HECTOR FLORES / HTEK.DEV

Building Agentic Workflows with GitHub Copilot

Define the outcome. Bound the action. Verify the result.

GitHub: Dynamic workflows / GitHub Agentic Workflows / Hector: Three pillars of Agentic DevOps

The artifact is no longer the only thing you design

Before	Now also	Always
Specify each implementation step	Specify the outcome and boundaries	Own the evidence and decision

Intent becomes an engineering artifact.

Hector: Context engineering / Hector: Three pillars of Agentic DevOps

From suggestions to governed processes

Assistance	Agent loop	Workflow
Suggest code in the editor	Plan, act, test, iterate	Reuse steps, rules, and handoffs

The shift is **who owns the next step**.

Introducing GitHub Copilot / Copilot general availability / Copilot CLI general availability / Dynamic workflows announcement

A goal, a loop, and a stop condition

Observe	Decide	Act	Verify
Relevant context	Analysis and judgment	Permitted tools	Evidence and feedback

Stop when the **acceptance criteria** are met, not when the answer sounds complete.

GitHub: Dynamic workflows / GitHub Agentic Workflows

Agency is not unlimited authority

The agent may	A human still owns
Investigate within scope	The goal and acceptable risk
Propose a change	The delivery decision
Request a permitted effect	Policy and approval boundaries

More autonomy is a choice, not a maturity score.

gh-aw: Safe outputs / GitHub: Dynamic workflows

One issue. Three surfaces.

> A Task Tracker endpoint handles an invalid status inconsistently.

GitHub	Copilot workflow	Extension tool	Human
Triage the issue	Investigate the behavior	Validate the proposal	Review the evidence

Synthetic fixture, not a claim about a current production defect.

A good goal describes observable success

Weak: "Fix our API."

Better: "Explain why invalid task statuses are accepted. Produce a failing acceptance test and a minimal proposal. Do not change unrelated endpoints."

> Outcome + scope + evidence + stop condition

Hector: Context engineering / GitHub: Dynamic workflows

Constraints are part of the solution

- **Scope:** this issue, these files, this repository
- **Authority:** read, propose, or execute?
- **Resources:** agreed budget and concurrency
- **Escalation:** missing evidence, ambiguity, or rejected validation

gh-aw: GitHub tools and allowlists / gh-aw: Safe outputs / gh-aw: Network policy

Better context beats a bigger prompt

Intent	Grounding	Feedback
Requirements and non-goals	Relevant code and conventions	Failing tests and real results

Give each stage **what it needs**, not everything you have.

Hector: Context engineering / Hector: Three pillars of Agentic DevOps

Use the simplest sufficient process

Task shape	Start with
Small local question	One agent
Dependent stages	A sequence
Independent investigations	Parallel workers
Consequential action	Verification and a human handoff

An agent count is not an outcome metric.

GitHub: Dynamic workflows / Hector: The maturity curve

Three surfaces. Different responsibilities.

Surface	Primary role
GitHub Agentic Workflows	Event/schedule-driven repository automation in Actions
Copilot dynamic workflows	A reusable process defined in code
CLI extensions	Session-aware capabilities implemented in JavaScript

An extension can **host** a dynamic workflow.

GitHub Agentic Workflows / GitHub: Dynamic workflows / GitHub: Copilot extensions

Put repository events to work

Trigger	Definition	Actions	Result
Issue, PR, schedule	Markdown + frontmatter	Agent execution	Permitted outputs

GitHub Agentic Workflows complements CI/CD.

Separate reasoning from write execution

Agent step	Request	Handler job
Analyze with scoped reads	Structured permitted output	Validate and execute

An upper bound is not a promise of exactly one action.

gh-aw: Safe outputs

Tool management is architecture

Question	Control to inspect
What can it call?	Toolsets and individual allowlists
What can it access?	Credential and repository scope
Where can it connect?	Network and hosted-retrieval policy
What can it change?	Outputs, permissions, policy gates

Exposure, permission, and approval are different things.

gh-aw: Tools / gh-aw: GitHub tools and allowlists / gh-aw: Network policy

Demo 1 / proposed storyboard / not recorded

The issue enters the workflow

1. Open a synthetic invalid-status issue.
2. Observe the configured event and run.
3. Inspect the scoped read tools.
4. Review the bounded triage result.

Evidence: issue, workflow revision, run, and actual output.

Public Task Tracker workflow examples / gh-aw: GitHub tools and allowlists / gh-aw: Safe outputs

The useful part was not the label

- A clear goal made the decision bounded.
- Relevant context made it explainable.
- Narrow output permissions limited the effect.
- A human still decides what work to accept.

> Automate the repetitive decision without hiding its consequences.

GitHub Agentic Workflows / gh-aw: Safe outputs / Hector: Context engineering

Copilot dynamic workflows / public preview

Define the process in code

Combine **deterministic operations** with **agent judgment**.

Sequence. Branch. Parallelize. Verify. Pause. Resume.

Available across CLI, app, and SDK; CLI demonstrations require experimental features.

GitHub: Dynamic workflows / GitHub: Using dynamic workflows / Dynamic workflows announcement

Autonomy, delegation, and orchestration

Feature	Who chooses the process?
Autopilot	The agent keeps choosing next steps
Fleet	The agent divides and coordinates work
Dynamic workflow	The author defines steps and handoffs in code

Not three names for "more agents."

GitHub: Dynamic workflows

Orchestrate the process, not every thought

Collect	Investigate	Synthesize	Verify
Issue and fixture	Independent bounded questions	Structured findings	Evidence and checkpoint

Write code for **control flow**. Ask agents for **judgment**.

GitHub: Dynamic workflows / GitHub: Using dynamic workflows

Demo 2 / proposed storyboard / not recorded

Investigate with structured handoffs

1. Explicitly pass the triaged issue into Copilot.
2. Collect the fixture and acceptance criteria.
3. Ask API and test investigators distinct questions.
4. Combine evidence; pause before applying a change.

No assumed Actions-to-local-CLI bridge.

GitHub: Dynamic workflows / GitHub: Using dynamic workflows / Public Task Tracker workflow examples

Design for the run that does not finish

- Keep partial evidence visibly incomplete.
- Make retried side effects idempotent.
- Validate meaning, not only JSON shape.
- Observe limits and pause at explicit boundaries.

"The workflow stopped" is not "the work is done."

GitHub: Using dynamic workflows / Copilot CLI programmatic reference

Copilot CLI extensions / experimental

Knowledge is not capability

Instructions / skills

What the agent should know

Extensions / tools

Operations implemented in code

Hooks / policy

Checks at defined boundaries

A CLI extension runs alongside the session as a Node.js process.

GitHub: Copilot extensions / Hector: Knowledge, capability, enforcement / Hector: CLI extensions guide

Turn a vague action into a narrow contract

Not: "Run whatever command fixes this."

Instead: validate a proposed task-status contract.

Input	Fixed code	Result
Known schema	No shell interpolation	Pass/fail + reasons

GitHub: Copilot extensions / Hector: Vibe testing and false confidence

Demo 3 / proposed storyboard / not recorded

Rejection is useful feedback

1. Submit an invalid status-contract proposal.
2. Inspect the validator's concrete rejection.
3. Correct it and validate again.
4. Present evidence for a human decision.

A passing check is not permission to merge.

GitHub: Copilot extensions / Hector: Vibe testing and false confidence

Human-in-the-loop, not human-as-cleanup

Boundary	Review question
Before execution	Is this the right scope and authority?
After investigation	Does the evidence support the finding?
Before delivery	Do we accept the change and its risk?

Make rejection a normal, tested path.

GitHub: Dynamic workflows / gh-aw: Safe outputs

What a polished demo can hide

- Untrusted issue text becomes an instruction.
- Context drifts; workers duplicate or conflict.
- Tests turn green by weakening assertions.
- Credentials, rate limits, retries, or costs surprise you.

> Evaluate failures as deliberately as successes.

gh-aw: GitHub tools and allowlists / gh-aw: Network policy / Hector: Vibe testing and false confidence / Hector: Context engineering

Scenarios worth trying

Need	First bounded artifact
Issue intake	Classification with evidence
Documentation drift	A reviewed update proposal
Failing CI	A diagnosis and reproducer
Release readiness	A checklist with explicit blockers
Migration	An isolated, tested change set

Scale the operating model

Ownership / Who maintains the workflow?

Isolation / What data, tools, and workspaces are shared?

Evidence / How do we know it helped?

Economics / What does a useful result actually cost?

gh-aw: GitHub tools and allowlists / gh-aw: Network policy / Hector: Three pillars of Agentic DevOps / Hector: The maturity curve

Pick one measurable workflow

Read-only	Propose	Automate narrowly
Report with evidence	Human-reviewed artifact	Known effects and rollback

Keep the simplest level that delivers value.

Define the outcome.

Bound the action.

Verify the result.

Your next engineering artifact might be the workflow.

Explore the handout, notes, and sources

Hector: Three pillars of Agentic DevOps / GitHub: Dynamic workflows / GitHub Agentic Workflows

APPENDIX / AVAILABILITY

Check the recording environment

- Dynamic workflows: **public preview**
- CLI dynamic workflows: experimental features enabled
- CLI extensions: **experimental**
- Check current eligibility, policy, permissions, and billing

Documentation reviewed for this presentation on November 12, 2026.

GitHub: Dynamic workflows / GitHub: Using dynamic workflows / GitHub: Copilot extensions / Copilot CLI command reference

APPENDIX / CONTROL CHECKLIST

Inspect the effective configuration

Layer	Ask
GitHub tools	Which groups and individual calls?
Authentication	Which repositories and permission scopes?
Network	Runner egress and hosted retrieval?
Safe outputs	Which writes and maximum counts?
Policy	What requires a separate human decision?

gh-aw: GitHub tools and allowlists / gh-aw: Safe outputs / gh-aw: Network policy

APPENDIX / CONCEPTUAL PSEUDOCODE

A reusable investigation process

```
validate input and collect bounded evidence

ask independent investigators distinct questions

preserve incomplete results explicitly

combine and verify findings

pause for review of this exact proposal

return evidence; do not deploy
```

Illustration of control flow, not copy-paste API syntax.

GitHub: Dynamic workflows / GitHub: Using dynamic workflows

APPENDIX / CHOOSE A SURFACE

What do you actually need?

Requirement	Start with
Reusable instructions and resources	Skill
External service tools	MCP
A lifecycle command	Hook
Session-aware JavaScript capability	CLI extension
Repeatable code-defined process	Dynamic workflow
Distributable capability bundle	Plugin

GitHub: Copilot extensions / GitHub: Dynamic workflows

APPENDIX / PRESENTATION CONTROLS

Take the material with you

- Arrow keys navigate; Esc opens overview.
- S opens speaker view; F toggles fullscreen.
- The handout includes public notes and source links.
- Download slide and handout PDFs from the toolbar.

  All notes are public. Demo sections are currently **storyboards**.

reveal.js speaker view / reveal.js PDF export