

【PTP SE 認定取得コース】 Red Hat SE Specialist - Automation ～ AAP 2.1 座学～

レッドハット株式会社 テクニカルセールス本部
パートナーソリューションアーキテクト部
アソシエイトプリンシパルソリューションアーキテクト
岡野 浩史

最終更新 : 2022年 3月

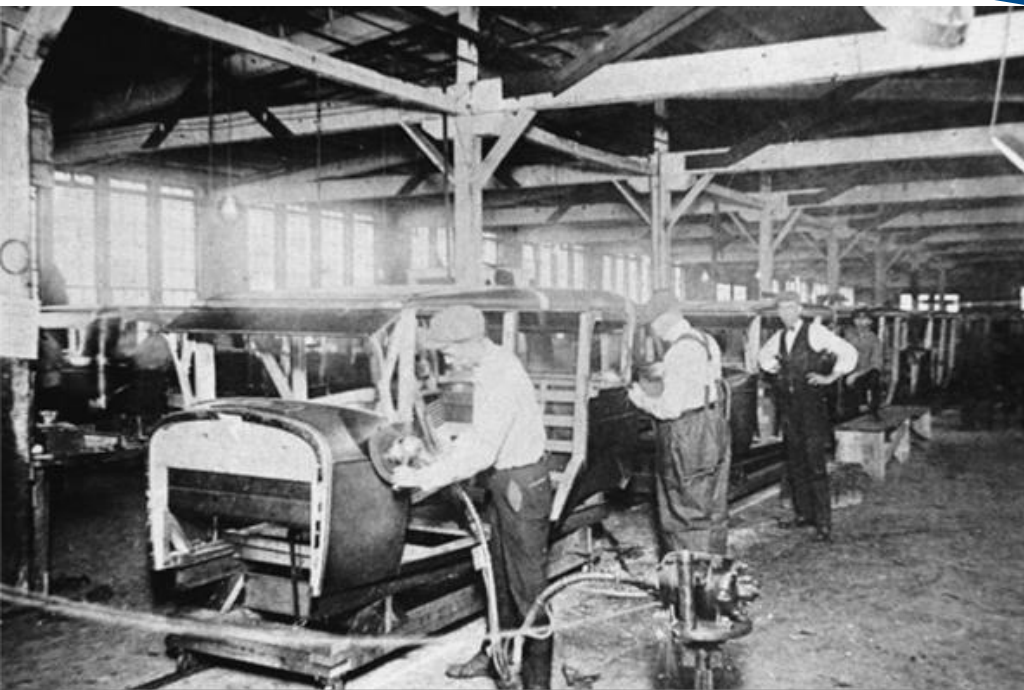


理想的なインフラ管理

人はもう『そこ』にはいない...

人が作業
(機械で補助する)

機械が作業
(人が管理する)



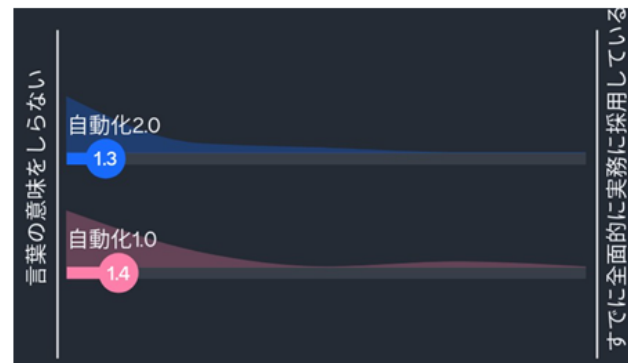
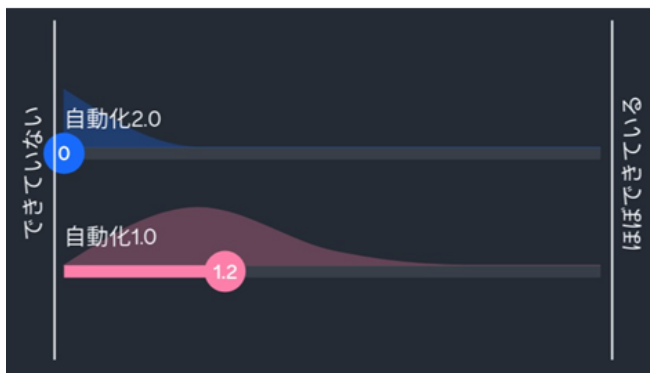
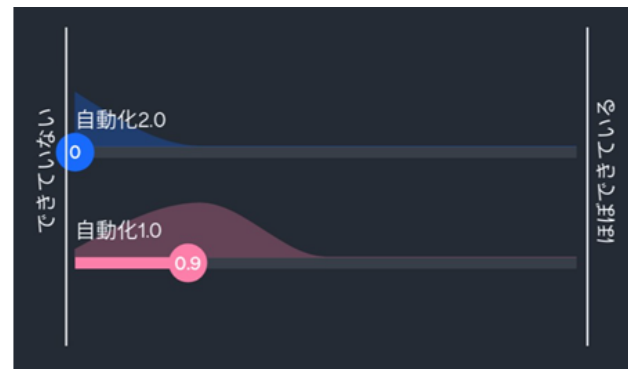
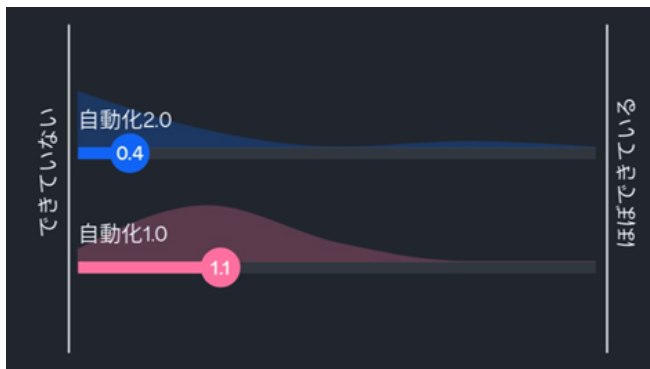
自動化以前～自動化1.0



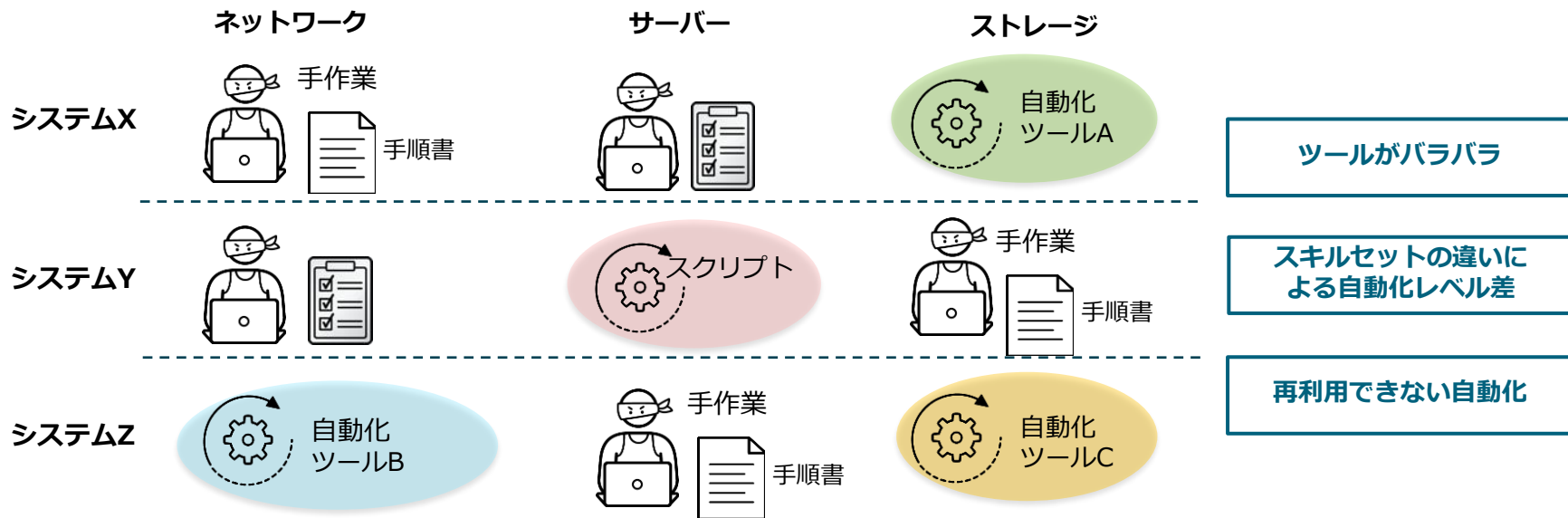
自動化 2.0 の世界

実態 - ほぼ全ての顧客は自動化1.0の状態

Automation Day ワークショップ結果より（4社の実例）



従来のインフラ管理が抱える課題（サイロ化）



自動化を進めても・・・

どの作業もあの人しか
わからない・できない

ブラックボックス化
ボトルネック化

横展開が不可能
標準化を阻害

全体の効率低下

自動化の効果が出ない。広がらない。本気で取り組まれない。

自動化で効果を出すための3つのポイント

ポイント1 自動化の標準化

- 自動化の対象ごとにバラバラの開発物、利用方法となっていた自動化を、標準化された自動化の開発と活用が可能な状態にする。
- 自動化2.0への移行難易度を下げる。

まずは『共通言語』

自動化1.0の課題を解決し
2.0への移行を加速する

ポイント2 自動化のサービス化

機能面特化のため一部のみ紹介

- 自動化を手順の置き換えではなく、サービスとして利用者に提供する。
- 作業に登場する「登場人物」を減らすことで、調整そのものが発生しない状態にする。

**作業をボタン化し
サービスとして提供**

自動化2.0を実現するためのアプローチ

ポイント3 インフラCI化

- 「人與人」の対話によって品質を上げてきた従来の方法から、「人とシステム」の対話へと切り替える。
- 事前の調整や前提の説明等の「対話のための状態同期コスト」を最適化する。

**インフラ運用担当者は
『サービス開発』へ**

Ansible Automation Platform の価値

～ 『対象外』を作らない強力かつ理解しやすいツール～

『標準化』に要求されるツールの性質

～ Ansible Automation Platform の特徴 ① ～



低い学習コスト

Simple

誰もが読める標準化
された自動化言語



例外なし

Powerful

多様な制御対象を
統一的手法で自動化

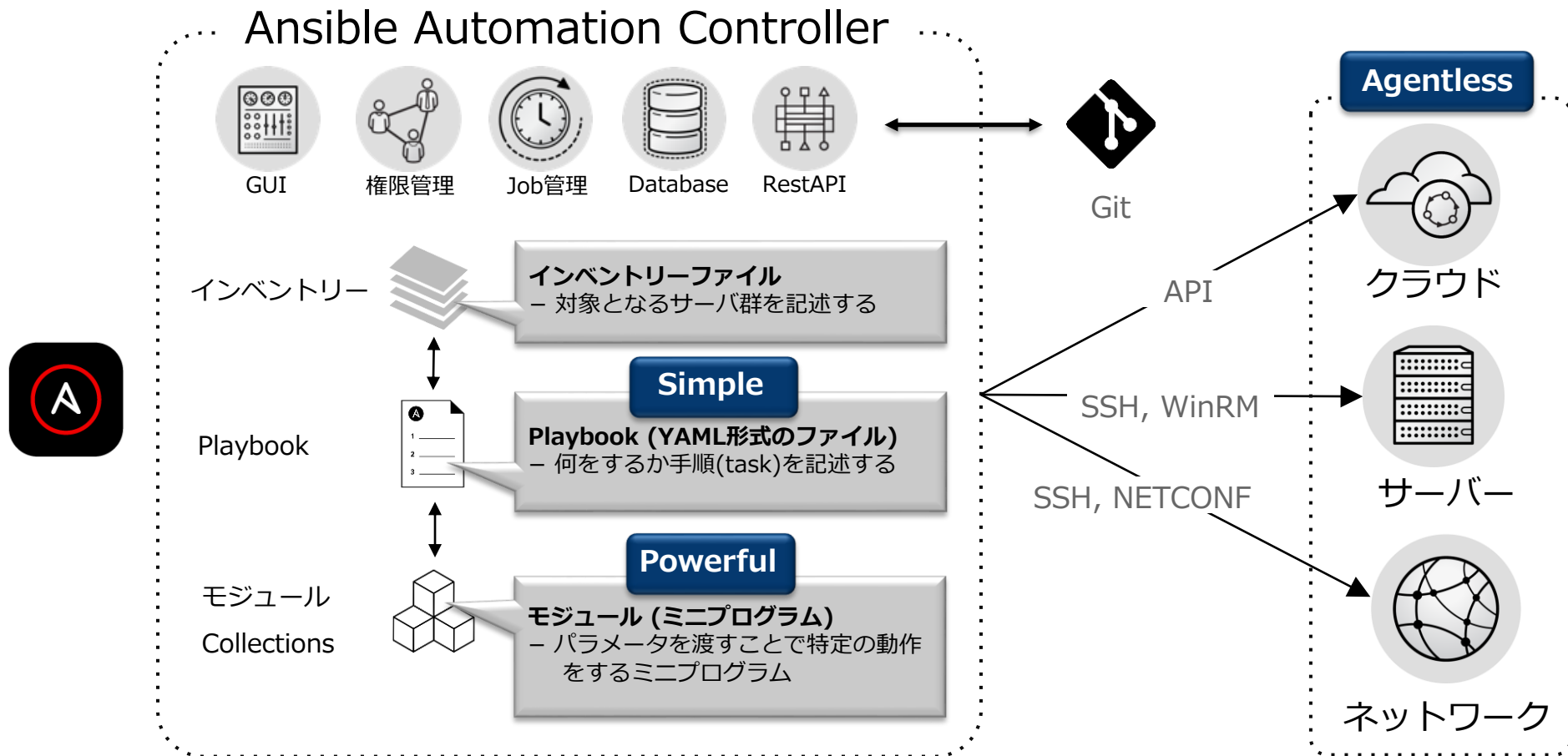


導入が容易

Agentless

追加も簡単
セキュリティ懸念無し

Ansible Automation Platform (AAP) 構成要素





ビュー

- ダッシュボード
- ジョブ
- スケジュール
- アクティビティストリーム
- ワークフローの承認

リソース

- テンプレート
- 認証情報
- プロジェクト
- インベントリ
- ホスト

アクセス

- 組織
- ユーザー
- チーム

管理

- 認証情報タイプ
- 通知
- 管理ジョブ
- インスタンスグループ
- アプリケーション
- 実行環境

設定

ダッシュボード



4
ホスト

0
失敗したホスト

1
インベントリ

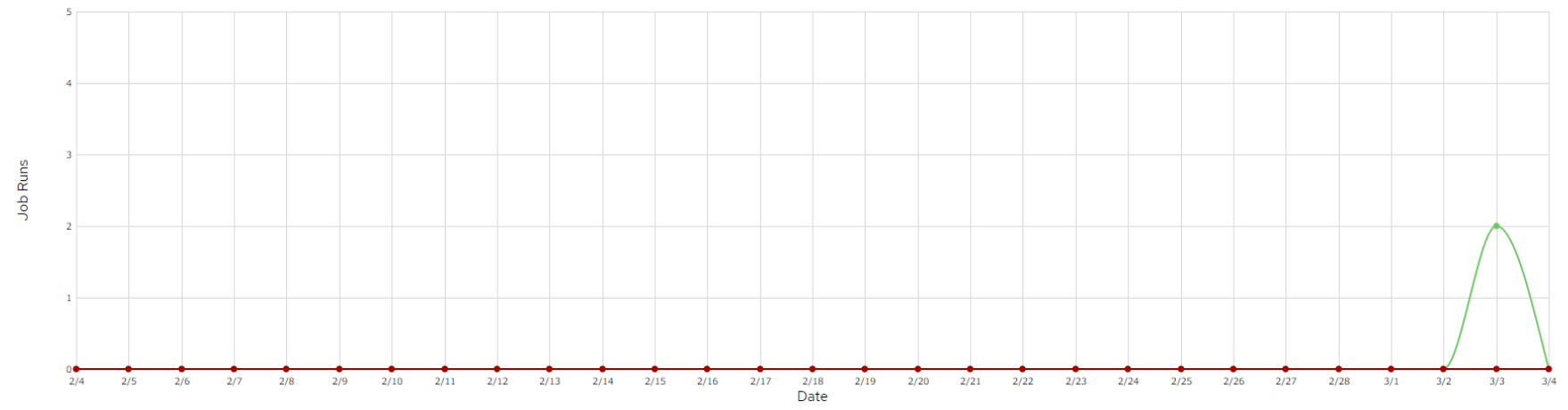
0
インベントリの同期の失敗

1
プロジェクト

0
プロジェクトの同期の失敗

ジョブステータス 最近のジョブ 最近のテンプレート

過去1ヵ月 すべてのジョブタイプ すべてのジョブ



Automation Controller UI

AAP の特徴 - Simple ①

～ 標準化を促進する簡単な言語 ～



Ansibleプレイブック

実行順序



- name: **Apacheのインストールと起動**
hosts: app
become: yes

#Playbook の説明
#app グループが対象（インベントリ）
#権限昇格の有無（プラグインで提供）

tasks:

#実行する手順の内容
#実行時に処理毎に表示される名前

- name: **httpd のインストール**
yum:

name: httpd
state: latest

- name: **httpd を起動**

service:

name: httpd
state: running

TARGET
セクション

タスク
セクション

モジュール

AAP の特徴 - Simple ②

～ プラグイン (使い易さを高める拡張機能)～

Ansible プラグイン

- Ansible のコア機能を拡張するコードの一部
- 外部ファイルとのアクセスや画面出力など便利な機能を提供

Example become plugin:

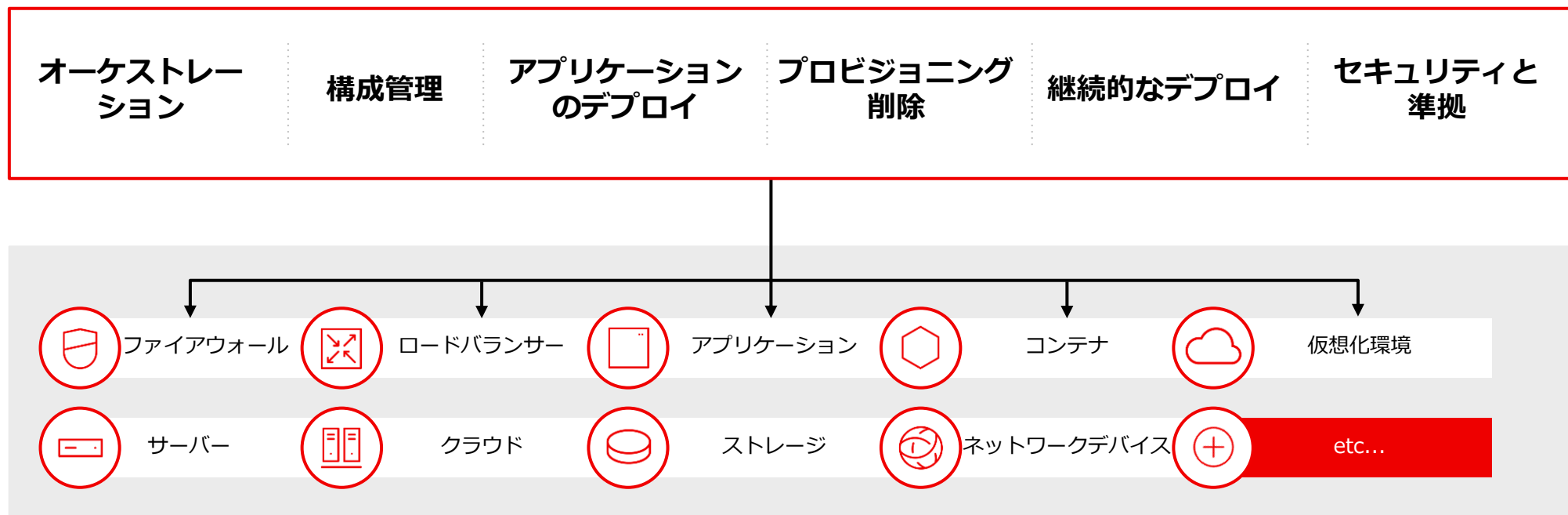
```
---  
- name: install and start apache  
  hosts: web  
  become: yes
```

Example filter plugins:

```
{{ some_variable | to_nice_json }}  
{{ some_variable | to_nice_yaml }}
```

AAP の特徴 - Powerful ①

～ 自動化対象の IT プラットフォームの例 ～



AAP の特徴 - Powerful ②

～ 多くのIT機器を対象とした自動化が可能 ～



インフラストラクチャー



クラウド



ネットワーク



セキュリティ

100+

非常に多くの認定プラットフォーム



ARISTA



Check Point
SOFTWARE TECHNOLOGIES LTD



FORTINET

AAP の特徴 - Powerful ③

～ モジュール詳細 ～

Ansible モジュール

- Ansible の中核的な機能
- Playbook のタスクの中で実行時に呼び出され対象ノードに作用する（動作イメージとしては Playbook と対象ノードを橋渡しするラッパーとして作用する）
- 通常 Python* (Windows では Powershell) で記述されるが基本的に言語の制限はない

*Playbook を書く際に Python を理解している必要はありません。
モジュールの使い方 (Playbook の書き方) を理解していれば大丈夫です。



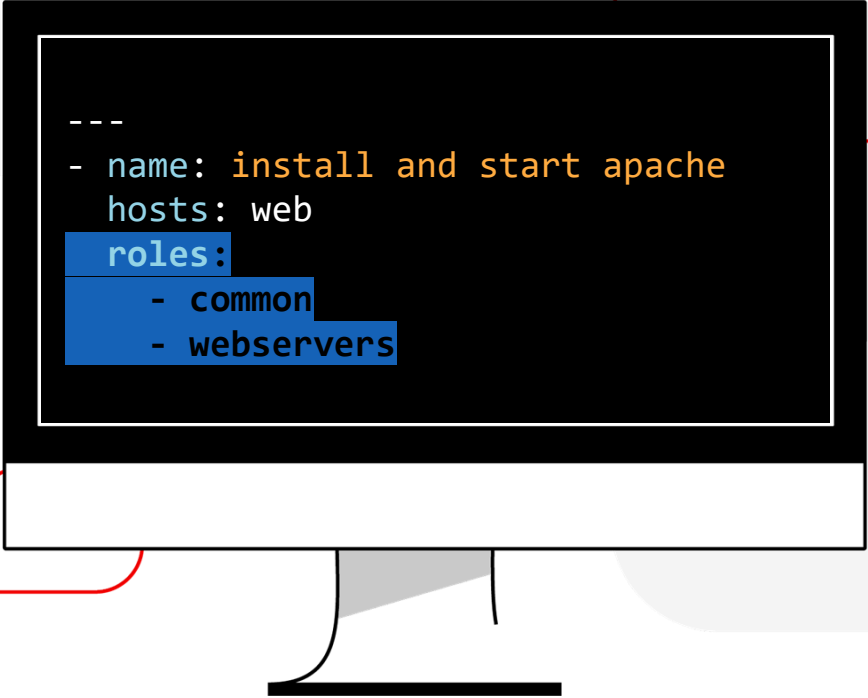
```
- name: latest index.html file ...  
  template:  
    src: files/index.html  
    dest: /var/www/html/
```

AAP の特徴 - Powerful ④

～ Ansible roles (自動化利用の促進) ～

Ansible roles

- ・ Playbook を再利用しやすい形に分解・定義
- ・ タスクと変数を再利用可能な構造にグループ化
- ・ roles 化しておくことで、自分だけではなく同じような自動化を行いたい他の人と共用可能

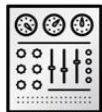


```
---  
- name: install and start apache  
  hosts: web  
  roles:  
    - common  
    - webservers
```

The image shows a stylized computer monitor with a black screen. On the screen, there is a code snippet for an Ansible role. The code is written in a light blue font on a dark background. The code defines a role named 'install and start apache' for the 'web' hosts, which includes two sub-roles: 'common' and 'webservers'. The monitor is depicted with a white bezel and a black stand. Red decorative lines are drawn around the monitor, including a large arc above it and a smaller one to the left.

『サービス化』に要求されるツールの性質

～ Ansible Automation Platform の特徴② ～



サービスの作成

Interface

使い易い共通インター
フェイスで複雑な作業
を簡単にサービス化



サービスの連結

Control

ワークフローで
自動化サービスを連結



必要な人が実行

Delegation

必要な人に権限を委譲
必要な人が自動化サー
ビスを実行

AAP の特徴 - 使い易いインターフェース

洗練された GUI で誰でも簡単に操作できます



外部ソフトウェアとの連携
には RESTAPI 完備

```
REST API / バージョン 2 / Job Template List

Job Template List2

GET /api/v2/job_templates/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: aap21-4.lab.local
X-API-Product-Name: Red Hat Ansible Automation Platform
X-API-Product-Version: 4.1.1
X-API-Time: 0.048s

{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 7,
      "type": "job_template",
      "url": "/api/v2/job_templates/7/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "labels": "/api/v2/job_templates/7/labels/",
        "inventory": "/api/v2/inventories/1/",
        "project": "/api/v2/projects/6/",
        "organization": "/api/v2/organizations/1/",
        "credentials": "/api/v2/job_templates/7/credentials/",
        "jobs": "/api/v2/job_templates/7/jobs/",
        "schedules": "/api/v2/job_templates/7/schedules/",
        "activity_stream": "/api/v2/job_templates/7/activity_stream/",
        "launch": "/api/v2/job_templates/7/launch/",
        "webhook_key": "/api/v2/job_templates/7/webhook_key/",
        "webhook_receiver": "",
        "notification_templates_started": "/api/v2/job_templates/7/notifi
```

CLI も充実！！ Controller CLI (AWX) コマンド！

awx job_templates launch <id>

<https://docs.ansible.com/automation-controller/latest/html/controllercli/usage.html>

AAP の特徴 - 使用権限を委譲（プロジェクト）

“プロジェクト” による Playbook ディレクトリの管理と権限の委譲

Playbook ディレクトリ： /var/lib/awx/projects/<各プロジェクト>/

利用オーナーを決めて使用権限を委譲する！！

```
[root@ansible projects]# pwd
/var/lib/awx/projects
[root@ansible projects]# tree
```

- apps (App 用 Playbook)
 - fw.yml
 - httpd.yml
 - service.yml
 - win_choco.yml
- aws (AWS 用 Playbook)
 - facts.yml
 - manage_ELB.yml
 - manage_instance.yml
 - manage_securityGroup.yml
- network (Network 用 Playbook)
 - backup.yml
 - gather_f5_facts.yml
 - gather_ios_facts.yml
 - gather_juniper_facts.yml
 - manage_f5_config.yml
 - manage_ios_config.yml
 - manage_juniper_config.yml
 - restore.yml
- vmware (VMware 用 Playbook)
 - facts.yml
 - ManageCL.yml
 - manageDC.yml
 - manageVMTag.yml
 - manageVM.yml
 - wait-tools.yml

Red Hat Ansible Automation Platform

プロジェクト > App-Manage
詳細の編集

名前 * App-Manage 説明 組織 * App-Org

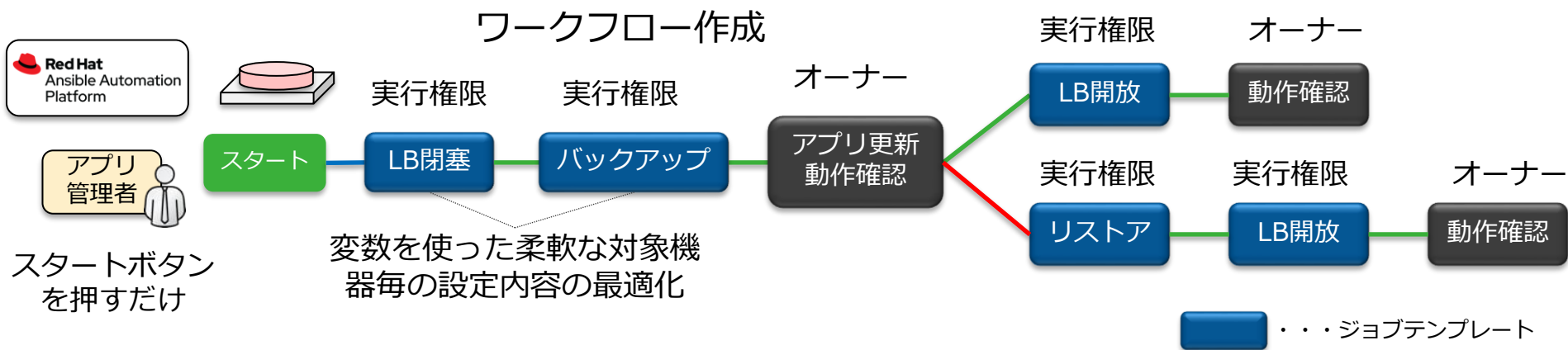
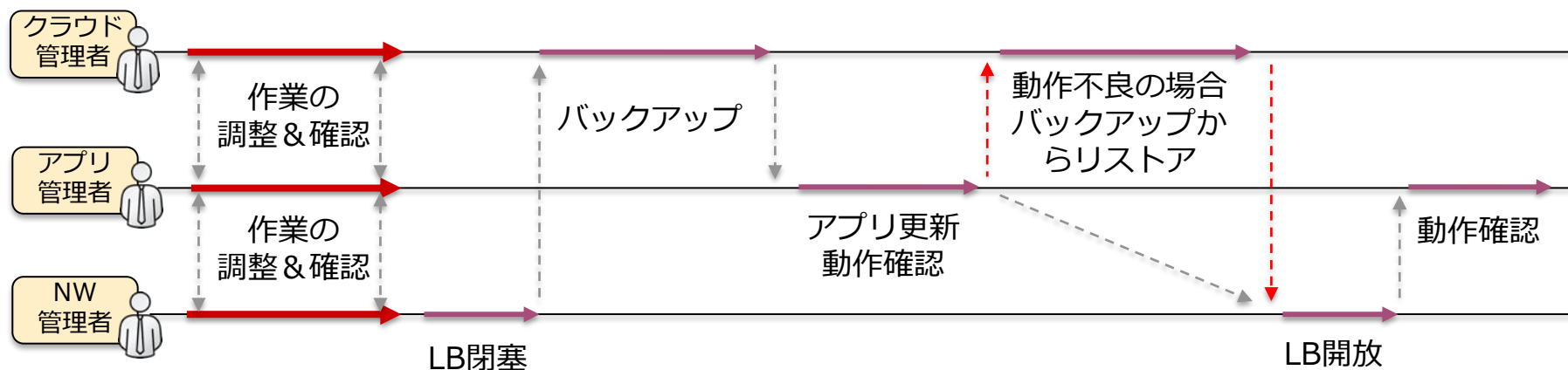
実行環境 ⓘ Default execution environment ソースコントロール認証情報タイプ * 手動

タイプの詳細

プロジェクトのベースパス ⓘ	Playbook ディレクトリー * ⓘ
/var/lib/awx/projects	apps

保存 取り消し

AAP の特徴 - ワークフローでサービスを連結



AAP 2.1 What's New

～ 使い易く拡張性に富んだコントローラーへと進化 ～

AAP 2.1 - 新たな価値の提供

- 実行環境のコンテナ化 (AAP 1.x では python の virtualenv)
複数の python モジュールや依存関係の管理と移行が容易に
『private automation hub』で一括管理・利用
- automation controller の機能分割実装が可能に (automation mesh)
 - Hybrid ノード
 - Control ノード
 - Execution ノード・・・ジョブ実行のみ担当
 - Hop ノード・・・Isolation されたネットワークへのホップのみ担当

旧 controller と比較して、スケーリングが容易かつ、遅延のあるリモートネットワーク上のノードも一括管理が可能に

AAP 2.1 概要 - 刷新されたコンポーネント一覧

AAP 2.1 Component	AAP 1.2
ansible core (2.12.2 約 70 モジュール / コンテナで提供)	2.9.x (3,000+モジュール / OS インストール)
ansible automation controller 4.1	Ansible Tower 3.8
ansible runner (v2 : コンテナ対応)	v1
ansible execution environment (コンテナ化された実行環境)	N/A (virtualenv)
private automation hub (コンテナ対応)	Collections/Galaxy Sync (コンテナ非対応)
ansible-navigator (TUI:コンテナ実行環境対応)	N/A
ansible-builder (独自のコンテナ実行環境の作成)	N/A
automation mesh	Isolated Node

ポイントは3つ

- ・コンテナ化された実行環境
- ・ansible-core 2.12.x は実行環境内（コンテナ）で提供
- ・Isolated Node → Automation Mesh へ

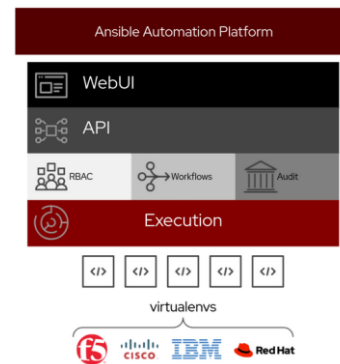
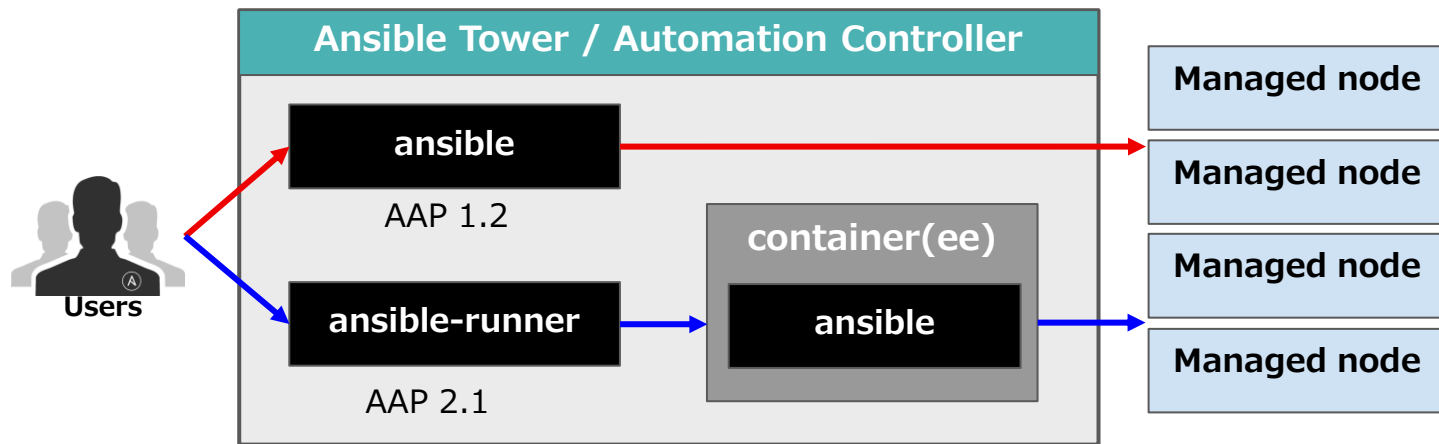
コンポーネント詳細 - execution environment (ee)

～ virtualenv → コンテナベースの実行環境へ ～

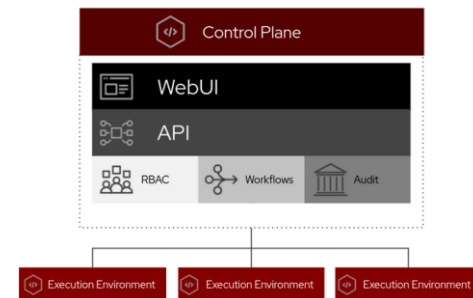
機器毎、担当者毎に設定されていた virtualenv の実行環境がコンテナベースに！

- ・ 複数の環境がベースOSと分離されより堅牢・安全に
- ・ 実行環境の可搬性と共有化などの利便性が大幅に拡大！
- ・ 『ansible』も実行環境の中で提供！

OSには直接インストールされません！！



AAP 1.2 virtualenv



AAP 2.1コンテナ化された実行環境

実行環境詳細

実行環境は Red Hat Ecosystem Catalog で提供！

2022年 3月4日現在以下の3種類が提供

1. Supported execution environment

AAP 2.1 環境でデフォルト利用が想定される実行環境

- ansible core : 2.12.x
- モジュール : 2.12 のコアモジュール (約70種類)* 及び collections の認定コンテンツ**の中で "Supported by Red Hat" となっているもの (約30種類) が含まれる
- Python 依存パッケージ : Compatibility execution environment と同等のものが含まれる

2. Compatibility execution environment

旧 AAP 1.2 と互換性の高い実行環境。AAP 1.2 → 2.1 への移行時の強い味方♪

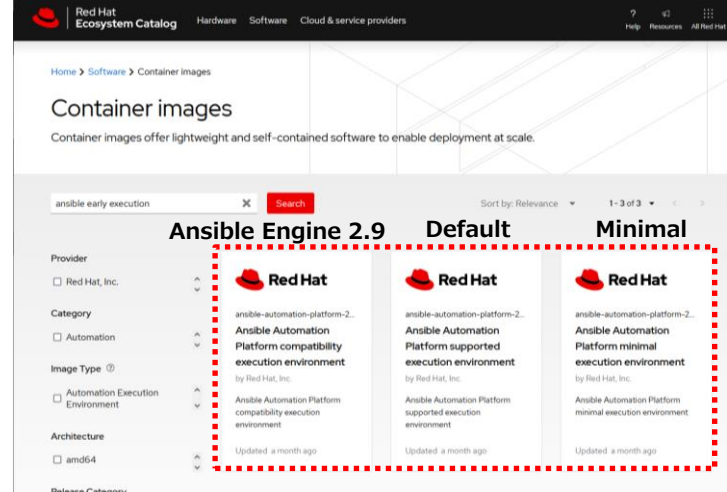
- ansible engine : 2.9.27 (2022年3月4日時点)
- モジュール : ansible engine に含まれるモジュール群 (3,000+) コミュニティモジュールも含む
- Python 依存パッケージ : コミュニティモジュールに必要なもの含め数多く含まれる

3. Minimal execution environment

Collections を含まない必要最低限のジョブ実行環境 (カスタマイズ際のベースイメージ)

*Ansible Builtin : <https://docs.ansible.com/ansible-core/2.12/collections/ansible/builtin/index.html>

**Ansible Automation Platform Certified Content : <https://access.redhat.com/articles/3642632>

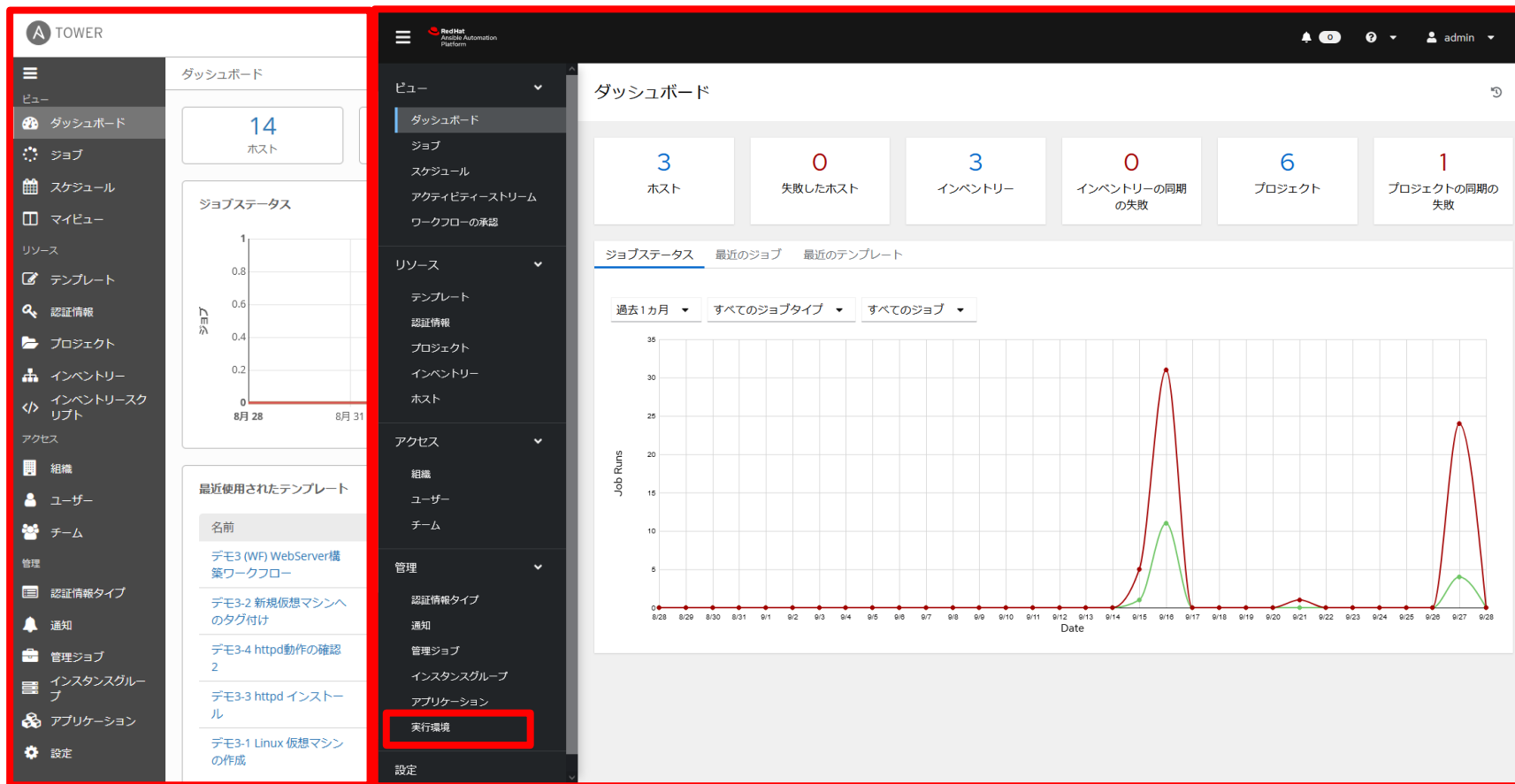


実行環境を使った Playbook の実行

～ automation controller 編 ～

Tower 3.8

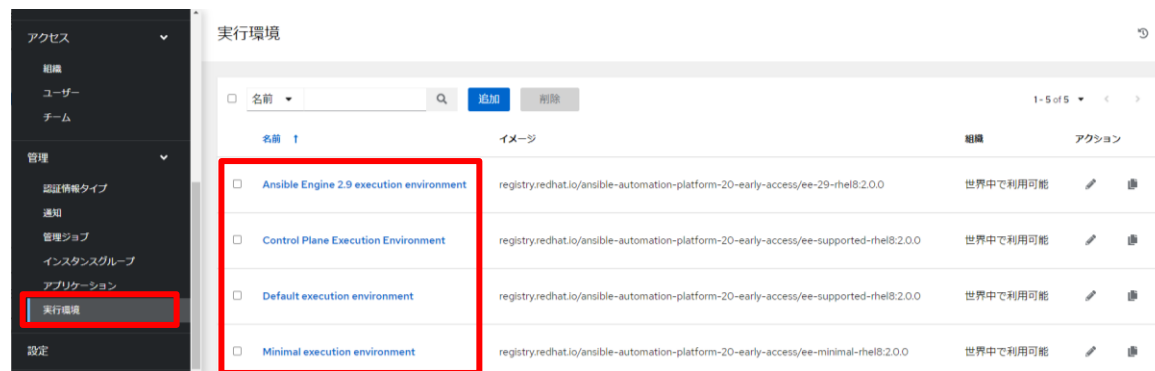
AAP 2.1 Automation Controller



実行環境を使った Playbook の実行

～ automation controller 編 ～

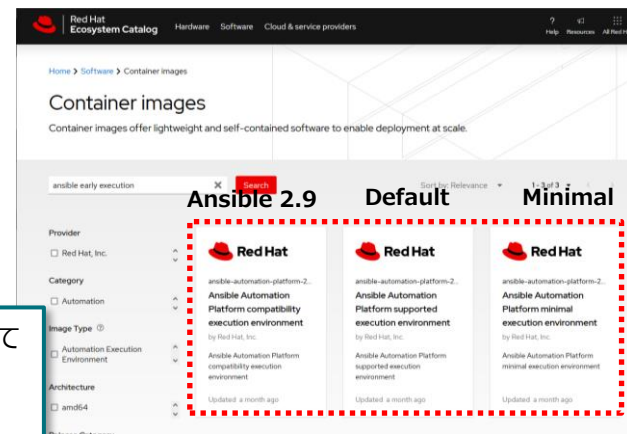
実行環境は Automation Controller にインストール時に定義（ダウンロードは初回利用時）
ジョブテンプレート（もしくはプロジェクト）で指定して実行するのみ、簡単です♪



registry.redhat.io への認証
情報もインストール時に定義



ジョブ実行時など必要に応じて
registry.redhat.io から
自動的にダウンロード

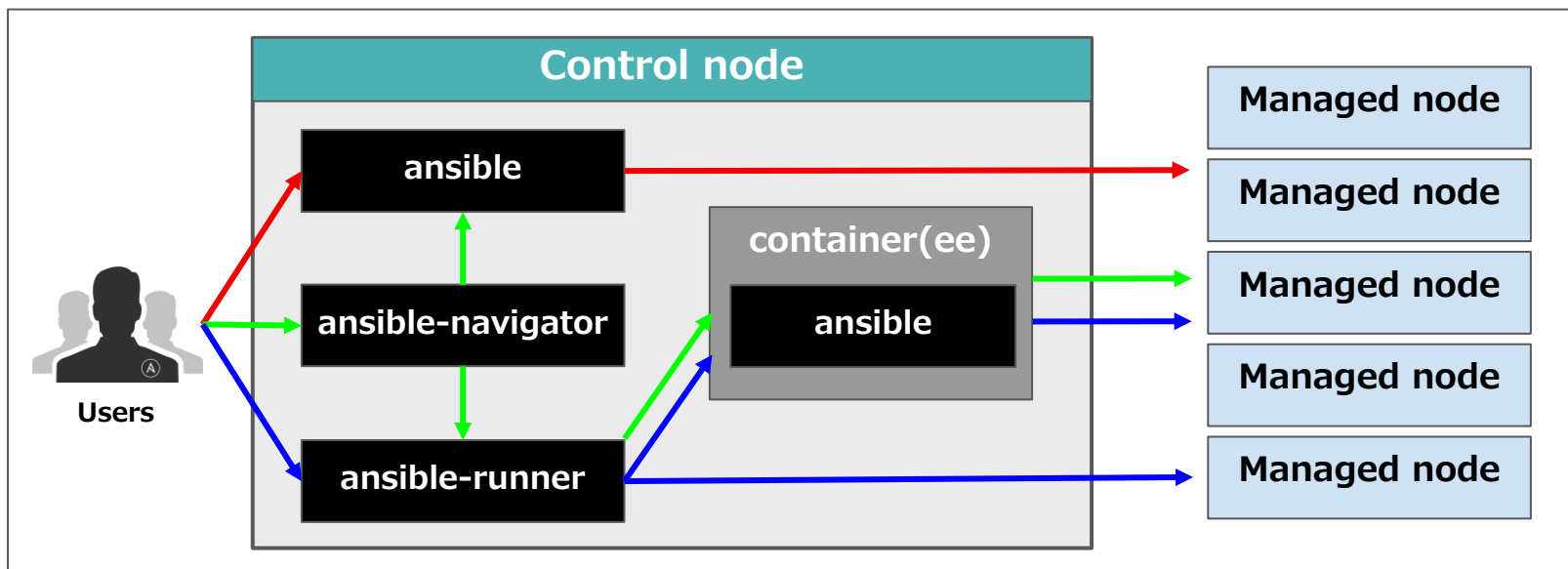


実行環境を使った Playbook の実行

～ ansible-navigator 編 ～

ansible-navigator

- AAP2.1 とは別インストールですが、非常に多機能な CLI / TUI ツール
- 実行環境コンテナの詳細確認
- 実行環境コンテナを利用した Playbook の実行と json での実行 Log の保存 etc.



ansible core 2.12.x - コアモジュール

モジュールの提供は約70個（2.9.x では 3,000+）

- ・ AAP 2.1 では、ansible は実行環境の中で提供されています。

現時点では、『ansible engine 2.9.27』 と 『ansible core 2.12.1』 の 2 種 Ansible core 2.12 でコアモジュールとして提供されているモジュールは約70個。debug, service, setup, template, yum など含まれますが、firewalld, Network, Cloud など含まれません。*

```
[root@aap2-2 ~]# ls /usr/lib/python3.8/site-packages/ansible/modules/
__init__.py      copy.py          get_url.py       include_vars.py  replace.py       systemd.py
__pycache__      cron.py          getent.py        iptables.py      rpm_key.py       sysvinit.py
add_host.py      debconf.py       git.py           known_hosts.py  script.py        tempfile.py
apt.py           debug.py         group.py         lineinfile.py   service.py       template.py
apt_key.py       dnf.py          group_by.py      meta.py          service_facts.py unarchive.py
apt_repository.py dpkg_selections.py hostname.py      package.py       set_fact.py      uri.py
assemble.py     expect.py       import_playbook.py package_facts.py set_stats.py     user.py
assert.py       fail.py         import_role.py   pause.py         setup.py         validate_argument_spec.py
async_status.py fetch.py         import_tasks.py  ping.py         shell.py         wait_for.py
async_wrapper.py file.py         include.py       pip.py          slurp.py         wait_for_connection.py
blockinfile.py  find.py        include_role.py  raw.py          stat.py          yum.py
command.py      gather_facts.py include_tasks.py reboot.py        subversion.py   yum_repository.py
```

実行環境の作成 - ansible-builder

- ・ 実行環境の作成・カスタマイズの機能を提供
- ・ AAP 2.1 とは別インストールの CLI ツール

```
$ ansible-builder build -t new-ee:0.0.1
```

requirements.yml (module)

```
---
collections:
  - ansible.posix
  - community.vmware
```

execution-environment.yml (作成する実行環境の定義ファイル) の例

build_arg_defaults:

#ベースの実行環境を指定

EE_BASE_IMAGE: 'registry.redhat.io/.../**ee-minimal-rhel8:latest**'

#Build に使う実行環境を指定

EE_BUILDER_IMAGE: 'registry.redhat.io/.../**ansible-builder-rhel8:latest**'

dependencies:

galaxy: **requirements.yml** ---> インストールする Collections を記載

python: **requirements.txt** ---> インストールする Python 依存関係を記載

system: **bindep.txt** ---> 必要なOSパッケージを記載

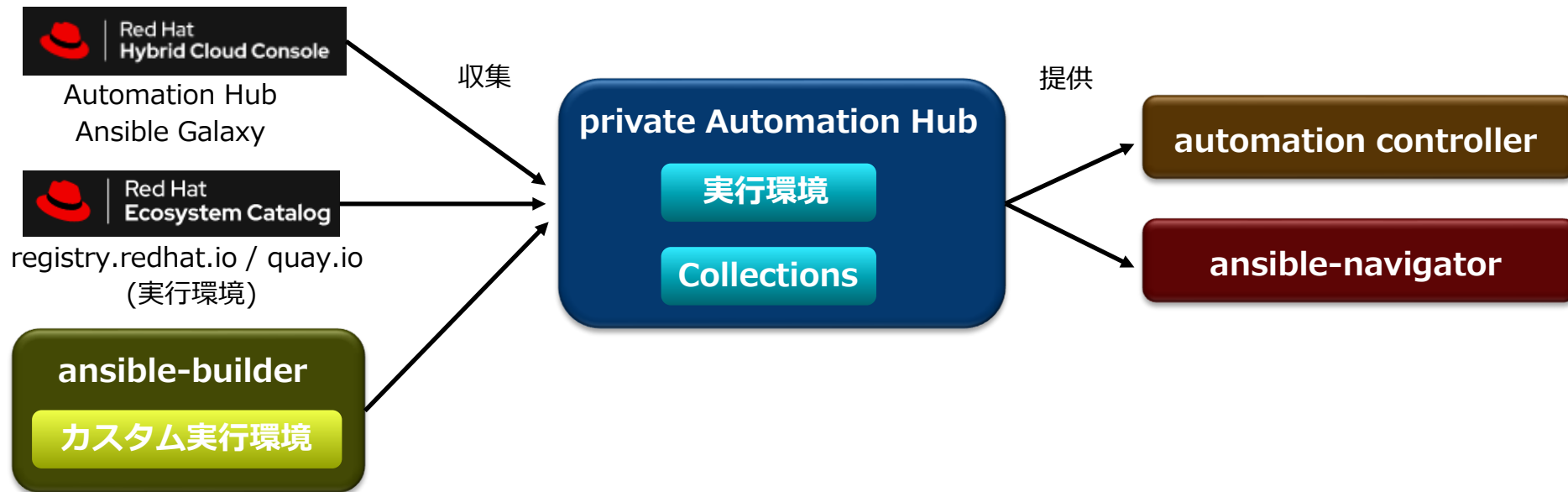
ベース
+

追加パッケージ

実行環境の収集と共有 - Private Automation Hub

AAP2.1で必要とされる 2 つの機能をプライベート環境下で提供

- ・ コンテナ化された実行環境を管理するレジストリの一括管理と提供
- ・ Collections や Galaxy で提供される Ansible モジュールなどの収集・提供



実行環境の収集と共有 - Private Automation Hub

- UI は automation controller など他の Red Hat ソフトウェアと統一
- 実行環境を直接、もしくは間接的に (podman pull / push) 収集
- インストールは automation controller 同様に setup.sh で行う

The screenshot shows the Red Hat Private Automation Hub interface. The top navigation bar includes the Red Hat logo and a user profile dropdown for 'admin'. The left sidebar contains a menu with items: Automation Hub, Collections (with a dropdown arrow), Namespaces, Repository Management, API Token, Approval, Container Registry (highlighted), Documentation (with an external link icon), and User Access (with a right arrow icon). The main content area is titled 'Container Registry' and features a search bar with a dropdown for 'Container repository name', a filter input 'Filter by container rep...', and a search icon. A link 'Push container images' with an external icon is also present. Below the search bar, a table displays registry entries. The table has columns for 'Container repository name', 'Description...', 'Created', and 'Last modified'. One entry is visible with the name 'ansible-automation-platform-20-early-access/ee-supported-rhel8', created 'a few seconds ago', and last modified 'a few seconds ago'. The table indicates '1-1 of 1' entries.

Container repository name ↓	Description...	Created ↑	Last modified ↓
ansible-automation-platform-20-early-access/ee-supported-rhel8		a few seconds ago	a few seconds ago

Isolated ノードは Automation mesh へ進化!!

～ AAP 2.1 ノードの種類 ～

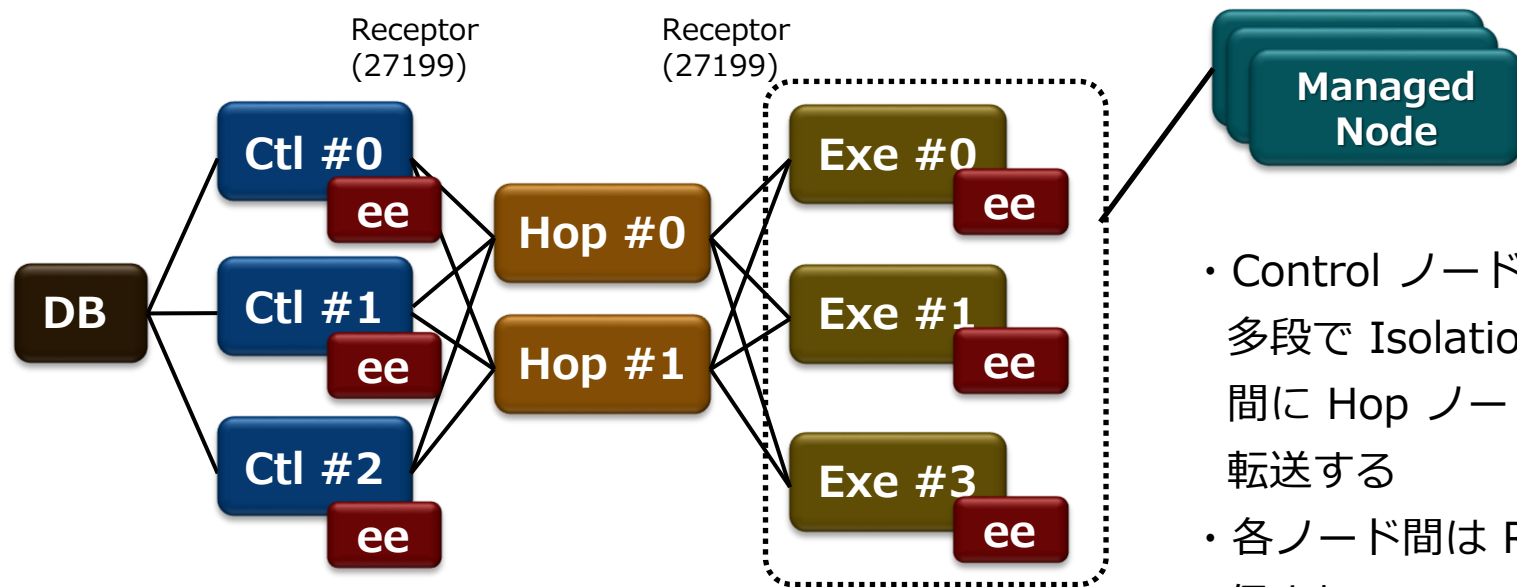
- ・ AAP2.1 のアーキテクチャを構成するノードは以下の4タイプ。
- ・ インストールプログラム(setup.sh)が参照するインベントリファイルで指定

#	ノードタイプ	役割
[1]	Control	ダッシュボードやAPI、そしてジョブスケジューラのようなバックグラウンドプロセスのみを実行します。ジョブの実行は[2]に送信
[2]	Execution	受信したジョブテンプレートをコンテナ化された実行環境内で実行
[3]	Hybrid	[1]の機能と[2]の機能をもちます。旧 Ansible Towerノードに相当
[4]	Hop	[1][3]からのジョブ実行処理を[2]に向けて転送

Isolated ノードは Automation mesh へ進化!!

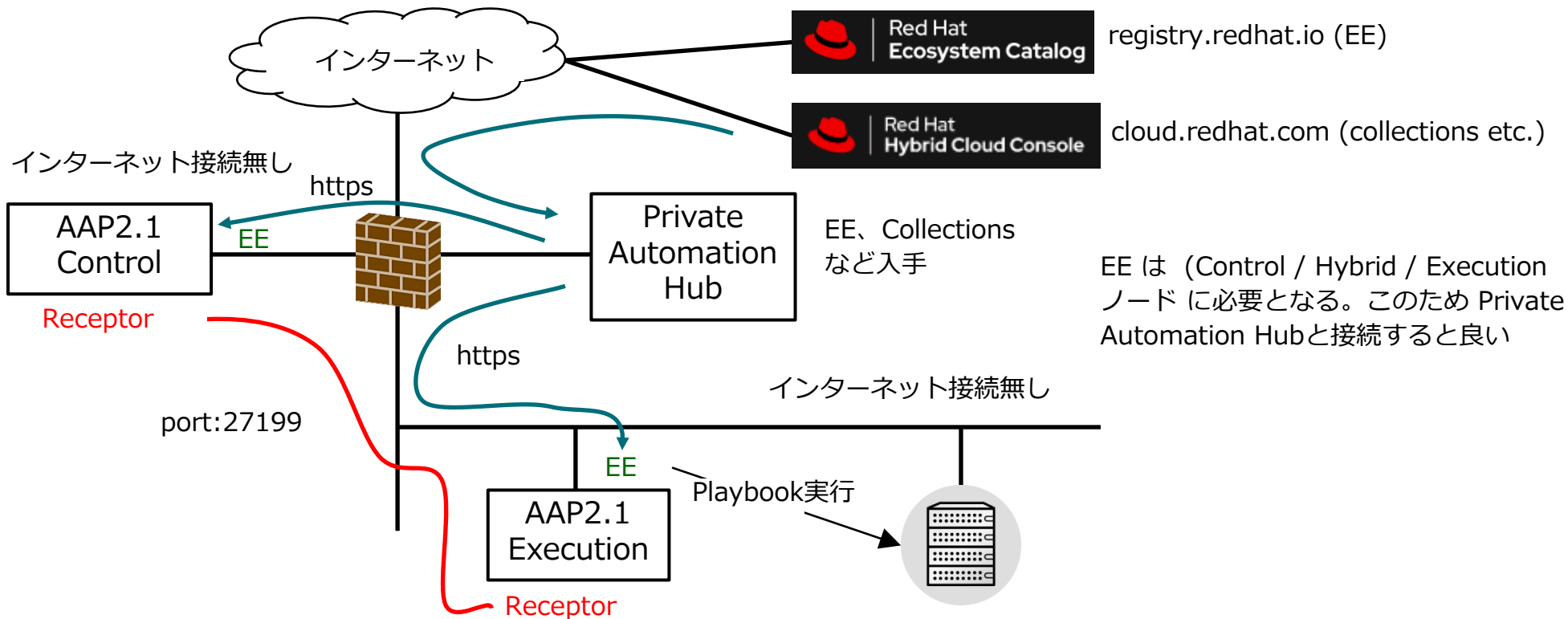
～ AAP 2.1 ノードと構成例③ ～

Control ノードと Execution ノード間に Hop ノードをはさんだ形
ネットワークが多段に Isolation されている場合に有用



- Control ノード Managed ノードが多段で Isolation されている場合は、間に Hop ノードをはさみ、ジョブを転送する
- 各ノード間は Receptor によって通信される

Automation mesh ネットワーク構成（例）



EE は (Control / Hybrid / Execution ノード) に必要となる。このため Private Automation Hubと接続すると良い

※ Hop は多段のIsolationの際に利用...など
Controller / Execution / Hop の接続は Receptor (Port:27199) が行う

AAP 2.1 への移行に関して

～ 注意点と実施方法 ～

AAP 1.2 から AAP 2.1へ - 考慮点①

～ 利用モジュールと Playbook の記述方法 ～

先述の通りAAP 2.1 ではモジュールの提供方法が変わります！

- ・ 利用しているモジュールが提供されているか？また入手方法（Collections / Galaxy）
- ・ Playbook / インベントリーの記述方法の違い（モジュール名 / localhost は NG! ）
などについて確認する ※Collections はモジュール名そのものが異なる
- ・ 初期は、『AAP2.1』 + 『2.9の実行環境』 + 『既存の Playbook』 を利用しながら徐々に2.12の環境に移行すると導入のハードルはぐっと下がります。

Ansible 2.9 (firewalld)

```
- firewalld:  
  service: https  
  permanent: yes  
  state: enabled
```

Ansible 2.12 posix (firewalld)

```
- ansible.posix.firewalld:  
  service: https  
  permanent: yes  
  state: enabled
```

以下のファイルを使って一括定義も可能です

https://github.com/ansible/ansible/blob/stable-2.12/lib/ansible/config/ansible_builtin_runtime.yml

AAP 1.2 から AAP 2.1へ - 考慮点②

～ virtualenv の実行環境コンテナへの移行 ～

- ・ 移行元の virtualenv 環境の "pip freeze" を表示
- ・ requirements.txt にコピーして ansible-builder で実行環境を作成！

```
[root@aap1.2 ~]# awx-manage export_custom_venv /paty/to/venv
# Virtual environment contents:
aiohttp==3.7.4.post0
async-timeout==3.0.1
attrs==21.2.0
chardet==4.0.0
idna==3.2
idna-ssl==1.1.0
multidict==5.1.0
psutil==5.8.0
python-memcached==1.59
six==1.16.0
typing-extensions==3.10.0.2
yarl==1.6.3
```

requirements.txt

```
aiohttp==3.7.4.post0
async-timeout==3.0.1
attrs==21.2.0
chardet==4.0.0
idna==3.2
idna-ssl==1.1.0
multidict==5.1.0
psutil==5.8.0
python-memcached==1.59
six==1.16.0
typing-extensions==3.10.0.2
yarl==1.6.3
```

AAP 1.2 から AAP 2.1へ - 考慮点③

～ アップグレード方法について ～

AAP1.2 からAAP2.1へのアップグレードパスは 2 つの方法がサポートされています。条件がありますので最適な方法を選択します。

1.インプレースアップグレード (./setup.sh)

移行元が RHEL8.4 (or later) + AAP 1.2 であることが条件

これより古い場合はあらかじめ上記バージョンに上げておく必要あり

- ・ RHEL 7.x → RHEL 8 へのインプレースアップグレードは[こちら](#)
- ・ Ansible Tower 3.7 以前 (既にEOLですが...) → 3.8 (AAP 1.2) に上げる際は、サブスクリプション適応方法が変わっている可能性があります、[こちら](#)ご参照ください。

AAP 1.2 から AAP 2.1へ - 考慮点③

～ アップグレード方法について ～

2.バックアップリストア (./setup.sh -b → ./setup.sh -r)

既存 OS が CentOS などでも新規で AAP 2.1 環境を構築する場合の手法

但しバックアップリストアは**同一 AAP バージョン間のみ**サポート！

つまり以下のような手順となります。

- ・新たに RHEL 8.4 (or later) をインストールする
- ・上記環境に AAP 1.2 をインストールする (./setup.sh)
- ・旧 AAP 1.2 環境をバックアップする (./setup.sh -b)
- ・新規インストールした AAP 1.2 環境にリストアする (./setup.sh -r)
- ・AAP 2.1 にインプレースアップグレードする

AAP2.1 の情報

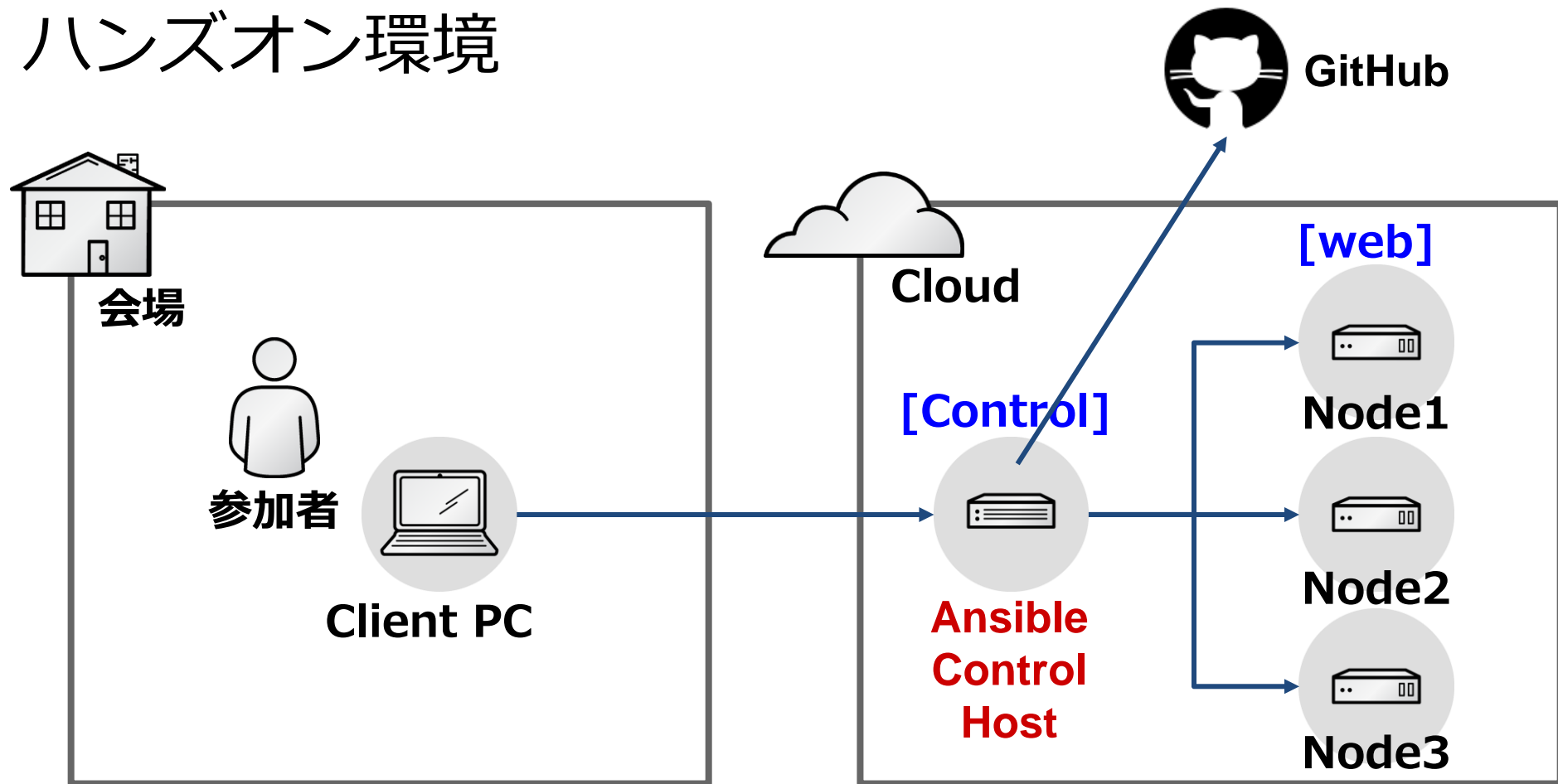
赤帽エンジニアブログもご参照ください！

- [Ansible Automation Platform 2.1 のご紹介 Part1](#)
- [Ansible Automation Platform 2.1 のご紹介 Part2 - アップグレード編（その1）](#)
- [2022年のAnsibleとわたし](#)
- [Ansible Automation Platform 2.1 がリリースされました](#)

...など

Command-line Ansible 演習

ハンズオン環境



1人1環境 (Control Host 1台 & Web Node 3台) をクラウド上に用意

ハンズオン - Ansible コマンドラインの演習

1.1 - 前提条件の確認

1.2 - Ansible の基本

1.3 - 初めての Playbook の作成

1.4 - 変数の使用

1.5 - 条件、ハンドラー、ループ

1.6 - テンプレート

1.7 - Roles

Playbook の例

ansible-playbookコマンドの実行

\$ ansible-playbook -i inventory_file playbook.yml

TARGET
セクション

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

Playbookの説明
対象ホストの指定 (インベントリの中より)
権限昇格の有無

TASKS
セクション

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

Task の説明
利用モジュールを宣言
httpd のインストール

httpdサービスの開始と有効化

index.html ファイルのコピー
 (Ansibleホスト→対象ノード)

実行順序

モジュール

Inventory ファイルの例

- 管理対象サーバを記述

- ホスト名
- IPアドレス
- ssh のユーザ名

- グループ化できる

- 変数の指定も可能

- ansible-playbook コマンドの `-i` オプションで指定する

```
[web]
web-1.example.com
web-2.example.com
web-3.example.com
```

グループ

```
[app]
app-1.example.com
app-2.example.com
```

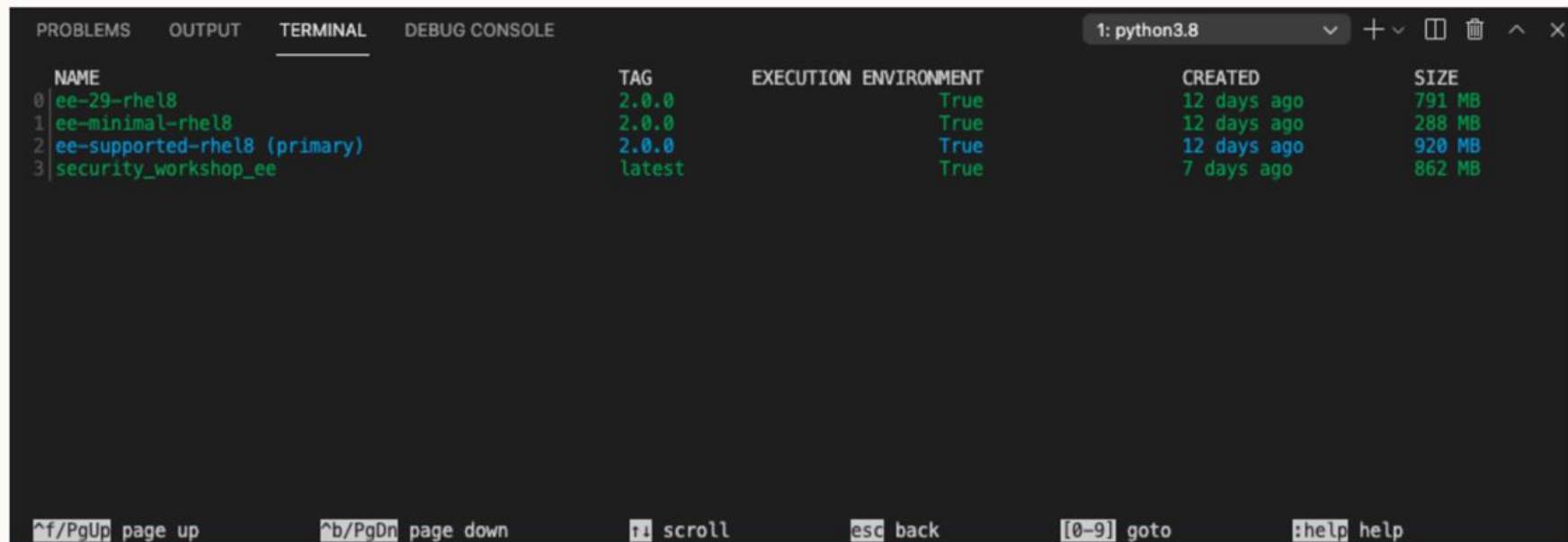
```
[db]
db-1.example.com
```

演習 1.1 - 前提条件の確認

それでは演習をやってみましょう！

ssh 環境にアクセス、ansible-navigatorで操作環境を閲覧します。

```
$ ansible-navigator images
```



The screenshot shows the output of the `ansible-navigator images` command. The interface has tabs for PROBLEMS, OUTPUT, TERMINAL (selected), and DEBUG CONSOLE. A dropdown menu shows '1: python3.8'. The table lists four images with their names, tags, execution environments, creation times, and sizes.

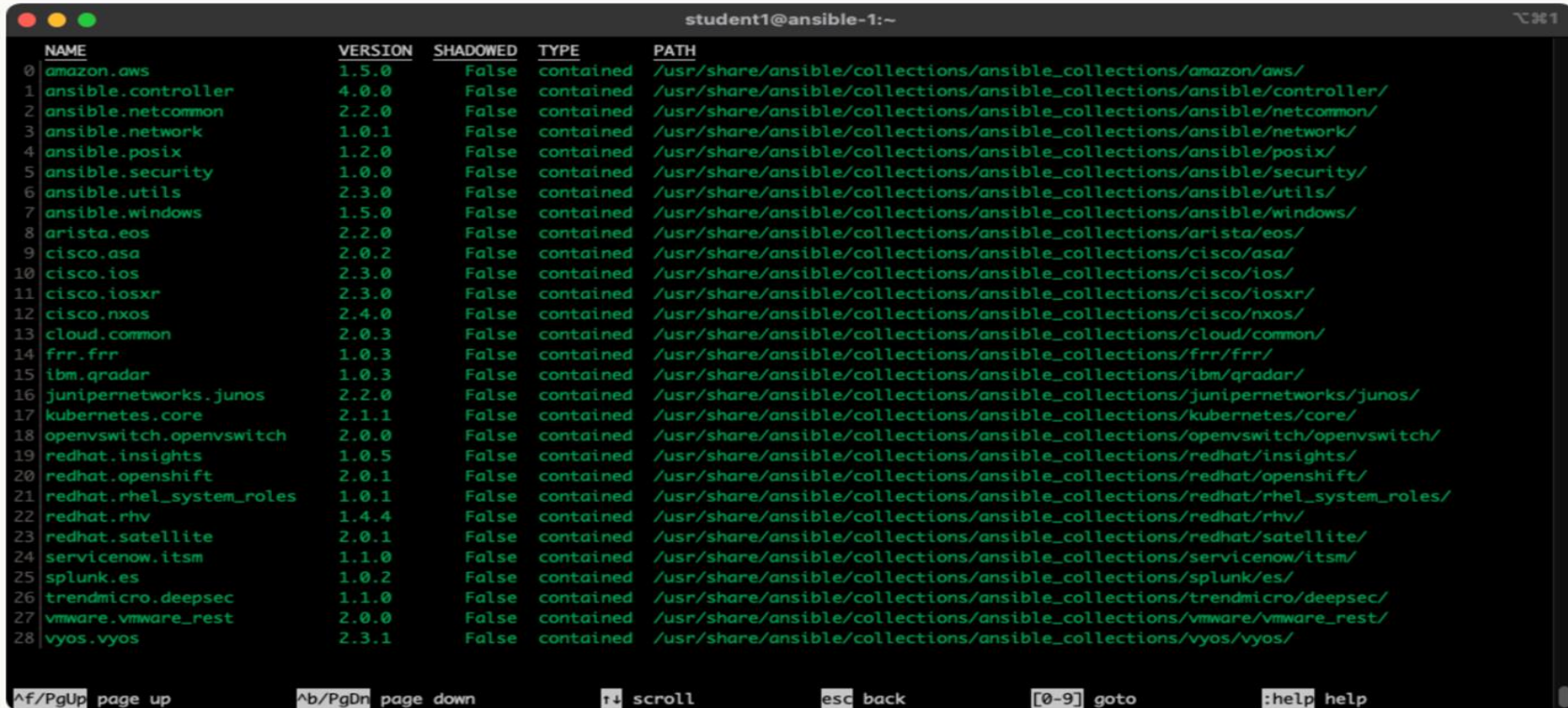
	NAME	TAG	EXECUTION ENVIRONMENT	CREATED	SIZE
0	ee-29-rhel8	2.0.0	True	12 days ago	791 MB
1	ee-minimal-rhel8	2.0.0	True	12 days ago	288 MB
2	ee-supported-rhel8 (primary)	2.0.0	True	12 days ago	920 MB
3	security_workshop_ee	latest	True	7 days ago	862 MB

At the bottom, there are keyboard shortcuts: `^f/PgUp` page up, `^b/PgDn` page down, `↑` scroll, `esc` back, `[0-9]` goto, and `:help` help.

演習 1.2 - Ansible の基本

ansible-navigatorを利用したInventoryの確認、Execution Environmentで特定のモジュール一覧や、コレクション利用方法などを参照します。

```
$ :collections
```



The screenshot shows the ansible-navigator interface with a table of installed collections. The table has columns for NAME, VERSION, SHADOWED, TYPE, and PATH. The interface includes a terminal window with a prompt 'student1@ansible-1:~' and a status bar at the bottom with navigation shortcuts like '^f/PgUp page up', '^b/PgDn page down', 'F4 scroll', 'esc back', '[0-9] goto', and ':help help'.

	NAME	VERSION	SHADOWED	TYPE	PATH
0	amazon.aws	1.5.0	False	contained	/usr/share/ansible/collections/ansible_collections/amazon/aws/
1	ansible.controller	4.0.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/controller/
2	ansible.netcommon	2.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/netcommon/
3	ansible.network	1.0.1	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/network/
4	ansible.posix	1.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/posix/
5	ansible.security	1.0.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/security/
6	ansible.utils	2.3.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/utils/
7	ansible.windows	1.5.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/windows/
8	arista.eos	2.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/arista/eos/
9	cisco.asa	2.0.2	False	contained	/usr/share/ansible/collections/ansible_collections/cisco/asa/
10	cisco.ios	2.3.0	False	contained	/usr/share/ansible/collections/ansible_collections/cisco/ios/
11	cisco.iosxr	2.3.0	False	contained	/usr/share/ansible/collections/ansible_collections/cisco/iosxr/
12	cisco.nxos	2.4.0	False	contained	/usr/share/ansible/collections/ansible_collections/cisco/nxos/
13	cloud.common	2.0.3	False	contained	/usr/share/ansible/collections/ansible_collections/cloud/common/
14	frr.frr	1.0.3	False	contained	/usr/share/ansible/collections/ansible_collections/frr/frr/
15	ibm.qradar	1.0.3	False	contained	/usr/share/ansible/collections/ansible_collections/ibm/qradar/
16	junipernetworks.junos	2.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/junipernetworks/junos/
17	kubernetes.core	2.1.1	False	contained	/usr/share/ansible/collections/ansible_collections/kubernetes/core/
18	openvswitch.openvswitch	2.0.0	False	contained	/usr/share/ansible/collections/ansible_collections/openvswitch/openvswitch/
19	redhat.insights	1.0.5	False	contained	/usr/share/ansible/collections/ansible_collections/redhat/insights/
20	redhat.openshift	2.0.1	False	contained	/usr/share/ansible/collections/ansible_collections/redhat/openshift/
21	redhat.rhel_system_roles	1.0.1	False	contained	/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/
22	redhat.rhv	1.4.4	False	contained	/usr/share/ansible/collections/ansible_collections/redhat/rhv/
23	redhat.satellite	2.0.1	False	contained	/usr/share/ansible/collections/ansible_collections/redhat/satellite/
24	servicenow.itsm	1.1.0	False	contained	/usr/share/ansible/collections/ansible_collections/servicenow/itsm/
25	splunk.es	1.0.2	False	contained	/usr/share/ansible/collections/ansible_collections/splunk/es/
26	trendmicro.deepsec	1.1.0	False	contained	/usr/share/ansible/collections/ansible_collections/trendmicro/deepsec/
27	vmware.vmware_rest	2.0.0	False	contained	/usr/share/ansible/collections/ansible_collections/vmware/vmware_rest/
28	vyos.vyos	2.3.1	False	contained	/usr/share/ansible/collections/ansible_collections/vyos/vyos/

演習 1.3 - 初めての Playbook の作成

```
---
- name: Apache server installed
  hosts: web
  become: yes
  tasks:
    - name: latest Apache version installed
      yum:
        name: httpd
        state: latest
    - name: Apache enabled and running
      service:
        name: httpd
        enabled: true
        state: started
    - name: copy index.html
      copy:
        src: ~/ansible-files/index.html
        dest: /var/www/html/
```

対象ノードや権限昇格、変数などTask
を実行するための条件を記述する領域

タスク と呼ばれる部分
具体的な実行内容をモジュールとともに
記述する部分
ここでは、yum モジュール、service モ
ジュール、copy モジュールがそれぞれ
オプションとともに記述されています。

コメント:

演習のplaybook作成はviベースでガイドされてますが、
vscodeで進めていただくのが簡単です。
vscode画面「Terminal→New Terminal」でコマンドライン
を呼び出し利用できます。

演習 1.3 - 初めての Playbook の作成

- name: Apache server installed

hosts: web

become: yes

tasks:

- name: latest Apache version installed

yum:

name: httpd

state: latest

- name: Apache enabled and running

service:

name: httpd

enabled: true

state: started

- name: copy index.html

copy:

src: ~/ansible-files/index.html

dest: /var/www/html/

スペース2個分あける！

スペース4個分あける！

スペース6個分あける！

プレイブックはわかりやすいのですがお作法にうるさいです。

特に先頭のスペースには、よく注意を払って書いてください！！

※ tab を使ってはいけません！

演習 1.4 - 変数の使用

Ansible のディレクトリ構造を利用した便利な変数 "**{{xxx}}**" の定義

- ~/ansible-files/deploy_index_html.yml ---> プレイブック実行ディレクトリ
- ~/ansible-files/**group_vars/web**・・・グループ [web] に対する変数値を定義
- ~/ansible-files/**host_vars/node2**・・・ホスト "node2" に対する変数の値を定義

deploy_index_html.yml

グループwebでは
stage 変数に
dev を入力

/group_vars/web

stage: dev

ノード "node2" で
は、stage 変数に
prod を入力

/host_vars/node2

stage: prod

```
---
- name: Copy index.html
  hosts: web
  become: yes
  tasks:
    - name: copy index.html
      copy:
        src: ~/ansible-files/{{ stage }}_index.html
        dest: /var/www/html/index.html
```

優先順位 host_vars > group_vars

コメント：
groupに対する変数、ホストに対する変数だとhost
が優先される点に注意！変数は複数設定が可能など
ころが便利ではありますが複雑になる場合も。

演習 1.4 変数 - その他定義場所と優先順位

group_vars / host_vars 以外にも変数は様々なところに定義できます。
その際の優先順位があります。

- | | |
|---|----------------------------------|
| 1. extra vars (-e 指定で実行) | 9. registered vars |
| 2. task vars (only for the task) | 10. host facts |
| 3. block vars (only for tasks in block) | 11. host_vars (playbook) |
| 4. role and include vars | 12. group_vars (playbook) |
| 5. play vars_files | 13. host_vars (inventory) |
| 6. play vars_prompt | 14. group_vars (inventory) |
| 7. play vars | 15. inventory vars |
| 8. set_facts | 16. role defaults |

演習 1.4 Ansible ファクト

Ansible を使っていると、ファクトという名前が良く出てきます。これは対象ノードのプロパティ情報を定義、表示、さらにプレイブックから利用可能にするためのものです。Ansible が非常に強力なところは、取得された膨大なファクト情報を自動的に、`ansible_xxx`という変数に入れてくれるところです。例えば、ファクトの値を元に特定のホストにだけプレイブックを実行！なども可能です。

```
$ ansible localhost -m setup
localhost | success >> {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "192.168.1.37",
      "alias": "wlan0",
      "gateway": "192.168.1.1",
      "interface": "wlan0",
      "macaddress": "c4:85:08:3b:a9:16",
      "mtu": 1500,
      "netmask": "255.255.255.0",
      "network": "192.168.1.0",
      "type": "ether"
    }
  },
}
```

例えば右の例では...

`ansible_default_ipv4.address`

`192.168.1.37`

`ansible_default_ipv4.gateway`

`192.168.1.1`

が自動的に定義され、取得可能です

※収集される変数全体は以下のコマンドで確認可能です。

`$ ansible <host/group_name> -m setup`

演習 1.5 - 条件、ハンドラー、ループ

- 特定のタスクのみ実行 (tag)
- 繰り返し (loop, with_item, with_nested, until ...)
- 条件分岐 (when, notify/handlers ...)
- ファイル操作 (copy, template)
- 他のplaybookの読み込み (include, role, ...)
- 外部情報の参照
 - 環境変数、ファイル など (environment, lookup, vars_prompt,...)
- カスタムモジュールを書いて拡張も可能

演習 1.5 条件分岐 - when

```
---  
- name: Install vsftpd on ftpservers  
  hosts: all  
  become: yes  
  tasks:  
    - name: Install FTP server when host in ftpserver group  
      yum:  
        name: vsftpd  
        state: latest  
        when: inventory_hostname in groups["ftpserver"]
```

hosts: all ですが、when の条件により、グループ "ftpserver" に属するものだけにこのタスクを実行

演習 1.5 ハンドラー - notify/handles

```
---
- name: manage httpd.conf
  hosts: web
  become: yes
  tasks:
    - name: Copy Apache configuration file
      copy:
        src: httpd.conf
        dest: /etc/httpd/conf/
      notify:
        - restart_apache
  handlers:
    - name: restart_apache
      service:
        name: httpd
        state: restarted
```

この場合は、ソースとターゲットの httpd.conf が異なるときに限り、handlers が呼び出されて httpd サービスのリスタートが実施されます。

notify が記述されたタスクが実行された時だけ、handlers に記述された内容が実行される。

演習 1.5 繰り返し - loop

```
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure three users are present
      user:
        name: "{{ item }}"
        state: present
      loop:
        - dev_user
        - qa_user
        - prod_user
```

この場合は、user モジュールにより、

1回目 : dev_user

2回目 : qa_user

3回目 : prod_user

が作成されます。

loop に指定された項目が順に item 変数に入力され繰り返し実行されます。

演習 1.6 - テンプレート

テンプレートモジュールは、変数を含むテンプレートファイルに変数の値を入力した上で対象ホストにファイルコピーを実施します。

motd-facts.j2

```
Welcome to {{ ansible_hostname }}.  
{{ ansible_distribution }} {{ ansible_distribution_version }}  
deployed on {{ ansible_architecture }} architecture.
```

```
---  
- name: Fill motd file with host data  
  hosts: node1  
  become: yes  
  tasks:  
    - template:  
      src: motd-facts.j2  
      dest: /etc/motd  
      owner: root  
      group: root  
      mode: 0644
```

テンプレート内にある4つの変数に、gather_factsで収集された変数が入力され、対象ホスト node1 の /etc/motd ファイルとしてコピーされます。

演習 1.7 - Roles

Role は、プレイブックを複数連携して実行しやすくするための仕組みを提供します。Ansible では1つのプレイブックから他の複数のプレイブックを実行することができますが、この1つ1つをロールと言います。

Roles ディレクトリ構造

```
site.yml
roles/
  common/
    files/
    templates/
    tasks/
    handlers/
    vars/
    defaults/
    meta/
  apache/
    files/
    templates/
    tasks/
    handlers/
```

メインプレイブック

```
# site.yml
---
- hosts: web
  roles:
    - common
    - apache
```

Role はメインのプレイブックから読みだして使います。

tasks/ 配下にメインタスクが入っており、defaults/、vars/ の変数などを使いながら実行されます。このとき、ディレクトリに関する配慮は不要です。

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.