Authorship of Minor Contributors in Kubernetes Configuration Scripts: An Exploratory Study

Akond Rahman
Auburn University
Auburn, Alabama, USA
akond@auburn.edu

Gerry Dozier
Auburn University
Auburn, Alabama, USA
doziegv@auburn.edu

Yue Zhang
Auburn University
Auburn, Alabama, USA
yzz0229@auburn.edu

Abstract

Kubernetes is a popular open source software (OSS) tool to manage containers at scale. Despite being beneficial for rapid deployment, Kubernetes-based software deployments are susceptible to defects that can lead to serious consequences. A systematic analysis of development-related factors that cause defects can aid practitioners on how to mitigate these defects. We conduct an exploratory empirical study where we use causal analysis to quantify the impact of one development factor called minor contributors, which refers to practitioners who author < 5% of the total code. By analyzing 29,028 commits from 157 OSS repositories, we observe (i) 5.6% of the 29,028 commits to be authored by minor contributors; and (ii) authorship of minor contributors to impact defects in configuration scripts. Based on our findings, we recommend researchers to (1) further investigate the characteristics of minor contributors; and (2) identify other development-related factors that may have a causal impact on defects in Kubernetes configuration scripts.

CCS Concepts

• **Software and its engineering → Software defect analysis**; **Empirical software validation**.

Keywords

causal inference, configuration, defect, devops, Kubernetes

ACM Reference Format:

Akond Rahman, Gerry Dozier, and Yue Zhang. 2025. Authorship of Minor Contributors in Kubernetes Configuration Scripts: An Exploratory Study. In *Proceedings of Causal Methods in Software Engineering (Cause 2025).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Kubernetes is a popular open source software (OSS) tool to implement the practice of container orchestration. Container orchestration is the practice of automatically managing multiple containers at scale [16]. The global market of Kubernetes is expected to be 11.78 billion by the year of 2032 [1]. Usage of Kubernetes has yielded benefits for organizations. For example, the U.S. Department of Defense (DoD) decreased their software deployment time from 3~8 months to 1 week [2].


```
metadata:
  labels:
     k8s-app: calico-typha-autoscaler
  annotations:
     scheduler.alpha.kubernetes.io/critical-pod: ''
```

Figure 1: A commit that includes a defect in a Kubernetes configuration script. The commit is authored by a minor contributor.

Despite reported benefits, Kubernetes-based software deployments are susceptible to configuration defects that can lead to example to serious consequences, such as outages. For example, a defect in a Kubernetes configuration script created an outage for Reddit, the popular social media platform [9]. Recent research also reports defects to be prevalent for Kubernetes configuration scripts. For example, Rahman et al. [16] reported 1,051 security defects in 2,039 configuration scripts.

The prevalence of defects in Kubernetes configuration scripts necessitate systematic understanding of the factors that can cause defects in configuration scripts. Anecdotal evidence from the OSS domain can provide us signals on what factors can contribute to defects. Let us consider Figure 1 in this regard, which presents a code snippet from an OSS repository [2]. The code snippet includes a defect as highlighted in red. The defect occurs because of providing an incorrect value for the configuration parameter `scheduler.alpha.kubernetes.io/critical-pod`. Manual exploration reveals that the defective code snippet is included in a commit which is authored by a 'minor contributor', i.e., a contributor who authors < 5% of the total lines of configuration scripts in the repository [17]. A systematic analysis can determine whether or not authorship of minor contributors can impact defects in Kubernetes configuration scripts. Such an analysis, which remains under-explored, can be useful for practitioners to mitigate defects in configuration scripts.

Accordingly, we answer the following research questions:

[1] https://www.skyquestt.com/report/kubernetes-market
[2] https://github.com/aws/amazon-vpc-cni-k8s/

Cause 2025, Trondheim, Norway

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- **RQ1**: *How frequently do minor contributors change Kubernetes configuration scripts?*
- **RQ2**: *How does authorship of minor contributors impact defects in Kubernetes configuration scripts?*

We conduct an empirical study with 29,028 commits mined from 157 OSS repositories that contain 44,401 Kubernetes configuration scripts. We apply a qualitative analysis technique called closed coding [18] to derive defect-related commits. We apply a causal analysis technique called causal inference [13] to quantify the impact of minor contributors' authorship on defect-related commits.

Contribution: Our contribution is an evaluation of causal relationship between minor contributors' authorship and defects in Kubernetes configuration scripts.

2 Background and Related Work

2.1 Background

Kubernetes is an OSS tool to implement the practice of container orchestration. Researchers have also referred to it as a cluster management tool as it is used to manage computing clusters [20, 21]. In order to specify which configurations are needed for their desired deployment, practitioners use Kubernetes configuration scripts. These scripts are developed in YAML syntax and are used to specify the configurations of Kubernetes entities, such as `pods`, `deployment`, and `service` [11]. Listing 1 shows an example configuration script that specifies the configurations of a service called 'sample-service' that will run using TCP through port # 80.

```
kind: Service
metadata:
  name: sample-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
```

Listing 1: Example of a Kubernetes configuration script.

2.2 Related Work

Researchers have shown increasing interest in quality assurance for Kubernetes in recent years. Rahman et al. [16] identified what types of Kubernetes objects are impacted by security defects. Shamim et al. [19] studied security tool performance for Kubernetes configuration scripts. Carmen et al. [1] in their study, created a new taxonomy for Kubernetes scheduling techniques, organizing the techniques into five main domains and highlighting where current scheduling techniques fall short, especially in terms of security and performance. Gu et al. [6], Sun et al. [20, 21], and Xu et al. [22] in separate publications focused on analyzing and detecting defects related to Kubernetes controllers. Xu et al. [22] focused on deriving a taxonomy for defects that occur in Kubernetes operators, which are specialized controllers.

Despite the growing body of literature in the domain of quality assurance for Kubernetes, we observe authorship of configuration scripts to be under-explored. We address this gap in this paper.

3 Methodology

We provide the methodology of our paper as follows.

3.1 Dataset Construction

We describe our methodology to construct our dataset in the following subsections.

3.1.1 Mine OSS Repositories with Kubernetes Configuration Scripts. We use the GHTorrent archive [5] that is hosted on Google Big Query to obtain OSS repositories that contain Kubernetes configuration scripts. We apply the following filtering criteria: (i) repository must be publicly available and contain the 'Kubernetes' label; (ii) at least 10% of the files in the repository are YAML files where each file contains Kubernetes objects; (iii) the repository is not a copy of another repository; and (iv) the repository has at least 10 contributors. Initially, we start with 14,747,836 repositories. After applying our filtering criteria, we find 157 repositories, which we use to answer RQ1 and RQ2. Attributes of these repositories is available in Table 1.

3.1.2 Qualitative Analysis to Determine Defect-related Commits. We apply a qualitative analysis technique called closed coding [18], where each fo the two raters maps each commit—in which a Kubernetes configuration script is modified—to a defect. If a commit maps to a defect, then we refer to that commit as a defect-related commit. In order to determine the mapping, first, we use a keyword-based approach similar to prior research on defect analysis [15]. We separate commits for which any of the following keywords appear: 'bug', 'defect', 'error', 'fault', 'fix', 'flaw', 'incorrect', 'issue', and 'mistake'. Using keyword search, we find 66 commits that contain at least one of the above-mentioned keywords.

Next, we use two raters of the paper who individually inspects the following: (i) problematic code exists in the commit diff; (ii) problematic code leads to an incorrect or undesired consequence that is explicitly expressed by a practitioner in the commit message; (iii) the commit message describes an immediate consequence of the defect; and (iv) the problematic code was repaired. One of the raters is a volunteer and the other is the last author of the paper. The last author and the volunteer respectively, identifies 50 and 61 commits. In the case of 23 disagreements, the first author is the resolver, and his decision is final. In all, we identify that 52 of the 66 commits to be defect-related. Each of these 52 commits contain defect-related code for configuration scripts.

Table 1: Dataset Attributes

Category	Data
Total Repositories	157
Total Commits	417,598
Total Kubernetes-related Commits	29,028
Total Contributors	21,559
Total Kubernetes Scripts	44,401
Total Size (LOC)	51,282,124

3.2 Methodology to Answer RQ1

We answer RQ1, first deriving minor contributors for each repository. Here, we determine a contributor of the repository to be a minor contributor if the contributors authors < 5% of the total lines in all configurations scripts that reside in that repository. After deriving all minor contributors for each of the 157 repositories, we determine whether or not a commit is authored by a minor contributor. Here, we use the output from the first step, where we identify if each commit is authored by a minor contributor. Next, for each commit we compute the following metrics:

- **Age**: The difference between the timestamp of the commit and the first commit in the repository measured in days;
- **Size**: The total number of lines added and deleted in a commit;
- **Files**: Total number of files changed in a commit.

We answer RQ1 by reporting: (i) count of commits authored by minor contributors; (ii) count of defect-related commits authored by minor contributors; (iii) and average, and standard deviation for size, file count, and age for commits authored by minor contributors as well as for commits not authored by minor contributors.

3.3 Methodology to Answer RQ2

We answer RQ2 by quantifying the causal relationships between authorship of minor contributors and defect-related commits. We use the potential outcomes framework, that focuses on quantifying the differences in outcomes to estimate the causal impact of the treatment variables [7]. Unlike causal discovery that focuses on inferring causal relationships amongst treatment variables, we use causal inference to quantify the relationship between pre-defined treatment variables and pre-defined outcome variables [13].

For RQ2, the pre-defined treatment variable is commit authorship, i.e., whether or not a commit is being authored by a minor contributor. The outcome variable is defect-related commits. We use three confounding factors, namely files, commit size, and age as each of these three factors are related with software quality [3, 14, 23]. Inclusion of these metrics as confounding factors can aid in estimating the causal impact of commit authorship adequately.

We implement causal inference using the CausalLib package [10] with inverse probability weighting (IPW). IPW is a model that can be used to obtain average effect estimation. IPW requires a model for estimating the probabilities of treatment assignments given the confounding factors [12]. For applying IPW, we use a logistic regression model [8] as our outcome is binary: 1 if the commit is defect-related, 0 otherwise.

We compute propensity density distribution (PDD) [4] to determine if our data is well-suited for applying IPW-based causal inference. Figure 2 shows PDD for both groups—commits with minor contributors and commits without minor contributors—to overlap. The implication of this finding is that our data with minor contributors is well-suited for using IPW for causal inference [4], as data points with similar characteristics can be compared between two groups: commits authored by minor contributors and commits that are not authored by minor contributors.

4 Results

We provide answers to our research questions as follows:

Answer to RQ1. In all, we identify 11,057 minor contributors in 157 repositories. A distribution of the minor contributors is available in Figure 3. We observe 1,643 out of 29,028 commits to be authored by minor contributors. Out of the 52 defect-related commits, 14 are authored by minor contributors. Table 2 reports statistics for age, size, and files for commits that are authored by minor contributors and for commits not authored by minor contributors respectively, using the 'Commits with Minor Contrib.' and 'Commits with No Minor Contrib.' columns.

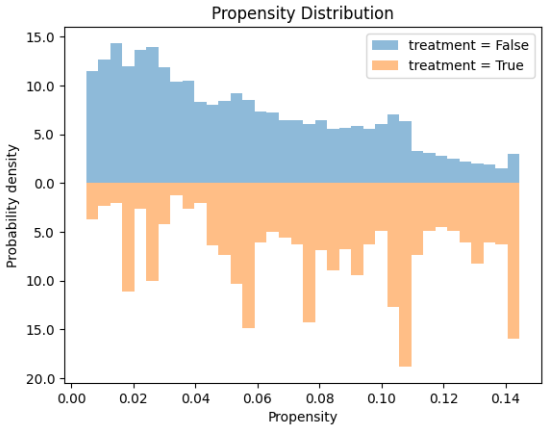

Figure 2: Propensity density distribution for commits authored by minor contributors ('treatment = True') and commits not authored by minor contributors ('treatment = False').

Answer to RQ2. : We answer RQ2 using Table 3. According to Table 3, modification of commits by a minor contributor results in a higher rate of defect likelihood by a factor of 7, i.e., from 0.1% to 0.7%.

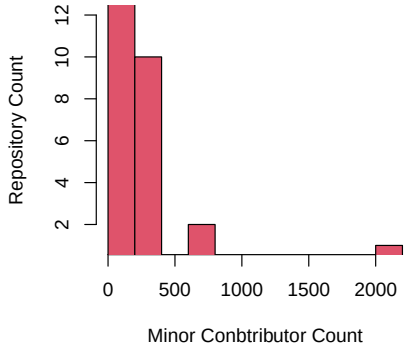

Figure 3: Distribution of minor contributors per repository.

Table 2: Distribution of Age, Size, and Files for Kubernetes-related Commits

Metric	Commits with Minor Contrib. (Avg., Std.Dev.)	Commits with No Minor Contrib. (Avg., Std.Dev.)
Age	463.1, 524.1	882.2, 639.4
Size	34.0, 58165.6	35.0, 100448.1
Files	4.0, 160.4	4.0, 290.9

Table 3: Results from Causal Inference

Minor Contributors	Outcome (%)
False	0.1
True	0.7

5 Discussion

We discuss the implications along with limitations as follows:

Implications for Practitioners. Our analysis shows that minor contributors' authorship has an impact on defects. This finding can be used a heuristic for prioritizing code review efforts. In the case of conducting code reviews, commits that are modified by minor contributors can be prioritized for inspection.

Implications for Researchers. Our answers to RQ2 show that minor contributors' authorship has impact, which necessitates further investigation on the nature of minor contributors for Kubernetes-based configuration management. We also advocate for further empirical research that will apply causal analysis techniques to identify other factors for defect proneness of Kubernetes configuration scripts.

Limitations. The limitations of our paper are:

Conclusion Validity: The defect dataset is constructed by two raters making the process susceptible to rater bias. We mitigate this limitation by assigning the first author for disagreement resolution. In our causal inference analysis, we use three confounding factors. There could be other confounding factors that we have not considered.

External Validity: Our analysis is limited to the repositories that we use from the OSS domain. Our findings may not generalize for proprietary repositories.

Construct Validity: Our analysis applies causal inference on defect-related commits that are mined from OSS repositories. Our analysis is subject to construct validity because of not considering other factors that are unavailable in repositories.

6 Conclusion

As defects in Kubernetes configuration scripts can cause serious consequences, it is important to identify the factors that cause these defects. We characterize one factor namely, authorship of minor contributors in Kubernetes configuration scripts. Our empirical study finds 11,057 minor contributors in 157 repositories. We also observe minor contributors' authorship to have an impact on defect-related commits when three confounding factors are applied: age, file, and size of the commit. Based on our findings, we recommend researchers to further characterize other development factors including minor contributors that can cause defects.

Acknowledgments

We thank the PASER group at Auburn University for their valuable feedback. This research was partially funded by the U.S. National Science Foundation (NSF) Award # 2312321.

References

[1] Carmen Carrión. 2022. Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Comput. Surv.* 55, 7, Article 138 (dec 2022), 37 pages. doi:10.1145/3539606

[2] CNCF. 2025. With Kubernetes, the U.S. Department of Defense Is Enabling DevSecOps on F-16s and Battleships. https://www.cncf.io/case-study/dod/

[3] Filipe Falcão, Caio Barbosa, Baldoino Fonseca, Alessandro Garcia, Márcio Ribeiro, and Rohit Gheyi. 2020. On Relating Technical, Social Factors, and the Introduction of Bugs. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 378–388. doi:10.1109/SANER48275.2020.9054824

[4] Melissa M Garrido, Amy S Kelley, Julia Paris, Katherine Roza, Diane E Meier, R Sean Morrison, and Melissa D Aldridge. 2014. Methods for constructing and assessing propensity scores. *Health services research* 49, 5 (2014), 1701–1720.

[5] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 12–21.

[6] Jiawei Tyler Gu, Xudong Sun, Wentao Zhang, Yuxuan Jiang, Chen Wang, Mandana Vaziri, Owolabi Legunsen, and Tianyin Xu. 2023. Acto: Automatic End-to-End Testing for Operation Correctness of Cloud System Management. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 96–112. doi:10.1145/3600006.3613161

[7] Grace Guo, Ehud Karavani, Alex Endert, and Bum Chul Kwon. 2023. Causalvis: Visualizations for Causal Inference. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 462, 20 pages. doi:10.1145/3544548.3581236

[8] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. Vol. 398. John Wiley & Sons.

[9] Jayme Howard. 2024. You Broke Reddit: The Pi-Day Outage. https://www.reddit.com/r/RedditEng/comments/11xx5o0/you_broke_reddit_the_piday_outage/. [Online; accessed 30-July-2024].

[10] Ehud Karavani. 2024. Causal Inference with Causallib. In *PyData Tel Aviv*.

[11] Kubernetes. 2024. Kubernetes Documentation. https://kubernetes.io/docs/home/. [Online; accessed 14-August-2024].

[12] Mohammad Ali Mansournia and Douglas G Altman. 2016. Inverse probability weighting. *Bmj* 352 (2016).

[13] Judea Pearl. 2009. Causal inference in statistics: An overview. *Statistics Surveys* 3, none (2009), 96 – 146. doi:10.1214/09-SS057

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

[15] Akond Rahman, Effat Farhana, Chris Parnin, and Laurie Williams. 2020. Gang of Eight: A Defect Taxonomy for Infrastructure as Code Scripts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 752–764. doi:10.1145/3377811.3380409

[16] Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. 2023. Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study. *ACM Trans. Softw. Eng. Methodol.* 32, 4, Article 99 (May 2023), 36 pages. doi:10.1145/3579639

[17] Foyzur Rahman and Premkumar Devanbu. 2011. Ownership, experience and defects: a fine-grained study of authorship. In *Proceedings of the 33rd International Conference on Software Engineering* (Waikiki, Honolulu, HI, USA) *(ICSE '11)*. Association for Computing Machinery, New York, NY, USA, 491–500. doi:10.1145/1985793.1985860

[18] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.

[19] Shazibul Islam Shamim, Hanyang Hu, and Akond Rahman. 2025. On Prescription or Off Prescription? An Empirical Study of Community-prescribed Security Configurations for Kubernetes. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering* (Ottawa, Canada) *(ICSE '25)*. Association for Computing Machinery, New York, NY, USA, 13 pages.

[20] Xudong Sun, Wenqing Luo, Jiawei Tyler Gu, Aishwarya Ganesan, Ramnatthan Alagappan, Michael Gasch, Lalith Suresh, and Tianyin Xu. 2022. Automatic reliability testing for cluster management controllers. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 143–159.

[21] Xudong Sun, Wenjie Ma, Jiawei Tyler Gu, Zicheng Ma, Tej Chajed, Jon Howell, Andrea Lattuada, Oded Padon, Lalith Suresh, Adriana Szekeres, et al. 2024. Anvil: Verifying Liveness of Cluster Management Controllers. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. 649–666.

[22] Qinxing Xu, Yu Gao, and Jun Wei. 2024. An Empirical Study on Kubernetes Operator Bugs. In *Proceedings of the 33nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Vienna, Austria) *(ISSTA 2024)*. Association for Computing Machinery, New York, NY, USA, 12 pages.

[23] Thomas Zimmermann, Nachiappan Nagappan, and Laurie Williams. 2010. Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista. In *2010 Third International Conference on Software Testing, Verification and Validation*. 421–428. doi:10.1109/ICST.2010.32