Ansible ワークブック

2020 年 05 月 10 日

Contents:

この資料は「Ansible Workshop - Ansible for Red Hat Enterprise Linux」の「Section 1 - Ansible Engine の演習」を元に作成しました。

第 1 章

Ansible とは

【トピックス】

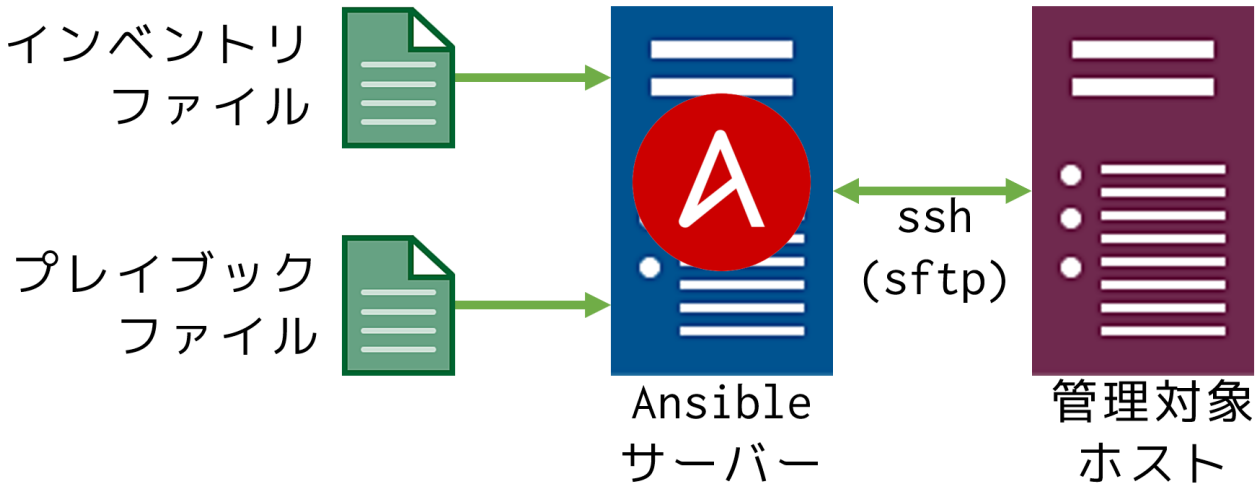
- 構成管理ツール
- 動作イメージ
- *Ansible* の用途

1.1 構成管理ツール

- Ansible は「構成管理ツール」の 1 つです。
- 構成管理ツールは対象の「あるべき状態」を維持するためのツールです。
  - 対象が「あるべき状態」のとき、構成管理ツールは何もしません　＝　何もする必要がありません。
  - 対象が「あるべき状態」と異なるとき、構成管理ツールは対象を「あるべき状態」に変更します。
- Ansible は対象の「あるべき状態」を定義し、それを対象に適用します。

1.2 動作イメージ

- Ansible は「インベントリファイル」と「プレイブックファイル」の 2 つのファイルを読み込み動作します。
- 「インベントリファイル」に Ansible が管理する対象（管理対象ホスト）の情報を記載します。
- 「プレイブックファイル」に 対象の「あるべき状態」を記載します。



1. Ansible サーバーで Ansible がプレイブックファイルの内容をもとに Python コードを生成します。
2. Ansible サーバーからインベントリファイルに記載した管理対象ホストに生成した Python コードを sftp で転送します。
3. 管理対象ホストは受け取った Python コードを実行します。
4. 管理対象ホストは Python コードを実行終了後に削除します。

1.3 Ansible の用途

Ansible は主に次の用途に使用されます。

- 端末のデプロイ　＝　OS をインストールした後に、各種のミドルウェアをインストールおよび設定
- 端末のメンテナンス　＝　ユーザー作成、パッチの適用、設定変更 等
- 端末から各種情報の収集

注釈: ここでいう「端末」は、サーバーやネットワーク機器を指します。

第 2 章

検証環境の構築

【トピックス】

- 事前にインストールしておくツール
- 検証用ホストの作成
- 検証環境の確認
- 環境に関する補足

2.1 事前にインストールしておくツール

検証のため事前に次のツールをインストールします。

- Oracle VirtualBox
- Vagrant
- Microsoft Vusual Studio Code (VS Code)
- Tera Term
- WinSCP

> **ご用心:** 32 bit 版と 64 bit 版があるときは 64bit 版をインストールします。

2.2 検証用ホストの作成

- Vagrant で検証用ホスト（ Ansible サーバー + 管理対象ホスト x 3 ）を作成します。
- OS はすべて CentOS 7 です。
- 使用するユーザーアカウントは vagrant です。

2.2.1 ホストの仕様

表 1　Ansible サーバー

ホスト名	MAC アドレス	IP アドレス	CPU 数	メモリ（MB)
ansible	000d58000151	192.168.1.151	2	8192

表 2　管理対象ホスト

ホスト名	MAC アドレス	IP アドレス	CPU 数	メモリ （MB)
node1	000d58000161	192.168.1.161	1	4096
node2	000d58000162	192.168.1.162	1	4096
node3	000d58000163	192.168.1.163	1	4096

2.2.2 Vagrantfile

```
$script1 = <<-'SCRIPT'
timedatectl set-timezone Asia/Tokyo
yum -y remove open-vm-tools
yum -y update
SCRIPT

$script2 = <<-'SCRIPT'
yum -y install epel-release
yum -y install vim-enhanced
yum -y install ansible
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
    vb.cpus = 1
    vb.customize [
      "modifyvm", :id,
```

(次のページに続く)

(前のページからの続き)

```
      "--ioapic", "on",
      "--graphicscontroller", "vmsvga"
    ]
  end

  config.vm.define :ansible do |ansible|
    ansible.vm.network "public_network", mac: "000d58000151", ip: "192.168.1.151"
    ansible.vm.hostname = "ansible"
    ansible.vm.provider "virtualbox" do |vb|
      vb.name = "Ansible"
      vb.memory = "8192"
      vb.cpus = 2
    end
    ansible.vm.provision "shell", inline: $script1
    ansible.vm.provision "shell", inline: $script2
  end

  config.vm.define :node1 do |node1|
    node1.vm.network "public_network", mac: "000d58000161", ip: "192.168.1.161"
    node1.vm.hostname = "node1"
    node1.vm.provider "virtualbox" do |vb|
      vb.name = "node1"
    end
    node1.vm.provision "shell", inline: $script1
  end

  config.vm.define :node2 do |node2|
    node2.vm.network "public_network", mac: "000d58000163", ip: "192.168.1.162"
    node2.vm.hostname = "node2"
    node2.vm.provider "virtualbox" do |vb|
      vb.name = "node2"
    end
    node2.vm.provision "shell", inline: $script1
  end

  config.vm.define :node3 do |node3|
    node3.vm.network "public_network", mac: "000d58000163", ip: "192.168.1.163"
    node3.vm.hostname = "node3"
    node3.vm.provider "virtualbox" do |vb|
      vb.name = "node3"
    end
    node3.vm.provision "shell", inline: $script1
  end

end
```

2.2.3 Vagrant コマンド

```
vagrant up
vagrant reload
```

2.2.4 Ansible サーバーに管理対象ホストの秘密鍵を設定

Vagrant で作成した仮想マシンの秘密鍵はすべての同じファイル名 private_key です。これを管理対象ホストごとにファイル名を変更し、 Ansible サーバーの~/.ssh/ディレクトリにコピーします。

1. 管理対象ホストの秘密鍵を別名にコピーします。

管理対象ホスト	元のファイル名	コピー後のファイル名
node1	private_key	node1_key
node2	private_key	node2_key
node3	private_key	node3_key

2. コピーした秘密鍵を WinSCP などで Ansible サーバーの~/.ssh/ディレクトリへコピーします。
3. コピーした秘密鍵のパーミッションを 0600 に変更します。

作業後の Ansible サーバーの~/.ssh/ディレクトリの状態です。

```
[vagrant@ansible ~]$ ls -l ~/.ssh
total 16
-rw-------. 1 vagrant vagrant  389 Apr 18 20:26 authorized_keys
-rw-------. 1 vagrant vagrant 1706 Apr 18 20:38 node1_key
-rw-------. 1 vagrant vagrant 1702 Apr 18 20:49 node2_key
-rw-------. 1 vagrant vagrant 1706 Apr 18 21:02 node3_key
[vagrant@ansible ~]$
```

2.2.5 管理対象ホストのフィンガープリントの収集

Ansible サーバーが管理対象ホストを管理するには、事前に管理対象ホストのフィンガープリントが保存されていなければなりません。フィンガープリントの収集と接続テストとして Ansible サーバーから管理対象ホストに ssh 接続します。

収集した管理対象ホストのフィンガープリントは~/.ssh/known_hosts ファイルに記録されます。

```
[vagrant@ansible ~]$ ssh 192.168.1.161 -l vagrant -i ~/.ssh/node1_key
The authenticity of host '192.168.1.161 (192.168.1.161)' can't be established.
ECDSA key fingerprint is SHA256:UNUhsVH7Qld9Ew0ED6X45Yb1f1ms8bDPQ1nl5pfY14k.
```

(次のページに続く)

(前のページからの続き)

```
ECDSA key fingerprint is MD5:e4:f6:82:7e:4f:c0:4c:63:e7:e9:52:45:47:27:ed:44.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.161' (ECDSA) to the list of known hosts.
[vagrant@node1 ~]$ logout
Connection to 192.168.1.161 closed.
[vagrant@ansible ~]$ ssh 192.168.1.162 -l vagrant -i ~/.ssh/node2_key
The authenticity of host '192.168.1.162 (192.168.1.162)' can't be established.
ECDSA key fingerprint is SHA256:YuZuGUV8z8ELd5Fw9ED4tinwkcF58/NUuX5w8hkghn8.
ECDSA key fingerprint is MD5:42:e7:0b:a9:42:84:31:13:68:fc:55:25:80:57:f8:01.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.162' (ECDSA) to the list of known hosts.
[vagrant@node2 ~]$ logout
Connection to 192.168.1.162 closed.
[vagrant@ansible ~]$ ssh 192.168.1.163 -l vagrant -i ~/.ssh/node3_key
The authenticity of host '192.168.1.163 (192.168.1.163)' can't be established.
ECDSA key fingerprint is SHA256:sQ3tRIeDTA925zIDEkQAAvJ6dkT0zgYWF8DqJLR41Mg.
ECDSA key fingerprint is MD5:b8:df:9d:10:8d:89:77:ee:49:97:be:b5:fe:d6:f0:07.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.163' (ECDSA) to the list of known hosts.
[vagrant@node3 ~]$ logout
Connection to 192.168.1.163 closed.
[vagrant@ansible ~]$ ls -l ~/.ssh/
total 20
-rw-------. 1 vagrant vagrant  389 Apr 18 20:26 authorized_keys
-rw-r--r--. 1 vagrant vagrant  525 Apr 18 22:19 known_hosts
-rw-------. 1 vagrant vagrant 1706 Apr 18 20:38 node1_key
-rw-------. 1 vagrant vagrant 1702 Apr 18 20:49 node2_key
-rw-------. 1 vagrant vagrant 1706 Apr 18 21:02 node3_key
[vagrant@ansible ~]$ cat ~/.ssh/known_hosts
192.168.1.161 ecdsa-sha2-nistp256␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGE6rOPnLm36McURf9DCfyixsu99TtX9Qzxi+zliKO+qAv5
192.168.1.162 ecdsa-sha2-nistp256␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBA3rcVFUncRu9ZT5qxnbJxs4rnt+VU7eusihPkUi2cp1mj8
↪5AOrCmooH878v5kMpPQ=
192.168.1.163 ecdsa-sha2-nistp256␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLBgk7iSI5wCseZapsb1BVFgOiAVRf/
↪9jE0tNWdzMsjkHRQb/bZzZhJfVxqobgL5DTF9jUyUu8xPa+tSc3SJwLA=
[vagrant@ansible ~]$
```

ちなみに: Ansible サーバーに~/.ssh/config ファイルを作成し次の内容を記述すると、事前にフィンガープリントの収集をしなくても管理対象ホストに接続できます。

```
Host *
  StrictHostKeyChecking no
  UserKnownHostsFile=/dev/null
```

この方法は便利ですが、セキュリティの観点からは好ましい方法ではありません。

2.3 検証環境の確認

「**検証用ホストの作成**」で Ansible を使用するための前提条件となるタスクは実行済みです。

- Ansible はインストール済みです。
- ssh 接続に必要な設定と鍵は設定済みです。
- root 権限が必要なコマンドが実行可能なように sudo は適切に設定され、権限昇格が可能になっています。

root になるには次のように実行します。

```
[vagrant@ansible ~]$ sudo -i
```

インストールされている Ansible のバージョンを確認します。

```
[root@ansible ~]# ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/
↪ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /bin/ansible
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red Hat␣
↪4.8.5-39)]
[root@ansible ~]#
```

root アカウントからログアウトするには次のように実行します。

```
[root@ansible ~]# exit
```

2.4 環境に関する補足

- Ansible はコマンドラインで操作することが多いです。必要に応じて、ソースコードやコマンドをコピー & ペーストしてください。
- 「**検証用ホストの作成**」で Ansible サーバーに vim をインストールしています。 vim の環境設定はホームディレクトリ内の `.vimrc` ファイルで行います。下記は `.vimrc` ファイルの一例です。

```
set noswapfile
set number
set cursorline
set showmatch
set expandtab
set softtabstop=2
set shiftwidth=2
set autoindent
set showcmd
set visualbell
syntax enable
```

第 3 章

アドホックコマンド

アドホックコマンドを実行して Ansible がどのように動作するのか学習します。 Ansible は `ansible` コマンドを使用してアドホックコマンドを実行します。`ansible` コマンドはプレイブックを作成しなくても管理対象ホストでタスクを実行します。複数の管理対象ホストにアドホックコマンドを実行するときはとても便利なコマンドです。

【トピックス】

- **インベントリの作成**
  - **インベントリの基本**
  - **管理対象ノードのグループ化**
  - **インベントリ内の共通項目をまとめる**
  - **インベントリのチェック**
  - **インベントリのデフォルトグループ**
- ***Ansible* の設定ファイル**
- **管理対象ホストへ *ping* モジュールを実行**
- **モジュールのリストとヘルプを表示**
- **コマンドモジュール**
- **権限昇格**
- **べき等性**
- **アドホックコマンドの演習**

3.1 インベントリの作成

3.1.1 インベントリの基本

ansible コマンドを使用するには Ansible サーバー上に管理対象ホストの一覧を定義したインベントリファイルが必要です。「**検証用ホストの作成**」で作成した管理対象ノードのインベントリです。インベントリファイル名は hosts.yml です。

```
---
all:
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node1_key
    node2:
      ansible_host: 192.168.1.162
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node2_key
    node3:
      ansible_host: 192.168.1.163
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node3_key
```

ansible コマンドのインベントリに関係するオプションです。

-i, --inventory, --inventory-file インベントリファイルを指定します。

--list-hosts インベントリ内で実行対象になる管理対象ホストを確認します。

「ホストパターン」を使用してインベントリ内で実行対象にする管理対象ホストを指定します。もっとも基本的なホストパターンは管理対象ホスト名です。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml --list-hosts
  hosts (1):
    node1
[vagrant@ansible ~]$
```

ホストパターンに all を指定するとインベントリ内のすべての管理対象ホストが対象になります。

```
[vagrant@ansible ~]$ ansible all -i hosts.yml --list-hosts
  hosts (3):
    node1
    node3
    node2
[vagrant@ansible ~]$
```

次のホストパターンも可能です。

```
[vagrant@ansible ~]$ ansible node1,node3 -i hosts.yml --list-hosts
  hosts (
      2):
    node1
    node3
[vagrant@ansible ~]$
```

```
[vagrant@ansible ~]$ ansible node* -i hosts.yml --list-hosts
  hosts (3):
    node1
    node3
    node2
[vagrant@ansible ~]$
```

3.1.2 管理対象ノードのグループ化

インベントリ内の管理対象ホストでグループを作成できます。今回は管理対象ホスト node2 と node3 で web グループを作成しました。

```
---
all:
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node1_key
  children:
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_user: vagrant
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_user: vagrant
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

これはこのようにも書けます。

```
---
all:
  hosts:
    node1:
```

(次のページに続く)

(前のページからの続き)

```
      ansible_host: 192.168.1.161
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node1_key
    node2:
      ansible_host: 192.168.1.162
      ansible_user: vagrant
      ansible_ssh_private_key_file: ~/.ssh/node2_key
    node3:
        ansible_host: 192.168.1.163
        ansible_user: vagrant
        ansible_ssh_private_key_file: ~/.ssh/node3_key
  children:
    web:
      hosts:
        node2:
        node3:
```

グループ名はホストパターンに使用できます。

```
[vagrant@ansible ~]$ ansible web -i hosts.yml --list-hosts
  hosts (2):
    node3
    node2
[vagrant@ansible ~]$
```

グループと管理対象ホスト名の組み合わせも可能です。

```
[vagrant@ansible ~]$ ansible web,node1 -i hosts.yml --list-hosts
  hosts (3):
    node3
    node2
    node1
[vagrant@ansible ~]$
```

3.1.3 インベントリ内の共通項目をまとめる

インベントリを記述するとき、共通部分をまとめることができます。今回はすべての管理対象ノードに ansible_user: vagrant が含まれているので、次のように書き直すことができます。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
```

(次のページに続く)

(前のページからの続き)

```
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
    node2:
      ansible_host: 192.168.1.162
      ansible_ssh_private_key_file: ~/.ssh/node2_key
    node3:
        ansible_host: 192.168.1.163
        ansible_ssh_private_key_file: ~/.ssh/node3_key
  children:
    web:
      hosts:
        node2:
        node3:
```

3.1.4 インベントリのチェック

ansible-inventory コマンドを使用して、インベントリをチェックできます。主なオプションです。

--graph グループとグループに含まれる管理対象ノードをツリー形式で表示します。

--list 管理対象ノードごとに、どの様に定義されているかをリスト形式に表示します。

--yaml --list 形式と併用し、結果を YAML 形式に表示します。

--graph オプションの実行結果です。先頭に @ が付いているのがグループです。

```
[vagrant@ansible ~]$ ansible-inventory -i hosts.yml --graph
@all:
  |--@ungrouped:
  |  |--node1
  |--@web:
  |  |--node2
  |  |--node3
[vagrant@ansible ~]$
```

--list オプションの実行結果です。

```
[vagrant@ansible ~]$ ansible-inventory -i hosts.yml --list
{
    "_meta": {
        "hostvars": {
            "node1": {
                "ansible_host": "192.168.1.161",
                "ansible_ssh_private_key_file": "~/.ssh/node1_key",
                "ansible_user": "vagrant"
            },
```

(次のページに続く)

(前のページからの続き)

```
            "node2": {
                "ansible_host": "192.168.1.162",
                "ansible_ssh_private_key_file": "~/.ssh/node2_key",
                "ansible_user": "vagrant"
            },
            "node3": {
                "ansible_host": "192.168.1.163",
                "ansible_ssh_private_key_file": "~/.ssh/node3_key",
                "ansible_user": "vagrant"
            }
        }
    },
    "all": {
        "children": [
            "ungrouped",
            "web"
        ]
    },
    "ungrouped": {
        "hosts": [
            "node1"
        ]
    },
    "web": {
        "hosts": [
            "node2",
            "node3"
        ]
    }
}
[vagrant@ansible ~]$
```

--yaml オプションの実行結果です。

```
all:
  children:
    ungrouped:
      hosts:
        node1:
          ansible_host: 192.168.1.161
          ansible_ssh_private_key_file: ~/.ssh/node1_key
          ansible_user: vagrant
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
```

(次のページに続く)

(前のページからの続き)

```
          ansible_user: vagrant
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
          ansible_user: vagrant
[vagrant@ansible ~]$
```

3.1.5 インベントリのデフォルトグループ

インベントリは必ず 2 つのデフォルトグループを含みます。

all　すべての管理対象ホストが含まれるグループです。

ungrouped　all 以外にどのグループにも属さない管理対象ホストが属すグループです。

すべてのグループの親グループにあたるのが all グループです。明示的に定義するグループは子（ children ）グループになります。

3.2 Ansible の設定ファイル

Ansible は ini 形式の設定ファイルで動作をカスタマイズできます。設定ファイルは複数の場所（ディレクトリ）に書けます。 Ansible が設定ファイルを検索する順序です。

1. 環境変数 `ANSIBLE_CONFIG` に設定されているパス（ファイル）
2. カレントディレクトリの `ansible.cfg` ファイル
3. ホームディレクトリの`.ansible.cfg` ファイル
4. `/etc/ansible/ansible.cfg` ファイル

Ansible は上記順序で設定ファイルを検索し、最初に見つけた設定ファイルを使用します。「**検証用ホストの作成**」で作成した Ansible サーバーの設定ファイルは`/etc/ansible/ansible.cfg` ファイルです。ほかのファイルは存在しません。

3.3 管理対象ホストへ ping モジュールを実行

Ansible の `ping` モジュールを使用して管理対象ホストへ ping を実行します。`ping` モジュールはネットワークで疎通確認に使用する ping コマンドと動作が異なります。管理対象ホストへ ping モジュールの実行が成功した場合、 Ansible がその管理対象ホスト上でコマンドを正しく実行できることを確認できたことになります。

ちなみに: モジュールは特定の処理をするためのプログラムです。ansible コマンドは-a オプションでモジュールに引数を渡します。

ansible コマンドは-m オプションで実行するモジュールを指定します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m ping
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[vagrant@ansible ~]$
```

3.4 モジュールのリストとヘルプを表示

Ansible で用意されているモジュールのリストを表示します。

```
[vagrant@ansible ~]$ ansible-doc -l
```

ちなみに:

- 表示の終了 →*Q* キー
- 表示内容のスクロール →*PgUp* キー /*PgDn* キー

特定のモジュールを検索するときは grep コマンドで絞り込みます。

```
[vagrant@ansible ~]$ ansible-doc -l | grep -i ping
win_ping                                                    A windows version of the␣
↪classic ping module
postgresql_ping                                             Check remote PostgreSQL␣
↪server availability
lambda_event                                                Creates, updates or␣
↪deletes AWS Lambda function event mappings
net_ping                                                    Tests reachability using␣
↪ping from a network device
ping                                                        Try to connect to host,␣
↪verify a usable python and return `pong' on success
fortios_system_switch_interface                             Configure software␣
↪switch interfaces by grouping physical and WiFi interfaces in Fortinet'..(次のページに続く)
```

(前のページからの続き)

```
ibm_sa_vol_map                                                  Handles volume mapping␣
↪on IBM Spectrum Accelerate Family storage systems
selogin                                                         Manages linux user to␣
↪SELinux user mapping
pingdom                                                         Pause/unpause Pingdom␣
↪alerts
fortios_firewall_shaping_profile                                Configure shaping␣
↪profiles in Fortinet's FortiOS and FortiGate
nxos_igmp_snooping                                              Manages IGMP snooping␣
↪global configuration
sefcontext                                                      Manages SELinux file␣
↪context mapping definitions
fortios_system_geoip_override                                   Configure geographical␣
↪location mapping for IP address(es) to override mappings from Forti...
fortios_firewall_shaping_policy                                 Configure shaping␣
↪policies in Fortinet's FortiOS and FortiGate
pn_igmp_snooping                                                CLI command to modify␣
↪igmp-snooping
nxos_ping                                                       Tests reachability using␣
↪ping from Nexus switch
junos_ping                                                      Tests reachability using␣
↪ping from devices running Juniper JUNOS
fortios_switch_controller_igmp_snooping                         Configure FortiSwitch␣
↪IGMP snooping global settings in Fortinet's FortiOS and FortiGate
icx_ping                                                        Tests reachability using␣
↪ping from Ruckus ICX 7000 series switches
ios_ping                                                        Tests reachability using␣
↪ping from Cisco IOS network devices
vyos_ping                                                       Tests reachability using␣
↪ping from VyOS network devices
netapp_e_lun_mapping                                            NetApp E-Series create,␣
↪delete, or modify lun mappings
[vagrant@ansible ~]$
```

特定のモジュールのヘルプや使用例を確認するときは次のようにします。

```
[vagrant@ansible ~]$ ansible-doc group
> GROUP    (/usr/lib/python2.7/site-packages/ansible/modules/system/group.py)

        Manage presence of groups on a host. For Windows targets, use the [win_group]␣
↪module instead.

  * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- gid
        Optional `GID' to set for the group.
```

(次のページに続く)

(前のページからの続き)

```
        [Default: (null)]
        type: int

- local
        Forces the use of "local" command alternatives on platforms that implement it.
        This is useful in environments that use centralized authentication when you␣
↪want to manipulate the local groups.
        (e.g. it uses `lgroupadd' instead of `groupadd').
        This requires that these commands exist on the targeted host, otherwise it␣
↪will be a fatal error.
        [Default: False]
        type: bool
        version_added: 2.6

= name
        Name of the group to manage.

        type: str

- non_unique
        This option allows to change the group ID to a non-unique value. Requires `gid
↪'.
        Not supported on macOS or BusyBox distributions.
        [Default: False]
        type: bool
        version_added: 2.8

- state
        Whether the group should be present or not on the remote host.
        (Choices: absent, present)[Default: present]
        type: str

- system
        If `yes', indicates that the group created is a system group.
        [Default: False]
        type: bool


SEE ALSO:
      * Module user
           The official documentation on the user module.
           https://docs.ansible.com/ansible/2.9/modules/user_module.html
      * Module win_group
           The official documentation on the win_group module.
           https://docs.ansible.com/ansible/2.9/modules/win_group_module.html


REQUIREMENTS:  groupadd, groupdel, groupmod
```

(次のページに続く)

(前のページからの続き)

```
AUTHOR: Stephen Fromm (@sfromm)
        METADATA:
          status:
          - stableinterface
          supported_by: core


EXAMPLES:

- name: Ensure group "somegroup" exists
  group:
    name: somegroup
    state: present


[vagrant@ansible ~]$
```

ちなみに: 必須オプションは OPTIONS の先頭が = になっています。

3.5 コマンドモジュール

command モジュールを使用して、管理対象ホスト上で Linux コマンドを実行します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m command -a 'id'
node1 | CHANGED | rc=0 >>
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined_
↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[vagrant@ansible ~]$
```

command モジュールは実行するコマンドを引数として受け取り、ホストパターンで指定した管理対象ホスト上で実行するモジュールです。ホストパターンに all を指定すると、すべての管理対象ホスト上でコマンドを実行します。

```
[vagrant@ansible ~]$ ansible all -i hosts.yml -m command -a 'id'
node2 | CHANGED | rc=0 >>
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined_
↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023
node3 | CHANGED | rc=0 >>
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined_
↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

(次のページに続く)

(前のページからの続き)

```
node1 | CHANGED | rc=0 >>
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined_
↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[vagrant@ansible ~]$
```

-o オプションを使用すると、実行結果を 1 行で表示します。

```
[vagrant@ansible ~]$ ansible all -i hosts.yml -m command -a 'id' -o
node3 | CHANGED | rc=0 | (stdout) uid=1000(vagrant) gid=1000(vagrant)␣
↪groups=1000(vagrant) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
node2 | CHANGED | rc=0 | (stdout) uid=1000(vagrant) gid=1000(vagrant)␣
↪groups=1000(vagrant) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
node1 | CHANGED | rc=0 | (stdout) uid=1000(vagrant) gid=1000(vagrant)␣
↪groups=1000(vagrant) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[vagrant@ansible ~]$
```

オプション付きの Linux コマンドも実行できます。

```
[vagrant@ansible ~]$ ansible all -i hosts.yml -m command -a 'uname -r'
node3 | CHANGED | rc=0 >>
3.10.0-1062.18.1.el7.x86_64
node1 | CHANGED | rc=0 >>
3.10.0-1062.18.1.el7.x86_64
node2 | CHANGED | rc=0 >>
3.10.0-1062.18.1.el7.x86_64
[vagrant@ansible ~]$
```

ご用心: command モジュールでパイプやリダイレクトなどは使用できません。パイプやリダイレクトなどを使用する場合は shell モジュールを使用します。

3.6 権限昇格

copy モジュールで管理対象ホスト node1 の/etc/motd ファイルの内容を変更します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m copy -a 'content="Managed by␣
↪Ansible\n" dest=/etc/motd'
node1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
```

(次のページに続く)

(前のページからの続き)

```
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "msg": "Destination /etc not writable"
}
[vagrant@ansible ~]$
```

失敗しました。出力に "Destination /etc not writable" とあるので、/etc/motd ファイルに書き込みできなかったようです。 node1 の/etc/motd ファイルの状態を確認します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m command -a 'ls -l /etc/motd'
node1 | CHANGED | rc=0 >>
-rw-r--r--. 1 root root 0 Jun  7  2013 /etc/motd
[vagrant@ansible ~]$
```

インベントリからわかるように Ansible サーバーから node1 にログインするときのユーザーは vagrant です。しかし、/etc/motd ファイルを書き換えることができるのは root だけです。このようなとき、Linux コマンドを実行するときは sudo を指定して権限昇格します。 Ansible では-b オプションを指定して、モジュールの実行時に権限昇格します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m copy -a 'content="Managed by␣
↪Ansible\n" dest=/etc/motd' -b
node1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "65a4290ee5559756ad04e558b0e0c4e3",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
    "src": "/home/vagrant/.ansible/tmp/ansible-tmp-1587289994.14-173455693341419/source
↪",
    "state": "file",
    "uid": 0
}
[vagrant@ansible ~]$
```

今度は実行が成功しました。動作確認のため node1 に接続し、/etc/motd ファイルの内容を確認します。

```
[vagrant@ansible ~]$ ssh 192.168.1.161 -l vagrant -i ~/.ssh/node1_key
Last login: Sun Apr 19 18:54:13 2020 from 192.168.1.151
Managed by Ansible
```

(次のページに続く)

(前のページからの続き)

```
[vagrant@node1 ~]$ cat /etc/motd
Managed by Ansible
[vagrant@node1 ~]$ logout
Connection to 192.168.1.161 closed.
[vagrant@ansible ~]$
```

ログイン時に/etc/motd ファイルに設定した文字列が表示されているので、正しく変更されたことがわかります。/etc/motd ファイルの内容も変更後の内容になっています。

3.7 べき等性

「**権限昇格**」で実行した/etc/motd ファイルを変更するアドホックコマンドをもう一度実行します。

```
[vagrant@ansible ~]$ ansible node1 -i hosts.yml -m copy -a 'content="Managed by␣
↪Ansible\n" dest=/etc/motd' -b
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "mode": "0644",
    "owner": "root",
    "path": "/etc/motd",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
    "state": "file",
    "uid": 0
}
[vagrant@ansible ~]$
```

実行は成功しました。出力メッセージ内に "changed": false と表示されています。これは「アドホックコマンド（タスク）の実行は成功したが、管理対象ホストの状態は変更されなかった」ことを意味します。具体的には、Ansible が/etc/motd ファイルの内容が変更したい（変更後の）内容と同じ状態だったので、ファイルの内容を変更しなかったということです。このように Ansible は管理対象ホストの状態をチェックし、変更が必要なら変更し、変更が不要なら変更しません。その結果、何度アドホックコマンドを実行しても、実行後の管理対象ホストの状態は常に同じになります。このように、何度コマンドを実行しても、実行後が常に同じ状態になることを「**『べき等性（冪等性）』を保っている**」と言います。これは**構成管理ツール**の大きな特徴であり、後述の**プレイブック**を使用した場合も当てはまります。

3.8 アドホックコマンドの演習

※ すべて管理対象ホスト　node1 に対して実施してください。

1. ソフトウェアパッケージを管理する yum を使用できるモジュールを見つけてください。

ヒント: `ansible-doc` コマンドでモジュールの一覧を表示し、その中から grep コマンドで絞り込みます。

---

2. 見つけたモジュールの EXAMPLES: を参考に、管理対象ホスト node1 に Apache の最新バージョンをインストールしてください。

ヒント:

- Apache = httpd
- 最新バージョン = latest
- パッケージ名とインストールするバージョンは`-a` オプションで指定します。
- Linux の yum コマンドでパッケージをインストールするときは sudo の指定が必要です。

---

3. べき等性を確認するため、もう一度 Apache の最新バージョンをインストールしてください。

注釈: 「べき等性」とはどういうことかを思い出して Apache の最新版のインストールの 1 回目と 2 回目の実行結果を比較してください。

---

4. 管理対象ホスト node1 に ssh 接続し、 yum コマンドを実行して Apache がインストールされていることを確認してください。確認後は node1 から logout してください。

ヒント:

yum コマンドでインストール済みパッケージを確認する方法

`yum list installed パッケージ名`

3.8.1 解答

```
[vagrant@ansible ansible-files]$ ansible-doc -l | grep -i yum
yum                                                Manages packages with␣
↪the `yum' package manager
```

(次のページに続く)

(前のページからの続き)

```
yum_repository                                             Add or remove YUM␣
↪repositories
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible-doc yum
> YUM    (/usr/lib/python2.7/site-packages/ansible/modules/packaging/os/yum.py)

        Installs, upgrade, downgrades, removes, and lists packages and groups with the␣
↪`yum' package manager. This module
        only works on Python 2. If you require Python 3 support see the [dnf] module.

  * This module is maintained by The Ansible Core Team
  * note: This module has a corresponding action plugin.

OPTIONS (= is mandatory):

- allow_downgrade
        Specify if the named package and version is allowed to downgrade a maybe␣
↪already installed higher version of that
        package. Note that setting allow_downgrade=True can make this module behave in␣
↪a non-idempotent way. The task could
        end up with a set of packages that does not match the complete list of␣
↪specified packages to install (because
        dependencies between the downgraded package and others can cause changes to␣
↪the packages which were in the earlier
        transaction).
        [Default: no]
        type: bool
        version_added: 2.4

- autoremove
        If `yes', removes all "leaf" packages from the system that were originally␣
↪installed as dependencies of user-
        installed packages but which are no longer required by any such package.␣
↪Should be used alone or when state is
        `absent'
        NOTE: This feature requires yum >= 3.4.3 (RHEL/CentOS 7+)
        [Default: no]
        type: bool
        version_added: 2.7

- bugfix
        If set to `yes', and `state=latest' then only installs updates that have been␣
↪marked bugfix related.
        [Default: no]
        version_added: 2.6

- conf_file
        The remote yum configuration file to use for the transaction.
```

(次のページに続く)

(前のページからの続き)

```
        [Default: (null)]
        version_added: 0.6

- disable_excludes
        Disable the excludes defined in YUM config files.
        If set to `all', disables all excludes.
        If set to `main', disable excludes defined in [main] in yum.conf.
        If set to `repoid', disable excludes defined for given repo id.
        [Default: (null)]
        version_added: 2.7

- disable_gpg_check
        Whether to disable the GPG checking of signatures of packages being installed.␣
↪Has an effect only if state is
        `present' or `latest'.
        [Default: no]
        type: bool
        version_added: 1.2

- disable_plugin
        `Plugin' name to disable for the install/update operation. The disabled␣
↪plugins will not persist beyond the
        transaction.
        [Default: (null)]
        version_added: 2.5

- disablerepo
        `Repoid' of repositories to disable for the install/update operation. These␣
↪repos will not persist beyond the
        transaction. When specifying multiple repos, separate them with a `","'.
        As of Ansible 2.7, this can alternatively be a list instead of `","' separated␣
↪string
        [Default: (null)]
        version_added: 0.9

- download_dir
        Specifies an alternate directory to store packages.
        Has an effect only if `download_only' is specified.
        [Default: (null)]
        type: str
        version_added: 2.8

- download_only
        Only download the packages, do not install them.
        [Default: no]
        type: bool
        version_added: 2.7
```

(次のページに続く)

(前のページからの続き)

```
- enable_plugin
        `Plugin' name to enable for the install/update operation. The enabled plugin␣
↪will not persist beyond the transaction.
        [Default: (null)]
        version_added: 2.5

- enablerepo
        `Repoid' of repositories to enable for the install/update operation. These␣
↪repos will not persist beyond the
        transaction. When specifying multiple repos, separate them with a `","'.
        As of Ansible 2.7, this can alternatively be a list instead of `","' separated␣
↪string
        [Default: (null)]
        version_added: 0.9

- exclude
        Package name(s) to exclude when state=present, or latest
        [Default: (null)]
        version_added: 2.0

- install_weak_deps
        Will also install all packages linked by a weak dependency relation.
        NOTE: This feature requires yum >= 4 (RHEL/CentOS 8+)
        [Default: yes]
        type: bool
        version_added: 2.8

- installroot
        Specifies an alternative installroot, relative to which all packages will be␣
↪installed.
        [Default: /]
        version_added: 2.3

- list
        Package name to run the equivalent of yum list --show-duplicates <package>␣
↪against. In addition to listing packages,
        use can also list the following: `installed', `updates', `available' and `repos
↪'.
        This parameter is mutually exclusive with `name'.
        [Default: (null)]

- lock_timeout
        Amount of time to wait for the yum lockfile to be freed.
        [Default: 30]
        type: int
        version_added: 2.8

- name
```

(次のページに続く)

(前のページからの続き)

```
        A package name or package specifier with version, like `name-1.0'.
        If a previous version is specified, the task also needs to turn `allow_
↪downgrade' on. See the `allow_downgrade'
        documentation for caveats with downgrading packages.
        When using state=latest, this can be `'*'' which means run `yum -y update'.
        You can also pass a url or a local path to a rpm file (using state=present).␣
↪To operate on several packages this can
        accept a comma separated string of packages or (as of 2.0) a list of packages.
        (Aliases: pkg)[Default: (null)]

- releasever
        Specifies an alternative release from which all packages will be installed.
        [Default: (null)]
        version_added: 2.7

- security
        If set to `yes', and `state=latest' then only installs updates that have been␣
↪marked security related.
        [Default: no]
        type: bool
        version_added: 2.4

- skip_broken
        Skip packages with broken dependencies(devsolve) and are causing problems.
        [Default: no]
        type: bool
        version_added: 2.3

- state
        Whether to install (`present' or `installed', `latest'), or remove (`absent'␣
↪or `removed') a package.
        `present' and `installed' will simply ensure that a desired package is␣
↪installed.
        `latest' will update the specified package if it's not of the latest available␣
↪version.
        `absent' and `removed' will remove the specified package.
        Default is `None', however in effect the default action is `present' unless␣
↪the `autoremove' option is enabled for
        this module, then `absent' is inferred.
        (Choices: absent, installed, latest, present, removed)[Default: (null)]

- update_cache
        Force yum to check if cache is out of date and redownload if needed. Has an␣
↪effect only if state is `present' or
        `latest'.
        (Aliases: expire-cache)[Default: no]
        type: bool
        version_added: 1.9
```

(次のページに続く)

(前のページからの続き)

```
- update_only
        When using latest, only update installed packages. Do not install packages.
        Has an effect only if state is `latest'
        [Default: no]
        type: bool
        version_added: 2.5

- use_backend
        This module supports `yum' (as it always has), this is known as `yum3'/`YUM3'/
↪`yum-deprecated' by upstream yum
        developers. As of Ansible 2.7+, this module also supports `YUM4', which is the
↪"new yum" and it has an `dnf' backend.
        By default, this module will select the backend based on the `ansible_pkg_mgr'␣
↪fact.
        (Choices: auto, yum, yum4, dnf)[Default: auto]
        version_added: 2.7

- validate_certs
        This only applies if using a https url as the source of the rpm. e.g. for␣
↪localinstall. If set to `no', the SSL
        certificates will not be validated.
        This should only set to `no' used on personally controlled sites using self-
↪signed certificates as it avoids
        verifying the source site.
        Prior to 2.1 the code worked as if this was set to `yes'.
        [Default: yes]
        type: bool
        version_added: 2.1


NOTES:
      * When used with a `loop:` each package will be processed individually, it is␣
↪much more efficient to pass the
        list directly to the `name` option.
      * In versions prior to 1.9.2 this module installed and removed each package␣
↪given to the yum module separately.
        This caused problems when packages specified by filename or url had to be␣
↪installed or removed together. In
        1.9.2 this was fixed so that packages are installed in one yum transaction.␣
↪However, if one of the packages
        adds a new yum repository that the other packages come from (such as epel-
↪release) then that package needs to
        be installed in a separate task. This mimics yum's command line behaviour.
      * Yum itself has two types of groups.  "Package groups" are specified in the rpm␣
↪itself while "environment
        groups" are specified in a separate file (usually by the distribution). ␣
↪Unfortunately, this division becomes
```

(次のページに続く)

(前のページからの続き)

```
        apparent to ansible users because ansible needs to operate on the group of␣
↪packages in a single transaction and
        yum requires groups to be specified in different ways when used in that way. ␣
↪Package groups are specified as
        "@development-tools" and environment groups are "@^gnome-desktop-environment".␣
↪Use the "yum group list hidden
        ids" command to see which category of group the group you want to install␣
↪falls into.
      * The yum module does not support clearing yum cache in an idempotent way, so it␣
↪was decided not to implement it,
        the only method is to use command and call the yum command directly, namely
↪"command: yum clean all"
        https://github.com/ansible/ansible/pull/31450#issuecomment-352889579


REQUIREMENTS:  yum

AUTHOR: Ansible Core Team, Seth Vidal (@skvidal), Eduard Snesarev (@verm666), Berend␣
↪De Schouwer (@berenddeschouwer), Abhijeet Kasurde (@Akasurde), Adam Mill
        METADATA:
          status:
          - stableinterface
          supported_by: core


EXAMPLES:

- name: install the latest version of Apache
  yum:
    name: httpd
    state: latest

- name: ensure a list of packages installed
  yum:
    name: "{{ packages }}"
  vars:
    packages:
    - httpd
    - httpd-tools

- name: remove the Apache package
  yum:
    name: httpd
    state: absent

- name: install the latest version of Apache from the testing repo
  yum:
    name: httpd
```

(次のページに続く)

(前のページからの続き)

```
    enablerepo: testing
    state: present

- name: install one specific version of Apache
  yum:
    name: httpd-2.2.29-1.4.amzn1
    state: present

- name: upgrade all packages
  yum:
    name: '*'
    state: latest

- name: upgrade all packages, excluding kernel & foo related packages
  yum:
    name: '*'
    state: latest
    exclude: kernel*,foo*

- name: install the nginx rpm from a remote repo
  yum:
    name: http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.
↪ngx.noarch.rpm
    state: present

- name: install nginx rpm from a local file
  yum:
    name: /usr/local/src/nginx-release-centos-6-0.el6.ngx.noarch.rpm
    state: present

- name: install the 'Development tools' package group
  yum:
    name: "@Development tools"
    state: present
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m yum -a 'name=httpd␣
↪state=latest' -b
node1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "changes": {
        "installed": [
            "httpd"
        ],
        "updated": []
    },
```

(次のページに続く)

(前のページからの続き)


```
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\n *␣
↪base: ftp.yz.yamagata-u.ac.jp\n * extras: ftp.yz.yamagata-u.ac.jp\n * updates: ftp.
↪yz.yamagata-u.ac.jp\nResolving Dependencies\n--> Running transaction check\n--->␣
↪Package httpd.x86_64 0:2.4.6-93.el7.centos will be installed\n--> Processing␣
↪Dependency: httpd-tools = 2.4.6-93.el7.centos for package: httpd-2.4.6-93.el7.centos.
↪x86_64\n--> Processing Dependency: system-logos >= 7.92.1-1 for package: httpd-2.4.6-
↪93.el7.centos.x86_64\n--> Processing Dependency: /etc/mime.types for package: httpd-
↪2.4.6-93.el7.centos.x86_64\n--> Processing Dependency: libaprutil-1.so.0()(64bit)␣
↪for package: httpd-2.4.6-93.el7.centos.x86_64\n--> Processing Dependency: libapr-1.
↪so.0()(64bit) for package: httpd-2.4.6-93.el7.centos.x86_64\n--> Running transaction␣
↪check\n---> Package apr.x86_64 0:1.4.8-5.el7 will be installed\n---> Package apr-
↪util.x86_64 0:1.5.2-6.el7 will be installed\n---> Package centos-logos.noarch 0:70.0.
↪6-3.el7.centos will be installed\n---> Package httpd-tools.x86_64 0:2.4.6-93.el7.
↪centos will be installed\n---> Package mailcap.noarch 0:2.1.41-2.el7 will be␣
↪installed\n--> Finished Dependency Resolution\n\nDependencies␣
↪Resolved\n\n================================================================================\n␣
↪Package             Arch          Version                     Repository  ␣
↪Size\n================================================================================\nInstalling:
↪httpd               x86_64        2.4.6-93.el7.centos         base        2.7␣
↪M\nInstalling for dependencies:\n apr                 x86_64        1.4.8-5.el7
         base        103 k\n apr-util            x86_64        1.5.2-6.el7
   base         92 k\n centos-logos        noarch        70.0.6-3.el7.centos
base         21 M\n httpd-tools         x86_64        2.4.6-93.el7.centos        base
       92 k\n mailcap             noarch        2.1.41-2.el7               base
  31 k\n\nTransaction␣
↪Summary\n================================================================================\nInstall␣
↪ 1 Package (+5 Dependent packages)\n\nTotal download size: 24 M\nInstalled size: 32␣
↪M\nDownloading packages:\n------------------------------------------------------------
↪----------------------\nTotal                                                 9.5 MB/s |
 24 MB  00:02     \nRunning transaction check\nRunning transaction test\nTransaction␣
↪test succeeded\nRunning transaction\n  Installing : apr-1.4.8-5.el7.x86_64
                          1/6 \n  Installing : apr-util-1.5.2-6.el7.x86_64
                     2/6 \n  Installing : httpd-tools-2.4.6-93.el7.centos.x86_64
                3/6 \n  Installing : centos-logos-70.0.6-3.el7.centos.noarch
            4/6 \n  Installing : mailcap-2.1.41-2.el7.noarch
       5/6 \n  Installing : httpd-2.4.6-93.el7.centos.x86_64
  6/6 \n  Verifying  : mailcap-2.1.41-2.el7.noarch                              1/
↪6 \n  Verifying  : apr-util-1.5.2-6.el7.x86_64                               2/6␣
↪\n  Verifying  : httpd-2.4.6-93.el7.centos.x86_64                           3/6 \n
 Verifying  : apr-1.4.8-5.el7.x86_64                                   4/6 \n  ␣
↪Verifying  : httpd-tools-2.4.6-93.el7.centos.x86_64                    5/6 \n  ␣
↪Verifying  : centos-logos-70.0.6-3.el7.centos.noarch                   6/6␣
↪\n\nInstalled:\n  httpd.x86_64 0:2.4.6-93.el7.centos
          \n\nDependency Installed:\n  apr.x86_64 0:1.4.8-5.el7
                           \n  apr-util.x86_64 0:1.5.2-6.el7
                      \n  centos-logos.noarch 0:70.0.6-3.el7.centos
                  \n  httpd-tools.x86_64 0:2.4.6-93.el7.centos
              \n  mailcap.noarch 0:2.1.41-2.el7
          \n\nComplete!\n"
```


(次のページに続く)

(前のページからの続き)

```
    ]
}
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m yum -a 'name=httpd␣
↪state=latest' -b
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "changes": {
        "installed": [],
        "updated": []
    },
    "msg": "",
    "rc": 0,
    "results": [
        "All packages providing httpd are up to date",
        ""
    ]
}
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ssh 192.168.1.161 -l vagrant -i ~/.ssh/node1_key
Last login: Sat May  9 17:16:05 2020 from 192.168.1.151
[vagrant@node1 ~]$ yum list installed httpd
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: mirrors.cat.net
 * extras: mirrors.cat.net
 * updates: mirrors.cat.net
Installed Packages
httpd.x86_64                                                    2.4.6-93.el7.
↪centos                                                        @base
[vagrant@node1 ~]$ logout
Connection to 192.168.1.161 closed.
[vagrant@ansible ansible-files]$
```

第 4 章

プレイブック

アドホックコマンドは簡単なオペレーションではとても役に立ちます。しかし、サーバーのデプロイなどの複雑なシナリオには適していません。複雑なシナリオには「プレイブック」を使用します。

【トピックス】

- **プレイブックの基本**
- **プレイブックの作成**
- **プレイブックの実行**
- **実行結果の確認**
- **プレイブックの演習**

4.1 プレイブックの基本

プレイブックの構成要素です。

タスク

- アドホックコマンドに該当します。
- モジュールにオプションを渡して実行します。

プレイ

- 1 つ以上のタスクが集まったものです。
- タスクの実行順序は記述順です。

プレイブック

- 1 つ以上のプレイの集合体です。
- プレイの実行順序は記述順です。
- 管理対象ホストでプレイブックを実行すると、管理対象ホストは「あるべき状態」になります。

プレイブックファイル

- プレイブックが書かれたファイルのことです。

プレイブックは YAML 形式で記述します。基本的なルールは以下のとおりです。

- "—" から始まります。
- インデントは半角空白 2 文字です。タブ文字は使用できません。

プレイ内の重要なキーワード (ディレクティブ）です。

name: プレイやタスクの名前を指定します。

hosts: プレイを適用する管理対象ホストをホストパターンで指定します。

tasks: モジュールを呼び出し、モジュールにオプションの値を渡して実行します。

become: プレイまたはタスクの実行時に権限昇格します。

プレイは**べき等性**を保つようにすべきです。管理対象ホストでプレイを実行して「あるべき状態」になった後は、再度プレイを実行しても状態が変更されないようにすべきです。

4.2 プレイブックの作成

インベントリ内の web グループに含まれる管理対象ホスト上に次の 3 ステップで Apache Web Server を構築するプレイを作成します。

1. Apache （ httpd パッケージ）のインストール
2. httpd サービスの起動
3. index.html の配置

4.2.1 作成準備

プレイを作成する準備として、ホームディレクトリ内に各ファイルの格納用に `ansible-files` ディレクトリを作成します。

```
[vagrant@ansible ~]$ cd
[vagrant@ansible ~]$ pwd
/home/vagrant
[vagrant@ansible ~]$ mkdir ./ansible-files/
[vagrant@ansible ~]$ cd ./ansible-files/
[vagrant@ansible ansible-files]$ pwd
/home/vagrant/ansible-files
[vagrant@ansible ansible-files]$
```

次に、「**アドホックコマンド**」で作成したインベントリファイルを ansible-files ディレクトリに移動します。

```
[vagrant@ansible ansible-files]$ mv ../hosts.yml ./
[vagrant@ansible ansible-files]$ ls -l
total 4
-rw-rw-r--. 1 vagrant vagrant 414 Apr 19 15:30 hosts.yml
[vagrant@ansible ansible-files]$
```

移動したインベントリファイルの内容です。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
    node2:
      ansible_host: 192.168.1.162
      ansible_ssh_private_key_file: ~/.ssh/node2_key
    node3:
        ansible_host: 192.168.1.163
        ansible_ssh_private_key_file: ~/.ssh/node3_key
  children:
    web:
      hosts:
        node2:
        node3:
```

4.2.2 targets セクション

vi コマンドなどで apache.yml ファイルを作成し、プレイの定義部分を記述します。この定義部分を targets セクションと呼びます。

```
1 ---
2 - name: Apache server installed
3   hosts: web
4   become: yes
```

1 行目

・プレイの開始です。

2 行目

・このプレイの名前を "Apache server installed" に設定しました。

3 行目

・管理対象をホストパターンで指定します。

・今回は管理対象に web グループを指定しました。

4 行目

・プレイ全体を権限昇格して実行することを宣言しました。

4.2.3 tasks セクション

次にタスクを定義します。タスクを定義する部分を tasks セクションと呼びます。 tasks セクションはキーワード `tasks:`から始まります。

まず、Apache のインストール部分です

```
1   tasks:
2   - name: latest Apache version installed
3     yum:
4       name: httpd
5       state: latest
```

どのモジュールも同じような形式で定義します。

1 行目

・ tasks セクションの開始です。

2 行目

・ このタスクの名前を "latest Apache version installed" に設定しました。

3 行目

・ タスクで使用するモジュールの宣言です。

・ Apache のパッケージをインストールするので `yum` モジュールを使用します。

4 〜 5 行目

・ モジュールの引数を定義します。

・ 今回は yum モジュールの引数を name オプションと state オプションで渡しています。

yum モジュールのオプションです。

name:

- インストールや削除するパッケージ名を指定します。

state:

- present：指定したパッケージをインストールした状態にします。
- latest ：指定したパッケージの最新版をインストールした状態にします。
- absent ：指定したパッケージを削除した状態にします。

次に httpd サービスを起動します。

```
1  - name: Apache enabled and running
2    systemd:
3      name: httpd.service
4      state: started
5      enabled: yes
```

CentOS7/RHEL7 からサービスの管理用に systemd が導入されました（従来の service も使用できる場合があります)。 Ansible でもそれに対応して systemd モジュールが追加されたので、それを使用します。書き方は Apache のインストールと同じ形式です。

systemd モジュールのオプションの説明です。

name:

- 起動や停止などするサービス名を指定します。

state:

- started　：サービスを開始した状態にします。
- stopped　：サービスを停止した状態にします。
- restarted：サービスの状態に関係なく、サービスを再起動した状態にします。
- reloaded ：サービスの状態に菅家なく、サービスをリロードした状態にします。

enabled:

- yes を指定すると、ホストの起動時に指定したサービスを自動起動します。

最後に表示する Web ページを管理対象にコピーします。

```
  - name: copy index.html
    template:
      src: ~/ansible-files/index.j2
      dest: /var/www/html/index.html
```

ファイルのコピーには copy モジュールと template モジュールが使用できます。今回は template モジュールを使用します。

template モジュールのオプションの説明です。

src:

- コピー元のパス（ファイル) を指定します
- コピー元のファイルには変数を埋め込むことができます。

dest:

- コピー先のパス（ファイル）を指定します。

コピー元の index.j2 ファイルの内容です。

```
Hello, Ansible world.<br>
Powerd by {{ inventory_hostname }}<br>
```

プレイ全体です。

```
---
- name: Apache server installed
  hosts: web
  become: yes

  tasks:
  - name: latest Apache version installed
    yum:
      name: httpd
      state: latest
  - name: Apache enabled and running
    systemd:
      name: httpd.service
      state: started
      enabled: yes
  - name: copy index.html
    template:
      src: ~/ansible-files/index.j2
      dest: /var/www/html/index.html
```

4.3 プレイブックの実行

ansible-playbook コマンドでプレイブックを実行します。以下のオプションは ansible コマンドと同じです。

-i, --inventory, --inventory-file インベントリファイルを指定します。

--list-hosts インベントリ内で実行対象になる管理対象ホストを確認します。

次のオプションも使用できます。

--syntax-check プレイブック内のプレイの構文をチェックします。

プレイブックを実行する前に、まず構文に誤りがないかチェックします。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml --syntax-check apache.
↪yml

playbook: apache.yml
[vagrant@ansible ansible-files]$
```

構文に誤りがないことを確認したら、次に実行対象になる管理対象ホストを確認します。今回は node2 と node3 が実行対象です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml --list-hosts apache.yml

playbook: apache.yml

  play #1 (web): Apache server installed        TAGS: []
    pattern: [u'web']
    hosts (2):
      node3
      node2
[vagrant@ansible ansible-files]$
```

次にプレイブックを実行します。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Apache server installed]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [latest Apache version installed]␣
↪*********************************************************************************************
```

(次のページに続く)

(前のページからの続き)

```
changed: [node2]
changed: [node3]

TASK [Apache enabled and running]␣
↪*********************************************************************************************
changed: [node2]
changed: [node3]

TASK [copy index.html]␣
↪*********************************************************************************************
changed: [node3]
changed: [node2]

PLAY RECAP␣
↪*********************************************************************************************
node2                      : ok=4    changed=3    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=4    changed=3    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

プレイブックの実行の成否は PLAY RECAP で確認します。実行対象の管理対象ホストごとに、実行状況のサマリが表示されます。

ok

- 実行が成功したタスクの数です。
- 管理対象ホストが変更されたか否かに関係なくカウントされます。

changed

- 管理対象ホストを変更したタスクの数です。

unreachable

- 管理対象ホストに接続できなかったタスクの数です。

failed

- タスクの実行に失敗した数です。

skipped

- 実行がスキップされた（飛ばされた）タスクの数です。

rescued

- block 内でエラーが発生したが、 rescure ブロックで処理を継続した数です。

ignored

- タスクの実行は失敗したが、その失敗が無視されたタスクの数です。

通常のプレイブックでは ok / changed / skipped だけカウントされていたら、そのプレイブックの実行は成功です。

もう一度、プレイブックを実行します。

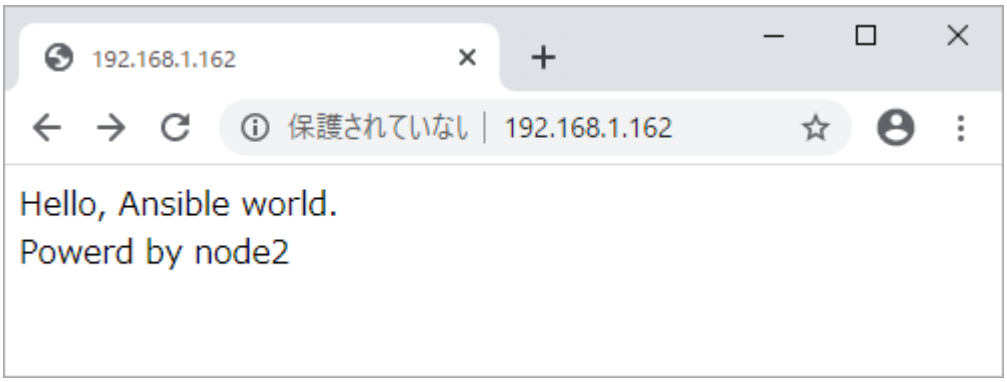
```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Apache server installed]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [latest Apache version installed]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [Apache enabled and running]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [copy index.html]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

PLAY RECAP␣
↪*********************************************************************************************
node2                      : ok=4    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=4    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

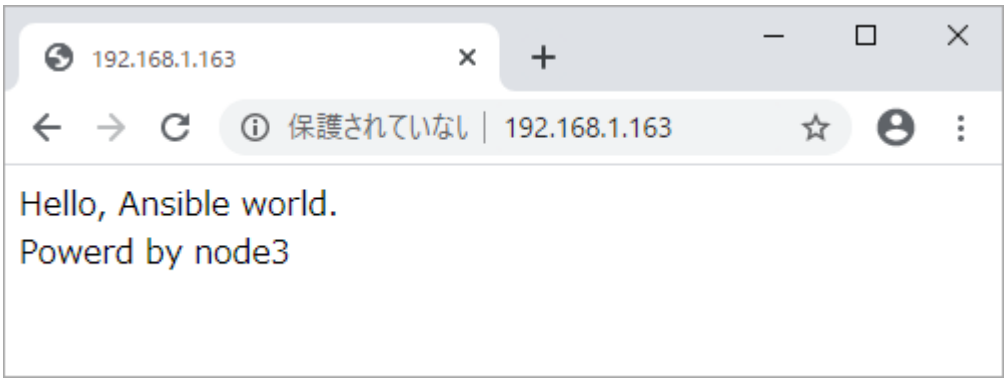
PLAY RECAP の結果が OK だけです。すでに管理対象ホストが「あるべき状態」になっているところにプレイブックを再度実行したので何も変わりませんでした。このように、アドホックコマンドと同様にプレイブックでもべき等性は保たれています。

4.4 実行結果の確認

プレイブックの実行結果を確認します。

まず、配置した index.html が表示されるかブラウザーで確認します。





Powerd by の後に管理対象ホスト名が表示されており、正しく index.html が配置されたことが確認できました。

次に管理対象ホストにログインし、次の状態を確認します。

- Apahce がインストールされていること。
- httpd サービスが起動していること。

```
[vagrant@ansible ansible-files]$ ssh 192.168.1.162 -l vagrant -i ~/.ssh/node2_key
Last login: Mon Apr 20 21:08:54 2020 from 192.168.1.151
[vagrant@node2 ~]$ yum list installed httpd
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: ftp.riken.jp
 * extras: ftp.riken.jp
 * updates: ftp.riken.jp
Installed Packages
httpd.x86_64                                                  2.4.6-90.el7.
↪centos                                                      @base
```

(次のページに続く)

(前のページからの続き)

```
[vagrant@node2 ~]$ rpm -qi httpd
Name        : httpd
Version     : 2.4.6
Release     : 90.el7.centos
Architecture: x86_64
Install Date: Mon 20 Apr 2020 08:57:21 PM JST
Group       : System Environment/Daemons
Size        : 9817301
License     : ASL 2.0
Signature   : RSA/SHA256, Fri 23 Aug 2019 06:25:32 AM JST, Key ID 24c6a8a7f4a80eb5
Source RPM  : httpd-2.4.6-90.el7.centos.src.rpm
Build Date  : Thu 08 Aug 2019 08:43:53 PM JST
Build Host  : x86-01.bsys.centos.org
Relocations : (not relocatable)
Packager    : CentOS BuildSystem <http://bugs.centos.org>
Vendor      : CentOS
URL         : http://httpd.apache.org/
Summary     : Apache HTTP Server
Description :
The Apache HTTP Server is a powerful, efficient, and extensible
web server.
[vagrant@node2 ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:␣
↪disabled)
   Active: active (running) since Mon 2020-04-20 20:57:24 JST; 38min ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 3746 (httpd)
   Status: "Total requests: 6; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─3746 /usr/sbin/httpd -DFOREGROUND
           ├─3747 /usr/sbin/httpd -DFOREGROUND
           ├─3748 /usr/sbin/httpd -DFOREGROUND
           ├─3749 /usr/sbin/httpd -DFOREGROUND
           ├─3750 /usr/sbin/httpd -DFOREGROUND
           ├─3751 /usr/sbin/httpd -DFOREGROUND
           └─3875 /usr/sbin/httpd -DFOREGROUND
[vagrant@node2 ~]$ logout
Connection to 192.168.1.162 closed.
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ssh 192.168.1.163 -l vagrant -i ~/.ssh/node3_key
Last login: Mon Apr 20 21:08:54 2020 from 192.168.1.151
[vagrant@node3 ~]$ yum list installed httpd
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: ftp-srv2.kddilabs.jp
 * extras: ftp-srv2.kddilabs.jp
```

(次のページに続く)

(前のページからの続き)

```
 * updates: ftp-srv2.kddilabs.jp
Installed Packages
httpd.x86_64                                                          2.4.6-90.el7.
↪centos                                                             @base
[vagrant@node3 ~]$ rpm -qi httpd
Name        : httpd
Version     : 2.4.6
Release     : 90.el7.centos
Architecture: x86_64
Install Date: Mon 20 Apr 2020 08:57:21 PM JST
Group       : System Environment/Daemons
Size        : 9817301
License     : ASL 2.0
Signature   : RSA/SHA256, Fri 23 Aug 2019 06:25:32 AM JST, Key ID 24c6a8a7f4a80eb5
Source RPM  : httpd-2.4.6-90.el7.centos.src.rpm
Build Date  : Thu 08 Aug 2019 08:43:53 PM JST
Build Host  : x86-01.bsys.centos.org
Relocations : (not relocatable)
Packager    : CentOS BuildSystem <http://bugs.centos.org>
Vendor      : CentOS
URL         : http://httpd.apache.org/
Summary     : Apache HTTP Server
Description :
The Apache HTTP Server is a powerful, efficient, and extensible
web server.
[vagrant@node3 ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:␣
↪disabled)
   Active: active (running) since Mon 2020-04-20 20:57:24 JST; 39min ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 3742 (httpd)
   Status: "Total requests: 4; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─3742 /usr/sbin/httpd -DFOREGROUND
           ├─3743 /usr/sbin/httpd -DFOREGROUND
           ├─3744 /usr/sbin/httpd -DFOREGROUND
           ├─3745 /usr/sbin/httpd -DFOREGROUND
           ├─3746 /usr/sbin/httpd -DFOREGROUND
           ├─3747 /usr/sbin/httpd -DFOREGROUND
           └─3873 /usr/sbin/httpd -DFOREGROUND
[vagrant@node3 ~]$ logout
Connection to 192.168.1.163 closed.
[vagrant@ansible ansible-files]$
```

管理対象ホストを再起動します。

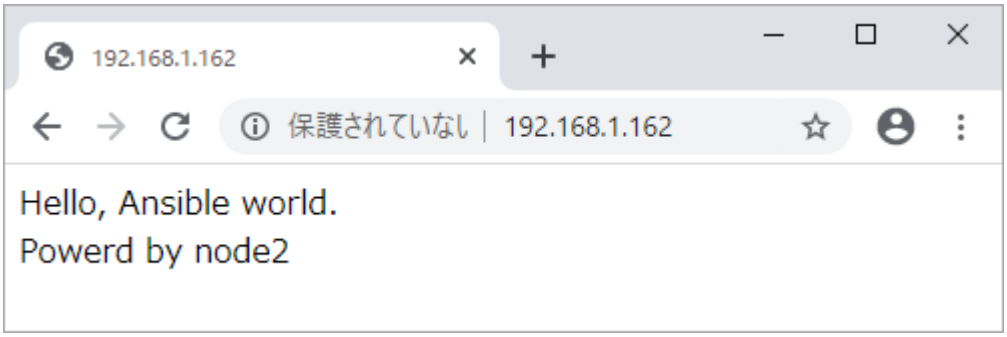
```
C:\vagrant\ansible-study>vagrant reload node2
==> node2: Attempting graceful shutdown of VM...
==> node2: Checking if box 'centos/7' version '1905.1' is up to date...
==> node2: Clearing any previously set forwarded ports...
==> node2: Fixed port collision for 22 => 2222. Now on port 2201.
==> node2: Clearing any previously set network interfaces...
==> node2: Preparing network interfaces based on configuration...
    node2: Adapter 1: nat
    node2: Adapter 2: bridged
==> node2: Forwarding ports...
    node2: 22 (guest) => 2201 (host) (adapter 1)
==> node2: Running 'pre-boot' VM customizations...
==> node2: Booting VM...
==> node2: Waiting for machine to boot. This may take a few minutes...
    node2: SSH address: 127.0.0.1:2201
    node2: SSH username: vagrant
    node2: SSH auth method: private key
==> node2: Machine booted and ready!
[node2] GuestAdditions 6.1.6 running --- OK.
==> node2: Checking for guest additions in VM...
==> node2: Setting hostname...
==> node2: Configuring and enabling network interfaces...
==> node2: Rsyncing folder: /cygdrive/c/vagrant/ansible-study/ => /vagrant
==> node2: Machine already provisioned. Run `vagrant provision` or use the `--
↪provision`
==> node2: flag to force provisioning. Provisioners marked to run always will still␣
↪run.

C:\vagrant\ansible-study>vagrant reload node3
==> node3: Attempting graceful shutdown of VM...
==> node3: Checking if box 'centos/7' version '1905.1' is up to date...
==> node3: Clearing any previously set forwarded ports...
==> node3: Fixed port collision for 22 => 2222. Now on port 2202.
==> node3: Clearing any previously set network interfaces...
==> node3: Preparing network interfaces based on configuration...
    node3: Adapter 1: nat
    node3: Adapter 2: bridged
==> node3: Forwarding ports...
    node3: 22 (guest) => 2202 (host) (adapter 1)
==> node3: Running 'pre-boot' VM customizations...
==> node3: Booting VM...
==> node3: Waiting for machine to boot. This may take a few minutes...
    node3: SSH address: 127.0.0.1:2202
    node3: SSH username: vagrant
    node3: SSH auth method: private key
==> node3: Machine booted and ready!
[node3] GuestAdditions 6.1.6 running --- OK.
==> node3: Checking for guest additions in VM...
```

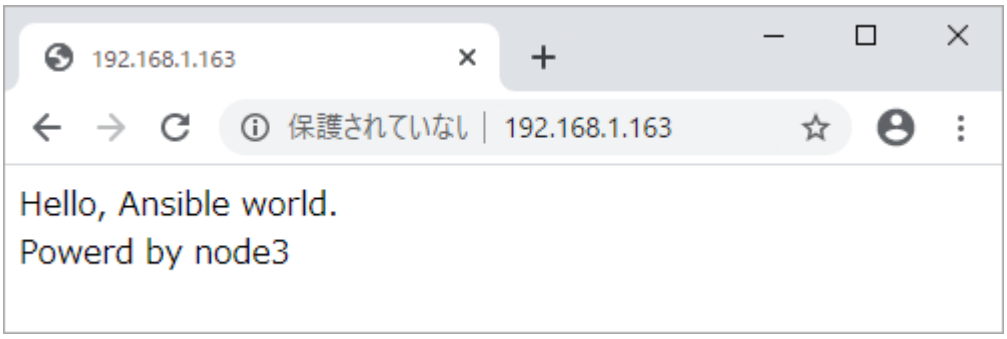
(次のページに続く)

(前のページからの続き)

```
==> node3: Setting hostname...
==> node3: Configuring and enabling network interfaces...
==> node3: Rsyncing folder: /cygdrive/c/vagrant/ansible-study/ => /vagrant
==> node3: Machine already provisioned. Run `vagrant provision` or use the `--
↪provision`
==> node3: flag to force provisioning. Provisioners marked to run always will still␣
↪run.

C:\vagrant\ansible-study>
```

同じ手順で動作確認します。






```
[vagrant@ansible ansible-files]$ ssh 192.168.1.162 -l vagrant -i ~/.ssh/node2_key
Last login: Mon Apr 20 21:35:08 2020 from 192.168.1.151
[vagrant@node2 ~]$ yum list installed httpd
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.riken.jp
 * extras: ftp.riken.jp
 * updates: ftp.riken.jp
Installed Packages
httpd.x86_64                                                    2.4.6-90.el7.
↪centos                                                        @base
[vagrant@node2 ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:␣
↪disabled)
```

(次のページに続く)

(前のページからの続き)

```
   Active: active (running) since Mon 2020-04-20 21:39:54 JST; 8min ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 834 (httpd)
   Status: "Total requests: 1; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─ 834 /usr/sbin/httpd -DFOREGROUND
           ├─ 856 /usr/sbin/httpd -DFOREGROUND
           ├─ 857 /usr/sbin/httpd -DFOREGROUND
           ├─ 858 /usr/sbin/httpd -DFOREGROUND
           ├─ 859 /usr/sbin/httpd -DFOREGROUND
           ├─ 860 /usr/sbin/httpd -DFOREGROUND
           └─2600 /usr/sbin/httpd -DFOREGROUND
[vagrant@node2 ~]$ logout
Connection to 192.168.1.162 closed.
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ssh 192.168.1.163 -l vagrant -i ~/.ssh/node3_key
Last login: Mon Apr 20 21:36:53 2020 from 192.168.1.151
[vagrant@node3 ~]$ yum list installed httpd
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp-srv2.kddilabs.jp
 * extras: ftp-srv2.kddilabs.jp
 * updates: ftp-srv2.kddilabs.jp
Installed Packages
httpd.x86_64                                                          2.4.6-90.el7.
↪centos                                                              @base
[vagrant@node3 ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:␣
↪disabled)
   Active: active (running) since Mon 2020-04-20 21:40:54 JST; 7min ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 833 (httpd)
   Status: "Total requests: 1; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─ 833 /usr/sbin/httpd -DFOREGROUND
           ├─ 857 /usr/sbin/httpd -DFOREGROUND
           ├─ 858 /usr/sbin/httpd -DFOREGROUND
           ├─ 859 /usr/sbin/httpd -DFOREGROUND
           ├─ 861 /usr/sbin/httpd -DFOREGROUND
           ├─ 862 /usr/sbin/httpd -DFOREGROUND
           └─2603 /usr/sbin/httpd -DFOREGROUND
[vagrant@node3 ~]$ logout
Connection to 192.168.1.163 closed.
[vagrant@ansible ansible-files]$
```

再起動後もサービスが正常に起動していることを確認できました。

4.5 プレイブックの演習

1. すべての管理対象ホストに対してプレイが実行できるよう変更してください。

ヒント: targets セクションを変更します。

2. 変更したプレイを実行し、ブラウザで動作を確認してください。

4.5.1 解答

変更後のプレイの内容です。

```
---
- name: Apache server installed
  hosts: all
  become: yes

  tasks:
  - name: latest Apache version installed
    yum:
      name: httpd
      state: latest
  - name: Apache enabled and running
    systemd:
      name: httpd.service
      state: started
      enabled: yes
  - name: copy index.html
    template:
      src: ~/ansible-files/index.j2
      dest: /var/www/html/index.html
```

第 5 章

変数

Ansible はプレイ内で値を格納するための変数をサポートしています。変数はプレイ内、インベントリ内、コマンドラインなどで定義でき、定義する場所に基づいて優先順位が設定されています。

変数をプレイ内で使用する場合、{{と}}でくくります。

【トピックス】

- **変数の使い方**
- **マジック変数**
- **ファクト変数**
- **変数の演習**

5.1 変数の使い方

インベントリ内で変数 stage を定義し、この変数の値によりプレイ内で管理対象ホストにコピーするファイルを変更する例で変数の使い方を学びます。

5.1.1 変数の定義

インベントリの内容です。

```
---
all:
  vars:
```

(次のページに続く)

(前のページからの続き)

```
4     ansible_user: vagrant
5   hosts:
6     node1:
7       ansible_host: 192.168.1.161
8       ansible_ssh_private_key_file: ~/.ssh/node1_key
9     node2:
10      ansible_host: 192.168.1.162
11      ansible_ssh_private_key_file: ~/.ssh/node2_key
12    node3:
13        ansible_host: 192.168.1.163
14        ansible_ssh_private_key_file: ~/.ssh/node3_key
15        stage: prod
16  children:
17    web:
18      vars:
19        stage: dev
20      hosts:
21        node2:
22        node3:
```

変数 satege を 15 行目で管理対象ホスト node3 に、18-19 行目で web グループ全体に適用するよう定義しています。変数が管理対象ホストにどの様に影響するのか `ansible-inventory` コマンドでインベントリの内容を確認します。

```
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --list --yaml
all:
  children:
    ungrouped:
      hosts:
        node1:
          ansible_host: 192.168.1.161
          ansible_ssh_private_key_file: ~/.ssh/node1_key
          ansible_user: vagrant
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
          ansible_user: vagrant
          stage: dev
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
          ansible_user: vagrant
          stage: prod
[vagrant@ansible ansible-files]$
```

同じ変数名の変数を定義した場合、定義した場所で優先順位が決定します。

web グループ内の管理対象ホストに影響する vars の宣言より個別の管理対象ホストに定義した変数のほうが優先順位が高いため、vars の宣言内で定義した変数の値が管理対象ホストに定義した変数の値で上書きされます。

5.1.2 プレイの作成

管理対象ホストにコピーするファイルを 2 つ作成します。

prod_index.html ファイルの内容です。

```
注意してください。これは製品版の Web サーバーです。<br>
```

dev_index.html ファイルの内容です。

```
これは開発中の Web server です。<br>
```

変数 stage に設定した値により、管理対象ホストにコピーするファイルを変更するプレイです。9 行目の管理対象ホストにコピーするファイルを指定する部分で変数を使用しています。

```
1  ---
2  - name: Apache server installed
3    hosts: web
4    become: yes
5
6    tasks:
7    - name: copy index.html
8      copy:
9        src: ~/ansible-files/{{ stage }}_index.html
10       dest: /var/www/html/index.html
```

5.1.3 プレイの実行

作成したプレイを実行します。

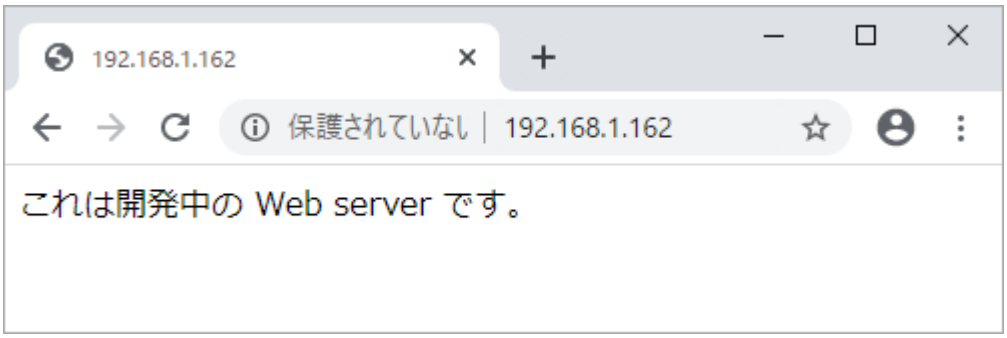
```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Apache server installed]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]
```

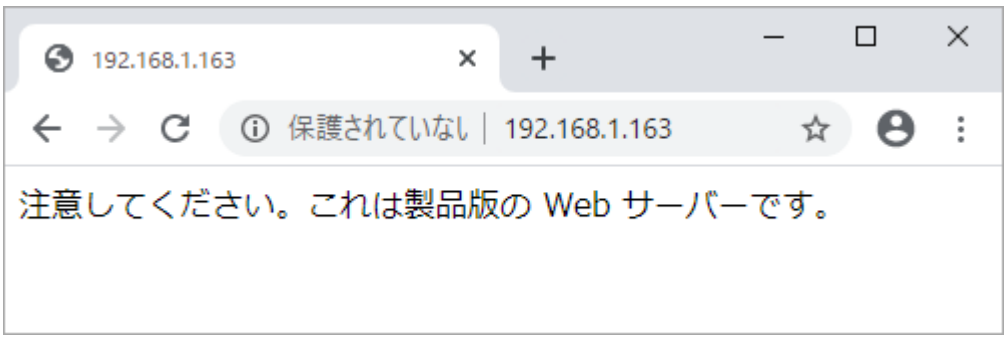
(次のページに続く)

(前のページからの続き)

```
TASK [copy index.html]␣
↪*********************************************************************************************************
changed: [node2]
changed: [node3]

PLAY RECAP␣
↪*********************************************************************************************************
node2                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

変数 stage の値にもとづき所定のファイルが正しくコピーされたかブラウザーで確認します。





変数 stage の値にもとづき、管理対象ホスト node2 に `dev_index.html` ファイルが、 node3 に `prod_index.html` ファイルが正しくコピーされました。

5.2 マジック変数

Ansible の定義済み変数を「マジック変数」と呼びます。定義済みなので、前段の変数 stage のような定義は不要です。主にインベントリの内容や実行環境情報を保持します。

5.2.1 主なマジック変数

inventory_hostname　インベントリ内の管理対象ホスト名

group_names　インベントリ内で管理対象ホストが属するグループの一覧

groups　インベントリ内の全グループと管理対象ホストの一覧

inventory_dir　インベントリファイルのディレクトリパス（インベントリファイル名は含まれない）

inventory_file　インベントリファイルのパス

playbook_dir　プレイブックファイルのディレクトリパス（プレイブックファイル名は含まれない）

host_vars　管理対象ホストの**ファクト変数**を集めたもの

5.2.2 主なマジック変数の確認

次の内容で主なマジック変数の値を確認します。

■ インベントリファイル

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
    node2:
      ansible_host: 192.168.1.162
      ansible_ssh_private_key_file: ~/.ssh/node2_key
    node3:
        ansible_host: 192.168.1.163
        ansible_ssh_private_key_file: ~/.ssh/node3_key
        stage: prod
  children:
    web:
      vars:
        stage: dev
      hosts:
        node2:
        node3:
```

■ プレイブックファイル

```
- name: 主なマジック変数の確認
  hosts: all

  tasks:
  - name: inventory_hostname の値
    debug:
      var: inventory_hostname
  - name: group_names の値
    debug:
      var: group_names
  - name: groups の値
    debug:
      var: groups
  - name: inventory_dir の値
    debug:
      var: inventory_dir
  - name: inventory_file の値
    debug:
      var: inventory_file
  - name: playbook_dir の値
    debug:
      var: playbook_dir
```

■ 確認

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml showmagcivars.yml

PLAY [主なマジック変数の確認]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node1]
ok: [node2]

TASK [inventory_hostname の値]␣
↪*********************************************************************************************
ok: [node1] => {
    "inventory_hostname": "node1"
}
ok: [node3] => {
    "inventory_hostname": "node3"
}
ok: [node2] => {
    "inventory_hostname": "node2"
}
```

(次のページに続く)

(前のページからの続き)

```
TASK [group_names の値]␣
↪*********************************************************************************************************
ok: [node1] => {
    "group_names": [
        "ungrouped"
    ]
}
ok: [node3] => {
    "group_names": [
        "web"
    ]
}
ok: [node2] => {
    "group_names": [
        "web"
    ]
}

TASK [groups の値]␣
↪*********************************************************************************************************
ok: [node1] => {
    "groups": {
        "all": [
            "node1",
            "node3",
            "node2"
        ],
        "ungrouped": [
            "node1"
        ],
        "web": [
            "node3",
            "node2"
        ]
    }
}
ok: [node3] => {
    "groups": {
        "all": [
            "node1",
            "node3",
            "node2"
        ],
        "ungrouped": [
            "node1"
        ],
        "web": [
            "node3",
```

(次のページに続く)

(前のページからの続き)

```
            "node2"
        ]
    }
}
ok: [node2] => {
    "groups": {
        "all": [
            "node1",
            "node3",
            "node2"
        ],
        "ungrouped": [
            "node1"
        ],
        "web": [
            "node3",
            "node2"
        ]
    }
}

TASK [inventory_dir の値]␣
↪*********************************************************************************************************
ok: [node1] => {
    "inventory_dir": "/home/vagrant/ansible-files"
}
ok: [node3] => {
    "inventory_dir": "/home/vagrant/ansible-files"
}
ok: [node2] => {
    "inventory_dir": "/home/vagrant/ansible-files"
}

TASK [inventory_file の値]␣
↪*********************************************************************************************************
ok: [node1] => {
    "inventory_file": "/home/vagrant/ansible-files/hosts.yml"
}
ok: [node3] => {
    "inventory_file": "/home/vagrant/ansible-files/hosts.yml"
}
ok: [node2] => {
    "inventory_file": "/home/vagrant/ansible-files/hosts.yml"
}

TASK [playbook_dir の値]␣
↪*********************************************************************************************************
ok: [node1] => {
```

(次のページに続く)

(前のページからの続き)

```
    "playbook_dir": "/home/vagrant/ansible-files"
}
ok: [node3] => {
    "playbook_dir": "/home/vagrant/ansible-files"
}
ok: [node2] => {
    "playbook_dir": "/home/vagrant/ansible-files"
}

PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=7    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node2                      : ok=7    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=7    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

5.3 ファクト変数

管理対象ホストのシステム情報が格納された変数を「ファクト変数」と呼びます。

5.3.1 ファクト変数の内容

ファクト変数はアドホックコマンドで setup モジュールで確認できます。次の実行結果は管理対象ホスト node1 のファクト変数の内容です。

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m setup
node1 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "10.0.2.15",
            "192.168.1.161"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::5054:ff:fe8a:fee6",
            "fe80::20d:58ff:fe00:161"
        ],
        "ansible_apparmor": {
            "status": "disabled"
```

(次のページに続く)

(前のページからの続き)


```
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/01/2006",
        "ansible_bios_version": "VirtualBox",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-1062.18.1.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "biosdevname": "0",
            "console": "ttyS0,115200n8",
            "crashkernel": "auto",
            "elevator": "noop",
            "net.ifnames": "0",
            "no_timer_check": true,
            "ro": true,
            "root": "UUID=8ac075e3-1124-4bb6-bef7-a6811bf8b870"
        },
        "ansible_date_time": {
            "date": "2020-04-26",
            "day": "26",
            "epoch": "1587867902",
            "hour": "11",
            "iso8601": "2020-04-26T02:25:02Z",
            "iso8601_basic": "20200426T112502105081",
            "iso8601_basic_short": "20200426T112502",
            "iso8601_micro": "2020-04-26T02:25:02.105340Z",
            "minute": "25",
            "month": "04",
            "second": "02",
            "time": "11:25:02",
            "tz": "JST",
            "tz_offset": "+0900",
            "weekday": "Sunday",
            "weekday_number": "0",
            "weeknumber": "16",
            "year": "2020"
        },
        "ansible_default_ipv4": {
            "address": "10.0.2.15",
            "alias": "eth0",
            "broadcast": "10.0.2.255",
            "gateway": "10.0.2.2",
            "interface": "eth0",
            "macaddress": "52:54:00:8a:fe:e6",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "10.0.2.0",
            "type": "ether"
        },
```


(次のページに続く)

(前のページからの続き)


```
        "ansible_default_ipv6": {},
        "ansible_device_links": {
            "ids": {
                "sda": [
                    "ata-VBOX_HARDDISK_VB3958e045-138cceec"
                ],
                "sda1": [
                    "ata-VBOX_HARDDISK_VB3958e045-138cceec-part1"
                ]
            },
            "labels": {},
            "masters": {},
            "uuids": {
                "sda1": [
                    "8ac075e3-1124-4bb6-bef7-a6811bf8b870"
                ]
            }
        },
        "ansible_devices": {
            "sda": {
                "holders": [],
                "host": "IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev␣
↪01)",
                "links": {
                    "ids": [
                        "ata-VBOX_HARDDISK_VB3958e045-138cceec"
                    ],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": "VBOX HARDDISK",
                "partitions": {
                    "sda1": {
                        "holders": [],
                        "links": {
                            "ids": [
                                "ata-VBOX_HARDDISK_VB3958e045-138cceec-part1"
                            ],
                            "labels": [],
                            "masters": [],
                            "uuids": [
                                "8ac075e3-1124-4bb6-bef7-a6811bf8b870"
                            ]
                        },
                        "sectors": "83884032",
                        "sectorsize": 512,
                        "size": "40.00 GB",
```


(次のページに続く)

(前のページからの続き)


```
                        "start": "2048",
                        "uuid": "8ac075e3-1124-4bb6-bef7-a6811bf8b870"
                    }
                },
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "noop",
                "sectors": "83886080",
                "sectorsize": "512",
                "size": "40.00 GB",
                "support_discard": "0",
                "vendor": "ATA",
                "virtual": 1
            }
        },
        "ansible_distribution": "CentOS",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/redhat-release",
        "ansible_distribution_file_variety": "RedHat",
        "ansible_distribution_major_version": "7",
        "ansible_distribution_release": "Core",
        "ansible_distribution_version": "7.7",
        "ansible_dns": {
            "nameservers": [
                "10.0.2.3"
            ],
            "search": [
                "flets-west.jp"
            ]
        },
        "ansible_domain": "",
        "ansible_effective_group_id": 1000,
        "ansible_effective_user_id": 1000,
        "ansible_env": {
            "HOME": "/home/vagrant",
            "LANG": "en_US.UTF-8",
            "LESSOPEN": "||/usr/bin/lesspipe.sh %s",
            "LOGNAME": "vagrant",
            "LS_COLORS": "rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;
↪35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;
↪41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;
↪31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;
↪31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.
↪lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
↪tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
↪sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.
↪rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.
↪pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.
↪png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.
↪mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;
↪35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;
↪35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;
```


(次のページに続く)

(前のページからの続き)


```
            "MAIL": "/var/mail/vagrant",
            "PATH": "/usr/local/bin:/usr/bin",
            "PWD": "/home/vagrant",
            "SELINUX_LEVEL_REQUESTED": "",
            "SELINUX_ROLE_REQUESTED": "",
            "SELINUX_USE_CURRENT_RANGE": "",
            "SHELL": "/bin/bash",
            "SHLVL": "2",
            "SSH_CLIENT": "192.168.1.151 50822 22",
            "SSH_CONNECTION": "192.168.1.151 50822 192.168.1.161 22",
            "SSH_TTY": "/dev/pts/0",
            "TERM": "xterm",
            "USER": "vagrant",
            "XDG_RUNTIME_DIR": "/run/user/1000",
            "XDG_SESSION_ID": "9",
            "_": "/usr/bin/python"
        },
        "ansible_eth0": {
            "active": true,
            "device": "eth0",
            "features": {
                "busy_poll": "off [fixed]",
                "fcoe_mtu": "off [fixed]",
                "generic_receive_offload": "on",
                "generic_segmentation_offload": "on",
                "highdma": "off [fixed]",
                "hw_tc_offload": "off [fixed]",
                "l2_fwd_offload": "off [fixed]",
                "large_receive_offload": "off [fixed]",
                "loopback": "off [fixed]",
                "netns_local": "off [fixed]",
                "ntuple_filters": "off [fixed]",
                "receive_hashing": "off [fixed]",
                "rx_all": "off",
                "rx_checksumming": "off",
                "rx_fcs": "off",
                "rx_gro_hw": "off [fixed]",
                "rx_udp_tunnel_port_offload": "off [fixed]",
                "rx_vlan_filter": "on [fixed]",
                "rx_vlan_offload": "on",
                "rx_vlan_stag_filter": "off [fixed]",
                "rx_vlan_stag_hw_parse": "off [fixed]",
                "scatter_gather": "on",
                "tcp_segmentation_offload": "on",
                "tx_checksum_fcoe_crc": "off [fixed]",
                "tx_checksum_ip_generic": "on",
                "tx_checksum_ipv4": "off [fixed]",
                "tx_checksum_ipv6": "off [fixed]",
```


(次のページに続く)

(前のページからの続き)

```
            "tx_checksum_sctp": "off [fixed]",
            "tx_checksumming": "on",
            "tx_fcoe_segmentation": "off [fixed]",
            "tx_gre_csum_segmentation": "off [fixed]",
            "tx_gre_segmentation": "off [fixed]",
            "tx_gso_partial": "off [fixed]",
            "tx_gso_robust": "off [fixed]",
            "tx_ipip_segmentation": "off [fixed]",
            "tx_lockless": "off [fixed]",
            "tx_nocache_copy": "off",
            "tx_scatter_gather": "on",
            "tx_scatter_gather_fraglist": "off [fixed]",
            "tx_sctp_segmentation": "off [fixed]",
            "tx_sit_segmentation": "off [fixed]",
            "tx_tcp6_segmentation": "off [fixed]",
            "tx_tcp_ecn_segmentation": "off [fixed]",
            "tx_tcp_mangleid_segmentation": "off",
            "tx_tcp_segmentation": "on",
            "tx_udp_tnl_csum_segmentation": "off [fixed]",
            "tx_udp_tnl_segmentation": "off [fixed]",
            "tx_vlan_offload": "on [fixed]",
            "tx_vlan_stag_hw_insert": "off [fixed]",
            "udp_fragmentation_offload": "off [fixed]",
            "vlan_challenged": "off [fixed]"
        },
        "hw_timestamp_filters": [],
        "ipv4": {
            "address": "10.0.2.15",
            "broadcast": "10.0.2.255",
            "netmask": "255.255.255.0",
            "network": "10.0.2.0"
        },
        "ipv6": [
            {
                "address": "fe80::5054:ff:fe8a:fee6",
                "prefix": "64",
                "scope": "link"
            }
        ],
        "macaddress": "52:54:00:8a:fe:e6",
        "module": "e1000",
        "mtu": 1500,
        "pciid": "0000:00:03.0",
        "promisc": false,
        "speed": 1000,
        "timestamping": [
            "tx_software",
            "rx_software",
```

(次のページに続く)

(前のページからの続き)


```
                "software"
            ],
            "type": "ether"
        },
        "ansible_eth1": {
            "active": true,
            "device": "eth1",
            "features": {
                "busy_poll": "off [fixed]",
                "fcoe_mtu": "off [fixed]",
                "generic_receive_offload": "on",
                "generic_segmentation_offload": "on",
                "highdma": "off [fixed]",
                "hw_tc_offload": "off [fixed]",
                "l2_fwd_offload": "off [fixed]",
                "large_receive_offload": "off [fixed]",
                "loopback": "off [fixed]",
                "netns_local": "off [fixed]",
                "ntuple_filters": "off [fixed]",
                "receive_hashing": "off [fixed]",
                "rx_all": "off",
                "rx_checksumming": "off",
                "rx_fcs": "off",
                "rx_gro_hw": "off [fixed]",
                "rx_udp_tunnel_port_offload": "off [fixed]",
                "rx_vlan_filter": "on [fixed]",
                "rx_vlan_offload": "on",
                "rx_vlan_stag_filter": "off [fixed]",
                "rx_vlan_stag_hw_parse": "off [fixed]",
                "scatter_gather": "on",
                "tcp_segmentation_offload": "on",
                "tx_checksum_fcoe_crc": "off [fixed]",
                "tx_checksum_ip_generic": "on",
                "tx_checksum_ipv4": "off [fixed]",
                "tx_checksum_ipv6": "off [fixed]",
                "tx_checksum_sctp": "off [fixed]",
                "tx_checksumming": "on",
                "tx_fcoe_segmentation": "off [fixed]",
                "tx_gre_csum_segmentation": "off [fixed]",
                "tx_gre_segmentation": "off [fixed]",
                "tx_gso_partial": "off [fixed]",
                "tx_gso_robust": "off [fixed]",
                "tx_ipip_segmentation": "off [fixed]",
                "tx_lockless": "off [fixed]",
                "tx_nocache_copy": "off",
                "tx_scatter_gather": "on",
                "tx_scatter_gather_fraglist": "off [fixed]",
                "tx_sctp_segmentation": "off [fixed]",
```


(次のページに続く)

(前のページからの続き)


```
                "tx_sit_segmentation": "off [fixed]",
                "tx_tcp6_segmentation": "off [fixed]",
                "tx_tcp_ecn_segmentation": "off [fixed]",
                "tx_tcp_mangleid_segmentation": "off",
                "tx_tcp_segmentation": "on",
                "tx_udp_tnl_csum_segmentation": "off [fixed]",
                "tx_udp_tnl_segmentation": "off [fixed]",
                "tx_vlan_offload": "on [fixed]",
                "tx_vlan_stag_hw_insert": "off [fixed]",
                "udp_fragmentation_offload": "off [fixed]",
                "vlan_challenged": "off [fixed]"
            },
            "hw_timestamp_filters": [],
            "ipv4": {
                "address": "192.168.1.161",
                "broadcast": "192.168.1.255",
                "netmask": "255.255.255.0",
                "network": "192.168.1.0"
            },
            "ipv6": [
                {
                    "address": "fe80::20d:58ff:fe00:161",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "00:0d:58:00:01:61",
            "module": "e1000",
            "mtu": 1500,
            "pciid": "0000:00:08.0",
            "promisc": false,
            "speed": 1000,
            "timestamping": [
                "tx_software",
                "rx_software",
                "software"
            ],
            "type": "ether"
        },
        "ansible_fibre_channel_wwn": [],
        "ansible_fips": false,
        "ansible_form_factor": "Other",
        "ansible_fqdn": "node1",
        "ansible_hostname": "node1",
        "ansible_hostnqn": "",
        "ansible_interfaces": [
            "lo",
            "eth1",
```


(次のページに続く)

(前のページからの続き)


```
            "eth0"
        ],
        "ansible_is_chroot": true,
        "ansible_iscsi_iqn": "",
        "ansible_kernel": "3.10.0-1062.18.1.el7.x86_64",
        "ansible_kernel_version": "#1 SMP Tue Mar 17 23:49:17 UTC 2020",
        "ansible_lo": {
            "active": true,
            "device": "lo",
            "features": {
                "busy_poll": "off [fixed]",
                "fcoe_mtu": "off [fixed]",
                "generic_receive_offload": "on",
                "generic_segmentation_offload": "on",
                "highdma": "on [fixed]",
                "hw_tc_offload": "off [fixed]",
                "l2_fwd_offload": "off [fixed]",
                "large_receive_offload": "off [fixed]",
                "loopback": "on [fixed]",
                "netns_local": "on [fixed]",
                "ntuple_filters": "off [fixed]",
                "receive_hashing": "off [fixed]",
                "rx_all": "off [fixed]",
                "rx_checksumming": "on [fixed]",
                "rx_fcs": "off [fixed]",
                "rx_gro_hw": "off [fixed]",
                "rx_udp_tunnel_port_offload": "off [fixed]",
                "rx_vlan_filter": "off [fixed]",
                "rx_vlan_offload": "off [fixed]",
                "rx_vlan_stag_filter": "off [fixed]",
                "rx_vlan_stag_hw_parse": "off [fixed]",
                "scatter_gather": "on",
                "tcp_segmentation_offload": "on",
                "tx_checksum_fcoe_crc": "off [fixed]",
                "tx_checksum_ip_generic": "on [fixed]",
                "tx_checksum_ipv4": "off [fixed]",
                "tx_checksum_ipv6": "off [fixed]",
                "tx_checksum_sctp": "on [fixed]",
                "tx_checksumming": "on",
                "tx_fcoe_segmentation": "off [fixed]",
                "tx_gre_csum_segmentation": "off [fixed]",
                "tx_gre_segmentation": "off [fixed]",
                "tx_gso_partial": "off [fixed]",
                "tx_gso_robust": "off [fixed]",
                "tx_ipip_segmentation": "off [fixed]",
                "tx_lockless": "on [fixed]",
                "tx_nocache_copy": "off [fixed]",
                "tx_scatter_gather": "on [fixed]",
```


(次のページに続く)

(前のページからの続き)

```
                "tx_scatter_gather_fraglist": "on [fixed]",
                "tx_sctp_segmentation": "on",
                "tx_sit_segmentation": "off [fixed]",
                "tx_tcp6_segmentation": "on",
                "tx_tcp_ecn_segmentation": "on",
                "tx_tcp_mangleid_segmentation": "on",
                "tx_tcp_segmentation": "on",
                "tx_udp_tnl_csum_segmentation": "off [fixed]",
                "tx_udp_tnl_segmentation": "off [fixed]",
                "tx_vlan_offload": "off [fixed]",
                "tx_vlan_stag_hw_insert": "off [fixed]",
                "udp_fragmentation_offload": "on",
                "vlan_challenged": "on [fixed]"
            },
            "hw_timestamp_filters": [],
            "ipv4": {
                "address": "127.0.0.1",
                "broadcast": "host",
                "netmask": "255.0.0.0",
                "network": "127.0.0.0"
            },
            "ipv6": [
                {
                    "address": "::1",
                    "prefix": "128",
                    "scope": "host"
                }
            ],
            "mtu": 65536,
            "promisc": false,
            "timestamping": [
                "rx_software",
                "software"
            ],
            "type": "loopback"
        },
        "ansible_local": {},
        "ansible_lsb": {},
        "ansible_machine": "x86_64",
        "ansible_machine_id": "a6398579871c2d45a1d70a9a863281d3",
        "ansible_memfree_mb": 3477,
        "ansible_memory_mb": {
            "nocache": {
                "free": 3615,
                "used": 174
            },
            "real": {
                "free": 3477,
```

(次のページに続く)

(前のページからの続き)


```
            "total": 3789,
            "used": 312
        },
        "swap": {
            "cached": 0,
            "free": 2047,
            "total": 2047,
            "used": 0
        }
    },
    "ansible_memtotal_mb": 3789,
    "ansible_mounts": [
        {
            "block_available": 9383586,
            "block_size": 4096,
            "block_total": 10480385,
            "block_used": 1096799,
            "device": "/dev/sda1",
            "fstype": "xfs",
            "inode_available": 20907909,
            "inode_total": 20971008,
            "inode_used": 63099,
            "mount": "/",
            "options": "rw,seclabel,relatime,attr2,inode64,noquota",
            "size_available": 38435168256,
            "size_total": 42927656960,
            "uuid": "8ac075e3-1124-4bb6-bef7-a6811bf8b870"
        }
    ],
    "ansible_nodename": "node1",
    "ansible_os_family": "RedHat",
    "ansible_pkg_mgr": "yum",
    "ansible_proc_cmdline": {
        "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-1062.18.1.el7.x86_64",
        "LANG": "en_US.UTF-8",
        "biosdevname": "0",
        "console": [
            "tty0",
            "ttyS0,115200n8"
        ],
        "crashkernel": "auto",
        "elevator": "noop",
        "net.ifnames": "0",
        "no_timer_check": true,
        "ro": true,
        "root": "UUID=8ac075e3-1124-4bb6-bef7-a6811bf8b870"
    },
    "ansible_processor": [
```


(次のページに続く)

(前のページからの続き)

```
            "0",
            "GenuineIntel",
            "Intel(R) Core(TM) i7-4930K CPU @ 3.40GHz"
        ],
        "ansible_processor_cores": 1,
        "ansible_processor_count": 1,
        "ansible_processor_threads_per_core": 1,
        "ansible_processor_vcpus": 1,
        "ansible_product_name": "VirtualBox",
        "ansible_product_serial": "NA",
        "ansible_product_uuid": "NA",
        "ansible_product_version": "1.2",
        "ansible_python": {
            "executable": "/usr/bin/python",
            "has_sslcontext": true,
            "type": "CPython",
            "version": {
                "major": 2,
                "micro": 5,
                "minor": 7,
                "releaselevel": "final",
                "serial": 0
            },
            "version_info": [
                2,
                7,
                5,
                "final",
                0
            ]
        },
        "ansible_python_version": "2.7.5",
        "ansible_real_group_id": 1000,
        "ansible_real_user_id": 1000,
        "ansible_selinux": {
            "config_mode": "enforcing",
            "mode": "enforcing",
            "policyvers": 31,
            "status": "enabled",
            "type": "targeted"
        },
        "ansible_selinux_python_present": true,
        "ansible_service_mgr": "systemd",
        "ansible_ssh_host_key_ecdsa_public":
↪"AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGE6rOPnLm36McURf9DCfyixsu99TtX9Qzxi+zliKO+qAv
↪",
        "ansible_ssh_host_key_ed25519_public":
↪"AAAAC3NzaC1lZDI1NTE5AAAAIKSgMmdTobNNa9V+v+NGivckMV1PJ31DksJ2Ap6MfcjE",
```

(次のページに続く)

(前のページからの続き)

```
        "ansible_ssh_host_key_rsa_public":
↪"AAAAB3NzaC1yc2EAAAADAQABAAABAQDWX9AKFBfFjQCiQR408nt8VB7RVFwPKwpUgKsMaYKDE9VSeFgsh+oVr+RXjEVxIFgfXe
↪y0Tr0h7z8yVgYYmkpvfSHyHJ7WdbdkxJtHaz+TDDEG8bnv1Z6qmZK2LqRwhNXCX8xO5/
↪uG0oJWFF3o5sMFFzDjGPJkVty760PDHjAS5sTamtrLKHJySJfyNUFAN3BpnS7gCFjB/
↪aTYdXfvt2rH86WaySS95oskyHx6wxd1CK0hYbAY14Sn8qba3QvMpNDZqQ8RQxOwvjp5n7ufRjkcRxb/
↪h0dRGHoVzKr1HV3",
        "ansible_swapfree_mb": 2047,
        "ansible_swaptotal_mb": 2047,
        "ansible_system": "Linux",
        "ansible_system_capabilities": [
            ""
        ],
        "ansible_system_capabilities_enforced": "True",
        "ansible_system_vendor": "innotek GmbH",
        "ansible_uptime_seconds": 11544,
        "ansible_user_dir": "/home/vagrant",
        "ansible_user_gecos": "vagrant",
        "ansible_user_gid": 1000,
        "ansible_user_id": "vagrant",
        "ansible_user_shell": "/bin/bash",
        "ansible_user_uid": 1000,
        "ansible_userspace_architecture": "x86_64",
        "ansible_userspace_bits": "64",
        "ansible_virtualization_role": "guest",
        "ansible_virtualization_type": "virtualbox",
        "discovered_interpreter_python": "/usr/bin/python",
        "gather_subset": [
            "all"
        ],
        "module_setup": true
    },
    "changed": false
}
[vagrant@ansible ansible-files]$
```

5.3.2 ファクト変数の収集タイミング

プレイの実行時にファクト変数の収集は次のメッセージが表示されたタイミングで実施します。

```
TASK [Gathering Facts]␣
↪*********************************************************************************************************
ok: [node3]
ok: [node2]
```

5.3.3 ファクト変数が不要なとき

プレイ内でファクト変数を使用しないときは、 5 行目のように gather_facts ディレクティブに no を設定するとファクト変数を収集しません。

```
1 ---
2 - name: Apache server installed
3   hosts: web
4   become: yes
5   gather_facts: no
```

5.3.4 ファクト変数の参照方法

ファクト変数は次のように参照します。

ansible_facts[' 変数名']

ファクト変数と参照の例です。

```
"ansible_distribution": "CentOS",
```

ansible_facts['distribution']

```
"ansible_default_ipv4": {
    "address": "10.0.2.15",
```

ansible_facts['default_ipv4']['address']

```
"ansible_all_ipv4_addresses": [
    "10.0.2.15",
    "192.168.1.161"
```

ansible_facts['all_ipv4_addresses'][0]

ansible_facts['all_ipv4_addresses'][1]

プレイ内で参照する場合、他の変数と同様に{{と}}でくくって使用します。

```
---
- name: show fact variables
  hosts: web
```

(次のページに続く)

(前のページからの続き)

```
  tasks:
  - name: show ansble_distribution
    debug:
      msg: "{{ inventory_hostname }} のディストリビューションは {{ ansible_facts[
↪'distribution'] }} です。"
  - name: show ansible_default_ipv4.address
    debug:
      msg: "デフォルトの IPv4 アドレスは {{ ansible_facts['default_ipv4']['address'] }} です。
"
  - name: show all_ipv4_addresses
    debug:
      msg: "{{ inventory_hostname }} の IP アドレスは {{ ansible_facts['all_ipv4_addresses
↪'][0] }} と {{ ansible_facts['all_ipv4_addresses'][1] }} です。"
```

実行結果です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml showfactvars.yml

PLAY [show fact variables]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node2]
ok: [node3]

TASK [show ansble_distribution]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "node3 のディストリビューションは CentOS です。"
}
ok: [node2] => {
    "msg": "node2 のディストリビューションは CentOS です。"
}

TASK [show ansible_default_ipv4.address]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "デフォルトの IPv4 アドレスは 10.0.2.15 です。"
}
ok: [node2] => {
    "msg": "デフォルトの IPv4 アドレスは 10.0.2.15 です。"
}

TASK [show all_ipv4_addresses]␣
↪*********************************************************************************************
ok: [node3] => {
```

(次のページに続く)

(前のページからの続き)

```
    "msg": "node3 の IP アドレスは 10.0.2.15 と 192.168.1.163 です。"
}
ok: [node2] => {
    "msg": "node2 の IP アドレスは 192.168.1.162 と 10.0.2.15 です。"
}

PLAY RECAP␣
↪*********************************************************************************************************
node2                      : ok=4    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=4    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

参考までに gather_facts ディレクティブに no を設定し、ファクト変数を収集しなかった場合です。

```
---
- name: show fact variables
  hosts: web
  gather_facts: no

  tasks:
  - name: show ansble_distribution
    debug:
      msg: "{{ inventory_hostname }} のディストリビューションは {{ ansible_facts[
↪'distribution'] }} です。"
  - name: show ansible_default_ipv4.address
    debug:
      msg: "デフォルトの IPv4 アドレスは {{ ansible_facts['default_ipv4']['address'] }} です。
"
  - name: show all_ipv4_addresses
    debug:
      msg: "{{ inventory_hostname }} の IP アドレスは {{ ansible_facts['all_ipv4_addresses
↪'][0] }} と {{ ansible_facts['all_ipv4_addresses'][1] }} です。"
```

この場合、ファクト変数自体が定義されないため、ファクト変数を参照しているタスクでエラーが発生します。

```
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml showfactvars.yml

PLAY [show fact variables]␣
↪*********************************************************************************************************

TASK [show ansble_distribution]␣
↪*********************************************************************************************************
fatal: [node3]: FAILED! => {"msg": "The task includes an option with an undefined␣
↪variable. The error was: 'dict object' has no attribute 'distribution'\n\nThe error␣
↪appears to be in '/home/vagrant/ansible-files/showfactvars.yml': line 7, column 5,␣
↪but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe␣
↪offending line appears to be:\n\n  tasks:\n  - name: show ansble_distribution␣
↪here\n"}
```

(次のページに続く)

(前のページからの続き)

```
fatal: [node2]: FAILED! => {"msg": "The task includes an option with an undefined␣
↪variable. The error was: 'dict object' has no attribute 'distribution'\n\nThe error␣
↪appears to be in '/home/vagrant/ansible-files/showfactvars.yml': line 7, column 5,␣
↪but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe␣
↪offending line appears to be:\n\n  tasks:\n  - name: show ansble_distribution\n    ^␣
↪here\n"}

PLAY RECAP␣
↪*********************************************************************************************
node2                      : ok=0    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=0    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

5.4 変数の演習

1. 管理対象ホスト node1 のホスト名と OS のバージョンを持つファクト変数を見つけてください。

   ヒント:

   - ホスト名　→　hostname
   - OS のバージョン　→　distribution_version

2. アドホックコマンドで node1 のホスト名とバージョンを確認してください。

   ヒント:

   - ホスト名　→**hostname** コマンド
   - OS のバージョン　→`/etc/redhat-release` ファイルの内容を確認

3. ファクト変数を使用して管理対象ホスト node1 のホスト名と OS のバージョンを表示するプレイを作成してください。

   ヒント: プレイ内でファクト変数の使用する方法は「**ファクト変数の参照方法**」を参照してください。

4. 作成したプレイを実行し、実際に管理対象ホスト node1 のホスト名と OS のバージョンを確認してください。

5.4.1 解答

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m setup | grep -i hostname
        "ansible_hostname": "node1",
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m setup | grep -i␣
↪distribution_version
        "ansible_distribution_version": "7.8",
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m command -a hostname
node1 | CHANGED | rc=0 >>
node1
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m command -a 'cat /etc/
↪redhat-release'
node1 | CHANGED | rc=0 >>
CentOS Linux release 7.8.2003 (Core)
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ vim show-info.yml
[vagrant@ansible ansible-files]$ cat show-info.yml
- name: show node1 information
  hosts: node1

  tasks:
  - name: show information
    debug:
      msg: "管理対象ホスト {{ ansible_facts['hostname'] }} の OS のバージョンは {{ ansible_
↪facts['distribution_version'] }} です。"

[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml show-info.yml

PLAY [show node1 information]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node1]

TASK [show information]␣
↪*********************************************************************************************
ok: [node1] => {
    "msg": "管理対象ホスト node1 の OS のバージョンは 7.8 です。"
}

PLAY RECAP␣
↪*********************************************************************************************
```

(次のページに続く)

(前のページからの続き)

```
node1                      : ok=2    changed=0    unreachable=0    failed=0
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

第 6 章

条件分岐、ハンドラー、ループ

6.1 管理対象ホストの追加

説明用に管理対象ホストを追加します。**従来のホストのディレクトリーとは別のディレクトリーに** `Vagrantfile` を作成し、管理対象ホストをデプロイします。以下は `Vagrantfile` の内容です。

```
$script1 = <<-'SCRIPT'
yum -y remove open-vm-tools
SCRIPT

$script2 = <<-'SCRIPT'
apt -y remove open-vm-tools
SCRIPT

$script3 = <<-'SCRIPT'
dnf -y remove open-vm-tools
SCRIPT

Vagrant.configure("2") do |config|
  if Vagrant.has_plugin?("vagrant-vbguest")
    config.vbguest.auto_update = false
  end

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
    vb.cpus = 1
    vb.customize [
      "modifyvm", :id,
      "--ioapic", "on",
      "--graphicscontroller", "vmsvga"
    ]
  end

  config.vm.define :centos7 do |centos7|
```

(次のページに続く)

(前のページからの続き)

```
    centos7.vm.box = "centos/7"
    centos7.vm.network "public_network", mac: "000d58000172", ip: "192.168.1.171"
    centos7.vm.hostname = "centos7"
    centos7.vm.provider "virtualbox" do |vb|
      vb.name = "centos7"
    end
    centos7.vm.provision "shell", inline: $script1
  end

  config.vm.define :centos8 do |centos8|
    centos8.vm.box = "centos/8"
    centos8.vm.network "public_network", mac: "000d58000173", ip: "192.168.1.172"
    centos8.vm.hostname = "centos8"
    centos8.vm.provider "virtualbox" do |vb|
      vb.name = "centos8"
    end
    centos8.vm.provision "shell", inline: $script3
  end

  config.vm.define :debian10 do |debian10|
    debian10.vm.box = "debian/buster64"
    debian10.vm.network "public_network", mac: "000d58000174", ip: "192.168.1.173"
    debian10.vm.hostname = "debian10"
    debian10.vm.provider "virtualbox" do |vb|
      vb.name = "debian10"
    end
  end

  config.vm.define :ubuntu18 do |ubuntu18|
    ubuntu18.vm.box = "ubuntu/bionic64"
    ubuntu18.vm.network "public_network", mac: "000d58000175", ip: "192.168.1.174"
    ubuntu18.vm.hostname = "ubuntu18"
    ubuntu18.vm.provider "virtualbox" do |vb|
      vb.name = "ubuntu18"
    end
    ubuntu18.vm.provision "shell", inline: $script2
  end

end
```

デプロイ後は「**検証用ホストの作成**」で行ったように作成した管理対象ホストの秘密鍵を Ansible サーバーにコピーしたり、**ssh** コマンドでフィンガープリントを収集します。

追加した管理対象ホスト用のインベントリファイル `hosts2.yml` の内容です。

```
---
all:
```

(次のページに続く)

(前のページからの続き)

```
  vars:
    ansible_user: vagrant
  hosts:
    centos7:
      ansible_host: 192.168.1.171
      ansible_ssh_private_key_file: ~/.ssh/centos7_key
    centos8:
      ansible_host: 192.168.1.172
      ansible_ssh_private_key_file: ~/.ssh/centos8_key
    debian10:
      ansible_host: 192.168.1.173
      ansible_ssh_private_key_file: ~/.ssh/debian10_key
      ansible_python_interpreter: /usr/bin/python
    ubuntu18:
      ansible_host: 192.168.1.174
      ansible_ssh_private_key_file: ~/.ssh/ubuntu18_key
```

6.2 条件分岐

タスクに when ディレクティブを使用して条件を設定することで、条件を満たしたときだけタスクを実行するようにできます。

【トピックス】

- **使用できる演算子**
- ***when* ディレクティブ**
  - **比較演算子**
  - **論理演算子**
  - ***in* 演算子**

6.2.1 使用できる演算子

条件を作成するときに使用できる演算子にはいくつかの種類があります。

表 1　比較演算子

演算子	説明（ true のとき）
X == Y	X の値と Y の値が等しいとき
X != Y	X の値と Y の値が等しくないとき
X > Y	X の値が Y の値より大きいとき
X >= Y	X の値が Y の値より大きいか等しいとき
X < Y	X の値が Y の値より小さいとき
X <= Y	X の値と Y の値より小さいか等しいとき

表 2　論理演算子（優先順位が高い順）

演算子	説明（ true のとき）
not 条件 X	条件 X が False のとき
条件 X and 条件 Y	条件 X と条件 Y がともに True のとき
条件 X or 条件 Y	条件 X か条件 Y のどちらかが True のとき

表 3　in 演算子

演算子	説明（ true のとき）
A in [X, Y, Z]	A と同じ値が X, Y, Z の中にあるとき
A not in [X, Y, Z]	A と同じ値が X, Y, Z の中にないとき

ご用心:

- 論理演算子を使用して複数の条件を結合するとき、論理演算子の優先（評価）順序を踏まえて結合します。
- () を使用して明示的に条件の評価順序を指定できます。

6.2.2 when ディレクティブ

when ディレクティブを使用してタスクに条件（条件式、条件文）を設定します。

注釈: 条件式に変数を含む場合、その変数は{{}}でくくりません。

比較演算子

比較演算子は 2 つの値を比較するときに使用します。例えば、CentOS の管理対象ホストだけをシャットダウンしたいのであれば**ファクト変数**の ansible_distribution の値が CentOS で判断できます。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS だけシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] == "CentOS"
```

実行前の管理対象ホストの状態です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   running (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

実行ログです。when ディレクティブに指定した条件に合致する管理対象ホストは changed です。合致しない管理対象ホストはタスクの実行をスキップしたので skipping です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts2.yml shutdown.yml

PLAY [特定の OS の管理対象ホストをシャットダウンする]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [debian10]
ok: [centos7]
ok: [ubuntu18]
ok: [centos8]

TASK [CentOS だけシャットダウンする]␣
↪*********************************************************************************************
skipping: [debian10]
```

(次のページに続く)

(前のページからの続き)

```
skipping: [ubuntu18]
[WARNING]: Module invocation had junk after the JSON data:   Broadcast message from␣
↪root@centos7 (Sun 2020-05-03 14:05:07 UTC):    The system is going down for power-
↪off at Sun 2020-05-03 14:06:07 UTC!
changed: [centos7]
changed: [centos8]

PLAY RECAP␣
↪*********************************************************************************************************
centos7                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
centos8                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
debian10                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
ubuntu18                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

実行後の管理対象ホストの状態です。実行ログで changed になった管理対象ホストが poweroff です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   poweroff (virtualbox)
centos8                   poweroff (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

論理演算子

論理演算子は 2 つ以上の条件を結合した少し複雑な条件を作成するときに使用します。論理演算子を用いた条件は優先順位にもとづいて評価順序が決定します。() を用いて条件をグループ化した場合、グループの評価が優先されます。

or 演算子

CentOS か Debian の管理対象ホストをシャットダウンするプレイです。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS か Debian だけシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] == "CentOS" or ansible_facts['distribution']␣
↪== "Debian"
```

when ディレクティブに指定した条件は次のように書き直せます。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS か Debian だけシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] == "CentOS"
       or ansible_facts['distribution'] == "Debian"
```

実行前の管理対象ホストの状態です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   running (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

実行ログです。 CentOS と Debian の管理対象ホストが when ディレクティブに指定した条件に合致したので changed です。Ubuntu の管理対象ホストは条件に合致せずタスクの実行をスキップしたので skipping です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts2.yml shutdown.yml

PLAY [特定の OS の管理対象ホストをシャットダウンする]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
```

(次のページに続く)

(前のページからの続き)

```
ok: [debian10]
ok: [centos7]
ok: [ubuntu18]
ok: [centos8]

TASK [CentOS か Debian だけシャットダウンする]␣
↪*********************************************************************************************
skipping: [ubuntu18]
changed: [debian10]
[WARNING]: Module invocation had junk after the JSON data:   Broadcast message from␣
↪root@centos7 (Mon 2020-05-04 01:00:44 UTC):    The system is going down for power-
↪off at Mon 2020-05-04 01:01:44 UTC!
changed: [centos7]
changed: [centos8]

PLAY RECAP␣
↪*********************************************************************************************
centos7                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
centos8                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
debian10                   : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
ubuntu18                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

実行後の管理対象ホストの状態です。実行ログと同じ結果です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   poweroff (virtualbox)
centos8                   poweroff (virtualbox)
debian10                  poweroff (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

and 演算子

CentOS8 の管理対象ホストをシャットダウンするプレイです。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS8 だけシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] == "CentOS" and ansible_facts['distribution_
↪major_version'] == "8"
```

when ディレクティブに指定した論理演算子がすべて and の場合、条件をリスト形式で書き直せます。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS8 だけシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when:
      - ansible_facts['distribution'] == "CentOS"
      - ansible_facts['distribution_major_version'] == "8"
```

実行前の管理対象ホストの状態です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   running (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

実行ログです。 CentOS8 の管理対象ホストが when ディレクティブに指定した条件に合致したので changed です。 CentOS8 以外の管理対象ホストは条件に合致せずタスクの実行をスキップしたので skipping です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts2.yml shutdown.yml

PLAY [特定の OS の管理対象ホストをシャットダウンする]␣
↪*********************************************************************************************

```

(次のページに続く)

(前のページからの続き)

```
TASK [Gathering Facts]␣
↪*********************************************************************************************************
ok: [debian10]
ok: [centos7]
ok: [ubuntu18]
ok: [centos8]

TASK [CentOS8 だけシャットダウンする]␣
↪*********************************************************************************************************
skipping: [centos7]
skipping: [debian10]
skipping: [ubuntu18]
changed: [centos8]

PLAY RECAP␣
↪*********************************************************************************************************
centos7                    : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
centos8                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
debian10                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
ubuntu18                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

実行後の管理対象ホストの状態です。実行ログと同じ結果です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   poweroff (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

論理演算子の応用

少し複雑な例として or と and　を併用した例を示します。CentOS8 か Ubuntu の管理対象ホストをシャットダウンするプレイです。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS8 か Ubuntu をシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] == "CentOS" and ansible_facts['distribution_
↪major_version'] == "8" or ansible_facts['distribution'] == "Ubuntu"
```

when ディレクティブ内の変数 ansible_facts['distribution_major_version'] が CentOS 用なのか Ubuntu 用なのかがわかりにく状態です。これをディストリビューションごとに () でくくって書き直した結果です。どの条件がどのディストリビューションのものか明確になり、誤読する可能性が低下しました。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS8 か Ubuntu をシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: (ansible_facts['distribution'] == "CentOS" and ansible_facts['distribution_
↪major_version'] == "8")
       or (ansible_facts['distribution'] == "Ubuntu")
```

実行前の管理対象ホストの状態です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   running (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

実行ログです。 CentOS8 と Ubuntu の管理対象ホストが when ディレクティブに指定した条件に合致したので changed です。それ以外の管理対象ホストは条件に合致せずタスクの実行をスキップしたので skipping です。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts2.yml shutdown.yml

PLAY [特定の OS の管理対象ホストをシャットダウンする]␣
↪*********************************************************************** (次のページに続く) ***********
```

(前のページからの続き)

```
TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [debian10]
ok: [centos7]
ok: [ubuntu18]
ok: [centos8]

TASK [CentOS8 か Ubuntu をシャットダウンする]␣
↪*********************************************************************************************
skipping: [centos7]
skipping: [debian10]
changed: [ubuntu18]
changed: [centos8]

PLAY RECAP␣
↪*********************************************************************************************
centos7                    : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
centos8                    : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
debian10                   : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
ubuntu18                   : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

実行後の管理対象ホストの状態です。実行ログと同じ結果です。

```
C:\vagrant\ansible-study2>vagrant status
Current machine states:

centos7                   running (virtualbox)
centos8                   poweroff (virtualbox)
debian10                  running (virtualbox)
ubuntu18                  poweroff (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\vagrant\ansible-study2>
```

in 演算子

or **演算子**で使用した CentOS か Debian の管理対象ホストをシャットダウンするプレイを in 演算子で書き直した

結果です。

```
- name: 特定の OS の管理対象ホストをシャットダウンする
  hosts: all

  tasks:
  - name: CentOS か Debian をシャットダウンする
    command: /sbin/shutdown -t 15
    become: yes
    when: ansible_facts['distribution'] in ["CentOS", "Debian"]
```

in 演算子は in の左側に書いた値が [] の中に書いた値と同じものがあるかを判断します（ not in の場合はないかを判断します)。今回のような使い方の場合、 or 演算子を使用するより in 演算子を使用した方が条件の意味がわかりやすくなります。

6.3 ハンドラー

タスクを実行した結果、管理対象ホストの状態が変更されたときだけ追加のタスクを実行したいときがあります。例えば、特定のパッケージをインストールや更新した後に、そのパッケージに関するサービスの再起動が必要になるときです。このようなときにハンドラーを使用します。ハンドラーは notify ディレクティブと handlers セクションの 2 つで構成します。

notify ディレクティブ

- 追加タスクが必要なタスクに設定します。
- 上記の場合、パッケージをインストール／更新するタスクに設定します。

handlers セクション

- 追加のタスクを定義します。
- 上記の場合、サービスを再起動するタスクを記述します。

次のインベントリの web グループに含まれる管理対象ホストに最新版の Apache をインストールします。もしすでに Apache がインストール済みであれば最新版に更新します。インストール／更新後は Apache のサービスを再起動します。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
```

(次のページに続く)

(前のページからの続き)

```
  children:
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

プレイです。 Apache のパッケージをインストール／更新する yum モジュールを使用したタスクに notify ディレクティブを指定し、handlers セクションにサービスを再起動するタスクを記述します。

```
---
- name: Apache server installed or updated
  hosts: all
  become: yes
  gather_facts: no

  tasks:
  - name: latest version Apache installed
    yum:
      name: httpd
      state: latest
    when: inventory_hostname in groups['web']
    notify: httpd service restarted
  - name: Process end message
    debug:
      msg: "{{ inventory_hostname }} is finished."

  handlers:
  - name: Apache service restarted
    systemd:
      name: httpd.service
      state: restarted
      enabled: yes
    listen: httpd service restarted
```

notify を設定したタスクと handlers セクションのタスクは notify ディレクティブで設定した文字列と handlers セクション内のタスクの listen ディレクティブで設定した文字列が合致するタスクが対応します。

実行ログです。 web グループに含まれる管理対象ホスト node2 と node3 は notify ディレクティブを設定したタスクの実行結果が changed になり、handlers セクション内のタスクも実行しました。 node1 は notify ディレクティブを設定したタスクの実行をスキップ（ skipping ）したので handlers セクションのタスクは実行しません。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Apache server installed or updated]␣
↪*********************************************************************************************

TASK [latest version Apache installed]␣
↪*********************************************************************************************
skipping: [node1]
changed: [node3]
changed: [node2]

TASK [Process end message]␣
↪*********************************************************************************************
ok: [node1] => {
    "msg": "node1 is finished."
}
ok: [node3] => {
    "msg": "node3 is finished."
}
ok: [node2] => {
    "msg": "node2 is finished."
}

RUNNING HANDLER [Apache service restarted]␣
↪*********************************************************************************************
changed: [node2]
changed: [node3]

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
node2                      : ok=3    changed=2    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=3    changed=2    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

実行ログからわかるように handlers セクションのタスクは tasks セクションの実行が終了した後に実行します。notify ディレクティブを設定したタスクが changed になった直後に実行しないため注意が必要です。

> **ご用心:** tasks セクション内のタスクが handlers セクション内のタスクを何回呼び出しても、handlers セクション内のタスクは 1 回だけ実行されます。

再度プレイを実行したときの実行ログです。 web グループの管理対象ホストの Apache は最新版になっているた

め実行結果は ok です。実行結果が changed ではないので handlers セクションのタスクは実行しません。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Apache server installed or updated]␣
↪*********************************************************************************************

TASK [latest version Apache installed]␣
↪*********************************************************************************************
skipping: [node1]
ok: [node2]
ok: [node3]

TASK [Process end message]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "node3 is finished."
}
ok: [node1] => {
    "msg": "node1 is finished."
}
ok: [node2] => {
    "msg": "node2 is finished."
}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=1    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
node2                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

6.4 ループ

【トピックス】

- 単純なループ
- ハッシュのループ

6.4.1 単純なループ

ループを使用しすると同じタスクを繰り返し実行できます。例えば、複数のユーザーアカウントを作成するとき人数分のタスクの実行が必要です。

```
---
- name: 3 ユーザーを追加
  hosts: node1
  become: yes

  tasks:
  - name: ユーザーの追加 1
    user:
      name: taro
      state: present
  - name: ユーザーの追加 2
    user:
      name: jiro
      state: present
  - name: ユーザーの追加 3
    user:
      name: hanako
      state: present
```

実行ログです。ユーザーを追加するタスク [ユーザーの追加 1]、 [ユーザーの追加 2]、 [ユーザーの追加 3] の 3 つのタスクが実行されました。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [3 ユーザーを追加]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node1]

TASK [ユーザーの追加 1]␣
↪*********************************************************************************************
changed: [node1]

TASK [ユーザーの追加 2]␣
↪*********************************************************************************************
changed: [node1]

TASK [ユーザーの追加 3]␣
↪*********************************************************************************************
changed: [node1]
```

(次のページに続く)

(前のページからの続き)

```
PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=4    changed=3    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

これをループを使用して書き直します。

```
---
- name: 3 ユーザーを追加
  hosts: node1
  become: yes

  tasks:
  - name: ユーザーの追加
    user:
      name: "{{ item }}"
      state: present
    loop:
      - taro
      - jiro
      - hanako
```

ループのルールです。

- loop ディレクティブを使用してループを定義します。
- loop ディレクティブを使用すると変数 item が用意されます。変数 item の定義は不要です。
- loop ディレクティブに列挙した値を順番に 1 つ取り出し、変数 item にセットしてタスクを実行します。この動作を loop ディレクティブに列挙した値の数だけ繰り返します。

実行ログです。タスク [ユーザーの追加] の実行結果内に changed が 3 行出力されました。これは、[ユーザーの追加] を 3 回繰り返し実行したということです。実行ログではループのたびに変数 item に設定された値を確認できます。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [3 ユーザーを追加]␣
↪*********************************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************************
ok: [node1]

TASK [ユーザーの追加]␣
↪*********************************************************************************************************
```

(次のページに続く)

(前のページからの続き)

```
changed: [node1] => (item=taro)
changed: [node1] => (item=jiro)
changed: [node1] => (item=hanako)

PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

アドホックコマンドで管理対象ホスト node1 の/etc/passwd ファイルの内容を確認した結果です。loop ディレクティブに列挙した値のアカウントが作成されました。

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m command -a 'cat /etc/
↪passwd'
node1 | CHANGED | rc=0 >>
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
vboxadd:x:997:1::/var/run/vboxadd:/bin/false
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/
↪sbin/nologin
taro:x:1001:1001::/home/taro:/bin/bash
jiro:x:1002:1002::/home/jiro:/bin/bash
hanako:x:1003:1003::/home/hanako:/bin/bash
[vagrant@ansible ansible-files]$
```

6.4.2 ハッシュのループ

ハッシュもループで扱えます。「**単純なループ**」はユーザー名だけでしたが、今回はパスワードも併せてハッシュで定義します。

```
- username: taro
  password: taropass
- username: jiro
  password: jiropass
- username: hanako
  password: hanakopass
```

上記のデータがどの様に変数 item に設定されるか確認します。

```
---
- name: 3 ユーザーを追加
  hosts: node1
  become: yes

  tasks:
  - name: 変数 item の確認
    debug:
      var: item
    loop:
      - username: taro
        password: taropass
      - username: jiro
        password: jiropass
      - username: hanako
        password: hanakopass
```

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [3 ユーザーを追加]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node1]

TASK [変数 item の確認]␣
↪*********************************************************************************************
ok: [node1] => (item={u'username': u'taro', u'password': u'taropass'}) => {
    "ansible_loop_var": "item",
    "item": {
        "password": "taropass",
        "username": "taro"
    }
```

(次のページに続く)

(前のページからの続き)

```
}
ok: [node1] => (item={u'username': u'jiro', u'password': u'jiropass'}) => {
    "ansible_loop_var": "item",
    "item": {
        "password": "jiropass",
        "username": "jiro"
    }
}
ok: [node1] => (item={u'username': u'hanako', u'password': u'hanakopass'}) => {
    "ansible_loop_var": "item",
    "item": {
        "password": "hanakopass",
        "username": "hanako"
    }
}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

変数 item の中に username と password が含まれるため、次のように参照します。

item[' キー名']

具体的には各値に次のように参照します。

item['username']

item['password']

ユーザーアカウントを作成するプレイです。

```
---
- name: 3 ユーザーを追加
  hosts: node1
  become: yes

  tasks:
  - name: ユーザーを追加
    user:
      name: "{{ item['username'] }}"
      password: "{{ item['password'] | password_hash('sha512') }}"
```

(次のページに続く)

(前のページからの続き)

```
      state: present
    loop:
      - username: taro
        password: taropass
      - username: jiro
        password: jiropass
      - username: hanako
        password: hanakopass
```

実行ログです。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [3 ユーザーを追加]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node1]

TASK [ユーザーを追加]␣
↪*********************************************************************************************
changed: [node1] => (item={u'username': u'taro', u'password': u'taropass'})
changed: [node1] => (item={u'username': u'jiro', u'password': u'jiropass'})
changed: [node1] => (item={u'username': u'hanako', u'password': u'hanakopass'})

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

アドホックコマンドで管理対象ホスト node1 の/etc/passwd ファイルと/etc/shadow ファイルの内容を確認した結果です。loop ディレクティブに列挙したアカウント情報が追加されました。

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m command -a 'cat /etc/
↪passwd'
node1 | CHANGED | rc=0 >>
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

(次のページに続く)

(前のページからの続き)

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
vboxadd:x:997:1::/var/run/vboxadd:/bin/false
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/
↪sbin/nologin
taro:x:1001:1001::/home/taro:/bin/bash
jiro:x:1002:1002::/home/jiro:/bin/bash
hanako:x:1003:1003::/home/hanako:/bin/bash
[vagrant@ansible ansible-files]$
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m command -a 'cat /etc/
↪shadow' -b
node1 | CHANGED | rc=0 >>
root:$1$QDyPlph/$oaAX/xNRf3aiW3l27NIUA/::0:99999:7:::
bin:*:17834:0:99999:7:::
daemon:*:17834:0:99999:7:::
adm:*:17834:0:99999:7:::
lp:*:17834:0:99999:7:::
sync:*:17834:0:99999:7:::
shutdown:*:17834:0:99999:7:::
halt:*:17834:0:99999:7:::
mail:*:17834:0:99999:7:::
operator:*:17834:0:99999:7:::
games:*:17834:0:99999:7:::
ftp:*:17834:0:99999:7:::
nobody:*:17834:0:99999:7:::
systemd-network:!!:18048::::::
dbus:!!:18048::::::
polkitd:!!:18048::::::
rpc:!!:18048:0:99999:7:::
rpcuser:!!:18048::::::
nfsnobody:!!:18048::::::
sshd:!!:18048::::::
postfix:!!:18048::::::
chrony:!!:18048::::::
vagrant:$1$C93uBBDg$pqzqtS3a9llsERlv..YKs1::0:99999:7:::
```

(次のページに続く)

(前のページからの続き)

```
vboxadd:!!:18387::::::
tss:!!:18387::::::
taro:$6$En5PNzmKmeHQDU26$q0NAYS182q6NHjCb73sBL61x9565Q0j3xjhEUqX71XRI93j/
 ↪ZCJ1LJu1GyH2TNg3CzODlB6Xc5Wqg9DvbNHDu/:18387:0:99999:7:::
jiro:$6$848sd124X0fBVWfj$YDAvHmp7KYYtq0IhTRFYAy3cp/1G/801DGs.T5JN.
 ↪MdZfpPjNLbhMYZ5qMRQuQE/38tRbIbRSWiRMEoV58jQE.:18387:0:99999:7:::
hanako:$6$dcLjKEy0Yinxpbo8$iznnNfrkvhr92c9mencHIG07iuONY2BgGnqwJnnE/
 ↪s8fhb1vDITXqv6TDOHlAIiDv50ftm/3bcqMJbpR8E.NM0:18387:0:99999:7:::
[vagrant@ansible ansible-files]$
```

第 7 章

テンプレート

Ansible で Jinja2 テンプレートエンジンを使用すると、Ansible サーバーから管理対象ホストに変数を埋め込んだファイルをコピーできます。埋め込まれた変数は管理対象ホストにコピーするときに展開されます。

ちなみに: テンプレート（template）は、文書などのコンピュータデータを作成する上で雛形となるデータ。

最も抽象的なテンプレートは、レイアウトのみのデータで、テキストを流し込むことでレイアウトつき文書となる。具象的なテンプレートは、それ自体文書であり、数箇所の修正または空白への書き込みで目的の文書となる。このほか、さまざまな段階がある。

テンプレートから文書を作る工程は、手動の場合も自動の場合もある。

出典：ウィキペディア　-　テンプレート (https://ja.wikipedia.org/wiki/テンプレート)

管理対象ホストにログインしたときのメッセージを表示する`/etc/motd` ファイルに値を設定するケースで説明します。

`/etc/motd` ファイルに何も設定されていないときのログインの状態です。

```
[vagrant@ansible ansible-files]$ ssh 192.168.1.161 -l vagrant -i ~/.ssh/node1_key
Last login: Tue May  5 21:17:17 2020 from 192.168.1.151
[vagrant@node1 ~]$ ls -l /etc/motd
-rw-r--r--. 1 root root 0 Jun  7  2013 /etc/motd
[vagrant@node1 ~]$
```

`/etc/motd` ファイルに設定する値を motd-facts.j2 ファイル（テンプレートファイル）に記述します。テンプレートファイルの拡張子は`.j2` です。

```
Welcome to {{ ansible_facts['hostname'] }}.
{{ ansible_facts['distribution'] }} {{ ansible_facts['distribution_version'] }}
deployed on {{ ansible_facts['architecture'] }} architecture.
```

このテンプレートファイル motd_facts.yml を管理対象ホストにコピーします。コピー時に変数（**ファクト変数**）が具体的な値に置き換わります。

```
---
- name: Fill motd file with host data
  hosts: node1

  tasks:
  - name: Copy the template file "motd-facts.j2" to managed node
    template:
      src: motd-facts.j2
      dest: /etc/motd
      owner: root
      group: root
      mode: 0644
    become: yes
```

実行ログです。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml motd-facts.yml

PLAY [Fill motd file with host data]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node1]

TASK [Copy the template file "motd-facts.j2" to managed node]␣
↪*********************************************************************************************
changed: [node1]

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

管理対象ホスト node1 にログインし、コピーしたメッセージが表示されるか確認します。

```
[vagrant@ansible ansible-files]$ ssh 192.168.1.161 -l vagrant -i ~/.ssh/node1_key
Last login: Wed May  6 08:35:51 2020 from 192.168.1.151
Welcome to node1.
CentOS 7.8
deployed on x86_64 architecture.
[vagrant@node1 ~]$ ls -l /etc/motd
-rw-r--r--. 1 root root 62 May  6 08:35 /etc/motd
```

(次のページに続く)

(前のページからの続き)

```
[vagrant@node1 ~]$ cat /etc/motd
Welcome to node1.
CentOS 7.8
deployed on x86_64 architecture.
[vagrant@node1 ~]$
```

ログイン時に表示されたメッセージからテンプレートファイル内の**ファクト変数**を展開しコピーしたことがわかります。

第 8 章

ロール

【トピックス】

- ロールとは
- ディレクトリ構造
- ロールの作成

8.1 ロールとは

ここまでの例はプレイを 1 つのプレイブックファイルに書いてきました。この方法ではプレイの内容が長くなり、どこに何が書いてあるのかを探すのが難しくなります。このような状態ではメンテナンスもしづらく、ミスも発生しやすくなります。

この問題の解決策です。

- タスクを機能単位のグループにまとめてプレイブックファイルと別のファイルに保存します。
- タスクを機能単位のグループにまとめたものをロールと呼びます。
- プレイから必要なロールを呼び出し実行します。

ロールは特定の機能だけを持つようにします。複数の機能を持たせると、長いプレイと同じ問題が発生します。特定の機能だけを持つロールは、他のプレイからも部品として呼び出す　＝　再利用できます。再利用の頻度が高いほど、部品としての精度向上が期待できます。

ちなみに: 「ロール」はロールプレイ（役を演じる）のロールと同じ意味です。

8.2 ディレクトリ構造

ロールは roles/ディレクトリ内に配置します。以下は Ansible の Best Practices で提示されているディレクトリレイアウトです。網掛けがロールに関する部分です。

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them␣
↪here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the␣
↪webtier role
    monitoring/           # ""
```

(次のページに続く)

(前のページからの続き)

```
    fooapp/               # ""
```

roles/ディレクトリ直下のディレクトリの名前がそのままロールの名前（ロール名）になります。common、webtire、monitoring、fooapp がロール名です。

ロール内のディレクトリ構成やファイル名などは基本的に共通です。

```
tasks/            #
    main.yml      #  <-- tasks file can include smaller files if warranted
handlers/         #
    main.yml      #  <-- handlers file
templates/        #  <-- files for use with the template resource
    ntp.conf.j2   #  <------- templates end in .j2
files/            #
    bar.txt       #  <-- files for use with the copy resource
    foo.sh        #  <-- script files for use with the script resource
vars/             #
    main.yml      #  <-- variables associated with this role
defaults/         #
    main.yml      #  <-- default lower priority variables for this role
meta/             #
    main.yml      #  <-- role dependencies
library/          # roles can also include custom modules
module_utils/     # roles can also include custom module_utils
lookup_plugins/   # or other types of plugins, like lookup in this case
```

この中で tasks/ディレクトリおよびその中の main.yml だけが必須です。他のディレクトリやファイルは必要に応じて作成・配置します。主なディレクトリの内容です。

tasks ロールで実行するタスクを main.yml ファイルに記述する。

handlers tasks/ディレクトリの main.yml 内のタスクの notify ディレクティブで呼び出される handlers セクション内のタスクを main.yml ファイルに記述する。

templates template モジュールで使用する jinja2 形式のテンプレートファイルを配置する。

files copy モジュールなどで使用するファイルを配置する。

vars ロール内で使用する変数を main.yml ファイルに記述する。

注釈: ロール内の templates/ディレクトリや files/ディレクトリ内のファイルを指定するときパスの指定は不要です。直接ファイル名を指定します。

8.3 ロールの作成

web グループの管理対象ホスト用に次のプレイを作成します。

1. Apache の最新版をインストールする。すでに Apache がインストール済みのときは最新版に更新する。
2. ファイアウォールの http ポートを開放する。
3. Apache の最新版をインストールまたは更新したときは Apache のサービスを再起動する。
4. バーチャルホスト用の設定ファイルをコピーする。
5. バーチャルホストのコンテンツ用のディレクトリ `/var/www/vhost_html` を作成する。
6. 元ドメイン用の `index.html` ファイルをコピーする。
7. バーチャルホスト用の `index.html` ファイルをコピーする。

8.3.1 インベントリファイル

インベントリファイル hosts.yml の内容です。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
  children:
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

8.3.2 必要なディレクトリの作成

要件にあわせて必要なディレクトリを作成します。

- ロール名：apache
- ロール apache 内に必要なディレクトリ：`tasks`、`handlers`、`templates`

```
[vagrant@ansible ansible-files]$ mkdir roles/
[vagrant@ansible ansible-files]$ cd roles/
[vagrant@ansible roles]$ mkdir apache/
[vagrant@ansible roles]$ cd apache/
[vagrant@ansible apache]$ mkdir tasks/
[vagrant@ansible apache]$ mkdir handlers/
[vagrant@ansible apache]$ mkdir templates/
[vagrant@ansible apache]$
```

ディレクトリ作成後の状態です。

```
[vagrant@ansible ansible-files]$ tree
.
├── hosts.yml
└── roles
    └── apache
        ├── handlers
        ├── tasks
        └── templates

5 directories, 1 file
[vagrant@ansible ansible-files]$
```

8.3.3 テンプレートファイルの作成

templates ディレクトリ内に管理対象ホストにコピーするテンプレートファイル（ index.html ）を作成します。

用途	ファイル名
親ドメイン	default-index.j2
バーチャルドメイン	virtual-index.j2

ファイル作成後の状態です。

```
[vagrant@ansible ansible-files]$ tree
.
├── hosts.yml
└── roles
    └── apache
        ├── handlers
        ├── tasks
        └── templates
            ├── default-index.j2
            └── virtual-index.j2
```

(次のページに続く)

(前のページからの続き)

```
5 directories, 3 files
[vagrant@ansible ansible-files]$
```

roles/apache/templates/default-index.j2

```
<html>
<body>
This is the page for the parent domain {{ ansible_facts['hostname'] }}.<br>
</body>
</html>
```

roles/apache/templates/virtual-index.j2

```
<html>
<body>
Virtual Domain : {{ ansible_facts['hostname'] }}<br>
</body>
</html>
```

8.3.4 バーチャルホスト用設定ファイルの作成

templates ディレクトリ内にバーチャルホスト用の設定ファイルを作成します。

ファイル作成後の状態です。

```
[vagrant@ansible ansible-files]$ tree
.
├── hosts.yml
└── roles
    └── apache
        ├── handlers
        ├── tasks
        └── templates
            ├── default-index.j2
            ├── vhost.j2
            └── virtual-index.j2

5 directories, 4 files
[vagrant@ansible ansible-files]$
```

roles/apache/templates/vhost.j2

```
<VirtualHost *:80>
  DocumentRoot /var/www/html
  ServerName {{ ansible_facts['hostname'] }}
</VirtualHost>

<VirtualHost *:80>
  DocumentRoot /var/www/vhost_html
  ServerName virtual-{{ ansible_facts['hostname'] }}
</VirtualHost>
```

8.3.5 タスクの作成

まず、メインとなるプレイブックファイル apache.yml を作成します。`import_role` モジュールを使用してロールを呼び出します。

```
---
- name: Deploy web server
  hosts: web
  become: yes

  tasks:
  - name: Display start message
    debug:
      msg: "Beginning web server configuration."
  - name: Deploy Apache Server
    import_role:
      name: apache
  - name: Display end message
    debug:
      msg: "Web server has been configured."
```

次にロール apache のタスクを `roles/apache/tasks/main.yml` ファイルに記述します。

```
1  ---
2  - name: Install/Update Apache
3    yum:
4      name: httpd
5      state: latest
6    notify: restart httpd service
7  - name: Open http port
8    firewalld:
9      service: http
10     permanent: yes
11     state: enabled
```

(次のページに続く)

(前のページからの続き)

```
12 - name: Copy virtual host configuration
13   template:
14     src: vhost.j2
15     dest: /etc/httpd/conf.d/vhost.conf
16     owner: root
17     group: root
18     mode: 0644
19   notify: restart httpd service
20 - name: Create a directory for the contents of the virtual host
21   file:
22     path: /var/www/vhost_html/
23     state: directory
24 - name: Copy the index.html file for the original domain.
25   template:
26     src: default-index.j2
27     dest: /var/www/html/index.html
28 - name: Copy the index.html file for the virtual host.
29   template:
30     src: virtual-index.j2
31     dest: /var/www/vhost_html/index.html
```

- 2 〜 6 行目　→　Apache の最新版をインストールする。すでに Apache がインストール済みのときは最新版に更新する。
- 7 〜 11 行目　→　ファイアウォールの http ポートを開放する。
- 12 〜 19 行目　→　バーチャルホスト用の設定ファイルをコピーする。
- 20 〜 23 行目　→　バーチャルホストのコンテンツ用のディレクトリ `/var/www/vhost_html` を作成する。
- 24 〜 27 行目　→　元ドメイン用の `index.html` ファイルをコピーする。
- 28 〜 31 行目　→　バーチャルホスト用の `index.html` ファイルをコピーする。

最後に `roles/apache/handlers/main.yml` ファイルに上記のタスクから呼び出される handlers を記述します。

```
---
- name: Apache service restart
  systemd:
    name: httpd.service
    state: restarted
    enabled: yes
  listen: restart httpd service
```

- `roles/apache/handlers/main.yml`→　Apache の最新版をインストールまたは更新したときは Apache のサービスを再起動する。

各ファイルの作成後の状態です。

```
[vagrant@ansible ansible-files]$ tree
.
├── apache.yml
├── hosts.yml
└── roles
    └── apache
        ├── handlers
        │ ^^c2^^a0^^c2^^a0 └── main.yml
        ├── tasks
        │ ^^c2^^a0^^c2^^a0 └── main.yml
        └── templates
            ├── default-index.j2
            ├── vhost.j2
            └── virtual-index.j2

5 directories, 7 files
[vagrant@ansible ansible-files]$
```

実行ログです。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Deploy web server]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [Display start message]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "Beginning web server configuration."
}
ok: [node2] => {
    "msg": "Beginning web server configuration."
}

TASK [apache : Install/Update Apache]␣
↪*********************************************************************************************
changed: [node3]
changed: [node2]

TASK [apache : Open http port]␣
↪*********************************************************************************************
changed: [node3]
```

(次のページに続く)

(前のページからの続き)

```
changed: [node2]

TASK [apache : Copy virtual host configuration]␣
↪*********************************************************************************************
changed: [node2]
changed: [node3]

TASK [apache : Create a directory for the contents of the virtual host]␣
↪*****************************************************************************
changed: [node2]
changed: [node3]

TASK [apache : Copy the index.html file for the original domain.]␣
↪*******************************************************************************************
changed: [node3]
changed: [node2]

TASK [apache : Copy the index.html file for the virtual host.]␣
↪*******************************************************************************************
changed: [node3]
changed: [node2]

TASK [Display end message]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "Web server has been configured."
}
ok: [node2] => {
    "msg": "Web server has been configured."
}

RUNNING HANDLER [apache : Apache service restart]␣
↪*********************************************************************************************
changed: [node2]
changed: [node3]

PLAY RECAP␣
↪*********************************************************************************************
node2                      : ok=10   changed=7    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=10   changed=7    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

べき等性が保たれているか確認するため、もう一度実行したときのログです。すべて ok になっており、べき等性が保たれています。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml apache.yml

PLAY [Deploy web server]␣
↪*********************************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [Display start message]␣
↪*********************************************************************************************
ok: [node3] => {
    "msg": "Beginning web server configuration."
}
ok: [node2] => {
    "msg": "Beginning web server configuration."
}

TASK [apache : Install/Update Apache]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [apache : Open http port]␣
↪*********************************************************************************************
ok: [node2]
ok: [node3]

TASK [apache : Copy virtual host configuration]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [apache : Create a directory for the contents of the virtual host]␣
↪*********************************************************************************
ok: [node3]
ok: [node2]

TASK [apache : Copy the index.html file for the original domain.]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]

TASK [apache : Copy the index.html file for the virtual host.]␣
↪*********************************************************************************************
ok: [node3]
ok: [node2]
```

(次のページに続く)

(前のページからの続き)

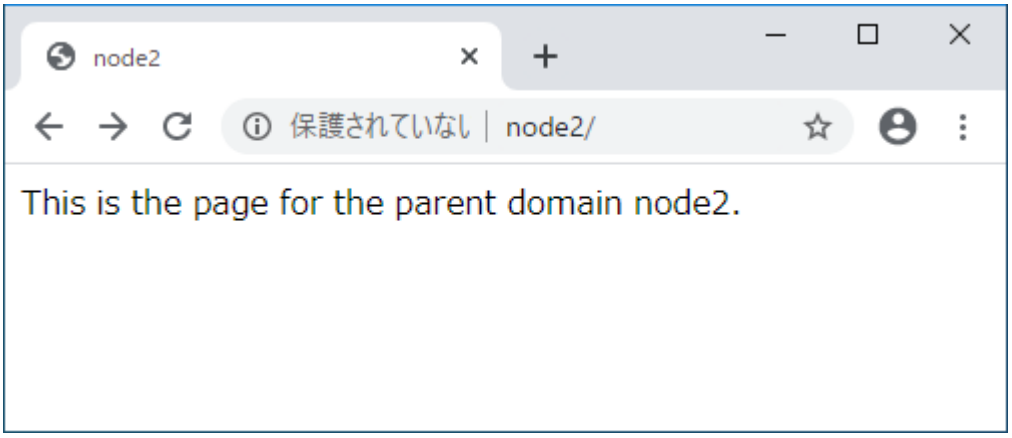
```
TASK [Display end message]␣
↪*********************************************************************************************************
ok: [node3] => {
    "msg": "Web server has been configured."
}
ok: [node2] => {
    "msg": "Web server has been configured."
}

PLAY RECAP␣
↪*********************************************************************************************************
node2                      : ok=9    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=9    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

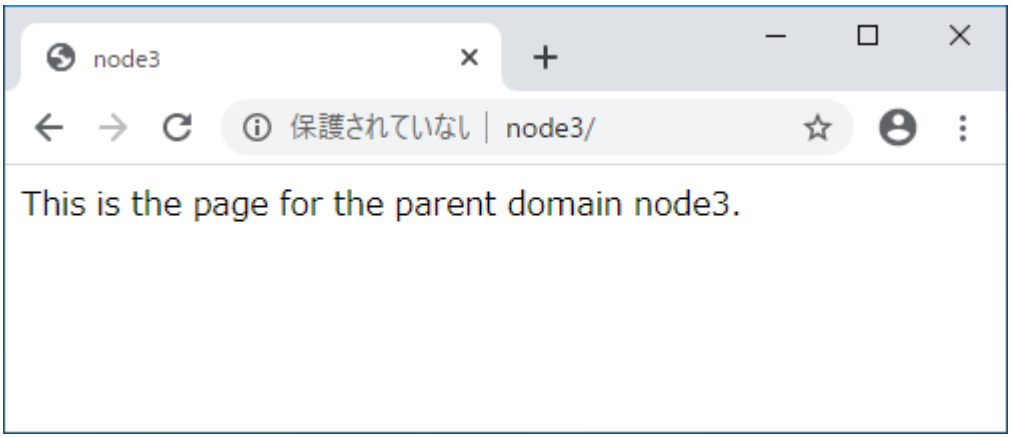
8.3.6 実行確認

名前解決のため C:\Windows\System32\drivers\etc\hosts ファイルに次の内容を追記します。

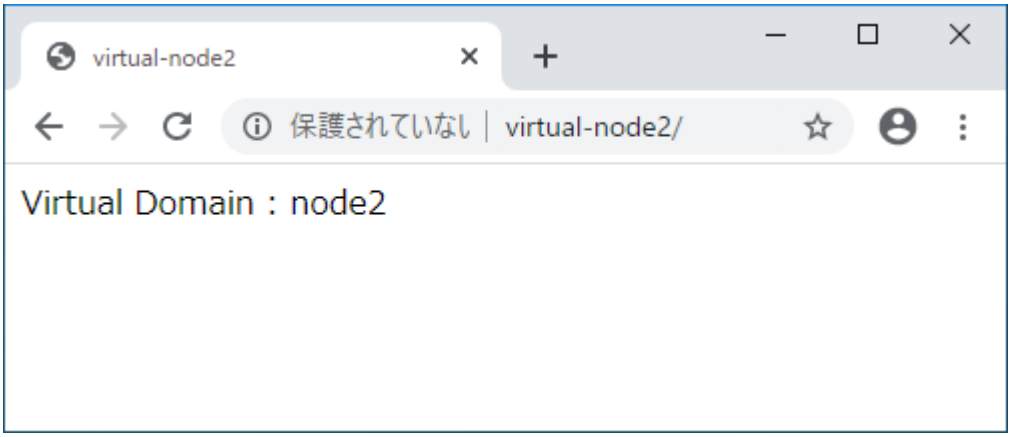
```
192.168.1.162       node2
192.168.1.162       virtual-node2
192.168.1.163       node3
192.168.1.163       virtual-node3
```

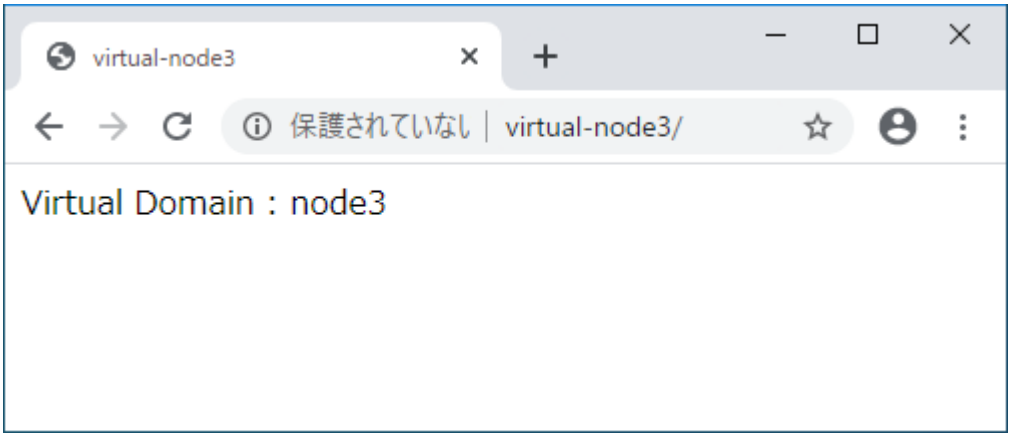
親ドメインのページ表示





バーチャルホストのページ表示





注釈: 実行確認後は C:\Windows\System32\drivers\etc\hosts ファイルに追記した 4 行は削除してください。

第 9 章

変数 その 2

「**変数**」の続きです。変数を少し掘り下げて、「**インベントリの分割**」と「**エラーハンドリング**」について説明します。

について説明します。

9.1 インベントリの分割

アドホックコマンドの「**インベントリの作成**」で使用したインベントは 1 つのインベントリファイルで構成されました。「**ロール**」でプレイを機能単位に分割したように、インベントリもその内容で複数のファイルに分割し、所定のディレクトリに配置できます。分割したファイルは**ロール**と同じように、他のインベントリの部品として使用できます。

【トピックス】

- **ディレクトリ構造**
- ***group_vars* ディレクトリと *host_vars* ディレクトリ**
- ***group_vars* ディレクトリ**
- ***host_vars* ディレクトリ**
- **インベントリの切り出し後の確認**
- **インベントリの演習**

9.1.1 ディレクトリ構造

以下は Ansible の Best Practices で提示されているディレクトリレイアウトです。網掛けがインベントリに関する部分です。

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them␣
↪here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the␣
↪webtier role
    monitoring/           # ""
    fooapp/               # ""
```

インベントリに関する部分の抜粋です。

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml
```

production ファイルと staging ファイルがインベントリファイルです。このワークブックは hosts.yml ファイルを使用しています。

9.1.2 group_vars ディレクトリと host_vars ディレクトリ

インベントリファイルの分割後のファイルは group_vars ディレクトリと host_vars ディレクトリに配置します。

group_vars ディレクトリ

- グループ全体に適用する変数を定義します。
- 定義した変数を「グループ変数」と呼びます。
- グループ名をファイル名にします。

host_vars ディレクトリ

- ホストだけに適用する（ホスト固有の）変数を定義します。
- 定義した変数を「ホスト変数」と呼びます。
- インベントリ内のホスト名をファイル名にします。

重要: group_vars ディレクトリと host_vars ディレクトリはインベントリファイルと同じディレクトリ内に作成しなければなりません。

下記のインベントリファイル hosts.yml を分割します。

```
---
all:
  vars:
```

(次のページに続く)

(前のページからの続き)

```
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
      stage: prod
  children:
    web:
      vars:
        stage: dev
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

ansible-inventory コマンドで状態を確認します。

```
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --list --yaml
all:
  children:
    ungrouped:
      hosts:
        node1:
          ansible_host: 192.168.1.161
          ansible_ssh_private_key_file: ~/.ssh/node1_key
          ansible_user: vagrant
          stage: prod
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
          ansible_user: vagrant
          stage: dev
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
          ansible_user: vagrant
          stage: dev
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --graph
@all:
  |--@ungrouped:
  |  |--node1
  |--@web:
```

(次のページに続く)

(前のページからの続き)

```
  |   |--node2
  |   |--node3
[vagrant@ansible ansible-files]$
```

9.1.3 group_vars ディレクトリ

group_vars ディレクトリを作成します

```
[vagrant@ansible ansible-files]$ mkdir group_vars/
[vagrant@ansible ansible-files]$
```

インベントリからグループに関する部分を抽出（網掛け部分）します。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
      stage: prod
  children:
    web:
      vars:
        stage: dev
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

最初は all グループ、 2 つめは web グループに関する部分なので、ファイル名は次のとおりです。

- all.yml
- web.yml

`group_vars/all.yml` ファイルの内容です。

```
---
ansible_user: vagrant
```

group_vars/web.yml ファイルの内容です。

```
---
stage: dev
```

インベントリファイル hosts.yml からグループの部分を切り出したあとの状態です。

```
---
all:
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
      stage: prod
  children:
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

9.1.4 host_vars ディレクトリ

host_vars ディレクトリを作成します

```
[vagrant@ansible ansible-files]$ mkdir host_vars/
[vagrant@ansible ansible-files]$
```

インベントリからグループに関する部分を抽出（網掛け部分）します。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
      ansible_host: 192.168.1.161
      ansible_ssh_private_key_file: ~/.ssh/node1_key
      stage: prod
  children:
    web:
      vars:
```

(次のページに続く)

(前のページからの続き)

```
        stage: dev
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
```

ホスト名がファイル名になるので、各管理対象ホストのファイル名は次のとおりです。

- node1.yml
- node2.yml
- node3.yml

host_vars/node1.yml ファイルの内容です。

```
---
ansible_host: 192.168.1.161
ansible_ssh_private_key_file: ~/.ssh/node1_key
stage: prod
```

host_vars/node2.yml ファイルの内容です。

```
---
ansible_host: 192.168.1.162
ansible_ssh_private_key_file: ~/.ssh/node2_key
```

host_vars/node3.yml ファイルの内容です。

```
---
ansible_host: 192.168.1.163
ansible_ssh_private_key_file: ~/.ssh/node3_key
```

インベントリファイル hosts.yml から各管理対象ホストの部分を切り出したあとの状態です。

```
---
all:
  vars:
    ansible_user: vagrant
  hosts:
    node1:
  children:
    web:
      vars:
```

(次のページに続く)

(前のページからの続き)

```
        stage: dev
      hosts:
        node2:
        node3:
```

9.1.5 インベントリの切り出し後の確認

インベントリからグループ固有の部分を *group_vars* **ディレクトリ**に、管理対象ホスト固有の部分を *host_vars* **ディレクトリ**に切り出した後の `hosts.yml` ファイルの状態です。

```
---
all:
  hosts:
    node1:
  children:
    web:
      hosts:
        node2:
        node3:
```

グループ構成と管理対象ホストだけのスッキリした状態です。ディレクトリとファイルの配置状況です。

```
[vagrant@ansible ansible-files]$ tree
.
├── group_vars
│ ^^c2^^a0^^c2^^a0 ├── all.yml
│ ^^c2^^a0^^c2^^a0 └── web.yml
├── hosts.yml
└── host_vars
    ├── node1.yml
    ├── node2.yml
    └── node3.yml

2 directories, 6 files
[vagrant@ansible ansible-files]$
```

`ansible-inventory` コマンドを実行し、各ファイルに切り出す前と同じか確認しました。同じ結果になっています。

```
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --list --yaml
all:
  children:
    ungrouped:
```

(次のページに続く)

(前のページからの続き)

```
      hosts:
        node1:
          ansible_host: 192.168.1.161
          ansible_ssh_private_key_file: ~/.ssh/node1_key
          ansible_user: vagrant
          stage: prod
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
          ansible_user: vagrant
          stage: dev
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
          ansible_user: vagrant
          stage: dev
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --graph
@all:
  |--@ungrouped:
  |  |--node1
  |--@web:
  |  |--node2
  |  |--node3
[vagrant@ansible ansible-files]$
```

9.1.6 インベントリの演習

1. インベントリに関するファイルを編集してください。

- `node1.yml` ファイルから変数 stage の行を削除
- `all.yml` ファイルの末尾に削除した行を追加

2. `ansible-inventory` コマンドを実行し、all グループと web グループのどちらの変数の値が優先されるのか確認してください。

解答

編集後の `host_vars/node1.yml` ファイルの内容です。

```
---
ansible_host: 192.168.1.161
ansible_ssh_private_key_file: ~/.ssh/node1_key
```

編集後の group_vars/all.yml ファイルの内容です。

```
---
ansible_user: vagrant
stage: prod
```

ansible-inventory コマンドの実行結果です。

```
[vagrant@ansible ansible-files]$ ansible-inventory -i hosts.yml --list --yaml
all:
  children:
    ungrouped:
      hosts:
        node1:
          ansible_host: 192.168.1.161
          ansible_ssh_private_key_file: ~/.ssh/node1_key
          ansible_user: vagrant
          stage: prod
    web:
      hosts:
        node2:
          ansible_host: 192.168.1.162
          ansible_ssh_private_key_file: ~/.ssh/node2_key
          ansible_user: vagrant
          stage: dev
        node3:
          ansible_host: 192.168.1.163
          ansible_ssh_private_key_file: ~/.ssh/node3_key
          ansible_user: vagrant
          stage: dev
[vagrant@ansible ansible-files]$
```

group_vars/all.yml ファイルの内容がすべてのグループ内の管理対象ホストに適用されてから group_vars/web.yml ファイルの内容が適用されました。この結果から同じ変数が複数のグループで定義されている場合、all グループよりも個別のグループ（今回は web グループ）の定義が優先されます。

9.2 エラーハンドリング

【トピックス】

- *Ansible* でタスクの実行時にエラーが発生したときの動作
- エラーハンドリング

9.2.1 Ansible でタスクの実行時にエラーが発生したときの動作

Ansible Documentation 内の「Modules That Are Useful for Testing」に次の一文があります。

Ansible is a fail-fast system, so when there is an error creating that user, it will stop the playbook run. You do not have to check up behind it.

Google 翻訳の結果です。

「 Ansible はフェイルファストシステムであるため、そのユーザーの作成中にエラーが発生すると、プレイブックの実行が停止します。 背後でチェックする必要はありません。」

少し補足して言い直すと次のようになります。

- ある管理対象ホストでタスクでエラーが発生したら、その管理対象ホストはエラーが発生したタスクで終了です。
- エラーが発生したタスク以降のタスク（＝　後続のタスク）は実行しません。

プレイの実行して動作を確認します。`tasks` セクション内に task-1 、task-2 、task-3 の 3 つのタスクがあります。タスク "task-2" は管理対象ホスト node2 だけにエラーが発生し、他の管理対象ホストは実行をスキップします。

```
- name: task error play
  hosts: all
  gather_facts: no

  tasks:
  - name: task-1
    debug:
  - name: task-2
    command: /bin/false
    when: inventory_hostname == "node2"
  - name: task-3
    debug:
```

実行結果です。管理対象ホスト node2 はタスク "task-2" でにエラーが発生したため、このタスクで終了です。タスク "task-3" は実行しません。 node2 以外の管理対象ホストは task-3 まで実行します。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml task-error.yml

PLAY [task error play]␣
↪*********************************************************************************************

TASK [task-1]␣
↪*********************************************************************************************
ok: [node1] => {
    "msg": "Hello world!"
}
ok: [node3] => {
    "msg": "Hello world!"
}
ok: [node2] => {
    "msg": "Hello world!"
}

TASK [task-2]␣
↪*********************************************************************************************
skipping: [node3]
skipping: [node1]
fatal: [node2]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": ["/bin/false"], "delta": "0:00:00.005796", "end
↪": "2020-05-09 22:03:45.298726", "msg": "non-zero return code", "rc": 1, "start":
↪"2020-05-09 22:03:45.292930", "stderr": "", "stderr_lines": [], "stdout": "",
↪"stdout_lines": []}

TASK [task-3]␣
↪*********************************************************************************************
ok: [node1] => {
    "msg": "Hello world!"
}
ok: [node3] => {
    "msg": "Hello world!"
}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
node2                      : ok=1    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0
node3                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

9.2.2 エラーハンドリング

「エラーハンドリング」とは発生したエラーに対処することです。 Ansible は fail-fast システムなので、エラーが発生したらそこで実行は終了です。べき等性を保つ観点から、多くの場合はこれで問題ありません。しかし、command モジュールや shell モジュールを使用して Linux コマンドを実行するとき、これでは問題が発生することがあります。

Linux コマンドを実行すると、必ず戻り値が設定されます。具体的には次の値が設定されます。

- コマンドが正常終了したとき　→　0 が設定されます
- コマンドが異常終了（エラーが発生した）とき　→　0 以外が設定されます

上述のプレイで使用した/bin/false コマンドの戻り値を確認します。

```
[vagrant@ansible ansible-files]$ /bin/false
[vagrant@ansible ansible-files]$ echo $?
1
[vagrant@ansible ansible-files]$
```

ls コマンドの戻り値を確認します。ファイルが存在したときの戻り値は 0 、ファイルが存在しないときの戻り値は 2 です。

```
[vagrant@ansible ansible-files]$ ls hosts.yml
hosts.yml
[vagrant@ansible ansible-files]$ echo $?
0
[vagrant@ansible ansible-files]$ ls abc
ls: cannot access abc: No such file or directory
[vagrant@ansible ansible-files]$ echo $?
2
[vagrant@ansible ansible-files]$
```

command モジュールや shell モジュールを使用して Linux コマンドを実行したときの戻り値が 0 以外の場合、Ansible はタスクの実行が失敗したと判断し fatal として処理します。これが上述のプレイでエラーが発生した理由です。

9.2.3 レジスタ変数

register ディレクティブで定義するレジスタ変数を使用すると、モジュールの実行結果や戻り値を確認できます。

次のプレイでレジスタ変数の値を確認します。

```
- name: レジスタ変数を確認するプレイ
  hosts: node1
  gather_facts: no

  tasks:
  - name: Linux コマンドを実行
    command: 'ls -a'
    register: result
  - name: レジスタ変数の確認
    debug:
      var: result
```

実行結果です。レジスタ変数 result が表示されています。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml register-variables.yml

PLAY [レジスタ変数を確認するプレイ]␣
↪*********************************************************************************************

TASK [Linux コマンドを実行]␣
↪*********************************************************************************************
changed: [node1]

TASK [レジスタ変数の確認]␣
↪*********************************************************************************************
ok: [node1] => {
    "result": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
        "cmd": [
            "ls",
            "-a"
        ],
        "delta": "0:00:00.007731",
        "end": "2020-05-09 23:03:17.472226",
        "failed": false,
        "rc": 0,
        "start": "2020-05-09 23:03:17.464495",
        "stderr": "",
        "stderr_lines": [],
        "stdout": ".\n..\n.ansible\n.bash_history\n.bash_logout\n.bash_profile\n.
↪bashrc\n.ssh",
        "stdout_lines": [
            ".",
            "..",
            ".ansible",
```

(次のページに続く)

(前のページからの続き)

```
            ".bash_history",
            ".bash_logout",
            ".bash_profile",
            ".bashrc",
            ".ssh"
        ]
    }
}

PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$ cat register-variables.yml
```

レジスタ変数の主な内容です。

changed

- タスクの実行で対象ノードが変更されたか否かの結果が設定されます
  - true : 対象ノードが変更されました
  - false : 対象ノードが変更されませんでした

failed

- 対象ノードでタスクの実行が失敗したか否かの結果が設定されます
  - true : タスクの実行が失敗しました
  - false : タスクの実行が失敗しませんでした

msg

- モジュール実行時のメッセージが設定されます

rc

- return code の略
- タスクの終了ステータス（戻り値）が設定されます
  - 0 : 成功
  - 1 : 失敗（使用するモジュールなどにより 1 以外のこともある）

skipped

- 対象ノードでタスクの実行がスキップされたか否かの結果が設定されます

  - true : タスクの実行がスキップされました
  - false : タスクの実行がスキップされませんでした

stderr

- 標準エラー出力に出力された文字列が設定されます
- 改行は改行文字 n で表され、 1 行で設定されます

stderr_lines

- stderr を 1 行ごとに分割した内容（= stderr の内容をリストで格納したもの）です

stdout

- 標準出力に出力された文字列です
- 改行は改行文字 n で表され、 1 行で設定されます

stdout_lines

- stdout を 1 行ごとに分割した内容（= stdout の内容をリストで格納したもの）です

9.2.4 Ansible のエラーハンドリング

Ansible は次のいずれかの方法でエラーハンドリングします。

- **エラーを無視する**
- **エラーとして扱う条件を設定し、条件に合致したときだけエラーとして扱う**

エラーを無視する

タスクに `ignore_errors` ディレクティブを指定すると、タスクでエラーが発生してもそのエラーを無視します。

`/bin/false` コマンドを実行するタスクに `ignore_errors` ディレクティブと `register` ディレクティブを指定して実行します。

```
- name: task error play
  hosts: all
  gather_facts: no

  tasks:
  - name: task-1
    debug:
  - name: task-2
```

(次のページに続く)

(前のページからの続き)

```
    command: /bin/false
    ignore_errors: yes
    register: result
    when: inventory_hostname == "node2"
  - name: task-3
    debug:
      var: result
```

実行結果です。管理対象ホスト node2 がタスク "task-2" の実行時エラーを無視したので**...ignoring** が表示されました。エラーを無視したので node2 もタスク "task-3" を実行しています。

タスク "task-3" は "task-2" の実行結果の表示です。 node2 に**"failed": true,** や**"rc": 1,** の表示があり、エラーが発生したことがわかります。RECAP に **ignored=1** の表示があり、ここでもエラーを無視したことがわかります。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml task-error.yml

PLAY [task error play]␣
↪*********************************************************************************************

TASK [task-1]␣
↪*********************************************************************************************
ok: [node1] => {
    "msg": "Hello world!"
}
ok: [node3] => {
    "msg": "Hello world!"
}
ok: [node2] => {
    "msg": "Hello world!"
}

TASK [task-2]␣
↪*********************************************************************************************
skipping: [node1]
skipping: [node3]
fatal: [node2]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": ["/bin/false"], "delta": "0:00:00.005399", "end
↪": "2020-05-09 23:40:32.337710", "msg": "non-zero return code", "rc": 1, "start":
↪"2020-05-09 23:40:32.332311", "stderr": "", "stderr_lines": [], "stdout": "",
↪"stdout_lines": []}
...ignoring

TASK [task-3]␣
↪*********************************************************************************************
ok: [node1] => {
    "result": {
        "changed": false,
```

(次のページに続く)

(前のページからの続き)

```
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}
ok: [node3] => {
    "result": {
        "changed": false,
        "skip_reason": "Conditional result was False",
        "skipped": true
    }
}
ok: [node2] => {
    "result": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
        "cmd": [
            "/bin/false"
        ],
        "delta": "0:00:00.005399",
        "end": "2020-05-09 23:40:32.337710",
        "failed": true,
        "msg": "non-zero return code",
        "rc": 1,
        "start": "2020-05-09 23:40:32.332311",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "",
        "stdout_lines": []
    }
}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0
node2                      : ok=3    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=1
node3                      : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=1    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

エラーとして扱う条件を設定し、条件に合致したときだけエラーとして扱う

「**エラーハンドリング**」で説明したとおり、Ansible は戻り値が 0 以外の時にエラーが発生したと判断し処理します。多くの場合、これで問題ありません。しかし、次のような場合に問題が生じます。

「diff コマンドを使用してファイルの内容が処理の前後で差異が発生していることを確認する」

diff コマンドの戻り値です。

- 2 つのファイルの内容に差異がなかったとき　→　0
- 2 つのファイルの内容に差異があったととき　→　1
- ファイルのどちらかまたは両方が存在しないとき　→　2

処理としては差異が発生している状態（＝　戻り値が 1 ）が正しく、それ以外は誤りになります。このようなときは、タスクにレジスタ変数と failed_when ディレクティブを組み合わせて、エラーとして扱う条件を設定します。設定した条件を満たすとき、タスクはエラーとして判断し処理します。

次のプレイで failed_when ディレクティブの動作を確認します。

```
- name: 2 つのファイルを比較する
  hosts: node1
  gather_facts: no

  tasks:
  - name: ファイルの比較
    command: 'diff file-1 file-2'
    register: diff_result
    failed_when: diff_result['rc'] != 1
  - name: レジスタ変数の内容確認
    debug:
      var: diff_result
```

ファイルの内容が異なる場合の実行ログです。本来ならタスク "ファイルの比較" でエラーになり処理が中断されますが、failde_when ディレクティブでエラーの条件を変更しているため最後まで処理できました。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml file-diff.yml

PLAY [2 つのファイルを比較する]␣
↪*********************************************************************************************

TASK [ファイルの比較]␣
↪*********************************************************************************************
changed: [node1]

TASK [レジスタ変数の内容確認]␣
↪*********************************************************************************************
ok: [node1] => {
```

(次のページに続く)

(前のページからの続き)

```
    "diff_result": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
        "cmd": [
            "diff",
            "file-1",
            "file-2"
        ],
        "delta": "0:00:00.006412",
        "end": "2020-05-10 13:25:45.035609",
        "failed": false,
        "failed_when_result": false,
        "msg": "non-zero return code",
        "rc": 1,
        "start": "2020-05-10 13:25:45.029197",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "1c1\n< abc\n---\n> xyz",
        "stdout_lines": [
            "1c1",
            "< abc",
            "---",
            "> xyz"
        ]
    }
}

PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

ファイルの内容が同じ場合の実行ログです。タスク "ファイルの比較" のエラーメッセージ内に**"rc": 0** が含まれています。本来ならエラーにならないのですが、failed_when ディレクティブに設定したエラーとして扱う条件に合致したためエラーとなり処理を中断しました。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml file-diff.yml

PLAY [2 つのファイルを比較する]␣
↪*********************************************************************************************************

TASK [ファイルの比較]␣
↪*********************************************************************************************************
```

(次のページに続く)

(前のページからの続き)

```
fatal: [node1]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": ["diff", "file-1", "file-2"], "delta":
↪"0:00:00.004663", "end": "2020-05-10 13:30:46.133139", "failed_when_result": true,
↪"rc": 0, "start": "2020-05-10 13:30:46.128476", "stderr": "", "stderr_lines": [],
↪"stdout": "", "stdout_lines": []}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=0    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

ファイル file-2 が存在しない場合の実行ログです。タスク "ファイルの比較" のエラーメッセージ内に**"rc": 2** が含まれています。failed_when ディレクティブに設定したエラーとして扱う条件に合致したためエラーとなり処理を中断しました。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml file-diff.yml

PLAY [2 つのファイルを比較する]␣
↪*********************************************************************************************

TASK [ファイルの比較]␣
↪*********************************************************************************************
fatal: [node1]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": ["diff", "file-1", "file-2"], "delta":
↪"0:00:00.007034", "end": "2020-05-10 13:34:52.272081", "failed_when_result": true,
 "msg": "non-zero return code", "rc": 2, "start": "2020-05-10 13:34:52.265047", "stderr
↪": "diff: file-2: No such file or directory", "stderr_lines": ["diff: file-2: No␣
↪such file or directory"], "stdout": "", "stdout_lines": []}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=0    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$
```

上述のプレイに ignore_errors ディレクティブを指定しました。

```
- name: 2 つのファイルを比較する
  hosts: node1
  gather_facts: no

  tasks:
  - name: ファイルの比較
    command: 'diff file-1 file-2'
```

(次のページに続く)

(前のページからの続き)

```
    register: diff_result
    failed_when: diff_result['rc'] != 1
    ignore_errors: yes
  - name: レジスタ変数の内容確認
    debug:
      var: diff_result
```

この場合は failed_when の条件に合致してエラーと判断した後、ignore_errors ディレクティブでエラーを無視します。ファイル file-2 が存在しないときの実行ログです。

```
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml file-diff.yml

PLAY [2 つのファイルを比較する]␣
↪*********************************************************************************************

TASK [ファイルの比較]␣
↪*********************************************************************************************
fatal: [node1]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": ["diff", "file-1", "file-2"], "delta":
↪"0:00:00.006167", "end": "2020-05-10 13:36:33.737414", "failed_when_result": true,
 "msg": "non-zero return code", "rc": 2, "start": "2020-05-10 13:36:33.731247", "stderr
↪": "diff: file-2: No such file or directory", "stderr_lines": ["diff: file-2: No␣
↪such file or directory"], "stdout": "", "stdout_lines": []}
...ignoring

TASK [レジスタ変数の内容確認]␣
↪*********************************************************************************************
ok: [node1] => {
    "diff_result": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
        "cmd": [
            "diff",
            "file-1",
            "file-2"
        ],
        "delta": "0:00:00.006167",
        "end": "2020-05-10 13:36:33.737414",
        "failed": true,
        "failed_when_result": true,
        "msg": "non-zero return code",
        "rc": 2,
        "start": "2020-05-10 13:36:33.731247",
        "stderr": "diff: file-2: No such file or directory",
        "stderr_lines": [
```

(次のページに続く)

(前のページからの続き)

```
            "diff: file-2: No such file or directory"
        ],
        "stdout": "",
        "stdout_lines": []
    }
}

PLAY RECAP␣
↪*********************************************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=1

[vagrant@ansible ansible-files]$
```

9.2.5 エラーハンドリングの演習

管理対象ホスト node1 にアカウントを追加します。ただし、/etc/passwd に登録済みのアカウントは追加や更新してはいけません。登録済みのときはエラーで処理を中断します。

1. アドホックコマンドで/etc/passwd にアカウントが登録されているとき、登録されていないときの戻り値を確認します。

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m shell -a 'cat /etc/
↪passwd | grep vagrant'
node1 | CHANGED | rc=0 >>
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m shell -a 'cat /etc/
↪passwd | grep hogehoge'
node1 | FAILED | rc=1 >>
non-zero return code
[vagrant@ansible ansible-files]$
```

ちなみに: パイプやリダイレクトなどを使用するときは shell モジュールを使用します。command モジュールはパイプなどを処理できません。

2. （ざっくりとした）処理を考えます。

- /etc/passwd ファイルに登録したいアカウントが登録されているかどうか確認
  - shell モジュール
  - register ディレクティブ

  - failed_when ディレクティブ

- アカウントを登録
  - user モジュール
  - become ディレクティブ（targets セクションで定義していたらタスクでの指定は不要）

3. プレイを作成・実行します。
4. アドホックコマンドで管理対象ホスト node1 の/etc/passwd ファイルの内容を参照し、アカウントが登録されていることを確認します。
5. 登録済みアカウントに変更し、エラーで処理を中断することを確認します。

解答

```
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m shell -a 'cat /etc/
↪passwd | grep vagrant'
node1 | CHANGED | rc=0 >>
vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash
[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m shell -a 'cat /etc/
↪passwd | grep hogehoge'
node1 | FAILED | rc=1 >>
non-zero return code
[vagrant@ansible ansible-files]$ vim useradd.yml
[vagrant@ansible ansible-files]$ cat useradd.yml
- name: アカウントを登録する
  hosts: node1
  gather_facts: no

  tasks:
  - name: アカウントが登録されているか確認
    shell: 'cat /etc/passwd | grep hogehoge'
    register: result
    failed_when: result['rc'] != 1
  - name: アカウント登録
    user:
      name: "hogehoge"
      state: present
    become: yes
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [アカウントを登録する]␣
↪*********************************************************************************************

TASK [アカウントが登録されているか確認]␣
↪*********************************************************************************************
changed: [node1]
```

(次のページに続く)

(前のページからの続き)

```
TASK [アカウント登録]␣
↪*********************************************************************************************
changed: [node1]

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=2    changed=2    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0

[vagrant@ansible ansible-files]$ ansible node1 -i hosts.yml -m shell -a 'cat /etc/
↪passwd | grep hogehoge'
node1 | CHANGED | rc=0 >>
hogehoge:x:1001:1001::/home/hogehoge:/bin/bash
[vagrant@ansible ansible-files]$ vim useradd.yml
[vagrant@ansible ansible-files]$ cat useradd.yml
- name: アカウントを登録する
  hosts: node1
  gather_facts: no

  tasks:
  - name: アカウントが登録されているか確認
    shell: 'cat /etc/passwd | grep vagrant'
    register: result
    failed_when: result['rc'] != 1
  - name: アカウント登録
    user:
      name: "vagrant"
      state: present
    become: yes
[vagrant@ansible ansible-files]$ ansible-playbook -i hosts.yml useradd.yml

PLAY [アカウントを登録する]␣
↪*********************************************************************************************

TASK [アカウントが登録されているか確認]␣
↪*********************************************************************************************
fatal: [node1]: FAILED! => {"ansible_facts": {"discovered_interpreter_python": "/usr/
↪bin/python"}, "changed": true, "cmd": "cat /etc/passwd | grep vagrant", "delta":
↪"0:00:00.009987", "end": "2020-05-10 14:45:01.615450", "failed_when_result": true,
↪"rc": 0, "start": "2020-05-10 14:45:01.605463", "stderr": "", "stderr_lines": [],
↪"stdout": "vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash", "stdout_lines": [
↪"vagrant:x:1000:1000:vagrant:/home/vagrant:/bin/bash"]}

PLAY RECAP␣
↪*********************************************************************************************
node1                      : ok=0    changed=0    unreachable=0    failed=1    ␣
↪skipped=0    rescued=0    ignored=0
```

(次のページに続く)

(前のページからの続き)

```
[vagrant@ansible ansible-files]$
```

第 10 章

改版履歴

2020/05/10

- 初版公開