YourHealthNS – Solution Architecture Proposal

Prepared by: Suresh Kumar Balasubramanian

Nova Scotia Health

Version 1.1

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.

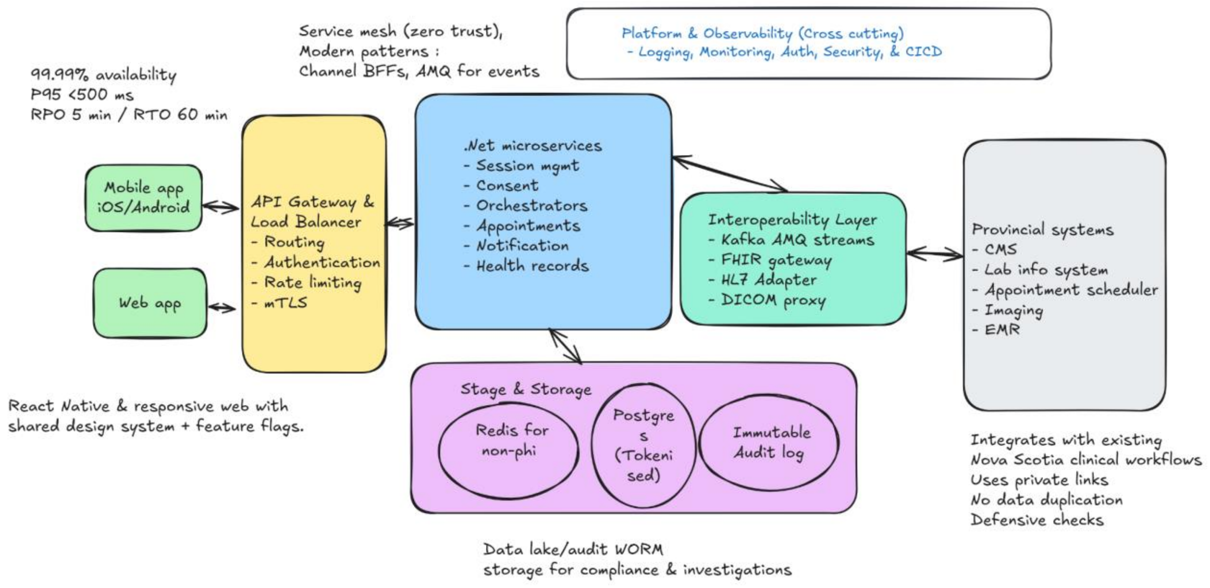


Figure 1 — High-level platform overview. Editable source Open link

2. Requirements & Constraints Mapping

The table below maps assessment requirements to corresponding architectural design responses.

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) and Jenkins pipelines.	No dependency on public cloud runtimes.

BFF layer for frontend apps	Implement Backend-for-Frontend (BFF) pattern. Web BFF in Node.js (Next.js API routes) and Mobile BFF in. NET 8 Minimal API (strongly typed)	Web team owns Web BFF; Mobile BFF defined by mobile team and implemented by backend team for reuse of .NET libraries
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
- UI/UX wireframe definitions and SOPs for Incident responses

4. Capacity & Performance Targets

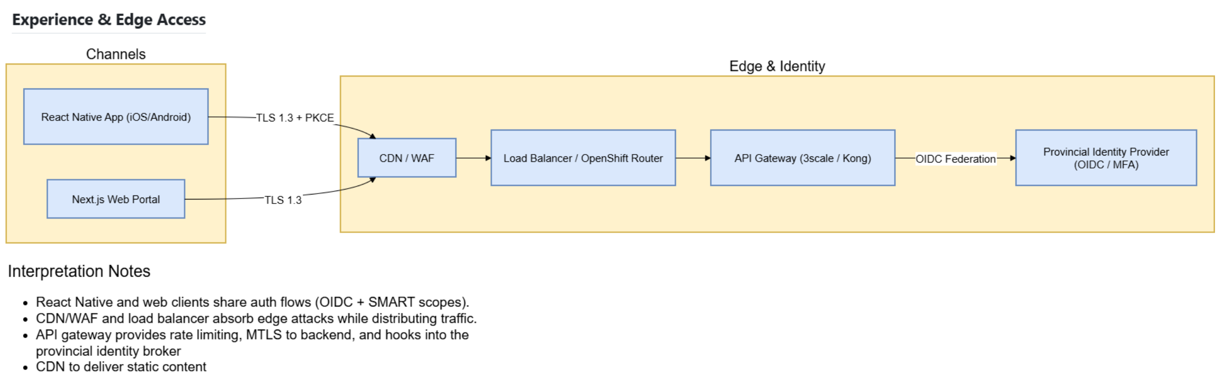
Targets: 350 K monthly active users, ~1 M API calls/day, and peak load of 1,000 RPS (~3–5 K concurrent sessions; up to 10 K burst).
Throughput: Kafka ≈ 1,200 msg/s, PostgreSQL ≈ 1,000 reads/s, Redis ≈ 600 ops/s.
SLOs: P95 latency < 500 ms, 99.99 % availability, RPO ≤ 5 min, RTO ≤ 60 min.

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.

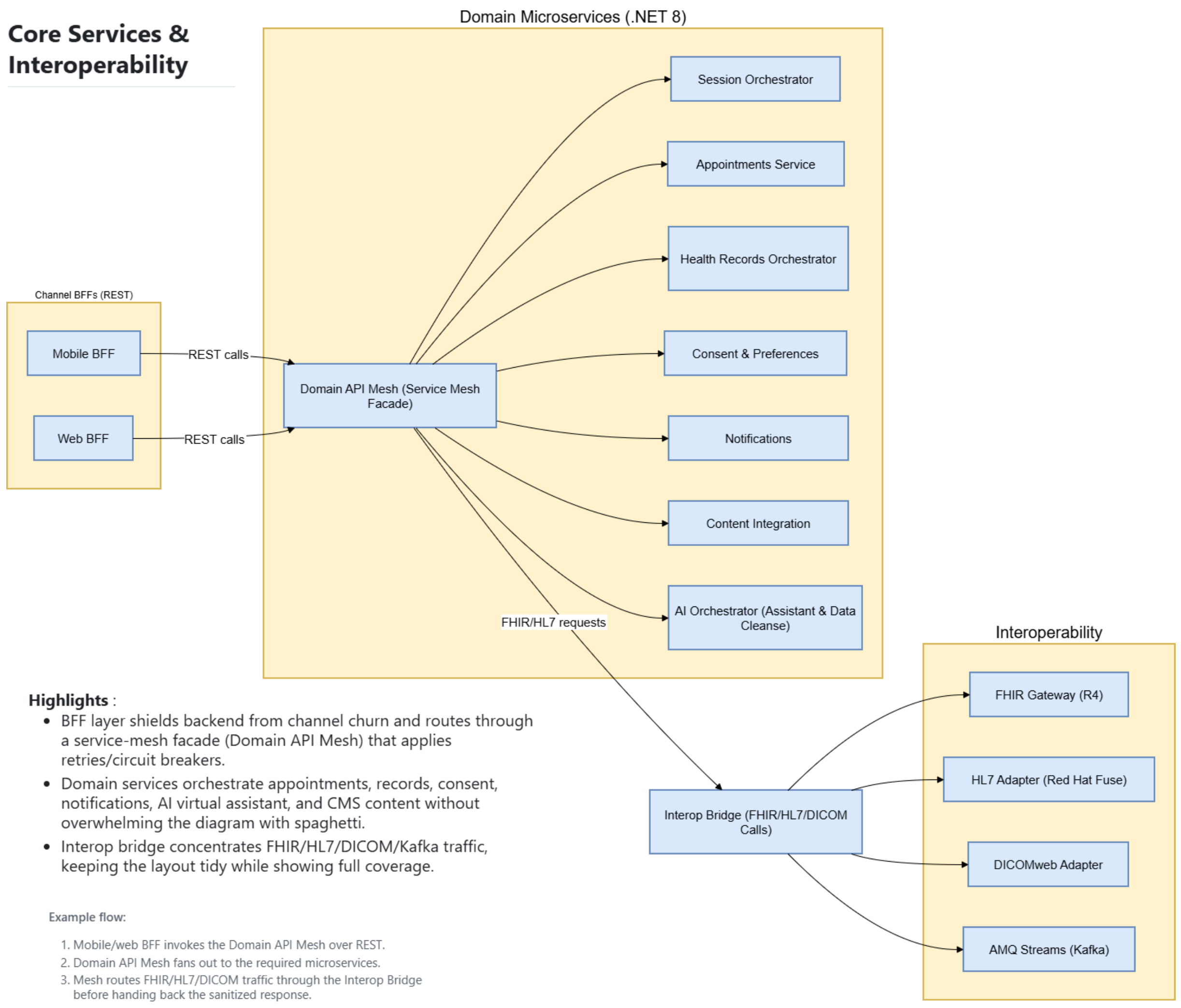

Experience & Edge Access
Channels
React Native App (iOS/Android)
Next.js Web Portal
TLS 1.3 + PKCE
TLS 1.3
Edge & Identity
CDN / WAF
Load Balancer / OpenShift Router
API Gateway (3scale / Kong)
OIDC Federation
Provincial Identity Provider (OIDC / MFA)
Interpretation Notes

- React Native and web clients share auth flows (OIDC + SMART scopes).
- CDN/WAF and load balancer absorb edge attacks while distributing traffic.
- API gateway provides rate limiting, MTLS to backend, and hooks into the provincial identity broker
- CDN to deliver static content

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.



Highlights :
- BFF layer shields backend from channel churn and routes through a service-mesh facade (Domain API Mesh) that applies retries/circuit breakers.
- Domain services orchestrate appointments, records, consent, notifications, AI virtual assistant, and CMS content without overwhelming the diagram with spaghetti.
- Interop bridge concentrates FHIR/HL7/DICOM/Kafka traffic, keeping the layout tidy while showing full coverage.

Example flow:

1. Mobile/web BFF invokes the Domain API Mesh over REST.
2. Domain API Mesh fans out to the required microservices.
3. Mesh routes FHIR/HL7/DICOM traffic through the Interop Bridge before handing back the sanitized response.

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.

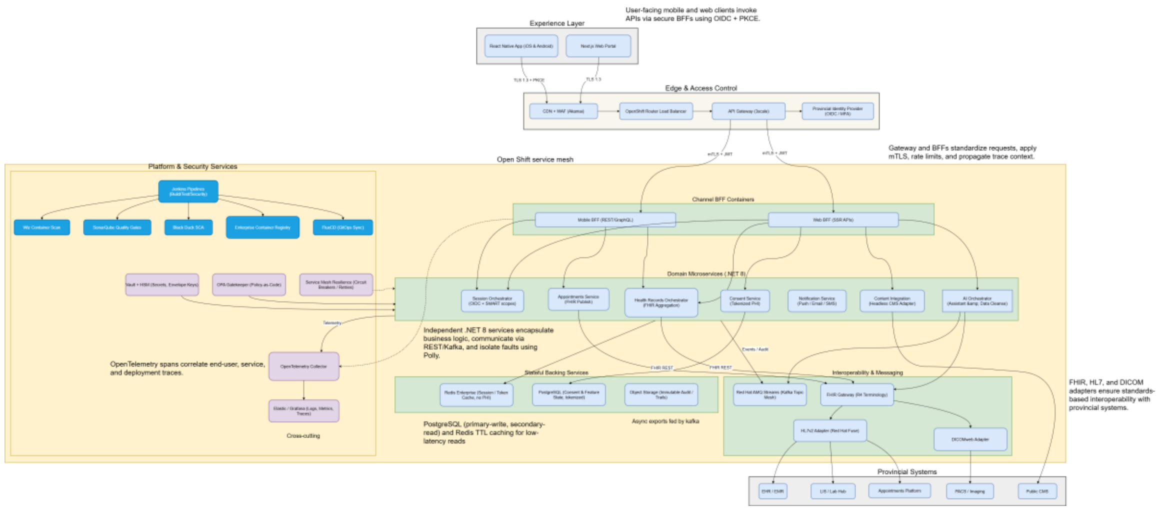

User-facing mobile and web clients invoke APIs via secure BFFs using OIDC + PKCE.
Experience Layer
React Native App (iOS & Android)
Next.js Web Portal
TLS 1.3 + PKCE
TLS 1.3
Edge & Access Control
CDN + WAF (Akamai)
OpenShift Router Load Balancer
API Gateway (3scale)
Provincial Identity Provider (OIDC / MFA)
mTLS + JWT
mTLS + JWT
Gateway and BFFs standardize requests, apply mTLS, rate limits, and propagate trace context.
Open Shift service mesh
Platform & Security Services
Jenkins Pipelines (Build/Test/Security)
Wiz Container Scan
SonarQube Quality Gates
Black Duck SCA
Enterprise Container Registry
FluxCD (GitOps Sync)
Vault + HSM (Secrets, Envelope Keys)
OPA Gatekeeper (Policy-as-Code)
Service Mesh Resilience (Circuit Breakers / Retries)
Telemetry
OpenTelemetry spans correlate end-user, service, and deployment traces.
OpenTelemetry Collector
Elastic / Grafana (Logs, Metrics, Traces)
Cross-cutting
Channel BFF Containers
Mobile BFF (REST/GraphQL)
Web BFF (SSR APIs)
Domain Microservices (.NET 8)
Session Orchestrator (OIDC + SMART scopes)
Appointments Service (FHIR Publish)
Health Records Orchestrator (FHIR Aggregation)
Consent Service (Tokenized PHI)
Notification Service (Push / Email / SMS)
Content Integration (Headless CMS Adapter)
AI Orchestrator (Assistant &amp; Data Cleanse)
Independent .NET 8 services encapsulate business logic, communicate via REST/Kafka, and isolate faults using Polly.
Events / Audit
FHIR REST
FHIR REST
Stateful Backing Services
Redis Enterprise (Session / Token Cache, no PHI)
PostgreSQL (Consent & Feature State, tokenized)
Object Storage (Immutable Audit / Trails)
PostgreSQL (primary-write, secondary-read) and Redis TTL caching for low-latency reads
Async exports fed by kafka
Interoperability & Messaging
Red Hat AMQ Streams (Kafka Topic Mesh)
FHIR Gateway (R4 Terminology)
HL7v2 Adapter (Red Hat Fuse)
DICOMweb Adapter
FHIR, HL7, and DICOM adapters ensure standards-based interoperability with provincial systems.
Provincial Systems
EHR / EMR
LIS / Lab Hub
Appointments Platform
PACS / Imaging
Public CMS

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

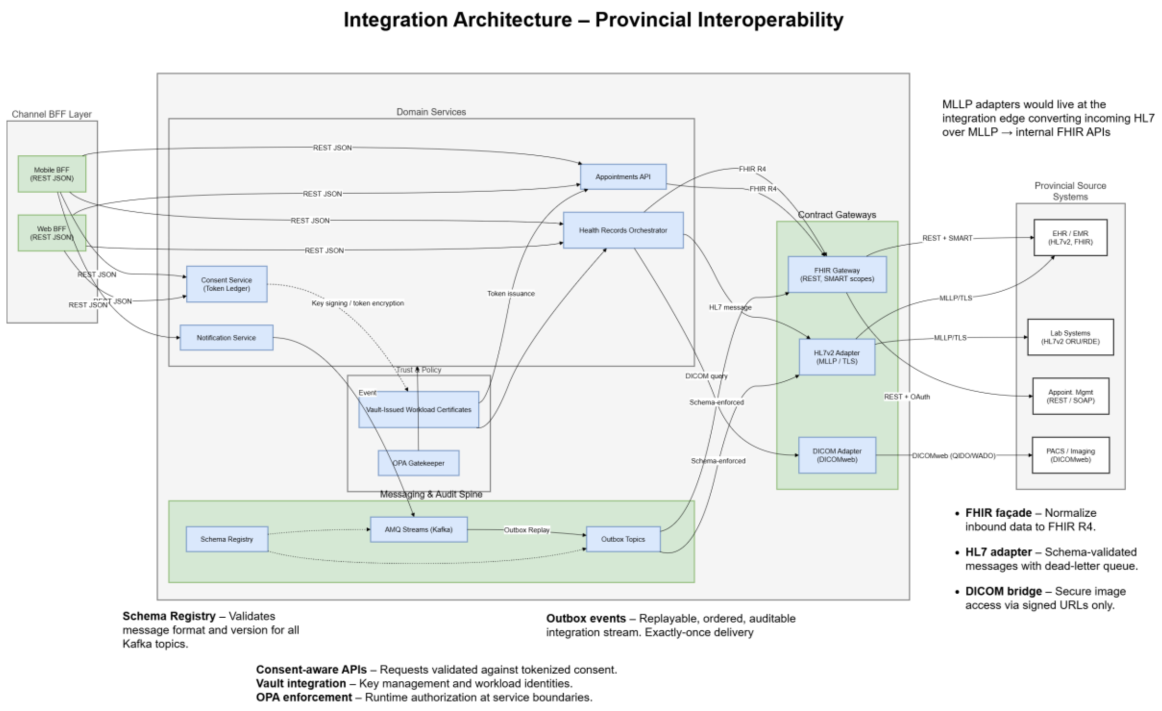
See also: Section 5.5 for corresponding physical topology.

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.
- 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.



MLLP adapters would live at the integration edge converting incoming HL7 over MLLP → internal FHIR APIs

- **FHIR façade** – Normalize inbound data to FHIR R4.
- **HL7 adapter** – Schema-validated messages with dead-letter queue.
- **DICOM bridge** – Secure image access via signed URLs only.

Schema Registry – Validates message format and version for all Kafka topics.

Outbox events – Replayable, ordered, auditable integration stream. Exactly-once delivery

Consent-aware APIs – Requests validated against tokenized consent.
Vault integration – Key management and workload identities.
OPA enforcement – Runtime authorization at service boundaries.

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.

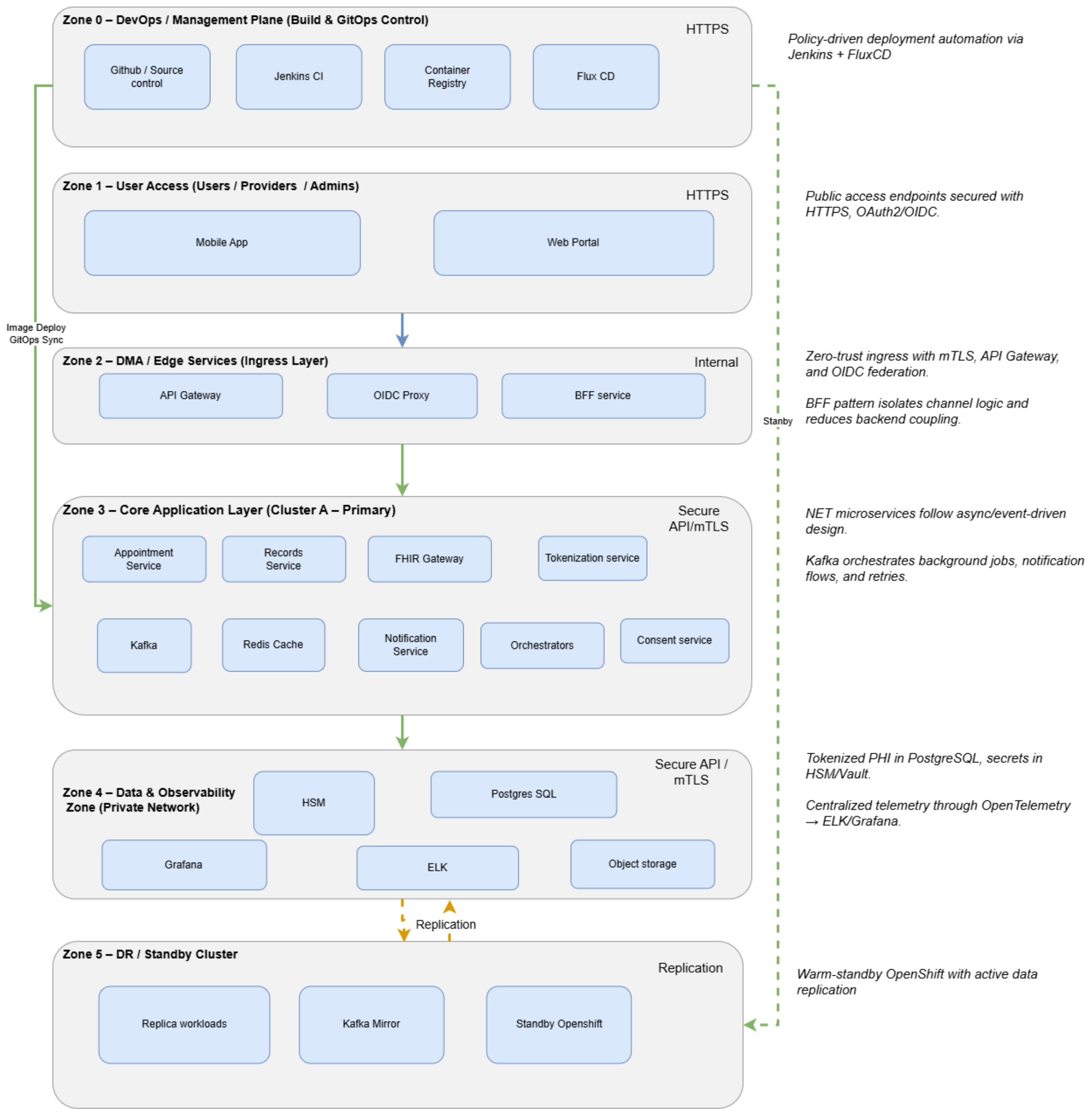

Zone 0 – DevOps / Management Plane (Build & GitOps Control)
HTTPS
Github / Source control
Jenkins CI
Container Registry
Flux CD
Policy-driven deployment automation via Jenkins + FluxCD
Zone 1 – User Access (Users / Providers / Admins)
HTTPS
Mobile App
Web Portal
Public access endpoints secured with HTTPS, OAuth2/OIDC.
Image Deploy GitOps Sync
Zone 2 – DMA / Edge Services (Ingress Layer)
Internal
API Gateway
OIDC Proxy
BFF service
Zero-trust ingress with mTLS, API Gateway, and OIDC federation.
BFF pattern isolates channel logic and reduces backend coupling.
Stanby
Zone 3 – Core Application Layer (Cluster A – Primary)
Secure API/mTLS
Appointment Service
Records Service
FHIR Gateway
Tokenization service
Kafka
Redis Cache
Notification Service
Orchestrators
Consent service
NET microservices follow async/event-driven design.
Kafka orchestrates background jobs, notification flows, and retries.
Zone 4 – Data & Observability Zone (Private Network)
Secure API / mTLS
HSM
Postgres SQL
Grafana
ELK
Object storage
Tokenized PHI in PostgreSQL, secrets in HSM/Vault.
Centralized telemetry through OpenTelemetry → ELK/Grafana.
Replication
Zone 5 – DR / Standby Cluster
Replication
Replica workloads
Kafka Mirror
Standby Openshift
Warm-standby OpenShift with active data replication

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.

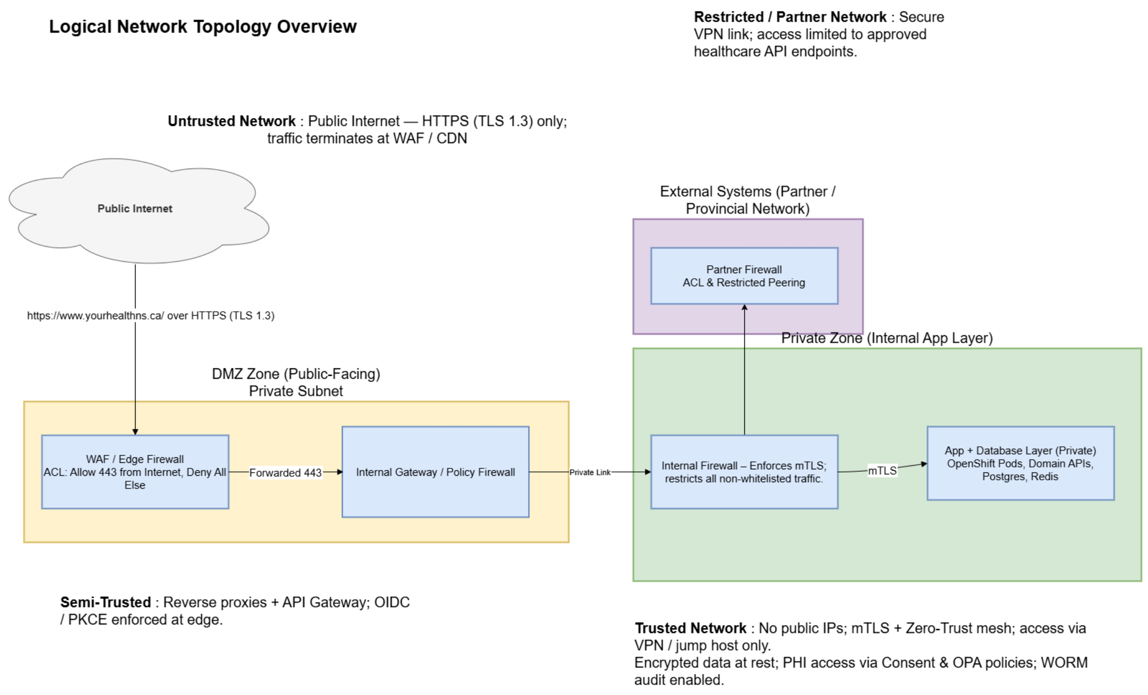
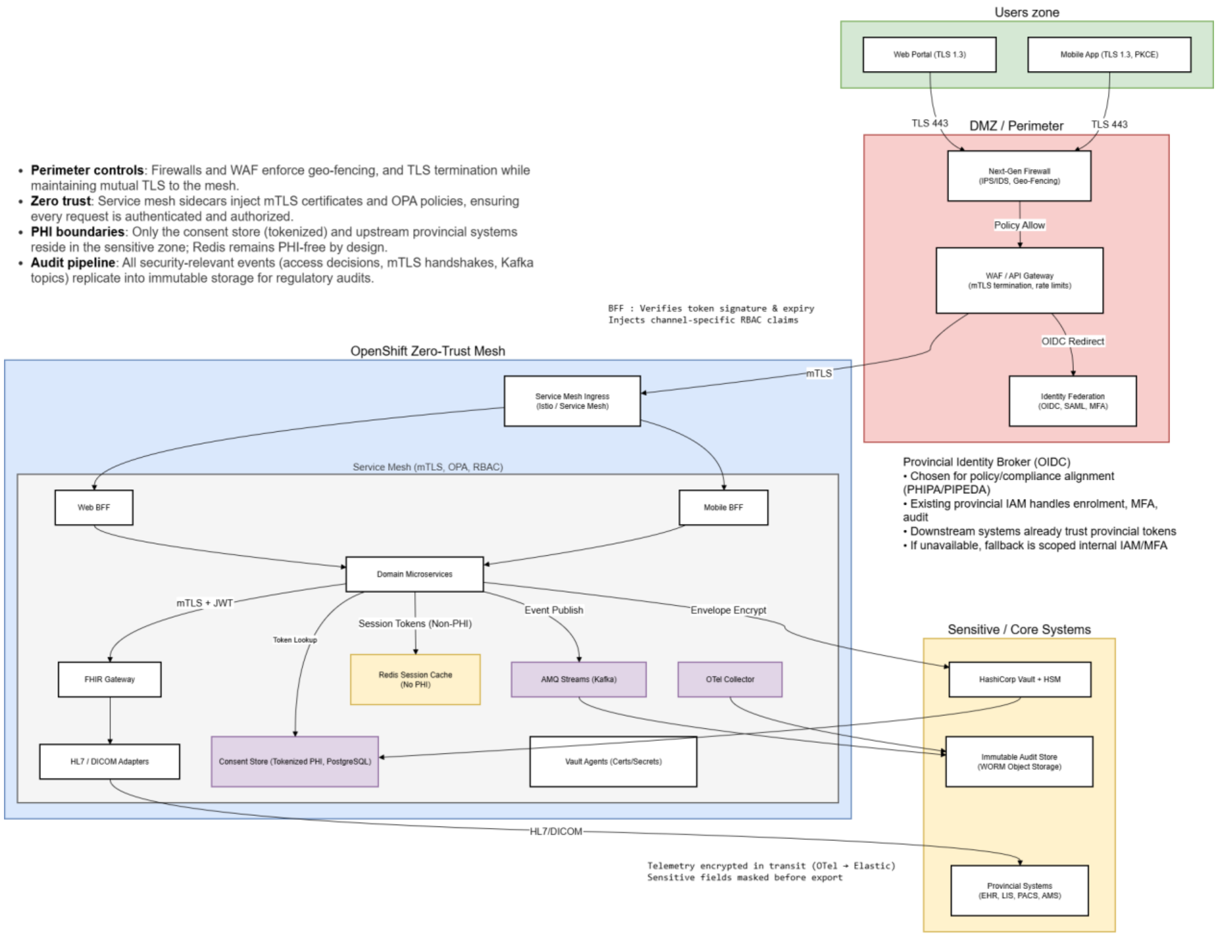

Logical Network Topology Overview
Restricted / Partner Network : Secure VPN link; access limited to approved healthcare API endpoints.
Untrusted Network : Public Internet — HTTPS (TLS 1.3) only; traffic terminates at WAF / CDN
Public Internet
https://www.yourhealthns.ca/ over HTTPS (TLS 1.3)
External Systems (Partner / Provincial Network)
Partner Firewall ACL & Restricted Peering
Private Zone (Internal App Layer)
DMZ Zone (Public-Facing) Private Subnet
WAF / Edge Firewall ACL: Allow 443 from Internet, Deny All Else
Forwarded 443
Internal Gateway / Policy Firewall
Private Link
Internal Firewall – Enforces mTLS; restricts all non-whitelisted traffic.
mTLS
App + Database Layer (Private) OpenShift Pods, Domain APIs, Postgres, Redis
Semi-Trusted : Reverse proxies + API Gateway; OIDC / PKCE enforced at edge.
Trusted Network : No public IPs; mTLS + Zero-Trust mesh; access via VPN / jump host only. Encrypted data at rest; PHI access via Consent & OPA policies; WORM audit enabled.

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.


Users zone
Web Portal (TLS 1.3)
Mobile App (TLS 1.3, PKCE)
TLS 443
DMZ / Perimeter
TLS 443
Next-Gen Firewall (IPS/IDS, Geo-Fencing)
Policy Allow
WAF / API Gateway (mTLS termination, rate limits)
OIDC Redirect
Identity Federation (OIDC, SAML, MFA)
mTLS
OpenShift Zero-Trust Mesh
Service Mesh Ingress (Istio / Service Mesh)
Service Mesh (mTLS, OPA, RBAC)
Web BFF
Mobile BFF
Domain Microservices
mTLS + JWT
Token Lookup
Session Tokens (Non-PHI)
Event Publish
Envelope Encrypt
FHIR Gateway
Redis Session Cache (No PHI)
AMQ Streams (Kafka)
OTel Collector
HL7 / DICOM Adapters
Consent Store (Tokenized PHI, PostgreSQL)
Vault Agents (Certs/Secrets)
HL7/DICOM
Sensitive / Core Systems
HashiCorp Vault + HSM
Immutable Audit Store (WORM Object Storage)
Provincial Systems (EHR, LIS, PACS, AMS)

- **Perimeter controls**: Firewalls and WAF enforce geo-fencing, and TLS termination while maintaining mutual TLS to the mesh.
- **Zero trust**: Service mesh sidecars inject mTLS certificates and OPA policies, ensuring every request is authenticated and authorized.
- **PHI boundaries**: Only the consent store (tokenized) and upstream provincial systems reside in the sensitive zone; Redis remains PHI-free by design.
- **Audit pipeline**: All security-relevant events (access decisions, mTLS handshakes, Kafka topics) replicate into immutable storage for regulatory audits.

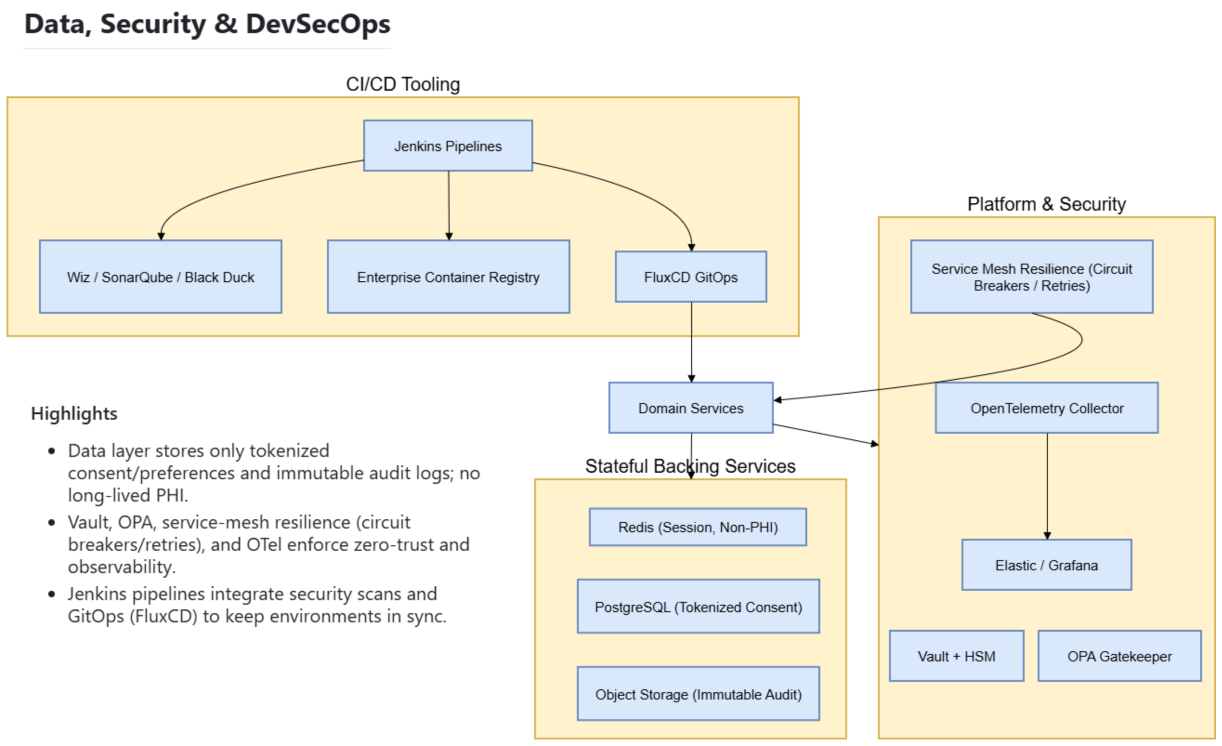
```
BFF : Verifies token signature & expiry
Injects channel-specific RBAC claims
```

Provincial Identity Broker (OIDC)
• Chosen for policy/compliance alignment (PHIPA/PIPEDA)
• Existing provincial IAM handles enrolment, MFA, audit
• Downstream systems already trust provincial tokens
• If unavailable, fallback is scoped internal IAM/MFA

```
Telemetry encrypted in transit (OTel → Elastic)
Sensitive fields masked before export
```

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

7. Data, Security & DevSecOps


CI/CD Tooling
Jenkins Pipelines
Wiz / SonarQube / Black Duck
Enterprise Container Registry
FluxCD GitOps
Platform & Security
Service Mesh Resilience (Circuit Breakers / Retries)
Domain Services
OpenTelemetry Collector
Stateful Backing Services
Redis (Session, Non-PHI)
PostgreSQL (Tokenized Consent)
Object Storage (Immutable Audit)
Elastic / Grafana
Vault + HSM
OPA Gatekeeper

Data, Security & DevSecOps

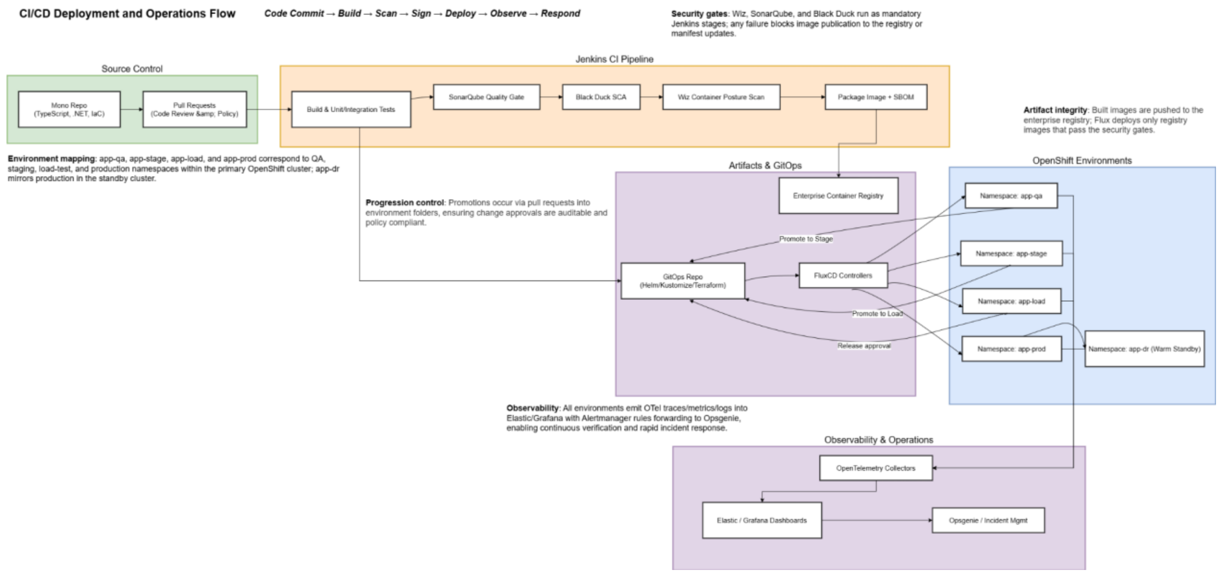
Highlights

- Data layer stores only tokenized consent/preferences and immutable audit logs; no long-lived PHI.
- Vault, OPA, service-mesh resilience (circuit breakers/retries), and OTel enforce zero-trust and observability.
- Jenkins pipelines integrate security scans and GitOps (FluxCD) to keep environments in sync.

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.

8. CI/CD & Release Management


CI/CD Deployment and Operations Flow
Code Commit → Build → Scan → Sign → Deploy → Observe → Respond
Source Control
Mono Repo (TypeScript, .NET, IaC)
Pull Requests (Code Review &amp; Policy)
Jenkins CI Pipeline
Build & Unit/Integration Tests
SonarQube Quality Gate
Black Duck SCA
Wiz Container Posture Scan
Package Image + SBOM
Artifacts & GitOps
Enterprise Container Registry
GitOps Repo (Helm/Kustomize/Terraform)
FluxCD Controllers
Promote to Stage
Promote to Load
Release approval
OpenShift Environments
Namespace: app-qa
Namespace: app-stage
Namespace: app-load
Namespace: app-prod
Namespace: app-dr (Warm Standby)
Observability & Operations
OpenTelemetry Collectors
Elastic / Grafana Dashboards
Opsgenie / Incident Mgmt

Security gates: Wiz, SonarQube, and Black Duck run as mandatory Jenkins stages; any failure blocks image publication to the registry or manifest updates.

Environment mapping: app-qa, app-stage, app-load, and app-prod correspond to QA, staging, load-test, and production namespaces within the primary OpenShift cluster; app-dr mirrors production in the standby cluster.

Artifact integrity: Built images are pushed to the enterprise registry; Flux deploys only registry images that pass the security gates.

Progression control: Promotions occur via pull requests into environment folders, ensuring change approvals are auditable and policy compliant.

Observability: All environments emit OTel traces/metrics/logs into Elastic/Grafana with Alertmanager rules forwarding to Opsgenie, enabling continuous verification and rapid incident response.

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.

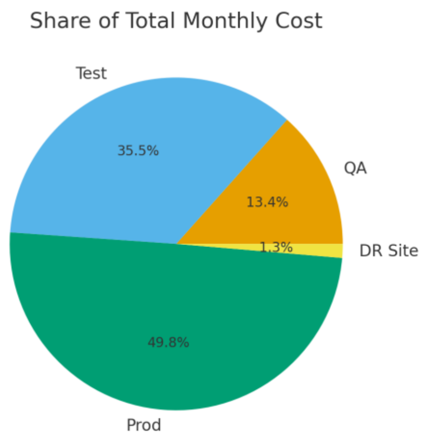


Figure 11 — Monthly cost by environment (illustrative).

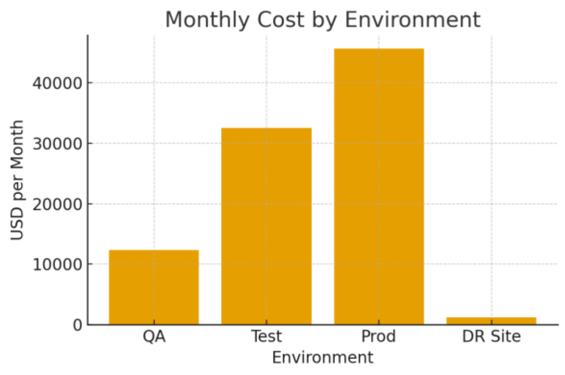


Figure 12 — Share of total monthly cost (illustrative).

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)

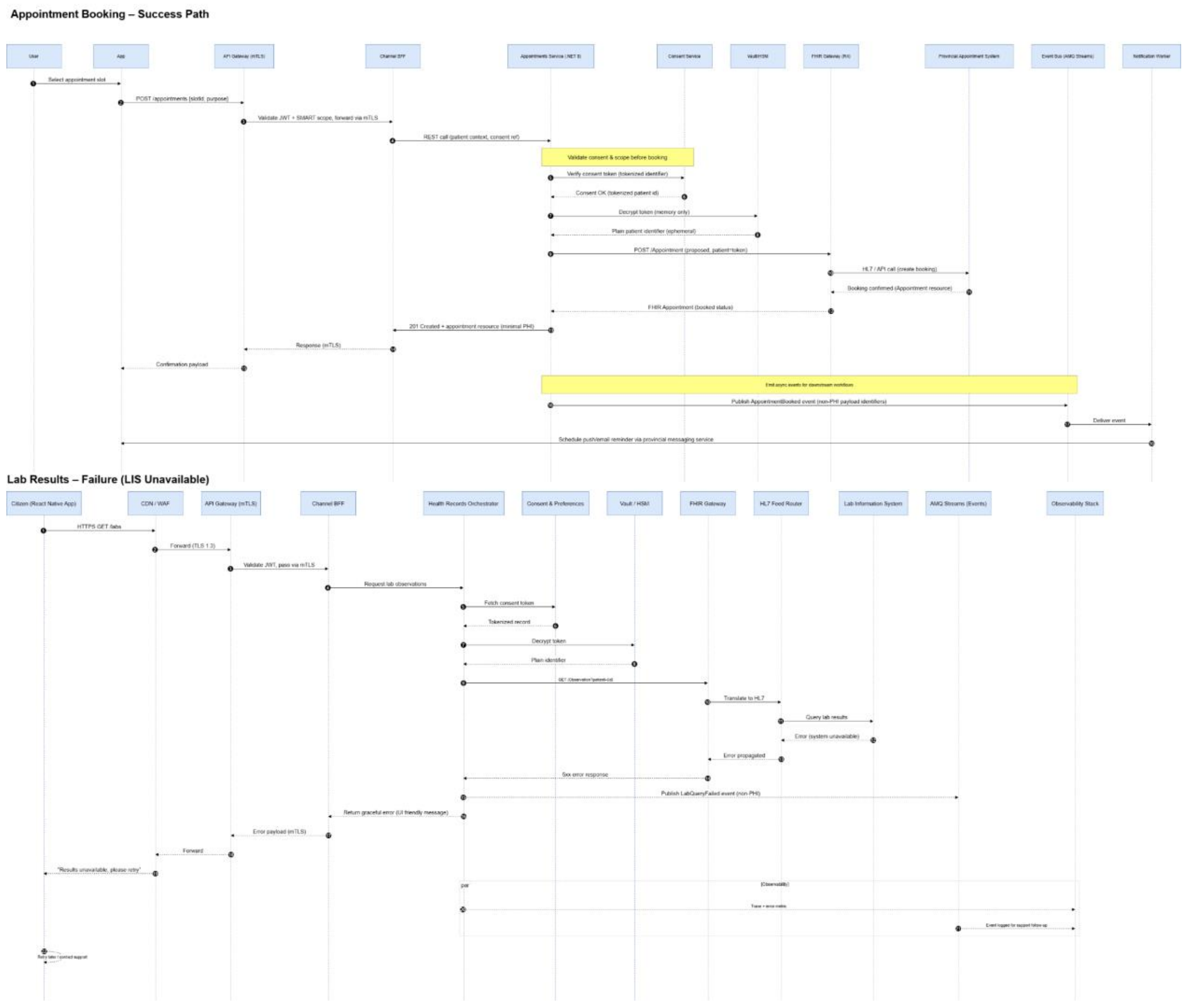
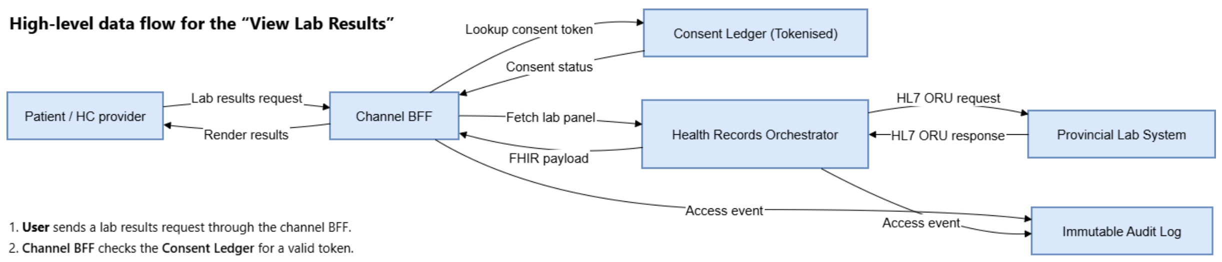

Appointment Booking – Success Path
User
App
API Gateway (mTLS)
Channel BFF
Appointments Service (.NET 8)
Consent Service
Vault/HSM
FHIR Gateway (R4)
Provincial Appointment System
Event Bus (AMQ Streams)
Notification Worker
Select appointment slot
POST /appointments (slotId, purpose)
Validate JWT + SMART scope, forward via mTLS
REST call (patient context, consent ref)
Validate consent & scope before booking
Verify consent token (tokenized identifier)
Consent OK (tokenized patient id)
Decrypt token (memory only)
Plain patient identifier (ephemeral)
POST /Appointment (proposed, patient=token)
HL7 / API call (create booking)
Booking confirmed (Appointment resource)
FHIR Appointment (booked status)
201 Created + appointment resource (minimal PHI)
Response (mTLS)
Confirmation payload
Emit async events for downstream workflows
Publish AppointmentBooked event (non-PHI payload identifiers)
Deliver event
Schedule push/email reminder via provincial messaging service
Lab Results – Failure (LIS Unavailable)
Citizen (React Native App)
CDN / WAF
API Gateway (mTLS)
Channel BFF
Health Records Orchestrator
Consent & Preferences
Vault / HSM
FHIR Gateway
HL7 Feed Router
Lab Information System
AMQ Streams (Events)
Observability Stack
HTTPS GET /labs
Forward (TLS 1.3)
Validate JWT, pass via mTLS
Request lab observations
Fetch consent token
Tokenized record
Decrypt token
Plain identifier
GET /Observation?patient={id}
Translate to HL7
Query lab results
Error (system unavailable)
Error propagated
5xx error response
Publish LabQueryFailed event (non-PHI)
Return graceful error (UI friendly message)
Error payload (mTLS)
Forward
"Results unavailable, please retry"
par
[Observability]
Trace + error metric
Event logged for support follow-up
Retry later / contact support

Figure A1 — Appointment booking sequence.



1. **User** sends a lab results request through the channel BFF.
2. **Channel BFF** checks the **Consent Ledger** for a valid token.
3. With consent confirmed, the BFF asks the **Health Records Orchestrator** for the lab panel.
4. The orchestrator requests the lab report from the **Provincial LIS** using an HL7 ORU message.
5. LIS returns the ORU payload; orchestrator reshapes it into a FHIR bundle back to the BFF.
6. BFF renders results to the citizen and both BFF + orchestrator log access events in the **Immutable Audit Store** for compliance.

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
- Audit Readiness
  - Internal portal helps access logs, incident timelines, retention proofs to assist legal
  - Regular review of auditing to clean up internal employees who left the company

17. System Evolution & Future Enhancements

- Active-active DR and cross-region federation
- FHIR bulk export and analytics integration
- Delegated access and adaptive consent
- Webhook integration and push-poll mechanism for critical workflows
- AI-driven PHI detection and compliance tagging
- Continuous verification and chaos testing