LLMstudio

Powering Enterprise Grade LLM Applications With Open-Source

tensorops

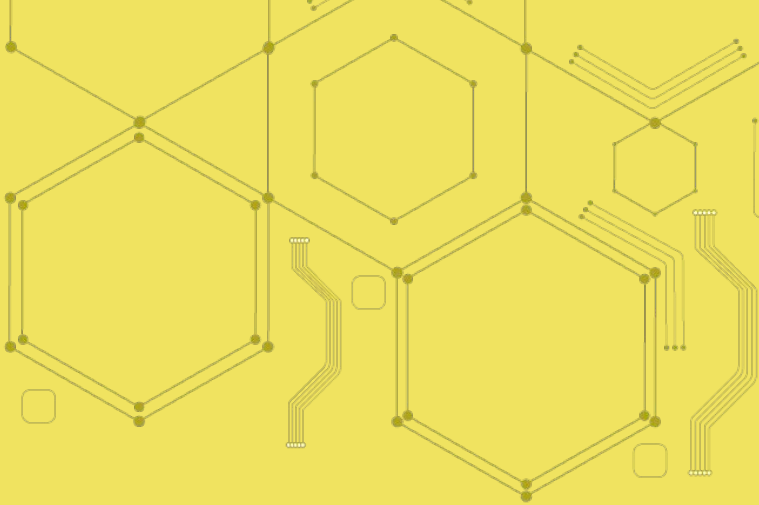
Cláudio Lemos

Co-Founder **@TensorOps**
linkedin.com/in/~claudio

tensorops

Agenda

1. Working with LLM
2. How to access LLMs in production
3. LLM Proxies
4. LLMstudio
5. Powering enterprise grade LLM apps



Working with LLMs

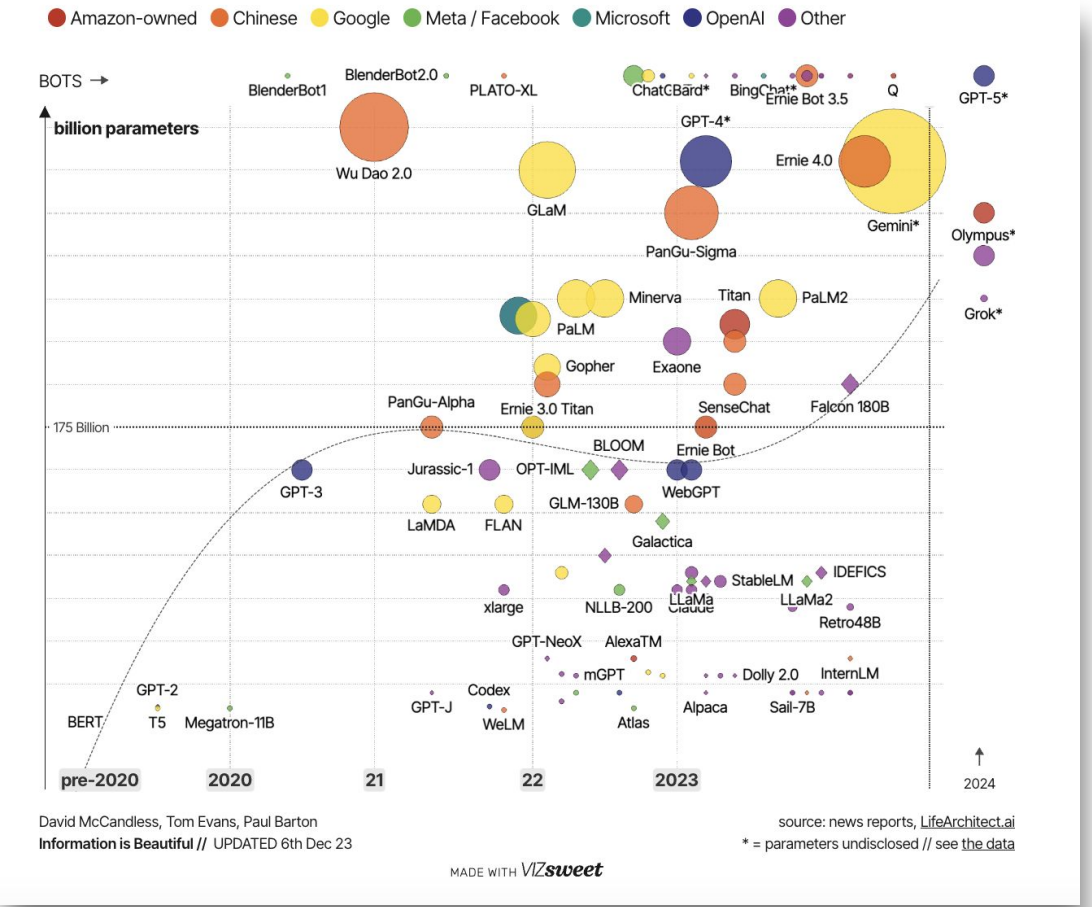
AI Applications

Chatbots

Code assistants

Automated analysis

The Evolution of LLMs



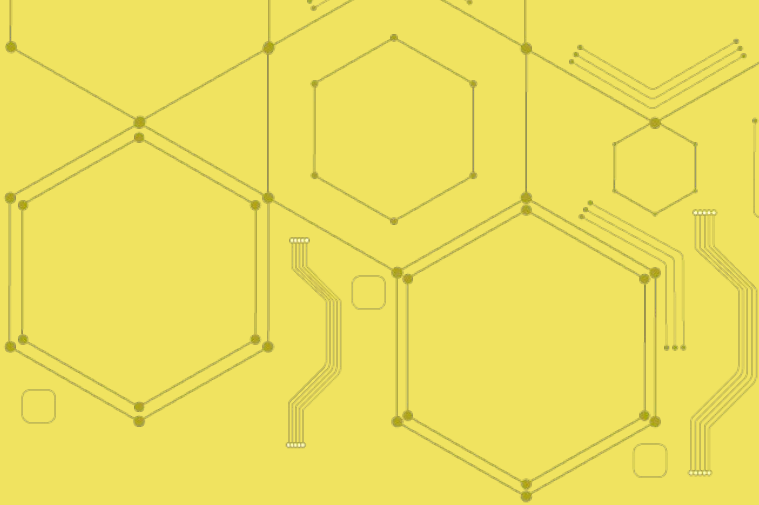
https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

tensorops

Infrastructure of hosting LLMs

Model Name	Default Instance Type	Price per Hour	# of GPUs	RAM (GiB)	vCPUs	Monthly Price (USD)
Llama-2-7b	ml.g5.2xlarge	$1.21	1	32	8	$872
Llama-2-13b	ml.g5.12xlarge	$5.67	4	192	48	$4,083
Llama-2-70b	ml.g5.48xlarge	$16.29	8	768	192	$11,727

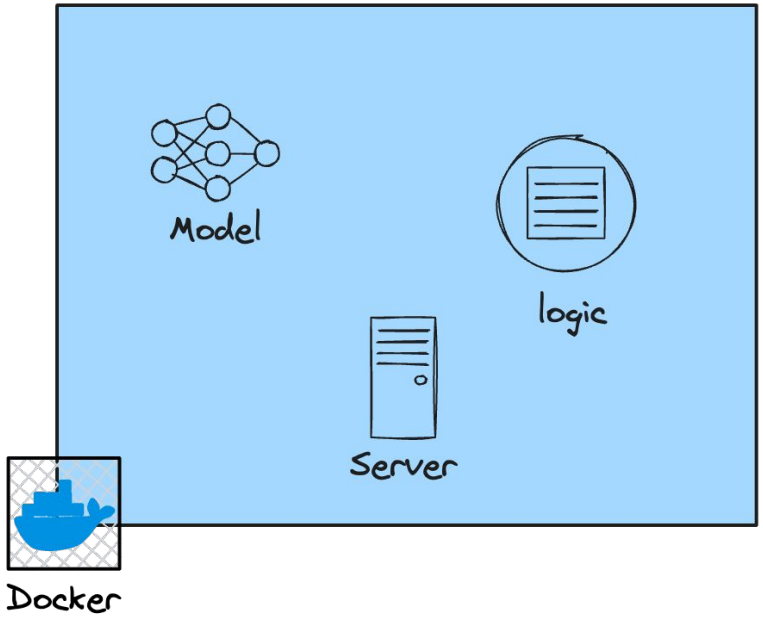
https://aws.amazon.com/blogs/machine-learning/llama-2-foundation-models-from-meta-are-now-available-in-amazon-sagemaker-jumpstart/

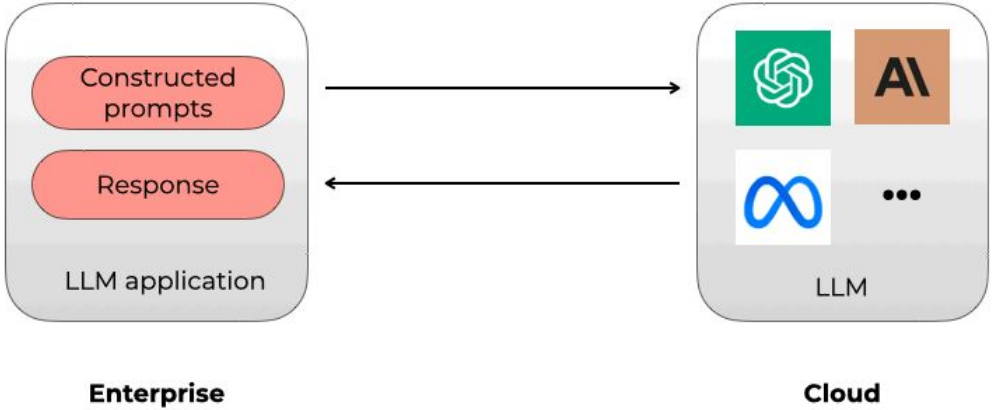


How to access LLMs in production

Traditional ML

could be packaged in a single container




Constructed prompts
Response
LLM application
LLM
Enterprise
Cloud

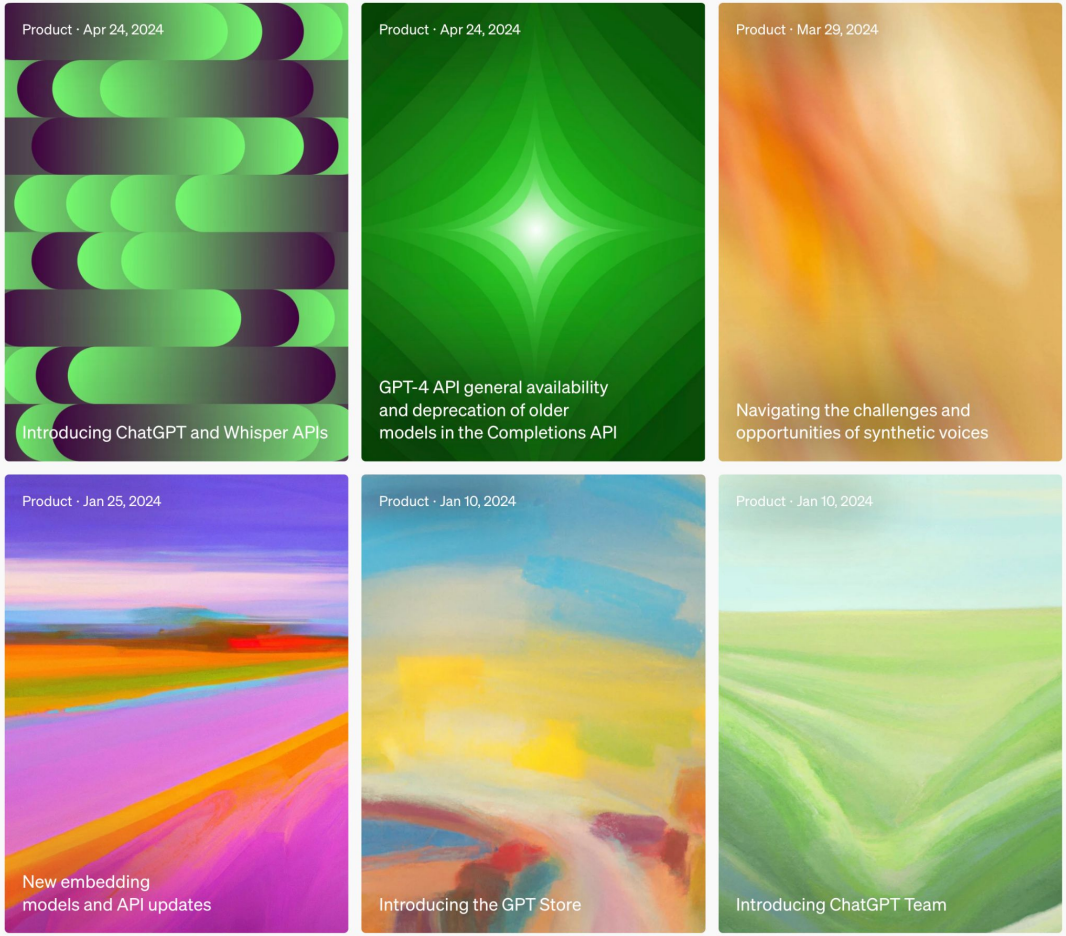
Different vendors = Different API & SDK

```python
import os
from anthropic import Anthropic

client = Anthropic(
    # This is the default and can be omitted
    api_key=os.environ.get("ANTHROPIC_API_KEY"),
)

message = client.messages.create(
    max_tokens=1024,
    messages=[
        {
            "role": "user",
            "content": "Hello, Claude",
        }
    ],
    model="claude-3-opus-20240229",
)
print(message.content)
```

```python
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a he
    {"role": "user", "content": "Hello!"}
  ]
)
```

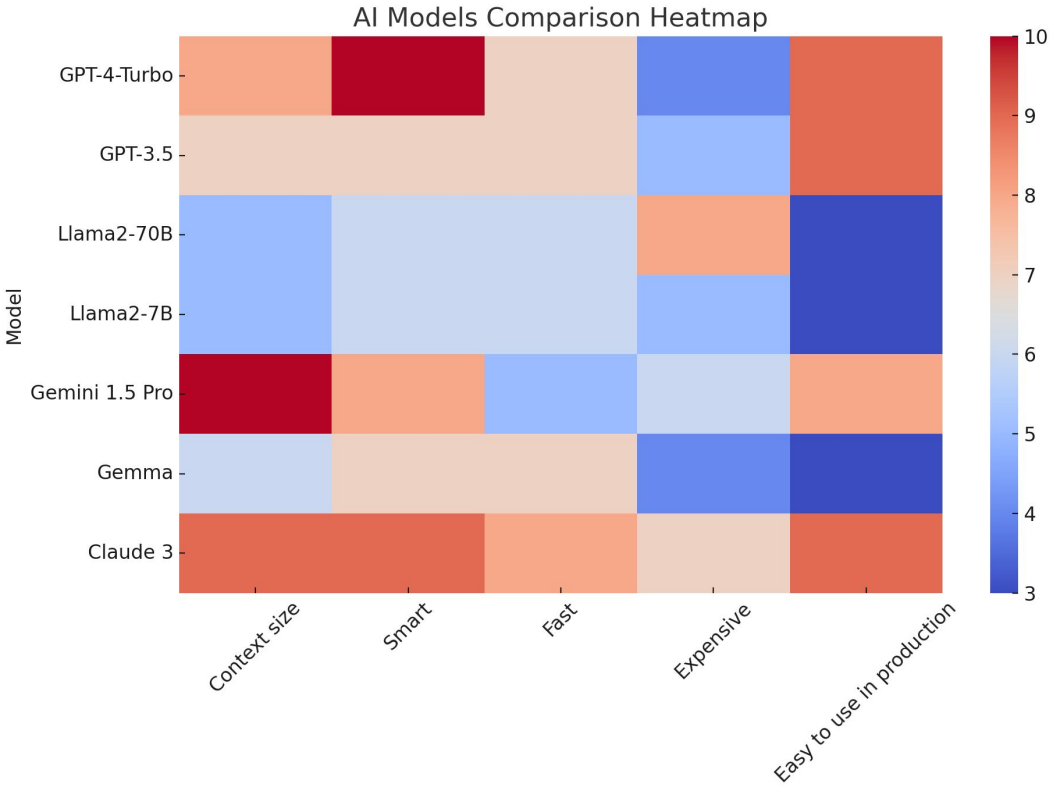
Adapt fast to code changes

LLMstudio

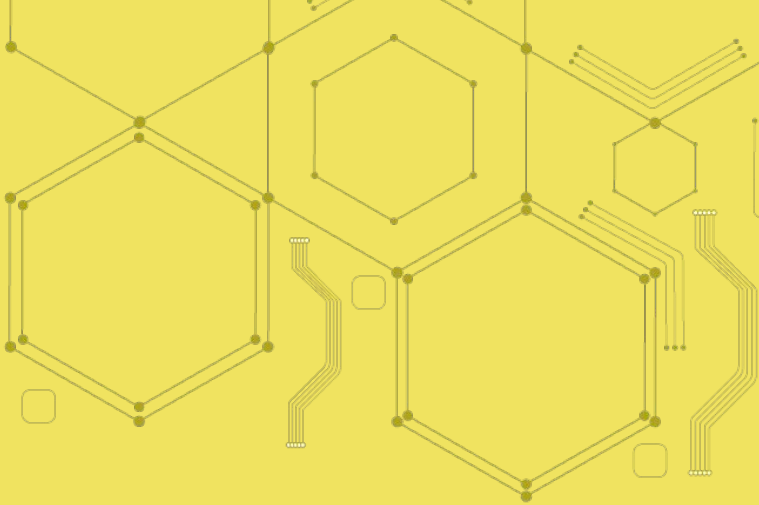


tensorops

What is the right LLM for me?



LLMstudio



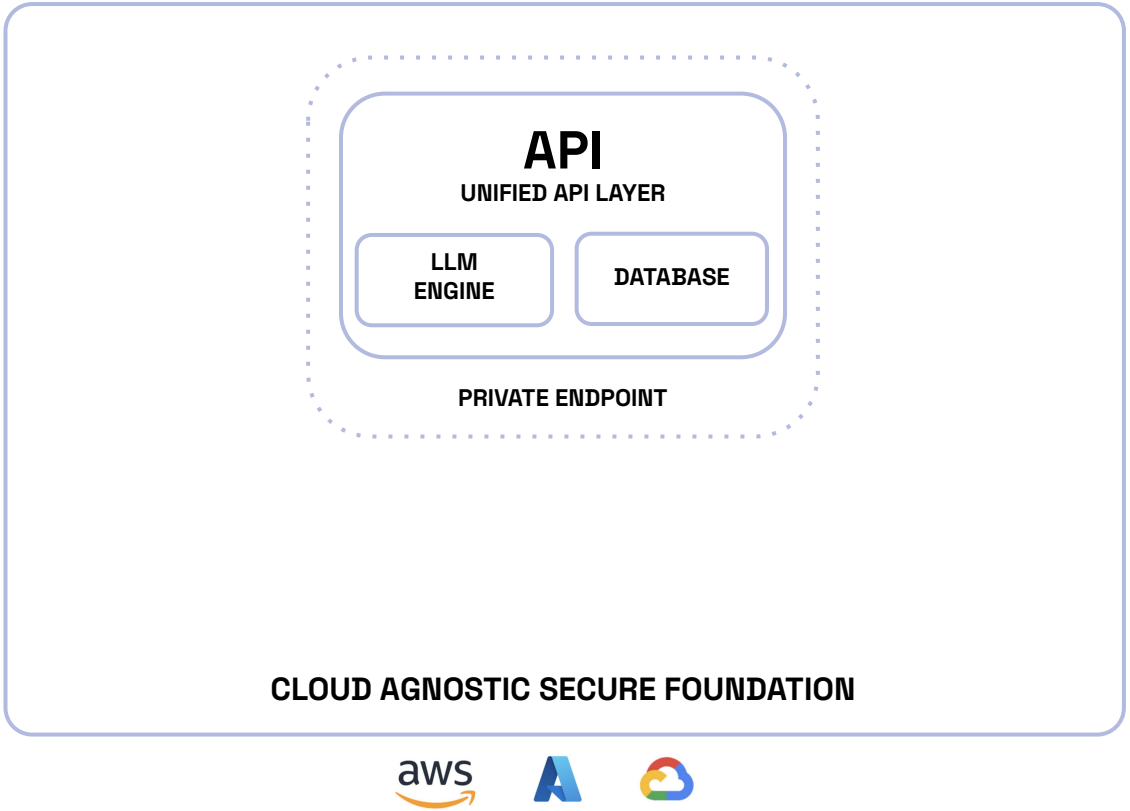
Introducing LLM Proxy

Unified LLM API Access

tensorops

LLMstudio

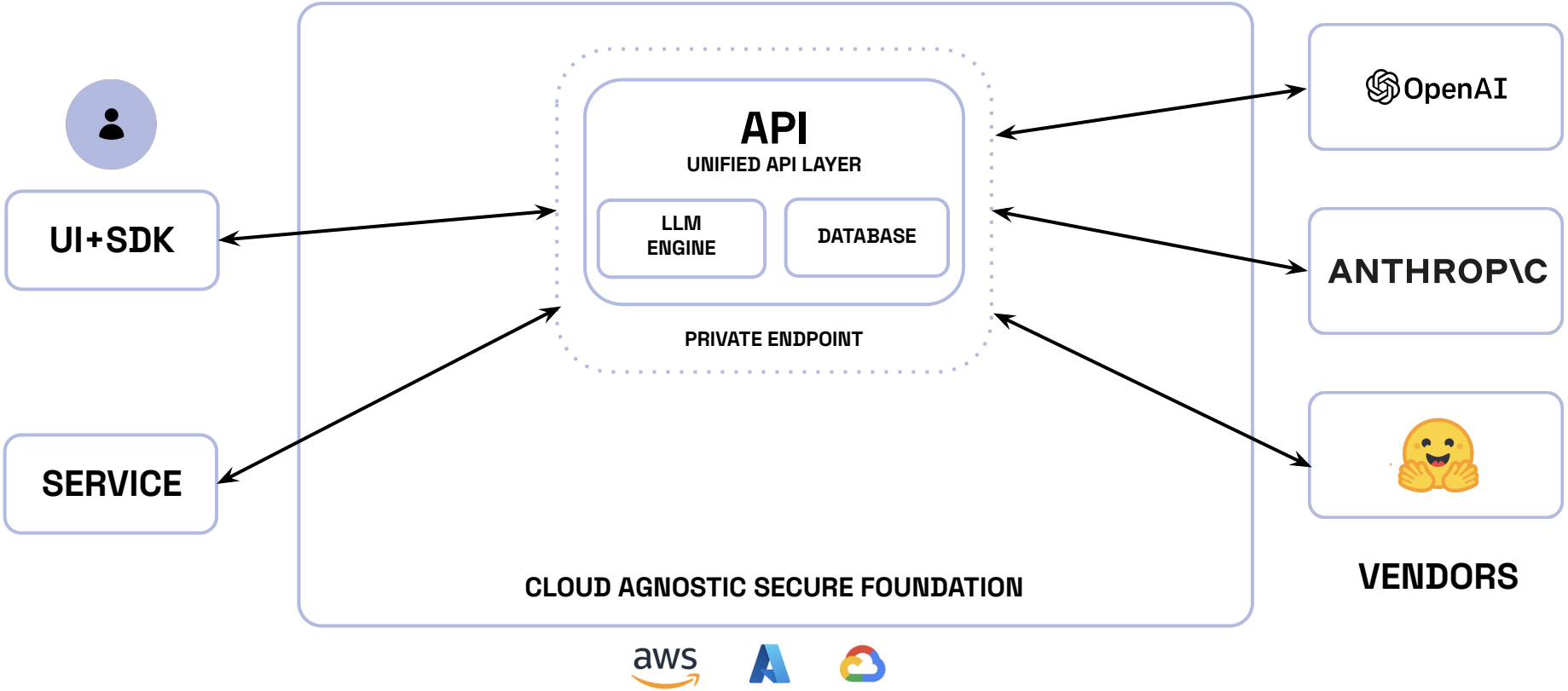
HIGH LEVEL OVERVIEW



tensorops

LLMstudio

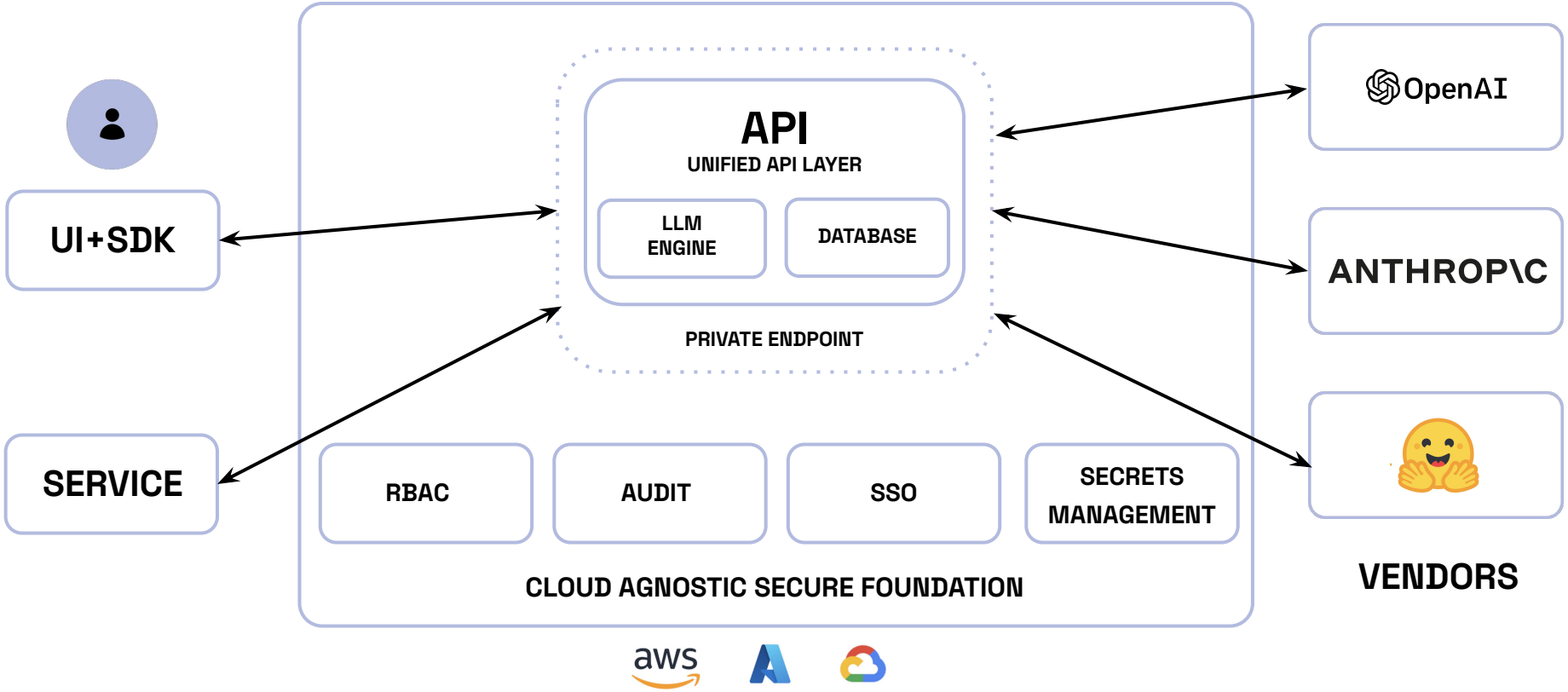
HIGH LEVEL OVERVIEW



tensorops

LLMstudio

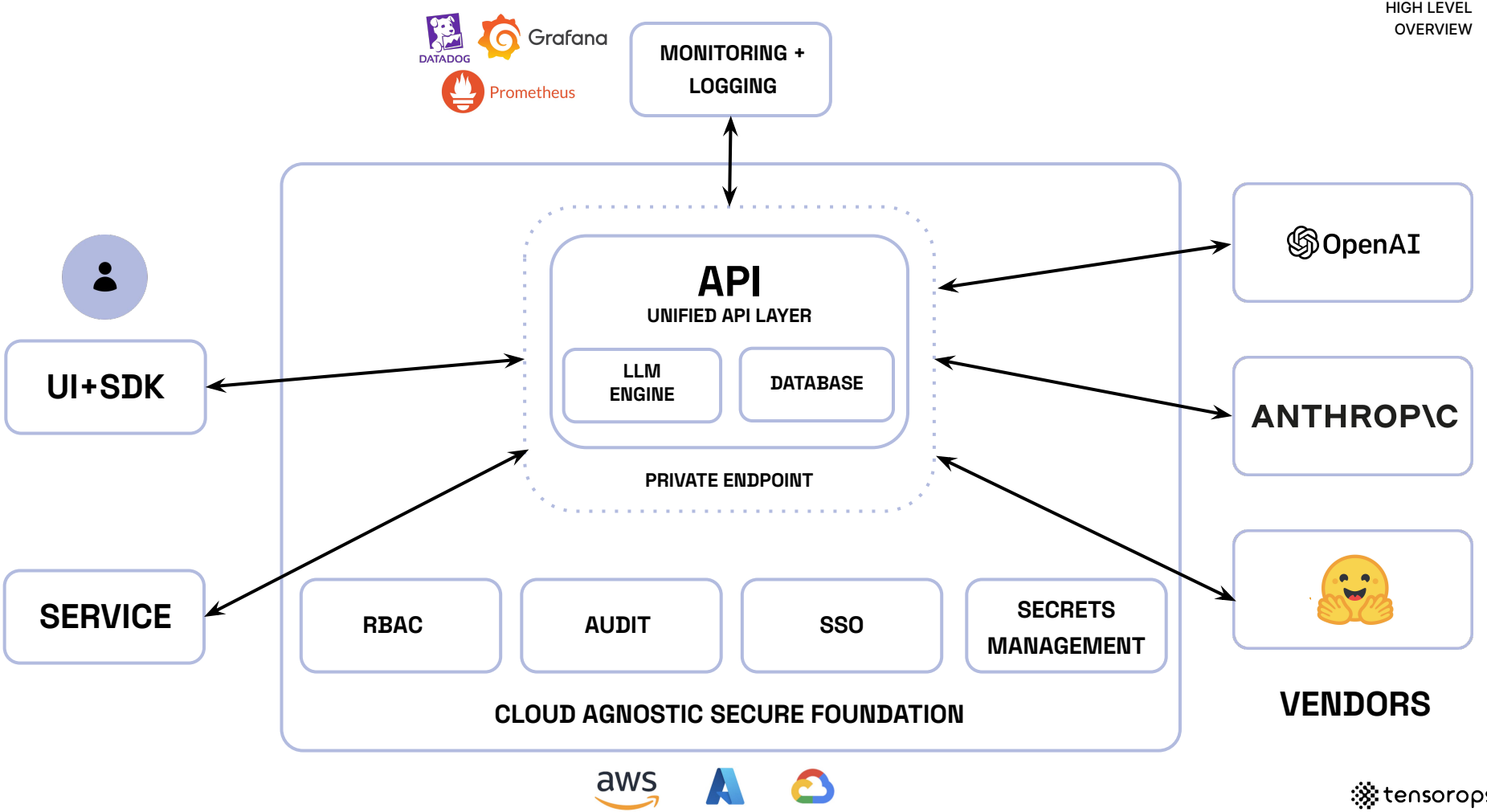
HIGH LEVEL OVERVIEW



tensorops

LLMstudio

HIGH LEVEL OVERVIEW



tensorops

LLMstudio

LLMstudio

Playground Compare Docs

Model: gpt-3.5-turbo-1106

Input Submit

Insert your prompt here...

Output

Export data

ID	Input	Output	Model	Cost	Input Tokens	Output Tokens	Latency	TTFT
1	Write me a poem about trains	Steel serpents slithering on...	openai gpt-4-1106-preview	$0.009240	6	306	29.681s	1.093s

LLMstudio

LLMstudio

Playground Dashboard Docs

Model: gpt-4-1106-preview

Search Models...

Anthropic

claude-2.1

claude-2

claude-instant-1.2

claude-instant-1

Ollama

llama2

OpenAI

gpt-4-1106-preview ✓

gpt-4

gpt-4-0314

Submit

Output

LLMstudio

LLMstudio.ai

```
>> pip3 install LLMstudio

>> LLMstudio server
```

#LLMstudio

tensorops

Quickstart

```
from llmstudio import LLM
model = LLM("anthropic/claude-2.1")
model.chat("What are Large Language Models?")
```

docs.llmstudio.ai/quickstart

LLMstudio standard logging

OpenAI standard format!

```
ChatCompletion(
    id='9faa33e3-cda0-4d23-9217-c82ff7325b94',
    choices=[
        Choice(
            finish_reason='stop',
            index=0,
            logprobs=None,
            message=ChatCompletionMessage(
                content="I am an artificial intelligence called OpenAI. I don't have
                role='assistant',
                function_call=None,
                tool_calls=None
            )
        )
    ],
    created=1718623278,
    model='gpt-4',
    object='chat.completion',
    system_fingerprint=None,
    usage=None,
    session_id=None,
    chat_input="What's your name",
    chat_output="I am an artificial intelligence called OpenAI. I don't have a perso
    context=[{'role': 'user', 'content': "What's your name"}],
    provider='openai',
    timestamp=1718623280.9204,
    parameters={'temperature': None, 'max_tokens': None, 'top_p': None, 'frequency_p
    metrics={
        'input_tokens': 4,
        'output_tokens': 23,
        'total_tokens': 27,
        'cost_usd': 0.0015,
        'latency_s': 2.6993088722229004,
        'time_to_first_token_s': 0.9924077987670898,
        'inter_token_latency_s': 0.0711073378721873,
        'tokens_per_second': 9.26162998879499
    }
)
```

HIGH LEVEL OVERVIEW

Monitoring and Logging

LLMstudio

returns performances metrics for each API call (stream and non-stream)

Response


```
"context": [
  {
    "role": "user",
    "content": "Hello! Who are you?"
  }
],
"provider": "openai",
"timestamp": 1718622757.612226,
"parameters": {
  "temperature": 1,
  "max_tokens": 2048,
  "top_p": 1,
  "frequency_penalty": 0,
  "presence_penalty": 0
},
"metrics": {
  "input_tokens": 6,
  "output_tokens": 26,
  "total_tokens": 32,
  "cost_usd": 0.000061,
  "latency_s": 1.0556859970092773,
  "time_to_first_token_s": 0.8302950859069824,
  "inter_token_latency_s": 0.008317514702125831,
  "tokens_per_second": 26.523038175483098
}
}
```


DATADOG

Grafana

Prometheus

Monitoring and Logging

```
class CustomProvider(Provider):
    def calculate_custom_metrics(self, input: str, output: str, model: str):
        # Custom metrics calculation
        hello_counter = output.lower().count('hello')
        return {'hello_counter': hello_counter}
```

Custom Metrics callback

Monitoring and Logging

Building monitoring pipelines

Cost reduction

Analytics tools

Looker

DWH / Logging System

Google BigQuery

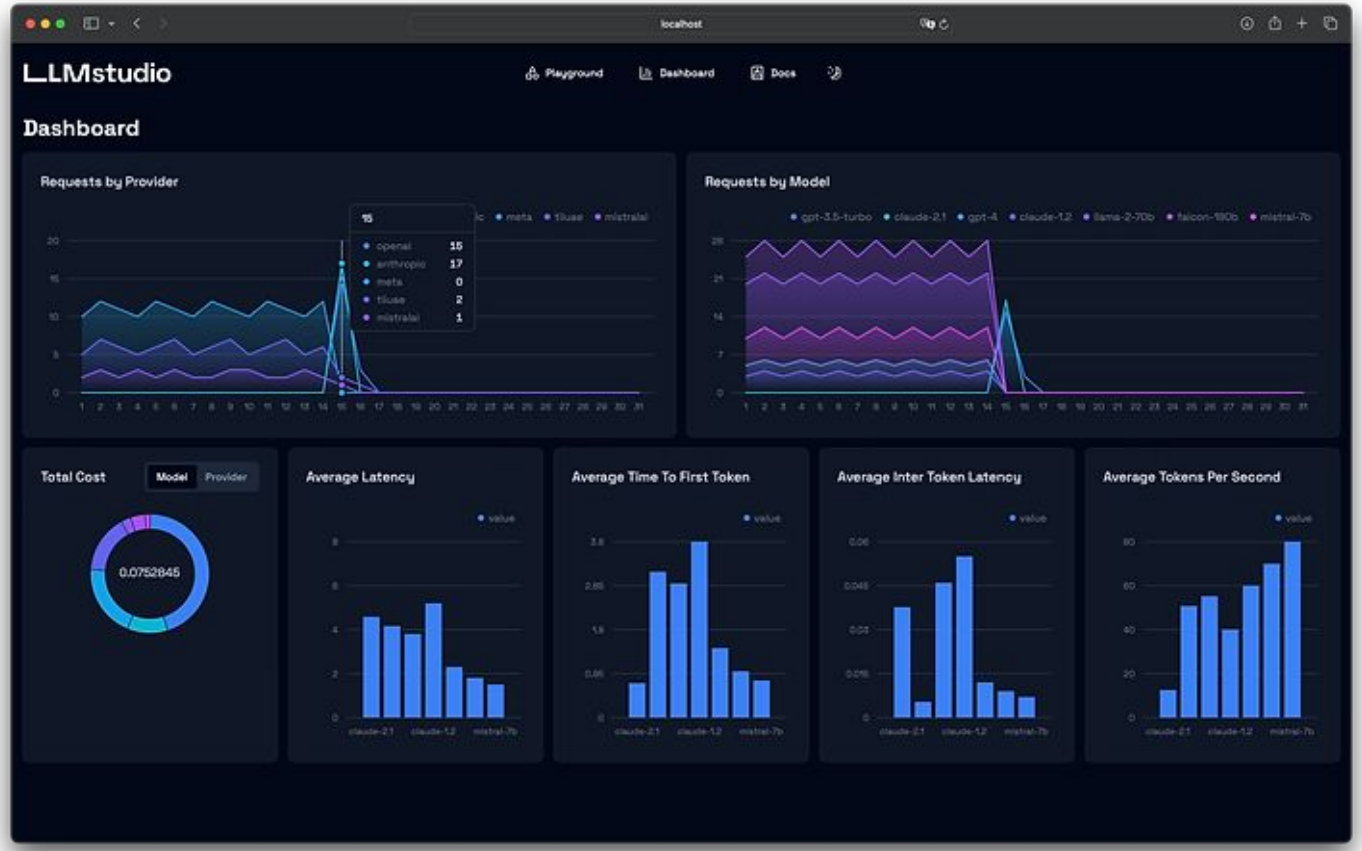
CSV

Logger / API Gateway

LLMstudio

LLM Vendor

LLMstudio



tensorops

Smart Routing
and Fallback

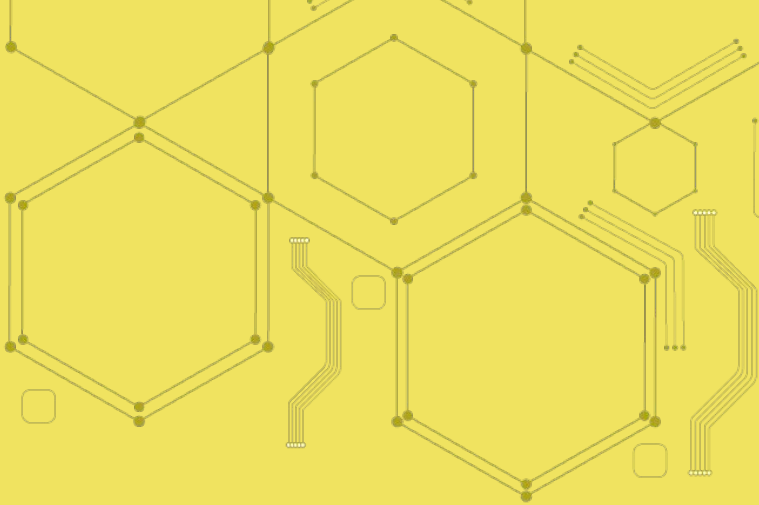
Batch calling

```python
from llmstudio import LLM
gpt4 = LLM("openai/gpt-4o")

inputs = [
    "What's your name?",
    "Tell me a joke.",
    "What's the weather like?",
    "Can you sing a song?",
    "Tell me about yourself."
]


responses = gpt4.batch_chat(inputs, num_threads=10)

for i, response in enumerate(responses):
    print(f"Input: {inputs[i]}")
    print(f"Response: {response.chat_output}\n")
```

Langchain integration

```python
from langchain.tools import tool
from langchain.agents import AgentType, initialize_agent
from llmstudio.llm.langchain import ChatLLMstudio

@tool
def get_departure(ticket_number: str):
    """Use this to fetch the departure time of a train"""
    return "12:00 AM"


def assistant(question: str)->str:
    tools = [get_departure]
    gpt3 = ChatLLMstudio(model_id='openai/gpt-3.5-turbo', temperature=0)

    agent_executor = initialize_agent(
        tools, gpt3, agent=AgentType.OPENAI_MULTI_FUNCTIONS
    )

    response = agent_executor.invoke(
        {
            "input": question
        }
    )

    return response

assistant('When does my train depart? My ticket number is 1234')
```

LLMstudio

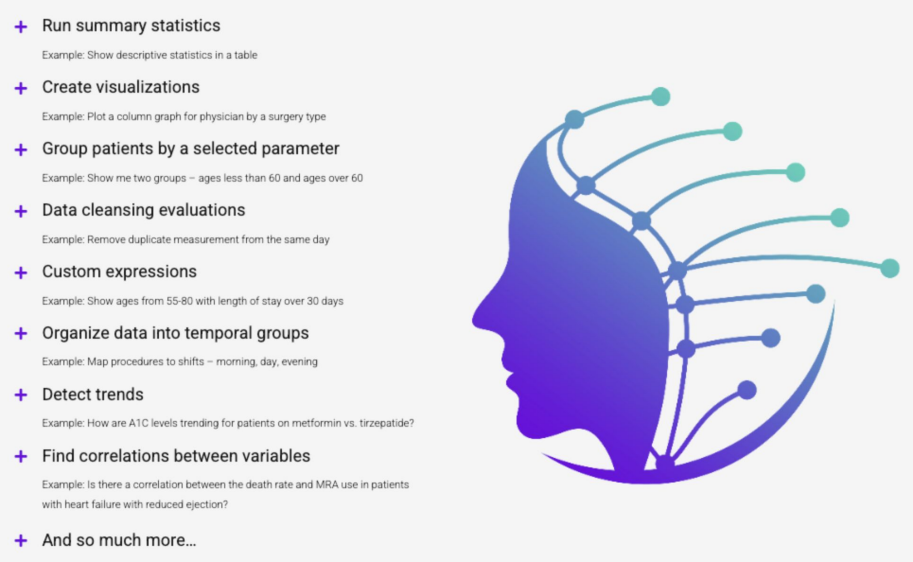


Powering Entreprise Grade

LLM applications

tensorops

MDclone's ADAM chatbot



- Allows non-technical users to perform complex data queries through a simple chat interface.
- Achieved MVP in three months using **LLMstudio** for routing, monitoring, and logging LLM calls.
- Extended support for LangChain to build complex agents and chains.

Quick Demo

LLMstudio

- LLM proxy access to the latest LLMs
- UI + SDK
- Prompt playground
- Unified OpenAI standard for all requests
- Monitoring and Logging
- LangChain integration
- Custom and local LLM support through Ollama
- Smart routing and fallback
- Batch calling
- Type casting