Poster: Defect Prediction Metrics for Infrastructure as Code Scripts in DevOps

Akond Rahman, Jonathan Stallings, and Laurie Williams
North Carolina State University
aarahan@ncsu.edu, jwstalli@ncsu.edu, andlawilli3@ncsu.edu

ABSTRACT
Use of infrastructure as code (IaC) scripts helps software teams manage their configuration and infrastructure automatically. Information technology (IT) organizations use IaC scripts to create and manage automated deployment pipelines to deliver services rapidly. IaC scripts can be defective, resulting in dire consequences, such as creating wide-scale service outages for end-users. Prediction of defective IaC scripts can help teams to mitigate defects in these scripts by prioritizing their inspection efforts. The goal of this paper is to help software practitioners in prioritizing their inspection efforts for infrastructure as code (IaC) scripts by proposing defect prediction model-related metrics. IaC scripts use domain specific languages (DSL) that are fundamentally different from object-oriented programming (OOP) languages. Hence, the OOP-based metrics that researchers used in defect prediction might not be applicable for IaC scripts. We apply Constructivist Grounded Theory (CGT) on defect-related commits mined from version control systems to identify metrics suitable for IaC scripts. By applying CGT, we identify 18 metrics. Of these metrics, 13 are related to IaC, for example, count of string occurrences in a script. Four of the identified metrics are related to churn, and one metric is lines of code.

CCS CONCEPTS
• Software and its engineering → Software defect analysis;

KEYWORDS
Continuous Deployment, DevOps, Infrastructure as Code, Metrics

1 INTRODUCTION
Information technology (IT) organizations are increasingly adopting DevOps practices [1]. DevOps organizations i.e. IT organizations that adopt DevOps, have strong collaboration between software development and operations teams to deliver software rapidly [2] [7]. DevOps organizations use technologies to automate repetitive work and increase transparency between software development and operations teams [5]. One technology that these organizations consider essential to implement DevOps is the use of infrastructure as code (IaC) scripts [7] [11]. DevOps organizations use IaC scripts such as Puppet 1 scripts, to automatically manage their configurations and operations infrastructure [6] [7].

IaC scripts help to create and manage automated deployment pipelines, and deploy software rapidly [7]. But similar to software source code, IaC scripts churn frequently [9] [10], and can contain defects [10]. Defects in IaC scripts can have dire consequences: for example, Github experienced a DNS outage caused by a defect in an IaC script [3]. Defect prediction of software modules helps software teams to prioritize inspection efforts [12] [4]. Prediction of defective IaC scripts can help IT organizations make informed decisions about allocating inspection efforts to those IaC scripts that are likely to be defective.

The goal of this paper is to help software practitioners in prioritizing their inspection efforts for infrastructure as code (IaC) scripts by proposing defect prediction model-related metrics.

We examine the following research question: RQ: What metrics can be used to characterize defective infrastructure as code (IaC) scripts?

2 METHODOLOGY
We first provide definitions, then we describe our methodology.

• Defect: An imperfection in an IaC script that needs to be replaced or repaired. We follow the IEEE definition of defects [8].
• Defect-related commit: A commit whose message indicates that an action was taken related to a defect.
• Defective script: An IaC script which is listed in a defect-related commit.

We use the following steps to derive necessary metrics:

Repository Collection: We mine open source version control repositories from three organizations: Mozilla, Openstack, and Wikimedia.
Commit Message Processing: We extract messages from commits, where at least one IaC script is modified.
Determining Defect-related Commits: We apply qualitative analysis with the help of multiple raters to determine defect-related commits.
Constructivist Grounded Theory (CGT): We apply CGT on defective commit messages to derive metrics.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/authors(s).

ICSE ’18 Companion, May 27-June 3, 2018, Gothenburg, Sweden © 2018 Copyright held by the owner/authors(s).
https://doi.org/10.1145/3183440.3195034

1https://puppet.com/
We discuss the limitations of our paper as following:

**Metrics:** We used defect-related commits to identify the 18 metrics. We acknowledge that our selection of metrics is not comprehensive. In the future, we plan to investigate other metrics, such as process metrics, for predicting defective IaC scripts.

**Datasets:** We use three datasets to evaluate our methodology. We acknowledge that more datasets can help generalizing our findings. Also, the datasets do not include temporal information i.e. we do not account for presence or absence of defects across times. We plan to include more datasets in future that will also account for the temporal information for defects in IaC scripts.

### 5 CONCLUSION

IaC is one of the fundamental pillars to implement DevOps. But similar to software source code, IaC scripts are susceptible to defects. Defect prediction models for IaC can help software teams to prioritize inspection efforts. Using Constructivist Grounded Theory technique, we identified one generic metric (lines of code), 13 IaC code-related metrics, and four churn metrics that characterize defective IaC scripts.

### REFERENCES


**Table 1: Metrics that Characterize Defective IaC Scripts**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Measurement Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code (LOC) [GE]</td>
<td>Total lines of code</td>
</tr>
<tr>
<td>Churn Count (CC) [CH]</td>
<td>Count of times a script was churned</td>
</tr>
<tr>
<td>Churn Deleted (CD) [CH]</td>
<td>Total lines deleted/Total lines in script</td>
</tr>
<tr>
<td>Churn Total (CT) [CH]</td>
<td>Total lines in the script added or modified</td>
</tr>
<tr>
<td>Churn per LOC (CT_PER_LOC) [CH]</td>
<td>CT_PER_LOC = CT/LOC</td>
</tr>
<tr>
<td>Comment (CMT) [CO]</td>
<td>Count of comments</td>
</tr>
<tr>
<td>Command Execution (CMD) [CO]</td>
<td>Count of cmd syntax occurrences</td>
</tr>
<tr>
<td>Ensure (E) [CO]</td>
<td>Count of ‘ensure’ syntax occurrences</td>
</tr>
<tr>
<td>FILE [CO]</td>
<td>Count of file syntax occurrences</td>
</tr>
<tr>
<td>File Mode (FM) [CO]</td>
<td>Count of ‘mode’ syntax occurrences</td>
</tr>
<tr>
<td>Include (INCL) [CO]</td>
<td>Count of include syntax occurrences</td>
</tr>
<tr>
<td>URL [CO]</td>
<td>Count of URL occurrences</td>
</tr>
<tr>
<td>Location (LOCA) [CO]</td>
<td>LOCA = FILE + URL</td>
</tr>
<tr>
<td>Require (REQ) [CO]</td>
<td>Count of ‘require’ syntax occurrences</td>
</tr>
<tr>
<td>SSH Authorized Key (SSH_KEY) [CO]</td>
<td>Count of ssh_authorized_key syntax occurrences</td>
</tr>
<tr>
<td>Strings per LOC (STR_PER_LOC) [CO]</td>
<td>STR_PER_LOC = STR/LOC</td>
</tr>
<tr>
<td>Value Assignment (VA) [CO]</td>
<td>Total count of ‘=&gt;’ usages</td>
</tr>
</tbody>
</table>