AlumnAI

A BUILD GUIDE FROM ALUMNAI

The Carousel Engine

Building an On-Demand Instagram Carousel Poster

A complete, working end-to-end system that turns any topic into a fully branded, multi-slide Instagram carousel — AI-written copy shaped to the angle you choose, AI-generated artwork, and a one-click publish, with a preview step before anything goes live.

TECHNICAL BUILD GUIDE · V1

Any Topic · 8 Content Types · Claude API · FLUX · Buffer API

ALUMNAI.CO.UK

CONTENTS

What's inside

Ten sections, one working pipeline — built once, ready for any topic and any content angle. Follow them in order; each one builds on the last.

STATUS — V1

This edition documents the pipeline as built and runtime-verified: dry-run tested end to end, then confirmed live on Instagram. The artwork step (Section 07) was swapped from OpenAI's `gpt-image-1` to FLUX 1.1 Pro after the original output read as visually generic — see Section 03 for the reasoning. Buffer's `createPost` mutation (Section 08) is documented against its live schema, not assumed.

BEFORE YOU START

This guide assumes basic comfort with Git, environment variables, and reading JSON. You don't need to be a developer — but you do need to be willing to follow instructions precisely. Every code sample is complete and working, not pseudocode.

SECTION 01

Why build this

A branded, on-topic Instagram carousel — on any subject, without opening a design tool

> *The problem is rarely coming up with something to post. It's turning that idea into five polished, on-brand slides — every time, without opening a design tool.*

A carousel that looks hand-designed, written in a voice that actually matches the topic, and posted in the right format — that's normally a 30-to-60-minute job: draft the copy, find or generate an image, lay out the text so it doesn't collide with the artwork, export five slides at the right size, upload them one at a time. Do that daily or even weekly and it's the first thing that slips.

This pipeline collapses that job to three inputs — a **topic**, a **page count**, and a **content type** — and produces a complete, previewable carousel in under a minute. The content type is what keeps the output from reading as generic: a "Tutorial" post is structured as sequential steps, a "Storytelling" post reads as a narrative arc with no numbering at all, an "Opinion" post states a stance and invites disagreement. Eight types, one engine.

Nothing publishes without a deliberate second action. Generating a carousel — the Claude call, the artwork, the upload — never touches Instagram. Publishing is a separate, explicit, confirmed step, whether that confirmation is a click in the web UI or a `dry_run` flag you turn off once you're happy with what you see.

5	8	10	1
PIPELINE STAGES	CONTENT TYPES	MAX SLIDES PER CAROUSEL	CONFIRM CLICK BEFORE ANYTHING GOES LIVE

WHO THIS IS FOR

- **Solo creators and small brands** who want a consistent, professional carousel format without hiring a designer for every post.
- **Anyone comfortable filling in a web form or running a GitHub Actions workflow** — day to day there's no coding involved, though the pipeline underneath is a Node.js repository you're free to extend.
- **People who want to own the pipeline** rather than rent it from a template subscription tool. Every piece of this runs on infrastructure you control.

PICK YOUR ANGLE

A topic on its own tells the pipeline *what* to write about. The content type tells it *how* — whether the slides read as a tutorial, a curated list, a stated opinion, or a story with no numbering at all. Section 05 covers all eight, and why a handful of angles were deliberately left out.

SECTION 02

The architecture

Five stages, one preview gate, publish only when you say so

1. **Input**
   You supply a topic, a page count (2–10), and a content type — via the web UI, the CLI, or a GitHub Actions manual dispatch. Nothing runs on a schedule by default; every carousel starts from a deliberate choice.

2. **Write**
   Claude receives the topic, the chosen content type's structure brief, and the brand voice in a single call, and returns structured JSON: a caption, a light-or-dark style assessment, an art-directed image brief, and every slide's kicker and headline.

3. **Generate**
   The image brief becomes one FLUX 1.1 Pro background photo (via fal.ai), shared across every slide. Slide copy is composited on top separately with `sharp` — bottom-anchored and scrim-protected so text never fights the artwork for space.

4. **Host**
   Every finished slide PNG is uploaded to Cloudinary, which gives Buffer the public URL it needs to fetch each image from.

5. **Preview & publish**
   The finished caption and every slide are shown before anything is sent anywhere. Publishing is a second, explicit, confirmed action — one Buffer GraphQL call posts the whole set as a native Instagram carousel.

DESIGN PRINCIPLE: NOTHING GOES LIVE BY ACCIDENT

Every stage up to and including hosting the finished images on Cloudinary is safe to run freely — generating a carousel never touches Buffer. The web UI separates `generateCarousel()` from `publishCarousel()` into two calls and gates the second behind a confirmation prompt; the CLI's `DRY_RUN` flag exercises the identical pipeline for testing without ever reaching Instagram.

WHY BUFFER, NOT THE INSTAGRAM GRAPH API DIRECTLY

Meta's own Graph API is technically free and technically direct — but getting there means a Meta Developer account, an app registration, and Business Suite's verification maze, which can eat the best part of a day even for an experienced developer. Buffer's API needs a personal API key generated in account settings, full stop. A few pounds a month at most — often free at this volume — buys you out of the worst part of this entire build.

SECTION 03

The stack, and why

Every tool here was chosen against a real alternative — including one we switched away from

LAYER	TOOL	WHY THIS, NOT THE ALTERNATIVE
Copywriting	Claude API	One call writes the caption, the style assessment, the image brief, and every slide together — splitting it into separate calls adds cost and latency with no quality gain here.
Content structure	8 hand-written type presets	A set of prompt fragments, not a second AI decision — keeps a "Tutorial" post reliably different in shape from a "Storytelling" post, rather than leaving it to chance.
Image generation	FLUX 1.1 Pro (fal.ai)	Started on OpenAI's `gpt-image-1`; switched after its output read as visually generic. FLUX responds far better to detailed art-direction — named lighting setups, lens characteristics, a specific illustration medium — rather than a vague description.
Text overlay	`sharp` + hand-built SVG	Image models still render typography inconsistently. Compositing the caption separately — bottom-anchored, behind a scrim — keeps it crisp and never competing with the artwork for the same space.
Hosting the media URL	Cloudinary	Buffer needs a public URL to fetch from, same as any publishing API. The free tier comfortably covers this volume.
Publishing	Buffer API	A personal API key from account settings — no Meta developer app, no business verification, no OAuth registration.
Interface	A small Express web UI	One HTML page, vanilla JS, no build step, no framework — matches the actual size of the job. A CLI does the identical work for scripted use.
Hosting the UI	Render (free tier)	Connects straight to the GitHub repo, environment variables in a plain dashboard — nothing to patch or manage.

WHAT WE DELIBERATELY LEFT OUT

The Instagram Graph API, direct — the setup cost is a Meta Developer account and Business Suite's verification maze. Buffer trades a few pounds a month, often nothing at this volume, for skipping all of it.

A second AI call to plan the visual style separately from the copy — one combined call keeps the artwork and the words conceptually aligned instead of generated in isolation from each other.

Content types that need real, user-supplied facts — case studies, testimonials, launch announcements, promotional offers. Generating those from a bare topic risks Claude inventing a customer quote, a result, or a discount that was never real. Left out until there's a way to feed in genuine specifics and a guardrail against fabrication, rather than built against today's understanding and reworked later.

SECTION 04

Foundations

Set these up once, before running the pipeline

ACCOUNTS YOU'LL NEED

- An **Anthropic** account for the Claude API (copy, style assessment, art direction).
- A **fal.ai** account for FLUX 1.1 Pro image generation, billed pay-per-image.
- A **Cloudinary** account (free tier) to host the rendered slide images publicly.
- A **Buffer** account with your Instagram professional account connected — done entirely through Buffer's own "Connect a channel" flow, the same login screen you'd use by hand.
- A **GitHub** repository, private is fine, to hold the pipeline and (optionally) run it on a schedule via Actions.
- A **Render** account (free tier), only if you want the web UI hosted somewhere other than your own machine.

ENVIRONMENT VARIABLES

VARIABLE	PURPOSE
`ANTHROPIC_API_KEY`	Writes the copy, style assessment, and image brief.
`FAL_KEY`	Generates the shared background photo. Optional — omit it and slides fall back to a flat branded gradient card.
`CLOUDINARY_CLOUD_NAME / _API_KEY / _API_SECRET`	Hosts every rendered slide PNG.
`BUFFER_API_KEY / BUFFER_CHANNEL_INSTAGRAM`	Publishes the finished carousel.
`APP_PASSWORD`	Required once the web UI is hosted anywhere reachable off your own machine — gates every route behind HTTP Basic Auth.
`BRAND_VOICE / ADAPTIVE_STYLE / BG_FROM / BG_TO / ACCENT`	Optional — tone of the copy, and whether the light-or-dark palette adapts per topic or stays fixed.

REPOSITORY STRUCTURE

```
claude-IG-prompt-post/
├─ .github/
│   └─ workflows/
│       └─ post.yml           # manual dispatch: topic, pages, content_type, dry_run
├─ lib/
│   └─ pipeline.js          # the pipeline itself — generateCarousel(), publishCarousel()
├─ carousel-post.js          # CLI entry point
├─ server.js                 # web UI — Express, one HTML page, no framework
├─ package.json
└─ .env.example
```

GETTING YOUR BUFFER API KEY AND CHANNEL ID

In Buffer, go to Settings → API and generate a personal API key — no app registration, no callback URL, no approval wait. Then, with the key in hand, query `{ account { organizations { id channels { id serviceType displayName } } } }` against `https://api.buffer.com` and take the `id` of the entry with `serviceType: "instagram"` — that's your `BUFFER_CHANNEL_INSTAGRAM`. A one-time lookup; it doesn't change unless the channel is disconnected and reconnected.

SECTION 05

Choosing your angle

Eight content types reshape what Claude writes — never how the slides are rendered

The content type is a set of prompt fragments composed into the same underlying Claude call, not a separate model or a second request. Picking one changes the structure Claude is told to follow and how the caption is angled — the rendering pipeline (artwork, text placement, palette) stays completely unaware of which type was chosen.

TYPE	WHAT IT PRODUCES
Let it decide	The original generic format — a cover hook plus N-1 numbered tips or prompts.
Prompt/Template	Each slide is a literal, copy-pasteable prompt in quotation marks.
Educational	Each slide explains one distinct fact or concept — teaching, not instructing.
Tutorial	Sequential numbered steps that build on each other, ending on the result.
Curated/Roundup	An independent list — "5 tools for X" — items in any order.
Opinion/Take	A stated stance plus supporting reasons; the caption invites debate.
Storytelling	A narrative arc — unnumbered story-beat labels, not a list at all.
Behind-the-scenes	A grounded walkthrough of a real process, in natural order.
Interactive	Ends on a direct question or poll prompt; the caption asks the reader to respond.

STYLE ADAPTS TO THE TOPIC, NOT JUST THE TYPE

Separately from content type, the same Claude call assesses whether the topic itself suits a **light, airy** look (wellness, food, morning routines) or a **dark, moody** one (security, warnings, urgency) — and picks an accent colour to match. That assessment drives the background image prompt's lighting direction, the text-scrim colour, and the text colour, so a "5 morning habits" carousel and a "warning signs your app has a security hole" carousel come out visually distinct even on the same content type. Set `ADAPTIVE_STYLE=false` to lock in one fixed brand palette instead, across every topic.

SECTION 06

Writing the copy

One Claude call, structured JSON, everything downstream reads from it

The topic, page count, and the chosen content type's structure brief go into a single request. The system prompt requires strict JSON back — no preamble, no markdown fences — covering the caption, the style assessment, the image brief, and every slide's copy in one response, so the visual direction and the words are never generated in isolation from each other.

```
{
  "caption": "...",
  "style": { "mood": "light | dark", "accent": "#hex" },
  "imagePrompt": "art-directed brief for FLUX — see Section 07",
  "slides": [
    { "kicker": "cover eyebrow", "headline": "the hook" },
    { "kicker": "1/ SHORT TITLE:", "headline": "slide content, per the chosen type's structure" }
    // ... one entry per remaining slide
  ]
}
```

The `kicker` field's own instruction changes with the content type's `kickerStyle`: numbered ("1/ SHORT TITLE:") for list-shaped types, or a short narrative label with no numbering at all ("THE SETUP", "THE TURNING POINT") for Storytelling and Behind-the-scenes.

NO AUTOMATIC RETRY ON MALFORMED JSON

If Claude's response doesn't parse as valid JSON, or the slide count doesn't match what was asked for, the pipeline fails fast with the raw response logged, rather than silently retrying. In practice this is rare given the strict system prompt — worth knowing as a known limitation rather than a claimed guarantee, and a reasonable first extension if it starts happening often (see Section 10).

SECTION 07

Generating the artwork

FLUX for the base image, a template pass for text that never fights it

A · GENERATE THE BASE IMAGE WITH FLUX

Send the `imagePrompt` to FLUX 1.1 Pro via fal.ai's synchronous endpoint — fast enough (single-digit seconds) that a blocking request is simpler than queue-and-poll here.

```
// lib/pipeline.js
fetch('https://fal.run/fal-ai/flux-pro/v1.1', {
  method: 'POST',
  headers: {
    Authorization: `Key ${FAL_KEY}`,  // fal.ai's own scheme — not Bearer
    'content-type': 'application/json',
  },
  body: JSON.stringify({
    prompt: imagePrompt,
    image_size: { width: 1088, height: 1344 },  // closest 4:5-ish, multiples of 32
    num_images: 1,
  }),
});
```

B · WRITE THE PROMPT LIKE AN ART DIRECTOR, NOT A CAPTION

The `imagePrompt` instruction Claude writes to is deliberately specific: a named lighting setup and time of day, a named photographic or illustration style (camera/lens characteristics, or a specific illustration medium), and an explicit steer away from overused AI-image clichés — a glowing laptop, a generic handshake, floating particles, a lightbulb. Text is never asked of the image model at all; it's composited afterward.

C · COMPOSITE THE TEXT, BOTTOM-ANCHORED

Every slide's kicker and headline render as SVG, then `sharp` composites that over the FLUX photo behind a dark-or-light scrim (matching the style assessment's mood). Text always anchors to the **bottom** of the frame — the same zone the image prompt is told to keep visually simple — so it never lands on top of whatever the model placed in the upper frame.

WHY THIS CHANGED

Two real fixes, in order. First: the original background generator was OpenAI's `gpt-image-1` with a fairly generic prompt — the output read as visually boring. Swapping to FLUX with an art-director-style prompt fixed that. Second, separately: non-cover slides originally pinned their kicker to a fixed position near the *top* of the frame, while the cover slide bottom-anchored — directly contradicting the image prompt's own promise to keep the lower half clear. Every slide but the cover was landing text on top of the artwork's main subject. Bottom-anchoring every slide consistently fixed it.

SECTION 08

Publishing via Buffer

One GraphQL call, corrected against Buffer's actual schema

Buffer's API is GraphQL, hit at a single endpoint (`https://api.buffer.com`) with the personal API key as a Bearer token. A carousel is just multiple entries in the `assets` array — there's no separate "carousel" type flag; more than one image is what makes it a carousel rather than a single-image post.

```
mutation CreatePost($input: CreatePostInput!) {
  createPost(input: $input) {
    ... on PostActionSuccess { post { id dueAt } }
    ... on MutationError { message }
  }
}

// input:
{
  text: caption,
  channelId: BUFFER_CHANNEL_INSTAGRAM,
  schedulingType: automatic,
  mode: shareNow,
  assets: slideAssets.map(a => ({ image: { url: a.url, metadata: { altText: a.altText } } })),
  metadata: { instagram: { type: post, shouldShareToFeed: true } },
}
```

CORRECTIONS AGAINST BUFFER'S LIVE SCHEMA

Type: `channelId` is the custom `ChannelId` scalar, not `String` — a mismatch Buffer's validator rejects outright.
Required Instagram metadata: every Instagram post needs both `metadata.instagram.type` and `metadata.instagram.shouldShareToFeed` — omit either and the mutation fails validation before it reaches Instagram at all.
Mode: `mode: addToQueue` adds the post to Buffer's own queue for its next scheduled slot — it does not publish. Use `shareNow` if "approved" is meant to mean "goes out now."

CAROUSEL-SPECIFIC LIMITS

Instagram's own app allows up to 20 carousel images, but the Graph API — which Buffer publishes through — caps third-party carousel posts at **10**. Every slide renders at 1080×1350 (4:5 portrait), Instagram's recommended carousel size; mismatched aspect ratios aren't possible here since every slide runs through the same template.

RATE LIMITS

Buffer's free plan allows 100 API requests per 24 hours and 3,000 per 30 days — not remotely a concern at the volume this pipeline is designed for. Paid plans raise this ceiling if that changes.

SECTION 09

Preview, then publish

Three ways to run it — a form, a terminal, or a manual workflow trigger

THE WEB UI

`npm run server`, then open the page: enter a topic, a page count, and a content type, click **Generate carousel**. The caption and every rendered slide appear inline — nothing has touched Buffer yet. Only **Publish to Instagram now**, behind a confirmation prompt, actually goes live. Hosted anywhere off your own machine, every route requires an `APP_PASSWORD` via HTTP Basic Auth — there's no login otherwise, and a public URL with none would let a stranger burn your API usage or publish to your account.

THE CLI

`node carousel-post.js "topic" 6 tutorial` — or run it with no arguments for interactive prompts. Publishes immediately unless `DRY_RUN=true` is set, in which case the identical pipeline runs through Cloudinary upload and stops, logging what it would have posted to `output/carousel-posts-dry-run.json`.

GITHUB ACTIONS (MANUAL DISPATCH)

The workflow accepts `topic`, `pages`, a `content_type` dropdown, and a `dry_run` checkbox, triggered from the Actions tab instead of a local shell — useful for testing against real secrets without installing anything locally.

```yaml
# .github/workflows/post.yml
on:
  workflow_dispatch:
    inputs:
      topic: { required: true, type: string }
      pages: { required: false, type: string, default: '6' }
      content_type: { type: choice, default: auto, options: [...] }
      dry_run: { type: boolean, default: false }
permissions:
  contents: write   # required — without it, committing the run log 403s
```

RUNNING IT LOCALLY VS. ON RENDER

WHERE	GOOD FOR	WATCH FOR
Local (`npm run server`)	Day-to-day use, fastest iteration, no `APP_PASSWORD` needed since only your machine can reach it.	Only runs while your laptop is on and the terminal's open.
Render (hosted)	Always-on, reachable from any browser, no local setup once deployed.	Free tier sleeps after inactivity — first request after a while takes 30-60s to wake.

ENV VARS LIVE IN THREE SEPARATE PLACES

Your local `.env`, GitHub Actions' repository secrets, and Render's Environment tab are three independent credential stores — none of them sync with each other. Regenerate a key (a fal.ai key that 401'd, say) and it only takes effect wherever you actually go and update it. The exact failure this causes: the pipeline runs fine end to end but silently falls back to the flat gradient card, since a failed background-image call is non-fatal by design (Section 07) — nothing looks broken, it just looks plainer than expected. If a run "succeeds" but the artwork is missing, check the environment you're actually running in has the *current* value, not just one of the other two.

SECTION 10

Running costs & troubleshooting

What it actually costs, and the failure modes to expect

ITEM	TYPICAL COST
Claude API (one call per carousel, small prompt)	A fraction of a penny per generation
FLUX via fal.ai (one image per carousel)	A few cents, billed pay-per-image — check fal.ai's current pricing for flux-pro/v1.1
Cloudinary (image hosting)	Free tier comfortably covers this volume
Buffer (one channel, free plan)	Free — upgrade only for more channels or higher-frequency posting
Render (web UI hosting, free tier)	Free — sleeps after inactivity, ~30-60s to wake
GitHub Actions (manual runs only)	Free — well within the free-tier minutes

COMMON FAILURE POINTS

- ✓ **Invalid Cloudinary `cloud_name`.** A stray quote, space, or the wrong field copied in — Cloudinary rejects it outright with a 401. Re-copy the exact value from the Dashboard homepage.
- ✓ **fal.ai 401.** An incomplete or regenerated `FAL_KEY` — re-copy the full key from fal.ai's dashboard. Non-fatal by design: the pipeline falls back to the flat gradient card rather than failing the whole run.
- ✓ **Missing `permissions: contents: write` on the workflow.** Without it, the run's own log-commit step 403s even when the actual pipeline succeeded — easy to misread as a pipeline failure when it's a housekeeping step.
- ✓ **`channelId` typed as `String`.** Buffer's validator rejects it with `GRAPHQL_VALIDATION_FAILED` — it must be the `ChannelId` scalar.
- ✓ **Missing Instagram metadata.** `metadata.instagram.type` and `.shouldShareToFeed` are both required on every Instagram post; the mutation fails validation without them.
- ✓ **Post accepted but nothing appears on Instagram.** Almost always `mode: addToQueue` — it queues for Buffer's own schedule rather than publishing. Use `shareNow`.
- ✓ **More than 10 slides requested.** Rejected before generation starts — Buffer's third-party carousel cap, regardless of what Instagram's own app allows.

WHERE TO TAKE THIS NEXT

Automatic retry with backoff for transient API errors (a 529 from Claude, a brief network blip) rather than failing the run outright; the case-study, testimonial, and promotional content types, once there's a real-data input field and a fabrication guardrail to go with them; and eventually Reels — which trades the FLUX step for a short AI-video generation call instead.

WANT THIS BUILT FOR YOU?

AlumnAI builds and hands over working pipelines like this one — configured for your topic, your brand, and your account, ready to switch on. Get in touch via alumnai.co.uk.