7ZA-08

# Ansible の保守性向上を目的とした DSL の提案と試作

阿部 創志朗† 明治大学理工学部†
早川 智一‡ 明治大学理工学部‡

## 1. はじめに

構成管理ツールは，IT（Information Technology）システムの構成や展開などを自動化するソフトウェアである．主要な構成管理ツールとして，Ansible [1] や Chef [2]・Puppet [3]・Salt [4] がある．

近年，構成管理ツールの中でも Ansible が広く用いられている．この理由としては，他の構成管理ツールと比べて運用が容易な点が挙げられる．具体的には，Ansible の実行に必要なのは SSH のみで，専用ソフトウェアの常駐を必要としない．

Ansible は，Playbook と呼ばれる設定ファイルに YAML（YAML Ain't Markup Language）形式で構成管理の設定を記述する．これにより，プログラミング言語などで設定を記述する他の構成管理ツールに比べ，より簡潔に設定を記述できる．

しかし，Playbook の仕様には保守性の課題があると我々は考える．なぜならば，Playbook で使用できる変数には 22 の優先順位が存在し，利用者がこれらを適切に理解して使用しないと，意図しない変数の上書き（隠蔽や再代入）が発生するためである．これにより，構成管理が正しく行えなくなる可能性がある．

この課題に対する先行研究は存在しないが，他の構成管理ツールでは利用者が意図しない変数の上書きを抑制する工夫がされている（2 章）．

この課題を解決するため，本論文では Playbook の代替フォーマットを提案する．具体的には，（1）Playbook を直接置換可能で，（2）一切の変数の上書きを許容しない——DSL（Domain-Specific Language）を用いて保守性の向上を試みる．

本論文の構成は次のとおりである．2 章では関連技術を紹介する．3 章では提案手法を示す．4 章では設計を説明する．5 章では実装を概説する．6 章では評価結果を報告する．7 章では今後の展望を述べる．

## 2. 関連技術

Puppet では，設定ファイルでの変数への値の再代入を不可能にしている．変数の不変性を上げている点で本研究と類似性がある．

Salt では，ドメイン名や IP アドレスなどのシステムのプロパティを設定ファイルの変数とは別のものとして扱う（Ansible では，システムのプロパティは Ansible facts として設定ファイルの変数の 1 つとして扱われる）．提案 DSL では，Salt と同様にシステムのプロパティを設定ファイルの変数とは別のものとして扱う．

## 3. 提案手法

本論文では，DSL とそのチェッカを提案する．

提案 DSL は，設定ファイルでの変数の上書きと Ansible facts への代入を禁止した Playbook のサブセットである（図 1）．具体的には，Playbook の文法から（1）同じ名前の変数を宣言する機能，（2）Ansible facts の変数へ値を代入する機能——を除いたものである．これにより，Ansible の実行時における変数の上書きを防止できる．

提案 DSL とチェッカの想定ユースケースを図 2 に示す．提案 DSL の文法は Playbook の文法のサブセットであるため，Ansible Lint などの既存のツールを用いて Playbook としての正しさをまず検査する．次に，提案 DSL のチェッカを用いて，変数の上書きに関する検査を行う．

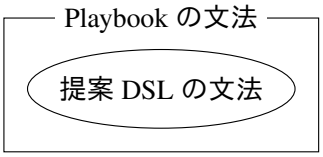


図 1　Playbook の文法と提案 DSL の文法の関係


図 2　提案 DSL のユースケース

A proposal for DSL that improves Ansible maintainability
†Soshiro Abe, School of Science and Technology, Meiji University, s_abe@cs.meiji.ac.jp
‡Tomokazu Hayakawa, School of Science and Technology, Meiji University, t_haya@cs.meiji.ac.jp

## 4. 設計

提案 DSL は，Ansible の実行時にすべての変数が不変となるように設計した．これにより，利用者が設定した変数の値が意図せずに上書きされることを防止できる．たとえば，コード 1 の 2 つ目のタスク `override variable` は，変数 `hello` を上書きするため提案 DSL ではエラーとなる．

提案 DSL のチェッカは，Ansible の設定ファイルを読み込み，設定ファイルに変数の上書きがある場合にエラーを表示する．

提案 DSL には，全ての変数の上書きを禁止したことで，利用者が意図したとしても変数の上書きを行えないという制限が存在する．この制限により，変数の再利用ができなくなり利便性が低下する恐れがある．一方で，提案 DSL によって Playbook の頑健性は向上するため，利便性と頑健性とのトレードオフであると我々は考える．

## 5. 実装

我々は，提案 DSL のチェッカを表 1 のソフトウェアを用いて実装した．チェッカの実装には Python と ANTLR [5] を用いた．これは，Ansible が Python で実装されているため，チェッカも Python で実装することで，Ansible の実行環境でチェッカを動作可能にするためである．

## 6. 評価

我々は，提案 DSL の有用性を評価するために，既存の Playbook を提案 DSL のチェッカに入力して受理率を測定した．具体的には，Ansible Examples [6] の 13 個のリポジトリのうち，実行できないリポジトリ 1 個を除いた 12 個のリポジトリを入力として評価に用いた．表 2 に評価結果を示す．表の適用可否が〇のリポジトリは，提案 DSL をそのまま適用できることを示している．

表より，75%（12 個中 9 個）のリポジトリに提案 DSL が適用可能であることが分かった．この評価結果から，一定数の既存の Playbook に対しても，提案 DSL を用いることでより安全な構成管理が実現できると我々は考える．

提案 DSL を適用できなかった 3 個のリポジトリでは，Playbook の記述を簡潔にするために変数を再利用していた．そのため，変数の再利用が変数の上書きとしてチェッカに検出されて，提案 DSL を適用できなかった．

コード 1　提案 DSL ではエラーとなる Playbook の一例

```
- name: ansible playbook example
  hosts: localhost
  tasks:
    - name: print Hello, world!
      ansible.builtin.debug:
        var: hello
      vars:
        hello: "Hello, world!"
    - name: override variable
      ansible.builtin.debug:
        var: hello
      vars:
        hello: "This variable is overridden"
```

表 1　提案 DSL のチェッカに使用したソフトウェア

| ソフトウェア | バージョン | 備考 |
|---|---|---|
| ANTLR | 4.7.2 | パーサジェネレータ |
| PyYAML | 6.0 | Playbook の読み込みの実装 |
| Python | 3.9.5 | 提案 DSL のチェッカの実装 |

表 2　提案 DSL が適用可能なリポジトリ

| リポジトリ名 | 適用可否 |
|---|---|
| jboss-standalone | 〇 |
| lamp_haproxy | × |
| lamp_simple | 〇 |
| lamp_simple_rhel7 | 〇 |
| mongodb | 〇 |
| phillips_hue | × |
| rust-module-hello-world | 〇 |
| tomcat-memcached-failover | 〇 |
| tomcat-standalone | 〇 |
| windows | × |
| wordpress-nginx | 〇 |
| wordpress-nginx_rhel7 | 〇 |

## 7. おわりに

本論文では，Ansible の保守性向上を目的とした DSL を提案した．評価の結果は，提案 DSL が既存の Playbook に一定程度適用可能であることを示した．この結果から，提案 DSL を用いることで，より安全に構成管理を行うことができる場合があるという結論を我々は得た．

今後の展望としては，大規模な構成管理を行う複雑な Playbook に対しても提案 DSL を適用できるように改良していく予定である．具体的には，変数の不変性に加えて可視性も考慮し，安全な構成管理を実現しつつ利便性を損ないにくい DSL への改良を検討している．

## 参考文献


[1] Red Hat, Inc.: Ansible is Simple IT Automation, Red Hat, https://www.ansible.com/.
[2] Chef Software, Inc.: Chef Software DevOps Automation Solutions | Chef | Chef, Chef Software, https://www.chef.io/.
[3] Puppet, Inc.: Powerful infrastructure automation and delivery | Puppet, Puppet, https://puppet.com/.
[4] SaltStack: Home - Salt Project, https://saltproject.io/.
[5] Parr, T.: ANTLR, https://www.antlr.org/.
[6] Red Hat, Inc.: ansible/ansible-examples, https://github.com/ansible/ansible-examples.