2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) | 978-1-7281-8710-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/MSR52588.2021.00078

Andromeda: A Dataset of Ansible Galaxy Roles and Their Evolution

Ruben Opdebeeck, Ahmed Zerouali, Coen De Roover
{ropdebee, azeroual, cderoove}@vub.ac.be
Vrije Universiteit Brussel, Brussels, Belgium

***Abstract*—Cloud-native applications increasingly provision infrastructure resources programmatically through Infrastructure as Code (IaC) scripts. These scripts have in turn become the subject of empirical software engineering research. However, an often-overlooked part are the software ecosystems that have grown around the IaC languages. For example, Galaxy is an ecosystem for the popular Ansible IaC language. Galaxy features a large number of so-called "roles", which are reusable collections of Ansible code akin to libraries for general-purpose languages. In contrast to, and despite their similarities, such IaC ecosystems have enjoyed far less attention in the literature than library ecosystems for general-purpose languages.**

In this data showcase paper, we present Andromeda, the first dataset capturing the Ansible Galaxy ecosystem, its roles, and their evolution. Andromeda provides structural representations of more than $125\,000$ role versions, and upwards of $800\,000$ concrete changes between such versions extracted from the underlying git repositories. Andromeda aims to provide an extensive view of the contributor side of the Galaxy ecosystem, which we hope will stimulate additional research on IaC ecosystems.

***Index Terms*—Infrastructure as Code; dataset; mining software repositories; change distilling; Ansible**

I. Introduction

Managing large-scale digital infrastructures, such as clusters of cloud instances, is a complicated and time-consuming task. Fortunately, Infrastructure as Code (IaC) can automate many of the responsibilities by enabling its users to specify the various steps in setting up, configuring, and managing these infrastructures, in domain-specific languages. Ansible is one such language that has become increasingly popular in industry over the last few years [1], and has recently caught the attention of researchers [2]–[4].

In Ansible, an infrastructure developer can specify the steps necessary to configure the machines in an infrastructure as a series of *tasks*. For instance, to set up a cluster of database servers, a developer could write a series of tasks to install the necessary packages, create the database tables, and configure the network interface. *Variables* can be used to parametrise these tasks, e.g., to specify the administrator credentials, network port, etc.

Note that regardless of the actual goal of the infrastructure, installing database software on a certain platform will follow roughly the same steps. To enable reuse of the tasks involved in such a process, Ansible offers its own concept of libraries, called *roles*. For example, one could write a role to install and configure PostgreSQL on Debian and Ubuntu-based platforms. Another developer could then easily reuse this role in their own infrastructure definition, and customise the variables to configure the final installation to their liking.

Such reusable roles, along with other content such as plugins, are collected and indexed by the Ansible Galaxy ecosystem[1]. Here, ecosystem contributors can publish their open-source roles, allowing ecosystem consumers to easily find and reuse their efforts. Although Galaxy can be compared to Maven or npm, the interaction between the ecosystem clients and the roles differs from what is found with traditional programming language libraries. For instance, roles do not offer methods to be called by users. Instead, they are included into a task list and parametrised with concrete values. As such, the Galaxy ecosystem is an ecosystem like no other.

Unfortunately, research into this ecosystem, and other IaC ecosystems, is nascent [5]. To stimulate new research into this domain, we have collected a dataset of Galaxy's content. In this data showcase paper, we present *Andromeda*, a dataset consisting of four parts:

- Metadata harvested from Galaxy for over 140K entities spread across seven entity types;
- Git commit and tag metadata from more than 25K repositories containing roles;
- Structural models for over 125K role versions;
- Distilled changes between structural model versions, totalling more than 800K concrete changes categorised into 41 change types.

This dataset represents an extensive snapshot of the ecosystem contributor side of Ansible Galaxy.

An open version of the dataset is available at https://doi.org/10.6084/m9.figshare.13664519. In this version of the dataset, personally identifiable information (PII) is either removed or obfuscated. We do not provide raw API responses from Galaxy, or the git repositories themselves, since we cannot guarantee this data is void of any PII.

II. Dataset Description

The Andromeda dataset consists of four core parts which are interlinked using cross-references. We provide all of the data as YAML files, and each part is accompanied by an index. We discuss each part individually below. For a detailed description of each attribute in the dataset, we refer to the documentation provided with the download. An overview of

[1] https://galaxy.ansible.com

978-1-7281-8710-5/21/$31.00 ©2021 IEEE
DOI 10.1109/MSR52588.2021.00078

Type	Count
Namespaces	27 030
Provider namespaces	27 358
Repositories	27 800
Content	26 769
Roles	26 834
Community surveys	850
Tags	7 151

(a) Galaxy metadata

Type	Count
Commits	2 308 309
Tags	113 237

(b) Git repository metadata

Type	Count
Struct. models	126 663
Change sets	101 047
Changes	814 025

(c) Structural models

TABLE I: Summary of content counts in Andromeda.

the dataset content and the number of entries for each type of content is provided in Table I.

A. Ansible Galaxy Metadata

The first part of our dataset is an extensive collection of metadata harvested from Ansible Galaxy, provided in the "GalaxyMetadata" directory. Figure 1 depicts its schema.

The Galaxy metadata consists of seven core entity types.

1) *Namespaces* contain information on namespaces in Galaxy, which group content;
2) *Provider namespaces* are similar to namespaces, but represent users or organisations on GitHub;
3) *Repositories* aggregate metadata on a GitHub repository, which may contain multiple pieces of content;
4) *Content* can be any type of content on Galaxy, e.g., roles, Ansible plugins, etc.;
5) A *role* is a specific type of content, containing additional metadata;
6) *Community surveys* contain responses to quality surveys for repositories, submitted by Galaxy users;
7) Finally, *tags* contain information on all content tags known to Galaxy.

We have removed personally identifiable information (PII), such as full names, company names, locations, and e-mail addresses, from all of these entities due to privacy concerns. For similar reasons, we do not provide any harvested information on Galaxy users except for numerical user IDs.

B. Git Repository Metadata

The second part of the Andromeda dataset contains additional metadata extracted from the underlying git repositories. This includes all commits to the main branch of the repository, as well as tags. The information is aggregated into one file per repository in the "RepositoryMetadata" directory, further subdivided into directories per GitHub repository owner.

For privacy reasons, we obfuscated commit author, committer, and tagger names and e-mail addresses. Rather than removing this information entirely, we hashed it with the SHA1 hash to enable identifying different commits of the same author, but to prevent an author from being identified as a specific person. The same hashes also enable interoperability with other datasets.

C. Structural Models for Roles

Andromeda's third part contains structural models of each role version whose tag follows the semantic versioning format, as well as for the latest commit in the repository. These are presented in the "StructuralModels" directory in separate files per role, named after the role's canonical ID, i.e., `<namespace name>.<role name>`. The structural model generalises over the role's source code, and standardises different syntactical styles. In what follows, we provide a brief summary of this model. A more detailed description can be found in our previous work [2], which introduced this structural representation, but did not actually include it in its dataset.

At the highest level, each model contains a collection of files, each representing one Ansible source file in the role repository. We distinguish between five types of files. The *metadata file* represents the `meta/main.yml` file of the repository, containing information similar to the one found in the role metadata from Galaxy, described in II-A. *Task files* contain a sequence of *blocks*, in turn containing a sequence of *tasks*. Blocks and tasks additionally contain the keywords defined on them in the source code. *Handler files* are similar to *task files*, but instead contain *handler blocks* and *handler tasks*. The main difference between the two is their semantics, as tasks are executed sequentially whereas handlers need to be notified explicitly. Finally, each model contains two types of files containing variables, namely *default variable files* and *role variable files*. These files contain key-value mappings of variable names to variable values. The difference between default and role variables is again semantic, since role variables are intended to be constants, whereas default variables establish the role's interface and can be overridden by a user.

In addition to these files, the structural model contains a list of files which could not be processed, e.g., due to syntax errors or invalid keywords. We amended this list with the reason why converting the file failed.

D. Distilled Structural Changes Between Role Versions

The final part of the Andromeda dataset consists of changes between consecutive versions of the structural model. These changes were distilled using the algorithm presented in [2]. They are stored in the "StructuralRoleEvolution" directory in files named after the role's canonical ID. Each file contains a list of change sets, where each change set stores the name of the old and new revision of the model, as well as a list of concrete changes.

Each of these concrete changes are categorised into one of 41 change types. These change types are constructed orthogonally from a combination of change kind (i.e., addition, removal, edit, or relocation) and the type of the structural model element affected. Possible combinations include the addition of a task file, the removal of a default variable, etc. A full overview of the change types is provided in [2].

For each concrete change, we store the location of the element. For example, `tasks/main.yml[0].block[0]`

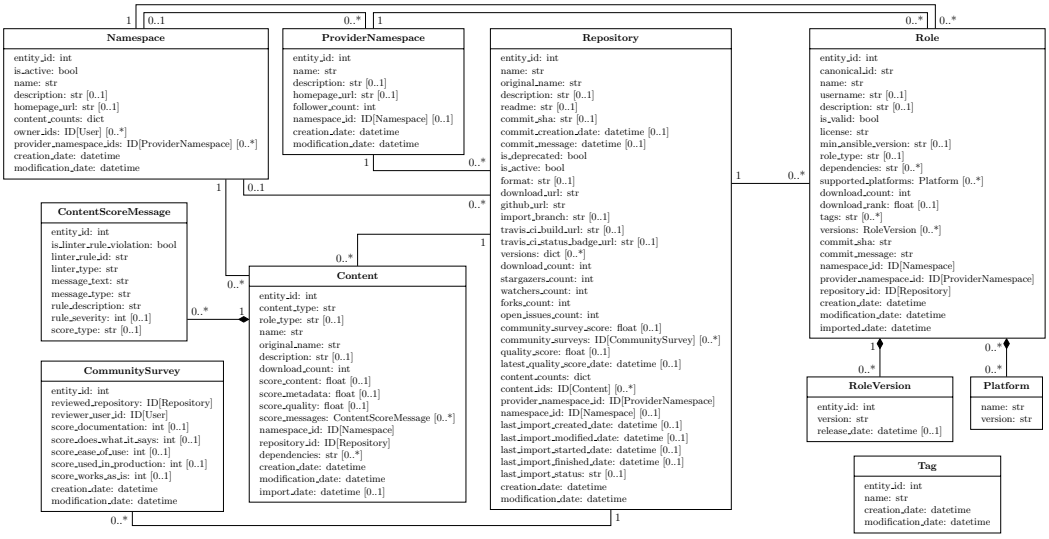

Fig. 1: UML schema of the collected Galaxy metadata and their relationships. Attributes with a cardinality of `[0..1]` may be `null`.

points to the first task of the first block in the `tasks/main.yml` file. Addition changes additionally store the added element, whereas removals store the removed element. Edits store the old and new values of the changed element, and relocations store the old and new location of the element.

III. Data Collection and Challenges

To build this dataset, we used our tool called Voyager, available at https://github.com/ROpdebee/Voyager. Voyager's data collection and extraction pipeline goes through a number of phases. We briefly summarise these phases below, and mention the main technical challenges we encountered.

In the first stage of the pipeline, Voyager polls various endpoints of the Galaxy API to collect raw metadata. We collected the latest data on January 20th, 2021. The main challenge we encountered were occasional internal server errors returned by the API. This occurred mostly when polling roles, and we found this error to be unrecoverable. To alleviate the issue, we additionally polled the role search endpoint, which did not exhibit these errors, and later deduplicated the data returned by both endpoints based on the role ID.

The tool subsequently converts these raw API responses into the schema presented in II-A. Most notably, this phase removes attributes that are redundant in the sense that they can be derived trivially from already-captured information, e.g., the git clone URL from the GitHub URL. Moreover, it cleans up certain values, such as `null` to 0 when a number is expected, and converts timestamps to RFC 3339 format. Finally, it converts references to other entities to cross-references for the schema. Importantly, we do not perform any filtering on the extracted data, since we intend this dataset to be a complete snapshot of Ansible Galaxy suited for various applications. As such, it would be unwise to impose our own filters, which may hamper the dataset's applicability.

In the next phase, we attempt to clone all git repositories present in the Galaxy metadata. Any repository that failed to clone, e.g., because of invalid URLs or private repositories, is ignored. For the remaining repositories, we extract the tag and commit information, as presented in II-B. For any repository whose Galaxy metadata indicates that it contains an Ansible role, the tool builds a structural model for all extracted tags and the latest commit to the repository.

We use Ansible's internal parser and data structures to build the structural model. This enables us to use its validations, thereby preventing malformed files from being converted. However, Ansible's internals are not designed with analysis in mind. For example, Ansible eagerly pre-loads dependencies and statically imported files, but defers the loading of dynamically included files until runtime. For the purposes of the structural model, both statically imported and dynamically included files are important to represent. On the other hand, dependencies do not need to be loaded, and in some cases, may not be available. Moreover, although roles contain multiple files, if one of the files contains a syntax error, Ansible refuses to load the role in its entirety. In previous work with the structural model, where we did not account for this, roughly 11% of the role versions therefore failed to load [2]. To overcome these challenges, we partially customised Ansible's internals to prevent aggressive pre-loading, and keep track of any file containing an error. These changes enabled us to (partially) parse all role versions, except for one role which recursively loaded itself, which led to a stack overflow.

In the final phase of the pipeline, Voyager distilled the structural changes from consecutive versions of the structural model. We only targeted versions that matched the semantic

versioning format[2]. In addition, we extracted the changes between the last version and the latest commit. We extracted no changes for repositories without tags, as there was only one structural model to be built.

IV. Research Opportunities

This dataset shows that Infrastructure-as-Code languages are supported by ecosystems. For general-purpose programming languages, research on their ecosystems is ubiquitous. However, IaC as a whole, and IaC ecosystems in particular, are currently understudied in academia.

The Andromeda dataset can be used to answer the general question of how the Ansible ecosystem differs from well-known software ecosystems. This is, of course, a very broad question, yet the dataset provides all the necessary information to answer specific sub-questions in empirical studies. For example, in previous work, we used an aggregation of extracted git tags and distilled code changes to study the use of semantic versioning in role evolution [2].

Likewise, tool builders can benefit from this dataset. For instance, NLP-based tools can utilise the large amount of textual content, such as content descriptions and commit messages, as a source of information. Furthermore, the structural models provide a solid basis for analysing the structure and evolution of role implementations. Existing Ansible analysis tools [3], [6], [7] have, in contrast, relied on syntactical representations so far.

V. Limitations and Future Work

The main limitation of our dataset is the extent of the structural model. The model only considers the Ansible code parts of a role. However, roles can contain auxiliary files, such as resources that need to be installed, test code, etc. Moreover, Galaxy contains content other than roles, such as plugins which are often written in Python. Neither the auxiliary files, nor the other types of content, are captured by the models we build. Consequently, since the change distilling algorithm inherently depends on these structural models, changes to such content are not represented in this dataset. In future work, we hope to overcome this limitation by expanding the scope of the structural model.

Finally, there exist other popular IaC languages which offer ecosystems similar to Galaxy, such as Puppet's PuppetForge[3], and Chef's Supermarket[4]. Therefore, similar datasets could be constructed for these ecosystems.

VI. Related Datasets

A number of IaC-related works have published their evaluation datasets [3], [4], [8]–[10]. However, these datasets only contain a relatively low number of projects, and consist of mostly analysis results. In contrast, the dataset we present in this paper consists of a large and diverse amount of content, complete with metadata aggregated from multiple sources. Moreover, we provide structural representations of the roles in the dataset, as well as the changes made between their versions. Finally, apart from [9], these datasets do not distinguish between the consumer side and the contributor side of the ecosystem. Our dataset focuses exclusively on the contributor side. Thus, some of these complementary datasets could be combined with ours to study the interaction between ecosystem contributors and consumers.

There is also some overlap between our dataset and the well-known GHTorrent [11], since we also extract tags and commits from git repositories. Similarly, we provide the number of stars, forks, etc., for GitHub repositories, although we harvested this data from Ansible Galaxy rather than from GitHub directly. The main distinction between our dataset and GHTorrent is that we do not aim to provide a grand overview of all of GitHub. Instead, our dataset focuses on Ansible roles, and thus only includes information from repositories containing such roles. Nonetheless, since we provide GitHub URLs for these repositories, the GHTorrent dataset can be queried to retrieve more information about the repository.

Finally, we have previously made available an earlier version of this dataset [2]. However, this earlier version of the dataset included very little data from the Galaxy ecosystem, and did not include structural models or structural differences. Moreover, in that paper, the dataset was not described in detail. In contrast, the dataset presented in this paper provides a much larger collection of Galaxy metadata, and includes structural models and changes.

VII. Conclusion

In this data showcase paper, we presented Andromeda, a dataset of Ansible Galaxy metadata and its roles. Andromeda provides metadata harvested from the Ansible Galaxy software ecosystem. In addition, it includes metadata harvested from Ansible role repositories, including information on more than 2 million commits. We also provide structural representations extracted from over 125 000 role versions, and upwards of 100 000 change sets of concrete changes applied between different versions of a role.

Andromeda thus provides an extensive overview of the Ansible Galaxy ecosystem. This makes it a valuable source of information and opens up interesting opportunities for new research, including empirical studies and technical research. With this dataset, we hope to stimulate research in the Infrastructure as Code domain, and more specifically on IaC ecosystems, an understudied domain today.

Acknowledgements

This research was partially funded Research Foundation – Flanders (FWO) (grant number 1SD4321N) and by the Excellence of Science project 30446992 SECO-Assist financed by FWO-Vlaanderen and F.R.S.-FNRS.

[2]Although we only targeted tags, Voyager is also equipped to distil changes between consecutive commits.

[3]https://forge.puppet.com/

[4]https://supermarket.chef.io/

REFERENCES

[1] Michele Guerriero, Martin Garriga, Damian A. Tamburri, and Fabio Palomba. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In *Proceedings of the 35th IEEE International Conference on Software Maintenance and Evolution (ICSME19), Industrial Track*, 2019.

[2] Ruben Opdebeeck, Ahmed Zerouali, Camilo Velázquez-Rodríguez, and Coen De Roover. Does Infrastructure as Code adhere to Semantic Versioning? an analysis of Ansible role evolution. In *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 238–248. IEEE, 2020.

[3] 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.

[4] S. Dalla Palma, D. Di Nucci, F. Palomba, and D. A. Tamburri. Within-project defect prediction of Infrastructure-as-Code using product and process metrics. *IEEE Transactions on Software Engineering*, pages 1–1, 2021.

[5] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams. A systematic mapping study of Infrastructure as Code research. *Information and Software Technology*, 108, 2019.

[6] 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.

[7] Stefano Dalla Palma, Dario Di Nucci, and Damian A. Tamburri. AnsibleMetrics: A Python library for measuring Infrastructure-as-Code blueprints in Ansible. *SoftwareX*, 12:100633, 2020.

[8] 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.

[9] Waldemar Hummer, Florian Rosenberg, Fábio Oliveira, and Tamar Eilam. Testing idempotence for infrastructure as code. In *Proceedings of the 14th International Middleware Conference (Middleware13)*, 2013.

[10] Tushar Sharma, Marios Fragkoulis, and Diomidis Spinellis. Does your configuration code smell? In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR16)*, pages 189–200, 2016.

[11] Georgios Gousios. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 233–236, Piscataway, NJ, USA, 2013. IEEE Press.