From Intent to Deploy: Validator-Centric Multi-Agent Orchestration for Reliable Infrastructure-as-Code

Rana Nameer Hussain Khan, Dawood Wasif, Jin-Hee Cho, Ali R. Butt
Virginia Tech, rnameerkhan@vt.edu

International Academy, Research, and Industry Association (IARIA) 2026
The Seventeenth International Conference on Cloud Computing, GRIDs, and Virtualization: CLOUD COMPUTING 2026

Presenter Resume

- Rana Nameer Hussain Khan
- PhD Researcher, Virginia Tech
- Focus: LLMs, Multi-agent Systems, IaC
- Experience: AI systems, Cloud, Program Synthesis
- Publication: MACOG (IARIA CC 2026)

Outline

- Motivation
- Problem Statement
- MACOG Overview
- Methodology
- Experiments & Results
- Discussions & Future Work

Motivation

Cloud infrastructure is:

- **Complex, Rapidly evolving, and Highly constrained**

IaC is essential, but:

- **Hard to write to write correctly**
- **Sensitive to schema and dependencies**

LLMs promise Natural language to IaC, but:

- **Syntax errors**
- **Policy violations**
- **Non-deployable configs**

Problem Statement

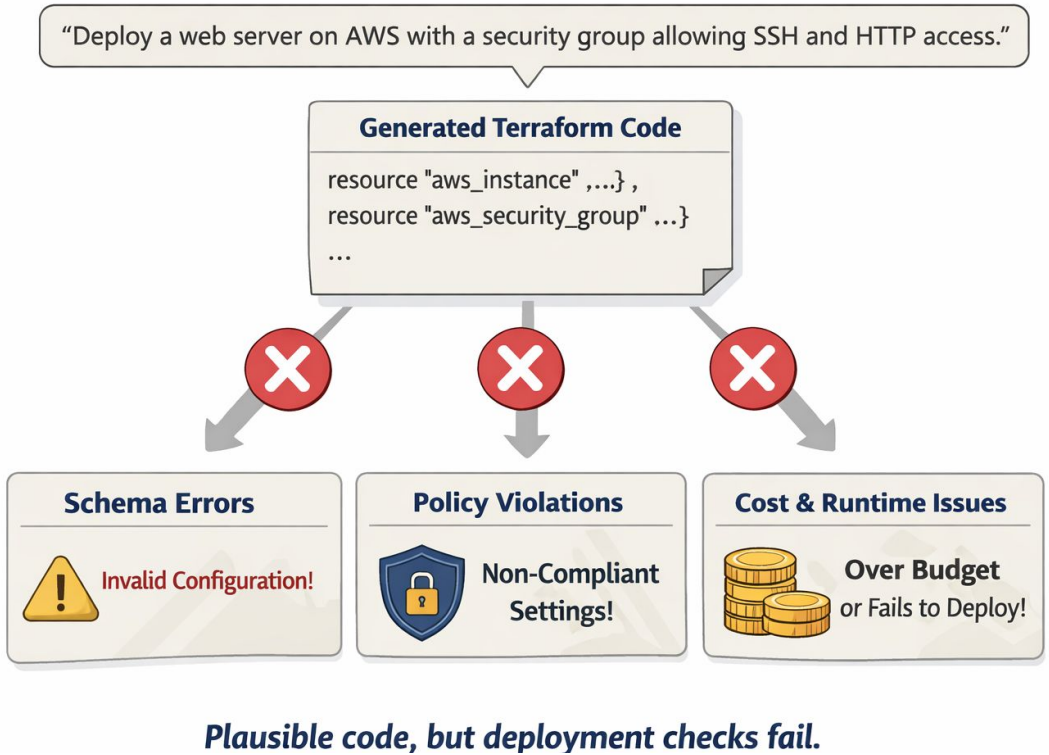
Our goal: **“Generate deployable, correct, and compliant Terraform from natural language”**

Challenges:

- **Scheme correctness**
- **Policy compliance**
- **Cost constraints**
- **Runtime deployability**

Existing approaches:

- **Few-shot, RAG, Iterative**



Proposed Approach: MACOG

Validator-Centric Multi-Agent System:

- Schema (Terraform)
- Policy Validation(OPA/Rego)
- Cost Estimation
- Deployment (plan/apply)
- Validators drive repair

Core idea:

- **Plan -> Generate -> Validate -> Repair -> Iterate**

Proposed Approach: MACOG

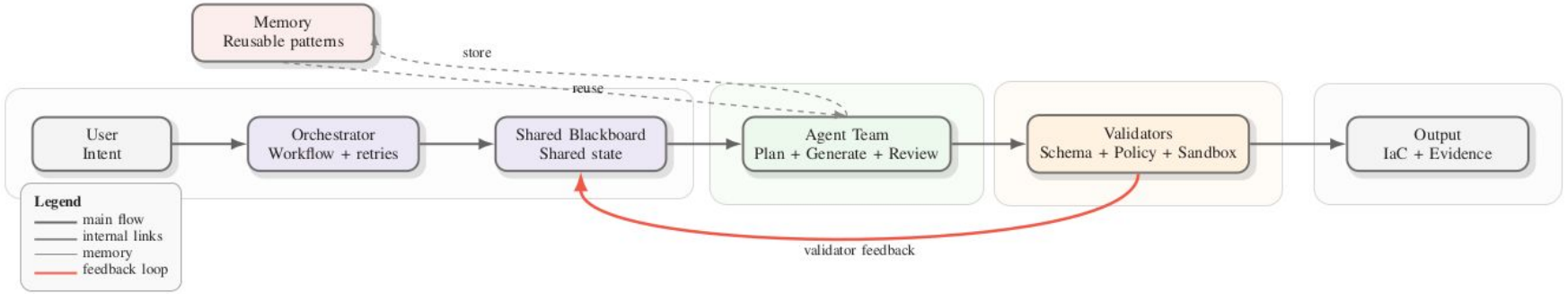


Figure 1. High-level MACOG architecture.

Proposed Approach: MACOG

Architecture:

- Multi-agent pipeline

Agents: **Architect, Harmonizer, Engineer, Reviewer, Security Prover, Cost Planner, DevOps, Memory Curator**

Core components:

- **Shared blackboard**
- **Finite-state orchestrator**
- **External validators**

Output: Terraform

Proposed Approach: MACOG

I-IR (Typed Representaion):

- Infrastructure Intermediate Representation

Graph: Resources, Dependencies, Constraints

P=⟨V,E,S⟩

V = resources

E = dependencies

S = constraints/effects

Key advantages:

- Explicit dependencies
- Typed schema enforcement
- Policy-aware planning

Experimental Setup

Benchmark: **IaC-Eval**

Models:

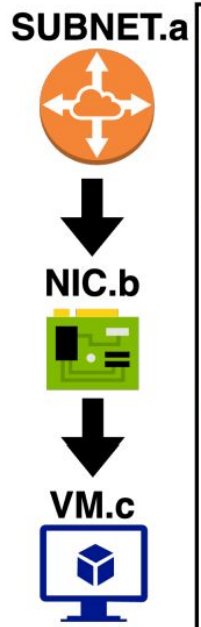
- **GPT-5, GPT-4, Gemini 2.5 Pro,**
- **Open source models**

Compared Strategies:

- **Few-shot**
- **CoT**
- **Multi-turn**
- **RAG**
- **MACOG**

Metrics:

IaC-Eval, BLEU, CodeBERTScore, LLM-judge




```
resource "SUBNET" "a" {
 CIDR      = "10.0.0.0/22"
}

resource "NIC" "b" {
 location  = "US-west"
 subnet_id = SUBNET.b.id
}

resource "VM" "c" {
 nic_ids   = [NIC.c.id]
 cpu_options {
    core_count = 2
  }
}
```

Results

MACOG outperforms in all baselines

- **GPT-5:**
  RAG -> 54.9
  MACOG -> 74.02
- **Gemini 2.5:**
  RAG -> 43.56
  MACOG -> 60.13

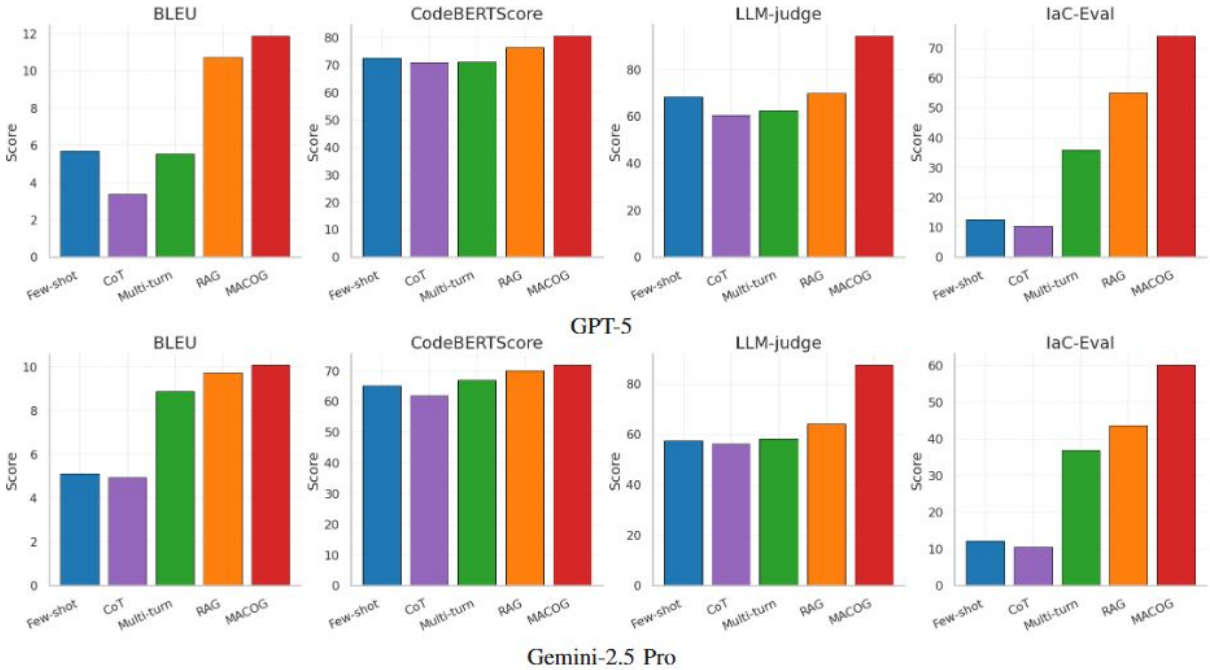


Figure 2. Metric profiles across enhancement strategies (BLEU, CodeBERTScore, LLM-judge, IaC-Eval).

Consistent gains across **Correctness, Semantic Quality and Deployability**.

Conclusion

Key insights:

- **Validators are essential**
- **Structured repair > free-form iteration**
- **Multi-agent decomposition improves reliability**

Future work:

- **Parallel validation**
- **Better policy coverage**
- **CI/CD integration**

MACOG enables:

- Reliable IaC generation
- Policy compliant infrastructure
- Deployable outputs

Key contribution:

- **Validator-centric multi-agent orchestration**

References

[1] P. T. Kon et al., “Iac-eval: A code generation benchmark for cloud infrastructure-as-code programs”, Advances in Neural Information Processing Systems, vol. 37, pp. 134 488–134 506, 2024.

[2] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, “A systematic mapping study of infrastructure as code research”, Information and Software Technology, vol. 108, pp. 65–77, 2019.

Thank you!