Red Hat Ansible Automation Platform 2.1

Ansible Builder ガイド

Red Hat Ansible Automation Platform 用の一貫性のある再現可能な自動化実行環境を作成するための実行環境ビルダー

Last Updated: 2022-04-03

Red Hat Ansible Automation Platform 2.1 Ansible Builder ガイド

Red Hat Ansible Automation Platform 用の一貫性のある再現可能な自動化実行環境を作成するための実行環境ビルダー

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

フィードバックの提供: 本書を改善するためのご意見がある場合や、エラー発見した場合は、テクニカルサポートに 連絡して Docsコンポーネントを使用して Ansible Automation PlatformJira プロジェクトで問題を作成してください。

目次

前書き

Ansible Builder を使用して、Red Hat Ansible Automation Platform のニーズに合わせて一貫性のある再現可能な自動化実行環境を作成します。

第1章 ANSIBLE BUILDER の概要

1.1. ANSIBLE BUILDER について

Ansible Builder は、さまざまな Ansible Collection で定義されたメタデータと、ユーザーが定義したメタデータを使用して、自動化実行環境を構築するプロセスを自動化するコマンドラインツールです。

1.1.1. Ansible Builder を使用する理由

Ansible Builder が開発される以前、Automation Platform ユーザーは、必要とするすべての依存関係がインストールされたカスタム仮想環境またはコンテナーを作成しようとすると、依存関係の問題と複数のエラーメッセージに対して実行される可能性がありました。

簡単でカスタマイズ可能な定義ファイルを使用することで、Ansible Builder は、Ansible、指定されたコレクション、およびその依存関係をインストールし、ジョブを実行するのに必要なすべての要件がバックグラウンドで満たされるようにします。

第2章 ANSIBLE BUILDER の使用

2.1. ANSIBLE BUILDER のインストール

Red Hat Subscription Management を使用して Ansible Builder をインストールして、Red Hat Ansible Automation Platform サブスクリプションに登録および割り当てることができます。端末で、以下のコマンドを実行します。

```
$ dnf install ansible-builder
```

Python Package Index (PyPI) から Ansible Builder をインストールすることもできます。この設定でインストールするには、以下を実行します。

```
$ pip install ansible-builder
```

2.2. 定義ファイルの構築

Ansible Builder をインストールしたら、自動化実行環境イメージを作成するために Ansible Builder が使用する定義ファイルを作成する必要があります。自動化実行環境イメージを構築する高レベルのプロセスは、Ansible Builder が定義ファイルを読み取って検証してから、**Containerfile** を作成し、最後に **Containerfile** を Podman に渡して、自動化実行環境イメージを作成します。Ansible Builder 用に作成する定義ファイルには **yaml** 形式が使用されており、さまざまなセクションが含まれています。これらのセクションについては、さらに詳しく説明します。

以下は定義ファイルの例です。

例2.1 定義ファイル

```
version: 1

build_arg_defaults: (1)
  ANSIBLE_GALAXY_CLI_COLLECTION_OPTS: "-v"

ansible_config: 'ansible.cfg' (2)

dependencies: (3)
  galaxy: requirements.yml
  python: requirements.txt
  system: bindep.txt

additional_build_steps: (4)
  prepend: |
    RUN whoami
    RUN cat /etc/os-release
  append:
    - RUN echo This is a post-install command!
    - RUN ls -la /etc
```

(1) ビルド引数のデフォルト値の一覧表示

(2) **ansible.cfg** ファイルパスの指定

3 各種要件ファイルの場所の指定

4 追加のカスタムビルド手順用のコマンド

これらの定義ファイルのパラメーターについての詳細は、こちらのセクション を参照してください。

2.3. ビルドの実行およびコマンドの作成

前提条件

- 定義ファイルを作成している

手順

自動化実行環境イメージをビルドするには、以下を実行します。

```
$ ansible-builder build
```

Ansible Builder は、デフォルトでは **execution-environment.yml** という名前の定義ファイルを探しますが、**-f** フラグを使用して異なるファイルパスを引数として指定できます。

```
$ ansible-builder build -f definition-file-name.yml
```

definition-file-name は、定義ファイルの名前を指定します。

2.4. 定義ファイルの内容の内訳

自動化実行環境のコンテナーイメージに含まれるコンテンツを指定するため、Ansible Builder で自動化実行環境を構築するには、定義ファイルが必要です。

次のセクションでは、定義ファイルのさまざまな部分について説明します。

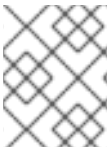
2.4.1. ビルド引数およびベースイメージ

定義ファイルの **build_arg_defaults** セクションは、キーが Ansible Builder への引数のデフォルト値を指定できるディクショナリーです。**build_arg_defaults** で使用できる値の一覧は、以下の表を参照してください。

値	説明
ANSIBLE_GALAXY_CLI_COLLECTION_OPTS	• ユーザーが **–pre** フラグを渡して、プレリリースコレクションのインストールを有効にすることを許可します • **-c** は **verify_ssl** を **false** に設定するのと同じです。
EE_BASE_IMAGE	自動化実行環境の親イメージを指定し、既存のイメージに基づいて新しいイメージを構築できるようにします。

値	説明
EE_BUILDER_IMAGE	コンパイルタイプのタスクに使用されるイメージを指定します。

build_arg_defaults で指定される値は **Containerfile** にハードコーディングされるため、**podman build** を手動で呼び出すとこれらの値が維持されます。



注記

CLI **--build-arg** フラグで同じ変数が指定されている場合は、CLI 値の優先順位が高くなります。

2.4.2. Ansible 設定ファイルパス

ansible.cfg ファイルを使用して、プライベートアカウントのトークンと他の設定を自動化ハブサーバーに渡す場合は、設定ファイルのパス (定義ファイルの場所を基準にした相対パス) を文字列として一覧表示し、ビルドの初期フェーズでビルド引数として有効にします。

ansible.cfg ファイルは以下の例のような形式にする必要があります。

例2.2 ansible.cfg ファイル

```
[galaxy]
server_list = automation_hub

[galaxy_server.automation_hub]
url=https://cloud.redhat.com/api/automation-hub/
auth_url=https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token
token=my_ah_token
```

自動化ハブからコレクションをダウンロードする方法は、関連する Ansible ドキュメントページ を参照してください。

2.4.3. 依存関係

2.4.3.1. Galaxy

galaxy エントリーは、**ansible-galaxy collection install -r ...** コマンドの有効な要件ファイルを参照します。

requirements.yml エントリーは、自動化実行環境定義のディレクトリーからの相対パス、または絶対パスである可能性があります。

requirements.yml ファイルの内容は以下のようになります。

例2.3 Galaxy の requirements.yml ファイル

```
collections:
  - geerlingguy.java
  - kubernetes.core
```

2.4.3.2. Python

定義ファイルの **python** エントリーは、**pip install -r ...** コマンドの有効な要件ファイルを参照します。

requirements.txt エントリーは、Collection がすでに Python の依存関係としてリストされているように追加の Python 要件をインストールするファイルです。自動化実行環境定義のフォルダーのディレクトリーからの相対パス、または絶対パスとして記載されている可能性があります。**requirements.txt** ファイルの内容は、**pip freeze** コマンドの標準出力と同様に、以下の例のような形式にする必要があります。

例2.4 Python の requirements.txt ファイル

```
boto>=2.49.0
botocore>=1.12.249
pytz
python-dateutil>=2.7.0
awxkit
packaging
requests>=2.4.2
xmltodict
azure-cli-core==2.11.1
python_version >= '2.7'
collection community.vmware
google-auth
openshift>=0.6.2
requests-oauthlib
openstacksdk>=0.13
ovirt-engine-sdk-python>=4.4.10
```

2.4.3.3. システム

定義の **システム** エントリーは、bindep 要件ファイルを指定しています。このファイルは、コレクションに依存関係として含まれていないシステムレベルの依存関係をインストールします。自動化実行環境定義のフォルダーのディレクトリーからの相対パス、または絶対パスとして記載されている可能性があります。

これは、**libxml2** パッケージおよび **subversion** パッケージをコンテナーに追加する **bindep.txt** ファイルの例です。

例2.5 bindep.txt ファイル

```
libxml2-devel [platform:rpm]
subversion [platform:rpm]
```

2.4.4. 追加のカスタムビルドの手順

prepend コマンドと **append** コマンドは、**additional_build_steps** セクションで指定できます。これにより、メインのビルド手順が実行される前または後に実行される **Containerfile** にコマンドが追加されます。

additional_build_steps の構文は以下のいずれかである必要があります。

- 複数行の文字列

  例2.6 複数行の文字列エントリー

  ```
  RUN whoami
  RUN cat /etc/os-release
  ```

- リスト

  例2.7 リストエントリー

  ```
  - RUN echo This is a post-install command!
  - RUN ls -la /etc
  ```

2.5. 任意のビルドコマンド引数

-t フラグは、自動化実行環境イメージに特定の名前を付けます。たとえば、以下のコマンドは **my_first_ee_image** という名前のイメージを構築します。

```
$ ansible-builder build -t my_first_ee_image
```

複数の定義ファイルがある場合は、**-f** フラグで、使用する定義ファイルを指定できます。

```
$ ansible-builder build -f another-definition-file.yml -t another_ee_image
```

上記の例 では、Ansible Builder は、デフォルトの **execution-environment.yml** の代わりに **another-definition-file.yml** ファイルで提供される仕様を使用して、**another_ee_image** という名前の自動実行環境イメージを構築します。

build コマンドで使用できるその他の仕様とフラグについては、**ansible-builder build --help** と入力して、追加オプションのリストを表示してください。

2.6. イメージの構築なしで CONTAINERFILE の作成

イメージをビルドせずに共有可能な **Containerfile** を作成するには、以下を実行します。

```
$ ansible-builder create
```

第3章 自動化実行環境の公開

3.1. 実行環境コンテナイメージを自動化ハブにプッシュ

前提条件

- 自動化ハブで実行環境のパーミッションがあり、新しいコンテナーを作成したり、既存のコンテナーにプッシュすることを許可している。

コンテナーレジストリーはコンテナーイメージを保存するためのリポジトリーです。自動化実行環境イメージを構築すると、そのコンテナーイメージを自動化ハブのインスタンスのレジストリー部分にプッシュする準備が整います。

自動化ハブの URL を使用して以下のコマンドを実行し、Podman にログインし、ユーザー名、パスワード、および自動化のハブ URL に置き換えてください。

```
$ podman login -u=username -p=password automation-hub-url
```

Podman にログインしたら、以下のコマンドを実行してコンテナーイメージを自動化ハブのコンテナーレジストリーにプッシュします。

```
$ podman push automation-hub-url/ee-image-name
```

注記

自動化実行環境イメージ名は、**ansible-builder build** コマンドに **-t** 引数で指定します。**-t** フラグを使用してカスタムイメージ名を指定しない場合、デフォルトのイメージタグは **ansible-execution-env:latest** になります。

3.2. 保護されたレジストリーからのプル

パスワードまたはトークンで保護されるレジストリーからコンテナーイメージをプルするには、自動化コントローラーで認証情報を作成します。

手順

1. 自動化コントローラーに移動します。
2. サイドメニューバーの **Resources** > **Credentials** をクリックします。
3. **Add** をクリックして、新規の認証情報を作成します。
4. 認証 URL、ユーザー名、およびパスワードを指定します。**保存** をクリックします。

詳細は、実行環境のドキュメントの「Pulling from Protected Registries」セクションを参照してください。

第4章 RED HAT ANSIBLE AUTOMATION PLATFORM が提供する既存のベース EE を基に構築

4.1. システムレベルの依存関係の収集

bindep 形式は、クロスプラットフォーム要件を指定する方法を提供します。最低限、コレクションが、**[platform:rpm]** に必要な要件を指定することが想定されます。

以下は、有効な **bindep.txt** ファイルからのコンテンツの例です。

例4.1 bindep.txt ファイル

```
python38-devel [platform:rpm compile]
subversion [platform:rpm]
git-lfs [platform:rpm]
```

複数のコレクションからのエントリーは単一ファイルに統合されます。これは **bindep** により処理され、その後 **dnf** に渡されます。プロファイルのない要件や、ランタイム要件がイメージにインストールされません。

4.2. PIP ベースの要件に関する注意

Python 要件ファイルは、複雑な構文をサポートするために **requirements-parser** ライブラリーを使用して単一のファイルに統合されます。同じパッケージ名を与える別々のコレクションからのエントリーは、制約が結合された状態で同じエントリーに結合されます。

ansible-builder でとくに無視されるパッケージ名が複数あります。コレクションのリストがこれらを一覧表示するには、統合ファイルに含まれません。これには、テストパッケージと、Ansible 自体を提供するパッケージが含まれます。

完全なリストは、ansible_builder.requirements.py モジュールの **EXCLUDE_REQUIREMENTS** にあります。

4.3. 既存の実行環境イメージのカスタマイズ

Ansible コントローラーには、以下の 3 つのデフォルト実行環境が同梱されています。

- **Ansible 2.9**: Controller モジュール以外のコレクションがインストールされない
- **minimal**: Ansible Runner とともに最新の Ansible 2.11 リリースが含まれるが、コレクションや他の追加コンテンツは含まれない
- **EE Supported**: Red Hat がサポートするコンテンツがすべて含まれる

これらの環境は多くの自動化ユースケースに対応しますが、追加の項目を追加して、特定のニーズに合わせてこれらのコンテナーをカスタマイズできます。以下の手順では、**kubernetes.core** コレクションを **ee-minimal** デフォルトイメージに追加します。

手順

1. Podman を使用して**registry.redhat.io** にログインします。

-

```
$ podman login -u="[username]" -p="[token/hash]" registry.redhat.io
```

2. 自動化実行環境イメージのプル

```
podman pull registry.redhat.io/ansible-automation-platform-20-early-access/ee-minimal-rhel8:2.0.1-8
```

3. Ansible Builder ファイルを、**ee-minimal** をベースとする新しい実行環境イメージに追加する追加のコンテンツを指定するように設定します。

   a. たとえば、Kubernetes Core Collection を Galaxy から イメージに追加するには、以下のように **requirements.yml** ファイルを入力します。

```
collections:
  - kubernetes.core
```

   b. 定義ファイルとそのコンテンツの詳細は、「定義ファイルの内訳」セクションを参照してください。

4. 実行環境定義ファイルの **EE_BASE_IMAGE** フィールドで、元の **ee-minimal** コンテナーファイルへのパスを指定します。その場合には、最終的な **execution-environment.yml** ファイルは以下のようになります。

例4.2 カスタマイズされた execution-environment.yml ファイル

```
version: 1

build_arg_defaults:
  EE_BASE_IMAGE: 'example.registry.com/my-base-ee'

dependencies:
  galaxy: requirements.yml
```

注記

この例では、自動化ハブからの認定コレクションではなく、コミュニティーバージョンの **kubernetes.core** を使用するため、**ansible.cfg** を作成したり、定義ファイルで参照したりする必要はありません。

5. 以下のコマンドを使用して、新しい実行環境イメージを構築します。

```
$ ansible-builder build -t registry.redhat.io/[username]/new-ee
```

ここで、**[username]** はユーザー名を指定し、**new-ee** は新しいコンテナーイメージの名前を指定します。

   a. **podman images** コマンドを使用して、新しいコンテナーイメージが一覧に表示されていることを確認します。

例4.3 new-ee イメージを使用した podman images コマンドの出力

```
REPOSITORY          TAG     IMAGE ID      CREATED         SIZE
localhost/new-ee    latest  f5509587efbb  3 minutes ago  769 MB
```

6. Ansible Navigator で新たに作成した実行環境イメージを確認します。
7. 自動化ハブで使用するイメージにタグを付けます。

```
$ podman tag registry.redhat.io/_[username]_/_new-ee_ [automation-hub-IP-address]/_[username]_/_new-ee_
```

8. Podman を使用して自動化ハブにログインします。

注記

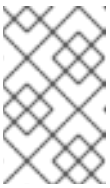
コンテナーをプッシュするには、自動化ハブの **管理** または適切なコンテナーリポジトリーのパーミッションが必要です。詳細は、Red Hat Ansible Automation Platform ドキュメント の「 **Managing containers in private automation hub**」を参照してください。

```
$ podman login -u="[username]" -p="[token/hash]" [automation-hub-IP-address]
```

9. イメージを自動化ハブのコンテナーレジストリーにプッシュします。

```
$ podman push [automation-hub-IP-address]/_[username]_/_new-ee_
```

10. 新しいイメージを自動化コントローラーインスタンスにプルします。
    a. 自動化コントローラーに移動します。
    b. ナビゲーションバーから、**Administration** > **Execution Environments** の順にクリックします。
    c. **Add** をクリックします。
    d. 適切な情報を入力して **Save** を押して新規イメージにプルします。



注記

自動化ハブのインスタンスがパスワードまたはトークンで保護されている場合は、適切なコンテナーレジストリーの認証情報が設定されていることを確認してください。