**1. Common Infrastructure Requirements**

**1.1 Kafka Cluster**

* Use **Apache Kafka KRaft mode** (no Zookeeper).
* Minimum **3 Kafka broker nodes** for high availability.
* **Replication factor:** At least 2.
* **Partitions:** Based on load estimation per topic (details per service below).
* **Kubernetes Deployment:** Kafka cluster deployed via Helm or custom manifests.
* **Docker Images:** Use official lightweight Kafka images supporting KRaft.
* **Storage:** Persistent volumes attached to brokers (SSD recommended).

**1.2 Kubernetes Cluster**

* Minimum **3 worker nodes** (adjust node specs based on load, e.g., 8 vCPU, 32 GB RAM per node).
* Each microservice as a **separate Kubernetes deployment**.
* **Helm charts** to manage deployments.

**1.3 Database Servers**

* **PostgreSQL:** For transactional logging and FHIR database.

**2. Microservices Requirements**

**2.1 Logging API**

**Purpose:**

Receive various log types via REST API and push to Kafka.

**Tech:**

.NET 8/9 Web API

**Functions:**

* Expose endpoints like /api/logs/{type} (type = error, info, audit, etc.).
* Validate incoming log payloads.
* Push validated log entries to corresponding Kafka topics.

**Kafka Topics:**

* logs.error
* logs.info
* logs.audit
* (etc.)

**Other Requirements:**

* API Key Authentication for internal service calls.
* Throttling to prevent overload.
* Response with success or error status per log.

**2.2 Logging Consumer Service**

**Purpose:**

Consume logs from Kafka, batch write into Postgres, and fail-safe to files if needed.

**Tech:**

.NET 8/9 Worker Service

**Functions:**

* Batch read from Kafka (e.g., every 500 records or 30 seconds).
* Insert into appropriate Postgres tables (error\_logs, info\_logs, audit\_logs, etc.).
* In addition to Postgres, write to a flat file (split by file size if necessary, e.g., 100MB). Consider encryption.

**Resiliency:**

* Retry mechanism with backoff.
* Circuit breaker on database write failures.
* Health check endpoint for Kubernetes readiness/liveness probes.

**2.5 DICOM Download Zip Builder Service**

**Purpose:**

Build downloadable ZIPs based on download requests.

**Tech:**

.NET 8/9 Worker Service

**Functions:**

* Consume download requests containing patient/study info.
* Locate DICOM files in storage.
* Build ZIP archive (large ZIPs must be streamed or chunked).
* Store ZIP file in storage bucket (e.g., local, NFS, or S3 compatible).
* Generate download URL.
* Push download URL to Notification API (via Kafka).

**Kafka Topic Consumed:**

* download.requests

**Other Requirements:**

* Progress tracking (optional for very large studies).
* Retry if download/zipping fails.

**2.6 DICOM Puller Service**

**Purpose:**

Download DICOM files from remote WADO URLs for new studies.

**Tech:**

.NET 8/9 Worker Service

**Functions:**

* Consume study details from Kafka (study ID, WADO URL, etc.).
* Connect to WADO server and download all series and instances.
* Save DICOM files to local or cloud storage.

**Kafka Topic Consumed:**

* study.created

**Other Requirements:**

* Authentication support for WADO if needed.
* Resumable download capability (optional, if remote server supports).

**2.7 Notification API**

**Purpose:**

Receive and route notification requests to Kafka.

**Tech:**

.NET 8/9 Web API

**Functions:**

* REST endpoint /api/notifications/send.
* Accept notification type (email/sms/push) + payload.
* Validate payload format.
* Push to appropriate Kafka topic.

**Kafka Topics:**

* notifications.email
* notifications.sms
* notifications.push

**Security:**

* API Key Authentication.
* Input validation to prevent abuse.

**2.8 Notification Consumer Service**

**Purpose:**

Process notification requests and send via appropriate channel.

**Tech:**

.NET 8/9 Worker Service

**Functions:**

* Consume from Kafka.
* Process based on notification type (dummy):
  + Email: Send via SMTP or Email API (Sendgrid, etc.).
  + SMS: Send via SMS gateway API (Twilio, etc.).
  + Push: Send via push notification service (Firebase, APNs, etc.).

**Resiliency:**

* Retry queue if 3rd party fails (with delay and retry policies).

**3. Security Requirements**

* All internal service-to-service communication via mTLS (or at least authentication).
* API endpoints protected with API keys if needed.
* Sensitive logs (containing PII) must not be stored unencrypted.

**Summary Table of Services**

| **Service** | **API or Worker** | **Kafka Topic (Publish/Consume)** | **Database Used** | **Output** |
| --- | --- | --- | --- | --- |
| Logging API | API | Publish (logs.\*) | - | Kafka |
| Logging Consumer | Worker | Consume (logs.\*) | Postgres | Insert to Postgres / Failover to Files |
| Download Builder | Worker | Consume (download.requests) | Storage (zip files) | URL to Notification API |
| DICOM Puller | Worker | Consume (study.created) | Storage | DICOM Files |
| Notification API | API | Publish (notifications.\*) | - | Kafka |
| Notification Consumer | Worker | Consume (notifications.\*) | - | Sends Email/SMS/Push |