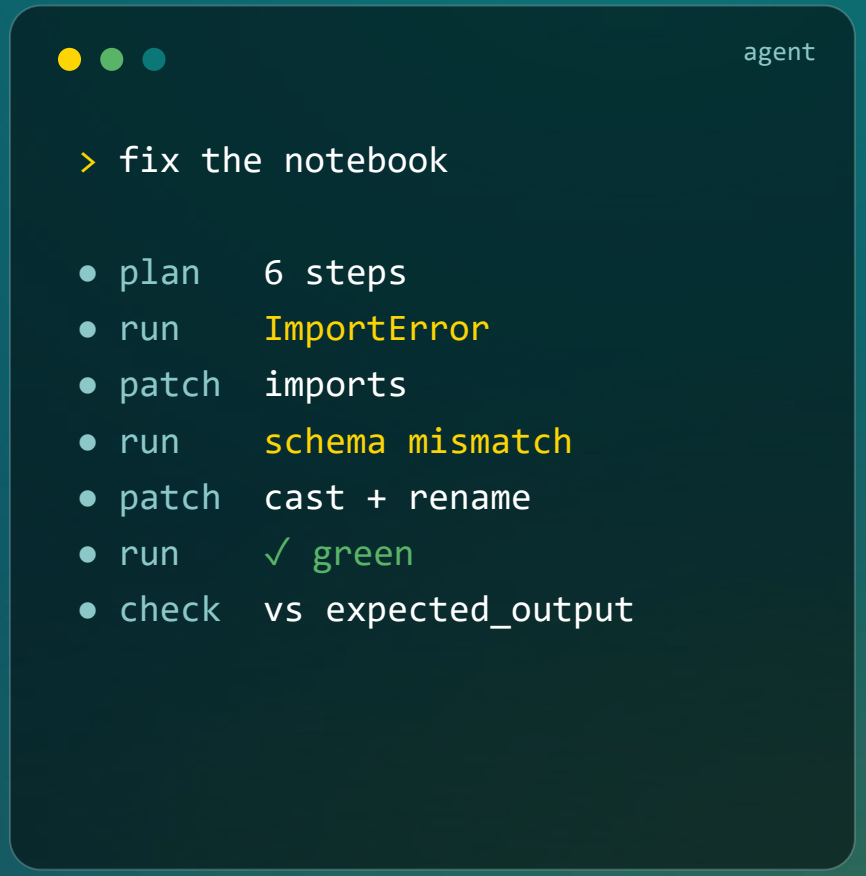
Data Brothers

// [Data Saturday Holland]

# Why your data team needs an AI agent (not a chatbot)

Live demo on Microsoft Fabric. No vision deck. No hype.




**Štěpán Rešl** · DataBrothers

// SPEAKER

# Štěpán Rešl

Co-founder / Lead technical consultant @ DataBrothers

Data Brothers · Microsoft MVP Most Valuable Professional · Power BI kafíčko · Data Meerkat

LINKEDIN

X

BLUESKY



# Thank you sponsors!

INSPARK

twintos

ilionx

InfoSupport
Solid Innovator

redgate

sopra steria

DATA GIANTS

BRANDER
company

SQL brander.nl
DATA brander.nl
CRM brander.nl
ERP brander.nl

Bull
Master your AI destiny

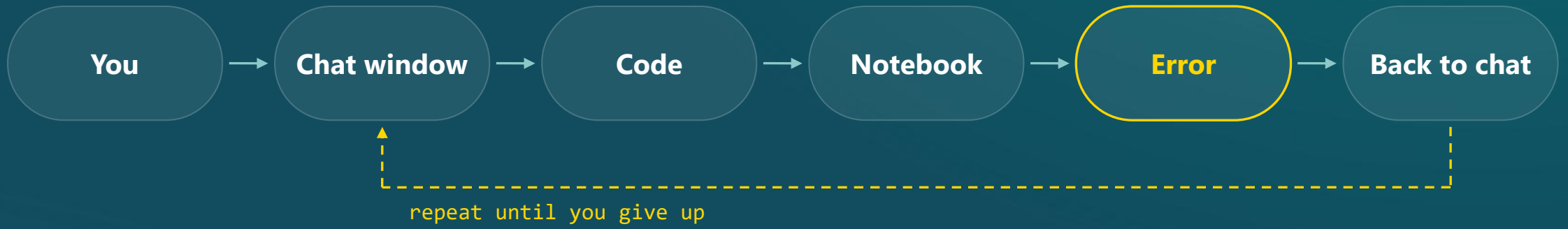
dbWatch

BBTG

Data Saturdays

// THE PROBLEM

# What 'AI at work' usually looks like




***The copy-paste loop.***

Show of hands: who does this daily?

// WHY IT BREAKS

# Four friction points



**01**

## No memory

Every tab is a stranger. Re-explain context every time.

**02**

## No tools

It cannot run your code. Only describe it.

**03**

## No iteration

It cannot see the error. You are the feedback loop.

**04**

## No context

It does not know your schema, conventions, or business rules.

// THE COST

# What this costs you per week

| | | |
|---|---|---|
| **15 min** | × 8 debug cycles | = 2 h / day |
| **5 min** | × 10 re-explaining context | = 50 min / day |
| **10 min** | × 4 'where did I leave off' | = 40 min / day |

TOTAL

**~17 h**

per developer, per week

*That's a conservative estimate. Now multiply it by everyone who raised their hand.*

// THE GAP

# Two mental models, one chasm

CHATBOT

## Text in, text out

- One turn at a time
- No tools
- No memory of project
- You close the loop

VS

AGENT

## Plan, act, observe, iterate

- ✓ Many turns, one goal
- ✓ Tools (file IO, shell, MCP)
- ✓ Project memory
- ✓ Closes its own loop

// MENTAL MODEL

# What is an agent

An LLM in a loop that can




toward a goal.

## Loop. Tools. Goal.

```
No loop  =  chatbot        No tools  =  LLM        No goal  =  drift
```

// MENTAL MODEL

# Four pillars

**01**

## Plans

Decomposes the task

**02**

## Tools

Turns text into action

**03**

## Memory

Knows your project

**04**

## Iteration

Closes its own loop

*90 seconds on each. Stay with me.*

// PILLAR 1

# Plans

## Agent decomposes the task before acting.

Each step has a verifiable result. A chat window generates once. Agents create checkpoints.

```
goal: "fix the notebook"
```

1. Read notebook
2. Identify errors
3. Fix imports
4. Validate schema
5. Rerun
6. Check output against expected

// PILLAR 2

# Tools

Tools turn text into action.

| | |
|---|---|
| **File IO** | `read, write, edit, list` |
| **Shell** | `run any command, see output` |
| **Code exec** | `Python, SQL, notebooks` |
| **MCP servers** | `Fabric, Power BI, GitHub, Outlook, Slack` |

**A chat window describes your code. An agent runs your code.**

// PILLAR 3

# Memory

Three tiers, ordered by lifespan.

| | | |
|---|---|---|
| **CLAUDE.md / AGENTS.md** | Schemas, conventions, business rules. Loaded every session. | per repo |
| **Skills** | Playbooks and reference. Loaded only when needed. | on demand |
| **Conversation** | What we just did. Resets per session. | per session |

***The project CLAUDE.md is the highest-leverage file you can write.***

// PILLAR 4

# Iteration

## Observe. Act. Observe.

The agent closes its own loop.

***This is why we earn the name 'agent'. The agent reads its own errors.***



// HONESTY

# What an agent is NOT

## Not autonomous

You still review and approve. The agent is a force multiplier, not a replacement.

## Not infallible

It makes confident mistakes. You must verify. Especially in production paths.

## Not magic

Bad context in, bad output out. The project memory determines quality.

***Buy back trust before you ask for it.***

// DEMO

# The 4PM handover

# Friday. 4 PM.

***Recognize this?***

**SA** **Senior analyst** Microsoft Teams

*"Migration to prod by Monday. Notebook is mostly done, just clean it up."*

You open the folder.

- ✕ Notebook does not run
- ✕ CSV is messy
- ✕ No README or definition
- ✕ Data engineer is Out of Office

// THE ARTIFACT

# What we are looking at

```
customer-revenue/
├─ customer_revenue_report.ipynb      broken Fabric PySpark notebook
├─ data/transactions_2026.csv         messy, nulls, typos
├─ data/customers.csv                 dimension table
├─ PROJECT.md                         business context (easy to miss)
└─ expected_output_sample.csv         what good looks like
```

**The PROJECT.md is the trap that a chat window cannot use.**

***The agent will find it on its own.***

// DEMO · PHASE 1 OF 5

# Run, fail, fix, run

01

**PRINCIPLE** Tool use + self-correction

Get the notebook to execute. No reading docs yet. Just run, fix, rerun until green. Watch the agent close its own loop on three different errors.

- **LIVE DEMO**

// PHASE 1 RECAP

# What just happened

- ✓ You said: 4 sentences
- ✓ Agent did: 7 actions (run, read, edit, rerun, read, edit, rerun)
- ✓ Time: ~90 seconds vs. ~15 minutes manual
- ✓ Bugs fixed: ImportError, schema mismatch, file path

> **It never asked you for the next step. It read its own traceback.**

# Read the project, not the file

02

**PRINCIPLE** **Persistent context**

Agent discovers PROJECT.md, learns the 2026 VIP threshold change and the gold table naming convention. Applies them without being told twice.

• LIVE DEMO

// PHASE 2 RECAP

# What just happened

- ✓ Agent read 1 file you did not ask it to read
- ✓ Found business rule for 2026 (VIP threshold 10 000 → 12 000)
- ✓ Applied naming convention from PROJECT.md
- ✓ Notebook is now stylistically consistent with the project

> **A teammate reads the README. A chat window cannot.**

# Find what I did not ask about

03

**PRINCIPLE** **Autonomous multi-step reasoning**

Notebook runs. I am not confident the numbers are right. Agent inspects data, compares to expected output, finds silent bugs that never threw an error.

- **LIVE DEMO**

// PHASE 3 RECAP

# What just happened

- ✓ Silent bug: nulls in amount column underreporting sum
- ✓ Silent bug: inner join drops orphan customers
- ✓ Edge case: VIP threshold uses > not >=
- ✓ Proposed fixes plus data quality assertions

> **Silent bugs are the most expensive bugs in data engineering. The agent caught three.**

# Build your own debugger

04

**PRINCIPLE** Reusable agent

One playbook becomes a plugin. Agent builds a Cowork plugin with two skills (chaos-monkey breaks things, notebook-doctor fixes them) plus a sub-agent. Closed loop tested live: chaos to green in 30 seconds.

• LIVE DEMO

// PHASE 4 RECAP

# What just happened

- ✓ Agent built a Cowork plugin with 2 skills + 1 sub-agent
- ✓ notebook-chaos randomly injects bugs from a 6-item catalogue
- ✓ notebook-doctor runs the 4-phase recovery playbook autonomously
- ✓ Closed loop tested live: random chaos to green output in 30 seconds

> **Next Friday at 4 PM is no longer a demo. It is a routine the agent owns.**

// DEMO · PHASE 5 OF 5

# Same agent, now Power BI

05

**PRINCIPLE** Skills + MCP

The gold table goes into a semantic model. Agent loads the powerbi-authoring skill (microsoft/skills-for-fabric), builds relationships and measures through the Power BI Modeling MCP, runs a best-practice audit and fixes its own findings.

• LIVE DEMO

// PHASE 5 RECAP

# What just happened

- ✓ Loaded the powerbi-authoring skill only when the task needed it
- ✓ Built the model via Power BI Modeling MCP: relationships, measures, descriptions
- ✓ Validated every measure by running DAX, not by guessing
- ✓ Ran a best-practice audit and fixed its own findings

> **MCP gives the agent hands. Skills give it know-how.**

// SCORECARD

# Manual vs. agent vs. agent + plugin

MANUAL

## ~3 h

- ✗ Bugs missed: 3 (all silent)
- ✗ Reusable artifact: 0
- ✗ Context to reload: full

AGENT

*what you just saw*

## ~6-7 min

- ✓ Bugs missed: 0
- ✓ Reusable artifact: 1
- ✓ Context to reload: session

AGENT + PLUGIN

*next time*

## ~2-3 min

- ✓ Bugs missed: 0
- ✓ Reusable artifact: plugin
- ✓ Context to reload: none

// BUILD YOUR OWN

# What you need to replicate this

1

## An agent runtime

Claude Code, GitHub Copilot CLI, Cursor, Codex

2

## An instructions file

CLAUDE.md or AGENTS.md

3

## Tools

File IO, shell, one MCP per system you touch

4

## Skills and plugins

Playbooks for recurring work, e.g. skills-for-fabric

***Nothing here is exotic. You can have this running by lunch tomorrow.***

// INGREDIENT 1

# CLAUDE.md design

## Keep it under 200 lines.

Anything the agent needs only sometimes belongs in a skill.

*Short file = the agent understands its job.*
*Long file = you are over-instructing.*

**CLAUDE.md**

```
You are a senior Fabric data engineer.

## Conventions
- Python with type hints
- PySpark, not pandas, for >1M rows
- Log every read and write

## Norms
- Never edit prod configs without asking
- Always check expected_output_sample.csv
- Prefer small reusable functions over big notebooks
```

// INGREDIENT 2

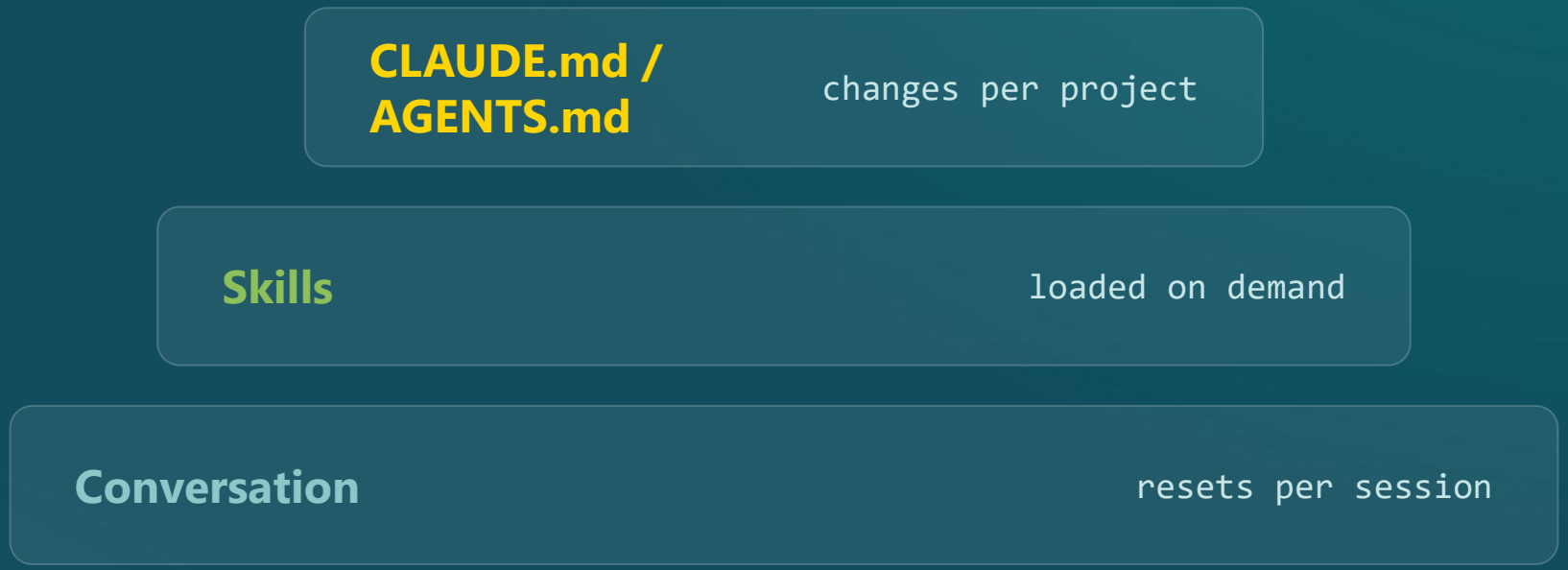
# Pick fewer tools than you think

| | |
|---|---|
| START | File read/write + shell |
| ADD | One MCP + its skill (e.g. Fabric MCP + skills-for-fabric) |
| AVOID | Every MCP under the sun |

***Tool sprawl is the most common failure mode I see. More tools = more wrong picks.***

// INGREDIENT 3

# Where memory lives




***Most teams skip the project CLAUDE.md. That is the highest-leverage file you can write.***

// INGREDIENT 4

# Where does it run



## 01

### Local laptop

Personal work, exploration, one-off tasks

`token cost only`

## 02

### Cloud VM / container

Shared team work, longer sessions, CI integration

`VM + tokens`

## 03

### Managed

Cowork, Claude Code on the web. Lowest friction, no infra ops

`subscription`

// ANTI-PATTERNS

# When NOT to use an agent

| | | |
|---|---|---|
| ✕ | **One-line questions** | → just use ChatGPT |
| ✕ | **No verifiable output** | → agent hallucinates confidently |
| ✕ | **Production writes without review** | → you are the safety net |
| ✕ | **Legal or compliance judgment** | → humans only |

// LANDSCAPE

# What about Copilot in Fabric?

PREVIEW

## Copilot in notebooks

Chat pane, Fix with Copilot. Strong inside one item.

PREVIEW

## Data Engineering Agent

Microsoft's native agent for migrations and ETL.

GA

## Fabric MCP + Fabric IQ MCP

Plug Fabric into any agent: Claude Code, Copilot CLI, VS Code.

***Not either-or. Copilot works inside one item. An agent works across your repo, tenant and tools.***

// HONESTY

# Where agents still fall short

- ! Long tasks drift without checkpoints - split them, add tests
- ! Cost adds up on tight loops - watch token usage
- ! Tool errors cascade - observability matters
- ! Cannot debug network or auth issues for you
- ! Cannot replace domain knowledge - garbage in, garbage out

// YOUR HOMEWORK

# Three things to do this week

1

**10 min**

Install Claude Code (or Copilot CLI) on one project

2

**30 min**

Write a CLAUDE.md for that repo

3

**1 hour**

Pick the task you hate most. Hand it to the agent.

***Zero barrier. Same 17 hours per week we talked about on slide 5 - waiting.***

// RESOURCES

# Where to go next



## Claude Code

`code.claude.com/docs`

Runtime, skills, plugins, memory



## Fabric skills

`github.com/microsoft/skills-for-fabric`

Fabric + Power BI skills for any agent



## Fabric MCP

`github.com/microsoft/mcp`

Official Fabric MCP servers



## DataBrothers

`databrothers.cz`

Let's talk about agents

# Rate this session

Why Your Data Team Needs an AI Agent (Not a Chatbot)



Data Brothers

# Questions?

Or come find me after at the booth.


**Štěpán Rešl**

stepan.resl@databrothers.cz
linkedin.com/in/stepanresl
github.com/DataBrothers


LINKEDIN

X

BLUESKY