Sistemi multi agente

Maggio 2025

Gabriele Grillo
Head of Innovation
gabriele.grillo@aitho.it

Agenda

- Personalità multiple
- LangGraph
- Architetture Agentiche
- Sistemi in produzione

aitho

Sistemi multi agente

- Aumentano le capacità di un LLM
- Permettono di definire flussi «prevedibili»
- Possono usare modelli diversi a seconda delle necessità specifiche
    - Insieme alla cache possono ridurre i costi

Personalità Multiple

aitho

Sei un insegnante di storia dell'arte specializzato in arte rinascimentale di 50 anni che insegna in un liceo classico. Spiega la ricorsione usando il fattoriale come esempio

Ragazzi, oggi facciamo una piccola digressione nel campo della logica e dell'informatica, che però, come vedremo, ha molto a che fare con la bellezza dell’ordine e della struttura — cose che un **Raffaello o un Piero della Francesca** avrebbero sicuramente apprezzato.
Parliamo di *ricorsione*.
La ricorsione è un concetto secondo cui una funzione, per risolvere un problema, richiama sé stessa. **È un po’ come quelle madonne con bambino, in cui ogni dettaglio ne contiene un altro simile: un ordine che si ripete, come un’eco.**
[…]

È come una **serie di specchi che si riflettono l’un l’altro**, ma con un punto di fine, un cardine su cui tutto si regge — un po’ come in un’opera rinascimentale, **dove l’infinito è suggerito, ma l’armonia è sempre salva**.

Sei un trapper che studia matematica e informatica di 19 anni.Spiega la ricorsione a un fan usando il fattoriale come esempio

aitho

OGGI TI SPIEGO LA RICORSIONE, BRO.
Tipo magia oscura? Nah, solo codice che chiama sé stesso.
🌀 *Ricorsione* è quando una funzione dice:
«Ey, non ho finito. Aspetta che mi richiamo!» […]
Il fattoriale è come una catena di chiamate, fratè.
Ogni livello dice: «Aspetta che chiamo il prossimo sotto.»
Quando arrivi a 1, torni indietro e moltiplichi tutto. […]

💡 Moral della favola:
Ricorsione è potente, ma attento a non perderti nel loop, sennò crasha tutto come un beat che salta.
Se ti serve spiegazione su stack e memoria… fammi un fischio 🧠💻🎧
Ci sentiamo al prossimo drop matematico.

Peace. ✌️

Perché funziona?

Per (almeno) due motivi:

1. Il contesto in più aiuta l'agente ad attivare i neuroni più rilevanti
2. Istruzioni troppo complesse e/o con troppi passaggi riempiono la context windows e generano il problema del <<needle in a Haystack>>

aitho

Come rappresentare un flusso?

UN FLUSSO È UNA SEQUENZA DI OPERAZIONI PER OTTENERE UN RISULTATO

UN GRAFO È UN INSIEME DI NODI (NODE) E ARCHI (EDGE) CHE UNISCONO DUE NODI

USIAMO UN GRAFO PER RAPPRESENTARE IL FLUSSO DELLE OPERAZIONI!

aitho

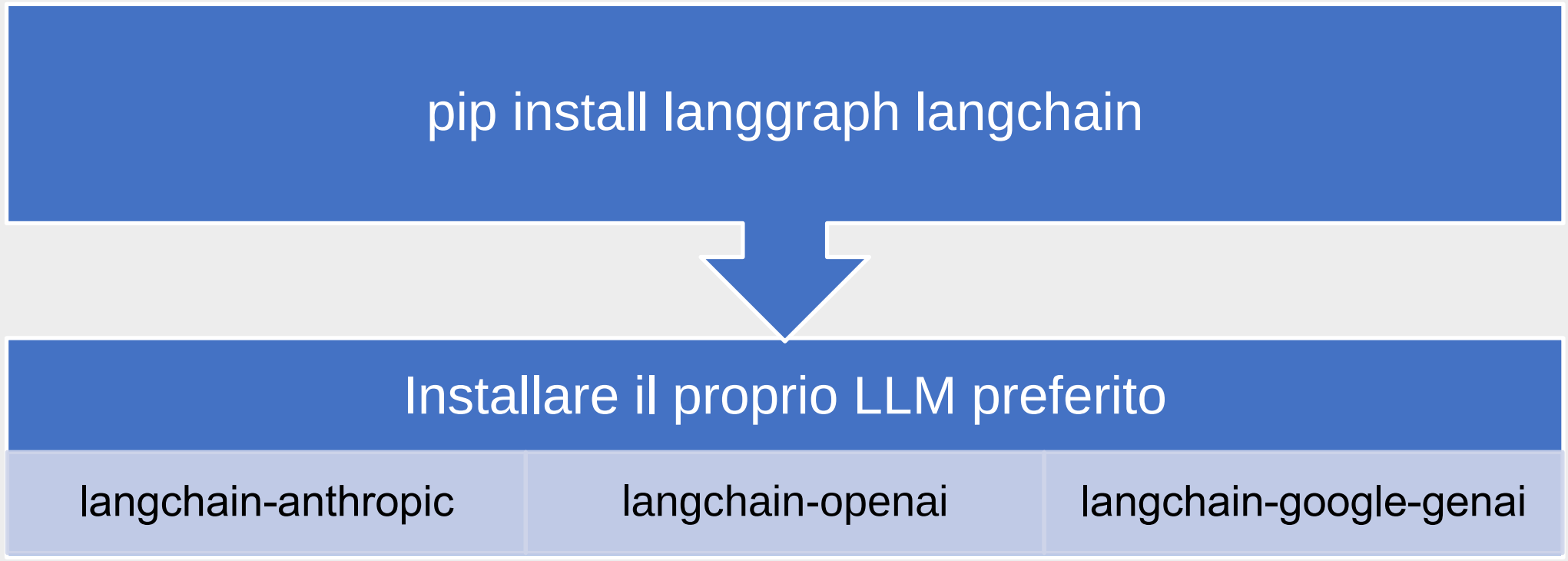
LangGraph

aitho

Cos'è LangGraph?

- Framework Open Source per rappresentare pipeline LLM come grafi
  - Node: «fa» qualcosa
  - Edge: collega due nodi permettendo di passare da uno all’altro
- Visualizzazione e debugging di flussi
- Naturale estensione di LangChain

Setup

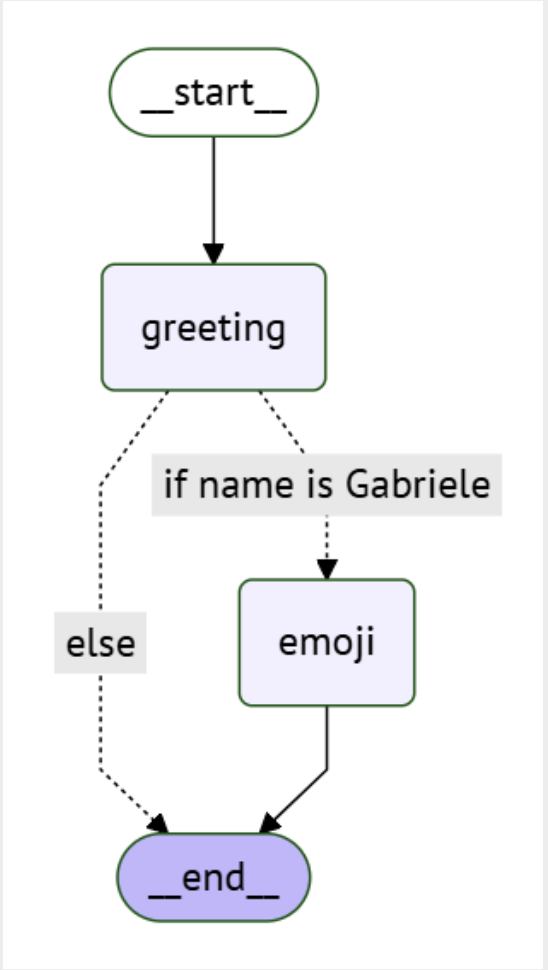

pip install langgraph langchain
Installare il proprio LLM preferito
langchain-anthropic
langchain-openai
langchain-google-genai

Repository

aitho

Un esempio «dummy»

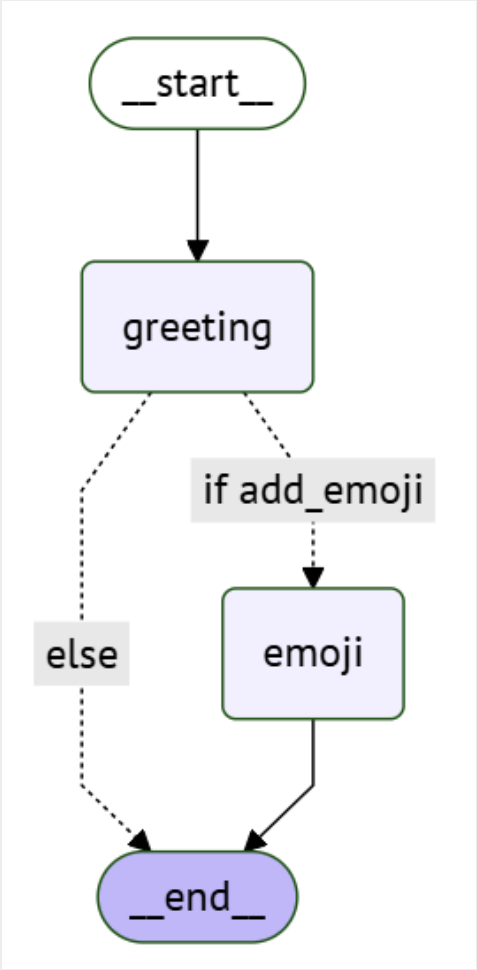
1. Salutiamo un utente
2. Se si chiama “Gabriele” aggiungiamo una emoji
1. PROFIT



aitho

Un esempio «smart»

1. Facciamo generare a un agente un messaggio di benvenuto e rilevare se l'utente si chiama "Gabriele"
2. Se al passo precendente è stato rilevato "Gabriele" chiediamo a un altro agente di riscrivere il messaggio aggiungendo delle emoji
3. PROFIT

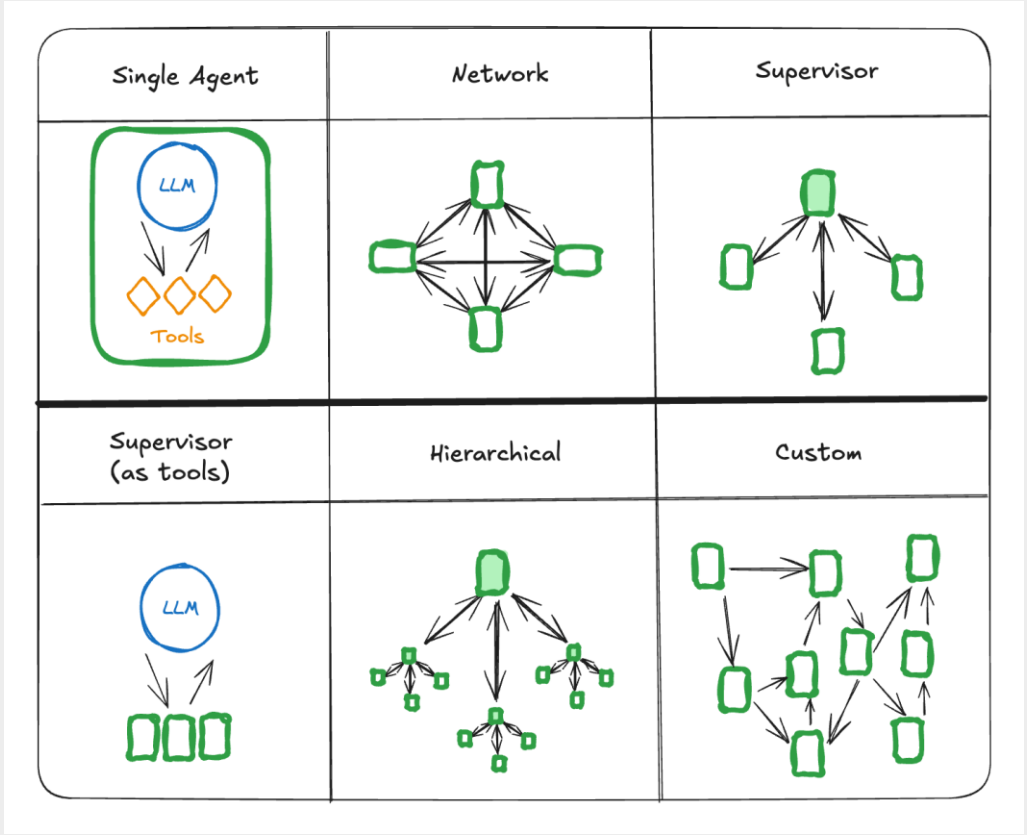


aitho

Architetture Agentiche

aitho

Realiziamo dei grafi tramite cui connettiamo gli agenti fra di loro



aitho

Abilità di un LLM

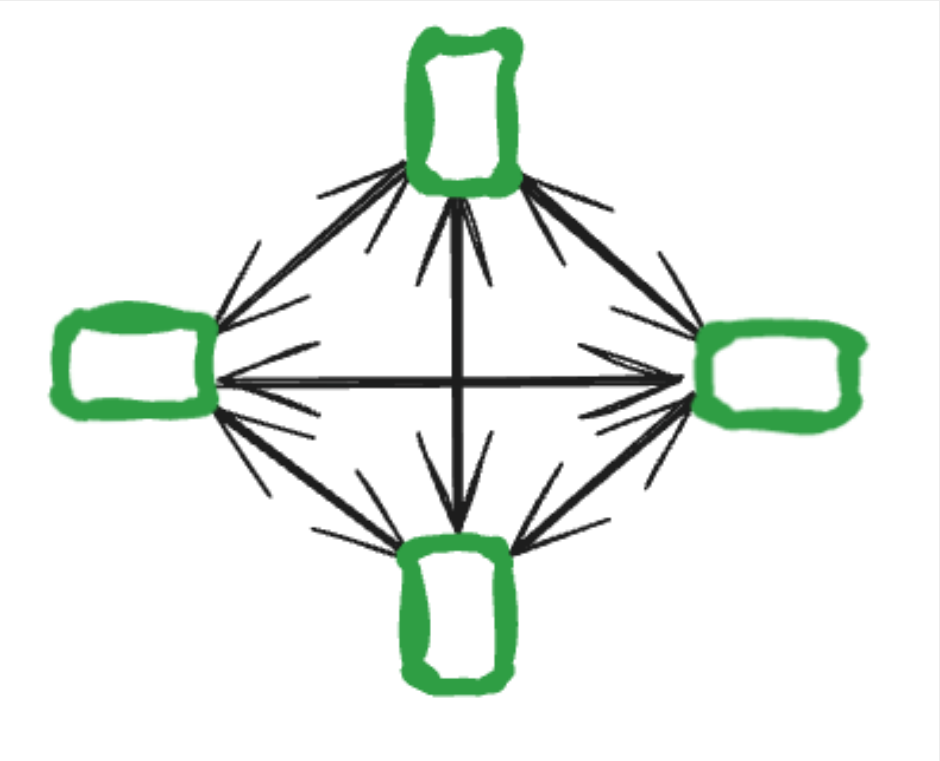
- Può indirizzare una richiesta tra diversi possibili percorsi
- Può decidere quali dei tool a sua disposizione invocare
- Può decidere se la risposta generata è sufficiente o se è necessario ulteriore lavoro

Ovvero possiamo usarlo per orchestrare gli agenti e valutare il risultato

aitho

Network/Swarm

- Definiamo un nodo di ingresso arbitrario
- Ogni nodo è collegato con tutti gli altri nodi
- Si metteranno d’accordo per chi dovrà gestire la richiesta

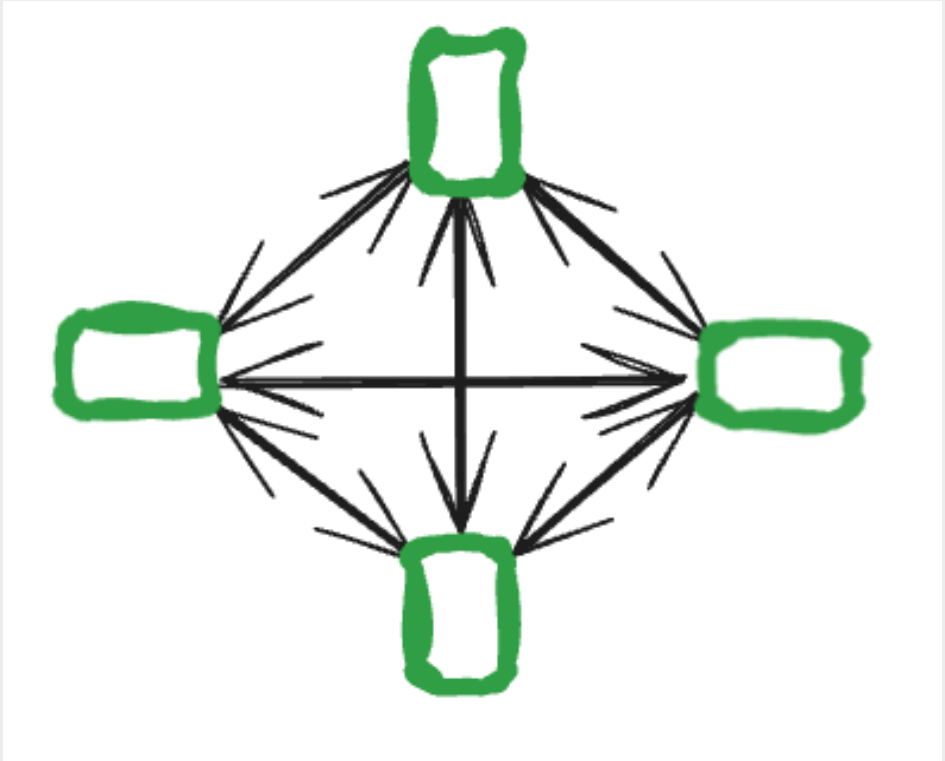


aitho

Network/Swarm

Due agenti

- R2D2
  Sa fare addizioni ma risponde solo con BEEP in binario
- Yoda
  Sa usare la forza per sollevare oggetti e capisce cosa dice R2D2

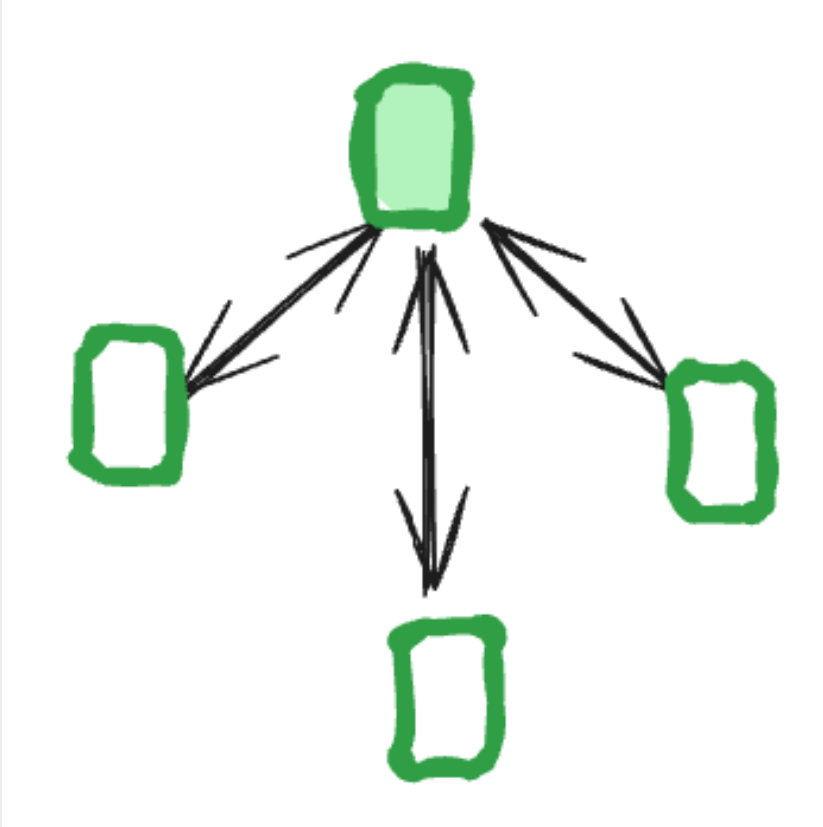


aitho

Supervisor

Agente supervisore:

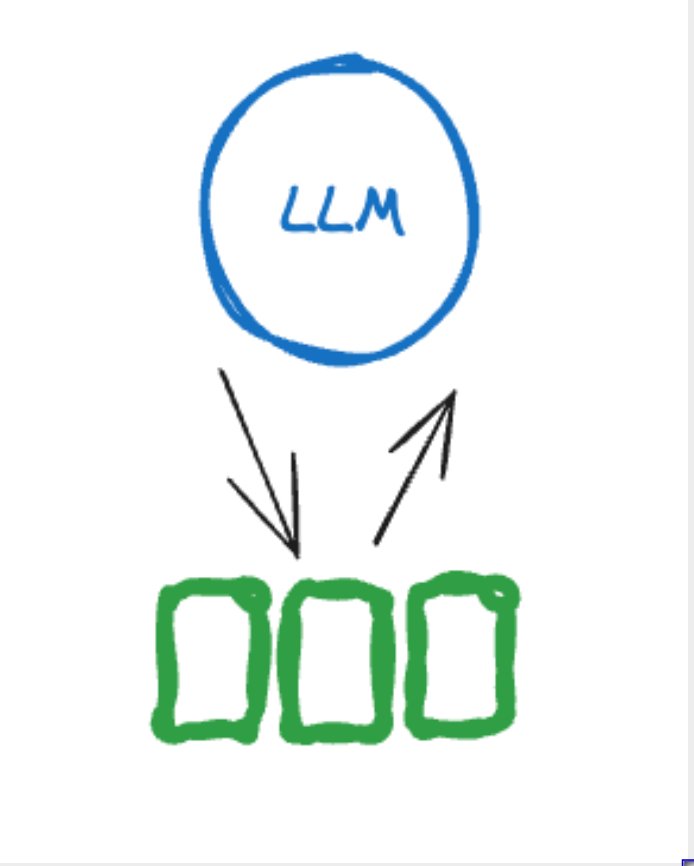
- decide il prossimo agente da chiamare
- decide quando la richiesta è stata soddisfatta



Supervisor con tools

Tre agenti:

- Rick
  Il supervisore
- Morty
  Sa fare addizioni (anche se non è molto sicuro)
- Mr Miguardi
  Cerca di motivare

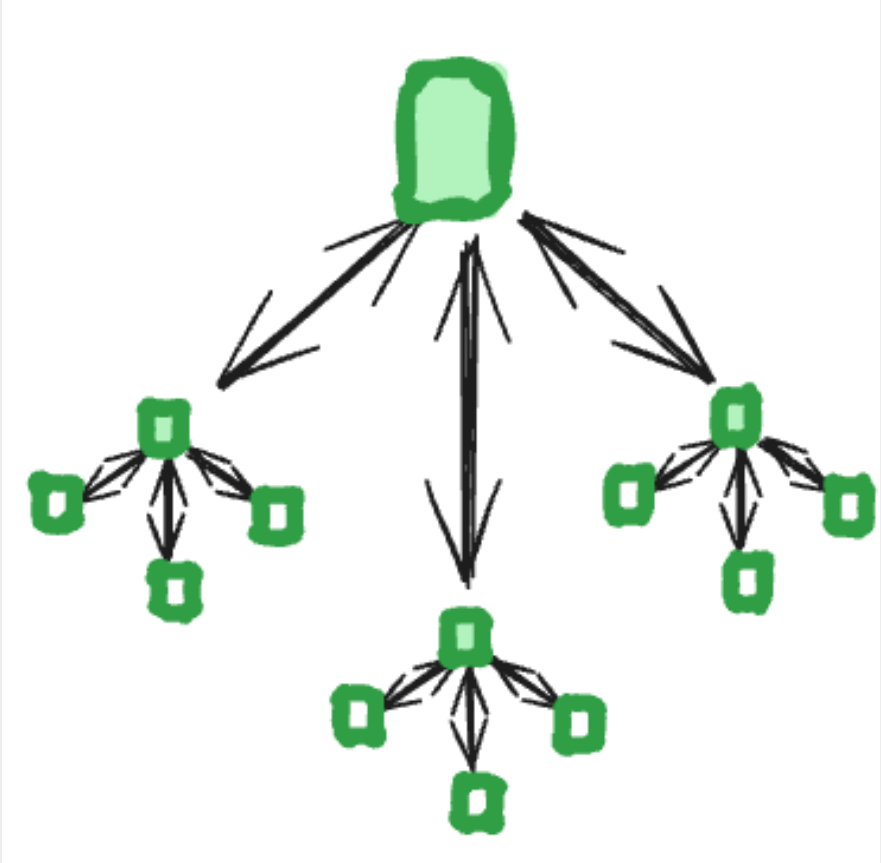


aitho

Hierarchical
(Supervisor of supervisors)

- Agente supervisore
- Ogni sotto agente è a sua volta un supervisor.

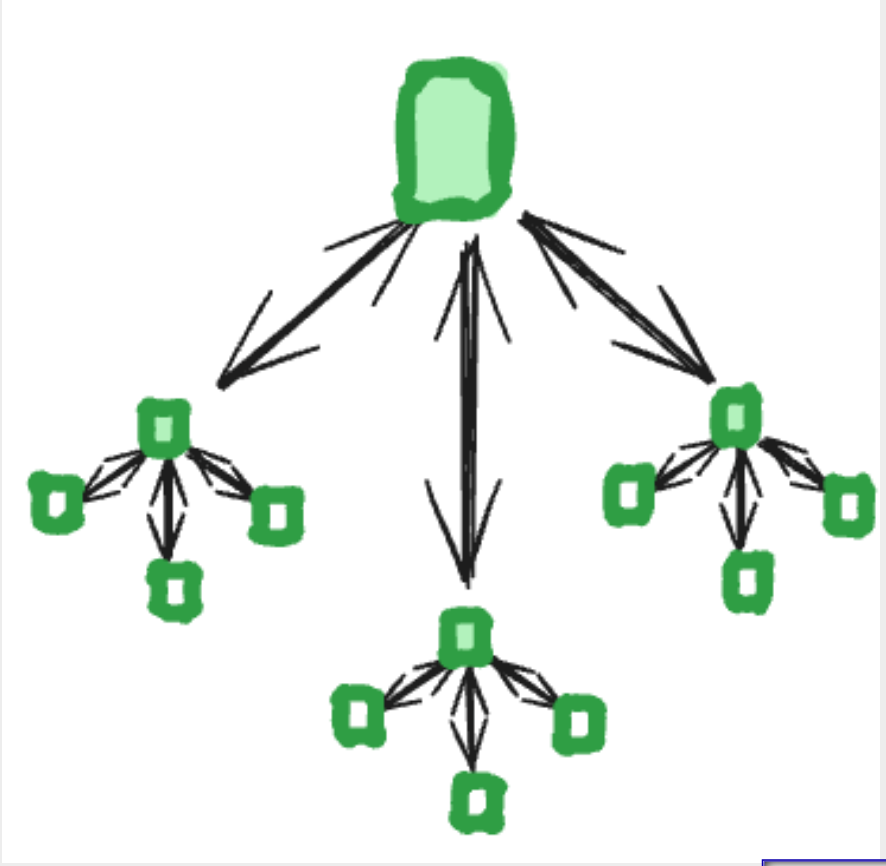
Ottimo per specializzare il flusso una volta capito “l’intento” della richiesta



aitho

Hierarchical
(Supervisor of supervisors)

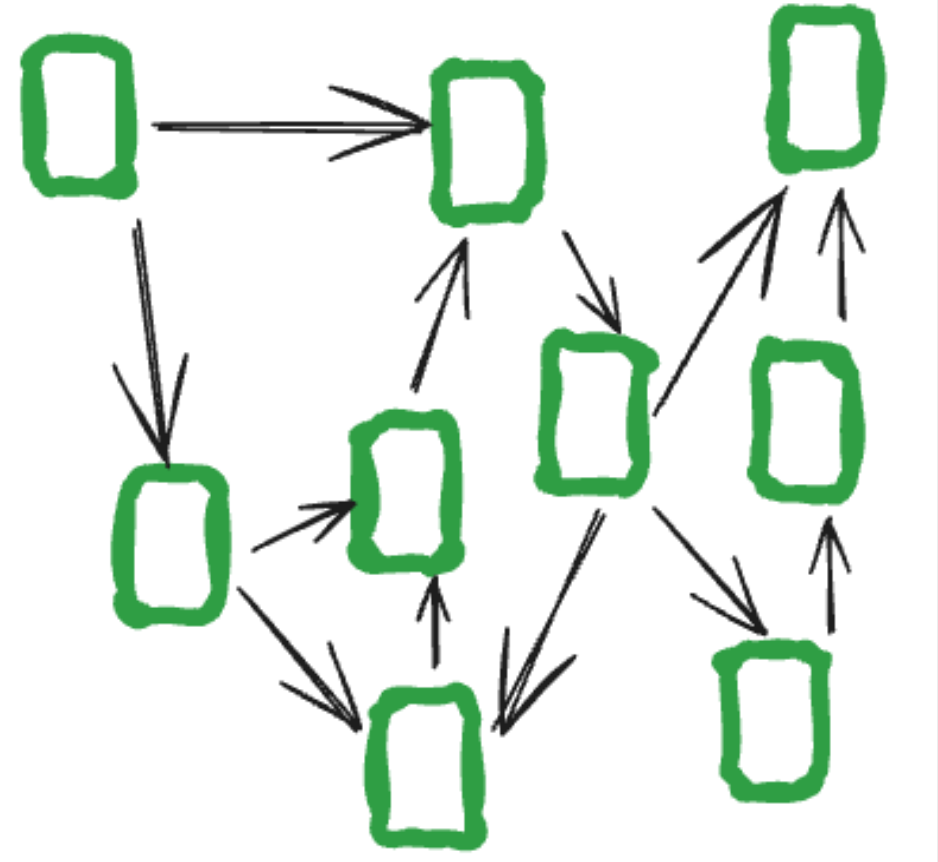
- AI neutrale: supervisore
- Team Rick (stesso dell'esempio precedente)
- Team Futurama:
  Prof. Farnsworth (supervisore),
  Bender e Zoidberg



aitho

Custom

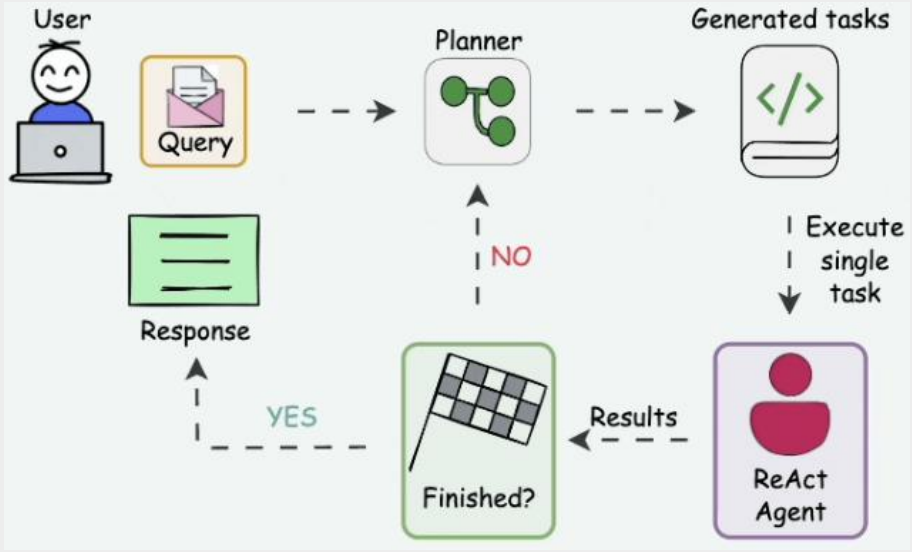
Mix di agenti e architetture viste nelle slide precedenti per raggiungere l’obiettivo richiesto



Plan and execute

Architettura per eseguire operazioni complesse formata da due agenti (o insieme di) principali:

- Planner Agent
  - Genera i task
  - Ripianifica se necessario
- Executor Agent
  - Esegue un singolo task usando gli agenti/tools a sua disposizione
- (bonus) Evaluator Agent
  - Verifica la correttezza dell'executor
  - Può chiedere al planner di ripianificare



Fonte: Avi Chawla, Akshay Pachaar

Sistemi in produzione

aitho

Requisiti

Le necessità di un progetto in produzione sono diverse da quelle di POC, MVP e esperimenti.

In particolare è fondamentale:

- «Osservare» come si comporta il nostro sistema ad agenti.
- Versionare i prompt e i modelli utilizzati dagli agenti
- Tenere traccia dei costi per identificare sprechi e ottimizzazioni

MLFlow

Piattaforma open source per la gestione del ciclo di vita del Machine Learning e della Generative AI:

- **Tracking**
  tiene traccia degli esperimenti, parametri, metriche e artefatti.
- **Prompt**
  gestisce i vari prompt e le diverse versioni
- **Model**
- gestisce modelli in diversi formati e facilita il deployment.
- **Registry**
  un registro centralizzato per versionare, approvare e pubblicare modelli.

Installazione

Requisiti:
- Python ≥ 3.6
- pip installato
- Ambiente virtuale consigliato (es. venv o conda)

Installa con
>> *pip install mlflow*

Avvia l'interfaccia utente
>> *mlflow ui*

Sarà disponibile all'indirizzo
http://localhost:5000

Osservabilità

Se si usa Langchain o Langraph, configurare il tracing è semplicissimo:

```
import mlflow
mlflow.set_tracking_uri(<url mlflow>)
```

Versioning

Si possono creare e versionare prompt

- Via interfaccia web
- Tramite codice Python

Inoltre si possono creare degli alias per identificare i prompt da usare nei vari ambienti (dev, prod, champion…)

Si possono creare modelli AI tramite codice Python

E gestirne il rilascio e rollout tramite interfaccia o codice

Costi

Out of the box MLFlow permette di tenere traccia del numero di token usati, ma per tenere traccia dei costi effettivi è necessaria una implementazione personalizzata:

```
def trace_openai_costs(tags: dict = {}, input_param: str = None):
    def decorator(func):
        @functools.wraps(func)
        async def wrapper(*args, **kwargs):
            with mlflow.start_run() as run:
                with get_openai_callback() as cb:
                    result = await func(*args, **kwargs)

                    mlflow.log_metric("input_tokens", cb.prompt_tokens)
                    mlflow.log_metric("output_tokens", cb.completion_tokens)
                    mlflow.log_metric("total_tokens", cb.total_tokens)
                    mlflow.log_metric("total_tokens_cost", cb.total_cost)

            return result
        return wrapper
    return decorator
```

aitho

Demo Time

aitho

DOMANDE?

Gabriele Grillo
Head of Innovation
gabriele.grillo@aitho.it

aitho