LLM
SEMANTIC ROUTER

CONTRIBUTOR GUIDE

Build the open decision layer for models and compute.

How vLLM Semantic Router works, where you can help, and how to grow from your first pull request to maintainer.



Edition 1 · November 2026
github.com/vllm-project/semantic-router
vllm-sr.ai

WELCOME

Intelligence beyond any one model.

vLLM Semantic Router (vLLM-SR) is an open, programmable decision layer for models and compute. It sits between AI applications and the models they call, understands each request, and selects or combines the right models on the right compute, guided by your policy. It is built in the open, inside the vLLM project, by a fast-growing community. This guide shows how it works and how you can help build it.

6,493	290	1.99M	125
GitHub stars	contributors	downloads	open models on Hugging Face

As of November 17, 2026. Downloads combine Hugging Face, PyPI and container pulls.

CONTENTS

01 **What vLLM-SR is** — The problem and the goal

02 **Architecture** — From one request to the right model

03 **Module design** — The routing pipeline and the repository

04 **Two model families** — Vela and Decision

05 **Work Groups** — Seven owners, one system

06 **Directions by priority** — What each Work Group builds next

07 **Join a Work Group** — Apply, claim, deliver

08 **Grow with the project** — Committer and Maintainer

09 **Get in touch** — Links and community

THREE WAYS TO START

Try it

Pick an example at `app.vllm-sr.ai`. It runs live in your browser, with a free key on first run.

Run it

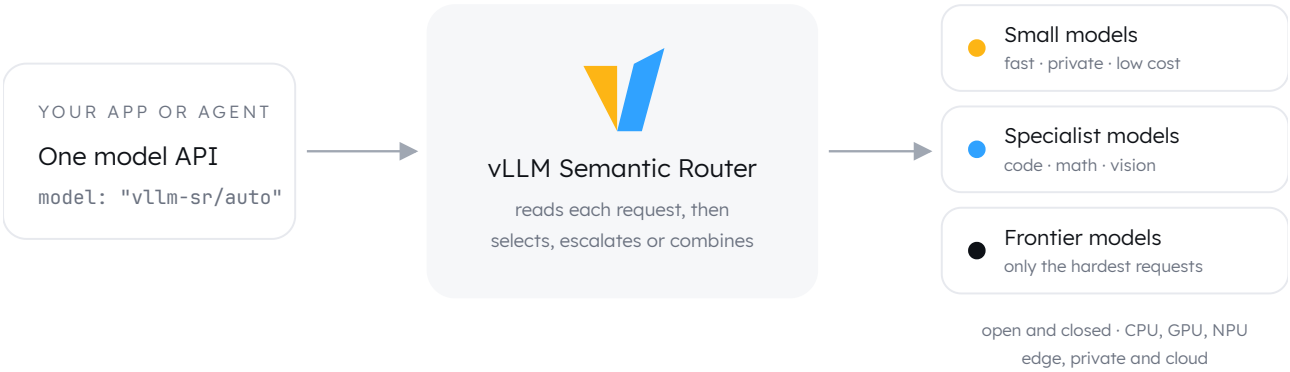
Install with the script at `vllm-sr.ai/install.sh`, then start the local stack with `vllm-sr serve`.

Build it

Find your Work Group on page 9, claim an AVAILABLE issue and open a focused pull request.

One model API.
A Mixture-of-Models behind it.

AI is moving from one model to many: large and small, open and closed, general and specialized, on CPUs, GPUs and NPUs, at the edge and in the cloud. No single model is best on quality, latency, cost and privacy at once. vLLM-SR gives applications one model API and decides, request by request, which intelligence to use.



MODELS

Different models excel at different tasks.

Select or combine models.

COMPUTE

Hardware varies in speed and capacity.

Choose among configured backends.

LOCATION

Edge, private and cloud.

Keep calls within approved locations.

PREFERENCE

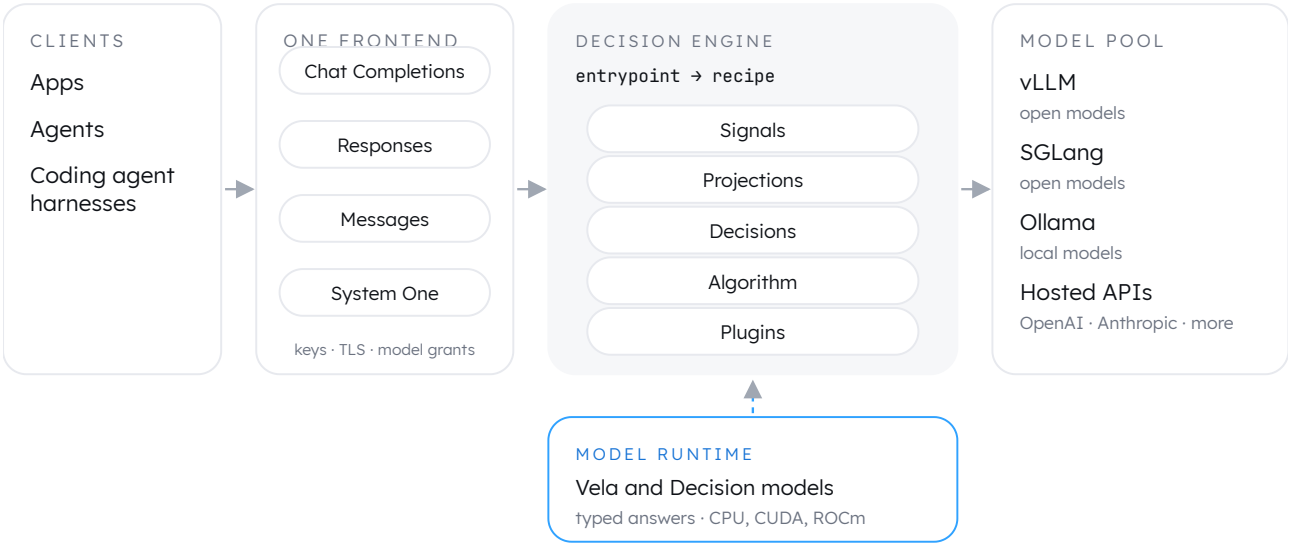
Priorities change by task.

Set quality, latency and cost priorities.

> **OUR GOAL**
>
> Match each request to the right model on the right compute: better answers, lower cost, enforced safety. Routing is the mechanism; the composite model is the product.

From one request to the right model.

Every client speaks a protocol it already knows. A public model name, such as `vllm-sr/auto`, resolves to an isolated recipe. The decision engine asks the decision models typed questions, applies the recipe's policy and sends the request to the chosen backend.



RUN IT ANYWHERE

Standalone, or behind Envoy-based gateways

Docker with the vllm-sr CLI · Helm · Kubernetes Operator

Router mode for routing · Engine mode for decisions only

OBSERVE AND IMPROVE

Every decision is traceable and measurable

Router Replay · metrics and traces · Dashboard

sr-bench evaluation · Router Learning

Standalone or behind a gateway

By default the Router serves the OpenAI-compatible listeners itself. It can also run as an Envoy ExtProc behind Envoy Gateway, Agent Router, Istio or KServe.

Router mode or Engine mode

`vllm-sr serve --config config.yaml` runs routing and System One. `vllm-sr serve MODEL --engine` serves typed decisions alone.

It complements serving

vLLM-SR decides what to run. vLLM, llm-d, Dynamo and providers decide how and where. It works with any open or closed model.

Six steps, one recipe.

A recipe is a versioned, portable description of how one model name behaves. Inside it, every request passes through the same six steps, each with one job, so a route stays readable, testable and easy to change.

1 **Signals**
EVIDENCE
One System One call answers the questions a recipe asks; cheap heuristics need no model call.

2 **Projections**
MEANING
Weigh and combine signals into scores and bands, such as reasoning effort.

3 **Decisions**
POLICY
Boolean rules with priorities pick the route; the first match wins.

4 **Algorithms**
CHOICE
Selectors pick one model; executors escalate, compare or run a workflow.

5 **Plugins**
ACTION
Mask, guard, cache, compress context and record each decision.

6 **Models**
RESULT
One response from the chosen model path, with a full trace.

Today: one System One primitive with 11 application signals and 7 heuristics; 9 algorithms, as 4 selectors and 5 executors.

Where the code lives

One repository: `github.com/vllm-project/semantic-router`. Each directory has an owner file, and every change links to one Work Group.

`src/semantic-router`
Router
Go router: ExtProc and standalone runtime, routing, APIs

`src/model-runtime`
Model runtime
Serves every model the Router uses, on CPU and GPU

`src/vllm-sr`
CLI
Python CLI and local stack orchestration

`src/agent-harness`
Harness plugins
Decision models inside Claude Code, Codex and VS Code

`config`
Configuration
Canonical config, fragments, schemas and maintained recipes

`dashboard`
Dashboard
React frontend and Go management backend

`deploy`
Deployment
Helm chart, Kubernetes Operator and deployment artifacts

`e2e · bench · perf`
Quality
End-to-end profiles, sr-bench evaluation, performance baselines

`tools`
Tooling
Build, development, release, security and agent tooling

`website`
Website
Public documentation at vllm-sr.ai

Open decision models, built for routing.

Decision models read a request once and answer typed questions with calibrated probabilities, in one forward pass and without generating text. That makes each decision fast and cheap enough to run on every request.

ROUTING-SPECIALIZED

Vela

Vela 3.0 · six open models · built with KR Labs

Trained for the Router's own judgments, such as domain, complexity, safety and tool choice. One call returns the decisions and the exact spans you name: personal data, unsupported claims, evidence, entities and relations.



vela3-fast: the Router's default, runs on a CPU

text and images

GENERAL MULTIMODAL

Decision

Decision 3.0 · the d3 family · 0.6B to 27B

Answers any classification question about a request or an agent step zero-shot, on text, images and video. The newest 27B model, d3-ultra, can also point: a box for each answer and a track through a video.



native in vLLM, llama.cpp, Ollama and Transformers

One API for both: System One

The Router asks these questions to route. Applications can ask them directly, with the same API.

Choice
one of N options, a probability for each

Yes / No
the probability of yes

Score
an expected level on a scale

Set
an independent probability per label

Span
the exact text behind an answer

```
POST /v1/systemone
{
  "model": "vllm-sr/d3-ultra",
  "state": "Our app has returned 500 errors since the 09:00 deploy.",
  "questions": {
    "route": {
      "type": "choice",
      "instructions": "Which team should handle this ticket?",
      "criteria": {"billing": "Payments and refunds",
                   "engineering": "Outages and bugs"}
    }
  }
}
```

Open weights under Apache-2.0 at huggingface.co/vllm-sr. Trained on AMD Instinct GPUs. Try both families free at app.vllm-sr.ai.

Seven Work Groups, one system.

A routed request crosses several responsibilities. Each Work Group owns one durable technical direction and its bounded Epics, so you can see where your work belongs and who you build with. A Work Group is a technical home, not a permission level.

ALONG THE LIFE OF ONE REQUEST

Developer Experience & Ecosystem	provides the CLI, Dashboard, APIs, recipes and learning path
Enterprise & Environment	secures management surfaces and owns lifecycle, capacity and deployment
Router Models & Inference Runtime	produces the signals used to route
MoM & Routing	chooses models and multi-model strategies
Agentic & Context	manages bounded context, memory and session continuity
Data Plane & Networking	executes the chosen path
Evaluation & Quality	measures results and catches regressions

HOW WORK IS ORGANIZED

Direction → Work Group → Domain → Bounded Epic → Accepted issue

Every piece of project work has exactly one Work Group owner. A bounded outcome is an Epic; the implementation issues under it are the tasks you can claim. Dependencies on other groups are recorded as shared interfaces in each group's charter.

EVERY WORK GROUP HAS

A charter
Goal, scope and the interfaces it shares with other groups

An Epic map
Bounded outcomes, split into issues you can claim

Leads
Keep tasks contributor-ready and coordinate triage

A label
`wg/<slug>` on every issue and pull request it owns



wg/mom-routing

MoM & Routing

Build measurable and continuously improving model pools, recipes, and multi-model routing.

OWNS

- Model pools, recipes and their lifecycle
- Multi-model collaboration algorithms and strategies
- Model-pool optimization and cross-model efficiency

P0 STARTING NOW

- Cache-aware model selection



wg/router-models-inference-runtime

Router Models & Inference Runtime

Improve Router Models and provide an extensible inference runtime for the model ecosystem.

OWNS

- Built-in model post-training and release
- Router-native model families beyond BERT
- Self-improvement, fine-tuning and runtime contracts

P0 STARTING NOW

- Faster decisions on more hardware
- Correct guard coverage
- Next-generation Vela
- Coding-agent decision model



wg/data-plane-networking

Data Plane & Networking

Run a portable and reliable online request path across standalone and gateway-integrated deployments.

OWNS

- OpenAI-compatible standalone data plane
- Envoy ExtProc, gateways and networking integrations
- Performance, streaming, dispatch, retries and telemetry

P0 STARTING NOW

- Request-path overhead, published and cut
- Load balancing with per-replica cache state



wg/enterprise-environment

Enterprise & Environment

Deliver production-grade security, operations, and deployments across supported environments and hardware.

OWNS

- Management authentication, authorization, identity and audit
- Insights, production observability and capacity planning
- Stable deployment APIs, reference stacks and hardware support

P0 STARTING NOW

- Health status and rollout alerts

wg/agentic-context

Agentic & Context

Optimize bounded context, memory, and session continuity for long-running and agentic workloads.

OWNS

- Context optimization, prompt-visible memory and session state
- Session budgets, tool-loop continuity and safe model switching
- Agent-aware boundaries for external runtimes

P0 STARTING NOW

- Measure cache reuse in agent sessions

wg/developer-experience-ecosystem

Developer Experience & Ecosystem

Make vLLM Semantic Router easy to adopt, configure, extend, diagnose, and contribute to.

OWNS

- First-run CLI, configuration, recipes, errors and troubleshooting
- Dashboard workflows on canonical Router contracts
- Agent-assisted workflows, docs, localization and ecosystem guides

P1 UP NEXT

- An upgrade guide for every release
- Dashboard editors follow the schema

wg/evaluation-quality

Evaluation & Quality

Make every supported capability measurable and every change verifiable.

OWNS

- Decision-level routing and first-class MoM evaluation
- Performance coverage, baselines and hardware qualification
- CI, behavioral E2E, compatibility, security and regression gates

P0 STARTING NOW

- Offline replay with decision diffs
- System-level evaluation on agent tasks
- Release checks for decision models

FIND YOUR WORK GROUP

Start from what you enjoy building.

- **Models and inference** — Router Models & Inference Runtime
- **Routing and recipes** — MoM & Routing
- **Gateways and protocols** — Data Plane & Networking
- **Production and security** — Enterprise & Environment
- **Agents and memory** — Agentic & Context
- **Tools and docs** — Developer Experience & Ecosystem
- **Benchmarks and CI** — Evaluation & Quality

What each Work Group builds next.

Ranked by the community in the roadmap survey for 2026 Q4 to 2027 Q1, then checked against what is already done. Each line is a direction that becomes Epics and claimable issues.

WORK GROUP	P0 NOW	P1 NEXT	P2 LATER
MoM & Routing	Cache-aware model selection	Multimodal MoM with vLLM-Omni System One and Two hand-offs (research) Cache-miss attribution	recipes.vllm-sr.ai Hardened multi-model collaboration Cross-model KV reuse (research)
Router Models & Inference Runtime	Faster decisions on more hardware Correct guard coverage Next-generation Vela Coding-agent decision model	Next-generation Decision models Calibrated thresholds for every model Reproducible training Images and video in signals Prefill-only inference (research)	Benchmarks on every model page Computer-use decision model Prompt injection in images (research) Speech routing (research)
Data Plane & Networking	Request-path overhead, published and cut Load balancing with per-replica cache state	vLLM-Omni speech, video and audio APIs	Remaining protocol gaps Per-model timeouts
Enterprise & Environment	Health status and rollout alerts	Compatibility and deprecation policy Cache cost accounting	Config preview and dry-run OIDC sign-in and append-only audit GPU reference stacks
Agentic & Context	Measure cache reuse in agent sessions	Model-switching studies Harness support table Context budgets per tokenizer Production Router Memory Credentials kept out of context	Agent-session datasets Decision models for memory (research)
Developer Experience & Ecosystem	None this quarter	An upgrade guide for every release Dashboard editors follow the schema	Good first issues with mentors Integration guides with results Evaluation skills
Evaluation & Quality	Offline replay with decision diffs System-level evaluation on agent tasks Release checks for decision models	Trustworthy CI signals	Public safety comparison

P0 is the focus now. P1 comes next. P2 waits for an owner. Tell us what matters to you at form.vllm-sr.ai/roadmap.

07 JOIN A WORK GROUP

Four steps to your first Work Group.

You do not need permission to start, only a good first task. Claim one issue at a time, so ownership stays clear and others can find open work.

1

Apply

Choose a Work Group in the join guide, open its charter and leave a short application comment.

2

Claim

Pick an AVAILABLE issue owned by that group and comment /assign. It is yours once you appear under Assignees.

3

Deliver

Open a focused pull request linked to the issue and get a substantive contribution merged.

4

Join the roster

The Work Group operations run adds you through a reviewed roster PR. You are now a Member.

YOUR APPLICATION COMMENT

```
I'd like to join this Workgroup.
Role: Member
Background: <relevant experience>
Interested in: <areas you want to help with>
```

Nothing fits? Comment `Need task: wg/<slug>` on the weekly Workgroup Issues entry.

ROLES INSIDE A WORK GROUP

Applicant	Chooses one Work Group, applies on its charter, claims one AVAILABLE issue and delivers a focused PR.
Member	Has at least one substantive merged contribution and keeps building in the group's direction, with discussion and review.
Lead	Maintains the charter and Epic map, keeps tasks contributor-ready and coordinates triage. A Committer or Maintainer, or a contributor with a merged commit and a sponsor.

HOW AN ISSUE MOVES

needs-acceptance → accepted → ready-for-dev → in-progress → closed

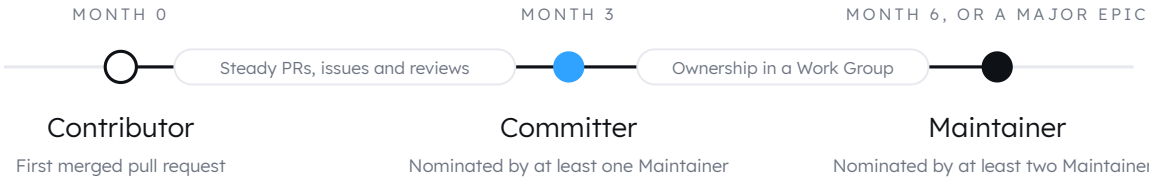
Maintainers accept work once its goal, scope and owning Work Group are clear. Good first issue and help wanted are curated subsets of ready-for-dev.

BEFORE YOU OPEN A PULL REQUEST

- Link an accepted issue with exactly one `wg/*` owner.
- Make one coherent change on a focused branch, with tests and docs.
- Sign every commit: `git commit -s`.
- Use the PR template and record the checks you ran.

From first pull request to maintainer.

Long-term contributors take on more ownership. The path rewards steady, visible work: code, reviews, issues, research and community.



START HERE

Contributor

WHO

- Anyone who contributes code, docs, research, issues or community work

HOW TO BEGIN

- Join a Work Group and claim one task
- Ship a focused pull request

→

TRUSTED REVIEWER

Committer

HOW TO GET THERE

- Contribute continuously for 3 months: pull requests, issues and PR reviews
- Be nominated by at least one Maintainer

RESPONSIBILITIES

- Review for correctness and triage issues
- Own accepted changes and mentor contributors
- Review and merge within trusted areas

→

PROJECT STEWARD

Maintainer

HOW TO GET THERE

- As a Committer, contribute continuously for 3 more months, or deliver a major Epic-level contribution in a Work Group
- Be nominated by at least two Maintainers

RESPONSIBILITIES

- Set engineering direction
- Approve releases and steward access
- Resolve escalations

WHAT COUNTS

Pull requests
Code, tests, docs and recipes that merge

Reviews
Substantive reviews of other people's pull requests

Issues
Clear reports, triage and design discussion

Epics
A Work Group direction delivered end to end

Two tracks, one community

Work Group roles (Applicant, Member, Lead) describe where you build. Project roles (Committer, Maintainer) describe the trust you hold across the repository. A Lead is a Committer or Maintainer, or a contributor with a merged commit and a sponsor.

Let’s build it together.

Open models, open systems, measurable intelligence. Pick a Work Group, claim a task and say hello. We review every first pull request with care.

Repository	github.com/vllm-project/semantic-router
Documentation	vllm-sr.ai
Work Groups	vllm-sr.ai/community/work-groups
Join guide	github.com/vllm-project/semantic-router/issues/15
Weekly issues	community.vllm-sr.ai
Roadmap survey	form.vllm-sr.ai/roadmap
Models	huggingface.co/vllm-sr
Try it	app.vllm-sr.ai
Chat	#semantic-router in the vLLM Slack
Meetings	Two community meetings a month, for APAC and the Americas

The repository

The Work Groups