RESTを使用して自動化する
ONTAP Select

NetApp
November 09, 2026

This PDF was generated from https://docs.netapp.com/ja-jp/ontap-select-9191/concept_api_rest.html on November 09, 2026. Always check docs.netapp.com for the latest.

目次

RESTを使用して自動化する 1
概念 1
ONTAP Select クラスターのデプロイと管理のための REST Web サービスについて 1
ONTAP Select Deploy APIへのアクセスについて 2
ONTAP Select Deploy APIの基本的な運用特性 2
ONTAP Select のリクエストとレスポンスの API トランザクション 4
ONTAP Select の Job オブジェクトを使用した非同期処理 7
ブラウザでアクセス 8
ブラウザを使用した ONTAP Select Deploy API へのアクセスについて 8
ONTAP Select Deployのドキュメントページにアクセスします 9
ONTAP Select Deploy API 呼び出しを理解して実行する 9
ワークフロープロセス 10
ONTAP Select Deploy API ワークフローの使用方法について学ぶ 10
ESXi 上での ONTAP Select シングルノード評価クラスタの作成 10
Pythonでアクセス 17
Python を使用した ONTAP Select Deploy API へのアクセスについて 17
ONTAP Select Deploy 向け Python スクリプトについて 18
Pythonコードサンプル 19
ONTAP Select クラスタを作成するスクリプト 19
ONTAP Select クラスタを作成するスクリプト用の JSON 26
ONTAP Selectノードライセンスを追加するスクリプト 31
ONTAP Select クラスタを削除するスクリプト 34
ONTAP Select の共通サポート Python モジュール 36
ONTAP Select クラスタノードのサイズを変更するスクリプト 40

RESTを使用して自動化する

概念

ONTAP Select クラスターのデプロイと管理のための REST Web サービスについて

Representational State Transfer（REST）は、分散型Webアプリケーションを作成するためのスタイルです。WebサービスAPIの設計に適用すると、サーバーベースのリソースを公開し、その状態を管理するための一連のテクノロジとベストプラクティスが確立されます。主流のプロトコルと標準を使用して、ONTAP Selectクラスタの導入と管理のための柔軟な基盤を提供します。

アーキテクチャと従来の制約

RESTは、2000年にUC Irvineで Roy Fielding が博士論文 "学位論文"の中で正式に提唱しました。これは、一連の制約を通じてアーキテクチャスタイルを定義するもので、これらの制約が集合的にWebベースのアプリケーションと基盤となるプロトコルを改善します。これらの制約により、ステートレス通信プロトコルを使用するクライアント/サーバーアーキテクチャに基づくRESTful Webサービスアプリケーションが確立されます。

リソースと状態の表示

リソースはWebベース システムの基本コンポーネントです。REST Webサービス アプリケーションを作成する場合、設計の早い段階で次の作業を行います。

- システムまたはサーバーベースのリソースの識別 すべてのシステムはリソースを使用し、維持します。リソースとは、ファイル、ビジネス取引、プロセス、または管理エンティティなどを指します。REST Webサービスに基づいたアプリケーションを設計する際の最初のタスクの1つは、リソースを特定することです。
- リソースの状態と関連する状態操作の定義 リソースは常に有限個の状態のいずれかにあります。状態、および状態変化に影響を与えるために使用される関連操作は、明確に定義されなければなりません。

クライアントとサーバー間でメッセージが交換され、一般的なCRUD（作成、読み取り、更新、削除）モデルに従ってリソースの状態にアクセスしたり変更したりします。

URIエンドポイント

すべてのRESTリソースは、明確に定義されたアドレス指定方式を使用して定義、提供される必要があります。リソースが置かれているエンドポイントは、Uniform Resource Identifier（URI）で識別されます。URIは、ネットワークの各リソースに一意の名前を作成するための一般的なフレームワークです。Uniform Resource Locator（URL）は、リソースを識別してアクセスするためにWebサービスで使用されるURIの一種です。リソースは、通常、ファイル ディレクトリに似た階層構造で公開されます。

HTTPメッセージ

ハイパーテキスト転送プロトコル（HTTP）は、Webサービスクライアントとサーバーがリソースに関するリクエストメッセージとレスポンスメッセージを交換するために使用するプロトコルです。Webサービスアプリケーションの設計の一環として、HTTP動詞（GETやPOSTなど）はリソースとそれに対応する状態管理アクションにマッピングされます。

HTTPはステートレスです。したがって、一連の関連するリクエストとレスポンスを1つのトランザクションに関連付けるには、リクエスト/レスポンスのデータフローとともに伝送されるHTTPヘッダーに追加情報を含める必要があります。

JSONの形式

クライアントとサーバー間で情報を構造化して転送する方法はいくつかありますが、最も一般的なオプション（Deploy REST APIで使用されるもの）はJavaScript Object Notation（JSON）です。JSONは、単純なデータ構造をプレーンテキストで表現するための業界標準であり、リソースを記述する状態情報の転送に使用されます。

ONTAP Select Deploy APIへのアクセスについて

REST Web サービスの本来の柔軟性により、ONTAP Select Deploy API には、いくつかの異なる方法でアクセスできます。

> ONTAP Select Deploy に付属の REST API にはバージョン番号が割り当てられます。API のバージョン番号は、Deploy のリリース番号とは無関係です。ONTAP Select 9.19.1 Deploy 管理ユーティリティには、バージョン 3 の REST API が含まれています。

ユーティリティネイティブユーザーインターフェイスを導入する

APIにアクセスする主な方法は、ONTAP Select Deploy Webユーザーインターフェイスからアクセスすることです。ブラウザはAPIを呼び出し、ユーザーインターフェースのデザインに合わせてデータを再フォーマットします。DeployユーティリティのコマンドラインインターフェースからもAPIにアクセスできます。

ONTAP Select Deploy オンライン ドキュメント ページ

ONTAP Select Deploy オンライン ドキュメント ページは、ブラウザを使用する際に代替のアクセス ポイントを提供します。このページでは、個々の API 呼び出しを直接実行する方法を提供するだけでなく、各呼び出しの入力パラメータやその他のオプションを含む、API の詳細な説明も掲載しています。API 呼び出しは、いくつかの異なる機能領域またはカテゴリに分類されています。

カスタムプログラム

Deploy APIには、さまざまなプログラミング言語やツールを使用してアクセスできます。人気のある選択肢としては、Python、Java、cURLなどが挙げられます。APIを使用するプログラム、スクリプト、またはツールは、REST Webサービスクライアントとして機能します。プログラミング言語を使用することで、APIをよりよく理解でき、ONTAP Selectデプロイメントを自動化する機会が得られます。

ONTAP Select Deploy APIの基本的な運用特性

RESTは共通の技術とベストプラクティスを確立するものの、各APIの詳細は設計上の選択によって異なる場合があります。APIを使用する前に、ONTAP Select Deploy APIの詳細と動作特性を把握しておく必要があります。

ハイパーバイザー ホストと ONTAP Select ノード

_ハイパーバイザー ホスト_は、ONTAP Select仮想マシンをホストするコア ハードウェア プラットフォームです。ONTAP Select仮想マシンがハイパーバイザー ホスト上に導入されてアクティブになると、その仮想マ

シンは_ONTAP Selectノード_とみなされます。Deploy REST APIのバージョン3では、ホスト オブジェクトとノード オブジェクトは別個のものです。これにより、1対多の関係が可能になり、1つ以上のONTAP Selectノードを同じハイパーバイザー ホスト上で実行できます。

オブジェクトID

各リソースインスタンスまたはオブジェクトは、作成時に一意の識別子が割り当てられます。これらの識別子は、ONTAP Select Deployの特定のインスタンス内でグローバルに一意です。新しいオブジェクトインスタンスを作成するAPI呼び出しを発行すると、関連付けられたID値がHTTPレスポンスの `location`ヘッダーで呼び出し元に返されます。識別子を抽出して、リソースインスタンスを参照する際の以降の呼び出しで使用できます。

> オブジェクト識別子の内容と内部構造は変更になることがあります。識別子を使用するのは、該当するAPI呼び出しで関連付けられているオブジェクトを参照するために必要な場合だけにしてください。

リクエスト識別子

成功したAPIリクエストにはすべて、一意の識別子が割り当てられます。識別子は、関連するHTTPレスポンスの `request-id`ヘッダーで返されます。リクエスト識別子を使用すると、特定の単一のAPIリクエスト・レスポンス取引における一連の活動をまとめて参照できます。例えば、リクエストIDに基づいて、トランザクションに関するすべてのイベントメッセージを取得できます。

同期呼び出しと非同期呼び出し

サーバーがクライアントから受信したHTTPリクエストを処理する主な方法は2つあります。

- 同期型：サーバーはリクエストを即座に実行し、ステータスコード200、201、または204で応答します。
- 非同期：サーバーはリクエストを受け入れ、ステータスコード202で応答します。これは、サーバーがクライアントからのリクエストを受け入れ、リクエストを完了するためのバックグラウンドタスクを開始したことを示しています。最終的な成功または失敗はすぐには判明せず、追加のAPI呼び出しによって判定する必要があります。

長時間実行されるジョブの完了を確認する

一般的に、完了までに時間がかかる可能性のある操作は、サーバー側のバックグラウンドタスクを使用して非同期的に処理されます。Deploy REST APIでは、すべてのバックグラウンドタスクは、タスクを追跡し、現在の状態などの情報を提供するJobオブジェクトによってアンカーされます。バックグラウンドタスクが作成されると、一意の識別子を含むJobオブジェクトがHTTPレスポンスで返されます。

Jobオブジェクトを直接照会することで、関連するAPI呼び出しの成否を判断できます。詳細については、「Jobオブジェクトを使用した非同期処理」を参照してください。

Job オブジェクトを使用する以外にも、リクエストの成功または失敗を判断する方法はいくつかあります。

- イベントメッセージ 元のレスポンスで返されたリクエストIDを使用すると、特定のAPI呼び出しに関連付けられたすべてのイベントメッセージを取得できます。イベントメッセージには通常、成功または失敗を示す情報が含まれており、エラー状態のデバッグにも役立ちます。
- リソースの状態またはステータス 一部のリソースは、状態またはステータス値を保持しており、これを照会することでリクエストの成功または失敗を間接的に判断できます。

セキュリティ

Deploy API は、以下のセキュリティ技術を使用しています：

- トランスポート層セキュリティ Deployサーバーとクライアント間のネットワーク上で送信されるすべてのトラフィックは、TLSによって暗号化されます。暗号化されていないチャネル上でHTTPプロトコルを使用することはサポートされていません。TLSバージョン1.2をサポートしています。
- HTTP認証　すべてのAPIトランザクションには基本認証が使用されます。ユーザー名とパスワードをbase64文字列で含むHTTPヘッダーが、すべてのリクエストに追加されます。

ONTAP Select のリクエストとレスポンスの API トランザクション

Deploy API呼び出しはすべて、Deploy仮想マシンへのHTTPリクエストとして実行され、その仮想マシンはクライアントに対して関連する応答を生成します。このリクエストとレスポンスのペアは、APIトランザクションとみなされます。Deploy APIを使用する前に、リクエストを制御するために使用できる入力変数と、レスポンス出力の内容について理解しておく必要があります。

API要求を制御する入力変数

HTTPリクエストに設定されたパラメータを通して、API呼び出しの処理方法を制御できます。

要求ヘッダー

HTTPリクエストには、以下のヘッダーを含める必要があります：

- content-type リクエスト本文に JSON が含まれる場合、このヘッダーは application/json に設定する必要があります。
- accept レスポンスボディに JSON が含まれる場合、このヘッダーは application/json に設定する必要があります。
- authorization 基本認証は、ユーザー名とパスワードをbase64文字列でエンコードして設定する必要があります。

要求の本文

要求の本文の内容は、それぞれの呼び出しに応じて異なります。HTTP要求の本文は、次のいずれかで構成されます。

- 入力変数（新しいクラスタの名前など）を含むJSONオブジェクト
- 空の

オブジェクトをフィルタリング

GETを使用するAPI呼び出しを発行する際、返されるオブジェクトを任意の属性に基づいて制限またはフィルタできます。たとえば、一致する完全な値を指定できます。

```
<field>=<query value>
```

完全一致に加えて、値の範囲にわたるオブジェクトのセットを返すための他の演算子も利用可能で

す。ONTAP Selectは以下のフィルタリング演算子をサポートしています。

演算子	説明
=	等しい
<	小なり
>	より大きい
⇐	次の値以下
>=	次の値以上
	または
!	等しくない
*	すべてに一致するワイルドカード

また、nullキーワードまたはその否定（!null）をクエリの一部として使用することで、特定のフィールドが設定されているかどうかに基づいてオブジェクトのセットを返すこともできます。

オブジェクトフィールドの選択

デフォルトでは、GETメソッドを使用してAPI呼び出しを行うと、オブジェクトを一意に識別する属性のみが返されます。この最小限のフィールドセットは、各オブジェクトのキーとして機能し、オブジェクトの種類によって異なります。フィールドクエリパラメータを使用して、以下の方法で追加のオブジェクトプロパティを選択できます：

- 安価なフィールドを指定 `fields=*`して、ローカルサーバーのメモリに保持されている、またはアクセスにほとんど処理を必要としないオブジェクトフィールドを取得します。
- 高価なフィールド `fields=**`を指定して、追加のサーバー処理が必要なフィールドを含むすべてのオブジェクトフィールドを取得します。
- カスタムフィールドの選択 `fields=FIELDNAME`を使用して、希望するフィールドを正確に指定してください。複数のフィールドを要求する場合は、値をスペースを入れずにカンマで区切る必要があります。



ベストプラクティスとして、必要な特定のフィールドを常に特定する必要があります。安価なフィールドまたは高価なフィールドのセットは、必要な場合にのみ取得してください。安価および高価の分類は、内部パフォーマンス分析に基づいてNetAppによって決定されます。特定のフィールドの分類はいつでも変更される可能性があります。

出力セット内のオブジェクトをソートします

リソースコレクション内のレコードは、オブジェクトによって定義されたデフォルトの順序で返されます。order_byクエリパラメータにフィールド名とソート方向を指定して、順序を変更できます。

```
order_by=<field name> asc|desc
```

例えば、typeフィールドを降順でソートし、次にidを昇順でソートすることができます。

```
order_by=type desc, id asc
```

複数のパラメータを指定する場合は、各フィールドをカンマで区切る必要があります。

ページネーション

GETを使用して同じタイプのオブジェクトのコレクションにアクセスするAPI呼び出しを発行すると、デフォルトで一致するすべてのオブジェクトが返されます。必要に応じて、リクエストでmax_recordsクエリパラメータを使用して、返されるレコード数を制限できます。例：
`max_records=20`

必要に応じて、このパラメータを他のクエリパラメータと組み合わせて、結果セットを絞り込むことができます。例えば、以下のコードは、指定された時刻以降に生成された最大10個のシステムイベントを返します：
`time⇒ 2019-04-04T15:41:29.140265Z&max_records=10`

イベント（または任意のオブジェクトタイプ）をページ送りするために、複数のリクエストを発行できます。以降のAPI呼び出しでは、前回の結果セットに含まれる最新のイベントに基づいて、新しい時間値を使用する必要があります。

API応答を解釈する

各APIリクエストは、クライアントへの応答を生成します。応答を検証することで、それが成功したかどうかを判断し、必要に応じて追加データを取得できます。

HTTPステータス コード

Deploy REST API で使用される HTTP ステータスコードを以下に説明します。

Code	説明	説明
200	OK	新しいオブジェクトを作成しない呼び出しが成功したことを示します。
201	Created	オブジェクトが正常に作成されました。ロケーション応答ヘッダーには、オブジェクトの一意の識子が含まれています。
202	Accepted	リクエストを実行するために長時間実行されるバックグラウンドジョブが開始されましたが、処理はまだ完了していません。
400	Bad request	要求の入力が認識されないか不適切です。
403	Forbidden	認証エラーのためアクセスが拒否されました。
404	Not found	要求で参照されているリソースが存在しません。
405	Method not allowed	リクエストで使用されているHTTP動詞は、このリソースではサポートされていません。
409	競合	オブジェクトがすでに存在するため、オブジェクトの作成に失敗しました。
500	内部エラー	サーバで一般的な内部エラーが発生しました。
501	未実装	URIは既知だが、リクエストを実行する能力がない。

応答ヘッダー

Deployサーバーによって生成されるHTTPレスポンスには、以下のヘッダーが含まれます：

- request-id 成功したすべてのAPIリクエストには、一意のリクエスト識別子が割り当てられます。
- オブジェクトが作成されると、location ヘッダーには、一意のオブジェクト識別子を含む、新しいオブジ

ェクトへの完全な URL が含まれます。

応答の本文

APIリクエストに関連付けられたレスポンスの内容は、オブジェクト、処理タイプ、およびリクエストの成功または失敗によって異なります。レスポンスボディはJSON形式で出力されます。

- 単一オブジェクト リクエストに基づいて、一連のフィールドを持つ単一のオブジェクトが返されます。例えば、GETを使用して、一意の識別子を用いてクラスタの選択したプロパティを取得できます。
- 複数のオブジェクト リソースコレクションから複数のオブジェクトを返すことができます。いずれの場合も、一貫した形式が使用され、 `num_records`でレコード数が示され、recordsにオブジェクトインスタンスの配列が含まれます。たとえば、特定のクラスタで定義されているすべてのノードを取得できます。
- JobオブジェクトAPI呼び出しが非同期で処理される場合、バックグラウンドタスクのアンカーとなるJobオブジェクトが返されます。例えば、クラスタの導入に使用されるPOSTリクエストは非同期で処理され、Jobオブジェクトが返されます。
- エラーオブジェクト エラーが発生した場合は、必ずエラーオブジェクトが返されます。たとえば、すでに存在する名前でクラスタを作成しようとすると、エラーが発生します。
- 空の場合：場合によってはデータが返されず、レスポンスボディが空になります。例えば、DELETEを使用して既存のホストを削除した場合、レスポンスボディは空になります。

ONTAP Select の Job オブジェクトを使用した非同期処理

Deploy API の呼び出しの中には、特にリソースを作成または変更する呼び出しは、他の呼び出しよりも完了に時間がかかる場合があります。ONTAP Select Deploy は、これらの長時間実行されるリクエストを非同期で処理します。

ジョブ オブジェクトを使用して記述された非同期要求

非同期的に実行されるAPI呼び出しを行うと、HTTP応答コード202が返されます。この応答コードは、要求が正常に検証され受け入れられたものの、まだ完了していないことを示します。要求はバックグラウンド タスクとして処理され、クライアントへの最初のHTTP応答後も引き続き実行されます。応答には、要求に対応するジョブ オブジェクトと、その一意の識別子が含まれます。

どの API 呼び出しが非同期で動作するかを判断するには、ONTAP Select Deploy オンライン ドキュメント ページを参照してください。

API リクエストに関連付けられた **Job** オブジェクトを照会する

HTTP応答で返されるジョブ オブジェクトには、いくつかのプロパティが含まれています。状態プロパティを照会して、要求が正常に完了したかどうかを確認できます。ジョブ オブジェクトは次のいずれかの状態になります。

- キュー登録済み
- 実行中
- 成功
- 失敗

ジョブ オブジェクトをポーリングしてタスクの最終状態（成功または失敗）を検出するには、2つの方法があります。

- 標準のポーリング要求 現在のジョブの状態はすぐに返されます
- ロングポーリングリクエストのジョブ状態は、以下のいずれかの条件が満たされた場合にのみ返されます：
  - ポーリング要求で指定された日時値よりも最近、状態が変更されました
  - タイムアウト値が経過しました（1～120秒）

標準ポーリングとロングポーリングは、ジョブオブジェクトを照会するために同じAPI呼び出しを使用します。ただし、ロングポーリングリクエストには2つのクエリパラメータが含まれます（`poll_timeout`および`last_modified`）。



Deploy 仮想マシンのワークロードを軽減するには、常にロング ポーリングを使用する必要があります。

非同期要求を発行するための一般的な手順

以下は、非同期API呼び出しを完了する手順の概要です。

1. 非同期API呼び出しを実行します。
2. 要求が正常に受け取られたことを示すHTTP応答202を受信します。
3. 応答の本文からジョブ オブジェクトの識別子を抽出します。
4. ループ内で、各サイクルで以下の処理を実行します：
   a. ロングポーリングリクエストを使用してジョブの現在の状態を取得します
   b. ジョブが非終了状態（キューに入っている、実行中）の場合は、ループを再度実行します。
5. ジョブが終了状態（成功、失敗）に達したら停止します。

ブラウザでアクセス

ブラウザを使用した **ONTAP Select Deploy API** へのアクセスについて

Deployのオンラインドキュメントページを使用する前に、いくつか注意すべき点があります。

導入計画

特定のデプロイメントや管理タスクを実行する際にAPI呼び出しを行う予定がある場合は、デプロイメントプランを作成することを検討してください。これらの計画は正式なものも非公式なものもあり、一般的には目標と使用するAPI呼び出しが含まれています。詳細については、Deploy REST APIを使用したワークフロープロセスを参照してください。

JSONの例とパラメータ定義

各API呼び出しは、一貫した形式を用いてドキュメントページで説明されています。内容には、実装に関する

注意事項、クエリパラメータ、HTTPステータスコードが含まれます。さらに、APIリクエストとレスポンスで使用されるJSONの詳細を以下のように表示できます：

- 値の例 API 呼び出しで 値の例 を選択すると、その呼び出しの典型的な JSON 構造が表示されます。必要に応じて例を修正し、リクエストの入力として使用できます。
- Model _Model_を選択すると、JSONパラメータの完全なリストが表示され、各パラメータの説明が表示されます。

API 呼び出しを行う際の注意点

Deploy に関するドキュメント ページを使用して実行するすべての API 操作は、ライブ操作です。設定データやその他のデータを誤って作成、更新、または削除しないように注意してください。

ONTAP Select Deployのドキュメントページにアクセスします

API ドキュメントを表示したり、手動で API 呼び出しを実行したりするには、ONTAP Select Deploy オンライン ドキュメント ページにアクセスする必要があります。

開始する前に

以下のものが必要です。

- ONTAP Select Deploy仮想マシンのIPアドレスまたはドメイン名
- 管理者のユーザー名とパスワード

手順

1. ブラウザに URL を入力して **Enter** キーを押してください。

   ```
   https://<ip_address>/api/ui
   ```

2. 管理者のユーザー名とパスワードを使用してサインインしてください。

結果

Deployに関するドキュメントのWebページが表示され、ページ下部にカテゴリ別に整理された呼び出しが表示されます。

ONTAP Select Deploy API 呼び出しを理解して実行する

すべての API 呼び出しの詳細は、ONTAP Select Deploy オンライン ドキュメント Web ページで共通のフォーマットを使用して文書化され、表示されます。単一の API 呼び出しを理解することで、すべての API 呼び出しの詳細にアクセスして解釈できます。

開始する前に

ONTAP Select Deploy オンライン ドキュメントの Web ページにサインインする必要があります。クラスタの作成時に ONTAP Select クラスタに割り当てられた一意の識別子が必要です。

タスク概要

一意の識別子を使用して、ONTAP Selectクラスタを記述する構成情報を取得できます。この例では、安価と分類されたすべてのフィールドが返されます。ただし、ベストプラクティスとして、必要な特定のフィールド

のみを要求する必要があります。

手順

1. メイン ページで、一番下までスクロールして **Cluster** を選択します。
2. **GET /clusters/{cluster_id}** を選択して、ONTAP Select クラスタに関する情報を返すために使用された API 呼び出しの詳細を表示します。

ワークフロープロセス

ONTAP Select Deploy API ワークフローの使用方法について学ぶ

ワークフロープロセスを見直し、活用できるように準備してください。

ワークフローで使用される **API** 呼び出しを理解する

ONTAP Select オンライン ドキュメント ページには、すべての REST API 呼び出しの詳細が記載されています。ここでそれらの詳細を繰り返すのではなく、ワークフロー サンプルで使用される各 API 呼び出しには、ドキュメント ページで呼び出しを見つけるために必要な情報のみが含まれています。特定のAPI呼び出しを見つけたら、入力パラメータ、出力形式、HTTPステータス コード、要求処理のタイプなど、呼び出しの詳細を確認できます。

ワークフローで使用している各API呼び出しについて、ドキュメント ページで見つけるのに役立つように次の情報を示します。

- カテゴリ API呼び出しは、ドキュメントページ上で機能的に関連する領域またはカテゴリごとに整理されています。特定のAPI呼び出しを探すには、ページの一番下までスクロールして、該当するAPIカテゴリを選択してください。
- HTTP動詞 HTTP動詞は、リソースに対して実行されるアクションを識別します。各API呼び出しは、単一のHTTP動詞によって実行されます。
- パス パスは、呼び出しを実行する際にアクションが適用される特定のリソースを決定します。コアのURLのあとにパス文字列を追加することで、リソースを識別する完全なURLが形成されます。

REST API に直接アクセスするための **URL** を構築します

ONTAP Select ドキュメント ページに加えて、Python などのプログラミング言語を介して Deploy REST API に直接アクセスすることもできます。この場合、コア URL は、オンライン ドキュメント ページにアクセスする際に使用する URL とは若干異なります。API に直接アクセスする場合は、ドメイン名とポート番号の文字列に /api を追加する必要があります。例：

```
http://deploy.mycompany.com/api
```

ESXi 上での **ONTAP Select** シングルノード評価クラスタの作成

vCenter で管理されている VMware ESXi ホスト上にシングルノード ONTAP Select クラスタをデプロイできます。このクラスタは評価版ライセンスで作成されます。

クラスタ作成のワークフローは、以下の状況で異なります。

- ESXi ホストは vCenter によって管理されていません（スタンドアロンホスト）。

- クラスター内では、複数のノードまたはホストが使用されます。
- クラスターは、購入済みライセンスを使用した本番環境にデプロイされます。
- KVM ハイパーバイザーは VMware ESXi の代わりに使用されます。

ステップ **1**：**vCenter** サーバーの認証情報を登録する

vCenterサーバーによって管理されているESXiホストにデプロイする場合は、ホストを登録する前にクレデンシャルを追加する必要があります。これにより、Deploy管理ユーティリティでそのクレデンシャルを使用してvCenterに対する認証を行うことができます。

カテゴリ	**HTTP**動詞	パス
導入	POST	/security/credentials

Curl

```
curl -iX POST -H 'Content-Type: application/json' -u admin:<password> -k
-d @step01 'https://10.21.191.150/api/security/credentials'
```

JSON入力（ステップ**01**）

```
{
  "hostname": "vcenter.company-demo.com",
  "type": "vcenter",
  "username": "misteradmin@vsphere.local",
  "password": "mypassword"
}
```

処理タイプ

非同期

出力

- ロケーション レスポンス ヘッダー内のクレデンシャル ID
- ジョブ オブジェクト

ステップ **2**：ハイパーバイザーホストを登録する

ONTAP Select ノードを含む仮想マシンが実行されるハイパーバイザー ホストを追加します。

カテゴリ	**HTTP**動詞	パス
クラスタ	POST	/hosts

Curl

```
curl -iX POST -H 'Content-Type: application/json' -u admin:<password> -k
-d @step02 'https://10.21.191.150/api/hosts'
```

JSON入力（ステップ**02**）

```
{
  "hosts": [
    {
      "hypervisor_type": "ESX",
      "management_server": "vcenter.company-demo.com",
      "name": "esx1.company-demo.com"
    }
  ]
}
```

処理タイプ

非同期

出力

- ロケーション応答ヘッダーのホスト ID
- ジョブ オブジェクト

ステップ**3**：クラスタを作成する

ONTAP Select クラスタを作成すると、基本的なクラスタ構成が登録され、Deploy がノード名を自動的に生成します。

カテゴリ	**HTTP**動詞	パス
クラスタ	POST	/clusters

Curl

シングルノード クラスタの場合は、クエリパラメータ node_count を 1 に設定する必要があります。

```
curl -iX POST -H 'Content-Type: application/json' -u admin:<password> -k
-d @step03 'https://10.21.191.150/api/clusters? node_count=1'
```

JSON入力（ステップ**03**）

```
{
  "name": "my_cluster"
}
```

処理タイプ
同期

出力
- ロケーション レスポンス ヘッダー内のクラスタ ID

ステップ**4**：クラスターを構成する

クラスターの設定の一環として、以下の属性を指定する必要があります。

カテゴリ	**HTTP**動詞	パス
クラスタ	PATCH	/clusters/{cluster_id}

Curl
クラスター ID を指定してください。

```
curl -iX PATCH -H 'Content-Type: application/json' -u admin:<password> -k
-d @step04 'https://10.21.191.150/api/clusters/CLUSTERID'
```

JSON入力（ステップ**04**）

```
{
  "dns_info": {
    "domains": ["lab1.company-demo.com"],
    "dns_ips": ["10.206.80.135", "10.206.80.136"]
    },
    "ontap_image_version": "9.5",
    "gateway": "10.206.80.1",
    "ip": "10.206.80.115",
    "netmask": "255.255.255.192",
    "ntp_servers": {"10.206.80.183"}
}
```

処理タイプ
同期

出力
None

ステップ**5**：ノード名を取得する

Deploy管理ユーティリティは、クラスターが作成されると、ノードの識別子と名前を自動的に生成します。ノードを設定する前に、割り当てられたIDを取得する必要があります。

カテゴリ	HTTP動詞	パス
クラスタ	GET	/clusters/{cluster_id}/nodes

Curl

クラスター ID を指定してください。

```
curl -iX GET -u admin:<password> -k
'https://10.21.191.150/api/clusters/CLUSTERID/nodes?fields=id,name'
```

処理タイプ

同期

出力

- 一意のIDと名前を持つ単一のノードをそれぞれ記述する配列レコード

ステップ**6**：ノードの設定

ノードの基本設定を提供します。これは、ノードを設定するために使用される 3 つの API 呼び出しのうちの最初のものです。

カテゴリ	HTTP動詞	パス
クラスタ	PATH	/clusters/{cluster_id}/nodes/{node_id}

Curl

クラスター ID とノード ID を入力してください。

```
curl -iX PATCH -H 'Content-Type: application/json' -u admin:<password> -k
-d @step06 'https://10.21.191.150/api/clusters/CLUSTERID/nodes/NODEID'
```

JSON入力（ステップ**06**）

ONTAP Select ノードが実行されるホスト ID を指定してください。

```
{
  "host": {
    "id": "HOSTID"
    },
  "instance_type": "small",
  "ip": "10.206.80.101",
  "passthrough_disks": false
}
```

処理タイプ

同期

出力

None

ステップ**7**：ノードネットワークを取得する

シングルノード クラスタのノードで使用されるデータネットワークと管理ネットワークを特定します。内部ネットワークはシングルノード クラスタでは使用されません。

カテゴリ	**HTTP**動詞	パス
クラスタ	GET	/clusters/{cluster_id}/nodes/{node_id}/networks

Curl

クラスター ID とノード ID を入力してください。

```
curl -iX GET -u admin:<password> -k 'https://10.21.191.150/api/
clusters/CLUSTERID/nodes/NODEID/networks?fields=id,purpose'
```

処理タイプ

同期

出力

- ノードの単一ネットワークをそれぞれ記述する2つのレコードの配列。一意のIDと目的を含む

ステップ **8**：ノード ネットワークの設定

データネットワークと管理ネットワークを設定します。内部ネットワークはシングルノード クラスタでは使用されません。

次のAPI呼び出しを2回実行します。各ネットワークにつき1回ずつです。

カテゴリ	**HTTP**動詞	パス
クラスタ	PATCH	/clusters/{cluster_id}/nodes/{node_id}/networks/{network_id}

Curl

クラスター ID、ノード ID、およびネットワーク ID を入力してください。

```
curl -iX PATCH -H 'Content-Type: application/json' -u admin:<password> -k
-d @step08 'https://10.21.191.150/api/clusters/
CLUSTERID/nodes/NODEID/networks/NETWORKID'
```

JSON入力（ステップ**08**）

ネットワーク名を入力してください。

```
{
```

```
  "name": "sDOT_Network"
}
```

処理タイプ
　同期

出力
　None

ステップ**9**：ノードストレージプールを構成する

ノードの設定における最後のステップは、ストレージプールを接続することです。利用可能なストレージプールは、vSphere Web クライアント、またはオプションで Deploy REST API 経由で確認できます。

カテゴリ	**HTTP**動詞	パス
クラスタ	PATCH	/clusters/{cluster_id}/nodes/{node_id}/networks/{network_id}

Curl
　クラスター ID、ノード ID、およびネットワーク ID を入力してください。

```
curl -iX PATCH -H 'Content-Type: application/json' -u admin:<password> -k
-d @step09 'https://10.21.191.150/api/clusters/ CLUSTERID/nodes/NODEID'
```

JSON入力（ステップ**09**）
　プール容量は2 TBです。

```
{
  "pool_array": [
    {
      "name": "sDOT-01",
      "capacity": 2147483648000
    }
  ]
}
```

処理タイプ
　同期

出力
　None

ステップ**10**：クラスターをデプロイする

クラスターとノードの設定が完了したら、クラスターをデプロイできます。

カテゴリ	HTTP動詞	パス
クラスタ	POST	/clusters/{cluster_id}/deploy

Curl

クラスター ID を指定してください。

```
curl -iX POST -H 'Content-Type: application/json' -u admin:<password> -k
-d @step10 'https://10.21.191.150/api/clusters/CLUSTERID/deploy'
```

JSON入力（ステップ**10**）

ONTAP管理者アカウントのパスワードを入力してください。

```
{
  "ontap_credentials": {
    "password": "mypassword"
  }
}
```

処理タイプ

非同期

出力

- ジョブ オブジェクト

関連情報

"ONTAP Select クラスタの 90 日間の評価インスタンスをデプロイします"

Pythonでアクセス

Python を使用した ONTAP Select Deploy API へのアクセスについて

サンプルPythonスクリプトを実行する前に、環境を準備する必要があります。

Pythonスクリプトを実行する前に、環境が正しく設定されていることを確認する必要があります：

- 最新バージョンのPython2をインストールする必要があります。サンプルコードはPython2を使用してテスト済みです。これらはPython3にも移植可能であるはずですが、互換性テストは実施されていません。
- Requests ライブラリと urllib3 ライブラリをインストールする必要があります。環境に応じて、pip またはその他の Python 管理ツールを使用できます。
- スクリプトが実行されるクライアント ワークステーションには、ONTAP Select Deploy 仮想マシンへのネットワーク アクセスが必要です。

さらに、以下の情報が必要です：

- Deploy仮想マシンのIPアドレス
- Deploy管理者アカウントのユーザー名とパスワード

ONTAP Select Deploy 向け Python スクリプトについて

サンプルPythonスクリプトを使用すると、さまざまなタスクを実行できます。実際のDeployインスタンスで使用する前に、スクリプトの内容を理解しておく必要があります。

共通の設計特性

これらのスクリプトは、以下の共通の特徴に基づいて設計されています。

- クライアントマシンのコマンドラインインターフェースから実行する 適切に設定されたクライアントマシンであれば、どのマシンからでもPythonスクリプトを実行できます。詳細については、_始める前に_をご覧ください。
- CLI入力パラメータを受け入れる 各スクリプトは、入力パラメータを介してCLIで制御されます。
- 入力ファイルの読み込み 各スクリプトは、その目的に応じて入力ファイルを読み込みます。クラスタを作成または削除する際には、JSON設定ファイルを提供する必要があります。ノードライセンスを追加する際には、有効なライセンスファイルを提供する必要があります。
- 共通サポートモジュールを使用する 共通サポートモジュール_deploy_requests.py_には、単一のクラスが含まれています。これは各スクリプトによってインポートされ、使用されます。

クラスタを作成します。

cluster.pyスクリプトを使用してONTAP Selectクラスタを作成できます。CLIパラメータとJSON入力ファイルの内容に基づいて、以下のようにスクリプトをデプロイ環境に合わせて変更できます。

- ハイパーバイザー：デプロイ先のリリースに応じて、ESXiまたはKVMにデプロイできます。ESXiにデプロイする場合、ハイパーバイザーはvCenterで管理することも、スタンドアロンホストとして動作することも可能です。
- クラスターサイズ：シングルノードまたはマルチノードのクラスターを導入できます。
- 評価版または本番環境用ライセンス　評価版ライセンスまたは購入した本番環境用ライセンスを使用してクラスターをデプロイできます。

スクリプトのCLI入力パラメータは次のとおりです：

- Deploy サーバのホスト名または IP アドレス
- 管理者ユーザーアカウントのパスワード
- JSON設定ファイルの名前
- メッセージ出力の詳細表示フラグ

ノードライセンスを追加する

本番環境クラスターをデプロイする場合は、スクリプト_add_license.py_を使用して各ノードにライセンスを追加する必要があります。ライセンスは、クラスターをデプロイする前でも後でも追加できます。

スクリプトのCLI入力パラメータは次のとおりです：

- Deploy サーバのホスト名または IP アドレス
- 管理者ユーザーアカウントのパスワード
- ライセンスファイルの名前
- ライセンスを追加する権限を持つ ONTAP ユーザー名
- ONTAP ユーザーのパスワード

クラスタを削除

スクリプト_delete_cluster.py_を使用して、既存のONTAP Selectクラスターを削除できます。

スクリプトのCLI入力パラメータは次のとおりです：

- Deploy サーバのホスト名または IP アドレス
- 管理者ユーザーアカウントのパスワード
- JSON設定ファイルの名前

Pythonコードサンプル

ONTAP Select クラスタを作成するスクリプト

以下のスクリプトを使用すると、スクリプト内で定義されたパラメータとJSON入力ファイルに基づいてクラスタを作成できます。

```
#!/usr/bin/env python
##-----------------------------------------------------------------------
#
# File: cluster.py
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-----------------------------------------------------------------------
```

```
import traceback
import argparse
import json
import logging

from deploy_requests import DeployRequests


def add_vcenter_credentials(deploy, config):
    """ Add credentials for the vcenter if present in the config """
    log_debug_trace()

    vcenter = config.get('vcenter', None)
    if vcenter and not deploy.resource_exists('/security/credentials',
                                              'hostname', vcenter[
'hostname']):
        log_info("Registering vcenter {} credentials".format(vcenter[
'hostname']))
        data = {k: vcenter[k] for k in ['hostname', 'username', 'password
']}
        data['type'] = "vcenter"
        deploy.post('/security/credentials', data)


def add_standalone_host_credentials(deploy, config):
    """ Add credentials for standalone hosts if present in the config.
        Does nothing if the host credential already exists on the Deploy.
    """
    log_debug_trace()

    hosts = config.get('hosts', [])
    for host in hosts:
        # The presense of the 'password' will be used only for standalone
hosts.
        # If this host is managed by a vcenter, it should not have a host
'password' in the json.
        if 'password' in host and not deploy.resource_exists(
'/security/credentials',
                                                             'hostname',
host['name']):
            log_info("Registering host {} credentials".format(host['name
']))
            data = {'hostname': host['name'], 'type': 'host',
                    'username': host['username'], 'password': host[
'password']}
            deploy.post('/security/credentials', data)
```

```
def register_unkown_hosts(deploy, config):
    ''' Registers all hosts with the deploy server.
        The host details are read from the cluster config json file.

        This method will skip any hosts that are already registered.
        This method will exit the script if no hosts are found in the
config.
    '''
    log_debug_trace()

    data = {"hosts": []}
    if 'hosts' not in config or not config['hosts']:
        log_and_exit("The cluster config requires at least 1 entry in the
'hosts' list got {}".format(config))

    missing_host_cnt = 0
    for host in config['hosts']:
        if not deploy.resource_exists('/hosts', 'name', host['name']):
            missing_host_cnt += 1
            host_config = {"name": host['name'], "hypervisor_type": host[
'type']}
            if 'mgmt_server' in host:
                host_config["management_server"] = host['mgmt_server']
                log_info(
                   "Registering from vcenter {mgmt_server}".format(**
host))

            if 'password' in host and 'user' in host:
                host_config['credential'] = {
                    "password": host['password'], "username": host['user
']}

            log_info("Registering {type} host {name}".format(**host))
            data["hosts"].append(host_config)

    # only post /hosts if some missing hosts were found
    if missing_host_cnt:
        deploy.post('/hosts', data, wait_for_job=True)


def add_cluster_attributes(deploy, config):
    ''' POST a new cluster with all needed attribute values.
        Returns the cluster_id of the new config
    '''
```

```
    log_debug_trace()

    cluster_config = config['cluster']
    cluster_id = deploy.find_resource('/clusters', 'name', cluster_config
['name'])

    if not cluster_id:
        log_info("Creating cluster config named {name}".format(
**cluster_config))

        # Filter to only the valid attributes, ignores anything else in
the json
        data = {k: cluster_config[k] for k in [
            'name', 'ip', 'gateway', 'netmask', 'ontap_image_version',
'dns_info', 'ntp_servers']}

        num_nodes = len(config['nodes'])

        log_info("Cluster properties: {}".format(data))

        resp = deploy.post('/v3/clusters?node_count={}'.format(num_nodes),
data)
        cluster_id = resp.headers.get('Location').split('/')[-1]

    return cluster_id


def get_node_ids(deploy, cluster_id):
    ''' Get the the ids of the nodes in a cluster. Returns a list of
node_ids.'''
    log_debug_trace()

    response = deploy.get('/clusters/{}/nodes'.format(cluster_id))
    node_ids = [node['id'] for node in response.json().get('records')]
    return node_ids


def add_node_attributes(deploy, cluster_id, node_id, node):
    ''' Set all the needed properties on a node '''
    log_debug_trace()

    log_info("Adding node '{}' properties".format(node_id))

    data = {k: node[k] for k in ['ip', 'serial_number', 'instance_type',
                                 'is_storage_efficiency_enabled'] if k in
node}
    # Optional: Set a serial_number
```

```
    if 'license' in node:
        data['license'] = {'id': node['license']}

    # Assign the host
    host_id = deploy.find_resource('/hosts', 'name', node['host_name'])
    if not host_id:
        log_and_exit("Host names must match in the 'hosts' array, and the
nodes.host_name property")

    data['host'] = {'id': host_id}

    # Set the correct raid_type
    is_hw_raid = not node['storage'].get('disks')  # The presence of a
list of disks indicates sw_raid
    data['passthrough_disks'] = not is_hw_raid

    # Optionally set a custom node name
    if 'name' in node:
        data['name'] = node['name']

    log_info("Node properties: {}".format(data))
    deploy.patch('/clusters/{}/nodes/{}'.format(cluster_id, node_id),
data)


def add_node_networks(deploy, cluster_id, node_id, node):
    ''' Set the network information for a node '''
    log_debug_trace()

    log_info("Adding node '{}' network properties".format(node_id))

    num_nodes = deploy.get_num_records('/clusters/{}/nodes'.format
(cluster_id))

    for network in node['networks']:

        # single node clusters do not use the 'internal' network
        if num_nodes == 1 and network['purpose'] == 'internal':
            continue

        # Deduce the network id given the purpose for each entry
        network_id = deploy.find_resource('/clusters/{}/nodes/{}/networks
'.format(cluster_id, node_id),
                                          'purpose', network['purpose'])
        data = {"name": network['name']}
        if 'vlan' in network and network['vlan']:
```

```
            data['vlan_id'] = network['vlan']

        deploy.patch('/clusters/{}/nodes/{}/networks/{}'.format(
cluster_id, node_id, network_id), data)


def add_node_storage(deploy, cluster_id, node_id, node):
    ''' Set all the storage information on a node '''
    log_debug_trace()

    log_info("Adding node '{}' storage properties".format(node_id))
    log_info("Node storage: {}".format(node['storage']['pools']))

    data = {'pool_array': node['storage']['pools']}  # use all the json
properties
    deploy.post(
        '/clusters/{}/nodes/{}/storage/pools'.format(cluster_id, node_id),
data)

    if 'disks' in node['storage'] and node['storage']['disks']:
        data = {'disks': node['storage']['disks']}
        deploy.post(
            '/clusters/{}/nodes/{}/storage/disks'.format(cluster_id,
node_id), data)


def create_cluster_config(deploy, config):
    ''' Construct a cluster config in the deploy server using the input
json data '''
    log_debug_trace()

    cluster_id = add_cluster_attributes(deploy, config)

    node_ids = get_node_ids(deploy, cluster_id)
    node_configs = config['nodes']

    for node_id, node_config in zip(node_ids, node_configs):
        add_node_attributes(deploy, cluster_id, node_id, node_config)
        add_node_networks(deploy, cluster_id, node_id, node_config)
        add_node_storage(deploy, cluster_id, node_id, node_config)

    return cluster_id


def deploy_cluster(deploy, cluster_id, config):
    ''' Deploy the cluster config to create the ONTAP Select VMs. '''
    log_debug_trace()
```

```
    log_info("Deploying cluster: {}".format(cluster_id))

    data = {'ontap_credential': {'password': config['cluster'][
'ontap_admin_password']}}
    deploy.post('/clusters/{}/deploy?inhibit_rollback=true'.format
(cluster_id),
                data, wait_for_job=True)


def log_debug_trace():
    stack = traceback.extract_stack()
    parent_function = stack[-2][2]
    logging.getLogger('deploy').debug('Calling %s()' % parent_function)


def log_info(msg):
    logging.getLogger('deploy').info(msg)


def log_and_exit(msg):
    logging.getLogger('deploy').error(msg)
    exit(1)


def configure_logging(verbose):
    FORMAT = '%(asctime)-15s:%(levelname)s:%(name)s: %(message)s'
    if verbose:
        logging.basicConfig(level=logging.DEBUG, format=FORMAT)
    else:
        logging.basicConfig(level=logging.INFO, format=FORMAT)
        logging.getLogger('requests.packages.urllib3.connectionpool'
).setLevel(
            logging.WARNING)


def main(args):
    configure_logging(args.verbose)
    deploy = DeployRequests(args.deploy, args.password)

    with open(args.config_file) as json_data:
        config = json.load(json_data)

        add_vcenter_credentials(deploy, config)

        add_standalone_host_credentials(deploy, config)

        register_unkown_hosts(deploy, config)
```

```
        cluster_id = create_cluster_config(deploy, config)

        deploy_cluster(deploy, cluster_id, config)


def parseArgs():
    parser = argparse.ArgumentParser(description='Uses the ONTAP Select
Deploy API to construct and deploy a cluster.')
    parser.add_argument('-d', '--deploy', help='Hostname or IP address of
Deploy server')
    parser.add_argument('-p', '--password', help='Admin password of Deploy
server')
    parser.add_argument('-c', '--config_file', help='Filename of the
cluster config')
    parser.add_argument('-v', '--verbose', help='Display extra debugging
messages for seeing exact API calls and responses',
                        action='store_true', default=False)
    return parser.parse_args()

if __name__ == '__main__':
    args = parseArgs()
    main(args)
```

ONTAP Select クラスタを作成するスクリプト用の JSON

Python コード サンプルを使用して ONTAP Select クラスタを作成または削除する場合は、スクリプトへの入力として JSON ファイルを指定する必要があります。導入計画に基づいて、適切な JSON サンプルをコピーして変更できます。

ESXi 上のシングルノード クラスタ

```
{
  "hosts": [
    {
      "password": "mypassword1",
      "name": "host-1234",
      "type": "ESX",
      "username": "admin"
    }
  ],

  "cluster": {
    "dns_info": {
      "domains": ["lab1.company-demo.com", "lab2.company-demo.com",
```

```
        "lab3.company-demo.com", "lab4.company-demo.com"
        ],

      "dns_ips": ["10.206.80.135", "10.206.80.136"]
      },
      "ontap_image_version": "9.7",
      "gateway": "10.206.80.1",
      "ip": "10.206.80.115",
      "name": "mycluster",
      "ntp_servers": ["10.206.80.183", "10.206.80.142"],
      "ontap_admin_password": "mypassword2",
      "netmask": "255.255.254.0"
  },

  "nodes": [
    {
      "serial_number": "3200000nn",
      "ip": "10.206.80.114",
      "name": "node-1",
      "networks": [
        {
          "name": "ontap-external",
          "purpose": "mgmt",
          "vlan": 1234
        },
        {
          "name": "ontap-external",
          "purpose": "data",
          "vlan": null
        },
        {
          "name": "ontap-internal",
          "purpose": "internal",
          "vlan": null
        }
      ],
      "host_name": "host-1234",
      "is_storage_efficiency_enabled": false,
      "instance_type": "small",
      "storage": {
        "disk": [],
        "pools": [
          {
            "name": "storage-pool-1",
            "capacity": 4802666790125
          }
```

```
        ]
      }
    }
  ]
}
```

vCenterを使用した**ESXi**上のシングルノード クラスタ

```
{
  "hosts": [
    {
      "name":"host-1234",
      "type":"ESX",
      "mgmt_server":"vcenter-1234"
    }
  ],

  "cluster": {
    "dns_info": {"domains": ["lab1.company-demo.com", "lab2.company-
demo.com",
      "lab3.company-demo.com", "lab4.company-demo.com"
      ],
      "dns_ips": ["10.206.80.135","10.206.80.136"]
  },

  "ontap_image_version":"9.7",
  "gateway":"10.206.80.1",
  "ip":"10.206.80.115",
  "name":"mycluster",
  "ntp_servers": ["10.206.80.183","10.206.80.142"],
  "ontap_admin_password":"mypassword2",
  "netmask":"255.255.254.0"
  },

  "vcenter": {
    "password":"mypassword2",
    "hostname":"vcenter-1234",
    "username":"selectadmin"
  },

  "nodes": [
    {
      "serial_number": "3200000nn",
      "ip":"10.206.80.114",
      "name":"node-1",
```

```
      "networks": [
        {
          "name":"ONTAP-Management",
          "purpose":"mgmt",
          "vlan":null
        },
        {
          "name": "ONTAP-External",
          "purpose":"data",
          "vlan":null
        },
        {
          "name": "ONTAP-Internal",
          "purpose":"internal",
          "vlan":null
        }
      ],

      "host_name": "host-1234",
      "is_storage_efficiency_enabled": false,
      "instance_type": "small",
      "storage": {
        "disk":[],
        "pools": [
          {
            "name": "storage-pool-1",
            "capacity":5685190380748
          }
        ]
      }
    }
  ]
}
```

KVM 上のシングルノード クラスタ

```
{
  "hosts": [
    {
      "password": "mypassword1",
      "name":"host-1234",
      "type":"KVM",
      "username":"root"
    }
  ],
```

```
  "cluster": {
    "dns_info": {
      "domains": ["lab1.company-demo.com", "lab2.company-demo.com",
        "lab3.company-demo.com", "lab4.company-demo.com"
      ],

      "dns_ips": ["10.206.80.135", "10.206.80.136"]
    },

    "ontap_image_version": "9.7",
    "gateway":"10.206.80.1",
    "ip":"10.206.80.115",
    "name":"CBF4ED97",
    "ntp_servers": ["10.206.80.183", "10.206.80.142"],
    "ontap_admin_password": "mypassword2",
    "netmask":"255.255.254.0"
  },
  "nodes": [
    {
      "serial_number":"3200000nn",
      "ip":"10.206.80.115",
      "name": "node-1",
      "networks": [
        {
          "name": "ontap-external",
          "purpose": "mgmt",
          "vlan":1234
        },
        {
          "name": "ontap-external",
          "purpose": "data",
          "vlan": null
        },
        {
          "name": "ontap-internal",
          "purpose": "internal",
          "vlan": null
        }
      ],

      "host_name": "host-1234",
      "is_storage_efficiency_enabled": false,
      "instance_type": "small",
      "storage": {
        "disk": [],
```

```
        "pools": [
          {
            "name": "storage-pool-1",
            "capacity": 4802666790125
          }
        ]
      }
    }
  ]
}
```

ONTAP Selectノードライセンスを追加するスクリプト

次のスクリプトを使用して、ONTAP Selectノードのライセンスを追加できます。

```
#!/usr/bin/env python
##-----------------------------------------------------------------------
#
# File: add_license.py
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-----------------------------------------------------------------------

import argparse
import logging
import json

from deploy_requests import DeployRequests


def post_new_license(deploy, license_filename):
    log_info('Posting a new license: {}'.format(license_filename))

    # Stream the file as multipart/form-data
```

```
    deploy.post('/licensing/licenses', data={},
                files={'license_file': open(license_filename, 'rb')})

    # Alternative if the NLF license data is converted to a string.
    # with open(license_filename, 'rb') as f:
    #    nlf_data = f.read()
    #    r = deploy.post('/licensing/licenses', data={},
    #                    files={'license_file': (license_filename,
nlf_data)})


def put_license(deploy, serial_number, data, files):
    log_info('Adding license for serial number: {}'.format(serial_number))

    deploy.put('/licensing/licenses/{}'.format(serial_number), data=data,
files=files)


def put_used_license(deploy, serial_number, license_filename,
ontap_username, ontap_password):
    ''' If the license is used by an 'online' cluster, a username/password
must be given. '''

    data = {'ontap_username': ontap_username, 'ontap_password':
ontap_password}
    files = {'license_file': open(license_filename, 'rb')}

    put_license(deploy, serial_number, data, files)


def put_free_license(deploy, serial_number, license_filename):
    data = {}
    files = {'license_file': open(license_filename, 'rb')}

    put_license(deploy, serial_number, data, files)


def get_serial_number_from_license(license_filename):
    ''' Read the NLF file to extract the serial number '''
    with open(license_filename) as f:
        data = json.load(f)

        statusResp = data.get('statusResp', {})
        serialNumber = statusResp.get('serialNumber')
        if not serialNumber:
            log_and_exit("The license file seems to be missing the
serialNumber")
```

```
        return serialNumber


def log_info(msg):
    logging.getLogger('deploy').info(msg)


def log_and_exit(msg):
    logging.getLogger('deploy').error(msg)
    exit(1)


def configure_logging():
    FORMAT = '%(asctime)-15s:%(levelname)s:%(name)s: %(message)s'
    logging.basicConfig(level=logging.INFO, format=FORMAT)
    logging.getLogger('requests.packages.urllib3.connectionpool').
setLevel(logging.WARNING)


def main(args):
    configure_logging()
    serial_number = get_serial_number_from_license(args.license)

    deploy = DeployRequests(args.deploy, args.password)

    # First check if there is already a license resource for this serial-
number
    if deploy.find_resource('/licensing/licenses', 'id', serial_number):

        # If the license already exists in the Deploy server, determine if
its used
        if deploy.find_resource('/clusters', 'nodes.serial_number',
serial_number):

            # In this case, requires ONTAP creds to push the license to
the node
            if args.ontap_username and args.ontap_password:
                put_used_license(deploy, serial_number, args.license,
                                 args.ontap_username, args.ontap_password)
            else:
                print("ERROR: The serial number for this license is in
use. Please provide ONTAP credentials.")
        else:
            # License exists, but its not used
            put_free_license(deploy, serial_number, args.license)
```

```
    else:
        # No license exists, so register a new one as an available license
for later use
        post_new_license(deploy, args.license)


def parseArgs():
    parser = argparse.ArgumentParser(description='Uses the ONTAP Select
Deploy API to add or update a new or used NLF license file.')
    parser.add_argument('-d', '--deploy', required=True, type=str, help=
'Hostname or IP address of ONTAP Select Deploy')
    parser.add_argument('-p', '--password', required=True, type=str, help
='Admin password of Deploy server')
    parser.add_argument('-l', '--license', required=True, type=str, help=
'Filename of the NLF license data')
    parser.add_argument('-u', '--ontap_username', type=str,
                        help='ONTAP Select username with privelege to add
the license. Only provide if the license is used by a Node.')
    parser.add_argument('-o', '--ontap_password', type=str,
                        help='ONTAP Select password for the
ontap_username. Required only if ontap_username is given.')
    return parser.parse_args()

if __name__ == '__main__':
    args = parseArgs()
    main(args)
```

ONTAP Select クラスタを削除するスクリプト

既存のクラスタを削除するには、次のCLIスクリプトを使用できます。

```
#!/usr/bin/env python
##-----------------------------------------------------------------------
#
# File: delete_cluster.py
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
```

```
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##----------------------------------------------------------------------

import argparse
import json
import logging

from deploy_requests import DeployRequests

def find_cluster(deploy, cluster_name):
    return deploy.find_resource('/clusters', 'name', cluster_name)


def offline_cluster(deploy, cluster_id):
    # Test that the cluster is online, otherwise do nothing
    response = deploy.get('/clusters/{}?fields=state'.format(cluster_id))
    cluster_data = response.json()['record']
    if cluster_data['state'] == 'powered_on':
        log_info("Found the cluster to be online, modifying it to be
powered_off.")
        deploy.patch('/clusters/{}'.format(cluster_id), {'availability':
'powered_off'}, True)


def delete_cluster(deploy, cluster_id):
    log_info("Deleting the cluster({}).".format(cluster_id))
    deploy.delete('/clusters/{}'.format(cluster_id), True)
    pass


def log_info(msg):
    logging.getLogger('deploy').info(msg)


def configure_logging():
    FORMAT = '%(asctime)-15s:%(levelname)s:%(name)s: %(message)s'
    logging.basicConfig(level=logging.INFO, format=FORMAT)
    logging.getLogger('requests.packages.urllib3.connectionpool').
setLevel(logging.WARNING)


def main(args):
    configure_logging()
    deploy = DeployRequests(args.deploy, args.password)
```

```
    with open(args.config_file) as json_data:
        config = json.load(json_data)

        cluster_id = find_cluster(deploy, config['cluster']['name'])

        log_info("Found the cluster {} with id: {}.".format(config[
'cluster']['name'], cluster_id))

        offline_cluster(deploy, cluster_id)

        delete_cluster(deploy, cluster_id)


def parseArgs():
    parser = argparse.ArgumentParser(description='Uses the ONTAP Select
Deploy API to delete a cluster')
    parser.add_argument('-d', '--deploy', required=True, type=str, help=
'Hostname or IP address of Deploy server')
    parser.add_argument('-p', '--password', required=True, type=str, help
='Admin password of Deploy server')
    parser.add_argument('-c', '--config_file', required=True, type=str,
help='Filename of the cluster json config')
    return parser.parse_args()

if __name__ == '__main__':
    args = parseArgs()
    main(args)
```

ONTAP Select の共通サポート Python モジュール

すべてのPythonスクリプトは、単一のモジュール内で共通のPythonクラスを使用しています。

```
#!/usr/bin/env python
##-----------------------------------------------------------------------
#
# File: deploy_requests.py
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
```

```
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-----------------------------------------------------------------------

import json
import logging
import requests

requests.packages.urllib3.disable_warnings()

class DeployRequests(object):
    '''
    Wrapper class for requests that simplifies the ONTAP Select Deploy
    path creation and header manipulations for simpler code.
    '''

    def __init__(self, ip, admin_password):
        self.base_url = 'https://{}/api'.format(ip)
        self.auth = ('admin', admin_password)
        self.headers = {'Accept': 'application/json'}
        self.logger = logging.getLogger('deploy')

    def post(self, path, data, files=None, wait_for_job=False):
        if files:
            self.logger.debug('POST FILES:')
            response = requests.post(self.base_url + path,
                                     auth=self.auth, verify=False,
                                     files=files)
        else:
            self.logger.debug('POST DATA: %s', data)
            response = requests.post(self.base_url + path,
                                     auth=self.auth, verify=False,
                                     json=data,
                                     headers=self.headers)

        self.logger.debug('HEADERS: %s\nBODY: %s', self.filter_headers
(response), response.text)
        self.exit_on_errors(response)

        if wait_for_job and response.status_code == 202:
            self.wait_for_job(response.json())
        return response
```

```
    def patch(self, path, data, wait_for_job=False):
        self.logger.debug('PATCH DATA: %s', data)
        response = requests.patch(self.base_url + path,
                                  auth=self.auth, verify=False,
                                  json=data,
                                  headers=self.headers)
        self.logger.debug('HEADERS: %s\nBODY: %s', self.filter_headers
(response), response.text)
        self.exit_on_errors(response)

        if wait_for_job and response.status_code == 202:
            self.wait_for_job(response.json())
        return response

    def put(self, path, data, files=None, wait_for_job=False):
        if files:
            print('PUT FILES: {}'.format(data))
            response = requests.put(self.base_url + path,
                                    auth=self.auth, verify=False,
                                    data=data,
                                    files=files)
        else:
            self.logger.debug('PUT DATA:')
            response = requests.put(self.base_url + path,
                                    auth=self.auth, verify=False,
                                    json=data,
                                    headers=self.headers)

        self.logger.debug('HEADERS: %s\nBODY: %s', self.filter_headers
(response), response.text)
        self.exit_on_errors(response)

        if wait_for_job and response.status_code == 202:
            self.wait_for_job(response.json())
        return response

    def get(self, path):
        """ Get a resource object from the specified path """
        response = requests.get(self.base_url + path, auth=self.auth,
verify=False)
        self.logger.debug('HEADERS: %s\nBODY: %s', self.filter_headers
(response), response.text)
        self.exit_on_errors(response)
        return response

    def delete(self, path, wait_for_job=False):
```

```
        """ Delete's a resource from the specified path """
        response = requests.delete(self.base_url + path, auth=self.auth,
verify=False)
        self.logger.debug('HEADERS: %s\nBODY: %s', self.filter_headers
(response), response.text)
        self.exit_on_errors(response)

        if wait_for_job and response.status_code == 202:
            self.wait_for_job(response.json())
        return response

    def find_resource(self, path, name, value):
        ''' Returns the 'id' of the resource if it exists, otherwise None
'''
        resource = None
        response = self.get('{path}?{field}={value}'.format(
                            path=path, field=name, value=value))
        if response.status_code == 200 and response.json().get(
'num_records') >= 1:
            resource = response.json().get('records')[0].get('id')
        return resource

    def get_num_records(self, path, query=None):
        ''' Returns the number of records found in a container, or None on
error '''
        resource = None
        query_opt = '?{}'.format(query) if query else ''
        response = self.get('{path}{query}'.format(path=path, query
=query_opt))
        if response.status_code == 200 :
            return response.json().get('num_records')
        return None

    def resource_exists(self, path, name, value):
        return self.find_resource(path, name, value) is not None

    def wait_for_job(self, response, poll_timeout=120):
        last_modified = response['job']['last_modified']
        job_id = response['job']['id']

        self.logger.info('Event: ' + response['job']['message'])

        while True:
            response = self.get('/jobs/{}?fields=state,message&'
                                'poll_timeout={}&last_modified=>={}'
.format(
```

```
                                    job_id, poll_timeout, last_modified))

            job_body = response.json().get('record', {})

            # Show interesting message updates
            message = job_body.get('message', '')
            self.logger.info('Event: ' + message)

            # Refresh the last modified time for the poll loop
            last_modified = job_body.get('last_modified')

            # Look for the final states
            state = job_body.get('state', 'unknown')
            if state in ['success', 'failure']:
                if state == 'failure':
                    self.logger.error('FAILED background job.\nJOB: %s',
job_body)
                    exit(1)    # End the script if a failure occurs
                break

    def exit_on_errors(self, response):
        if response.status_code >= 400:
            self.logger.error('FAILED request to URL: %s\nHEADERS: %s
\nRESPONSE BODY: %s',
                              response.request.url,
                              self.filter_headers(response),
                              response.text)
        response.raise_for_status()    # Displays the response error, and
exits the script

    @staticmethod
    def filter_headers(response):
        ''' Returns a filtered set of the response headers '''
        return {key: response.headers[key] for key in ['Location',
'request-id'] if key in response.headers}
```

ONTAP Select クラスタノードのサイズを変更するスクリプト

次のスクリプトを使用して、ONTAP Select クラスタ内のノードのサイズを変更できます。

```
#!/usr/bin/env python
##-----------------------------------------------------------------------
#
# File: resize_nodes.py
```

```
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##------------------------------------------------------------------------

import argparse
import logging
import sys

from deploy_requests import DeployRequests


def _parse_args():
    """ Parses the arguments provided on the command line when executing
this
        script and returns the resulting namespace. If all required
arguments
        are not provided, an error message indicating the mismatch is
printed and
        the script will exit.
    """

    parser = argparse.ArgumentParser(description=(
        'Uses the ONTAP Select Deploy API to resize the nodes in the
cluster.'
        ' For example, you might have a small (4 CPU, 16GB RAM per node) 2
node'
        ' cluster and wish to resize the cluster to medium (8 CPU, 64GB
RAM per'
        ' node). This script will take in the cluster details and then
perform'
        ' the operation and wait for it to complete.'
    ))
    parser.add_argument('--deploy', required=True, help=(
        'Hostname or IP of the ONTAP Select Deploy VM.'
    ))
```

```
    parser.add_argument('--deploy-password', required=True, help=(
        'The password for the ONTAP Select Deploy admin user.'
    ))
    parser.add_argument('--cluster', required=True, help=(
        'Hostname or IP of the cluster management interface.'
    ))
    parser.add_argument('--instance-type', required=True, help=(
        'The desired instance size of the nodes after the operation is
complete.'
    ))
    parser.add_argument('--ontap-password', required=True, help=(
        'The password for the ONTAP administrative user account.'
    ))
    parser.add_argument('--ontap-username', default='admin', help=(
        'The username for the ONTAP administrative user account. Default:
admin.'
    ))
    parser.add_argument('--nodes', nargs='+', metavar='NODE_NAME', help=(
        'A space separated list of node names for which the resize
operation'
        ' should be performed. The default is to apply the resize to all
nodes in'
        ' the cluster. If a list of nodes is provided, it must be provided
in HA'
        ' pairs. That is, in a 4 node cluster, nodes 1 and 2 (partners)
must be'
        ' resized in the same operation.'
    ))
    return parser.parse_args()


def _get_cluster(deploy, parsed_args):
    """ Locate the cluster using the arguments provided """

    cluster_id = deploy.find_resource('/clusters', 'ip', parsed_args
.cluster)
    if not cluster_id:
        return None
    return deploy.get('/clusters/%s?fields=nodes' % cluster_id).json()[
'record']


def _get_request_body(parsed_args, cluster):
    """ Build the request body """

    changes = {'admin_password': parsed_args.ontap_password}
```

```
    # if provided, use the list of nodes given, else use all the nodes in
the cluster
    nodes = [node for node in cluster['nodes']]
    if parsed_args.nodes:
        nodes = [node for node in nodes if node['name'] in parsed_args
.nodes]

    changes['nodes'] = [
        {'instance_type': parsed_args.instance_type, 'id': node['id']} for
node in nodes]

    return changes


def main():
    """ Set up the resize operation by gathering the necessary data and
then send
        the request to the ONTAP Select Deploy server.
    """

    logging.basicConfig(
        format='[%(asctime)s] [%(levelname)5s] %(message)s', level=
logging.INFO,)

    logging.getLogger('requests.packages.urllib3').setLevel(logging
.WARNING)

    parsed_args = _parse_args()
    deploy = DeployRequests(parsed_args.deploy, parsed_args
.deploy_password)

    cluster = _get_cluster(deploy, parsed_args)
    if not cluster:
        deploy.logger.error(
            'Unable to find a cluster with a management IP of %s' %
parsed_args.cluster)
        return 1

    changes = _get_request_body(parsed_args, cluster)
    deploy.patch('/clusters/%s' % cluster['id'], changes, wait_for_job
=True)

if __name__ == '__main__':
    sys.exit(main())
```

著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用権を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用権については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、http://www.netapp.com/TMに記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。