Case-Event-Feed-Handler Technical Report

Executive Summary

The Case-Event-Feed-Handler is a critical streaming event processor in Datadog's case management domain that bridges workflow automation events with timeline creation and enables ServiceNow integration for maintenance window management. It consumes events from the Event Platform via Kafka streams and processes them to create timeline entries and synchronize maintenance windows across multiple datacenters.

Service Overview

Primary Functions

1. **Workflow Timeline Creation**: Processes workflow automation events (source ID 375) to create timeline entries for case executions
2. **ServiceNow Integration**: Handles ServiceNow change requests (source ID 105) to manage maintenance windows
3. **Event Stream Processing**: Consumes and processes events from Kafka streams with CBOR decoding
4. **Multi-Datacenter Deployment**: Deployed across staging, production, and government environments

Key Metrics

- **Event Sources**: 2 primary sources (Workflow Automation: 375, ServiceNow: 105)
- **Stream Processing**: Up to 1000 messages per pull with 100ms timeout
- **Memory Limits**: 500MB total, 4MB per message
- **Deployment Flavors**: 12+ different stream-specific configurations per datacenter

Architecture Overview

System Architecture

The system follows a stream processing architecture where: - **Event Platform** acts as the central Kafka-based message broker - **Case Event Feed Handler** consumes and processes events from multiple sources - **Case Management APIs** provide timeline and maintenance window services - **External Systems** like ServiceNow integrate via the Event Platform

Event Flow Architecture

The event flow demonstrates two main processing paths: 1. **Workflow Events** (Source 375): Stream to Timeline Service for case timeline creation 2. **ServiceNow Events** (Source 105): Stream to Maintenance Window Service for synchronization

Event Platform Integration

Stream Consumer Configuration

- **Consumer Group**: `case_event_feed_handler`
- **Kafka Bootstrap**: `event-config-loader-default.logs-general.all-clusters.local-dc.fabric.dog:9093`
- **Event Filter**: `evp.type == 'feed' and (feed.evt.source_id == 375 or feed.evt.source_id == 105)`
- **Batch Size**: 1000 messages per pull
- **Timeout**: 100ms per pull
- **Acknowledgment Mode**: Commit

Event Processing Pipeline

The event processing pipeline includes: 1. **Message Decoding**: CBOR decoding from Kafka messages 2. **Event Validation**: Source ID and event type validation 3. **Route Processing**: Workflow events → Timeline creation, ServiceNow events → Maintenance windows 4. **Error Handling**: Skip, retry, or discard based on error classification

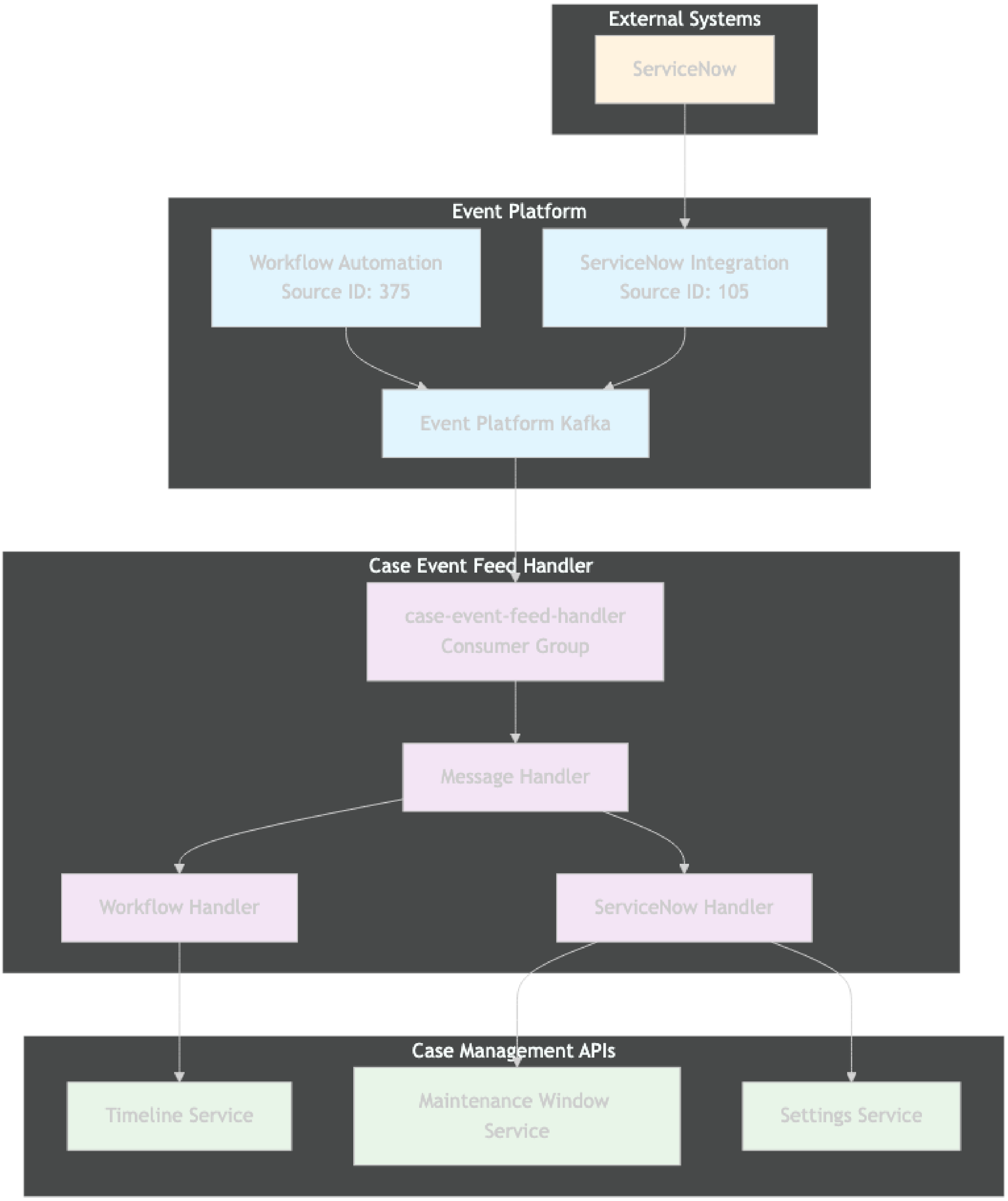

Figure 1: System Architecture

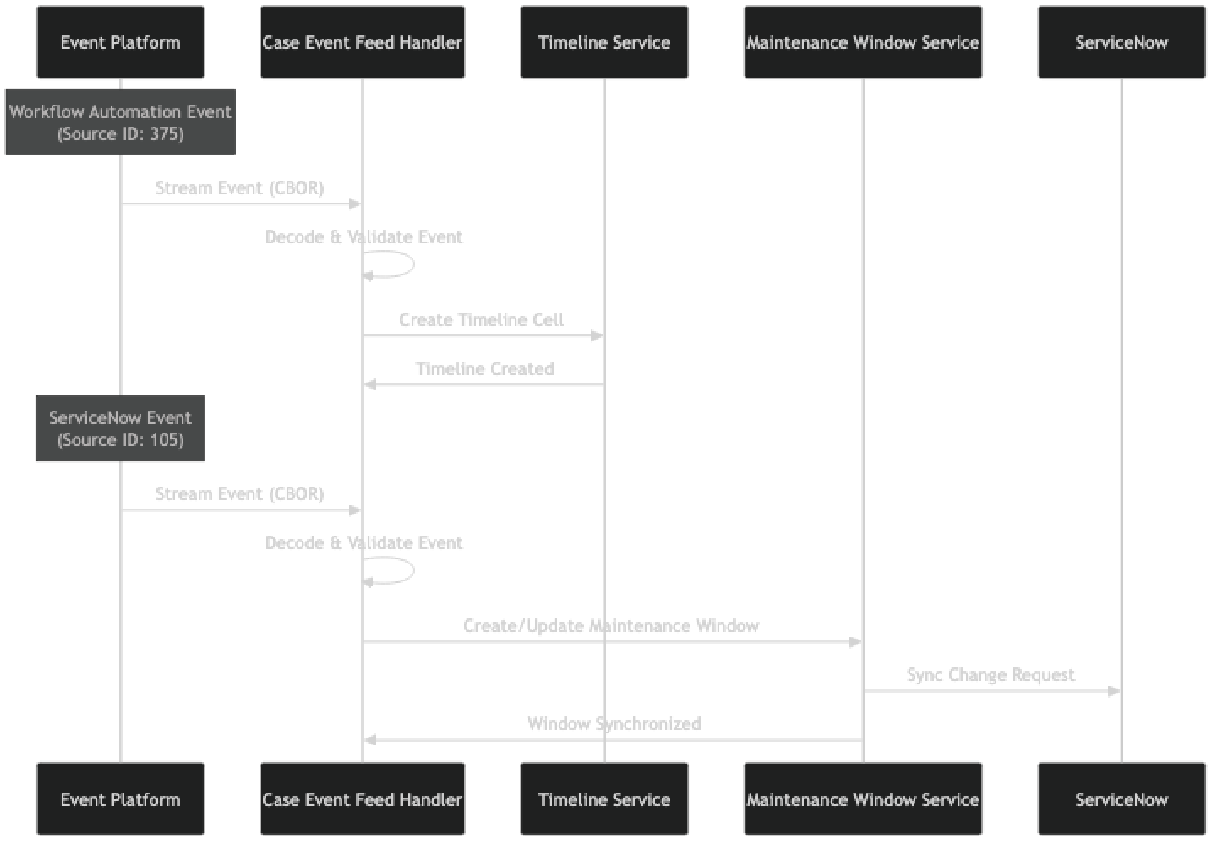

Figure 2: Event Flow

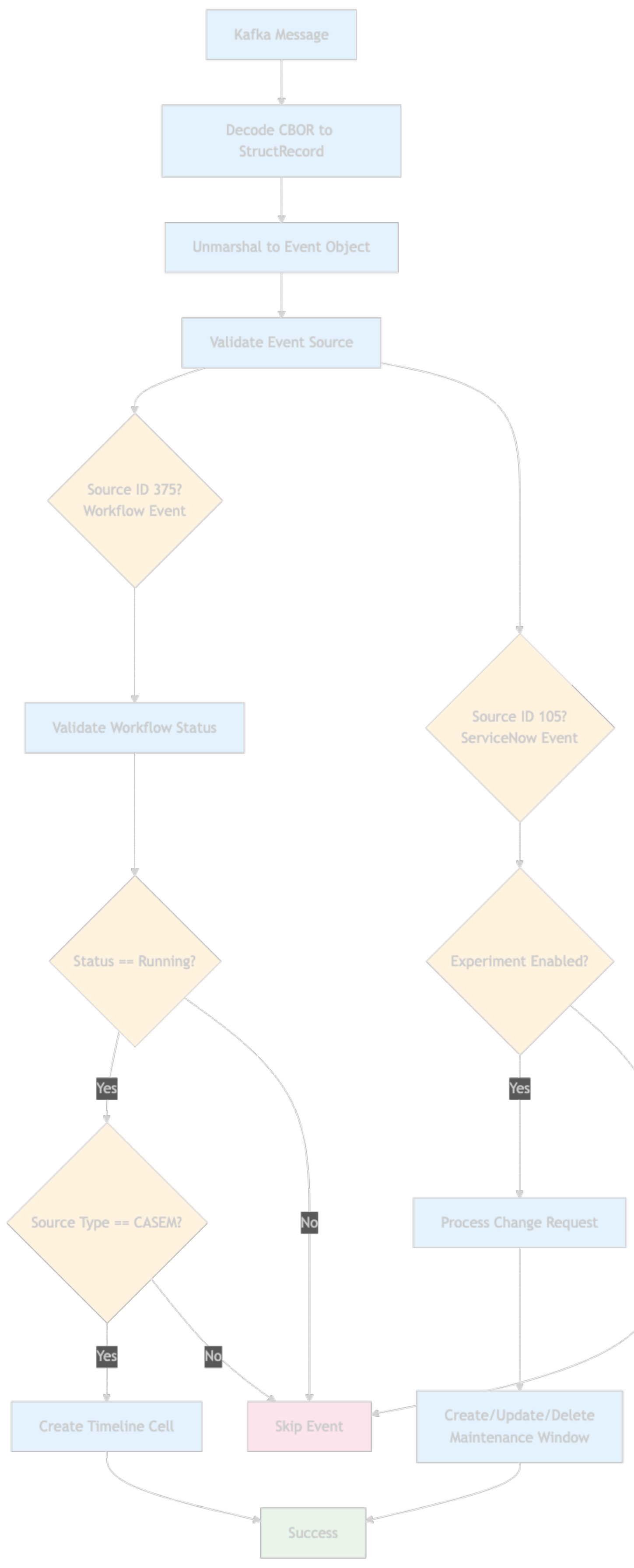

Figure 3: Event Processing Pipeline

Event Sources and Tracks

Source ID 375: Workflow Automation

- **Purpose**: Captures workflow execution events for case management
- **Event Type**: `feed`
- **Processing**: Creates timeline entries for completed workflow runs
- **Requirements**:
  - Workflow status must be `running`
  - Source type must be `casem` (case management)

Source ID 105: ServiceNow Integration

- **Purpose**: Synchronizes ServiceNow change requests with maintenance windows
- **Event Type**: `feed`
- **Processing**: Creates, updates, or deletes maintenance windows
- **Requirements**:
  - Must have `change_request` tag
  - Experiment `case-management-maintenance-window-servicenow-sync` must be enabled

Datacenter Deployment Strategy

Multi-Environment Deployment

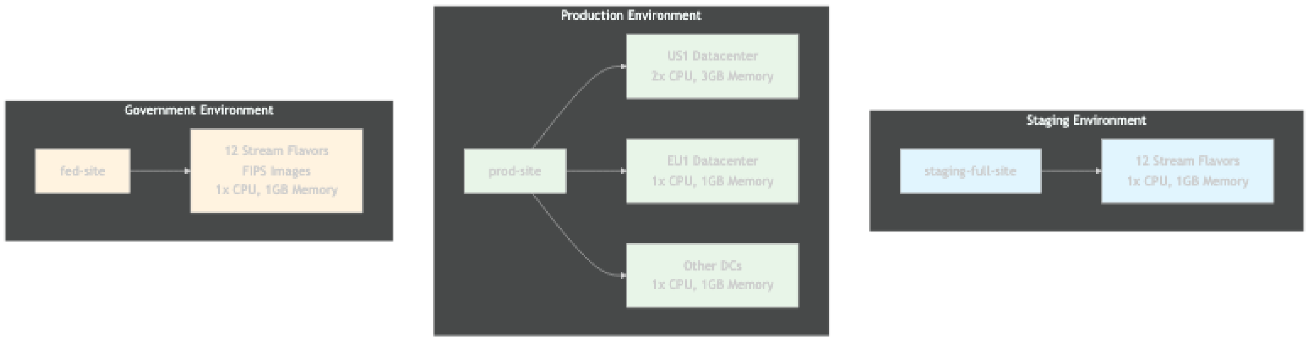

Figure 4: Deployment Strategy

The service deploys across three main environments: - **Staging**: Single site with all flavors for testing - **Production**: Multi-datacenter with resource optimization (US1 gets 2x resources) - **Government**: FIPS-compliant deployments for federal requirements

Deployment Flavors by Stream

Each datacenter deploys multiple flavors processing different event streams:

Stream-Based Flavors

- `all-datadog--structured--EVP`
- `other-datadog--structured--EVP`
- `shared-datadog--structured--EVP`

Feed-Specific Flavors

- `feed-cake--structured--EVP`
- `feed-flux--structured--EVP`
- `feed-pant--structured--EVP`
- `feed-soup--structured--EVP`
- `feed-volt--structured--EVP`

Shared Service Flavors

- `shared-kali--structured--EVP`
- `shared-snow--structured--EVP`
- `shared-taxi--structured--EVP`
- `shared-upon--structured--EVP`

Resource Allocation by Datacenter

Environment	Datacenter	CPU Request/Limit	Memory Request/Limit	Special Features
Staging	All	1 core	1GB/2GB	Parallel deployment
Production	US1	2 cores	3GB/3GB	High-traffic DC
Production	EU1	1 core	1GB/2GB	Standard allocation
Production	Other	1 core	1GB/2GB	Standard allocation
Government	All	1 core	1GB/2GB	FIPS-compliant images

Autoscaling Configuration

WatermarkPodAutoscaler (WPA) Settings: - **Algorithm**: Absolute scaling - **Min Replicas**: 2 - **Max Replicas**: 10 - **Upscale Window**: 60 seconds - **Downscale Window**: 600 seconds (10 minutes) - **CPU Tolerance**: 100m - **Target Watermarks**: Low 35%, High 50% - **Stream-based scaling**: Uses logs-admin API for stream-aware autoscaling

ServiceNow Integration Details

Maintenance Window Management

The ServiceNow handler provides comprehensive maintenance window lifecycle management:

Change Request Processing Flow The ServiceNow integration follows this state machine:

1. **Event Received**: ServiceNow change request event arrives
2. **Experiment Check**: Validates `case-management-maintenance-window-servicenow-sync` experiment
3. **Filter Check**: Applies org-specific filters with variable substitution
4. **Action Determination**: Determines create/update/delete action
5. **API Operations**: Calls Maintenance Window Service API
6. **Success/Skip**: Completes processing or skips filtered events

Filter System The ServiceNow integration supports sophisticated filtering:

- **Variable Substitution**: Dynamic filter values from change request attributes
- **Conditional Logic**: AND/OR operations for complex filtering rules
- **State Validation**: Checks change request states and types
- **Configurable Rules**: Org-specific filter configurations via Settings API

Performance and Observability

Key Metrics Collected

Event Processing Metrics

- `case_event_feed_handler.decode.record.struct.{success/error}` - CBOR decoding metrics by org
- `case_event_feed_handler.decode.event.{success/error}` - Event unmarshaling metrics by org
- `case_event_feed_handler.handle.event.{success/skip/retry/discard}` - Event handling outcomes by org

Latency Metrics

- `case_event_feed_handler.latency` - Event processing latency distribution by org and outcome
- Measured from event timestamp to processing completion

Business Metrics

- `case_event_feed_handler.create.workflow_executed.timeline.cell.{success/error}` - Timeline creation metrics by org and workflow ID

Error Handling Strategy

The error handling system classifies errors into three categories:

Error Classification

- **Skip Errors**: Experiment disabled, wrong workflow status/type → Log and continue
- **Retry Errors**: Retriable API failures, temporary service unavailability → Crash and restart service
- **Discard Errors**: Permanent failures, malformed events → Log and drop event

This approach ensures: - **No Data Loss**: Retry errors cause service restart for reprocessing - **Graceful Handling**: Skip errors don't block processing - **Fault Isolation**: Discard errors prevent poison messages

Security and Compliance

Security Features

- **Service Account**: Uses case-management shared service account with RBAC
- **Network Policies**: Cilium network policies for service isolation
- **TLS**: All API communications use TLS encryption
- **FIPS Compliance**: Government deployments use FIPS-compliant container images

Configuration Management

- **Consul Integration**: Dynamic configuration via consul-template
- **Secret Management**: Integrated with Datadog's secret management system
- **Environment Isolation**: Separate configurations per environment

Monitoring and Alerting

Health Checks

- **Liveness Probe**: `/readiness` endpoint on port 8080
- **Readiness Probe**: Same endpoint with 10s initial delay, 3s period
- **Failure Threshold**: 3 consecutive failures trigger restart

Observability Stack

- **Tracing**: Distributed tracing with dd-trace-go
- **Metrics**: StatsD metrics to Datadog Agent
- **Logging**: Structured logging with contextual information
- **Dashboards**: Environment-specific monitoring dashboards

Technical Implementation Details

Core Components

Main Entry Point (`main.go`)

- **Application Bootstrap**: Uses ddapp framework for service initialization
- **Stream Consumer Setup**: Configures libstreaming with specific event filters

- **API Client Initialization**: Sets up Case API clients for timeline and maintenance window operations
- **Health Check Service**: Provides liveness/readiness endpoints on port 8080

Message Handler (`handler.go`)

- **Event Loop**: Pulls batches of 1000 messages with 100ms timeout
- **CBOR Decoding**: Converts Kafka messages to structured records
- **Event Routing**: Routes events to appropriate processors based on source ID
- **Retry Logic**: Implements comprehensive error classification and retry mechanisms

ServiceNow Handler (`servicenow_handler.go`)

- **Maintenance Window Lifecycle**: Create/update/delete operations
- **Filter Engine**: Configurable filtering with variable substitution
- **Schedule Extraction**: Parses change request metadata for scheduling information
- **API Integration**: Interfaces with Maintenance Window and Settings services

Build and Deployment Configuration

Build System (`BUILD.bazel`)

- **Go Binary**: Profile-Guided Optimization (PGO) in release builds
- **Docker Images**: Multi-architecture support (linux-amd64/arm64)
- **FIPS Compliance**: Special FIPS-compliant images for government deployments
- **Integration Tests**: Comprehensive test suite with fixtures

Kubernetes Configuration

- **Helm Chart**: Version 0.0.5 with Datadog helpers dependency
- **Multi-Environment**: Staging, production, and government configurations
- **Resource Optimization**: Environment-specific resource allocation
- **Autoscaling**: WatermarkPodAutoscaler with stream-aware scaling

Future Considerations

Scalability Enhancements

- **Stream-Specific Scaling**: Enhanced autoscaling based on individual stream metrics
- **Multi-Region Deployment**: Geographic distribution for reduced latency
- **Queue Depth Monitoring**: More sophisticated backpressure handling

Feature Extensions

- **Additional Event Sources**: Support for new workflow and integration sources
- **Enhanced Filtering**: More sophisticated ServiceNow filter capabilities
- **Real-Time Analytics**: Live event processing metrics and alerting

Operational Improvements

- **Automated Validation**: Deployment validation with synthetic tests
- **Enhanced Recovery**: Improved error recovery and replay mechanisms
- **Performance Optimization**: Stream-specific performance tuning

Conclusion

The Case-Event-Feed-Handler serves as a critical bridge between Datadog's Event Platform and Case Management system, processing workflow automation events and ServiceNow integrations with high reliability and scalability. Its multi-datacenter deployment strategy, comprehensive error handling, and sophisticated ServiceNow integration make it a robust component of the case management infrastructure.

The service's architecture enables: - **High Throughput**: Processing thousands of events per second across multiple streams - **Reliable Processing**: Comprehensive error classification prevents data loss - **Global Scale**: Multi-datacenter deployment with environment-specific optimizations - **Regulatory Compliance**: FIPS-compliant deployments for government requirements

---

This report provides a comprehensive technical overview of the Case-Event-Feed-Handler service, its integration with the Event Platform, and its deployment architecture across Datadog's global infrastructure.