serverspec: 宣言的記述でサーバの状態をテスト可能な汎用性の高いテストフレームワーク

宮下 剛輔[†1,†2,a)]　栗林 健太郎[†1,b)]　松本 亮介[†3,c)]

概要：システムの大規模・複雑化に伴い，サーバの構築・運用を効率化するために，サーバの状態をコードで記述する手法が数多く提案されている．それらの手法を効率良く扱うプロセスとして，テスト駆動開発の手法をサーバ構築に応用した Test-Driven Infrastructure が提案されている．このプロセスを支援するテストフレームワークもいくつか登場しているが，あるものは特定の構成管理ツールに依存，またあるものは OS 毎の違いを自ら吸収しなければならないなど，汎用性に難がある．そこで，本論文では，特定の構成管理ツールや OS に依存することなく，サーバの状態を汎用的かつ可読性の高いコードでテスト可能なテストフレームワークを提案する．提案手法では，汎用性を高めるために，これまでの OS や構成管理ツール固有の振る舞いを整理して一般化し，汎用コマンド実行フレームワークとして定義する．続いて，テストコード記述の抽象度を高め可読性を上げるために宣言的な記法で汎用コマンド実行フレームワークを操作できる制御テストフレームワークを定義する．これにより，管理者が OS や構成管理ツールの違いを気にすることなくサーバの状態を容易にテストできるようになり，サーバの運用・管理コストを低減できる．また，フレームワークを用途別に分離して定義することにより，制御テストフレームワークを独自の記述に変更する事も容易である．提案するテストフレームワークを serverspec と名付けた．

1. はじめに

システムの大規模・複雑化に伴い，サーバの構築・運用を効率化するために，サーバ構成を宣言的なコードで記述・管理するという手法が提案され，その実装として構成管理ツール CFEngine[1] が 1993 年に登場している．その後様々な構成管理ツールが生み出されているが [2]，2005 年の Puppet の登場 [3] と 2006 年の Amazon EC2 の登場 [4] をきっかけに “Infrastructure as Code” という概念が台頭し，“Agile infrastructure and operations”[5] という流れが生まれている．これはアジャイル開発の手法をサーバ構築・運用に適用しようという潮流である．（ここで言う “Infrastructure” はアプリケーションを載せるための土台を意味し，OS やミドルウェアといったソフトウェアレイヤーを含む．）

この流れの中で，“Agile infrastructures and operations” を効率よく実践するためのプロセスとして，テスト駆動開発の手法をサーバ構築・運用に応用した “Test-Driven Infrastructure”[6] というプロセスが提案されており，このプロセスを支援するテストフレームワークがいくつも登場している [7][8][9][10][11][12].

これらのうち，ChefSpec, rspec-puppet は，構成管理ツール固有の言語で書かれたコードの内容をテストするのみで，実際にコードをサーバに適用した結果はテストしない．そのため，単体テストしては利用できるが結合テスト用途には利用できない．Cucumber-chef, minitest-chef-handler は Chef という特定の構成管理ツールに依存している．そのため，Chef 以外の構成管理ツールでは利用することができない．Test Kitchen や rspec-system は，テスト用 VM の作成，テスト用 VM への構成管理ツールの適用，テストの実行をトータルで行う統合テストスイートであるが，組み込みのテスト機構は汎用性に乏しく，特定の構成管理ツールに依存していたり，OS やディストリビューション毎の違いを意識したテストコードを書く必要がある．

我々は，テストフレームワークの汎用性を高めるために，構成管理ツール特有の振る舞い（例えば，パッケージのインストールやシステムユーザの作成など）を抽出・一般化し，それらをテストするためのコマンドを OS やディストリビューション毎に分離，その上で OS や実行形式の違いを吸収するレイヤーを設けることにより，汎用コマンド実行フレームワークを定義した．

[†1] 現在，株式会社 paperboy&co.
[†2] 現在，帝京大学
[†3] 現在，京都大学
[a)] gosukenator@gmail.com
[b)] kentarok@gmail.com
[c)] matsumoto_r@net.ist.i.kyoto-u.ac.jp

続いて，テストコードの記述の抽象度を高め可読性を上げるために，宣言的な記法で汎用コマンド実行フレームワークを操作できる制御テストフレームワークを定義した．これにより，テストコードのメンテナンス性を高め，サーバの運用・管理コストを低減することができる．また，フレームワークを用途別に分離して定義することにより，制御テストフレームワークを独自の記法に変更することも容易である．例えば，本論文で提案するテストフレームワークでは，テスト記法として RSpec[13] を採用しているが，minitest[14] に差し替えたり，あるいはまったく独自の記法に差し替えたりすることも可能である．このテストフレームワークを serverspec[15] と名付けた．serverspec を採用している企業も既に存在する [16].

本論文の構成について述べる．2 章ではサーバの構成管理とテスト手法について更に詳しく述べる．3 章では提案するサーバのテスト手法を詳細に述べ，4 章では提案手法によりもたらされた成果を論じ，5 章で結びとする．

2. サーバの構成管理とテスト手法

本論文で提案するストフレームワークは，サーバ構成を記述したコードのテストをいかに効率よく行うか，という観点から出発している．そこで代表的な構成管理ツールを例に，ツールと言語の特徴，テスト手法ならびに従来手法の問題点について言及する．

2.1 CFEngine から Chef へ

安価な UNIX ライク OS を搭載したサーバが普及し，TCP/IP により異なる OS を搭載したサーバ同士がネットワーク接続可能になったことにより，システムが大規模化・複雑化したことで，シェルスクリプトに代わり，サーバの設定を宣言的なコードで扱う構成管理手法が提案され，その実装として CFEngine[1] が登場した．

2005 年には CFEngine に影響を受け，発展させた Puppet[3] が登場, 2009 年には Puppet から影響を受けた Chef? が登場している．CFEngine, Puppet は独自の DSL でサーバの構成を記述するが，Chef は Ruby による内部 DSL で記述する．その特性ゆえ，Chef はシステム管理者よりも開発者に強く訴求し，Amazon EC2 の様な開発者自身がサーバ構築・運用を行える環境の普及とともに，広まりを見せている．

2.2 Chef から Test-Driven Infrastructure へ

CFEngine や Puppet では，記述できることが独自 DSL の範囲内に収まるため，コードは比較的簡易なものとなるが，Chef は Ruby でできることはどんなことでも記述できるため，システムが複雑になるにともない，それを記述するコードも複雑になる．そこでサーバ構成を記述したコードに対し，アジャイル開発におけるテスト駆動開発のプロセスを適用する事で，効率よくコードを記述することができる，といった考えが生まれる．Test-Driven Infrastucture という概念は Chef コミュニティ周辺で発生したものであるが，それは偶然ではなく，このような Chef が持つ言語特性ゆえである．

2.3 Test-Driven Infrastructure におけるテスト手法の分類

Test-Driven Infrastructure におけるテストの種類は，テスト駆動開発における以下の 3 つに分類できる．

(1) 単体テスト
(2) 結合テスト
(3) 受け入れテスト

単体テストは構成管理ツールにおける “モジュール” に対するテストで，実際にコードをサーバに適用する前の段階で行うテストである．結合テストはコードをサーバに適用した後に行うテストで，コードが期待通りにサーバの設定を行ったかどうかをテストする．受け入れテストもコードをサーバに適用した後に行うテストだが，結合テストがサーバ内部の状態をテストするホワイトボックステストなのに対し，受け入れテストはサーバの外から見た振る舞いをテストするブラックボックステストであるという違いがある．

2.4 従来テスト手法の問題点

単体テストツールとしては Chef には ChefSpec，Puppet には rspec-puppet というツールが存在する．その名が示すとおり，それぞれ Chef と Puppet 専用のツールであり，単体テストツールとしては十分であるが，実際にコードをサーバに適用した結果をテストすることはできないため，単体テストツールだけでは不十分である．

結合テストツールとしては，minitest-chef-handler，Test Kitchen，rspec-system が存在する．この内 minitest-chef-handler と Test Kitchen は Chef に依存したツールであり，他の構成管理ツールとともに利用することができない．Test Kitchen や rspec-system は，テスト用 VM の作成，テスト用 VM への構成管理ツールの適用，テストの実行をトータルで行う統合テストスイートであるが，組み込みのテスト機構は汎用性に乏しく，特定の構成管理ツールに依存していたり，OS やディストリビューション毎の違いを意識したテストコードを書く必要がある．

受け入れテストツールとしては cucumber-chef，leibniz が存在するが，これらは Chef に依存したツールであり，他の構成管理ツールとともに利用することができない．

参考文献

[1] Mark Burgess, “CFEngine: a system configuration engine”, University of Oslo report 1993,

http://cfengine.com/markburgess/papers/cfengine_history.pdf
[2] “Comparison of open-source configuration management software”, http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software.
[3] Luke Kanies, “Puppet: Next-Generation Configuration Management”, USENIX ;login, February 2006, Volume 31, Number 1 https://c59951.ssl.cf2.rackcdn.com/807-kanies_0.pdf.
[4] “Release: Amazon EC2 on 2006-08-23”, http://aws.amazon.com/releasenotes/Amazon-EC2/353.
[5] Patrick Debois, “Agile infrastructure and operations: how infra-gile are you?”, Agile, 2008. AGILE ’08. Conference, http://www.jedi.be/presentations/IEEE-Agile-Infrastructure.pdf.
[6] Stephen Nelson-Smith, “Test-Driven Infrastructure with Chef, 2nd Edition Bring Behavior-Driven Development to Infrastructure as Code” ISBN: 978-1449372200.
[7] “ChefSpec”, http://code.sethvargo.com/chefspec/.
[8] “rspec-puppet”, http://rspec-puppet.com/.
[9] “Cucumber-Chef”, http://www.cucumber-chef.org/.
[10] “minitest-chef-handler”, https://github.com/calavera/minitest-chef-handler.
[11] “Test Kitchen”, http://kitchen.ci/.
[12] “rspec-system”, https://github.com/puppetlabs/rspec-system.
[13] “RSpec“, http://rspec.info/.
[14] “minitest”, http://docs.seattlerb.org/minitest/.
[15] “serverspec”, http://serverspec.org/.
[16] “採用情報：キャリア採用 募集要項 - 京都勤務の募集職種 開発部門”, http://www.nintendo.co.jp/jobs/career/kyoto_sec1.html#p03.
[17] Mark Burgess, “CFEngine: a site configuration engine”, USENIX Computing systems, Vol8, No. 3 1995, http://cfengine.com/markburgess/papers/paper1.pdf.
[18] John Allspaw, Jesse Robbins, 角 征典, “ウェブオペレーション ーサイト運用管理の実践テクニック”, P57-58, ISBN: 978-4873114934.
[19] Sudhir, Pandey, “Investigating Community, Reliability and Usability of CFEngine, Chef and Puppet”, https://www.duo.uio.no/handle/10852/9083.