IntelliTopo: An IaC Generation Service for Industrial Network Topology Construction

Mingyu Shao[12], Zhao Liu[2*], Weihong Han[2], Cuiyun Gao[1], Jiachen Liu[1], Qing Liao[12*]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[2]Pengcheng Laboratory, Shenzhen, China
24b951080@stu.hit.edu.cn, liuzh08@pcl.ac.cn, hanwh@pcl.ac.cn,
gaocuiyun@hit.edu.cn, 200110207@stu.hit.edu.cn, liaoqing@hit.edu.cn

***Abstract*—Network topology construction in this paper refers to designing the structural layouts and configuration rules among network devices according to natural language requirements in network simulation. Relatedly, Infrastructure as Code (IaC) enables the configuration and management of network devices through machine-readable code. Although there exist IaC generation approaches powered by Large Language Models (LLMs), they only focus on generating isolated configurations without consideration for holistic topology structure, leading to failure to form a complete, functional topology. Additionally, due to the LLMs' limited knowledge of industry-specific device images, existing approaches struggle to adapt to diverse industry scenarios.**

In this paper, we introduce IntelliTopo, which, to the best of our knowledge, is the first IaC generation framework targeted at industrial network topology construction. Specifically, IntelliTopo enhances the capabilities of LLMs through two novel mechanisms: (1) Through *semantic topology parsing*, we enhance the LLMs' understanding of the holistic topology structure; (2) Through *domain-aware image retrieval*, the outputs of IntelliTopo are more aligned with real-world industry scenarios. Deployed on our PaaS system, the IntelliTopo service has operated continuously for 3 months, handling 50+ network simulation tasks across 10+ industries. IntelliTopo reduces average network topology deployment time from days to hours while requiring less computational power for LLM reasoning. This work bridges the gap between high-level requirements and executable infrastructure, providing a scalable solution for network topology construction.

***Index Terms*—Network Topology, Infrastructure as Code, Large Language Models.**

I. Introduction

Network topology construction is the foundation of network simulation, which has become indispensable in modern enterprise IT infrastructure, serving as the backbone for network architecture planning, security assessment, and critical scenarios such as disaster recovery testing and capacity expansion validation[1–6]. For instance, universities leverage network simulation to test multi-zone isolation and bandwidth allocation[1], while enterprises use it to verify VRRP gateway redundancy and VPN remote access configurations[4].

A cloud platform is a service-oriented layer built on top of IT infrastructure. Leveraging cloud platforms enables the simulation of real-world production environments to the greatest extent possible[7–9]. However, constructing network topologies via cloud platforms remains a significant challenge. This is largely due to the manual nature of traditional workflows, where network engineers must spend most of their time translating high-level requirements (e.g., "isolate payment processing subnets") into low-level device settings (e.g., firewall rules, VLAN configurations). The manual construction of network topology generally involves two processes: i. Requirement analysis and resource allocation; and ii. Configuration and verification. As shown in Fig. 1, these steps are not only time-consuming but also error-prone. In a production environment, it may take days for a skilled network engineer to set up an enterprise scenario with dozens of nodes on a cloud platform.

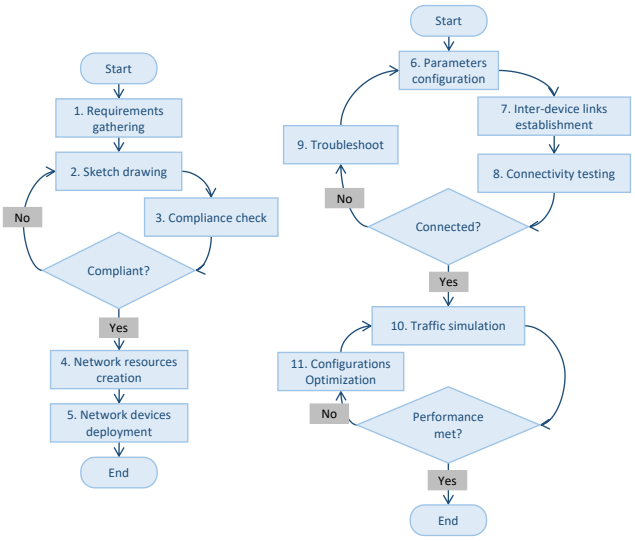
Infrastructure as Code (IaC)[10–12] uses machine-readable code to automate the management and provisioning of IT infrastructure as well as cloud platforms, replacing manual configuration with consistent, scalable, and version-controlled workflows. IaC tools like Terraform[13] and Pulumi[1] have mitigated some inefficiencies in network topology construction, but they still require manual translation of topology semantics into declarative code. For example, defining a "3-tier security zone" in Terraform demands explicit coding of subnet hierarchies, route tables, and security groups—tasks that remain inaccessible to non-experts and prone to misalignment with original requirements.

Recent advancements in Large Language Models (LLMs) have spurred IaC generation services [14–16], improving the efficiency of network topology construction to a certain extent. But these tools focus narrowly on isolated resource configurations (e.g., generating Dockerfiles for containers, Kubernetes manifests for pods, or AWS CloudFormation templates for VMs). They fail to capture holistic topology structures such as interdependencies between routers and firewalls, or security zone boundaries. As a result, a large proportion of LLM-generated IaC scripts require manual revisions to fix topology logic errors. Notably, industry-specific network topologies rely on specialized "images"—here referring to device operating system images, firmware, or preconfigured environment packages (not visual pictures). These images embed industry-critical features. Due to the LLMs' stagnant knowledge, exist-

* These authors are co-corresponding authors

[1]https://github.com/pulumi/pulumi

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00298



(a) (b)

Fig. 1: Process of the manual network topology construction. (a): Requirement analysis and resource allocation; (b): Configuration and verification.

ing approaches struggle to obtain the latest industry-specific image information and adapt to various industry scenarios.

To address the above limitations, this paper introduces IntelliTopo, an end-to-end LLM-powered IaC generation framework. To the best of our knowledge, this is the first approach designed specifically for industrial network topology construction. IntelliTopo bridges the gap between high-level natural language requirements and executable infrastructure. Deployed on our PaaS system, IntelliTopo service has operated continuously for 3 months, supporting 50+ topology emulation tasks across 10+ industries, including education, IoV (Internet of Vehicles), medicine, etc.

The contributions of this paper are as follows:

- We propose IntelliTopo, the first LLM-powered IaC generation framework designed for industrial network topology construction. It enables seamless translation from high-level natural language requirements to operational network environments, reducing manual effort and LLM computational overhead.
- We introduce an advanced prompt strategy named semantic topology parsing to enhance the LLMs' understanding of the holistic topology structure. Meanwhile, we proposed a unified intermediate template to guide the structured IaC generation process and achieve topology visualization.
- To address the stagnant knowledge limitations of LLMs and enhance alignment with real-world scenarios. We adopt a Model Context Protocol (MCP)-based image retrieval process named domain-aware image retrieval, which enables LLMs to obtain the latest industry-specific images in real time.

II. IntelliTopo

This section details the framework of IntelliTopo. The framework comprises two core modules: *Semantic Topology Parsing*, and *Domain-aware Image Retrieval*. These modules work synergistically to bridge high-level requirements and executable infrastructure, minimizing human intervention while ensuring industry adaptability. The overall framework is shown in Fig. 2.

A. Semantic Topology Parsing

The Semantic Topology Parsing module addresses the critical challenge of translating vague, unstructured natural language descriptions into precise, machine-understandable topology semantics. This module integrates six sequential components as prompts to ensure accuracy and robustness, as illustrated in Fig. 3.

1) Raw Requirement Input: The raw requirement input refers to the initial information provided by users, typically in free-form natural language , which includes descriptions for network division, network devices (routers, firewalls, and switches), and instance requirements. These descriptions often lack technical precision, requiring further processing to extract actionable topology details. The following is an example of a network topology description for a water supply factory:

```
Create a network topology including the DMZ,
office area, central control zone ... proces
s control zone contains 5 network segments: w
ater intake, chemical dosing, reaction tank,
... are connected to the intranet switch thro
ugh a confidential network gateway ... The of
fice area contains 8 terminals ...
```

2) Domain Knowledge Injection: To enhance the LLMs' understanding of network-specific concepts, we inject a curated network terminology repository into the parsing process. This repository includes 100+ domain terms (e.g., "VLAN partitioning", "DMZ zone", "stateful firewall") and their contextual definitions. For instance, when the input mentions "OT subnet," the module automatically associates it with "operational technology subnet, typically isolated from IT networks via firewalls" to avoid misinterpretation as a generic subnet. We also inject a list of detailed instance specifications for various devices. The definitions of these specifications are shown in Table I. In addition, we mandate the LLM in the prompt to select from the specified options while generating the intermediate template in Section II-B.

3) Few-Shot Examples: We embed 3 domain-specific successful cases into the prompt to guide the LLMs' parsing logic. These examples are selected to cover diverse scenarios and scales: i. A simple office network (9 nodes); ii. A medium-complexity enterprise campus network (18 nodes); and iii. A high-complexity industrial OT/IT converged network (35 nodes). Each example pairs a natural language requirement with its parsed topology template (Section II-B), enabling the LLMs to learn patterns in translating ambiguity into structure.

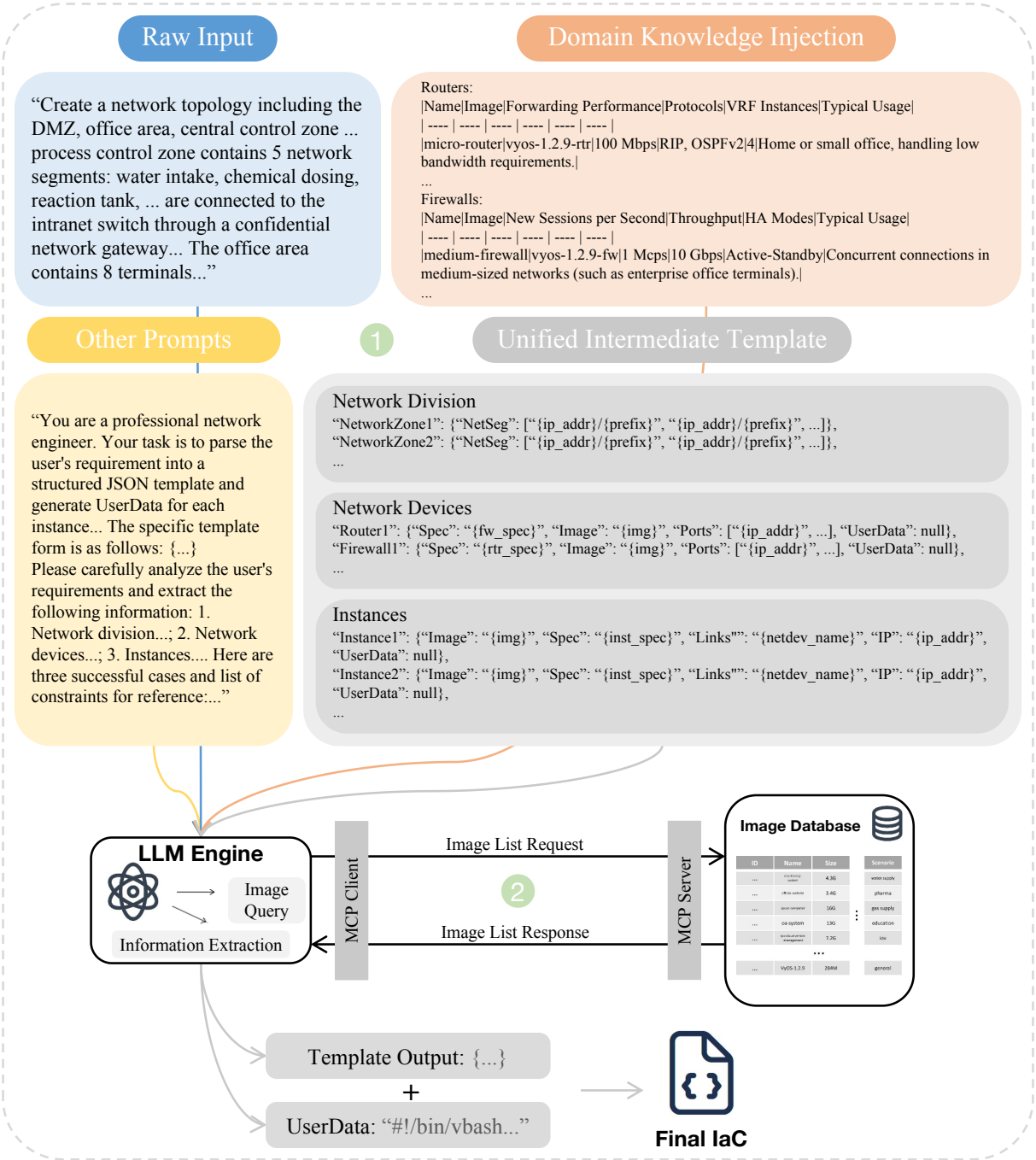


Fig. 2: The overall framework of IntelliTopo. It consists of two core modules: 1. Semantic Topology Parsing; and 2. Domain-aware Image Retrieval. The framework is encapsulated into a service, allowing users to submit requirements in natural language. IntelliTopo then automatically parse these requirements and generate deployable IaC code.

4) Constraint Explicitation: Network topologies are governed by implicit design rules that users may omit (e.g., "firewalls must be placed at network boundaries"). This component explicitly surfaces these constraints by mapping input requirements to a predefined set of topology-specific rules. For example: If the input specifies a "healthcare network" the module automatically appends "patient data servers must be isolated in a dedicated subnet with access restricted to authorized medical workstations only" (aligned with HIPAA requirements for protected health information); For "OT net-

TABLE I: Instance Specifications.

Attribute	Description	Example Values
	Router	
Name	Router specification name	micro-router, medium-router
Image	Router's image name	vyos-1.2.9-rtr, vyos-1.3.0-rtr
Forwarding Performance	Router's ability to process and forward data packets	100 Mbps, 1 Gbps
Protocols	Rules/algorithms for exchanging routing info	BGP, RIP, OSPFv2
VRF Instances	Number of independent virtual routing instances	4, 8
Typical Usage	Typical application scenarios	"Home or small office, low bandwidth requirements."
	Firewall	
Name	Firewall specification name	micro-firewall, medium-firewall
Image	Firewall's image name	vyos-1.2.9-fw, vyos-1.3.0-fw
New Sessions per Second	Firewall's ability to establish new connections per second	1 Mcps
Throughput	Data processing capacity under typical network traffic	1 Gbps, 10 Gbps
HA Modes	Modes ensuring uninterrupted operation	Active-Standby, Active-Passive
Typical Usage	Typical application scenarios	"Concurrent connections in medium-sized networks"
	Switch	
Name	Switch specification name	micro-switch, medium-switch
Image	Firewall's image name	vyos-1.2.9-sw, vyos-1.3.0-sw
Backplane Bandwidth	Maximum data flow the switch backplane can handle	100 Gbps, 1Tbps
PoE Power	Total power supplied via PoE ports to support PoE devices	370 W, 1200 W
Buffer Capacity	Buffer size to prevent packet loss	500 MB, 1 GB
Typical Usage	Typical application scenarios	"High-density data exchange (48-96 10/25 Gbps interfaces)"
	Instance	
Name	Instance specification name	small-workstation, medium-server
CPU	Number of vCPUs, affecting multitasking capability	2, 4
Memory	Temporary storage for running programs/data	4 GB, 8 GB
Storage	Long-term storage for OS, apps, and data	128 GB, 256 GB
Typical Usage	Typical application scenarios	"Personal PC", "Database server"

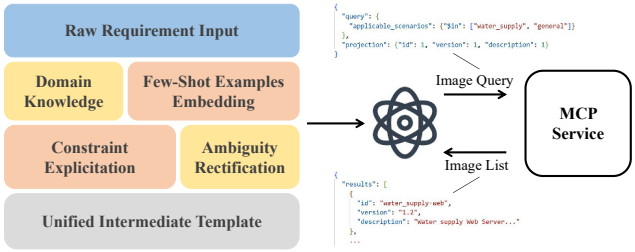


Fig. 3: Information acquisition process of the LLM. The left side represents Semantic Topology Parsing, where the LLM determines the required images and generates query statements based on user input. The right side represents the LLM calling the MCP service to obtain the corresponding image list.

works" it adds "PLC subnets must not be reachable from the public internet" (a critical industrial security norm). These constraints are encoded as structured rules to ensure the LLM interprets them unambiguously. An example is provided below:

```
{
  "rule_type": "isolation",
  "source": "PLC_subnet",
  "destination": "public_internet",
  "action": "block"
}
```

5) Ambiguity Rectification: Natural language inputs often contain vague expressions (e.g., "several servers", "secure connections") that hinder precise parsing. This component detects such ambiguities using a two-step process: i. Flagging terms with low specificity (e.g., "several" $\rightarrow$ undefined quantity, "secure" $\rightarrow$ undefined security controls); and ii. Resolving them via context-aware reasoning. For example: i. "Several servers" in a "small office network" is refined to "2 servers (1 for file storage + 1 for printer management)" based on typical office needs; ii. "Secure connections" in a school network is mapped to "links with 802.1X authentication" (a standard for securing classroom-to-data-center connections in educational settings). This rectification relies on a domain-ambiguity corpus manually curated by network engineers, ensuring consistency with real-world usage patterns.

6) Unified Intermediate Template: IntelliTopo generates IaC code based on our custom Pulumi library. Generating long and complex IaC code—especially when it embeds intricate topology relationships, poses significant challenges for LLMs. Direct code generation often leads to syntax errors, disconnected topology logic, or misalignment between device configurations and their intended roles in the network. To address these issues, we introduce a unified intermediate template, which includes 4 sections: Network Division, Network Devices, Instances, and UserData. This unified intermediate template not only guides LLMs to generate more accurate network topologies but also facilitates our visual rendering of network topologies, which will be elaborated on in Section II-B.

TABLE II: Attributes of device images.

Attribute	Description	Example Values
id	Unique identifier for the device image	ubuntu-2004, centos-8-minimal
name	Human-readable name of the image	Ubuntu 20.04 Base, Centos 8 Cloud
format	File format of the image	qcow2, raw, vmdk, iso, docker-tar
size (GB)	Storage size of the image file	2.5, 8, 14.7
os_middleware	Embedded operating system or middleware	Ubuntu 20.04 LTS, Centos 8.9, Alpine 3.20, OpenJDK 17
min_vcpu	Minimum number of vCPUs required to run the image	2, 4
min_memory (GB)	Minimum RAM required to run the image	4, 8
min_storage (GB)	Minimum storage space required for deployment	20, 100
architecture	Supported hardware architecture	x86_64, ARM64
hash	Cryptographic hash for integrity verification	"sha256:7a3b9c..."
version	Version number of the image	2.3.1
applicable_scenarios	Target use cases or industries	["edge node lightweight deployment", "pharma"]
deployment_requirements	Special conditions for deployment	"Min vCPU/RAM: 2/4", "virtio driver support", "cloud-init"
status	Current availability status	"Active", "Deprecated"
creation_time	Timestamp when the image was added to the database	"2025-03-20T14:15:00Z"
description	Additional notes	"pre-configured for pharma network security"

B. Unified Intermediate Template

To mitigate the complexity of direct IaC generation, we introduce a unified intermediate template in the form of a human-readable and machine-parseable JSON structure that formalizes all topology elements without tying them to a specific IaC syntax. This template acts as a "blueprint" that consolidates structured topology semantics (from Section II-A) and device image metadata (from Section II-C), enabling systematic validation and modular code generation. Moreover, a unified template also facilitates the visual rendering of the topology. The composition of the unified template is as follows (for brevity, non-core portions are omitted):

Network Division-Logical groupings of devices with boundary rules to enforce isolation and access controls. An example of network division for a water supply company is as follows:

```
{
  "Network Division": {
    "DMZ": {"NetSeg": ["192.168.0.0/24"]},
    "Office Area": {"NetSeg": ["192.168.0.0/24"]},
    "External Area": {"NetSeg": ["10.1.1.0/24"]},
    "Central Control Zone": {"NetSeg":
     ["192.168.1.0/24"]},
    "Process Control Zone": {"NetSeg": ["192.168.2.0/24",
     "192.168.3.0/24", "192.168.4.0/24", "192.168.5.0/24",
      "192.168.6.0/24"]},
    ...
  }
}
```

Network Devices-A list of all network devices, including "Routers&Firewalls" and "Switches". The device names are mapped directly from Section II-A. The "Spec" attribute of each device is selected from the specification list provided in Section II-A2. The "Image" attribute of each device is obtained through the MCP module in Section II-C. The "UserData" attribute is left "null" to be generated separately later. From the "Ports" and "Links" attributes, we can extract which network segments the device is connected to and its superior network devices, thereby obtaining the main structure of a network topology. The edge structure of the network topology will be further refined through attributes in "Instances". An example of network devices for a water supply company is as follows:

```
{
  "Routers&Firewalls": {
    "Firewall": {"Spec": "medium-firewall", "Image": "vyos
     -1.2.9", "Ports": ["192.168.0.2", "10.1.1.1"], "
     UserData": null},
    "Gateway": {"Spec": "medium-firewall", "Image": "vyos
     -1.2.9", "Ports": ["192.168.0.1", "192.168.1.1"], "
     UserData": null},
    ...
  }
  "Switches": {
    "Core Switch": {"Spec": "large-switch", "Image": "vyos
     -sw", "Links": ["Firewall", "Gateway"]},
    "Office Area Switch": {"Spec": "medium-switch", "Image
     ": "vyos-sw", "Links": ["Core Switch"]},
    ...
  }
}
```

Instances-A list of all workload instances that run specific services, with instance names mapped directly from Section II-A and image from Section II-C. The configuration for "Spec" and "UserData" is the same as that for Network Devices. The "Links" attribute indicates the devices it is connected to, reflecting the edge topological structure. If the "IP" attribute is not empty, an IP address will be assigned to the instance. Otherwise, the instance will automatically obtain an IP via DHCP. An example is provided below:

```
{
  "Instances": {
    "DMZ": [
      {"Web Server": {"Image": "water_supply-web", "Spec":
       "medium-server", "Links": ["Core Switch"], "IP":
       "192.168.0.3", "UserData": null}},
      {"Database Server": {"Image": "water_supply-database
      ", "Spec": "medium-server", "Links": ["Core Switch"],
       "IP": "192.168.0.4", "UserData": null}},
      ...
    ],
    "Office Area": [
      {"Office Terminal 1": {"Image": "water_supply-office
      ", "Spec": "small-workstation", "Links": ["Office
      Area Switch"], "IP": null, "UserData": null}},
      {"Office Terminal 2": {"Image": "water_supply-office
      ", "Spec": "small-workstation", "Links": ["Office
      Area Switch"], "IP": null, "UserData": null}},
    ...
    ],
    ...
  }
}
```

UserData-Scripts that initialize post-deployment instances, which are critical for operational readiness but unrelated to the core topology structure. Generating UserData alongside topology code often leads to errors or misalignment with instance functions. To avoid this, we generate UserData separately and anchor it to the instances via a placeholder. In IntelliTopo, UserData is mainly used to configure routers to divide network segments and assign IP addresses, as well as to configure firewall rules for regional isolation. Examples are provided below:

```
{
  "Firewall": [
    "#!/bin/vbash",
    "source /opt/vyatta/etc/functions/script-template",
    "configure",
    "set protocols rip network 192.168.0.0/24",
    "set protocols rip network 10.1.1.0/24",
    "set protocols rip redistribute connected",
    "commit",
    "save"
  ],
  "Process Control Router": [
    "#!/bin/vbash",
    "source /opt/vyatta/etc/functions/script-template",
    "configure",
    "laneth0=`ip a | grep 192.168.6.1 | awk '{print $7
     }'`",
    "lanip0=`ip a | grep 192.168.6.1 | awk '{print $2}'`",
    "laneth1=`ip a | grep 192.168.4.1 | awk '{print $7
     }'`",
    "lanip1=`ip a | grep 192.168.4.1 | awk '{print $2}'`",
    ...
    "del interface ethernet $laneth0 address dhcp",
    "set interface ethernet $laneth0 address $lanip0",
    "del interface ethernet $laneth1 address dhcp",
    "set interface ethernet $laneth1 address $lanip1",
    ...
    "set protocols rip network 192.168.1.0/24",
    "set protocols rip network 192.168.2.0/24",
    ...
    "set protocols rip redistribute connected",
    ...
    "commit",
    "save"
  ],
  ...
}
```

Combining the *Network Division*, *Network Devices*, *Instances* with separately generated *UserData* yields a complete template, which is then converted into deployable IaC code via our custom conversion service. A key advantage of this design is flexibility: one template can be translated into code compatible with different IaC frameworks. In our implementation, we adopted a proprietary custom Pulumi library. Another advantage is that it facilitates the visual rendering of network topologies, which will be elaborated on in Section IV.

C. Domain-aware Image Retrieval

While LLMs excel at parsing natural language, they suffer from two critical limitations in generating industry-aligned IaC: i. Stagnant knowledge (training data frozen at a specific timestamp, leaving them unaware of latest device images); and ii. Lack of industry specificity (inability to distinguish industry-specific preferences). These limitations result in IaC that references obsolete images or misaligns with real-world deployment norms, leading to deployment failures or impractical topology simulations. To address these gaps, our Domain-aware Image Retrieval module leverages the Model Context Protocol[2] (MCP) to enable dynamic interaction with a curated industry image database. MCP facilitates real-time retrieval of the latest, industry-specific device images, ensuring the generated IaC references resources that are both current and aligned with industry-specific deployment standards. MCP adopts a client-server architecture:

1) MCP Server: We utilize MongoDB as our database service for its strengths in managing unstructured and semi-structured data—essential for storing the diverse attributes of device images. And we adopt MongoDB MCP Server[3] to interact with the database. The MongoDB MCP Server enables developer tools with MCP clients to interact directly with a MongoDB database and to handle a range of administrative tasks, such as managing cluster resources, as well as data-related operations like querying and indexing.

2) MCP Client: We use Cline[4], an open-source MCP Client tool designed for seamless integration with LLM workflows, as the bridge between the LLMs and MongoDB MCP Server. MCP Client receives parsed requirements from the LLM. These requests encode specific hardware constraints and contextual attributes derived from the LLMs' analysis. Table II outlines the attributes of the images, along with their descriptions and examples. These attributes are also provided to the LLM to enable the MCP workflow to translate LLM-parsed requirements into targeted queries, ensuring retrieved images align with both technical specifications and practical deployment needs.

III. Evaluation

Most existing IaC evaluation datasets[17–19] focus on single-resource provisioning (e.g., VM configuration or subnet creation) and lack coverage of network topology construction, which involves multi-device relationships, subnet interdependencies, and industry-specific constraints. To validate the effectiveness of IntelliTopo, we constructed a domain-specific dataset comprising 60 network simulation requirements derived from real-world scenarios, ensuring diversity in scale and industry sources to enhance generalizability. Specifically, the dataset is stratified by: i. Scale: small (≤ 10 nodes), medium ($11-30$ nodes), and large (≥ 31 nodes), to test performance across complexity levels; and ii. Industry Source: including IoV (Internet of Vehicles), Pharma, University Campus, Water Supply, etc, reflecting diverse deployment norms. Table III shows the composition of our dataset. Key evaluation metrics include: i. Deployment Success Rate (DSR), measuring syntactic correctness and topological completeness; and ii. Average Construction Time, end-to-end latency from inputting natural language requirements to a fully deployed IaC. We designed a comprehensive evaluation covering three dimensions: *Performance*, *Parameter Sensitivity*, and *Efficiency* to answer the following research questions:

[2]https://www.anthropic.com/news/model-context-protocol
[3]https://github.com/mongodb-js/mongodb-mcp-server
[4]https://github.com/cline/cline

RQ1. How does IntelliTopo perform against state-of-the-art IaC generation methods?

RQ2. What is the contribution of IntelliTopo's core modules (*Semantic Topology Parsing* and *Domain-aware Image Retrieval*)?

RQ3. How do LLMs' parameters affect IntelliTopo's performance in IaC generation?

RQ4. How does IntelliTopo's efficiency compare to manual construction across topology scales?

TABLE III: Composition of our dataset.

Scale	Industry Source
Small ($\leq$ 10 nodes, 60%)	Classroom, Office
Medium (11-30 nodes, 20%)	Gas Supply, Office
Large ($\geq$ 31 nodes, 20%)	IoV, Pharma, University Campus, Water Supply

A. Performance

A key challenge in evaluating topology-focused IaC generation is that most IaC generation tools are not designed for topology construction and lack native support for encoding multi-device relationships, subnet interdependencies, or industry-specific image database. To address this, we selected general LLMs + AIAC[5] as comparative baselines for IntelliTopo. AIAC is a command line tool for generating IaC templates, configurations, utilities, and queries. It offers compatibility with diverse LLM providers (both commercial APIs and local deployments), making it a representative proxy for general-purpose IaC generation workflows.

To ensure fair comparison, we augmented the baseline setup by embedding our industry-specific image database into prompts, enabling general LLMs to access the same device image inventory as IntelliTopo. All evaluations were conducted on locally deployed LLMs to ensure consistency with data security requirements.

Table IV quantifies IntelliTopo's performance advantage in network topology IaC generation. We evaluated the 7B version of each LLM, which is provided by all the large model families involved in the comparison. Compared to baseline methods (general LLMs + AIAC), IntelliTopo achieves a $30-40\%$ improvement in DSR. Critically, when controlling for base model, the consistent outperformance of our framework confirms that its topology-aware design drives the improvement. These results directly answer **RQ1** by demonstrating IntelliTopo's superiority in addressing the unique challenges of network topology IaC generation.

We conducted ablation experiments on IntelliTopo with Qwen2.5-32B as the backend model to quantify the contribution of its core modules and components, with results summarized in Table V. Removing the MCP reduces the average DSR by 10%, with more pronounced drops in large-scale scenarios, confirming MCP's critical role in grounding outputs in real-world, industry-specific resources.

[5]https://github.com/gofireflyio/aiac

TABLE IV: DSR of IntelliTopo vs. Baselines.

Method	Backend	Small	Medium	Large	Avg.
IntelliTopo	Qwen2.5	83.3%	50%	25%	65%
	Deepseek-R1	77.8%	58.3%	16.7%	61.7%
	Llama3.1	86.1%	41.7%	16.7%	63.3%
AIAC	Qwen2.5	41.7%	16.7%	0%	28.3%
	Deepseek-R1	38.9%	8.3%	0%	25%
	Llama3.1	38.9%	16.7%	0%	26.7%

Omitting the unified intermediate template leads to a 15% average DSR reduction, driven primarily by syntax errors (e.g., mismatched subnet-device IDs) and fragmented topology logic (e.g., firewalls unlinked to protected zones), underscoring the template's function in enforcing structural consistency across complex networks.

Meanwhile, relying solely on basic prompt engineering without few-shot examples or explicit constraint yields a low 26.7% DSR. This reflects LLMs' struggles to parse multi-device relationships (e.g., misinterpreting "OT-IT subnet isolation" as generic segmentation), highlighting the necessity of advanced prompt design in enhancing semantic understanding of topology-specific concepts.

These findings collectively validate that IntelliTopo's performance stems from the synergistic integration of its core modules, directly answering **RQ2** by quantifying how each component addresses distinct challenges in generating industry-aligned, logically consistent network topologies.

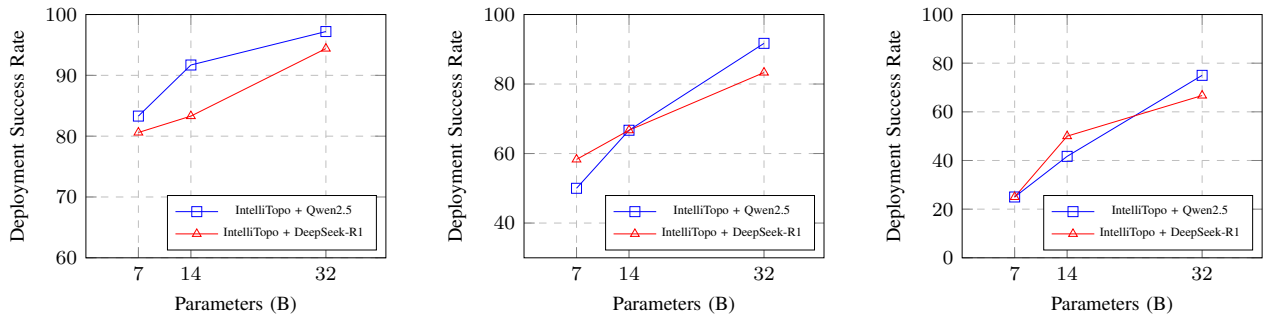
TABLE V: Ablation Study.

Method	Small	Medium	Large	Avg.
IntelliTopo + Qwen2.5-32B	97.2%	91.7%	75%	91.7%
w/o MCP	86.1%	83.3%	66.7%	81.7%
w/o MCP&Template	75%	66.7%	41.7%	66.7%
w/o MCP&Template&STP	36.1%	25%	0%	26.7%

B. Parameter Sensitivity

To isolate the impact of model parameters, we tested IntelliTopo with two mainstream open-source LLM families: Qwen2.5 and DeepSeek-R1, across varying parameter scales of 7B, 14B, and 32B. All models were deployed locally to ensure consistency with data security requirements. Fig. 4 plots the DSR across the two model families and their respective parameter scales. Analysis of the results is as follows:

14B models achieve an average 77.1% DSR in Fig. 4a and Fig. 4b, which is sufficient for most small-to-medium topologies (3–30 nodes). 32B models achieve an average 70.9% DSR in Fig. 4c , excelling in large-scale topologies ($\geq$ 31 nodes). All these results were obtained on locally deployed LLMs. Compared to commercial APIs, our framework reduces VRAM usage significantly and avoids prohibitive computational overheads such as higher latency and excessive power consumption. This analysis answers **RQ3**:



(a) Small (b) Medium (c) Large

Fig. 4: Deployment Success Rate of LLM families across different topology scales.

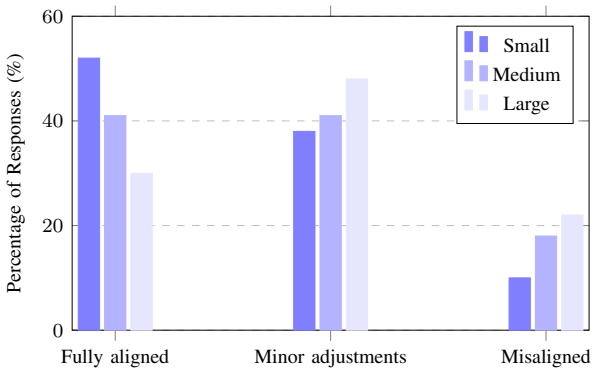


Fig. 5: Distribution of human satisfaction survey responses evaluating alignment of IntelliTopo's outputs with user requirements on different network topology scales.

While model parameters positively correlate with IntelliTopo's performance, 14B–32B open-source models already deliver satisfactory DSR ($\geq$ 75%). This balance of accuracy and resource efficiency makes IntelliTopo both computationally feasible and accessible for enterprises with constrained infrastructure, validating the practicality of our approach in real-world deployment.

C. Efficiency

We compare the end-to-end time requirement for IntelliTopo and manual workflows to construct and deploy network topologies across different scales.

Manual Construction Baseline: We recruited 5 network engineers with 5+ years of experience to establish a realistic human performance benchmark. Each engineer was provided with identical requirement documents, one from each scale category, and tasked with writing code using industry-standard IaC tools. Time was measured from receipt of the requirement document to completion of a fully deployed topology.

IntelliTopo's Runtime Environment: The framework was configured with Qwen2.5-32B as the backend LLM, deployed on a server with a single NVIDIA A6000 48GB GPU for reasoning. IntelliTopo received the same requirement documents as the engineers and was tested 10 times per scale to ensure result stability. The average time across runs was recorded, encompassing all stages from semantic parsing to fully deployed topologies.

Table VI summarizes the average time for each workflow across topology scales. Results show IntelliTopo significantly outperforms manual construction in efficiency across all scales, with time savings increasing with topology complexity: 50% for small, 70% for medium and large topologies. This directly addresses **RQ4** by demonstrating that IntelliTopo significantly reduces the time consuming for network topology construction. It should be noted that the document provided in our evaluations are derived from cases that have been successfully deployed on our PaaS system. In real-world scenarios, construction time often extends to days due to additional steps not included in our controlled experiment (e.g., multi-team approval, on-site device provisioning, and compatibility testing with legacy systems).

TABLE VI: Human Comparison.

Method	Scale	Avg. Time
IntelliTopo	small	27 min
	medium	42 min
	large	1.2 h
Human	small	51 min
	medium	2.2 h
	large	4.3 h

We supplemented the efficiency analysis with a human satisfaction survey among 50 users to evaluate the alignment of IntelliTopo's outputs with user requirements. Fig. 5 illustrates the distribution of responses: On average, 45.2% of evaluations fell into "Fully aligned", 40.8% into "Minor adjustments", and only 14% into "Misalignment". This breakdown, with over 85% of outputs requiring little to no modification, corroborates the efficiency gains observed in time comparisons, demonstrates that IntelliTopo not only accelerates topology construction but also produces results that align closely with human expectations.

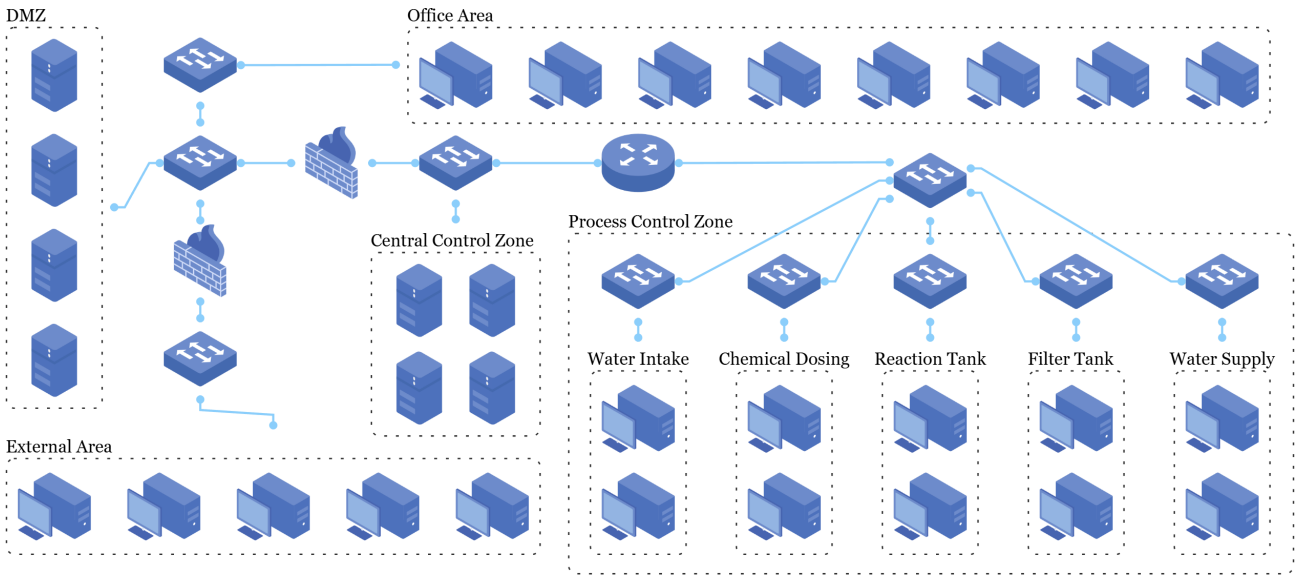


Fig. 6: The network topology of a water supply company rendered from our unified intermediate template.

IV. Discussion

A. Why can IntelliTopo extract the topological structures?

IntelliTopo's ability to extract network topology structures stems from its fundamental shift from treating topology as a collection of isolated resources to modeling it as an interconnected system of relationships. IntelliTopo's semantic topology parsing module actively identifies and formalizes topological logic, parsing natural language to extract not just devices and subnets, but critical relational constraints via domain knowledge injection and constraint explicitation. This relational understanding is then anchored in a unified intermediate template that enforces structural consistency, ensuring dependencies like router-firewall connections or security zone boundaries are preserved in code generation.

B. Unified Intermediate Template: Beyond Prompt

The unified intermediate template goes beyond its role in semantic topology parsing. In prompt design, it structures LLM outputs to prevent fragmented topology logic, ensuring device links and zone boundaries are explicitly defined. More importantly, its structured format enables seamless, automated visualization of network topologies. As shown in the water supply company case (Fig. 6), critical details like firewall-isolated process control zone become intuitively visible from the template. This dual utility turns the template into a bridge between machine-readable code and human-understandable topology, simplifying validation and collaboration.

C. Limitations

A key limitation of IntelliTopo is that the current framework requires full regeneration when outputs misalign with requirements, lacking support for local modifications. Additionally, when IntelliTopo rectifies ambiguous user inputs, it provides no explicit feedback on these adjustments. To address these limitations, future work will develop an interactive conversational agent that enables real-time, human-in-the-loop refinement, allowing users to propose modifications via natural language, with the agent dynamically updating the topology and IaC code while maintaining consistency, thus bridging the gap between automated generation and adaptive human feedback.

V. Related Works

Infrastructure as Code (IaC) revolutionizes IT infrastructure management by defining and provisioning resources through machine-readable code, replacing manual configuration with automated, consistent, and scalable workflows. As a core DevOps practice[20–22], IaC enables version control, reproducibility, and error reduction, addressing the inefficiencies of traditional manual operations[23, 24]. IaC tools are broadly categorized into two types: configuration management tools (e.g., Ansible, Chef) for installing and managing software on pre-existing infrastructure, and provisioning tools (e.g., Terraform, Pulumi) for deploying infrastructure components across cloud providers. While IaC offers significant benefits—including automation, transparency, and scalability—it faces challenges such as increased complexity in multi-platform environments, dependency management, and the need for specialized expertise to handle intricate configurations[25]. These challenges have motivated research into automating IaC generation, particularly leveraging large language models to bridge the gap between natural language requirements and technical implementations.

Large Language Models (LLMs) have emerged as a promising solution for automating Infrastructure as Code (IaC) generation, addressing the complexity and manual effort in traditional infrastructure configuration. Existing research

highlights LLMs' capability to translate natural language requirements into deployable IaC scripts across tools like Terraform, Ansible, and Pulumi. For instance, decoder-based models such as CodeGen[14] and GPT-3.5/4[26], pre-trained on extensive code repositories, demonstrate proficiency in generating YAML playbooks[27] and Terraform configurations by leveraging in-context learning and fine-tuning on domain-specific datasets[28]. Studies show that LLMs outperform rule-based tools in adapting to diverse infrastructure scenarios, with GPT-3.5 achieving over 59% functional correctness in AWS configuration tasks, far exceeding smaller models like CodeParrot[29]. However, challenges persist, including syntactic errors in complex topologies, reliance on static training data, and security risks from unvalidated configurations[30, 31], underscoring the need for domain-specific optimizations and validation mechanisms. Existing methods primarily target single-resource provisioning (e.g., VMs, subnets) and lack support for network topology construction, which requires encoding multi-device relationships, subnet interdependencies, and industry-specific device constraints. This limitation leads to fragmented or incorrect outputs when generating large-scale network topologies, as LLMs struggle to model holistic structural logic.

Prompt engineering plays a critical role in enhancing LLMs' performance in specialized tasks like code generation[32] and logical reasoning[33], enabling precise control over model outputs without parameter modifications. Key techniques include zero-shot[34] and few-shot[35, 36] prompting, which guide models through task descriptions or examples, and advanced methods like Chain-of-Thought (CoT) prompting[33], which decomposes complex problems into step-by-step reasoning chains—improving accuracy in math and code-related tasks by up to 90% in benchmarks like GSM8K[37]. Extensions such as Auto-CoT[38] and Self-Consistency[39] further automate reasoning chain generation and reduce errors by aggregating diverse solutions. For IaC generation, structured prompting techniques like Structured Chain-of-Thought (SCoT)[32] and Chain-of-Code (CoC)[40] explicitly incorporate program structures and pseudocode, enhancing alignment with code syntax and reducing configuration conflicts. These methods collectively bridge the gap between natural language intent and technical implementation, though challenges remain in handling ambiguous requirements and scaling to large-scale topologies.

Model Context Protocol (MCP) has emerged as a standardized framework to streamline interactions between AI models and external tools, addressing fragmentation in traditional tool-invocation workflows. MCP's architecture—comprising a host (AI application), client (intermediary), and server (tool/resource provider)—enables dynamic discovery, selection, and orchestration of external resources (e.g., device images, APIs) and tools. Key applications include integrating LLMs with code repositories (GitHub[6]), development environments (Claude Desktop[7], Cursor[8]), and cloud services (Cloudflare[9]), where MCP simplifies multi-step workflows like real-time data retrieval and automated configuration.

[6]https://github.com/modelcontextprotocol/servers

VI. Conclusion

We presents IntelliTopo, a novel framework designed to generate topology-aware, industry-aligned Infrastructure as Code (IaC) by enhancing large language models (LLMs) with domain-specific modules. Key innovations include semantic topology parsing that converts natural language requirements into structured topology semantics, including a unified intermediate template that ensures structural consistency and enables seamless topology visualization, the Model Context Protocol (MCP) for dynamic retrieval of industry-specific device images. Evaluations demonstrate that IntelliTopo outperforms state-of-the-art baselines with a 91.7% average Deployment Success Rate (DSR), while efficiently operating with 14B–32B parameter models and reducing construction time by 70% compared to manual workflows. Human evaluations further validate its practicality, with 85% of outputs requiring minimal or no modifications. By addressing the limitations of existing IaC generation tools, particularly their inability to model multi-device relationships and industry constraints. IntelliTopo not only advances the state-of-the-art in automated infrastructure configuration but also provides a scalable, accessible solution for network engineers across diverse industries. Future work will develop an interactive conversational agent to enable real-time, human-in-the-loop refinement, further bridging the gap between automated generation and adaptive practical needs.

Acknowledgment

This work was supported by the Major Key Project of Peng Cheng Laboratory (Grant No. PCL2024A05), the National Natural Science Foundation of China (Grant Nos. 62227808, 62402168, U23A20322), and the Natural Science Foundation of Hunan Province (Grant No. 2024JJ6156).

References

[1] O. David, P. Thornley, and M. Bagheri, "Software defined networking (SDN) for campus networks, wan, and datacenter," in *International Conference on Smart Applications, Communications and Networking, SmartNets 2023, Istanbul, Turkey, July 25-27, 2023*. IEEE, 2023, pp. 1–8. [Online]. Available: https://doi.org/10.1109/SmartNets58706.2023.10215722

[2] L. Xu, H. Hu, and Y. Liu, "Sfcsim: a network function virtualization resource allocation simulation platform," *Clust. Comput.*, vol. 26, no. 1, pp. 423–436, 2023. [Online]. Available: https://doi.org/10.1007/s10586-022-03670-8

[3] Y. Jia, Z. Gu *et al.*, "Artificial intelligence enabled cyber security defense for smart cities: A novel attack detection framework based on the MDATA model," *Knowl. Based Syst.*, vol. 276, p. 110781, 2023. [Online]. Available: https://doi.org/10.1016/j.knosys.2023.110781

[4] C. Hu, Y. Ruan, and J. Guo, "Network planning design and simulation for smes," in *Fourth International Conference on Electronics Technology and Artificial Intelligence (ETAI 2025)*, vol. 13692. SPIE, 2025, pp. 1833–1842.

[5] X. Wang and J. Ma, "Cloud-network-end collaborative security for wireless networks: Architecture, mechanisms, and applications," *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 18–33, 2024.

[7]https://modelcontextprotocol.io/quickstart/user

[8]https://docs.cursor.com/en/context/mcp

[9]https://www.cloudflare.com/

[6] A. G. Filho, E. K. Viegas *et al.*, "A dynamic network intrusion detection model for infrastructure as code deployed environments," *J. Netw. Syst. Manag.*, vol. 33, no. 4, p. 75, 2025. [Online]. Available: https://doi.org/10.1007/s10922-025-09940-1

[7] S. Kumar and H. Vardhan, "Cyber security of OT networks: A tutorial and overview," *CoRR*, vol. abs/2502.14017, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2502.14017

[8] A. Alzu'Bi, A. Alomar *et al.*, "A review of privacy and security of edge computing in smart healthcare systems: issues, challenges, and research directions," *Tsinghua Science and Technology*, vol. 29, no. 4, pp. 1152–1180, 2024.

[9] E. Khezri, H. Hassanzadeh *et al.*, "Security challenges in internet of vehicles (iov) for its: A survey," *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1700–1723, 2025.

[10] K. Morris, *Infrastructure as code: managing servers in the cloud.* " O'Reilly Media, Inc.", 2016.

[11] P. Quéval, N. E. Hörner *et al.*, "On the understandability of coupling-related practices in infrastructure-as-code based deployments," *Inf. Softw. Technol.*, vol. 185, p. 107761, 2025. [Online]. Available: https://doi.org/10.1016/j.infsof.2025.107761

[12] C. Pahl, N. G. Gunduz *et al.*, "Infrastructure as code: Technology review and research challenges," in *Proceedings of the 15th International Conference on Cloud Computing and Services Science, CLOSER 2025, Porto, Portugal, April 1-3, 2025*, V. Cardellini and M. van Steen, Eds. SCITEPRESS, 2025, pp. 151–158. [Online]. Available: https://doi.org/10.5220/0013247700003950

[13] M. Howard, "Terraform - automating infrastructure as a service," *CoRR*, vol. abs/2205.10676, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.10676

[14] E. Nijkamp, B. Pang *et al.*, "Codegen: An open large language model for code with multi-turn program synthesis," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=iaYcJKpY2B_

[15] H. Touvron, T. Lavril *et al.*, "Llama: Open and efficient foundation language models," *CoRR*, vol. abs/2302.13971, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.13971

[16] L. Ouyang, J. Wu *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed *et al.*, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html

[17] P. T. J. Kon, J. Liu *et al.*, "Iac-eval: A code generation benchmark for cloud infrastructure-as-code programs," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, A. Globersons, L. Mackey *et al.*, Eds., 2024. [Online]. Available: http://papers.nips.cc/paper_files/paper/2024/hash/f26b29298ae8acd94bd7e839688e329b-Abstract-Datasets_and_Benchmarks_Track.html

[18] Y. Xu, Y. Chen *et al.*, "Cloudeval-yaml: A practical benchmark for cloud configuration generation," in *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*, P. B. Gibbons, G. Pekhimenko, and C. D. Sa, Eds. mlsys.org, 2024. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2024/hash/554e056fe2b6d9fd27ffcd3367ae1267-Abstract-Conference.html

[19] S. Munshi, S. Pathak *et al.*, "Acse-eval: Can llms threat model real-world cloud infrastructure?" *CoRR*, vol. abs/2505.11565, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2505.11565

[20] R. Punjabi and R. Bajaj, "User stories to user reality: A devops approach for the cloud," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2016, pp. 658–662.

[21] V. Manolov, D. Gotseva, and N. Hinov, "Practical comparison between the CI/CD platforms azure devops and github," *Future Internet*, vol. 17, no. 4, p. 153, 2025. [Online]. Available: https://doi.org/10.3390/fi17040153

[22] A. Saxena, S. Singh *et al.*, "Devops automation pipeline deployment with iac (infrastructure as code)," *CoRR*, vol. abs/2503.16038, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2503.16038

[23] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[24] C. A. Cois, J. Yankel, and A. Connell, "Modern devops: Optimizing software development through effective system interactions," in *2014 IEEE International Professional Communication Conference, IPCC 2014, Pittsburgh, PA, USA, October 13-15, 2014*. IEEE, 2014, pp. 1–7. [Online]. Available: https://doi.org/10.1109/IPCC.2014.7020388

[25] C. de Siebra, R. Lacerda *et al.*, "From theory to practice: The challenges of a devops infrastructure as code implementation," in *Proceedings of the 13th International Conference on Software Technologies, ICSOFT 2018, Porto, Portugal, July 26-28, 2018*, L. A. Maciaszek and M. van Sinderen, Eds. SciTePress, 2018, pp. 461–470. [Online]. Available: https://doi.org/10.5220/0006826104610470

[26] J. Ye, X. Chen *et al.*, "A comprehensive capability analysis of GPT-3 and GPT-3.5 series models," *CoRR*, vol. abs/2303.10420, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.10420

[27] S. Pujar, L. Buratti *et al.*, "Invited: Automated code generation for information technology tasks in YAML through large language models," in *60th ACM/IEEE Design Automation Conference, DAC 2023, San Francisco, CA, USA, July 9-13, 2023*. IEEE, 2023, pp. 1–4. [Online]. Available: https://doi.org/10.1109/DAC56929.2023.10247987

[28] L. Gao, S. Biderman *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *CoRR*, vol. abs/2101.00027, 2021. [Online]. Available: https://arxiv.org/abs/2101.00027

[29] L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[30] S.-Y. G. B.-X. F. Q. L. Xu YANG, Ji-Yuan FENG, "Fedpd: personalized federated learning based on partial distillation," *Frontiers of Computer Science*, vol. 20, no. 3, p. 2003604, 2026. [Online]. Available: https://journal.hep.com.cn/fcs/EN/abstract/article_52020.shtml

[31] S. Guo, X. Wang *et al.*, "A federated learning scheme meets dynamic differential privacy," *CAAI Transactions on Intelligence Technology*, vol. 8, no. 3, pp. 1087–1100, 2023.

[32] J. Li, G. Li *et al.*, "Structured chain-of-thought prompting for code generation," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, pp. 37:1–37:23, 2025. [Online]. Available: https://doi.org/10.1145/3690635

[33] J. Wei, X. Wang *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed *et al.*, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[34] A. Radford, J. Wu *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[35] T. B. Brown, B. Mann *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato *et al.*, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[36] L. Du, Z. Gu *et al.*, "A few-shot class-incremental learning method for network intrusion detection," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 2, pp. 2389–2401, 2024. [Online]. Available: https://doi.org/10.1109/TNSM.2023.3332284

[37] K. Cobbe, V. Kosaraju *et al.*, "Training verifiers to solve math word problems," *CoRR*, vol. abs/2110.14168, 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[38] Z. Zhang, A. Zhang *et al.*, "Automatic chain of thought prompting in large language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=5NTt8GFjUHkr

[39] X. Wang, J. Wei *et al.*, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=1PL1NIMMrw

[40] C. Li, J. Liang *et al.*, "Chain of code: Reasoning with a language model-augmented code emulator," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=vKtomqlSxm