Profice

AI ACADEMY

Applicare l’Intelligenza Artificiale nello sviluppo software

AI ACADEMY

LLM via API (multi-vendor)
20/06/2025

INTRODUZIONE DELL'ISTRUTTORE

Tamas Szakacs

Formazione

- Laureato come programmatore matematico
- MBA in management

Principali esperienze di lavoro

- Amministratore di sistemi UNIX
- Oracle DBA
- Sviluppatore di Java, Python e di Oracle PL/SQL
- Architetto (solution, enterprise, security, data)
- Ricercatore tecnologico e interdisciplinare di IA

Dedicato alla formazione continua

- Teorie, modelli, framework IA
- Ricerche IA
- Strategie aziendali
- Trasformazione digitale
- Formazione professionale

email: tamas.szakacs@proficegroup.it

MOTIVI E RIASSUNTO DEL CORSO

L'**Intelligenza Artificiale (AI)** è oggi il motore dell'innovazione in ogni settore, grazie alla sua capacità di analizzare dati, automatizzare processi e generare nuove soluzioni. Questo corso offre una panoramica completa e pratica sullo sviluppo di applicazioni AI moderne, guidando i partecipanti dall'ideazione al rilascio in produzione.

Attraverso una **combinazione di teoria chiara ed esercitazioni pratiche**, saranno affrontate le tecniche e gli strumenti più attuali: **machine learning, deep learning, reti neurali, Large Language Models (LLM), Transformers, Retrieval Augmented Generation (RAG)** e progettazione di agenti AI.
Le competenze acquisite saranno applicate in progetti concreti, dallo sviluppo di chatbot all'integrazione di modelli generativi, fino al deploy di soluzioni AI in ambienti reali e collaborativi.

Il percorso è pensato per chi vuole imparare a progettare, valutare e integrare sistemi AI di nuova generazione, con particolare attenzione alle best practice di programmazione, collaborazione in team, sicurezza, valutazione delle performance ed etica dell'AI.

DURATA: 17 GIORNI

OBIETTIVI

Il percorso formativo è progettato per **giovani consulenti junior**, con una conoscenza base di programmazione, che stanno iniziando un percorso professionale nel settore AI.

L’obiettivo centrale è fornire una panoramica pratica, completa e operativa sull’intelligenza artificiale moderna, guidando ogni partecipante attraverso tutte le fasi fondamentali.

OBIETTIVI

- Allineare conoscenze AI, ML, DL di tutti i partecipanti
- Saper usare e orchestrare modelli LLM (closed e open-weight)
- Costruire pipeline RAG complete (retrieval-augmented generation)
- Progettare agenti AI semplici con strumenti moderni (LangChain, tool calling)
- Capire principi di valutazione, robustezza e sicurezza dei sistemi GenA
- Migliorare la produttività come sviluppatori usando tool GenAI-driven
- Padroneggiare best practice di sviluppo, versioning e deploy AI
- Introdurre i fondamenti di Graph Data Science e Knowledge Graph
- Ottenere capacità di valutazione dei modelli e metriche
- Comprensione dell'etica e dei bias nei modelli di intelligenza artificiale
- Approfondire le normative di riferimento: AI Act, compliance e governance AI

Il corso è **estremamente pratico** (circa il 40% del tempo in esercitazioni hands-on, notebook, challenge e hackathon), con l'utilizzo di Google Colab, GitHub, e tutti gli strumenti necessari per lavorare su progetti reali e simulati.

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
1	Git & Python clean-code	Collaborazione su progetti reali, versionamento, codice pulito e testato
2	Machine Learning Supervised	Modelli supervisionati per predizione e classificazione
3	Machine Learning Unsupervised	Clustering, riduzione dimensionale, scoperta di pattern
4	Prompt Engineering avanzato	Scrivere e valutare prompt efficaci per modelli generativi
5	LLM via API (multi-vendor)	Uso pratico di modelli LLM via API, autenticazione, deployment
6	Come costruire un RAG	Pipeline end-to-end per Retrieval-Augmented Generation
7	Tool-calling & Agent design	Progettare agenti AI che usano strumenti esterni
8	Hackathon: Agentic RAG	Challenge pratica: chatbot agentico RAG in team

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
9	Hackathon: Rapid Prototyping	Da prototipo a web-app con Streamlit e GitHub
10	AI Productivity Tools	Workflow con IDE AI-powered, automazione e refactoring assistito
11	Docker & HF Spaces Deploy	Deployment di app GenAI containerizzate o su HuggingFace Spaces
12	AI Act & ISO 42001 Compliance	Fondamenti di compliance e governance AI
13	Knowledge Base & Graph Data Science	Introduzione a Knowledge Graph e query con Neo4j
14	Model evaluation & osservabilità	Metriche avanzate, explainability, strumenti di valutazione
15	AI bias, fairness ed etica applicata	Analisi dei rischi, metriche e mitigazione dei bias
16-17	Project Work & Challenge finale	Lavoro a gruppi, POC/POD, presentazione e votazione progetti

METODOLOGIA DEL CORSO

1. Approccio introduttivo ma avanzato

Il corso è introduttivo nei concetti base dell'AI applicata allo sviluppo, ma affronta anche tecnologie, modelli e soluzioni avanzate per garantire un apprendimento completo.

2. Linguaggio adattato

Il linguaggio utilizzato è chiaro e adattato agli studenti, con spiegazioni dettagliate dei termini tecnici per favorirne la comprensione e l'apprendimento graduale.

3. Esercizi pratici

Gli esercizi pratici sono interamente svolti online tramite piattaforme come Google Colab o notebook Python, eliminando la necessità di installare software sul proprio computer.

4. Supporto interattivo

È possibile porre domande in qualsiasi momento durante le lezioni o successivamente via email per garantire una piena comprensione del materiale trattato.

NOTA

Il corso segue un **approccio laboratoriale**: ogni giornata combina sessioni teoriche chiare e concrete con molte attività pratiche supervisionate, per sviluppare *competenze reali* immediatamente applicabili.

I partecipanti lavoreranno spesso in gruppo, useranno notebook in Colab e versioneranno codice su GitHub, vivendo una vera simulazione del lavoro in azienda AI.

Nessun prerequisito avanzato richiesto: si partirà dagli strumenti e flussi fondamentali, con una crescita graduale verso le tecniche più attuali e richieste dal mercato.

ORARIO TIPICO DELLE GIORNATE

Orario	Attività	Dettaglio
09:00 – 09:30	Teoria introduttiva	Concetti chiave, schema della giornata
09:30 – 10:30	Live coding + esercizio guidato	Esempio pratico, notebook Colab
10:30 – 10:45	*Pausa breve*	
10:45 – 11:30	Approfondimento teorico	Tecniche, best practice
11:30 – 12:30	Esercizio hands-on individuale	Sviluppo o completamento di codice
12:30 – 13:00	Discussione soluzioni + Q&A	Condivisione e correzione
13:00 – 14:00	*Pausa pranzo*	
13:30 – 14:15	Teoria avanzata / nuovi tools	Nuovi strumenti, pattern, demo
14:15 – 15:30	Esercizio a gruppi / challenge	Lavoro di squadra su task reale
15:30 – 15:45	*Pausa breve*	
15:45 – 16:30	Sommario teorico e pratico	
16:30 – 17:00	Discussioni, feedback	Riepilogo, best practice, domande aperte

DOMANDE?

Cominciamo!

OBIETTIVI DELLA GIORNATA

Obiettivi della giornata

- Presentazione dei modelli LLM.
- Capire come utilizzare modelli LLM (GPT, Llama, ecc.) tramite API, conoscendo le differenze tra fornitori (Azure, Hugging Face, ecc.).
- Saper configurare endpoint API per task diversi (chat, completions, embeddings), comprendendo parametri, limiti e flussi di autenticazione.
- Gestire chiavi, credenziali, tier di servizio e regioni, imparando a usare rate-limit, retry e streaming dove necessario.
- Impostare e testare chiamate batch e in streaming per casi d'uso reali.
- Analizzare e confrontare la facilità d'uso, i costi e le performance tra fornitori API principali (Azure OpenAI, Hugging Face).
- Costruire codici pratici in cui chiamare API di LLM per generazione testo, Q&A, embeddings e altri task, usando chiavi live in sicurezza.
- Prepararsi a implementare LLM nei flussi aziendali: conoscenza di limiti di sicurezza, privacy e best practice nell'uso di API.
- Saper diagnosticare e risolvere errori comuni nelle chiamate API.

INTRODUZIONE A NATURAL LANGUAGE PROCESSING

Introduzione a Natural Language Processing (NLP)

Cosa fa l'NLP?

Il Natural Language Processing (NLP) si occupa di comprendere, generare e manipolare il linguaggio naturale. È applicato in traduzione automatica, analisi del sentiment, estrazione di informazioni, chatbot e molto altro. Utilizza tecniche di machine learning e deep learning per analizzare testo e parlato.

Come arriviamo all'NLP?

L'NLP si sviluppa a partire dai concetti fondamentali del deep learning, come reti convoluzionali (CNN) per analisi dei pattern e reti ricorrenti (RNN, LSTM) per dati sequenziali. Gli approcci moderni includono trasformazioni avanzate, come embedding per rappresentazioni numeriche del linguaggio, e l'uso dell'attenzione per identificare relazioni chiave nei dati.

Verso i Large Language Models (LLM)

I LLM, come ad es. Llama, BERT e GPT, sono basati su architetture avanzate come i Transformers. Essi utilizzano miliardi di parametri per elaborare grandi quantità di dati e risolvere problemi complessi di linguaggio. Il futuro degli LLM include modelli più efficienti, personalizzati e capaci di solvere compiti complessi in tempo reale, come agenti.

INTRODUZIONE A NATURAL LANGUAGE PROCESSING

Utilizzi tipici del Natural Language Processing (NLP)

Classificazione del Testo

- Categorizzazione di documenti o messaggi in base a temi o argomenti.
- Esempio: Filtrare email come spam o non spam.

Riconoscimento di Entità Nominate (NER)

- Estrazione di entità significative come nomi, luoghi, date o organizzazioni da un testo.
- Esempio: Estrarre "Profice" come azienda e "Mantova" come luogo in un articolo.

Ricerca di Informazioni (Information Retrieval)

- Recupero di informazioni pertinenti da un database o da documenti testuali.
- Esempio: Trovare elementi specifici in un documento.

Correzione Grammaticale e Ortografica

- Identificazione e correzione di errori nel testo.
- Esempio: Strumenti come Grammarly.

INTRODUZIONE A NATURAL LANGUAGE PROCESSING

Utilizzi tipici del Natural Language Processing (NLP)

Analisi del Sentiment

- Identificazione del tono emotivo in testi come recensioni, post sui social media o feedback dei clienti.
- Esempio: Determinare se un commento è positivo, negativo o neutro.

Sommario Automatico

- Creazione di versioni brevi e concise di documenti più lunghi.
- Esempio: Generare un riassunto di un articolo di notizie.

Traduzione Automatica

- Conversione di testo da una lingua all'altra utilizzando modelli NLP.
- Esempio: Tradurre dall'italiano all'inglese.

Generazione del Linguaggio Naturale (NLG)

- Creazione automatica di testo comprensibile e coerente per chatbot, assistenti vocali e sistemi di risposta automatica.
- Esempio: Rispondere a una domanda in un chatbot.

INTRODUZIONE A NATURAL LANGUAGE PROCESSING

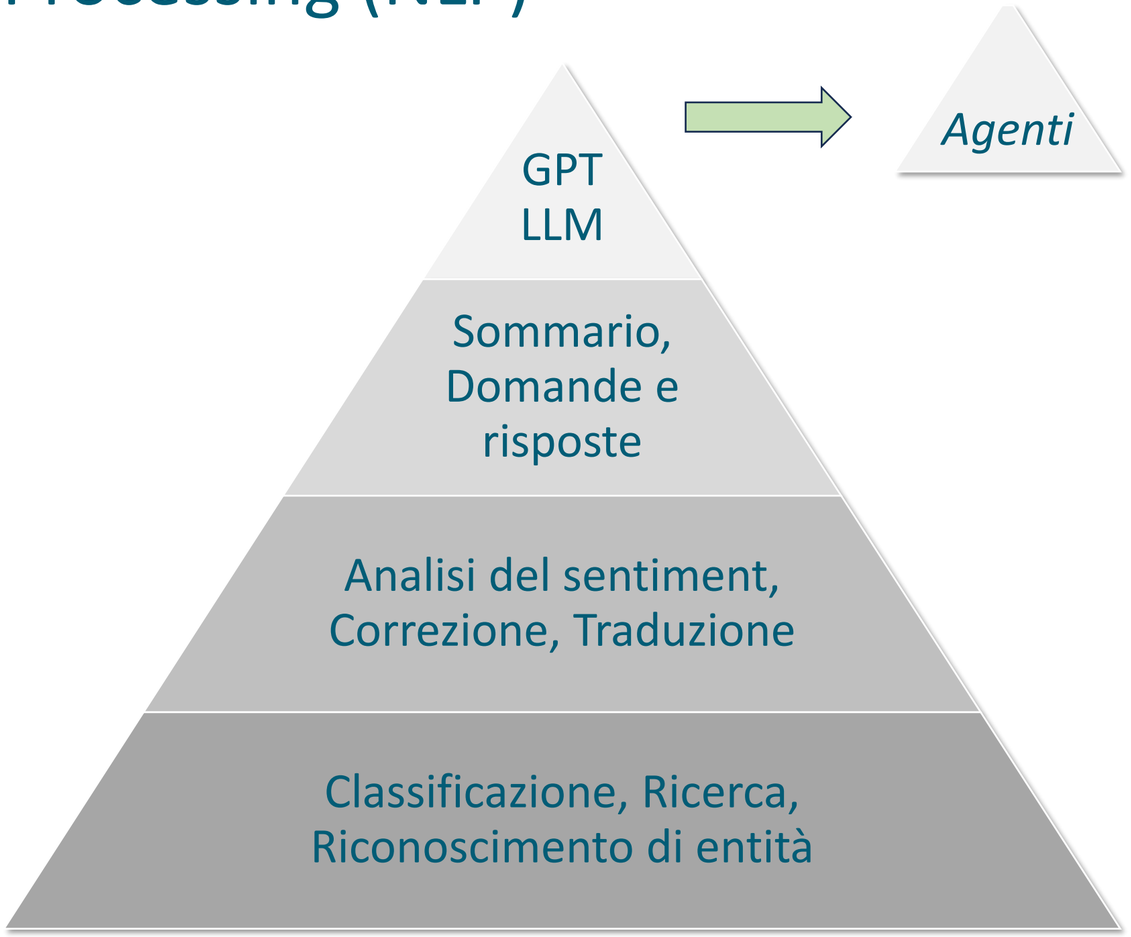
Utilizzi tipici del Natural Language Processing (NLP)

Domande e Risposte (Question Answering)

- Rispondere a domande specifiche basandosi su un contesto o una base di conoscenza.
- Esempio: Rispondere alla domanda "Chi è il CEO di Tesla?" leggendo da un articolo.

Chatbot e Assistenti Virtuali

- Automazione di conversazioni per servizio clienti, supporto tecnico o assistenza personale.
- Esempio: Siri, Alexa, ChatGPT.



ANALISI DOCUMENTALE E PROTEZIONE DATI

SmartDocs Srl – Analisi documentale e protezione dati

Scenario:
SmartDocs Srl, media azienda europea, deve gestire email e documenti contenenti dati sensibili di clienti (es: IBAN, codice fiscale, indirizzi, nomi, numeri di telefono).

L’azienda vuole automatizzare:
Estrazione di entità e dati sensibili (NER, pattern matching)
Riepilogo automatico e risposta alle richieste clienti
Tutelare la privacy (alcuni dati NON devono mai lasciare il server locale)
Minimizzare i costi cloud e garantire risposte rapide

ANALISI DOCUMENTALE E PROTEZIONE DATI

Useremo due modelli LLM per la soluzione aziendale

Architettura semplificata

1. **Modello locale open source (es: TinyLLaMA)**
    1. Viene eseguito localmente, direttamente sul vostro computer o su server aziendali.
    2. Si occupa delle operazioni più “sensibili”, come l’estrazione di dati personali e la protezione della privacy (Named Entity Recognition e masking dei dati).
    3. È veloce, economico, e mantiene i dati riservati all’interno dell’azienda.

2. **Modello cloud avanzato (es: Azure OpenAI GPT-3.5/4)**
    1. Viene utilizzato tramite API esterne, in cloud.
    2. Si occupa di attività più complesse come il riepilogo automatico, l’analisi semantica e la generazione di risposte ai clienti.
    3. Permette di gestire testi lunghi e offre maggiore potenza di calcolo e qualità delle risposte.

ANALISI DOCUMENTALE E PROTEZIONE DATI

L’obiettivo:

Sfruttare **i punti di forza di entrambi i modelli**:

- La privacy e la velocità del modello locale
- La potenza e la flessibilità del modello cloud

Garantire che **i dati più delicati non escano dall’azienda**, mentre sfruttiamo le migliori tecnologie disponibili per la produttività e l’automazione.

Distribuzione dei lavori

- Tutti lavorano sullo **stesso problema reale** ma con strumenti diversi.

Lavoro locale autonomo

- privacy, sicurezza, regex/NER, masking.

Lavoro cloud con l’aiuto dell’istruttore

- prompt, parametri, API, gestione delle risposte.

Altri componenti futuri per i giorni successivi

- RAG, contestualizzazione, configurazione, deployment ecc.
- La **collaborazione** tra i modelli con una architettura AI.

RUOLI DEI DUE MODELLI

1. Modello locale (TinyLLaMA o simili) — “Difensore/controllore”
Viene **eseguito localmente.**
Obiettivi:

- **NER (Named Entity Recognition)**: trova nomi, indirizzi, IBAN, codice fiscale, numeri, ecc.
- **Pattern Detection**: segnala se sono presenti dati sensibili o “red flag”.
- **RAG** (Retrieval Augmented Generation) opzionale: usa una knowledge base aziendale locale per arricchire le risposte.
- **Controlla l’output** prima che venga inviato al modello cloud, mascherando o anonimizzando i dati sensibili (es. sostituendo “Mario Rossi” con “[NOME]”).

2. Modello cloud (Azure GPT-3.5, GPT-4, ecc) — “Analista/colloquio”
Viene usato tramite API cloud, guidato passo-passo.
Obiettivi:

- Riceve documenti **già “ripuliti”** o parzialmente anonimizzati dal modello locale.
- Esegue:
  - Riepilogo (summary)
  - Analisi semantica
  - Generazione di risposte per i clienti
- Gestisce solo dati che non violano la privacy.

ESEMPIO DI PIPELINE COLLABORATIVA

Input:
L’utente carica un documento/email.

Passo 1 (locale):
Il modello locale fa NER, segnala dati sensibili e li anonimizza (es: “Mario Rossi” → “[NOME]”, “IT60X0542811101000000123456” → “[IBAN]”).

Passo 2 (controllo):
L’output ripulito viene controllato/validato (gli esperti possono anche vedere che i dati sono davvero rimossi).

Passo 3 (cloud):
Il testo anonimizzato viene inviato all’API Azure che fa il riepilogo, classifica la richiesta e prepara una risposta.

Passo 4 (output):
L’output finale può essere ricomposto localmente, reinserendo alcune entità dove permesso, oppure consegnato così.

SCEGLIERE IL PROVIDER E PROCURARE IL MODELLO

Per usare i LLM via API o localmente, dobbiamo **scegliere un provider** e individuare il modello adatto al nostro task (es: NER, generazione testo).

Provider principali e come scegliere

Azure OpenAI

Cos'è:
Servizio cloud di Microsoft, offre modelli GPT (es. GPT-3.5, GPT-4) via API.

Tiers (Livelli di servizio):

- **Trial/Free:** Crediti gratuiti iniziali per test e sviluppo.
- **Pay-as-you-go:** Paghi solo per quanto usi, adatto a uso flessibile.
- **Enterprise:** Per aziende con bisogni avanzati, offre SLA e supporto.

Come funziona:
Il modello resta in cloud, si accede tramite endpoint API, gestione centralizzata di sicurezza e billing.

SCEGLIERE IL PROVIDER E PROCURARE IL MODELLO

Hugging Face (HF)

Cos'è:
Community open source e servizio cloud con migliaia di modelli AI pronti, scaricabili o usabili tramite API.

Types (Categorie di modelli):

- **Text generation** (generazione testo)
- **NER** (Named Entity Recognition)
- **Summarization** (riepilogo)
- **Translation** (traduzione)
- **Classification** (classificazione)
- **Embeddings** (vettorizzazione)

Come funziona:
Si può:

- Scaricare e usare modelli **in locale** (privacy totale, serve hardware)
- Usare **endpoint cloud** (gratuito con limiti, poi piani a consumo)

SCEGLIERE IL PROVIDER E PROCURARE IL MODELLO

In sintesi (per qualsiasi progetto):

(Dopo aver analizzato il progetto, individuato scopi, disegnato architettura, ecc.) – fatto già per l'esercizio.

- Conosci i vari modelli
- Benchmark su dataset
- Scegli il provider (Azure, Hugging Face, ecc.)
- Cerca il modello giusto per il tuo compito (es: NER, text generation)
- Controlla il tier (Azure) o il type (HF) per capire costi, modalità di accesso e limiti.

CONCETTI UTILI PER L'IMPLEMENTAZIONE

Authentication

Cos'è:

Processo per identificarsi e autorizzare l'accesso alle API (token, chiave API).

Nota:

Ogni chiamata API necessita della chiave corretta; non condividere pubblicamente!

Rate Limit

Definizione:

Numero massimo di richieste API consentite in un intervallo di tempo (es: 60/min).

Superamento:

Porta a errori temporanei (es: 429 Too Many Requests).

Retry e Back-off

Retry:

Tentare nuovamente la richiesta dopo un errore temporaneo.

Back-off:

Aumentare progressivamente l'intervallo tra i retry (es: 1s, 2s, 4s...) per evitare "flooding" dell'API.

CONCETTI UTILI PER L'IMPLEMENTAZIONE

Endpoints

Definizione:

URL specifico per inviare richieste API al servizio desiderato (es: /v1/chat/completions).

Varia per modello e provider.

Endpoint Types

Esempi:

- Chat (conversazione)
- Completion (completamento testo)
- Embedding (vettorizzazione)

Autoscaling

Cos'è:

Capacità del provider cloud di allocare più risorse automaticamente in base al carico.

Vantaggio:

Performance costante anche con molti utenti.

CONCETTI UTILI PER L'IMPLEMENTAZIONE

Batch vs Streaming Completions

Batch:

Invio e ricezione di molte richieste tutte insieme; risposta finale unica.

Streaming:

Ricezione della risposta "a pezzi" man mano che viene generata (utile per risposte lunghe / conversazione).

Model Types

Esempi:

- **Text-generation:** GPT, LLaMA, Falcon, TinyLLaMA. Mistral.
- **NER:** modelli per Named Entity Recognition.
- **Classification, Summarization, Translation:** altri task AI.

Pipelines (HF Transformers)

Cos'è:

Metodo rapido per usare modelli preaddestrati con una sola funzione (es: pipeline("ner"), pipeline("text-generation")).

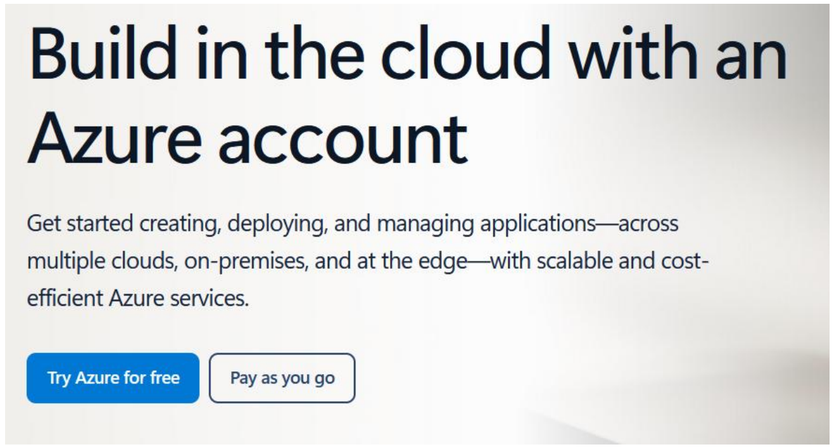
Vantaggi:

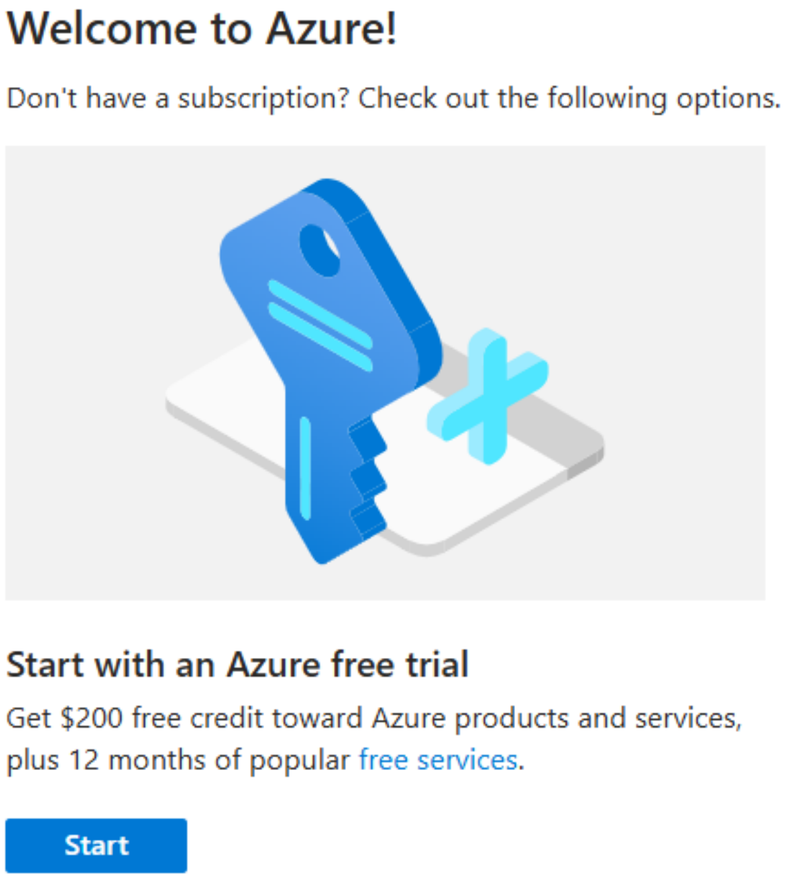
Semplifica l'uso, anche per principianti.

PERCORSO SU AZURE

Registrazione su Azure

1. Vai su https://azure.microsoft.com/free/
2. Clicca su **“Inizia gratis”**.
3. Registrati con la tua email personale e 2FA.
   - Servirà C.F. e **una carta di credito** per la verifica (**nessun addebito reale**).
   - Inserisci le informazioni richieste (nome, indirizzo, ecc.).
4. Attiva il tuo account Azure: avrai **$200 di credito gratis** per 30 giorni.

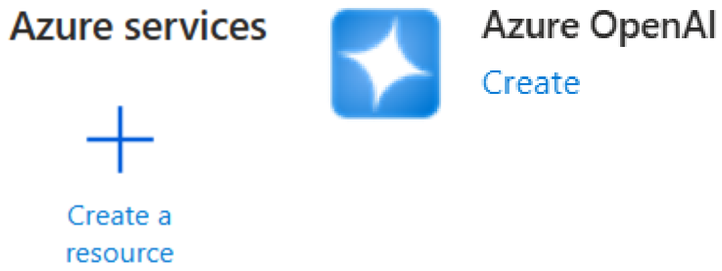


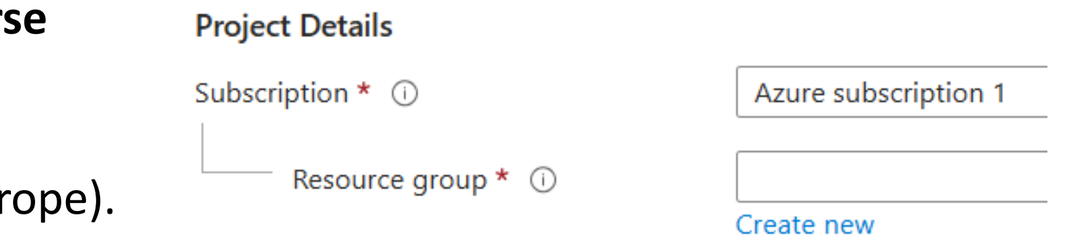


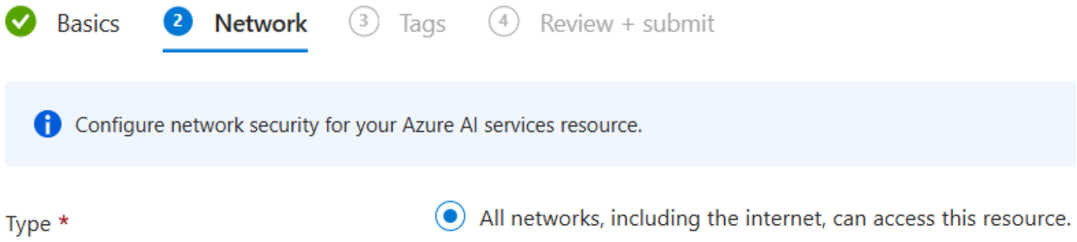
PERCORSO SU AZURE

Creazione della risorsa Azure OpenAI

1. Dopo il login, entra nel portale Azure.
2. Clicca su **“Crea una risorsa”** (in alto a sinistra).
3. Cerca **“Azure OpenAI”** e seleziona la risorsa corrispondente.
4. Clicca su **“Crea”**.
   - Scegli **abbonamento** (Subscription) e **gruppo di risorse** (Resource Group) (puoi crearne uno nuovo).
   - Dai un nome alla risorsa.
   - Scegli la regione (consigliata: West Europe, North Europe).
   - Clicca su **“Rivedi e crea”**, poi ancora su **“Crea”**.
5. Attendi la creazione (1–2 minuti).







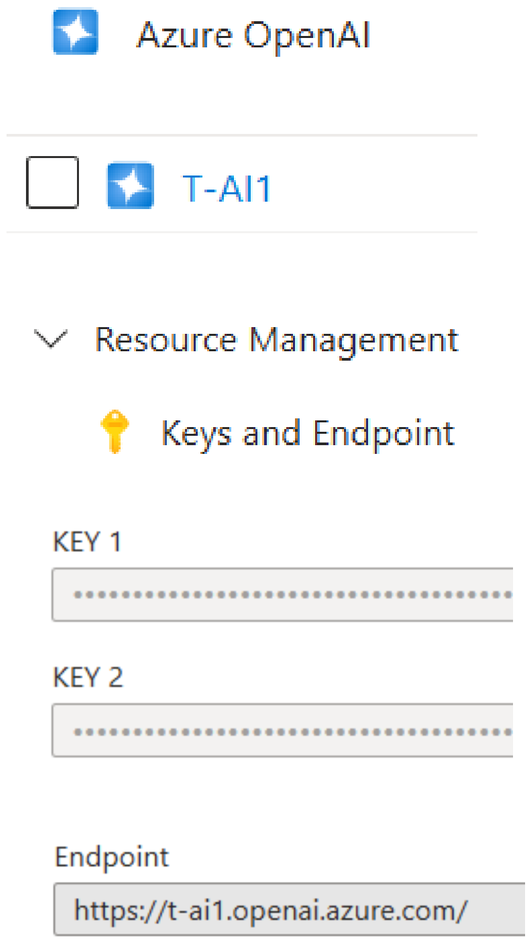
PERCORSO SU AZURE

Ottenere il token API (“chiave”)

1. Vai alla **risorsa Azure OpenAI** che hai appena creato.
2. Nel menu a sinistra, in Gestione delle risorse clicca su **“Chiavi e endpoint” (Keys and Endpoint)**.
3. Troverai due “Key” (API key): **copia una** di queste chiavi — ti servirà per autenticare le chiamate API.

Ottenere l’Endpoint

- Nella stessa pagina di **Keys and Endpoint** vedrai il tuo **Endpoint** personale, che inizia per https://…openai.azure.com/
- **Copia e incolla** questo endpoint: serve nel codice per collegarti.

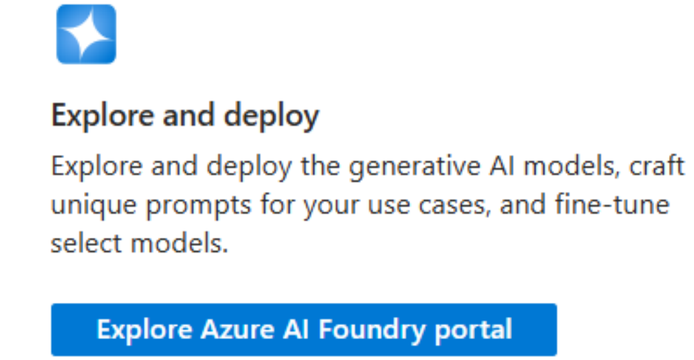


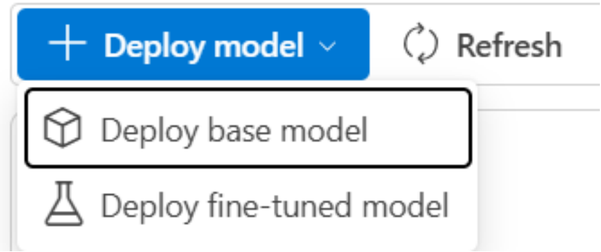
PERCORSO SU AZURE

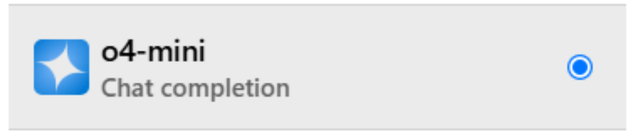
Creare un Deployment Model

1. Nel menu della risorsa OpenAI, vai su **“Deployments”** (Deployment dei modelli) e **Deploy model**.
2. Clicca su **“Deploy model”**.
3. Scegli il modello (es: o4-mini).
4. Dai un nome unico al deployment (es: “chatgpt-demo”).
5. Conferma.

Il nome scelto sarà usato come engine nel codice Python!



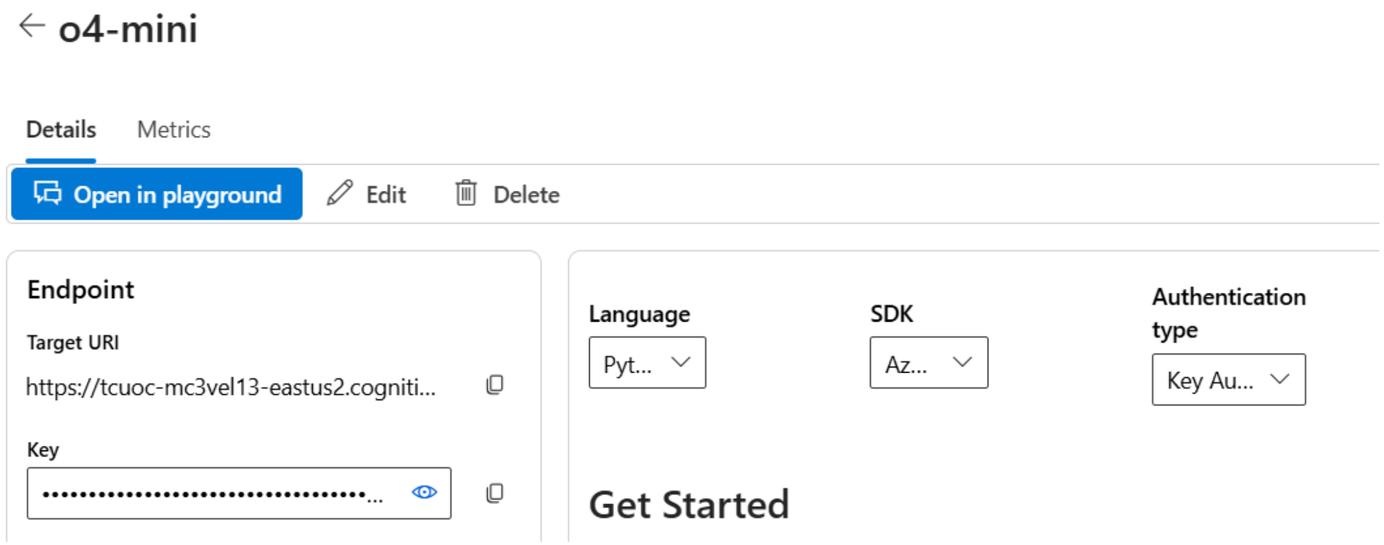




Sei pronto!

- Ora hai:
  - **Token API** (la chiave)
  - **Endpoint**
  - **Nome del deployment**
- Puoi usarli subito nel codice Python (vedi risposta precedente).

PERCORSO SU AZURE



3. Run a basic code sample

This sample demonstrates a basic call to the chat completion API. The call is synchronous.

```
endpoint =
```

```
subscription_key =
```

```
 raise self._make_status_error_from_response(err.response) from None
openai.NotFoundError: Error code: 404 - {'error': {'code': 'DeploymentNotFound', 'message': 'The API deployment for this resource does not exist. If you created the deployment within the last 5 minutes, please wait a moment and try again.'}}
```

PERCORSO SU AZURE

```python
import openai

# Crea il client Azure OpenAI
client = openai.AzureOpenAI(
    api_key="LA_TUA_KEY",  # <-- La tua chiave API di Azure OpenAI
    azure_endpoint=«IL_TUO_ENDPOINT",  # <-- Il tuo  API endpoint di Azure OpenAI
    api_version="2024-12-01-preview",  # <-- O la versione sul portale Azure
)

response = client.chat.completions.create(
    model="o4-mini",  # <-- il nome esatto del deployment in Azure
    messages=[
        {"role": "system", "content": "Sei un assistente AI."},
        {"role": "user", "content": "Qual è la capitale dell'Italia?"}
    ],
    max_completion_tokens=256, # <-- Valore minimo per ottenere output
    temperature=1, # <-- Non puoi usare altro qui
)

print(response.choices[0].message.content)
```

```
>python G5E1_Azure_OpenAI.py
La capitale d'Italia è Roma.
```

PERCORSO SU HUGGING FACE

- **Registrati** gratis su Hugging Face.
- **Crea il token** (Settings > Access Tokens > New token > "Read" permission).
- **Copia il token** nel tuo codice.
- **Installa la libreria requests:**

```
pip install requests
```

- **Lancia il codice**! (Funziona anche su Colab)

DOMANDE?

PAUSA

DOMANDE?

Cominciamo il lavoro!

GRAZIE PER L'ATTENZIONE