APP COMPOSER

App Composer and the Loftwah project portfolio

The completed product vision

The story, the applications, the creative studio and the engineering behind an owned project that can keep improving.

Prepared for creators, technical collaborators, potential customers and partners

Dean Lofts

This edition describes the intended experience once the accepted backlogs are delivered to a high standard. It is an account of the completed vision, rather than a statement that every capability is available today. Research outcomes, project holds and retired products keep their stated boundaries.

Includes a one-sentence explanation, a short pitch, technical and nontechnical summaries, a detailed project catalogue, architecture, use cases and linked primary sources.

2026/11/20 Australia/Melbourne AEDT (UTC+11)
2026/11/20 UTC

Contents

Choose the short versions for a quick explanation, or follow the complete account. Chapter titles link to the corresponding section.

1 The short versions....3

2 The story behind App Composer....5

3 The mission and the people it serves....7

4 The experience from idea to ongoing operation....9

5 Media Studio and finished creative work....12

6 Share and a library that improves through use....15

7 The practical applications in the Composer family....17

8 The wider portfolio and what each project proves....21

9 WordPress migration and the EmDash opportunity....26

10 How the system works underneath....28

11 Agents ownership and dependable operation....31

12 Quality benchmarks and proof....33

13 Economics and the value of choosing well....35

14 The selling points and practical use cases....37

15 What success looks like....39

16 Plain language glossary....41

17 Sources and scope....42

1 The short versions

TLDR in one sentence

App Composer helps people and their chosen AI agents turn an idea into a useful project they own, give it a polished public presence, and keep improving it with sound engineering underneath.

TLDR in thirty seconds

Imagine describing an application, website, game or creative project, showing a few examples of what you like, and receiving something you can actually use. App Composer connects the work needed to get there: planning, software, design, media, deployment, publishing and ongoing operation. You can stay in a guided experience or inspect the source and infrastructure whenever you want. The result belongs to you, including the material and project history needed to change it later. Cloudflare provides the first production foundation; your agent and the connected applications do the appropriate work. [P01] [P02]

TLDR for a technical reader

App Composer is an open-source creation and operations platform built around independently owned applications and explicit service boundaries. Its Astro and TypeScript application uses EmDash for native administration and content workflows. Cloudflare Workers is the primary deployment target, with storage, identity, AI and coordination services selected for the workload. GitHub retains source and change history. Supported external agents implement authorised work through scoped integrations, while Composer retains the brief, project relationships, reusable capability contracts and a coherent view of results. Media Studio owns creative production and its asset lifecycle; connected apps own their own records, permissions and releases. [P01] [P03]

The engineering proposition is continuity across the full lifecycle. A generated preview becomes an owned deployment; a requested change preserves existing content and customisation; an asset retains its source and rights through a derivative; an interrupted job can be reconciled; a release can be tied to the exact source that produced it. The system exposes useful operations to humans and agents without making Composer a compulsory runtime dependency for every app.

TLDR for someone who does not work in technology

You should be able to make a useful thing without first learning how to run an internet company. App Composer gives you a place to explain the idea, make important choices, see the result and change your mind. It helps with the work that usually surrounds the idea: setting up the application, making it look good, preparing images and videos, putting it online, sharing it and looking after it.

You might use it to launch a personal website, a small business tool, an interactive game, a music project or an automated research service. You can use an existing example as a starting point, combine useful pieces, or ask an agent to create a missing feature. When you return, your project is still there with its previous decisions and editable work. You do not have to begin the explanation again.

The executive summary

The central customer problem is the amount of disconnected work between having an idea and owning a good result. Software creation is only part of that work. People also need design, content, media, accounts, deployment, updates, distribution, support and some understanding of cost. Each handoff can lose context or leave a task unfinished.

App Composer carries one project through those handoffs. Its value comes from making the next useful action clear and connecting it to the right application, agent or service. A beginner sees an understandable workflow. An engineer sees the underlying contracts and evidence. Both work on the same project.

A customer's application does not require the founder's private accounts or the Woodhouse installation to remain useful. It can continue under its owner's control, with its own source, assets and service connections. [P03] [P06]

The surrounding portfolio gives the platform practical substance. Paperwork handles real PDFs. LDraw provides an editable diagramming workspace. Linkarooie and Here / there demonstrate connected creator applications. Media Studio produces artwork, audio, films and reusable assets. LoftwahFM supplies an independent music platform. Games such as Fighter, SHOALSHOT, Bubbles and Protocol 11 exercise interaction, performance, persistence and presentation in ways a conventional business form cannot. These products remain useful in their own right while contributing tested methods and components to the wider system. [P01] [P20] [P21]

The mission has two related parts. For the customer, it is easier creation and continued ownership of useful work. For the Cloudflare ecosystem, it is more people building on Workers and using Workers AI for tasks that are worth doing. Useful repeat use connects those missions. Increasing token consumption without improving an outcome would miss the customer purpose. [P02] [P22]

The defining acceptance test is demanding: a person gives the agent an idea, the system delivers a useful mini-app and its launch material, authorised publication actually happens, the project continues operating, and the owner is proud to share the result without repairing it themselves. Technical checks, creative judgement and observed operation all contribute to that result. [P23]

How to read the full document

For the story and purpose, read Chapters 2 and 3. For the product experience, read Chapters 4 to 6. For the project catalogue, read Chapters 7 and 8. For WordPress migration, read Chapter 9. Engineers will find the architectural detail in Chapters 10 and 11. Chapters 12 to 15 explain quality, economics, commercial value and what success means. The glossary explains unfamiliar terms, and the source register identifies the product contracts, repositories and backlog themes behind the account.

2 The story behind App Composer

The recurring problem

I kept finding things I wanted to make, then working out which existing tools would get me there. Sometimes the useful skill was writing code. Often it was remembering that somebody had already solved a difficult part of the problem, understanding whether their solution fitted, and joining the pieces carefully.

That habit connects my work in music, websites, infrastructure and AI. These experiences overlapped. App Composer grows out of the accumulated method rather than a neat sequence of unrelated careers. [P04]

Music created practical reasons to build

Music production meant there were releases to present, listeners to reach, merchandise to sell and websites to maintain. My work around Shake Down Records and Grind Mode Cypher brought WordPress, WooCommerce and the practical demands of online publishing into the same world as making music.

A track on a hard drive was only one part of a release. The work around it could include artwork, a page, a store, social posts and a way for people to find or play it. Those needs were connected from the beginning. App Composer's combination of software and creative production reflects that experience. [P04]

Reuse was a deliberate engineering choice

My WordPress approach had three practical levels. Reuse a good plugin or component when it solved the problem well. Pay for a specialist capability when its value justified the cost. Build the missing piece when the available options did not fit.

A lifetime licence could make sense for something I expected to use repeatedly. An occasional paid tool could make more sense for a narrow specialist task. Custom code was valuable where it added something the existing components could not provide. The judgement was in knowing which situation I was looking at.

The same approach applied to GitHub projects. Finding a capable project, checking its assumptions and adapting it responsibly could be much more useful than rebuilding its functionality. That work still required technical understanding. A component's existence did not establish that it was maintained, compatible, secure or appropriate for the intended user. [P04]

App Composer makes that method available as a product. Its reusable building blocks can be small interface components, complete applications, content models, workflows, agent skills or creative assets. A creator can discover and combine useful pieces, while an agent implements what is missing. The platform records the ownership and compatibility information needed to do this responsibly. [P01] [P03]

Systems work added responsibility for what happens next

My background includes communications work in the Royal Australian Navy, followed by work with systems, networks and infrastructure. DevOps work brought deployment, permissions, automation and recovery into the centre of the problem. Applications needed to keep working after the initial build. Changes needed a repeatable path into production. Failures needed an understandable response. [P04]

That experience explains why App Composer includes operations in its basic promise. A creator does not gain much from an impressive demonstration that cannot be maintained. A useful system needs a clear owner, a known deployment, access to its records, a way to understand costs and a route back from a bad change.

The ServerPilot analogy captures a particular part of this idea. Tools that made hosting and HTTPS manageable removed work people had to do before they could concentrate on their websites. App Composer applies that instinct to AI agents, creation workflows and Cloudflare. The goal is to let the person reach the interesting part quickly, while retaining a sensible foundation underneath. [P02]

AI expanded what could be attempted

AI agents made it possible to initiate much more ambitious work. I could describe a result, provide references, ask for an investigation and let agents attempt the implementation. I could then inspect what they made, identify what was wrong and direct the next revision.

The games became an especially useful environment for exercising this approach. A game makes defects difficult to hide. Controls either respond well or they do not. A screen is readable on a phone or it is not. A progression loop is satisfying or it falls flat. The player notices missing sound, confusing feedback, awkward menus and broken saved progress immediately.

That experience shaped a higher standard for the whole portfolio. A working code path matters, but so do the visual hierarchy, the pace of interaction, the quality of the media and the user's ability to understand what happened. App Composer carries that standard into business tools and creative applications as well as games. [P01] [P21]

The product is the method made available to other people

App Composer brings together the habits that proved useful across those experiences. Discover capable tools. Preserve the user's intent. Reuse what fits. Build the missing part. Inspect the actual outcome. Keep the work understandable enough to improve later.

The ambition is broader than giving a person an empty editor and asking them to become a developer. A creator may arrive with a rough idea, a screenshot, an existing website, a piece of music or simply something they wish existed. App Composer helps turn that starting point into a concrete project and connects the work needed to complete it.

This is also why personal taste matters. Two people can request the same kind of application and want very different results. Their references, language, preferred level of detail and rejected examples influence what good means. The platform preserves those choices alongside the technical specification, so later revisions can remain recognisably theirs. [P01] [P05]

3 The mission and the people it serves

The customer mission

Our customer mission is to make it simple for people to create, own and improve useful projects with sound engineering underneath. A person should be able to express an intention, see a working result and continue shaping it without losing their source, data, creative decisions or control. [P01] [P02]

The word useful carries weight. A project needs to do a job for somebody. It might save an administrative task, make a game enjoyable, help a musician present a release or give a small business a better way to publish. The technology serves that purpose.

Ownership also has a practical meaning. The owner can identify where the application runs, obtain its source and relevant data, retain its editable assets, understand which accounts it uses and continue operating it independently. A downloadable picture of an application would not meet that standard. [P03] [P06]

The Cloudflare mission

App Composer's infrastructure mission is to encourage useful adoption of Cloudflare Workers and recurring use of Workers AI. It makes those services approachable by connecting them to outcomes people already want. Someone may arrive to make a club website or a research tool and leave with a working project in their own account. [P02] [P22]

For Cloudflare, this creates a route from technical capability to sustained use. For the customer, the attraction is a capable application with manageable operation. App Composer earns its place by connecting the two. It remains an independently owned product built on the platform; this positioning does not imply a funding relationship or endorsement.

Creators and small organisations

Creators want to publish, sell, teach, entertain or organise. Their obstacles often include unfamiliar technical setup and the number of tools needed around the main idea. App Composer provides guided starting points, examples, useful defaults and clear next actions.

A musician can build a release page, prepare artwork and clips, connect a listening experience and schedule supported promotion. A local organisation can create a small information site, collect appropriate submissions and moderate them. An independent maker can turn a calculation or repeated workflow into a small application and share it with others.

The common benefit is less coordination work. The person can concentrate on whether the result is useful and appropriate, while the system carries the relevant context between tasks.

Developers and technically confident owners

Developers gain a structured way to connect existing code, agents, reusable capabilities and deployment services. They can inspect manifests, source pins, permissions, changes and

evidence. They can use Composer as a front door without surrendering the normal tools that make technical work manageable. [P03]

The platform also accommodates the person who enjoys building but does not want to repeat setup work for every project. They can begin from a reviewed recipe, replace a component, add a custom capability and keep a normal source history. The useful abstraction reduces routine work while leaving the implementation accessible.

Agencies and platform teams

An agency can reuse an effective setup across client projects while preserving each client's identity, accounts and data. It can manage previews, approvals, content workflows and handover through consistent processes. A platform team can offer approved application patterns with clear ownership, environments, quotas and audit records. [P24]

These users need more than an attractive generator. They need to know who can deploy, which changes reached a particular environment, what a project costs and how to recover it. Composer's enterprise capabilities extend those existing contracts to teams and controlled self-service.

The system still distinguishes the service being offered. An application in a customer's own Cloudflare account has different responsibilities from a centrally managed hosting service. Identity federation, compliance requirements and provider-specific deployment options are selected and qualified for the actual organisation. [P02] [P24]

Agents are users of the platform too

An agent needs clear interfaces, relevant project context and operations it can safely call. App Composer exposes structured capabilities alongside the human interface. It distinguishes permission to inspect from permission to edit, execute, deploy or publish. [P03] [P25]

This lets the person choose a comfortable level of involvement. They can make direct edits, ask an agent for a bounded change or authorise a continuing workflow. Routine work can proceed within existing grants. Meaningful decisions remain visible, including cost commitments and changes to public exposure.

4 The experience from idea to ongoing operation

Start with an outcome

The opening question is what the person wants to accomplish. They can describe an idea, choose a useful example, bring an existing project or browse until something gives them a starting point. They do not need to name an infrastructure product before seeing what they could make.

A meaningful brief captures the audience, the task, the desired experience and the constraints. References help make taste concrete. The person might prefer the density of one interface, the colour treatment of another and the simplicity of a third. Composer records what each reference contributes, including which directions were rejected. [P01] [P08]

The product can show an early prototype or visualisation when it will make a consequential choice easier. The owner can touch the controls, inspect a screen or play with the behaviour. This turns an abstract discussion into something they can judge.

Get Ready removes setup uncertainty

Get Ready establishes whether the selected working environment can actually perform the intended task. It helps the person obtain or connect a supported agent, check the relevant accounts and tools, and understand what remains to be done. Its state can be saved and resumed. [P07]

For a beginner, that means actionable guidance rather than a page of unfamiliar requirements. For an engineer, it means observed readiness rather than assumptions inferred from a browser. A checked connection has a particular identity, capability and scope.

The account journey explains the practical difference between a chat subscription, API access, cloud resources and a custom domain. It avoids making a paid model service compulsory when the person's supported agent and accounts can already do the job. Some browser-local tools are useful immediately and do not need cloud setup at all. [P01] [P25]

Init turns the brief into a usable specification

Init captures the structure of the proposed application. A creator might define a directory of local services, a set of articles, a submission form or a catalogue of collectible characters. The fields, relationships, validation and relevant workflows become a portable specification. [P08]

This draws on an approachable lesson from tools such as ACF and Meta Box: people can understand that a record has a title, image, category and published state without having to begin with a database schema. The technical representation underneath still needs to be explicit enough for an agent to implement and validate.

The resulting specification is project context. Execution authority comes from the user's actual permissions and selected environment. Keeping those separate means a file received from someone else cannot silently grant itself access to an account.

The agent builds and the person can inspect

Composer selects relevant reusable blocks and gives the chosen agent the task, references and acceptance conditions. The agent can implement a missing feature when the catalogue does not already contain an appropriate solution. The result returns as real source and a working preview tied to a particular revision. [P03] [P05]

The person can point to an element in that preview and request a change. A useful request might be, "This card is too large on my phone," or, "Keep this layout, but let visitors filter by location." The system retains the selected element or region, the viewport and the version being reviewed. The next preview can therefore be compared with the right starting point. [P26]

For engineers, this connection reduces ambiguity around stale feedback. For everyone else, it makes the application feel editable in an ordinary way. A request refers to the thing they are looking at rather than to a new interpretation reconstructed from a chat transcript.

Make the result presentable

The software, copy and media develop as one project. Brand Studio preserves the approved identity. Media Studio creates and edits the required artwork, audio or film. Launch Kit packages appropriate icons and sharing images. The application uses genuine product captures where it needs to demonstrate behaviour. [P27] [P28]

Presentation is part of the outcome. Text should be readable. Controls should be understandable. The mobile view should be designed deliberately. The public explanation should describe what the actual application does. A generated hero image can support a product, but it cannot stand in for evidence that the product works.

Publish to an owned destination

The supported deployment path connects reviewed source to the selected account and environment. Preview and production have distinct identities. Failed candidates preserve the last known working release, and the owner can tell which source version is live. [P09]

Cloudflare is the primary foundation for the maintained recipes. A custom domain can be connected when it is useful; a provider hostname can support the first working result. The project retains its code, configuration and relevant recovery material.

Publishing creative work uses a separate permission. A private Studio project can produce a deliberately public derivative without exposing the original references, prompts or unpublished variants. Application deployment and public media publication are related steps with different effects. [P03] [P28]

Launch through supported channels

The approved application becomes the destination for its launch material. The system prepares suitable copy, images, clips and links, then uses the enabled publication routes within the owner's authority. It records what was drafted, exported, scheduled and actually published. [P29]

The maintained social direction starts with Bluesky and eligible primary Instagram and Facebook Page routes. X receives export packages under the current product policy. Channel limitations remain visible. A file ready to post has a useful place in the workflow, while a real publication requires a destination receipt and a read-back showing the intended content at the intended destination. Scheduled material is checked after its due time. [P23] [P29]

Return to the same project

After launch, the workspace explains what is live, what needs attention and what happened recently. The person can request another feature, update content, revise a film or inspect cost and usage. Their project identity, accepted design and existing records remain connected. [P03] [P30]

An application can evolve without casually overwriting the owner's customisations. A creative revision can preserve the accepted shots while changing the weak one. An interrupted operation can retain enough state to determine what completed before another attempt is made.

That return visit is an essential part of the promise. A project is valuable because it can continue to serve its owner, adapt to new needs and survive ordinary mistakes. The first impressive generation is the beginning of that relationship.

5 Media Studio and finished creative work

A production application connected to the project

Media Studio is the creative production application in the App Composer family. It turns ideas and source material into finished artwork, sound, films, motion graphics and other useful assets. A creator can direct the work conversationally, make visual edits, inspect the result and return to the same saved production later. [P28]

Consider a game trailer. The creator wants a particular character, footage of real play, a suitable soundtrack and captions that remain readable on a phone. The Studio can work with those actual materials and their saved identities. A request to change the final shot can preserve the approved opening, music and overall timing. The new export remains connected to the source project and earlier version.

Studio owns the creative project, media bytes, production jobs and releases. Composer connects those outputs to the application, brand, library and launch workflow. This division allows an independently installed Studio to keep working without depending on the founder's running Composer or Woodhouse installation. [P03] [P28]

The reference desk makes inputs usable

Creative work often begins with a mixture of URLs, screenshots, PDFs, voice notes, existing images and half-formed instructions. The Reference Desk gives that material a deliberate place. It retains the original, records where it came from and shows what was extracted or inferred. [P31]

A reference might establish a visual treatment, a camera relationship, a production constraint or a fact that the script needs to explain. These are different contributions. The Studio can attach an observation to a source page, frame or timecode so the creator can inspect and correct it.

Reference analysis also makes learning more specific. Instead of describing a film as cinematic, the project can identify what a particular shot does: where the subject is, how the camera moves, which sound marks a transition and how a later image changes the meaning. Those observations become inputs to original work. The source's existence does not grant permission to reuse its media.

Conversation and direct editing share the same work

The Creative Director is a conversational way to operate the Studio. A person can ask for a different crop, shorter scene, new title, clearer narration or a particular character. The visual workbench remains available for direct control. Both routes work against the same saved project and operations. [P32]

This matters when an instruction is precise. "Move the title up slightly" should modify the existing title on the version being viewed. It should not start a new film with different artwork. The technical model uses revision identities and bounded changes so outdated instructions can be recognised before they overwrite newer decisions.

Direct editing also gives the owner an escape from conversational friction. Some adjustments are easier to make by dragging an element, correcting a word or choosing a take. The application supports that action without making the person abandon the project context that their agent uses.

Image tools produce practical assets

Image work includes both creative generation and ordinary preparation. The Studio can generate or refine an image through a qualified model, then use deterministic tools to crop it, remove unwanted metadata, extract a palette, trim transparent space, make a contact sheet or prepare a sprite atlas. [P33]

Browser-local operations can process selected files without uploading them. Provider generation is a different action with its own data exposure and cost. The interface makes that distinction understandable. Saving a local result into a Studio project is an explicit step, preserving the relevant source and output identities.

The useful output is a file that fits its purpose. A game atlas needs correct rectangles and dimensions. A social image needs an appropriate crop and readable text. A transparent asset needs useful edges. The workflow checks the actual exported result, including the case where a technically valid image is aesthetically unsuitable.

Characters retain an identity across productions

A reusable character needs more than a short prompt. Studio retains an identity, approved references, lookbook versions, expressions or poses, and the relationships between those materials. A production can specify which character version and reference roles it is using. [P34]

This supports a character appearing in a game, a piece of cover art and a short film without treating every appearance as an unrelated invention. It also makes intentional change possible. A revised outfit or visual treatment can become a new version while older productions retain their original identity.

Public character pages provide an optional home for approved appearances and background material. Private reference art and unpublished takes keep their own visibility. A public profile does not open the entire working collection.

Films keep their editing structure

A finished film is an arrangement of decisions: script, scenes, shots, sound, timing, captions and selected takes. Studio keeps those decisions editable. It can combine generated footage with owned recordings, motion graphics and product captures, then retain the relationship between the project and its render. [P35]

For a music video, the song is a central timing source. Scenes and takes refer to actual portions of the track. The creator can compare alternatives and replace a weak shot without casually moving the musical structure. Earlier renders remain available for comparison.

For a product explanation, narration and claims must match the real application. A polished film that implies a feature works differently from the product would fail the purpose. The Studio's content-planning and review processes connect the script to the project evidence and the intended audience. [P36]

The plan also respects what the creator can practically film. Someone working alone with a phone can receive a suitable script and shot plan. Screen recordings, talking-head clips, hands-only demonstrations and faceless productions each have appropriate workflows. Approved audience, voice and brand context carry into the next piece without another long briefing.

Review considers the craft relevant to the film: pacing, visual hierarchy, character and scene continuity, caption readability, intelligible narration, soundtrack balance and a satisfying ending. These qualities determine whether a technically valid export works as a finished piece. [P35] [P36] [P39]

Accurate word timing enables reliable speech editing

A transcript records what was said. Word timing records where each word occurs. That difference is essential when an agent removes a phrase, rearranges excerpts or synchronises captions. [P37]

The editing model distinguishes speech recognition, alignment, speaker attribution and sung lyrics. A corrected transcript can require new alignment. A sequence made from disconnected excerpts needs a mapping between source time and final-film time; applying one caption offset to the whole export would be wrong.

The practical benefit is that an instruction such as "Remove this sentence and keep the captions aligned" can refer to actual edit boundaries. A change to visuals can preserve captions tied to continuous music. Ordinary caption corrections and trimming can reuse existing footage without initiating unnecessary paid generation.

Motion and three dimensional work use qualified methods

Motion Lab provides editable scenes, layers, text and timing for appropriate motion-graphics work. The same broader production system can accommodate original SVG identity, seeded pixel effects, interactive Three.js explainers, mathematical animation, Blender authoring, generated three-dimensional asset bundles and compatible skeletal motion. [P38]

Each capability has a specific contract. A Blender prop retains its editable source as well as an independently inspected portable export. An animated character requires a compatible rig and verified motion transfer. A generated mesh needs its textures and dependent files. An interactive explanation and its film should represent the same underlying model.

These requirements keep the catalogue useful. Completing a qualification exercise can result in adoption, a narrower supported capability or rejection of an unsuitable method. The system presents the options that passed their relevant tests; it does not imply that every model supports every rig, file format or production scale.

Review is attached to the exact thing being judged

A reviewer can comment on an image region, a film timecode or a passage of script. Private review rooms identify the selected saved version. Approval of one render remains attached to that render. A revised film receives its own review state. [P39]

The creator can compare versions and see which requested changes were made. Selective correction preserves accepted work: fix the text, scene, voice line or crop that needs attention, then inspect the new result. This is useful creatively and economically because good work does not need to be discarded to repair a small defect.

The production desk makes the next step clear. A project might need a selected take, a missing voice line, a response to a review note or a successful export. The desk points to the existing authoritative jobs and records, giving the person an understandable view of the production.

6 Share and a library that improves through use

Work remains useful after export

App Composer Share is the portfolio's creative discovery, collection and reuse experience. It connects finished work, meaningful experiments, project kits and the explanation of how an output was made. Media Studio and other originating applications retain authority over their underlying assets. [P40]

A visitor can explore public work, watch or listen to it, inspect a permitted source story and find an appropriate reuse path. An owner has a richer private view containing originals, unreviewed outputs, rejected variants and work that is not eligible for publication.

One creative work can contain several versions. The gallery gives it a useful primary presentation while retaining access to alternatives. This avoids making every iteration look like a separate project or deleting a result simply because a later version was preferred.

Reuse carries context and rights

A useful asset record identifies its creator or source, exact version, licence or permission, relevant references, visibility and intended use. A derivative retains the relationship to its source. The new work can have its own review and publication decision without rewriting the original. [P40] [P41]

For the user, that means the reuse button leads to a concrete next step. Open the asset in Studio, place it in a drawing, adopt it into a project or download the selected package. The output should be the actual version the person chose, with the material required to use it.

Publication, listing and withdrawal have distinct effects. Withdrawing controlled public access stops future authorised delivery through those surfaces, while the owner can retain private source work and history. A copy that somebody previously downloaded has already left that controlled environment.

Brand Studio and Launch Kit maintain coherence

Brand Studio provides a place to develop and apply a project's visual identity. The owner can compare treatments in real application screens, retain drafts and publish or restore a selected version. A brand decision is useful when it governs the actual interface, typography, imagery and tone across the project. [P27]

Launch Kit turns selected identity material into concrete files such as icons, social images, profile cards and installation guidance. It preserves the crop and settings used to produce them. The generated package can be reviewed and adopted deliberately into the target application. [P42]

Together, these capabilities reduce a common source of inconsistency. The public page, app icon, social card and trailer can all refer to the same approved identity, while each format receives an appropriate treatment.

Experiments become a usable body of work

The creative experiment loop is designed to try a broad range of useful methods and make the results easy to judge. Each substantive attempt records the brief, relevant references, skill

revision, model or tool, output version and available time and cost information. Failed attempts retain an intelligible record too. [P43]

The owner can use thumbs up or thumbs down without writing an essay. They can also give a precise reason, compare two alternatives or point to the part that feels wrong. That feedback remains attached to the version that prompted it.

A response such as "I like the colours, but the movement is wrong" should preserve both observations. It should not become a global dislike of the colour scheme. Explicit choices, personal preference and machine-inferred suggestions remain distinguishable and scoped to the relevant project.

Learning changes the next attempt in an inspectable way

The system can retrieve relevant accepted examples, apply a corrected recipe and show how the new attempt differs. A weak result can lead to a revised prompt, a better reference, a different model, a deterministic editing step or retirement of the method. [P43]

This is operational learning. It can improve instructions, retrieval, selection and evaluation without claiming that every preference retrains a model's underlying weights. The owner can inspect what changed and decide whether it helped.

Useful reuse provides stronger evidence than a pleasing isolated sample. A skill that helps make a second, materially different project has demonstrated more value than one that succeeds only on a rehearsed example. The library and benchmark systems preserve that distinction.

Search helps people find what the portfolio has learnt

Share supports discovery through measured metadata, human tags, qualified inferred labels and related work. Technical readers can explore the method, recipe or tool behind an output. Creators can browse by the appearance or result they want. [P44]

Where a compact decision model or conventional search method is sufficient, the system can use it instead of asking a large language model to re-evaluate the same item repeatedly. Corrections remain visible, access rules apply to derived indexes, and simply viewing a gallery does not initiate a new paid generation.

Woodhouse provides the history

Woodhouse was the earlier editorial and software-factory publication, including the original Share experience and public portfolio evidence. App Composer becomes the home for new creative-library, reuse and distribution work. The transition preserves eligible content, provenance and important links. [P45]

The useful inheritance includes the habit of explaining how a project was made and showing real results. App Composer carries that forward in a product another creator can use independently. The older publication remains part of the story without becoming a second competing centre of new product development.

7 The practical applications in the Composer family

Why the small applications matter

The mini-apps demonstrate that the platform can produce focused tools people would choose to use. Each has a clear job, its own identity and an appropriate boundary around data and execution. They can be useful alone and contribute reusable capabilities to a larger project. [P20]

Application	The job it does	What it contributes to the wider platform
Paperwork	Complete and mark up a PDF in the browser	Private file handling and dependable exports
LDraw	Create editable diagrams and drawings	A shared visual object for people and agents
Here / there	Collect and moderate visitor photographs and postcards	Consented intake and source-owned publication
Linkarooie	Present an authored profile and curated links	Independent identity and cross-app curation
Creator Hub	Combine a profile with approved visitor contributions	A complete composition with separate ownership
Link Check	Inspect a public link before sharing it	Observed metadata and useful launch diagnostics
Hello World	Prove a chosen setup can publish the expected app	Reproducible onboarding and destination verification
Launch Kit	Prepare the files that make a launch look finished	Portable branding and explicit asset adoption
Carry / forward	Demonstrate encrypted, selective assistant context	Understandable disclosure and portability boundaries

Paperwork

Paperwork handles the familiar moment when a PDF needs to be filled in, marked up and returned. A person can complete normal form fields, type over a scanned form, draw, highlight, add ticks or circles, place a visual signature or initials, then save or print the result. The page, zoom, mark placement and undo history remain coherent within one editor. [A01]

Its browser-local design is part of the value. The supported editing path works on selected files in memory without an account or automatic upload. Native fields can remain editable when the output is reopened, while added marks become page content. A signature image is a visual mark; it does not create a certificate-based digital signature.

The completed experience gives particular care to accurate placement previews, phone controls and clear unsaved or exported states. Its supported PDF boundaries are explained at the point of use. The application contributes private file processing, deterministic output checks and focused interface patterns to Composer.

LDraw

LDraw provides an editable drawing and diagramming workspace built on Excalidraw. People can start with a blank scene, project artwork, reusable components or a source-grounded architecture diagram, then retain the native scene alongside PNG or SVG exports. [A02]

The important technical property is that the human editor and agent interfaces use the same scene model and renderer. An agent can create or revise a diagram, and the owner can continue working on it directly. The underlying structure survives instead of becoming a picture that can only be regenerated.

Local storage and a previous snapshot help protect the work. Revision checks prevent an agent or another tab from silently overwriting a newer drawing. Source credit and intended use travel with reusable assets. LDraw demonstrates how agents can participate in a visual application without creating a separate, incompatible editing experience.

Here and there

Here / there is a moderated collection of photographs and visitor postcards. A visitor contributes through an allowed submission path, an owner reviews the material, and approved moments can appear publicly or in a connected application. Invitations, upload bounds, consent and publication remain deliberate. [A03]

The application owns its photographs and moderation decisions. A consuming app checks whether an item is still eligible rather than assuming that a previous approval lasts forever. A later withdrawal therefore affects future controlled display.

Its reusable value extends beyond a guestbook. The same pattern supports carefully scoped submissions, review queues and consented enquiries in a creator's broader project. Private contact details stay separate from public content and receipts.

Linkarooie

Linkarooie gives a person a small, authored presence on the web. It brings together a profile, curated links, selected highlights and, where connected, approved visitor moments. The owner controls what appears and how it is arranged. [A04]

It works as a standalone profile and can cooperate with Here / there through a defined contract. Linkarooie owns curation and profile records; the guestbook owns the original photographs and their current eligibility. Each application keeps its own release and data responsibilities.

The completed editor and public experience preserve the creator's identity rather than imposing a universal founder-branded page. Drafts, publication and hiding are meaningful states. Revision checks protect newer work from stale changes.

Creator Hub

Creator Hub demonstrates how Linkarooie and Here / there become a coherent application together. The person has a public profile and a place for approved contributions, with native owner controls for editing and moderation. The pattern can be reproduced into an independent owner's resources and adapted to their identity. [A05]

Its significance is architectural as well as practical. The two apps cooperate without merging all their data or granting each other unrestricted access. They retain separate source histories, databases and authority over their own records.

The existing example repositories and fictional showcase content provide learning material. They are examples of the pattern, rather than separate customer organisations or evidence of commercial adoption.

Link Check

Link Check answers a useful pre-publication question: what does this public destination actually expose, and what needs fixing before it is shared? It inspects page identity, search and social metadata, image responses and conflicting or missing information, then produces an understandable report. [A06]

Its public-fetch boundary is deliberately bounded. It handles redirects, size and time limits, and private-network restrictions. The report distinguishes observed responses from unavailable information.

An approximate social preview helps the person review the intended presentation. It does not prove that a particular social network has fetched the link, and metadata inspection does not guarantee search ranking. The selling point is concrete diagnostic evidence that can feed the next repair.

Hello World

Hello World is the small independent application used to prove that a person's setup works. The goal is to serve the expected source from the intended account and destination. It does not need a database or real user information to demonstrate that. [A07]

Get Ready supplies the guided preparation around it. Hello provides the actual result that confirms the chosen agent, repository and deployment route can work together. Exact application and source identity matter because an unrelated page returning successfully would not prove the setup.

The same preparation and verification patterns support progressively more capable projects. A small first success gives the person confidence without disguising the responsibilities of a larger application.

Launch Kit

Launch Kit converts selected artwork and text into useful launch assets: icons, touch and maskable icons, profile cards, social images, a manifest template and a portable package with installation guidance. The creator reviews the actual output images before adopting them. [A08]

The supported workflow runs locally in the browser. It records the source, settings and output identities so the result can be inspected and reproduced within the documented environment. Importing the package into a project is a deliberate action that preserves unrelated configuration.

It solves a small but visible part of product completion. A working application still needs the correct icon, useful share image and coherent presentation when installed or linked from elsewhere.

Carry forward

Carry / forward makes portable assistant context tangible through fictional profiles. A person can choose which invented memories to disclose, inspect the outgoing material, encrypt an export and reopen it later. It provides a clear demonstration of selective disclosure and context ownership. [A09]

The current accepted demonstrator and its maintenance scope preserve fictional data. The broader backlog's security work governs any separately authorised move towards operational private memory. Completion does not silently expand the demonstrator's data permissions.

The technical lessons include authenticated encryption, versioned data envelopes, keeping the passphrase separate and refusing tampered or incorrectly decrypted content. It also teaches an important limit: encryption cannot promise recovery of a forgotten key or protection from a compromised device.

Additional catalogue projects and bounded demonstrations

The backlog extends beyond the established mini-apps. These projects have distinct intended jobs and acceptance scopes; some are deliberately bounded demonstrations rather than universal services. [P46]

Tower Stack is a satisfying, remixable arcade-game example. It contributes a small playable creation path with game-specific controls and feedback. **Ask Me Anything** is a candidate for moderated questions and optional video replies, demonstrating a creator-audience workflow.

Migration Map reconciles old and new URLs locally and produces verified redirect exports for supported Cloudflare destinations. It makes one difficult part of moving a website visible and reviewable.

Oddpull is a working-name capsule-collecting demonstration built through Composer's actual creation path. Its scope emphasises free original-character pools, explicit odds, authenticated collection ownership and retry-safe receipts. The planned interface and native editorial administration sit above a common domain model. Its bounded scope does not imply paid random rewards or a final approved commercial brand.

Company Watch produces saved, source-linked briefs explaining what changed in selected companies' public information. Its AI Infrastructure Watch scenario tests the same application on appropriate issuers. Deterministic calculations retain their reporting periods, units and evidence; optional AI explains the admitted facts. The application is a research tool, with clear coverage and unknowns. [P47]

Team Hub, **Incident Desk** and **Plugin Trust Centre** demonstrate private collaboration, incident decisions and evidence about plugin trust. They belong to the operational and team capabilities of Composer. Their value is the complete workflow and inspectable outcome, rather than the presence of another dashboard.

8 The wider portfolio and what each project proves

Independent products with a shared method

The wider portfolio includes music, games, creative collections, desktop recovery and a connected-device concept. These products have their own users and identities. Composer can connect their context and reuse proven capabilities without taking ownership of every runtime or making every product look the same. [P21]

The variety is useful because different applications reveal different weaknesses. A music player tests continuity. A game tests response and legibility under pressure. An archive tool tests careful access to valuable local files. A collecting platform tests ownership and repeatable transactions. Together they provide a demanding environment for improving the creation method.

LoftwahFM

LoftwahFM is an artist-led music platform combining a catalogue, continuous browser listening, radio-style programming, original and custom music, and reusable listening experiences. A person can discover tracks, queue them, browse the site and share what they are hearing while playback remains coherent. [A10]

The completed business-audio scope brings together background music, custom soundtracks, branded announcements, music on hold and private customer material. Locations and zones can have appropriate programming, opening hours and device state. Rights and access remain tied to the relevant catalogue and customer relationship.

Its technical contribution is a dependable media service with a versioned client contract. Games can use LoftwahFM radio through their own adapters while preserving their gameplay, sound effects and fallback behaviour. A public embed does not gain access to private tracks merely because the listener has signed in elsewhere.

LoftwahFM also gives Media Studio a real creative destination. Artwork, release films and promotional assets can support actual listening experiences. Analytics retain clear definitions so a playback start is not casually described as a unique listener or a sale.

The completed engineering scope does not override the owner's separate hold on live checkout. The product can be explained as a capable business-audio offering without implying that payment collection has been authorised, that customers already exist or that every third-party recording is cleared for every use. [A10]

LOFTWAH FIGHTER

Fighter is a local-first, real-time squad battler built around authored characters, charge timing and meaningful combat choices. Players collect characters, assemble a small lineup and choose when to commit moves. Story, Quick Fight and Tournament provide different ways into the game. [A11]

The current authored launch grammar includes a six-character roster, character-specific moves and cut-ins, an introductory story, missions, collection systems and a three-round

tournament with carried health and earned trophies. Accessories, statuses, switching and character development give the player reasons to experiment.

The completed experience makes those decisions legible. The active fighter and important actions receive appropriate visual weight. Charge is easy to read. Reserve information stays compact. Phone and desktop layouts serve the game rather than forcing the player through a conventional website interface.

Underneath, deterministic mechanics and authored content contracts support reproducible testing and balance work. Saved progress has a versioned recovery path. Original artwork and sound are applied in the game itself. A later character or campaign can be created within the supported gameplay structure, validated and released without discarding the game's identity.

For Composer, Fighter contributes content-driven engines, character pipelines, game-specific interfaces and rigorous outcome testing. It demonstrates why a game needs more than a generated scene and a collection of buttons.

SHOALSHOT

SHOALSHOT is an original tide-pool target game. Players read moving shoals, place shots, select launcher ranks and use finite tools while completing tasks and building a collection. Authored formations, special encounters and migration events give the water its own rhythm. The aquarium home provides a place to enjoy the world between dives. [A12]

Its current content grammar includes eighteen creatures and three launcher families with three ranks each. The completed direction adds expressive companions, appropriate dialogue, polished progression and sound that supports the play. A finished dive can produce a designed result image and caption based on what actually happened.

The project is a clean-room original. Its successor research informs the work without making it a rebranded or decompiled copy of Fishing Diary. Browser-local progress has import, export and recovery behaviour. Local result cards describe a personal run rather than claiming server-verified competition.

SHOALSHOT contributes lessons about dense moving scenes, readable phone interaction, formation and encounter design, saved progress and the relationship between soundtrack and action. Its demanding visual and performance conditions help expose whether an agent-produced change improves the actual experience.

Bubbles

Bubbles brings bubble-shooter play into a polished browser cabinet. Clear aim, readable outcomes and satisfying matches are the core experience. Different modes use the common engine while preserving their own rules and consequences. [A13]

Classic retains a defined ruleset. Practice supports extended solo play. Race introduces deterministic rivals. Arcade and Puzzles provide their own objectives and authored challenges. Achievements, replayable boards and appropriate leaderboards give results a life beyond one session; optional cabinet experiments retain their declared scope.

The technical foundation separates deterministic simulation from presentation. Rules and configurations are versioned. Eligible submitted results can be replayed and derived by the server rather than accepted because a browser reported a score.

For the player, this supports fairer and more understandable records. For Composer, it provides reusable patterns for simulation kernels, mode separation, result verification, responsive game controls and performance-sensitive feedback. The project also shows why visual polish must reach the aiming, animation and menus that players use every session.

Protocol 11

Protocol 11 is an illustrated card game built around eleven canonical protocols. It supports one person playing against rivals or two to four total players through the supported local and online modes. A guided first hand teaches the real rules, while clear turn ownership and legal-action feedback help the player concentrate on decisions. [A14]

House Circuit supplies an authored progression of tables and personalities. Cosmetic cards and keepsakes can reflect achievements without buying a gameplay advantage. Rival conversation responds to appropriate public events and respects the distinction between public information and hidden cards.

The architecture uses shared rules with server-authoritative online rooms and explicit coordination of state. Reconnection, turn ownership and outcome records must remain coherent when people leave, return or act at nearly the same time. Cloudflare Durable Objects provide an appropriate coordination primitive for the room model. [A14] [C05]

Protocol 11 contributes multiplayer authority, reconnectable sessions, rules shared between people and bots, and a strong example of conversational character behaviour that cannot alter hidden game information or randomness.

Foilbury

Foilbury turns original character artwork into a collecting experience. A person can browse a cabinet, open a capsule, inspect the front, back and finish of a card, keep a particular serialised copy and build an album. Authored reveals, collector identity and collection milestones make the work feel like a coherent world. [A15]

The product also includes an authoring and administration system. A catalogue owner can curate editions, manage their own economy, invite an appropriately scoped team and operate through the website, Discord, REST, CLI or MCP. The same domain commands and award ledger govern those interfaces.

That shared authority is crucial. A retried request must not award another copy or reroll the result. An editorial change must not rewrite a person's ownership. A bot owner must not acquire platform-administrator powers by using a different interface.

Art Lab keeps editions, source relationships, candidate feedback and publication decisions visible. The result is a strong environment for reusing approved original characters across products while retaining the identity of each artwork edition. It is a digital collecting system; its ownership model does not require a cryptocurrency or NFT claim.

The public showcase uses approved original work. Member and private collections retain separately controlled audiences.

Foilbury contributes transaction discipline, retry-safe rewards, scoped administration and a common operation model across human and agent channels. It also tests whether a large creative collection can remain attractive, navigable and properly attributed.

MAX

MAX focuses on a concrete accessibility task: helping someone find the walker they already own. The central interaction is FIND MAX, an audible and visible response, and an immediate STOP control. A companion application, authorised support people and a device connection support that bounded task. [A16]

Voice can provide a more natural input, while deterministic capabilities control the action. The core find operation does not require a language model to invent a command. The typed protocol establishes what the app, gateway and device are allowed to do.

The project separates a simulator from physical-device evidence. A successful demonstration through the simulated control chain proves a particular software interaction. Physical behaviour needs its own observed acceptance. The product scope is a connected locator and companion, without extending the description into medical monitoring or emergency response.

MAX contributes accessible controls, careful hardware-software boundaries and a useful model for bounded AI interpretation. It makes the consequence of an action especially clear: the system must operate the intended capability and allow the user to stop it.

Liferaft

Liferaft is a desktop recovery tool for email archives. A person opens an MBOX export, searches by sender, subject, phrase or attachment, inspects the source message and recovers the wanted file or original message. The archives stay read-only while a separate local index makes them easier to search. [A17]

Optional local agent access uses the same search and extraction engine. The person chooses the allowed archives and recovery folder. The agent can operate within that scope, and a later change to the source or permissions invalidates stale evidence where necessary.

This is valuable when the data exists but finding the one important document is difficult. The tool remains useful without an agent. If the person chooses a cloud model, any snippets deliberately sent to that model have that separate exposure; a local connector does not make the model itself local.

Liferaft contributes evidence-bound retrieval, source preservation and human-agent parity in a desktop workflow. Its maintained packages carry their actual platform support. The portfolio story does not assume identical releases on every operating system.

Bogan Hustler

Bogan Hustler is an Australian-themed fictional trading game. Players make market decisions, travel, manage limited cargo and respond to uncertain events. The completed overhaul improves the information available to the player, separates Cred from Heat, sharpens run pacing and brings short, readable arcade combat into a phone-oriented experience. [A18]

The engineering work centres on explicit state transitions and a balance harness. The information design matters because decisions should have understandable consequences. A distinctive authored voice gives the game its character while a family-friendly content option supports a different presentation.

It is an independent game in the wider portfolio. Its contribution to Composer is the method and qualified reusable work, rather than an assumption that it is already mounted inside the platform. It provides practical lessons about economy modelling, pacing, simulation and product personality.

Preserved projects and the lessons they leave

Pirates is preserved as a paused project. Its source-first reconstruction work on Master of the Secret Sea established a strict approach to behavioural parity: understand the original states, transitions, assets and quirks before modernising them. That is a useful engineering lesson for migration and compatibility work. The owner's indefinite pause remains part of its status. Reference assets retain their original rights boundaries. [A19]

Plans With You, formerly Club, is retired. Its small service concept explored calendar commitments, member journeys, notifications and the way product positioning shapes an otherwise simple workflow. It belongs in the portfolio's history rather than in a list of active services accepting customers. [P21]

Asset Hunter is also retired. Its discovery and provenance work informed the importance of recording where resources came from and how they may be used. Those lessons continue in the newer creative-library direction without implying that the old service remains active.

Woodhouse supplied the earlier software-factory narrative, public project evidence and Share experience. Its useful patterns pass into Composer while its historical material remains attributable. Supporting items such as repository templates, setup examples and portfolio-analysis scripts are infrastructure or evidence, rather than additional customer-facing products. [P45]

9 WordPress migration and the EmDash opportunity

Preserve the business function

Many website owners understand the site they need but do not want to maintain its underlying machinery. App Composer offers a managed path from an existing website to an appropriate owned application, including EmDash and Cloudflare where they fit. The first task is to understand what the existing site actually does. [P48]

An assessment covers content, forms, users, editorial work, commerce, media, URLs and the plugins or custom code that supply important behaviour. It also identifies what the owner likes about their existing workflow. A migration is successful when those necessary outcomes survive and the owner can continue making changes.

EmDash supplies the content system

EmDash is an open-source content management system built on Astro. Its Cloudflare deployment uses Workers for the application, D1 for content and settings, and R2 for media. It provides administration, plugins and an MCP server for supported AI-tool access. [C01]

Composer supplies the surrounding guided experience: requirements, application composition, agent handoff, preview, deployment, operating controls and the next change. An owner who needs editorial administration can keep an appropriate CMS workflow. A simple Markdown-led site can use a lighter static route when that is the right fit. [P01] [P02]

Migration includes comparison and recovery

The completed migration journey begins with an inventory and an explicit mapping of required features. It creates a separate preview, imports or transforms eligible content, prepares media and URLs, then checks representative user and owner tasks. Redirects and private-content behaviour receive deliberate attention. [P48]

EmDash's import tooling provides a starting point for WordPress content and media. Custom shortcodes, page builders, fields and plugin behaviour can need adaptation. The plugin-porting documentation reflects this need to translate the required behaviour into the destination's model. [C02] [C03]

Composer makes the differences reviewable. The owner can compare pages, check a form, edit an article and verify that important journeys still work. A controlled cutover retains the recovery material needed to respond to a failed release or missing requirement. The next owner edit is included in the acceptance, because a migrated site must remain usable after handover.

Security improves through clear responsibilities

Managed infrastructure reduces the work of maintaining underlying servers and runtime services. EmDash's sandboxed plugin format adds declared capability boundaries to supported plugins. Native plugins have a different execution model and do not share that same isolation boundary. [C04]

These are useful architectural benefits. Application permissions, secrets, dependencies, content handling and business logic still require careful engineering. Composer makes those responsibilities visible through its recipes, trust information and operating workflows.

A credible alternative for the right owner

The selling point is an owned site with an understandable editing experience, manageable operation and a supported way to keep improving it. A migration decision can favour EmDash, a direct Astro application, a hybrid arrangement or retention of an existing specialised service. [P48]

A complex commerce workflow, for example, can retain its appropriate checkout owner while the surrounding content experience changes. The system should explain that choice and its trade-offs. It does not need to reproduce every WordPress mechanism to preserve the business result, and a successful import alone does not establish full functional parity.

10 How the system works underneath

A connected workspace with distinct owners

Engineers can understand App Composer as a control plane: the place that coordinates project context, discovery and permitted actions across systems. The connected systems retain authority over the work they own. A creator sees one coherent project without requiring every database and application to become one service. [P03]

This produces clear responsibility boundaries. GitHub holds the authoritative source and change history. Studio holds creative originals and production records. An independent app remains authoritative for its domain records. EmDash manages its relevant editorial lifecycle. The selected provider reports infrastructure state. Composer records relationships and presents the useful combined view. The user retains the appropriate ownership and control across those systems.

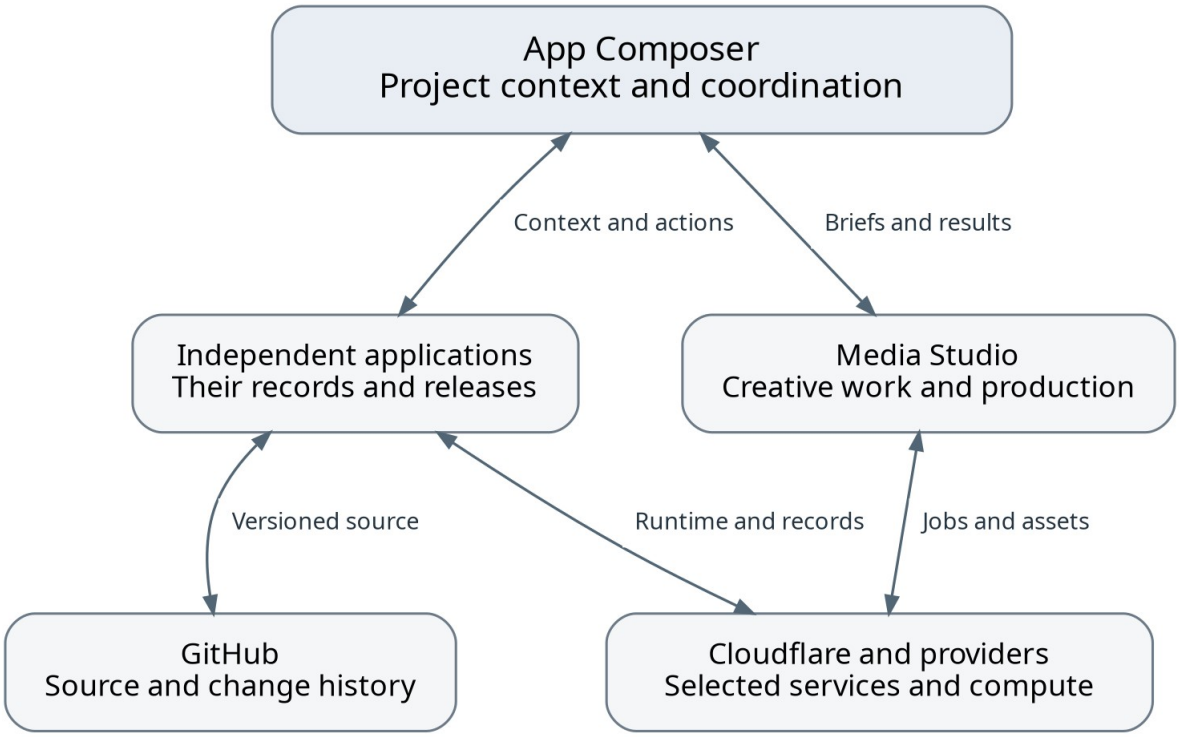


Figure 1 Selected responsibility relationships. Each system remains authoritative for the work it manages. Composer connects context and permitted operations; the relationships apply where the project uses those services.

The diagram shows responsibility, rather than a requirement that every project use every service. A local PDF tool can remain local. A multiplayer game can use a room coordinator. A film can use a specialised renderer. Each choice is made for the task.

Cloudflare services have different jobs

Service	Plain-language role	Architectural significance
Workers	Runs the application when people use it	Managed execution for appropriate APIs, services and frontend delivery
Workers AI	Runs selected AI models without the owner operating GPU machines	Inference through a binding or API, with model-specific capabilities and usage
D1	Stores structured records	Managed SQL with SQLite semantics and defined capacity and recovery behaviour
R2	Stores images, audio, video and documents	Object storage with explicit access policy and separate metadata relationships
Durable Objects	Coordinates a live room or stateful session	A named object with associated compute and strongly consistent local storage
Queues	Holds background work until it can be processed	At-least-once delivery requires duplicate-aware consumers
Workflows	Keeps a multi-step process moving across waits and retries	Durable progress and external-event handling within defined execution limits
AI Gateway	Helps observe and control AI requests	Usage visibility, routing, caching and resilience for supported providers

These are underlying platform capabilities. Composer's contribution is the supported composition, user experience and verification around them. Services are included when the workload needs them; a simple application does not benefit from an unnecessary stack. [C05] [C06] [C07] [C08] [C09] [C10] [C11] [C12]

A reusable block has an explicit contract

A block records what kind of thing it is, who owns it, which version is selected, what it accepts and produces, and what dependencies or permissions it needs. Licences, provenance and evidence accompany the relevant kinds of material. [P03] [P49]

The catalogue can therefore distinguish an image, an interface component, a prompt recipe, an executable action and a deployable application. That distinction matters for both people and agents. An image can be adopted into a design; it cannot be called as if it were an API.

Typed composition checks the relationship between capabilities. Here / there can supply an eligible public item to Linkarooie through an appropriate interface. The consumer does not need direct access to the producer's database. If the producer withdraws the item, the contract still respects current eligibility.

A reusable blueprint also needs a real second use. Copying a source pin into a catalogue is less demanding than rebuilding the supported project in another owner's resources. The latter exercises dependencies, credentials, retained state and the ability to continue after the original demonstration is gone. [P49]

Source versions and data have different lifecycles

The released code should identify the exact source that produced it. An update should be reviewable and should preserve intended local customisations. A code rollback restores a code version; it does not automatically reverse database changes, restore deleted media or cancel an external effect. [P09]

Recovery therefore considers code, configuration, structured records, objects and account access separately. A useful plan records compatibility and the supported restoration procedure. Restoring into an isolated destination provides stronger evidence than simply observing that a backup file exists.

This distinction is central to ownership. Someone who can download the source but cannot recover their records or reconnect their services has only part of what they need to continue.

Media work uses suitable execution environments

Workers is useful for coordinating production, enforcing permissions and serving results. Heavy video encoding, native creative tools or GPU workloads may require a browser capability, container, remote runner or specialist provider. Workflows can retain progress while that work happens elsewhere. [C13]

The Studio keeps its common project and job model across qualified executors. It can retain provider output after an interrupted save and reconcile an uncertain job before another paid request. Large media objects are stored as assets, with references passed through orchestration where appropriate. [P28]

This is also where optional Fungi Computer, Oyster, Shed, Veil or other execution integrations fit. They provide particular upstream capabilities through explicit adapters. Composer owns the project experience; the upstream system owns its execution or credential boundary. Qualified optional integration does not turn an independently maintained service into a mandatory dependency for every customer. [P50]

Interfaces serve the product

Astro and EmDash support the core application and content journey. React and TypeScript serve interactive workspaces where appropriate. Games can retain Phaser or another suitable rendering engine with an accessible interface layer for menus and controls. Browser-local tools can avoid a server round trip for work that fits the device. [P01] [P28] [A01] [A11] [A13]

The important commonality is understandable behaviour and maintainable ownership. Components, navigation, state transitions and accessibility are shared where they help. A game remains a game, a diagram editor remains an editor, and an administrative screen presents the detail its user actually needs.

11 Agents ownership and dependable operation

Bring a supported agent and keep the account relationship clear

App Composer accommodates the owner's supported agent and accounts. The integration identifies which environment performs the work, which tools it can use and which entitlement covers the operation. Portable context and returned results reduce dependence on one conversation or model. [P25]

Different tasks can use different capabilities. Difficult planning or review may justify a stronger model. A bounded implementation can use a more economical capable executor. Deterministic code can perform calculations or transformations that do not benefit from inference. Selection is based on observed fitness for the job and the relevant constraints.

An optional hosted helper or remote environment can add continuity when the owner closes a laptop or needs specialised compute. It has its own cost and authority. Existing chat access is not treated as evidence of API credit or permission to operate a remote service. [P02] [P50]

Autonomy is work inside a clear grant

A person can authorise a useful continuing workflow once, with an understood project, destination, budget and scope. The agent can then perform the routine work covered by that grant. The interface should avoid repeated requests to approve the same already-authorised action. [P25]

At the same time, the system distinguishes reading, editing, executing, deploying and publishing. A contributor who can comment on a film does not gain permission to publish it. An agent that can inspect a project does not receive a broad credential vault. Access is checked where the operation occurs, including after relevant revocation.

EmDash's scoped MCP model illustrates the same principle at the underlying content-tool boundary: token scope and user role both matter. Composer applies the relevant permission model to the wider project journey. [C14]

Incoming material is evidence

Projects may consume webpages, emails, documents, source files and model outputs. That material can be useful without becoming an instruction that changes the system's authority. A source can supply facts or creative references; it cannot grant itself permission to publish, spend or read another project's records. [P03] [P51]

This distinction makes connected research and inbox workflows more dependable. The application retains the origin of information, validates it for the intended use and keeps operational decisions within the authorised task. Sensitive or private source material does not become public merely because an agent used it in a draft.

Job state makes failure understandable

Useful long-running work needs explicit state. A job can be proposed, admitted, running, waiting for a provider, awaiting review, completed, failed or cancelled according to its real lifecycle. An uncertain response is recorded as uncertain until it can be reconciled. [P28] [P30]

This matters whenever an operation has an external effect. A lost response after publication does not establish that nothing was posted. A retry after a media-provider timeout can create another charge if the original request actually succeeded. The system checks the relevant source of truth before repeating such work.

Cloudflare Queues provides at-least-once delivery. Applications therefore need idempotency or equivalent duplicate handling where repeated messages could create repeated effects. Foilbury's award ledger and Studio's production jobs illustrate why that property has a direct user benefit. [C12] [A15]

Operational attention is tied to user outcomes

Composer's operating view brings together app health, connections, recent activity, costs and recovery actions. An incident workflow can gather evidence, propose or perform a permitted repair, verify the relevant user journey and report the result. [P30] [P52]

The valuable observation is specific. A deployment exists, an endpoint responds, a form submits successfully and a person completes a real task establish progressively different things. The interface retains that scope rather than converting every green signal into a claim that the entire product is healthy.

A scheduled workflow also needs an understandable lifecycle. The owner can inspect what is scheduled, its destination and timezone, pause it, see its attempts and understand any missed or uncertain outcome. Recurrence should be a managed commitment rather than a hidden process that continues spending after it has lost its purpose.

Teams add clear ownership

Workspace roles identify who owns the service, who maintains it and who can perform consequential actions. Environments separate development, previews and production. Quotas and cost attribution make shared resources understandable. A team's approved application pattern can guide self-service without granting unrestricted access to every account. [P24]

Enterprise connectors, change-control processes and alternative deployment targets are qualified against an actual need. An assurance record can explain relevant controls, evidence and data-location choices. It does not turn an architectural design into a certification or imply that an upstream provider's certification automatically covers the whole application.

Independence is tested through continuation

The strongest ownership test is practical. Another owner can reproduce the supported project, connect their own credentials, use their own data and operate it after the original Composer installation is detached. They can export relevant work, make a later change and exercise the documented recovery path. [P06] [P49]

The open-source core avoids a compulsory licence callback for the created application. Cloud services and third-party providers still have their own terms and operational requirements. Independence means those dependencies are explicit and under the owner's control, with a tested exit where the product claims one.

12 Quality benchmarks and proof

Completion is an observed outcome

The high standard in this vision combines function, craft, ownership and continued use. A completed issue produces the intended behaviour in the appropriate environment. The evidence identifies what was tested and which version was observed. [P53]

A code test can establish a rule. A browser journey can establish an interaction. An exported file can be decoded and inspected. A film can be watched with sound. A game can be played on its intended devices. The owner's acceptance establishes whether the result meets their taste and purpose. These forms of evidence complement one another.

The defining benchmark

The idea-to-operation benchmark asks the system to deliver the whole intended outcome from an agreed brief and readiness state. Its six gates make the promise assessable. [P23]

Gate	The result required	Evidence that makes it meaningful
Useful application	The live project performs the intended job	Real task completion on the declared devices
Finished presentation	Interface, copy, identity and media work together	Inspection of the actual delivered experience
Accurate launch package	Marketing describes the real product	Matching claims, screenshots, links and asset versions
Real distribution	Approved material reaches the permitted destination	Publication receipts and accessible intended output
Continued operation	The project handles routine work and a later change	A declared observation window, recovery and release evidence
Proud to share	The owner accepts it without doing the repair work	A verdict attached to the exact delivered versions

Passing one gate cannot compensate for missing another. A beautiful trailer does not rescue a broken app. A correct calculation does not establish that the interface is usable. The required scope is declared before the run so difficult criteria cannot disappear after a failure.

Measure the work that the person still has to do

The strict autonomous outcome requires zero implementation or coordination interventions after the initial idea and agreed readiness. Setup and the final owner verdict are recorded separately. Human debugging, manual editing, deployment, uploading, recovery and task chasing all count as interventions. Work secretly done by another person still counts. [P23]

The final owner review is a judgement step. Feedback that repairs the result starts a recorded correction cycle, preserving the first-delivery verdict. Agent-only correction before delivery can remain autonomous while its time and resource costs are still counted.

This lets Composer report meaningful distinctions: accepted on first delivery, accepted after feedback, accepted with manual implementation, partial, blocked or awaiting review. Those labels help the product improve instead of hiding the intervention needed to make a demonstration succeed.

Benchmarks Studio compares methods fairly

Benchmarks Studio makes model, agent, tool and application evaluations discoverable and repeatable. A comparison records the task, inputs, versions, environment, quality criteria and available costs. It can compare full workflows as well as individual model outputs. [P54]

The portfolio supplies real scenarios. A usable animated game asset must work in its intended game. A film revision must preserve accepted material while correcting the requested element. A migration must retain the owner's important workflows. A company brief must preserve the meaning of its figures and sources.

External products can provide references for capability and quality. A comparison does not establish parity merely because both interfaces contain similar buttons. The test needs the same meaningful task and an inspectable result.

Creative acceptance needs human judgement

Automated checks can find missing media, bad dimensions, invalid files, broken timing and some visible defects. They can also support structured critique. The owner or their chosen reviewer decides whether the result represents them well. [P39] [P43]

Pairwise comparison, optional written notes and precise frame or element feedback make that judgement useful for the next attempt. The system preserves disagreement between machine scores and human preference. A high model rating does not override a clear rejection of the delivered work.

Evidence includes failures and coverage

The benchmark retains unsuccessful attempts, correction costs, unsupported cases and missing measurements. It distinguishes observed values from estimates. A result based on one rehearsed project remains a demonstration until the claim has been tested across appropriate independent or held-out work. [P23] [P54]

This discipline strengthens sales material. A narrow, repeatable result with clear conditions is more useful to a serious buyer than an impressive number that cannot be connected to their workload.

13 Economics and the value of choosing well

Measure the accepted result

The useful economic question is what it costs to deliver an acceptable outcome and keep it useful. That includes software work, inference, media generation, failed attempts, infrastructure, external tools and human intervention. A low price per model request can still produce an expensive project if many attempts are rejected. [P23] [P55]

Composer makes those costs more visible. It can explain the expected incremental cost before a job, record the actual available usage afterwards and separate unknown amounts from zero. Provider subscriptions, API balances, free allowances and optional commercial services remain different categories.

Use the smallest appropriate method

Many operations are predictable transformations. Cropping an image, validating a field, calculating a period change or packaging an icon does not inherently require a large language model. A compact classifier may suit a repeated categorisation task. A stronger model may be worthwhile for ambiguous planning, complex implementation or creative judgement. [P44] [P55]

The system can combine those approaches. It can cache an appropriate result, reuse an approved asset, avoid regenerating unaffected media and choose a qualified executor for the task. The improvement is judged by cost per accepted outcome at the required quality.

This is consistent with the mission to increase useful Workers AI adoption. A smaller number of well-chosen calls can make an application affordable enough to be used regularly. Wasteful inference would weaken that adoption.

Understand the infrastructure bill

Cloudflare's managed services can reduce conventional server administration. They still have usage limits and billing rules. Storage, requests, compute, databases, inference, domains and outside services can all contribute to the total. [C06] [C07] [C08] [C13]

R2's absence of egress bandwidth charges is useful for media delivery. Storage and operations can still cost money, with retrieval charges applying to the relevant storage class. The product should connect those costs to the actual workload rather than advertise storage as universally free. [C15]

Cross-cloud comparisons therefore use equivalent outcomes and reasonable optimisation on both sides. They include operational work where it can be measured. A generous allowance or introductory credit can help a trial, while long-term affordability depends on the ongoing cost model. [P55]

Commerce connects to a clear transaction owner

The completed commerce direction supports appropriate Shopify, Stripe, PayPal and Printful connections. These integrations join the broader project workflow: product presentation, checkout, verified payment state, order follow-up and fulfilment where authorised. [P56]

Each order has a clear checkout owner. External payment state is verified, and an uncertain response is reconciled before another charge or fulfilment request. Preparing a product page, collecting payment and starting physical production have different consequences and permissions.

This creates practical value for creators who want to sell something alongside their content or application. It also allows an existing specialised store to remain in place while the rest of the experience improves.

Open source and commercial value can coexist

App Composer's open-source ownership model gives a person a useful project they can continue to operate. Optional commercial value can come from reliable hosted operation, managed support, agency workflows, specialised services, training or team capabilities where customers find them worthwhile. [P01] [P02]

Those are business possibilities rather than an invented price list or claim of proven revenue. The reason to pay must be continuing value: less operational effort, dependable execution, easier collaboration or a service the customer does not want to run themselves.

The platform should be judged by retained owners and useful repeat activity. Completed engineering makes that proposition possible. Customer acquisition, willingness to pay and sustainable support economics still need their own evidence.

14 The selling points and practical use cases

A complete project reduces handoff work

The broadest selling point is that the person can keep one intention coherent across software, design, media, launch and operation. They do not have to reconstruct the brief every time work moves to another tool. The project's history explains what was chosen and why the current version exists. [P01] [P03]

That benefit becomes concrete in several different situations.

A creator launches an original project

A creator starts with an idea and a few references. Composer helps produce a working destination, an appropriate identity, artwork and a short demonstration. Supported launch material is prepared and published under the existing grant. The creator returns to improve the same project using actual audience observations.

The value is a finished public presence with editable source and a manageable next step. A single attractive output would leave much of that work undone.

A business owner changes an existing website

The owner brings a real site and identifies the workflows they rely on. Composer maps the requirements, shows a preview of an appropriate destination, checks the important content and interactions, then supports the cutover and the next editorial change. [P48]

The value is confidence about what will change, what remains supported and how the site will be maintained. A migration decision becomes understandable to the person responsible for the business.

An agency repeats a successful delivery

An agency uses reviewed patterns for client applications, applies each client's identity and accounts, and manages changes through a consistent process. Shared lessons become better reusable components while client records and permissions remain isolated. [P24] [P49]

The value is repeatability with room for genuine client differences. The agency can preserve technical ownership and offer continuing service based on work the client understands.

A developer builds an unfamiliar product

A developer can begin with a tested deployment recipe and relevant capabilities, inspect their contracts and add original code where needed. Studio supplies usable creative assets. Benchmarks and release evidence provide a way to judge the finished result. [P03] [P54]

The value is less repetitive assembly and better continuity around the code. The developer retains source-level control and can replace parts as the project grows.

A platform team offers governed self service

A team can request a supported application pattern, deploy into the proper environment and operate within its role and quota. The service has a named owner and a clear path for changes, recovery and cost review. [P24]

The value is a useful internal platform that connects requests to functioning applications. Approval and evidence belong to the work itself, rather than to a disconnected form that says a task is complete.

Learning material is part of adoption

The Learn experience accompanies the actual creation path with tasks, expected outcomes and recovery help. A newcomer can understand what a choice gives them and why the system asks for a particular account or permission. Public examples remain available without forcing a person to begin setup. [P01]

Reproducible prompts, recipes, diagrams and how-made stories let technically curious visitors go deeper. An example becomes more valuable when the reader can inspect its source, adapt it and see where the method succeeds or fails.

Research and growth connect to the product

Research discovery, SEO work and advertising workflows help a creator understand the audience, prepare useful material and measure the response. Source-linked research retains evidence. Advertising claims refer to the delivered product. Search diagnostics lead to inspectable repairs. [P57]

Packaging work can compare title and thumbnail variants, while the system retains the hypothesis behind each. Audience behaviour, personal preference and machine critique remain different kinds of information. The application can improve the next campaign without pretending that a popular reference guarantees the same result.

The distinction a buyer can assess

An AI coding tool supplies implementation ability. A CMS supplies editorial structure. A cloud platform supplies services. A media application supplies production capabilities. A publishing tool supplies distribution. App Composer earns its place by connecting those capabilities into an owned project that can be revisited, evaluated and operated. [P01] [P03]

That position is complementary to capable existing tools. Its defensibility grows through the quality of the complete journey, reliable contracts, tested recipes, useful project history and demonstrated results. The existence of more integrations is valuable only when those connections improve what the person can accomplish.

15 What success looks like

A demanding demonstration

Consider a creator who asks for a small playable browser game built around an original character. They provide the audience, a few visual references and the desired tone. Their supported agent, accounts and spending boundaries are already ready. This is an illustrative combined journey through the completed platform. [P23]

Composer turns the request into a concrete brief and chooses appropriate existing capabilities. The agent builds the game and tests the important behaviour. Studio develops or adapts the permitted character material, produces usable assets and creates a short trailer from real play. Brand Studio and Launch Kit keep the public presentation coherent.

The owner can open the actual game on a phone and desktop. The controls work, the text is readable and saved progress behaves as promised. The trailer shows the delivered game. The launch page explains it accurately and the sharing image appears correctly in the supported diagnostics.

The system deploys to the intended account and publishes through the authorised channels. It retains the source, assets, release identity, selected campaign material and publication receipts. Scheduled work continues over a declared observation window. A representative routine failure is handled within the agreed scope and checked against the real user experience.

Later, the owner asks for a new game challenge and a shorter trailer. The agent changes the same project, preserves existing good work, produces the revised media and verifies the release. The owner can compare versions and keep both the old and new creative sources.

The final judgement is personal and concrete: would they share the delivered result under their name without doing the implementation or repair work themselves? That is the standard the platform is designed to meet.

A successful portfolio produces reusable knowledge

Each finished project can contribute something useful to the next. A better upload boundary from the guestbook can improve another intake workflow. A precise review operation from Studio can inform preview feedback. A transaction lesson from Foilbury can improve duplicate handling. A game performance lesson can improve another interactive application.

That transfer is deliberate. A candidate block has its source, assumptions and contract recorded, then earns reuse in another real setting. The portfolio becomes a collection of working products and a body of tested methods. [P49]

Product quality still needs commercial judgement

Even a completed high-quality backlog does not automatically create demand. The platform can fail commercially if the full journey is too difficult to discover, setup remains too demanding, the cost exceeds the value or customers do not return. It can also lose coherence if optional capabilities overwhelm the person who only wants to make one useful thing. [P02]

The response is to measure complete journeys and retained use. The product can offer a simple starting point while allowing deeper control. It can compare actual outcomes with alternatives and improve the handoffs where people struggle. It can keep optional services proportional to their demonstrated value.

The long term result

A successful App Composer makes a larger range of ideas practical for more people. Its users can make small useful tools, maintain professional sites, create original media, develop games and operate connected workflows. They can start with limited technical knowledge and gain understanding as they need it.

For engineers, the result is inspectable software with clear owners, explicit boundaries and tested operating behaviour. For creators, it is work that reflects their intention and is good enough to share. For the wider ecosystem, it is recurring demand created by applications people choose to keep using.

The person returns to their project, asks for the next useful change and still recognises it as their own.

16 Plain language glossary

Term	Meaning in this document
Agent	AI software that can use tools to carry out a task within its available permissions
API	A defined way for one application to ask another application for data or an action
Blueprint	A versioned description and recipe for reproducing a supported project
Capability	A clearly described function that a person, app or agent can use
CMS	A content management system used to edit and publish structured website content
Control plane	The coordination and management part of a system
Deployment	A particular version of an application made available in a running environment
Deterministic	Producing the same defined result from the same relevant inputs
Durable state	Information that survives a process stopping or restarting
Federation	Independent systems cooperating while retaining authority over their own work
GPU	Hardware suited to many AI and graphics workloads
Idempotency	Handling a repeated request without repeating the intended effect
Inference	Running a model to produce an output from an input
MCP	Model Context Protocol, a standard interface through which supported AI tools can access capabilities
Native asset	An editable source format, such as a drawing scene or project file
Object storage	Storage designed for files such as images, audio, video and archives
Provenance	The origin and history of an item, including relevant source and version relationships
Receipt	A record identifying an action, its outcome and the version or external effect involved
Revision	A particular saved state of code, content or creative work
Runtime	The environment in which application code or a tool actually executes
Scoped grant	Permission limited to specified actions, resources and conditions
Serverless	A service model in which the provider manages the underlying servers while the application still has limits and costs
Source pin	An exact source revision selected for use or release
Tenant	A separately controlled customer or workspace within a shared service
Typed contract	A structured definition of valid data, inputs and outputs
Webhook	A request sent by one service to notify another that something happened

17 Sources and scope

Reading the references

The P references identify App Composer product contracts, architecture and backlog themes. The A references identify individual application sources. The C references identify official Cloudflare and EmDash documentation. They establish the product direction and underlying technical capabilities used in this completed-vision account. An issue's acceptance criteria describe intended outcomes; the hypothetical premise does not turn the source issue into a present-day release receipt.

Research and qualification work is represented by its useful evaluated outcome. Paused projects, retired services, fictional-data restrictions and explicit commercial holds retain their meaning. Individual projects and integrations have their own support boundaries. Examples in the prose illustrate a completed journey and do not claim a named customer, commercial contract, external endorsement or measured financial return.

Product direction and capabilities

[P01] App Composer product contract
Product direction and boundaries

[P02] App Composer roadmap
Mission and outcome gates

[P03] App Composer architecture
Federated project architecture

[P04] Founder story
Public story | Authored source

[P05] Integrated creator experience
Master journey issue 590

[P06] Independent ownership
Flagship creator proof issue 425

[P07] Get Ready
Readiness engine issue 16 | Onboarding issue 17

[P08] Init and structured application briefs
Init issue 21

[P09] Updates and recovery
Preserve customisation issue 8 | Release and recovery issue 9

[P20] Composer application directory
README and independent apps

[P21] Wider portfolio scope
Portfolio inventory | Current project register

[P22] Workers AI value and adoption
Useful models and costs issue 936

[P23] Idea to ongoing operation benchmark
Six gates and intervention accounting issue 1025

[P24] Teams and platform capabilities
Platform teams issue 216 | Provisioning issue 219 | Usage and quotas issue 220

[P25] Supported agents and scoped work
Bring your own agent issue 311 | Connections issue 291

[P26] Point at the preview and revise
Preview feedback issue 1009

[P27] Brand Studio
Native brand experience issue 353 | Project voice and preference issue 446

[P28] Media Studio product and architecture
Product contract | Source and authority map | README

[P29] Social production and distribution
Owned social factory issue 586 | Publishing issue 604

[P30] Operations and health
Owner attention issue 118 | Critical journeys issue 445

[P31] Creative references
Reference Desk issue 64 | Reference Intelligence issue 171

[P32] Conversational creative direction
Creative Director issue 15

[P33] Local image tools
Image Tools issue 183

[P34] Reusable character identity
Character canon issue 16 | Public character worlds issue 147

[P35] Finished films
Music video issue 14 | Film craft issue 222

[P36] Practical content production
Audio-led films issue 19 | Content planning issue 495

[P37] Speech and lyric timing
Editable timing and captions issue 17

[P38] Motion and advanced creative capabilities
Motion Lab issue 150 | Creative compute issue 151 | Studio capability backlog

[P39] Precise creative review
Agent review issue 487 | Video feedback issue 482

[P40] Composer Share
Creative library and project kits issue 477

[P41] Studio publication and provenance
Public gallery issue 13 | Portable projects

[P42] Launch packaging
Launch Kit guide

[P43] Creative experimentation and learning
Creative experiment loop issue 1016

[P44] Decision assisted discovery
Decision classifier qualification issue 970 | Share discovery issue 1020

[P45] Woodhouse and the Share transition
Woodhouse architecture | Composer successor journey issue 477

[P46] Additional applications and demonstrations
Tower Stack issue 26 | Ask Me Anything issue 27 | Migration Map issue 194 | Oddpull issue 404 | Team Hub issue 354 | Plugin Trust Centre issue 365 | Incident Desk issue 367

[P47] Company Watch
Company Watch issue 1023 | AI Infrastructure Watch scenario 1024

[P48] Website migration
WordPress owner journey issue 1018 | Migration parity issue 167 | Migration research issue 358

[P49] Reusable capabilities and independent blueprints
Typed composition issue 476 | Reusable catalogue issue 600 | Blueprint documentation

[P50] Optional execution integrations
Oyster integration issue 422 | Optional execution issue 597

[P51] Security and authority
Project architecture | Security and abuse boundaries issue 10

[P52] Incident response
Agentic Responder issue 215 | Incident Desk acceptance issue 367

[P53] Release quality
Supported release evidence issue 11

[P54] Benchmarks Studio
Evaluation system issue 601 | Evaluation experience issue 933 | Creative production comparison issue 1019

[P55] Cost and workload decisions
FinOps issue 312 | Allowances and credits issue 959 | Equivalent outcome cost comparison issue 1022

[P56] Commerce integrations
Commerce issue 105 | Shopify issue 293 | Stripe issue 294 | PayPal issue 295 | Printful issue 296

[P57] Research and audience growth
Research discovery issue 999 | Advertising issue 1017 | SEO workbench issue 397

Individual applications and project sources

[A01] Paperwork
Application source | Open Paperwork

[A02] LDraw
Application source | Open LDraw

[A03] Here / there
Guestbook source | Open Here / there

[A04] Linkarooie
Showcase source | Open Linkarooie

[A05] Creator Hub
Composition blueprint | Example guestbook | Example profile

[A06] Link Check
Application source | Open Link Check

[A07] Hello World
Application source | Open Hello World

[A08] Launch Kit
Application source | Open Launch Kit

[A09] Carry / forward
Demonstrator source | Open fictional demonstrator

[A10] LoftwahFM
Product contract | Business audio scope | Open LoftwahFM

[A11] LOFTWAH FIGHTER
Game source | Reusable game direction | Open Fighter

[A12] SHOALSHOT
Game source | Open SHOALSHOT

[A13] Bubbles
Game source | Product vision | Open Bubbles

[A14] Protocol 11
Product contract | Game source | Open Protocol 11

[A15] Foilbury
Product contract | Application source | Open Foilbury

[A16] MAX
Product contract | Application source | MAX project site

[A17] Liferaft
Application source | Agent recovery boundaries | Liferaft project site

[A18] Bogan Hustler
Game source | Gameplay and UX issue 15 | State correctness issue 18

[A19] Pirates
Preserved project source | Lifecycle record

Official platform documentation

[C01] EmDash on Cloudflare
Official framework guide

[C02] WordPress content migration
Official migration guide

[C03] WordPress plugin adaptation
Official porting guide

[C04] EmDash plugin boundaries
Plugin formats | Capabilities and security

[C05] Durable Objects
Official overview

[C06] Workers
Official overview

[C07] Workers AI
Official overview | Pricing reference

[C08] D1
Official overview

[C09] R2
Official overview

[C10] Workflows
Official overview

[C11] AI Gateway
Official overview

[C12] Queues delivery semantics
Delivery guarantees

[C13] Execution limits and suitable compute
Workers limits | Workflows limits | Containers

[C14] EmDash agent access
MCP server reference

[C15] R2 costs
Official pricing reference