Detecting Configuration Drift in Continuous Deployment Pipelines

Ada Lovelace [1] and Second Author[2]

Abstract: Configuration drift – the gradual divergence of a running system's configuration from its declared state – undermines the reproducibility guarantees that continuous deployment pipelines are meant to provide. We present a lightweight static-analysis approach that flags drift-prone configuration patterns before deployment, and report a case study on an internal pipeline where the approach reduced post-deployment incidents.

Keywords: configuration management, continuous deployment, software reliability

1 Introduction

Modern deployment pipelines promise that a system's running state matches its declared configuration. In practice, manual hotfixes, partial rollouts and stale caches cause *configuration drift* [HF10], a divergence that is rarely detected until it causes an incident.

2 Related Work

Prior work on configuration verification largely targets post-hoc auditing rather than pre-deployment prevention [Oli+17]. Our approach instead analyses the pipeline's declared configuration artefacts directly, before any deployment step runs.

3 Approach

We statically analyse the pipeline's configuration manifests for three drift-prone patterns: unpinned dependency versions, environment-specific overrides with no corresponding declaration, and configuration values set outside the pipeline's own tooling. Table 1 summarises the patterns and their detection rate in our case study.

4 Case Study

We applied the approach described in Section 3 to an internal deployment pipeline over a three-month period. Post-deployment incidents attributable to configuration drift fell from a baseline of roughly six per month to one per month over the study period.

[1] Institute of Analytical Engines, Berlin, Germany, ada@example.edu, https://orcid.org/0000-0002-1825-0097
[2] Department of Computer Science, Example University, Germany, second@example.org

Tab. 1: Drift-prone configuration patterns and detection results.

Pattern	Instances found	True positives
Unpinned dependency	34	31
Undeclared override	12	12
Out-of-band value	7	6

5 Conclusion

Static analysis of configuration manifests can catch a meaningful share of drift-prone patterns before they reach production, at a fraction of the cost of runtime auditing.

References

[HF10] Humble, J.; Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010.

[Oli+17] Oliveira, F. et al.: Understanding and Validating Database System Administration. In: Proceedings of the USENIX Annual Technical Conference. Pp. 213–228, 2017.