TECHNICAL WHITE PAPER: **August 2024**

VMware Cloud Foundation Automation ABX Actions for Ansible Automation Platform

Deployment Guide

Table of contents

Revision History

Date	Version	Description	Modified By
08/22/2025	1	Initial release	• Charles Lee • Dharmesh Bhatt

About the VMware Cloud Foundation Automation ABX Actions for Ansible Automation Platform Deployment Guide

The *VMware Cloud Foundation Automation ABX Actions for Ansible Automation Platform Deployment Guide* ("Guide") provides instructions on installing and configuring sample ABX actions to enable Aria Automation interfacing with Red Hat Ansible Automation Platform.

Intended Audience

This Guide is intended for data center cloud administrators who manage VMware Cloud Foundation Automation environment in their organization. The information in this guide is written for experienced data center cloud administrators who are familiar with:

- VMware Cloud Foundation Automation: How to administer, configure, and use the VMware Cloud Foundation Automation Assembler.
- Red Hat Ansible Automation Platform: Creating and managing users, organizations, and interfacing with revision control platforms, such as GitLab and GitHub.
- Python 3: Working knowledge of Python 3 and Python packages.

Related Publications

- Getting Started with VMware Cloud Foundation Automation
- *VMware Cloud Foundation Automation ABX Actions for Ansible Automation Platform User Guide*
- Using Automation Assembler
- Red Hat Ansible Automation Platform operations guide

Preparing the Environment

This solution expects the following components already installed and configured.

- VMware Aria Automation 8.16+
- Red Hat Ansible Automation Platform 2.x

VMware Cloud Foundation Automation

VMware Cloud Foundation Automation™ is a cloud infrastructure automation solution powering VMware Cloud Foundation. It delivers a self-service private cloud with governance and resource lifecycle management across on-premises data centers or on any supported public cloud. It leverages a service-driven cloud computing interface, a policy controlled self-service catalog, Infrastructure as Code (IaC), and infrastructure pipelining. It enables Cloud Ops teams to maintain frictionless governance and control while empowering developers with a high level of agility and flexibility.

Red Hat Ansible Automation Platform

Red Hat® Ansible® Automation Platform is a unified solution for strategic automation. It combines the security, features, integrations, and flexibility needed to scale automation across domains, orchestrate essential workflows, and optimize IT operations to successfully adopt enterprise AI.

Solution Configuration

This section introduces the resources and configurations:

• Architecture diagram

• Software resources

Architecture Diagram

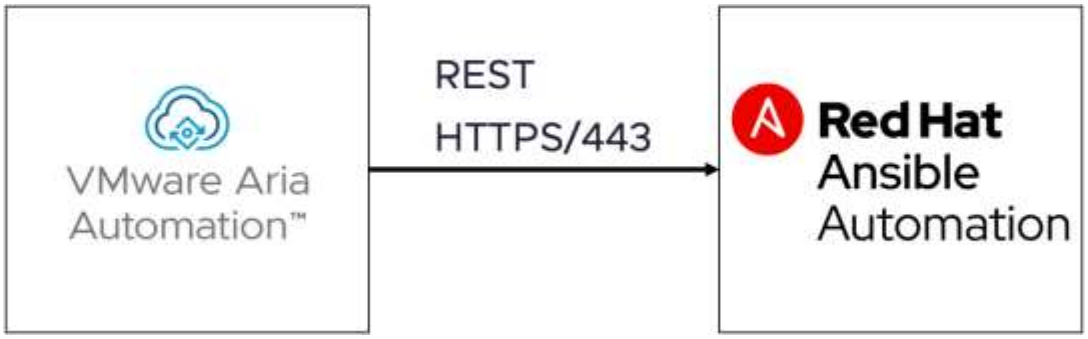


Figure 1 AAP API System Diagram

Software Resources

The following table lists the software resources used in this solution.

Table 1 - Software Resources

Software	Version	Purpose
VMware Cloud Foundation Automation	8.16+	VMware Automation Infrastructure as a Service (IaaS) Platform
Red Hat Ansible Automation Platform	2.4+	Red Hat Ansible Configuration Management (CM) server
AAP API	0.4.1	Open-source ABX actions: *https://github.com/vmware-workloads/aap-api*

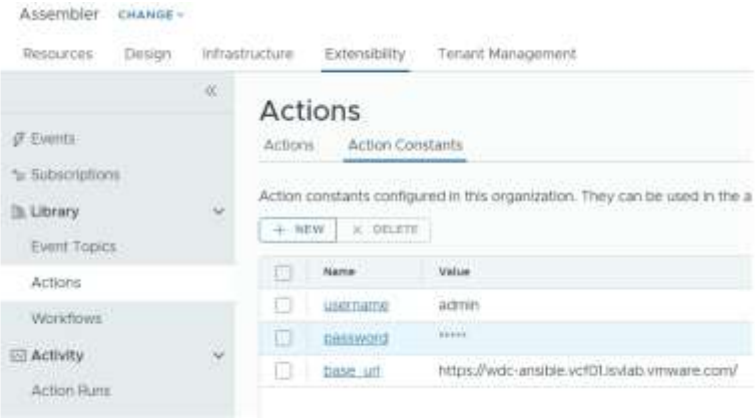
Deployment

This section outlines the steps to deploy and configure the Ansible Automation Platform API in Aria Automation Assembler. These steps are based and tested on Aria Automation 8.16.

Action Constants

We start by creating action constants for the Ansible Automation Platform URL and credentials. These constants define which Ansible Automation Server will be used and credentials with sufficient privilege to run the

1. Open Aria Automation Assembler.
2. Select **Extensibility**, then **Actions Constants**.
3. Add the following parameters (**Figure 2**):



Action constants configured in this organization. They can be used in the a

	Name	Value
□	username	admin
□	password	*****
□	base_url	https://wdc-ansible.vcf01.isvlab.vmware.com/

Figure 2 VMware Cloud Foundation Automation Actions Constants

- *username*: user defined in Ansible Automation Platform
- *password*: password for the user
- *base_url*: url of the Ansible Automation Platform server

ABX Actions

In the second step, we create the required Assembler ABX actions. There are two different ways to create the actions, either using the zip bundles available on the GitHub repo, or manually creating the actions by copying the required source code. Both methods achieve the same outcome, with different

Zip Bundles

Installation using zip bundles provides a simple installation method that includes all the required Python dependencies. This method simplifies the distribution of the actions and provides a solution for air-gaped environments where Aria Automation would not be able to download dependencies (e.g. 'requests').

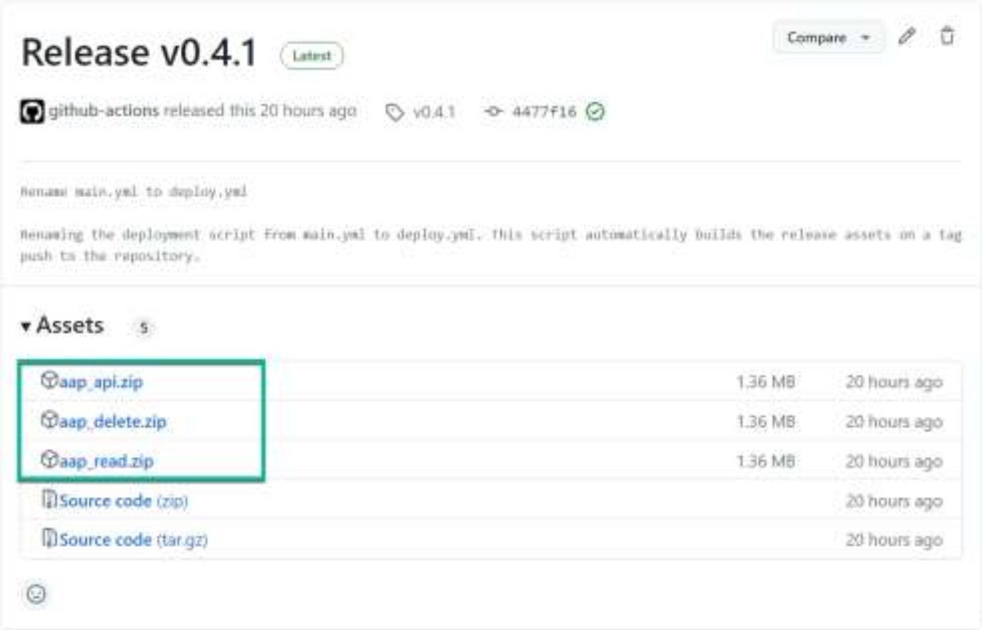
The zip bundles can be found at the following URL:

https://github.com/vmware-workloads/aap-api/releases

Note. The zip bundles are provided as downloads on the project releases. The bundles can also be created by using a published procedure. For more details, please reference the following article

Create a ZIP package for Python runtime extensibility actions.

1. Download the required zip bundles from the GitHub repository releases.



Release v0.4.1

github-actions released this 20 hours ago v0.4.1 4477f16

Rename main.yml to deploy.yml

Renaming the deployment script from main.yml to deploy.yml. This script automatically builds the release assets on a tag push to the repository.

Assets 5

aap_api.zip	1.36 MB	20 hours ago
aap_delete.zip	1.36 MB	20 hours ago
aap_read.zip	1.36 MB	20 hours ago
Source code (zip)		20 hours ago
Source code (tar.gz)		20 hours ago

Figure 3 AAP API zip bundles

- ○ aap_api.zip
- ○ aap_read.zip
- ○ aap_delete.zip

2. In Aria Automation Assembler, select **Extensibility**, **Library**, then **Actions**. Select **New.**

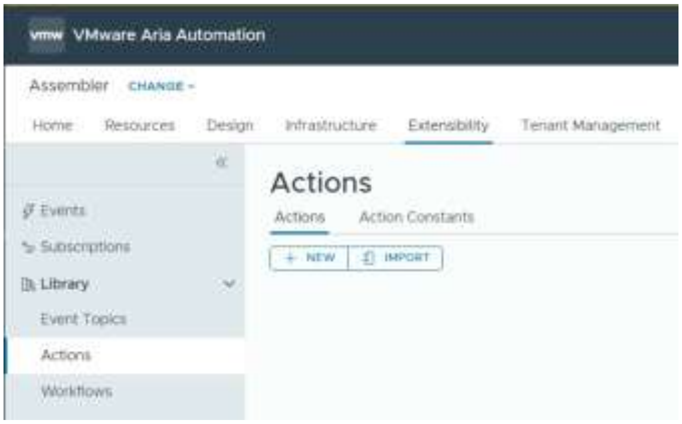
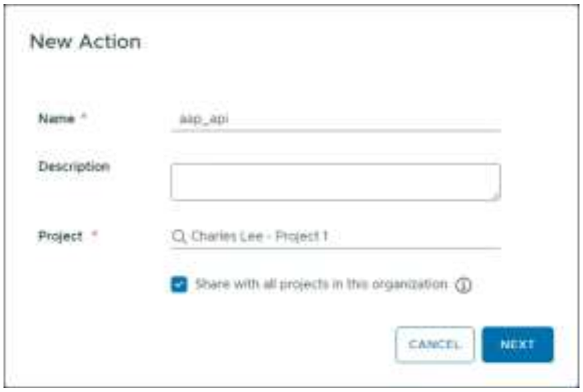


Figure 4 VCF Automation Assembler Actions

3. At the ***New Action*** window, enter the following information, then click **Next**.



New Action

Name *: aap_api

Description:

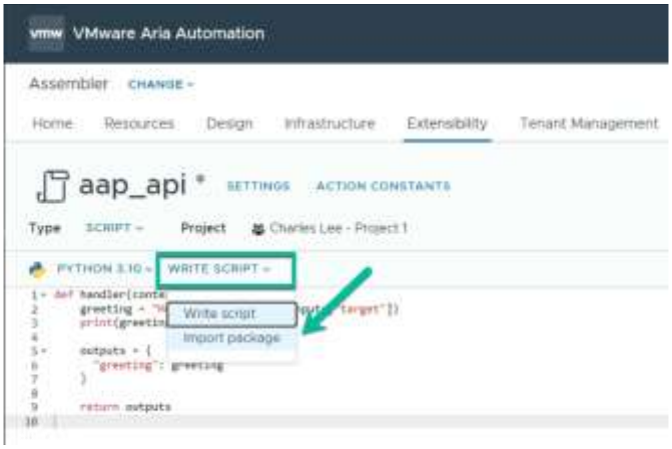
Project *: Charles Lee - Project 1

☑ Share with all projects in this organization

Figure 5 VCF Automation Assembler New Action

- **Name**: aap_api
- **Project**: <select the appropriate project>
- **Share with all projects in the organization**: <enable as required>

4. In the action properties, select the drop-down, then select **Import Package.**



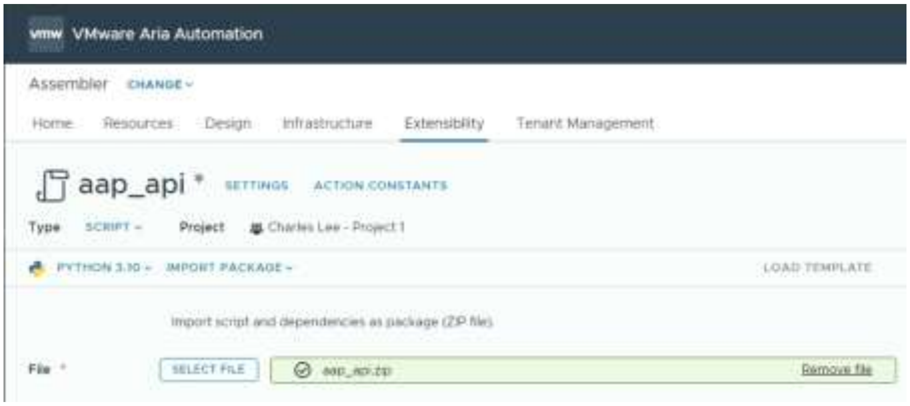
Type: SCRIPT

Project: Charles Lee - Project 1

```
def handler(conte
    greeting = "H                    puts["target"])
    print(greetin

    outputs = {
      "greeting": greeting
    }

    return outputs
```

Figure 6 VCF Automation Assembler Action Import Package

5. Click the ***Select File*** button, then choose the appropriate zip bundle.



aap_api *

Type SCRIPT

Project Charles Lee - Project 1

Import script and dependencies as package (ZIP file)

File * aap_api.zip

Figure 7 VCF Automation Assembler New Action Select File

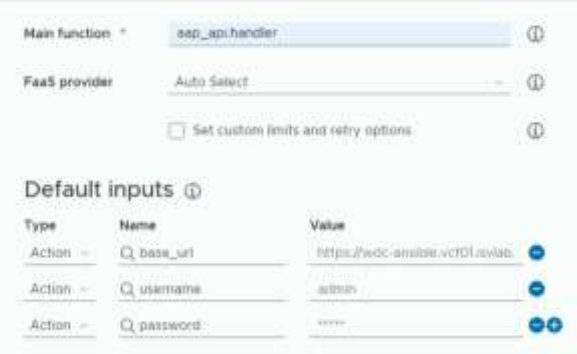
- ○ aap_api → aap_api.zip
- ○ aap_read → aap_read.zip
- ○ aap_delete → aap_delete.zip

6. At the action properties, select the following action constants, fill the remaining fields using information based on the table below (Table 2), then click **Save** and **Close**.

Note: The action constants are the variables created in the previous **Action Constants** section.

Table 2 New Action Parameters

Custom Resource	Action Name	Main Function	FaaS Provider
Create	aap_api	aap_api.handler	Auto Select
Read	aap_read	aap_read.handler	Auto Select
Update			
Destroy	aap_delete	aap_delete.handler	Auto Select



Main function * aap_api.handler

FaaS provider Auto Select

□ Set custom limits and retry options

Default inputs

Type	Name	Value
Action	base_url	https://wdc-ansible.vcf01.isvlab.
Action	username	admin
Action	password	*****

Figure 8 VCF Automation Assembler New Action Parameters

- ○ Action Constant: base_url

- Action Constant: username
- Action Constant: password

Source Code

Installation using source code creates the actions by copying the script code. This method allows the code to be easily edited for development and debugging. It is important to note that when using source code Assembler will need to download additional libraries and imports declared in the action. This approach is therefore not suitable for deployments that have no internet access or located in air-gapped environments.

The source code can be found at the following URL:

https://github.com/vmware-workloads/aap-api/

Note: It is also possible to clone the repository and add synchronization via the 'Integration > GitHub' facility in the infrastructure section of aria automation (see *Using Automation Assembler*).

1. In Aria Automation Assembler, open **Extensibility**, then select **Actions**.

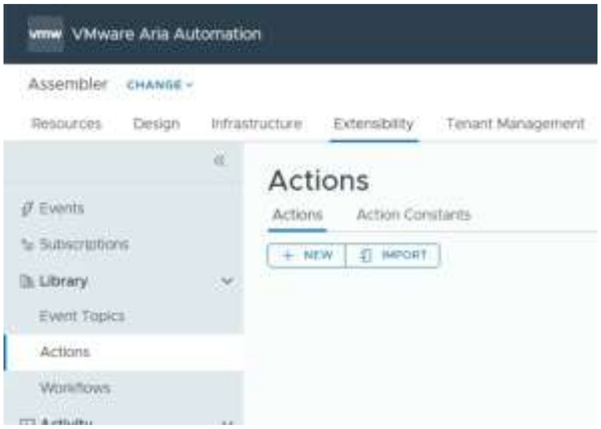
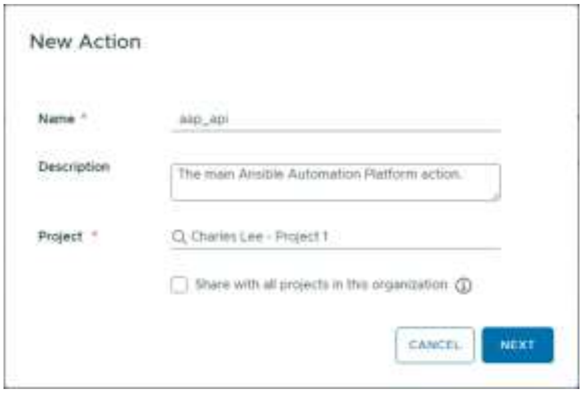


Figure 9 VCF Automation Assembler Actions

2. At the ***New Action*** window, enter the following information, then click **Next**.


New Action

Name *: aap_api

Description: The main Ansible Automation Platform action.

Project *: Charles Lee - Project 1

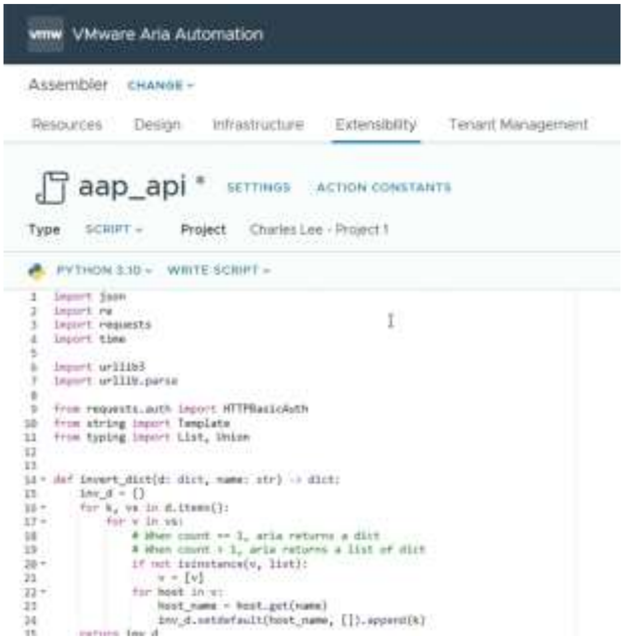
□ Share with all projects in this organization

Figure 10 VCF Automation Assembler New Action

- **Name**: aap_api

- **Project**: <select the appropriate project>
- **Share with all projects in the organization**: <enable as required>

3. At the new action select the following and copy the script code into the script area.



aap_api *

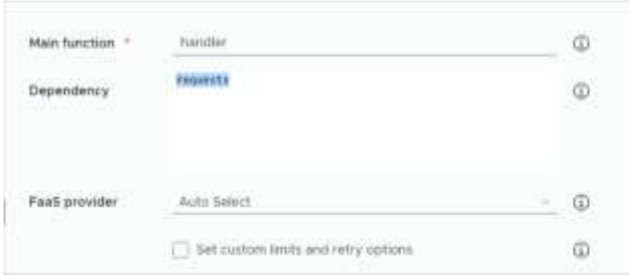
Type: SCRIPT
Project: Charles Lee - Project 1

```python
import json
import re
import requests
import time

import urllib3
import urllib.parse

from requests.auth import HTTPBasicAuth
from string import Template
from typing import List, Union


def invert_dict(d: dict, name: str) -> dict:
    inv_d = {}
    for k, vs in d.items():
        for v in vs:
            # When count == 1, aria returns a dict
            # When count > 1, aria returns a list of dict
            if not isinstance(v, list):
                v = [v]
            for host in v:
                host_name = host.get(name)
                inv_d.setdefault(host_name, []).append(k)
    return inv_d
```

Figure 11 VCF Automation Assembler New Action Python Script

- Select Python 3.10
- Select Write Script
- Copy and paste the script code in the code section

4. At the New Action properties enter the following parameters.



Main function *: handler

Dependency: requests

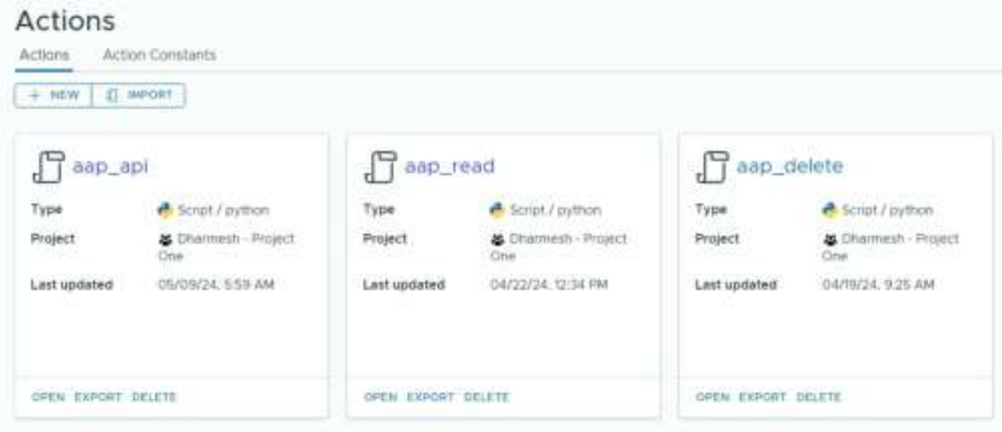
FaaS provider: Auto Select

☐ Set custom limits and retry options

Figure 12 VCF Automation Assembler New Action Parameters

- Main function: handler
- FaaS provider: Auto Select
- Dependency:
  - requests

5. Repeat steps 1 to 4 for the following actions scripts to create the required actions.



aap_api

Type: Script / python
Project: Dharmesh - Project One
Last updated: 05/09/24, 5:59 AM

aap_read

Type: Script / python
Project: Dharmesh - Project One
Last updated: 04/22/24, 12:34 PM

aap_delete

Type: Script / python
Project: Dharmesh - Project One
Last updated: 04/19/24, 9:25 AM

Figure 13 VCF Automation ABX Actions for Ansible Integration Actions

Table 3 New Action Source Code Scripts and Handlers

Custom Resource	Script Name	Main Function	FaaS Provider
Create	aap_api.py	handler	Auto Select
Read	aap_read.py	handler	Auto Select
Update			
Destroy	aap_delete.py	handler	Auto Select

Custom Resource

When creating a cloud template in Automation Assembler, the resource type palette includes resource types for the supported cloud account and integration endpoints. There are use cases where the creation of new resource types can provide required additional functionality and integration. Once implemented, new custom resource types can be added to the design canvas to create cloud templates that support any design or deployment needs. This section outlines how to create a new Custom Resource that implements the Ansible Automation Platform integration actions.

1. In Aria Automation Assembler, open **Design**, then select **Custom Resources**.

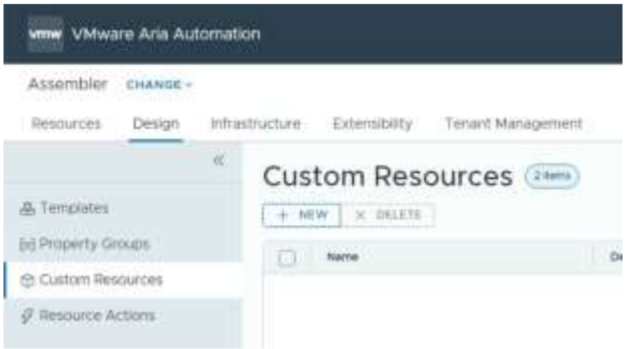
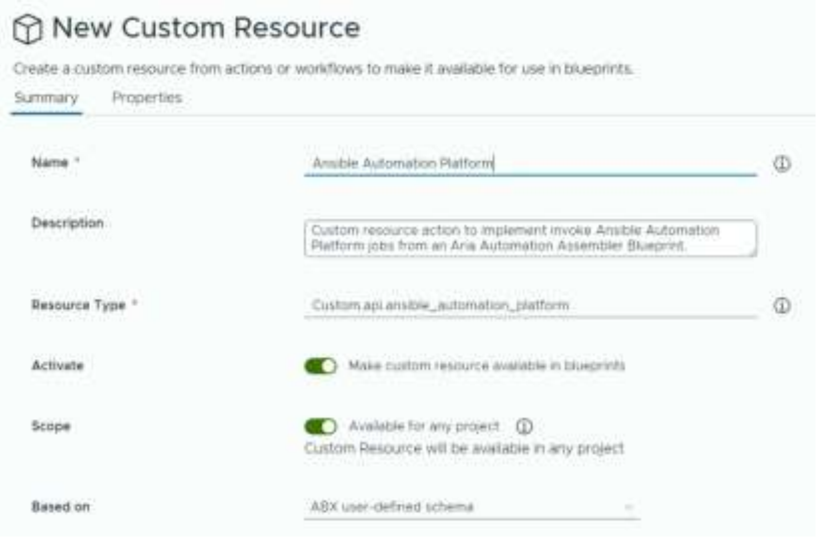


Figure 14 VCF Automation Assembler Custom Resources

2. Select **New** and enter the following.



New Custom Resource

Create a custom resource from actions or workflows to make it available for use in blueprints.

Name *: Ansible Automation Platform

Description: Custom resource action to implement invoke Ansible Automation Platform jobs from an Aria Automation Assembler Blueprint.

Resource Type *: Custom.api.ansible_automation_platform

Activate: Make custom resource available in blueprints

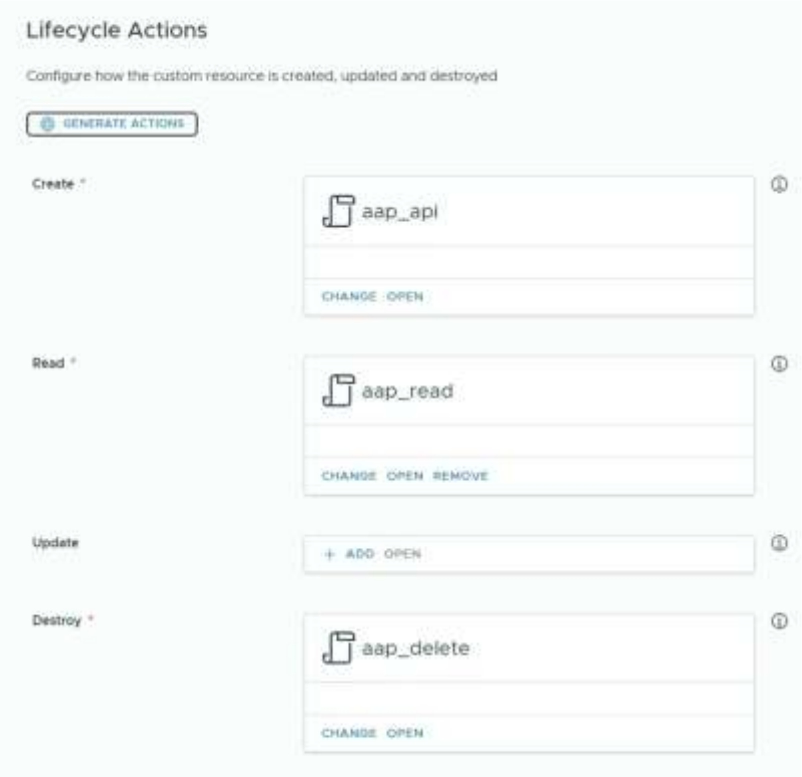
Scope: Available for any project
Custom Resource will be available in any project

Based on: ABX user-defined schema

Figure 15 VCF Automation Assembler New Custom Resource

- **Name**: *Ansible Automation Platform*
- **Resource Type**: *custom.api.ansible_automation_platform*
- **Activate**: *enabled*
- **Scope**: *<as required>*
- **Based on**: *ABX user defined schema*

3. Scroll down to the **Lifecycle Actions** and select the ABX actions previously created, then click **Create**. Use the following table (Table 4) to select the correct mapping between the Custom Resource Action and ABX Action.



Lifecycle Actions

Configure how the custom resource is created, updated and destroyed

Create * aap_api

Read * aap_read

Update

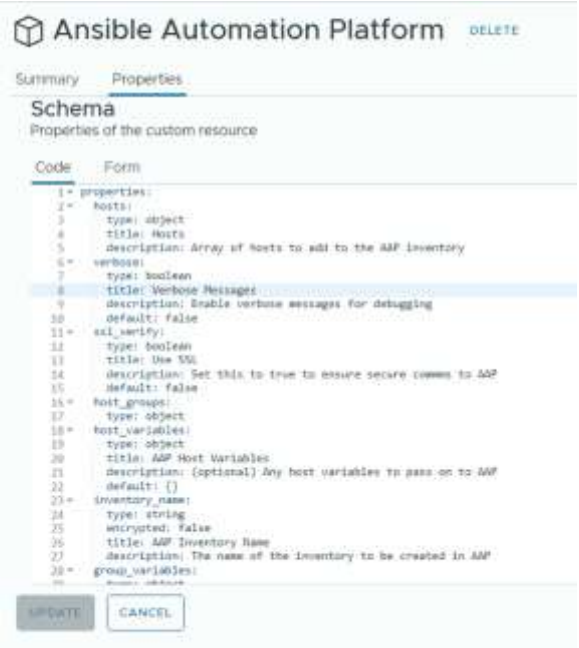
Destroy * aap_delete

Figure 16 Custom Resource Lifecycle Actions

Table 4 Custom Resource Action Mapping

Custom Resource Action	ABX Action	Description
Create	aap_api	Action called when the resources is created.
Read	aap_read	Action called when the resources state is read.
Update		
Destroy	aap_delete	Action called when the resources is deleted.

4. Select the **Properties** tab, select the **Code** tab, then create the following properties.

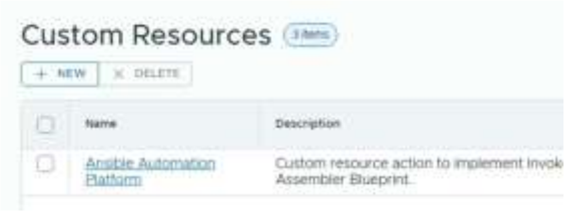

Ansible Automation Platform
DELETE
Summary
Properties
Schema
Properties of the custom resource
Code
Form
UPDATE
CANCEL

```
properties:
  hosts:
    type: object
    title: Hosts
    description: Array of hosts to add to the AAP inventory
  verbose:
    type: boolean
    title: Verbose Messages
    description: Enable verbose messages for debugging
    default: false
  ssl_verify:
    type: boolean
    title: Use SSL
    description: Set this to true to ensure secure comms to AAP
    default: false
  host_groups:
    type: object
  host_variables:
    type: object
    title: AAP Host Variables
    description: (optional) Any host variables to pass on to AAP
    default: {}
  inventory_name:
    type: string
    encrypted: false
    title: AAP Inventory Name
    description: The name of the inventory to be created in AAP
  group_variables:
```

Figure 17 Custom Resource Properties

```
properties:
  hosts:
    type: object
  verbose:
    type: boolean
    default: false
  host_groups:
    type: object
  host_variables:
    type: object
    default: {}
  inventory_name:
    type: string
    encrypted: false
  group_variables:
    type: object
    default: {}
  job_template_name:
    type: string
  organization_name:
    type: string
    default: Default
  inventory_variables:
    type: object
    default: {}
```

5. Once complete, select **Create** to save and close the new custom resource. The new custom resource is listed on the Custom Resources page.



□	Name	Description
□	Ansible Automation Platform	Custom resource action to implement Invok Assembler Blueprint.

Figure 18 Ansible Automation Platform Custom Resource

Reference

- *VMware Cloud Foundation*
- *VMware Aria Automation*
- *Red Hat Ansible Automation Platform*
- *VMware Aria Ansible Automation API*

About the Authors

Charles Lee, *Product Marketing Engineer,* in the VCF Technical Marketing team of VMware Cloud Foundation Business Unit of VMware by Broadcom, wrote the original version of this paper.

Dharmesh Bhatt, *Product Marketing Engineer,* in the VCF Technical Marketing team of VMware Cloud Foundation Business Unit of VMware by Broadcom, wrote the original version of this paper