Ansibleことはじめ

Ansibleって何？

構成管理ツール

なぜAnsible？

Ansibleでどうやって構成書くの？

PlayBookに書ける基本要素

モジュール

Role

Roleの構成

Roleの変数

ハンドラー

Taskで利用できる制御構造

Ansibleは構成管理ツールです！

構成管理ツールって何？

コードでインフラ/ミドルウェア構成を記述できるツール

作業効率UP！

間違えない/速い/深夜実行とかもできる

サービス/ツール間の連携

- バージョン管理: Git
- 自動テスト: Serverspec
- CI: CircleCI
- 監視: Zabbix
- チャット: Slack

なぜAnsible？

競合にChefなどあるが...

- エージェントレス： 管理マシンの制限が少ない（SSH接続さえできればよい）
- YAML： 誰が書いても似た記述（≒制約強い）。非プログラマでも読みやすい。

※ エージェント型ツールの場合、管理マシンに予めエージェントをインストールする必要がある（プロビジョニングのためのプロビジョニング）

Ansibleでどうやって構成書くの？

最低限必要↓↓

Inventory

デプロイ対象のホストを定義

PlayBook

デプロイ内容を定義

PlayBookに書ける超基本要素

Play

所属するTask全体への設定（ホストのグループなど指定）

Task

モジュールを指定してなにがしかの処理実行

モジュール

```
tasks:
  - name: nginxインストール
    apt:
      name: nginx
```

モジュール（この場合は apt ）にパラメータを渡して様々な処理を実行

組み込みモジュールが無数にあるので、基本的にこれらを組み合わせてタスクを作ればよい（自作もできる）

Role

構成が複雑化 → PlayBookが秘伝タレ化..

Role＝PlayBookのモジュール化

PlayBookから複数のRoleを組み合わせて利用

複数のプロジェクトで共有可能！

（変数がRole間でかぶらない工夫が必要）

標準構成を使えば読みやすい、書きやすい！

Roleの構成

```
├── myweb.yml
└── roles
    └── nginx
        ├── defaults
        │   └── main.yml
        ├── files
        │   └── web.ini
        ├── handlers
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── index.html
        └── vars
            └── main.yml
```

Role用変数

vars/main.yml

```
---
nginx_service_name: nginx
```

tasks/main.yml

```
---
- name: 起動/自動起動設定
  service:
    name: "{{ nginx_service_name }}"
    state: started
    enables: true
```

ハンドラー

tasks/main.yml

```
- name: nginx.conf展開
  template
    src: nginx.fonf.j2
    dest: /etc/nginx/nginx.conf
  vars:
    nginx_default_port: 80
  notify:
    - リロード
```

handlers/main.yml

```
- name: リロード
  service:
    name: nginx
    state: reloaded
```

タスクによって構成に変化があった場合のみ、 notify で指定されているハンドラーが実行される

Taskで利用できる制御構造

シンプルなループ： with_items

```
- name: ユーザ作成
  user:
    name: "{{ item.name }}"
    uid: "{{ item.uid }}"
  with_items:
    - { name: ichiro, uid: 1001 }
    - { name: jiro, uid: 1002 }
    - { name: saburo, uid: 1003 }
```

複数タスクのまとめ：block

```
- block:
  - name: エラーが発生するかもしれないタスク
    command: /user/local/bin/maybe-error
  - name: サービス起動
    service:
      name: some-service
      state: started
  rescue:
    - name: 切り戻し処理
      command: /user/local/bin/clean-up
    - name: デプロイ失敗
      fail:
  always:
    - name: hoge
      debug:
        msg: hoge
```

ブロック単位で処理制御（実行有無、変数定義）

例外処理

Inventry指定での変数

```
├── hosts
├── group_vars
│   ├── all.yml
│   ├── web.yml
│   └── db.yml
└── host_vars
    ├── web1.host.yml
    ├── web2.host.yml
    └── db.host.yml
```

グループごと、ホストごとに変数定義

変数の優先順位

基本は「狭いブロックが優先！」

- エクストラ変数（コマンドラインで指定）
- ...
- タスク変数
- ...
- Roleの vars で定義した変数
- ...
- Inventryの host_vars で定義した変数
- Inventryの group_vars で定義した変数
- ...
- Roleの defaults で定義した変数

VagrantのプロビジョニングはPlayBookで書ける

```
Vagrant.configure(2) do |config|
  config.vm.box = 'ubuntu/trusty64'

  ...

  config.vm.provision 'ansible' do |ansible|
    ansible.playbook = 'playbook.yml'
  end
end
```