AIコーディングのための ローカルLLM構築入門

みやはら とおる
株式会社びぎねっと

Begi:net

講師 自己紹介

宮原 徹

- 1972年1月 神奈川県生まれ
- 1994年3月 中央大学法学部法律学科卒業
- 1994年4月 日本オラクル株式会社入社
  - PCサーバー向けRDBMS製品マーケティングに従事
  - Linux版Oracle8の日本市場向け出荷に貢献
- 2001年1月 株式会社びぎねっと 設立
- 2004年9月 Open Source Conferenceを開始
- 2006年12月 日本仮想化技術株式会社 設立

本日のアジェンダ

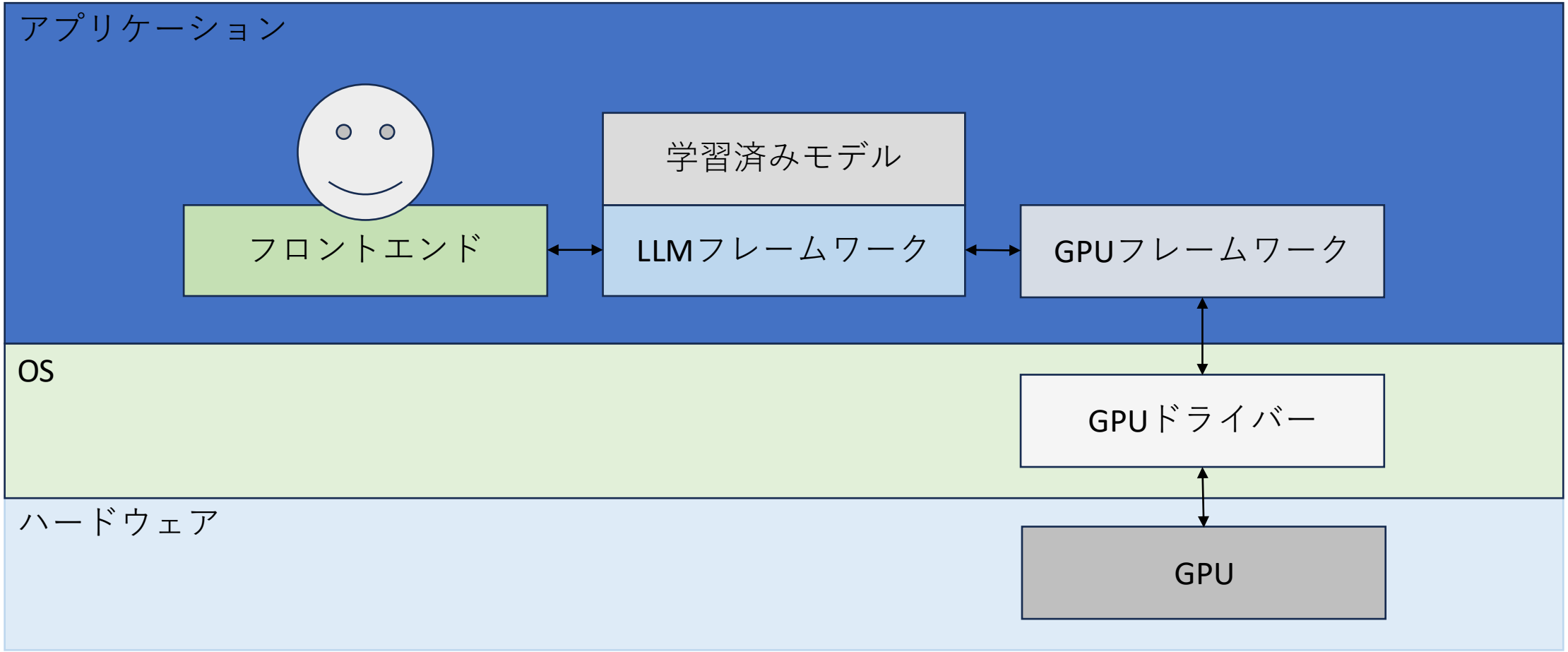
- AIを使うには
- AlmaLinuxでLLMを動かそう
- AIコーディングしてみよう

AIを使うには

AIについて

- Artificial Intelligence（人工知能）
- コンピューターの性能が向上したことで人間の知的生産活動を模して処理が行えるようになった
- 文章や画像、動画を生成する「生成AI」
  - 文章を生成するLLM（Large Language Models：大規模言語モデル）
  - 画像や動画を生成する拡散モデル（Diffusion Model）

LLM実行環境の構成図



LLMの構成要素

- 学習済みモデル
  - 様々な文章データを学習させたデータの集合体
  - モデルリポジトリとしてHagging Faceが有名
- LLMフレームワーク
  - モデルを読み込んで推論を実行するプログラム
- フロントエンド
  - チャットや外部APIなどを提供するプログラム

よく使われているLLMフレームワーク

- LM Studio
  - LM Studioはフロントエンド（非OSS・業務可）
  - バックエンドでllama.cppというLLMフレームワークを実行（OSS）
- Ollama（OSS）
  - 性能と使いやすさのバランスがいい（らしい）
  - 様々なAI連携ツールと組み合わせられる
- vLLM（OSS）
  - 高速実行を目指したフレームワーク
  - 並列実行に強み

LLMを動かすには

- GPUを使用できるハードウェア
  - OSからGPUを認識できる（仮想マシンではPCIパススルーが必要）
  - CPUでも動かせなくはないが遅い
- GPUを動作させるドライバー
- GPUを活用するためのGPUフレームワーク
  - NVIDIAのCUDA
  - AMDのROCm（ロックエム）
  - 汎用的なVulkan（旧OpenGL）
- LLMフレームワーク
  - 使用するGPUフレームワークへの対応が必要
- フロントエンドツール

GPU選び

- とことん性能を追求しないのであれば、とりあえず有り物でOK
- メモリ8GBぐらいでもそこそこのモデルは動く
- モデルのパラメータ数である◯B＞◯Gぐらいのイメージ
- 大きめのモデルを動かしたいなら20GB～32GBぐらいのGPUを
  - お高くなります（25万～）
- さらに大きいモデルを動かしたいならUMA
  - UMA（Unified Memory Architecture）はメインメモリとGPUメモリが共用のアーキテクチャ
  - Apple SiliconやRyzen AI Max+ 395、NVIDIA GB10など
  - 128GB搭載モデルで50万～100万程度

AlmaLinuxでLLMを動かそう

AlmaLinuxでNVIDIA製GPUを動かす

1. NVIDIA製GPUを搭載したPCにAlmaLinuxをインストール
2. GPUドライバーをインストール
3. CUDAをインストール
4. LLMをインストール
5. モデルをダウンロード

LLM動かすまでの作業手順

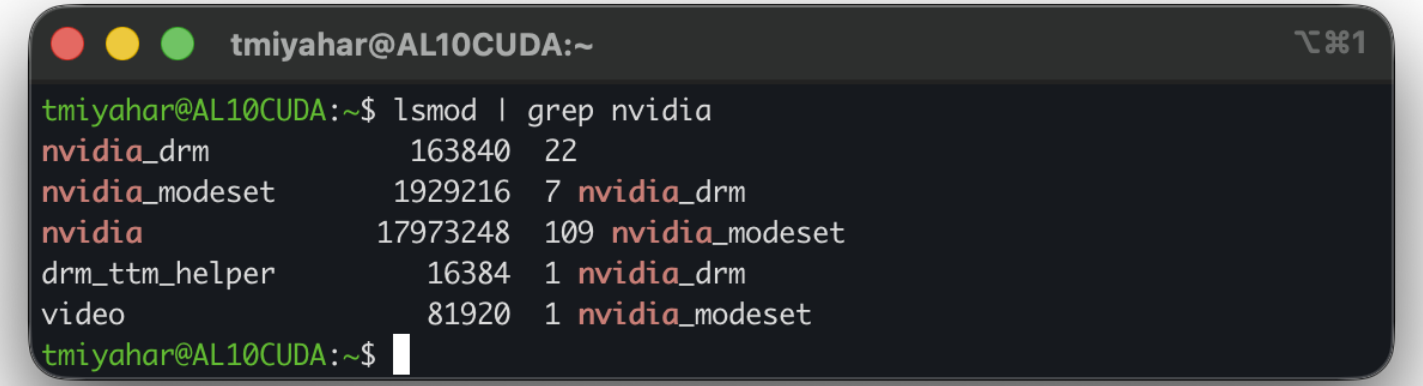
1. リポジトリの登録
2. NVIDIAドライバーのインストール
3. NVIDIA GPU関連ツールのインストール
4. GPUを認識しているかの確認
5. CUDAのインストール
6. LM Studioのインストール
7. LLMモデルのダウンロードと実行

リポジトリの登録

- $ sudo dnf install almalinux-release-nvidia-driver
  - NVIDIA Driver &CUDAリポジトリ
- epel-releaseパッケージも入る
  - EPEL10リポジトリが有効化
  - 実際にはCodeReady Builder (CRB)も有効化される

NVIDIAドライバーのインストール

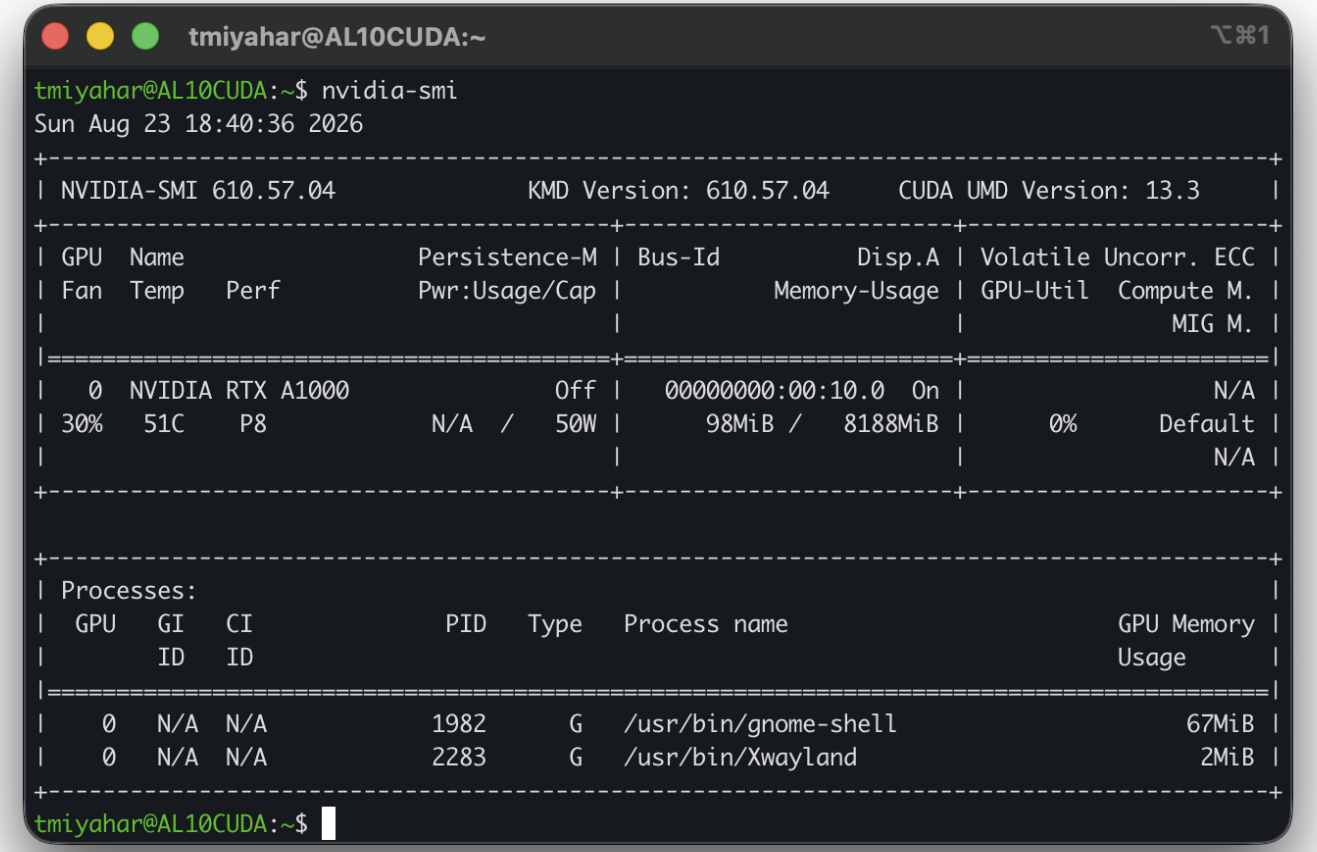
- $ sudo dnf install nvidia-open-kmod nvidia-driver
- インストール後に再起動
- lsmodコマンドでnvidia関連モジュールがロードされていることを確認



Begi:net

関連ツールのインストールとGPU確認

- $ sudo dnf install nvidia-driver-cuda
- $ nvidia-smi



Begi:net

CUDAのインストール

- $ sudo dnf install cuda
- CUDA関連の各種パッケージがインストールされる
- ダウンロードされるパッケージの合計サイズが大きいので注意

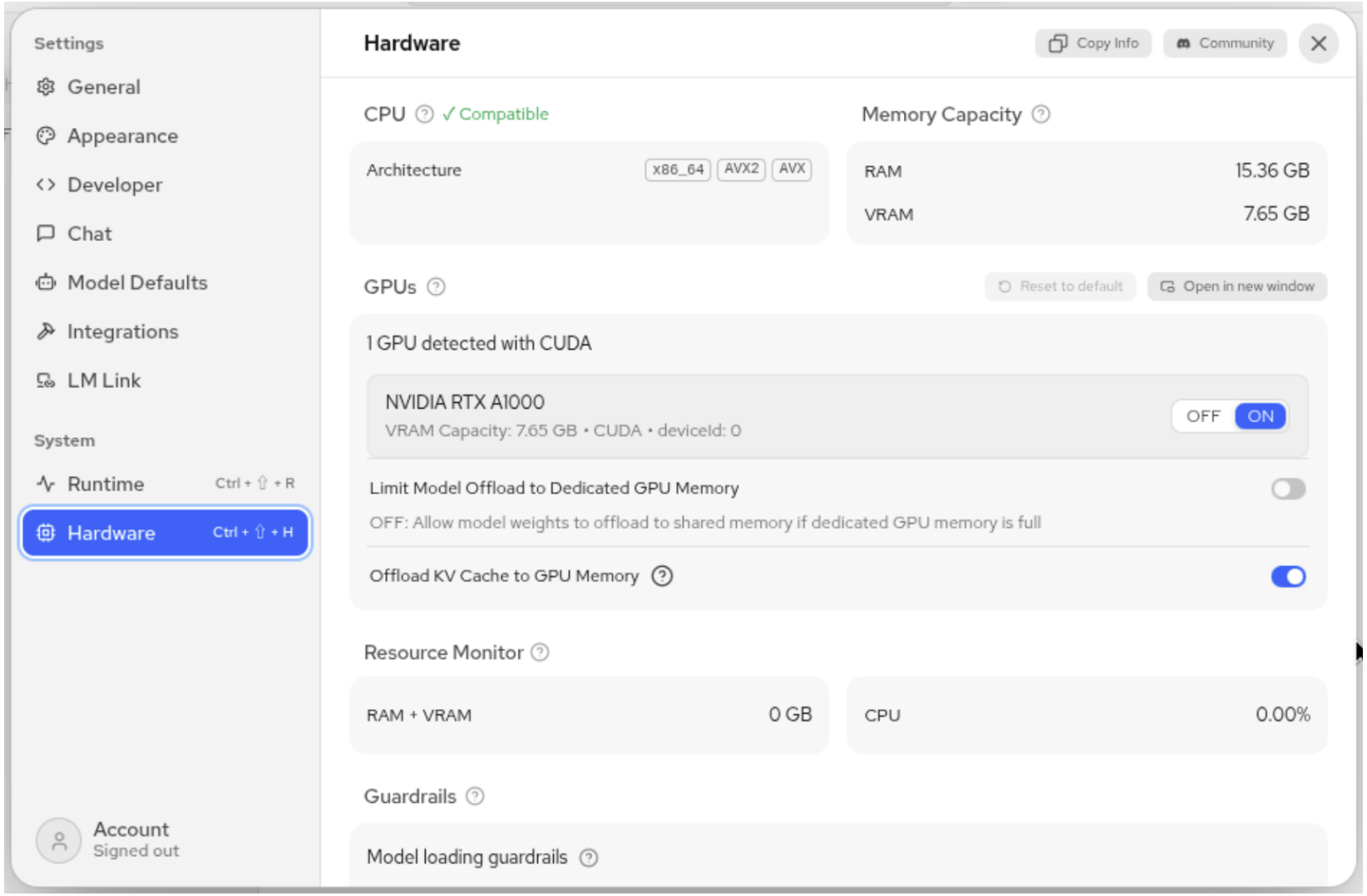
LM Studioのインストール

- LM Studioのダウンロードページへ
  - https://lmstudio.ai/download
  - $ wget https://lmstudio.ai/download/latest/linux/x64?format=AppImage
- AppImage形式で提供
  - 実行にはFUSEが必要なので事前にインストール
    - $ sudo dnf install fuse
  - ダウンロードしたファイルに実行権限を付与
    - $ chmod +x LM*
  - ダブルクリック、またはコマンドラインから実行

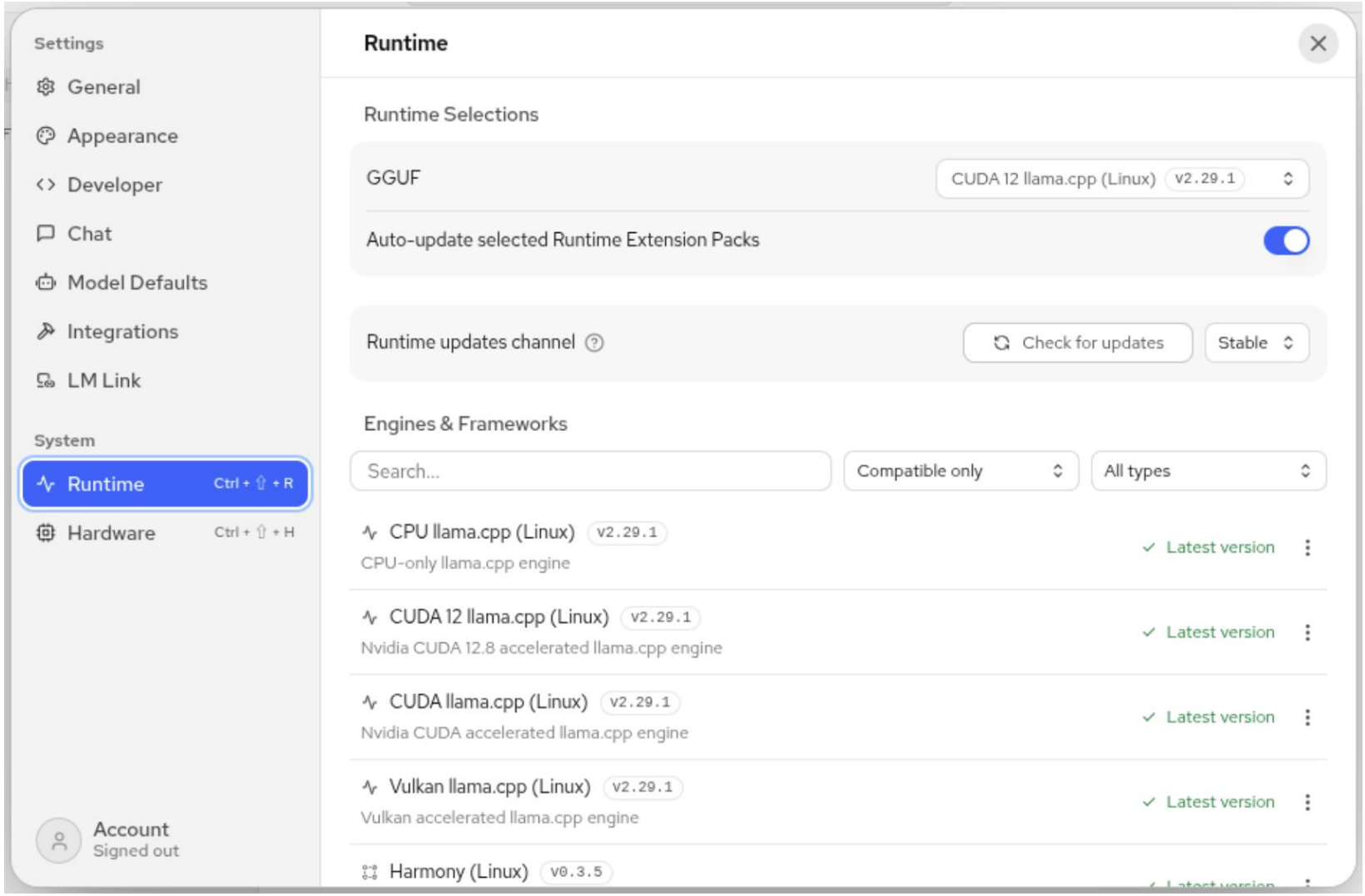
実行環境の確認

- LM Studioの環境設定の呼び出し
- 「Hardware」の確認
  - Memory Capacity、GPUsを確認
- 「Runtime」の確認
  - Engines & Frameworksが最新かを確認
  - Updateがある場合にはアップデートしておく
  - Runtime Selectionsで適切なものを選んでるかを確認

Hardware


Settings
General
Appearance
Developer
Chat
Model Defaults
Integrations
LM Link
System
Runtime Ctrl + ⇧ + R
Hardware Ctrl + ⇧ + H
Account
Signed out
Hardware
Copy Info
Community
CPU ✓ Compatible
Architecture x86_64 AVX2 AVX
Memory Capacity
RAM 15.36 GB
VRAM 7.65 GB
GPUs
Reset to default
Open in new window
1 GPU detected with CUDA
NVIDIA RTX A1000
VRAM Capacity: 7.65 GB • CUDA • deviceId: 0
OFF ON
Limit Model Offload to Dedicated GPU Memory
OFF: Allow model weights to offload to shared memory if dedicated GPU memory is full
Offload KV Cache to GPU Memory
Resource Monitor
RAM + VRAM 0 GB
CPU 0.00%
Guardrails
Model loading guardrails

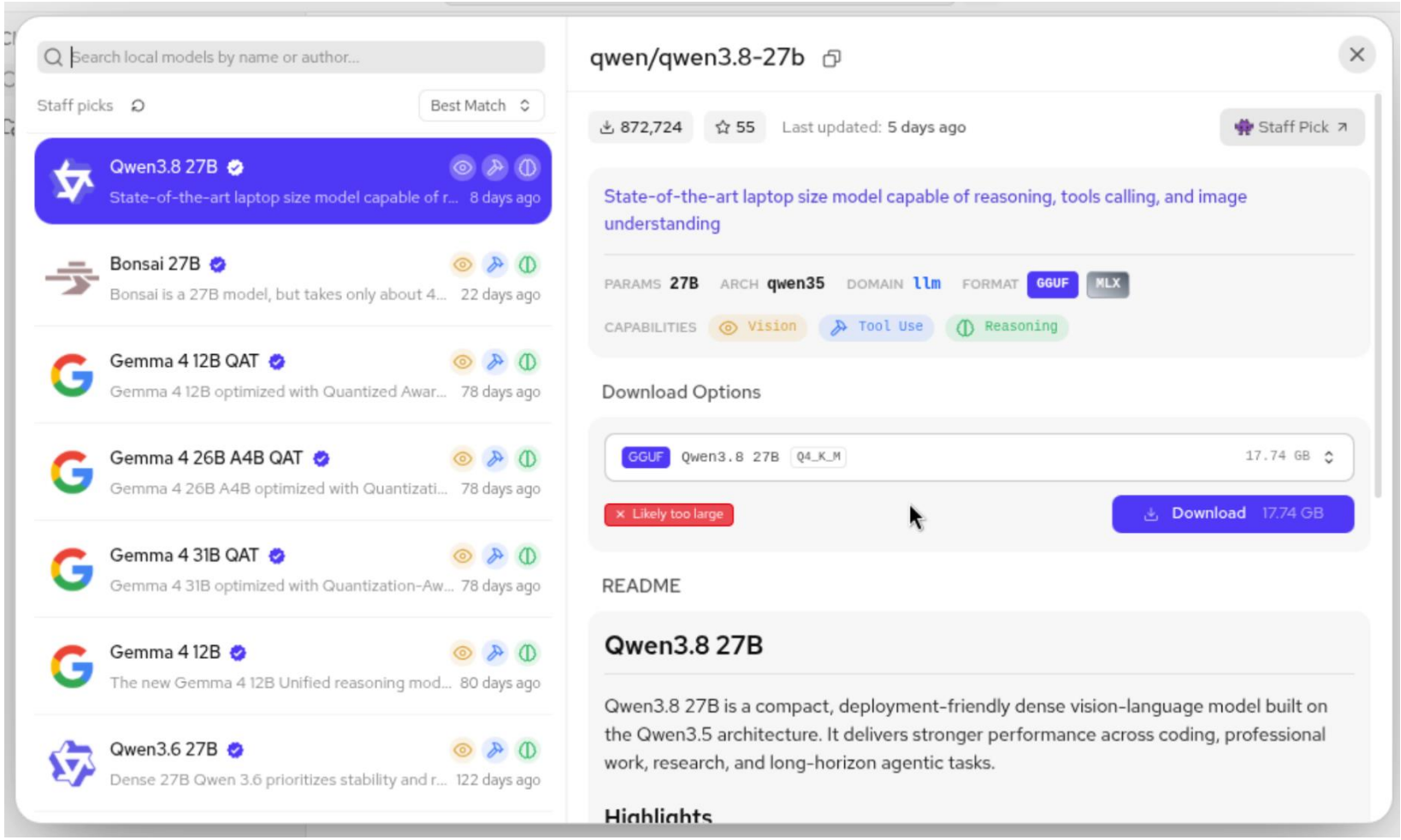
Runtime


Settings
General
Appearance
Developer
Chat
Model Defaults
Integrations
LM Link
System
Runtime Ctrl + ⇧ + R
Hardware Ctrl + ⇧ + H
Account
Signed out
Runtime
Runtime Selections
GGUF
CUDA 12 llama.cpp (Linux) v2.29.1
Auto-update selected Runtime Extension Packs
Runtime updates channel
Check for updates
Stable
Engines & Frameworks
Search...
Compatible only
All types
CPU llama.cpp (Linux) v2.29.1
CPU-only llama.cpp engine
Latest version
CUDA 12 llama.cpp (Linux) v2.29.1
Nvidia CUDA 12.8 accelerated llama.cpp engine
Latest version
CUDA llama.cpp (Linux) v2.29.1
Nvidia CUDA accelerated llama.cpp engine
Latest version
Vulkan llama.cpp (Linux) v2.29.1
Vulkan accelerated llama.cpp engine
Latest version
Harmony (Linux) v0.3.5

Runtimeについて

- 実行環境に合わせて色々な選択肢
  - CUDA：NVIDIA GPU環境
  - ROCm：AMD GPU環境
  - CPU：CPU環境
  - Vulkan：OpenGLの後継API
  - Metal、MLX：macOS用

Model Search



モデルのダウンロード

- Model Search画面を呼び出し
  - ウインドウ左側の虫眼鏡アイコン
- GPUメモリ、メインメモリの容量などを考慮してモデルを選択
  - モデルサイズ＜GPUメモリかどうか（Full GPU Offload）
  - モデルサイズ＜GPU＋メインメモリかどうか（Partial GPU Offload）

LLMモデルについて

- GGUFフォーマット
  - Georgi Gerganov氏が作成
  - 元々GGMLだったが拡張性の高いフォーマットに
- MLXフォーマット
  - Apple Silicon用に最適化
- Q◯
  - 量子ビットを減らすQuantizeしたもの
  - 4ビット（Q4）ぐらいで良いが、8ビット（Q8）も
- E◯B
  - 効率的な、効果的なモデル
- A◯B
  - MoEアーキテクチャのアクティブパラメータ数
- QAT
  - 量子化認識トレーニングモデル

LLMモデルの機能

- Reasoning
  - 論理的推論を行ってくれる
- Tool Use
  - 開発ツールなどと連携して動作できる
- Vision
  - 画像入力から推論できる

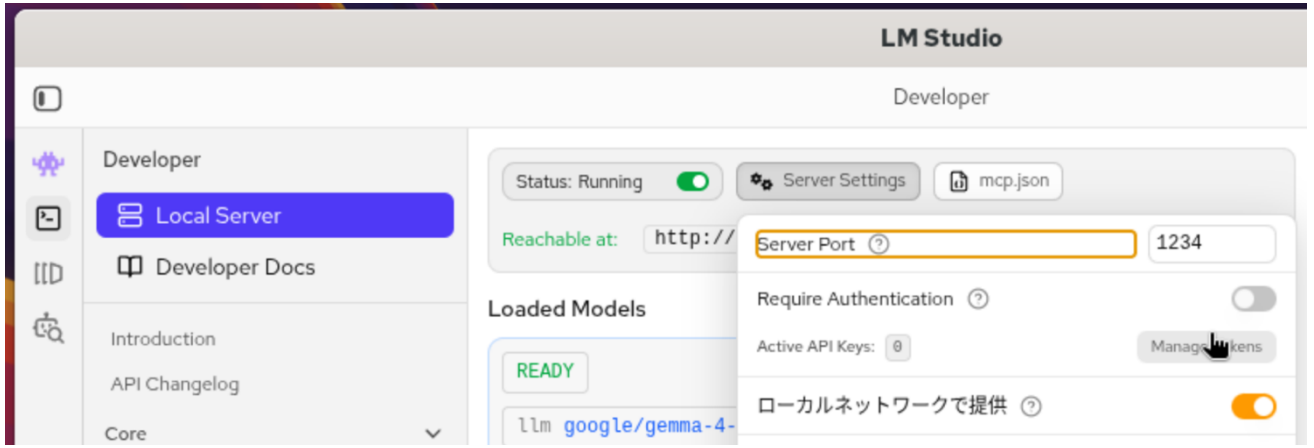
モデルのロード

- Chatを作成し、画面上部中央からロード
  - 切り替える毎にモデルをロードし直し
- Developer画面からモデルをロード
  - 同時に複数のモデルをロード可能
  - ロードされているモデルの分だけメモリを消費
  - Chatで簡単にモデル切り替え可能
  - Serverを起動し外部に対してAPIを提供

AIコーディングしてみよう

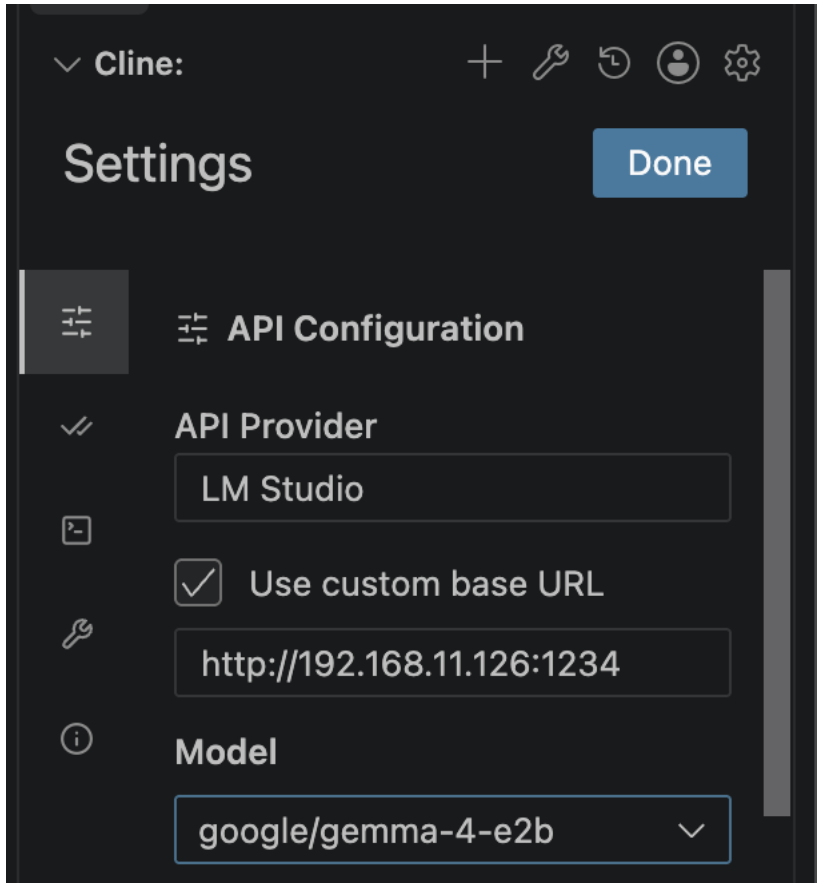
VS Code＋Cline＋LM StudioでAIコーディング

- LM StudioでLLM Serverを動作
  - Developerモードで起動
  - Server Settingsで「ローカルネットワークで提供」をオン
  - ファイアーウォールの設定を変更
    - $ sudo firewall-cmd --permanent --add-port=1234/tcp
    - $ sudo firewall-cmd --reload



Clineプラグインの設定

- VS CodeにClineプラグインをインストール
- API設定でLM Studioに接続
  - 同じマシンで動作させていれば簡単
  - 別のマシンの場合カスタムURLで
  - LLMサーバーのファイアーウォールを通せるようにするように注意
  - 接続するとモデルが選択できます



Clineの使用

- Planモードで仕様を検討
  - 細かい仕様を聞いてくるので答えてあげる
  - ファイルの書き込みなどはできない
- Actモード
  - 実装を進めてくれる
  - ファイルの書き込みに失敗するなどストレスが多い

今のところの評価

- あまり複雑な開発は行わせられない？
- ファイルの保存が苦手
  - index.htmlを1度は作れるが、修正版を上書き保存できない
  - なんとか保存させようとするとループする
- 小さいモデルの限界？
  - 小さいモデルでも性能が高いものが出てきている
  - 仕様の検討などはまだまだ大きいモデルが有利
  - モデルを大きくするとスループット性能は落ちる
  - 大きめのプログラムになるとコンテキスト長を長くする必要がある
  - GPUメモリ容量との兼ね合いが大きくなる