RESTで自動化
ONTAP Select

NetApp
January 29, 2026

目次

RESTで自動化

概念

ONTAP Selectクラスタの導入と管理のための REST Web サービス基盤

Representational State Transfer（REST）は、分散Webアプリケーションの作成に使用される形式です。WebサービスAPIの設計においては、これによってサーバベースのリソースの公開とその状態の管理に関する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クラスタを導入および管理するための柔軟な基盤を提供します。

建築と古典的な制約

RESTはRoy Fielding博士によって正式に発表されました。 "論文" 2000年にカリフォルニア大学アーバイン校で発表されました。一連の制約を通してアーキテクチャスタイルを定義し、Webベースのアプリケーションとその基盤となるプロトコルを総合的に改善します。これらの制約により、ステートレスな通信プロトコルを用いたクライアント/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構築されます。

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。リソースには、ファイル、ビジネス トランザクション、プロセス、管理エンティティなどがあります。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に行う最初の作業の1つは、リソースを識別することです。
- 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の数の状態のいずれかになります。状態は明確に定義する必要があり、状態の変化に作用する操作も明確に定義する必要があります。

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、変更するために、クライアントとサーバーの間でメッセージが交換されます。

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

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際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) が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、関連する要求と応答のセットを 1 つのトランザクションに関連付けるには、要求/応答データ 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

情報はさまざまな方法で構造化してクライアントとサーバー間で転送できますが、最も一般的なオプション (Deploy REST API で使用されるオプション) 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、リソースについての状態情報の転送に使用されます。

ONTAP Select Deploy API にアクセスする方法

REST Web サービスの本質的な柔軟性により、 ONTAP Select Deploy API にはさまざまな方法でアクセスできます。

デプロイユーティリティのネイティブユーザーインターフェース

APIにアクセスする主な方法は、 ONTAP Select DeployのWebユーザインターフェイスを使用することです。ブラウザはAPIを呼び出し、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のオンラインドキュメントページは、ブラウザ使用時の代替アクセスポイントとしてご利用いただけます。個々のAPI呼び出しを直接実行する方法に加えて、このページには、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の詳細な説明も記載されています。API呼び出しは、複数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、様々なプログラミング言語とツールからアクセスできます。Python、Java、cURLなどが一般的です。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をより深く理解し、 ONTAP Selectの導入を自動化する機会が得られます。

ONTAP Select Deploy APIのバージョン管理

ONTAP Select Deployに含まれるREST APIにはバージョン番号が割り当てられています。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。Deployのリリースに含まれるAPIのバージョンと、それがAPIの使用にどのような影響を与えるかを把握しておく必要があります。

Deploy管理ユーティリティの現在のリリースには、REST APIバージョン3が含まれています。Deployユーティリティの過去のリリースには、以下のAPIバージョンが含まれています。

2.8以降をデプロイする

ONTAP Select Deploy 2.8 以降のすべてのリリースには、REST API 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デプロイする

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 バージョン 2 が含まれています。

REST API バージョン 2 と 3 には互換性がありません。APIバージョン 2 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、API に直接アクセスする既存のコードと、コマンドラインインターフェースを使用するスクリプトをすべて更新する必要があります。

ONTAP Select Deploy API の基本的な動作特性

RESTで共通のテクノロジとベストプラクティスは確立されますが、各APIの詳細は設計内容に応じ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理解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。ONTAPONTAP Select仮想マシンがハイパーバイザーホストに導入され、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。DeployREST API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独立したオブジェクトです。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**

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それぞれ割り当てられます。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値が呼び出し元に返されます。`location` HTTP 応答のヘッダー。リソース インスタンスを以降の呼び出しで参照する際は、この識別子を抽出して使用できます。

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リクエストにはそれぞれ固有の識別子が割り当てられます。識別子は `request-id`関連するHTTPレスポンスのヘッダー。リクエストIDを使用すると、特定のAPIリクエスト・レスポンス・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。例えば、リクエストIDに基づいて、ある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

サーバー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 サーバーは要求を直ちに実行し、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ます。最終的な成功または失敗はすぐには確認できず、追加のAPI呼び出しを通じて判断す

る必要があります。

長時間実行ジョブの完了を確認する

通常、完了までに長時間かかる操作は、サーバー側で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。DeployREST APIでは、すべてのバックグラウンドタスクはジョブオブジェクトによってアンカーされ、このオブジェクトはタスクを追跡し、現在の状態などの情報を提供します。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ジョブオブジェクトがHTTPレスポンスで返されます。

ジョブオブジェクトを直接クエリすることで、関連するAPI呼び出しの成功または失敗を確認できます。詳細については、「ジョブ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、次のような方法でリクエストの成功または失敗を判断できます。

- 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を使用して、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、成功または失敗の情報が含まれ、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 リソースのいくつかは状態またはステータス値を維持しており、これを照会することで、要求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 は次のセキュリティ テクノロジを使用します。

- トランスポート層セキュリティ（TLS） デプロイ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でのHTTPプロトコルの使用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のリクエストおよびレスポンス API トランザクション

すべてのDeploy API呼び出しは、Deploy仮想マシンへのHTTPリクエストとして実行され、クライアントへのレスポンスが生成されます。Deployこの要求と応答のペアでAPIトランザクションが構成されます。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じて、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、次のようないくつか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 が含まれている場合、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。

- 認証 基本認証は、base64 文字列でエンコードされたユーザー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（新しいクラスターの名前など）を含む JSON オブジェクト
- 空の

フィルターオブジェクト

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、値の範囲に含まれるオブジェクトのセットを返すための演算子もいくつかあります。ONTAPONTAP Select は、以下に示すフィルタリング演算子をサポートしています。

オペレーター	説明
=	等しい
<	小なり
>	より大きい
⇐	以下
>=	より大きいか等しい
	または
!	等しくない
*	すべてに一致するワイルドカード

クエリの一部として null キーワードまたはその否定 (!null) を使用することで、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を使用してAPI呼び出しを実行すると、1つまたは複数の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 タイプによって異なります。次の方法で、fields クエリ パラメータを使用して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する `fields=*`ローカル サーバー メモリに保持されているオブジェクト フィールド、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。
- 高価なフィールドを指定する `fields=**`アクセスするために追加のサーバー処理を必要とするものも含め、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使用 `fields=FIELDNAME`必要なフィールドを正確に指定します。複数のフィールドを要求する場合は、各値の間に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、必要なフィールドを常に個別に指定することを推奨します。必要な場合にのみ、安価なフィールドと高価なフィールドのセットを取得してください。安価なフィールドと高価なフィールドの分類は、 NetAppが社内の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並べ替える

リソース コレクション内のレコードは、オブジェクトで定義されたデフォルトの順序で返されます。次のように、フィールド名と並べ替え方向を指定した 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 フィールドを降順で並べ替え、次に id フィールドを昇順で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、デフォルトでは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、リクエストにmax_recordsクエリパラメータを指定して、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、次のクエリは、指定された時刻以降に生成されたシステムイベントを最大10件返します。
`time⇒ 2019-04-04T15:41:29.140265Z&max_records=10`

複数のリクエストを発行して、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API呼び出しでは、前回の結果セットの最後のイベントに基づいて新しい時間の値を使用する必要があります。

APIレスポンスを解釈する

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 コードについて以下に説明します。

コード	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	作成	オブジェクトが正常に作成されました。場所応答ヘッダーには、オブジェクトの一意の識別子が含まれます。
202	承認済み	要求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、操作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。

コード	説明	説明
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	リクエスト内の HTTP 動詞はリソースではサポートされていません。
409	対立	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、要求を実行することができません。

応答ヘッダー

デプロイ サーバーによって生成される HTTP 応答には、次のようないくつか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 ID が割り当てられます。
- 場所 オブジェクトが作成されると、場所ヘッダーには、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、リクエストの成功または失敗によって異なります。レスポンス本文はJSON形式でレンダリングされます。

- 単一オブジェクト リクエストに基づいて、フィールドのセットを含む単一のオブジェクトを返すことができます。たとえば、GETで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。
- 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ます。 `num_records`レコード数と、オブジェクトインスタンスの配列を含むレコード数を示します。例えば、特定のクラスター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呼び出しが非同期的に処理される場合、バックグラウンドタスクをアンカーするジョブオブジェクトが返されます。例えば、クラスターのデプロイに使用されるPOSTリクエストは非同期的に処理され、ジョブオブジェクトを返します。
- エラーオブジェクト エラーが発生した場合、常にエラーオブジェクトが返されます。例えば、既に存在する名前でクラスターを作成しようとするとエラーが発生します。
- 空 場合によっては、データが返されず、レスポンス本文が空になります。例えば、DELETEを使用して既存のホストを削除した場合などです。

ONTAP Selectのジョブオブジェクトを使用した非同期処理

一部のDeploy API呼び出し、特にリソースの作成または変更を行う呼び出しは、他の呼び出しよりも完了までに時間がかかる場合があります。ONTAPONTAP Select Deployは、これらの長時間実行される要求を非同期的に処理します。

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 呼び出しが非同期に動作するかを判断するには、 ONTAP Select Deploy のオンライン ドキュメント ページを参照する必要があります。

APIリクエストに関連付けられたジョブオブジェクトをクエリする

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
- 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。

- 標準ポーリング要求現在のジョブ状態が直ちに返されます
- ロングポーリング要求ジョブ状態は、次のいずれかが発生した場合にのみ返されます。
  - 状態は、ポーリング要求で指定された日時値よりも最近に変更されました
  - 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、同じAPI呼び出しを使用してジョブオブジェクトをクエリします。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`そして `last_modified`。

デプロイ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。

非同期リクエスト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. ループ内では、各サイクルで次の操作を実行します。
   a. ロングポーリングリクエストでジョブの現在の状態を取得する
   b. ジョブが非終了状態（キューに入れられ、実行中）の場合、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

ブラウザで**ONTAP Select Deploy API**にアクセスする前に

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

展開計画

特定のデプロイメントまたは管理タスクの一環としてAPI呼び出しを発行する予定がある場合は、デプロイメント計画の作成を検討してください。この計画は正式なものでも非公式なものでも構いません。通常、計画には目標と使用するAPI呼び出しが含まれます。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呼び出しは、ドキュメントページで一貫した形式で説明されています。コンテンツには、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の詳細を次のように表示できます。

- サンプル値 API呼び出しで「サンプル値」をクリックすると、その呼び出しの典型的なJSON構造が表示されます。必要に応じてサンプル値を変更し、リクエストの入力として使用できます。
- モデル 「モデル」をクリックすると、JSON パラメータの完全なリストと各パラメータの説明が表示されます。

API呼び出しを発行する際の注意

デプロイ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、API 呼び出しを手動で発行したりするには、 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

次のものが必要です:

- 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、**Enter** キーを押します。

   ```
   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Sign in。

結果

デプロイ 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、表示されます。1つの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、解釈することができます。

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している必要があります。また、クラスタ作成時に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

ONTAP Selectクラスタの構成情報は、一意の識別子を使用して取得できます。この例では、低コスト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、必要なフィールドのみを要求することをお勧めします。

手順

1. メインページで一番下までスクロールし、「クラスター」をクリックします。
2. **GET /clusters/{cluster_id}** をクリックすると、 ONTAP Selectクラスタに関する情報を返すために使用される API 呼び出しの詳細が表示されます。

ワークフロープロセス

ONTAP Select Deploy 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する必要があります。

ワークフローで使用されるAPI呼び出しを理解する

ONTAP Selectの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。ここでは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ている各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記載されています。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に整理されています。特定のAPI呼び出しを見つけるには、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動詞を使用して実行されます。
- パス パスは、呼び出しの実行の一部として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を構築する

ONTAP Selectの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、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のURLと若干異なります。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 上にONTAP Selectシングルノード評価クラスタを作成する

vCenter によって管理される VMware ESXi ホストに、単一ノードの ONTAP Selectクラスタを導入できます。クラスタは評価ライセンスを使用して作成されます。

クラスター作成ワークフローは、次の状況によって異なります。

- ESXiホストはvCenter（スタンドアロンホスト）によって管理されていません
- クラスタ内で複数のノードまたはホストが使用される
- 購入したライセンスを使用してクラスタを本番環境に導入します
- VMware ESXiの代わりにKVMハイパーバイザーが使用される

1.vCenter Server の資格情報を登録する

vCenter Server 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認証情報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、この認証情報を使用して vCenter への認証を行うことができます。

カテゴリ	HTTP動詞	パス
導入	POST	/セキュリティ/資格情報

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

- ロケーション応答ヘッダーの認証情報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 ホストを追加する必要があります。

カテゴリ	**HTTP**動詞	パス
クラスタ	POST	/ホスト

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

3.クラスタを作成します。

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、ノード名が Deploy によって自動的に生成されます。

カテゴリ	**HTTP**動詞	パス
クラスタ	POST	/クラスター

Curl

単一ノード クラスターの場合、クエリ パラメータ node_count は 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

- ロケーション応答ヘッダー内のクラスタID

4.クラスターを構成する

クラスターの構成の一環として指定する必要が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	PATCH	/クラスター/{クラスターID}

Curl

クラスター 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（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
　なし

5.ノード名を取得する

Deploy管理ユーティリティは、クラスターの作成時にノードの識別子と名前を自動的に生成します。ノードを構成する前に、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'
```

処理タイプ
　同期

出力
　・一意のIDと名前を持つ単一のノードを記述する配列レコード

6.ノードを構成する

ノードを構成するために使用される 3 つの API 呼び出しの最初のものである、ノードの基本構成を提供する必要があります。

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

Curl
　クラスター 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（ステップ**06**）
　ONTAP Selectノードが実行されるホスト 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
}
```

処理タイプ

同期

出力

なし

7.ノードネットワークを取得する

単一ノードクラスタ内のノードが使用す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。単一ノードクラスタでは内部ネットワークは使用されません。

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'
```

処理タイプ

同期

出力

- ノードの単一の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）

8.ノードネットワークを構成する

データネットワークと管理ネットワークを構成する必要があります。単一ノードクラスタでは内部ネットワークは使用されません。

次の API 呼び出しを、ネットワークごとに 1 回ずつ、合計 2 回発行します。

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、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（ステップ**08**）

ネットワークの名前を指定する必要があります。

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

処理タイプ

同期

出力

なし

9.ノードストレージプールを構成する

ノード構成の最終ステップは、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 Client、またはオプションでDeploy REST APIから確認できます。

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、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（ステップ**09**）

プールの容量は2 TBです。

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

処理タイプ
　同期

出力
　なし

10.クラスターをデプロイする

クラスターとノードが構成されたら、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（ステップ**10**）
　ONTAP管理者アカウントのパスワードを入力する必要があります。

```
{
  "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日間評価インスタンスを導入する"

Pythonでアクセスする

Pythonを使用してONTAP Select Deploy APIにアクセスする前に

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、環境が適切に構成されていることを確認する必要があります。

- Python2の最新バージョン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。Python3にも移植可能ですが、互換性テストは実施されていません。
- RequestsライブラリとurIlib3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。
- スクリプトが実行されるクライアント ワークステーションは、ONTAP Select Deploy 仮想マシンへのネットワーク アクセスできる必要があります。

さらに、次の情報も必要です。

- デプロイ仮想マシンの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 の 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、様々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

スクリプトは、次の共通の特性を考慮し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 Pythonスクリプトは、適切に設定された任意のクライアントマシンから実行できます。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ます。各スクリプトは、入力パラメータを通じて CLI で制御されます。
- 入力ファイルの読み取り 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。クラスターを作成または削除する場合は、JSON 構成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場合は、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、1つのクラスが含まれています。このクラス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。

ONTAP Selectクラスタは、cluster.pyスクリプトを使用して作成できます。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環境に合わせて変更できます。

- ハイパーバイザーは、Deploy 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 によって管理することも、スタンドアロンホストとして使用することもできます。
- クラスター サイズ 単一ノードまたは複数ノードのクラスターを展開できます。
- 評価ライセンスまたは本番環境ライセンス 評価ライセンスまたは本番環境用に購入した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。

- デプロイサーバーのホスト名または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する

本番環境の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のデプロイ前でもデプロイ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。

- デプロイサーバーのホスト名または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

クラスターを削除する

スクリプト *delete_cluster.py* を使用して、既存のONTAP Selectクラスタを削除できます。

スクリプトの CLI 入力パラメータは次のとおりです。

- デプロイサーバーのホスト名またはIPアドレス
- 管理者ユーザーアカウントのパスワード
- JSON構成ファイルの名前

Pythonコードサンプル

ONTAP Selectクラスタを作成するスクリプト

次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ーを作成できます。

```
#!/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ためのスクリプトの 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ファイルを提供する必要があります。導入計画に応じ

て、適切なJSONサンプルをコピーして修正することができます。

ESXi上の単一ノードクラスタ

```
{
  "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** 上の単一ノード 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 上の単一ノード 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

次の CLI スクリプトを使用して、既存のクラスターを削除できます。

```
#!/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の共通の 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}
```

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())
```