Cloud-Agent Workflows

Bounded issues, human review, and policy-approved comparison

Use Enterprise Cloud as the governance baseline. Check current official GitHub documentation and customer administrator policy before delivery.

Access and cost preflight

1. Confirm the repository, data classification, participant role, and approved task.
2. Verify current official documentation and customer policy for the selected workflow.
3. Use a non-sensitive, bounded issue with explicit acceptance criteria.
4. For metered work, define a customer-owned threshold, escalation route, and stop guard.
5. Prepare the issue-writing and review exercise as the no-access fallback.

An issue is the work contract

```
## Problem
State the user-visible problem and the bounded scope.

## Acceptance criteria
- [ ] Expected behavior is testable.
- [ ] Relevant tests pass.
- [ ] A reviewer can verify the change.

## Constraints
- Approved files and data only
- Required checks
- Human review before merge
```

Review before merge

Use the same review rubric for each proposed change:

- correctness and test evidence;
- repository conventions;
- data, permissions, and dependency changes;
- scope against the issue;
- required human approval.

Do not treat an automated proposal as proof that the change is ready.

Comparing approved agents

Customer policy controls third-party agent availability, workflow, model selection, data handling, and metering. Check current official GitHub documentation and customer policy. Do not assume a named agent is enabled.

Compare	Evidence
Access	Is this agent approved for this repository and data class?
Quality	Does it meet the same acceptance criteria and tests?
Safety	What tools, permissions, and data are involved?
Reviewability	Is the change understandable and suitable for human review?
Measurement	What does the customer-defined meter show?

No-access fallback

If no approved agent is available:

1. Write the issue and acceptance criteria.
2. Perform the change manually in the repository sandbox.
3. Use the PR review checklist.
4. Compare the manual baseline with recorded, approved evidence if available.

When to pause

Pause when policy, data scope, access, review ownership, or the stop guard is unresolved. Use the fallback.

Agenda and time plan

Time	Topic	Trainer move
0:00	Async agent mental model	Trace issue to draft PR
0:08	Repository configuration	Explain setup boundaries
0:18	Issue contracts	Improve a weak issue
0:28	Live or recorded assignment	Observe progress evidence
0:42	Security and session management	Apply stop conditions
0:50	Agent comparison and team workflow	Use one rubric
0:58	Lab handoff	Confirm reviewer ownership

What makes the cloud agent different

The cloud agent works asynchronously from an issue in an isolated environment.

- Work starts from an assigned GitHub issue.
- The agent creates a branch and proposes repository changes.
- It can run configured setup, builds, and tests.
- Progress is visible through an agent session.
- The result is a draft pull request for human review.

Say: “Async changes location and timing. Accountability stays the same.”

Issue-to-PR lifecycle

```
Bounded issue assigned
        ↓
Environment prepared
        ↓
Agent plans, edits, and validates
        ↓
Progress reported in session
        ↓
Draft pull request opened
        ↓
Human feedback, revision, approval, or close
```

Branch protection and required reviewers remain authoritative.

Repository setup responsibilities

Use current official documentation when configuring the selected repository:

- setup steps and required development tools;
- runner or environment assumptions;
- firewall and network allowlists;
- secrets and permission boundaries;
- repository instructions and validation commands;
- branch protection and review requirements.

Start with least privilege and a disposable training repository.

Setup steps example

```
name: Copilot setup steps
steps:
  - name: Install dependencies
    run: npm ci
  - name: Verify baseline
    run: npm test
```

This is a teaching example, not a complete schema reference. Verify the current filename, syntax, environment, and permitted commands before use.

Instructions guide repeatable behavior

```
# Repository instructions

- Make the smallest change that satisfies the issue.
- Do not add dependencies without explicit approval.
- Follow patterns in `src/` and update focused tests.
- Run `npm test` and report failures accurately.
- Never modify generated or deployment files unless the issue names them.
```

Instructions support an issue. They cannot clarify vague acceptance criteria.

Weak issue, strong issue

Weak

```
Fix validation.
```

Strong

```
Reject blank task titles in `POST /tasks`.

- Return HTTP 400 with `{ "error": "title is required" }`.
- Preserve valid creation behavior.
- Add focused tests for blank, whitespace, and valid titles.
- Do not add dependencies or change other endpoints.
```

Concrete target change

```
export function normalizeTitle(value) {
  if (typeof value !== "string" || value.trim() === "") {
    throw new ValidationError("title is required");
  }
  return value.trim();
}
```

The reviewer checks error mapping, existing conventions, focused tests, and unexpected files in the diff.

Live assignment narrative

1. Show a clean issue with explicit non-goals.
2. Confirm repository, access, metering guard, and reviewer.
3. Assign through the currently supported workflow.
4. Open the agent session and identify plan, tool use, and checks.
5. Do not wait silently; explain the prepared fallback.
6. Open the draft PR, compare it with the issue, and inspect tests.
7. Leave bounded feedback rather than rewriting the task.

Feedback that agents can act on

```
The whitespace test passes, but the valid-title test now changes the response
schema. Restore the existing response shape. Keep the validation helper and tests.
Run the focused suite and report the result.
```

Good feedback cites evidence, restates the constraint, and stays in scope.

Agent session management

Track:

- repository, issue, branch, and session owner;
- current phase and last useful evidence;
- commands, failures, and revisions;
- metered-work observation and stop guard;
- feedback awaiting response;
- final PR, close, or fallback decision.

Stop stale or looping sessions instead of leaving them to consume resources.

Security boundaries

Boundary	Safe default
Network	Allow only required destinations
Tools	Minimum approved toolset
Shell	Review setup and avoid destructive commands
Secrets	Provide only task-required, scoped credentials
Data	Synthetic or approved repository data
Merge	Human review plus repository protections

Never place credentials in issues, instructions, or comments.

Comparing approved agents

Assign the same bounded issue only when policy and metering permit it.

Compare:

1. correctness against identical acceptance criteria;
2. scope and diff clarity;
3. tests and failure reporting;
4. tool, network, and data exposure;
5. review effort and required rework;
6. measured usage under the customer-defined method.

Named third-party agents are examples. Their availability is not assumed.

A junior-developer workflow

Treat an agent contribution like work from a junior teammate:

- assign a suitable, bounded task;
- provide repository conventions;
- expect questions or course correction;
- inspect implementation and evidence;
- give specific review feedback;
- retain accountable human approval.

Do not use an agent as an unmonitored backlog queue.

Demo fallback and debrief

If live execution is delayed or unavailable:

- review a prepared issue, session timeline, and draft diff;
- identify the earliest point to intervene;
- write one actionable PR comment;
- apply the same review rubric to a manual implementation;
- list checks required before a future live assignment.

Ask: “What evidence increased or reduced your confidence?”

Lab handoff

Learners will:

1. review repository setup and security boundaries;
2. write three issues of increasing but bounded complexity;
3. assign work only through approved agents;
4. monitor sessions and stop one simulated scope drift;
5. review draft PRs and provide evidence-based feedback;
6. compare results under one rubric.

Deliverable: Reviewed agent PRs or equivalent fallback evidence.

Key takeaways

1. The issue is the cloud agent’s work contract.
2. Setup, network, tools, and instructions define boundaries.
3. Agent sessions require active ownership and stop conditions.
4. Draft PRs require the same tests and human review as any contribution.
5. Compare agents by evidence, never reputation alone.

Questions and lab readiness

- Which issue is safe to assign?
- Who reviews the resulting draft PR?
- What event triggers the stop guard?