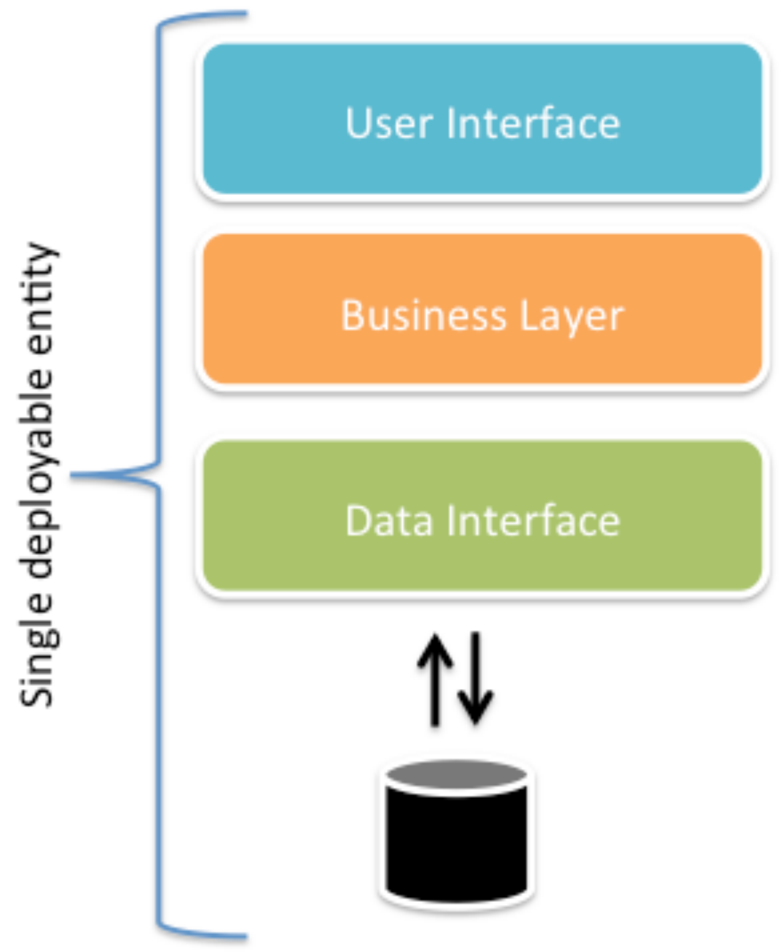
JOURNEY TO

MICROSERVICES

JOURNEY TO MICROSERVICES

MONOLITH

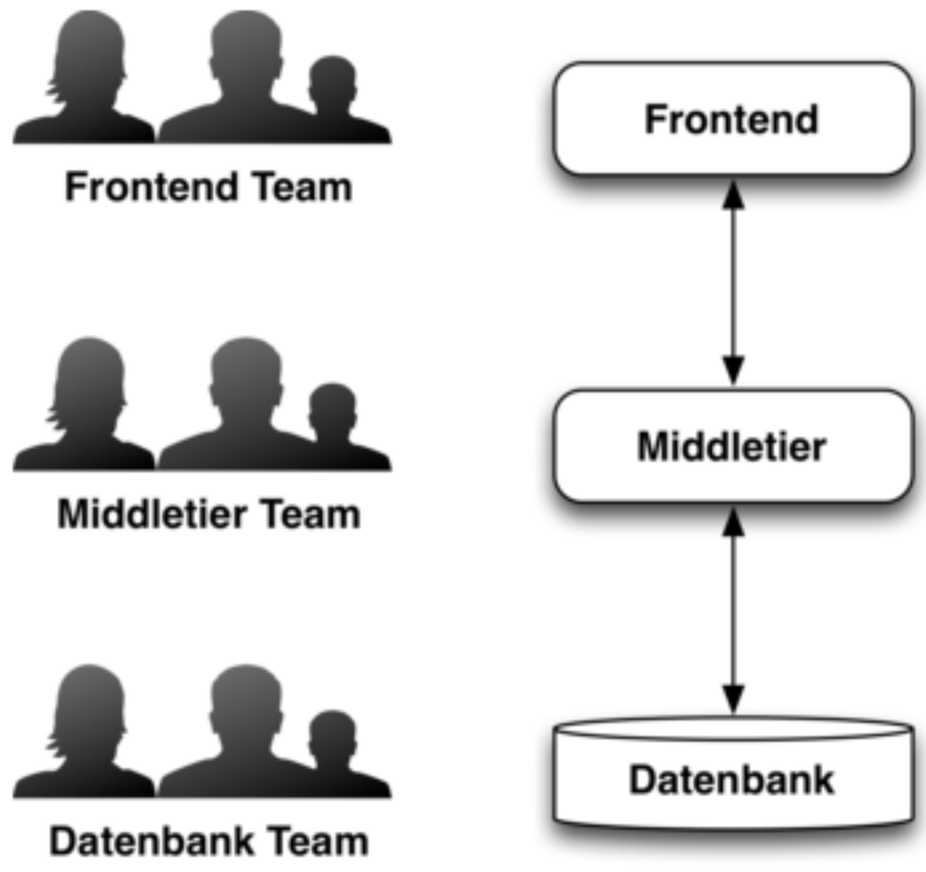
CHARACTERISTICS



- **Centralized** dependencies
- „One and Only“ **specific Tech-Stack**
- **Growing Code-Base and Complexity**
- **„One-Fits-All“ Data-Model** approach
- **Single Deployment & Runtime Unit**
- Difficult **side-effects** over time
- etc.

CONWAYS LAW

„Organizations design systems which copy the organization."



- The team structure reflects the architecture
- Technical & organisational dependencies
- „Masterplan" responsibility
- Coordination of
  - Business requirements
  - Implementation
  - Big-Bang Releases

JOURNEY TO MICROSERVICES

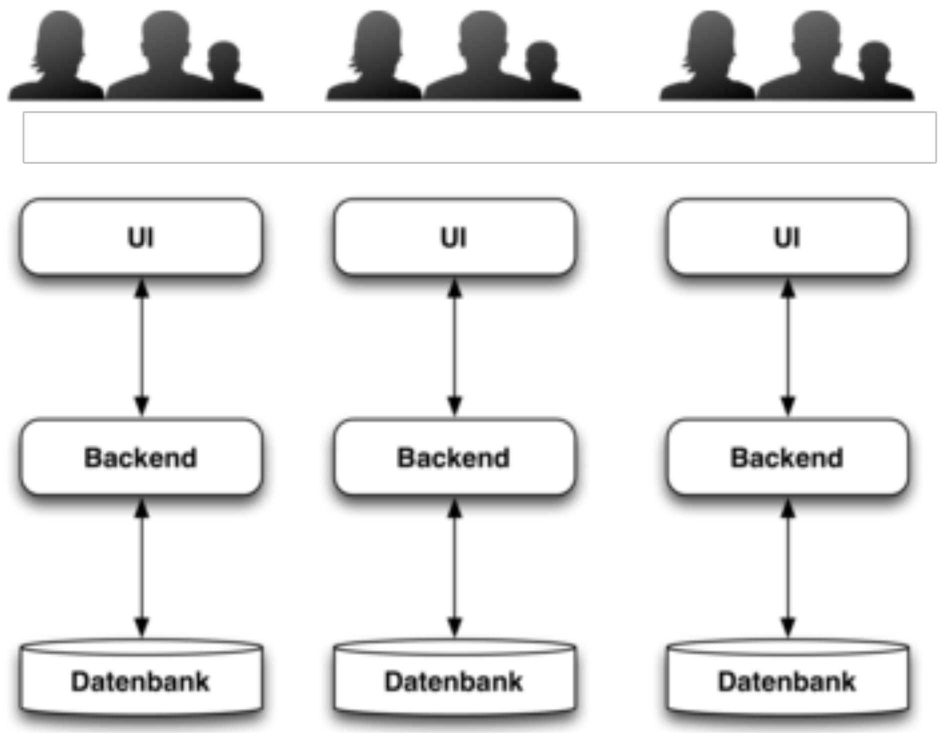
AGILITY

MIND SET

- Learn and Adopt Business-Requirements
- Design Tech-Agnostic
- System-Thinking
- Continuous Delivery

CONWAYS LAW

„Organizations design systems which copy the organization."



- Team structure reflects the architecture
- Fast moving Business-Domains
- „Business-Unit" responsibility
- Independent
    - Business requirement analysis
    - Design and Implementation
    - Releases

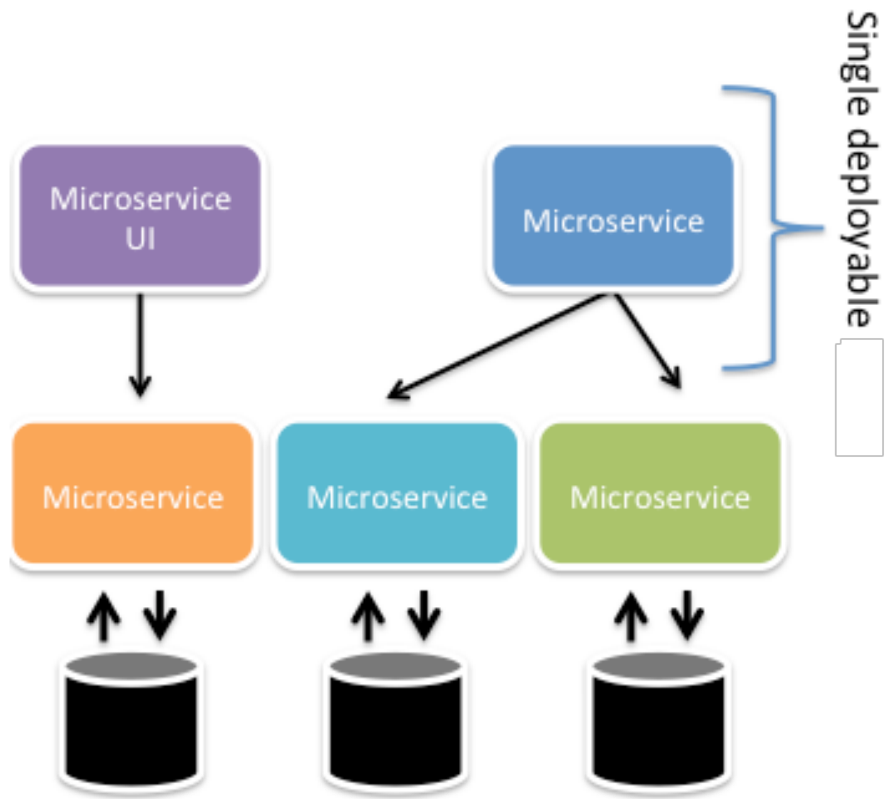
FUNCTIONAL / DOMAIN TEAMS

- **Owner** of
  - Tech-Stack
  - Architecture
  - API / UI
  - Domain-Logic
  - Data-Model(s)
  - Documentation

PLATFORM TEAMS

- **Owner** of
  - Deployment Stack
  - Persistent Stack
  - Messaging Stack
  - Tracking / Logging / Statistics Stack
  - UI / UX Design Guides

CHARACTERISTICS



- **System of Systems**
- **Risk Diversification**
- **Tech-Stack Diversification**
- **Architecture Diversification**
- **Model Diversification**
- **Location-Transparency**
- **Fast** and **Independent Delivery**
- **System-Thinking** Philosophy
- **Sync + Async** communication

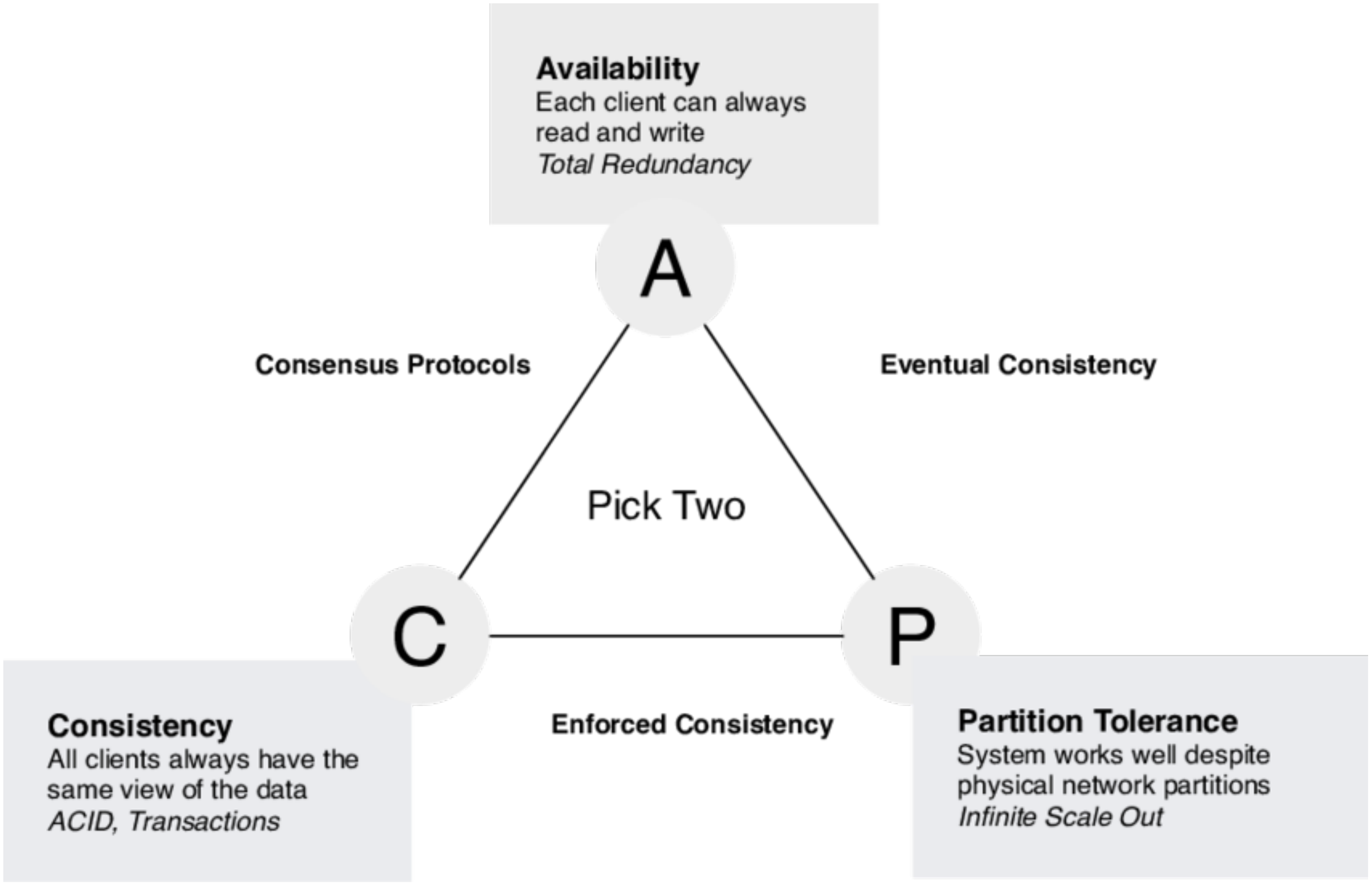
JOURNEY TO MICROSERVICES

PRINCIPLES

SYSTEM OF SYSTEMS

- Distributed by Design
- Choose CAP
- Independent & Isolated
- Integration Layer
- Platform Concepts
- Tech-Agnostic

CAP – RECAP – PICK TWO



JOURNEY TO MICROSERVICES

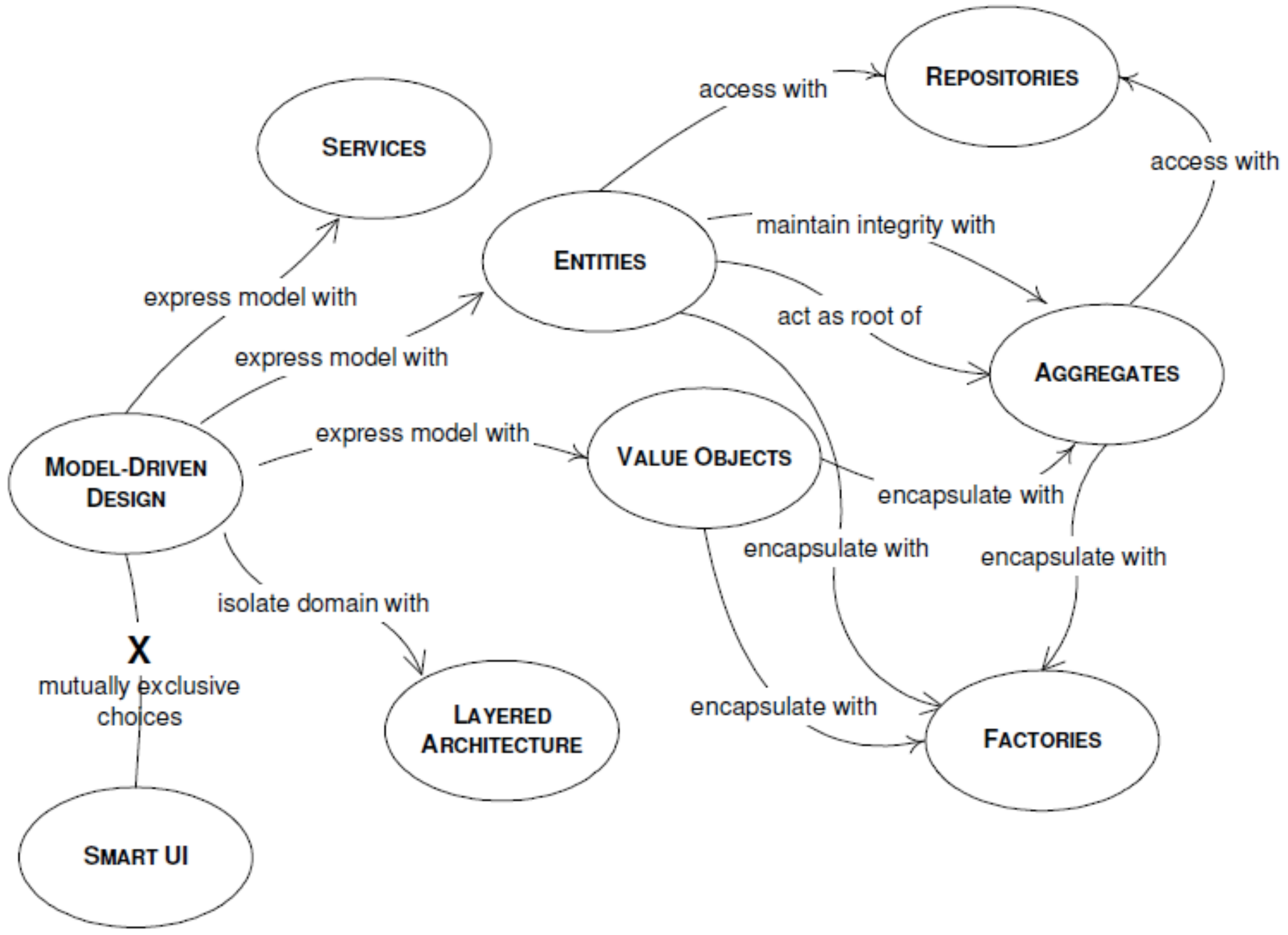
DDD TECHNIQUES

DDD

- **Language** - Consistent syntax and semantic / Glossary
- **Core Domain** - Business strategy
- **Sub-Domain** - Provider-, Adapter-, or External-Systems
- **Bounded Context** - Commands, Events, Entities and Aggregates
- **Application-Services** Anti-Corruption, Process-Manager, Sagas

BOUNDED CONTEXT

- Tackling Complexity
- Decomposition & Composition
- Follows Domain-Experts
- Business-Model Integrity
- Consistency Boundaries
- Smart UI/UX

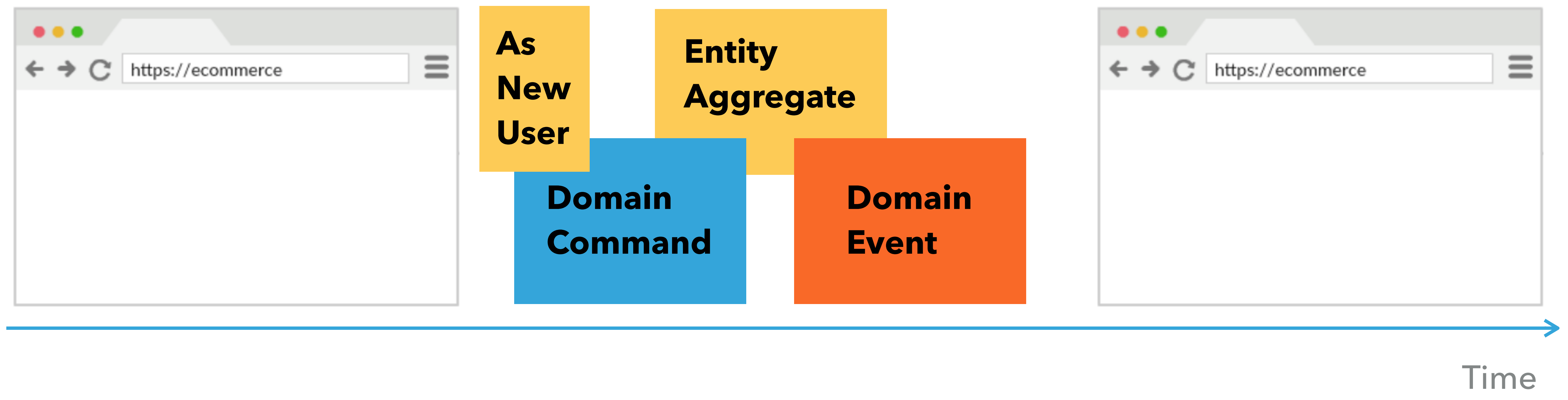


EVENT-STORMING COMPACT

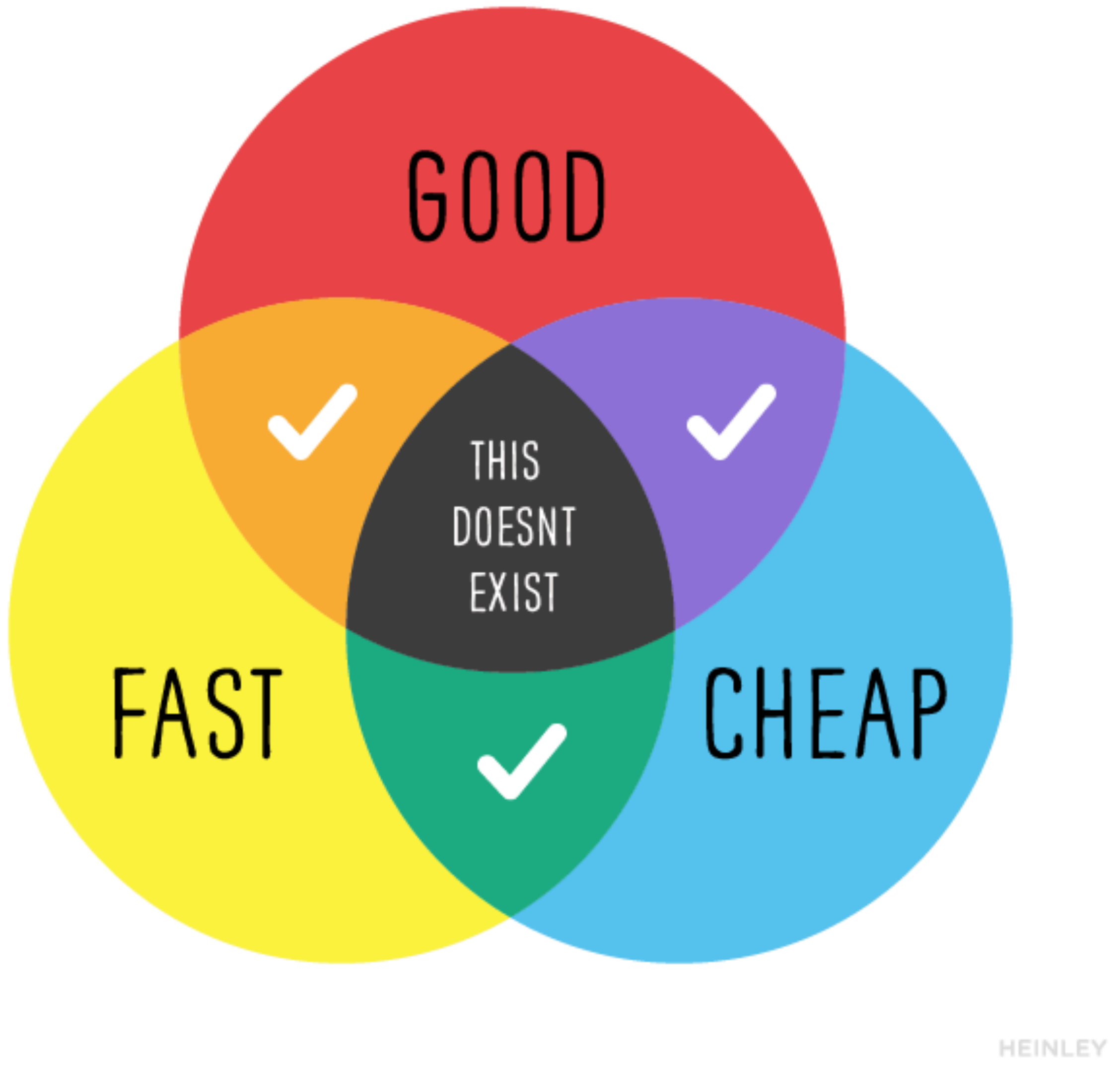
- **Aggregate (yellow)** – Processes Commands and emits Domain Events (DDD pattern)
- **Command (blue)** – A request to do something isolated domains
- **Domain Event (orange)** – Something that happened, past tense + verb (Event Sourcing/CQRS pattern)
- **External System (purple)** – Just what you would expect
- **Policy (pink)** – Algorithm or decision, manual or automated (Strategy Pattern)
- **Read Model (green)** – Data needed to make a decision (CQRS concept)
- **User (small yellow)** – Actor, user, persona, or role
- **User Interface (white)** – Just what you would expect

EVENT-STORMING COMPACT

- Start with Domain-Event (Behavior)
- Add Domain-Command (Trigger/Action)
- Add Entity/Aggregate (Data-Container)
- Optional add User (Roles)



TIMEBOXING - PICK TWO



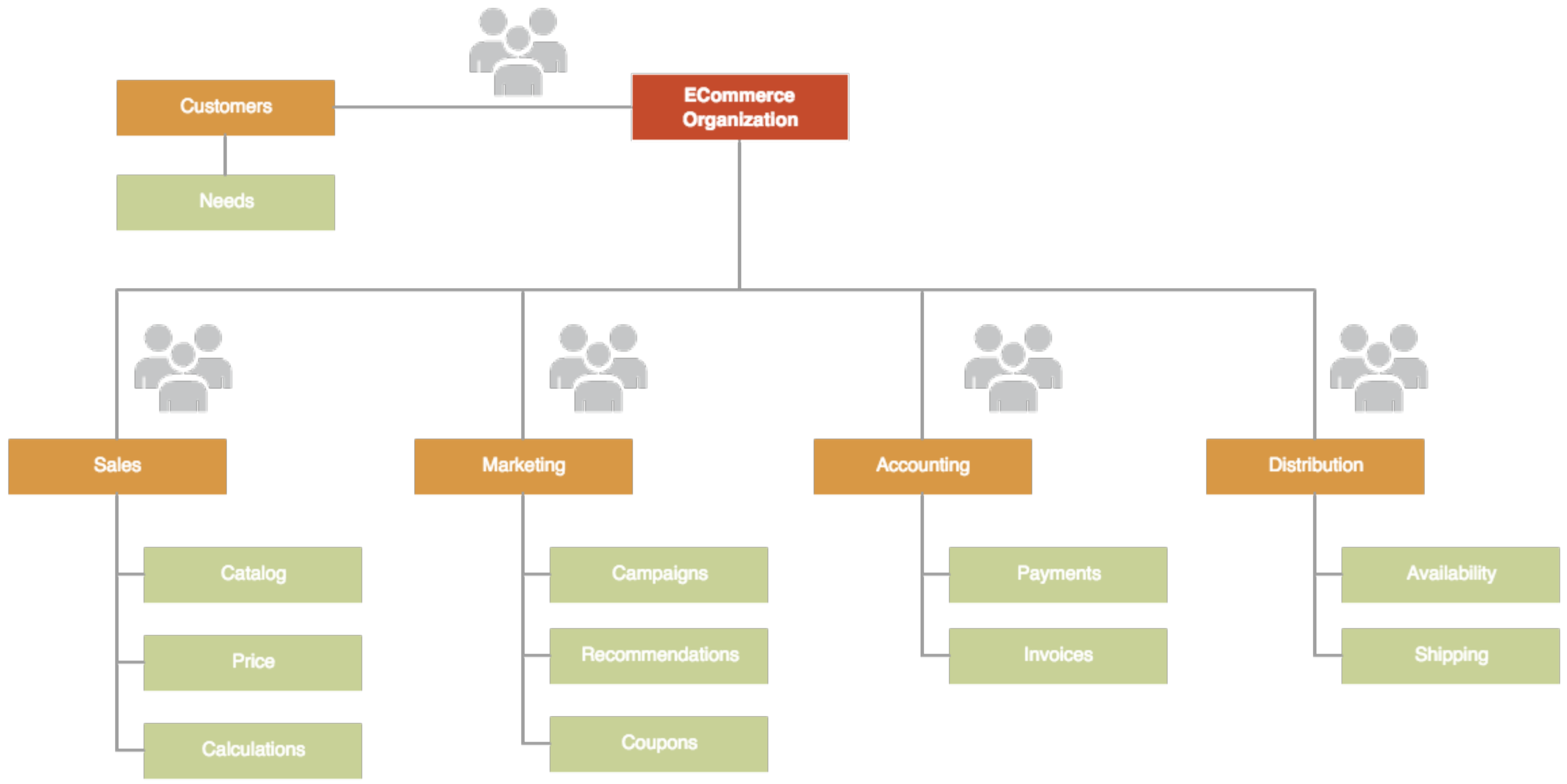
TIMEBOXING - ESTIMATE

Component Type	Simple (hours)	Complex (hours)
Domain Event		
Command		
Aggregate		

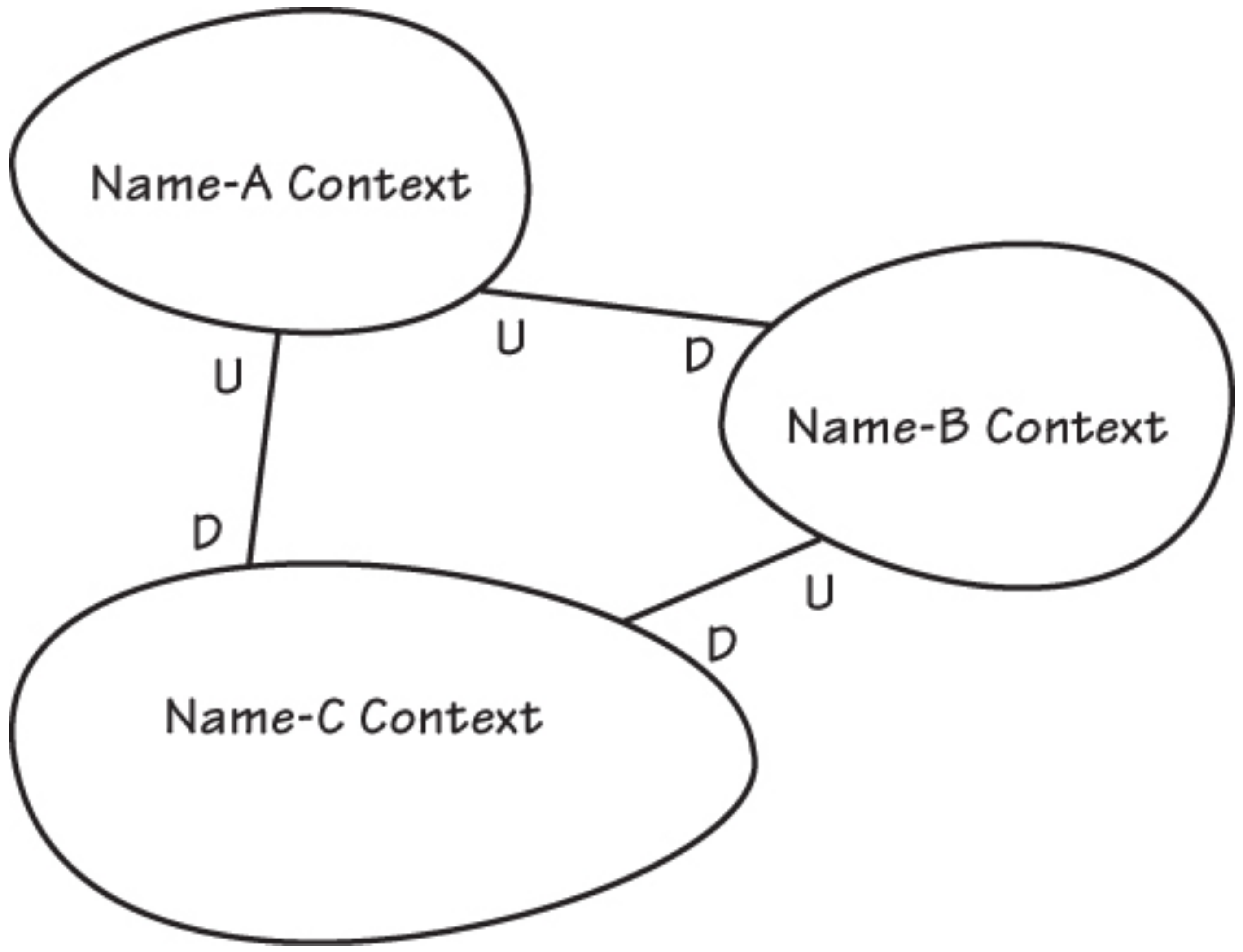
TIMEBOXING - TASKS

To Do	In Progress	Done
ABC Events		
ABC Commands		
ABC Aggregate		

BUSINESS DECOMPOSITION



CONTEXT MAPPING



- Relations
- Processes
- Policies
- Dependencies
- Core-Domain
- Sub-Domains
- External
- ACLs
- Roles

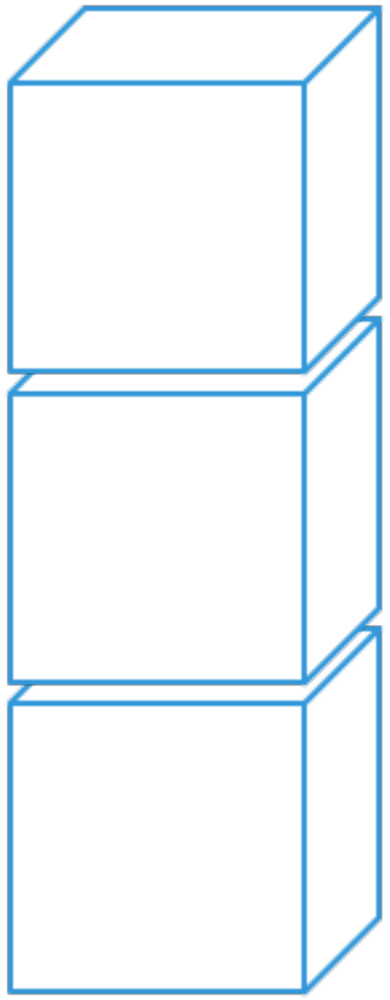
BUSINESS DECOMPOSITION

- **Domain Commands & Events**
  Represents atomic and idempotent operations
- **Entities / Aggregates**
  Consistency and Integrity Boundary
- **Sagas / Process Managers**
  Long Running Processes, Correlation, Transactions, Compensations
- **Application Services / MicroService**
  Hosting and API-Contract

JOURNEY TO MICROSERVICES

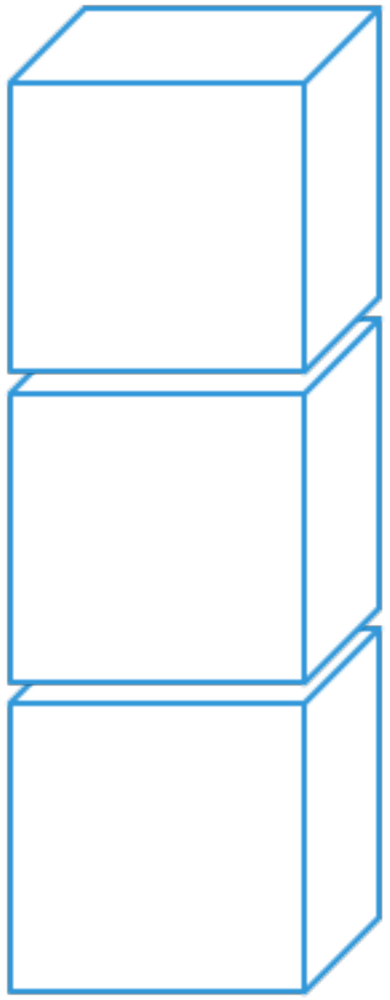
SELF-CONTAINED

CHARACTERISTICS



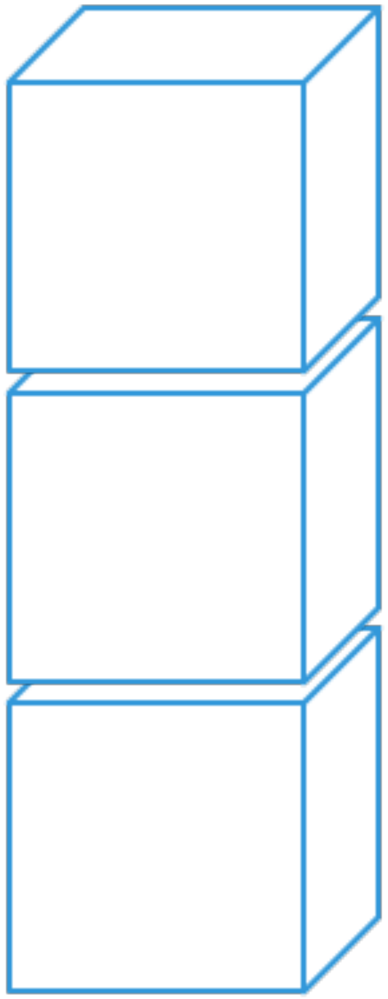
- Independent Deployable
- Independent Implementation
- Independent Maintenance
- Independent Ownership
- Independent from other (Sub)Domains

CHARACTERISTICS



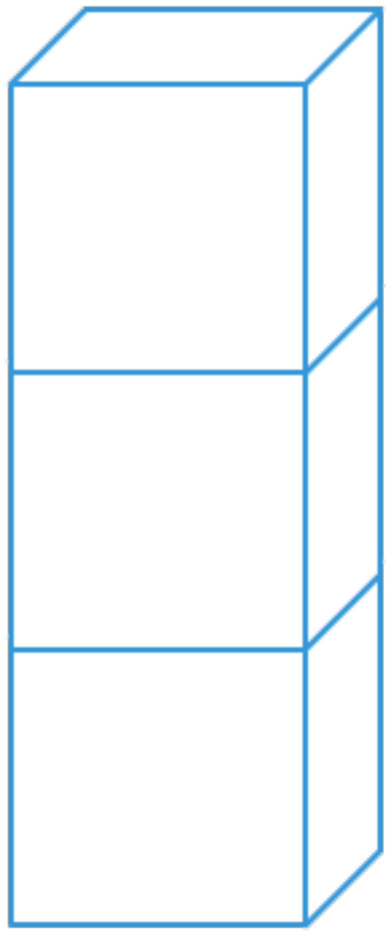
- Unix philosophy
- Isolated Environment
- Integrity & Consistency Guarantees
- Model Owner
- Data Owner

APPLICATION SERVICE



- **Sync + Async** Communication
- **Dedicated Contract**
- **Domain-Centric** Operations (Commands)
- Publish/Subscribe **Event-Based-API**
- Aggregates/Feeds (Req/Res) **HTTP-API**
- **Partial-UI** (HTML-GET/FORM-POST) **API**

12-FACTOR-APPS



- Explicit **Port Bindings**
- **Location Transparency**
- **Stateless / Temporary State**
- **Graceful** Startup / Shutdown
- **Scale via Processes**
- Config via **Environment-Variables**
- Log / Tracing to **StdOut/StdErr**

DESIGN RULES

- **Atomic** & **Idempotent** Operations
- **Vertical-Slices of** Business-Requirements
- **Aggregates** instead of Relations
- **Replication** instead of Normalization
- **Horizontal** instead of vertical **scaling**
- **Distribution** instead centralization
- **Independent deployment** units
- **Relaxed Consistency** instead transactions

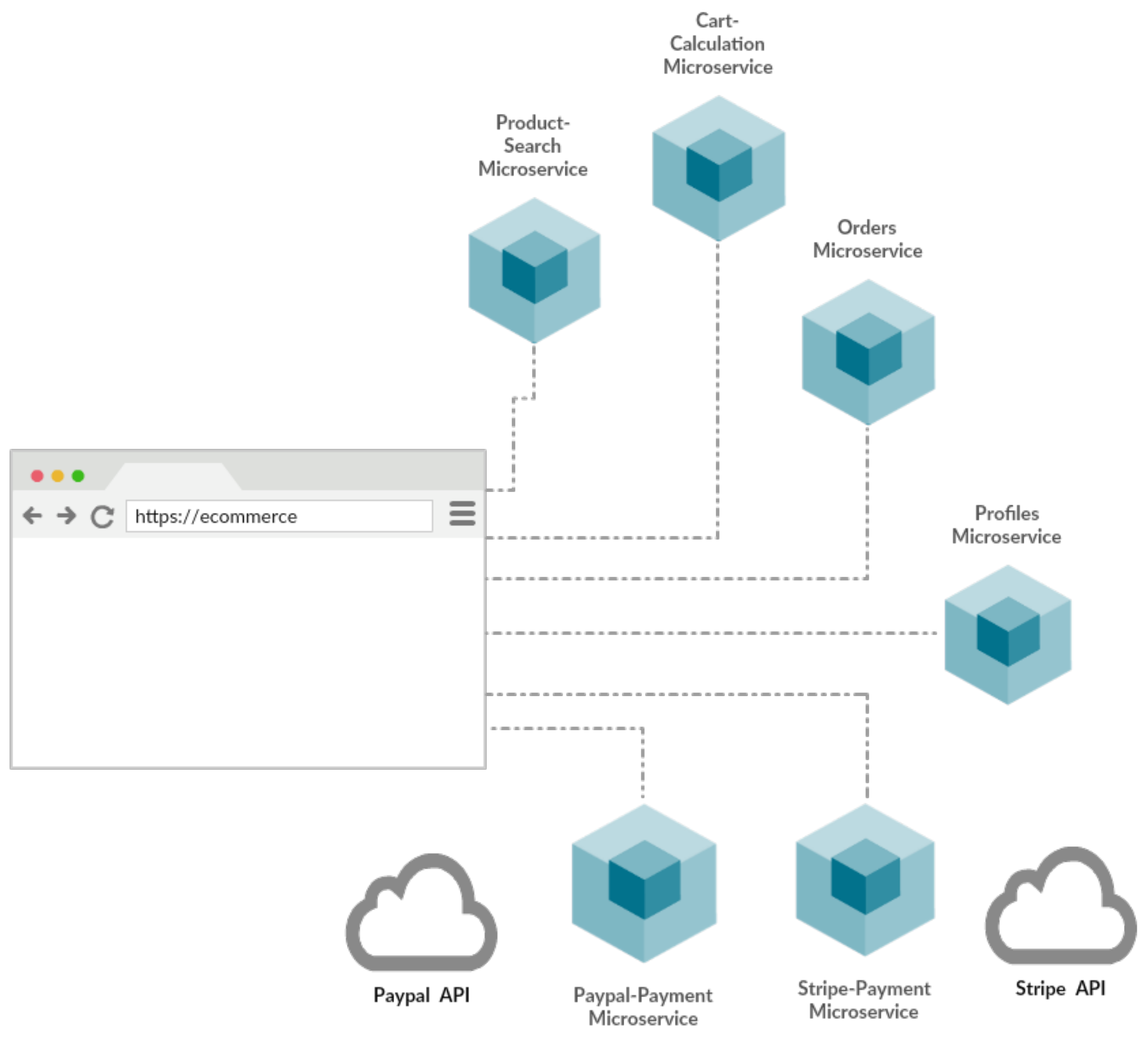
JOURNEY TO MICROSERVICES

INTEGRATION + PATTERNS

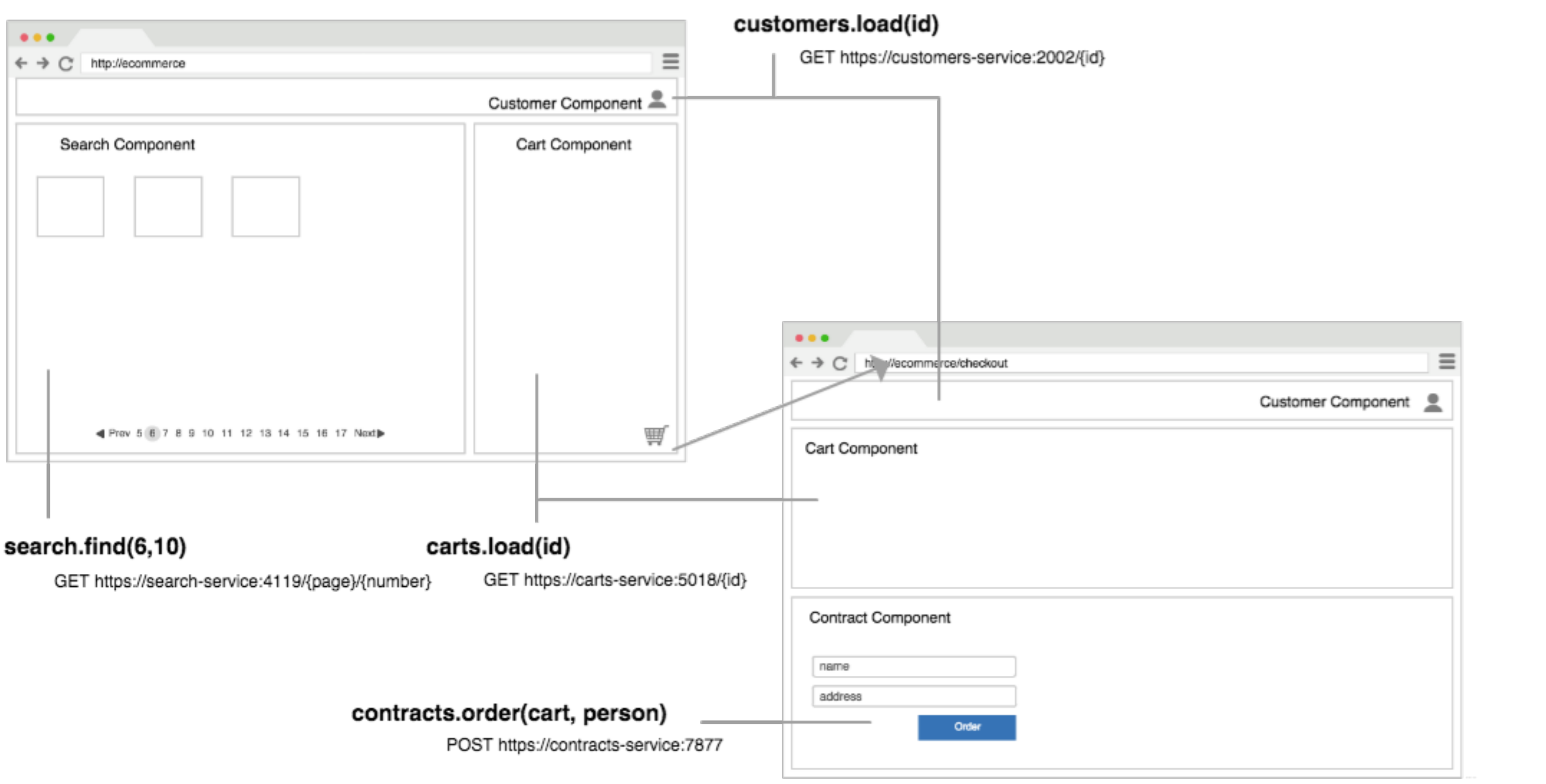
OVERVIEW

- Composite UI / Task-Based UI
- Async Communication via HTTP
- HTTP-Feeds
- RESTful APIs
- API-Gateway (Hypermedia)
- API-Gateway (GraphQL)
- Auth with JSON Web Token
- CQRS
- Event-Sourcing
- Process Manager / Sagas
- Data Replication
- Extract-Transform-Load
- Service Discovery
- Function-as-a-Service

COMPOSITION (COMPOSITE UI)


Cart-Calculation Microservice
Product-Search Microservice
Orders Microservice
Profiles Microservice
https://ecommerce
Paypal API
Paypal-Payment Microservice
Stripe-Payment Microservice
Stripe API

COMPOSITE UI



customers.load(id)

GET https://customers-service:2002/{id}

search.find(6,10)

GET https://search-service:4119/{page}/{number}

carts.load(id)

GET https://carts-service:5018/{id}

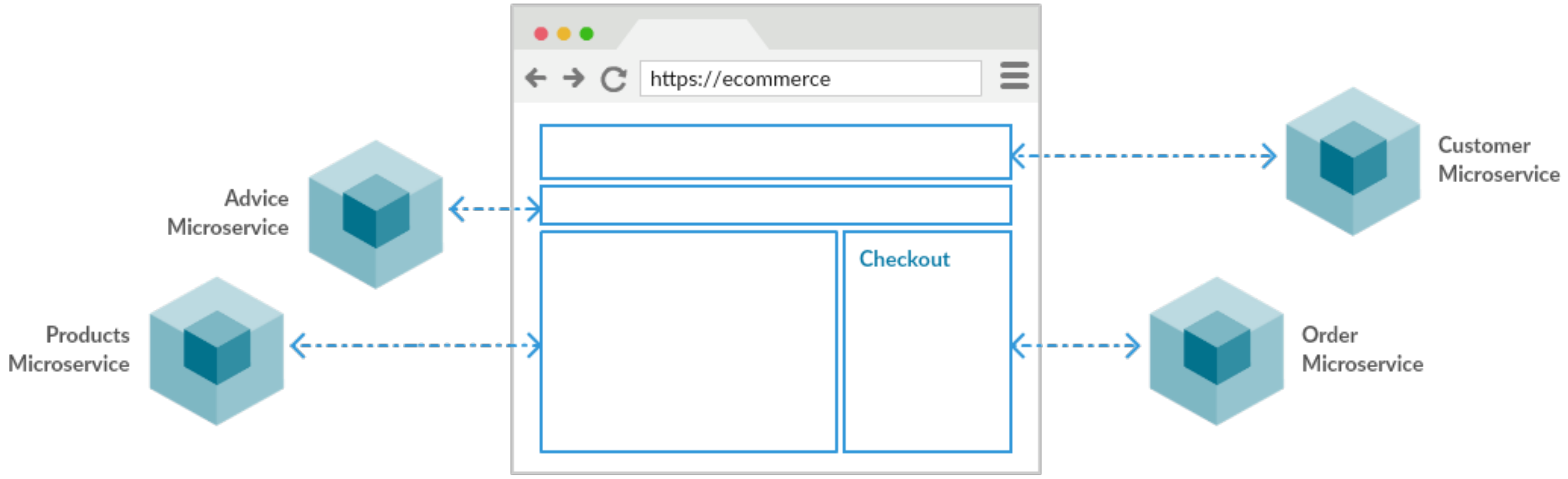
contracts.order(cart, person)

POST https://contracts-service:7877

COMPOSITE UI

- Web-Views (WPF, WinForms, Native Mobile)
- Web-Links / Web-Forms
- AJAX-Web-Container
- HTTP-Client in Single Page Applications

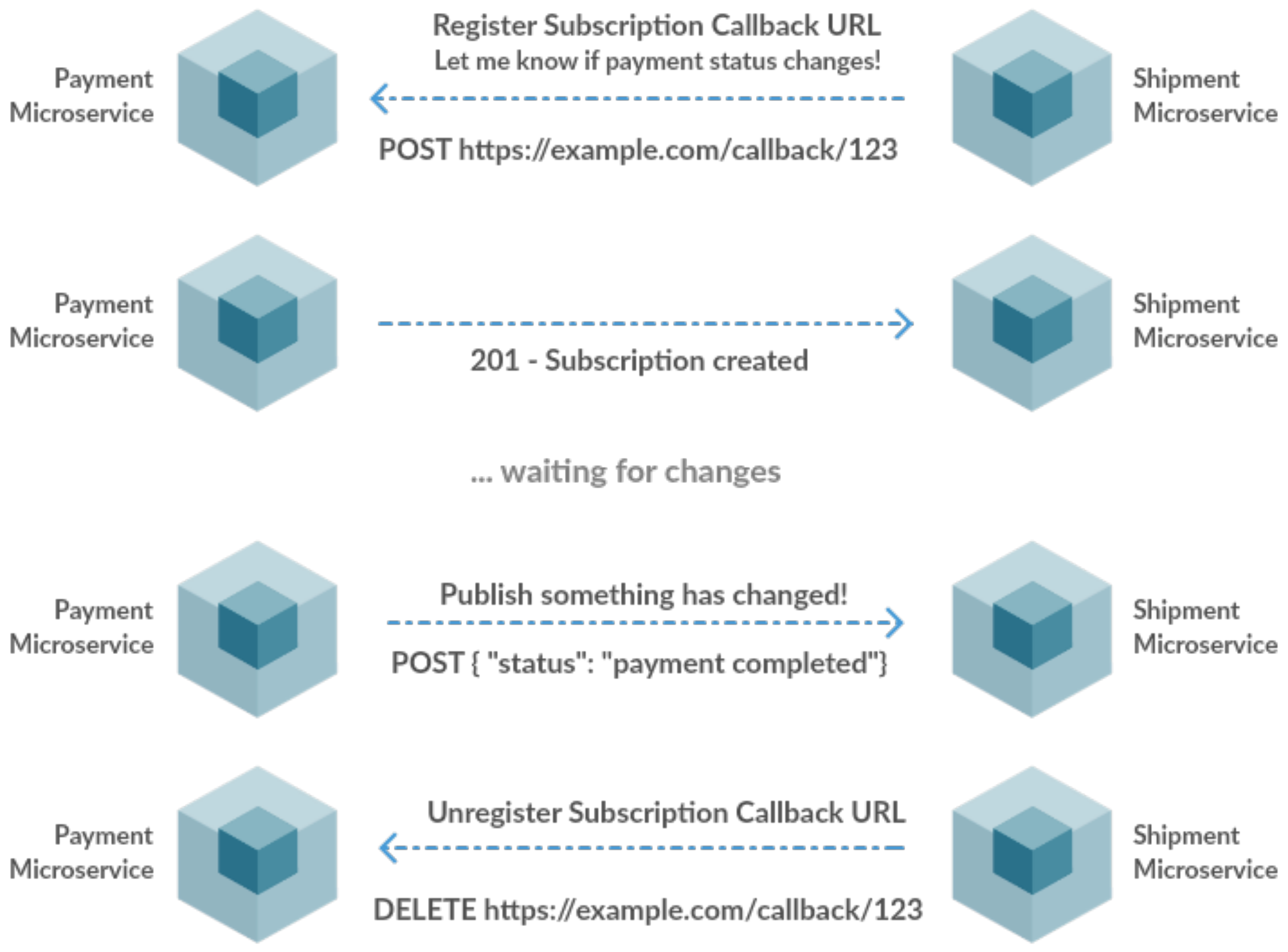
TASK-BASED UI



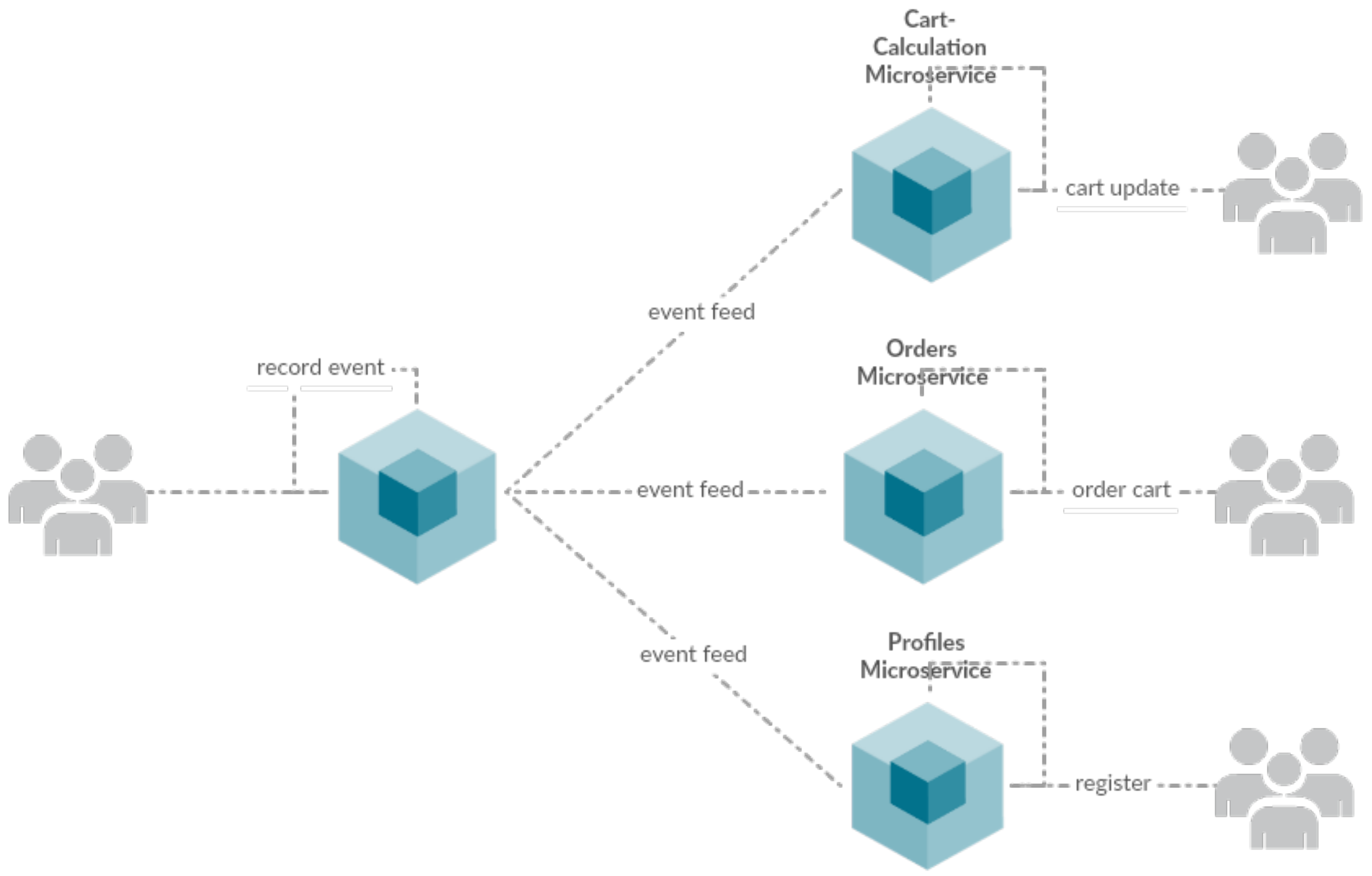
ASYNC COMMUNICATION VIA HTTP

- Long-Polling
- HTTP-Streams (EventSource)
- WebHooks / HTTP-Subscriptions
- HTTP (Atom) Feeds

WEBHOOKS / HTTP-SUBSCRIPTIONS


Payment Microservice
Register Subscription Callback URL
Let me know if payment status changes!
POST https://example.com/callback/123
Shipment Microservice
Payment Microservice
201 - Subscription created
Shipment Microservice
... waiting for changes
Payment Microservice
Publish something has changed!
POST { "status": "payment completed"}
Shipment Microservice
Payment Microservice
Unregister Subscription Callback URL
DELETE https://example.com/callback/123
Shipment Microservice

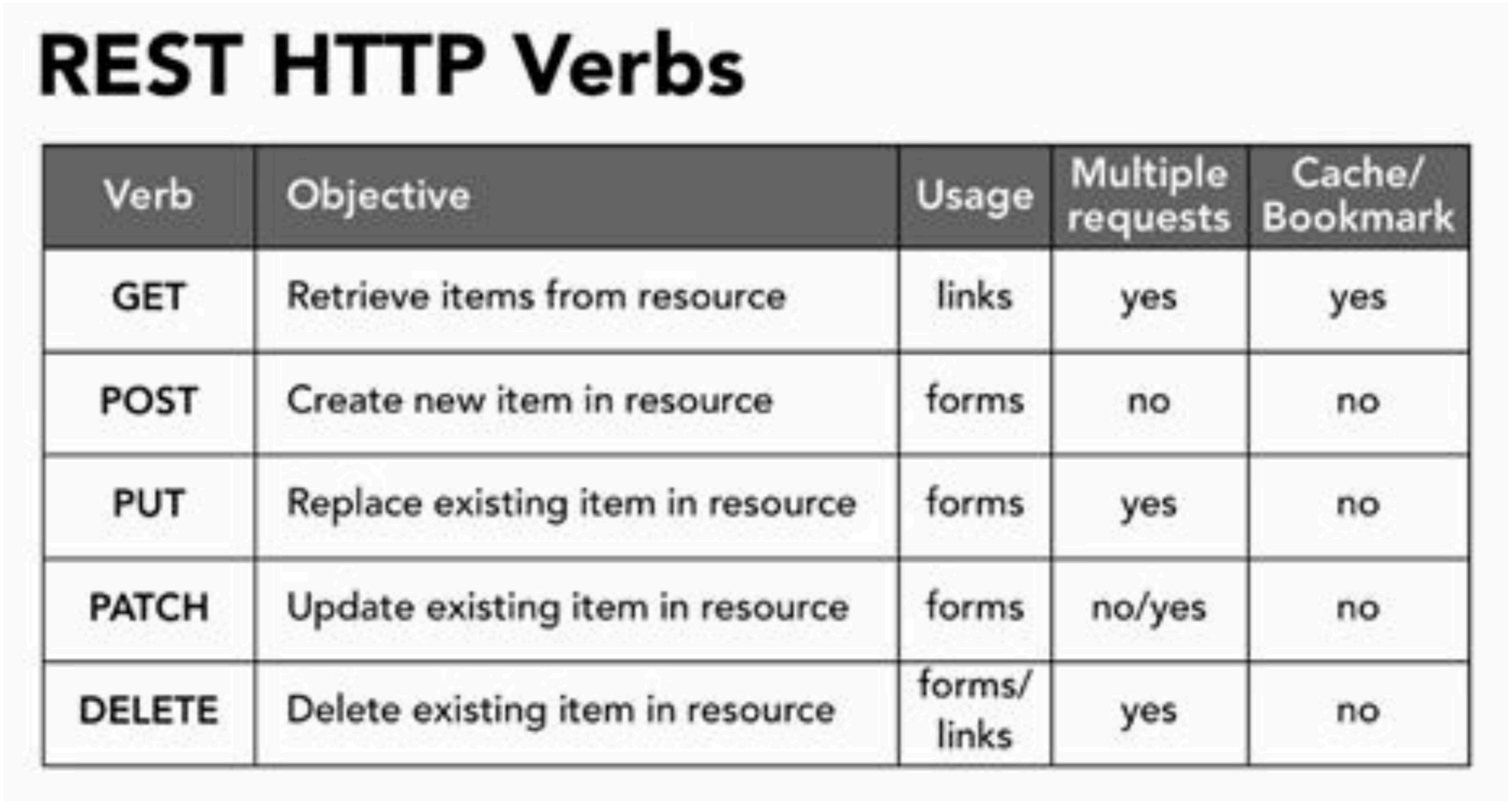
HTTP (ATOM/JSON) & FEEDS (EVENT-LOG)



RESTFUL - REPRESENTATIONAL STATE TRANSFER

- **Resource** oriented
- **Stateless** communication
- **Conventions** over HTTP
- **Cacheable** Request / Response
- **Versioning**
  (e.g. via DNS, URL, Headers)
- **Self-Descriptive**
  Content Formatting, Resource Locations, etc.

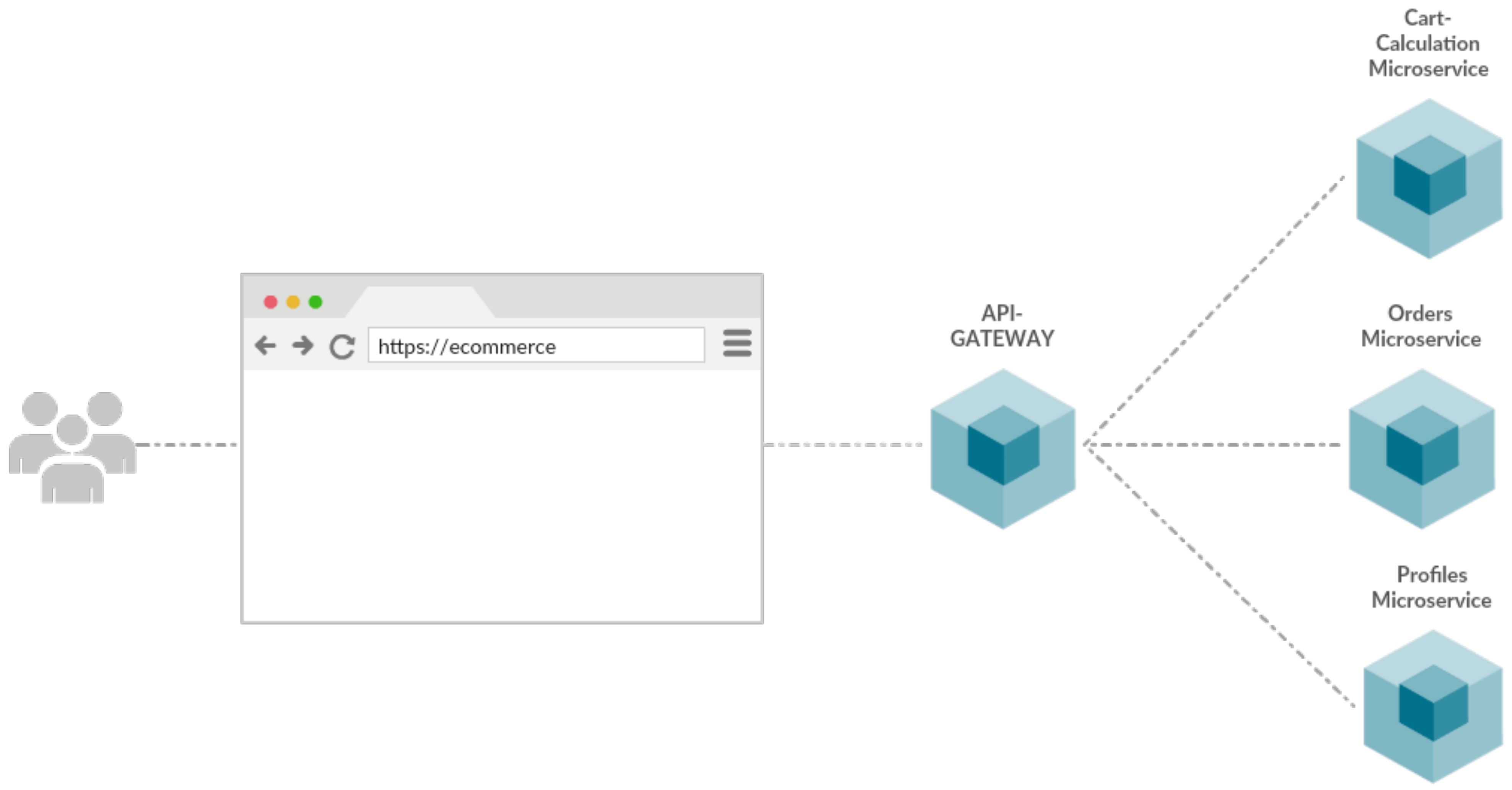
RESTFUL APIS



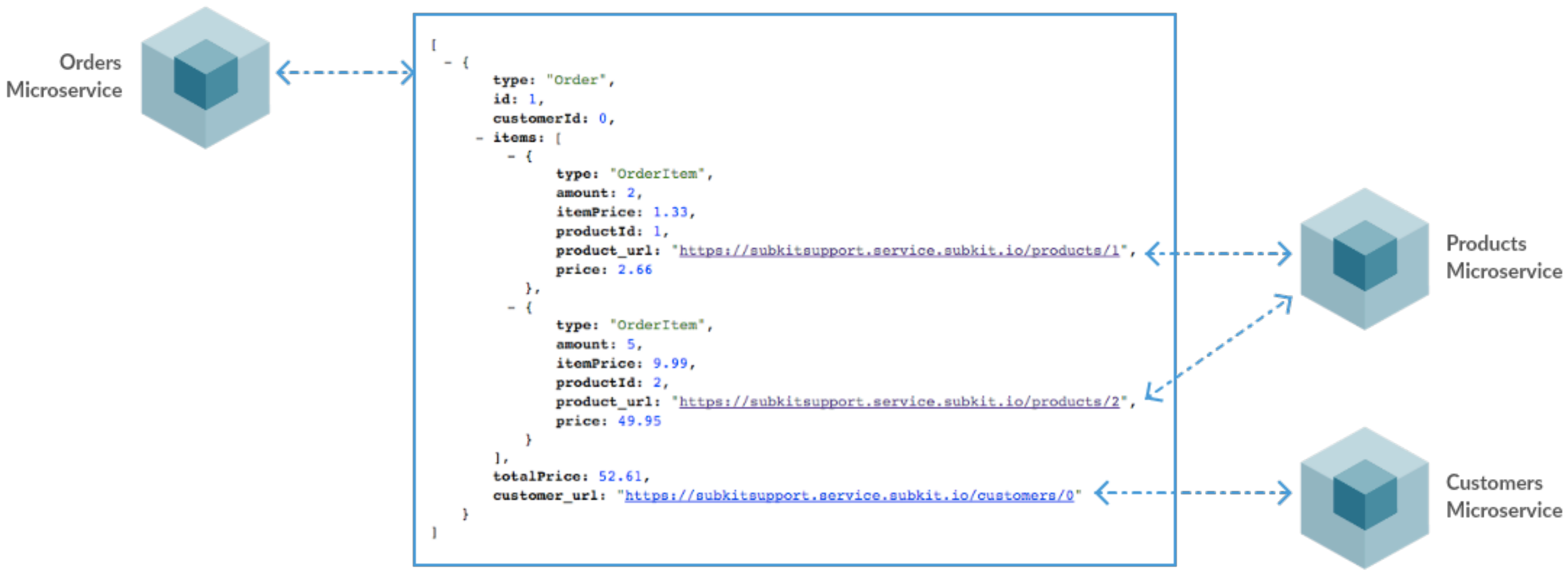
REST HTTP Verbs

Verb	Objective	Usage	Multiple requests	Cache/ Bookmark
GET	Retrieve items from resource	links	yes	yes
POST	Create new item in resource	forms	no	no
PUT	Replace existing item in resource	forms	yes	no
PATCH	Update existing item in resource	forms	no/yes	no
DELETE	Delete existing item in resource	forms/ links	yes	no

API GATEWAY (HYPERMEDIA)

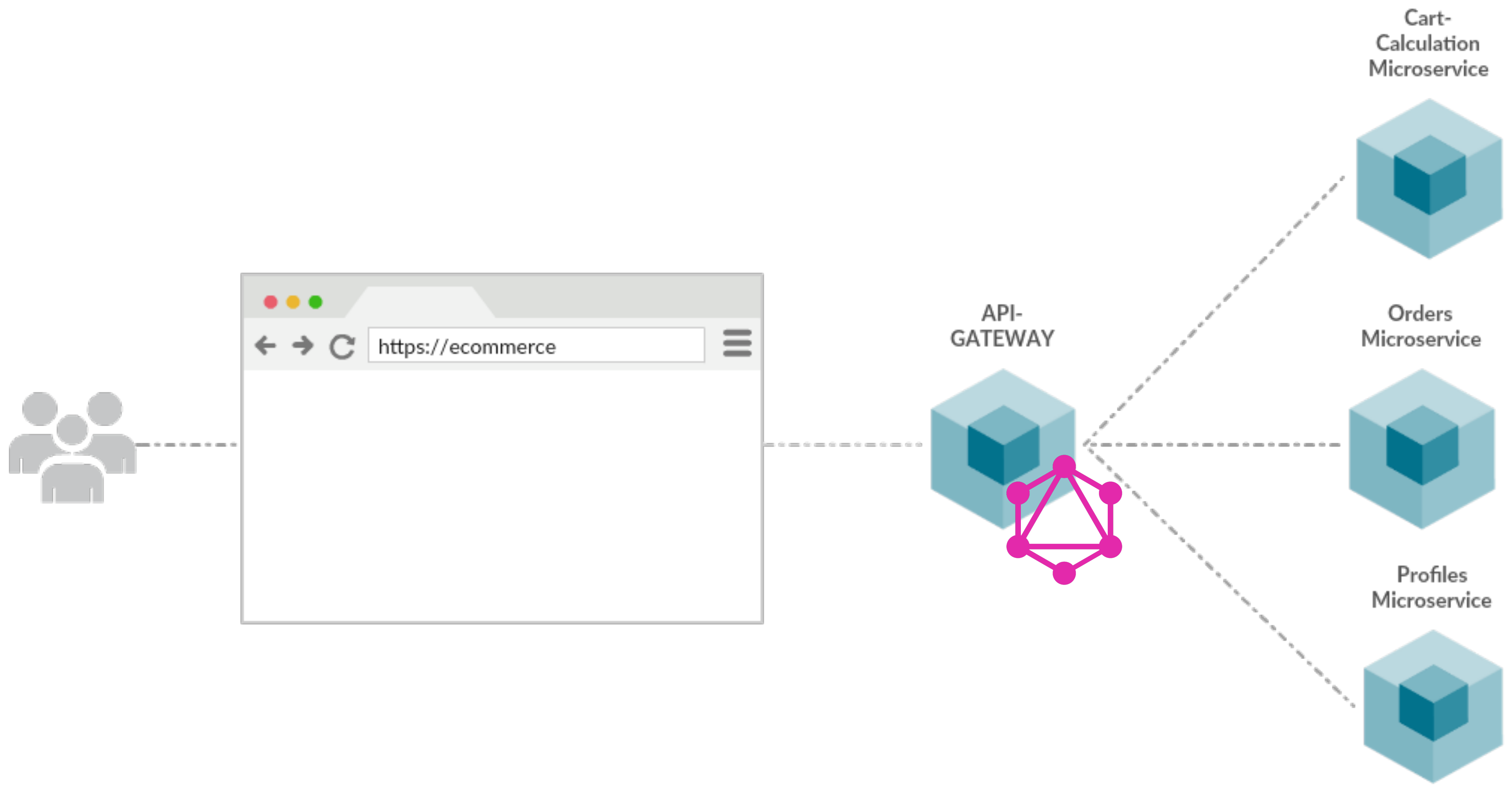


HYPERMEDIA AS THE ENGINE OF APPLICATION STATE

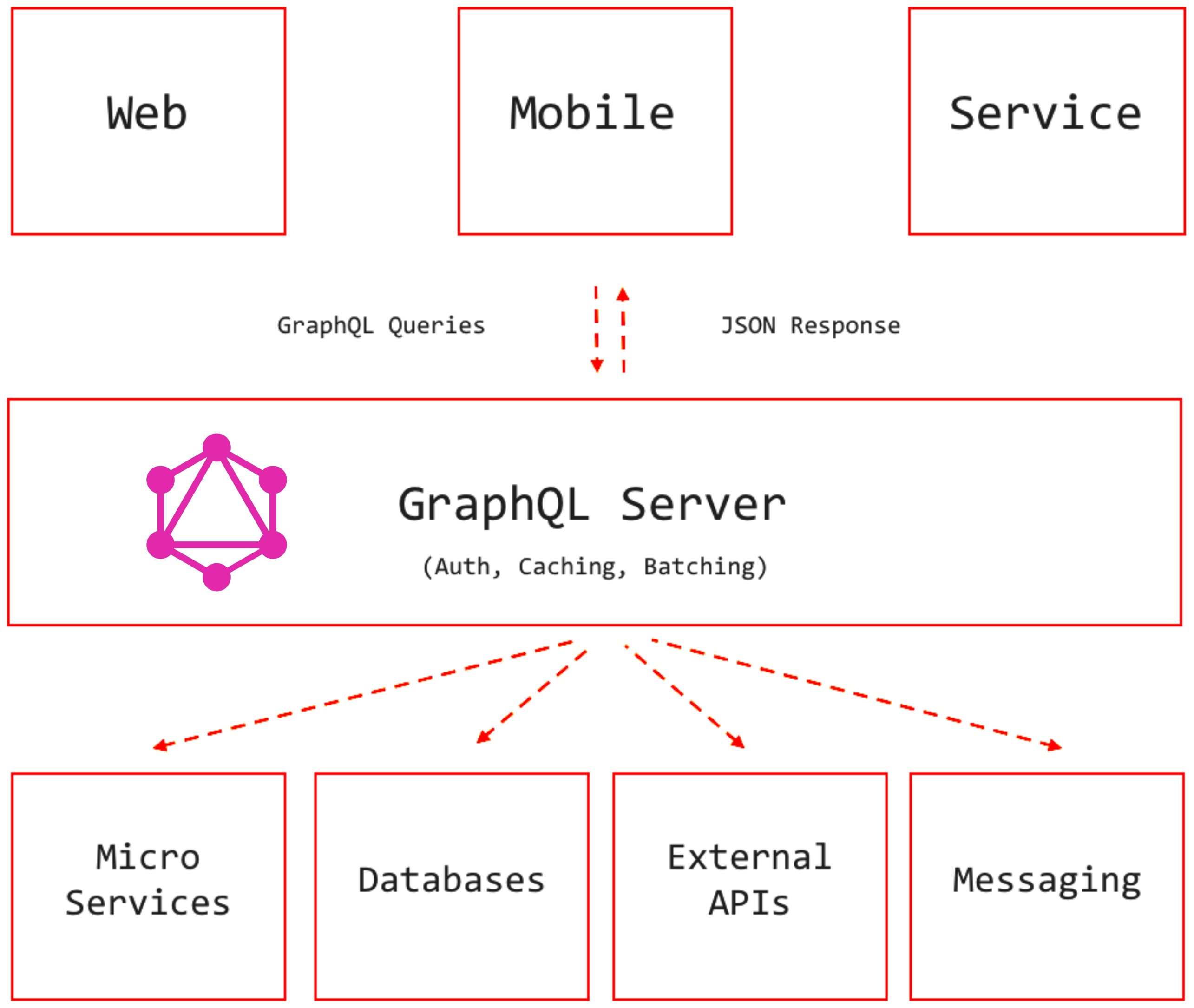


```
[
  - {
        type: "Order",
        id: 1,
        customerId: 0,
      - items: [
          - {
                type: "OrderItem",
                amount: 2,
                itemPrice: 1.33,
                productId: 1,
                product_url: "https://subkitsupport.service.subkit.io/products/1",
                price: 2.66
            },
          - {
                type: "OrderItem",
                amount: 5,
                itemPrice: 9.99,
                productId: 2,
                product_url: "https://subkitsupport.service.subkit.io/products/2",
                price: 49.95
            }
        ],
        totalPrice: 52.61,
        customer_url: "https://subkitsupport.service.subkit.io/customers/0"
    }
]
```

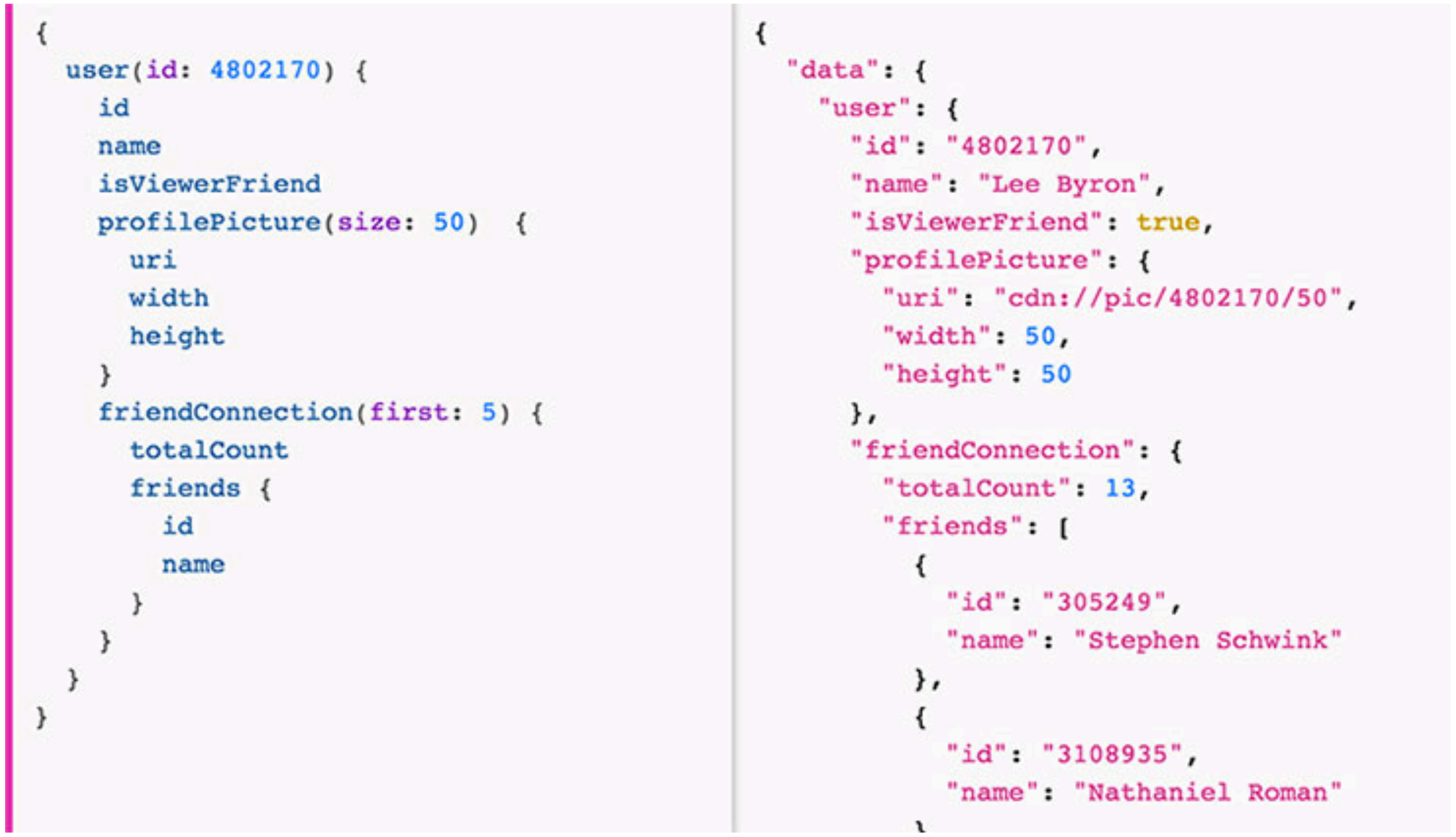
API GATEWAY (GRAPHQL)



API GATEWAY (GRAPHQL)



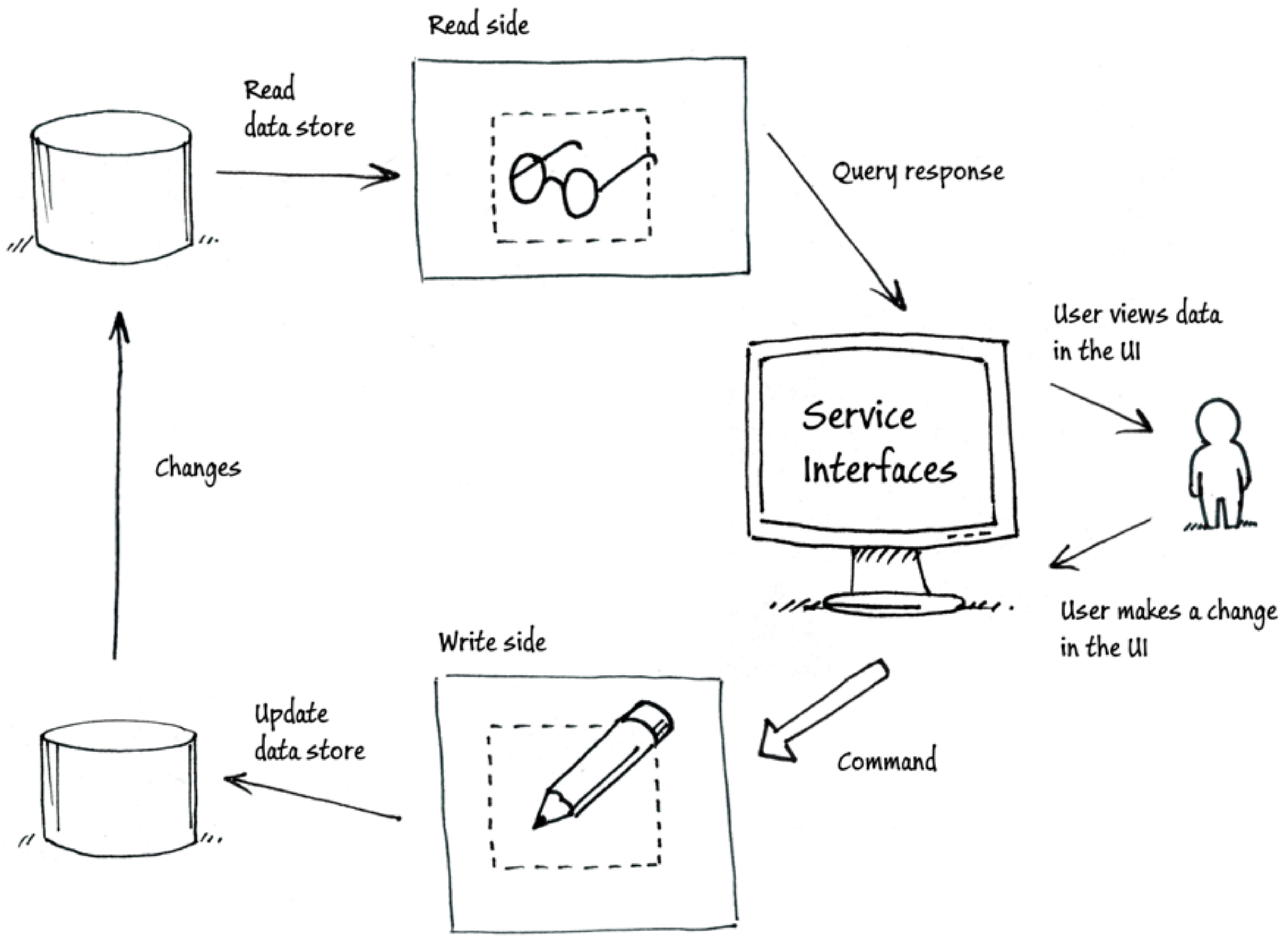
API GATEWAY (GRAPHQL)



```
{
  user(id: 4802170) {
    id
    name
    isViewerFriend
    profilePicture(size: 50)  {
      uri
      width
      height
    }
    friendConnection(first: 5) {
      totalCount
      friends {
        id
        name
      }
    }
  }
}
```

```
{
  "data": {
    "user": {
      "id": "4802170",
      "name": "Lee Byron",
      "isViewerFriend": true,
      "profilePicture": {
        "uri": "cdn://pic/4802170/50",
        "width": 50,
        "height": 50
      },
      "friendConnection": {
        "totalCount": 13,
        "friends": [
          {
            "id": "305249",
            "name": "Stephen Schwink"
          },
          {
            "id": "3108935",
            "name": "Nathaniel Roman"
```

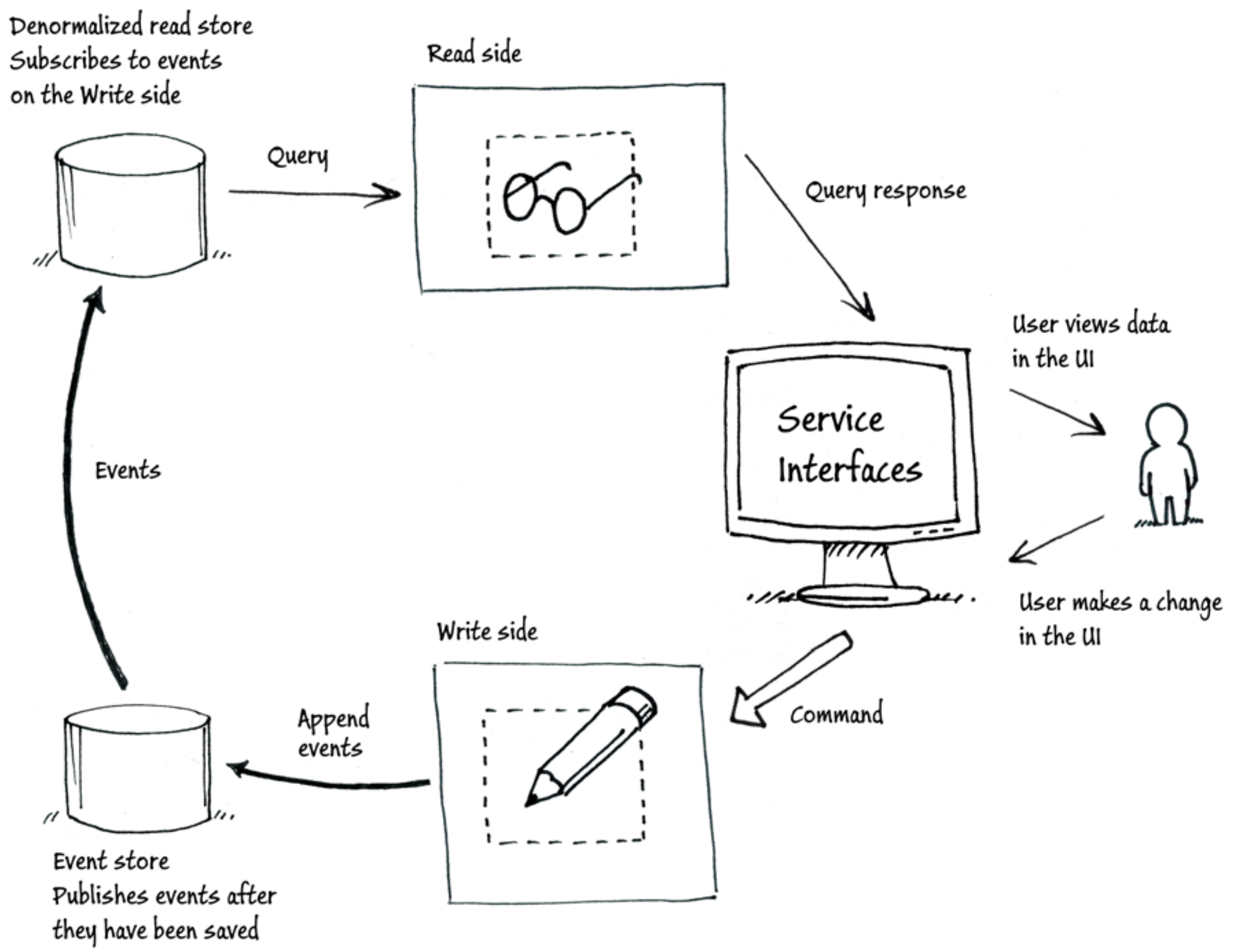
CQRS


Read side
Read data store
Query response
User views data in the UI
Service Interfaces
User makes a change in the UI
Changes
Write side
Update data store
Command

CQRS

- Different Write / Read Model
- Aggregate- / Document-Centric
- Async by Design
- Relaxed Consistency Guarantees
- Fits good to EventSourcing
- Persistency agnostic
- Optimistic Concurrency via Aggregate-Sequence-Number

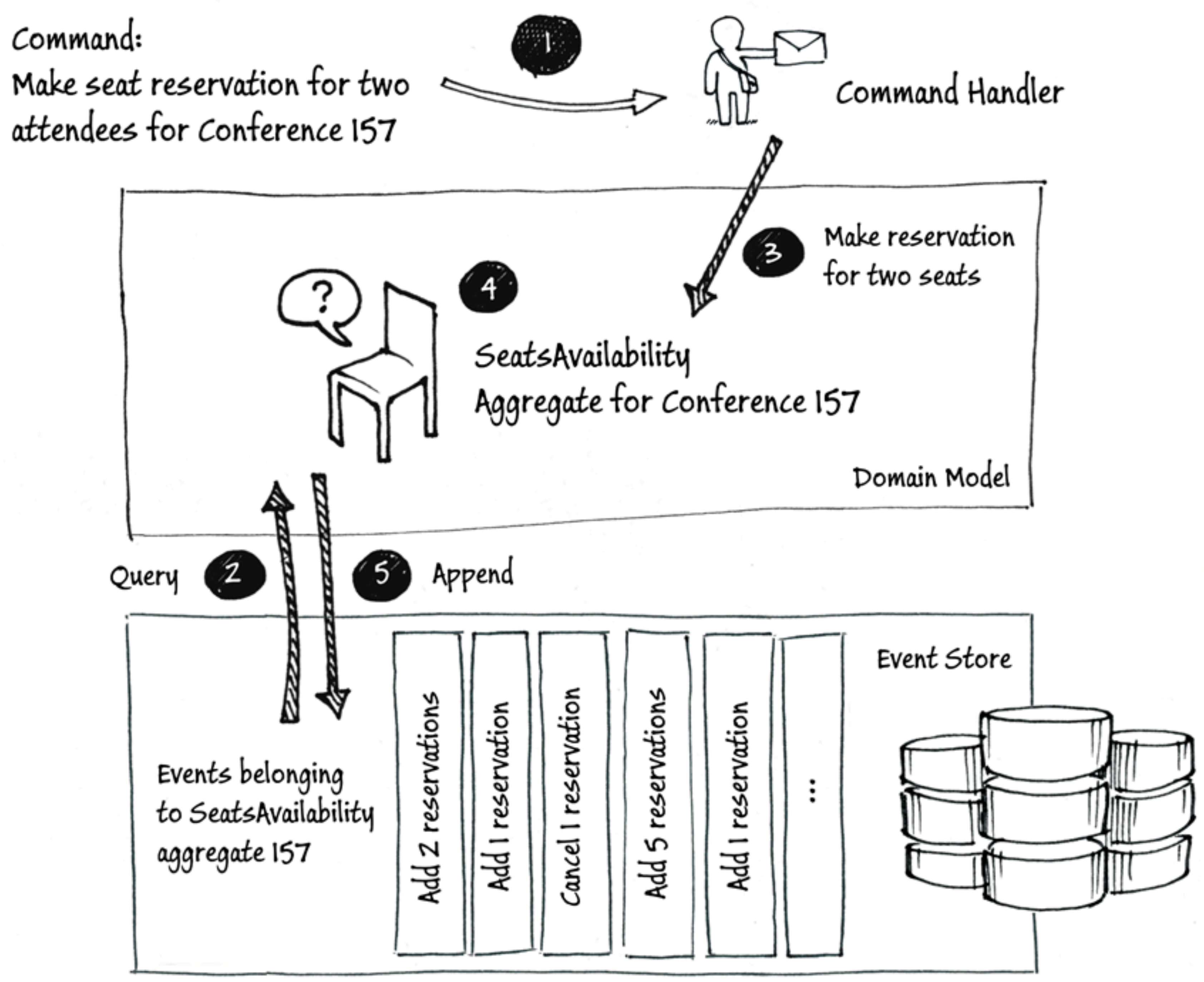
EVENT-SOURCING



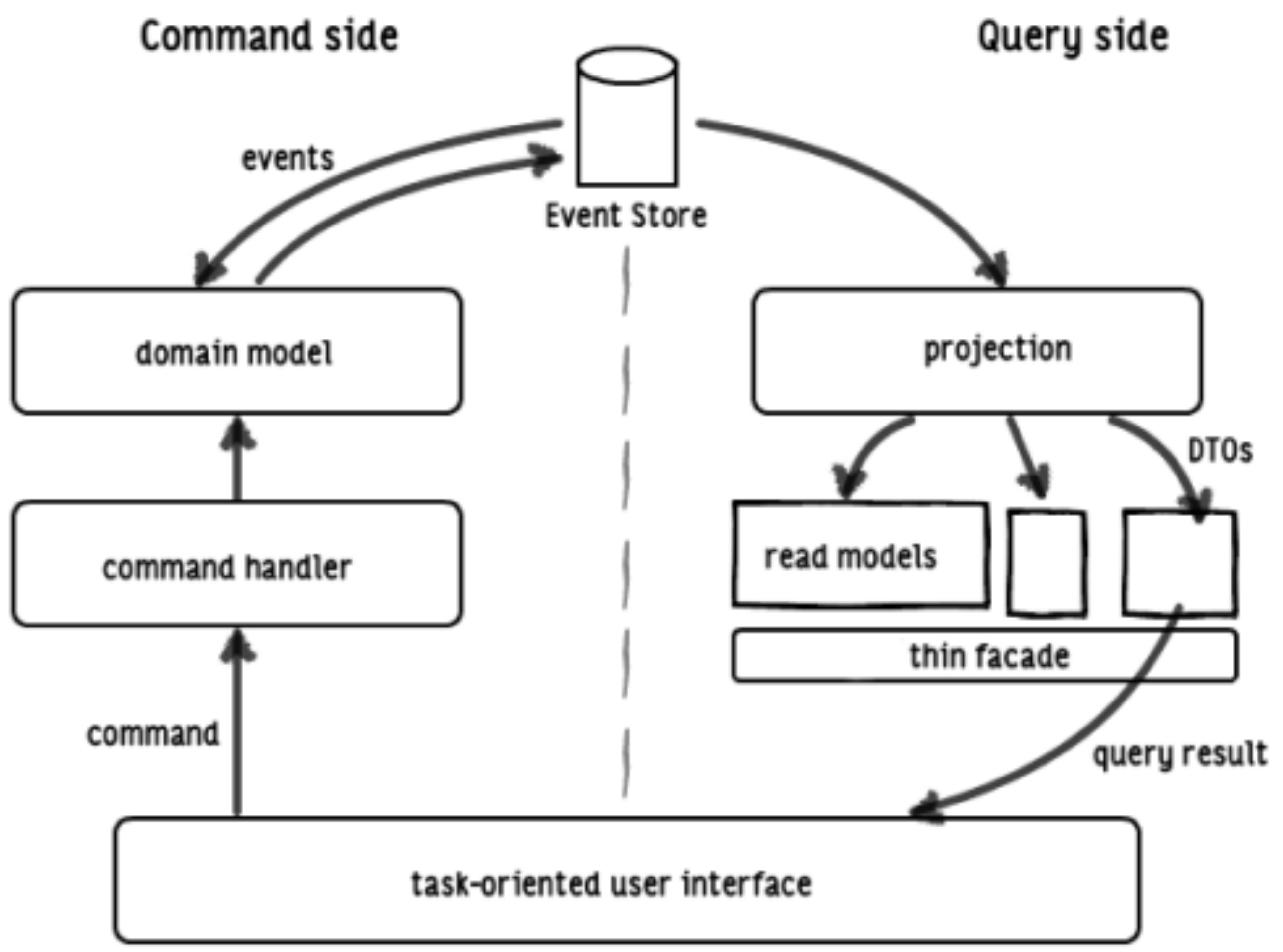
EVENT-SOURCING

- Append Only Data-Model
- Collaborative Domains
- Relaxed Consistency Guarantees
- Represents Changes over Time
- Ad-Hoc Queries / Projections via Fold-Left / Reduce / Aggregates
- Analytics over Event-Log (e.g. deduplication)
- Event-Store / Event-Log for Hydration / Dehydration

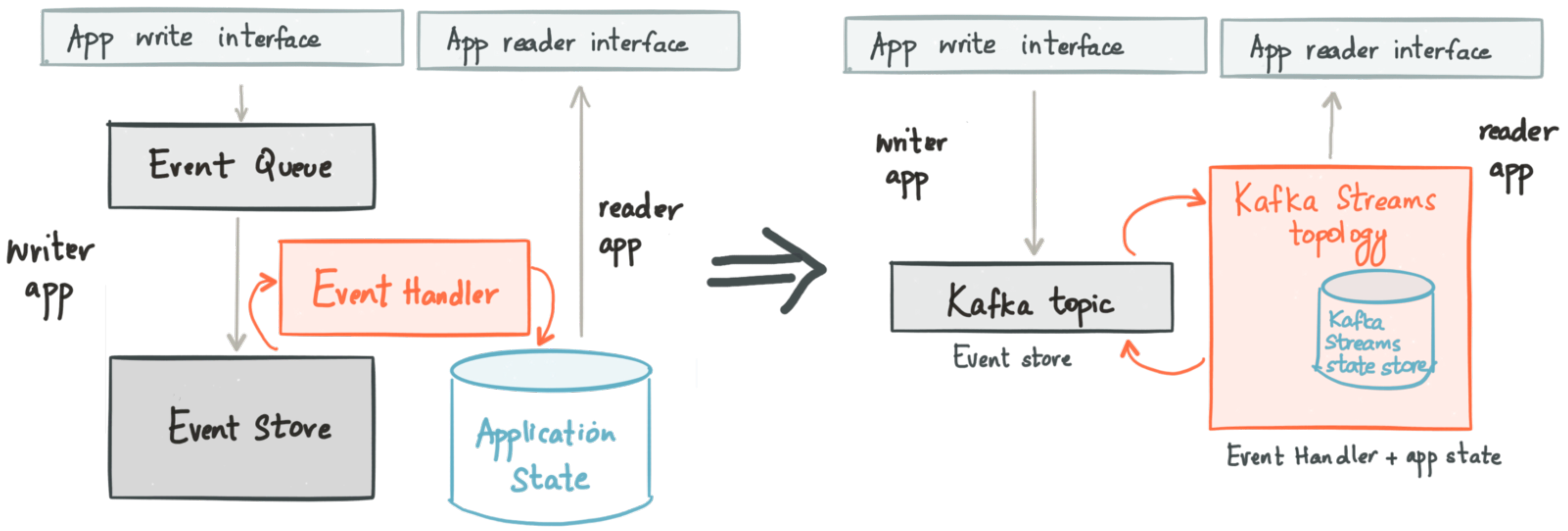
EVENT-SOURCING


Command:
Make seat reservation for two attendees for Conference 157
1
Command Handler
3
Make reservation for two seats
4
SeatsAvailability Aggregate for Conference 157
Domain Model
Query
2
5
Append
Event Store
Events belonging to SeatsAvailability aggregate 157
Add 2 reservations
Add 1 reservation
Cancel 1 reservation
Add 5 reservations
Add 1 reservation
...

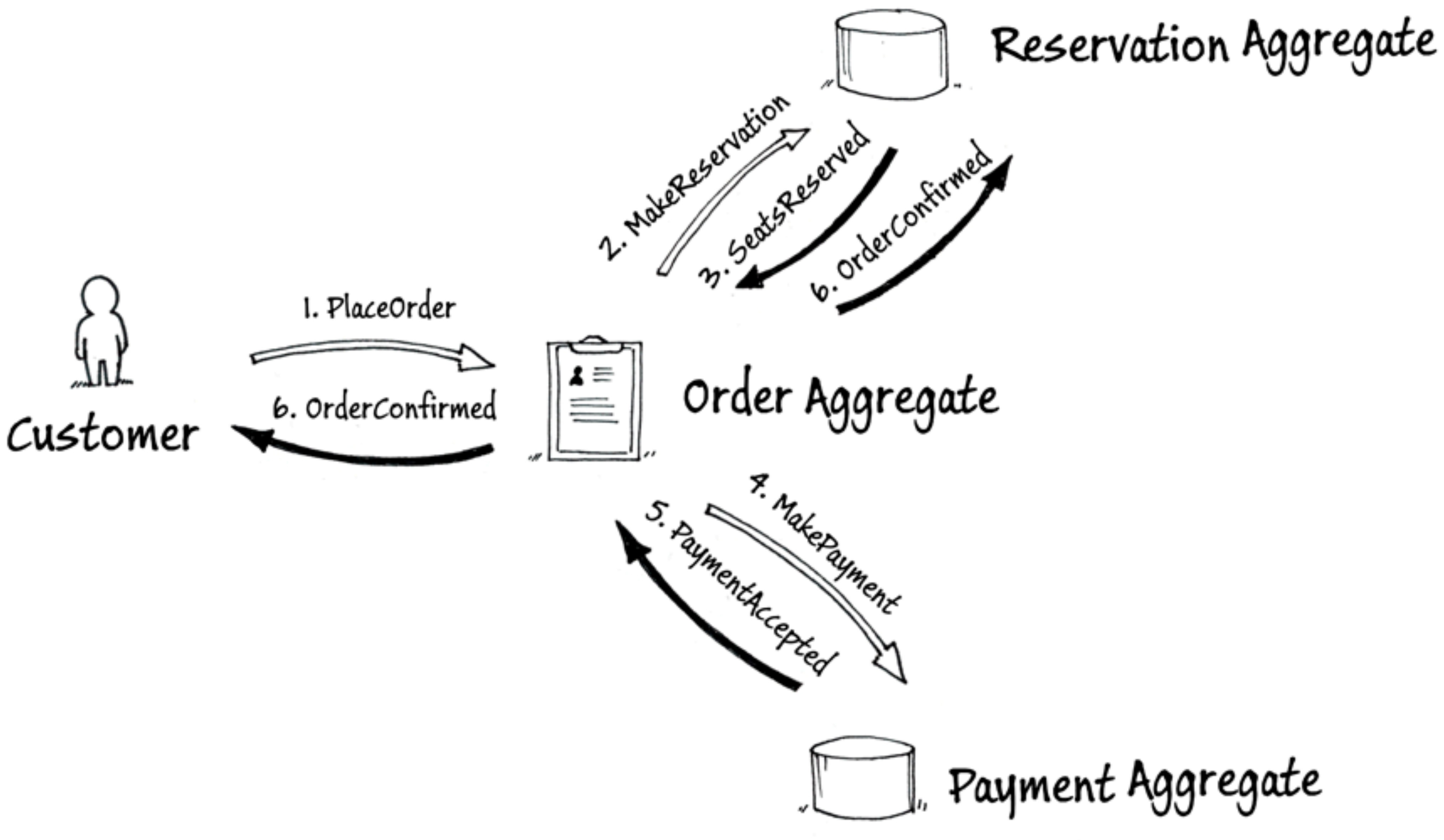
EVENT-SOURCING (E.G. EVENT-STORE)



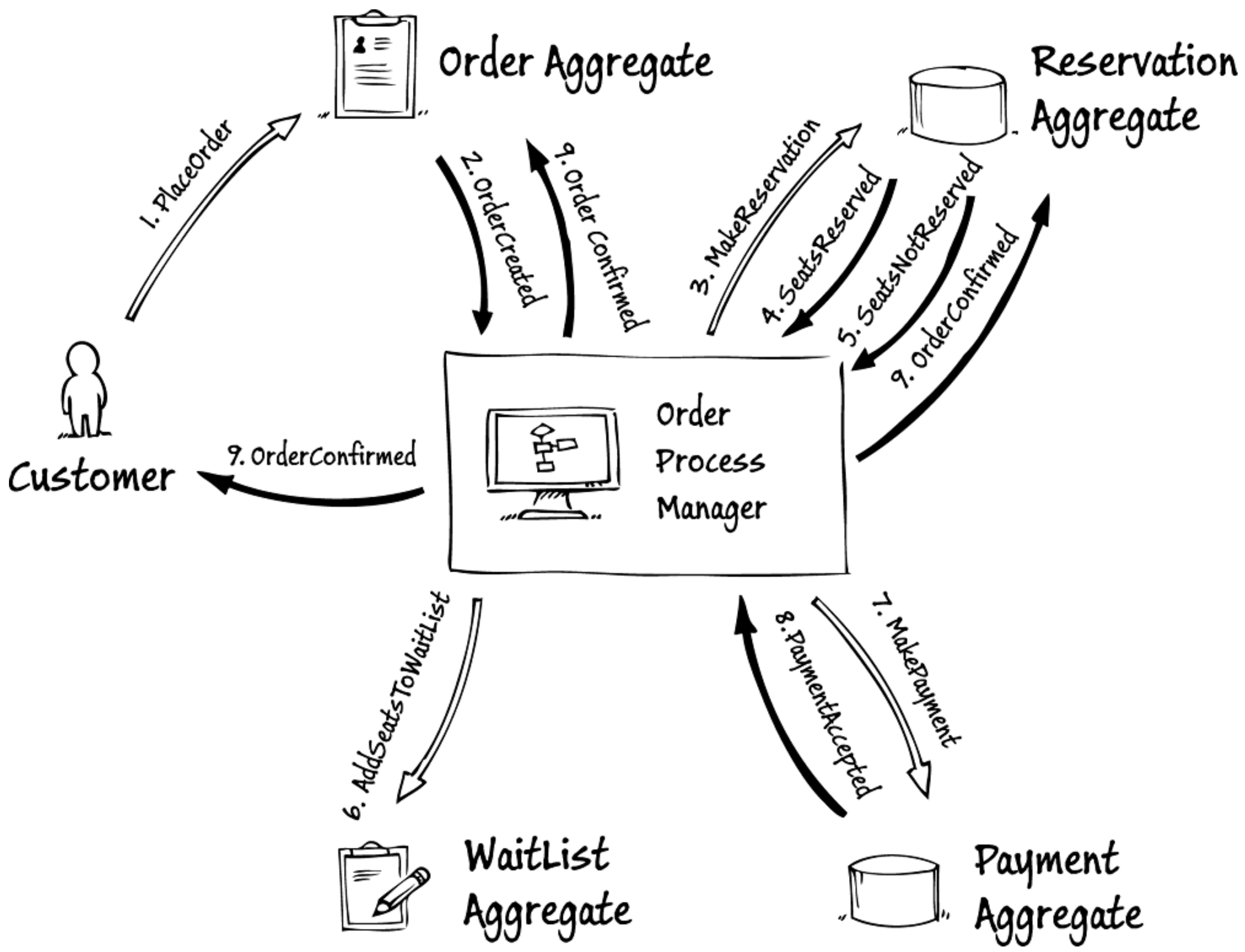
EVENT-SOURCING (E.G. KAFKA)



APPLICATION SERVICE AGGREGATE



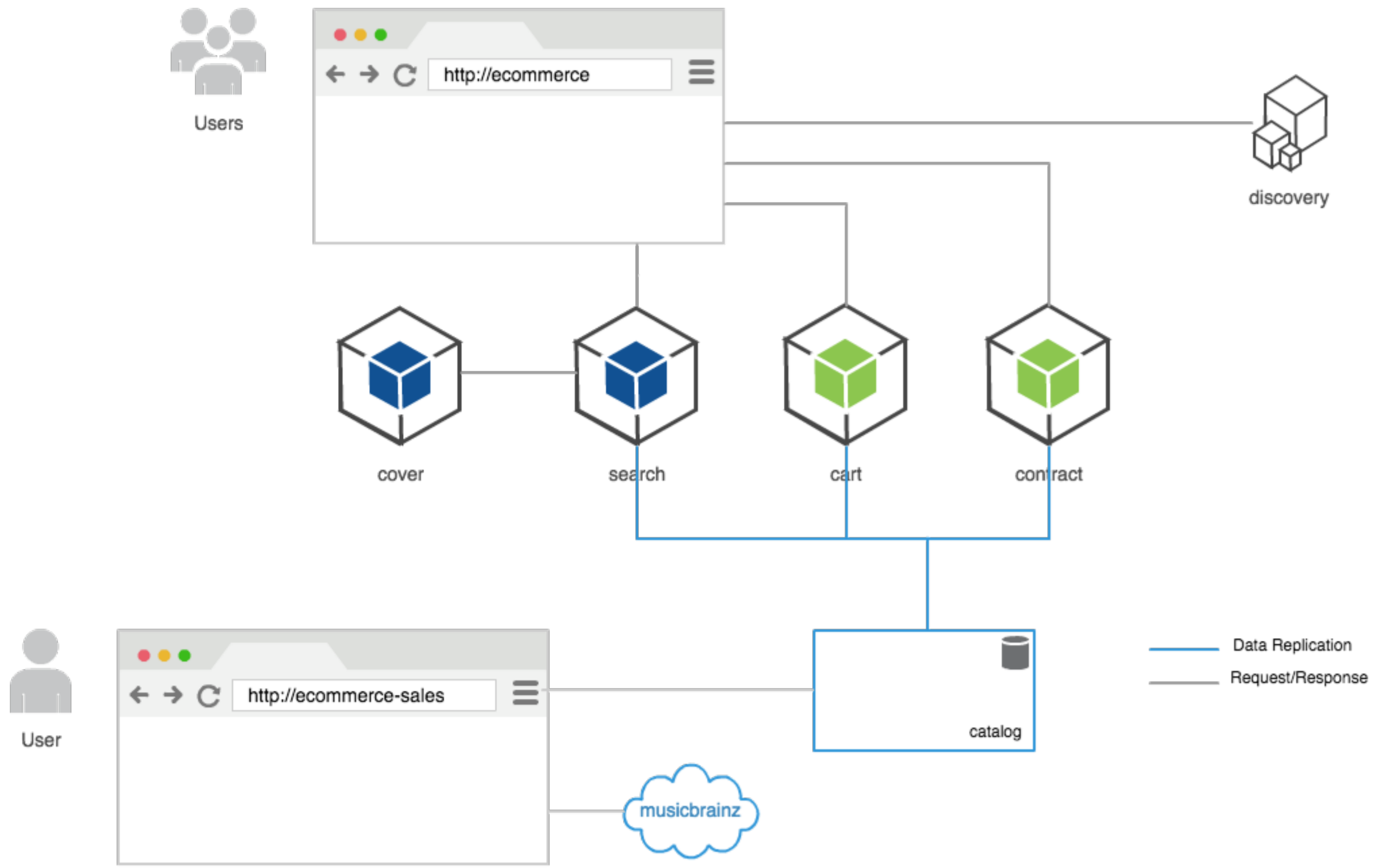
PROCESS MANAGER / SAGA



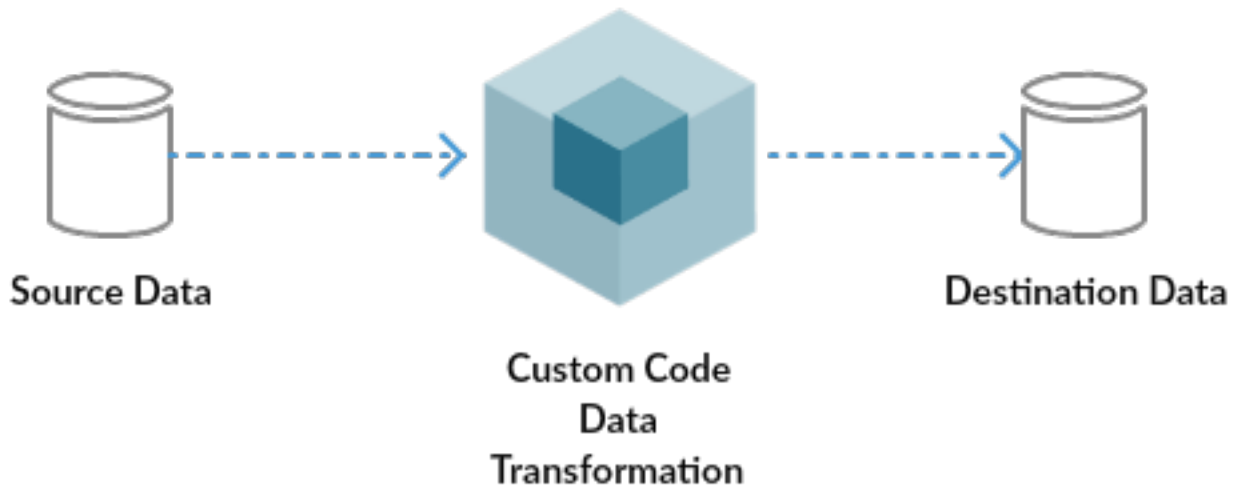
PROCESS MANAGER / SAGA

- Business Transaction Coordinator
- Explicit Transaction + Compensation
- Document- or Event-Log-Based Hydration / Dehydration
- Long-Running Processes
- Time-Based Processes
- Caution! Collaborative Domains!

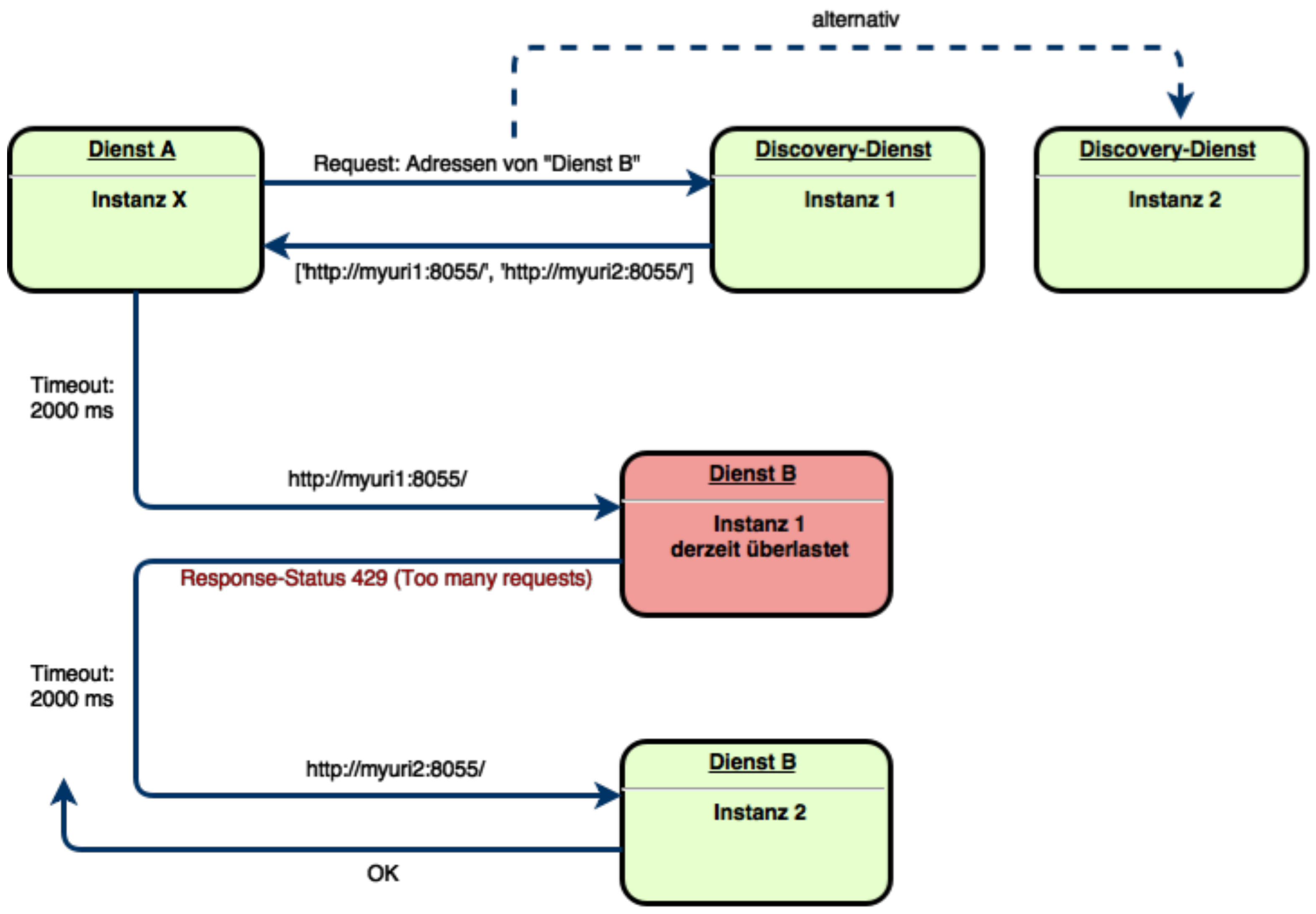
DATA REPLICATION / EXTRACT-TRANSFORM-LOAD



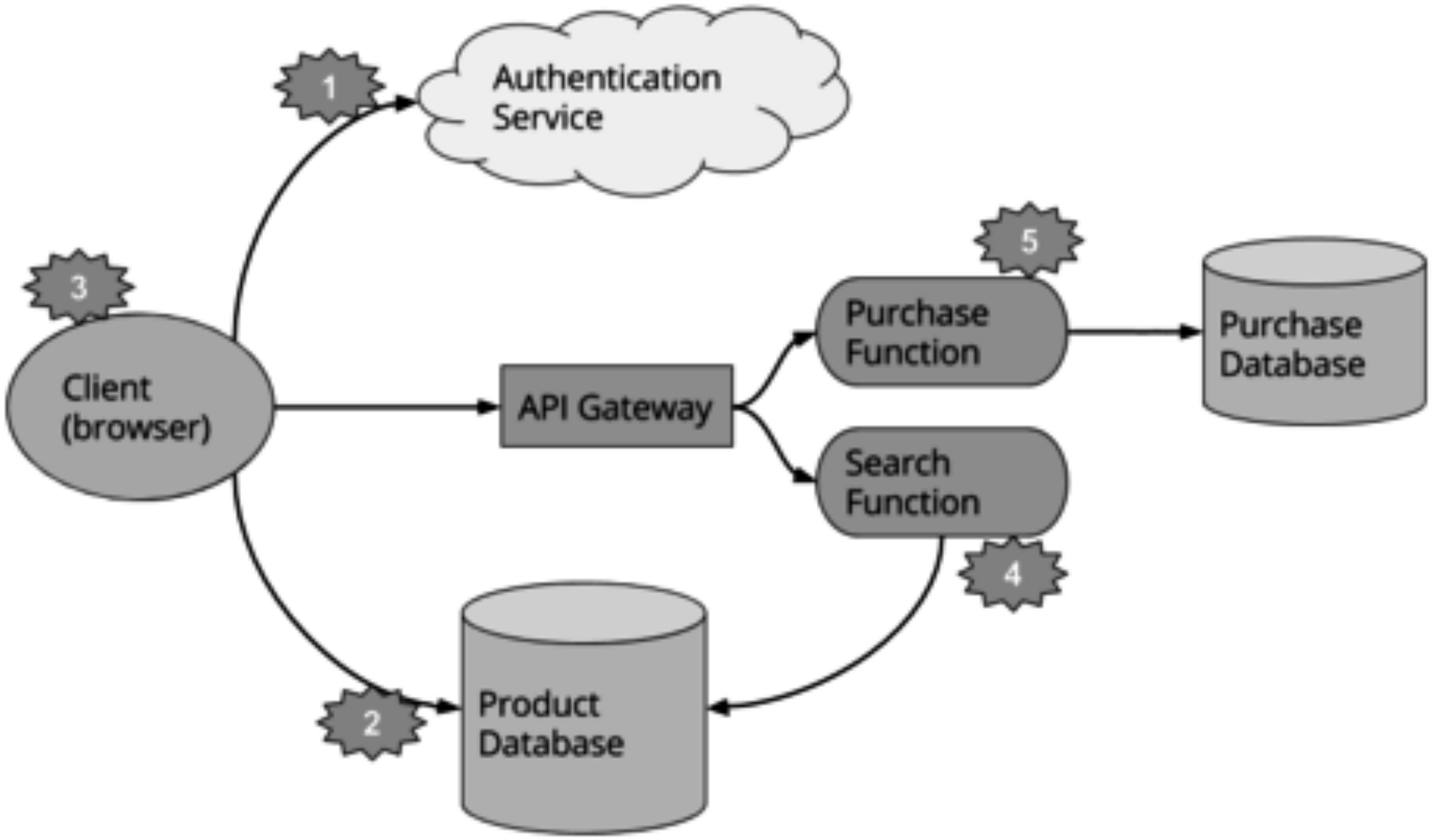
EXTRACT TRANSFORM LOAD



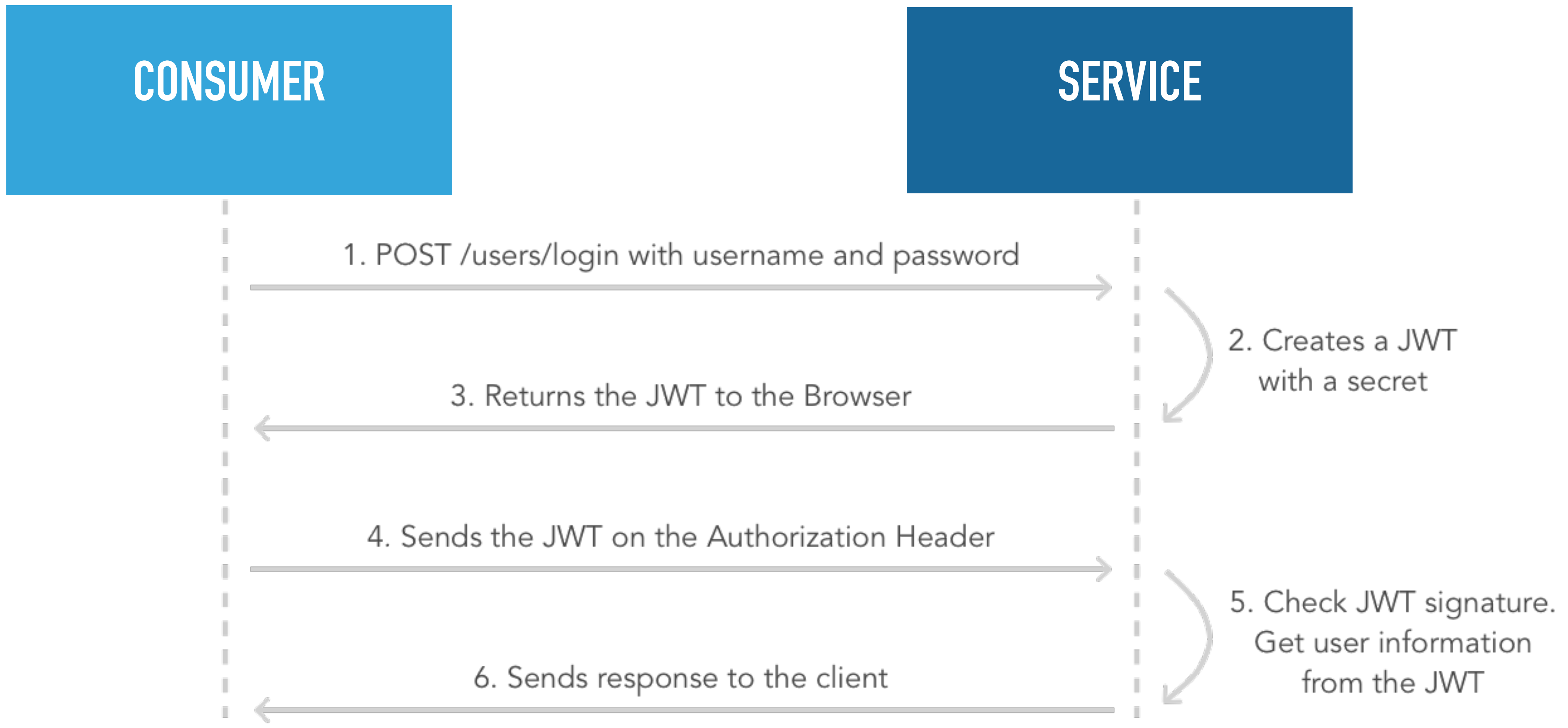
SERVICE DISCOVERY (DNS / PROXY-SERVICES)



FUNCTION AS A SERVICE



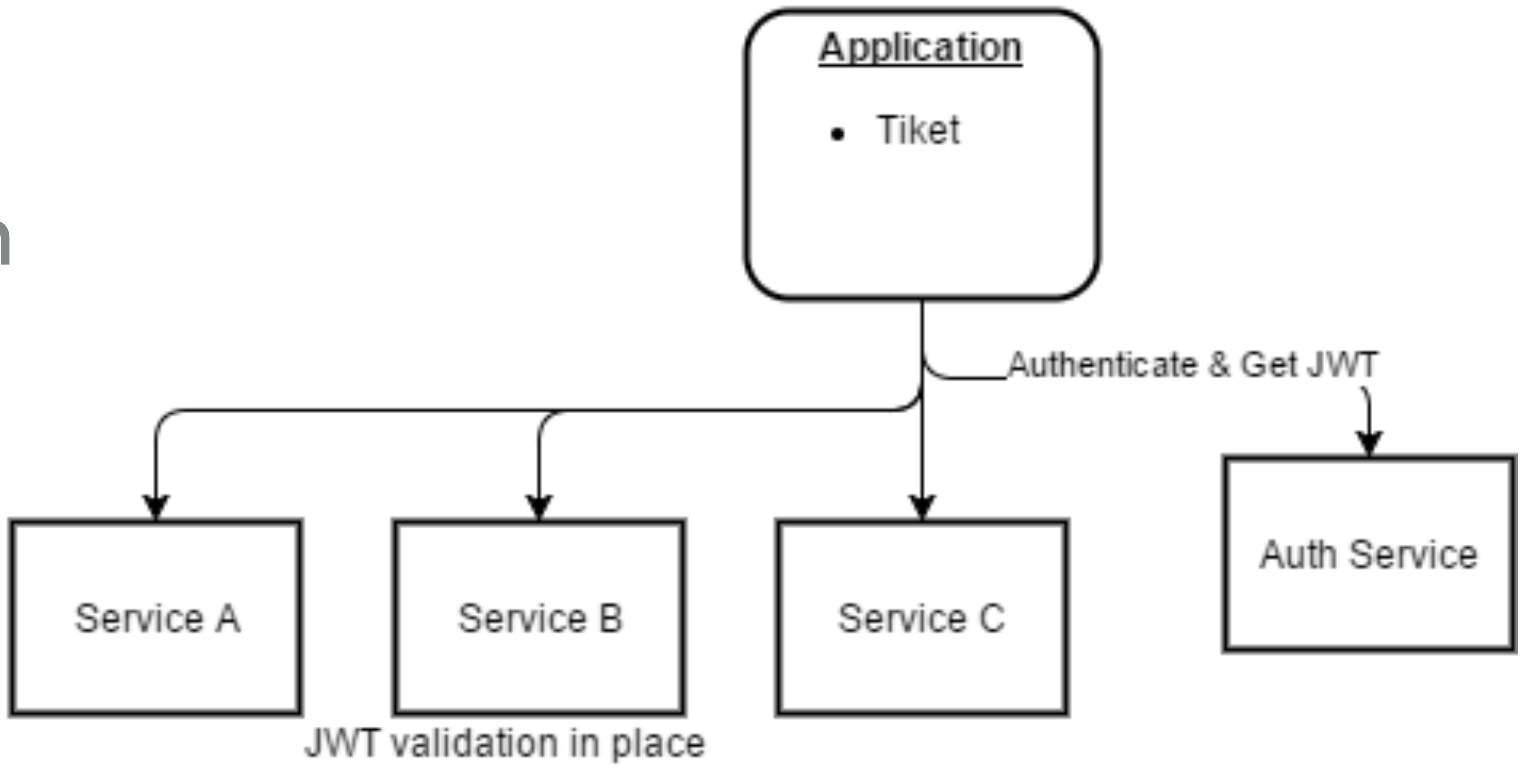
AUTH VIA JSON WEB TOKEN


CONSUMER
SERVICE
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature. Get user information from the JWT
6. Sends response to the client

Authorization-Field as Bearer-Token: `Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`
Cookie-Field: `Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`

AUTH VIA JSON WEB TOKEN

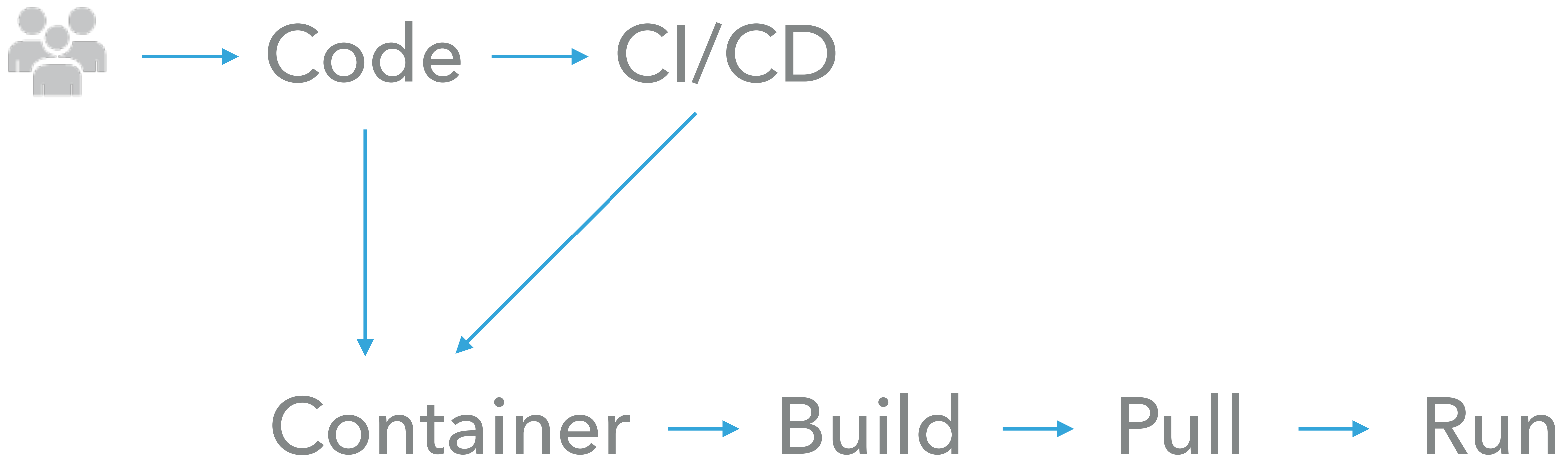
- Claim-base Access-Token
- Self-Contained
- Signature Verification
- Base64 Encoded



JOURNEY TO MICROSERVICES

OPERATIONS

CONTINUOUS DELIVERY PIPELINE

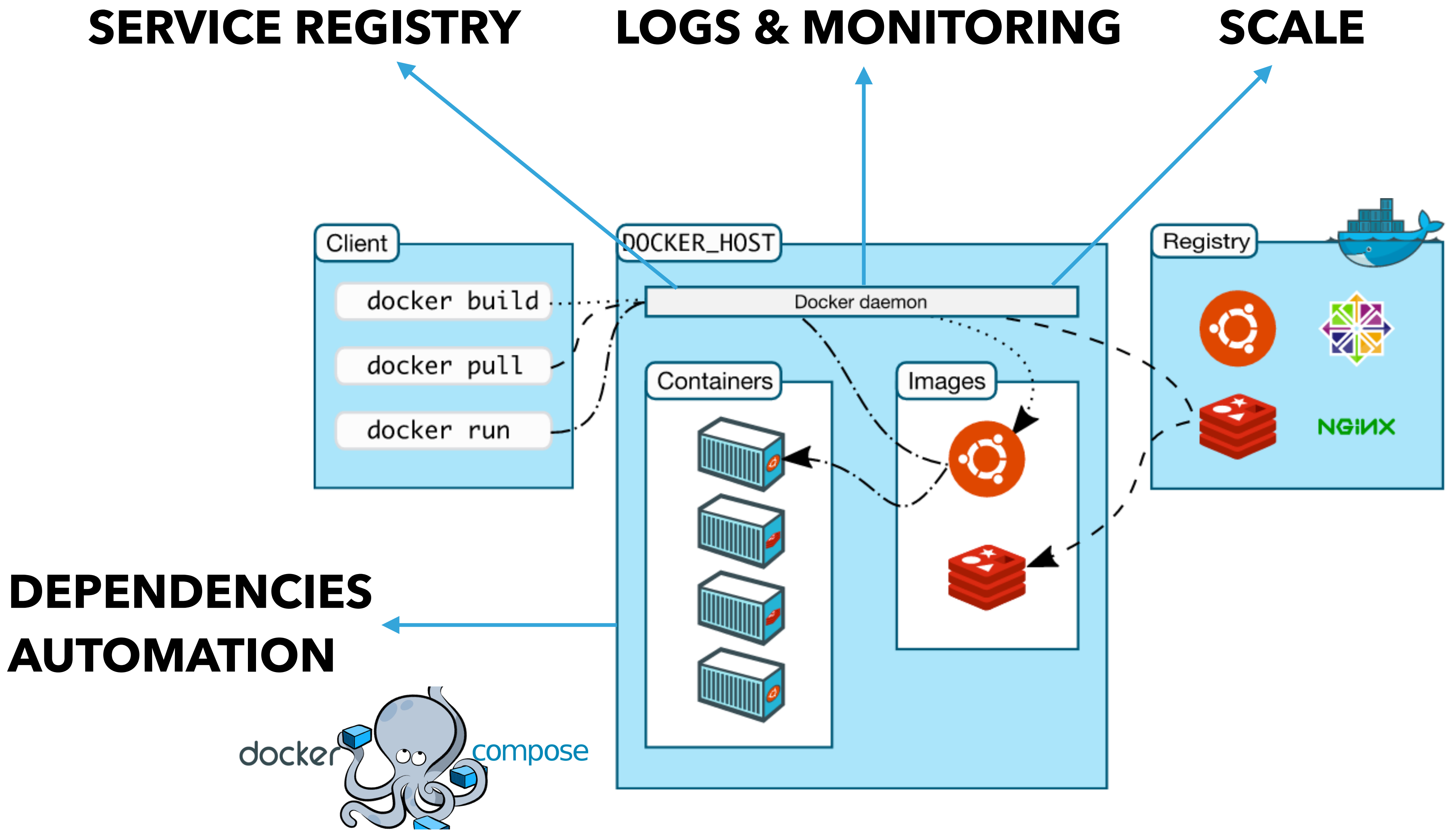


CONTAINERS

$$\begin{tikzcd}
\text{Dev} \arrow[r] & \text{Container} \arrow[r] \arrow[dr] \arrow[ddr] & \text{Ops} \\
 & & \text{Dev} \\
 & & \text{Demo}
\end{tikzcd}$$

- Shipping & Scale easily
- Explicit & Declarative dependency management
- Working for Ops automation

DOCKER LOGGING / TRACING / MONITORING



DOCKER – IMMUTABLE CONTAINER

```
FROM mhart/alpine-node

WORKDIR .
ADD . .

RUN npm install

EXPOSE 80
EXPOSE 443

cmd node app.js
```

DOCKER - DEPENDECIES / AUTOMATION

```
version: '3'
services:
  database:
    image: klaemo/couchdb:2.0.0
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]
    ports:
      - "5984:5984"
    environment:
      COUCHDB_USER: ${DROPSTACK_USER}
      COUCHDB_PASSWORD: ${DROPSTACK_SECRET}
      COUCHDB_HTTP_PORT: 5984
    volumes:
      - ./dropstack-database:/opt/couchdb/data
    extra_hosts:
      - "${DROPSTACK_MANAGER_HOST_NAME}.${DROPSTACK_DOMAIN_NAME}:${DROPSTACK_MANAGER_HOST_ADDRESS}"
  server:
    image: dropstack/server:2.7.20
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]
    depends_on:
      - database
      - registry
    ports:
      - "30000:80"
      - "53:53/udp"
      - "53:53/tcp"
    environment:
      HTTP_SERVICE_PORT: 80
      HTTPS_SERVICE_PORT: 443
      DNS_SERVICE_PORT: 53
      DOMAIN_NAME: ${DROPSTACK_DOMAIN_NAME}
      REGISTRY_URL: ${DROPSTACK_MANAGER_HOST_NAME}.${DROPSTACK_DOMAIN_NAME}:5000
      REGISTRY_USERNAME: ${DROPSTACK_USER}
      REGISTRY_PASSWORD: ${DROPSTACK_SECRET}
      SYNC_URL: http://${DROPSTACK_USER}:${DROPSTACK_SECRET}@${DROPSTACK_MANAGER_HOST_NAME}.${DROPSTACK_DOMAIN_NAME}:5984
      JWT_SECRET: ${DROPSTACK_SECRET}
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    extra_hosts:
      - "${DROPSTACK_MANAGER_HOST_NAME}.${DROPSTACK_DOMAIN_NAME}:${DROPSTACK_MANAGER_HOST_ADDRESS}"
  proxy:
    image: dropstack/proxy:2.0.3
    deploy:
      mode: "global"
    depends_on:
      - database
      - server
```

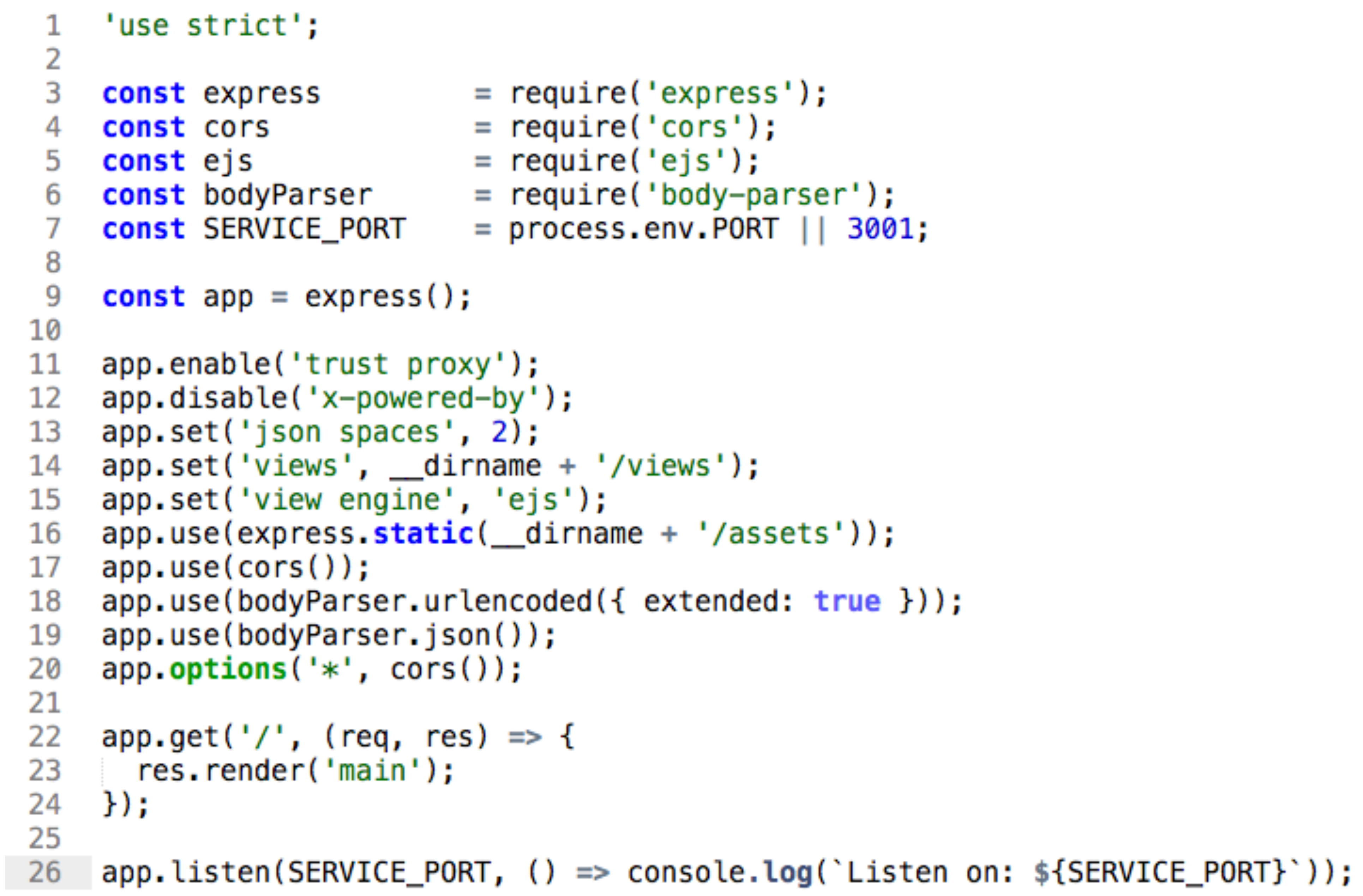
JOURNEY TO MICROSERVICES

PRACTICE

NODEJS

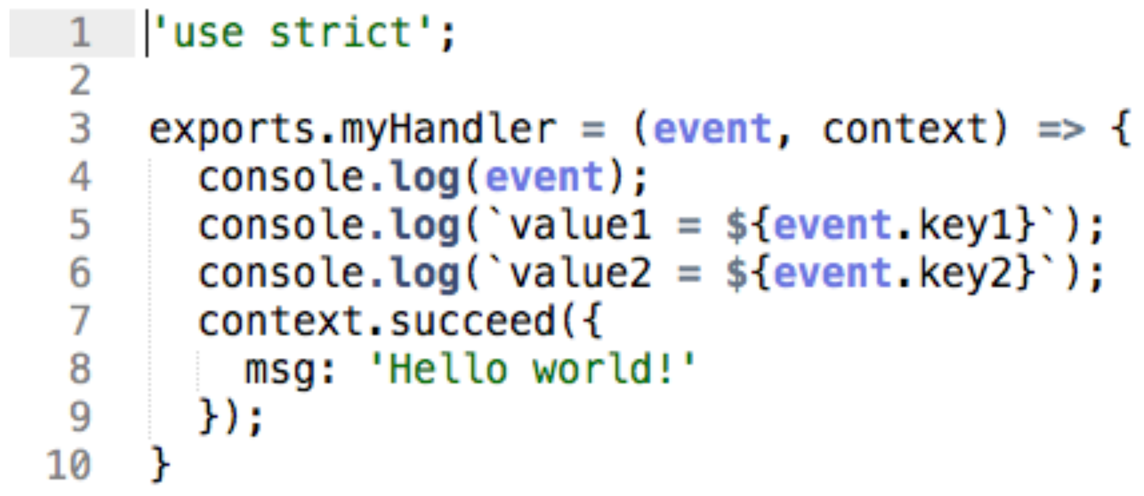
- Event-Driven I/O
- Lightweight HTTP/Web-Stack
- Full-Stack JavaScript Development
- > 230.000 NPM Packages
- Rich Build & Delivery Tools
- Extendable via child-process

HTTP API



```javascript
'use strict';

const express           = require('express');
const cors              = require('cors');
const ejs               = require('ejs');
const bodyParser        = require('body-parser');
const SERVICE_PORT      = process.env.PORT || 3001;

const app = express();

app.enable('trust proxy');
app.disable('x-powered-by');
app.set('json spaces', 2);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.static(__dirname + '/assets'));
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.options('*', cors());

app.get('/', (req, res) => {
  res.render('main');
});

app.listen(SERVICE_PORT, () => console.log(`Listen on: ${SERVICE_PORT}`));
```

FUNCTION AS A SERVICE



```
'use strict';

exports.myHandler = (event, context) => {
  console.log(event);
  console.log(`value1 = ${event.key1}`);
  console.log(`value2 = ${event.key2}`);
  context.succeed({
    msg: 'Hello world!'
  });
}
```