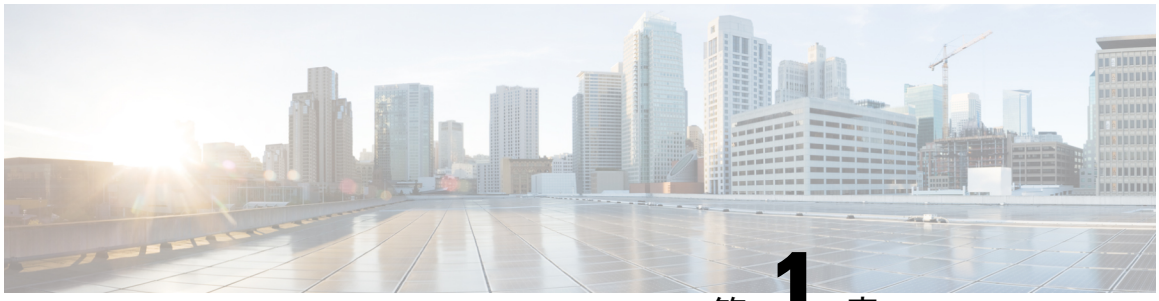
Cisco Crosswork Workflow Manager 1.1 管理者ガイド

最終更新：2024 年 8 月 7 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/



第 1 章

OVA を使用した CWM のインストール

ここでは、次の内容について説明します。

OVA を使用した CWM のインストール

Crosswork Workflow Manager 1.1 は、VMware vSphere 6.7（およびそれ以降）仮想化プラットフォームを使用してOVAイメージを展開することにより、ゲスト仮想マシンとしてインストールされます。

前提条件

- **ed25519** SSH 公開キーと秘密キーのペア。

システム要件

最小システム要件	
サーバー	ESXi 6.7 以降のホストを使用する **VMware vSphere 6.7 以降** のアカウント
CPU	8 コア
メモリ	64 GB
ストレージ	100 GB

CWM パッケージのダウンロード

CWM 1.1 ソフトウェアパッケージを取得するには、次の手順を実行します。

ステップ 1 シスコ ソフトウェア ダウンロード サービスに移動し、検索バーに「**Crosswork Workflow Manager**」と入力し、検索リストからこれを選択します。

ステップ 2 [ソフトウェアタイプの選択（Select a software type）] から、[Crosswork Workflow Manager ソフトウェア（Crosswork Workflow Manager Software）] を選択します。

ステップ 3 Linux 用の Crosswork Workflow Manager ソフトウェアパッケージをダウンロードします。

ステップ 4 端末で、`sh` コマンドを使用して、ダウンロードした **.signed.bin** ファイルを抽出し、証明書を検証します。参考として、次の出力例を参照してください。

```
sh cwm-1.1.signed.bin
Unpacking...
Verifying signature...
Retrieving CA certificate from http://www.cisco.com/security/pki/certs/crcam2.cer ...
Successfully retrieved and verified crcam2.cer.
Retrieving SubCA certificate from http://www.cisco.com/security/pki/certs/innerspace.cer ...
Successfully retrieved and verified innerspace.cer.
Successfully verified root, subca and end-entity certificate chain.
Successfully fetched a public key from tailf.cer.
Successfully verified the signature of cwm-1.1.tar.gz using tailf.cer
```

`cwm-1.1.tar.gz` ファイルおよびその他のファイルが抽出され、署名ファイルに対して検証されています。

ステップ 5 `cwm-1.1.tar.gz` ファイルを抽出するには、ファイルをダブルクリックするか（Mac ユーザー）、gzip ユーティリティを使用します（Linux および Windows ユーザー）。これにより、インストールに使用される CWM OVA ファイルが抽出されます。

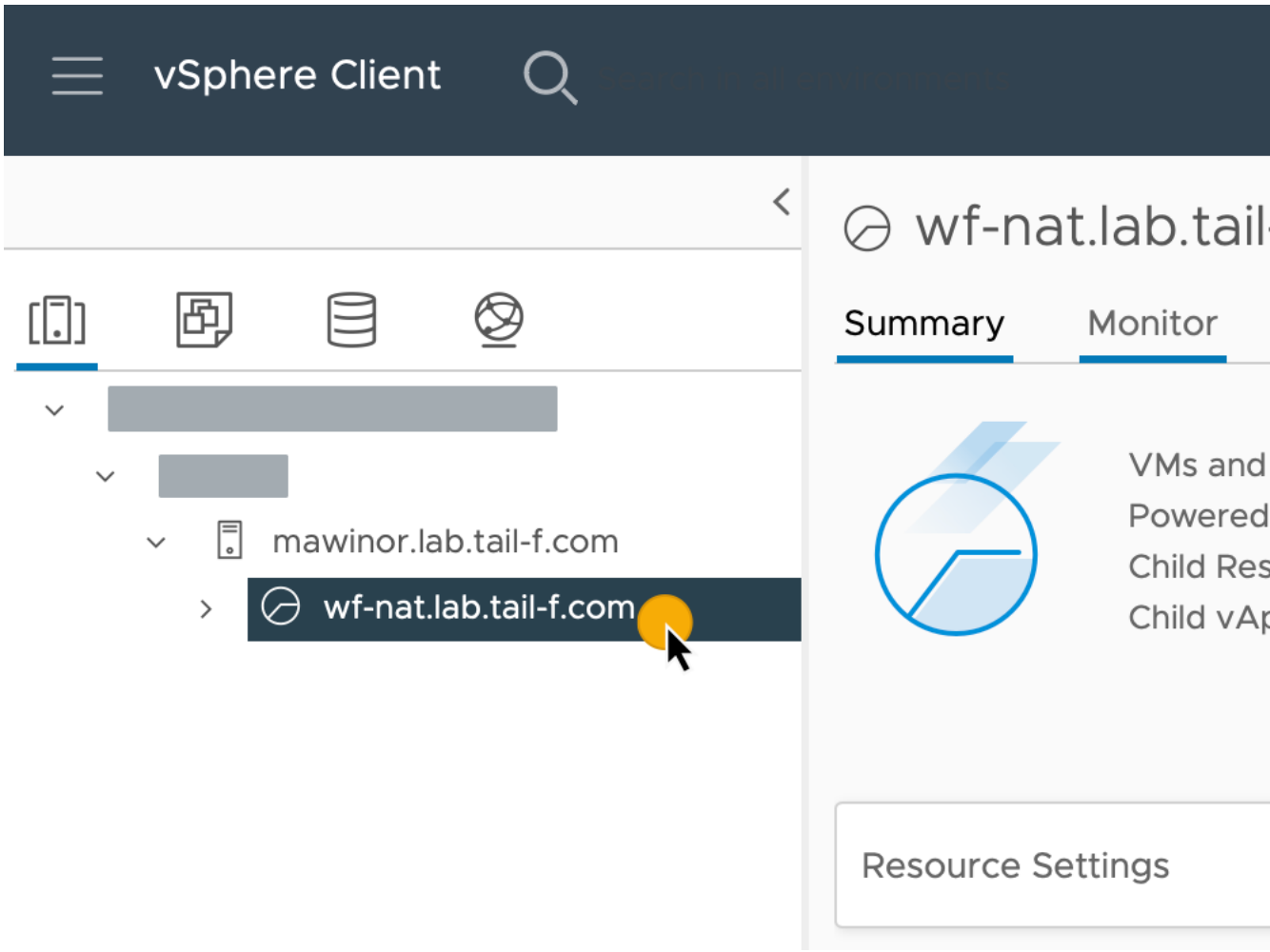
OVA の展開と VM の起動

ダウンロードした OVA イメージを使用して仮想マシンを作成するには、次の手順を実行します。

ステップ 1 vSphere アカウントにログインします。

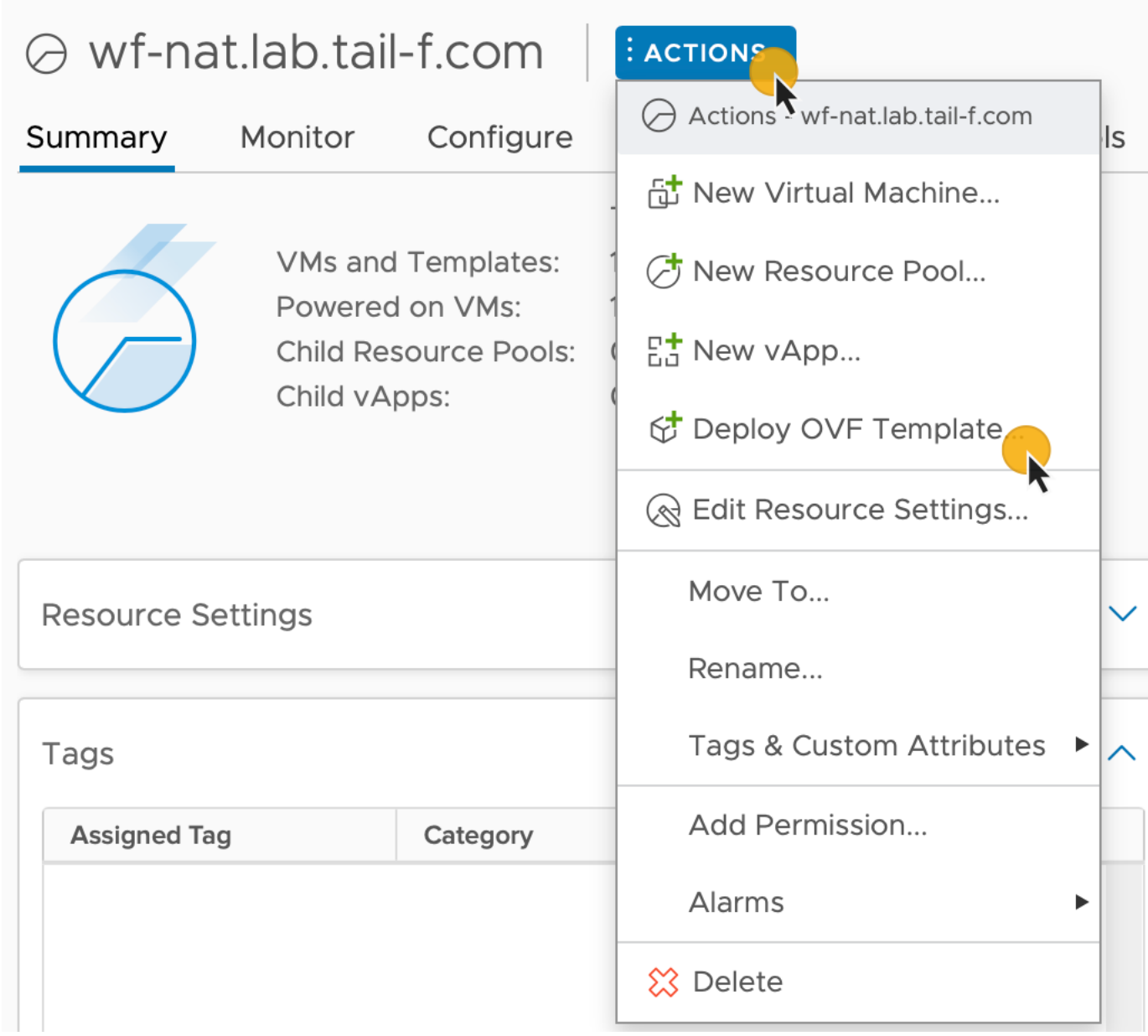
ステップ 2 [ホストおよびクラスタ（Hosts and Clusters）] タブで、ホストを展開してリソースプールを選択します。

図 *1* :



ステップ 3 [アクション（Actions）] メニューをクリックし、[OVF テンプレートの展開（Deploy OVF Template...）] を選択します。

図 *2:*



ステップ 4 [OVF テンプレートの選択（Select an OVF template）] ステップで、[ローカルファイル（Local file）]、[ファイルの選択（Select files）] の順にクリックし、CWM OVA イメージを選択します。[次へ（Next）] をクリックします。

ステップ 5 [名前とフォルダの選択（Select a name and folder）] ステップで、VM の名前を入力して場所を選択します。[次へ（Next）] をクリックします。

ステップ 6 [コンピューティングリソースの選択（Select a compute resource）] ステップで、リソースプールを選択します。[次へ（Next）] をクリックします。

ステップ 7 [詳細の確認（Review details）] ステップで、[次へ（Next）] をクリックします。

ステップ 8 [ストレージの選択（Select storage）] ステップで、[仮想ディスクフォーマットの選択（Select virtual disk format）] を [シンプロビジョニング（Thin provision）] に設定し、ストレージを選択して、[次へ（Next）] をクリックします。

ステップ 9 [ネットワークの選択（Select network）] ステップで、**コントロールプレーン**と**ノースバウンド**の宛先ネットワークを選択する必要があります。

a) [コントロールプレーン（Control Plane）] : [プライベートネットワーク（PrivateNetwork）] を選択します。選択できない場合は、[VMネットワーク（VM Network）] を選択します。

（注） コントロールプレーンの設定は、HA クラスタのセットアップの場合にのみ必要です。単一ノードのセットアップでは、コントロールプレーンの設定を指定する必要がありますが、必須ではなく、制御ネットワークに接続されている他のデバイスと競合しないようにする必要があります。

b) [ノースバウンド（Northbound）] : [VMネットワーク（VM Network）] を選択します。

c) [次へ（Next）] をクリックします。

ステップ 10 [テンプレートのカスタマイズ（Customize template）] ステップで、次の選択されたプロパティを指定します。

a) [インスタンスのホスト名（Instance Hostname）] : インスタンスの名前を入力します。

b) [SSH 公開キー（SSH Public Key）] : VM へのコマンドラインアクセスに使用される **ed25519** SSH 公開キーを指定します。

c) [ノード名（Node Name）] : インストールノードの名前を指定します。

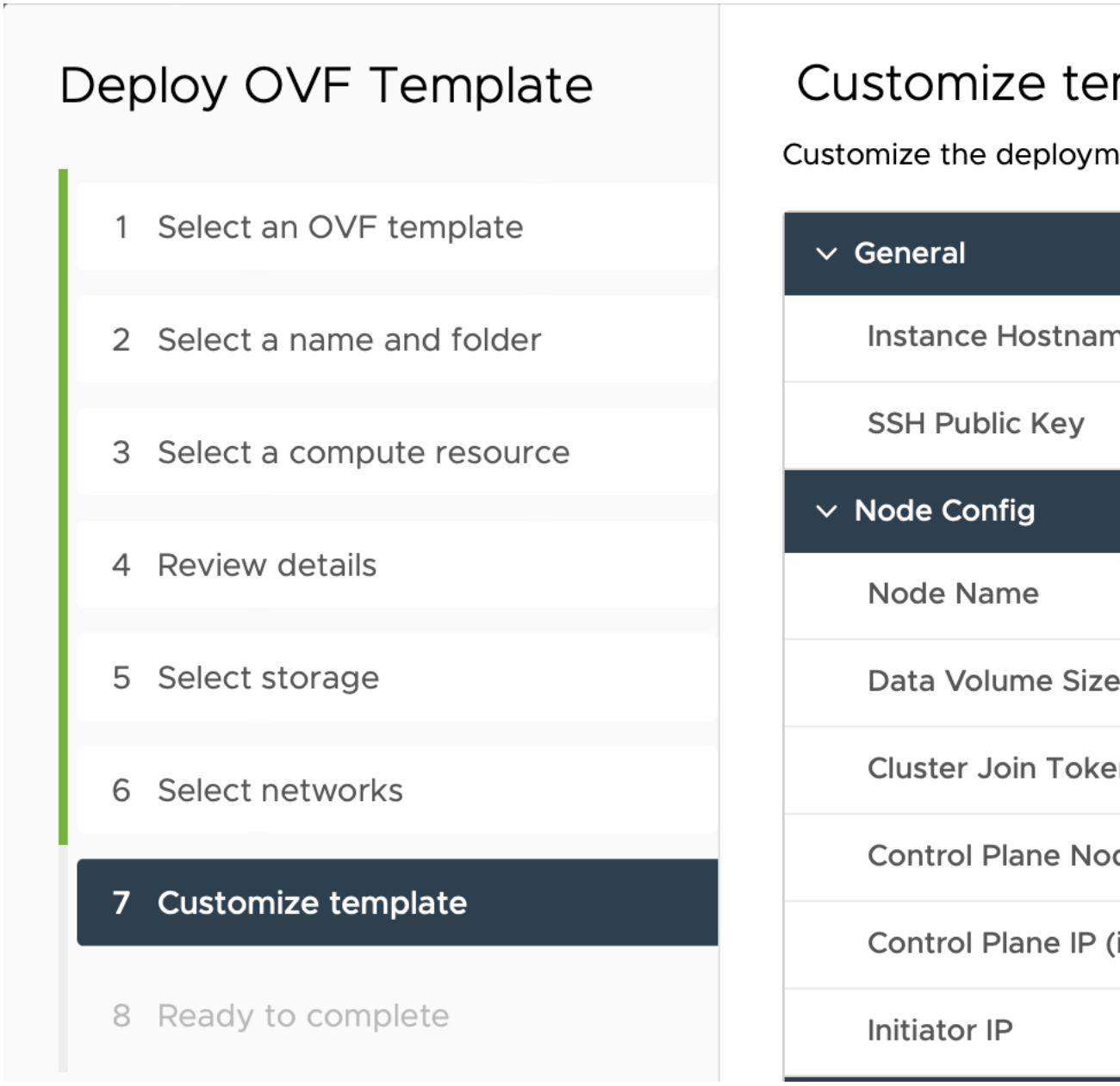
（注） 単一ノード設定の場合、ノード名を変更することは推奨されません。変更する場合は、次の [ゾーン A ノード名（Zone-A Node Name）] と一致している必要があることに注意してください。

d) [コントロールプレーンノード数（Control Plane Node Count）] : HA クラスタセットアップの場合にのみ、1 以上に変更します。CWM バージョン 1.1 ではサポートされていません。

e) [コントロールプレーン IP（IP サブネット）（Control Plane IP (ip subnet)）] : コントロールプレーンのネットワークアドレスを指定します。このアドレスは、制御ネットワーク内の他のデバイスと競合するこはできません。ただし、単一ノードのセットアップでは必須ではありません。デフォルトのサブネットマスクは `/24` であることに注意してください。ネットワーク設定に該当する場合は、カスタムサブネットマスク値を追加できます。

f) [イニシエータ IP（Initiator IP）] : スターターノードのイニシエータ IP を設定します。単一ノードセットアップでは、[コントロールプレーン IP（Control Plane IP）] * と同じアドレスです。

図 *3 :*



g) [IP（IP サブネット）- DHCP を使用していない場合（IP (ip subnet) - if not using DHCP）]：ノードのネットワークアドレスを指定します。デフォルトのサブネットマスクは `/24` であることに注意してください。ネットワーク設定に該当する場合は、カスタムサブネットマスク値を追加できます。

h) [ゲートウェイ - DHCP を使用していない場合（Gateway - if not using DHCP）]：ゲートウェイアドレスを指定します。デフォルトでは、`192.168.1.1` です。

i) [DNS]：DNS のアドレスを指定します。デフォルトでは `8.8.8.8` ですが、ローカル DNS を使用することもできます。

j) [ノースバウンド仮想IP（Northbound Virtual IP）]：アクティブなクラスタノードのネットワークアドレスを指定します。単一ノードのセットアップでも、このアドレスは必須です。このアドレスで HTTP サービスが機能するためです。

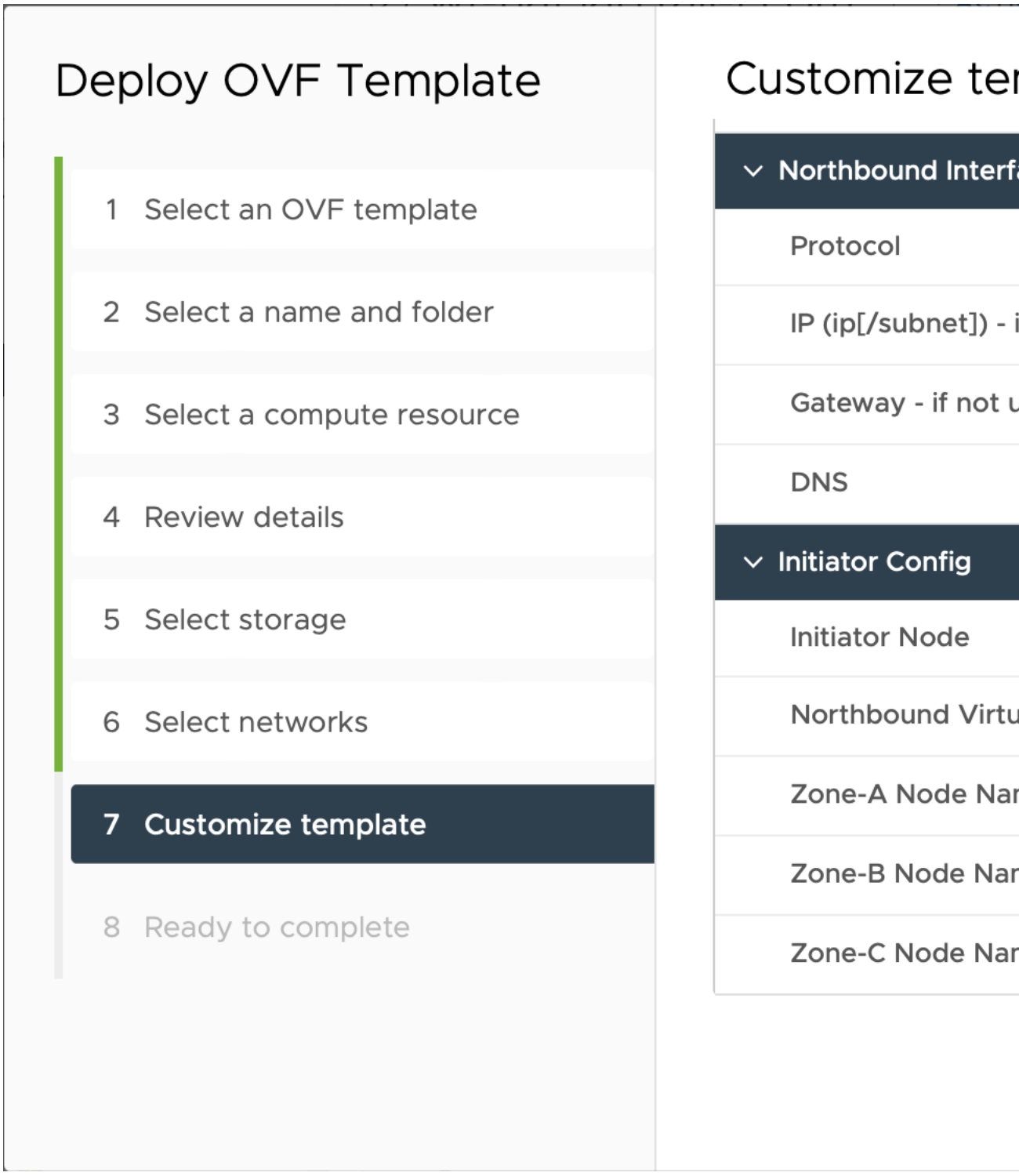
k) [ゾーン A ノード名（Zone-A Node Name）]：ゾーン A ノードの名前を指定します。上記の [ノード名（Node Name）] と一致する必要があることに注意してください。

l) [ゾーン B ノード名（Zone-B Node Name）]：ゾーン B ノードの名前を指定します。単一ノード設定の場合、これは必須ではないため、変更しないでください。

m) [ゾーン C ノード名（アービトレータ）（Zone-C Node Name (Arbitrator)）]：ゾーン C アービトレータノードの名前を指定します。単一ノード設定の場合、これは必須ではないため、変更しないでください。

n) [次へ（Next）] をクリックします。

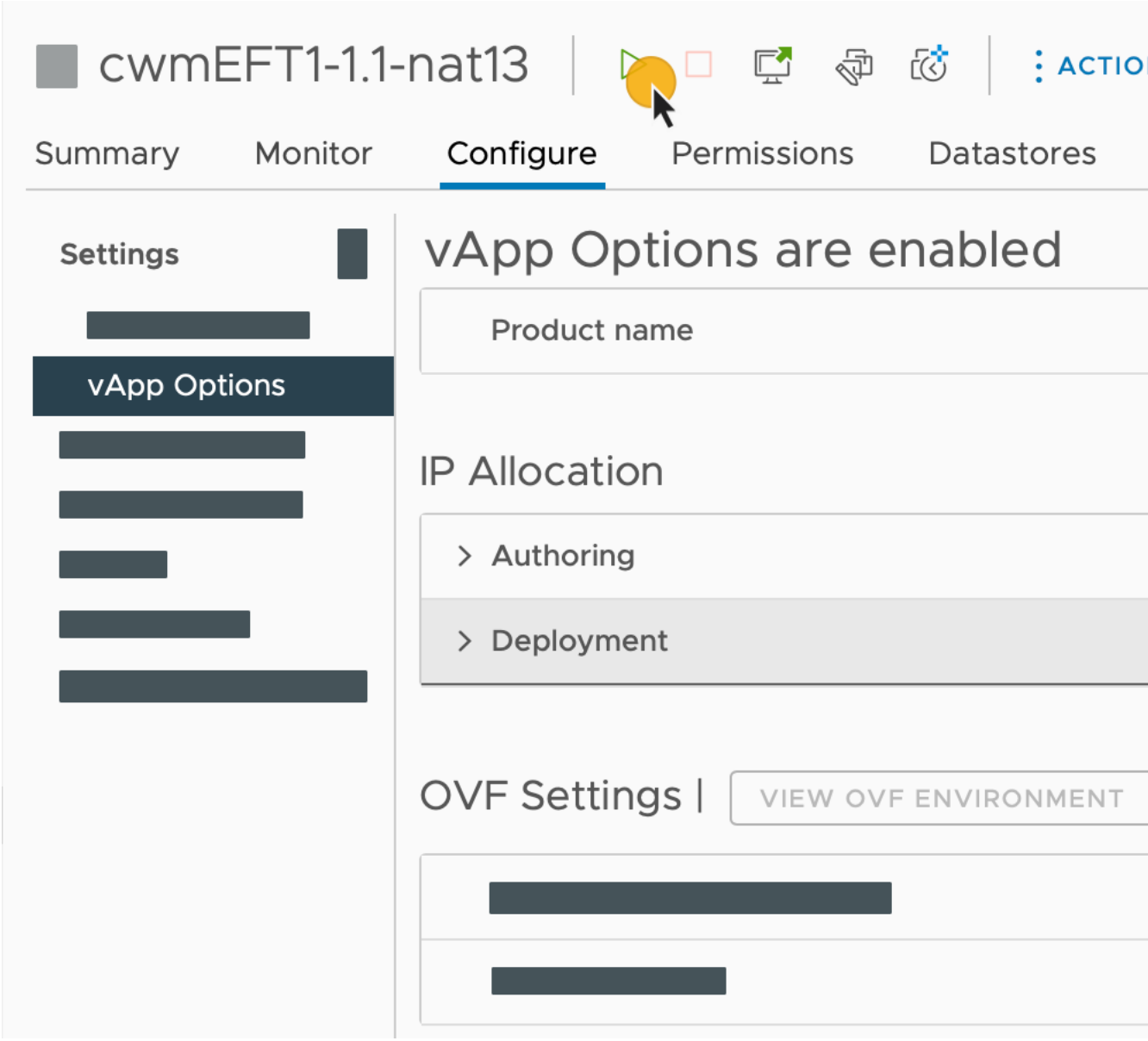
図 *4 :*


Deploy OVF Template
1 Select an OVF template
2 Select a name and folder
3 Select a compute resource
4 Review details
5 Select storage
6 Select networks
7 Customize template
8 Ready to complete
Customize ter
Northbound Interfa
Protocol
IP (ip[/subnet]) - i
Gateway - if not u
DNS
Initiator Config
Initiator Node
Northbound Virtu
Zone-A Node Nar
Zone-B Node Nar
Zone-C Node Nar

ステップ 11 [準備完了（Ready to Complete）] ページで [終了（Finish）] をクリックします。展開には数分かかる場合があります。

ステップ 12 [リソースプール（Resource pool）] リストから、新しく作成した仮想マシンを選択し、[電源オン（Power on）] アイコンをクリックします。

図 *5*:



（注） VM の電源が正常にオンにならない場合は、NxF が原因で断続的なインフラストラクチャ エラーが発生している可能性があります。回避策として、既存の VM を削除し、新しい VM に OVA を再展開します。

インストールの確認とユーザーの作成

CWM UI に初めてログインするためのプラットフォームユーザーアカウントを作成する前に、インストールが正常に完了し、システムが稼働しているかどうかを確認します。

ステップ 1 コマンドラインターミナルを使用して、SSH でゲスト OS の NxF にログインします。

```
ssh -o UserKnownHostsFile=/dev/null  -p 22  nxf@<virtual_IP_address>
```

（注） デフォルトでは、仮想 IP アドレスは [IP（IP サブネット）- DHCP を使用していない場合（IP (ip subnet) - if not using DHCP）] で設定したアドレスです。vCenter のセットアップ方法に応じて、特定のポートとともにリソースプールアドレスを指定できます。不明な点がある場合は、ネットワーク管理者に確認してください。

オプション：初めてログインする場合は、秘密キーのパス名を入力します。

```
ssh -i <ed25519_ssh_private_key_name_and_location> nxf@<virtual_IP_address>
```

（注） SSH のデフォルトポートは 22 です。必要に応じてカスタムポートに変更してください。

ステップ 2 NxF ブートログを確認します。

```
sudo journalctl -u nxf-boot
```

（注） インストールが完了するまで数分かかる場合があることに注意してください。表示される NxF ログの下部で、[NXF：マシンのセットアップ完了（NXF: Done setup up machine）] メッセージを探します。ログで問題がレポートされた場合は、CWM の再インストールを検討してください。

ステップ 3 すべての Kubernetes ポッドが稼働しているかどうかを確認します。

```
kubectl get pods -A
```

これにより、次のようなステータスとともにポッドのリストが表示されます。

```
NAMESPACE             NAME                                       READY   STATUS             RESTARTS
       AGE
kube-flannel          kube-flannel-ds-vh4js                      1/1     Running            0
       7m35s
kube-system           coredns-9mnzv                              1/1     Running            0
       7m35s
kube-system           etcd-node1                                 1/1     Running            0
       7m44s
kube-system           kube-apiserver-node1                       1/1     Running            0
       7m50s
kube-system           kube-controller-manager-node1              1/1     Running            0
       7m50s
kube-system           kube-proxy-6hwg9                           1/1     Running            0
       7m35s
kube-system           kube-scheduler-node1                       1/1     Running            0
       7m42s
local-path-storage    local-path-provisioner-54c455f95-mbhc9     1/1     Running            0
       7m34s
nxf-system            authenticator-f74c7c87f-m8p4x              2/2     Running            0
       6m25s
nxf-system            controller-76686f8f5f-gpqvc                2/2     Running            0
       6m27s
nxf-system            ingress-ports-node1-zchwz                  1/1     Running            0
```

```
       4m17s
nxf-system           ingress-proxy-bcb8c9fff-lzm9p            1/1     Running            0
       6m23s
nxf-system           kafka-0                                  1/1     Running            0
       7m34s
nxf-system           loki-0                                   3/3     Running            0
       6m33s
nxf-system           metrics-5qnzb                            2/2     Running            0
       6m30s
nxf-system           minio-0                                  2/2     Running            0
       7m34s
nxf-system           postgres-0                               2/2     Running            0
       6m59s
nxf-system           promtail-t7dp4                           1/1     Running            0
       6m33s
nxf-system           registry-5486f46b54-c6tf9                2/2     Running            0
       7m2s
nxf-system           vip-node1                                1/1     Running            0
       6m12s
zone-a               cwm-api-service-67bd9db5c7-vfszs         2/2     Running            2 (3m37s
ago)   4m16s
zone-a               cwm-dsl-service-7ffd6975ff-wlrwt         2/2     Running            4 (3m21s
ago)   4m15s
zone-a               cwm-engine-frontend-6754445fc-67t5h      2/2     Running            2 (3m52s
ago)   4m15s
zone-a               cwm-engine-history-c4dfffddd-t2fgv       2/2     Running            1 (2m35s
ago)   4m14s
zone-a               cwm-engine-history-c4dfffddd-wr5v2       2/2     Running            2 (3m51s
ago)   4m14s
zone-a              cwm-engine-history-c4dfffddd-zz74q       2/2     Running           4 (48s ago)
     4m14s
zone-a               cwm-engine-matching-78dfdf858f-q8wg2     2/2     Running            2 (3m46s
ago)   4m14s
zone-a               cwm-engine-ui-6b74755499-jwbld           2/2     Running            0
       4m13s
zone-a               cwm-engine-worker-589b6bc88b-hs2ch       2/2     Running            0
       4m13s
zone-a               cwm-event-manager-5b95bb49db-gw6g5       2/2     Running            0
       4m12s
zone-a               cwm-plugin-manager-76f798446c-qgx27      2/2     Running            1 (2m29s
ago)   4m12s
zone-a               cwm-ui-779bdb44-98d5v                    2/2     Running            0
       4m11s
zone-a               cwm-worker-manager-7bd8795b56-f4czp      2/2     Running            1 (112s
ago)    4m10s
zone-a               logcli-5f8cc8c585-fq7wm                  2/2     Running            0
       4m10s
```

（注） システムがすべてのポッドを実行するまでに数分かかる場合があることに注意してください。いずれかのポッドのステータスが `Running` 以外の場合は、`kubectl delete pod <pod_name> -n <namespace>` コマンドを使用して再起動することを検討してください。

UI ログイン用のユーザーの作成

VM へのコマンドラインアクセスを使用して、CWM プラットフォームのユーザーアカウントを作成できます。その方法を次に説明します。

ステップ 1 コマンドラインターミナルを使用して、SSH でゲスト OS の NxF にログインします。

```
ssh -o UserKnownHostsFile=/dev/null  -p 22  nxf@<virtual_IP_address>
```

オプション：初めてログインする場合は、秘密キーのパス名を入力します。

```
ssh -i <ed25519_ssh_private_key_name_and_location> nxf@<virtual_IP_address>
```

（注） SSH のデフォルトポートは 22 です。必要に応じてカスタムポートに変更してください。

ステップ 2 ユーザーとパスワードを作成するには、次のコマンドを実行します。

a) まず、パスワードの最小複雑度を設定します（デフォルトは 3で、0 は複雑度が無効です）。

```
sedo security password-policy set --min-complexity-score 1
```

b) 次に、ユーザーアカウントとパスワードを作成します。

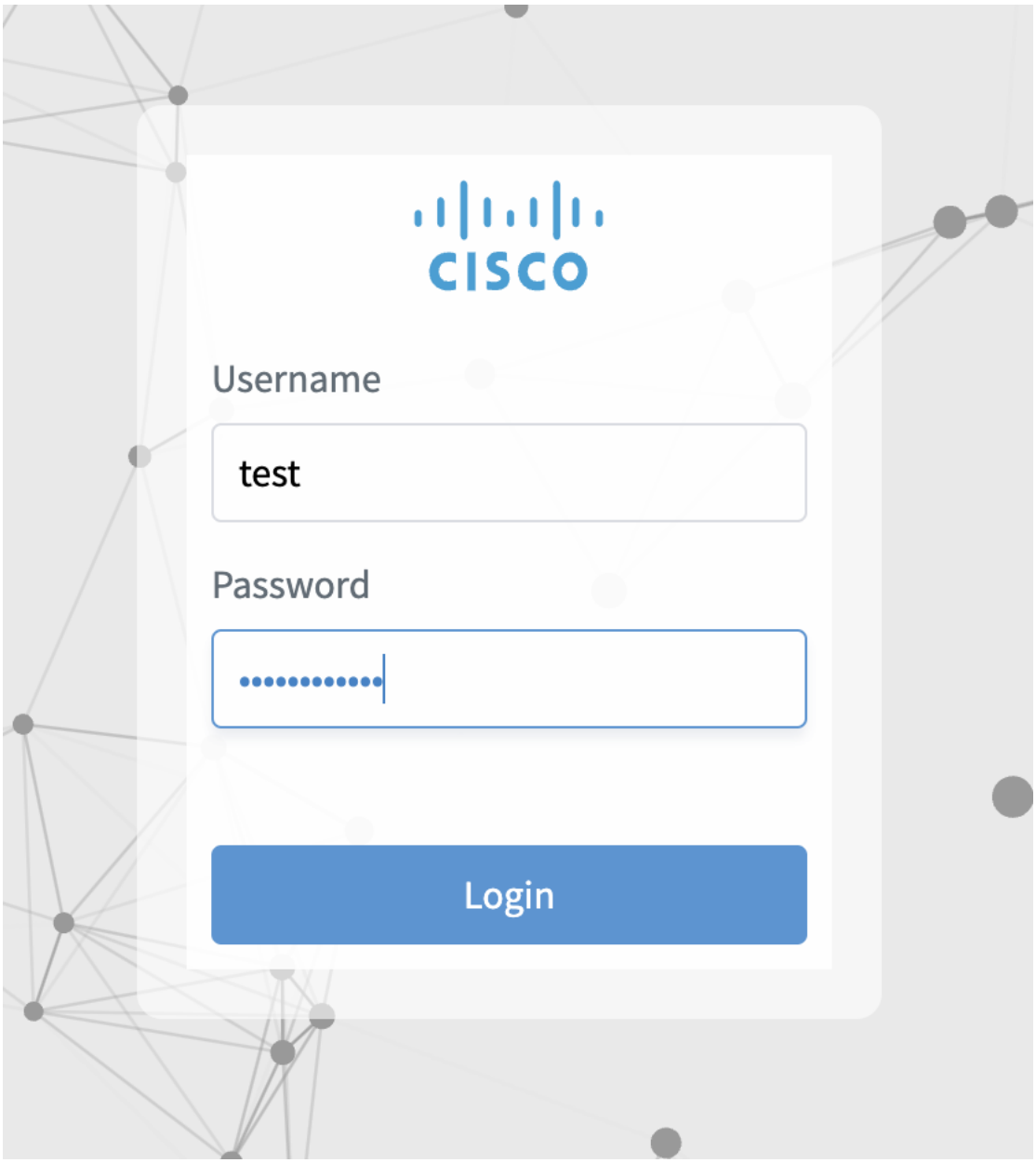
```
echo -en 'Password123!' | sedo security user add --password-stdin \
--access permission/admin --access permission/super-admin \
--access permission/user --display-name Tester test
```

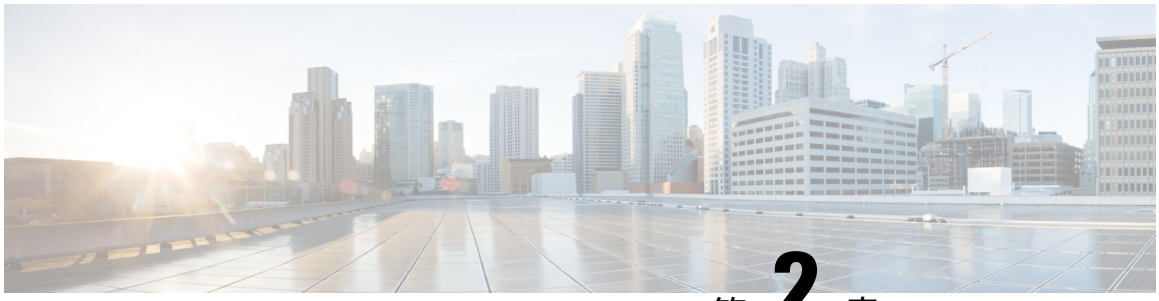
c) 必要に応じて、テストユーザーのパスワード変更要件を無効にします。

```
sedo security user set test  --must-change-password=false
```

ステップ 3 CWM UI を表示するには、ノースバウンド IP とデフォルトのポート 8443 用に選択したアドレスに移動します。例：https://192.168.1.233:8443/

ステップ 4 test ユーザー名とパスワードを使用してログインします。


CISCO
Username
test
Password
Login



第 2 章

System

ここでは、次の内容について説明します。

アーキテクチャの概要

Crosswork Workflow Manager アーキテクチャは、Kubernetes コンテナ オーケストレーション システム上で動作するマイクロサービスベースのソリューションです。このセクションでは、コア アーキテクチャ コンポーネントを示す図と、それぞれの簡単な説明を示します。

- **ユーザーインターフェイス（UI）**：オペレータは、ワークフローの追加とインスタンス化、ワークフローデータの入力、実行中のワークフローの一覧表示、ジョブの進行状況の監視を行うことができます。UI の [管理（Admin）] セクションでは、ワーカーの追加、ワーカープロセスの管理、およびアダプタからワーカーへのアクティビティの割り当てを行うことができます。
- **REST API**：CWM アプリケーションとのすべての連携（アダプタの展開、ワークフローの公開とインスタンス化、ワーカー、リソース、およびシークレットの管理）が含まれます。
- **制御サーバー**：関連するマイクロサービスに API 要求をディスパッチします。
- **ワークフローエンジン**：ワークフローの処理方法を制御するコアコンポーネントです。ワークフロー定義の実行を解釈および管理します。
- **実行エンジン（ワークフローワーカー）**：ワークフロータスクの実行を担当します。**ワークフローエンジン**からワークフロータスクを受信し、正しい順序で実行し、結果を**ワークフローエンジン**に返します。
- **アダプタワーカー**：ワークフロー定義とアダプタコードで定義されたタスクの実行を担うプロセスです。**ワークフローワーカー**からタスクを受信して実行し、結果を**ワークフローワーカー**に送り返します。実行ワーカーは、追加のアダプタをプラグインとしてロードできるため、さまざまなシステムやテクノロジーと連携できます。
- **アダプタ**：外部システム、アプリケーション、およびテクノロジーとのインターフェイスとなり、これらと統合します。アダプタ内部では、ワークフローで使われるアクティビティが定義されます。
- **アダプタ SDK**：外部システムと統合するための新しいアダプタを作成する開発者を支援するソフトウェア開発キット。
- **ワークフロー定義**：サーバレスワークフロー仕様に基づいて JSON 形式で記述されたワークフローコード。
- **K8s インフラストラクチャ**：CWM アプリケーション用のランタイムプラットフォーム。これは、Kubernetes クラスタ内のアプリケーションの展開と管理をサポートするために必要なインフラストラクチャを提供するサービスの集合です。
- **PostgreSQL**：データを保存および管理するためにシステムで使用されるデータベースです。

正常性とログの確認

CWM は、Kubernetes クラスタアーキテクチャをランタイム環境として活用するマイクロサービスベースのアプリケーションです。したがって、Kubernetes コマンドを使用して CWM アプリケーションの正常性を確認できます。

（注） サポートされているすべての kubectl コマンドを表示するには、VM の OS にログインし、kubectl --help を使用します。

ポッドステータスの確認

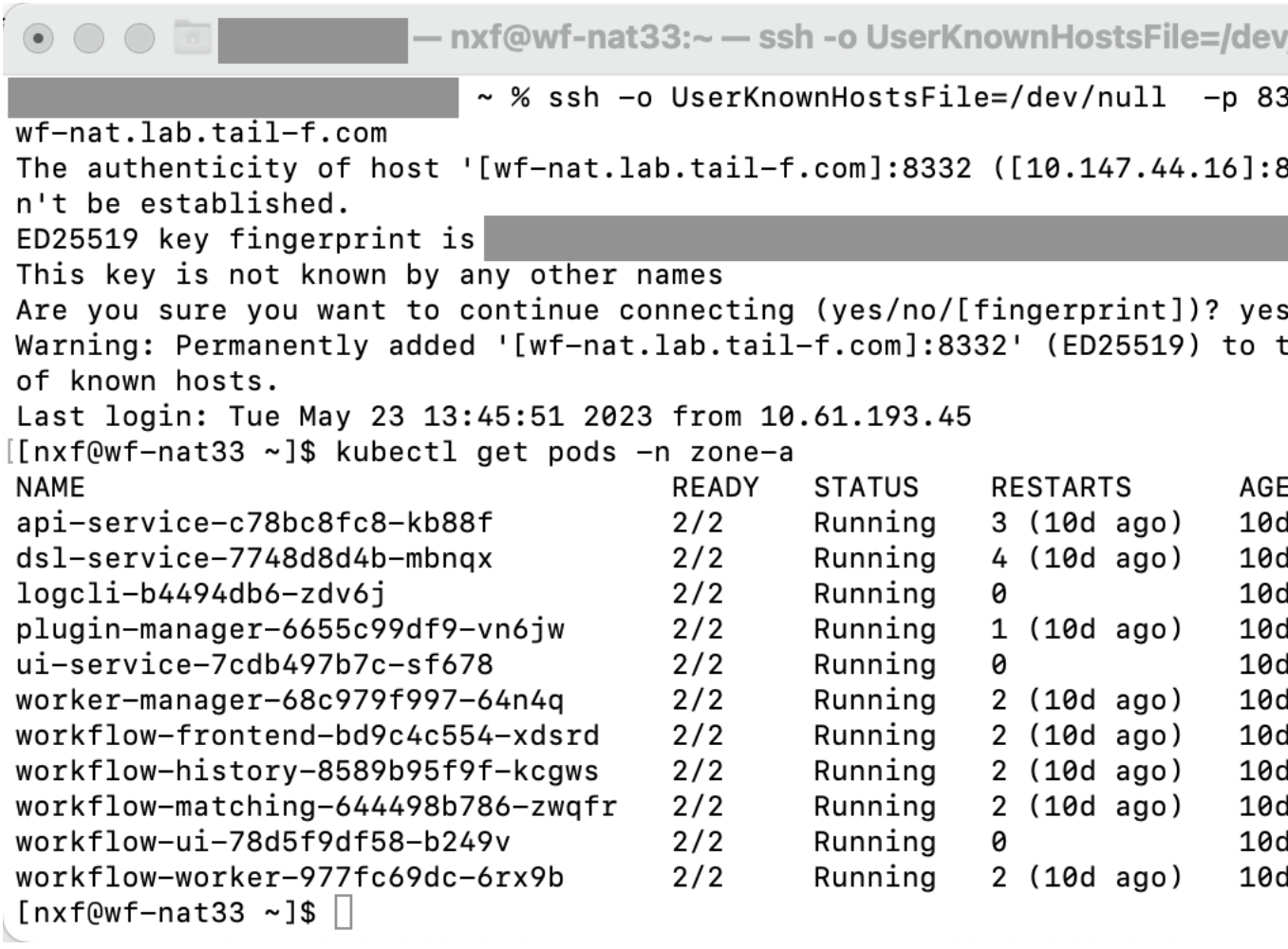
ステップ 1 コマンドラインターミナルを使用して、SSH で仮想マシンの OS にログインします。

```
ssh -o UserKnownHostsFile=/dev/null  -p 22  nxf@<your_resource_pool_address>
```

ステップ 2 名前空間 zone-a（これは、CWM マイクロサービスを含むポッドのデフォルトの名前空間です）のポッドのステータスを確認するには、次のコマンドを実行します。

```
kubectl get pods -n zone-a
```

ステップ 3 ポッドのリストが表示されます。



ステップ 4 ポッドのステータスが `Running` 以外の場合は、次のコマンドを使用してポッドを「再起動」できます。

```
kubectl delete pod <pod_name> -n zone-a
```

ポッドは削除されますが、Kubernetes 設定は宣言型であるため、削除されたポッドは効果的に再作成されて再実行します。

ログの確認と収集

アプリケーションログは、**Loki logCLI** コマンドライン インターフェイスで確認できます。CWM プラットフォームからログを収集するには、次の手順を実行します。

ステップ 1 コマンドラインターミナルを使用し、SSH クライアントを使用してシステムに接続します。

```
ssh  -pSSH_PORT nxf@ip_address_of_deployment
```

（注） `SSH_PORT` と `ip_address_of_deployment` を適宜調整します。

ステップ 2 ログインに成功したら、次のコマンドを使用して、実行中のすべてのポッドを一覧表示します。

```
kubectl get pods -A
```

結果の例：

```
[nxf@wf-nat-08 ~]$ kubectl get pods -A
NAMESPACE            NAME                                      READY   STATUS      RESTARTS
AGE
kube-flannel         kube-flannel-ds-trr95                     1/1     Running     0
103m
kube-system          coredns-htg9j                             1/1     Running     0
103m
kube-system          etcd-wf-nat-08                            1/1     Running     0
103m
kube-system          kube-apiserver-wf-nat-08                  1/1     Running     0
103m
kube-system          kube-controller-manager-wf-nat-08         1/1     Running     0
103m
kube-system          kube-proxy-c25f5                          1/1     Running     0
103m
kube-system          kube-scheduler-wf-nat-08                  1/1     Running     0
103m
local-path-storage   local-path-provisioner-6fb6f599c7-ckcjc   1/1     Running     0
103m
nxf-system           authenticator-5db8885675-qlrmg            2/2     Running     0
102m
nxf-system           controller-cbd87f8c5-6tg6f                2/2     Running     1 (102m ago)
102m
nxf-system           ingress-proxy-56f7c9899d-6st6j            1/1     Running     0
102m
nxf-system           kafka-0                                   1/1     Running     0
102m
nxf-system           loki-7c994678f8-fnrs9                     3/3     Running     0
102m
nxf-system           minio-0                                   2/2     Running     0
103m
nxf-system           postgres-0                                2/2     Running     0
102m
```

```
nxf-system           promtail-v6tb4                          1/1     Running     0
102m
nxf-system           registry-7dd84db44f-n5q7h               2/2     Running     0
102m
nxf-system           vip-wf-nat-08-28131000-772k5            0/1     Completed   0
3m42s
zone-a               api-service-745759bffc-v6r25            2/2     Running     2 (100m ago)
100m
zone-a               dsl-service-77d5fc96cc-5nv42            2/2     Running     3 (100m ago)
100m
zone-a               logcli-5c7ddbc95d-mkpcc                 2/2     Running     0
100m
zone-a               plugin-manager-665b7bbd4d-jvqdk         2/2     Running     1 (100m ago)
100m
zone-a               ui-service-57cf6d6bcc-smmvt             2/2     Running     0
100m
zone-a               worker-manager-6d6b445d46-r6nzk         2/2     Running     1 (99m ago)
100m
zone-a               workflow-frontend-77bc897549-kcz5k      2/2     Running     1 (99m ago)
100m
zone-a               workflow-history-58bdb85b8d-88t25       2/2     Running     1 (99m ago)
100m
zone-a               workflow-history-58bdb85b8d-h22bd       2/2     Running     1 (99m ago)
100m
zone-a               workflow-history-58bdb85b8d-ph5fh       2/2     Running     1 (99m ago)
100m
zone-a               workflow-matching-86cfc5577c-4mxhb      2/2     Running     1 (99m ago)
100m
zone-a               workflow-ui-68f857645-9mq9v             2/2     Running     0
100m
zone-a               workflow-worker-8496898f7b-wcrqs        2/2     Running     1 (99m ago)
100m
```

ステップ 3 `zone-a` 名前空間で使用可能な logcli ツールを特定します。この例では、`logcli-5c7ddbc95d-mkpcc` という名前のポッドです。

ステップ 4 正しいポッドに接続し、フィルタを適用可能なログラベルを一覧表示します。

```
kubectl exec --namespace=zone-a -ti logcli-5c7ddbc95d-mkpcc -- logcli labels
app
container
filename
level
namespace
node_name
pod
stream
```

ステップ 5 "zone-a" 名前空間で実行されているすべてのアプリケーションからログを収集し、単一のファイルに保存します。トラブルシューティング イベントが発生したときに関連する期間からログを収集するように、`-since` オプションを調整してください。

```
kubectl exec --namespace=zone-a -ti logcli-5c7ddbc95d-mkpcc -- logcli query '{namespace="zone-a"}'
 --since 60m > zone-a.log
```

ステップ 6 同様に、便宜上別のファイルを使用して、他の名前空間からログを収集します。

```
kubectl exec --namespace=zone-a -ti logcli-5c7ddbc95d-mkpcc -- logcli query '{namespace="nxf-system"}'
 --since 60m > nxf-system.log
```

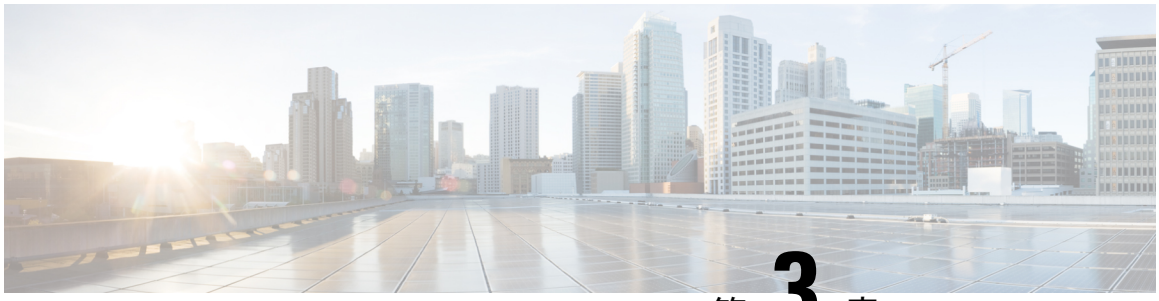
```
kubectl exec --namespace=zone-a -ti logcli-5c7ddbc95d-mkpcc -- logcli query '{namespace="kube-system"}'
 --since 60m > kube-system.log
```

ステップ 7 SCP ツールを使用して、システムからデスクトップにログファイルをコピーします。

```
scp -P SSH_PORT nxf@ip_address_of_deployment:"*.log".
```

ステップ 8 最後に、ログをサポートに送信し、発生している問題の詳細な説明を提供できます。

（注） logCLI コマンドと使用方法の詳細については、logCLI Grafana のドキュメントを参照してください。



第 3 章

API

ここでは、次の内容について説明します。

CWM API の概要

CWM API は、Representational State Transfer（REST）の設計原則に従って開発されました。API には、JSON データ形式を使用した HTTP を使用してアクセスします。関連する HTTP 応答コードで、要求の処理の成否が示されます。データ取得メソッドには GET 要求が必要です。一方、データを追加、変更、または削除するメソッドには POST、PUT、PATCH、または DELETE メソッドが必要です。要求が誤ったリクエストタイプで送信されると、エラーが返されます。

CWM API の使用方法

CWM API は次の 2 つの方法で使用できます。

- Swagger インターフェイスを介して、または
- Postman コレクションを介して

CWM UI からアクセスする Swagger インターフェイスが製品に直接組み込まれていますが、使いやすさを考慮して、サンプル要求を含む Postman コレクションも提供されています。

[API を介した管理（Manage via API ）] セクションのすべてのチュートリアルでは、Swagger の使用を前提としています。

Swagger の使用

CWM Swagger API にアクセスするには、左側のナビゲーションメニューから、[swagger] アイコンをクリックします。

CWM Postman コレクションの使用

前提条件

- Postman Web アプリケーションアカウントまたは Postman デスクトップがインストールされている。

JSON コレクションファイルのダウンロード

このリンクをクリックして、JSON 形式の Postman コレクションをダウンロードします。zip アーカイブを解凍します。

コレクションのインポートと環境の設定

ステップ 1 Postman を開き、[コレクション（Collections）] に移動します。

ステップ 2 [インポート（Import）] をクリックし、[インポートするには任意の場所にドロップ（DropAnywhere to Import）] 画面からフォルダを選択し、zip アーカイブから解凍したフォルダをポイントします。

ステップ 3 [環境（Environment）] に移動し、新しくインポートした**テスト**環境を選択します。

ステップ 4 CWM の IP アドレスとポートに合わせて **baseUrl** 変数と **port** 変数の現在の値を指定し、変更を保存します。

これで設定が完了し、コレクションを使用する準備が整いました。

アダプタの管理

外部ターゲットシステムと連携するには、CWM にアダプタが必要です。アダプタは、**オペレータ**ガイドで説明されているように CWM UI で、または CWM API を使用して管理できます。アダプタの処理には、次の API エンドポイントを使用できます。

- `GET/adapter`：CWM アプリケーションに存在するアダプタのリストを取得します。
- `POST/adapter`：アダプタの **.tar** ファイルを CWM ストレージにアップロードします。
- `GET/adapter/{adapterId}`：CWM アプリケーションに存在する特定のアダプタの詳細を取得します。これには、アダプタで使用可能なすべてのアクティビティの一覧表示が含まれます。

- `PUT/adapter/{adapterId}`：既存のアダプタファイルを新しいアダプタバージョンで更新します。
- `DELETE/adapter/{adapterId}`：CWM アプリケーションからアダプタを削除します。
- `POST/adapter/{adapterId}/deploy`：アップロードされたアダプタファイルに基づいてシステムにアダプタを展開します。
- `PATCH/adapter/{adapterId}`：アダプタのデフォルトステータスを更新します。

アダプタのインストール

CWM アダプタは、**.tar** インストールファイルに含まれています。ワークフローで使用するには、事前にストレージにアップロードしてシステムに展開する必要があります。その方法について説明します。

アダプタファイルのアップロード

アダプタを展開する前に、アダプタの **.tar** ファイルを CWM ストレージにアップロードする必要があります。

ステップ 1 最新のアダプタ インストール ファイルを取得するか、独自のアダプタを作成します。

ステップ 2 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 3 [adapters] セクションで、`POST/adapter` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 4 表示されるサブセクションで、[ファイルの選択（Choose File）] をクリックし、アダプタの **.tar** インストールファイルを選択して [アップロード（Upload）] をクリックし、[実行（Execute）] をクリックします。

サーバーの応答コードが `201` の場合、アダプタファイルは CWM データベースに正常にアップロードされています。

アダプタの展開

ステップ 1 CWM API の [アダプタ（adapters）] セクションで、`GET/adapter` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] と[実行（Execute）] をクリックします。

ステップ 2 サーバーの応答本文から、アップロードしたアダプタの `id` フィールドの値をコピーします。

ステップ 3 CWM API の [アダプタ（adapters）] セクションで、`POST/adapter/{adapterId}/deploy` エンドポイントをクリックして展開します。

ステップ 4 エンドポイント内で、[試す（Try it out）] をクリックします。[アダプタID（Adapter ID）] フィールドにアダプタ ID を貼り付けます。

ステップ 5 [createWorker] フィールドで、`createWorker` パラメータを `true` に設定できます。アダプタ ID と同じ名前のワーカーが作成されます。

ステップ 6 [実行（Execute）] をクリックします。

サーバーの応答コードが `201` の場合、アダプタプラグインは正常にインストールされており、問題なく続行できます。

アダプタの削除

アダプタをストレージから完全に削除してアンインストールするには、次の手順を実行します。

ステップ 1 CWM API の [アダプタ（adapters）] セクションで、`GET/adapter` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] と[実行（Execute）] をクリックします。

ステップ 2 サーバーの応答本文から、アップロードしたアダプタの `id` フィールドの値をコピーします。

ステップ 3 CWM API の [アダプタ（adapters）] セクションで、`DELETE/adapter/{adapterId}` エンドポイントをクリックして展開します。

ステップ 4 エンドポイント内で、[試す（Try it out）] をクリックします。[アダプタID（Adapter ID）] フィールドにアダプタ ID を貼り付けます。

ステップ 5 [実行（Execute）] をクリックします。

ワーカーの管理

ワーカーは、ワークフロー定義とアダプタコードで定義されたアクションを実行するプロセスです。**オペレータ**ガイドで説明されているように CWM UI を使用してワーカーを管理できます。または以下で説明しているように CWM API を使用して管理できます。

ワーカーを管理するための次のアクションを使用できます。

- `GET/worker`：CWM アプリケーションに存在するワーカーのリストを取得します。
- `POST/worker`：CWM アプリケーション内に新しいワーカーを作成します。
- `GET/worker/{workerName}`：CWM アプリケーションに存在する特定のワーカーの詳細を取得します。
- `PUT/worker/{workerName}`：既存のワーカーを新しいパラメータ値で更新します。
- `DELETE/worker/{workerName}`：CWM アプリケーションからワーカーを削除します。
- `POST/worker/{workerName}/start`：アプリケーションで作成されたワーカーをアクティブにします。
- `POST/worker/{workerName}/stop`：アプリケーションで作成されたワーカーを非アクティブ化します。

ワーカーの作成

ステップ 1 CWM にログインし、左側のナビゲーションメニューから [Swagger] アイコンをクリックします。

ステップ 2 CWM API の [ワーカー（workers）] セクションで、`POST/worker` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 3 [ワーカーデータ（Worker data）] フィールドに、必要な値を入力します。

a) [activities] : 展開されたアダプタの ID または特定のアダプタアクティビティを貼り付けます。

b) [startWorker] : `true` に設定します。

c) [workerName] : ワーカーの名前を指定します。

ステップ 4 [実行（Execute）] をクリックします。

ワーカーの開始

ステップ 1 CWM API の [ワーカー（workers）] セクションで、`POST/{workerName}/start` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 2 次の**パラメータ**のフィールドに、必要な値を入力します。

a) [開始するワーカーの名前（Name of a worker to start）] : 開始するワーカーの名前を貼り付けます。

b) [forceReload]: ワーカーを強制的に起動する場合は `true` に設定します。

ステップ 3 [実行（Execute）] をクリックします。

ワーカーの停止

ステップ 1 CWM API の [ワーカー（workers）] セクションで、`POST/{workerName}/stop` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 2 次の**パラメータ**のフィールドに、必要な値を入力します。

a) [停止するワーカーの名前（Name of a worker to stop）] : 停止するワーカーの名前を貼り付けます。

b) [forceStop]: ワーカーを強制的に停止する場合は `true` に設定します。

ステップ 3 [実行（Execute）] をクリックします。

ワークフローの管理

ワークフロー定義は、**オペレータ**ガイドで説明されているように CWM UI で、または CWM API を使用して管理できます。

- `GET/workflow`：CWM アプリケーションに存在するワークフロー定義のリストを取得します。
- `POST/workflow`：CWM アプリケーション内に新しいワークフロー定義を作成します。
- `GET/workflow/{workflowId}`：CWM アプリケーションに存在する特定のワークフローの詳細を取得します。
- `PUT/workflow/{workflowId}`：CWM アプリケーション内に存在するワークフロー定義を更新します。
- `DELETE/workflow/{workflowId}`：選択したワークフロー定義を CWM アプリケーションから削除します。
- `GET/workflowExport`：指定されたワークフロー定義 ID の配列に基づいてワークフロー定義をエクスポートします。
- `POST/workflowImport`：CWM アプリケーションにワークフロー定義を一括でインポートします。

（注） ワークフローを管理するには、CWM UI を使用することが推奨されます。詳細については、**オペレータ**ガイドを参照してください。

リソースとシークレットの管理

概要

CWM では、アダプタは、他のシステムやアプリケーションなどの外部エンティティでアクションを実行できるようにするアクティビティを定義します。これらのエンティティは、ほとんどの場合、通常は接続と認証データを必要とする API を介して統合されます。CWM は、アクティビティがワークフローで使用されるときに、接続エンドポイントの詳細と認証データを実行時に渡すことができるフレームワークを提供します。したがって、ワークフローを実行するオペレータは、IP アドレス、ポート、ユーザー名、パスワードなど、これらのシステム（リソース）の詳細を知らない場合があります。

CWM は、データベース内のリソースとシークレットを安全に処理し、それぞれの ID でそれらを識別するためのフレームワークを提供します。ワークフローを実行する場合は、リソース ID のみを渡す必要があり、残りのデータはリソースマネージャによってアダプタに送信され

ます。オペレータの介入やアダプタ開発者の追加開発は必要ありません。シークレットとリソースは、**オペレータ**ガイドで説明されているようにCWM UIで、またはCWM APIを使用して管理できます。

リソースおよびシークレットのタイプ

リソースおよびシークレットのタイプは、ユーザーが作成したリソースとシークレットをタイプ別に整理するために使用される入れ物と考えることができます。タイプは特定のアダプタ内で定義され、アダプタのインストール時に自動的にシステムに追加されます。
`GET/secretType/{secretTypeId}` APIエンドポイントを使用して、特定のタイプに属するシークレットを一覧表示できます。

シークレット API エンドポイント

シークレットを管理するための次のアクションを使用できます。

- `GET/secret`：CWMアプリケーションに存在するシークレットのリストを取得します。
- `POST/secret`：CWMアプリケーション内に新しいシークレットを作成します。
- `GET/secretType/{secretTypeId}`：CWMアプリケーションに存在する、特定のタイプに属するシークレットを一覧表示します。
- `GET/secretType`：CWMアプリケーションに存在するシークレットのタイプのリストを取得します。
- `GET/secret/{secretId}`：既存のシークレットの詳細を取得します。
- `DELETE/secret/{secretId}`：CWMアプリケーションからシークレットを削除します。
- `PATCH/secret/{secretId}`：CWMアプリケーションに存在するシークレットを新しいパラメータ値で更新します。

リソース API エンドポイント

リソースを管理するための次のアクションを使用できます。

- `GET/resource`：CWMアプリケーションに存在するリソースのリストを取得します。
- `POST/resource`：CWMアプリケーションに新しいリソースを作成します。
- `GET/resource/{resourceId}`：CWMアプリケーションに存在する特定のリソースの詳細を取得します。
- `PATCH/resource/{resourceId}`：既存のリソースを新しいパラメータ値で更新します。
- `DELETE/resource/{resourceId}`：CWMアプリケーションからリソースを削除します。
- `GET/resourceType`：CWMアプリケーションに存在するリソースタイプのリストを取得します。
- `GET/resourceType/{resourceTypeId}`：既存のリソースの種類の詳細を取得します。

シークレットの作成

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [シークレット（secrets）] セクションで、[POST /secret] エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、[シークレット入力（Secret input）] フィールドにデータを入力します。入力の例を次に示します。

```
{
"secret": {
"username": "admin",
"password": "admin"
},
"secretId": "NSOSecret",
"secretType": "basicAuth"
}
```

ステップ 4 [実行（Execute）] をクリックします。

サーバーの応答コードが 201 の場合、シークレットは正常に作成されており、シークレットを関連付けるリソースの作成を開始できます。

リソースの作成

ステップ 1 [CWM APIリソース（CWM API resources）] セクションで、[POST /resource] エンドポイントをクリックして展開します。

ステップ 2 エンドポイント内で、[試す（Try it out）] をクリックし、[リソース入力（Resource input）] フィールドにデータを入力します。入力の例を次に示します。

```
{
    "resource": {
        "scheme": "http",
        "host": "127.0.0.1",
        "port": 8080
    },
    "resourceId": "NSOLocal",
    "resourceType": "cisco.nso.resource.v1.0.0",
    "secretId": "NSOSecret"
}
```

ステップ 3 [実行（Execute）] をクリックします。

サーバー応答コードが 201 の場合、リソースは正常に作成されています。

スケジュールの管理

繰り返し発生する操作を自動化したり、将来の特定の日時に確定したりするために、スケジュールを作成できます。CWM には、次の 2 種類のスケジュールがあります。

- ワンタイム：単一のジョブを実行する日時を定義します。
- 繰り返し：ジョブの実行が繰り返されるときのルール（間隔、カレンダー、または cron 式に基づいて）を定義します。

CWM スケジューラ API を使用すると、スケジュールを作成、更新、一時停止/一時停止を解除することができます。{{ version.CWM }} で基本スケジュールを（スケジュールの削除などの他の操作とともに）作成することは UI を介して可能ですが、スケジューラのより多くの機能を使用して高度なスケジュールを定義することは API を介してのみ可能です。

スケジュールの処理には、次の API エンドポイントを使用できます。

- `GET/schedule`：CWM アプリケーションに存在するスケジュールのリストを取得します。
- `POST/schedule`：CWM アプリケーション内に新しいスケジュールを作成します。
- `GET/schedule/{scheduleId}`：CWM アプリケーションに存在する特定のスケジュールの詳細を取得します。
- `PATCH/schedule/{scheduleId}`：既存のスケジュールを新しい詳細で更新します。
- `DELETE/schedule/{scheduleId}`：CWM アプリケーションからスケジュールを削除します。

スケジュールの作成

スケジュールを作成するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [スケジューラ（scheduler）] セクションで、`POST/schedule` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、[スケジュールリクエスト（Schedule request）] で選択した値を指定します。

ステップ 4 [実行（Execute）] をクリックします。サーバー応答コードが `201` の場合、スケジュールは正常に作成されています。

スケジュールの更新

スケジュールを編集するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [スケジューラ（scheduler）] セクションで、`PATCH/schedule` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、[スケジュール ID（Schedule ID）] を指定し、[スケジュール更新リクエスト（Schedule update request）] で選択した値を編集します。

ステップ 4 [実行（Execute）] をクリックします。サーバー応答コードが `200` の場合、スケジュールは正常に更新されています。

スケジュールを更新すると、構成全体が置き換えられます。つまり、既存の値を 1 つだけ変更する場合、それ以外は同じであっても、すべての詳細を再度渡す必要があります。

スケジュールの一時停止

メンテナンス期間など、選択した時間のスケジュールを一時停止できます。スケジュールが一時停止されると、実行されると想定されていた実行がスキップされます。スケジュールを一時停止するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [スケジューラ（scheduler）] セクションで、`PATCH/schedule` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、[スケジュール ID（Schedule ID）] を指定し、[スケジュール更新リクエスト（Schedule update request）] で [一時停止（paused）] フィールドの値を `true` に設定します。

ステップ 4 [実行（Execute）] をクリックします。サーバー応答コードが `200` の場合、スケジュールは正常に一時停止されています。

次のリクエストで再開するまでの間、スケジュールは一時停止されたままになります。

スケジュールの一時停止の解除

スケジュールを再開するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから [swagger] アイコンをクリックします。

ステップ 2 CWM API の [スケジューラ（scheduler）] セクションで、`PATCH/schedule` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、[スケジュール ID（Schedule ID）] を指定し、[スケジュール更新リクエスト（Schedule update request）] で [一時停止（paused）] フィールドの値を `false` に設定します。

ステップ 4 [実行（Execute）] をクリックします。サーバー応答コードが `200` の場合、スケジュールは正常に再開されています。

失敗時の一時停止

`pauseOnFailure` フィールドを `true` に設定すると、ジョブのいずれかが失敗した後にスケジュールが自動的に一時停止されます。たとえば、スケジュールに関連付けられたワークフロー定義が削除される場合など、問題に対処する機会が与えられます。失敗時の一時停止の値を変更するには 、「**スケジュールの更新**」の一般的な手順に従います。

オーバーラップポリシーの変更

オーバーラップポリシーは、前の実行が引き続き実行されているときに、同時にスケジュールによって次のジョブを開始する必要がある場合の動作を制御します。デフォルトのポリシーは [スキップ（Skip）] です（[オーバーラップ（overlap）] フィールドの値は `1` に設定される）。つまり、前のジョブがまだ実行中の場合、次にスケジュールされた実行は開始されず、スキップされます。既存の実行が完了すると、その時間より後の次にスケジュールされた実行のみが考慮されます。オーバーラップポリシーを変更するには、「**スケジュールの更新**」の一般的な手順に従います。

使用可能なポリシー：

ポリシータイプ	[オーバーラップ（**overlap**）] フィールド値	説明
スキップ（**Skip**）	1	これは、実行の重複を防ぐデフォルトのポリシーです。スケジュールの前の実行がまだ実行されているときに、次の実行を実行する必要がある場合、次の実行はスキップされます。
1 つをバッファ（**Buffer One**）	2	現在の実行が完了するとすぐに次の実行を開始します。1 つの実行のみがバッファリングされます。現在のジョブの実行中に発生することが想定されている複数の実行である場合、それらはスキップされ、実行中のジョブの終了後にキュー内の最初の実行のみが実行されます。

ポリシータイプ	[オーバーラップ（**overlap**）]フィールド値	説明
すべてをバッファ（**Buffer All**）	3	現在のジョブの実行中に発生することが想定されているすべての実行をバッファリングします。バッファリングされた実行はすべて、実行中のジョブが完了した直後に順番に実行されます。
その他をキャンセル（**Cancel Other**）	4	実行中のジョブをキャンセルし、古いジョブのキャンセルが完了した後に新しいジョブを開始します。
その他を終了（**Terminate Other**）	5	実行中のジョブを終了し、即座に新しいジョブを開始します。
すべて許可（**Allow All**）	[6]	任意の数の同時実行を開始します。

ジョブの管理

ジョブは、オペレータガイドで説明されているようにCWM UIで、またはCWM APIを使用して管理できます。

!!!「重要」複数のタグに基づくジョブのクエリなど、{{version.CWM}}の一部の機能は、APIを介してのみ使用できます。

ジョブを管理するための次のアクションを使用できます。

- `GET/job`：CWMアプリケーションに存在するジョブのリストを取得します。
- `POST/job`：指定されたパラメータに基づいてCWMアプリケーションで実行される新しいジョブを作成します。
- `GET/job/{jobId}/runs/{runId}`：CWMアプリケーションに存在する特定のジョブの詳細を返します。
- `POST/job/{jobId}/runs/{runId}/cancel`：ワークフローワーカーがワークフロー定義から進行中のタスクの実行を完了した後、実行中のジョブの実行をキャンセルします。
- `GET/job/{jobId}/runs/{runId}/events`：特定のジョブのイベント履歴を返します。
- `POST/job/{jobId}/runs/{runId}/terminate`：実行中のジョブの実行をただちに終了します。

ジョブの実行

ジョブを実行するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [ジョブ（jobs）] セクションで、`POST/job` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で [試す（Try it out）] をクリックし、[ジョブ実行リクエスト（Job Execution Request）] で選択した値を指定します。次に例を示します。

```
{
"data": {},
"jobName": "test API job",
"tags": [
    "test", "API"
],
"workflowName": "Test cisco workflow",
"workflowVersion": "1.1"
}
```

（注） ジョブタグはオプションですが、将来的に特定のジョブのフィルタリングが容易になる場合があります。

ステップ 4 [実行（Execute）] をクリックします。

ジョブ実行が正常に作成された場合は、コード `200` とジョブ ID と実行 ID が返されるサーバー応答を取得する必要があります。

複数タグによるジョブのフィルタ処理

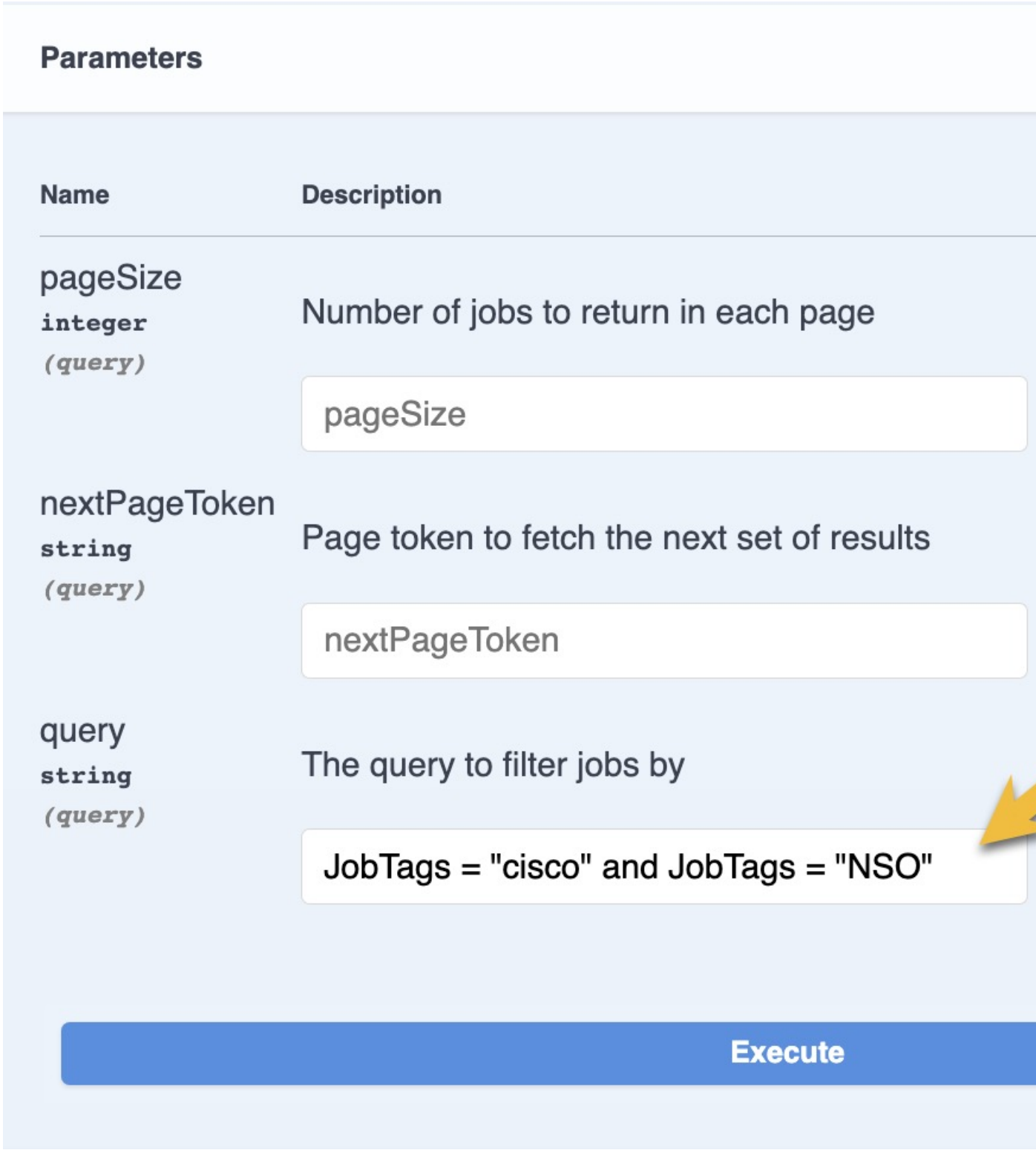
関連付けられたタグなど、特定のクエリでフィルタ処理された CWM に存在するジョブのリストを取得できます。リストを取得するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [ジョブ（jobs）] セクションで、`GET/job/{jobId}/runs/{runId}` エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で [試す（Try it out）] をクリックし、次の例のように、`query` パラメータで `JobTags = "tag_name1" and JobTags = "tag_name2"` というスキーマに従って、ジョブをフィルタ処理するタグを指定します。

図 *6:* ジョブイベントログ



ステップ 4 [実行（Execute）] をクリックします。

フィルタ処理されたジョブの詳細とともに、ステータスコード 200 のサーバー応答が表示されます。

既知の問題：ジョブの詳細を含む応答本文で、workflowId という名前のフィールドは、実際にはワークフロー定義 ID ではなくジョブ ID です。

イベント履歴の取得

特定のジョブのイベント履歴（すべてまたはフィルタ処理済み）のリストを取得するには、次の手順を実行します。

ステップ 1 CWM にログインし、左側のナビゲーションメニューから [swagger] アイコンをクリックします。

ステップ 2 CWM API の [ジョブ（jobs）] セクションで、 GET/job/{jobId}/runs/{runId}/events エンドポイントをクリックして展開します。

ステップ 3 エンドポイント内で、[試す（Try it out）] をクリックし、イベント履歴を取得するジョブの実行 ID とジョブ ID を指定します。

ステップ 4 必要に応じて、現在実行中のジョブをクエリする場合は、isLongPoll パラメータを true に設定できます。その後、特定のジョブの実行が終了するまで接続が開かれ、ジョブの実行が完了すると応答が返されます。false に設定すると、リクエストの送信後すぐに、リクエストの時点までに完了したイベントで構成されるイベント履歴が表示されます。

ステップ 5 必要に応じて、filterType パラメータを選択した値（all または close_event）に設定できます。

（注） close_event 値は、WorkflowExecutionFailed や WorkflowExecutionCompleted など、すでに終了したジョブのクローズイベントのみをフィルタ処理します。フィルタとして close_event を選択し、ジョブが現在実行中の場合は、400 エラーが表示されます。

ステップ 6 [実行（Execute）] をクリックします。

フィルタ処理されたイベントとともに、ステータスコード 200 のサーバー応答が表示されます。

イベントタイプの管理

イベントタイプは、CWM によって受信または生成され、ワークフロー定義で参照される信号のカテゴリです。**オペレータ**ガイドで説明されているように CWM UI を使用してワーカーを管理できます。または以下で説明しているように CWM API を使用して管理できます。

イベントタイプを管理するための次のアクションを使用できます。

- GET/eventType：CWM アプリケーションに存在するイベントタイプのリストを取得します。

- `POST/eventType`：CWM アプリケーションで新しいイベントタイプを作成します。
- `GET/eventType/{name}`：CWM アプリケーションに存在する特定のイベントタイプの詳細を取得します。
- `PUT/eventType/{name}`：既存のイベントタイプを新しいパラメータ値で更新します。
- `DELETE/eventType/{name}`：CWM アプリケーションからイベントタイプを削除します。
- `POST/eventType/{name}/start`：イベントリスナーを開始または停止します。

イベントタイプの作成

ステップ 1 CWM にログインし、左側のナビゲーションメニューから :simple-swagger: アイコンをクリックします。

ステップ 2 CWM API の [eventType] セクションで、`POST/eventType` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 3 [eventType データ（eventType data）] フィールドで、必要な値を変更します。

```
    {
"correlation": [
    {
    "contextAttributeName": "string",
    "contextAttributeValue": "string"
    }
],
"createWorkflow": false,
"dataOnly": true,
"endpoint": "string",
"kind": "string",
"name": "string",
"resourceId": "string",
"source": "string",
"type": "string",
"workflowName": "string",
"workflowVersion": "string"
}
```

ステップ 4 [実行（Execute）] をクリックします。

イベントタイプリスナーの開始/停止

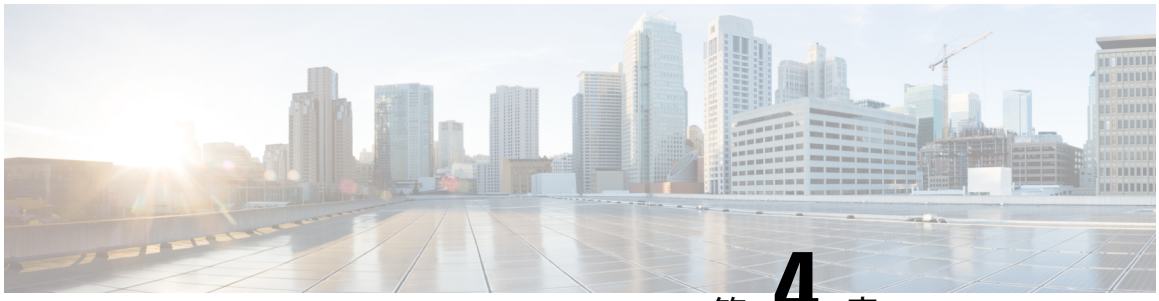
ステップ 1 CWM API の [eventType] セクションで、`POST/{name}/{action}` エンドポイントをクリックして展開します。エンドポイント内で、[試す（Try it out）] をクリックします。

ステップ 2 次の**パラメータ**のフィールドに、必要な値を入力します。

a) "Name"：リスナーを開始/停止する必要があるイベントタイプの名前を貼り付けます。

b) "action"：リスナーを開始する場合は `start` に設定し、実行中のリスナーを停止する場合は `stop` に設定します。

ステップ 3 [実行（Execute）] をクリックします。



第 4 章

Users

ここでは、次の内容について説明します。

ユーザーアクセスの管理

NxF はセキュリティのレイヤーを追加し、単一認証エージェントとして機能するため、ローカル、LDAP、および SAML の各ユーザーを共有します。CWM では NxF を介してユーザーアクセスを管理できます。

CWM 内の NxF 機能

NxF 機能は、管理者ユーザーが CWM UI の [設定（Settings）] タブから使用できます。CWM の NxF 機能にアクセスするには、次の手順を実行します。

ステップ 1 CWM で、左端のナビゲーションメニューに移動します。

ステップ 2 [設定（Settings）]アイコンをクリックします。

図 *7 : NxF* の設定

ステップ 3 展開されたドロワに、次の項目が表示されます。

図 *8 : NxF* ドロワの設定

a) A)[システム情報（System Info）]セクションには、NxFおよびCWMマイクロサービスの最新バージョンに関する情報が表示されます。
b) B) [セキュリティ（Security）] セクションには、アクセス管理に関する次の項目が表示されます。
   - [ローカルユーザー（Local Users）]：UIを介してローカルユーザーを表示、作成、および編集できます。
   - [LDAP]：ユーザー認証の LDAP 設定を構成できます。
   - [SAML SSO]：ユーザー認証の SAML シングルサインオン設定を構成できます。
   - [権限マッピング（Permission Mapping）]：シスコ ポリシー管理ツールを使用して権限管理を操作できます。

ローカルユーザーの追加

ステップ 1 CWM で、左端のナビゲーションメニューに移動します。

ステップ 2 CWM（Cisco アイコン）から [ローカルユーザー（Local Users）] タブに移動します。

ステップ 3 [追加...（Add...）] をクリックします。

ステップ 4 [ユーザーの追加（Add User）]パネルで、必須フィールド（アスタリスクでマークされているフィールド）の [ユーザー名（Username）]（CWMへのログインに使用）、[パスワード（Password）]、[パスワードの確認（Confirm Password）]、[アクセス権限（Access Permissions）]（`permission/user` と入力）に入力します。[説明（Description）] と [表示名（Display Name）]（CWM でユーザー名の横に表示される）はオプションのフィールドです。

図 *9 : NxF* ユーザーの追加

ステップ 5 オプションボタンを使用して、ユーザーステータスを設定します。両方のオプションボタンを同時に無効または有効にできます。

a) [アクティブ有効（Active enabled）] : ユーザーは CWM にログインできます。
b) [アクティブ無効（Active disabled）] : ユーザーは CWM へのログインが禁止されます。
c) [ロック有効（Locked enabled）] : ユーザーの削除を防止します。
d) [Lロック無効（Locked disabled）] : ユーザーの削除を許可します。

ステップ 6 [保存（Save）] をクリックします。

LDAP を介した認証の設定

CWM では、ローカルユーザーのサポートに加えて、LDAP（Lightweight Directory Access Protocol）サーバーとの統合によって LDAP ユーザーを追加できます。

ステップ 1 CWM で、左端のナビゲーションメニューに移動します。

ステップ 2 CWM（Cisco アイコン）から、 [LDAP] タブに移動します。

ステップ 3 [有効（Enabled）] オプションボタンをクリックします。

ステップ 4 必須フィールド（アスタリスクでマークされているフィールド）の [LDAPサーバーアドレス（LDAP Server Address）]、[バインドDN（Bind DN）]、[バインドクレデンシャル（Bind Credentials）]、および [検索フィルタ（Search Filter）] に入力します。[検索ベース（Search Base）] と [ルートCA（Root CAs）] はオプションです。

図 *10 : NxF LDAP*

ステップ 5 [保存（Save）] をクリックします。

SAML SSO を介した認証の設定

CWM は、SAML（セキュリティアサーションマークアップ言語）プロトコルに基づいてシングルサインオンアクセスを取得するために、LDAP ユーザーと非 LDAP ユーザーの両方をサポートする SAML SSO 機能を提供します。CWM の SAML SSO は、LDAP と同時に、または LDAP なしで有効にできます。

ステップ 1 CWM で、左端のナビゲーションメニューに移動します。

ステップ 2 CWM（Cisco アイコン）から [SAML SSO] タブに移動します。

ステップ 3 [有効（Enabled）] オプションボタンをクリックします。

ステップ 4 必須フィールド（[ログインURL（Login URL）]、[エンティティID（Entity ID）]、[ベースURL（Base URL）]、[署名証明書（Signing Certificate）]、および [グループ属性名（Groups Attribute Name）]）に入力します。

図 *11 : NxF SAMLSSO*

ステップ 5 [保存（Save）] をクリックします。

権限マッピングの設定

シスコ ポリシー管理ツール（PMT）を使用して、ユーザーのグループに特定の権限を付与できます。

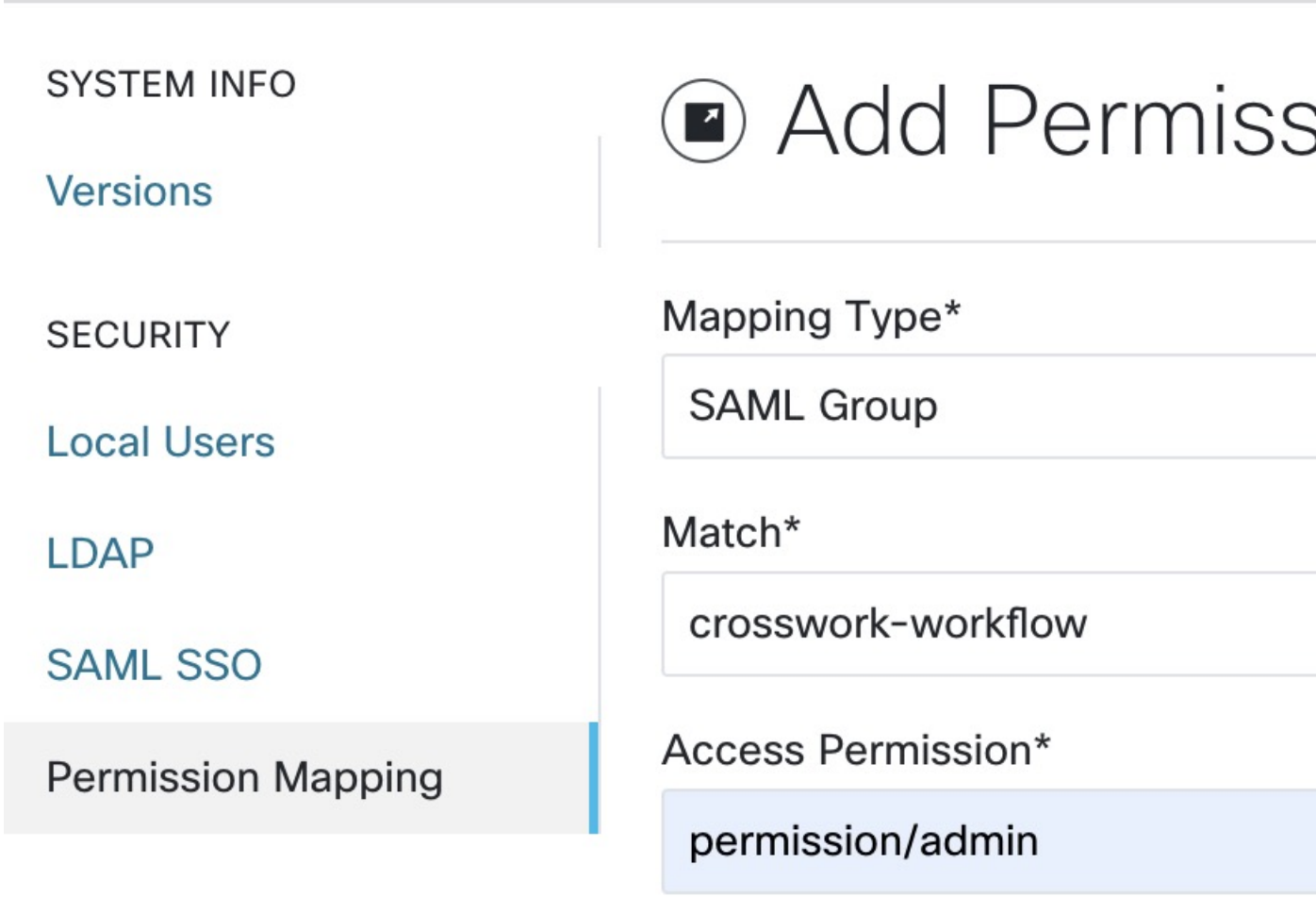
ステップ 1 CWM で、左端のナビゲーションメニューに移動します。

ステップ 2 CWM（Cisco アイコン）から [権限マッピング（Permission Mapping）] タブに移動します。

ステップ 3 [追加...（Add...）] をクリックします。.

ステップ 4 [権限マッピングの追加（Add Permission Mapping）] パネルで、ドロップダウンメニューからマッピングタイプ（SAML ユーザー、SAML グループ、LDAP ユーザー、または LDAP グループ）を選択します。

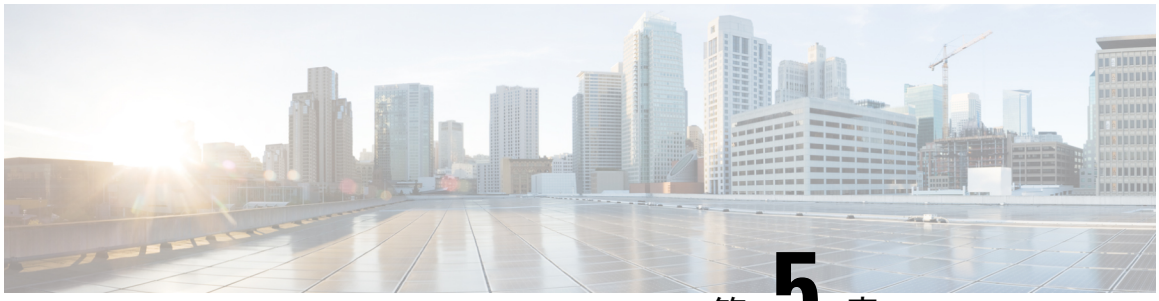
図 *12 : NxF* の権限マッピング



ステップ 5 [一致（Match）] フィールドに、シスコ ポリシー管理ツールのエントリを入力します。一致は、ポリシー管理ツールの UI から [OAuthクライアント（OAuth Clients）] タブに移動して、[クライアントID（Client ID）] 列で見つけることができます。

ステップ 6 [アクセス権限（Access Permission）] フィールドに適切な権限（例：`permission/admin`）を入力します。

ステップ 7 [保存（Save）] をクリックします。



第 5 章

アダプタ

ここでは、次の内容について説明します。

generic-email アダプタの使用

電子メールアダプタ（generic-email）は、SMTP サーバーを使用して電子メールを送信するための基本機能を提供することで、ワークフローにレポートの要素を追加します。1.0.0 アダプタバージョンでは、送信 アクティビティを使用して、ワークフロー定義内で定義されたメッセージを含む電子メールを送信できます。

generic-email アダプタの取得

CWM 1.1 ソフトウェアパッケージをダウンロードしてください。
cwm.v1.1.generic.email.v1.0.0.tar.gz ファイルは、パッケージ内に含まれています。

アダプタのインストール

アダプタをインストールするには、オペレータガイドのアダプタのインストール方法に関する指示に従ってください。

SMTP リソースとシークレットの作成

ワークフロー内での電子メールメッセージの定義の詳細に進む前に、リソースとシークレットを CWM に追加する必要があります。後でワークフロー内でそれらを参照する必要があります。

シークレットの追加

ステップ 1 CWM で、[管理（Admin）] -> [シークレット（Secrets）] タブに移動します。

ステップ 2 [シークレットの追加（Add Secret）] をクリックします。

ステップ 3 [新しいシークレット（New secret）] ビューで、次を指定します。

a) [シークレット ID（Secret ID）]：シークレットに名前を付けます。このシークレット ID は、後でリソースとワークフロー内で参照する必要があります。例：`emailSecret`

b) [シークレットタイプ（Secret type）]：`basicAuth` を選択します。

ステップ 4 シークレットタイプを選択すると、[シークレットタイプの詳細（Secret type details）] セクションに一連の追加フィールドが表示されます。次のフィールドに入力します。

a) password：送信者の電子メールアドレスにパスワードを指定します。

b) username：電子メールの送信元アドレスを `sender@address.com` のフォーマットで指定します。

ステップ 5 [シークレットの作成（Create Secret）] をクリックします。

リソースの追加

ステップ 1 CWM で、[管理（Admin）] -> [リソース（Resources）] タブに移動します。

ステップ 2 [リソースの追加（Add Resource）] をクリックします。

ステップ 3 [新しいリソース（New resource）] ウィンドウで、次を指定します。

a) [リソース名（Resource name）]：リソースに名前を付けます。後でワークフロー内で参照としてリソース ID を指定する必要があります。例：`emailResource`

b) [リソースの種類（Resource type）]：`generic.email.resource.v1.0.0` を選択します。

c) [シークレット ID（Secret ID）]：追加したシークレットの ID を指定します。

d) 接続：

- [ホスト（Host）]：使用する SMTP サーバーのアドレスを指定します。
- [ポート（Port）]：SMTP ポートを指定します。暗号化された電子メール送信用の標準 SMTP ポートは、`587` または `25` です。
- [スキーム（Scheme）]：このフィールドは必須ではありません。
- [タイムアウト（Timeout）]：このフィールドは必須ではありません。
- [非セキュアを許可（Allow Insecure）]：`true` を選択します。

ステップ 4 [リソースの作成（Create a Resource）] をクリックします。

ワークフローでの送信アクティビティの定義

ワークフローでアダプタの送信アクティビティを使用する方法を学習します。

アクティビティ参照の設定

CWM では、アダプタの 送信 アクティビティは、generic.email.smtp.Send と呼ばれます。ワークフローを定義する場合は、functions の operations パラメータの値として指定する必要があります。

```
"functions": [
  {
    "name": "smtp.send",
    "operation": "generic.email.v1.0.0.smtp.Send"
  }
]
```

（注） name パラメータ内で、後でアクションを定義するときに refName パラメータで参照するアクティビティ名を指定します。

actions での電子メールメッセージの定義

ワークフローの状態の一部として電子メールを送信するアクションを定義できるようになりました。

アクションで使用可能な入力パラメータは次のとおりです。

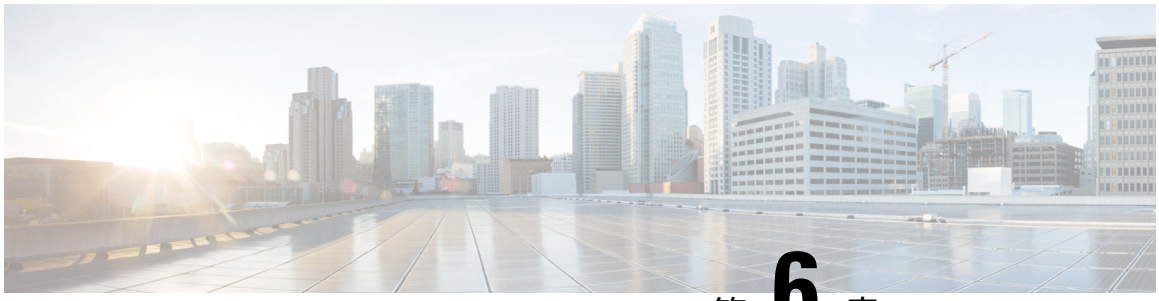
フィールド	タイプ	ラベル	説明
from	string		送信者の電子メールアドレス
to	string	repeated	受信者の電子メールアドレスのリスト
cc	string	repeated	cc 受信者の電子メールアドレスのリスト
bcc	string	repeated	bcc 受信者の電子メールアドレスのリスト
subject	string		電子メールのタイトル
text	string		テキストとしての電子メール本文
html	string		html としての電子メール本文

次に示すように、SendEmail のサンプルアクションを定義する arguments 内で、使用可能なフィールドを input キー/値ペアとして使用します。

```
"states":[
  {
    "name":"EmailState",
    "type":"operation",
    "end": true,
    "actions": [
      {
        "name": "SendEmail",
        "functionRef": {
          "refName": "smtp.send",
          "arguments": {
```

```
            "input": {
              "to": ["recipient1@address.com", "recipient2@address.com"],
              "from": "sender@address.com",
              "text": "Hello, this is some placeholder email text.",
              "subject": "A test email from CWM"
            },
            "config": {
              "resourceId": "emailResource"
            }
          }
        }
      }
    ]
  }
]
```

条件に基づいて電子メールアクションをトリガーする場合は、スイッチの状態を使用し、その中に dataConditions パラメータを定義できます。詳細については、スイッチの状態に関するサーバレスワークフロー仕様のドキュメントを参照してください。



第 6 章

イベント

ここでは、次の内容について説明します。

イベント処理の概要

イベント処理メカニズムにより、CWM はアウトバウンドおよびインバウンドのイベントを処理するために外部ブローカと連携できます。ワークフローは、新しいワークフローを開始したり、既存のワークフローを通知したりするために使用できるイベントのコンシューマまたはプロデューサとして機能できます。さらに、CWM で定義するイベントタイプごとに、イベントをフィルタ処理し、特定の属性値を含むイベントを待機しているワークフローにイベントをルーティングするための相関属性を追加できます。

イベントメッセージは、クラウドイベント仕様に従って定義する必要があります。詳細については、イベントフォーマットのセクションを参照してください。

ブローカとプロトコル

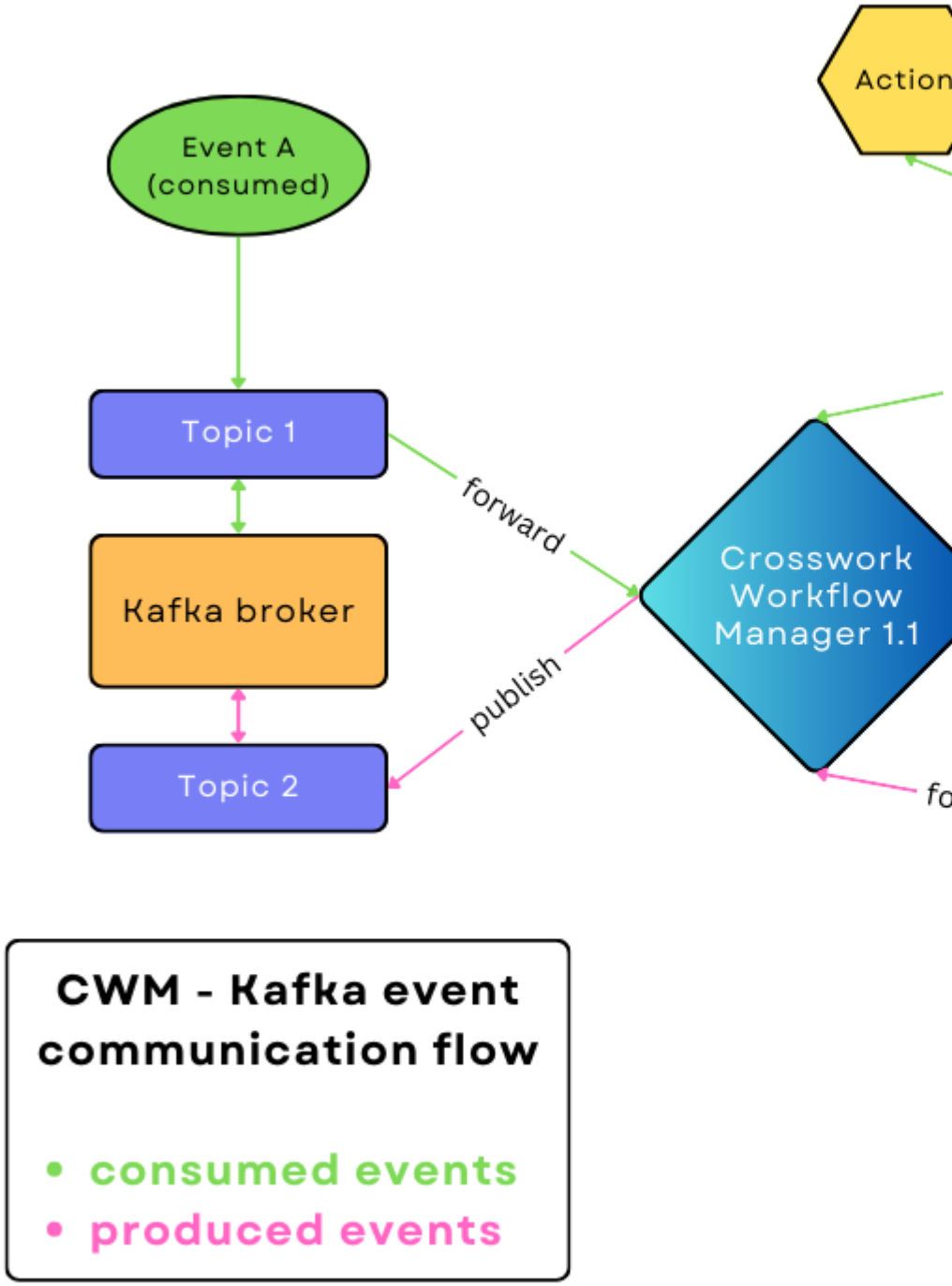
Crosswork Workflow Manager 1.1 は、イベントを処理するために Kafka ブローカと AMQP および HTTP プロトコルをサポートしています。イベントは、CWM 内で実行されているワークフローによって**消費**されるか、実行中のワークフローによって**生成**（ブローカによって転送される着信イベント）され、外部システムに転送（ブローカによって受信される発信イベント）されます。

（注） CWM はイベントブローカ自体として機能しないことに注意してください。外部ブローカに接続してメッセージ/イベントを転送する手段を提供します。

Kafka ブローカ

consume イベントの種類の場合、CWM は Kafka ブローカに接続し、トピックの特定のイベントタイプをリッスンします。特定のタイプのイベントが適切なトピックに登録されると、CWM はイベントデータを取得し、実行中のワークフローに転送します。次に、ワークフローはイベントステート内で定義されたアクションを実行するか、別のワークフロー実行を実行します（選択されている場合）。

produce イベントの種類の場合、実行中のワークフローは単一のイベントまたは一連のイベントを生成し、CWM がブローカに転送し、それらは適切なトピックで公開されます。



Kafkaブローカは、有効なコンテンツタイプが送信される限り、言語固有のSDKでサポートされているすべてのイベント メッセージ フォーマットを受け入れます。サポートされているフォーマットのリストは、https://github.com/cloudevents/spec?tab=readme-ov-file にあります。

AMQP プロトコル（RabbitMQ ブローカなど）

consume イベントの種類の場合、CWM は AMQP ブローカに接続し、キューで特定のイベントタイプをリッスンします。Kafka ブローカと同様に、特定のタイプのイベントが適切なキューに登録されると、CWM はイベントデータを取得し、実行中のワークフローに転送します。次に、ワークフローはイベントステート内で定義されたアクションを実行するか、別のワークフロー実行を実行します（選択されている場合）。

produce イベントの種類の場合、実行中のワークフローは単一のイベントまたは一連のイベントを生成し、CWM がブローカに転送し、それらは適切なキューで公開されます。

AMQP ブローカは、有効なコンテンツタイプが送信される限り、特定の SDK でサポートされているすべてのイベントメッセージフォーマットを受け入れます。サポートされているフォーマットのリストは、https://github.com/cloudevents/spec?tab=readme-ov-file にあります。

HTTP プロトコル

consume イベントの種類の場合、CWM は着信イベントをリッスンする HTTP エンドポイントを公開します。特定のタイプのイベントが発生すると、このイベントタイプを待機する実行中のワークフローに転送されます。

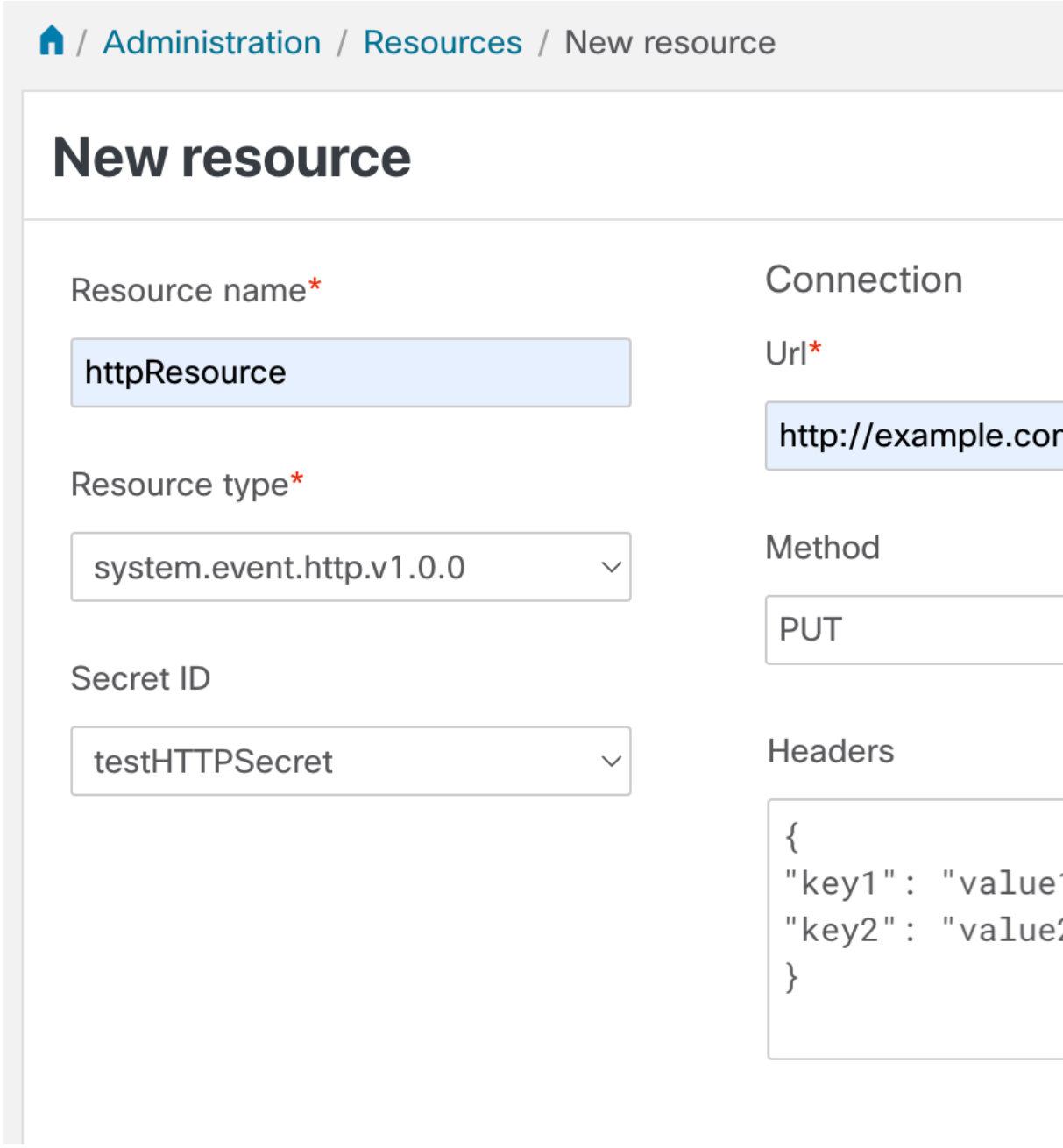
（注） イベントが消費されると、CWM は接続先 HTTP サーバーとして機能します。したがって、CWM サーバーの URL は、特定の HTTP イベントタイプのリソースとして効果的に指定されます。

イベントメッセージは HTTP *POST* リクエストである必要があり、メッセージ本文はクラウドイベントを表す JSON フォーマットである必要があります。

（注） 例： `{ "specversion": "1.0", "id": "2763482-4-324-32-4", "type": "com.github.pull_request.opened", "source": "/sensors/tn-1234567/alerts", "datacontenttype": "text/xml", "data": "<test=\"xml\"/>", "contextAttrName": "contextAttrValue" }`

produce イベントの種類の場合、ワークフローはクラウドイベントフォーマットでイベントを生成し、CWM はそれを HTTP *POST* リクエストとして外部システムによって公開されている HTTP エンドポイントに転送します。HTTP エンドポイントアドレスは、CWM のリソース構成で定義されたホスト **URL** と、ワークフロー定義内のイベント定義の [エンドポイント（End point）] フィールドを連結したものです。リソース構成内で、リクエストメソッドを *PUT* また

はその他に変更し、キーと値のペアをヘッダー（JSON フォーマット）として追加できます。



イベントシステム構成

シークレット

イベント構成では、イベントを送信または受信するサードパーティサービスによって公開されているブローカまたはエンドポイントへの接続を有効にするために必要なログイン情報がシー

クレットに保存されます。これには、基本認証（ユーザー名とパスワード）が含まれます。シークレットの作成時に指定したシークレット ID は、リソースの作成時に参照されるため、事前にシークレットを追加する必要があります。その方法については、シークレットの追加に関するセクションを参照してください。

リソース

リソースは、サードパーティサービスによって公開されたイベントブローカまたはエンドポイントに到達するために必要なすべての接続の詳細（シークレットを含む）を提供する場所です。使用するブローカ/プロトコルに応じて、次の 3 つのデフォルトのイベントリソースタイプから選択できます。

- `system.event.amqp.v1.0.0`
- `system.event.kafka.v1.0.0`
- `system.event.http.v1.0.0`

それぞれに異なる構成フィールドのセットがあることに注意してください。

- AMQP の場合は、 `amqp://localhost:5723` のフォーマットで **ServerDSN** を指定します。
- Kafka の場合：
  - [KafkaVersion]：Kafka のバージョンを指定します。Kafka のバージョンを確認する標準的な方法は、端末で `bin/kafka-topics.sh --version` を実行することです。
  - [ブローカ（Brokers）]：`["localhost:9092", "192.168.10.9:9092"]` のフォーマットで Kafka ブローカアドレスを指定します。
  - [OtherSettings]：デフォルトの Kafka 設定値を含む編集可能なリスト。これらの値は必要に応じて変更できます。
- HTTP の場合：
  - **produce** イベントの種類： [URL] フィールドに入力し、必要に応じて [メソッド（Method）] と [ヘッダー（Headers）]（たとえば、JSON オブジェクトとしてのクライアント ID ヘッダー名と値）を入力します。

（注） **URL** は接続先 HTTP サーバーのアドレスである必要がありますが、URL パスは不要です。イベントタイプを設定するときに、URL パスを**エンドポイント**として指定します。

  - **consume** イベントの種類：[URL] フィールドに CWM インスタンスのサーバー URL を入力します（例：`192.168.10.9:9092`）。

（注） URL パス（/event/http）なしで CWM インスタンスの URL を指定することを忘れないでください。後でイベントタイプを設定するときに、URL パスを**エンドポイント**として使用します。

Event type

（注） 新しいイベントタイプを作成するには、リソースとシークレットを CWM に追加する必要があります。

イベントタイプを追加する場合は、次のフィールドを使用できます。

- [イベントタイプ名（Event type name）]：イベントタイプの名前。後でワークフロー定義内で参照されます。
- [リソース（Resource）]：以前に CWM に追加されたリソースのリスト。
  - [イベントソース（Event source）]：ワークフロー定義で参照される完全にユーザー定義のエントリ。produce イベントの種類の場合は必須です。
  - [エンドポイント（End point）]：Kafka トピック（イベントストリーム）、AMQP エンドポイント（終端）、または HTTP URL（ホスト）パスの名前。

（注） HTTP **consume** イベントの種類の場合は、**エンドポイント**として /event/http を指定します。

- [種類の選択（Select kind）]：consume または produce のイベントの種類の 2 つのオプションから構成されるリスト。

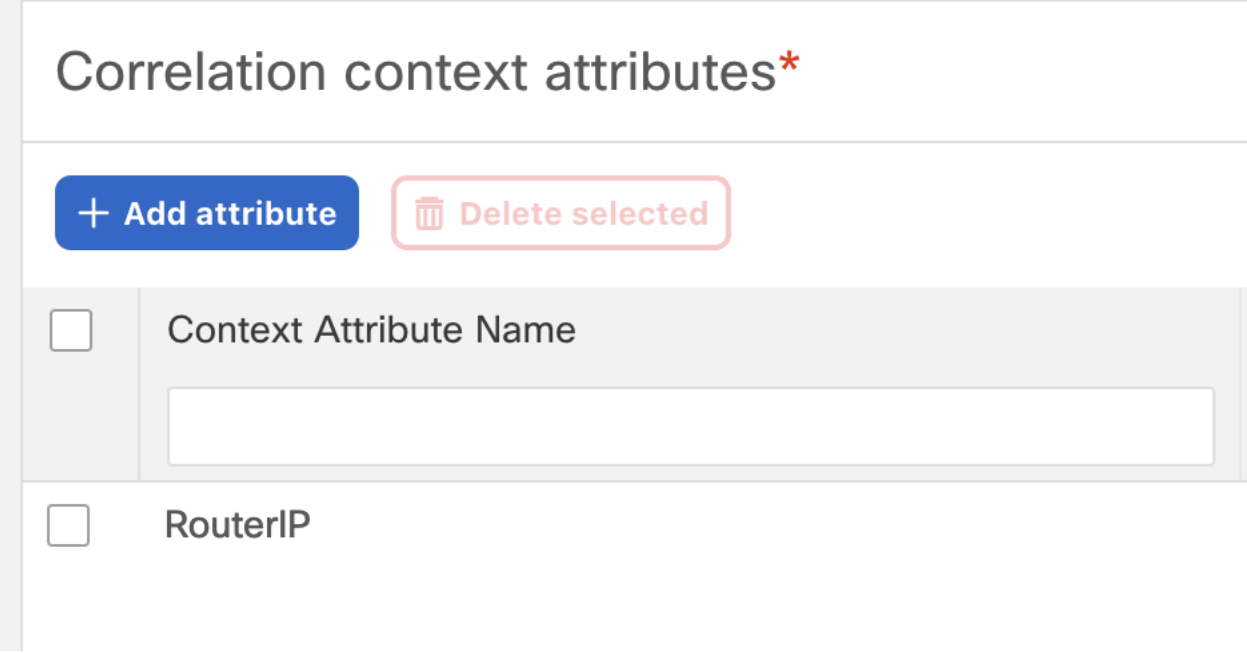
（注） both オプションは、{{ version.CWM }} ではまだサポートされていません。

- [リスナーの開始（Start listener）]（consume の種類の場合のみ）：クリックすると、定義されたイベントタイプのリスニングが開始されます。
- [ジョブの実行（Run job）]（consume の種類の場合のみ）：イベントの受信時にワークフローをトリガーする場合は、このチェックボックスをオンにします。その後、リストから目的のワークフローを選択します。

相関属性

必要に応じて、イベントのコンテキスト属性を設定できます。これらは consume イベントの種類にのみ適用され、ワークフローを選択的にトリガーするために使用されます。それらは、インバウンドイベントデータを絞り込み、相関属性の特定の値を持つイベントタイプをリッスンする適切なワークフローにルーティングする一種のカスタムフィルタとして表示できます。

イベントタイプに属性を追加するには、[属性の追加（Add attribute）] をクリックし、属性名と値を指定して、[追加（Add）] をクリックします。



（注） 相関属性は完全にユーザー定義です。特定のワークフローにルーティングされるクラウドイベントメッセージ内に記載されている JSON キーと値のペアと一致する必要があります。

イベントメッセージのフォーマット

イベントメッセージは、クラウドイベント仕様のフォーマットに従う必要があります。最小実行可能イベントメッセージには、次のパラメータが含まれます。

```
{
  "specversion": "1.0",
  "id": "00001",
  "type": "com.github.pull_request.opened",
  "source": "/sensors/tn-1234567/alerts"
}
```

メッセージには、"datacontenttype"、"data"、相関コンテキスト属性名（この例では contextAttrName）などの追加のパラメータを含めることができます。

```
  {
  "specversion": "1.0",
  "id": "2763482-4-324-32-4",
```

```
  "type": "com.github.pull_request.opened",
  "source": "/sensors/tn-1234567/alerts",
  "datacontenttype": "text/xml",
  "data": "<test data=\"xml\"/>",
  "contextAttrName": "contextAttrValue"
}
```

ワークフローイベントの定義とステート

ワークフロー定義には、ワークフローが待機するイベントを処理するために使用する2つの主要な構文要素があります。次のものがあります。

- イベント定義：イベントタイプとそのプロパティを定義するために使用されます。

```
{
   "name": "applicant-info",
   "type": "org.application.info",
   "source": "applicationssource",
   "correlation": [
    {
      "contextAttrName": "applicantId"
    }
   ]
}
```

- イベントステート：イベントが発生したときに実行するアクションを定義するために使用されます。

```
{
"name": "MonitorVitals",
"type": "event",
"onEvents": [
        {
          "actions": [
            {
              "functionRef": {
                "refName": "uppercase",
                "arguments": {
                  "input": {
                    "in": "patient ${ .patient } has high temperature"
                  }
                }
              }
            }
          ],
          "eventRefs": [
            "HighBodyTemperature"
          ]
        }
      ]
}
```

Kafka イベントの定義

前提条件

- セットアップされた Kafka サービス（または AMQP の場合は AMQP 1.0 プラグインを使用した RabbitMQ、または任意の HTTP クライアント）。
- CWM 1.1 が OVA を使用してインストールされている。

ステップ 1：Kafka シークレットとリソースの作成

Kafka サービスへのセキュアな接続を有効にするには、Kafka ログイン情報を使用してシークレットを作成し、接続の詳細を含むリソースを作成する必要があります。作成方法は次のとおりです。

シークレットの作成

ステップ 1 CWM で、[管理（Admin）] -> [シークレット（Secrets）] タブに移動します。

ステップ 2 [シークレットの追加（Add Secret）] をクリックします。

ステップ 3 [新しいシークレット（New secret）] ビューで、次を指定します。

a) [シークレット ID（Secret ID）]：`KafkaSecret`

b) [シークレットタイプ（Secret type）]：`basicAuth`

ステップ 4 シークレットタイプを選択すると、[シークレットタイプの詳細（Secret type details）] セクションに一連の追加フィールドが表示されます。フィールドに入力します。

a) [パスワード（password）]：Kafka へのログインに使用されるパスワード。

b) [ユーザー名（username）]：Kafka へのログインに使用されるユーザー名。

ステップ 5 [シークレットの作成（Create Secret）] をクリックします。

リソースの作成

ステップ 1 CWM で、[管理（Admin）] -> [リソース（Resources）] タブに移動します。

ステップ 2 [リソースの追加（Add Resource）] をクリックします。

ステップ 3 [新しいリソース（New resource）] ウィンドウで、次を指定します。

a) [リソース名（Resource name）]：`KafkaResource`

b) [リソースの種類（Resource name）]：`cisco.cwm.kafka.v1.0.0`（または、代わりにこれらのプロトコルを使用する場合は `cisco.cwm.amqp.v1.0.0` または `cisco.cwm.http.v1.0.0`）

c) [シークレット ID（Secret ID）]：`KafkaSecret`
d) 接続：

- [KafkaVersion]：Kafka のバージョンを指定します。これを確認する標準的な方法は、端末で `bin/kafka-topics.sh --version` を実行することです。
- [ブローカ（Brokers）]：`["localhost:9092"]` の形式で Kafka ブローカアドレスを指定します。
- [OtherSettings]：デフォルトの Kafka 設定値を含む編集可能なリスト。これらの値は必要に応じて変更できます。

（注） 接続設定は、**AMQP** と **HTTP** リソースタイプの場合は異なります。

- AMQP の場合は、**ServerDNS** を「amqp://localhost:5723」形式で指定します。- HTTP の場合は、**URL** と追加の **headers**（クライアント ID ヘッダー名と値など）を指定します。URL はホストアドレスである必要がありますが、URL パスは不要である点に注意してください。これは、リソースの種類の設定時に **End point** として指定します。

ステップ 4 [リソースの作成（Create a Resource）] をクリックします。

ステップ 2：CWM へのイベントタイプの追加

シークレットとリソースを用意したら、CWM によって消費または生成されるイベントのタイプを指定します。

ステップ 1 CWM UI で、左側のナビゲーションメニューから [管理（Admin）] タイルを選択します。

ステップ 2 [イベントシステム（Event system）] パネルで、[イベントタイプの追加（Add event type）] をクリックします。

ステップ 3 [新しいイベントタイプ（New event type）] モーダルで、必要な入力を行います。

a) [イベントタイプ名（Event type name）]：イベントタイプの名前を指定します。後でワークフロー定義内で参照します。

b) [リソース（Resource）]：リストから `KafkaResource` を選択します。

c) [イベントソース（Event source）]：イベントソースを定義します。これは完全にユーザー定義であり、ワークフロー定義で参照されます。`produce` イベントの種類の場合は必須です。

d) [エンドポイント（End point）]：Kafka の場合、Kafka トピック（イベントストリーム）を指定します。AMQP の場合は、エンドポイント（終端）を指定します。HTTP の場合は、URL（ホスト）パスを指定します。

e) [種類の選択（Select kind）]：リストから `consume` を選択します。

（注） ワークフローによって生成され、別のシステムによって消費されるイベントを定義するには、`Produce` を使用します。この場合、これ以降に示されている残りの **ステップ 2** の設定は適用されません。`both` オプションは、CWM 1.1 ではまだサポートされていません。

f) [リスナーの開始（Start listener）]：クリックすると、定義されたイベントタイプのリスニングが開始されます。

g) [ジョブの実行（Run job）]：イベントの受信時にワークフローをトリガーする場合は、このチェックボックスをオンにします。その後、リストから目的のワークフローを選択します。

h) [相関コンテキスト属性（Correlation context attributes）]：必要に応じて、イベントのコンテキスト属性を設定できます。これらは `consume` イベントの種類にのみ適用され、ワークフローを選択的にトリガーするために使用されます。それらは、インバウンドイベントデータを絞り込み、コンテキスト属性に基づいて「リスニング」ワークフロー内で定義されたアクションをトリガーする一種のカスタムフィルタとして表示できます。

i) [属性の追加（Add attribute）] をクリックし、属性名と値（完全にユーザー定義）を指定します。

ステップ 4 [イベントタイプの作成（Create Event type）] をクリックします。

ステップ 3：ワークフローでのイベントの定義

イベントタイプが追加されたので、このイベントタイプに登録して、イベントが CWM で受信されたときにアクションを実行するワークフローを作成できます。そのためには、イベント定義を使用してイベントを定義し、イベントステートを指定し、イベントが発生したときに実行

するアクションを定義する必要があります。例として、ルータ過熱アラーム（インバウンドイベント）がワークフローイベントステートをトリガーし、2つの修復アクションが実行されるシナリオを取り上げてみましょう。

```
{
  "id": "HighRouterTempWorkflow",
  "name": "Router Overheating Alarm Workflow",
  "start": "RemediateHighTemp",
  "events": [
    {
      "kind": "consumed",
      "name": "HighRouterTemp",
      "type": "HighRouterTemp",
      "source": "monitoring.app"
    }
  ],
  "states": [
    {
      "end": {
        "terminate": true
      },
      "name": "RemediateHighTemp",
      "type": "event",
      "onEvents": [
        {
          "actions": [
            {
              "functionRef": {
                "refName": "DispatchTech",
                "contextAttributes": {
                  "RouterIP": "${ .RouterIP }"
                },
                "resultEventTimeout": "PT30M"
              }
            },
          ],
          "eventRefs": ["HighRouterTemp"],
           },
           {
           "actions": [
            {
              "functionRef": {
                "refName": "MoveTraffic",
                "contextAttributes": {
                  "RouterIP": "${ .RouterIP }"
                },
                "resultEventTimeout": "PT30M"
              }
            }
          ],
          "timeouts": {
            "actionExecTimeout": "PT60M"
          }
        }
      ]
    }
  ],
  "version": "1.0.0",
  "description": "Remediate router overheating",
  "specVersion": "0.8"
}
```

（注） この例は完全なワークフローではないことに注意してください。ワークフロー内でイベントを定義し、それに基づいてアクションを実行する方法の例を示します。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。