Adral

OFFICIAL PLAYBOOK

FROM FIRST IDEA TO A LIVE SITE

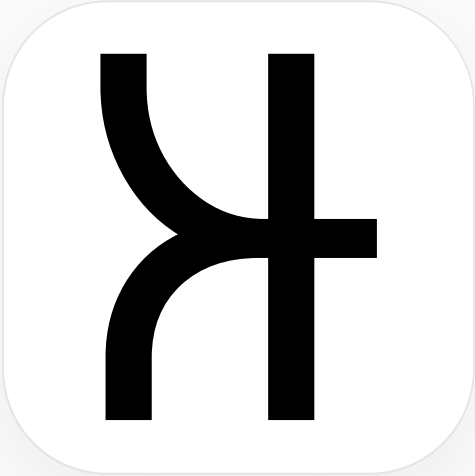
Build a Full-Stack App with Adral

Agent mode researches and tests, the Workspace plans and prototypes, and Adral Code builds and ships. One worked example, six steps, all three tools.

● AGENT MODE ■ WORKSPACE ▲ ADRAL CODE

Adral | 𐑨𐑛𐑮𐑨𐑤 | ஆற்றல்

ār̲r̲al — energy, capability



RELEASE
Edition 1 • September 2026

APPLIES TO
Adral desktop app 1.1.3

WORKED EXAMPLE
Brew Club, a filter-coffee subscription site

adral.ai

Contents

Adral gives you three tools behind one sign-in, and a full-stack app uses all three. **Agent mode** researches and tests, the **Workspace** plans and prototypes, and **Adral Code** builds and ships. This playbook takes one real app from first idea to launch: **Brew Club**, a filter-coffee subscription website with customer accounts, a delivery dashboard and an admin view.

THE TOOLS

Meet the three tools

All three sit in the toolbar at the top of the desktop app, share one sign-in, and run on your Adral plan — there are no model keys to manage. Each is best at a different part of the job.

AGENT TAB

Agent mode

WHAT IT IS

Bots: AI coworkers, each with its own computer and web browser

BEST AT

Research on the live web, and testing a site the way a customer would

HOME TAB

Workspace

WHAT IT IS

Ask in plain words and get a finished doc, sheet, diagram, design, deck or working app

BEST AT

Planning, specs, and quick hosted prototypes

CODE TAB

Adral Code

WHAT IT IS

A coding agent working on a project folder on your computer

BEST AT

The real product, in any stack, with a git history and deployment

THE ONE IDEA TO HOLD ON TO

Don't make one tool do everything. A Bot is poor at writing a codebase, and Adral Code can't browse your competitors' sites. Used for what each does best, the three cover a build end to end.

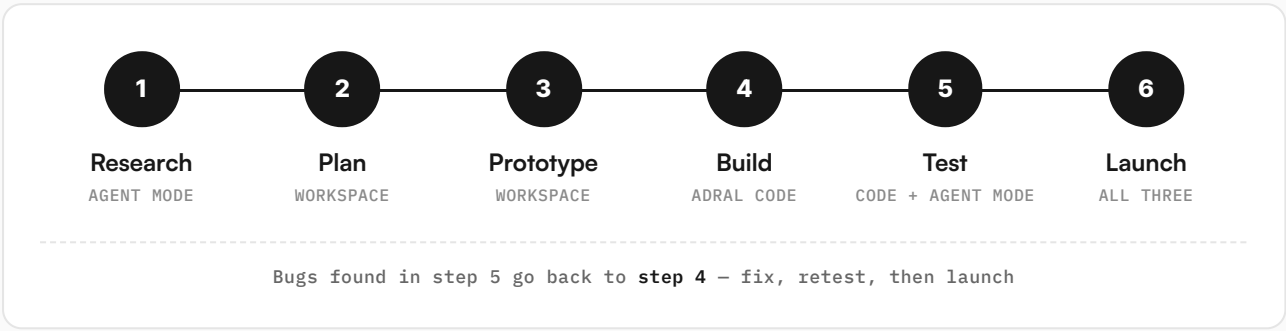
Before you start

- [ ] Install the Adral desktop app and sign in on the **Home** tab. Adral Code shows "Sign in to use Code" until you do.
- [ ] Install Node.js and git. Adral Code installs itself, not your developer tools.
- [ ] Use a paid plan for a build this size. Bots and Adral Code both work in many small steps, and each step uses your plan's allowance.

OVERVIEW

The build at a glance

Six steps take Brew Club from an idea to a live site. The tools hand work to each other, so the output of each step is the input to the next.



STEP	TOOL	WHAT YOU HAVE AT THE END
1 • Research	● AGENT MODE	A sourced brief on competitors' plans and prices
2 • Plan	■ WORKSPACE	A spec, a data-model diagram and a pricing sheet
3 • Prototype	■ WORKSPACE	A clickable version your team has tried and commented on
4 • Build	▲ ADRAL CODE	A working app on your computer, in git
5 • Test	▲ ADRAL CODE ● AGENT MODE	Passing tests, and a QA report from a Bot that used the live site
6 • Launch	ALL THREE	The site live, launch material made, and a Bot keeping watch

WHAT BREW CLUB DOES

VISITORS
See three plans and sign up with an email and password.

CUSTOMERS
Get a dashboard where they can pause, skip or change their deliveries.

ADMIN
A page that lists subscribers and upcoming deliveries.

That is a real full-stack app: screens, a server, a database, accounts and roles.

STEP 01 ● AGENT MODE

01/06

Research the market with a Bot

Before you design anything, find out what customers already pay and get elsewhere. A Bot does this on the live web, in its own browser, and cites every page it used.

---

1. **Make a Bot.** Open the **Agent** tab. Pick a colour and shape, name it **Market Researcher**, and press **Get started**. Or pick the **Researcher** suggestion, which is set up to "dig into a topic and write a clear brief". The Bot opens in its own channel and says hello.

2. **Give it the job, with the output shape spelled out.**

> PROMPT · AGENT MODE
>
> Find three Indian specialty coffee roasters that sell a monthly subscription and compare them in a short table, one line per cell: price per month for 250 g, how often it delivers, and a link to each page.

3. **Watch it work.** Press the screen icon at the top right (**Watch this Agent's screen**) to see its browser and a live **Activity** feed of every step. A **Stop the Agent** button shows while it's working; when that disappears, it's done. A research task like this takes a few minutes.

4. **Read what comes back.** In our run on 22 September 2026 it returned Blue Tokai at ₹500 a month, Kāpikottai at ₹370 and Black Baza at ₹625. It worked each figure out from a 12-delivery plan, explained that working, and linked every roaster's page. Open two of the links yourself before relying on the numbers — prices change, and Bots can misread a page.

WHAT OUR BOT RETURNED	22 SEP 2026
Blue Tokai	₹500 /month
Kāpikottai	₹370 /month
Black Baza	₹625 /month

Each worked out from the roaster's 12-delivery, 250 g plan, with a link to its page.

5. **Ask the follow-ups that shape your product.**

> PROMPT · AGENT MODE
>
> What do customers of these three complain about in reviews? Give me the five most common complaints, with a link for each.

Complaints make better features than guesses. If reviews keep saying "I can't skip a month", skip-a-delivery belongs in your first version.

HAND-OFF

→ STEP 2 Use the copy button on each answer and keep the text — you'll paste it into the Workspace next.

STEP 02 ■ WORKSPACE

02/06

Plan in the Workspace

Turn the research into three planning documents: a spec that says what to build, a diagram of the data, and a sheet that checks the prices make money. The spec becomes Adral Code's instructions in Step 4, so time spent here saves rework later.

HOW THE WORKSPACE WORKS

On the **Home** tab, type what you want in the box and send it. Adral may ask one question first ("What tone should it use?") — answer it and it carries on. The result opens in a workspace beside the chat. Changes arrive as **Pending changes**: look them over, then press **Accept changes** to keep them or **Discard...** to drop them. Ask for all three documents in the same workspace chat so they sit together.

1 **The spec (a Doc).** Paste your Step 1 research at the end of the message.

> **PROMPT · WORKSPACE** DOC
>
> Write a one-page product spec for Brew Club, a filter-coffee subscription website. Sections: who it is for; three plans (Basic 250 g at ₹449, Standard 500 g at ₹799, Premium 1 kg at ₹1,399 a month); user stories for sign-up and login, a customer dashboard (pause, skip a delivery, change plan) and an admin page (subscribers, upcoming deliveries); a data model listing each table and its fields; and what is out of scope for version 1 (payments). Use this research: `[paste]`

2 **The data model (a Diagram).**

> **PROMPT · WORKSPACE** DIAGRAM
>
> Draw a database (ER) diagram for Brew Club from the spec: Customer, Plan, Subscription and Delivery, with their fields and how they relate.

Keep it to four to six tables. Read the arrows: a customer has one subscription, a subscription has many deliveries. If the diagram disagrees with the spec, fix the spec now — it's one message here and a database migration later.

3 **The pricing check (a Sheet).**

> **PROMPT · WORKSPACE** SHEET
>
> Build a spreadsheet for Brew Club's three plans: price, coffee cost per delivery, delivery cost, and margin per plan. Add a 12-month revenue forecast with the starting subscribers, monthly growth and churn in their own yellow input cells.

The cells hold real formulas, so you can change an assumption and watch the forecast move. If a plan loses money, change its price in the spec before anything is built.

HAND-OFF

→ STEP 4 Open the spec, select all, and copy it. You'll give it to Adral Code as a file called `SPEC.md`.

STEP 03 ■ WORKSPACE

03/06

Prototype the customer journey

Before writing real code, build a working prototype in the same workspace and put it in front of your team. Changing a screen here is one message; changing it after Step 4 means reworking code, tests and data.

---

1. **Ask for the whole journey in one message.**

   > PROMPT · WORKSPACE — APP
   >
   > From the spec, build a clickable prototype of Brew Club: a landing page with the three plans, a sign-up form, a customer dashboard where you can pause, skip a delivery or change plan, and an admin page listing subscribers and next week's deliveries. Use sample data stored on the server. Instead of real logins, add a "View as" switch at the top to flip between a customer and the admin. Add a Feedback button that saves notes on the server, and a page that lists them.

   The prototype is a real app with a real server, so data you enter is kept. It runs in the preview beside the chat, with tabs for the app, its **Code** and its **Connections**.

2. **Check it keeps data.** Sign up a test customer, reload the preview, and make sure they are still there. If they vanish, say: "Store customers on the server, not in memory."

3. **Accept, then share.** Press **Accept changes** — people you share with only ever see accepted changes. Then use **Share** to invite your team by email or link with **Use** access, which lets them use the app but not edit it or the chat.

4. **Collect and apply the feedback.** Your team clicks through and leaves notes with the Feedback button. Read them on the feedback page, then change one thing per message: "Move the skip button onto each delivery row, not the top of the page." Accept each change once you've tried it.

5. **Fold the decisions back into the spec.**

   > PROMPT · WORKSPACE
   >
   > Update the spec doc with everything we changed in the prototype: screens, fields and wording.

   Copy the updated spec again — that version is what Adral Code builds from.

WHY THE PROTOTYPE ISN'T THE PRODUCT

A Workspace app can't hold real per-customer logins, can't install extra code libraries, and its public published copy can't keep what visitors type. Those are exactly the things Step 4 adds.

STEP 04 ▲ ADRAL CODE

04/06

Build the real app with Adral Code

Adral Code builds Brew Club as an ordinary project on your computer: real logins, a real database, any library you need, and a git history. It edits files and runs commands directly, so git is your undo button — commit after every step that works.

---

1. **Open a project folder.** Go to the **Code** tab, then `File → Open Folder in Code…` (Cmd+Shift+O) and create a new folder called `brew-club`. The first time you use Documents, Desktop or an external drive, macOS asks for permission — allow it in `System Settings → Privacy & Security → Files and Folders`.

2. **Give it the spec, and make it plan before it builds.**

> **PROMPT · ADRAL CODE**
>
> Create SPEC.md with the text below. Read it, then reply with the pages, server routes and database tables you plan to build. Don't write any code until I say go. `[paste the spec from Step 3]`

Read the plan against your prototype. Fix disagreements now, in words.

3. **Set up the project, with the stack named.** If you don't name a stack, the agent picks one.

> **PROMPT · ADRAL CODE**
>
> Go. Set up a Next.js app with TypeScript and Tailwind, Prisma with SQLite for the database, and email-and-password login with hashed passwords. Initialise git, add a README with the commands to run it, and commit.

4. **Build it in layers — one message each, a commit after each.**

DATABASE LAYER 1	Write the Prisma schema from the data model in SPEC.md, plus a seed script with the three plans and five sample customers. Run the migration and the seed, then commit.
ACCOUNTS LAYER 2	Add sign-up, log-in and log-out. Only logged-in customers can open /dashboard, and only admins can open /admin. Commit.
SERVER LOGIC LAYER 3	Add server actions to pause, skip the next delivery and change plan, each checking the customer owns the subscription. Commit.
SCREENS LAYER 4	Build the landing page, sign-up, dashboard and admin pages to match the prototype: `[describe it, or paste its layout notes]`. Commit.

Building in layers keeps each step small enough to check, and the security rule — "each checking the customer owns the subscription" — sits in the same message as the feature it protects.

5. **Run it.** Ask it to "install everything and start the dev server", then open `http://localhost:3000` in your own browser. The Code tab has no preview pane, but the agent can screenshot local pages to check its own work.

6. **Leave payments for later.** Taking money is a project of its own. Get the core working and tested first, then add your payment provider in test mode, as its own set of layers.

STEP 05 ▲ ADRAL CODE ● AGENT MODE

05/06

Test it

Testing happens twice: Adral Code tests the code on your computer, then a Bot tests the running site the way a customer would. They catch different bugs, so do both.

---

1. **Code-level tests, with Adral Code.**

> PROMPT · ADRAL CODE
>
> Add tests: for the server actions (a customer can't skip or pause someone else's subscription), and one end-to-end test that signs up, skips a delivery and checks the dashboard. Run them all and fix anything that fails. From now on, run the tests after every change.

The ownership test matters most: it's the one that stops one customer changing another's deliveries.

2. **Put a staging copy online.** A Bot works on its own computer, so it can't open `localhost` on yours. It needs a real web address.

   a. **Move to a hosted database first.** Most hosts can't keep a SQLite file between deploys. Create a free Postgres database with a hosted provider, then say: "Switch the database from SQLite to Postgres, read the connection string from DATABASE_URL, and create a fresh migration."

   b. **Log in to your host yourself.** Open Terminal and sign in to your host's command-line tool, such as Vercel, Render or Fly.io. Your passwords never go through the agent.

   c. **Deploy.** "Deploy a staging copy with the Vercel command-line tool and give me the address." Put the database connection string in the host's settings page, not in the project folder.

3. **Customer-level testing, with a QA Bot.** In the **Agent** tab, make a Bot called **QA Tester** and give it a script:

> PROMPT · AGENT MODE — QA TESTER
>
> Test the website at `[staging address]` as a brand-new customer. Sign up with a new test email, choose the Standard plan, skip the next delivery, pause the subscription, log out, log back in, and check the skip and the pause are still there. Then try to open /admin — it should refuse you. Report each step as pass or fail, with exactly what you saw when something failed.

Watch its screen as it goes: you'll see your site through a stranger's eyes, including the confusing bits no test catches.

4. **Feed the failures back to Adral Code, test first.**

> PROMPT · ADRAL CODE
>
> The QA test found this: `[paste the failed step]`. First write a test that fails because of it. Then fix it, run all the tests, and commit.

Writing the failing test first proves the fix works, and keeps the bug from quietly coming back. Redeploy staging and rerun the Bot's script until every step passes.

STEP 06 ALL THREE

06/06

Launch and run it

Launch uses all three tools at once: Adral Code ships the site, the Workspace makes the launch material, and a Bot keeps an eye on things afterwards.

---

1. **Ship with Adral Code.** Give production its own database — never share one with staging — and put its connection string in the host's settings. Then ask for a pre-flight check before the real deploy:

> PROMPT · ADRAL CODE
>
> Before we deploy to production, list everything that could break there — settings, database migrations, the first admin account — and check each one. Then deploy to production and give me the address.

Make your own account the admin with a one-off script ("make me an admin with a script I run once"). Don't add a hidden admin sign-up page.

2. **Make the launch material in the Workspace.** Same Home tab, same way of asking:

> A POSTER AND SOCIAL POSTS — DESIGN
>
> Design an Instagram post and an A4 poster for Brew Club's launch: warm colours, the three plans and prices, and a QR placeholder.

> A LAUNCH DECK — SLIDES
>
> Make a five-slide deck on Brew Club for our partners: the problem, the plans, how it works, launch dates, and the ask.

> CUSTOMER FAQ — DOC
>
> Write an FAQ for Brew Club customers covering plans, pausing and skipping, delivery areas, and cancelling.

Designs export as PNG at the exact size each platform wants, or as PDF for print.

3. **Keep a Bot on watch.** The **Night Shift** Bot suggestion is made for work while you're away — it "works overnight and preps your morning digest". Give it a standing brief:

> PROMPT · AGENT MODE — NIGHT SHIFT
>
> Every morning, open `[live address]`, sign up with a new test email, and confirm the dashboard loads. Then check the three roasters from our research for price changes. Send me a short digest: what passed, what failed, and what changed.

Read its first few digests closely to be sure it's doing what you asked. Ask Adral Code to leave the Bot's test emails out of the admin totals, so they don't inflate your numbers.

GO ROUND THE LOOP FOR EVERY NEW FEATURE

Research it with a Bot, add it to the spec in the Workspace, build it in layers with Adral Code, then have the QA Bot walk it on staging before it ships. Brew Club's next feature, payments, follows the same six steps.

REFERENCE

Prompt cheat sheet

One rule covers all three tools: say what you want, what it should contain, and what "done" looks like. The habits below are that rule applied to each tool.

TOOL	HABIT	SAY IT LIKE THIS
● AGENT MODE	Name the output's shape	"...in a short table, one line per cell: price, frequency, link."
● AGENT MODE	Ask for proof	"Give a link for every number."
● AGENT MODE	Test as a numbered script	"Sign up, skip, log out, log in, check the skip stuck. Report pass or fail per step."
■ WORKSPACE	Name every screen and every piece of data	"Screens: landing, sign-up, dashboard, admin. Store customers on the server."
■ WORKSPACE	One change at a time	"Move the skip button onto each delivery row. Change nothing else."
▲ ADRAL CODE	Plan before code	"Reply with your plan. Don't write code until I say go."
▲ ADRAL CODE	Name the stack	"Next.js, TypeScript, Prisma, Postgres."
▲ ADRAL CODE	Close every step	"Run all the tests, then commit."
ANY	Report bugs precisely	"I pressed Save with an empty email. Nothing happened. Expected an error under the field. Fix only that."

When a build goes wrong

SYMPTOM	TOOL	FIX
It looks stuck	■ WORKSPACE	Scroll up — it's usually waiting for an answer to a question
A change broke things	■ WORKSPACE	**Discard...** the pending changes, or undo that turn, then ask again more precisely
"The operation was aborted"	■ WORKSPACE	The run was cut off, not your app. Press **Retry**, or split the request
Data vanishes on reload	■ WORKSPACE	"Store it on the server, not in memory"
A change broke things	▲ ADRAL CODE	"Revert the last commit" — then ask again in smaller steps
`node: command not found`	▲ ADRAL CODE	Run `which node` in Terminal and give the agent the full path
Going in circles after three tries	■ WORKSPACE ▲ CODE	Start a new chat with a short summary: what works, what doesn't, the one thing to change
Lands on the wrong pages	● AGENT MODE	Give it the exact address to start from
Taking too long	● AGENT MODE	Press **Stop the Agent** and give it a narrower task

CONTINUED

SYMPTOM	TOOL	FIX
A number looks wrong	AGENT MODE	Ask for the link behind it and check the page yourself

THE MOST IMPORTANT HABIT

The most important habit is the plainest: **undo, don't patch.** Asking a tool to fix its own broken fix tends to pile mistakes on top of each other. Go back to the last good state and ask again, more precisely.

REFERENCE

Limits to know

Each tool has limits, and most exist for a good reason — the Workspace is sealed off so it's safe to build on your real accounts. Every limit below has a workaround, and it's often the next tool along.

TOOL	LIMIT	WORKAROUND
WORKSPACE	Apps can't add outside code libraries	Ask for charts drawn in SVG and icons as emoji. Need a library? Build it in Adral Code
WORKSPACE	App code can't call the internet directly	Add a Connection — Google Sheets, Gmail, Slack, Notion, GitHub or any MCP server — and approve it when asked
WORKSPACE	Apps can't hold per-person logins	Control who gets in by sharing with **Use** or **Build** access. Real customer accounts belong in Adral Code
WORKSPACE	A published public copy doesn't keep what visitors type	Share the app instead, or build public forms in Adral Code
WORKSPACE	Export is PDF, PNG or MP4 — no Word or Excel	Copy the text out, or have the app write rows to a connected Google Sheet
WORKSPACE	A new scheduled job starts switched off	Turn it on under the app's **Connections → Hooks**
WORKSPACE	Unsupported styling is silently ignored	Ask it to use standard classes or inline styles for that element
AGENT MODE	A Bot can't open `localhost` on your computer	Put a staging copy online (Step 5) and give it that address
AGENT MODE	Some sites block automated browsers	Point it at another source, or at the page's direct address
AGENT MODE	It can misread a page	Ask for the link behind every number, and spot-check them
AGENT MODE	It browses with its own logins	Test with made-up test accounts, never your personal ones

CONTINUED

TOOL	LIMIT	WORKAROUND
▲ ADRAL CODE	Doesn't install Node.js or git	Install them yourself before Step 4
▲ ADRAL CODE	No preview pane	Open `http://localhost:3000` in your browser; the agent can screenshot pages to check itself
▲ ADRAL CODE	No built-in deploy	Sign in to your host's command-line tool in Terminal, then ask the agent to use it
▲ ADRAL CODE	Can read and change everything in the project folder	Keep passwords and production secrets out of the folder, and commit often so every change can be undone
ALL THREE	Every step uses your plan's allowance	Plan first, ask precisely, and build in small layers — fewer wasted steps

WHERE TO GO NEXT

Once Brew Club is live, go back to Step 1 for its next feature. Each time round, the loop gets faster: the research is already in your Bots' channels, the spec already exists, and the codebase already has tests.

Adral

𐰀𐰇𐰼𐰃 · ஆற்றல்

ADRAL.AI

This playbook describes the Adral desktop app, version 1.1.3, as of September 2026. Features, limits and plans can change; adral.ai has the current details.

Brew Club is a fictional example. The roaster prices in Step 1 are what a Bot found on 22 September 2026, and those names belong to their owners.

Edition 1. Set in Satoshi and IBM Plex Mono.