AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

GITHUB COPILOT

Training

Hands-on course

2025.

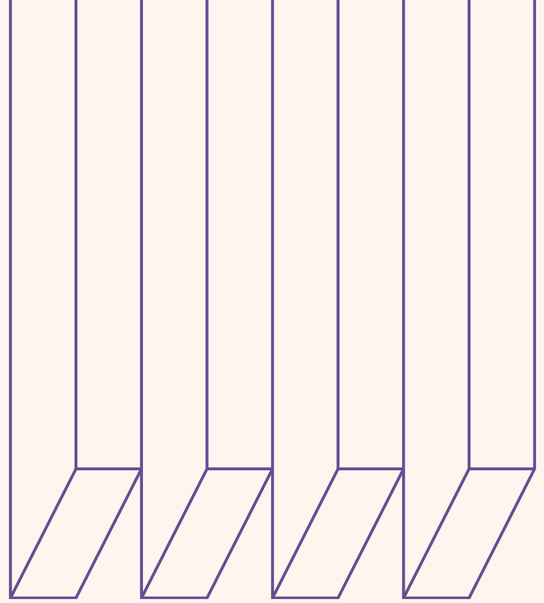
Presented by
Maria Sukhareva

Website
www.mariasukhareva.com

AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

Table of Contents

Foundations



Date: June 11, 2030

AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

AI-Assisted Coding Motivation

Multiple studies have shown that AI-assisted coding increases both productivity and the quality of code

52% developers say AI had a positive effect on productivity [1]

53,2% increased likelihood of passing all unit tests [3]

90% developers are using AI [5]

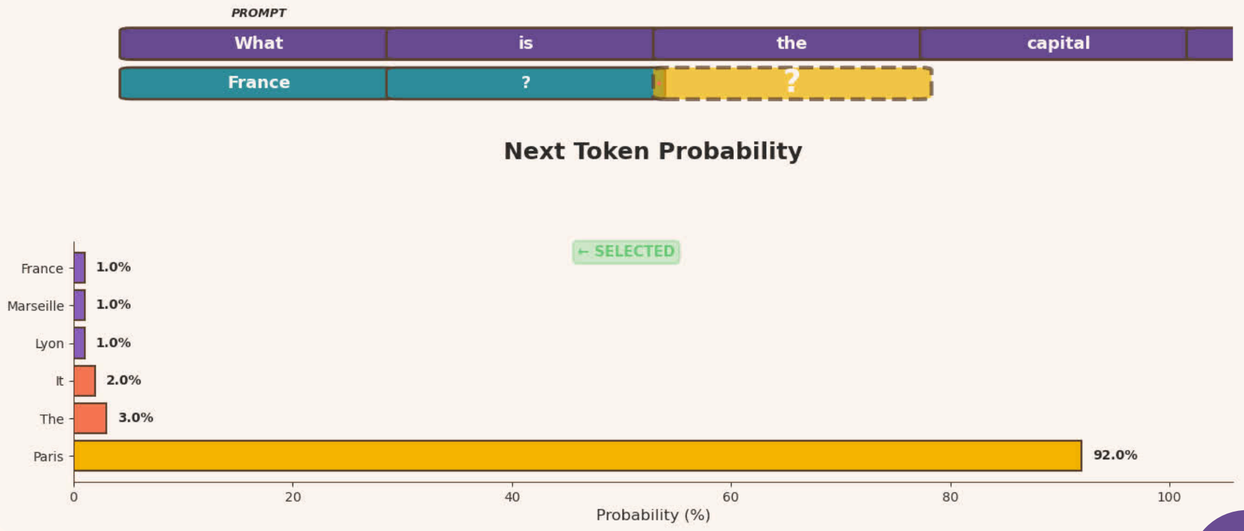
10+ hours saved per week as reported by developers [2]

~25% productivity boost [4]

References:

1. Stack Overflow Developer Survey 2025
2. Atlassian 2025 report
3. GitHub blog
4. IEEE Spectrum (AMD engineering op-ed, 2025)
5. DORA 2025

LLMs: Next Token Predictors



Large Language Models Have One Job

Predict the next Token Based On Provided Context

- LLMs do not have a communicative intent (e.g., inform, lie).
- LLMs do not reason.
- They rely on patterns and token probabilities.

> **Critics call LLMs “stochastic parrots” because they sample next tokens from learned probabilities, not understanding.**

Why LLMs Are Great For Programming Languages

AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

Artificial vs. Natural Languages

Key differences between programming and natural languages

Precision

Designed to be unambiguous so machines parse them reliably

Fixed, Immutable Grammar

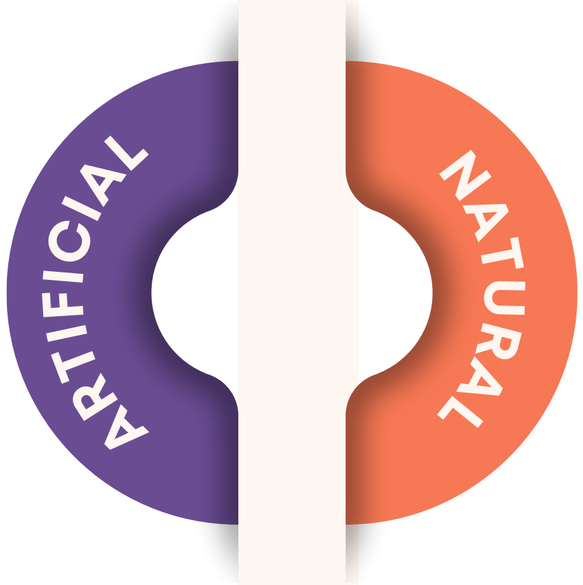
have stable, explicit grammars with (virtually) no exceptions.

Objective “Gold Standard”

has objectively correct/optimal forms that can be evaluated and tested.

Creativity Penalized

sampling randomness (“vibe coding”) quickly degrades correctness and maintainability



Ambiguity

Ambiguous on every level and rely on common sense

Evolving, Exception-ridden

evolve and teem with irregularities and exceptions

Many Valid Phrasings

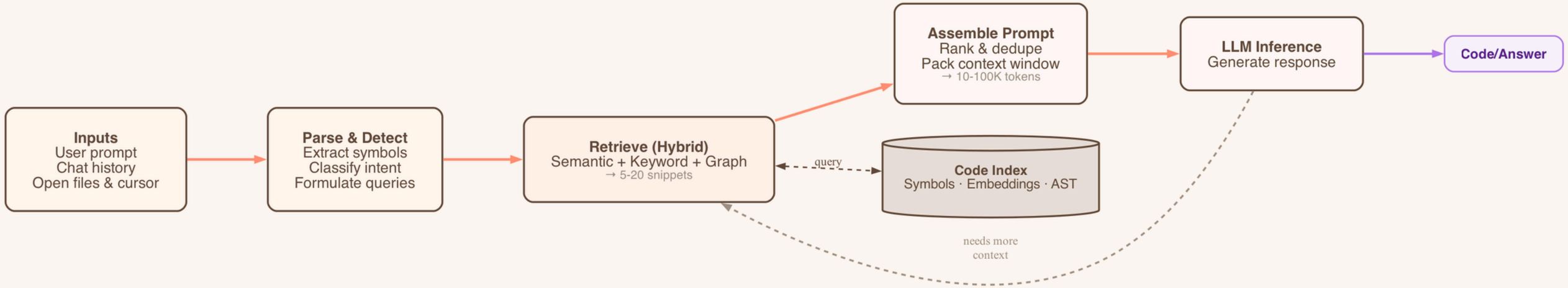
allows multiple equally acceptable ways to say the same thing

Creativity Tolerated

can be non-deterministic and varied without harm

AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

How Coding Copilots Work



Retrieval Augmented Generation

The workspace is saved into a search index

Question One

Why is RAG needed here?

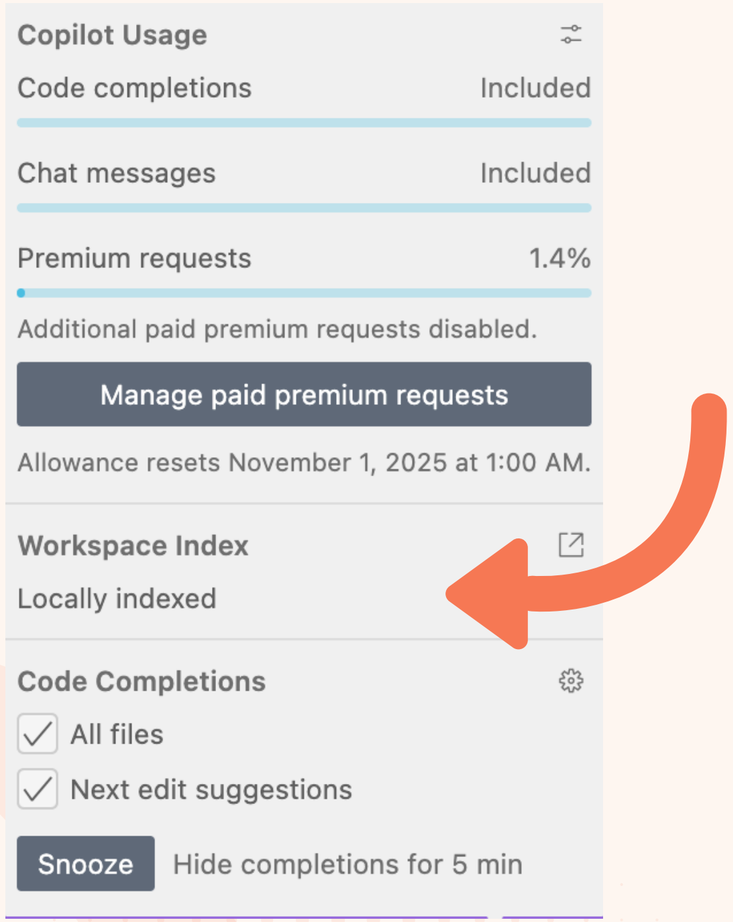
The Assembled Prompt Contains

Code snippets, chat history, graph, repo description, user query etc.

Question Two

Why is long context window of the model essential for coding assistance?

Check Indexing Status

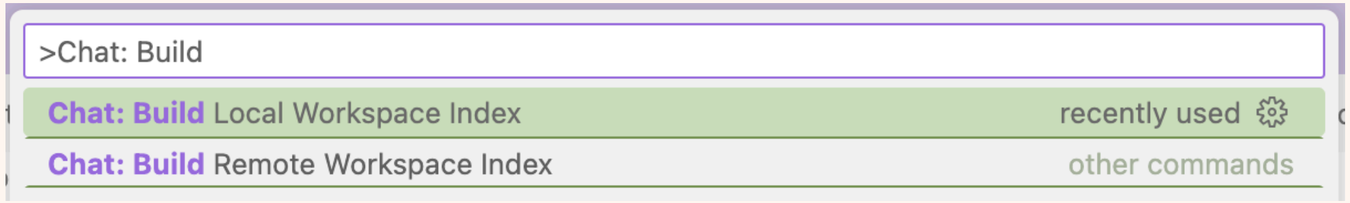


Check Status

Bottom bar → click Copilot item → Status/Details.

Create Local Workspace Index

- ⌘⇧P (mac) / Ctrl+Shift+P (win/linux)
- Chat: Build Remote Workspace Index
- Check Status Again



AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

Best Models In Github Copilot

01

Claude Sonnet 4.5

Cost: $8.26, Multiplier: 1
SWE-bench: 69.8%
Reasoning heavy, deep refactor

02

GPT-5 Codex

Cost: $2.21, Multiplier: 1
SWE-bench: 69.4%
Multi-file refactor and code review

03

GPT-5

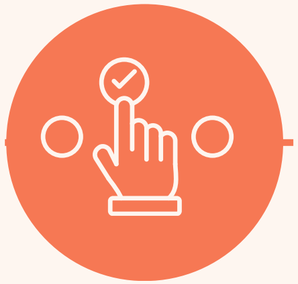
Cost: $1.41, Multiplier: 1
SWE-bench: 69.4%
Comparable with GPT-5-Codex

04

Gemini 2.5 Pro

Cost: $0.38, Multiplier: 1
SWE-bench: 55.6%
Long context precision

Claude Opus has multiplier 10!



05

Grok Code Fast

Cost: $1.96, Multiplier: 0.25
SWE-bench: 52.4%
Fast coding, big context

06

GPT-5-mini

Cost: $0.94, Multiplier: 0 (free)
SWE-bench: 59.2%
Fast coding, simple tasks

07

GPT-4.1 (default)

Cost: $0.45, Multiplier: 0
SWE-bench: 47.4%
Fast coding, simple tasks

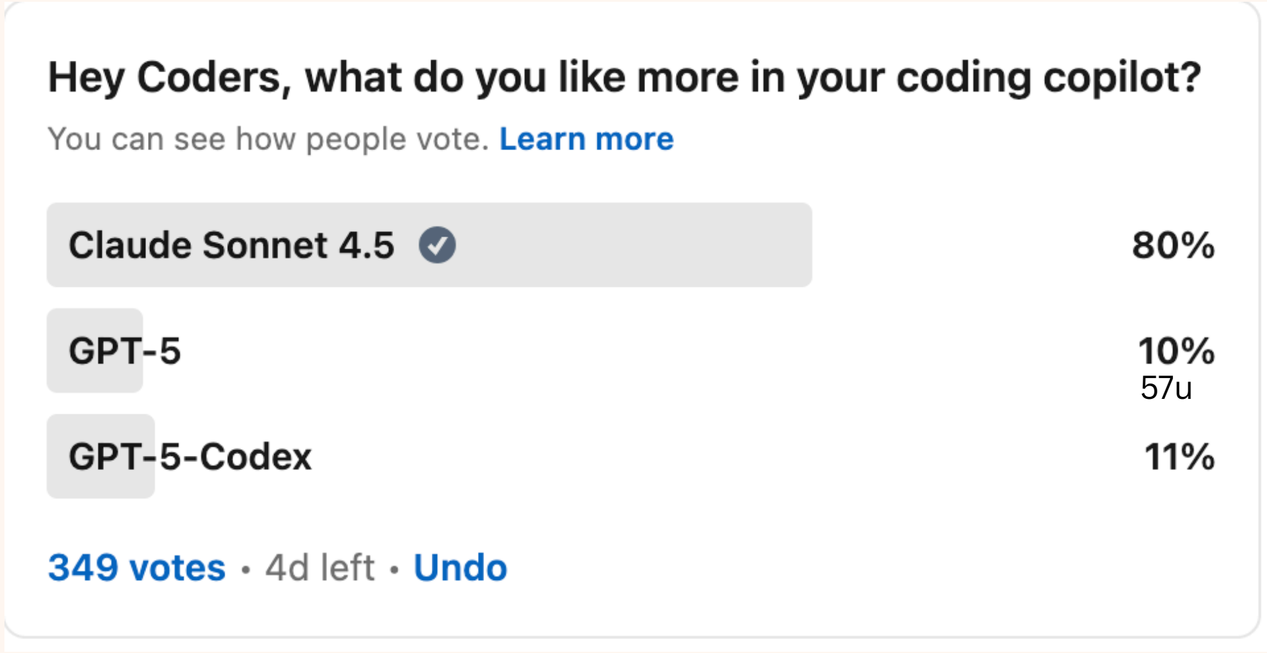
08

GPT-4o

Cost: $1.53, Multiplier 0
SWE-bench: 27.2%
Very simple tasks

Cost can be found here

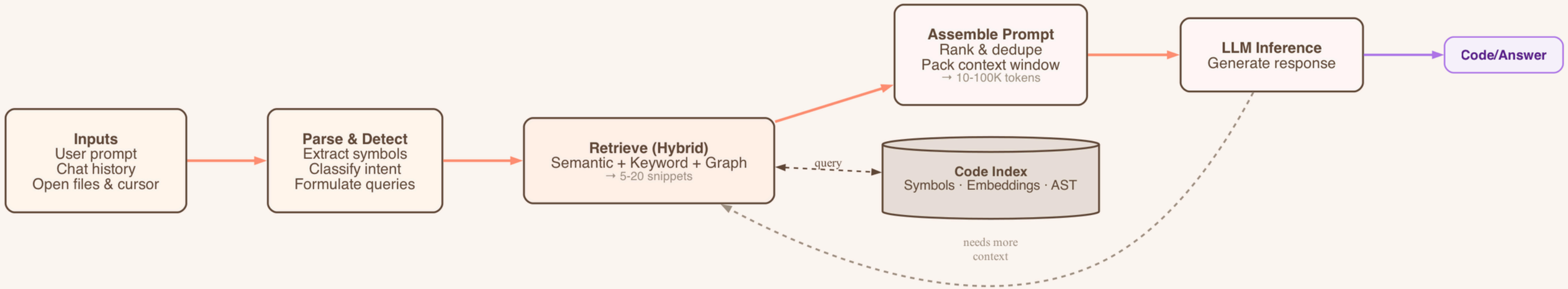
Best Models In Github Copilot



Just try and decided yourself!

Search vs. LLM

What is more important?



Search is The Key

Grounding

A short summary of the fundamental decisions and solution strategies

Task Focus

They are the basis for many other detailed decisions or implementation rules.

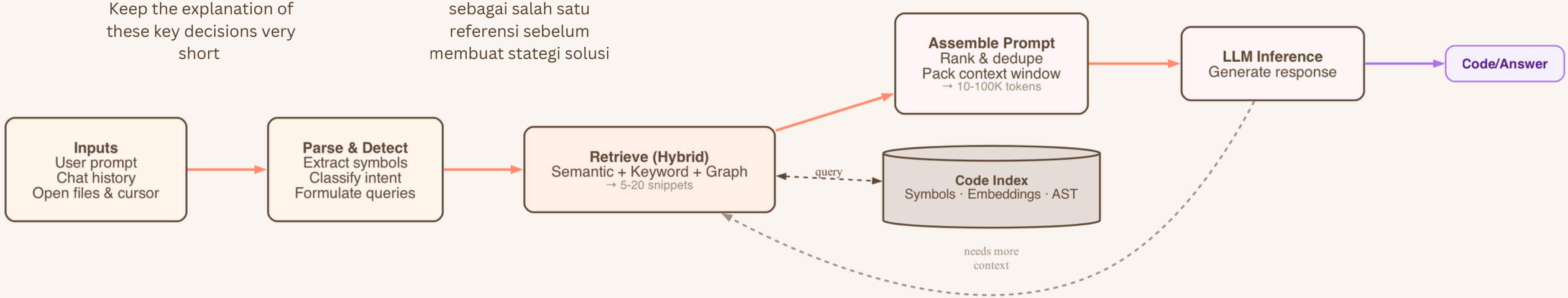
Token Economy

Keep the explanation of these key decisions very short

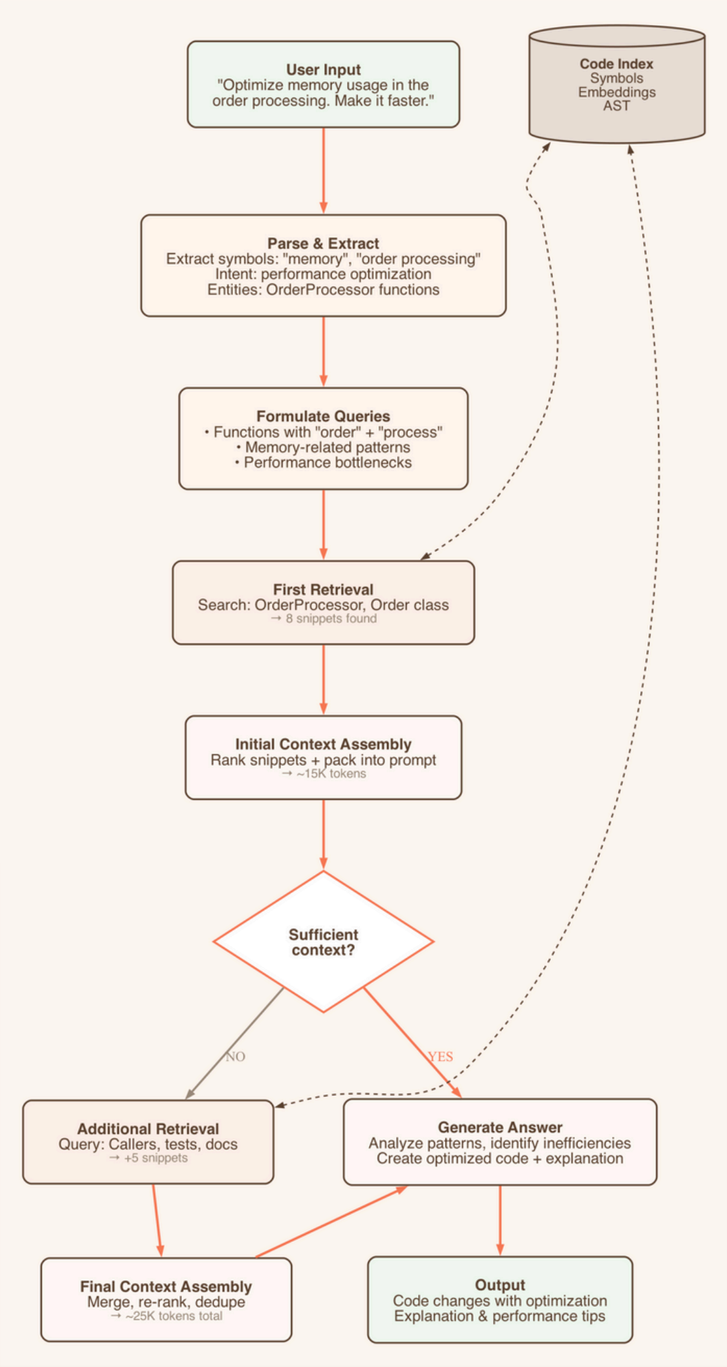
Determinism

sebagai salah satu referensi sebelum membuat stategi solusi

If search breaks or returns nothing, the LLM can't "magic" the answer.



Example 1



Prompt:

“Optimize memory usage in the order processing. Make it faster.”

Prompt parsing

Needs to guess intent, symbols, scope and acceptance criteria - pure vibe coding!

Context/search miss

Needs to find the right files and symbols in the workspace

Symbol resolution drift

Needs to figure out which processOrder to change, might be confused by dynamic inputs or name collisions

Tool execution failures (build/test/lint)

It needs to figure out which tool to execute and how to test that the task is implemented

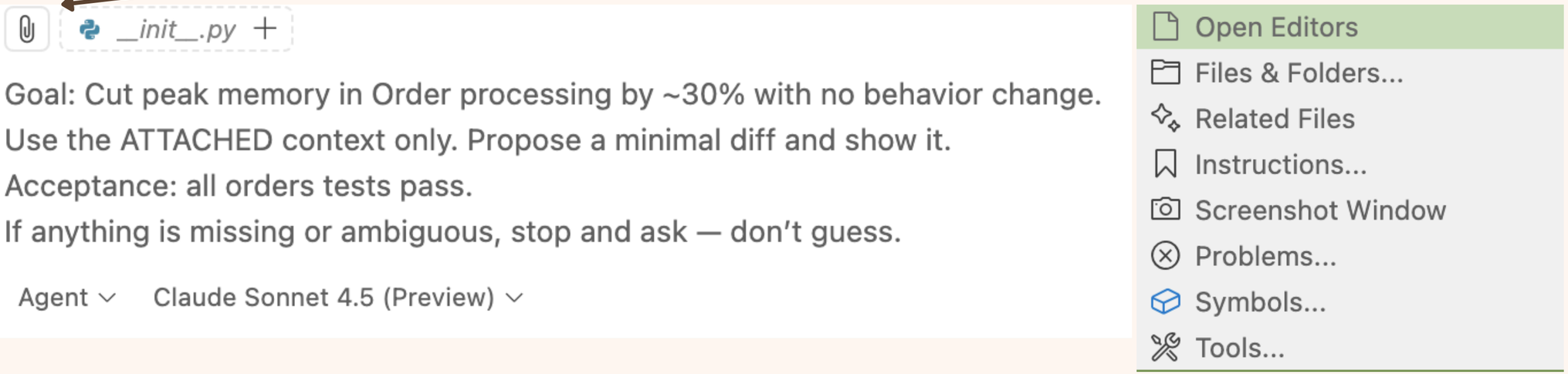
Example 2: Prevent Vibe Coding!

"Goal: Cut peak memory in Order processing by ~30% with no behavior change.
Use the ATTACHED context only. Propose a minimal diff and show it.
Acceptance: all orders tests pass.
If anything is missing or ambiguous, stop and ask — don't guess."

Rule of Thumb:

- Avoid ambiguity
- State the goal
- Name acceptance
- Force model to ask and prevent guessing!
- Configure context such as tools, symbols, files etc

Add context



GUI Changes, Principals Stay The Same

Simplify search by adding context

Shorten the prompt; attach the facts (files, selections, symbols, docs).

Clean the context

Make sure the chat history is relevant, correct context is needed, cursor is at the right place

Prevent the LLM from guessing

Tell it what to do and when to ask.

Keep changes small; verify early

Prefer minimal diffs and short loops.

Don’t overthink prompt engineering

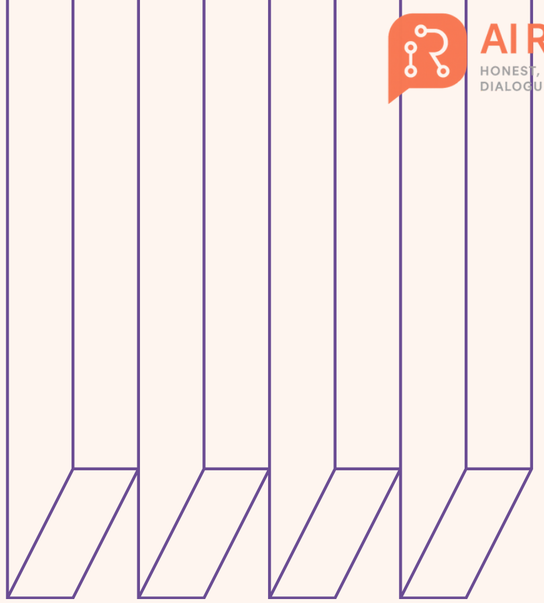
Use a tiny template; let context carry the weight.

Acceptance gates

Encourage self-correction and self-reflection

AI Realist
HONEST, DATA-DRIVEN DIALOGUE ABOUT AI

Automation Paradigm & Tooling



Date: June 11, 2030

AI Changes The Paradigm: Machines Work, Humans Observer

Learn to delegate actions to AI

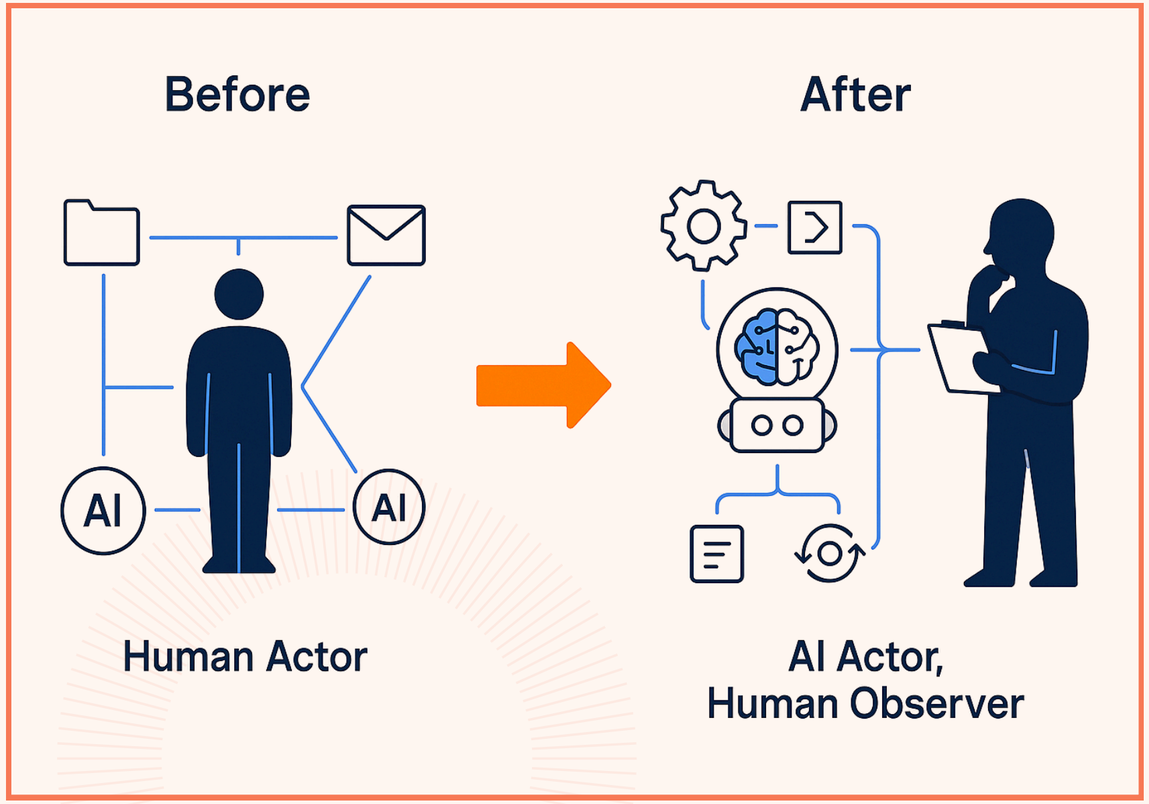
The more you successfully delegate the better

Learn to observe

Evaluate, consulte, guide and ensure the quatlity

Learn to increase automation level

The goal is to move as far as possible in the degree of automation



Will We All Get Fired?

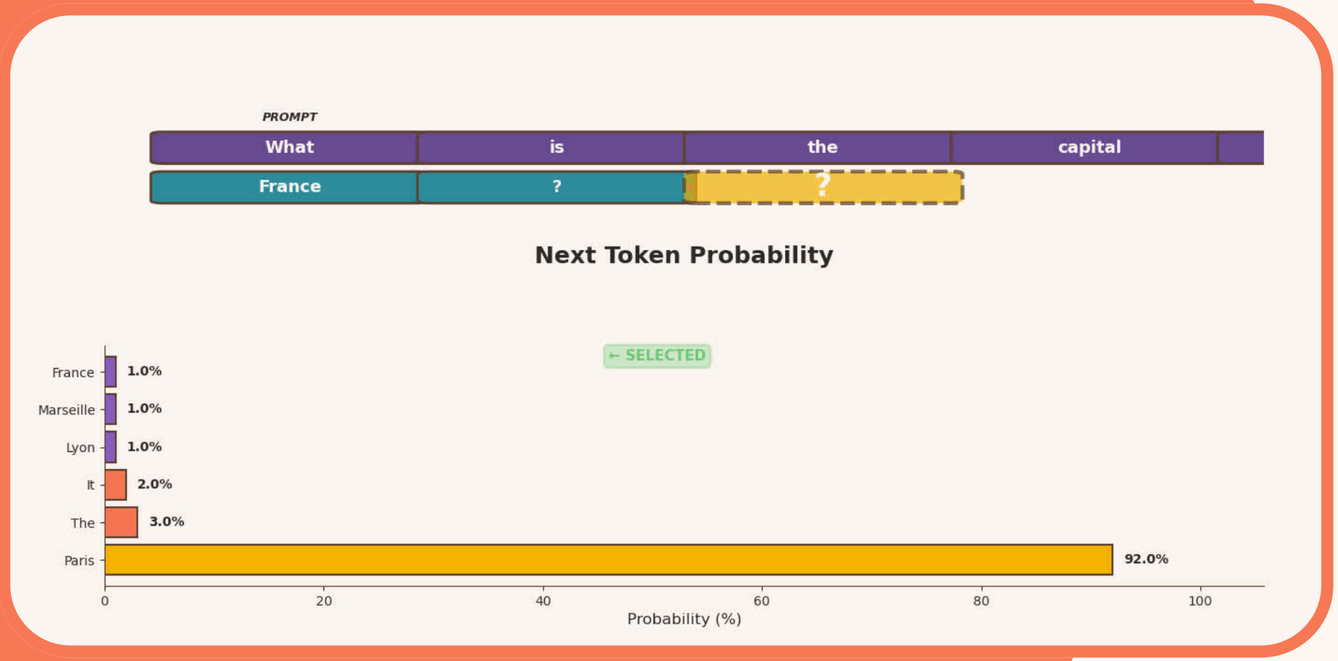
GitLab

Bill Staples
CEO of GitLab

“Imagine what can happen when we double, triple, quadruple, 10x the number of people who can now create software through natural language.”

Coding LLMs: Tool Usage Prediction

The latest models starting from o1 series, Claude 3.5, Gemini 2.0 are explicitly trained for tool selection



Modern LLMs have two jobs: predict next word and tools

Thought-Action-Observation Cycle

How AI Agents are Defined and Trained

01 Plan

Decide goal+minimal approach (which symbols, files, what to change)

02 Action

Call a tool (search repo, make diff, build, run tests)

03 Observe

Read tool output (snippets, logs, errors, test results)

04 Verify

Check against acceptance (build ok? test green? policy met?)

It used to be a prompt-based approach;
The latest trend is that model providers now train this loop into the model.

What Training Data Look Like


```
"issue_title": "Rounding error: 19.995 becomes 19.00 instead of 20.00",
"initial_tests": "failing",
"trajectory": [
  {
    "step": 1,
    "think": "Reproduce the failure to see the exact assertion and stack trace.",
    "act": { "tool": "pytest", "args": ["-q"] },
    "observe": {
      "exit_code": 1,
      "stdout": "F..F...\nE   AssertionError: expected 20.00, got 19.00\nFAILED tests/t
      "stderr": ""
    }
  },
  {
    "step": 2,
    "think": "The failure is in pricing. Open the suspected module to inspect rounding.'
    "act": { "tool": "open_file", "args": ["discounts/pricing.py"] },
    "observe": {
      "content_head": "def round_price(x: float) -> int:\n    # TODO: simplify rounding'
      "lines": 120
    }
  },
  {
    "step": 3,
    "think": "int(x) truncates instead of rounding. Replace with round-half-up behavior
```


The agent is trained to generate trajectories:

- think
- act
- observe

The more context it gets that fit this pattern e.g. it is told how to act, which context to look at, what to observe, the less iterations it does

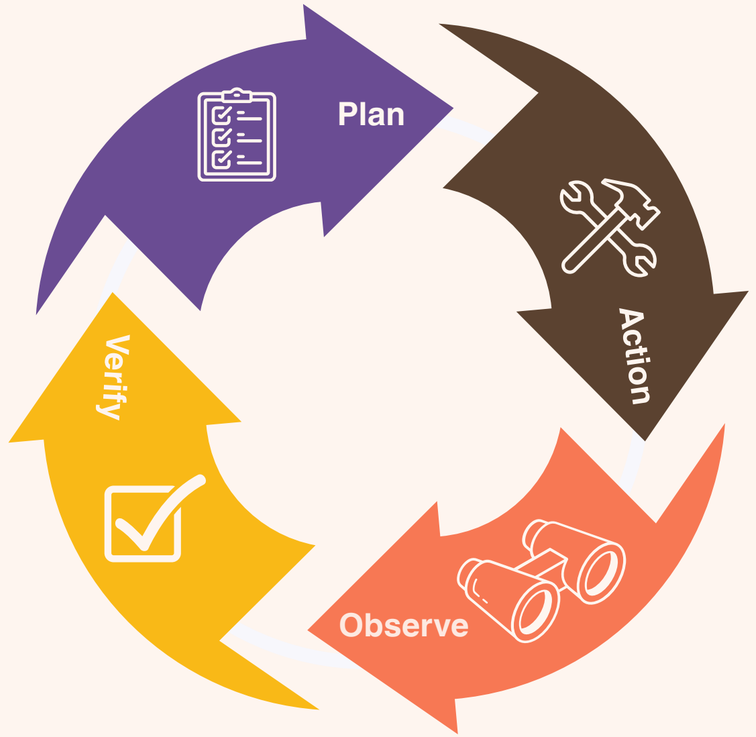
Coding Agents

Not just a theory — this is how modern coding models are trained/aligned

AI coding agent

A program that uses a large language model (LLM) to plan, act, observe, and verify when making changes to software:

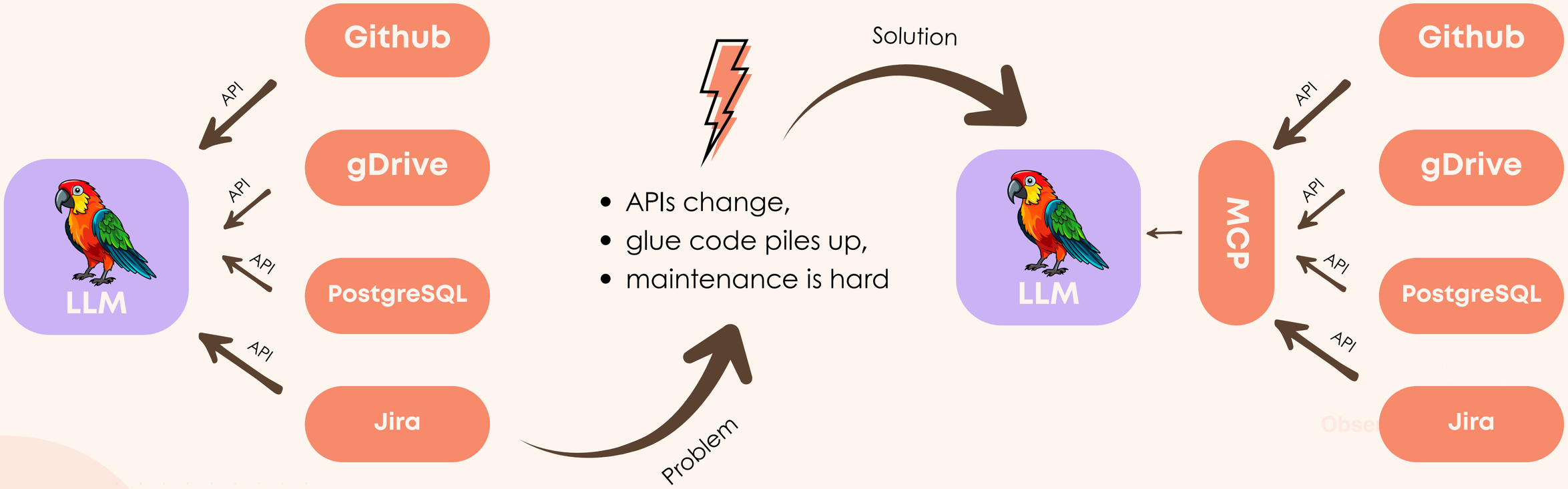
- **Plan**: derives the goal, outlines steps, and sets acceptance criteria (e.g., tests must pass, performance target).
- **Act**: chooses and invokes concrete tools (e.g. repo search, edit/patch, compiler, test runner).
- **Observe**: reads and interprets the environment's feedback (e.g. tool outputs, logs, errors, code diffs, runtime results).
- **Verify**: checks results against the acceptance criteria (e.g build/test pass, style/policy met, perf budgets)



Behind the scenes, **every modern model runs a plan** → act → observe → verify cycle.
Reduce guessing: attach precise context and state clear acceptance tests.

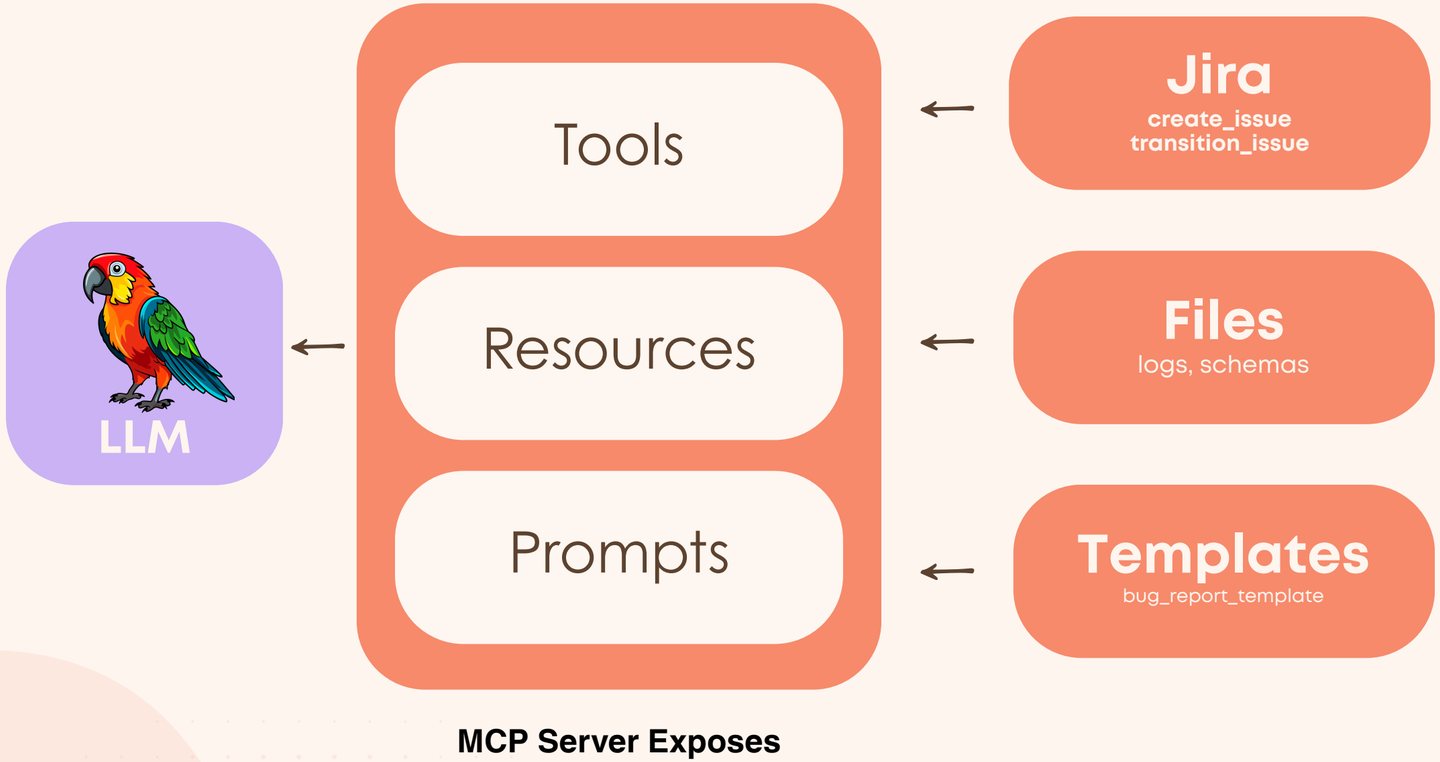
MCP Servers

Unified way to bring context and tools to LLMs



MCP Servers

Unified way to bring context and tools to LLMs



GitHub Copilot only supports tools

Workaround:

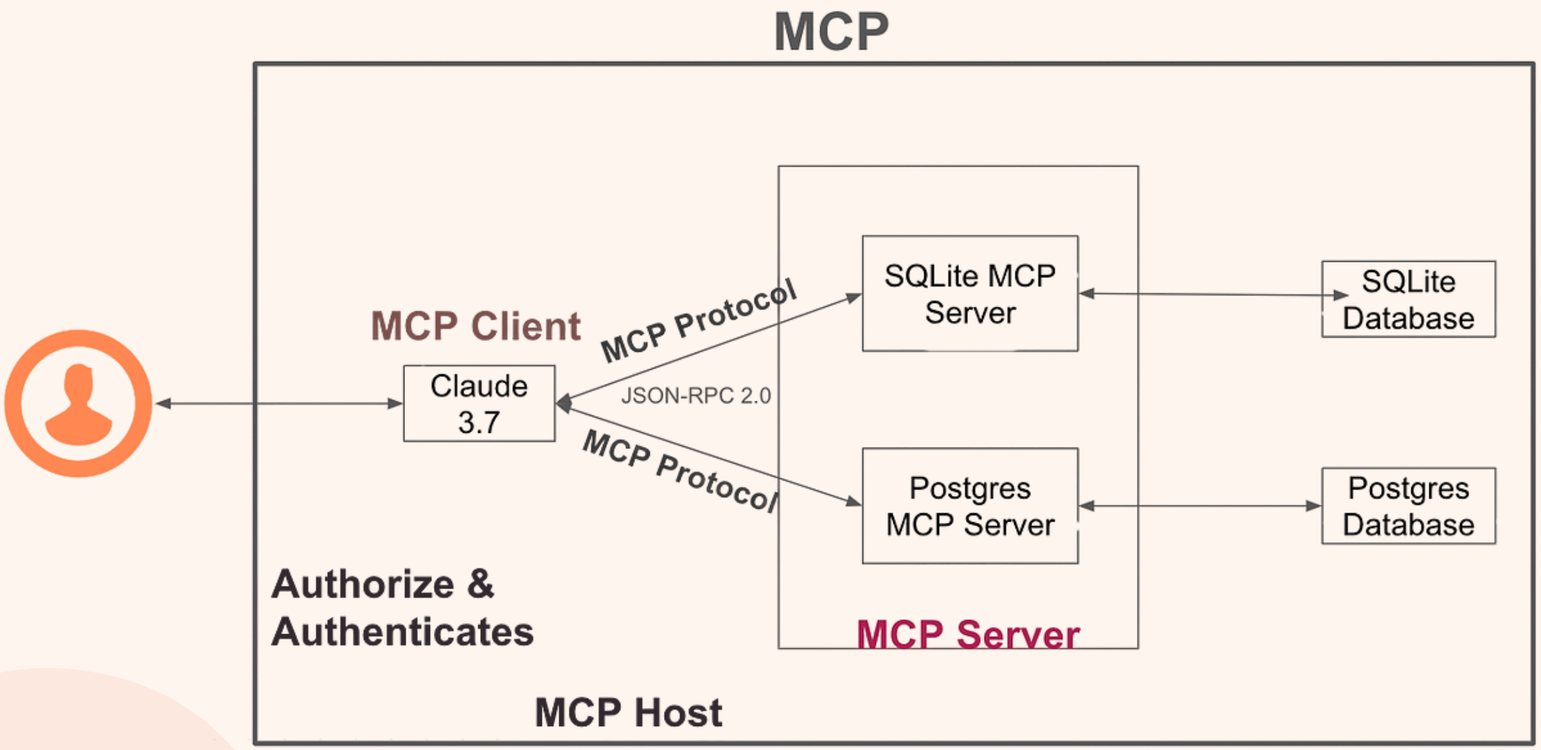
- wrap read-only context as tools (e.g., read_logs, fetch_schema)
- wrap reusable prompt templates as a tool (e.g., get_prompt_template({name, vars})).

Best Practices:

- **In-repo:** Add context (files, selections, symbols) with Copilot's native controls.
- **Off-repo:** Use MCP tools to fetch external schemas, logs, APIs, etc.

MCP Servers

Unified way to bring context and tools to LLMs



JSON-RPC 2.0

Remote Procedure Call protocol using JSON

- **Every custom MCP server MUST implement a JSON-RPC endpoint that Copilot can call.**

JSON-RPC Methods

initialize
handshake with client

tools/list
list available tools

tools/call
Execute a tool

shutdown
Clean shutdown

Must **Should**

MCP Servers

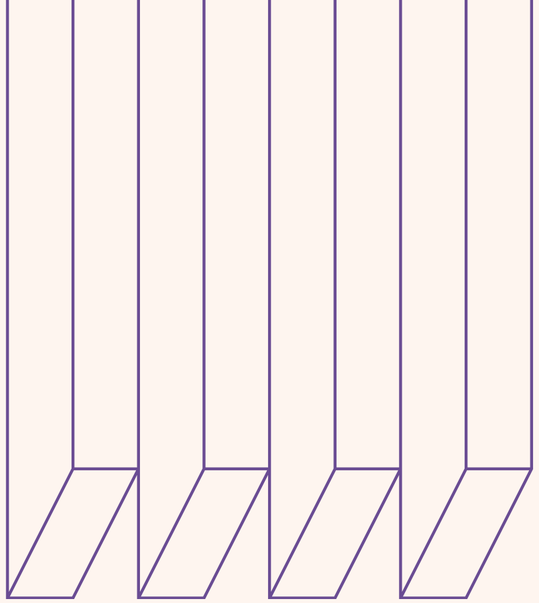
Unified way to bring context and tools to LLMs

```python
@app.post("/mcp")
async def mcp(request: Request):
    payload = await request.json()
    method = payload.get("method")
    rpc_id = payload.get("id")

    if method == "initialize":
        return {"jsonrpc": "2.0", "id": rpc_id, "result": {
            "protocolVersion": "2024-11-05",
            "capabilities": {"tools": {}},
            "serverInfo": {"name": "my-server", "version": "1.0"}
        }}

    if method == "tools/list":
        return {"jsonrpc": "2.0", "id": rpc_id, "result": {
            "tools": [
                {
                    "name": "my_custom_tool",
                    "description": "Does something amazing",
                    "input_schema": {"type": "object", "properties": {}}
                }
            ]
        }}

    if method == "tools/call":
        tool_name = payload["params"]["name"]
        args = payload["params"]["arguments"]

        if tool_name == "my_custom_tool":
            # Your business logic here
            result = do_something_amazing(args)
            return {"jsonrpc": "2.0", "id": rpc_id, "result": {
                "content": [{"type": "text", "text": result}]
            }}

    return {"jsonrpc": "2.0", "id": rpc_id, "error": {
        "code": -32601, "message": "Method not found"
    }}
```

Implementation Checklist

- [ ] JSON-RPC endpoint (POST /mcp)
- [ ] Handle initialise method
- [ ] Handle tools/list method
- [ ] Handle tools/call method
- [ ] Define your custom tools
- [ ] Implement tool business logic (what it actually does)
- [ ] Return proper JSON-RPC responses
- [ ] Configure VS Code MCP settings

Practical Lab



Date: June 11, 2030

AIRLINE DISCOUNT ML PROJECT

Check VENV FIRST!!!!!!!!!!

Synth CLI Tool

⚡ Generate Data

📄 generated_data.json
"500 passengers"
"25 routes"
"500 discounts"

Database

🗄️ SQLite Database
airline_discount.db

📊 passengers: id, name, travel_history JSON

🛫 routes: origin, destination, distance_km

💰 discounts: passenger_id, route_id, discount_value

Foreign key constraints enforced

ML Pipeline

🎯 Goal: Predict the discount

🔎 **Pattern Discovery**
Passengers with 10+ trips
averaging $500/trip
flying 3000+ km routes
typically receive 18–25%

📈 Feature Engineering — passenger_profiler.py
• build_features()

🤖 ML Model Training — discount_predictor.py

MCP Integration

🔌 Custom MCP Server (FastAPI)

🔧 synth_generate
🔍 query_db
📚 train_model
🎯 predict
📊 evaluate

💻 VS Code – GitHub Copilot Chat + MCP Tools

4 custom tools for Copilot

Airline Discount ML Project

1. **Predicts airline discount percentages for passengers based on travel history and route data**
2. **Uses synthetic data only (SQLite database) - no real customer information**
3. **Implements a complete ML pipeline: data → features → model → predictions**

EXERCISE ONE
SETUP

Set up GitHub Copilot Instructions
Install repo-wide and path-scoped

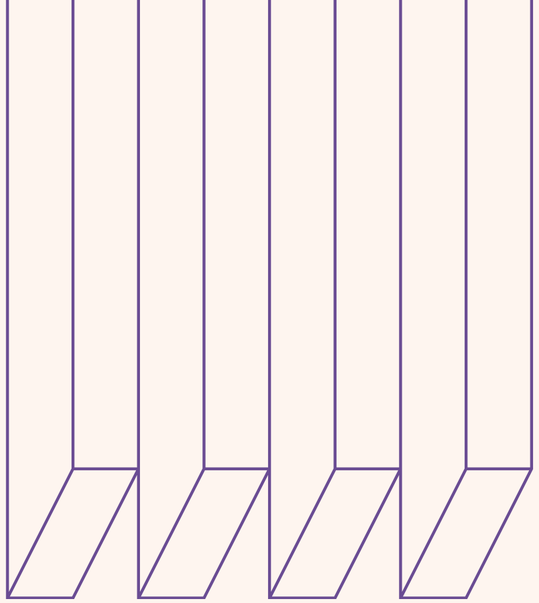
Enforce "ask, don't guess" behavior
Configure Copilot to request clarification

Use path-scoped instructions with applyTo globs
Apply different coding rules to specific parts of your codebase

Verify instruction loading in VS Code
Test that Copilot correctly reads and follows both instruction layers by checking

Generate minimal, targeted diffs
Practice using attached context (selections, symbols, files)

Governance and Scale-Out



Date: June 11, 2030

MCP Servers For Team Collaboration

Remote Deployment and Repo-scoped Config

Knowledge Sharing

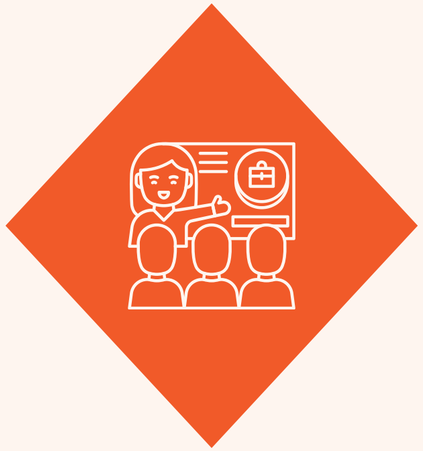
MCP Servers shared across teams allow to share prompts, resources, tool implemenations

Unified Setup

Every member of the team will have the same setup in seconds

Governance

ownership, audit, and access control in one place



Faster Onboarding

Shared MCP servers give new teammates instant access to the same prompts, resources, and tools the rest of the org uses

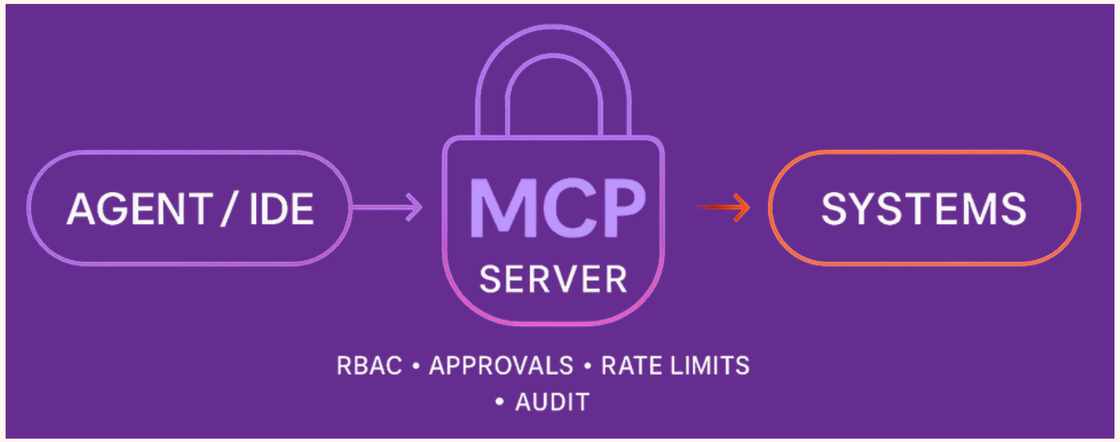
Why Governance for MCP

Put controls where the power is - on the server.

Client allowlists ≠ security: enforce policy server-side

Scale needs repeatability: least-privilege access, standardized tool contracts, consistent rollouts.

Auditability = trust: log who/what/when; measure cost, latency, and errors.



Empower teams without risking data, cost, or uptime.

We don't rely on the client's allowlist—governance lives at the MCP server with roles, approvals, caps, and audit so we can move fast safely.

From Local to Team-Wide:

Remote MCP Server Deployment

Works Locally

Your MCP server works locally (http://127.0.0.1:8010)

No Team Work

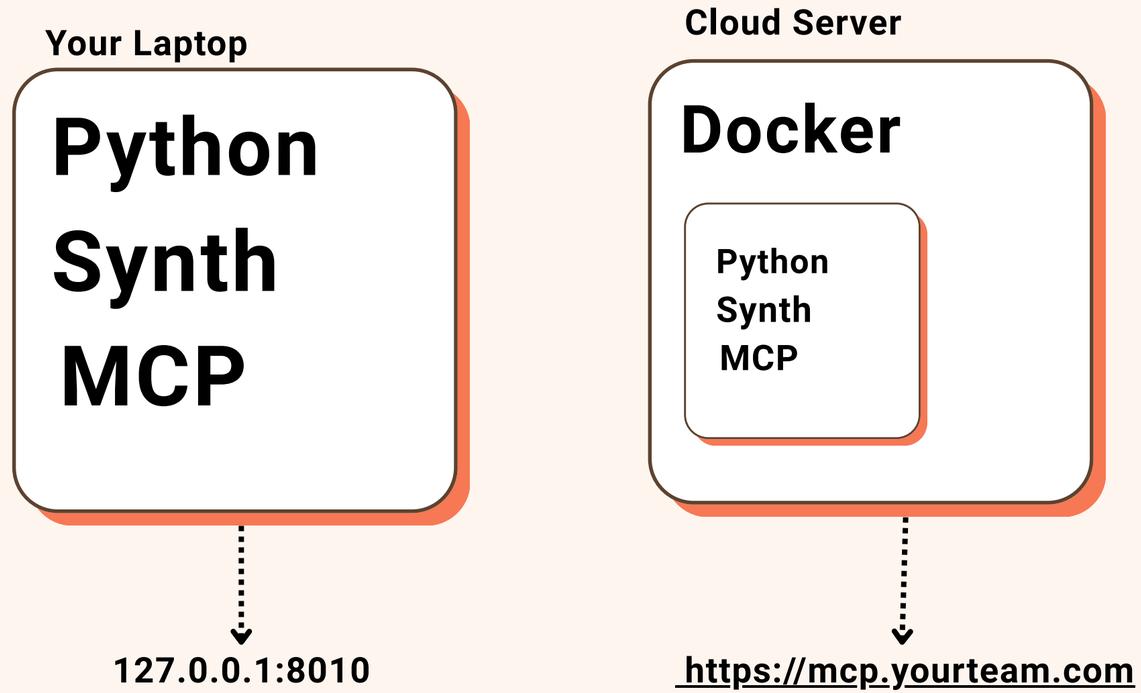
Each teammate needs their own setup

Device-specific errors

"Works on my machine" syndrome

Manual Work

Manual installations of Python + Synth + dependencies



Minimal MCP Governance

Best Practices

```
{
  "servers": {
    ▷Start | More...
    "github-mcp": {
      "url": "https://api.githubcopilot.com/mcp/",
      "type": "http"
    },
    ⊗Error | Restart
    "synth-local": {
      "type": "http",
      "url": "http://127.0.0.1:8010/mcp",
      "toolAllowList": ["synth_generate", "synth_inspect_model", "preview_tal
    }
  },
```

No authorisation for the server of synth-local

anyone can just call the server if they know the API

Unrestricted GitHub MCP

you might want to configure which tool permissions users have for your workspace github

No environment separation or secret handling

Hardcoded URLs as well as no separation of dev/stage/prod.

Client allowlist ≠ security

"toolAllowList" only hides tools in the UI but does not enforce on the server. You need server-side allow/deny, size/time caps, approvals, and rate limits..

Minimal MCP Governance

Best Practices

```
{
  "servers": {
    "github-mcp": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp/",
      "toolAllowList": ["list_issues", "create_issue"]
    },
    "synth-remote": {
      "type": "http",
      "url": "${env:SYNTH_MCP_URL}/mcp",
      "headers": { "Authorization": "Bearer ${env:SYNTH_MCP_TOKEN}" },
      "toolAllowList": [
        "synth_generate",
        "synth_inspect_model",
        "preview_table_head"
      ]
    }
  }
}
```

No authorisation for the server of synth-local

anyone can just call the server if they know the API

Unrestricted GitHub MCP

you might want to configure which tool permissions users have for your workspace github

No environment separation or secret handling

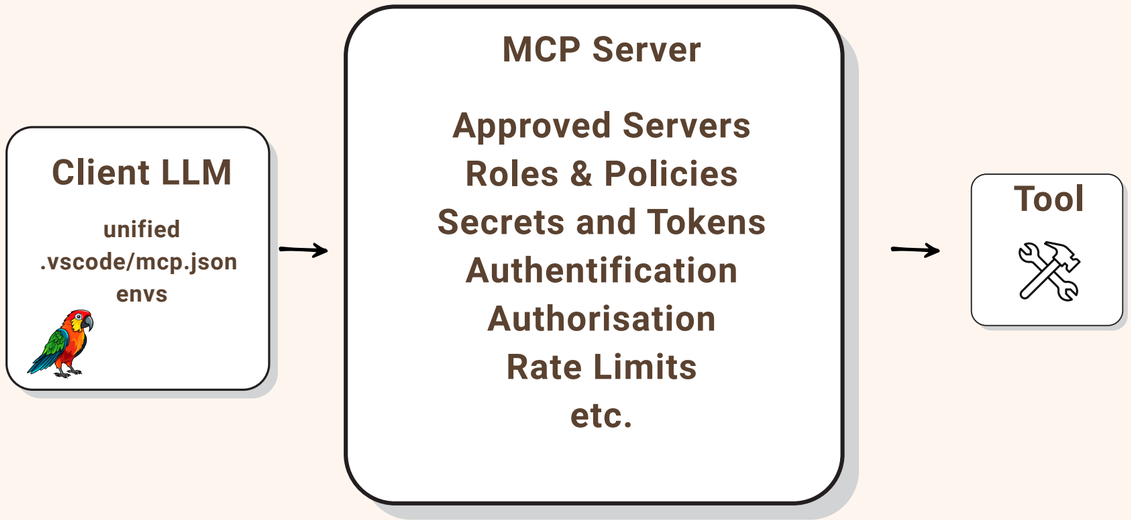
Hardcoded URLs as well as no separation of dev/stage/prod.

Client allowlist ≠ security

"toolAllowList" only hides tools in the UI but does not enforce on the server. You need server-side allow/deny, size/time caps, approvals, and rate limits..

Team Collaboration

Best Practices



Create unified mcp.json
Enforce roles and limits on the server
Share mcp.json and env.template with the team

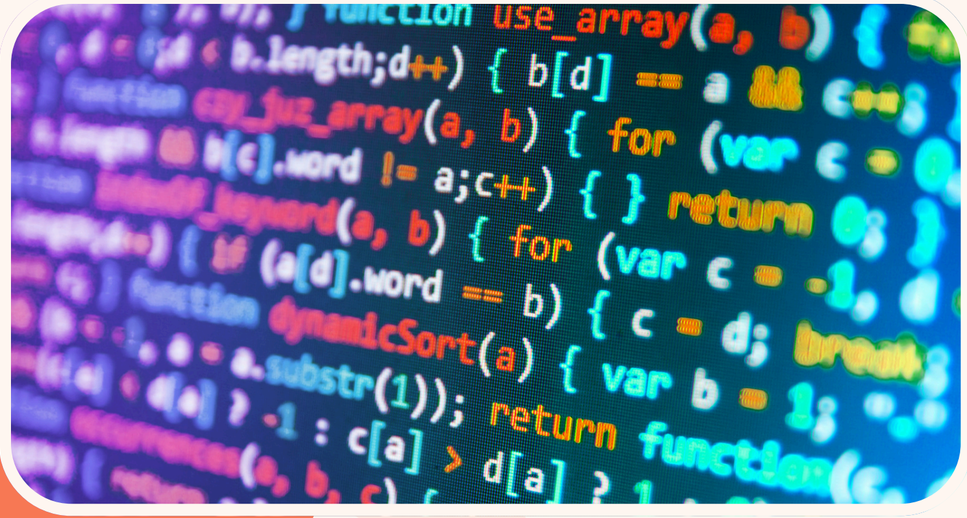
Create unified mcp.json
Enforce roles and limits on the server
Share mcp.json and env.template with the team

Conclusion

Coding Copilots, MCP servers etc. are still being developed and evolve significantly. Focus on understanding principals and not on UX.

Covered Topics:

- How to ground copilot in context
- How to create and use MCP servers
- How to generate unit tests
- How to govern MCP servers



Thank You

For Your Attention

LinkedIn
msukhareva

Email
info@mariasukhareva.com

Website
www.mariasukhareva.com

Presented by **Maria Sukhareva**