Hour 3 – Detailed Component Design & Technology Choices

> Goal: Provide concrete technology selections, detailed component designs, example schemas, API contracts, sequence diagrams (text), deployment patterns, and differential privacy mechanics to convert the Hour-2 architecture into an implementable blueprint.

---

0) Design Principles (recap)

- Keep the system **modular**: clear separation of ingress, processing, policy, storage, and governance.
- Prefer **battle-tested OSS** with cloud-native managed options for operational simplicity.
- Optimize for **observability, security, and deterministic latency**.

---

1) Technology Stack (recommended)

- **Ingress & Messaging**
- API Gateway: **Kong / Envoy / AWS API Gateway (regional)**
- Messaging backbone: **Apache Kafka (Confluent)** or **Redpanda** (single-binary, lower ops)
- Lightweight pub/sub for control plane: **NATS JetStream**

- **Compute & Orchestration**

- Kubernetes (EKS/GKE/AKS or on-prem k8s)
- Container runtime: containerd
- Service mesh: **Istio** or **Linkerd** for mTLS & telemetry

- **Storage**

- Object store: **S3 / MinIO** (for on-prem)
- Columnar lake: **Delta Lake / Apache Iceberg** on top of S3
- Metadata store: **Postgres** (timescaled for audit metadata if needed)
- Ledger: **Trillian / immudb** for append-only proofs
- Search: **OpenSearch** (for PHI-free indices)

- **Inference & ML**

- Model serving: **NVIDIA Triton**, **TorchServe** (fallback)
- Model training infra: **KubeFlow**, **MLflow** for registry
- FL coordinator: **Flower** / custom coordinator with gRPC

- DP libs: **TensorFlow Privacy**, **Opacus** (PyTorch), custom moments accountant

- **OCR/ASR/Imaging**

- OCR: **Tesseract** (baseline) + commercial options (Google/Vision or AWS Textract) or custom Vision-LM models
- ASR: **WhisperX**, **Kaldi** pipelines, or Whisper running on GPU for accuracy + diarization (pyannote, ECAPA)

- DICOM: **pydicom**, **Orthanc** for PACS integration

- **Policy & Authorization**

- Policy engine: **Open Policy Agent (OPA)** or **Cedar**

- AuthN/AuthZ: **Keycloak** / **Auth0** or cloud IAM

- **KMS/HSM & Crypto**

- Cloud KMS (AWS KMS, Google KMS) + HSM (Cloud HSM or Thales) for key escrow

- Envelope encryption libraries: **libsodium**, **Google Tink**

- **Observability**

- Metrics: **Prometheus**
- Tracing: **Jaeger** or **OpenTelemetry**
- Logs: **Loki** + Grafana

- Dashboarding: **Grafana** (SLO dashboards)

- **CI/CD & Security**

- GitHub Actions / GitLab CI
- Image scanning: **Clair**, **Trivy**
- SAST/DAST: **SonarQube**, **OWASP ZAP**

---

2) Component Designs & Interfaces

2.1 Ingestion API (Concrete Contract)

Endpoint: `POST /v1/documents` **Headers:** `Authorization: Bearer <JWT>`, `X-Tenant-Id`, `Content-Type: multipart/form-data` **Query params:** `use_case=[billing|research|clinical|analytics]`

Body (multipart) - `metadata` (JSON): `{ "document_id": "uuid", "source_system": "epic", "received_at": "2025-08-28T10:12:00Z", "format": "pdf" }` - `file` (binary)

Response (sync small docs) - `200 OK`
`{ "document_id":"uuid","status":"accepted","ingest_token":"abc123" }`

Semantic guarantees: idempotent ingest via `document_id` + `X-Tenant-Id`.

2.2 Redaction API (Sync with SLA)

Endpoint: `POST /v1/redact` **Body:**

```
{
  "document_id":"uuid",
  "tenant":"acme_hosp",
  "use_case":"research",
  "content":"base64...",
  "content_type":"text/plain"
}
```

Response: `{ "redacted_content":"base64...","pseudo_ids":{...},"audit_hash":"hex" }`

SLA: Small docs (<32KB) processed sync with p95<100ms.

2.3 Event Schema (Canonical Avro Example)

```
{
  "type":"record",
  "name":"ClinicalDocument",
  "namespace":"com.company.health",
  "fields":[
    {"name":"document_id","type":"string"},
    {"name":"tenant_id","type":"string"},
    {"name":"ingest_ts","type":{"type":"long","logicalType":"timestamp-
millis"}},
    {"name":"modality","type":"string"},
    {"name":"payload_ref","type":"string"},
    {"name":"pii_spans","type":["null",{"type":"array","items":{
      "type":"record","name":"PiiSpan","fields":[
        {"name":"type","type":"string"},
        {"name":"start","type":"int"},
        {"name":"end","type":"int"},
        {"name":"confidence","type":"float"}
```

```
      ]
    }}],"default":null}
  ]
}
```

---

3) Sequence Diagrams (Textual)

3.1 Ingest → OCR → NER → Redact (sync)

1. Client -> API Gateway: `POST /v1/documents`
2. API Gateway -> Ingest Service: validate, stamp tenant, push to Kafka `raw.documents`
3. OCR Worker consumes `raw.documents`, emits `normalized.documents`
4. NER Worker consumes `normalized.documents`, emits `pii_found`
5. Policy Engine (sync call) returns redaction recipe
6. Redactor applies FPE / pseudonymization → writes `redacted.documents`
7. Audit Service hashes input+policy+output → appends to ledger
8. Ingest Service returns sync `200` with `audit_hash`

3.2 Audio Streaming (Realtime)

1. Client -> WebSocket/gRPC Stream opens
2. Chunk producer -> Stream Ingest service (Kafka streaming or NATS)
3. ASR DAG processes frames, emits partial transcripts
4. Diarization attaches speaker IDs; NER marks PHI spans with timestamps
5. Redactor masks live transcript and emits sanitized stream to subscriber
6. Audit events buffered & anchored periodically

---

4) Pseudonymization Design

- **Stable Pseudo-IDs:** `pseudo_id = HMAC_SHA256(salt || canonical_entity_key)` where `salt` = per-tenant secret stored in KMS and `canonical_entity_key` = normalized name+DOB+hash(MPI_hint).
- **Entity Graph:** Graph DB (Neo4j / DGraph) to maintain relationships (family links) and cross-reference identities; mapping stored as entity nodes with deterministic `pseudo_id`.
- **Reversal policy:** De-pseudonymization requires multi-party approval; keys accessed only via HSM with audit gating.

---

5) Differential Privacy: Operational Mechanics

- **Per-request DP Mode:** API caller may request DP sanitation with `dp_profile` header. The policy engine reserves ε from tenant budget.

- **Mechanisms:** DP-SGD for model releases; Laplace/Gaussian mechanisms for numeric aggregates; randomized response for categorical where applicable.
- **Accounting:** Moments accountant implementation (reference: Abadi et al. 2016) + privacy ledger storing (query_id, ε_spent, composition).
- **Example:** Suppose research query asks for aggregated counts. System computes noisy count: `noisy = true_count + N(0, σ^2)`, where σ computed from ε via Gaussian mechanism. The API returns `noisy` + `epsilon_used`.

---

6) Security: Key Flows & Escrow

- Keys per tenant + per purpose (ingest, pseudonymization, escrow).
- HSM for root keys; KMS envelopes for per-tenant salts.
- Decryption/de-pseudonymization flows must be signed and multi-party approved (e.g., 2-of-3 approvers) with time-boxed session tokens.

---

7) Deployment Patterns & Sizing (Kubernetes)

- **Namespace per tenant** for control-plane separation (not data). Use network policies to isolate.
- **Pod types**: stateless workers (scale HPA), GPU pools (node pools) for OCR/ASR, stateful sets for Kafka & DBs (or managed services).
- **Autoscaling triggers**: Kafka lag, CPU/GPU utilization, queue depth.
- **Canary & Blue/Green** for policy bundles and models.

---

8) Observability: Concrete Metrics & Traces

- **Ingress:** requests_total, requests_latency_ms_bucket, auth_failures
- **Processing:** messages_consumed, processing_latency_ms, detector_recall_estimate, false_negative_rate_est
- **DP:** epsilon_spent_total, budgets_remaining
- **Audit:** ledger_write_latency, ledger_anchor_latency
- **Tracing:** request_id propagated (x-request-id) across services; spans for OCR, NER, policy_evaluate, redaction

---

9) Example: Near Miss Report (JSON)

```
{
  "document_id":"uuid",
  "detector_id":"ner.v2.1",
  "expected_entity":"SSN",
  "found":false,
```

```
  "context":"Patient social security number appears in scanned image at 0x...",
  "confidence_scores":[],
  "ingest_ts":"2025-08-28T10:12:00Z",
  "sample_for_labeling":"s3://bucket/sample/uuid.png"
}
```

---

10) Testing & Validation Plan

- Unit tests for parsers/policy bundles.
- Integration tests with synthetic PHI corpora (prescribed redaction expectations).
- Perf tests: k6 + custom Kafka load harness to validate 100k docs/s and p95 latency.
- Security: periodic pen-tests, automated SCA, CI gating for policy changes.

---

11) Implementation Roadmap (90 Days)

- **Week 0–2:** Bootstrapping infra: k8s, Kafka, API gateway, basic ingest API, schema registry.
- **Week 3–6:** Text/PDF path end-to-end: OCR, NER Tier-1, redaction, audit ledger v1, dashboards.
- **Week 7–9:** Audio path, streaming ASR & diarization, live redaction.
- **Week 10–12:** DICOM & structured data connectors (HL7/FHIR), entity graph & pseudonymization.
- **Weeks 13–16:** DP accountant, FL coordinator, adversarial robustness training, compliance validations.

---

12) Outstanding Decisions for Architect Sign-Off

1. On-prem vs SaaS defaults for top 10 customers.
2. Choice between Confluent vs Redpanda based on ops footprint and cost.
3. Level of commercial OCR/ASR to license vs build in-house.
4. Default ε/δ profiles to ship as safe presets.

---

13) Deliverables (next steps)

- Sequence diagrams in PlantUML for key flows.
- Terraform modules for infra bootstrap (k8s, kafka, object store).
- Prototype: a containerized pipeline that ingests PDF → OCR → Tier-1 NER → redacts → audit record.

End of Hour 3 deliverable.