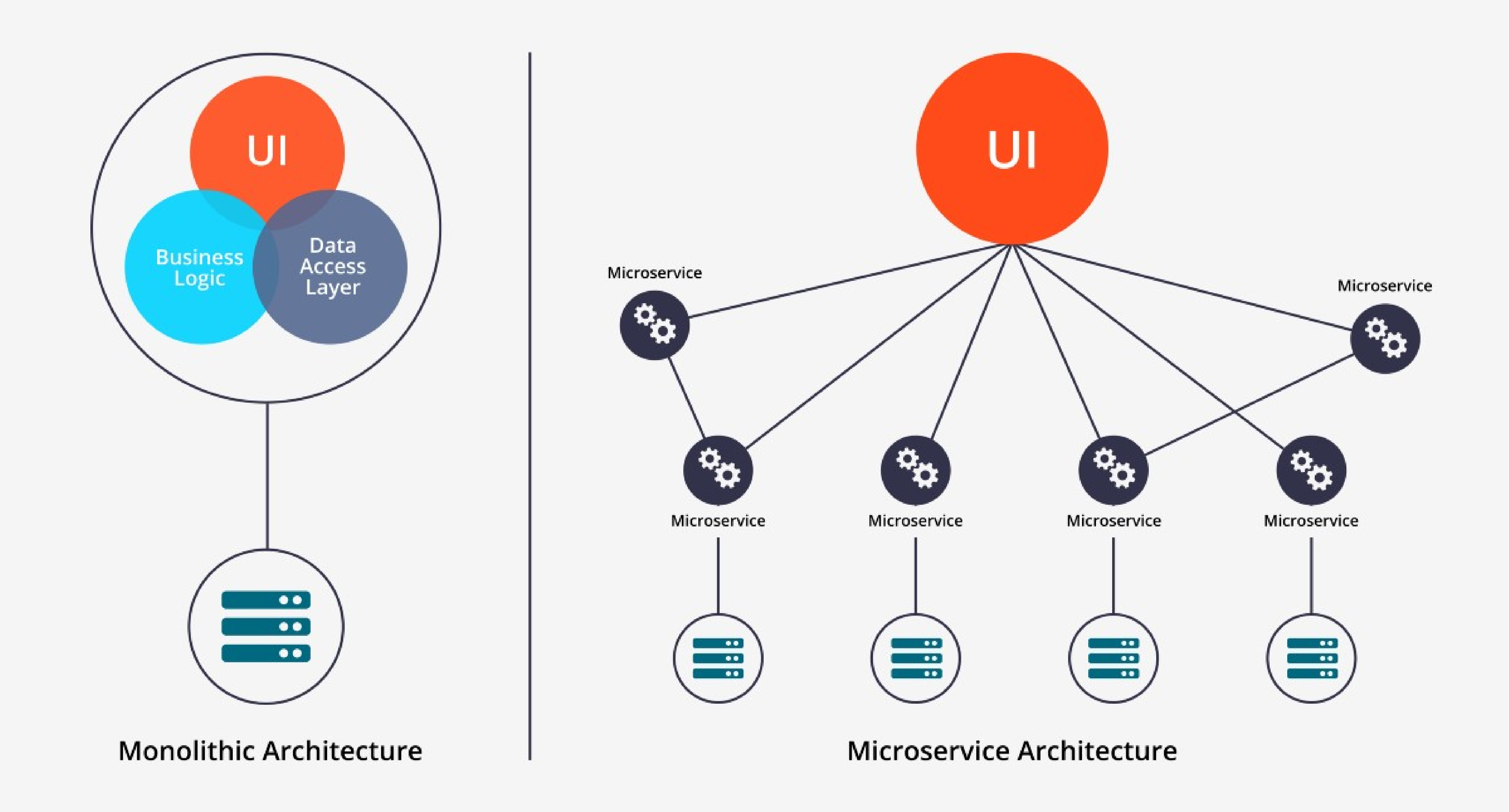
Microservices

Request Based Microservices

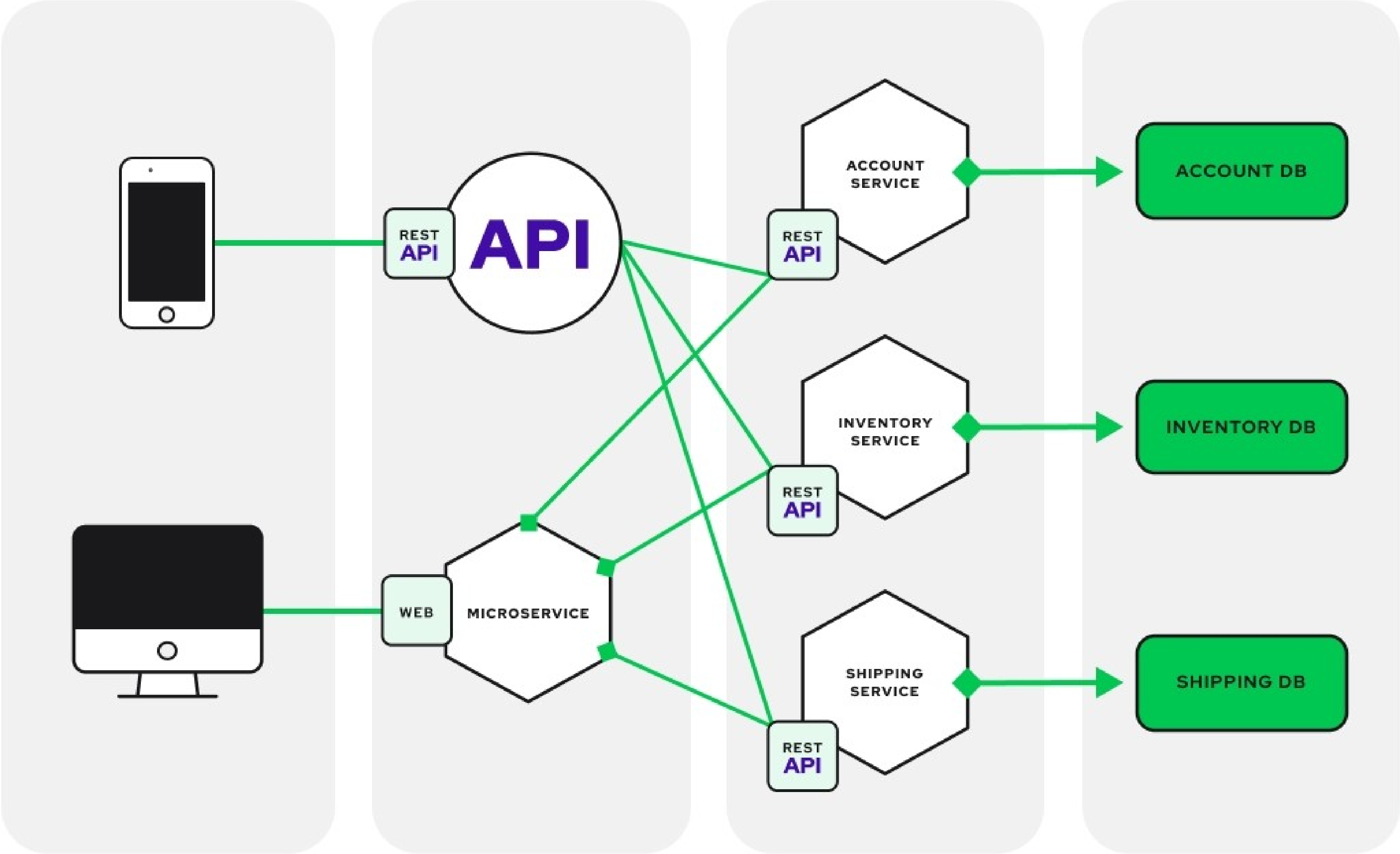
General Types of MS Patterns

- Request based
  - Use HTTP and other related protocols
  - Client Server model – responds to requests
  - Employs tools like Kubernetes, REST gRPC
- Event based
  - Uses stream based tools like Kafka
  - Pub-sub model – event producers and consumers
- Microservices often use a mix of these two models
- In this session take a look at request versus event based tools

Monolith Versus Microservices



Request Based

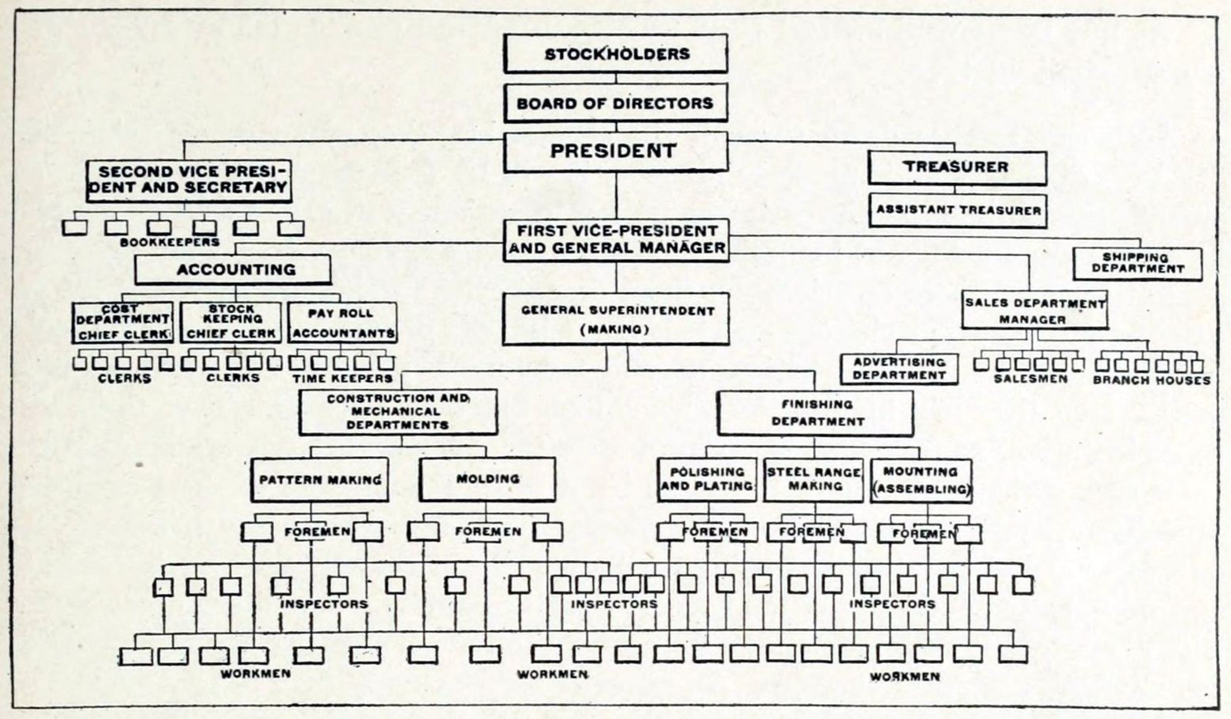


Complex Systems

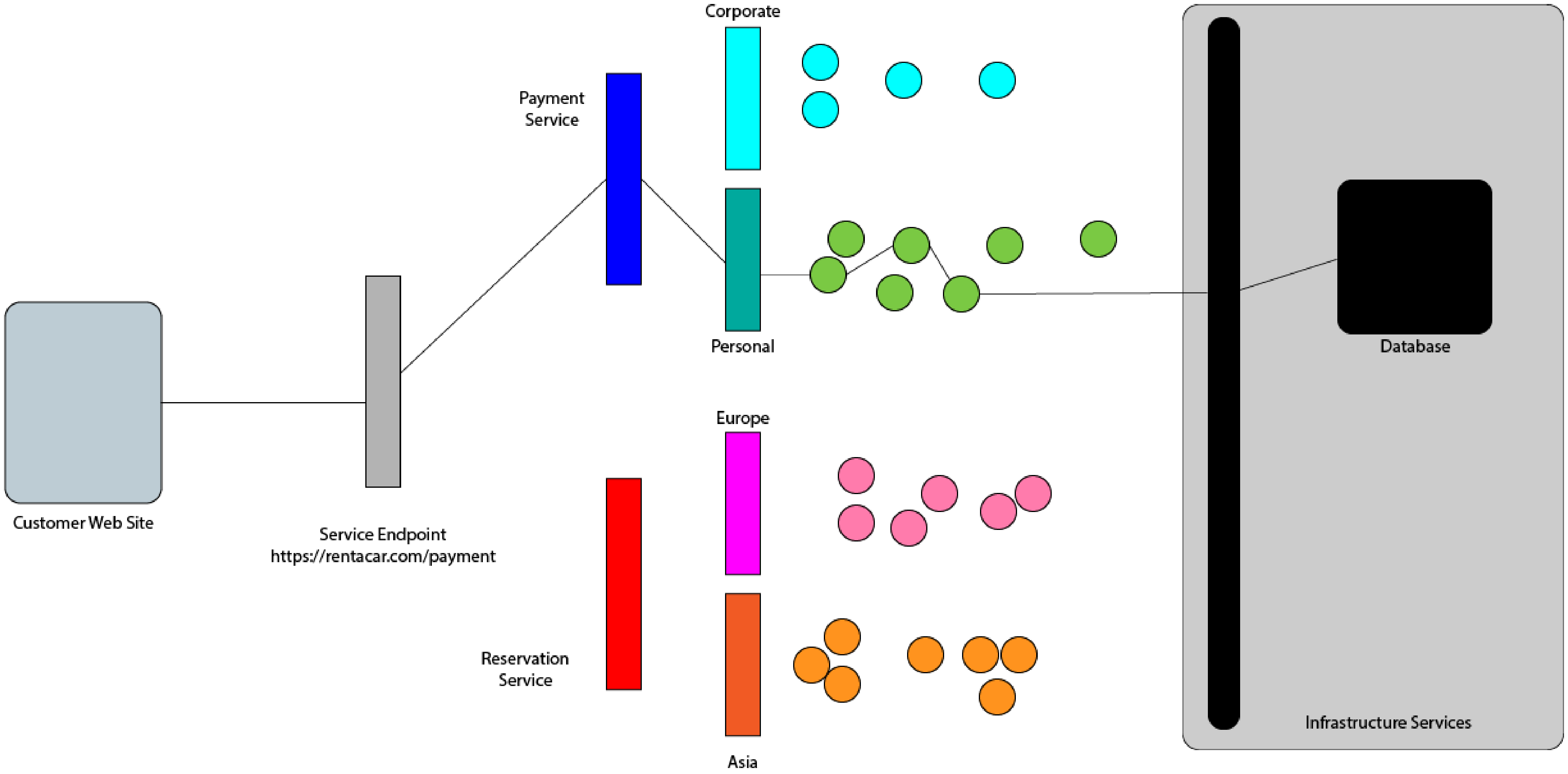
Frequently, complexity takes the form of a hierarchy, whereby a complex system is composed of interrelated subsystems that have in turn their own subsystems, and so on, until some lowest level of elementary components is reached

Courtois

On Time and Space Decomposition of Complex Structures
Communications of the ACM, 1985, 28(6)



Request Based



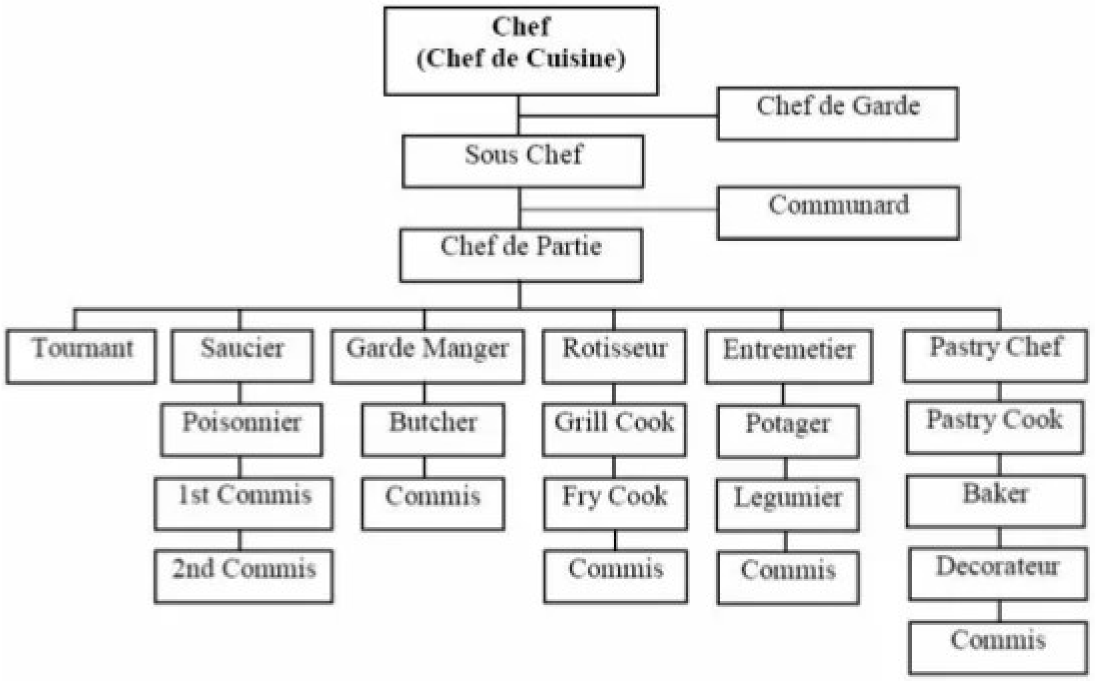
Processing a Request

- Request arrives at an endpoint
- Depending on the service requested
  - The message is routed to an internal service that handles that type of request
  - That service routes the message to a running process to handle the process
  - The process retrieves any state data it needs from the data service
  - Request is processed
  - State data store is updated
  - Response returned to service that returns to the endpoint that received the request
- More that one process type may be involve
  - Processes are stateless and horizontally scale-able
  - The main challenge: how do you orchestrate the flow of messages?

French Kitchen Brigade

Kitchen Organization Chart

The Classical Brigade



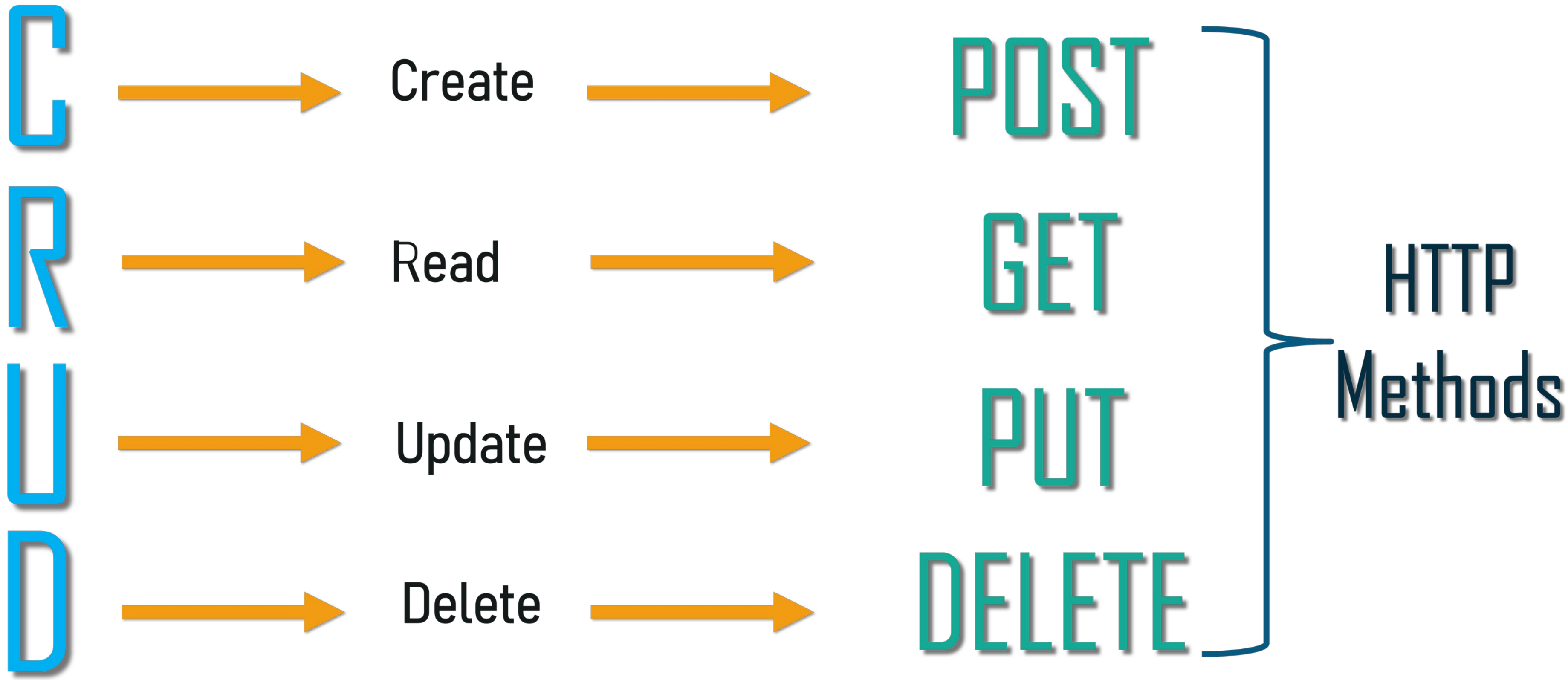
Bad Brigade

REST Protocol

- Assumes that “things the system does” are represented by domain objects
  - An order, a registration, a sale, a shipment, etc.
  - This is the Command design pattern
- There are only a limited number of things we can do with those objects
  - Provide data to create a new business object
  - Retrieve an existing business object
  - Update an existing business object with new data
  - Delete an existing business object
- CRUD functionality

REST Protocol

- REST rides on HTTP
- They type of HTTP message determines the CRUD request



Sample Car Registration Service

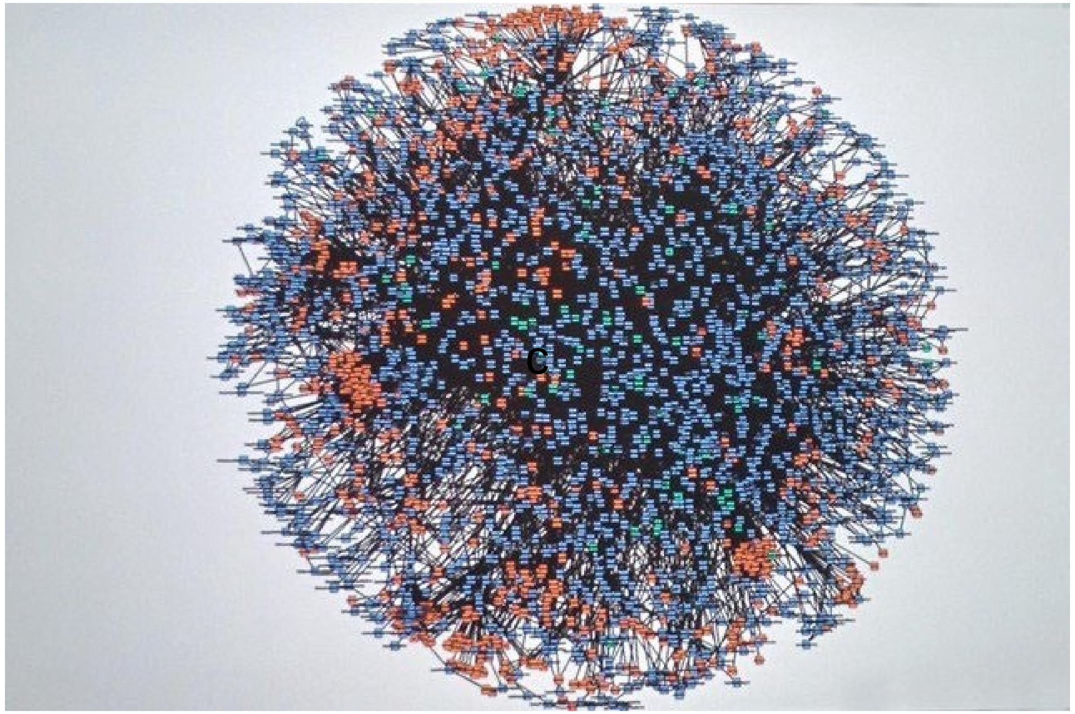
```
GET http://api.coolcars.io/cars/
GET http://api.coolcars.io/cars/{id}
DELETE http://api.coolcars.io/cars/{id}
POST http://api.coolcars.io/cars/
  {
   "make":"chevrolet",
   "model":"Silverado 3500",
   "year": 2004,
   "vin":"1GCJK33104F173427"
   }

Response: {”data”:{“id”: “8b7138db-0c7c-4e2e-8494-bd5daf1788e0”}
```

The Operations Challenge

- Processes are usually deployed in Docker or similar containers
- But we have to solve:
  - Coordinating the activity of possibly thousand of containers that need to work together
  - Creating and maintaining connections between containers
  - Ensure the whole system operates well enough to meet Service Level Agreements (SLAs)
- We need to deal with non-functional requirements
  - Loading, throughput, stress, response times
  - Disaster recovery
  - Security
- The lack of an effective way to do this was a major impediment to the deployment of microservice based applications

Site Reliability Engineering

- Practices designed to ensure large systems are operational
- Continuously checking for potential problems
- Manages a set of mitigation responses to react to problems
- Recent examples
  - Rogers Canada 2022 network failure
  - Facebook October 2021 upgrade failure
  - Check out risks.org
- As applications scale, this becomes increasingly difficult

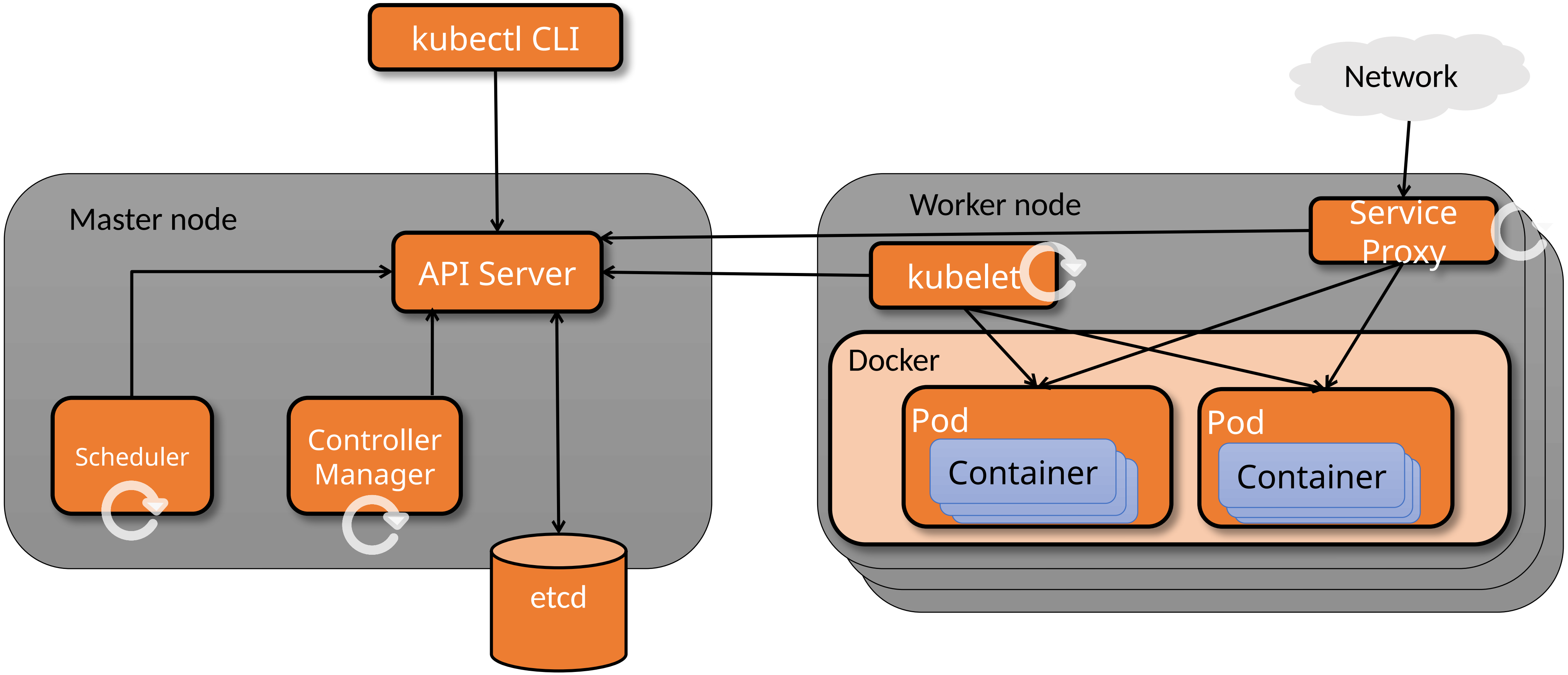


Kubernetes

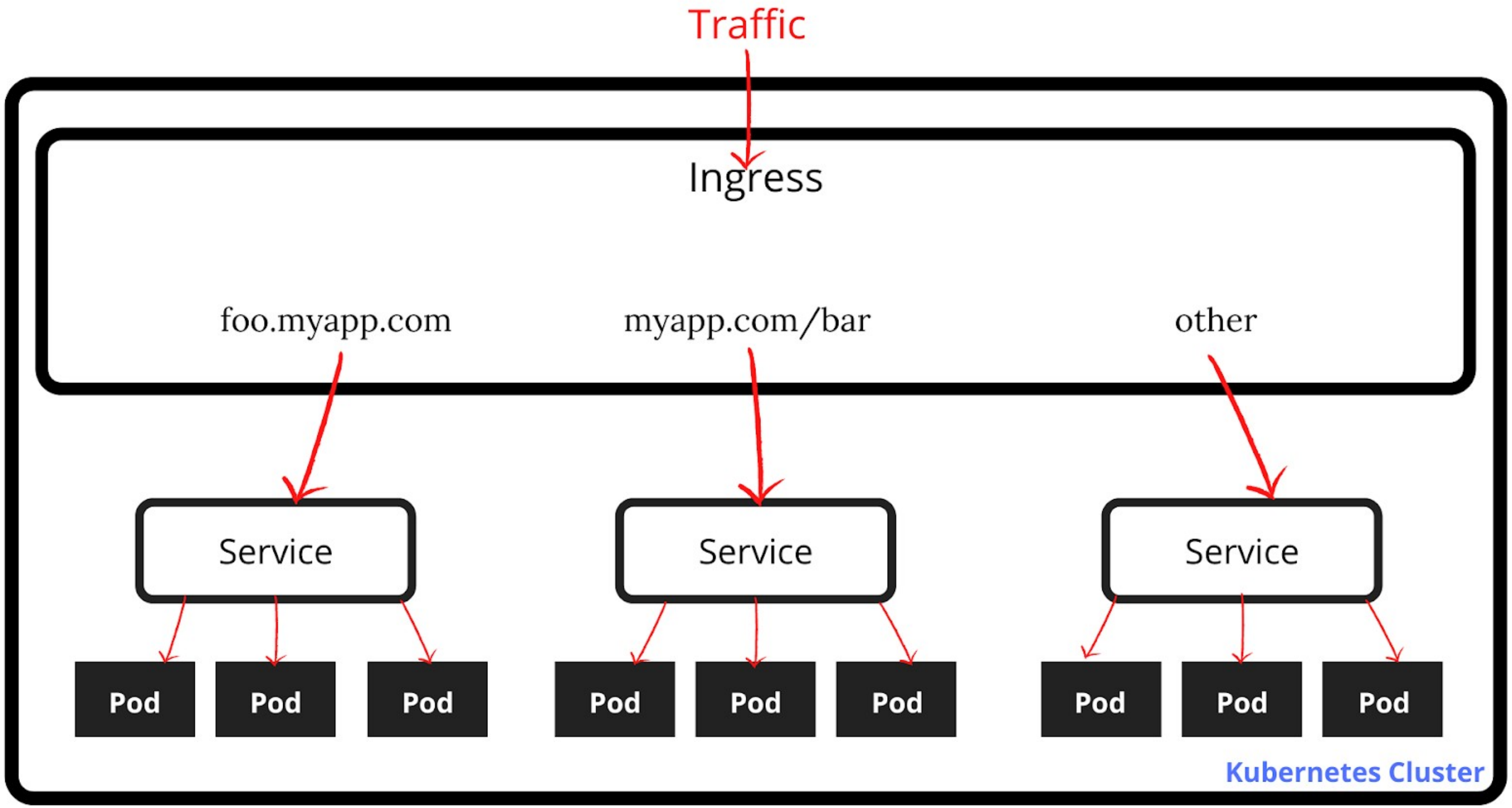
- Kubernetes is a container orchestration manager
  - Not the only manager
  - Docker Swarm does the same
  - Kubernetes is “industrial strength”
- Orchestration:
  - Manages “clusters” of containers
  - Provides service discovery
  - Manages scaling and failover
  - Works at the Ops level
  - Infrastructure as Code

Kubernetes Architecture

- Kubernetes nodes can be physical hosts or VM’s running a container-friendly Linux



Kubernetes Service

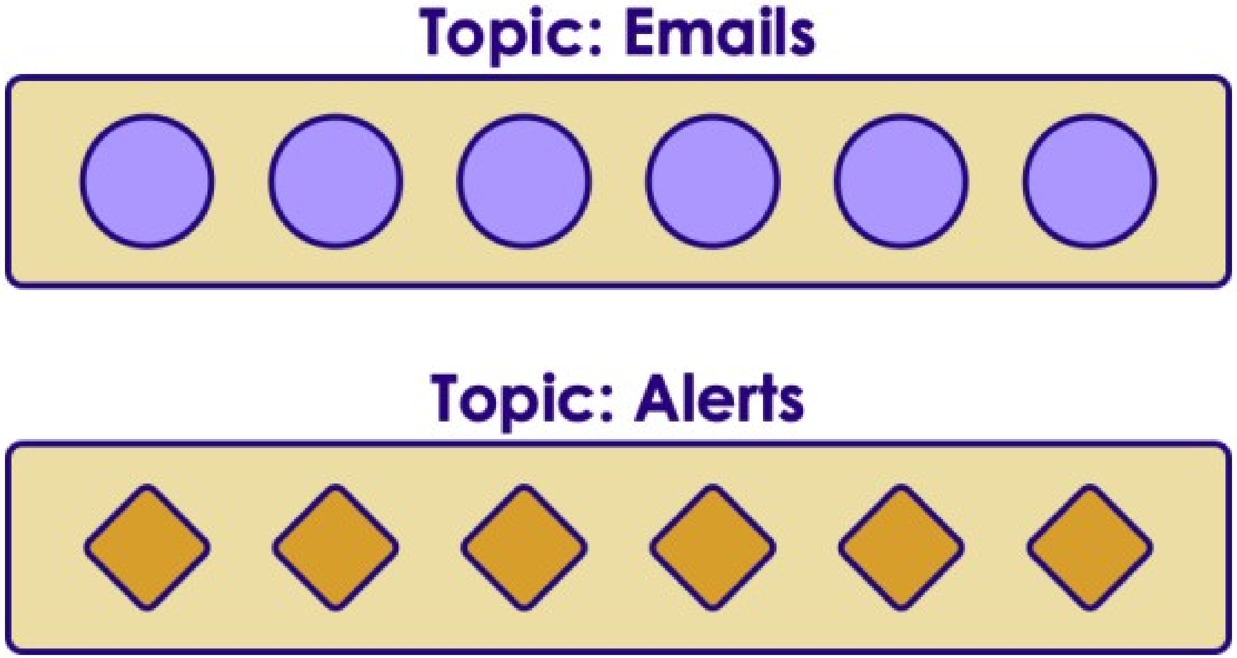


Event Based

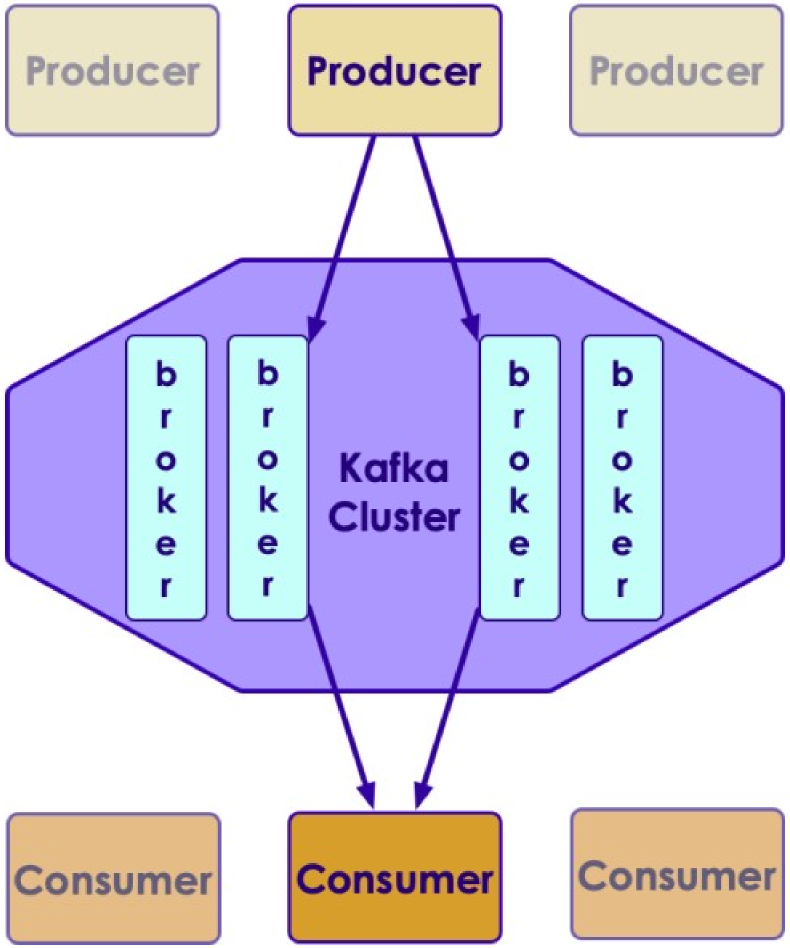
- Instead of messages, we think in terms of events
- An event some data item of interest in the domain
- Instead of a cluster, the main artifact is a stream or queue
- Publishers put events onto the queue
- Subscribers get events off of the queue
- Generally referred to as the “pub-sub” model
- The primary technology used is Kafka
    - Acts as a asynchronous buffer between publishers and subscribers

Kafka Concepts

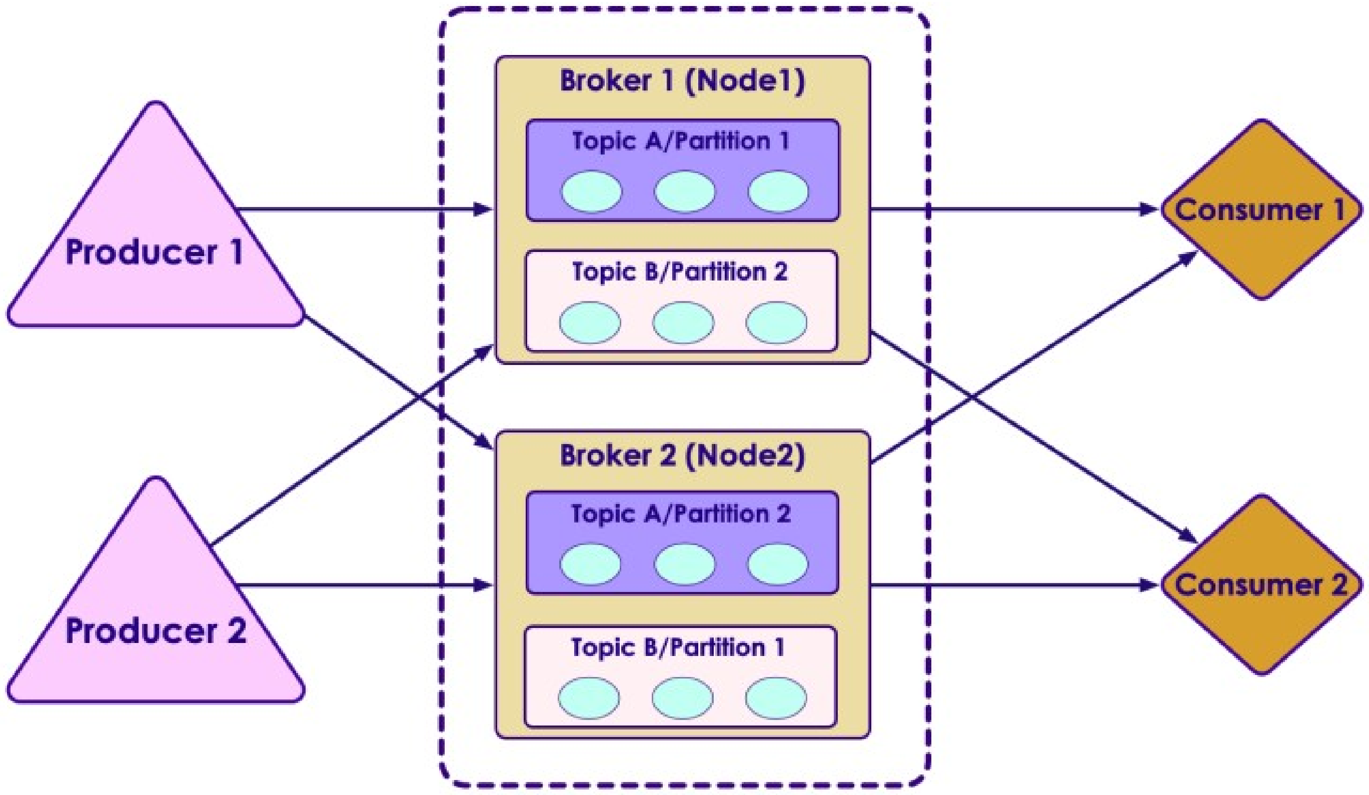
- In Kafka a basic unit of data is a 'message'
  - Message can be email / connection request / alert event
- Messages are stored in 'topics'
  - Topics are like 'queues'
  - Sample topics could be: emails / alerts



Kafka Concepts



Kafka Concepts

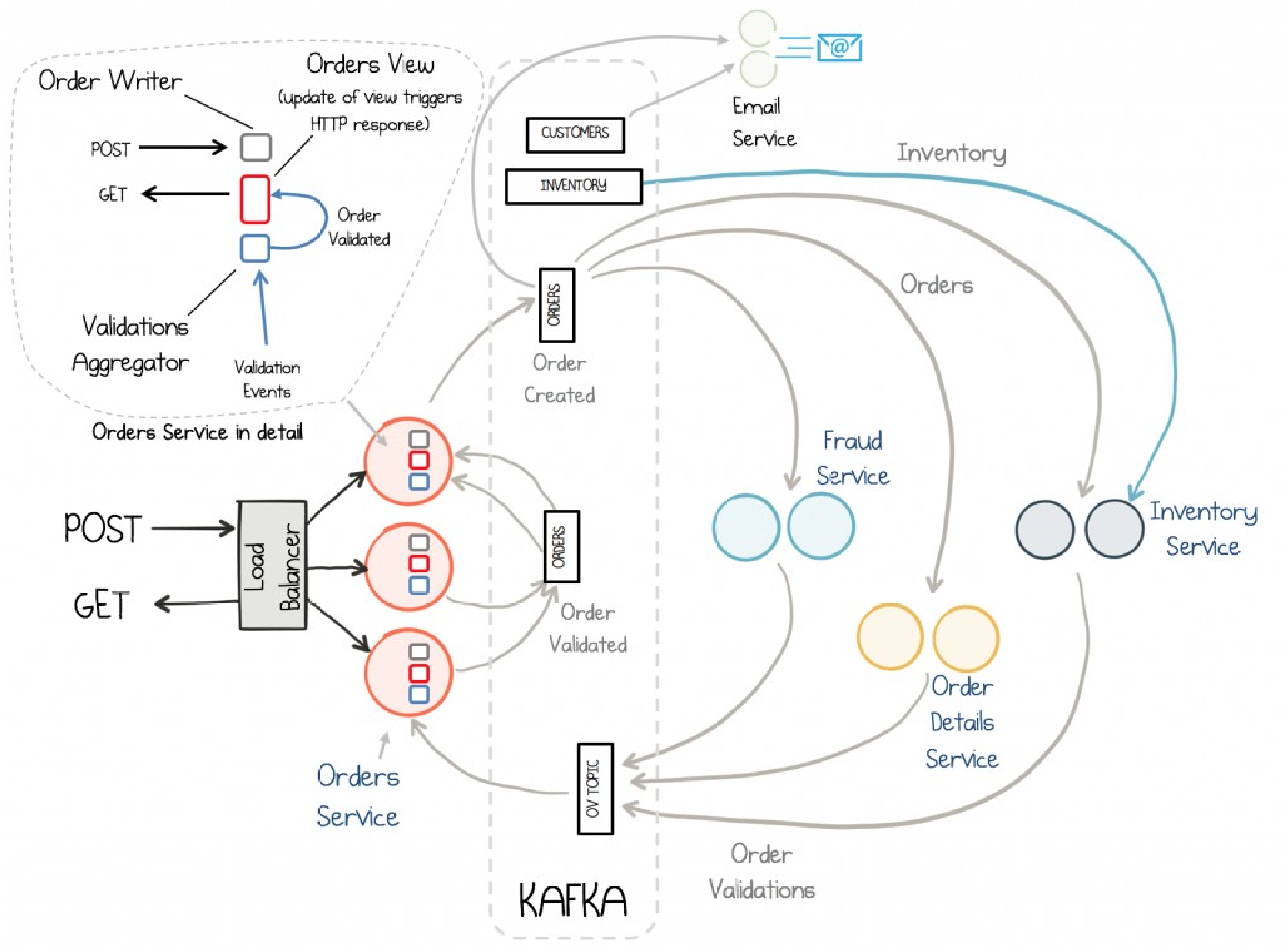


Kafka Cluster

Note: Partition replicas are not shown

○ event

Example


Order Writer
Orders View
(update of view triggers HTTP response)
POST
GET
Order Validated
Validations Aggregator
Validation Events
Orders Service in detail
POST
GET
Load Balancer
Orders Service
CUSTOMERS
INVENTORY
ORDERS
Order Created
ORDERS
Order Validated
OV TOPIC
KAFKA
Email Service
Inventory
Orders
Fraud Service
Inventory Service
Order Details Service
Order Validations

Message Protocols

- Unlike Kuber

End of Module