IDE Agent Workflows

Plan, act, observe, and review

Access and cost preflight

Use Enterprise Cloud as the governance baseline.

1. Verify current official GitHub documentation and customer administrator policy.
2. Confirm repository scope, data classification, and permitted tools.
3. Use a non-sensitive, bounded task with tests.
4. For metered work, define a customer-owned threshold, escalation route, and stop guard.

The agent loop

```
Plan → Act → Observe → Review → Revise or stop
```

Use the loop to review the work. Confirm current controls and behavior before a live demonstration.

A safe task contract

```
## Goal
Add one small, testable behavior.

## Constraints
- Approved files and data only
- Required tests
- No unapproved dependencies

## Completion
- Tests and review evidence are available
- A human approves the change
```

Review checkpoints

- Inspect the proposed file changes.
- Review every command before it is run.
- Check test results and unexpected side effects.
- Stop when scope, data, policy, or the customer-defined guard is unclear.

No-access fallback

Complete the same brief manually:

1. Write the plan.
2. Make the smallest change.
3. Run the same tests.
4. Review against the same acceptance criteria.

Agenda and time plan

Time	Topic	Trainer move
0:00	Assistant-to-agent shift	Contrast task shapes
0:08	Agent loop and tools	Trace one iteration
0:18	Task contracts and trust	Define boundaries
0:28	Multi-file live demo	Pause at checkpoints
0:43	Chat vs. agent mode	Apply decision framework
0:50	Intervention and TODO delegation	Show safe handoff
0:57	Lab handoff and Q&A	Confirm stop rules

Transition: “Autocomplete predicts text; an agent pursues a goal through tools.”

From autocomplete to agent

Mode	Developer supplies	Surface returns
Inline	Local code context	A continuation
Chat	A question and references	Guidance or a proposed edit
Agent	Goal, constraints, and repository	A multi-step change with evidence

More autonomy needs a bounded scope, observable progress, and review.

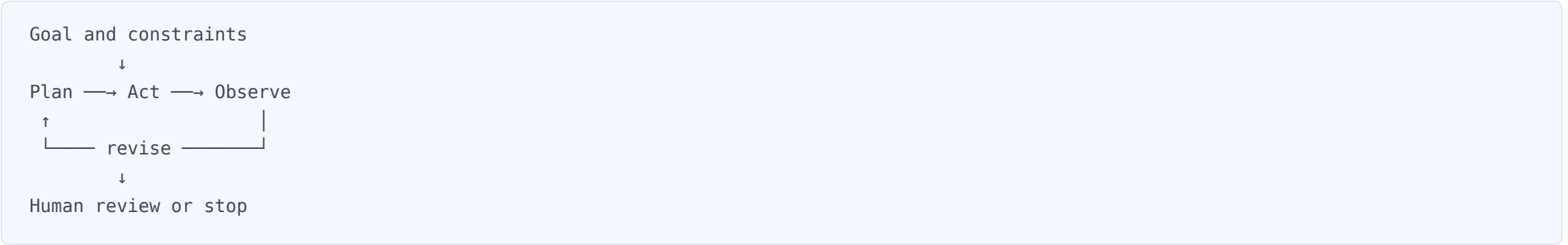
What agent mode can do

Subject to current product support and policy, agent mode may:

- inspect repository files and conventions;
- plan a sequence of changes;
- edit multiple related files;
- run approved terminal commands and tests;
- observe failures and revise;
- use approved tools, including configured MCP tools.

Product capability does not grant approval. Check what policy allows.

One loop, many checkpoints



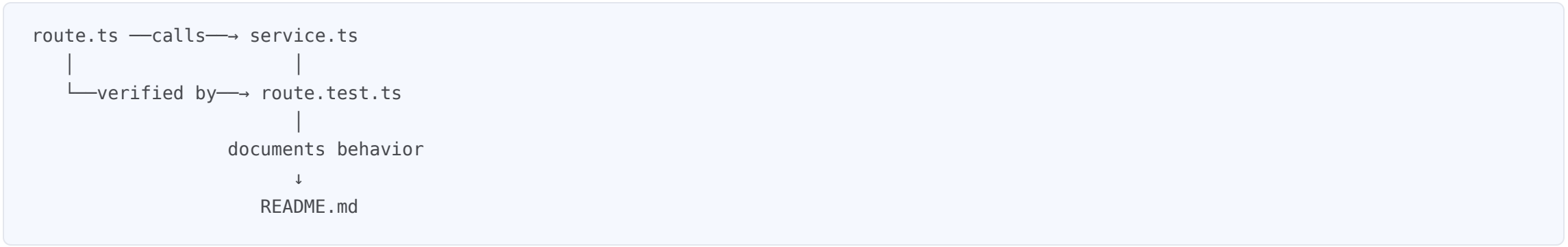
Trainer cue: Pause after each arrow and ask what evidence should be visible.

Tool use and trust boundaries

Tool action	Review question
Read a file	Is this path and data approved?
Edit a file	Is it required by the task?
Run a command	Is it understood, reversible, and scoped?
Add a dependency	Is it necessary, approved, and reviewed?
Search the web	May repository context leave the environment?
Call an MCP tool	Are server identity and permissions approved?

Stop rather than infer authorization.

Multi-file work is a graph



Name the behavior, not a file count. Review every changed node and check for missing connections.

Demo brief: one feature, three files

```
Add `completed` filtering to the task list endpoint.

Acceptance criteria:
- `GET /tasks?completed=true` returns completed tasks only.
- Invalid values return HTTP 400.
- Existing behavior without the parameter is unchanged.
- Focused tests cover true, false, and invalid values.

Constraints: no new dependencies; edit route, service, and tests only.
```

Concrete demo code

```
const value = req.query.completed;
if (value !== undefined && value !== "true" && value !== "false") {
  return res.status(400).json({ error: "completed must be true or false" });
}

const completed = value === undefined ? undefined : value === "true";
return res.json(taskService.list({ completed }));
```

Ask the agent to implement and test the smallest valid change, then explain it.

Narrate the demo

1. Show a clean working tree and the focused test command.
2. Paste the task contract and request a plan before edits.
3. Check that the plan names route, service, and tests.
4. Approve only commands needed for the task.
5. When a test fails, ask the agent to explain before revising.
6. Inspect the final diff and run tests independently.

Transition: “A passing test is evidence. It does not complete the review.”

Intervene, redirect, stop

Use a correction that preserves the original contract:

```
Stop. Do not change the response schema or add dependencies.
Keep the existing service API and add only the optional filter.
Show the revised plan before editing.
```

Stop for unexpected file scope, destructive commands, restricted data, unapproved network access, or repeated changes without new evidence.

Same task: Chat or agent mode?

Use Chat when...	Use agent mode when...
You need explanation or options	The outcome spans related files
You want one surgical edit	Tests can drive iteration
Requirements are still forming	Requirements and constraints are explicit
You need to retain manual control	Tool use creates useful evidence

Implement manually when policy, risk, or ambiguity makes an agent unsuitable.

TODO code actions

A TODO can become a concise delegation point:

```
// TODO: reject duplicate task titles within the same project.
```

Before delegating:

- convert the comment into explicit acceptance criteria;
- confirm the target repository and branch;
- remove sensitive context;
- define tests and reviewer ownership;
- verify current support and customer policy for the selected handoff.

Demo debrief

Ask learners:

- Which plan step gave the earliest useful signal?
- Which command required the most scrutiny?
- Did the agent touch exactly the expected files?
- Which acceptance criterion was easiest to miss?
- Would Chat have been safer or faster for this task?

Use the answers to discuss task selection.

Lab handoff

Learners will scaffold, implement, test, and refactor bounded features while:

1. observing tool calls and multi-file edits;
2. redirecting the agent after an intentional scope change;
3. comparing manual and agent-mode implementations;
4. trying a TODO delegation only when access is approved;
5. recording the evidence used to accept or reject output.

Deliverable: A tested feature and a reflection on agent decisions.

Remember

1. Agent mode is best for explicit, testable, multi-step work.
2. Task contracts convert autonomy into reviewable boundaries.
3. Tool approval and file scope are trust decisions.
4. Intervention is expected, not a failure.
5. Human review closes the loop.

Questions and lab readiness

Where should a reviewer intervene in your team’s workflow?

- What is your stop condition?
- Which test will you run independently?
- Is the starter repository and access ready?