How the publishing pod business runs.

One operating system for every company we work with: **company folders**, **shared production queues**, **temporary campaign lists**, acquisition paths as **fields, not folders**, and testing aimed at **the biggest current constraint**, never at whatever looks interesting. Sections 1 through 16 are the operating standard. Platform-specific facts live in Appendix A, because those facts may change and the operating principles should not.

THE THESIS

The operating standard

Shared craft and repeatable production workflows live in central queues.

Company strategy and campaign coordination live in creator folders.

Campaign work lives in a temporary launch list.

Acquisition paths are fields, not folders.

One operating system. One source of truth. Continuous testing. No random testing. Every test must attack the biggest constraint.

Where ClickUp sits

ClickUp is one tool in a system, not the system. Keeping the boundaries clean is what keeps it usable:

SLACK coordinates	**CLICKUP** assigns and tracks	**CRM** measures economics
DOCS preserve operating knowledge	**TESTING** attacks the bottleneck	

The point is not to run more experiments. The point is to remove the constraint that prevents the business from making more money or delivering better outcomes, with a system simple enough that the team actually uses it. One or two systems implemented completely beats ten implemented halfway `[$100M Playbook Marketing Machine, Page 10]`. The whole standard gets proven on one company (Roan) for 30 days before it touches anyone else.

PART 01

The shape: three spaces

One workspace, three spaces. Companies are folders inside one Pods space, which keeps every cross-company rollup intact and keeps the sidebar sane at any roster size.

```
▸ 🎯 HQ                              company-level operating layer
  – Meetings & Agendas               recurring agenda tasks w/ checklists (§12)
  – Offers Registry                  one task per offer: price, econ, ascension map, health
  – Issues & Decisions               Open → Discussing → Decided → Implemented (§10)
  – Scorecards                       the weekly numbers that get read out Friday
  📄 Docs: SOPs + template index     no separate library space until volume demands it

▸ 👥 PODS                            all company/creator delivery work
  ▪ Roan      ← the pilot pod; identical structure stamped from the template (§02)
  ▪ Ariel     ← stamped after the pilot proves the shape
  ▪ Justin

▸ 🏭 PRODUCTION                      shared craft + repeatable workflows, cross-company
  – Ads Engine                       ad concepts and performance (§05)
  – Design Studio                    design production, form-fed (§06)
  – Partner Content Queue            all standardized filming requests (§07)
  – Split Test Lab                   constraint-driven experiments (§08, §09)
```

Why work splits between Pods and Production: creator folders hold strategy, campaign coordination, and creator-specific work. Production holds the workflows that are identical no matter whose name is on the offer, each with one queue where the whole flow is visible. A designer looks at one queue, not seven folders.

The test for a central queue is whether the process is genuinely standardized: same request brief, same submission process, same review, same reminders, same reporting, same approval logic. When all six are true, the work belongs in Production, and pods get their view of it through filters rather than through copies. What keeps both worlds reporting as one company is the shared field set (§02), not shared statuses.

> **PODS ARE PEOPLE, NOT HIERARCHY**
>
> Who owns Roan is expressed by a **Pod Lead** field and assignees, not by structure. Pods own folders, specialists float across Production, and re-balancing pods never requires moving a single task.

PART 02

The unit model: companies, offers, paths

Roan is the company. His webinar, his workshop, and his $27 blueprint are not separate companies, and they are not separate permanent lists either. They are acquisition paths, expressed as **field values on tasks**, and the folder stays lean:

```
▪ Roan
    📌 Roan HQ (pinned doc)        brand kit, offers, entry paths, links, contacts
    – Ongoing                      recurring weekly cadence · ONGOING status set (§10)
    – Oct Webinar Sprint           TEMPORARY campaign list · CAMPAIGN status set (§04, §10)
    – Backlog & Ideas              parking lot · ONGOING status set
```

Where work goes

- Work with a **hard start and end date** belongs in a temporary campaign list.
- Work that **repeats indefinitely** belongs in Ongoing.
- **Shared craft and repeatable production workflows** belong in Production, in the queue that owns that workflow.
- **One-off ideas with no committed date** belong in Backlog & Ideas.

Do not create a campaign list for every recurring activity. And if an entry path someday develops a distinct fulfillment team, its own cadence, and its own backlog, it earns a permanent list then, and not before.

The offer fields

Three fields describe the money model on every piece of delivery and production work:

FIELD	DEFINITION
`Offer`	The offer this task **directly supports**: the thing being built or improved.
`Entry Path`	How the prospect enters the funnel: Webinar, Workshop, $27 Blueprint, or Multi-path.
`Ascension Target`	The next offer or action this path is designed to create.

> `ONE TASK, ONE ENTRY PATH`
>
> **If the deliverable materially differs by entry path, create separate tasks. If the deliverable is genuinely shared, use Multi-path.**
>
> Entry Path is single-select. Never enter comma-separated values.

For Roan:

TASK	OFFER	ENTRY PATH	ASCENSION TARGET
Blueprint checkout page	$27 Blueprint	$27 Blueprint	Booked-call offer
Webinar emails	Booked-call offer	Webinar	Booked-call offer
Workshop registration page	Booked-call offer	Workshop	Booked-call offer
Webinar sales script	Booked-call offer	Webinar	—
Workshop sales script	Booked-call offer	Workshop	—
Shared sales script	Booked-call offer	Multi-path	—

THE $27 BLUEPRINT IS AN ATTRACTION OFFER

It exists to turn strangers into customers and move qualified buyers toward the core booked-call offer. It remains an attraction offer, and the system still identifies the offer being directly built or improved on every task `[$100M Money Models, Page 66]`. The Offers Registry records each offer's ascension map, so the money model stays legible in one place.

The full field model

FIELD	VALUES (ROAN EXAMPLE)	WHAT IT MEANS
`Company/Creator`	Roan	The company lens. In Production, where company ≠ folder, this is the only company handle.
`Offer`	Booked-call offer / $27 Blueprint	The offer this task directly supports.
`Entry Path`	Webinar / Workshop / $27 Blueprint / Multi-path	How the prospect enters. Single-select.
`Ascension Target`	Booked-call offer	What this path is designed to create next.

FIELD	VALUES (ROAN EXAMPLE)	WHAT IT MEANS
Campaign/Launch	Oct Webinar Sprint	The specific event or promotion.
Workstream	Copy / Design / Ads / Tech / Sales / Operations	The craft lens.
Constraint	Booking rate	The constraint a test attacks. Lab tasks only.
On-Camera Partner	Roan, an affiliate, a customer	The person responsible for filming the asset. If the creator films everything, the value is the creator. This is what "which partners deliver" reports on.
Usage rights confirmed	Yes / No	Whether we may actually use the footage. Gates Approved.
Waiting on	Creator / On-Camera Partner / Vendor / Platform / Internal Dependency	Whose court the ball is in.
Health	On Track / At Risk / Blocked	A human judgment call, reviewed Monday.

Required fields, by task category

These must be filled for the task to be valid. ("Task category" is where the task lives and what it is for; the **Work Type** field in §15 is a separate axis that classifies what kind of thing it is.)

TASK CATEGORY	REQUIRED FIELDS
Pod delivery task	Company, Offer, Entry Path, Workstream, Assignee, Due date
Campaign-bound task	All of the above + Campaign/Launch
Production task	Company, Offer, Entry Path, Workstream, Assignee, Due date

TASK CATEGORY	REQUIRED FIELDS
Test task	Company, Offer, Entry Path, Constraint, the test-card fields (§09), Assignee, Due date
Partner content task	Company, Offer, Entry Path, Campaign, On-Camera Partner, the request brief and queue fields (§07), Assignee, Due date
HQ / admin task	Assignee, due date, and Work Type only

Conditional fields

These are required only when their condition is true, and left empty otherwise. Forcing a value where none applies produces data nobody can trust:

FIELD	REQUIRED WHEN
Ascension Target	The task supports an ascension path. Empty is a valid answer.
Campaign/Launch	The work is campaign-bound.
Constraint	The task is a Split Test Lab task.
On-Camera Partner	The task is a partner content request.
Usage rights confirmed	Before a partner video can reach Approved.
Waiting on	The status is WAITING. In the Partner Content Queue the status already says whose court the ball is in, so it is not required there.
Health	Campaign anchor task and Offers Registry entry only. Never per-task.

Assignee, due date, and Work Type are universal: exactly one owner, always a date, and a classification (§15) on every task in every queue. Priority uses ClickUp's native flag, set when it means something.

PART 03

How work connects: one master task

Every unit of work has exactly one home and one source of truth. Related work is connected, never copied.

> THE MASTER TASK RULE
>
> **A launch milestone, ad, design request, and experiment are different units of work. Each gets one master task. Related tasks are connected by relationships or dependencies, never duplicated.**

In practice, when an ad needs a designed asset:

- The **campaign task** ("Launch ads live for Oct Webinar Sprint") sits in Roan's campaign list. It is the launch-side unit of work.
- The **ad task** (`cr0042…`) sits in the Ads Engine. It is the master for that creative: single owner, definition of done is "live on platform with tracking verified."
- The **design task** sits in the Design Studio with its own owner and definition of done: "final files delivered to spec."
- The links: the campaign task is **waiting on** the ad task; the ad task is **waiting on** the design task. Each closes only when its own work meets the definition of done (§13), nobody can close past an unfinished dependency, and each owner is notified the moment the blocker clears.

Two status sets in the pods

Campaign work needs distinct approval, scheduling, and live states. Recurring work does not, so it does not carry them:

ONGOING LISTS (AND BACKLOG & IDEAS)

BACKLOG → TO DO → IN PROGRESS → WAITING → REVIEW → DONE

CAMPAIGN LISTS

BACKLOG → TO DO → IN PROGRESS → WAITING → REVIEW → APPROVED → SCHEDULED → LIVE → DONE

THE WAITING RULE

WAITING means work cannot continue until a dependency clears. The Waiting on field identifies whose court the ball is in.

Use `Waiting on = Creator` for the saved "Waiting on Influencer" view (§11). WAITING can be entered from any active state and returns to the state it came from.

Each workflow keeps its own status language. Ads, design, and tests each run their own sets, and every status in every queue is defined in the status dictionary (§10). What standardizes across the company is the meaning of fields, dates, and owners, and that is what all reporting runs on (§11).

PART 04

Campaigns: temporary lists from one template

Every campaign has a hard start and end date, becomes one temporary list created from the **Launch Runbook** template, and gets archived after the debrief. The template anchors on launch day: a task built as "T-minus 14" always lands 14 days out, weekends skipped. The template gets built during the pilot from a real Roan campaign, and every lesson upgrades it so the next campaign inherits everything learned.

PHASE	ANCHOR	COVERS
1 · Lock the offer	T-21 → T-16	Offer, price, stack, deadline mechanics, entry-path plan. Kickoff meeting task included.
2 · Build assets	T-16 → T-5	Pages, email sequences, ad batch briefed into the Ads Engine, social, partner pack. Copy runs In Progress → Waiting (creator) → Review → Approved.
3 · Tech & tracking	T-7 → T-2	Checkout, tags, automations, test purchases. Nothing ships without the test-purchase task closed.

PHASE	ANCHOR	COVERS
4 · Launch week	T-1 → close	Day-of runbook, send checks, stats checkpoints. 🚀 **Launch Day** and 🛑 **Cart Close** are Milestone tasks.
5 · Post-launch	close +1 → +5	Numbers recap, debrief, observations recorded (§08); those tied to the current constraint become test cards. Template upgrade task, then archive the list.

The campaign's anchor task carries the campaign's **Health**, set in Monday planning. Because launch-day and cart-close tasks are Milestones, one Calendar view filtered to milestones is the master launch calendar across every company, with zero upkeep (§11). Template tasks carry both a start and a due date as a matter of discipline; the platform reason lives in Appendix A.

PART 05

Ads Engine: the creative pipeline

One queue holds every ad for every company, worked in **batches of 3 to 5 concepts** per company per cycle. The loop: research feeds briefs, briefs become batches, batches ship weekly, results feed the next brief. The board makes that loop visible, on its own status language:

ANGLE → BRIEF → PRODUCTION → WAITING → REVIEW → READY → LIVE → ANALYZING → CONCLUDED

An ad task is created from the ad-brief template: objective, audience, angle, hook, proof elements, deliverables with aspect ratios, CTA and offer, reference ads, and a "what to test next" list so every brief seeds its successors. The ad task is the master for its creative and stays in this queue for its whole life; needed assets are linked design tasks (§03), never copies.

Creator approval: when the creator must approve the creative, move the task to **WAITING** and set `Waiting on = Creator`. Once approved, return it to **Review** for final internal verification. Creator approval blocks progress, so it gets the blocked state, not a review state.

Names that close the loop

Every creative carries an ID from task name into the ad platform, so performance rows map straight back to the task and batch: `cr0042_priceanchor_hook01_vid_roan_v01` (creative ID · concept · hook · format · creator · version). Fields on the task: the standard set (§02) plus Format, Angle, Hook, Batch (`ROA-B07`), and Results (ROAS / CPA / CTR / hook rate at conclusion, with a Verdict dropdown: Winner / Paused / Fatigued).

> **WINNERS SPAWN ITERATIONS, NOT AUTOMATIC TESTS**
>
> A new hook built on a winning angle is an **Iteration** (§08): it becomes `v02` in this queue, linked to its parent, so the winning angle's family tree stays queryable. It enters the Split Test Lab only if it meets the entry rule and attacks the current constraint. Friday's review (§12) walks everything in Analyzing: winners scale and iterate, losers conclude with a one-line reason, and next week's briefs open from what just won.

PART 06

Design Studio: the request process

One rule makes the design team scalable: **work enters through the form or it doesn't exist.** The form creates the task in the queue with every field filled, routes it to the design lead for triage, and applies the design-task template. No Slack-DM requests, no verbal specs, no mystery priorities.

NEW → TRIAGE → QUEUED → IN PROGRESS → REVIEW → REVISIONS → APPROVED → DELIVERED

The request form

- **Requester**, **Company/Creator**, **Offer**, **Entry Path**, and **Campaign** (mapped straight onto the standard fields)
- **Asset type**: ad static · ad video edit · email graphic · landing page section · sales page · social · slide deck · other
- **Specs**: dimensions / format pack, per the one-page specs cheat sheet linked on the form
- **Copy doc link** (required for anything with words: no copy, no design start)
- **References**: inspiration links or file uploads
- **Due date** and **tier**, plus a size guess: S / M / L

Turnaround tiers: internal targets, not promises

These tiers exist so "everything is urgent" becomes testable. They are **internal targets only** until the design lead confirms capacity and the pilot measures 30 days of actuals; nothing below gets published as a guarantee to anyone until it has survived a month of reality.

TIER	INTERNAL TARGET (PILOT)	MEANT FOR
🔴 Urgent	24 h	Launch-week fixes and live-campaign breakage. Costs a triage conversation; if everything's urgent, nothing is.
🟡 Standard	2–3 business days	Ad statics, email graphics, social assets. The default.
🔵 Build	Scoped per project	Landing pages, sales pages, brand systems. Gets a kickoff and two stage-gate reviews instead of a single due date.

Revisions: a Revision Round counter on the task, capped at two internal rounds plus one creator round. After each review, agreed changes get summarized in a pinned comment, and only the latest approved file is marked FINAL. **Connected work:** if the request serves an ad or launch task, the requester links it at intake (the master task is waiting on this one); the design task is never copied into a pod folder. One assignee per task, always.

PART 07

Partner Content Pipeline

Getting video out of partners is the bottleneck this pipeline exists to remove. The fix is not chasing people. It is a brief so complete they never need to ask, a page simple enough to finish on a phone, and one place where feedback lives.

THE BRIEF TEST

What exactly do I film, how should I film it, where do I upload it, and when is it due?

If the partner has to ask five questions, the brief is bad. Fix the brief, not the partner.

THE ACTIVATION POINT

Activation is not "partner received the brief." It is the *first usable video submitted and approved*.

Onboarding should drive every partner to that action, because activation is what creates downstream value [$100M Playbook Retention, Page 17] [$100M Playbook Retention, Page 20].

Where it lives: a central queue

The **Partner Content Queue lives in Production**, not in any creator's folder. The process is already identical no matter whose offer it serves: same request brief, same partner submission process, same review, same reminders, same reporting, same approval logic. That is exactly where a shared queue earns its existence.

It also keeps each list internally coherent. This pipeline has its own status flow, and a campaign list already has a different one; a single list cannot run two workflows. Pods get their visibility through **filters, not copies** — the queue filters by Company/Creator, Campaign, Offer, Entry Path, On-Camera Partner, Content type, Status, and Due date, so any pod can see exactly its own slice without a task ever being duplicated.

1 · The request template

Every request is created from one task template. The brief content lives in the task description; only what gets filtered or reported becomes a field:

```
PARTNER VIDEO REQUEST · ROAN-PVR-B03-V02
On-Camera Partner:        who is filming
Company/Creator:          whose business this serves
Offer / Entry Path / Campaign:
Content type:             testimonial · demo · VSL segment · UGC ad · organic
Batch ID / Video ID:      the filming session, and this video within it
Objective:                what this video must accomplish
Intended use:             where it will run: ad, email, sales page, organic
Hook:                     the first line, written for them
Script:                   linked doc, or verbatim below
Talking points:           bullets they can film straight from
Reference videos:         2-3 examples of what good looks like
Filming instructions:     setting, framing, lighting, audio, wardrobe
Requested format:         e.g. 9:16 vertical, 60-90 sec, .mp4
Upload destination:       the exact link
Due date:
Usage rights confirmed:   yes / no
Owner:                    one person on our side
```

Batches: one session, many videos

Creators film several videos in one sitting, so the queue separates the **filming session** from each **usable creative**:

The ID carries both: `ROAN-PVR-B03-V02` (company/creator · partner video request · batch, meaning the filming session · video within that batch).

Queue fields beyond the standard set: **Batch ID**, **Video ID**, **Content type**, **Requested format**, **Intended use**, and at approval, **Final file link**, **Asset ID**, and **Approval date**. One batch can be requested, reminded, and filmed as a unit while each video is reviewed and approved on its own.

2 · The partner page

The partner gets **one personalized link**, never a thread of instructions. The page carries the video request, the script and examples, a short filming checklist, the upload button, the due date, a "Need clarification?" button, and a confirmation after submission.

Mobile-first is the requirement, not a preference. They open it on their phone, film, and upload in under five minutes.

> TOOLING: KEEP IT NO-CODE, KEEP VIDEO OUT OF THE FORM
>
> Use a no-code form or upload page to start. Tally sends completed-form notifications straight into Slack and supports webhooks for later automation. Large video files should not ride as form attachments; Airtable's own guidance is to host the file elsewhere and submit the link instead. So the form captures the metadata and the link, and the video itself lands in one cloud folder. Confirm the upload limits of whichever tool you pick before the first real request goes out.

3 · What happens on submission

Manual during the pilot, automated only after 10 to 20 requests have run by hand:

1. The file is stored in **one cloud folder**.
2. It is attached or linked to the ClickUp task.
3. Task status changes to **Submitted**.
4. The owner is notified in Slack.
5. The partner receives a confirmation.

ClickUp's API supports attachments, so step 2 automates cleanly once the manual workflow is proven.

4 · One workflow

`REQUESTED` → `ACKNOWLEDGED` → `FILMING` → `SUBMITTED` → `INTERNAL REVIEW` → `REVISION NEEDED` → `APPROVED` / `CANCELLED`

> THE FEEDBACK RULE
>
> **Feedback lives on the ClickUp task. The partner gets one consolidated revision request, never scattered DMs.**

Cancelled is a real terminal state, not an abandoned task. A request is cancelled with a one-line reason when the offer changed, the campaign was cancelled, rights are unavailable, it duplicates another request, or the partner is unavailable. Without it, dead requests sit open forever and every report lies.

What Approved actually means

THE APPROVAL BAR

Approved means the video is usable, rights-cleared, and linked to the requesting ad, campaign, or content task.

Approval is a handoff, not a compliment. These are filled before the status moves:

```
BEFORE APPROVED · the handoff
Final file link:              the actual usable file, in the cloud folder
Usage rights confirmation:    confirmed, not assumed
Related ad or campaign task:  linked, never pasted (§03)
Asset ID:
Intended use:                 where it runs
Approval date:                stamps the activation point
```

5 · Reminders

Email or Slack during the pilot; automate once the manual version has proven itself:

- **48 hours before deadline** — a nudge with the link
- **Deadline morning** — a second nudge
- **After a missed deadline** — escalation to the owner
- **Weekly report** — requested, submitted, approved, overdue

What we track

THE PRIMARY METRIC

Days from request to approved, rights-cleared, usable creative — tracked by partner.

> The goal is not collecting files. It is producing usable ads fast enough to feed the marketing machine, which exists to continuously create usable advertising assets [$100M Playbook Marketing Machine, Page 9].

The supporting metrics exist to explain that number when it moves:

METRIC	TELLS US
`Submission rate`	How many requests ever produce a file.
`On-time rate`	Whether deadlines mean anything.
`Revision rate`	Brief quality. A rising revision rate is a brief problem, not a partner problem.
`Approval time`	Our own speed once the file arrives. Slow approval is our bottleneck, not theirs.
`Cancellation rate and reasons`	Whether we are briefing work that should never have been requested.
`Which partners consistently deliver`	Who to build the next campaign around.

First build: this week

1. **Create the ClickUp request template** with the fields and brief above.
2. **Create one mobile upload page** and the cloud folder behind it.
3. **Run it with one partner**, start to finish.
4. **Document every question they ask.** Each question is a defect in the brief.
5. **Fix the brief until they can complete it without help.** Then it is ready for the second partner.

Document, demonstrate, duplicate [$100M Leads, Page 198] [$100M Leads, Page 199]. This is the one build that does not wait on the pilot's open questions, because the bottleneck is live now.

PART 08

The testing philosophy: attack the constraint

THE TESTING PRINCIPLE

We are always testing, but we never test randomly. Every test must come from the company's biggest current constraint.

THE CONTINUITY RULE

Every active company must have either *one* active experiment or *one* documented constraint-removal action. It does not need multiple simultaneous tests.

Tests are continuous, but the queue is prioritized by the biggest constraint. Running several at once splits attention and muddies attribution; the goal is to remove the constraint, not to maximize test volume. Keep the money model deliberate rather than adding complexity prematurely [$100M Money Models, Page 155].

Testing is not an activity metric. Testing exists to remove the bottleneck limiting growth. Tests are never chosen because they are interesting, easy, or fashionable; they are chosen because the constraint demands them.

The operating sequence

1. Identify the current constraint.
2. Quantify its impact.
3. Form a hypothesis about what may improve it.
4. Run the smallest valid test capable of producing a decision.
5. Measure the predefined primary metric.
6. Make a decision.
7. Apply the learning to the operating system.

8. Choose the next test based on the next biggest constraint.

The constraint may be any of these, and naming it precisely is half the work:

`Lead volume` `Lead quality` `Booking rate` `Show rate` `Close rate`
`Average order value` `Cash collected` `Activation` `Fulfillment capacity`
`Customer results` `Retention` `Ad efficiency` `Checkout conversion`
`Creator approval speed`

> **FIX FIRST, TEST SECOND**
>
> If the constraint is a broken checkout, missing tracking, or a fulfillment failure, fix that first. Do not create a fake experiment to satisfy a testing quota. The goal is not more tests. The goal is more constraint removal.

Four Work Types, four routes

Most work is not an experiment. These four values of the **Work Type** field (§15) are the ones that decide whether something enters the lab, and routing them correctly is what keeps the lab honest and the queues fast:

TYPE	DEFINITION	EXAMPLES	WHERE IT GOES
Fix	A correction to something broken or missing.	Broken checkout · missing tracking · incorrect link · customer-facing error · compliance issue	The relevant workflow, at the right priority. No Split Test Lab entry.
Iteration	A new version based on an existing result.	New hook from a winning angle · new subject line from a prior winner · new headline from a known objection	The relevant workflow, linked to the prior task.
Experiment	A controlled comparison designed to answer a specific question under uncertainty.	Price test on the blueprint · webinar registration page A/B · booking-flow change	The Split Test Lab, meeting the full entry rule (§09).

TYPE	DEFINITION	EXAMPLES	WHERE IT GOES
Learning	A documented result or insight, including a signal that does not yet qualify as a controlled experiment.	"Show rate dipped this month" · "UGC seems to outperform" · a striking debrief number	Recorded as a Learning with its next action (§15). Learnings never become tests automatically.

THE ROUTING RULE

A new ad, new email, or new page is not automatically a split test. It becomes a split test only when it has a hypothesis, comparison, primary metric, decision rule, and a decision that will change future action.

PART 09

Split Test Lab: the workflow

THE ENTRY RULE

If the test cannot be connected to the current constraint and a measurable business outcome, it does not enter the Split Test Lab.

No test enters the lab without a control, challenger, primary metric, decision rule, and declared decision date.

The test card

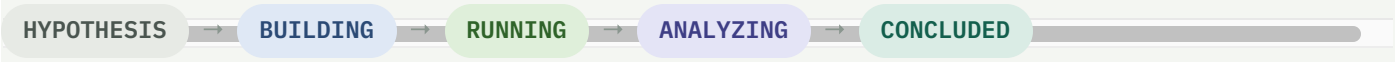
Every test carries all of this before it runs. The card is the task description template; copy and fill:

```
TEST CARD · TST-###                          owner fills every line before Running
Current constraint:      from the Monday constraint review (§12)
Baseline metric:         the number today
Est. economic impact:    what moving it is worth
Hypothesis:              If we [change], [audience] will [outcome], because [mechanism].
Variable being changed:  one variable
```

```
Control:                current version, linked
Challenger:             the test version, linked
Primary metric:         exactly one; never promoted after the fact
Secondary metrics:
Minimum sample/spend/duration:
Decision rule:          what result triggers which call; no peeking-and-calling-early
Owner:                  one person
Decision date:          declared before launch
Data source:
Next action if won:
Next action if lost:
PRIORITY                Impact(1–5) × Confidence(1–5) × Ease(1–5) = ___
```

Prioritization: Priority Score = Constraint Impact × Confidence × Ease of Execution, each scored 1 to 5. The highest-scoring test gets prioritized. No scoring system beyond this.

The statuses



Statuses describe where the test is. Two fields describe how it ended, and they are not the same thing:

FIELD	VALUES	ANSWERS
Result	Won · Lost · Inconclusive · Cancelled	What the data said.
Decision	**Rolled Out** (winning treatment implemented as the new standard) · **Killed** (losing or invalid test stopped) · **Inconclusive** (evidence insufficient, reason documented)	What we did about it.

A test also carries its **Learning** (one or two paragraphs of why we think it happened) and its **next action**, executed or scheduled before the card closes. Tests carry Test ID (`TST-041` , never reused), Company/Creator, Offer, Entry Path, Constraint, and Surface. Two saved views do the work: **Running Now** (board, grouped by company) for the weekly meetings, and **Learnings Library** (concluded tests, grouped by surface) that anyone writing a brief reads before starting.

PART 10

Status dictionary

WHY THIS PAGE EXISTS

Statuses are not labels. They are operating contracts. Without a shared definition, five people will interpret "Review" five different ways.

Every status in every queue, with what it means and what has to be true to leave it. Status names are standardized *within* each workflow. Each queue has its own status language, and every status is defined locally in this dictionary. Cross-company reporting never depends on status names. Two further rules govern the whole dictionary:

- **Not every task passes through every state.** Skip what does not apply: an internal campaign doc may go Review → Approved → Done without ever being Scheduled or Live.
- **Backlog exists only in the Ongoing and Campaign sets.** The Production queues each open with their own first state (Angle, New, Hypothesis) and deliberately do not use Backlog.

ONGOING LISTS & BACKLOG & IDEAS

STATUS	MEANS	EXIT WHEN
Backlog	Accepted work, not yet scheduled for a specific week.	It is scheduled into the current week.
To Do	Scheduled for this week; work has not started.	The owner starts the work.
In Progress	Actively being worked by its single owner.	An output exists that is complete enough to review.
Waiting	Work cannot continue until the specified dependency clears. **Waiting on** is required.	The dependency clears; the task returns to the state it came from.

STATUS	MEANS	EXIT WHEN
Review	Work is complete enough for review and the reviewer is tagged.	The reviewer accepts it, or returns it to In Progress with specific changes.
Done	Definition of done is satisfied and evidence is attached.	Terminal.

CAMPAIGN LISTS

STATUS	MEANS	EXIT WHEN
Backlog	Accepted campaign work, not yet scheduled for a specific week.	It is scheduled into the current week.
To Do	Scheduled for this week; work has not started.	The owner starts the work.
In Progress	Actively being worked by its single owner.	An output exists that is complete enough to review.
Waiting	Work cannot continue until the specified dependency clears. **Waiting on** is required.	The dependency clears; the task returns to the state it came from.
Review	Work is complete enough for review and the reviewer is tagged.	The reviewer accepts it, or returns it to In Progress with specific changes.
Approved	The required approver accepted the deliverable. For creator-facing work this is the creator.	**Approved exits to Scheduled, Live, or Done depending on the task type.** Not every campaign task is scheduled or goes live.
Scheduled	Approved and queued to go out at a set date and time.	It goes live on its date.
Live	The asset or change is active in the intended environment.	The definition of done is met and evidence is attached.

STATUS	MEANS	EXIT WHEN
Done	Definition of done is satisfied and evidence is attached.	Terminal.

ADS ENGINE

STATUS	MEANS	EXIT WHEN
Angle	A candidate angle or hook, tied to an offer and entry path.	It is selected into a batch.
Brief	The ad-brief template is being filled.	The brief is complete: objective, audience, angle, hook, deliverables, CTA.
Production	Copy and assets are being made; any design dependency is linked and open.	All deliverables exist to spec.
Waiting	Progress cannot continue until a dependency clears, most often the creator approving the creative. **Waiting on** is required.	The dependency clears; the task returns to Review for final internal verification.
Review	Creative is complete and the reviewer is tagged.	The reviewer accepts it. If the creator must approve, it goes to Waiting first and comes back here afterward.
Ready	Approved, named to convention, nothing blocking launch.	It is launched.
Live	Active in the intended environment with tracking verified.	The batch's read threshold or date is reached.
Analyzing	Spend or time threshold met; results being read against the verdict criteria.	A verdict is recorded.
Concluded	Verdict recorded (Winner / Paused / Fatigued) with a one-line reason; iterations spawned if any.	Terminal.

DESIGN STUDIO

STATUS	MEANS	EXIT WHEN
New	Submitted through the form, not yet triaged.	The design lead triages it.
Triage	Design lead is validating specs, copy link, tier, and sizing.	Accepted into the queue, or returned to the requester for missing inputs.
Queued	Accepted and sequenced; not started.	The designer starts.
In Progress	Actively being designed by its single owner.	A draft is ready to review.
Review	Draft is complete and the reviewer is tagged.	Accepted, or changes requested.
Revisions	Specific changes agreed and summarized in the pinned comment; round counter incremented.	The revised draft returns to Review.
Approved	The required approver accepted the deliverable; the final file is marked FINAL.	Files are handed to the requester.
Delivered	Final files delivered to spec and linked on the requesting task. Definition of done satisfied.	Terminal.

PARTNER CONTENT PIPELINE

STATUS	MEANS	EXIT WHEN
Requested	The brief is complete and the partner has their link.	The partner acknowledges it.
Acknowledged	The partner has confirmed they received it and will film.	Filming begins. Reminders fire from here.
Filming	The partner is producing. The ball is with them.	A file is submitted.

STATUS	MEANS	EXIT WHEN
Submitted	The file is in the cloud folder and linked on the task.	The owner picks it up for review.
Internal Review	The owner is checking it against the brief: objective, hook, format, length, usage rights.	It is approved, or one consolidated revision request is sent.
Revision Needed	One consolidated revision request has been sent to the partner.	A revised file is submitted.
Approved	The video is usable, rights-cleared, and linked to the requesting ad, campaign, or content task, with the handoff fields filled (§07). **This is the activation point.**	Terminal.
Cancelled	The request will not continue: offer changed, campaign cancelled, rights unavailable, duplicate request, or partner unavailable. Requires a one-line reason.	Terminal.

SPLIT TEST LAB

STATUS	MEANS	EXIT WHEN
Hypothesis	Tied to a documented constraint, with a complete hypothesis.	It passes the entry rule and is prioritized.
Building	Assets, tracking, audience, and measurement are being prepared.	The test card is complete and everything is ready to launch.
Running	Launched and collecting data. **No test runs without a completed test card.**	The minimum sample, spend, or duration is met, or the planned period ends.
Analyzing	The decision rule has been met; the primary metric is being read.	A decision is made. **No test concludes without one.**

STATUS	MEANS	EXIT WHEN
Concluded	Result, Decision, Learning, and next action are documented, and the next action is executed or scheduled.	Terminal.

HQ · ISSUES & DECISIONS

STATUS	MEANS	EXIT WHEN
Open	The issue or decision has been logged.	The relevant people pick it up.
Discussing	The relevant people are evaluating it.	A decision is reached.
Decided	The decision is recorded.	The decision is applied.
Implemented	The decision has been applied to the business, task, SOP, or system.	Terminal.

PART 11

Views and reporting: fields, not statuses

Because the workflows keep their own status languages, nothing cross-company ever reports on status. The rollup layer runs on what is uniform everywhere: **due date, assignee, Company/Creator, Offer, Entry Path, Waiting on, and Health.** Five saved views carry the whole company through the pilot, no dashboards required:

VIEW	SHOWS	WHO LIVES IN IT
📅 **This Week**	Every open task due in the next 7 days, all spaces, grouped by Company/Creator	The Monday meeting; you, daily
🔥 **Overdue**	Anything past due, grouped by assignee	Monday meeting, 60 seconds; should trend to zero

VIEW	SHOWS	WHO LIVES IN IT
⏳ `Waiting on Influencer`	Filtered to `Waiting on = Creator`, sorted by time waiting. Catches pod work and ads awaiting creator approval in one place	Pod leads; the nudge ritual
👤 `My Tasks`	Your tasks, grouped by due date	Everyone's home screen; the standup that needs no meeting
🚀 `Launch Calendar`	Every launch day and cart close, all companies, from Milestone tasks	Everyone; pinned everywhere
🎬 `Partner Videos`	The partner pipeline by status, overdue flagged. The source for the weekly requested / submitted / approved / overdue report (§07)	Request owners; Friday review

Because the waiting view filters on a field rather than a status, it works across every queue and at workspace level, which is exactly why reporting runs on fields. Campaign and company **Health** reads out in Monday planning from the campaign anchor tasks and the Offers Registry: a one-glance row of On Track / At Risk / Blocked across everything active. Dashboards and reporting polish arrive after the pilot (§15); build mechanics live in Appendix A.

PART 12

The weekly rhythm

Two standing meetings run directly from the views, both existing as recurring agenda tasks in HQ with these checklists baked in. The cadence work itself (Roan's weekly content and emails) lives as recurring tasks in his Ongoing list.

MON
45 min

Weekly planning

Walk 📅 **This Week** company by company, clear 🔥 Overdue, check the Launch Calendar for what's inside 21 days, drain Issues & Decisions, set Health on active campaigns and companies, confirm the week's ad batch and design priorities. Then the constraint review, per active company:

```
CONSTRAINT REVIEW · MONDAY
1. Current constraint
2. Evidence it is the constraint
3. Baseline metric
4. Economic impact of removing it
5. Highest-priority experiment or corrective action
6. Owner
7. Decision date
8. What result would change our direction
```

Everyone leaves knowing what ships by Friday and which constraint the week is attacking.

FRI
30 min

Weekly review

Read the Scorecards, walk Ads Engine "Analyzing" (scale / iterate / conclude), triage the design queue for next week, read the partner video report (requested, submitted, approved, overdue), and log anything that slipped with the reason into Monday's agenda. Then the results review:

```
CONSTRAINT REVIEW · FRIDAY
1. What tests concluded?
2. Did the result address the constraint?
3. What changed because of the result?
4. What is now the biggest constraint?
5. What test or action comes next?
```

WEEKLY
10 min

Hygiene audit

The system owner runs this before Monday planning, so the views the meeting runs on are true:

- Tasks missing an owner, a due date, or a **Work Type** (§14)
- WAITING tasks missing **Waiting on** or a follow-up date, and stale Waiting on values on tasks no longer waiting
- Decisions missing implementation links
- Learnings missing next actions
- Experiments missing complete test cards, or running without one
- Partner videos missing rights confirmation or a related task
- Automated notifications firing incorrectly
- Commitments living in Slack with no ClickUp task behind them
- Tasks nobody has touched in 14 days

Daily needs no meeting: 👤 **My Tasks** is the standup. And the lab is never filled with random ideas to appear productive; if the constraint review didn't surface it, it waits.

PART 13

The working rules

THE DEFINITION OF DONE

A task is not complete because someone worked on it. It is complete only when the stated output is delivered, approved where required, usable by the next owner, and linked to the relevant evidence or final asset.

This applies to campaign tasks, ads, design requests, experiments, launch milestones, technical tasks, and weekly recurring work. No exceptions by work type.

- **One master task per unit of work.** Related tasks are connected by relationships or dependencies, never duplicated (§03).
- **Every task has one assignee, a due date, and a Work Type.** No exceptions; the views run on it (§14).
- **Automate transitions, never judgment.** Machines expose state changes, route work, and remind people. People classify, interpret, and decide (§14).
- **Fields are required by work type** (§02), and nowhere else. HQ and admin tasks carry an assignee and a date, nothing more. The field registry is owned by the system owner; nobody else creates fields.
- **One task, one entry path.** Separate tasks when the deliverable differs by path; Multi-path when it is genuinely shared. Never comma-separated values.
- **Statuses are local, fields are universal.** Each queue's statuses belong to that queue and mean exactly what the status dictionary says (§10); reporting never depends on them.
- **Requests through forms.** Design work that arrives via Slack gets a polite link to the form.
- **Templates or it's twice-work.** Launches, ad briefs, design requests, test cards, meeting agendas: all stamped, never rebuilt.
- **Tasks stay put.** A task lives its whole life in its home list; progress is a status change, never a list move.
- **Route work by type.** Fixes and iterations go to their workflow; only constraint-tied experiments enter the lab; observations are recorded, not promoted (§08).
- **Archive completed work.** Finished campaign lists archive after the debrief; archives stay searchable. Nothing is deleted.

PART 14

Task classification and automation

The rest of this standard defines the workflows. This part defines how ClickUp tells a fix from a decision from an experiment, and which parts of that a machine is allowed to do.

THE CORE RULE

> **Automate state changes, routing, reminders, notifications, and required fields. Do not automate judgment.**

The Work Type field

A single-select field on **every task in every queue**, alongside assignee and due date. It answers "what kind of thing is this," which is a separate question from where the task lives:

WORK TYPE	MEANS	WHERE IT GOES
`Standard Task`	Normal execution work with an owner, deadline, and definition of done.	Its own workflow.
`Fix`	A correction to something broken or missing: broken checkout, missing tracking, incorrect link, customer-facing error, compliance issue.	The relevant workflow, at the right priority. Never the Split Test Lab.
`Request`	A new piece of work requiring an owner and a deadline.	A ClickUp task before execution begins. No task, no work.
`Blocker`	Work that cannot proceed without intervention or dependency resolution.	Stays in its workflow in WAITING, with a linked task, owner, cost of waiting, and the person or dependency that can unblock it.
`Decision`	A choice that changes priorities, execution, offer structure, process, or resource allocation.	HQ → Issues & Decisions, recording the decision, owner, effective date, and what changes.
`Learning`	A documented result or insight from a campaign, ad, experiment, partner-content request, sales process, or operational event.	Recorded with what was tried, what happened, what we believe it means, and what happens next.
`Iteration`	A new version based on a previous result.	The relevant workflow, linked to the prior task.

WORK TYPE	MEANS	WHERE IT GOES
Experiment	A controlled comparison under uncertainty.	The Split Test Lab, only when it attacks the current biggest constraint and has a complete test card (§09).

The automation principle

> THE AUTOMATION PRINCIPLE
>
> **Automations expose meaningful state changes. They do not replace judgment.**

Automate only these nine things:

1. Required fields based on workflow or status
2. Notifications when tasks enter Review, Waiting, Approved, Live, Analyzing, or Concluded
3. Reminders for upcoming and overdue deadlines
4. Escalations for tasks waiting beyond the defined threshold
5. Slack notifications for urgent blockers, creator approvals, concluded experiments, and approved partner content
6. Creation of follow-up tasks when a decision or learning requires action
7. Updating task status when a form submission or approved upload occurs
8. Linking related tasks where the platform supports it
9. Weekly summary reports for blockers, decisions, learnings, experiments, and partner content

> NEVER AUTO-CLASSIFY FROM KEYWORDS
>
> Do not automatically generate a Decision, Learning, Blocker, or Experiment because certain words appeared in a task or a Slack message. A Slack `[BLOCKER]`, `[DECISION]`, or `[LEARNING]` tag is a prompt for a human to create the task, never a trigger that creates it. Keyword auto-creation produces duplicates and garbage; classification is judgment, and judgment stays with the operator.

Automation rules by trigger

WHEN A TASK ENTERS WAITING

REQUIRE	THEN
Waiting on · Blocker description · Person or dependency that can unblock it · Follow-up date · Cost of waiting	Notify the task owner. If `Waiting on = Creator` and the task exceeds 48 hours, surface it in the creator report.

WHEN WORK TYPE = DECISION

REQUIRE	THEN
Decision question · Options considered · Decision · Decision owner · Effective date · What changes · Linked implementation task, SOP, or project brief	When the decision is marked Implemented, notify the relevant Slack channel with a link.

WHEN WORK TYPE = LEARNING

REQUIRE	THEN
What we tried · Result · Interpretation · What changes · Next action · Source task or campaign	A Learning cannot be closed without a next action, or an explicit "No action required" explanation.

WHEN WORK TYPE = EXPERIMENT

REQUIRE	THEN
Current constraint · Baseline · Estimated economic impact · Hypothesis · One variable changed · Control · Challenger · Primary metric · Minimum sample, spend, or duration · Decision rule · Owner · Decision date · Next action if won · Next action if lost	No experiment may enter Running without the complete test card (§09).

WHEN A PARTNER VIDEO IS SUBMITTED

REQUIRE	THEN

REQUIRE	THEN
File link · On-Camera Partner · Video ID · Usage rights confirmed · Related campaign or ad task · Review owner	Move the task to Internal Review and notify the owner. At Approved, require the final usable file link and intended use: approved means rights-cleared, usable, and connected to the next task (§07).

Automation phasing

Pilot manually for the first 30 days. Each phase earns the next:

PHASE 1
manual

Prove the process by hand

Templates, required fields, manual status changes, manual Slack links, manual weekly reports.

PHASE 2
10–20 runs

Automate the obvious transitions

Reminders, notifications, escalations, form-to-task creation, partner-upload updates. Only after 10 to 20 real examples have run manually.

PHASE 3
proven

Automate the reporting layer

Automated Slack summaries, follow-up task creation, CRM and ClickUp integrations, dashboards, API scripting. Only after the workflow is proven.

THE PHASING RULE

Do not automate a process before it has been repeated enough to expose its edge cases.

The division of labor

ClickUp does this automatically

Makes work **visible, routed, reminded, and reportable.**

People still decide this

- What the constraint is
- Whether something is a fix, iteration, or experiment
- What the result means
- What action follows
- Whether a decision is worth implementing

PART 15

The Roan pilot: 30 days, minimum build

THE PILOT OBJECTIVE

Can Roan's team plan, execute, review, and learn from work without using side systems?

The objective is not "build ClickUp." The system gets proven manually on one company before anything is automated, because automating a bad process just lets you fail faster. Build only this:

1. **Roan folder.** Hub doc, Ongoing, Backlog, stamped clean. Active work migrated in; the old mess archived, not reorganized.
2. **One real campaign.** The active launch as a campaign list, run for real. When it's done, the cleaned-up list becomes the Launch Runbook template. Reality first, template second.
3. **Ads Engine.** The queue, its statuses, the ad-brief template, the naming convention.

4 **Design Studio.** The queue, the request form, the internal turnaround targets, the revision cap.

5 **Split Test Lab.** The board and the test card. Empty until the constraint review sends something in.

6 **Core views.** The five saved views, built and pinned.

7 **Weekly meetings.** Monday planning and Friday review as recurring agenda tasks with the constraint-review checklists, walked from the views, decisions captured as tasks in the meeting.

8 **Hygiene audit.** Ten minutes weekly by the system owner, logged as done or not.

9 **Partner Content Queue (§07).** The central queue in Production, the request template, one mobile upload page, one cloud folder, run with one partner. This is a live bottleneck, so it starts this week rather than waiting on the questions below.

Explicitly out of scope for the pilot

- Dashboards
- Advanced automations
- API scripting
- Influencer guest access
- Top One tests
- Advanced reporting polish
- Any workflow not required for the active Roan campaign

Pass criteria

Four straight weeks in which all of the following were true:

- Both weekly meetings ran from ClickUp views.
- The hygiene audit found fewer than five violations.
- Every design request entered through the form.
- No private side lists.
- Every experiment decision was documented.
- **The current constraint was documented every week.**
- **Every active test could be traced to that constraint.**

The last two exist so the pilot cannot pass on administrative compliance while the team tests random ideas. Pass all seven, and the structure has earned its rollout: stamp Ariel's and Justin's folders from what the pilot proved, then bring in the deferred pieces aimed at a process that has already proven itself by hand.

Before rollout: effort and capacity

For the Roan pilot, **task count is enough**. One company, one campaign, a small team, and a queue you can eyeball. That stops being true at three companies.

So before rolling to Ariel and Justin, add **task estimates or another capacity method** for design and any other constrained team, so the queue can be planned against real capacity instead of hope. ClickUp's Workload view measures assigned work by task count, time estimates, or a capacity field, and the controls vary by plan and role (Appendix A). Pick the method during the pilot's final week, when 30 days of real design throughput are on record and the numbers can be set from evidence rather than guesswork.

PART 16

Open questions before day one

Q1 Who owns the system?

The hygiene audit, the field registry, and the template library need one named owner from day one. Who is it?

Q2 Which Roan campaign anchors the pilot?

Build item 2 runs a real launch inside the system. Which upcoming Roan campaign is it, and when does the 30-day clock start?

Q3 Who is the design lead?

They triage the queue, and the turnaround targets stay internal until they confirm capacity and 30 days of actuals are measured. Who signs?

Q4 Copywriters: pod or central?

Copy currently lives inside pods (core deliverable, needs the creator's voice). If copywriters are actually a shared service like design, they get a Copy Desk queue in Production instead. Which is true?

APPENDIX A

Platform facts and build notes

Everything above is how the company works and should survive any tool change. Everything below is true of ClickUp as of August 2026, verified against its current docs, and may change. When a fact here changes, the operating standard does not.

Plan and pricing

The pilot runs on the current plan (Unlimited or better): saved List and Calendar views, forms, templates, relationships, and dependencies are all available there. The **Business tier ($12/user/mo annual)** is the unlock for dashboards, Workload views, and automation conditions; it lands at rollout, not during the pilot.

Dashboard and Workload metering

Below Business, dashboards and Workload views have **lifetime** use caps (100 uses on Unlimited), and merely opening a dashboard consumes a use that never resets. This is why the pilot is views-only and dashboards are out of scope until the Business upgrade.

Workload measures assigned work by **task count, time estimates, or a capacity field**, with per-person capacity limits set by day, week, or month. Which controls are available varies by plan and role, so confirm the method against the plan at the time of the rollout decision (§15), not before.

Why cross-company views filter on fields

ClickUp's docs contradict themselves on whether a custom status that exists in only some lists can be filtered in workspace-level views. Custom *fields* filter cleanly at every level, which is why the waiting view keys on `Waiting on = Creator` rather than on a status, and why all cross-company reporting runs on fields. It also makes that view catch ads awaiting creator approval, which a status-based version would miss.

Seven status sets, configured per list

ClickUp sets statuses per space, folder, or list, and a list can override its parent, so the Ongoing, Campaign, Ads, Design, Partner Content, Lab, and HQ sets coexist without conflict. Two consequences: same-named statuses in different lists are a documented source of "missing status" confusion when filtering, and status colors follow the list a task was created in. Both are reasons reporting runs on fields (above).

Build them once as **Status templates** (saved, reusable status sets, available on every plan), then apply the right one per list: right-click the list → List statuses → use custom statuses → pick the template. That turns seven sets into a one-time build plus a few clicks per list, rather than seven rebuilds.

One catch: a status template is a copy, not a live link. Editing one changes only the location you edited, and every other location that used it silently detaches and keeps its statuses relabelled "Custom." There is no propagate-on-edit, so changing a set later means touching each list that uses it. Settle the seven sets during the pilot, not after.

Dependency behavior

Dependency-driven date shifting only fires when the blocking task has a due date and the blocked task has a start date, which is why every templated task carries both dates (§04). Applied templates only reproduce dependencies between tasks inside the same template; links from a campaign task to Production tasks are created at brief time, not baked into templates.

Moving tasks between lists

Moving a task between lists with different status sets throws an interactive remap prompt, and a careless answer to the fields prompt can strip reporting fields. Hence the "tasks stay put" rule (§13), and the standard fields being defined at workspace level so they exist everywhere a task could ever

land.

Guest permissions

ClickUp cannot scope a guest to a space, only to folders, lists, and tasks. If creators (or their teams) ever get direct visibility post-pilot, the clean patterns are a folder-scoped guest seat or a read-only portal doc with embedded filtered views. Out of pilot scope either way.

Forms

Real form use needs a paid plan (Free allows one form and can't share it externally). Targeting a list, auto-assigning, and applying a task template on submission are standard form settings; conditional show/hide logic is effectively enterprise-tier, so the request form stays flat with a linked specs cheat sheet. Verify auto-assign and template-on-submit in-product on day one.

What the API can and cannot build

Verified against ClickUp's published OpenAPI specifications (v2, 83 endpoints; v3, 23 endpoints), August 2026. Auth is a personal `pk_` token sent as a bare `Authorization` header with no `Bearer` prefix, and the rate limit is 100 requests per minute on Free, Unlimited, and Business.

Scriptable: spaces, folders and subfolders, lists, tasks (assignees, dates, priority, subtasks, and the built-in Milestone type), setting custom field values, dependencies and task links, webhooks, and views — List, Board, Calendar, Table, Timeline, Gantt — including their filters, grouping, and sorting.

No API at all, so UI-only: creating or applying **custom statuses** at any level; creating **any template**; **Form views**; **dashboards**; **automations** (which cannot even be listed, let alone created); **recurring-task settings**; marking a field required; and every ClickApp outside the nine the Space endpoint exposes. Statuses are the one to plan around permanently: ClickUp formally closed that feature request in August 2025 as "not on the roadmap."

Custom fields: verify before assuming. Field creation is undocumented and officially "create them in the app," but one August 2026 report has `POST /v2/list/{list_id}/field` returning success with dropdown options defined. Single unverified source, so treat it as a five-minute test against a throwaway list rather than a plan. Field *updates* are confirmed blocked either way, so options get defined correctly the first time. This is the highest-value unknown in the whole audit: if it works, field creation leaves the manual column entirely.

The rollout mechanism. Templates cannot be created by API, but they can be *applied* by API, and the apply call carries exactly the pieces the API otherwise cannot build: `old_statuses`, `custom_fields`, `include_views`, `automation`, `recur_settings`, plus date remapping. So one canonical creator

folder is hand-built once, saved as a Folder template, and each additional creator becomes a single call to `POST /v2/space/{space_id}/folder_template/{template_id}`. That is how Ariel and Justin get stamped from the proven Roan structure.

The prerequisite that makes or breaks it: a Folder template only carries statuses the folder *owns*. Statuses inherited from the parent Space are explicitly excluded. So the canonical creator folder must have custom statuses activated on the folder itself, not inherited, or every stamped copy arrives with the wrong workflow. Same logic applies to subfolders inheriting from their parent folder.

Three things to settle with a throwaway space before the rollout is designed around them, all cheap and all decisive: whether `POST /v2/list/{list_id}/field` really creates a dropdown with its options; whether `include_views` carries a Form view through a template; and whether `old_statuses` reproduces list-level status overrides or only the parent set (its description, "Import tasks' status settings," is genuinely ambiguous).

One trap: API actions fire ClickUp's normal notifications, so a scripted build will spam everyone who can see the location. Build the structure before adding members.

Status: Pod OS, the Roan Pilot Operating Standard. Nothing has been built or changed in ClickUp yet; the pilot begins when Q1 and Q2 are answered.

Citations: $100M Money Models p. 66 (attraction offers and offer identification) · $100M Playbook Marketing Machine p. 10 (complete beats half-built).