
AI Safety Template¹

Creating standardized AI safety evaluations that run in a hardened & private way

A. David Giagnocavo
Independent Researcher

Kazuki Kimura
AI Safety Tokyo

Zane Estere Gruntmane
Independent Researcher

Siddharth Putta
Boston University

Gaurav Joshi
Independent Researcher

Satoshi Nakamura
Coinbase

With
Apart Research

Abstract

Post-compute, non-inference time AI safety (e.g. evals, benchmarks, automated red-teaming) lacks a unified standardized infrastructure suitable for international collaboration. Results aren't comparable across frameworks. Reproducibility depends on undocumented assumptions. Trust requires transparency that current tooling doesn't provide.

We built a working prototype for creating streamlined, standardized AI safety evaluations, benchmarks & automated red-teaming with a private, secure & isolated environment to run them. The architecture consists of a sealed controller that serves as the only external dependency for evaluation implementation, enabling streamlined development. This provides separation of concern by handling environment orchestration, air-gapped execution and isolated, private model interfacing & containerization separately from the evaluation code.

International AI safety collaboration needs common infrastructure that is collaborative, transparent, neutral, verifiable & privacy-preserving for sensitive IP. Not to replace existing initiatives, but to unite them. This template is a working step toward that mission.

¹ Research conducted at the [Technical AI Governance Challenge](#), 2026

1. Introduction

Post-compute AI safety work (evaluations, benchmarks, automated red-teaming) lacks standardized infrastructure for international collaboration.

Current approaches fail on five critical dimensions:

Inconsistency: Frameworks use varying approaches, metrics, formats - incompatible comparisons

Irreproducibility: No enforced versioning, fixed datasets, or controlled execution rules.

Transparency: Centralized images aren't auditable or neutral reducing multiparty trust.

Weak isolation: Docker shares host kernel; insufficient for untrusted/risky code execution.

Hidden behaviour: API-only access obscures true model behavior, making evaluations less reliable.

Threat model: Evaluations that cannot be trusted, reproduced, or compared across organizations undermine the foundation of international AI safety governance. The absence of trusted evaluation infrastructure creates a critical gap in international AI governance capabilities and impedes the political will and means for establishing international technically-enforced AI safety standards.

To address this issue, we propose AI Safety Template, a conceptual piece of large initiative, in which safety evaluations execute within a standardized, privacy-preserving, security-hardened two-container controller - template architecture that enforces reproducibility, auditability, versioning, code surface restrictions, and cross-organization comparability by design.

2. Related Work

Prior Work	Relevance	<i>Gap We Address</i>
HELM	Multi-metric holistic evaluation framework	No TEE-ready sandboxing of custom evaluation scenarios. Lacks a hardened controller architecture.
EleutherAI LM-Evaluation-Harness	Widely adopted, industry use (NVIDIA, Cohere, Mistral)	Less red-teaming/CCBRN support, simpler metrics over more comprehensive reporting. Lacks a hardened controller architecture.
HarmBench (UC)	510 behaviors, 18+ attack	Standalone - not integrated into standardized
AIR-Bench 2024	Regulation-aligned (EU AI Act), 314 risk categories	Requires GPT-4o judge, no reproducible offline execution. Not internationally scoped.

Docker containerization	Packages software to run identically	Doesn't enforce standardization, reproducibility, trust, or strong isolation
--------------------------------	--------------------------------------	--

Existing frameworks provide evaluation capabilities but **lack the engineering infrastructure for verifiable, standardized international collaboration**, which requires transparent execution, auditable code verification, and TEE-ready security hardening.

3. Methods and Results

i. AI Safety Template Design

The AI Safety Template separates the evaluation infrastructure into two isolated sandboxes connected by a private network:

Sandbox/TEE 1 — Evaluation Environment

- **UserComponent (Python):** A sandboxed, user-authored safety evaluation package. Restricted to dependency access only, with no direct network, filesystem, or external calls. Communicates exclusively through the Controller.
- **Controller (C/C++ interface with Python bindings):** The sole intermediary between the UserComponent and everything else. Validates the UserComponent, exposes a constrained set of allowed actions, wraps dependencies and datasets, mediates all interaction with the AI Assembly, and handles output to a standardized report format (JSON/XML).

Sandbox/TEE 2 — AI Assembly

Hosts the model under test and its inference runtime. This would be a local inference engine serving the raw model weights with minimal input and output modification that otherwise cannot be validated if the model was served through a standard API from the model owners' infrastructure.

This two-TEE design enforces three properties simultaneously:

1. The evaluation code cannot exfiltrate model weights or proprietary data
2. Evaluation execution is verifiable and reproducible
3. Enforces that model providers serve the raw model weights while preserving the protection of proprietary information.

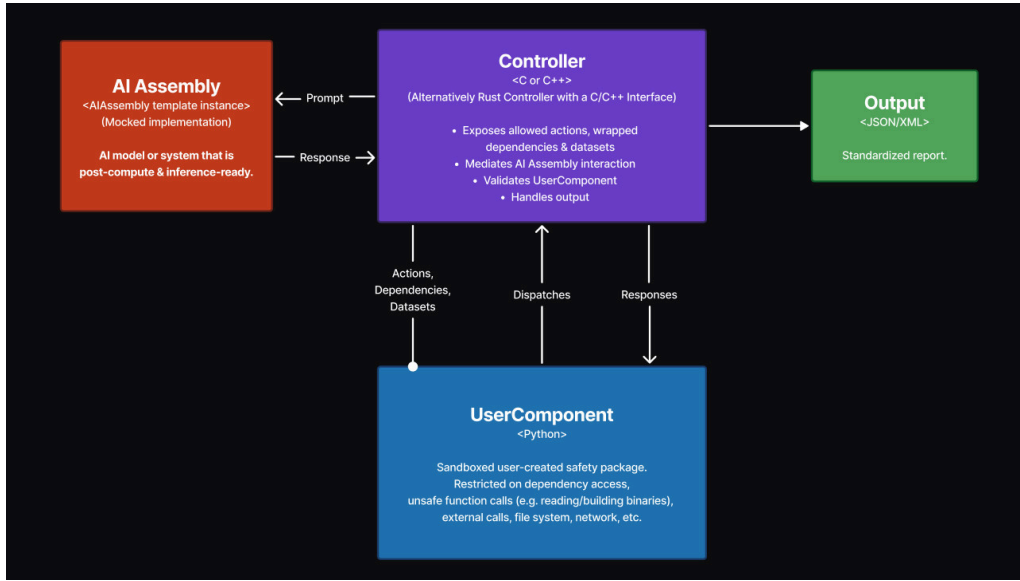


Figure 1

ii. Framework Selection

AI Safety Template will leverage a large existing body of eval frameworks to allow for rapid evaluation development. LM-Evaluation-Harness and Stanford HELM were systematically compared across sovereignty, adoption, red-teaming support, and multi-metric reporting. As the first framework to be supported, HELM (v0.5.12) was selected based on superior CCBRN coverage, red-teaming capacity, extendability, and scenario-based multi-metric reporting.

iii. LLM Interface Analysis and Selection

We investigated vLLM and llama.cpp/GGUF for reproducibility, TEE compatibility, and security hardening. We devised a hybrid strategy: llama.cpp/GGUF as the default inference engine for their deterministic quantization, TEE-compatibility, and offline-capability; vLLM for cases where high-concurrency (>10 evaluations) is required.

For simplicity and rapid prototyping, we opted to use the HuggingFace API to simulate a local LLM inference in a separate proxy container.

iv. Benchmark Selection and Dependency Mapping

In order to secure the sandbox, python dependencies and datasets are restricted and verified (not in the scope of the sprint). They were constructed based on 6 popular benchmarks (HarmBench, AIR-Bench 2024, MMLU, GSM8K, NaturalQuestions). We then produced a comprehensive dependency documentation which included runtime requirements, judge model needs, and installation workflows to prove the pattern. We selected HarmBench (red-teaming, 510 behaviors) as the audit-friendly evaluation suite to provide an example of use.

v. Prototype Design and Implementation

This project implements a user component (Python/HELM) that defines logic for an evaluation, which is run inside an isolated docker container. The sealed controller component (C++/Python) validates the user component, manages the execution environment, and handles model inference requests (done by routing requests to HuggingFace API). This architecture enforces reproducibility in a secure environment and enables TEE-ready isolation.

The prototype was able to execute HarmBench on Hugging Face SmolLM3-3B in a sandboxed environment with minimal Controller interface exposed to the evaluation code.

4. Discussion and Limitations

Limitations

- The proof-of-concept implementation lacks local inference of an LLM model and is dependent on a third-party inference provider;
- Since our prototype design heavily restricts Python execution for security purposes, it is not targeted at some types of evaluations, such as terminal use, code generation, or advanced cybersecurity.

Future Work

- Support local model inference (llama.cpp/VLLM) inside a separate sandboxed environment from evaluation code and Controller execution;
- Implement TEE (Trusted Execution Environment) integration;
- Add attestation layer to verify evaluation results and execution environment;
- Cover more comprehensive HELM evaluation suites.
- Add logging for performance metrics and standardize results format.
- Make the user experience more streamline with Controller utilities.
- Expand on the Controller security hardening.

5. Conclusion

This project establishes a standardized approach for running AI safety evaluations. By separating the evaluation logic from the model execution environment through a controller and sandboxing, the framework enables secure and reproducible benchmarking. This design also provides the foundation for building protected and auditable evaluations which addresses the need for transparency in evaluations.

Code and Data

Code Repository: <https://github.com/ais-zero/AISafetyTemplate>

References

1. **Confident AI.** (2023). *DeepEval: The open-source LLM evaluation framework [Software]*. GitHub. <https://github.com/confident-ai/deepeval>
2. **Gao, L., Tow, J., et al.** (2023). *A framework for few-shot language model evaluation*. Zenodo. <https://doi.org/10.5281/zenodo.10256836>
3. **Giskard AI.** (2023). *Giskard: The open-source testing framework for LLMs and ML models [Software]*. GitHub. <https://github.com/Giskard-AI/giskard>
4. **Liang, P., Bommasani, R., et al.** (2023). *Holistic evaluation of language models*. *Transactions on Machine Learning Research*. <https://openreview.net/forum?id=iO4LZibEqW>
5. **Lin, S., Hilton, J., & Evans, O.** (2022). *TruthfulQA: Measuring how models mimic human falsehoods*. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3214–3252. <https://aclanthology.org/2022.acl-long.229/>
6. **UK AI Safety Institute.** (2024). *Inspect: A framework for large language model evaluations [Software]*. GitHub. https://github.com/UKGovernmentBEIS/inspect_ai
7. **Wijk, H., Lin, T., et al.** (2024). *RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts*. *arXiv preprint arXiv:2411.15114*. <https://arxiv.org/abs/2411.15114>
8. **Zhang, Z., Lei, L., et al.** (2024). *SafetyBench: Evaluating the safety of large language models with multiple choice questions*. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. <https://aclanthology.org/2024.acl-long.0/>
9. **Perlitz, Y., et al.** (2024). *Efficient Benchmarking (of Language Models) [Conference paper]*. NAACL 2024. <https://aclanthology.org/2024.naacl-long.139.pdf>
10. **Stanford CRFM.** (2024). *air-bench-2024 [Software]*. GitHub. <https://github.com/stanford-crfm/air-bench-2024>
11. **Zeng, Y., et al.** (2024). *AIR-Bench 2024: A Safety Benchmark Based on Risk Categories from Regulations and Policies [Preprint]*. *arXiv*. <https://arxiv.org/abs/2407.17436>
12. **Center for AI Safety.** (2024). *mmlu dataset [Dataset]*. Hugging Face. <https://huggingface.co/datasets/cais/mmlu>
13. **Stanford CRFM.** (2025). *HELM Capabilities [Web page]*. Stanford CRFM. <https://crfm.stanford.edu/2025/03/20/helm-capabilities.html>
14. **OpenAI.** (2025). *gsm8k dataset [Dataset]*. Hugging Face. <https://huggingface.co/datasets/openai/gsm8k>
15. **VERL.** (2025). *GSM8K Example [Documentation]*. Read the Docs. https://verl.readthedocs.io/en/latest/examples/gsm8k_example.html
16. **Google Research.** (2019). *Natural Questions README [Documentation]*. GitHub. <https://github.com/google-research-datasets/natural-questions/blob/master/README.md>

17. **Kwiatkowski, T., et al. (2019).** *Natural Questions: A Benchmark for Question Answering Research* [Journal article]. *Transactions of the Association for Computational Linguistics*. <https://aclanthology.org/Q19-1026.pdf>
18. **Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020).** *Measuring Massive Multitask Language Understanding (MMLU)* [Preprint]. *arXiv*. <https://arxiv.org/abs/2009.03300>
19. **Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, Ł., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021).** *Training Verifiers to Solve Math Word Problems* [Conference paper]. *Advances in Neural Information Processing Systems (NeurIPS)*. <https://arxiv.org/abs/2110.14168>
20. **Kaiyom, F., Ahmed A., Mai, Yifan., Klyman, K, Bommasani, R., Liang, P., (2024).** *HELM Safety: Towards Standardized Safety Evaluations of Language Models*. Stanford University. <https://crfm.stanford.edu/2024/11/08/helm-safety.html>

LLM Usage Statement

We leveraged various consumer LLM applications such as Lumo, ChatGPT and Claude. To assist with technical brainstorming and implementations, Cursor and Claude Code were used. All claims were verified independently through trusted sources (e.g., official documentation), and all lines of code have been manually reviewed by the developers.

Template version 2.0 - January 2026