YourHealthNS – Solution Architecture Proposal

Prepared by: Suresh Kumar Balasubramanian

Nova Scotia Health

Version 1.1

November 2025

Table of Contents

Contents

1. Executive Summary

YourHealthNS is a secure, citizen-facing healthcare platform designed to modernize digital access for Nova Scotia residents. It encompasses mobile (iOS/Android), responsive web, on-premises OpenShift-hosted backend microservices, and integration with existing provincial systems. The design prioritizes privacy-first principles, zero-trust security, and compliance with healthcare standards such as FHIR, HL7, and DICOM.

Key goals: (1) Privacy & compliance over feature speed, (2) Standards over custom integrations, (3) High observability, (4) Automated DevSecOps with human oversight, (5) Cost transparency and operational resilience.

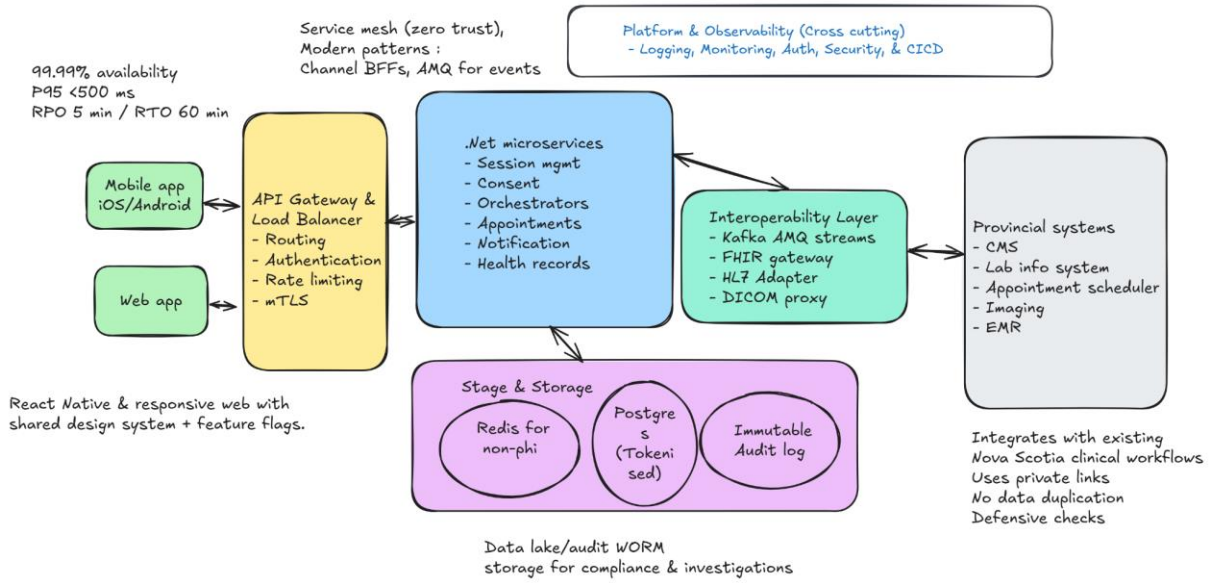

Figure 1 — High-level platform overview. Editable source Open link

2. Requirements & Constraints Mapping

The table below maps assessment requirements to corresponding architectural design responses. Detailed rationale and trade-off analyses for the architectural responses listed in this section are documented in the *YourHealthNS Architecture Decision Record (ADR) Register*.

Requirement	Architectural Response	Notes
Mobile apps deployable to App/Play Stores	React Native-based apps using a shared TypeScript design system and feature flags.	PKCE + OIDC flows; compliant with Apple/Google policies.
Responsive web application	Next.js web portal served via CDN/WAF; shared design system.	WCAG 2.1 AA accessibility level.
Backend on provincial OpenShift	.NET 8 microservices managed by GitOps (FluxCD)	.NET 8 was chosen for its maturity, enterprise support,

	and Jenkins pipelines.	built-in OIDC and gRPC libraries, and native containerization support in OpenShift. It aligns predictable performance and long-term support (LTS)
Centralized routing, security, and traffic control	API Gateway integrated with Service Mesh (Istio) for mTLS, rate limiting, retries, and observability	Zero-Trust enforcement, uniform telemetry, and seamless traffic routing
Minimal PHI storage	Tokenized consent store; Redis for non-PHI sessions; immutable WORM audit logs.	Data fetched on demand from source systems.
Healthcare standards (FHIR/HL7/DICOM)	FHIR gateway, HL7 v2 adapter, and DICOMweb proxy for interoperability.	Schema validation and audit logging.
Security & compliance (PHIPA/PIPEDA)	Zero-trust mesh (mTLS, OPA), Vault+HSM, CI/CD gates, immutable audit store.	Quarterly access reviews and 7-year retention.
Core citizen features	Appointments, records, and notifications via orchestrated .NET microservices and Kafka events.	Async design improves reliability.
Scalability & availability	HPA scaling, circuit breakers, and warm DR setup.	99.99% uptime, RPO ≤ 5 min, RTO ≤ 60 min.

3. Scope

In Scope

- iOS/Android mobile apps and responsive web portal
- Backend services, API gateway, observability stack, and CI/CD pipelines
- Provincial OIDC integration, consent management, and audit retention
- Healthcare standards interoperability (FHIR, HL7, DICOM)

Out of Scope

- Cross-province federation and multi-region active-active DR
- AI/ML-based clinical insights (future phase)
- Service-level and database schema design
- Detailed UI design and front-end theming will be handled by the product team outside this proposal's scope

4. Capacity & Performance Targets

Supports 350 K monthly users and ~1 M API calls per day, handling peaks of 1 K RPS (≈ 3–5 K concurrent sessions, 10 K burst).
Kafka processes ~1.2 K messages/s; PostgreSQL ~1 K reads/s; Redis ~600 ops/s.
Meets SLOs of P95 < 500 ms, 99.99 % availability, RPO ≤ 5 min, RTO ≤ 60 min.

Note - PostgreSQL is higher since it handles transactional, tokenized data, while Redis caches short-lived session and token metadata

5. Layered Architecture Overview

The following subsections present a layered view of the system from user-facing experiences to backend deployment layers.

5.1 Experience & Edge Access

Describes entry points via mobile/web apps with BFFs, API gateway, and mTLS authentication.

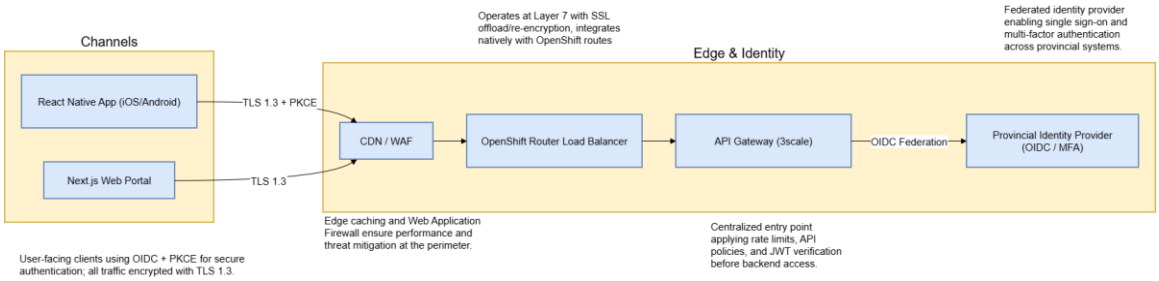

Figure 2 — Experience and edge layer. *Editable Source:* Open Draw.io File

5.2 Core Services & Interoperability

Shows orchestration across .NET microservices and FHIR/HL7 adapters.

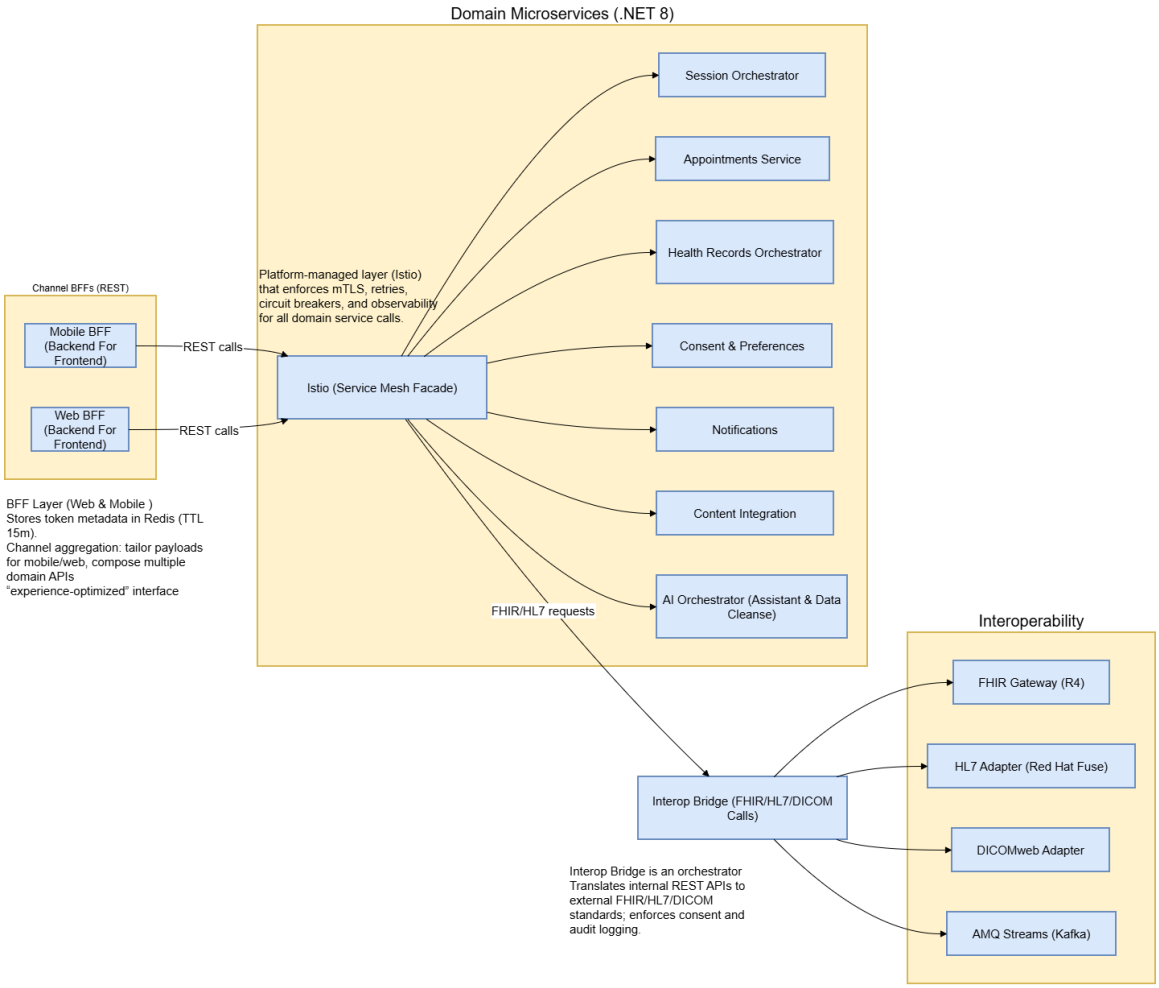

Figure 3 — Core services and interoperability. *Editable Source:* Open [Draw.io File]

5.3 Component & Container View

Logical breakdown of service groupings and containerized deployments.

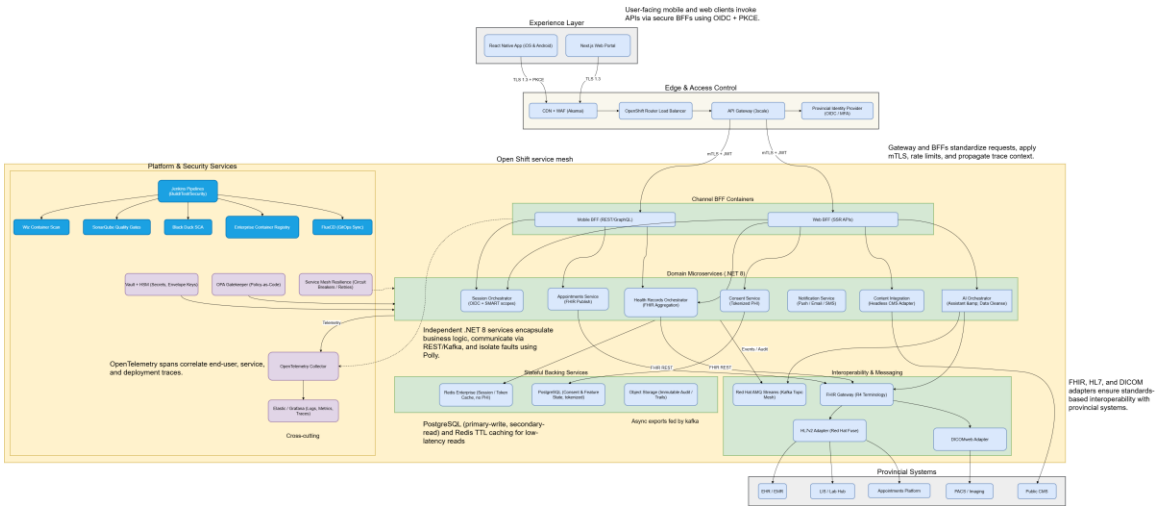

Figure 4 — Component–Container view. *Editable Source:* Open [Draw.io File]

See also: Section 5.5 for corresponding physical topology.

The component and container view describes the logical organization of all backend services and how they interact in a modular, scalable way.

- Follows .NET 8 Clean Architecture with clear separation between domain, application, and infrastructure layers.
- Services are deployed as stateless containers on OpenShift, horizontally scaled via Kubernetes HPA.
- Circuit breakers, retries, and fallback mechanisms are implemented using Polly and Istio Service Mesh.
- OpenTelemetry instrumentation provides full observability for tracing and metrics across containers.
- Integrated observability stack exports metrics and alerts to Opsgenie, ensuring real-time incident response and escalation.
- API Gateway, Domain Services, BFFs, and Shared Utilities form the four core groupings of this layer.
- Visually represents how each business capability (appointments, notifications, consent) operates independently.
- Supports faster development and deployment cycles by isolating functions into manageable microservices.
- Improves reliability if one service fails, others continue to operate. Easier to maintain.
- Depicts data and event flow between provincial systems via Kafka and FHIR APIs.

5.4 Integration Architecture

The integration layer serves as a standardized gateway between provincial systems such as EHR, LIS, booking, and imaging services, and the YourHealthNS application.

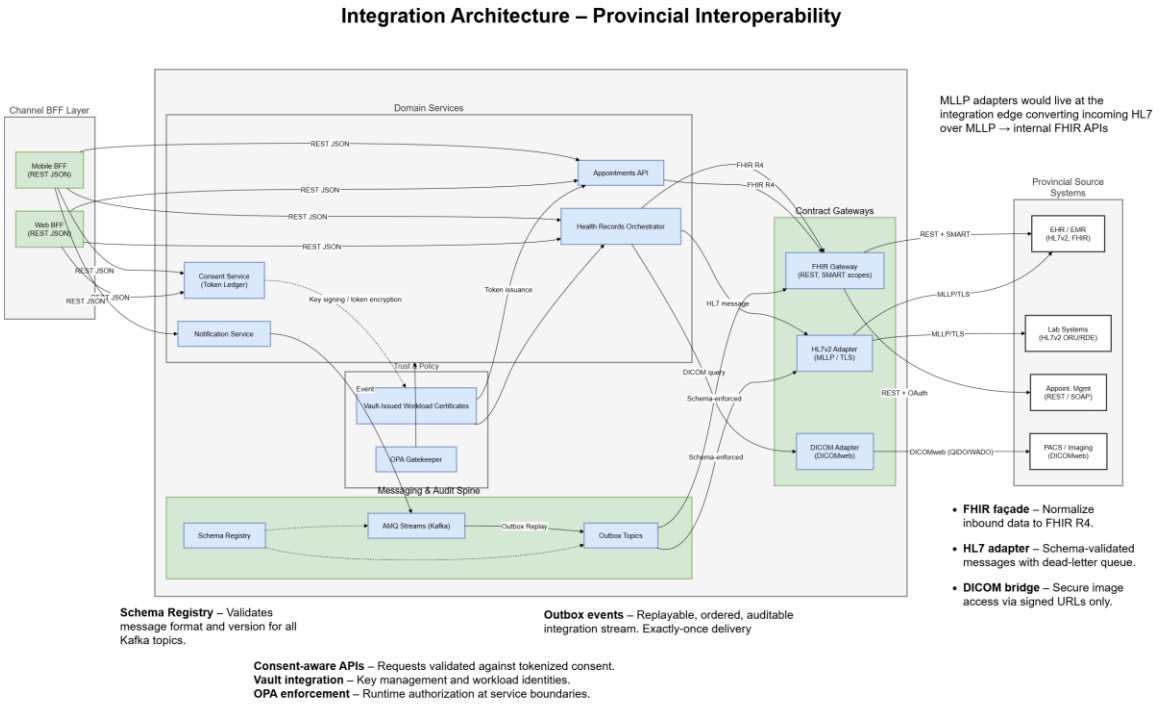

Figure 5 — Integration architecture overview. *Editable Source:* Open Draw.io File

- Allows different healthcare systems to share information in a consistent and secure manner. Reduces integration complexity by avoiding fragile point-to-point connections.
- Ensures data flows are auditable, compliant, and easily extendable for future systems.
- Enables faster onboarding of new provincial services with minimal disruption.
- Incorporates defensive validations, deduplication, and schema conformance checks to ensure reliable and idempotent data exchange.
- Invalid or duplicate messages are routed to DLQ for inspection no silent failures.
- Define strict API contact specs with external components.

5.5 Physical Deployment Topology

Illustrates the OpenShift deployment zones and cross-cluster communication between services.

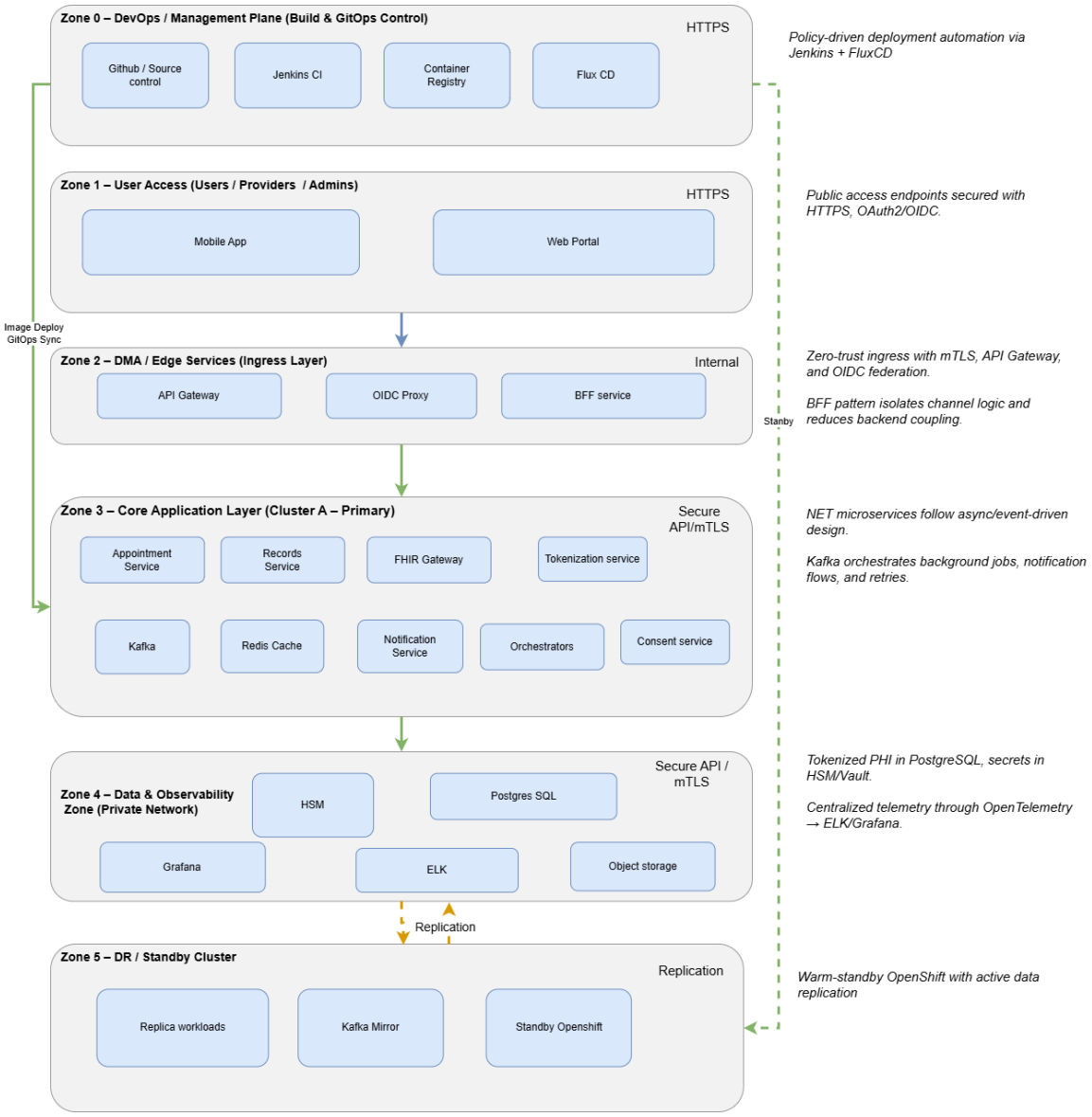

Figure 6 — Physical deployment topology. *Editable Source:* Open Draw.io File

Refer to Section 5.3 for logical component relationships.

6. Network Topology & Security Zones

Describes user zones, DMZ, service mesh, and core system boundaries ensuring zero-trust compliance.

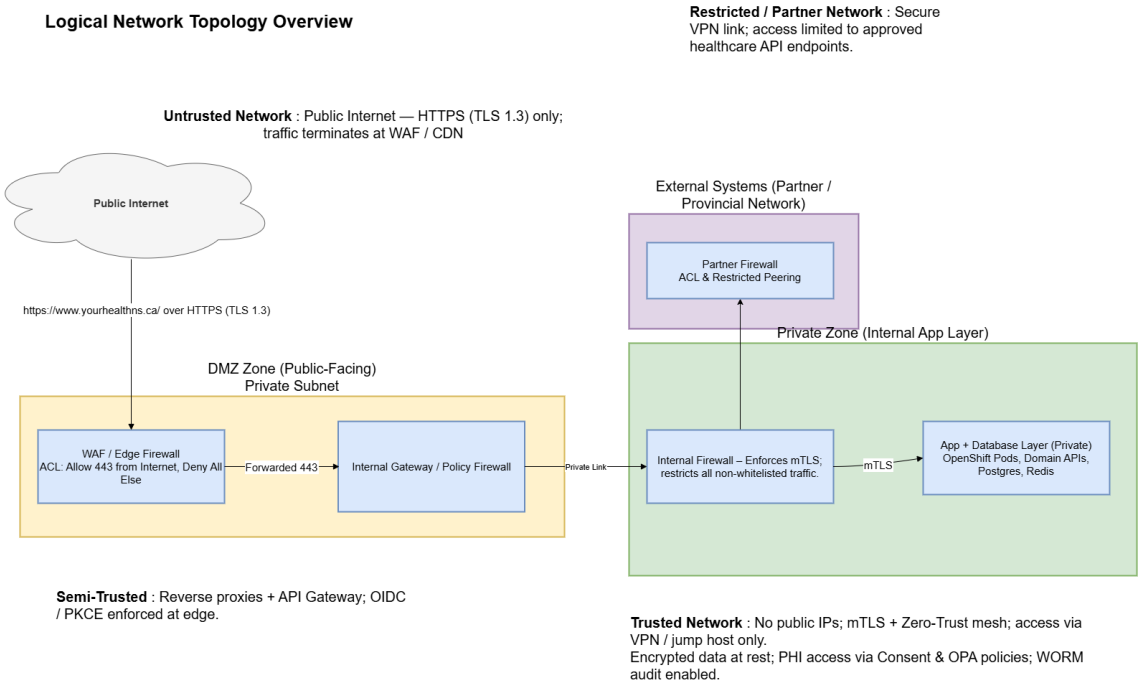

Figure 7 — Network topology and zones. *Editable Source:* Open Draw.io File

The network topology establishes a clear separation of concerns across zones from the public-facing CDN and WAF layer through the API Gateway and BFF services to the internal domain APIs and data stores hosted within OpenShift. Each zone is protected through layered network controls and mutual TLS (mTLS) enforced by the service mesh.

Building on this foundation, the following Security Architecture diagram illustrates how identity, access, and data protection are implemented end-to-end across these zones. It highlights the integration of OIDC-based authentication, PKCE flows, Zero-Trust principles with mTLS, and runtime authorization using OPA and Vault-backed secret management. Together, these components ensure that every request whether user-initiated or service-to-service is authenticated, authorized, encrypted, and auditable.

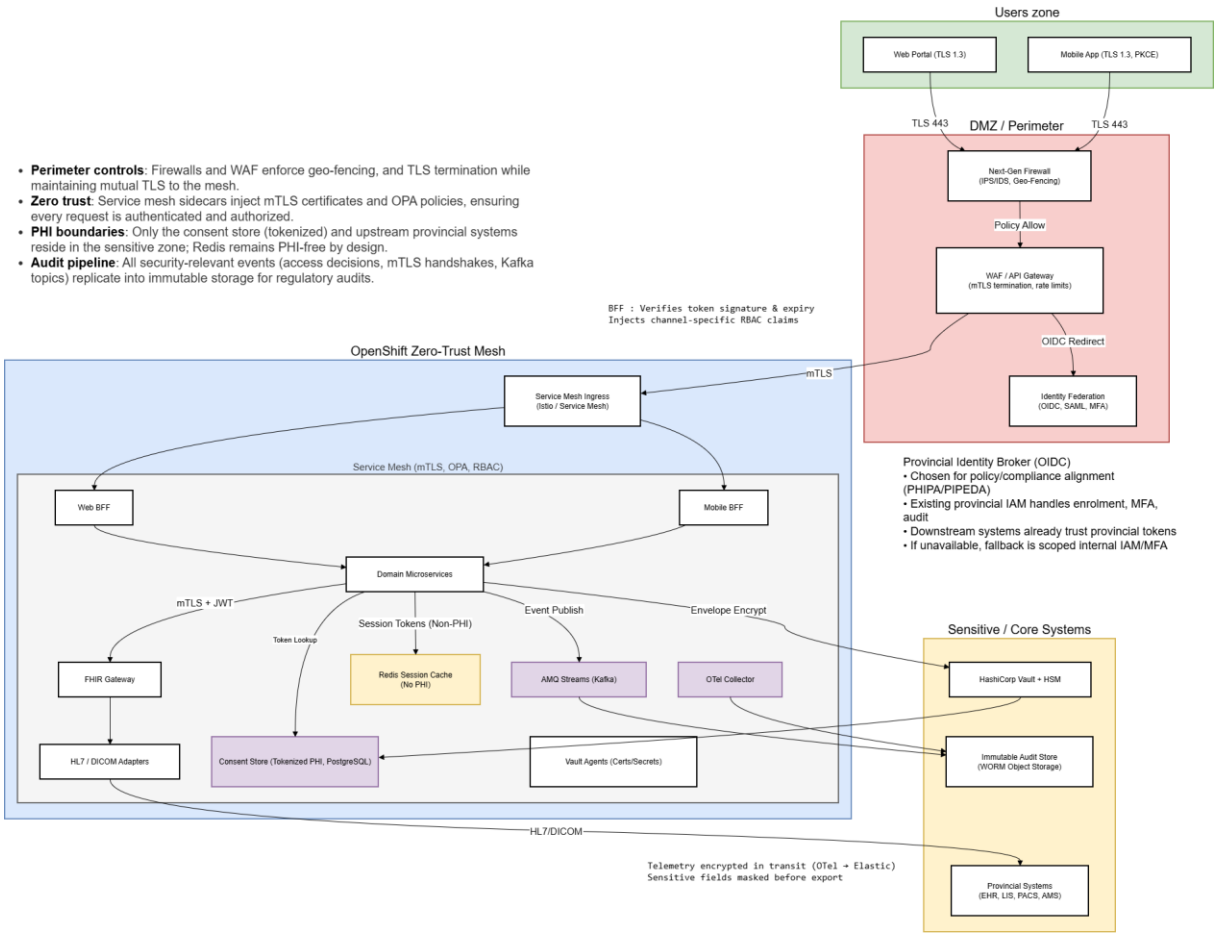

Figure 8 — Security architecture with zero-trust enforcement. *Editable Source:* Open [Draw.io File]

7. Data, Security & DevSecOps

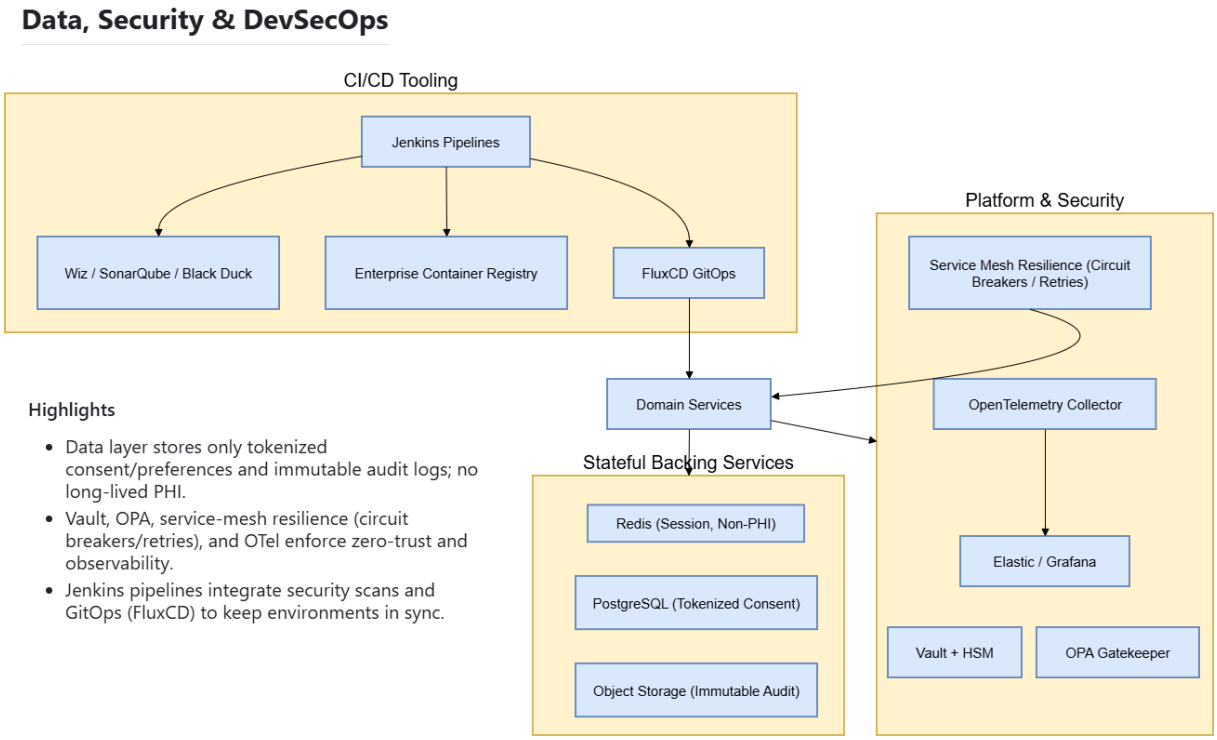

Figure 9 — Data, security, and DevSecOps layers. *Editable Source:* Open Draw.io File

- The platform employs Redis for high-speed caching of session tokens, calendars, and appointment slots, ensuring low-latency responses and graceful degradation through auto-replication and in-memory eviction policies.
- PostgreSQL stores tokenized ePHI such as lab results; data is encrypted with Vault-managed keys, with asynchronous replication and point-in-time recovery for resilience and zero data loss within RPO targets. Vertical scaling for intensive queries.
- Daily incremental and weekly full backups are validated and retained per compliance standards; backup jobs are integrated into CI/CD for audit traceability.
- All auditable API actions flow through Kafka into immutable WORM storage, providing seven-year, tamper-proof retention, Kafka brokers scale horizontally by partition for sustained throughput.
- Vault-managed envelope encryption secures PHI at rest and in transit; Redis caches only non-PHI data.
- All inter-service communication uses mTLS with OPA Gatekeeper policies enforcing least privilege.
- Liveness and readiness probes configured for all microservices ensure proactive fault detection and traffic isolation during failures.

8. CI/CD & Release Management

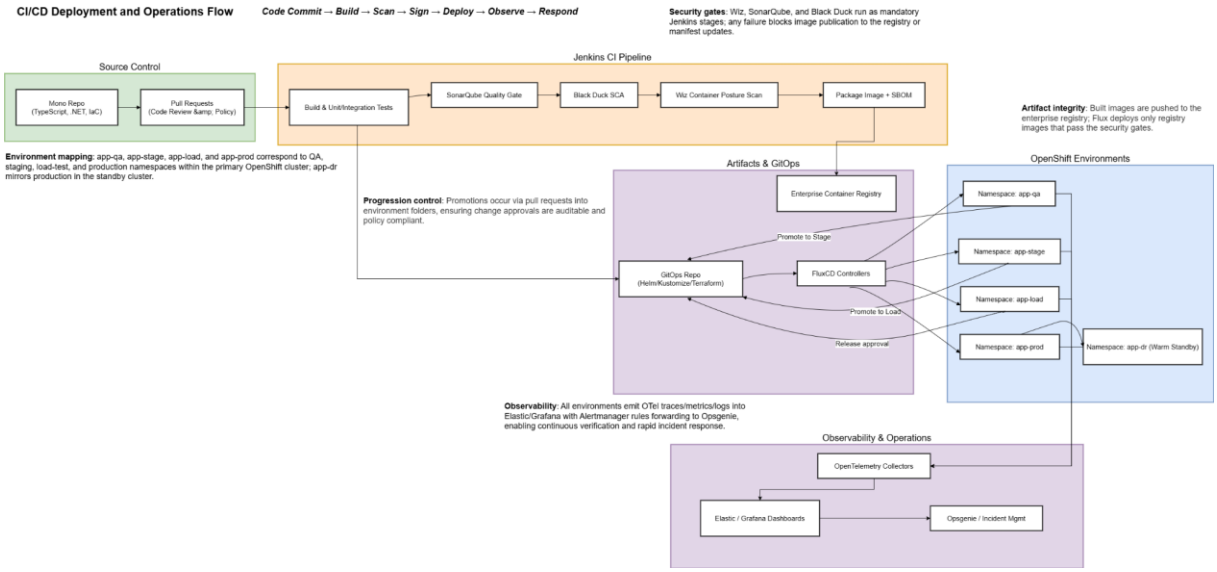

Figure 10 — CI/CD deployment and operations flow. *Editable Source:* Open Draw.io File

Jenkins CI pipelines manage code integration, while FluxCD controllers handle progressive deployment. Approvals are required for production promotion. Observability ensures all deployments are validated in real time.

9. NFR Domain Matrix

This matrix maps key non-functional requirements across the infrastructure, application, and data domains, highlighting how each layer contributes to scalability, resiliency, security, and compliance within the YourHealthNS platform.

NFR / Quality Attribute	Infrastructure Layer	Application Layer	Data Layer
Scalability	OpenShift HPA; stateless containers; zone-level scaling.	Independent scaling of BFFs, APIs, adapters; Kafka partition scaling.	Redis for burst reads. Postgres read replicas, shardable model.
Availability	Multi-AZ OpenShift; redundant gateways; rolling updates.	Active-active APIs and web apps; readiness/liveness probes.	Postgres synchronous replication; Redis HA cluster.
Resiliency	Kubernetes self-healing; automated failover.	Polly circuit breakers isolate failing dependencies; DLQs & outbox pattern protect message flow.	Replayable Kafka topics; back-pressure from consumers prevents overload.
External Integration Resiliency	N/A	If provincial APIs (FHIR, HL7, DICOM) degrade or fail, platform slows gracefully but remains stable; failed calls	Audit trail and pending transaction states preserved until partner recovery.

		retried or queued.	
Performance	Optimized compute allocation; low-latency cluster networking.	Async .NET 8 APIs; Redis (15-min TTL) caching for sub-500 ms P95 latency.	Indexed Postgres reads; Redis for hot data; Kafka event batching.
Security / Zero-Trust	Network segmentation by zone; mTLS mesh; private subnets only.	OIDC + PKCE; OPA Gatekeeper for access; SMART-scope tokens.	Vault-managed encryption; tokenized ePHI; TLS 1.3 enforced.
Reliability	Node recovery automation; HA gateways; pod restart policies.	Idempotent APIs with consistent retry; compensating transactions.	Primary-write / secondary-read model; replication slots maintain sync.
Auditability & Compliance	OpenShift cluster audit trails.	Audit middleware tags API calls → Kafka → WORM.	WORM (S3-compatible) immutable store for 7-year PHI retention.
Observability	OpenTelemetry agents capture traces, metrics, logs.	Correlated trace IDs across BFF, APIs, Kafka; logs aggregated via Elastic.	Query performance and access logs tied to service traces. Custom tools like Redgate.
Maintainability	FluxCD GitOps; Helm versioning; Manifests file	Jenkins CI/CD; rollback ≤5 min; automated testing gates.	Schema versioning.
Compliance & Retention	Keeping up to date documentation. Retention based on vendor	Consent enforcement via SMART scopes; security reviewed in SDLC.	WORM lifecycle policies; encrypted PHI backups.

10. Testing & Quality Strategy

Type of testing to conduct

- Collaborate with Product, QA, and Engineering leads to define, review, and publish comprehensive test cases.
- Prioritize automation in high-value areas to detect defects early and enable a strong shift-left testing strategy.
- Automated Unit, Integration, and Regression test suites are integrated into the CI/CD pipeline.
- Common user and API workflows are auto tested on every commit; regression suites run on nightly builds.
- Periodic Load and Stress tests validate system scalability and concurrency thresholds.
- UI and End-to-End automation ensure seamless user journeys; manual exploratory testing validates critical health workflows and accessibility.
- Security testing covers static and dynamic scans, dependency checks, and scheduled penetration tests.

11. Cost Governance & FinOps

- Establish a centralized dashboard (Power BI / Grafana) to capture and visualize environment-wise cloud and on-prem costs compute, storage, and licensing.
- Integrate automated cost export from OpenShift, Azure, and Jenkins pipelines to attribute spend by service, namespace, or team.
- If automation is constrained, maintain a structured Excel tracker to record monthly costs and publish a lightweight dashboard, ensuring visibility into trends, waste, and optimization opportunities.

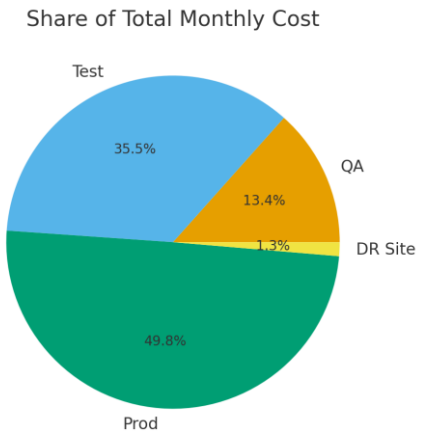

Figure 11 — Monthly cost by environment (illustrative).

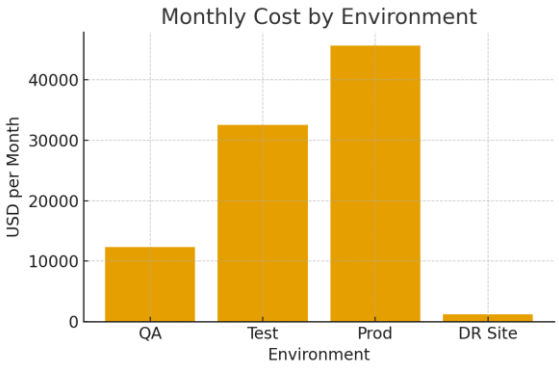

Figure 12 — Share of total monthly cost (illustrative).

Budgets are tagged by environment and service. Alerts trigger at 75/90/100% thresholds. Idle resources in non-prod environments are auto paused. Metrics include $/1K requests and $/active user.

12. Disaster Recovery & Business Continuity

- Warm standby across data centers with replication setup
- GitOps-based restoration, and biannual DR drills ensure continuity.

- Failover tests confirm RPO/RTO targets are met consistently.
- Up to date runbooks and SOPs, no dependency of engineers changing projects
- Document metrics from every DR drill

13. Risk Register & Mitigations

Risk	Impact	Probability	Mitigation
Legacy EHR latency	High	Medium	Asynchronous buffering and clear UI feedback.
Vault/HSM outage	High	Low	HA cluster and transient cache with limited access.
Traffic surge	High	Medium	Autoscaling and runbook-driven prewarming.
FHIR schema drift	Medium	Medium	Contract testing and semantic monitoring.
Insider misuse of PHI	Critical	Low	Least privilege, immutable audit, anomaly detection.
Observability cost growth	Medium	High	OTel sampling, ILM tiering, retention policies.
Physical infrastructure not managed through GitOps	Medium	Medium	Clearly define ownership with IT Ops; Establish change control process.

14. Feature Walkthroughs (Appendix)

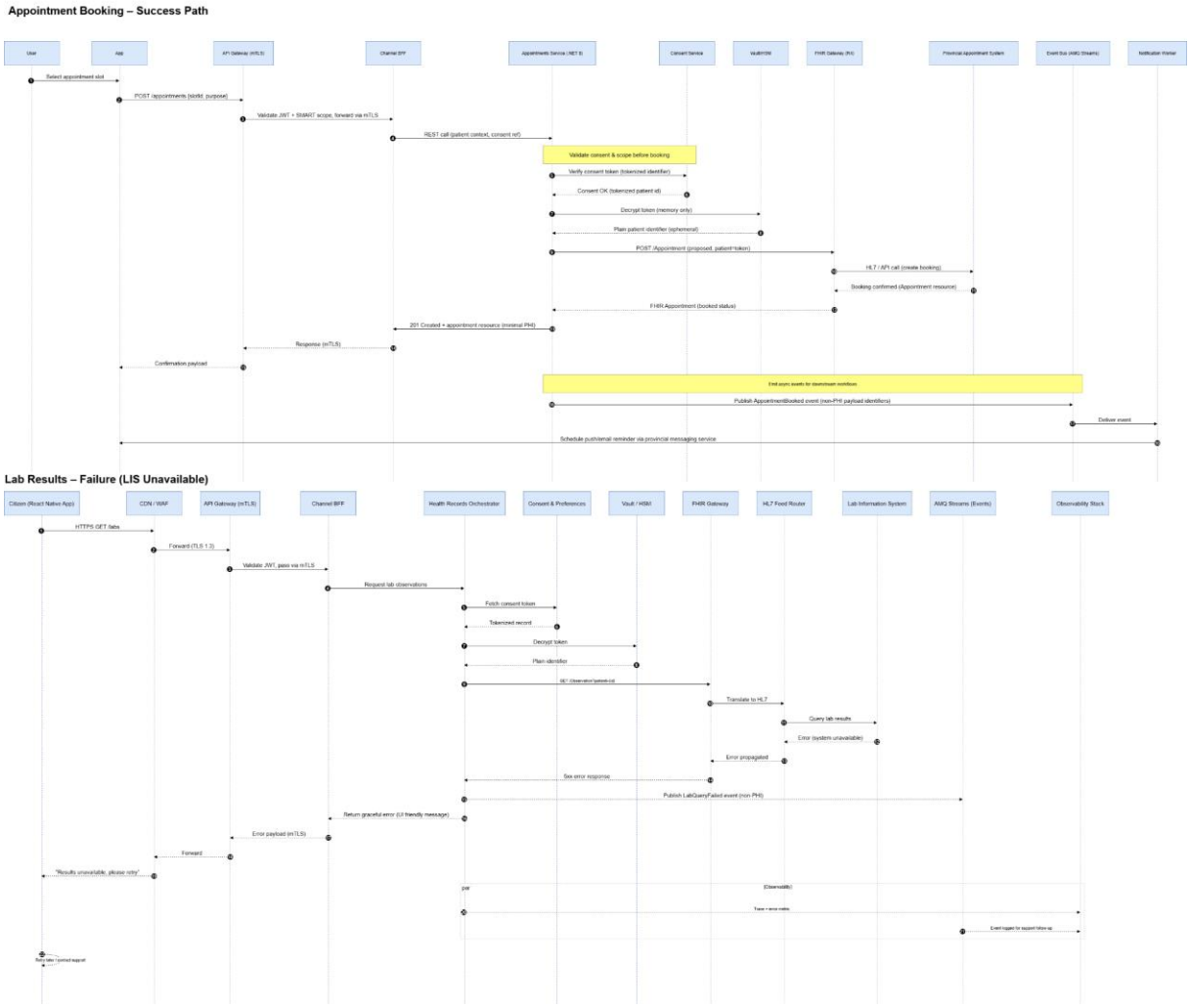

Figure A1 — Appointment booking sequence.

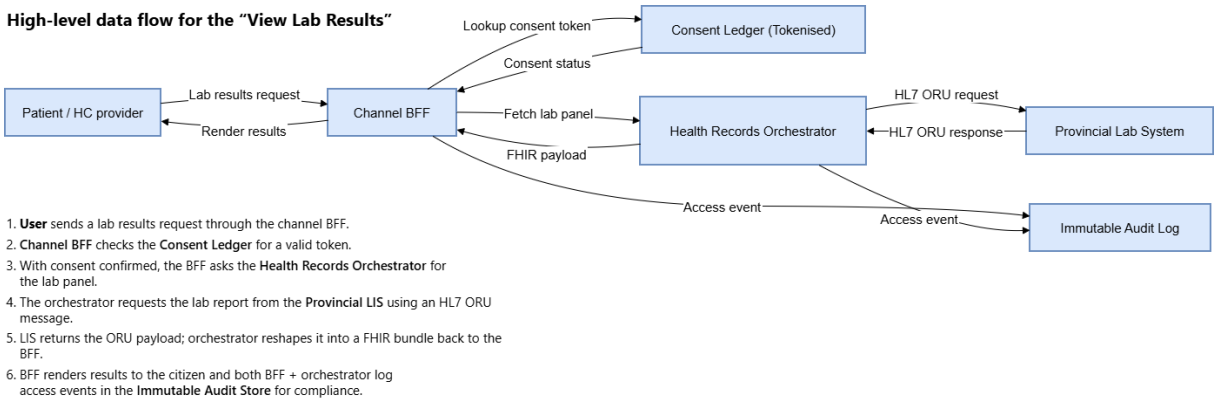

Figure A2 — Data flow for 'View Lab Results'. Editable source Open Draw.io File

15. Compliance Checklist

- PHIPA/PIPEDA: Consent tokens, immutable audit, restricted PHI scope
- SOC 2 Type II: CI/CD controls, incident response logging
- OWASP ASVS/MASVS: Secure coding and mobile protections
- WCAG 2.1 AA: Accessibility compliance
- CIS Benchmarks: Hardened images and cluster policies

16. Compliance & Security playbook

- Protect PHI & Immutability
    - Data retention compliant with govt, Immutable WORM buckets
    - Encryption at rest and in transit, opsgenie as guardrail when jobs detect drift
- Observability & Log Governance
    - Elastic with hot (0-7d), warm (8-30d), cold (>30d) tiers; adaptive sampling 15%
    - Log pipelines redact ePHI
- Interoperability Assurance
    - HL7/FHIR schema validation, SMART scope profiles versioned in Git
- Access & Consent Control
    - OPA Gatekeeper deny-all baseline, RBAC mapped to data managers, admin, regular users, Token issuance
- Data Quality & Governance
    - Kafka ensure exactly once delivery
    - Automated data quality checks either at data store level or using AI

17. System Evolution & Future Enhancements

- **Delivery Milestones:**
    - Q1 – Platform Foundations: Deliver minimum viable architecture with OpenShift, security, CI/CD, and observability.
    - Q2 – Core Services: Build and expose MVP APIs in QA with FHIR/HL7 integration.
    - Q3 – Production Enablement: Scale MVP to production with DR, FinOps, and monitoring.
    - Q4 – Optimization & Compliance: Harden platform for full rollout, audits, and handover.
- **Future Enhancements:**
    - Active-active DR and cross-region federation
    - FHIR bulk export and analytics integration
    - Webhook integration and push-poll mechanism for critical workflows
    - AI-driven PHI detection and compliance tagging
    - Continuous verification and chaos testing