Gmail **Amos Amissah <theonlyamos@gmail.com>**

A Practical Introduction to Agentic AI[Guest] [Agents]
1 message

Devansh from Artificial Intelligence Made Simple <artificialintelligencemadesimple@substack.com>
Reply-To: Devansh from Artificial Intelligence Made Simple <reply+2i5po9&1wkbyv&&a9bff6f11936b7889568de943df69112b0af82ef434cdc1947b4f4cdbd4bfe44@mg1.substack.com>
To: theonlyamos@gmail.com

Mon, Nov 11, 2024 at 5:57 AM

Forwarded this email? Subscribe here for more

A Practical Introduction to Agentic AI[Guest] [Agents]

An Overview of the AI Agent Landscape from a group that has built Agents for over 2 Billion Users

DEVANSH
NOV 11

READ IN APP ↗

Hey, it's Devansh 👋👋

Our chocolate milk cult has a lot of experts and prominent figures doing cool things. In the series Guests, I will invite these experts to come in and share their insights on various topics that they have studied/worked on. If you or someone you know has interesting ideas in Tech, AI, or any other fields, I would love to have you come on here and share your knowledge.

I put a lot of effort into creating work that is informative, useful, and independent from undue influence. If you'd like to support my writing, please consider becoming a paid subscriber to this newsletter. Doing so helps me put more effort into writing/research, reach more people, and supports my *crippling chocolate milk addiction. Help me democratize the most important ideas in AI Research and Engineering to over 100K readers weekly.* **Many companies have a learning budget that you can expense this newsletter to. You can use the following for an email template to request reimbursement for your subscription.**

Help me buy chocolate milk

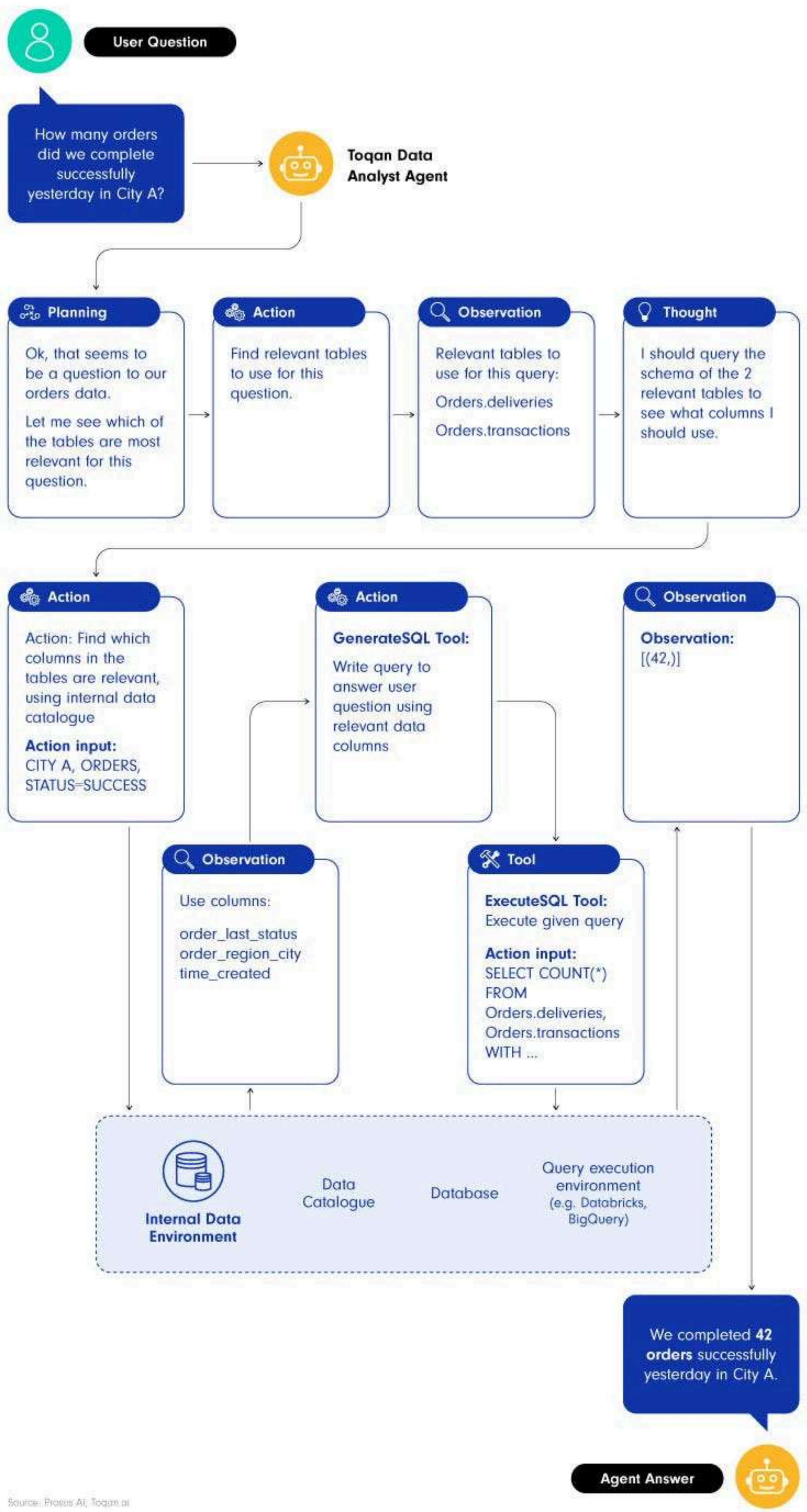
PS- We follow a "pay what you can" model, which allows you to support within your means, and support my mission of providing high-quality technical education to everyone for less than the price of a cup of coffee. Check out this post for more details and to find a plan that works for you.

As you might have picked up from our last few posts, we're gearing up for a series on Agentic AI. For part 1, I figured I'd bring in insights from folks who are putting AI agents into production. As luck would have it The MLOps Community and Prosus will be having a free virtual conference on November 13th with over 40 speakers who are actively working with AI agents in production (sign up for free here).

So I synced up with Demetrios Brinkmann (super fun guy btw), one of the leaders in the MLOps community. DB shared some key insights from the Prosus team and their experiences building their AI agent- Toqan. Prosus is a global consumer internet group and are one of the largest technology investors AND operators in the world.

This article will introduce the space from the perspective of both a successful builder who successfully monetized their agents and investors who are surveying the space.

Example: How Does an SQL/Data Agent Work?



User Question

How many orders did we complete successfully yesterday in City A?

Toqan Data Analyst Agent

Planning

Ok, that seems to be a question to our orders data.

Let me see which of the tables are most relevant for this question.

Action

Find relevant tables to use for this question.

Observation

Relevant tables to use for this query:

Orders.deliveries

Orders.transactions

Thought

I should query the schema of the 2 relevant tables to see what columns I should use.

Action

Action: Find which columns in the tables are relevant, using internal data catalogue

Action input:
CITY A, ORDERS, STATUS=SUCCESS

Observation

Use columns:

order_last_status
order_region_city
time_created

Action

GenerateSQL Tool:

Write query to answer user question using relevant data columns

Tool

ExecuteSQL Tool:
Execute given query

Action input:
SELECT COUNT(*)
FROM
Orders.deliveries,
Orders.transactions
WITH ...

Observation

Observation:
[(42,)]

Internal Data Environment

Data Catalogue

Database

Query execution environment
(e.g. Databricks, BigQuery)

We completed **42 orders** successfully yesterday in City A.

Agent Answer

Source: Prosus AI, Toqan.ai

A preview of what you have to look forward to.

In the Prosus AI team, we continuously explore how new AI capabilities can be used to help solve real problems for the **2 billion users** we collectively serve across companies in the Prosus Group.

For the last 2 years we’ve built and tested AI agents for a wide range of use cases, from conversational data analysis to educational tutors and intelligent marketplace assistants that lead the buying and selling experience for food and groceries ordering, e-commerce and second-hand marketplaces.

We’ve learned that building useful agents is, surprise surprise… hard.

But when they do work, they can be hugely valuable and allow a reinventing of the user experience. Below we share some of our learnings as well as an overview of the agent & tooling ecosystem.

What’s in an agent?

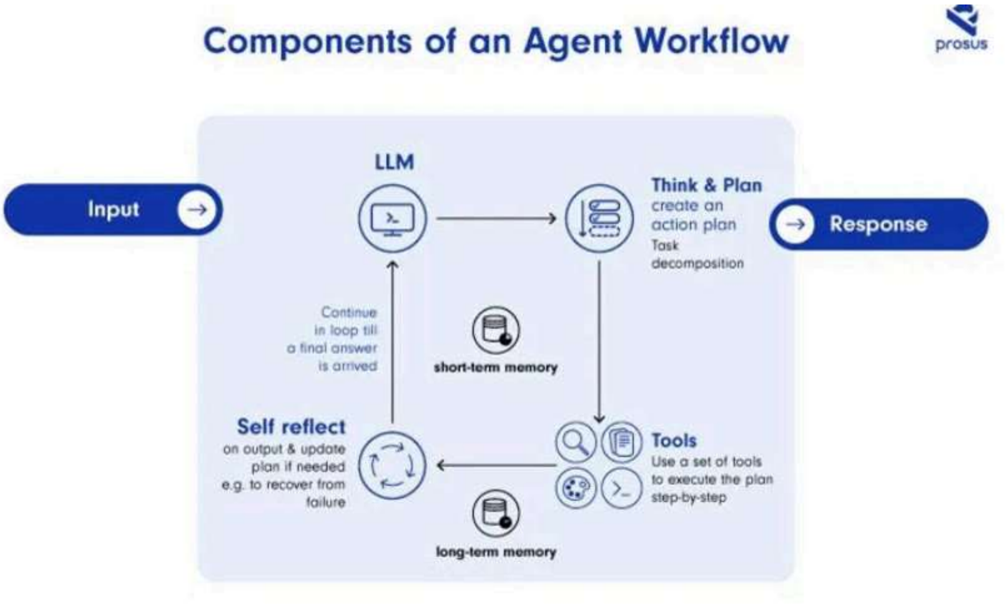
Key to any conversation about agentic systems is a common understanding of exactly what they are. Broadly, agents are AI systems that can make decisions and *take actions* on their own, following general instructions from a user. They tend to have four main components:

1. **Powerful large language model** (LLM), which understands the user’s intent and creates an action plan based on the objective and the tools the agent has access to.

2. **Tools** that add additional capabilities over the core LLM such as web search, document retrieval, code execution, database integration and maybe include other AI models. These tools enable the agent to execute actions like creating a document, executing database queries, creating a chart, etc.

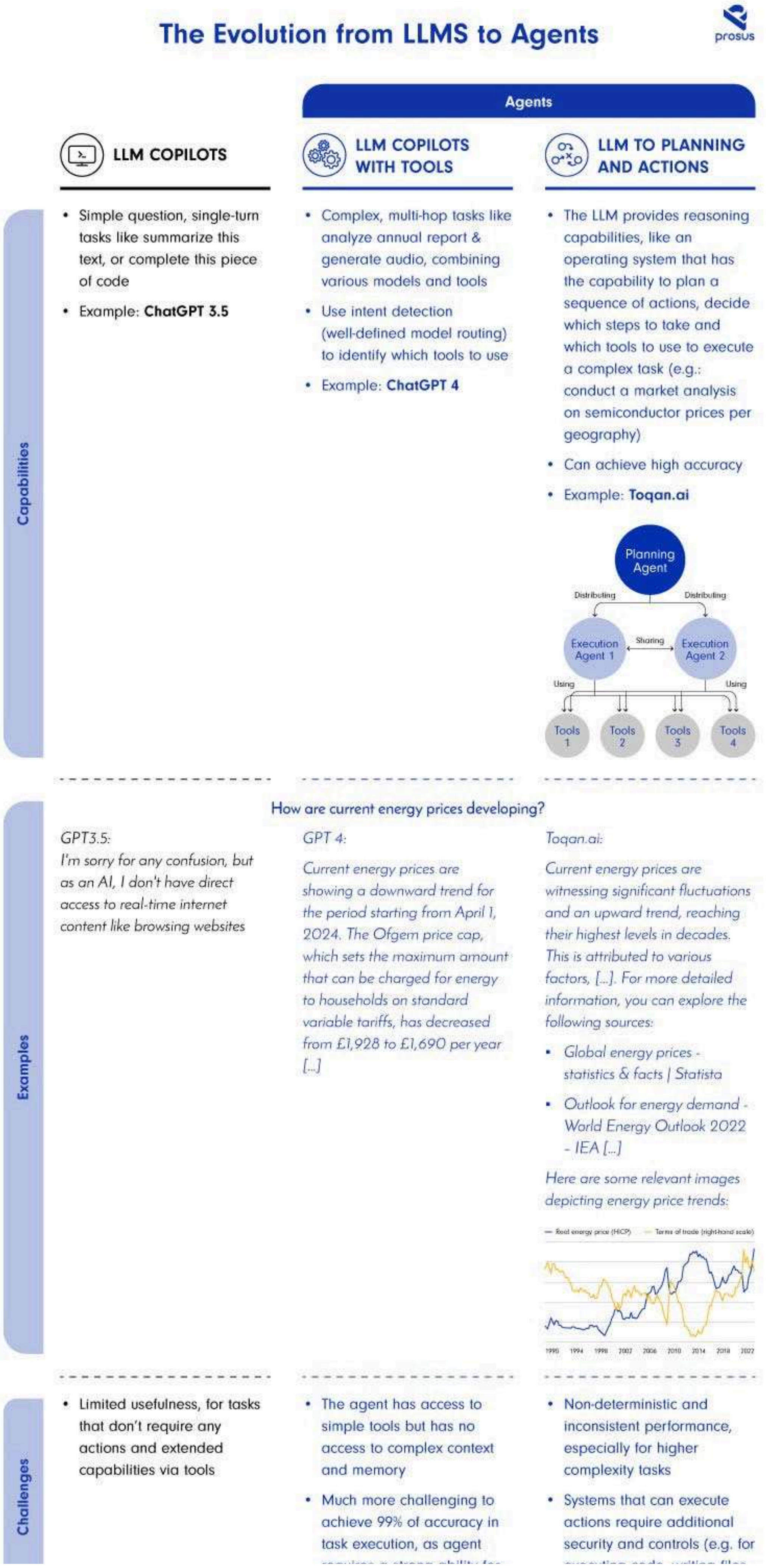
3. **Memory** that includes access to relevant knowledge such as databases (long-term memory) and the ability to retain request-specific information over the multiple steps it may take to complete the action plan (short-term memory).

4. **Reflection and self-critiquing**: more advanced agents also have the ability to spot and correct mistakes they might make as they execute their action plan and reprioritise steps.


Components of an Agent Workflow
prosus
Input
LLM
Think & Plan
create an action plan
Task decomposition
Response
Continue in loop till a final answer is arrived
short-term memory
Self reflect
on output & update plan if needed
e.g. to recover from failure
Tools
Use a set of tools to execute the plan step-by-step
long-term memory

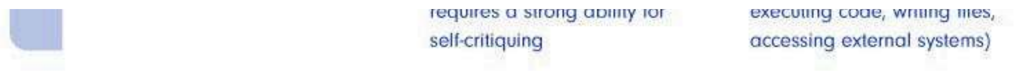
Agents can come in various degrees of sophistication, depending on the quantity and quality of the tools, as well the LLM used, and the constraints and controls placed on workflows created by the agents.

See below (Evolution from LLMs to Agents), a comparison of a single-turn chatbot to two agents.



The Evolution from LLMS to Agents

		Agents	
	LLM COPILOTS	**LLM COPILOTS WITH TOOLS**	**LLM TO PLANNING AND ACTIONS**
Capabilities	- Simple question, single-turn tasks like summarize this text, or complete this piece of code - Example: **ChatGPT 3.5**	- Complex, multi-hop tasks like analyze annual report & generate audio, combining various models and tools - Use intent detection (well-defined model routing) to identify which tools to use - Example: **ChatGPT 4**	- The LLM provides reasoning capabilities, like an operating system that has the capability to plan a sequence of actions, decide which steps to take and which tools to use to execute a complex task (e.g.: conduct a market analysis on semiconductor prices per geography) - Can achieve high accuracy - Example: **Toqan.ai**
Examples	*How are current energy prices developing?*		
	GPT3.5: *I'm sorry for any confusion, but as an AI, I don't have direct access to real-time internet content like browsing websites*	*GPT 4:* *Current energy prices are showing a downward trend for the period starting from April 1, 2024. The Ofgem price cap, which sets the maximum amount that can be charged for energy to households on standard variable tariffs, has decreased from £1,928 to £1,690 per year [...]*	*Toqan.ai:* *Current energy prices are witnessing significant fluctuations and an upward trend, reaching their highest levels in decades. This is attributed to various factors, [...]. For more detailed information, you can explore the following sources:* - *Global energy prices - statistics & facts | Statista* - *Outlook for energy demand - World Energy Outlook 2022 - IEA [...]* *Here are some relevant images depicting energy price trends:*
Challenges	- Limited usefulness, for tasks that don't require any actions and extended capabilities via tools	- The agent has access to simple tools but has no access to complex context and memory - Much more challenging to achieve 99% of accuracy in task execution, as agent	- Non-deterministic and inconsistent performance, especially for higher complexity tasks - Systems that can execute actions require additional security and controls (e.g. for


requires a strong ability for
self-critiquing
executing code, writing files,
accessing external systems)

Source: Prosus AI, March 2024

Why we build agentic systems

The drive to create agentic systems is grounded in the shortcomings of existing AI Copilots.

As of today most AI co-pilots are still limited to single-turn tasks like “summarise this text”, or “complete this piece of code”.

Agents on the other hand promise to complete more complex, multi-hop tasks such as: “find me the best running shoe and buy it for me”, “analyse this annual report to provide a perspective on the growth potential of this company”, or “provide an overview of the market for wearables including our own internal revenue data”.

In essence an agent is debating with itself where it can choose between these conditions:

```
# Run until broken
while True:



    elif condition_generate_plan:
        plan = generate_plan()
        execute_plan(plan)



    elif condition_think:
        thought = think()
        process_thought(thought)



    if condition_call_tool:
        tool_output = call_single_tool()
        process_tool_output(tool_output)
        # (this breaks the loop)
        elif condition_ask_user:
            user_response = ask_user()
            break
```

For the specific running shoes example:

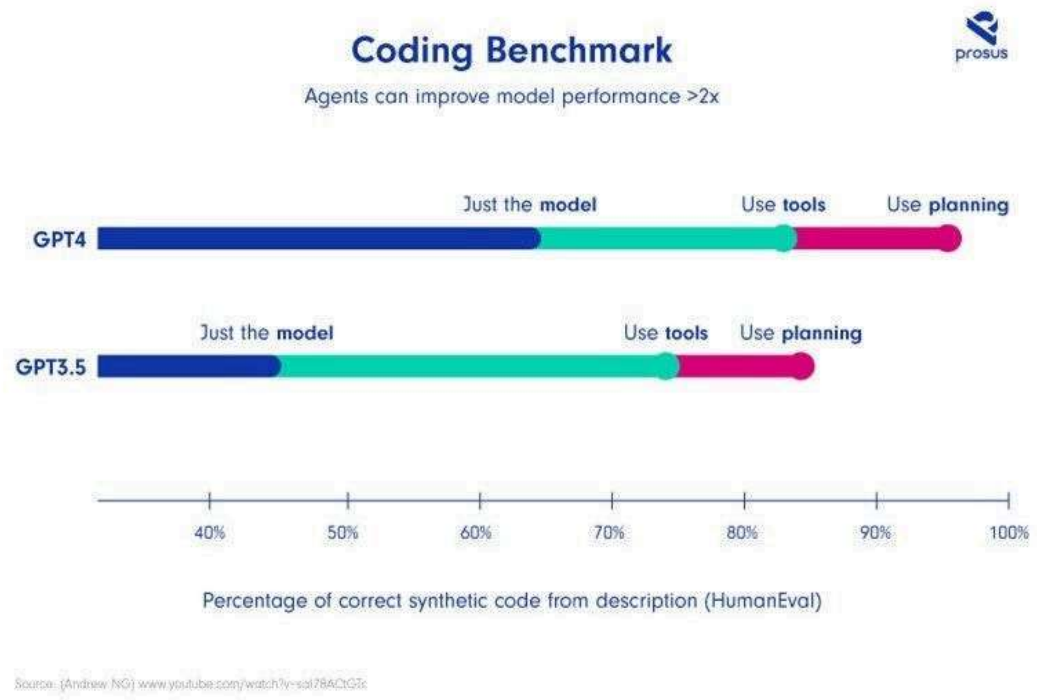
```
# Generate and execute plan
plan = generate_plan()
print(f"Generated Plan: {plan}")
execute_plan(plan)



# Think and process thought
thought = think()
process_thought(thought)



# Search for shoes
tool_output_search = call_single_tool("search_shoes", {})
shoes = tool_output_search
process_tool_output(tool_output_search)



# Fetch reviews for each shoe
reviews_dict = {}
for shoe in shoes:
    tool_output_reviews = call_single_tool("fetch_
reviews", {"shoe": shoe})
    reviews_dict[shoe] = tool_output_reviews
    process_tool_output(tool_output_reviews["reviews"])



# Determine the best shoe based on reviews
best_shoe = max(reviews_dict, key=lambda shoe:
```

```
reviews_dict[shoe]["reviews"])
print(f"Best Shoe: {best_shoe}")


# Purchase the best shoe
tool_output_purchase = call_single_tool("purchase_shoe",
{"shoe": best_shoe})
process_tool_output(tool_output_purchase["status"])


# Final outputs
print(f"Tool Output Search: {tool_output_search}")
print(f"Tool Output Reviews: {reviews_dict[best_shoe]['
reviews']}")
print(f"Purchase Status: {tool_output_purchase['status'
]}")
```

The impact versus simple co-pilots can be seen in the answers each can provide to the same basic request.

More comprehensive answers could provide a reason for building agents on its own. However, as an added value, agents can also quantifiably improve the quality of results, as was recently described by Andrew Ng.

The chart below (Coding Benchmark) shows that using GPT3.5 with agentic capabilities can comfortably beat a much larger model such as GPT-4 –one of today’s most powerful models—at a complex coding task. When just using the model, GPT3.5 significantly underperforms at the same task compared to its more capable larger brother.


Coding Benchmark

Agents can improve model performance >2x

Source: (Andrew NG) www.youtube.com/watch?v=sal78ACtGTc

Not all plain sailing

Of course, that begs the question that if agents are so great, why doesn't all AI interaction use agents already?

Despite the progress, the journey towards fully realising the potential of AI agents with large LLMs is new and is filled with challenges that pose significant hurdles to building useful and reliable agentic systems. We see the challenges are broadly in three categories: technology readiness, scalability of agentic systems, tooling and integrations.

Challenges and remedies

I want to quickly point out a few challenges we have encountered as we tried to shop agents into production and the solutions we created to try and solve them.

Evaluating Outputs (i.e function calling, action planning, etc)—This was one of the main headaches. We didn't want to reinvent the wheel but ended up creating our own LLM benchmark leaderboard that can evaluate an LLM capability across 10 different test sets. My favorite of course is the function calling eval set—it's such a cool way to test how different models choose the right function at the right time. Like, does a model actually do a web-search when it needs fresh or time-dependent data, or is it just pulling from what it already knows?

Having a scalable tool orchestration framework—After much deliberation and experimentation, we chose Go as our preferred framework, moving away from Python. This shift was driven by two key challenges: First, our agents' vast toolset and capabilities made runtime-generated code unreliable—we'd often discover bugs days into testing simply because of unique edge cases. Go, being a compiled language, forces better error handling and caught these issues early. Second, we needed both scalability and customization potential, which we achieved by implementing event sourcing alongside Go.

Personalization—Sometimes tools need to behave differently based on the user that uses it. A simple one is Knowledge bases. I just want to see my files, not others! Propagating this from input to tool is a tricky one.

We have a universal configuration file that can customize many parts of the agent and tools to the level of a single user or channel. Each service that might need changing based on users can tap into this system.

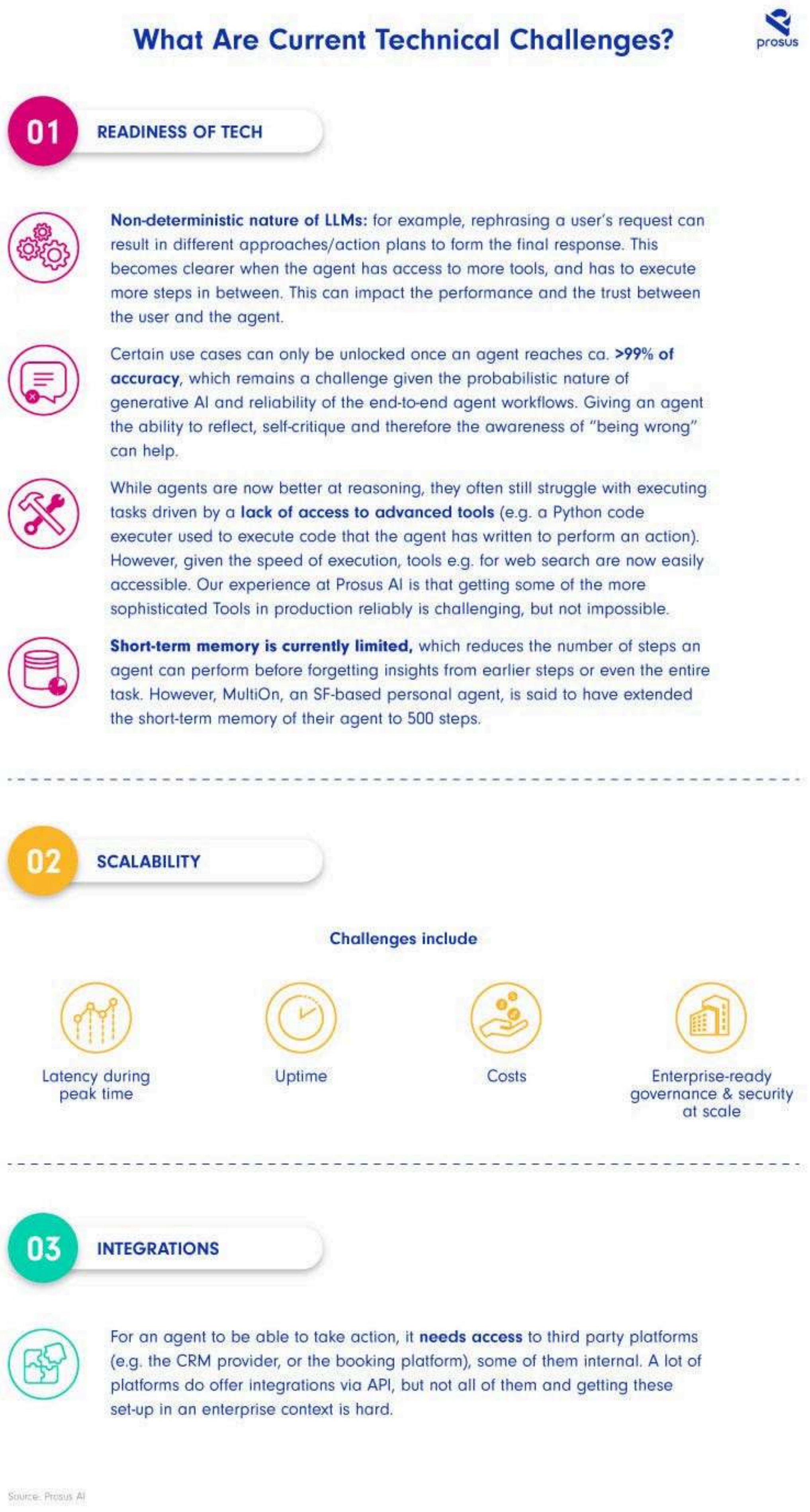
Sharing tool output in a meaningful way—I am not sure we fully figured this one out. We are constantly experimenting with different thresholds for when to put outputs directly into the LLM call and when to attach files.

For example; Were one might see benefits in getting a markdown table, someone with a bit larger data frame rather sees it returning a CSV.

End user adoption—This maybe the biggest hurdle. Because:

1. Users don't (always) understand how the agent is different from LLMs/chatGPT. Meaning, they don't understand what/how they can utilise the agent's tools for their ends—tools are chosen by the agent and often hidden from the user.
2. Users sometimes uncomfortable with or intimidated by an iterative way of working and give up quickly on prompt engineering
3. They're 'too busy' to train themselves and experiment on an agent properly
4. Biggest hurdle yet is finding use cases that are suitable for the tasks in their role specifically. How can we make the experience more intuitive for the user?
5. Feature requests don't always align with usage. Requested features don't end up getting used because another part of the flow does not

work and they dont spend the time to try and make it work. (See numbers 2 and 3)



What Are Current Technical Challenges?

prosus

01 READINESS OF TECH

Non-deterministic nature of LLMs: for example, rephrasing a user's request can result in different approaches/action plans to form the final response. This becomes clearer when the agent has access to more tools, and has to execute more steps in between. This can impact the performance and the trust between the user and the agent.

Certain use cases can only be unlocked once an agent reaches ca. **>99% of accuracy**, which remains a challenge given the probabilistic nature of generative AI and reliability of the end-to-end agent workflows. Giving an agent the ability to reflect, self-critique and therefore the awareness of "being wrong" can help.

While agents are now better at reasoning, they often still struggle with executing tasks driven by a **lack of access to advanced tools** (e.g. a Python code executer used to execute code that the agent has written to perform an action). However, given the speed of execution, tools e.g. for web search are now easily accessible. Our experience at Prosus AI is that getting some of the more sophisticated Tools in production reliably is challenging, but not impossible.

Short-term memory is currently limited, which reduces the number of steps an agent can perform before forgetting insights from earlier steps or even the entire task. However, MultiOn, an SF-based personal agent, is said to have extended the short-term memory of their agent to 500 steps.

02 SCALABILITY

Challenges include

- Latency during peak time
- Uptime
- Costs
- Enterprise-ready governance & security at scale

03 INTEGRATIONS

For an agent to be able to take action, it **needs access** to third party platforms (e.g. the CRM provider, or the booking platform), some of them internal. A lot of platforms do offer integrations via API, but not all of them and getting these set-up in an enterprise context is hard.

Source: Prosus AI

Opportunities with task- and industry-specific agents

As mentioned above, we experienced that agents usually get better as they are built for a specific domain or a narrower set of tasks.

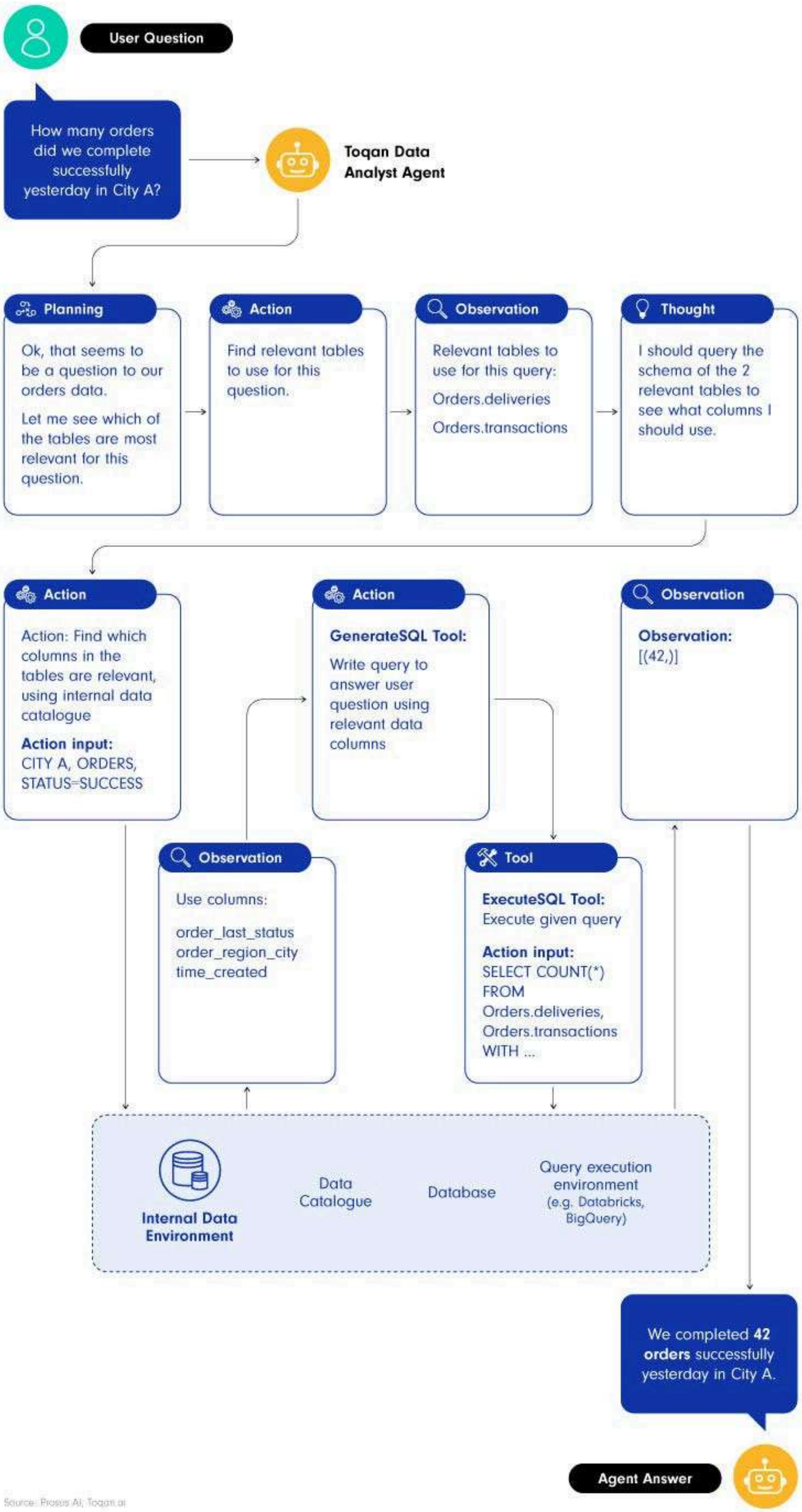
As a result, while the landscape is still evolving, we are increasingly excited about task- and industry-specific agents which promise to offer tailored solutions that address specific challenges and requirements and help resolve some of the issues faced when building agents.

One such example is a task-specific agent we built to assist with conversational data analysis to give anyone in an organisation access to internal data without the need to know how to query databases.

Getting the right information at the right time, to enable fact-based decision making because often the data sits in internal databases and requires data analysts who understand the data sources, to write queries to extract the data from the relevant databases—this is a complex workflow.

With task-specific agents focused on areas like data-analysis, it becomes easier to search through information, access databases, evaluate the relevance of the information, and synthesise it to answer a user's question. We used that approach when building our Toqan data analyst agent below. This is how it works:

Example: How Does an SQL/Data Agent Work?



User Question

How many orders did we complete successfully yesterday in City A?

Toqan Data Analyst Agent

Planning

Ok, that seems to be a question to our orders data.

Let me see which of the tables are most relevant for this question.

Action

Find relevant tables to use for this question.

Observation

Relevant tables to use for this query:

Orders.deliveries

Orders.transactions

Thought

I should query the schema of the 2 relevant tables to see what columns I should use.

Action

Action: Find which columns in the tables are relevant, using internal data catalogue

Action input:
CITY A, ORDERS, STATUS=SUCCESS

Action

GenerateSQL Tool:

Write query to answer user question using relevant data columns

Observation

Observation:
[(42,)]

Observation

Use columns:

order_last_status
order_region_city
time_created

Tool

ExecuteSQL Tool:
Execute given query

Action input:
SELECT COUNT(*)
FROM
Orders.deliveries,
Orders.transactions
WITH ...

Internal Data Environment

Data Catalogue

Database

Query execution environment
(e.g. Databricks, BigQuery)

We completed **42 orders** successfully yesterday in City A.

Agent Answer

Source: Prosus AI, Toqan.ai

Using this workflow and tuning it over time, we now have a scalable framework that improved upon answer accuracy significantly, from 50% perceived accuracy to up to 100% on specific high-impact use cases.

For more information on the inner workings of the Toqan Data Analyst, see our blog detailing the technical learnings.

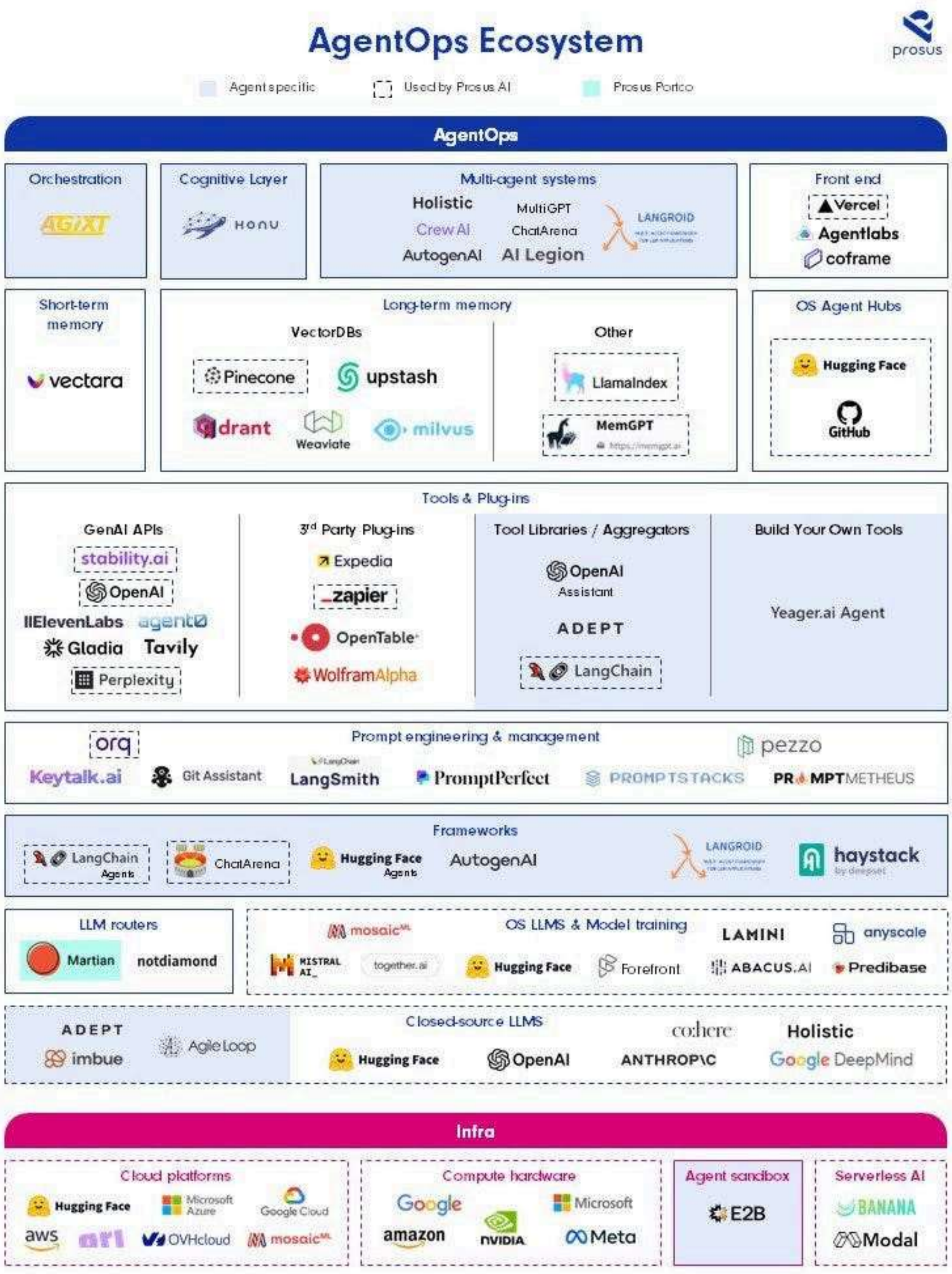
The Ops of Agents

As we've discussed, building agent systems entails more than simply prompt engineering a powerful LLM—though it is ongoing advancements in model training like function calling (which makes it possible to call external tools) and more powerful LLMs which can reason and plan that make agents possible.

Creating an effective agent requires building tools for the agent to access (e.g. writing and executing their own code, browsing the web, writing and reading databases), building execution environments, integrating with systems, enabling a level of planning & self-reflection, and more.

As we built Toqan and other agent-based systems we frequently found ourselves solving hard technical problems and looking for tools to build with.

As a result, we put together the AgentOps landscape together with our Prosus Ventures team to highlight some of the tools we considered—see below. We hope this is a useful guide for others interested in following Agents.

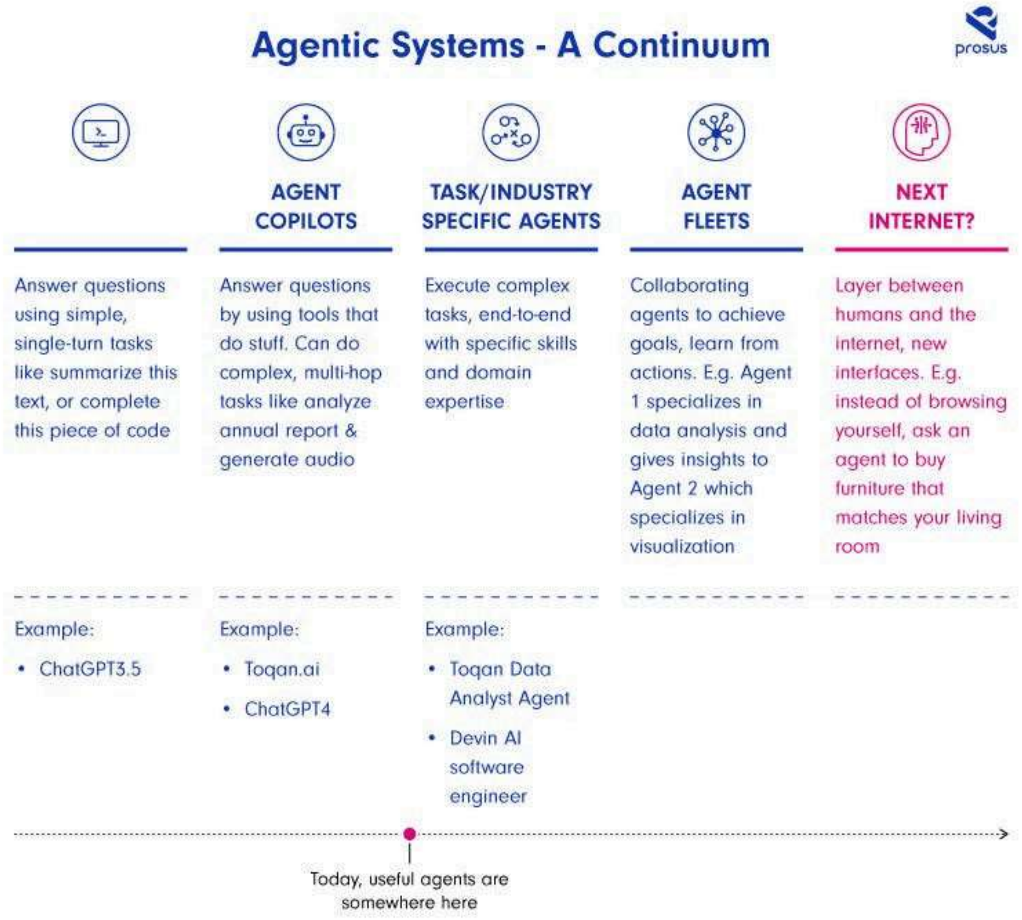


Source: Prosus AI, E2B, Activant Capital, Aura VC

Just the beginning

While it may not feel like it to those of us building the systems, AI agents are still very much in their infancy.

The journey towards building effective and ubiquitous autonomous AI agents is still one of continuous exploration and innovation. As we navigate the complexities and challenges of productising these agents, the potential for transformative change is increasingly evident.



Agentic Systems - A Continuum

	AGENT COPILOTS	TASK/INDUSTRY SPECIFIC AGENTS	AGENT FLEETS	NEXT INTERNET?
Answer questions using simple, single-turn tasks like summarize this text, or complete this piece of code	Answer questions by using tools that do stuff. Can do complex, multi-hop tasks like analyze annual report & generate audio	Execute complex tasks, end-to-end with specific skills and domain expertise	Collaborating agents to achieve goals, learn from actions. E.g. Agent 1 specializes in data analysis and gives insights to Agent 2 which specializes in visualization	Layer between humans and the internet, new interfaces. E.g. instead of browsing yourself, ask an agent to buy furniture that matches your living room
Example: • ChatGPT3.5	Example: • Toqan.ai • ChatGPT4	Example: • Toqan Data Analyst Agent • Devin AI software engineer		

Today, useful agents are somewhere here

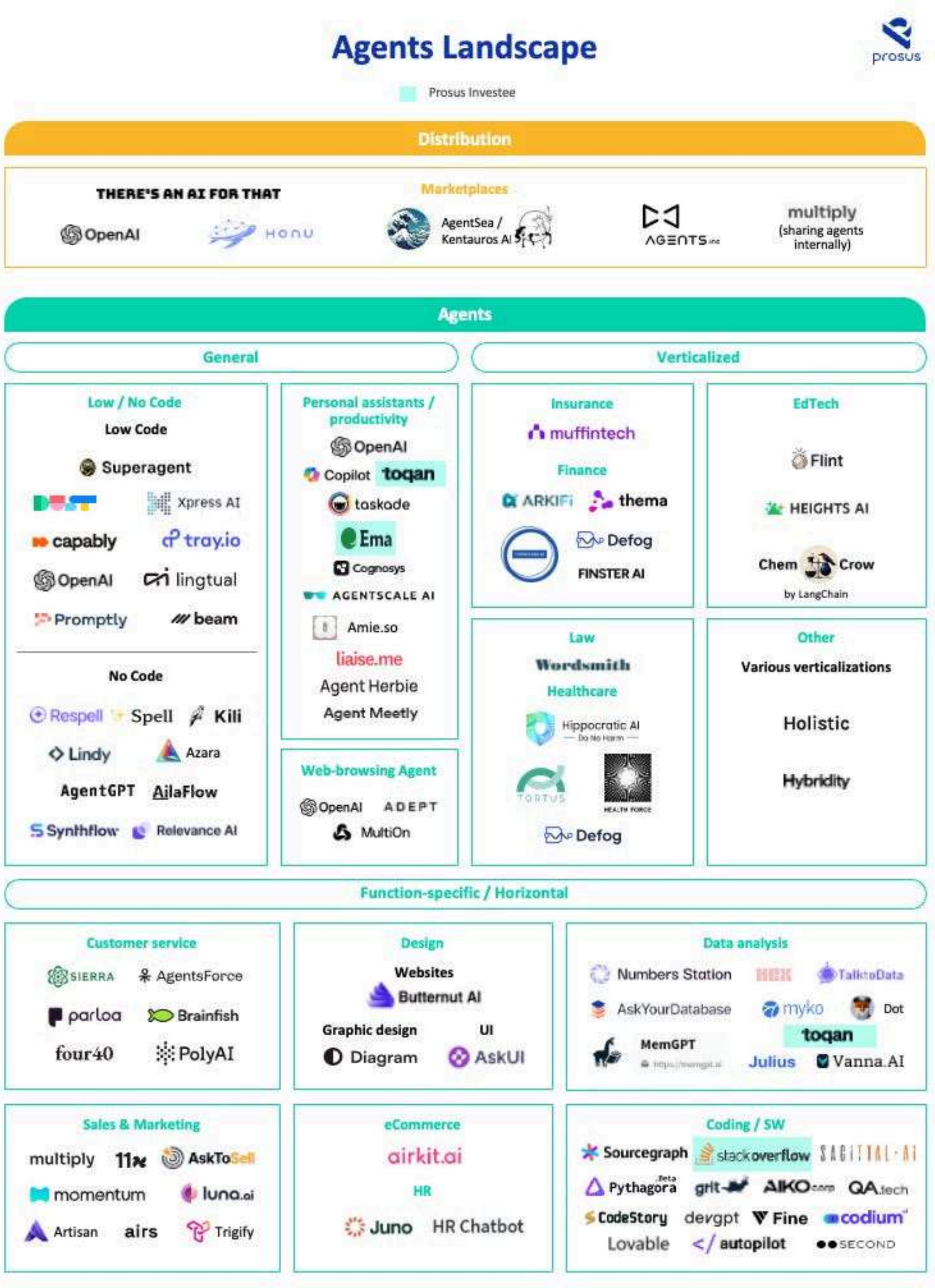
We expect that agents will appear in co-pilots and AI Assistants during this year and will become mainstream for experimentation and for non-mission critical applications such as market research, data visualisation online shopping assistants.

Addendum — Market Landscape

We have also developed a market landscape for existing agents to illustrate the emerging ecosystem.

We think that together they can provide a useful guide for others interested in following agents. However, the space is hot and a new tool comes out every week.

If you think there are other companies we should add to either landscape, please contact me at paul@prosus.com.



Source: Prosus Ventures, Prosus AI, E2B, Activant Capital, Aura VC

Sign up for the damn conference already

I provide various consulting and advisory services. If you'd like to explore how we can work together, reach out to me through any of my socials over here or reply to this email.

I put a lot of work into writing this newsletter. To do so, I rely on you for support. If a few more people choose to become paid subscribers, the Chocolate Milk Cult can continue to provide high-quality and accessible