# Plan work with GitHub Issues and MCP

Session M7-03 | Product and Delivery Teams | Intermediate

# Turn the brief into GitHub issues

Session M7-02 produced an approved decision brief.

Session M7-03 turns it into real GitHub issues.

This is the third step in **Delivery Decision Studio**.

# MCP in one sentence

An MCP server gives Copilot named tools for reading or changing another system.

You use the tools in this session without building the server.

A plugin may bundle an MCP server with skills or other customizations. Check the active GitHub MCP connection itself before reading or writing issues.

# Write issues others can act on

The issue should let a contributor decide whether they can start, what success looks like, and who will review it.

| Issue element | What it prevents |
| --- | --- |
| Outcome | Work that repeats a vague request |
| Results needed for acceptance | "Done" without a way to prove it |
| Non-goal | Scope growth during implementation |
| Dependency | Work that starts before a needed decision |
| Ownership gap | Silent assumptions about who decides |

Write enough detail for the next contributor to act.

# Parent and child issues answer different questions

The parent issue holds the outcome, shared constraints, and review path. Each child issue owns one independently reviewable result.

```
Parent: Improve request-status visibility
  ├─ Child: Public status mapping
  ├─ Child: Requester status view
  ├─ Child: Empty and stale states
  └─ Child: Requester-access verification
```

If two child issues must edit the same behavior at the same time, sequence them or clarify the design first.

# Review each MCP write

Copilot can prepare the issue plan. A person approves the write only after seeing the target repository and expected result.

After the write, fetch the issue again. Compare what GitHub stored with the approved plan instead of relying on the tool's success message.

The Markdown plan is a temporary preview. **The created GitHub issues are authoritative.**

# Read before writing

1. Check the repository.
2. Inspect existing labels and milestones.
3. Search for duplicate issues.
4. Read the approved brief.
5. Report permission or context uncertainty.

# Decide what to do with existing work

| Candidate | Decision |
| --- | --- |
| Exact duplicate | Reuse or close according to repository policy |
| Partial overlap | Extend the existing issue and create only the missing work |
| Conflict | Keep separate and record why |
| Unrelated | Leave unchanged |

The lab includes open, closed, overlapping, and conflicting status issues.

# Decompose by outcome

Create one parent issue and child issues with defined scope.

Each child needs:

- an independently reviewable result;
- results needed for acceptance;
- a non-goal;
- dependencies;
- an owner or visible ownership gap.

# Preview the complete write

Before approval, show:

- repository owner and name;
- issue count;
- every title and body;
- labels and milestone;
- relationships;
- fields left unset.

# People approve the operation

Copilot proposes the action.

Record **Approve**, **Revise**, or **Reject** before the write.

Do not create issues without approval.

# Check what GitHub saved

After the write:

1. start a fresh request;
2. retrieve the issues;
3. compare them with the approved plan;
4. correct one defect;
5. retrieve the corrected issue again.

# Lab result

- Temporary reviewed issue preview
- Authoritative live issue set
- One parent issue
- Four required child roles; an optional fifth stays outside Session M7-05
- Saved issues checked through a fresh GitHub read
- Parent-to-child links that Session M7-04 reads directly

**Stop if you cannot review the tool call or check the saved issues.**