

Empowering Non-Technical Chatbot Designers Using MCP Servers



The Challenge

The current production chatbot is a **static system with fixed, pre-approved answers**, which limits its ability to handle **nuanced customer queries**.

The business wants to move to an **LLM-powered chatbot** while preserving the same level of **control, auditability and policy alignment** required.

Today, **Conversational AI engineers** design prompts and flows and then hand them over to **non-technical CX/chatbot designers**, who are responsible for **reviewing and approving responses**; this creates **slow feedback cycles and long iteration times to production**, making it hard to safely and quickly improve the bot.



The Solution

CX designers interact with the chatbot through a **no-code UI** that lets them **create and refine prompts**, while only the **system and engineering team can deploy changes to production**.

Behind the scenes, the system can **query recent chatbot conversations** from a **secure database**, without ever exposing **raw PII**. Access is controlled via **role-based access control (RBAC)**, ensuring designers see only **sanitized, policy-safe data and metrics**.

New changes are validated through an **LLM evaluation pipeline** – combining **LLM-as-judge scoring with golden records** – so CX designers can **iterate quickly and safely** without needing to **write any code**.

How did we solve this issue?

Give non-technical CX designers a way to shape the LLM Bot, while the system enforces guardrails, access control and pre-deployment evaluation.

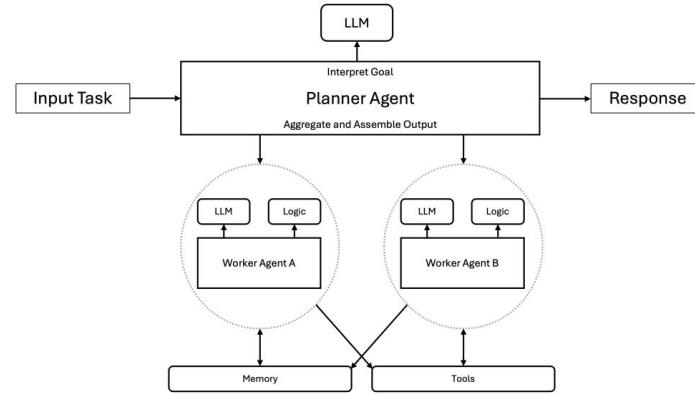
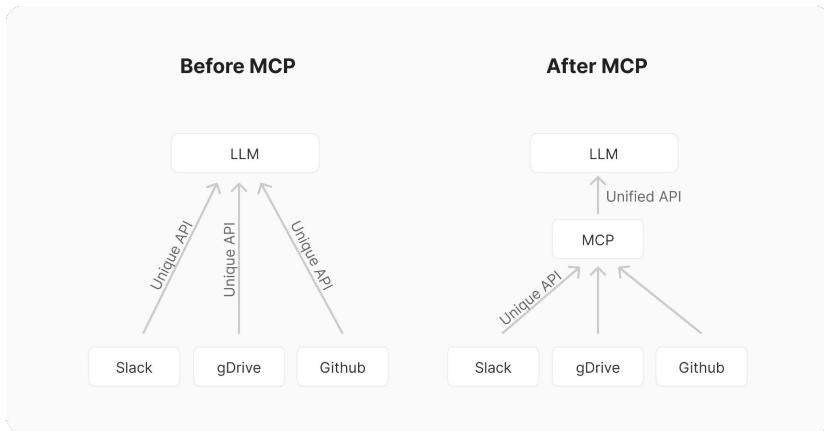
The **agentic orchestration layer** is introduced to align with the long-term strategy of the organisation: to democratise access to advanced AI tooling for CX designers and reduce the friction of the learning curve for non-technical users.

By adding more **MCP-based tools** into a common **ecosystem**, the architecture stays **flexible for future developments** and supports seamless integration of **new capabilities over time**. The layer also enforces **stronger, centralised access control**, ensuring that sensitive actions and data are always governed by **clear roles and permissions**.

Finally, it creates a foundation for **access to a wider range of current and future tools/technologies**, without needing to redesign the CX workflow each time something new is introduced.

Simplified system diagram

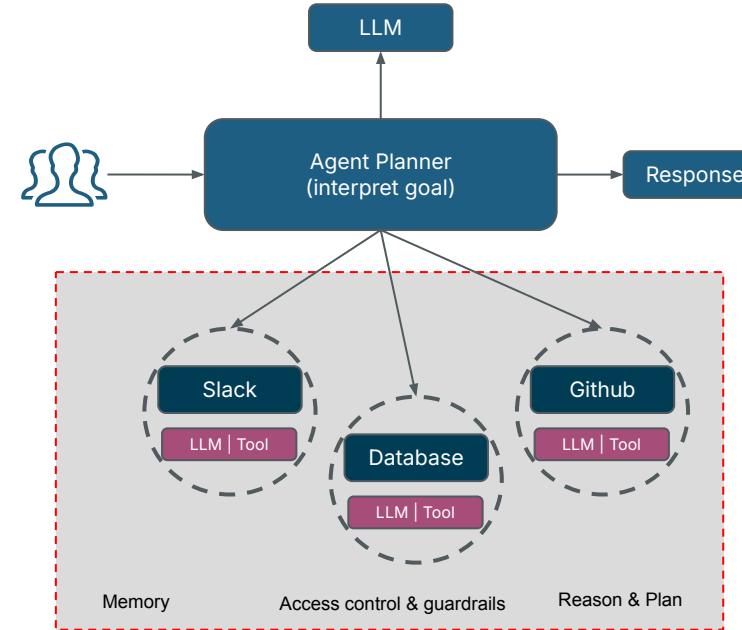
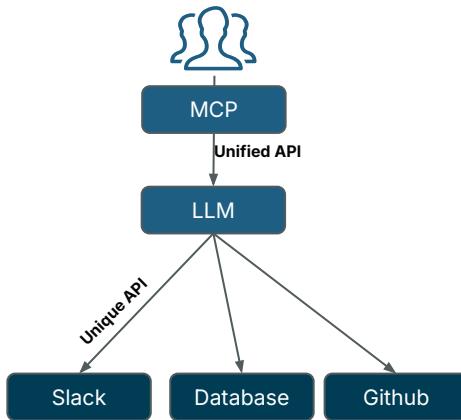
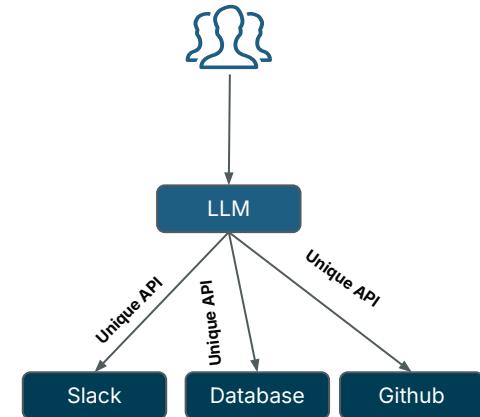
Before MCP, after MCP and with



Agent (Planner)
↓ Reason & Plan
LLM
↓ Unified API
MCP
↓ Unique APIs
Slack | gDrive | GitHub | Database | Search

Simplified system diagram and different approaches

Before MCP, after MCP and beyond



Before MCP

- Function call

After MCP

- Need to know tools to use

Agentic AI/System

What is Agentic AI?

Systems designed to act autonomously as agents on behalf of the **human users and other **AI systems****

Autonomy

They can operate without continuous human guidance

Goal orientation

They are programmed with clear objectives to achieve

Adaptability

They learn from their interactions and improve based on learning

Finding the Right Balance: LLMs & Agents

When building applications with LLMs, better to use simplest solution possible, and only increasing complexity when needed

It will be better combined approach with **LLMs with RAG** and **Agentic Workflows** to balance structured and dynamic decision-making, optimising cost, latency and efficiency.

	LLMs with RAG	Agentic Workflow	Full Agentic System
Task Complexity	Structured and highly controllable behaviour	Structured and generative processes	Dynamic decision-making & unpredictable processes
Latency	< 15 seconds	> 15 seconds	> 30 seconds
Value of Work	< \$0.10	> \$1	> \$1
Business risk tolerance	Minimal	Moderate	High

Best for

Simple, structured tasks

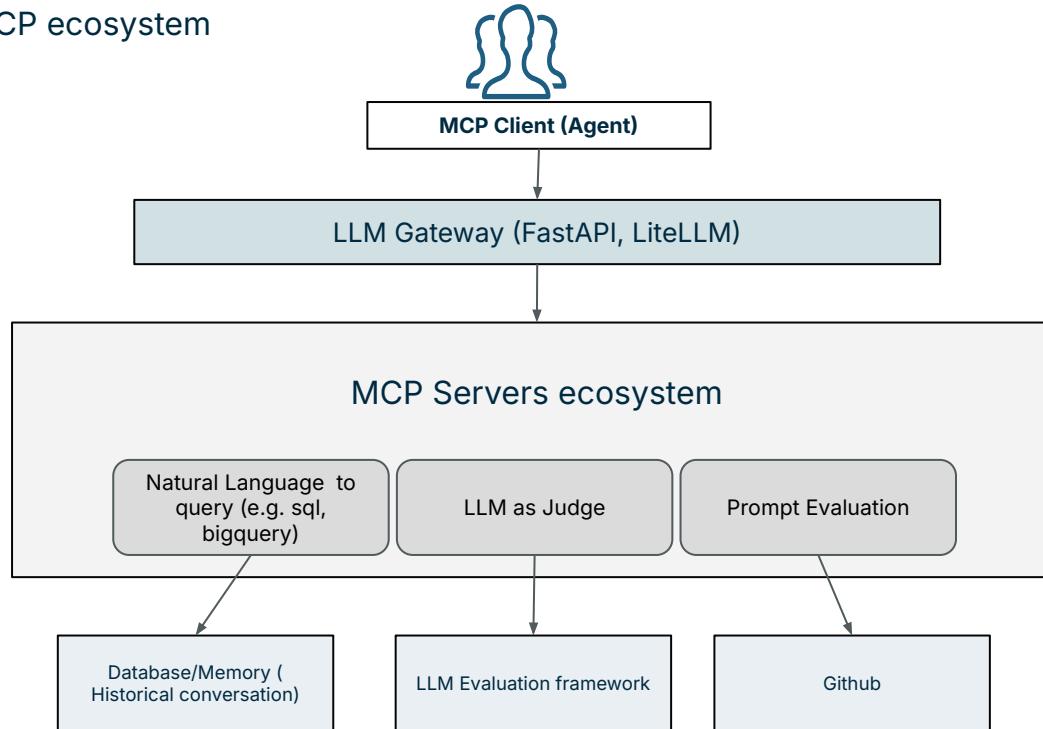
Enhancing deterministic workflows with creative input/output

Complex tasks, fuzzy business logic, high risk tolerance, high-value decisions

Increased Autonomy, Cost & Latency →

HLD of deployed system

Logical components of the MCP ecosystem



Demo

Deployment – MCP Servers & Planner Agent

AWS lambda hosted MCP servers

All the MCP servers are deployed ad as AWS Lambda function.

- tool_manifest.json
- yaml config file with allowed tools

Planner agent as MCP client in local machine.

Other way to deploy MCP servers and agent in AWS

Option 1

Planner Agent and MCP servers as Lambda

- Deploy the planner as its own **Lambda function** and expose it via a **REST API**.
- Easy to integrate with existing **microservices**, front-ends, or internal systems.

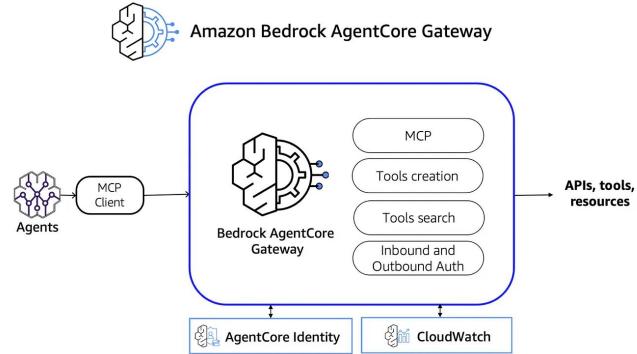
Option 2

Planner Agent on Amazon Bedrock AgentCore

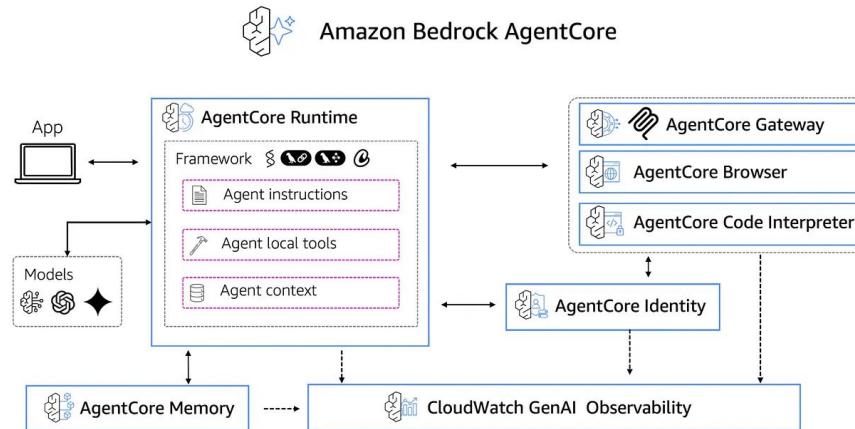
- Run the planner as a **fully managed agent** in **Amazon Bedrock AgentCore**.
- Offloads **infrastructure management**, provides **native integration** with Bedrock models and tools, and standardises the **agent lifecycle**.

Latest in AgentCore

AgetnCore Gateway and AgetnCore Identity



- For identity and access control we used open source LiteLLM as gateway which require a really significant engineering effort.
- Bedrock AgentCore Gateway which supports all the features and more out of box.



Thank you