To transition your monolithic **ITSMDeskPro** platform into a fully functional, resilient microservices architecture, we can break it down logically by domain boundaries (Bounded Contexts).
Based on your existing setup (including controllers, service boundaries, entities, and security), the system splits naturally into **7 distinct Microservices** and **3 Infrastructure services**. We will use **Spring Cloud OpenFeign** for inter-service communication and standard application.properties configuration for each.

Part 1: List of Microservices & Infrastructure Component Roles

1. **Config Server & Service Registry (infra-service-registry)**: Centralized Eureka discovery registry so services can locate and communicate with each other using OpenFeign.
2. **API Gateway (api-gateway)**: Entry point for all clients. Handles central routing, rate limiting, and global cross-origin resource sharing (CORS).
3. **Identity & Auth Service (auth-service)**: Handles user registration, authentication, JWT generation, validation, and administrative user profile management.
4. **Incident & SLA Service (incident-service)**: Responsible for logging incidents, escalating tickets, managing workflows, handling resolution processes, and tracking end-user satisfaction ratings.
5. **Problem & Known Error Service (problem-service)**: Focuses on deep-dive root cause investigations, logging problem records, and maintaining the Known Error Database (KEDB).
6. **Change Management Service (change-service)**: Handles Change Requests (RFC), managing the Change Advisory Board (CAB) review tracking, and recording implementation outcomes.
7. **Service Catalog & Request Service (catalog-service)**: Governs the user-facing service offerings catalog and handles individual standard service fulfillment lifecycle requests.
8. **Asset & CMDB Service (asset-cmdb-service)**: Tracks structural hardware assets, handles configuration items (CIs), models dependency relationships, and manages software license seats.
9. **Notification Service (notification-service)**: Aggregates event-driven or reactive communication logs to alert individual users about system status updates or actions.
10. **Analytics & Audit Service (analytics-audit-service)**: Generates reporting snapshots, operational SLA metric performance data, and stores transactional audit trails for security profiling.

Part 2: Detailed Microservice-by-Microservice File & Folder Structures

Below is the complete, exact file structure for every single service. Every file from your monolith has been carefully assigned to its respective architectural domain package matching your code layout.

1. Discovery & Registration Base (infra-service-registry)

```
infra-service-registry/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── registry/
        │               └── ServiceRegistryApplication.java
        └── resources/
            └── application.properties
```

2. Central Entry Routing Core (api-gateway)

```
api-gateway/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── gateway/
        │               └── ApiGatewayApplication.java
        └── resources/
            └── application.properties
```

3. Identity & User Profile Boundary (auth-service)

```
auth-service/
├── pom.xml
└── src/
    ├── main/
    │   ├── java/
    │   │   └── com/
    │   │       └── itsm/
    │   │           └── deskpro/
    │   │               ├── AuthApplication.java
    │   │               ├── config/
    │   │               │   ├── DataSeeder.java
    │   │               │   ├── OpenApiConfig.java
    │   │               │   └── SecurityConfig.java
    │   │               ├── controller/
    │   │               │   ├── AuthController.java
    │   │               │   └── UserController.java
    │   │               ├── dto/
    │   │               │   ├── request/
    │   │               │   │   ├── LoginRequest.java
    │   │               │   │   └── RegisterRequest.java
    │   │               │   └── response/
    │   │               │       └── AuthResponse.java
    │   │               ├── entity/
    │   │               │   └── User.java
    │   │               ├── enums/
    │   │               │   ├── Role.java
    │   │               │   └── UserStatus.java
    │   │               ├── exception/
    │   │               │   ├── BadRequestException.java
    │   │               │   ├── GlobalExceptionHandler.java
    │   │               │   └── ResourceNotFoundException.java
    │   │               ├── repository/
    │   │               │   └── UserRepository.java
    │   │               ├── security/
    │   │               │   ├── JwtAuthFilter.java
    │   │               │   ├── JwtUtils.java
    │   │               │   └── UserDetailsServiceImpl.java
```

```
│   │           └── service/
│   │               ├── AuthService.java
│   │               ├── UserService.java
│   │               └── impl/
│   │                   ├── AuthServiceImpl.java
│   │                   └── UserServiceImpl.java
│   └── resources/
│       └── application.properties
└── test/
    └── java/
        └── com/
            └── itsm/
                └── deskpro/
                    ├── controller/
                    │   └── AuthControllerTest.java
                    └── service/
                        └── AuthServiceTest.java
```

4. Core Ticket & Workflow Pipeline (incident-service)

```
incident-service/
├── pom.xml
└── src/
    ├── main/
    │   ├── java/
    │   │   └── com/
    │   │       └── itsm/
    │   │           └── deskpro/
    │   │               ├── IncidentApplication.java
    │   │               ├── client/
    │   │               │   ├── AuditClient.java
    │   │               │   └── NotificationClient.java
    │   │               ├── config/
    │   │               │   ├── OpenApiConfig.java
    │   │               │   └── SecurityConfig.java
    │   │               ├── controller/
```

```
│   │           │   └── IncidentController.java
│   │           ├── dto/
│   │           │   └── request/
│   │           │       ├── EscalateRequest.java
│   │           │       ├── IncidentNoteRequest.java
│   │           │       ├── IncidentRequest.java
│   │           │       └── SatisfactionRequest.java
│   │           ├── entity/
│   │           │   ├── Incident.java
│   │           │   ├── IncidentNote.java
│   │           │   └── SatisfactionScore.java
│   │           ├── enums/
│   │           │   ├── IncidentCategory.java
│   │           │   ├── IncidentStatus.java
│   │           │   ├── NoteType.java
│   │           │   ├── Priority.java
│   │           │   └── Role.java
│   │           ├── exception/
│   │           │   ├── BadRequestException.java
│   │           │   ├── GlobalExceptionHandler.java
│   │           │   └── ResourceNotFoundException.java
│   │           ├── repository/
│   │           │   ├── IncidentNoteRepository.java
│   │           │   ├── IncidentRepository.java
│   │           │   └── SatisfactionScoreRepository.java
│   │           ├── security/
│   │           │   ├── JwtAuthFilter.java
│   │           │   ├── JwtUtils.java
│   │           │   └── UserDetailsServiceImpl.java
│   │           └── service/
│   │               ├── IncidentService.java
│   │               └── impl/
│   │                   └── IncidentServiceImpl.java
│   └── resources/
│       └── application.properties
└── test/
    └── java/
        └── com/
            └── itsm/
                └── deskpro/
                    ├── controller/
                    │   └── IncidentControllerTest.java
                    └── service/
```

```
                    └── IncidentServiceTest.java
```

5. KEDB & Root Cause Domain (problem-service)

```
problem-service/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── deskpro/
        │               ├── ProblemApplication.java
        │               ├── config/
        │               │   ├── OpenApiConfig.java
        │               │   └── SecurityConfig.java
        │               ├── controller/
        │               │   └── ProblemController.java
        │               ├── dto/
        │               │   └── request/
        │               │       ├── KnownErrorRequest.java
        │               │       └── ProblemRequest.java
        │               ├── entity/
        │               │   ├── KnownError.java
        │               │   └── ProblemRecord.java
        │               ├── enums/
        │               │   ├── KnownErrorStatus.java
        │               │   ├── ProblemPriority.java
        │               │   ├── ProblemStatus.java
        │               │   └── Role.java
        │               ├── exception/
        │               │   ├── BadRequestException.java
        │               │   ├── GlobalExceptionHandler.java
        │               │   └── ResourceNotFoundException.java
        │               ├── repository/
        │               │   ├── KnownErrorRepository.java
```

```
│           │   └── ProblemRecordRepository.java
│           ├── security/
│           │   ├── JwtAuthFilter.java
│           │   ├── JwtUtils.java
│           │   └── UserDetailsServiceImpl.java
│           └── service/
│               ├── ProblemService.java
│               └── impl/
│                   └── ProblemServiceImpl.java
└── resources/
    └── application.properties
```

6. Formal Deployment Management Loop (change-service)

```
change-service/
├── pom.xml
└── src/
    ├── main/
    │   ├── java/
    │   │   └── com/
    │   │       └── itsm/
    │   │           └── deskpro/
    │   │               ├── ChangeApplication.java
    │   │               ├── config/
    │   │               │   ├── OpenApiConfig.java
    │   │               │   └── SecurityConfig.java
    │   │               ├── controller/
    │   │               │   └── ChangeManagementController.java
    │   │               ├── dto/
    │   │               │   └── request/
    │   │               │       ├── CABReviewRequest.java
    │   │               │       ├── ChangeRequestRequest.java
    │   │               │       └── ImplementationRequest.java
    │   │               ├── entity/
    │   │               │   ├── CABReview.java
    │   │               │   ├── ChangeImplementation.java
```

```
│   │           │   └── ChangeRequest.java
│   │           ├── enums/
│   │           │   ├── CABDecision.java
│   │           │   ├── ChangeStatus.java
│   │           │   ├── ChangeType.java
│   │           │   ├── ImplementationOutcome.java
│   │           │   ├── ReviewStatus.java
│   │           │   ├── RiskLevel.java
│   │           │   └── Role.java
│   │           ├── exception/
│   │           │   ├── BadRequestException.java
│   │           │   ├── GlobalExceptionHandler.java
│   │           │   └── ResourceNotFoundException.java
│   │           ├── repository/
│   │           │   ├── CABReviewRepository.java
│   │           │   ├── ChangeImplementationRepository.java
│   │           │   └── ChangeRequestRepository.java
│   │           ├── security/
│   │           │   ├── JwtAuthFilter.java
│   │           │   ├── JwtUtils.java
│   │           │   └── UserDetailsServiceImpl.java
│   │           └── service/
│   │               ├── ChangeManagementService.java
│   │               └── impl/
│   │                   └── ChangeManagementServiceImpl.java
│   └── resources/
│       └── application.properties
└── test/
    └── java/
        └── com/
            └── itsm/
                └── deskpro/
                    └── service/
                        └── ChangeManagementServiceTest.java
```

7. Catalog Request & Fulfillment Domain (catalog-service)

```
catalog-service/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── deskpro/
        │               ├── CatalogApplication.java
        │               ├── config/
        │               │   ├── OpenApiConfig.java
        │               │   └── SecurityConfig.java
        │               ├── controller/
        │               │   └── ServiceCatalogController.java
        │               ├── dto/
        │               │   └── request/
        │               │       ├── CatalogItemRequest.java
        │               │       └── ServiceRequestRequest.java
        │               ├── entity/
        │               │   ├── ServiceCatalogItem.java
        │               │   └── ServiceRequest.java
        │               ├── enums/
        │               │   ├── CatalogCategory.java
        │               │   ├── Role.java
        │               │   └── ServiceRequestStatus.java
        │               ├── exception/
        │               │   ├── BadRequestException.java
        │               │   ├── GlobalExceptionHandler.java
        │               │   └── ResourceNotFoundException.java
        │               ├── repository/
        │               │   ├── ServiceCatalogItemRepository.java
        │               │   └── ServiceRequestRepository.java
        │               ├── security/
        │               │   ├── JwtAuthFilter.java
        │               │   ├── JwtUtils.java
        │               │   └── UserDetailsServiceImpl.java
        │               └── service/
        │                   ├── ServiceCatalogService.java
        │                   └── impl/
        │                       └── ServiceCatalogServiceImpl.java
        └── resources/
            └── application.properties
```

8. Topology & License Inventory (asset-cmdb-service)

```
asset-cmdb-service/
├── pom.xml
└── src/
    ├── main/
    │   ├── java/
    │   │   └── com/
    │   │       └── itsm/
    │   │           └── deskpro/
    │   │               ├── AssetApplication.java
    │   │               ├── config/
    │   │               │   ├── OpenApiConfig.java
    │   │               │   └── SecurityConfig.java
    │   │               ├── controller/
    │   │               │   └── AssetController.java
    │   │               ├── dto/
    │   │               │   └── request/
    │   │               │       ├── AssetRequest.java
    │   │               │       ├── ConfigItemRequest.java
    │   │               │       └── LicenseRequest.java
    │   │               ├── entity/
    │   │               │   ├── AssetRecord.java
    │   │               │   ├── ConfigurationItem.java
    │   │               │   └── SoftwareLicense.java
    │   │               ├── enums/
    │   │               │   ├── AssetStatus.java
    │   │               │   ├── AssetType.java
    │   │               │   ├── CIStatus.java
    │   │               │   ├── Environment.java
    │   │               │   ├── LicenseStatus.java
    │   │               │   ├── LicenseType.java
    │   │               │   └── Role.java
    │   │               ├── exception/
    │   │               │   ├── BadRequestException.java
```

```
│   │           │   ├── GlobalExceptionHandler.java
│   │           │   └── ResourceNotFoundException.java
│   │           ├── repository/
│   │           │   ├── AssetRecordRepository.java
│   │           │   ├── ConfigurationItemRepository.java
│   │           │   └── SoftwareLicenseRepository.java
│   │           ├── security/
│   │           │   ├── JwtAuthFilter.java
│   │           │   ├── JwtUtils.java
│   │           │   └── UserDetailsServiceImpl.java
│   │           └── service/
│   │               ├── AssetService.java
│   │               └── impl/
│   │                   └── AssetServiceImpl.java
│   └── resources/
│       └── application.properties
└── test/
    └── java/
        └── com/
            └── itsm/
                └── deskpro/
                    └── service/
                        └── AssetServiceTest.java
```

9. Alerting Event Processing Queue (notification-service)

```
notification-service/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── deskpro/
        │               ├── NotificationApplication.java
        │               ├── config/
```

```
│           │   ├── OpenApiConfig.java
│           │   └── SecurityConfig.java
│           ├── controller/
│           │   └── NotificationController.java
│           ├── entity/
│           │   └── Notification.java
│           ├── enums/
│           │   ├── NotificationCategory.java
│           │   ├── NotificationStatus.java
│           │   └── Role.java
│           ├── exception/
│           │   ├── BadRequestException.java
│           │   ├── GlobalExceptionHandler.java
│           │   └── ResourceNotFoundException.java
│           ├── repository/
│           │   └── NotificationRepository.java
│           ├── security/
│           │   ├── JwtAuthFilter.java
│           │   ├── JwtUtils.java
│           │   └── UserDetailsServiceImpl.java
│           └── service/
│               ├── NotificationService.java
│               └── impl/
│                   └── NotificationServiceImpl.java
└── resources/
    └── application.properties
```

10. Ledger Tracking & Dashboard Snapshot (analytics-audit-service)

```
analytics-audit-service/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
```

```
│       └── deskpro/
│           ├── AnalyticsAuditApplication.java
│           ├── config/
│           │   ├── OpenApiConfig.java
│           │   └── SecurityConfig.java
│           ├── controller/
│           │   ├── AuditLogController.java
│           │   └── ReportController.java
│           ├── entity/
│           │   ├── AuditLog.java
│           │   └── ITSMReport.java
│           ├── enums/
│           │   ├── ReportScope.java
│           │   └── Role.java
│           ├── exception/
│           │   ├── BadRequestException.java
│           │   ├── GlobalExceptionHandler.java
│           │   └── ResourceNotFoundException.java
│           ├── repository/
│           │   ├── AuditLogRepository.java
│           │   └── ITSMReportRepository.java
│           ├── security/
│           │   ├── JwtAuthFilter.java
│           │   ├── JwtUtils.java
│           │   └── UserDetailsServiceImpl.java
│           └── service/
│               ├── AuditService.java
│               ├── ReportService.java
│               └── impl/
│                   ├── AuditServiceImpl.java
│                   └── ReportServiceImpl.java
└── resources/
    └── application.properties
```

To implement centralized configuration in your microservices platform, you need a **Spring Cloud Config Server**. In a standard production setup, this is often separated from the Eureka Service Registry, or they can run as twin infrastructure blocks.

Here is the exact file layout, role, and configuration file contents for your **Central Config Server**, followed by how your other microservices use their application.properties to read from it.

1. Central Config Server (infra-config-server)

The Config Server acts as a central repository for all microservices' application.properties. It can fetch properties from a local classpath folder, a Git repository, or a file system.

Project Structure:

None

```
infra-config-server/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── com/
        │       └── itsm/
        │           └── config/
        │               └── ConfigServerApplication.java
        └── resources/
            ├── application.properties
            └── shared/
                ├── auth-service.properties
                ├── incident-service.properties
                ├── problem-service.properties
                ├── change-service.properties
                ├── catalog-service.properties
                ├── asset-cmdb-service.properties
                ├── notification-service.properties
                └── analytics-audit-service.properties
```

ConfigServerApplication.java

None

```
package com.itsm.config;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {
    public static void main(String[] args) {
```

```
        SpringApplication.run(ConfigServerApplication.class, args);
    }
}
```

Config Server's own application.properties

This configures the server to run on port 8888 and host the configuration files natively from its own classpath folder (src/main/resources/shared):

None

```
spring.application.name=infra-config-server
server.port=8888

# Using native (file system / classpath) profile instead of Git for simplicity
spring.profiles.active=native
spring.cloud.config.server.native.search-locations=classpath:/shared

# Register itself with the Eureka Discovery Server
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

2. How the Client Services Connect to the Config Server

When your individual services (like auth-service or incident-service) startup, they need to query this Config Server to fetch their database credentials, JWT secrets, and port allocations.

In Spring Boot 3.x, client services utilize their local application.properties to locate the config server using **Config First Bootstrap**:

Example: Client-Side application.properties **(for** auth-service**)**

Instead of putting all database setups locally inside the service, the local application.properties is stripped down to just connection pointers:

None

```
spring.application.name=auth-service
spring.profiles.active=dev

# Inform Spring Boot to import configuration from the Config Server
spring.config.import=optional:configserver:http://localhost:8888
```

```
# Eureka client config to discover other services later
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

3. Example Shared Property File Hosted on the Config Server

Inside the Config Server directory src/main/resources/shared/auth-service.properties, you would place the full functional properties extracted from your original monolith:

None

```
server.port=8081

# Database Configuration (Isolated schema for auth)
spring.datasource.url=jdbc:mysql://localhost:3306/itsm_auth?createDatabaseIfNotExist=true&useSSL=false&serverTimezone=UTC
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root

# JPA / Hibernate parameters
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

# JWT Security parameters
app.jwt.secret=ITSMDeskProSuperSecretKey2024MustBe256BitsLongForHS256Algorithm!!
app.jwt.expiration-ms=86400000

# Documentation settings
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger-ui.html
```

Benefits of adding this Config Server:

- **Zero-Downtime Property Changes**: You can modify variables (like token expiration times or database passwords) inside the config server's properties files, call a /actuator/refresh endpoint on your microservice, and it updates instantly without rebuilding or restarting the target service.
- **Separation of Concerns**: Your individual microservices packages remain perfectly clean, housing code logic instead of hardcoded system environments.