International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| AECTIC2025-Conference Proceeding
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.AECTIC-110

Original Article

AI Advisor Copilot: An AWS-Native, Spring Boot Orchestration Layer for Wealth Proposal Generation

Prashant Singh
Senior Technical Architect, Coforge Limited

Abstract - *This journal paper presents an AI Advisor Copilot architecture built on Amazon Web Services (AWS) and Spring Boot microservices for large-scale wealth management platforms. Rather than replacing existing engines such as model catalogs, risk scoring, proposal generation, and RTQ services, the Copilot introduces a reasoning and orchestration layer that automates manual, cognitively intensive advisor workflows. The Copilot focuses on external account migration, multi-goal allocation design, and explainable model selection. It integrates Amazon EKS, PI Gateway, RDS, DynamoDB, MSK/SQS, EventBridge, S3, CloudWatch, and Amazon Bedrock to orchestrate enterprise services while keeping the advisor in full control of every recommendation.We describe the architecture, algorithms, workflows, security model, and business impact of this solution, and discuss how it can be adopted incrementally in a real wealth management organization.*

Keywords *- Wealth management, AI Copilot, AWS, Spring Boot, microservices, explainability, external account migration, model matching, financial proposals.*

1. Introduction

Advisors in modern wealth management firms are expected to deliver highly personalized advice across multiple goals, accounts, and custodians, often under tight time constraints. Organizations already invest heavily in strategist-curated model portfolios, risk engines, proposal tools, and planning systems. However, advisors still perform much of the critical reasoning manually: They interpret external account holdings, map those to internal models, adjust for risk and goals, and document the rationale. This gap between available tools and actual human workflow motivates the AI Advisor Copilot. The Copilot does not attempt to become an autonomous robo-advisor. Instead, it acts as aco-pilot that orchestrates existing services, surfaces the right models, proposes allocations, and writes explanations, all under the advisor's supervision. This paper formalizes the architecture,workflows, and governance required to build such a system on AWS using Spring Boot microservices.

2. Background and Problem Context

From a system perspective, most enterprise wealth platforms provide:

- A model catalog consisting of strategist models, advisor models, and third-party strategies.
- Risk engines that can evaluate the risk score of a portfolio or individual holdings.
- Goal and RTQ services capturing risk tolerance and objectives.
- Proposal generation engines that assemble final client-facing documents.

While these components are individually powerful, they are not coordinated around the advisor's actual decision-making process. When onboarding an external account, advisors must understand foreign holdings, estimate risk, identify suitable internal model blends, and adjust them to reflect updated goals or risk appetite. Each of these steps requires multiple systems, deep product knowledge, and several manual iterations. The problem is not lack of models or analytics; it is lack of an intelligent, explainable orchestration layer that sits above existing engines and helps the advisor reason quickly and consistently.

3. Requirements for an Enterprise-Grade Advisor Copilot

A Copilot designed for a production wealth platform must satisfy both technical and business requirements:

- Reuse, not replace, existing models and services. It should integrate with current risk scoring, model catalogs, proposal engines, and client data stores.
- Maintain human-in-the-loop control. Advisors must be able to review, edit, and override any recommendation.
- Provide high explainability. Every suggestion must include clear, data-backed rationale suitable for clients, advisors, and compliance.

- Scale horizontally. The platform should support thousands of advisors and millions of accounts with predictable performance.
- Respect security and regulatory constraints. Identity, entitlements, encryption, and audit trails are mandatory.
- Support gradual rollout. The Copilot should first operate in a read-only assistive mode before evolving into a more automated orchestration role.

4. High-Level Architecture

The Copilot architecture introduces an Advisor Intelligence Layer built on AWS and Spring Boot. Figure 1 shows the high-level microservice layout. At the top, advisors interact with the system via a web or desktop interface that connects through Amazon API Gateway or an Application Load Balancer (ALB). Requests are routed to a copilot-gateway-service, which then calls dedicated microservices for portfolio insights, model matching, goal-based allocation, explainability, and proposal orchestration.

Each microservice runs as a Spring Boot application packaged in a Docker container and deployed on Amazon EKS. Persistent data such as household records, proposals, and advisor preferences is stored in Amazon RDS, while fast session state and model metadata caching live in Amazon DynamoDB. External portfolio files and generated proposals are stored in Amazon S3. Messaging between services leverages Amazon MSK or Amazon SQS and EventBridge for asynchronous workflows like external account ingestion and background scoring.

This layered design separates concerns cleanly: the user interface never directly accesses core engines; instead, it interacts with a stable Copilot API, and the Copilot orchestrates calls to existing enterprise services.

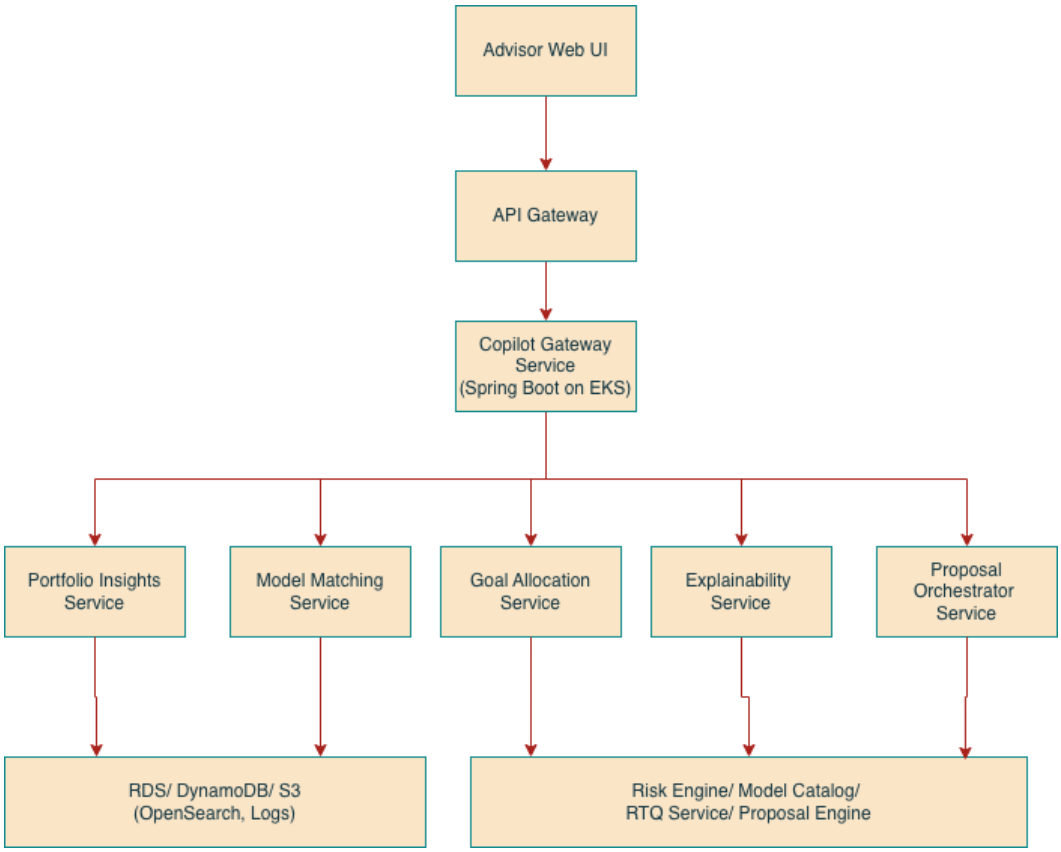

Fig 1: Copilot Microservice Architecture

Fig. 1. High-level Spring Boot microservices forming the Advisor Copilot orchestration layer on Amazon EKS.

5. AWS Infrastructure Layers

Figure 2 presents the main AWS layers used by the Copilot. The presentation layer sits at the top, followed by networking and API ingress, EKS-based compute, messaging, data storage, analytics and AI services, and finally observability and security. This

view is valuable when aligning responsibilities between cloud platform teams, application teams, and security/governance stakeholders.

6. Microservice Responsibilities

Each Spring Boot microservice has a clear, bounded responsibility:

- Copilot-gateway-service – authenticates advisors, exposes REST/GraphQL APIs, performs request routing and coarse-grained authorization.
- Portfolio-insights-service – maps external holdings to internal identifiers, calls risk engines, and computes style, sector, and concentration metrics.
- Model-matching-service – evaluates internal models and blends using similarity, risk alignment, and advisor preference rules.
- Goal-allocation-service – maps household assets and goals to model assignments per account or bucket.
- Explainability-service – turns structured decision data into natural language explanations using templates and Amazon Bedrock.
- Proposal-orchestrator-service – integrates with the existing proposal engine to assemble the final client-facing document.

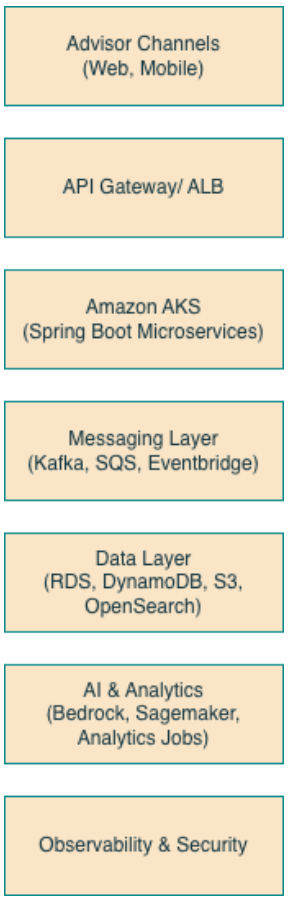

Fig 2: AWS Infrastructure Layers

Fig. 2. Conceptual mapping of the Advisor Copilot onto AWS infrastructure services, from channels down to observability and security.

7. External Account Migration Workflow

One of the highest-value use cases for the Copilot is external account migration. Advisors frequently bring assets from other custodians into the platform. Without assistance, they must inspect each security, infer the strategy, and then search for equivalent internal models. Figure 3 summarizes the automated workflow. The process begins when an external file or feed containing holdings is dropped into an S3 bucket. The Custodian Integration Service normalizes the data, maps tickers to internal instruments,

and publishes an event. The portfolio-insights-service consumes this event, computes the risk score and exposures, and stores a normalized portfolio view. The model-matching-service then proposes candidate internal models or blends that preserve the client's risk and high-level allocation characteristics. The explainability-service generates justifications for each suggested option. Finally, the advisor reviews the options, makes adjustments, and triggers proposal orchestration.

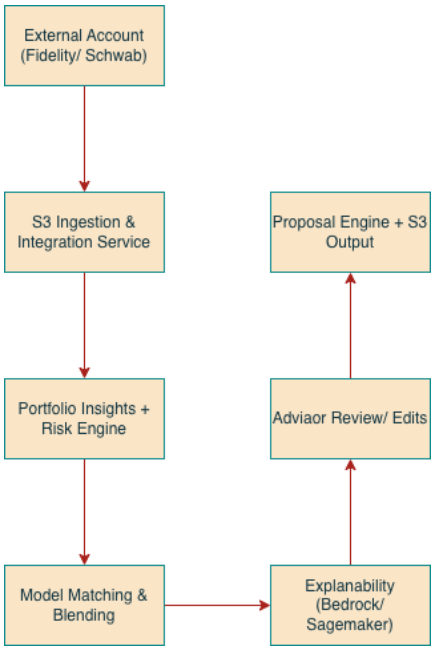

Fig 3: External Account Migration Flow

Fig. 3. End-to-end workflow for migrating an external portfolio into internal models via the Advisor Copilot.

8. Explainability Framework

Explainability is a first-class requirement for any AI-enabled wealth platform. The explainability-service does not independently invent recommendations; instead, it receives structured inputs describing what was decided and why: selected model IDs, risk scores before and after, drift metrics, cost comparisons, and any constraints used (such as ESG exclusions or tax considerations).

As shown in Figure 4, these structured inputs are submitted to the explainability-service, which prepares a prompt template and calls Amazon Bedrock for LLM inference. The service then returns three explanation variants: a technical version for advisors, a simplified version for clients, and a detailed version for compliance and supervision teams. All inputs and outputs are logged to S3 and RDS for auditability. This pattern ensures that AI is used only for language generation, not for core allocation logic, and that every explanation is traceable and reproducible.

Fig 4: Explainability Pipeline

Fig. 4. Explainability pipeline using structured decision data and Amazon Bedrock to generate advisor, client, and compliance views.

9. Model Matching and Blending Algorithms

The model-matching-service computes similarity scores between the external or current portfolio and candidate internal models. The algorithm can combine factors such as style vector similarity, risk score distance, sector exposure alignment, and fee impact into a single composite score. When no single model meets the thresholds, the engine constructs 2–3 model blends and evaluates them according to the same metrics. This yields a ranked list of human-readable options for the advisor.

10. Security, Compliance, and Governance

Security and compliance considerations span IAM, encryption, audit logs, and model governance. All microservices operate inside a VPC with strict security group rules. IAM roles are defined per service. Data at rest is encrypted with KMS-managed keys. Every recommendation is logged with its inputs, outputs, and explanations. This creates a robust audit trail for regulatory review and supervision use cases.

11. Scalability, Latency, and Reliability Engineering

Amazon EKS, combined with horizontal pod autoscaling and managed message queues, allows the Copilot to scale with advisor demand. Non-blocking, asynchronous workflows are used where possible, especially for heavy operations such as external portfolio ingestion and similarity computation. Performance-sensitive operations such as proposal preview and what-if analysis are optimized through caching in DynamoDB and careful use of Bedrock inference.

12. Business Impact and Advisor Productivity

From a business perspective, the Copilot directly reduces the time advisors spend on repetitive mechanical tasks. Instead of manually looking up models, calculating drifts, and crafting explanations, advisors focus on validating and personalizing the Copilot's suggestions. The net effect is higher proposal throughput, more consistent recommendations, and improved client experience, without sacrificing the human relationship at the core of advice.

13. Limitations and Future Work

This architecture intentionally avoids full automation of investment decisions. The advisor remains responsible for the final proposal. Future work can extend the Copilot into proactive monitoring, tax-aware rebalancing suggestions, and multi-custody optimization while maintaining the same explainability and control principles.

14. Conclusion

The AI Advisor Copilot described in this paper demonstrates how AWS and Spring Boot microservices can be combined to create an intelligent orchestration layer for wealth management. By building on existing enterprise engines rather than replacing them, the Copilot offers a pragmatic path to AI augmentation that is explainable, secure, and scalable. This design can be adapted by other financial institutions seeking to modernize advisor workflows while preserving regulatory trust and human judgment.

15. Advisor Copilot Demo Storyboard

Scene 1 – Advisor Login
Advisor opens the LPL advisor workstation. A new "Copilot" icon appears in the navigation menu.
Scene 2 – External Account Upload
Advisor selects 'Migrate External Account'. Drags a Fidelity CSV/PDF into the uploader.
Scene 3 – Copilot Processing
Copilot reads the file, identifies all holdings, normalizes tickers, calls Risk Engine.
Scene 4 – Insights Dashboard
Copilot displays portfolio risk, sectors, top holdings, concentration warnings.
Scene 5 – Model Suggestions
Copilot shows 3 options: Single Model, Blend, Tax-Efficient Alternative.
Scene 6 – Explainability Panel
Copilot generates rationale: 'Suggested Model A because allocation drift = 4.3% and risk alignment is within ±1 band'.
Scene 7 – Goal Planning
Advisor adds an 'Education Goal (2040)' → Copilot suggests a moderate-growth model blend.
Scene 8 – Advisor Edits
Advisor increases equities by 5% → Copilot recalculates drift and updates explanation.
Scene 9 – Proposal Assembly
Advisor clicks 'Generate Proposal'. Copilot invokes Proposal Service and creates PDF stored in S3.
Scene 10 – Client Review Ready
Proposal appears with explanations, charts, and recommended allocations. Advisor sends to client.

References

[1] Amazon Web Services, "AWS Well-Architected Framework," 2024.
[2] M. Richards, "Microservices vs. Service-Oriented Architecture," O'Reilly Media, 2023.

[3] N. Doshi et al., “Explainable AI: A Technical Review,” ACM Computing Surveys, 2023.
[4] S. Newman, “Building Microservices: Designing Fine-Grained Systems,” O’Reilly, 2022.
[5] NIST, “AI Risk Management Framework,” National Institute of Standards and Technology, 2023.
[6] Amazon Web Services, “Amazon Bedrock – Foundation Models for Enterprise AI,” AWS Technical Overview, 2024.
[7] C. Eaton et al., “Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data,” IBM Redbooks, 2021.
[8] D. Guttman et al., “Financial Portfolio Optimization using Hybrid ML Approaches,” IEEE Transactions on Computational Finance, 2022.
[9] J. Dean et al., “Large Scale Distributed Systems and Cloud Computing,” Google Research, 2023.
[10] M. Arrieta et al., “Explainable AI: From Black Box to Glass Box,” IEEE Access, 2022.