A Generative AI-Powered IT Service Management Solution for Enhanced Efficiency

This report outlines a comprehensive solution designed to enhance the efficiency and effectiveness of IT service delivery. The proposed system leverages Amazon Bedrock AgentCore, Kiro IDE, and other AWS services to automate critical ITIL-aligned processes such as incident correlation, proactive health monitoring, and performance reporting. By building autonomous agents that can analyze patterns, execute remediation tasks, and generate actionable insights, this solution aims to significantly improve technician productivity, optimize time management, and accelerate service request fulfillment. The following sections detail the core idea, its unique value proposition, the underlying technology stack, architectural design, use case scenarios, and key performance metrics, providing a complete blueprint for submission to a hackathon or for development into a production-ready prototype.

Core Idea: An Autonomous IT Operations Agent

The core concept of this solution is the creation of an intelligent, autonomous agent designed to function as a "digital teammate" for IT operations teams [7]. This agent operates on the Amazon Bedrock AgentCore framework, leveraging the power of Large Language Models (LLMs) to understand complex operational data, identify patterns, and take action without continuous human intervention [1] [7]. Its primary goal is to address common pain points in IT service delivery by automating manual, repetitive, and time-consuming tasks. The agent is engineered to work within an existing ITSM ecosystem, not to replace it, but to augment it with generative AI capabilities for proactive problem resolution and enhanced decision-making. It is built upon the foundation of ITIL standards, ensuring its outputs are meaningful and aligned with established best practices in incident and problem management [10].

The central functionality of the agent revolves around three key pillars. First, it provides Automated Incident Correlation, which intelligently groups related support tickets based on a combination of criteria including affected systems, user group categories, symptom descriptions, and historical data from previously resolved incidents [2] [4]. This directly tackles the issue of duplicate work and fragmented views of major problems, allowing technicians to focus on root cause analysis rather than ticket triage. Second, the agent performs Proactive Health Checks. By analyzing data from sources like Amazon CloudWatch, the agent runs nightly checks on infrastructure performance, identifying potential issues before they impact users. It then generates a concise, prioritized "top 3 issues to fix" list for the morning, enabling a proactive rather than reactive approach to maintenance [2] [4]. Third, it powers a Management Information System (MIS) Dashboard, which provides daily reports on Key Performance Indicators (KPIs) relevant to ITIL service fulfillment. This dashboard offers a unified view of service performance, tracking metrics such as response time, resolution time, first contact resolution rate, and user satisfaction scores, thereby improving transparency and accountability [10] [11]. This multi-faceted approach ensures the agent delivers tangible value across the entire IT service lifecycle, from initial request intake to strategic performance analysis.

Unique Value Proposition for ITIL-Based Organizations

The unique selling proposition (USP) of this AI-powered IT service management solution lies in its ability to transform traditional, process-heavy ITIL workflows into dynamic, intelligent, and predictive operations. Unlike many ITSM tools that simply digitize manual processes, this agent introduces a layer of cognitive automation that learns and adapts over time. Its primary USP is the significant enhancement of technician productivity by eliminating redundant work through automated ticket correlation [4]. By automatically grouping tickets with similar symptoms, affected systems, and impacted user groups, the agent prevents multiple technicians from working in parallel on the same underlying problem. This creates a single, unified source of truth for each major incident, streamlining communication and accelerating root cause analysis [2] [4]. This directly aligns with ITIL principles aimed at minimizing disruption and optimizing resource utilization.

Furthermore, the agent's capability for proactive health monitoring represents a paradigm shift from reactive troubleshooting to predictive maintenance. By generating a prioritized "top 3 issues to fix" list each morning, the agent equips IT teams with foresight, allowing them to address potential bottlenecks before they escalate into full-blown incidents [2] [4]. This proactive stance reduces unplanned downtime, improves overall service availability, and enhances the user experience, which is a key driver of User Satisfaction Scores [10]. The MIS dashboard further solidifies this value proposition by delivering on the promise of data-driven management. Instead of relying on manual report generation, managers receive a daily summary of critical ITIL metrics, providing clear visibility into service performance and highlighting areas for improvement [10]. This empowers organizations to make informed decisions about resource allocation, process improvements, and vendor management, ultimately driving down costs and increasing operational efficiency. The combination of these features—automated correlation, proactive alerts, and data-driven dashboards—creates a powerful tool that not only improves day-to-day efficiency but also elevates the strategic role of the IT department.

Feature	Traditional ITSM Approach	Proposed AI Agent Approach	Alignment with ITIL Standards
Incident Correlation	Manual grouping by service desk analysts; prone to human error and delay.	Automated grouping of related tickets based on system, symptom, and history. Reduces duplicate work [4].	Aligns with Incident & Problem Management practices for efficient problem isolation [10].
Health Monitoring	Reactive alerts after incidents occur; periodic manual checks.	Proactive overnight checks using CloudWatch data; generates a "top 3 issues" list for the next day [2] [4].	Supports Continual Improvement and Service Operation by preventing incidents before they happen [10].
Performance Reporting	Manual collection of data for weekly/	Automated daily MIS dashboard with real-time	Enables effective measurement and monitoring as per

Feature	Traditional ITSM Approach	Proposed AI Agent Approach	Alignment with ITIL Standards
	monthly reports; delayed insights.	KPIs (e.g., Resolution Time, FCR Rate) [10] [11].	Continual Improvement practice [10].
Remediation	Technicians follow static runbooks for known issues.	Agent uses Retrieval Augmented Generation (RAG) to create dynamic, context-specific runbooks from S3 documents [2].	Enhances Knowledge Management and supports faster resolutions.

This table illustrates how the proposed solution moves beyond simple automation to introduce intelligent augmentation, directly supporting and enhancing established ITIL frameworks.

Technology Stack: Leveraging Amazon Bedrock AgentCore and AWS Services

The proposed solution is architected entirely on a modern, scalable, and secure cloud-native stack, primarily utilizing Amazon Web Services (AWS). The choice of technologies is deliberate, focusing on the Amazon Bedrock AgentCore framework as the central nervous system for the AI agents, while integrating a suite of complementary AWS services to provide the necessary data storage, processing, security, and observability. This combination allows for the rapid development of sophisticated, enterprise-grade AI applications using the Kiro IDE for coding and orchestration [1] [9].

At the heart of the architecture is Amazon Bedrock AgentCore, a managed service designed for building, deploying, and scaling AI agents securely [1]. It provides several critical components. The AgentCore Runtime enables long-running, asynchronous workloads of up to eight hours, which is essential for tasks like overnight health checks or complex, multi-step incident investigations [1]. Persistent memory capabilities allow agents to maintain state across sessions, crucial for coordinating multi-agent systems [1]. For orchestration, the framework integrates seamlessly with popular libraries like LangGraph and CrewAI, which will be used to define the workflow between specialized agents [4] [9]. AgentCore also includes a Code Interpreter, a secure sandbox environment for executing code (Python, JavaScript, TypeScript) for data analysis and visualization, pre-installed with libraries like pandas and matplotlib [5]. For tool integration, the AgentCore Gateway acts as a serverless proxy that allows agents to discover and invoke external tools, such as AWS Lambda functions or REST APIs, using the Model Context Protocol (MCP) [6]. Finally, Observability is deeply integrated via Amazon CloudWatch, providing detailed metrics, logs, and distributed traces to monitor agent performance and debug interactions [3] [9].

The broader AWS technology stack complements AgentCore perfectly. Amazon S3 serves as the central repository for all unstructured data, including raw log files, configuration documents, and most importantly, runbook assets stored in formats like PDF or Word [2]. These documents are converted into vector embeddings and indexed in Amazon OpenSearch Serverless for high-performance retrieval by the agents using Retrieval Augmented Generation (RAG) [2]. This RAG mechanism is vital for grounding the LLM's responses in factual, organization-specific knowledge, preventing hallucinations and ensuring reliable guidance [2]. For direct interaction with existing ITSM platforms, the agents will use their Code Interpreter capabilities to call the APIs of systems like Jira or ServiceNow, creating or updating tickets programmatically [4] [9]. All actions performed by the agents are securely audited and authenticated using AWS Identity and Access Management (IAM), with fine-grained permissions assigned according to the principle of least privilege [2]. The entire solution is designed to be deployed using Infrastructure as Code (IaC) with AWS CDK, ensuring reproducibility and scalability [2]. This cohesive stack of services provides a robust, flexible, and secure foundation for building the proposed IT service management agents.

Architectural Blueprint: High-Level System Components and Data Flow

The architectural design for this solution is a modular, multi-agent system built on a serverless foundation to ensure scalability, resilience, and cost-effectiveness. The architecture is composed of several distinct but interconnected components that handle different aspects of the data flow, from ingestion to analysis, action, and visualization. This high-level design is intended to be adaptable, capable of integrating with existing ITSM tools like ServiceNow or Jira [2] [4]. The data flows sequentially through these components, with each stage adding value and moving the information closer to actionable intelligence.

The architecture begins with Data Ingestion and Storage. The system ingests diverse data types from various sources. Log files and metric data from IT infrastructure are sent to Amazon CloudWatch, which acts as the primary source for proactive health monitoring [4]. Unstructured documentation, such as incident runbooks and technical guides, is uploaded to Amazon S3 [2]. The diagram below illustrates this initial stage.

```
+------------------------+     +-----------------------------------+
| Infrastructure Logs/Metrics |---->| Amazon CloudWatch                 |
| (e.g., EC2, RDS, Lambda)|     +-----------------------------------+
+------------------------+             ^
        |                              |
        v                              |
+------------------------+     +-----------------------------------+
| Technical Documentation |<----| Amazon S3 (for Runbooks, Docs)   |
| (PDF, Word, etc.)      |     +-----------------------------------+
+------------------------+
```

The second stage is Knowledge Processing and Analysis. Here, the system prepares the unstructured data for consumption by the AI agents. A one-time or periodic process converts the documents in S3 into vector embeddings using an embedding model available through Amazon Bedrock. These vectors are then indexed in Amazon OpenSearch Serverless, creating a searchable knowledge base for the agents [2]. Concurrently, the agents themselves operate in the Analysis and Action stage. A dedicated Monitoring Agent, orchestrated by LangGraph, periodically polls CloudWatch for new alarms and metrics [4]. Another component, the Supervisor Agent, orchestrates the overall workflow, delegating tasks to more specialized agents [4] [9]. These agents utilize the Bedrock AgentCore Runtime and can leverage the Code Interpreter for complex data analysis tasks [5]. They retrieve relevant information from OpenSearch and, when appropriate, invoke remediation actions by calling AWS Lambda functions or third-party APIs via the AgentCore Gateway [2] [6].

The final stage is Action and Reporting. Based on their analysis, the agents can perform several actions. They can create tickets in an external system like Jira or ServiceNow using the Code Interpreter's API call capabilities [4] [9]. They can also populate a dashboard with performance data. The MIS dashboard would pull data from CloudWatch metrics and any internal databases where agent activity and resolution times are logged. All activities and events within the system are captured by Amazon CloudWatch, providing centralized logging, metrics, and distributed tracing for comprehensive observability and debugging [3] [9]. This end-to-end data flow ensures that raw operational data is transformed into intelligent, actionable outcomes, effectively closing the loop on IT service management.

Use Case Scenarios: From Nightly Health Checks to Automated Ticketing

The architectural design translates into practical, value-driven use cases that directly address the core objectives of improving IT service delivery. The following scenarios illustrate how the proposed AI agents would function in real-world situations, demonstrating the automation of ticket correlation, proactive health monitoring, and MIS dashboard generation.

Use Case 1: Automated Incident Correlation

- Trigger: A service desk analyst manually flags a newly created ticket for review.
- Process:
    1. The ticket's details (description, affected system, user group) are passed to the Correlation Agent.
    2. This agent queries a database of recent, unresolved tickets for matches based on the provided criteria.
    3. Using Natural Language Processing (NLP), it analyzes the text of the new ticket against the descriptions of older ones to find semantic similarities, a technique supported by solutions using Bedrock Agents [4].
    4. The agent cross-references this with a knowledge base of previously resolved incidents stored in OpenSearch to identify recurring patterns [2].
    5. If a match is found, the agent presents the analyst with a list of potential duplicates for confirmation. If confirmed, it automatically merges the tickets or links them under a parent incident.
- Outcome: The analyst is freed from tedious manual search and comparison, reducing the risk of missing related incidents and ensuring a unified response to a single problem. This directly improves technician productivity and adherence to ITIL Problem Management guidelines [10].

Use Case 2: Proactive Health Check and Alert Generation

- Trigger: The scheduled nightly process initiates.

- Process:
    1. The Monitoring Agent, configured to run periodically, uses the Bedrock AgentCore Runtime to query Amazon CloudWatch for aggregated metrics over the past 24 hours. It focuses on key indicators like CPU utilization, memory usage, and error rates for critical EC2 instances, Lambda functions, and RDS databases [4].
    2. The agent analyzes this data, looking for anomalies, trends, or values exceeding predefined thresholds. It may use the Code Interpreter to perform statistical analysis on the metric streams [5].
    3. Based on its analysis, the agent compiles a ranked list of the top three most critical issues requiring attention. It generates a concise report for each issue, summarizing the metrics and suggesting a course of action.
    4. The following morning, the agent sends this report to the IT team lead via email (using Amazon SES) and populates a notification channel [2].
- Outcome: Potential infrastructure failures are identified and addressed proactively, preventing service degradation and reducing MTTR. This shifts the IT team's focus from firefighting to preventative maintenance, aligning with the ITIL Continual Improvement practice [10].

Use Case 3: MIS Dashboard Generation and Reporting

- Trigger: The end-of-day process triggers the generation of the MIS dashboard report.
- Process:
    1. The Dashboard Agent is invoked. It connects to multiple data sources: Amazon CloudWatch for raw metric data, the ticketing system's API (via the AgentCore Gateway) for incident and request data, and potentially a custom database where agent performance metrics are logged [3] [4].
    2. The agent calculates the required KPIs. For example, it might calculate Resolution Time by querying the ticket database for closed incidents and averaging the time difference between creation and closure timestamps. It calculates the First Contact Resolution Rate by counting incidents marked as resolved in the first interaction versus the total number of resolved incidents [10].
    3. The agent formats this data into a visually appealing and easily digestible format, such as charts and tables.
    4. This formatted report is then published to a shared location accessible by IT managers, such as an internal portal or a scheduled email attachment.
- Outcome: IT leadership gains immediate access to a daily snapshot of service performance, enabling them to track progress towards business goals, identify systemic bottlenecks, and make data-driven strategic decisions regarding resource allocation and process optimization [10]. This dashboard provides a concrete measure of the IT department's contribution to organizational success.

Key Performance Indicators and Observability

To validate the effectiveness of the AI agents and demonstrate their value to the organization, it is essential to track a set of well-defined Key Performance Indicators (KPIs) and implement robust observability. The chosen KPIs are derived from standard ITIL metrics, ensuring they are relevant and meaningful within a professional ITSM context [10]. The observability strategy leverages native AWS tools to provide deep insight into the agents' operational performance.

The primary KPIs for measuring the impact of the solution are outlined in the table below. These metrics are categorized to provide a holistic view of IT service delivery efficiency.

Metric Category	KPI Name	Description	Relevance to Solution	Source(s)
Service Performance	MTTR (Mean Time to Resolve)	Average time taken to fully resolve an incident from submission to closure.	Directly measures the speed of the fulfillment process improved by automated runbooks and diagnostics.	[10]
	MTBF (Mean Time Between Failures)	Average time elapsed between inherent failures of a service or system.	Measures the stability of the IT environment, influenced by proactive health checks.	[10]
	Service Availability	Percentage of time a service is available and functioning correctly.	Reflects the overall reliability of the IT infrastructure, which proactive monitoring aims to improve.	[10]
Incident Management	Repeat Incident Rate	Percentage of incidents that recur for the same reason.	Indicates the effectiveness of root cause analysis and permanent fixes, which the agent aids.	[10]
	First Contact Resolution Rate	Percentage of incidents resolved during the first interaction with the service desk.	Measures the initial efficiency of the service desk, enhanced by AI-provided runbooks.	[10]
	Escalation Rate	Percentage of incidents escalated to a higher support level.	Shows whether issues are being handled efficiently at the first point of contact.	[10]
User Experience	User Satisfaction Score (CSAT/NPS)	Feedback from users on their experience with the service fulfillment process.	Quantifies the qualitative impact of faster, more effective service on the end-user.	[10]

Metric Category	KPI Name	Description	Relevance to Solution	Source(s)
	Inferred CSAT Score	A score generated by the agent based on user feedback loops within the system.	Provides a direct, quantifiable measure of user sentiment. ServiceNow provides a similar feature.	[11]
Cost & Efficiency	Cost Per Ticket	Total cost of handling an incident divided by the number of tickets.	Measures the operational cost-effectiveness of the IT service desk.	[10]
	Efficiency Gain (%)	The percentage improvement in task completion time when assisted by an AI agent compared to manual methods.	Directly measures the productivity gain for technicians. ServiceNow provides a dashboard for this.	[11]

For observability, the solution relies heavily on Amazon CloudWatch. Enabling observability within Bedrock AgentCore provides a powerful console view that allows developers and operators to inspect agent interactions, monitor session duration, and analyze latency [3]. CloudWatch captures detailed metrics for every aspect of the agent's operation, including token usage, invocation latency, session duration, and error rates [1]. This data is invaluable for performance tuning and debugging. Furthermore, distributed tracing can be implemented using tools like LangSmith or by leveraging OpenTelemetry integration, providing a granular view of the entire workflow, from the initial trigger to the final API call [4] [9]. This level of observability ensures that the system is not only functional but also transparent, reliable, and continuously improvable.

Reference

1. Amazon Bedrock AgentCore (Preview) - AWS https://aws.amazon.com/bedrock/agentcore/
2. Automate IT operations with Amazon Bedrock Agents - AWS https://aws.amazon.com/blogs/machine-learning/automate-it-operations-with-amazon-bedrock-agents/
3. Amazon Bedrock AgentCore Agents - Amazon CloudWatch https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AgentCore-Agents.html
4. Multi-agent system: AWS Incident Response | by Madhur Prashant https://medium.com/@madhur.prashant7/multi-agent-system-aws-incident-response-2790f95b7a6f
5. Introducing the Amazon Bedrock AgentCore Code Interpreter - AWS https://aws.amazon.com/blogs/machine-learning/introducing-the-amazon-bedrock-agentcore-code-interpreter/

6. Introducing Amazon Bedrock AgentCore Gateway https://aws.amazon.com/blogs/machine-learning/introducing-amazon-bedrock-agentcore-gateway-transforming-enterprise-ai-agent-tool-development/
7. Unlock Agent Productivity with ITSM AI Agents (Age... - ServiceNow https://www.servicenow.com/community/itsm-articles/unlock-agent-productivity-with-itsm-ai-agents-agentic-ai/ta-p/3117919
8. AI Agents - ServiceNow https://www.servicenow.com/products/ai-agents.html
9. Incident response & Operations Multi-Agent A2A System ... - Medium https://medium.com/@madhur.prashant7/incident-response-operations-multi-agent-a2a-system-with-bedrock-agentcore-primitives-openai-7a0ccb991d5d
10. ITIL Measurement and Reporting: Enhancing IT Performance https://iseoblue.com/post/measuring-reporting/
11. AI Agent Analytics dashboard - ServiceNow https://www.servicenow.com/docs/bundle/yokohama-intelligent-experiences/page/administer/now-assist-ai-agents/concept/ai-agent-dashboard.html