Infrastructure as Code

-Infrastructure as Codeとは

インフラの構成管理を宣言的なコードで記述し，**ソフトウェア開発のプラクティスを適応すること**

-Infrastructure as Codeのプラクティス

- バージョン管理（Git）
- Pull request（レビュー）
- TDI（TDD）（テスト駆動インフラストラクチャ）
- CI（継続的インテグレーション）
- CD（継続的デプロイ）

Ansible勉強＆実践

Ansibleの特徴

- Python製
- 非エージェント型の構成管理ツール
- プラットフォームを問わずに利用できる
- yml，jsonで構成定義ファイル（Playbook）を書けるため，学習コストが低い
- Red Hat社がメインで開発しているOSS
- NASA，NEC，HP，Juniper，CISCO，EA，CocaColaなど名だたる大企業が使っている

Ansibleの構成要素

- Ansible本体
- Inventory $\cdots$ 捜査対象のマシン（ホスト）の管理ファイル
- Module $\cdots$ 操作対象のマシンを操作する
    - コアモジュールと有志モジュールが多数
    - 自作モジュールも簡単に作れる
- Playbook

YAMLファイル

YAMLファイルの構成

- 先頭行は `---` で始めること
- インデントは半角スペース2つ
- コメントは `#`

```
---
hoge
  fuga
```

Playbook

基本構成

```
- name: play book
  hosts: all # リモートホスト(インベントリのホスト or グループ)
  tasks:
    # tasksの引数
    - name: <task name>
      <module name>:
        # moduleの引数
        <module arg1>: <arg1 value>
        <module arg2>: <arg2 value>
```

実行方法

```
> ansible-playbook -v path/to/playbook.yml
```

※ `-v` で実行内容の表示

タスクの基本構成

```
~ 中略 ~
  tasks:
    - name: <task name>
      <module name>:
        <module arg1>: <arg1 value>
        <module arg2>: <arg2 value>
```

例1)

```
tasks:
  - name: install nginx
    yml:
      name: nginx
      state: present
```

または（簡略化する書式）

```
tasks:
  - name: install nginx
    yml: name=nginx state=present
```

例2)

```
- hosts: all
  user: all
  tasks:
    - name: Install the package "libvirt-bin"
      apt:
        name: <package name>
        <module arg1>: <arg1 value>
        <module arg2>: <arg2 value>
```

管理しやすい構成にする

- 各タスクに `name` を付与することで実行履歴を追いやすくなる！
- `when` や `loop` などはタスク内のどこでも定義できる

```
# サンプルコードでよくある構成
- name: debug code
  debug:
    msg: "debug"
  when: item
  loop:
    - True
    - False
    - True
    - True

# 管理しやすいように変更してもOK
# 順番は問わず
- name: debug code
  when: item
  loop:
    - True
    - False
    - True
    - True
  debug:
    msg: "debug"
```

操作対象のホストから情報収集しない場合

```
gater_facts: false
```

管理者権限で実行

```
become: true

# v1.9未満
sudo: yes
```

ユーザ指定実行

```
become_user: xxxx

# v1.9未満
sudo_user: yes
```

apt module

パラメーター一覧

Moduel arg	values	説明
allow_unauthenticated	`no(default) / yes`	aptコマンドの `--allow-unauthenticated` オプション. 意味はパッケージを確認できない場合に無視し，それについて質問しない.
autoclean	`no(default) / yes`	取得したパッケージのローカルリポジトリを掃除する？
autoremove	`no(default) / yes`	
cache_valid_time	`0(default)`	最新のrepositoryを維持したい！けどパッケージインストール毎にアップデートしたくない！って時に使いそう
deb	`< debian package path >`	ネットワーク上もしくはローカルに指定のdebファイルを指定してインストールする
default_release	`< distribution >`	aptコマンドの `-t` オプションに相当 特定のディストリビューション(trusty, xenial等)からパッケージを検索させてインストールすることができる 指定方法は以下のようにディストリビューション名を指定する `default_release: xenial`

Moduel arg	values	説明
dpkg_options	`force-confdef,force-confold(default)`	aptコマンドの `-o` オプションに相当 デフォルトで `force-confdef, force-confold` が付けられている オプションはコンマ区切りのリストとして指定する必要がある
force	`no(default) / yes`	aptコマンドの `--force-yes` に相当 どのような処理であってもプロンプトを発生させず非対話的に処理が進められる
force_apt_get	`no(default) / yes`	パッケージインストール時に `apt` ではなく `apt-get` を用いる
install_recommends	`no / yes`	aptコマンドの `--no-install-recommends` に相当 ざっくり言うと，推奨パッケージも一緒にインストールするかしないかを選択できる
only_upgrade	`no(default) / yes`	aptコマンドの `--only-upgrade`に相当 パッケージがすでにインストールされている場合のみアップグレードする `state: latest`との併用が必要
purge	`no(default) / yes`	
state	present(default) absent build-dep latest	パッケージ操作後の状態を規定する present:パッケージが既にインストールされている absent: アンインストール時に指定する（autoremove併用はお好みで） build-dep: `apt build-dep` に相当．ソースファイルからインストールする latest: 最新版となっている（最新版をインストールする）
update_cache	`no(default) / yes`	
upgrade	no(default) dist full safe yes	`apt upgrade`に相当．オプションは以下の通り． -no: 何もしない -dist: `dist-upgrade`（インストールされているカーネルの最新（ubuntu）/ディストリビューションの更新） -full: `full-upgrade` 前述以外もupgrade（パッケージを削除しないと更新できないパッケージも処理） -yes or safe: safe-upgrade（パッケージの構成を替えない範囲でアップグレードする）

aptを使ったパッケージのインストール

- install 1 package

```
tasks:
- name: Install the package "libvirt-bin"
  apt:
    name: libvirt-bin
```

- install select multi packages

どこかのvirsionからかwith_itemsが使えなくなったらしい

```
tasks:
- name: Install multi packages
  apt:
    name:
      - vim-gnome
      - vlc
      - gimp
```

- install select version

複数パッケージ&バージョン指定したい場合は，前述と組み合わせでいける

```
tasks:
- name: Install the package "libvirt-bin" selected version
  apt:
    name: libvirt-bin=1.3.1-1ubuntu10.25
```

-install use debfile

```
tasks:
- name: Install vlc use DebFile
  apt:
    deb: /var/cache/apt/archives/vlc_2.2.2-5ubuntu0.16.04.4_amd64.deb
```

aptを使ったパッケージのアンインストール

-remove（単なるアンインストール）

この場合，confファイルが残る（dpkg上，「rc」扱いになる）

```
tasks:
- name: Remove the package "libvirt-bin"
  apt:
```

```
    name: libvirt-bin
    state: absent
```

-purge(conf含めてアンインストール)

configも含めて削除（`apt purge`相当）の場合は「purge yes」を追加する これだと指定したパッケージの依存関係パッケージは削除されない

```
tasks:
- name: Purge the package "libvirt-bin"
  apt:
    name: libvirt-bin
    state: absent
    purge: yes
```

-autoremove & purge（--auto-removeオプション）

```
tasks:
- name: Autoremove the package "libvirt-bin"
  apt:
    name: libvirt-bin
    state: absent
    purge: yes
    autoremove: yes
```

apt Update/Upgrade

-apt update

```
tasks:
- name: apt update
  apt:
    update_cache: yes
```

-apt upgrade

```
tasks:
- name: apt upgrade
  apt:
    upgrade: yes
```

アトリビュート

-リトライ処理（retries）

Webサーバの立ち上げ＆死活確認する場合は，retriesを用いる untilの条件になるまで指定されたretries数を実行する 実行間隔はdelay

```
- name: wait app is available
  uri:
    url: "https://qiita.com"
    method: "GET"
  register: _response_result
  until: _response_result.status == 200
  retries: 5
  delay: 30
```

-出力ログにパスワードを非表示にする（no_log）

```
- name: secret task
  shell: /usr/bin/do_something --value={{ secret_value }}
  no_log: True
```

-ファイル操作（copy）

```
- name: deploy web contents
  copy:
    src: ./www/site-a/index.html
    dest: /usr/share/nginx/html/site-a/index.html
```

-コピー先のファイルを上書きしないようにする

```
  copy:
    force: false
```

-コピー先のファイルを上書きせずに古いファイルをBackUpする

```
copy:
    backup: true
```

※ etc/xxx.d/のように自動でファイル読み込みする場合は，古いファイルも読み込まれるので要注意

任意コマンドを実行（command）

```
- name: make ssh key in tmp dir
  command: "/usr/bin/ssh-keygen -b 2048 -t rsa -N '' -f /tmp/new-id_rsa"
    args:
      creates: /tmp/new-id_rsa
- name: echo home env
  command: "echo {{ ansible_env.HOME | quote }}"
```

※パイプ，リダイレクト（<，>）が使えない（shellモジュールでは使えるが推奨されていない） ※$記号を使った環境変数を使えない ※変数を扱う場合は，quiteフィルタでサニタイズすること