



Agentic AI Foundation

AGNTCon + MCPCon
North America

Self-Hosting Agents

Small dense models change the economics

Miriah Peterson · @Soypete

AGENTCON / MCP CON · OCTOBER 23, 2026



WHO I AM

Miriah Peterson

DAY JOB

Co-founder of Haikei
Labs

ON THE MIC

Co-host, **Domesticating**
AI podcast

SINCE 2022

Building AI systems

Haikei — *the background for intelligent action.*

What is an agent?



Agent AI Foundation

AGNTCon + MCPCon
North America

observe → decide → call a tool → inspect the result

↺ REPEAT UNTIL DONE ↻

A model running inside a loop.

Every pass through the loop is another inference.

What do agents enable?



Agentic AI Foundation

AGNTCon + MCPCon
North America

One-off

Completes a task while I wait

Background

Runs a workflow without me watching

Autonomous

Notifies, decides, acts, and recovers

Always on

Keeps doing the job whenever work appears

The value grows as the agent needs less of my attention.

Always means always



Agentic AI Foundation

AGNTCon + MCPCon
North America



The problem: "always on" is expensive for everyone



Agentic AI Foundation

AGNTCon + MCPCon
North America

YOU PAY

- Tokens on every turn
- Repeated agent loops
- Availability

THE PROVIDER PAYS

- Compute for every token
- Accelerators and memory
- Idle capacity and scheduling

Same incentive on both sides: less compute per token.

Open weights got good. Now where do they run?



Agent AI Foundation

AGNTCon + MCPCon
North America

Modal

serverless GPUs

RunPod

rented GPUs

Self-host

the 5090 in my house

Open-weight models are strong enough for real agent work.

You pay for compute, not tokens.

Choosing hardware: start with memory

UNIFIED MEMORY

CPU

GPU



One shared pool

simple to size: does the model fit?

- Lower bandwidth than discrete VRAM

Mac · DGX Spark · AMD Strix Halo

SPLIT MEMORY

GPU



VRAM

fast · small

spill
costs
speed ≠

CPU



System RAM

big · cheap

RTX series (my 5090) · datacenter GPUs

First question: does the model plus its KV cache fit, and where?

What memory is used for



Agentic AI Foundation

AGNTCon + MCPCon
North America

GPU + VRAM

Runs inference — in parallel, which is the speed

Holds the **KV cache**: the model's working memory for the conversation

KV cache
offload



more context,
slower

CPU + system RAM

- Run the inference engine
- Load the entire model
- Can run inference alone — just slowly

More on hardware: [Domesticating AI Ep 3 ·youtu.be/LZ0N6q6S1WY](https://youtu.be/LZ0N6q6S1WY)

Pick the engine and the model

vLLM

llama.cpp

Ollama

LM Studio

Hugging Face

Choose the model, then filter by what it can do: **multimodal** · **tool calling** · **reasoning**. That matters for our agents.

What the model and the engine do



Agentic AI Foundation

AGNTCon + MCPCon
North America

Inference

Turns weights into tokens: speed and context length.

Reasoning

Thinking tokens before the answer.

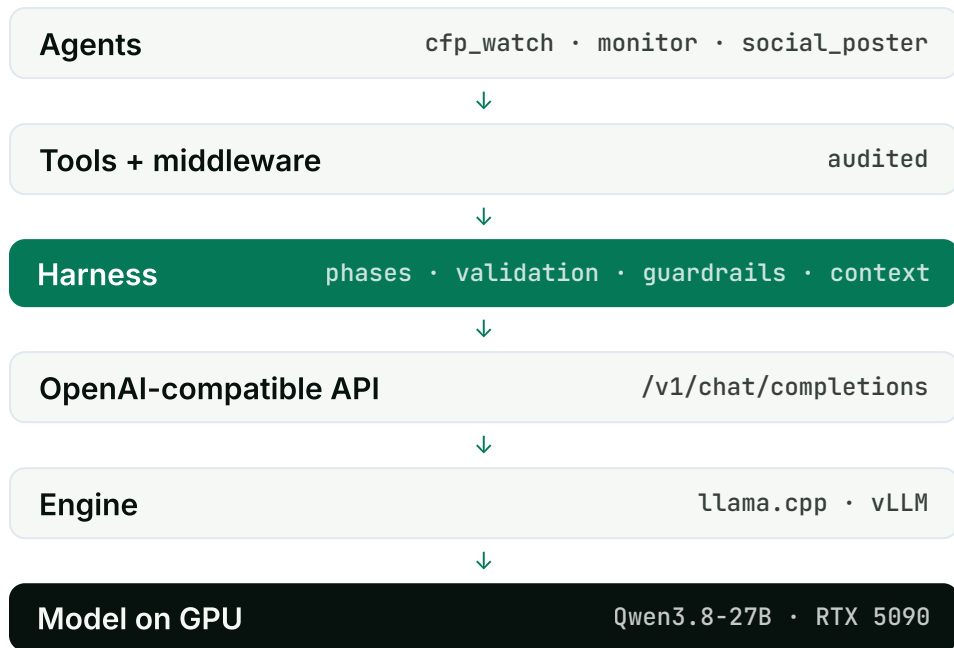
Tool calls

Structured calls the harness runs.

Model — what it can do: multimodal, reasoning, tool calling.

Engine — how fast, and whether those features actually work.

The stack, end to end



WHAT RUNS WHERE

Top layers run on cheap, always-on CPU: API, auth, queues, context assembly, scheduling.

Only the bottom two need the accelerator.

Swap the engine or the model and the agents never notice — everything meets at one API.

Start with MoE or a small dense model



Agentic AI Foundation

AGNTCon + MCPCon
North America

MoE — large local model

397B total
↓ route each token
17B active

- Capacity through MoE
- Weights spill into system RAM
- Fewer parameters active per token
- Maximize capability

Small dense model

27B total
↓ every token
27B active · all in VRAM

- Efficiency through dense weights
- Weights fit on one GPU
- Every parameter active per token
- Minimize cost per completed job

Choose for the workload — not for the leaderboard.

"We are kids in a candy store"



Agentic AI Foundation

AGNTCon + MCPCon
North America

"We are kids in a candy store."

– Chris Brousseau

► [Watch the short on YouTube](#)



AI Mode
Domestic...





Agentic AI Foundation

AGNTCon + MCPCon
North America

The second half: the harness

Reliability is the work on small models

What does "reasoning" mean here?



Agentic AI Foundation

AGNTCon + MCPCon
North America

interpret the request



choose an action or tool



observe the result



decide: continue, recover, or stop

↺ repeat

The harness asks the model to repeatedly choose the next step.

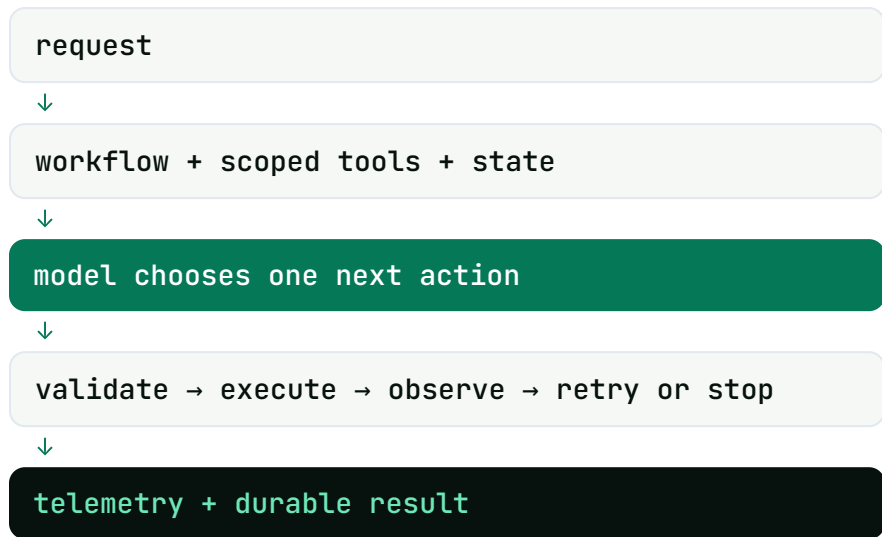
Ambiguity creates decisions before the real work even begins.

The harness is the agent system



Agentic AI Foundation

AGNTCon + MCPCon
North America



Owens the loop around the model: decomposition, visible tools, durable state, context, validation, retries, stopping, observability.

**The model supplies judgment.
The harness supplies reliability.**

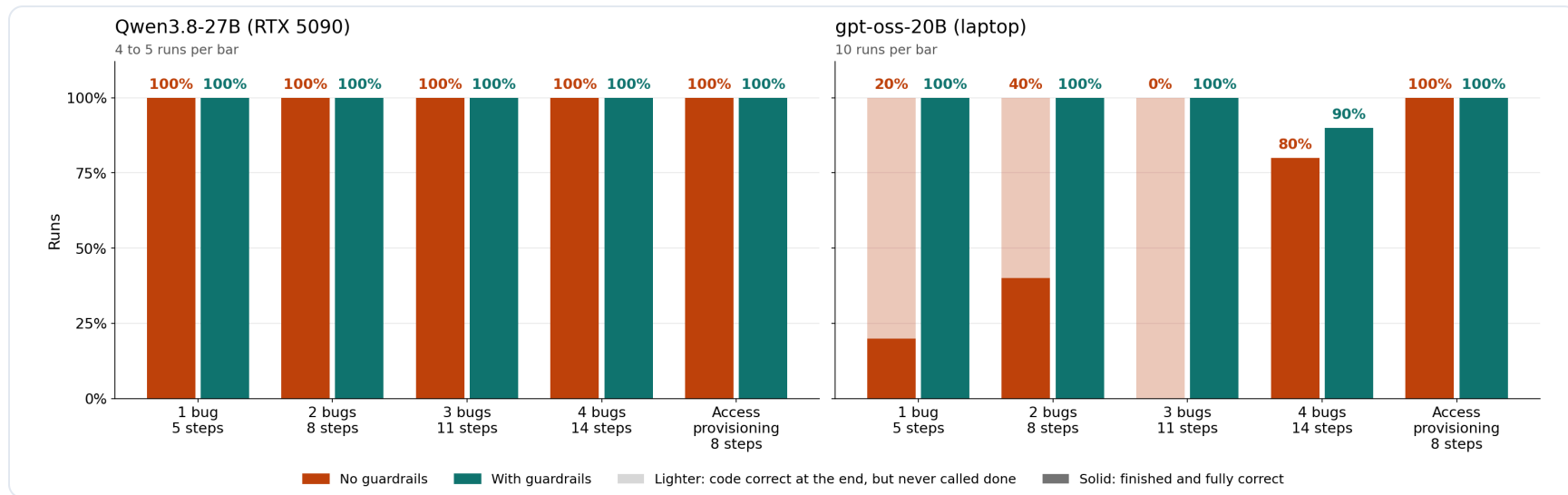
Forge: reliability outside the model

Antoine Zambelli's [Forge](#) adds five things around the model:

- | | | |
|-----------|--------------------------------|--|
| 01 | Response validation | catch unknown or malformed calls |
| 02 | Rescue parsing | recover calls from fences, XML, vendor formats |
| 03 | Corrective retry nudges | return the precise error, not a blind retry |
| 04 | Workflow enforcement | optional: stop repeated or premature steps |
| 05 | Context compaction | stop every turn carrying the whole past |

The model proposes the next action; the harness makes the loop survivable.

Guardrails change completion rates



Same tasks, same harness, bare vs. guardrails. Light bars: the code was right, but the run never called done.

Forge's original 8B results: [CAIS 2026 demo and paper](#)

Same task, same model: bare vs. guardrails



Agentic AI Foundation

AGNTCon + MCPCon
North America

```
Last login: Mon Oct 19 18:36:03 on ttys004  
Recording 35s to out/guardrails-demo.mov ...
```

Two runs recorded separately on one laptop, shown together at real speed: gpt-oss-20B, fix three bugs. No audio.

Why Forge does "crazy" stuff



Agentic AI Foundation

AGNTCon + MCPCon
North America

wrong format → rescue the call → or → explain the error and retry →

continue with compacted context

Rescue

avoids paying for
another inference

Nudges

prevent blind retries

Step enforcement

stops repeated or
premature actions

Compaction

stops every turn
carrying the whole past

Reliability guardrails are token controls.



Agentic AI Foundation

AGNTCon + MCPCon North America

Demo: one GPU, real agents

Qwen3.8-27B · RTX 5090 · my house

Demo: measure the local server



Agentic AI Foundation

AGNTCon + MCPCon
North America

```
cd ~/code/pedro/pedro-ops/scripts/pedrogpt/benchmark

go run . -url http://pedrogpt:8000/v1 \
  -model qwen3.8-27b -n 3 -max-tokens 128

go run . \
  -prompt "Write a Go HTTP server with graceful shutdown." \
  -max-tokens 512
```

TTFT

prompt tok/s

gen tok/s

p50 / p95

Benchmark the serving path, not the model card.

Demo: the run



Agentic AI Foundation

AGNTCon + MCPCon
North America

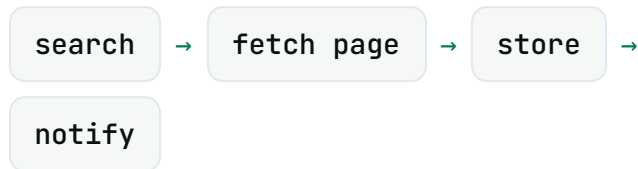


The first command, recorded live against the 5090 over Tailscale. No audio.

Tool calls: my research agents

```
base_url = os.environ.get(
    "LLAMA_CPP_BASE_URL", "http://.../v1"
)
model_name = os.environ.get(
    "LLAMA_CPP_MODEL", "qwen3.8-27b"
)

tools = [duckduckgo_search, fetch_cfp_page,
         store_cfp, send_discord_message]
```



Tool schemas in, tool call out, tool result back in — repeat until the job is done.

These run against the 5090 in my house.

Production serving



Agentic AI Foundation

AGNTCon + MCPCon
North America

vLLM

- NVIDIA throughput
- Continuous batching
- Scheduling + metrics
- OpenAI-compatible tools

llama.cpp

- Hardware portability
- GGUF quantizations
- Lightweight server
- OpenAI-compatible tools

Both speak the OpenAI tool-calling API — the harness never learns which one you picked.



Agentic AI Foundation

AGNTCon + MCPCon
North America

Back to the problem

Agents are expensive.

One response becomes many turns · one task becomes an always-on workload

Every turn has a token bill



Instructions and tool schemas

resent every turn

Conversation and workflow state

keeps growing

Tool results

return as input on the next turn

A ten-turn job repeatedly pays to reconstruct the model's working state.

Reliability multiplies the bill

bad call → error → retry → corrected call → verification

Rescue parsing

reuses output already generated

Corrective nudges

avoid blind retries

Compaction

reduces every remaining turn

One mistake can add several turns — or restart the job.

Reliability guardrails are also token controls.

When the economics work



Agentic AI Foundation

AGNTCon + MCPCon
North America

PROS

- Per-token rent becomes fixed infrastructure
- Well-defined jobs fit smaller, efficient models
- High utilization amortizes the GPU
- Good context engineering reduces inference and tokens

COSTS

- Idle GPU time is still your bill
- Model loading creates cold starts
- Scheduling, queues, memory and failures become your problem

Known job + reliable harness + high utilization = a compelling local case.

Compute cost is not token cost



Agentic AI Foundation

AGNTCon + MCPCon
North America

$\text{compute cost per job} = \text{GPU \$/second} \times \text{occupied seconds} \div \text{completed jobs}$

EXAMPLE CAPACITY

CURRENT RATE

RunPod RTX 5090 Pod

\$0.99 / GPU-hour

RunPod RTX 5090 Serverless

\$1.58 / active GPU-hour

Modal L40S

\$0.000542 / second $\approx$ \$1.95 / GPU-hour

Rates checked Aug 14, 2026 · runpod.io/pricing · modal.com/pricing

Another example: local models in CI/CD



Agentic AI Foundation

AGNTCon + MCPCon
North America

pull request / nightly



eval suites



pass rate

- The model is a dependency — test it like one
- Runs per model profile: a local 5090 and a hosted model
- Below threshold, the check fails

Same evals that picked the model now guard it.

agent-evals-report summary

✓ Agent evals (agent): passed

Scope: all suites (evals/model-profiles.yaml changed)

Threshold 95.0% of passed runs / total runs per suite per model · 3 run(s) per case · commit [b0c7ecc](#) · [run](#)

Suite	deepseek-v4-flash	qwen3.8-27b	Δ vs main
agentware.beta-tools	✓ 100.0% (72/72 runs)	✓ 100.0% (72/72 runs)	n/a

Deterministic table tests (EV-C1); no LLM grader. Δ vs main is not yet available: no baseline store.

[Job summary generated at run-time](#)

The takeaway

Find the smallest model that reliably completes the job.

01

Measure tokens, turns, runtime, and completion rate.

02

Reduce ambiguity with context engineering.

03

Put validation, rescue, retries, and compaction around the loop.

04

Keep the inference infrastructure utilized.

Optimize the system — not just the model.



Agentic AI Foundation

AGNTCon + MCPCon North America

Thank you

Miriah Peterson · @Soypete

Co-founder, Haikei Labs · Co-host, Domesticating AI

github.com/HaikeiLabs/Agentware

Agentware — Forge's reliability pattern in Go, Python and TypeScript



Agentic AI Foundation

AGNTCon + MCPCon

North America

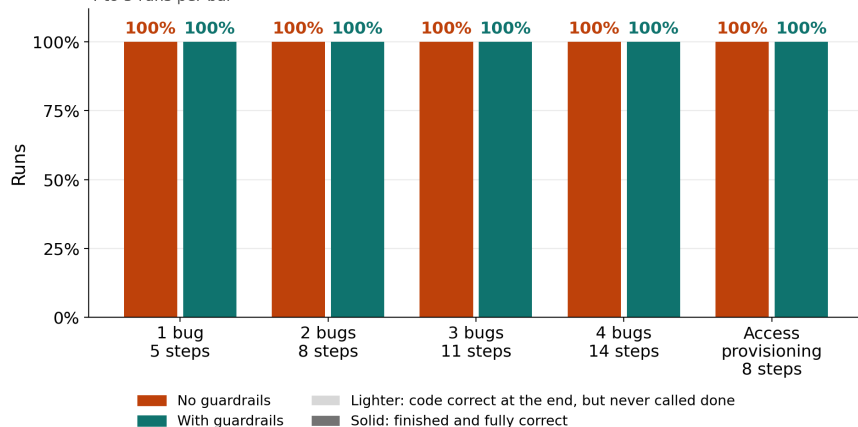
Appendix

Optional discussion after the outro

Per-model results

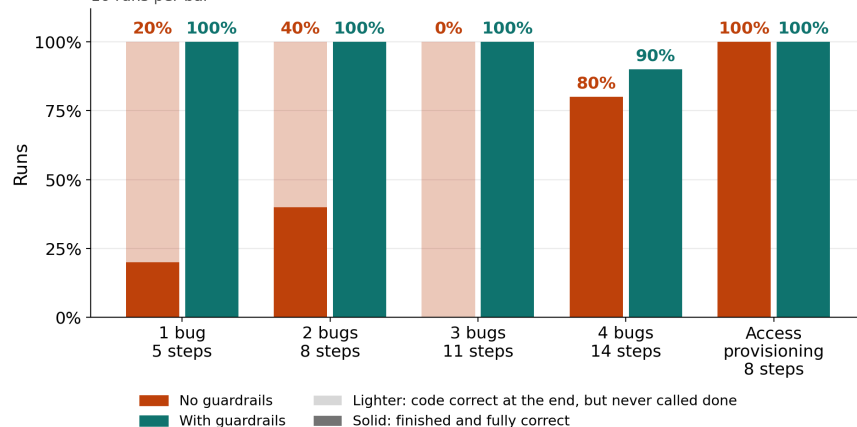
Qwen3.8-27B (RTX 5090): fixing failing tests and provisioning access

4 to 5 runs per bar



gpt-oss-20B (laptop): fixing failing tests and provisioning access

10 runs per bar



Qwen3.8-27B finishes everything either way. gpt-oss-20B: the code was right, but it never called done without the nudge.

Term to know: MoE



Agentic AI Foundation

AGNTCon + MCPCon
North America

More capacity, less compute per token. A router selects a few experts instead of using every parameter.

397B total parameters

↓ route each token

17B active parameters

- Local hosts can keep inactive experts in system RAM and move selected experts through the GPU.

Picking a model: total parameters sets memory; active parameters sets speed.

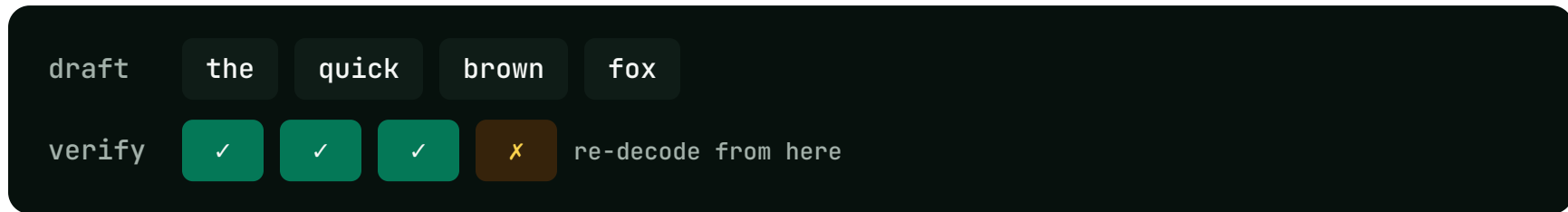
Term to know: MTP



Agent AI Foundation

AGNTCon + MCPCon
North America

Multi-token prediction: the model drafts several tokens ahead, then verifies them in one pass.



Acceptance rate is the whole story. High: tokens nearly free. Low: you paid to draft and threw it away.

Picking a model: does it ship MTP, and does your server use it?

Pedro CLI: I rolled my own harness



Agentic AI Foundation

AGNTCon + MCPCon
North America

```
type Phase struct {  
    Name           string  
    SystemPrompt   string  
    Tools          []string  
    MaxRounds      int  
    Validator      func(*PhaseResult) error  
}
```

Each phase scopes the prompt, tools, budget, and definition of done.

Context is an operational resource



Agentic AI Foundation

AGNTCon + MCPCon
North America

save prompts + responses + tool calls + full tool results



at 75% of the context window



summarize old rounds • keep recent rounds • measure the reduction

- Truncate what goes over the wire
- Preserve full tool output for retrieval
- Record tokens before and after compaction

Context management is part of the runtime — not prompt decoration.

Observe and evaluate the whole job



Agentic AI Foundation

AGNTCon + MCPCon
North America

job → phases → rounds → inference + tool calls

- Prompt, completion, and total tokens
- Model latency and tokens per second
- Tool calls, failures, rounds, and exit reason
- Completion score across repeated trials

The unit of reliability is the completed workflow.

Where this goes next: LoRAs



Agent AI Foundation

AGNTCon + MCPCon
North America

task + context + tool calls → successful result

Long-running agents create labeled trajectories.

- Specialize the model for the repeated job
- Improve accuracy
- Reduce context shipped per request
- Toggle adapters without another always-on model

Keep the control plane on CPU



Agent AI Foundation

AGNTCon + MCPCon
North America



Use inexpensive, always-on CPU for:

- API, authentication, and routing
- Queues, context assembly, and scheduling
- Health checks and GPU lifecycle

GPU scheduling is hard



Agentic AI Foundation

AGNTCon + MCPCon
North America

- Model weights and KV cache compete for VRAM
- Batching improves throughput but adds request latency
- Queues require priorities, limits, and backpressure
- Failures can strand work or GPU capacity

A fast model does not make a reliable serving system.

If asked: what about model quality?



Agentic AI Foundation

AGNTCon + MCPCon
North America

Model-quality comparisons are not the focus of this talk.

Can the smallest reliable model complete **your defined job at the utilization and cost you need?**

From one-off to always on



Agent AI Foundation

AGNTCon + MCPCon
North America

prompted once



scheduled



event-driven



continuously available

An always-on agent needs:

- Durable state between runs
- Tools it can call without supervision
- Recovery when a step fails
- Inference whenever new work arrives

The agent stops being a feature and becomes infrastructure.



HAIKEI LABS · AGENTWARE

Forge's pattern, ported beyond Python

native tool calling

validation · rescue ·
retries

context compaction

→ Go · Python · TypeScript

The same harness idea around whichever agent framework you already use.

github.com/HaikēiLabs/Agentware

Running it: the monitor agent



Agent AI Foundation

AGNTCon + MCPCon
North America

```
from pedro_agentware.middleware import (  
    InMemoryAuditor, MiddlewareImpl  
)  
  
mw, auditor = build_middleware()  
tools = apply_middleware(tools, mw)
```

Every tool call goes through one wrapper — schemas preserved, decision and result recorded.