Dissecting the Agentic Stack

Threats Layer by Layer

About me

Miguel Fontanilla
Platform Engineering Lead @ sennder

AWS:
- Community Builder (Containers)
- AWS Certified Professional Architect
- AWS Certified Professional GenAI Developer

Platform Engineering Ambassador

CK{A,AD}

Github: mifonpe
Web: kubesandclouds.com
Linkedin: www.linkedin.com/in/phontee/

sennder

- Europe’s leading Digital Freight Forwarder
- Platform Engineering paradigm
- Agents
  - Developer self-sufficiency
  - Code Quality
  - Incident Management

Agenda

01 Agent Architecture

02 Frameworks

03 Threats and Mitigations

04 Conclusions

01

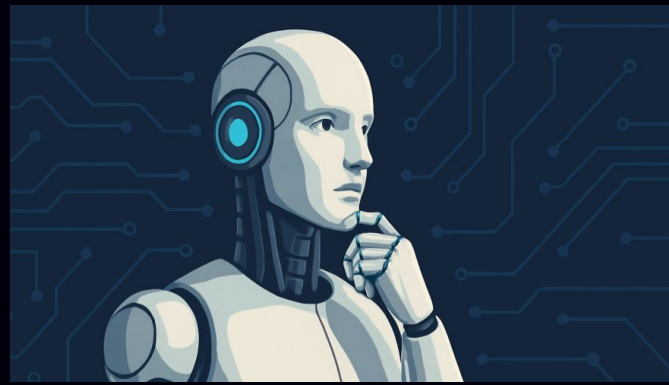
Agent Architecture

More than LLMs

AI Agents

"An AI agent is an autonomous software system that leverages LLMs to reason, plan, and act on its own to achieve specific goals, rather than just generating text like a chatbot. They use tools, remember context, and work independently to perform complex, multi-step tasks"

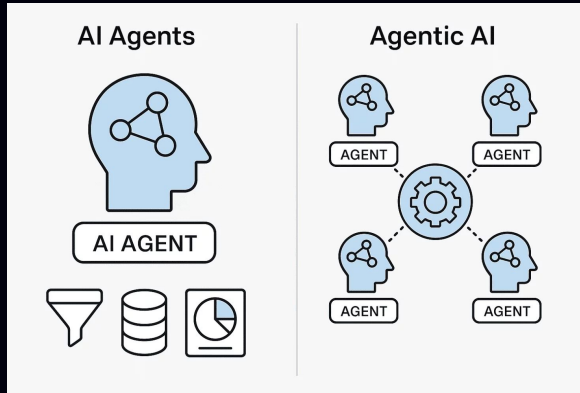
Google



Agentic AI

"Agentic AI is an artificial intelligence system that can accomplish a specific goal with limited supervision. It consists of AI agents that mimic human decision-making to solve problems in real time"

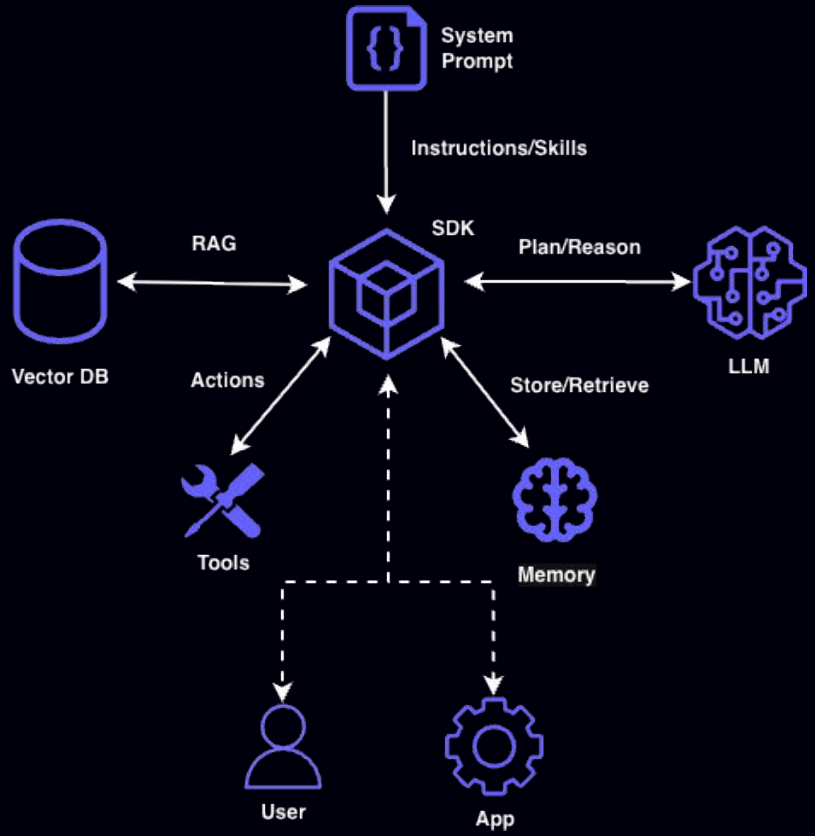
IBM



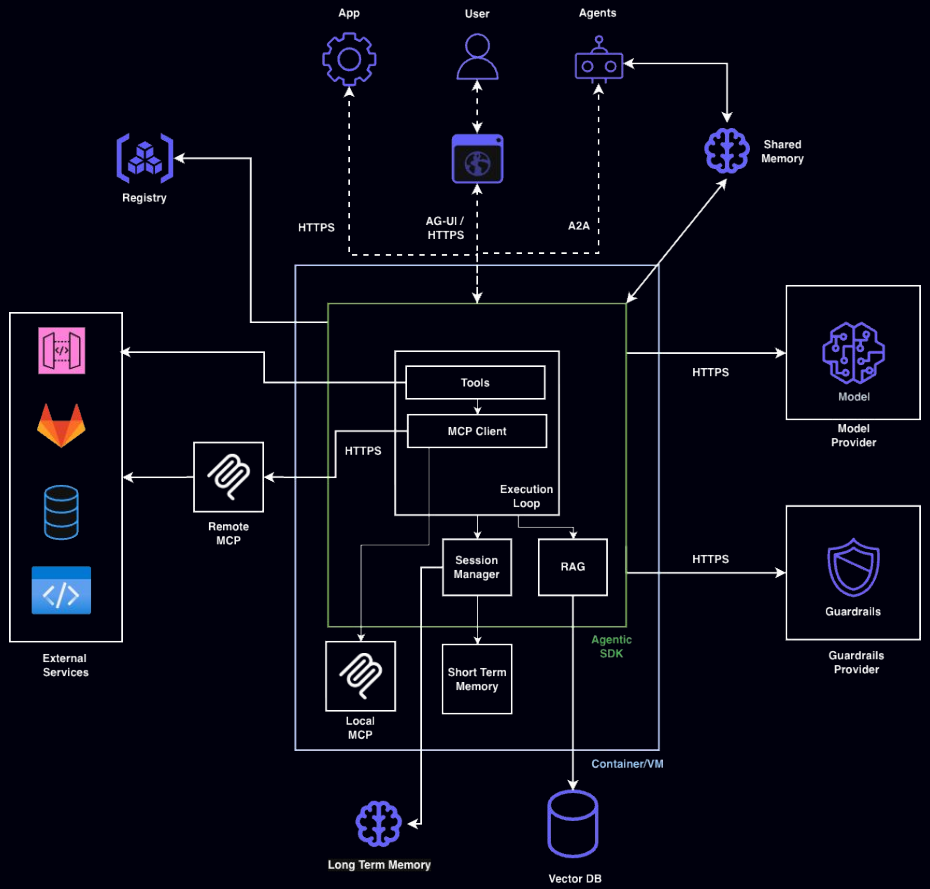
Architecture

- LLMs → Planning/Reasoning
- SDKs/Framework → Adaptative/Proactive
- Prompts/Instructions → Goals
- Memory
- Tools (MCP)
- Collaboration (A2A, AG-UI)

Architecture (High Level)



Architecture



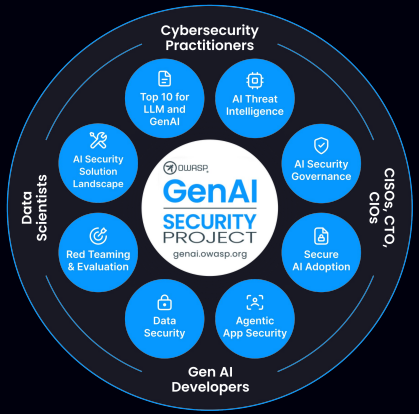
02

Frameworks

Modeling Agentic AI Threats

Frameworks & Resources

- OWASP GenAI
- Cloud Security Alliance
  - MAESTRO

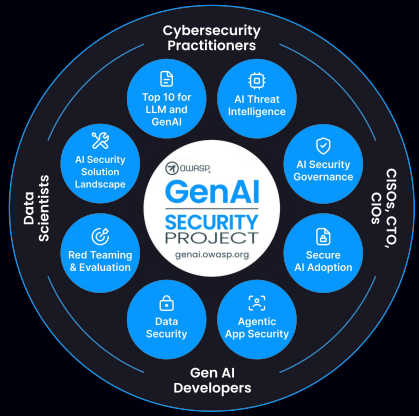


OWASP GenAI

- Top 10 for Agentic Applications
- Agentic AI Threats and Mitigations

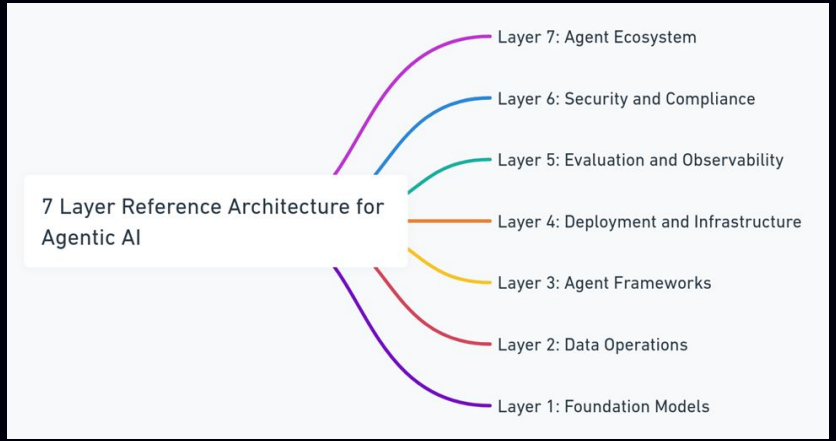
Other:

- Top 10 for LLM Applications
- Practical Guide for MCP server development
- Practical Guide for Securely Using Third-Party MCP Servers
- Data Security Risks & Mitigations 2026



MAESTRO

- Layered threat model
- Focus on Agentic Landscape
- Cross-Layer interactions
- Resources
  - Paper
  - Analyzer
  - Review

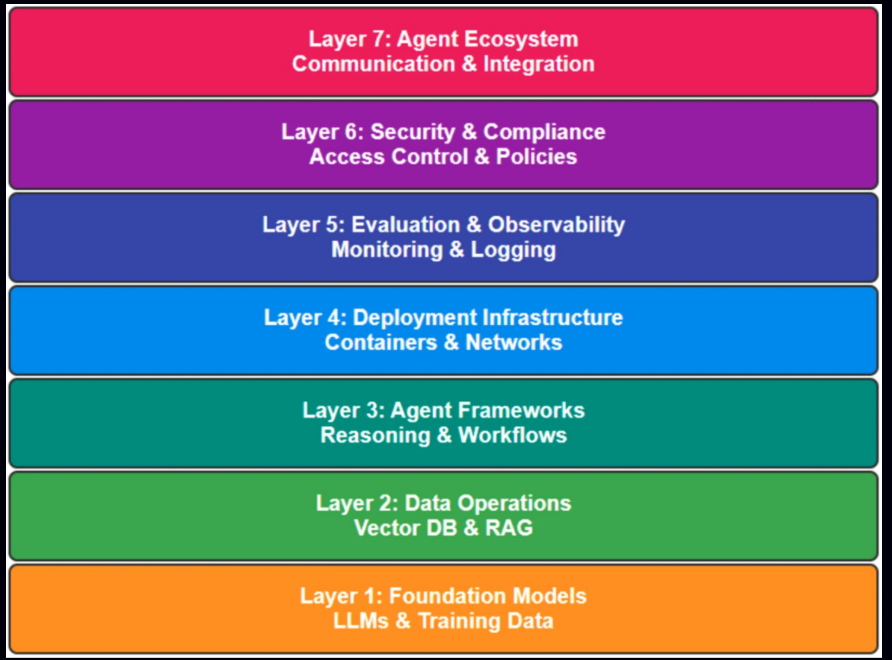


03

Threats & Mitigations

Threats & Layers

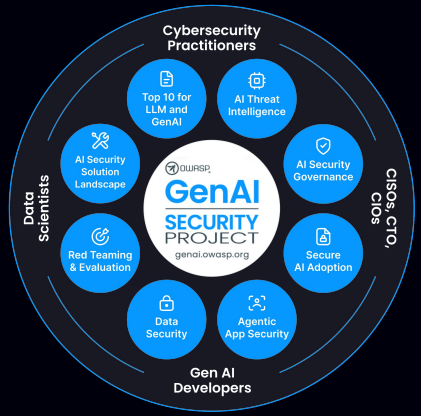
- L1
  - Training data
  - Weights
- L2
  - RAG
  - KBs
  - Memory
  - Embeddings
- L3
  - SDKs
  - Agent Frameworks
- L4
  - Runtimes
  - Network
  - Orchestration
- L5
  - Audit
  - Logging
  - Behavior monitoring
  - HITL
- L6
  - IAM
  - Guardrails
  - Compliance
- L7
  - Protocols
  - Discovery
  - Governance



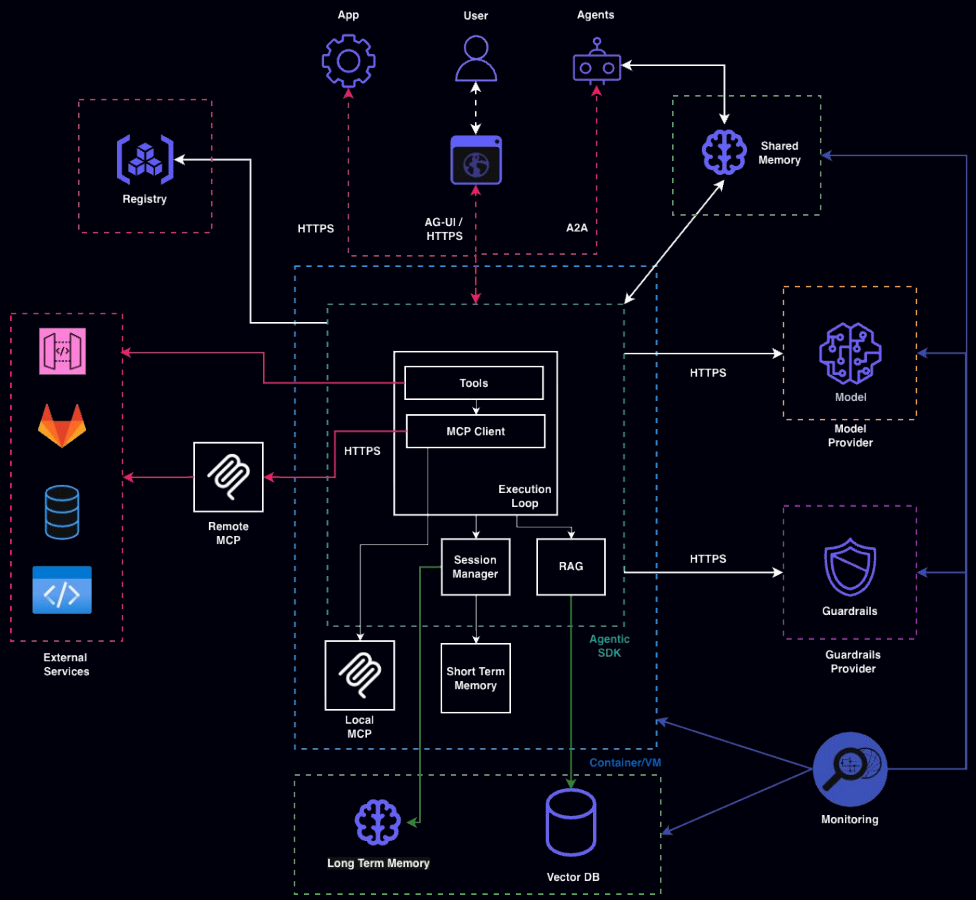
https://github.com/Chainlit/chainlit

Threats & Layers

- Agent Goal Hijack
- Tool misuse
- Identity & Privilege Abuse
- Unexpected Code Execution
- Context & Memory Poisoning
- Insecure inter-agent communication
- Cascading failures
- Agent Identity compromise
- Rogue Agents
- Overwhelming HITL
- Resource overload
- Supply Chain attacks
- ...



Architecture



Layer 7: Agent Ecosystem
Communication & Integration

Layer 6: Security & Compliance
Access Control & Policies

Layer 5: Evaluation & Observability
Monitoring & Logging

Layer 4: Deployment Infrastructure
Containers & Networks

Layer 3: Agent Frameworks
Reasoning & Workflows

Layer 2: Data Operations
Vector DB & RAG

Layer 1: Foundation Models
LLMs & Training Data

Agent Goal Hijack

- Threat
  - Attacker-controlled prompts, files, emails, RAG data, or tool outputs redirect the agent's goal
  - The agent may silently change its plan, scope, or task priorities
  - Multi-step autonomy can turn one injected instruction into real system actions
- Impact
  - L3 → Agent decision making compromised
- Remediation
  - L3 → Lock goals, system prompts, and permitted actions through controlled configuration
  - L5 → Validate user and agent intent before high-impact or goal-changing actions*
  - L6 → Treat all external and natural-language inputs as untrusted*

https://github.com/ChainIt/chainIt

Tool misuse

- Threat
  - Legitimate tools within their existing permissions
  - Can delete data, send messages, trigger refunds, or exfiltrate information
  - Tool chaining can combine safe-looking actions into harmful workflows
- Impact
  - L3 → Unauthorized API Calls
  - L4 → Infrastructure tool exploitation
  - L7 → Malicious use of external services
- Remediation
  - L5 → Require approval for destructive or high-impact actions
  - L6 → Least privilege principle for tools*
  - L7 → schemas, rate limits, and egress controls before execution*

Context & Memory Poisoning

- Threat
  - Attackers poison memory, RAG stores, embeddings, summaries, or shared context
  - Poisoned memory persists and influences future planning or tool use
  - Shared memory can spread bad context across agents, users, or tenants
- Impact
  - L1 → Corrupted Training Data (if used)
  - L2 → Poisoned Vector embeddings
  - L3 → Contaminated decision history
- Remediation
  - L2 → Validate and scan memory writes before storing them
  - L2 → Segment memory by user, tenant, task, and trust level
  - L2 → Track provenance, expire low-trust entries

https://github.com/Chainlit/chainlit

Insecure A2A Communication

- Threat
  - Weak agent-to-agent communication enables spoofing, tampering, replay, or interception
  - Attackers can forge agent identities or poison discovery and routing
  - Modified messages can change intent, context, or delegated authority
- Impact
  - L4 → Communication interception
  - L5 → HITL Bypass/Logging modification
  - L6 → IAM Bypass/Privilege escalation
- Remediation
  - L4 → Use mutual authentication, encryption, and signed messages
  - L5 → Add timestamps, session IDs, and replay protection
  - L7 → Enforce typed schemas, protocol pinning, and verified agent registries*

Identity & Privilege Abuse

- Threat
  - Agents inherit excessive privileges from users, parent agents, or workflows
  - Cached credentials or memory can leak across sessions or users
  - Low-privilege agents can abuse trust to trigger high-privilege actions
- Impact
  - L4 → SA compromise & Container escape
  - L6 → IAM Bypass/Privilege escalation
  - L7 → Unauthorized access to external services
- Remediation
  - L6 → Use distinct per-agent identities and short-lived, task-scoped credentials
  - L5 → Re-authorize every privileged action and context switch
  - L2 → Isolate memory per session
  - L3 → Isolate credentials, and execution context per task or session

https://github.com/Chainlit/chainlit

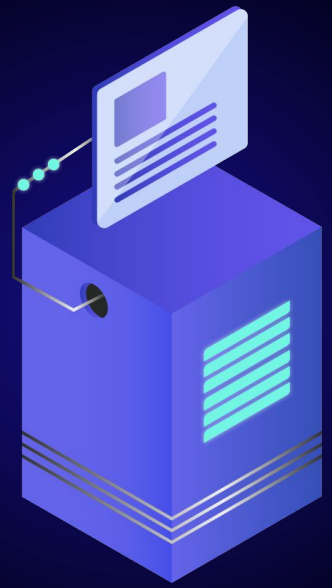
Unexpected Code Execution

- Threat
  - Prompt injection can make agents execute shell commands, scripts, or unsafe code.
  - Generated code may contain backdoors, vulnerable packages, or unsafe deserialization
  - Multi-tool chains can escalate into RCE or host compromise
- Impact
  - L4 → Container escape & Host Compromise
  - L6 → IAM Bypass/Privilege escalation
- Remediation
  - L3 → Separate code generation from execution with validation gates
  - L4 → Isolated sandboxes with strict network and filesystem limits

04

Conclusions

Lessons Learned



Conclusions

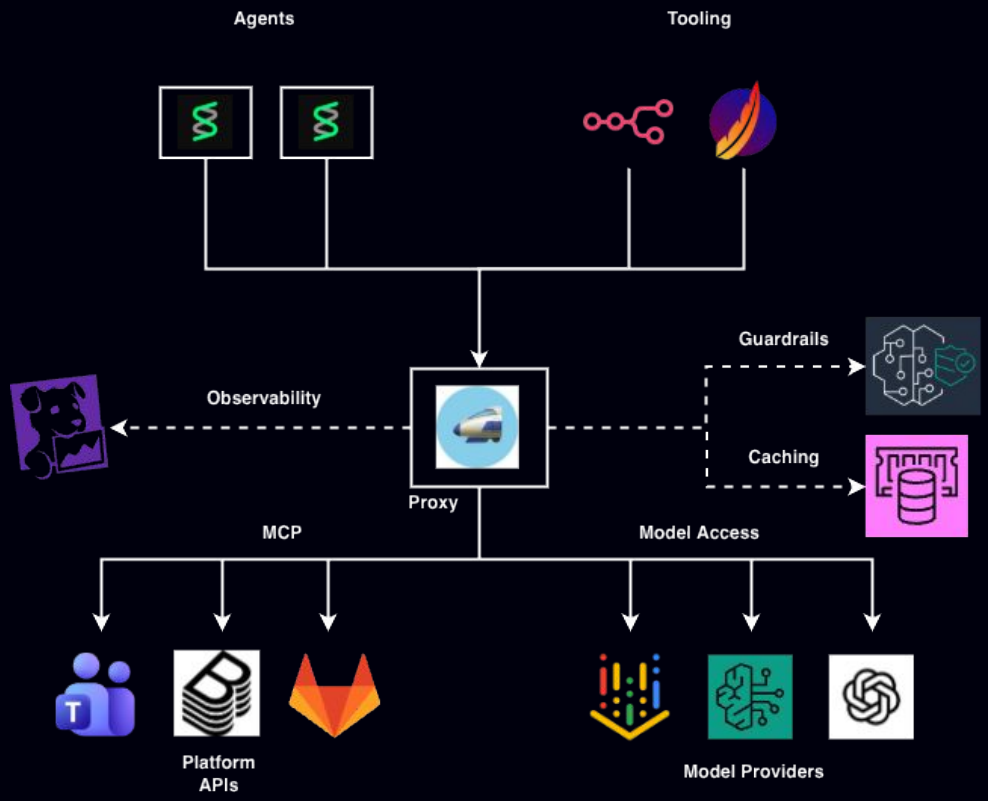
- Agentic Systems attack surface is complex
  - Multi Layer
  - Multi Agent
  - Multi Step
  - Emergent Behaviors
- Traditional Threat Modeling frameworks fall short
- MAESTRO
  - Layered
  - From models to ecosystems
  - Cross-layer dependencies
  - Expandable

https://github.com/Chainlit/chainlit

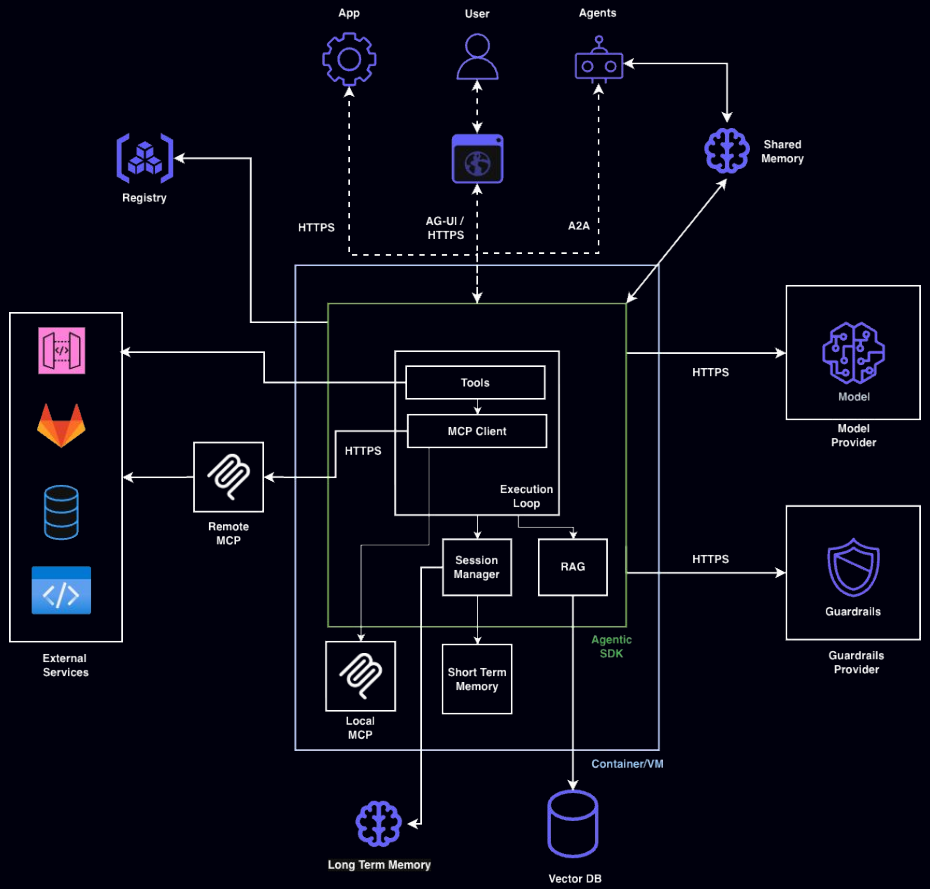
Proxies/Gateways

- LLM Observability
- Token consumption
- Model Router/Gateway
- MCP Gateway
- Access Control Granularity
- Rate limiting
- Caching
- Budgets
- Guardrails

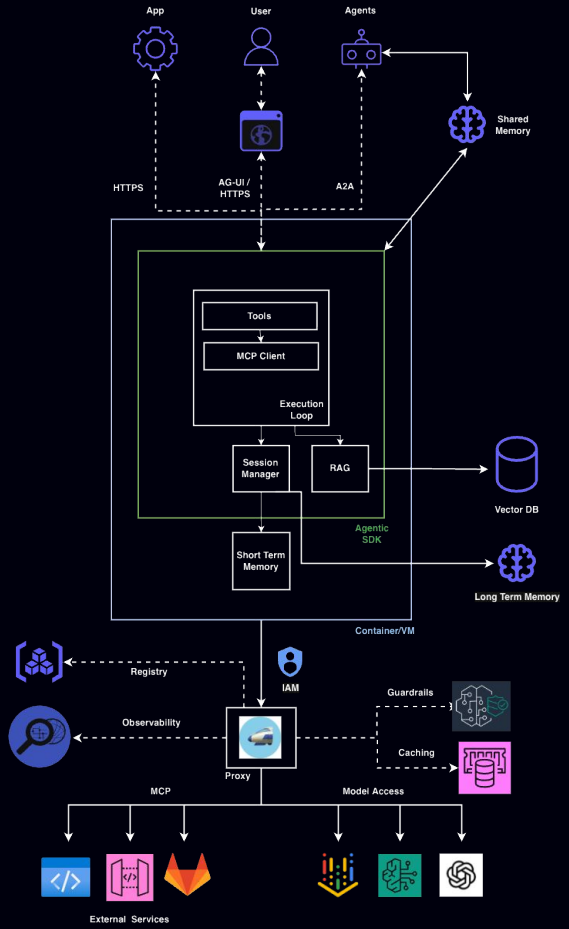
Proxy/Gateway



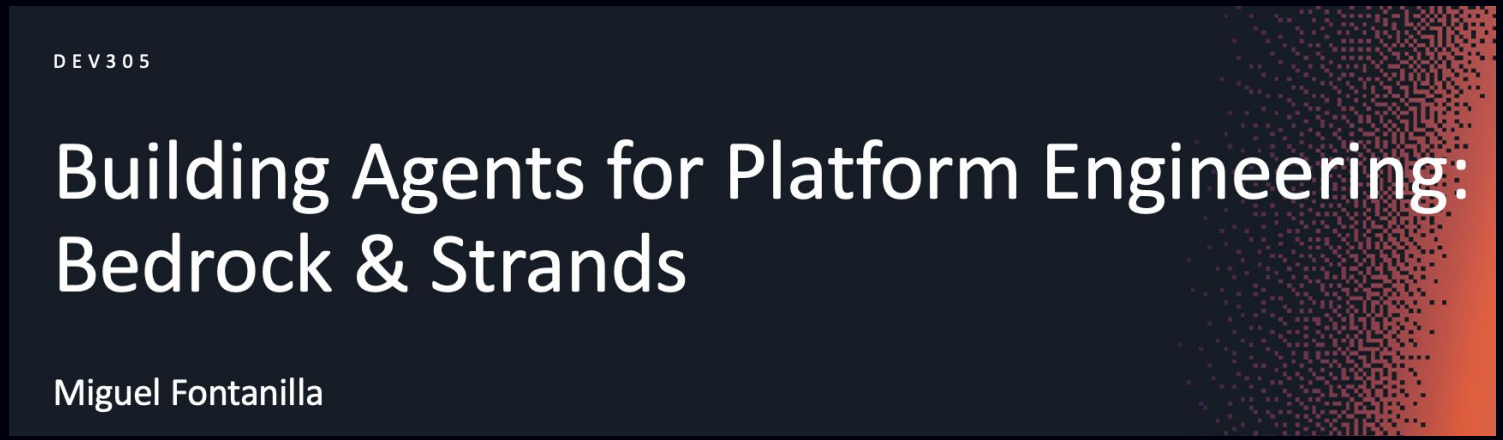
Architecture



Proxy/Gateway



One More Thing



See you in the AWS Summit!

Q&A

Thank You!

Repo

LinkedIn

Talks

Template: SlidesGo Icons: FlatIcon