# RESTによる自動化

## ONTAP Select

NetApp
February 19, 2025

# 目次

# RESTによる自動化

## 概念

### 基盤となるREST Webサービス

Representational State Transfer（REST）は、分散Webアプリケーションを作成するための形式です。WebサービスAPIの設計に適用すると、サーバベースのリソースを公開して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が使用されており、 ONTAP Select クラスタの導入と管理のための柔軟な基盤が提供されます。

#### アーキテクチャと従来の制約

レストは2000年にカリフォルニア大学アーバイン校で博士号を取得したロイ・フィールディングによって正式に明言された "失策"。アーキテクチャスタイルは、一連の制約によって定義されます。これらの制約は、全体的に 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。制約は、ステートレス通信プロトコルを使用するクライアント / サーバアーキテクチャに基づいて、 RESTful Web サービスアプリケーションを確立するものです。

#### 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際の初期設計タスクには、次のものがあります。

- すべてのシステムが使用するシステムまたはサーバベースのリソースを識別し、リソースを維持します。リソースには、ファイル、ビジネストランザクション、プロセス、または管理エンティティがあります。REST Webサービスに基づいて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。
- リソースの状態の定義と関連する状態処理リソースは、常に有限数の状態のいずれかになります。状態、および状態の変化に影響を与えるために使用される関連操作は、明確に定義する必要があります。

クライアントとサーバの間でメッセージを交換しながら、一般的な CRUD （ Create 、 Read 、 Update 、 Delete ）モデルに従ってリソースにアクセスしてその状態を変更します。

#### 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および利用可能にする必要があります。リソースが配置され、識別されるエンドポイントでは、Uniform Resource Identifier（URI）が使用されます。URIは、ネットワーク内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を提供します。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

Hypertext Transfer Protocol（HTTP）は、Webサービスのクライアントとサーバがリソースに関する要求と応答の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）はリソースおよび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、関連する一連の要求と応答を 1 つのトランザクションで関連付けるには、要求 / 応答のデータフローで伝送される 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の間で情報を構造化して転送する方法は複数ありますが、最も広く使用されている方法（ Deploy REST API で使用）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。

## Deploy API へのアクセス方法

REST Web サービスは柔軟性が高いため、 ONTAP Select Deploy API にはいくつかの方法でアクセスできます。

**Deploy** ユーティリティの標準のユーザインターフェイス

ONTAP Select Deploy の Web ユーザインターフェイスから API にアクセスする場合は、主に API にアクセスします。ブラウザは、 API を呼び出して、ユーザインターフェイスの設計に従ってデータを再フォーマットします。また、 Deploy ユーティリティのコマンドラインインターフェイスから API にアクセスします。

**ONTAP Select Deploy** のオンラインドキュメントページです

ONTAP Select Deploy のオンラインドキュメントページでは、ブラウザを使用する際に別のアクセスポイントを使用できます。個々の API 呼び出しを直接実行する方法に加え、各呼び出しの入力パラメータやその他のオプションなど、 API の詳細な概要 も含まれています。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

さまざまなプログラミング言語やツールを使用して Deploy API にアクセスできます。広く利用されているのは、 Python 、 Java 、 cURL などです。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 サービスの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と、 API をより詳しく理解し、 ONTAP Select の導入を自動化することができます。

## API のバージョン管理を導入

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関係ありません。Deploy のリリースに含まれている API のバージョンと、 API の使用にどのような影響があるかを確認しておく必要があります。

Deploy 管理ユーティリティの最新リリースには、バージョン 3 の REST API が含まれています。Deploy ユーティリティの以前のリリースには、次のバージョンの API が含まれていました。

**2.8** 以降を導入します

ONTAP Select Deploy 2.8 以降のすべてのリリースには、 REST API のバージョン 3 が含まれています。

**2.7.2** 以前のバージョンを導入します

ONTAP Select Deploy 2.7.2 およびそれ以前のすべてのリリースには、 REST API のバージョン 2 が含まれています。

REST API のバージョン 2 と 3 には互換性がありません。バージョン 2 の API を含む以前のリリースから Deploy 2.8 以降にアップグレードする場合は、コマンドラインインターフェイスを使用して、 API に直接アクセスする既存のコードおよびスクリプトを更新する必要があります。

## 基本的な動作特性

RESTでは共通のテクノロジとベストプラクティスが確立されますが、各APIの詳細は設計の選択内容によって異なります。API を使用する前に、 ONTAP Select Deploy API の詳細と運用上の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** ノードです

a_hypervisor host_ 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 仮想マシンを導入してハイパーバイザーホストでアクティブにすると、その仮想マシンは _ ONTAP Select node_name 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々になります。これにより、 1 つ以上の ONTAP Select ノードを同じハイパーバイザーホストで実行できる 1 対多の関係が実現します。

### オフシエクト**ID**

各リソースインスタンスまたはオブジェクトには、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を呼び出したあと、関連付けられているIDの値がHTTP応答のヘッダーで呼び出し元に返され `location`ます。識別子を抽出して以降の呼び出しでリソースインスタンスを参照する際に使用できます。

オブジェクトIDの内容と内部構造はいつでも変更できます。識別子は、関連するオブジェクトを参照する場合にのみ、該当するAPI呼び出しで使用してください。

### 要求 **ID**

成功したすべての API 要求には、一意の識別子が割り当てられます。識別子は、関連付けられたHTTP応答のヘッダーで返され `request-id`ます。要求 ID を使用すると、単一の API 要求と応答のトランザクションのアクティビティをまとめて参照できます。たとえば、要求 ID に基づいて、トランザクションの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

サーバがクライアントから受信した HTTP 要求を実行する主な方法は 2 つあります。

- 同期サーバは要求をただちに実行し、ステータスコード 200 、 201 、または 204 で応答します。
- 非同期サーバは要求を受け入れ、ステータスコード 202 で応答します。これは、サーバがクライアント要求を受け入れ、要求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ます。最終的な成功または失敗はすぐには確認できないため、追加の API 呼び出しで確認する必要があります。

### 長時間実行されているジョブの完了を確認する

通常、完了までに時間がかかる操作は、サーバでバックグラウンドタスクを使用して非同期に処理されます。Deploy REST API では、タスクを追跡し、現在の状態などの情報を提供するジョブオブジェクトによっ

てすべてのバックグラウンドタスクがアンカーで設定されます。一意の識別子を含むジョブオブジェクトは、バックグラウンドタスクが作成されたあとの HTTP 応答で返されます。

ジョブオブジェクトを直接照会することで、関連する API 呼び出しの成功または失敗を確認できます。ジョブ object_for 追加情報 を使用した非同期処理を参照してください。

ジョブオブジェクトを使用する以外に、要求の成功または失敗を判断する方法がいくつかあります。これには次のものがあります。

- イベントメッセージ特定の API 呼び出しに関連するすべてのイベントメッセージを取得するには、元の応答で返された要求 ID を使用します。通常、イベントメッセージには成功または失敗の兆候が含まれており、エラー状態のデバッグ時にも役立ちます。
- リソースの状態または状態いくつかのリソースには状態またはステータスの値が保持されており、リクエストの成功または失敗を間接的に判断するためにクエリできます。

#### セキュリティ

Deploy API では、次のセキュリティテクノロジを使用します。

- Transport Layer Security ： Deploy サーバとクライアントの間でネットワークを介して送信されるすべてのトラフィックは、 TLS を介して暗号化されます。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 がサポートされています。
- HTTP 認証：すべての API トランザクションにベーシック認証が使用されます。base64 文字列のユーザ名とパスワードを含む HTTP ヘッダーがすべての要求に追加されます。

## 要求と応答の**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 要求として実行され、クライアントへの関連する応答が生成されます。この要求と応答のペアは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、要求の制御に使用できる入力変数と応答出力の内容を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

API 呼び出しの処理方法は、 HTTP 要求で設定されたパラメータを使用して制御できます。

#### 要求ヘッダー

HTTP 要求には、次のようなヘッダーを含める必要があります。

- content-type 要求の本文に JSON が含まれている場合は、このヘッダーを application/json に設定する必要があります。
- 応答の本文に JSON が含まれる場合は受け入れます。このヘッダーを application/json に設定する必要があります。
- 認証基本認証は、 base64 文字列でエンコードされたユーザ名とパスワードを使用して設定する必要があります。

### リクエストの本文

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。

- JSON オブジェクトと入力変数（新しいクラスタの名前など）
- 空

### オブジェクトのフィルタ

GETを使用するAPI呼び出しを実行するときに、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リングできます。たとえば、次のように完全に一致する値を指定できます。

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

完全一致に加えて、他の演算子を使用して、一連のオブジェクトを一定範囲の値で返すことができます。ONTAP Select では、次のフィルタ演算子がサポートされています。

| 運用者 | 製品説明 |
|---|---|
| = | 等しい |
| < | より小さい |
| > | 次の値より大きい |
| ⇐ | 以下 |
| >= | 以上 |
| | または |
| なんだ | 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して、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を使用してAPI呼び出しを実行すると、オブジェクトを一意に識別する属性のみが返されます。このフィールドの最小セットは、各オブジェクトのキーとして機能し、オブジェクトタイプによって異なります。fields query パラメータを使用すると、次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。

- 安価なフィールドローカルサーバメモリに保持されているオブジェクトフィールドを取得するか、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指定します `fields=*`。
- 高価なフィールドは、アクセスするために追加のサーバ処理が必要なフィールドも含め、すべてのオブジェクトフィールドを取得するように指定します `fields=**`。
- カスタムフィールドの選択目的のフィールドを正確に指定するために使用し `fields=FIELDNAME`ます。複数のフィールドを要求する場合は、値をカンマで区切ってスペースなしで指定する必要があります。

ベストプラクティスとして、必要なフィールドを常に個別に指定することを推奨します。安価なフィールドや高コストのフィールドは、必要な場合にのみ取得してください。低コストでコストな分類は、ネットアップが社内パフォーマンス分析に基づいて決定します。特定のフィールドの分類は、いつでも変更できます。

出力セット内のオブジェクトをソートする

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。次のように'フィールド名とソート方向を指定したORDER_BYクエリパラメータを使用して順序を変更できます
`order_by=<field name> asc|desc`

たとえば、typeフィールドを降順に並べ替え、idを昇順に並べ替えることができます。
`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。

ページ付け

GET を使用する API 呼び出しを発行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場合、一致するすべてのオブジェクトがデフォルトで返されます。必要に応じて、 max_records クエリパラメータを要求とともに使用して返されるレコード数を制限することもできます。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。たとえば、次の例では、指定した時間が経過したあとに生成されたシステムイベントが最大10個返されます。
`time⇒ 2019-04-04T15:41:29.140265Z&max_records=10`

複数の要求を問題 で送信して、各イベント（または任意のオブジェクトタイプ）をページングできます。以降の各API呼び出しでは、最後の結果セットの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 応答を解釈します

各API要求でクライアントへの応答が生成されます。応答を調べて成功したかどうかを確認し、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次に示します。

| コード | 意味 | 製品説明 |
|---|---|---|
| 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 | 作成済み | オブジェクトが作成されました。場所の応答ヘッダーにはオブジェクトの一意の識別子が含まれています。 |
| 202 | 承認済み | 長時間のバックグラウンドジョブで要求の実行が開始されましたが、処理はまだ完了していません。 |
| 400 | 無効な要求です | 要求の入力が認識されないか不適切です。 |

| コード | 意味 | 製品説明 |
| --- | --- | --- |
| 403 | 禁止 | 認証エラーによりアクセスが拒否されました。 |
| 404 | 見つかりません | 要求で参照されているリソースが存在しません。 |
| 405 | 許可されていないメソッド | 要求内の HTTP 動詞はリソースで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 | 実装されていません | URI は既知ですが、要求を実行できません。 |

応答ヘッダー

Deploy サーバによって生成される HTTP 応答には、次のようなヘッダーが含まれています。

- request-id 成功したすべての API 要求には、一意の要求 ID が割り当てられます。
- Location ：オブジェクトが作成されると、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 が格納されます。

応答の本文

API 要求に関連する応答の内容は、オブジェクト、処理タイプ、および要求の成功または失敗によって異なります。応答の本文は JSON 形式になります。

- 単一のオブジェクト単一のオブジェクトを要求に基づいて一連のフィールドとともに返すことができます。たとえば、GETを使用すると、一意の識別子を使用してクラスタの選択したプロパティを取得できます。
- 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、オブジェクトインスタンスの配列を含むレコードとレコードの数を示す一貫した形式が使用され `num_records` ます。たとえば、特定のクラスタに定義されているすべてのノードを取得できます。
- ジョブオブジェクト API 呼び出しが非同期で処理されると、バックグラウンドタスクのアンカーを設定するジョブオブジェクトが返されます。たとえば、クラスタの導入に使用された POST 要求は非同期で処理され、ジョブ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、常にエラーオブジェクトが返されます。たとえば、既存の名前を使用してクラスタを作成しようとするとエラーが表示されます。
- 空の場合もあります。データが返されず、応答の本文が空になっていることもあります。たとえば、DELETE を使用して既存のホストを削除したあとは、応答の本文が空になります。

## ジョブオブジェクトを使用した非同期処理

Deploy API 呼び出し、特にリソースの作成や変更を行う呼び出しは、他の呼び出しよりも完了に時間がかかることがあります。ONTAP Select Deploy は、これらの長時間実行される要求を非同期で処理します。

## ジョブオブジェクトを使用した非同期要求の説明

非同期で実行されるAPI呼び出しを実行すると、HTTP応答コード202が返されます。これは、要求が正常に検証されて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、一意の識別子を含む、要求をアンカーするジョブオブジェクトが含まれます。

非同期的に処理する API 呼び出しを決定するには、 ONTAP Select Deploy のオンラインドキュメントページを参照してください。

### API要求に関連付けられているジョブ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。

- キューに登録済み
- 実行中
- 成功
- 障害

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、次の2つの方法があります。

- 標準のポーリング要求の現在のジョブの状態がすぐに返されます
- ロングポーリング要求のジョブ状態は、次のいずれかの場合にのみ返されます。
  - 状態が、ポーリング要求で指定された日時の値よりも最近変更されました
  - タイムアウト値が期限切れ（ 1 ～ 120 秒）

標準のポーリングとロングポーリングでは、同じ API 呼び出しを使用してジョブオブジェクトが照会されます。ただし、長いポーリング要求には、と `last_modified` の2つのクエリパラメータが含まれています `poll_timeout`。

Deploy 仮想マシンのワークロードを減らすためには、常に長いポーリングを使用してください。

### 非同期要求を発行するための一般的な手順

非同期API呼び出しを完了する手順の概要は次のとおりです。

1. 非同期API呼び出しを実行します。
2. 要求が正常に受け入れられたことを示す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、各サイクルで次の手順を実行します。
   a. 長時間のポーリング要求で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（待機中、実行中）の場合は、もう一度ループを実行します。

5. ジョブが終了状態（ success または failure ）になったら停止します。

# ブラウザを使用してにアクセスします

## ブラウザから **API** にアクセスする前に

### Deploy のオンラインドキュメントページを使用する前に、いくつかの点に注意する必要があります。

導入計画

特定の導入タスクまたは管理タスクを実行する際に問題 API 呼び出しを行う場合は、導入計画を作成することを検討してください。これらのプランは正式なプランでも非公式なプランでもあり、通常は目標と使用する 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** の例とパラメータの定義

各 API 呼び出しについて、ドキュメントページで一貫した形式で説明しています。このコンテンツには、実装メモ、クエリパラメータ、および HTTP ステータスコードが含まれます。また、 API の要求と応答で使用される JSON に関する詳細を次のように表示することもできます。

- 値の例 API 呼び出しで _example value_on をクリックすると、呼び出しの典型的な JSON 構造が表示されます。この例は必要に応じて変更でき、要求の入力として使用できます。
- モデル (Model) - モデル (*Model*) をクリックすると 'JSON パラメータの完全なリストが表示され ' 各パラメータの概要 が表示されます

**API** 呼び出しを実行する際の注意事項

Deploy のドキュメントページを使用して実行する API 処理は、すべてライブ処理です。構成ファイルやその他のデータを誤って作成、更新、削除しないように、注意してください。

## **Deploy**のドキュメントページへのアクセス

### API ドキュメントを表示する場合や、 API 呼び出しを手動で問題 する場合は、 ONTAP Select Deploy の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

次の情報が必要です。

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

手順

1. ブラウザに URL を入力し、 **Enter** キーを押します。

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

2. 管理者のユーザ名とパスワードを使用してサインインします。

結果

Deploy のドキュメントの Web ページが表示され、ページの下部にカテゴリ別に分類された呼び出しが表示されます。

## API呼び出しの理解と実行

すべての API 呼び出しの詳細が、 ONTAP Select Deploy のオンラインドキュメント Web ページに共通の形式で文書化されて表示されます。1 つの API 呼び出しについて理解すれば、他の API 呼び出しの詳細も同様に表示して解釈できるようになります。

開始する前に

ONTAP Select Deploy のオンラインドキュメント 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。

タスクの内容

一意の識別子を使用して、 ONTAP Select クラスタについての設定情報を取得できます。この例では、 expensive 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、必要なフィールドだけを指定することを推奨します。

手順

1. メインページで一番下までスクロールし、 * Cluster * をクリックします。
2. Get / clusters / ｛ cluster_id ｝ * をクリックして、 ONTAP Select クラスタに関する情報の取得に使用する API 呼び出しの詳細を表示します。

# ワークフロープロセス

## API ワークフローを使用する前に

ワークフロープロセスを確認して使用するための準備をしておく必要があります。

ワークフローで使用されるAPI呼び出しについて理解する

ONTAP Select のオンラインドキュメントページに、すべての REST API 呼び出しの詳細が記載されています。ここでは、それらの詳細を繰り返すのではなく、ワークフローのサンプルで使用している各 API 呼び出しについて、その呼び出しをドキュメントページで見つけるために必要な情報だけを示しています。特定のAPI呼び出しを検索したら、入力パラメータ、出力形式、HTTPステータスコード、要求処理タイプなど、呼び出しのすべての詳細を確認できます。

ワークフロー内の各API呼び出しについて、ドキュメントページで呼び出しを見つけるのに役立つ次の情報が含まれています。

- カテゴリ：ドキュメントページでは、機能的な関連領域またはカテゴリ別に API 呼び出しが分類されています。特定のAPI呼び出しを検索するには、ページの一番下までスクロールし、該当するAPIカテゴリをクリックします。
- HTTP 動詞 HTTP 動詞は、リソースに対して実行する操作を示します。各API呼び出しは、単一のHTTP動詞を使用して実行されます。

- パス：このパスは、呼び出しの実行中に環境 が処理する特定のリソースを指定します。パス文字列がコアURLに追加され、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**を作成する

ONTAP Select のドキュメントページに加え、 Python などのプログラミング言語を使用して、 Deploy REST API に直接アクセスすることもできます。この場合のコア URL は、オンラインドキュメントページにアクセスするときに使用する URL とは少し異なります。API に直接アクセスする場合は、ドメインとポートの文字列に /api を追加する必要があります。例：
`http://deploy.mycompany.com/api`

## ワークフロー**1**：**ESXi**にシングルノードの評価クラスタを作成する

vCenter で管理されている VMware ESXi ホストにシングルノードの ONTAP Select クラスタを導入できます。クラスタは、評価用ライセンスで作成されます。

クラスタの作成ワークフローは次の場合に異なります。

- ESXi ホストが vCenter で管理されない（スタンドアロンホスト）
- クラスタ内で複数のノードまたはホストが使用されている場合
- クラスタは購入ライセンスを使用して本番環境に導入されます
- KVMハイパーバイザーは、VMware ESXiの代わりに使用されます

**1.vCenter Server**クレデンシャルの登録

vCenter サーバで管理されている ESXi ホストに導入する場合は、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。その後、 Deploy 管理ユーティリティは、のクレデンシャルを使用して vCenter への認証を行います。

| カテゴリ | **HTTP**動詞 | パス |
|---|---|---|
| 導入 | 投稿 | /security/ クレデンシャル |

カール

```
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（**step01**）

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

処理のタイプ

非同期

出力

- ロケーション応答ヘッダーのクレデンシャル ID
- ジョブオブジェクト

**2.**ハイパーバイザーホストを登録する

ONTAP Select ノードが含まれる仮想マシンを実行するハイパーバイザーホストを追加する必要があります。

| カテゴリ | **HTTP**動詞 | パス |
|---|---|---|
| クラスタ | 投稿 | /hosts |

カール

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

**JSON**入力（**step02**）

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

処理のタイプ

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

**3.**クラスタを作成

ONTAP Select クラスタを作成すると、基本的なクラスタ設定が登録され、 Deploy によってノード名が自動的に生成されます。

| カテゴリ | **HTTP**動詞 | パス |
|---|---|---|
| クラスタ | 投稿 | /clusters |

カール

シングルノードクラスタの場合、クエリパラメータの 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（**step03**）

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

処理のタイプ

同期

出力

- location 応答ヘッダーにクラスタ ID が含まれます

**4.**クラスタの設定

クラスタの設定の一環として指定する必要のある属性がいくつかあります。

| カテゴリ | **HTTP**動詞 | パス |
| --- | --- | --- |
| クラスタ | パッチ | / クラスタ / ｛ cluster_id ｝ |

カール

クラスタ ID を指定する必要があります。

```
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（**step04**）

```
{
  "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"}
}
```

処理のタイプ

同期

出力

なし

**5.**ノード名を取得する

Deploy 管理ユーティリティは、クラスタの作成時にノード ID と名前を自動的に生成します。ノードを設定する前に、割り当てられている 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 | 取得 | / クラスタ / ｛ cluster_id ｝ / ノード |

カール

クラスタ ID を指定する必要があります。

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

処理のタイプ

同期

出力

- Array は、それぞれ固有の ID と名前を持つ単一のノードを記述している

**6.**ノードを設定

ノードの基本設定を指定する必要があります。これは、ノードの設定に使用される最初の 3 つの API 呼び出しです。

| カテゴリ | HTTP動詞 | パス |
| --- | --- | --- |
| クラスタ | パス | / クラスタ / ｛cluster_id｝ / ノード / ｛node-id｝ |

カール

クラスタ ID とノード ID を指定する必要があります。

```
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（手順 **06** ）

ONTAP Select ノードを実行するホスト ID を指定する必要があります。

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

処理のタイプ

同期

出力

なし

**7.**ノードネットワークを取得

シングルノードクラスタ内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する必要があります。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 | 取得 | / クラスタ / ｛cluster_id｝ / ノード / ｛node-id｝ / ネットワーク |

カール

クラスタ ID とノード ID を指定する必要があります。

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

処理のタイプ

同期

出力

- 2 つのレコードの配列。各レコードは、一意の ID と目的を含め、ノードの単一のネットワークを表します

**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する必要があります。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。

問題 次の API 呼び出しは、ネットワークごとに 2 回ずつ実行されます。

| カテゴリ | **HTTP**動詞 | パス |
|---|---|---|
| クラスタ | パッチ | / クラスタ / ｛ cluster_id ｝ / ノード / ｛ node-id ｝ / ネットワーク / ｛ network_id ｝ |

カール

クラスタ ID 、ノード ID 、およびネットワーク ID を指定する必要があります。

```
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（**step08**）

ネットワークの名前を指定する必要があります。

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

処理のタイプ

同期

出力

なし

**9.**ノードのストレージプールを設定

ノードを設定する最後の手順は、ストレージプールを接続することです。使用可能なストレージプールは、vSphere Web Client を介して、または必要に応じて Deploy REST API を使用して確認できます。

| カテゴリ | HTTP動詞 | パス |
| --- | --- | --- |
| クラスタ | パッチ | / クラスタ / ｛cluster_id｝ / ノード / ｛node-id｝ / ネットワーク / ｛network_id｝ |

カール
　クラスタ ID 、ノード ID 、およびネットワーク ID を指定する必要があります。

```
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（**step09**）
　プールの容量は 2TB です。

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

処理のタイプ
　同期

出力
　なし

**10.**クラスタを導入

クラスタとノードの設定が完了したら、クラスタを導入できます。

| カテゴリ | HTTP動詞 | パス |
| --- | --- | --- |
| クラスタ | 投稿 | / クラスタ / ｛cluster_id｝ / 導入してください |

カール
　クラスタ ID を指定する必要があります。

```
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（手順 10 ）**

ONTAP 管理者アカウントのパスワードを指定する必要があります。

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

処理のタイプ

非同期

出力

- ジョブオブジェクト

# Python を使用してアクセスします

## Python を使用して API にアクセスする前に、次の手順を実行

サンプルの 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、環境が適切に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 がインストールされている必要があります。サンプルコードは python2 でテストされています。また、 Python3 に移植可能ですが、互換性についてはテストされていません。
- Requests ライブラリと Ullib3 ライブラリがインストールされている必要があります。環境に応じて、 pip などの Python 管理ツールを使用できます。
- スクリプトを実行するクライアントワークステーションに、 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。

また、次の情報が必要です。

- 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名とパスワード

## Pythonスクリプトを理解する

サンプルの Python スクリプトを使用すると、いくつかの異なるタスクを実行できます。スクリプトをライブ Deploy インスタンスで使用する前に、それらのスクリプトについて理解しておく必要があります。

共通の設計特性

スクリプトは、次の一般的な特性で設計されています。

- クライアントマシンでコマンドラインインターフェイスから実行する適切に設定された任意のクライアントマシンから Python スクリプトを実行できます。詳細については、を参照してください。
- CLI 入力パラメータの受け入れ各スクリプトは、入力パラメータを使用して CLI で制御されます。
- 読み取り入力ファイル各スクリプトは、その目的に基づいて入力ファイルを読み取ります。クラスタを作成または削除する場合は、 JSON 構成ファイルを指定する必要があります。ノードライセンスを追加する場合は、有効なライセンスファイルを指定する必要があります。
- 共通サポートモジュールを使用共通サポートモジュール _deploy_requests_py_contains a single classインポートされ、各スクリプトで使用されます。

クラスタを作成

スクリプト cluster.py を使用して、 ONTAP Select クラスタを作成できます。JSON 入力ファイルの CLI パラメータと内容に基づいて、次のようにスクリプトを導入環境に変更できます。

- ハイパーバイザーESXiまたはKVMに導入できます（Deployリリースによって異なります）。ESXi に導入する際、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にすることもできます。
- クラスタサイズ：シングルノードクラスタまたはマルチノードクラスタを導入できます。
- 評価用ライセンスまたは本番用ライセンス：本番環境用の評価用ライセンスまたは購入ライセンスを使用してクラスタを導入できます。

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
- admin ユーザアカウントのパスワード
- JSON 構成ファイルの名前
- メッセージ出力の詳細フラグ

ノードライセンスを追加

本番環境クラスタの導入を選択した場合は、 script_add_license.py__ を使用して各ノードの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の導入前または導入後に追加できます。

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
- admin ユーザ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ための権限を持つ ONTAP ユーザ名
- ONTAP ユーザのパスワード

クラスタを削除します

既存の ONTAP Select クラスタは、 script_delete_cluster.py _ を使用して削除できます。

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
- admin ユーザアカウントのパスワード
- JSON 構成ファイルの名前

# Python コードサンプル

クラスタを作成するスクリプト

次のスクリプトを使用して、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)
```

## クラスタを作成するスクリプトの 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** で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
          }
        ]
      }
    }
  ]
}
```

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)
```

### 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場合は、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)
```

## 共通サポートモジュール

すべての Python スクリプトは、 1 つのモジュールで共通の Python クラスを使用します。

```
#!/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}
```

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すると、 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



## 商標に関する情報