Ansible｜サーバ構築テンプレート作成

Ansibleによる構成管理

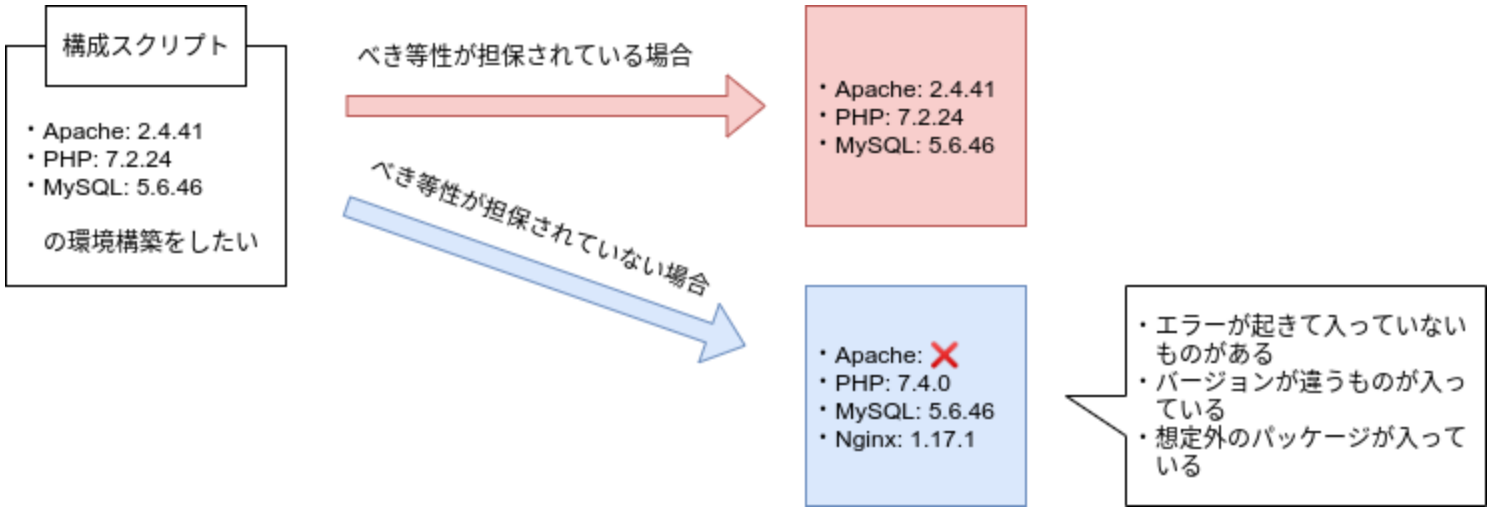
べき等性

べき等性とは、そのスクリプトを一回実行した結果と複数回実行した結果が変わらないことを示す

関数で言うところの参照透過性、副作用のない関数とほぼ同等の意味合いと考えて良い

構成管理においても、このべき等性が担保されていることが重要である

同じスクリプトを実行して、違う環境が構成されてしまっては構成管理ツールの意味がなくなってしまうからである



べき等性を担保するためには以下の点を意識してスクリプトを書くことが望ましい

- インストールするパッケージのバージョンを指定する
- スクリプトが実行された環境の情報を取得し、差分を処理する

Ansibleインストール

Ansibleはローカルマシンにインストールする必要がある

Dockerですぐに使える環境を作成したため、ここではそれを利用する

```
# GitHubからDockerCompose構成ファイルダウンロード
$ wget -O - https://github.com/amenoyoya/ansible/releases/download/0.1.0/docker-ansible.tar.gz | tar xzvf -

# プロジェクトディレクトリに移動
$ cd docker-ansible

# docker-compose run でコンテナ実行（--rm: 実行完了したらコンテナ削除）
# ※ 初回起動時のみイメージのPull＆Buildに時間かかる

## 例: ansible バージョン確認
$ docker-compose run --rm ansible --version

## 例: ansible-playbook 実行
$ docker-compose run --rm ansible-playbook -i inventoryfile playbookfile.yml
```

AnsibleによるCentOS7仮想サーバ構成管理

Ansibleを使ってみる

GitHubにAnsible学習用のプログラムを準備したため、そちらを参照

https://github.com/amenoyoya/ansible/tree/master/docker-centos7

サーバ構成のテンプレート化

CentOS 7 サーバ構築

仕事では CentOS 6, 7系のVPSを使うことが多いため、ここでは CentOS 7 を基本としてテンプレートを作成していく

なお、完成したPlaybookは https://github.com/amenoyoya/ansible/tree/master/docker-centos7/ansible/centos7-basic に置いてある

共通セキュリティ設定

- 参考: そこそこセキュアなlinuxサーバーを作る

とりあえず行わなければならない基本的なセキュリティ設定は以下の通り

1. services（サービス関連）
2. sshd（SSH接続関連）
3. iptables（Firewall関連）

services（サービス関連）

- **SELinux関連設定**
  - CentOS 7 など、モダンなOSでは、SELinuxという、Linuxのカーネルに強制アクセス制御 (MAC) 機能を付加するモジュールが有効化されている
  - 管理が複雑化したり、想定外の動作が起きたりすることもあるため無効化することも多いが、セキュリティ的にはなるべく有効化しておきたい
  - ここでは、SSH接続ポート変更のためのポリシーの設定のみ行う
- **pyenv導入**
  - CentOS7 のデフォルトのPythonは2系で、日本語環境との親和性が非常に悪い
  - そのため、複数バージョンのPython環境を管理できる pyenv を導入し、3系Pythonをインストールしておきたい

sshd（SSH接続関連）その1

- **ポート変更**
  - SSHのデフォルト接続ポート25番は一般に知れ渡っており標的の対象となりやすいため、別のポートに変更する
- **rootログイン不可**
  - rootはすべてのサーバにあるユーザであるため総当り攻撃される危険性がある
  - root権限はサーバ内のあらゆる操作が可能であるため乗っ取られると非常に危険
- **パスワード認証不可**
  - パスワード認証は総当り攻撃の対象になるため禁止する（公開鍵認証のみ許可とする）

sshd（SSH接続関連）その2

- **SSH接続できないユーザの作成**
  - Web制作をしていると、デザイナやコーダーなどがFTP（SFTP）でファイルアップロードしたいという要望があるため、SSH接続不可でSFTP接続のみ可能なユーザを作成しておくと便利
    - 参考: sshで接続したくないけどSFTPは使いたい時の設定
- **sshdプロトコルの設定**
  - sshdプロトコル1には脆弱性があるらしいので、2に設定する（最近のディストリビューションは最初から設定されているが念の為）
- **認証猶予時間と試行回数の制限**
  - 制限をきつくすると締め出されてしまう危険性もあるが、緩めに制限しておくと多少安心

iptables（ファイウォール関連）

- **外部公開ポートの制限**
  - 外部に公開されているポートが多いと、それだけ攻撃を受けやすくなる
  - そのため、最低限外部接続可能なポート（httpポート, httpsポート, ssl（sftp）ポート等）以外のポートを閉じておく
    - 参考: ファイアウォールiptablesを簡単解説
- **ポートスキャン対策**
  - ポートスキャンとは、どのポートが開いているか外部から調査する攻撃手法

ディレクトリ構成

ディレクトリ構成は、Ansibleのベストラクティスを参考にしつつ以下のような構成とした

- 参考: Best Practice - Ansible Documentation

なお、共通セキュリティ設定関連は、**management**グループとしてまとめることとした

```
./
|_ production.yml # 本番サーバ用インベントリファイル
|_ main.yml # メインPlaybook｜playbooks/***.yml を読み込んで実行
|_ group_vars/ # 変数定義ファイル格納ディレクトリ
|    |_ all.yml # 全グループ共通の変数定義ファイル
|    |_ management/ # 共通セキュリティ設定関連の変数定義ファイルの格納ディレクトリ
|    |    |_ settings.yml # 共通セキュリティ設定の変数定義ファイル
|    |    |_ ports.yml # ポート設定関連の変数定義ファイル
|    |    |_ users.yml # ユーザ管理はこのファイルで行う想定
|    |
|    |_ pyenv/ # Python環境整備関連の変数定義ファイルの格納ディレクトリ
|
|_ playbooks/ # 実際にサーバに対する操作を行うファイルを格納するディレクトリ｜インフラ管理者以外は触らない想定
|    |_ management.yml # 共通セキュリティ設定を行うPlaybook｜roles/management/tasks/main.yml のタスクを実行
|    |_ roles/ # Playbookで実行されるタスクを役割ごとに格納するディレクトリ
|         |_ management/ # このディレクトリ名（role）は親Playbookの名前と揃える
|         |    |_ tasks/  # 共通セキュリティ設定で実行するタスクを格納するディレクトリ
|         |    |_ templates/ # Jinja2テンプレートファイル格納ディレクトリ
|         |
|         |_ pyenv/ # Python環境整備タスク格納ディレクトリ
|
|_ ssh/ # ユーザごとの秘密鍵を格納するディレクトリ
        ## この部分の運用については考える必要があるかもしれない
```

運用方法にもよるが、基本的にグループ名とPlaybook名、Role名は統一したほうが分かりやすい

今回の場合、以下の名前はすべて `management` や `pyenv` で統一する

- インベントリファイルに記述するグループ名
- `group_vars` ディレクトリ配下の、グループ内変数ファイル格納ディレクトリ名
- `playbooks` ディレクトリ配下の、実行Playbookファイル名
- `playbooks/roles` ディレクトリ配下の、実行タスク格納ディレクトリ名（Role名）

production.yml

本番サーバ用のインベントリファイル

今回は `production.yml` のみ作成しているが、本来はステージングサーバや開発用サーバも用意していることがほとんどなので、`staging.yml`, `development.yml` 等のインベントリファイルも必要になるはず

インベントリファイルの基本的な記述内容は以下の通り

```
---
# 各グループ名は group_vars/***/, playbooks/***.yml と統一する
management: # 共通セキュリティ設定グループ
  hosts: # ホスト定義
    web: # ホスト名は基本的に web で統一する
      ansible_host: 172.17.8.100 # 指定サーバのIPアドレス
      ansible_ssh_port: 22 # SSH接続ポートは通常 22番
      ansible_ssh_user: root # SSH接続ユーザ名
      ansible_ssh_private_key_file: ~/.ssh/private_key # SSH秘密鍵
      ansible_sudo_pass: XXX # rootユーザパスワード
```

yamlファイル先頭の `---` はなくても動くが、慣習的につけることが多いようなのでつけている

group_vars/all.yml

全グループで共通して使用可能な変数を定義するためのファイル（このファイルはPlaybook実行時に自動的に読み込まれる）

ここでは、サーバ（マシン）再起動の待ち時間のみ定義している

```
---
# サーバ再起動待ちのタイムアウト時間 [秒]
## それなりに長い時間を指定する
reboot_wait_timeout: 300
```

main.yml

Playbook実行時のエントリーファイル

`playbooks` ディレクトリ内のRoleごとのPlaybookファイルをimportするだけ

```
---
# Python環境整備タスク
## 個人的にPythonの環境整備は最初にやっておきたい
- import_playbook: playbooks/pyenv.yml

# 共通セキュリティ基本設定
- import_playbook: playbooks/management.yml
```

playbooks/management.yml

共通セキュリティ設定を定義するPlaybookファイル

```
---
- hosts: web # ホスト名は基本的に web で統一する
  become: true # root権限で実行
  roles:
    - management # ./roles/management/tasks/main.yml を実行
```

playbooks/roles/management/tasks/main.yml

対応するrole名のPlaybookから呼び出される各種タスクを定義するファイル

このファイルではタスクを直接記述せず、関連タスクを別ファイルに記述して include するようにした方が運用しやすい

```
---
# SELinux関連設定
## includeモジュールで 別ファイルの中身をそのまま展開できる
- include: selinux.yml

# 管理者ユーザ設定
- include: admin_users.yml

# 一般ユーザ設定
- include: users.yml

# sshd, iptables設定
- include: services.yml
```

SELinux関連設定

SELinuxを使用するか無効化するかは慎重に検討する必要がある

CentOS 7 ではデフォルトで有効化されているため、ここでは有効化されたままの運用を想定している

group_vars/management/ports.yml

ポート関連の設定変数を定義している

本番運用の際は、SSH接続ポートをデフォルトの 22番ポートにしておくのは危険だが、設定を失敗するとサーバに対する設定が一切できなくなってしまうため、開発段階では 22番ポートも設定しておくと安全

yamlでは、先頭に `-` をつけることで配列を表現することができるため、有効活用すると良い

- 参考: YAML Syntax - Ansible Documentation

```yaml
---
# SSH接続ポート
## 推測されにくく、他のアプリケーションポートと重複しないポートを指定する
## 開発段階では、22番を含む複数ポートを指定しておくと安全
ssh_ports:
  - 22
  - 1022

# 外部からのアクセスを許可するポート
accept_ports:
  # httpポート
  - port: 80
    protocol: 'tcp'

  # httpsポート
  - port: 443
    protocol: 'tcp'
```

playbooks/roles/management/tasks/selinux.yml

SELinux関連の設定を行うタスクを定義している

```
---
- name: AnsibleのSELinux関連モジュールを使うためのパッケージをインストール
  # yumモジュール｜name=<パッケージ名（リスト指定可）> state=<present|absent> ...
  yum:
    name:
      - libselinux-python
      - policycoreutils-python
    state: present

- name: SELinux｜sshポート開放
  # seportモジュール｜ports=<ポート番号> proto=<tcp|udp> setype=<ポートのSELinuxタイプ> state=<present|absent> ...
  seport:
    # with_items: '配列' で配列を処理できる｜foreach item in 配列
    ## {{変数名}} で変数の展開が可能
    ports: '{{ item }}'
    setype: ssh_port_t
    proto: tcp
    state: present
    reload: yes # 設定反映のためにSELinux再起動
  with_items: '{{ ssh_ports }}'
```

ユーザ管理と sshd, iptables 設定

続いて、運用に工夫が必要なユーザ管理と sshd, iptables の設定を行う

- 参考: Ansible Playbookでユーザ管理（登録・削除）をまるっとやる

sshd, iptables, SELinux はポート制御関連で相互に関わり合っている部分も多いため、なるべく一緒に管理したほうが良い

group_vars/management/users.yml

ユーザの追加・削除をしたい場合は、この変数定義ファイルに設定を記述する

SSH接続してサーバ内の各種操作が可能なユーザは `admin_users`, FTP（SFTP）接続してファイルの更新だけが可能なユーザは `users` に設定する

各ユーザがサーバ接続するための鍵ファイルは `ssh` ディレクトリに生成される

```yaml
---
# ユーザ設定
## ユーザを追加・削除する場合はここを変更する

# 管理者ユーザ: sshログイン & sudo権限あり
admin_users:
  - name: 'centos-admin'
    uid: 1001 # 重複しないユーザIDを指定すること

# 一般ユーザ: sftp接続のみ可
users:
  - name: 'centos-user'
    uid: 2001

# ---

# ユーザグループ設定
## 基本的に編集しない
admin_group: 'admin'
user_group: 'developers'
```

playbooks/roles/management/tasks/admin_users.yml

管理者ユーザ（SSH接続可・sudo権限あり）の追加・削除を行うタスクを定義している

```yaml
---
# (一部のみ抜粋)

- name: 管理者グループ作成
  # groupモジュール｜name=<グループ名> state=<present（作成）|absent（削除）>
  group: name={{ admin_group }} state=present

- name: 実際に管理者グループに属するユーザのリストを取得
  shell: 'getent group {{ admin_group }} | cut -d: -f4 | tr "," "\n"'
  register: present_sudoers # => シェルの実行結果を present_sudoers という変数に代入

- name: すでに存在しないユーザを削除
  # ユーザのホームディレクトリごと削除
  user: name={{ item }} state=absent remove=yes
  # 実際に管理者グループに属するユーザと admin_users に設定されたユーザの差分 => 削除するべきユーザ
  ## `[{key: val}, ...] | map(attribute="key") | list` で dictの指定キーのみを取り出してリスト化可能
  with_items: '{{ present_sudoers.stdout_lines | difference(admin_users | map(attribute="name") | list) }}'
  ignore_errors: yes # ユーザが削除済みの場合があるためエラーは無視

- name: 管理者ユーザの秘密鍵ダウンロード
  fetch: src=/home/{{ item.name }}/.ssh/id_rsa dest=../ssh/{{ item.name }}-id_rsa flat=yes
  with_items: '{{ admin_users }}'
```

playbooks/roles/management/tasks/users.yml

一般ユーザ（SFTP接続してファイルの更新のみ可）の追加・削除を行うタスクを定義している

`admin_users.yml` と似たような内容のため省略

playbooks/roles/management/tasks/services.yml

sshd と iptables の設定タスクを定義するが、注意点として、CentOS 7 以降は iptables の代わりに firewalld がデフォルトのアクセス制御サービスとなっているため、iptables を使うように設定する

今後は firewalld が主流になっていくと思われるが、現時点ではナレッジが十分に蓄積されていない

```yaml
---
- name: sshd設定
  # Jinja2テンプレートエンジンを利用してテンプレートファイルを展開してアップロード
  # templateモジュール｜src=<テンプレートファイル> dest=<アップロード先ファイルパス> ...
  ## テンプレートファイル内では {{変数名}} で変数を展開できる（詳しくは Jinja2 公式リファレンス参照）
  template: src=../templates/sshd_config.j2 dest=/etc/ssh/sshd_config owner=root group=root mode=0600

- name: sshd再起動＆スタートアップ登録
  # serviceモジュール｜name=<サービス名> state=<reloaded|restarted|started|stopped> enabled=<false|true> ...
  ## enabled=true にするとスタートアップサービスに登録することができる
  service: name=sshd state=restarted enabled=true

- name: iptables設定
  template: src=../templates/iptables.j2 dest=/etc/sysconfig/iptables owner=root group=root mode=0600

# CentOS 7 以降は iptables が入っていないことがあるためインストールする
- name: iptablesインストール
  # yumモジュール｜name=<パッケージ名> state=<present|absent> ...
  yum: name=iptables-services state=present

# CentOS 7 以降は iptables の代わりに firewalld がデフォルトになっているため停止する
- name: firewalld停止＆スタートアップ削除
  service: name=firewalld state=stopped enabled=false
  ignore_errors: yes # CentOS 6 なら firewalld は存在しないためエラーは無視

- name: iptables（再）起動＆スタートアップ登録
  service: name=iptables state=restarted enabled=true
```

playbooks/roles/management/templates/sshd_config.j2

sshd設定ファイルの内容をJinja2テンプレートを用いて記述している

Jinja2で使える表記法については、公式リファレンスを参照

```
#  （一部設定のみ抜粋）

# rootログイン不可
PermitRootLogin no

# パスワード認証不可｜公開鍵認証のみ許可
PasswordAuthentication no
ChallengeResponseAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys

# sftp で chroot させたい場合は internal-sftp を使う必要あり
Subsystem  sftp  internal-sftp

# sftpのみ許可するユーザグループの設定
## {{変数名}} で変数を展開可能
Match Group {{ user_group }}
    # ログインシェルに internal-sftp を強制
    ## => ssh接続はできず、sftp接続のみ可能になる
    ForceCommand internal-sftp
```

Playbook実行

設定できたら動作確認を行う

```
# Playbook実行
$ ansible-playbook -i production.yml main.yml
    :
management  : ok=22  changed=18  unreachable=0  failed=0  skipped=6  rescued=0  ignored=0

## => SELinuxを使う想定で設定しているため、SELinux無効化関連タスクはskipされる
## => ssh/centos-admin-id_rsa, centos-user-id_rsa が保存されるはず

# 作成された管理者ユーザでSSH接続確認
$ cp ssh/centos-admin-id_rsa ~/.ssh/centos-admin-id_rsa
$ chmod 600 ~/.ssh/centos-admin-id_rsa
$ ssh -i ~/.ssh/centos-admin-id_rsa centos-admin@172.17.8.100
```

```
## => 問題なく接続できたら、ポートの確認を行う

# iptables の開放ポートを確認
[centos-admin ~]$ sudo iptables -nL

# SELinuxを有効化している場合: SELinuxのポリシーを確認
## 普通に semanage を実行すると UnicodeEncodingError が起こるため PYTHONIOENCODING環境変数を設定しながら実行する
[centos-admin ~]$ sudo PYTHONIOENCODING=utf-8 python /usr/sbin/semanage port -l

## => iptables, SELinux の開放ポート（特にSSHポート）が想定通り設定されていればOK

# SSH切断
[centos-admin ~]$ exit
```

```
# 一般ユーザでSSH接続試行
$ cp ssh/centos-user-id_rsa ~/.ssh/centos-user-id_rsa
$ chmod 600 ~/.ssh/centos-user-id_rsa
$ ssh -i ~/.ssh/centos-user-id_rsa centos-user@172.17.8.100

## => This service allows sftp connections only.
### 上記のようなエラーが出て接続を拒否されればOK

# 一般ユーザでSFTP接続確認
$ sftp -i ssh/centos-user-id_rsa centos-user@172.17.8.100

---
## => 問題なく接続できたらOK
sftp> exit
```

なお、ここでは、全部の設定を行ってから動作確認をしているが、本来はもう少し細かい単位の設定ごとに動作確認をしていくほうが安全である