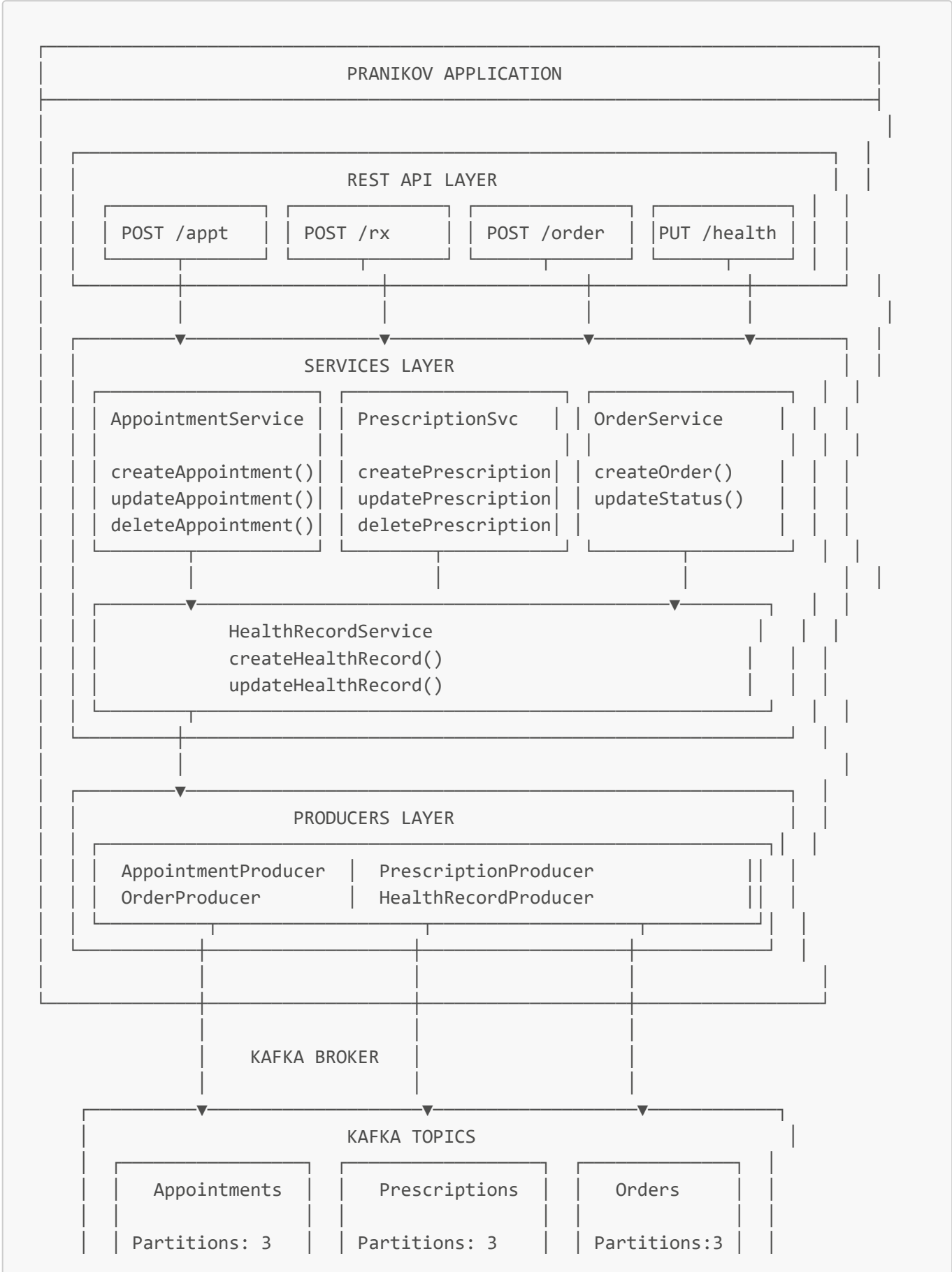
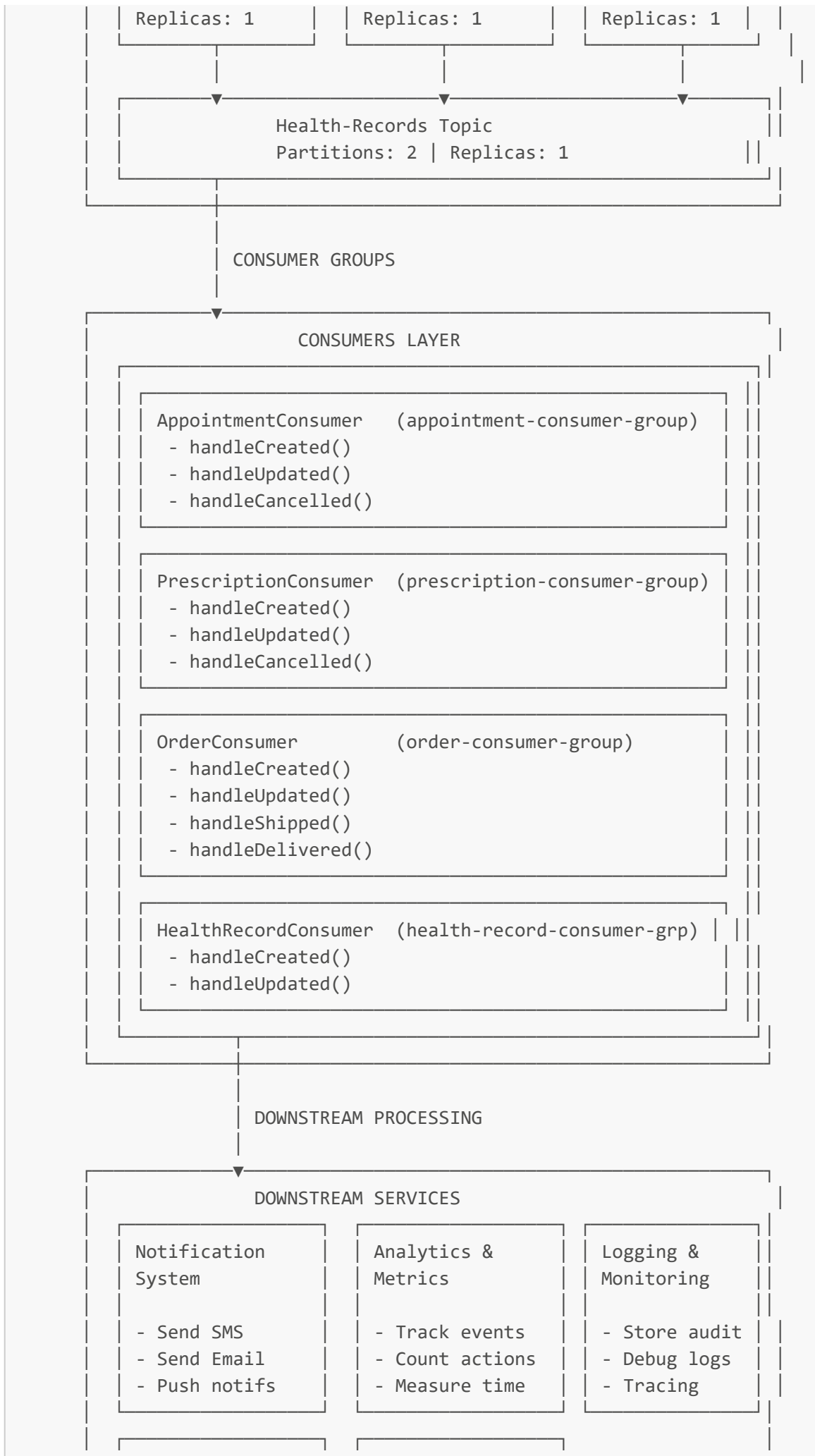
Kafka Integration - Visual Architecture Guide

System Architecture Diagram




Replicas: 1
Replicas: 1
Replicas: 1
Health-Records Topic
Partitions: 2 | Replicas: 1
CONSUMER GROUPS
CONSUMERS LAYER
AppointmentConsumer (appointment-consumer-group)
- handleCreated()
- handleUpdated()
- handleCancelled()
PrescriptionConsumer (prescription-consumer-group)
- handleCreated()
- handleUpdated()
- handleCancelled()
OrderConsumer (order-consumer-group)
- handleCreated()
- handleUpdated()
- handleShipped()
- handleDelivered()
HealthRecordConsumer (health-record-consumer-grp)
- handleCreated()
- handleUpdated()
DOWNSTREAM PROCESSING
DOWNSTREAM SERVICES
Notification System
- Send SMS
- Send Email
- Push notifs
Analytics & Metrics
- Track events
- Count actions
- Measure time
Logging & Monitoring
- Store audit
- Debug logs
- Tracing

```
    │   │ Archival &       │   │ Search & Index   │                  │
    │   │ Storage          │   │                  │                  │
    │   │                  │   │ - Elasticsearch  │                  │
    │   │ - Store records  │   │ - Lucene         │                  │
    │   │ - Backup data    │   │ - Solr           │                  │
    │   └──────────────────┘   └──────────────────┘                  │
    └────────────────────────────────────────────────────────────────┘
```

Event Flow Sequence Diagram

```
User          API           Service         Producer      Kafka Topic
Consumer
 │              │               │               │              │              │
 │─POST req────>│               │               │              │              │
 │              │─create()─────>│               │              │              │
 │              │               │─publishEvent()>│              │              │
 │              │               │               │─send msg─────>│              │
 │              │               │               │              │─process()───>│
 │              │               │               │              │              │
 │<─response────│<─return───────│               │              │              │
 │              │               │               │              │              │
                                                                │              │
                                          [Event stored in Kafka]              │
                                          [Available for replay]               │
                                                                │              │
                                                                │[Consumer
processes]
                                                                │   - Sends
notifications
                                                                │   - Updates
analytics
                                                                │   - Archives data
```

Topic Partitioning & Consumer Groups

```
APPOINTMENTS TOPIC (3 partitions)
├─ Partition 0 ──────────────────┐
│                                │  Consumer Group: appointment-consumer-group
├─ Partition 1 ──────────────────┼─> Consumer 1
│                                │
└─ Partition 2 ──────────────────┘

PRESCRIPTIONS TOPIC (3 partitions)
├─ Partition 0 ──────────────────┐
│                                │  Consumer Group: prescription-consumer-group
├─ Partition 1 ──────────────────┼─> Consumer 1
│                                │
└─ Partition 2 ──────────────────┘
```

```
ORDERS TOPIC (3 partitions)
├─ Partition 0 ──────────────────┐
│                                │  Consumer Group: order-consumer-group
├─ Partition 1 ──────────────────┼─> Consumer 1
│                                │
└─ Partition 2 ──────────────────┘

HEALTH-RECORDS TOPIC (2 partitions)
├─ Partition 0 ──────────────────┐
│                                │  Consumer Group: health-record-consumer-group
└─ Partition 1 ──────────────────┼─> Consumer 1
```

Event Lifecycle

```
┌──────────────────────────────────────────────────────────────────────┐
│                        EVENT LIFECYCLE                               │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│  1. EVENT CREATION                                                   │
│     └─> Service creates event object with action & data              │
│         AppointmentEvent(appointmentId, patientId, action="created")
│                                                                      │
│  2. EVENT PUBLISHING                                                 │
│     └─> Producer sends event to Kafka topic                          │
│         kafka-topic: "appointments"                                  │
│         message: JSON serialized event                               │
│                                                                      │
│  3. EVENT STORAGE                                                    │
│     └─> Kafka broker stores message in partition                     │
│         offset: auto-incremented                                     │
│         replication: distributed across brokers                      │
│                                                                      │
│  4. CONSUMER SUBSCRIPTION                                            │
│     └─> Consumer group reads from topic                              │
│         group.id: "appointment-consumer-group"                       │
│         auto.offset.reset: "earliest"                                │
│                                                                      │
│  5. MESSAGE PROCESSING                                               │
│     └─> Consumer deserializes and processes event                    │
│         action: "created" -> handleCreated()                         │
│         action: "updated" -> handleUpdated()                         │
│         action: "cancelled" -> handleCancelled()                     │
│                                                                      │
│  6. DOWNSTREAM ACTION                                                │
│     └─> Specific handler executes business logic                     │
│         - Send notifications                                         │
│         - Update cache                                               │
│         - Archive records                                            │
│         - Update metrics                                             │
│                                                                      │
```

```
│  7. OFFSET MANAGEMENT                                              │
│     └─> Consumer commits offset after processing                   │
│         offset committed: enables at-least-once processing         │
│         supports replay from any offset                            │
│                                                                    │
└────────────────────────────────────────────────────────────────────┘
```

Data Model - Event Structure

```
AppointmentEvent
├── appointmentId: String       # Unique identifier
├── patientId: String           # Reference to patient
├── doctorId: String            # Reference to doctor
├── date: LocalDate             # Appointment date
├── time: String                # Appointment time (HH:mm format)
├── reason: String              # Medical reason for appointment
├── status: String              # scheduled | confirmed | completed | cancelled
├── visitType: String           # in_person | virtual
└── action: String              # created | updated | cancelled

PrescriptionEvent
├── prescriptionId: String      # Unique identifier
├── patientId: String           # Reference to patient
├── doctorId: String            # Reference to doctor
├── medication: String          # Drug name
├── dosage: String              # Dosage amount
├── frequency: String           # How often to take
├── startDate: LocalDate        # When to start
├── endDate: LocalDate          # When to stop
├── status: String              # active | inactive | completed
└── action: String              # created | updated | cancelled

OrderEvent
├── orderId: String             # Unique identifier
├── patientId: String           # Reference to customer
├── pharmacyId: String          # Reference to pharmacy
├── totalPrice: Double          # Total order amount
├── status: String              # pending | processing | shipped | delivered
├── orderDate: LocalDateTime    # When ordered
├── deliveryDate: LocalDateTime # Expected delivery
├── items: List<OrderItemEvent> # Items in order
│   ├── productId: String
│   ├── quantity: Integer
│   └── price: Double
└── action: String              # created | updated | shipped | delivered

HealthRecordEvent
├── recordId: String            # Unique identifier
├── patientId: String           # Reference to patient
├── doctorId: String            # Reference to doctor
├── recordType: String          # lab_result | diagnosis | treatment
```

```
├── description: String         # Medical details
├── recordDate: LocalDate       # Date of record
└── action: String              # created | updated
```

Configuration Hierarchy

```
Application Configuration
│
├─ application.properties (Environment-specific)
│  ├─ spring.kafka.bootstrap-servers
│  ├─ spring.kafka.producer.*
│  ├─ spring.kafka.consumer.*
│  └─ kafka.topic.*
│
├─ KafkaConfig.java (Spring Configuration)
│  ├─ ProducerFactory (KafkaTemplate setup)
│  │  ├─ Serializers
│  │  ├─ Compression
│  │  ├─ Retries
│  │  └─ Acknowledgment
│  │
│  ├─ ConsumerFactory
│  │  ├─ Deserializers
│  │  ├─ Group ID
│  │  ├─ Auto offset reset
│  │  └─ Concurrency
│  │
│  ├─ Topic Beans
│  │  ├─ Partitions
│  │  ├─ Replication factor
│  │  └─ Retention policy
│  │
│  └─ KafkaAdmin (Cluster management)
│
└─ @KafkaListener Annotations (Consumer setup)
   ├─ Topic name
   ├─ Consumer group ID
   ├─ Concurrency settings
   └─ Error handlers
```

Processing Patterns

1. Simple Producer Pattern

```
Service → Producer → Topic → Done
```

2. Consumer Handler Pattern

```
Topic → Consumer → Handler (based on action)
                   ├─ handleCreated()
                   ├─ handleUpdated()
                   └─ handleDeleted()
```

3. Fan-Out Pattern (Multiple consumers per topic)

```
                  ┌─→ Notification Consumer
Topic ────────────┼─→ Analytics Consumer
                  ├─→ Logging Consumer
                  └─→ Archive Consumer
```

4. Event Sourcing Pattern

```
Create Event → Topic (immutable log) → Replay from offset
                                     → Reconstruct state
```

Error Handling Flow

```
Producer Error Flow
─────────────────────

Event → Send to Kafka → Failure
                        ├─ Retry 3 times
                        ├─ Wait 100ms between retries
                        └─ Log error & throw exception


Consumer Error Flow
─────────────────────

Message → Deserialize → Error
                        ├─ Log error
                        ├─ Send to DLQ (Dead Letter Queue)
                        └─ Continue with next message
```

Scaling Architecture

```
Current Setup (Single Instance)
──────────────────────────────

┌──────────────┐
│  Pranikov    │─┐
│  App (1)     │ │
└──────────────┘ │
                 ├─> Single Kafka Broker (localhost:9092)
```

```
                    │
                    │
  ┌──────────────┐  │
  │ Consumers    │──┘
  │ (embedded)   │
  └──────────────┘


Scaled Setup (Multiple Instances)
──────────────────────────────────

  ┌──────────────┐
  │  Pranikov    │─┐
  │  App (1)     │ │
  └──────────────┘ │
                   │
                   │
  ┌──────────────┐ │  ┌──────────────────────────────┐
  │  Pranikov    │─┼─>│   Kafka Cluster (3+ nodes)   │
  │  App (2)     │ │  │  - Replication factor: 3     │
  └──────────────┘ │  │  - Partitions: 9             │
                   │  │  - Load balanced             │
  ┌──────────────┐ │  └──────────────────────────────┘
  │  Pranikov    │─┘
  │  App (3)     │
  └──────────────┘

Each app instance independently:
- Publishes events
- Consumes from assigned partitions
- Maintains offset tracking
```

Last Updated: December 2024
Format: ASCII Diagrams
Use Case: Visual learning and documentation