概念
ONTAP Select

NetApp
February 09, 2024

目次

概念

基盤としての **REST Web** サービス

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、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

残りは、博士号の Roy Fielding によって正式にまとめられました "失策" 2000 年に UC アーバインにて。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、リソースを使用および管理します。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。
- リソースの状態および関連する状態操作の定義
  リソースの状態の数は有限で、リソースは必ずそのいずれかの状態にあります。状態、および状態の変化に影響する関連操作を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使用可能にする必要があります。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）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 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す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。

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。

オブジェクト **ID**

各リソースインスタンスまたはオブジェクトには、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呼び出しを発行すると、関連付けられているid値が `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には、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、いつでも変更される可能性があります。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ときに必要に応じてのみです。

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 `request-id` 関連付けられたHTTP応答のヘッダー。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処理は、
サーバーでのバックグラウンドタスク。Deploy REST APIでは、すべてのバックグラウンドタスクがタスクを追跡し、現在の状態などの情報を提供するジョブオブジェクト。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も、ジョブオブジェクトの成功または失敗を判断する方法があります。次のようなリクエストがあります。

- イベントメッセージ
  元の応答で返された要求IDを使用すると、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できます。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ステータス
  一部のリソースには、状態またはステータスの値が保持されています。この値を照会して、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 では、次のセキュリティテクノロジを使用します。

- トランスポートレイヤのセキュリティ
  Deployサーバとクライアントの間でネットワーク経由で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トランザクションで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で API トランザクションが構成されます。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。

- コンテンツタイプ
  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。
- 同意する
  応答の本文にJSONが含まれる場合は、このヘッダーをapplication/jsonに設定する必要があります。
- 許可
  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ベーシック認証を設定する必要があります。

本文を要求します

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）
- 空です

オブジェクトのフィルタ

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、一致する正確な値を指定できます。

```
<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。

演算子	説明
=	等しい
<	より小さい
>	が次の値より大きい
⇐	が次の値以下です
>=	が次の値以上である必要があります
		または
！	と等しくない
*	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 を使用する API 呼び出しを発行すると、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。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
  を指定します `fields=*` 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取得します。
- 高価なフィールド
  を指定します `fields=**` に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します。
- カスタムフィールドの選択
  使用 `fields=FIELDNAME` 必要な正確なフィールドを指定します。複数のフィールドを要求する場合は、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。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、タイプフィールドを降順でソートし、 ID を昇順でソートできます。
`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 API 呼び出しでは、前回の結果セットの最新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 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、必要に応じて追加データを取得したかどうか。

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	説明
200です	わかりました	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年だ	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です	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だ	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。
403です	禁止されている	認証エラーによりアクセスが拒否されました。
404です	が見つかりません	要求で参照されているリソースが存在しません。
405です	メソッドを使用できません	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だ	競合しています	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ドル	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。

- 要求ID
  成功するたびに、一意の要求識別子が割り当てられます。
- 場所
  オブジェクトが作成されると、一意のオブジェクトID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
  1 つ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 では、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、で一貫した形式が使用されます `num_records` 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します。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。
- 空です
  場合によっては、データが返されず、応答の本文が空になることがあります。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。

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の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。

- キューに登録され
- 実行中です
- 成功
- 失敗

ジョブオブジェクトをポーリングするときに、タスクの終了状態（成功または失敗）を検出するために使用できる 2 つの方法があります。

- 標準ポーリング要求
  現在のジョブの状態がすぐに返される
- 長時間のポーリング要求
  ジョブの状態は、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次の2つのクエリパラメータが含まれます。 `poll_timeout` および `last_modified`。

Deploy 仮想マシンのワークロードを減らすためには、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大まかな手順を次に示します。

1. 問題：非同期 API 呼び出し。
2. 要求が正常に受け取られたことを示す HTTP 応答 202 を受信します。
3. 応答の本文からジョブオブジェクトの識別子を抽出します。
4. ループ内で、各サイクルで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。

著作権に関する情報

商標に関する情報