Fred: A Modular Multi-Agent AI Assistant for Kubernetes Deep Analysis, Management, and Optimization

Tanguy Jouannic[1], Dimitri Tombroff[1], Lorenzo Gerardi[1], and Reyyan Tekkin[1]

[1]Thales Services Numériques, France.

Abstract

Kubernetes is a critical technology for modern IT infrastructures, yet its operational complexity often exceeds the capabilities of many organizations. To address this challenge, we introduce Fred, an open-source AI assistant designed to simplify Kubernetes application management through a modular, multi-agent architecture. Fred leverages generative AI and advanced reasoning techniques, such as plan-and-solve prompting and chain-of-thought reasoning, to provide in-depth analysis, eco-design recommendations, and actionable optimizations. By integrating domain-specific expertise with contextual insights, Fred enables architects and administrators to enhance energy efficiency, reduce costs, and ensure compliance with best practices. This paper explores Fred's architecture, its implementation challenges, and its potential for Kubernetes management.

1 INTRODUCTION

Kubernetes is a strategic technology for Thales Group and Thales Services Numériques in particular. We maintain, like many, our own distribution (Kubernetes together with a number of key cloud-native technologies) to serve as common platform/sdk for our many data processing use cases.

As part of an internal innovation project, we decided to tackle the issue of Kubernetes applications operational complexity, with a focus on a particular use case: frugality. How can we efficiently design, maintain, or improve kubernetes applications while ensuring that their overall costs are understood and, we hope, reduced, without sacrificing the expected functional requirements. These are tough questions. Despite all the well-known benefits of Kubernetes (standardization, declarative orchestration patterns, open communities, etc.), mastering the exploding CNCF eco-system requires a multi-domain expertise that is out of reach or expensive for many companies. Worst of all, what is smart today is obsolete the year after. Architects are asked to produce frugal, scalable, secured, easy-to-maintain, and easy-to-modernize solutions. Needless to say, they need help.

Recognizing these challenges, we decided to use generative AI to develop an open-source AI assistant, called *Fred*, to help our architects and administrators manage their Kubernetes applications. In contrast to the many generative AI initiatives ([1], [2], [3] to name a few), our focus is on tackling

higher-level architectural considerations. Does my architecture conform to best-practice eco-design recommendations? how could I reduce my energy consumption? Can you help me assess the data flow of my entire application?

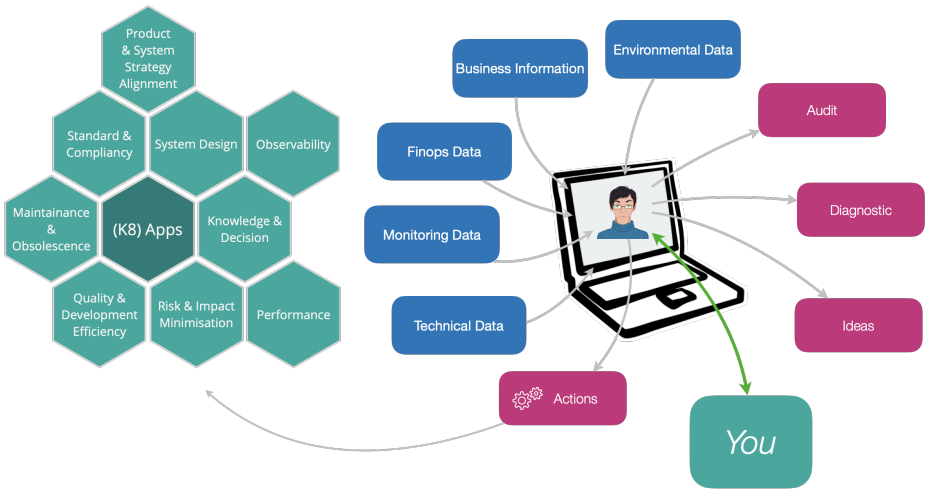

Figure 1: Fred overview

For this purpose, Fred features a modular multi-agent architecture that is well-suited for building *domain-specific* experts. By customizing this architecture, we have created an assistant capable of deep analysis, management, and optimization of Kubernetes clusters.

This article presents an in-depth look at Fred's architecture and details how we have adapted it to meet the specific needs of Kubernetes management. In its current status, Fred provides automatic documentation and explanations of a Kubernetes application, generates so-called eco-score reports for each components, and goes up to provide actionable insights and optimizations.

Our development process includes extensive data engineering focused on Kubernetes components, enabling Fred to perform in-context learning and adapt to various operational scenarios. Importantly, Fred emphasizes the identification and implementation of optimizations to reduce operational costs, energy consumption, and carbon footprint.

2 FRED ARCHITECTURE

Drawing from contemporary AI literature, Fred leverages a modular multi-agent system that integrates planning, chain-of-thought reasoning, specialized expertise, and secure tool usage to provide comprehensive solutions. This architecture illustrated at figure 2 is engineered to address intricate tasks by decomposing them into manageable steps, assigning them to domain-specific experts, and ensuring that the outcomes meet the user's objectives. This approach aligns with the Plan-and-Solve Prompting method ([4]), which emphasizes creating structured plans followed by step-by-step solutions to enhance reasoning capabilities in large language models.

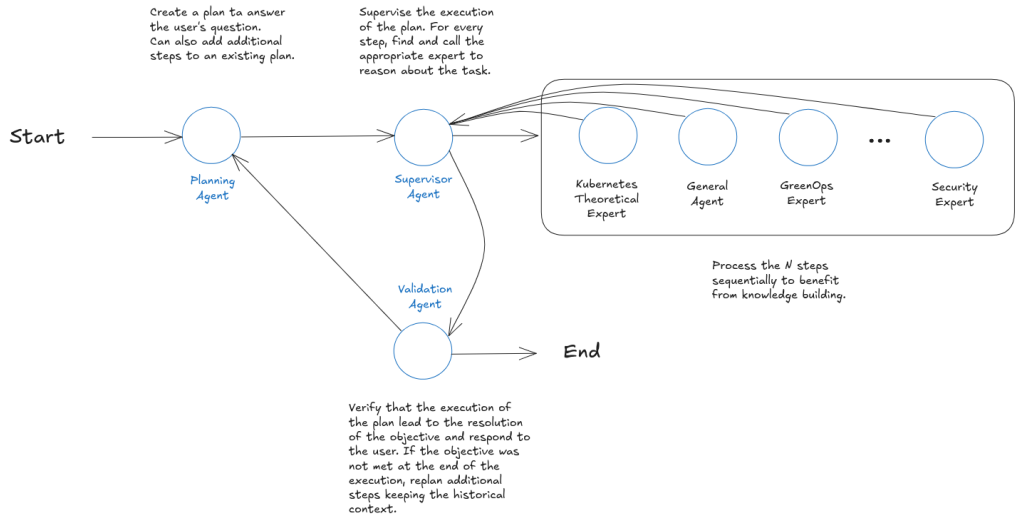

Figure 2: Fred workflow

2.1 Planning Agent: Crafting Tailored Strategies

At the core of Fred's operation is the Planning Agent, which initiates the problem-solving process. This agent is responsible for understanding the user's request and devising a strategic plan to address it. The Planning Agent employs a dynamic and adaptive approach to problem decomposition, tailoring its strategies to the complexity of the task. Simpler tasks result in concise plans, while more intricate problems lead to detailed plans with additional steps. This adaptability ensures Fred's responses are both efficient and comprehensive, regardless of task difficulty.

The planning is not generic; it is tailored to the collective expertise of the available agents, ensuring that each step is assigned to the most suitable expert. By leveraging the unique strengths of specialized agents, the Planning Agent optimizes the execution of each task, ensuring high-quality results across domains.

The Planning Agent's ability to structure effective plans is informed by principles of zero-shot reasoning. Drawing inspiration from findings in [5], the Planning Agent uses its inherent reasoning capabilities to generate detailed plans without requiring extensive task-specific training. By leveraging techniques like Zero-shot-CoT (Chain of Thought), which involves prompting the system to "think step by step," Fred's Planning Agent achieves enhanced performance on tasks involving arithmetic, logical reasoning, and symbolic processing.

2.2 Supervisor Agent: Orchestrating Expert Collaboration

Once the plan is established, the **Supervisor Agent** takes charge of orchestrating the workflow. This agent dispatches tasks to the appropriate experts in a sequential manner, ensuring that each step builds upon the results of the previous ones. By sequentially adding the outcomes of earlier steps to the context, the Supervisor Agent maintains a coherent flow of information and reasoning throughout the problem-solving process. This sequential execution is crucial for tasks that require

cumulative knowledge and incremental reasoning.

The effectiveness of this approach is supported by findings in [6], which demonstrate that chain-of-thought prompting can elicit advanced reasoning capabilities in large language models, enhancing their ability to handle multi-step tasks.

2.3 Specialized Expert Agents: Domain-Specific Problem Solvers

Fred's architecture features a suite of **Specialized Expert Agents**, each designed to tackle tasks within their specific domains of expertise. These agents are powered by extensive **knowledge bases** that are enhanced through in-context learning or integrated using Retrieval Augmented Generation (RAG) methods. This empowers them to reason more effectively and provide solutions deeply informed by specialized knowledge.

In addition to their knowledge bases, these experts are equipped with **domain-specific capabilities**. For example, a Kubernetes expert agent might have the ability to query the Kubernetes API to retrieve real-time information about workloads running in a cluster. By integrating such capabilities, the experts can perform actions beyond static analysis, dynamically interacting with the environment to collect information and implement changes.

Complementing these specialized agents is the **General Agent**, which handles tasks requiring broad understanding rather than deep domain-specific knowledge. This agent ensures that Fred can address general queries and provide comprehensive support, filling in gaps that specialized experts might not cover. The General Agent is crucial for maintaining the assistant's versatility, allowing it to respond effectively to a wide spectrum of user requests.

All agents operate within the **ReAct (Reasoning and Acting) framework**, granting them access to various **tools** and functions necessary for their tasks. These tools range from web search capabilities to APIs that modify system configurations. Importantly, the use of these tools is governed by security protocols that incorporate **human-in-the-loop** interactions. Before an agent performs a critical action—especially one that modifies system states—the process pauses to allow for user approval. This ensures all significant changes are vetted, maintaining system **integrity and security**. Non-critical actions that do not alter system configurations may proceed without interruption, streamlining the workflow.

2.4 Validation Agent: Ensuring Goal Fulfillment

After all tasks have been executed, the **Validation Agent** assesses whether the plan has successfully met the user's objectives. This agent reviews the outcomes of each step, ensuring that the collective results align with the intended goals. If the objectives are met, the Validation Agent summarizes the findings and presents them to the user in a coherent and actionable format.

If the objectives are not fully achieved, the process is iterative. The Validation Agent triggers a feedback loop by consulting the Planning Agent to revise the plan. Additional steps are added as necessary, and the cycle of execution and validation repeats. This iterative process continues until the user's goals are satisfactorily met, ensuring a thorough and effective problem-solving approach.

3 IMPLEMENTATION CHALLENGES

Implementing Fred within Kubernetes environments presents a series of significant challenges due to the inherent complexity and scale of Kubernetes operations. These challenges stem from the vast volume of data, the diversity of domains involved, and the necessity for deep contextual understanding. This section explores these obstacles in detail.

3.1 Data Volume and Complexity

Kubernetes clusters involve a large amount of data, primarily YAML configuration files that define the state and behavior of every component within the system. These YAML files are not only extensive but also highly intricate, encompassing detailed specifications for deployments, services, configurations, and policies. The sheer volume and complexity of this data pose a substantial challenge for large language models (LLMs) that power Fred's expert agents. Without a refined strategy, Fred would suffer from:

- **Decreased Accuracy**: The model may struggle to maintain precision when processing large datasets.
- **Loss of Specificity**: Important details might be overlooked or generalized, reducing the effectiveness of analyses.
- **Hallucinations**: The model might generate information that is not present in the input data, leading to unreliable outputs.
- **Context Window Limitations**: The vast amount of information exceeds the context window of LLMs, making it difficult to process raw data effectively.

These factors collectively hinder Fred's ability to accurately analyze and interpret Kubernetes configurations, thereby limiting its effectiveness in managing and optimizing Kubernetes environments.

3.2 Diversity of Domains to Explore

Kubernetes management is inherently multidisciplinary, spanning a vast array of domains that require specialized knowledge and expertise. While the following are key examples, they represent only a subset of the broader spectrum of considerations involved in effectively managing Kubernetes environments:

- **Kubernetes System Internals**: Understanding the core functionalities, components, and operational mechanics of Kubernetes.
- **Scaling and Performance Optimization**: Ensuring workloads and clusters can scale efficiently to meet demand while maintaining performance.

- **Cybersecurity**: Securing the cluster against vulnerabilities, threats, and ensuring robust configuration management.
- **Regulatory Compliance**: Adhering to various legal and industry-specific standards, such as GDPR, HIPAA, and others.
- **Financial Operations (FinOps)**: Optimizing the cost-efficiency of cloud resources and workloads.
- **Green Operations (GreenOps)**: Minimizing energy consumption and environmental impact in Kubernetes operations.
- **Software Architecture**: Aligning deployments with best practices in software design and architecture to maintain system robustness and flexibility.

The breadth of these domains makes it challenging for a single AI model or agent to provide accurate and relevant analyses across all areas. Relying on zero-shot prompting, where the model attempts to generate responses without domain-specific training or context, often results in generic or irrelevant outputs. This diversity necessitates a more nuanced approach to ensure that each domain is adequately addressed, complicating the implementation of Fred for Kubernetes.

4 A DATA ENGINEERING ISSUE

As always in solving complex data problems, a careful and initial data engineering effort is the way to go. We started by making Fred construct a foundational knowledge model of our target Kubernetes application. This first phase is automatic. This model helps each expert in using its specialized tools effectively to gather additional specific, detailed information as needed.

Note that studies like [7] have highlighted that while LLMs are capable of handling extensive token sequences, they often struggle to maintain high performance in complex, context-rich scenarios. This is why the focus of our data engineering effort consists mostly in *reducing* the information volume and focusing on critical content. We then mitigate issues related to context window limitations and maintain the models' accuracy and relevance. This overall approach ensures that Fred's expert agents can process complex information without being hindered by the limitations of current LLM technology.

Here is how it works in practice: Fred first collects, via Kubernetes API calls, a sound understanding of the application. It then transforms this into a high-level overview of the cluster's architecture and workloads. By presenting its experts with this global topology, Fred enables them to comprehend the relationships and dependencies between different components.

Fred then iteratively condenses and improves its topology model and focuses on essential information. Sometimes it gathers information using specific API points (config maps, Kubernetes operators API, external sources). In turn, this further reduces the cognitive load on each expert agent.

To implement this strategy, we collaborate closely with our company's Kubernetes experts and support teams to identify key data points, such as container images, software versions, scaling parameters like replica counts, and ingress configurations. Extracting and summarizing this critical information provides the agents with a concise yet comprehensive snapshot of the system's operational state. We proceed with the same approach for the key cloud-native components we heavily rely upon in our projects. For example, Kafka, OpenSearch, Minio, ClickHouse, and Keycloak (to name just a few) are often used. We provide Fred with insights for each so that it can further complete its knowledge.

To facilitate this process, we implemented a hierarchical, or pyramidal, summarization strategy. Through advanced LLM engineering techniques, we generate natural language overviews at multiple levels of abstraction. We start by creating summaries for individual workloads (deployments, statefulsets, jobs, etc.), then aggregate these into summaries for namespaces, and finally compile a cluster-wide overview. This multi-tiered representation enables Fred's specialized expert agents to navigate from broad overviews to specific details seamlessly. It enhances their ability to identify patterns, anomalies, and areas of concern within the cluster.

5 FRED SPECIALIZED EXPERT AGENTS

We just described the overall logic of Fred internal workflow. Let us now describe the current specialized agents, each designed to address specific domains within Kubernetes management. Fred is of course designed to facilitate the plugin of additional experts.

5.1 Kubernetes Technical Expert

The **Kubernetes Technical Expert** focuses on providing precise diagnostics and troubleshooting within Kubernetes clusters.

- **Simplified Kubernetes API Access**: Queries Kubernetes APIs to retrieve raw YAML object definitions for deployments, services, and configurations.
- **Current Knowledge Integration**: References up-to-date documentation and community resources for accurate recommendations.
- **ReAct Framework**: Utilizes structured reasoning to assess misconfigurations, deprecated API usage, and optimization opportunities.

This agent delivers granular analyses of cluster components, aiding administrators in identifying and addressing technical issues efficiently.

5.2 Theoretical Kubernetes Expert

The **Theoretical Kubernetes Expert** ensures cluster configurations align with best practices and theoretical guidelines.

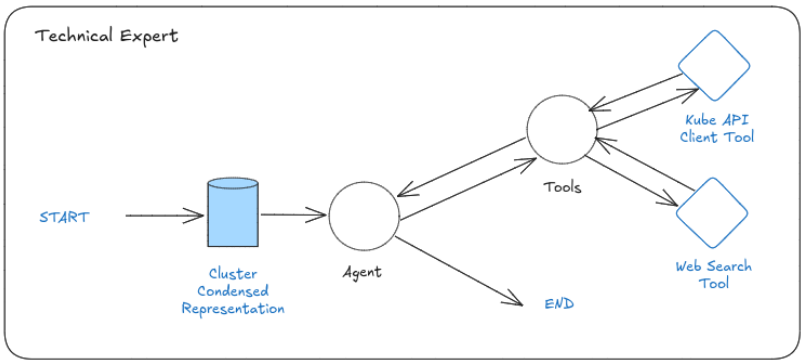

Figure 3: Technical Expert Overview

- **Cluster Topology Insight**: Builds on a high-level understanding of the cluster's architecture to evaluate design adherence.
- **Guideline-Driven Analysis**: Employs Kubernetes documentation and best practices as a knowledge base, using Retrieval Augmented Generation (RAG) for precise recommendations.
- **Context-Aware Reasoning**: Identifies deviations from standards and provides actionable guidance.

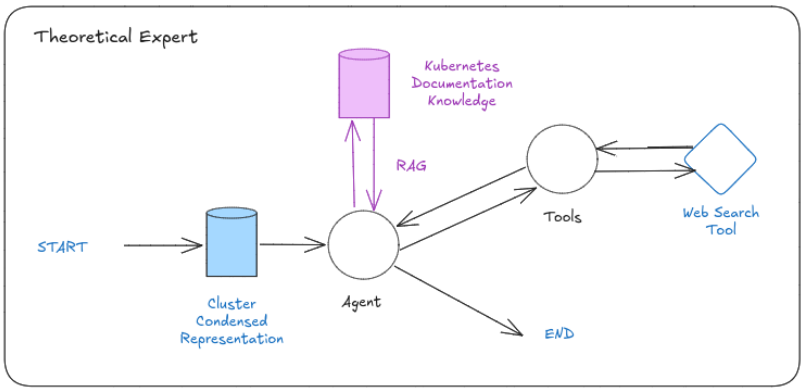

Figure 4: Theoretical Expert Overview

This expert is particularly valuable for compliance and for enhancing system maintainability through alignment with theoretical frameworks.

5.3 GreenOps Expert

The **GreenOps Expert** addresses sustainability by focusing on reducing the cluster's energy consumption and carbon footprint.

- **Energy Data Analysis**: Utilizes tools like Kepler probes to access and analyze energy consumption metrics at the node and workload levels.

- **Energy Mix Insights**: Incorporates information on renewable versus non-renewable energy sources to suggest eco-friendly optimizations.
- **Targeted Recommendations**: Proposes workload adjustments and scheduling strategies to balance performance with sustainability.

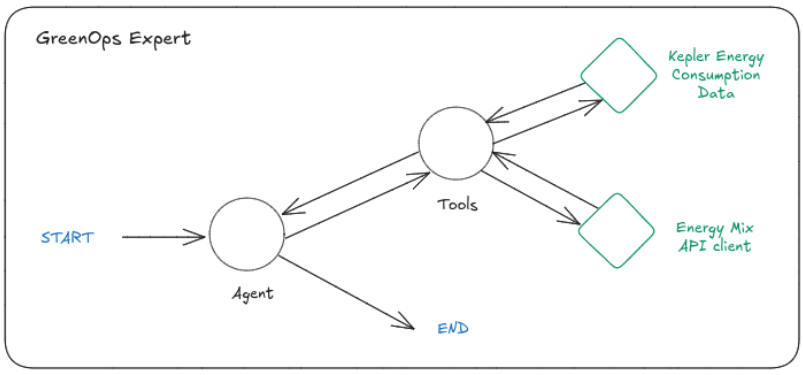

Figure 5: GreenOps Expert Overview

This agent supports both environmental goals and cost savings by optimizing energy use.

5.4 Scaling Expert

The **Scaling Expert** optimizes workload scalability and resource efficiency within Kubernetes environments.

- **Dynamic Scaling Strategies**: Leverages Keda Kubernetes Event-Driven Autoscaling to adjust resources based on workload demand.
- **Human-in-the-Loop Validation**: Ensures administrators can review and approve scaling actions before execution.
- **Collaborative Optimization**: Works alongside other agents to balance scalability with sustainability.

This agent enhances system responsiveness and operational efficiency by aligning resource allocation with demand.

Our goal is to have Fred generate precise scale-up and scale-down configurations leveraging technologies such as *Keda* [8] and *Karpenter* [9]. Configuring a Kubernetes application with these technologies can yield significant cost and energy reductions. Yet it is rarely implemented, likely due to the overall complexity and expertise level required.

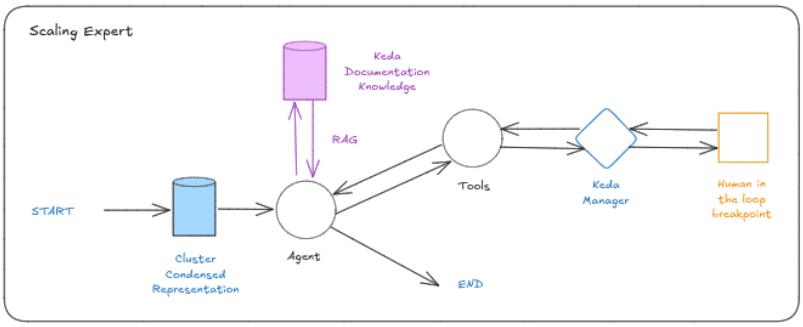

Figure 6: Scaling Expert Overview

5.5 Collaborative Expertise

Last, Fred's agents collaborate through the Supervisor Agent to address complex, multi-domain challenges. For example, the GreenOps Expert and Scaling Expert might jointly optimize resource usage while minimizing energy consumption, delivering solutions that are both efficient and sustainable.

6 INCORPORATING CONTEXTUAL FACTS FOR ENHANCED DECISION MAKING

Technical data often falls short in capturing the full spectrum of considerations necessary for optimal decision-making. To bridge this gap, we have implemented a mechanism that allows users, developers, and architects to input **Facts**—natural language annotations linked to specific components within the system, such as workloads, namespaces, or the entire cluster. These Facts provide crucial contextual information that enhances Fred's ability to deliver tailored and accurate recommendations.

These Facts encompass a wide range of contextual information:

- **Technical Constraints**: Details about hardware limitations, software dependencies, or network requirements that influence how components should be managed.
- **Usage Patterns**: Information on application usage, including peak traffic periods, critical user interactions, or specific workload characteristics affecting performance.
- **Justifications for Technical Choices**: Explanations behind particular configurations or architectural decisions, providing rationale that might not be evident from the system's state.
- **Business Constraints**: Insights into budgetary limits, cost optimization goals, or strategic priorities impacting resource allocation and scaling decisions.
- **Legal and Compliance Requirements**: Details on regulatory standards, data privacy laws, or industry-specific compliance mandates that the system must adhere to.

6.1 Enhancing Expert Agents' Reasoning Capabilities

By integrating these Facts into Fred's reasoning process, the expert agents gain a deeper understanding of the operational environment. This additional layer of context enables them to make more informed and precise recommendations, ensuring that optimization strategies are not only technically sound but also aligned with organizational objectives.

For example, if a workload is annotated with a Fact indicating that it handles sensitive financial data subject to stringent compliance regulations, the Security Expert Agent will prioritize recommendations that enhance data protection and regulatory compliance over purely performance-based optimizations.

6.2 Improving Accuracy and Personalization

The inclusion of Facts helps to prevent generic or irrelevant suggestions that might arise from a lack of contextual awareness. By tailoring advice to the specific needs and constraints of different components, Fred ensures that optimizations are aligned with the organization's priorities.

This approach offers several benefits:

- **Enhanced Reasoning**: With access to context that goes beyond raw technical metrics, the expert agents can better infer the implications of various optimization actions.
- **Improved Accuracy**: Contextual data helps prevent recommendations that might technically optimize the system but conflict with business objectives or legal requirements.
- **Personalized Recommendations**: Tailoring advice to the specific needs and constraints of different components ensures that optimizations are relevant and actionable.

6.3 Aligning AI with Human Expertise

Our approach aligns with contemporary AI research emphasizing the importance of contextual information in effective decision-making. By enabling users to provide natural language Facts, we create a collaborative environment where human expertise complements AI capabilities. Users contribute their domain knowledge and strategic insights, while Fred processes this information alongside technical data to generate holistic recommendations.

This synergy between human input and AI processing enhances the overall effectiveness of Kubernetes management, ensuring that both technical and non-technical considerations are accounted for. We believe it is one of our work's original contributions.

7 CASE STUDY

Consider a scenario where a namespace is annotated with the following Fact:

> *"This namespace hosts a customer-facing application that experiences peak traffic during business hours in the Eastern Time Zone. It is critical for maintaining SLA commitments with our top-tier clients."*

With this context, the Scaling Expert Agent can:

- Prioritize resource allocation during peak hours to maintain performance and meet SLA commitments.
- Schedule maintenance and updates during off-peak hours to minimize impact on users.
- Coordinate with the GreenOps Expert to optimize energy consumption without compromising service quality.

Similarly, Facts related to legal and compliance requirements enable Fred to align its recommendations with regulatory mandates. For instance, annotating a workload with a Fact such as:

> *"This application processes personal health information and must comply with HIPAA regulations."*

This informs the Security Expert Agent to:

- Ensure that data encryption is enforced both at rest and in transit.
- Verify that access controls and audit logs meet HIPAA standards.
- Recommend configurations that enhance data privacy and security compliance.

8 STATE OF THE ART

In this section, we cite the most relevant papers and explain how they influenced the design of Fred. The **Selected Papers** provided direct inspiration and are foundational to Fred's architecture. The **Evaluated Papers**, while valuable, were analyzed but deemed less pertinent for Fred's current implementation due to constraints like computational cost or alignment with our objectives.

8.1 Selected Strategies

The following papers have strongly inspired the design of Fred.

Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models *[26/05/2023]* [10]
This study explores enhancements to zero-shot prompting for large language models (LLMs) in tackling multi-step reasoning tasks. While Zero-shot Chain-of-Thought (CoT) prompting has demonstrated effectiveness by encouraging models to "think step by step," it still struggles with calculation, missing-step, and semantic misunderstanding errors. To address these issues, the authors introduce Plan-and-Solve (PS) prompting, a method that first creates a structured plan dividing tasks into smaller steps, followed by a step-by-step solution process. An improved version, PS+ prompting, adds further detailed instructions to reduce errors and improve reasoning quality. This paper is one of the main pillars of the Fred architecture. The planning step is entirely inspired by this. It shows that LLM planning drastically improves reasoning capabilities. In the case of Fred, it is also used to benefit from the "separation of concerns," allowing experts to focus on more specific tasks.

ReAct: Synergizing Reasoning and Acting in Language Models *[10/03/2023]* [11]
This paper introduces ReAct, a method using large language models (LLMs) to generate interleaved reasoning and actions, enhancing task performance and interpretability. By combining reasoning traces with task-specific actions, ReAct improves over traditional methods in tasks like question answering and decision making, reducing hallucinations and error propagation, and outperforming baseline models in both success rate and human interpretability. ReAct is the major architecture that we chose to represent reasoning in the experts. It balances computational cost and reasoning capabilities, as well as accuracy and interpretability.

Large Language Models are Zero-Shot Reasoners *[29/01/2023]* [12]
This study demonstrates that large language models (LLMs) can achieve high performance in complex reasoning tasks using zero-shot prompting, simply by adding "Let's think step by step" before responses. This approach, called Zero-shot-CoT, significantly improves accuracy across diverse reasoning benchmarks without using task-specific examples. Results show marked performance gains on arithmetic, symbolic, and logical reasoning tasks, highlighting LLMs' potential for broad zero-shot reasoning capabilities. This paper was a major inspiration for Fred. The Zero-shot-CoT approach is directly leveraged in Fred's planning step to enhance reasoning.

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models *[10/01/2023]* [13]
This study demonstrates that using "chain-of-thought" prompting—providing examples of intermediate reasoning steps—significantly enhances large language models' reasoning abilities. Testing on three models showed marked improvements in arithmetic, commonsense, and symbolic tasks, with notable gains, such as state-of-the-art performance on the GSM8K math benchmark when using only a few exemplars. While not a direct inspiration for Fred's architecture, this paper shows that chaining reasoning steps improves reasoning capabilities. This supports the planning step in Fred and the use of each expert's response as input for the next expert.

Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents *[08/03/2022]* [14]
This paper explores whether large language models (LLMs) can generate actionable steps for high-level tasks in interactive environments. While previous methods relied on explicit, step-by-step training, the authors find that sufficiently large, pre-trained LLMs can break down complex tasks into intermediate plans when prompted correctly. This paper motivates the case for the planning step in Fred, showing that LLMs can generate plans that improve reasoning capabilities.

GPT3-to-plan: Extracting Plans from Text Using GPT-3 *[14/06/2021]* [15]
This paper examines using GPT-3 for extracting structured workflows from natural language descriptions in industries like finance. Initial results indicate that GPT-3 performs comparably to existing plan extraction methods, suggesting its potential for automating repetitive procedural tasks. This

was one of the first papers to show that LLMs can generate plans, providing some initial motivation for Fred.

8.2 Evaluated Papers

The following papers were evaluated but judged less directly pertinent for Fred's current architecture:

An LLM Compiler for Parallel Function Calling *[05/06/2024]* [16]
This paper introduces LLMCompiler, enabling parallel function execution through a three-component system. It significantly reduces latency and costs while improving accuracy compared to sequential methods. While this technique could allow faster and cheaper reasoning in Fred, its integration into the current architecture requires further evaluation.

Language Agent Tree Search (LATS): Unifying Reasoning, Acting, and Planning in Language Models *[06/10/2023]* [17]
This paper presents LATS, combining Monte Carlo Tree Search with LLM-driven value functions and self-reflection for decision-making. While promising, the computational cost is too high for Fred's current focus.

ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models *[23/05/2023]* [18]
ReWOO separates reasoning from external observations, enhancing computational efficiency. This approach could improve Fred's experts with lower computational costs, but the planning step would require significant changes.

Inner Monologue: Embodied Reasoning Through Planning with Language Models *[12/07/2022]* [19]
This research shows that LLMs can use natural language feedback to improve reasoning in embodied tasks. While potentially valuable for human-in-the-loop operations in Fred, this approach is not the current focus.

9 CONCLUSION

Fred simplifies Kubernetes management through a modular multi-agent architecture designed for specialized analysis, optimization, and decision-making. By addressing technical, theoretical, sustainability, and scalability challenges, Fred delivers actionable insights tailored to the complexities of Kubernetes environments.

Its collaborative framework, supported by human-in-the-loop interactions and enriched with contextual Facts, ensures that recommendations align with both operational and strategic goals. As Kubernetes continues to evolve, Fred's adaptability and modularity position it as a valuable tool for improving efficiency, sustainability, and innovation in cloud computing.

The integration of Facts not only improves immediate decision-making but also contributes to continuous improvement over time. As more contextual information is added, Fred's expert agents refine their understanding of the operational environment, leading to progressively better recommendations. This mechanism allows for dynamic updates to the system's context as business priorities, regulatory environments, or technical constraints evolve, ensuring that Fred's guidance remains relevant and up-to-date.

Fred exemplifies how generative AI can transform Kubernetes management, offering a scalable, future-ready solution for modern IT infrastructures. After one year of development, Fred is now used internally on real applications.

Acknowledgments

Thanks to the Thales Services Numériques technical direction for funding and support. Thanks to Thales Kast team for insights about Kubernetes intricaties. Thanks to the DIL unit to provide us with cloud resources.

References

1. Avst T and contributors. KoPylot: Open Source Generative AI for Kubernetes Workloads. `https://github.com/avsthiago/kopylot`. Web Resource. Accessed: 2024-11-18. 2023.
2. AI K. Kubert: Generative AI for Kubernetes Architectures. `https://mykubert.com/`. Web Resource. Accessed: 2024-11-18. 2023.
3. AI C. CAST AI: Kubernetes Optimization with AI-Driven Automation. `https://cast.ai/`. Web Resource. Accessed: 2024-11-18. 2023.
4. Yao S, Wu W, Liu J, et al. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. arXiv preprint arXiv:2305.04091 2023.
5. Huang TH, Wu L, Zhang X, and Cheng Y. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. arXiv preprint arXiv:2201.07207 2022.
6. Wei J, Wang X, Schuurmans D, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv:2201.11903 2023.
7. Li X, Zhang G, and Tang Y. Reducing Complexity in Large Language Models through Advanced Contextual Engineering. BME Frontiers 2023;2023:1–3.
8. Keda: Kubernetes Event-Driven Autoscaling. `https://keda.sh/`. Accessed: 2024-11-21.
9. Karpenter: Kubernetes Cluster Autoscaler. `https://karpenter.sh/`. Accessed: 2024-11-21.
10. Huang T et al. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. arXiv preprint 2023.
11. Yaodong L, Sun K, Lin Y, Dong N, He J, and Li J. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv preprint arXiv:2210.03629 2023.

12. Wei J et al. Large Language Models are Zero-Shot Reasoners. arXiv preprint 2023. Accessed: 2024-11-21.
13. Wei J et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint 2023. Accessed: 2024-11-21.
14. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. arXiv preprint 2022. Accessed: 2024-11-21.
15. GPT3-to-plan: Extracting Plans from Text Using GPT-3. arXiv preprint 2021. Accessed: 2024-11-21.
16. An LLM Compiler for Parallel Function Calling. arXiv preprint 2024. Accessed: 2024-11-21.
17. Language Agent Tree Search (LATS): Unifying Reasoning, Acting, and Planning in Language Models. arXiv preprint 2023. Accessed: 2024-11-21.
18. ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models. arXiv preprint 2023. Accessed: 2024-11-21.
19. Inner Monologue: Embodied Reasoning Through Planning with Language Models. arXiv preprint 2022. Accessed: 2024-11-21.