GOOGLE AI ULTRA BUILDER SERIES • LATE 2026 EDITION

Mastering Antigravity & Jules

The complete visual handbook to Google's agent-first software studio—covering Antigravity 2.0, the IDE, CLI, Subagent Swarms, Customizations, and the Jules Async Workflow.

- Antigravity 2.0 Desktop App
- Antigravity IDE (Tab & Cmd+I)
- Jules Async GitHub Workflows
- Subagents & Parallel Swarms
- Rules, Skills, Plugins & MCP
- CLI (`agy`) & Python SDK



Four Unified Surfaces
Work seamlessly across the Desktop App, VS Code-based IDE, Terminal CLI (`agy`), and Python SDK.

Jules DNA Built Right In
Includes the complete late-2026 Jules-to-Antigravity migration map for async branches, PRs, and cron runs.

Powered by AI Ultra
Unlock 5x–20x frontier Gemini quotas so multi-agent swarms and overnight `/goal` runs never stall.

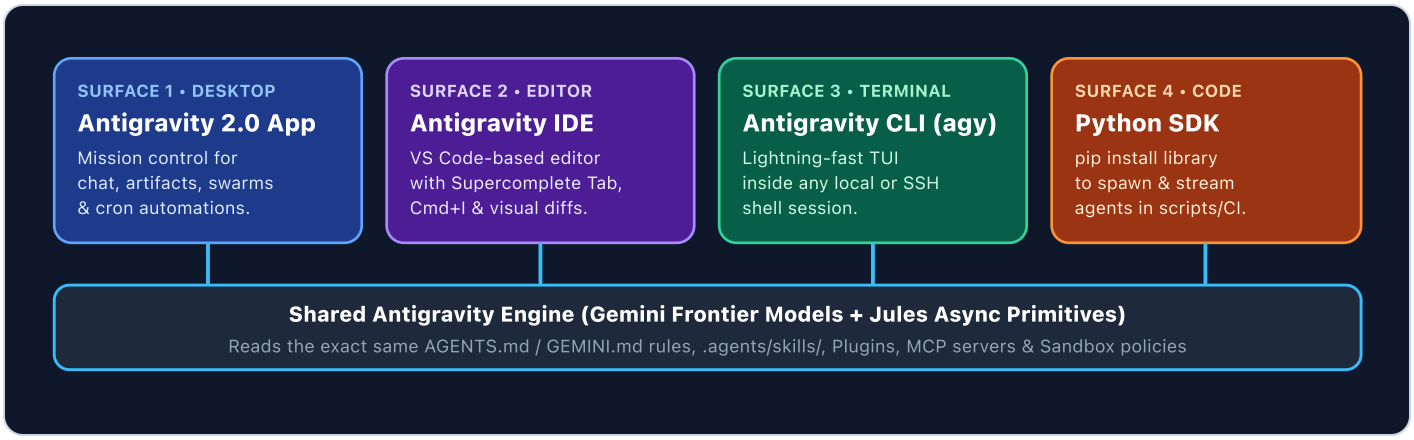
The Big Picture: Antigravity + Jules in One Studio

Google Antigravity (`antigravity.google`) is Google's flagship agent-first software creation platform—designed not just to autocomplete lines of code, but to plan, build, test, browse, and ship entire projects alongside you.

Quick Navigation

1. **Antigravity + Jules Overview** p. 2
2. **Setup, Surfaces & AI Ultra Perks** p. 3
3. **Antigravity 2.0 App UI Tour** p. 4
4. **Antigravity IDE** (Tab, Cmd+I, Agent) p. 5
5. **Slash Commands & `@` Context** p. 6
6. **End-to-End Feature Build Walkthrough** p. 7
7. **Jules Guide & 2026 Consolidation** p. 8
8. **Subagents, Worktrees & Swarms** p. 9
9. **Customizations** (Rules, Skills, MCP) p. 10
10. **CLI (`agy`), Python SDK & Browser** p. 11
11. **Security Sandbox & Cheat Sheet** p. 12

Visualization: Four Ways to Drive One Shared Agent Brain



Important Late-2026 Update: Jules Is Now Folding Into Antigravity!

Google's standalone asynchronous GitHub agent, **Jules** (`jules.google`), pioneered background cloud-VM bug fixing and automated Pull Requests. In late 2026 (read-only **Nov 11, 2026**; full sunset **Dec 31, 2026**), Google consolidated Jules's best async capabilities—branched background subagents, scheduled cron tasks, `AGENTS.md` memory, and GitHub PR automation—directly into **Google Antigravity**. Page 8 covers both Jules's workflow and your exact migration map!

First-Time Setup & Your Google AI Ultra Advantage

You don't need to be a senior software engineer to use Antigravity. Non-coders use **Antigravity 2.0** to generate guides, websites, and automations, while engineers pair it with the **IDE** and **CLI**.

3-Step Setup (Under 3 Minutes)

1 Install Your Surface

Download **Antigravity 2.0** or **Antigravity IDE** from `antigravity.google`, or install the terminal CLI (`agy`) and Python SDK.

2 Sign In w/ AI Ultra

Authenticate with the Google account tied to your **Google AI Ultra** plan to unlock frontier models and 5x–20x rate limits automatically.

3 Open or Import Config

Select a workspace folder. Coming from Cursor, Claude Code, or Codex? Ask the agent: *"Migrate my agent settings into a plugin."*

Which Antigravity Surface Should You Open Today?

Surface	Best For	Everyday Superpower
Antigravity 2.0 Desktop App	Creators, PMs, researchers & multi-agent orchestrators	Clean workspace without code clutter. Delegate research, build apps from scratch, inspect rich Markdown/HTML **Artifacts**, and schedule daily cron jobs.
Antigravity IDE Standalone Editor	Hands-on coding, refactoring & debugging	Full VS Code compatibility plus single-keystroke **Tab Supercomplete**, `Cmd+I` inline edits, inline Code Lenses, and red/green visual diff review.
Antigravity CLI `agy` in Terminal	Terminal lovers, SSH servers & quick repo tasks	Type `agy` inside any folder to launch an interactive terminal agent with full slash commands and subagent support (`~/.gemini/antigravity-cli/settings.json`).
Python SDK `google-antigravity`	Custom automation pipelines & CI test suites	Spawn programmable `Agent()` instances inside Python scripts, stream live thought/tool events, and wire custom local tools.

What Your Google AI Ultra Plan Unlocks Here

1. Unthrottled Multi-Agent & Swarm Runs

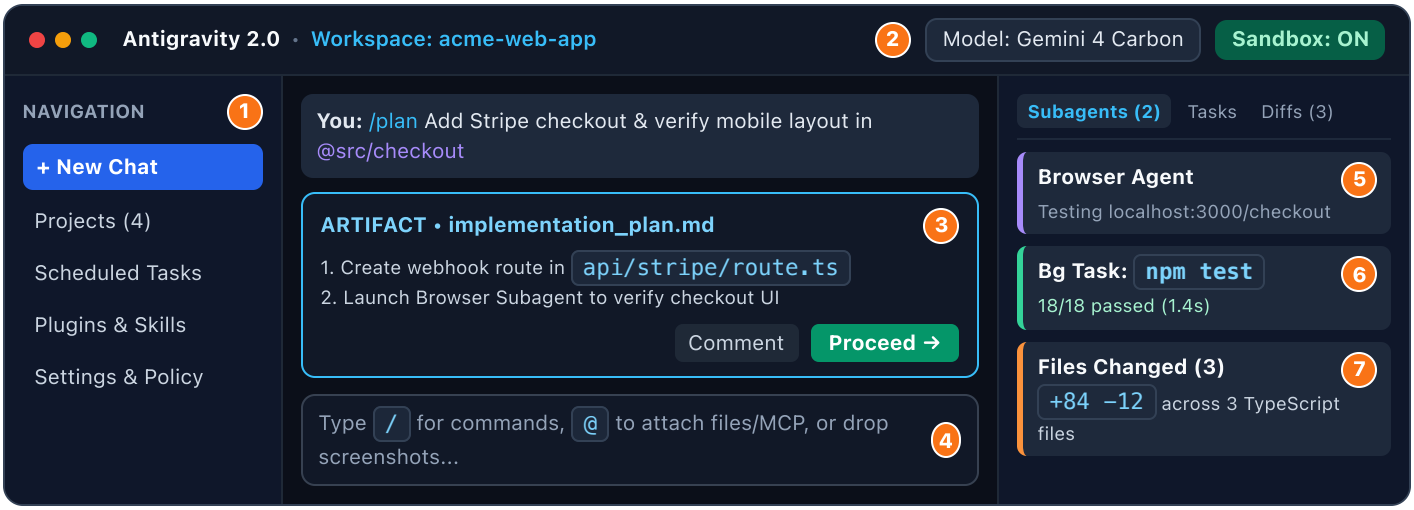
Features like `/teamwork` (parallel agent swarms), `/boost` (multi-perspective deep verification), and `/goal` (long overnight runs) consume heavy tokens. AI Ultra's **5x–20x quota multiplier** keeps them humming without hitting mid-task caps.

2. Frontier Models + Cloud Developer Credits

Ultra gives you priority access to flagship reasoning models (Gemini Pro / Carbon / Deep Think) inside the Model Selector, plus **Google Developer Program Premium** monthly Google Cloud & Gemini API credits for deploying what you build.

Antigravity 2.0 Desktop App: Annotated Visual Tour

Antigravity 2.0 separates high-level thinking from raw code files. You converse in the center **Chat Canvas** while live subagents, terminals, artifacts, and file diffs update in the right-hand **Auxiliary Pane**.



Numbered Interface Reference

1 **Left Sidebar:** Switch workspaces, start fresh chats, manage cron *Scheduled Tasks*, toggle *Plugins & Skills*, or tune security rules.

2 **Model & Sandbox Bar:** Pick Gemini Flash (speed), Pro/Carbon (deep coding), and inspect active terminal sandbox status.

3 **Interactive Artifacts:** Plans, reports, Mermaid diagrams, and diffs open as rich documents—click **Proceed** to approve execution.

4 **Smart Composer:** Trigger workflows with `/`, pull exact context with `@`, or drag UI mockups and PDFs straight into chat.

5 **Subagents & Swarm Monitor:** Watch parallel research, coding, or live Webview Browser agents work without cluttering main chat.

6 **Background Tasks & Terminals:** Dev servers and long test suites run asynchronously and automatically wake the agent when done.

7 **Files Changed & Sidecars:** Inspect every added/deleted line across your repo, or open plugin UI panels (like Android Device Hub or Compose Preview).

Antigravity IDE: Three Speeds of AI Pair Programming

Built on VS Code so all your favorite extensions and themes carry over, **Antigravity IDE** gives you three distinct gears depending on whether you are editing one line, one function, or twenty files.

Gear 1: Passive (`Tab`)

Millisecond Flow State. Predicts your next edit, multi-line diff (*Supercomplete*), missing import, or next cursor jump.

```
Press Tab to accept
Press Cmd+→ word-by-word
Press Tab again to jump!
```

Gear 2: Inline (`Cmd+I`)

Surgical Single-Block Edit. Highlight a function, press `Cmd+I` (`Ctrl+I`), and ask for a refactor, docstring, or type fix.

```
Edits only highlighted lines.
Shows red/green inline diff.
Zero risk to rest of file.
```

Gear 3: Sidebar Agent

Multi-File Autonomous Builder. Reads codebase, edits across folders, runs tests, checks browser, and fixes errors.

```
Toggle Planning Mode first
for complex architecture
before touching code.
```

Essential Antigravity IDE Features & Shortcuts

Feature / Action	Shortcut (macOS / Win)	How to Use It Effectively
Supercomplete & Tab-to-Jump	`Tab` / `Esc`	After renaming a prop or parameter, wait 0.2s—Antigravity highlights the next callsite and lets you teleport there with `Tab`.
Tab-to-Import	`Tab`	Type a component or utility name from another file; pressing `Tab` automatically inserts the clean top-of-file `import` statement.
Inline Command	`Cmd+I` / `Ctrl+I`	Place cursor on an empty line to generate a new helper from scratch, or select messy code and type *"Extract into pure async helper."*
Inline Code Lenses	Click lens above symbol	One-click buttons floating above classes and functions: *Explain*, *Refactor*, or *Write Unit Tests* scoped to that exact symbol.
Diagnostic Auto-Fix	Hover red squiggle / Problems	Click *Fix with Antigravity* on any TypeScript, Python, Rust, or linter error to patch root causes using surrounding context.

Pro Workflow: Run Antigravity IDE + Antigravity 2.0 Side by Side

Both apps share the exact same workspace and `.agents/` rules! Many Ultra users keep **Antigravity 2.0** running long research or multi-agent swarms on a second monitor while making fast `Tab` and `Cmd+I` tweaks inside **Antigravity IDE**.

Steering Agents with / Slash Commands & @ Mentions

Typing a single slash command at the start of your prompt shifts Antigravity into specialized operating modes—from interactive design interviews to unstoppable overnight execution.

Complete Built-In Slash Command Cheat Sheet

Command	What Mode It Activates	When to Reach for It
`/plan`	Step-by-Step Blueprint Mode	Before any multi-file feature or refactor. Writes a reviewable plan artifact before editing code.
`/grill-me`	Interactive Design Interview	When your idea is still fuzzy—the agent interviews you on edge cases and trade-offs until aligned.
`/goal`	Relentless Autonomous Mode	Long or overnight tasks (e.g., *"Get test coverage to 90% and fix all build errors"*)—won't stop early.
`/boost`	Deep Multi-Perspective Mode	Hard architecture bugs or research requiring deep reasoning, competing hypotheses, and strict proof.
`/teamwork`	Multi-Agent Swarm Mode	Large projects where independent slices (frontend, API, tests, review) can be built in parallel.
`/browser`	Live Webview Browser Agent	Testing `localhost` web apps, clicking through UI flows, or interacting with live JS-rendered sites.
`/schedule`	Recurring Cron Automation	Setting up daily/weekly background runs (also accessible via `/automation`), like morning PR digests.
`/learn`	Persistent Memory Capture	Right after you correct a mistake or solve a tricky setup—saves the lesson as a reusable rule/skill!
`/compact`	Context Window Optimizer	Summarizes a very long chat history so your session stays sharp, fast, and uncluttered.

Precision Context with @ Mentions

@ Files & Folders

Attach `@package.json` or `@src/auth/` so the agent reads exact targets immediately without guessing.

@ Terminals & Chats

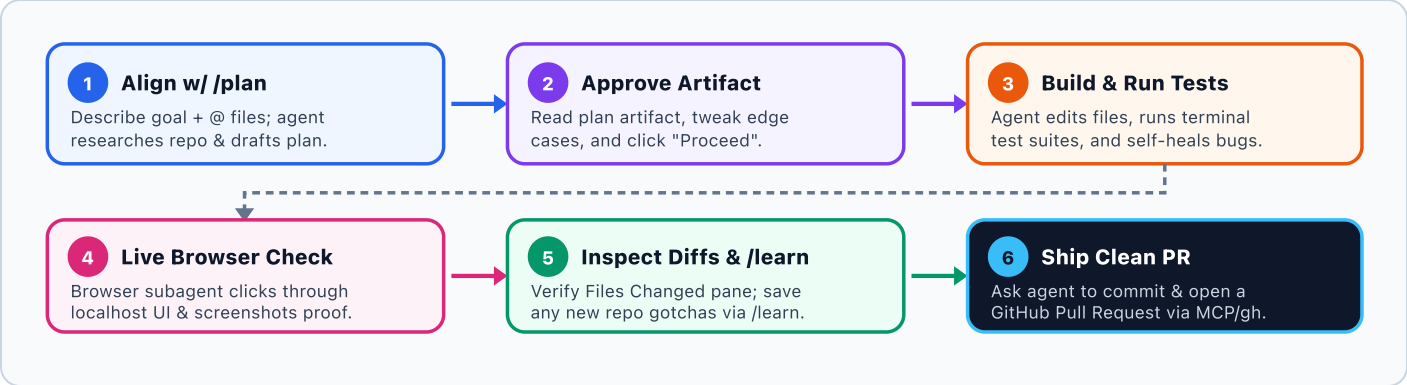
Reference a failing build with `@terminal` or carry decisions forward from `@previous-conversation`.

@ Rules & MCP Tools

Explicitly invoke a style guide via `@rule` or query connected databases/GitHub via `@mcp`.

The 6-Step Agentic Build Loop (Idea to Verified PR)

Whether you are building a personal dashboard from scratch or adding a feature to an existing repo, this 6-step loop keeps you in full control while the agent handles the heavy lifting.



Concrete Example: Building a Personal Expense Tracker Web App

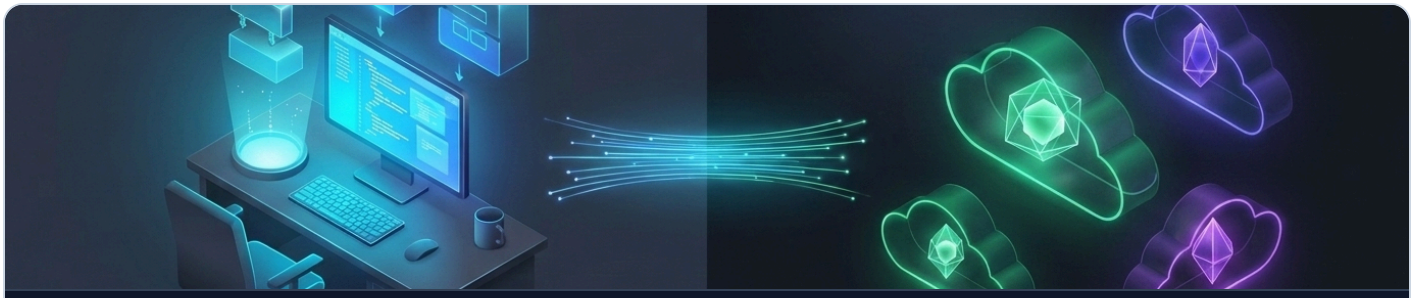
Turn	Exact Prompt You Type	What Antigravity Does Behind the Scenes
Turn 1: Plan	`/plan Build a clean Next.js + SQLite expense tracker with CSV import and monthly charts.`	Creates a task checklist & architecture plan artifact with tech choices and schema; pauses for feedback.
Turn 2: Build	Click **Proceed** on the plan artifact (or reply: *"Add dark mode toggle too, then proceed."*)	Scaffolds project, writes database & chart components, runs `npm run build` and unit tests until green.
Turn 3: Verify UI	`Start the dev server and open localhost:3000 in the browser to test uploading a sample CSV.`	Starts background server, delegates to the **Browser Subagent**, uploads a test CSV, and verifies charts render.
Turn 4: Ship	`Initialize git, write a clean commit, and push to a new private GitHub repo via GitHub MCP.`	Commits tested code and creates your remote repository + clean README automatically.

Zero-Polling Reactive Architecture

When Antigravity starts a long test suite, dev server, or subagent, you **never** need to babysit or ask "is it done yet?" Background tasks automatically wake the agent the instant they complete.

Google Jules: Async Pioneer & Antigravity Migration

Google Jules (`jules.google`) introduced "fire-and-forget" asynchronous engineering—cloning GitHub repos into cloud Linux VMs, proposing a plan, running tests, and opening Pull Requests while you worked elsewhere.



[LEFT: INTERACTIVE MODE] Live pair-programming in IDE/App

—— Unified in Antigravity (Late 2026) →

[RIGHT: JULES ASYNC MODE] Parallel branched workers & overnight cron PRs

The Late-2026 Timeline & 1-to-1 Jules → Antigravity Map

Rather than splitting your quota between two separate coding products, Google folded Jules's async architecture into Antigravity (Jules entered read-only mode on **Nov 11, 2026** ahead of full sunset on **Dec 31, 2026**). Every beloved Jules workflow now has a direct—and more powerful—home inside Antigravity:

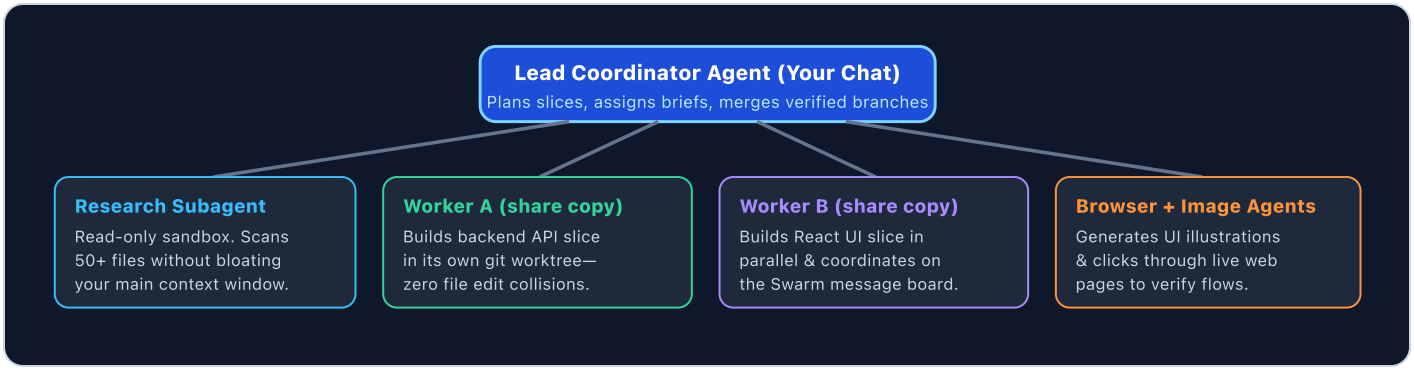
Classic Jules Feature (`jules.google`)	How You Do It Better in Google Antigravity Today
"Give Me a Plan" Approval Gate	Use `/plan` (or IDE Planning Mode). Antigravity writes a structured Markdown plan Artifact with an interactive **Proceed** button.
60 Parallel Cloud Tasks (AI Ultra) `jules remote new --parallel 3`	Use `/teamwork` (Swarm) or spawn subagents with `workspace: "share"` / `"branch"` so each agent edits its own isolated git copy in parallel.
Repo `AGENTS.md` Memory File	**100% Native!** Antigravity automatically discovers and obeys your existing root and subdirectory `AGENTS.md` (and `GEMINI.md`) files with zero changes.
GitHub Issue / PR Automation	Connect the built-in **GitHub MCP Server** (or `gh` CLI skill) + `/goal` to pull an issue, fix it on a branch, run tests, and open a ready-to-merge PR.
Scheduled Scans & Audio Recaps	Use `/schedule` (or `/automation`) to run nightly dependency/security audits or morning changelog digests automatically.

Used Jules Previously? Export Your History Before Dec 31, 2026

Run `npx @google/jules` in your terminal and authenticate via `jules login` to export past session patches, setup scripts, and repository memories straight into your local Antigravity `.agents/` folder before standalone `jules.google` retires.

Multiplying Yourself: Subagents & Native Swarms

With a **Google AI Ultra** plan, you aren't limited to a single single-threaded assistant. Antigravity can dispatch specialized specialist agents—or an entire collaborative team—that work simultaneously.



Subagents vs. Swarms: When to Use Each

A Focused Subagents (Automatic)

Antigravity automatically delegates focused jobs—like surveying a huge repo (`research`), clicking around a web app (`browser`), or generating assets (`image-generator`)—and receives a concise summary back.

B Native Swarms (`/teamwork`)

For ambitious builds or cross-checking critical code, type `/teamwork`. Multiple named agents (`author-1`, `author-2`, `verifier`) collaborate on a shared public message board!

Understanding Workspace Isolation Modes

Workspace Mode	How It Works & Why It Prevents File Conflicts
`inherit` (Default)	Works directly inside your active folder. Ideal for read-only researchers, code reviewers, or single-author tasks.
`share` (Worktree)	Creates an instant lightweight working copy sharing `.git` storage—letting 3+ authors edit and build simultaneously without overwriting each other!
`branch` (Clone)	Creates a completely isolated clone for risky refactors or competing architectural experiments (just like Jules's parallel `--parallel 3` runs).

Customizing Antigravity: Rules, Skills, Plugins & MCP

Out of the box, Antigravity is a brilliant generalist. With five modular customization building blocks, you turn it into a specialist tailored to your exact voice, codebase, and third-party apps.



1. Rules (`GEMINI.md` / `AGENTS.md`): Always-on guardrails, tone & repo conventions.

2. Skills & Plugins (`SKILL.md`): On-demand playbooks & 1-click Marketplace bundles.

3. MCP & Hooks: Live connections to GitHub, Drive, Figma, Databases & scripts.

The 5 Customization Building Blocks Compared

Type	Where It Lives	What It Does
Rules	`AGENTS.md`, `GEMINI.md`, or `.agents/rules/*.md`	Enforces coding standards, tech stack choices, or response style. Loaded hierarchically from edited subfolder up to repo root.
Skills	`.agents/skills/<name>/SKILL.md` (or `~/.gemini/config/skills/`)	Step-by-step SOPs (like PDF creation, security audits, or TDD). Uses **progressive disclosure**: full text loads *only* when relevant!
Plugins	Marketplace or `~/.gemini/config/plugins/<name>/`	Packages skills, rules, subagents, and MCP servers into one toggleable unit (e.g., Google Docs/Drive/Sheets, Vercel, Figma, Android).
MCP Servers	`mcp_config.json` (or inside a Plugin)	Model Context Protocol bridges giving the agent live tools for GitHub PRs, SQLite/Postgres, Chrome DevTools, Linear, or Canva.
Hooks	`hooks.json`	Shell scripts triggered automatically at lifecycle events (e.g., auto-formatting or secret scanning before/after tool calls).

Team vs. Personal Scope (Priority Order)

Put team-shared rules and skills in your repo's `.agents/` folder and commit them to Git—**Workspace** configs always take top priority! Put personal preferences (like your preferred writing voice or global plugins) in `~/.gemini/config/` so they follow you across every project on your Mac.

Headless Power: `agy` CLI, Python SDK & Cron Jobs

When you want agentic power directly inside your macOS terminal, automated Python scripts, or recurring morning briefings, reach for these three headless workflows.

1 Antigravity CLI (`agy`)

Open Terminal in any project folder and run `agy`:

```
$ cd ~/my-project && agy
> /help         # list TUI slash commands
> /plan Fix flaky auth test
> /exit         # or Ctrl+D twice
```

Settings live in `~/.gemini/antigravity-cli/settings.json`.

2 Scheduled Automations (`/automation`)

Run standing jobs on a cron schedule—even while away from your desk:

- Type `/automation` (or `/schedule`) in Antigravity 2.0.
- Pick cadence (Hourly, Daily, Weekly, or custom cron).
- Antigravity writes a background sidecar config that wakes a fresh agent session on schedule (e.g., *"Every weekday at 8:30 AM, check open GitHub PRs and Google Calendar and write a priority brief"*).

3. Programmatic Orchestration with the Official Python SDK

Install via `pip install google-antigravity` (includes the compiled local runtime binary) to embed full Antigravity agents inside your own Python pipelines, CI checks, or data workflows:

```python
import asyncio, sys
from google.antigravity import Agent, LocalAgentConfig, CapabilitiesConfig

async def run_audit():
    # Default is safe read-only mode; pass CapabilitiesConfig() to enable file edits/commands
    config = LocalAgentConfig(
        system_instructions="You are a concise release-readiness auditor.",
        capabilities=CapabilitiesConfig(),
    )
    async with Agent(config) as agent:
        response = await agent.chat("Run pytest and summarize any failing edge cases.")
        async for call in response.tool_calls:  # inspect live tool calls!
            print(f"[Tool] {call.name}")
        async for token in response:
            sys.stdout.write(token)

asyncio.run(run_audit())
```

Safety Controls, Troubleshooting & Quick Cheat Sheet

Autonomous agents are only fun when you trust their boundaries. Antigravity gives you granular global and per-project guardrails under **Settings → Permissions & Sandbox**.

The 4 Permission Dials You Control

1 Tool Execution Policy

Choose `proceed-in-sandbox` (recommended sweet spot: fast autonomy inside an OS sandbox), `request-review` (ask before shell commands), or `strict`.

2 Workspace Boundary Policy

Set **Non-Workspace File Access** to `ask` or `deny` so an agent working on one project never touches unrelated folders without permission.

3 Command Allowlist / Denylist

Pre-approve safe daily commands (`npm test`, `git status`) while hard-blocking destructive ones (`rm -rf`, production deploy scripts).

4 Browser & Network Allowlist

Restrict which external domains the Browser Subagent can visit; sensitive login and payment pages always pause and hand control back to you.

Quick Troubleshooting Fixes

Common Situation	Fastest Fix
Agent repeats a mistake you already fixed	Type `/learn` or add a 1-line rule to your project's `AGENTS.md` so it sticks forever.
Long conversation starts feeling slow or fuzzy	Run `/compact` to distill context, or start a fresh chat referencing `@previous-conversation`.
Need Google Docs, Figma, or Linear integration	Ask: *"Search the Marketplace for Figma / Google Docs"* and click **Add** on the plugin card.
Want Jules-style isolated parallel attempts	Ask Antigravity to *"spawn 2 branched subagents comparing Approach A vs. Approach B."*

Power-User Pre-Flight Checklist: 6 Habits for Effortless Sessions

- ☐ **1. Drop an** `AGENTS.md` in repo root with build/test commands.
- ☐ **2. Start multi-file work with** `/plan` or `/grill-me`.
- ☐ **3. Tag exact targets with** `@file` or `@folder`.
- ☐ **4. Enable** `proceed-in-sandbox` for safe, interruption-free speed.
- ☐ **5. Verify web UI changes** with the live `/browser` agent.
- ☐ **6. Use** `/goal` **or** `/teamwork` on AI Ultra for big builds!