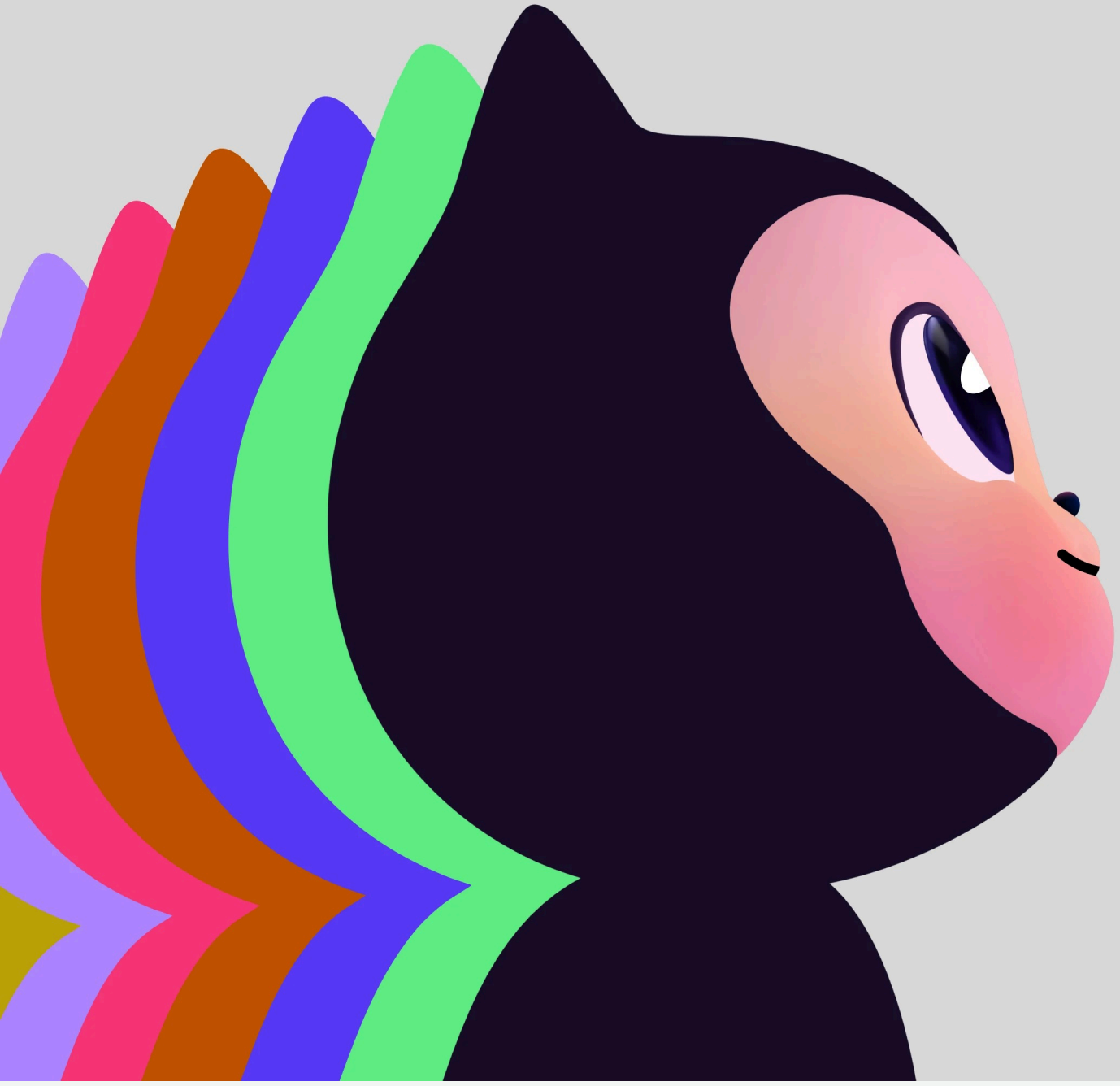


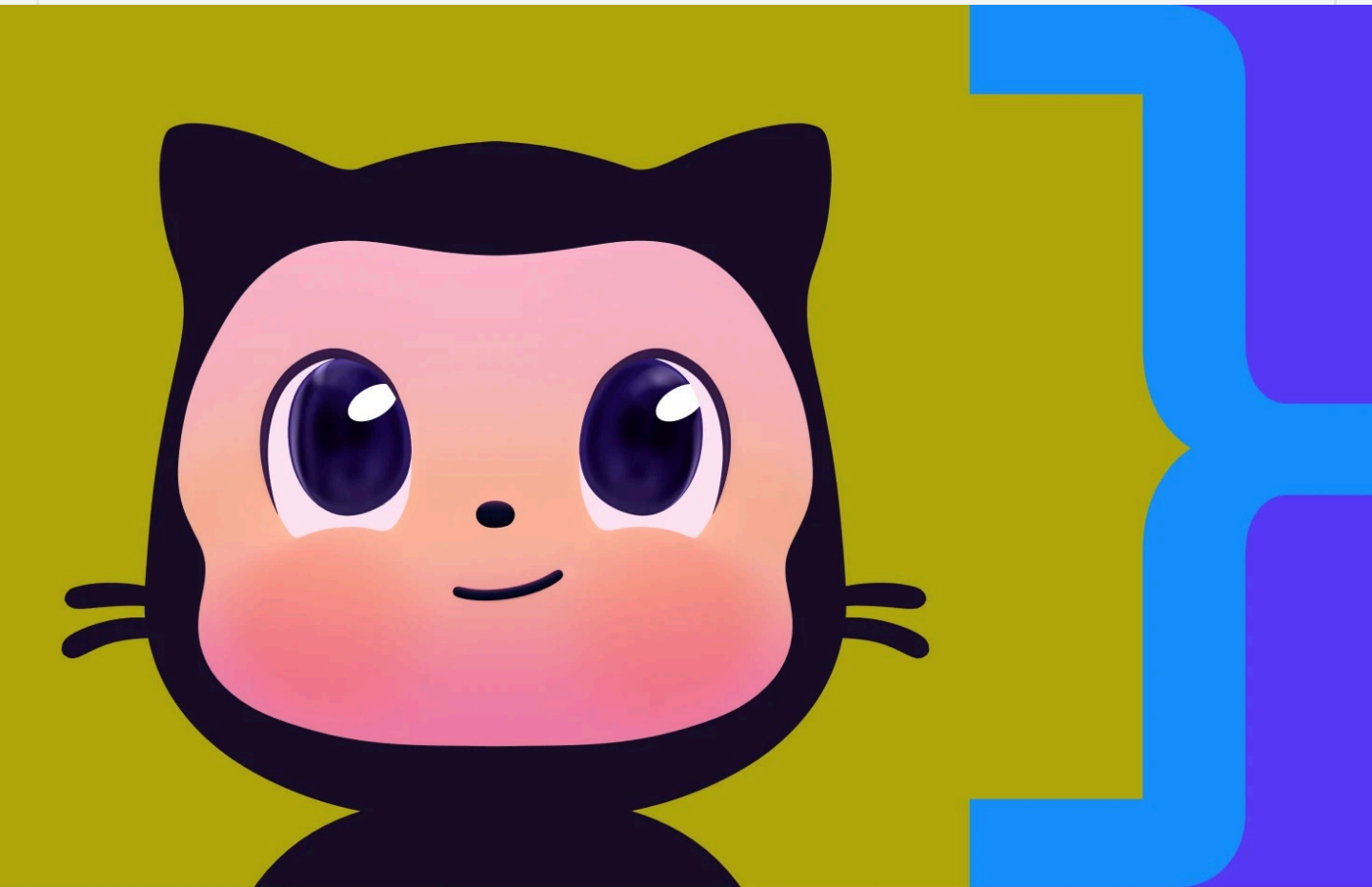
AI summary

VOXO

Build once, run on any agent: a practical guide to agent skills

Summary

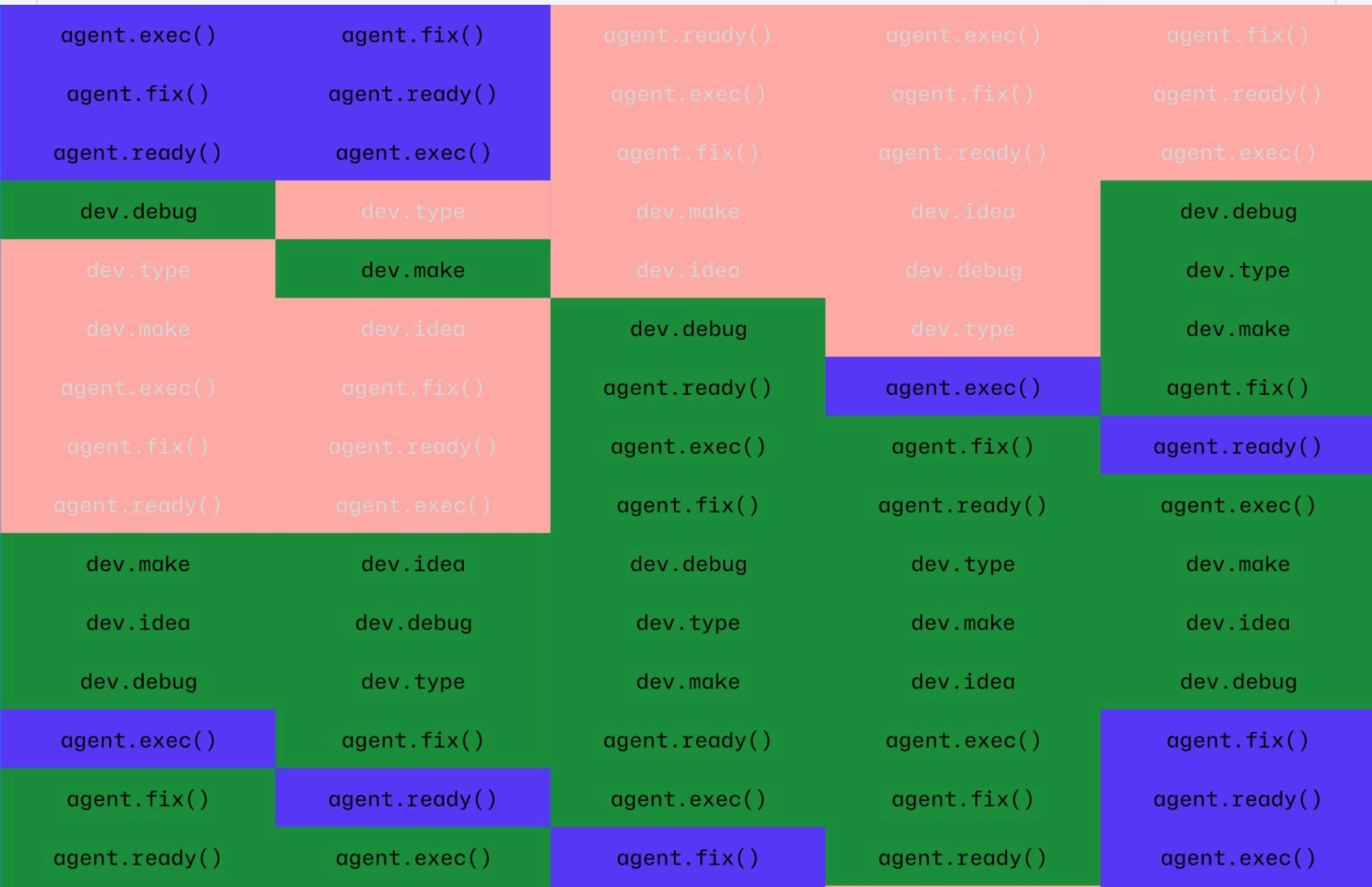
Andrea Griffiths and Shishir Tewari open by describing a familiar frustration: repeatedly re-explaining the same task to different AI agents and tools. Much of the necessary judgment lives in engineers' heads or in "Bob," the go-to expert who gets called whenever something breaks. Their goal is to show how to capture that tacit knowledge as reusable agent skills, so teams can build once, run on many agents, and stop losing context every time a chat window closes.



Griffiths introduces the concept of a skill as a versioned folder an agent can load when a request matches its description, centered on a skill.md file holding name, description, and instructions. She demonstrates her "proof agent" verifier skill, which receives only the original request and the code diff, deliberately excluding the worker agent's self-assessment to avoid bias. The verifier classifies outcomes as pass, fail, or partial, and must always return concrete evidence such as commits, diffs, or test runs.

This verification loop runs inside GitHub Actions on every pull request in the demo repository. The agent's verdicts do not block merges by themselves; instead they augment CI checks with structured findings that engineers can inspect and challenge. Multiple passes may be needed as the worker agent fixes issues and the verifier reevaluates, sometimes still ending in a partial assessment when human judgment or missing documentation is required. Griffiths emphasizes that skills should support, not replace, engineers' ownership of release decisions.

Tewari then presents an enterprise data-engineering scenario, where only a few experts know how to turn vague product tickets into reliable metrics pipelines. His DataForge framework encodes that expertise across coordinated skills such as ForgeInit, ForgeDBX, and ForgeLint. ForgeInit connects to platforms like DuckDB or Databricks, discovers existing tables and naming conventions, and writes a resolved manifest. Downstream skills reuse this manifest to generate schema, queries, and tests, without each skill inventing its own database connection or standards.



The framework walks engineers through discovery, design, schema generation, testing, and pull-request creation with deliberate human-in-the-loop checkpoints. It surfaces proposed grains, column choices, and counting logic for example using count distinct incident IDs and asks for approval before writing objects or opening PRs. DataForge also updates GitHub issues with findings, ensuring traceability from ticket to code. Griffiths shows the same skills orchestrated from different environments, including IDEs and the GitHub Copilot CLI, illustrating that skills travel as versioned files across agents.

In closing, the speakers distinguish repeatable skills from sturdier apps used for shared state and governance, urging teams to design both around standards and explicit enforcement. They propose Monday-morning "homework": identify the recurring questions that always go to an expert, write the business configurations down once, centralize execution interfaces, and ensure agents load standards before generating code. By codifying judgment as reusable skills, teams reduce on-call emergencies, make decisions auditable, and stop depending on "Bob" every Friday night.

Key takeaways

Codify Repeated Expertise As Reusable Skills

Agents do not know your team's unwritten standards, so engineers repeatedly re-explain tasks or rely on one expert "Bob." By codifying that judgment as reusable skills—versioned, reviewed instruction files with supporting assets—you teach agents once and reuse the process everywhere.

Separate Verification And Demand Evidence

Griffiths' proof-agent skill separates worker and verifier agents: the reviewer sees only the request and diff, not the worker's self-assessment. Each outcome—pass, fail or partial—must cite concrete evidence such as commits or file diffs, keeping humans in the loop for final decisions.

Orchestrate Data Pipelines With Config-Driven Skills

Tewari's DataForge framework shows multi-skill orchestration: one skill discovers database schemas, others lint, generate DDL, tests and create PRs, all driven by shared configuration and JSON "memory." Human approvals gate deployment, and the same skills port between DuckDB, Databricks or Copilot CLI.

Participants

Andrea Griffiths
Senior Developer Advocate, GitHub

Shishir Tewari
Senior Manager, Data Engineering, Procore Technologies

Powered by



voxo.ai

WEDNESDAY, OCTOBER 28, 2026
BUILD ONCE, RUN ON ANY AGENT: A PRACTICAL GUIDE TO AGENT SKILLS