Upstream Live – End-to-End Technical Architecture

Goal: Show exactly how data moves from wells/pipelines (SCADA & sensors) to a secure cloud backend and into a real-time Python dashboard with alerting, analytics, and audit trails.

---

1) High-Level Flow (10,000-ft view)

```
[Field Assets]
  ├─ Wells, Pipelines, Compressors, Flow Stations, Terminals
  └─ Existing SCADA/DCS + New IoT Sensors
         │
         ▼
[Edge/Data Collection Layer]
  ├─ OPC UA Client / Modbus → Edge Gateway
  ├─ IoT Sensor Hubs (LoRaWAN/4G/5G/Sat)
  └─ Secure Data Broker (MQTT) on site
         │  TLS/mTLS + Signing
         ▼
[Secure Ingestion]
  ├─ API Gateway (REST/WebSocket)
  ├─ MQTT Broker (Cloud)
  └─ SFTP/Batch (NPMS-style CSV/XLS)
         │
         ▼
[Streaming & Processing]
  ├─ Kafka (topics: readings, events, anomalies)
  ├─ Stream processors (Python/Faust or Flink/Spark)
  └─ Anomaly/Rules Engine (scikit-learn + rule DSL)
         │
         ▼
[Storage]
  ├─ Time-Series DB (TimescaleDB/InfluxDB)
  ├─ Relational DB (PostgreSQL) – assets, users, licenses
  ├─ Object Store (raw files, satellite rasters, NPMS dumps)
  └─ Search (OpenSearch/Elasticsearch) – logs/metadata
         │
         ▼
[Services & APIs]
  ├─ FastAPI microservices (asset, telemetry, alerts, reports)
  ├─ Auth (Keycloak/OAuth2 + RBAC)
  └─ Celery/Workers (ETL, notifications)
         │
```

```
          ▼
[Visualization & Access]
  ├─ Python Dash/Streamlit (internal ops dashboard)
  ├─ React Web (exec portal + maps)
  └─ Public Transparency (aggregated, non-sensitive)
```

---

2) Field/Edge Integration (SCADA + IoT)

- **Existing SCADA/DCS:**
- Protocols: **OPC UA**, Modbus TCP, proprietary drivers from vendors (Schneider/Emerson/Honeywell).
- Strategy: Deploy **Edge Gateway** (industrial PC) that runs connectors to read tags/points and **publishes to MQTT** with TLS/mTLS.
- Sampling: 1–60s cadence; edge buffers when links drop.
- **New IoT Sensors** (for sites without SCADA):
- Flow, pressure, temperature, vibration, leak detection.
- Connectivity: **LoRaWAN** (rural), **4G/5G**, or **satellite** (offshore).
- Edge firmware publishes MQTT messages with site/well IDs.
- **Data Minimization at Edge:**
- Downsample + compress; attach **SHA-256 signatures** and **device cert IDs**.
- Local **rule checks** (e.g., shut-in alert) for faster safety responses.

---

3) Secure Ingestion & Identity

- **API Gateway**: Single front door for REST/WebSocket ingestion (`/v1/telemetry`), rate limits, WAF, mTLS from trusted sites.
- **MQTT Broker (cloud)**: Topics by operator/asset: `ops/{operator}/{assetId}/readings`.
- **Batch/SFTP**: Accept daily **NPMS-style CSV/XLS** when real-time isn't available.
- **AuthZ/AuthN**: OAuth2/OIDC (Keycloak/Cognito/Entra ID). **RBAC** roles: `regulator-admin`, `analyst`, `operator`, `public`.
- **PIA compliance & Audit**: Every write is **signed & timestamped**; immutable logs in **WORM storage** (e.g., S3 Object Lock).

---

4) Streaming, Cleansing, and Analytics

- **Message Bus:** Apache **Kafka** topics
- `telemetry.raw` → ingest from MQTT/API
- `telemetry.clean` → unit normalization (SPE units), schema check (Avro/JSON Schema)
- `events.alerts` → rule & ML-triggered alerts
- `billing.revenue` → computed royalties/variances
- **Stream Processing (Python-first):**
- **Faust** (Python stream processing) or **Flink/Spark Structured Streaming**.
- Enrich with asset metadata (operator, field, license, geo).
- **Data Quality**: null checks, drift detection, outlier clipping.
- **Anomaly & Rules Engine:**
- **Rules DSL** (YAML): thresholds, rate-of-change, stuck sensor detection.
- **ML models** (scikit-learn/PyTorch): isolation forest, ARIMA/LSTM for forecast vs actual.
- Output → `events.alerts` with severity (P1–P3).

---

5) Storage Architecture

- **Time-Series DB**: **TimescaleDB** (PostgreSQL extension) or **InfluxDB**
- Writes: per-second/minute readings keyed by `asset_id, metric, ts`.
- Retention policies + downsampling (e.g., raw → 1-min → 15-min → hourly).
- **Relational DB (PostgreSQL)**
- Tables: `assets`, `wells`, `pipelines`, `facilities`, `operators`, `licenses`, `users`, `roles`, `events`, `work_orders`.
- **Object Storage**
- Buckets: `raw/npms/`, `raw/satellite/`, `exports/`, `reports/`.
- Keep **parquet** versions for analytics; lifecycle policies.

- **Search/Logs**: OpenSearch/Elasticsearch for log analytics and free-text search over incidents.

---

6) Data Model (Core Tables)

assets

- `asset_id (uuid)` | `asset_type (well/pipeline/facility)` | `name` | `operator_id` | `license_id` | `lat` | `lon` | `status`

readings (Timescale hypertable)

- `reading_id (uuid)` | `asset_id` | `metric (oil_bopd, gas_mmscfd, pressure_psi, temp_c, flow_bph, status)` | `value (double)` | `ts (timestamptz)` | `quality_flag`

events

- `event_id` | `asset_id` | `type (anomaly, downtime, spill_suspect)` | `severity` | `details` | `ts_start` | `ts_end` | `ack_by`

royalty_ledger

- `ledger_id` | `operator_id` | `period` | `declared_volume` | `computed_royalty` | `paid_amount` | `variance`

---

7) Public/Partner APIs (FastAPI Design)

Auth: OAuth2 Authorization Code with PKCE; JWT with roles.

Sample Endpoints

- `GET  /v1/assets?type=well&operator=...`
- `GET  /v1/readings?asset_id=...&metric=oil_bopd&from=...&to=...`
- `GET  /v1/events?severity=P1&since=24h`
- `POST /v1/ingest` *(signed JSON lines – for operators without MQTT)*
- `WS   /v1/stream/readings` *(live push to dashboards)*

Example Reading Payload

```
{
  "asset_id": "9c7a…",
  "metric": "pressure_psi",
  "value": 1540.3,
  "ts": "2025-08-14T18:25:43Z",
  "quality_flag": "GOOD",
  "signature": "base64-edsig…"
}
```

8) Dashboards (Python-First)

- **Internal Ops (Streamlit/Plotly Dash):**
- **Global Overview:** total production, active wells, P1 alerts, last update time.
- **Live Map (folium/leaflet):** wells/pipelines with status chips; click-through to asset panel.
- **Asset Panel:** sparkline trends, pressure/flow gauges, recent events, predicted next-24h output.
- **Compliance/Revenue Tab:** declared vs computed volumes, royalty variance heatmap.
- **Executive Web (React + FastAPI):**
- KPI cards, trend charts (Recharts), export to PDF.
- **Public Transparency (optional):**
- Aggregated, non-sensitive stats (monthly by basin/state).

9) Alerts & Notifications

- **Trigger Sources:** rules engine + ML.
- **Channels:** email/SMS (Gov't SMS gateway), Microsoft Teams/Slack webhooks.
- **De-duplication:** group repeated alerts; escalation logic (P3→P2→P1).
- **Runbooks:** each alert type links to SOPs; acknowledgment and resolution timestamps for audit.

10) Security, Compliance & Governance

- **Transport security:** TLS 1.2+; **mutual TLS** for field → cloud.
- **Device identity:** X.509 certs, short-lived tokens from IoT Core.
- **RBAC/ABAC:** role-based + attribute-based access (operator can only see own assets).
- **PIA & Data Residency:** host in **Nigeria-region cloud** or gov DC; VPC peering with NUPRC.
- **Audit:** append-only logs; SIEM forwarding; monthly key rotation.
- **Backup/DR:** PITR for DB, cross-region object replication, RPO ≤ 15 min, RTO ≤ 2 hrs.

11) Observability & SRE

- **Metrics:** Prometheus scraping from services; Grafana dashboards.
- **Logs:** structured JSON to OpenSearch; correlation IDs per request.
- **Tracing:** OpenTelemetry → Jaeger/Tempo.
- **SLOs:** API p99 latency < 500 ms; data freshness targets (≤15 min for NPMS/batch, ≤60 s for live).

---

12) Vendor-Agnostic & Cloud Options

- **AWS:** IoT Core, MSK (Kafka), RDS (Postgres/Timescale), OpenSearch, S3, Lambda, SageMaker optional.
- **Azure:** IoT Hub, Event Hubs, Azure Database for PostgreSQL, Data Lake, Monitor, ML.
- **GCP:** IoT (partner), Pub/Sub, Cloud SQL Postgres, GCS, Vertex AI, Operations Suite.
- **On-Prem Hybrid:** K3s/Kubernetes + EMQX (MQTT) + Kafka + Postgres.

---

13) Phased Delivery Plan (Pragmatic)

Phase 1 (0–3 months):

- Ingest NPMS CSV/Excel; build Timescale schema; Streamlit MVP: overview + trends + map.
- Simulate live feed using cron (5–15 min) and WebSocket push.

Phase 2 (3–9 months):

- Add operator API/MQTT integrations where available; enable alerts + anomaly detection.
- Bring in revenue cross-check and variance reporting.

Phase 3 (9–18 months):

- Edge gateways to non-SCADA sites; satellite/environmental overlays; predictive models.
- Executive portal + public transparency module.

---

14) Minimal Viable Data Contracts (Schemas)

Telemetry JSON Schema (Avro/JSON Schema)

```
{
  "$id": "https://nuprc.gov.ng/schemas/telemetry.json",
  "type": "object",
  "required": ["asset_id", "metric", "value", "ts"],
  "properties": {
    "asset_id": {"type": "string", "format": "uuid"},
    "metric": {"type": "string", "enum":
["oil_bopd","gas_mmscfd","water_bwpd","pressure_psi","temp_c","flow_bph","status"]},
    "value": {"type": "number"},
    "ts": {"type": "string", "format": "date-time"},
    "quality_flag": {"type": "string", "enum": ["GOOD","SUSPECT","BAD"]},
    "signature": {"type": "string"}
  }
}
```

Alert Event

```
{
  "event_id": "uuid",
  "asset_id": "uuid",
  "type": "anomaly|downtime|spill_suspect|maintenance",
  "severity": "P1|P2|P3",
  "rule_id": "string",
  "message": "text",
  "ts_start": "date-time",
  "ts_end": null,
  "evidence": {"zscore": 3.1, "expected": 1000, "observed": 700}
}
```

---

15) Reference Python Components

- **Ingestion (FastAPI)**: `/v1/ingest` endpoint with HMAC/mTLS, writes to Kafka.
- **Stream Worker (Faust)**: subscribes to `telemetry.raw`, normalizes units, enriches, emits anomalies.
- **Scheduler (Celery)**: pulls NPMS files daily; backfills gaps; regenerates aggregates.
- **Dashboard (Streamlit/Dash)**: subscribes to WebSocket stream; shows KPIs, map (folium), alerts feed.

---

16) Risks & Mitigations

- **Operator data sharing delays** → Start with NPMS + simulated feeds; demo value early.
- **Network unreliability** → Edge buffering + idempotent writes + at-least-once semantics.
- **Data sensitivity** → Strict RBAC, data masking, per-operator tenants.
- **Model drift** → Weekly retraining schedule; shadow evaluation; human-in-the-loop.

---

17) Quick Win Demo Plan (for leadership buy-in)

1. 7-day sample from NPMS or synthetic CSV.
2. Stand up TimescaleDB + Streamlit dashboard (overview + map + alerts).
3. Simulate a pressure drop event → live P1 alert → email/Teams notification.
4. Show royalty variance mockup (declared vs computed).

Outcome: A tangible, clic