GitHub Copilot in the CLI

Choose the surface that fits the work

Session M2-01 | 3 hours

Agenda

Segment	Time
CLI fit and setup	10 min
Modes and checked plan critique	10 min
Programmatic use	10 min
Agents and fleet supervision	10 min
Controls, queue, and recovery	10 min
Surface choice and handoff	10 min

Start with the task

Choose the CLI when the work starts or ends in the terminal:

- reading logs or command output;
- working over SSH or in a container;
- running a repeatable command sequence;
- producing output for another script.

Use the IDE when visual navigation and diff review matter more.

Match the mode to the risk

Mode	Use it when
Standard	The task is small or still changing
Plan	You want to review the approach before edits
Autopilot	The task has clear limits and a recovery plan

Permissions still apply. More autonomy requires a narrower task.

Revise the plan before accepting it

```
/plan Find the three failing behaviors. Name the files and tests for each fix.
Do not edit project files.
```

Add a constraint: preserve the response fields and status codes.
Ask which steps change. Check the diff before approval.

Save both plans and the assumption you challenged.

Ask a different critic before approval

```
/rubber-duck Critique the repair plan against the source and tests.
Look for a hidden defect or response-contract change. Do not edit files.
```

Rubber duck is built in and read-only. Copilot automatically selects a contrasting model when one is available. Check the findings yourself.

Inspect `/subagents` Proactive invocation; leave the existing setting unchanged. That opt-in persists and adds usage. This lab requires a manual invocation.

Fleet needs independent work

Plan separate read-only questions about routes, utilities, and test gaps.

1. Leave Plan mode and run `/fleet` with that breakdown.
2. Open `/tasks` and inspect actual child-task results.
3. Check claims against files or focused tests.
4. Reconcile disagreements and inspect `/usage`.

Do not promise an agent count or speedup. Extra agents can use more credits.

Shared commands, different surfaces

Workflow	Surface
`/plan`, `/fleet`, `/review`	CLI and app
`/orchestrate`	App built-in skill; Session M7-05
`copilot -p` and shell pipelines	CLI
Cloud changes through `/delegate`	Session M3-02

Check the current command picker before delivery.

A task with clear limits is easier to review

State:

- the working directory;
- the files or commands in scope;
- the expected result;
- the checks to run;
- when Copilot must stop.

Do not automate an unclear request.

Programmatic mode needs a contract

```
copilot -p "Summarize the failing test output"
```

Before another script consumes the answer, define:

- stable input;
- a predictable output format;
- a failure signal;
- limited permissions.

Free-form output is not a reliable pipeline interface.

`--output-format=json` emits **JSONL events**. Parsing each line does not prove that the task succeeded or that the answer meets the requested schema.

CLI and IDE can share one workflow

```
CLI
  reproduce the failure
  capture focused evidence
        ↓
IDE
  inspect the code and diff
  review the proposed change
        ↓
CLI
  rerun checks
  record the result
```

Switch surfaces when the work changes.

Context is not authority

The CLI can see terminal state, files, Git state, and command output.

That context helps Copilot reason. It does not decide:

- which data is approved;
- which commands are safe;
- whether the change is correct;
- whether the result should ship.

A person owns those decisions.

Check before changing files or systems

Before a command changes files or external systems:

1. check the directory and scope;
2. inspect the proposed action;
3. limit tools and credentials;
4. define a rollback or stop rule;
5. check the result outside the conversation.

Use the smallest permission that completes the task.

Inspect approval, sandbox, and limit separately

```
/permissions show
/sandbox status
/sandbox policy node --version
/limits
```

Approval controls whether a tool may run. The sandbox limits process access.
Inspect both without weakening them.

AI-credit caps are soft. GitHub's references disagree about reset scope.
Check the installed UI with the trainer; stop before metered work if unclear.
Do not force exhaustion or treat a cap as a hard budget.

Queue, stash, and background are different

Control	Result
`Ctrl+Q`	Queue a follow-up while busy; check the pending label
`Ctrl+S`	Stash or restore an unsent draft
`Ctrl+X, b`	Promote a running command/task to background
`/tasks`	Inspect its actual state, output, and exit status

Use the lab's harmless 15-second command.
Check the queued summary against the real result.

Background work remains in the CLI session. It is not file isolation.

Recover before branching an alternative

1. Commit the checked repair.
2. Let Copilot create only a disposable note.
3. Inspect `/rewind` choices; restore conversation **and files**.
4. Check the note is gone and the repair remains.

Rewind does not undo external actions or replace Git rollback.
Later manual edits may be skipped; inspect the result.

Context fork is not file isolation

Command	Check
`/fork`	Conversation fork; do not infer a separate checkout
`/fork worktree`	Context plus a new checkout of `HEAD`
`/worktree`	New checkout; uncommitted edits stay behind
`/move`	Transfer the uncommitted draft into a new worktree
`/resume`	Return to the parent and check its path/branch/status

Move only the analysis note. Do not merge the alternative in this lab.

Save the work for review

Keep:

- the final diff;
- test output and exit status;
- generated reports;
- the prompt and stated limits;
- the final human decision.

Save these files and results so another engineer can check the work without the conversation.

Lab handoff

Use `/plan`, challenge an assumption, and approve the revision before repair. Inspect live `/fleet` results through `/tasks` and check the accepted claims. Check the critique against the source. Record the controls, queue/background result, and recovery/worktree state.

120 minutes: setup 15; repair 40; programmatic/fleet 35; recovery/choice 30.

Submit: passing checks, a reviewed diff, the health report, and `workflow-record.md` with the plan revision and observed usage.