

# Verwenden des VMware vRealize Orchestrator Client

06. Oktober 2020

vRealize Orchestrator 8.2

Die aktuellste technische Dokumentation finden Sie auf der VMware-Website unter:

<https://docs.vmware.com/de/>

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

**VMware Global, Inc.**  
Zweigniederlassung Deutschland  
Willy-Brandt-Platz 2  
81829 München  
Germany  
Tel.: +49 (0) 89 3706 17 000  
Fax: +49 (0) 89 3706 17 333  
[www.vmware.com/de](http://www.vmware.com/de)

Copyright © 2008-2020 VMware, Inc. Alle Rechte vorbehalten. [Urheberrechts- und Markenhinweise](#).

# Inhalt

<b>1</b>	<b>Verwenden des VMware vRealize Orchestrator Client</b>	<b>6</b>
<b>2</b>	<b>Der VMware vRealize Orchestrator Client</b>	<b>7</b>
	Dashboard zur Nutzung von vRealize Orchestrator Client	8
	Inhaltsorganisation im vRealize Orchestrator Client	9
	Erstellen eines Ordners oder Unterordners im vRealize Orchestrator Client	10
<b>3</b>	<b>Einrichten des vRealize Orchestrator Client</b>	<b>13</b>
	Verwalten von Rollen und Gruppen im vRealize Orchestrator Client	13
	Zuweisen von Rollen im vRealize Orchestrator Client	15
	Erstellen von Gruppen im vRealize Orchestrator Client	16
	Konfigurieren von vRealize Orchestrator Client-Rollen in vRealize Automation	16
	Versionsverlauf von vRealize Orchestrator-Objekten	17
	Wiederherstellen eines Workflows auf eine frühere Version	18
	Visueller Vergleich zwischen Workflow-Versionen	19
	Zurücksetzen der vRealize Orchestrator-Inhaltsbestandsliste auf einen vorherigen Status mit Git	20
<b>4</b>	<b>vRealize Orchestrator-Anwendungsbeispiele</b>	<b>22</b>
	Vorgehensweise zum Integrieren von Amazon Web Services in vRealize Orchestrator mithilfe von Python	22
	Erstellen eines anfänglichen Python-Skripts	23
	Erstellen der Amazon Web Services-Aktion	24
	Debuggen der Amazon Web Services-Aktion	25
	Aktualisieren der Amazon Web Services-Aktion	29
	Vorgehensweise zum Verwenden von Git-Zweigen zum Verwalten meiner vRealize Orchestrator-Objektbestandsliste	29
	Vorbereiten der GitLab-Umgebung	30
	Konfigurieren einer Verbindung mit einem Git-Repository	31
	Weitergeben von Änderungen an ein Git-Repository	32
	Vorgehensweise zur Verwendung von Drittanbietermodulen zum Aufrufen der vRealize Automation-Projekt-API	35
	Erstellen eines Python-Skripts zum Aufrufen der vRealize Automation-Projekt-API	35
	Erstellen eines Node.js-Skripts zum Aufrufen der vRealize Automation-Projekt-API	37
	Erstellen eines PowerShell-Skripts zum Aufrufen der vRealize Automation-Projekt-API	40
<b>5</b>	<b>Verwalten von Workflows</b>	<b>43</b>
	Standard-Workflows in der vRealize Orchestrator-Workflow-Bibliothek	44
	Erstellen von Workflows im vRealize Orchestrator Client	44

Bearbeiten von Workflows und Aktionen über den übergeordneten Workflow	45
vRealize Orchestrator-Eingabeformular-Designer	45
Erstellen des Dialogfelds „Workflow-Eingabeparameter“ im vRealize Orchestrator Client	46
Eingabeparametereigenschaften im vRealize Orchestrator Client	47
Verwenden von Aktionen zum Validieren von vRealize Orchestrator-Workflow-Eingaben	48
Anforderungen zur Benutzerinteraktion im vRealize Orchestrator Client	49
Planen von Workflows im vRealize Orchestrator Client	50
Bearbeiten einer geplanten Aufgabe im vRealize Orchestrator Client	51
<b>6 Verwalten von Aktionen</b>	<b>52</b>
Erstellen von Aktionen im vRealize Orchestrator Client	52
Ausführen und Debuggen von Aktionen	54
Ausführen von Aktionen im vRealize Orchestrator Client	54
Debug-Aktionen im vRealize Orchestrator Client	54
Kernkonzepte für Python-, Node.js- und PowerShell-Skripts	55
Laufzeitgrenzwerte für Python-, Node.js- und PowerShell-Skripts	57
<b>7 Verwalten von Konfigurationselementen</b>	<b>59</b>
Erstellen von Konfigurationselementen im vRealize Orchestrator Client	59
<b>8 Verwalten von Richtlinien</b>	<b>61</b>
Erstellen und Anwenden von Richtlinien im vRealize Orchestrator Client	61
Richtlinienelemente im vRealize Orchestrator Client	62
Verwalten von Richtlinienausführungen im vRealize Orchestrator Client	63
<b>9 Verwalten von Ressourcenelementen</b>	<b>64</b>
<b>10 Verwalten von Paketen</b>	<b>65</b>
Erstellen eines Pakets im vRealize Orchestrator Client	65
Exportieren eines Pakets im vRealize Orchestrator Client	66
Importieren eines Pakets im vRealize Orchestrator Client	67
<b>11 Fehlerbehebung im vRealize Orchestrator Client</b>	<b>69</b>
Metrikdaten im vRealize Orchestrator Client	69
Erstellen von Profilen für Workflows im vRealize Orchestrator Client	69
Verwenden des vRealize Orchestrator-System-Dashboards	70
Verwenden der Workflow-Token-Wiedergabe im vRealize Orchestrator Client	71
Validieren von vRealize Orchestrator-Workflows	72
Validieren eines Workflows und Beheben von Validierungsfehlern im vRealize Orchestrator Client	73

[Debuggen von Workflow-Skripts im vRealize Orchestrator Client](#) 74

[Debuggen von Workflows nach Schemaelement](#) 75

# Verwenden des VMware vRealize Orchestrator Client

1

Unter *Verwenden des VMware vRealize Orchestrator Client* erhalten Sie Informationen über die Funktionen zur Workflow-Automatisierung des vRealize Orchestrator Client.

## Zielgruppe

Diese Informationen sind für erfahrene Systemadministratoren bestimmt, die ein Tool zum Ausführen und Verwalten von vRealize Orchestrator-Workflows suchen.

# Der VMware vRealize Orchestrator Client

# 2

Verwenden Sie den vRealize Orchestrator Client, um vRealize Orchestrator-Dienste und -Objekte zu verwalten.

Sie können vRealize Orchestrator-Objekte mithilfe des vRealize Orchestrator Client erstellen und verwalten. Sie finden den vRealize Orchestrator Client unter [https://your\\_orchestrator\\_FQDN / orchestration-ui](https://your_orchestrator_FQDN/orchestration-ui).

## **REST-API-Kommunikation**

Der vRealize Orchestrator Client wird auf dem vRealize Orchestrator Control Center-Server ausgeführt. Der Client kommuniziert mit der vRealize Orchestrator-REST-API über einen REST-Proxy.

## **Workflow-Verwaltung**

Workflows werden im vRealize Orchestrator Client erstellt, bearbeitet, geplant, ausgeführt und gelöscht.

## **Aktionsverwaltung**

Aktionen werden im vRealize Orchestrator Client erstellt, bearbeitet und gelöscht. Der Aktionseditor unterstützt die automatische Vervollständigung für gängige Skript-Elemente, die im vRealize Orchestrator-API-Explorer enthalten sind.

## **Richtlinienverwaltung**

Richtlinien werden im vRealize Orchestrator Client erstellt, bearbeitet, ausgeführt und gelöscht.

## **Konfigurationsverwaltung**

Konfigurationselemente werden im vRealize Orchestrator Client erstellt, ausgeführt und gelöscht.

## **Ressourcenverwaltung**

Ressourcenelemente werden im vRealize Orchestrator Client exportiert, importiert und aktualisiert.

## **Git-Integration**

Erstellen Sie eine Integration in ein Git-Repository und verwenden Sie die Integration, um die Entwicklung von Workflows und anderen vRealize Orchestrator über mehrere Bereitstellungen hinweg zu verwalten. Weitere Informationen finden Sie unter [Vorgehensweise zum Verwenden von Git-Zweigen zum Verwalten meiner vRealize Orchestrator-Objektbestandsliste](#).

## Metrikdaten

Über das System-Dashboard und die Profilerstellungsfunktion von vRealize Orchestrator Client sammeln Sie nützliche Metrikdaten zu Ihrer vRealize Orchestrator-Umgebung und Ihren Workflows.

## Paketverwaltung

Sie können Pakete mit vRealize Orchestrator-Objekten über den vRealize Orchestrator Client erstellen, löschen, exportieren und importieren.

## Berechtigungsverwaltung

Benutzer mit Administratorrechten können Benutzern im vRealize Orchestrator Client Rollen zuweisen und diese zu Gruppen hinzufügen.

## API-Explorer

Erkunden Sie die im vRealize Orchestrator Client verfügbaren API-Befehle.

Dieses Kapitel enthält die folgenden Themen:

- [Dashboard zur Nutzung von vRealize Orchestrator Client](#)
- [Inhaltsorganisation im vRealize Orchestrator Client](#)

# Dashboard zur Nutzung von vRealize Orchestrator Client

Das vRealize Orchestrator Client-Dashboard bietet ein nützliches Tool für die Überwachung, Verwaltung und Fehlerbehebung von vRealize Orchestrator Client-Workflows.

Die Informationen im vRealize Orchestrator Client-Dashboard werden in fünf Bereichen angezeigt.

Fenster	Beschreibung
Workflow-Ausführungen	Liefert visuelle Daten über die Anzahl der aktiven, wartenden und fehlgeschlagenen Workflow-Ausführungen.
Favoriten-Workflows	Zeigt die Workflows an, die den Favoriten hinzugefügt wurden.
Warten auf Eingabe	Zeigt die ausstehenden Workflow-Ausführungen an, die eine weitere Benutzerinteraktion erfordern. Diese Workflows werden auch im Benachrichtigungsmenü in der oberen rechten Ecke der Benutzeroberfläche angezeigt.



Fenster	Beschreibung
Letzte Workflow-Ausführungen	Verwalten Sie die letzten Workflow-Ausführungen. Zeigt den Namen und Status sowie das Start- und Enddatum der Workflow-Ausführung an.
Erfordern Aufmerksamkeit	Zeigt fehlgeschlagene Workflow-Ausführungen und Leistungsmetriken zur Workflow-Ausführung an.

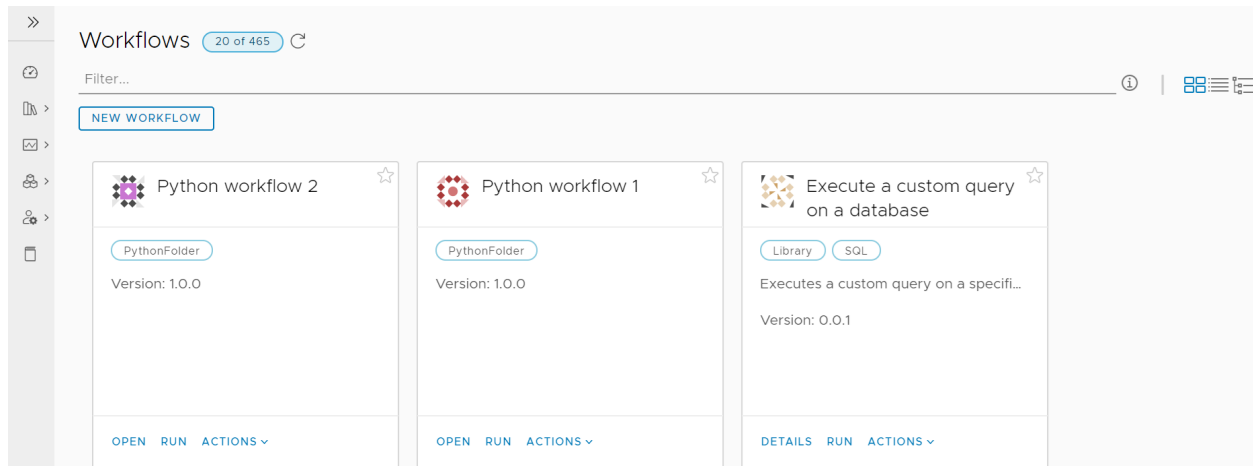
## Inhaltsorganisation im vRealize Orchestrator Client

Verwalten Sie, wie Ihre vRealize Orchestrator-Objektbestandsliste im vRealize Orchestrator Client angezeigt wird.

Der vRealize Orchestrator Client unterstützt drei unterschiedliche Ansichtstypen für Objekte wie Workflows, Aktionen, Richtlinien, Ressourcen und Konfigurationen: Kartenansicht, Listenansicht und Baumansicht. Sie können den aktuellen Ansichtstyp in der oberen rechten Ecke der Seite ändern.

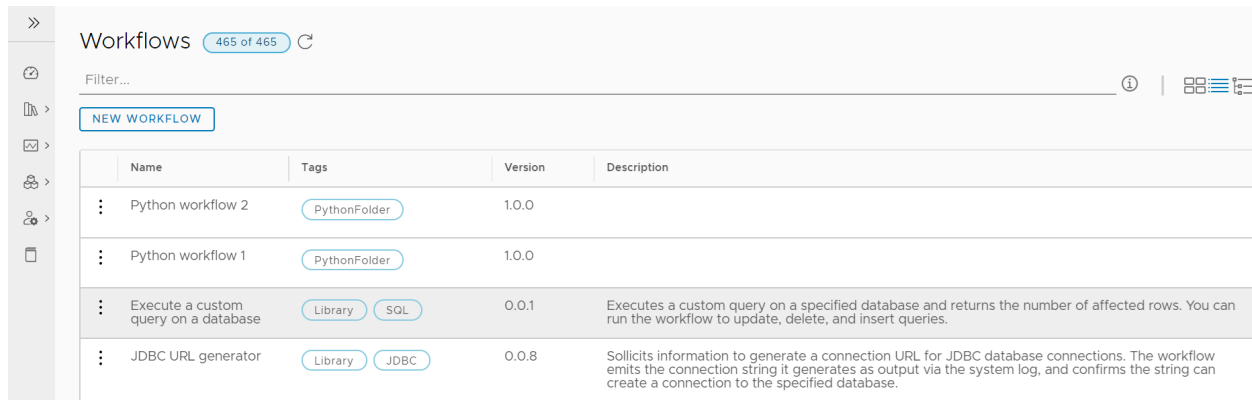
### Kartenansicht

Die Kartenansicht ist der Standardansichtstyp, der im vRealize Orchestrator Client verwendet wird. Informationen zum einzelnen Bestandslistenobjekt, z. B. einem Workflow, werden in einem eigenen Kartenelement angezeigt.



### Listenansicht

In der Listenansicht werden Informationen zu Ihren vRealize Orchestrator-Objekten angezeigt, die als eine Liste organisiert sind. Für weitere Informationen zu den Aktionen, die Sie für das Objekt durchführen können, klicken Sie auf das vertikale Auslassungssymbol links neben dem Objekt.



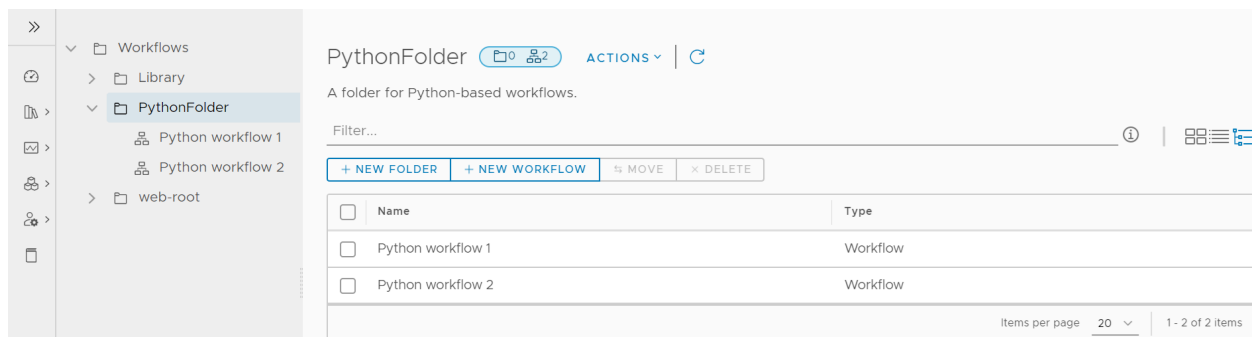
## Baumansicht

Sie können die Bestandsliste der Objekte in hierarchischen Ordnern in der Baumansicht verwalten. Jeder vRealize Orchestrator-Objekttyp hat einen Ordner auf Root-Ebene. Im Root-Ordner können Sie keine neuen Objekte wie z. B. Workflows erstellen. Sie müssen getrennte Ordner erstellen, die unter dem Root-Ordner organisiert sind. Jeder Ordner enthält Tools, die Ihnen dabei helfen, dessen Inhalt zu verwalten, z. B. einen Inhaltsfilter.

**Hinweis** Jeder Ordner verfügt über einen eigenen Inhaltsfilter. Sie können Inhalte nicht über Ordner hinweg filtern.

Weitere Informationen zu Ordnern finden Sie unter [Erstellen eines Ordners oder Unterordners im vRealize Orchestrator Client](#).

**Hinweis** Wenn Sie ein Objekt in der Baumansicht auswählen, wird es im schreibgeschützten Modus geöffnet. Um den Objekthalt wie beispielsweise Workflow-Variablen oder das Workflow-Schema zu bearbeiten, klicken Sie im oberen Optionsmenü auf **Bearbeiten**.




## Erstellen eines Ordners oder Unterordners im vRealize Orchestrator Client

Organisieren Sie Ihre vRealize Orchestrator-Objekte mithilfe einer hierarchischen Ordnerstruktur.

Sie können Ordner und Unterordner erstellen, um die folgenden Typen von vRealize Orchestrator-Objekten zu organisieren:

- Workflows
- Aktionen
- Richtlinien
- Konfigurationselemente
- Ressourcenelemente

#### Verfahren


- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie im linken Navigationsbereich eine Objektseite aus, z. B. **Workflows**.
- 3 Wählen Sie oben rechts das Symbol für die Baumansicht () aus.
- 4 (Optional) Um einen Unterordner zu erstellen, wählen Sie in r Baumansicht auf der linken Seite einen übergeordneten Ordner aus.
- 5 Klicken Sie auf **Neuer Ordner**.
- 6 Geben Sie einen Namen und eine Beschreibung ein und klicken Sie auf **Speichern**.
- 7 Fügen Sie dem neu erstellten Ordner Objekte oder Unterordner hinzu.
- 8 (Optional) Um den Ordernamen zu bearbeiten, wählen Sie **Aktionen > Bearbeiten**.

### Verschieben von Objekten und Ordnern im vRealize Orchestrator Client

Organisieren Sie Ihre vRealize Orchestrator-Inhalte neu, indem Sie sie in einen anderen Ordner verschieben.

Sie können keine Aktionen zwischen Aktionsmodulen oder Objekte in einen Root-Ordner verschieben. Der Root-Ordner enthält die Hauptobjektordner und zugehörigen Unterordner, kann aber nicht zum Speichern von Objekten verwendet werden.

#### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie im linken Navigationsbereich eine Objektseite aus, z. B. **Workflows**.
- 3 Wählen Sie oben rechts das Symbol für die Baumansicht () aus.
- 4 Erweitern Sie die Baumansicht und wählen Sie das Objekt oder den Ordner aus, das bzw. den Sie verschieben möchten.

- 5 Ziehen Sie das Objekt oder den Ordner in den neuen übergeordneten Ordner.

---

**Hinweis** Sie können Objekte auch direkt über den Objekteditor in neue Ordner verschieben. Klicken Sie auf der Registerkarte **Übersicht** auf **Ordner auswählen** und wählen Sie den neuen übergeordneten Ordner für das Objekt aus. Eine andere Option zum Verschieben ist die Auswahl von Objekten aus der Tabelle auf der Seite „Ordner“. Diese Option ist nützlich, um Batch-Vorgänge durchzuführen, die mehrere vRealize Orchestrator-Objekte umfassen.


---

## Löschen eines Ordners oder Unterordners im vRealize Orchestrator Client

Löschen Sie veraltete Ordner oder Unterordner aus Ihrem vRealize Orchestrator Client.

Sie können den entsprechenden Ordner auf Root-Ebene der einzelnen vRealize Orchestrator-Objekttypen nicht löschen.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie im linken Navigationsbereich eine Objektseite aus, z. B. **Workflows**.
- 3 Wählen Sie oben rechts das Symbol für die Baumansicht () aus.
- 4 Aktivieren Sie das Kontrollkästchen neben dem Ordner, den Sie löschen möchten.

---

**Hinweis** Um einen Unterordner zu löschen, wählen Sie den übergeordneten Ordner in der Baumansicht aus und klicken Sie dann auf das Kontrollkästchen.

---

- 5 Klicken Sie auf **Löschen**.
- 6 Wenn der ausgewählte Ordner leer ist:
  - a Bestätigen Sie, dass Sie den Ordner löschen möchten.
  - b Klicken Sie auf **Löschen**.
- 7 Wenn der ausgewählte Ordner vRealize Orchestrator Client-Objekte oder Unterordner enthält:
  - a Bestätigen Sie, dass Sie den Ordner löschen möchten.
  - b Klicken Sie auf **Löschen**.

Sie erhalten sinngemäß die Meldung Das Element „your\_foldername“ konnte nicht gelöscht werden: Ordner „your\_foldername“ ist nicht leer.

- c Um den Ordner und seinen gesamten Inhalt zu löschen, klicken Sie auf **Löschen erzwingen**.
- d Bestätigen Sie, dass Sie den Ordner löschen möchten, und klicken Sie auf **Löschen**.

---

**Hinweis** Sie können auch einen Batch-Löschvorgang durchführen, indem Sie mehrere Objekte in der Tabelle im Ordnermenü auswählen.

---

# Einrichten des vRealize Orchestrator Client

# 3

Um die Funktionalität des vRealize Orchestrator Client umfassend nutzen zu können, müssen Sie Ihre Benutzerberechtigungen konfigurieren und wissen, wie Sie den Versionsverlauf zum Verwalten Ihrer Objekte verwenden.

Dieses Kapitel enthält die folgenden Themen:

- [Verwalten von Rollen und Gruppen im vRealize Orchestrator Client](#)
- [Versionsverlauf von vRealize Orchestrator-Objekten](#)

## Verwalten von Rollen und Gruppen im vRealize Orchestrator Client

Als Administrator können Sie den vRealize Orchestrator Client verwenden, um Benutzerrollen und Gruppenberechtigungen für vRealize Orchestrator-Funktionen und -Inhalte festzulegen.

Nach der Authentifizierung der vRealize Orchestrator-Instanz kann der Administrator Berechtigungen festlegen, die den Zugriff auf die Funktionen und Inhalte steuern. Die Berechtigungen im vRealize Orchestrator Client sind in Rollenverwaltung und Gruppenberechtigungen aufgeteilt. Mit der Rollenverwaltung steuern Sie die vRealize Orchestrator Client-Funktionen, die von Benutzern angezeigt und verwendet werden können. Mit Gruppenberechtigungen steuern Sie die vRealize Orchestrator Client-Inhalte, die von Benutzern angezeigt und verwendet werden können. Der mit Gruppenberechtigungen abgedeckte Inhaltzugriff umfasst Workflows, Aktionen, Richtlinien, Konfigurationselemente und Ressourcenelemente. Sie können Gruppen verwenden, um Benutzer in gemeinsamen Projekten zu organisieren. Sie können beispielsweise eine Gruppe mit Benutzern erstellen, die an der Entwicklung eines benutzerdefinierten vRealize Orchestrator-Plug-Ins arbeiten.

---

**Hinweis** Der Zugriff auf vorkonfigurierte vRealize Orchestrator-Inhalte wie Standard-Workflows und -Aktionen ist für alle Benutzer freigegeben, sofern nicht anderweitig über Gruppenberechtigungen konfiguriert.

---

### Rollen

Die clientseitige Rollenverwaltung ist nur für vRealize Orchestrator-Instanzen verfügbar, die mit vSphere authentifiziert wurden und die eine vRealize Automation-Lizenz verwenden. Für Bereitstellungen, die eine vRealize Automation-Authentifizierung verwenden, müssen Sie die Identitäts- und Zugriffsverwaltungsfunktion von vRealize Automation verwenden. Weitere Informationen finden Sie unter [Konfigurieren von vRealize Orchestrator Client-Rollen in vRealize Automation](#).

Rolle	Beschreibung
<b>Administrator</b>	<p>Kann auf alle vRealize Orchestrator Client-Funktionen und -Inhalte zugreifen, einschließlich der von bestimmten Gruppen erstellten Inhalte. Verantwortlich für das Festlegen von Benutzerrollen, das Erstellen und Löschen von Gruppen und das Hinzufügen von Benutzern zu Gruppen.</p> <p><b>Hinweis</b> Mandantenadministratoren aus der vRealize Automation-Umgebung, die zur Authentifizierung von vRealize Orchestrator verwendet wird, verfügen standardmäßig über <b>Administratorrechte</b>.</p>
<b>Workflow-Designer</b>	<p>Kann eigene vRealize Orchestrator Client-Inhalte erstellen, ausführen, bearbeiten und löschen. Kann eigene Inhalte zur zugewiesenen Gruppe hinzufügen. Hat keinen Zugriff auf die Verwaltungs- und Fehlerbehebungsfunktionen des vRealize Orchestrator Client.</p>

**Hinweis** vRealize Automation-Benutzer ohne vordefinierte Rolle können sich weiterhin beim vRealize Orchestrator Client anmelden, haben aber nur eingeschränkten Zugriff auf Clientfunktionen. Wenn die Benutzer Teil einer Gruppe sind, können sie Inhalte anzeigen und ausführen, die dieser Gruppe zugeordnet sind.

## Gruppen

Gruppenberechtigungen im vRealize Orchestrator Client können verwendet werden, um mehrere Benutzer zu verbinden, die an einem gemeinsamen vRealize Orchestrator arbeiten, z. B. bei der Entwicklung eines benutzerdefinierten Plug-Ins.

Gruppenbenutzerberechtigungen	Beschreibung
<b>Ausführen und bearbeiten</b>	Nur für vRealize Orchestrator-Instanzen verfügbar, die eine vRealize Automation-Lizenz verwenden. Kann vRealize Orchestrator-Objekte für die Verwendung in der Gruppe erstellen, bearbeiten, hinzufügen und ausführen.
<b>Ausführen</b>	Kann vRealize Orchestrator-Objekte anzeigen und ausführen, die in der Gruppe enthalten sind.

**Hinweis** Gruppenberechtigungen sind an das Rollenverwaltungssystem im vRealize Orchestrator Client gebunden. Benutzer ohne vordefinierte Rolle können beispielsweise über die Berechtigungen **Ausführen und bearbeiten** verfügen. Sie können eigene oder Gruppeninhalte aber nur anzeigen und ausführen. Das Erstellen, Bearbeiten und Hinzufügen von Inhalten ist nicht möglich.

## Zuweisen von Rollen im vRealize Orchestrator Client

Als Administrator können Sie Benutzer zum vRealize Orchestrator Client hinzufügen und festlegen, welche Funktionen von den Benutzern angezeigt und verwendet werden können.

Mit der Rollenverwaltung wird der Zugriff von Benutzern aus dem vRealize Orchestrator-Identitätsanbieter auf die Funktionen des vRealize Orchestrator Client gesteuert. Die Rollenverwaltung umfasst sowohl die vRealize Orchestrator Client-Benutzeroberfläche als auch die API-Funktionen.

**Hinweis** Die clientseitige Rollenverwaltung ist nur für vRealize Orchestrator-Instanzen verfügbar, die mit vSphere authentifiziert wurden und die eine vRealize Automation-Lizenz verwenden. Informationen zum Zuweisen von Rollen zu vRealize Orchestrator, die mit vRealize Automation authentifiziert wurden, finden Sie unter [Konfigurieren von vRealize Orchestrator Client-Rollen in vRealize Automation](#).

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.
- 2 Navigieren Sie zu **Administration > Rollenverwaltung**.
- 3 Klicken Sie auf **Hinzufügen**.
- 4 Suchen Sie nach dem Benutzer oder der Gruppe, den bzw. die Sie dem vRealize Orchestrator Client hinzufügen möchten.
- 5 Wählen Sie die Rolle des Benutzers aus. Weitere Informationen zu Rollen finden Sie unter [Verwalten von Rollen und Gruppen im vRealize Orchestrator Client](#).
- 6 Klicken Sie auf **Speichern**.

## Erstellen von Gruppen im vRealize Orchestrator Client

Als Administrator können Sie Gruppen zum Festlegen der vRealize Orchestrator-Inhalte verwenden, die von Benutzern im vRealize Orchestrator Client angezeigt werden können und die für den Zugriff bereitstehen.

Mit dem vRealize Orchestrator Client können Sie Gruppenberechtigungen für vRealize Orchestrator-Workflows, -Aktionen, -Richtlinien, -Konfigurationselemente, -Ressourcenelemente und -Pakete festlegen.

---

**Hinweis** Benutzer von vRealize Orchestrator-Instanzen, die mit vSphere authentifiziert wurden, können die Gruppenberechtigung **Ausführen** haben.

---

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.
- 2 Wechseln Sie zu **Administration > Gruppen**.
- 3 Klicken Sie auf **Neue Gruppe**.
- 4 Fügen Sie auf der Registerkarte **Übersicht** einen Namen und eine Beschreibung für die Gruppe hinzu.
- 5 Klicken Sie auf der Registerkarte **Benutzer** auf **Hinzufügen**.
  - a Suchen Sie nach einem Benutzer, den Sie der Gruppe hinzufügen möchten.
  - b Weisen Sie dem Benutzer Gruppenberechtigungen zu.
  - c Klicken Sie auf **Hinzufügen**.
- 6 Fügen Sie der Gruppe auf der Registerkarte **Elemente** vRealize Orchestrator-Objekte hinzu.

---

**Hinweis** Sie können ein Objekt auch vorhandenen Gruppen hinzufügen, wenn das Objekt im vRealize Orchestrator Client erstellt wird. Um das Objekt hinzuzufügen, wählen Sie die Gruppe aus dem Dropdown-Menü **Aufrufbar von** auf der Registerkarte **Übersicht/Allgemein** des Objekteditors aus.

---

- 7 Klicken Sie auf **Speichern**.

## Konfigurieren von vRealize Orchestrator Client-Rollen in vRealize Automation

Sie können Dienstrollen für den vRealize Orchestrator Client auf der Seite **Identitäts- und Zugriffsverwaltung** in vRealize Automation zuweisen. Dienstrollen können für die eingebetteten vRealize Orchestrator Client- und eigenständigen vRealize Orchestrator-Instanzen zugewiesen werden, die mit vRealize Automation authentifiziert wurden.



vRealize Orchestrator-Dienstrollen verwalten, auf welche Funktionen die eingebetteten vRealize Orchestrator Client-Benutzer zugreifen können. Weitere Informationen zu vRealize Orchestrator-Rollen finden Sie unter [Verwalten von Rollen und Gruppen im vRealize Orchestrator Client](#).

---

**Hinweis** Eigenständige vRealize Orchestrator-Instanzen, die mit vSphere authentifiziert wurden und eine vRealize Automation-Lizenz verwenden, können Rollen direkt im vRealize Orchestrator Client zuweisen. Weitere Informationen finden Sie unter [Zuweisen von Rollen im vRealize Orchestrator Client](#).

---

#### Voraussetzungen

- Stellen Sie sicher, dass die entsprechenden Benutzer und Gruppen aus einer gültigen vIDM-Instanz importiert werden.
- Stellen Sie vor dem Zuweisen einer vRealize Orchestrator-Dienstrolle zum Benutzer sicher, dass dem Benutzer in vRealize Automation eine Organisationsrolle zugeteilt wurde. Weitere Informationen finden Sie unter *Verwalten von Benutzern und Gruppen in vRealize Automation* in *Verwalten von vRealize Automation*.

#### Verfahren

- 1 Wählen Sie im Dropdown-Menü oben rechts die Option **Identitäts- und Zugriffsverwaltung** aus.
- 2 Suchen Sie auf der Registerkarte **Aktive Benutzer** nach der E-Mail-Adresse des Benutzers, den Sie vRealize Orchestrator zuweisen möchten.
- 3 Aktivieren Sie das Kontrollkästchen neben dem Benutzer und klicken Sie auf **Rollen bearbeiten**.
- 4 Klicken Sie auf **Dienstzugriff hinzufügen**.
- 5 Wählen Sie im linken Dropdown-Menü **Orchestrator** aus.
- 6 Wählen Sie im rechten Dropdown-Menü die Rolle aus, die Sie dem Benutzer zuweisen möchten.
- 7 Klicken Sie auf **Speichern**.

## Versionsverlauf von vRealize Orchestrator-Objekten

Der vRealize Orchestrator Client behält einen Versionsverlaufsdatensatz für jedes vRealize Orchestrator-Objekt bei. Mithilfe des Versionsverlaufs können Sie verschiedene vRealize Orchestrator-Objektversionen vergleichen und eine vorherige Version wiederherstellen.

vRealize Orchestrator erstellt einen Versionsverlaufsdatensatz für jedes vRealize Orchestrator-Objekt, wenn Sie das Objekt speichern. Bei nachfolgenden Änderungen am vRealize Orchestrator-Objekt wird ein neuer Versionsverlaufsdatensatz erstellt. Die Verlaufsdatensätze der vorherigen Versionen werden beibehalten und können verwendet werden, um Änderungen am Objekt nachzuverfolgen und das Objekt auf eine vorherige Version zurückzusetzen. Wenn Sie ein Objekt auf eine vorherige Version zurücksetzen, wird ein neuer Versionsverlaufsdatensatz erstellt.

Der vRealize Orchestrator Client verfolgt den Versionsverlauf der folgenden vRealize Orchestrator-Objekte:

- Workflows
- Aktionen
- Pakete
- Richtlinien
- Konfigurationselemente

Auf den Versionsverlauf eines Objekts können Sie über die Registerkarte **Versionsverlauf** der Objekteditorseite zugreifen. Wenn Sie ein Objekt gleichzeitig mit einem anderen Benutzer bearbeiten, kann es zu einem Zusammenführungskonflikt kommen. Um den Zusammenführungskonflikt zu lösen, klicken Sie auf **Auflösen** rechts neben der Fehlermeldung. Im Fenster **Konflikte lösen** haben Sie drei Optionen:

- **Ihre verwenden.** Lösen Sie den Zusammenführungskonflikt, indem Sie die vom anderen Benutzer vorgenommenen Änderungen verwenden.
- **Unsere verwenden.** Lösen Sie den Zusammenführungskonflikt, indem Sie Ihre Änderungen verwenden.
- **Auflösen.** Lösen Sie den Zusammenführungskonflikt, indem Sie das angezeigte Änderungsmodell bearbeiten. Wenn das bereitgestellte Modell ungültig ist, ist diese Option nicht verfügbar.

## Wiederherstellen eines Workflows auf eine frühere Version

Sie können einen Workflow auf eine zuvor gespeicherte Version wiederherstellen.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Workflows** und wählen Sie einen Workflow aus.
- 3 Wählen Sie die Registerkarte **Versionsverlauf** aus.
- 4 Um einen Vergleich zwischen den Versionen anzuzeigen, wählen Sie eine Workflow-Version aus und wählen Sie eine andere Version aus dem Dropdown-Menü **Unterschied zu** aus.

Es wird ein Fenster eingeblendet, in dem die Unterschiede zwischen der aktuellen und der ausgewählten Workflowversion angezeigt werden.

- 5 Um den Workflow auf eine andere Version wiederherzustellen, klicken Sie auf **Wiederherstellen**.

Der Workflowzustand wird auf den Zustand der ausgewählten Version zurückgesetzt.

---

**Hinweis** Sie können auch eine Workflow-Version aus der Ansicht des Tools für grafische Unterschiede wiederherstellen. Weitere Informationen finden Sie unter [Visueller Vergleich zwischen Workflow-Versionen](#).

---

## Visueller Vergleich zwischen Workflow-Versionen

Vergleichen Sie die Änderungen zwischen Workflow-Versionen mit dem Tool für grafische Unterschiede.

Standardmäßig zeigt der vRealize Orchestrator-Versionsverlauf Unterschiede zwischen Workflow-Versionen in einem YAML-Formular. Sie können auch einen visuellen Vergleich zwischen verschiedenen Workflow-Versionen durchführen. Sie können Änderungen an folgenden Elementen anzeigen:

- Den allgemeinen Workflow-Informationen, z. B. Versionsnummer und Workflow-Beschreibung.
- Den im Workflow verwendeten Variablen.
- Den Ein- und Ausgabeparametern des Workflows.
- Dem Workflow-Schema.

### Voraussetzungen

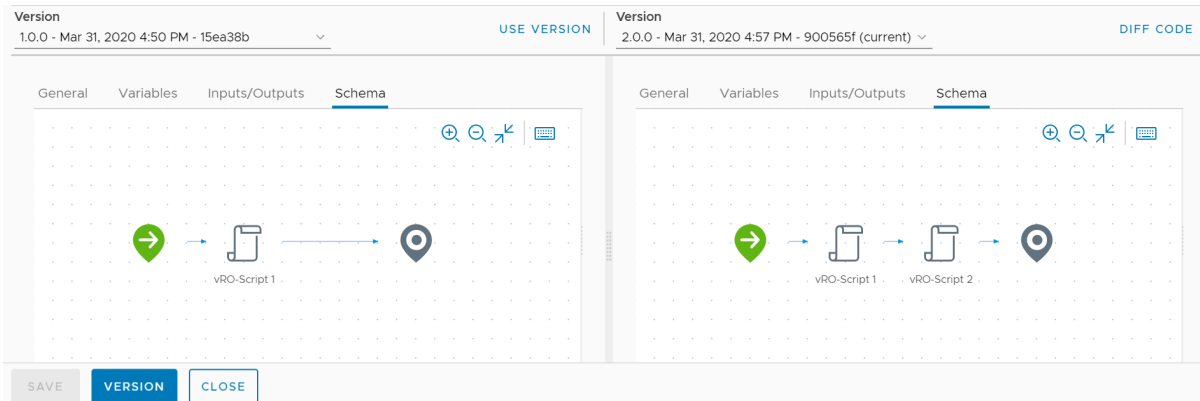
Erstellen Sie einen Workflow.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Workflows** und wählen Sie einen Ihrer Workflows aus.
- 3 Bearbeiten Sie den Inhalt des Workflows.  
Sie können z. B. eine zusätzliche **Skriptfähige Aufgabe** auf der Registerkarte **Schema** hinzufügen.
- 4 Klicken Sie auf **Speichern**.
- 5 Wählen Sie die Registerkarte **Versionsverlauf** aus.

## 6 Wählen Sie oben rechts **Unterschied visuell** aus.

Sie können nun einen visuellen Vergleich zwischen zwei ausgewählten Workflow-Versionen durchführen. Sie können im Dropdown-Menü **Version** auswählen, welche Versionen verglichen werden sollen.



## 7 (Optional) Sie können einen Workflow auf eine andere Version wiederherstellen, indem Sie **Version verwenden** auswählen.

## Zurücksetzen der vRealize Orchestrator-Inhaltsbestandsliste auf einen vorherigen Status mit Git

Wenn Sie einen früheren Git-Commit verwenden, können Sie Ihren vRealize Orchestrator-Inhalt auf einen früheren Status zurücksetzen.

Sie können Ihren vRealize Orchestrator-Inhalt auf einen vorherigen Status zurücksetzen, indem Sie einen bestimmten Commit auswählen.

### Voraussetzungen

- Konfigurieren Sie eine Verbindung zu einem GitHub- oder GitLab-Repository. Siehe [Konfigurieren einer Verbindung mit einem Git-Repository](#).
- Geben Sie einen lokalen Änderungssatz an das konfigurierte Git-Repository weiter.

### Verfahren

- 1 Melden Sie sich bei vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Verwaltung > Git-Verlauf**.
- 3 Wählen Sie einen Änderungssatz aus, den Sie zurücksetzen möchten, und klicken Sie auf **Auf diesen zurücksetzen**.
- 4 Bestätigen Sie, dass Sie auf diesen bestimmten Commit zurücksetzen möchten, und klicken Sie auf **OK**.

Die Inhaltsbestandsliste von vRealize Orchestrator wird auf den Status zurückgesetzt, der im Commit angegeben wurde. Der relevante vRealize Orchestrator-Inhalt wird auf eine vorherige Version zurückgesetzt. Wenn der Inhalt bei der Übertragung des Commits nicht vorhanden war, wird er aus der Bestandsliste entfernt.

## Nächste Schritte

Um die Bestandsliste von vRealize Orchestrator bis zum letzten im Git-Repository gespeicherten Status wiederherzustellen, führen Sie über das Fenster **Git-Verlauf** einen Pull-Befehl aus.

# vRealize Orchestrator- Anwendungsbeispiele

# 4

Diese Anwendungsbeispiele zeigen einen Teil der Funktionalität der vRealize Orchestrator-Plattform.

Diese Anwendungsbeispiele stellen nur Beispielwerte dar. Ihre Umgebungsstruktur und Benennungskonventionen können davon abweichen.

Dieses Kapitel enthält die folgenden Themen:

- [Vorgehensweise zum Integrieren von Amazon Web Services in vRealize Orchestrator mithilfe von Python](#)
- [Vorgehensweise zum Verwenden von Git-Zweigen zum Verwalten meiner vRealize Orchestrator-Objektbestandsliste](#)
- [Vorgehensweise zur Verwendung von Drittanbietermodulen zum Aufrufen der vRealize Automation-Projekt-API](#)

## Vorgehensweise zum Integrieren von Amazon Web Services in vRealize Orchestrator mithilfe von Python

Dieses vRealize Orchestrator-Anwendungsbeispiel zeigt , wie Sie Python verwenden können, um die Funktionen Ihrer vRealize Orchestrator-Bereitstellung zu erweitern.

Ab vRealize Orchestrator 8.1 können Sie drei neue Laufzeiten für die Verwendung in Ihren Aktionen und Workflow-Skripts verwenden.

- Python 3.7
- Node.js 12
- PowerCLI 11/PowerShell 6.2

- PowerCLI 12/PowerShell 7

---

**Hinweis** Die PowerCLI-Laufzeit umfasst PowerShell und die folgenden Module: VMware.PowerCLI, PowerNSX, PowervRA.

---

**Wichtig** Sie können die neuen Laufzeiten nur dann verwenden, wenn Ihre vRealize Orchestrator-Bereitstellung eine vRealize Automation-Lizenz verwendet.

---

Dieses Anwendungsbeispiel zeigt, wie Sie ein Python-Skript erstellen können, das EC2-Instanzen in Amazon Web Services (AWS) aufruft.

---

**Wichtig** Machen Sie sich mit den Kernkonzepten zur Verwendung von Python-, Node.js- und PowerShell-Skripts in vRealize Orchestrator vertraut, bevor Sie mit der Entwicklung Ihres benutzerdefinierten Skripts beginnen. Weitere Informationen finden Sie unter [Kernkonzepte für Python-, Node.js- und PowerShell-Skripts](#).

---

## Verfahren

### 1 Erstellen eines anfänglichen Python-Skripts

Erstellen Sie auf Ihrer lokalen Maschine Ihr Python-Skript und packen Sie das Skript und eine boto3-Bibliothek in einen ZIP-Ordner.

### 2 Erstellen der Amazon Web Services-Aktion

Erstellen Sie eine vRealize Orchestrator-Aktion, die das Python-Skript verwendet.

### 3 Debuggen der Amazon Web Services-Aktion

Die ursprüngliche Version des Python-Skripts weist einen absichtlich integrierten Fehler auf, sodass Sie lernen können, wie Sie Ihr Skript debuggen.

### 4 Aktualisieren der Amazon Web Services-Aktion

Löschen Sie das aktualisierte Python-Skript und führen Sie die Aktion erneut aus.

## Erstellen eines anfänglichen Python-Skripts

Erstellen Sie auf Ihrer lokalen Maschine Ihr Python-Skript und packen Sie das Skript und eine boto3-Bibliothek in einen ZIP-Ordner.

### Voraussetzungen

- Laden Sie Python 3 herunter und installieren Sie es. Weitere Informationen finden Sie auf der [Python-Downloadseite](#).
- Laden Sie Visual Studio Code herunter und installieren Sie das Programm. Weitere Informationen finden Sie auf der [Downloadseite für Visual Studio Code](#).
- Vergewissern Sie sich, dass Sie die Python-Erweiterung für Visual Studio Code installiert haben. Weitere Informationen finden Sie im [Visual Studio Marketplace](#).

## Verfahren

- 1 Erstellen Sie auf Ihrem lokalen Rechner einen Ordner mit dem Namen vro-python-aws und installieren Sie dort das boto3-Python-SDK.

```
mkdir vro-python-aws
cd vro-python-aws
mkdir lib
pip install boto3 -t lib/
```

- 2 Öffnen Sie einen Editor und erstellen Sie das Python-Hauptskript. Für dieses Anwendungsbeispiel verwenden Sie Visual Studio Code.

```
import boto3

def handler(context, inputs):
    ec2 = boto3.resource('ec2')
    filters = [{
        'Name': 'instance-state-name',
        'Values': ['running']
    }]

    instances = ec2.instances.filter(Filters=filters)
    for instance in instances:
        print('Instance: ' + instance.id)
```

Dieses Python-Skript listet alle ausgeführten EC2-Instanzen in einer bestimmten Region auf.

- 3 Speichern Sie das erstellte Skript als Datei main.py im Ordner vro-python-aws.
- 4 Melden Sie sich bei einer Befehlszeilenschnittstelle an.
- 5 Navigieren Sie zu dem Ordner vro-python-aws.

```
cd vro-python-aws
```

- 6 Erstellen Sie ein ZIP-Paket, das das Python-Skript enthält.

```
zip -r --exclude=*.zip -X vro-python-aws.zip .
```

---

**Hinweis** Sie können das ZIP-Paket auch mithilfe eines ZIP-Tools wie 7-Zip erstellen.

---

## Ergebnisse

Sie haben das grundlegende Python-Skript erstellt und es für den Import in Ihre vRealize Orchestrator-Bereitstellung vorbereitet.

## Erstellen der Amazon Web Services-Aktion

Erstellen Sie eine vRealize Orchestrator-Aktion, die das Python-Skript verwendet.



## Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Aktionen**.
- 3 Klicken Sie auf **Neue Aktion**.
- 4 Geben Sie auf der Registerkarte **Allgemein** einen Namen, ein Modul und eine Versionsnummer für die Aktion ein.
- 5 Wählen Sie auf der Registerkarte **Skript** die Option **Python 3.7** als Laufzeit und **ZIP** als Skripttyp aus.
- 6 Klicken Sie auf **Importieren**.
- 7 Navigieren Sie zum Ordner vro-python-aws und wählen Sie das ZIP-Paket aus, das das Python-Skript enthält.
- 8 Geben Sie im Textfeld **Eingabe-Handler** den Text **main.handler** ein.

---

**Hinweis** Der Eingabe-Handler der Aktion basiert auf dem Hauptskript im importierten ZIP-Paket. Da sich das Hauptskript in einer Datei mit dem Namen `main.py` und einer Funktion mit dem Namen **handler** befindet, muss der Eingabe-Handler **main.handler** sein. Wenn Sie Ihre Hauptskriptdatei anders genannt haben, ändern Sie den Wert für den Eingabe-Handler entsprechend.

---

- 9 Speichern Sie die Aktion und klicken Sie auf **Ausführen**.

Bei der Aktionsausführung tritt ein Fehler auf.

- 10 Wählen Sie die Registerkarte **Protokolle** aus.

In den Protokollen der Aktionsausführung wird eine Fehlermeldung des Typs „`botocore.exceptions.NoRegionError: Sie müssen eine Region angeben.`“ angezeigt. Es handelt sich hierbei um ein erwartetes Verhalten, da das anfängliche Python-Skript keine Region definiert.

## Nächste Schritte

Debuggen Sie das Python-Skript. Weitere Informationen finden Sie unter [Debuggen der Amazon Web Services-Aktion](#).

## Debuggen der Amazon Web Services-Aktion

Die ursprüngliche Version des Python-Skripts weist einen absichtlich integrierten Fehler auf, sodass Sie lernen können, wie Sie Ihr Skript debuggen.

## Voraussetzungen

Melden Sie sich bei Ihrem Amazon Web Services-Konto (AWS) an und erstellen Sie einen IAM-Benutzer speziell für dieses Anwendungsbeispiel-Szenario. Weitere Informationen hierzu finden Sie unter [Erstellen eines IAM-Benutzers in Ihrem AWS-Konto](#). Der IAM-Benutzer muss über die folgenden Berechtigungen verfügen:

```
"Effect": "Allow",
"Action": "ec2:DescribeInstances",
"Resource": "*"

```

## Verfahren

- 1 Bereiten Sie die vRealize Orchestrator Appliance vor.

---

**Vorsicht** Debuggen Sie keine Skripts in Ihrer vRealize Orchestrator-Produktionsbereitstellung. Debuggen Sie in einer vRealize Orchestrator-Bereitstellung mit einem einzelnen Knoten, die Sie für Entwicklungs- und Testzwecke verwenden.

---

- a Melden Sie sich über SSH als **root**-Benutzer bei der Befehlszeile der vRealize Orchestrator Appliance an.
- b Führen Sie den Befehl `vracli dev tools` aus.
- c Sie werden aufgefordert zu bestätigen, dass Sie fortfahren möchten. Geben Sie **ja** ein, um fortzufahren, oder **nein**, um den Vorgang abubrechen.

---

**Wichtig** Indem Sie den Befehl `vracli dev tools` ausführen, öffnen Sie die Ports, die zum Debuggen des Python-Skripts erforderlich sind. Sie müssen die aktuelle SSH-Sitzung während des Debug-Prozesses geöffnet lassen.

---

- 2 Starten Sie die Debug-Konfiguration.

- a Melden Sie sich beim vRealize Orchestrator Client an.
- b Öffnen Sie die AWS-Aktion und klicken Sie auf **Debuggen**.  
Der Debug-Prozess beginnt, und die Aktionsausführung wird angehalten.
- c Wählen Sie die Registerkarte **Debug-Konfiguration** aus.  
Die Registerkarte enthält eine Konfiguration vom Typ `.json`, die Sie zum Debuggen des Python-Skripts remote an Ihre IDE anhängen können.
- d Kopieren Sie den Konfigurationsinhalt manuell oder klicken Sie auf **In Zwischenablage kopieren**.

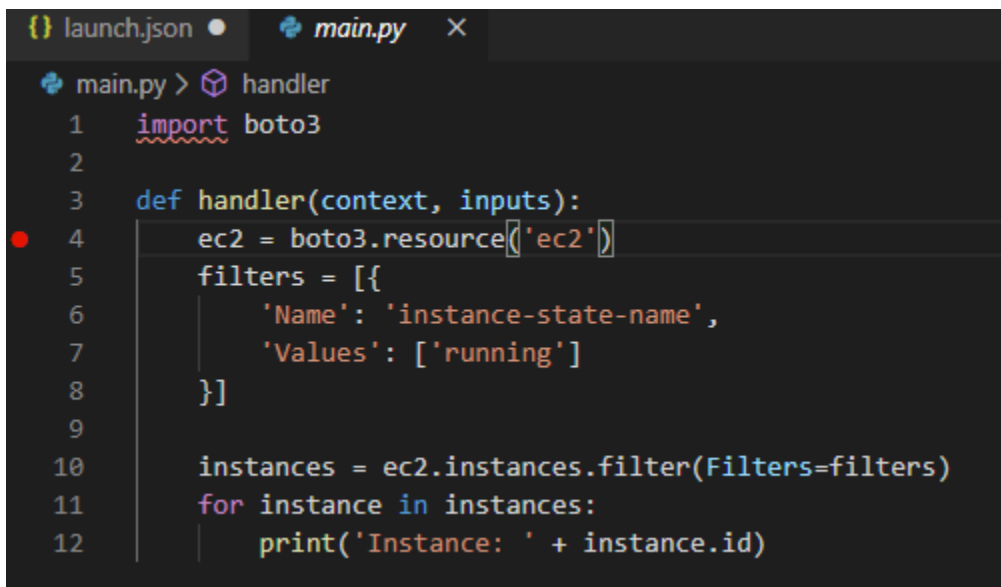
- 3 Debuggen Sie das Python-Skript.

- a Öffnen Sie Visual Studio Code.
- b Öffnen Sie den Ordner `vro-python-aws`.

- c Wählen Sie im oberen Navigationsbereich **Ausführen > Konfigurationen öffnen**.
- d Wählen Sie **Python-Datei** aus.
- e Behalten Sie die Attribute "version" und "configuration" an ihren aktuellen Positionen bei und fügen Sie den Inhalt der .json-Konfiguration ein, die aus dem vRealize Orchestrator Client kopiert wurde. Die generierte launch.json-Datei muss dem folgenden Beispiel gleichen:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "request": "attach",
      "port": 18281,
      "name": "vRO Python debug 8302f4c7-5beb-40da-848a-5003c0296f7b",
      "host": "es-sof-vc-vm-225-190.sof-mbu.eng.vmware.com",
      "type": "python",
      "pathMappings": [
        {
          "localRoot": "${workspaceFolder}",
          "remoteRoot": "/var/run/vco-polyglot/function"
        }
      ]
    }
  ]
}
```

- f Wählen Sie die Skriptdatei main.py aus und fügen Sie der Zeile `ec2 = boto3.resource('ec2')` einen Haltepunkt hinzu.



```
launch.json main.py
main.py > handler
1 import boto3
2
3 def handler(context, inputs):
4     ec2 = boto3.resource('ec2')
5     filters = [{
6         'Name': 'instance-state-name',
7         'Values': ['running']
8     }]
9
10    instances = ec2.instances.filter(Filters=filters)
11    for instance in instances:
12        print('Instance: ' + instance.id)
```

- g Wählen Sie im oberen Navigationsbereich **Ausführen > Debuggen starten** aus.

- h Wenn der Debugger den Haltepunkt erreicht, führen Sie einen Überspringen-Vorgang durch.

Die Debug-Ausführung gibt an, dass dem Python-Skript eine angegebene Region und ein AWS-Zugriffsschlüssel fehlt.

- i Gehen Sie zurück zur geöffneten vRealize Orchestrator Appliance-Sitzung und drücken Sie die **Eingabetaste**, um die Ports zu schließen, die Sie für diese Debug-Sitzung geöffnet haben.

#### 4 Fügen Sie dem Python-Skript die fehlenden Informationen hinzu.

- a Erstellen Sie in Visual Studio Code eine Datei mit dem Namen `awsconfig`, die den AWS-Zugriffsschlüssel des IAM-Benutzers und die AWS-Region enthält, die Sie mit dem Python-Skript anpingen möchten.

```
[default]
aws_access_key_id=your key ID
aws_secret_access_key=your secret access key
region=your-region
```

- b Speichern Sie `awsconfig` als Konfigurationsdatei (`.cfg`) im Ordner `vro-python-aws`.
- c Öffnen Sie die Datei `main.py` und bearbeiten Sie sie, damit die boto3-Bibliothek die Datei `awsconfig.cfg` verwenden kann.

```
import boto3

import os
os.environ['AWS_CONFIG_FILE'] = os.getcwd() + '/awsconfig.cfg'

def handler(context, inputs):
    ec2 = boto3.resource('ec2')
    filters = [{
        'Name': 'instance-state-name',
        'Values': ['running']
    }]

    instances = ec2.instances.filter(Filters=filters)
    for instance in instances:
        print('Instance: ' + instance.id)
```

- d Erstellen Sie ein neues ZIP-Paket, das die Datei `main.py`, die Datei `awsconfig.cfg` und die boto3-Bibliothek enthält.

```
zip -r --exclude=*.zip -X vro-python-aws.zip .
```

---

**Hinweis** Sie können das ZIP-Paket auch mithilfe eines ZIP-Tools wie 7-Zip erstellen.

---

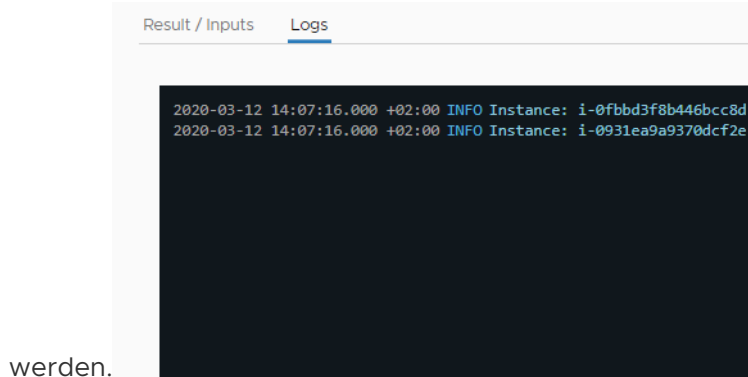
## Aktualisieren der Amazon Web Services-Aktion

Löschen Sie das aktualisierte Python-Skript und führen Sie die Aktion erneut aus.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Aktionen** und wählen Sie die Originalaktion für Amazon Web Services (AWS) aus.
- 3 (Optional) Ändern Sie auf der Registerkarte **Allgemein** die Versionsnummer.
- 4 Entfernen Sie das alte ZIP-Paket und klicken Sie auf **Importieren**.
- 5 Wählen Sie das aktualisierte ZIP-Paket aus.
- 6 Speichern Sie die Aktion und klicken Sie auf **Ausführen**.
- 7 Nach Abschluss der Aktionsausführung wählen Sie die Registerkarte **Protokolle** aus.

In den Protokollen werden die EC2-Instanzen angezeigt, die von der Aktion abgefragt



### Nächste Schritte

Erstellen Sie einen vRealize Orchestrator-Workflow, der die aktualisierte AWS-Aktion als **Aktionselement** verwendet.

## Vorgehensweise zum Verwenden von Git-Zweigen zum Verwalten meiner vRealize Orchestrator-Objektbestandsliste

Verwenden Sie Zweige, um zu organisieren, wie vRealize Orchestrator-Inhalt in Ihrem Git-Repository verwaltet wird.

Durch die Verwendung von Git können Sie die Flexibilität für Ihre vRealize Orchestrator-Entwickler erhöhen, indem Sie ein zentrales Repository bereitstellen. Beispielsweise können Sie Git verwenden, um die Workflow-Entwicklung über mehrere vRealize Orchestrator-Umgebungen hinweg zu verwalten.

---

**Hinweis** Um Git zum Verwalten Ihrer Objektbestandsliste zu verwenden, muss Ihre vRealize Orchestrator-Bereitstellung eine vRealize Automation-Lizenz verwenden. Weitere Informationen finden Sie im Abschnitt zur *vRealize Orchestrator-Funktionsaktivierung mit Lizenzen* in *Installieren und Konfigurieren von vRealize Orchestrator*.

---

Ab vRealize Orchestrator 8.1 können Sie Objekte an Zweige weitergeben und daraus abrufen. Sie können Zweige verwenden, um die Entwicklung bestimmter vRealize Orchestrator-Objekte zu verwalten, bevor diese wieder mit Ihrem Hauptzweig zusammengeführt werden.

In diesem Anwendungsbeispiel verwenden Sie ein GitLab-Projekt, um vRealize Orchestrator-Objekte zu verwalten, die die Python-Laufzeit verwenden. Dieses Anwendungsbeispiel ist ein Beispiel für die Git-Funktionen in vRealize Orchestrator und stellt nicht den ganzen Funktionsumfang dar.

---

**Hinweis** Wenn Sie mit GitHub vertraut sind, können Sie ein GitHub-Repository für dieses Anwendungsbeispiel verwenden.

---

## Verfahren

### 1 Vorbereiten der GitLab-Umgebung

Erstellen Sie einen Git-Zweig für Ihre Python-Objekte in vRealize Orchestrator.

### 2 Konfigurieren einer Verbindung mit einem Git-Repository

Als **Administrator** können Sie eine Verbindung zwischen Ihrer vRealize Orchestrator-Bereitstellung und einem Git-Repository oder -Projekt konfigurieren.

### 3 Weitergeben von Änderungen an ein Git-Repository

Geben Sie Ihre Änderungen an lokalen vRealize Orchestrator-Objekten an Ihr integriertes Git-Repository weiter. Für dieses Anwendungsbeispiel werden Änderungen an eine vRealize Orchestrator-Aktion auf einen bestimmten Git-Zweig weitergegeben.

## Vorbereiten der GitLab-Umgebung

Erstellen Sie einen Git-Zweig für Ihre Python-Objekte in vRealize Orchestrator.

### Voraussetzungen

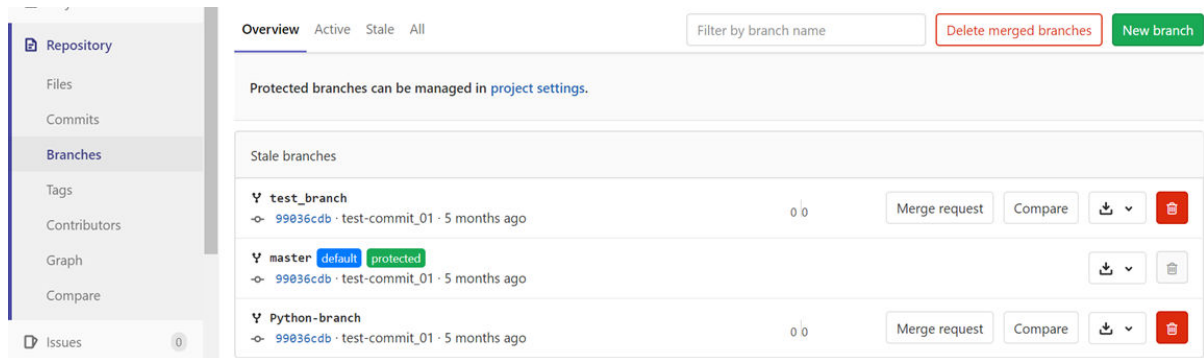
Erstellen Sie ein GitLab-Projekt für Ihre vRealize Orchestrator-Umgebung. Weitere Informationen finden Sie unter [Erstellen eines Projekts](#).

### Verfahren

- 1 Melden Sie sich bei Ihrem GitLab-Konto an.
- 2 Navigieren Sie zu Ihrem GitLab-Projekt.

- 3 Wählen Sie im linken Navigationsbereich **Repository > Zweige**.
- 4 Klicken Sie auf der Registerkarte **Übersicht** auf **Neuer Zweig**.
- 5 Geben Sie unter **Zweigname** den Namen **Python-branch** ein.
- 6 Lassen Sie die Option **Erstellen von** als **master** festgelegt.
- 7 Klicken Sie auf **Zweig erstellen**.

Sie haben einen Zweig für Ihre Python-basierten vRealize Orchestrator-Objekte erstellt.



## Konfigurieren einer Verbindung mit einem Git-Repository

Als **Administrator** können Sie eine Verbindung zwischen Ihrer vRealize Orchestrator-Bereitstellung und einem Git-Repository oder -Projekt konfigurieren.

Um Git für die Verwaltung Ihrer vRealize Orchestrator-Objektbestandsliste zu verwenden, müssen Sie zuerst mithilfe des vRealize Orchestrator Client eine Verbindung zu Ihrem Git-Repository konfigurieren.

### Voraussetzungen

- Vergewissern Sie sich, dass Ihre vRealize Orchestrator-Umgebung eine vRealize Automation-Lizenz verwendet.
- Generieren Sie ein Zugriffstoken für Ihr GitLab-Projekt und kopieren Sie es für die Verwendung während des Konfigurationsvorgangs in Ihre Zwischenablage. Weitere Informationen finden Sie unter [Erstellen eines privaten Zugriffstokens](#).

**Hinweis** Für dieses Anwendungsbeispiel verwenden Sie ein GitLab-Projekt. Wenn Sie mit GitHub vertraut sind, können Sie ein GitHub-Repository verwenden. Informationen zum Generieren eines GitHub-Tokens finden Sie unter [Erstellen eines privaten Zugriffstokens für die Befehlszeile](#).

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client als **Administrator** an.
- 2 Navigieren Sie zu **Verwaltung > Git-Repositories**.
- 3 Klicken Sie auf **Repository hinzufügen**.

- 4 Geben Sie die URL-Adresse Ihres Git-Repositorys ein.

Beispiel: <https://gitlab.com/myusername/my-vro-repo>.

---

**Hinweis** Sie können auch eine Verbindung mit dem SSH-Protokoll herstellen.

---

- 5 Geben Sie den Benutzernamen Ihres Git-Profiles ein.
- 6 Geben Sie das Zugriffstoken Ihres Git-Repositorys ein.
- 7 Um die Verbindung zum Git-Repository zu validieren, klicken Sie auf **Validieren**.
- 8 (Optional) Ändern Sie den Namen, der zur Bezeichnung des Repositorys im vRealize Orchestrator Client verwendet wird.
- 9 (Optional) Fügen Sie eine kurze Beschreibung für das verbundene Git-Repository hinzu.
- 10 Zum Aktivieren des verbundenen Git-Repositorys klicken Sie auf **Repository aktivieren**

---

**Hinweis** Es kann immer nur jeweils ein Git-Repository aktiv sein. Sie können das aktive Git-Repository auf der Seite **Git-Repositorys** ändern.

---

- 11 Wählen Sie den Zweig aus, an den die Änderungen weitergegeben werden sollen. Für dieses Anwendungsbeispiel verwenden Sie **Python-branch**. Weitere Informationen finden Sie unter [Vorbereiten der GitLab-Umgebung](#).

---

**Hinweis** Sie können den ausgewählten Git-Zweig jederzeit ändern, nachdem Sie die anfängliche Git-Konfiguration abgeschlossen haben.

---

- 12 Um den Konfigurationsvorgang abzuschließen, klicken Sie auf **Speichern**.

#### Nächste Schritte

Navigieren Sie zurück zum Menü **Git-Repositorys** und bestätigen Sie, dass der Status des Repositorys **Aktiv** ist.

## Weitergeben von Änderungen an ein Git-Repository

Geben Sie Ihre Änderungen an lokalen vRealize Orchestrator-Objekten an Ihr integriertes Git-Repository weiter. Für dieses Anwendungsbeispiel werden Änderungen an eine vRealize Orchestrator-Aktion auf einen bestimmten Git-Zweig weitergegeben.

Sie können einen lokalen Änderungssatz an ein Git-Repository weitergeben. Jeder Änderungssatz kann aus einem oder mehreren geänderten vRealize Orchestrator-Objekten bestehen.

---

**Hinweis** Das Weitergeben und Verwerfen von Änderungssätzen in einem Git-Repository ist nicht durch Gruppenberechtigungen beschränkt. Deshalb kann ein Workflow-Entwickler aus einer Gruppe lokale Änderungen weitergeben oder verwerfen, die von einem anderen Entwickler vorgenommen wurden.

---



## Voraussetzungen

- Überprüfen Sie, ob Sie einen Git-Zweig erstellt haben. Weitere Informationen finden Sie unter [Vorbereiten der GitLab-Umgebung](#).
- Stellen Sie sicher, dass Sie eine Verbindung mit einem Git-Repository konfiguriert haben. Weitere Informationen finden Sie unter [Konfigurieren einer Verbindung mit einem Git-Repository](#).
- Stellen Sie sicher, dass Ihre Git-Integration so eingestellt ist, dass Änderungen an den Git-Zweig **Python-branch** weitergegeben werden.
- Erstellen Sie ein Python-basiertes vRealize Orchestrator-Objekt. Ein Beispiel finden Sie unter [Vorgehensweise zum Integrieren von Amazon Web Services in vRealize Orchestrator mithilfe von Python](#).

## Verfahren

- 1 Melden Sie sich bei vRealize Orchestrator Client an.
- 2 Bearbeiten Sie Ihre Python-Aktion.
  - a Navigieren Sie zu **Bibliothek > Aktionen** und wählen Sie Ihre Python-Aktion aus.
  - b Nehmen Sie einige geringfügige Änderungen an der Aktion vor, z. B. eine Änderung der Beschreibung.
  - c Speichern Sie die Aktion.

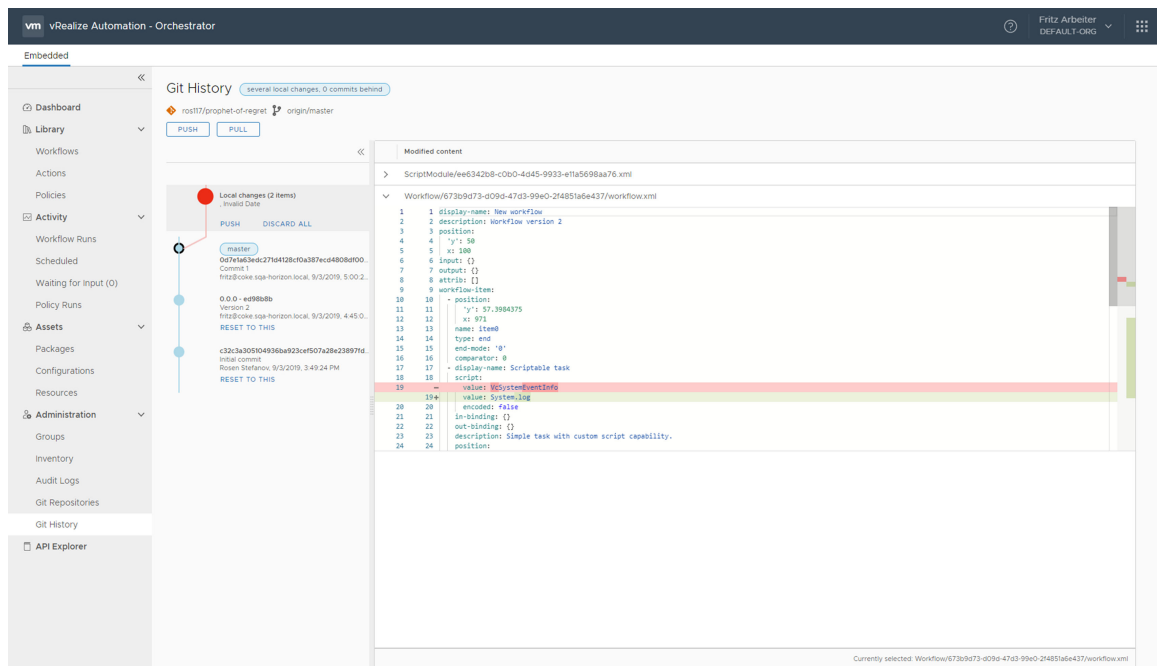
### 3 Geben Sie Ihre Änderungen an das Git-Repository weiter.

**Hinweis** Sie können auch lokale Änderungen für einzelne Objekte weitergeben, indem Sie auf die Option **Version** klicken, die am unteren Rand des Objekt-Editors angezeigt wird.

#### a Navigieren Sie zu **Administration > Git-Verlauf**.

Unter **Git-Verlauf** werden die aktuellen Unterschiede zwischen dem Zweig der lokalen Version und dem Zweig des ausgewählten Git-Repositorys angezeigt. Sie können den Eintrag für ein beliebiges geändertes vRealize Orchestrator-Objekt erweitern, um die Versionsunterschiede anzuzeigen.

**Hinweis** Sie können einen lokalen Änderungssatz mit **Alle verwerfen** verwerfen.



#### b Klicken Sie auf **Weitergeben**.

#### c Geben Sie einen Commit-Titel ein.

#### d (Optional) Geben Sie eine kurze Beschreibung für den Commit ein.

#### e Wählen Sie die Änderungen an Ihrer Python-Aktion aus, die Sie an das Git-Repository weitergeben möchten.

### 4 Um die Weitergabe der lokalen Änderungen an das Git-Repository abzuschließen, klicken Sie auf **Weitergeben**.

#### Nächste Schritte

Nachdem Sie die Entwicklung in Ihrem Git-Zweig abgeschlossen haben, führen Sie ihn mit dem Hauptzweig zusammen. Weitere Informationen finden Sie unter [Erstellen einer Zusammenführungsanforderung](#).

## Vorgehensweise zur Verwendung von Drittanbietermodulen zum Aufrufen der vRealize Automation-Projekt-API

In diesem vRealize Orchestrator-Anwendungsfall wird dargestellt, wie die vRealize Automation-Projekt-API mithilfe von Drittanbietermodulen aufgerufen werden kann.

Ab vRealize Orchestrator 8.1 können Sie die folgenden Laufzeiten verwenden:

- Python 3.7
- Node.js 12
- PowerCLI 11/PowerShell 6.2
- PowerCLI 12/PowerShell 7

---

**Hinweis** Die PowerCLI-Laufzeit umfasst PowerShell und die folgenden Module: VMware.PowerCLI, PowerNSX, PowervRA.

---

In diesem Anwendungsfall erhalten Sie Informationen zum Erstellen von vRealize Orchestrator-Aktionen, die Drittanbieter-Abhängigkeitsmodule zum Herstellen einer Verbindung mit der vRealize Automation-Projekt-API verwenden.

---

**Wichtig** Machen Sie sich mit den Kernkonzepten zur Verwendung von Python-, Node.js- und PowerShell-Skripts in vRealize Orchestrator vertraut, bevor Sie mit der Entwicklung Ihres benutzerdefinierten Skripts beginnen. Weitere Informationen finden Sie unter [Kernkonzepte für Python-, Node.js- und PowerShell-Skripts](#).

---

### Erstellen eines Python-Skripts zum Aufrufen der vRealize Automation-Projekt-API

Erstellen Sie ein Beispielskript, das Python zum Aufrufen der vRealize Automation-Projekt-API verwendet.

#### Voraussetzungen

Stellen Sie sicher, dass Sie Python 3 sowie das Installationsprogramm für das PIP-Paket installiert haben. Weitere Informationen finden Sie auf der Seite [Python-Downloads](#) und im [Python-Paketindex](#).

#### Verfahren

- 1 Öffnen Sie auf Ihrer lokalen Maschine eine Befehlszeilen-Shell.
- 2 Erstellen Sie einen Ordner mit der Bezeichnung vro-python-vra.

```
mkdir vro-python-vra
```

- 3 Navigieren Sie zum Ordner vro-python-vra.

```
cd vro-python-vra
```

#### 4 Erstellen Sie ein Python-Skript mit dem Namen `handler.py`.

```
touch handler.py
```

Im Skript `handler.py` muss eine Funktion definiert werden, die zwei Argumente, den Kontext der vRealize Orchestrator-Workflow-Ausführung und die gebundenen vRealize Orchestrator-Eingaben akzeptiert.

```
def handler(context, inputs):
    print('Hello, your inputs were ' + inputs)
    return None
```

**Hinweis** Mithilfe von Standardprotokollierungsbibliotheken werden alle protokollierten Einträge in der Aktion, die das Skript verwendet, auch im Workflow-Protokoll angezeigt. Die Eingaben und die Rückgabe des Skripts müssen entsprechende im vRealize Orchestrator-Client konfigurierte Eingabeparameter und Rückgabetypen aufweisen. Beispielsweise muss die Eingabe `vRAUrl` im Skript einen entsprechenden Eingabeparameter mit der Bezeichnung `vRAUrl` im vRealize Orchestrator-Client aufweisen. Wenn vom Skript ein Zeichenfolgenwert zurückgegeben wird, muss der im vRealize Orchestrator-Client konfigurierte Rückgabetypp ebenfalls vom Typ „Zeichenfolge“ sein. Wenn von der Aktion ein komplexes Objekt zurückgegeben wird, können Sie den Rückgabetypp `Properties` oder `Composite Type` verwenden.

#### 5 Installieren Sie das Python-Anforderungsmodul.

**Wichtig** Abhängigkeitsmodule von Drittanbietern müssen in einem Ordner auf Root-Ebene im Ordner des Hauptskripts `vro-python-vra` installiert werden. In diesem Anwendungsfall erstellen Sie einen Ordner mit dem Namen `lib` für das Anforderungsmodul.

- a Erstellen Sie einen Ordner mit dem Namen `lib`.

```
mkdir lib
```

- b Installieren Sie das Anforderungsmodul.

```
pip3 install requests -t lib/
```

#### 6 Fügen Sie dem Skript `handler.py` das Anforderungsmodul hinzu.

```
import requests

def handler(context, inputs):
    print('Hello, your inputs were ' + inputs)
    return None
```

#### 7 Erstellen Sie eine GET-Anforderung für die vRealize Automation-Projekt-API.

```
token = ''
vRAUrl = ''
```

```
r = requests.get(vRAUrl + '/iaas/api/projects', headers={'Authorization': 'Bearer ' + token})

print('Got response ' + r.text)
```

**8** Definieren Sie die Werte token und vRAUrl .

- a Rufen Sie das Zugriffstoken mithilfe der vRealize Automation-Identitätsdienst-API ab. Weitere Informationen finden Sie unter [Abrufen des Zugriffstokens für die vRealize Automation-API](#)
- b Definieren Sie das Skript für den Wert vRAUrl so, dass es einen vRealize Orchestrator-Eingabeparameter mit demselben Namen verwendet.

```
vRAUrl = inputs["vRAUrl"]
```

- c Fügen Sie der Datei handler.py die neuen Werte hinzu.

```
import requests

def handler(context, inputs):
    token = 'ACCESS_TOKEN'
    vRAUrl = inputs["vRAUrl"]

    r = requests.get(vRAUrl + '/iaas/api/projects', headers={'Authorization': 'Bearer ' + token})

    print('Got response ' + r.text)

    return r.json()
```

**Hinweis** Da die Antwort von der vRealize Automation-Projekt-API in einem JSON-Format zurückgegeben wird, verwenden Sie einen Properties- oder Composite Type-Rückgabebetyp für Ihre vRealize Orchestrator-Aktion.

- 9** Erstellen Sie ein ZIP-Paket, das die Datei handler.py und den Ordner lib Ihres Anforderungsmoduls enthält.

```
zip -r --exclude=*.zip -X vro-python-vra.zip .
```

### Nächste Schritte

Importieren Sie das PowerShell-Skript in eine vRealize Orchestrator-Aktion. Weitere Informationen finden Sie unter [Erstellen von Aktionen im vRealize Orchestrator Client](#).

## Erstellen eines Node.js-Skripts zum Aufrufen der vRealize Automation-Projekt-API

Erstellen Sie ein Beispielskript, das Node.js zum Aufrufen der vRealize Automation-Projekt-API verwendet.

## Voraussetzungen

Laden Sie Node.js 12 herunter und installieren Sie es. Weitere Informationen finden Sie unter [Node.js-Downloads](#).

## Verfahren

- 1 Öffnen Sie auf Ihrer lokalen Maschine eine Befehlszeilen-Shell.
- 2 Erstellen Sie einen Ordner mit der Bezeichnung vro-node-vra.

```
mkdir vro-node-vra
```

- 3 Navigieren Sie zum Ordner vro-node-vra.

```
cd vro-node-vra
```

- 4 Erstellen Sie ein Node.js-Skript mit dem Namen handler.js.

```
touch handler.js
```

Im Skript handler.js muss eine Funktion definiert werden, die zwei Argumente, den Kontext der vRealize Orchestrator-Workflow-Ausführung und die gebundenen vRealize Orchestrator-Eingaben akzeptiert.

```
exports.handler = (context, inputs) => {
  console.log('Hello, your inputs were ' + inputs);
  return null;
}
```

---

**Hinweis** Mithilfe von Standardprotokollierungsbibliotheken werden alle protokollierten Einträge in der Aktion, die das Skript verwendet, auch im Workflow-Protokoll angezeigt. Die Eingaben und die Rückgabe des Skripts müssen entsprechende im vRealize Orchestrator-Client konfigurierte Eingabeparameter und Rückgabebetypen aufweisen. Beispielsweise muss die Eingabe vRAUrl im Skript einen entsprechenden Eingabeparameter mit der Bezeichnung vRAUrl im vRealize Orchestrator-Client aufweisen. Wenn vom Skript ein Zeichenfolgenwert zurückgegeben wird, muss der im vRealize Orchestrator-Client konfigurierte Rückgabebetyp ebenfalls vom Typ „Zeichenfolge“ sein. Wenn von der Aktion ein komplexes Objekt zurückgegeben wird, können Sie den Rückgabebetyp Properties oder Composite Type verwenden.

---

- 5 Installieren Sie das Node.js-Anforderungsmodul.

```
npm install request
```

---

**Wichtig** Abhängigkeitsmodule von Drittanbietern müssen im Ordner node\_modules auf Root-Ebene im Ordner des Hauptskripts vro-node-vra installiert werden. Dieser Ordner sollte weder verschoben noch umbenannt werden.

---

**6** Fügen Sie dem Skript `handler.js` das Anforderungsmodul hinzu.

```
const request = require('request');

exports.handler = (context, inputs) => {
  console.log('Hello, your inputs were ' + inputs);
  return null;
}
```

**7** Erstellen Sie eine GET-Anforderung für die vRealize Automation-Projekt-API.

```
const token = '';
const vRAUrl = '';
request.get(vRAUrl + '/iaas/api/projects', { 'auth': { 'bearer': token } }, function (error,
response, body) {
  console.log('Got response ' + body);
});
```

**8** Definieren Sie die Werte `token` und `vRAUrl`.

- a Rufen Sie das Zugriffstoken mithilfe der vRealize Automation-Identitätsdienst-API ab. Weitere Informationen finden Sie unter [Abrufen des Zugriffstokens für die vRealize Automation-API](#).
- b Definieren Sie das Skript für den Wert `vRAUrl` so, dass es einen vRealize Orchestrator-Eingabeparameter mit demselben Namen verwendet.

```
const vRAUrl = inputs.vRAUrl;
```

- c Fügen Sie der Datei `handler.js` die neuen Werte hinzu.

```
const request = require('request');
exports.handler = (context, inputs, callback) => {
  const vRAUrl = inputs.vRAUrl;
  const token = 'ACCESS_TOKEN';
  request.get(vRAUrl + '/iaas/api/projects', { 'auth': { 'bearer': token } }, function
(error, response, body) {
    console.log('Got response ' + body);
    callback(null, JSON.parse(body));
  });
}
```

**Hinweis** Da die Antwort von der vRealize Automation-Projekt-API in einem JSON-Format zurückgegeben wird, verwenden Sie einen Properties- oder Composite Type-Rückgabebetyp für Ihre vRealize Orchestrator-Aktion.

**9** Erstellen Sie ein ZIP-Paket, das die Datei `handler.js` und den Ordner `node_modules` Ihres Anforderungsmoduls enthält.

```
zip -r --exclude=*.zip -X vro-node-vra.zip .
```

## Nächste Schritte

Importieren Sie das Node.js-Skript in eine vRealize Orchestrator-Aktion. Weitere Informationen finden Sie unter [Erstellen von Aktionen im vRealize Orchestrator Client](#).

## Erstellen eines PowerShell-Skripts zum Aufrufen der vRealize Automation-Projekt-API

Erstellen Sie ein Beispielskript, das PowerShell zum Aufrufen der vRealize Automation-Projekt-API verwendet.

### Verfahren

- 1 Öffnen Sie auf Ihrer lokalen Maschine eine Befehlszeilen-Shell.
- 2 Erstellen Sie einen Ordner mit der Bezeichnung vro-powershell-vra.

```
mkdir vro-powershell-vra
```

- 3 Navigieren Sie zum Ordner vro-powershell-vra.

```
cd vro-powershell-vra
```

- 4 Erstellen Sie ein PowerShell-Skript mit der Bezeichnung handler.ps1.

```
touch handler.ps1
```

Im Skript handler.ps1 muss eine Funktion definiert werden, die zwei Argumente, den Kontext der vRealize Orchestrator-Workflow-Ausführung und die gebundenen vRealize Orchestrator-Eingaben akzeptiert.

```
function Handler {
    Param($context, $inputs)

    $inputsString = $inputs | ConvertTo-Json -Compress
    Write-Host "Inputs were $inputsString"
}
```

**Hinweis** Mithilfe von Standardprotokollierungsbibliotheken werden alle protokollierten Einträge in der Aktion, die das Skript verwendet, auch im Workflow-Protokoll angezeigt. Die Eingaben und die Rückgabe des Skripts müssen entsprechende im vRealize Orchestrator-Client konfigurierte Eingabeparameter und Rückgabebetypen aufweisen. Beispielsweise muss die Eingabe vRAUrl im Skript einen entsprechenden Eingabeparameter mit der Bezeichnung vRAUrl im vRealize Orchestrator-Client aufweisen. Wenn vom Skript ein Zeichenfolgenwert zurückgegeben wird, muss der im vRealize Orchestrator-Client konfigurierte Rückgabebetyp ebenfalls vom Typ „Zeichenfolge“ sein. Wenn von der Aktion ein komplexes Objekt zurückgegeben wird, können Sie den Rückgabebetyp Properties oder Composite Type verwenden.



## 5 Installieren Sie das PowerShell-Assert-Modul.

**Wichtig** Abhängigkeitsmodule von Drittanbietern müssen in einem Ordner auf Root-Ebene im Ordner des Hauptskripts vro-powershell-vra installiert werden. In diesem Anwendungsfall erstellen Sie einen Ordner mit dem Namen `Module` für Ihr Assert-Modul.

- a Erstellen Sie einen Ordner mit dem Namen `Module`.

```
mkdir Modules
```

- b Installieren Sie das Assert-Modul.

```
powershell -c "Save-Module -Name Assert -Path ./Modules/ -Repository PSGallery"
```

## 6 Fügen Sie dem Skript `handler.ps1` das Assert-Modul hinzu.

```
Import-Module Assert

function Handler {
    Param($context, $inputs)

    $inputsString = $inputs | ConvertTo-Json -Compress
    Write-Host "Inputs were $inputsString"
}
```

## 7 Erstellen Sie eine GET-Anforderung für die vRealize Automation-Projekt-API, die das cmdlet `Invoke-RestMethod` verwendet.

```
$token = ''
$vraUrl = ''
$projectsUrl = $vraUrl + "/project-service/api/projects"
$response = Invoke-RestMethod $projectsUrl + '/iaas/api/projects' -Headers @{ 'Authorization' = "Bearer $token" } -Method 'GET'

Write-Host "Got response: $response"
```

## 8 Definieren Sie die Werte `token` und `vraUrl`.

- a Rufen Sie das Zugriffstoken mithilfe der vRealize Automation-Identitätsdienst-API ab. Weitere Informationen finden Sie unter [Abrufen des Zugriffstokens für die vRealize Automation-API](#).
- b Fügen Sie die Attribute `Assert-NotNull` und `Assert-Type` des Assert-Moduls hinzu.

```
$token | Assert-NotNull
$token | Assert-Type String
```

- c Definieren Sie das Skript für den Wert `vRAUrl` so, dass es einen vRealize Orchestrator-Eingabeparameter mit demselben Namen verwendet.

```
$vRAUrl = $inputs.vRAUrl
```

- d Fügen Sie der Datei `handler.ps1` die neuen Werte hinzu.

```
Import-Module Assert
$ErrorActionPreference = "Stop"
function Handler {
    Param($context, $inputs)
    $token = "ACCESS_TOKEN"
    $token | Assert-NotNull
    $token | Assert-Type String
    $vRAUrl = $inputs.vRAUrl
    $projectsUrl = $vRAUrl + "/project-service/api/projects"
    $response = Invoke-RestMethod $projectsUrl -Headers @{ 'Authorization' = "Bearer $token" } -
    Method 'GET'

    Write-Host "Got response: $response"

    return $response
}
```

**Hinweis** Da die Antwort von der vRealize Automation-Projekt-API in einem JSON-Format zurückgegeben wird, verwenden Sie einen Properties- oder Composite Type-Rückgabetyp für Ihre vRealize Orchestrator-Aktion.

- 9 Erstellen Sie ein ZIP-Paket, das die Datei `handler.ps1` und den Ordner `Module` Ihres Assert-Moduls enthält.

```
zip -r --exclude=*.zip -X vro-powershell-vra.zip .
```

### Nächste Schritte

Importieren Sie das PowerShell-Skript in eine vRealize Orchestrator-Aktion. Weitere Informationen finden Sie unter [Erstellen von Aktionen im vRealize Orchestrator Client](#).

# Verwalten von Workflows

# 5

Ein Workflow besteht aus einer Folge von Aktionen und Entscheidungen, die Sie nacheinander ausführen. vRealize Orchestrator stellt eine Bibliothek mit Workflows bereit, die gängige Verwaltungsaufgaben ausführen. vRealize Orchestrator stellt darüber hinaus Bibliotheken der einzelnen Aktionen bereit, die von den Workflows durchgeführt werden.

Workflows kombinieren Aktionen, Entscheidungen und Ergebnisse, die eine bestimmte Aufgabe oder einen bestimmten Prozess in einer virtuellen Umgebung ausführen, wenn sie in einer bestimmten Reihenfolge ablaufen. Workflows führen verschiedene Aufgaben aus, beispielsweise die Bereitstellung von virtuellen Maschinen, Datensicherung, periodische Wartungsaufgaben, Versand von E-Mails, SSH-Operationen, Verwaltung der physischen Infrastruktur und andere allgemeine Vorgänge im allgemeinen Betrieb. Workflows akzeptieren Eingangsdaten je nach ihrer Funktion. Sie können Workflows erstellen, die mit vordefinierten Zeitplänen ablaufen, oder die in Gang gesetzt werden, wenn bestimmte, erwartete Ereignisse eintreten. Die Informationen können von Ihnen oder anderen Benutzern, von einem anderen Workflow oder einer Aktion sowie von einem externen Prozess, beispielsweise einem Webdienstauftrag aus einer Anwendung geliefert werden. Workflows übernehmen die Validierung und Filterung von Informationen in einem bestimmten Ausmaß, bevor sie ausgeführt werden.

Workflows können andere Workflows aufrufen. Beispielsweise können Sie einen Workflow haben, der einen anderen Workflow aufruft, um eine neue virtuelle Maschine zu erstellen.

Sie erstellen Workflows über die integrierte Entwicklungsumgebung (IDE) der vRealize Orchestrator Client-Schnittstelle, die Zugriff auf die Workflow-Bibliothek und die Möglichkeit zum Ausführen von Workflows in der Workflow-Engine bietet. Die Workflow-Engine kann auch Objekte aus externen Bibliotheken verwenden, die Sie als Plug-In in vRealize Orchestrator integrieren. Durch diese Funktion können Sie Prozesse anpassen oder Funktionen implementieren, die von Drittanbieteranwendungen bereitgestellt werden.

Dieses Kapitel enthält die folgenden Themen:

- [Standard-Workflows in der vRealize Orchestrator-Workflow-Bibliothek](#)
- [Erstellen von Workflows im vRealize Orchestrator Client](#)
- [Bearbeiten von Workflows und Aktionen über den übergeordneten Workflow](#)
- [vRealize Orchestrator-Eingabeformular-Designer](#)

- [Anforderungen zur Benutzerinteraktion im vRealize Orchestrator Client](#)
- [Planen von Workflows im vRealize Orchestrator Client](#)

## Standard-Workflows in der vRealize Orchestrator-Workflow-Bibliothek

vRealize Orchestrator stellt eine Standardbibliothek mit Workflows bereit, die zum Automatisieren von Vorgängen in Ihrer virtuellen Infrastruktur genutzt werden können. Die Workflows in der Standardbibliothek sind schreibgeschützt. Um einen Standard-Workflow anzupassen, müssen Sie diesen Workflow duplizieren. Duplikatworkflows oder angepasste Workflows, die von Ihnen erstellt werden, können umfassend bearbeitet werden.

Auf die Inhalte der Workflow-Bibliothek kann über das Menü **Bibliothek > Workflows** des HTML5-basierten vRealize Orchestrator Client zugegriffen werden. Sowohl Standard- als auch benutzerdefinierte Workflows im Client werden mithilfe von Tags verwaltet. Sie können z. B. auf den Workflow **Schlüsselpaar generieren** zugreifen, indem Sie **SSH** in das Suchfeld der Workflowbibliothek eingeben.

---

**Hinweis** Sie können Standardworkflows keine neuen Tags hinzufügen, es sei denn, Sie duplizieren den Workflow.

---

## Erstellen von Workflows im vRealize Orchestrator Client

Sie können den vRealize Orchestrator Client verwenden, um Workflows zu erstellen und zu bearbeiten.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie **Bibliothek > Workflows** aus.
- 3 Klicken Sie auf **Neuer Workflow**.
- 4 Geben Sie den Namen des neuen Workflows ein und klicken Sie auf **Erstellen**.
- 5 Verwenden Sie den Workflow-Editor zum Konfigurieren der Variablen, Workflow-Eingaben und -Ausgaben, der Schemastruktur und der Präsentation des Workflows.
- 6 Um die Bearbeitung des Workflows abzuschließen, klicken Sie auf **Speichern**.

---

**Hinweis** Sie können Änderungen an Workflows auf der Registerkarte **Versionsverlauf** verfolgen. Weitere Informationen finden Sie unter [Versionsverlauf von vRealize Orchestrator-Objekten](#).

---

## Nächste Schritte

Sie können die vRealize Orchestrator-Funktion zur Token-Wiedergabe verwenden, um die Leistung Ihrer Workflows zu optimieren. Weitere Informationen finden Sie unter [Verwenden der Workflow-Token-Wiedergabe im vRealize Orchestrator Client](#).

## Bearbeiten von Workflows und Aktionen über den übergeordneten Workflow

Bearbeiten Sie Workflows und Aktionen direkt vom übergeordneten Workflow aus im vRealize Orchestrator Client.

Das Bearbeiten von untergeordneten Workflows und Aktionen vom übergeordneten Workflow kann dabei helfen, die Entwicklung von Workflows zu rationalisieren.

### Voraussetzungen

Erstellen Sie einen Workflow, der einen anderen Workflow, eine Aktion oder beides aufruft.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Workflows** und wählen Sie Ihren Workflow aus.
- 3 Wählen Sie die Registerkarte **Schema**.
- 4 Doppelklicken Sie in Abhängigkeit vom Objekttyp auf das **Workflow-Element** oder das **Aktionselement** in der Workflow-Arbeitsfläche.
- 5 Bearbeiten Sie das Objekt.
- 6 Um die Bearbeitung des untergeordneten Workflows oder der Aktion abzuschließen, klicken Sie auf **Speichern**.
- 7 Um zum übergeordneten Workflow zurückzukehren, schließen Sie den Objekt-Editor.

## vRealize Orchestrator-Eingabeformular-Designer

Wenn ein Workflow Eingabeparameter erfordert, öffnet er ein Dialogfeld, in dem Benutzer die erforderlichen Werte eingeben. Sie können den Inhalt, das Layout und die Darstellung dieses Dialogfelds mit dem Eingabeformular-Designer organisieren.

Der Eingabeformular-Designer befindet sich auf der Registerkarte **Eingabeformular** des Workflow-Editors. Dieser Designer besteht aus einem Navigationsmenü, einer Design-Arbeitsfläche und einem Eigenschaftenmenü. Sie können Eingaben und generische Elemente aus dem linken Menü auf die Design-Arbeitsfläche ziehen. In der Arbeitsfläche können Sie die Position der Eingabeparameter festlegen, sie in getrennten Eingaberegisterkarten organisieren und die Eigenschaften der Eingabeparameter konfigurieren.

---

**Hinweis** Sie können keine Inhalte von der Registerkarte **Variablen** des Workflow-Editors im Eingabeformular-Designer verwenden. Sie können nur Parameter aus der Registerkarte **Eingabe/Ausgabe** verwenden.

---

### Generische Elemente

Sie können dem Eingabeformular-Designer generische Elemente wie Dropdown-Menüs und Kennworttextfelder hinzufügen. Generische Elemente entsprechen nicht den tatsächlichen Eingabeparametern, können aber an Eingabeparameter gebunden werden.

## Erstellen des Dialogfelds „Workflow-Eingabeparameter“ im vRealize Orchestrator Client

Sie können den Eingabeformular-Designer verwenden, um das Dialogfeld „Workflow-Eingabeparameter“ zu erstellen und anzupassen.

### Voraussetzungen

Vergewissern Sie sich, dass der Workflow eine definierte Liste von Eingabeparametern hat.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Bibliothek > Workflow**.
- 3 Wählen Sie Ihren benutzerdefinierten Workflow aus.
- 4 Klicken Sie auf die Registerkarte **Eingabeformular**.
- 5 (Optional) Erstellen Sie Registerkarten zur Verwendung im Eingabedialogfeld.  
Sie können Registerkarten verwenden, um die Struktur Ihres Dialogfelds zu organisieren.
- 6 Wählen Sie Ihre Eingabeparameter aus.
- 7 Bearbeiten Sie die Eigenschaften der Eingabeparameter.  
Weitere Informationen zu Eingabeparametereigenschaften finden Sie unter [Eingabeparametereigenschaften im vRealize Orchestrator Client](#).
- 8 (Optional) Fügen Sie der Arbeitsfläche generische Elemente hinzu und binden Sie sie an Eingabeparameter.

- 9 (Optional) Fügen Sie den Eingabeparametern eine externe Validierung hinzu. Weitere Informationen finden Sie unter [Verwenden von Aktionen zum Validieren von vRealize Orchestrator-Workflow-Eingaben](#).

- 10 Klicken Sie auf **Speichern**.

## Ergebnisse

Sie haben das Layout des Workflow-Dialogfelds erstellt und die Eigenschaften der Eingabeparameter festgelegt.

## Eingabeparametereigenschaften im vRealize Orchestrator Client

Sie können Parametereigenschaften festlegen, um die Eingabeparameter zu beschränken, die Benutzer beim Ausführen von vRealize Orchestrator-Workflows bereitstellen.

Mit vRealize Orchestrator können Sie die Parametereigenschaften definieren, die zur Quantifizierung der in den Workflows verwendeten Eingabeparameterwerte verwendet werden. Die von Ihnen definierten Parametereigenschaften legen Grenzwerte für Typen und Werte der Eingabeparameter fest, die von Benutzern in vRealize Orchestrator-Workflows bereitgestellt werden können.

Parametereigenschaften validieren die Eingabeparameter und ändern die Darstellung der Textfelder, die im Dialogfeld der Eingabeparameter angezeigt werden. Einige Parametereigenschaften können Abhängigkeiten zwischen Parametern schaffen.

Parametereigenschaft	Beschreibung
<b>Bezeichnung</b>	Legen Sie die Bezeichnung für den Eingabeparameter fest.
<b>Anzeigetyp</b>	Legen Sie den Anzeigetyp für das Eingabetextfeld fest.
<b>Sichtbarkeit</b>	Legen Sie die Sichtbarkeit des Eingabeparameters fest.
<b>Schreibgeschützt</b>	Legen Sie das Eingabetextfeld als schreibgeschützt fest.
<b>Benutzerdefinierte Hilfe</b>	Legen Sie die Wegweiser-Beschreibung für den Eingabeparameter fest.
<b>Standardwert</b>	Legen Sie den Standardwert für den Eingabeparameter fest.
<b>Schritt</b>	Wird für Eingaben vom Typ „Zahlen“ verwendet. Legen Sie fest, um wie viel sich der Wert des Eingabeparameters pro Klick erhöht.
<b>Erforderlich</b>	Legt fest, ob der Wert des Eingabeparameters obligatorisch ist.
<b>Regulärer Ausdruck</b>	Validiert die Eingabe mithilfe eines regulären Ausdrucks.
<b>Minimalwert</b>	Legt den Minimalwert oder die Mindestlänge des Parameters fest.
<b>Maximalwert</b>	Legt den Maximalwert oder die Maximallänge des Parameters fest.

Parametereigenschaft	Beschreibung
<b>Textfeld abgleichen</b>	Legt den Wert für den Eingabeparameter so fest, dass er mit dem Wert eines anderen Eingabeparameters übereinstimmt.
<b>Wertquelle</b>	<p>Legt die Wertquelle der Parametereigenschaften auf den Registerkarten <b>Darstellung</b>, <b>Wert</b> und <b>Beschränkungen</b> fest.</p> <p><b>Hinweis</b> Sie können den Wert externer Aktionen importieren, indem Sie <b>Externe Quelle</b> verwenden. Das Filtern der verfügbaren Aktionen erfolgt nach Parametertyp.</p>

## Verwenden von Aktionen zum Validieren von vRealize Orchestrator-Workflow-Eingaben

Verwenden Sie externe Aktionen, um die Eingaben Ihrer benutzerdefinierten Workflows zu validieren.

### Voraussetzungen

Erstellen Sie einen benutzerdefinierten Workflow mit Eingabeparametern. Weitere Informationen finden Sie unter [Erstellen von Workflows im vRealize Orchestrator Client](#).

Sie können den Eingabeformular-Designer verwenden, um externe Validierungen für Ihre Workflow-Eingaben zu erstellen. Externe Validierungen verwenden Aktionsskripte, die einen Zeichenfolgenwert zurückgeben, wenn der Wert des Eingabeparameters einen Fehler enthält. Wenn der Wert des Eingabeparameters gültig ist, gibt die externe Validierung nichts zurück.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Erstellen Sie eine Validierungsaktion.
  - a Navigieren Sie zu **Bibliothek > Aktionen**.
  - b Klicken Sie auf **Neue Aktion**.
  - c Geben Sie die erforderlichen Informationen auf der Registerkarte **Übersicht** ein.
  - d Geben Sie die Eingabeparameter für die Validierungsaktion ein.

**Hinweis** Die Namen der Eingabeparameter für die Validierungsaktion müssen mit den Namen der Workflow-Eingabeparameter identisch sein, die gerade validiert werden.



- e Geben Sie das Skript der Validierungsaktion auf der Registerkarte **Skript** ein.

```
if (in_1=="invalid") {
    return "in_1 can't be invalid!";
}

if (in_2=="invalid") {
    return "in_2 can't be invalid!";
}

//inputs are valid, return nothing
```

**Hinweis** Das vorangehende Skript ist ein einfaches Beispiel und stellt nicht den vollständigen Geltungsbereich der Validierungsskripte dar, die verwendet werden können.

- f Klicken Sie auf **Speichern**.

### 3 Wenden Sie die externe Validierung an.

- Navigieren Sie zu **Bibliothek > Workflows**.
- Wählen Sie Ihren benutzerdefinierten Workflow aus.
- Wählen Sie die Registerkarte **Eingabeformular** aus.
- Wählen Sie das Zwischenablagensymbol oben links auf dem Bildschirm aus.
- Ziehen Sie ein vRealize Orchestrator-Validierungselement auf die Arbeitsfläche.
- Wählen Sie das Validierungselement aus, geben Sie eine Validierungsbezeichnung ein und wählen Sie die Validierungsaktion aus.
- (Optional) Erstellen Sie zusätzliche Validierungselemente.
- Klicken Sie auf **Speichern**.

### 4 Führen Sie den Workflow aus.

Wenn bei der Validierung ein Fehler auftritt, wird eine Zeichenfolge zurückgegeben. Wenn die Validierung erfolgreich ist, gibt die Validierung nichts zurück, und die Workflow-Ausführung wird fortgesetzt.

#### Ergebnisse

Sie haben eine externe Validierung für Ihren benutzerdefinierten vRealize Orchestrator-Workflow erstellt.

## Anforderungen zur Benutzerinteraktion im vRealize Orchestrator Client

Workflows können weitere Benutzereingaben anfordern, bevor sie abgeschlossen werden können.

Workflows, die weitere Benutzerinteraktionen erfordern, halten Vorgänge an, bis die angeforderten Eingabeparameter vom Benutzer eingegeben werden. Workflows legen fest, welche Benutzer die angeforderten Informationen eingeben können, und senden die Anforderungen für Interaktionen an die entsprechenden Benutzer. Workflows, die auf Benutzereingaben warten, werden im Bereich **Kürzlich ausgeführte Workflows** des vRealize Orchestrator Client-Dashboards und im oberen rechten Benachrichtigungsменю angezeigt.

## Planen von Workflows im vRealize Orchestrator Client

Mithilfe der Planung können Sie Ihre vRealize Orchestrator-Workflow-Ausführungen automatisieren.

Bei der Planung von Workflow-Ausführungen werden Datum, Uhrzeit und Intervalle festgelegt, zu denen die geplante Aufgabe ausgeführt wird.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie Ihren Workflow im Menü **Bibliothek** aus und klicken Sie im Bereich „Workflow“ auf **Zeitplan**.
- 3 Konfigurieren Sie die Parameter für die geplante Aufgabe in den Kategorien **Allgemein**, **Planung** und **Workflow**.

**Hinweis** Die Parameterkategorie **Workflow** wird nur für Workflows angezeigt, die Eingabeparameter benötigen.

Parameter	Beschreibung
<b>Name</b>	Der Name der geplanten Aufgabe.
<b>Beschreibung</b>	Eine kurze Beschreibung mit Details zum Zweck der geplanten Aufgabe.
<b>Starten</b>	Datum und Uhrzeit der ersten geplanten Ausführung des Workflows
<b>Starten, wenn in der Vergangenheit</b>	Wählen Sie aus, ob der Workflow gestartet werden soll, wenn die Zeit in der Vergangenheit liegt. Mit <b>Ja</b> wird der geplante Workflow sofort gestartet. Mit <b>Nein</b> wird der Workflow bei der nächsten geplanten Wiederholung gestartet.
<b>Planen</b>	Legen Sie die Einträge für das wiederkehrende Muster und die Ereignisauslöser der geplanten Aufgabe fest.
<b>Enddatum</b>	Wird nur angezeigt, wenn <b>Keine Wiederholung</b> ausgewählt ist. Legen Sie das Datum und die Uhrzeit fest, zu der die geplante Aufgabe beendet wird.
<b>Workflow</b>	Geben Sie die Eingabeparameter des Workflows ein.

- 4 Klicken Sie auf **Erstellen**.

## Ergebnisse

Sie haben eine geplante Aufgabe für den Workflow erstellt. Geplante Workflows werden unter **Aktivität > Geplant** angezeigt. Sie können geplante Aufgaben löschen, indem Sie im Bereich „Zeitplan“ auf **Löschen** klicken.

## Bearbeiten einer geplanten Aufgabe im vRealize Orchestrator Client

Geplante Aufgaben können bearbeitet werden, um Parameter wie Datum, Uhrzeit und Wiederholung des geplanten Workflows zu ändern.

### Voraussetzungen

Erstellen Sie eine geplante Workflow-Aufgabe.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wählen Sie die geplante Aufgabe unter **Aktivität > Geplant** aus.
- 3 Klicken Sie im Bereich „Workflow“ auf **Bearbeiten**.
- 4 Bearbeiten Sie den Zeitplan und klicken Sie auf **Speichern**.

---

**Hinweis** Eingabeparameter, die beim Erstellen der geplanten Aufgabe festgelegt werden, sind schreibgeschützt und können nicht bearbeitet werden. Wenn Sie diese Parameter ändern möchten, erstellen Sie für diesen Workflow eine neue geplante Aufgabe.

---

# Verwalten von Aktionen

# 6

Sie können Ihre vRealize Orchestrator-Workflows durch Hinzufügen von Aktionsskripten ändern.

Der vRealize Orchestrator Client bietet Bibliotheken mit vordefinierten Aktionen und einen Aktionseditor für benutzerdefinierte Aktionsskripte. Aktionen stellen individuelle Funktionen dar, die Sie als Bausteine in Workflows verwenden.

Aktionen sind JavaScript-Funktionen. Aktionen können mehrere Eingabeparameter besitzen und haben einen einzelnen Rückgabewert. Aktionen können beliebige Objekte in der vRealize Orchestrator-API oder Objekte in einer beliebigen API aufrufen, die Sie mithilfe eines Plug-Ins in vRealize Orchestrator importieren.

Wenn ein Workflow ausgeführt wird, nimmt eine Aktion Eingabeparameter aus den Variablen des Workflows. Diese Variablen können entweder die ursprünglichen Eingabeparameter des Workflows sein oder Variablen, die von anderen Elementen im Workflow bei ihrer Ausführung festgelegt werden.

Der Aktionseditor enthält eine Autovervollständigungsfunktion für Skripte und einen API-Explorer mit den verfügbaren Skripttypen und deren Dokumentation.

Dieses Kapitel enthält die folgenden Themen:

- [Erstellen von Aktionen im vRealize Orchestrator Client](#)
- [Ausführen und Debuggen von Aktionen](#)
- [Kernkonzepte für Python-, Node.js- und PowerShell-Skripts](#)
- [Laufzeitgrenzwerte für Python-, Node.js- und PowerShell-Skripts](#)

## Erstellen von Aktionen im vRealize Orchestrator Client

Sie können den vRealize Orchestrator Client verwenden, um Aktionsskripte zu erstellen, zu bearbeiten und zu löschen.

Ab vRealize Orchestrator 8.1 können Sie die folgenden Laufzeiten verwenden:

- Python 3.7
- Node.js 12

- PowerCLI 11/PowerShell 6.2
- PowerCLI 12/PowerShell 7

---

**Hinweis** Die PowerCLI-Laufzeit umfasst PowerShell und die folgenden Module: VMware.PowerCLI, PowerNSX, PowervRA.

---

### Voraussetzungen

Machen Sie sich vor dem Erstellen eines Python-, Node.js- oder PowerShell-Skripts mit den Kernkonzepten zur Entwicklung kompatibler vRealize Orchestrator-Skripts vertraut, die diese Laufzeiten verwenden. Weitere Informationen finden Sie unter [Kernkonzepte für Python-, Node.js- und PowerShell-Skripts](#).

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Bibliothek > Aktionen**.
- 3 Klicken Sie auf **Neue Aktion**.
- 4 Geben Sie auf der Registerkarte **Allgemein** den Namen und den Modulnamen der Aktion ein.

---

**Hinweis** Der Name und der Modulname müssen für jede Aktion eindeutig sein. Der Aktionsname muss eine gültige JavaScript-Funktion sein. Der Aktionsname muss ein einzelnes Wort sein, das nur Buchstaben, Ziffern und das Dollarzeichen (\$) und den Unterstrich (\_) enthalten kann. Der Name des Moduls muss aus durch Punkt (.) getrennten Wörtern bestehen.

---

- 5 (Optional) Erstellen Sie eine Beschreibung, Versionsnummer, Tags und Gruppenberechtigungen für die Aktion.
- 6 Fügen Sie auf der Registerkarte **Skript** Aktionseingaben hinzu, wählen Sie den Rückgabebetyp der Ausgabe aus und schreiben Sie das Skript.

---

**Hinweis** Durch Auswahl von **Zip** im Dropdown-Menü **Typ** können Sie eine externe Skriptquelle und gegebenenfalls deren Abhängigkeitsmodule importieren.

---

- 7 Um die Bearbeitung der Aktion abzuschließen, klicken Sie auf **Speichern**.  
Eine Meldung gibt an, dass die Aktion gespeichert wurde.

### Nächste Schritte

Ein Anwendungsfallbeispiel zur Verwendung von vRealize Orchestrator-Aktionen finden Sie unter [Vorgehensweise zum Integrieren von Amazon Web Services in vRealize Orchestrator mithilfe von Python](#).

## Ausführen und Debuggen von Aktionen

Sie können Ihre Aktionen verbessern, indem Sie sie direkt über den Aktionseditor ausführen und debuggen.

Ab vRealize Orchestrator 8.1 können Sie Aktionen direkt über den Aktionseditor des vRealize Orchestrator Client ausführen und debuggen. Mit dieser Funktion können Sie gewährleisten, dass Ihre Aktionen wie erwartet ausgeführt werden, nachdem sie in Ihre Workflows integriert wurden.

### Ausführen von Aktionen im vRealize Orchestrator Client

Als Workflow-Designer möchten Sie Ihre Aktionen ausführen, bevor Sie sie in einen Workflow integrieren.

#### Voraussetzungen

Erstellen Sie eine Aktion. Weitere Informationen finden Sie unter [Erstellen von Aktionen im vRealize Orchestrator Client](#).

#### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Aktionen** und wählen Sie die Aktion aus, die Sie ausführen möchten.
- 3 Klicken Sie auf **Ausführen**.
- 4 Geben Sie die angeforderten Eingabeparameter ein und klicken Sie auf **Ausführen**.

Klicken Sie nach Abschluss der Aktionsausführung auf die Registerkarte **Ergebnis/Eingaben**. Wenn bei der Aktionsausführung ein Fehler aufgetreten ist, wird er auf dieser Registerkarte in roter Farbe angezeigt. Sie können die Details der Aktionsausführung über das Element **Aktionsergebnisse** anzeigen.

---

**Hinweis** Die Ergebnisse der Aktionsausführung werden nicht gespeichert.

---

### Debug-Aktionen im vRealize Orchestrator Client

Als Workflow-Designer können Sie Aktionen debuggen, indem Sie Haltepunkte in Ihr Skript einfügen.

vRealize Orchestrator enthält ein integriertes Debugging-Tool, mit dem Sie das Skript und die Eingabeeigenschaften Ihrer Aktion debuggen können. Der Debug-Vorgang kann im Aktionseditor initiiert werden, indem Haltepunkte in die Skriptlinien Ihrer Aktion eingefügt werden.

---

**Hinweis** Das integrierte Debugging-Tool funktioniert nur mit Aktionen, die die Standardlaufzeit von JavaScript verwenden. Ein Beispiel dafür, wie Sie Aktionsskripts mit anderen Laufzeiten debuggen, finden Sie unter [Debuggen der Amazon Web Services-Aktion](#).

---

## Voraussetzungen

Erstellen Sie eine Aktion. Weitere Informationen finden Sie unter [Erstellen von Aktionen im vRealize Orchestrator Client](#).

## Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Aktionen** und wählen Sie die Aktion aus, die Sie debuggen möchten.
- 3 Fügen Sie im Aktionseditor den Zeilen Ihres Aktionsskript, das Sie debuggen möchten, Haltepunkte hinzu.
- 4 Klicken Sie auf **Debuggen**.
- 5 Geben Sie die Eingabeparameter Ihrer Aktion ein und klicken Sie auf **Ausführen**.

Eine Aktionsausführung im Debug-Modus beginnt.

- 6 Wenn die Aktion nach Erreichen eines Haltepunkts angehalten wird, wählen Sie eine der folgenden Optionen:

Option	Beschreibung
<b>Weiter</b>	Setzt die Ausführung der Aktion fort, bis ein anderer Haltepunkt erreicht wird oder die Aktionsausführung abgeschlossen ist.
<b>Springen zu</b>	Springt zur aktuellen Aktionsfunktion. Wenn der Debugger nicht tiefer in die aktuelle Zeile der Funktion gehen kann, führt er einen <b>Überspringen</b> -Vorgang aus.
<b>Überspringen</b>	Der Debugger fährt mit der nächsten Zeile der aktuellen Funktion fort.
<b>Schritt zurück</b>	Der Debugger geht in die Zeile, die bei Rückgabe der aktuellen Funktion ausgeführt wird.

- 7 (Optional) Fügen Sie auf der Registerkarte **Debugger** Ausdrücke hinzu.
- 8 (Optional) Bearbeiten Sie auf der Registerkarte **Debugger** den Wert Ihrer Variablen.

## Kernkonzepte für Python-, Node.js- und PowerShell-Skripts

Beim Erstellen des Skripts zur Verwendung in vRealize Orchestrator müssen Sie sicherstellen, dass das Skript die korrekte Struktur und Formatierung aufweist.

## Unterstützte Laufzeiten

Ab vRealize Orchestrator 8.1 können Sie die folgenden Laufzeiten verwenden:

- Python 3.7
- Node.js 12
- PowerCLI 11/PowerShell 6.2

- PowerCLI 12/PowerShell 7

---

**Hinweis** Die PowerCLI-Laufzeit umfasst PowerShell und die folgenden Module: VMware.PowerCLI, PowerNSX, PowervRA.

---

Sie können den neuen Laufzeiten beliebigen benutzerdefinierten Quellcode hinzufügen. Zur Übernahme von Kontext und Eingaben und Rückgabe von Ergebnissen aus und an das vRealize Orchestrator-Modul, müssen Sie das korrekte Funktionsformat verwenden.

## Empfehlungen zur Skripterstellung

Für einfachere Skripterstellungsaufgaben können Sie **Skriptfähige Aufgabe**-Elemente zu Ihrem Workflow-Schema hinzufügen. Sie können vRealize Orchestrator-Aktionen für komplexere Skripterstellungsaufgaben verwenden.

Der Einsatz von Aktionen bietet zwei spezifische Vorteile:

- Aktionen können unabhängig von Workflows erstellt, aktualisiert, importiert und exportiert werden.
- Aktionen sind eigenständige Objekte, die in ihrer eigenen Umgebung ausgeführt und gedebuggt werden können, was zu einem reibungsloseren Entwicklungsprozess führen kann. Weitere Informationen finden Sie unter [Ausführen und Debuggen von Aktionen](#).

## Anforderungen an die Skriptfunktion

Der Standardname für die Skriptfunktion lautet **handler**. Die Funktion akzeptiert zwei Argumente, Kontext und Eingabe. Beim Kontext handelt es sich um ein Zuordnungsobjekt mit Systeminformationen. Beispielsweise kann `vroURL` die URL der aufzurufenden vRealize Orchestrator-Instanz enthalten, während `executionId` die Token-ID einer Workflow-Ausführung enthält.

Bei einer Eingabe handelt es sich um ein Zuordnungsobjekt mit allen Eingaben, die für die Aktionen bereitgestellt wurden. Wenn Sie beispielsweise eine Eingabe in der Aktion mit dem Namen `myInput` definieren, können Sie je nach Laufzeit über das Eingabeargument, wie z. B. `inputs.myInput` oder `inputs["myInput"]`, darauf zugreifen. Alles, was von der Funktion zurückgegeben wird, gilt als Ergebnis der Aktion. Aus diesem Grund muss der Rückgabotyp der Aktion dem Inhaltstyp entsprechen, der vom Skript in vRealize Orchestrator zurückgegeben wird. Wenn Sie eine einfache Zahl zurückgeben, muss der Rückgabotyp der Aktion ein Zahlentyp sein. Wenn Sie eine Zeichenfolge zurückgeben, muss der Rückgabotyp der Aktion ein Zeichenfolgentyp sein. Wenn Sie ein komplexes Objekt zurückgeben, muss der Rückgabotyp entweder `Properties` oder `Composite` Type zugeordnet werden. Dieselben Prinzipien gelten auch für Arrays.

Unterstützte Eingabe- und Ausgabeparametertypen für Python-, Node.js- und PowerShell-Laufzeiten:

- String
- Number



- Boolean
- Date
- Properties
- Composite Type

## Definieren des Eingabe-Handlers

Standardmäßig wird `handler.handler` als Wert für den Eingabe-Handler verwendet. Dieser Wert bedeutet, dass die vRealize Orchestrator-Engine im ZIP-Paket nach einer Datei auf oberster Ebene mit dem Namen `handler.py`, `handler.js` oder `handler.ps1` sucht, die eine Funktion mit der Bezeichnung `handler` enthält. Alle Unterschiede bei den Namen der Funktion und Handler-Datei müssen sich im Wert des Eingabe-Handlers widerspiegeln. Wenn der Haupt-Handler beispielsweise den Namen `index.js` und die Funktion den Namen `callMe` aufweist, müssen Sie den Wert für den Eingabe-Handler auf **`index.callMe`** setzen.

## Debuggen von Laufzeitskripts in einer externen IDE

vRealize Orchestrator unterstützt das Debuggen von Python- und Node.js-Skripts in einer externen IDE. PowerShell-Skripts können nicht in einer externen IDE gedebuggt werden.

## Laufzeitgrenzwerte für Python-, Node.js- und PowerShell-Skripts

Für bestimmte Python-, Node.js- oder PowerShell-Skripts müssen die Arbeitsspeicher- und Zeitüberschreitungswerte im vRealize Orchestrator Client geändert werden.

Der vRealize Orchestrator Client verwendet einen Satz an standardmäßigen Arbeitsspeicher- und Zeitüberschreitungswerten für Python-, Node.js- und PowerShell-Skripts:

- Arbeitsspeicher: 64 MB
- Zeitüberschreitung: 180 Sekunden

Wenn das Aktionsskript einen oder beide dieser Standardwerte überschreitet, schlägt die Ausführung der Aktion fehl. Beispiel: Im Aktionsskript werden möglicherweise mehrere Abhängigkeitsmodule von Drittanbietern verwendet. In einem solchen Szenario reicht der standardmäßige Arbeitsspeichergrenzwert von 64 MB möglicherweise nicht aus.

Um zu verhindern, dass die Ausführung der Aktionen aufgrund unzureichender Ressourcen fehlschlagen, ändern Sie die Arbeitsspeicher- und Zeitüberschreitungswerte im Aktionseditor.

---

**Hinweis** Sie können das Skript auch in mehrere skriptfähige Aufgabenelemente aufteilen, die den Workflows hinzugefügt werden können.

---

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.

- 2 Navigieren Sie zu **Bibliothek > Aktionen** und wählen Sie die gewünschte Aktion aus.
- 3 Wählen Sie die Registerkarte **Skript** aus.
- 4 Ändern Sie die Werte für Arbeitsspeicher und Zeitüberschreitung unter **Grenzwerte für Laufzeit**.
- 5 Klicken Sie auf **Speichern**.
- 6 Klicken Sie zum Testen der neuen Grenzwerte für die Laufzeit auf **Debuggen**.

# Verwalten von Konfigurationselementen

# 7

Ein Konfigurationselement ist eine Liste mit Variablen, mit deren Hilfe Sie Konstanten über eine gesamte vRealize Orchestrator-Serverbereitstellung hinweg konfigurieren können.

Unter Verwendung von Konfigurationselementen können Sie Variablen für alle Workflows, Aktionen und Richtlinien zur Verfügung stellen, die auf dem vRealize Orchestrator-Server ausgeführt werden.

Wenn Sie ein Paket erstellen, das einen Workflow, eine Aktion oder Richtlinie enthält, die eine Variable aus einem Konfigurationselement verwendet, wird das Konfigurationselement von vRealize Orchestrator automatisch in das Paket eingefügt. Wenn Sie ein Paket mit einem Konfigurationselement auf einen anderen vRealize Orchestrator-Server importieren, können Sie die Variablenwerte des Konfigurationselements ebenso importieren. Beispiel: Sie erstellen einen Workflow, der Variablenwerte benötigt, die sich nach dem vRealize Orchestrator-Server richten, auf dem der Workflow ausgeführt wird. Wenn Sie diese Variablen in einem Konfigurationselement festlegen, können Sie den Workflow exportieren, sodass ein anderer vRealize Orchestrator-Server ihn verwenden kann. Dadurch können Sie Workflows, Aktionen und Richtlinien mithilfe von Konfigurationselementen einfacher zwischen Servern austauschen.

---

**Hinweis** Sie können keine Werte einer Konfigurationselementvariablen aus einem Konfigurationselement importieren, das aus vRealize Orchestrator 5.1 oder früher exportiert wurde.

---

Dieses Kapitel enthält die folgenden Themen:

- [Erstellen von Konfigurationselementen im vRealize Orchestrator Client](#)

## Erstellen von Konfigurationselementen im vRealize Orchestrator Client

Mit Konfigurationselementen können Sie allgemeine Variablen auf einem vRealize Orchestrator-Server festlegen. Alle Elemente, die auf dem Server ausgeführt werden, können die von Ihnen in einem Konfigurationselement festgelegten Variablen verwenden.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.

- 2 Wechseln Sie zu **Assets > Konfigurationen**.
- 3 Wählen Sie **Neue Konfiguration** aus.
- 4 Geben Sie den Namen des Konfigurationselements ein.
- 5 Wählen Sie die Registerkarte **Variablen** aus.
- 6 Klicken Sie zum Erstellen einer lokalen Variable auf **Neu**.
  - a Geben Sie den Variablennamen ein.
  - b Wählen Sie den Variablentyp aus.

---

**Hinweis** Um ein Array von Konfigurationsvariablen zu erstellen, aktivieren Sie das Kontrollkästchen **Array**.

---

- c (Optional) Geben Sie einen Wert für die Konfigurationsvariable ein.
  - d Klicken Sie auf **Speichern**.
- 7 Um das Erstellen eines Konfigurationselements abzuschließen, klicken Sie auf **Speichern**.

#### Nächste Schritte

Mithilfe des Konfigurationselements können Sie Variablen für Workflows, Aktionen oder Richtlinien bereitstellen.

# Verwalten von Richtlinien

# 8

Richtlinien sind Ereignisauslöser, die die Aktivität des Systems überwachen. Richtlinien reagieren auf vordefinierte Ereignisse, die durch Änderungen im Status oder die Leistung bestimmter vRealize Orchestrator-Objekte ausgegeben werden.

Bei Richtlinien handelt es sich um mehrere Regeln, Kontrollen, Schwellenwerte und Ereignisfilter, die bestimmte Workflows oder Skripte ausführen, wenn bestimmte vordefinierte Ereignisse in vRealize Orchestrator oder in den Technologien auftreten, auf die vRealize Orchestrator über Plug-Ins zugreift. vRealize Orchestrator wertet laufend die Richtlinienregeln aus, während die Richtlinie ausgeführt wird. Sie können beispielsweise Richtlinienkontrollen und Schwellenwerte implementieren, die das Verhalten von vCenter Server-Objekten der Typen VC:HostSystem und VC:VirtualMachine überwachen.

Dieses Kapitel enthält die folgenden Themen:

- [Erstellen und Anwenden von Richtlinien im vRealize Orchestrator Client](#)
- [Richtlinienelemente im vRealize Orchestrator Client](#)
- [Verwalten von Richtlinienausführungen im vRealize Orchestrator Client](#)

## Erstellen und Anwenden von Richtlinien im vRealize Orchestrator Client

Sie können Richtlinien verwenden, um die Aktivität des vRealize Orchestrator-Systems für bestimmte Ereignisse zu überwachen.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Richtlinien**.
- 3 Wählen Sie **Neue Richtlinie** aus.  
Sie haben eine leere Richtlinie erstellt.
- 4 Geben Sie einen Richtliniennamen und eine Versionsnummer ein.
- 5 Wählen Sie die Registerkarte **Variablen** aus.

**6** Klicken Sie zum Erstellen einer lokalen Variable auf **Neu**.

- a Geben Sie den Variablennamen ein.
- b Wählen Sie den Variablentyp aus.

---

**Hinweis** Um ein Array von Richtlinienvariablen zu erstellen, aktivieren Sie das Kontrollkästchen **Array** aus.

---

- c Geben Sie den Variablenwert ein.

---

**Hinweis** Um den Wert einer Konfigurationselementvariablen zu importieren, können Sie **An Konfiguration binden** verwenden.

---

- d Klicken Sie auf **Speichern**.

**7** Auf der Registerkarte **Definition** fügen Sie Richtlinienelemente hinzu und legen Ereignishandler fest.

Weitere Informationen zu Richtlinienelementen finden Sie unter [Richtlinienelemente im vRealize Orchestrator Client](#).

**8** Klicken Sie auf **Speichern**.

Sie haben die Richtlinie konfiguriert.

**Nächste Schritte**

Um eine Richtlinie zu starten, wählen Sie die Richtlinie aus und klicken Sie auf **Ausführen**. Geben Sie den Namen der Richtlinienausführung und, wenn Sie dazu aufgefordert werden, die erforderlichen Eingabeparameter ein.

Zur Anzeige des Richtlinienstatus navigieren Sie zu **Aktivität > Richtlinienausführungen**.

## Richtlinienelemente im vRealize Orchestrator Client

Sie können Richtlinienelemente verwenden, um vordefinierte vRealize Orchestrator-Workflows oder -Skripts auszuführen, wenn ein Ereignis eintritt.

Sie können ein Richtlinienelement zum Auslösen von Workflow- oder Skriptausführungen als Reaktion auf Ereignisse hinzufügen, die von Objekten ausgelöst werden. Mit dem periodischen Ereigniselement können Sie Workflow- oder Skriptausführungen planen. Mit dem Root-Element können Sie das Start- oder Stoppverhalten von Richtlinien festlegen. Richtlinienelemente können über Ereignishandler verfügen, die den Zeitpunkt der Ausführung von Richtlinienelementen festlegen.

---

**Hinweis** Ereignishandler, die Richtlinienelemente aktivieren, können entweder Workflows oder Aktionsskripte sein. Wenn Sie einem Ereignishandler sowohl einen Workflow als auch ein Skript hinzufügen, ignoriert die Richtlinie den Skriptauslöser und verwendet nur den Workflow-Auslöser.

---

Ereignishandler	Beschreibung
<b>OnInit</b>	Das Richtlinienelement wird jedes Mal ausgelöst, wenn Sie die Richtlinie starten.
<b>OnExit</b>	Das Richtlinienelement wird jedes Mal ausgelöst, wenn Sie die Richtlinie anhalten.
<b>OnExecute</b>	Wird vom Element für periodische Ereignisse verwendet. Löst das Richtlinienelement während des im Element für periodische Ereignisse angegebenen Zeitraums aus.

**Hinweis** Technologien, die an die vRealize Orchestrator-Datenbank angeschlossen sind, können über eindeutige Ereignishandler verfügen. Über das SNMP-Plug-In können Sie beispielsweise den **OnTrap**-Ereignishandler beim Erstellen von SNMP-basierten Richtlinienelementen verwenden.

Richtlinienelemente werden auf der Registerkarte **Definition** des Richtlinienbearbeitungsfensters konfiguriert.

## Verwalten von Richtlinienausführungen im vRealize Orchestrator Client

Sie können den vRealize Orchestrator Client verwenden, um die Richtlinienpriorität und das Serverstartverhalten von Richtlinien zu verwalten, wenn der vRealize Orchestrator-Server neu gestartet wird.

### Voraussetzungen

Erstellen Sie eine Richtlinie und führen Sie sie aus. Weitere Informationen finden Sie unter [Erstellen und Anwenden von Richtlinien im vRealize Orchestrator Client](#).

### Verfahren

**1** Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.

**2** Wechseln Sie zu **Aktivität > Richtlinienausführungen**.

**3** Klicken Sie auf die Richtlinienausführung, die Sie verwalten möchten.

**4** Klicken Sie auf **Beenden**.

Der Richtlinienstatus wird in **Angehalten** geändert.

**5** Legen Sie auf der Registerkarte **Allgemein** die Richtlinienpriorität und das Serverstartverhalten fest.

**6** Um die Richtlinie neu zu starten, klicken Sie auf **Ausführen**.

Der Richtlinienzustand wird in **Wird ausgeführt** geändert.

# Verwalten von Ressourcenelementen

## 9

Workflows können Objekte verwenden, die Sie unabhängig von vRealize Orchestrator als Attribute erstellen. Zur Nutzung externer Objekte als Attribute in Workflows importieren Sie diese als Ressourcenelemente auf den Server.

Zu Objekten, die von vRealize Orchestrator-Workflows als Ressourcenelemente verwendet werden können, gehören Image-Dateien, Skripte, XML-Vorlagen, HTML-Dateien usw. Workflows, die auf dem vRealize Orchestrator-Server ausgeführt werden, können beliebige in vRealize Orchestrator importierte Ressourcenelemente verwenden.

Nach dem Import eines Objekts als Ressourcenelement in vRealize Orchestrator können Sie Änderungen am Objekt in einem einzelnen Speicherort vornehmen und diese Änderungen automatisch an alle Workflows weiterleiten, die dieses Ressourcenelement verwenden.

Die Maximalgröße eines Ressourcenelements beträgt 16 MB.

Sie können ein Ressourcenelement importieren, exportieren, wiederherstellen, aktualisieren und löschen.



# Verwalten von Paketen

# 10

Verwenden Sie den vRealize Orchestrator Client zum Erstellen, Exportieren und Importieren von Paketen. Pakete können zum Exportieren von Workflow-Objekten zur Nutzung in anderen vRealize Orchestrator-Instanzen verwendet werden.

Pakete können Workflows, Aktionen, Richtlinien, Konfigurationselemente oder Ressourcenelemente enthalten.

Wenn Sie einem Paket ein Element hinzufügen, führt vRealize Orchestrator eine Überprüfung auf Abhängigkeiten durch und fügt abhängige Elemente zum Paket hinzu. Beispiel: Wenn Sie einen Workflow hinzufügen, der Aktionen oder andere Workflows verwendet, fügt vRealize Orchestrator diese Aktionen und Workflows dem Paket hinzu.

Wenn Sie ein Paket importieren, vergleicht der Server die Versionen der verschiedenen Elemente seines Inhalts mit übereinstimmenden lokalen Elementen. Der Vergleich zeigt Unterschiede in den Versionen zwischen den lokalen und importierten Elementen. Der Benutzer kann entscheiden, ob das Paket importiert wird, oder bestimmte Elemente zum Importieren auswählen.

Für die meisten in vRealize Orchestrator Client erstellten Objekte stellen neben den Ressourcenelementen Pakete die einzige Möglichkeit zum Exportieren und Importieren dieser Objekte dar.

Pakete verwenden Digital Rights Management, um zu kontrollieren, wie der empfangende Server die Inhalte des Pakets verwenden kann. vRealize Orchestrator signiert die Pakete und verschlüsselt die Pakete aus Datenschutzgründen. Pakete können mithilfe von X509-Zertifikaten verfolgen, welche Benutzer Elemente exportieren und weiterverteilen.

## Erstellen eines Pakets im vRealize Orchestrator Client

Sie können Workflows, Richtlinien, Aktionen, Plug-In-Referenzen, Ressourcenelemente und Konfigurationselemente in Paketen exportieren und importieren. Alle abhängigen Elemente, die mit Paketobjekten in Beziehung stehen, werden automatisch zum Paket hinzugefügt, um Kompatibilität zwischen den Versionen sicherzustellen. Zum Löschen abhängiger Elemente müssen Sie zunächst das zugehörige Paketobjekt entfernen.

Für die meisten in vRealize Orchestrator Client erstellten Objekte stellen neben den Ressourcenelementen Pakete die einzige Möglichkeit zum Exportieren und Importieren dieser Objekte dar.

### Voraussetzungen

Stellen Sie sicher, dass der vRealize Orchestrator-Server Objekte wie Workflows, Aktionen und Richtlinien enthält, die Sie einem Paket hinzufügen können.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Assets > Pakete**.
- 3 Klicken Sie auf **Neues Paket**.
- 4 Geben Sie auf der Registerkarte **Allgemein** einen Namen und eine Beschreibung für das Paket ein.

---

**Hinweis** Beim Benennen von Paketen im vRealize Orchestrator Client dürfen keine Sonderzeichen verwendet werden.

---

- 5 Klicken Sie auf der Registerkarte **Inhalt** auf **Hinzufügen**.
- 6 Wählen Sie die Objekte aus, die Sie zum Paket hinzufügen möchten, und klicken Sie auf **Hinzufügen**.

---

**Hinweis** Abhängige Elemente werden automatisch zum Paket hinzugefügt, während der Paketerstellung aber nicht auf der Registerkarte **Inhalt** angezeigt. Um abhängige Elemente anzuzeigen, wählen Sie nach der Paketerstellung die Registerkarte **Inhalt** aus.

---

- 7 Um das Erstellen des Pakets abzuschließen, klicken Sie auf **Erstellen**.

## Exportieren eines Pakets im vRealize Orchestrator Client

Sie können den vRealize Orchestrator Client verwenden, um Pakete in eine andere vRealize Orchestrator-Umgebung zu exportieren.

### Voraussetzungen

Erstellen Sie ein Paket, das die zu exportierenden vRealize Orchestrator-Objekte enthält. Weitere Informationen finden Sie unter [Erstellen eines Pakets im vRealize Orchestrator Client](#).

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Assets > Pakete**.
- 3 Klicken Sie für das Paket auf **Exportieren**.

#### 4 (Optional) Wählen Sie zusätzliche Exportoptionen.

Option	Beschreibung
<b>Attributwerte der Konfiguration zu Paket hinzufügen</b>	Exportiert die Attributwerte der Konfigurationselemente.
<b>Attributwerte der SecureString-Konfiguration zu Paket hinzufügen</b>	Exportiert die Attributwerte der SecureString-Konfiguration.
<b>Globale Tags zu Paket hinzufügen</b>	Exportiert die globalen Tags.

#### 5 Legen Sie die Zugriffsrechte für Benutzer fest, die das Paket importieren.

Option	Beschreibung
<b>Inhalte anzeigen</b>	Der Benutzer kann den Paketinhalt anzeigen.
<b>Zu Paket hinzufügen</b>	Der Benutzer kann Inhalte aus dem importierten Paket anderen Paketen hinzufügen.
<b>Inhalte bearbeiten</b>	Der Benutzer kann den Paketinhalt bearbeiten.

#### 6 Klicken Sie auf **OK**.

**Hinweis** Auf Ihrem lokalen Computer werden Dateien mit der Erweiterung `.package` in einem Standardordner gespeichert. Um einen benutzerdefinierten Ordner festzulegen, können Sie die Speichereinstellungen in Ihrem Browser ändern.

### Ergebnisse

Sie haben das Paket exportiert. Sie können die exportierten Objekte nun in einer anderen vRealize Orchestrator-Umgebung verwenden.

## Importieren eines Pakets im vRealize Orchestrator Client

Zum Importieren von Workflow-Paketen können Sie den vRealize Orchestrator Client verwenden. Durch Importieren von Paketen können Sie Objekte eines vRealize Orchestrator-Servers auf einem anderen Server wiederverwenden.

### Voraussetzungen

- Sichern Sie alle vRealize Orchestrator-Standardobjekte, die Sie geändert haben.
- Erstellen Sie auf dem Remoteserver ein Paket mit den zu importierenden Objekten und exportieren Sie es.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Assets > Pakete**.

- 3 Klicken Sie auf **Importieren**, navigieren Sie zur `.package`-Datei, die importiert werden soll, und klicken Sie auf **Öffnen**.
- 4 Überprüfen Sie die Informationen des importierten Pakets.
  - a Die Registerkarte **Allgemein** enthält Informationen zum importierten Paket wie Name, Beschreibung, Anzahl der darin enthaltenen Elemente und Zertifikatsinformationen.  
  
Vor dem Importieren der Datei werden Sie möglicherweise dazu aufgefordert, dem Zertifikat des Herausgebers der vRealize Orchestrator-Quellinstanz zu vertrauen.
  - b Auf der Registerkarte **Paketelemente** sind die in der Importdatei enthaltenen Objekte aufgelistet. Wenn die Version eines Objekts im Paket eine höhere Version aufweist als der Server, wird diese Objektversion vom System zum Importieren ausgewählt. Frühere Versionen von vRealize Orchestrator-Elementen müssen manuell ausgewählt werden.
  - c Deaktivieren Sie die Option **Attributwerte der Konfiguration importieren**, wenn Sie die Attributwerte der Konfigurationselemente aus dem Paket nicht importieren möchten.
  - d Geben Sie im Dropdown-Menü an, ob Sie Tags importieren möchten.
- 5 Klicken Sie auf **Importieren**.

# Fehlerbehebung im vRealize Orchestrator Client

# 11

Sie können Fehlerbehebung und Überwachung der vRealize Orchestrator-Instanz mithilfe von Metriken, Token-Wiedergabe, Validierung und Debugging durchführen.

Dieses Kapitel enthält die folgenden Themen:

- [Metrikdaten im vRealize Orchestrator Client](#)
- [Verwenden der Workflow-Token-Wiedergabe im vRealize Orchestrator Client](#)
- [Validieren von vRealize Orchestrator-Workflows](#)
- [Debuggen von Workflow-Skripts im vRealize Orchestrator Client](#)
- [Debuggen von Workflows nach Schemaelement](#)

## Metrikdaten im vRealize Orchestrator Client

vRealize Orchestrator-Administratoren können Workflow-Profilerstellung und die Metriken des System-Dashboards zur Fehlerbehebung des vRealize Orchestrator-Systems und der Workflows verwenden.

Die Profilerstellungsfunktion erfasst Metrikdaten zu Workflow-Ausführungen. Die Workflow-Profilerstellung ist standardmäßig aktiviert. Sie können die automatische Profilerstellung unter **Control Center > Erweiterungseigenschaften > profiler-8.2.0** deaktivieren.

Die andere Quelle für Metrikdaten im vRealize Orchestrator Client ist das System-Dashboard, das Metriken auf Systemebene bereitstellt. Weitere Informationen finden Sie unter [Verwenden des vRealize Orchestrator-System-Dashboards](#).

## Erstellen von Profilen für Workflows im vRealize Orchestrator Client

Sie können für Workflow-Ausführungen ein Profil erstellen, um in Ihrer vRealize Orchestrator-Umgebung Fehlerbehebung und Optimierungen durchzuführen.

Mithilfe der Profilerstellungsfunktion des vRealize Orchestrator Client können Sie nützliche Metrikdaten zu Ihren Workflow-Ausführungen erfassen. Diese Daten können Sie zur Optimierung der Leistung Ihrer Workflows verwenden. Standardmäßig werden für Workflow-Ausführungen automatisch Profile erstellt. Sie können die automatische Profilerstellung auf der Seite **Erweiterungseigenschaften** des vRealize Orchestrator Control Center deaktivieren und den Profiler manuell ausführen. Um eine manuelle Profilerstellung durchzuführen, suchen Sie nach Ihrem Workflow in der Bibliothek und wählen Sie **Aktionen > Profil** aus.

### Voraussetzungen

Führen Sie einen Workflow aus.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Wechseln Sie zu **Aktivität > Workflow-Ausführungen**.
- 3 Wählen Sie eine Workflow-Ausführung aus.

Im Workflow-Ausführungsschema können Sie Daten zu den einzelnen Workflowelementen sehen. Zu den Daten zählen die gesamte Ausführungsdauer, die maximale Dauer und die Anzahl der Elementausführungen. Sie können diese Informationen aus dem Dropdown-Menü oben rechts auf der Seite herausfiltern.

- 4 Wählen Sie die Registerkarte **Leistung** aus.

Diese Registerkarte bietet Ihnen Metrikdaten zu den CPU-Zeiten der Workflow-Ausführung, zur Ausführungsdauer, zur Tokengröße und zu den Workflow-Elementdaten.

---

**Hinweis** Wenn die Workflow-Ausführung unterbrochen wird, weil der Workflow beispielsweise auf eine weitere Eingabe wartet, wird mit der Metrik der CPU-Zeiten nur der Laufzeit-Thread vor der Fertigstellung erfasst.

---

### Nächste Schritte

Verwenden Sie die von der Profilerstellung erfassten Daten, um Ihren Workflow zu optimieren.

## Verwenden des vRealize Orchestrator-System-Dashboards

Als Administrator können Sie das vRealize Orchestrator Client-System-Dashboard verwenden, um nützliche Metrikdaten über die Knoten der vRealize Orchestrator-Umgebung zu erfassen.

Sie können auf das System-Dashboard zugreifen, indem Sie auf die Registerkarte **System** oben auf der Seite des vRealize Orchestrator Client-Dashboards klicken. Die bereitgestellten Daten umfassen:

- Knotenstatus
- Knoteneigenschaften

- Clustereinstellungen. Sie können die Clustereinstellungen nur über das System-Dashboard anzeigen. Um diese Einstellungen zu ändern, wechseln Sie zur Seite **Verwaltung des Orchestrator-Clusters** des vRealize Orchestrator Control Center.
- Thread-Informationen
- Heap-Speicher
- Nicht-Heap-Speicher
- Dateisystemverwendung
- Authentifizierungsdaten
- Verbindungspools für Orchestrator-Datenbank
- Prozesseingabeargumente

Diese Daten können verwendet werden, um den Status einzelner Knoten Ihrer vRealize Orchestrator-Umgebung zu überwachen und Probleme zu beheben. Um zwischen einzelnen Knoten zu navigieren, klicken Sie auf die Registerkarte, die mit einem Knoten im oberen Bereich des System-Dashboards verknüpft ist.

## Verwenden der Workflow-Token-Wiedergabe im vRealize Orchestrator Client

Sie können die Token-Wiedergabefunktion verwenden, um die Übergänge zwischen Elementen in Workflow-Ausführungen anzuzeigen.

Die Token-Wiedergabefunktion zeichnet Kontextinformationen für jeden Übergang zwischen Workflowelementen auf. Für jedes Workflow-Element werden bei der Token-Wiedergabe der Start und das Ende der Workflow-Ausführung sowie die Variablen aufgezeichnet, die am Ende der Ausführung des Workflow-Elements geändert wurden. Die Token-Wiedergabe referenziert auch die generierten Skriptprotokollmeldungen für jedes Workflowelement.

---

**Hinweis** Daten zu Workflow-Elementübergängen werden in der vRealize Orchestrator-PostgreSQL-Datenbank gespeichert. Diese Daten werden aus der Datenbank entfernt, wenn die Workflow-Ausführung gelöscht wird.

---

### Voraussetzungen

- Aktivieren Sie die Token-Wiedergabefunktion vom Control Center aus.
  - a Melden Sie sich beim Control Center als **root** an.
  - b Wählen Sie **Erweiterungseigenschaften**.
  - c Klicken Sie auf **tokenreplay-8.2.0**.
  - d Um die Token-Wiedergabefunktion zu aktivieren, klicken Sie auf **Aktivieren**.

- e Klicken Sie auf **Speichern**.

---

**Hinweis** Es kann bis zu 5 Minuten dauern, bis der vRealize Orchestrator-Server die Erweiterung aktualisiert hat.

---

- Führen Sie einen Workflow aus.

---

**Hinweis** Standardmäßig wird die Token-Wiedergabe nicht automatisch für alle Workflow-Läufe auf Ihrem vRealize Orchestrator-Server ausgeführt. Sie können Token-Wiedergaben für jeden Workflow einzeln ausführen oder die Token-Wiedergabe-Erweiterung für alle Workflows auf der Seite **Erweiterungseigenschaften** des Control Centers aktivieren.

---

#### Verfahren

- 1 (Optional) Aktivieren Sie die Token-Wiedergabe für alle Workflow-Ausführungen auf Ihrem vRealize Orchestrator-Server.

---

**Hinweis** Um einzelne Token-Wiedergaben auszuführen, ohne die Funktion vom Control Center aus zu aktivieren, klicken Sie auf **Mit Wiedergabe ausführen** auf der Workflow-Editor-Seite.

---

- a Melden Sie sich beim Control Center als **root** an.
- b Wählen Sie **Erweiterungseigenschaften**.
- c Klicken Sie auf **tokenreplay-8.2.0**.
- d Um die Token-Wiedergabefunktion für alle Workflows zu aktivieren, vergewissern Sie sich, dass **Wiedergabe für alle Workflow-Ausführungen aufzeichnen** aktiviert ist.
- e Klicken Sie auf **Speichern**.

---

**Hinweis** Es kann bis zu 5 Minuten dauern, bis der vRealize Orchestrator-Server die Erweiterung aktualisiert hat.

---

- 2 Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.
- 3 Wechseln Sie zu **Aktivität > Workflow-Ausführungen**.
- 4 Wählen Sie eine Workflow-Ausführung aus.
- 5 Wählen Sie im linken Menü ein Element für die Workflow-Ausführung aus.

Die Registerkarten **Variable** und **Protokolle** zeigen jetzt die für dieses Workflowelement spezifischen Informationen an.

## Validieren von vRealize Orchestrator-Workflows

vRealize Orchestrator bietet ein Tool zur Workflow-Validierung. Durch das Validieren eines Workflows können Sie Fehler im Workflow erkennen und überprüfen, ob die Daten von einem Element zum nächsten korrekt fließen.



Standardmäßig führt vRealize Orchestrator immer eine Workflow-Validierung durch, wenn ein Workflow ausgeführt wird.

Wenn Sie einen Workflow validieren, erstellt das Validierungstool eine Liste der Fehler oder Warnungen. Durch einen Klick auf einen Fehler in der Liste wird das Workflowelement markiert, in dem der Fehler aufgetreten ist.

Wenn Sie das Validierungstool im Workfloweditor ausführen, schlägt das Tool schnell Korrekturen für die erkannten Fehler vor. Bestimmte Schnellkorrekturen erfordern die Eingabe von zusätzlichen Informationen oder Eingabeparametern. Andere Schnellkorrekturen beheben den Fehler an Ihrer Stelle.

Die Workflowvalidierung überprüft die Datenbindungen und Verbindungen zwischen Elementen. Die Workflowvalidierung prüft nicht die Datenverarbeitung, die jedes Element im Workflow durchführt. Dies kann dazu führen, dass ein gültiger Workflow falsch abläuft und fehlerhafte Ergebnisse liefert, wenn eine Funktion in einem Schemaelement fehlerhaft ist.

## Validieren eines Workflows und Beheben von Validierungsfehlern im vRealize Orchestrator Client

Sie müssen einen Workflow validieren, bevor Sie ihn ausführen können. Validierungsfehler können Sie nur beheben, wenn Sie den Workflow zur Bearbeitung geöffnet haben.

### Voraussetzungen

Überprüfen Sie, ob Sie einen vollständigen Workflow zum Validieren haben, in dem Schemaelemente verknüpft und Bindungen definiert sind.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.
- 2 Navigieren Sie zu **Bibliothek > Workflows** und wählen Sie den Workflow aus, der validiert werden soll.
- 3 Klicken Sie auf **Bearbeiten**.
- 4 Klicken Sie im oberen Menü auf **Validieren**.

Wenn der Workflow gültig ist, erscheint eine Bestätigungsmeldung. Wenn der Workflow ungültig ist, erscheint eine Liste der Fehler.

- 5 Klicken Sie für einen ungültigen Workflow auf eine Fehlermeldung und führen Sie die entsprechenden Schritte zur Behebung des Problems durch.

Das Validierungstool markiert das Schemaelement mit dem Fehler durch ein rotes Symbol. Wenn möglich, zeigt das Validierungstool auch eine Aktion zur schnellen Problembehebung.

- Wenn Sie mit der Aktion zur schnellen Problembehebung einverstanden sind, klicken Sie darauf, um diese Aktion vorzunehmen.

- Wenn Sie mit der vorgeschlagenen Aktion zur schnellen Problembehebung nicht einverstanden sind, schließen Sie das Dialogfeld für die Workflowvalidierung und beheben Sie das Problem des Schemaelements manuell.

---

**Wichtig** Prüfen Sie immer, ob die von vRealize Orchestrator vorgeschlagene Fehlerbehebung geeignet ist.

---

Beispielsweise kann die vorgeschlagene Aktion das Löschen eines nicht benötigten Attributs sein, während das wirkliche Problem in einer fehlerhaften Bindung des Attributs liegt.

- 6 Wiederholen Sie die vorhergehenden Schritte, bis Sie alle Validierungsfehler eliminiert haben.

### Ergebnisse

Sie haben einen Workflow validiert und Validierungsfehler behoben.

### Nächste Schritte

Sie können den Workflow jetzt ausführen.

## Debuggen von Workflow-Skripts im vRealize Orchestrator Client

Sie können ein Debugging der Workflow-Ausführungen durchführen, indem Sie Unterbrechungspunkte in das Skript von Workflowelementen einfügen.

Bei Erreichen eines Unterbrechungspunkts haben Sie verschiedene Optionen, den Debugging-Vorgang fortzusetzen. Wenn Sie das Debugging eines Elements aus dem Workflowschema durchführen, können Sie die allgemeinen Informationen zur Ausführung des Workflows anzeigen, die Workflowvariablen ändern, zu überwachende Ausdrücke hinzufügen und Protokollmeldungen anzeigen.

---

**Hinweis** Führen Sie das gesamte Skript-Debugging in einer Nicht-Produktionsumgebung durch.

---

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator-Client als Administrator an.
- 2 Wählen Sie einen Workflow aus der Bibliothek aus.
- 3 Öffnen Sie das Workflow Schema, wählen Sie ein Workflowelement aus und klicken Sie auf die Registerkarte **Skripterstellung**.
- 4 Um einen Unterbrechungspunkt einzufügen, klicken Sie auf den roten Kreis links neben der Zeilennummer.

---

**Hinweis** Sie können nur Unterbrechungspunkte in Workflowelementen mit Skripterstellung einfügen.

---

- 5 Um den Workflow im Debugging-Modus auszuführen, klicken Sie auf **Debuggen**.

Wenn der Workflow Eingabeparameter erfordert, müssen Sie diese angeben.

- 6 Wenn die Ausführung des Workflows nach Erreichen eines Unterbrechungspunkts angehalten wird, wählen Sie eine der verfügbaren Optionen.

Option	Beschreibung
<b>Weiter</b>	Setzt die Ausführung des Workflows fort, bis ein anderer Unterbrechungspunkt erreicht ist oder die Workflowausführung abgeschlossen ist.
<b>Springen zu</b>	Sie können diese Option verwenden, um zu einem Workflow-Element zu springen. Sie können nicht zu einem verschachtelten Workflowelement springen, wenn Sie einen Workflow im Workfloweditor bearbeiten.
<b>Überspringen</b>	Überspringt das aktuelle Element im Schema und hält die Ausführung des Workflows beim nächsten Element an.

**Hinweis** Sie können den Debugger anweisen, den aktuellen Unterbrechungspunkt zu ignorieren, indem Sie darauf klicken. Dadurch wird das Unterbrechungspunktsymbol in ein grünes Dreieck geändert.

- 7 (Optional) Fügen Sie auf der Registerkarte **Debugger** die zu überwachenden Ausdrücke ein. Sie können Ausdrücke verwenden, um den Abschluss bestimmter Variablen zu verfolgen.
- 8 (Optional) Ändern Sie auf der Registerkarte **Debugger** die Werte von Variablen.

## Debuggen von Workflows nach Schemaelement

Als Workflow-Designer können Sie einzelne Schemaelemente debuggen.

### Verfahren

- 1 Melden Sie sich beim vRealize Orchestrator Client an.
- 2 Navigieren Sie zu **Bibliothek > Workflows** und wählen Sie Ihren Workflow aus.
- 3 Wählen Sie die Registerkarte **Schema**.
- 4 Wählen Sie das Workflow-Element aus, das Sie debuggen möchten, und klicken Sie oben links im Element auf die Debug-Schaltfläche.

**Hinweis** Durch das Hinzufügen eines Haltepunkts zu einem Schemaelement des Typs **Workflow-Element** können Sie untergeordnete Workflows direkt aus dem übergeordneten Workflow debuggen. Wenn der Debugger das Schemaelement **Workflow-Element** erreicht, wird die Schemaansicht des untergeordneten Workflows geöffnet.

- 5 Wiederholen Sie den Vorgang für alle anderen Schemaelemente, die Sie debuggen möchten.
- 6 Klicken Sie auf **Debuggen**.

- 7 Geben Sie die angeforderten Eingabeparameterwerte ein und klicken Sie auf **Ausführen**.

Die Ausführung des Workflows beginnt und wird angehalten, wenn der Debugger ein Schemaelement mit einem Haltepunkt erreicht.

- 8 Wenn Sie sich bei einem Haltepunkt befinden, wählen Sie eine der folgenden Optionen aus:

Option	Beschreibung
<b>Weiter</b>	Setzt die Ausführung des Workflows fort, bis ein anderer Haltepunkt erreicht ist oder die Workflowausführung abgeschlossen ist.
<b>Springen zu</b>	Springen Sie zur Funktion des aktuellen Workflows. Wenn der Debugger nicht tiefer in die aktuelle Zeile der Funktion gehen kann, führt er einen <b>Überspringen</b> -Vorgang aus.
<b>Überspringen</b>	Der Debugger fährt mit der nächsten Zeile der aktuellen Funktion fort.
<b>Schritt zurück</b>	Der Debugger geht in die Zeile, die bei Rückgabe der aktuellen Funktion ausgeführt wird.

- 9 (Optional) Bearbeiten Sie auf der Registerkarte **Variablen** den Wert Ihrer Workflow-Variablen.

The screenshot displays the VMware vRealize Orchestrator Client interface during a debugging session. The top bar shows the workflow name 'debug.by.schema test' and its status 'Debugging'. Below this, a workflow diagram on the left shows a sequence of steps, with 'My Orchestrator Task' highlighted. The right pane shows the 'Scripting' tab for 'Item1 (My Orchestrator Task)', displaying a JavaScript script. The bottom section is divided into two panels: 'Watch expressions' on the left and a variable watch table on the right.

**Scripting Code:**

```

1 var i=wfId.length;
2 for (var i=0; i < len; i++) {
3   {
4     var VM = VMs[i];
5     //var workflowLaunch = Server.getWorkflowById("workflowId");
6     var workflowLaunch = workflowId;
7     if (workflowLaunch == null) {
8       throw "Workflow not found";
9     }
10    var workflowParameters = new Properties();
11    workflowParameters.put("vm",VM);
12    var wfToken = workflowLaunch.execute(workflowParameters);
13    System.log ("Run workflow on " +VM.name);

```

**Variable Watch Table:**

Variable	Value
i	Not set
VM	test
workflowToLaunch	Not set
workflowParameters	Not set
wfToken	Not set
VM	test
workflowToRun	test