preseed と ansible によるサーバ構築について

常三島技術部門
情報システムグループ　　山中　卓也 (Takuya Yamanaka)

１．はじめに

これまで多くのサーバを構築・運用しているが，その作業は基本的に対話的に行うものであり，インストール作業を手順に従い１つ１つ設定したり，必要なソフトウェアをインストールするコマンドを入力したりする必要がある。しかし近年，管理するサーバ数の増大にともない，それらの構築・運用の手間をより減らすことが望まれる。そこで，preseed, ansible という自動化ツールを用いて，インストール作業を行った。

２．preseed

２．１　preseed による自動インストール

preseed は OS インストール作業を自動化するためのツールである。OS インストール作業をするときは，文字コードの設定，ネットワークの設定, HDD のパーティション分けの設定といったさまざまな内容を設定する必要がある。通常はこれらを対話的に，設定していくことになるが, preseed のような自動インストールツールでは，あらかじめ preseed.cfg というファイルにそれらの情報を記入しておくことで，各種設定を自動で行ってくれる。preseed は Debian，Ubuntu といった Linux 用のツールである。Redhat，CentOS といった Linux には, kickstart, FreeBSD では BSDinstall という同様のツールがある。

２．２　preseed.cfg

preseed.cfg は事前設定ファイルであり，ここにすべてのインストール時の情報を記述しておく必要がある[1]。
図１のような形式になっており, 必要に応じ，コメントをはずし，各設定行に自分の環境の値を設定していく。すべての設定の解説はできないため，次項以降に preseed の設定の中で特に注意が必要な点について書く。

```
#### Contents of the preconfiguration file (for jessie)
### Localization
# Preseeding only locale sets language, country and locale.
d-i debian-installer/locale string ja_JP.UTF-8
# Keyboard selection.
d-i keyboard-configuration/xkb-keymap select jp
# d-i keyboard-configuration/toggle select No toggling
                    :
                    :
```

図１　preseed.cfg の例

２．３　preseed.cfg の配置と取得

自動インストールを行うためには，インストールを始める前にサーバから設定を記述した preseed.cfg を読み込む必要がある。これにはいくつかの方法があるが, もっとも簡単な, 同一ネットワーク上の別サーバから設定ファイルを取り込む方法を用いた。この方法は，インストール開始前のメニューからオプションとして preseed.cfg の置き場所を指定する。

自動インストールメニューから"Advanced options"→"Automated install"を選び，Tab キー入力すると，入力可能状態となるので，末尾に” url=http://192.168.1.20/preseed.cfg”と入力する。インストール対象サーバと preseed.cfg を置いているサーバとのネットワーク構成は，以下図２に示した。

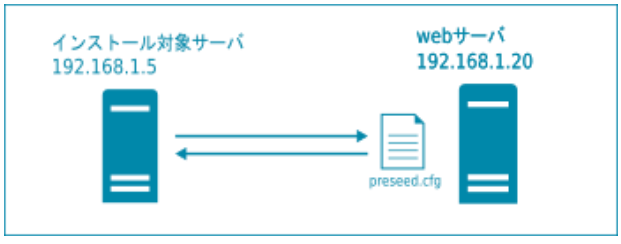

図２　preseed.cfg とインストール対象サーバの関係

２．４　静的 IP の設定

DHCP 設定の場合はそれほど問題とならないが, 静的 IP を設定する場合はいくつか工夫が必要となる。

下記図３に示すような preseed.cfg となり，"Network configuration"以下がネットワーク関係の設定である。これの preseed/run として netcfg.sh を指定しているところがポイントとなる(図４として netcfg.sh を示す)。

```
### Network configuration
d-i preseed/run string http://192.168.1.20/preseed/netcfg.sh
d-i netcfg/choose_interface select auto
d-i netcfg/disable_autoconfig boolean true
d-i netcfg/dhcp_failed note
d-i netcfg/dhcp_options select Configure network manually

# Static network configuration.
d-i netcfg/get_ipaddress string 192.168.1.5
d-i netcfg/get_netmask string 255.255.255.0
d-i netcfg/get_gateway string 192.168.1.1
d-i netcfg/get_nameservers string 192.168.1.10
d-i netcfg/confirm_static boolean true
```

図３　network 設定に関する preseed.cfg

この方法で，最初に DHCP により preseed.cfg ファイルを取得し，その後一度 netcfg を終了して，netcfg による設定を静的 IP で行うことができる。このことから同一ネットワーク内に DHCP サーバを用意しておく必要があることにも注意すること。

```
#!/bin/sh
killall.sh; netcfg
```

図４　netcfg.sh

２．５　HDD のパーティショニング

HDD は500G の HDD を表１のように分割した。

表１　HDD パーティション分割

partition	容量(GB)
/boot	0.256
/	125
/var	125
/home	残り(250程度)
swap	1

図５に partition 分割に関する preseed.cfg を示す。3つの数字がそれぞれ，最小サイズ，優先値，最大サイズを意味する。サイズの単位は MB である。優先値は低いものから優先する。最大サイズが-1のときは，残り全部を意味する。詳細は partman-auto-recipe.txt [2]を参照のこと。

２．６　実際の適用結果

上記 preseed による実際のインストールを行った。最初にまず仮想マシンなどを用いテストを行い，ある程度の preseed.cfg の確認を行ったのちに,数台の実マシンに適応させた。インストールするソフトウェアの量によるが，一台につき15から20分程度でインストールが完了した。最初に起動させてインストール画面から preseed.cfg の指定さえすればあとは自動なので，同時並行で数台インストールすることも可能である。また，思い通りに設定できなかったときにインストールをやり直す敷居も低くなったこと，しばらくのちに似たような構成の OS をインストールするときにも役に立った。

```
d-i partman-auto/expert_recipe string                   ¥
    boot-root ::                                        ¥
        40 300 256 ext4                                 ¥
            $primary{ } $bootable{ }                    ¥
            method{ format } format{ }                  ¥
            use_filesystem{ } filesystem{ ext4 }        ¥
            mountpoint{ /boot }                         ¥
        .                                               ¥
        125000 2000 125000 ext4                         ¥

    $lvmok{ }
¥
            method{ format } format{ }                  ¥
            use_filesystem{ } filesystem{ ext4 }        ¥
            mountpoint{ / }                             ¥
        .                                               ¥
        125000 1000 125000 ext4                         ¥

    $lvmok{ }
¥
            method{ format } format{ }                  ¥
            use_filesystem{ } filesystem{ ext4 }        ¥
            mountpoint{ /var }                          ¥
        .                                               ¥
        200000 10000 -1 ext4                            ¥

    $lvmok{ }
¥
            method{ format } format{ }                  ¥
            use_filesystem{ } filesystem{ ext4 }        ¥
            mountpoint{ /home }                         ¥
        .                                               ¥
    $lvmok{ }
¥
            method{ swap } format{ }                    ¥
```

図５　partition 分割に関する preseed.cfg

３．ansible

３．１　構成管理ツール ansible

ansible は構成管理ツールと呼ばれるものの一つである。構成管理ツールとは，ソフトウェアのインストールや，設定ファイルの編集，必要なコンテンツのアップロードなどを適切に設定，維持するツールである。構成管理ツールは管理用のホスト一台にインストールされ，そこから管理対象となるサーバに対して各サーバの操作・設定を行う。設定の際はなんらかの設定ファイルを用意しておき，その内容に従って各サーバに設定を適用していくイメージとなる(図６)。

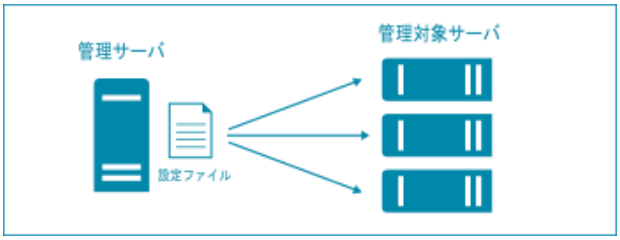

図６　ansible の実行イメージ

ansible 以外に広く使われている構成管理ツールとして puppet，chef がある。この２つより ansible は新しい。また管理対象となるサーバに対してエージェントとなるソフトウェアは必要なく(puppet と chef では必要)，openssh と python があれば動くため，導入への障壁が小さいこともメリットとなる。

```
$ ansible -i hosts servers -a "hostname"
192.168.1.64 | success | rc=0 >>
vbox4

192.168.1.65 | success | rc=0 >>
wadazima
```

```
[servers]
192.168.1.64
192.168.1.65
```

図７　ansible コマンド実行例(上)と hosts ファイル(下)

３．２　ansible コマンド

ansible コマンドを使うと，管理対象サーバに対してコマンドを実行できる(図７上)。管理対象サーバを hosts ファイル(図７下)に記載することで，サーバが増えても容易に対象に追加できる。

３．３　ansible-playbook コマンド

playbook と呼ばれる YAML(Yet Another Markup Language)形式のテキストファイルを用意することで，対象サーバに対してまとまった処理を行うことができる。この playbook を記述することが，ansible を使うときの中心となる作業となる。playbook の例として install_apache.yml を図８，各項目について表２に示した。

```
- hosts: servers
  sudo: true
  tasks:
  - name: apache is installed
    apt: name=apache2 state=present update_cache=yes
```

図８　install_apache.yml

表２　playbook の各項目説明

項目	内容
hosts	hosts ファイル内のサーバを指定する
sudo	sudo を用いて ansible を実行
tasks	以下に task(一連の処理)を記載する
name	処理の説明を書く
apt	適用するモジュールを指定する。ansible では数多くのモジュールが用意されており，モジュールごとにパラメータを指定する。apt は debian 系 OS のパッケージ管理用モジュールである

playbook は図９のように ansible-playbook コマンドで実行する。実行すると結果が表示される。

３．４　Best Practices

playbook で必要な設定を次々と書いていくと playbook が長く複雑になり，また再利用もしにくくなってしまう。

それを解消するため，Best Practice[3] という方法が推奨されている。これは playbook を role という単位に分割してディレクトリごとに管理する方法である。ssh やアクセス制限の設定といった共通的な playbook は common, メールサーバをインストールする設定には mail・・・というように様々な role(役割)に分ける。こうすることで playbook の見通しもよくなり，あるサーバで使った role を別のサーバで使う，といったことも容易となる。今回筆者の環境では，図１０のようなディレクトリ構成で playbook をまとめ，roles を作成した。

３．５　サーバへの適用

ansible を用いて，メールサーバ，バックアップサーバ，データベースサーバの３台に設を行った。これも preseed と同様に仮想サーバにて playbook の動作確認をした。メールサーバとバックアップサーバに対しては15分程度で全設定が完了した。しかしデータベースサーバは trac や subversion を使ったもので，既存のデータをリストアする必要があった。そのため手順が複雑となってしまい playbook を作るのにかなり時間を要し，playbook もかなり複雑で汎用性のないものとなってしまった。既存データを移行する際の ansible の使い方にはもう少し習熟が必要と感じた。

４．まとめ

preseed と ansible という２つのツールを用いて，サーバのインストールと環境構築を行った。これらのツールの有用性が確認できた。他のまだ適用していないサーバに対しても，少しずつ適用するといったことも可能なので，管理するすべてのサーバに対してこれらのツールを適用していきたいと考えている。また新たに構築する，もしくはサーバ入れ替えの際の再構築のときなどにも今回使用した設定ファイルなどが流用することができ，効率化の一助となるだろう。

```
$ ansible-playbook -K -i hosts install_apache.yml
sudo password:

PLAY [servers]
*************************************************
GATHERING FACTS
*************************************************
ok: [192.168.1.64]
ok: [192.168.1.65]

TASK: [apache is installed]
*************************************************
changed: [192.168.1.64]
changed: [192.168.1.65]

PLAY RECAP
*************************************************
192.168.1.65          : ok=2   changed=1   unreachable=0
failed=0
192.168.1.64          : ok=2   changed=1   unreachable=0
failed=0
```

図９　playbook を実行したところ

```
playbooks
├── backup-server
│   ├── group_vars
│   └── roles
│       ├── common
│       ├── dhcpd
│       ├── rsync-client
│       └── rsync-server
├── postfix-server
│   ├── group_vars
│   └── roles
│       ├── common
│       ├── dovecot
│       ├── ircd
│       ├── postfix
│       └── procmail
└── trac-server
    ├── group_vars
    └── roles
        ├── apache
        ├── common
        ├── subversion
        └── trac
```

図１０　筆者の環境の playbook のディレクトリ構成および roles

参考文献

[1] http://www.debian.org/releases/stable/kfreebsd-i386/apbs04.html.ja

[2] http://github.com/xobs/debian-installer/blob/master/doc/devel/partman-auto-recipe.txt

[3] http://docs.ansible.com/ansible/latest/playbooks_best_practices.html