THE WORKSHOP

Retrospectives

A pattern for creating a structured, safe space to inspect how the team is working and decide one or two things to change, so that continuous improvement stops being a poster on the wall and starts being a thing that actually happens.

2026-10-02
barkingiguana.com/writing/the-workshop-retrospectives/

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

Contents

The session where the team gets honest about how it actually works. A retro turns "this sprint went OK I guess" into one or two concrete actions that reduce the chance of the same problem next sprint. Worked example: Catching the Wrong Kind of Fast.

Retrospectives

A retrospective creates a structured, safe, time-bounded space for a team to inspect how it's working, decide one to three small changes, and commit to follow-through, at whichever scale (sprint, project, quarter) fits the cadence of the work. Formalised by Norm Kerth in his 2001 book *Project Retrospectives*, with the Prime Directive (*"regardless of what we discover, we understand and truly believe that everyone did the best job they could"*) that sets the tone for every good session since; also called retro, post-mortem (when focused on an incident), lessons learned, after-action review (the military version), or kaizen meeting (the lean version). Frequently confused with status meetings (which look backward at what was done) and one-on-ones (which are for individual feedback); a retrospective is the team looking at how it works together.

At a glance

- *Who, for how long:* a facilitator who doesn't participate (rotated over time), plus the whole team, no managers, no stakeholders, no spectators. Four to eight people for a sprint retro, 60-90 minutes.
- *What you walk out with:* one to three concrete actions, each with a single named owner and a deadline (usually "by the next retro"), plus an updated action tracker and a brief summary sent to the team within 24 hours.
- *When to reach for it:* at the end of every sprint, after a project ends, after an incident, or when team dynamics feel off and nobody is naming what's wrong. Not for individual feedback (that's a one-on-one), and not when the team has no power to change anything or last retro's actions haven't been addressed.

What's It For

A team runs retrospectives every two weeks. The same complaints come up every time: too many meetings, unclear priorities, code reviews take too long. The team generates sticky notes. Someone writes down the actions. Nothing happens. Two weeks later the same sticky notes appear. After six months the team is running retros out of obligation, everyone is cynical, and the complaints have calcified into identity: *"we're the team with too many meetings."*

Or: a team ships a major project that went badly. Missed deadlines, rework, moments of panic. Everybody has theories about what went wrong. The theories contradict each other. Nobody sits down to reconcile them, because the project is done and the next one is starting. The next project makes the same mistakes because the lessons were never extracted, just felt.

Or: a team has an incident. A post-mortem is written. It names a person. The person gets defensive. The post-mortem becomes about blame instead of about systems. The fix is *"the developer will be more careful next time,"* which is not a fix, and the next incident happens to a different developer and everyone is surprised.

These are all the same problem: continuous improvement is structurally fragile. It requires time, safety, and follow-through, and in the absence of deliberate design all three decay. A retrospective, done well, is the deliberate design. A retrospective, done badly, is worse than no retrospective; it teaches the team that improvement is theatre.

The pattern is the difference between "done well" and "done badly." It's small adjustments applied consistently: review last time's actions first, keep observations specific, limit actions to one to three, name an owner, protect the psychological safety, rotate the format so the ritual doesn't go numb. Done well, a team improves so gradually that the change is invisible week to week and dramatic quarter to quarter. Done badly, you generate sticky notes forever.

Reach for it when:

- At the end of every sprint or iteration; this should be routine, not exceptional
- After a project ends, at any scale (a feature, a migration, a product launch)
- After an incident, as a structured post-mortem focused on systems rather than people
- At the end of a quarter or at significant organisational milestones
- When team dynamics feel off and nobody is naming what's wrong

Benefits when it lands:

- Small, concrete changes to how the team works, compounding over time into dramatic differences over a quarter
- Reinforcement of habits that are already working (the underrated half of the practice)
- Psychological safety built deliberately rather than hoped for
- Early warning when team dynamics are drifting, before the drift becomes a crisis
- A shared vocabulary for talking about *how we work*, not just *what we built*

What It's Not For

Skip it when:

- It's a substitute for a difficult one-on-one conversation. Retros are for team-level issues, not for individual feedback.
- The team has no power to change anything. Surfacing problems nobody is allowed to fix breeds cynicism faster than any other pattern.
- Last retro's actions haven't been addressed. Fix the follow-through problem first; running another retro on top of unresolved actions is performative.
- The "team" is really a set of individuals who don't work together. Retros need a shared experience to reflect on.

Costs to weigh:

- 60-90 minutes per sprint, whole team, non-trivial person-hours over a year
- The emotional cost of honest conversations, which is both a cost and part of the value
- The political cost of findings that are uncomfortable for people outside the team

- Coordination cost of protecting the space; the retro that always gets cancelled when things are busy is the one the team most needs

Failure modes to watch for:

- Actions never get done, the retro becomes venting, the team goes cynical
- Same format every time, the ritual goes numb, people start skipping
- A manager is in the room and the honesty evaporates silently
- Blame replaces systems thinking, and the retro becomes a trial
- Findings are surfaced that the team has no power to change, and the retro becomes an exercise in learned helplessness

Stop signals:

- Three retros in a row with no actions completed
- Participants actively asking to skip the retro
- The same complaints in the same words every time, with no change in tone
- Someone has stopped speaking entirely in retros and you don't know why

The retrospective's real work isn't producing the actions; it's protecting the follow-through. A retro that produces one action that actually happens teaches the team that they can change how they work. That belief, that the team has the power to improve its own situation, is worth more than any specific improvement, and it's the thing the facilitator is really protecting. A retrospective that changes one small thing is infinitely more valuable than a retrospective that identifies ten big problems and changes none.

Definitions & Background

The portable retro pattern is Larsen and Derby's *Agile Retrospectives* five-stage model (Esther Derby and Diana Larsen, authors of *Agile Retrospectives: Making Good Teams Great*, 2006):

1. Set the stage: arrival, ground rules, the Prime Directive read aloud (*"regardless of what we discover, we understand and truly believe that everyone did the best job they could"*), a one-word check-in.
2. Gather data: the team produces observations. Start/Stop/Continue, sailboat, timeline; these are *gather-data activities*, not whole-retro formats. Pick one for variety; the stage stays the same.
3. Generate insights: discuss and cluster the observations. The stage most often skipped, which is why so many retros jump straight to actions and produce shallow fixes.
4. Decide what to do: pick one or two concrete actions with named owners and dates.
5. Close: appreciation, summary, next-steps confirmation.

The facilitator is distinct from the participants and speaks only to manage the process. The team speaks as equals. The Prime Directive (*everyone did the best they could with what they knew and what they had*) is the frame that lets honest observation happen without turning into blame; read it aloud in stage 1, every retro, every time.

Silent writing before open discussion is the key move inside *gather data*: it prevents the loudest voice from shaping what others think to write. By the time people are talking, everyone has already committed their observations to paper, and the debate is about the observations, not about who spoke first.

For project and quarterly retrospectives, there's often a second rhythm: individual reflection, small-group clustering, whole-group synthesis. Larger groups need smaller groups inside them.

Inputs

- Last retro's actions, written somewhere everyone will see them on entry. This single artefact is the difference between a retro that compounds and one that resets every time.
- Sticky notes and pens, or the remote-tool equivalent. One observation per note.
- A wall, whiteboard, or shared canvas with the chosen format drawn on it before participants arrive (Start/Stop/Continue columns, the Sailboat picture, a Timeline, etc.).
- A timer the room can see; the time-box is part of the protection.
- A private room. Psychological safety needs four walls and a closed door, or the remote equivalent: a call with no observers and no recording.
- The Prime Directive, written or printed where it can be read aloud at the top of the session.

Bring nothing else except the actions from the last retro and a willingness to hear uncomfortable things.

Outputs

What lands at the end of the session:

- One to three concrete actions, each with a specific observable change, a single named owner, and a timeline (usually *"by the next retro"* for sprint-level actions). Not five. Not seven. One to three.
- An acknowledged review of last retro's actions: which happened, which didn't, which to carry forward, drop, or amend.
- A photograph of the board for the team's records.
- A short written summary the facilitator sends to the team within 24 hours: the top themes, the committed actions, and the owner of each. Not the raw notes; those are for the team, not for outside consumption.
- An updated action tracker: the simple shared document that lists *action / owner / status / retro date*. This is the cheap tool that makes compound improvement possible.

These outputs feed into:

- Event Storming: a post-incident Event Storming *is* a retrospective with sticky notes on a wall instead of a conversation around a table. Use it when the incident crossed enough systems that a conversation can't hold the shape. Use a standard retrospective when the issue is about *how the team worked,* not *what the system did*.
- Threat Modelling: after a security incident, combining a retrospective (how did the team miss this) with a threat model (what was the attack, what category did we not apply) produces better lessons than either alone.

- Assumption Mapping: when a retrospective surfaces a disagreement about why something failed, it's often really a disagreement about which assumptions the team was carrying. Assumption Mapping turns the disagreement into testable statements.
- Impact Mapping: a quarterly retrospective naturally leads into an Impact Mapping session for the next quarter. The retro answers *"what did we learn about how we work,"* and Impact Mapping answers *"given that, what are we going to try to achieve next."*
- Ensemble Programming: ensemble programming surfaces team dynamics faster than any other technique, and the retrospective is where those dynamics get named and adjusted. The first few retros after a team starts ensembling are some of the most valuable ones they'll run.

Who's Needed

Facilitator. Runs the session, protects the psychological safety, enforces the time box, ensures actions have owners. On a first retrospective the facilitator should be someone the team trusts and who is not an authority figure over them. Over time, facilitation can and should rotate.

The team. The whole team. Not the people who attended the last one. The whole team. If someone can't attend, reschedule if possible; if not, the facilitator's job expands to bring the absent voice into the room via pre-submitted notes.

Nobody else. This is the hardest rule and the most important one. Retrospectives require psychological safety, and psychological safety evaporates when people feel observed by someone with power over their career, their budget, or their reputation. If a manager controls promotions, they don't belong in the team's retro. If a stakeholder gets upset about criticism, they don't belong. If someone outside the team needs to hear the findings, the facilitator can share a summary afterwards with the team's permission.

The exception is the project retrospective, where the boundaries are different: a cross-functional project may include stakeholders and adjacent teams by explicit invitation, because the lessons are cross-functional. Even then, invite with care.

Group size: typically 4–8 for a sprint retro, larger for project and quarterly retros. Above 10 people, the discussion dynamics change; split into smaller groups for the observation phase and bring them back together for the action phase.

Who to leave out:

- Managers outside the team. Even well-meaning ones. Their presence changes what people say, whether or not they mean it to.
- Stakeholders for sprint retros. Stakeholder feedback has its own place; the sprint retro isn't it.
- Spectators. *"I just want to observe"* is not a role. Either they participate as a team member or they're not in the room.
- Anyone whose presence is vetoed by the team. If someone is causing the team not to speak freely, the team's ability to be honest matters more than that person's feelings about being excluded.

How To Run It

The sprint-level format below is the default. Project and quarterly levels adjust the durations and the format, not the shape (see *Variants*).

Phase	Duration (sprint)	Materials	Key question
Check-in and review of last retro's actions	10 min	Previous actions visible	"One word: how are you feeling? Did we do what we said?"
Generate observations	15 min	Sticky notes, format drawn on wall	"What should we start, stop, continue?"
Discuss and cluster	20 min	The wall	"What's the pattern here? What's the impact?"
Decide on actions	10 min	,	"What's one concrete thing we'll change?"
Close	5 min	,	"One word: how do you feel about the next sprint?"
Total	60 min (one-week sprint) / 90 min (two-week sprint)		

The 90-minute version doesn't proportionally stretch every phase; it mostly expands the observation and discussion phases, because two weeks of material needs more space.

Choose the format before the session. Rotating formats matters; the same format every two weeks goes numb. Three good ones to rotate between:

Start / Stop / Continue. Three columns. Simple, fast, good for routine sprint retros.

Sailboat. The team draws a boat, then captures wind (helping forces), anchors (slowing forces), rocks (risks ahead), and an island (the goal). The wind pushing us forward, the anchor holding us back, the rocks ahead, the island we're heading for. Good for forward-looking reflection.

Timeline. Draw a timeline of the period under review. Everyone adds events, feelings, and observations along it. Good for project and quarterly retros where sequence matters.

The playbook below is for Start/Stop/Continue. Adapt Phase 2 for other formats; the rest is the same.

Before anyone arrives, write last retro's actions where everyone will see them on entry. This single move is the difference between a retro that compounds and one that resets every time. The first thing the team should see walking into the room is what they committed to last time, and whether it happened.

Phase 1: Check-in and review of last retro's actions (10 min)

Open with a one-word check-in. Go round the room:

> *"One word each. How are you feeling about the last sprint?"*

Two minutes. Don't editorialise; just capture. The purpose is twofold: everyone speaks early (which makes it easier to speak later), and the facilitator gets a temperature read of the room before the real work starts.

Then turn to the previous actions:

> *"Here's what we said we'd change last time. Let's go through each one. Did we do it? Did it help? If we didn't, why not?"*

For each action: acknowledge whether it happened, whether it worked, whether to carry it forward, drop it, or amend it. Keep this brisk; three to five minutes for the review.

What to say:

> *"Last retro we said we'd post a summary after planning. Did that happen? Did it help? Show of hands: should we keep doing it?"*

> *"We said someone would investigate the deploy pipeline this sprint. Who took that on, what happened?"*

> *"We didn't do this one. That's okay; I want to understand why, not blame anyone. Was it the wrong action, or did we not prioritise it, or did something get in the way?"*

What to watch for:

- No actions were completed. The most corrosive pattern in retrospectives. Name it directly: *"We're identifying the same things every retro and not acting on them. What's preventing us from making changes? That's the first thing we need to talk about today, before we generate more actions we won't do."* This conversation is more valuable than any observation phase.
- Dismissing the check-in. *"Can we skip the feelings stuff and get to the real work?"* No. The check-in creates the safety that makes the real work honest. Keep it brief, but keep it.
- A review that turns into a relitigation. *"We said that but it was the wrong thing."* Maybe, but relitigating past decisions is not the point. Note the disagreement and move forward.

Phase 2: Generate observations (15 min)

Set a timer for ten minutes. Everyone writes silently, one observation per sticky note. Require everyone to write at least two notes in each of the three categories:

- Start: things the team should begin doing
- Stop: things the team should stop doing
- Continue: things that are working and should keep happening

The requirement to write in the Continue column is deliberate. Teams that only focus on what's broken burn out; celebrating what's working reinforces good habits and keeps the session balanced.

After ten minutes, everyone places their notes in the appropriate column on the wall.

What to say:

> *"Ten minutes of silent writing. At least two notes in each column. Don't skip Continue; we need to know what's working so we don't accidentally stop doing it."*

> *"One observation per note. If you have a big observation, break it into smaller pieces."*

> *"Observations about the team's work, not about individuals. If it's about one person, save it for a one-on-one."*

What to watch for:

- All notes in one column. If Stop is full and Continue is empty, the team is exhausted or demoralised. Name it gently: *"It sounds like it was a tough sprint. Let's make sure we also name what's working so we don't accidentally stop doing good things."*
- Vague notes. *"Communication."* Push for specifics: *"Can you say more? Communication between whom, about what?"* A vague observation can't lead to an action; specificity is the job.
- Named-and-shamed notes. *"[Person] should stop being late to standup."* Redirect: *"Let's frame observations about the team, not individuals. What's the team impact of late standups? What could we change about the standup itself?"*
- Too few notes. People are being cautious. The safety isn't there yet. Options: extend the silent writing to fifteen minutes, switch to anonymous notes that the facilitator reads aloud without attribution, or run a private conversation with the team after the session to find out what's blocking honesty.
- The same complaints as last time. Note them. If the same things keep coming up, the actions to address them haven't been working; that's the real topic for today.

Phase 3: Discuss and cluster (20 min)

Read through all the notes. Cluster similar ones into themes. Then dot-vote: each person gets a fixed number of dots to stick on the items they care most about. Three dots each, place them on the *clusters* you think would most improve next sprint if changed. Discuss in order of votes, not size.

Cluster size and impact are different things. The biggest cluster is often a symptom: a topic that grabs the most attention because it's loudest, not because it's the highest-impact thing to change. Voting by impact lets the room route its time to the cluster whose change matters most.

For each cluster, the facilitator asks:

> *"Can someone explain what this cluster is about?"*

> *"Does everyone see it the same way, or are we hearing different things?"*

> *"What's the impact on the team?"*

> *"What could we actually do about it?"*

Time-box each cluster to five minutes. If a topic needs more time than that, it needs its own follow-up session, not a deeper retro conversation.

What to say:

> *"I'm going to cluster these. Anyone see themes? Tell me where I should move things."*

> *"Let's start with the biggest cluster. Who wants to explain what this is about?"*

> *"Thanks. Does anyone see this differently? I want to make sure we're not converging on one person's view."*

> *"Five minutes on this cluster. If we can't land a possible action, we park it and move on."*

What to watch for:

- One person dominating. Redirect with a question: *"Thanks. Does anyone see this differently?"* Go round the room if needed.
- The blame game. The conversation shifts to whose fault something was. Redirect to systems: *"Instead of who caused this, let's ask what about our process allowed this to happen. If this person left tomorrow, would the problem go away or land on someone else?"*
- Circular discussion. The team is orbiting the same point without reaching a conclusion. Intervene: *"I think we've understood the issue. Let's move to actions. What specifically would we change?"*
- The elephant in the room. Everyone is carefully discussing minor issues while avoiding the big one. If you sense this, name it carefully: *"Is there something we're not talking about that we should be?"* If the team isn't ready, don't force it, but the question plants the seed.
- Continue being skipped. Don't let the Continue column get rushed. Spending three minutes on *"these things are working and we should keep doing them"* is valuable positive reinforcement and changes how the team leaves the room.

Phase 4: Decide on actions (10 min)

From the discussion, identify one to three concrete actions. Not five. Not seven. One to three. More than that and nothing gets done.

Each action must have three things:

- A specific, observable change. *"Improve communication"* is not an action. *"Post a summary in Slack after every planning session"* is.
- An owner. One person whose name is written next to it. Not *"the team."* One name.
- A timeline. Usually *"by the next retro"* for sprint-level actions.

If there are more candidate actions than slots, use dot voting. Give each person two dots. Top voted items become the actions.

What to say:

> *"Maximum three actions. More than that and we don't do any of them."*

> *"Is this specific enough that we'll know next retro whether we did it? If not, sharpen it."*

> *"Whose name is on this? Not the team. One person. Who owns it?"*

> *"Do we actually believe this will help? If nobody believes in it, let's find something we do believe in."*

What to watch for:

- Actions that are too big. *"Rewrite the deployment pipeline"* is a project, not a retro action. Break it down: *"Two hours this sprint to list the three slowest steps in the deploy pipeline and what it would take to automate the slowest one. Owner: one person, named."*
- Actions nobody believes in. If the action feels forced, it won't happen. Better to commit to one action everyone believes in than three that nobody does.
- No owner. Every action needs a name. No name, no action.

- The same actions as last time. The problem isn't the observation; it's the follow-through. Make the action about follow-through itself: *"One person pings the team on Wednesday to check progress on the standup change, and reports back on Friday."*

Phase 5: Close (5 min)

End with a quick round:

> *"One word each. How are you feeling about the next sprint?"*

Thank the team. Remind them of the actions and who owns each one. Tell them you'll send the summary.

What to say:

> *"We committed to three things. [Name] owns [action], [name] owns [action], [name] owns [action]. I'll send a summary after the session. See you next retro."*

What to watch for:

- Ending on a downer. If the retro was heavy, end with a Continue item; remind the team what's working. The team should leave believing improvement is possible, not that everything is broken.
- Running over time. Retros that run long lose energy and goodwill. Hit the time. If a topic isn't done, carry it forward; don't extend the session.

Worked example

See Retrospectives at Every Scale, the Greenbox team running their sprint retros, a project retro after the Domain-Driven Design migration, and a quarterly retro at the end of their first full quarter. The moment the external facilitator stops running the sprint retros and the team runs them themselves is the moment the pattern becomes the team's own, not something being done *to* them. And the moment the quarterly retro produces a structural change instead of another list of complaints is the moment the scale earns its cost.

What Can Go Wrong

The silent room. Nobody writes anything. The safety isn't there. *Recovery:* Switch to anonymous notes; facilitator reads them aloud without attribution. Or ask specific questions: *"What was the most frustrating moment this sprint? What was the best?"* Or end the session and have a private conversation to understand what's blocking honesty. *Stop if:* Even anonymous notes produce nothing. The team doesn't believe the retro can change anything. That belief is the real problem; address it separately.

The venting session. Everyone complains but nobody proposes changes. *Recovery:* After ten minutes of venting, redirect: *"I hear the frustration. Now, what's one thing we could actually change? It doesn't have to be big. What's one small experiment we could try?"* *Stop if:* Every suggestion is immediately dismissed. Venting has become a culture, and the retrospective won't fix it alone. Raise it outside the retro with whoever can address the underlying issues.

The blame retro. The conversation keeps pointing at one person or team. *Recovery:* Invoke the Prime Directive. *"Everyone did the best they could with what they had. What about our process allowed this? If this person left tomorrow, would the problem go away or land on someone else?"* *Stop if:* The team refuses to frame it as systems. The blame is probably about something the retro can't resolve, a personal conflict that belongs in a mediated one-on-one, not a team meeting.

The "everything is fine" retro. Nobody has any feedback. *Recovery:* Rare but real. Ask directly: *"If you could change one thing about how we work, what would it be?"* If the answer is genuinely nothing, the team is in flow, celebrate and end early. *Stop if:* You suspect it's performative. *"Everything is fine"* from a team with visible problems means the safety isn't there, and pushing harder won't fix it. Investigate privately.

The same retro every time. The notes look identical to last sprint's. *Recovery:* Change the format. Change the facilitator. Change the question. *"What went well / what didn't"* gets stale; try *"What was the biggest surprise this sprint?"* or *"If we could replay the last two weeks, what would we do differently?"* *Stop if:* Even with a new format the output is identical. The team has a structural issue that retros can't touch; escalate.

The elephant everyone is avoiding. A big topic is visibly being talked around. *Recovery:* Name it carefully. *"I notice we're not talking about [thing]. Is that something we should be discussing, or is there a reason we're leaving it?"* Leave the team room to say no. *Stop if:* Naming it crashes the session. The topic is too big for a retro; handle it separately with whoever needs to be involved.

The action that keeps failing. The same action has been committed to for three retros in a row and keeps not happening. *Recovery:* Make the action smaller. *"Instead of 'automate the deploy pipeline,' let's commit to a two-hour spike this sprint. That's it."* If a small action can't happen either, investigate what's actually blocking it. *Stop if:* The blocker is outside the team's control. Surface that as the real finding; the action can't exist where the team has no power.

Next Steps

The session ends; the work begins.

Same day, the facilitator:

- Sends a brief summary to the team: the top themes, the committed actions, and the owner of each. Don't send the raw notes, they're for the team, not for outside consumption.
- Shares a photograph of the board, with the team only.
- Updates the action tracker, the simple shared document that lists *action / owner / status / retro date*. This is the cheap tool that makes compound improvement possible.

This week, the product owner (or equivalent):

- Checks in with action owners mid-sprint. Not to chase, but to remove blockers. *"How's the spike going? Do you need anything?"* The check-in signals that the actions matter.
- Protects the retro slot. When the sprint is busy and someone suggests cancelling the retro, don't. The sprint that's too busy for a retro is the sprint that most needs one. Cancellations teach the team that the retro is optional, and an optional retro has no power.

- Tracks cross-retro patterns. If the same type of action keeps appearing across sprints, the underlying problem is structural. Raise it explicitly in the next retro: *"We've committed to something like this four times now. What's the real problem?"*
- Shares upward carefully. If leadership wants to hear what the retros are surfacing, share themes and aggregate patterns, never raw notes or attributed quotes. The team's trust is worth more than any single piece of information leadership might find useful.

Ongoing, the team:

- Runs retros consistently. Every sprint, every time. The team that skips retros when things are busy is the team that most needs them.
- Rotates the facilitator. Different facilitators bring different questions, different energy, and different blind spots.
- Rotates the format. Three or four formats on rotation over the course of a quarter keeps the ritual fresh.
- Maintains the action tracker. A retro's actions are only real if someone can see whether they happened.
- Runs project retros at project close and quarterly retros at the end of quarters. Different scales catch different patterns; a team that only runs sprint retros misses the systemic issues.

Variants

Sprint Level (default). One sprint or iteration, 60-90 minutes, the whole team. Output: one to three small actions owned by team members. This is the workhorse, run every iteration, small stakes, small actions, compounding over time. The rest of this post describes it.

Start / Stop / Continue. The default Phase 2 format. Three columns: things to begin doing, things to stop doing, things to keep happening. Simple, fast, good for routine sprint retros.

Sailboat. A drawn picture instead of three columns. The team draws a boat, then captures wind (helping forces), anchors (slowing forces), rocks (risks ahead), and an island (the goal). Good for forward-looking reflection.

Timeline. Draw a timeline of the period under review. Everyone adds events, feelings, and observations along it. Good for project and quarterly retros where sequence matters.

Project retro (zoom out, 2 hours). A project or major initiative, a feature, a migration, a product launch. Use the Timeline format. The team maps the project chronologically, placing events, emotions, and turning points along the line. Phase 2 expands to thirty minutes; phase 3 to forty. Invite cross-functional participants if appropriate: stakeholders, adjacent teams, people affected by the project. Actions at the project level are often structural: *"in future projects, we start with an event-storming session,"* not *"someone will ping the team on Wednesday."* Output: lessons learned, structural changes, sometimes team-level changes.

Quarterly / organisational retro (zoom further out, half day). A quarter, or a cross-team initiative. Use a structured workshop: individual reflection (30 min) -> small-group discussions (60 min) -> full-group share-out and action planning (90 min). Focus on systemic issues: team structure, inter-team communication, shared tools and processes, strategic alignment. The actions should match the investment, structural changes, resource allocation, policy decisions. If the output of a half-day session is *"be better at communication,"* the session went wrong. Output: systemic changes, resource or policy decisions, leadership-level actions.

Post-incident post-mortem. A structured retro after an incident, focused on systems rather than people. The Prime Directive matters even more here than usual: name what happened, not who did it. For incidents that crossed enough systems that a conversation can't hold the shape, run Event Storming instead, a post-incident Event Storming *is* a retrospective with sticky notes on a wall instead of a conversation around a table.

Remote. A Miro or Mural board with the columns or picture pre-drawn, video call for the conversation. Slightly slower than in-person, but the structure transfers cleanly. Anonymous notes are easier remotely, use that affordance when the safety isn't there yet.

Level	Scope	Duration	Output
Sprint Level *(default)*	One sprint or iteration	60-90 min	1-3 small actions owned by team members
Project Level *(zoom out)*	A project or major initiative	2 hours	Lessons learned, structural changes, sometimes team-level changes
Quarterly / Organisational Level *(zoom further out)*	A quarter, or cross-team initiative	Half day	Systemic changes, resource or policy decisions, leadership-level actions

Sprint Level is the workhorse, run every iteration, small stakes, small actions, compounding over time. Project Level runs once per project, typically with a wider participant list. Quarterly Level is rare but powerful, and the actions should be at a scale that matches the investment.

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-retrospectives/.

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