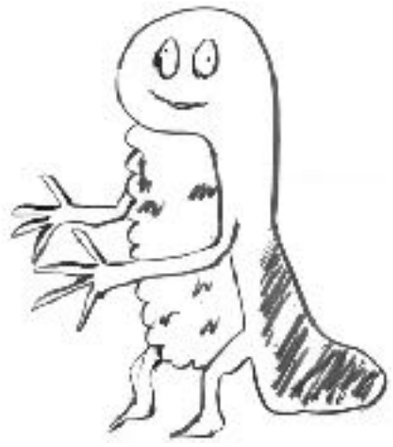
# 実践 Infrastructure as Code

## ~ Internet Week 2016 ~

Hideki Saito
Internet Initiative Japan Inc.
2016/11/30

# 自己紹介

- 氏名: 齊藤 秀喜 (さいとう ひでき)
- TwitterID: @saito_hideki
- 所属:
  - 株式会社インターネットイニシアティブ
  - 日本OpenStackユーザ会ボードメンバー
- 趣味: OpenStack, Ansible, IT Automation



# 本日のトピック

本セッションでは、DevOpsというコンセプトを実現するうえで不可欠な要素である、Infrastructure as Codeの本質と、それを達成するための方式(ツールチェーン)のサンプル実装を紹介します。

(1) 実践Infrastructure as Code 2016

(2) ツールチェーン デモンストレーション

# 実践
# Infrastructure as Code 2016

# Infrastructure as Code (IaC)

IaC ...手作業手順書をコード化して自動実行する。

「例えば１日に何度も行うような作業がある...そういった作業は、人間が実行するよりも、コード化してコンピュータに実行させたほうが効率も良いしミスも少ない！」 => 素晴らしい考えだ！

ところが、実際に始めてみると、
考えていたものと少し違っていた...
我々が触ってるレイヤーのインフラ設備って
そんなに高頻度で作業してたっけ？

# 誤解されがちですが

IaC によって得られる恩恵に関して、「自動化による直接的なOPEXの削減」という点が強調されていることが多いようです。

(1) オペレータの工数削減と負荷軽減

(2) 作業ミスの抑止

(3) 人海戦術からの脱却

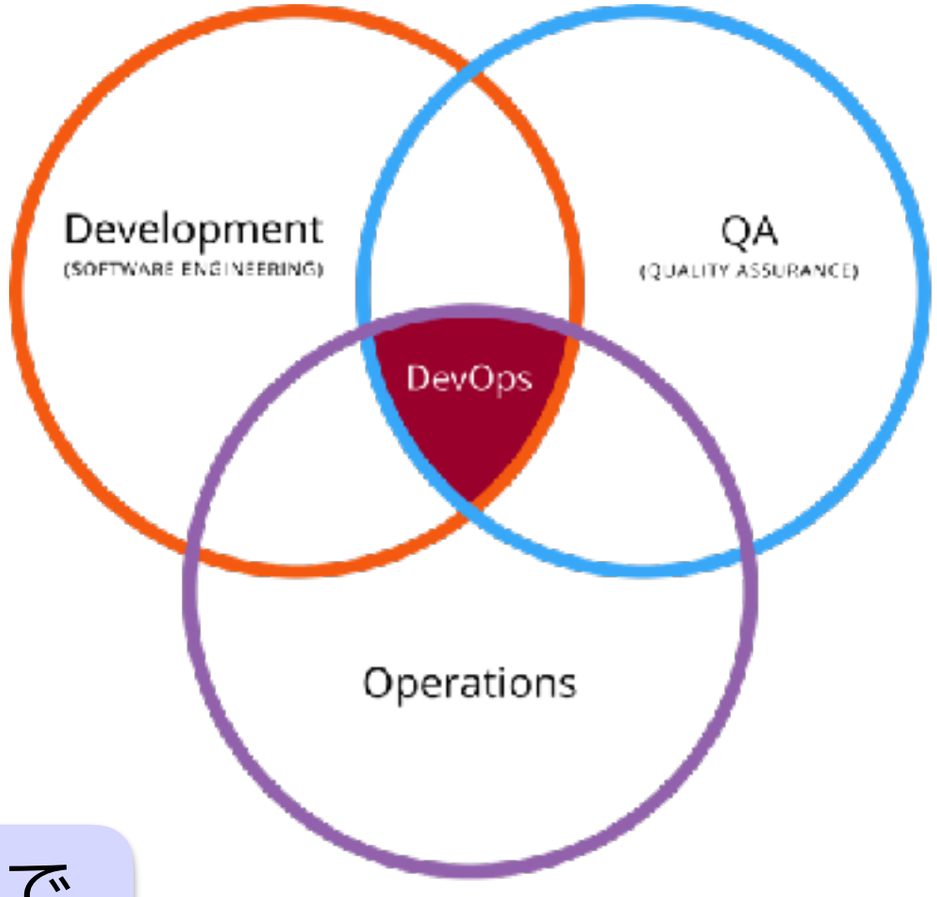
確かに正しいのですが、
目的は「それだけ」ではありません。

# 本来の目的は？

Devの現場で利用されている、QA/テストの仕組みを、我々インフラエンジニアの仕事にも適用できて、作業品質を高められるのが最大のメリットではないでしょうか。

1.手順書のコード化
2.コードのリビジョンの管理
3.チケットシステムによる課題管理
4.コードレビュー
5.テスト
6.デプロイシステム
7.インフラの構成管理

1～7をシステマチックに連携させることで
Infrastructure as Codeを実現する



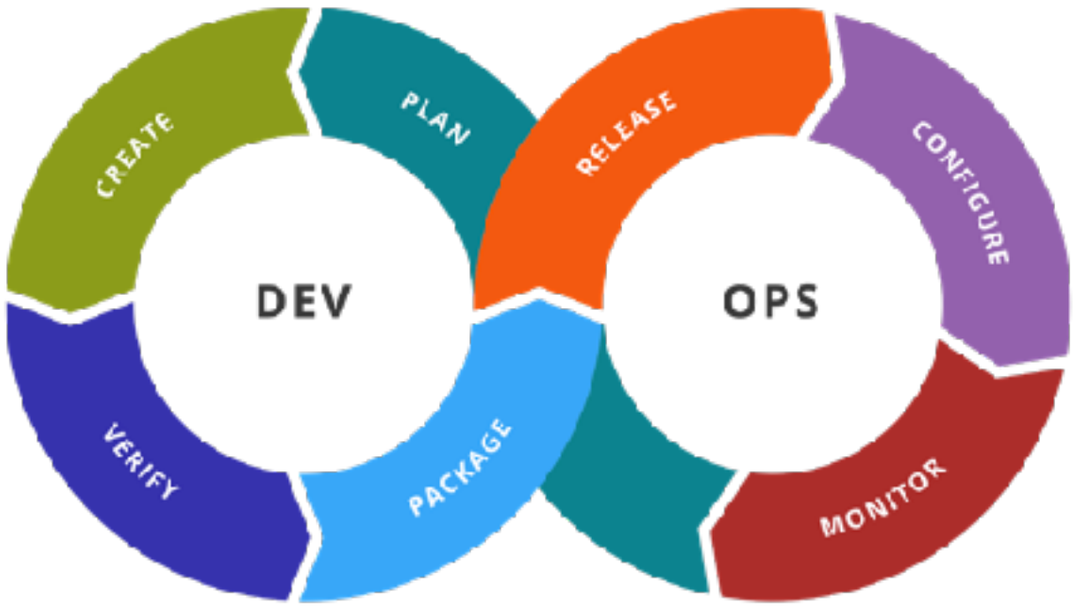

出展: Wikipedia

# IaCの現状とツールチェーン

2016年現在、Infrastructure as Code を単一のOSSプロダクトだけで実現することはできず、役割毎に、いくつかのツールを連携させることで実現してるケースがほとんどです。

[役割]

- ソースコード管理
- コードレビュー
- QA/テスト
- テストインフラ
- デプロイ



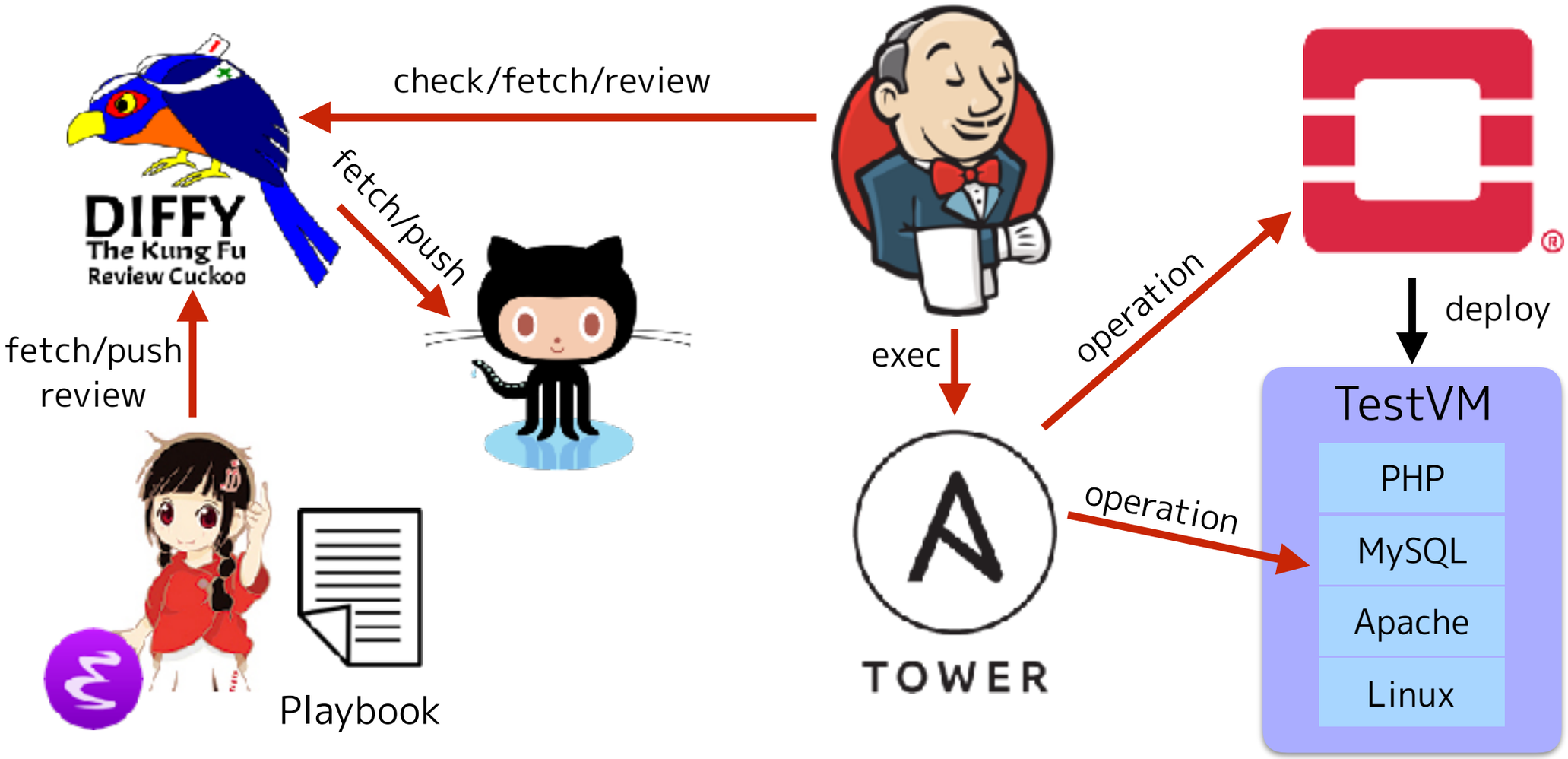

出展: Wikipedia

# 各ツールの役割

本セッションでは、(★)のツールについて、デモを交えつつ、少し掘り下げてご紹介します。

| | 役割 | 実例 | 機能 |
|---|---|---|---|
| 1 | ソースコード管理 | - Git | コード化された作業手順書を管理する |
| 2 | コードレビュー支援(★) | - Gerrit | 手順書のレビューを支援する |
| 3 | テスト支援 | - Jenkins | コードレビューの一貫として自動テストを行う |
| 4 | デプロイ管理(★) | - Ansible<br>- Ansible Tower | 自動化された作業手順を様々な環境に適用する |
| 5 | インフラ管理 | - OpenStack | テスト環境を提供する |

# 自動テスト環境構成例

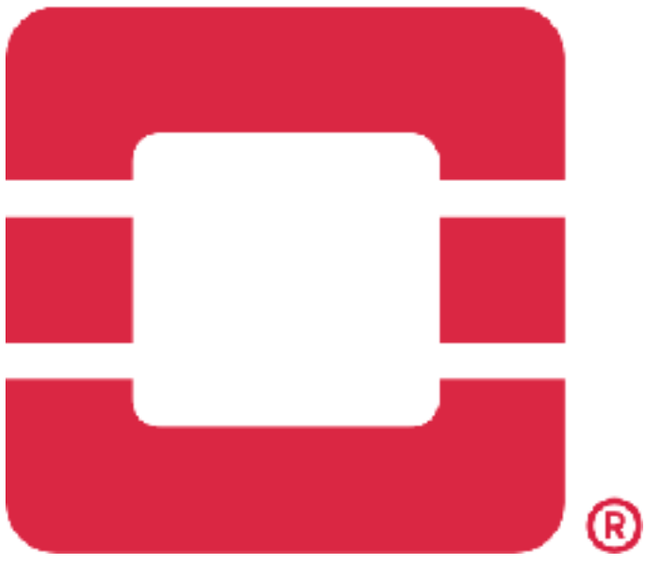

DIFFY
The Kung Fu
Review Cuckoo
check/fetch/review
fetch/push
fetch/push
review
exec
operation
deploy
operation
TestVM
PHP
MySQL
Apache
Linux
TOWER
Playbook

# コードレビュー: Gerrit + Git + Jenkins

コードレビューに特化したOSSで、さまざまなOSSプロダクト(OpenStackやAndroid等)のレビューに利用されています。
バックエンドでのソースコード管理にはGitを利用しています。


# デプロイ: Jenkins + Ansible + OpenStack

Gerritのイベントをトリガーとして、テストジョブを実行します。
テスト環境は、Ansibleを利用して動的にOpenStackなどのクラウド基盤上に構築します。




# ツールチェーン

本日ご紹介するデモは、以下のような流れです。

(1) プロジェクトをソースコードリポジトリからチェックアウト ...
**(Git, Gerrit)**

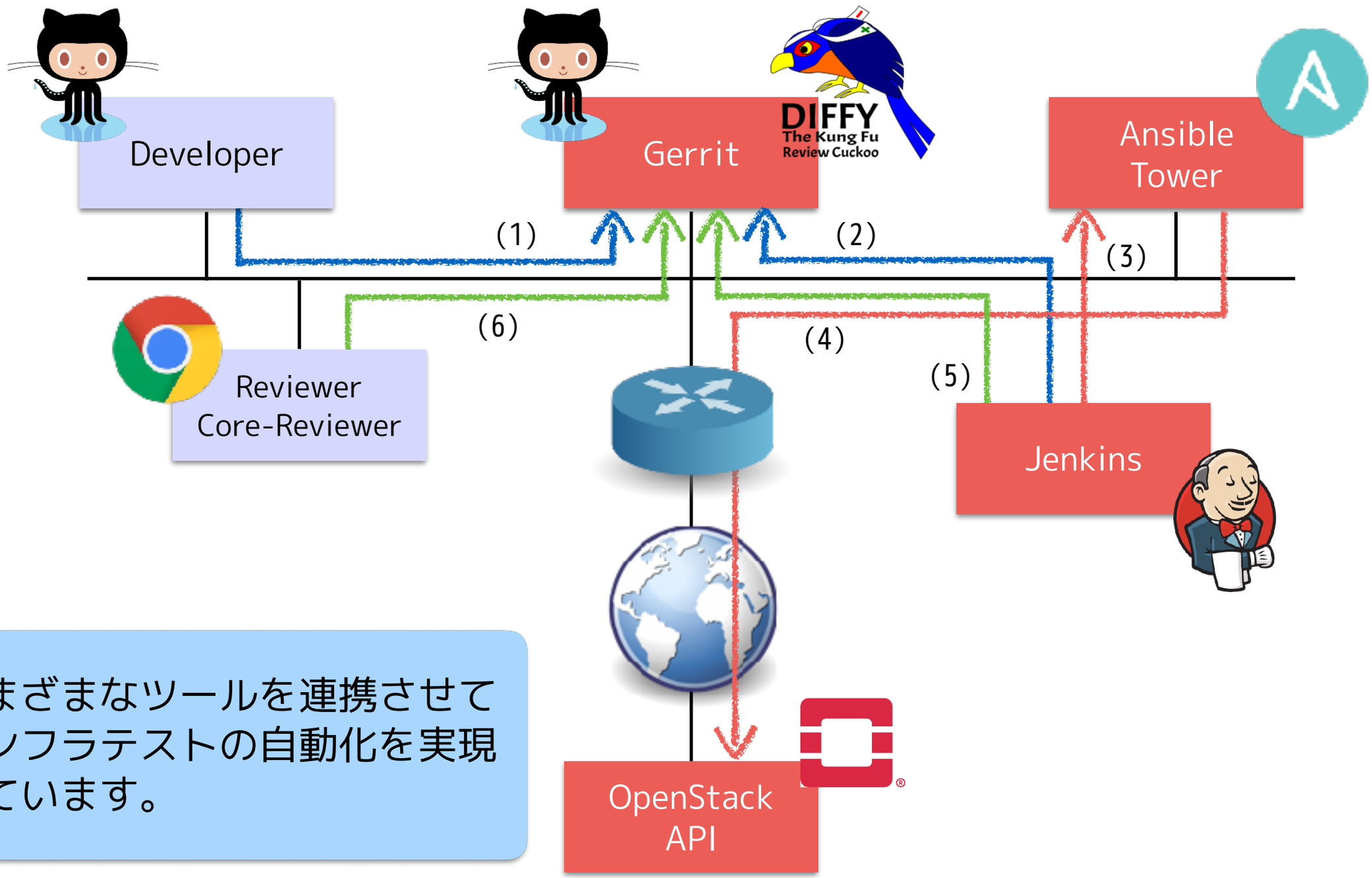
(2) ソースコードを改変 ... **(Git,Gerrit)**

(3) Gerritにレビュー登録 ... **(Git,Gerrit)**

(4) ツールチェーンによりテスト環境をデプロイ ...
**(Gerrit,Jenkins,Ansible,Ansible Tower)**

(5) レビューとテストを完了して改変コードを承認する ...
**(Git,Gerrit)**

# デモ




さまざまなツールを連携させて
インフラテストの自動化を実現
しています。

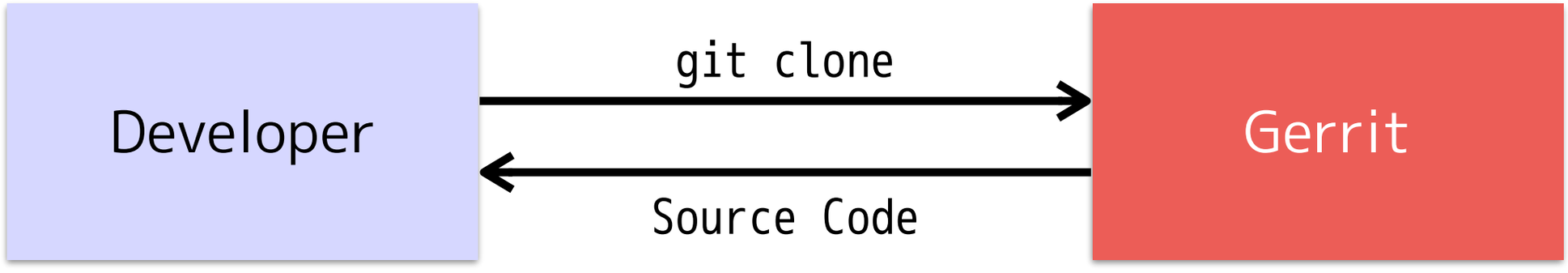
# コードのチェックアウト

## Playbookをチェックアウトします(1)

- サンプルプロジェクト(sandbox)

```
$ git clone http://<gerrit_host:gerrit_port>/sandbox.git
$ cd sandbox
$ git remote add gerrit http://<gerrit_host>:<gerrit_port>/sandbox.git
$ git config --local user.name "Hideki Saito"
$ git config --global user.email "foo@example.com"
```

Gerritのリポジトリを設定する

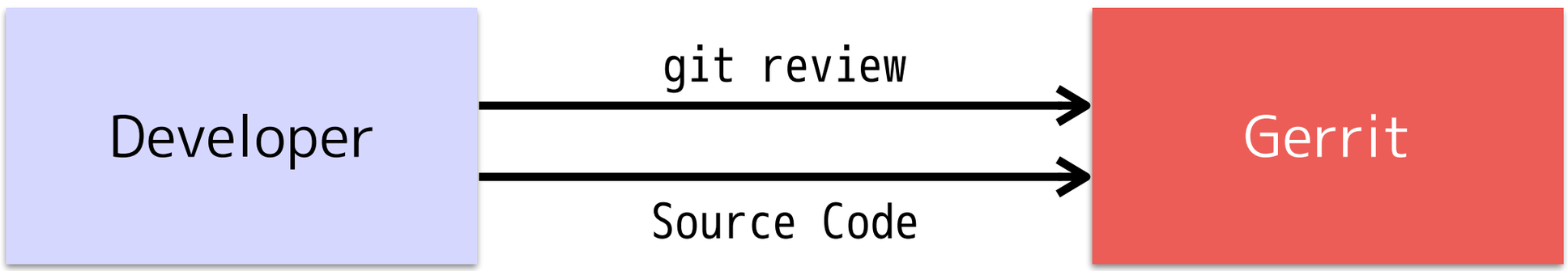


# レビュー申請

## Playbookを改変してレビューを申請します(1)

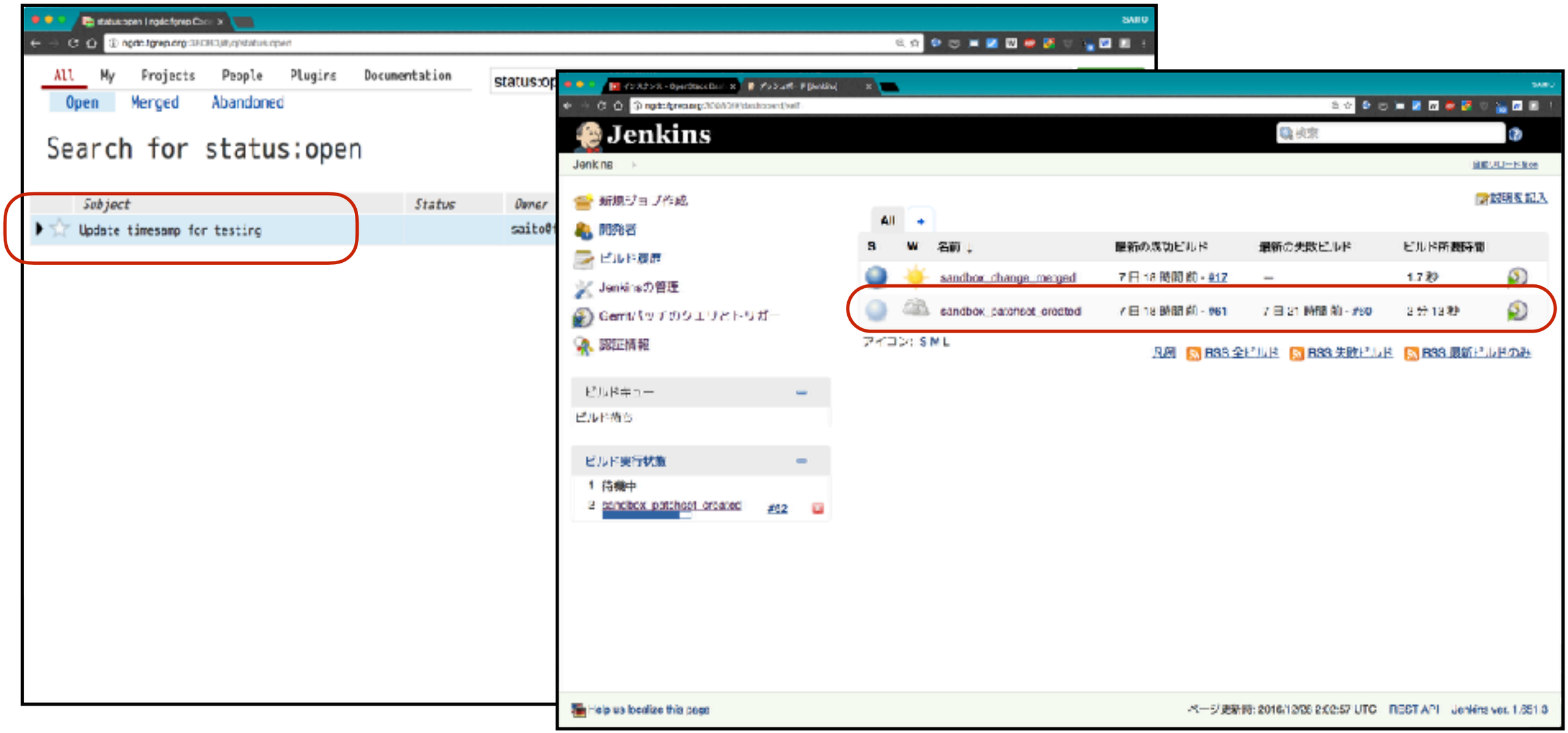
```
$ vi config_switch/site.yml

<... コード改変 ...>

$ git commit -a
$ git-review
```

Gerritへのpushとレビュー登録を行う



# Gerrit->Jenkins->Tower->OpenStack

Gerrit上にコードレビューが登録されると、Jenkinsに登録されているジョブが発動(2)して、Ansible Towerが自動テスト用のPlaybookを実行(3,4)します。

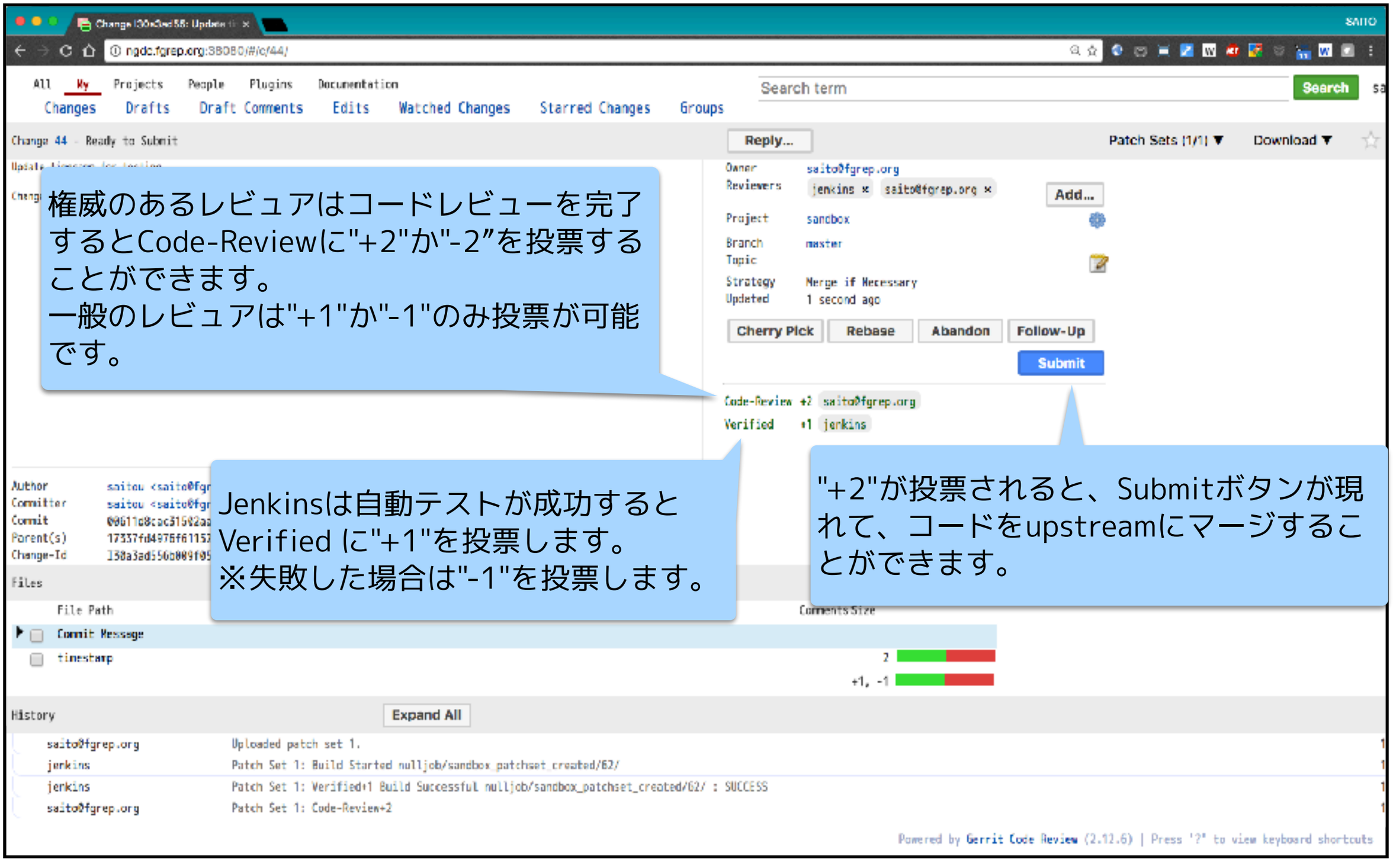


# Gerrit->Jenkins->Tower->OpenStack

Ansible TowerとOpenStackが連携して、テスト用の仮想マシンインスタンスが起動します。

```
$ openstack --os-cloud=poc server list -c ID -c Name -c Status -c Networks
+--------------------------------------+---------+--------+-------------------------------------+
| ID                                   | Name    | Status | Networks                            |
+--------------------------------------+---------+--------+-------------------------------------+
| c882daad-0106-4615-8d89-339036838604 | cumulus | ACTIVE | poc_backend=192.168.0.58, 172.16.165.136 |
| 75c5f76a-4571-42b6-89ea-92c82eaa16b3 | jenkins | ACTIVE | poc_backend=192.168.0.19, 172.16.165.134 |
| 4548da5a-670f-458e-95fa-b47d904d3d27 | gerrit  | ACTIVE | poc_backend=192.168.0.17, 172.16.165.133 |
| 6704e65e-a40f-432f-90ec-ad7b28d4d0f3 | tower   | ACTIVE | poc_backend=192.168.0.15, 172.16.165.132 |
+--------------------------------------+---------+--------+-------------------------------------+
```

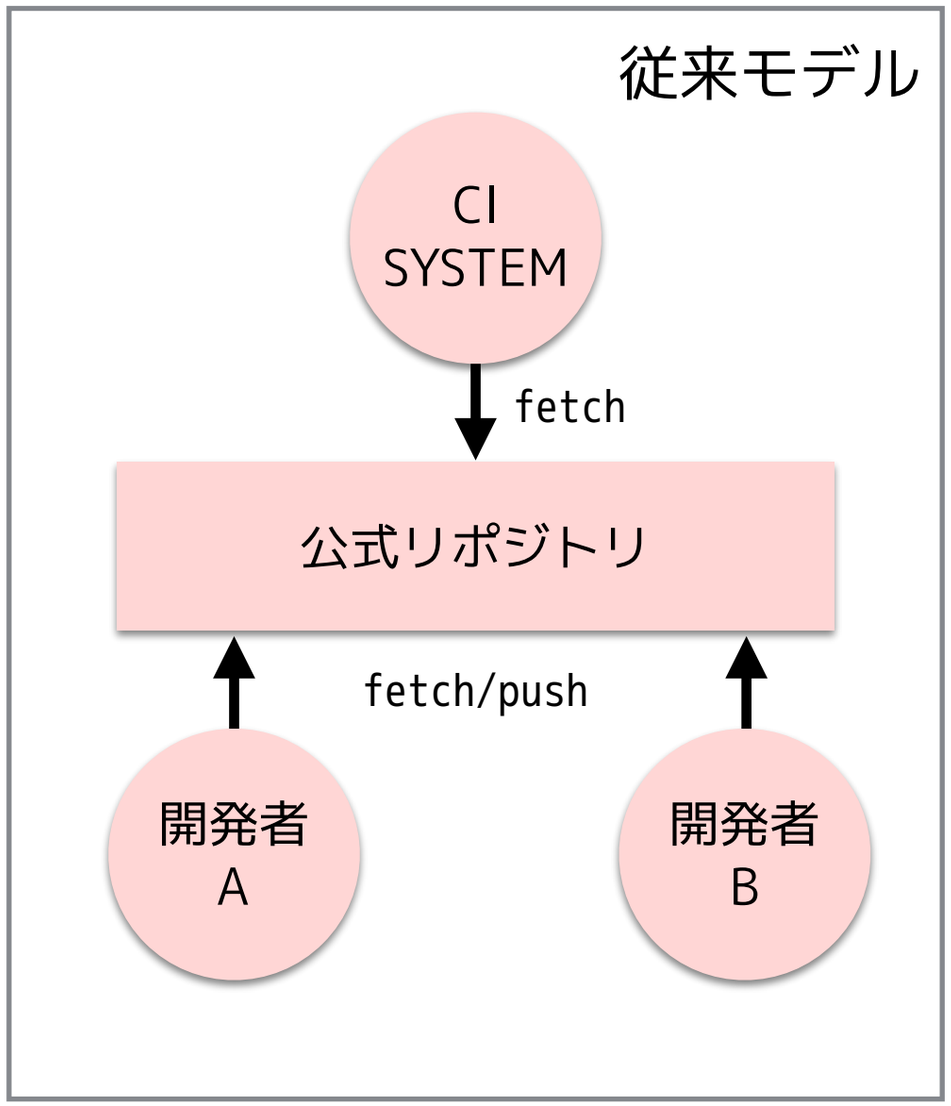
# 改変したコードの承認

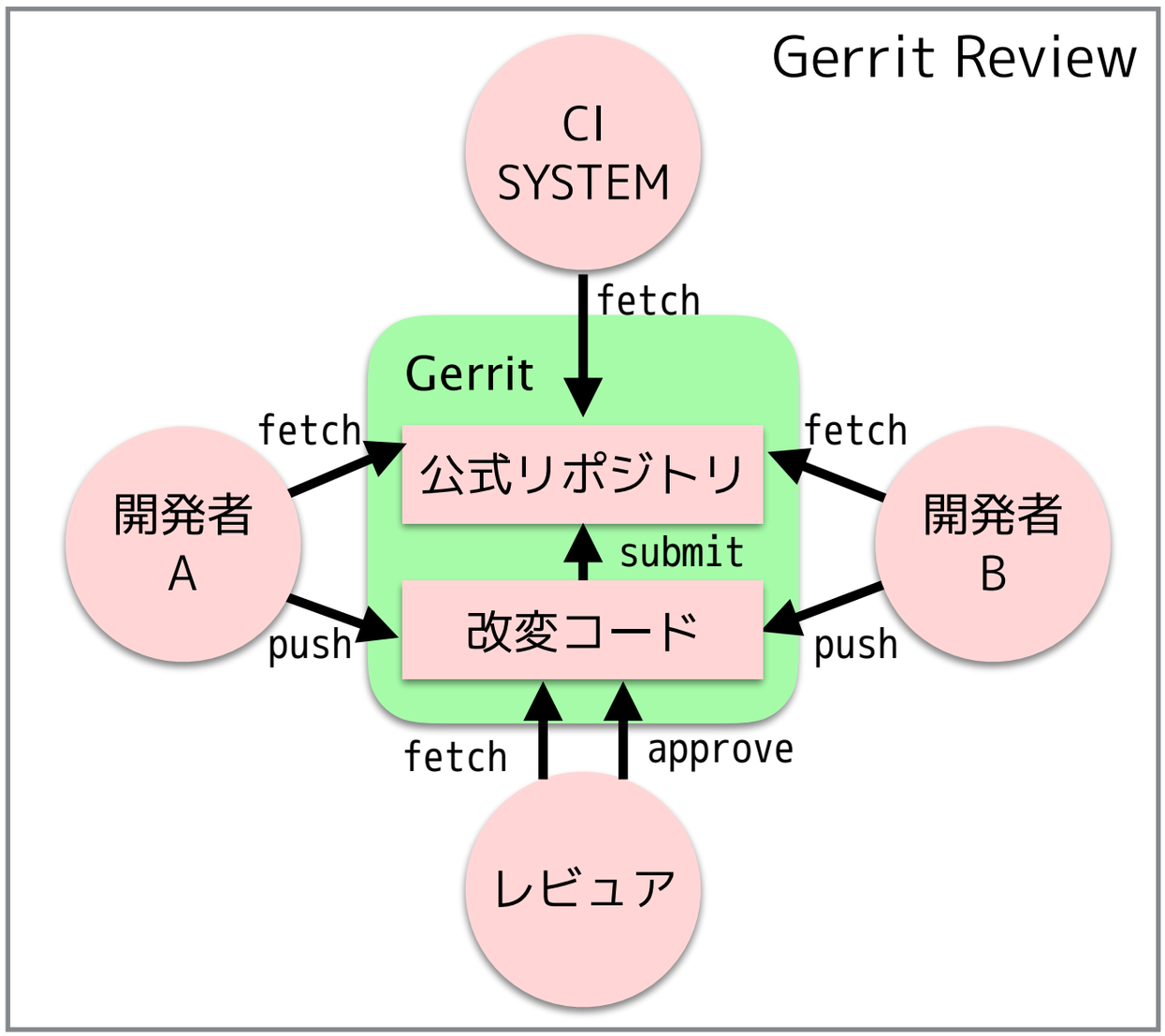

権威のあるレビュアはコードレビューを完了するとCode-Reviewに"+2"か"-2”を投票することができます。
一般のレビュアは"+1"か"-1"のみ投票が可能です。
Jenkinsは自動テストが成功すると
Verified に"+1"を投票します。
※失敗した場合は"-1"を投票します。
"+2"が投票されると、Submitボタンが現れて、コードをupstreamにマージすることができます。
All
My
Projects
People
Plugins
Documentation
Changes
Drafts
Draft Comments
Edits
Watched Changes
Starred Changes
Groups
Search term
Search
Reply...
Patch Sets [1/1]
Download
Owner
Reviewers
Add...
Project
sandbox
Branch
master
Topic
Strategy
Merge if Necessary
Updated
1 second ago
Cherry Pick
Rebase
Abandon
Follow-Up
Submit
Code-Review +2
Verified +1
jenkins
Author
Committer
Commit
Parent(s)
Change-Id
Files
File Path
Comments
Size
Commit Message
timestamp
History
Expand All
Uploaded patch set 1.
Patch Set 1: Build Started nulljob/sandbox_patchset_created/62/
Patch Set 1: Verified+1 Build Successful nulljob/sandbox_patchset_created/62/ : SUCCESS
Patch Set 1: Code-Review+2
Powered by Gerrit Code Review (2.12.6) | Press '?' to view keyboard shortcuts

# GerritとAnsibleについて少しだけ

# Gerrit

Gitの仕組みを上手くラップして、コードのレビューを
手助けしてくれる仕組みです(Gitの知識が必須です)。



# Gerrit関連情報

- 公式サイト
  - https://www.gerritcodereview.com/
- 公式ドキュメント
  - https://goo.gl/ZIKNX9
- Jenkins連携
  - https://goo.gl/E1TKoc
- 実例: OpenStackのコードレビューサイト
  - https://review.openstack.org

# Ansible

Infrastructure as Codeというコンセプトを実現するための自動化ツールで、以下の特徴を持っています。

(1) システム管理にエージェントを必要としない

(2) 構成管理データベースを持たない

(3) 多くの機能を提供する充実したモジュール群を提供

(4) 公式モジュールでは冪等性が担保される

(5) Playbookにワークフローの実現

(6) 数多い利用実績

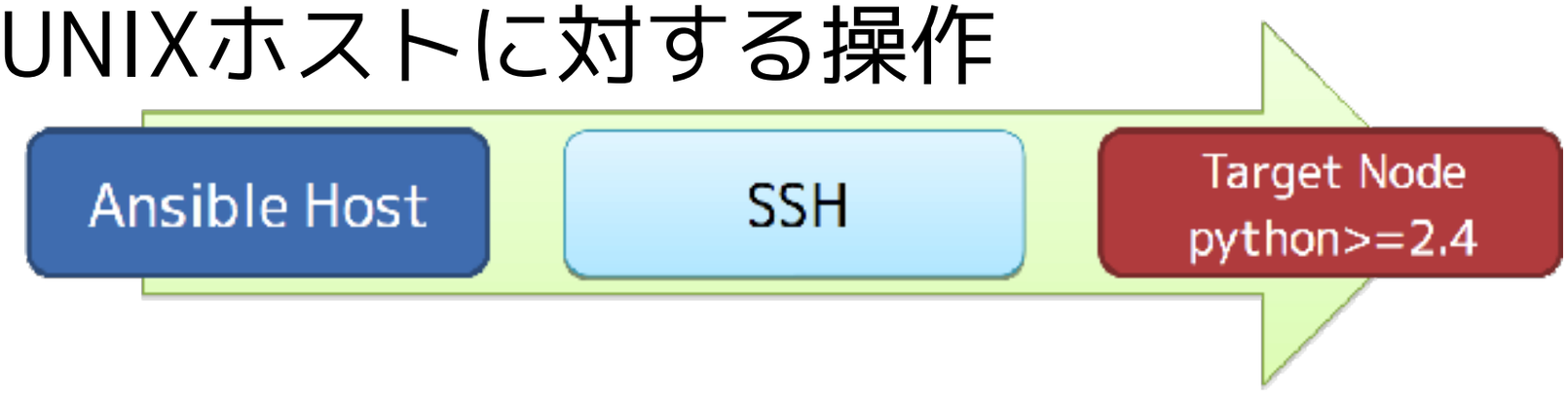
# Ansibleの構成要素

Ansibleの主な構成要素は以下の通りです。

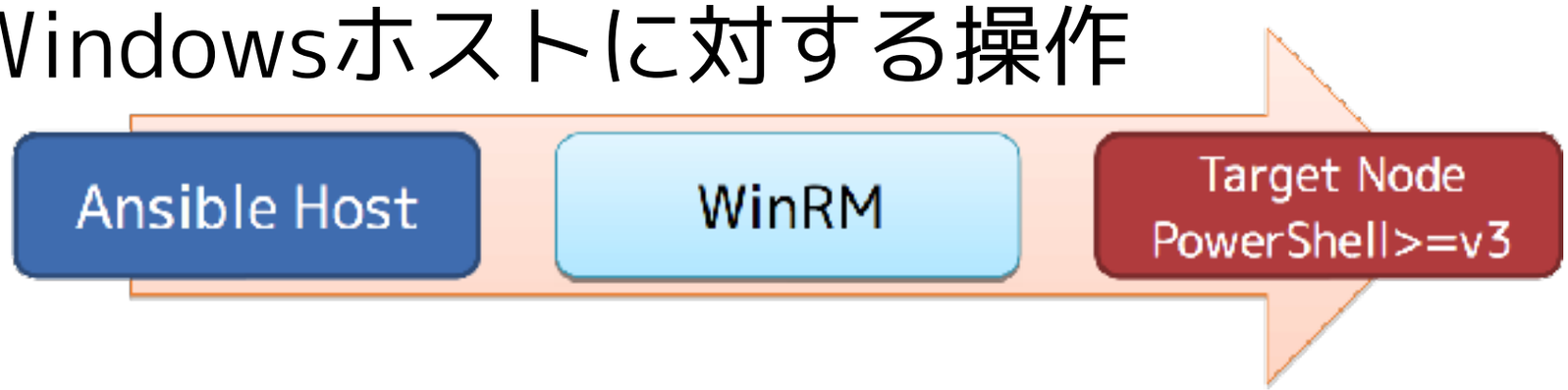
| # | 構成要素 | 概要 |
|---|---|---|
| 1 | Configuration | Ansibleの振る舞いを決める設定ファイル |
| 2 | Inventory | 管理対象ホストの一覧が記述されたファイル |
| 3 | Module | タスクとしてAnsibleが実行するプログラム |
| 4 | Command | タスクやPlaybookを実行するためのコマンド群 |
| 5 | Playbook | 複数のタスクから構成されるワークフロー定義ファイル |

# Ansibleによる自動化の仕組み

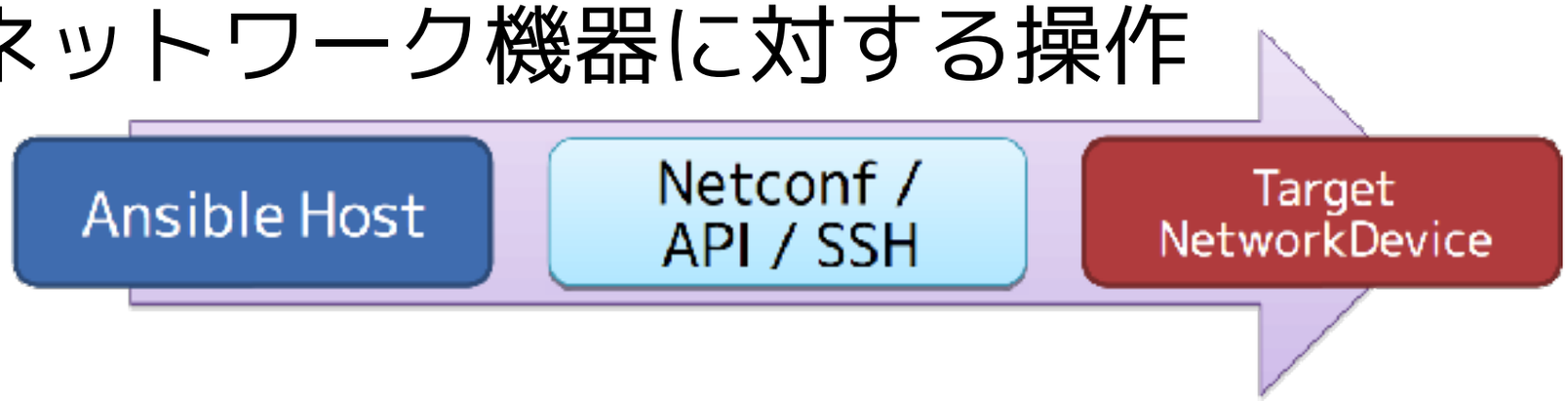
(a) UNIXホストに対する操作



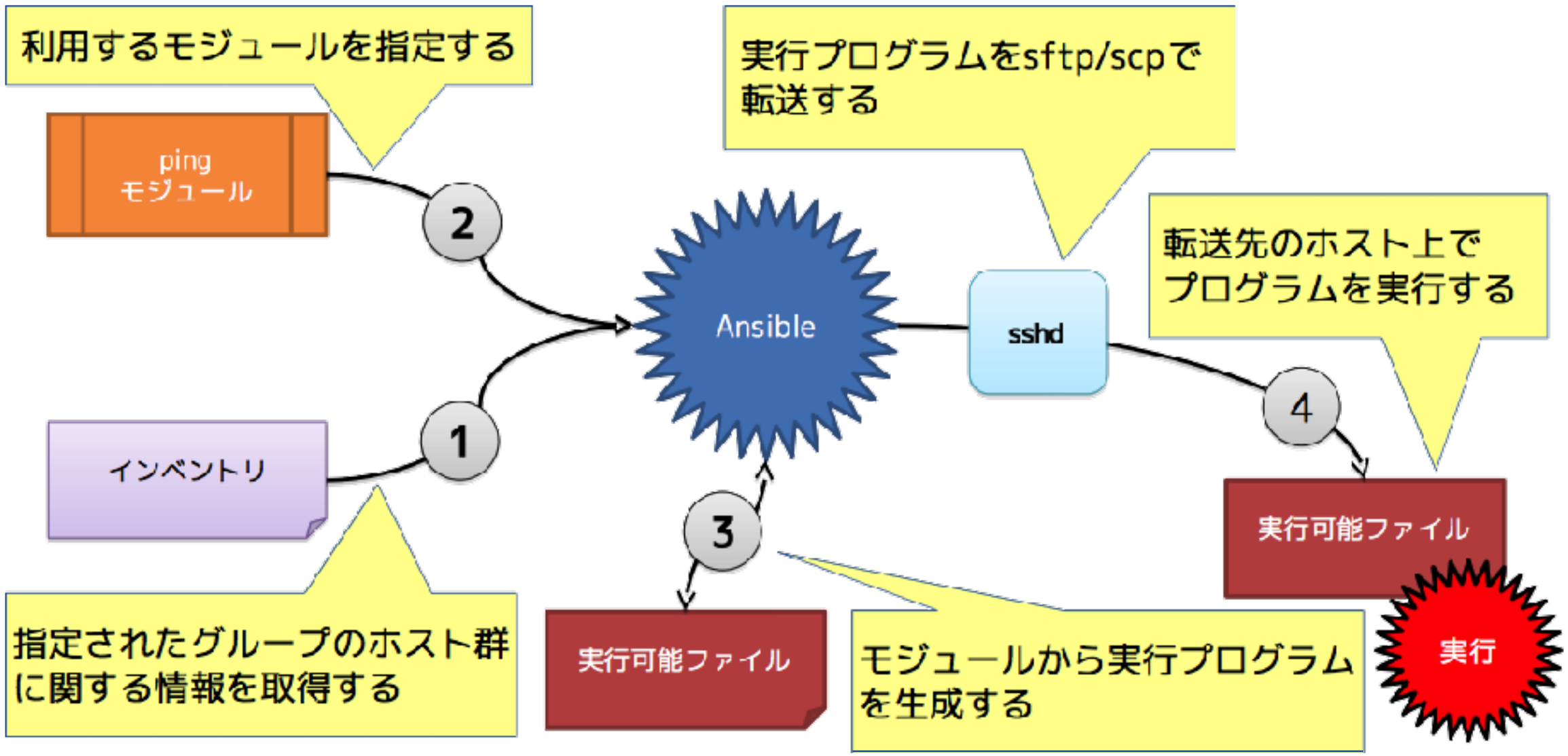

(b)Windowsホストに対する操作




(c)ネットワーク機器に対する操作

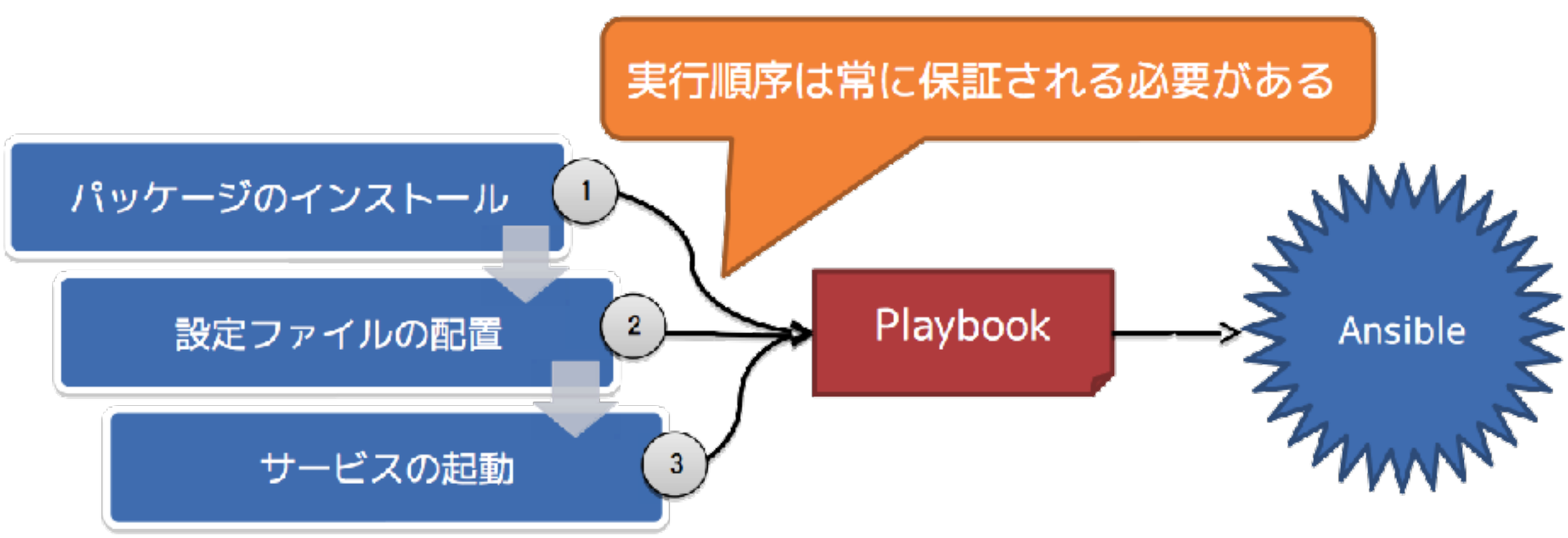


# ansibleコマンドによるタスク実行



# ansible-playbookコマンドによるシナリオ実行

AnsibleはPlaybookにより、オペレーションのコード化を実現しています。
Playbookは、実施する一連のタスクを、順序立ててYAML形式で記述したものです。



# Ansible Tower

Towerは、Playbookを利用したジョブ管理システムです。
本来はCLIから利用するPlaybookを、ジョブ管理システムのジョブの１つとして利用することができます。
今回のデモでは機能の制限されたトライアル版を利用しています(OSS化に向けて準備を進めているそうです)

**[Towerが拡張するAnsibleの機能]**

1. ジョブ (Playbook)単位でユーザに実行権限を付与
2. ジョブのAPIエンドポイント・Callbackエンドポイントを提供
3. オペレータがより安全に利用できるCLIを利用可能
4. 実行時のログを集約して管理可能
5. gitを利用したソースコード管理システムとの連携が可能

# Ansible関連情報

- 公式サイト
  - https://www.ansible.com/
- Tower公式サイト
  - https://www.ansible.com/tower
- 公式ドキュメント
  - http://docs.ansible.com/ansible/index.html
- ソースコードリポジトリ
  - https://github.com/ansible/ansible
  - https://github.com/ansible/ansible-modules-core
  - https://github.com/ansible/ansible-modules-extras

# Ansible関連情報

- 公式サイト
  - https://www.ansible.com/
- Tower公式サイト
  - https://www.ansible.com/tower
- 公式ドキュメント
  - http://docs.ansible.com/ansible/index.html
- ソースコードリポジトリ
  - https://github.com/ansible/ansible
  - https://github.com/ansible/ansible-modules-core
  - https://github.com/ansible/ansible-modules-extras

# まとめ

本セッションでは、Infrastructure as Codeによって実現するべきものは何なのか？、そして、それを実現するために利用できるツール群と、その組み合わせ手法(ツールチェーン)についてご紹介しました。

今回、例として紹介した実現手法は、そのままではないにせよ、OpenStackのようなOSSの開発にも利用されています。

本セッションの内容が、みなさんのお役にたてば幸いです。