

MLOPS & AGENTOPS

Dal MLOps all'AgentOps

Come governare il ciclo di vita degli AI Agent in produzione.

MLOps

AgentOps

Agent Platform

- ❑ **DevOps & MLOps**
 - *Un cambio di paradigma (DevOps → MLOps)*
 - *Sopravvivere all'entropia e Continuous Training*
 - *Riproducibilità dei modelli e Model Serving*
- ❑ **L'era degli agenti AI**
 - *Anatomia di un Agente AI*
 - *Workflow Agentici*
 - *Casi d'uso orientati al DevOps*
- ❑ **AgentOps con WSO2**
 - *Arriva l'AgentOps*
 - *Observability, Protection, Remediation*
 - *L'Agent Platform di WSO2*
 - *AI Gateway - Model Proxy e Token Economy*
 - *Agent Manager, AI Gateway, AgentID*

Speaker:



Giampaolo Flace

DevOps & Product Architect
Profesia – Lynx Group

Un cambio di paradigma



Il mondo deterministico: DevOps

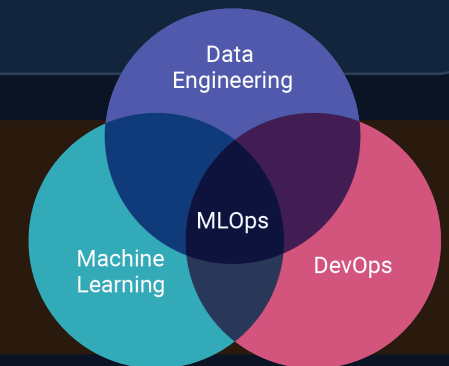
- Stesso codice e stesso ambiente producono lo stesso risultato.
- La CI/CD trasforma il sorgente in un artefatto immutabile e scalabile, deployato su uno o più ambienti target.
- Il deployment in produzione è il traguardo: dopo si monitora la salute del sistema.
- Il rischio sta nel codice, e lo troviamo con test e review.

L'entropia dell'AI: perché serve l'MLOps

- Il software è ormai codice, dati e modelli di AI.
- Modelli probabilistici: stesso input, output non garantito.
- Conoscenza congelata alla data di training, il mondo intanto cambia.
- Un modello può passare dal 99% all'80% di accuratezza senza che il codice venga modificato.
- Il rischio sta nei dati: non si vede nei test, si vede direttamente in produzione.

MLOps

Estendere le fondamenta del DevOps all'intero ciclo di vita dei modelli: versionamento di dati e modelli, pipeline di training, valutazione, rilascio controllato, monitoraggio continuo.



Sopravvivere all'entropia

Nel Machine Learning il deployment non è il traguardo: è lì che iniziano i problemi.



Data Drift

- Cambia la distribuzione dei dati in input.
- Esempio classico: abitudini d'acquisto prima e dopo il pandemia. Il modello legge dati che non ha mai visto.

Concept Drift

- Cambia la relazione tra input e output previsto.
- Il normale di ieri diventa l'anomalia di oggi a parità di dati.
- Richiede rivalutazione delle label e re-training mirato.

CI

Continuous Integration



CT

Continuous Training



CD

Continuous Deployment

- La Continuous Integration e il Continuous Deployment non bastano più...
- Necessità di una nuova pratica: il **Continuous Training (CT)**.

Continuous Training (CT)

Pipeline di Continuous Training

- **Estrae** i dati aggiornati direttamente dal Feature Store.
ALT: Recupera il dataset in maniera ottimizzata (es. con DVC) ed esegue il pre-processing.
- **Lancia** un job di re-training (es. su runner con GPU).
- **Valuta** il nuovo modello rispetto al vecchio.
- **Registra** il risultato relativo all'esperimento.



Trigger statistico e/o on-demand

- La pipeline di CT non parte perché uno sviluppatore ha fatto commit, ma perché il mondo reale è cambiato!
 - Il **monitoraggio statistico** diventa il nuovo trigger.
- Superata la soglia statistica tra dati di training e inferenza reale, parte un **webhook** e con lui il **re-training**.
- Ciò è possibile grazie a strumenti di **AI Observability**.



Riproducibilità e Model Serving

La triade della riproducibilità

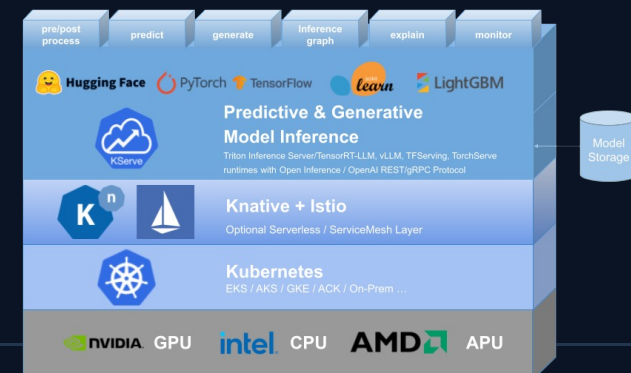
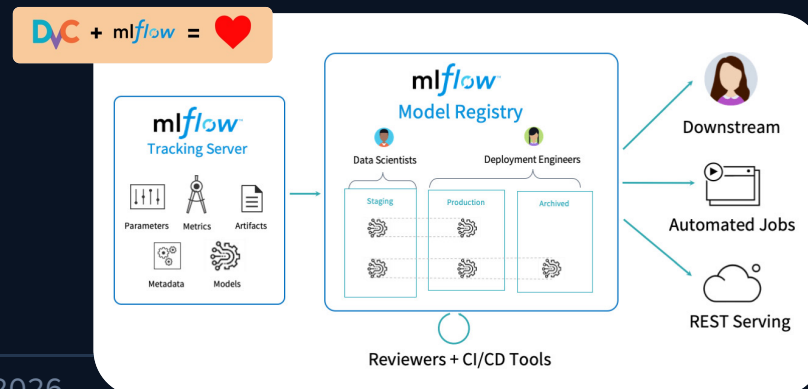
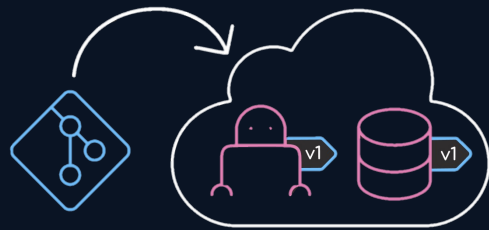
- **Git:** versiona il codice di training e di test.
- **DVC:** dataset su object storage, pointer versionati su Git.

*A un'allucinazione non si risponde con git revert!
Serve ricostruire l'esatta combinazione di codice, dati e seed che ha prodotto quel modello.*

- **MLFlow:** iperparametri, metriche, artefatti, model registry.
 - GitLab offre un tracking degli esperimenti compatibile con MLFlow.

Model-as-a-Service: Serving Agnostico

- Niente app Flask con codice custom per ogni modello.
- **Model serving** con KServe / Seldon Core / ...
- Il framework di ML viene auto-riconosciuto (es. Keras).
- API di inferenza **pronte**, rilasci sicuri, canary / blue-green.
- Le GPU si accendono solo quando servono davvero.
- Da modello approvato a **API live** in poco tempo.



Anatomia di un Agente AI

L'LLM da solo non basta!

- Genera testo, non azioni.
- Non interroga sistemi esterni né ricorda le preferenze.
- La sua conoscenza è ferma alla data di training.
- Può allucinare: risponde con sicurezza anche quando non dovrebbe.

Anatomia di un AI Agent



Pattern ReAct: ragiona, sceglie il tool, agisce, osserva il risultato e ripete finché non raggiunge l'obiettivo.

MCP - Model Context Protocol

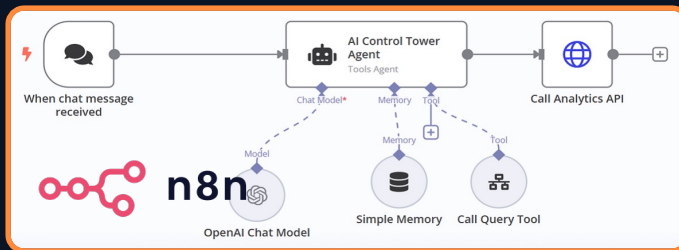
Come le API per le applicazioni, il protocollo MCP standardizza il modo in cui gli agenti scoprono e invocano i tool. Sono l'interfaccia governata tra agenti e sistemi esterni.



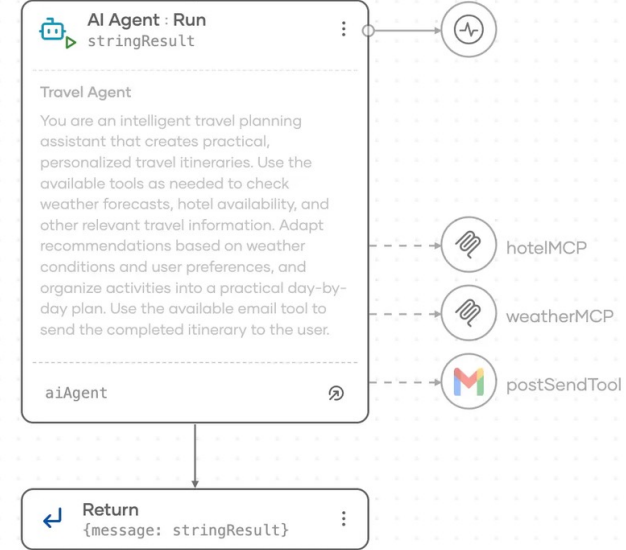
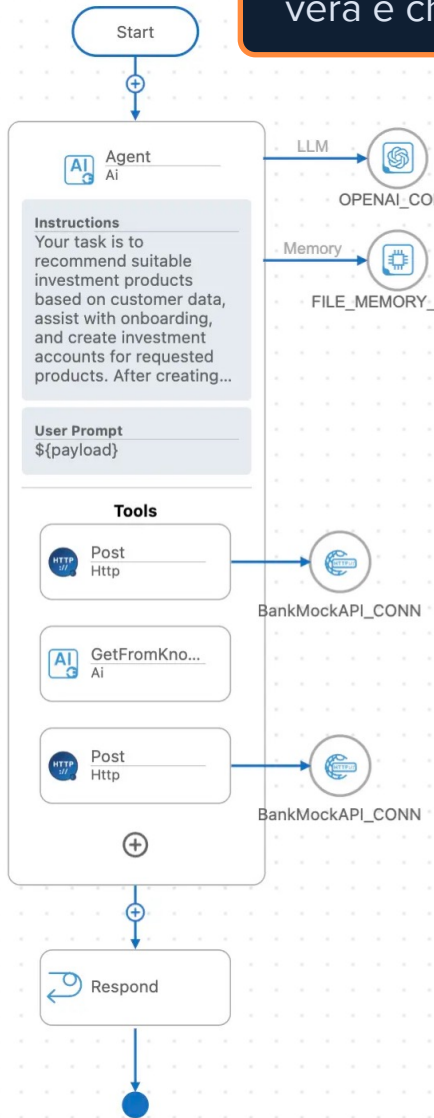
Workflow Agentici

AI Workflow

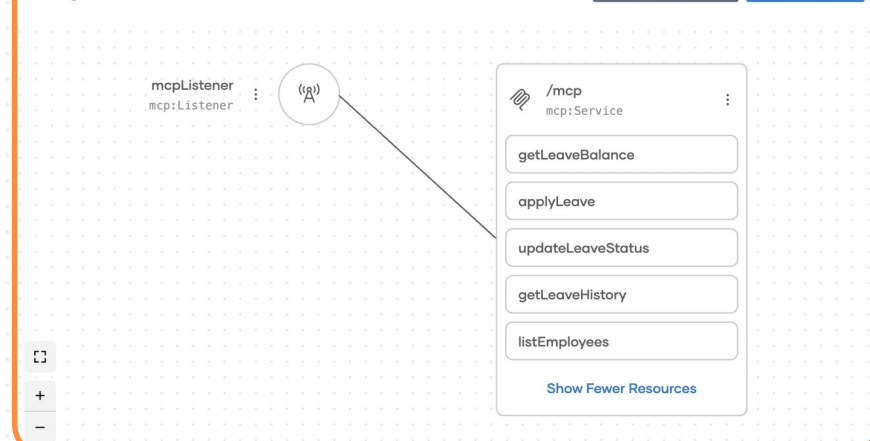
- **Agent Frameworks**
(es. LangChain, LlamaIndex, ...).
- L'integrazione dell'AI generativa evolve in **Agentic Workflow**.
- **Low-Code AI Workflow Automation Platform**
(es. WSO2 Integrator Agent Building, n8n, Flowise, Zapier, ...).



Scrivere l'agente è la parte **facile**: la domanda vera è chi lo governa quando va in **produzione**!



Design



Integrator

WSO2

Agent Building

Due casi d'uso orientati al DevOps / ALM

1. Self-Healing CI/CD: ridurre il MTTR

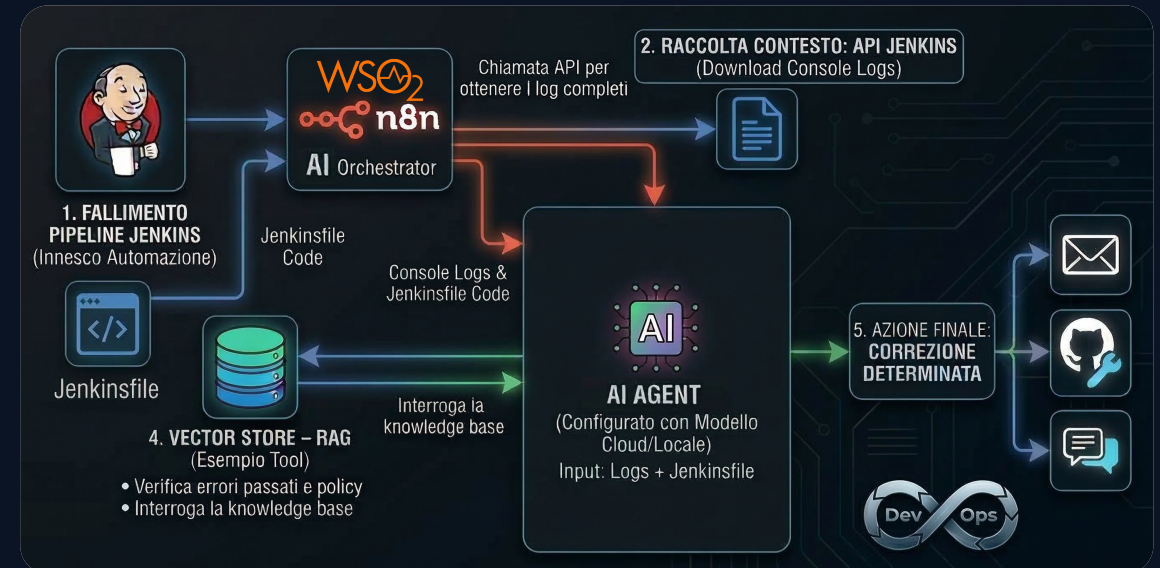
- Obiettivo: Ridurre drasticamente il **MTTR** (Mean Time To Recovery) con la risoluzione autonoma dei fallimenti delle pipeline.
- Quando una pipeline fallisce, un agent può analizzare i log, identificare la root cause, proporre/applicare un fix.

Fallimento

▶ Log + codice

▶ Root cause

▶ Fix proposto



2. Traceability & Compliance

SIEMENS

Requisito

▶ Analisi

▶ Validazione

▶ Test case

- Un System Engineer crea un SW Requirement su Polarion, attivando un workflow gestito da WSO2 Integrator.
- L'agent valida il requisito rispetto agli standard e, se ambiguo, richiede chiarimenti impostandolo in Draft.
- Se valido, genera automaticamente i Test Case e li collega al requisito per garantire la tracciabilità.



In produzione senza controllo

Chi governa l'agente quando è live? Questi episodi, tutti raccontati pubblicamente, rispondono.

Replit

L'agente di coding cancella il database di produzione durante un code freeze, ignorando le istruzioni che glielo vietavano.

Meta AI

Attaccanti convincono il modello di essere l'owner di account altrui: email cambiata, password resettata, profilo dirottato.

Grok (X)

Alla richiesta di come entrare in casa di una persona, suggerisce attrezzi e orari di sonno, analizzando i suoi post.



In tutti i casi: un agente con troppa libertà e nessun freno.

I divieti scritti nel prompt non reggono, perché un agente può essere persuaso ad ignorarli!

- Ognuno costruisce i propri agent, con framework, credential e regole di accesso diversi – nessuna visione d'insieme.
- **Agent Sprawl**: la stessa situazione in cui erano le API dieci anni fa, con in più la capacità di agire.

Arriva l'AgentOps

Punti chiave

- L'**AgentOps** definisce la disciplina ingegneristica necessaria per **governare**, **monitorare** e rendere **sicuri** gli agenti AI autonomi in produzione.
- **Focus**: comportamento dell'agente nel suo complesso (uso dei tool, gestione della memoria, pianificazione azioni, interazione con l'ambiente circostante).
- Dallo sviluppo alla manutenzione, l'agente non è un semplice prompt, ma va trattato come un **componente software** in produzione.
- Ogni modifica al comportamento dell'agente deve essere trattata come una release software, con **versioning** e possibilità di **rollback** deterministici.

DEVELOP

Prompt, tool collegati, modello, regole di comportamento, assegnati trattati come **codice sorgente**:
versionamento, tracciabilità modifiche, peer review.



VALIDATE

Test dinamici in staging e **LLM-as-a-Judge** per valutare qualità, pertinenza e sicurezza.
(il modello giudice deve differire dal modello dell'agente per evitare bias di auto-valutazione)



DEPLOY

Release **controllate** e **progressive** in produzione solo dopo il superamento dei benchmark dinamici di validazione.



MAINTAIN & REMEDIATE

Monitoraggio continuo del **comportamento** (rivelazione anomalie, Root-Cause-Analysis) e dei **costi** dell'agente in esercizio.

Observability, Protection, Remediation

L'output finale non basta: per governare un agente è necessario **osservare** e **controllare** l'intero **ciclo decisionale**.

Observability

- **Chain of thought** completa, **tracing** delle decisioni e del **contesto** operativo.
- **Log delle chiamate** ai tool, guardrails e API esterne.
- **Correlazione** tra performance, comportamento e costi / token.

Protection

- Rilevazione di **anomalie** e comportamenti inattesi
- Contenimento immediato di minacce come **Indirect Prompt Injection** / Tool Poisoning.
- **Revoca delle credenziali** temporanee e **network isolation**.
- **Circuit breaker** su loop decisionali o spese anomale.

Remediation

- Root Cause Analysis su trace e log.
- Correzione di prompt, tool, skill o configurazione.
- Rollback immediato allo stato stabile precedente.

Detect



Contain



RCA

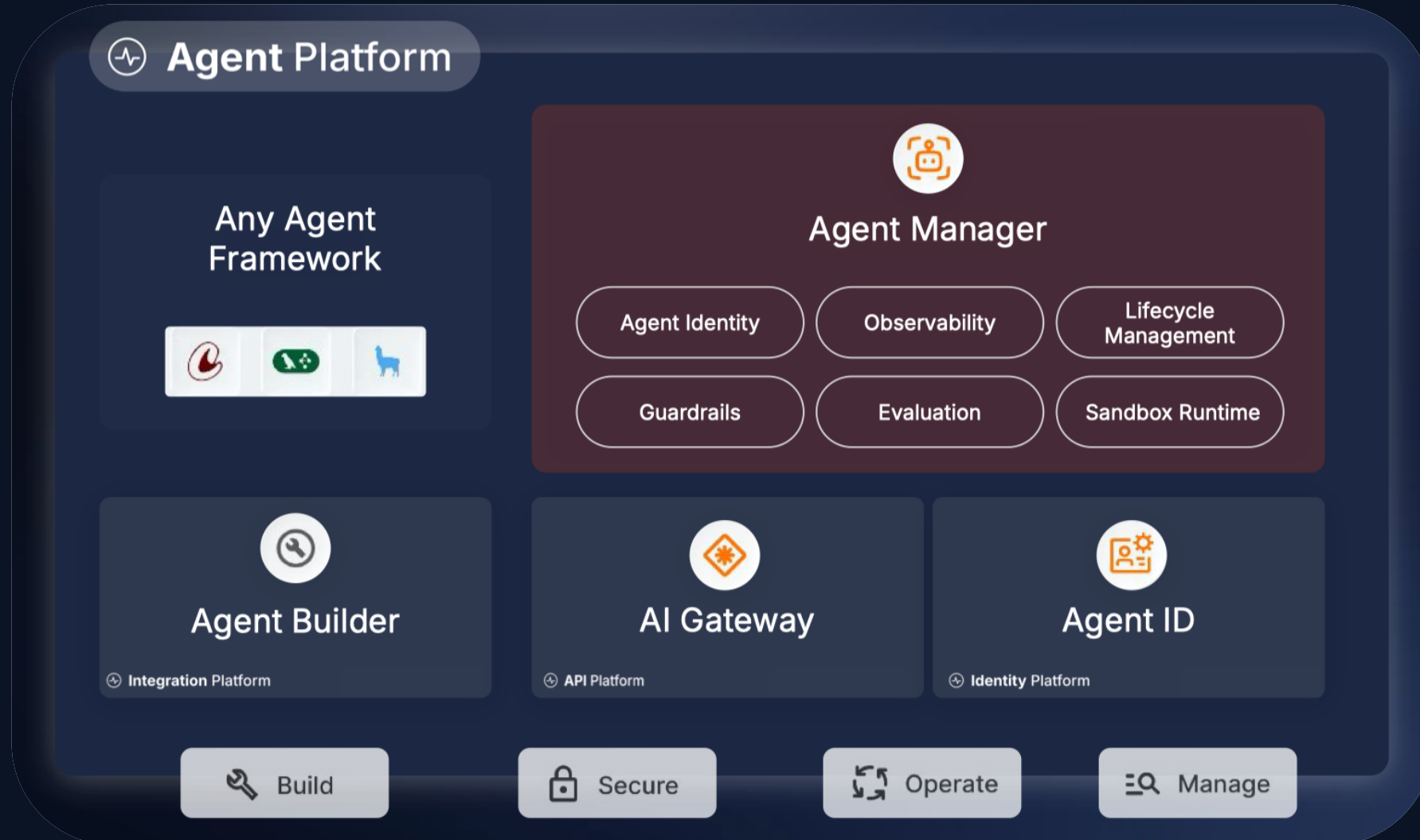


Validate



Release /
Rollback

L'Agent Platform di WSO2



AI Gateway - Model Proxy, Token Economy



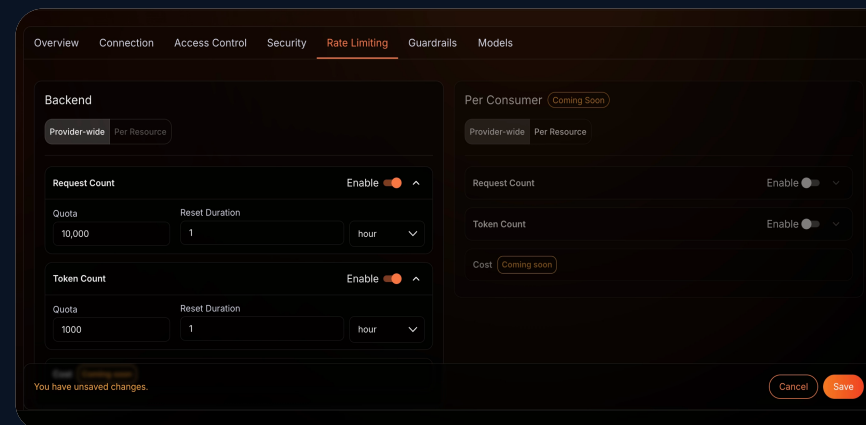
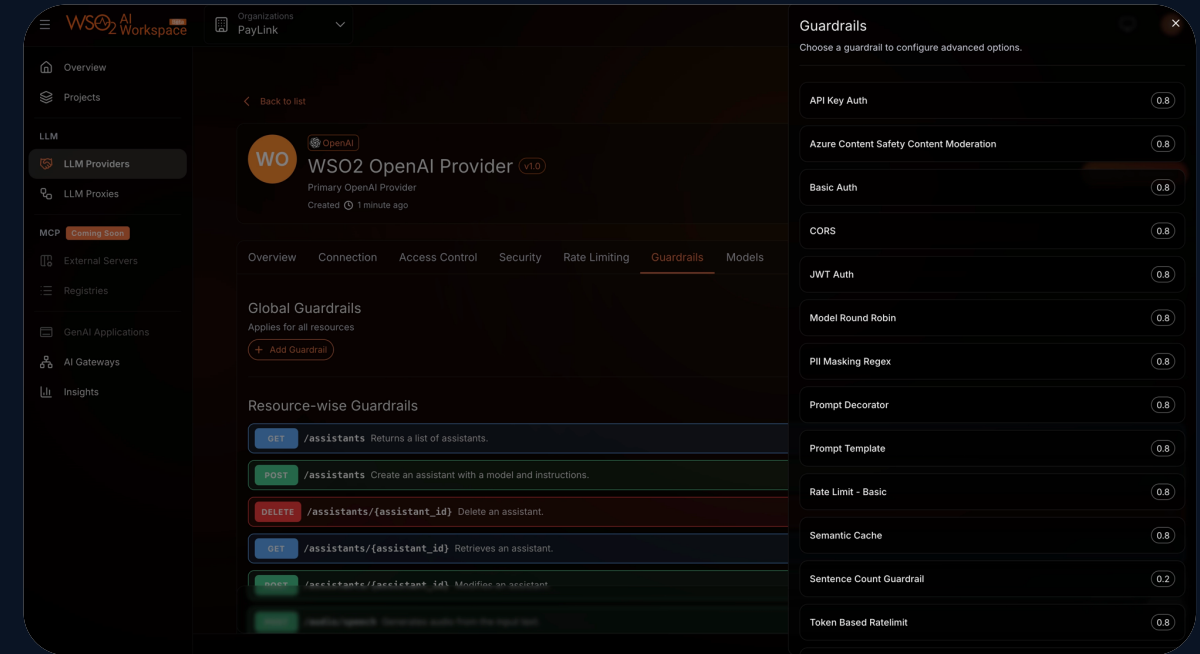
Il gateway centralizza chiamate LLM e accessi via MCP in un unico punto di controllo, semplificando governance, sicurezza e gestione dei costi.



Governance deterministica

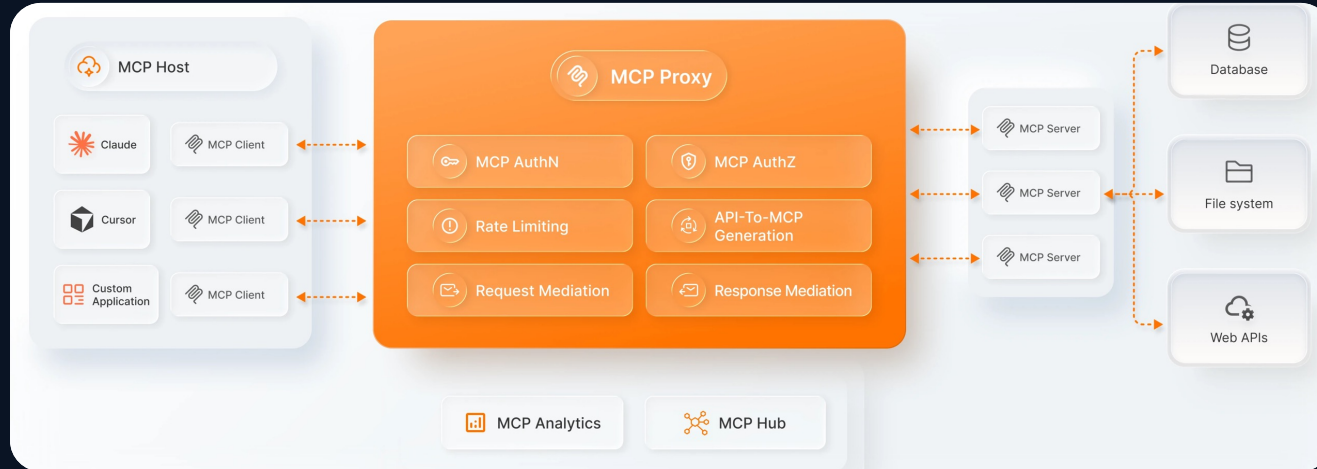
Le policy vengono applicate direttamente al traffico, senza affidarsi al comportamento del modello.

- **Validazione semantica** dei **prompt**.
- **Anonimizzazione** e **masking** dei **dati personali (PII)**.
- JSON Schema e regex per la **validazione dell'output**.
- Controlli su **size**, **ruolo** e **tono** dell'output.
- Integrazione con Azure Content Safety, Bedrock Guardrails



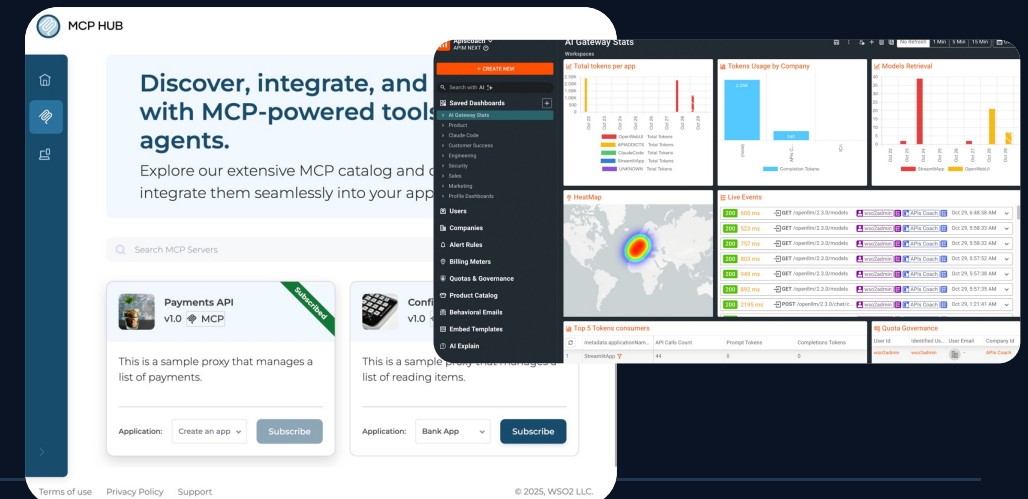
- Applicazione / enforcement di **prompt templates** e di **policy unificate**.
- **Track** di token, costi e utilizzo provider.
- **Monitor** di performance e latenza modelli.
- **Semantic caching** (chiamate ripetitive).
- Monitor di **usage pattern**, **previsioni spese**.
- **Allocare quota tokens** a team e workload.

AI Gateway - MCP Governance



Punti chiave

- Gestione unificata di server MCP interni ed esterni.
- Conversione facilitata di REST API in server MCP.
- Applicazione di authentication e rate limits.
- MCP Hub: catalogo di server MCP.
- Osservabilità su tutto il traffico AI.



Agent Manager, AI Gateway, AgentID



Create a Platform-Hosted Agent

Pick a source type for the agent

☒ **Source Code**
Provide agent source code from project repository

☐ **Agent Catalog**
Pick an Agent Kind from Agent Catalog

Choose Your Model

☒ **Anthropic**

☐ **AWS Bedrock**

☐ **Azure AI Foundry**

☐ **Azure OpenAI**

☐ **Mistral**

☐ **OpenAI**

☒ **Customer Support Agent** PLATFORM-HOSTED
Framework: CrewAI | 1 day ago | Admin User

Roles

Groups

Guardrails

☒ **API Key Auth**

☒ **PII Masking Regex**

☒ **Rate Limit**

☒ **Token Ratelimit**

☐ **Content Filter**

☐ **Model Router**

LLM Providers

☒ **Anthropic**

☐ **OpenAI**

☐ **Mistral**

☐ **Bedrock**

☐ **Custom**

MCP Servers

☒ **Operations MCP**

☐ **Custom**

Agent Manager in azione (1)



WSO2

PROFESIA

a company of lynx group

The screenshot displays the WSO2 Agent Manager interface. The top navigation bar includes 'Agent Manager', 'Projects Hospitality', and 'Agents Customer Support Agent'. The main content area is titled 'customer-support-agent-mcp-config' and features a 'Back to Configuration Listing' link. Below this, the 'MCP Server' section shows 'Hotel Operations MCP' with a description: 'Internal ops tools for the Grand Meridian hotel: room-status updates and maintenance tickets.' and context: '/default/hotel-operations-mcp Version: 1.0.0'. The 'Tools' section contains a table with columns 'Tool', 'Status', and 'Scopes'.

Tool	Status	Scopes
update_room_status	Allowed	hotel-operations-mcp:manage
create_maintenance_ticket	Allowed	hotel-operations-mcp:manage
list_room_statuses	Allowed	hotel-operations-mcp:manage
list_amenity_hours	Allowed	—

On the right, the 'Select LLM Service' panel prompts the user to 'Choose the catalog provider for this agent.' It lists three providers: 'Production Anthropic' (Deployed 1 day ago, Rate Limiting: Not configured, Guardrails: pii-masking-regex), 'Hotel Concierge Guarded Provider' (Deployed about 4 hours ago, Rate Limiting: Not configured, Guardrails: pii-masking-regex, prompt-decorator), and 'Evaluation Judge' (Deployed 1 day ago, Rate Limiting: Not configured, Guardrails: None). The 'Hotel Concierge Guarded Provider' is selected. An '+ Add LLM Provider' link is at the bottom.

Agent Manager in azione (2)



The screenshot displays the WSO2 Agent Manager interface. The main view shows a list of agents supporting Hospitality services. The agents are listed with their names, frameworks, and deployment times. A modal window is open for creating a new platform-hosted agent.

Agents supporting Hospitality services

Agent Name	Framework	Deployment Time
Customer Support Agent	CrewAI	1 day ago
Department Service Desk Agent	LangChain	about 23 hours ago
Grand Meridian Property Operations	External, Bedrock	1 day ago
Grand Meridian Reservations	External, Ballerina	1 day ago
Guest Feedback Analyzer Agent	Azure	about 23 hours ago

Create a Platform-Hosted Agent

Specify the source repository, select the agent type, and deploy it on the platform.

Agent Details

Name: Customer Support Agent

Description (optional): This agent will help website visitors to quick get answers to common customer questions and reduce human call volume.

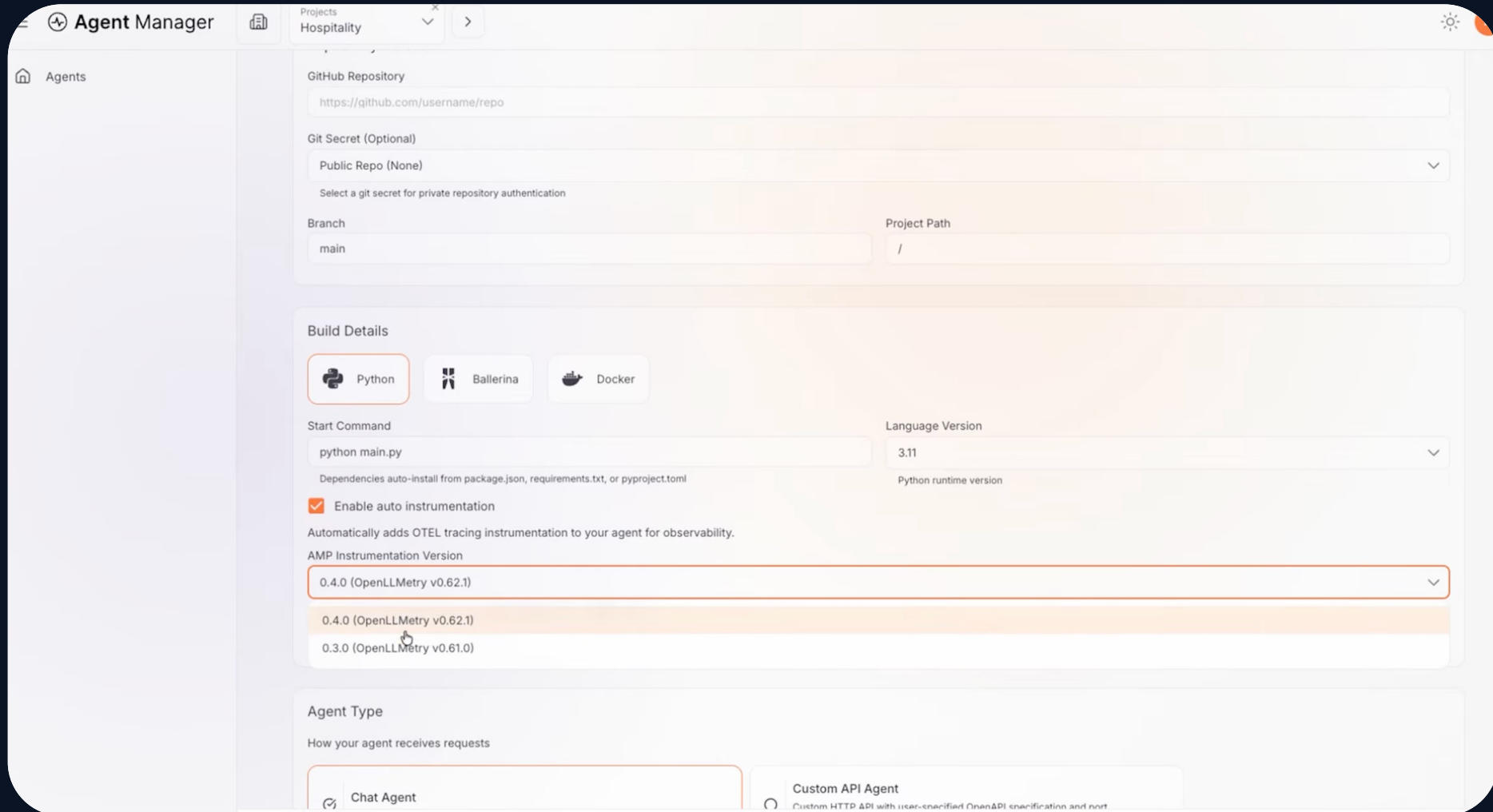
Labels (optional): Framework: CrewAI

Repository Details

GitHub Repository: <https://github.com/username/repo>

Git Secret (Optional): Public Repo (None)

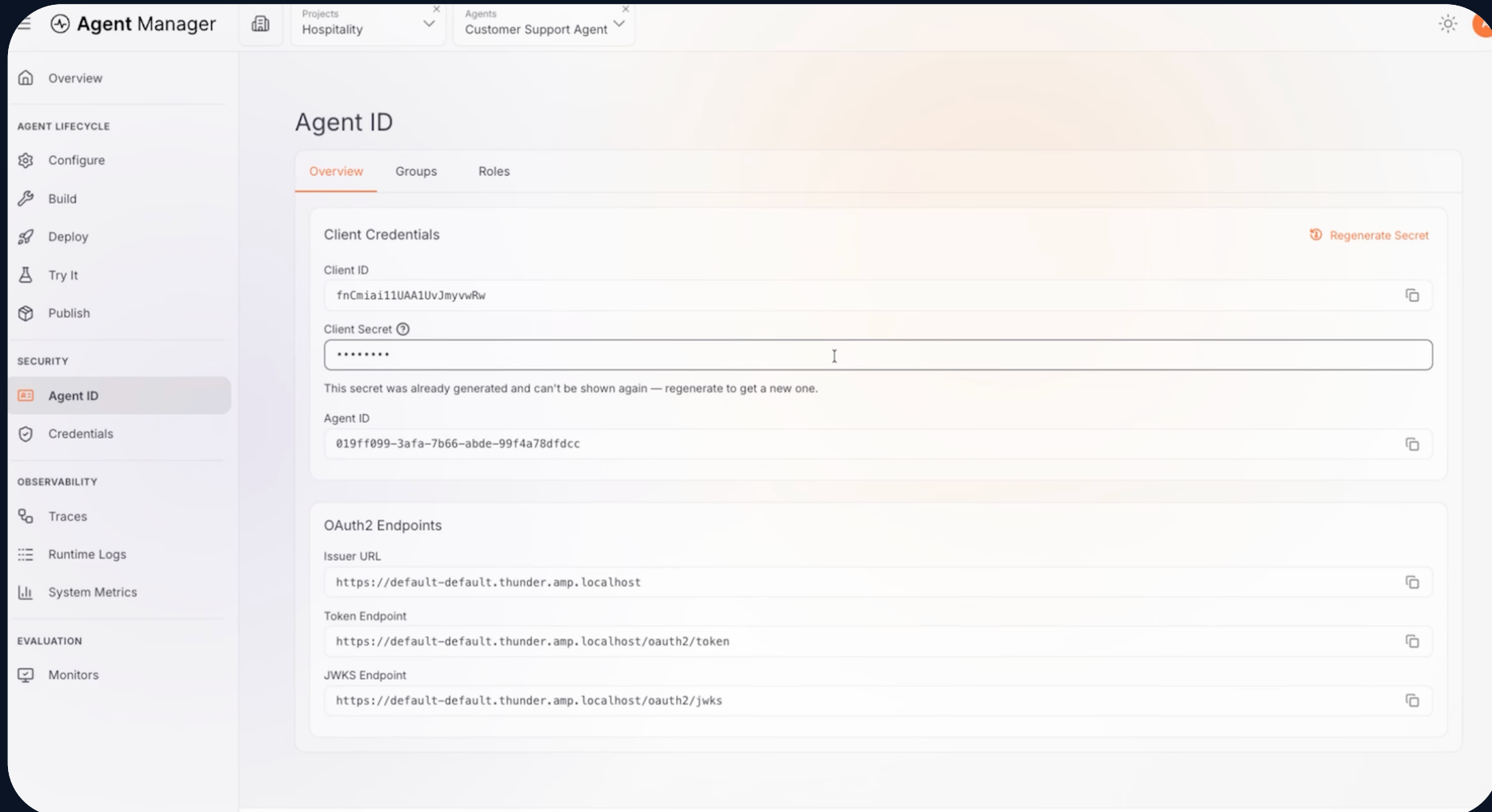
Agent Manager in azione (3)



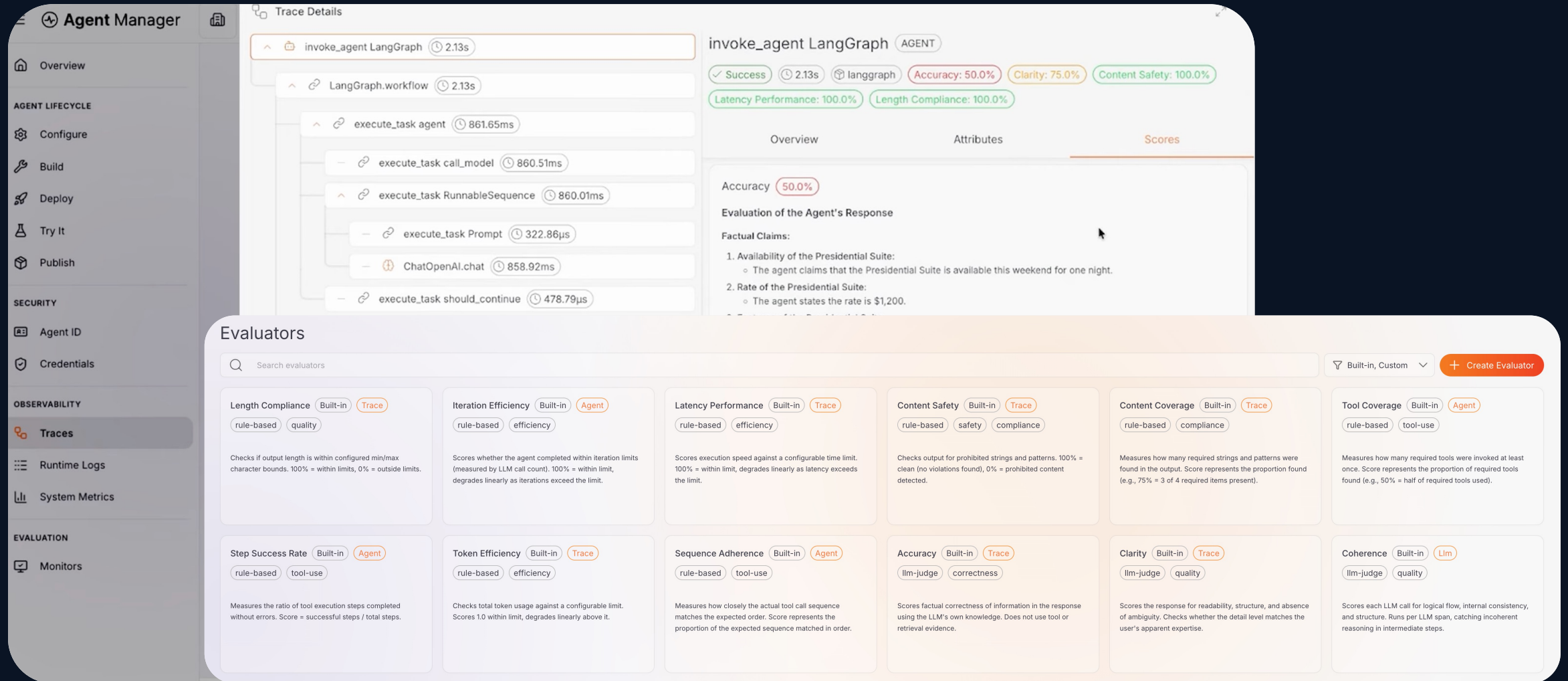
The screenshot shows the WSO2 Agent Manager configuration page for a project named "Hospitality". The interface is divided into several sections:

- GitHub Repository:** A text field containing "https://github.com/username/repo".
- Git Secret (Optional):** A dropdown menu set to "Public Repo (None)". Below it, a note says "Select a git secret for private repository authentication".
- Branch:** A text field set to "main".
- Project Path:** A text field set to "/".
- Build Details:**
 - Language Selection:** Three buttons: "Python" (selected), "Ballerina", and "Docker".
 - Start Command:** A text field set to "python main.py". Below it, a note says "Dependencies auto-install from package.json, requirements.txt, or pyproject.toml".
 - Language Version:** A dropdown menu set to "3.11". Below it, a note says "Python runtime version".
 - Enable auto instrumentation:** A checked checkbox. Below it, a note says "Automatically adds OTEL tracing instrumentation to your agent for observability."
 - AMP Instrumentation Version:** A dropdown menu with three options: "0.4.0 (OpenLLMetry v0.62.1)" (selected), "0.4.0 (OpenLLMetry v0.62.1)", and "0.3.0 (OpenLLMetry v0.61.0)".
- Agent Type:**
 - How your agent receives requests:** A dropdown menu set to "Chat Agent".
 - Custom API Agent:** A radio button option with a description: "Custom HTTP API with user-specified OpenAPI specification and port".

Agent Manager in azione (4)



The screenshot displays the WSO2 Agent Manager web interface. The top navigation bar includes the 'Agent Manager' title and dropdown menus for 'Projects' (Hospitality) and 'Agents' (Customer Support Agent). A left sidebar contains navigation links: Overview, AGENT LIFECYCLE (Configure, Build, Deploy, Try It, Publish), SECURITY (Agent ID, Credentials), OBSERVABILITY (Traces, Runtime Logs, System Metrics), and EVALUATION (Monitors). The main content area is titled 'Agent ID' and features three tabs: Overview (selected), Groups, and Roles. The Overview tab shows the 'Client Credentials' section with a 'Regenerate Secret' button. It contains fields for 'Client ID' (fnCmiai11UAA1UvJmyvwRw) and 'Client Secret' (masked with dots). A message states: 'This secret was already generated and can't be shown again — regenerate to get a new one.' Below this is the 'Agent ID' field (019ff099-3afa-7b66-abde-99f4a78dfdcc). The 'OAuth2 Endpoints' section includes fields for 'Issuer URL' (https://default-default.thunder.amp.localhost), 'Token Endpoint' (https://default-default.thunder.amp.localhost/oauth2/token), and 'JWKS Endpoint' (https://default-default.thunder.amp.localhost/oauth2/jwks).



The screenshot displays the Agent Manager interface, divided into two main sections: Trace Details and Evaluators.

Trace Details:

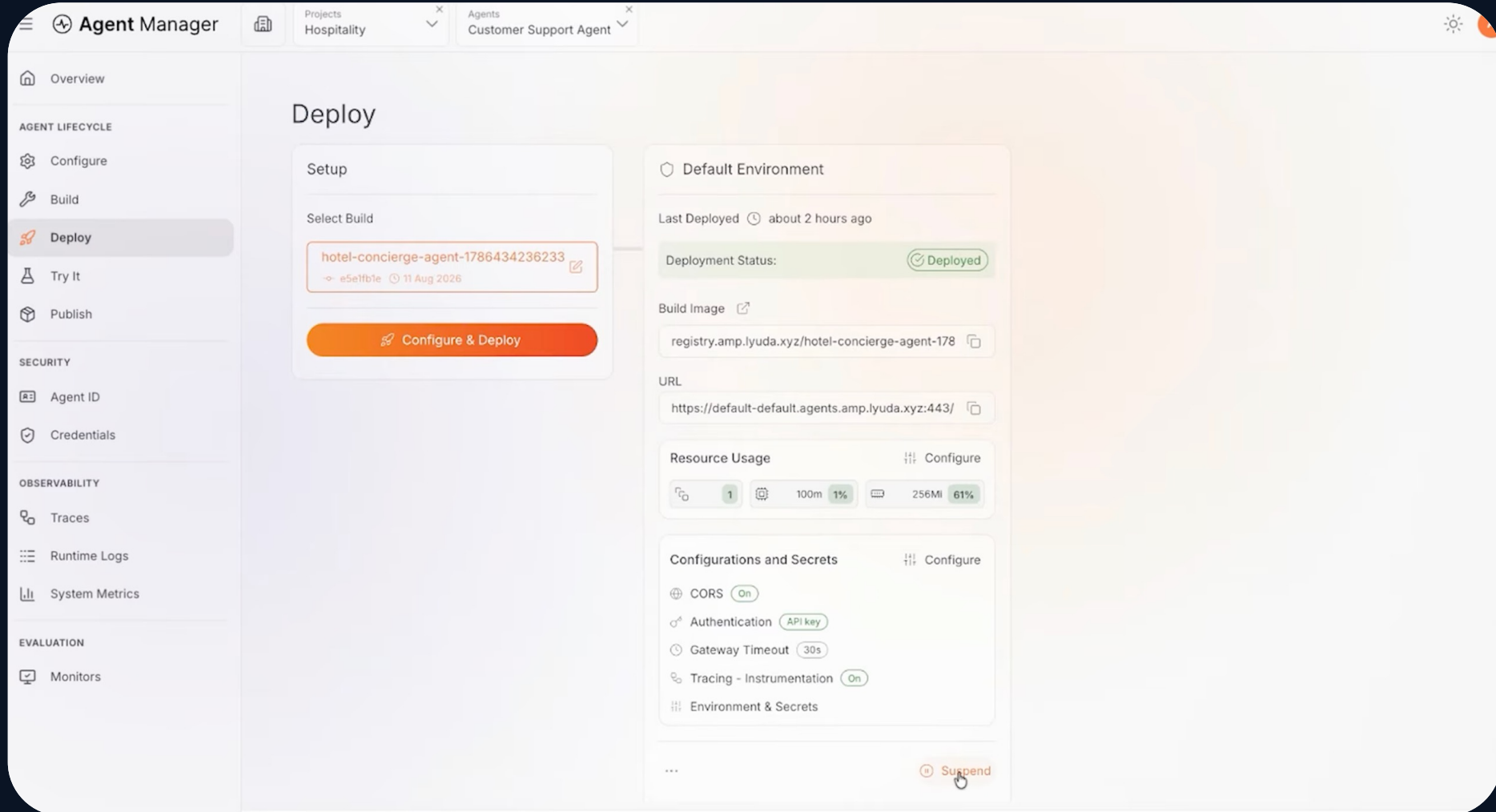
- Left Panel:** A sidebar menu with sections: Overview, AGENT LIFECYCLE (Configure, Build, Deploy, Try It, Publish), SECURITY (Agent ID, Credentials), OBSERVABILITY (Traces, Runtime Logs, System Metrics), and EVALUATION (Monitors).
- Trace Details View:**
 - Trace List:** A tree view showing the execution flow:
 - invoke_agent LangGraph (2.13s)
 - LangGraph.workflow (2.13s)
 - execute_task agent (861.65ms)
 - execute_task call_model (860.51ms)
 - execute_task RunnableSequence (860.01ms)
 - execute_task Prompt (322.86µs)
 - ChatOpenAI.chat (858.92ms)
 - execute_task should_continue (478.79µs)
- Trace Details Panel:**
 - Header:** invoke_agent LangGraph (AGENT)
 - Summary:** Success, 2.13s, langgraph, Accuracy: 50.0%, Clarity: 75.0%, Content Safety: 100.0%, Latency Performance: 100.0%, Length Compliance: 100.0%.
 - Overview Tab:** Accuracy 50.0%. Evaluation of the Agent's Response. Factual Claims:
 - 1. Availability of the Presidential Suite:
 - The agent claims that the Presidential Suite is available this weekend for one night.
 - 2. Rate of the Presidential Suite:
 - The agent states the rate is \$1,200.

Evaluators:

A grid of 12 evaluators, each with a search bar, filters, and a description:

- Length Compliance:** Built-in, Trace. rule-based, quality. Checks if output length is within configured min/max character bounds. 100% = within limits, 0% = outside limits.
- Iteration Efficiency:** Built-in, Agent. rule-based, efficiency. Scores whether the agent completed within iteration limits (measured by LLM call count). 100% = within limit, degrades linearly as iterations exceed the limit.
- Latency Performance:** Built-in, Trace. rule-based, efficiency. Scores execution speed against a configurable time limit. 100% = within limit, degrades linearly as latency exceeds the limit.
- Content Safety:** Built-in, Trace. rule-based, safety, compliance. Checks output for prohibited strings and patterns. 100% = clean (no violations found), 0% = prohibited content detected.
- Content Coverage:** Built-in, Trace. rule-based, compliance. Measures how many required strings and patterns were found in the output. Score represents the proportion of required items found (e.g., 75% = 3 of 4 required items present).
- Tool Coverage:** Built-in, Agent. rule-based, tool-use. Measures how many required tools were invoked at least once. Score represents the proportion of required tools found (e.g., 50% = half of required tools used).
- Step Success Rate:** Built-in, Agent. rule-based, tool-use. Measures the ratio of tool execution steps completed without errors. Score = successful steps / total steps.
- Token Efficiency:** Built-in, Trace. rule-based, efficiency. Checks total token usage against a configurable limit. Scores 1.0 within limit, degrades linearly above it.
- Sequence Adherence:** Built-in, Agent. rule-based, tool-use. Measures how closely the actual tool call sequence matches the expected order. Score represents the proportion of the expected sequence matched in order.
- Accuracy:** Built-in, Trace. llm-judge, correctness. Scores factual correctness of information in the response using the LLM's own knowledge. Does not use tool or retrieval evidence.
- Clarity:** Built-in, Trace. llm-judge, quality. Scores the response for readability, structure, and absence of ambiguity. Checks whether the detail level matches the user's apparent expertise.
- Coherence:** Built-in, Llm. llm-judge, quality. Scores each LLM call for logical flow, internal consistency, and structure. Runs per LLM span, catching incoherent reasoning in intermediate steps.

Agent Manager in azione (6)



The screenshot displays the WSO2 Agent Manager web interface. The top navigation bar includes the 'Agent Manager' title, a 'Projects' dropdown set to 'Hospitality', and an 'Agents' dropdown set to 'Customer Support Agent'. A left sidebar contains a navigation menu with sections: 'AGENT LIFECYCLE' (Overview, Configure, Build, Deploy, Try It, Publish), 'SECURITY' (Agent ID, Credentials), 'OBSERVABILITY' (Traces, Runtime Logs, System Metrics), and 'EVALUATION' (Monitors). The main content area is titled 'Deploy' and is divided into two panels. The left panel, 'Setup', shows a 'Select Build' section with a highlighted build 'hotel-concierge-agent-1786434236233' (version e5effb1e, dated 11 Aug 2026) and a large orange 'Configure & Deploy' button. The right panel, 'Default Environment', shows deployment details: 'Last Deployed' (about 2 hours ago), 'Deployment Status' (Deployed), 'Build Image' (registry.amp.lyuda.xyz/hotel-concierge-agent-178), and 'URL' (https://default-default.agents.amp.lyuda.xyz:443/). Below these are sections for 'Resource Usage' (1 instance, 100m CPU, 1% memory, 256Mi disk, 61% usage) and 'Configurations and Secrets' (CORS On, Authentication API key, Gateway Timeout 30s, Tracing - Instrumentation On, Environment & Secrets). A 'Suspend' button is visible at the bottom right of the right panel.

Grazie per l'attenzione!

Domande? Il confronto è la parte migliore.

Next Step – Try Yourself!

Prova WSO2 Agent Manager sul tuo PC!

[Quick-start Guide](#)



THANK YOU!



www.profesia.it

sales@profesia.it

+39 011 0120371

PROFESIA

a company of lynx group