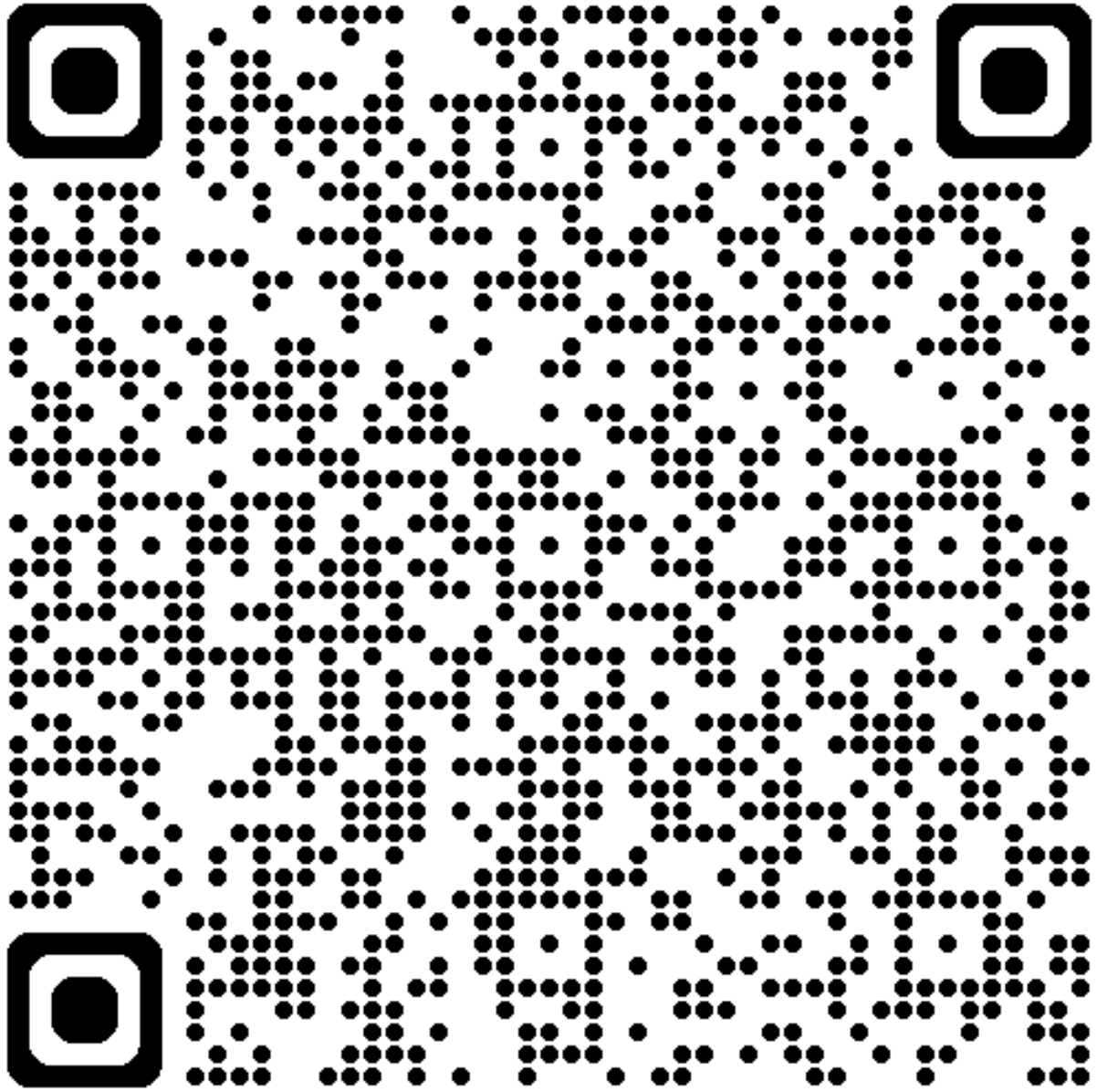
Put An Agent Inside Your App In 10 Minutes Or Less

With the GitHub Copilot SDK

Daniel Ward



daninacan.com

Me

- Software developer, consultant
- Microsoft .NET MVP
- Co-organizer of the San Antonio/Austin .NET User Group
- daninacan.com
- @danielwarddev
- daniel-ward-dev
- danielwarddev.bsky.social

LEAN TECHNIQUES

MVP

SAN ANTONIO .NET

AUSTIN .NET

https://www.meetup.com/sadnug/

https://www.meetup.com/austin-net-user-group/

Outline

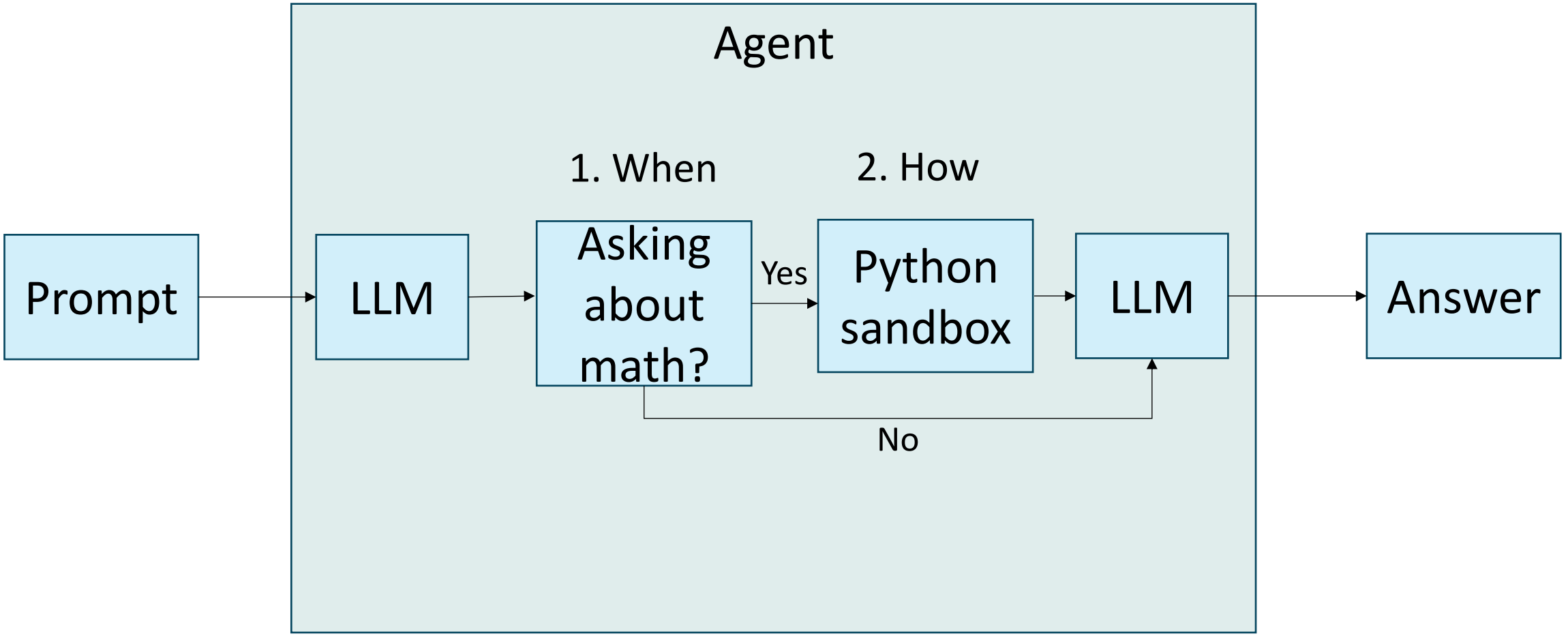
- GitHub Copilot SDK
- What’s an agent?
- How the SDK works + features
- Coding demo –“10 minutes or less”
- Why use the SDK?
- Coding demo – realistic example
- Prod considerations

What is the GitHub Copilot SDK?

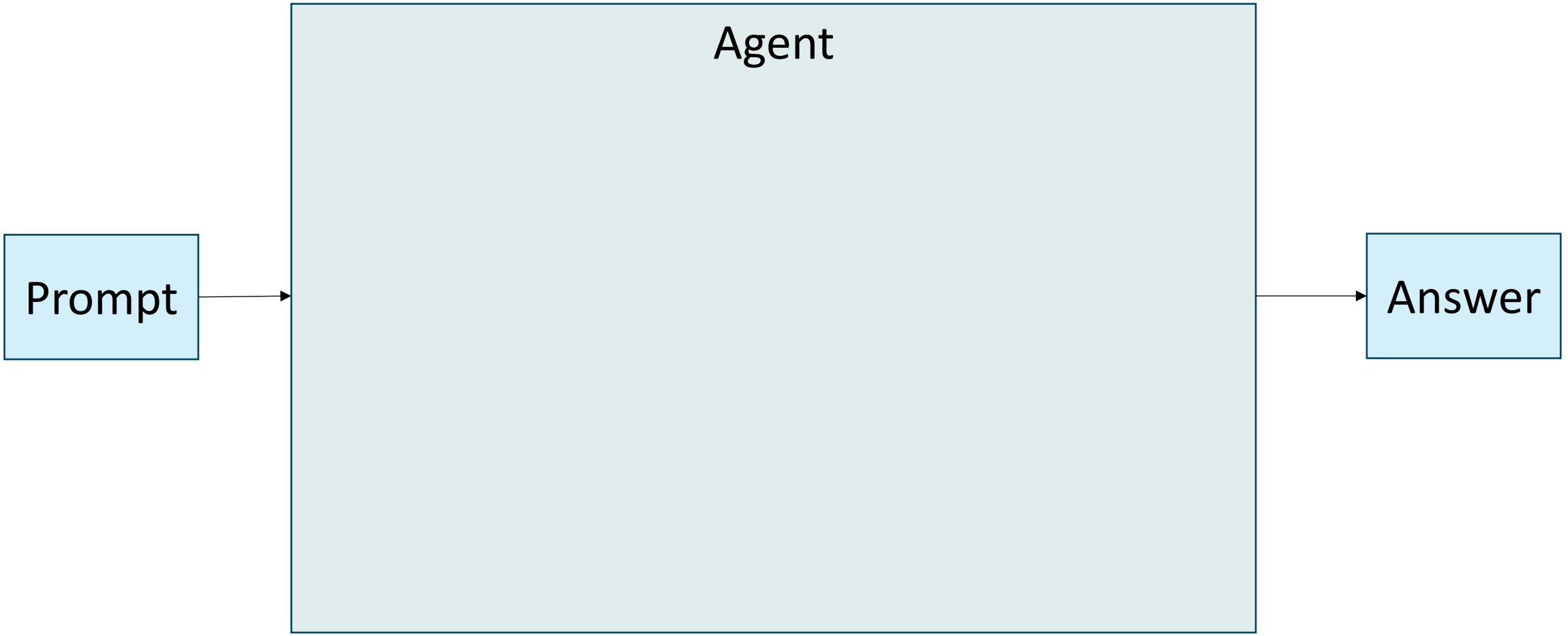
- Open-source SDK to call GitHub Copilot from code, letting you create your own agents
- Wait!
- What is an agent?
- Why do I want my app to have an agent/be an agent?

What is an agent?

What is an agent?



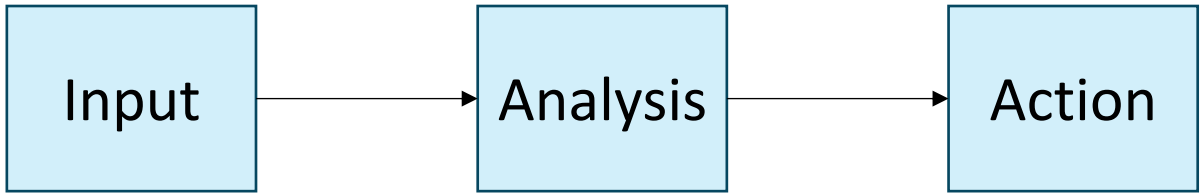
What is an agent?



What is an agent?

- LLM that we've given decision-making capabilities by adding tools
- When to use the tools
- How to use the tools

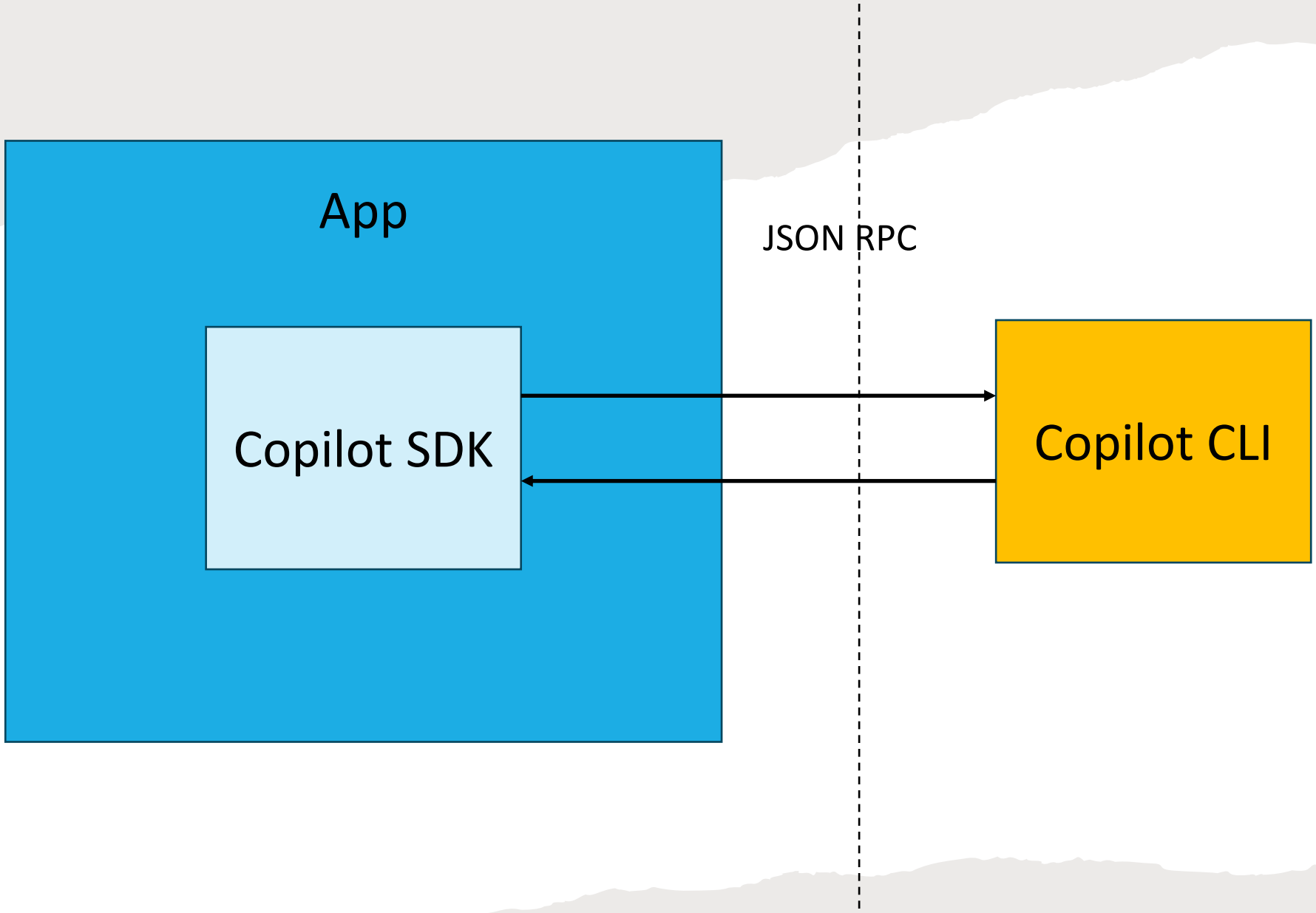
Why would I want an agent in my app?

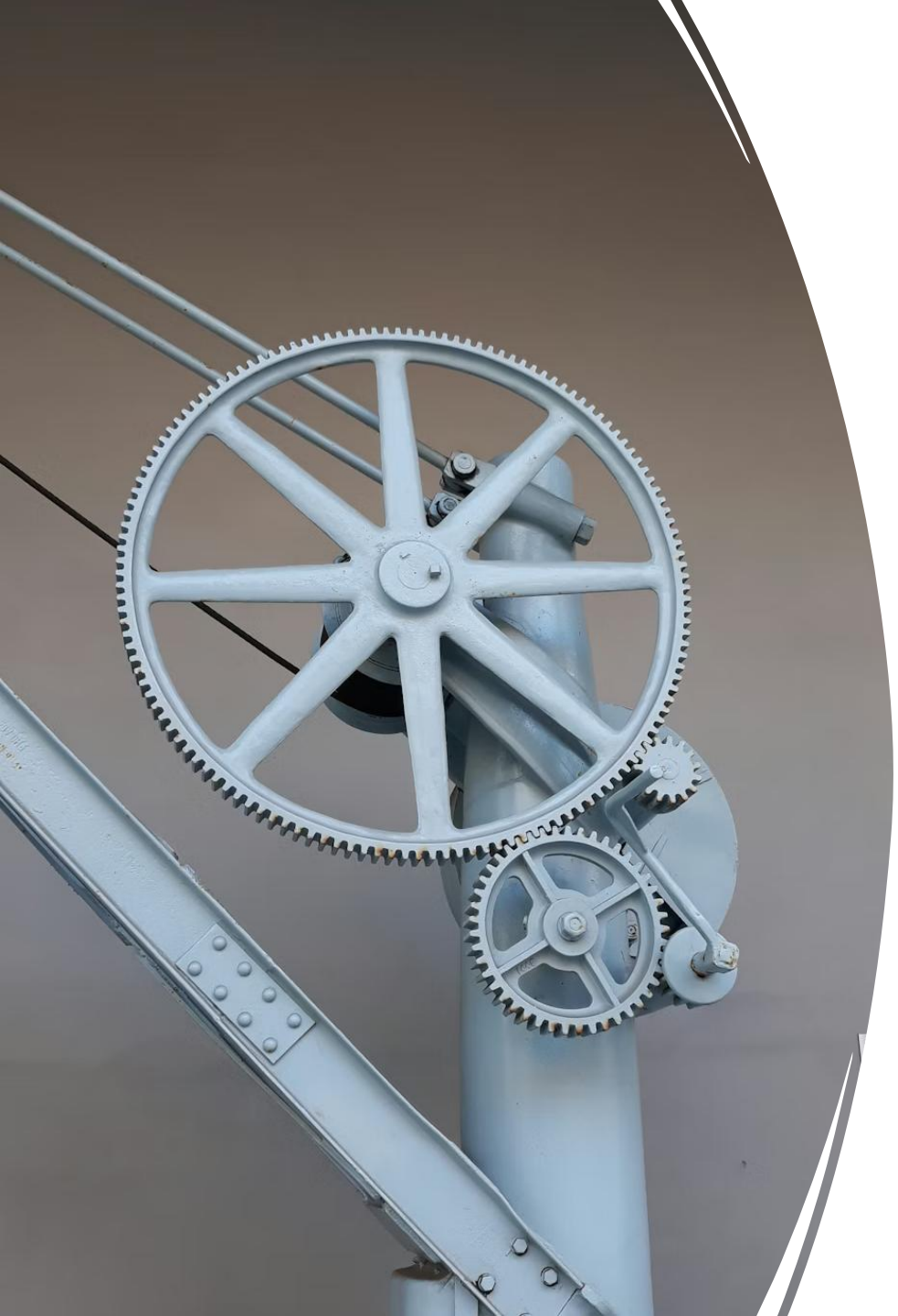


1. Fix a production error by collecting logs, looking at the source code, and suggesting a code change
2. Review a customer's order history to identify churn risk and give personalized offers
3. Analyze current market signals and news to find quality leads and draft outreach

What is the GitHub Copilot SDK?

- Open-source SDK to call GitHub Copilot from code, letting you create your own agents
  - Official: Typescript, Python, Go, C#
  - Unofficial: Java, Rust, Clojure, C++
  - Not yet in version 1.0
- Talks to Copilot CLI through JSON-RPC


App
Copilot SDK
JSON RPC
Copilot CLI

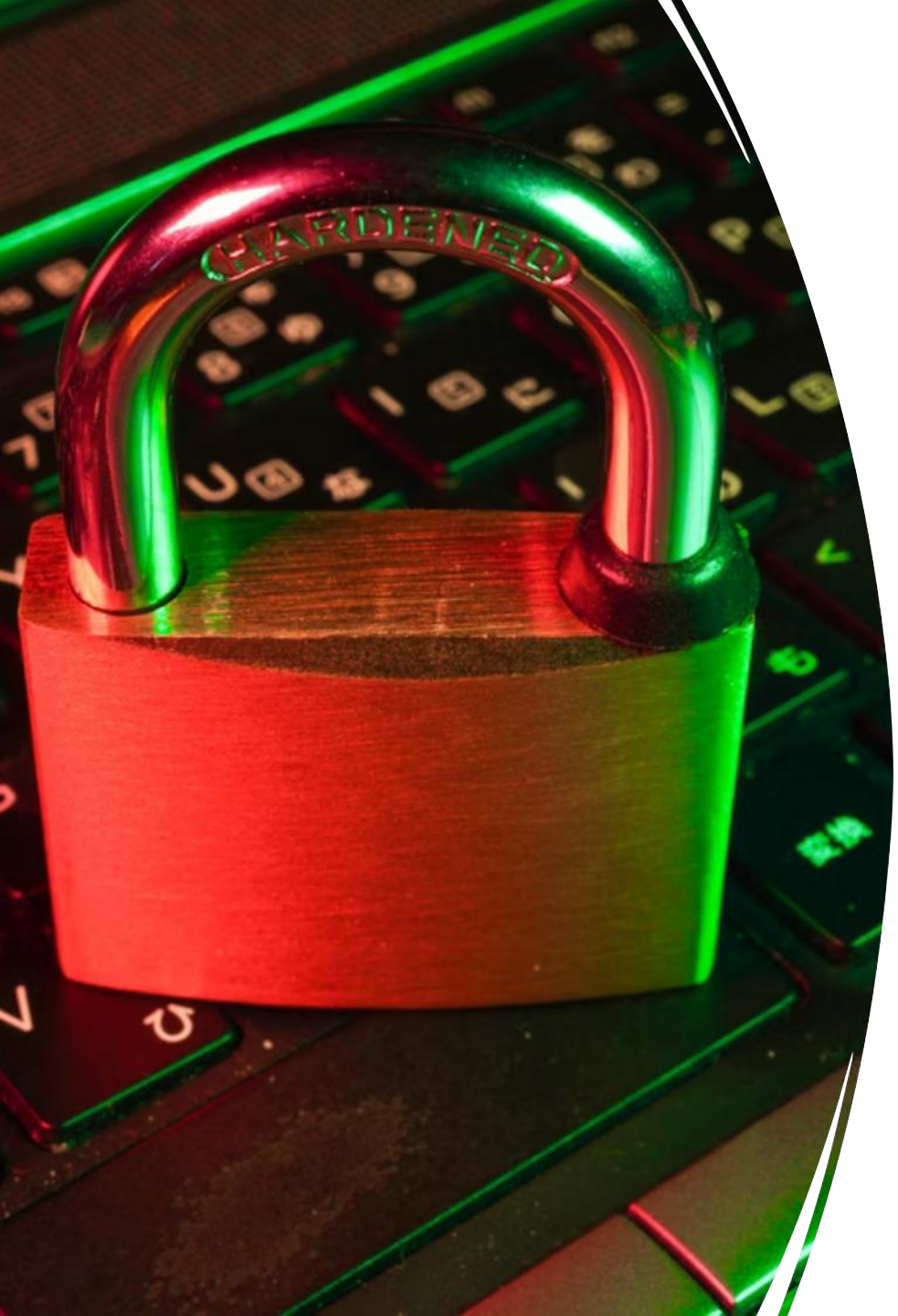


GitHub Copilot SDK features – CLI options

- Run CLI installed on the host machine (default)
- Bundle CLI with app
- Run on remote server

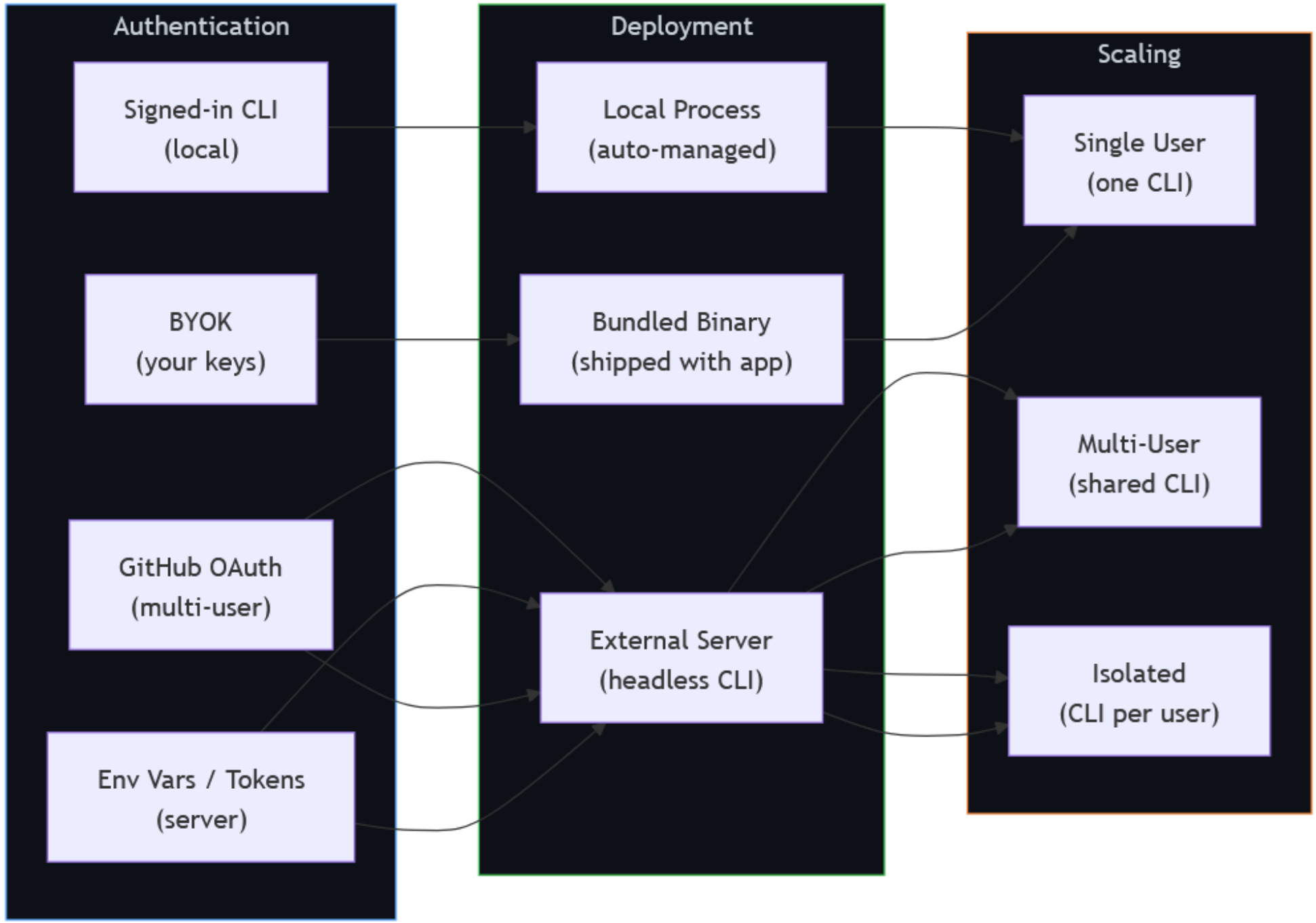
GitHub Copilot SDK features – CLI Parity

- Everything Copilot normally has in the CLI – agents, skills, MCP, etc.
- Some features are not available by design
  - Export to file, interactive UI, YOLO mode, more



GitHub Copilot SDK features - Auth

- Signed in CLI user (default)
- GitHub Oauth
- Env vars
- BYOK – use API key from OpenAI, Azure, Anthropic, etc.


Authentication
Signed-in CLI
(local)
BYOK
(your keys)
GitHub OAuth
(multi-user)
Env Vars / Tokens
(server)
Deployment
Local Process
(auto-managed)
Bundled Binary
(shipped with app)
External Server
(headless CLI)
Scaling
Single User
(one CLI)
Multi-User
(shared CLI)
Isolated
(CLI per user)

Tool calling

- Normal code that you register as tools to the LLM

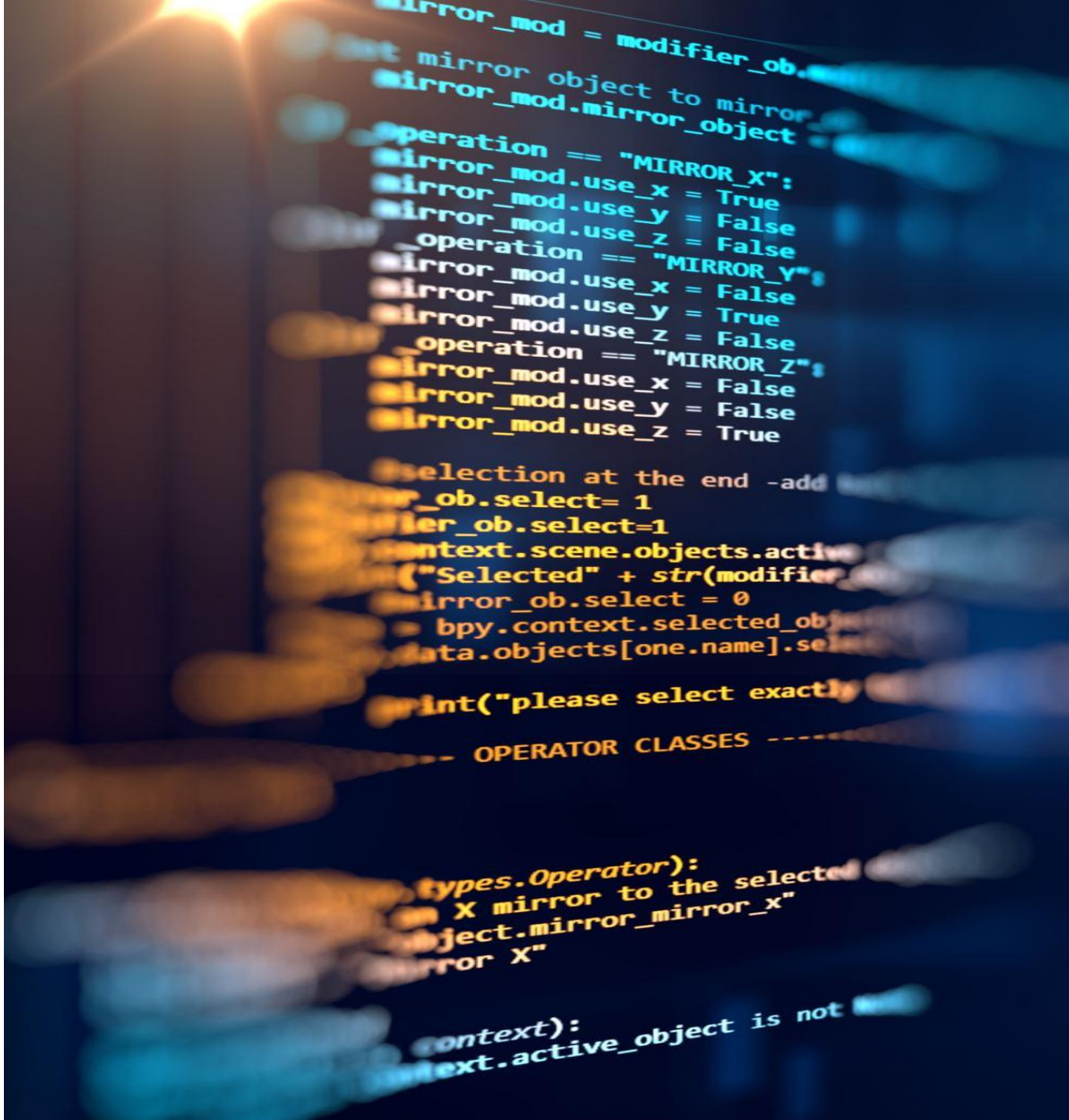
```
[DisplayName("get_weather")]
[Description("Gets the current weather information in the requested city")]
1 reference
WeatherData GetWeather([Description("The name of the city to get the weather for")] string cityName)
{
    return new WeatherData(Temperature: 100, Condition: "It's too dang hot for this time of year");
}

2 references
record WeatherData(int Temperature, string Condition);
```

```
var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = "gpt-5-mini",
    Tools = [AIFunctionFactory.Create(GetWeather)]
});
```

Coding Demo

- Requirements
  - Copilot license (free)
  - CLI installed



Wait! Can’t I just do that with prompts?

- Yes!
- But…
- App workflow vs. prompt workflow

Code vs. AI

Code

- Deterministic
- Good for predictable results, more steps in the process doesn’t make it less reliable
- Can’t be creative

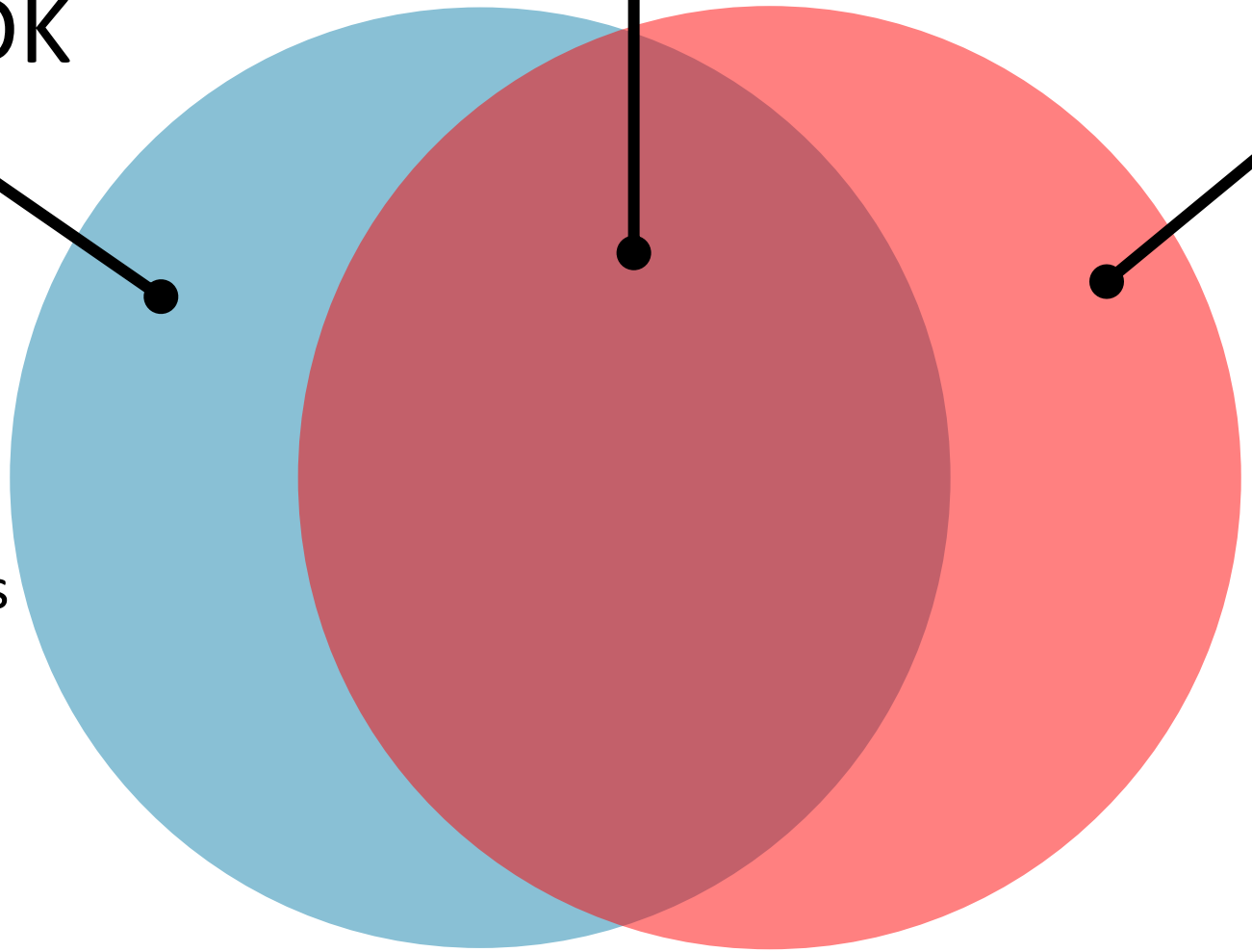
AI

- Non-deterministic
- Bad for predictable results, more prompts chained together makes problems more likely
- Good for creative analysis

Creative analysis

Copilot SDK

- App owns the flow
- Put deterministic flow/guardrails around AI work
- The more steps you have, the better it is
- + More reliable
- - More setup

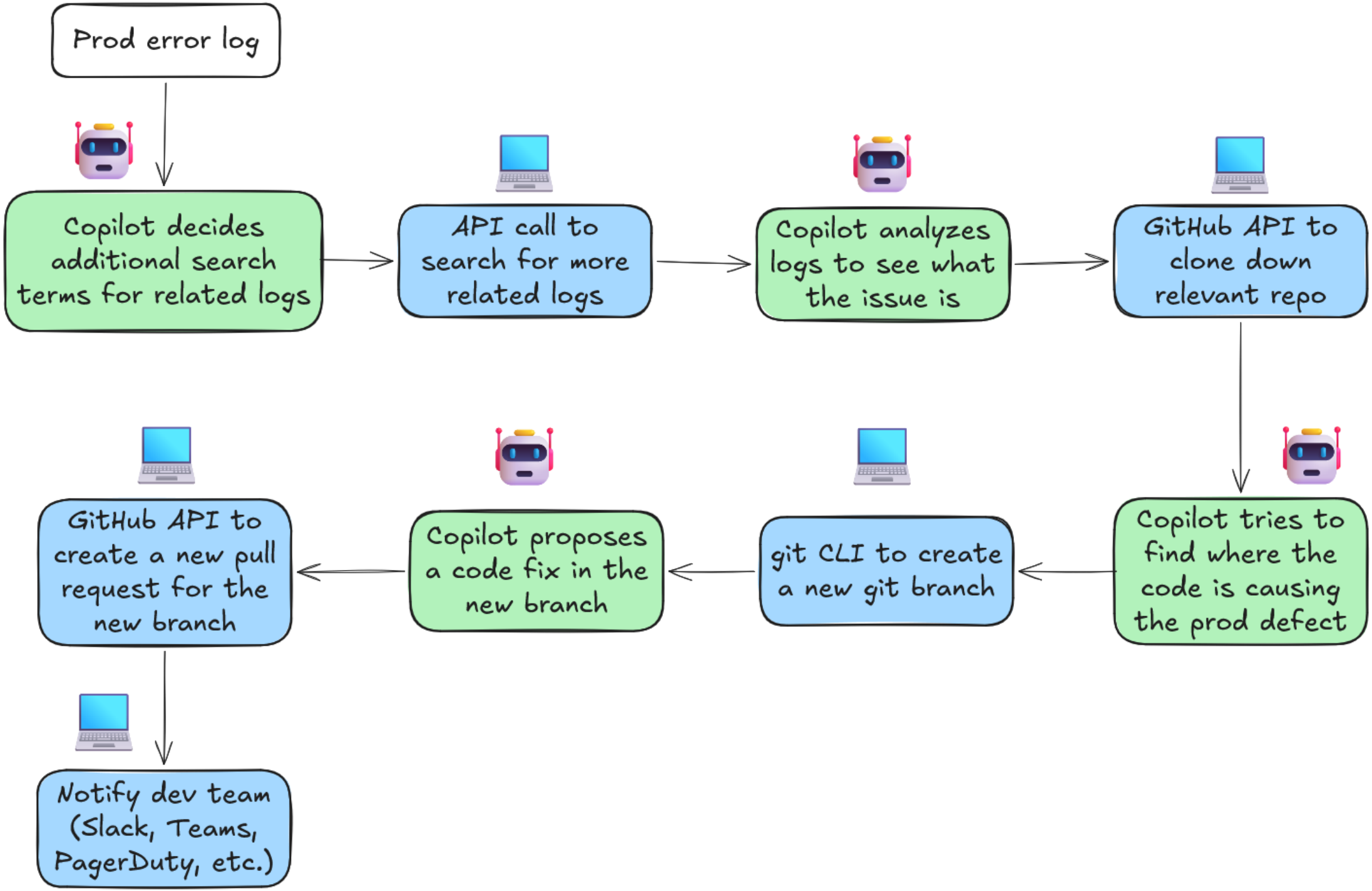


Normal prompting

- User owns the flow
- Prompt chaining
- The more steps you have, the worse it is
- - Less reliable
- + Less setup

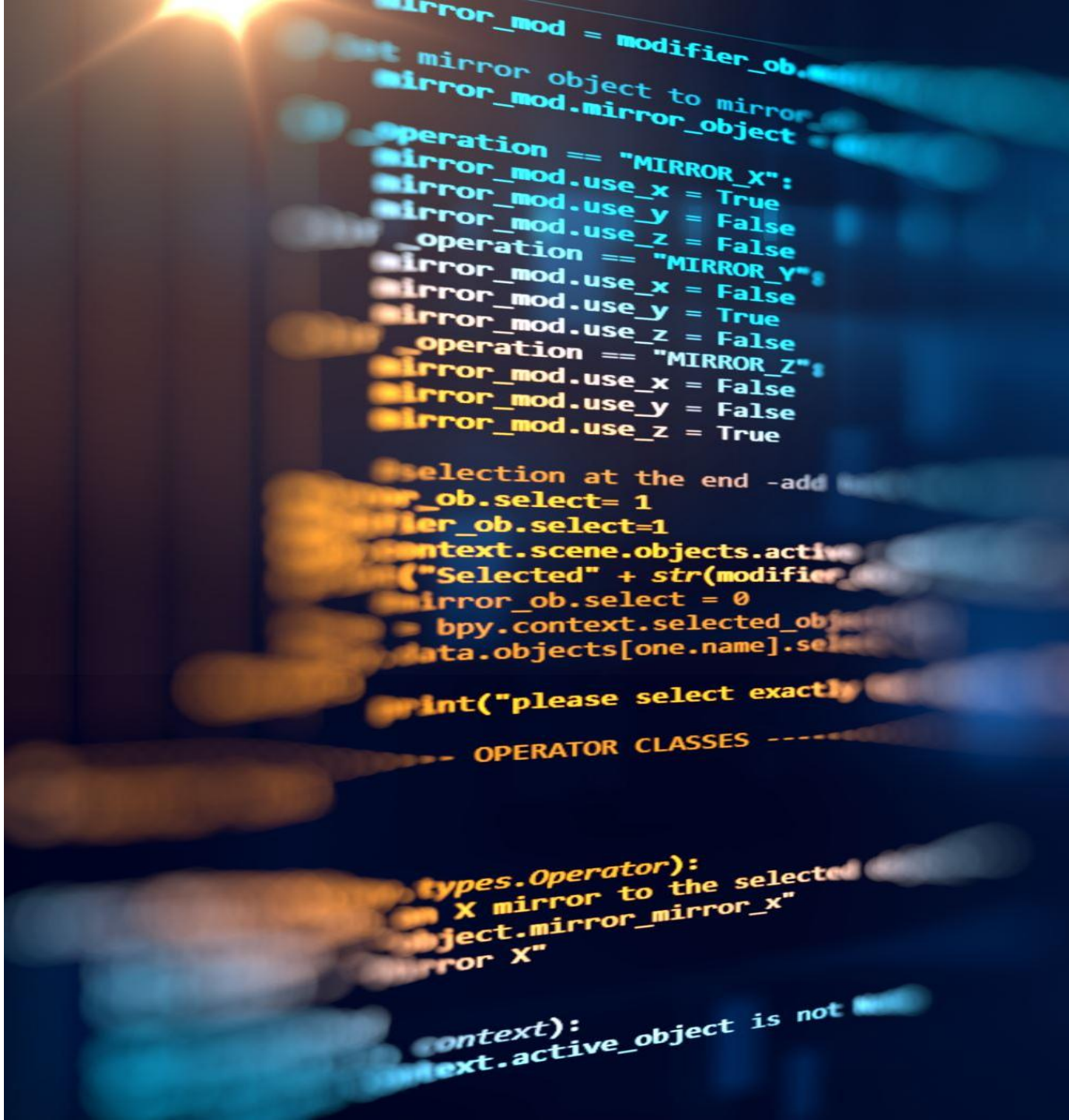
Some examples

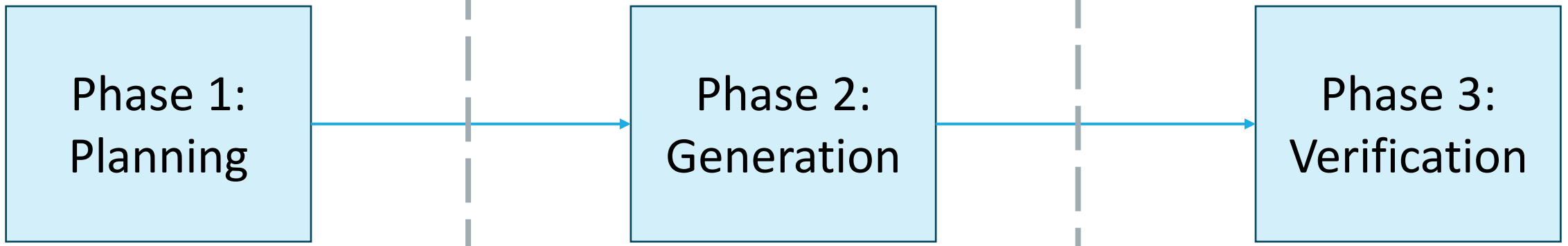
Use case	SDK or prompting?	Reason
Triage production defects	SDK	Lots of deterministic steps to get data; AI pieces things together
Pair programming/code review	Prompting	Task is unknown; task changes quickly
In-app product recommendations	SDK	AI embedded in the existing data flow
Brainstorming product names	Prompting	No tools needed; conversation is the process


Prod error log
Copilot decides additional search terms for related logs
API call to search for more related logs
Copilot analyzes logs to see what the issue is
GitHub API to clone down relevant repo
Copilot tries to find where the code is causing the prod defect
git CLI to create a new git branch
Copilot proposes a code fix in the new branch
GitHub API to create a new pull request for the new branch
Notify dev team (Slack, Teams, PagerDuty, etc.)

Demo app

- MVP generator
- Input
  - Markdown file describing the app and its requirements
  - Optional additional files for context
- Output
  - Working C# app complete with tests, UI, etc.





- 💻 Locate mvp.md
- 🤖 Generate slug
- 💻 Validate slug regex
- 💻 Create sln + copy over scripts
- 🤖 Generate implementation plan + spec files
- 💻 Validate spec count (1-10)

- 💻 Loop over each spec file
- 🤖 Implement the spec fully as code, verify tests pass + builds

- 🤖 Final pass - verify all user flows work as expected, fix if not
- 🤖 Generate README

Taking it to production

- Sandboxing
- You control the tools (no YOLO mode!)
- UX – streaming responses, wait times, etc.
- Log everything!

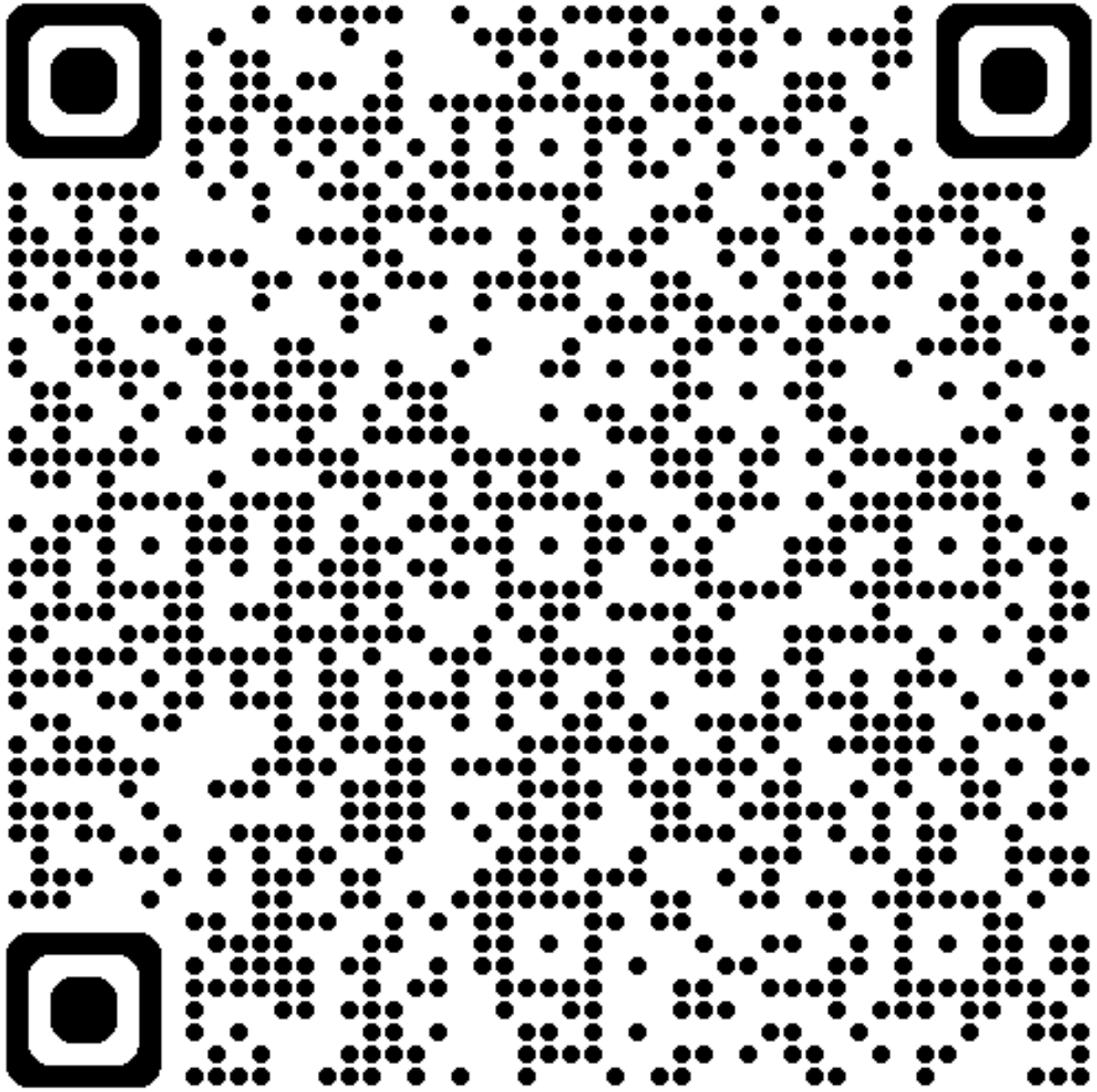
Recap

- What’s an agent?
- GitHub Copilot SDK – CLI, features, auth
- Coding demo –“10 minutes or less”
- Why use the SDK over prompts?
- Coding demo – realistic example
- Prod considerations

Resources

- https://github.com/github/copilot-sdk/
  - Getting Started
  - Lots of code samples
- https://github.com/github/awesome-copilot/tree/main/cookbook/copilot-sdk
  - Practical examples, patterns

Thank you!



daninacan.com