Continuous Configuration Testing

Tianyin Xu
University of Illinois at Urbana-Champaign
tyxu@illinois.edu

TL;DR

We will present our recent work on *continuous configuration testing*, a new testing technique that enables configuration changes to be unit-tested in DevOps-based continuous integration/deployment.

Abstract

Configuration management is an integral part of modern DevOps-based cloud system management. Many critical operations are done by updating configurations to dynamically change system behavior in production. Today, large-scale cloud and Internet services evolve rapidly, with hundreds to thousands of configuration changes deployed *daily* [1, 6, 7, 10, 12]. For example, at Facebook, thousands of configuration changes are committed every day, outpacing the frequency of code changes [12]. Other cloud services from Google and Azure also frequently deploy configuration changes [3, 4, 7]. It is not surprising to hear that "*cloud feels more about configuration management than software engineering*" [5].

With the high velocity of changes, faulty configurations inevitably have become major causes of system failures and service outages [2, 13, 14]. For example, faulty configurations are reported as the second largest cause of service disruptions in a main Google production service [2]. At Facebook, 16% of service-level incidents are induced by configuration changes [12]. Many configuration-induced failures led to catastrophic impacts. For instance, in March 2019, a misconfiguration led to Facebook's largest outage in terms of duration (14 hours) [11]; in June 2021, a seemingly-valid configuration change at Fastly triggered an undiscovered software bug and broke the Internet for an hour [8, 9].

We argue that continuous testing is a key missing piece of today's configuration management practice. Despite the "configuration-as-code" movement, there is no widely-used, systematic configuration testing technique and thus configuration changes are not unit-tested—imagining a world where code changes only go through manual review and static analysis, without regression testing.

We will introduce the idea of *configuration testing*, a new testing technique that enables configuration changes to be unit-tested in DevOps-based continuous integration/deployment. The basic idea of configuration testing is connecting system configurations to software tests so that configuration changes can be tested in the context of code affected by the changes. We will introduce a new type of tests, termed *Ctests*, to fill the critical need of configuration testing. Ctests complement static validation (the *de facto* protection [12]), analogous to how testing complements static analysis:

- Ctests can detect failure-inducing configuration changes where the failure root causes are in the code, e.g., valid configuration value changes that trigger dormant bugs.
- Ctests can detect sophisticated misconfigurations (e.g., those that violate undocumented, hidden constraints) by capturing the resulting unexpected system behavior.

We will demonstrate that ctests can be generated by transforming existing software tests that are abundant in mature software projects. The generated ctests selectively inherit test logic and assertions from the original tests. The inherited assertions hold for all correct configuration values. We have successfully generated 7,974 ctests for five widely-used open-source cloud systems (Hadoop Common, HDFS, HBase, ZooKeeper, and Alluxio).

We will also show that the generated ctests are effective and outperform state-of-the-art static configuration validation techniques, based on extensive evaluations using real-world failure cases, synthesized misconfigurations, and deployed configuration files in public Docker images.

Last but not least, we will discuss our ongoing and future work, including: unified regression tests for both source-code and configuration changes, cross-system configuration testing, configuration compatibility testing, and others.

References

[1] Adkins, H., Beyer, B., Blankinship, P., Oprea, A., Lewandowski, P., and Stubblefield, A. *Building Secure and Reliable Systems: Best Practices for Designing, Implementing and Maintaining Systems*. O'Reilly Media Inc., March 2020. https://static.googleusercontent.com/media/sre.google/en//static/pdf/building_secure_and_reliable_systems.pdf.

[2] Barroso, L. A., Hölzle, U., and Ranganathan, P. *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, 3 ed. Morgan and Claypool Publishers, October 2018. https://doi.org/10.2200/S00874ED3V01Y201809CAC046.

[3] Beyer, B., Murphy, N. R., Rensin, D. K., Kawahara, K., and Thorne, S. *Site Reliability Workbook: Practical Ways to Implement SRE*. O'Reilly Media Inc., August 2018. https://sre.google/workbook/table-of-contents/.

[4] Bhagwan, R., Kumar, R., Maddila, C. S., and Philip, A. A. Orca: Differential Bug Localization in Large-Scale Services. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation (OSDI'18)* (October 2018). https://www.usenix.org/conference/osdi18/presentation/bhagwan.

[5] Dogan, J. Cloud sometimes feels more about configuration management than software engineering, December 2020. https://twitter.com/rakyll/status/1338598475185852416.

[6] Maurer, B. Fail at Scale: Reliability in the Face of Rapid Change. *Communications of the ACM 58*, 11 (November 2015), 44–49. https://doi.org/10.1145/2838344.2839461.

[7] Mehta, S., Bhagwan, R., Kumar, R., Ashok, B., Bansal, C., Maddila, C., Bird, C., Asthana, S., and Kumar, A. Rex: Preventing Bugs and Misconfiguration in Large Services using Correlated Change Analysis. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)* (February 2020). https://www.usenix.org/conference/nsdi20/presentation/mehta.

[8] Rockwell, N. Summary of June 8 outage. https://www.fastly.com/blog/summary-of-june-8-outage, June 2020.

[9] Salter, J. Fastly broke the Internet for an hour this morning. https://arstechnica.com/gadgets/2021/06/fast-ly-broke-the-internet-for-an-hour-this-morning/, June 2020.

[10] Sherman, A., Lisiecki, P., Berkheimer, A., and Wein, J. ACMS: Akamai Configuration Management System. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)* (May 2005). https://www.usenix.org/legacy/publications/library/proceedings/nsdi05/tech/full_papers/sherman/sherman.pdf.

[11] Shieber, J. Facebook blames a server configuration change for yesterday's outage. https://techcrunch.com/2019/03/14/facebook-blames-a-misconfigured-server-for-yesterdays-outage/, 2019.

[12] Tang, C., Kooburat, T., Venkatachalam, P., Chander, A., Wen, Z., Narayanan, A., Dowell, P., and Karl, R. Holistic Configuration Management at Facebook. In *Proceedings of the 25th ACM Symposium on Operating System Principles (SOSP'15)* (October 2015). https://doi.org/10.1145/2815400.2815401.

[13] Xu, T., Jin, X., Huang, P., Zhou, Y., Lu, S., Jin, L., and Pasupathy, S. Early Detection of Configuration Errors to Reduce Failure Damage. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)* (November 2016). https://www.usenix.org/conference/osdi16/technical-sessions/presentation/xu.

[14] Xu, T., Zhang, J., Huang, P., Zheng, J., Sheng, T., Yuan, D., Zhou, Y., and Pasupathy, S. Do Not Blame Users for Misconfigurations. In *Proceedings of the 24th Symposium on Operating System Principles (SOSP'13)* (November 2013). https://doi.org/10.1145/2517349.2522727.