使用REST自動化

ONTAP Select

NetApp
May 09, 2024

目錄

使用REST自動化

概念

REST Web服務基礎

代表性狀態傳輸（REST）是建立分散式Web應用程式的風格。當套用到Web服務API的設計時、它會建立一套技術和最佳實務做法、以揭露伺服器型資源並管理其狀態。它使用主流傳輸協定和標準、為部署和管理ONTAP Select 等叢集提供靈活的基礎。

架構與傳統限制

REST由Roy Fielding在博士課程中正式表達 "論文" 於2000年在UC爾灣舉行。它透過一組限制來定義架構風格、這些限制共同改善了網路型應用程式和基礎傳輸協定。這些限制會使用無狀態通訊協定、根據用戶端/伺服器架構來建立RESTful Web服務應用程式。

資源和狀態表示

資源是網路型系統的基本元件。建立REST Web服務應用程式時、早期的設計工作包括：

- 識別系統或伺服器型資源
  每個系統都會使用及維護資源。資源可以是檔案、商業交易、程序或管理實體。根據REST Web服務設計應用程式的首要任務之一、就是識別資源。
- 資源狀態和相關狀態作業的定義
  資源永遠處於有限的狀態之一。必須清楚定義狀態、以及用來影響狀態變更的相關作業。

用戶端與伺服器之間會交換訊息、以根據一般CRUD（建立、讀取、更新及刪除）模式來存取及變更資源狀態。

URI端點

每個REST資源都必須使用明確定義的定址方案來定義和提供。資源所在及識別的端點使用統一資源識別元（URI）。URI提供一般架構、可為網路中的每個資源建立唯一名稱。統一資源定位器（URL）是一種與Web服務搭配使用的URI、用於識別及存取資源。資源通常會以階層式結構公開、類似檔案目錄。

HTTP 訊息

超文字傳輸傳輸協定（HTTP）是Web服務用戶端和伺服器用來交換有關資源的要求和回應訊息的傳輸協定。在設計Web服務應用程式時、HTTP動詞（例如GET和POST）會對應至資源及對應的狀態管理動作。

HTTP為無狀態。因此、若要將一組相關的要求和回應與一筆交易建立關聯、則必須在隨要求/回應資料流一起提供的HTTP標頭中加入額外資訊。

JSON格式化

雖然資訊可透過多種方式在用戶端和伺服器之間進行結構化和傳輸、但最受歡迎的選項（以及與部署REST API搭配使用的選項）是JavaScript物件標記法（Json）。Json是以純文字表示簡單資料結構的產業標準、用於傳輸描述資源的狀態資訊。

如何存取部署API

由於REST Web服務固有的靈活度、ONTAP Select 因此可以透過多種不同的方式存取《支援》API。

部署公用程式原生使用者介面

存取API的主要方法是ONTAP Select 透過「功能不整合」網路使用者介面。瀏覽器會呼叫API、並根據使用者介面的設計重新格式化資料。您也可以透過部署公用程式命令列介面存取API。

部署線上文件頁面ONTAP Select

使用瀏覽器時、「支援功能」線上文件頁面可提供替代存取點。ONTAP Select除了提供直接執行個別API呼叫的方法之外、此頁面也包含API的詳細說明、包括每個呼叫的輸入參數和其他選項。API呼叫分為多個不同的功能區域或類別。

自訂程式

您可以使用多種不同的程式設計語言和工具來存取Deploy API。熱門選項包括Python、Java和Curl。使用API的程式、指令碼或工具會做為REST Web服務用戶端。使用程式設計語言可讓您更深入瞭解API、並提供自動化ONTAP Select 部署的機會。

部署API版本管理

隨附於Rashdeploy的REST API ONTAP Select 會指派版本編號。API版本編號與部署版本編號無關。您應該注意部署版本隨附的API版本、以及這會如何影響API的使用。

部署管理公用程式的目前版本包含REST API的第3版。舊版的部署公用程式包括下列API版本：

部署2.8及更新版本
: 包含REST API第3版的更新版本。ONTAP Select

部署2.7.2及更早版本
: 包含REST API第2版的更新版本。ONTAP Select

> REST API的第2版和第3版不相容。如果您從包含API第2版的舊版升級至部署2.8或更新版本、則必須更新任何直接存取API的現有程式碼、以及使用命令列介面的任何指令碼。

基本營運特性

REST建立一套通用的技術和最佳實務做法、但每個API的詳細資料可能會因設計選項而異。在使用API之前、您應該先瞭解ONTAP Select 到「更新部署API」的詳細資料和操作特性。

Hypervisor主機與ONTAP Select 非節點

Hypervisor主機_是裝載ONTAP Select 一個整套虛擬機器的核心硬體平台。當在Hypervisor主機上部署及啟用某部支援的虛擬機器時、該虛擬機器被視為_這個節點_。ONTAP Select ONTAP Select在部署REST API的第3版中、主機和節點物件是分開且獨立的。這可建立一對多關係、讓一或多ONTAP Select 個支援節點可在同一

個Hypervisor主機上執行。

物件識別碼

每個資源執行個體或物件在建立時都會指派一個唯一的識別碼。這些識別碼在ONTAP Select 特定的例子中是全域唯一的。發出建立新物件執行個體的 API 呼叫後、會將相關的 ID 值傳回給中的呼叫者 `location` HTTP回應的標頭。您可以擷取識別碼、並在參照資源執行個體時用於後續通話。

> 物件識別碼的內容和內部結構可隨時變更。當您參照相關的物件時、只能視需要在適用的API呼叫上使用識別碼。

要求識別碼

每個成功的API要求都會指派一個唯一的識別碼。識別碼會傳回至 `request-id` 相關 HTTP 回應的標頭。您可以使用要求識別碼、統稱為單一特定API要求回應交易的活動。例如、您可以根據要求ID擷取交易的所有事件訊息。

同步和非同步呼叫

伺服器執行從用戶端接收的HTTP要求的主要方法有兩種：

- 同步
  伺服器會立即執行要求、並以 200 、 201 或 204 的狀態碼回應。
- 非同步
  伺服器接受要求、並以狀態代碼 202 回應。這表示伺服器已接受用戶端要求、並開始執行背景工作以完成要求。最終成功或失敗無法立即取得、必須透過額外的API呼叫來判斷。

確認已完成長時間執行的工作

一般而言、任何可能需要較長時間才能完成的作業、都會使用非同步處理
伺服器的背景工作。使用部署 REST API 時、每個背景工作都會以錨定
工作物件、可追蹤工作並提供資訊、例如目前狀態。工作物件、
在建立背景工作之後、 HTTP 回應會傳回其唯一識別碼、

您可以直接查詢「工作」物件、以判斷相關聯的API呼叫是否成功。
如需其他資訊、請參閱_使用工作物件進行非同步處理_。

除了使用工作物件之外、還有其他方法可以判斷的成功或失敗
申請、包括：

- 事件訊息
  您可以使用傳回原始回應的要求 ID 、擷取與特定 API 呼叫相關的所有事件訊息。事件訊息通常包含成功或失敗的指示、也可在偵錯錯誤條件時使用。
- 資源狀態
  有幾個資源會維持一個狀態或狀態值、您可以查詢該值、以間接判斷要求的成功或失敗。

安全性

部署API使用下列安全技術：

- 傳輸層安全性

透過網路在部署伺服器和用戶端之間傳送的所有流量都會透過 TLS 加密。不支援在未加密的通道上使用HTTP傳輸協定。支援TLS 1.2版。

- HTTP 驗證
  每項 API 交易都會使用基本驗證。每個要求都會新增一個HTTP標頭、其中包含基礎64字串中的使用者名稱和密碼。

要求及回應**API**交易

每個部署API呼叫都會以HTTP要求的形式執行、以供部署虛擬機器產生與用戶端相關的回應。此要求/回應配對被視為API交易。在使用部署API之前、您應該先熟悉可用於控制要求的輸入變數、以及回應輸出的內容。

控制**API**要求的輸入變數

您可以透過HTTP要求中設定的參數來控制API呼叫的處理方式。

要求標頭

您必須在HTTP要求中包含多個標頭、包括：

- 內容類型
  如果要求主體包含 JSON 、則必須將此標頭設定為 application/json 。
- 接受
  如果回應本文將包含 JSON 、則必須將此標頭設定為 application/json 。
- 授權
  必須使用以 base64 字串編碼的使用者名稱和密碼來設定基本驗證。

申請本文

申請本文的內容會因特定通話而有所不同。HTTP要求本文包含下列其中一項：

- 具有輸入變數的Json物件（例如新叢集的名稱）
- 空白

篩選物件

發出使用Get的API呼叫時、您可以根據任何屬性來限制或篩選傳回的物件。例如、您可以指定要符合的確切值：

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

除了完全符合的項目、還有其他運算子可以傳回一系列值的一組物件。支援下列篩選操作員。ONTAP Select

營運者	說明
=	等於
<	小於
>	大於

營運者	說明
&l;=	小於或等於
>=	大於或等於
	或
!	不等於
*	貪婪的萬用字元

您也可以使用null關鍵字或其否定（！null）做為查詢的一部分、根據是否設定特定欄位來傳回一組物件。

選取物件欄位

根據預設、使用Get發出API呼叫時、只會傳回唯一識別物件的屬性。這組最小欄位可做為每個物件的金鑰、而且會根據物件類型而有所不同。您可以使用「欄位查詢」參數、以下列方式選取其他物件屬性：

- 經濟實惠的欄位
  指定 `fields=*` 擷取在本機伺服器記憶體中維護的物件欄位、或只需少量處理即可存取。
- 昂貴的領域
  指定 `fields=**` 擷取所有物件欄位、包括需要額外伺服器處理才能存取的欄位。
- 自訂欄位選擇
  使用 `fields=FIELDNAME` 指定您要的確切欄位。要求多個欄位時、必須使用不含空格的逗號分隔值。

> 最佳實務做法是、務必找出您想要的特定欄位。您只能在需要時擷取一組廉價或昂貴的欄位。價格低廉且昂貴的分類是由NetApp根據內部效能分析所決定。特定欄位的分類可隨時變更。

排序輸出集中的物件

資源集合中的記錄會以物件定義的預設順序傳回。您可以使用 order_by 查詢參數變更訂單、並使用欄位名稱和排序方向、如下所示：

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

例如、您可以依遞增順序、以遞減順序排序類型欄位、然後依ID排序：

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

包含多個參數時、您必須以逗號分隔欄位。

分頁

使用Get存取同一類型物件的集合時發出API呼叫、預設會傳回所有相符的物件。如有需要、您可以使用main_Records查詢參數搭配要求來限制傳回的記錄數。例如：

```
max_records=20
```

如有需要、您可以將此參數與其他查詢參數合併、以縮小結果集範圍。例如、下列項目最多會傳回指定時間之後產生的 10 個系統事件：

```
time⇒ 2019-04-04T15:41:29.140265Z&max_records=10
```

您可以針對事件（或任何物件類型）發出多個分頁要求。每次後續的API呼叫都應根據最後結果集中的最新事件、使用新的時間值。

解讀API回應

每個API要求都會對用戶端產生回應。您可以檢查回應以判斷是否成功、並視需要擷取其他資料。

HTTP狀態代碼

部署REST API所使用的HTTP狀態代碼如下所述。

程式碼	意義	說明
200	好的	表示未建立新物件的通話成功。
201.	已建立	已成功建立物件；位置回應標頭包含物件的唯一識別碼。
202.02	已接受	執行要求的背景工作已開始執行、但作業尚未完成。
400	錯誤要求	無法辨識或不適當的要求輸入。
403.	禁止	由於授權錯誤、存取遭拒。
404.04	找不到	要求中提及的資源不存在。
405	不允許使用方法	資源不支援要求中的HTTP動詞。
409.	衝突	建立物件的嘗試失敗、因為物件已經存在。
500	內部錯誤	伺服器發生一般內部錯誤。
501..	未實作	URI已知但無法執行要求。

回應標頭

部署伺服器產生的HTTP回應中包含數個標頭、包括：

- 要求識別碼
  每個成功的 API 要求都會指派唯一的要求識別碼。
- 位置
  建立物件時、位置標頭會包含新物件的完整 URL 、包括唯一物件識別碼。

回應本文

與API要求相關的回應內容會因物件、處理類型、以及要求的成功或失敗而有所不同。回應本文會以Json呈現。

- 單一物件
  單一物件可根據要求傳回一組欄位。例如、您可以使用「Get」（取得）、使用唯一識別碼擷取叢集的選定內容。
- 多個物件
  可從資源集合傳回多個物件。在任何情況下、都會使用一致的格式 `num_records` 指出包含物件執行個體陣列的記錄和記錄數。例如、您可以擷取在特定叢集中定義的所有節點。
- 工作物件
  如果API呼叫以非同步方式處理、則會傳回工作物件、以固定背景工作。例如、用於部署叢集的POST要求會以非同步方式處理、並傳回工作物件。
- 錯誤物件
  如果發生錯誤、一律會傳回錯誤物件。例如、當您嘗試建立已存在名稱的叢集時、會收到錯誤訊息。

- 空白
  在某些情況下、不會傳回任何資料、回應本文是空的。例如、使用DELETE刪除現有主機之後、回應本文為空白。

使用工作物件進行非同步處理

有些部署API呼叫（尤其是建立或修改資源的呼叫）可能需要比其他呼叫更長的時間才能完成。以非同步方式部署這些長時間執行的要求。ONTAP Select

使用工作物件說明的非同步要求

在非同步執行 API 呼叫之後、 HTTP 回應代碼 202 表示該要求已成功驗證並接受、但尚未完成。此要求會以背景工作的形式處理、並在對用戶端的初始 HTTP 回應之後繼續執行。回應包括繫留要求的工作物件、包括其唯一識別碼。

> 您應該參閱ONTAP Select 「非同步部署」線上文件頁面、以判斷哪些API呼叫是以非同步方式運作。

查詢與 **API** 要求相關的工作物件

HTTP回應中傳回的工作物件包含數個內容。您可以查詢狀態內容、以判斷要求是否成功完成。工作物件可以處於下列其中一種狀態：

- 已佇列
- 執行中
- 成功
- 故障

輪詢工作物件以偵測工作的終端機狀態時、您可以使用兩種技巧：成功或失敗：

- 標準輪詢要求
  目前工作狀態會立即傳回
- 長輪詢要求
  只有在發生下列其中一種情況時、才會傳回工作狀態：
  - 狀態變更的時間比輪詢要求上提供的日期時間值還要晚
  - 逾時值已過期（1至120秒）

標準輪詢和長輪詢使用相同的API呼叫來查詢工作物件。不過、長輪詢要求包含兩個查詢參數： `poll_timeout` 和 `last_modified`。

> 您應該永遠使用長輪詢來減少部署虛擬機器上的工作負載。

發出非同步要求的一般程序

您可以使用下列高階程序來完成非同步 API 呼叫：

1. 發出非同步 API 呼叫。

2. 接收 HTTP 回應 202 、表示已成功接受要求。
3. 從回應本文擷取工作物件的識別碼。
4. 在迴圈內、在每個週期中執行下列步驟：
    a. 以長時間輪詢要求取得工作的目前狀態
    b. 如果工作處於非終端機狀態（佇列中、執行中）、請再次執行迴圈。
5. 當工作達到終端機狀態（成功、失敗）時停止。

使用瀏覽器存取

使用瀏覽器存取**API**之前

在使用「部署」線上文件頁面之前、您應該瞭解幾項事項。

部署計畫

如果您打算在執行特定部署或管理工作時發出API呼叫、則應考慮建立部署計畫。這些計畫可以是正式或非正式的、通常包含您的目標和要使用的API呼叫。如需詳細資訊、請參閱使用部署REST API的工作流程程序。

Json範例和參數定義

每個API呼叫都會以一致的格式在文件頁面中說明。內容包括實作附註、查詢參數及HTTP狀態代碼。此外、您也可以顯示API要求和回應所使用的Json詳細資料、如下所示：

- 範例值
  如果您在 API 呼叫上按一下 *example. value* 、則會顯示通話的典型 JSON 結構。您可以視需要修改範例、並將其做為申請的輸入。
- 模型
  如果您按一下 *Model* 、就會顯示完整的 JSON 參數清單、並提供每個參數的說明。

發出**API**呼叫時的注意事項

您使用「部署」文件頁面執行的所有API作業都是即時作業。您應小心不要錯誤地建立、更新或刪除組態或其他資料。

存取部署文件頁面

您必須存取ONTAP Select 「更新版本」線上文件頁面、才能顯示API文件、以及手動發出API呼叫。

開始之前

您必須具備下列條件：

- 物件部署虛擬機器的IP位址或網域名稱ONTAP Select
- 系統管理員的使用者名稱和密碼

步驟

1. 在瀏覽器中輸入URL、然後按* Enter *：

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

2. 使用管理員使用者名稱和密碼登入。

結果

「部署」文件網頁會顯示、頁面底部會顯示依類別排列的通話。

瞭解並執行 **API** 呼叫

所有API呼叫的詳細資料均以通用格式記錄及顯示在ONTAP Select 「更新」線上文件網頁上。只要瞭解單一API呼叫、即可存取及解譯所有API呼叫的詳細資料。

開始之前

您必須登入ONTAP Select 到「Webdeploy線上文件」網頁。建立叢集時、您必須將唯一識別碼指派給ONTAP Select 您的叢集。

關於這項工作

您可以ONTAP Select 使用獨特的識別碼來擷取描述某個叢集的組態資訊。在此範例中、會傳回所有歸類為「廉價」的欄位。不過、最佳實務做法是、您只能要求所需的特定欄位。

步驟

1. 在主頁面上、捲動至底部、然後按一下「叢集」。
2. 按一下「* Get /cluster / ｛cluster _id｝ *」以顯示API呼叫的詳細資料、該API呼叫用於傳回ONTAP Select 有關某個叢集的資訊。

工作流程程序

使用**API**工作流程之前

您應該準備好檢閱及使用工作流程程序。

瞭解工作流程中使用的 **API** 呼叫

《支援》線上文件頁面包含每次REST API呼叫的詳細資料。ONTAP Select工作流程範例中使用的每個API呼叫都只包含您在文件頁面上找到呼叫所需的資訊、而非在此重複這些詳細資料。找到特定API呼叫之後、您可以檢閱通話的完整詳細資料、包括輸入參數、輸出格式、HTTP狀態代碼及要求處理類型。

工作流程中的每個API呼叫都包含下列資訊、可協助您在文件頁面上找到呼叫：

- 類別
  API呼叫會在文件頁面上組織成功能相關的區域或類別。若要尋找特定的API呼叫、請捲動至頁面底部、然後按一下適用的API類別。
- HTTP動詞
  HTTP動詞可識別在資源上執行的動作。每個API呼叫都是透過單一HTTP動詞來執行。
- 路徑
  路徑會決定動作在執行通話時套用到的特定資源。路徑字串會附加至核心URL、以構成識別資源的完整

URL。

建構 **URL** 以直接存取 **REST API**

除了「支援資訊」文件頁面、您也可以直接透過編程語言（例如Python）存取「部署REST API」ONTAP Select。在此情況下、核心URL與存取線上文件頁面時所使用的URL略有不同。直接存取API時、您必須將/API附加至網域和連接埠字串。例如：
`http://deploy.mycompany.com/api`

工作流程 1 ：在 ESXi 上建立單一節點評估叢集

您可以在ONTAP Select 由vCenter管理的VMware ESXi主機上部署單節點的VMware ESXi叢集。叢集是以評估授權所建立。

叢集建立工作流程在下列情況下有所不同：

- ESXi主機並非由vCenter（獨立主機）管理
- 叢集內使用多個節點或主機
- 叢集部署於已購買授權的正式作業環境中
- 使用KVM Hypervisor而非VMware ESXi

> ◦ 從功能更新至功能更新至功能更新、您將無法再在KVM Hypervisor上部署新叢集ONTAP Select 。
> ◦ 從功能支援的版本起、除了「離線」和「刪除」功能之外、所有的管理功能都不再適用於現有的KVM叢集和主機ONTAP Select 。

1.登錄**vCenter**伺服器認證

部署至由vCenter伺服器管理的ESXi主機時、您必須先新增認證、才能登錄主機。然後、部署管理公用程式就可以使用認證來驗證vCenter。

類別	**HTTP**動詞	路徑
部署	貼文	/安全性/認證

捲髮

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

Json輸入（步驟**01**）

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

處理類型

非同步

輸出

- 位置回應標頭中的認證ID
- 工作物件

2.註冊Hypervisor主機

您必須新增Hypervisor主機、以便ONTAP Select 執行包含此節點的虛擬機器。

類別	**HTTP**動詞	路徑
叢集	貼文	/主機

捲髮

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

Json輸入（步驟**02**）

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

處理類型

非同步

輸出

- 位置回應標頭中的主機ID

- 工作物件

3.建立叢集

當您建立ONTAP Select 一個叢集時、系統會登錄基本的叢集組態、並透過部署自動產生節點名稱。

類別	**HTTP**動詞	路徑
叢集	貼文	/叢集

捲髮

單一節點叢集的查詢參數node_count應設為1。

```
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輸入（步驟**03**）

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

處理類型

同步

輸出

- 位置回應標頭中的叢集ID

4.設定叢集

在設定叢集時、您必須提供幾項屬性。

類別	**HTTP**動詞	路徑
叢集	修補程式	叢集/ {cluster}

捲髮

您必須提供叢集ID。

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

Json輸入（步驟**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"}
}
```

處理類型
　同步

輸出
　無

5.擷取節點名稱

當建立叢集時、部署管理公用程式會自動產生節點識別碼和名稱。您必須先擷取指派的ID、才能設定節點。

類別	**HTTP**動詞	路徑
叢集	取得	叢集/｛cluster｝/節點

捲髮
　您必須提供叢集ID。

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

處理類型
　同步

輸出
- 陣列會記錄每個以唯一ID和名稱描述單一節點的資料

6.設定節點

您必須提供節點的基本組態、這是用來設定節點的三個API呼叫中的第一個。

類別	**HTTP**動詞	路徑
叢集	路徑	叢集/｛cluster｝/節點/｛node_id｝

捲髮

您必須提供叢集ID和節點ID。

```
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輸入（步驟06）

您必須提供ONTAP Select 執行此節點的主機ID。

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

處理類型

同步

輸出

無

7.擷取節點網路

您必須識別單節點叢集中節點所使用的資料和管理網路。內部網路不適用於單一節點叢集。

類別	**HTTP**動詞	路徑
叢集	取得	叢集/｛cluster｝/節點/｛node_id｝/網路

捲髮

您必須提供叢集ID和節點ID。

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

處理類型

同步

輸出

- 兩筆記錄的陣列、每筆記錄分別說明節點的單一網路、包括唯一ID和用途

8.設定節點網路

您必須設定資料和管理網路。內部網路不適用於單一節點叢集。

發出下列API呼叫兩次、每個網路一次。

類別	**HTTP**動詞	路徑
叢集	修補程式	叢集/｛cluster｝/節點/｛node_id｝/網路/｛network_id｝

捲髮

您必須提供叢集ID、節點ID和網路ID。

```
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輸入（步驟**08**）

您需要提供網路名稱。

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

處理類型

同步

輸出

無

9.設定節點儲存資源池

設定節點的最後一步是附加儲存資源池。您可以透過vSphere Web用戶端或透過部署REST API（選用）來判斷可用的儲存資源池。

類別	**HTTP**動詞	路徑
叢集	修補程式	叢集/｛cluster｝/節點/｛node_id｝/網路/｛network_id｝

捲髮

您必須提供叢集ID、節點ID和網路ID。

```
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輸入（步驟09）

集區容量為2 TB。

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

處理類型

同步

輸出

無

10.部署叢集

設定叢集和節點之後、即可部署叢集。

類別	**HTTP**動詞	路徑
叢集	貼文	叢集/｛cluster｝/部署

捲髮

您必須提供叢集ID。

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

Json輸入（步驟10）

您必須提供ONTAP 該管理員帳戶的密碼。

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

處理類型

非同步

輸出

- 工作物件

使用Python存取

使用Python存取API之前

在執行範例Python指令碼之前、您必須先準備環境。

在執行Python指令碼之前、您必須確定環境設定正確：

- 必須安裝最新適用版本的Python2。
  樣本代碼已使用Python2進行測試。它們也應可攜至Python3、但尚未經過相容性測試。
- 必須安裝要求和urllib3程式庫。
  您可以根據環境使用pip或其他Python管理工具。
- 執行指令碼的用戶端工作站必須能夠透過網路存取ONTAP Select 到該物件部署虛擬機器。

此外、您必須具備下列資訊：

- 部署虛擬機器的IP位址
- 部署系統管理員帳戶的使用者名稱和密碼

瞭解 Python 指令碼

Python指令碼範例可讓您執行多項不同的工作。您應該先瞭解指令碼、再在即時部署執行個體中使用。

通用設計特性

指令碼的設計具有下列常見特性：

- 從用戶端機器的命令列介面執行
  您可以從任何正確設定的用戶端機器執行 Python 指令碼。請參閱_開始前_以取得更多資訊。
- 接受 CLI 輸入參數
  每個指令碼都是透過輸入參數在 CLI 中控制。
- 讀取輸入檔
  每個指令碼都會根據其用途讀取輸入檔。建立或刪除叢集時、您必須提供Json組態檔。新增節點授權時、您必須提供有效的授權檔案。
- 使用通用支援模組
  一般支援模組 *deploy_request.py* 包含單一類別。每個指令碼都會匯入並使用此指令碼。

建立叢集

您可以ONTAP Select 使用指令碼叢集.py來建立一個不穩定叢集。根據Json輸入檔的CLI參數和內容、您可以將指令碼修改為部署環境、如下所示：

- 從功能更新至功能更新至功能更新、您將無法再在KVM Hypervisor上部署新叢集ONTAP Select 。
- 從功能支援的版本起、除了「離線」和「刪除」功能之外、所有的管理功能都不再適用於現有的KVM叢集和主機ONTAP Select 。

- Hypervisor
  您可以部署至 ESXi 或 KVM （視部署版本而定）。部署至ESXi時、Hypervisor可以由vCenter管理、也可以是獨立式主機。
- 叢集大小
  您可以部署單一節點或多節點叢集。
- 評估或正式作業授權
  您可以部署具有評估版或購買正式作業授權的叢集。

指令碼的CLI輸入參數包括：

- 部署伺服器的主機名稱或IP位址
- 管理使用者帳戶的密碼
- Json組態檔的名稱
- 訊息輸出的詳細旗標

新增節點授權

如果您選擇部署正式作業叢集、則必須使用指令碼_add_license .py_為每個節點新增授權。您可以在部署叢集之前或之後新增授權。

指令碼的CLI輸入參數包括：

- 部署伺服器的主機名稱或IP位址
- 管理使用者帳戶的密碼
- 授權檔案名稱
- 具備新增授權權限的使用者名稱ONTAP
- 密碼ONTAP

刪除叢集

您可以ONTAP Select 使用指令碼_delete_cluster .py_刪除現有的叢集。

指令碼的CLI輸入參數包括：

- 部署伺服器的主機名稱或IP位址
- 管理使用者帳戶的密碼
- Json組態檔的名稱

Python程式碼範例

建立叢集的指令碼

您可以使用下列指令碼、根據指令碼中定義的參數和Json輸入檔來建立叢集。

```
#!/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)
```

用於建立叢集之指令碼的Json

使用ONTAP Select Python程式碼範例建立或刪除一個故障叢集時、您必須提供一個Json檔案作為指令碼的輸入。您可以根據部署計畫來複製及修改適當的Json範例。

ESXi上的單節點叢集

```
{
  "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
          }
        ]
      }
    }
  ]
}
```


使用**vCenter**在**ESXi**上使用單節點叢集


```
{
  "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上的單節點叢集

- 從功能更新至功能更新至功能更新、您將無法再在KVM Hypervisor上部署新叢集ONTAP Select 。
- 從功能支援的版本起、除了「離線」和「刪除」功能之外、所有的管理功能都不再適用於現有的KVM叢集和主機ONTAP Select 。

```
{
  "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
          }
        ]
      }
    }
  ]
}
```


新增節點授權的指令碼

您可以使用下列指令碼來新增ONTAP Select 適用於某個節點的授權。

```
#!/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)
```

刪除叢集的指令碼

您可以使用下列CLI指令碼來刪除現有的叢集。

```
#!/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)
```

通用支援模組

所有Python指令碼都會在單一模組中使用通用的Python類別。

```
#!/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}
```

調整叢集節點大小的指令碼

您可以使用下列指令碼來調整ONTAP Select 叢集中節點的大小。

```
#!/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())
```