# Vẽ và giải thích góc nhìn triển khai của Event-Driven Architecture được đề xuất trong bài lab 04? Liệt kê các công cụ có thể sử dụng và các bước cần thực hiện để triển khai hệ thống theo góc nhìn đề xuất.

1. Góc nhìn Triển khai (Deployment View) của Event-Driven Architecture
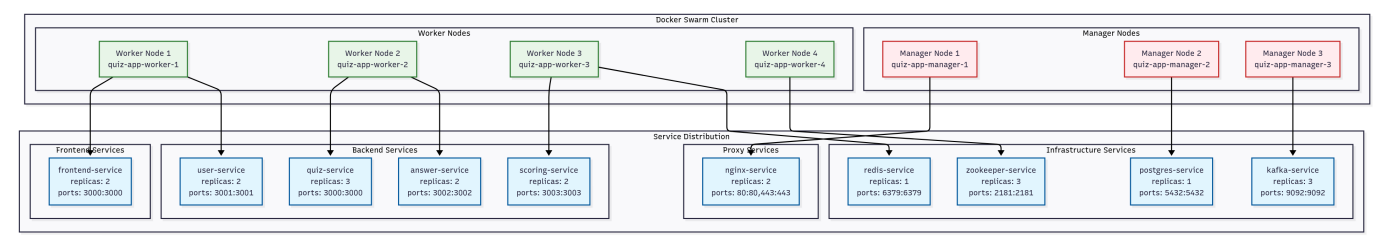
1.1 Tổng quan Deployment Architecture

Góc nhìn triển khai mô tả cách các thành phần phần mềm được phân phối và triển khai trên hạ tầng vật lý/ảo hoá, bao gồm servers, containers, networks và dependencies.
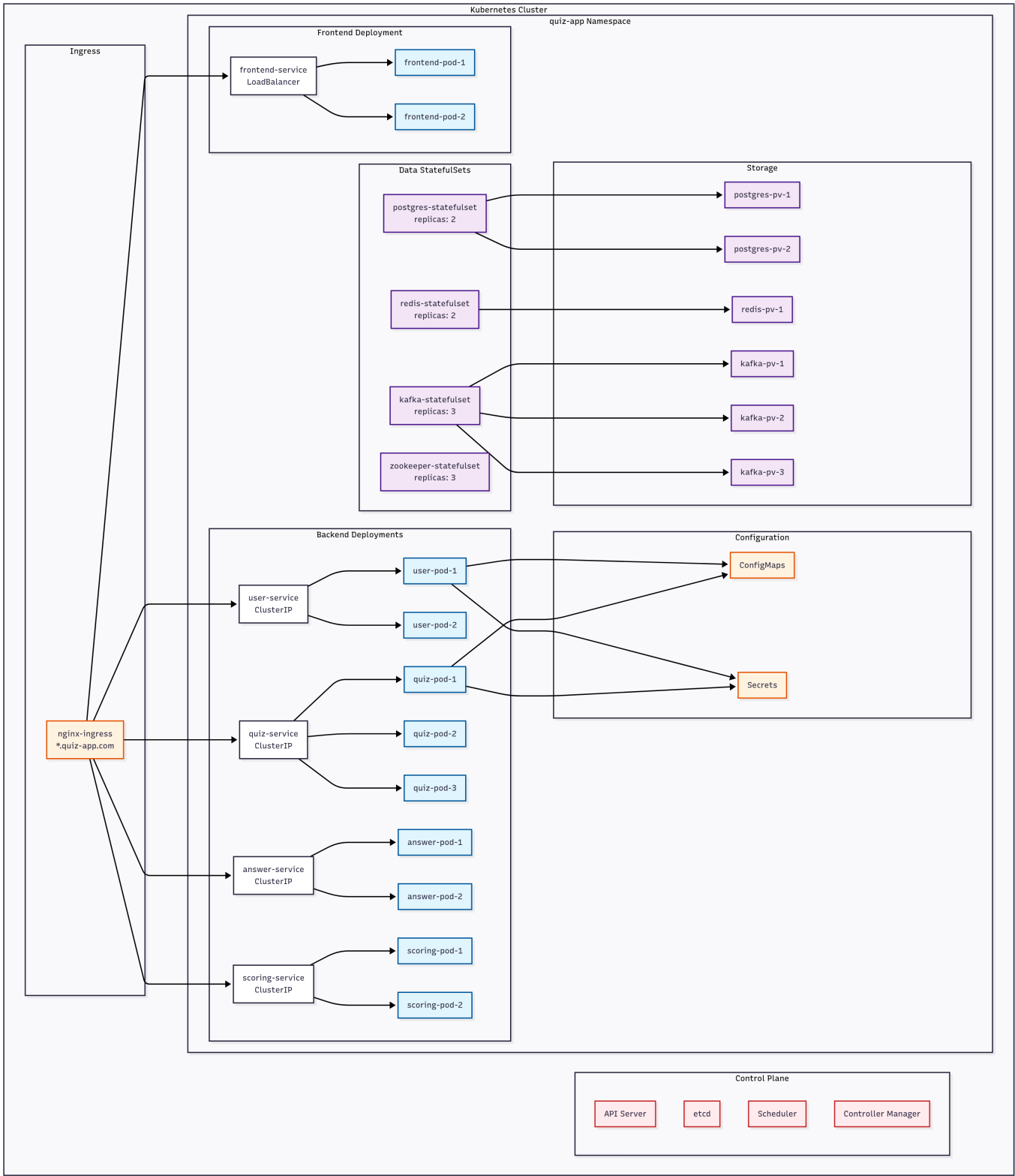
Production Deployment Architecture:
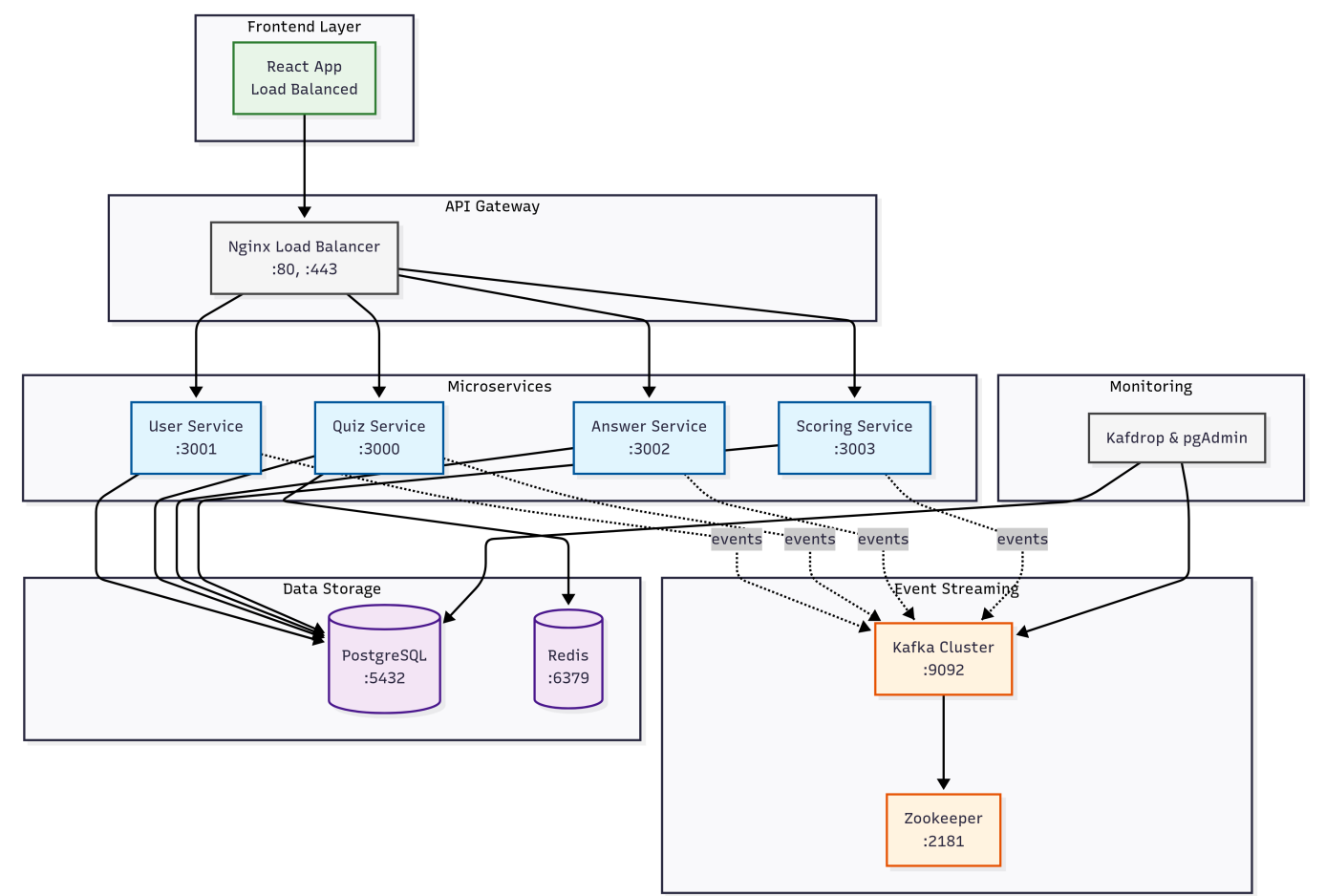
1.2 Container Deployment với Docker Swarm

Docker Swarm Stack Architecture:

1.3 Kubernetes Deployment Architecture

Kubernetes Cluster Layout:

2. Công cụ Triển khai

2.1 Container Orchestration Tools

A. Docker & Docker Compose

- **Docker Engine**: Container runtime
- **Docker Compose**: Multi-container management

- **Usage**: Local development, testing

B. Docker Swarm

- **Docker Swarm Mode**: Simple clustering
- **Benefits**: Easy setup, built-in load balancing
- **Usage**: Small to medium production deployments

C. Kubernetes

- **Kubernetes**: Enterprise orchestration
- **Helm**: Package management
- **kubectl**: CLI management
- **Usage**: Large scale production deployments

2.2 Infrastructure as Code

Terraform Example:

```
resource "aws_ecs_cluster" "quiz_app" {
  name = "quiz-app-cluster"
}

resource "aws_ecs_service" "quiz_service" {
  name          = "quiz-service"
  cluster       = aws_ecs_cluster.quiz_app.id
  desired_count = 3
}
```

2.3 CI/CD Pipeline

Basic GitLab CI:

```
stages:
  - build
  - deploy

build:
  script:
    - docker build -t quiz-service .
    - docker push quiz-service

deploy:
  script:
    - kubectl apply -f k8s/
```

3. Các bước Triển khai Hệ thống

3.1 Development Environment Setup

Quick Setup Commands:

```
# 1. Clone và setup
git clone https://github.com/your-org/quiz-app-event-driven.git
cd quiz-app-event-driven

# 2. Setup environment
cp .env.example .env

# 3. Start infrastructure
docker-compose up -d postgres redis kafka

# 4. Start services
docker-compose up -d

# 5. Verify
curl http://localhost:3000/health
```

Basic Docker Compose:

```yaml
# docker-compose.yml
version: '3.8'
services:
  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: quiz_app
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"

  kafka:
    image: confluentinc/cp-kafka:7.0.1
    ports:
      - "9092:9092"
    environment:
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181

  user-service:
    build: ./services/user-service
    ports:
      - "3001:3001"
```

```
    environment:
      DATABASE_URL: postgresql://postgres:password@postgres:5432/quiz_app

  quiz-service:
    build: ./services/quiz-service
    ports:
      - "3000:3000"
    environment:
      DATABASE_URL: postgresql://postgres:password@postgres:5432/quiz_app
      KAFKA_BROKERS: kafka:9092

  frontend:
    build: ./frontend
    ports:
      - "80:80"
```

3.2 Staging Environment Deployment

Docker Swarm Setup:

```
# 1. Initialize Swarm
docker swarm init

# 2. Deploy services
docker stack deploy -c docker-stack.yml quiz-app

# 3. Verify
docker service ls
```

Basic Stack Configuration:

```
# docker-stack.yml
version: '3.8'
services:
  quiz-service:
    image: quiz-app/quiz-service:latest
    environment:
      NODE_ENV: staging
      DATABASE_URL: postgresql://postgres:password@postgres:5432/quiz_app
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure

  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: quiz_app
      POSTGRES_USER: postgres
```

```
    POSTGRES_PASSWORD: password
  deploy:
    replicas: 1
```

3.3 Production Kubernetes Deployment

Basic Kubernetes Setup:

```
# 1. Create namespace
kubectl create namespace quiz-app

# 2. Deploy services
kubectl apply -f k8s/

# 3. Verify
kubectl get pods -n quiz-app
```

Basic Kubernetes Manifest:

```
# k8s/quiz-service.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: quiz-service
  namespace: quiz-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: quiz-service
  template:
    metadata:
      labels:
        app: quiz-service
    spec:
      containers:
      - name: quiz-service
        image: quiz-app/quiz-service:latest
        ports:
        - containerPort: 3000
        env:
        - name: DATABASE_URL
          value: "postgresql://postgres:password@postgres:5432/quiz_app"
        resources:
          requests:
            memory: "256Mi"
            cpu: "250m"
          limits:
            memory: "512Mi"
```

```
            cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
  name: quiz-service
  namespace: quiz-app
spec:
  selector:
    app: quiz-service
  ports:
  - port: 3000
    targetPort: 3000
```

Helm Deployment:

```
# 1. Create basic values
cat > values.yml << EOF
replicaCount: 3
image:
  repository: quiz-app/quiz-service
  tag: latest
EOF

# 2. Deploy
helm upgrade --install quiz-app ./chart \
  --values values.yml \
  --namespace quiz-app
```

3.4 Monitoring & Observability Setup

Basic Monitoring Setup:

```
# monitoring/prometheus.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
data:
  prometheus.yml: |
    global:
      scrape_interval: 15s
    scrape_configs:
    - job_name: 'quiz-services'
      static_configs:
      - targets: ['quiz-service:3000', 'user-service:3001']
```

Simple Health Checks:

```yaml
# monitoring/alerts.yml
groups:
- name: quiz-app-alerts
  rules:
  - alert: ServiceDown
    expr: up == 0
    for: 1m
    annotations:
      summary: "Service is down"

  - alert: HighErrorRate
    expr: rate(http_requests_total{status=~"5.."}[5m]) > 0.1
    for: 2m
    annotations:
      summary: "High error rate detected"
```

Kiến trúc triển khai Event-driven này đảm bảo **high availability**, **scalability**, và **maintainability** cho Quiz App system.