THE WORKSHOP

Sprint Planning

A pattern for turning a refined backlog into a sprint the team believes in — a goal, a set of stories, task breakdown, and an explicit commitment — in one focused session.

2026-04-17
barkingiguana.com/writing/the-workshop-sprint-planning/

A facilitator's playbook -- print, annotate, and keep on the table.

Contents

This is the pattern. If you want the story of sprint planning in action, read Sprint Planning: Turning Sticky Notes into Delivery – the Greenbox team running their first planning session and discovering why the goal matters more than the story list. This post is the reference you keep open the morning of.

Intent

Turn a refined, prioritised backlog into a sprint the team can commit to: a single clear goal, a set of stories selected against capacity, a concrete task breakdown, and a stated commitment from everyone in the room.

Also Known As

Often just called **planning** or **iteration planning**. Sometimes confused with **roadmap planning** (roadmap planning spans months; sprint planning spans one sprint) and with the ongoing story-preparation work that should happen *before* planning, not inside it. The ritual comes from Scrum, but the pattern – "here is what we will do and here is why we believe it" – predates Scrum by decades and shows up in every delivery discipline under a different name.

Motivation

A team finishes a sprint having delivered six of the eight stories they pulled in. The two unfinished stories get carried over. The next sprint they pull in ten stories to "catch up" and finish five. The sprint after that they pull in twelve. Velocity is now invisible, commitment has become theatre, and the team quietly stops believing the numbers.

What went wrong wasn't the work. It was the planning. Nobody said out loud what the sprint was actually *for*. Stories got pulled in because they were next in the list, not because they served a goal. When a fire came up mid-sprint, the team had no way to decide whether to fight it or park it, because there was no goal to measure the fire against. Every sprint became a slightly different version of "do as much as possible," which is indistinguishable from "do whatever's loudest."

Sprint Planning exists to break that pattern. The sprint goal is the contract. The stories are the plan. When the plan has to change – and it always does – the goal is the thing you steer by. Without a goal, a sprint is a to-do list. With one, it's a commitment.

Applicability

Use when:

- You work in sprints of one or two weeks
- The top of the backlog has been refined – stories have acceptance criteria, have been through Example Mapping or equivalent, and are sized
- The whole team can attend
- Someone can articulate what the sprint should *achieve*, not just what should be done

Don't use when:

- The top of the backlog is a mess — prepare the stories first, in a separate session; planning is not story preparation with an audience
- The team operates in continuous flow and has no sprint boundary
- You're planning more than one sprint ahead — that's roadmap work, a different session
- The product owner can't attend — reschedule

Inputs. A refined backlog, the team's actual capacity for this sprint (holidays, on-call, carry-over, external commitments), and recent velocity. Nothing else needs to be prepared in-session — if it does, the preparation didn't happen.

Participants

Facilitator. Runs the session, holds the clock, keeps the team off preparation detours. Often the Scrum Master, team lead, or a rotating role.

Product owner. Mandatory. They set the sprint goal, explain the stories, and make the trade-off calls when capacity doesn't match ambition. Without them in the room, you are planning to build the wrong thing.

Developers. The whole development team. Sprint Planning is the one meeting where partial attendance breaks the ritual — if a developer isn't in the room, they haven't committed, and the commitment is the whole point.

Tester / QA. If they sit with the team, they're part of the team, and they plan with the team. Testing capacity is capacity. Treat it that way.

Operations / SRE. For any team whose sprint includes deployment work, on-call rotations, or infrastructure change, ops is a first-class planning participant. On-call load is a real capacity drain and if it isn't in the plan it will consume the plan anyway.

Group size: the whole team, typically 4–9. Sprint Planning is one of the few patterns that scales with team size — larger teams need longer sessions, not different ones.

Who to leave out:

- **Stakeholders.** They shape the backlog before planning and see the output afterwards. They don't attend planning itself. Observers warp the commitment conversation because people self-censor in front of the people they serve.
- **Other teams.** Dependencies on other teams belong in the task breakdown as risks, not in the room as people.
- **Senior leaders with "just a quick ask."** The just-a-quick-ask is the thing that destroys the sprint goal you're supposed to be setting. Leadership input happens in the story-preparation work upstream of planning, not in planning itself.

Structure

Phase	Duration (1-week sprint)	Duration (2-week sprint)	Key question
Sprint goal	10 min	15 min	"What must this sprint achieve?"
Story selection	20 min	40 min	"What fits, against real capacity?"
Task breakdown	25 min	50 min	"What are the concrete steps?"
Commitment check	5 min	10 min	"Does everyone believe we can do this?"
Total	**1 hour**	**~2 hours**	

The rule of thumb is one hour per sprint week. Most teams beat that once they've run the pattern a few times and once upstream story preparation is reliable. If you're consistently running longer, the problem is upstream: stories are arriving at planning unready.

Collaborations

Sprint Planning is less about sticky notes and more about explicit agreement. Four phases, each with a different shape:

- **Goal-setting** is a conversation between the product owner and the team, moderated by the facilitator. The product owner proposes, the team pressure-tests, the facilitator writes the agreed goal somewhere everyone can see it for the rest of the session.
- **Story selection** is a negotiation. The product owner defends priority; the team asserts capacity; the facilitator holds the capacity number honest.
- **Task breakdown** is team work. The product owner is available for questions but not driving. Developers, testers, and ops decompose each story into tasks that fit inside a day.
- **Commitment check** is a round-the-room moment. Each person says, in their own words, whether they believe the plan is achievable. This is the contract.

The rhythm is **set, select, break down, commit** — and if any of the four collapse into the others, the ritual stops working.

Facilitator Playbook

Phase 1 — Sprint goal (10–15 minutes)

Before any story is discussed, the product owner proposes a sprint goal. Not a list. A sentence.

Open with:

> *"Before we look at stories, let's agree what this sprint is for. Product owner, if we could only ship one thing this sprint — one capability the team delivers, one metric we move, one problem we solve — what is it?"*

Write whatever they say on the whiteboard. Then pressure-test it with the team:

> *"Is this achievable in one sprint? Is this worth a sprint? Does everyone in this room understand why this matters?"*

The goal should be **specific**, **achievable in one sprint**, and **measurable or demonstrable**. A good goal sounds like: *"Subscribers can pause and resume their subscriptions through the web app."* A bad goal sounds like: *"Make progress on the subscription area."*

Once the team accepts the goal, write it in large letters at the top of the whiteboard. Everything that follows is in service of this goal.

What to watch for:

- **No goal, just a list.** The product owner says *"let's just pull in as much as we can."* Push back: *"If we could only ship one thing this sprint, what would it be?"* Refuse to move to story selection until a goal is on the board.
- **Vague goal.** *"Make progress on subscriptions"* is not a goal. Push: *"What would a subscriber be able to do at the end of this sprint that they can't do now?"*
- **Three goals disguised as one.** *"Pause, resume, and billing integration"* is a roadmap item, not a sprint goal. Help the product owner pick the most important one; the others come next sprint.
- **SRE sprint goal looks different.** For an ops-heavy sprint, the goal might be *"Reduce deployment rollback rate from 1 in 5 to 1 in 20"* or *"Move the billing cron to the new scheduler with zero missed runs."* Same shape, same test: specific, achievable, demonstrable.

Phase 2 — Story selection (20–40 minutes)

Start from the top of the backlog. For each story, the product owner gives a thirty-second explanation of what it is and why it serves the sprint goal. The team confirms they understand it (a quick nod around the room is enough — if it's more than a nod, the story wasn't refined). Then the team decides whether it fits.

Keep a running tally of points (or t-shirt sizes, or hours, or whatever you size in) against capacity, visible at the edge of the whiteboard. When the tally reaches capacity, stop pulling.

Say it explicitly when you get there:

> *"We're at 24 points. That's our capacity. Any story we pull in now has to push another one out. Product owner, is there a trade you want to make?"*

What to watch for:

- **Over-commitment.** The team pulls in more than their average velocity because *"this sprint feels different."* It never is. Use the velocity. Optimism is not a planning strategy, and a team that over-commits twice loses the ability to trust itself.
- **Under-commitment.** The team sandbagging because they got burned. A sprint or two of under-commitment to rebuild confidence is fine; persistent under-commitment means the stories are bigger than estimated or there's hidden work the estimates don't cover.
- **Skipping the priority order.** *"Let's skip story 3 and do story 7 instead."* Only the product owner can approve a priority swap, and they should say why out loud. Otherwise the backlog order stops meaning anything.

- **Gold-plating during selection.** The team starts redesigning a story. Redirect: *"We're deciding what's in, not how to build it. Task breakdown is next."*
- **Carry-over invisible.** Last sprint's unfinished work is coming in. Make it visible in the tally: *"We have 8 points of carry-over. That leaves 16 for new work."* Carry-over that isn't counted is how velocity quietly disappears.

Phase 3 — Task breakdown (25–50 minutes)

For each selected story, the team breaks it into tasks. A task is concrete enough that one person can do it in a day or less.

The facilitator's job in this phase is mostly to keep moving. Give each story a time budget (*"five minutes per story"*) and hold it. If a story needs more than five minutes of task breakdown, it wasn't ready for planning — pull it and prepare it separately.

For each story, the team identifies:

- **What needs to happen** — the concrete tasks, written on sticky notes or into the tracker
- **Who's likely to do what** — not assignments, but a flag for tasks that need specific expertise
- **Dependencies** — between tasks, between stories, between teams
- **The not-obvious work** — testing, deployment, migration, documentation, feature flag cleanup, on-call handover

What to watch for:

- **Tasks that are too big.** *"Build the UI for pausing"* is probably three tasks: form, validation, API wiring. If a task is longer than a day, split it.
- **Missing the unglamorous work.** Teams forget testing, migrations, deployment, feature flag management, observability wiring, documentation updates, on-call runbook edits. Prompt: *"What else has to happen before we can call this done?"*
- **One-person bottlenecks.** If every story has the same developer flagged as essential, that's a risk. Don't solve it now — flag it and discuss pairing or knowledge-sharing in the retrospective.
- **External team dependencies.** If a task needs another team's API, approval, or review, name it and name the person who'll chase it. Better still: can you start with a mock or stub so the dependency isn't blocking?
- **On-call capacity.** If a developer is carrying the pager this sprint, their capacity is not the same as a developer who isn't. Build that in during task assignment, not after the fact.

Phase 4 — Commitment check (5–10 minutes)

Read the sprint goal aloud. Read the list of selected stories. Then go round the room and ask each person the same question:

> *"Do you believe we can deliver this sprint as planned?"*

This is not a vote. It's a check. You are looking for the person whose body language doesn't match their words. You are giving the quiet team member a direct invitation to raise a concern that would otherwise stay silent until the retrospective.

If someone says *"we'll try,"* that is a no. Accept it as a no. Ask what would turn it into a yes. Usually the answer is "remove this one story," which you then remove.

What to watch for:

- **Silent discomfort.** Someone doesn't object but their face says the plan is too much. Ask them directly, by name: *"You've gone quiet. Which story are you worried about, and what would it take to remove the worry?"*
- **"We'll try."** That's not commitment, that's politeness. *"What would it take to make this a yes?"*
- **Scope-cutting inside stories.** *"We can do it if we skip the error handling."* No. Error handling is in the acceptance criteria or it isn't. Cut scope by removing stories, never by cutting corners inside them.
- **The product owner negotiating down.** The product owner offers to cut a story to reassure the team. Let them. This is the ritual working.

When everyone has said yes — genuinely said yes, not politely nodded — the sprint is committed. Write the commitment down with a date. The sprint starts.

Steering When It Goes Sideways

The preparation session in disguise. Fifteen minutes debating what a story means, mid-planning. *Recovery: "This story isn't ready. Let's pull it out of the sprint and work through it properly this week — maybe with Example Mapping — before it comes back to planning."* *Stop if:* Three stories in a row need that treatment. The top of the backlog isn't ready for planning. End the session, schedule the preparation work, and come back.

The wish list. The product owner keeps adding *"just one more."* *Recovery:* Hold the capacity number honest: *"We're at 24. This story is 5. Which 5-point story do you want to remove to make room?"* Force the trade, every time. *Stop if:* The product owner won't accept capacity as a constraint. That's a systemic problem, not a session problem. Flag it to leadership outside the room.

The architecture debate. Developers start debating framework choices during task breakdown. *Recovery:* Park it: *"Capture that as a design question. We need to know can we do it this sprint, not how."* *Stop if:* The debate is blocking task breakdown entirely. The story needs a time-boxed design investigation before it's plannable; pull it.

The absent product owner. Product owner cancels the morning of. *Recovery:* Reschedule, same day if possible, next day if not. *Stop if:* This keeps happening. The ritual is broken; escalate the pattern, not the individual session.

The permanent over-commitment. Every sprint the team commits to 30 points and delivers 20, and nobody adjusts. *Recovery:* In the next planning session, write the last three sprints' *delivered* totals on the board before story selection. Plan to the delivered number, not the committed one. Watch what happens. *Stop if:* The product owner insists on the committed number anyway. That's a systemic trust problem; a planning session won't solve it.

The silent veto. Commitment check passes, but one person clearly doesn't believe it. They've said yes because saying no feels rude. *Recovery:* Take a break. Talk to them privately. Bring the concern back into the room as *"There's a worry about the migration story that I don't think we surfaced properly — can*

we talk through it before committing?" so the objection is legitimised by the facilitator, not left to the quiet person to defend alone. *Stop if:* They still won't speak. The team has a safety problem, not a planning problem.

Consequences

Benefits

- A sprint goal the whole team can state from memory — the one line that makes mid-sprint trade-offs decidable
- A plan everyone has explicitly committed to, not silently acquiesced to
- A task breakdown concrete enough that the daily standup has something to work against
- Dependencies, on-call load, and carry-over made visible before the sprint starts, not after it breaks

Costs

- One hour per sprint week, every sprint, forever
- A product owner who has to be available and prepared
- A story-preparation process upstream that reliably produces ready stories — without it, planning becomes preparation with an audience and doubles in length
- Discipline to say no to "just one more" every single time

Failure modes

- The sprint goal gets skipped and the sprint becomes a to-do list
- Carry-over isn't counted against capacity and velocity quietly collapses
- The commitment check becomes theatre and nobody actually believes the plan
- Task breakdown is skipped because "we know what to do," and the team discovers mid-sprint that they didn't
- The session runs so long the team arrives at their first ticket exhausted

Stop signals

- The backlog isn't ready — reschedule and prepare the stories first
- The product owner is absent — reschedule
- Two sprints in a row the committed plan wasn't achievable — the sprint length, the preparation process, or the estimation practice is broken, not the planning session

Ending planning and scheduling something else is not failure. Committing to a sprint nobody believes is.

Worked Example

See Sprint Planning: Turning Sticky Notes into Delivery for the Greenbox team running their first planning session — including the moment they discover that a sprint goal changes every single decision that follows it, and the moment one of the developers realises the commitment check exists precisely for people who were about to nod along with a plan they didn't believe in.

Outputs & Follow-up

Facilitator's close-out (same day)

- Sprint goal written where everyone can see it: team board, wiki, Slack topic, the top of the sprint in the tracker.
- All selected stories moved into the sprint in the tracker, with tasks attached.
- External dependencies communicated to the teams they touch, today, before the sprint starts.
- Photograph the whiteboard if the task breakdown happened on physical sticky notes.

The product owner's week

This is where the pattern earns its cost, and the work is mostly the product owner's.

- **Protect the sprint goal.** Every "just one more thing" request that arrives this sprint gets measured against the goal. If it doesn't serve the goal, it goes into the backlog for next sprint. If it does, it replaces something that doesn't. The product owner is the only person who can make that trade.
- **Watch the burndown at the midpoint.** For a two-week sprint, Thursday of week one. For a one-week sprint, the morning of day three. If you're behind, have the conversation about what to cut *now*, not on the last day. The goal survives a scope cut; it doesn't survive a last-day scramble.
- **Prepare the top of the backlog for the next planning session.** This is the unglamorous work that makes next sprint's planning an hour instead of three. Run Example Mapping on the candidate stories. Answer the red cards. Size the stories. Arrive at the next planning session with a backlog that is actually plannable.
- **Walk the goal to anyone who matters.** Stakeholders, leadership, dependent teams. "Here's what we're doing this sprint and here's why" said once at the start prevents five "what are you working on" interruptions mid-sprint.

Ongoing

- Track velocity honestly. It's the single most useful planning input and it only works if you measure what actually shipped, not what was committed.
- If planning sessions consistently run long, the preparation happening before them needs work. Stories should arrive at planning ready to plan.
- Retrospect on the planning session itself periodically. Is the goal still the contract? Is commitment still meaningful? Or has the ritual become theatre?

Related Patterns

- **Example Mapping** — the gate before planning. A story that hasn't been through Example Mapping is not ready to plan; plan it anyway and you'll be mid-sprint when you find out why.
- **User Story Mapping** — Story Mapping produces the backlog; Sprint Planning commits to a slice of it. If the backlog is chaotic, Story Mapping is where that gets fixed, not in planning.
- **Impact Mapping** — the sprint goal should trace back to an impact and a goal on the Impact Map. If it doesn't, you're planning features, not outcomes.

- **Retrospectives** — the retrospective is where you notice that planning sessions have become theatre and fix the ritual before it fully collapses.
- **Ensemble Programming** — when the task breakdown keeps flagging one developer as the sole person who can touch a system, ensemble work is the pattern that breaks the bottleneck.

About this playbook

This playbook is part of *The Workshop*, a reference series of facilitator playbooks published at barkingiguana.com. The canonical, up-to-date version lives at barkingiguana.com/writing/the-workshop-sprint-planning/.

These posts are LLM-aided. Backbone, original writing, and structure by Craig. Research and editing by Craig + LLM. Proof-reading by Craig.