cmgenerate

Local LLM inference in Rust: a library, CLI, chat client, HTTP server, and tool agent. Reads memory-mapped weights and runs quantized kernels on the CPU, with CUDA and Metal acceleration. The core depends on `half`, `memmap2`, and `serde_json`; interactive clients also use `rustyline`.

Build and run

```
cargo build --release --workspace
./target/release/cmgenerate model.gguf "Explain Rust ownership." -n 128 -t 0
./target/release/cmchat model.gguf
./target/release/cmserve model.gguf --listen 8080
./target/release/cmagent model.gguf "Summarize this repository."
./target/release/cmfinetune model.gguf train.jsonl
./target/release/cmtranscribe phonon-2/ talk.wav
./target/release/cmspeech kokoro-82m/ "Hello, world." -o hello.wav
./target/release/cmvoiceagent --asr phonon-2/ --tts kokoro-82m/ model.gguf --gpu
```

Use `cargo build --release` to build only `cmgenerate`. With no model argument, `cmgenerate` runs a small demonstration with synthetic weights.

Model input can be a GGUF file, safetensors file, llama2.c `.bin`, or HuggingFace directory containing `config.json`. For cached downloads, use the snapshot directory. Chat formatting is detected automatically; `--raw` continues plain text. `--help` lists each program's options, `/help` the commands inside a conversation. The manual pages in `man/` explain the options in detail: `man -l man/cmgenerate.1`; likewise `cmchat.1`, `cmserve.1`, `cmagent.1`, `cmfinetune.1`, `cmtranscribe.1`, `cmspeech.1`, `cmvoiceagent.1`.

Program	Purpose
`cmgenerate`	Generate, inspect, score, or convert; also `--repl` and `--serve`
`cmchat`	Persistent chat with line editing and history
`cmserve`	HTTP server with address and model-name options
`cmagent`	Chat with file tools and optional shell commands
`cmagent-webui`	egui/WASM agent workspace with passkey + TOTP login
`cmfinetune`	LoRA finetuning on the CPU, and merging an adapter into a GGUF
`cmtranscribe`	Speech to text with Parakeet TDT (Phonon-2 or a Parakeet GGUF) or streaming Parakeet RNNT (`.nemo`, `--live`, speakers with `--diarize`); text, JSON, SRT or VTT
`cmspeech`	Text to speech with Kokoro-82M; English built in, more languages with espeak-ng; German through German Kokoro fine-tunes
`cmvoiceagent`	Spoken front end for `cmagent`: microphone, Parakeet, Kokoro, spoken approvals

Options

```
$ ./target/release/cmgenerate --help
usage: cmgenerate [OPTION]... MODEL [PROMPT]

MODEL is a .gguf, .safetensors, llama2.c .bin or Hugging Face directory.
Without MODEL, a demo runs on stand-in weights. Details: cmgenerate(1).

generation:
  -n, --steps N          tokens to generate (default: until end of turn;
```

```
                        64 without a chat template)
  -c, --context N       context length to allocate (default: 4096)
      --window N        sliding-window attention over the last N tokens
      --sinks N         initial tokens kept visible with --window (default: 4)
  -t, --temperature F   sampling temperature, 0 is greedy (default: 1.0)
  -k, --top-k N         top-k sampling, 0 disables (default: 40)
  -p, --top-p F         nucleus sampling, 1 disables (default: 0.9)
  -s, --seed N          random seed (default: time)
      --raw             no chat template; continue PROMPT as text
      --image FILE      an image (PNG, JPEG, WebP, GIF, BMP) ahead of PROMPT;
                        repeatable. Qwen3.5 and Qwen3-VL models
      --mmproj PATH     their vision tower: an mmproj GGUF or a HuggingFace
                        checkout (default: the checkout itself, else an
                        mmproj*.gguf beside the model that fits it)
      --speculate N     max draft tokens per step, 0 disables (default: auto)
      --no-mtp          draft by prompt lookup even if the model has MTP
      --mtp-from PATH   load the MTP layer from safetensors (file or dir), or
                        draft with a Gemma 4 assistant GGUF (mtp-*.gguf)

modes:
      --repl            interactive session
      --serve [ADDR]    HTTP server (default: 127.0.0.1:8080)
      --tokens          print the prompt's tokens and exit
      --score           print the prompt's perplexity and exit
      --inspect         print model metadata and exit; with PROMPT, list
                        tensors whose names contain it
      --trace           print the residual stream after each layer and exit
      --experts         report expert routing counts (MoE)
      --quantise FMT    write the model as FMT and exit (Q4_K, Q5_K, Q6_K,
                        Q8_0; BITCOS for ternary models)
  -o, --out PATH        output of --quantise (default: beside the model)

placement:
      --gpu             put as many weights as fit on the GPU, rest on the CPU
      --gpu-backend API auto, cuda, metal or vulkan; implies --gpu
      --gpu-device N,.. Vulkan devices, or CUDA ones with --gpu-backend cuda;
                        several split the layers
      --gpu-split R,..  each device's share of the layers, like 3,1
                        (default: by free memory)
      --gpu-as FMT      quantise to FMT for the GPU if the model does not fit;
                        formats the GPU cannot read (FP8 and NVFP4
                        other than under F8_E8M0 scales on CUDA) always
      --cpu-as FMT      quantise weights to FMT at load (Q4_K, Q5_K, Q6_K,
                        Q8_0); narrower weights are kept
      --kv-format FMT   KV cache format: q8 or f32 (default: q8)
      --threads N       worker threads, ahead of CMGENERATE_THREADS
      --lora PATH       apply a cmfinetune adapter at load
      --lora-as FMT     format of the matrices the adapter changes
                        (default: Q8_0)

environment:
  CMGENERATE_THREADS=N          worker threads (--threads N overrides)
  CMGENERATE_KERNELS=NAME       limit SIMD kernels: scalar, float, avx2
  CMGENERATE_QOS=0|all          macOS: threads requesting performance cores
  CMGENERATE_SPIN=N             spin iterations before a worker sleeps
  CMGENERATE_GPU_MEMORY=GIB     cap GPU memory use
```

```
CMGENERATE_CUDA_PTX=FILE      CUDA kernels to load in place of the built-in
                              ones (nvcc -ptx -arch=sm_NN src/gpu/kernels.cu)
CMGENERATE_EXPERT_CACHE=F     fraction of expert VRAM used as cache (0..1)
CMGENERATE_EXPERT_FETCH=F     fixed fraction of cache misses fetched to GPU
CMGENERATE_TRANSFER_QUEUE=0   no Vulkan transfer queue (no expert cache)
CMGENERATE_ANCHORS=0          no recurrent-state snapshots for rewinds
```

Weights and cache must fit available memory for practical performance. Mapping a model larger than RAM succeeds, but repeatedly reading its weights from disk can make generation very slow. Keep `--context` near the capacity you need.

`--window` bounds what the cache holds instead: every attention layer the model attends everything with sees only the last `n` tokens and the first `--sinks` of the sequence, and keeps a ring of that size rather than the context. It is the training-free sliding window of "Sliding-window beats linear attention" (arXiv 2608.28444), and the sinks are what make it work: on an 8k-token document, Qwen3.8-27B scores perplexity 6.81 attending to everything, 7.07 through a window of 4096, 8.17 through 1024 — and 13.98 through 1024 without the sinks. What is outside the window is gone, though: a fact placed 3k tokens back is not recalled through a window of 1024, so this is a memory economy for long sessions, not a free lunch. A layer the model trained with its own window keeps it; a model attending through a latent or an indexer takes no window.

CPU kernels select supported SIMD instructions at runtime. Thread count can matter more than core count; measure on your machine:

```
CMGENERATE_THREADS=8 ./target/release/cmgenerate model.gguf "Hello" -n 64 -t 0
```

`CMGENERATE_KERNELS=scalar|float|avx2` selects fallback kernels for comparison. See notes/PERFORMANCE.md for measured Qwen3.8-27B improvements, benchmark commands, hardware, and limitations of the measurements.

Vulkan

All inference frontends, including the Web UI, accept `--gpu-backend vulkan`:

```
./target/release/cmagent-webui model.gguf --gpu-backend vulkan --gpu-as q4_k -c 8192
```

The Vulkan backend supports every weight format, batched projections, expert slices, fused SiLU/GELU and Apertus xIELU feed-forward, and gated-delta layers with persistent GPU state. `--gpu-device N` selects a Vulkan device; `--gpu-device 0,1` splits the layers across several (also with `--gpu-backend cuda`, where the numbers are CUDA ordinals). Normal builds use committed shaders and load the Vulkan driver at runtime; no SDK is needed. See backend support, limitations and tests.

Finetuning

`cmfinetune` trains low-rank adapters on a frozen model, on the CPU, and writes them as safetensors. Every program takes `--lora <adapter>`, which adds it to the weights in memory (as Q8_0, or BF16 for float weights) before any offload; `--lora-as q6_k` holds the changed matrices in less memory. `--merge` writes the sum out as a GGUF instead.

```
./target/release/cmfinetune model.gguf train.jsonl --eval eval.jsonl --epochs 3 --lr 1e-3
./target/release/cmchat model.gguf --lora model.lora.safetensors
./target/release/cmfinetune model.gguf --merge model.lora.safetensors -o tuned.gguf
```

Each line of the data is `{"text": …}` (every token trained), `{"prompt": …, "completion": …}` or `{"messages": [{"role": …, "content": …}, …]}` (assistant turns and their end-of-turn token trained, in the model's chat template). Adapters go on `q,k,v,o,gate,up,down` by default (`--targets`).

Every architecture the library runs qualifies: attention, latent attention with or without the sparse indexer, gated-delta layers (Qwen3-Next, Qwen3.5, Kimi), hyper-connections, and dense, xIELU or routed feed-forwards. Frozen parts pass the gradient; adapters go on the projections. In a mixture the routed experts stay frozen and `gate,up,down` adapt the shared expert; in a recurrent layer `q` adapts the fused `attn_qkv`; in a latent one `q,k,v` adapt `attn_q_b`, `attn_kv_a` and `attn_kv_b`. `{"tokens": [...]}` lines train on token ids directly, for a model without a vocabulary. Gradients through the frozen weights use transposed Q8_0 copies, about 1.6x the size of a

Q4 model; `--backward-as none` decodes the weights as it goes instead, holding no copies, ~12% slower. A merge writes the matrices it changes as Q8_0 (`--adapted-as`): re-quantising them to Q4_K loses much of a small adapter. `--lora` keeps the file as it is. llama-3.2-1b trains at ~43 tokens/s at 512 tokens a sequence on 12 cores, ~170 with `--gpu-backend vulkan`, which puts the model and the transposed copies on the card.

HTTP

```
./target/release/cmserve model.gguf --listen 8080 --name local
curl http://127.0.0.1:8080/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -d '{"model":"local","messages":[{"role":"user","content":"Hello"}],"max_tokens":64}'
```

Supports `GET /v1/models`, `POST /v1/chat/completions` (including streaming), and `POST /v1/completions`. Point compatible clients at `http://127.0.0.1:8080/v1`. `cmgenerate model.gguf --serve` starts the same server.

Inference is serialized through one loaded model. Connections read independently, with bounded request sizes and receive deadlines. There is no authentication or TLS; the default address is loopback.

Agent tools

`cmagent` offers file discovery, numbered reads (a PDF as its text, extracted in pure Rust), and text search. Search uses ripgrep when installed (`regex: true` for a regular expression, `context: N` for the lines around each match); results are bounded and include truncation hints. `outline` gives a file's symbols with the lines each spans, from the language server or, without one, from the lines that look like declarations. Where there is a `.git`, `git` answers status, diff, log, show and blame, read-only. The web tools reach providers, each with a trust level: local (machine or institution), trusted (a service under contract or own control), untrusted (public internet). `--web LEVEL` (off, local, trusted, untrusted; `--allow-fetch-url` and `--allow-network` mean untrusted) offers those at or below it, and `--web-ask LEVEL` makes calls at or above it ask first (`--ask-web`: local). `fetch_url` reads a page over http or https as text (markup, scripts and menus dropped; `links: true` keeps links) through `curl`, `web_search` lists title, address and snippet from the search provider (DuckDuckGo's HTML page by default, untrusted), and `news_search` the newest articles of the last days (https://newsradio.creative-memory.de/2/, trusted). `~/.config/cmagent/providers` (or `--providers FILE`; never a project's `.cmagent/`) adds providers, one a line, e.g. `index search local searxng http://10.0.0.5:8888/search`. An endpoint `cmd:PROGRAM ARGS` runs a local program with the parameters as a JSON object on stdin, e.g. newsradio's `newsradio news local newsradio cmd:news-browse /srv/news/index` for its index on this machine; only a local or trusted one there may be at a private address, and `*.HOST` covers subdomains. `--fetch listed` makes those fetch lines a whitelist: `fetch_url` reaches nothing else. `--list-providers` prints them. A `model` line gives an `--api-base` service its level, e.g. `fhgenie model trusted openai https://fhgenie-preview.c.fraunhofer.de/v1` (else local at a loopback address, untrusted elsewhere; `--api-level` states it); `--model-level LEVEL` refuses a service above it, since it receives the whole conversation. Local weights are local. With `--write` or `--run` the model also gets `changes` (the task's diff against its checkpoint), `run_check` (the project's check on request), `undo_edit` (the last action put back, asked about like any action), `move_file` and `delete_file`; `write_file` makes the directories on the way. `--write` enables `write_file`, `edit_file` (one exact replacement) and `patch_file` (several, in order, each matched against the file as the ones before leave it, all or none); `--shell SANDBOX` enables shell commands and `check_command` for ones started in the background, run in the sandbox named: `offline` (no network; `--run`), `online` (the host's network; `--run --allow-network`), or a wrapper from `~/.config/cmagent/sandboxes` (`NAME none|net COMMAND...`, e.g. `rust none podman run --rm -i --network=none -v {root}:{root} -w {root} rust:1 sh -c`), which gets each command line as its last argument. Wrappers run outside bubblewrap, so only the user's file is read, not the project's. Full edit previews require confirmation unless `--yes` is supplied, and intervening changes are refused.

Typical setups. In the web UI the flags are server ceilings; an agent profile's `web:`, `web-ask:` and `shell:` set a new conversation's choices below them, and a value above a ceiling is dropped (Meeting Protocol has `web: off`, the other examples `web: on`, i.e. the server's level).

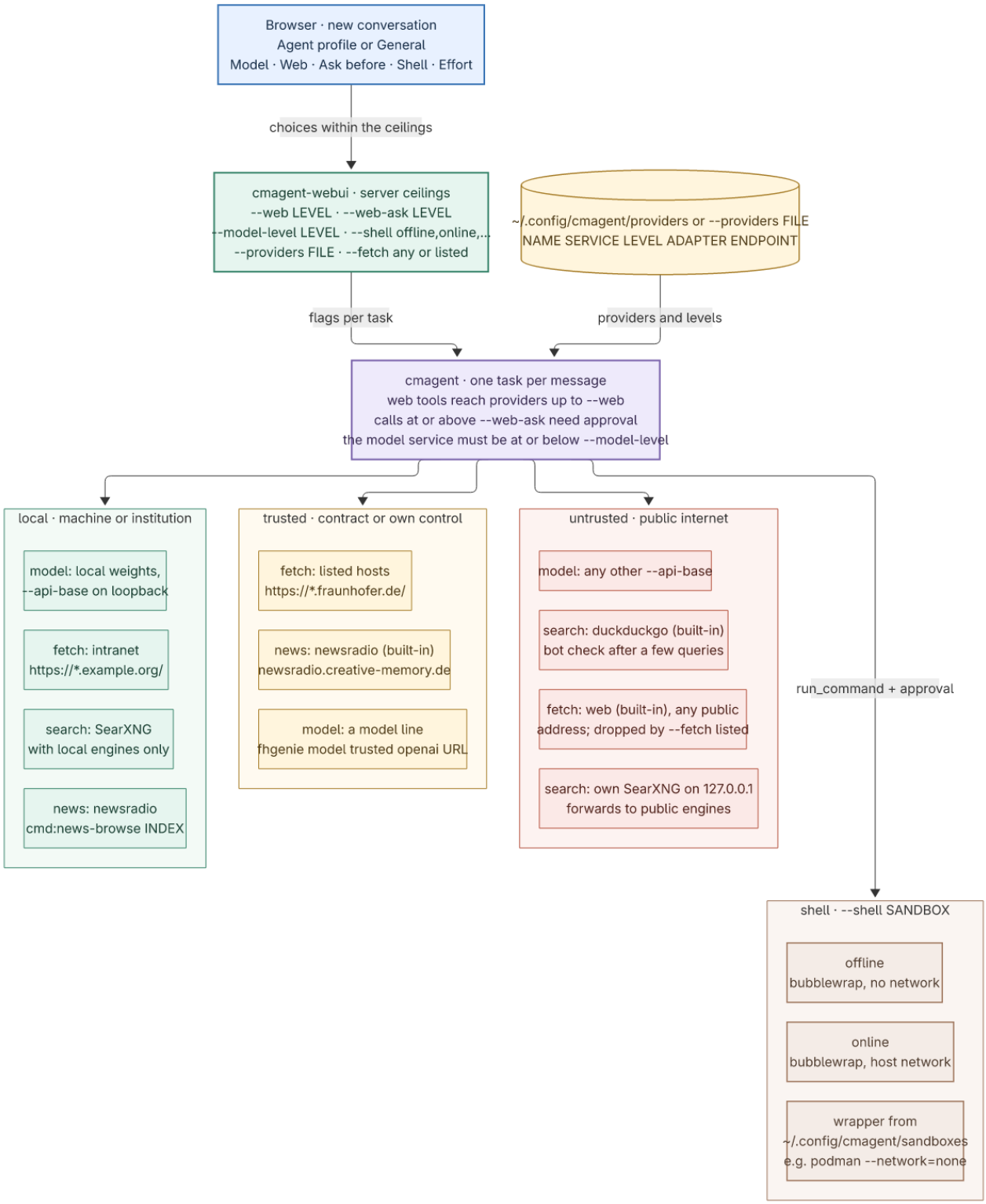


Figure 1: Providers, levels and sandboxes: the browser picks a profile and settings within the web UI's ceilings; the server starts cmagent per task with –web, –web-ask, –shell and –model-level; the providers file assigns each search, news, fetch and model endpoint the level local, trusted or untrusted; run_command goes to the offline, online or a wrapper sandbox.

Setup	Flags	Lines in `~/.config/cmagent/providers`
Nothing leaves the machine	local weights, `--web local`, `--fetch listed`, `--shell offline`	`newsradio news local newsradio cmd:news-browse /srv/news/index`, `intra fetch local pages https://*.example.org/`, a SearXNG with local engines only as `search local searxng`
Institute with a contracted model service	`--api-base URL --model-level trusted --web trusted`	`fhgenie model trusted openai https://fhgenie-preview.c.fraunhofer.de/v fhg fetch trusted pages https://*.fraunhofer.de/`
Public web research	`--web untrusted --web-ask untrusted` (each public call asks)	`searx search untrusted searxng http://127.0.0.1:8888/search` before the others: an own SearXNG spreads queries over several engines; the built-in DuckDuckGo page answers a bot check after a few queries in a row

After every edit the project's check runs in the sandbox and its first complaints follow the edit's result: `.cmagent/check` names the command, or `cargo check --all-targets -q --message-format=short` is implied by a `Cargo.toml` (`go vet ./...` by a `go.mod`); `--no-check` turns it off. A Cargo project also gets a language server — `rust-analyzer` from the path or from `rustup`, needing the `rust-src` component; `.cmagent/lsp` names another — started in the sandbox on the first question and offered as `find_definition`, `find_references`, `describe_symbol` (signature and documentation) and `find_symbol` (by part of a name, project-wide). Each takes a file, a line as `read_file` numbers them, and the name on that line.

`delegate` runs a sub-task in the same window after the call and rewinds it when the sub-task answers; only its answer stays, as the call's result. A sub-task cannot delegate, compact, or make more than 40 calls, and is told to answer when a quarter of the window is left.

Each call is held to valid JSON with a known tool name while it is sampled (for Apertus, an array of single-key objects); `--free-calls` turns that off. The system turn's cache is kept under `~/.cache/cmagent`, one file per model and project, so the next start reads only the task; `--no-prefix-cache` turns that off.

The model keeps its own notes: `remember` appends a dated line to `.cmagent/notes.md`, the latest 3 KiB of which open every task, and `recall` searches all of them. They are the model's, not yours — edit the file to change them; `/memory` shows them beside your pins.

An approval can be kept: for exactly that command or path, for every command of that program (or every path of that tool), or for a pattern (`cargo test *`, `src/*.rs`) — for the session, or for the project in `.cmagent/allowed.json`. A command is covered only if every `&&`/`|`/`;` segment is, and one with a `$(…)` substitution only by an exact answer. A refusal can be kept the same way — `/never rm`, `/never git push *`, `/never edit_file migrations/*`, or `!` before a letter at the prompt — and it wins over every approval, over `--yes`, and over a `y`; one segment is enough, and what is inside a `$(…)` counts. `/allowed` lists the rules and `/forget N` drops one; the web UI has the same under Settings.

Options can be kept in `~/.config/cmagent/config` and then `.cmagent/config`, one a line with its value after it (`--allow-read ../other`), `#` a comment, a bare line the model; the command line overrides both. `AGENTS.md` at the root goes into the system turn as the project's standing notes.

Executable files under `.cmagent/hooks` run at three points: `after-edit` in place of the check, `before-run` with each command line on its input (exit non-zero to refuse it; what it prints is the reason the model is given), and `after-task` once a task that acted is answered (a failure goes back to the model once). An executable under `.cmagent/tools` is a tool of the project's own, offered under its file name: it answers `--describe` with a JSON object `{about, acts, takes: [{name, about, kind, required}]}` (kind one of `string`, `integer`, `boolean`) and

is run with the arguments as one JSON object on its input, in the sandbox; what it prints is the result. One that acts is offered with `--write` or `--run` and asks like the others.

A directory with a `SKILL.md` under `.cmagent/skills` or `~/.config/cmagent/skills` is a skill: instructions for a kind of task. Its front matter's `name` and `description` are listed in the system turn, and `use_skill` returns the rest, or another file of the skill's directory; the project's skill wins on a shared name. With the web tools on, a built-in `web-search` skill (search with `web_search`, read with `fetch_url`, cite) is listed unless a skill on disk has that name.

At the prompt, Tab completes commands and paths, the history is kept in `.cmagent/history`, a line typed while the model works goes to it with the next tool result, `!command` runs your own shell and `!!command` keeps its output for the next task, `/edit` writes a task in `$EDITOR`, `/last` shows the latest tool result whole, `/jobs` the background commands, `--vi` gives vi keys and `--quiet` drops the startup report. `--tui` draws the same agent in a screen: transcript, status line and input, with approvals answered in place; it runs the agent as a child over `--bridge`.

File tools stay within `--root` (default: working directory). Shell commands need Linux and Bubblewrap: the project is writable, system binaries are read-only, the toolchain caches under `$HOME` (`.rustup`, `.cargo/registry`, `.npm`, pip) are read-only and credential files are left out, the GPU is visible, and in the `offline` sandbox network access is disabled. Commands share one network namespace for the session, held by a sleeping bwrap process and joined with `nsenter`, so a server one command starts in the background is reachable at 127.0.0.1 by the next and not from the host. `/tmp` is `.cmagent/scratch` and persists between commands; `read_file` can read it. `--allow-read PATH` and `--allow-write PATH` grant exceptions; `--no-gpu-in-shell` and `--no-home-in-shell` take those two away. Agent state stays protected; `.git` is writable, so a command can commit. A command that needs the accelerator's memory for itself — a model or a benchmark on the same GPU — says `gpu: true`: on Vulkan the agent's own model is moved off the device for the duration (weights re-read from the file afterwards, the rest of the device's memory kept on the host) and put back before the model reads the result.

Budget	Default
`--hops N`	no limit; a call repeated three times with the same answer is refused instead
`--command-timeout SECONDS`	3600 at most; 300 unless the command says `timeout` or runs in the background
`--task-timeout SECONDS`	14400
`--max-output BYTES`	16384 per result; the whole output is kept in `/tmp/output-N.txt`
`-c TOKENS`	32768

On Unix, Ctrl-C stops the task and keeps the model loaded. Commands and background children are terminated; generation stops between prefill batches or tokens.

Sessions and action evidence save under `.cmagent`; `--resume` restores them without replaying actions. New runs archive the previous session. `--log PATH` adds a JSONL log; `--once --require-tool` fails unless a tool actually succeeds.

- `/undo`, `/undo-task`: restore an action or task; refuse conflicting edits.
- `/changes`, `/recover`: inspect changes or review and undo interrupted work.
- `/remember TEXT`, `/memory`: pin and inspect decisions or unfinished work.
- `/compact`: archive the transcript and summarize context; also automatic near capacity. Goals and pins are kept verbatim. Pin critical decisions: summaries can lose detail. Sampling uses the current command-line settings.

Checkpoints cover source files, including binaries and deletions, up to 64 MiB. Ignored/build files, logs, Git metadata, and external write exceptions are excluded. Apertus uses native tool declarations and call arrays, but local 4B/8B trials still described calls instead of executing them. Qwen3-30B passed live tool checks. See agent checks for repeatable evaluations.

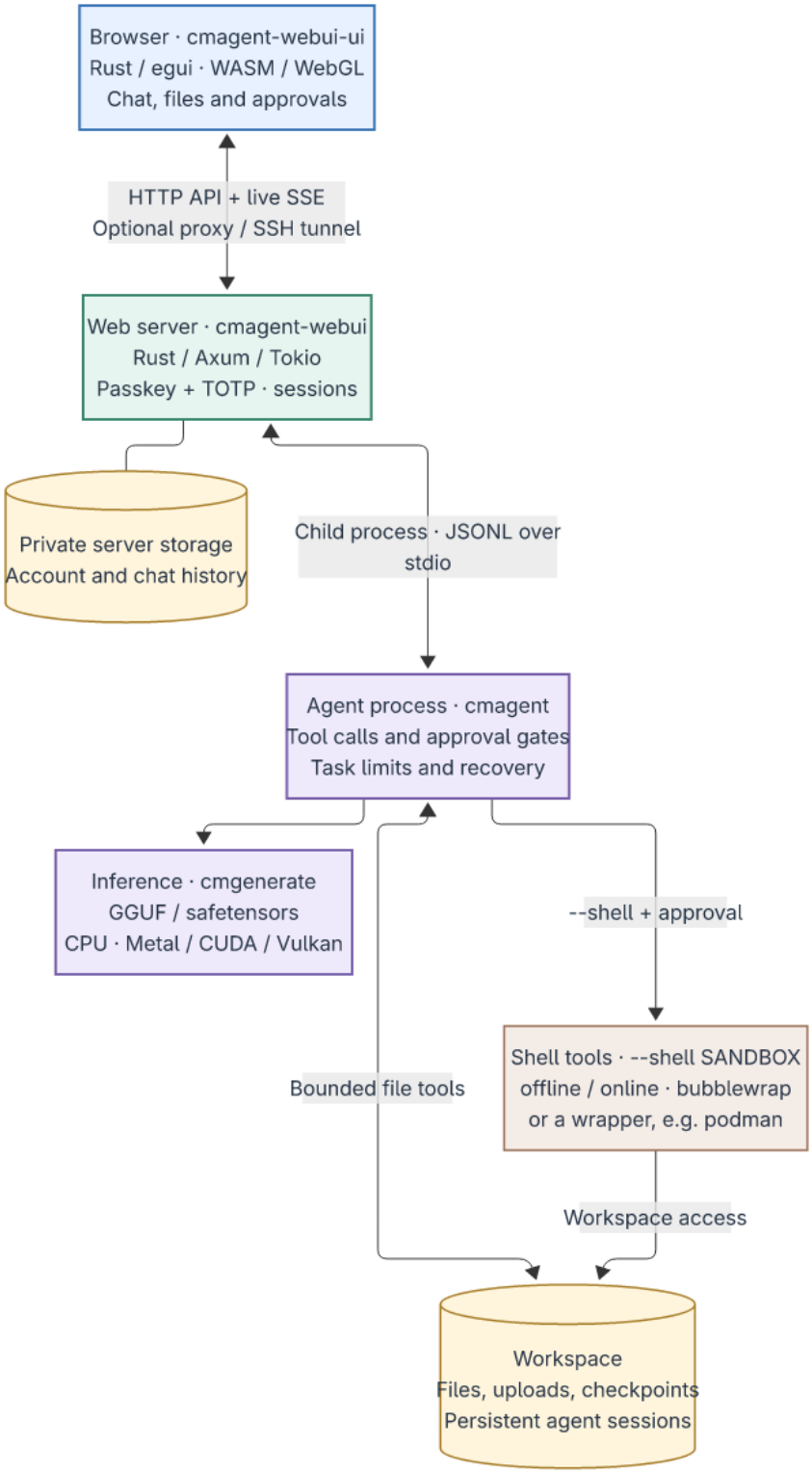


Figure 2: Web UI stack: egui/WASM browser connects to an Axum server over HTTP and SSE; the server manages authentication, chat storage, and a cmagent child process, which uses cmgenerate inference, workspace tools, and a shell in a named sandbox (bubblewrap offline or online, or a wrapper).

Agent web UI

An egui/WASM browser client with passkey + TOTP login, searchable chats and saved drafts, inline tool results and edit approvals, task progress, reviewed recovery, workspace previews, uploads, reconnecting live updates, and Hugging Face model downloads. Build assets once, then run:

```
./target/release/cmagent-webui model.gguf --gpu --write
```

Open **http://localhost:8787** and enroll with the terminal's setup token. See build and access setup.

Models and conversion

Supports dense and mixture-of-experts transformers, plus supported recurrent hybrids. Compatibility depends on the model's tensors and architecture, not just its file extension. Unsupported mechanisms are reported during loading.

Apertus GGUF and original HuggingFace safetensors models use an ungated xIELU feed-forward with per-layer coefficients and Apertus chat formatting. GPU offload uses device matrix operations with the activation on the CPU for CUDA and Metal; Vulkan also runs the xIELU activation on the GPU. The fused Metal pass does not yet support xIELU.

Muse Glimmer GGUF (`meta-models/Muse-Glimmer-30B-GGUF`) runs with its gated attention, three-local-one-global layer pattern with no position in the global layers, normalised embedding, scaled and softcapped head, and its ATEM chat template, in which the model names its own recipient (`to=self` while it reasons, `to=user` for the answer). The DFlash drafter and the vision projector in the same repository are not used. HuggingFace checkouts of it run too, including unsloth's bitsandbytes 4-bit one.

DeepSeek-V4.1 (`deepseek-ai/DeepSeek-V4.1-Flash-1017`, the HuggingFace checkout as published: `F8_E4M3` matrices and four-bit experts under power-of-two `F8_E8M0` scales) runs on the processor through an engine of its own: one shared key-and-value latent a token attended over a window of 128, the rest of the sequence compressed by a few layers and chosen from by a two-level indexer, hyper-connections, a grouped low-rank output projection, the `sqrtsoftplus` router and the engram n-gram tables. The activation quantisation the reference implementation keeps from training is kept. Its chat format follows the checkout's `encoding.py` (`--effort none` turns reasoning off, `low` is 50, the default 75). Not read: images, the DSpark drafting layers, `DSML` tool calls; a session's prefix is not saved. `--gpu` on CUDA puts the dense matrices on the first card and loads the routed experts in use into the rest of every card, in the file's formats (`CMGENERATE_EXPERT_CACHE=0`: experts by layer order instead). unsloth's GGUF of it (`deepseek41`, a split file: give any shard or the folder) runs through the same engine; its dense matrices are `Q8_0`, and its logits are 8–18% from the checkout's. Its logits are checked against the checkout's own `inference/model.py` (`reference/deepseek_v41`). The checkout is 475 GiB and is mapped, so it wants that much memory to run at speed: 10 tokens a second on two EPYC 9754 (`--threads 64`), 12 with two A10 cards.

Images: Qwen3.5 and Qwen3-VL models read them through the llama.cpp `mmproj` GGUF from the same repository (`clip.projector_type = qwen3vl_merger`, the 27-block tower with the 2×2 merger), found beside the model or given with `--mmproj`, or through a HuggingFace checkout's `model.visual.*` tensors: a checkout reads images by itself (Ornith 9B), and `--mmproj DIR` lends its tower to a GGUF written from it. GGUFs this crate wrote before 2026-10-13 lack `rope.dimension_sections` and read no images; write them again. An image is scaled to whole 32-pixel blocks of at most 1024 tokens (`CMGENERATE_IMAGE_TOKENS`) and placed with Qwen's interleaved M-RoPE. With `--gpu` the tower runs on the card as one job (CUDA; Vulkan with tensor cores): 0.05 s for 300 tokens and 0.28 s for 972 on an RTX 5070 Ti, against 3 s and 15 s on the processor. Metal multiplies its matrices only.

Muse Glimmer reads images through its `mmproj` (`muse-glimmer`: 50 blocks of 1536 over 14-pixel patches, windows of 32×32 patches, 2×2 patches a token into a three-matrix MLP). Images are scaled with Lanczos to whole 28-pixel blocks, at most 1024 tokens; each token takes one text position, framed by `<|image_start|>` and `<|image_end|>`. The tower runs on the processor, its matrices on the card with `--gpu` where there is room: 768 tokens in 13 s (CPU), 9.4 s (CUDA products).

Gemma 4 12B reads images through its HuggingFace checkout's unified vision (`gemma4_unified_vision`: no tower, one dense layer over 48-pixel patches; the text model does the rest): give the checkout with `--mmproj DIR`. 40 to 280 tokens (`CMGENERATE_IMAGE_TOKENS`), framed by `<|image>` and `<image|>`; an image's tokens see each other both ways, as llama.cpp decodes them, and are attended on the processor.

```
./target/release/cmgenerate gemma-4-12b-it-Q4_K_M.gguf --mmproj models--google--gemma-4-12B/snapshots/… --
```

```
./target/release/cmgenerate Qwen3.6-35B-A3B-Q4_K_M.gguf --gpu --image page.jpg "What is the headline?"
```

cmagent gives a task's images to the model with its first turn: each line `Attached file: PATH` naming an image in the workspace, as the web UI writes one per upload. The model also sees the line. Other projectors (Muse Glimmer, Gemma 4) and video are not read.

Container	Readable weight types
GGUF, including split files	F32, F16, BF16; Q8_0, Q4_0, Q4_1, Q5_0; Q2_K–Q6_K; IQ1_S/M, IQ2_XXS/XS/S, IQ3_XXS/S, IQ4_NL/XS; Q2_0 (group 128), BITCOS; MXFP4, MXFP8 (DeepSeek-V4.1)
Safetensors, including shards	F32, F16, BF16, F8_E4M3, NVFP4, bitsandbytes NF4; DeepSeek-V4.1's four-bit experts and F8_E8M0 scales
llama2.c	F32

bitsandbytes NF4 matrices are converted once to the `NF4` block (IQ4_NL's layout with the NF4 table, 4.5 bits a weight) and kept in `~/.cache/cmgenerate/nf4-<hash>.bin` (or under `XDG_CACHE_HOME`); later loads map that file. It is not a model: open the checkout, not the cache. Delete it to reclaim the space; it is rebuilt when needed.

Convert supported input to GGUF:

```
./target/release/cmgenerate model.gguf --quantise Q4_K -o smaller.gguf
```

The writer also supports `Q5_K`, `Q6_K` and `Q8_0`. Small vectors retain their precision; already quantized tensors are copied. Reading a weight format does not imply that every accelerator supports it.

Ternary models (Prism's Ternary Bonsai, in its `Q2_0` GGUF at group 128, ggml types 42 and 142) run on the CPU. `--quantise BITCOS` re-encodes one as a presence bitmap plus compacted signs, after Georganas, Heinecke and Dubey (arXiv 2609.16338): `2 - z` bits a weight for a zero fraction `z`, plus one f16 per 128, against `Q2_0`'s 2.125. Bonsai 8B goes from 2.0 GiB to 1.7 GiB. The sign runs are a second tensor beside each matrix, so only this crate reads the result. Vulkan and Metal run both ternary formats, Vulkan's `BITCOS` only where the device has 32-lane subgroups and integer dot products; CUDA leaves them on the processor, or takes them re-encoded under `--gpu-as`. On Metal a `BITCOS` tensor is copied into a buffer of its own rather than read where it lies, since its kernel wants the sign run after the blocks. A source that is not ternary at a scale per 128 is refused by name.

Current limits include text-only inference, no multi-token prediction, no NPU backend, and no concurrent sequence batching. Accelerator coverage varies by architecture and format. CPU prefill batches supported projections; decoding still generates one token at a time.

Development

```
cargo test --workspace
cargo clippy --workspace --all-targets -- -D warnings
```

- `src/format/`, `src/quant/`: containers, conversion, and weight formats.
- `src/model.rs`, `src/block.rs`, `src/attention.rs`, `src/delta_net.rs`: inference.
- `src/tensor.rs`, `src/simd.rs`, `src/threads.rs`: kernels and CPU execution.
- `src/session.rs`, `src/chat.rs`, `src/tokenizer/`: conversations and tokens.
- `crates/`: chat, server, and agent front ends.
- `examples/`: focused performance and diagnostic tools.