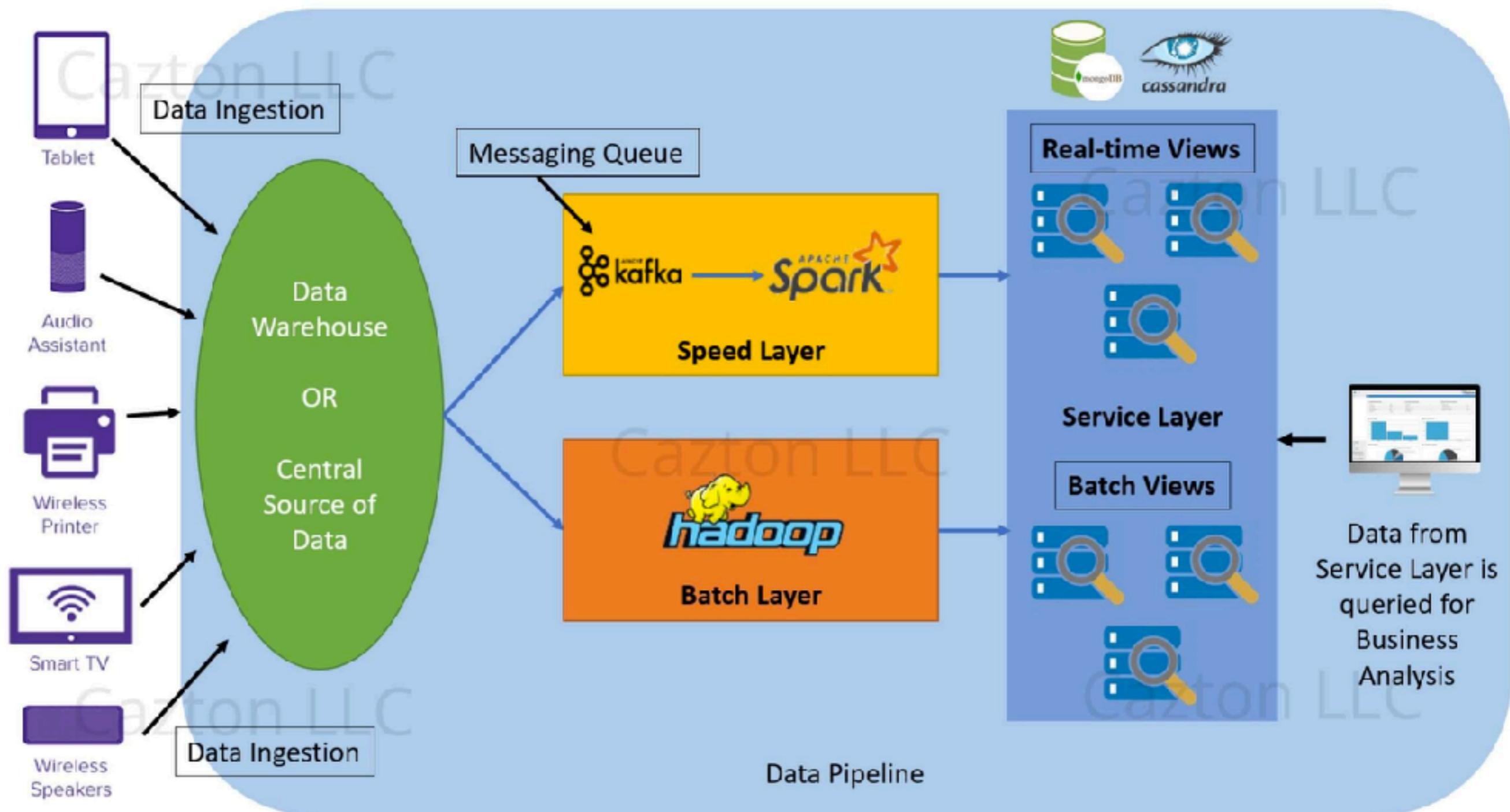


Data Architecture overview

Welldoc

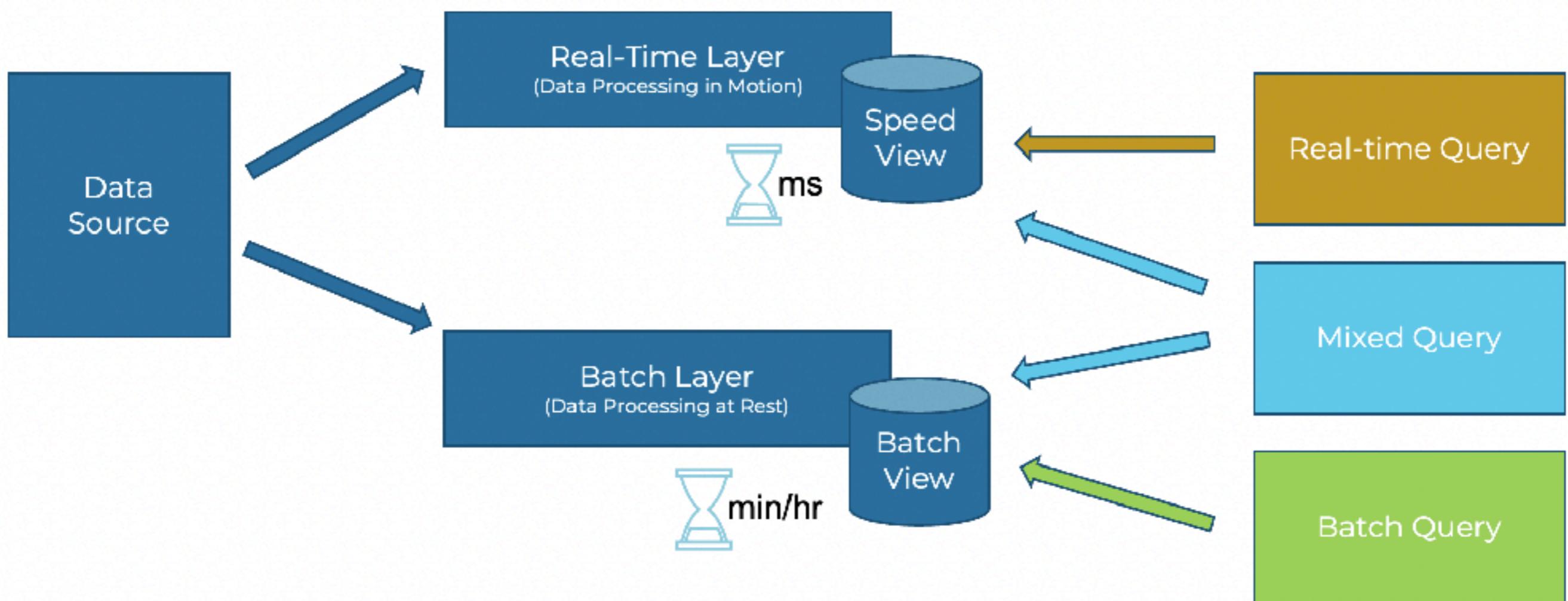


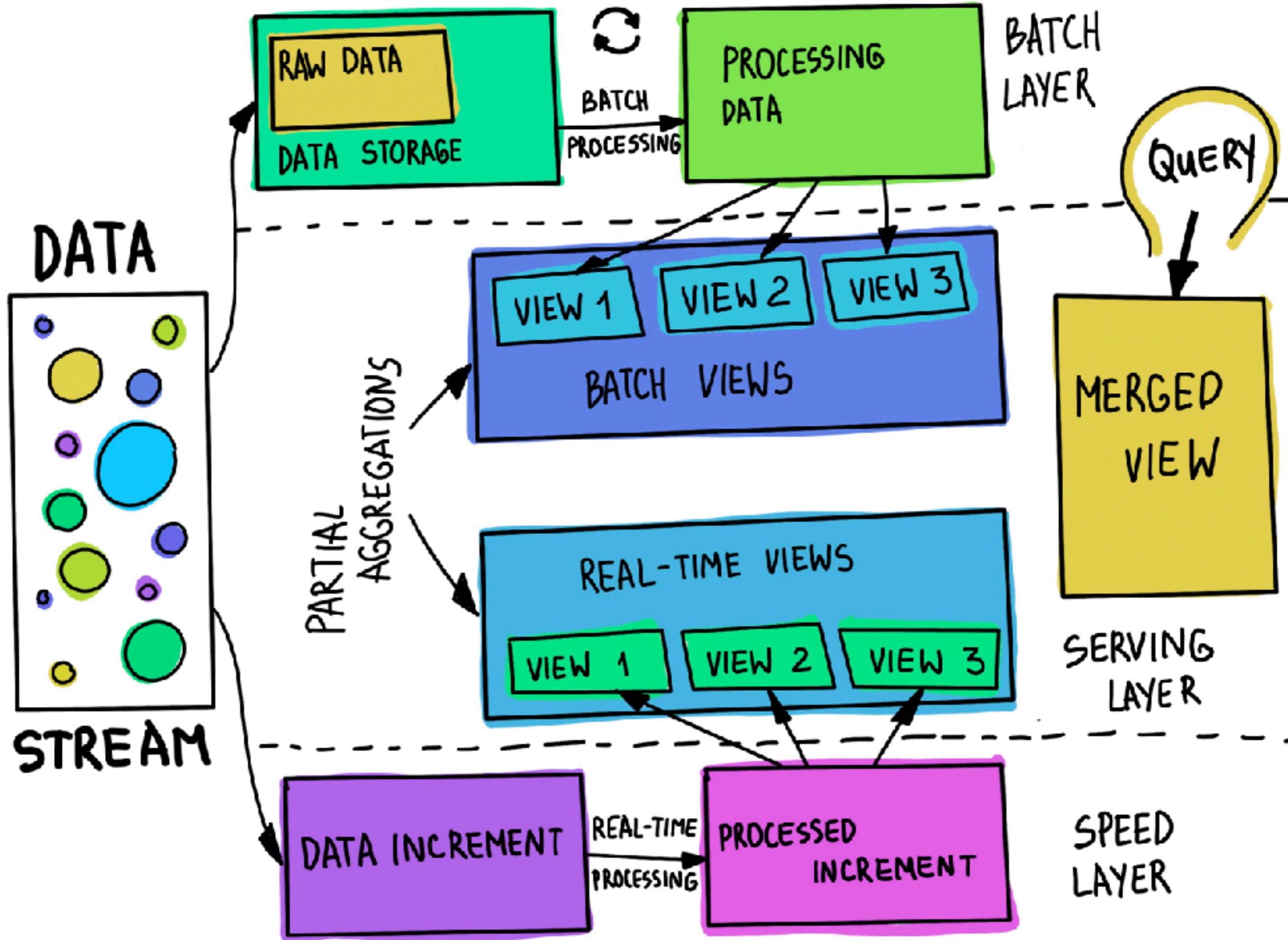
Lambda Architecture



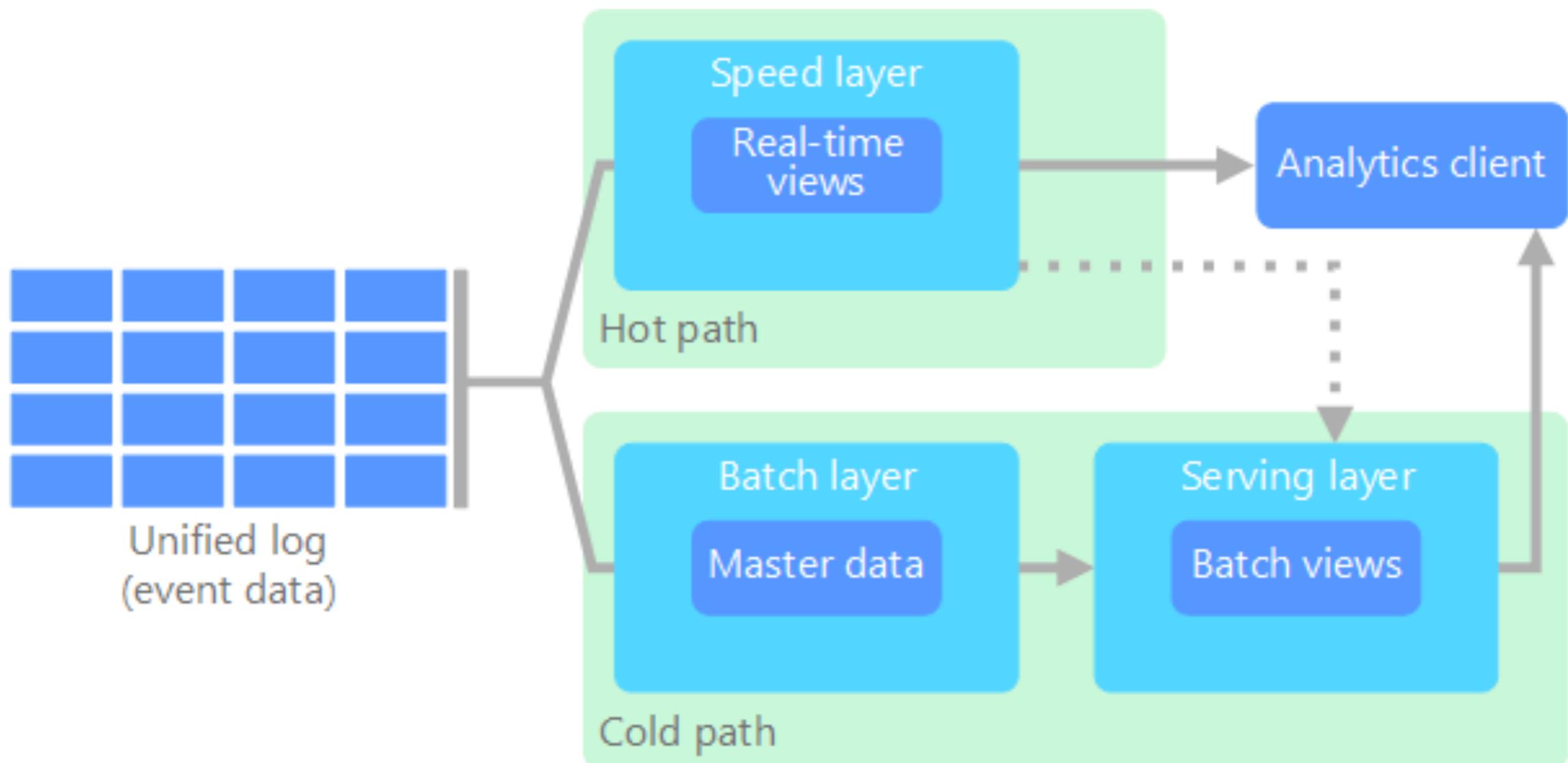
Lambda Architecture

Option 2: Separate serving layers



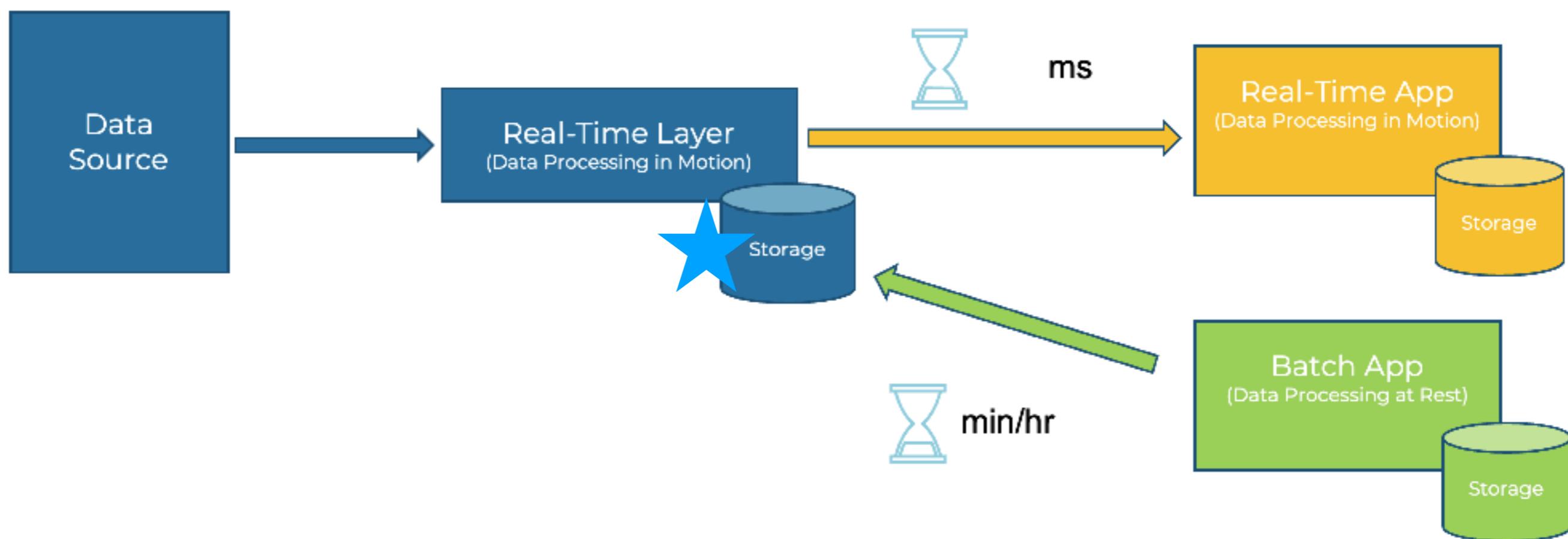


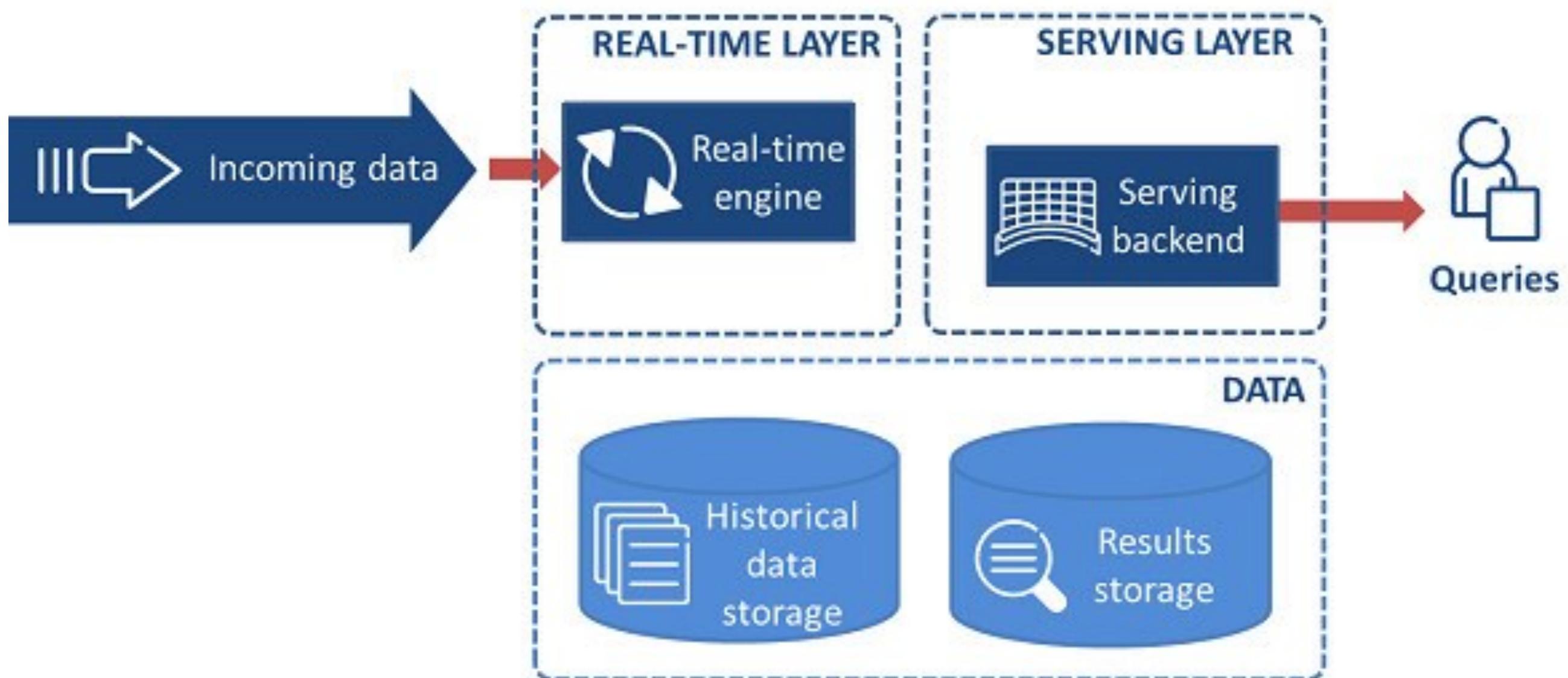
Lambda Architecture



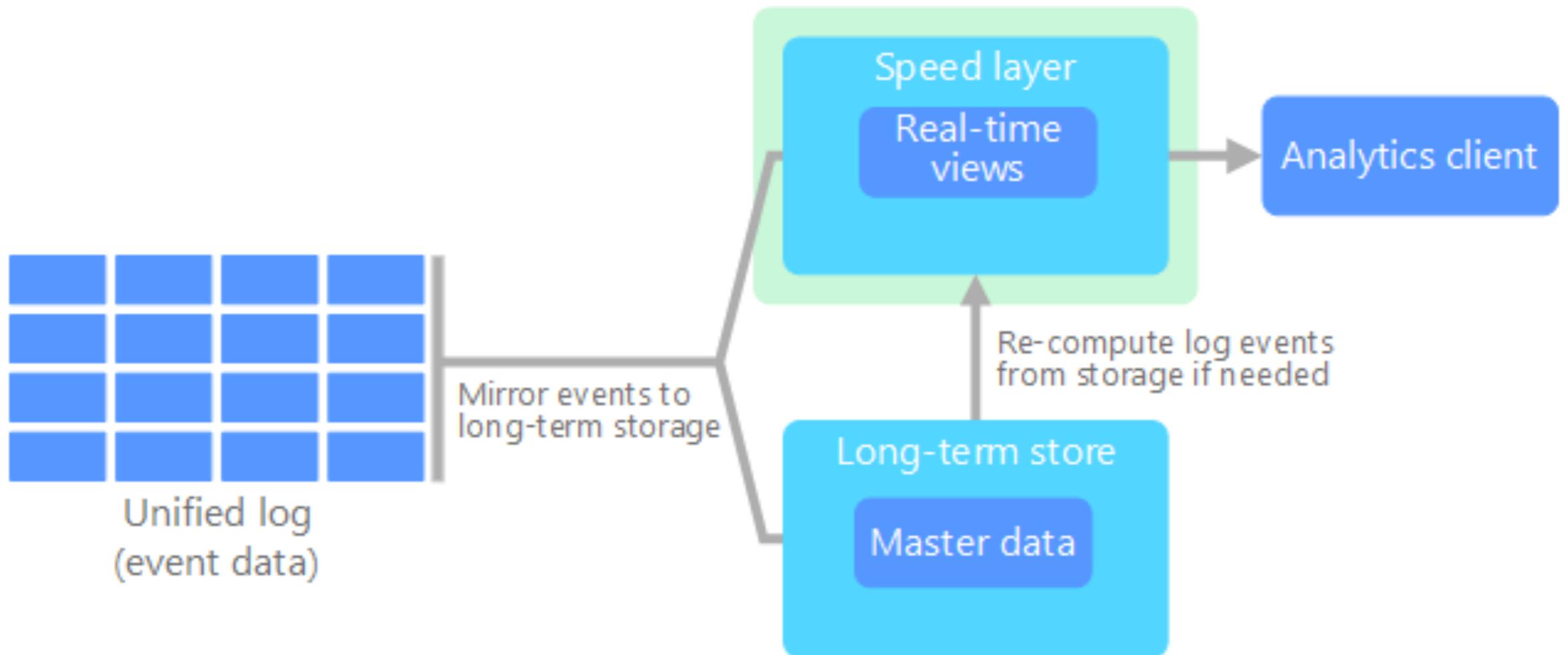
Kappa Architecture

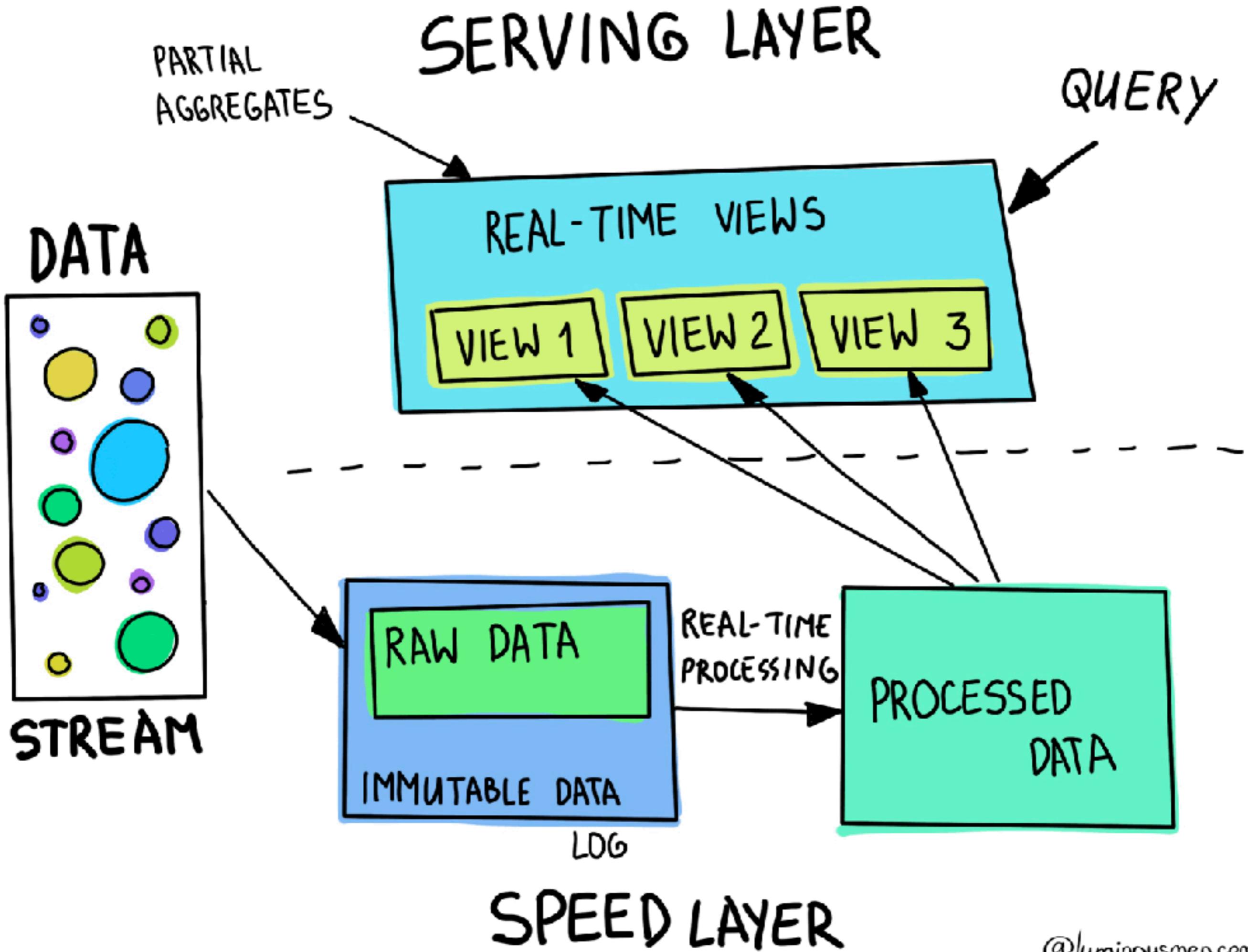
One pipeline for real-time and batch consumers





Kappa Architecture





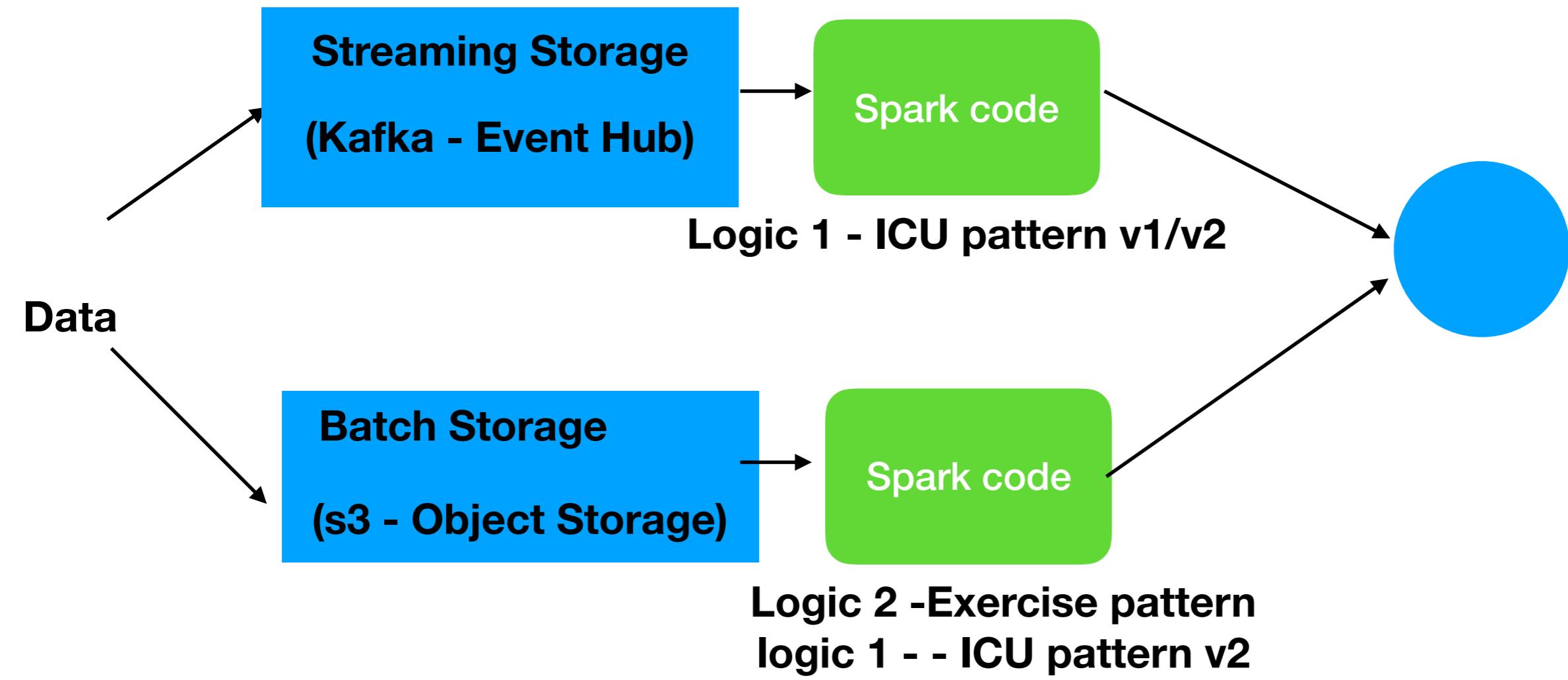
What Architecture to Choose?

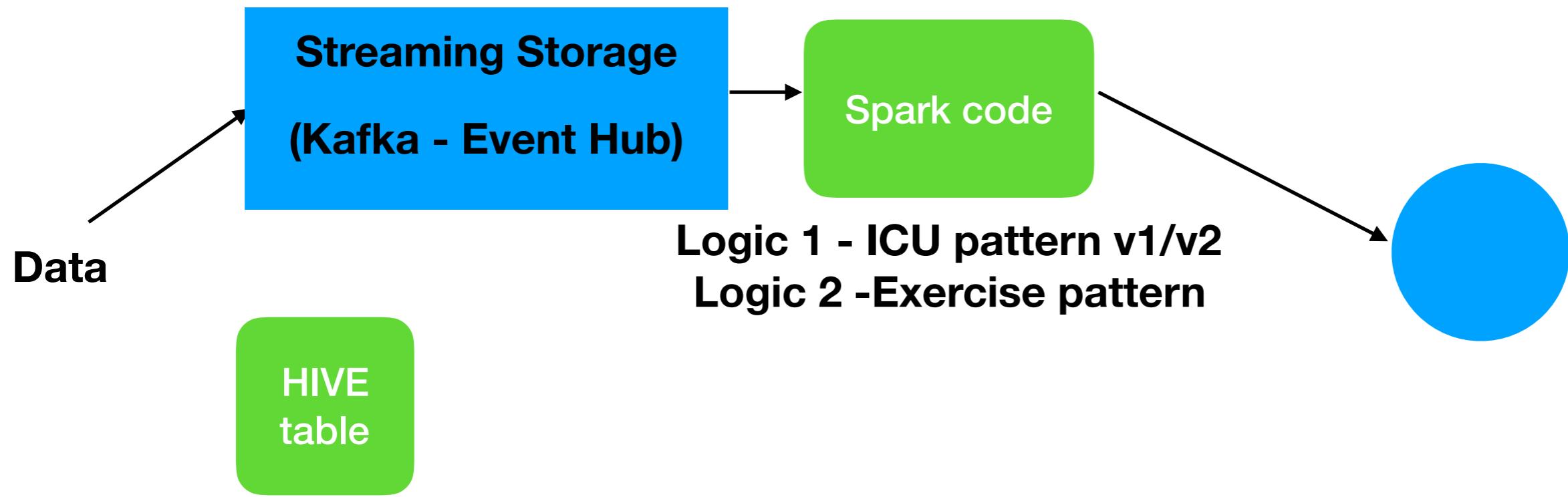


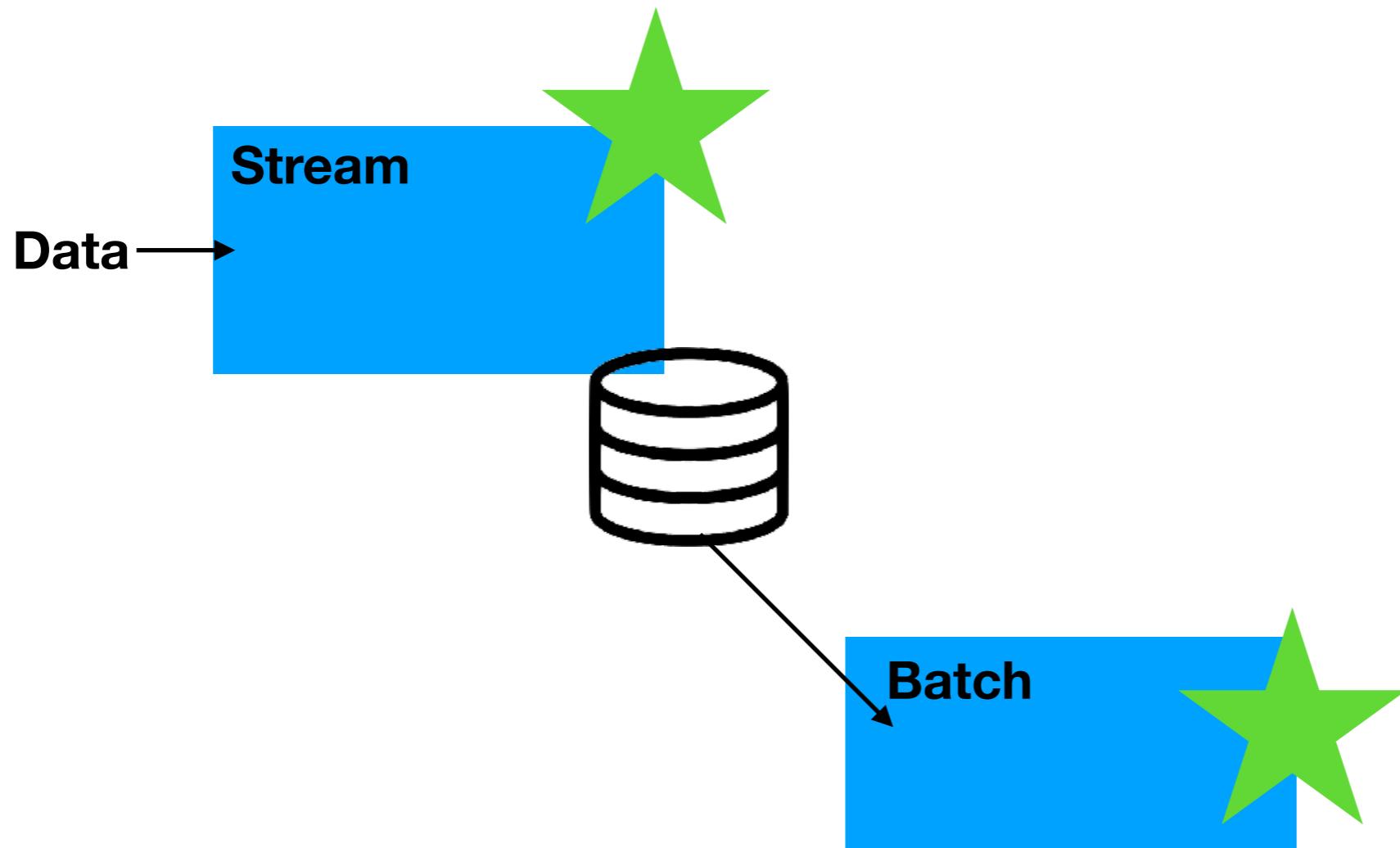
Kappa
Real-Time
Data in Motion

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**



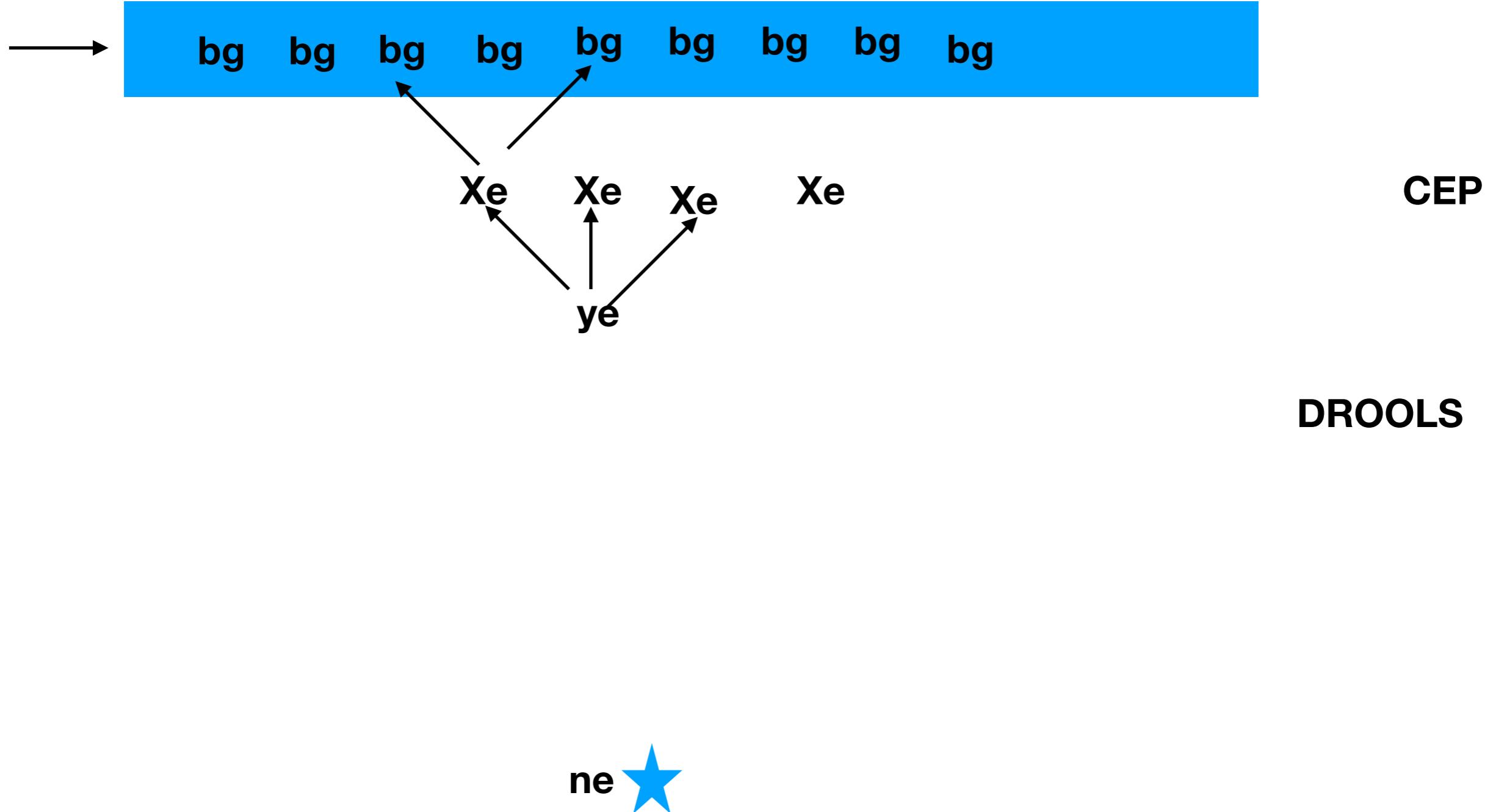
bg bg bg bg bg bg

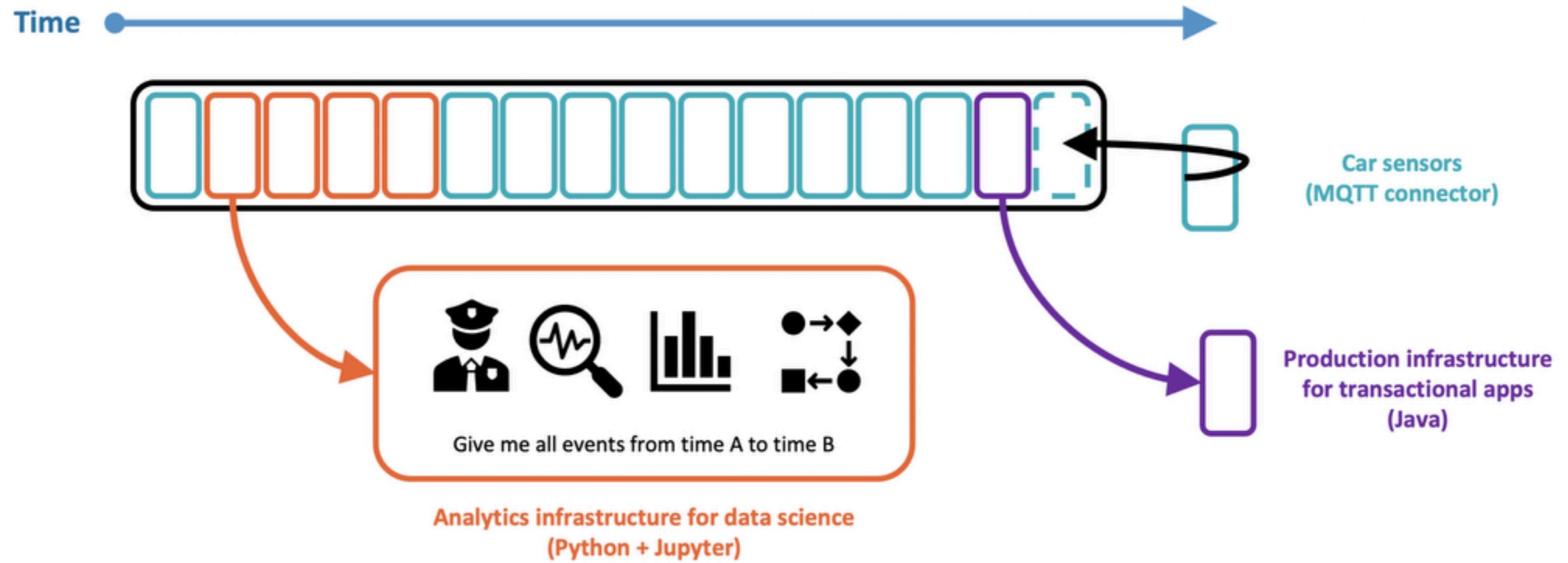
bg

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

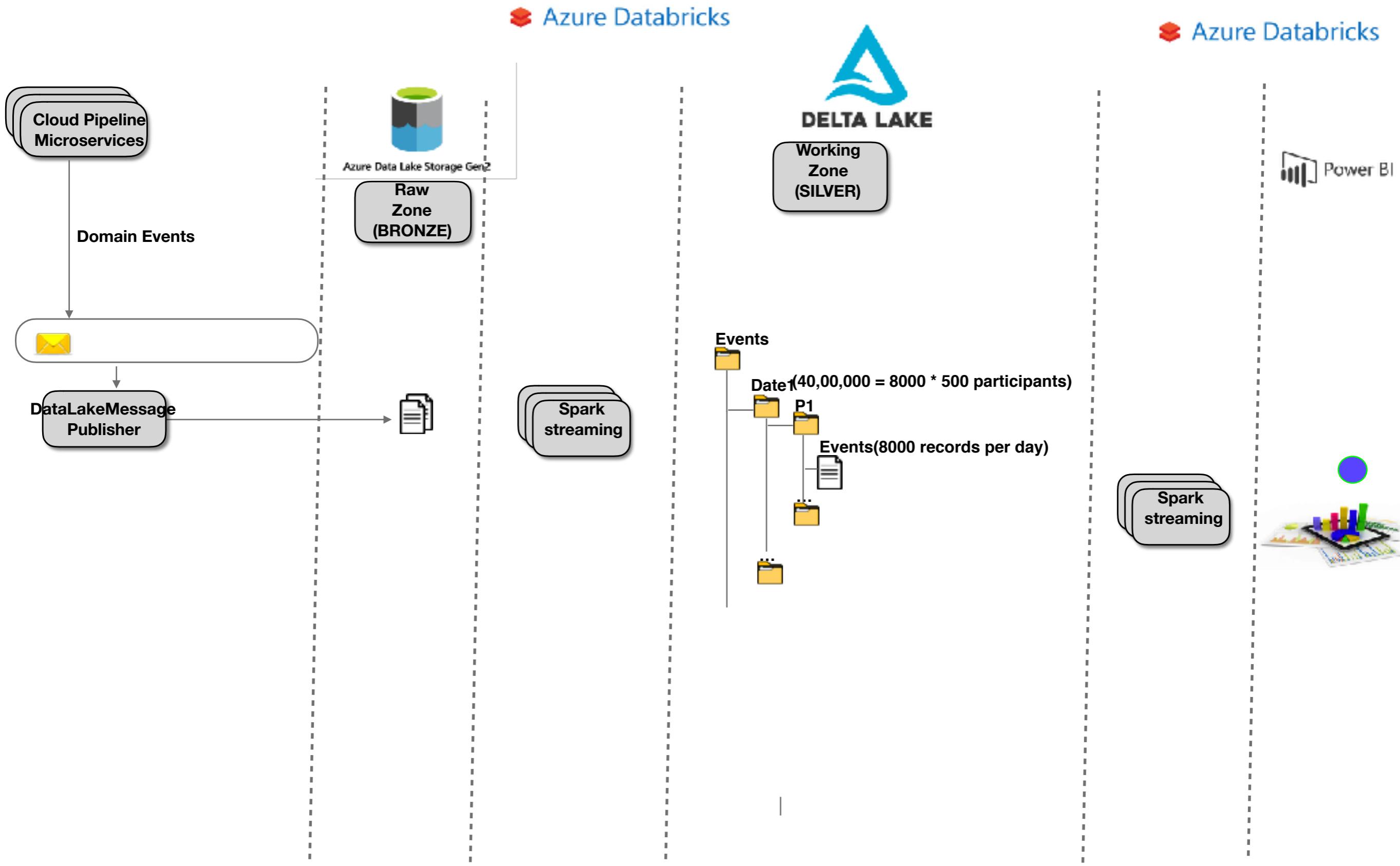
Analytics

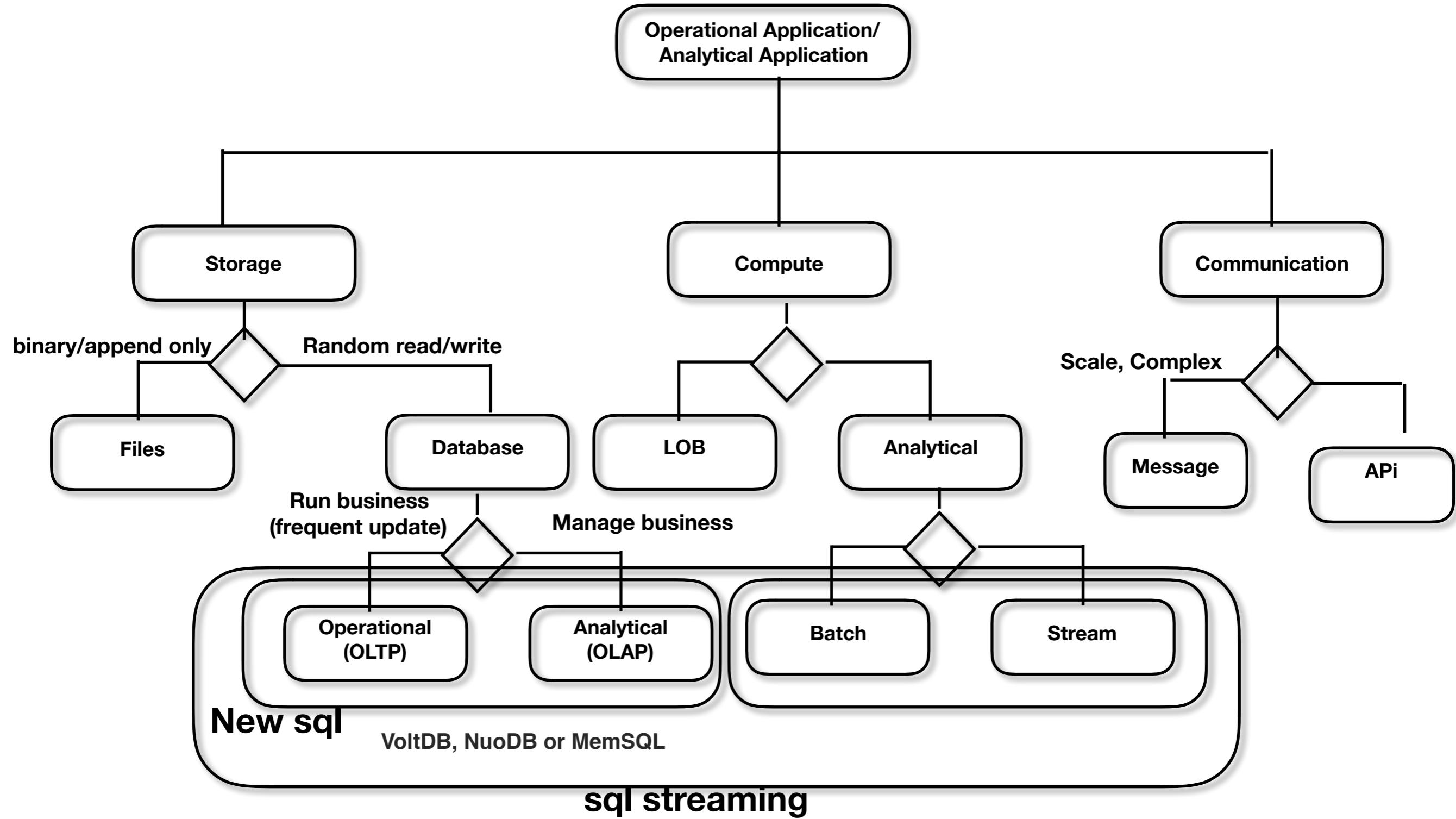
Stream

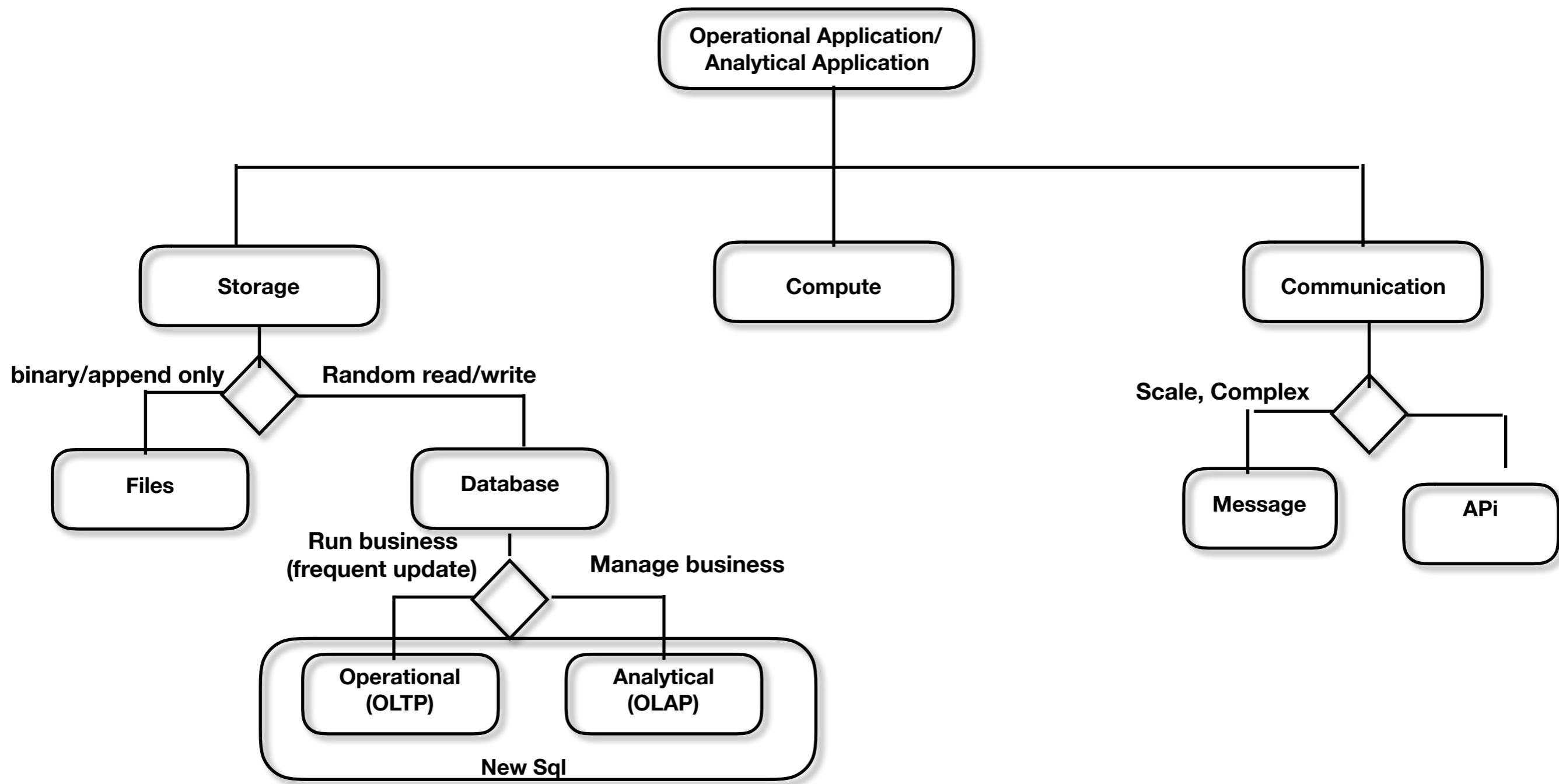
Batch

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

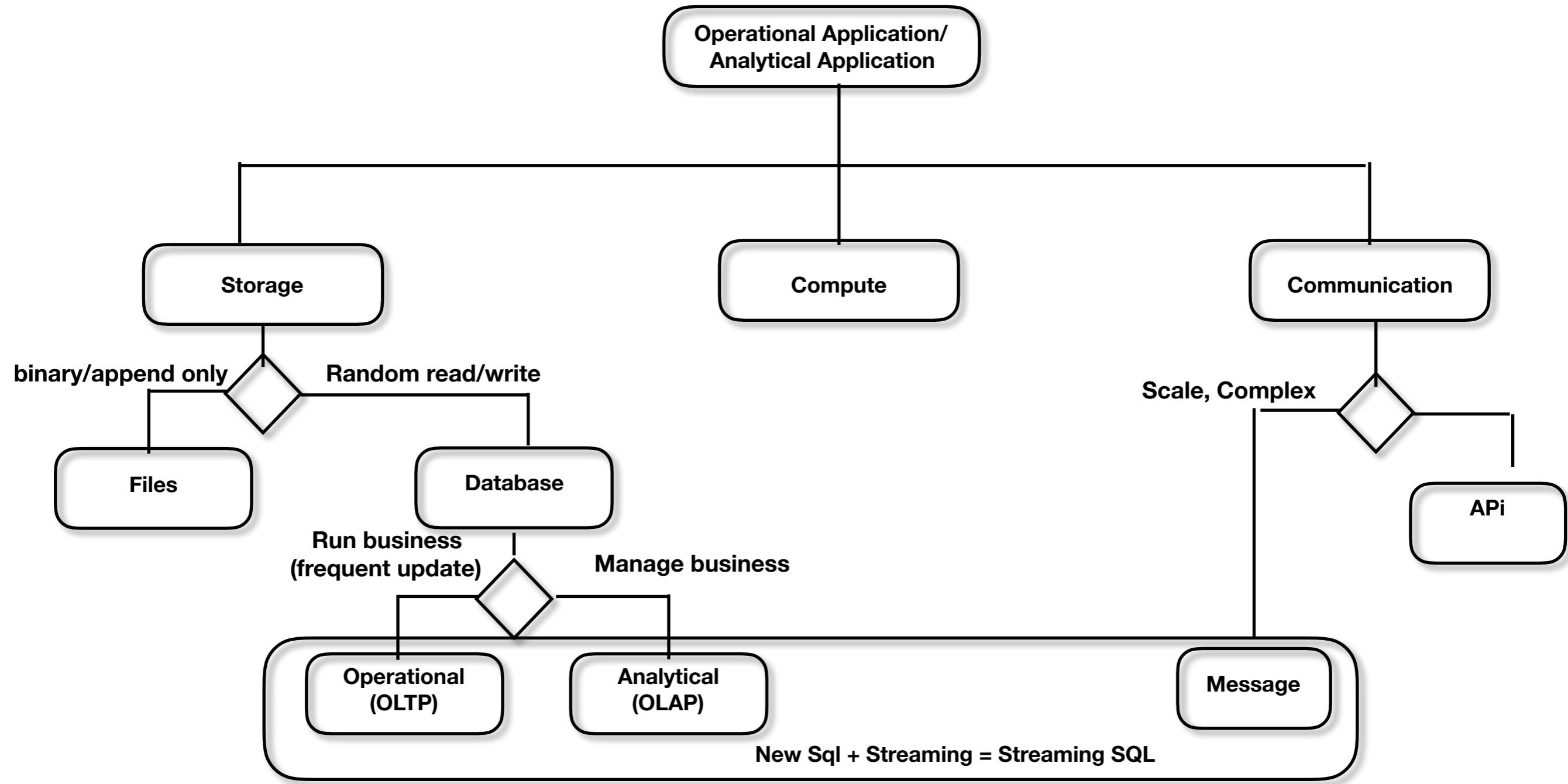
- Standalone: Simple cluster-manager, limited in features, incorporated with Spark.
- Apache Mesos.
- Hadoop YARN
- Kubernetes

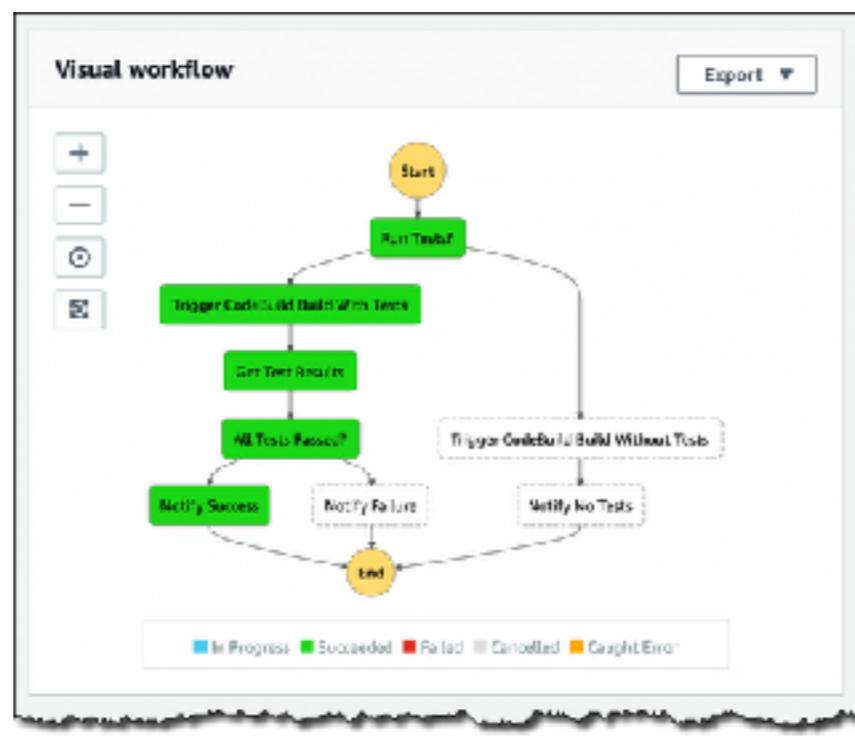






VoltDB, NuoDB or MemSQL





AWS Step

Azure Logic App

Biztalk

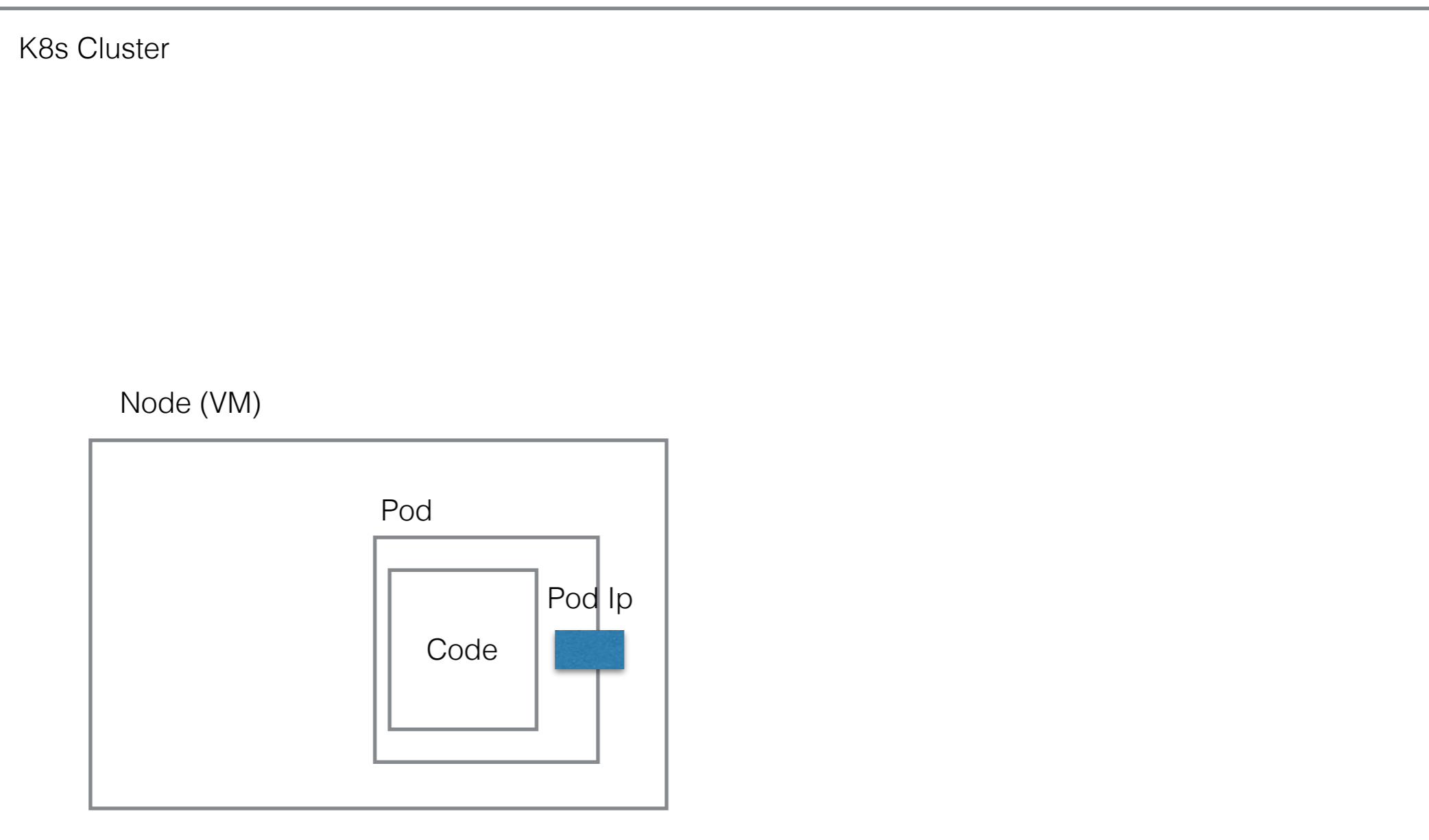
Webmethods

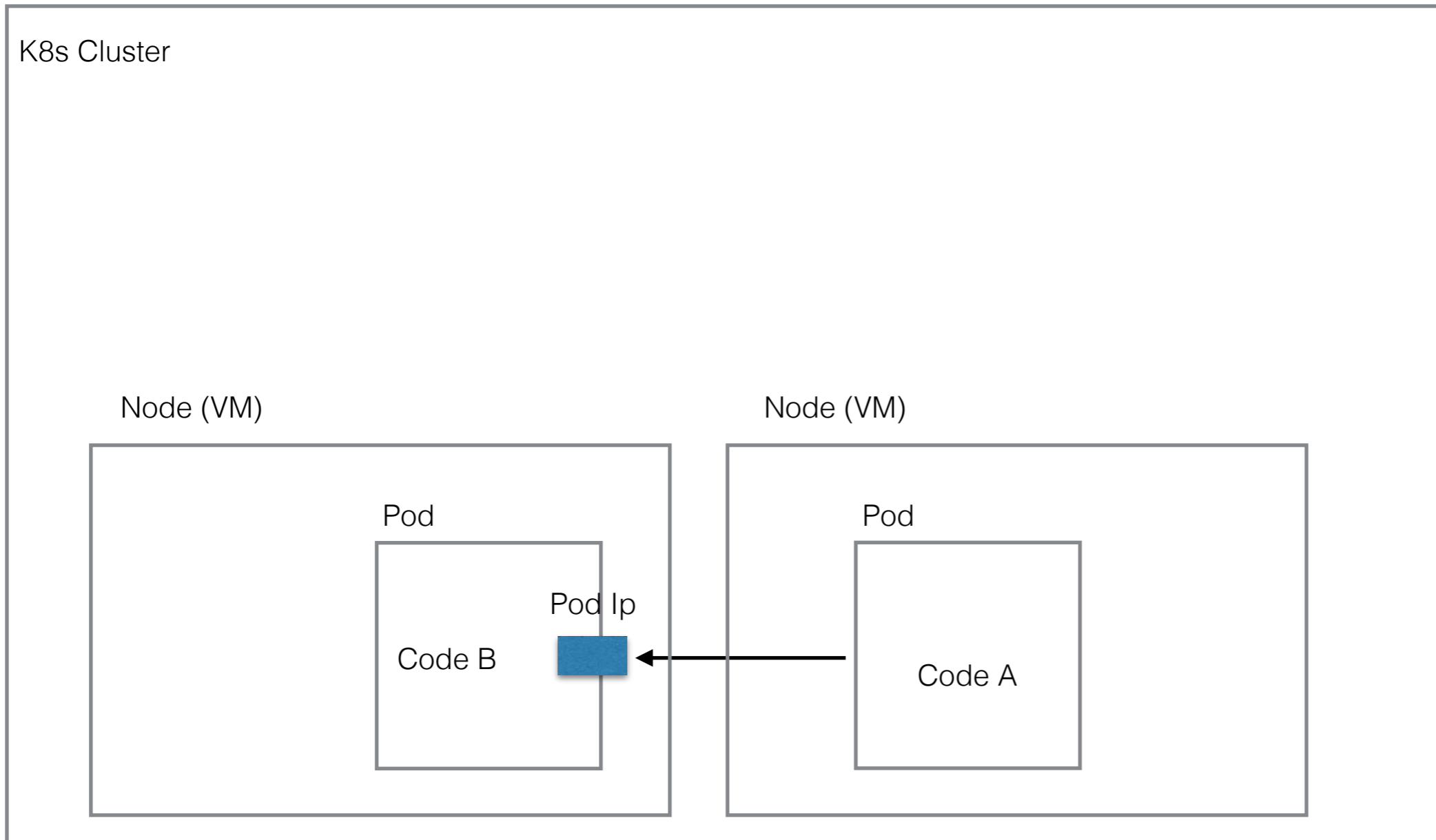
Mule

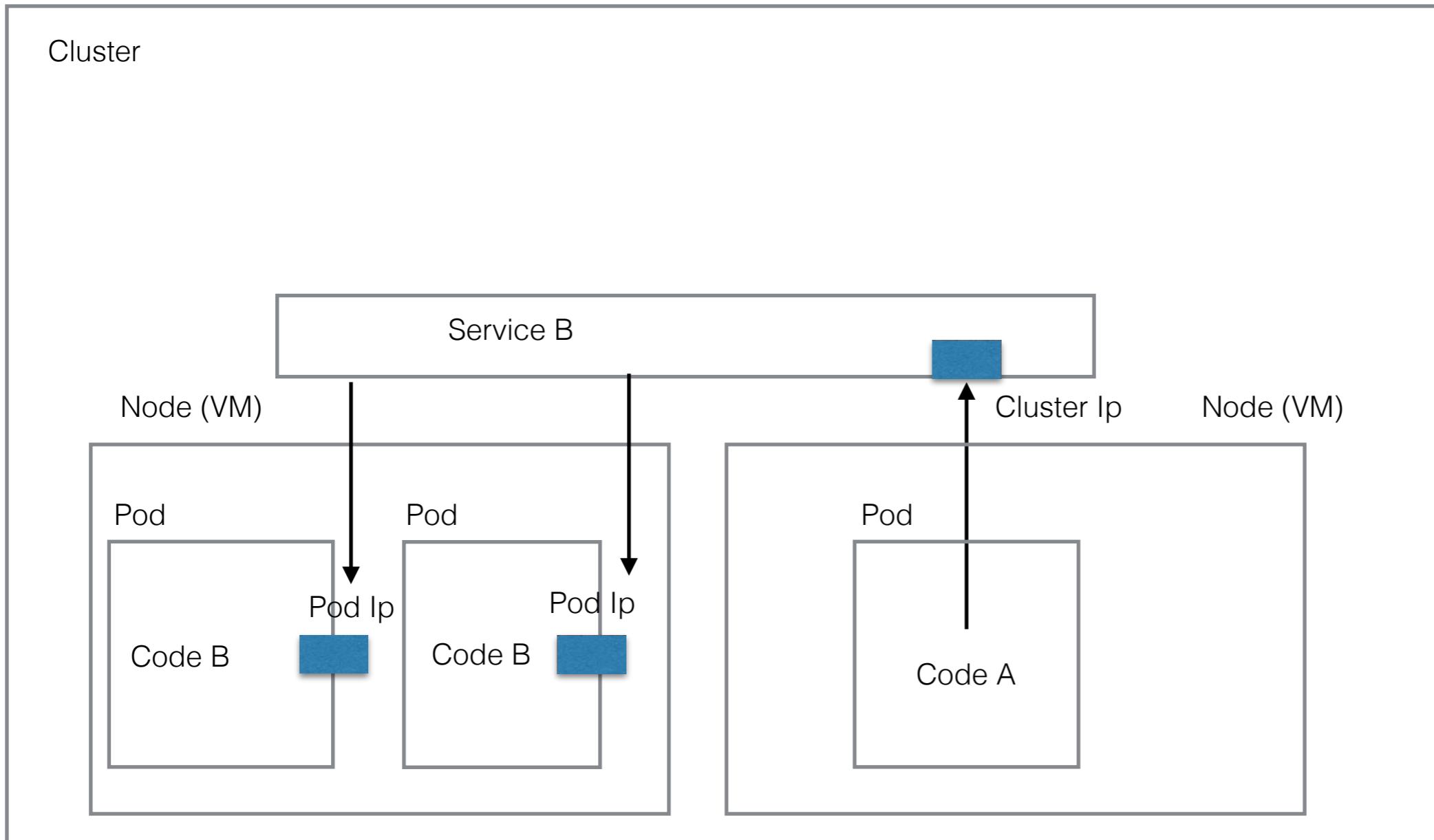
ServiceNow

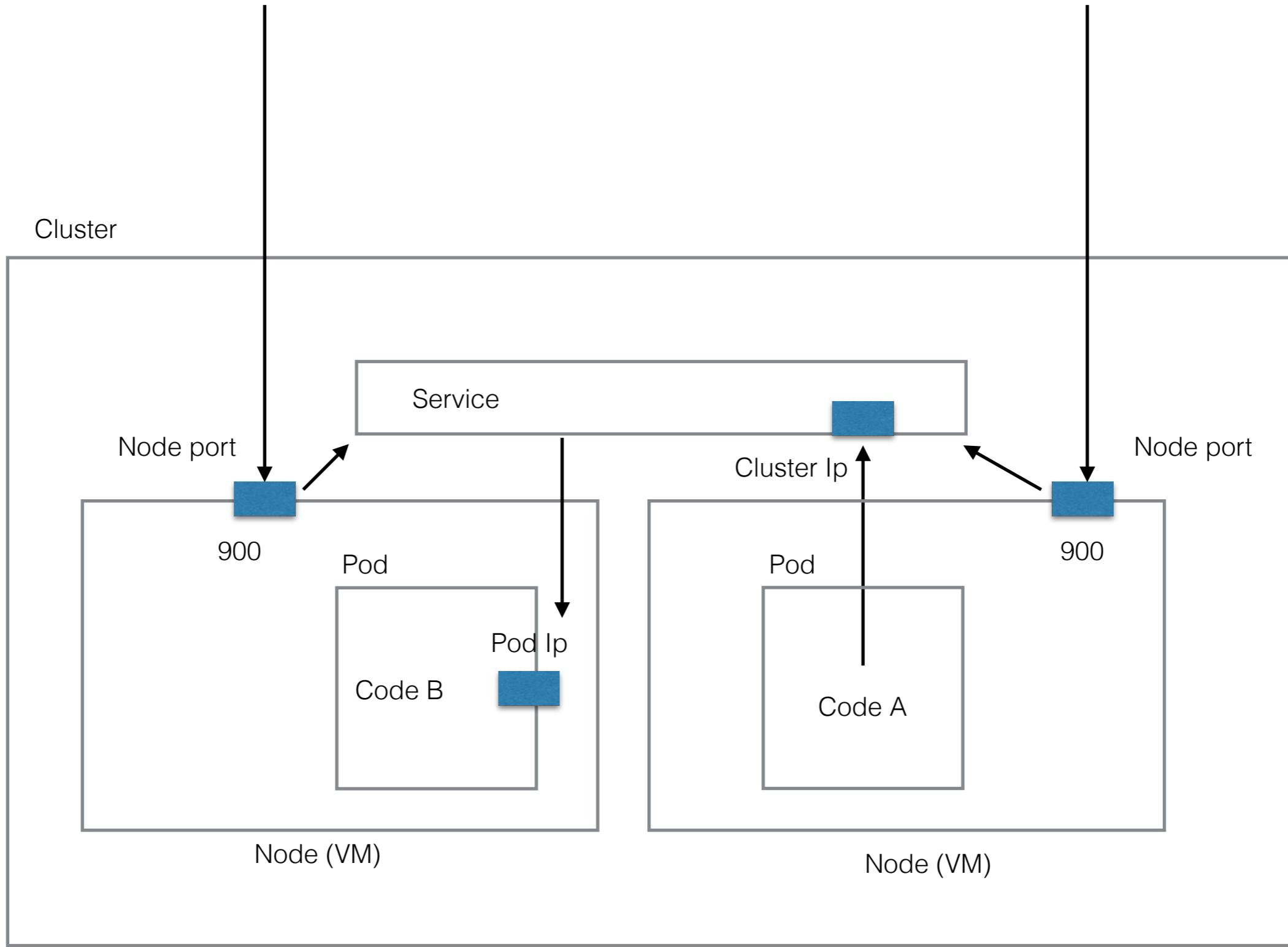
#

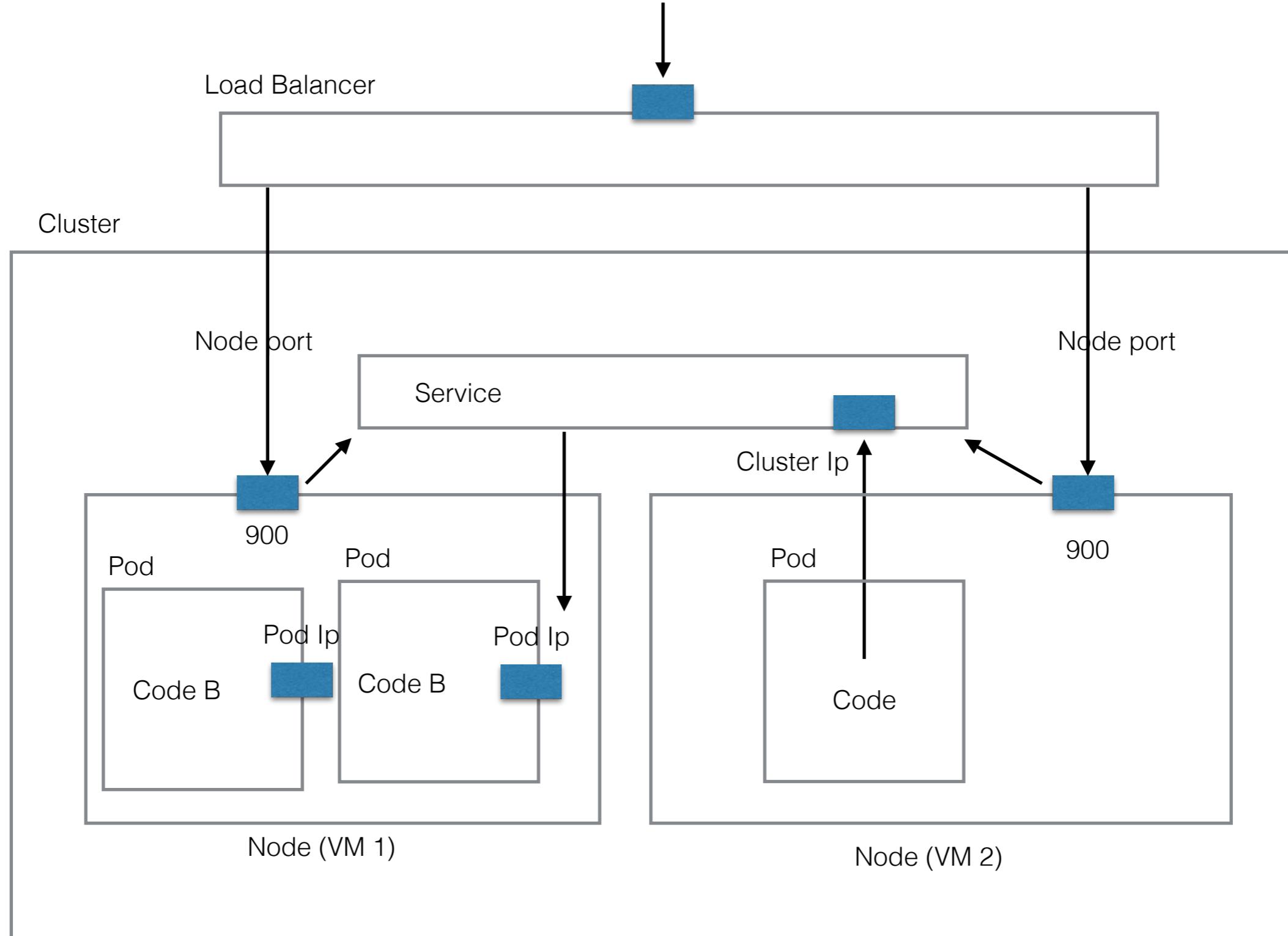
Deployment View

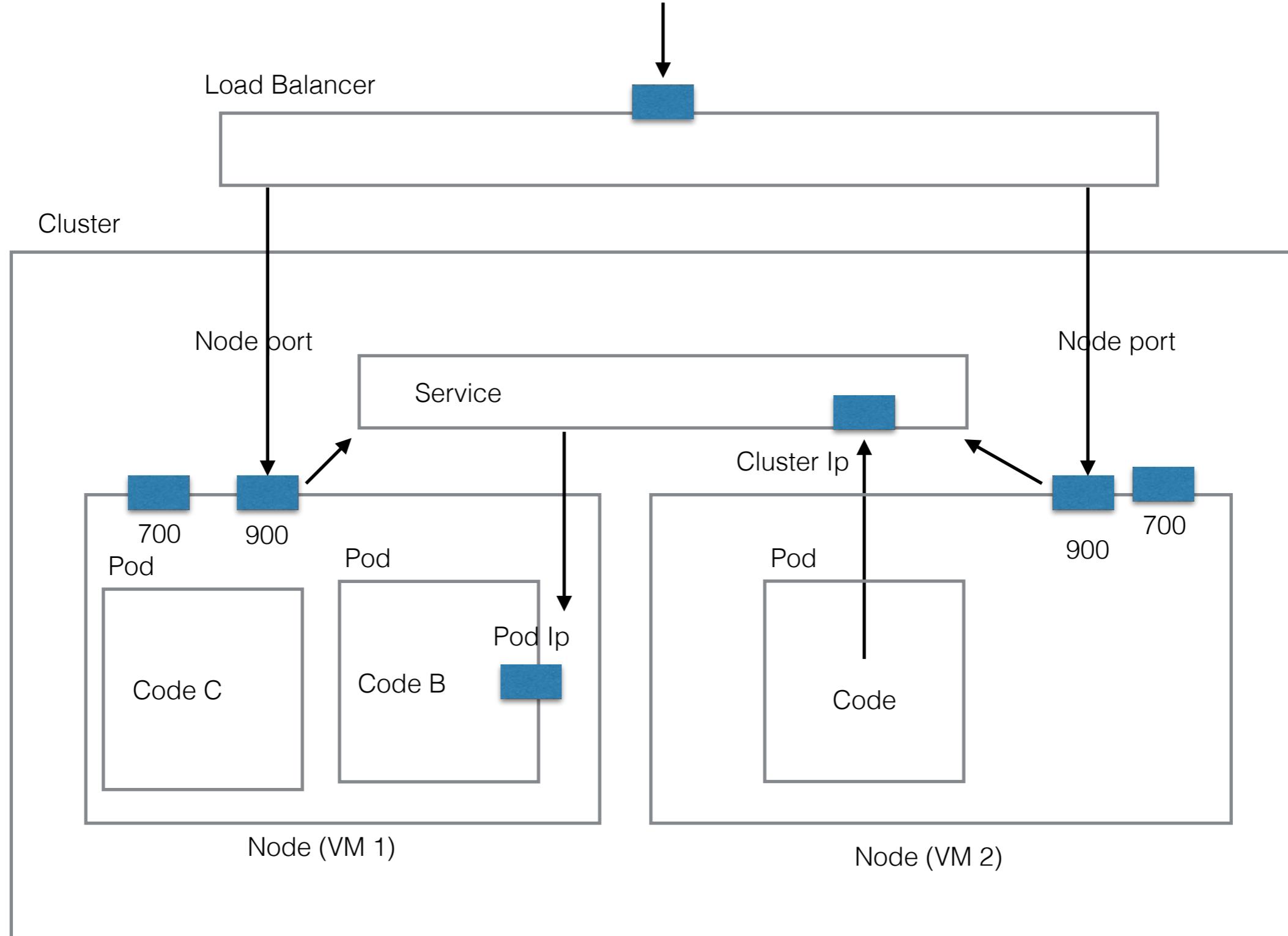




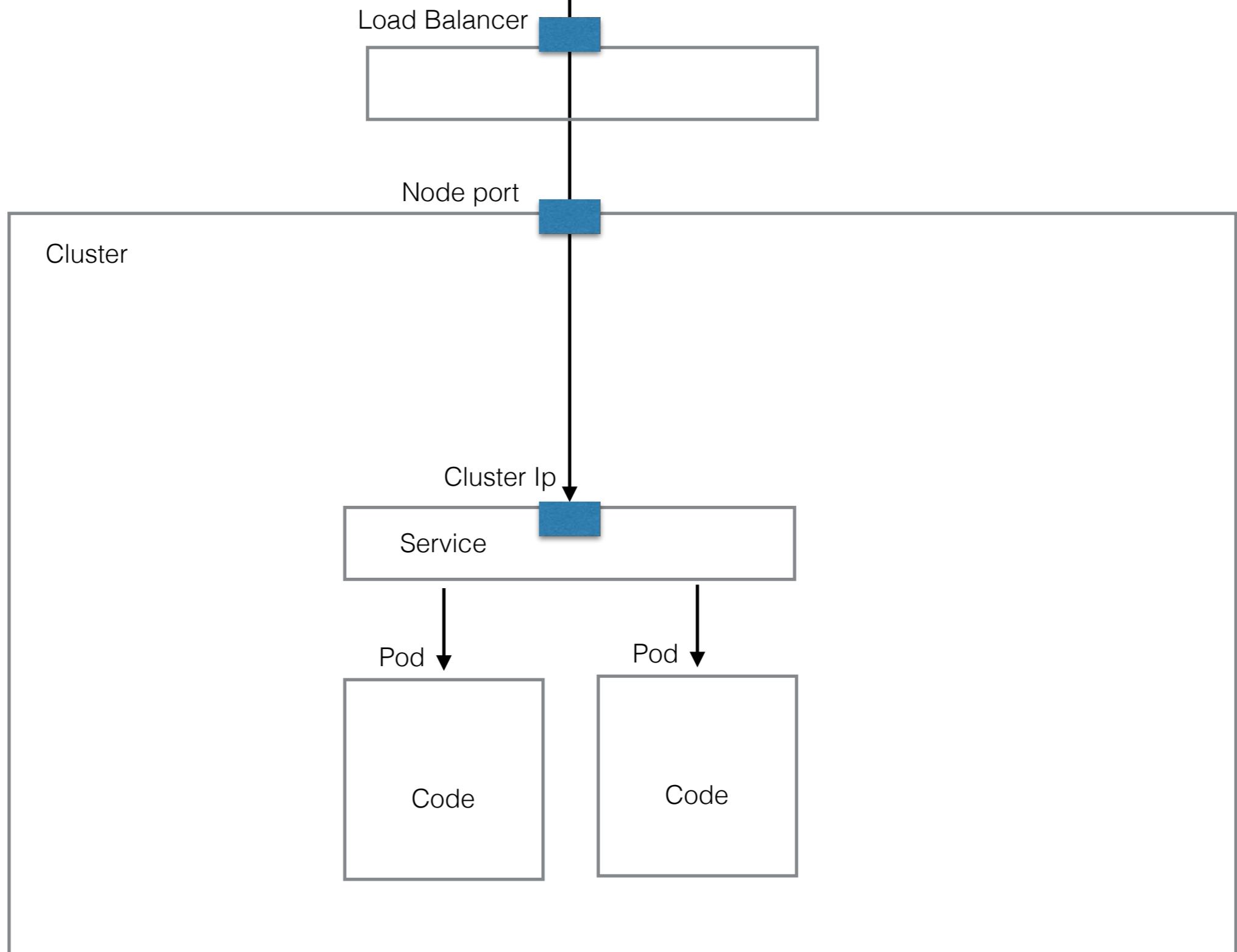




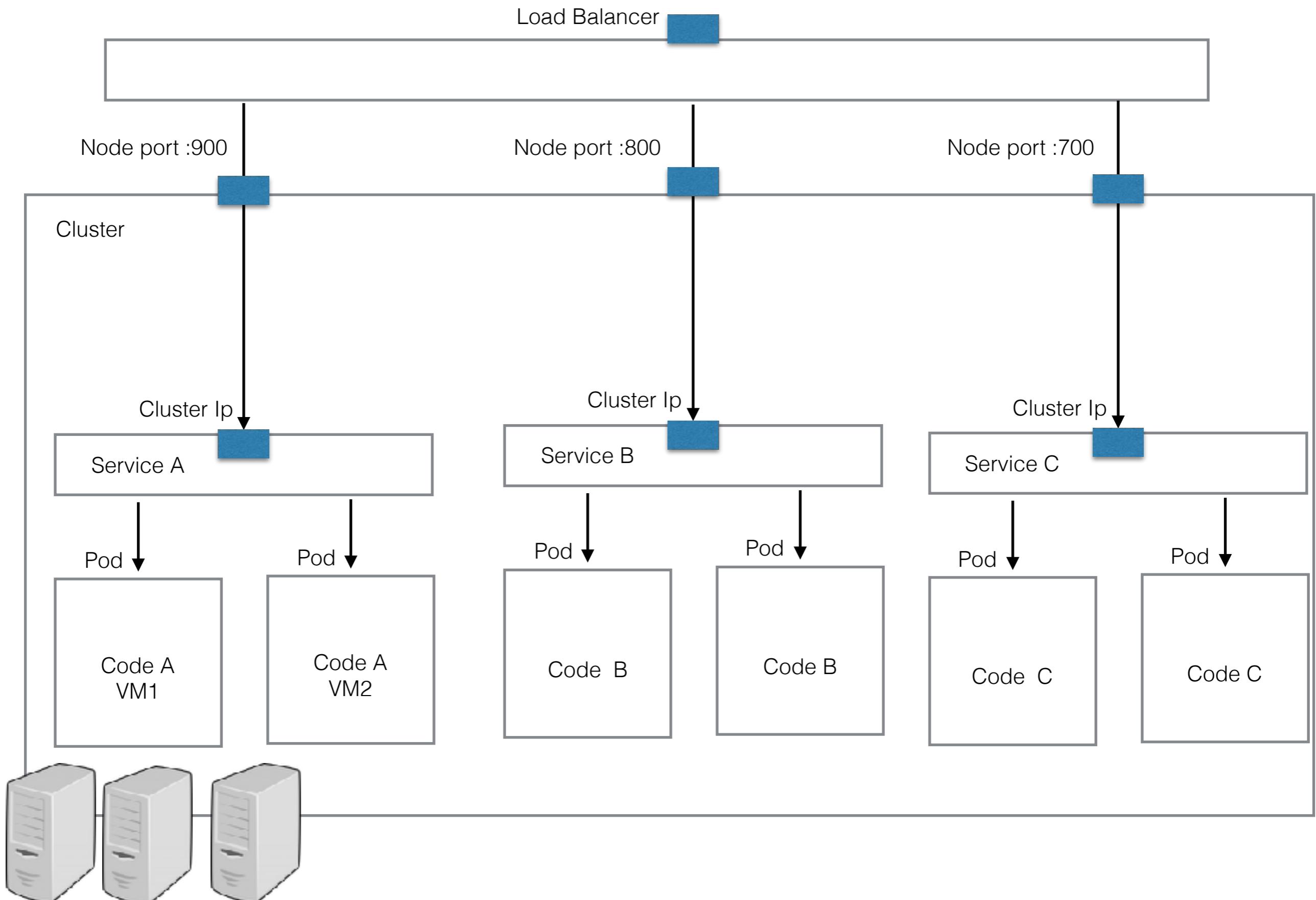




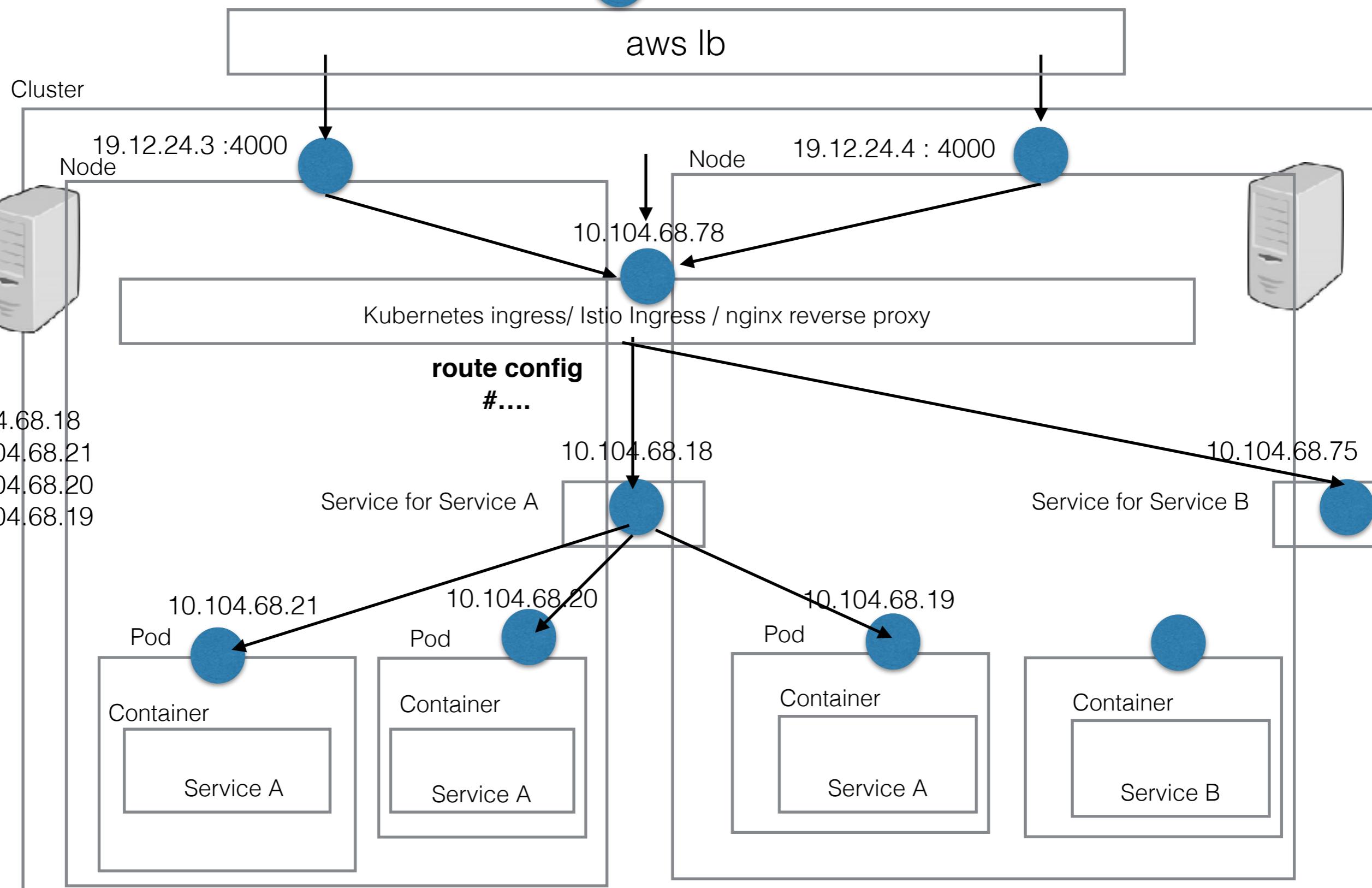
`curl http://10.97.245.99:8080/date`

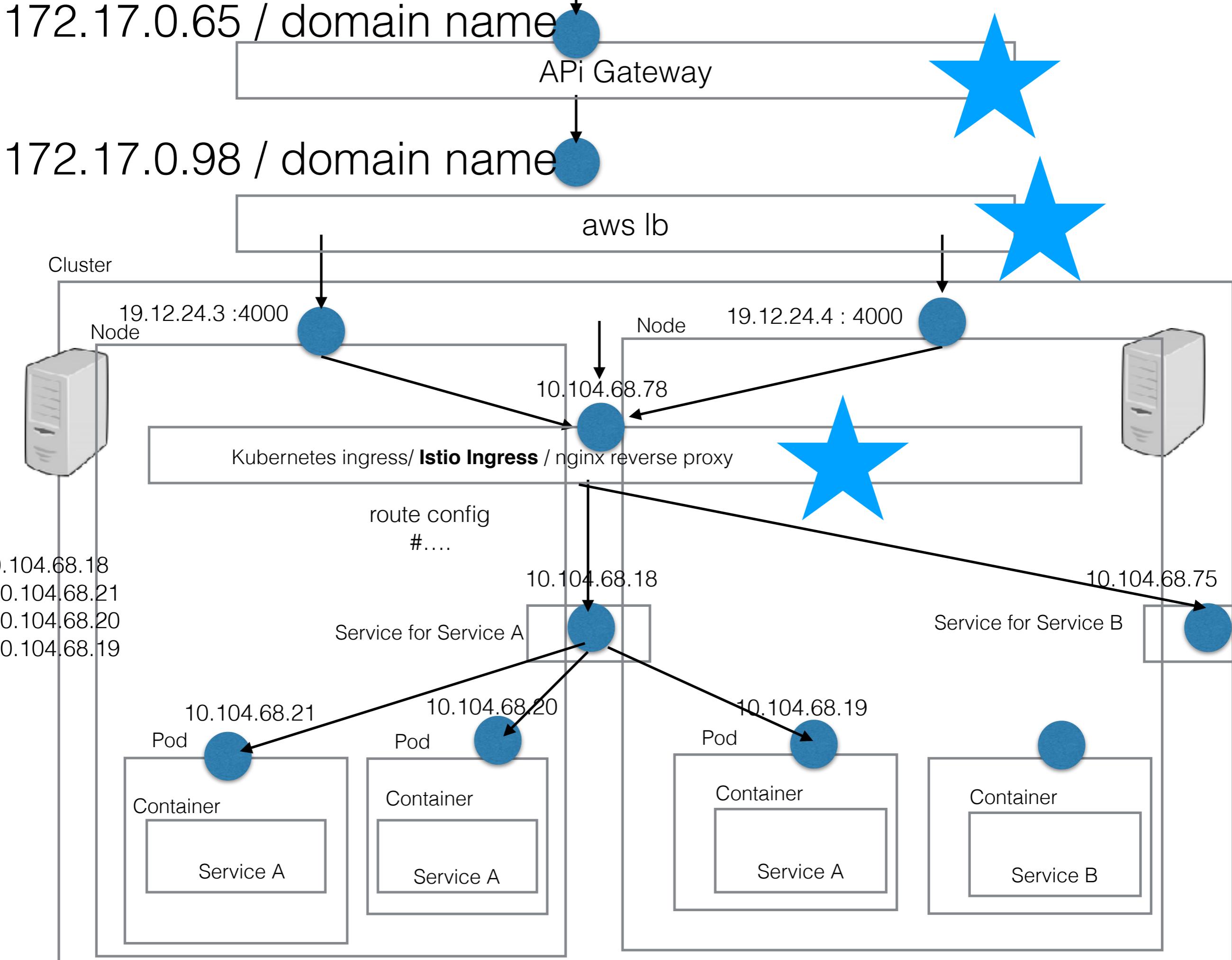


`curl http://10.97.245.99:8080/date`



172.17.0.98/sadd/hgfhgf/hghg





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

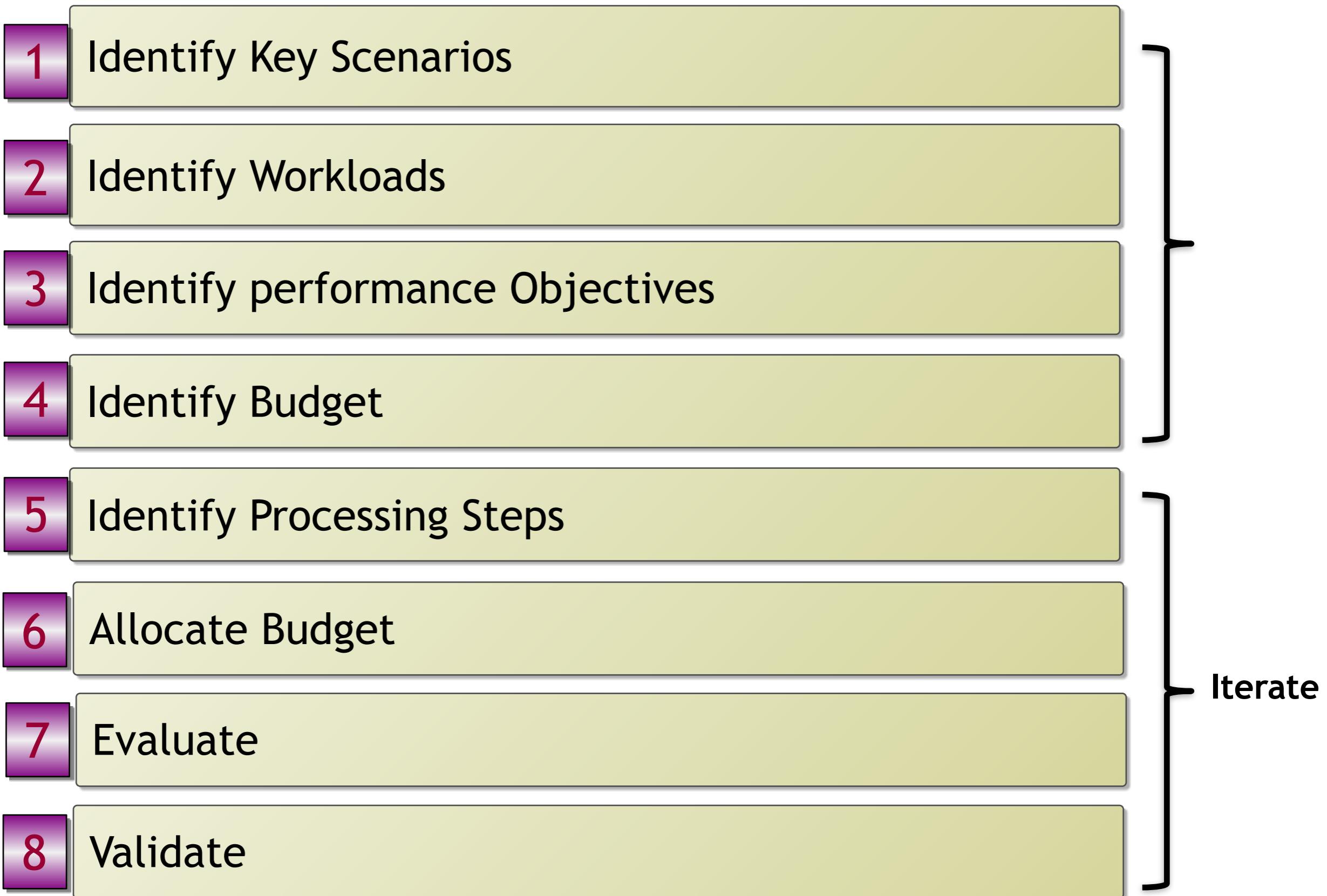
Testing
Load Testing
Stress Testing
Capacity Testing

Tuning
Network
System
Platform
Application

Roles
(Architects, Developers, Testers, Administrators)

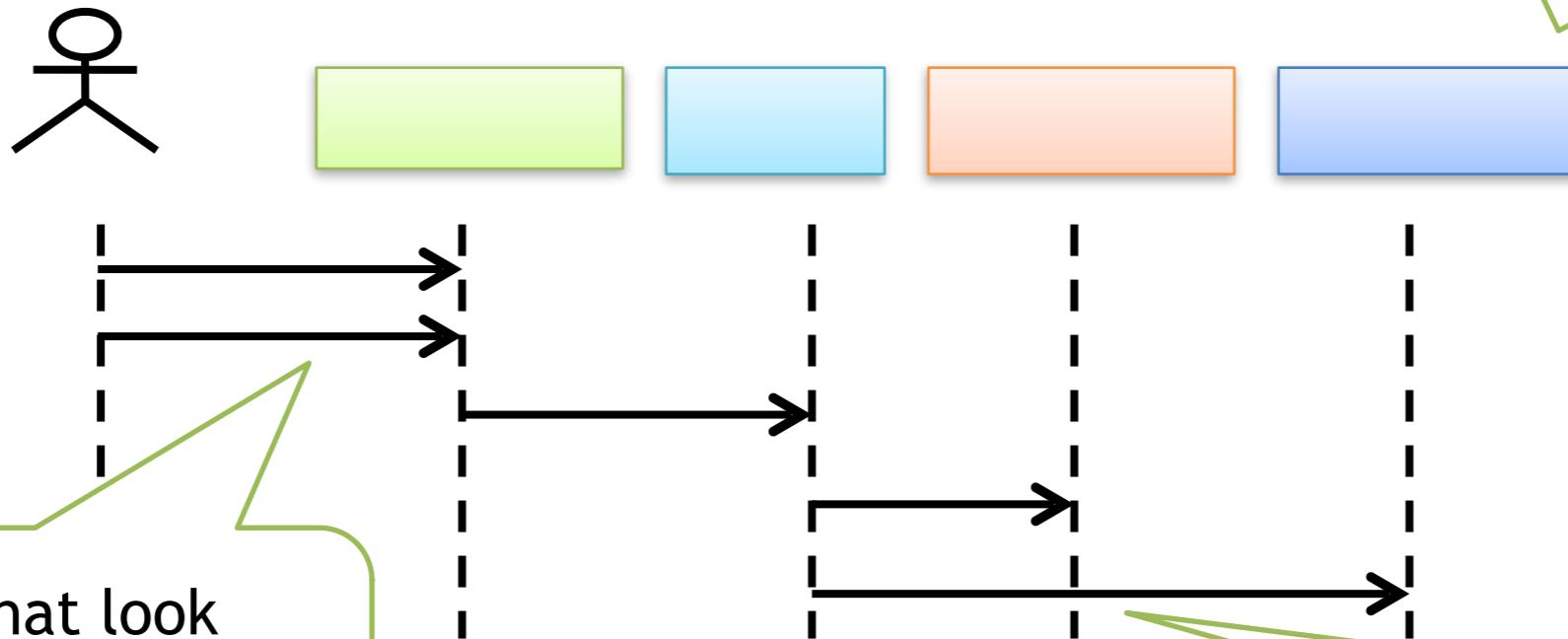
Life Cycle
(Requirements, Design, Develop, Test, Deploy, Maintain)

Performance Modeling



Allocate Budget

Know the cost of your materials.



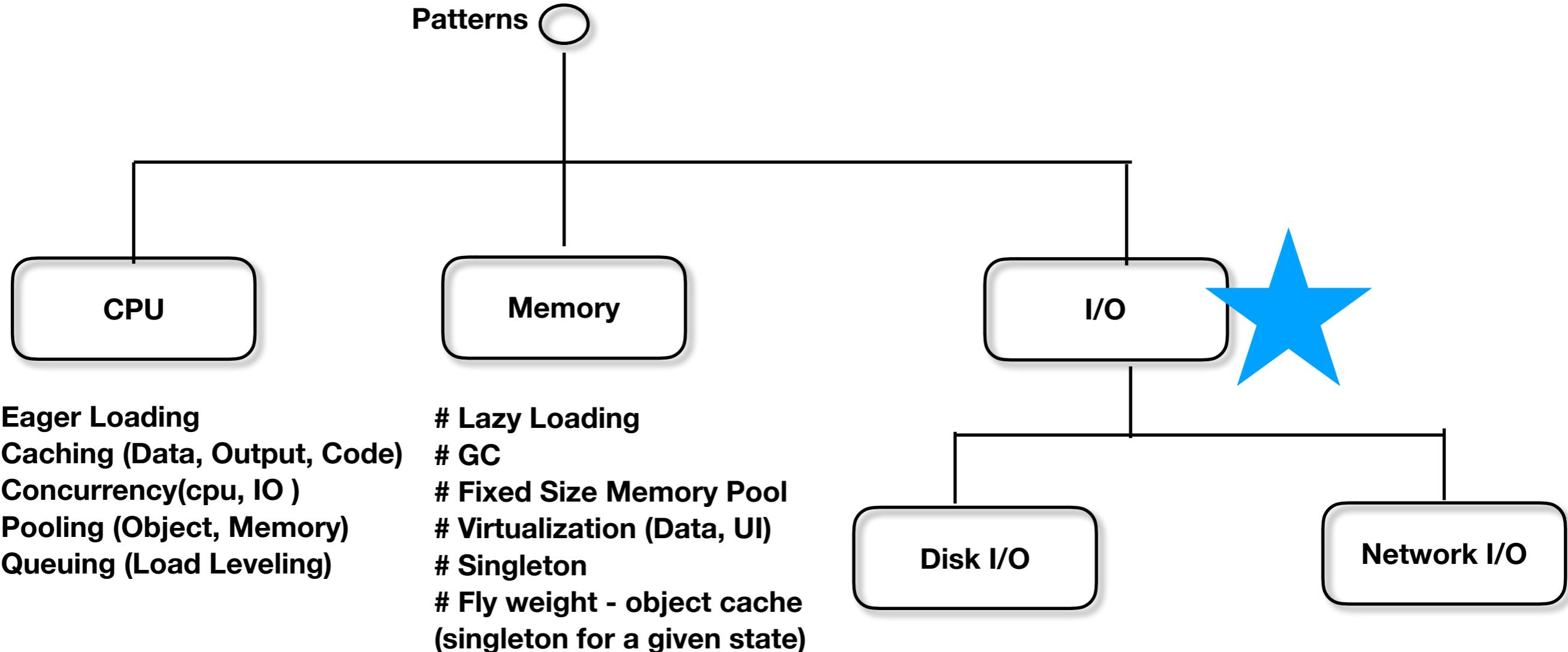
For the ones that look
risky, conduct some
experiments (prototypes)

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

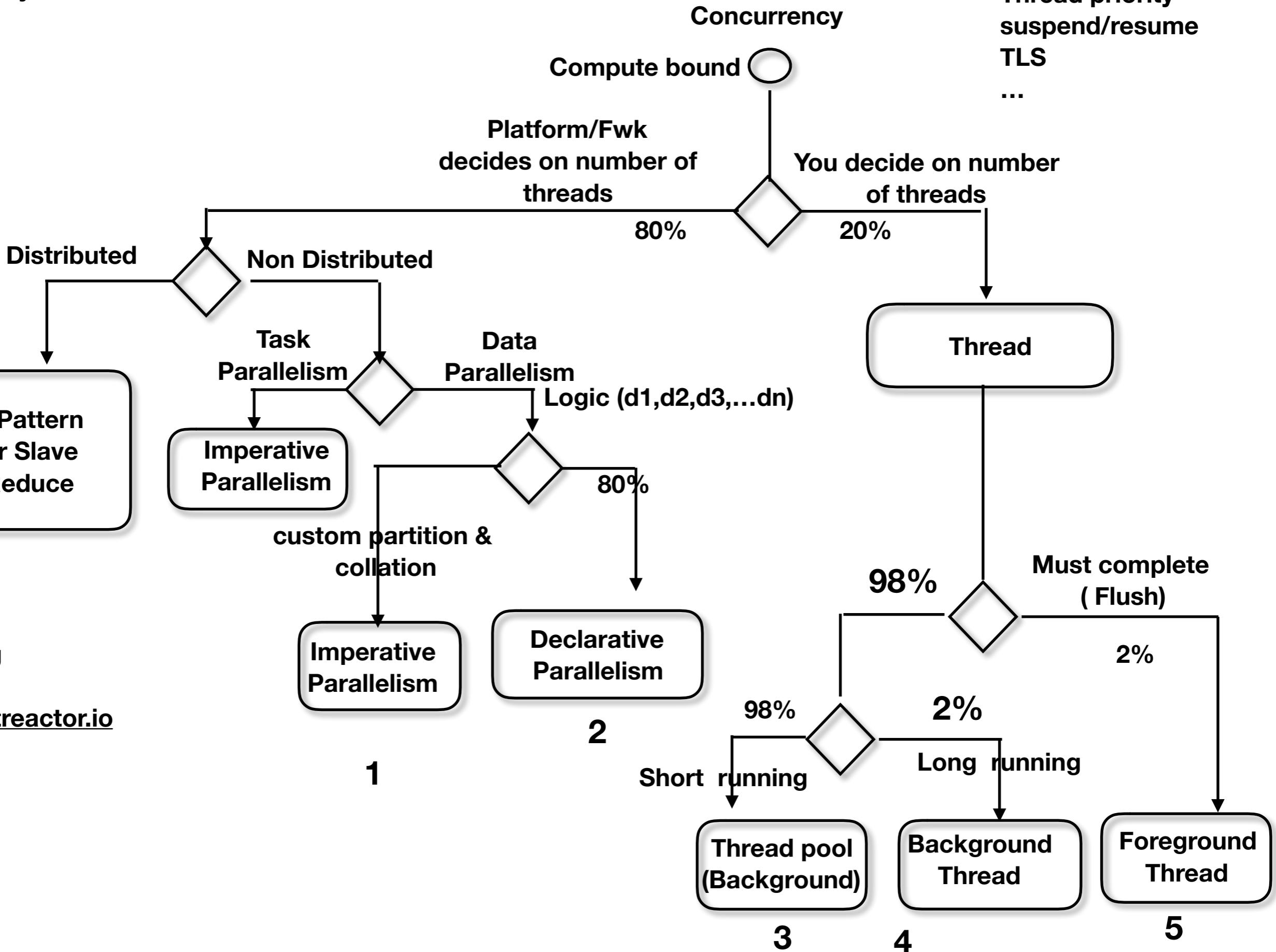
Spread your budget across your processing steps

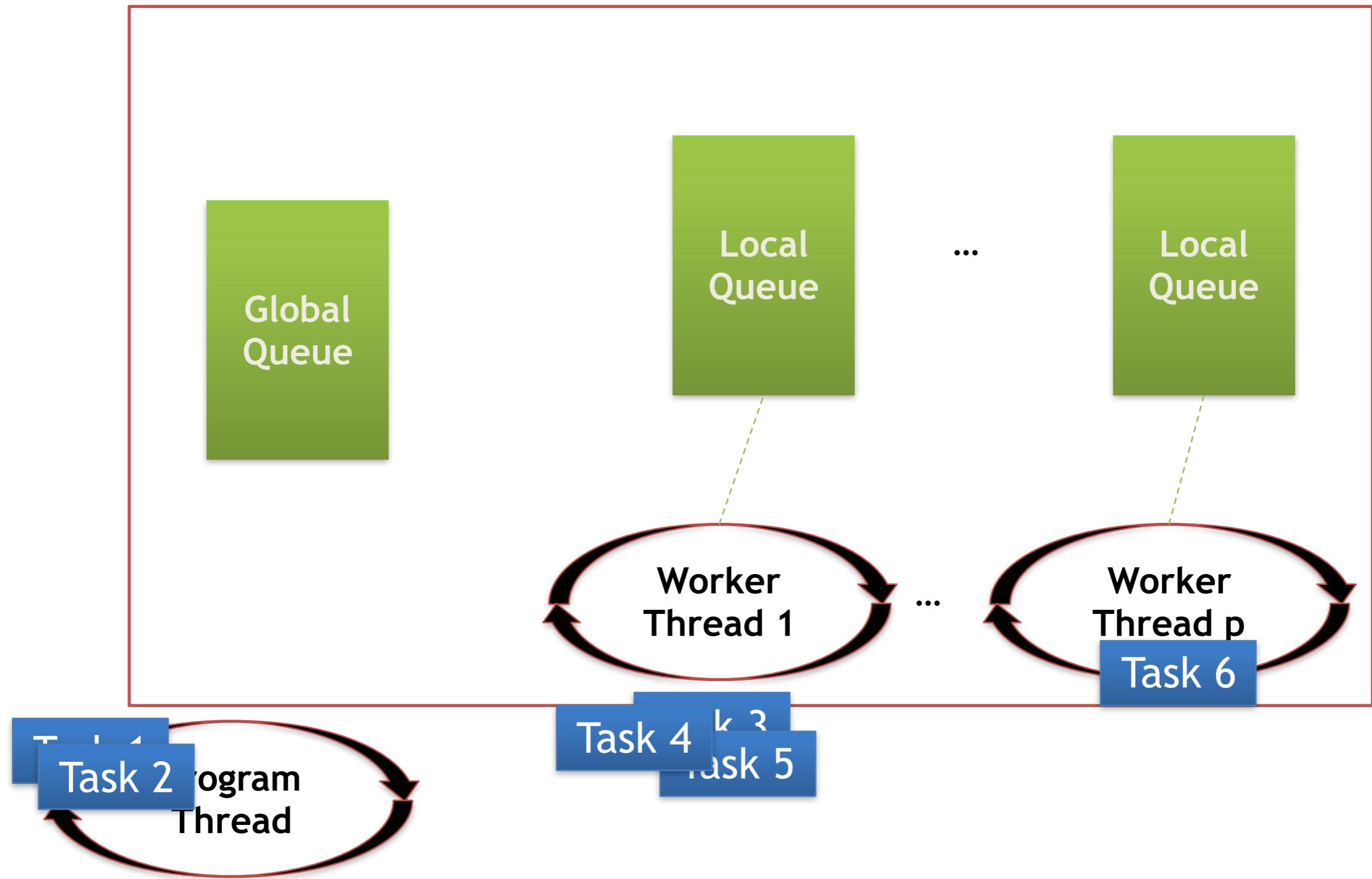
Performance

Patterns



Reactive System





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

v/s

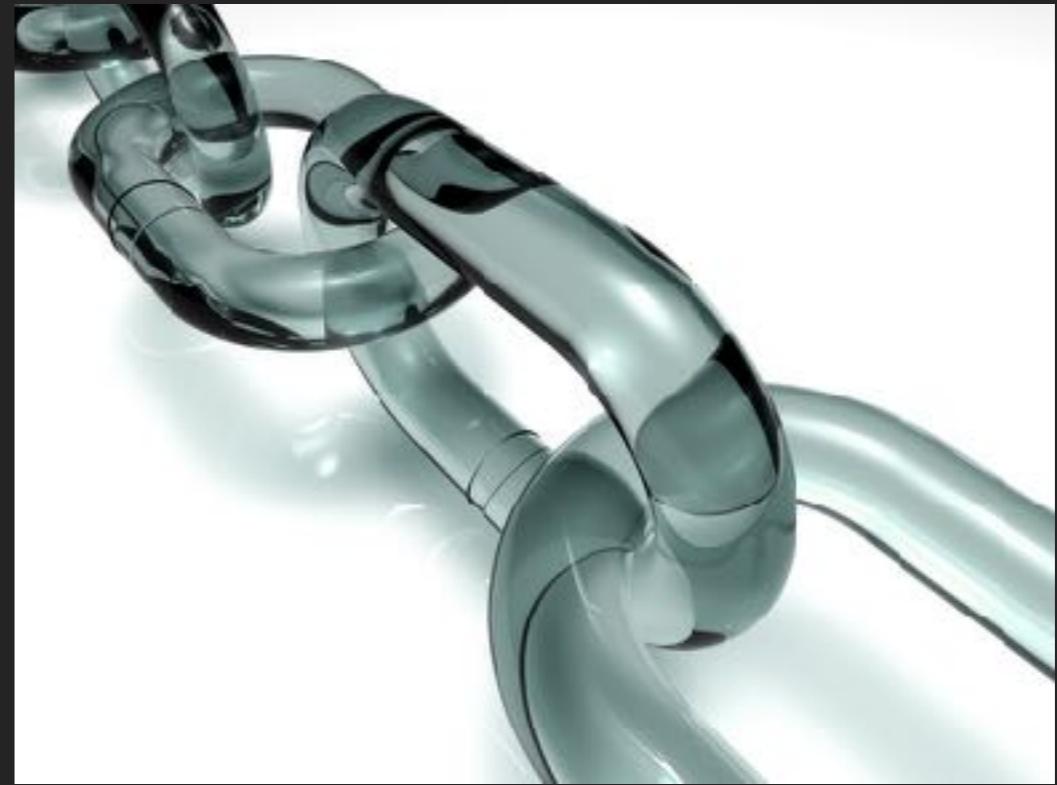
```
foreach(var item in items)
{
    Task.Factory.StartNew(() => do(item));
}
```

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

v/s

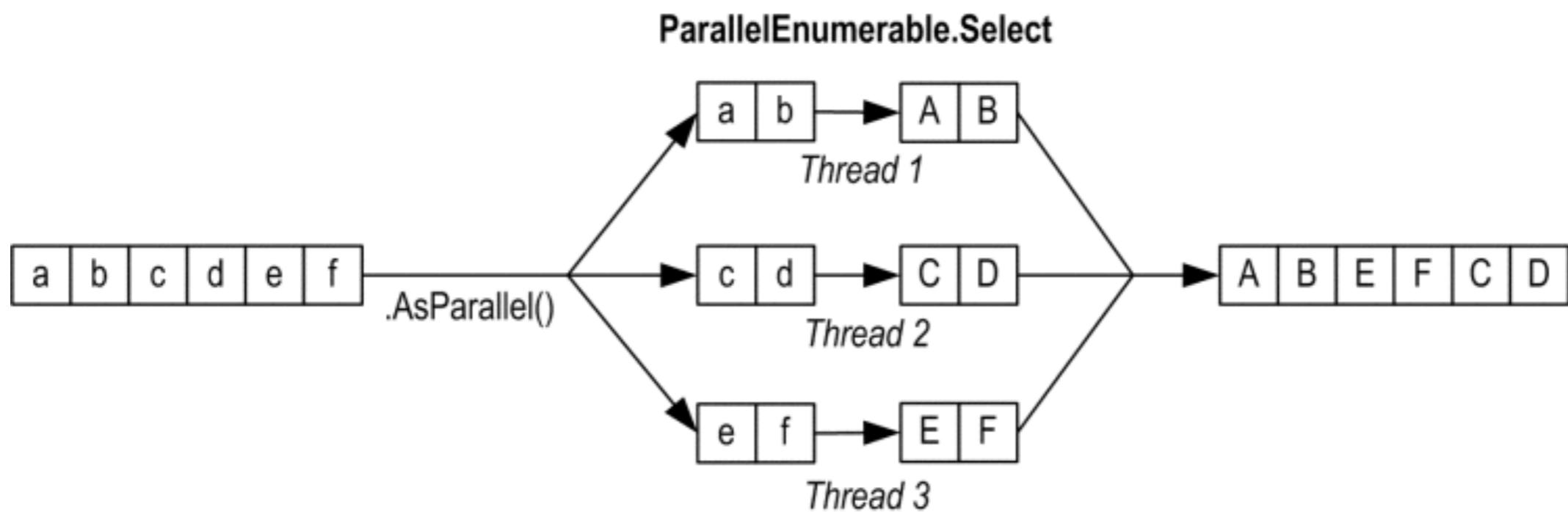
```
foreach(var item in items)  
{  
    Task.Factory.StartNew(() => do(item));  
}
```

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

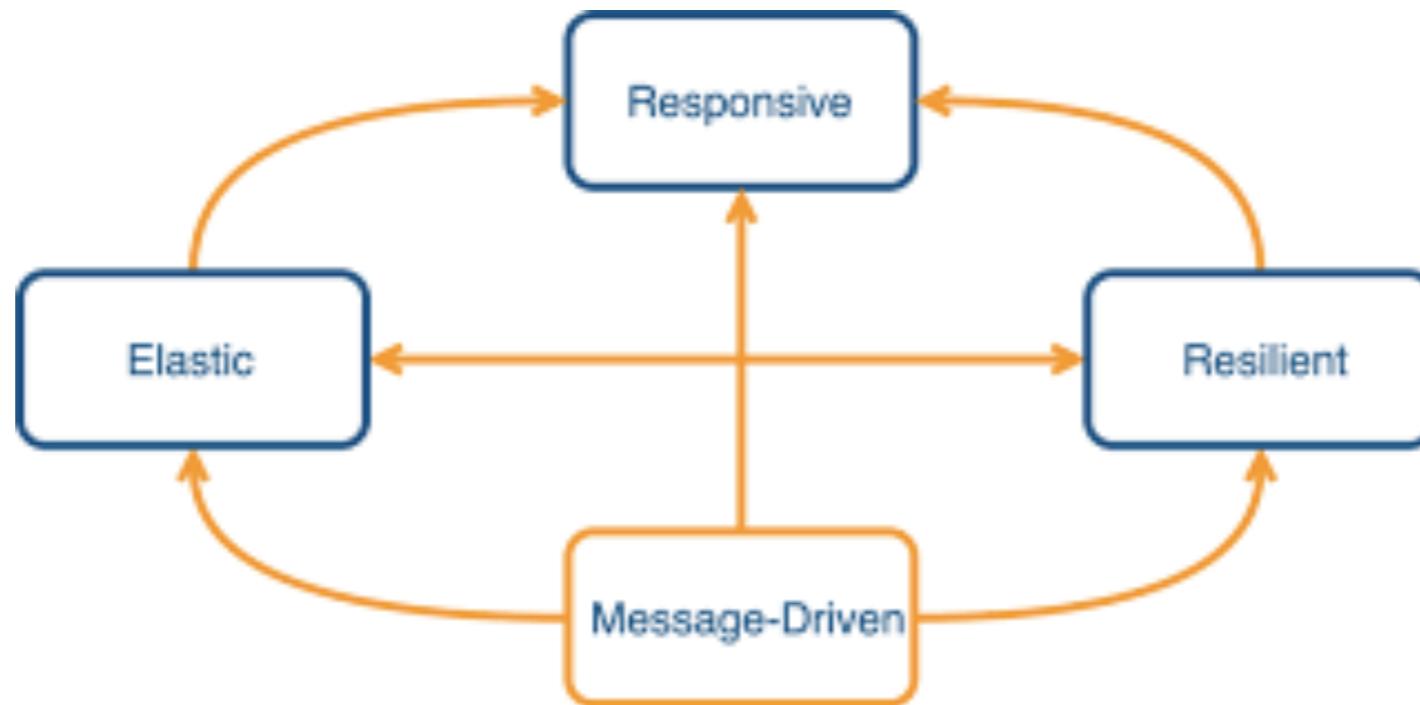


- Custom partitioning available through `Partitioner<T>`

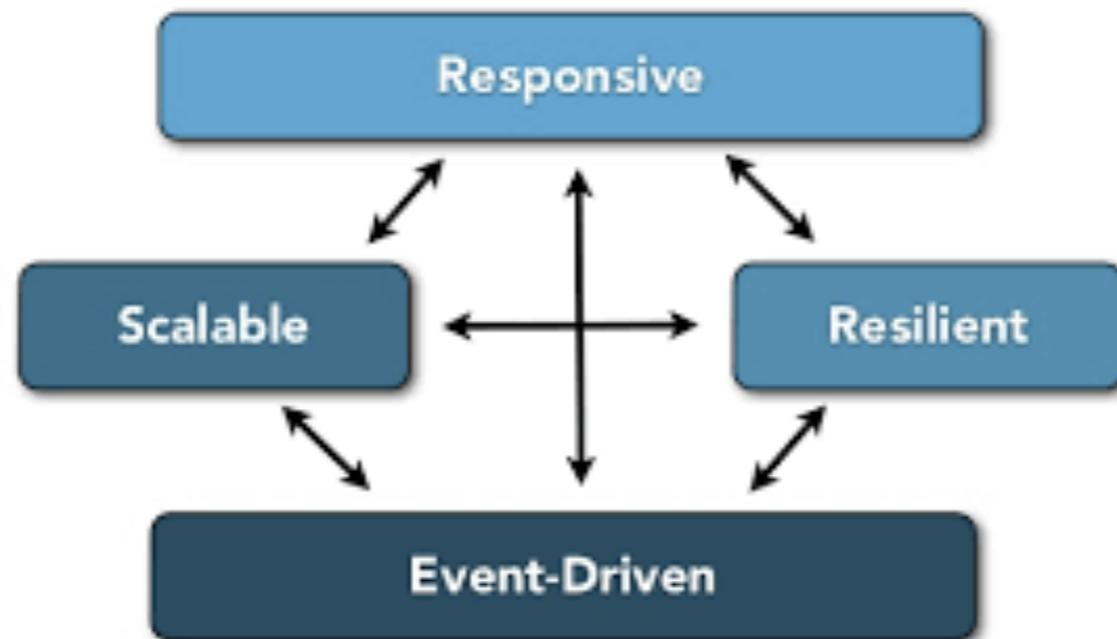
- `Partitioner.Create` available for tighter control over built-in partitioning schemes

Reactive Programming != Reactive System

The Reactive Manifesto



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

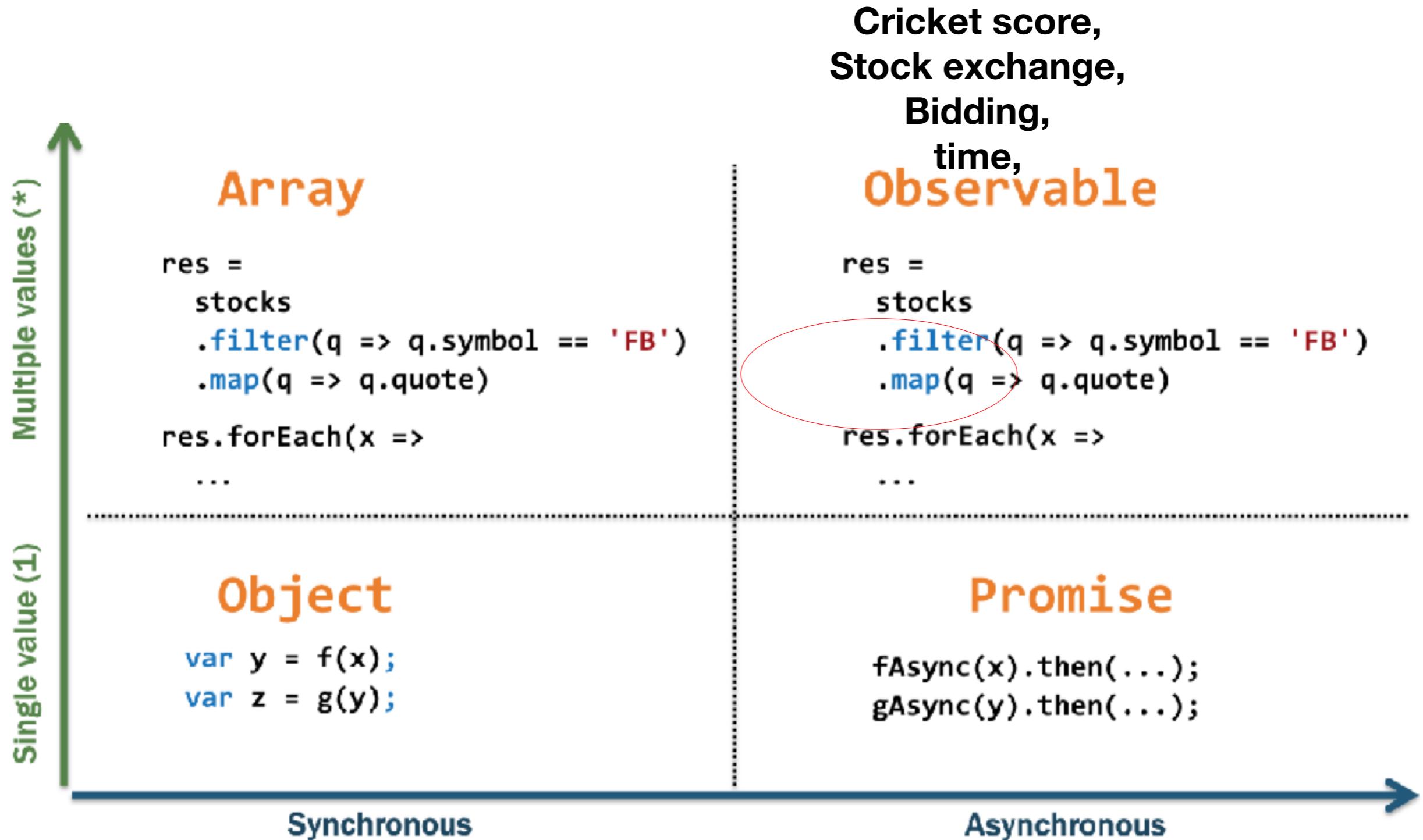


reactive manifesto

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.

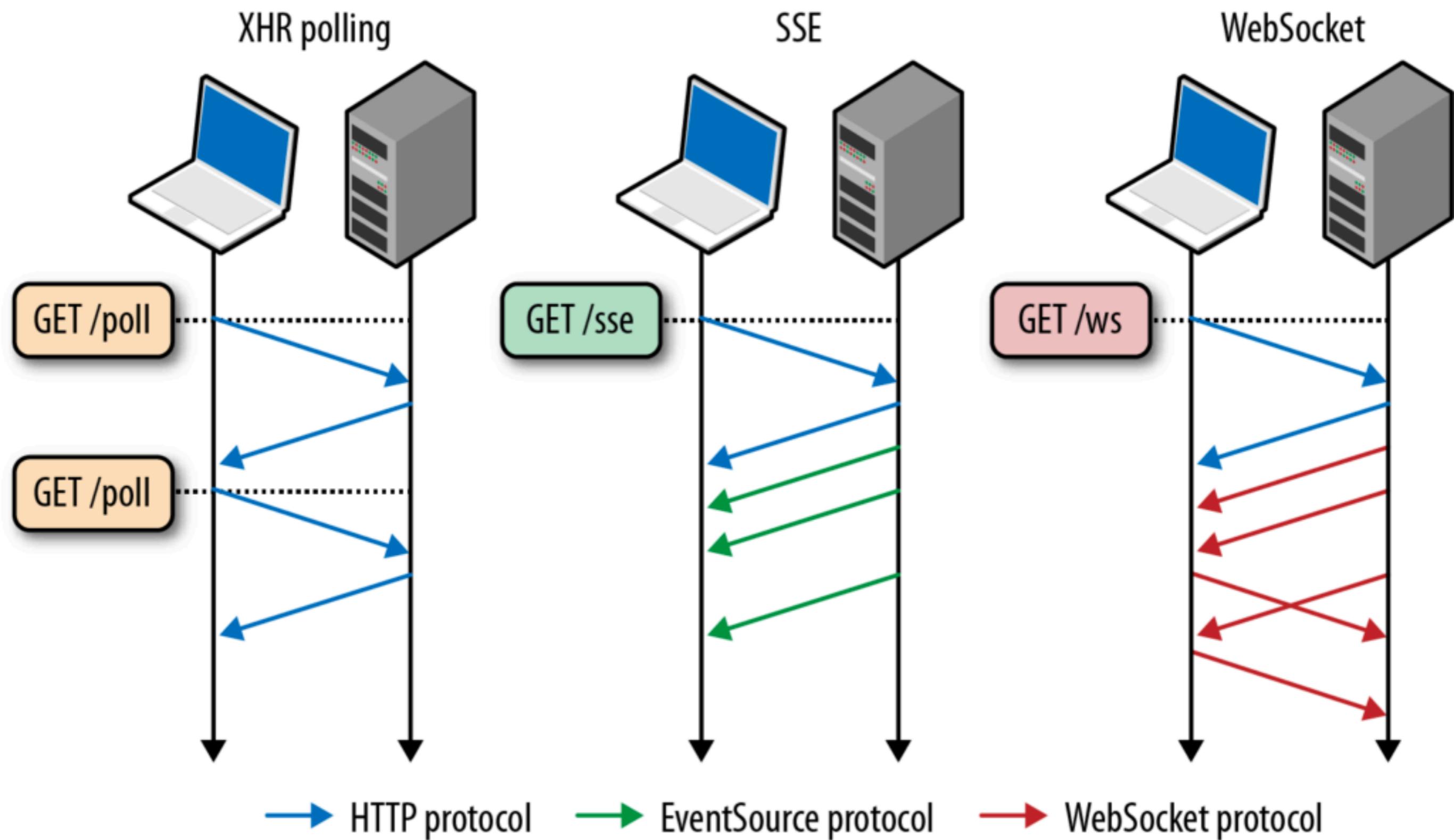
Despite the simplicity of these fundamental principles of reactive systems, building one of them is tricky. Typically, each node needs to embrace an asynchronous non-blocking development model, a task-based concurrency model and uses non-blocking I/O.

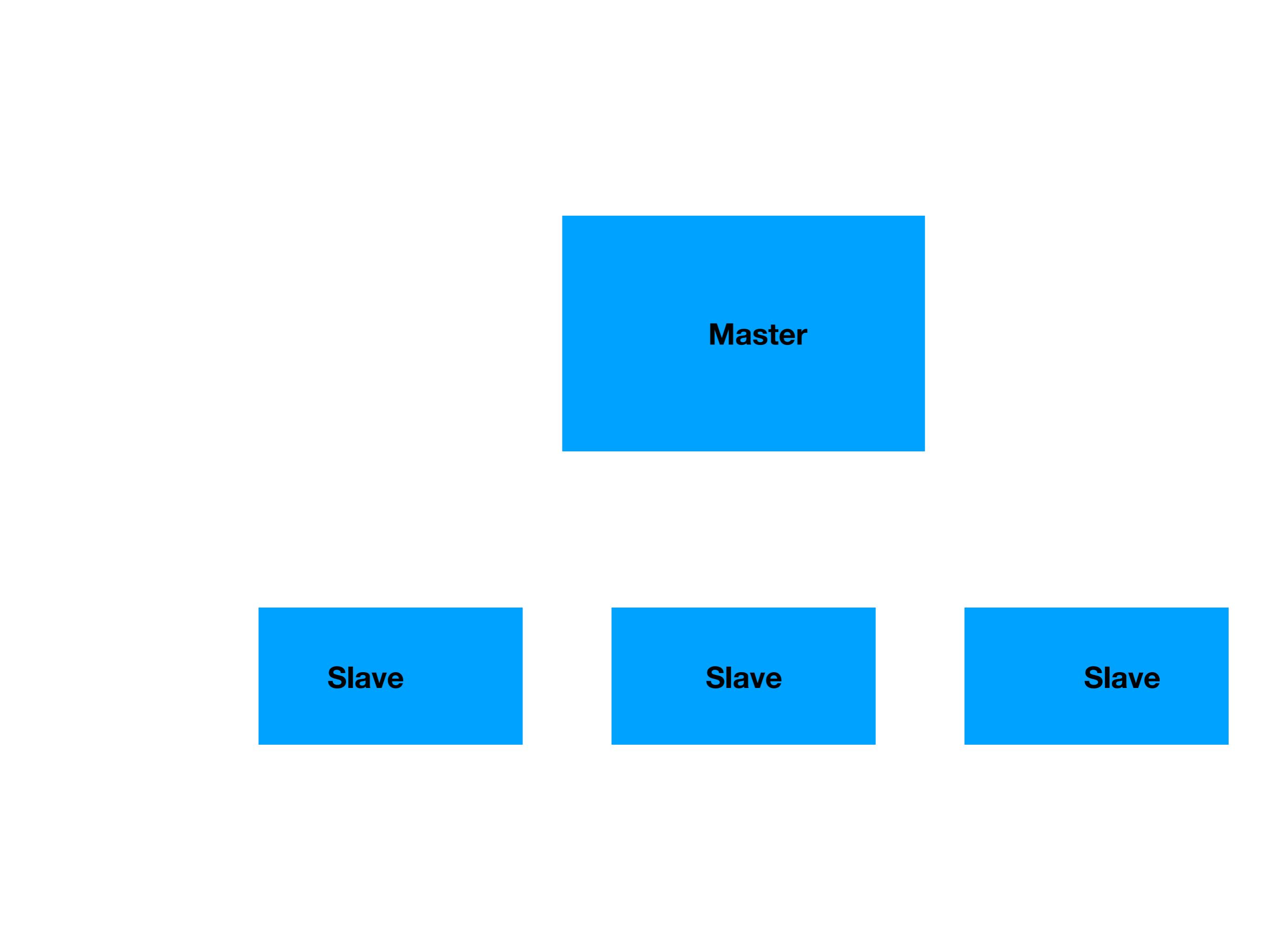


- Asynchronous data streams
- **Everything** is a stream

|

|





Master

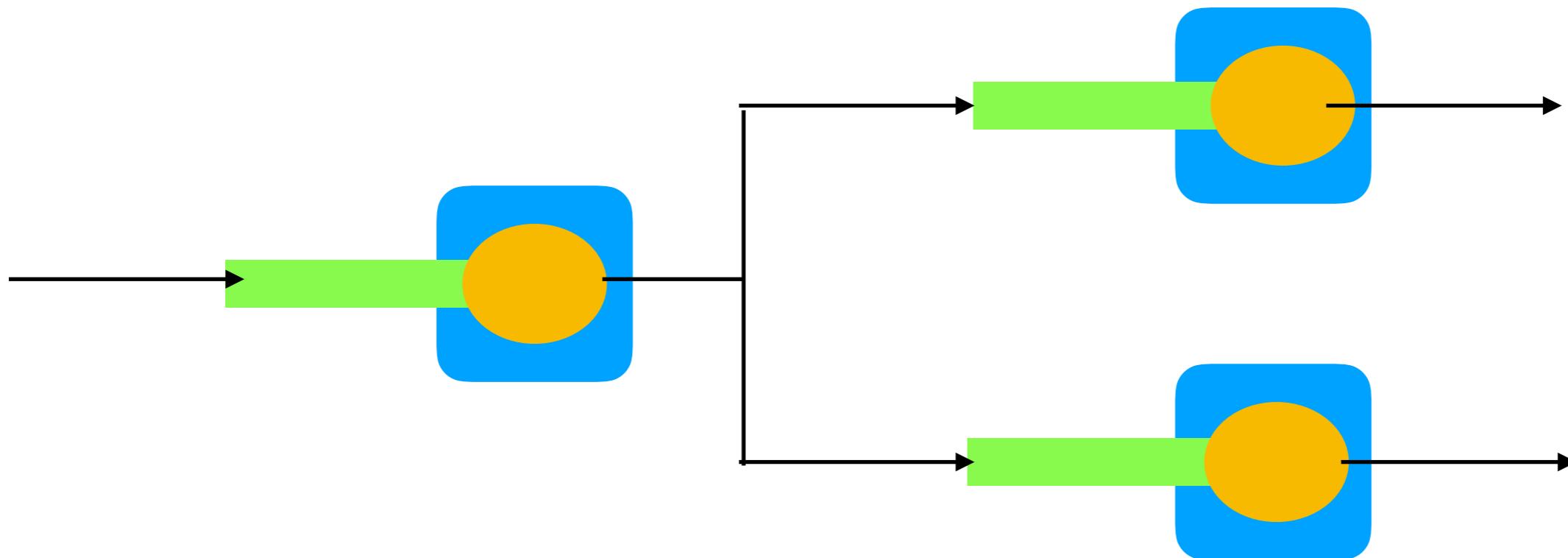
Slave

Slave

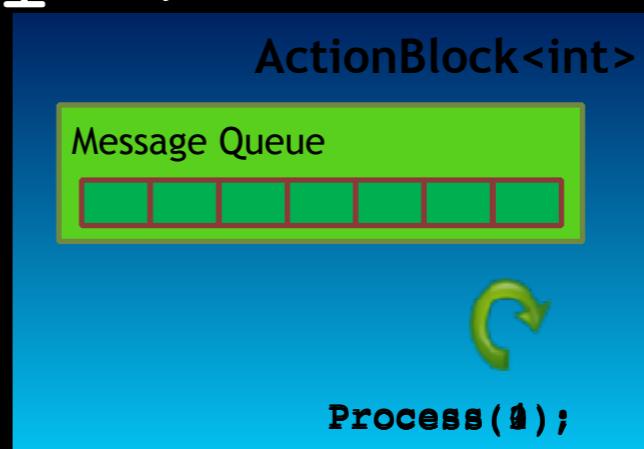
Slave

dict[Key,value]

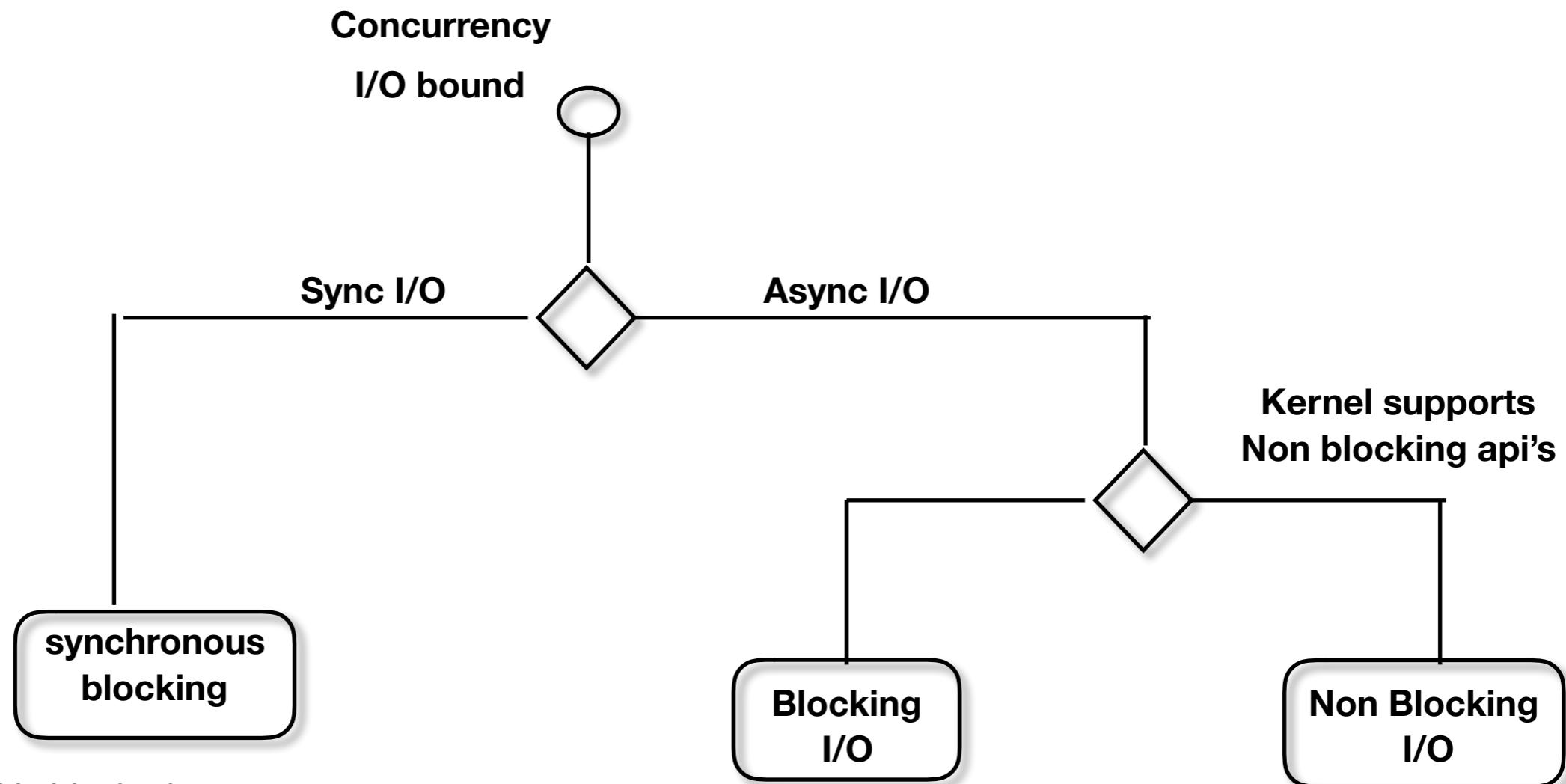
Actor Pattern



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



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



Thread is blocked while hardware performs the work.

Thread is not blocked, But there is still a thread lower down the stack which is blocked on the IO

The kernel doesn't block your thread and returns from the IO call immediately.

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

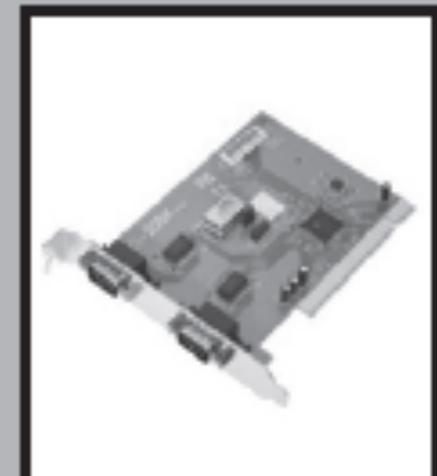
Network



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.

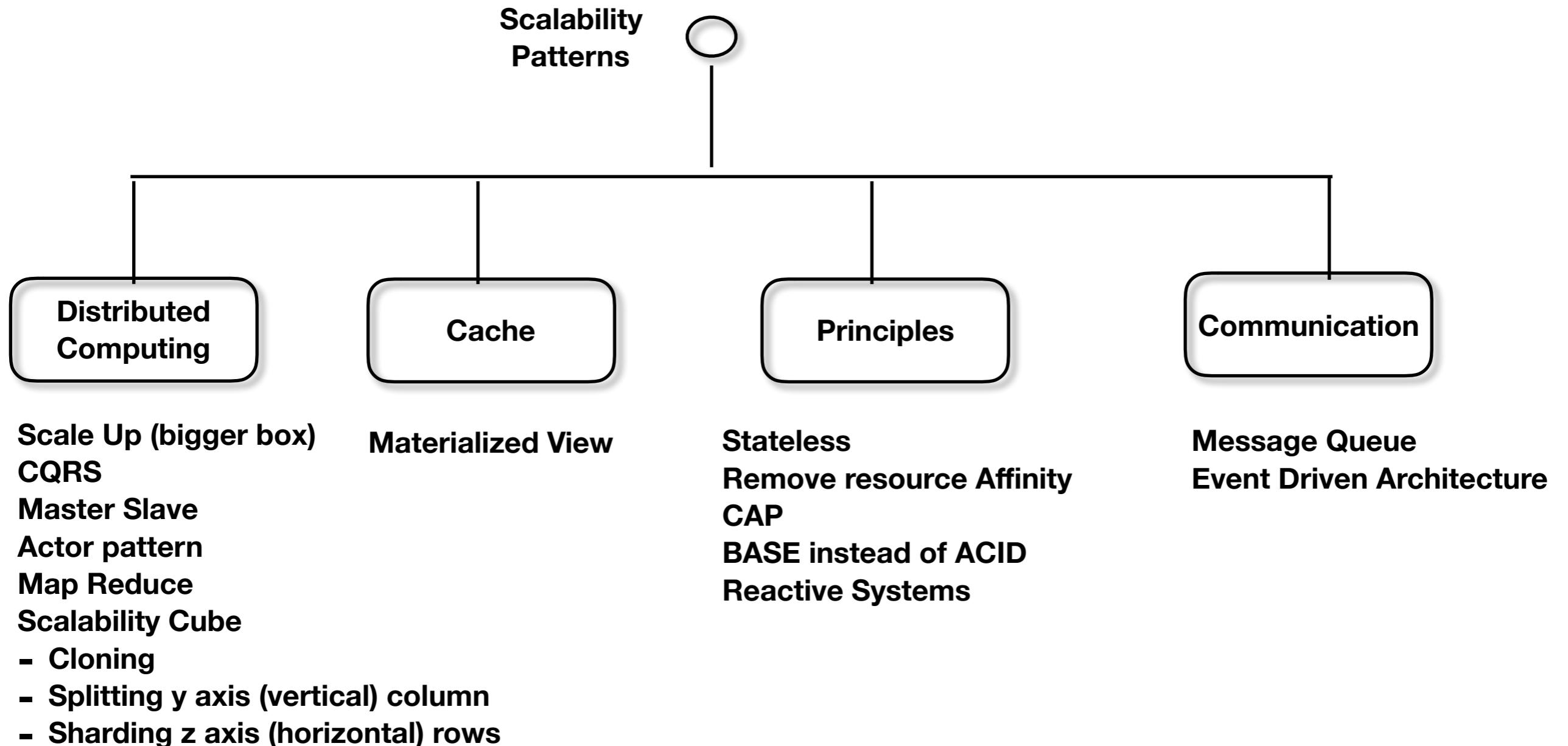
Notify

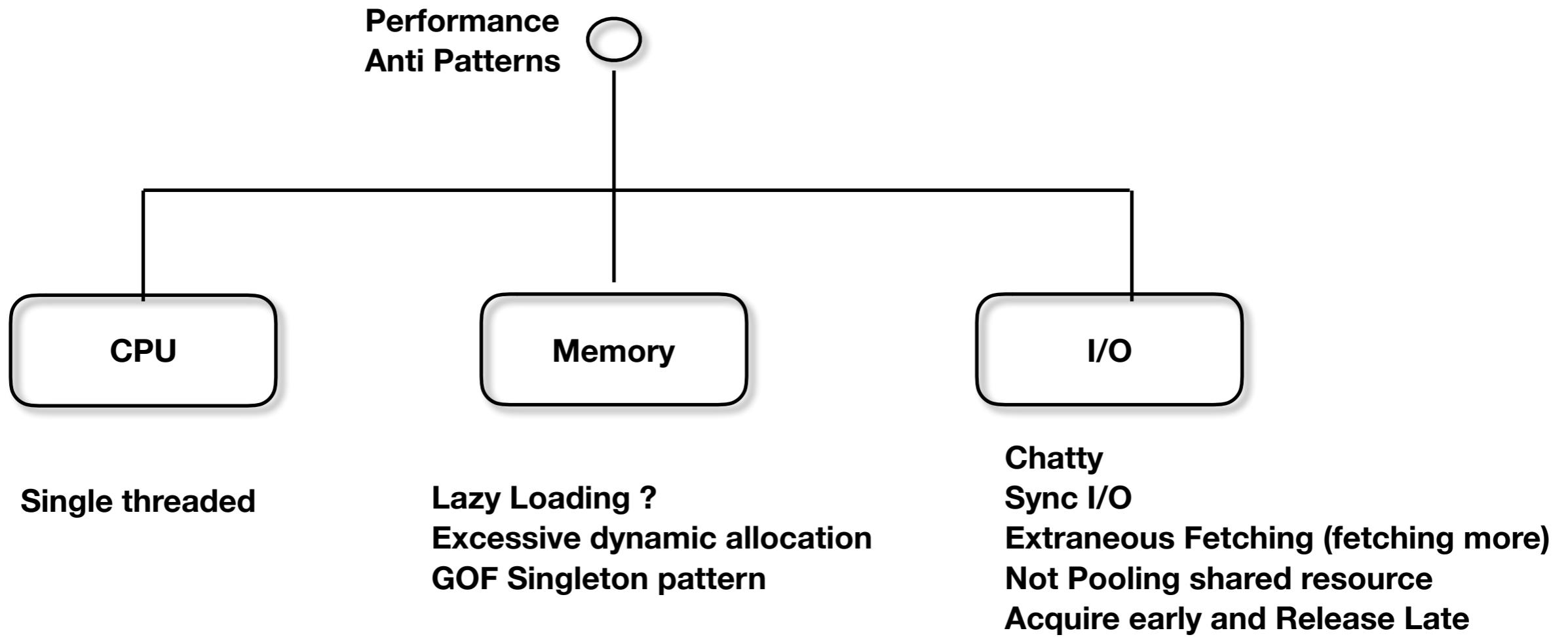
Signaling
constructs

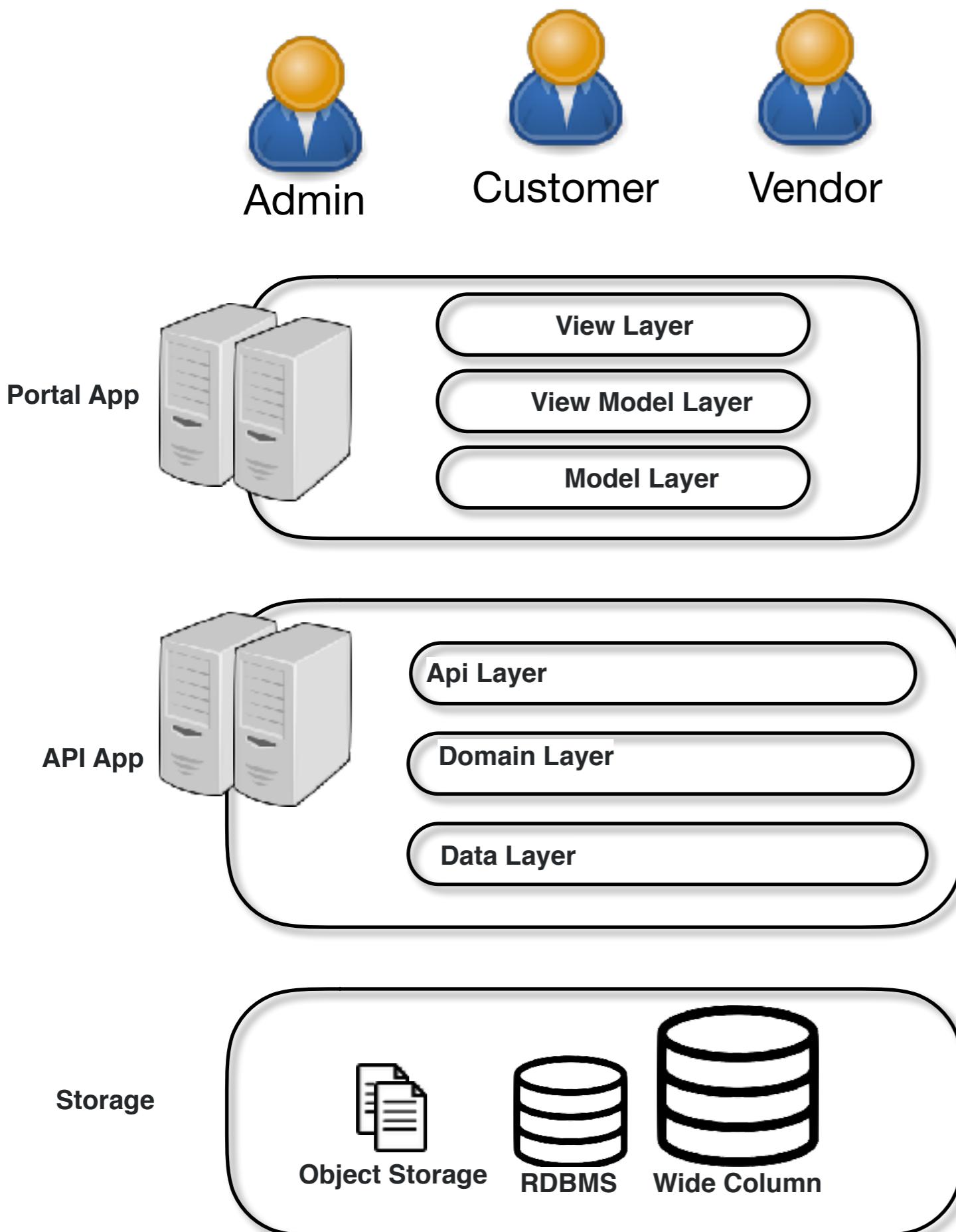
- event wait
- Latch
- Barrier



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







Horizontal scaling

portal
API
database sharding(geo, category, ..)

Vertical Splitting

database splitting (vehicle, user, order,...)

UI Virtualization

pagination of view
#

SOC

Historical data (column)
CQRS ?

BASE

Eventual Consistency (CUD vehicles, Order)
#

network

Compression (gzip)
circuit breaker

Cache

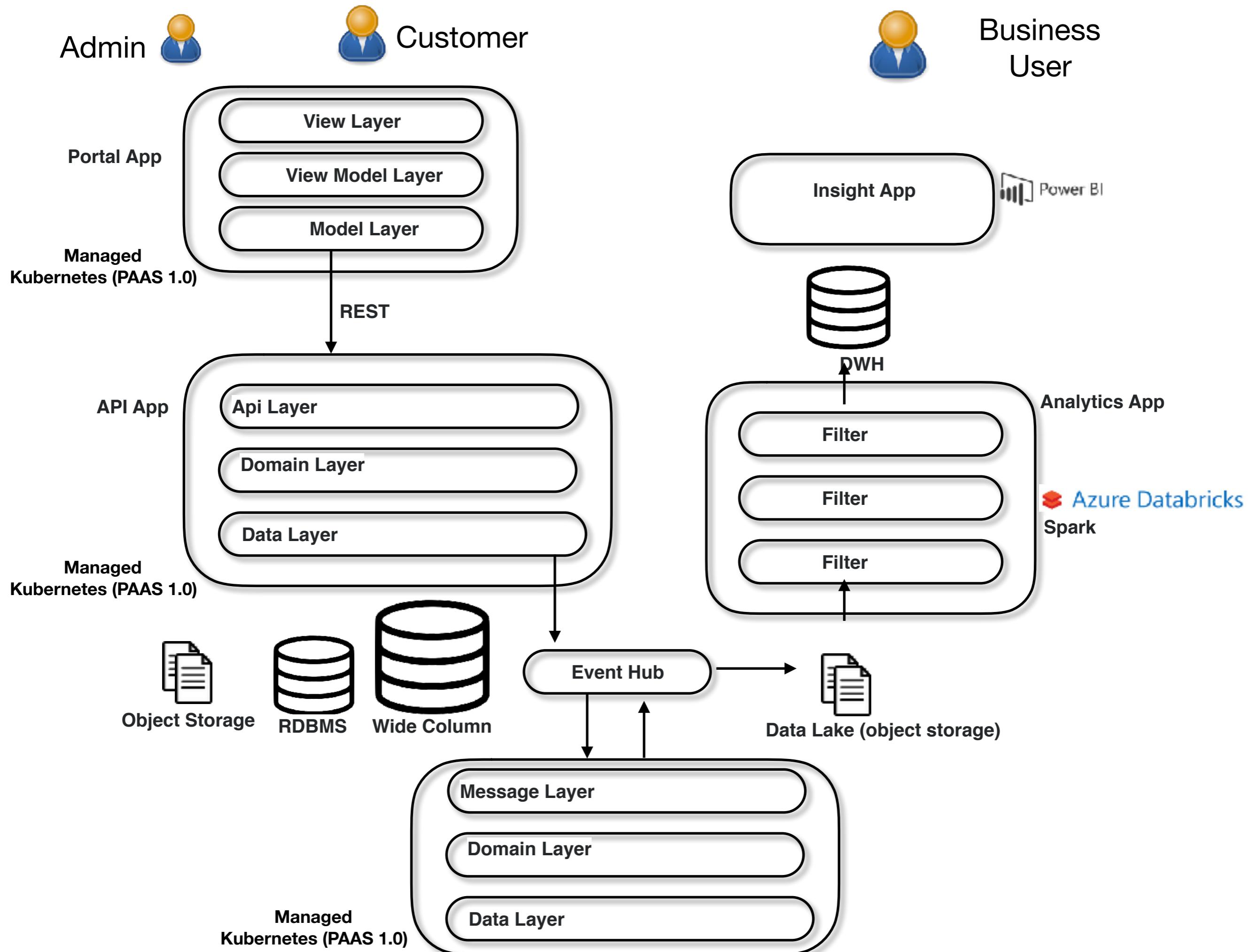
Browser Cache
API response
static content

CDN Cache

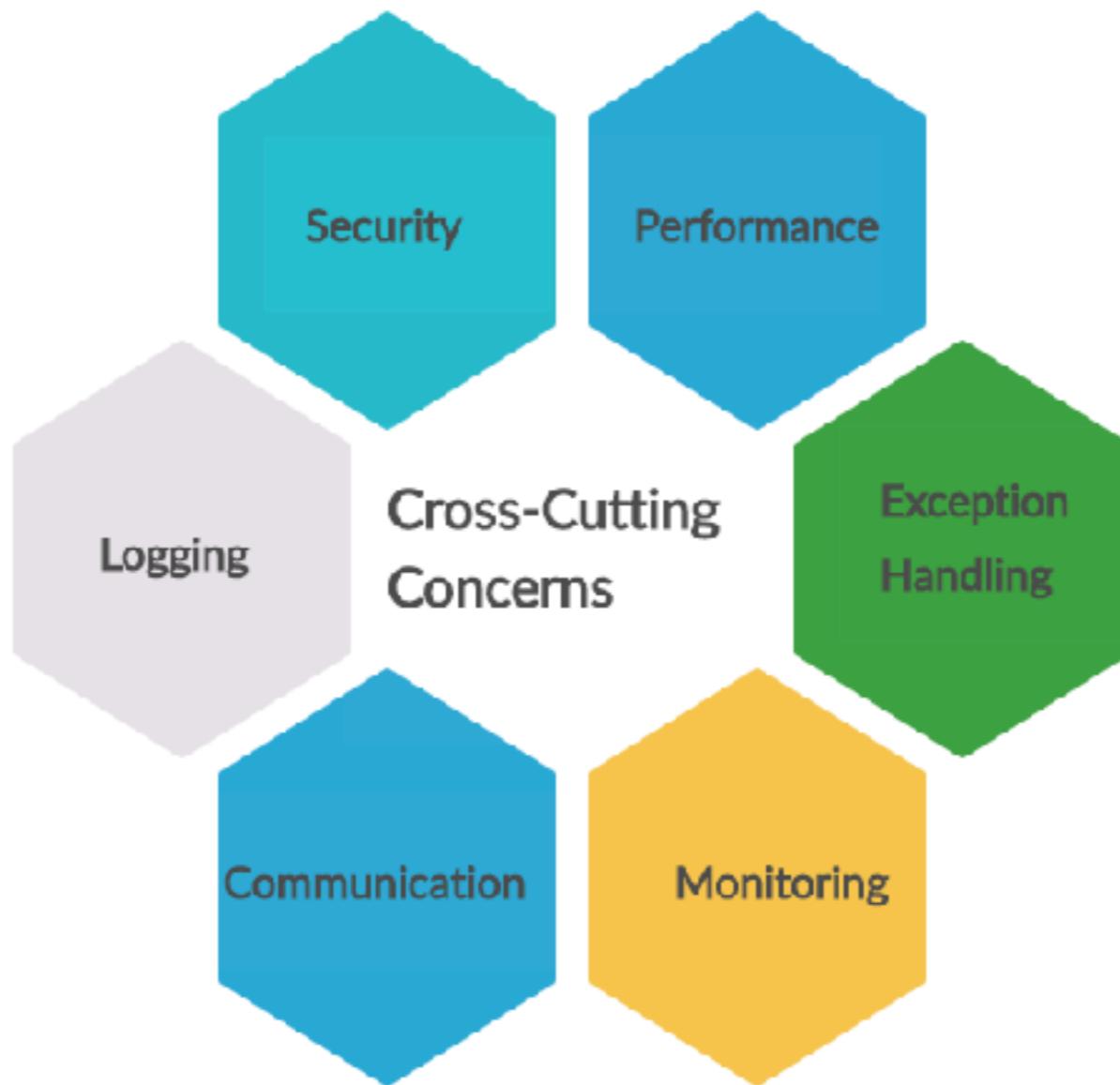
images
js
styles

Reverse Proxy Cache
API response

Storage Cache
query Cache

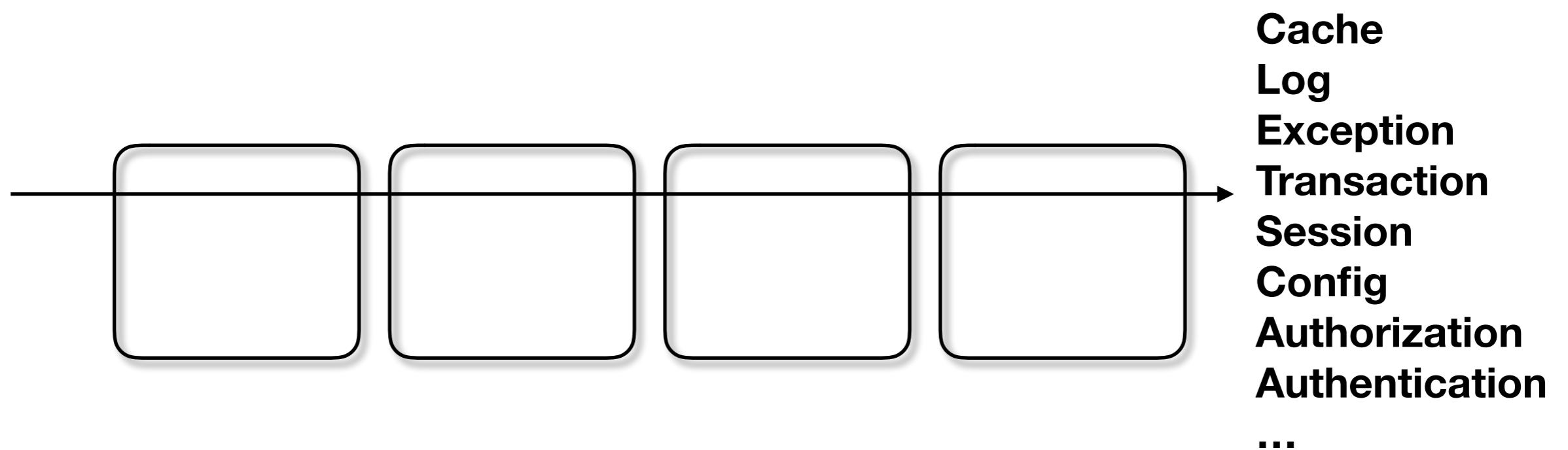


Cross cutting concerns

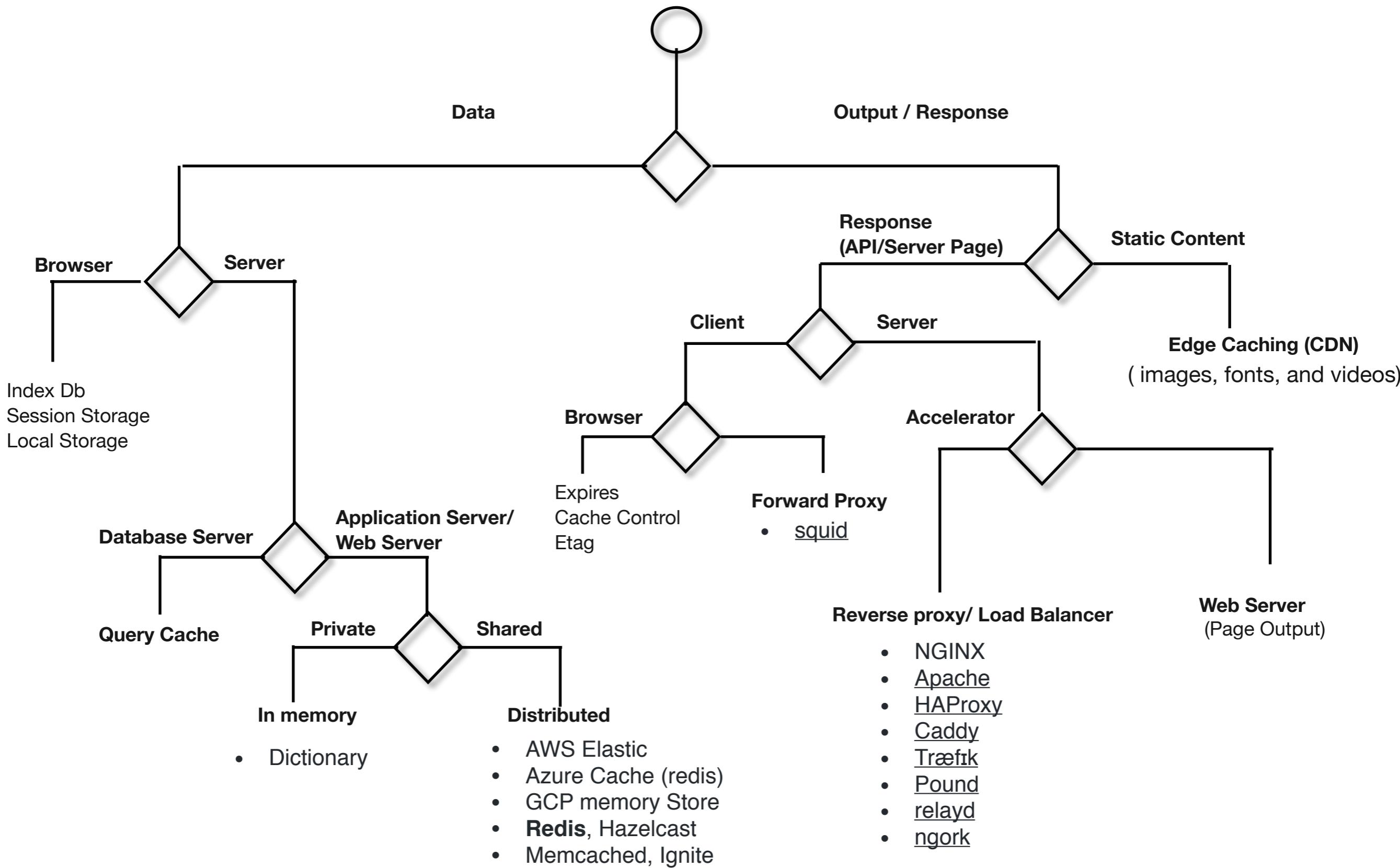


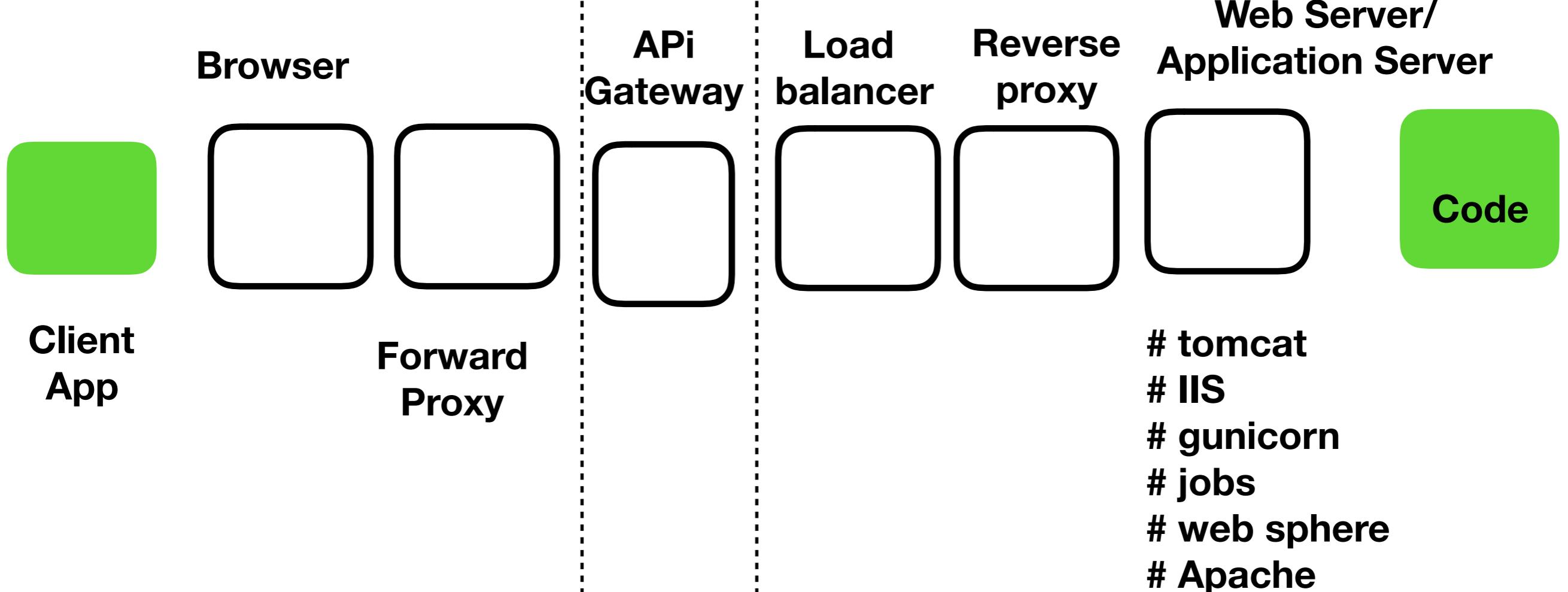
The **crosscutting concern** is a concern which is needed in almost every module of an application.

Service



Choose Cache

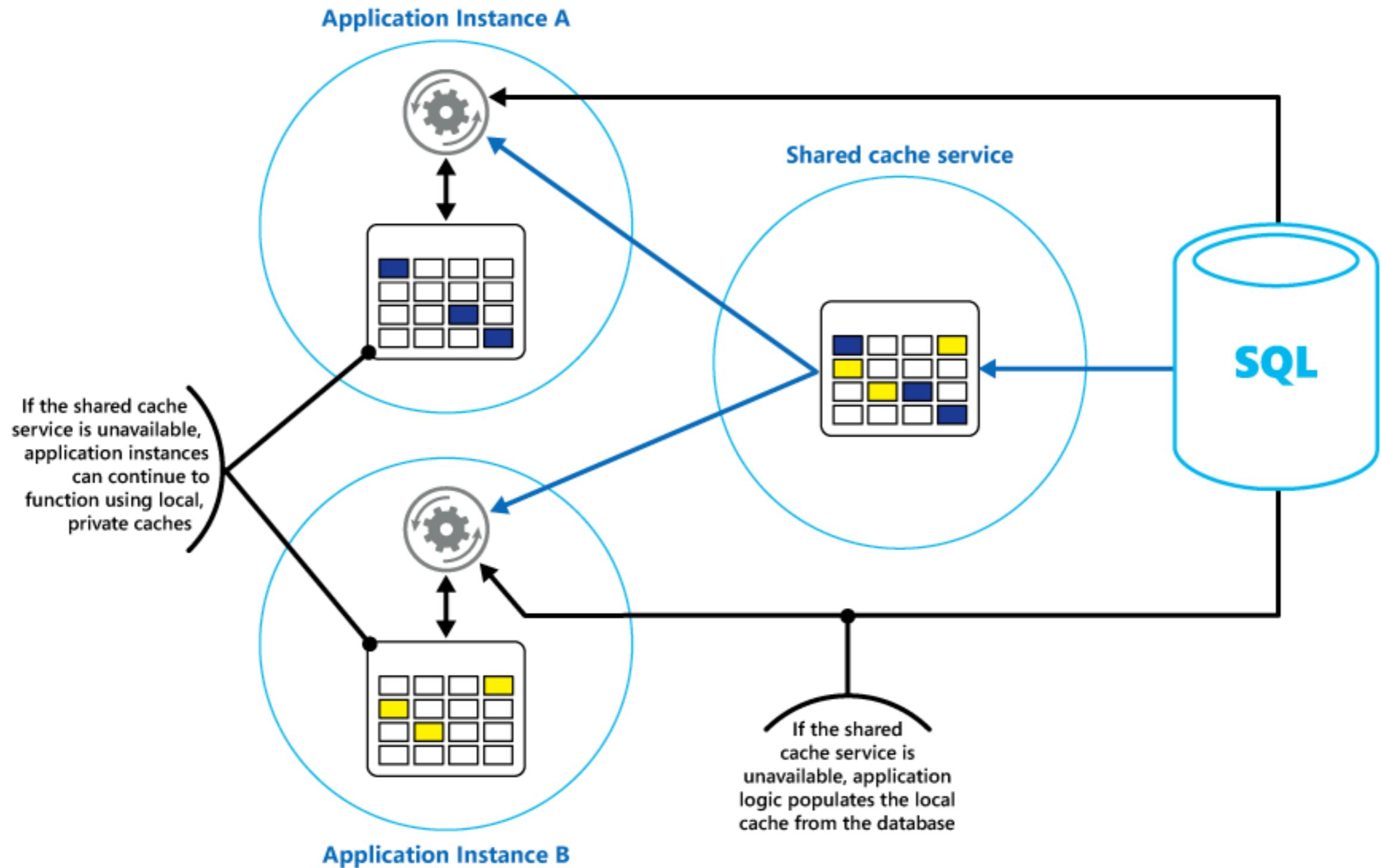




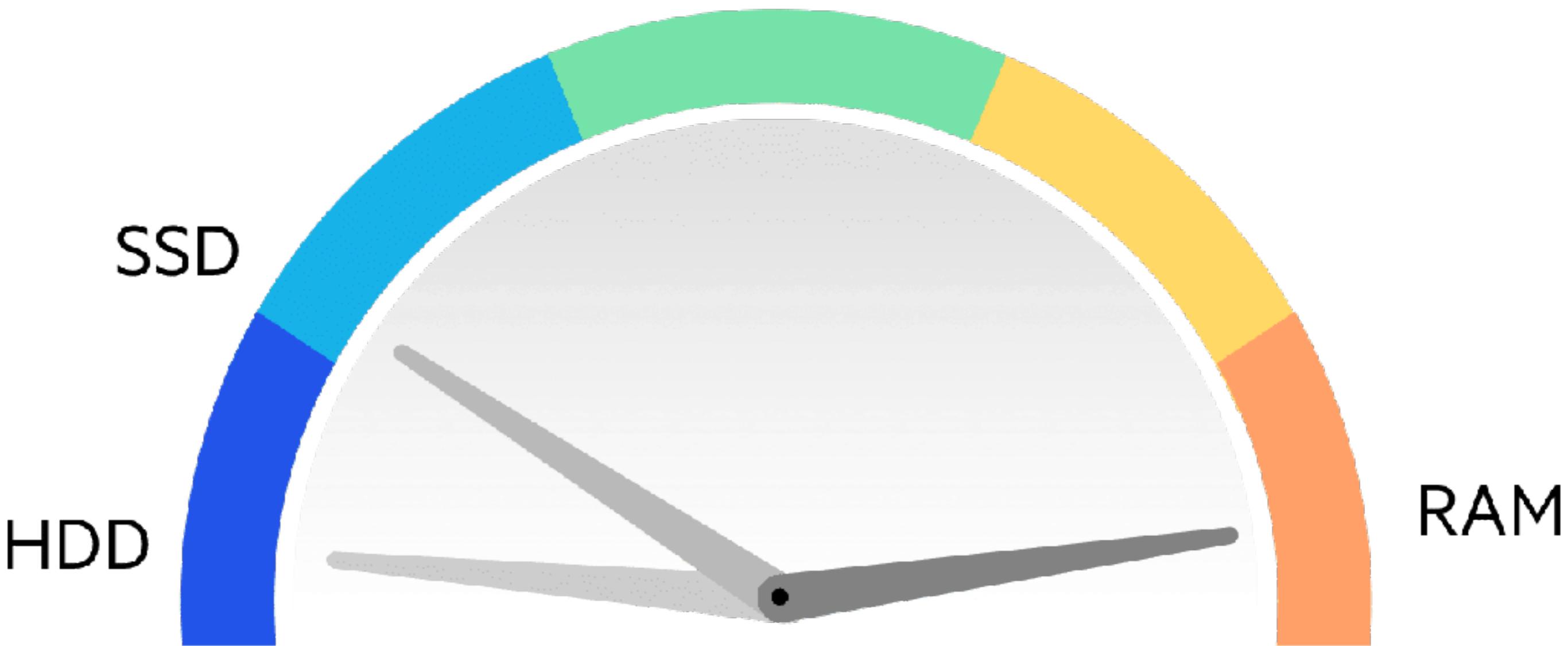
Web acceleration

- SSL/TLS Processing
- Compression
- Caching and Prefetching

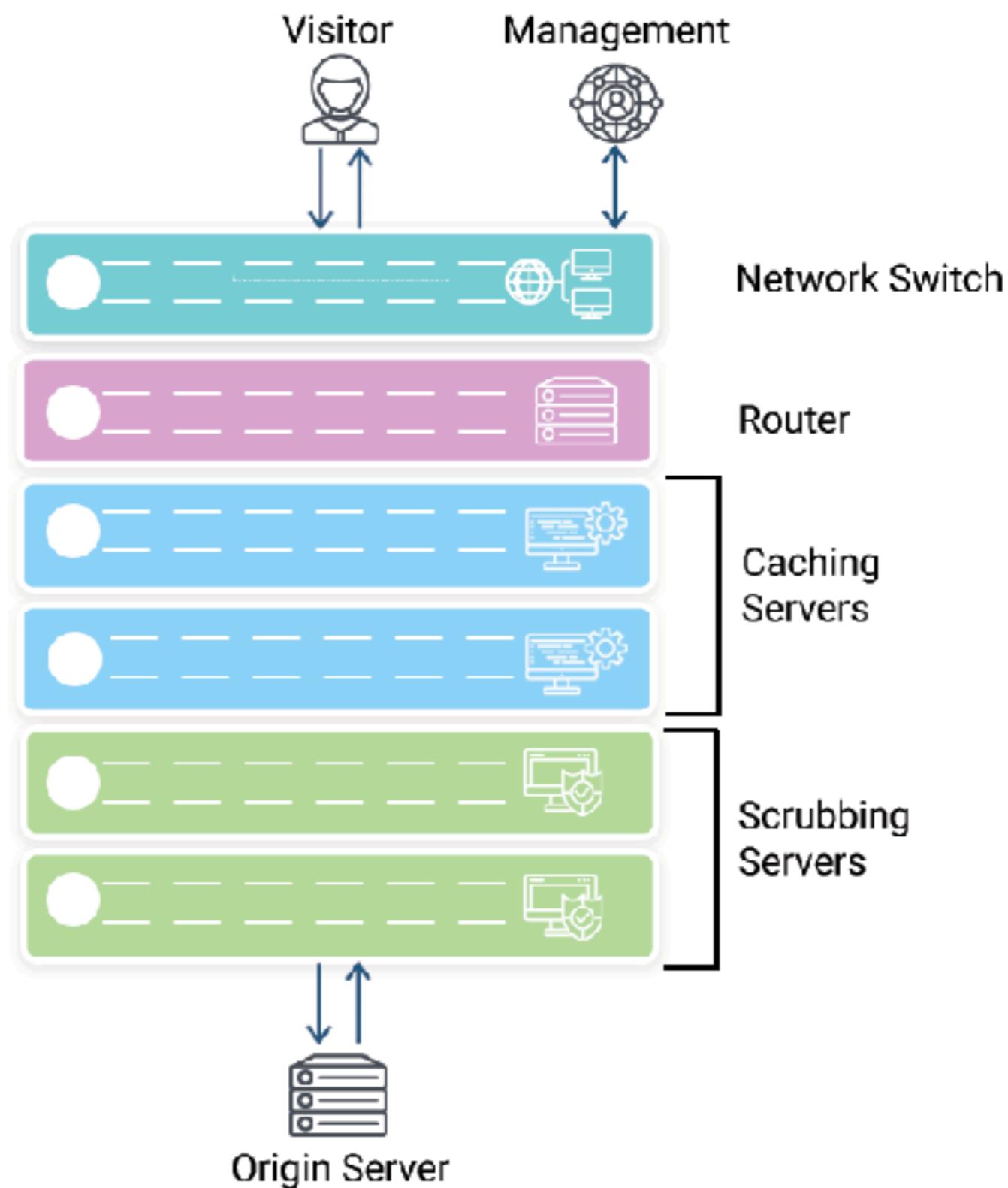
Using a local private cache with a shared cache.



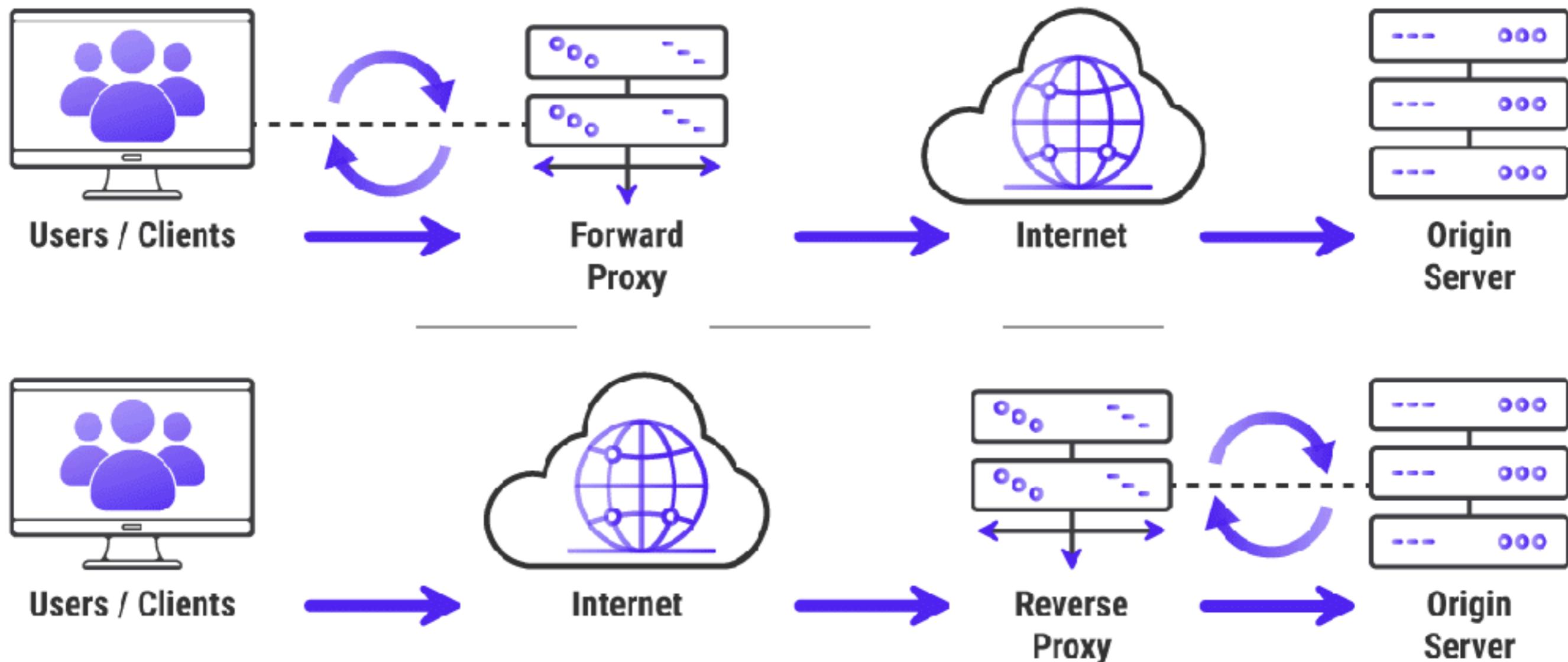
Speed Comparison



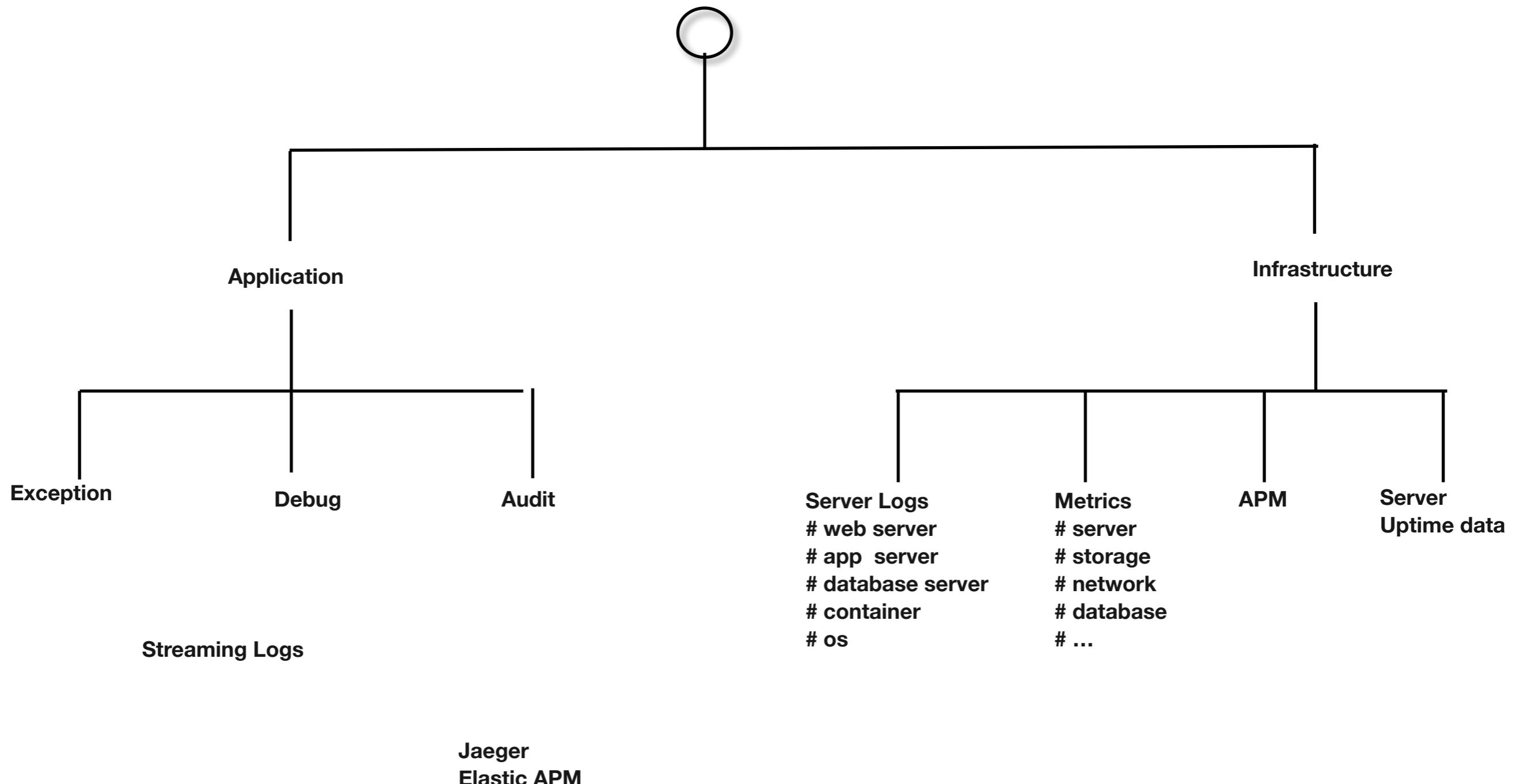
HOW CONTENT DELIVERY NETWORK FUNCTIONS

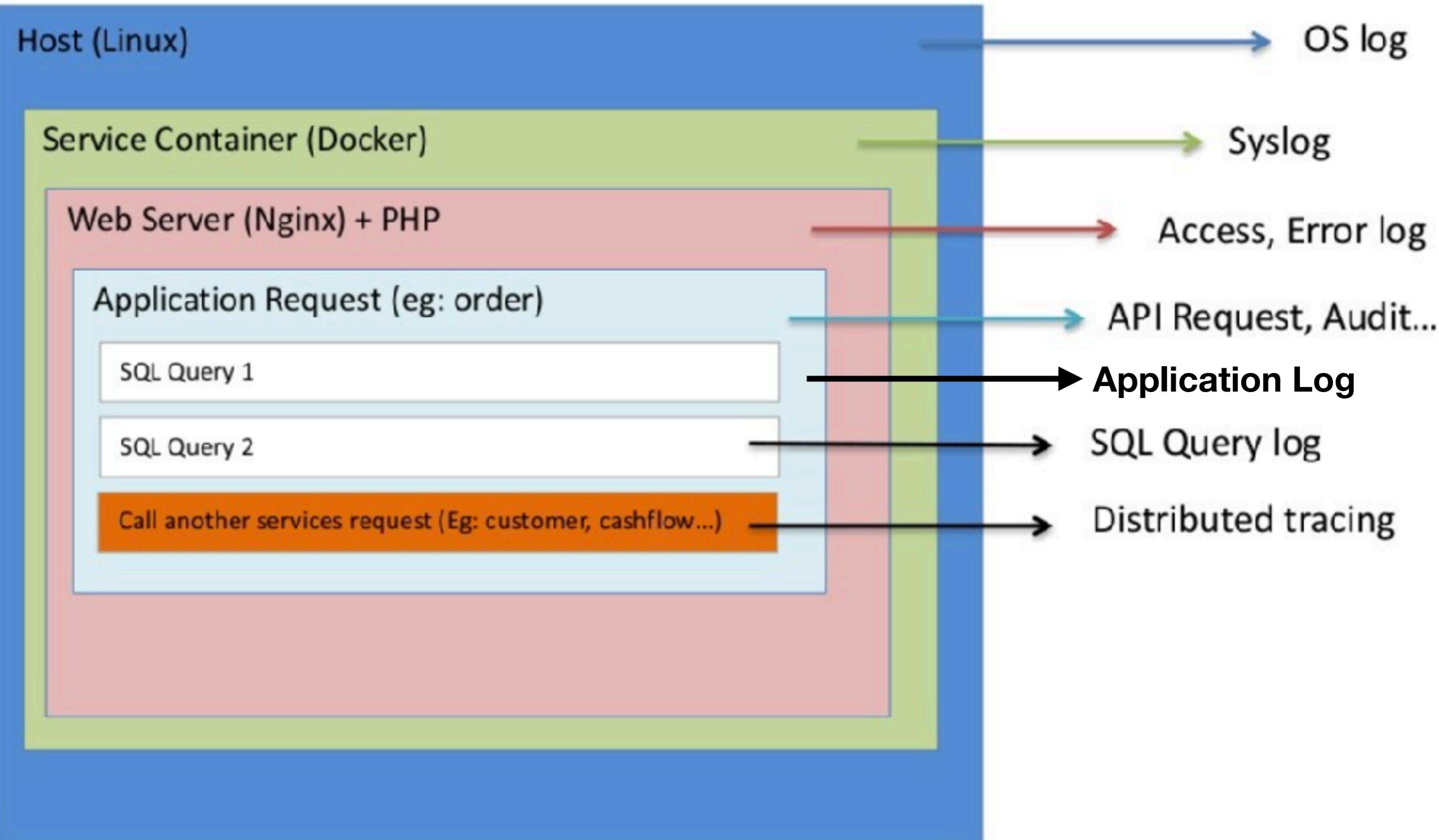


Forward Proxy vs Reverse Proxy



Logs





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

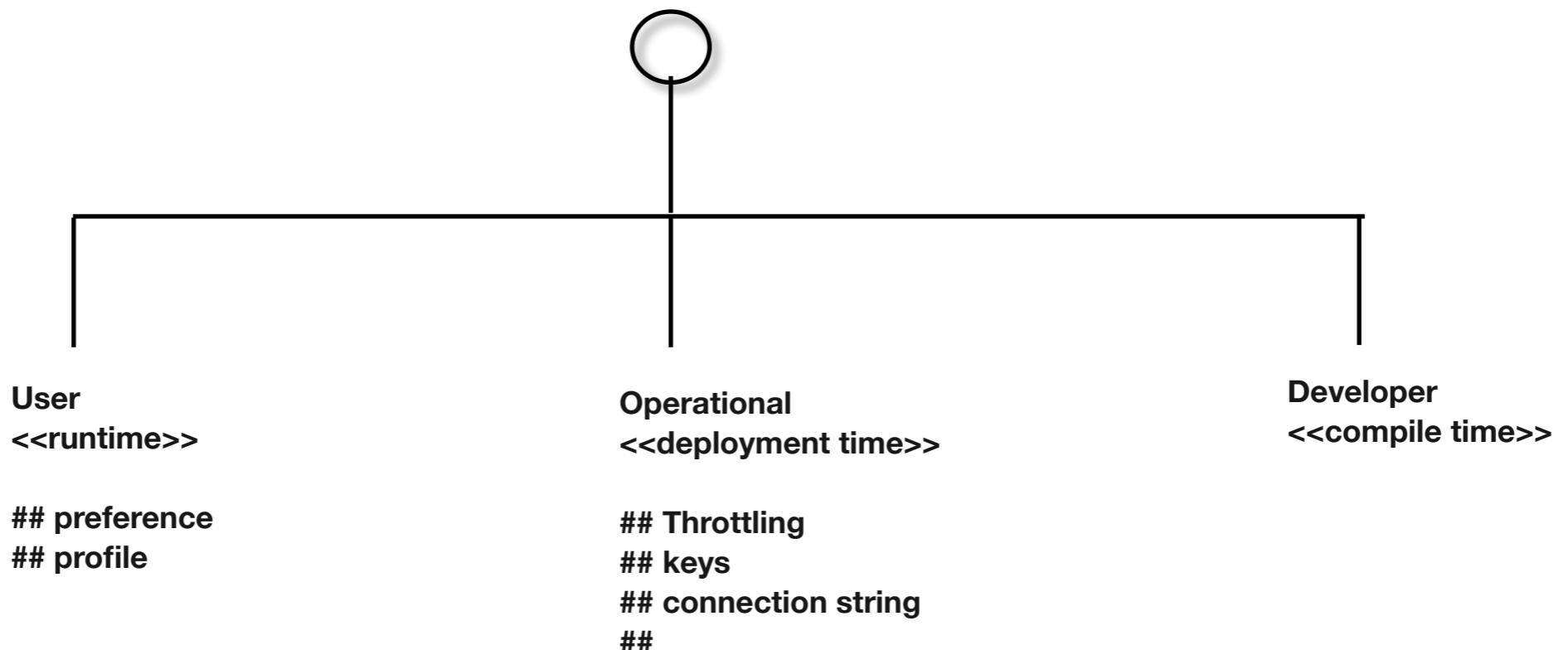
Uptime Data

Uptime
Response Time

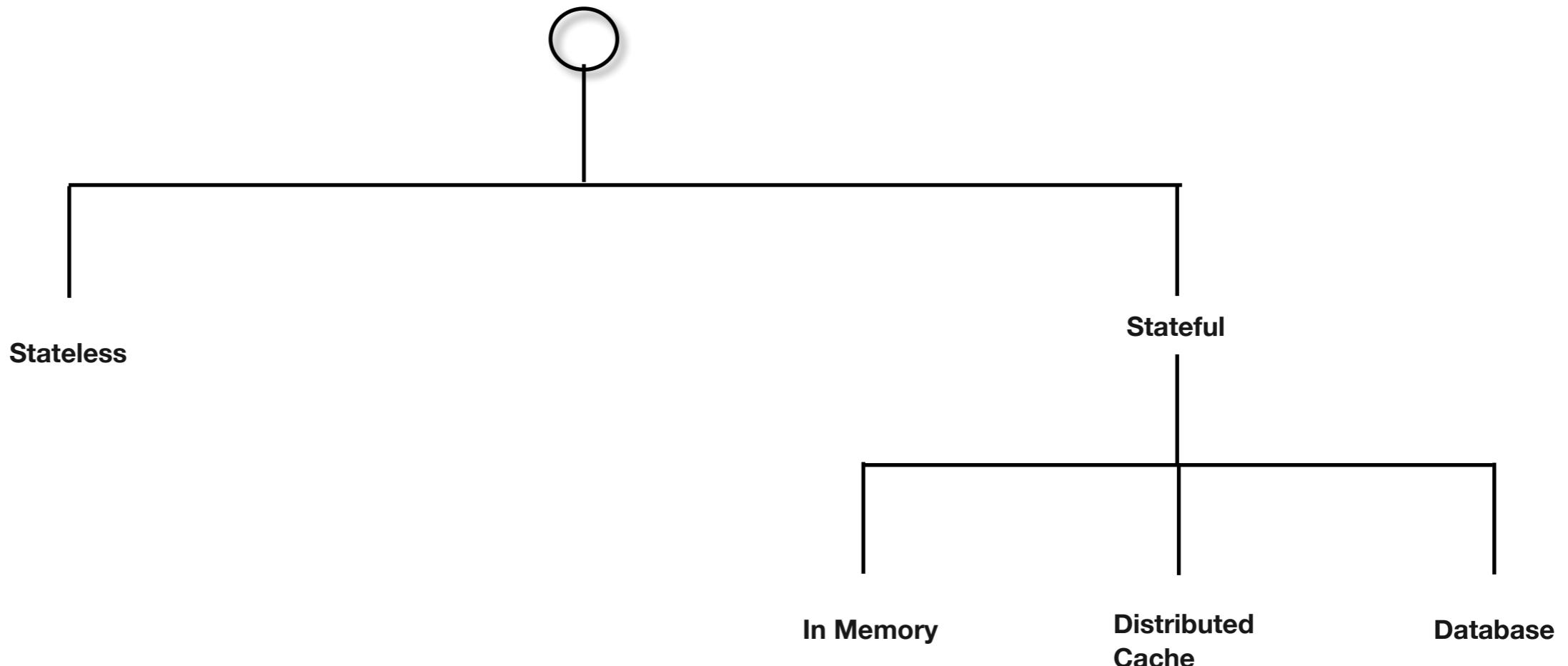
Activity

Resource

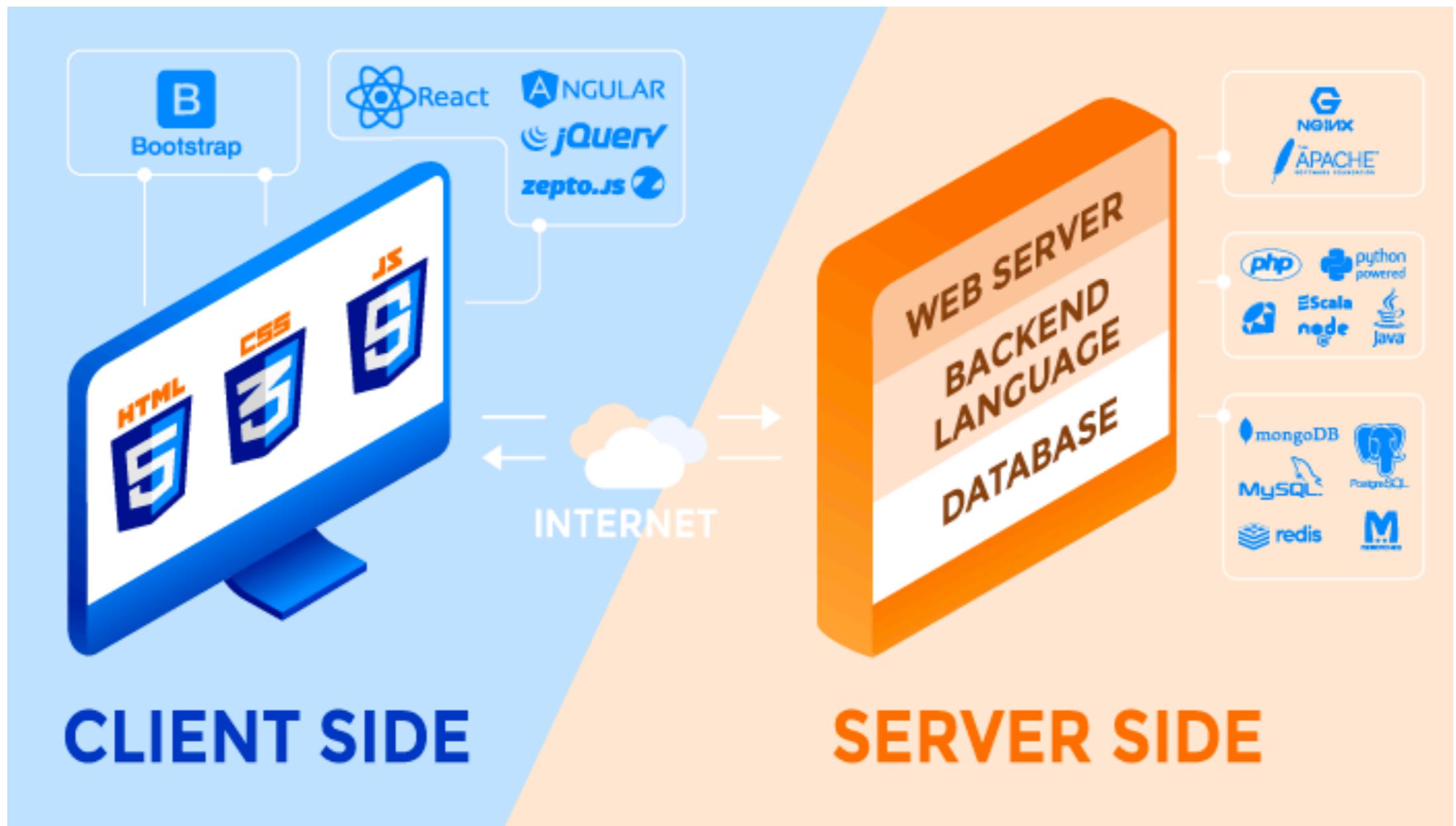
Config

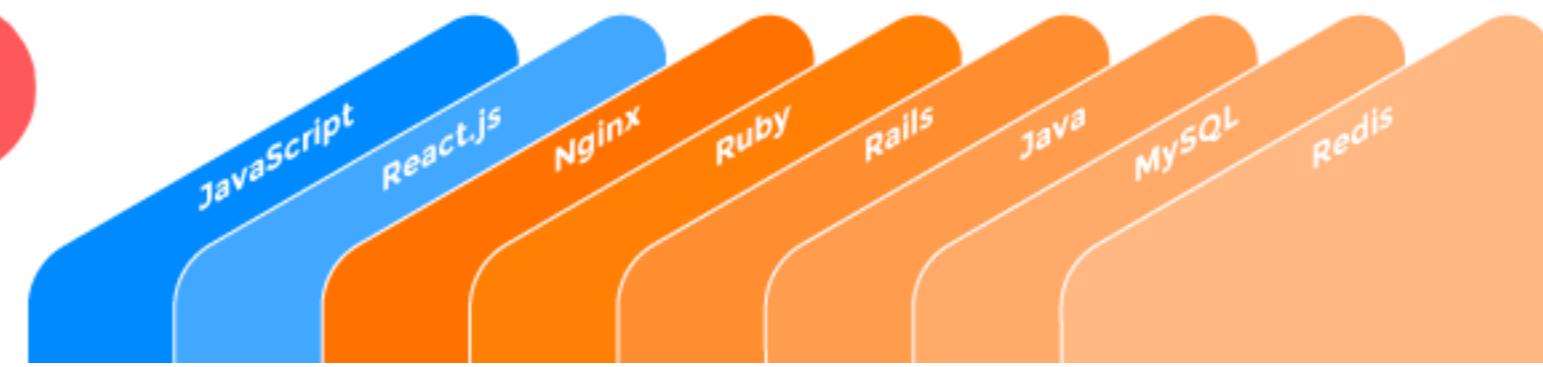


State Management

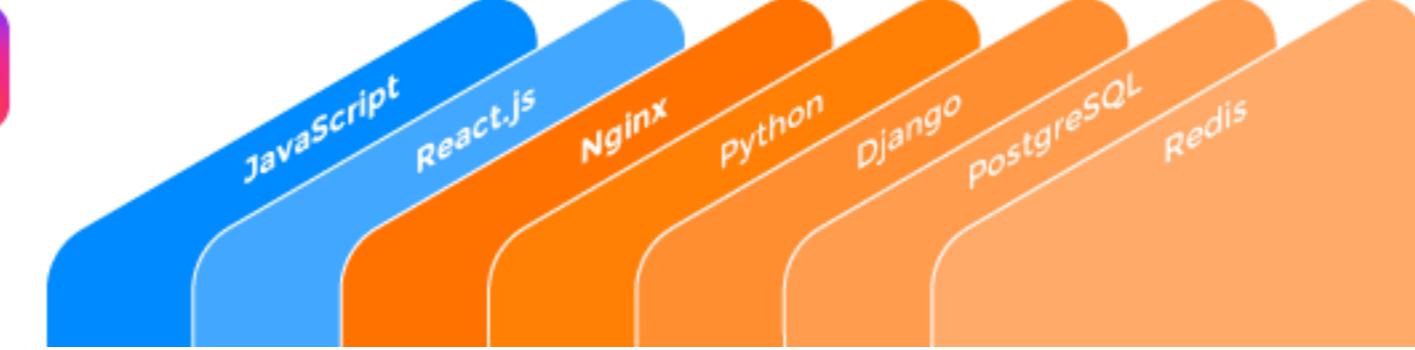
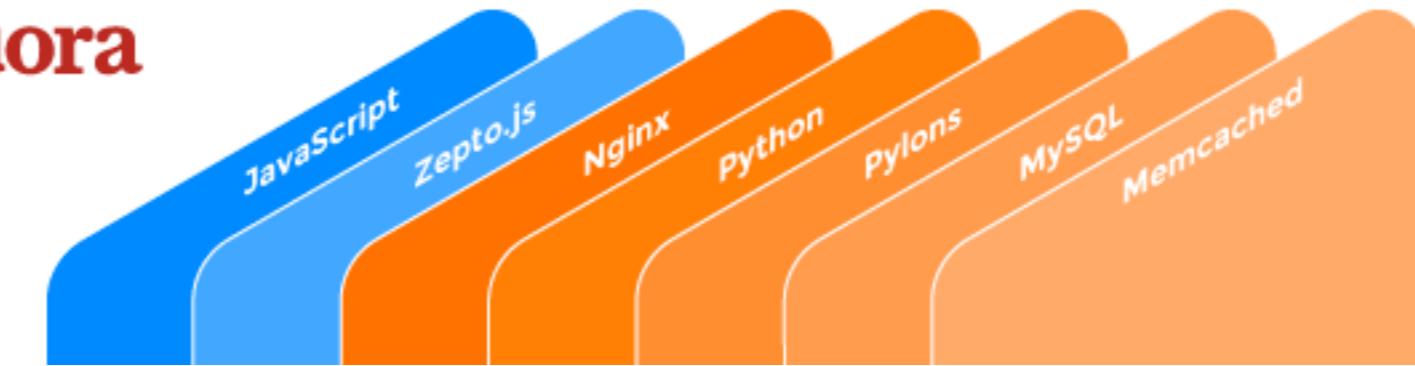


Technology choice

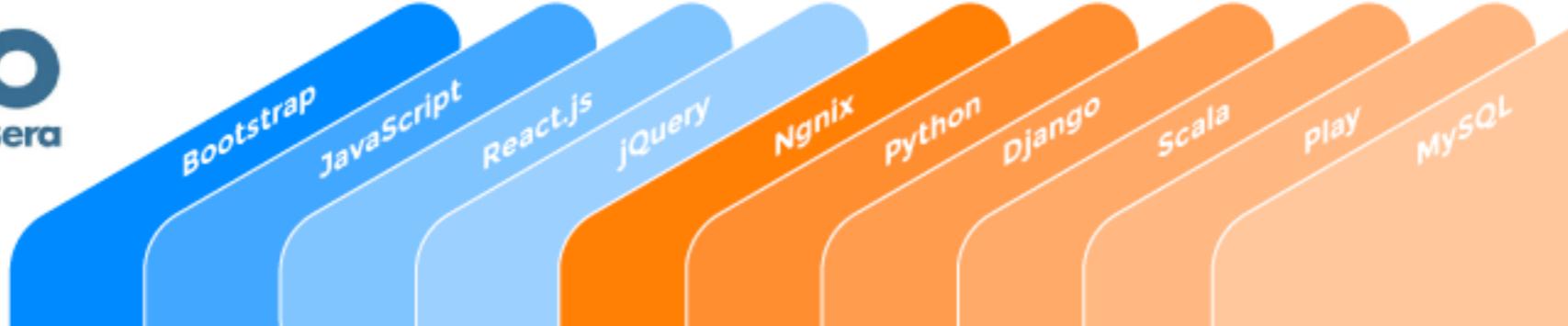




Quora

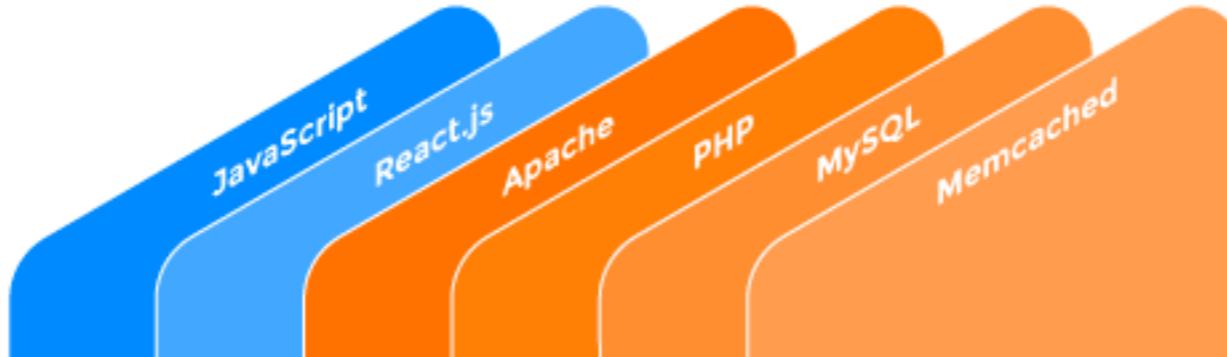
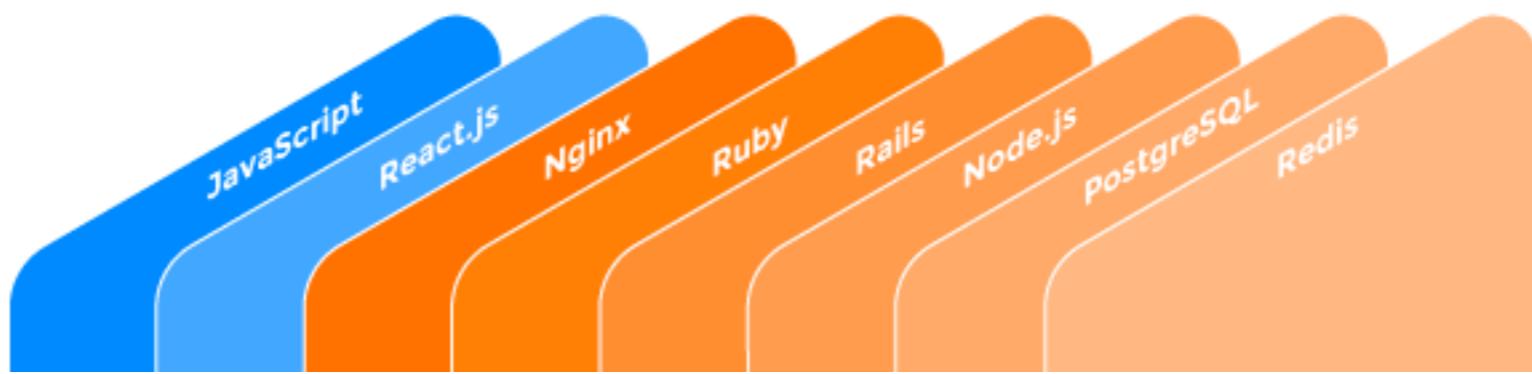


[Instagram](#)





Product Hunt



NUMBER OF CONTRIBUTORS ON GITHUB

JUNE 2017



AVERAGE DEVELOPER SALARIES IN THE US BY TECHNOLOGY



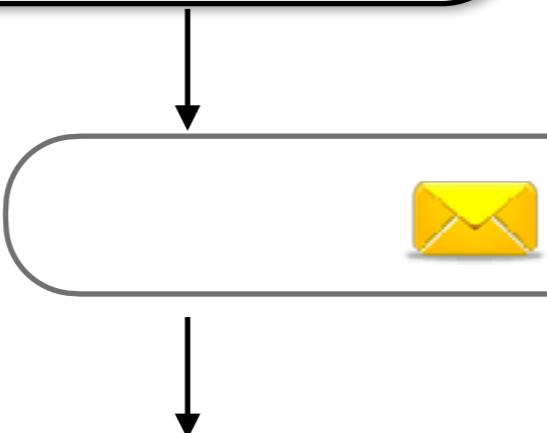
Source: Indeed.com

ToDo Portal
(Single Page Application)



Https(443)

ToDo API Service
(Api Application)



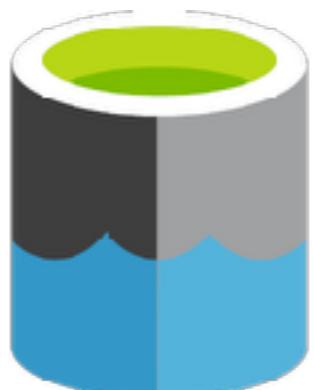
ToDo Calendar Service
(Background Application)



Message queue



ToDo Prediction Service
(Background Application)



Azure Data Lake

Budget

Libraries

Learning curve

Support

License

Expertise

Infra

Community

Security

Development time

Vendor

Application Security View



Application Security

Infra Security

AAA

- Authentication (first line of defence)
- Authorization
- Audit (last defence) who did , what
- Input validation
- Asset Handling (in rest, in transit)
- Exception Handling
- Session Handling
- Key Management
- Library, web server, OS, db server vulnerabilities
- Docker base image
- ..

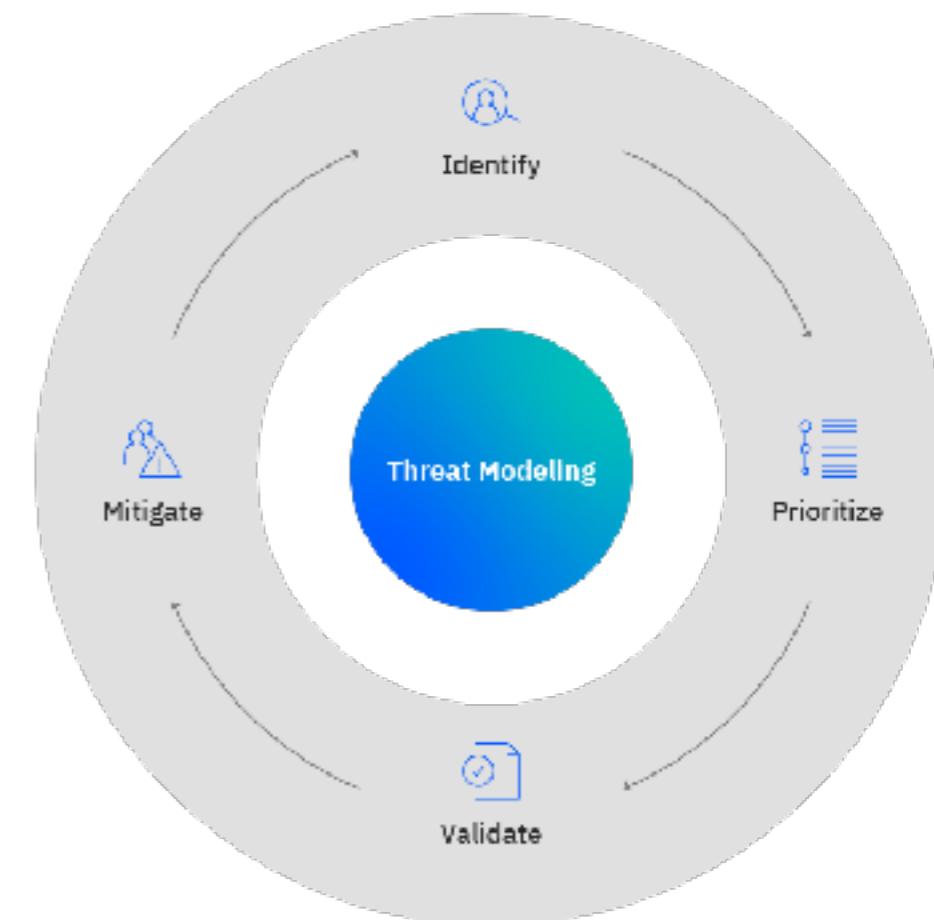
- Authentication (first line of defence)
- Authorization
- Audit (last defence) who did , what
- Input validation (fwk)
- Asset Handling (in rest, in transit)
- Exception Handling (fwk)
- Session Handling (fwk)
- Key Management

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



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

STRIDE

Spoofing

Authentication

Elevation of Privilege

Session Handling

Tampering

Input Validation

Repudation

Audit

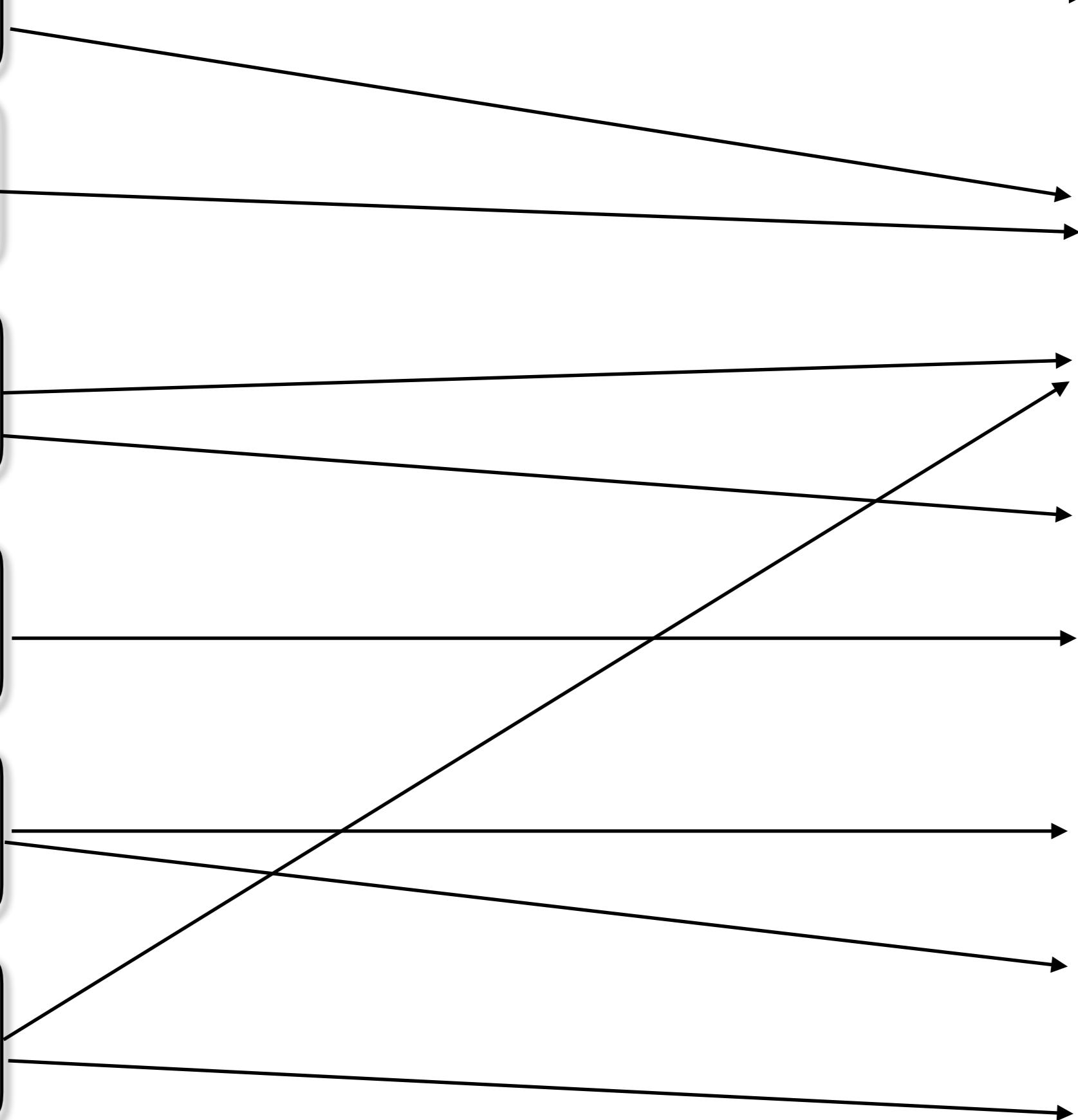
Information Disclosure

Exception Handling

Denial of Service

Confidentiality

Availability



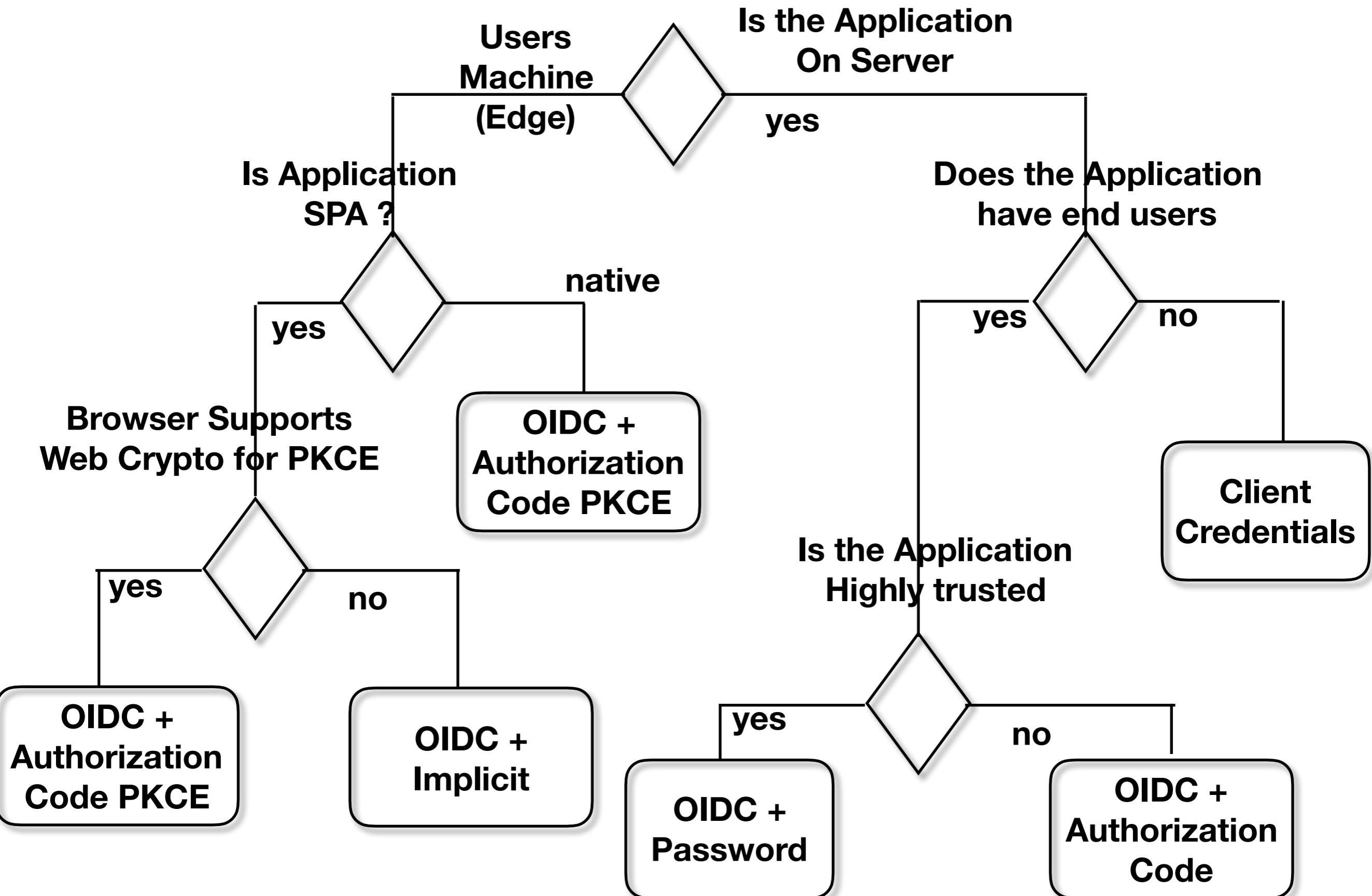
App Concerns

- 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

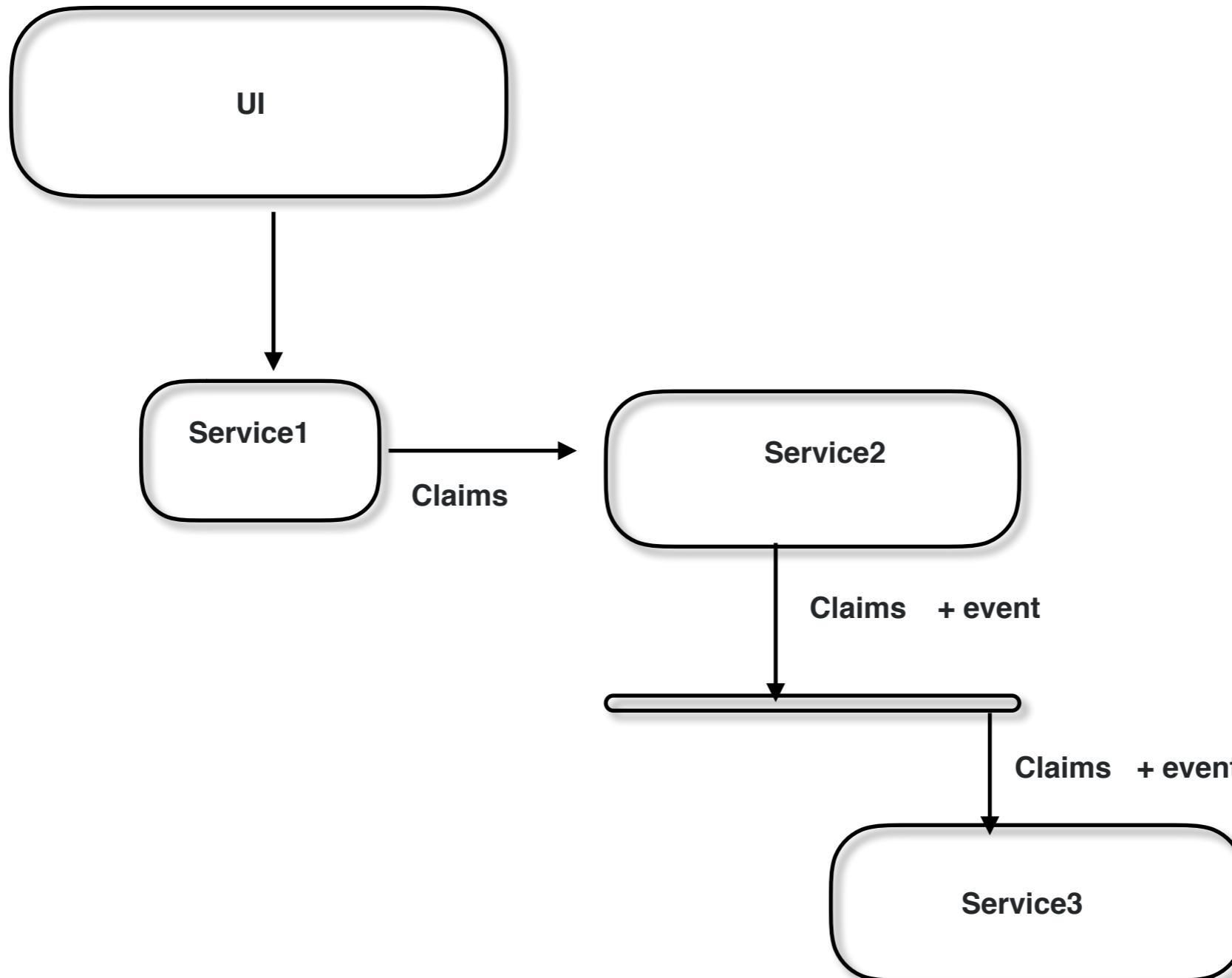
Service2

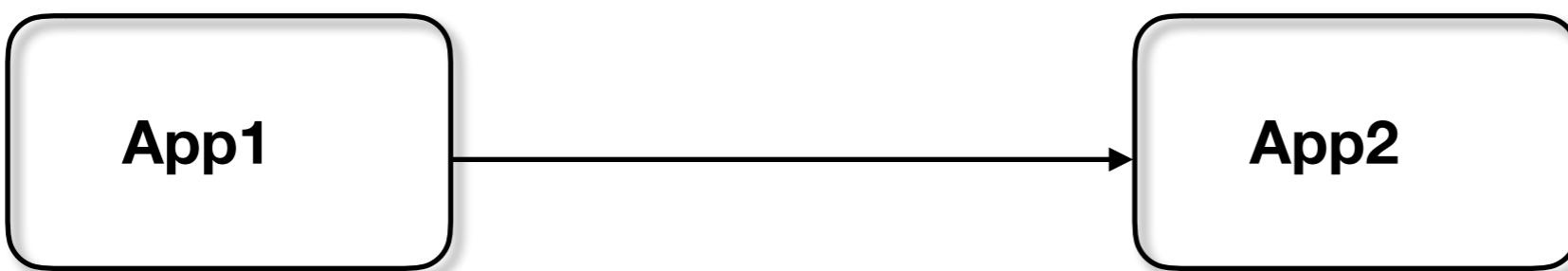
Claims

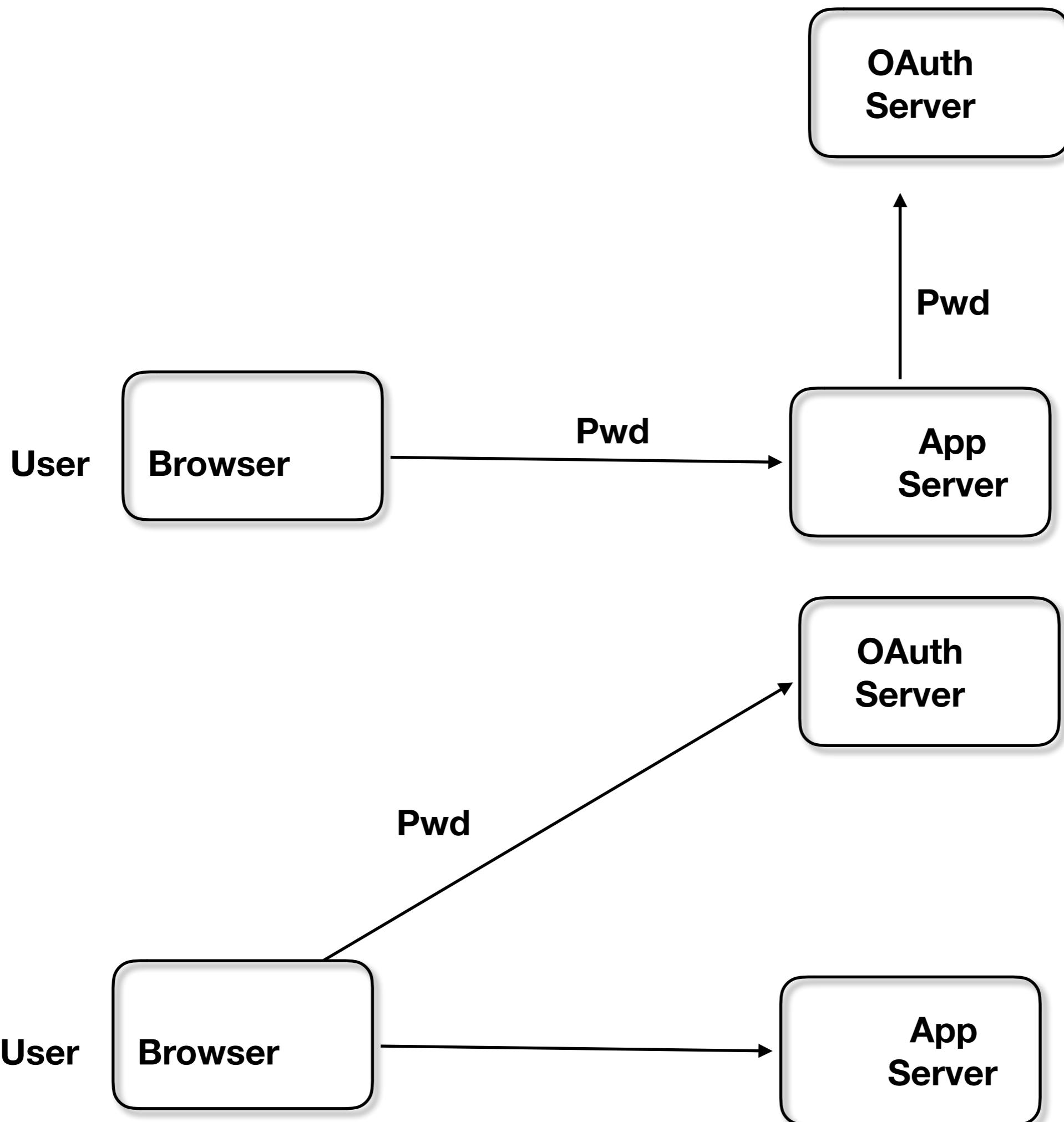
Claims + event

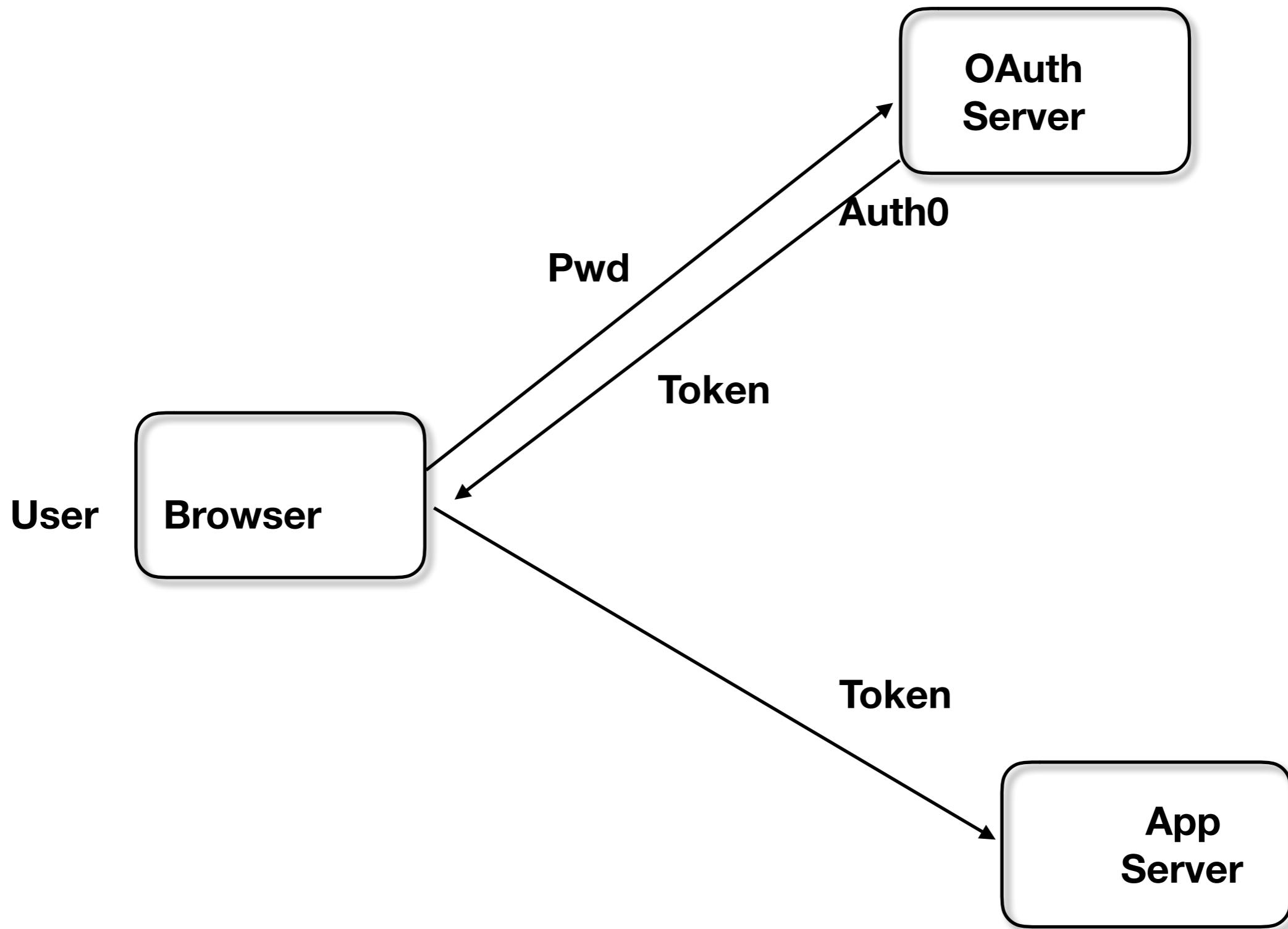
Claims + event

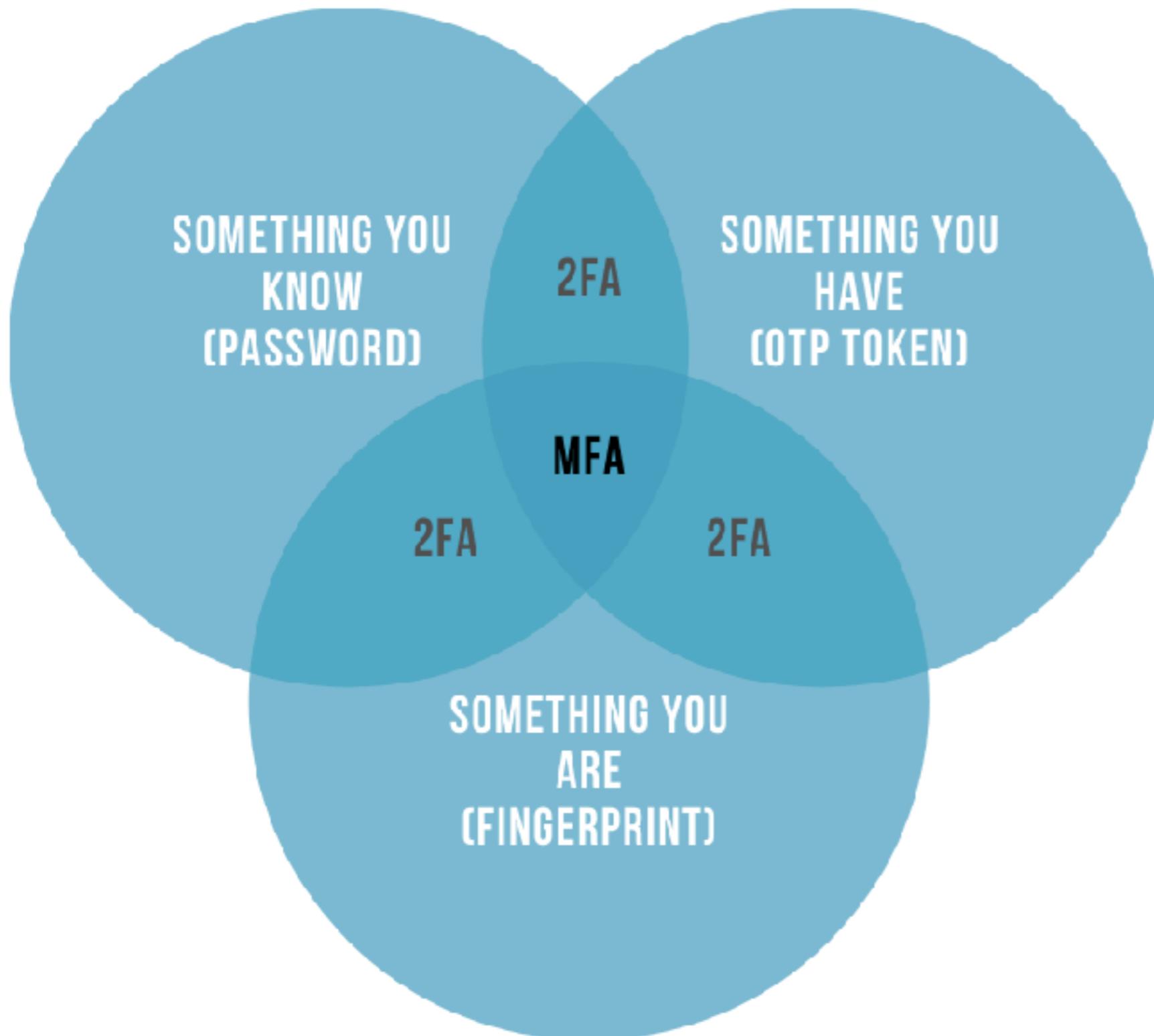
Service3





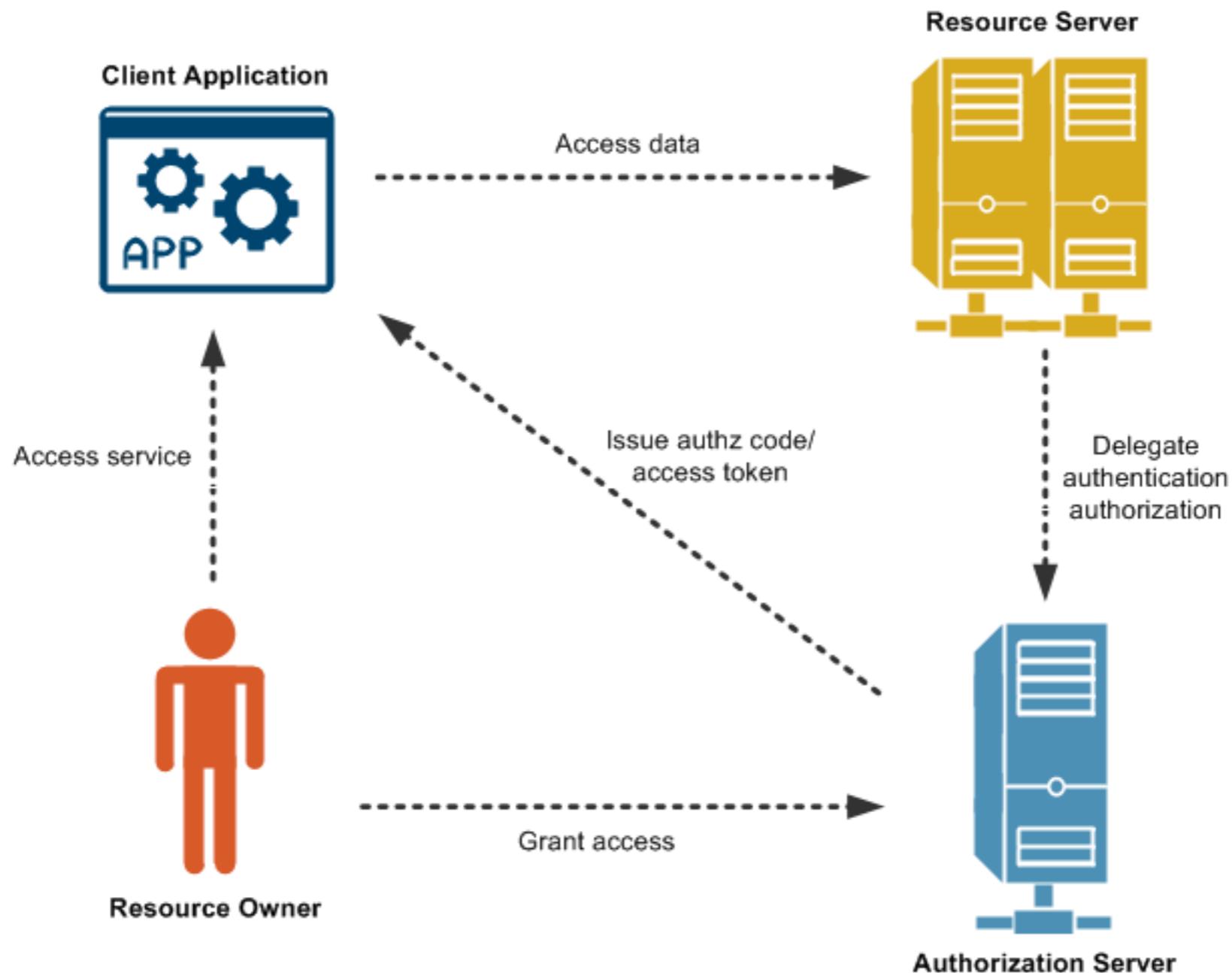




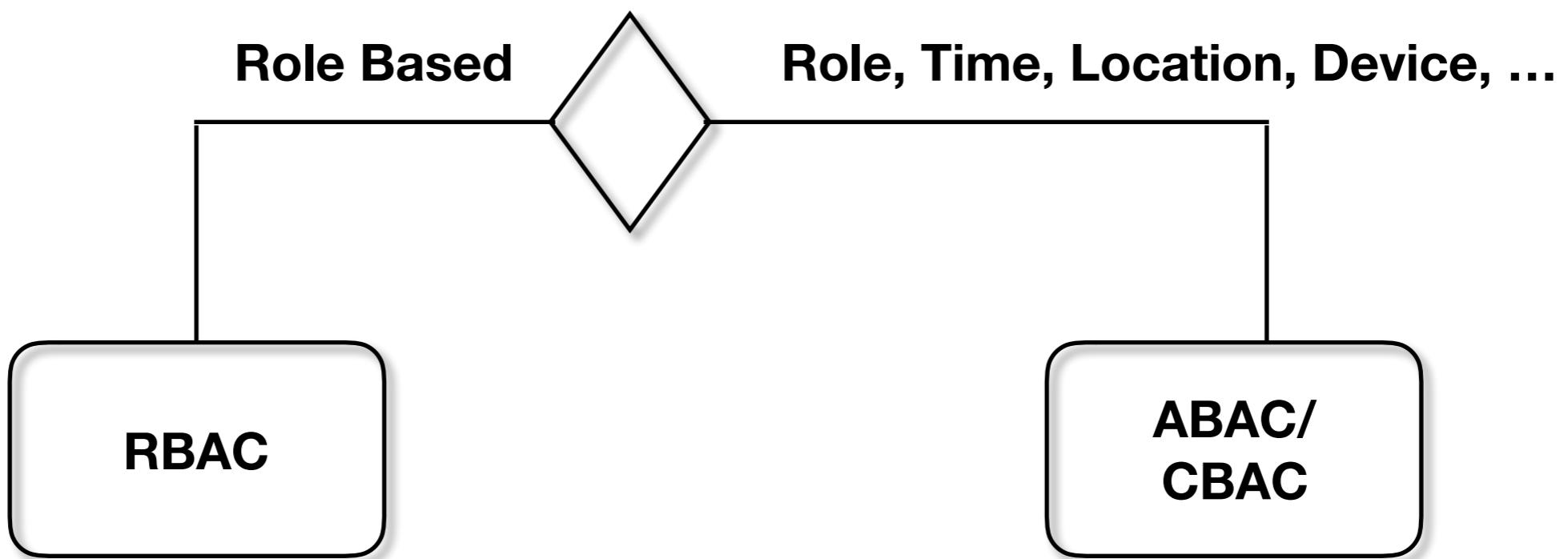


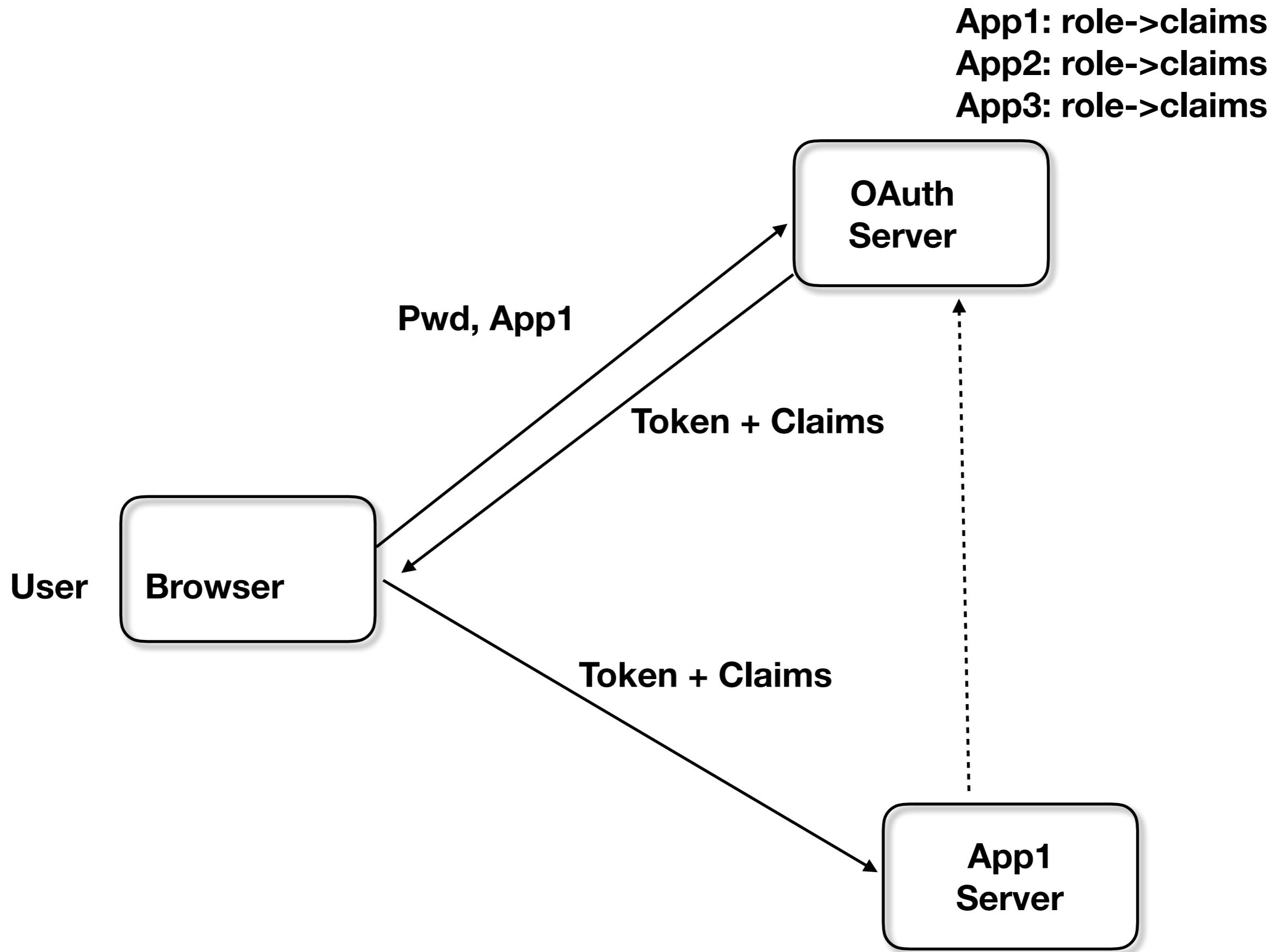
Something you know - pwd(oauth2) , security question <– 1
Something you Have - otp, cert, rsa, email, <– 2
Where you are - country, office, ... <– 3
Something you are - voice, face, dna, ..
Something you do - behaviour

Centralize

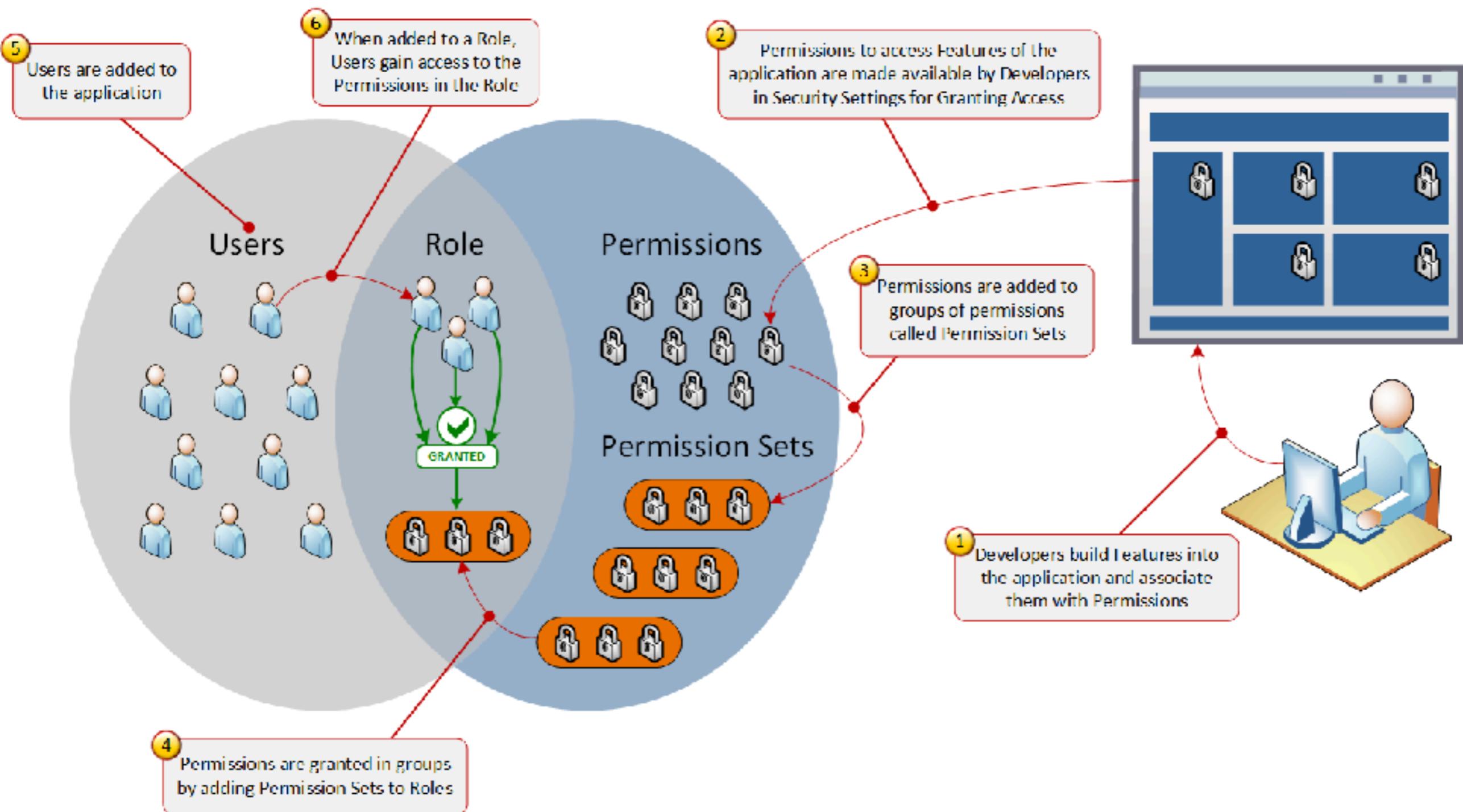


Step 2. Authorization

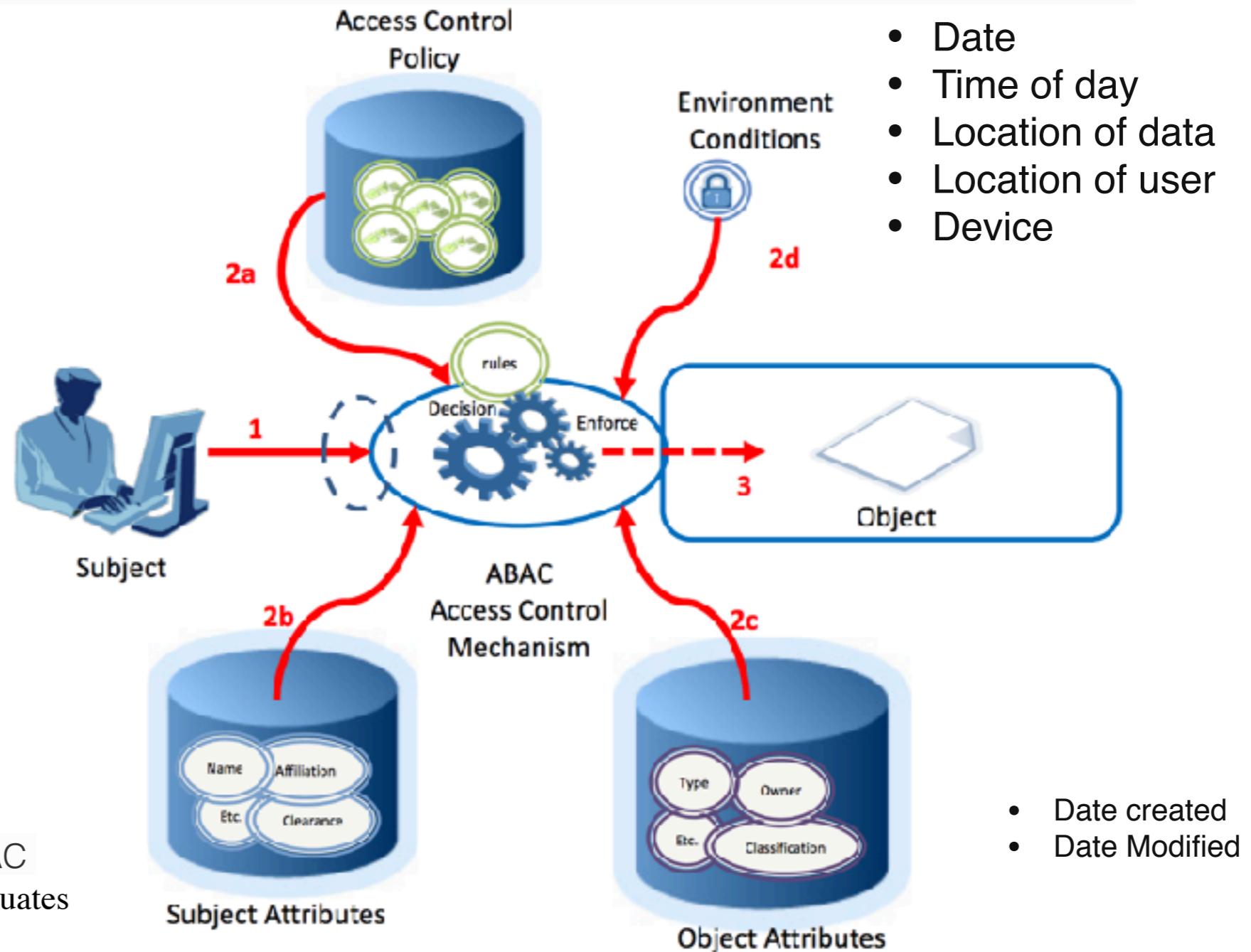




Role Based Access Control (RBAC)

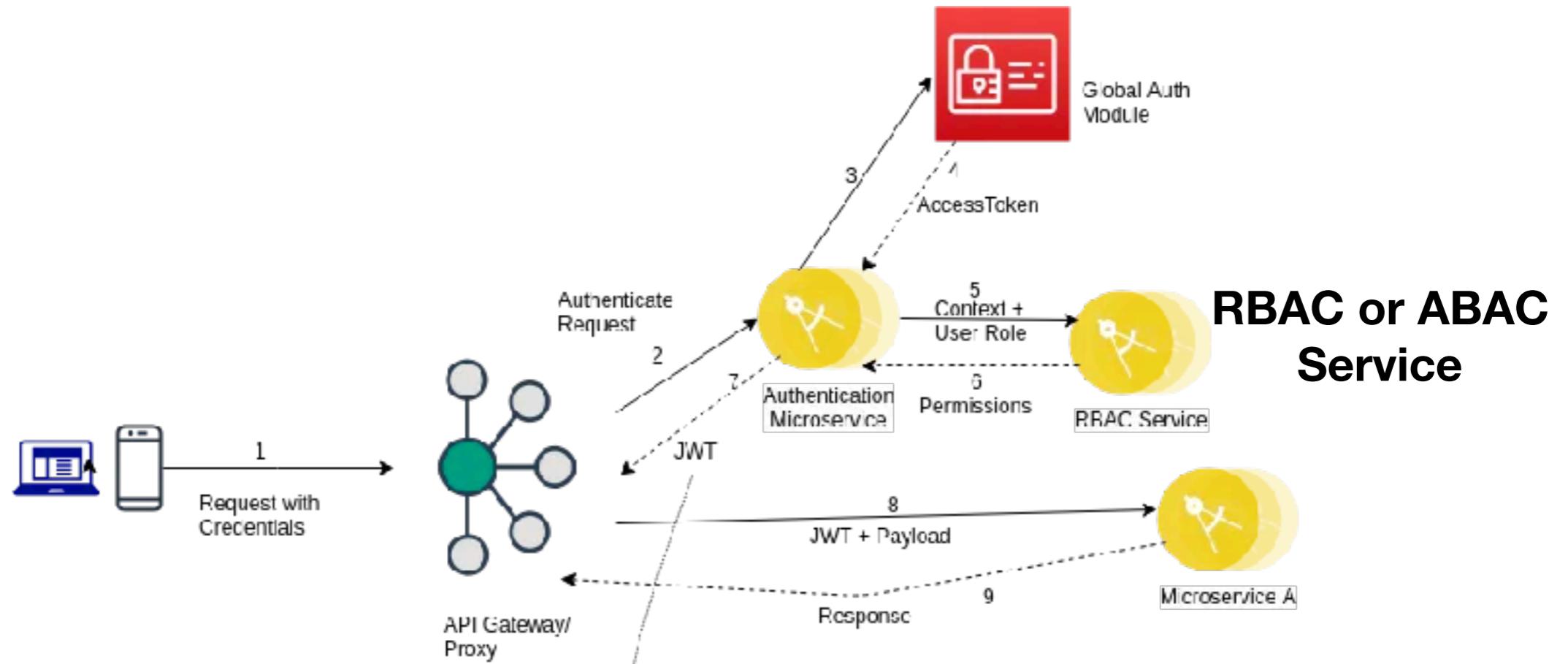


Attribute Based Access Control (ABAC) / Claim Based Access Control (CBAC)



ABAC is an extension of RBAC
Access Control Mechanism evaluates
a) Rules (RBAC),
b) Subject Attributes,
c) Object Attributes
d) Environment Conditions
to compute a decision

- Date created
- Date Modified



```
{
  "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



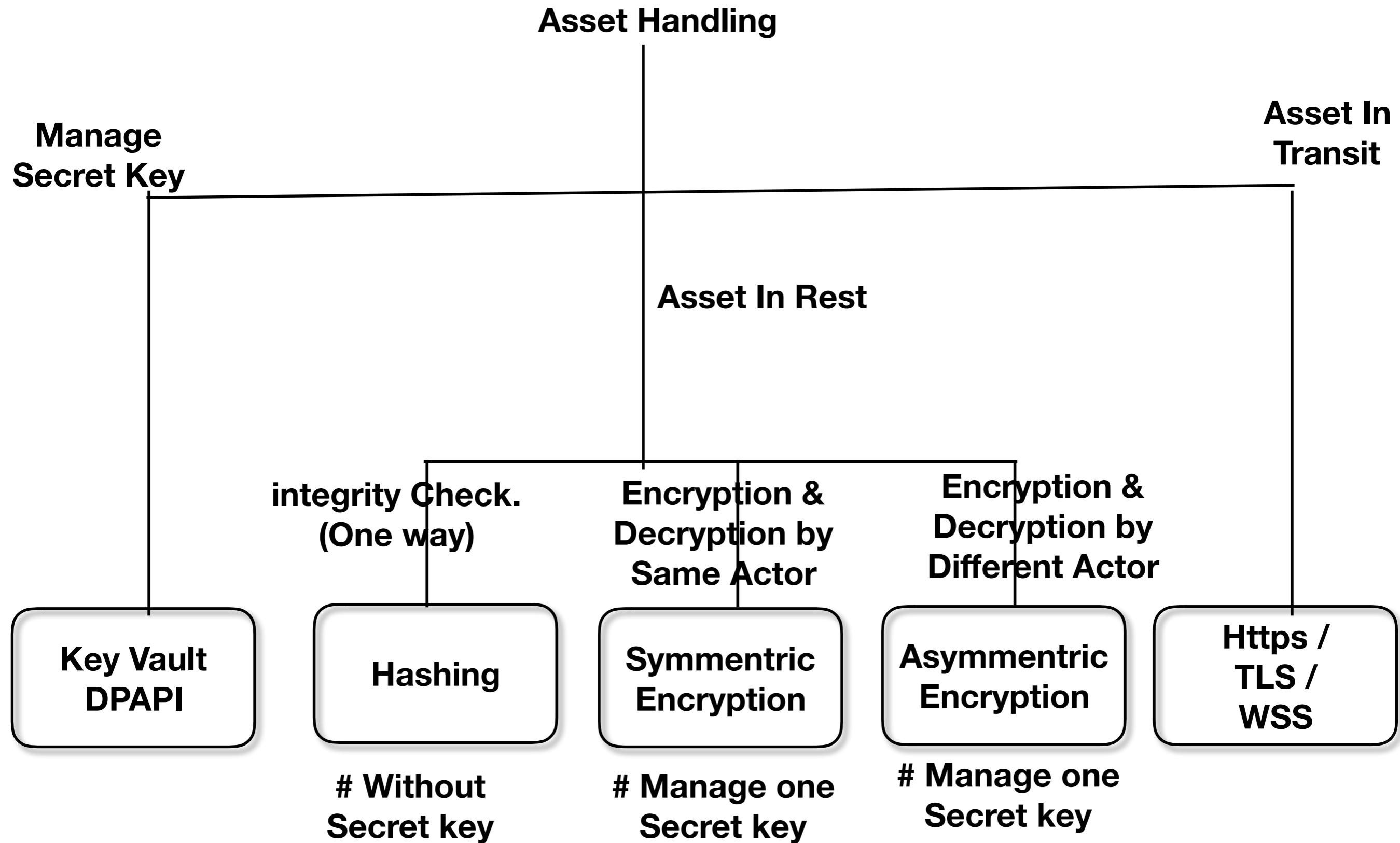
who - Identity

what - Action

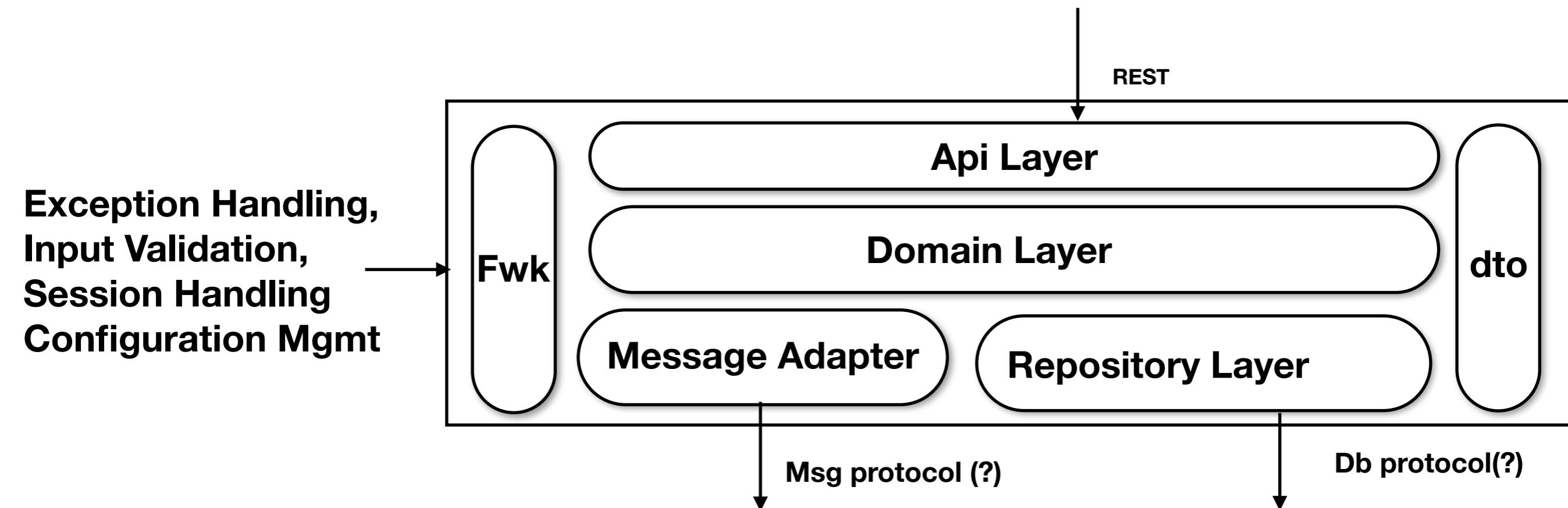
where - Component/Service/Object/Method

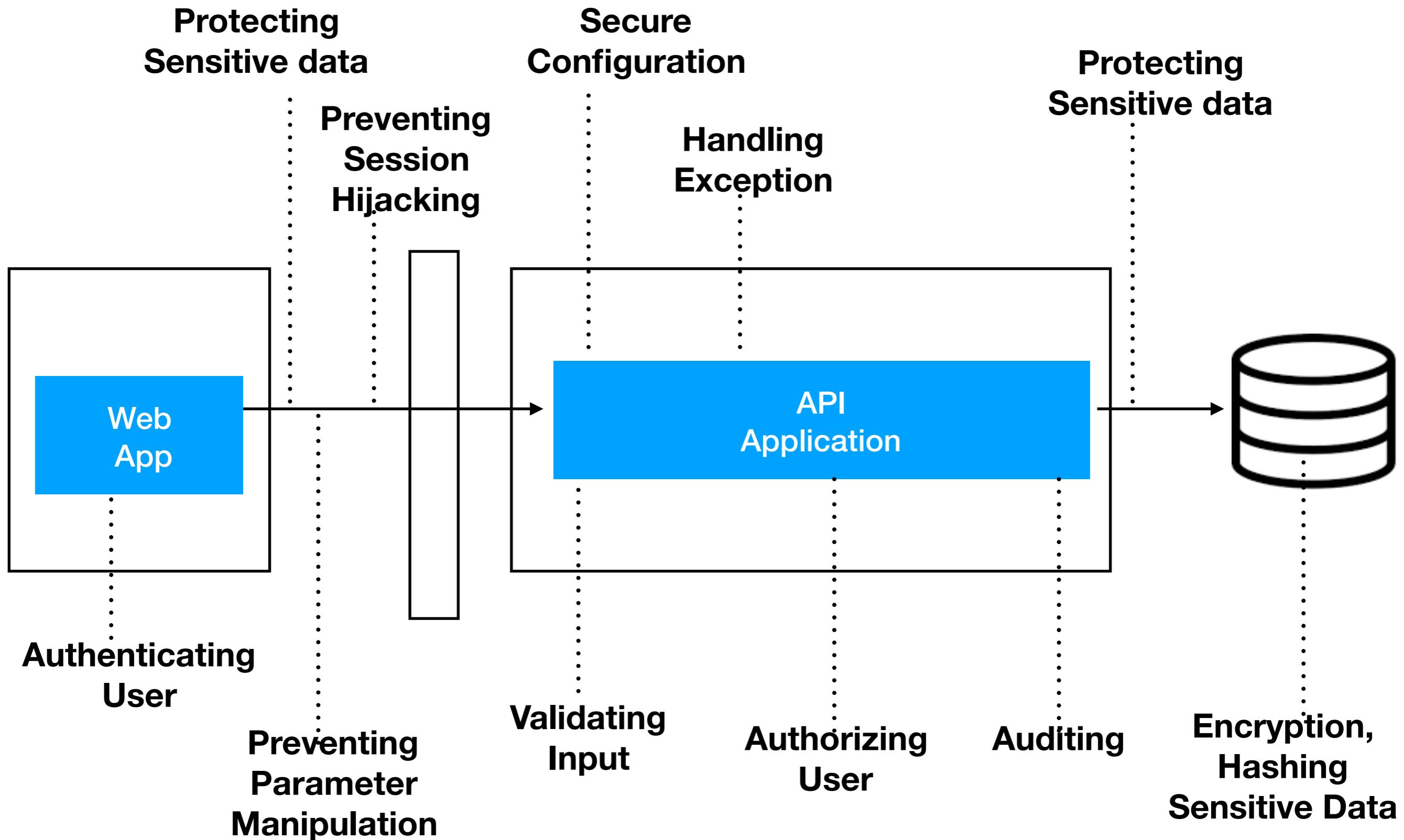
when - Timestamp:

Step 4. Privacy and Confidentiality



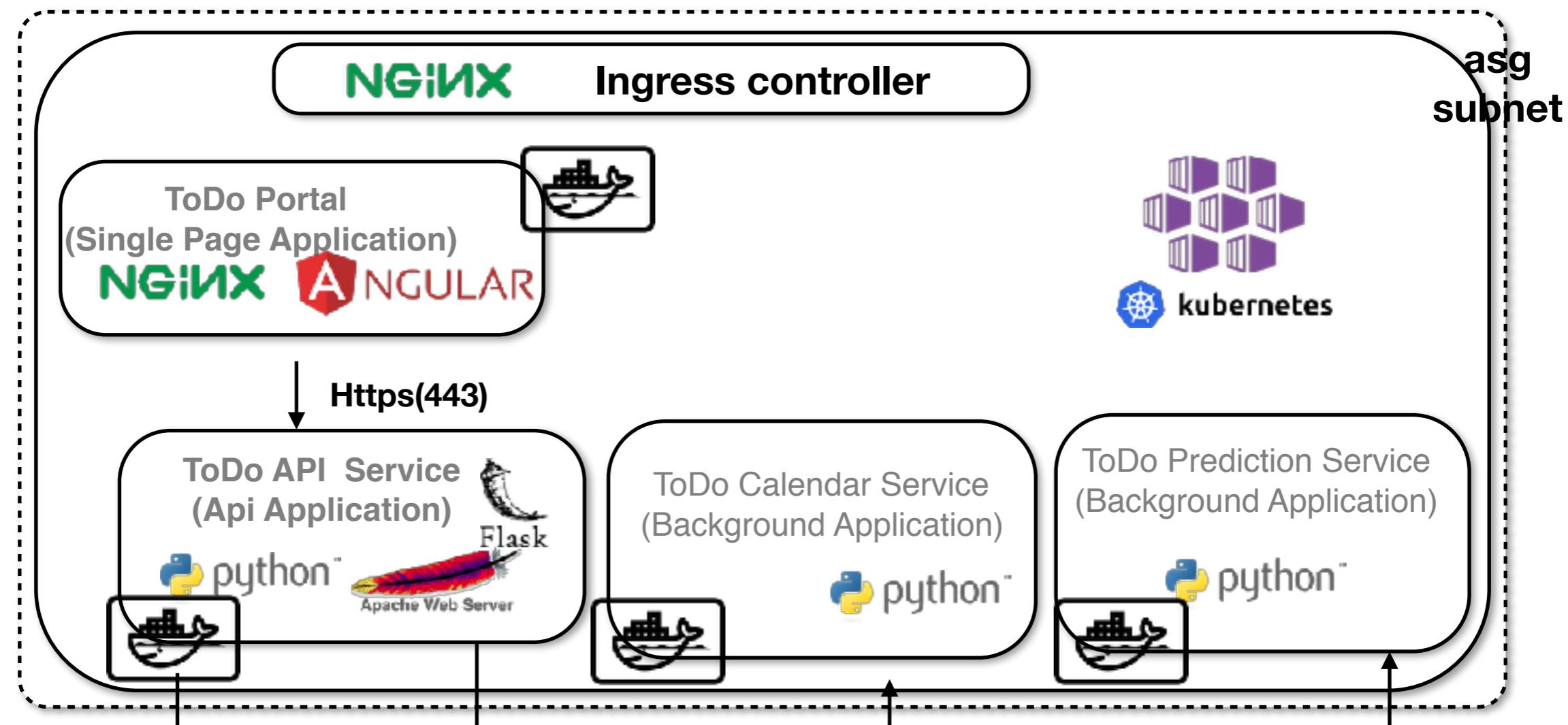
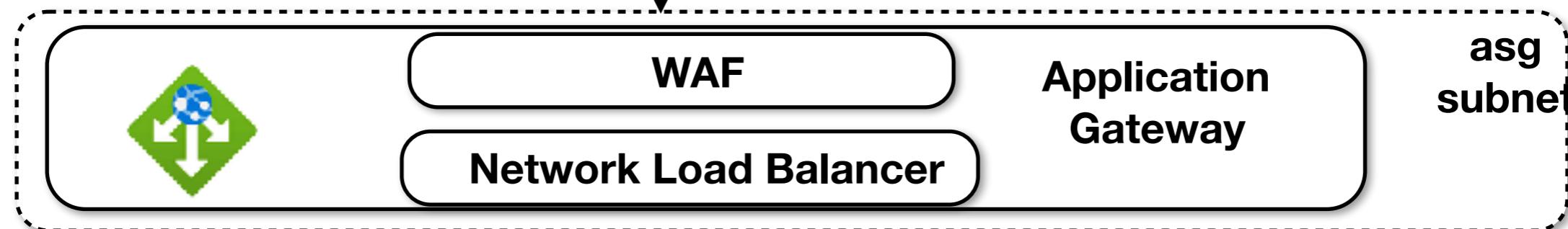
Step 5. Cross cutting concerns





vnet

Public IP address



mongoDB
Atlas



mongoDB



Message queue

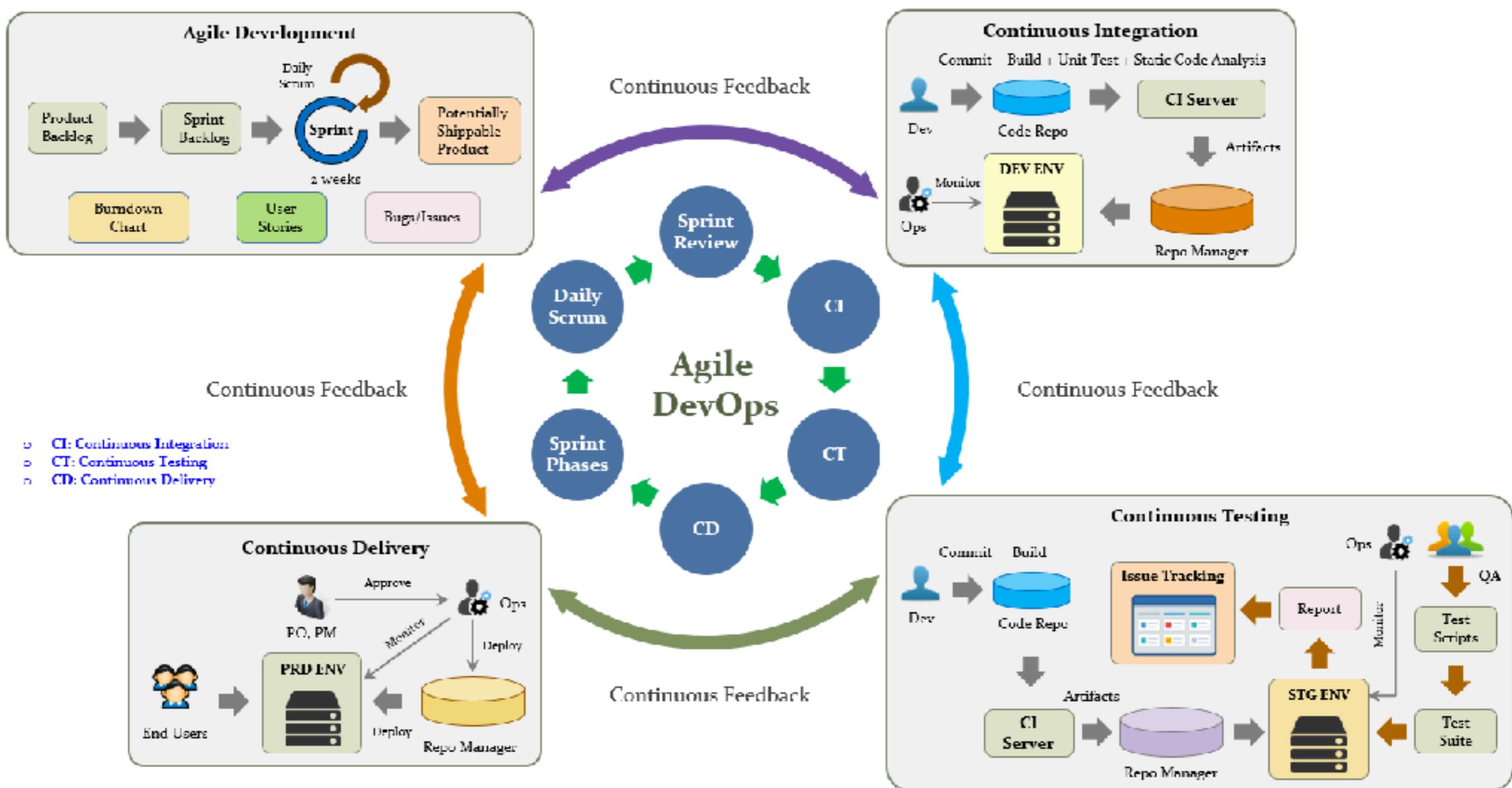


Azure Queue

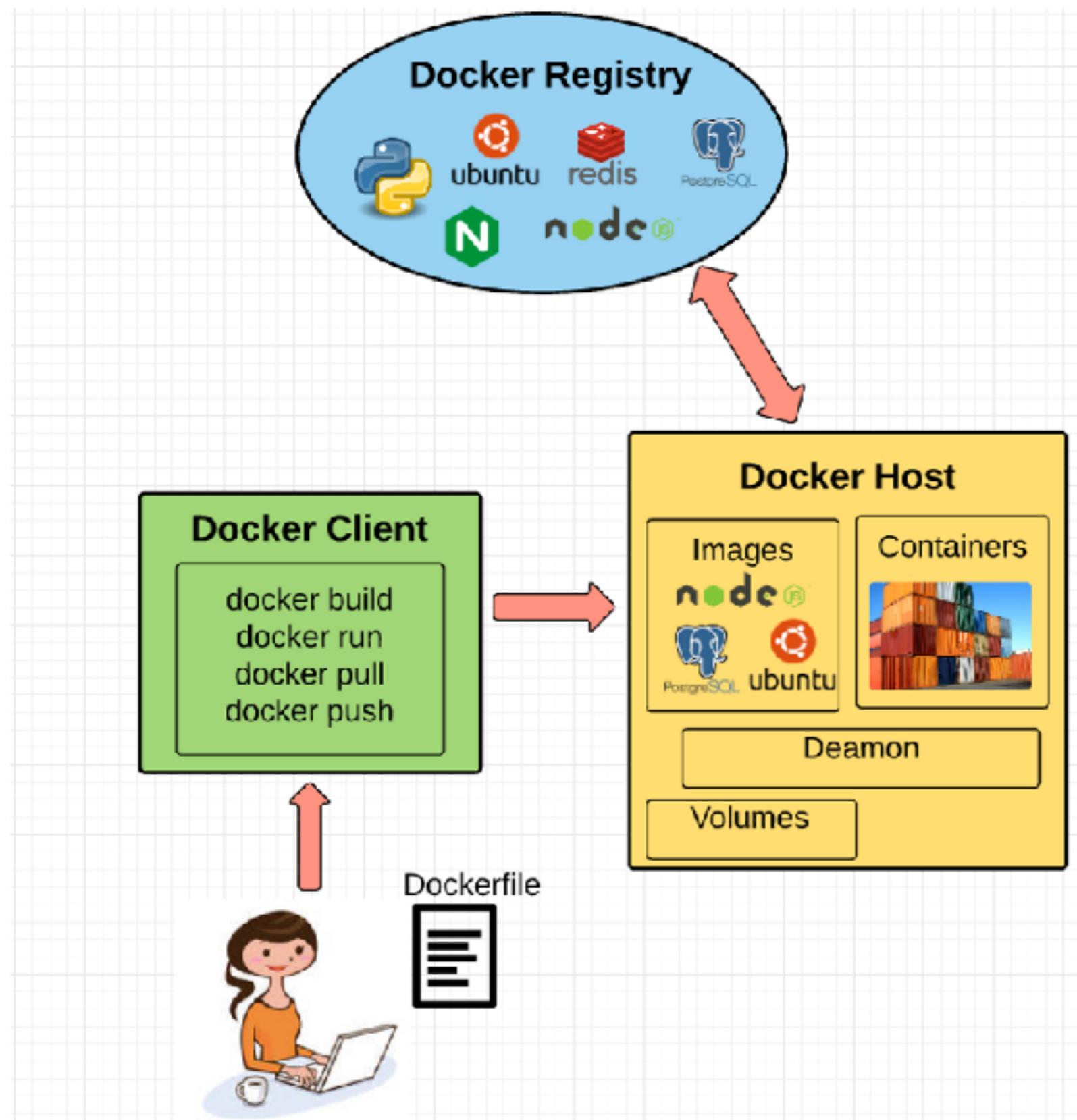


Azure Data Lake

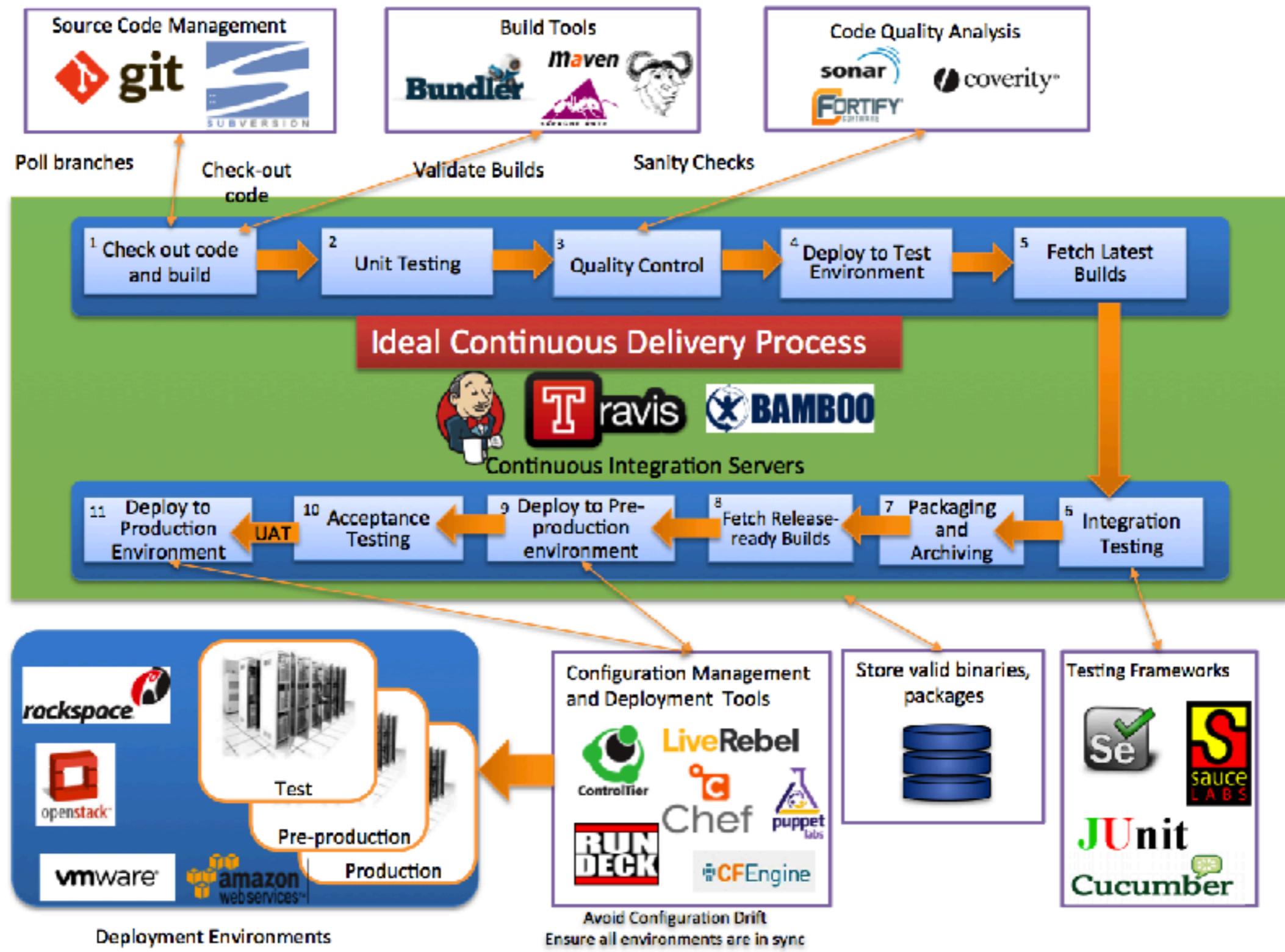
DevOps View



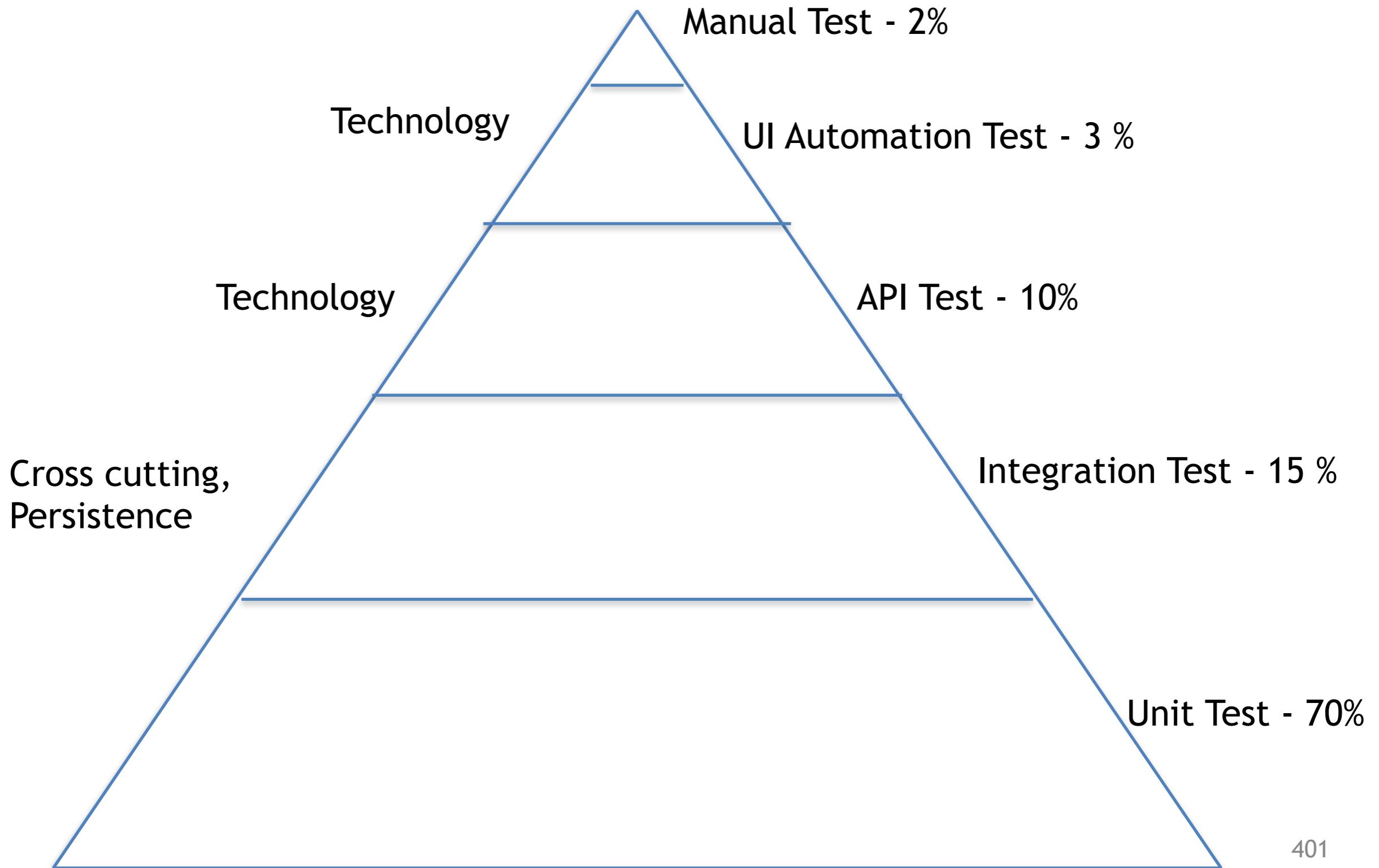
Step 1. Containerizing



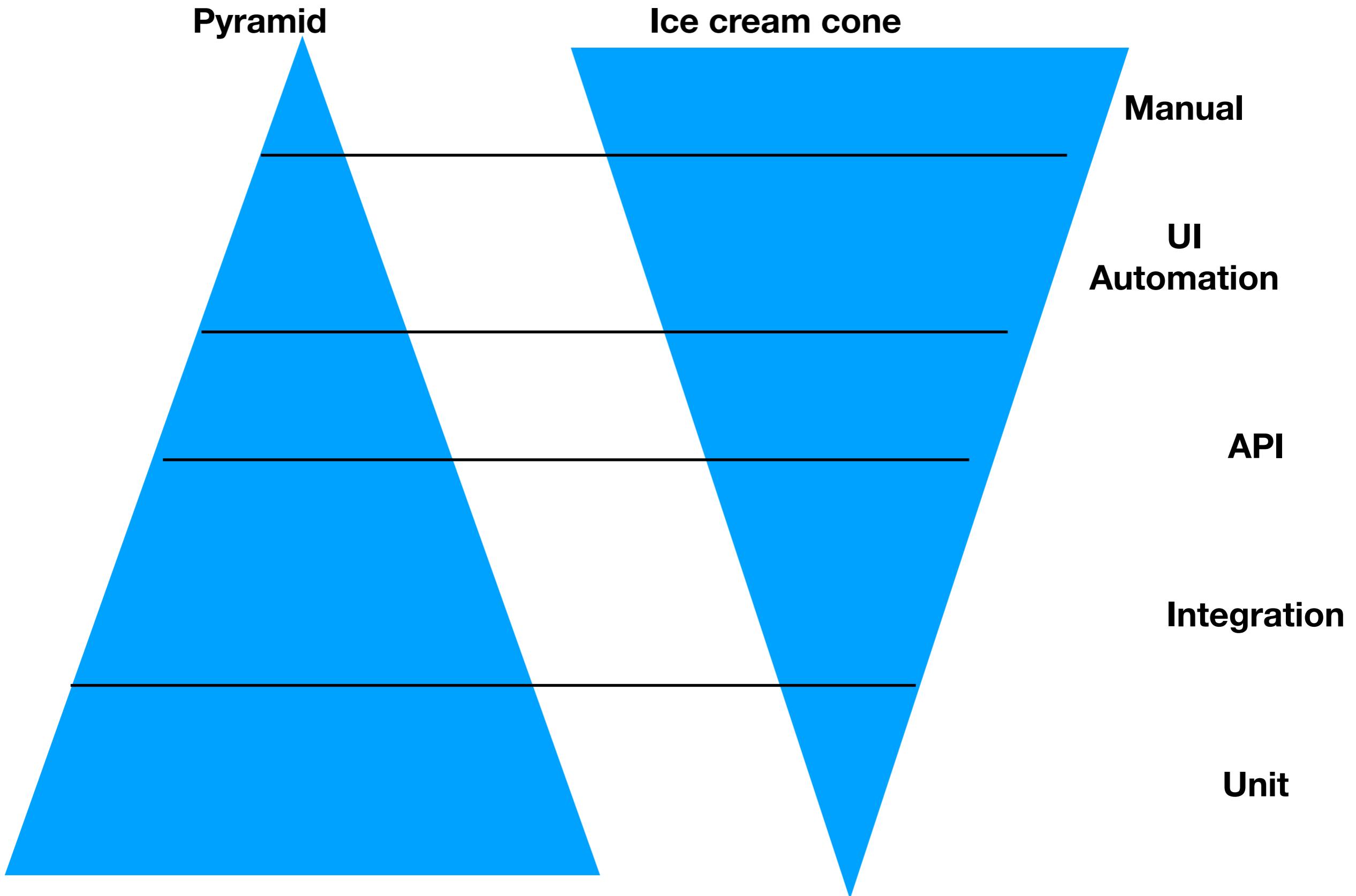
Step 2. Integrating infrastructure automation with CI/CD

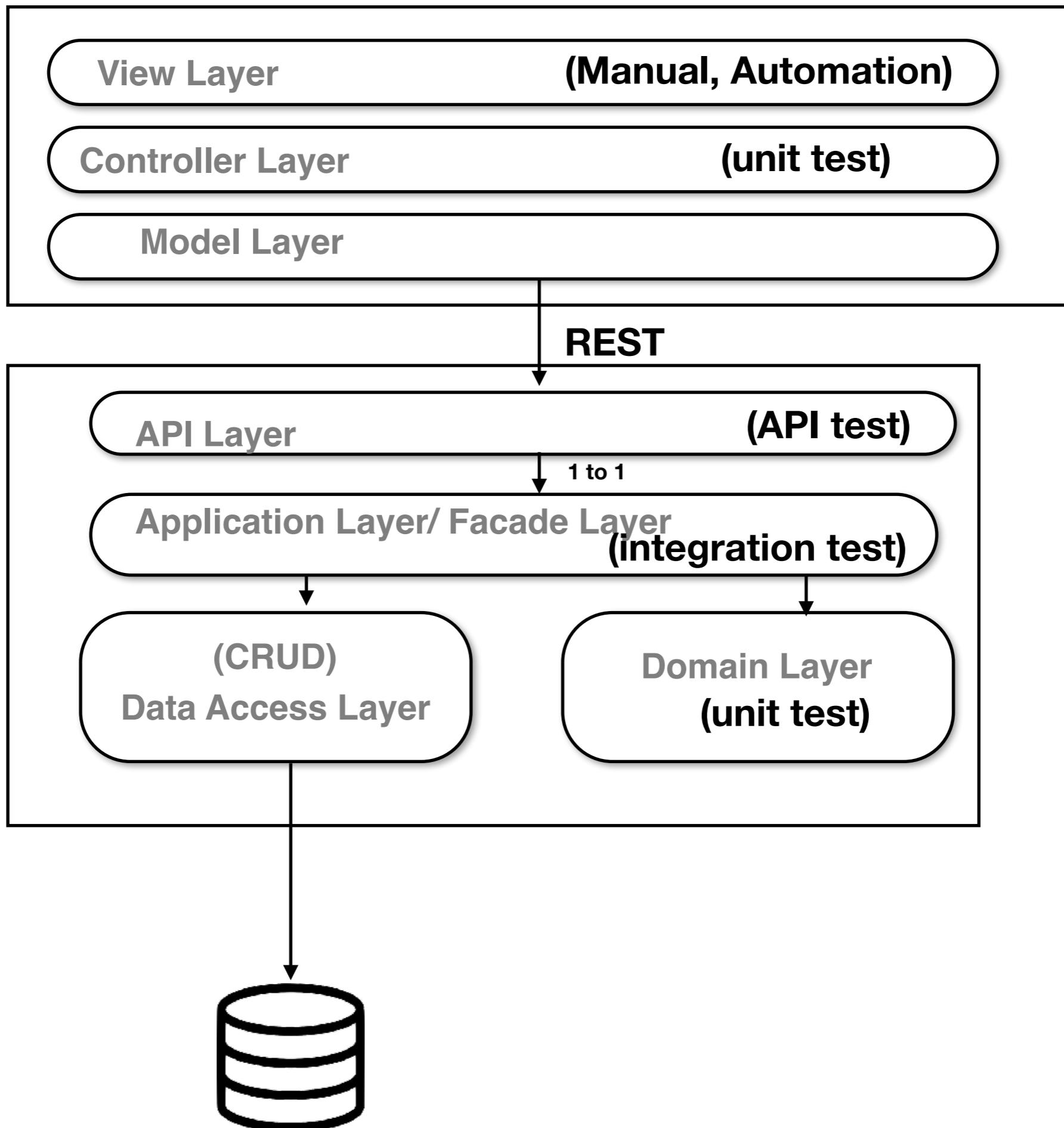


Step 3. test automation and aligning QA with development

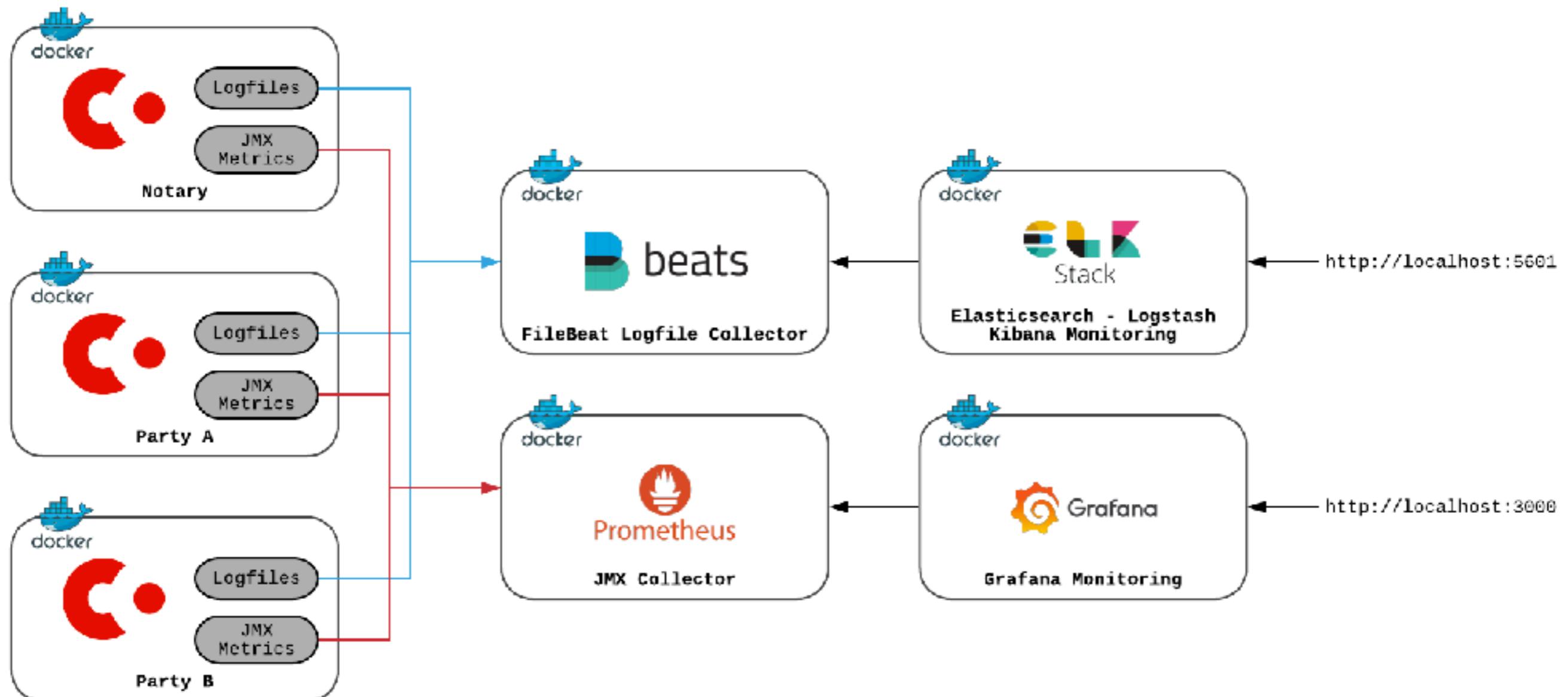


Pyramid Test

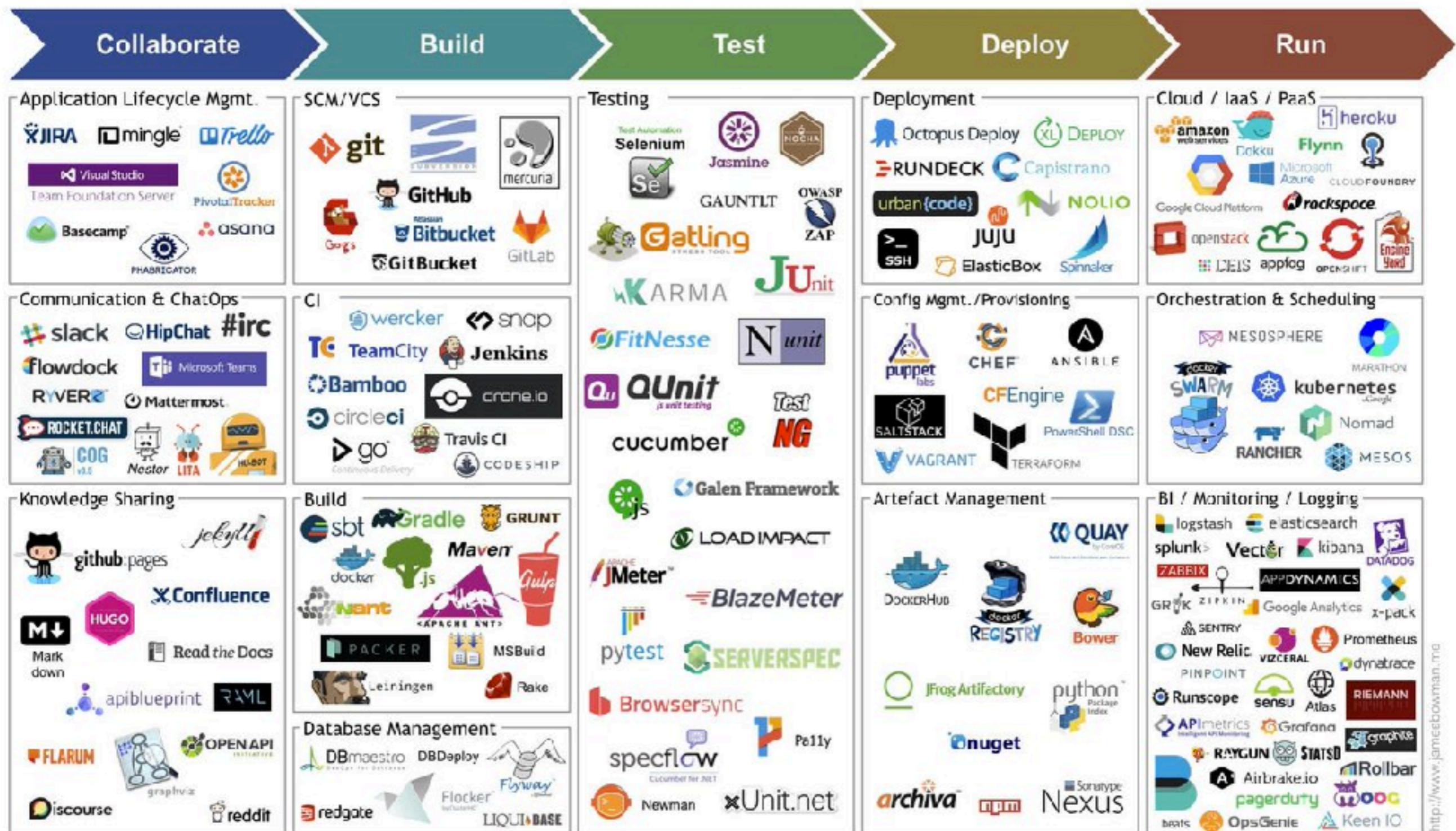




Step 4. Setup Monitoring

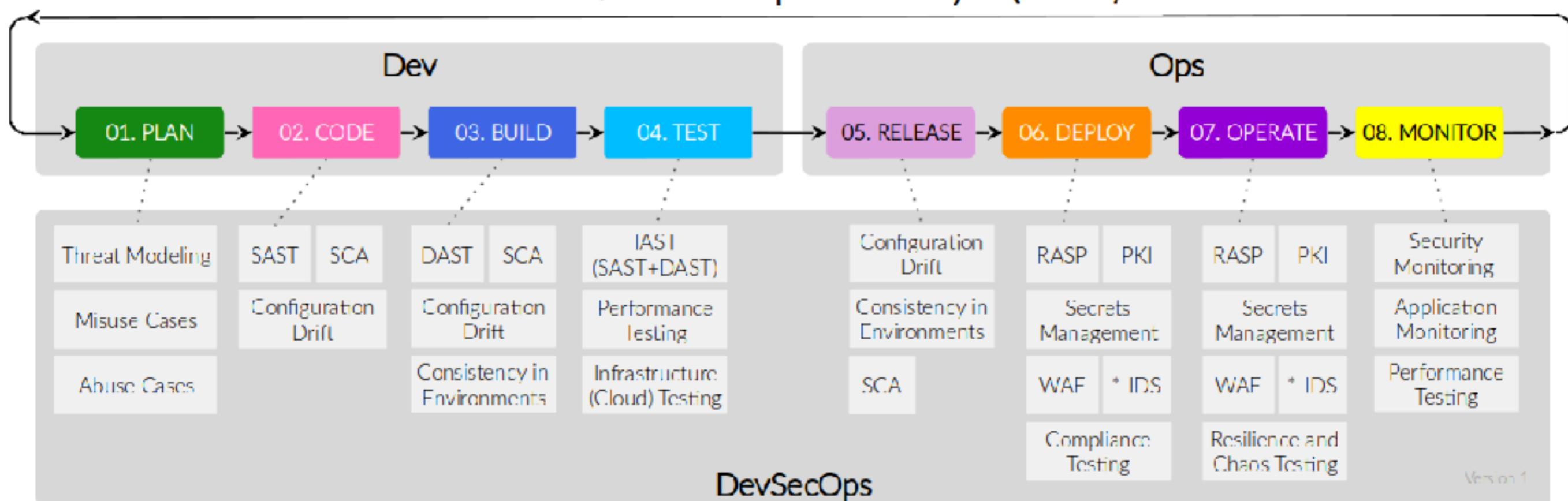


Reference



Step 5. SecOps

Secure Software Development Life Cycle (SSDLC)



DAST vs. SAST

Vulnerability Coverage

SAST

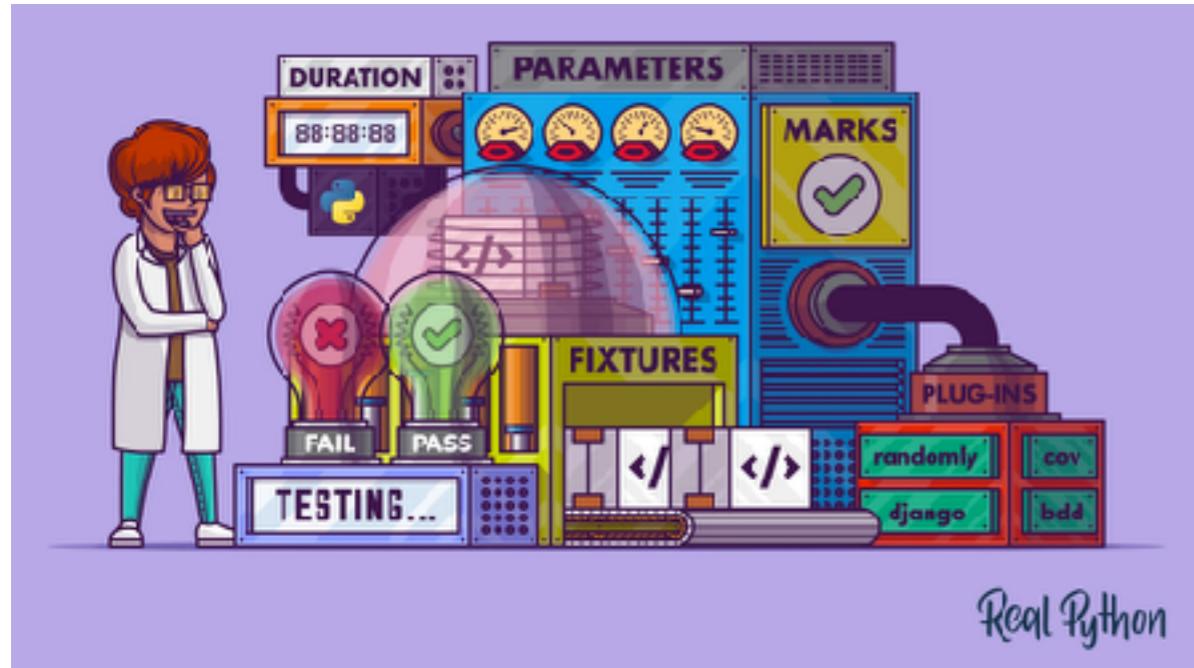
- + Null Pointer Dereference
- + Threading Issues
- + Code Quality Issues
- + Insecure Crypto Issues
- + Issues in Non Web application Code
- Higher number of FP
- Run time Code generation
- Dynamic Languages (Ruby + Python)

DAST

- + SQL Injection
- + Cross Site Scripting (XSS)
- + OS Commanding
- + HTTP Response Splitting
- + LDAP Injection
- + XPATH Injection
- + Path Traversal
- + Buffer Overflows
- + Format String Issues

- + Runtime Privilege Issues
- + Authentication Issues
- + Session Management Issues
- + Insecure 3rd Party Libraries
- + Business Logic Vulnerabilities
- + Protocol Parser Issues
- Web2.0, JSON, Flash, HTML 5.0,
- Integrity and Availability violations
- Long Execution Times

IAST: Interactive application security testing. Monitoring an application for security vulnerabilities while it is running — at testing time.



RASP: Runtime application self protection. Monitoring an application to detect attacks while it is running — at production time.

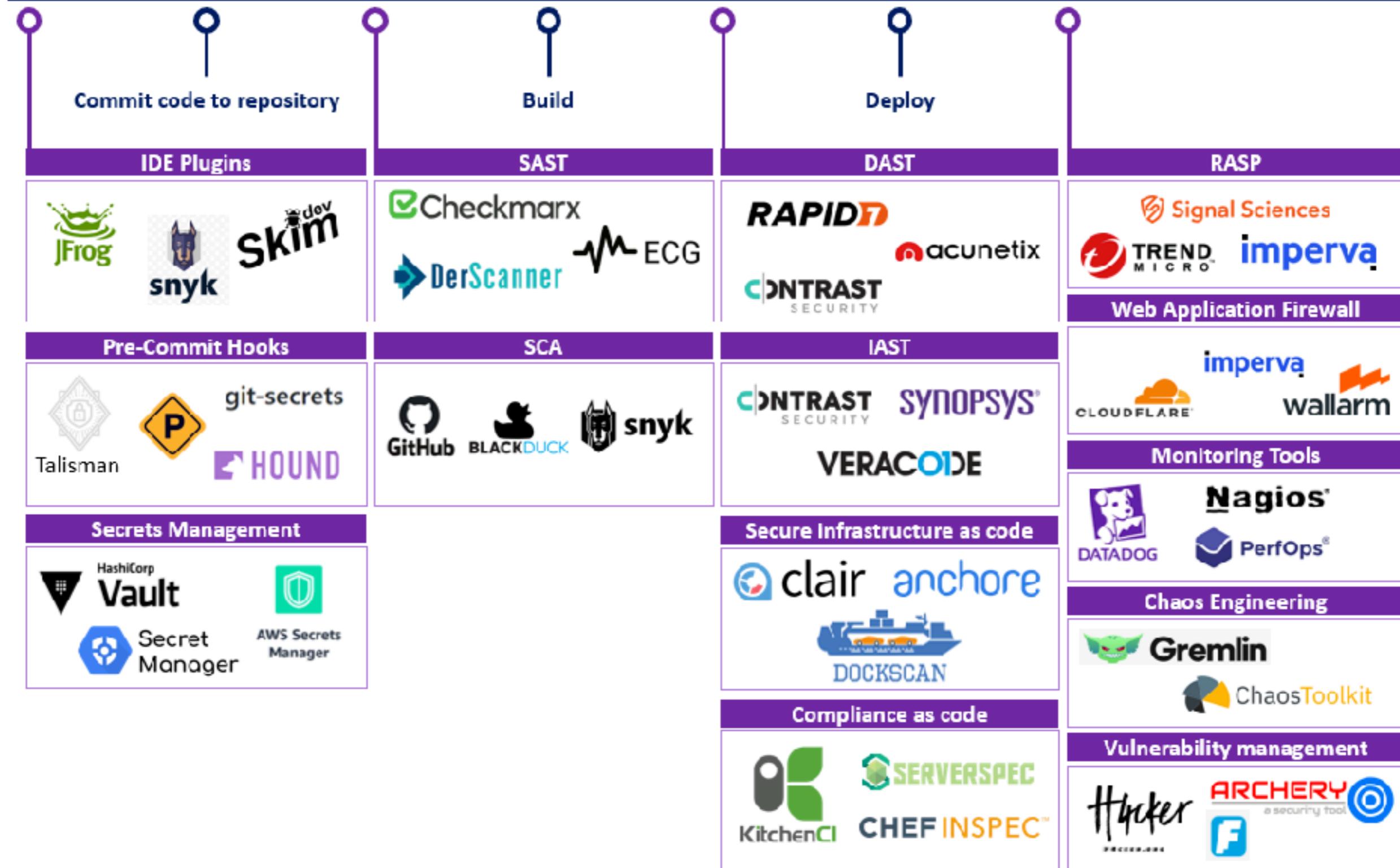
IAST tools look for security vulnerabilities, whereas **RASP** monitors the application for attacks, **and** protects the application against them when it senses an attack happening.

DevSecOps Pipeline Techstack

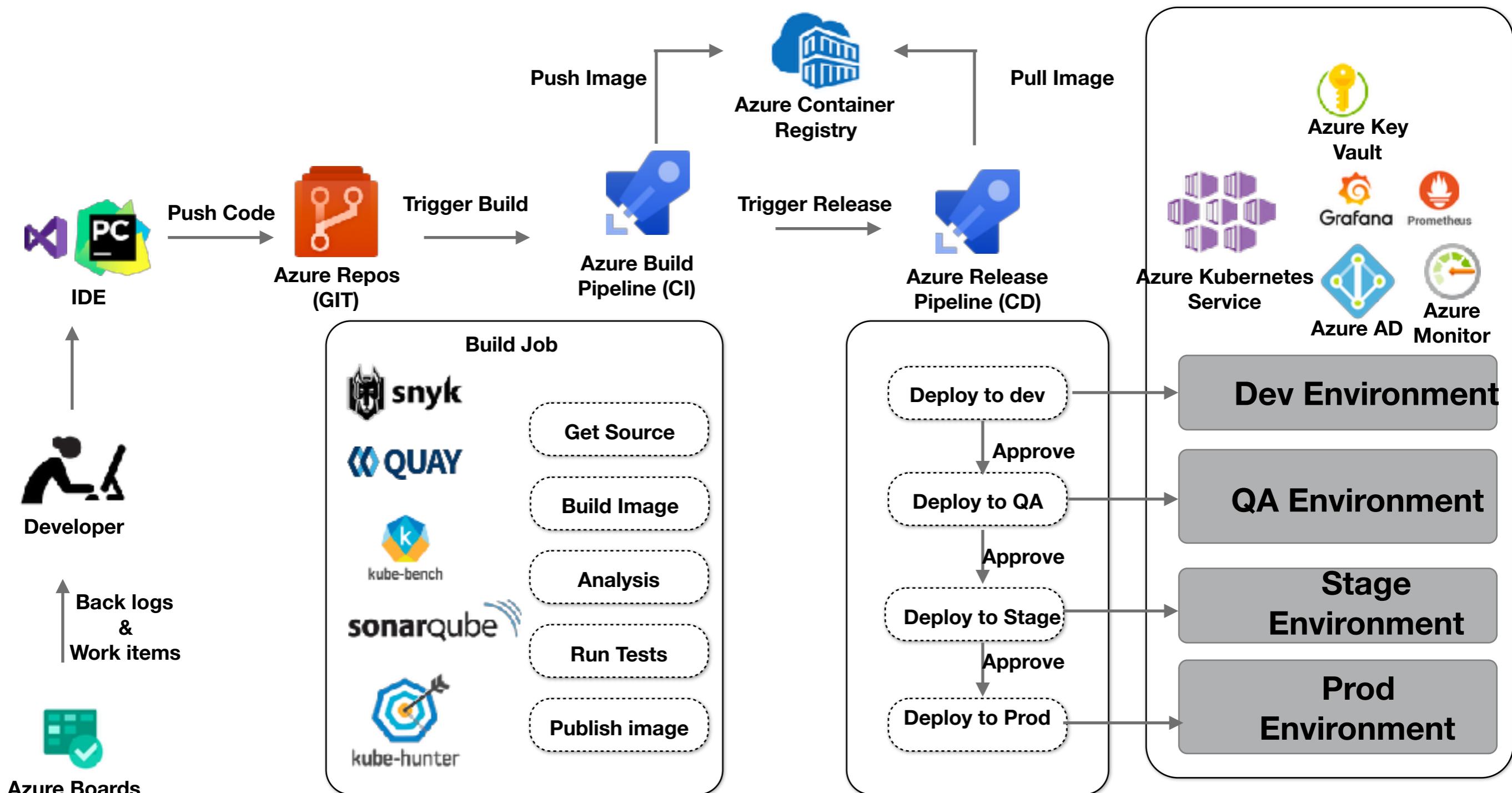
Legend: ● DevOps ○ DevSecOps

INOVO | VENTURE PARTNERS

CI/CD Pipeline

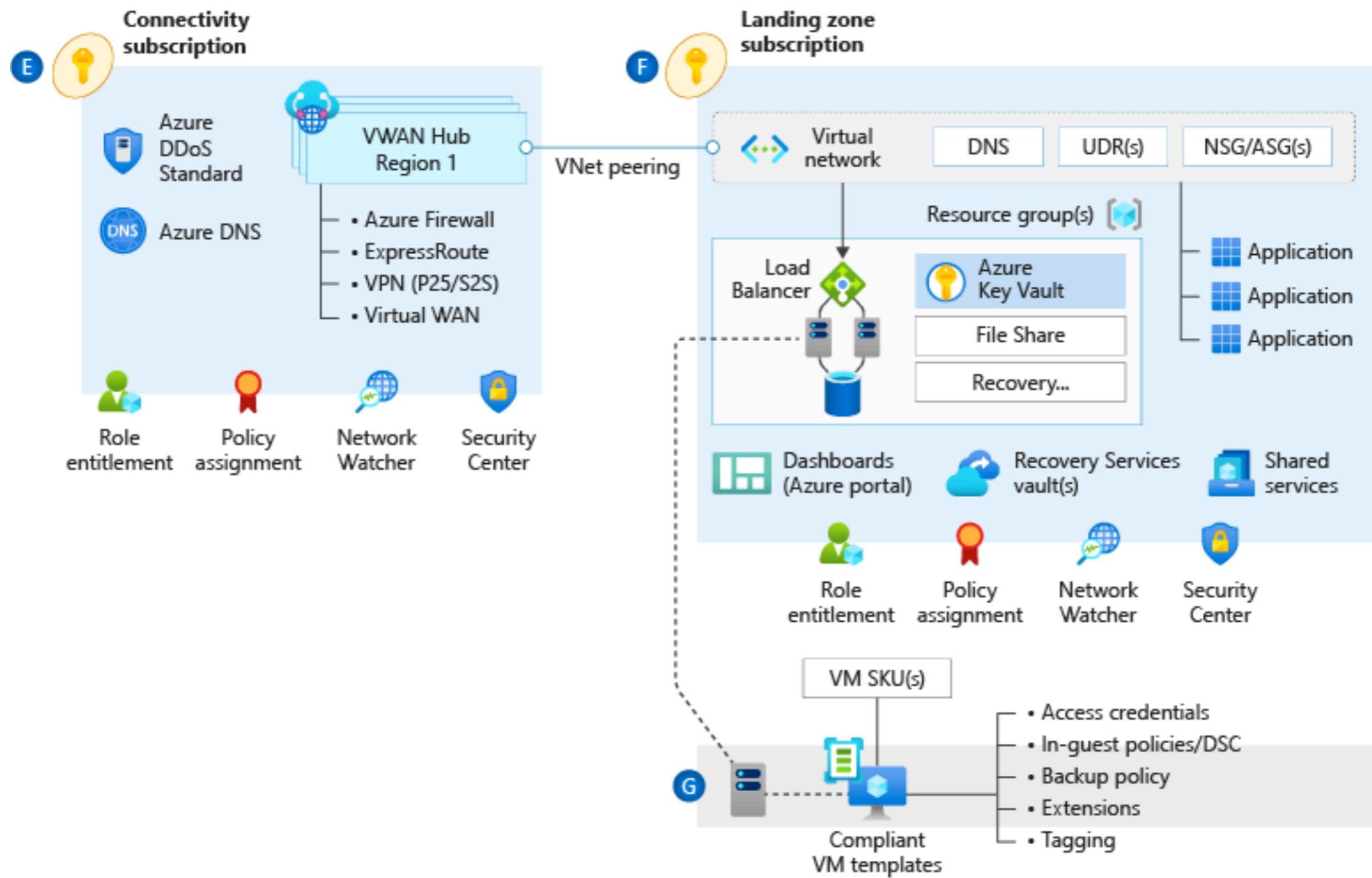


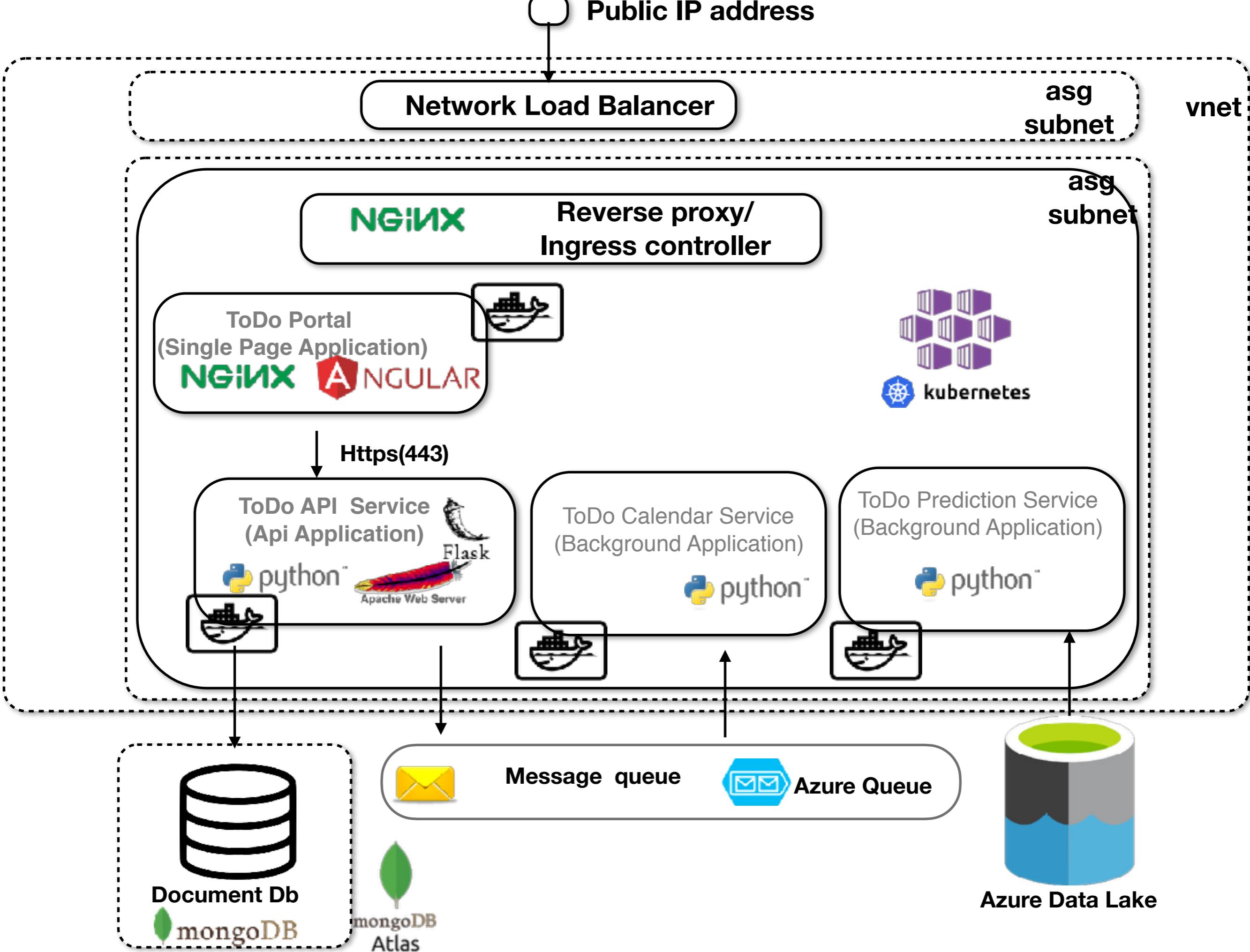
Reference model

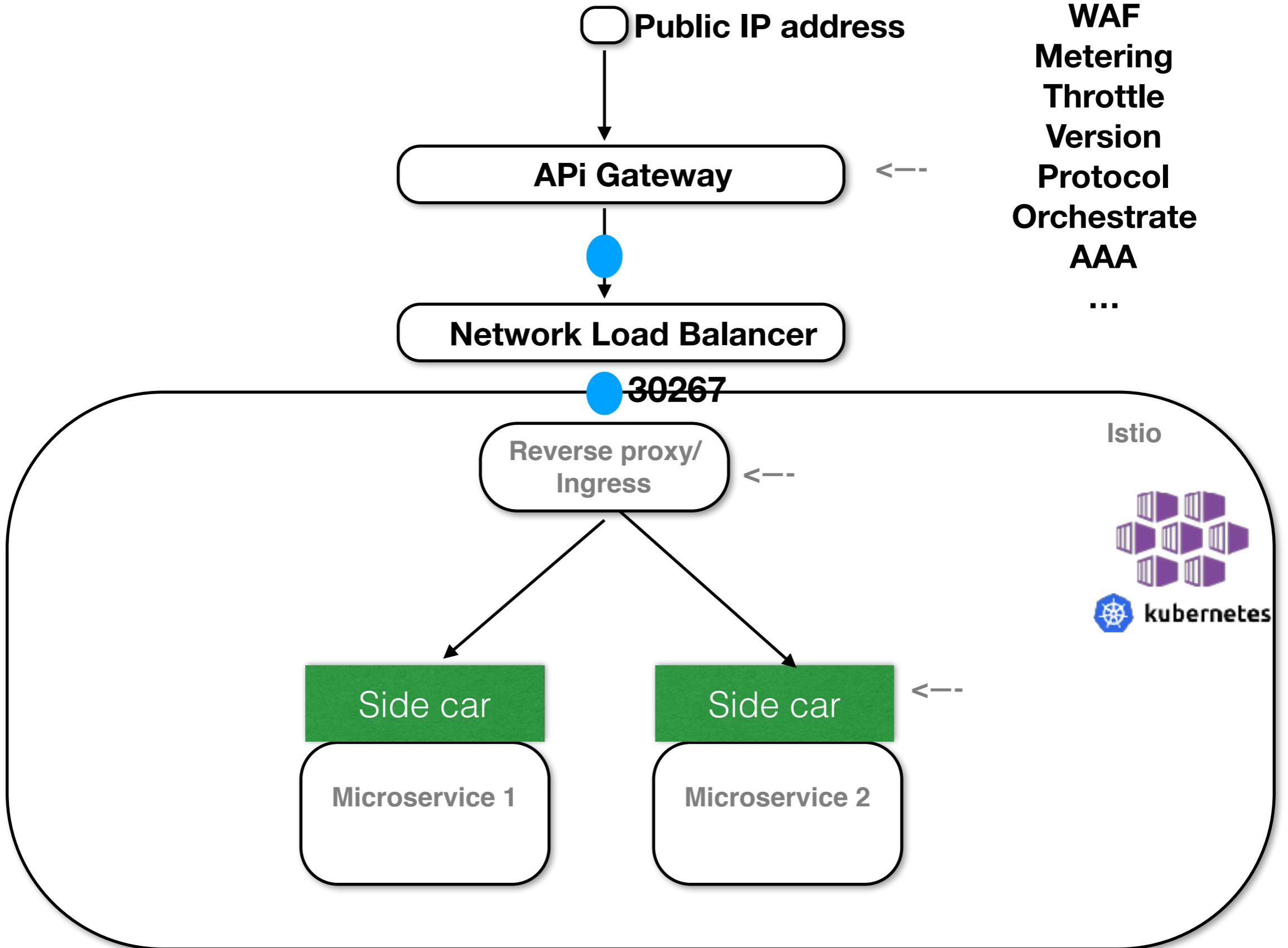


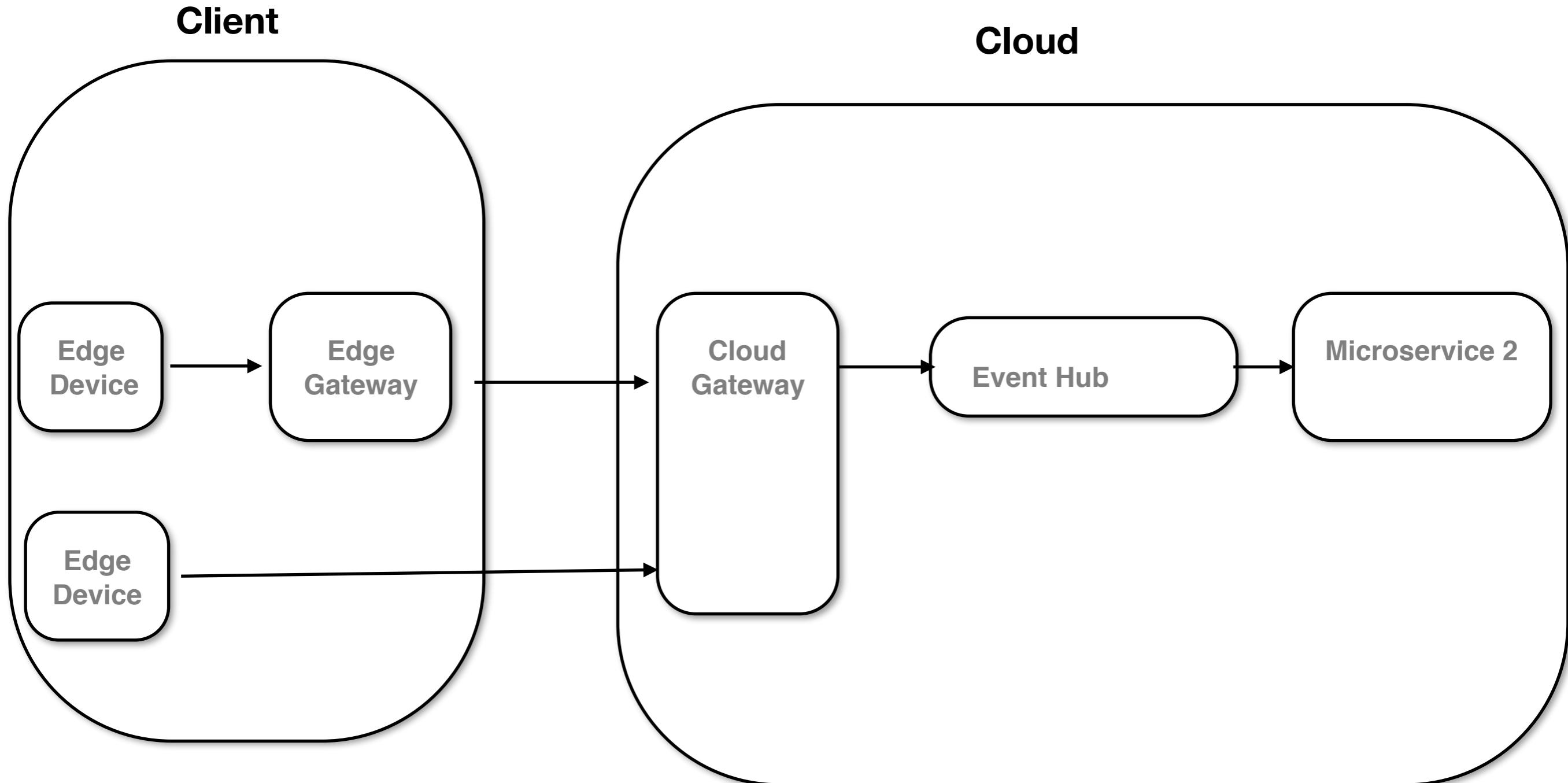
Infrastructure View

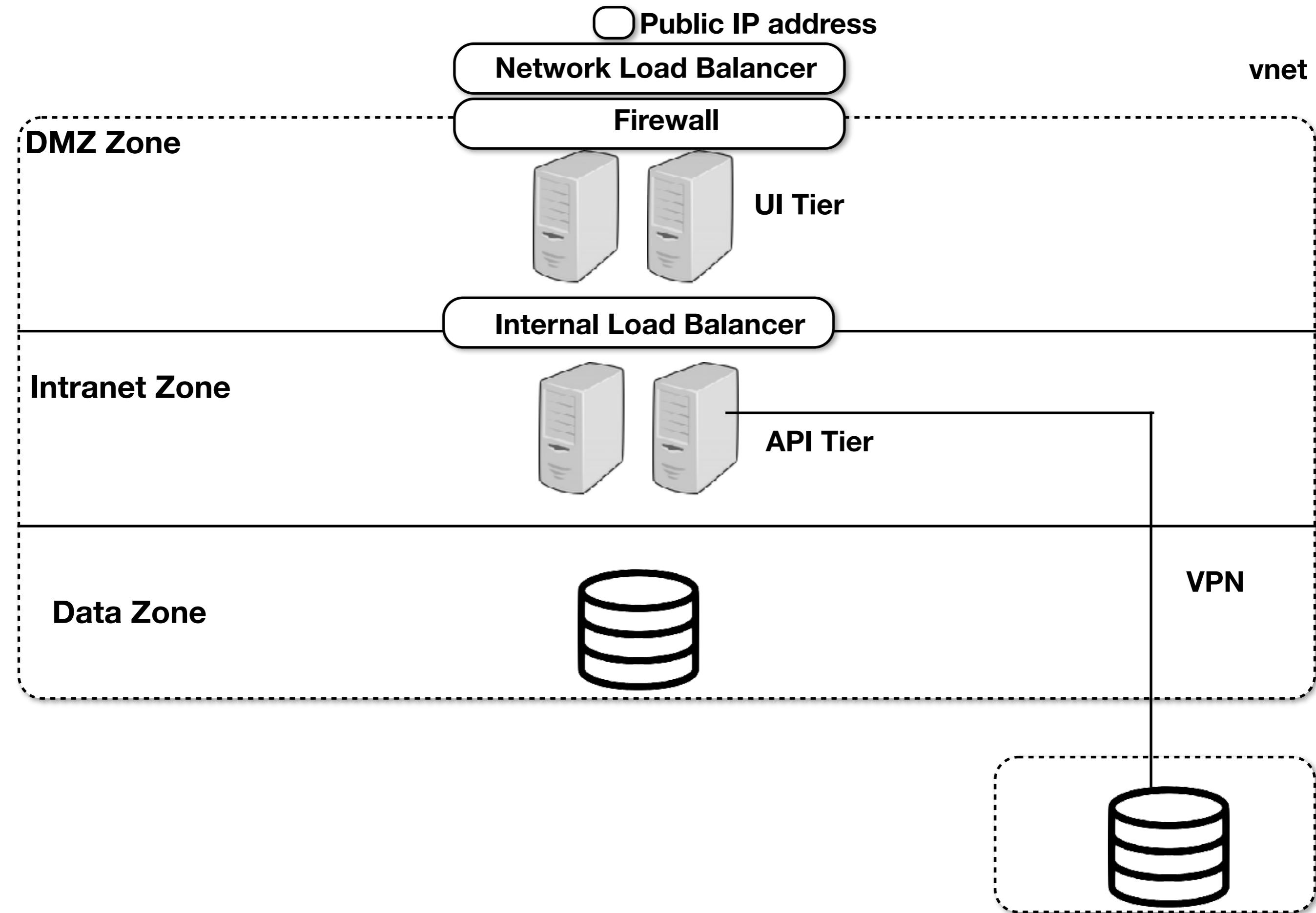
Physical View











DropBox

Logical View

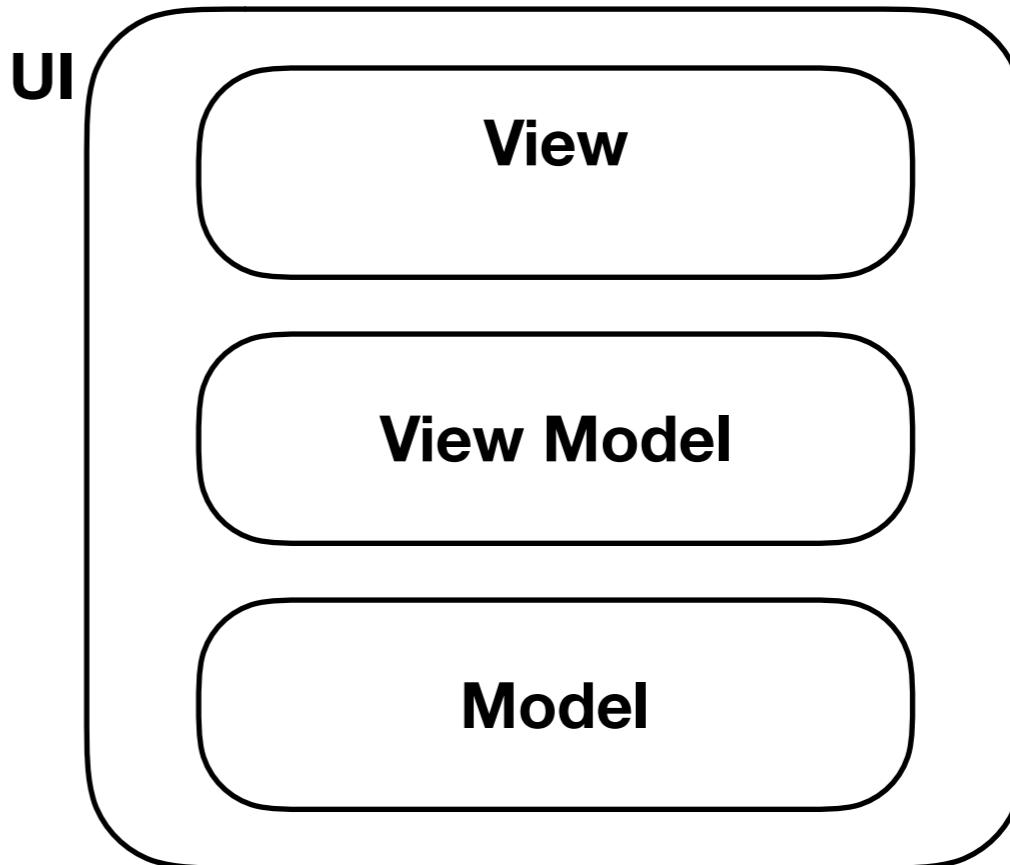
Web App
<< SPA >>

Mobile App
<< Client >>

Desktop App
<<client>>

DropBox API
<<server>>

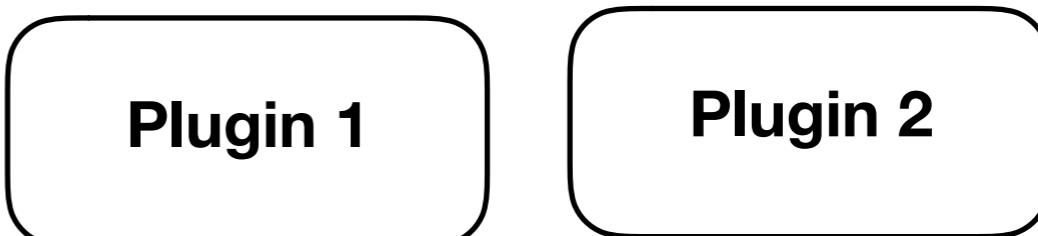
Desktop App
<<client>>

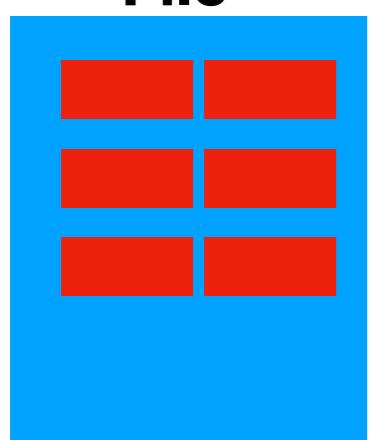


Domain responsibilities

Core

Interoperability





File



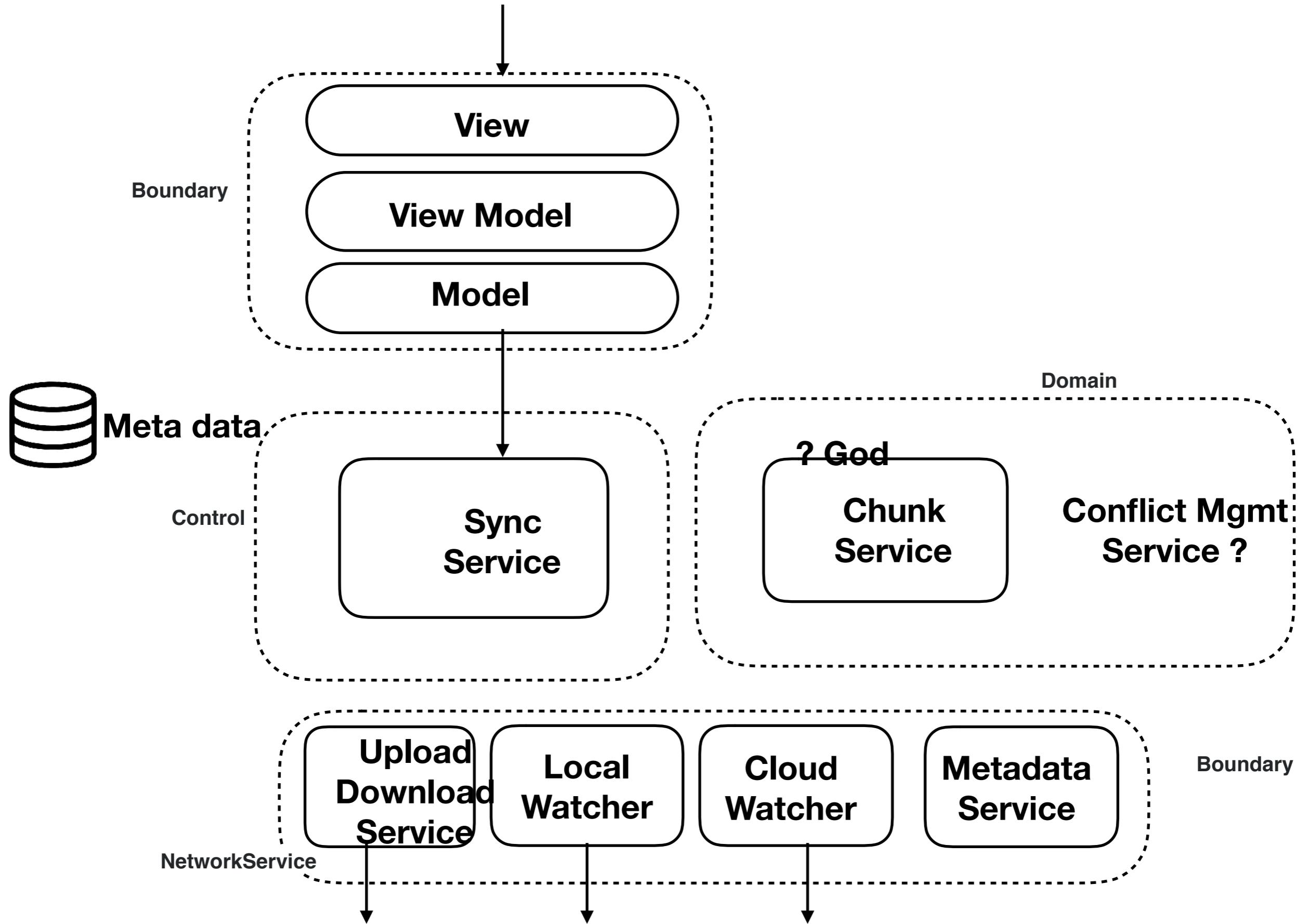
Meta data

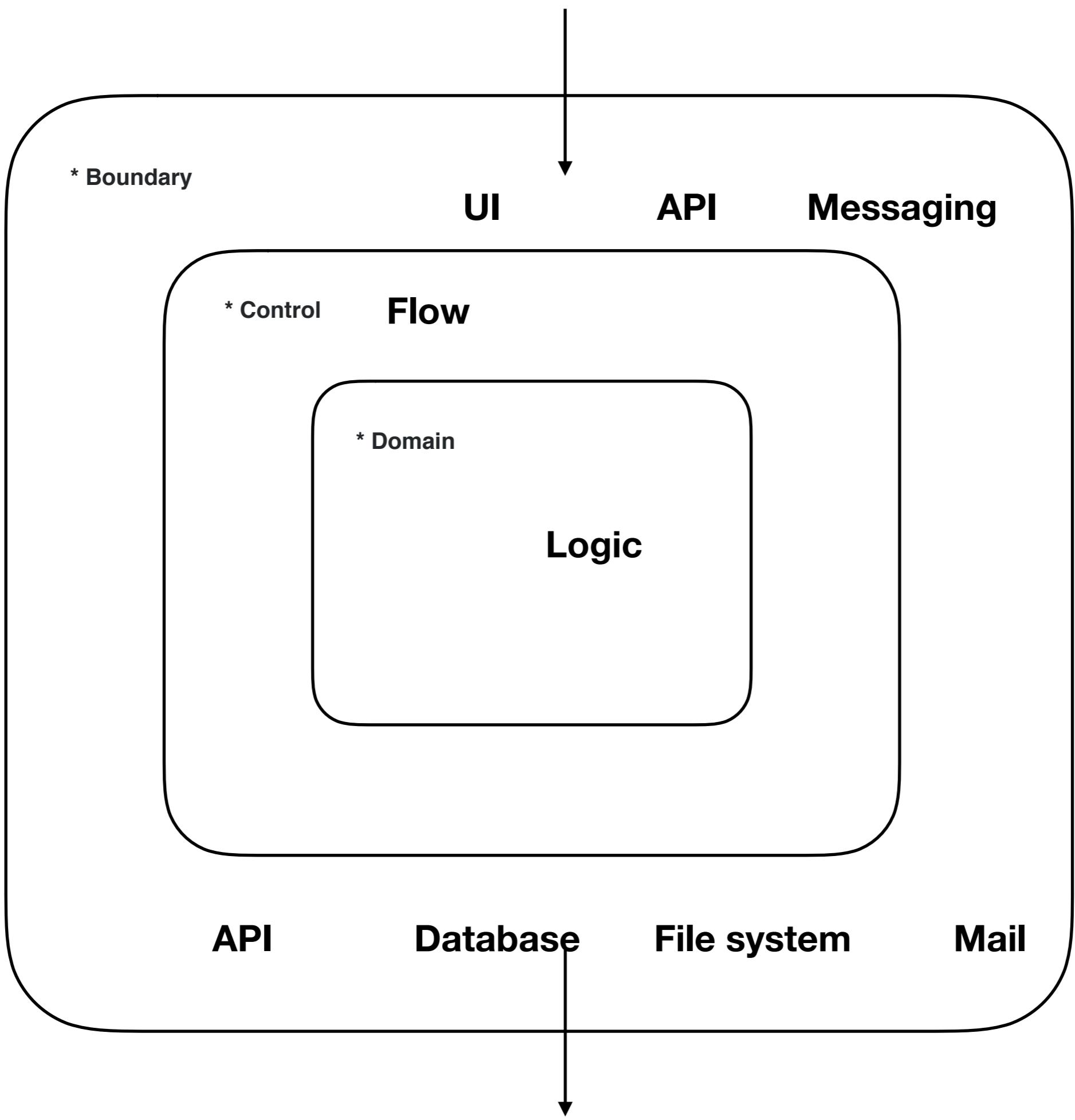
**Chunk
Service**

**Cloud
Watcher**

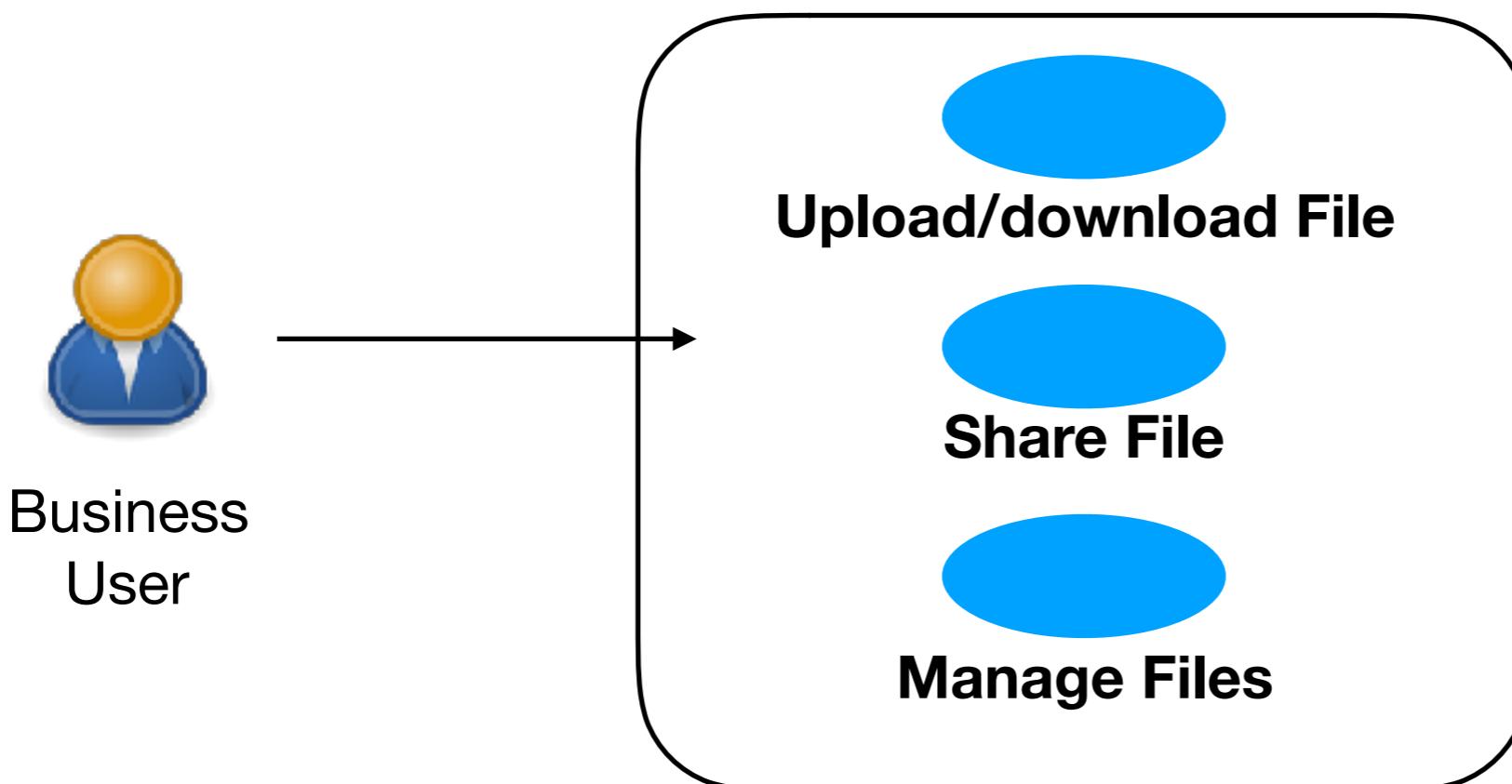
**Sync
Service**

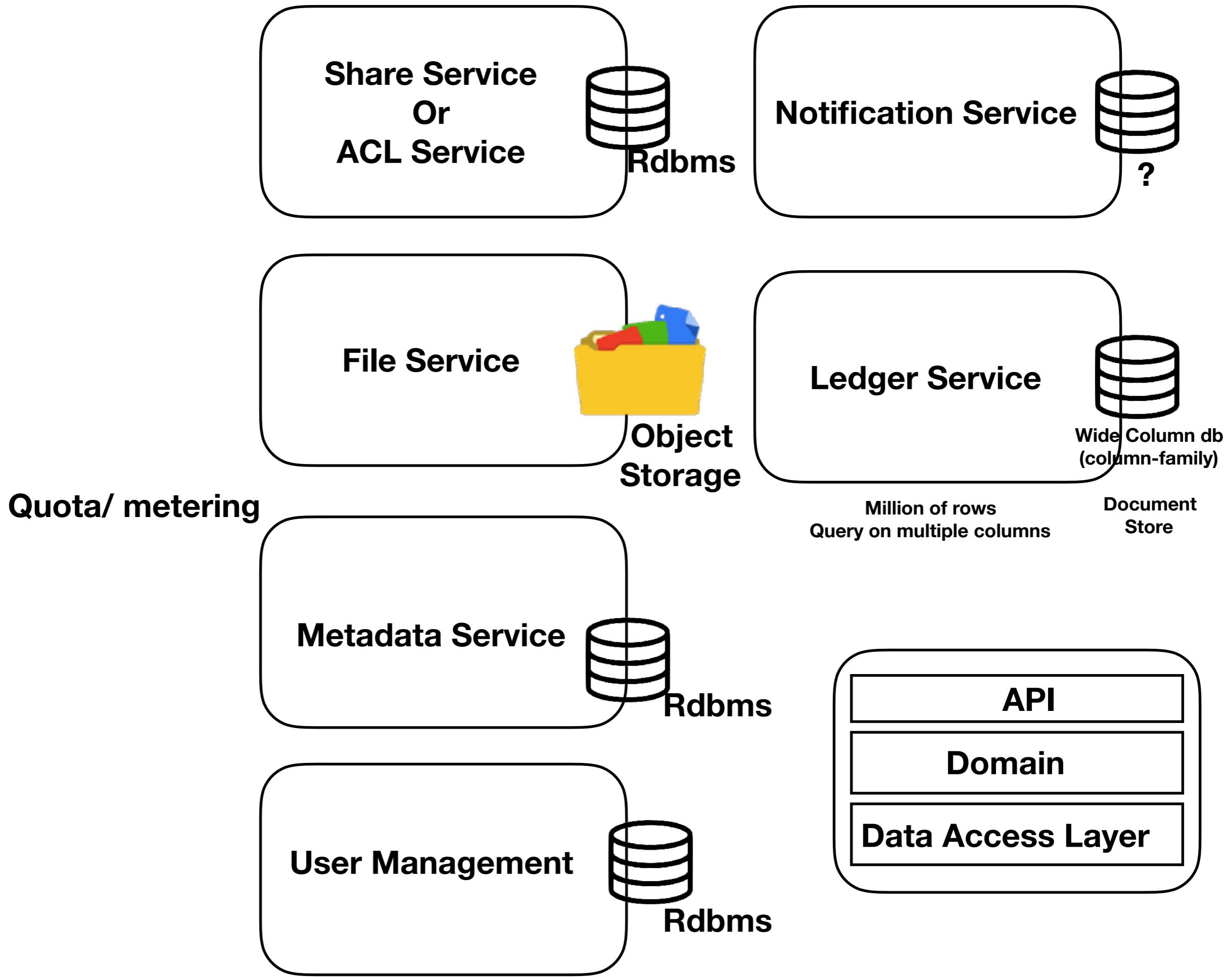
**Local
Watcher**

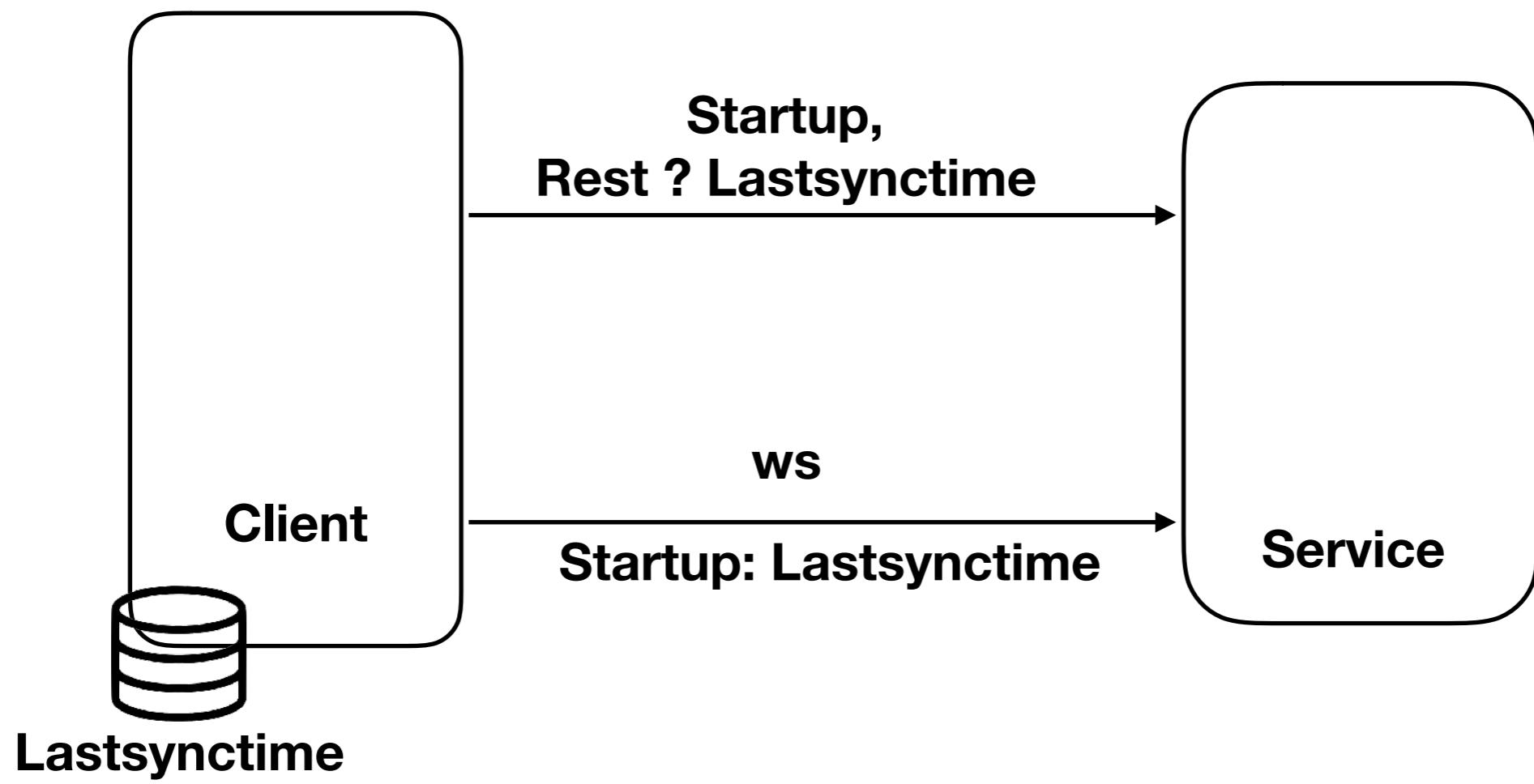


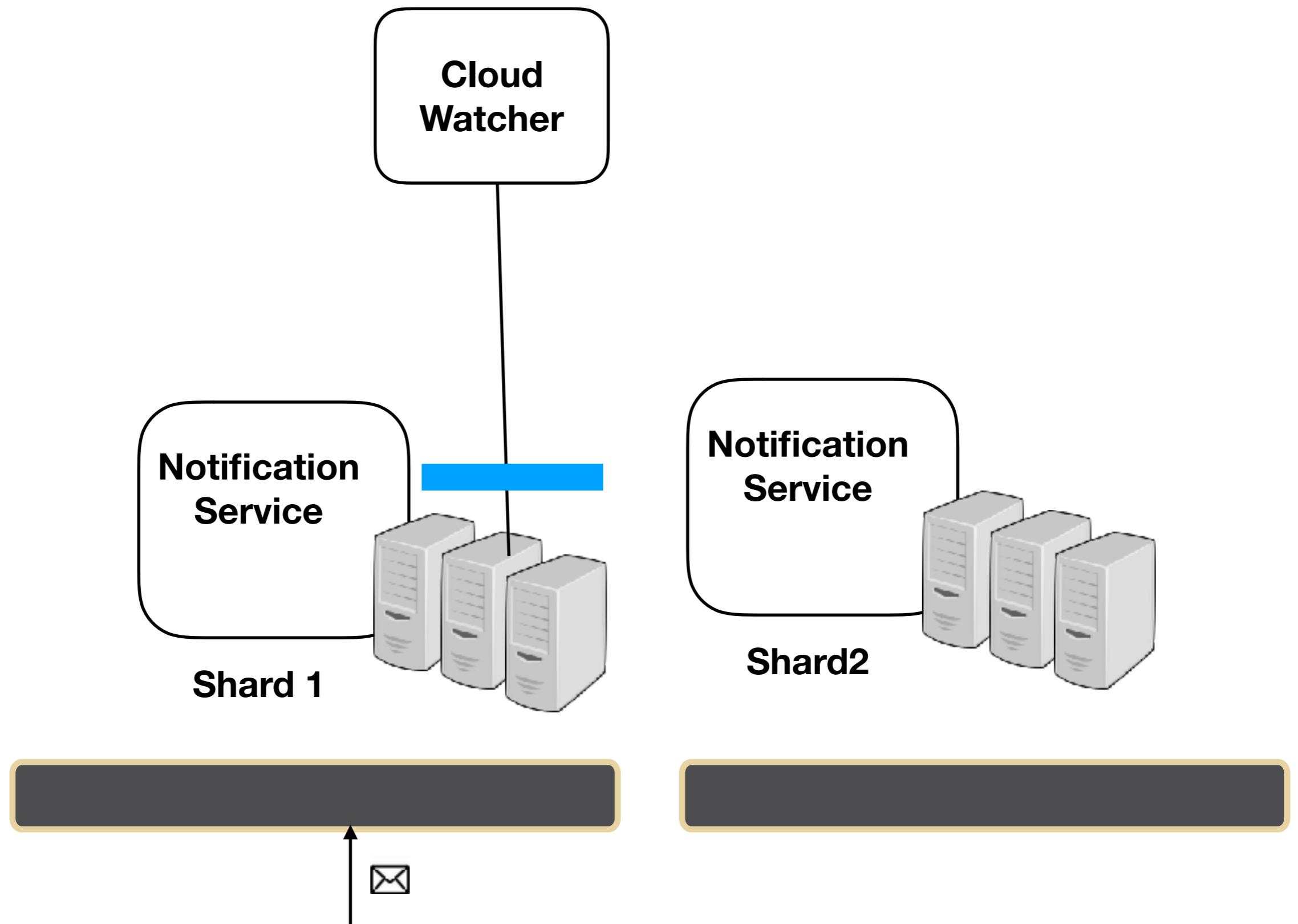


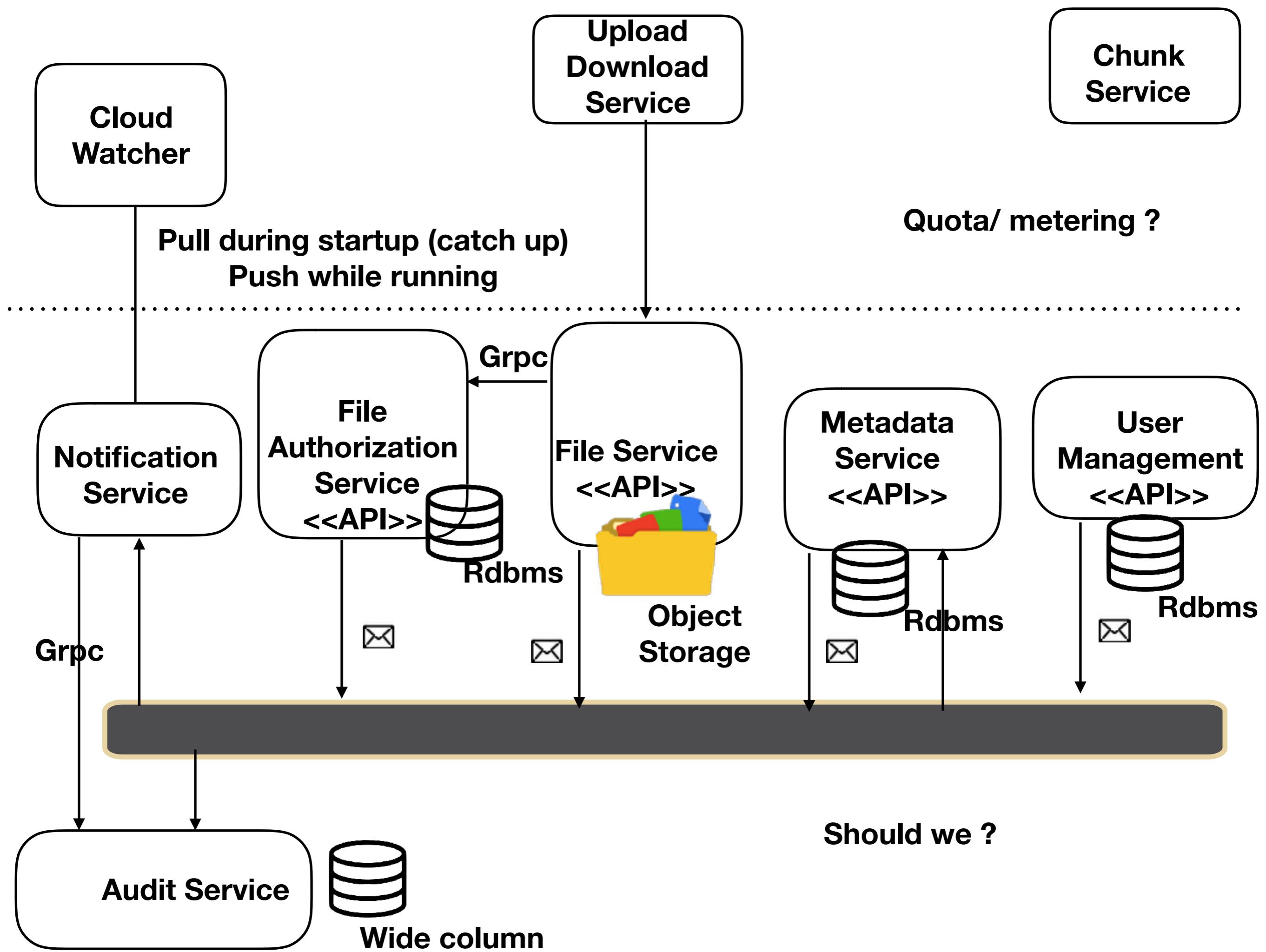
Functional view

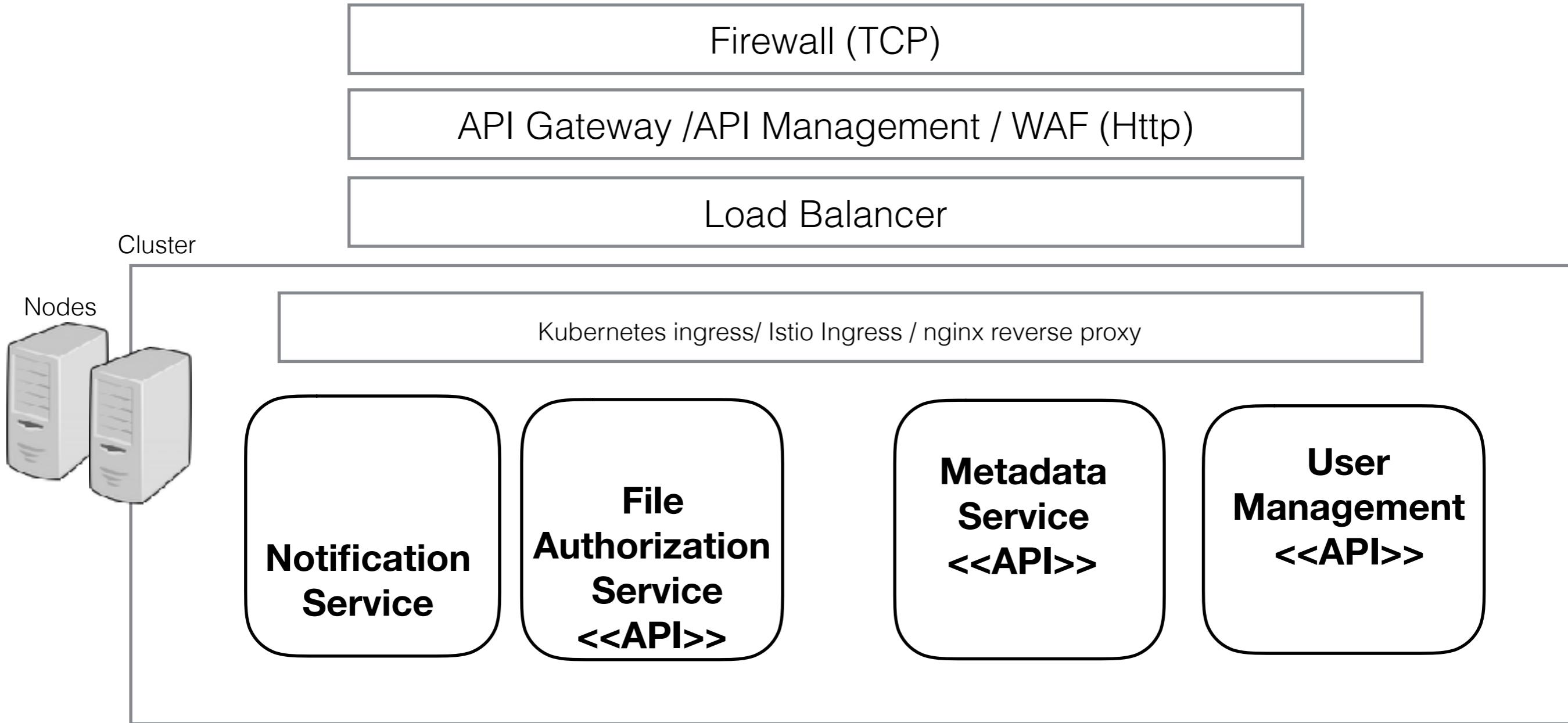












**Object
Storage**



Wide column

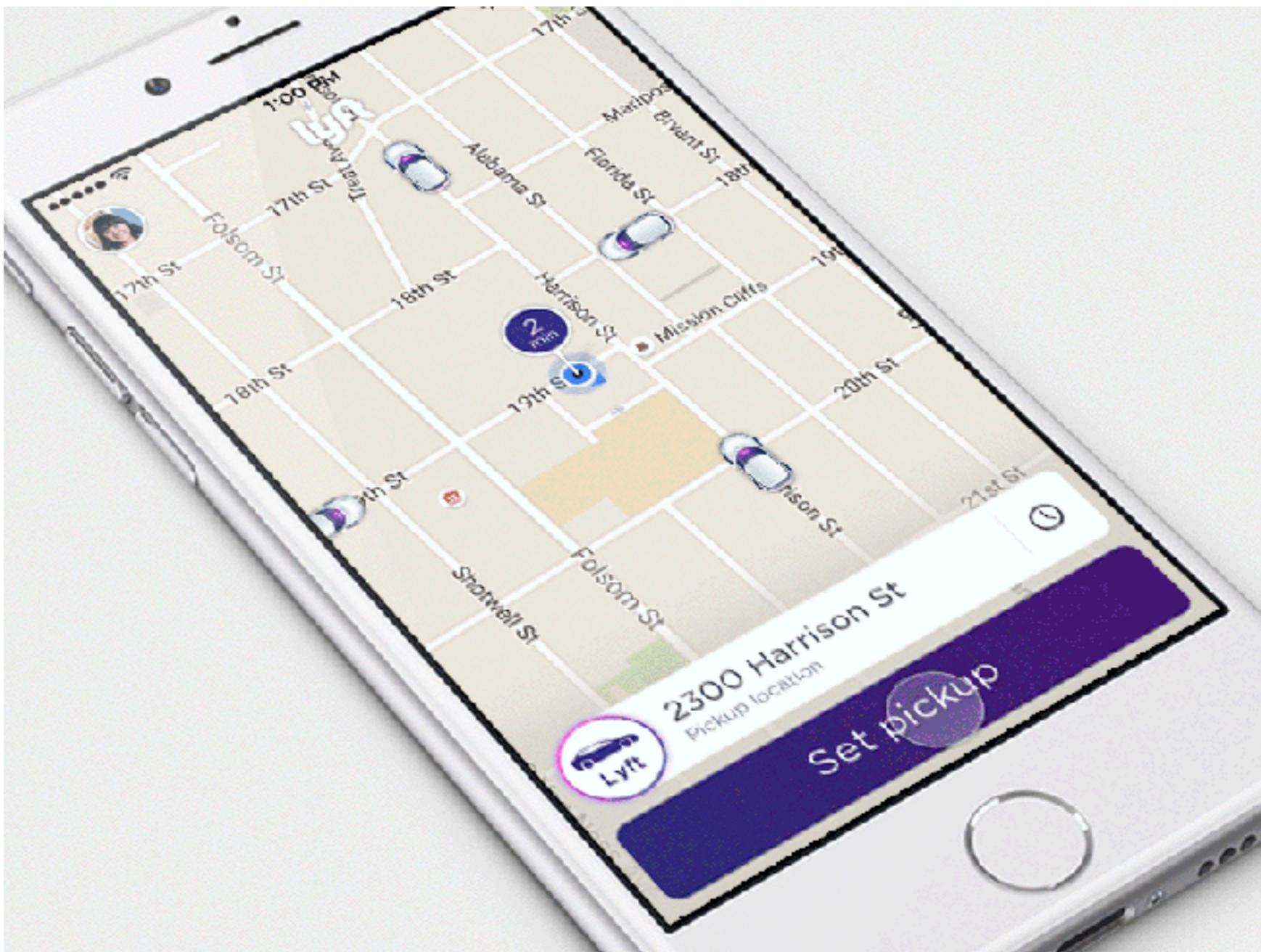


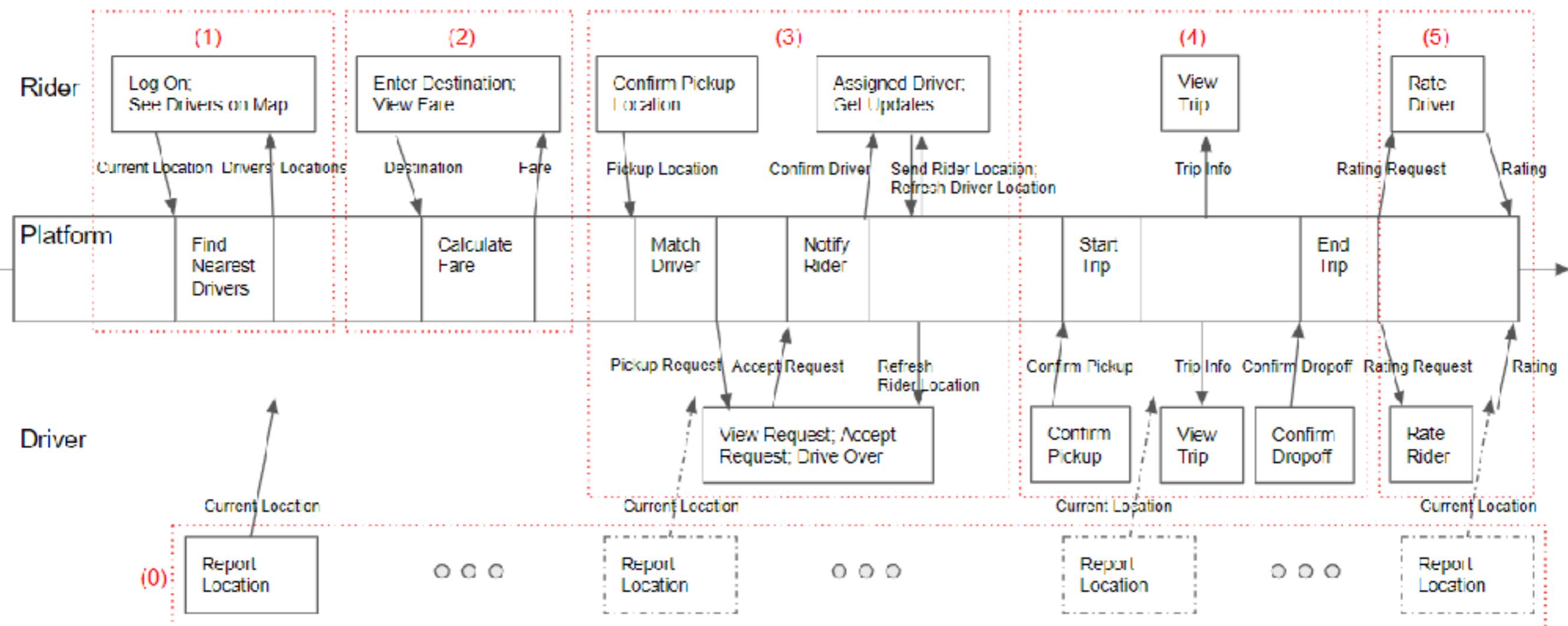
Rdbms



**Message
Queue**

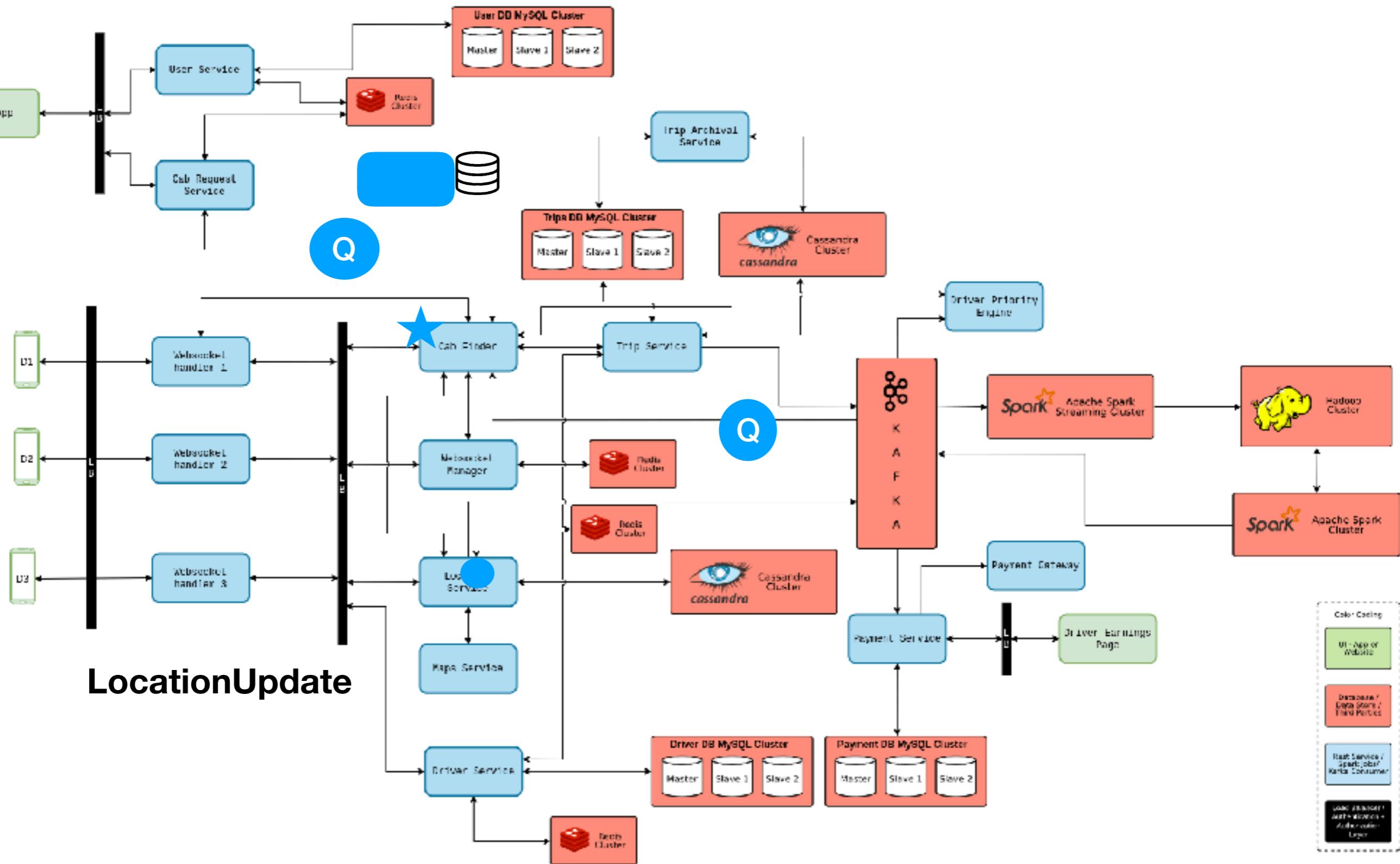
Uber reference architecture





Uber/Lyft/Ola System Design

<code karle>

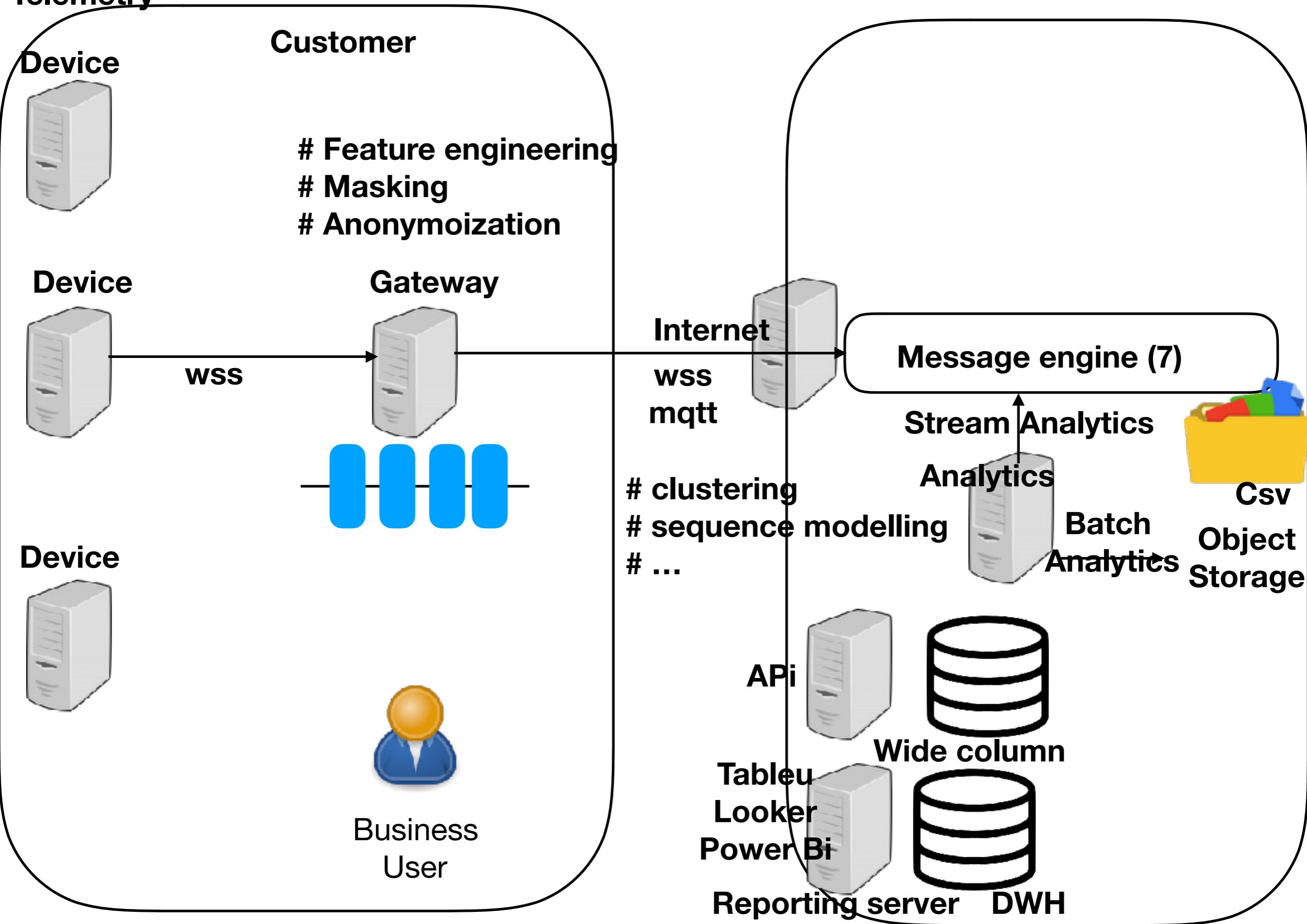


LocationUpdate

1.collect Telemetry

2.secure

3. Analyse





Connected vehicles,
Fleet management



Connected
manufacturing



Facilities
management



- Authentication

-

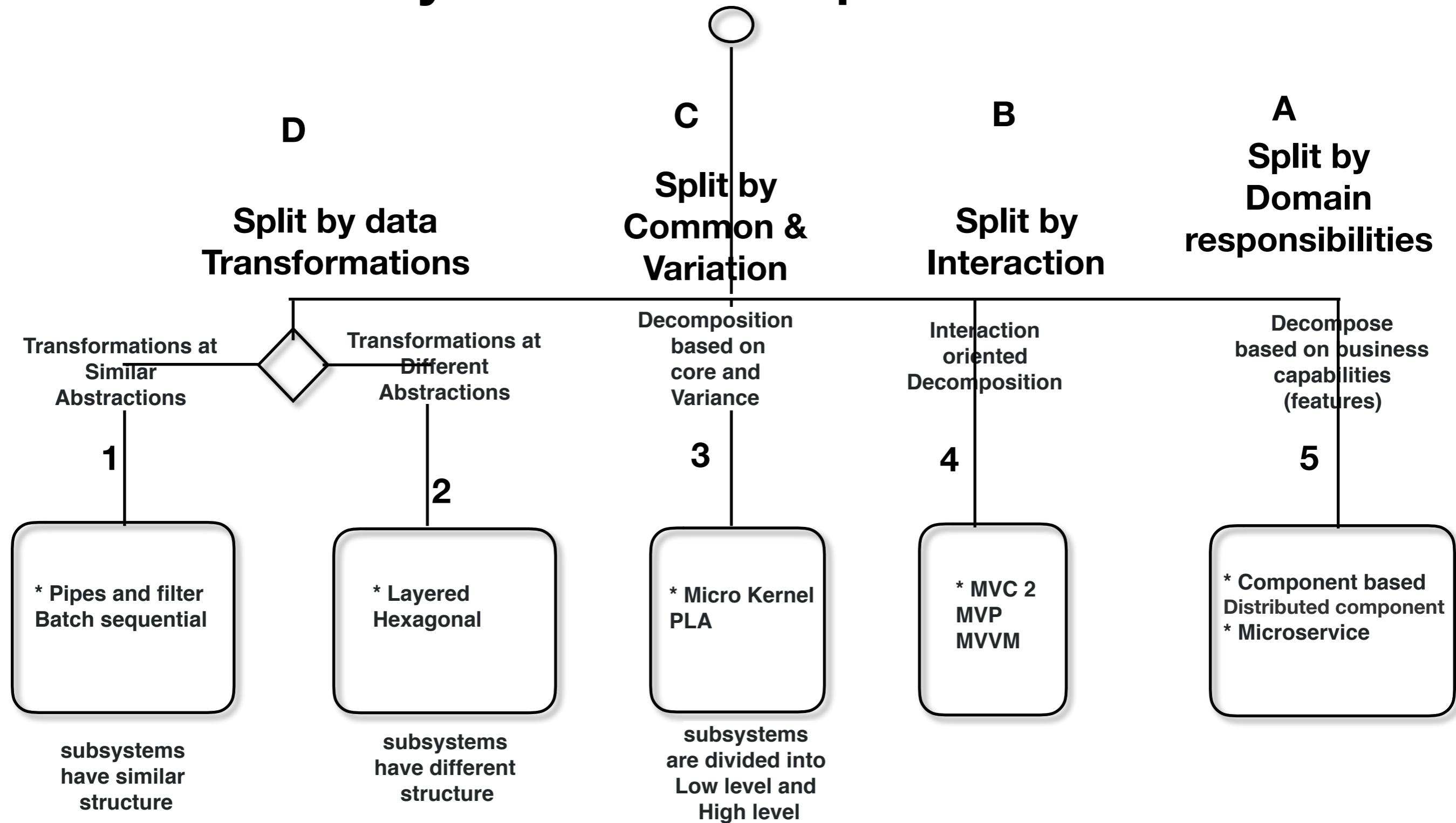
All the active drivers who are ready to accept trips are connected to a **WebSocket handler**

WebSocket manager which keeps track of which WebSocket handlers are connected to which drivers in a **Redis**. Redis, along with keeping track of which host is connected to which driver, will also keep a reverse mapping for which driver is connected to which host.

The MySQL stores information about all the trips that are about to start or are in progress. Once the trip completes this information is moved to Cassandra.

Cab finder gets a driver who will make the trip and sends the trip and driver information back to the cab request service.

Choose System Decomposition Patterns



Actor *

Arch Risk

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

Evaluate Architecture (ATAM)

identify all Architectural approaches
(design)

A1 (CQRS)

A2(Cube)

A3 (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

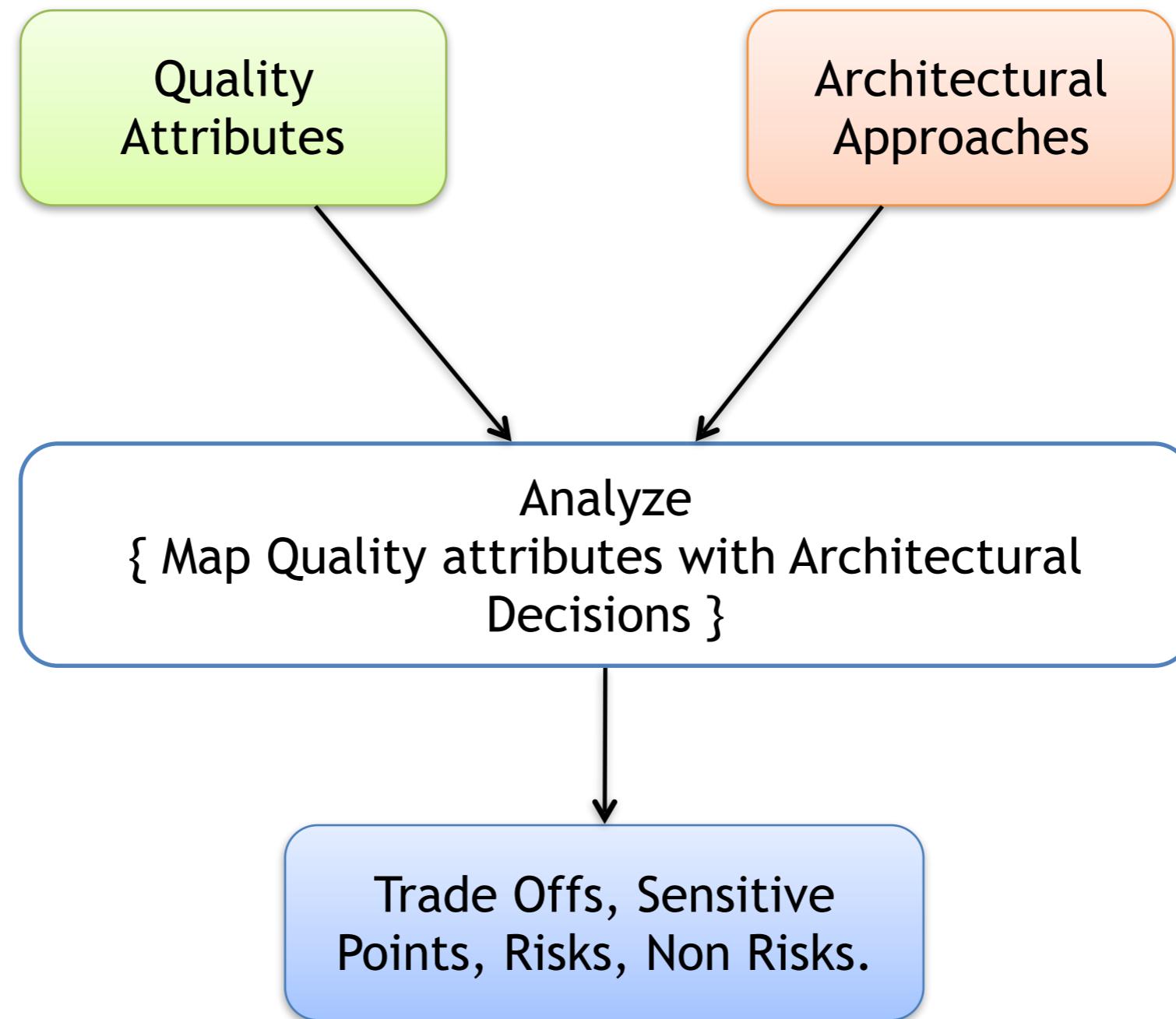
S10-> ?

=> Risk & trade off's

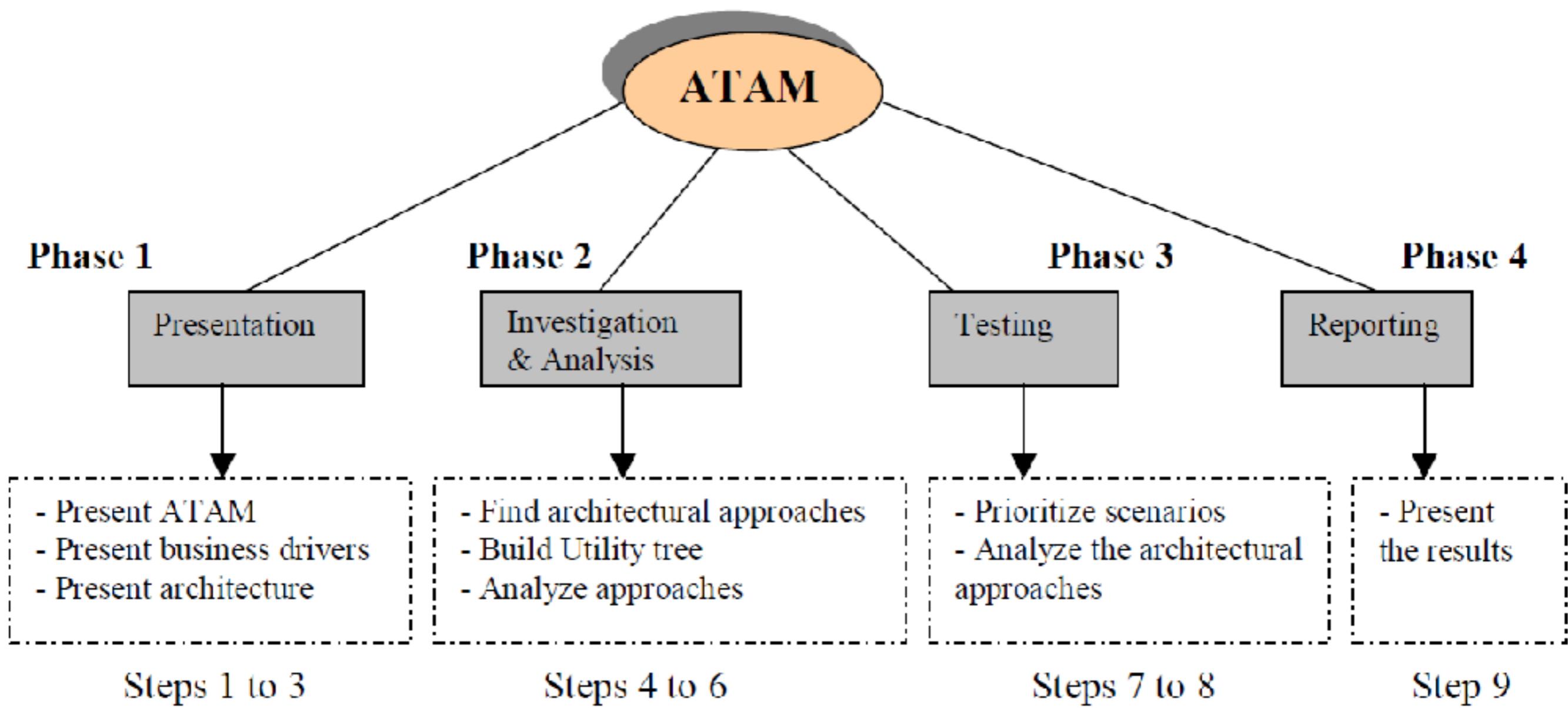
=> Risk & trade off's

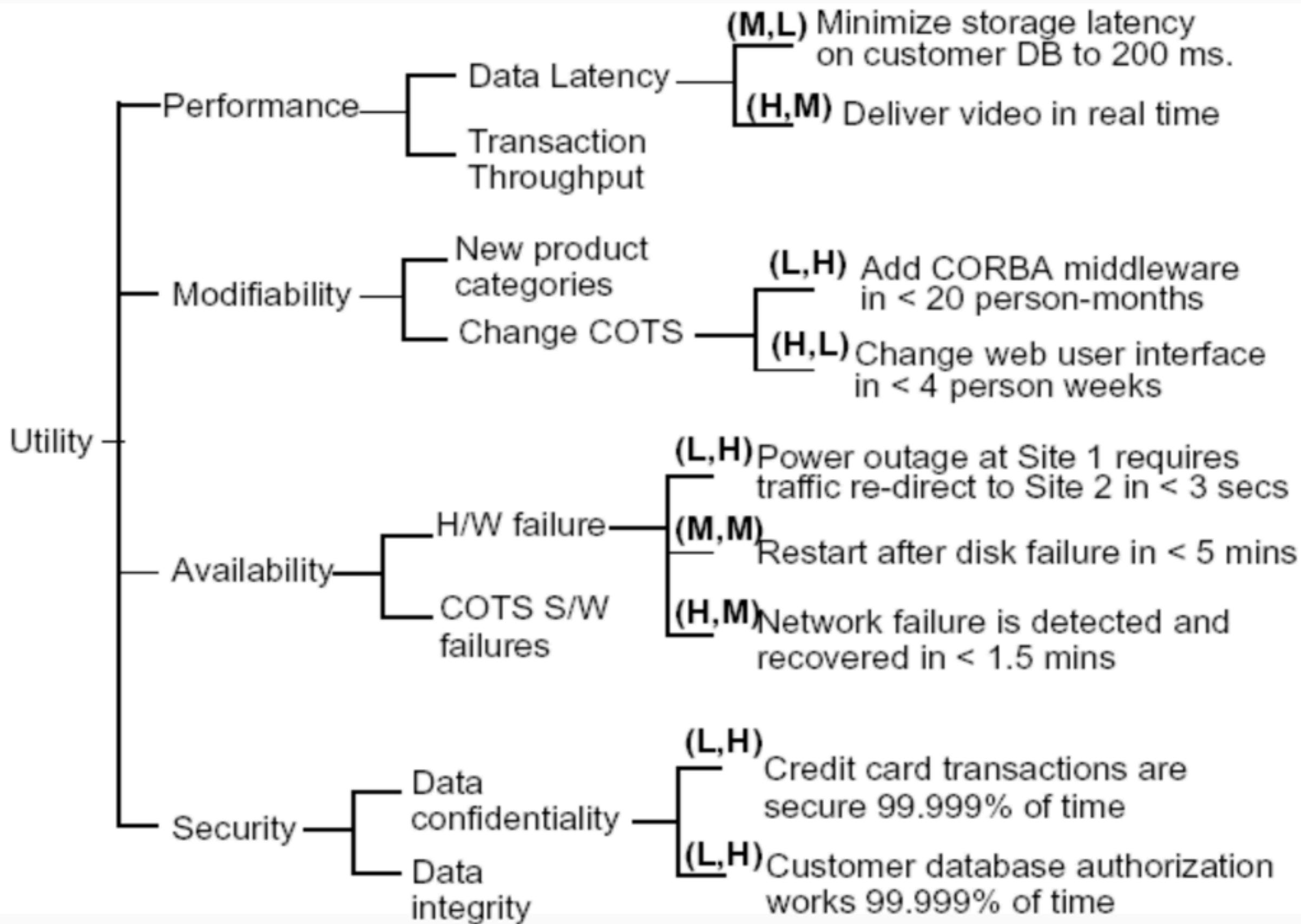


Evaluation Team



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.	