AI

Supporting Scientific Computing

MANIAC LAB

Ilija Vukotic
University of Chicago

ATLAS
EXPERIMENT

Preface

10^9 people are trying to use AI to do more/faster/better/cheaper/easier.

Physicists and computer scientists supporting them are no different.

My perspective comes from things that I use daily:

Analysis Facility, UC Analytics, WFMS, Rucio/FTS, CRIC, … and discussions I had with some of ATLAS sites.

Nobody figured it out completely yet and thing are changing every day.

Thanks to Lincoln Bryant for his comments.

Scope

There are two distinct uses for AI/ML techniques:

Directly impacting physics

Low

level

High

- CNNs on our FPGAs
- Generative methods in MC generation
- Analysis
- Suggesting new searches

Helping physicists do physics

Low

level

High

- Writing/Exposing documentation
- Help coding
- Help understanding state / detecting issues
- Help running systems

Will discuss only this.

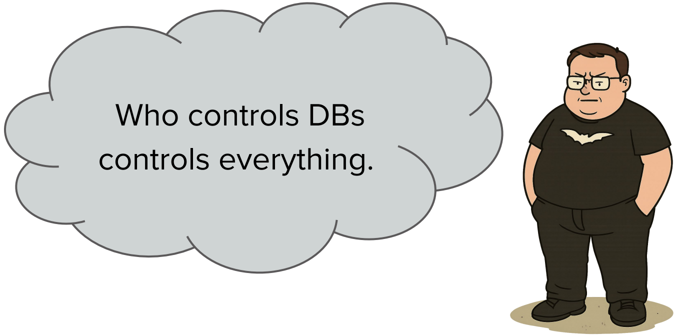
	Keeping informed	Alerts	Fixes
Site Admins	Is my site running smoothly? What are problems with my site? How many slots with 32GB/core are currently available?	Alert me on any issue. Give me details. Only problems we can fix.	Stop black hole node. Restart a node with a CVMFS issue. Renew cert Update software
Users	Are my jobs running correctly? What's the status of this task? What problems are associated with this task?	Alert me on failing jobs/tasks. Give details on why are they failing. Is it something I can fix or not?	Stop tasks Change submit script Create support ticket
Service Operators	How many active transfers to site XYZ? What is disk usage on RSE XYZ? What is task mixture? Is site XYZ “full”? What is bandwidth used? How are storage servers doing?	Issue with the service eg. FTS, Rucio Stuck tasks, idle sites Network links down Network saturations Security issues Broken disks, high load, slowness	Fix misconfigured FTS endpoints Report missing files to Rucio Change queue settings Restart XCache

Current situation

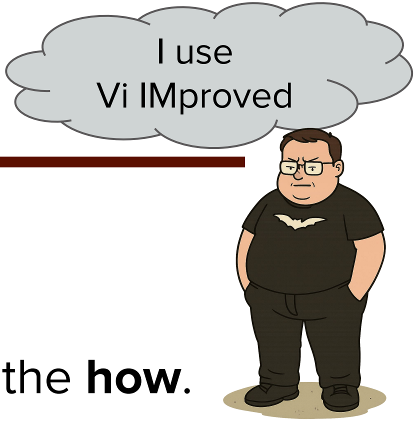
We have almost all the data needed to answer these questions. Some in DBs, some in analytics platform(s), some in logs.

Despite several attempts, classical ML methods were never capable enough, especially when crossing system boundaries.

AI would be capable but is all the required knowledge encapsulated in our DBs and documentation?



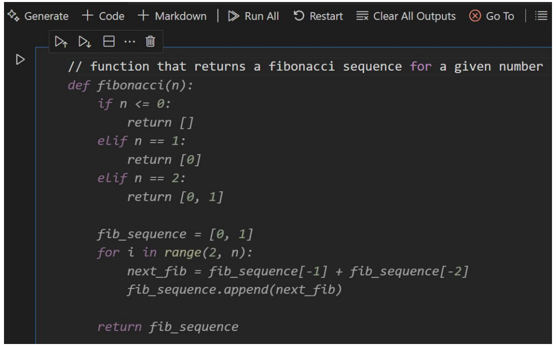
What can we do now?



Let AI make us more productive…

GitHub Copilot (… Gemini code assist, Q Developer, JetBrains AI, Cody)

- **Vibe coding** - you express the **what** and let AI handle most of the **how**.
- Reviews your code
- Produces documentation
- Creates issues
- Creates PRs for existing issues
- Assisted code review
- Workspace issue-to-PR flow. Drop an issue in; Copilot drafts a plan, generates code in a disposable branch, runs tests, and opens a PR.

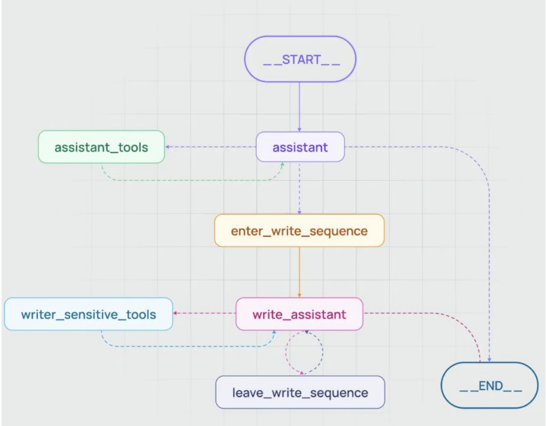


```
// function that returns a fibonacci sequence for a given number
def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]

    fib_sequence = [0, 1]
    for i in range(2, n):
        next_fib = fib_sequence[-1] + fib_sequence[-2]
        fib_sequence.append(next_fib)

    return fib_sequence
```

BIG QUESTION 1. How are people paying for access?

AI Agents

- Usually Reinforcement Learning fine-tuned LLMs
- Can access web - no cutoff dates
- Create and execute code - it is actually nicer language for a lot of tasks
- Use tools - read pdfs, OCR, file search, etc.
- Multi-agent collaborations have huge benefits
  - Can correct itself by backtracking
  - Much less prone to hallucinations
  - Harder to jailbreak
  - Human in the loop option
  - Much slower (1-20 min) and more expensive.



Packed with different options based on a vendor:

OpenAI - DeepSearch, WebBrowse, Operator,

LangChain (+LangGraph), Microsoft AutoGen, CrewAI, Manus, Google ADK ...

What can we do now?

Let the AI use a web browser...

- AI can use existing web frontends (eg. WebBrowse, OpenAI Operator, Google Project Mariner)
    - Extract data without a REST interface or DB access.
    - Let it do hard-to-automate, repetitive operations.
- Works for “regular” looking sites
- I am not holding my breath for it to be able to understand bigpanda, ami,...
- Even some of our regular sites have accessibility issues.

Go to atlas-cric.cern.ch, look for experiment sites and for each one check that only CERN frontier is present. If there are others delete them.

Browse twiki pages at xxx, find all pages where dataset YYY is mentioned, and change it to ZZZ.

Go to FTS website, find transfers from MWT2 and make throughput plot over last 6h. Split series per destination.

***check WebArena and OSWorld**

What can we do now?

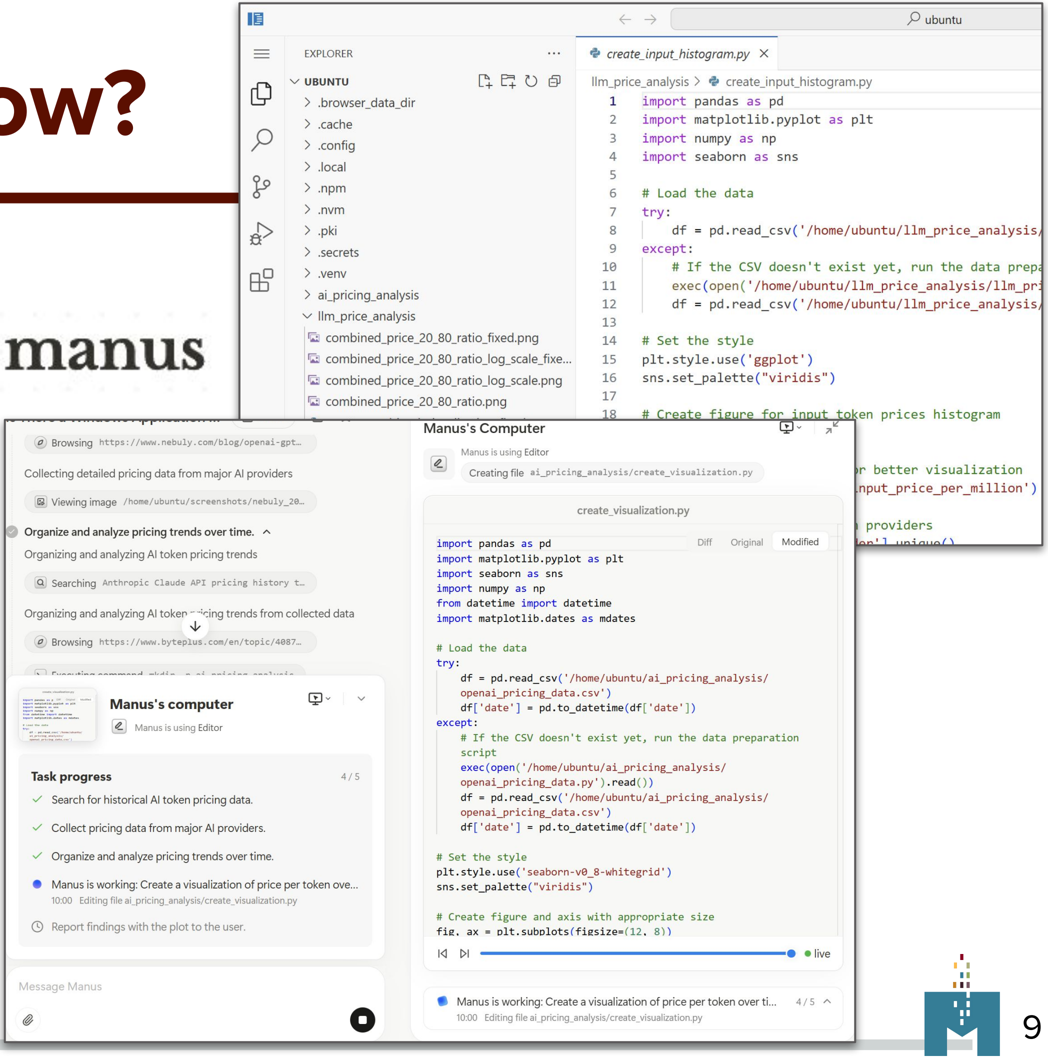
Let the AI use the whole computer…

manus

- Runs an Ubuntu container
- Creates and executes tasks
- A lot of tasks solved using scripting/coding (CodeAct).
- You can jump in, make edits, (re)use code.
- Can run it locally (eg. OpenManus)
- Learns preferences

What to use it to:

- Test your tutorial.
- Test your services/interfaces.
- Do data analysis and draw a conclusion.
- Automate report generation or any repetitive task


ubuntu
EXPLORER
UBUNTU
.browser_data_dir
.cache
.config
.local
.npm
.nvm
.pki
.secrets
.venv
ai_pricing_analysis
llm_price_analysis
combined_price_20_80_ratio_fixed.png
combined_price_20_80_ratio_log_scale_fixe...
combined_price_20_80_ratio_log_scale.png
combined_price_20_80_ratio.png
create_input_histogram.py
llm_price_analysis > create_input_histogram.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Load the data
try:
    df = pd.read_csv('/home/ubuntu/llm_price_analysis/
except:
    # If the CSV doesn't exist yet, run the data prepa
    exec(open('/home/ubuntu/llm_price_analysis/llm_pri
    df = pd.read_csv('/home/ubuntu/llm_price_analysis/

# Set the style
plt.style.use('ggplot')
sns.set_palette("viridis")

# Create figure for input token prices histogram
```

Browsing `https://www.nebuly.com/blog/openai-gpt…`

Collecting detailed pricing data from major AI providers

Viewing image `/home/ubuntu/screenshots/nebuly_20…`

Organize and analyze pricing trends over time.

Organizing and analyzing AI token pricing trends

Searching `Anthropic Claude API pricing history t…`

Organizing and analyzing AI token pricing trends from collected data

Browsing `https://www.byteplus.com/en/topic/4087…`

Manus's computer

Manus is using Editor

Task progress 4 / 5

- ✓ Search for historical AI token pricing data.
- ✓ Collect pricing data from major AI providers.
- ✓ Organize and analyze pricing trends over time.
- ● Manus is working: Create a visualization of price per token ove…
  10:00 Editing file ai_pricing_analysis/create_visualization.py
- Report findings with the plot to the user.

Manus's Computer

Manus is using Editor

Creating file `ai_pricing_analysis/create_visualization.py`

create_visualization.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from datetime import datetime
import matplotlib.dates as mdates

# Load the data
try:
    df = pd.read_csv('/home/ubuntu/ai_pricing_analysis/
    openai_pricing_data.csv')
    df['date'] = pd.to_datetime(df['date'])
except:
    # If the CSV doesn't exist yet, run the data preparation
    script
    exec(open('/home/ubuntu/ai_pricing_analysis/
    openai_pricing_data.py').read())
    df = pd.read_csv('/home/ubuntu/ai_pricing_analysis/
    openai_pricing_data.csv')
    df['date'] = pd.to_datetime(df['date'])

# Set the style
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette("viridis")

# Create figure and axis with appropriate size
fig, ax = plt.subplots(figsize=(12, 8))
```

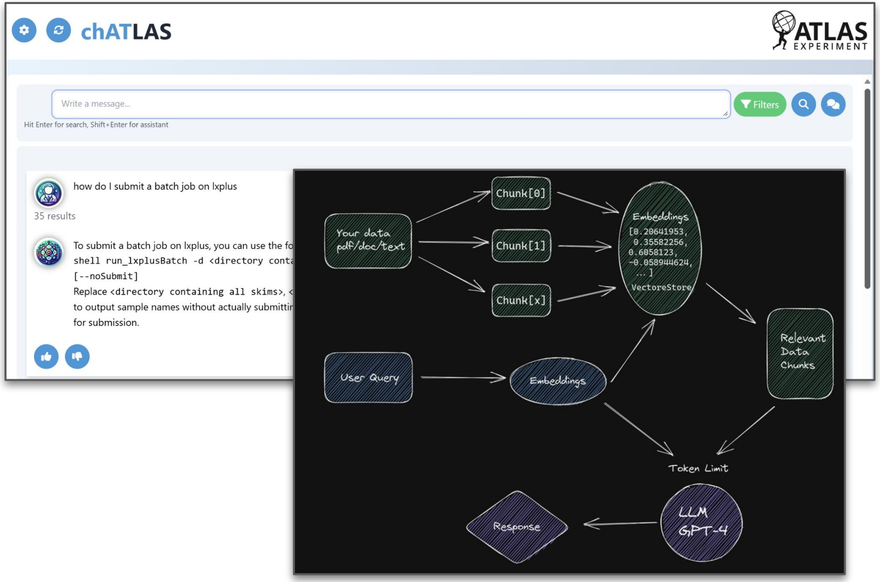
Manus is working: Create a visualization of price per token over ti… 4 / 5
10:00 Editing file ai_pricing_analysis/create_visualization.py

Agents limitations

- Out-of-box, most won’t do things that could be used nefariously: send emails, login to nodes or web sites, impersonate you.
- Access to our data and frameworks functions has to be implemented by us.
- From time to time they make mistakes.

QUESTION 2. Do we need a disclaimer? eg. This result/page was fully/partially AI generated

Existing agents



how do I submit a batch job on lxplus

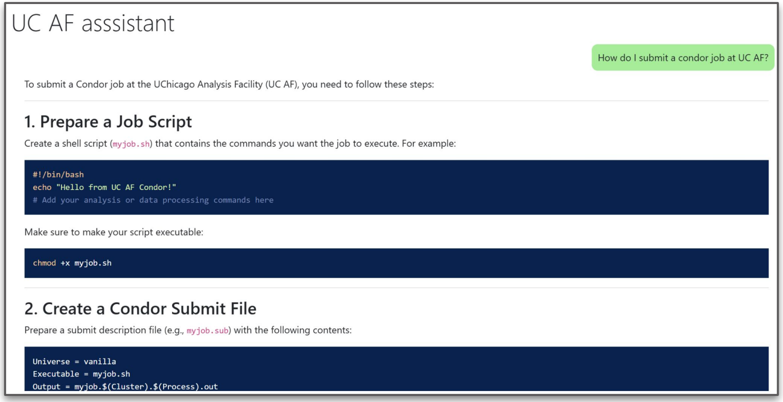
35 results

To submit a batch job on lxplus, you can use the fo
shell run_lxplusBatch -d <directory conta
[--noSubmit]
Replace <directory containing all skims>, <
to output sample names without actually submittir
for submission.

chATLAS has TWiki and CDS, eGroups, indico, scrapped, converted, chunked, embedded.

Deployed on CERN's PaaS.

Spare time effort



UC AF asssistant

How do I submit a condor job at UC AF?

To submit a Condor job at the UChicago Analysis Facility (UC AF), you need to follow these steps:

1. Prepare a Job Script

Create a shell script (`myjob.sh`) that contains the commands you want the job to execute. For example:

```
#!/bin/bash
echo "Hello from UC AF Condor!"
# Add your analysis or data processing commands here
```

Make sure to make your script executable:

```
chmod +x myjob.sh
```

2. Create a Condor Submit File

Prepare a submit description file (e.g., `myjob.sub`) with the following contents:

```
Universe = vanilla
Executable = myjob.sh
Output = myjob.$(Cluster).$(Process).out
```

UChicago AF assistant is based on OpenAI.

Documentation in a cloud vector store.

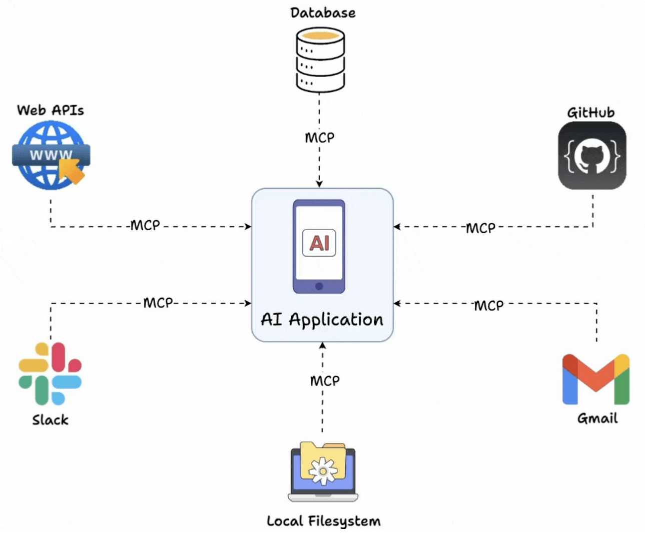
It can access Elasticsearch cluster indices with monitoring and accounting data.

It can logging into UC Analysis Facility and do same things as any other user.

Runs at UChicago AF k8s cluster.

Two sister assistant projects for: Soteria, WFMS

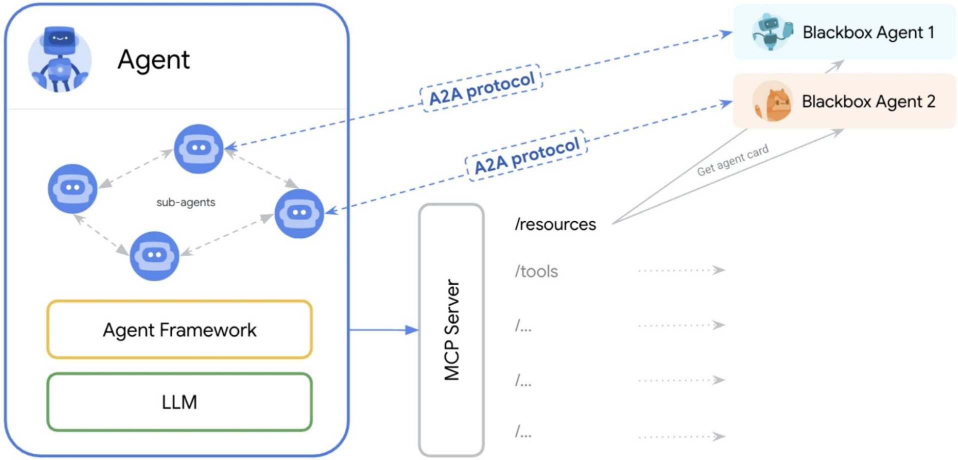
Model Context Protocol



MCP is an open standard that connects Agents to data sources.

- Widely supported: MCP Servers catalogue
- TypeScript, Python, C#, and other MCP SDKs already exist.
- Stdio, SSE or streamable HTTP transport.
- Supports sessions and stateless.
- Authentication & Authorization not yet fixed but will be OAuth 2.1, with third-party IdPs to be default and IdP discovery by WWW-Authenticate. Some services we have (eg. Elasticsearch) will still require user/pass.
- Discovery still under discussion (mcp:// ?). Windows 11 preview now comes with an MCP registry.

A2A



Agent-to-Agent communication protocol.

Google proposed, getting accepted.

Key ideas:

- Agents run independently, no shared state or central control.
- Regular HTTP, SSE, JSON-RPC
- Built-in OIDC. Discovery .well-known/agent.json
- Real-time progress, state tracking, human-in-the-loop support.
- Multimodal
- Geared for B2B, scalable, stateful tasks.

Workflow automation

A lot of tools to ease agent reasoning and orchestration. All provide multi-agent collaboration.

LangChain is designed for building AI agents that integrate LLMs with external tools, memory, and reasoning capabilities. Open source, free except for LangSmith (debugging tool).

- **Multi-step reasoning**: Allows agents to break down complex tasks into smaller steps.
- **Tool integration**: Supports APIs, databases, and external knowledge sources.
- **Memory management**: Helps agents retain context across interactions.
- **Modular design**: Easily customizable for different applications.

Google Agent Development Kit (ADK) a brand-new framework. Open source, free.

- **Rich model ecosystem**: Works with various AI models, including Google's Gemini.
- **Built-in evaluation**: Provides systematic assessment of agent performance.
- **Streaming capabilities**: Enables real-time bidirectional audio and video interactions.

n8n No code workflow automation with AI enhancements.

Way forward - Option 1 - *Laissez-faire*

- AI coding tools will improve our code and better systems are easier to support.
    - Less bugs
    - Better documentation
    - Faster development cycle
    - Much cheaper to reimplement from scratch and move to modern frameworks.
- Web based AI operators will automate some of manual work.

Pros:
- (Almost) no effort.
- Developer person power can be x3.
- No infrastructure needed.
- We can do more once things stabilize.

Cons:
- Uptake is not guaranteed.
- Web based agents are slow.
- Our frontends are not exactly friendly.

Way forward - Option 2 - *AI for individual systems*

Imagine this chat:

How many files in our dCache older than 1 year?

Single agent would be enough.

There are XXX files older than 1 year, taking YYY TB.

How many connections on dCache xrootd door. How does it compare to BNL dCache?

Single agent and an MCP server

Currently there are X connections to your dCache xrootd door. This is 53% less than on BNL dCache.

Way forward - Option 2 - *AI for individual systems*

Could be a computing site specific (eg. batch), or a system (eg. FTS).

Eg. dCache, HTCondor, … could/should make an agent that monitors it, and/or an MCP that can be used by site's agent or site's admin.

Pros:

- Can be done piecewise
- It would be done by people with expert knowledge.
- Reduced load on people, better uptime/utilization.

Cons:

- Could be duplicating effort if disparate choices are made.
- Could require hardware that wouldn't be efficiently used.
- Can't see the global picture.

QUESTION 3. Let start with HTCondor. Where should we ask for it? :)

Way forward - Option 3 - *A wholesale solution*

Imagine this chat:

What’s the state of my grid job?

Your job was running at XXX, it started at xx:xx. Logs show that it failed accessing the file yyy. I asked a local site dCache and it appears the file is not there. I declared the file missing and resubmitted your job.

This is possible but requires a bunch of individually not very complex agents: Panda, Pilot, dCache, Rucio.

Your job had a high failure rate. Looking at its resource usage it seems memory requested was insufficient. I have submitted a new request with higher requested memory.

This can be done by two agents.

Way forward - Option 3 - *A wholesale solution*

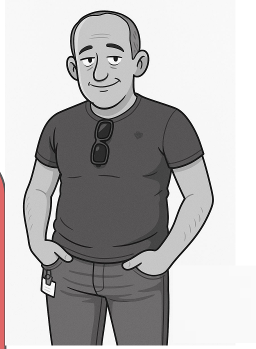
Make and deploy MCP servers for all the systems, exposing most of functionality.

Place them behind OAuth 2.1. Will need a common source of user roles.

Select and deploy a framework to run agents. Create agents.

Create a database of “common knowledge” describing systems and where do they fit.

We started work on this one. Let us now if you want to contribute.



Pros:

- Can see whole picture. The more we expose, more capable AI gets.
- We could retire some of the current monitoring/alarming/alerting systems.

Cons:

- Quite a bit of effort and coordination. Can’t be done in spare time.
- Hardware.
- More potential for an accident.

Way forward - Option 3 - *A wholesale solution*

QUESTION 4. Who should do it. 3 summer students won’t cut it.

QUESTION 5. Experiment specific, WLCG specific, …

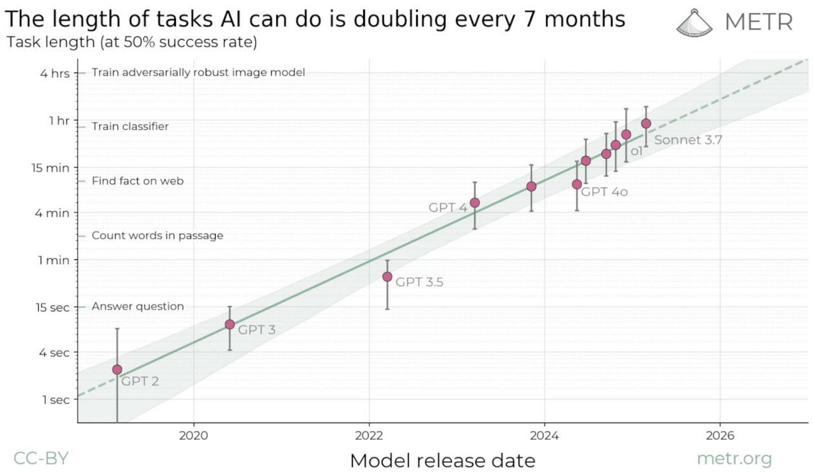
QUESTION 6. Cloud or On premise? Single AI infrastructure?

QUESTION 7. What orchestration framework to use?

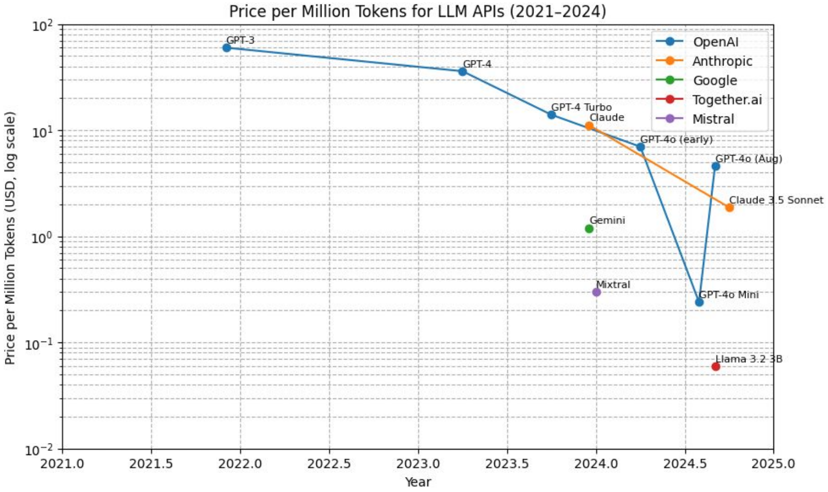
QUESTION 8. Authentication and authorization?

QUESTION 9. How do we do evals?

QUESTION 10. Timeline?

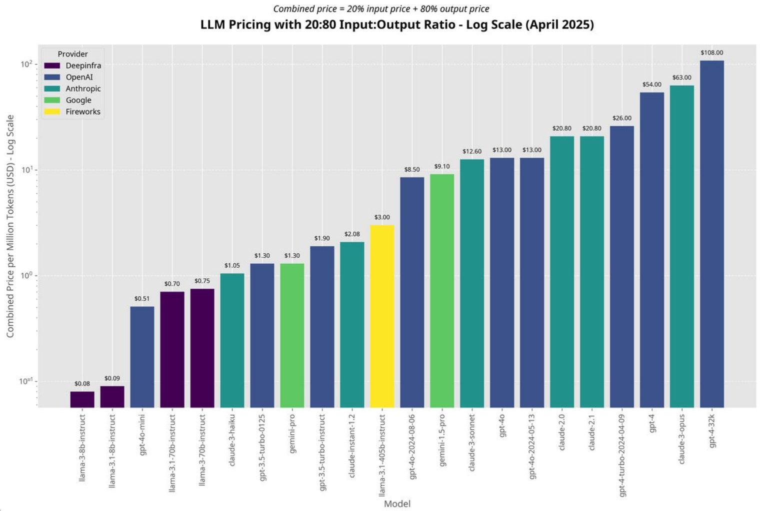


Infrastructure



Cloud

- Prices are falling quickly
- Huge variety of models
- We could pick whatever is the best value for a task at hand
- Everybody promises not to keep or learn from your data. Still some things would have to be anonymized and some things we simply can’t do in cloud.



On premise

- Most of GPUs we have are dedicated to physics, training, grid jobs,...
- We don’t have GPUs dedicated to inference. A100s are too expensive to be occupied 24/7 but used 10% of the time. Old GPUs are too slow for this.
- One could imagine one small dedicated facility shared among experiments.
- Soon we should be able to buy an NVidia DGX Spark for a price of a fancy laptop
    - Grace Blackwell GB10
    - 1PFlop of FP4
    - 128 GB of unified GDDR5x - fits 200B models.
    - 30 ARM cores

Risks and Safety

Web search tools - how to prevent prompt injection?

“Computer Use” tool present unique challenges

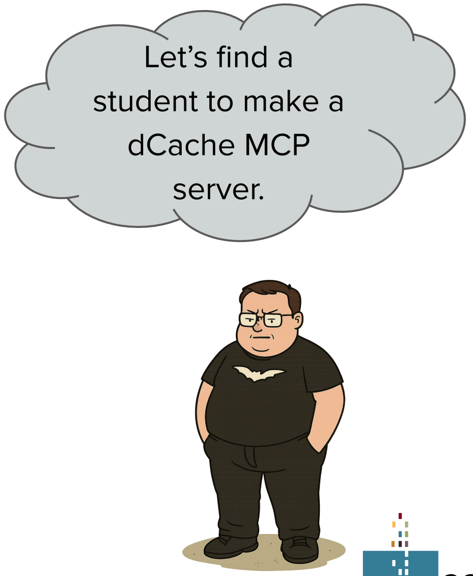
For start:

- We need human-in-the-loop for important stuff
- Block and allow lists
- Save interactions and user IDs.

Conclusion

- AI is not a magic bullet. Can’t fix broken systems.
- But it could help fixing, monitoring, and using them.
- It could help in understanding system specific issues and issues coming from interactions between systems.
- It could help running systems. First as a “shadow” admin, then as a full admin (with a human signoff for big destructive operations).
- **How much we gain from using AI depends on how many of our systems we expose.**

QUESTION 11. How much are we willing to expose?



Gemini

Meta

GitHub
Copilot

deepseek

Backup slides

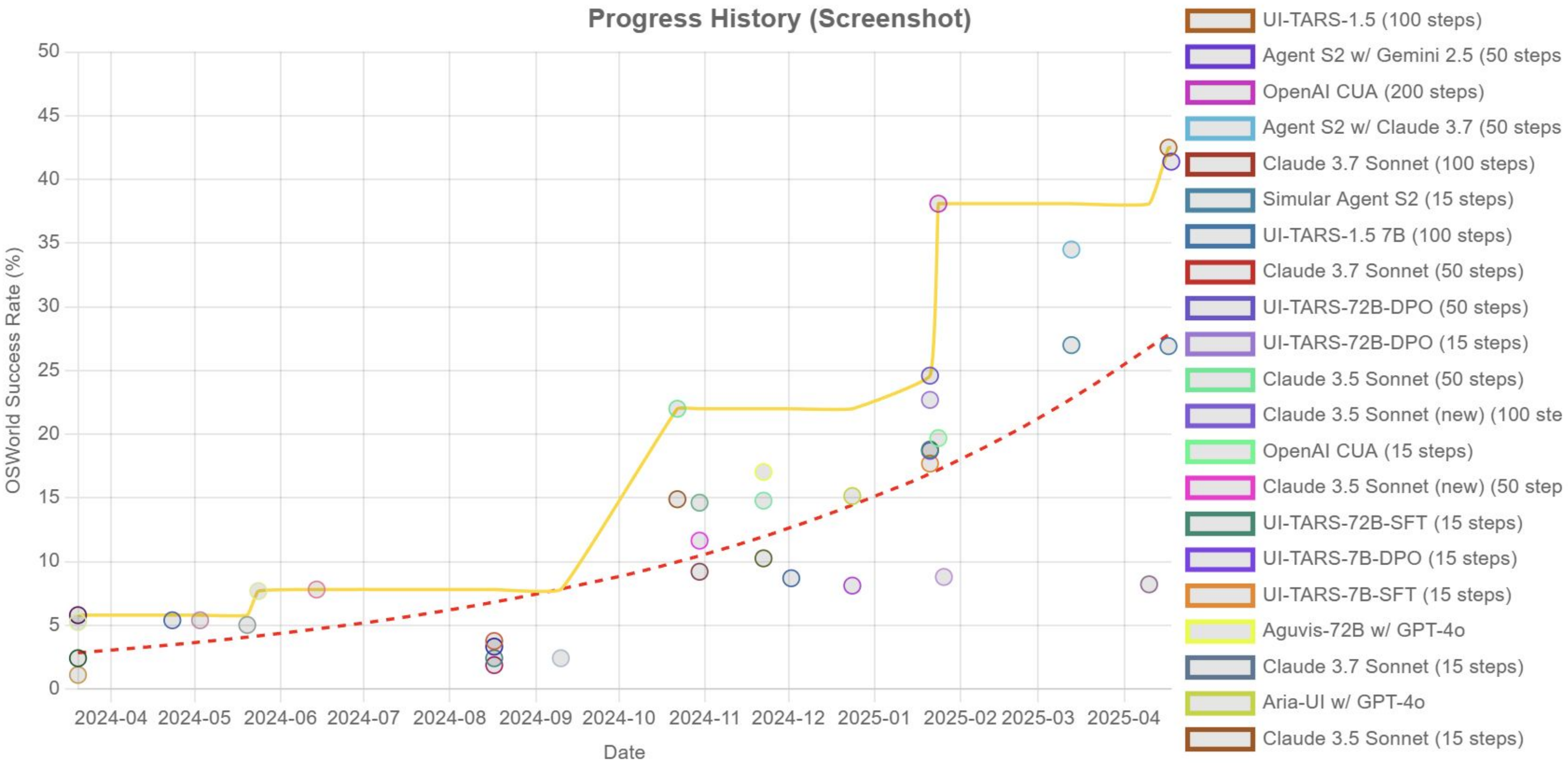
manus

ANTHROP\C

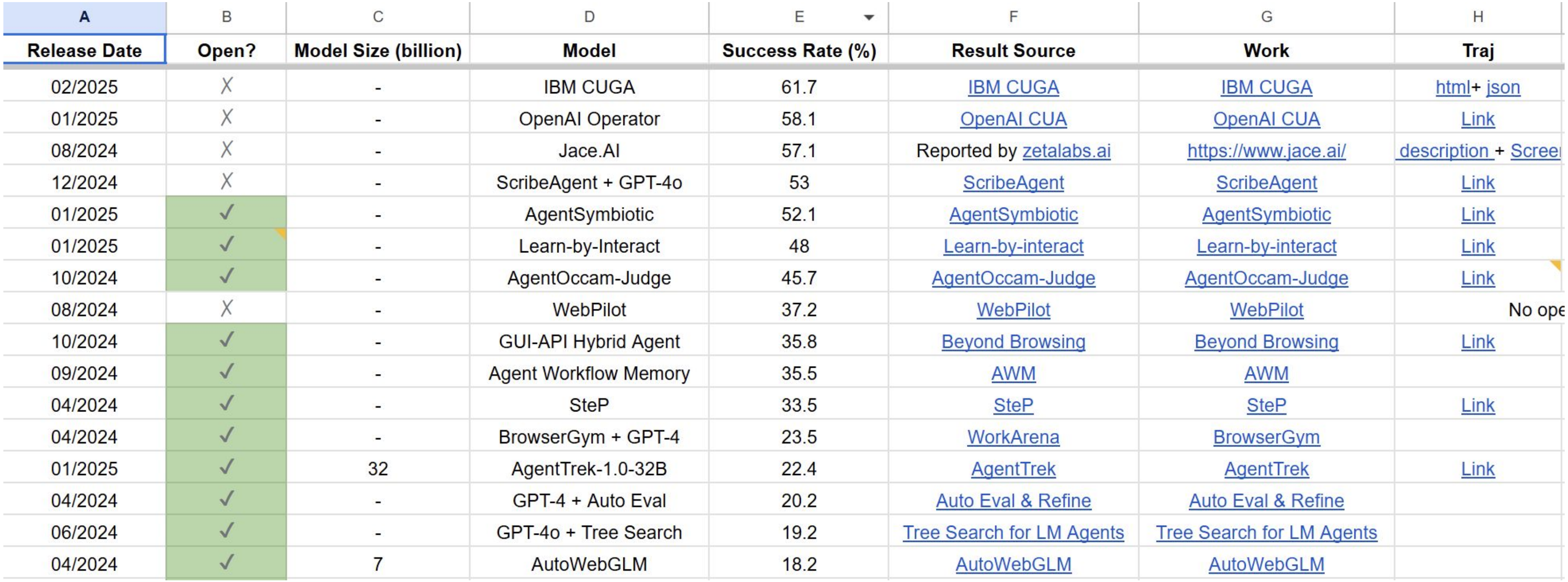
OpenAI

Alibaba

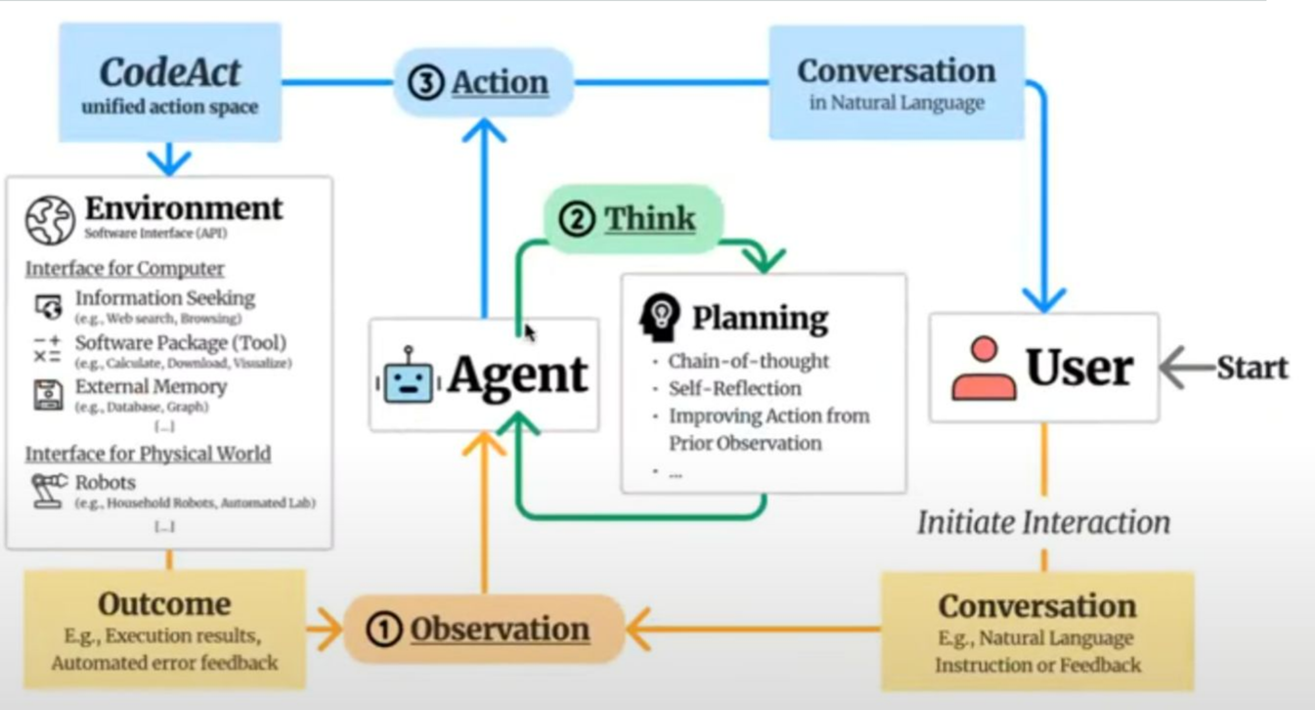
OSWorld


Progress History (Screenshot)
OSWorld Success Rate (%)
50
45
40
35
30
25
20
15
10
5
0
2024-04
2024-05
2024-06
2024-07
2024-08
2024-09
2024-10
2024-11
2024-12
2025-01
2025-02
2025-03
2025-04
Date
UI-TARS-1.5 (100 steps)
Agent S2 w/ Gemini 2.5 (50 steps
OpenAI CUA (200 steps)
Agent S2 w/ Claude 3.7 (50 steps
Claude 3.7 Sonnet (100 steps)
Simular Agent S2 (15 steps)
UI-TARS-1.5 7B (100 steps)
Claude 3.7 Sonnet (50 steps)
UI-TARS-72B-DPO (50 steps)
UI-TARS-72B-DPO (15 steps)
Claude 3.5 Sonnet (50 steps)
Claude 3.5 Sonnet (new) (100 ste
OpenAI CUA (15 steps)
Claude 3.5 Sonnet (new) (50 step
UI-TARS-72B-SFT (15 steps)
UI-TARS-7B-DPO (15 steps)
UI-TARS-7B-SFT (15 steps)
Aguvis-72B w/ GPT-4o
Claude 3.7 Sonnet (15 steps)
Aria-UI w/ GPT-4o
Claude 3.5 Sonnet (15 steps)

WebArena



Release Date	Open?	Model Size (billion)	Model	Success Rate (%)	Result Source	Work	Traj
02/2025	X	-	IBM CUGA	61.7	IBM CUGA	IBM CUGA	html+ json
01/2025	X	-	OpenAI Operator	58.1	OpenAI CUA	OpenAI CUA	Link
08/2024	X	-	Jace.AI	57.1	Reported by zetalabs.ai	https://www.jace.ai/	description + Scree
12/2024	X	-	ScribeAgent + GPT-4o	53	ScribeAgent	ScribeAgent	Link
01/2025	✓	-	AgentSymbiotic	52.1	AgentSymbiotic	AgentSymbiotic	Link
01/2025	✓	-	Learn-by-Interact	48	Learn-by-interact	Learn-by-interact	Link
10/2024	✓	-	AgentOccam-Judge	45.7	AgentOccam-Judge	AgentOccam-Judge	Link
08/2024	X	-	WebPilot	37.2	WebPilot	WebPilot	No ope
10/2024	✓	-	GUI-API Hybrid Agent	35.8	Beyond Browsing	Beyond Browsing	Link
09/2024	✓	-	Agent Workflow Memory	35.5	AWM	AWM	
04/2024	✓	-	SteP	33.5	SteP	SteP	Link
04/2024	✓	-	BrowserGym + GPT-4	23.5	WorkArena	BrowserGym	
01/2025	✓	32	AgentTrek-1.0-32B	22.4	AgentTrek	AgentTrek	Link
04/2024	✓	-	GPT-4 + Auto Eval	20.2	Auto Eval & Refine	Auto Eval & Refine	
06/2024	✓	-	GPT-4o + Tree Search	19.2	Tree Search for LM Agents	Tree Search for LM Agents	
04/2024	✓	7	AutoWebGLM	18.2	AutoWebGLM	AutoWebGLM	



Instead of calling tools (functions) that return json or text, agent generates code that it executes.

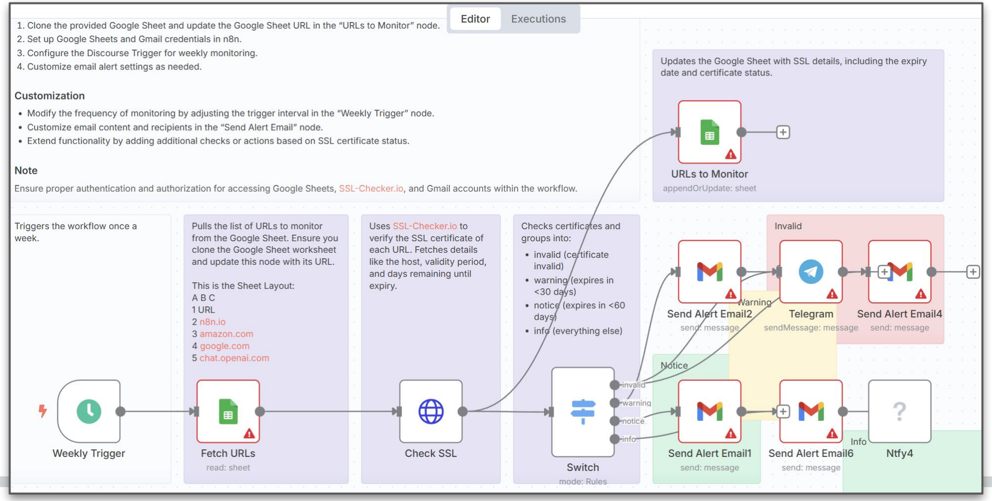
- 10-20% better results. Much faster as there are less iterations.
- First implemented in Manus. Now exists in Langgraph-codeact

Workflow automation

N8N a workflow automation with AI enhancements

n8n

- **No-code/low-code automation**: Users can design AI workflows visually.
- **Flexible AI integrations**: Supports **LLMs** like OpenAI, LangChain, and other AI tools.
- **Multi-agent coordination**: Enables AI agents to collaborate and make decisions.
- **Scalability**: Works for simple automations and complex multi-agent systems.
- **Open-source with commercial use restrictions.**



1. Clone the provided Google Sheet and update the Google Sheet URL in the “URLs to Monitor” node.
2. Set up Google Sheets and Gmail credentials in n8n.
3. Configure the Discourse Trigger for weekly monitoring.
4. Customize email alert settings as needed.

Customization

- Modify the frequency of monitoring by adjusting the trigger interval in the “Weekly Trigger” node.
- Customize email content and recipients in the “Send Alert Email” node.
- Extend functionality by adding additional checks or actions based on SSL certificate status.

Note

Ensure proper authentication and authorization for accessing Google Sheets, SSL-Checker.io, and Gmail accounts within the workflow.

Updates the Google Sheet with SSL details, including the expiry date and certificate status.

Triggers the workflow once a week.

Pulls the list of URLs to monitor from the Google Sheet. Ensure you clone the Google Sheet worksheet and update this node with its URL.

This is the Sheet Layout:
A B C
1 URL
2 n8n.io
3 amazon.com
4 google.com
5 chat.openai.com

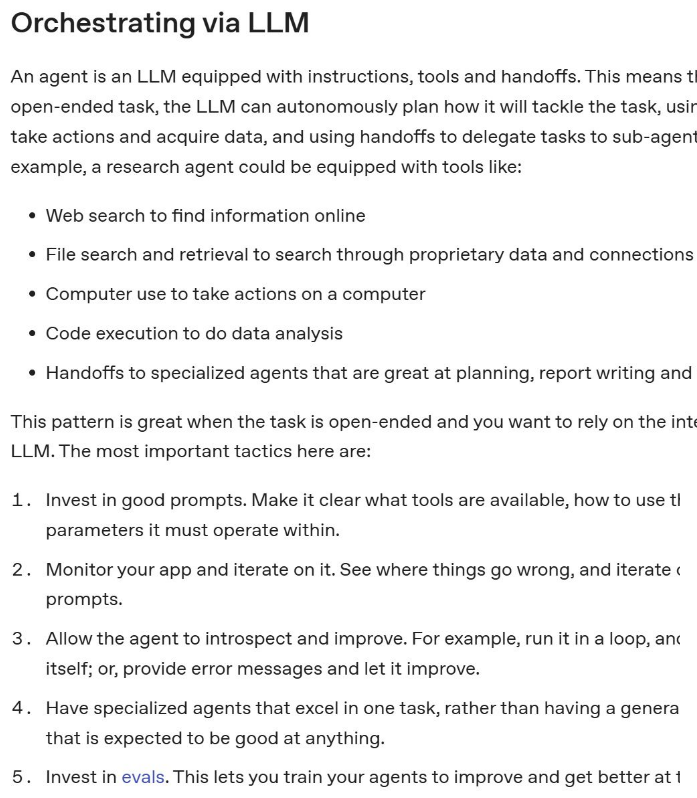
Uses SSL-Checker.io to verify the SSL certificate of each URL. Fetches details like the host, validity period, and days remaining until expiry.

Checks certificates and groups into:

- invalid (certificate invalid)
- warning (expires in <30 days)
- notice (expires in <60 days)
- info (everything else)

SSL expiry alert template

- Streaming - OpenAI Agents SDK
- LangGraph
- Microsoft Copilot Studio/Azure AI Foundry. Will come with MCP server registry. Agentic web.
- AutoGen .4 is being replaced by Semantic Kernel
- Google Vertex AI engine with ADK.



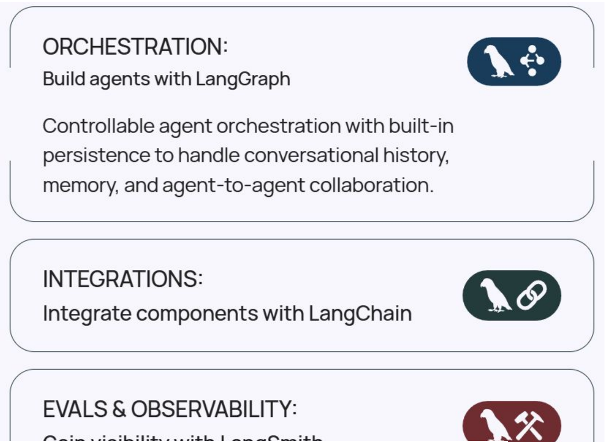
Orchestrating via LLM

An agent is an LLM equipped with instructions, tools and handoffs. This means that given an open-ended task, the LLM can autonomously plan how it will tackle the task, using tools to take actions and acquire data, and using handoffs to delegate tasks to sub-agents. For example, a research agent could be equipped with tools like:

- Web search to find information online
- File search and retrieval to search through proprietary data and connections
- Computer use to take actions on a computer
- Code execution to do data analysis
- Handoffs to specialized agents that are great at planning, report writing and more.

This pattern is great when the task is open-ended and you want to rely on the intelligence of an LLM. The most important tactics here are:

1. Invest in good prompts. Make it clear what tools are available, how to use t parameters it must operate within.
2. Monitor your app and iterate on it. See where things go wrong, and iterate prompts.
3. Allow the agent to introspect and improve. For example, run it in a loop, an itself; or, provide error messages and let it improve.
4. Have specialized agents that excel in one task, rather than having a genera that is expected to be good at anything.
5. Invest in evals. This lets you train your agents to improve and get better at

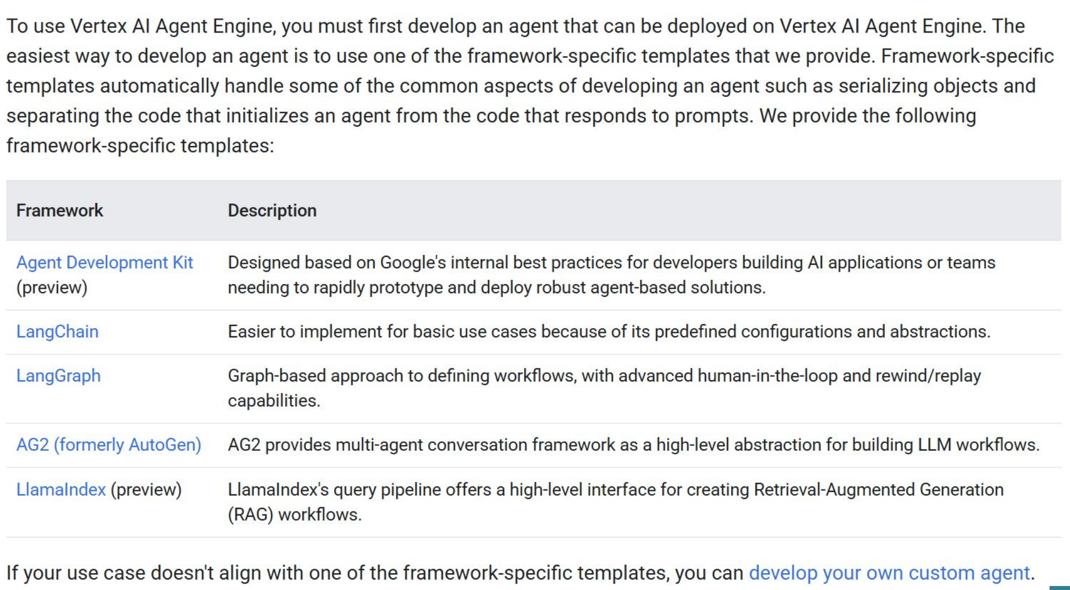


ORCHESTRATION:
Build agents with LangGraph

Controllable agent orchestration with built-in persistence to handle conversational history, memory, and agent-to-agent collaboration.

INTEGRATIONS:
Integrate components with LangChain

EVALS & OBSERVABILITY:



To use Vertex AI Agent Engine, you must first develop an agent that can be deployed on Vertex AI Agent Engine. The easiest way to develop an agent is to use one of the framework-specific templates that we provide. Framework-specific templates automatically handle some of the common aspects of developing an agent such as serializing objects and separating the code that initializes an agent from the code that responds to prompts. We provide the following framework-specific templates:

Framework	Description
Agent Development Kit (preview)	Designed based on Google's internal best practices for developers building AI applications or teams needing to rapidly prototype and deploy robust agent-based solutions.
LangChain	Easier to implement for basic use cases because of its predefined configurations and abstractions.
LangGraph	Graph-based approach to defining workflows, with advanced human-in-the-loop and rewind/replay capabilities.
AG2 (formerly AutoGen)	AG2 provides multi-agent conversation framework as a high-level abstraction for building LLM workflows.
LlamaIndex (preview)	LlamaIndex's query pipeline offers a high-level interface for creating Retrieval-Augmented Generation (RAG) workflows.

If your use case doesn't align with one of the framework-specific templates, you can develop your own custom agent.

Backup slides
UC AF assistant

Idea

We have three analysis facilities in US ATLAS: University of Chicago, BNL, and SLAC.

We offer a lot of services: login/interactive work, Jupyter, Batch on a lot of hardware (CPU, GPU, storage).

We are supporting more and more users.

Create a chatbot to help users who don't like reading the documentation, discourse forums eg.

> *How do I send a condor job? What nodes I can log in? How much scratch space do I get?*

Make it able to answer runtime questions eg.

> *What are currently available GPUs, what state are my submitted jobs in.*

Make it a able to answer accounting questions that are relevant to Analysis Facility managers eg.

> *Make a pie chart showing a number of unique users that logged in at different analysis facilities.*

Technology choice

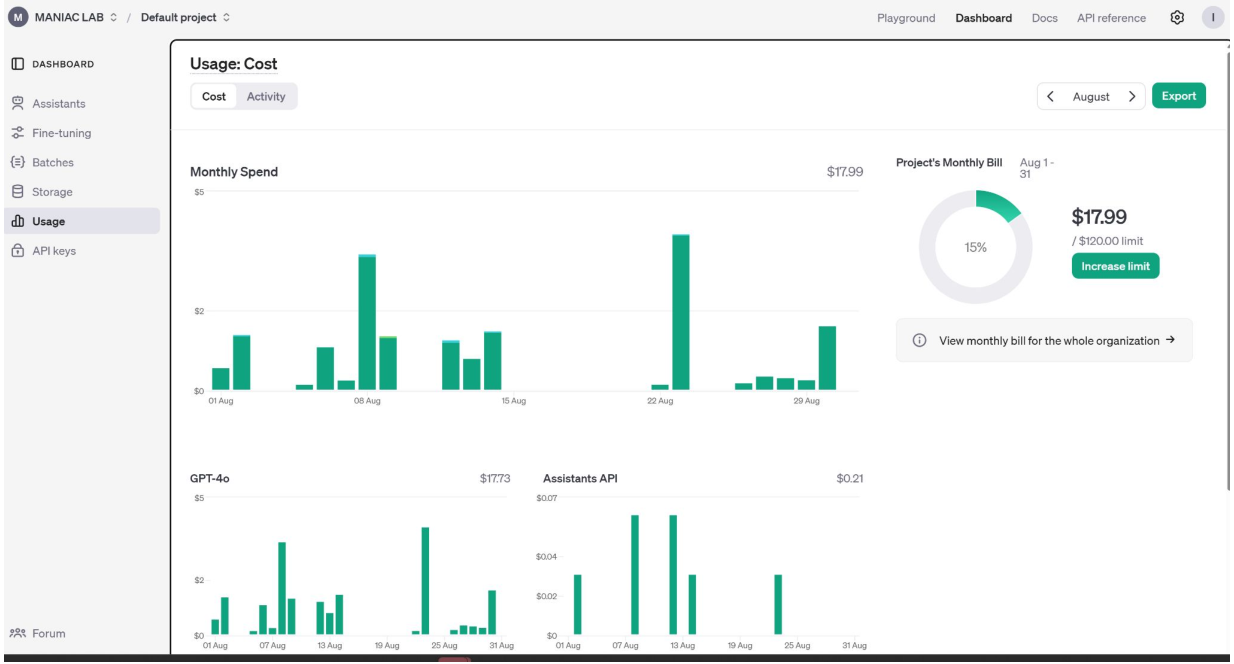
These days there are options. We went with the easiest - OpenAI.

Using GPT-4o.

Cost is not prohibitive.

The only data stored in OpenAI is our documentation.

They don't learn from our prompts, but we do.



Frontend

- A node.js application with Express and Pug.
  - Completely async, multithreaded, event driven.
- We store all prompts/requests in Elasticsearch.
- Access limited to users logged in to AF.
  - Assistant already knows who you are.

Answering based on documentation

Sounded trivial… just upload .md files from our mkdoc site add it to vector store and that's it.

- Then it appeared some of the data was in HTML (used a web converter to transform it to md).
- But we had an another set of documents with wider scope and hard to limit to only UC AF.
- Have to keep track of which imported md file can be accessed at which URL so that references in responses would be correct.
- Even a brief testing revealed deficiencies in the documentation that were fixed by Lincoln B.

We use Discourse forums to provide user support. All of the relevant data (topic, date, state, all replies, etc.) have been pulled out of the database by Louis Pelosi. We will try to use it to improve answers. Many potential issues: some threads are obsolete, some issues were temporary, some problems were user caused.

Answering based on Elasticsearch data

aka RAG (Retrieval Augmented Generation)

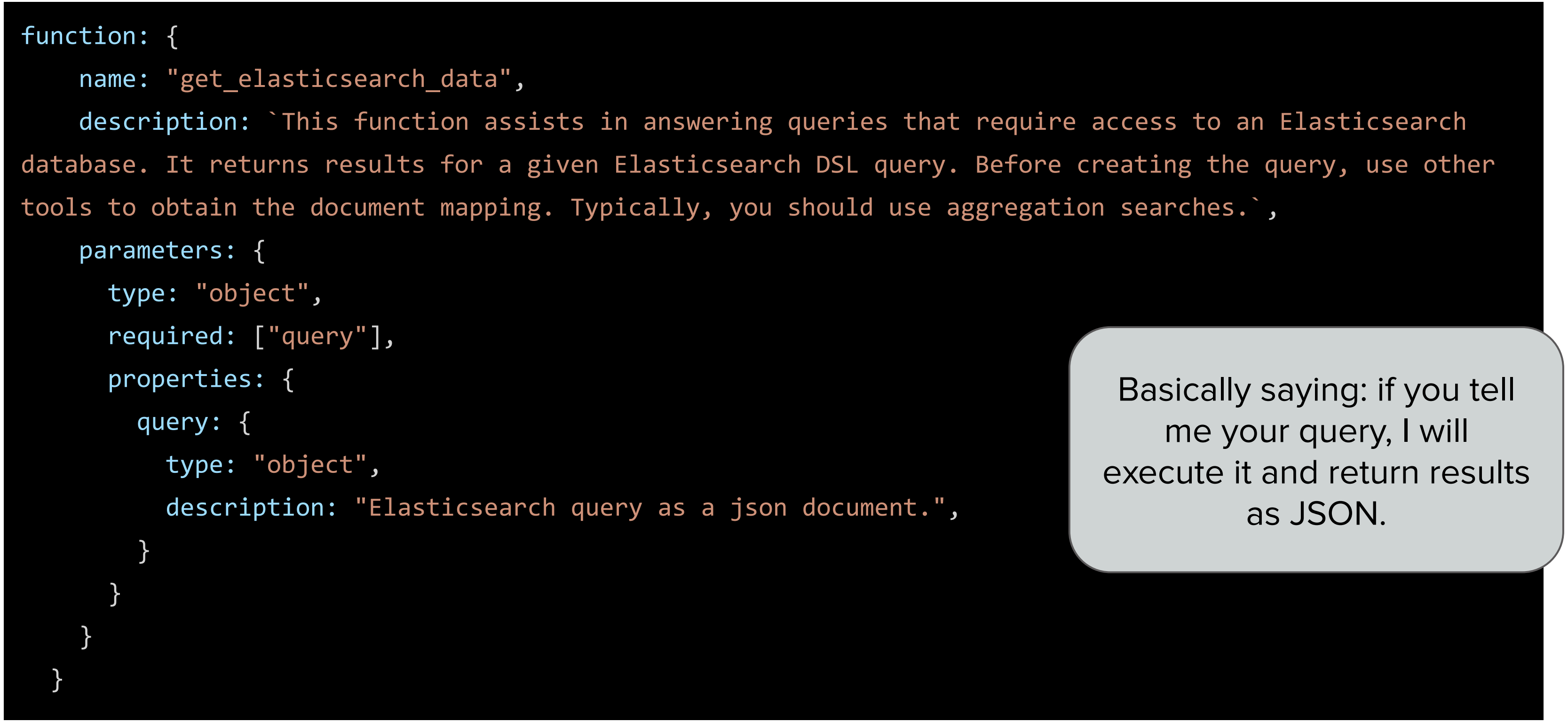
We don’t ship our data to OpenAi (it would be way too costly).

We provide it with functions it can call if it needs data from the EL.

We explain what are parameters and what the function will return.

We had to add some scripted variables, change names to make them selfdescriptive.

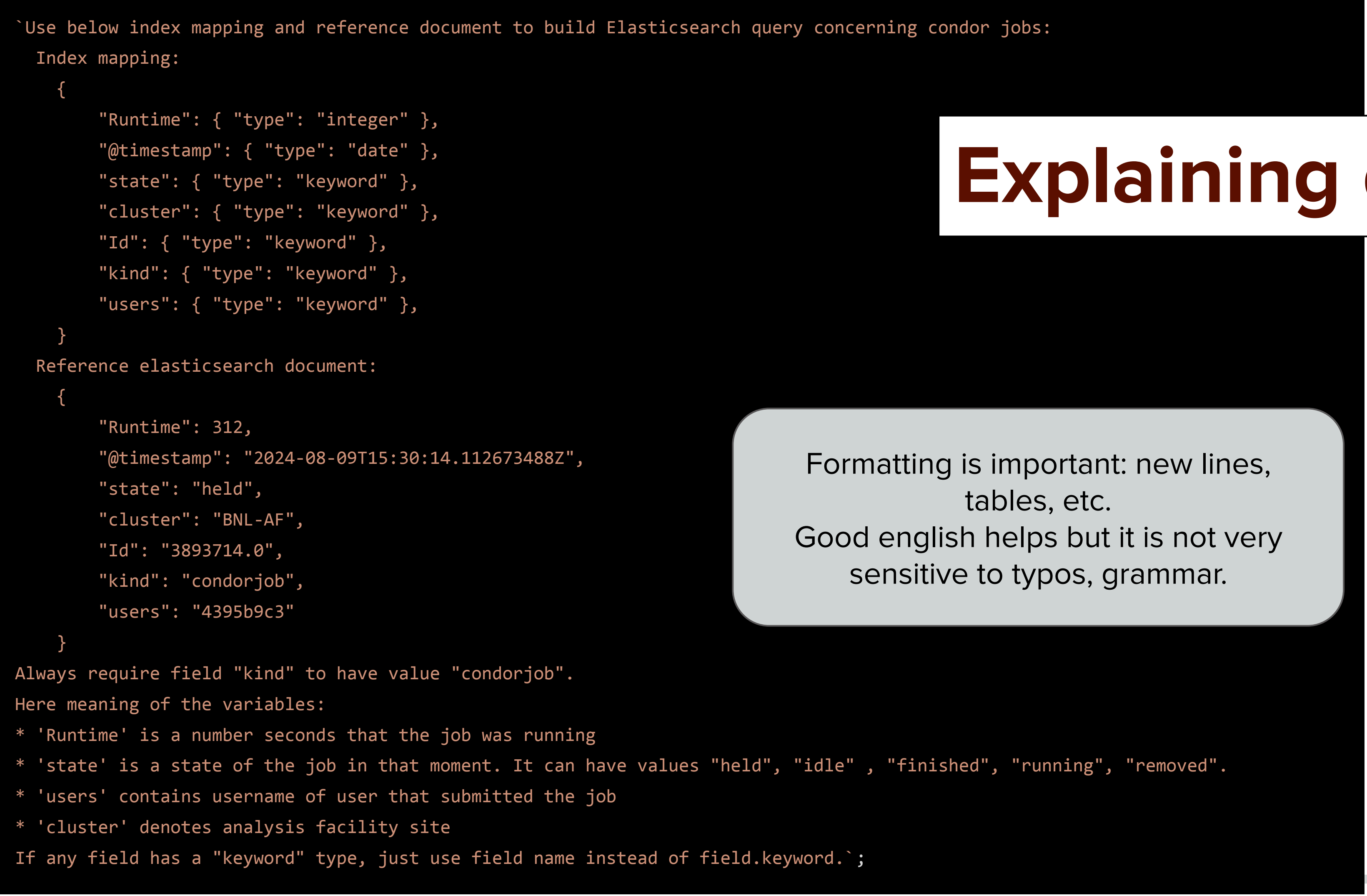
Example function description I



```
function: {
    name: "get_elasticsearch_data",
    description: `This function assists in answering queries that require access to an Elasticsearch
database. It returns results for a given Elasticsearch DSL query. Before creating the query, use other
tools to obtain the document mapping. Typically, you should use aggregation searches.`,
    parameters: {
      type: "object",
      required: ["query"],
      properties: {
        query: {
          type: "object",
          description: "Elasticsearch query as a json document.",
        }
      }
    }
  }
```

Basically saying: if you tell me your query, I will execute it and return results as JSON.

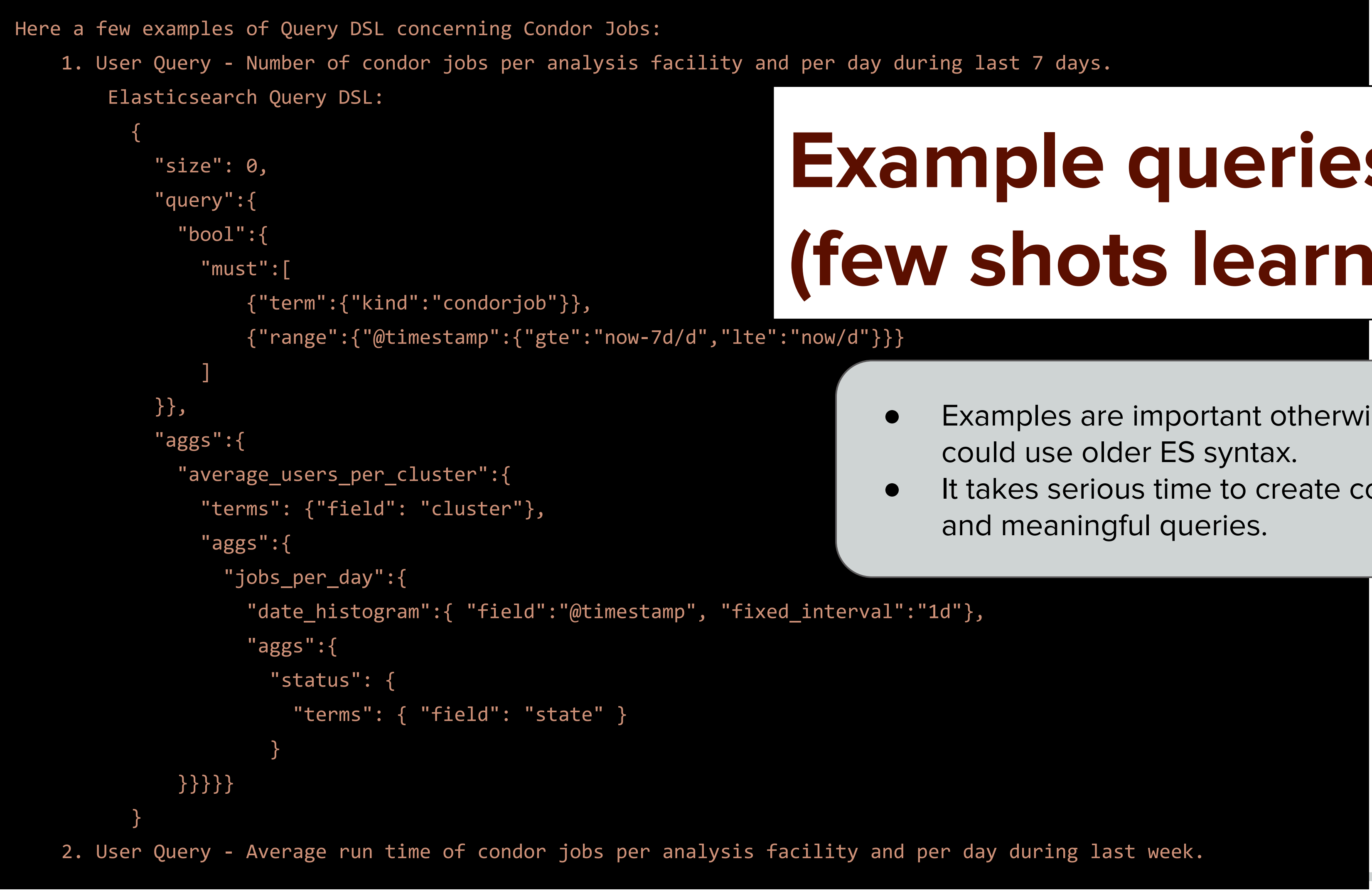
Explaining data



```
`Use below index mapping and reference document to build Elasticsearch query concerning condor jobs:
  Index mapping:
    {
        "Runtime": { "type": "integer" },
        "@timestamp": { "type": "date" },
        "state": { "type": "keyword" },
        "cluster": { "type": "keyword" },
        "Id": { "type": "keyword" },
        "kind": { "type": "keyword" },
        "users": { "type": "keyword" },
    }
  Reference elasticsearch document:
    {
        "Runtime": 312,
        "@timestamp": "2024-08-09T15:30:14.112673488Z",
        "state": "held",
        "cluster": "BNL-AF",
        "Id": "3893714.0",
        "kind": "condorjob",
        "users": "4395b9c3"
    }
Always require field "kind" to have value "condorjob".
Here meaning of the variables:
* 'Runtime' is a number seconds that the job was running
* 'state' is a state of the job in that moment. It can have values "held", "idle" , "finished", "running", "removed".
* 'users' contains username of user that submitted the job
* 'cluster' denotes analysis facility site
If any field has a "keyword" type, just use field name instead of field.keyword.`;
```

Formatting is important: new lines, tables, etc.
Good english helps but it is not very sensitive to typos, grammar.

Example queries (few shots learning)



```
Here a few examples of Query DSL concerning Condor Jobs:
  1. User Query - Number of condor jobs per analysis facility and per day during last 7 days.
    Elasticsearch Query DSL:
     {
      "size": 0,
      "query":{
       "bool":{
        "must":[
          {"term":{"kind":"condorjob"}},
          {"range":{"@timestamp":{"gte":"now-7d/d","lte":"now/d"}}}
        ]
      }},
      "aggs":{
       "average_users_per_cluster":{
        "terms": {"field": "cluster"},
        "aggs":{
         "jobs_per_day":{
          "date_histogram":{ "field":"@timestamp", "fixed_interval":"1d"},
          "aggs":{
           "status": {
            "terms": { "field": "state" }
           }
       }}}}}
     }
  2. User Query - Average run time of condor jobs per analysis facility and per day during last week.
```

- Examples are important otherwise it could use older ES syntax.
- It takes serious time to create correct and meaningful queries.

Function calls

OpenAI events tells us what function it wants called and with what parameters. We evaluate it and return results to OpenAI. OpenAI interprets the result.

It frequently asks for simultaneous execution of several functions. This significantly improves response time.

A user session creates a single OpenAI thread and the thread is following user over time and tabs. Some functions will be called only once per user (eg. function explaining ES index), so subsequent queries are slightly faster.

Development experience

Application is relatively straightforward to make.

Takes some experience to make it optimal.

RAG part is very different from regular programming:

- Can't really automatize testing as responses even when correct are always a bit different.
- LLM will hide a lot of errors in code:
  - phrase an answer to be technically correct
  - will try multiple times in different ways until it gets result
  - give vague approximate answers
- Once RAG code is debugged, it is amazingly reliable. We haven't noticed any hallucinations. Sometimes it is a bit overconfident and just look at query examples and not at all at document examples.

Development experience II

We did not spend time in trying to limit nefarious prompts

- That should be on OpenAI
- We have records of all prompts/responses with user names.

Data discovery

- Users should have an idea of what kind of data we have so they know what they can ask.
- Will they try prompt like: “What data do you know about?”

Correct question phrasing

- Even users that know data inside out will frequently ask questions that have no sense or can not be answered with with the data we have.

  Eg. “Who is the biggest UC AF user?”

  - Is a person with 1000 condor job a bigger user than someone with 30 jupyter notebooks or someone logged in 24/7?
  - Over what period?
  - Most CPU spent? What about GPU usage?

Future - AF assistant

- Now it is open for users. Learn from questions and responses:
    - What and why it does not answers correctly
    - What are we missing in our documentation
    - Look for signs of abuse
- Get more experience and check scaling to more complex datasets.
    - Make it more robust against OpenAI issues eg.

```
APIError: The server had an error processing your request. Sorry about that! You can retry your request, or contact us
through our help center at help.openai.com if you keep seeing this error. (Please include the request ID
req_c779a1a154c805c3151598cb22e99ab0 in your email.)
```

- Check effects of fine-tuning

Future - next applications

- Soteria project - answer questions about docker images, CVEs, etc. (chatbot)
- PhysLite analyzer
  - Answers prompts like:

    Make a plot of Higgs invariant mass where Higgs decays into four leptons based on period X datasets.
  - Teach it about variables in physlite files.
  - Make a query language (or reuse func_adl)
  - Make an infrastructure that will extremely quickly extract the data (something akin to ServiceX).

Backup slides

Lower level options

Options

1. Fix original issues.
2. A band-aid AI.
   - Helps in detecting, understanding issues
   - Does not solve or mitigate
3. Medium level AI.

   Have it help WFMS in brokering tasks, moving datasets around, etc.

4. High level AI

   AI interprets what a physicist want, creates queries, runs jobs/filters, returns to user a small chunk of data to be locally analysed.

Options - Fix original issues

An ideal solution.
If that’s hard, then the system is probably over-engineered, complex, or grew “organically”.

Cons:

- A very significant effort.
- Big changes that we probably don’t have a manpower or stomach for.

Options - band-aid AI

1. Is my site running smoothly?
2. Are there any problems with my site?
3. Are my jobs running correctly?
4. What's the status of this task?

Create an AI assistant that can lookup panda jobs and tasks information in ES, interpret results.

Can be done quickly (1-2 months) and cheaply using RAG approach

- We already have all the data.
- Create web frontend, create an assistant
- Make functions that assistant can use to mine the data
- Assistant in the cloud, no special infrastructure needed

Cons:

- Users might be completely unaware of its capabilities and simply not use it.

Options - band-aid AI

5. What problems are associated
with this task?

- Have all logging info from panda, rucio, FTS, jobs streamed through a fancy filtering:
  - Noise reduction
  - Time synchronization
  - Enrichment, error code mapping
- Store it in Elasticsearch in vector form
- Continuously run anomaly detection (eg. isolation tree)
- Continuously run root cause analysis (LLM)
- Run issue prioritisation, tagging (LLM)

Options - band-aid AI

Cons:

- Effort to get ALL the logs. Effort to digest it.
- More hardware needed for ES.
- Large effort to make and custom tune a model. Quite a few GPUs for quite some time.
- Uncertain outcome.
- Running it would require special on-premise hardware (eg. NVidia DIGITS)
- It would be one more system needing a long-term support.

Options - medium level AI

Have AI agents helping running our systems.

- Each agent would have full access its system eg. run all Rucio commands - from looking up replicas, creating and deleting rules, etc. eg. submit and cancel tasks
- Have continuous access to system state and actions.
- Fine tune different agent for each system.
- Agents are independent but could communicate with each other.
- Run on-premise .

Options - medium level AI

Have AI agents helping running our systems.

Cons:

- While it is easy to make an agent run as a user or even a superuser, probably systems don’t expose its state well enough.
- There is a possibility that things get out of control.
- Quite a bit of effort to develop.
- Large hardware requirements.
- Uncertain outcome.

Options - high level AI

AI interprets what a physicist want, creates queries, runs jobs/filters, returns to user a small chunk of data to be locally analysed.

- Have a fast, distributed system to filter/augment the data and return to users exactly what they need, where they need it, in the format they want. Something like ServiceX/Y.
- Have AI learn the language and enable it to create queries by itself.

Cons:

- Need to make Filtering work as advertised. Need it distributed. Need the language correctly and completely described.
- Teaching an LLM how to use it (relatively simple).
- User adoption can be shaky.