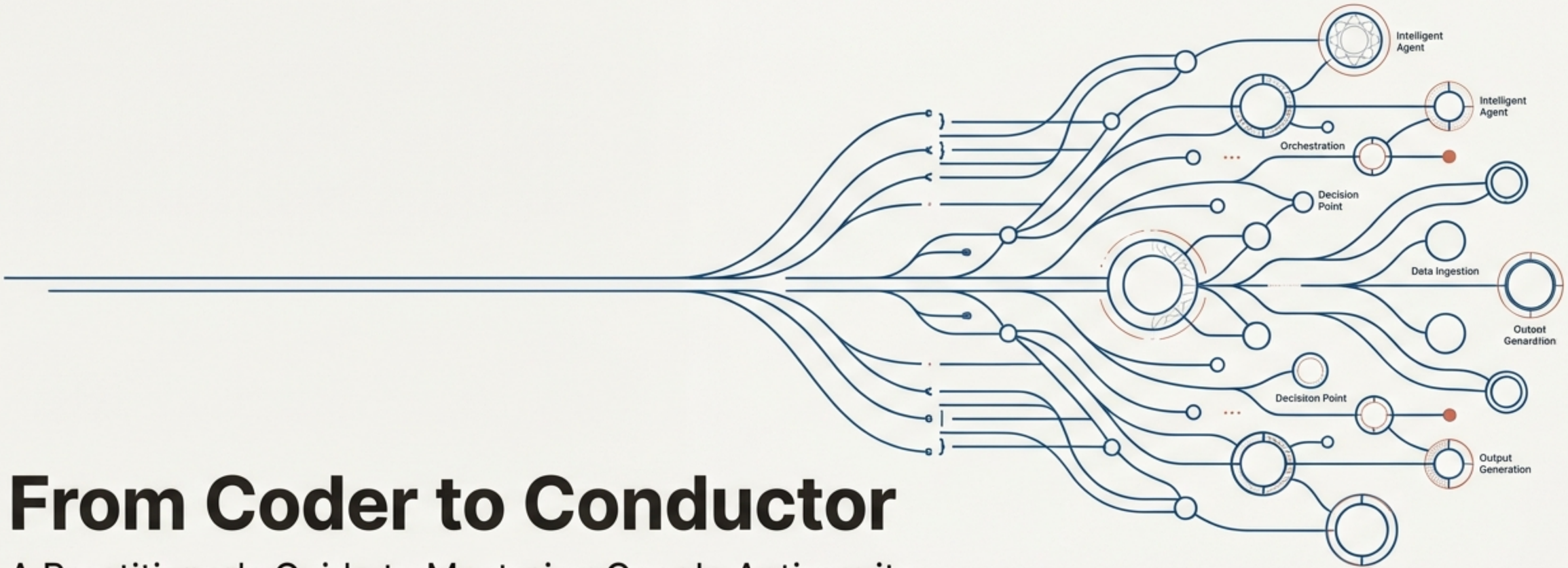


From Coder to Conductor

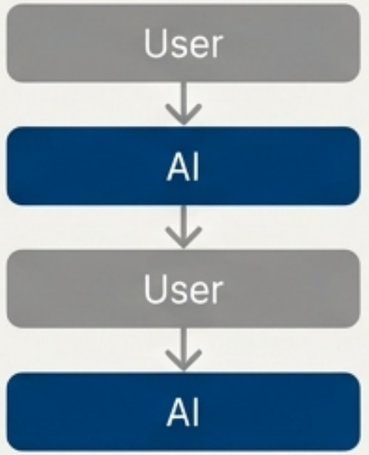
A Practitioner's Guide to Mastering Google Antigravity and Maximizing Agentic Value

> This guide goes beyond a feature list. It's a strategic playbook for adapting your development workflows to an agent-first world. Your new focus is architecting the solution, not implementing every single step. We'll walk you through a clear path from initial setup to advanced, "Antigravity-native" project architecture.

A New Paradigm: From Chatbot to Mission Control

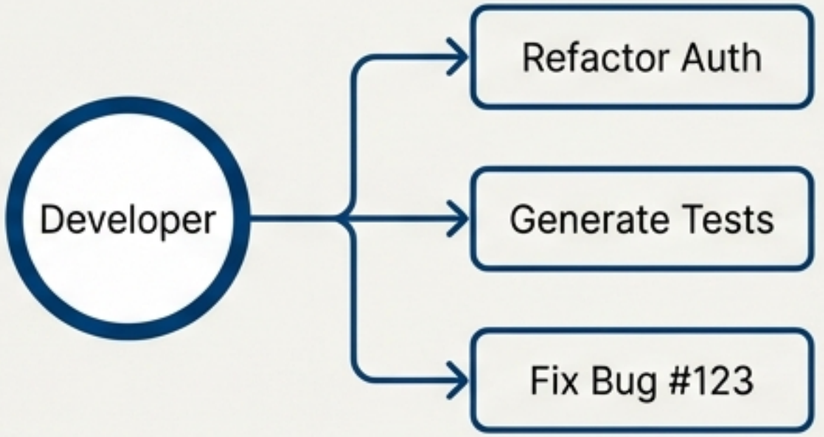
Antigravity fundamentally alters the IDE experience to prioritize agent management over text editing. It moves beyond the linear, synchronous limitations of a chatbot.

Linear Chat



Developer must wait for the AI to finish before asking the next question.

Mission Control



Dispatch multiple agents to work on different tasks simultaneously, multiplying your throughput.

Agent Manager

Your dashboard for high-level orchestration. Spawn, monitor, and interact with multiple agents operating asynchronously.

Editor

The familiar VS Code foundation, respecting your muscle memory for when you need to dive into the code.

Crawl: Your First Decisions as a Conductor

Getting started involves more than just installation; it's about defining the autonomy of your AI agents from the outset.

Key Configuration Steps

1. **Installation:** Available on Mac, Windows, and Linux. Requires a personal Gmail account and Chrome. (Note: Currently in preview with a free quota for premier models.)
2. **Crucial Setup Choice:** When prompted 'How do you want to use Antigravity agent,' you are setting two key policies. These can be changed at any time.

Terminal Execution Policy

Governs the agent's ability to execute shell commands.

- **Off**
  Never auto-execute (most secure, uses an Allow list).
- **Auto**
  Agent decides when to ask for permission.
- **Turbo**
  Always auto-execute (fastest, uses a Deny list).

Review Policy

Determines when the agent pauses for human review of its plans and artifacts.

- **Always Proceed**
  Agent never asks for review.
- **Agent Decides**
  Agent decides when to ask for review.
- **Request Review**
  Agent always asks for review.

Recommendation: The Codelab recommends '**Agent-assisted development**' as a balanced starting point (sets **Terminal** to `Auto` and **Review** to `Agent Decides`).

Your First Mission and Solving the ‘Trust Gap’

The Quick Win

Your first task should leverage a unique Antigravity capability. Instead of a simple coding task, try a web-based one.

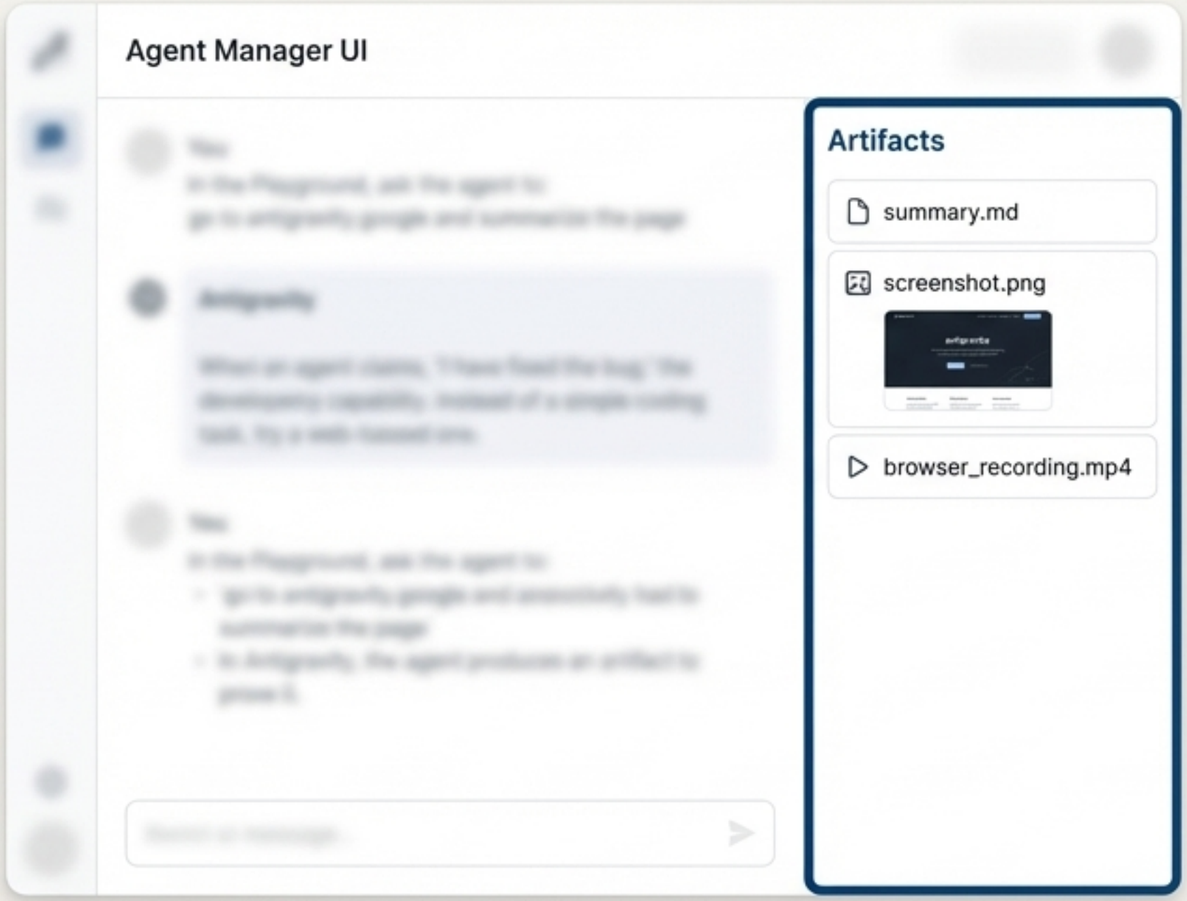
> In the Playground, ask the agent to:
> `go to antigravity.google and summarize the page`

The agent will prompt you to install the Antigravity browser extension, a one-time setup that unlocks its web capabilities.

Introducing Artifacts

As the agent works, it produces `Artifacts` to communicate its progress and results. This solves the “Trust Gap.”

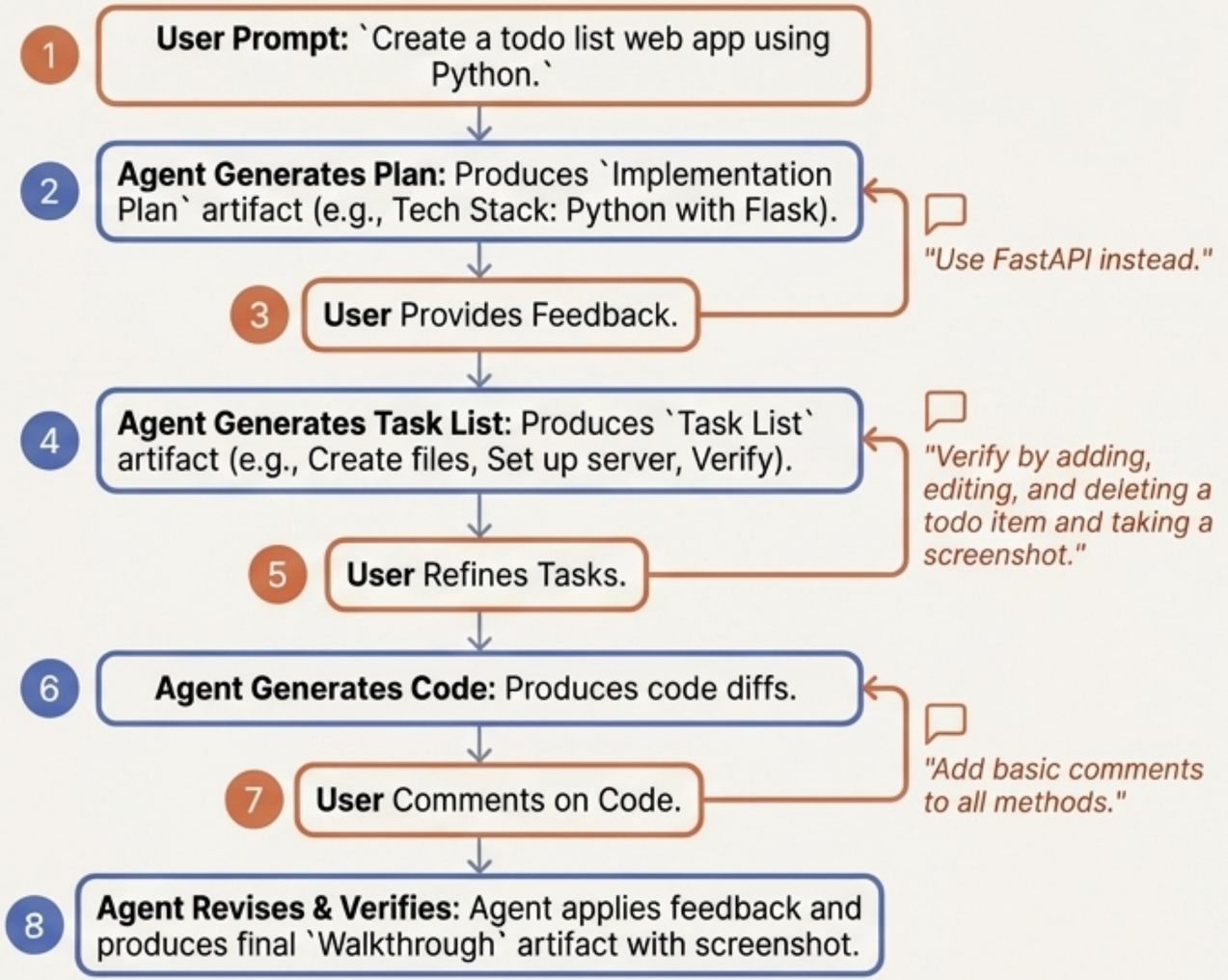
> *When an agent claims, ‘I have fixed the bug,’ the developer previously had to read the code to verify. In Antigravity, the agent produces an artifact to prove it.*



Artifacts make the agent’s work transparent and verifiable from day one.

Walk: Mastering the Interactive Feedback Loop

True agentic development isn't a single prompt; it's a conversation. Antigravity is built for iteration using a feedback mechanism similar to Google Docs comments.



🗣️ **Key Tip:** Always use `Planning` mode (instead of `Fast`) for complex tasks. This ensures the agent produces plans you can review and comment on before it starts coding.

Connecting Your Agent to Data with MCP

The Challenge

An AI agent's power is limited by what it “knows.” To build truly useful applications, your agent needs to understand your organization's data.

The Solution: Model Context Protocol (MCP)

> *Think of it like a USB-C port for AI. It allows the LLMs in your IDE to plug into your data sources in a standardized way.*

No more wrestling with config files. Connecting an agent to your data is a UI-driven experience via the Antigravity MCP Store.

AlloyDB

- The agent gains tools like `list_tables`, `get_table_schema`, and `execute_sql` to explore schemas and test queries directly in the IDE.

BigQuery

- The agent becomes a data analyst with tools like `forecast` to predict trends and `analyze_contribution` to understand metrics.

Looker

- The agent bridges code and business logic, using `get_explores` to find correct metric names and `run_look` to validate app output against official reports.

Customizing Agent Behavior with Rules & Workflows

Stop repeating yourself in prompts. Teach the agent your preferences and create shortcuts for common tasks.

Rules

System Instructions or global guidelines.
Guide the agent's behavior on all tasks within a workspace.

code-style-guide.md

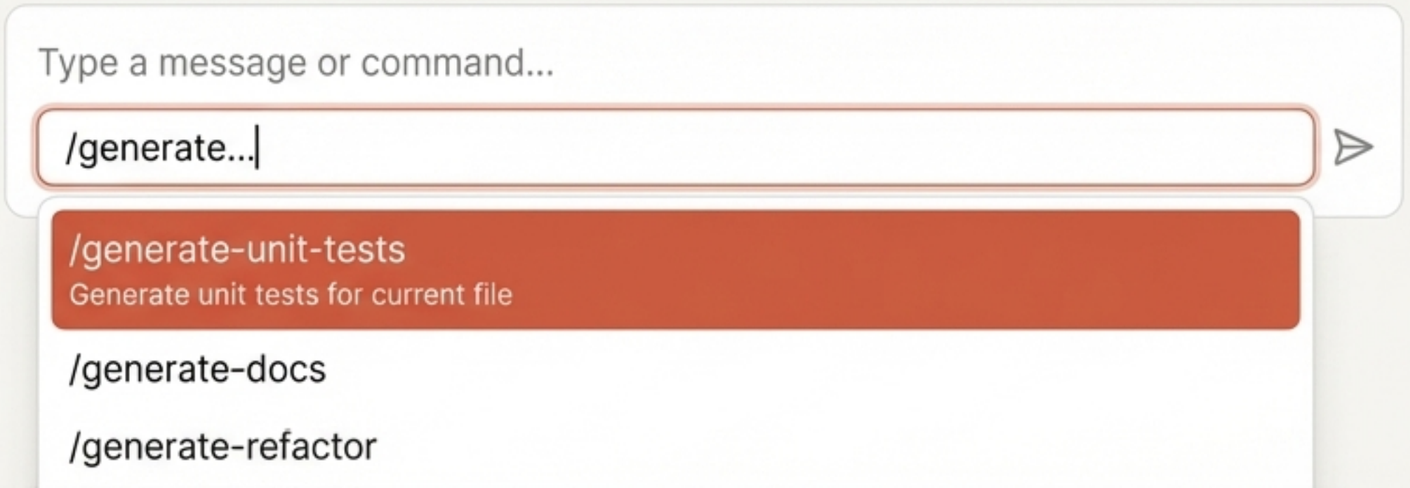
```
- Make sure all the code is styled with PEP 8 style
  guide.
- Make sure all the code is properly commented.
```

Workflows

Saved, on-demand prompts.
Trigger a specific, complex instruction with a simple command.

/generate-unit-tests

```
- Generate unit tests for each file and each method.
- Make sure the unit tests are named similar to files
  but with `test_` prefix.
```



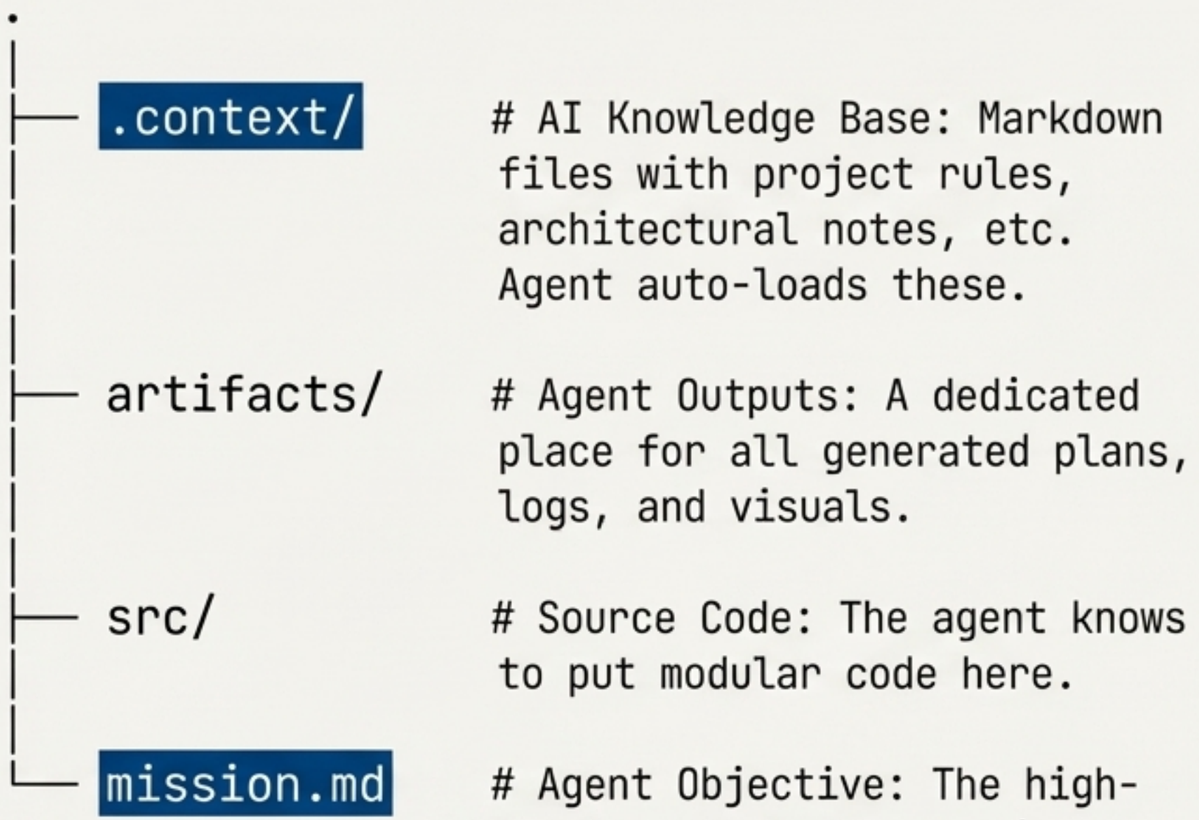
Run: Architecting an Agent-Native Workspace

The Problem with Standard Prompts

> “Repetitive labor is a waste of creativity. My ideal workflow is: Git Clone -> IDE already knows what to do.”

The Solution: A “Brainwashed” Workspace

Structure your project to pre-load an agent with essential context, rules, and objectives. This turns generic agents into project-specific senior engineers.



The Artifact-First Protocol

This workspace enforces the **Artifact-First** protocol. The Agent does not just write code; it produces tangible outputs (Artifacts) for every complex task, ensuring clarity and verifiability.

Old Way (Manual Prompting)

```
"Please write a snake game. Make sure to use
modular code. Put files in src. Don't forget
comments..."
```

The Antigravity Way (With a Brainwashed Workspace)

```
"Build a snake game."
```

The Agent's Automated Process

Pause

The agent consults its rules: "According to protocols, I must plan first."

Document

It generates an artifact: `artifacts/plan_snake_game.md`.

Build

It writes modular, well-documented code directly into the `src/` directory.

By structuring your workspace, you shift from giving low-level instructions to defining high-level g

The Future is Collaborative: Multi-Agent Swarms

Move beyond a single agent to a team of collaborating specialists. The `antigravity-workspace-template` introduces a Multi-Agent Swarm Protocol to collaborate at scale.

Architecture: The Router-Worker Pattern

```
🧭 [Router] Analyzing task: "Build a calculator and review it for security"
📤 [Router → Coder] Build a calculator
💻 [Coder] Creating calculator implementation... Done!
📤 [Router → Reviewer] Review code for security
🔍 [Rsviewer] Analyzing code... Review complete!
🎉 [Router] Task Completed!
```

Example Execution Log

This isn't just about code generation; it's about orchestrating an automated software development lifecycle.

Essential Governance: Securing Your Agent

Giving an AI agent access to your terminal and browser enables powerful automation but also introduces risks like **Prompt Injection** and Data Exfiltration.

Antigravity's Granular Permission System

Terminal Security

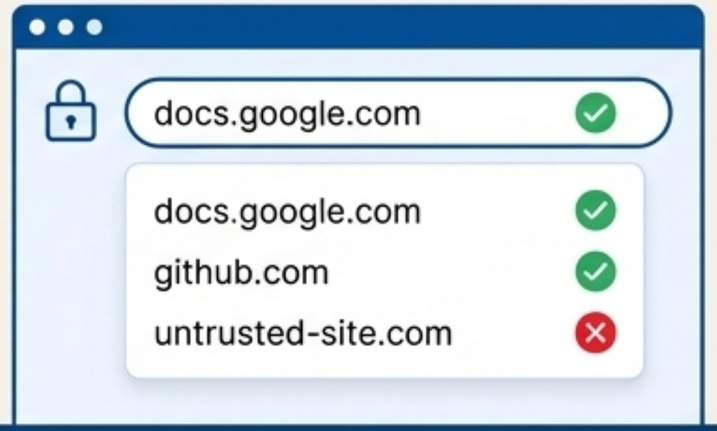
Policies: `Off`, `Auto`, `Turbo` modes give you control over command execution autonomy.

Allow List: Used with the `Off` policy. A "positive security model" where everything is forbidden unless expressly permitted. (e.g., allow `ls -al` but block `rm`).

Deny List: Used with the `Turbo` policy. A "negative security model" where everything is allowed unless forbidden. (e.g., deny `rm`, `sudo`, `curl`).

Browser Security

Browser **URL Allowlist**: Found in settings, this file (`~/.gemini/antigravity/browserAllowlist.txt`) lets you restrict the agent to a list of trusted domains, preventing it from visiting compromised sites.



You are in control. These tools allow you to balance speed and security according to your project's needs.

The Reality Check: Notes from the Field

Antigravity is a preview product, and early adoption comes with sharp edges. We hear the community feedback, and it's important to be transparent about the current state.

Aggressive Quotas

"Out of credits after about 20 mins." The free quota on premier models like `Gemini 3 Pro` can be hit quickly.

No Paid Tier (Yet)

"What's the strategy here? If I am into your IDE and your LLM, how do I actually use it? I can't pay for it..."

Preview Status & Bugs

"Users have reported UI glitches, crashes... stuck on 'Setting Up Your Account'... and some extensions (like `Pylance`) are not compatible."

The 'Killed by Google' Fear

"Acknowledging the elephant in the room. Many experienced developers are wary of adopting new Google tools due to a history of discontinued products."

This is an **active area** of development. The goal of the preview is to gather this feedback to build a robust, **production-ready** platform.

Your Path to Agentic Mastery

CRAWL: Master the Fundamentals.

Understand the Agent Manager, set your policies, and use Artifacts to build trust.

WALK: Integrate Agentic Workflows.

Master the interactive feedback loop, connect to data with MCP, and customize behavior with Rules.

RUN: Architect for an Agent-First World.

'Brainwash' your workspace with context, adopt the Artifact-First protocol, and explore multi-agent swarms.

Mastering Antigravity is more than learning a new IDE. It's about evolving your role from a line-by-line coder into a high-level conductor, orchestrating autonomous agents to achieve complex goals. The journey starts now.