MCARTOR OPERATIONS ADVISORY

LAW FIRM OPERATIONS — SYSTEMS THAT HOLD UP

A FREE TOOL FOR CLIENTS & COLLEAGUES

The Project Setup Tool

Set up any Claude project so it stays organized — a repeatable folder structure, the instructions that enforce it, and a cleanup routine that keeps it that way.

Twenty minutes at the start of a project. Five minutes a month after that. Nothing to install, nothing to buy — copy the prompts inside and make it your standard first step on every project you run.

Prefer one-click copy? An interactive version of this guide, with a copy button on every prompt, is at **mcartoradvisory.com**.

What this is

A twenty-minute setup you run once at the start of every Claude project, and a five-minute review you run every month. It gives you the same folder structure every time, a written set of filing rules, and project instructions that make Claude follow those rules in every future conversation.

It is free to use, yours to adapt, and there is nothing to install.

> **NEW TO CLAUDE PROJECTS?**
>
> Read the companion primer — *Projects in Claude: A Practical Primer* — first. It covers what a project is, how to create one, how to connect a folder, and which of a project's four parts does what. Fifteen minutes, and this tool will make more sense afterwards. Free at **mcartoradvisory.com**.

The problem it solves

Claude projects start clean and then quietly become a pile. It happens the same way every time:

- **Structure gets invented per project**, so nothing is ever where you expect it, and two projects doing similar work end up organized differently.
- **The rules live in your head.** You know draft contracts go one place and executed ones another. Claude doesn't — so every session guesses, and every guess is a small inconsistency you clean up later.
- **New material has nowhere to land.** Something arrives that doesn't fit, so it sits in the root folder. Then so does the next one.
- **Nothing prunes.** Old drafts sit beside current ones. Six months in, the most expensive question in the project is *"which version is real?"*

None of that is a discipline problem. It's a missing system.

What the tool does

1. Builds a consistent structure — adapted to the actual project

Every project gets the same fixed spine: a place for things that need action, a place for things that need holding, a place for the cross-cutting picture, and an archive. The rest is proposed from your actual workstreams — and **Claude proposes, you confirm, and nothing is created until you say yes.** Folders nobody asked for are the reason people stop trusting a filing system.

2. Writes the rules down where Claude reads them

Setup produces a filing guide and a set of project instructions. From then on, every session in that project files the same way without being reminded. This is the step most people skip, and it is the one that makes the difference.

3. Keeps the structure matched to the work

A monthly review prompt audits what's actually on disk against what the guide says — finding misfiled files, duplicate versions, and dead material, and flagging where the work has outgrown the structure it was given.

Two places knowledge lives

Before the folders, the split that decides where everything goes. A Claude project has two knowledge surfaces, and they behave differently:

	THE INSTRUCTIONS FIELD	THE CONNECTED FOLDER
Read	Automatically, every session	On request, when something needs it
Size	Small — a screen or two	As large as the work is
Changes	Rarely, deliberately	Constantly
Holds	Rules, boundaries, standing decisions	Drafts, reference, code, deliverables

The one-line test: if getting it wrong would be embarrassing or costly, it belongs in Instructions. If it's substantive and changes over time, it belongs in the folder. Instructions are read whether or not anyone thinks to look, which is exactly why they should stay short — every sentence that doesn't have to be there dilutes the ones that do.

The structure

Four folders are fixed and appear in every project:

FOLDER	WHAT IT'S FOR
`00-Inbox/`	Drop zone for things that need **action**. Processed only when you ask. Should be empty most of the time.
`01-Parking-Lot/`	Drop zone for half-formed ideas that need to **hold**. Never touched unprompted.
`05-Ecosystem/`	The whole-picture layer: the timeline, the handovers that started things, roadmap and priorities, and a log of decisions and why they were made.
`90-Archive/`	Superseded versions and dead ends. Moved here, never deleted.

Everything else lives in a numbered band between them, decided per project:

```
Project-Name/
├── 00-Inbox/
├── 01-Parking-Lot/
├── 05-Ecosystem/
│   ├── TIMELINE.md              ← the 30,000-foot dated view (generated)
│   ├── handovers/               ← the documents that started things
│   ├── roadmap-and-priorities/
│   └── decisions-log/
│
├── 10-Strategy/          ← direction-setting material
├── 20-Shared-Assets/     ← identity, templates, anything more than one area uses
├── 30-Workstream-A/      ← your actual streams of work,
├── 40-Workstream-B/        one area each
├── 60-External/          ← clients, prospects, stakeholders
├── 70-Product/           ← the thing you're actually producing
│
├── 90-Archive/
├── README.md             ← the fifteen-second map
└── FILING-GUIDE.md       ← the rules
```

WHY NUMBERS?

They sort in workflow order rather than alphabetically, and the gaps mean a new area added two years from now slots in as `35-` without renaming anything. Renaming a top-level folder breaks every path anyone has written down. Inserting a number breaks nothing.

WHY SO FEW?

Because a missing folder is a nuisance and an unused folder is corrosive. A missing one gets created the moment it's needed, with better information than you have today. An unused one teaches everybody that the structure is decorative — and once that lesson lands, filing discipline is gone. Start lean; let the project earn the rest.

Inside 05-Ecosystem: the two pieces people miss

`handovers/` **— the documents that started things.** Every project accumulates them: the original brief, the migration handover, the context file written to carry work across a gap. They record the trigger and the first thinking, and their value is that you can come back later and check where the project now sits *relative to the original intent*.

Without a named home they scatter — and worse, they drift into `90-Archive/`, because an archive is where anything unnamed eventually goes. That folder means *superseded and dead*. A handover is the opposite: live reference. **Handovers never get archived.** Name them `YYYY-MM-DD-handover-<topic>.md` so the folder reads chronologically.

One distinction worth stating, because it costs nothing now and is expensive later: a handover that is *still being edited* is a status file wearing a handover's name. It belongs with its workstream, cross-referenced from `handovers/`. The folder collects the **frozen** ones. Miss this and it slowly fills with live documents and stops being a record of origins.

`TIMELINE.md` **— the 30,000-foot view.** One line per event, oldest first: decisions, milestones, things that shipped, and the deadlines ahead. Not a log of every file that changed — a view you can read in one pass and know where a project stands and how it got there.

It is **generated, not maintained**, and that is the whole design. Everything it needs is already written down and already dated, in `handovers/`, `decisions-log/`, `roadmap-and-priorities/`, and the dated filenames across the areas. Claude rebuilds it from those. Three things follow: it **can't rot**, because nobody keeps it by hand; it **can't contradict the decisions log**, because it's derived from it — the log answers *why did we choose this*, at length, and the timeline is the index over it; and **refreshing is nearly free**, which is what makes it worth doing often.

> **WHEN IT REGENERATES**
>
> At the end of every inbox run and every review — triggers that already exist and already fire in proportion to how much work is happening, so the cadence follows the work instead of a calendar. A fixed weekly rebuild is worse than it sounds: over a busy stretch it's too coarse, and over a quiet one it produces a diff of nothing, which teaches you to stop reading it. Any time you want it sooner — after a heavy week, or before sending anything that quotes status — say **"refresh the timeline."** Nothing is typed into it by hand; anything that is gets lost on the next rebuild. Put the substance in a decisions-log entry and it appears in the timeline on its own.

How to set it up

1 Create the folder and the project

Make one folder on your computer for the project. Create a Claude project with the same name and connect the folder to it.

2 Run the setup prompt

*Already have a project with a folder connected? The five steps below are for a fresh start — the **Already have a project?** section further on covers the retrofit path.*

Start a session in the new project and paste this:

```
Help me set up this project's folder structure using a fixed spine plus
areas chosen for this specific project.

First, ask me about: what this project is for and whether it ends or is
ongoing; what the distinct workstreams are; what's the actual product or
deliverable versus supporting material; whether there's shared material more
than one workstream draws on; what already exists that has to land somewhere;
whether anything is confidential or has a retention rule; and who else
touches these files.

Then propose a tree. These four are fixed and always included:

  00-Inbox/        drop zone for ACTION — processed only when I ask
  01-Parking-Lot/  drop zone for HOLD — never touched unprompted
  05-Ecosystem/    TIMELINE.md (generated), handovers/, roadmap-and-priorities/
                   and decisions-log/
  90-Archive/      superseded versions — moved here, never deleted

Everything else goes in a numbered 10–80 band, in dependency order: direction-
setting material low, things that consume it higher. Leave gaps so new areas
can be inserted later without renaming anything. Keep the tree two levels deep
unless a third is clearly justified.

Before proposing any area, test it: would this material be MISFILED in every
existing area (required)? Will it keep producing files for MONTHS? Will it hold
enough that browsing beats searching — say five files and two subfolders? Is it
about ONE subject, statable without "and" or "misc"? It needs the first plus at
least two others. Bias toward fewer areas: a missing folder is a nuisance, an
unused folder teaches everyone to ignore the structure.

Present the tree in a code block with a one-line comment on every folder, then:
why it's shaped this way, what you considered and left out and which area
absorbs that material instead, and any judgment calls you made on thin
information.

Then STOP and ask me to confirm. Do not create anything until I say yes.
```

3 Confirm the structure

Read the proposal properly. Two things to push on:

- **Anything you'd add.** Ask Claude to run it through the same test out loud rather than just adding it.
- **Anything that looks aspirational.** If you can't name three files that will be in a folder six months from now, it's probably a subfolder.

When it fits, say so, and have Claude create the folders.

4 Generate the rules and wire them in

Paste this:

```
Now create the two root documents.

README.md — a fifteen-second map: one sentence on what this folder is, the
Claude project it's connected to, and a one-line description of each top-level
folder. One screen, no more.

FILING-GUIDE.md — the full rules: the complete tree with a comment per folder;
why it's shaped this way, including why the workstreams were split where they
were; the Inbox-means-act / Parking-Lot-means-hold distinction; the inbox
protocol; naming conventions; the rule that every file has one home and
everything else references it; any confidentiality or retention boundaries; and
a dated log of structural changes. Fill it from the decisions we actually made
— no placeholder language.

It must also state two things about 05-Ecosystem: that handover, continuity,
and context-brief documents go to 05-Ecosystem/handovers/ named
YYYY-MM-DD-handover-<topic>.md and are NEVER moved to 90-Archive/ — with the
note that a handover still being edited stays with its workstream and is
cross-referenced instead; and that 05-Ecosystem/TIMELINE.md is generated from
the dated material in the project, never hand-edited, and is rebuilt at the end
of every inbox run and every review, or on request when I say "refresh the
timeline".

Then draft custom instructions for this Claude project covering what the
project is, where the folder lives, the tree in brief, the naming rules, the
capture-surface rules, the requirement to log decisions to
05-Ecosystem/decisions-log/ with date and reasoning, and this rule:

  Any time you add, rename, merge, or remove a folder, rewrite FILING-GUIDE.md
  in the same response — the corrected file, written to the folder, with a
  dated line appended to its change log saying what changed and why. Rewrite
  README.md too if the top-level map changed. Do this without being asked; it
  is part of making the change, not a follow-up task for me. If the top-level
  areas changed, also give me a replacement block for this project's custom
  instructions, since that lives in a settings screen you can't write to.

  Regenerate 05-Ecosystem/TIMELINE.md at the end of every inbox run and every
  review, and whenever I say "refresh the timeline". Build it from the dated
  material already in the project — handovers/, decisions-log/,
  roadmap-and-priorities/, and dated filenames across the areas. One line per
  event, oldest first: decisions, milestones, things shipped, deadlines ahead.
  Not routine file edits. Never hand-edit it; it is overwritten each time.
```

Then do three things:

1. Paste the generated instructions into the project's **custom instructions**.
2. Save `FILING-GUIDE.md` into the **project's docs** as well as the folder, so sessions can read the rules even when the folder isn't reachable.
3. Test it: open a fresh session and ask *"where does [some file] go?"* A good answer cites the guide. A vague one means the instructions need tightening.

5 Load what you already have

Move existing material into `00-Inbox/` and say **"process the inbox."** Claude files each item, renames anything vague, and reports what went where. You'll see the system work — and you'll catch a wrong assumption on day one rather than in month three.

Already have a project?

You don't need to start over, and you shouldn't. A project with real material in it has something a new one doesn't: **evidence.** The files that exist tell you what the work actually is, which is better information than any answer you'd give to a setup questionnaire.

The retrofit runs in three passes — **read, propose, migrate** — and nothing moves until you've seen the whole plan. Two things worth knowing before you start:

- **Nothing is deleted.** Every operation is a move. Material that looks finished goes to `90-Archive/`, not to the trash.
- **Anything that can't be placed confidently goes to `00-Inbox/`**, not into a guessed home. A wrong guess is worse than an obvious pile, because a pile is visible and a wrong guess isn't.

If it would settle your nerves, duplicate the folder first and keep the copy until you're happy. It costs a minute.

Paste this into a session in the existing project:

```
Help me retrofit an existing project onto a standard folder structure. This
project and its folder already exist and already have material in them.

Work in three passes. Do not move anything during the first two.

PASS 1 — READ WHAT'S THERE
Inventory the connected folder: the full tree, file counts per folder, and the
date range of files in each. Read any README, guide, or notes file already in
the folder. Then tell me, in plain terms, what this project actually appears to
be about and what the distinct streams of work look like, based on the files
that exist rather than on what I've told you. If the evidence contradicts what
I've said, say so.

PASS 2 — PROPOSE THE TARGET
Propose a target structure using the standard spine — 00-Inbox/,
01-Parking-Lot/, 05-Ecosystem/ (with TIMELINE.md, handovers/,
roadmap-and-priorities/ and decisions-log/), and 90-Archive/ — plus numbered
10-80 areas that fit the work you actually found.

Before proposing any area, test it: would this material be MISFILED in every
existing area (required)? Will it keep producing files for MONTHS? Will it hold
enough that browsing beats searching? Is it about ONE subject, statable without
"and" or "misc"? It needs the first plus at least two others.

Then give me a MIGRATION MAP: for every existing file and folder, where it
lands in the target. Group it — "these 14 files -> 30-Website/content/" —
rather than listing every file separately. Flag these four things separately:

- files you can't place confidently, and what you'd need to know to place them
- handover, continuity, session-summary and context-brief documents, wherever
  they are now — including anything already sitting in an archive folder.
  These go to 05-Ecosystem/handovers/, renamed YYYY-MM-DD-handover-<topic>.md
  by the date the document was written. They are never archived. If one is
  still being actively edited, leave it with its workstream and say so
- material that looks superseded or finished, proposed for 90-Archive/
- duplicates and competing versions, with which one looks canonical and how
  you can tell
- anything already well organized that should be left exactly as it is

Where my existing structure is better than the standard for something specific,
keep mine and tell me why. This is not a demand that I conform to a template.

Then STOP and let me confirm. Move nothing yet.

PASS 3 — MIGRATE IN STAGES
Once I confirm, work in this order and report after each stage:

  1. Create the new structure alongside what already exists.
  2. Move the material you placed confidently.
  3. Move the archive candidates to 90-Archive/.
  4. Put anything you could not place into 00-Inbox/ — not into a guessed home.
  5. Remove the old folders that are now empty, and list what you removed.

Move, never delete. If a move would overwrite an existing file, stop and ask.

Finally: write README.md and FILING-GUIDE.md describing the structure that now
exists, with a change log entry dated today recording the migration and what
was left deliberately unchanged. Generate 05-Ecosystem/TIMELINE.md from the
```

```
dated material you just filed. Then draft replacement custom instructions for
this project for me to paste in.
```

What to expect

Pass 1 is worth reading slowly. Claude describing your project back to you from the files alone is a genuinely useful mirror. If the description is wrong, that's not a failure — it means the folder is telling a different story than you think it is, and that gap is the actual finding.

Push back in Pass 2. The migration map is a proposal, and you know things the files don't show. *"Leave that alone," "those two are the same thing,"* and *"that's dead, archive it"* are all normal responses.

Pass 3 is where a big project should go slowly. If the folder holds hundreds of files, run one area at a time — *"do stage 2 for the website material only"* — and check the result before continuing. There is no prize for doing it in one pass, and a staged migration is far easier to correct.

Afterwards, run the monthly review a week later rather than a month later. The first review after a migration always finds something.

The two rules that make it work

Inbox means act. Parking Lot means hold.

A file's location is an instruction. This one convention does more work than everything else combined, because it means you never have to explain what you want done with something — you just put it somewhere.

Move, never delete.

Superseded material goes to `90-Archive/` keeping its name. This is what makes decisive filing safe: the cost of a wrong call is a move, not a loss. People file confidently when filing is reversible.

Keeping it clean

Run this monthly, or sooner after a burst of activity:

```
Run a project review and cleanup on the folder connected to this project.

Read FILING-GUIDE.md first — it is the standard. Then audit the actual tree
against it. Do not move, rename, or archive anything yet. Report first.

Cover all ten checks and report findings under these headings:

1. INVENTORY — the current tree, file counts per area, and the date of the
   most recent file in each. Call out any area with no activity in 90 days.

2. STRUCTURE DRIFT — folders that exist on disk but not in the guide, and
   folders in the guide that don't exist on disk. Empty folders. Folders
   holding a single file. Folders holding so much that browsing has stopped
   working (roughly 25+ files at one level).

3. MISFILED — files sitting somewhere the guide says they shouldn't be, and
   files in the root that should be in an area. For each, name the correct
   destination.

4. DUPLICATES AND VERSIONS — the same file in two places, and multiple
   versions of the same document where it isn't clear which is current. For
   each set, say which one you believe is canonical and how you can tell.

5. ARCHIVE CANDIDATES — superseded drafts, completed one-off work, dead ends.
   For each, one line on why it's finished.

6. CAPTURE SURFACES — what's sitting in 00-Inbox and how long it's been
   there. A summary of 01-Parking-Lot by subfolder, with any parked items
   that now look ready to promote. Do not move anything out of the Parking
   Lot; just flag it.

7. NAMING — files that break the convention in the guide, especially undated
   files where versioning clearly matters, and names too vague to be findable
   ("notes.docx", "draft2", "Untitled").

8. STRUCTURE FIT — the judgment call, and the most useful part. Where has the
   work outgrown the structure? Areas that should split, areas that should
   merge, material that has quietly become its own workstream and deserves a
   top-level area, and areas that were created hopefully and never used.

9. ORIGIN MATERIAL — handover, continuity, session-summary and context-brief
   documents sitting anywhere other than 05-Ecosystem/handovers/, especially
   any in 90-Archive/. Those are misfiled by definition: the archive means
   dead, and these are live reference. Flag any that are still being actively
   edited — those stay with their workstream.

10. INSTRUCTIONS DRIFT — read the decisions-log entries since the last review.
   Which of them are standing rules rather than one-time calls? The log is read
   on request; the Instructions field is read automatically every session, so a
   standing rule left only in the log is a rule that quietly stops applying.
   Propose the exact lines to promote, and say what to cut to make room.

Then close with a RECOMMENDED ACTIONS list, ordered by value, with each item
marked [safe] — obvious, no judgment required — or [confirm] — a judgment
call I should weigh in on. Note anything that would change FILING-GUIDE.md.

I'll confirm what to execute. Nothing gets deleted; superseded material moves
to 90-Archive/ keeping its name.

Once I confirm, execute the moves and then do all of this in the same pass,
```

```
without waiting to be asked:

- Rewrite FILING-GUIDE.md in the folder so it matches the tree that now
  exists, and add a dated line to its change log for each structural change
  and the reason for it.
- Rewrite README.md if the top-level map changed.
- Regenerate 05-Ecosystem/TIMELINE.md from the dated material in the project.
- Only if the top-level areas changed: give me the updated FILING-GUIDE.md and
  a complete replacement block for this project's custom instructions, and tell
  me exactly where each one goes. If nothing structural changed, say "no paste
  needed" and skip this.
- Close with: what moved, what the guide says now that it didn't before, what
  I still need to paste, and the date of the next review.
```

Between reviews, this two-minute version catches most drift:

```
Quick project check: What's in 00-Inbox and how long has it been there? Any
files in the folder root that shouldn't be? Any area that's changed shape
enough that FILING-GUIDE.md is now wrong about it? Report only — don't move
anything.
```

You don't maintain the guide — Claude does

An updated structure with a stale filing guide is worse than no review at all, because the guide is what everyone trusts. Which is exactly why keeping them matched is the tool's job, not yours.

Both prompts above include the rule, and the project instructions you set up in Step 4 make it permanent: **any time the structure changes, Claude rewrites `FILING-GUIDE.md` in the same response** — the corrected file, saved to the folder, with a dated line in its change log recording what changed and why. It rewrites `README.md` too when the top-level map moves. You are never asked to go edit either one.

Two things still need your hands, and only these two:

WHAT	WHEN
Paste updated **custom instructions** into the project settings	Only when the top-level areas change — a settings screen is the one place Claude can't write to
Drop the updated **filing guide** into the project's docs	Same trigger, same reason

Most reviews change nothing structural, and a review that changes nothing structural asks nothing of you at all — the prompt tells Claude to say *"no paste needed"* and stop. Expect to touch settings once or twice a year, not monthly.

> **IF EVEN THAT IS MORE THAN YOU WANT**
>
> The second copy of the guide in the project docs exists so sessions can read the rules when the folder isn't connected. If you always work with the folder connected, keep the folder copy only and skip the project-docs copy entirely. You trade a little resilience for zero maintenance.

Making it standard practice

The tool only pays off if it's the first thing you do, every time. Three habits:

- **No project without a structure.** Setup takes twenty minutes and costs a fraction of what a disorganized project costs at month six.
- **The guide is the authority.** When a session files something oddly, the fix is usually a sentence in the guide, not a correction in the chat. Chat corrections don't survive the session; the guide does.
- **Put the review on the calendar.** Monthly, thirty minutes. A structure with no scheduled review reverts to a pile within a quarter.

Copyright, reuse, and independence

About McArtor Operations Advisory

McArtor Operations Advisory helps growing law firms fix the operational systems that stop scaling — intake, document management, matter workflow, and the knowledge that currently lives in people's heads. When a firm's execution stops keeping up with its ambition, the cause is almost never effort. It's a missing system.

This tool is one small piece of a larger method: systems that hold up because the rules are written down where the people — and now the software — doing the work can actually read them.

Bill McArtor · bill@mcartoradvisory.com · mcartoradvisory.com