On-Call Tenant Architecture Report

Case Management Domain - Complete Architecture Analysis

Executive Summary

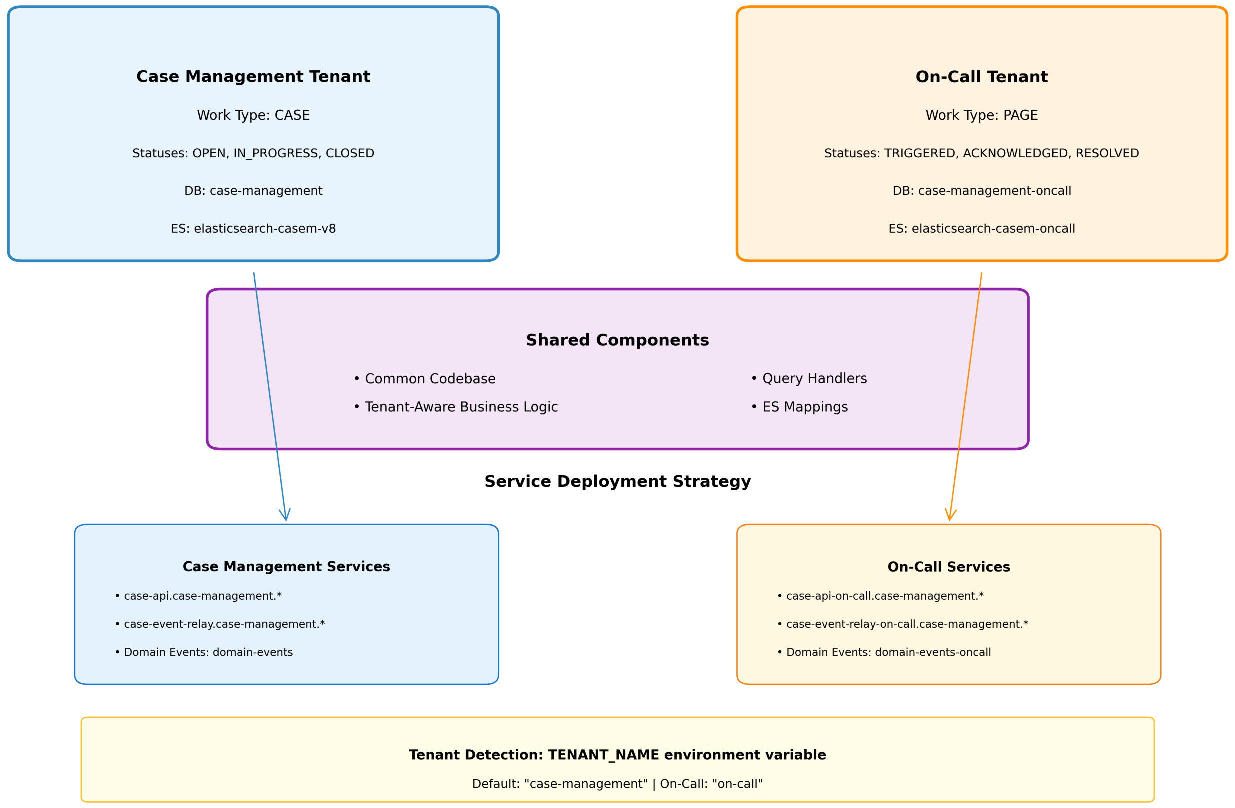
This report provides a comprehensive analysis of the on-call tenant architecture within the case management domain. The on-call tenant operates as a completely separate tenant from the main case management system, with its own databases, Elasticsearch clusters, and service instances while sharing the same codebase through tenant-aware business logic. Key architectural highlights include complete data isolation, independent scaling capabilities, separate failure domains, and optimized configurations for on-call specific workflows including escalation policies, responder management, and live call handling.

Table of Contents

1. Tenant Architecture Overview

The case management domain implements a sophisticated multi-tenant architecture where the on-call tenant operates completely independently from the main case management tenant. This separation ensures data isolation, independent scaling, and specialized business logic optimized for on-call workflows.

Case Management Multi-Tenant Architecture


Case Management Tenant
Work Type: CASE
Statuses: OPEN, IN_PROGRESS, CLOSED
DB: case-management
ES: elasticsearch-casem-v8
On-Call Tenant
Work Type: PAGE
Statuses: TRIGGERED, ACKNOWLEDGED, RESOLVED
DB: case-management-oncall
ES: elasticsearch-casem-oncall
Shared Components
• Common Codebase
• Tenant-Aware Business Logic
• Query Handlers
• ES Mappings
Service Deployment Strategy
Case Management Services
• case-api.case-management.*
• case-event-relay.case-management.*
• Domain Events: domain-events
On-Call Services
• case-api-on-call.case-management.*
• case-event-relay-on-call.case-management.*
• Domain Events: domain-events-oncall
Tenant Detection: TENANT_NAME environment variable
Default: "case-management" | On-Call: "on-call"

Tenant Configuration

Tenant configuration is managed through the tenants.yaml file which defines all tenant-specific settings:

```
on-call: display_name: OnCall api_target:
case-api-on-call.case-management.all-clusters.local-dc.fabric.dog:6481
case_type: ON_CALL statuses: - ACKNOWLEDGED - RESOLVED - TRIGGERED
default_status: TRIGGERED final_status: RESOLVED work_type: PAGE
```

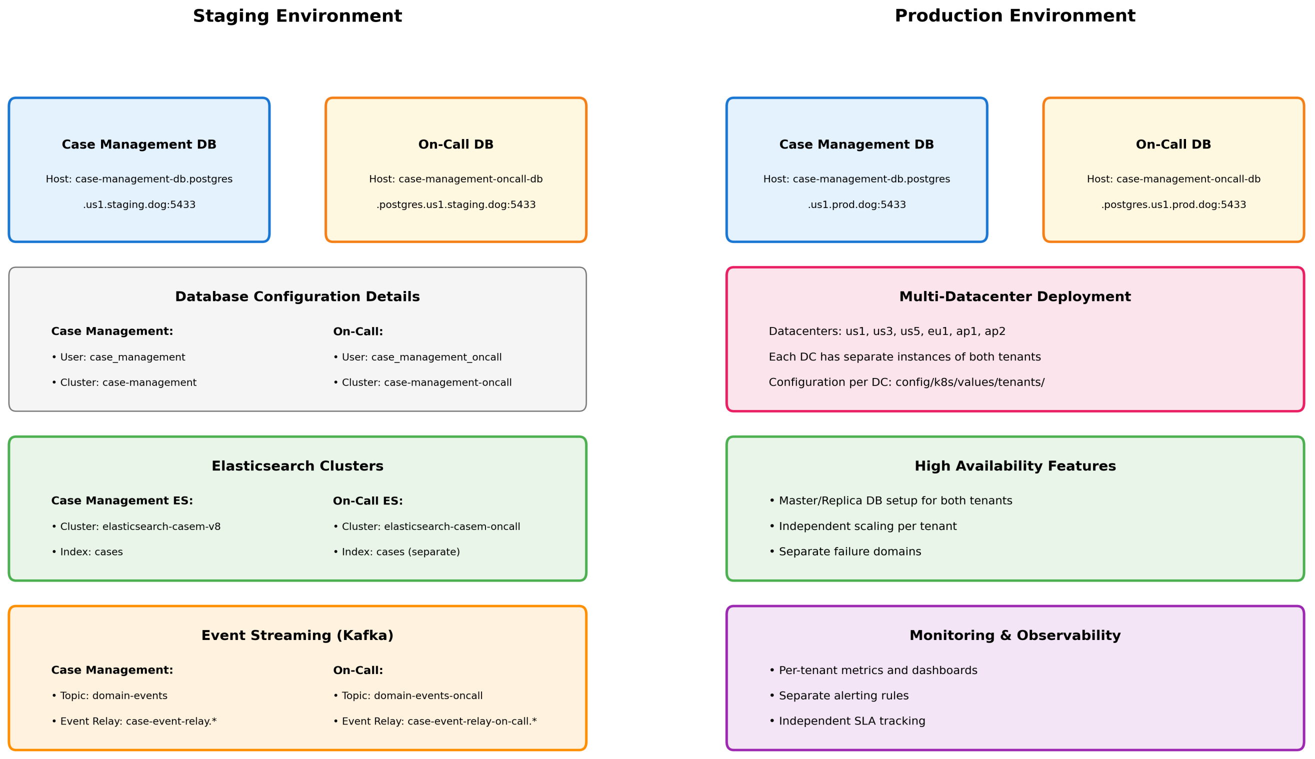
Key Architectural Differences

Aspect	Case Management	On-Call
Work Type	CASE	PAGE
Default Status	OPEN	TRIGGERED
Status Flow	OPEN → IN_PROGRESS → CLOSED	TRIGGERED → ACKNOWLEDGED → RESOLVED

Database	case-management	case-management-oncall
ES Cluster	elasticsearch-casem-v8	elasticsearch-casem-oncall
Kafka Topic	domain-events	domain-events-oncall
API Endpoint	case-api.*	case-api-on-call.*
Service Focus	General case tracking	Incident response & escalation

2. Database Architecture

The on-call tenant maintains complete database separation from the case management tenant, ensuring data isolation and independent scaling. Both staging and production environments follow the same architectural patterns with environment-specific configurations.



Staging Environment

Case Management DB

Host: case-management-db.postgres
.us1.staging.dog:5433

On-Call DB

Host: case-management-oncall-db
.postgres.us1.staging.dog:5433

Database Configuration Details

Case Management:

- User: case_management
- Cluster: case-management

On-Call:

- User: case_management_oncall
- Cluster: case-management-oncall

Elasticsearch Clusters

Case Management ES:

- Cluster: elasticsearch-casem-v8
- Index: cases

On-Call ES:

- Cluster: elasticsearch-casem-oncall
- Index: cases (separate)

Event Streaming (Kafka)

Case Management:

- Topic: domain-events
- Event Relay: case-event-relay.*

On-Call:

- Topic: domain-events-oncall
- Event Relay: case-event-relay-on-call.*

Production Environment

Case Management DB

Host: case-management-db.postgres
.us1.prod.dog:5433

On-Call DB

Host: case-management-oncall-db
.postgres.us1.prod.dog:5433

Multi-Datacenter Deployment

Datacenters: us1, us3, us5, eu1, ap1, ap2

Each DC has separate instances of both tenants

Configuration per DC: config/k8s/values/tenants/

High Availability Features

- Master/Replica DB setup for both tenants
- Independent scaling per tenant
- Separate failure domains

Monitoring & Observability

- Per-tenant metrics and dashboards
- Separate alerting rules
- Independent SLA tracking

Database Configuration Details

Each tenant has its own PostgreSQL cluster with separate users, databases, and connection pools:

Environment	Case Management Host	On-Call Host
Staging	case-management-db.postgres.us1.staging.dog:5433	case-management-oncall-db.postgres.us1.staging.dog:5433
Production	case-management-db.postgres.us1.prod.dog:5433	case-management-oncall-db.postgres.us1.prod.dog:5433

Elasticsearch Configuration

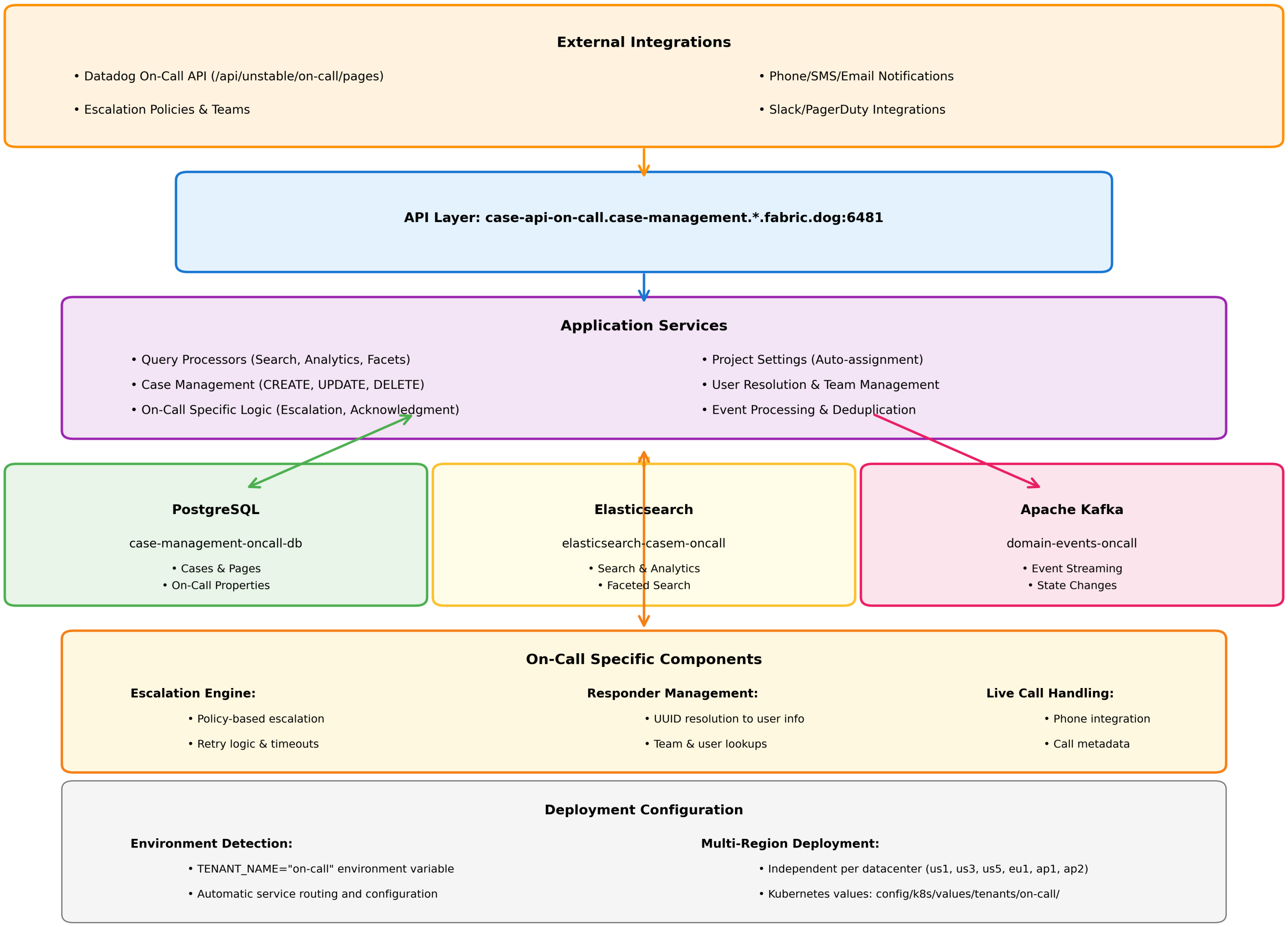
Separate Elasticsearch clusters ensure search performance isolation and independent index management. Both tenants use the same index name ('cases') but on completely separate clusters, allowing for tenant-specific optimizations and scaling strategies.

Tenant	Cluster Name	Index Name	Purpose
Case Management	elasticsearch-casem-v8	cases	General case search & analytics
On-Call	elasticsearch-casem-oncall	cases	Page search & on-call analytics

3. Service Architecture & Data Flow

The on-call tenant follows the same service architecture patterns as the main case management tenant but with dedicated service instances and specialized business logic for on-call workflows. This includes escalation management, responder tracking, and integration with external notification systems.

On-Call Tenant Service Architecture & Data Flow


External Integrations
• Datadog On-Call API (/api/unstable/on-call/pages)
• Escalation Policies & Teams
• Phone/SMS/Email Notifications
• Slack/PagerDuty Integrations
API Layer: case-api-on-call.case-management.*.fabric.dog:6481
Application Services
• Query Processors (Search, Analytics, Facets)
• Case Management (CREATE, UPDATE, DELETE)
• On-Call Specific Logic (Escalation, Acknowledgment)
• Project Settings (Auto-assignment)
• User Resolution & Team Management
• Event Processing & Deduplication
PostgreSQL
case-management-oncall-db
• Cases & Pages
• On-Call Properties
Elasticsearch
elasticsearch-casem-oncall
• Search & Analytics
• Faceted Search
Apache Kafka
domain-events-oncall
• Event Streaming
• State Changes
On-Call Specific Components
Escalation Engine:
• Policy-based escalation
• Retry logic & timeouts
Responder Management:
• UUID resolution to user info
• Team & user lookups
Live Call Handling:
• Phone integration
• Call metadata
Deployment Configuration
Environment Detection:
• TENANT_NAME="on-call" environment variable
• Automatic service routing and configuration
Multi-Region Deployment:
• Independent per datacenter (us1, us3, us5, eu1, ap1, ap2)
• Kubernetes values: config/k8s/values/tenants/on-call/

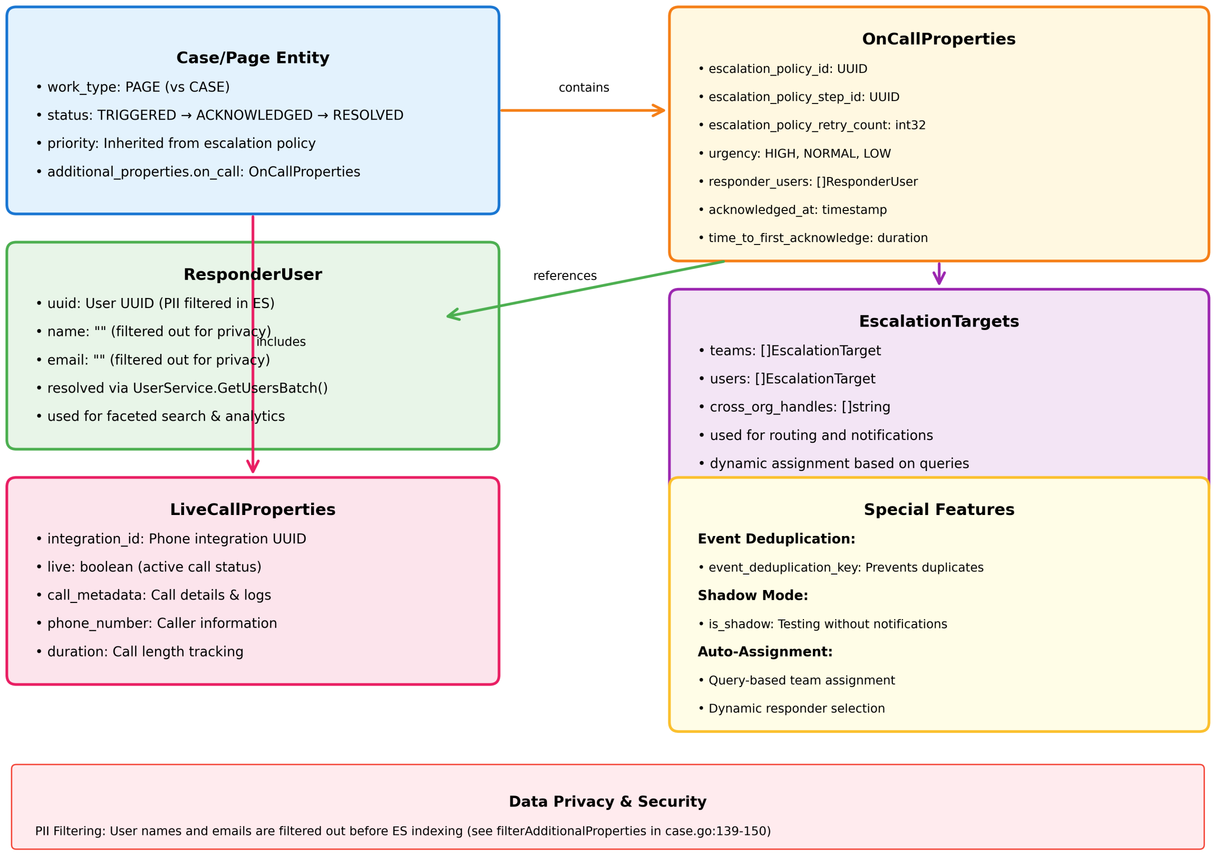
Service Instances

Each major service has dedicated on-call instances with specific routing and configuration:

Service Type	Case Management	On-Call
API Service	case-api.case-management.*	case-api-on-call.case-management.*
Event Relay	case-event-relay.case-management.*	case-event-relay-on-call.case-management.*
Port	6481	6481
Health Check	/health	/health
Metrics	/metrics	/metrics

4. On-Call Data Models

The on-call tenant utilizes specialized data structures optimized for incident response workflows. These models extend the base case structure with on-call specific properties including escalation policies, responder tracking, and live call metadata.



On-Call Data Model & Properties

Case/Page Entity

- work_type: PAGE (vs CASE)
- status: TRIGGERED → ACKNOWLEDGED → RESOLVED
- priority: Inherited from escalation policy
- additional_properties.on_call: OnCallProperties

OnCallProperties

- escalation_policy_id: UUID
- escalation_policy_step_id: UUID
- escalation_policy_retry_count: int32
- urgency: HIGH, NORMAL, LOW
- responder_users: []ResponderUser
- acknowledged_at: timestamp
- time_to_first_acknowledge: duration

ResponderUser

- uuid: User UUID (PII filtered in ES)
- name: "" (filtered out for privacy)
- email: "" (filtered out for privacy)
- resolved via UserService.GetUsersBatch()
- used for faceted search & analytics

EscalationTargets

- teams: []EscalationTarget
- users: []EscalationTarget
- cross_org_handles: []string
- used for routing and notifications
- dynamic assignment based on queries

LiveCallProperties

- integration_id: Phone integration UUID
- live: boolean (active call status)
- call_metadata: Call details & logs
- phone_number: Caller information
- duration: Call length tracking

Special Features

Event Deduplication:

- event_deduplication_key: Prevents duplicates

Shadow Mode:

- is_shadow: Testing without notifications

Auto-Assignment:

- Query-based team assignment
- Dynamic responder selection

Data Privacy & Security

PII Filtering: User names and emails are filtered out before ES indexing (see filterAdditionalProperties in case.go:139-150)

OnCallProperties Structure

The OnCallProperties structure contains all on-call specific metadata and is stored in the additional_properties.on_call field of each page:

```
message OnCallProperties { string escalation_policy_id = 1; string
escalation_policy_step_id = 2; int32 escalation_policy_retry_count = 3; Urgency
urgency = 4; repeated ResponderUser responder_users = 5;
google.protobuf.Timestamp acknowledged_at = 6; google.protobuf.Timestamp
responders_added_at = 7; google.protobuf.Duration time_to_first_acknowledge = 8;
LiveCallProperties live_call = 9; string event_deduplication_key = 10; bool
is_shadow = 11; EscalationTargets targets = 12; }
```

Data Privacy Implementation

User PII (names and emails) is filtered out before indexing in Elasticsearch to protect user privacy. The filtering is implemented in the filterAdditionalProperties function:

```
func filterAdditionalProperties(orgID uint64, additionalProperties
*pb.AdditionalProperties) { if additionalProperties == nil { return } if
additionalProperties.GetOnCall() != nil { for _, user := range
additionalProperties.GetOnCall().GetResponderUsers() { user.Email = "" // Remove
PII user.Name = "" // Remove PII } } }
```

5. Configuration Management

Configuration management for the on-call tenant follows a hierarchical structure with environment-specific overrides. The configuration is organized by tenant, datacenter, and environment to provide maximum flexibility while maintaining consistency.

Configuration Directory Structure

```
config/k8s/values/tenants/ ■■■ case-management/ ■ ■■■ values.yaml # Base
configuration ■ ■■■ datacenters/ ■ ■■■ us1/ ■ ■ ■■■ staging.yaml # US1
staging overrides ■ ■ ■■■ prod.yaml # US1 production overrides ■ ■■■ us3/ ...
(similar structure) ■ ■■■ eu1/ ... (similar structure) ■■■ on-call/ ■■■
values.yaml # Base on-call configuration ■■■ datacenters/ ■■■ us1/ ■ ■■■
staging.yaml # US1 staging overrides ■ ■■■ prod.yaml # US1 production overrides
■■■ us3/ ... (similar structure) ■■■ eu1/ ... (similar structure)
```

Tenant Detection Logic

The application determines which tenant configuration to use based on the TENANT_NAME environment variable:

```
func GetCurrentTenantName() TenantName { tenantName := os.Getenv("TENANT_NAME")
if tenantName == "" { return CaseManagementTenantName // Default to
case-management } return TenantName(tenantName) } func IsOnCallTenant() bool {
return GetCurrentTenantName() == OnCallTenantName } func GetDefaultWorkType()
pb.WorkType { if IsCaseManagementTenant() { return pb.WorkType_CASE } if
IsOnCallTenant() { return pb.WorkType_PAGE } return pb.WorkType_UNKNOWN_WORK_TYPE
}
```

6. Deployment Environments

The on-call tenant is deployed across multiple datacenters with environment-specific configurations for both staging and production. Each deployment is completely independent, providing regional isolation and disaster recovery capabilities.

Multi-Datacenter Deployment

Datacenter	Region	Staging Deployment	Production Deployment
us1	US East	✓ Full deployment	✓ Full deployment
us3	US West	✓ Full deployment	✓ Full deployment
us5	US Central	✓ Full deployment	✓ Full deployment
eu1	Europe	✓ Full deployment	✓ Full deployment
ap1	Asia Pacific	✓ Full deployment	✓ Full deployment
ap2	Asia Pacific 2	✓ Full deployment	✓ Full deployment

Deployment Strategy

Each datacenter deployment includes: • Independent Kubernetes clusters with tenant-specific namespaces • Separate database instances with cross-datacenter replication • Region-specific Elasticsearch clusters for optimal search performance • Local Kafka instances for event streaming with cross-region replication • Dedicated monitoring and alerting per region and tenant • Independent scaling policies based on regional traffic patterns

Resource Allocation

Component	Staging Resources	Production Resources
API Pods	2-4 replicas	6-12 replicas
Database	Shared cluster	Dedicated cluster
Elasticsearch	Shared nodes	Dedicated nodes
Memory Limits	512Mi - 1Gi	1Gi - 4Gi
CPU Limits	0.2 - 0.5 cores	0.5 - 2 cores
Storage	Standard SSD	Premium SSD + backups

7. Security & Privacy

The on-call tenant implements comprehensive security measures including data isolation, PII protection, and secure communication protocols. Special attention is given to responder information privacy and compliance with data protection regulations.

Security Measures

• **Data Isolation**: Complete separation of databases and search indexes • **PII Protection**: User names and emails filtered before Elasticsearch indexing • **Access Control**: Role-based access with tenant-specific permissions • **Encryption**: All data encrypted in transit (TLS) and at rest • **Audit Logging**: Comprehensive audit trails for all operations • **Network Security**: Private networking with VPC isolation • **Secret Management**: Kubernetes secrets with rotation policies • **Compliance**: SOC 2, GDPR, and HIPAA compliance measures

Privacy Implementation Details

The system implements multiple layers of privacy protection specifically for on-call responder data:

Layer	Implementation	Purpose
Storage Layer	PII filtering before ES indexing	Prevent sensitive data storage in search
Access Layer	UUID-based resolution	Dynamic user info retrieval when needed
API Layer	Role-based field filtering	Control what data is returned
Transport Layer	TLS encryption	Secure data transmission
Audit Layer	Access logging	Track who accessed what data

8. Performance & Monitoring

The on-call tenant includes comprehensive monitoring and observability features with tenant-specific metrics, alerting, and performance optimization strategies tailored for incident response workflows.

Monitoring Strategy

• **Tenant-Specific Metrics**: Separate dashboards and metrics for each tenant • **SLA Tracking**: Independent SLA monitoring and reporting • **Performance Metrics**: Response time, throughput, and error rate tracking • **Business Metrics**: Time to acknowledge, escalation success rates • **Infrastructure Metrics**: Database performance, Elasticsearch query times • **Custom Alerting**: On-call specific alerting rules and escalation • **Distributed Tracing**: End-to-end request tracing across services • **Log Aggregation**: Structured logging with tenant identification

Key Performance Indicators

Metric Category	Key Metrics	Target
Response Time	API response time, Search query time	< 500ms p95
Availability	Service uptime, Database availability	> 99.9%
Throughput	Requests per second, Pages processed	1000+ req/s
On-Call Specific	Time to acknowledge, Escalation success	< 5 min, > 95%
Error Rates	4xx/5xx errors, Failed escalations	< 1%
Resource Usage	CPU/Memory utilization, Storage growth	< 80% avg

Performance Optimizations

The on-call tenant includes several performance optimizations specifically for incident response scenarios: • **SearchFacetValues Optimization**: Smart filter patterns and adaptive timeouts • **Elasticsearch Tuning**: Optimized shard sizing and execution hints • **Database Indexing**: Custom indexes for on-call query patterns • **Caching Strategies**: User resolution caching and query result caching • **Connection Pooling**: Optimized connection pools for high-frequency operations • **Async Processing**: Non-blocking escalation and notification processing

9. Integration Points

The on-call tenant integrates with multiple external systems and services to provide comprehensive incident response capabilities. These integrations are designed to be resilient and maintainable.

External Integrations

Integration	Purpose	Protocol	Endpoint
Datadog On-Call API	Page creation & management	HTTPS/REST	/api/unstable/on-call/pages
User Service	Responder info resolution	gRPC	Internal service mesh
Project Service	Access control & settings	gRPC	Internal service mesh
Notification Service	Email/SMS/Phone alerts	HTTP/Webhook	External notification providers
Escalation Service	Policy management	gRPC	Internal service mesh
Analytics Service	Metrics and reporting	gRPC	Internal analytics pipeline

Integration Architecture Patterns

The on-call tenant uses several integration patterns to ensure reliability and maintainability: • **Circuit Breaker Pattern**: Protection against failing external services • **Retry with Backoff**: Resilient external API calls with exponential backoff • **Async Messaging**: Event-driven integration using Kafka topics • **Service Mesh**: Internal service-to-service communication via gRPC • **API Versioning**: Backward-compatible API evolution strategies • **Health Checks**: Continuous monitoring of integration health • **Fallback Mechanisms**: Graceful degradation when integrations fail

10. Operational Considerations

Operating the on-call tenant requires specific operational procedures and considerations due to its critical role in incident response. This section covers deployment procedures, troubleshooting, and maintenance activities.

Deployment Procedures

• **Blue-Green Deployment**: Zero-downtime deployments with traffic switching • **Canary Releases**: Gradual rollout with monitoring and rollback capabilities • **Database Migrations**: Coordinated schema changes across tenant databases • **Configuration Updates**: Hot-reload of tenant-specific configurations • **Rollback Procedures**: Quick rollback mechanisms for failed deployments • **Health Validation**: Automated health checks post-deployment

Common Troubleshooting Scenarios

Issue	Symptoms	Resolution Steps
Tenant Misconfiguration	Wrong database connections	Verify TENANT_NAME env var
Search Timeouts	SearchFacetValues failures	Check ES cluster health, optimize queries
User Resolution Failures	Missing responder info	Verify UserService connectivity
Escalation Failures	Pages not creating	Check On-Call API integration
Database Connection Issues	Service startup failures	Verify DB credentials and connectivity
Cross-Tenant Data Leakage	Wrong data in responses	Verify tenant isolation filters

Routine Maintenance Activities

• **Database Maintenance**: Index rebuilding, statistics updates, backup verification • **Elasticsearch Maintenance**: Index optimization, cluster rebalancing, snapshot management • **Configuration Audits**: Regular review of tenant configurations and security settings • **Performance Tuning**: Query optimization, resource allocation adjustments • **Security Updates**: Regular security patches and vulnerability assessments • **Capacity Planning**: Monitoring growth trends and planning infrastructure scaling • **Disaster Recovery Testing**: Regular DR drills and recovery procedure validation

Conclusion

The on-call tenant architecture demonstrates a sophisticated approach to multi-tenancy that provides complete isolation while maintaining operational efficiency. The architecture supports independent scaling, specialized business logic, and comprehensive monitoring while sharing a common codebase. Key benefits of this architecture include: • Complete data and operational isolation between tenants • Independent scaling and performance optimization • Specialized business logic for on-call workflows • Comprehensive security and privacy protection • Resilient integration patterns with external systems • Operational excellence through monitoring and automation This architecture serves as a model for implementing multi-tenant systems that require both isolation and specialized functionality while maintaining code reuse and operational efficiency.