An Interview Study with Infrastructure as Code (IaC) Practitioners in Bangladesh

by

Maisha Mostofa Prima
22266007

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science

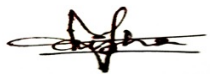
Department of Computer Science and Engineering
Brac University
March 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Maisha Mostofa Prima
22266007

Approval

The thesis/project titled "An Interview Study with Infrastructure as Code (IaC) Practitioners in Bangladesh" submitted by

Maisha Mostofa Prima (22266007)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science on March 13th, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. S M Taiabul Haque
Associate Professor
Department of Computer Science
and Engineering
Brac University

Examiner:
(Internal)

Dr. Farida Chowdhury
Professor
Department of Computer Science
and Engineering
Brac University

Examiner:
(External)

Dr. Akond Rahman
Assistant Professor
Department of Computer Science
and Software Engineering
Auburn University

Program Coordinator:
(Member)

Dr. Md. Sadek Ferdous, PhD
Professor
Department of Computer Science
and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi, PhD
Chairperson
Department of Computer Science
and Engineering
Brac University

Ethics Statement

This research has been conducted in accordance with ethical standards and guidelines to ensure the protection, respect, and dignity of all participants involved. Given the professional nature of the study, particularly its focus on Infrastructure as Code (IaC) practices among software developers and DevOps engineers in Bangladesh, the following measures were implemented to uphold ethical integrity:

Participant Consent:

All participants were fully informed about the objectives, scope, and potential risks of the research prior to their involvement. Verbal or written informed consent was obtained before conducting interviews, and participants were assured that their participation was entirely voluntary. They were also made aware that they could withdraw from the study at any time without providing an explanation or facing any penalties.

Anonymity and Confidentiality:

To protect participants' identities, pseudonyms were used in all documentation and publications. No personally identifiable information was collected or disclosed. Data was securely stored in encrypted files, with access restricted to the primary researcher. Additional measures were taken to ensure that any shared quotes, insights, or data could not be traced back to individual participants or their respective organizations.

Sensitivity to Professional Context:

Recognizing the professional nature of the participant group, the study was designed to minimize any risks of reputational harm, professional distress, or exposure. Interview questions were carefully framed to ensure the discussion remained focused on general practices rather than individual performance or confidential organizational information. Participants were given the opportunity to review their responses and make any necessary adjustments before final inclusion.

Cultural and Legal Sensitivities:

The research took into account the cultural and legal considerations of the software development and DevOps communities in Bangladesh. The study did not involve any practices that could be construed as promoting or facilitating unethical behavior in software development. Instead, it aimed to explore the challenges, successes, and insights related to IaC practices to inform policy, academic discussions, and best practices within the industry.

Institutional Approval:

This study was reviewed and approved by the Institutional Review Board (IRB) of BRAC University to ensure compliance with ethical research standards.

Transparency and Reciprocity:

Participants were informed about how their data would be used and how the findings might contribute to academic, social, or policy-level discussions regarding IaC practices. Efforts were made to ensure that participants felt respected and valued throughout the research process, and the results of the study will be shared with participants and relevant stakeholders to ensure transparency and reciprocity.

By adhering to these ethical principles, this study aims to contribute meaningfully to the academic understanding of Infrastructure as Code practices in Bangladesh, while upholding the highest standards of integrity, confidentiality, and respect for the professionals whose experiences are central to the research.

Abstract

The DevOps paradigm promotes automated software engineering practices such as infrastructure as code (IaC) to rapidly deploy software to the end users. IaC focuses on automatically provisioning and managing computing infrastructure with machine-readable files. Despite its popularity among organizations, there is a lack of understanding of how the technology is adopted by the software developers. We address this gap by conducting an interview study with 24 professional software developers in Bangladesh who use IaC tools. We report 13 adoption factors and techniques used by these professionals, including compliance, code review, and usage of AI, that have not been reported in the prior literature. We found that IaC has a steep learning curve and in addition to monetary factors such as automation and rapid deployment, non-monetary factors such as transparency play an important role in the adoption of IaC. While these professionals have a preference for open source IaC tools, their transition from one tool to another is influenced by a variety of factors. We conclude by discussing the implications for designing more reliable tools for IaC as well as the need for community-driven efforts to enhance the experience of the practitioners.

Keywords: DevOps; Empirical Study; Configuration Management; Infrastructure as Code; Interview; Thematic Analysis

Acknowledgement

First and foremost, all praise is due to Allah, whose blessings and guidance have enabled me to complete this thesis without any major interruptions.

I would like to express my sincere gratitude to my supervisor, Dr. S. M. Taiabul Haque, for his unwavering support, guidance, and mentorship throughout my research journey. His insightful feedback and encouragement were crucial to the completion of this work. I am truly grateful for his consistent presence and assistance whenever I encountered challenges.

I am deeply thankful to Dr. Akond Rahman from Auburn University for his invaluable contributions to my work. His expertise and guidance greatly enhanced the rigor and depth of my research.

Additionally, I am grateful to Pratyasha Saha, from whom I learned the intricacies of qualitative data analysis. Her guidance in this area was instrumental in shaping the analytical approach of my research. I truly appreciate her support and dedication throughout this process.

My heartfelt thanks go to my defense panel members, Dr. Farida Chowdhury and Dr. Md. Sadek Ferdous, for their constructive feedback and insightful suggestions. Their careful review of my thesis and valuable input strengthened my research and helped bring it to completion.

I would also like to extend my deepest appreciation to my parents. Their constant support, encouragement, and prayers have been the backbone of my academic journey. Without their love and dedication, this achievement would not have been possible.

This thesis would not have reached its conclusion without the collective effort, guidance, and support of all these incredible individuals. I am deeply indebted to each of them.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AKS Azure Kubernetes Service

$ARMTemplate$ Azure Resource Manager templates

AWS Amazon Web Service

CRD Custom Resource Definitions

DB Database

$DevOps$ Development and Operation

EKS Elastic Kubernetes Service

GCP Google Cloud Platform

$GDPR$ General Data Protection Regulation

GKE Google Kubernetes Engine

HCI Human Computer Interaction

IaC Infrastructure as Code

$K8s$ Kubernetes

LTS Long-Term Support

PV Persistent Volume

PVC PersistentVolumeClaim

QA Quality Assurance

SLA Service Level Agreement

SRE Sight Reliability and Engineering

Chapter 1

Introduction

1.1 Motivation, and the Emergence of Infrastructure as Code

Before the emergence of "development and operations (DevOps)", configurations of computing infrastructure (servers, for example) were generally managed by the system administrators or the members of the operations team [83]. For configuration management, these practitioners used a time-consuming approach where configuration files were developed and executed manually at runtime for each computing instance [34, 54]. With the advent of cloud computing utilities, this manual approach introduced difficulties as it did not scale for organizations that have to manage hundreds of computing instances [59]. To alleviate these difficulties, experts have advocated the use of infrastructure as code (IaC) – a practice that minimizes human intervention [120, 54]. The usage of IaC can increase transparency between teams within an organization [21], which makes it a key enabler for DevOps [121].

IaC is the practice of automatically provisioning and managing computing infrastructure at scale using machine-readable files [75]. IaC is implemented using automated tools such as Ansible, Puppet, and Terraform [95], which reduces the need for human intervention [120, 54] and allows practitioners to manage their computing infrastructure similar to that of managing software source code [59]. Since the advent of cloud-based infrastructure such as Amazon Web Services (AWS) and Microsoft Azure, the usage of IaC has increased, which is expected to reach a global market worth 3.5 billion USD by 2030 [48]. The use of IaC has yielded benefits for organizations operating in different domains, including banking [7], research [89], and retail [88]. For example, IaC helped the Asian Development Bank to save hundreds of work hours and manage thousands of servers [7]. Walmart leveraged IaC to reduce the provisioning time from four weeks to hours for its 49,000 servers that are used to serve 230 million customers every week [88]. Similarly, CERN uses IaC to automatically set up and manage their computing infrastructure that consists of 15,000 servers and 300,000 virtual machines [131] [89]. Industry experts consider usage of IaC as "*the new normal*" in modern software engineering [83].

1.2 Challenges and the Need for Human-Centered Research in IaC

Despite the growing popularity of IaC, there is a lack of understanding regarding the factors influencing its adoption among software developers, as well as the techniques they use to implement IaC [98]. Addressing this gap is important from a cognitive viewpoint, as the adoption of a technology depends on its ability to enhance professional goals while reducing human effort [25]. Without understanding the adoption factors and perceived ease of use, proper utilization of a technology could be hindered [25]. Furthermore, if human factors such as adoption and technique usage are not considered, IaC may suffer from the 'discoverability problem'—the phenomenon where practitioners are unaware of tools that could aid in solving computational tasks [74, 72, 32]. Grossman et al. highlighted that users often face the issue of not being aware of specific tools or operations available for use [32]. As a result, practitioners might develop their own solutions, leading to potential inefficiencies and unnecessary expenditures [74].

Although researchers have explored coding patterns [103, 99, 117, 135], bugs [112, 107, 100, 96, 93], and testing [68, 45, 37, 119] in IaC, the underlying human factors related to its adoption and use remain understudied. The HCI community has a strong tradition of investigating developers' practices and perceptions in areas such as security and privacy mechanisms [4, 10], tool discovery [73], software updates [52], the Web3 ecosystem [118], AR/VR applications [57], and LLM-based solutions [18, 55]. Likewise, as DevOps evolves as a methodology, understanding human and organizational factors has become increasingly critical, with research shedding light on these issues [91, 132, 109, 53, 85]. However, the adoption factors related to IaC and the associated development practices have not been thoroughly examined.

This study addresses a significant gap in the existing literature regarding IaC adoption and practices, specifically among software developers in Bangladesh. Despite IaC's growing importance, few studies have explored how professionals, particularly in Bangladesh, adopt and implement IaC. Our research aims to fill this gap, contributing to both the HCI literature and global understanding of IaC adoption. By focusing on Bangladeshi professionals, who are actively engaged in the rapidly growing tech industry, we aim to uncover the unique challenges and practices within this specific socio-economic and cultural context. Ultimately, our findings will contribute to the broader body of knowledge, addressing the research void on IaC adoption and practices both locally and globally.

1.3 Research Questions, Objectives, and Key Findings

To this end, we conducted one-on-one interviews with 19 software developers from Bangladesh who use IaC tools at their workplace. Table 3.1 summarizes the demographic information of the participants.

We find that IaC has a steep learning curve and in addition to monetary factors, such as automation and rapid deployment, non-monetary factors such as transparency

play an important role in the adoption of IaC. While these professionals have a preference for open source IaC tools, their transition from one tool to another is influenced by a variety of factors. We report a range of compliance techniques adopted by these participants as well as a diverse set of activities related to quality assurance. Our participants express a lack of confidence in the use of AI assistants and sustainability is not a major concern for them. We also report their nuanced perceptions and activities related to other critical concepts in IaC, such as state reconciliation and configuration drift. Taken together, our work paves the way for the HCI community to gain further insights into this understudied community that works with a burgeoning technology. This, in turn, advances the HCI literature to understand the relationship between the specialized software development professionals and the emerging technologies that they interact with.

Based on the findings of this first study, we observed a recurring theme among the interviewees: Kubernetes was consistently mentioned as a popular container orchestration tool within the DevOps field. Kubernetes, a container orchestration tool, has become essential in managing the complexities of deploying, scaling, and maintaining containerized applications. Participants highlighted Kubernetes not only as an essential tool for automating infrastructure but also as a platform that intersects directly with IaC principles. This overlap is especially notable as both IaC and Kubernetes share objectives in enhancing automation, scalability, and reproducibility within software environments.

The initial study's findings on IaC adoption pointed toward a need for more nuanced understanding of specific tools and techniques that practitioners are using. Kubernetes stood out because, while it operates as a container orchestration tool, it has increasingly been applied with IaC principles to manage infrastructure and deployments. Many IaC professionals are either working with Kubernetes directly or using IaC to provision and maintain Kubernetes clusters, integrating it seamlessly into their workflows. Given its relevance, we recognized the need for a follow-up study focusing explicitly on Kubernetes to understand its role within IaC-driven environments and its unique set of adoption factors, challenges, and user experiences.

The follow-up study has generated a preliminary understanding of Kubernetes-specific practices, providing a complementary layer to our existing insights on IaC. While our initial research sheds light on broad IaC adoption factors, economic considerations, and technical techniques, our Kubernetes-focused follow-up introduces insights into container orchestration challenges, scalability, and best practices unique to Kubernetes. Combining both studies in this thesis allows us to present a more comprehensive view of infrastructure automation, examining both the foundational practices of IaC and the specialized implementations seen in Kubernetes environments.

1.4 Thesis Contributions

***Contributions*:** Our paper makes three contributions. A list of contributions of our paper is as follows:

1. Generates novel findings from an understudied software development community (i.e., professionals who use IaC) regarding their practices;

2. Provides insights into the underlying human and economic factors regarding the usage and adoption of IaC; and
3. Offers a set of recommendations to the HCI and the software engineering communities to address the human loop in IaC.

Through these combined studies, our research now encompasses both the overarching principles of IaC and the in-depth operational specifics of Kubernetes. This integrated perspective is crucial, as it enables us to offer a thorough exploration of the DevOps field, from code-based infrastructure management to the intricate processes of container orchestration. In my thesis, we aim to bring these findings together to provide a holistic view of the current state of infrastructure automation, capturing the evolution of practitioner needs, tool usage, and organizational impacts in the context of DevOps.

This is the first study regarding IaC practices and tools in the context of Bangladesh.

Chapter 2

Related Work

We first provide the necessary background information regarding IaC.

2.1 Background

2.1.1 Background on IaC

The 'development and operations' paradigm, which is also known as DevOps, emphasizes communication and collaboration between teams with diverse responsibilities, such as coding, testing, and deployment [51]. It advocates the utilization of automation practices, such as IaC to reduce the time required to set up necessary deployment infrastructure [148]. IaC is the practice of using automation tools for the purpose of provisioning and managing computing infrastructure at scale through machine-readable files [75]. IaC tools, such as Ansible, Puppet, and Terraform provide programming languages that have their own syntax and semantics to specify information, such as configurations for the desired infrastructure. Usage of these languages is mandatory to implement the practice of IaC and we refer to these as 'IaC languages'. IaC languages are examples of domain specific languages, which is different from general purpose programming languages, such as C and Java with respect to syntax and semantics [136, 94]. Source code files that are generated by using IaC languages are referred to as 'IaC scripts'. IaC tools compile and execute IaC scripts. Figure 2.1 provides two examples of IaC scripts, where one script is developed in Ansible (Figure 2.1a) and the other script is developed in Terraform (Figure 2.1b). Along with developing IaC scripts, the practice of IaC also advocates for well-known software engineering techniques, such as testing.

Figure 2.1 presents examples of IaC scripts to create a file named 'example.txt'. The file is expected to include one sentence: "Hello world!". The same file can be created using two IaC languages, e.g., Figures 2.1a and 2.1b respectively, show code snippets in Ansible and Terraform to create 'example.txt'. We observe syntactical differences between the two languages. For example, in the case of Ansible a 'task', i.e., the fundamental unit to manage infrastructure, is used using `name`. In the case of Terraform, the equivalent of `task` is a `resource`. We also observe, `ansible.builtin.file` to be used in Ansible, whereas `local_file` is used in Terraform. In the case of specifying the path, one has to use `path` in Ansible, whereas

```yaml
- name: Create `example.txt'
  ansible.builtin.file:
      content: Hello world!
      mode: 0755
      path: /temp/example.txt
```

a

```
resource "local_file" "example.txt" {
  content = "Hello world!"
  filename = "/temp/example.txt"
  file_permission = "0755"
}
```

b

Figure 2.1: Examples of IaC scripts to create a file called 'example.txt'. Figures 2.1a and 2.1b presents examples of code developed respectively, in Ansible and Terraform.

in Terraform `filename` is used.

Despite syntactical and semantic differences, all IaC tools conduct state reconciliation, an approach unique to IaC [101, 93, 39]. In the context of IaC, a state represents the software artifacts and their corresponding configurations for the infrastructure of interest. State reconciliation is the process of finding differences between the current and the desired states of the infrastructure [39], and performing provisioning if differences exist. As part of applying state reconciliation, the following activities would be performed:

- *First*, the IaC tool reads the actual state of the target infrastructure;
- *Second*, the tool compares the actual state obtained by querying the infrastructure, and the desired state specified in the script;
- *Third*, if there are differences between the actual and desired states, then the IaC tool will apply changes so that the target infrastructure state is exactly the same as the desired state; and
- *Finally*, the tool will report the changes made to the infrastructure to the user.

In the context of Figures 2.1a and 2.1b respectively, Ansible and Terraform will first determine if 'example.txt' exists within the specified configurations. If not, then the IaC tool will make the necessary changes.

2.1.2 Background on Kubernetes

Kubernetes, an open-source container orchestration tool, has transformed the way organizations deploy, manage, and scale applications. Originally developed by Google, it provides a robust framework for automating the deployment, scaling, and operation of application containers across clusters of hosts.

Kubernetes is designed to address the challenges of managing containerized applications in a microservices architecture.The rapid evolution of software development practices has necessitated a shift towards more agile and efficient methodologies. Kubernetes embodies this transformation by enabling velocity, scaling, infrastructure abstraction, and efficiency in application management.

Together, these principles form the foundation of Kubernetes' appeal, making it an indispensable tool for organizations aiming to innovate rapidly while maintaining robust operational capabilities.

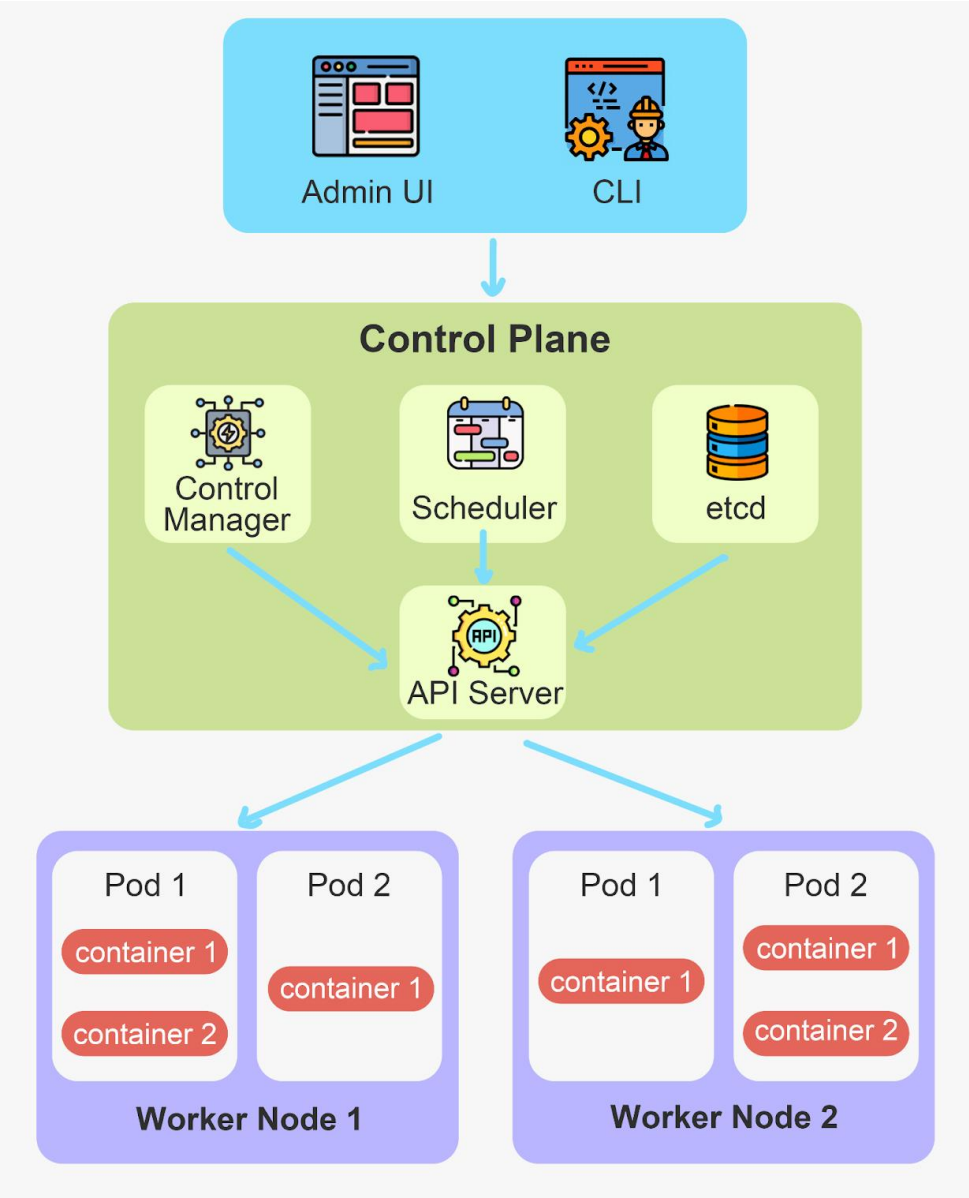


Figure 2.2: Kubernetes Architecture and Key Components.

At its core, Kubernetes leverages containerization technologies like Docker to encapsulate applications along with their dependencies into lightweight containers. This ensures that applications run consistently across different environments, eliminating the "it works on my machine" problem that often plagues software development. The process of deploying a Kubernetes cluster can vary based on organizational needs and infrastructure preferences. Kubernetes architecture is comprised of several critical components that work together to manage containerized applications effectively. Some of them are:

- **Master Node**: The control plane for the Kubernetes cluster. It manages the cluster's state, including scheduling, maintaining the desired state of applications, and scaling operations. It consists of several components:
  - **API Server**: Serves as the frontend for the Kubernetes control plane. It exposes the Kubernetes REST API and is the entry point for all commands.
  - **Controller Manager**: Governs controllers that are responsible for maintaining the desired state of the system (e.g., scaling pods, ensuring deployments are running, etc.).
  - **Scheduler**: Assigns work (i.e., pods) to worker nodes based on available resources and constraints.
  - **etcd**: A distributed key-value store that holds all cluster data, including the configuration, state, and metadata of the cluster.
- **Node**: These are the worker machines in Kubernetes. A node can be either a physical or virtual machine, and each node contains the necessary components to run containers and manage them, including:
  - **Kubelet**: An agent that runs on each node in the cluster. It ensures containers are running in a Pod and reports the node's status to the control plane.
  - **Kube Proxy**: Maintains network rules for pod communication. It manages the network traffic and load balancing to ensure proper connectivity between services and pods.
  - **Container Runtime**: The software that runs and manages containers, such as Docker, containerd, or CRI-O.
- **Pods**: The smallest and simplest unit in Kubernetes. A pod can contain one or more containers that share the same network namespace, storage, and configuration. Pods provide a way to group related containers together and manage them as a single unit.
- **ReplicaSets**: Ensures that a specified number of pod replicas are running at any given time. It automatically replaces any pod that is deleted or terminated, maintaining the desired number of running pods and also resulting in high availability and fault tolerance.
- **Deployments**: A higher-level abstraction over ReplicaSets. It defines the desired state for your application, such as the number of pod replicas, and

helps manage rollouts and rollbacks of updates to applications.

- **Service**: A Kubernetes resource that defines a logical set of pods and a policy to access them. It provides a stable IP address and DNS name to access a set of pods, enabling load balancing and discovery.
- **Namespace**: Used to organize resources within the cluster. Namespaces allow multiple virtual clusters to run on the same physical cluster, providing isolation and resource management.
- **ConfigMap**: A Kubernetes resource that allows you to store configuration data in the form of key-value pairs. This allows decoupling configuration from application code and sharing it among different components of the application.
- **Secret**: Similar to ConfigMap but used to store sensitive information, such as passwords, API tokens, or SSH keys, in a more secure manner. Secrets are encoded, and Kubernetes provides mechanisms for managing and controlling access to them.
- **Ingress**: A collection of rules that allow inbound connections to reach the cluster services. It provides HTTP and HTTPS routing, load balancing, and SSL termination.

These components work together to create a robust and scalable container orchestration system that can manage the lifecycle of applications in a cloud-native environment.

2.2 Related Work

In this section, we summarize prior research related to IaC. Next we highlight the HCI studies related to software development.

2.2.1 Prior Research on IaC

Guerriero et al. [34] is one of the first to advocate for integration of quality assurance into IaC. They interviewed practitioners who use IaC, and identified IaC-related research areas that researchers can focus on. Taking motivation from their work [34], a significant body of work has investigated various aspects of quality assurance for IaC scripts, particularly those written in Ansible, Puppet, and other IaC languages. For instance, research work has identified various code properties associated with defects and maintainability challenges in IaC scripts. Sharma et al. [117] analyzed code properties that contribute to maintainability issues in IaC scripts. Dalla Palma et al. [23] introduced a set of 46 metrics, which includes specific properties of Ansible scripts aimed at detecting defective scripts. Further, Dalla Palma et al. [24] demonstrated that code metrics are more effective than development activity metrics in predicting the defects in Ansible scripts, a finding that contrasts with projects that use general-purpose programming languages [106].

The identification of code smells, i.e., coding patterns that cause maintainability issues, and their impact on IaC scripts has also been a critical area of investigation. Opdebeeck et al. [81] pinpointed various code smells that could lead to defects in

Ansible scripts, while Hassan and Rahman [68] developed a taxonomy categorizing the defects found in Ansible test scripts. Research by Sotiropoulos et al. [120] focused on dependency-related defects in Puppet scripts, and Shambaugh et al. [116] explored non-determinism defects in IaC scripts.

Security weaknesses in IaC scripts have also been a critical area of research. Rahman et al. developed a taxonomy of security vulnerabilities in Ansible scripts and created SLAC, an automated tool for detecting these vulnerabilities [100]. However, Hortlund [40] replicated this study and observed a lower density of security weaknesses than Rahman et al. [100] had reported due to the false positives produced by SLAC. In a follow-up study, Rahman et al. [92] identified the best practices for mitigating security weaknesses, such as hard-coded passwords. Furthermore, educators have integrated these findings into their curriculum to improve learning about security in IaC scripts [102].

Subsequent research has sought to improve the detection of security weaknesses in IaC scripts. Saavedra and Ferreira [112] developed an intermediate representation to enhance security detection in Ansible scripts, while Reis et al. [107] improved a pre-existing security linter for Puppet scripts by incorporating feedback from practitioners, leading to better detection capabilities. Opdebeeck et al. [82] also emphasized the importance of including data flow analysis and control for security vulnerability detection in IaC scripts. Beyond defects and security, other quality issues have been examined. Kokuryo et al. [56] noted that one issue with Ansible script quality was the way external scripts were executed. Additionally, Hu et al. [43] examined the alerts from static analysis for Terraform manifests, contributing to the broader understanding of IaC script quality.

While existing research has extensively explored these aspects of IaC, more recent studies have started to address other critical gaps. For example, Rahman et al. [95] systematically categorized and quantified task infections in Ansible scripts, demonstrating the relationship between task infections and development activity as well as source code metrics. Another study by Hassan et al. [39] demonstrated the defects that occur while implementing state reconciliation in IaC tools. These studies lay a solid foundation for understanding the challenges in managing IaC, particularly concerning task infection and state reconciliation defects, which are crucial for maintaining reliable and secure IaC systems. This current body of literature focuses on finding quality concerns in IaC scripts and tools, but does not address the economics, human factors, and techniques that are related to IaC usage.

2.2.2 Software Configurations

In software configuration, there is a trend of exposing more and more configuration knobs to users. However, there are instances where a lot of these configurations are abstracted away to avoid careless mistakes [145]. One suggestion can be to prioritize the parameters and values set by a more extensive user base over the fewer users [144] after carefully considering the user's configuration statistics. Moreover, much of the state-of-the-art software of the present day has 'latent configuration errors', which are hidden or unnoticed errors in the system configurations that can affect the system's reliability, availability, and serviceability. One possible solution can be to use Document-as-code techniques can be applied to eliminate inconsistencies

between configuration documents and configuration design/implementation [147].

Other solutions include using tools such as Ctest, Ctest4J, AutoBash, and cDep. The idea of Ctest is to connect production system configurations to software tests so that configuration changes can be tested in the context of code affected by the changes. So, Ctests can detect configuration changes that expose dormant software bugs and diverse misconfiguration [126]. Ctest4J, which is a practical configuration testing framework for Java, works faster than the original Ctest because it runs parallel operations instead of sequences. 41.3X times (3.4X on average) [138]. AutoBash works at the operating system level for speculative execution and causality tracking to automate many time-consuming tasks when users deal with the complexity of modern computer systems [125]. Finally, cDep automatically discovers five types of configuration dependencies from bytecode using static program analysis. This tool revealed that configuration dependencies are more prevalent and diverse than previously reported [19].

2.2.3 HCI and Software Development

The HCI research community has extensively investigated a diverse range of issues related to software development, including tool discovery, adoption of new technology, security and privacy concerns, and usability assessment, among others.

Prior research has revealed that discovering new tools typically occurs through peer interaction, peer recommendation, or ad-hoc software testing patches, thus reinforcing the reliance on peer influence and need-based discoveries [73, 52]. Programmers discover tools from their peers both in a co-located and remote settings [73]. A separate study with the software developers reported that tool adoption depends on social environments and communication channels, where developers with strong reputation on the Internet are trusted as much as their own colleagues [143]. Additionally, heightened security concerns can drive the adoption of new tools, especially since security tools are primarily preventive measures [142, 143].

As practitioners adopt new tools, various considerations come into play, particularly within specific domains. One area of concern is within the AR/VR domain, where existing tools may not align with practitioners' differing use cases and frameworks [57]. In the current era of generative AI expansion, LLM-based tools promise to alleviate developers' workloads. However, their benefits are contingent upon the developers' expertise level, as demonstrated in a recent work [18]. ChatGPT-based LLM-tools, lauded for providing well-articulated answers to programming questions, exhibit a significant 50% inaccuracy rate [55]. A recent body of work has investigated the ethical perceptions of the computing professionals regarding AI and LLM-based tools [79, 114]

Another strand of work investigated the role of end users and developers in perceiving and managing security concerns. Research results have demonstrated that, if equipped with frameworks and proper understanding, end users can identify potential security issues and take appropriate measures, thus highlighting their empowerment and responsibility in ensuring overall system security [146, 118]. Furthermore, the erosion of privacy-preserving computations may lead to a comprehensive security approach that impacts both developers and end users [4].

Since the early days of software development, the HCI community has contributed significantly in measuring software usability. Several usability assessment questionnaires and scales have been developed and tested in this regard, including QUIS [20], SSD-SES [137], UBS [124], and SUS [14], among others. Our work is informed by this rich body of HCI literature related to software development and assessment.

2.2.4 Human Aspects in DevOps

Although there is a dearth of knowledge of the associated human factors in IaC, prior research has revealed that human factors play a critical role in both supporting and hindering DevOps adoption. Studies have categorized these factors based on their impact, revealing that effective communication and strong leadership are crucial for overcoming resistance to change [91]. In addition to resistance to change, power relations and the lack of top management support have been identified as the significant barriers to the adoption of DevOps [132]. Furthermore, the complexity of development and production environments presents an additional challenge to the DevOps professionals [109]. While the organizations report positive experiences with DevOps, there is a notable lack of quantitative data to support these claims [29]. Emerging trends such as IaC highlight the need for these organizations to focus on the cultural and social aspects of DevOps, ensuring that teams are well-prepared for these changes [53]. The success of DevOps adoption hinges not only on technical practices but also on creating a collaborative culture that supports continuous improvement [26]. As a result, extending knowledge regarding human factors in DevOps can help tailor adoption processes and improve implementation strategies [85]. Horizontal DevOps teams, which work closely with product teams, have been found to enhance software delivery performance by offering more services and acting as a DevOps Center of Excellence [62]. Finally, the lack of a clear definition of DevOps culture adds to the difficulty of setting clear targets for successful adoption [115]. Our work contributes to this existing body of literature related to DevOps by specifically focusing on IaC—a key enabler of DevOps [121].

2.2.5 Prior Research on Kubernetes

Kubernetes has established itself as the leading container orchestration platform, empowering organizations to manage containerized applications at scale. However, despite its widespread adoption, various aspects of Kubernetes, including performance, deployment methods, custom scheduling, security, and challenges faced by users, have been subjects of intense research. This section reviews key studies on these topics, shedding light on the benefits, challenges, and evolving solutions within the Kubernetes ecosystem.

A significant area of research in Kubernetes involves its performance in different deployment scenarios and resource usage. Telenyk et al. (2021) [129] compared Kubernetes with lightweight alternatives such as K3S and MicroKubernetes. Their study highlighted the trade-offs between these platforms in terms of resource utilization, startup speed, and orchestration performance. They found that while Kubernetes outperforms K3S and MicroKubernetes in most tests, K3S offers better disk utilization, making it a suitable choice for resource-constrained environments. However, MicroKubernetes consistently showed poorer performance across various

metrics, indicating that the lightweight nature of these platforms often comes at the cost of efficiency in large-scale operations. This comparison is crucial as organizations strive to balance the power of Kubernetes with the practical need for lighter, more resource-efficient alternatives in specific use cases.

Another area of focus is the deployment of Kubernetes clusters, particularly for automating software production environments. Poniszewska-Marańda and Czechowska (2021) [86] conducted an in-depth evaluation of Kubernetes deployment methods, comparing popular cloud-based tools like kops and eksctl in AWS environments. Their study emphasized the importance of understanding production environment requirements before choosing a deployment method. Kubernetes' flexibility allows various approaches to cluster deployment, such as do-it-yourself (DIY) or on-premises options, each offering unique advantages and challenges. This work underscores the growing trend of Kubernetes' use in Continuous Integration and Delivery (CI/CD) pipelines, driven by the need to manage microservices.

Kubernetes' default scheduler, while highly configurable, may not be optimal for emerging applications like machine learning or edge computing, which require specialized resource management. Rejiba and Chamanara (2021) [108] explored the limitations of Kubernetes' standard scheduling and reviewed various custom scheduling solutions designed to address the needs of such advanced workloads. Their survey categorized different approaches based on the types of workloads and environments considered, offering a comprehensive overview of the solutions proposed in the literature. They identified significant gaps in current research, providing a roadmap for future studies aimed at enhancing Kubernetes' ability to manage complex, non-traditional tasks.

Carrión (2021) [17] expanded this discussion by focusing specifically on Kubernetes scheduling. In her research, she presented a new taxonomy for Kubernetes scheduling that highlights different Quality of Service (QoS) parameters such as response time, energy efficiency, and resource utilization. Carrión identified ongoing challenges in Kubernetes scheduling, such as ensuring efficient resource allocation and meeting the performance demands of containerized applications. The study also pointed out future research opportunities, including the development of more sophisticated scheduling techniques to improve Kubernetes' performance in diverse environments. This paper contributes to the growing body of research aimed at enhancing the Kubernetes scheduler to handle more complex workloads effectively.

Autoscaling is another crucial aspect of Kubernetes that has drawn significant research attention. Balla et al. (2020) [13] proposed an adaptive autoscaler, Libra, designed to optimize resource allocation for Kubernetes pods. Libra automatically detects the optimal resource set for a pod and adjusts the horizontal scaling process based on changing load and virtualized environments. The scalability of Kubernetes remains a primary concern, and Libra's dynamic adaptation to fluctuating demands enhances the platform's ability to efficiently handle workloads, particularly in cloud environments. This research contributes to improving Kubernetes' autoscaling capabilities, which is vital for maintaining optimal resource utilization in real-world applications.

Security remains a critical concern for Kubernetes users, particularly because the complexity of Kubernetes environments often leads to misconfigurations, which are

the primary cause of vulnerabilities. Morfonios (2023) [69] focused on evaluating Kubernetes' security posture using tools like kube-hunter and Kubescape. The study highlighted how misconfigurations pose significant security risks and demonstrated the potential damage that can be caused by exploiting these flaws. Morfonios also provided insights into mitigating discovered vulnerabilities, emphasizing the importance of secure configurations in protecting Kubernetes clusters. This research is pivotal for organizations aiming to strengthen their security practices and prevent breaches in production environments.

Research on Kubernetes has explored a wide range of topics, from performance and resource utilization to security and custom scheduling. Telenyk et al. (2021) [129] and Poniszewska-Marańda et al. (2021) [86] have contributed valuable insights into the trade-offs between lightweight Kubernetes platforms and more resource-demanding setups, as well as the various deployment strategies available for Kubernetes clusters. In the area of scheduling, studies by Rejiba and Chamanara (2021) [108] and Carrión (2021) [17] have highlighted the importance of custom schedulers to address the needs of advanced applications. Meanwhile, Balla et al. (2020) [13] and Morfonios (2021) [69] have focused on scalability and security, respectively, emphasizing the ongoing need to improve Kubernetes' ability to handle dynamic workloads and mitigate configuration-related risks. Together, these studies form a foundation for future research aimed at enhancing Kubernetes' performance, scalability, security, and adaptability.

While previous research has extensively explored various aspects of Kubernetes, including performance, deployment methods, custom scheduling, and security, there is a noticeable gap in understanding the intersection between Kubernetes and Infrastructure as Code (IaC) principles. While Kubernetes is recognized as a powerful tool for container orchestration, its role within IaC-driven environments has not been fully examined. The initial study in this thesis highlighted Kubernetes' growing importance within the DevOps field and its close alignment with IaC objectives such as automation, scalability, and reproducibility. This study also pointed to a need for a deeper understanding of the specific tools and techniques used by practitioners in Kubernetes environments. To address this gap, this thesis presents a follow-up study that focuses on Kubernetes' adoption, challenges, and practices within IaC workflows. By integrating insights from both the broader IaC adoption literature and the Kubernetes-specific follow-up study, this thesis contributes to a more comprehensive understanding of infrastructure automation, shedding light on both general IaC principles and the specialized approaches that Kubernetes introduces.

Chapter 3

Methodology

Our study focuses on software development professionals from Bangladesh, a region that experiences rapid technological growth. Through in-depth interviews, we aim to gain insight into how IaC tools are used in professional environments, the challenges faced by practitioners, and the broader implications of IaC adoption. In this section, we outline the research methodology used to collect and analyze the data for this study.

3.1 First Study Methodology

Our first study is based on in-depth interviews with 19 software development professionals from Bangladesh who utilize IaC tools and technology in professional settings. We describe the methodology of our work in the following subsections.

3.1.1 Recruitment of Participants

Our participant pool (n=19) includes a diverse range of software development practitioners, including DevOps engineers, software engineers, service managers, cloud engineers, technical team leader, and engineering country leader. Our selection criteria were the following:

1. The participant must have at least one year of experience in working with IaC in a professional capacity at their workplace; and
2. The participant must have used at least one IaC tool in a professional setting.

Table 3.1 provides detailed demographic information about the participants. Our recruitment method involved a combination of personal contacts, snowball sampling, and social media outreach. Initially, we began with four participants who were our colleagues or friends. From there, we expanded our recruitment by reaching out to more of our acquaintances and by posting study invitations on Facebook. This approach yielded ten additional participants from our network. We contacted most of the participants through email, with some communication occurring via WhatsApp and Messenger. We recruited the remaining five participants through snowball sampling, where we asked current participants to refer others who fit the study criteria.

Table 3.1: An overview of the 19 interviewees and conducted interviews.

ID	Length	Codes[1]	Recruitment	Industry Exp.[2]	Role	Ansible	ARM Template	AWS Cloudformation	Azure DevOps	Docker	In-house	Jenkins	Kubernetes	Plugins	Terraform
P01	00:32:19	15	Personal Contact	>1	DevOps Eng.	○	○	○	○	○	○	○	●	○	●
P02	00:26:59	16	Personal Contact	>1	SW Eng.	○	○	●	○	○	○	○	○	○	●
P03	00:25:04	16	Personal Contact	>2	SW Eng.	○	○	○	○	○	○	○	○	○	●
P04	00:20:56	18	Personal Contact	>1	DevOps Eng.	●	○	●	○	○	○	○	○	○	○
P05	00:33:03	13	Snowball Sampling	>5	DevOps Eng.	○	○	●	●	○	○	●	○	○	○
P06	00:23:38	21	Snowball Sampling	>4	DevOps Eng.	○	○	●	○	○	○	○	○	○	●
P07	00:22:23	19	Personal Contact	>3	DevOps Eng.	●	○	○	○	○	○	○	○	○	●
P08	00:43:15	17	Personal Contact	>12	Tech. Lead	●	○	○	○	○	○	●	○	○	○
P09	00:36:18	15	Personal Contact	>10	Country Lead of Eng.	○	○	○	○	○	●	○	○	○	○
P10	00:27:10	22	Snowball Sampling	>2	SW Eng.	●	○	○	○	○	○	○	○	○	●
P11	01:26:09	24	Personal Contact	>2	DevOps Eng.	○	○	●	○	○	●	○	○	○	●
P12	00:25:13	18	Personal Contact	>2	DevOps Eng.	○	○	○	●	●	○	○	○	○	○
P13	00:25:13	8	Snowball Sampling	>1	SW Eng.	○	○	○	●	●	○	○	○	○	○
P14	00:29:54	19	Personal Contact	>1	DevOps Eng.	○	●	○	○	○	○	○	○	○	●
P15	00:21:13	16	Personal Contact	>1	DevOps Eng.	●	○	○	○	●	○	○	●	○	●
P16	00:09:42	17	Personal Contact	>1	DevOps Eng.	●	○	○	○	○	○	○	○	○	●
P17	01:00:40	18	Personal Contact	>8	DevOps Eng.	●	○	●	○	○	○	○	○	○	●
P18	00:30:00	10	Personal Contact	>7	SW Eng.	○	○	○	○	●	○	○	●	●	○
P19	00:21:13	16	Personal Contact	>5	SW Eng.	●	○	●	○	○	○	○	○	○	●
Sum		318				8	1	7	3	4	2	2	3	1	12

● Used the corresponding IaC tool. ○ Did not use the IaC tool.
[1] Codes assigned to the interview transcript [2]. [2] Professional experience measured in years.

Despite our efforts to recruit as many participants as possible, several practitioners declined to participate due to their company policy, which requires official permission and confidentiality agreements. We also could not find any female participants. We highlight the limitations of our study in Section 5.2.

3.1.2 Interview Guide

The semi-structured interviews were guided by a questionnaire [1], which was designed to minimize cognitive load and encourage participants to deeply reflect on each topic. The interview included open-ended questions and started with demographic questions, such as occupation, position within the company, and years of job experience. Participants were also asked about their involvement in developing source code. The core of the questionnaire focused on IaC usage within their companies, both by themselves and their colleagues. Participants were asked how they employ IaC, whether exclusively for setting up and managing cloud instances or for broader IT automation tasks such as user account management or package management. Discussions often centered on recent or ongoing projects, through which we explored tool usage, cloud vendor preferences, technology stacks, government regulations and compliance, and collaborative approaches. Participants were also asked about their preferences for IaC-related tools, and previous transitions between IaC tools, and how they handle state management and configuration drifts. Addition-

ally, some follow-up questions were asked during the interviews to explore specific topics of interest further.

We asked additional questions to understand the perceived environmental impact of IaC, the use of AI tools in developing IaC scripts, and the application of software engineering techniques for bug finding. Finally, participants were asked about the challenges to recruit practitioners with decent IaC skills and were invited to share any additional thoughts on IaC in the context of Bangladesh and similar developing countries.

3.1.3 Interview Sessions

Interviews were conducted online in Bengali using video conferencing tools such as Google Meet or Zoom. In order to be able to conduct in-depth interviews in a stress-free environment, we conducted all the interviews outside the office hours. To compensate for their time, each participant received a compensation of either 1,000 Bangladeshi Taka (BDT)[1], or a gift card of 10 USD, based on their preference. Before each interview, we explained the purpose of the study and asked for their permission to audio record the interview. The interviews varied in length, lasting between 10 to 90 minutes, with an average duration of 31.6 minutes, yielding a total of 10 hours of audio recordings. These recordings were transcribed in Bengali and then translated into English for analysis.

3.1.4 Interview Analysis

We conducted thematic analysis [22] on our data, taking an inductive approach. Inductive thematic analysis allows for flexibility as the researchers do not have pre-determined codes and themes and do not have to fit the data into any pre-existing framework [78]. Each participant's data was coded by two independent researchers. Both of them were born and raised in Bangladesh. They developed codes from the data of the first five participants, compared those codes, and then iterated with more participants' data until a consistent codebook was developed. Once the codebook was finalized, two researchers independently coded the remaining participants' data. After all participants' data were coded, each researcher spot-checked the other's coded data and found no inconsistencies. Finally, we organized our codes into higher-level themes.

3.2 Second Study Methodology

Our second follow-up study is based on pilot interviews with 5 software development professionals from Bangladesh who utilize Kubernetes tool in professional settings. We describe the methodology of our work in the following subsections.

3.2.1 Recruitment of Participants

Our participant pool (n=5) includes a diverse range of software development practitioners, including DevOps engineers, software engineers, service managers, cloud

[1] 1 USD = 120 BDT

engineers, technical team leader, and country lead of engineering. Our selection criteria were as following:

1. The participant must have at least one year of experience in working with Kubernetes in a professional capacity at their workplace

Table 3.2 provides a detailed demographic information about the participants. Our recruitment method involved a combination of personal contacts, and participants from our previous study. We began with two participants who were our colleague and particpant from our previous study. From there, we expanded our recruitment by reaching out to more of our acquaintances. This approach yielded 3 additional participants from our network and through snowball sampling. We contacted all of the participants through email.

Despite our efforts to recruit as many participants as possible, several practitioners declined participation due to their company policy, which requires official permission and confidentiality agreements. We also could not find any female participants, likely due to the gender imbalance in the field, where male practitioners are more prevalent. We highlight the limitations of our study in Section 5.2.

3.2.2 Interview Sessions

Three interviews were conducted online in Bengali using video conferencing tools like Google Meet, while two interviews were conducted in person. To compensate for their time, each participant received a compensation of either 1,000 Bangladeshi Taka (BDT)[2], or a gift card of 10 USD, based on their preference. Before each interview, we explained the purpose of the study and asked for their permission to audio record the interview. The interviews varied in length, lasting between 27 to 55 minutes, with an average duration of 40.3 minutes, yielding a total of 2.7 hours hours of audio recordings. These recordings were transcribed in Bengali and then translated into English for analysis.

3.2.3 Interview Analysis

We conducted thematic analysis [22] on our data, taking an inductive approach. Inductive thematic analysis allows for flexibility as the researchers do not have pre-determined codes and themes and do not have to fit the data into any pre-existing framework [78]. Each participant's data was coded by two independent researchers. Both of them were born and raised in Bangladesh. They developed codes from the data of the first five participants, compared those codes, and then iterated with more participants' data until a consistent codebook was developed. Once the codebook was finalized, two researchers independently coded the remaining participants' data. After all participants' data were coded, each researcher spot-checked the other's coded data and found no inconsistencies. Finally, we organized our codes into higher-level themes.

[2] 1 USD = 120 BDT

Table 3.2: An overview of the 5 interviewees and conducted interviews.

ID	Length	Codes[1]	Recruitment	Industry Exp.[2]	Role
P01	00:35:30	23	Personal Contact	>6	Senior Solution Cloud Architect
P02	00:54:45	31	Personal Contact	>3	Site Reliability and Eng.
P03	00:27:25	21	Personal Contact	>2	Senior SW Eng.
P04	00:27:25	21	Personal Contact	>2	SW Eng.
P05	00:43:26	23	Personal Contact	<1	DevOps Eng.
Sum		119			

[1] Codes assigned to the interview transcript [3]. [2] Professional experience measured in years.

3.3 Ethical Considerations and Positionality

Our study protocol was reviewed and approved by the local institutional review board (IRB). The interviews were conducted by following the ethical guidelines on research involving human subjects. All participants provided informed consent prior to the interviews and were made aware of their right to withdraw at any point. All the data were securely stored and accessible only to the research team, with no personally identifiable information of the participants. Our research team includes an experienced academic researcher on DevOps/IaC as well as a professional local DevOps engineer, and as such, we are familiar with the global and local trends in the IaC industry.

Chapter 4

Findings

We report our findings from both research in this section. First, we will discuss the findings from the first study, which can be organized into three broad categories: economics, techniques, and state reconciliation. Then, we will discuss the findings from the second study. Table 4.1 provides the definitions of the terminologies mentioned by our participants.

4.1 Economics

4.1.1 Expense, Revenue, and Local Conditions

In our first study, we found that basic economic considerations as well as the local conditions play an important role in the use and adoption of IaC. P10, P12, P13 , and P18 mentioned the expense of IaC and generated revenue when discussing IaC usage. P12 particularly referred to the price of IaC tools for being a deterrent for adoption, "*There is not much advancement with respect to IaC usage because of the high initial setup cost for tools.*" According to P18, "*Investment in IaC is non-profitable for a company whose net profit is 5,000 dollars in a year.*" P02 mentioned the size of the infrastructure that the organizations manages, "*I don't think every company in Bangladesh needs IaC*", P02 stated, "*For companies that have a small infrastructure, adoption of IaC will lead to extra developer overhead or extra time to maintain developed IaC scripts, which I don't think is worth the effort.*"

As a practitioner working in a developing country, P10 mentioned the lack of relevance of IaC for a developing country. He added, "*As far as I know, there are not too many companies in Dhaka (the capital city of Bangladesh) that have adopted IaC because they [the companies] do not want to invest in IaC-related tooling and knowledge base.*" P10's views related to maturity are also echoed by P13 who finds his country's IT industry to be not as mature as China's or India's. P10 also mentioned the lack of international payment options in their country, "*If international payment becomes easy in Bangladesh, then it is likely that companies [in Bangladesh] will find fresh graduates that know DevOps, are interested in DevOps, and can design infrastructure with IaC.*" P14 mentioned that government-based organizations might be less willing to use IaC-based tools due to the security concerns associated with managing sensitive data.

Table 4.1: Terminologies mentioned by the professionals and their definitions. The table is sorted alphabetically based on the terminology.

Terminology	**Definition**
AI assistant	An automated agent that uses artificial intelligence (AI) to complete software engineering tasks, such as code generation [63].
Code review	The technique of using tool or peers to obtain feedback on developed source code [50].
Compliance	The practice of a team or an organizations' adherence to a set of recommendations set by a stakeholder, a federal authority, or the organization itself [61].
Configuration drift	The phenomenon when systems that were initially identical become different over time due to manual changes or inconsistent updates, making them harder to manage and automate, causing inconsistencies across the infrastructure [70].
Continuous deployment	A process where incremental software changes are automatically tested and frequently deployed to the end users of the software [104].
Debugging	The activity of identifying the root cause of software bugs [36].
Flow Testing	The testing activity that focuses on the functionality of the entire software instead of testing individual portions of the software [6].
Integration Testing	The activity of testing a component of a software that includes multiple smaller units [47].
Logging	The technique of generating, transmitting, storing, analyzing, and disposing log data from software artifacts [76].
Microservice	A software engineering approach for software organization where the software is composed of small independent services [77].
Software build	The process of translating source code, libraries, and data files into a set of deliverables, such as executables for end-users [66].
Software package management	The process of compiling, installing, upgrading, and removing computer programs [122].
Testing	The technique of executing a software or a software component under specified conditions, recording the execution results, and performing an evaluation based on the recorded results [49].
Unit Testing	The activity of testing a small unit of a software [47].
Version control	Version control is the technique of managing different versions of computer files using a dedicated archive, which is referred to as a repository [50].

Usage and transition of tools We found that Terraform and Ansible are the two most popular tools among our participants. Table 3.1 shows the list of the IaC tools used by our participants. As can be seen from Table 3.1, there is a preference for free, open source tools among the IaC users in Bangladesh. Among the eight IaC tools in Table 3.1, five are open source while ARM Template and Azure DevOps – two proprietary tools in the list – are used less frequently as AWS CloudFormation is the only popular proprietary tool among our participants. According to P02, "*As far as I know, there is a premium version of Terraform. But it is a bit expensive, which is why we have not decided to use it yet.*" This suggests that cost is a very important consideration for the local IaC industry.

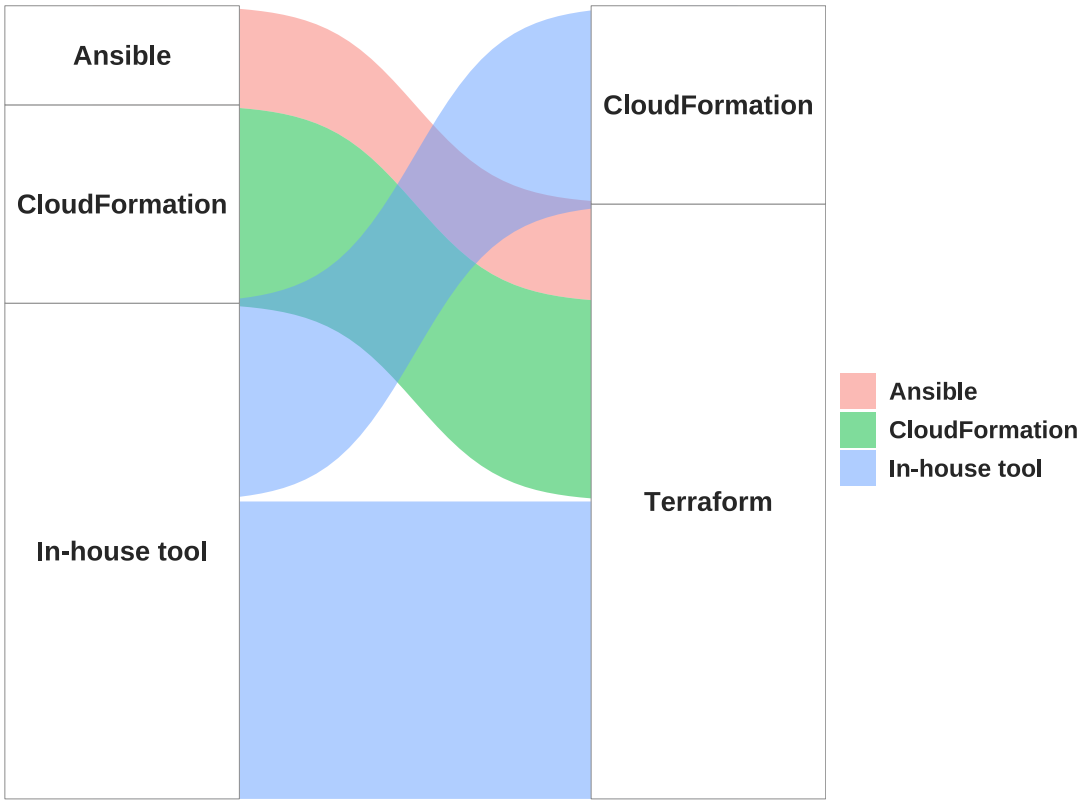


Figure 4.1: Frequency of transitioning from one IaC tool to another.

The transition from one IaC tool to another, however, depends on a variety of factors. Our participants mentioned how they had started with one IaC tool, and then transitioned to another one because of organizational requirements. P06 mentioned the demand of cloud–agnostic solutions, where an IaC tool is not tied to a cloud-based infrastructure. For example, AWS CloudFormation works only for AWS-based cloud infrastructure, and not for other vendors, such as Azure. According to P06, "*Initially, we were using CloudFormation but it is not cloud-agnostic as we could only use it for AWS, even though it was an IaC tool. Now that we have transitioned to Terraform, we can use IaC to deploy in Azure, Alibaba, and GCP—everywhere, anywhere, independent!*" P04 mentioned compatibility with existing developer workflows when describing their transition from Ansible to Terraform. P10 mentioned how Terraform is better at state reconciliation compared to that of

an in-house IaC tool, which triggered the transition from the in-house IaC tool to Terraform. According to P10, this transition also reduced effort for the team, "*In the case of our in-house IaC tool, we had to do the necessary maintenance. Now, that we have switched to Terraform, that overhead is gone, as we do not have to manage the IaC tool ourselves.*" Figure 4.1 summarizes the reported transitions from one IaC tools to another.

4.1.2 Economics of Adoption

When discussing software economics, Boehm and Sullivan argued that along with monetary values, there also exists non-monetary values for developing and adopting software systems [16]. Our interview analysis supports this statement as we observe our participants to highlight both monetary and non-monetary factors that facilitate the adoption of IaC. Figure 4.2 presents the frequency of adoption factors, as mentioned by our participants.

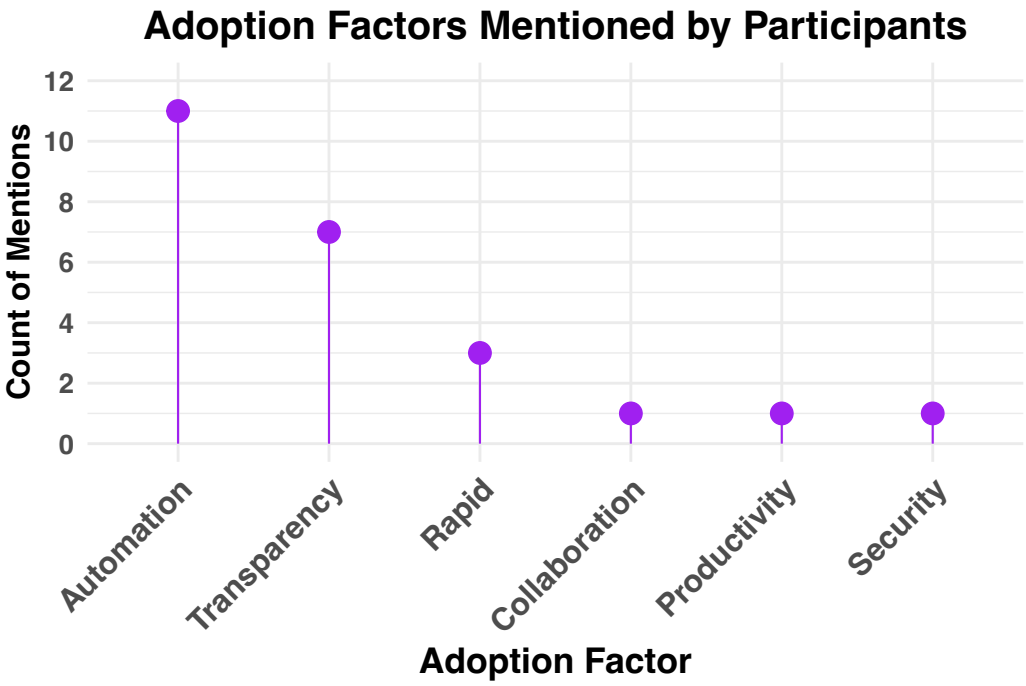


Figure 4.2: Frequency of mentioned adoption factors for IaC.

Monetary factors for adoption : We identify four monetary factors of adoption from our data: automation, rapid deployment, productivity, and security. Automated management of computing infrastructure reduces manual effort and saves organizations time and money [87]. Eleven participants talked about automation. P12 mentioned "*It reduces repeatability. I need to do the same thing repeatedly: add a feature, deploy it to test, deploy it to train, deploy it to production, test it again—doing the same thing repeatedly for any feature addition. It reduces this repetition and also decreases the need for human resources. Human errors are almost zero because everything is automated. In this sense, IaC has helped us a lot. It has removed a lot of the tension that would otherwise stay in our minds during operations.*" And P5 elaborated this factor, "*I prefer Ansible when I need to set up the*

same configuration on 15/20 servers at the same time. Instead of manually working on those 20 servers repeatedly, I just prepare the Ansible script, prepare the playbook, run the playbook, and that's it. Your infrastructure will automatically be ready on those servers, and the provisioning you want will be available." Similarly, rapid deployment leads to the reduction in deployment time of software changes, which, in turn, increases the revenue. P1 said "*You could say that IaC helps us with faster deployment. Through IaC, we're able to speed up the deployment process.*" And P10 also mentioned "*I think using IaC speeds up infrastructure provisioning, making it seamless. This could be a benefit, as server runtime may be reduced.*"

Productivity and security can be considered as monetary factors as well because productivity is directly related to monetary gains [15], whereas security vulnerabilities can result in the loss of revenue [128]. P1 said "*In terms of security, IaC is a very good thing. The security rules we want to impose can be easily configured in IaC. Something that would have been a hassle to do manually can be easily introduced through IaC from a security improvement perspective. So, from a security improvement standpoint, IaC is a very nice thing to have.*" However, as can be seen in Figure 4.2, these factors were least frequently highlighted by our participants.


```
"instances": [
   {
        "attributes": {
            "ami": "ami-0beaa649c482330f7",
            "availability_zone": "us-east-2c",
            "id": "i-instanceid",
            "cpu_core_count": 1,
            "cpu_threads_per_core": 1,
            "instance_state": "running",
            "instance_type": "t2.micro",
            "monitoring": false
        }
    }
]
```


Figure 4.3: An example of state file used for Terraform. The file is in JSON format and stores information of the infrastructure at a certain timestamp. For example, we observe one AMI instance to be provisioned. Eight pieces of information about the infrastructure are represented: 'ami', 'availability_zone', 'id', 'cpu_core_count', 'cpu_threads_per_core', 'instance_state', 'instance_type', and 'monitoring'.

Non-monetary factors for adoption We also identify two non-monetary factors, namely collaboration and transparency. For example, P11 mentions how the "state file" of Terraform can facilitate collaboration: "*Wherever you or your team members are, you can work on the shared state file concurrently and collaboratively as it is shared with everyone.*" The "state file" in Terraform stores all the states of

the infrastructure. Figure 4.3 showcases an example state file where the information of an Amazon Machine Image (AMI) instance is documented. An AMI is an artifact that contains the source code needed to set up and start an Amazon EC2 instance [5].

Our participants also highlighted how IaC allows transparency of the desired infrastructure by specifying configurations of the infrastructure as code. As P14 said, "*From the very beginning, if my infrastructure is based on IaC, then later on it becomes very easy to manage. Furthermore, even if someone leaves the organizations, someone unfamiliar with the code can understand the specifics of the infrastructure.*" This view was also echoed by P03 who finds IaC to prevent the culture of 'unfireable' employees: "*I have seen that if there is only one employee who knows how to manage the infrastructure, then it is bad from a business perspective. The employee then becomes 'completely unfireable', and [hypothetically] can delete the entire infrastructure if they are disgruntled. However, if infrastructure as code is used, then we do not have to rely on that one employee as we can just launch the entire infrastructure using code.*"

4.1.3 Economics of Learning

Khetri [58] pinpointed learning to be a mandatory economic factor when using open source operating systems, such as Linux. From our interview analysis, we found four participants talking about learnability. P10, P12, P13, and P18 mentioned the steep learning curve of IaC when it comes to usability. P18 highlighted the technical aspects of IaC by citing Kubernetes as an example, "*Anyone can pull and push a Docker image. However, it is pivotal to understand what happens under the hood. For example, in the case of Kubernetes, one needs to understand the differences between a 'node' and a 'pod'. One also needs to understand what a control plane is in Kubernetes, just acquiring basic knowledge is not enough.*" P13 also mentioned the time that one needs to invest when designing IaC-based infrastructure, "*When you want to do something new, you have to spend a lot of time for designing the infrastructure. So initially, you need time, along with good expertise.*" Except for P05 , all the other participants acknowledged that there is a lack of expertise in IaC in the local job market. P12 further elaborated the underlying economic implications, "*Finding and recruiting good DevOps engineers is very tough as the learning curve for IaC is costly. If I need to learn something like Azure DevOps, then I will need premium subscription and pay as you go – all of which I have to pay, meaning that it's costly.*"

To mitigate this, our participants adopt different strategies. P02, P03, and P19 mentioned "self learning", whereas P06, P11 and P14 talked about community-based efforts, such as blog posts and LinkedIn certifications to aid in learning IaC. P11 described his experience in writing blog posts related to IaC, "*There is a good community where I feel like, people are blogging all the time, even I myself have written blogs related to DevOps, IaC etc. I can share my experiences and code using my blog posts.*" P06, however, stated that even though there are community-driven efforts, these are not enough as they "*do not get the 'actual help' that is needed at the professional level.*" All these responses indicate that the professional community is still at a nascent stage where the individual developers need to put a considerable

effort to overcome the steep learning curve of the IaC tools and technology.

4.1.4 IaC and Sustainability

As the advent of IaC is related with the proliferation of cloud-based infrastructure [21], we asked our participants about certain aspects of IaC development that can lead to energy efficiency. Green cloud is defined as cloud-based infrastructure that does not consume unwanted energy [140]. According to Talukder et al. [127] green cloud can contribute to the "*larger social objectives of energy efficiency and environmental protection and sustainable development.*"

In response, we found only P12 out of all 19 participants to acknowledge the issue of energy consumption, "*The cloud computing stack that we use involves giant servers, which can lead to unnecessary power consumption. The more power it [cloud-based server] consumes, the more fossil fuels are burnt.*" However, the majority of the participants did not acknowledge the relationship between IaC and sustainability. For example, P9 said "*No, I don't think so. Even if there is an impact, it is so insignificant compared to the other things that we are doing to the environment, which makes me believe that there is no negative effect.*"

4.2 Techniques

We found that our participants utilize IaC for a wide range of automation tasks. Our interview analysis also reveals a diverse set of techniques that the participants use for working with IaC tools. We provide an in-depth description of these techniques below.

4.2.1 IaC and Automation

We first provide an overview of the computing infrastructure for which IaC is used.

Computing Infrastructure Provisioned with IaC In our interviews, practitioners mentioned two categories of computing infrastructure: on-prem and cloud-based. On-prem computing infrastructure refers to the physical servers whereas cloud-based infrastructure relies on cloud service provider such as Amazon Web Services (AWS) and Google Cloud Platform (GCP). We observed the majority of the participants to use cloud-based computing infrastructure, which is consistent with the existing research that reports that IaC was established as a practice to manage cloud-based infrastructure at scale [21]. Among the professionals who provision on-prem computing infrastructure, they mainly do so because of client requirements. For example, P16 explained that their team works with a telecommunication company that solely uses on-prem servers. A complete breakdown of the computing infrastructure can be seen in Figure 4.4.

We found three types of automation techniques: IaC for infrastructure management, IaC for package management, and IaC for user account management.

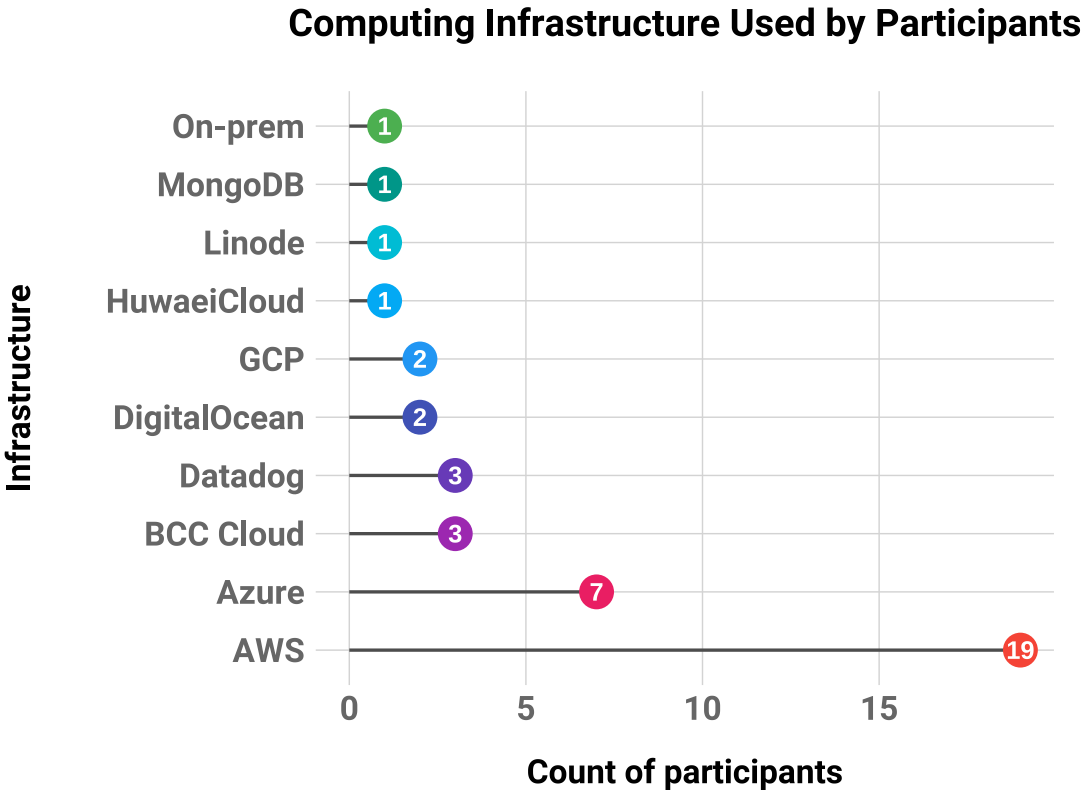


Figure 4.4: Categories of computing infrastructure used for provisioning with IaC.

IaC for Infrastructure Management The majority of the participants reported IaC as a tool for automated provisioning and management of computing infrastructure. This computing infrastructure can be used to perform software builds (P12), implement continuous deployment (P06), manage on-prem servers (P16), and execute microservices (P19). The IaC languages provide a wide range of coding options to provision and manage the desired infrastructure. According to P19, "*As microservices are getting complex. the [non-IaC] tools we used could no longer maintain these microservices. We started the migrations three years ago and adopted these IaC languages in our technological stack. Now, our infrastructure is fully managed with Terraform.*".

IaC for Package Management As software packages have dependencies to resolve, IaC allows practitioners to install and manage software packages through certain coding patterns. We found two participants using IaC for software package management.

IaC for User Account Management We also found two participants P11 and P17 mentioning IaC to be used for account management. P11 mentioned how their team managed user accounts for a RabbitMQ[1] service using Ansible.

4.2.2 Techniques for IaC

Our participants apply multiple techniques for IaC, including compliance, logging, quality assurance, and version control. Figure 4.5 summarizes the frequency of using these techniques.

Compliance Our interview analysis revealed the majority of the participants using some form of compliance techniques when developing IaC scripts. Overall, 15 out of 19 participants reported about different categories of compliance. For example, P01, P02, and P19 mentioned organization-based compliance activities that IaC users need to adhere to. P04 and P11 talked about stakeholder-based compliance where the organization has to adhere to the guidelines set by the customers. P11, on the other hand, highlighted service level agreement (SLA)[2], "*We have a few internal SLAs, not government-based SLA, but set up by customers, such as recording the recovery time from an outage, having a backup recovery policy, and conducting a backup validation check.*"

We found some region specific nuances as well. P07, P14, P17, and P18 talked about regional differences as compliance activities in the context of IaC can vary from one region to another. P18 stated, "*My company is based in Europe, so they [the company] have to abide by a lot of GDPR and relevant security regulations. After developing scripts, the security team provides feedback, which, in turn, is addressed by the IaC script developers.*" Furthermore, P07 mentioned the compliance activities related to Bangladesh stating, "*The Bangladesh government requires that*

[1]RabbitMQ is an open-source message-broker software that is used to convert a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver.

[2]An SLA is an official contract between an organization who provides software-based services and its end-users [141].

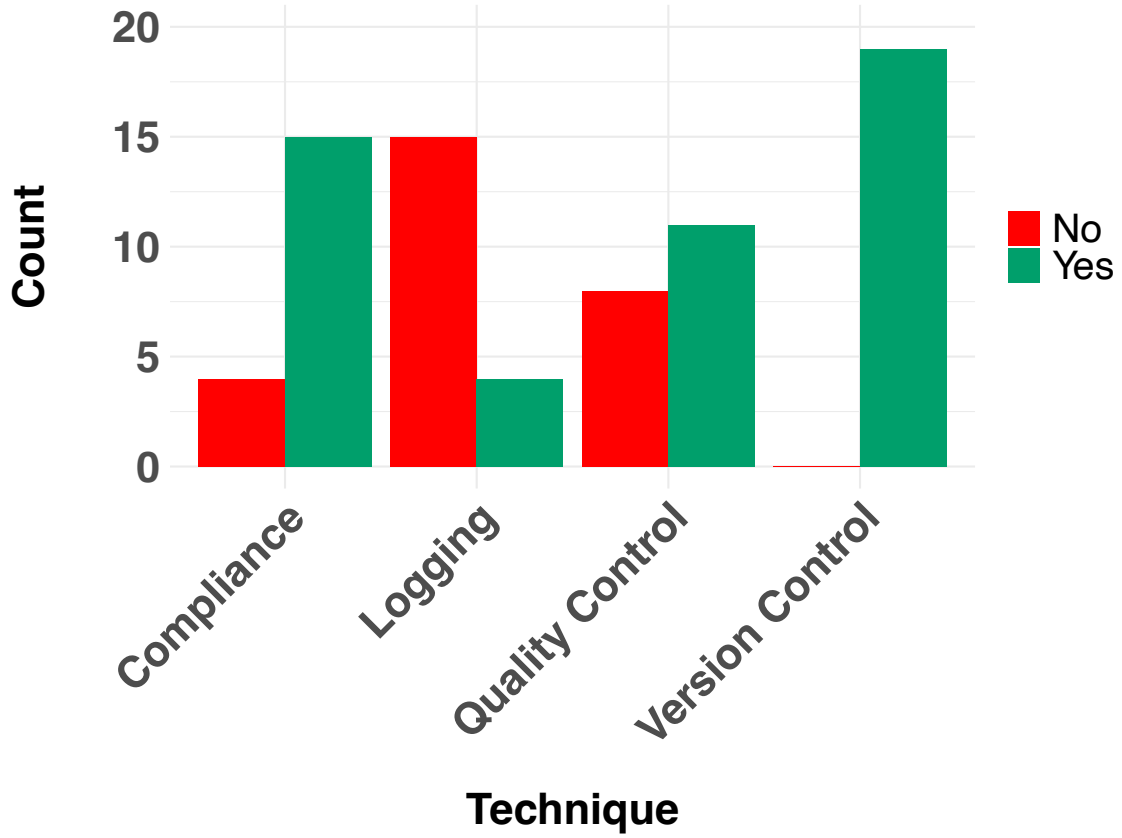


Figure 4.5: Frequency of mentioned techniques used for IaC script development.

any data related to our country must reside in Bangladesh, and as such, tools like Terraform cannot be used in this context". P18 provided an example of the feedback obtained from the security team, "*For example, making sure access tokens are not leaking.*" Thus, our data reflects a wide range of perceptions related to compliance.

Logging Overall, four of our participants talked about logging during IaC script development. For logging purposes, these participants mostly use utilities that come with IaC tools. For example, P19 referred to an Ansible utility where a user can provide the `-v` flag from the command line. P11 cited a few examples for two categories of tools: Elastisearch and Opensearch for log collection, and Datadog, Grafana, and Prometheus, for log analysis. P08, on the other hand, described their experience in developing and using a custom logger that they built and used to collect logs. Logging is also mentioned as a bug finding technique by three participants: P08, P09, and P11.

Quality Assurance (QA) Our interview analysis also provides insights on the activities that are conducted for quality assurance (QA) of IaC scripts, which we describe below. Figure 4.6 summarizes the frequency of the QA activities.

QA - Code review Five participants, namely P04, P05, P09, P12, and P16, mentioned that they conduct code review for developed IaC scripts. We observe two categories of code review approaches:

1. Peer-based: P05 highlighted a peer-based technique: "*After the changes are*

committed, a pull request is created that triggers a push review. We have one mandatory reviewer and two/three optional reviewers who review the committed changes. On top of that, we have a 'technology lead' who reviews the submitted code changes." The approach is manual where no tools are used.

2. Tool-based: In this case, there is no human involvement. P16 mentioned two automated static analysis tools in this regard: "*We use two scanners, you may have heard about them: 'Trivy' and 'Sonarqube'. So, these are the things that we use in the case of conducting code reviews.*"

QA - Debugging: Only one participant (P6) mentioned the use of a tool called for AWS CodeDeploy [3] for debugging.

QA - Testing: Eight of the participants explicitly spoke about the testing of IaC scripts. P01 finds testing to be "*a challenging matter.*" To alleviate this challenge, P01 thinks that the testing tools provided by Terraform to be helpful for unit testing, "*To find bugs in IaC scripts, we write test cases similar to the way we write test cases for general purpose programming languages. There is something called 'TF Test', which is very nice. While this is available as part of the Terraform installation, it [the testing tool] is not available for Ansible or CloudFormation.*" Similarly, P08 and P10 also reported about unit testing. Furthermore, P08 and P10 respectively, mentioned integration testing and flow testing. As IaC scripts are used for deploying microservices, P10 mentioned the usage of flow testing where a tester can test the flow of steps that a microservice should go through. According to P10, "*We use Testim [130], which is a zero code or almost zero code platform that allows a practitioner with little software testing experience to set and test the flows.*"

[3] https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/aws-codedeploy.html

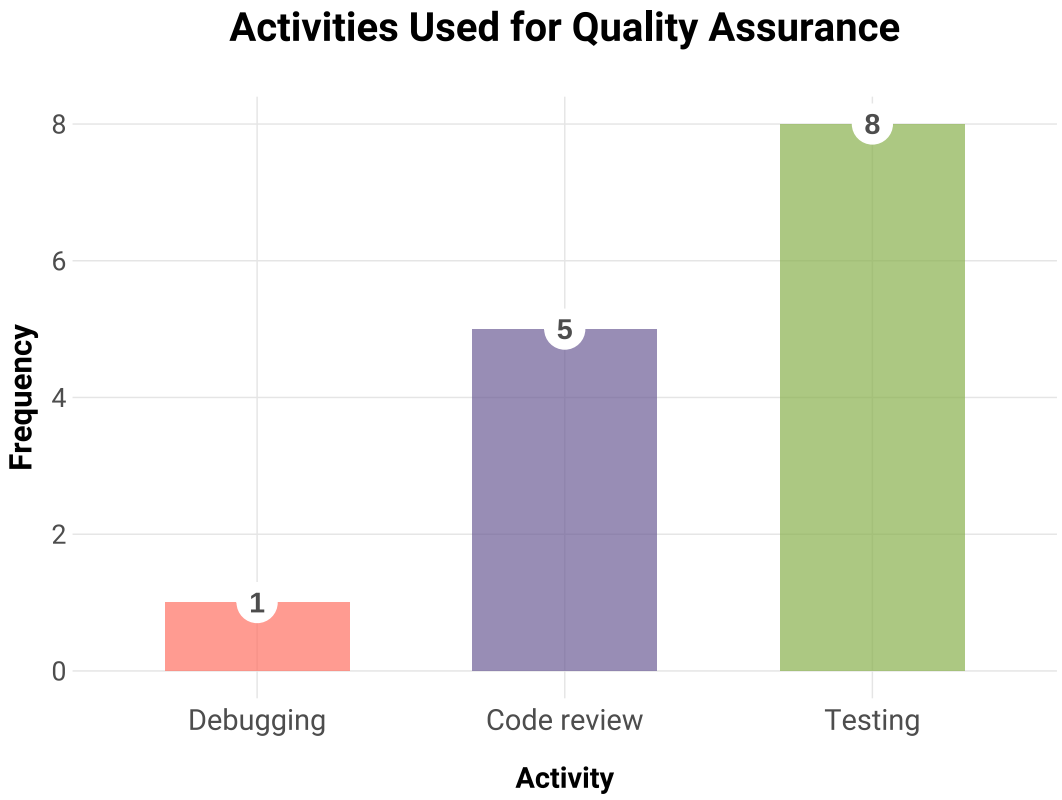


Figure 4.6: Frequency of quality assurance activities mentioned by our participants.

Usage of Version Control The technique of version control is implemented using tools such as GitHub and GitLab. All of our participants reported about version control tools being used for storing IaC scripts, and except for one, all the other participants reported the utilization of a private version control repository. As for the reason for keeping IaC-related repositories private, the majority of the participants attributed to organizational policy. P04 specifically talked about a particular security issue: "*The access keys reside in the repository, so definitely it cannot be made public.*" Some teams take additional steps in keeping their IaC repositories private. P04 stated, "*We do not even keep it [the private repository] on a public network.*" According to P17, "*This [IaC repository] is something that outside of the infrastructure team, no one will get access to; not even other teams within the organization, such as the application developer team.*" We discuss the security implications of these findings in Section 5.

4.2.3 Usage of AI Assistants for IaC

The advent of artificial intelligence (AI) assistants that generate code has created a profound impact in software engineering [60]. These tools are typically trained on a large corpus of source code and can generate code automatically based on the given prompts about a certain computational task [110]. From the responses of our participants, we found that 12 out of 19 of them use AI assistants, such as ChatGPT, GitHub Copilot, and Google Gemini. The remaining seven participants cited

several reasons for not using AI assistants, including organizational policies (P09), AI assistants' inadequacy to solve IaC-related automation tasks (P03), and poor quality of the generated IaC code (P02). Additionally, P02 offered a nuanced perspective in this regard, "*The code generated by AI assistants for traditional software engineering is safer as there are many testing mechanisms. In comparison, I'm not much aware of the testing mechanisms that would allow me to test AI-generated code for IaC.*"

The lack of confidence in AI assistants was also echoed by P11, who had used an AI assistant for IaC in the past. According to P11, AI assistants are "*not good enough in regard to generating a full production grade solution by retaining the full context of the project of interest.*" Instead, P11 finds AI assistants to be better suited for documentation-related support. We obtained similar responses from P15 and P17. P18 highlighted the usefulness of AI assistants in syntax-related assistance, "*The main reason of using Copilot is that you have to write a lot of boilerplate code in IaC. When writing these code, one could make a spelling or a syntax mistake, which Copilot fixes*". P15 described an additional issue when using AI assistants for generating Terraform code. According to their observation, generated code from AI assistants can include sensitive information such as access keys that can be used to login to an AWS cluster. AI assistants (ChatGPT, for example) are trained on large corpora of publicly-available source code on code sharing platforms like GitHub and StackOverflow [9]. These codes often include sensitive information, such as hard-coded passwords and access keys, which, in turn, is used for training the AI assistants [84]. As a result, when code is generated by the AI assistants, they can also include sensitive information [84]. According to P15, these sensitive information have to be removed prior to execution. In summary, despite being used by the majority of the participants, there are certain limitations of using AI assistants for generating IaC code snippets with regards to functionality and safety.

4.3 State Reconcillation

As clarified in Section 2.1, state reconciliation is a concept unique to IaC. From our interview analysis, we identify three issues related to state reconciliation, which we describe as follows:

4.3.1 Configuration Drift in State Reconciliation

Out of 19 participants, 11 found configuration drift to be a challenge for ensuring correct state reconciliation. Similar to software code, IaC scripts can be modified by multiple practitioners within a certain period of time [97], which can lead to configuration drift. Manually changing the configurations of the desired infrastructure without making changes to the script can also cause configuration drift [139]. As a workaround for configuration drift, we found the participants undertaking multiple initiatives. P02, P07, P10, P11, and P14 talked about monitoring alerts to find if the infrastructure state is inconsistent with the configurations specified in the IaC scripts. P02 provided a detailed example of the alert mechanism, "*The scenario that I was trying to refer to is the case when the code hosted in a GitHub repository has configurations that are not in our cloud infrastructure. In situations like*

this, we use an alert mechanism, through which we periodically know – in every one or two hours – if the state declared in our GitHub repository aligns to that of our cloud infrastructure. If it does not align, then someone manually logs into the cloud infrastructure to fix the misalignment."

Along with alert mechanisms, participants also mentioned manual approaches (P09, P12, P14, P15), custom scripts (P03), and code inspection (P03) to detect configuration drift. P15 mentioned manual intervention of their project leader for state reconciliation, "*What happens in our case is that without manual approval from the project lead, no code revision is deployed. As a result, the chances of errors are minimal.*" In contrast, P09 is not fully confident regarding the effectiveness of manual intervention, "*In the case that there are some changes that seem to take place in the local system or showing in the local system, but not reflected in the state, manual intervention could be required, but I am not 100% sure about it.*"

When discussing code inspection, P03 mentioned a collaborative approach in detecting configuration drifts, "*If there is [configuration] drift, we try to resolve it. In our team, we have many folks, so we can discuss by looking at the code in the script. This is possible as everyone is aware about the full infrastructure.*" Participants also spoke about the utilities that are available as a part of the IaC tools for detecting configuration drifts. P06 and P11 respectively, mentioned 'Terraform import' and 'Terraform plan', to detect configuration drifts for their Terraform-based infrastructure.

4.3.2 "In Tools We Trust!" – Trusting IaC Tools for State Reconciliation

State reconciliation is integral to IaC and this process is automatically managed by the IaC tools such as Ansible, Chef, and Terraform. P01, P02, P16, and P17 explicitly mentioned that they rely on the IaC tools to perform state reconciliation. For example, P17 explained, "*In the case of CloudFormation, states are internally managed by the tool itself. Because of this, we do not have to worry at all about it [state reconciliation].*" Similarly, P02 described how state transitions are handled by the IaC tools, "*Terraform, Pulumi ... these IaC tools act as state machines. They [the tools] have a state file that is used to manage the state. This means that the state, i.e., the configuration of the state, is stored somewhere locally.*" These responses suggest that some of our participants inherently trust the IaC tools to deal with state reconciliation.

4.3.3 Usage of Development Activities for State Reconciliation

We also found some of the participants highlighting development activities related to state reconciliation that are independent of the IaC tools. For example, P07 emphasized the importance of monitoring for state reconciliation, "*Without constant monitoring, you cannot do state reconciliation and understand which state is going where.*" Furthermore, P06, P10, P11, P14, and P15 explicitly described the practice of storing state files and state logs securely in cloud-based storage. As P06 explained, " *We store the state file using a secure S3 bucket right now so that no one*

can destroy it. This state file is pivotal for infrastructure identification, right? That is why I am using a secure multi-factor authentication (MFA)-enabled S3 bucket, which no one can access except for the authorized individuals." Our findings thus reveal the additional measures undertaken by some of the professionals that are related to state reconciliation.

Table 4.2: Example of Open Coding Process of Study-1

Quote	**Initial Code**	**Code**
" I prefer Ansible when I need to set up the same configuration on 15/20 servers at the same time. Instead of manually working on those 20 servers repeatedly, I just prepare the Ansible script, prepare the playbook, run the playbook, and that's it. Your infrastructure will automatically be ready on those servers, and the provisioning you want will be available."	Automation	Monetary factors
" It reduces repeatability. I need to do the same thing repeatedly: add a feature, deploy it to test, deploy it to train, deploy it to production, test it again—doing the same thing repeatedly for any feature addition. It reduces this repetition and also decreases the need for human resources. Human errors are almost zero because everything is automated. In this sense, IaC has helped us a lot. It has removed a lot of the tension that would otherwise stay in our minds during operations."	Automation	
" You could say that IaC helps us with faster deployment. Through IaC, we're able to speed up the deployment process."	Rapid Deployment	
" I think using IaC speeds up infrastructure provisioning, making it seamless. This could be a benefit, as server runtime may be reduced."	Rapid Deployment	

Quote	Initial Code	Code
" In terms of security, IaC is a very good thing. The security rules we want to impose can be easily configured in IaC. Something that would have been a hassle to do manually can be easily introduced through IaC from a security improvement perspective. So, from a security improvement standpoint, IaC is a very nice thing to have."	Security	
" The scenario that I was trying to refer to is the case when the code hosted in a GitHub repository has configurations that are not in our cloud infrastructure. In situations like this, we use an alert mechanism, through which we periodically know – in every one or two hours – if the state declared in our GitHub repository aligns to that of our cloud infrastructure. If it does not align, then someone manually logs into the cloud infrastructure to fix the misalignment."	Configuration Drift	State Reconciliation
" If there is [configuration] drift, we try to resolve it. In our team, we have many folks, so we can discuss by looking at the code in the script. This is possible as everyone is aware about the full infrastructure."	Configuration Drift	
" What happens in our case is that without manual approval from the project lead, no code revision is deployed. As a result, the chances of errors are minimal."	Configuration Drift	
" In the case that there are some changes that seem to take place in the local system or showing in the local system, but not reflected in the state, manual intervention could be required, but I am not 100% sure about it."	Configuration Drift	

4.4 Findings in Second Study

4.4.1 Use of AI Tools

Participants had mixed experiences with AI tools for Kubernetes management. P01 mentioned, "*Yes, in most cases, we test some AI tools for debugging Kubernetes in-house. These tools are mostly open-source, and they haven't been fully integrated yet, but they help with Kubernetes internal debugging.*" P02 elaborated, "*Overmind is trying to solve this problem by estimating the blast radius of a change because it knows so much about your infrastructure,*" suggesting optimism for future AI applications. In contrast, P03 stated, "*No, we don't use any AI tools for Kubernetes management.*" and similarly, P04 noted, "*We don't use AI tools for management either.*"

4.4.2 Specific Kubernetes Platforms

Different cloud platforms were utilized by the participants for Kubernetes. P02 highlighted, "*So, currently, we mainly use cloud-specific Kubernetes, one being AWS's EKS and the other being Google's GKE,*" whereas P05 explained, "*We use data centers located in Bangladesh. Initially, we used MetroNet, and now we use Coloasia. Coloasia is located in Gulshan 1, and MetroNet is in Mirpur, I believe. Our entire Kubernetes infrastructure is built up at Coloasia.*" P04 added, "*Whether it's AWS or Azure, it really depends on the client and what they want in production.*"

4.4.3 Kubernetes Components

There was a focus on the diverse components of Kubernetes. P02 mentioned, "*We use everything that Kubernetes offers; I can't think of anything we don't use.*" On the other hand, P03 specified, "*The components we work with the most are deployments, Replica Sets, Daemon Sets, Ingress Controllers, Services, and Persistent Volumes.*"

4.4.4 Environmental Impact

Participants expressed mixed views on the environmental impact of Kubernetes. P01 highlighted that every computational resource has an environmental cost, noting that, "*In Oracle Cloud, there is an environmental index for each deployment, showing the amount of carbon dioxide generated due to that specific deployment.*" In contrast, P04 believed the environmental cost remains constant across different platforms, stating, "*Yes, there is an environmental cost since we are running servers. The impact is always there, and it would exist on any platform, not just with Docker or virtual machines. The environmental cost is the same, there's no significant difference.*" P02 was more optimistic, suggesting that Kubernetes could have a positive environmental impact due to its ability to optimize resource utilization, emphasizing that "*if we implement it efficiently in a Kubernetes cluster, it can definitely have a positive impact on the environment.*" P03 agreed that Kubernetes could reduce the environmental footprint, particularly by adjusting resource usage during off-peak hours, as they explained, "*Yes, there is definitely an environmental impact. For example, at night, most websites don't have much traffic, so the scale of the pods reduces, right? This means less resource utilization, and power consumption goes down. Ultimately, this results in reduced current utilization and lower carbon emissions.*" Finally, P05 believed that Kubernetes minimizes manual intervention, leading to smoother and more efficient operations, which in turn benefits the environment: "*It makes everything run smoothly and efficiently.*"

4.4.5 Future Kubernetes Releases

Suggestions for future Kubernetes releases included improvements in cluster migration and storage management. P01 emphasized, "*Kubernetes needs to do more work in the supply chain, and I believe CNCF is working on that. Apart from that, there is a lot more work to be done with Kubernetes' internal cluster storage, especially regarding persistent volumes. From my experience and the nature of my work, these are a few things I believe Kubernetes should focus on, and they are already working on them. Hopefully, they will extend the supply chain assessment during the deployment in Kubernetes clusters and ensure that the storage connected to the cluster becomes more optimized and resilient. Additionally, if there is a provision for auto-scaling the storage, it would be great.*" P02 suggested, "*There could definitely be a way to make the cluster migration process more seamless...The work will always be complicated, but one change that could help would be to revise the versioning scheme for Kubernetes. Instead of having releases every six months, maybe Kubernetes could introduce a long-term support (LTS) release along with an edge release. The edge release could bring bleeding-edge features every six months, while the LTS version could be supported for two or three years. Having a stable release with long-term support would help make migrations easier.*" P04 added, "*It's like the documentation for corner cases isn't very good. As I mentioned earlier, when corner cases arise, we don't have good documentation to follow. Maybe in a few more years, the solutions will be more widely available, but right now, we have to figure it out ourselves when these cases arise.*"

4.4.6 Kubernetes Culture in Bangladesh

The Kubernetes culture in Bangladesh is evolving, with more organizations adopting it. P01 pointed out, "*Many organizations...deploy monolithic applications within the Kubernetes environment, which ultimately wastes Kubernetes resources.*" P02 highlighted the emerging DevOps culture, mentioning, "*I can mention a company like AppsCode. They are doing some great work related to Kubernetes.*" P04 also noted the increasing interest in Kubernetes, stating, "*Since Kubernetes is booming worldwide, especially for deployment, I think everyone in the country should pay attention to it, especially containerization, as it is the future of deployment.*"

4.4.7 Compliance

Participants shared a range of perspectives on compliance and regulatory requirements related to Kubernetes. P01 discussed specific benchmarks for Kubernetes environments, highlighting the use of the CIS Benchmarks for Oracle Kubernetes Engine and compliance standards for financial and health-related applications. He stated, "*There are specific benchmarks for Kubernetes itself (like the CIS benchmark) and also for the applications we deploy.*" P02 emphasized the importance of disaster recovery and SOC 2 audits, noting that while Kubernetes itself does not have specific compliance requirements, the infrastructure as a whole must meet certain standards. They mentioned, "*We undergo a SOC 2 audit, which is focused on the cluster's disaster recovery (DR) strategy.*" P03 mentioned that compliance regulations vary by client, explaining that for clients in regions like Bangladesh, they often recommend using managed services like AWS or Azure to handle compliance, while also ensuring that local clusters comply with standards like PCI DSS. They noted, "*We have to ensure compliance with standards like PCI DSS...we have to ensure the network is structured and pods are properly managed.*" Finally, P04 emphasized the strict regulations faced by industries like banking, where data storage must adhere to geographic restrictions, stating, "*We cannot store data on Singapore's AWS servers but must use local infrastructure.*" These findings suggest that while Kubernetes itself does

not have universal compliance rules, its use must align with broader regulatory frameworks depending on the industry and client needs.

4.4.8 Managing Configurations

Participants employed tools like Terraform, YAML, and Helm for managing Kubernetes configurations. P01 described, "*We mostly use YAML for our deployments. For defining the Kubernetes cluster, we use Terraform. For deploying applications within the Kubernetes cluster, we use YAML, CRD, and Helm.*" P02 highlighted, "*We manage configurations through Ansible and Terraform...pod-specific or deployment-specific configurations are kept in Helm's values files.*" P04 mentioned, "*We manage configurations at a very basic level, and we don't use any external services. External services like S3 and Prometheus can help us with managing configurations, but since we require manual intervention, we do it ourselves.*"

4.4.9 Other Container Orchestration Tools

Several participants mentioned their experiences with other container orchestration tools alongside Kubernetes. P01 noted, "*Yes, we have worked with all the major container orchestration tools, such as Docker,*" emphasizing their broad experience in this area. P02 added insights about exploring Docker Swarm and OpenShift, stating, "*OpenShift seems like a good alternative to Kubernetes,*" reflecting ongoing interest in alternative solutions.

4.4.10 Challenges Faced Using Kubernetes

Challenges included understanding the architecture, managing persistent volumes, and the steep learning curve. P01 shared, "*The main challenge we faced was not having a comprehensive understanding of each Kubernetes component's architecture.*" P05 emphasized, "*Understanding and learning the configuration-level details of Kubernetes is quite difficult.*" P04 noted, "*In our local development environment, things happen quite often. Since we use local servers, sometimes a hard disk fails or gets damaged, and then we have to get back to work.*"

Table 4.3: Example of Open Coding Process of Study-2

Code	Quote	ParticipantID
Using AI tool	"Yes, in most cases, we test some AI tools for debugging Kubernetes in-house. These tools are mostly open-source, and they haven't been fully integrated yet, but they help with Kubernetes internal debugging."	P1
	"So far, no. Not specifically for Kubernetes, but in the SRE or DevOps domain, I've been following AI applications in general. Personally, I consider AIOps to be a novel concept. We're still in the pipeline stage, and haven't fully explored it yet."	P2
	"No, we don't use any AI tools for Kubernetes management."	P3
	"We don't use AI tools for management either."	P4
	"No, we haven't used any AI tools for that purpose yet, but yes, there is a plan for it."	P5
Kubernetes platform	"Yes, in Oracle, we use something called OKE (Oracle Kubernetes Engine), which is used on Oracle Cloud infrastructure."	P1
	"So, currently, we mainly use cloud-specific Kubernetes, one being AWS's EKS and the other being Google's GKE."	P2
	"Normally, we use AWS, specifically EKS, but we've also used Azure or Google. We generally prefer EKS on AWS; it's our preferred option."	P3
	"Well, whether it's AWS or Azure, it really depends on the client and what they want in production."	P4

Code	Quote	ParticipantID
	"Yes, the first thing to mention is that, since we are a fintech company, we don't use platforms like Amazon, Google Cloud, or Azure. The reason is that, with services like Amazon, if we store our data in their data centers, we are trusting them to guarantee our privacy and manageability. However, if there is ever a breach, there's no turning back. So, we use data centers located in Bangladesh. Our entire Kubernetes infrastructure is built up at Coloasia. Fintech companies generally prefer this approach, keeping data nearby rather than outsourcing it to far-off locations."	P5

Chapter 5

Discussion

In this section, we summarize the key insights of our study, highlight the novel findings that have not been reported in the prior literature, and discuss the implications of our findings. We also describe the limitations of our work.

In regard to novelty, our findings advance the current body of knowledge on IaC by reporting 13 topics that have not been reported in the existing literature. As summarized in Table 5.1, topics such as adoption, usage and transition of tools, learning, and perceptions on sustainability remained largely unexplored in the existing literature, and our study offers new perspectives on these areas. In addition, important topics such as compliance in IaC development are notably missing from the current scholarly work, highlighting important gaps that our research seeks to address. We reported about several other areas that have not been studied before, including logging and code review in IaC development, the use of AI assistants for IaC, configuration drift, and state reconciliation. Shedding light on these understudied topics, our study contributes new perspectives and address gaps in the existing literature on IaC.

Among the topics that have been previously explored, our findings are consistent with the prior studies. For example, prior research has revealed that the professionals utilize IaC for package management and user account management [94]. Similarly, prior studies have advocated for pro-active detection of bugs in IaC scripts [99, 107, 100, 112, 103, 24], and we found similar activities conducted by our participants. In prior work, researchers have also documented the usage of version control by the software development professionals for managing IaC scripts [54]. Our study further augments this finding by reporting that professionals who use IaC almost exclusively use private version control repositories.

Table 5.1: The topics reported in our study. The highlighted cells in green represent the topics that have not been addressed in prior research.

Category	**Topics**	**Existing IaC Literature**
Economics	Expense, revenue, and local conditions	Not Reported
	Usage and Transition of Tools	Not Reported
	Economics of adoption	Not Reported
	Economics of learning	Not Reported
	IaC and Sustainability	Not Reported
Techniques	Infrastructure Management	Reported ([93, 95])
	Package management	Reported ([94])
	User account management	Reported ([94])
	Compliance in IaC	Not Reported
	Logging in IaC	Not Reported
	Code review for QA in IaC	Not Reported
	Debugging for QA in IaC	Not Reported
	Testing for QA in IaC	Reported ([68, 45, 119])
	Version control in IaC	Reported ([54])
	Usage of AI assistants for IaC	Not Reported
State Reconciliatio	Configuration drift	Not Reported
	Usage of development activities for state reconciliation	Not Reported
	Trust in IaC tools for state reconciliation	Not Reported

5.1 Key Insights in Kubernetes Practice

1. ***Insights on AI Tools in Kubernetes***:
   The varied use of AI tools suggests differing levels of adoption and integration within organizations. While some participants see potential benefits in AI tools for debugging and efficiency, others remain skeptical or have yet to implement such technologies.

2. ***Experience and Expertise***:
   The diverse range of experience among participants highlights the different stages of Kubernetes adoption within organizations. More experienced users have a deeper understanding and longer history with Kubernetes, whereas newer users may still be in the learning phase.

3. ***Platform Preferences and Considerations***:
   The preference for specific cloud platforms underscores the importance of evaluating the pros and cons of each platform based on organizational needs. The emphasis on privacy by the participants who use local data centers, contrasts with the broader use of AWS and GCP by other participants.

4. ***Evolving Job Roles and Responsibilities***:
   The varied job roles and responsibilities suggest that Kubernetes management is a multifaceted discipline requiring diverse skill sets. Organizations may benefit from fostering cross-functional teams that combine expertise in DevOps, software engineering, and system administration to manage Kubernetes effectively.

5. ***Comprehensive Understanding and Documentation***:
   The challenges related to understanding Kubernetes architecture and managing configurations suggest that comprehensive documentation and hands-on experience are crucial. Participants noted the importance of resources such as Kubernetes playgrounds and detailed documentation to facilitate learning and troubleshooting.

6. ***Environmental Impact and Sustainability***:
   The mixed views on the environmental impact of Kubernetes indicate a need for further research and optimization in this area. While some participants see potential benefits in reducing resource waste and improving efficiency, others remain uncertain.

7. ***Future Developments and Community Support***:
   Participants' suggestions for future Kubernetes releases highlight areas for improvement, such as storage management, cluster migration, and long-term support. The growing Kubernetes community in Bangladesh provides a promising foundation for addressing these challenges through collaborative efforts and shared knowledge.

In conclusion, this qualitative analysis provides valuable insights into the current state and future potential of Kubernetes adoption, challenges, and best practices. By addressing the identified issues and leveraging community support, organizations can enhance their Kubernetes implementations and drive further innovation in this space.

5.2 Limitations

We discuss the limitations of our paper by following the guidelines related to interview studies [111]:

- ***Construct Validity***: A potential threat to construct validity is the reliance on participants' self-reported data, which may not fully capture the IaC usage context. To mitigate this, our semi-structured interviews were carefully designed with clear, specific questions that aligned with our research objectives. However, the translation process from Bengali to English might have introduced subtle biases or nuances that could affect the interpretation of responses. Our usage of interview studies is an example of using the respondent strategy according to design science, which increases the likelihood of generalizability but decreases the likelihood of realism [123]. We mitigate this limitation by selecting the practitioners from industry who are using IaC.

- ***Conclusion Validity***: The qualitative nature of the study, relying on thematic analysis and coding, introduces subjectivity into the interpretation of data. While we follow established guidelines for qualitative analysis, such as using multiple rounds of discussion to identify themes related to tooling and state management, the possibility of researcher bias influencing the conclusions remains. This was mitigated by involving multiple researchers in the coding process.

- ***Internal Validity***: In our study, the use of snowball sampling presents a potential threat to internal validity, as it may have led to a sample biased toward certain professional networks or shared experiences, particularly in tooling preferences or economic strategies. This bias could influence the findings, especially if participants recommended colleagues with similar views. Additionally, the varying levels of experience among participants may have affected their responses, leading to inconsistencies in how they manage state reconciliation or select tools, which impacts the internal consistency of our data. Another critical aspect requiring attention is the validity associated with translating responses from Bengali to English. Since our interviews were conducted in Bengali, translating them into English for analysis introduces challenges that could affect the accuracy and reliability of the data. Cultural nuances and contextual meanings in Bengali may not have direct equivalents in English, leading to misinterpretations or loss of meaning. The subjective nature of translation also introduces variability in how responses are understood, potentially affecting the reliability of our findings, especially regarding technical terms or local practices. To address these limitations, we recruited a diverse range of participants in terms of job roles and experience levels, though this threat cannot be entirely eliminated. Future research should consider employing multilingual methodologies or involving native speakers in both data collection and analysis phases to further mitigate these concerns. By acknowledging these translation validity issues within our internal validity discussion, we aim to provide a comprehensive understanding of the limitations inherent in our study while highlighting our efforts to address them effectively.

- ***External Validity***: The sample size of 19 participants in the first study and 5 participants in the pilot interview of the second study limits the generalizability of our findings. Given that our study focused solely on practitioners from Bangladesh, the findings may not be directly applicable to other countries or regions with different cultural, economic, or technological environments. Additionally, the context of IaC in Bangladesh and the overall industry experience of practitioners may influence the reporting of tooling choices and activities related to state reconciliation.

5.3 Learning from the Peers: Documenting the Usage of Techniques

As can be seen throughout Section 4, the reported techniques and practices have different degrees of adoption among our participants. For example, Figure 4.5 demonstrates that while version control is adopted by all the participants, few mentioned logging as a technique used for IaC development. Our analysis also suggests that most of these techniques are applied on an ad hoc basis and there exists no consistent set of practices adopted across different software firms and organizations that work with the IaC tools and technology.

Based on our findings, we recommend documenting and promoting an established set of software engineering practices for professionals who use IaC. Since prior HCI/CSCW literature has shown software developers to effectively trust and learn from their peers [73, 52, 143], and our data reveals that these professionals who use IaC are actively engaged in IaC-related community forums, we suggest that a community-driven effort may pave the way for effectively learning from the peers. Additionally, researchers in collaboration with practitioners can systematically collect data from organizations globally through large-scale surveys in order to document and quantify the usage of activities for IaC. The obtained survey data would be helpful for:

- The organizations that are new to IaC and would like to understand the standard techniques for this technology; and
- The organizations that already use IaC and would be interested in comparing their techniques to those of their peers.

In software engineering, quantification of software development techniques are well-established and have aided practitioners. For example, researchers have quantified the usage of techniques in industry for agile methodologies [104, 71, 134] and software testing [30, 113, 46]. Furthermore, practitioner-led efforts such as the 'Building Security In Maturity Model (BSIMM)' have also aided the professionals in understanding the usage of techniques for software security [65]. Our conjecture is that similar initiatives for IaC will aid the professionals in the same manner.

In fact, our qualitative data shows that compliance-driven efforts constitute a major portion of the activities conducted by these professionals. Since compliance is a recurring theme across many maturity models[11], we hypothesize that a maturity model could eventually be developed for the burgeoning IaC industry in the future. However, we refrain from making any such attempts as our current sample size is low due to the nature of this study, in which we rely on in-depth one-one-on interviews as opposed to large scale surveys. We hope that future research projects will build on our work to propose a maturity model for IaC as well as solve other related issues to grow the science of IaC.

5.4 The Curious Case of Sensitive Information

Our findings reveal that the practice of storing sensitive information such as access keys (secrets that are used to log into cloud-based infrastructure) prevails among the IaC professionals. This finding is consistent with prior research works that also found practitioners to hard-code sensitive information such as passwords in IaC scripts [105, 99, 100]. Storing sensitive information is considered as a bad software engineering practice even if the repositories are private and not connected to the external networks [105, 99]. In the software

engineering community, sensitive information is commonly referred to as secrets, which can "*lead to the exposure of resources or functionality to unintended actors, possibly providing attackers with sensitive information or even execute arbitrary code.*" [67]. Based on these findings, we recommend practitioners use secret management tools that allow practitioners to store and retrieve secrets programmatically with libraries. There exist both language-specific (Hiera for Puppet [90], for example) and language-agnostic (Hashicorp Vault [38], for example) secret management tools that could be used in this regard.

5.5 The Nuances of AI Assistants for IaC

Recent research has revealed AI assistants to be beneficial for a wide range of software engineering tasks, including bug finding [39], bug fixing [42] and code generation [28]. Yet, our findings demonstrate participants' lack of confidence in using AI assistants for IaC. While 12 out of 19 participants mentioned the use of AI assistants, we found them to be mostly skeptic about the current capabilities of the AI assistants with respect to code generation for IaC. Prior HCI worked has provided a variety of suggestions and recommendations that can be used in this regard to improve these AI assistants. First, practitioners working with infrastructure administration expect situational awareness from their tools in order to troubleshoot infrastructure-related failures [35]. To become more useful, the AI assistants need to learn from the development context as these professionals interact with these tools to generate solutions for infrastructure-related tasks. Second, similar to a recent study [80], we found our participants to be skeptic about the quality of the AI-generated code. To gain more trust from the professionals, in addition to generating solutions, these tools could also offer additional features that include but are not limited to, reporting test cases that pass for the generated code, seeking feedback from the user, and providing multiple solutions for a given task. Third, in the case of interaction agents such as AI assistants, the time required to generate a solution plays a pivotal role [33, 41, 44]. Prior research has reported that when solving infrastructure-related tasks, the fastest professionals can find and access their most helpful source of information is less than two minutes [27]. While this may vary depending on the task, this finding demonstrates the importance of generating solutions in a timely manner. We acknowledge that the development context will be different from one practitioner to another, and thus support epistemological pluralism [133] that advocates for technologies tailored for individual needs.

5.6 Implications Related to Green Cloud

As can be seen in Section 4.1.4, our participants are not much concerned about sustainability and green cloud. Unlike reliability aspects, such as defects [103, 24] and security vulnerabilities [99, 107], there is a lack of guidelines on how the software professionals should develop IaC scripts so that unwanted energy consumption is reduced. One approach to mitigate this gap is to review the existing resources related to energy consumption of cloud computing, such as the 'Sustainability Pillar - AWS Well-Architected Framework' [12], and then quantitatively evaluate if these guidelines are applicable for the IaC scripts.

According to Manotas et al. [64], software professionals who use general purpose programming languages consider the design aspects of a software to determine how energy-greedy that particular software is. As such, we advocate for identification of design aspects such as energy anti-patterns for IaC. These energy anti-patterns could be coding patterns within the IaC script that practitioners inadvertently use or architectural patterns on how multiple cloud-based infrastructure is provisioned with IaC. Manotas et al. [64] also reported that professionals prefer static analysis and profiling tools that can aid in identifying which

portions of the code consume unwanted energy. We recommend designing, developing, and evaluating such tools that are focused on energy consumption for IaC. Furthermore, anecdotally – from blog posts – we observe practitioners to describe how certain coding patterns such as an inefficient `loop` in Ansible could incur an additional execution time by a factor of 87 [8]. Researchers can synthesize these practitioner-reported experiences and validate their synthesis with empirical data.

5.7 With Trust Comes Responsibility—The Importance of Reliable State Reconciliation

As reported in Section 4.3.2, we observed that some of our participants trust the IaC tools to perform state reconciliation adequately. However, if the state reconciliation approach is not reliable, it can lead to serious consequences, including exposure of sensitive information [31]. Empirical research has showed that defects related to state reconciliation are frequent, e.g., 5,110 defects have been observed for Ansible that are related to state reconciliation [39]. Our findings, therefore, call for a more rigorous assessment of the IaC tools to test their efficacy in conducting reliable state reconciliation.

Chapter 6

Conclusion and Future Work

Despite the popularity of IaC among different organizations for managing their computing infrastructure, the existing research on IaC has mainly focused on quality and maintainability issues, and there is a lack of understanding of the human aspects in regard to this burgeoning technology. We addressed this gap in the HCI literature by interviewing 24 practitioners who use IaC in a professional setting. Our findings advance the science of IaC as we identify multiple attributes of IaC usage and adoption that are related to economics, tool usage and transition, compliance, trust, and state reconciliation, that have not been reported in the existing literature. We offer a set of guidelines to the HCI researchers for developing better tools for professionals who use IaC. We also connect our findings to the broader issues in HCI that are related to collaboration, sustainability, artificial intelligence, and security, among others. Taken together, our work makes a significant contribution to the IaC literature by providing an in-depth understanding of the techniques, perceptions, and challenges of the professionals in this area.

Building on the findings of this study, future work could explore the deeper integration of Kubernetes with Infrastructure as Code (IaC) practices, particularly focusing on the human aspects of Kubernetes adoption and its impact on workflow automation, scalability, and security. Additionally, further research could involve a broader, more diverse set of participants from different geographical and industrial contexts to increase the generalizability of the findings. Future studies could also expand on the exploration of tooling choices, automation frameworks, and best practices in Kubernetes environments, especially as IaC continues to evolve. Finally, a comparative study between Kubernetes-centric IaC practices and other orchestration tools could provide valuable insights into the comparative effectiveness and challenges of various solutions within the DevOps ecosystem.

References

[1] IaC-Ethno-Study-Interview-Questionnaire — docs.google.com. https://docs.google.com/document/d/1hQWkYuqbGCMLuwJFlEtZhkERKPfN-KREJKT4KRF_6Rs/edit?usp=sharing, 2024. [Accessed 23-08-2024].

[2] IaC-Ethno2024 — docs.google.com. https://docs.google.com/spreadsheets/d/1DDqgMgiHia4fjQ_qDX0HCslFF94EtHEJZaZyuzgecWk/edit?usp=sharing, 2024. [Accessed 23-08-2024].

[3] K8S-Ethno2025 — docs.google.com. https://docs.google.com/spreadsheets/d/1lppL6EjAPuJjyTH2Q70h3NtQ3zOTbIWka9ImTqorUCo/edit?usp=sharing, 2024. [Accessed 23-08-2024].

[4] Nitin Agrawal, Reuben Binns, Max Van Kleek, Kim Laine, and Nigel Shadbolt. Exploring design and governance challenges in the development of privacy-preserving computation. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Amazon. Aws ec2. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html, 2024. [Online; accessed 24-Aug-2024].

[6] Paul Ammann and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2016.

[7] ansible. Adb uses red hat ansible automation platform to boost infrastructure management. https://www.redhat.com/en/resources/asian-development-bank-case-study, 2022. [Online; accessed 25-August-2024].

[8] Ansible. Github page for ansible. https://github.com/ansible/ansible/issues/76380, 2024. [Online; accessed 24-Aug-2024].

[9] Owura Asare, Meiyappan Nagappan, and N. Asokan. A user-centered security evaluation of copilot. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery.

[10] Hala Assal and Sonia Chiasson. 'think secure from the beginning': A survey with software developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[11] SANS Security Awareness. Maturity model. https://www.sans.org/security-awareness-training/resources/maturity-model/, 2024. [Online; accessed 04-Sep-2024].

[12] AWS. Sustainability pillar - aws well-architected framework. https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/sustainability-pillar.html, 2024. [Online; accessed 07-Sep-2024].

[13] David Balla, Csaba Simon, and Markosz Maliosz. Adaptive scaling of kubernetes pods. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, 2020.

[14] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[15] B. Boehm. Improving software productivity. *Computer*, 20(09):43–57, sep 1987.

[16] Barry W Boehm and Kevin J Sullivan. Software economics: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 319–343, 2000.

[17] Carmen Carrión. Kubernetes scheduling: Taxonomy, ongoing issues and challenges. *ACM Comput. Surv.*, 55(7), December 2022.

[18] John Chen, Xi Lu, Yuzhou Du, Michael Rejtig, Ruth Bagley, Mike Horn, and Uri Wilensky. Learning agent-based modeling with llm companions: Experiences of novices and experts using chatgpt & netlogo chat. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[19] Qingrong Chen, Teng Wang, Owolabi Legunsen, Shanshan Li, and Tianyin Xu. Understanding and discovering software configuration dependencies in cloud and datacenter systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 362–374, 2020.

[20] John P. Chin, Virginia A. Diehl, and Kent L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, page 213–218, New York, NY, USA, 1988. Association for Computing Machinery.

[21] Jurgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. The making of cloud applications: an empirical study on software development for the cloud. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 393–403, New York, NY, USA, 2015. Association for Computing Machinery.

[22] Daniela S. Cruzes and Tore Dyba. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, 2011.

[23] Stefano Dalla Palma, Dario Di Nucci, Fabio Palomba, and Damian Andrew Tamburri. Toward a catalog of software quality metrics for infrastructure code. *Journal of Systems and Software*, 170:110726, 2020.

[24] Stefano Dalla Palma, Dario Di Nucci, Fabio Palomba, and Damian Andrew Tamburri. Within-project defect prediction of infrastructure-as-code using product and process metrics. *IEEE Transactions on Software Engineering*, 48(6):2086–2104, 2022.

[25] Fred D Davis, RP Bagozzi, and PR Warshaw. Technology acceptance model. *J Manag Sci*, 35(8):982–1003, 1989.

[26] Breno B Nicolau de França, Helvio Jeronimo, and Guilherme Horta Travassos. Characterizing devops by hearing multiple voices. In *Proceedings of the XXX Brazilian Symposium on Software Engineering*, pages 53–62, 2016.

[27] Cleidson R. B. De Souza, Claudio S. Pinhanez, and Victor Cavalcante. Knowledge and information and needs of system administrators in it service factories. In *Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, IHC+CLIHC '11, page 81–90, Porto Alegre, BRA, 2011. Brazilian Computer Society.

[28] X. Du, M. Liu, K. Wang, H. Wang, J. Liu, Y. Chen, J. Feng, C. Sha, X. Peng, and Y. Lou. Evaluating large language models in class-level code generation. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, pages 982–994, Los Alamitos, CA, USA, apr 2024. IEEE Computer Society.

[29] F. M. A. Erich, C. Amrit, and M. Daneva. A qualitative study of devops usage in practice. *Journal of Software: Evolution and Process*, 29(6):e1885, 2017. e1885 smr.1885.

[30] Vahid Garousi and Junji Zhi. A survey of software testing practices in canada. *Journal of Systems and Software*, 86(5):1354–1376, 2013.

[31] GitHub Advisory Database. Ansible insertion of sensitive information into log file vulnerability. https://github.com/advisories/GHSA-588w-w6mv-3cw5, 2022. [Online; accessed 31-March-2024].

[32] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 649–658, 2009.

[33] Ken Gu, Madeleine Grunde-McLaughlin, Andrew McNutt, Jeffrey Heer, and Tim Althoff. How do data analysts respond to ai assistance? a wizard-of-oz study. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[34] M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 580–589, 2019.

[35] Eben Haber. Sensemaking sysadmins: Lessons from the field. In *Sensemaking Workshop at ACM CHI*, 2005.

[36] Brent Hailpern and Padmanabhan Santhanam. Software debugging, testing, and verification. *IBM Systems Journal*, 41(1):4–12, 2002.

[37] Oliver Hanappi, Waldemar Hummer, and Schahram Dustdar. Asserting reliable convergence for configuration management scripts. *SIGPLAN Not.*, 51(10):328–343, October 2016.

[38] Hashicorp. Vault by hashicorp - manage secrets and protect sensitive data. https://www.vaultproject.io/, 2021. [Online; accessed 04-Nov-2021].

[39] Md Mahadi Hassan, John Salvador, Shubhra Kanti Karmaker Santu, and Akond Rahman. State reconciliation defects in infrastructure as code. *Proc. ACM Softw. Eng.*, 1(FSE), jul 2024.

[40] Andreas Hortlund. Security smells in open-source infrastructure as code scripts : A replication study, 2021.

[41] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page 159–166, New York, NY, USA, 1999. Association for Computing Machinery.

[42] Soneya Binta Hossain, Nan Jiang, Qiang Zhou, Xiaopeng Li, Wen-Hao Chiang, Yingjun Lyu, Hoan Nguyen, and Omer Tripp. A deep dive into large language models for automated bug localization and repair. *Proc. ACM Softw. Eng.*, 1(FSE), jul 2024.

[43] Hanyang Hu, Yani Bu, Kristen Wong, Gaurav Sood, Karen Smiley, and Akond Rahman. Characterizing static analysis alerts for terraform manifests: An experience report. In *2023 IEEE Secure Development Conference (SecDev)*, pages 7–13, 2023.

[44] Scott Hudson, James Fogarty, Christopher Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny Lee, and Jie Yang. Predicting human interruptibility with sensors: a wizard of oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 257–264, New York, NY, USA, 2003. Association for Computing Machinery.

[45] Waldemar Hummer, Florian Rosenberg, Fábio Oliveira, and Tamar Eilam. Testing idempotence for infrastructure as code. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 368–388. Springer, 2013.

[46] T. Hynninen, J. Kasurinen, A. Knutas, and O. Taipale. Software testing: Survey of the industry practices. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1449–1454, 2018.

[47] IEEE SA. Ieee24765-2017 - iso/iec/ieee international standard - systems and software engineering–vocabulary. https://standards.ieee.org/standard/24765-2017.html, 2017. [Online; accessed 21-Oct-2020].

[48] Fortune Business Insights. Infrastructure as code market size, share. https://www.fortunebusinessinsights.com/infrastructure-as-code-market-108777, 2024. [Online; accessed 26-Aug-2024].

[49] ISO. Iso/iec/ieee 29119-1:2022 software and systems engineering — software testing. https://www.iso.org/standard/81291.html, 2024. [Online; accessed 24-Aug-2024].

[50] ISO ISO. Iec/ieee international standard-systems and software engineering–system life cycle processes. *ISO/IEC/IEEE 15288 First edition 2015–05–15, Technical report*, 2015.

[51] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. What is devops? a systematic mapping study on definitions and practices. In *Proceedings of the scientific workshop proceedings of XP2016*, pages 1–11, 2016.

[52] Adam D G Jenkins, Linsen Liu, Maria K Wolters, and Kami Vaniea. Not as easy as just update: Survey of system administrators and patching behaviours. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[53] Amitkumar V Jha, Riya Teri, Subra Verma, Susmita Tarafder, Wriddhi Bhowmik, Sunil Kumar Mishra, Bhargav Appasani, Avireni Srinivasulu, and Nsengiyumva Philibert. From theory to practice: Understanding devops culture and mindset. *Cogent Engineering*, 10(1):2251758, 2023.

[54] Yujuan Jiang and Bram Adams. Co-evolution of infrastructure and source code: An empirical study. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 45–55, Piscataway, NJ, USA, 2015. IEEE Press.

[55] Samia Kabir, David N. Udo-Imeh, Bonan Kou, and Tianyi Zhang. Is stack overflow obsolete? an empirical study of the characteristics of chatgpt answers to stack overflow questions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[56] Shoma Kokuryo, Masanari Kondo, and Osamu Mizuno. An empirical study of utilization of imperative modules in ansible. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 442–449, 2020.

[57] Veronika Krauß, Alexander Boden, Leif Oppermann, and René Reiners. Current practices, challenges, and design implications for collaborative ar/vr application development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[58] Nir Kshetri. Economics of linux adoption in developing countries. *IEEE software*, 21(1):74–81, 2004.

[59] Julien Lepiller, Ruzica Piskac, Martin Schäf, and Mark Santolucito. Analyzing infrastructure as code to prevent intra-update sniping vulnerabilities. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 105–123. Springer, 2021.

[60] Jenny T Liang, Chenyang Yang, and Brad A Myers. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[61] Tom CW Lin. Compliance, technology, and modern finance. *Brook. J. Corp. Fin. & Com. L.*, 11:159, 2016.

[62] Daniel López-Fernández, Jessica Díaz, Javier García, Jorge Pérez, and Ángel González-Prieto. Devops team structures: Characterization and implications. *IEEE Transactions on Software Engineering*, 48(10):3716–3736, 2021.

[63] Alexander Maedche, Christine Legner, Alexander Benlian, Benedikt Berger, Henner Gimpel, Thomas Hess, Oliver Hinz, Stefan Morana, and Matthias Söllner. Ai-based digital assistants: Opportunities, threats, and research perspectives. *Business & Information Systems Engineering*, 61:535–544, 2019.

[64] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th international conference on software engineering*, pages 237–248, 2016.

[65] Gary McGraw. Software security and the building security in maturity model (bsimm). *J. Comput. Sci. Coll.*, 30(3):7–8, jan 2015.

[66] Shane McIntosh, Bram Adams, Thanh H.D. Nguyen, Yasutaka Kamei, and Ahmed E. Hassan. An empirical study of build maintenance effort. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, page 141–150, New York, NY, USA, 2011. Association for Computing Machinery.

[67] MITRE. CWE-Common Weakness Enumeration. https://cwe.mitre.org/index.html, 2021. [Online; accessed 02-May-2021].

[68] Hassan Mohammad Mehedi and Akond Rahman. As code testing: Characterizing test quality in open source ansible development. In *2022 15th IEEE Conference on Software Testing, Verification and Validation (ICST)*, Los Alamitos, CA, USA, apr 2022. IEEE Computer Society.

[69] Ioannis Morfonios. Kubernetes cybersecurity. Master's thesis, University of Piraeus, April 2023.

[70] Kief Morris. *Infrastructure as code*. O'Reilly Media, 2020.

[71] Brendan Murphy, Christian Bird, Thomas Zimmermann, Laurie Williams, Nachiappan Nagappan, and Andrew Begel. Have agile techniques been the silver bullet for software development at microsoft? In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 75–84, 2013.

[72] Emerson Murphy-Hill and Gail C. Murphy. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, page 405–414, New York, NY, USA, 2011. Association for Computing Machinery.

[73] Emerson Murphy-Hill and Gail C. Murphy. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, page 405–414, New York, NY, USA, 2011. Association for Computing Machinery.

[74] Emerson Murphy-Hill, Edward K Smith, Caitlin Sadowski, Ciera Jaspan, Collin Winter, Matthew Jorde, Andrea Knight, Andrew Trenk, and Steve Gross. Do developers discover new tools on the toilet? In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 465–475. IEEE, 2019.

[75] National Institute of Standards and Technology. Infrastructure as code. https://csrc.nist.gov/glossary/term/infrastructure_as_code, 2024. [Online; accessed 09-Aug-2024].

[76] National Institute of Standards and Technology. Management. https://csrc.nist.gov/glossary/term/management, 2024. [Online; accessed 24-Aug-2024].

[77] Sam Newman. *Building microservices*. " O'Reilly Media, Inc.", 2021.

[78] Lorelli S Nowell, Jill M Norris, Deborah E White, and Nancy J Moules. Thematic analysis: Striving to meet the trustworthiness criteria. *International journal of qualitative methods*, 16(1):1609406917733847, 2017.

[79] Afroza Nowshin, Shammi Akhter Shiba, Pratyasha Saha, Farig Yousuf Sadeque, and Taiabul Haque. Understanding the perceptions and practices of the machine learning professionals in bangladesh. In *Companion of the 2024 ACM Conference on Computer Supported Cooperative Work and Social Computing*, New York, NY, USA, 2024. Association for Computing Machinery.

[80] Behrooz Omidvar Tehrani, Ishaani M, and Anmol Anubhai. Evaluating human-ai partnership for llm-based code migration. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA, 2024. Association for Computing Machinery.

[81] Ruben Opdebeeck, Ahmed Zerouali, and Coen De Roover. Smelly variables in ansible infrastructure code: Detection, prevalence, and lifetime. In *Proceedings of the 19th International Conference on Mining Software Repositories*, MSR '22, page 61–72, New York, NY, USA, 2022. Association for Computing Machinery.

[82] Ruben Opdebeeck, Ahmed Zerouali, and Coen De Roover. Control and data flow in security smell detection for infrastructure as code: Is it worth the effort? In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 534–545, 2023.

[83] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, M. Stumm, S. Whitaker, and L. Williams. The top 10 adages in continuous deployment. *IEEE Software*, 34(3):86–95, 2017.

[84] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 754–768, 2022.

[85] Juanjo Pérez-Sánchez, Saima Rafi, Juan Manuel Carrillo de Gea, Joaquín Nicolás Ros, and José Luis Fernández Alemán. A taxonomy on human factors that affect devops adoption. In *World Conference on Information Systems and Technologies*, pages 314–324. Springer, 2024.

[86] Aneta Poniszewska-Marańda and Ewa Czechowska. Kubernetes cluster for automating software production environment. *Sensors*, 21(5), 2021.

[87] Puppet. Nyse and ice: Compliance, devops and efficient growth with puppet enterprise. Technical report, Puppet, April 2018.

[88] Puppet. Practicing world-class it at walmart with puppet. https://www.puppet.com/customers/case-studies/walmart, 2023. [Online; accessed 04-July-2024].

[89] Puppet. Cern stores + analyzes mountains of data to answer science's biggest questions with puppet. https://www.puppet.com/customers/case-studies/cern, 2024. [Online; accessed 04-Sep-2024].

[90] Puppet. Separating data (hiera). https://www.puppet.com/docs/puppet/7/hiera, 2024. [Online; accessed 24-Aug-2024].

[91] Juanjo Pérez-Sánchez, Saima Rafi, Juan Manuel Carrillo de Gea, Joaquín Nicolás Ros, and José Luis Fernández Alemán. A theory on human factors in devops adoption. *Computer Standards and Interfaces*, 92:103907, 2025.

[92] A. Rahman, F. Barsha, and P. Morrison. Shhh!: 12 practices for secret management in infrastructure as code. In *2021 IEEE Secure Development (SecDev)*, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[93] A. Rahman and C. Parnin. Detecting and characterizing propagation of security weaknesses in puppet-based infrastructure management. *IEEE Transactions on Software Engineering*, 49(06):3536–3553, June 2023.

[94] A. Rahman and L. Williams. Characterizing defective configuration scripts used for continuous deployment. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 34–45, April 2018.

[95] Akond Rahman, Dibyendu Brinto Bose, Yue Zhang, and Rahul Pandita. An empirical study of task infections in ansible scripts. *Empirical Software Engineering*, 29(1):34, 2024.

[96] Akond Rahman, Effat Farhana, Chris Parnin, and Laurie Williams. Gang of eight: A defect taxonomy for infrastructure as code scripts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 752–764, New York, NY, USA, 2020. Association for Computing Machinery. pre-print: https://akondrahman.github.io/papers/icse20_acid.pdf.

[97] Akond Rahman, Effat Farhana, and Laurie Williams. The 'as code' activities: development anti-patterns for infrastructure as code. *Empirical Software Engineering*, 25(5):3430–3467, 2020.

[98] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. A systematic mapping study of infrastructure as code research. *Information and Software Technology*, 2018.

[99] Akond Rahman, Chris Parnin, and Laurie Williams. The seven sins: Security smells in infrastructure as code scripts. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 164–175. IEEE, 2019.

[100] Akond Rahman, Md Rayhanur Rahman, Chris Parnin, and Laurie Williams. Security smells in ansible and chef scripts: A replication study. *ACM Trans. Softw. Eng. Methodol.*, 30(1), January 2021.

[101] Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. Security misconfigurations in open source kubernetes manifests: An empirical study. *ACM Trans. Softw. Eng. Methodol.*, 32(4), May 2023.

[102] Akond Rahman, Shazibul Islam Shamim, Hossain Shahriar, and Fan Wu. Can we use authentic learning to educate students about secure infrastructure as code development? In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2*, ITiCSE '22, page 631, New York, NY, USA, 2022. Association for Computing Machinery.

[103] Akond Rahman and Laurie Williams. Source code properties of defective infrastructure as code scripts. *Information and Software Technology*, 112:148 – 163, 2019.

[104] Akond Ashfaque Ur Rahman, Eric Helms, Laurie Williams, and Chris Parnin. Synthesizing continuous deployment practices used in software development. In *Proceedings of the 2015 Agile Conference*, AGILE '15, pages 1–10, Washington, DC, USA, 2015. IEEE Computer Society.

[105] Akond Ashfaque Ur Rahman and Laurie Williams. Different kind of smells: Security smells in infrastructure as code scripts. *IEEE Security Privacy*, pages 2–10, 2021.

[106] Foyzur Rahman and Premkumar Devanbu. How, and why, process metrics are better. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 432–441, 2013.

[107] Sofia Reis, Rui Abreu, Marcelo d'Amorim, and Daniel Fortunato. Leveraging practitioners' feedback to improve a security linter. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery.

[108] Zeineb Rejiba and Javad Chamanara. Custom scheduling in kubernetes: A survey on common problems and solution approaches. *ACM Comput. Surv.*, 55(7), December 2022.

[109] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. Devops adoption benefits and challenges in practice: A case study. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*, pages 590–597. Springer, 2016.

[110] Steven I Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 491–514, 2023.

[111] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[112] Nuno Saavedra and João F. Ferreira. Glitch: Automated polyglot security smell detection in infrastructure as code. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery.

[113] Trina Saha and Rajesh Palit. Practices of software testing techniques and tools in bangladesh software industry. In *2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pages 1–10, 2019.

[114] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.

[115] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. Characterizing devops culture: A systematic literature review. In Ioannis Stamelos, Rory V. O'Connor, Terry Rout, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 3–15, Cham, 2018. Springer International Publishing.

[116] Rian Shambaugh, Aaron Weiss, and Arjun Guha. Rehearsal: A configuration verification tool for puppet. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, page 416–430, New York, NY, USA, 2016. Association for Computing Machinery.

[117] Tushar Sharma, Marios Fragkoulis, and Diomidis Spinellis. Does your configuration code smell? In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 189–200, New York, NY, USA, 2016. ACM.

[118] Janice Jianing Si, Tanusree Sharma, and Kanye Ye Wang. Understanding user-perceived security risks and mitigation strategies in the web3 ecosystem. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[119] Daniel Sokolowski, David Spielmann, and Guido Salvaneschi. Automated infrastructure as code program testing. *IEEE Transactions on Software Engineering*, 50(6):1585–1599, 2024.

[120] Thodoris Sotiropoulos, Dimitris Mitropoulos, and Diomidis Spinellis. Practical fault detection in puppet programs. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 26–37, New York, NY, USA, 2020. Association for Computing Machinery.

[121] Diomidis Spinellis. Don't install software by hand. *IEEE software*, 29(4):86–87, 2012.

[122] Diomidis Spinellis. Package management systems. *IEEE Software*, 29(2):84–86, 2012.

[123] Margaret-Anne Storey, Neil A Ernst, Courtney Williams, and Eirini Kalliamvakou. The who, what, how of software engineering research: a socio-technical framework. *Empirical Software Engineering*, 25:4097–4129, 2020.

[124] Hyewon Suh, Nina Shahriaree, Eric B. Hekler, and Julie A. Kientz. Developing and validating the user burden scale: A tool for assessing user burden in computing systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 3988–3999, New York, NY, USA, 2016. Association for Computing Machinery.

[125] Chengnian Sun, Vu Le, and Zhendong Su. Finding and analyzing compiler warning defects. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 203–213. IEEE, 2016.

[126] Xudong Sun, Runxiang Cheng, Jianyan Chen, Elaine Ang, Owolabi Legunsen, and Tianyin Xu. Testing configuration changes in context to prevent production failures. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 735–751. USENIX Association, November 2020.

[127] Asoke K Talukder and Lawrence Zimmerman. Cloud economics: Principles, costs, and benefits. *Cloud computing: Principles, systems and applications*, pages 343–360, 2010.

[128] Rahul Telang and Sunil Wattal. An empirical analysis of the impact of software vulnerability announcements on firm stock price. *IEEE Transactions on Software Engineering*, 33(8):544–557, 2007.

[129] Sergii Telenyk, Oleksii Sopov, Eduard Zharikov, and Grzegorz Nowakowski. A comparison of kubernetes and kubernetes-compatible platforms. In *2021 11th IEEE*

International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), volume 1, pages 313–317, 2021.

[130] Testim. Testim website. https://www.testim.io/, 2024. [Online; accessed 24-Aug-2024].

[131] Steve Traylen. A puppet infrastructure at cern. https://cds.cern.ch/record/2258306/files/20120711-PuppetCamp-Traylen.pdf, 2012. [Online; accessed 05-Sep-2024].

[132] António Trigo, João Varajão, and Leandro Sousa. Devops adoption: Insights from a large european telco. *Cogent Engineering*, 9(1):2083474, 2022.

[133] Sherry Turkle and Seymour Papert. Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of women in culture and society*, 16(1):128–157, 1990.

[134] Akond Ashfaque Ur Rahman and Laurie Williams. Software security in devops: Synthesizing practitioners' perceptions and practices. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, CSED '16, pages 70–76, New York, NY, USA, 2016. ACM.

[135] Eduard van der Bent, Jurriaan Hage, Joost Visser, and Georgios Gousios. How good is your puppet? an empirically defined and validated quality model for puppet. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 164–174, March 2018.

[136] Markus Voelter. *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. CreateSpace Independent Publishing Platform, USA, 2013.

[137] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. Building and validating a scale for secure software development self-efficacy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–20, New York, NY, USA, 2020. Association for Computing Machinery.

[138] Shuai Wang, Xinyu Lian, Qingyu Li, Darko Marinov, and Tianyin Xu. Ctest4j: A practical configuration testing framework for java. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pages 562–566, 2024.

[139] Aaron Weiss, Arjun Guha, and Yuriy Brun. Tortoise: Interactive system configuration repair. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 625–636, Piscataway, NJ, USA, 2017. IEEE Press.

[140] Santoso Wibowo, Marilyn Wells, et al. Green cloud computing and economics of the cloud: Moving towards sustainable future. *GSTF Journal on Computing (JoC)*, 5(1), 2016.

[141] Philipp Wieder, Joe M Butler, Wolfgang Theilmann, and Ramin Yahyapour. *Service level agreements for cloud computing*. Springer Science & Business Media, 2011.

[142] Jim Witschey, Olga Zielinska, Allaire Welk, Emerson Murphy-Hill, Chris Mayhorn, and Thomas Zimmermann. Quantifying developers' adoption of security tools. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 260–271, 2015.

[143] Shundan Xiao, Jim Witschey, and Emerson Murphy-Hill. Social influences on secure development tool adoption: why security tools spread. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 1095–1106, 2014.

[144] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. Hey, you have given me too many knobs!: Understanding and dealing with over-designed configuration in system software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 307–319, 2015.

[145] Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy. Do not blame users for misconfigurations. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 244–259, 2013.

[146] Yaman Yu, Tanusree Sharma, Sauvik Das, and Yang Wang. " don't put all your eggs in one basket": How cryptocurrency users choose and secure their wallets. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2024.

[147] Yuanliang Zhang, Haochen He, Owolabi Legunsen, Shanshan Li, Wei Dong, and Tianyin Xu. An evolutionary study of configuration design and implementation in cloud systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 188–200, 2021.

[148] Xiaofan Zhao, Tony Clear, and Ramesh Lal. Identifying the primary dimensions of devsecops: A multi-vocal literature review. *Journal of Systems and Software*, 214:112063, 2024.

IaC Related Interview Questionnaire

1. What is your position in your company?
2. Have you used infrastructure as code (IaC) for your profession?
3. Do you also develop other types of source code?
4. How was IaC used for the organizations that you have worked for? Can you elaborate?
5. What tools do you typically use for IaC?
6. What things do you like and dislike about IaC and IaC-related tools? Can you please elaborate?
7. What development activities do you use for IaC-related management and bug finding?
8. Do you have any public /private repositories where you store and maintain IaC scripts? [Can you please share the URL to your public repository(ies)?
9. What AI and non-AI tools do you use in developing IaC scripts?
10. Is your technology stack related with the IaC tool that you use? What are the typical technology stacks that you use with IaC?
11. Do you deal with compliance/standards/regulations when developing IaC scripts?
12. What cloud vendors do you typically use in the context of IaC-based infrastructure management?
13. How hard or easy is it to find and on board professionals with decent IaC skills?
14. What are your perspectives on environmental aspects of IaC-based infrastructure provisioning?
15. Anything else you want to add regarding IaC in the context of Bangladesh and Bangladesh-like developing countries?

Kubernetes Related Interview Questionnaire

1. What is your position in your company?
2. Have you used Kubernetes in your professional work? Which components of Kubernetes, e.g., operators and add-ons, do you work with? What are the use cases of Kubernetes in your organization?
3. How long have you been using Kubernetes?
4. What specific Kubernetes platform, e.g., Amazon EKS, Azure AKE, and GKE does your organization use?
5. How has Kubernetes been implemented in the organizations you have worked for? Can you elaborate?
6. What types of applications do you typically deploy using Kubernetes?
7. Are there any specific tools or extensions you use with Kubernetes?
8. Do you also work with other container orchestration tools or technologies?
9. Do you use any AI tools to enhance your Kubernetes management or operations?
10. What do you like and dislike about Kubernetes? Can you please elaborate?
11. What challenges have you faced while using Kubernetes, and how do you solve these challenges? Are these challenges unique to the use cases for your organization?
12. Do you deal with compliance or regulations when managing Kubernetes clusters?
13. Have you experienced any incidents in your organization related to Kubernetes? Can you please elaborate?
14. How do you manage configurations for your Kubernetes deployments?
15. How hard or easy is it to find onboard professionals with decent Kubernetes skills?
16. What are your perspectives on the environmental impacts of Kubernetes-based infrastructure provisioning?
17. What features would you like to see improved in future Kubernetes releases?
18. Is there anything else you would like to add regarding Kubernetes practices in the context of Bangladesh and other developing countries?