Autonomous Coding Agents (IDE) - Home Use (Free)

This tutorial is aimed at helping you use Cline, Roo, Kiro, or Kilo **at home** with no, or minimal, cost to learn about large language models (LLMs), AI, and autonomous coding. By doing so, you can improve your coding skills, explore AI-driven development, and enhance your overall productivity.

Best course of approach would be to export this page as PDF or Doc, and email it to yourself as 3rd Party LLMs are filtered and blocked in the Group.

Introduction

The world of software development is continuously evolving, and developers are always in search of tools that enhance productivity and efficiency. Autonomous coding agents like Cline and RooCode represent a groundbreaking advancement in this quest. These AI-powered tools automate routine coding tasks, allowing developers to focus on more innovative and complex aspects of their work.

The effectiveness of Cline and RooCode largely depends on the quality of their underlying AI models. For example, OpenAI's GPT-4 excels in understanding and generating human-like text, making it excellent for code generation and debugging. Similarly, Codex, another model by OpenAI, is fine-tuned specifically for programming tasks, providing robust assistance in writing and refining code.

By integrating Cline and RooCode into your workflow, you can significantly reduce time spent on repetitive tasks and minimize errors, leading to higher-quality and more reliable software. These agents democratize sophisticated AI-driven development, supporting everyone from seasoned professionals to beginners.

In this article, we'll show you how to set up and use Cline and RooCode effectively. By the end, you'll be ready to take advantage of these tools to enhance your coding process and improve your productivity.

For Work related use, rather than reinvent the wheel, I'll direct you to two articles that detail how to set up Cline and RooCode in your work environment (credits to the authors):

Cline → https://commbank.atlassian.net/wiki/spaces/GAH/pages/1069289205
Roo → https://commbank.atlassian.net/wiki/spaces/CEIMS/pages/1346734697

Benefits

Key Features

1. **Read and Write Multiple Files:** Both Cline and RooCode offer the ability to open, read, and write multiple files simultaneously. This is especially useful for operations requiring changes across several files like refactoring, updating license headers, or enforcing coding standards.
2. **Run Terminal Commands:** Built-in support for executing terminal commands directly from within VSCode. You can trigger these commands based on specific events like file save, project build, or through manual invocation. Both tools capture output from terminal commands, process it, and display it in an organized manner within the editor.
3. **Fix Command Output Errors:** Cline and RooCode can parse command outputs and identify errors, providing suggestions or automatic fixes for common script or command errors.
4. **Git Checkpoint Comparison and Restore**
   - **Snapshot Management:** Integration with version control systems like Git to create, compare, and restore code checkpoints. This is useful for tracking changes and reverting to previous states.
   - **Version Comparison:** Compare file versions easily, highlighting differences and allowing quick restoration of previous code snippets.
   - **Interactive Diff Tool:** Provides an intuitive interface to view differences between file versions, branches, or checkpoints. Supports side-by-side and inline diff views.
   - **Checkpoint Creation and Restoration:** Simplifies the creation of checkpoints at significant stages of development. Facilitates easy rollback to known good states in case of issues.
5. **Build and Compile code**, etc...

Use Cases

- Learning, Linting, Refactoring, or Converting Code
- Building a GUI Front-End to Your Back-End CLI
- Understanding GitHub Repos and Creating Documentation: Generate summaries, README files, and visual documentation like Mermaid or SVG diagrams for GitHub repositories.
- Updating Your Code and Website Pages: for instance you could translate the entirety of your website in another language.

- Understanding Tax-Related Matters: Automatically analyse and categorise financial documents like ATO income statements, bank statements, stock trading reports, and crypto transaction reports for tax purposes.

And there are a lot possibilities... as captured here in the Model Context Protocol section
https://docs.cline.bot/mcp/mcp-overview#use-cases

Choosing a Model

Most AI models are not free to use and often operate on a pay-as-you-go model, charging per requests and tokens processed. This can add up quickly depending on your usage (LLM Model Cost Map), so it's important to choose the right model based on your needs and budget.
We'll explore here the models without incuring pay-as-you-go fees...
We will not explore models offering a time-limited free API, as these would expire after a few months... (e.g.)

Gemini

Google Gemini offers a powerful API for leveraging advanced machine learning models in your development projects. To get started with Google Gemini, you need to create API keys, ensuring you comply with Google's eligibility and usage policies.

Step by Step to get access to the Gemini free-tier

1. **Requesting an API Key:** Visit Google AI Studio to request your key.

> If that API key doesn't seem to work, check the next section: Tips and Best practices

2.

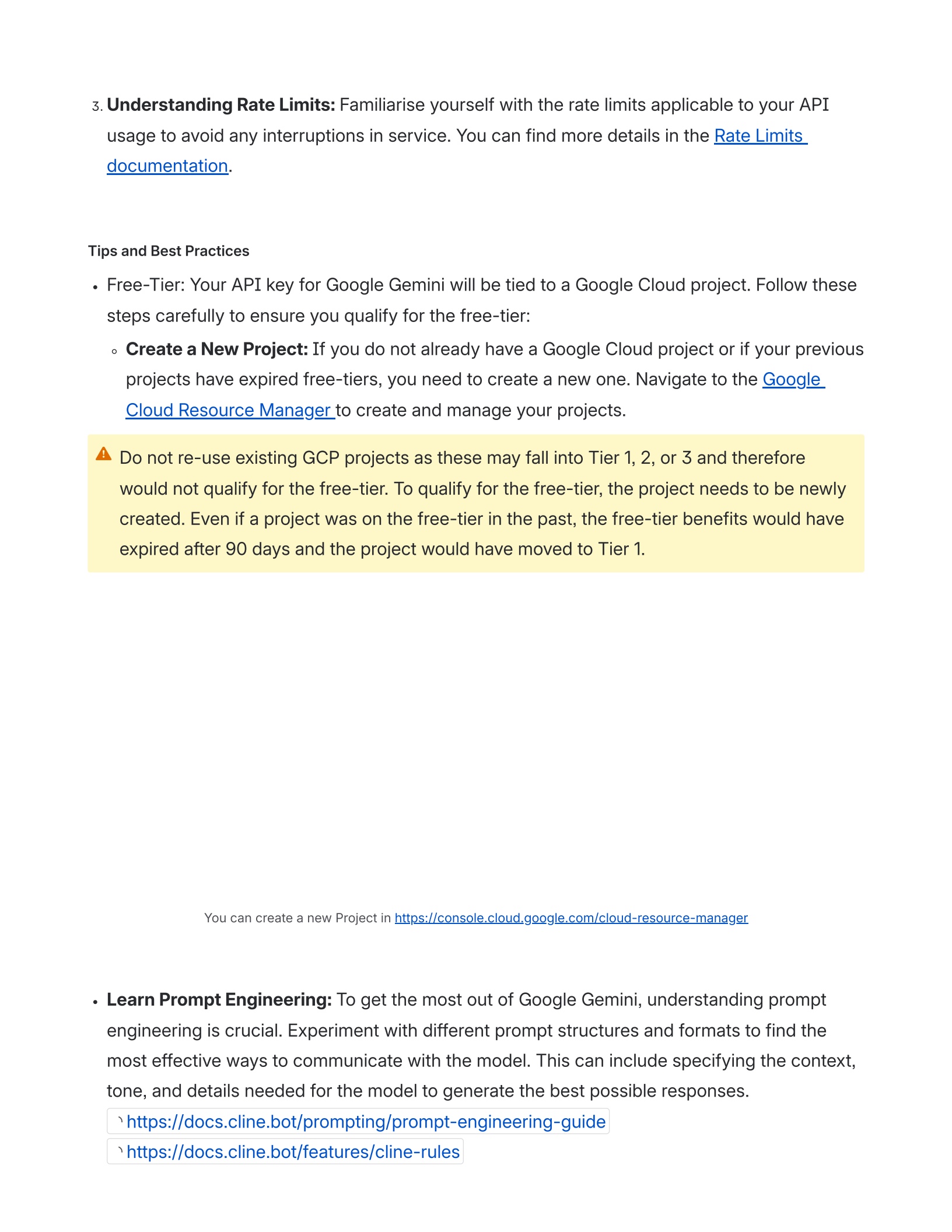
Make sure you are eligible to use Google Gemini by reviewing Available Regions document

3. **Understanding Rate Limits:** Familiarise yourself with the rate limits applicable to your API usage to avoid any interruptions in service. You can find more details in the Rate Limits documentation.

Tips and Best Practices

- Free-Tier: Your API key for Google Gemini will be tied to a Google Cloud project. Follow these steps carefully to ensure you qualify for the free-tier:
  - **Create a New Project:** If you do not already have a Google Cloud project or if your previous projects have expired free-tiers, you need to create a new one. Navigate to the Google Cloud Resource Manager to create and manage your projects.

> ⚠ Do not re-use existing GCP projects as these may fall into Tier 1, 2, or 3 and therefore would not qualify for the free-tier. To qualify for the free-tier, the project needs to be newly created. Even if a project was on the free-tier in the past, the free-tier benefits would have expired after 90 days and the project would have moved to Tier 1.



- **Learn Prompt Engineering:** To get the most out of Google Gemini, understanding prompt engineering is crucial. Experiment with different prompt structures and formats to find the most effective ways to communicate with the model. This can include specifying the context, tone, and details needed for the model to generate the best possible responses.
  `https://docs.cline.bot/prompting/prompt-engineering-guide`
  `https://docs.cline.bot/features/cline-rules`

Mistral

To get started, you need to create API keys for Mistral and Codestral. Follow these steps:

Note, there is a trade-off with this approach. The reason this is free is because your data, althought de-identified will be used by Mistral and product improvment.If you are not confortable with this, best to skip this section and jump to the Ollama Section below.

In terms, of limits, Mistral is ~~quite...~~ very generous with the free-tier (to-date... 21-May-2025)
`https://api.allorigins.win/raw?url=https://help.mistral.ai/en/articles/225174-what-are-the-limits-of-the-free-tier`

Step by Step to get access to the free-tier on Mistral

1. Go to the Mistral platform.
2. Navigate to the Billing / Experimental section.

Subscribing to the free experimental plan

3. Create your API keys

Repeat the same process for Codestral.

LiteLLM & Mistral

Installation

LiteLLM requires Python.

Follow these steps to install LiteLLM:

```
pip install 'litellm'
pip install 'litellm[proxy]'
```

Configuration

For Mistral:

```
export MISTRAL_API_KEY=<TheAPIKeyYourRequestedForMistral>
```

```
litellm --model mistral/mistral-large-latest
```

For Codestral:

```
export CODESTRAL_API_KEY=<TheAPIKeyYourRequestedForCodestral>
litellm --model codestral/codestral-latest
```

Tips and Tricks

- You can run multiple LiteLLM proxies by using different ports with the `--port` flag.

- The default socket is `0.0.0.0:<PORT>`, which means anyone in your subnet can connect to your authenticated instance. Why do you care? Because whoever uses it, can exhaust your limits or incur charges that you will need to pay. A simple nmap scan can reveal open ports:
  `sudo nmap -sV -p 4000,11434,8000,5000 10.0.0.0/16`
  To restrict access, run LiteLLM it with `--host 127.0.0.1`

- Run LiteLLM in the background and set it to auto-start. You can create a Windows service, use the Start folder, or use the current version run method.

Other *free* Models

To explore additional "free-mium" models (often with time or credit limitations), you can browse around. Check out this page: https://github.com/cheahjs/free-llm-api-resources

You can also start with free credits on Cline (app.cline.bot), available AI Models are Anthropic Claude 3.5-Sonnet (recommended for coding), DeepSeek Chat, Google Gemini 2.0 Flash. But free credits are credits, once they are consumed, the model is not longer free to use...

Ollama (Self-Hosted)

Ollama is a versatile tool for running machine learning models locally.
It allows you to pull and run various models based on your compute power.

https://ollama.com/search

In this section, we will walk through the steps to install Ollama, pull a model, and run it locally.

Installation

1. Download and install Ollama.
2. Pull the model of your choice based on your compute power
   `ollama pull mistral-small3.1`
3. Run the model:
   `ollama run mistral-small3.1`

Watch Ollama swallowing all your resources... Video Editing, PCVR, Mining Crypto, building rainbow tables, or running a model; You'll have to choose !

Ollama supports a wide range of models, including language models, image recognition models, and more. We will discuss how to configure Ollama in VSCode for seamless integration with your development environment in the next section.

Configuration

In your IDE / VSCode

Installing the extension

1. Download Cline or Roo Code extensions in VSCode
2. Open Cline or Roo Code, and select the gear wheel in the top right

3. In the settings, choose LiteLLM or Ollama as API Provider with the right URL and port.

4. No need to authenticate here
    a. For Mistral: We're already authenticated on the local (LiteLLM) proxy with our API Keys.
    b. For Ollama, this is running locally hence doesn’t require authentication

Setting up Cline to use your Browser

Browser Section in Cline

In order to connect, you'll need to enable the "Remote Browser" for Chromium type browsers (e.g. Edge, Chrome, Brave, etc...) `--remote-debugging-port=9222`

Edge running with remote debugging

⚠ While Firefox offers advantages, particularly its ability to use custom HTTP proxies (unlike Chrome-based browsers which are locked to system proxies via JAMF), **it is not compatible** with Cline and Roo Remote Browser according to the official documentation . Although you can start Firefox with remote debugging enabled (`/Applications/Firefox.app/Contents/MacOS/firefox --remote-debugging-port=9222 --width=1920 --height=1080`), Cline and Roo will not establish a connection. This limitation means Chrome-based browsers must be used, but they're restricted by JAMF-enforced system proxies, preventing custom proxy chains like:

- `Charles → Alpaca → Prisma`
- `Fiddler → CNTLM → Prisma`

Instead, all traffic must go through Prisma as the first hop, limiting testing visibility and inspection capabilities.

- Capturing Firefox traffic is covered in detail here: `https://commbank.atlassian.net/wiki/spaces/SBD/pages/989943066/Web+Debugging#Firefox-setup` (Note that Firefox Dev Tools also provides the ability to monitor HTTP requests and responses)
- Capturing Chrome-based traffic or all traffic is documented here but is for education purpose only as doing it will breach corporate policies and fair use of SOEs: `https://commbank.atlassian.net/wiki/pages/createpage.action?spaceKey=SBD&title=Local%20MITM%20%28capture%20all%20traffic%29%20from%20the%20SOE&linkCreation=true&fromPageId=1367284427`

One workaround is to use Brave (unmanaged) with FoxyProxy extension to use a browser-proxy (the local proxies), then hop to the enterprise proxy (Prisma).
You can then have a browser, blocking all noise (Ads, including your custom rules with Adblock syntax), and without interruption/noise with/from your other browsing activity.
You can for instance create a custom entry in your host file for:
`127.0.0.1 whatever.example.com`
and access it via `Brave` or `Firefox` (since the local proxy knows it's local) , whereas Edge or Chrome would not resolve it since Prisma proxy does not know the host and would drop the traffic.

Suggest you use a different browser than your main one for Cline and Roo so you do not interupt Cline/Roo work and vice-versa. For instance, if you use Chrome as your main browser, you might want to use Edge for Cline and Roo.

At start-up, and in the background

`nohup /Applications/Microsoft\ Edge.app/Contents/MacOS/Microsoft\ Edge --args --user-data-dir --remote-debugging-port=9222 &` if you want it to run in the background.
You can make this a KeepAlive LaunchAgent or Daemon as well, or a simply script to be launched at startup in System Settings / Login Items

OpenMyApps is a simply Shell script calling Apps in the background with nohup

Setting up Cline to use your Terminal

Cline can both use Bash, ZSH, or other interpretters, however in my experience, it does not like:

- Cygwin on Windows, use GIT Bash instead if you do not want to use CMD / PS
- `bash` or `zsh` on MacOS / Linux, but without `GNU Screen` and without `oh-my-zsh or powerlevel10K` theme which breaks compatiblity. You then see an error that Cline and VSCode cannot see the Terminal output (stdout, stderr). If you love OMZ as I do, simply, change the “default interpreter` to bash in VSCode only… so your Terminal still uses ZSH & co.

In your Repo

Cline Rules

Cline Rules allow you to provide Cline with system-level guidance. Think of them as a persistent way to include context and preferences for your projects or globally for every conversation.
You can create a `.clinerules` directory in the base directory in your project.

I this `.clinerules` directory, you can store several MD syntaxed file, it could be something like:

```
# Project Guidelines

## Documentation Requirements

-   Update relevant documentation in /docs when modifying features
-   Keep README.md in sync with new capabilities
-   Maintain changelog entries in CHANGELOG.md
```

```
## Architecture Decision Records

Create ADRs in /docs/adr for:

-   Major dependency changes
-   Architectural pattern changes
-   New integration patterns
-   Database schema changes
    Follow template in /docs/adr/template.md

## Code Style & Patterns

-   Generate API clients using OpenAPI Generator
-   Use TypeScript axios template
-   Place generated code in /src/generated
-   Prefer composition over inheritance
-   Use repository pattern for data access
-   Follow error handling pattern in /src/utils/errors.ts

## Testing Standards

-   Unit tests required for business logic
-   Integration tests for API endpoints
-   E2E tests for critical user flows
```

Cline Ignore

The `.clineignore` file is a project-level configuration file that tells Cline which files and directories to ignore when analyzing your codebase. Similar to `.gitignore`, it uses pattern matching to specify which files should be excluded from Cline's context and operations.

```
# Dependencies
node_modules/
**/venv/
.pnp
.pnp.js

# Build outputs
/build/
/dist/
/.next/
/out/

# Testing
/coverage/

# Environment variables
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

```

```
# Large data files
*.csv
*.svg
```

Conclusion

In this tutorial, we have covered the steps to use Cline or Roo Code and various models for free. By following these instructions, you can set up and run models locally using LiteLLM and Ollama.
We have also discussed how to create API keys for Mistral and Codestral, install LiteLLM, and configure it for use with different models.

Finally, we have walked through the steps to install and configure Ollama for running machine learning models locally. With these tools and techniques, you can leverage the power of machine learning and network scanning to enhance your development and security practices.

Cline and Free Models
Watch it do its magic at 20:25!

That's all folks... Happy Coding!

In the background, LiteLLM & Mistral are submitting and processing your tasks...