DESIGN SYSTÈME COMPLET – PLATEFORME AGRODEEP

Architecture Technique Détaillée & Spécifications de Déploiement

---

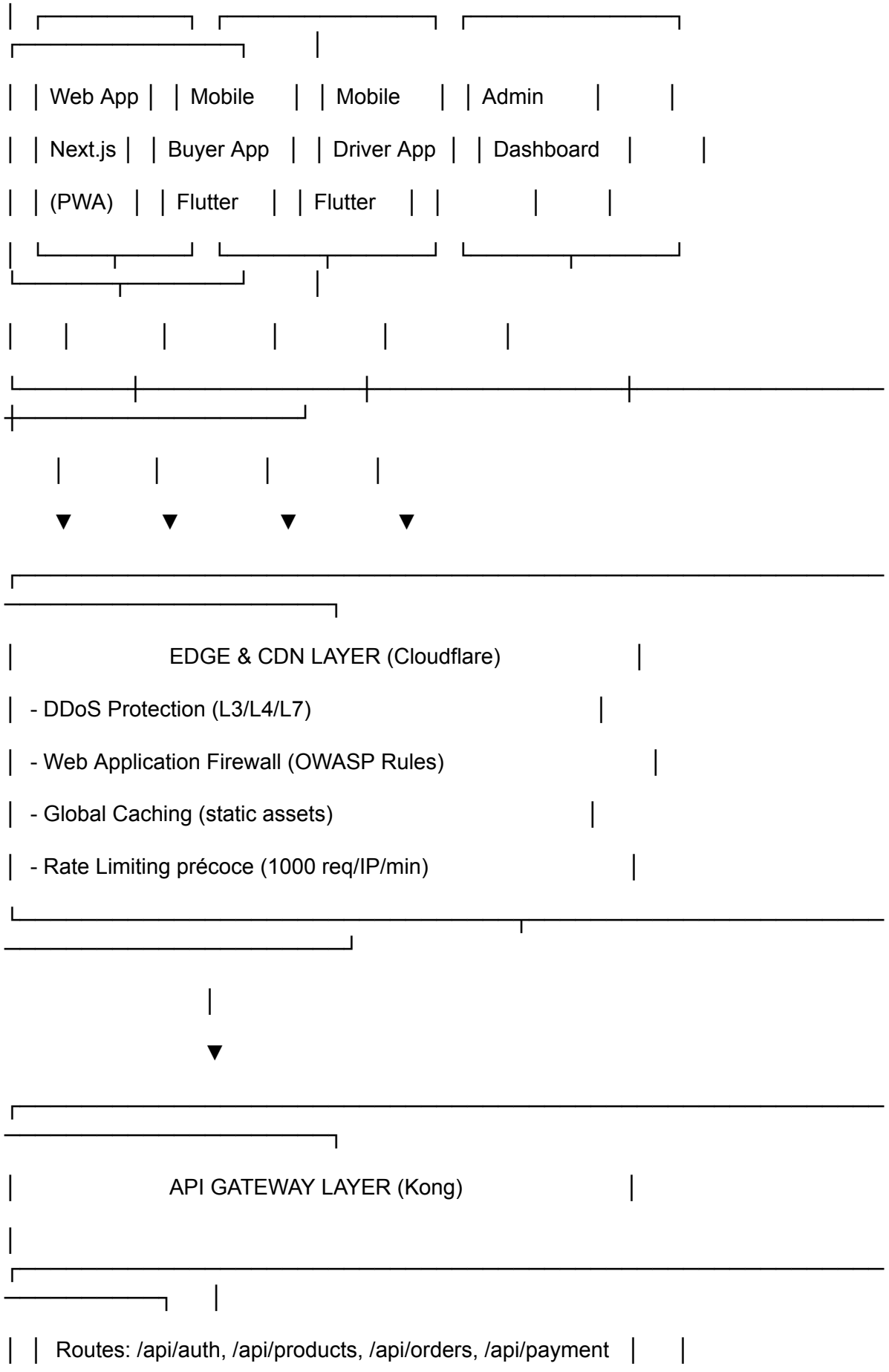
1. ARCHITECTURE GLOBALE & PRINCIPES DE CONCEPTION

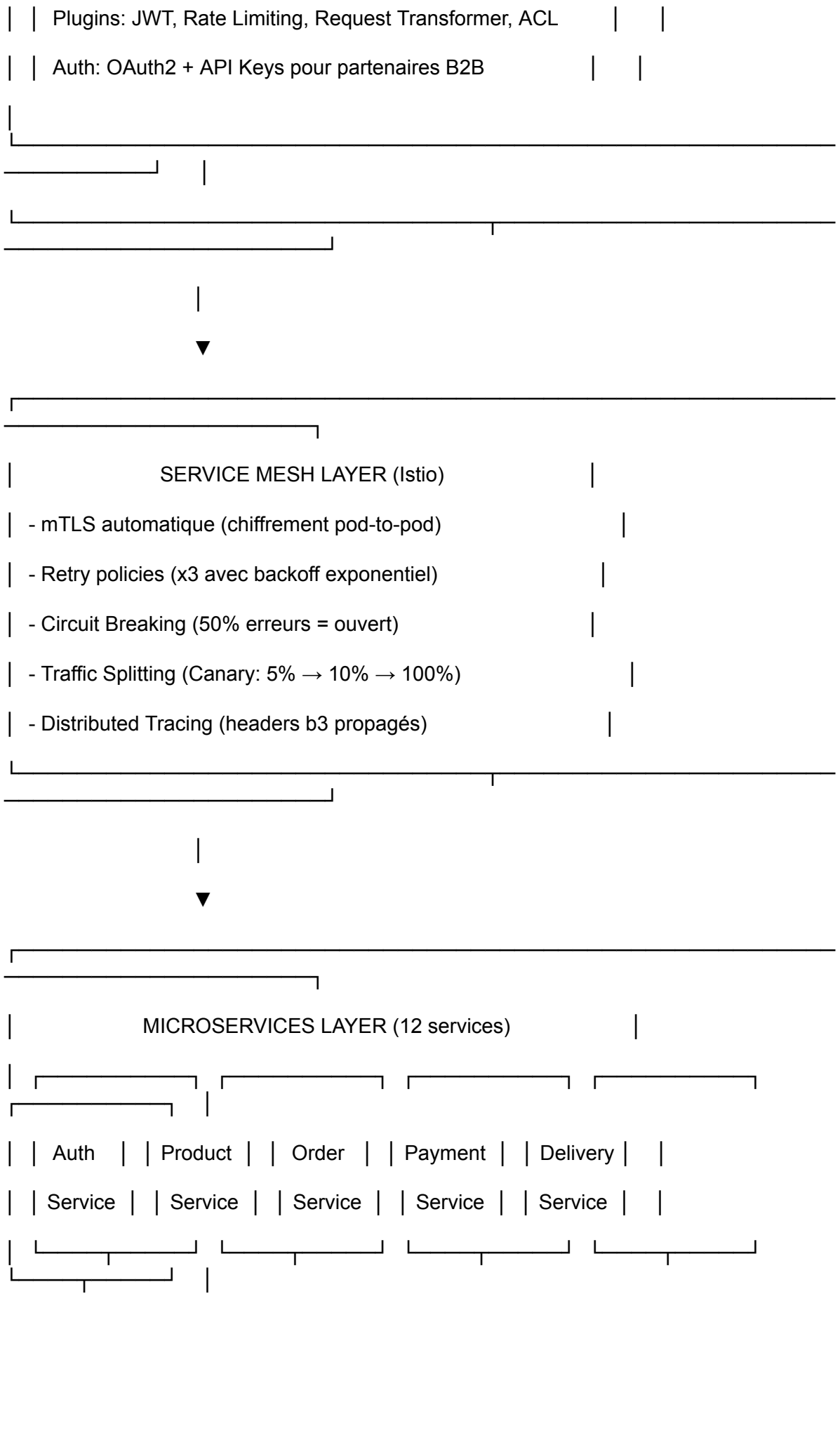
1.1 Philosophie Architecturale

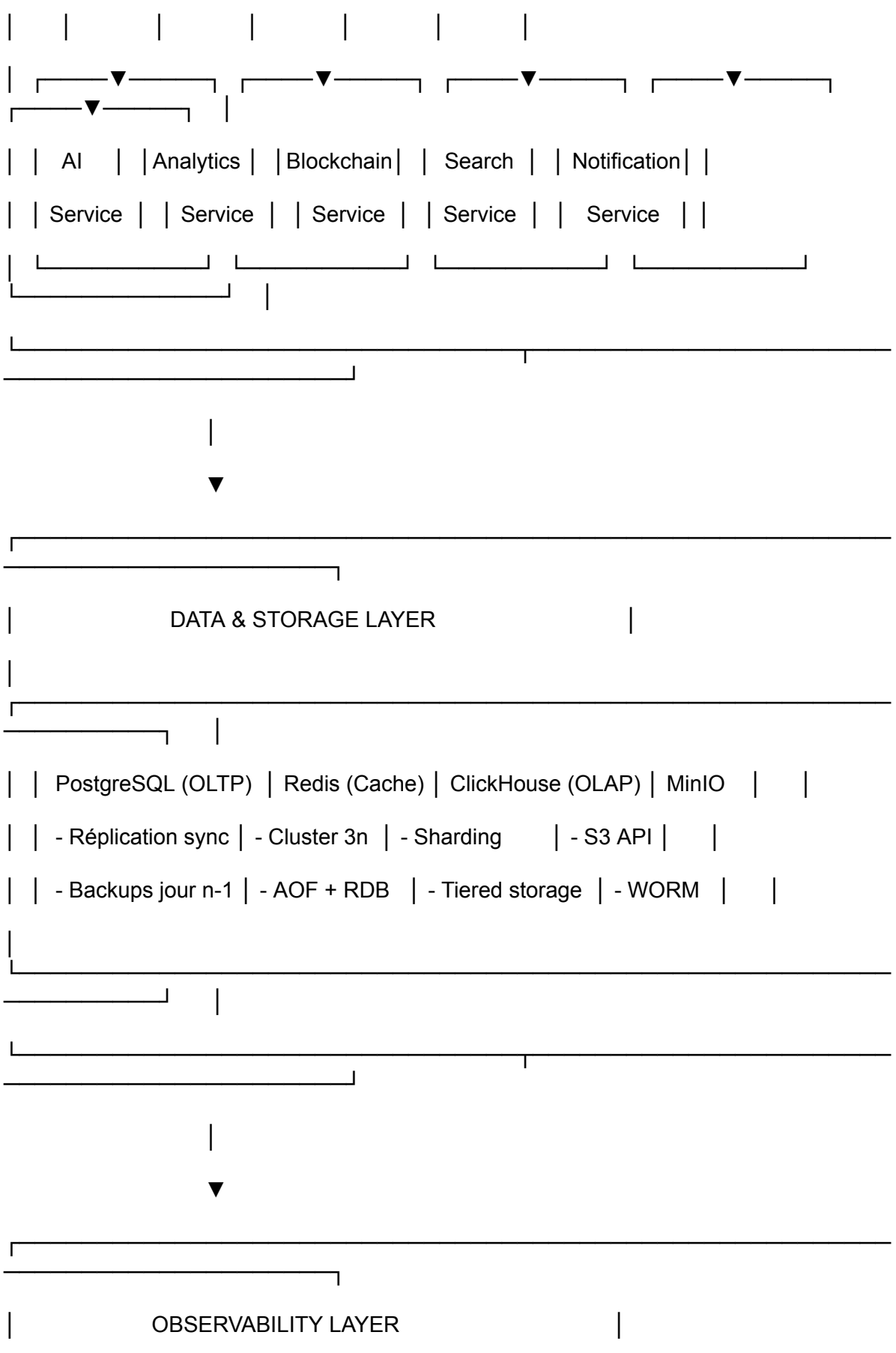
```yaml
Design Principles:
  - API-First: Toutes les communications passent par des contrats API stricts
  - Event-Driven: Décomplage des services via Kafka pour les flux critiques
  - Database per Service: Isolation des données et autonomie des équipes
  - Stateless Services: Services sans état pour scaling horizontal facile
  - Immutable Infrastructure: Pas de modification en place, seulement remplacement
  - Infrastructure as Code: 100% reproductible via Terraform & Helm
  - Observability by Design: Métriques, logs, traces intégrés dès le code
```

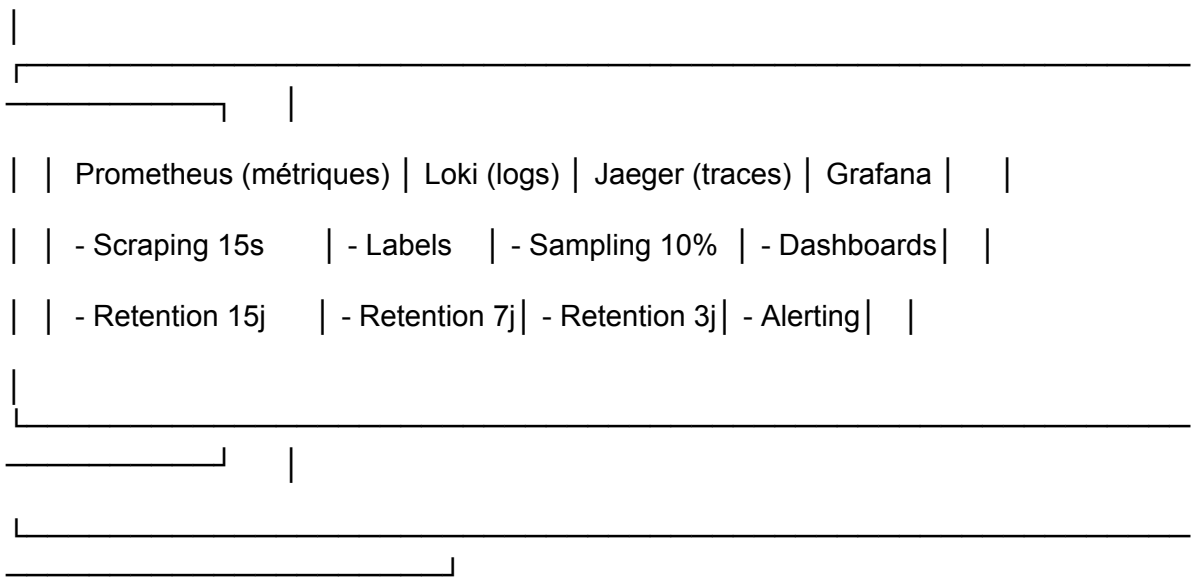
1.2 Schéma d'Architecture Niveau 0

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                CLIENT LAYER                                  │
```


Web App
Next.js
(PWA)
Mobile
Buyer App
Flutter
Mobile
Driver App
Flutter
Admin
Dashboard
EDGE & CDN LAYER (Cloudflare)
- DDoS Protection (L3/L4/L7)
- Web Application Firewall (OWASP Rules)
- Global Caching (static assets)
- Rate Limiting précoce (1000 req/IP/min)
API GATEWAY LAYER (Kong)
Routes: /api/auth, /api/products, /api/orders, /api/payment


Plugins: JWT, Rate Limiting, Request Transformer, ACL
Auth: OAuth2 + API Keys pour partenaires B2B
SERVICE MESH LAYER (Istio)
- mTLS automatique (chiffrement pod-to-pod)
- Retry policies (x3 avec backoff exponentiel)
- Circuit Breaking (50% erreurs = ouvert)
- Traffic Splitting (Canary: 5% → 10% → 100%)
- Distributed Tracing (headers b3 propagés)
MICROSERVICES LAYER (12 services)
Auth Service
Product Service
Order Service
Payment Service
Delivery Service


AI Service
Analytics Service
Blockchain Service
Search Service
Notification Service
DATA & STORAGE LAYER
PostgreSQL (OLTP)
- Réplication sync
- Backups jour n-1
Redis (Cache)
- Cluster 3n
- AOF + RDB
ClickHouse (OLAP)
- Sharding
- Tiered storage
MinIO
- S3 API
- WORM
OBSERVABILITY LAYER




```

---

## **2. COUCHE FRONTEND – DESIGN SPÉCIFIQUE**

### **2.1 Web App (Next.js 15) – Architecture des Dodles**

```typescript
// Structure des dossiers
src/
├── app/                    # App Router (Next.js 13+)
│   ├── (auth)/             # Routes d'authentification
│   │   ├── login/
│   │   │   └── page.tsx
│   │   └── register/
│   │       └── page.tsx
│   ├── (dashboard)/        # Dashboard protégé
```
```

```
│   │   ├── layout.tsx       # Sidebar + Header
│   │   ├── products/
│   │   │   └── page.tsx
│   │   └── analytics/
│   │       └── page.tsx
│   └── api/                 # API Routes Next.js (si besoin)
├── components/
│   ├── ui/                  # Composants génériques (shadcn/ui)
│   │   ├── Button.tsx
│   │   └── DataTable.tsx
│   ├── features/            # Composants métier
│   │   ├── ProductCard.tsx
│   │   ├── OrderTrackingMap.tsx
│   │   └── DeliveryRouteOptimizer.tsx
│   └── layouts/
│       └── DashboardLayout.tsx
├── lib/
│   ├── api/                 # Clients API
│   │   ├── auth.ts          # AuthService client
│   │   ├── products.ts      # ProductService client
│   │   └── websocket.ts     # WebSocket client
│   ├── hooks/               # React hooks personnalisés
│   │   ├── useAuth.ts
│   │   └── useRealtimeTracking.ts
│   └── utils/
│       ├── formatters.ts    # Format prix, dates
```

```
│       └── validators.ts    # Validation formulaires
├── store/                  # Zustand/Jotai state management
│   ├── authStore.ts
│   └── cartStore.ts
├── styles/
│   └── globals.css         # Tailwind CSS
└── types/
    └── index.ts            # Types TypeScript
```

2.1.1 Configuration next.config.js

```javascript
// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  // Performance
  poweredByHeader: false,
  compression: true,

  // Sécurité
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
```

```
          { key: 'X-Frame-Options', value: 'DENY' },
          { key: 'X-Content-Type-Options', value: 'nosniff' },
          { key: 'Referrer-Policy', value: 'strict-origin-when-cross-origin' },
          { key: 'Content-Security-Policy', value: "default-src 'self'; script-src 'self' 'unsafe-inline' *.googletagmanager.com;" }
        ]
      }
    ];
  },

  // PWA
  pwa: {
    dest: 'public',
    register: true,
    skipWaiting: true,
    disable: process.env.NODE_ENV === 'development'
  },

  // API Routes
  api: {
    responseLimit: '4mb',
  }
}

module.exports = nextConfig;
```

2.1.2 Client API Design Pattern

```typescript
// lib/api/client.ts
import axios, { AxiosInstance, AxiosError } from 'axios';

class ApiClient {
  private client: AxiosInstance;

  constructor(baseURL: string) {
    this.client = axios.create({
      baseURL,
      timeout: 10000,
      headers: {
        'Content-Type': 'application/json',
      },
    });

    // Intercepteur pour ajouter JWT
    this.client.interceptors.request.use((config) => {
      const token = localStorage.getItem('access_token');
      if (token) {
        config.headers.Authorization = `Bearer ${token}`;
      }
      return config;
    });
```

```
  // Intercepteur pour gérer refresh token
  this.client.interceptors.response.use(
    (response) => response,
    async (error: AxiosError) => {
      if (error.response?.status === 401) {
        await this.refreshToken();
        return this.client(error.config!);
      }
      return Promise.reject(error);
    }
  );
}

private async refreshToken() {
  const refreshToken = localStorage.getItem('refresh_token');
  const { data } = await this.client.post('/auth/refresh', { refresh_token: refreshToken });
  localStorage.setItem('access_token', data.access_token);
}

// Méthodes CRUD génériques
async get<T>(url: string, params?: object): Promise<T> {
  const { data } = await this.client.get<T>(url, { params });
  return data;
}
```

```
  async post<T>(url: string, body: object): Promise<T> {
    const { data } = await this.client.post<T>(url, body);
    return data;
  }

  // ... put, delete
}

// Instances par service
export const authClient = new ApiClient('/api/auth');
export const productClient = new ApiClient('/api/products');
export const orderClient = new ApiClient('/api/orders');
```

2.1.3 WebSocket Real-time Tracking

```typescript
// lib/api/websocket.ts
class WebSocketClient {
  private ws: WebSocket | null = null;
  private reconnectInterval: NodeJS.Timeout | null = null;
  private readonly url: string;

  constructor(url: string) {
    this.url = url;
  }
```

```
connect() {
  this.ws = new WebSocket(this.url);

  this.ws.onopen = () => {
    console.log('WebSocket connected');
    this.stopReconnecting();
  };

  this.ws.onmessage = (event) => {
    const data = JSON.parse(event.data);
    // Dispatch vers store
    useTrackingStore.getState().updatePosition(data);
  };

  this.ws.onclose = () => {
    console.log('WebSocket closed, reconnecting...');
    this.reconnect();
  };

  this.ws.onerror = (error) => {
    console.error('WebSocket error:', error);
  };
}

private reconnect() {
```

```
  if (this.reconnectInterval) return;

  this.reconnectInterval = setInterval(() => {
    this.connect();
  }, 5000); // Retry toutes les 5s
}

private stopReconnecting() {
  if (this.reconnectInterval) {
    clearInterval(this.reconnectInterval);
    this.reconnectInterval = null;
  }
}

subscribe(orderId: string) {
  this.ws?.send(JSON.stringify({
    type: 'SUBSCRIBE',
    orderId
  }));
}

unsubscribe(orderId: string) {
  this.ws?.send(JSON.stringify({
    type: 'UNSUBSCRIBE',
    orderId
  }));
```

```
  }
}

export const trackingWs = new WebSocketClient('wss://api.agrodeep.com/ws/tracking');
```

2.2 Mobile Apps (Flutter) – Architecture des Composants

```dart
// Structure projet
lib/
├── core/
│   ├── config/              # Configuration (dev/staging/prod)
│   │   └── app_config.dart
│   ├── network/             # Client API & WebSocket
│   │   ├── api_client.dart
│   │   ├── auth_interceptor.dart
│   │   └── websocket_service.dart
│   ├── storage/             # Local storage (SQLite + SharedPrefs)
│   │   ├── database.dart
│   │   └── secure_storage.dart
│   └── utils/
│       ├── validators.dart
│       └── formatters.dart
```

```
├── features/
│   ├── auth/
│   │   ├── presentation/
│   │   │   ├── pages/login_page.dart
│   │   │   └── widgets/login_form.dart
│   │   ├── domain/
│   │   │   ├── entities/user.dart
│   │   │   └── repositories/auth_repository.dart
│   │   └── data/
│   │       ├── datasources/auth_api.dart
│   │       └── models/user_model.dart
│   ├── products/
│   │   └── ... (même structure)
│   └── orders/
│       └── ... (même structure)
├── shared/
│   ├── widgets/            # Composants réutilisables
│   │   ├── loading_indicator.dart
│   │   └── error_dialog.dart
│   └── theme/              # Design system
│       ├── colors.dart
│       └── text_styles.dart
└── main.dart               # Point d'entrée
```

2.2.1 Configuration Environnements

```dart
// core/config/app_config.dart
enum Environment { dev, staging, production }

class AppConfig {
  static final AppConfig _instance = AppConfig._internal();
  factory AppConfig() => _instance;

  late Environment environment;
  late String apiBaseUrl;
  late String websocketUrl;
  late bool enableLogging;

  void init(Environment env) {
    environment = env;
    switch (env) {
      case Environment.dev:
        apiBaseUrl = 'http://10.0.2.2:8000'; // Android emulator localhost
        websocketUrl = 'ws://10.0.2.2:8000/ws';
        enableLogging = true;
        break;
      case Environment.staging:
        apiBaseUrl = 'https://staging-api.agrodeep.com';
        websocketUrl = 'wss://staging-api.agrodeep.com/ws';
        enableLogging = true;
```

```
        break;
      case Environment.production:
        apiBaseUrl = 'https://api.agrodeep.com';
        websocketUrl = 'wss://api.agrodeep.com/ws';
        enableLogging = false;
        break;
    }
  }
}
```

2.2.2 Offline-First Architecture – Driver App

```dart
// core/storage/database.dart
// Floor (SQLite ORM pour Flutter)
@Database(version: 1, entities: [Delivery, TrackingPoint])
abstract class AppDatabase extends FloorDatabase {
  DeliveryDao get deliveryDao;
  TrackingPointDao get trackingPointDao;
}

// data/model/delivery_model.dart
@Entity(tableName: 'deliveries')
class Delivery {
  @PrimaryKey()
```

```
  final String id;

  final String orderId;
  final String driverId;
  final String status; // pending, in_progress, completed

  @ColumnInfo(name: 'delivery_address')
  final String deliveryAddress;

  final double lat;
  final double lng;

  final String? signaturePath; // Photo signature locale
  final int? deliveredAt;

  Delivery({
    required this.id,
    // ... autres champs
  });
}

// Domain entity
class DeliveryEntity {
  final String id;
  final DeliveryStatus status; // Enum typé
  final GeoPoint coordinates;
```

```
  DeliveryEntity.fromModel(Delivery model) : /* conversion */ ;
}

// Repository pattern
class DeliveryRepository {
  final DeliveryApi api;
  final DeliveryDao dao;

  Future<List<Delivery>> getTodaysDeliveries() async {
    try {
      // 1. Essayer API
      final deliveries = await api.getTodaysDeliveries();
      // 2. Cacher localement
      await dao.insertAll(deliveries);
      return deliveries;
    } catch (e) {
      // 3. Fallback SQLite
      return await dao.getAllDeliveries();
    }
  }

  Future<void> markAsDelivered(String deliveryId, String signaturePath) async {
    // 1. Mettre à jour local
    await dao.updateStatus(deliveryId, 'completed', signaturePath);
```

```
    // 2. Tenter sync
    try {
      await api.markAsDelivered(deliveryId, signaturePath);
    } catch (e) {
      // 3. Marquer pour sync ultérieur
      await dao.markPendingSync(deliveryId);
    }
  }
}
```

2.2.3 Background Location Tracking – Android/iOS

```dart
// lib/core/services/location_service.dart
import 'package:background_locator_2/background_locator.dart';

class LocationService {
  static void callback(LocationDto location) async {
    // Appelé toutes les 10 secondes (configurable)
    final trackingPoint = TrackingPoint(
      lat: location.latitude,
      lng: location.longitude,
      timestamp: DateTime.now().millisecondsSinceEpoch,
      accuracy: location.accuracy,
    );
```

```
  // 1. Sauvegarde locale
  final db = await AppDatabase.instance.database;
  await db.trackingPointDao.insert(trackingPoint);

  // 2. Tentative envoi si connecté
  final connectivity = Connectivity().checkConnectivity();
  if (connectivity == ConnectivityResult.wifi ||
      connectivity == ConnectivityResult.mobile) {
    try {
      await api.sendTrackingPoints([trackingPoint]);
      // 3. Supprime si envoyé avec succès
      await db.trackingPointDao.delete(trackingPoint.id);
    } catch (e) {
      // Laisse en local pour retry ultérieur
    }
  }
}

static void startTracking() {
  BackgroundLocator.registerLocationUpdate(
    callback,
    settings: LocationSettings(
      notificationTitle: "AgroDeep en cours d'utilisation",
      notificationMsg: "Transmission position en temps réel",
      interval: 10, // secondes
```

```
        distanceFilter: 50, // mètres
        accuracy: LocationAccuracy.NAVIGATION,
      ),
    );
  }
}
```

---

3. COUCHE BACKEND – DESIGN DES MICROSERVICES

3.1 Service Mesh (Istio) – Configuration Détaillée

```yaml
# istio/ingress-gateway.yaml
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: agrodeep-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
```

```
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "api.agrodeep.com"
    - "staging-api.agrodeep.com"
    # Redirection HTTPS forcée
    tls:
      httpsRedirect: true
  - port:
      number: 443
      name: https
      protocol: HTTPS
    hosts:
    - "api.agrodeep.com"
    tls:
      mode: SIMPLE
      credentialName: agrodeep-tls-cert

---
# istio/virtual-service.yaml
apiVersion: networking.istio网状_structure.istio.io/v1beta1
kind: VirtualService
metadata:
  name: api-routes
  namespace: production
```

```
spec:
  hosts:
  - "api.agrodeep.com"
  gateways:
  - istio-system/agrodeep-gateway
  http:
  - match:
    - uri:
        prefix: /api/auth
    route:
    - destination:
        host: auth-service
        port:
          number: 80
    timeout: 5s
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: "5xx,reset,connect-failure"
    fault:
      abort:
        percentage:
          value: 0.1
        httpStatus: 500  # Chaos engineering: 0.1% de requêtes killées
  - match:
    - uri:
```

```
      prefix: /api/payment
    route:
    - destination:
        host: payment-service
        port:
          number: 80
    timeout: 10s  # Paiement plus lent
    retries:
      attempts: 2
  # ... autres routes similaires
```

3.1.2 Circuit Breaker & Rate Limiting

```yaml
# istio/destination-rule.yaml
apiVersion: networking.istio网状_structure.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: payment-service-circuit-breaker
  namespace: production
spec:
  host: payment-service
  trafficPolicy:
    connectionPool:
      tcp:
```

```
        maxConnections: 100
      http:
        http1MaxPendingRequests: 50
        maxRequestsPerConnection: 10
    circuitBreaker:
      consecutiveErrors: 5
      interval: 30s
      baseEjectionTime: 30s
      maxEjectionPercent: 50
    loadBalancer:
      simple: LEAST_REQUEST
      localityLbSetting:
        enabled: true
```

3.2 API Gateway (Kong) – Plugins & Configuration

```yaml
# kong/kong.yml (déclaratif)
_format_version: "3.0"

services:
- name: auth-service
  url: http://auth-service.production.svc.cluster.local:80
```

```
  routes:
  - name: auth-route
    paths: [/api/auth]
    strip_path: true
    methods: [GET, POST, PUT, DELETE]
  plugins:
  - name: rate-limiting
    config:
      minute: 100
      policy: local
      fault_tolerant: true
  - name: request-transformer
    config:
      add:
        headers:
        - "X-Request-ID:$uuid"  # Correlation ID
  - name: jwt
    config:
      secret_is_base64: false
      run_on_preflight: false

- name: payment-service
  url: http://payment-service.production.svc.cluster.local:80
  routes:
  - name: payment-route
    paths: [/api/payment]
```

```
    strip_path: true
  plugins:
  - name: rate-limiting
    config:
      minute: 20  # Plus restrictif
  - name: bot-detection
    config:
      whitelist: ["127.0.0.1"]
  - name: request-size-limiting
    config:
      allowed_payload_size: 128  # Kb
      size_unit: kilobytes

- name: websocket-service
  host: websocket-service.production.svc.cluster.local
  port: 80
  protocol: ws
  routes:
  - name: websocket-route
    paths: [/ws/tracking]
    strip_path: false
```

3.3 Design Pattern par Microservice

3.3.1 Auth Service – JWT & Session Management

```go
// auth-service/main.go (exemple en Go pour performance)
package main

import (
    "github.com/gofiber/fiber/v2"
    "github.com/golang-jwt/jwt/v5"
    "golang.org/x/crypto/bcrypt"
)

// Architecture: Clean Architecture
// - Entities: User, Role, Permission
// - Use Cases: Login, Register, RefreshToken, ValidateToken
// - Controllers: HTTP Handlers
// - Repositories: PostgreSQL interface

// Configuration JWT
const (
    AccessTokenTTL  = 15 * time.Minute
    RefreshTokenTTL = 7 * 24 * time.Hour
    JWTSecret       = "your-secret-key" // Depuis Vault
)
```

```
// Entity
type User struct {
    ID           uuid.UUID `json:"id"`
    Email        string    `json:"email"`
    PasswordHash string    `json:"-"` // Ne jamais sérialiser
    Role         string    `json:"role"`
    MFAEnabled   bool      `json:"mfa_enabled"`
    MFASecret    string    `json:"-"`
    CreatedAt    time.Time `json:"created_at"`
    UpdatedAt    time.Time `json:"updated_at"`
}

// Login Use Case
func (s *AuthService) Login(email, password string) (*TokenPair, error) {
    user, err := s.userRepo.GetByEmail(email)
    if err != nil {
        return nil, errors.New("invalid credentials")
    }

    // Timing attack safe comparison
    if err := bcrypt.CompareHashAndPassword([]byte(user.PasswordHash), []byte(password));
err != nil {
        return nil, errors.New("invalid credentials")
    }

    // MFA check si activé
    if user.MFAEnabled {
```

```
        // Retourner challenge, ne pas générer tokens encore
        return nil, errors.New("mfa_required")
    }

    return s.generateTokens(user)
}

// Token generation
func (s *AuthService) generateTokens(user *User) (*TokenPair, error) {
    accessToken := jwt.NewWithClaims(jwt.SigningMethodHS256, jwt.MapClaims{
        "sub":   user.ID.String(),
        "email": user.Email,
        "role":  user.Role,
        "exp":   time.Now().Add(AccessTokenTTL).Unix(),
        "iat":   time.Now().Unix(),
        "jti":   uuid.New().String(), // JWT ID unique
    })

    accessString, err := accessToken.SignedString([]byte(JWTSecret))
    if err != nil {
        return nil, err
    }

    refreshToken := jwt.NewWithClaims(jwt.SigningMethodHS256, jwt.MapClaims{
        "sub":  user.ID.String(),
        "exp":  time.Now().Add(RefreshTokenTTL).Unix(),
```

```
        "type": "refresh",
    })

    refreshString, err := refreshToken.SignedString([]byte(JWTSecret))
    if err != nil {
        return nil, err
    }

    // Stocker refresh token en DB (revocation possible)
    hashedRT := sha256.Sum256([]byte(refreshString))
    if err := s.tokenRepo.StoreRefreshToken(user.ID, hex.EncodeToString(hashedRT[:])); err
!= nil {
        return nil, err
    }

    return &TokenPair{
        AccessToken:  accessString,
        RefreshToken: refreshString,
        ExpiresIn:    int(AccessTokenTTL.Seconds()),
    }, nil
}

// Middleware de validation
func JWTProtected() fiber.Handler {
    return func(c *fiber.Ctx) error {
        authHeader := c.Get("Authorization")
        if authHeader == "" {
```

```
    return c.Status(401).JSON(fiber.Map{"error": "missing token"})
}

tokenString := strings.Replace(authHeader, "Bearer ", "", 1)
token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
    if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
        return nil, errors.New("unexpected signing method")
    }
    return []byte(JWTSecret), nil
})

if err != nil || !token.Valid {
    return c.Status(401).JSON(fiber.Map{"error": "invalid token"})
}

claims, ok := token.Claims.(jwt.MapClaims)
if !ok || !token.Valid {
    return c.Status(401).JSON(fiber.Map{"error": "invalid claims"})
}

// Vérifier expiration
if exp, ok := claims["exp"].(float64); ok {
    if time.Unix(int64(exp), 0).Before(time.Now()) {
        return c.Status(401).JSON(fiber.Map{"error": "token expired"})
    }
}
```

```
        // Injecter user dans contexte
        c.Locals("user_id", claims["sub"])
        c.Locals("user_role", claims["role"])

        return c.Next()
    }
}
```

3.3.2 Product Service – CQRS Pattern

```typescript
// product-service/src/application/use-cases/
// CQRS: Command Query Responsibility Segregation

// COMMAND (Write Model)
interface CreateProductCommand {
  name: string;
  categoryId: string;
  price: number;
  stock: number;
  producerId: string;
  images: string[];
}
```

```
class CreateProductHandler {
  constructor(
    private productRepo: ProductRepository,
    private eventBus: EventBus,
    private searchIndexer: SearchIndexer
  ) {}

  async execute(command: CreateProductCommand): Promise<Product> {
    // Validation métier
    if (command.price < 0) {
      throw new DomainError('Price must be positive');
    }

    // Création
    const product = Product.create({
      ...command,
      id: uuid(),
      createdAt: new Date(),
    });

    // Sauvegarde PostgreSQL (écriture)
    await this.productRepo.save(product);

    // Publication événement
    await this.eventBus.publish({
      type: 'PRODUCT_CREATED',
```

```
      data: product.toJSON(),
      timestamp: new Date(),
    });

    // Indexation async Elasticsearch (projection)
    await this.searchIndexer.indexProduct(product);

    return product;
  }
}

// QUERY (Read Model)
interface GetProductsQuery {
  categoryId?: string;
  minPrice?: number;
  maxPrice?: number;
  sortBy?: 'price' | 'popularity' | 'newest';
  page?: number;
  limit?: number;
}

class GetProductsHandler {
  constructor(
    private productViewRepo: ProductViewRepository
  ) {}
```

```
  async execute(query: GetProductsQuery): Promise<PaginatedProducts> {
    // Lecture depuis Vue Optimisée (matérialisée)
    // Pas de logique métier, juste du filtrage
    return this.productViewRepo.find(query);
  }
}

// Domain Event Handler (Projection)
class ProductCreatedProjection {
  constructor(
    private productViewRepo: ProductViewRepository
  ) {}

  async handle(event: ProductCreatedEvent) {
    // Met à jour la vue lecture
    await this.productViewRepo.upsert({
      id: event.data.id,
      name: event.data.name,
      categoryId: event.data.categoryId,
      price: event.data.price,
      stock: event.data.stock,
      searchableText: this.buildSearchText(event.data),
    });
  }

  private buildSearchText(product: Product): string {
```

```
    // Combine nom, catégorie, tags pour recherche full-text
    return `${product.name} ${product.category} ${product.tags.join(' ')}`.toLowerCase();
  }
}
```

3.3.3 Order Service – Saga Pattern (Distributed Transaction)

```python
# order-service/src/saga/order_saga.py
# Gère la transaction distribuée sur plusieurs services

class OrderSaga:
    """
    Saga Choreography: Chaque service écoute des événements et agit
    Saga Compensation: En cas d'échec, rollback étape par étape
    """

    def __init__(self):
        self.steps = []
        self.compensations = []

    async def create_order(self, user_id: UUID, items: List[OrderItem]):
        saga_id = uuid4()

        try:
```

```
# ÉTAPE 1: Créer order (état PENDING)
order = await self.order_repo.create(
    saga_id=saga_id,
    user_id=user_id,
    status=OrderStatus.PENDING,
    items=items
)
self.steps.append(lambda: order.id)  # Pour compensation

# Publier event ORDER_INITIATED
await self.event_bus.publish(
    OrderInitiatedEvent(
        saga_id=saga_id,
        order_id=order.id,
        user_id=user_id,
        total_amount=order.total_amount,
        items=items
    )
)

# ÉTAPE 2: Réserver stock (écoute par Product Service)
# Product Service publie STOCK_RESERVED ou STOCK_RESERVATION_FAILED

# ÉTAPE 3: Traiter paiement (écoute par Payment Service)
# Payment Service publie PAYMENT_SUCCEEDED ou PAYMENT_FAILED
```

```
            # ÉTAPE 4: Créer livraison (si paiement ok)
            # Delivery Service publie DELIVERY_SCHEDULED

            # ÉTAPE 5: Confirmer order
            await self.order_repo.update_status(
                order.id,
                OrderStatus.CONFIRMED
            )

            # Tout est ok, Saga terminée avec succès

        except StockReservationFailed as e:
            await self.compensate(saga_id, failed_step=1)
            raise OrderCreationError("Stock unavailable")

        except PaymentFailed as e:
            await self.compensate(saga_id, failed_step=2)
            raise OrderCreationError("Payment failed")

        except Exception as e:
            await self.compensate(saga_id, failed_step=len(self.steps))
            raise OrderCreationError("Unexpected error")

    async def compensate(self, saga_id: UUID, failed_step: int):
        """Rollback des étapes déjà effectuées"""
        for step in reversed(self.steps[:failed_step]):
```

```
            if step.type == "RESERVE_STOCK":
                await self.product_service.release_stock(
                    step.product_id,
                    step.quantity
                )
            elif step.type == "CREATE_DELIVERY":
                await self.delivery_service.cancel_delivery(
                    step.delivery_id
                )
            elif step.type == "PROCESS_PAYMENT":
                await payment_service.refund(step.payment_id)

            # Log compensation
            await self.audit_log.log(
                saga_id=saga_id,
                action="COMPENSATED",
                step=step.type
            )
```

3.4 Communication Inter-Services – Mécanismes et Protocoles

3.4.1 Synchronous (HTTP/gRPC)

```protobuf
// auth-service/proto/auth.proto
syntax = "proto3";

package auth;

service AuthService {
  rpc ValidateToken (ValidateTokenRequest) returns (ValidateTokenResponse);
  rpc GetUserInfo (GetUserInfoRequest) returns (GetUserInfoResponse);
}

message ValidateTokenRequest {
  string token = 1;
}

message ValidateTokenResponse {
  bool valid = 1;
  string user_id = 2;
  string role = 3;
  repeated string permissions = 4;
}

message GetUserInfoRequest {
  string user_id = 1;
}
```

```
message GetUserInfoResponse {
  string id = 1;
  string email = 2;
  string role = 3;
  string first_name = 4;
  string last_name = 5;
}
```

Appel depuis Go service :

```go
// Inter-service call with retry and circuit breaker
func (s *OrderService) CreateOrder(ctx context.Context, req *CreateOrderRequest) (*Order, error) {
    // Create gRPC client with istio sidecar
    conn, err := grpc.Dial("auth-service:80",
        grpc.WithTransportCredentials(insecure.NewCredentials()),
        grpc.WithUnaryInterceptor(retry.UnaryClientInterceptor(
            retry.WithMax(3),
            retry.WithPerRetryTimeout(2*time.Second),
        )),
    )
    if err != nil {
        return nil, err
    }
    defer conn.Close()
```

```
    client := auth.NewAuthServiceClient(conn)

    // Validate token
    authResp, err := client.ValidateToken(ctx, &auth.ValidateTokenRequest{
        Token: req.Token,
    })
    if err != nil {
        return nil, status.Errorf(codes.Unauthenticated, "invalid token")
    }

    userID := authResp.UserId
    // Continue order creation...
}
```

3.4.2 Asynchronous (Kafka – Event Bus)

```yaml
# kafka/topics.yaml
topics:
  - name: order.events
    partitions: 6
    replication-factor: 3
    retention: 168h  # 7 jours
    config:
      cleanup.policy: delete
```

```
    compression.type: lz4

- name: payment.events
  partitions: 3
  replication-factor: 3
  retention: 336h  # 14 jours (audit)

- name: delivery.events
  partitions: 6
  replication-factor: 3
  retention: 168h

- name: product.events
  partitions: 3
  replication-factor: 3
  retention: 720h  # 30 jours (historique produit)

- name: user.events
  partitions: 3
  replication-factor: 3
  retention: 168h

- name: analytics.events
  partitions: 12  # Haut débit
  replication-factor: 2  # Moins critique
  retention: 24h  # Court, juste pour streaming
```

```

**Event Schema (CloudEvents v1.0) :**

```json
{
  "specversion": "1.0",
  "type": "com.agrodeep.order.created",
  "source": "order-service",
  "id": "a3f5c9d2-e1b4-4f6a-8c9d-2e1b4f6a8c9d",
  "time": "2024-08-15T10:30:00Z",
  "datacontenttype": "application/json",
  "dataschema": "https://api.agrodeep.com/schemas/order.json",
  "data": {
    "order_id": "ord_12345",
    "user_id": "usr_67890",
    "total_amount": 125.50,
    "currency": "EUR",
    "items": [
      {
        "product_id": "prod_abc",
        "quantity": 3,
        "price": 25.50
      }
    ],
    "shipping_address": { /* ... */ }
  },
```
```

```
  "traceparent": "00-0af7651916cd43dd8448eb211c80319c-b7ad6b7169203331-01"
}
```

Producer (TypeScript) :

```typescript
// shared/kafka-producer.ts
import { Kafka, Producer } from 'kafkajs';

class EventProducer {
  private producer: Producer;

  constructor() {
    const kafka = new Kafka({
      clientId: 'order-service',
      brokers: ['kafka-0.kafka:9092', 'kafka-1.kafka:9092', 'kafka-2.kafka:9092'],
      retry: {
        initialRetryTime: 100,
        retries: 3
      },
    });

    this.producer = kafka.producer({
      allowAutoTopicCreation: false, // Important: topics gérés par ops
      transactionTimeout: 30000,
    });
```

```
}

async connect() {
  await this.producer.connect();
}

async publishOrderCreated(event: OrderCreatedEvent) {
  await this.producer.send({
    topic: 'order.events',
    messages: [{
      key: event.data.order_id, // Partition par order_id (ordering)
      value: JSON.stringify(event),
      headers: {
        'traceparent': event.traceparent || '',
      },
    }],
  });
}

async publishWithTransaction(events: Event[]) {
  // Pour saga: publier plusieurs events atomiquement
  const transaction = await this.producer.transaction();
  try {
    for (const event of events) {
      await transaction.send({
        topic: event.topic,
```

```
          messages: [event.message],
        });
      }
      await transaction.commit();
    } catch (err) {
      await transaction.abort();
      throw err;
    }
  }
}
```

Consumer :

```typescript
// payment-service/src/consumers/order-consumer.ts
import { Kafka, Consumer } from 'kafkajs';

class OrderEventsConsumer {
  private consumer: Consumer;

  constructor() {
    const kafka = new Kafka({
      clientId: 'payment-service',
      brokers: ['kafka-0.kafka:9092', 'kafka-1.kafka:9092', 'kafka-2.kafka:9092'],
    });
```

```
  this.consumer = kafka.consumer({
    groupId: 'payment-service-group',
    sessionTimeout: 30000,
    heartbeatInterval: 3000,
  });
}

async subscribe() {
  await this.consumer.connect();
  await this.consumer.subscribe({
    topic: 'order.events',
    fromBeginning: false, // Commencer à la fin du topic
  });

  await this.consumer.run({
    eachMessage: async ({ topic, partition, message }) => {
      const event = JSON.parse(message.value.toString());
      const span = tracer.startSpan('consume-order-event', {
        childOf: tracer.extract(FORMAT_HTTP_HEADERS, message.headers),
      });

      try {
        if (event.type === 'ORDER_CREATED') {
          await this.handleOrderCreated(event.data);
        } else if (event.type === 'ORDER_CANCELLED') {
          await this.handleOrderCancelled(event.data);
```

```
        }

        // Commit offset manuel pour garantir traitement
        await this.consumer.commitOffsets([{
          topic, partition,
          offset: (parseInt(message.offset) + 1).toString(),
        }]);
      } catch (error) {
        span.setTag('error', true);
        span.log({ event: 'error', message: error.message });

        // 1. Retry avec backoff
        // 2. Dead Letter Queue si échoue 3x
        await this.sendToDLQ(event, error);
      } finally {
        span.finish();
      }
    },
  });
}

private async handleOrderCreated(data: OrderCreatedData) {
  // 1. Vérifier si paiement déjà processed (idempotency)
  const exists = await this.paymentRepo.findByOrderId(data.order_id);
  if (exists) {
    console.log(`Payment already processed for order ${data.order_id}`);
```

```
  return;
}

// 2. Créer payment intent Stripe
const paymentIntent = await stripe.paymentIntents.create({
  amount: data.total_amount * 100, // centimes
  currency: data.currency,
  metadata: { order_id: data.order_id },
});

// 3. Sauvegarder en DB
await this.paymentRepo.create({
  order_id: data.order_id,
  stripe_payment_intent_id: paymentIntent.id,
  amount: data.total_amount,
  status: 'pending',
});

// 4. Publier event PAYMENT_INTENT_CREATED
await this.eventProducer.publish({
  type: 'PAYMENT_INTENT_CREATED',
  data: {
    order_id: data.order_id,
    payment_intent_id: paymentIntent.id,
    status: 'pending',
  },
```

```
    });
  }
}
```

---

4. COUCHE DONNÉES – DESIGN COMPLET

4.1 PostgreSQL – Schema & Partitionnement

```sql
-- ============================================
-- SCHEMA: agrodeep_production
-- ============================================

-- Tablespace pour performances
CREATE TABLESPACE fastspace LOCATION '/mnt/ssd/pgdata';
CREATE TABLESPACE slowspace LOCATION '/mnt/hdd/pgdata';

-- Users
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100),
```

```
    last_name VARCHAR(100),
    role VARCHAR(50) NOT NULL CHECK (role IN ('buyer', 'producer', 'driver', 'admin')),
    phone VARCHAR(20),
    email_verified BOOLEAN DEFAULT false,
    mfa_enabled BOOLEAN DEFAULT false,
    mfa_secret VARCHAR(32),
    last_login_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
) TABLESPACE fastspace;

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
CREATE INDEX idx_users_created_at ON users(created_at DESC);

-- Products
CREATE TABLE products (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    producer_id UUID NOT NULL REFERENCES users(id),
    name JSONB NOT NULL, -- Multilingue: {"en": "Tomato", "fr": "Tomate"}
    description JSONB,
    category_id UUID NOT NULL,
    price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
    currency VARCHAR(3) DEFAULT 'EUR',
    stock_quantity INTEGER NOT NULL DEFAULT 0 CHECK (stock_quantity >= 0),
    unit VARCHAR(50), -- "kg", "piece", "crate"
```

```
    images TEXT[], -- URLs vers MinIO
    tags TEXT[],
    is_active BOOLEAN DEFAULT true,
    is_organic BOOLEAN DEFAULT false,
    is_fair_trade BOOLEAN DEFAULT false,
    qr_code_data VARCHAR(255) UNIQUE, -- Pour traçabilité
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
) PARTITION BY RANGE (created_at);

-- Partitionnement mensuel pour produits (croissance importante)
CREATE TABLE products_2024_08 PARTITION OF products
    FOR VALUES FROM ('2024-08-01') TO ('2024-09-01');
CREATE TABLE products_2024_09 PARTITION OF products
    FOR VALUES FROM ('2024-09-01') TO ('2024-10-01');
-- ... script automatique pour créer partitions futures

CREATE INDEX idx_products_producer ON products(producer_id);
CREATE INDEX idx_products_category ON products(category_id);
CREATE INDEX idx_products_price ON products(price);
CREATE INDEX idx_products_active ON products(is_active) WHERE is_active = true;
CREATE INDEX idx_products_tags ON products USING GIN(tags);
CREATE INDEX idx_products_name_trgm ON products USING GIN (name gin_trgm_ops);
-- Recherche floue

-- Orders
CREATE TABLE orders (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    buyer_id UUID NOT NULL REFERENCES users(id),
    order_number VARCHAR(50) UNIQUE NOT NULL, -- Format: ORD-2024-08-15-0001
    status VARCHAR(50) NOT NULL CHECK (status IN ('pending', 'confirmed', 'processing',
'shipped', 'delivered', 'cancelled')),
    total_amount DECIMAL(10,2) NOT NULL,
    currency VARCHAR(3) DEFAULT 'EUR',
    shipping_address JSONB NOT NULL, -- {street, city, zip, lat, lng}
    billing_address JSONB,
    payment_method VARCHAR(50),
    payment_status VARCHAR(50),
    delivery_id UUID,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
) PARTITION BY RANGE (created_at);

CREATE TABLE orders_2024_08 PARTITION OF orders
    FOR VALUES FROM ('2024-08-01') TO ('2024-09-01');

CREATE INDEX idx_orders_buyer ON orders(buyer_id);
CREATE INDEX idx_orders_status ON orders(status);
CREATE INDEX idx_orders_created_at ON orders(created_at DESC);
CREATE INDEX idx_orders_order_number ON orders(order_number);

-- Order Items
CREATE TABLE order_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    order_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,
    product_id UUID NOT NULL REFERENCES products(id),
    quantity INTEGER NOT NULL CHECK (quantity > 0),
    unit_price DECIMAL(10,2) NOT NULL,
    total_price DECIMAL(10,2) NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_order_items_order ON order_items(order_id);
CREATE INDEX idx_order_items_product ON order_items(product_id);

-- Deliveries
CREATE TABLE deliveries (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    order_id UUID NOT NULL REFERENCES orders(id),
    driver_id UUID REFERENCES users(id),
    tracking_number VARCHAR(50) UNIQUE NOT NULL,
    status VARCHAR(50) NOT NULL CHECK (status IN ('assigned', 'picked_up', 'in_transit',
'delivered', 'failed')),
    pickup_address JSONB NOT NULL,
    delivery_address JSONB NOT NULL,
    estimated_delivery_at TIMESTAMPTZ,
    actual_delivery_at TIMESTAMPTZ,
    signature_url TEXT, -- URL MinIO
    photo_proof_url TEXT,
    coordinates JSONB, -- {lat, lng, accuracy, timestamp}
    created_at TIMESTAMPTZ DEFAULT NOW(),
```

```
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_deliveries_driver ON deliveries(driver_id);
CREATE INDEX idx_deliveries_status ON deliveries(status);
CREATE INDEX idx_deliveries_order ON deliveries(order_id);

-- Blockchain Events
CREATE TABLE blockchain_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    product_id UUID REFERENCES products(id),
    event_type VARCHAR(100) NOT NULL,
    transaction_hash VARCHAR(255) UNIQUE NOT NULL,
    block_number INTEGER,
    data JSONB NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_blockchain_product ON blockchain_events(product_id);
CREATE INDEX idx_blockchain_tx ON blockchain_events(transaction_hash);

-- Refresh tokens (revocation)
CREATE TABLE refresh_tokens (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id),
    token_hash VARCHAR(255) NOT NULL, -- SHA256 du token
```

```
    expires_at TIMESTAMPTZ NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    UNIQUE(user_id, token_hash)
);

CREATE INDEX idx_refresh_tokens_user ON refresh_tokens(user_id);
CREATE INDEX idx_refresh_tokens_expires ON refresh_tokens(expires_at)
    WHERE expires_at > NOW();

-- Audit log
CREATE TABLE audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    action VARCHAR(100) NOT NULL,
    resource_type VARCHAR(50),
    resource_id UUID,
    changes JSONB,
    ip_address INET,
    user_agent TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_audit_logs_user ON audit_logs(user_id);
CREATE INDEX idx_audit_logs_action ON audit_logs(action);
CREATE INDEX idx_audit_logs_created ON audit_logs(created_at DESC);
```

```
-- Fonction de mise à jour automatique updated_at
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER update_users_updated_at BEFORE UPDATE ON users
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
CREATE TRIGGER update_products_updated_at BEFORE UPDATE ON products
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
-- ... idem pour toutes les tables

-- Row Level Security (RLS) - pour multi-tenant SaaS
ALTER TABLE products ENABLE ROW LEVEL SECURITY;
ALTER TABLE orders ENABLE ROW LEVEL SECURITY;

-- Politique: un producteur ne voit que ses produits
CREATE POLICY product_owner_policy ON products
    FOR ALL
    TO producer_role
    USING (producer_id = current_setting('app.current_user_id')::UUID);

-- Politique: un acheteur ne voit que ses commandes
```

```
CREATE POLICY order_buyer_policy ON orders
    FOR ALL
    TO buyer_role
    USING (buyer_id = current_setting('app.current_user_id')::UUID);

-- Politique admin peut tout voir
CREATE POLICY admin_all_policy ON products
    FOR ALL
    TO admin_role
    USING (true);
```

4.2 Redis – Clés et Stratégie de Cache

```typescript
// redis/cache-strategy.ts
// Cache patterns

class CacheManager {
  private redis: Redis;

  constructor(redis: Redis) {
    this.redis = redis;
  }
```

```
// Pattern 1: Cache-Aside (lazy loading)
async getProduct(productId: string): Promise<Product | null> {
  const cacheKey = `product:${productId}`;

  // 1. Essaye cache
  const cached = await this.redis.get(cacheKey);
  if (cached) {
    return JSON.parse(cached);
  }

  // 2. Si miss, charge DB
  const product = await this.productRepo.findById(productId);
  if (!product) return null;

  // 3. Met en cache avec TTL
  await this.redis.setex(
    cacheKey,
    300, // 5 minutes
    JSON.stringify(product)
  );

  return product;
}

// Pattern 2: Write-Through (cache + DB同步)
```

```
async updateProduct(productId: string, updateData: Partial<Product>): Promise<Product> {
  // 1. Update DB
  const updated = await this.productRepo.update(productId, updateData);

  // 2. Update cache immédiatement
  const cacheKey = `product:${productId}`;
  await this.redis.setex(cacheKey, 300, JSON.stringify(updated));

  // 3. Invalider cache associé (liste produits)
  await this.redis.del('products:list:*');

  return updated;
}

// Pattern 3: Cache stampede prevention (mutex)
async getProductWithMutex(productId: string): Promise<Product | null> {
  const cacheKey = `product:${productId}`;
  const lockKey = `lock:${cacheKey}`;

  // Tenter d'acquérir lock
  const lock = await this.redis.set(lockKey, '1', 'EX', 10, 'NX');

  if (lock) {
    try {
      // Seul ce process charge la DB
      const product = await this.productRepo.findById(productId);
```

```
      if (product) {
        await this.redis.setex(cacheKey, 300, JSON.stringify(product));
      }
      return product;
    } finally {
      await this.redis.del(lockKey);
    }
  } else {
    // Attendre que lock soit libéré, puis réessayer cache
    await new Promise(resolve => setTimeout(resolve, 100));
    return this.getProduct(productId);
  }
}

// Pattern 4: Rate Limiting (sliding window)
async checkRateLimit(key: string, limit: number, windowSeconds: number):
Promise<boolean> {
  const windowKey = `ratelimit:${key}`;
  const now = Date.now();
  const windowStart = now - (windowSeconds * 1000);

  // Supprimer entrées hors fenêtre
  await this.redis.zremrangebyscore(windowKey, 0, windowStart);

  // Compter requêtes dans la fenêtre
  const count = await this.redis.zcount(windowKey, windowStart, now);
```

```
  if (count >= limit) {
    return false; // Rate limited
  }

  // Ajouter cette requête
  await this.redis.zadd(windowKey, now.toString(), now.toString());
  await this.redis.expire(windowKey, windowSeconds);

  return true;
}

// Pattern 5: Pub/Sub pour invalidation de cache
async subscribeToInvalidations() {
  const subscriber = this.redis.duplicate();
  await subscriber.connect();

  await subscriber.subscribe('cache:invalidations', (message) => {
    const { keys } = JSON.parse(message);
    keys.forEach(key => this.redis.del(key));
  });
}

async publishInvalidation(keys: string[]) {
  await this.redis.publish('cache:invalidations', JSON.stringify({ keys }));
}
```

```
  // Pattern 6: Distributed lock pour opération critique
  async acquireLock(lockName: string, ttl: number): Promise<string | null> {
    const lockValue = uuid();
    const acquired = await this.redis.set(
      `lock:${lockName}`,
      lockValue,
      'EX',
      ttl,
      'NX'
    );
    return acquired ? lockValue : null;
  }

  async releaseLock(lockName: string, lockValue: string): Promise<boolean> {
    // Script Lua pour release atomique (évite release par autre process)
    const script = `
      if redis.call("get", KEYS[1]) == ARGV[1] then
        return redis.call("del", KEYS[1])
      else
        return 0
      end
    `;
    const result = await this.redis.eval(script, 1, `lock:${lockName}`, lockValue);
    return result === 1;
  }
}
```

```

---

### **4.3 ClickHouse – Schema Analytique & Requêtes**

```sql
-- ============================================
-- BASE DE DONNÉES: agrodeep_analytics
-- ============================================

-- Events raw (ingestion haute vélocité)
CREATE TABLE events_raw (
    -- Colonnes génériques
    event_id UUID DEFAULT generateUUIDv4(),
    event_type LowCardinality(String),
    event_time DateTime DEFAULT now(),
    event_date Date DEFAULT toDate(event_time),

    -- Données utilisateur
    user_id UUID,
    user_role LowCardinality(String),
    session_id String,

    -- Données métier
    order_idNullable(UUID),
```
```

```
    product_idNullable(UUID),
    delivery_idNullable(UUID),

    -- Données techniques
    service LowCardinality(String),
    trace_id String,
    ip_address IPv4,
    user_agent String,

    -- Payload flexible
    properties String -- JSON stringifié
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(event_date)
ORDER BY (event_type, event_date, user_id)
TTL event_date + INTERVAL 30 DAY  -- Garder 30 jours en hot storage
SETTINGS index_granularity = 8192;

-- Matérialized View: Agrégation livraisons par jour
CREATE MATERIALIZED VIEW deliveries_daily_mv
ENGINE = SummingMergeTree()
PARTITION BY toYYYYMM(delivery_date)
ORDER BY (delivery_date, driver_id, status)
AS SELECT
    toDate(actual_delivery_at) as delivery_date,
    driver_id,
    status,  -- 'completed', 'failed'
```

```
    count() as count_deliveries,
    avg(delivery_duration_seconds) as avg_duration,
    avg(distance_km) as avg_distance
FROM events_raw
WHERE event_type = 'DELIVERY_COMPLETED'
GROUP BY delivery_date, driver_id, status;

-- Requêtes analytiques exemples

-- 1. Taux de conversion panier -> commande par jour
SELECT
    toDate(event_time) as day,
    countIf(event_type = 'CART_CHECKOUT_INITIATED') as carts_initiated,
    countIf(event_type = 'ORDER_CREATED') as orders_created,
    round(
        100 * orders_created / carts_initiated,
        2
    ) as conversion_rate
FROM events_raw
WHERE event_type IN ('CART_CHECKOUT_INITIATED', 'ORDER_CREATED')
GROUP BY day
ORDER BY day DESC
LIMIT 30;

-- 2. Produits les plus vus mais non achetés (analyse churn produit)
SELECT
```

```
    product_id,
    countIf(event_type = 'PRODUCT_VIEWED') as views,
    countIf(event_type = 'ORDER_ITEM_ADDED') as purchases,
    round(views / nullif(purchases, 0), 2) as view_to_purchase_ratio
FROM events_raw
WHERE event_type IN ('PRODUCT_VIEWED', 'ORDER_ITEM_ADDED')
GROUP BY product_id
HAVING views > 100
ORDER BY view_to_purchase_ratio DESC
LIMIT 100;


-- 3. Performance livraison par driver (SLA)
WITH delivery_metrics AS (
    SELECT
        delivery_id,
        driver_id,
        minIf(event_time, event_type = 'DELIVERY_ASSIGNED') as assigned_at,
        minIf(event_time, event_type = 'DELIVERY_PICKED_UP') as picked_up_at,
        minIf(event_time, event_type = 'DELIVERY_COMPLETED') as completed_at,
        dateDiff('second', assigned_at, completed_at) as total_duration,
        dateDiff('second', picked_up_at, completed_at) as travel_duration
    FROM events_raw
    WHERE event_type IN ('DELIVERY_ASSIGNED', 'DELIVERY_PICKED_UP',
'DELIVERY_COMPLETED')
    GROUP BY delivery_id, driver_id
)
SELECT
```

```
    driver_id,
    count() as total_deliveries,
    avg(total_duration) as avg_total_duration_seconds,
    quantile(0.95)(total_duration) as p95_duration,
    sumIf(1, travel_duration > 3600) as late_deliveries  > 1h
FROM delivery_metrics
WHERE completed_at >= now() - INTERVAL 7 DAY
GROUP BY driver_id
ORDER BY avg_total_duration_seconds DESC;

-- 4. Rétention utilisateurs (cohort analysis)
WITH user_cohorts AS (
    SELECT
        user_id,
        min(toDate(event_time)) as first_order_date,
        toStartOfMonth(first_order_date) as cohort_month
    FROM events_raw
    WHERE event_type = 'ORDER_CREATED'
    GROUP BY user_id
),
user_activity AS (
    SELECT
        user_id,
        toStartOfMonth(toDate(event_time)) as activity_month
    FROM events_raw
    WHERE event_type = 'ORDER_CREATED'
```

```
    GROUP BY user_id, activity_month
)
SELECT
    cohort_month,
    count(DISTINCT user_id) as cohort_size,
    -- Mois 1
    countDistinctIf(user_id, activity_month = date_add('month', 1, cohort_month)) as
month_1_retained,
    round(100 * month_1_retained / cohort_size, 2) as month_1_retention,
    -- Mois 3
    countDistinctIf(user_id, activity_month = date_add('month', 3, cohort_month)) as
month_3_retained,
    round(100 * month_3_retained / cohort_size, 2) as month_3_retention
FROM user_cohorts
LEFT JOIN user_activity USING (user_id)
GROUP BY cohort_month
ORDER BY cohort_month DESC
LIMIT 12;

-- 5. Anomalies détection (livraisons anormalement longues)
SELECT
    delivery_id,
    driver_id,
    total_duration,
    quantile(0.99)(total_duration) OVER () as p99_duration,
    total_duration > 3 * p99_duration as is_anomaly
FROM delivery_metrics
```

```
WHERE completed_at >= now() - INTERVAL 1 DAY;
```

4.4 Qdrant – Vector Database Design

```python
# ai-service/src/vector_store.py
from qdrant_client import QdrantClient, models

class VectorStore:
    def __init__(self, url: str = "http://qdrant:6333"):
        self.client = QdrantClient(url)
        self.product_collection = "products"
        self.user_collection = "users"

    def setup_collections(self):
        """Créer collections avec configuration optimale"""

        # Collection produits: recherche sémantique + filtrage
        self.client.create_collection(
            collection_name=self.product_collection,
            vectors_config=models.VectorParams(
                size=768,  # Dimension du modèle multilingual-e5-large
                distance=models.Distance.COSINE,
```

```
        hnsw_config=models.HnswConfig(
            m=16,  # Nombre de connexions par noeud
            ef_construct=100,  # Param construction index
        )
    ),
    optimizers_config=models.OptimizersConfigDiff(
        default_segment_number=5,  # Parallélisation
        indexing_threshold=10000,  # Indexer après 10k points
    ),
    quantization_config=models.ScalarQuantization(
        scalar=models.ScalarQuantizationConfig(
            type=models.ScalarType.INT8,  # Compression 4x
            quantile=0.99
        )
    )
)

# Collection users: embeddings préférences
self.client.create_collection(
    collection_name=self.user_collection,
    vectors_config=models.VectorParams(
        size=768,
        distance=models.Distance.COSINE,
        hnsw_config=models.HnswConfig(m=8, ef_construct=50)
    )
)
```

```
    # Créer payload indexes pour filtrage rapide
    self.client.create_payload_index(
        collection_name=self.product_collection,
        field_name="category_id",
        field_type=models.PayloadSchemaType.KEYWORD
    )

    self.client.create_payload_index(
        collection_name=self.product_collection,
        field_name="price",
        field_type=models.PayloadSchemaType.FLOAT
    )

    self.client.create_payload_index(
        collection_name=self.product_collection,
        field_name="is_active",
        field_type=models.PayloadSchemaType.BOOL
    )

def upsert_product(self, product: Product, embedding: List[float]):
    """Indexe produit avec embedding"""
    self.client.upsert(
        collection_name=self.product_collection,
        points=[
            models.PointStruct(
```

```
                id=product.id,
                vector=embedding,
                payload={
                    "name": product.name,
                    "description": product.description,
                    "category_id": str(product.category_id),
                    "producer_id": str(product.producer_id),
                    "price": float(product.price),
                    "stock_quantity": product.stock_quantity,
                    "is_active": product.is_active,
                    "is_organic": product.is_organic,
                    "tags": product.tags,
                    "created_at": product.created_at.isoformat(),
                }
            )
        ]
    )

def search_similar_products(
    self,
    query: str,
    user_id: str,
    limit: int = 10,
    filter: Optional[Dict] = None
) -> List[ScoredProduct]:
    """Recherche sémantique avec filtrage"""
```

```
        # 1. Générer embedding query (via service ML)
        query_embedding = self.embedding_model.encode(query)

        # 2. Récupérer embedding user (préférences historiques)
        user_embedding = self.get_user_embedding(user_id)

        # 3. Combiner (hybrid search)
        hybrid_embedding = self.combine_embeddings(query_embedding, user_embedding,
alpha=0.7)

        # 4. Construire filtre
        search_filter = models.Filter(
            must=[
                models.FieldCondition(
                    key="is_active",
                    match=models.MatchValue(value=True)
                ),
                models.FieldCondition(
                    key="stock_quantity",
                    range=models.Range(gte=1)
                ),
            ]
        )

        if filter?.get('category_id'):
            search_filter.must.append(
```

```
        models.FieldCondition(
            key="category_id",
            match=models.MatchValue(value=filter['category_id'])
        )
    )

if filter?.get('max_price'):
    search_filter.must.append(
        models.FieldCondition(
            key="price",
            range=models.Range(lte=filter['max_price'])
        )
    )

# 5. Search avec scoring
results = self.client.search(
    collection_name=self.product_collection,
    query_vector=hybrid_embedding,
    query_filter=search_filter,
    limit=limit,
    with_payload=True,
    with_vector=False,
    score_threshold=0.3,  # Filtrer faibles scores
)

return [
```

```
            ScoredProduct(
                id=point.id,
                score=point.score,
                payload=point.payload,
            )
            for point in results
        ]

    def recommend_for_user(self, user_id: str, limit: int = 10) -> List[str]:
        """Recommandation collaborative basée user embedding"""

        user_embedding = self.get_user_embedding(user_id)

        # Chercher users similaires
        similar_users = self.client.search(
            collection_name=self.user_collection,
            query_vector=user_embedding,
            limit=20,
            with_payload=True
        )

        # Extraire produits achetés par users similaires
        similar_user_ids = [point.payload['user_id'] for point in similar_users]

        # Query ClickHouse pour produits fréquents
        product_ids = self.clickhouse.query("""
```

```
            SELECT product_id, COUNT(*) as purchase_count
            FROM order_items
            WHERE user_id IN (%(user_ids)s)
              AND product_id NOT IN (
                SELECT product_id FROM order_items WHERE user_id = %(target_user)s
              )
            GROUP BY product_id
            ORDER BY purchase_count DESC
            LIMIT %(limit)s
        """, {
            'user_ids': similar_user_ids,
            'target_user': user_id,
            'limit': limit
        })

        return [row['product_id'] for row in product_ids]
```

---

5. DESIGN DES APIS – CONTRATS D'INTERFACE

5.1 Architecture API Gateway – Routes & Versions

```yaml
# api-gateway/routes.yaml
```

```yaml
base_path: /api

versions:
  - name: v1
    status: stable
    deprecation_date: null
    routes:

    # Auth
    - path: /v1/auth/login
      method: POST
      service: auth-service
      rate_limit: 10/min
      timeout: 5s
      description: Authenticate user

    - path: /v1/auth/refresh
      method: POST
      service: auth-service
      rate_limit: 5/min
      description: Refresh access token

    - path: /v1/auth/me
      method: GET
      service: auth-service
      middlewares: [jwt]
```

```
  description: Get current user info

# Products
- path: /v1/products
  method: GET
  service: product-service
  rate_limit: 100/min
  cache:  # Redis cache
    ttl: 300
    key: "products:list:{query.params.category}"
  description: List products

- path: /v1/products/:id
  method: GET
  service: product-service
  rate_limit: 200/min
  cache:
    ttl: 600
    key: "product:{params.id}"
  description: Get product details

- path: /v1/products
  method: POST
  service: product-service
  middlewares: [jwt, role:producer]
  rate_limit: 20/min
```

```
  body_size_limit: 10MB  # Pour images
  description: Create product

# Orders
- path: /v1/orders
  method: POST
  service: order-service
  middlewares: [jwt, role:buyer]
  rate_limit: 30/min
  description: Create order

- path: /v1/orders/:id
  method: GET
  service: order-service
  middlewares: [jwt]
  rate_limit: 100/min
  description: Get order

- path: /v1/orders/:id/cancel
  method: POST
  service: order-service
  middlewares: [jwt]
  rate_limit: 5/min
  description: Cancel order

# Payment
```

```
- path: /v1/payments/intent
  method: POST
  service: payment-service
  middlewares: [jwt]
  rate_limit: 10/min
  description: Create payment intent

- path: /v1/payments/webhook
  method: POST
  service: payment-service
  rate_limit: 100/min
  ip_whitelist: [Stripe IPs]  # Security
  description: Stripe webhook

# Delivery
- path: /v1/deliveries/:id/track
  method: GET
  service: delivery-service
  middlewares: [jwt]
  rate_limit: 50/min
  description: Get tracking info

# Search
- path: /v1/search
  method: GET
  service: search-service
```

```
      rate_limit: 100/min
      description: Search products

    # AI
    - path: /v1/ai/recommendations
      method: GET
      service: ai-service
      middlewares: [jwt]
      rate_limit: 200/min
      cache:
        ttl: 60
        key: "recommendations:{jwt.user_id}"
      description: Get product recommendations

    # Admin
    - path: /v1/admin/*
      method: '*'
      service: admin-service
      middlewares: [jwt, role:admin]
      rate_limit: 500/min
      description: Admin endpoints

# WebSocket routes
websocket:
  - path: /ws/tracking
    service: delivery-service
```

```
    middlewares: [jwt]
    description: Real-time delivery tracking

  - path: /ws/notifications
    service: notification-service
    middlewares: [jwt]
    description: Real-time notifications
```

5.2 API REST – Exemples de Contrats Complets

5.2.1 Authentification – JWT Flow

```http
### 1. Login
POST /v1/auth/login
Content-Type: application/json

{
  "email": "fermier@example.com",
  "password": "securePassword123",
  "mfa_code": "123456"  // Optionnel, si MFA activé
}
```

Response 200 OK:


```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refresh_token": "v2.local.eyJzdWIiOiIxMjM0NTY3ODkwIiw...",
  "expires_in": 900,
  "token_type": "Bearer",
  "user": {
    "id": "usr_12345",
    "email": "fermier@example.com",
    "role": "producer",
    "first_name": "Jean",
    "last_name": "Dupont",
    "mfa_enabled": true,
    "email_verified": true
  }
}
```


Response 401 Unauthorized:


```
{
  "error": "invalid_credentials",
  "message": "Email ou mot de passe incorrect"
}
```


Response 403 Forbidden (MFA requis):


```
{
  "error": "mfa_required",
```

```
  "message": "Code MFA requis",
  "mfa_method": "totp"
}
```

2. Rafraîchir token

```
POST /v1/auth/refresh
Content-Type: application/json

{
  "refresh_token": "v2.local.eyJzdWIiOiIxMjM0NTY3ODkwIiw..."
}
```

Response 200 OK:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expires_in": 900
}
```

3. Informations utilisateur

```
GET /v1/auth/me
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response 200 OK:

```
{
  "id": "usr_12345",
  "email": "fermier@example.com",
```

```
  "role": "producer",
  "permissions": [
    "product:create",
    "product:update:own",
    "order:view:own",
    "delivery:view:assigned"
  ],
  "profile": {
    "first_name": "Jean",
    "last_name": "Dupont",
    "phone": "+33 6 12 34 56 78",
    "company": "Ferme du Coin",
    "address": { /* ... */ },
    "verified": true,
    "created_at": "2024-08-01T10:00:00Z"
  }
}
```

4. Logout (revocation)

```
POST /v1/auth/logout
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "refresh_token": "v2.local.eyJzdWIiOiIxMjM0NTY3ODkwIiw..."
}
```

Response 204 No Content

5. Reset password

POST /v1/auth/reset-password

Content-Type: application/json

{

"email": "fermier@example.com"

}

Response 202 Accepted (email envoyé même si existe pas - security by obscurity)

6. Update password

PUT /v1/auth/password

Authorization: Bearer ...

Content-Type: application/json

{

"current_password": "oldPassword123",

"new_password": "newSecurePassword456"

}

Response 200 OK

```
```

5.2.2 Product Service – CRUD & Recherche

```http
### 1. Créer produit
POST /v1/products
Authorization: Bearer <token_producer>
Content-Type: application/json

{
  "name": {
    "fr": "Tomates Bio Marmande",
    "en": "Organic Marmande Tomatoes"
  },
  "description": {
    "fr": "Tomates rouges juteuses, cultivées en plein champ",
    "en": "Juicy red tomatoes, field-grown"
  },
  "category_id": "cat_vegetables",
  "price": 4.50,
  "currency": "EUR",
  "unit": "kg",
  "stock_quantity": 150,
  "images": [
    "https://cdn.agrodeep.com/products/tomato1.jpg",
    "https://cdn.agrodeep.com/products/tomato2.jpg"
  ],
```

```
  "tags": ["bio", "local", "saison"],
  "is_organic": true,
  "qr_code_data": "LOT-2024-08-FERMIER123-TOMATE"
}
```

Response 201 Created:

```
{
  "id": "prod_a1b2c3",
  "name": { ... },
  "price": 4.50,
  "stock_quantity": 150,
  "status": "active",
  "created_at": "2024-08-15T10:30:00Z",
  "updated_at": "2024-08-15T10:30:00Z",
  "qr_code_url": "https://api.agrodeep.com/v1/products/prod_a1b2c3/qrcode"
}
```

2. Lister produits (avec filtres)

GET
/v1/products?category=vegetables&min_price=1&max_price=10&is_organic=true&page=1&limit=20

Authorization: Bearer <token>

Cache-Control: public, max-age=300

Response 200 OK:

```
{
  "data": [
```

```
    {
      "id": "prod_a1b2c3",
      "name": {
        "fr": "Tomates Bio Marmande",
        "en": "Organic Marmande Tomatoes"
      },
      "price": 4.50,
      "currency": "EUR",
      "stock_quantity": 150,
      "is_organic": true,
      "images": [/*...*/],
      "producer": {
        "id": "usr_12345",
        "company": "Ferme du Coin",
        "rating": 4.8
      },
      "distance_km": 12.5  -- Calculé en temps réel depuis position user
    }
  ],
  "meta": {
    "current_page": 1,
    "last_page": 5,
    "total": 97,
    "per_page": 20
  }
}
```

3. Recherche full-text

GET /v1/products/search?q=tomate+bio&sort=relevance&lang=fr

Authorization: Bearer <token>

Response 200 OK:

```
{
  "data": [...],
  "meta": {
    "took_ms": 45,
    "query": "tomate bio",
    "suggestions": ["tomates bio", "tomato organic"]
  }
}
```

4. Update produit

PUT /v1/products/prod_a1b2c3

Authorization: Bearer <token_producer (owner)>

Content-Type: application/json

```
{
  "price": 4.80,
  "stock_quantity": 120
}
```

Response 200 OK (full product object)

5. Delete produit

DELETE /v1/products/prod_a1b2c3

Authorization: Bearer <token_producer>

Response 204 No Content

6. Batch update stock (pour producteurs)

PATCH /v1/products/batch

Authorization: Bearer <token_producer>

Content-Type: application/json

```
{
  "updates": [
    { "id": "prod_a1b2c3", "stock_quantity": 100 },
    { "id": "prod_d4e5f6", "stock_quantity": 50 }
  ]
}
```

Response 200 OK:

```
{
  "updated": 2,
  "errors": []
}
```

7. Upload image produit

POST /v1/products/prod_a1b2c3/images

Authorization: Bearer <token_producer>

Content-Type: multipart/form-data

file: <binary image data>

Response 201 Created:

{

"image_url": "https://cdn.agrodeep.com/products/prod_a1b2c3/image_1.jpg",

"thumbnail_url": "https://cdn.agrodeep.com/products/prod_a1b2c3/thumb_1.jpg"

}

```

#### **5.2.3 Order Service – Workflow Complexe**

```http

### 1. Créer commande

POST /v1/orders

Authorization: Bearer <token_buyer>

Content-Type: application/json

Idempotency-Key: key_unique_pour_ne_pas_dupliquer

{

"items": [

{

"product_id": "prod_a1b2c3",

```
      "quantity": 2,
      "notes": "Bien mûres s'il vous plaît"
    },
    {
      "product_id": "prod_d4e5f6",
      "quantity": 1
    }
  ],
  "shipping_address": {
    "street": "123 Rue du Marché",
    "city": "Lyon",
    "zip_code": "69001",
    "country": "FR",
    "latitude": 45.7640,
    "longitude": 4.8357
  },
  "billing_address": { /* optionnel, si différent */ },
  "preferred_delivery_date": "2024-08-17",
  "notes": "Livrer avant 12h si possible"
}
```

Response 202 Accepted (asynchrone):

```
{
  "order_id": "ord_789xyz",
  "order_number": "ORD-2024-08-15-0001",
  "status": "pending",
```

```
  "estimated_total": "13.50 EUR",
  "next_steps": [
    {
      "action": "payment_required",
      "url": "/v1/payments/intent"
    }
  ],
  "expires_at": "2024-08-15T11:00:00Z"  -- 30 min timeout stock
}
```

2. Suivre statut commande

```
GET /v1/orders/ord_789xyz
Authorization: Bearer <token_buyer>
```

Response 200 OK:

```
{
  "id": "ord_789xyz",
  "order_number": "ORD-2024-08-15-0001",
  "status": "processing",  -- pending -> processing -> shipped -> delivered
  "status_history": [
    {
      "status": "pending",
      "at": "2024-08-15T10:30:00Z",
      "actor": "system"
    },
    {
```

```
    "status": "processing",
    "at": "2024-08-15T10:35:00Z",
    "actor": "producer_123"
  }
],
"items": [
  {
    "product_id": "prod_a1b2c3",
    "name": "Tomates Bio Marmande",
    "quantity": 2,
    "unit_price": "4.50 EUR",
    "total_price": "9.00 EUR",
    "status": "reserved",  -- stock réservé
    "producer": {
      "id": "usr_12345",
      "company": "Ferme du Coin"
    }
  }
],
"total_amount": "13.50 EUR",
"currency": "EUR",
"shipping_address": { /* ... */ },
"delivery": {
  "tracking_number": "TRK-2024-08-15-0001",
  "status": "assigned",
  "driver": {
```

```
    "name": "Sophie Martin",
    "phone": "+33 6 98 76 54 32"
  },
  "estimated_delivery_at": "2024-08-17T10:00:00Z"
 },
 "timeline": {  -- ETA calculée
  "estimated_preparation": "2024-08-16T14:00:00Z",
  "estimated_pickup": "2024-08-17T08:00:00Z",
  "estimated_delivery": "2024-08-17T10:00:00Z"
 },
 "qr_code_url": "https://api.agrodeep.com/v1/orders/ord_789xyz/qrcode"  -- Pour suivi mobile
}
```

3. Annuler commande

```
POST /v1/orders/ord_789xyz/cancel
Authorization: Bearer <token_buyer>
Content-Type: application/json

{
 "reason": "Changement d'avis"
}

Response 200 OK:
{
 "status": "cancelled",
 "refund_status": "processing",  -- Si déjà payée
```

```
  "refund_amount": "13.50 EUR"
}
```

4. Lister commandes utilisateur

```
GET /v1/orders?status=delivered&page=1&limit=10
Authorization: Bearer <token_buyer>
```

Response 200 OK:

```
{
  "data": [...],
  "meta": { /* pagination */ }
}
```

5. Export facture

```
GET /v1/orders/ord_789xyz/invoice.pdf
Authorization: Bearer <token_buyer>
```

Response 200 OK:

```
Content-Type: application/pdf
Content-Disposition: inline; filename="invoice-ORD-2024-08-15-0001.pdf"
[binary PDF data]
```

6. Reorder (commande rapide)

```
POST /v1/orders/ord_789xyz/reorder
Authorization: Bearer <token_buyer>
Idempotency-Key: ...
```

Response 202 Accepted avec nouvelle commande

```

#### **5.2.4 WebSocket API – Protocole Real-time**

```javascript
// Connexion
const ws = new WebSocket('wss://api.agrodeep.com/ws/tracking');

// Authentification (après connexion)
ws.onopen = () => {
  ws.send(JSON.stringify({
    type: 'AUTH',
    token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'
  }));
};

// Messages du serveur
ws.onmessage = (event) => {
  const message = JSON.parse(event.data);

  switch (message.type) {
    case 'AUTH_SUCCESS':
      console.log('WebSocket authentifié');
      // S'abonner à une commande
```
```

```
      ws.send(JSON.stringify({
        type: 'SUBSCRIBE',
        order_id: 'ord_789xyz'
      }));
      break;

    case 'AUTH_ERROR':
      console.error('Auth failed:', message.error);
      break;

    case 'DELIVERY_UPDATE':
      // Position du livreur mise à jour
      updateMap(message.data);
      break;

    case 'ORDER_STATUS_CHANGED':
      // Statut commande changé
      updateUI(message.data);
      break;

    case 'NOTIFICATION':
      // Notification push
      showNotification(message.data);
      break;
  }
};
```

```
// Format message DELIVERY_UPDATE
{
  "type": "DELIVERY_UPDATE",
  "timestamp": "2024-08-17T09:15:30Z",
  "data": {
    "order_id": "ord_789xyz",
    "tracking_number": "TRK-2024-08-15-0001",
    "status": "in_transit",
    "coordinates": {
      "lat": 45.7500,
      "lng": 4.8400,
      "accuracy": 10,
      "timestamp": "2024-08-17T09:15:28Z"
    },
    "eta_minutes": 15,
    "driver_info": {
      "name": "Sophie Martin",
      "phone": "+33 6 98 76 54 32",
      "vehicle": "Van Peugeot"
    }
  }
}

// Format message ORDER_STATUS_CHANGED
{
```

```
  "type": "ORDER_STATUS_CHANGED",
  "data": {
    "order_id": "ord_789xyz",
    "old_status": "processing",
    "new_status": "shipped",
    "changed_at": "2024-08-17T08:00:00Z",
    "actor": "driver_456"
  }
}

// Format message NOTIFICATION
{
  "type": "NOTIFICATION",
  "data": {
    "id": "notif_123",
    "title": "Livreur à 5 minutes",
    "message": "Sophie arrive dans 5 min",
    "type": "delivery_approaching",
    "priority": "high",
    "actions": [
      {
        "label": "Appeler le livreur",
        "action": "call",
        "phone": "+33 6 98 76 54 32"
      }
    ]
```

```
  }
}
```

---

6. SÉCURITÉ – ARCHITECTURE DE DÉFENSE EN PROFONDEUR

6.1 Modèle de Menaces & Contrômes

```mermaid
graph TD
    A[Attaquant] -->|1. DDoS| B[Cloudflare WAF]
    A -->|2. SQL Injection| B
    A -->|3. XSS| B
    A -->|4. Brute Force| B

    B -->|Clean traffic| C[Kong API Gateway]

    C -->|5. Invalid Token| D[JWT Validation]
    C -->|6. Rate Limit| D
    C -->|7. API Abuse| D

    D -->|Authenticated| E[Istio Service Mesh mTLS]

    E -->|Inter-service| F[Services]
```

```
F -->|8. Privilege Escalation| G[RBAC per service]

F -->|9. Data Leak| G

F -->|10. Injection| G


G -->|Validated| H[Databases]


H -->|11. Data at Rest| I[Encryption AES-256]

H -->|12. Backup| I


subgraph "Monitoring"

    J[Alertes]

    K[Audit Logs]

end


F --> J

G --> K


style A fill:#f00

style B fill:#0f0

style C fill:#0f0

style D fill:#0f0

style E fill:#0f0

style F fill:#ff0

style G fill:#0f0

style H fill:#ff0
```

```
    style I fill:#0f0
    style J fill:#00f
    style K fill:#00f
```

6.2 Contrôles par Couche

6.2.1 Couche Edge (Cloudflare)

```yaml
rules:
  # 1. Rate Limiter global
  - name: global-rate-limit
    expression: "true"
    action: rate_limit
    rate_limit:
      characteristics: [cf.colo.id, ip.src]
      requests_per_period: 1000
      period: 60
      mitigation_timeout: 600

  # 2. Bloquer User-Agents suspects
  - name: block-suspicious-ua
    expression: |
      http.user_agent contains "sqlmap" or
```

```
    http.user_agent contains "nikto" or
    len(http.user_agent) lt 10
  action: block

# 3. Challenge JS pour IPs suspectes
- name: js-challenge-risky-ips
  expression: |
    (ip.geoip.country in {"CN" "RU"} and not ip.src in $trusted_ips)
  action: js_challenge

# 4. WAF règles OWASP
- name: owasp-top-10
  expression: |
    (http.request.uri.path contains "../") or
    (http.request.uri.query contains "union select") or
    (lower(http.request.body.raw) contains "<script>")
  action: block

# 5. Bloquer accès admin sauf IPs whitelist
- name: admin-ip-whitelist
  expression: |
    http.request.uri.path contains "/admin" and
    not ip.src in $admin_ips
  action: block
```

6.2.2 Couche API Gateway (Kong)

```yaml
# kong/plugins/security.yml
plugins:
  # 1. JWT validation
  - name: jwt
    config:
      uri_param_names: [jwt]
      cookie_names: [token]
      key_claim_name: iss
      secret_is_base64: false
      run_on_preflight: false

  # 2. Rate limiting par consumer
  - name: rate-limiting-advanced
    config:
      limit: [100, 2000]  # 100 par minute, 2000 par heure
      window_size: [60, 3600]
      identifier: consumer
      sync_rate: 10
      namespace: agrodeep-rate-limit

  # 3. Request validation (JSON Schema)
  - name: request-validator
```

```
    config:
      body_schema: |
        {
          "type": "object",
          "properties": {
            "email": { "type": "string", "format": "email" },
            "password": { "type": "string", "minLength": 8 }
          },
          "required": ["email", "password"]
        }

  # 4. Response transformer (cacher headers sensibles)
  - name: response-transformer
    config:
      remove.headers: [X-Powered-By, Server]

  # 5. IP restriction admin
  - name: ip-restriction
    config:
      whitelist: [10.0.0.0/8, 172.16.0.0/12]
      deny: [192.168.1.100]  # Bloquer IP spécifique
```

6.2.3 Couche Service (gRPC Interceptors)

```go
// auth-service/interceptors/auth.go
func UnaryAuthInterceptor() grpc.UnaryServerInterceptor {
    return func(
        ctx context.Context,
        req interface{},
        info *grpc.UnaryServerInfo,
        handler grpc.UnaryHandler,
    ) (interface{}, error) {
        // 1. Extraire metadata
        md, ok := metadata.FromIncomingContext(ctx)
        if !ok {
            return nil, status.Error(codes.Unauthenticated, "missing metadata")
        }

        // 2. Valider JWT
        authHeader := md.Get("authorization")
        if len(authHeader) == 0 {
            return nil, status.Error(codes.Unauthenticated, "missing token")
        }

        token := strings.TrimPrefix(authHeader[0], "Bearer ")
        claims, err := validateJWT(token)
        if err != nil {
            return nil, status.Error(codes.Unauthenticated, "invalid token")
        }
```

```
        // 3. Injecter dans context
        ctx = context.WithValue(ctx, "user_id", claims.UserID)
        ctx = context.WithValue(ctx, "user_role", claims.Role)

        // 4. Vérifier permissions pour cette méthode
        requiredRole := getRequiredRole(info.FullMethod)
        if !hasPermission(claims.Role, requiredRole) {
            return nil, status.Error(codes.PermissionDenied, "insufficient permissions")
        }

        // 5. Audit log
        auditLog := &AuditLog{
            UserID:    claims.UserID,
            Action:    info.FullMethod,
            Timestamp: time.Now(),
            IP:        getClientIP(md),
        }
        go saveAuditLog(auditLog) // Async

        return handler(ctx, req)
    }
}
```

6.2.4 Couche Base de Données

```sql
-- PostgreSQL security functions

-- Row Level Security activé
ALTER TABLE products ENABLE ROW LEVEL SECURITY;
ALTER TABLE orders ENABLE ROW LEVEL SECURITY;

-- Politiques RBAC
CREATE POLICY product_owner ON products FOR ALL
    TO producer_role
    USING (producer_id = current_setting('app.user_id')::UUID)
    WITH CHECK (producer_id = current_setting('app.user_id')::UUID);

CREATE POLICY order_buyer ON orders FOR ALL
    TO buyer_role
    USING (buyer_id = current_setting('app.user_id')::UUID);

-- Audit trigger
CREATE OR REPLACE FUNCTION audit_trigger_func()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO audit_logs (
        table_name,
        record_id,
```

```
        operation,
        old_values,
        new_values,
        user_id
    ) VALUES (
        TG_TABLE_NAME,
        COALESCE(NEW.id, OLD.id),
        TG_OP,
        to_jsonb(OLD),
        to_jsonb(NEW),
        current_setting('app.user_id', true)::UUID
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER audit_products
    AFTER INSERT OR UPDATE OR DELETE ON products
    FOR EACH ROW EXECUTE FUNCTION audit_trigger_func();

-- Chiffrement colonnes sensibles
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Paiements: ne jamais stocker PAN (Primary Account Number)
CREATE TABLE payment_methods (
    id UUID PRIMARY KEY,
```

```
    user_id UUID REFERENCES users(id),
    stripe_payment_method_id VARCHAR(255),  -- Token externe
    last_four VARCHAR(4),
    brand VARCHAR(20),
    exp_month INTEGER,
    exp_year INTEGER,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    -- Colonne chiffrée (juste au cas où)
    encrypted_data BYTEA  -- Données additionnelles chiffrées
);

-- Fonction pour chiffrer/déchiffrer
CREATE OR REPLACE FUNCTION encrypt_sensitive(data TEXT, key TEXT)
RETURNS BYTEA AS $$
BEGIN
    RETURN pgp_sym_encrypt(data, key, 'cipher-algo=aes256');
END;
$$ LANGUAGE plpgsql;

-- Backup encryption (WAL segments encryptés)
ALTER SYSTEM SET wal_encryption = on;
```

6.3 Gestion des Secrets

```yaml
# vault/values.yaml (exemple values pour Vault Helm)
global:
  enabled: true

server:
  ha:
    enabled: true
    replicas: 3

  dataStorage:
    enabled: true
    size: 10Gi
    storageClass: fast-ssd

  auditStorage:
    enabled: true
    size: 10Gi

  standalone:
    enabled: false

ui:
  enabled: true
  serviceType: LoadBalancer
```

```
  serviceNodePort: null

injector:
  enabled: true
  # Injection automatique dans pods annotés

secrets:
  - name: stripe-api-key
    path: secret/data/stripe
    data:
      secret_key: sk_live_...
      publishable_key: pk_live_...

  - name: db-credentials
    path: secret/data/postgres
    data:
      username: agrodeep_admin
      password: verysecurepassword

  - name: jwt-secret
    path: secret/data/jwt
    data:
      secret: supersecret512bitkeyatleast64characterslong
      refresh_secret: another512bitkey

  - name: minio-s3
```

```
    path: secret/data/minio
    data:
      access_key: minioadmin
      secret_key: minioadmin123
```

Injection dans pod Kubernetes :

```yaml
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    metadata:
      annotations:
        vault.hashicorp.com/agent-inject: "true"
        vault.hashicorp.com/role: "agrodeep-app"
        vault.hashicorp.com/agent-inject-secret-stripe: "secret/data/stripe"
        vault.hashicorp.com/agent-inject-template-stripe: |
          {{ with secret "secret/data/stripe" -}}
          STRIPE_SECRET_KEY={{ .Data.data.secret_key }}
          STRIPE_PUBLISHABLE_KEY={{ .Data.data.publishable_key }}
          {{- end }}
        vault.hashicorp.com/agent-inject-secret-db: "secret/data/postgres"
        vault.hashicorp.com/agent-inject-template-db: |
          {{ with secret "secret/data/postgres" -}}
```

```
        DB_USER={{ .Data.data.username }}
        DB_PASS={{ .Data.data.password }}
        {{- end }}
    spec:
      containers:
      - name: app
        env:
        - name: STRIPE_SECRET_KEY
          valueFrom:
            secretKeyRef:
              name: stripe
              key: secret_key
```

6.4 Monitoring Sécurité

Dashboard Grafana – Security Metrics :

```promql
# Requêtes Prometheus pour sécurité

# 1. Nombre de tentatives de login échouées par IP
sum(rate(http_requests_total{status="401", path="/v1/auth/login"}[5m])) by (client_ip)
```

2. IPs avec plus de 10 échecs/heure

count(sum(rate(http_requests_total{status="401", path="/v1/auth/login"}[5m])) by (client_ip) > 10/3600)

3. Requêtes bloquées par rate limiter

sum(rate(kong_http_status{status="429"}[5m]))

4. Tentatives d'accès admin non autorisées

sum(rate(http_requests_total{status="403", path=~"/v1/admin.*"}[5m]))

5. Tokens JWT invalides

sum(rate(jwt_validation_failures_total[5m]))

6. Mises à jour non autorisées (RBAC violations)

sum(rate(rbac_denied_total[5m]))

7. Tentatives d'accès à données d'autres users

sum(rate(api_access_violations_total[5m]))

8. Volume traffic par pays (GeoIP)

sum(rate(http_requests_total[5m])) by (geoip_country_code)

```

**Alertes critiques :**

- **Brute force** : > 50 login fails / 10 min depuis même IP → Bloquer IP via Cloudflare API
- **Data exfiltration** : > 1000 requêtes / min depuis un token → Révoquer token
- **RBAC bypass** : Accès admin détecté → PagerDuty immédiat
```

- **Secrets leak** : "BEGIN RSA PRIVATE KEY" dans logs → Alert critique + rotate secrets

---

7. MONITORING & OBSERVABILITÉ – DESIGN COMPLET

7.1 Prometheus – Scraping Configuration

```yaml
# prometheus.yml (fichier de config)
global:
  scrape_interval: 15s
  evaluation_interval: 15s
  external_labels:
    cluster: agrodeep-prod
    region: fr-par

# Remote write pour long term storage (Thanos/Cortex)
remote_write:
- url: http://thanos-receive:19291/api/v1/receive
  queue_config:
    capacity: 2500
    max_samples_per_send: 1000
    batch_send_deadline: 5s
  write_relabel_configs:
  - source_labels: [__name__]
```

```
    regex: 'go_.*'
    action: drop  # Drop metrics inutiles

# Scrape configs
scrape_configs:

# 1. Kubernetes pods (découverte auto)
- job_name: 'kubernetes-pods'
  kubernetes_sd_configs:
  - role: pod
    namespaces:
      names: [production, staging]
  relabel_configs:
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
    action: keep
    regex: true
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
    action: replace
    target_label: __metrics_path__
    regex: (.+)
  - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
    action: replace
    regex: ([^:]+)(?::\d+)?;(\d+)
    replacement: $1:$2
    target_label: __address__
  - action: labelmap
```

```
    regex: __meta_kubernetes_pod_label_(.+)
  - source_labels: [__meta_kubernetes_namespace]
    action: replace
    target_label: kubernetes_namespace
  - source_labels: [__meta_kubernetes_pod_name]
    action: replace
    target_label: kubernetes_pod_name

# 2. Kubernetes nodes
- job_name: 'kubernetes-nodes'
  static_configs:
  - targets: ['node-exporter:9100']
  metric_relabel_configs:
  - source_labels: [mountpoint]
    regex: "/var/lib/kubelet/pods/.*"
    action: drop  # Drop metrics pods

# 3. API Gateway Kong
- job_name: 'kong'
  static_configs:
  - targets: ['kong-admin:8001']
  metrics_path: /metrics

# 4. PostgreSQL
- job_name: 'postgresql'
  static_configs:
```

```
  - targets: ['postgres-exporter:9187']

# 5. Redis
- job_name: 'redis'
  static_configs:
  - targets: ['redis-exporter:9121']

# 6. RabbitMQ (si utilisé)
- job_name: 'rabbitmq'
  static_configs:
  - targets: ['rabbitmq-exporter:9419']

# 7. Blackbox exporter (uptime external)
- job_name: 'blackbox-http'
  metrics_path: /probe
  params:
    module: [http_2xx]
  static_configs:
  - targets:
    - https://api.agrodeep.com/health
    - https://web.agrodeep.com
  relabel_configs:
  - source_labels: [__address__]
    target_label: __param_target
  - source_labels: [__param_target]
    target_label: instance
```

```
  - target_label: __address__
    replacement: blackbox-exporter:9115

# 8. Custom application metrics
- job_name: 'custom-metrics'
  static_configs:
  - targets:
    - 'auth-service:3000'
    - 'product-service:3000'
    - 'order-service:3000'
    - 'payment-service:3000'
    - 'delivery-service:3000'
    - 'ai-service:3000'
```

7.2 Custom Metrics – Instrumentation Application

```typescript
// shared/metrics.ts
import { Counter, Histogram, Gauge, register } from 'prom-client';

// 1. Counter: Nombre d'opérations
export const httpRequestsTotal = new Counter({
  name: 'http_requests_total',
```

```
  help: 'Total HTTP requests',
  labelNames: ['method', 'path', 'status', 'service'],
});

export const ordersCreatedTotal = new Counter({
  name: 'orders_created_total',
  help: 'Total orders created',
  labelNames: ['status', 'payment_method'],
});

export const paymentsFailedTotal = new Counter({
  name: 'payments_failed_total',
  help: 'Total payment failures',
  labelNames: ['error_code', 'gateway'],
});

// 2. Histogram: Latence distributions
export const httpRequestDuration = new Histogram({
  name: 'http_request_duration_seconds',
  help: 'HTTP request latency',
  labelNames: ['method', 'path', 'status'],
  buckets: [0.001, 0.01, 0.05, 0.1, 0.5, 1, 2, 5],
});

export const dbQueryDuration = new Histogram({
  name: 'db_query_duration_seconds',
```

```
  help: 'Database query latency',
  labelNames: ['operation', 'table'],
  buckets: [0.001, 0.01, 0.05, 0.1, 0.5],
});

// 3. Gauge: Valeurs instantanées
export const activeConnections = new Gauge({
  name: 'active_connections',
  help: 'Active connections',
  labelNames: ['type'],
});

export const productStockGauge = new Gauge({
  name: 'product_stock_quantity',
  help: 'Current stock per product',
  labelNames: ['product_id'],
  collect() {
    // Mise à jour périodique
    const stocks = db.query('SELECT id, stock_quantity FROM products');
    stocks.forEach(row => {
      this.set({ product_id: row.id }, row.stock_quantity);
    });
  }
});

// 4. Middleware Express
```

```
export function metricsMiddleware(req: Request, res: Response, next: NextFunction) {
  const start = Date.now();

  res.on('finish', () => {
    const duration = (Date.now() - start) / 1000;
    const route = req.route ? req.route.path : req.path;

    httpRequestsTotal.inc({
      method: req.method,
      path: route,
      status: res.statusCode.toString(),
      service: process.env.SERVICE_NAME,
    });

    httpRequestDuration.observe({
      method: req.method,
      path: route,
      status: res.statusCode.toString(),
    }, duration);
  });

  next();
}

// 5. Health check endpoint
app.get('/metrics', async (req, res) => {
```

```
  res.set('Content-Type', register.contentType);
  res.end(await register.metrics());
});

app.get('/health', (req, res) => {
  // Liveness probe
  res.json({
    status: 'ok',
    timestamp: new Date().toISOString(),
    uptime: process.uptime(),
  });
});

app.get('/ready', async (req, res) => {
  // Readiness probe: vérifie dépendances
  try {
    await db.query('SELECT 1');
    await redis.ping();
    res.json({ status: 'ready' });
  } catch (err) {
    res.status(503).json({ status: 'unready', error: err.message });
  }
});
```

7.3 Loki – Logging Pipeline

```yaml
# promtail-config.yml (agent de collecte)
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
- url: http://loki:3100/loki/api/v1/push
  batchwait: 1s
  batchsize: 153600

scrape_configs:
- job_name: kubernetes-pods
  kubernetes_sd_configs:
  - role: pod
    namespaces:
      names: [production, staging]
  pipeline_stages:
  - cri: {}  # Parse format CRI-O/containerd
  - labeldrop:
```

```
  - filename  # Label inutile
- match:
    selector: '{app="auth-service"}'
    stages:
    - json:
        expressions:
          level: level
          trace_id: trace_id
          user_id: user_id
    - labels:
        level:
        trace_id:
    - template:
        source: output_msg
        template: '{{ .level }} | user={{ .user_id }} | {{ .msg }}'
    - output:
        source: output_msg


relabel_configs:
- source_labels: [__meta_kubernetes_pod_node_name]
  target_label: __host__
- action: labelmap
  regex: __meta_kubernetes_pod_label_(.+)
- action: replace
  replacement: $1
  separator: /
```

```
    source_labels:
    - __meta_kubernetes_namespace
    - __meta_kubernetes_pod_name
    target_label: job
  - action: replace
    source_labels:
    - __meta_kubernetes_pod_name
    target_label: pod
  - action: replace
    source_labels:
    - __meta_kubernetes_namespace
    target_label: namespace

- job_name: application-logs
  static_configs:
  - targets:
      - localhost
    labels:
      job: app-logs
      app: agrodeep
      environment: production
  pipeline_stages:
  - json:
      expressions:
        level: level
        msg: message
```

```
      trace_id: traceid
      span_id: spanid
  - labels:
      level:
  - timestamp:
      format: RFC3339Nano
      source: timestamp
```

Application logging (Winston) :

```typescript
// shared/logger.ts
import winston from 'winston';

const logger = winston.createLogger({
  level: process.env.LOG_LEVEL || 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()  # Format JSON pour Loki
  ),
  defaultMeta: {
    service: process.env.SERVICE_NAME,
    version: process.env.VERSION,
    environment: process.env.NODE_ENV,
  },
```

```
  transports: [
    new winston.transports.Console({
      format: winston.format.combine(
        winston.format.colorize(),
        winston.format.simple()
      ),
    }),
    new winston.transports.File({
      filename: 'logs/error.log',
      level: 'error',
      maxsize: 5242880, // 5MB
      maxFiles: 5,
    }),
    new winston.transports.File({
      filename: 'logs/combined.log',
      maxsize: 5242880,
      maxFiles: 5,
    }),
  ],
});

// Middleware pour logs HTTP
export function loggerMiddleware(req: Request, res: Response, next: NextFunction) {
  const start = Date.now();
  const traceId = req.headers['x-trace-id'] || generateTraceId();
```

```
  // Ajouter trace ID aux logs
  logger.defaultMeta = {
    ...logger.defaultMeta,
    trace_id: traceId,
    user_id: req.user?.id,
  };

  res.on('finish', () => {
    const duration = Date.now() - start;
    logger.info({
      message: `${req.method} ${req.path} ${res.statusCode}`,
      method: req.method,
      path: req.path,
      status_code: res.statusCode,
      duration_ms: duration,
      user_agent: req.get('User-Agent'),
      ip: req.ip,
      query_params: req.query,
      // Ne jamais logger body/mots de passe
    });
  });

  next();
}

// Usage
```

```
logger.info("Order created", { order_id: order.id, amount: order.total });
logger.warn("Stock low", { product_id: product.id, stock: 5 });
logger.error("Payment failed", { error: err.message, payment_intent: pi.id });
```

7.4 Jaeger – Distributed Tracing

```yaml
# jaeger/values.yaml
storage:
  type: elasticsearch
  elasticsearch:
    server_urls: http://elasticsearch:9200
    username: elastic
    password: changeme

collector:
  enabled: true
  max_traces_per_second: 10000
  queue_size: 2000

agent:
  enabled: true
  strategy_type: probabilistic
```

```
  sampling_param: 0.1  # 10% de sampling en production

query:
  enabled: true
  service_type: LoadBalancer

ingester:
  enabled: true
```

Instrumentation OpenTelemetry :

```typescript
// shared/tracing.ts
import { NodeSDK } from '@opentelemetry/sdk-node';
import { getNodeAutoInstrumentations } from '@opentelemetry/auto-instrumentations-node';
import { JaegerExporter } from '@opentelemetry/exporter-jaeger';
import { Resource } from '@opentelemetry/resources';
import { SemanticResourceAttributes } from '@opentelemetry/semantic-conventions';

const sdk = new NodeSDK({
  resource: new Resource({
    [SemanticResourceAttributes.SERVICE_NAME]: process.env.SERVICE_NAME,
    [SemanticResourceAttributes.SERVICE_VERSION]: process.env.VERSION,
    [SemanticResourceAttributes.DEPLOYMENT_ENVIRONMENT]:
process.env.NODE_ENV,
  }),
  traceExporter: new JaegerExporter({
```

```
    endpoint: 'http://jaeger-collector:14268/api/traces',
  }),
  instrumentations: [getNodeAutoInstrumentations({
    // Custom instrumentation pour express, http, pg, redis
    '@opentelemetry/instrumentation-http': {
      ignoreIncomingPaths: [/\/health/, /\/metrics/, /\/ready/],
    },
  })],
});

sdk.start();

// Helper pour créer spans manuels
import { trace } from '@opentelemetry/api';

export function createSpan(name: string) {
  const tracer = trace.getTracer(process.env.SERVICE_NAME);
  return tracer.startSpan(name);
}

// Usage dans API
app.post('/v1/orders', async (req, res) => {
  const span = createSpan('create-order');

  try {
    span.setAttribute('user.id', req.user.id);
```

```
    span.setAttribute('items.count', req.body.items.length);

    // Sous-span pour validation
    const validationSpan = createSpan('validate-order');
    await validateOrder(req.body);
    validationSpan.end();

    // Sous-span pour création DB
    const dbSpan = createSpan('insert-order-db');
    const order = await orderRepo.create(req.body);
    dbSpan.end();

    // Sous-span pour event Kafka
    const kafkaSpan = createSpan('publish-order-event');
    await eventProducer.publishOrderCreated(order);
    kafkaSpan.end();

    res.status(201).json(order);
  } catch (error) {
    span.setStatus({ code: 2, message: error.message });
    span.recordException(error);
    throw error;
  } finally {
    span.end();
  }
});
```

```

---

### **7.5 Dashboards Grafana – Exemples**

**Dashboard: API Overview**

```json
{
  "dashboard": {
    "title": "API Overview",
    "panels": [
      {
        "title": "Requests/sec",
        "targets": [{
          "expr": "sum(rate(http_requests_total[5m]))",
          "legendFormat": "Total"
        }],
        "type": "graph"
      },
      {
        "title": "Error Rate %",
        "targets": [{
          "expr": "sum(rate(http_requests_total{status=~'5..'}[5m])) /
sum(rate(http_requests_total[5m])) * 100",
          "legendFormat": "5xx Error %"
        }],
```
```

```
        "type": "stat",
        "thresholds": "0.1,1"
      },
      {
        "title": "Latency p95",
        "targets": [{
          "expr": "histogram_quantile(0.95,
sum(rate(http_request_duration_seconds_bucket[5m])) by (le))",
          "legendFormat": "p95"
        }],
        "type": "graph"
      },
      {
        "title": "Active Connections",
        "targets": [{
          "expr": "sum(active_connections)",
          "legendFormat": "Active"
        }],
        "type": "stat"
      }
    ]
  }
}
```

Dashboard: Business Metrics (Superset)

```sql
```

```
-- Dataset: orders (ClickHouse)
SELECT
    toDate(o.created_at) as order_date,
    count(*) as total_orders,
    sum(o.total_amount) as revenue,
    countDistinct(o.buyer_id) as unique_customers,
    avg(o.total_amount) as avg_order_value,
    -- Cohort: première commande de chaque client
    countIf(o.buyer_id IN (
        SELECT buyer_id FROM orders
        WHERE toDate(created_at) = today() - 30
        GROUP BY buyer_id HAVING count(*) = 1
    )) as new_customers,
    -- Taux de retour produit
    countIf(o.status = 'returned') / count(*) as return_rate
FROM orders o
WHERE toDate(o.created_at) >= today() - 30
GROUP BY order_date
ORDER BY order_date DESC
```

---

8. DÉPLOIEMENT – PIPELINE CI/CD COMPLET

8.1 GitOps avec ArgoCD

```yaml
# argocd/application.yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: agrodeep-production
  namespace: argocd
spec:
  project: default

  source:
    repoURL: https://github.com/agrodeep/infrastructure.git
    targetRevision: main
    path: k8s/overlays/production

  destination:
    server: https://kubernetes.default.svc
    namespace: production

  syncPolicy:
    automated:
      prune: true  # Supprimer ressources supprimées du repo
      selfHeal: true  # Auto-corriger drift
      allowEmpty: false
```

```
    syncOptions:
    - CreateNamespace=true
    - PrunePropagationPolicy=foreground

    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m

  # Health checks
  ignoreDifferences:
  - group: apps
    kind: Deployment
    jsonPointers:
    - /spec/replicas  # Laisser HPA gérer le scaling

  info:
  - name: 'Description'
    value: 'Production deployment of AgroDeep platform'
```

8.2 GitHub Actions – Workflow de déploiement

```yaml
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]
    paths:
    - 'services/**'
    - 'k8s/**'

env:
  REGISTRY: ghcr.io/agrodeep
  KUBE_CONFIG: ${{ secrets.KUBE_CONFIG }}

jobs:
  # Job 1: Tests & Security
  test:
    runs-on: ubuntu-latest
    outputs:
      image_tag: ${{ steps.meta.outputs.tags }}

    steps:
    - uses: actions/checkout@v3
      with:
```

```
    fetch-depth: 0  # Pour changed-files

- name: Get changed services
  id: changed
  uses: tj-actions/changed-files@v35
  with:
    files: services/**

- name: Set up Node.js
  if: steps.changed.outputs.any_changed == 'true'
  uses: actions/setup-node@v3
  with:
    node-version: '20'
    cache: 'npm'

- name: Install dependencies
  if: steps.changed.outputs.any_changed == 'true'
  run: npm ci

- name: Run linter
  if: steps.changed.outputs.any_changed == 'true'
  run: npm run lint

- name: Run tests
  if: steps.changed.outputs.any_changed == 'true'
  run: npm test -- --coverage
```

```
    - name: Security scan (Trivy)
      if: steps.changed.outputs.any_changed == 'true'
      run: |
        for service in ${{ steps.changed.outputs.all_changed_files }}; do
          docker build -t test:$service .
          trivy image --exit-code 1 --severity=HIGH,CRITICAL test:$service
        done

    - name: Build images
      if: steps.changed.outputs.any_changed == 'true'
      run: |
        for service in ${{ steps.changed.outputs.all_changed_files }}; do
          docker build -t $REGISTRY/$service:${{ github.sha }} -f services/$service/Dockerfile .
        done

    - name: Push images
      if: steps.changed.outputs.any_changed == 'true'
      run: |
        echo ${{ secrets.GITHUB_TOKEN }} | docker login ghcr.io -u ${{ github.actor }}
--password-stdin
        for service in ${{ steps.changed.outputs.all_changed_files }}; do
          docker push $REGISTRY/$service:${{ github.sha }}
          docker tag $REGISTRY/$service:${{ github.sha }} $REGISTRY/$service:latest
          docker push $REGISTRY/$service:latest
        done
```

```
    - name: Generate image tags
      id: meta
      uses: docker/metadata-action@v4
      with:
        images: ${{ env.REGISTRY }}/app
        tags: |
          type=sha,prefix={{branch}}-
          type=raw,value=latest,enable={{is_default_branch}}

  # Job 2: Deploy to staging
  deploy-staging:
    needs: test
    runs-on: ubuntu-latest
    environment: staging

    steps:
    - uses: actions/checkout@v3

    - name: Setup kubectl
      uses: azure/setup-kubectl@v3
      with:
        version: 'v1.28.0'

    - name: Configure kube config
      run: echo "${{ secrets.KUBE_CONFIG_STAGING }}" | base64 -d > kubeconfig
```

```
    - name: Update image tags in staging
      run: |
        cd k8s/overlays/staging
        kustomize edit set image app=$REGISTRY/app:${{ github.sha }}
        for service in ${{ needs.test.outputs.changed_services }}; do
          kustomize edit set image $service=$REGISTRY/$service:${{ github.sha }}
        done

    - name: Deploy to staging
      run: |
        kubectl apply -k k8s/overlays/staging --kubeconfig=kubeconfig

    - name: Wait for rollout
      run: |
        kubectl rollout status deployment/auth-service -n staging --timeout=5m
--kubeconfig=kubeconfig
        kubectl rollout status deployment/product-service -n staging --timeout=5m
--kubeconfig=kubeconfig
        # ... autres services

    - name: Run smoke tests
      run: |
        npm run test:smoke -- --env=staging --url=https://staging.agrodeep.com

    - name: Run E2E tests
      run: |
        npm run test:e2e -- --env=staging
```

```
  - name: Notify Slack
    if: failure()
    uses: slackapi/slack-github-action@v1
    with:
      payload: |
        {
          "text": "❌ Déploiement staging échoué",
          "service": "${{ needs.test.outputs.changed_services }}",
          "commit": "${{ github.sha }}"
        }

# Job 3: Deploy to production (manual approval)
deploy-production:
  needs: [test, deploy-staging]
  runs-on: ubuntu-latest
  environment: production
  # Attente approval manuel dans GitHub

  steps:
  - uses: actions/checkout@v3

  - name: Setup kubectl
    uses: azure/setup-kubectl@v3

  - name: Configure kube config
```

```
      run: echo "${{ secrets.KUBE_CONFIG_PROD }}" | base64 -d > kubeconfig

    - name: Update image tags in production
      run: |
        cd k8s/overlays/production
        kustomize edit set image app=$REGISTRY/app:${{ github.sha }}
        for service in ${{ needs.test.outputs.changed_services }}; do
          kustomize edit set image $service=$REGISTRY/$service:${{ github.sha }}
        done

    - name: Create canary deployment (10%)
      run: |
        # 1. Créer canary avec 10% du traffic
        kubectl apply -k k8s/overlays/production-canary --kubeconfig=kubeconfig

        # 2. Attendre 5 min
        sleep 300

        # 3. Vérifier erreurs
        ERROR_RATE=$(kubectl exec -n monitoring prometheus-0 -- \
          curl -s
'http://localhost:9090/api/v1/query?query=sum(rate(http_requests_total{status=~\"5..\",canar
y=\"true\"}[5m]))/sum(rate(http_requests_total{canary=\"true\"}[5m]))')

        if (( $(echo "$ERROR_RATE > 0.01" | bc -l) )); then
          echo "ERROR: Canary error rate too high: $ERROR_RATE"
          kubectl delete -k k8s/overlays/production-canary --kubeconfig=kubeconfig
```

```
          exit 1
        fi

    - name: Progressive rollout (50%)
      run: |
        # Mettre à jour à 50%
        kubectl patch argocd agrodeep-production --type='json' -p='[{"op": "replace", "path": "/spec/source/path", "value": "k8s/overlays/production-50"}]'
        sleep 600
        # Vérifications...

    - name: Full rollout (100%)
      run: |
        kubectl apply -k k8s/overlays/production --kubeconfig=kubeconfig

    - name: Tag release
      if: success()
      run: |
        git tag -a v1.2.3-${{ github.sha }} -m "Release $(date)"
        git push origin v1.2.3-${{ github.sha }}

    - name: Notify Slack
      uses: slackapi/slack-github-action@v1
      with:
        payload: |
          {
            "text": "✅ Déploiement production réussi",
```

```
            "version": "v1.2.3-${{ github.sha }}",
            "services": "${{ needs.test.outputs.changed_services }}"
          }
```

8.3 Helm Charts – Package Kubernetes

```yaml
# helm/agrodeep-service/Chart.yaml
apiVersion: v2
name: agrodeep-service
description: Helm chart pour services AgroDeep
type: application
version: 0.1.0
appVersion: "1.0.0"

dependencies:
- name: postgresql
  version: 12.5.6
  repository: https://charts.bitnami.com/bitnami
  condition: postgresql.enabled
- name: redis
  version: 17.14.3
  repository: https://charts.bitnami.com/bitnami
```

```
  condition: redis.enabled

# helm/agrodeep-service/values.yaml
replicaCount: 3

image:
  repository: ghcr.io/agrodeep/service
  pullPolicy: IfNotPresent
  tag: ""

serviceAccount:
  create: true
  annotations: {}
  name: ""

podAnnotations:
  prometheus.io/scrape: "true"
  prometheus.io/port: "3000"
  prometheus.io/path: "/metrics"

podSecurityContext:
  fsGroup: 1000

securityContext:
  runAsNonRoot: true
  runAsUser: 1000
```

```
  readOnlyRootFilesystem: true
  allowPrivilegeEscalation: false

service:
  type: ClusterIP
  port: 80
  targetPort: 3000

ingress:
  enabled: true
  className: nginx
  annotations:
    kubernetes.io/ingress.class: nginx
    cert-manager.io/cluster-issuer: letsencrypt-prod
    nginx.ingress.kubernetes.io/rate-limit: "100"
    nginx.ingress.kubernetes.io/rate-limit-window: "1m"
  hosts:
  - host: api.agrodeep.com
    paths:
    - path: /api/service
      pathType: Prefix
  tls:
  - secretName: agrodeep-tls
    hosts:
    - api.agrodeep.com
```

```
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 100m
    memory: 128Mi

autoscaling:
  enabled: true
  minReplicas: 3
  maxReplicas: 20
  targetCPUUtilizationPercentage: 70
  targetMemoryUtilizationPercentage: 80
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300
      policies:
      - type: Percent
        value: 10
        periodSeconds: 60
    scaleUp:
      stabilizationWindowSeconds: 60
      policies:
      - type: Percent
        value: 100
```

```
      periodSeconds: 60
    - type: Pods
      value: 2
      periodSeconds: 60
    selectPolicy: Max

nodeSelector: {}

tolerations: []

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app.kubernetes.io/name
          operator: In
          values:
          - agrodeep-service
      topologyKey: kubernetes.io/hostname

env:
  NODE_ENV: production
  LOG_LEVEL: info

envFromSecret:
```

```
- name: app-secrets  # Reference Kubernetes secret

# Probes configuration
livenessProbe:
  httpGet:
    path: /health
    port: http
  initialDelaySeconds: 30
  periodSeconds: 10
  timeoutSeconds: 5
  failureThreshold: 3

readinessProbe:
  httpGet:
    path: /ready
    port: http
  initialDelaySeconds: 5
  periodSeconds: 5
  timeoutSeconds: 3
  failureThreshold: 2

startupProbe:
  httpGet:
    path: /ready
    port: http
  initialDelaySeconds: 10
```

```
  periodSeconds: 5
  timeoutSeconds: 3
  failureThreshold: 30  # Attendre max 150s pour démarrer
```

---

9. SCALING & PERFORMANCE – STRATÉGIES

9.1 Horizontal Pod Autoscaler (HPA)

```yaml
# k8s/hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: order-service-hpa
  namespace: production
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: order-service
  minReplicas: 3
  maxReplicas: 50
  metrics:
```

```
- type: Resource
  resource:
    name: cpu
    target:
      type: Utilization
      averageUtilization: 70
- type: Resource
  resource:
    name: memory
    target:
      type: Utilization
      averageUtilization: 80
- type: Pods
  pods:
    metric:
      name: http_requests_per_second
    target:
      type: AverageValue
      averageValue: "1000"  # 1000 req/s par pod
- type: External
  external:
    metric:
      name: kafka_consumer_lag
      selector:
        matchLabels:
          topic: order.events
```

```
        consumer_group: order-service
    target:
      type: Value
      value: "1000"  # Scale si lag > 1000 messages

  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300  # Attendre 5min avant de descendre
      policies:
      - type: Percent
        value: 10  # Descendre de 10% max par période
        periodSeconds: 60
    scaleUp:
      stabilizationWindowSeconds: 60
      policies:
      - type: Percent
        value: 100  # Monter de 100% max
        periodSeconds: 60
      - type: Pods
        value: 4  # Ou ajouter 4 pods
        periodSeconds: 60
      selectPolicy: Max  # Prendre la policy la plus agressive
```

Metrics custom pour HPA :

```yaml
```

```yaml
# k8s/custom-metrics.yaml
apiVersion: custom.metrics.k8s.io/v1beta1
kind: MetricValueList
metadata:
  name: http-requests-per-second
spec:
  target:
    type: Pods
    pods:
      metricName: http_requests_per_second
      selector:
        matchLabels:
          app: order-service
---
# Prometheus Adapter config
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-adapter-config
data:
  config.yaml: |
    rules:
    - seriesQuery: 'http_requests_total{namespace="production",service="order-service"}'
      resources:
        overrides:
          namespace: {resource: "namespace"}
```

```
        service: {resource: "service"}
      name:
        matches: "http_requests_total"
        as: "http_requests_per_second"
      metricsQuery: 'sum(rate(http_requests_total{<<.LabelMatchers>>}[5m])) by (<<.GroupBy>>)'
```

9.2 Vertical Scaling (VPA)

```yaml
# k8s/vpa.yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: payment-service-vpa
  namespace: production
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: payment-service
  updatePolicy:
    updateMode: "Auto"  # Auto-update les ressources
  resourcePolicy:
```

```
    containerPolicies:
    - containerName: payment-service
      minAllowed:
        cpu: 100m
        memory: 128Mi
      maxAllowed:
        cpu: 2000m  # 2 coeurs
        memory: 4Gi
      controlledResources: ["cpu", "memory"]
      controlledValues: RequestsAndLimits  # Ajuster requests ET limits
```

9.3 Database Scaling

PostgreSQL – Read Replicas

```yaml
# values.postgresql.yaml
architecture: replication
primary:
  persistence:
    enabled: true
    storageClass: fast-ssd
    size: 100Gi
  resources:
```

```
    requests:
      cpu: 2
      memory: 4Gi
    limits:
      cpu: 4
      memory: 8Gi

readReplicas:
  replicaCount: 3
  persistence:
    enabled: true
    storageClass: fast-ssd
    size: 100Gi
  resources:
    requests:
      cpu: 1
      memory: 2Gi

pgpool:
  enabled: true
  replicaCount: 2
  adminPassword: admin123
  customUsers:
    usernames: "agrodeep_user"
    passwords: "userpass"
```

Routing lecture/écriture dans app :

```typescript
// db/connection.ts
const PRIMARY_DB = process.env.DATABASE_PRIMARY_URL;  // master
const REPLICA_DB = process.env.DATABASE_REPLICA_URL;   // pool read replicas

// Write operations (INSERT, UPDATE, DELETE) → primary
export const writePool = new Pool({ connectionString: PRIMARY_DB });

// Read operations (SELECT) → replicas
export const readPool = new Pool({ connectionString: REPLICA_DB });

// Middleware pour router automatiquement
export const dbRouter = (req: Request, res: Response, next: NextFunction) => {
  req.db = req.method === 'GET' ? readPool : writePool;
  next();
};
```

Redis Cluster

```yaml
# values.redis.yaml
architecture: replication
auth:
  enabled: true
```

```
  password: redis123

master:
  persistence:
    enabled: true
    storageClass: fast-ssd
    size: 10Gi
  resources:
    requests:
      memory: 2Gi

replica:
  replicaCount: 3
  persistence:
    enabled: true
    size: 10Gi

sentinel:
  enabled: true
  quorum: 2  # (n/2)+1 pour 3 nodes
  downAfterMilliseconds: 5000
  failoverTimeout: 60000
```

ClickHouse Sharding

```sql
```

```
-- Cluster configuration: 3 shards, 2 replicas
-- Sur chaque node: /etc/clickhouse-server/config.d/cluster.xml
<remote_servers>
  <agrodeep_cluster>
    <shard>
      <replica>
        <host>clickhouse-0</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>clickhouse-1</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <replica>
        <host>clickhouse-2</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>clickhouse-3</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <replica>
```

```
        <host>clickhouse-4</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>clickhouse-5</host>
        <port>9000</port>
      </replica>
    </shard>
  </agrodeep_cluster>
</remote_servers>

-- Créer table distribuée
CREATE TABLE events_dist ON CLUSTER agrodeep_cluster
AS events_raw
ENGINE = Distributed(agrodeep_cluster, default, events_raw, rand());
```

9.4 Caching Stratégique

Cache multi-niveau :

```typescript
// lib/cache/multi-tier.ts
class MultiTierCache {
```

```
constructor(
  private l1: Redis,         // Cache local (1ms)
  private l2: RedisCluster, // Cache distribué (5ms)
  private l3: string         // MinIO/S3 pour gros objets
) {}

async get(key: string): Promise<any> {
  // L1: Mémoire locale (Node.js Map)
  const local = this.localCache.get(key);
  if (local) return local;

  // L2: Redis
  const redis = await this.l2.get(key);
  if (redis) {
    this.localCache.set(key, JSON.parse(redis), 60); // TTL 60s local
    return JSON.parse(redis);
  }

  // L3: Database + mise en cache async
  const dbData = await this.db.query('SELECT * FROM cacheable WHERE key = ?', [key]);
  if (dbData) {
    // Pas await, pour répondre rapidement
    this.set(key, dbData).catch(console.error);
    return dbData;
  }
```

```
    return null;
  }

  async set(key: string, value: any, ttl: number = 300) {
    // L1
    this.localCache.set(key, value, ttl);

    // L2
    await this.l2.setex(key, ttl, JSON.stringify(value));

    // L3 (optionnel pour gros objets)
    if (Buffer.byteLength(JSON.stringify(value)) > 100000) {
      await this.s3.putObject({
        Bucket: 'cache',
        Key: key,
        Body: JSON.stringify(value),
        Expires: new Date(Date.now() + ttl * 1000)
      });
    }
  }
}
```

Cache warming :

```typescript
// jobs/cache-warmer.ts
```

```
// S'exécute toutes les heures
async function warmProductCache() {
  const popularProducts = await db.query(`
    SELECT product_id FROM (
      SELECT product_id, COUNT(*) as c
      FROM order_items
      WHERE created_at > NOW() - INTERVAL 7 DAY
      GROUP BY product_id
      ORDER BY c DESC
      LIMIT 100
    )
  `);

  for (const { product_id } of popularProducts) {
    const product = await productService.getProduct(product_id);
    await cache.set(`product:${product_id}`, product, 3600); // 1h
  }
}
```

9.5 CDN & Asset Optimization

Cloudflare Configuration :

```yaml
```

```yaml
# cloudflare/page-rules.yml
rules:
  - name: Static Assets Cache
    pattern: "cdn.agrodeep.com/*"
    settings:
      cache_level: cache_everything
      edge_cache_ttl: 86400  # 24h
      browser_cache_ttl: 3600
      always_online: on

  - name: API No Cache
    pattern: "api.agrodeep.com/*"
    settings:
      cache_level: bypass
      always_use_https: on

  - name: Redirect HTTP to HTTPS
    pattern: "*agrodeep.com/*"
    settings:
      always_use_https: on

  - name: Security Headers
    pattern: "*agrodeep.com/*"
    settings:
      security_level: high
      waf: on
```

```
      rocket_loader: off  # Pas pour API
```

Image optimization (MinIO + Thumbor) :

```yaml
# docker-compose for image processing
version: '3.8'
services:
  thumbor:
    image: minimalcompact/thumbor:7.5.2
    environment:
      SECURITY_KEY: your-secret-key
      MAX_WIDTH: 2000
      MAX_HEIGHT: 2000
      QUALITY: 85
      AUTO_WEBP: "True"
    ports: ["8000:8000"]

  # Upload flow:
  # 1. Client upload image brute vers MinIO
  # 2. Lambda/Trigger notifie Thumbor
  # 3. Thumbor génère multiples variants (thumb, medium, large) et les stocke dans MinIO
  # 4. URLs: https://cdn.agrodeep.com/thumb/{hash}.jpg
```

---

10. DISASTER RECOVERY – PLAN DE REPRISE

10.1 Backup Strategy

```bash
#!/bin/bash
# backup.sh - Exécuté tous les jours à 2h

# 1. PostgreSQL
pg_dumpall -U postgres -h postgres-primary | \
  gzip | \
  aws s3 cp - s3://agrodeep-backups/postgres/$(date +%Y-%m-%d).sql.gz \
  --storage-class STANDARD_IA

# 2. Redis
redis-cli BGSAVE
sleep 10
aws s3 cp /data/redis/dump.rdb s3://agrodeep-backups/redis/$(date +%Y-%m-%d).rdb \
  --storage-class STANDARD_IA

# 3. ClickHouse
clickhouse-client --query="BACKUP TABLE events_raw TO Disk('s3',
'backups/clickhouse/$(date +%Y-%m-%d)')"

# 4. MinIO (snapshot)
mc mirror --remove --overwrite minio/agrodeep-storage s3://agrodeep-backups/minio/
```

```
# 5. Kubernetes manifests
kubectl get all -n production -o yaml | \
  gzip | \
  aws s3 cp - s3://agrodeep-backups/k8s/$(date +%Y-%m-%d).yaml.gz

# 6. TTL: garder 30 jours
aws s3 ls s3://agrodeep-backups/ --recursive | \
  awk '$1 < "'$(date -d '30 days ago' +%Y-%m-%d)'" {print $4}' | \
  xargs aws s3 rm
```

10.2 Recovery Procedures

```yaml
# Disaster: Cluster Kubernetes complet down

procedure:
  1. rebuild_cluster:
     - Provisionner nouveau cluster (Terraform)
     - Restorer ArgoCD: kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
     - Configurer repo Git: argocd repo add https://github.com/agrodeep/infrastructure
     - Sync applications: argocd app sync agrodeep-production --prune

  2. restore_databases:
     - PostgreSQL: pg_restore from S3
```

```
    - Redis: RDB restore et restart
    - ClickHouse: RESTORE FROM S3
    - MinIO: mirror depuis S3

  3. validate:
    - Health checks
    - Smoke tests
    - Traffic 1% → 10% → 100%

  rto: 1h  # Recovery Time Objective
  rpo: 15min  # Recovery Point Objective (perte max données)
```

---

11. DOCUMENTATION – DEVELOPER PORTAL

11.1 Swagger UI – API Explorer

```yaml
# swagger/openapi.yaml
openapi: 3.0.3
info:
  title: AgroDeep API
  version: 1.0.0
  description: |
```

```
    Plateforme agricole connectée - API publique
    Toutes les requêtes nécessitent un token JWT.

servers:
- url: https://api.agrodeep.com/v1
  description: Production
- url: https://staging-api.agrodeep.com/v1
  description: Staging

security:
- bearerAuth: []

paths:
  /auth/login:
    post:
      summary: Login
      tags: [Authentication]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [email, password]
              properties:
                email:
```

```
            type: string
            format: email
            example: fermier@example.com
          password:
            type: string
            format: password
            example: securePassword123
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TokenResponse'
      '401':
        $ref: '#/components/responses/Unauthorized'

/products:
  get:
    summary: List products
    tags: [Products]
    parameters:
    - name: category
      in: query
      schema:
        type: string
```

```
          example: vegetables
        - name: page
          in: query
          schema:
            type: integer
            example: 1
        - name: limit
          in: query
          schema:
            type: integer
            example: 20
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/PaginatedProducts'


components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
```

```
schemas:
  TokenResponse:
    type: object
    properties:
      access_token:
        type: string
        example: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
      refresh_token:
        type: string
      expires_in:
        type: integer
        example: 900

  Product:
    type: object
    properties:
      id:
        type: string
        format: uuid
      name:
        type: object
        additionalProperties:
          type: string
      price:
        type: number
        format: float
```

```
  PaginatedProducts:
    type: object
    properties:
      data:
        type: array
        items:
          $ref: '#/components/schemas/Product'
      meta:
        type: object
        properties:
          total:
            type: integer
          page:
            type: integer


responses:
  Unauthorized:
    description: Invalid credentials
    content:
      application/json:
        schema:
          type: object
```