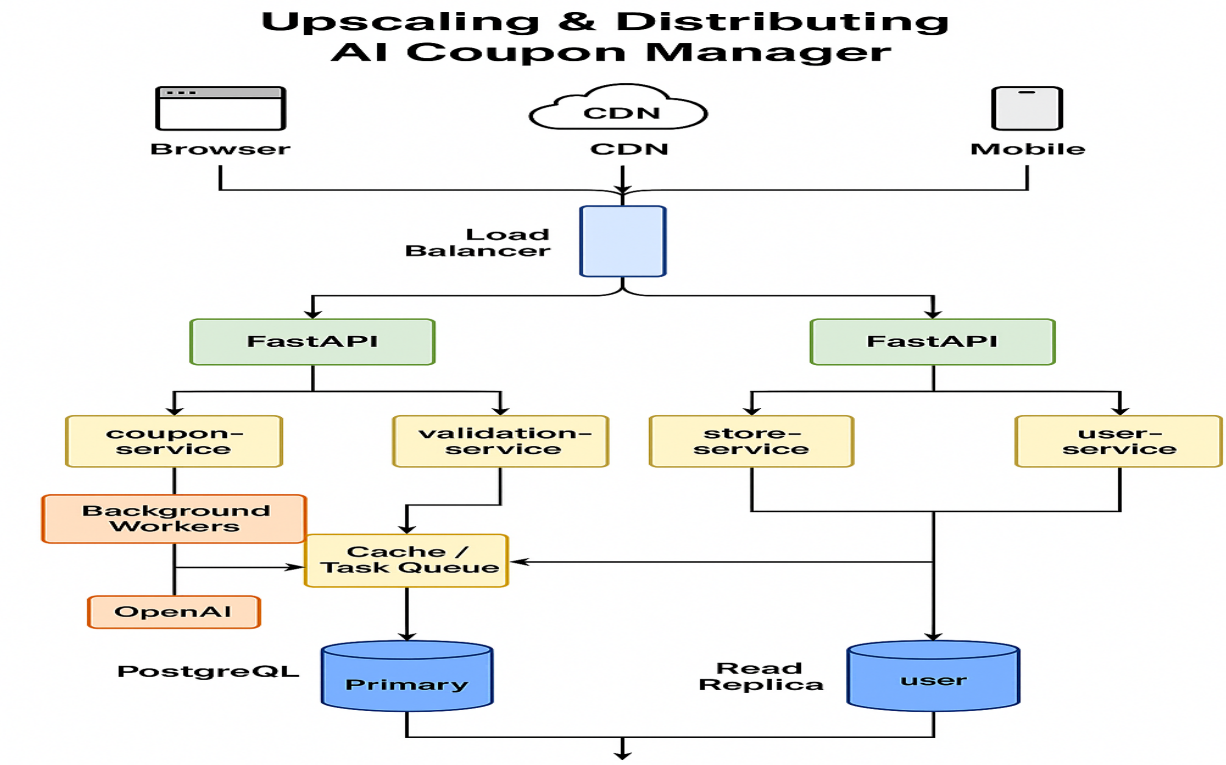
Upscaling & Distributing AI Coupon Manager



Component Explanations:

Load Balancer: Directs incoming traffic to available backend servers to prevent overload and ensure reliability.

API Gateway: Acts as the single entry point for client requests; handles routing, authentication, rate limiting, and logging.

Frontend (Next.js App): User interface that renders pages and interacts with backend APIs for a smooth experience.

Backend Services (FastAPI): Processes application logic like coupon submissions, AI-based validation, and data storage.

Redis Cache: Stores frequently accessed data temporarily to speed up responses and reduce load on databases.

PostgreSQL Database (Clustered): Stores all structured data with read replicas and failover for high availability.

OpenAI API / AI Microservice: Integrates AI capabilities for generating and validating coupons via LLMs.

Worker Queues (Celery + RabbitMQ): Handles background tasks like AI validation asynchronously to keep the app responsive.

CDN (Content Delivery Network): Distributes static assets globally for faster load times without hitting the origin server.

Monitoring & Logging: Collects metrics and logs from services to track health, debug issues, and send alerts.

Kubernetes / Docker Orchestration: Manages container deployments, scaling, and recovery automatically across servers.

Fault Tolerance Features:

Server crash: Handled by Load Balancer, Kubernetes - Health checks and auto-restart of failed containers.

API overload: Handled by API Gateway, Redis - Rate limiting and caching to manage spikes.

Slow AI API: Handled by Worker Queue, Circuit Breaker - Async processing and retries with backoff.

Database read failure: Handled by PostgreSQL Replication - Reads from healthy replicas.

Network latency: Handled by CDN, Redis - Edge caching for faster content delivery.

Service failure: Handled by Microservice Isolation - Contain failures within individual services.

Node crash: Handled by Kubernetes - Automatic rescheduling and redeployment of containers.

What is Kubernetes?

Kubernetes (K8s) is an orchestration system that manages containerized applications. It starts, stops, monitors, and scales containers automatically, ensuring high availability and resilience.

Key Kubernetes Actions:

• Auto-restarts crashed containers.

• Scales containers up or down based on load.

• Rolls out updates and can roll back on failure.

• Distributes containers across healthy nodes.

• Provides built-in monitoring and self-healing.

Kubernetes Configuration Files:

Deployment YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coupon-backend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: coupon-backend
  template:
    metadata:
      labels:
        app: coupon-backend
    spec:
      containers:
        - name: backend
          image: rohan/coupon-backend:latest
          ports:
            - containerPort: 8000
```

Service YAML:

```
apiVersion: v1
kind: Service
metadata:
  name: coupon-backend-service
spec:
  selector:
    app: coupon-backend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
  type: ClusterIP
```

Ingress YAML:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: coupon-ingress
spec:
  rules:
    - host: couponapp.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: coupon-backend-service
```

```
              port:
                number: 80
```

Horizontal Pod Autoscaler YAML:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: coupon-backend-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coupon-backend
  minReplicas: 2
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 60
```

Explanation of Configuration Files:

Deployment YAML explanations:

• apiVersion: apps/v1 - Kubernetes API version for deployments

• kind: Deployment - Specifies creation of a Deployment resource

• metadata.name - Name identifier for the deployment

• spec.replicas - Number of pod replicas to run

• spec.selector.matchLabels - Match pods with this label

• spec.template.metadata.labels - Labels assigned to pods

• spec.template.spec.containers - Container specs including image and port

Service YAML explanations:

• apiVersion: v1 - Core API group version

• kind: Service - Specifies creation of a Service

• metadata.name - Name identifier for the service

• spec.selector - Selects pods by label

• spec.ports.port - Exposed service port

• spec.ports.targetPort - Pod container port

• spec.type - Service exposure type (ClusterIP)

Ingress YAML explanations:

• apiVersion: networking.k8s.io/v1 - API group for Ingress

• kind: Ingress - Specifies creation of an Ingress resource

• metadata.name - Name identifier for the ingress

• spec.rules.host - Domain name for routing

• spec.rules.http.paths - Path routing configuration

• backend.service.name - Service to forward traffic

• backend.service.port.number - Service port to use

Horizontal Pod Autoscaler YAML explanations:

• apiVersion: autoscaling/v2 - API for autoscaling resources

• kind: HorizontalPodAutoscaler - Specifies HPA resource

• metadata.name - Name identifier for the HPA

• spec.scaleTargetRef - Target deployment to scale

• spec.minReplicas - Minimum number of pods

• spec.maxReplicas - Maximum number of pods

• spec.metrics - Metric type and target utilization