ONTAP

NetApp Automation

NetApp
November 18, 2025

目錄

ONTAP

第 0/1 天

ONTAP Day 0/1 解決方案總覽

您可以使用ONTAP day 0/1 自動化解決方案，透過 Ansible 部署和設定ONTAP叢集。解決方案可從以下途徑取得： "NetApp Console自動化中心"。

靈活的 ONTAP 部署選項

視您的需求而定、您可以使用內部部署硬體或模擬 ONTAP 、使用 Ansible 來部署和設定 ONTAP 叢集。

內部部署硬體

您可以使用執行 ONTAP 的內部部署硬體來部署此解決方案、例如 FAS 或 AFF 系統。您必須使用 Linux VM 來使用 Ansible 來部署和設定 ONTAP 叢集。

模擬 ONTAP

若要使用 ONTAP 模擬器部署此解決方案、您必須從 NetApp 支援網站下載最新版的模擬 ONTAP 。模擬 ONTAP 是 ONTAP 軟體的虛擬模擬器。模擬 ONTAP 在 Windows 、 Linux 或 Mac 系統上的 VMware Hypervisor 中執行。對於 Windows 和 Linux 主機、您必須使用 VMware Workstation Hypervisor 來執行此解決方案。如果您有 Mac OS 、請使用 VMware Fusion Hypervisor 。

分層設計

Ansible 架構可簡化自動化執行與邏輯工作的開發與重複使用。此架構可區分決策工作（邏輯層）和自動化的執行步驟（執行層）。瞭解這些層的運作方式、可讓您自訂組態。

Ansible 「教戰手冊」從開始到結束都會執行一系列工作。 `site.yml`本教戰手冊包含了 `logic.yml`教戰手冊和 `execution.yml`教戰手冊。

執行要求時、教戰手冊會 `site.yml`先呼叫 `logic.yml`教戰手冊、然後呼叫 `execution.yml`教戰手冊以執行服務要求。

您不需要使用架構的邏輯層。邏輯層提供選項、可將架構的功能擴充至硬式編碼的執行值以外的範圍。這可讓您視需要自訂架構功能。

邏輯層

邏輯層由下列項目組成：

- 教 `logic.yml`戰手冊
- 目錄中的邏輯工作檔案 `logic-tasks`

邏輯層提供複雜決策的功能、無需大量自訂整合（例如連線至 ServiceNow ）。邏輯層是可設定的、可為微服務提供輸入。

此外也提供繞過邏輯層的功能。如果您想略過邏輯層、請勿定義 `logic_operation`變數。直接呼叫 `logic.yml`教戰手冊可讓您在不執行的情況下進行某種程度的偵錯。您可以使用「 DEBUG 」陳述式來驗證的值是否 `raw_service_request`正確。

重要考量：

- 教戰手冊會 `logic.yml`搜尋 `logic_operation`變數。如果在要求中定義變數、它會從目錄載入工作檔案 `logic-tasks`。工作檔案必須是 .yml 檔案。如果沒有相符的工作檔案且已定義變數、則 `logic_operation`邏輯層會失敗。
- 變數的預設值 `logic_operation`為 `no-op`。如果未明確定義變數、則預設為 `no-op`、不會執行任何作業。
- 如果 `raw_service_request`已定義變數、則執行會繼續執行至執行層。如果未定義變數、則邏輯層會失敗。

執行層

執行層由下列項目組成：

- 教 `execution.yml`戰手冊

執行層會呼叫 API 以設定 ONTAP 叢集。 `execution.yml`教戰手冊要求 `raw_service_request`在執行時定義變數。

支援自訂

您可以根據自己的需求、以各種方式自訂此解決方案。

自訂選項包括：

- 修改 Ansible 教戰手冊
- 新增角色

自訂 **Ansible** 檔案

下表說明本解決方案所包含的可自訂 Ansible 檔案。

位置	說明
`playbooks/inventory/hosts`	包含一個包含主機和群組清單的檔案。
`playbooks/group_vars/all/*`	Ansible 提供一種便利的方法、可一次將變數套用至多個主機。您可以修改此文件夾中的任何或所有文件，包括 `cfg.yml`、、 `clusters.yml` `defaults.yml` `services.yml` `standards.yml`、和 `vault.yml`。
`playbooks/logic-tasks`	支援 Ansible 內部的決策工作、並維持邏輯與執行的分離。您可以將檔案新增至此資料夾、以對應至相關服務。
`playbooks/vars/*`	Ansible 教戰手冊和角色中使用的動態值、可實現組態的自訂、靈活度及重新使用。如有必要、您可以修改此資料夾中的任何或所有檔案。

自訂角色

您也可以透過新增或變更 Ansible 角色（也稱為微服務）來自訂解決方案。如需詳細資訊"自訂"、請參閱。

準備使用 **ONTAP Day 0/1** 解決方案

部署自動化解決方案之前、您必須先準備好 ONTAP 環境、並安裝及設定 Ansible 。

初始規劃考量

在使用此解決方案部署 ONTAP 叢集之前、您應該先檢閱下列需求和考量事項。

基本需求

若要使用此解決方案、您必須符合下列基本要求：

- 您必須能夠在內部部署或透過 ONTAP 模擬器存取 ONTAP 軟體。
- 您必須知道如何使用 ONTAP 軟體。
- 您必須知道如何使用 Ansible 自動化軟體工具。

規劃考量

部署此自動化解決方案之前、您必須先決定：

- 執行 Ansible 控制節點的位置。
- ONTAP 系統、內部部署硬體或 ONTAP 模擬器。
- 您是否需要自訂。

準備 **ONTAP** 系統

無論您是使用內部部署 ONTAP 系統或模擬 ONTAP 、都必須先準備好環境、才能部署自動化解決方案。

您也可以選擇安裝及設定模擬 **ONTAP**

如果您想要透過 ONTAP 模擬器部署此解決方案、則必須下載並執行模擬 ONTAP 。

開始之前

- 您必須下載並安裝要用來執行模擬 ONTAP 的 VMware Hypervisor 。
  - 如果您有 Windows 或 Linux 作業系統、請使用 VMware Workstation 。
  - 如果您有 Mac OS 、請使用 VMware Fusion 。

> 如果您使用的是 Mac OS 、則必須使用 Intel 處理器。

步驟

請使用下列程序在您的本機環境中安裝兩個 ONTAP 模擬器：

1. 從下載模擬 ONTAP "NetApp 支援網站"。

> 雖然您安裝了兩個 ONTAP 模擬器、但只需下載一份軟體複本。

2. 如果尚未執行、請啟動 VMware 應用程式。

3. 找到下載的模擬器檔案、然後按一下滑鼠右鍵、以 VMware 應用程式開啟該檔案。
4. 設定第一個 ONTAP 執行個體的名稱。
5. 等待模擬器開機、並依照指示建立單一節點叢集。

   針對第二個 ONTAP 執行個體重複步驟。

6. 或者、您也可以新增完整的磁碟補充。

   從每個叢集執行下列命令：

```
security unlock -username <user_01>
security login password -username <user_01>
set -priv advanced
systemshell local
disk assign -all -node <Cluster-01>-01
```

ONTAP 系統狀態

您必須驗證 ONTAP 系統的初始狀態、無論是內部部署或透過 ONTAP 模擬器執行。

確認符合下列 ONTAP 系統需求：

- ONTAP 已安裝並在尚未定義叢集的情況下執行。
- ONTAP 已開機並顯示存取叢集的 IP 位址。
- 可連線至網路。
- 您擁有管理認證。
- 當日訊息（ MOTD ）橫幅會顯示管理位址。

安裝所需的自動化軟體

本節提供如何安裝 Ansible 及準備部署自動化解決方案的資訊。

安裝 Ansible

Ansible 可以安裝在 Linux 或 Windows 系統上。

Ansible 用於與 ONTAP 叢集通訊的預設通訊方法是 SSH 。

請參閱"NetApp與Ansible快速入門：安裝Ansible"以安裝 Ansible 。

> Ansible 必須安裝在系統的控制節點上。

下載並準備自動化解決方案

您可以使用下列步驟下載並準備部署的自動化解決方案。

1. 下載 "ONTAP - Day 0/1 ；健全狀況檢查" 透過控制台 Web 使用者介面實現自動化解決方案。此解決方案打包如下： `ONTAP_DAY0_DAY1.zip`。
2. 解壓縮 zip 資料夾、並將檔案複製到 Ansible 環境中控制節點上的所需位置。

初始 **Ansible** 架構組態

執行 Ansible 架構的初始組態：

1. 瀏覽至 `playbooks/inventory/group_vars/all`。
2. 解密 `vault.yml`檔案：

   ```
   ansible-vault decrypt playbooks/inventory/group_vars/all/vault.yml
   ```

   當系統提示您輸入資料保險箱密碼時、請輸入下列暫時密碼：

   ```
   NetApp123!
   ```

   > "NetApp123!" 是一種用來解密檔案和對應資料保險箱密碼的暫 `vault.yml`存密碼。第一次使用後、您 * 必須 * 使用自己的密碼來加密檔案。

3. 修改下列 Ansible 檔案：

   - `clusters.yml`- 修改此檔案中的值以符合您的環境。
   - `vault.yml`- 解密檔案後、請修改 ONTAP 叢集、使用者名稱和密碼值、以符合您的環境。
   - `cfg.yml`- 設定的檔案路徑 `log2file`，並在 [設定] 底下 `cfg`**設定** `show_request`**為** `True` [顯示 `raw_service_request`] 。

     此 `raw_service_request`變數會在記錄檔和執行期間顯示。

   > 列出的每個檔案都包含註解、並說明如何根據您的需求進行修改。

4. 重新加密 `vault.yml`檔案：

   ```
   ansible-vault encrypt playbooks/inventory/group_vars/all/vault.yml
   ```

   > 系統會提示您在加密時為資料保險箱選擇新密碼。

5. 瀏覽 `playbooks/inventory/hosts`並設定有效的 Python 解譯器。
6. 部署 `framework_test`服務：

   下列命令會以值為的 `cluster_identity_info`**方式執行** `na_ontap_info`**模組** `gather_subset`。這會驗證基本組態是否正確、並確認您可以與叢集通訊。

   ```
   ansible-playbook -i inventory/hosts site.yml -e
   cluster_name=<CLUSTER_NAME>
   -e logic_operation=framework-test
   ```

為每個叢集執行命令。

如果成功、您應該會看到類似下列範例的輸出：

```
PLAY RECAP
*********************************************************************
*********
localhost : ok=12 changed=1 unreachable=0 failed=0 skipped=6
The key is 'rescued=0' and 'failed=0'..
```

使用解決方案部署 **ONTAP** 叢集

完成準備和規劃之後、您就可以使用 ONTAP Day 0/1 解決方案、使用 Ansible 快速設定 ONTAP 叢集。

在本節步驟中的任何時間、您都可以選擇測試要求、而非實際執行要求。若要測試要求、請將命令列上的教戰手冊變更 `site.yml`為 `logic.yml`。

`docs/tutorial-requests.txt`此位置包含此程序中所使用之所有服務要求的最終版本。如果執行服務要求時遇到困難、您可以將相關要求從檔案複製 `tutorial-requests.txt`到該 `playbooks/inventory/group_vars/all/tutorial-requests.yml`位置、並視需要修改硬編碼值（ IP 位址、集合名稱等）。接著您應該能夠成功執行要求。

開始之前

- 您必須安裝 Ansible 。
- 您必須已下載 ONTAP Day 0/1 解決方案、並將資料夾解壓縮至 Ansible 控制節點上所需的位置。
- ONTAP 系統狀態必須符合要求、而且您必須擁有必要的認證。
- 您必須已完成本節所述的所有必要工作"準備"。

本解決方案中的範例使用「 Cluster_01 」和「 Cluster_02 」作為兩個叢集的名稱。您必須使用環境中叢集的名稱來取代這些值。

步驟 **1** ：初始叢集組態

在此階段、您必須執行一些初始叢集組態步驟。

步驟

1. 瀏覽至該 `playbooks/inventory/group_vars/all/tutorial-requests.yml`位置、並檢閱 `cluster_initial`檔案中的要求。為您的環境進行任何必要的變更。
2. 在資料夾中建立服務要求的檔案 `logic-tasks`。例如，建立名為的檔案 `cluster_initial.yml`。

   將下列各行複製到新檔案：

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial cluster configuration
  set_fact:
    raw_service_request:
```

3. 定義 `raw_service_request`變數。

   您可以使用下列其中一個選項、在您在資料夾中建立的檔案 `logic-tasks`中定義 `raw_service_request`變數 `cluster_initial.yml`：

   。* 選項 1* ：手動定義 `raw_service_request`變數。

   使用編輯器開啟 `tutorial-requests.yml`檔案、並將內容從第 `11` 行複製到第 `165` 行。將內容貼到新檔案中的變數 `cluster_initial.yml`下方 `raw service request`、如下列範例所示：

顯示範例

範例 `cluster_initial.yml`檔案：

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial cluster configuration
  set_fact:
    raw_service_request:
     service:          cluster_initial
     operation:         create
     std_name:           none
     req_details:

      ontap_aggr:
      - hostname:                   "{{ cluster_name }}"
        disk_count:                 24
        name:                       n01_aggr1
        nodes:                      "{{ cluster_name }}-01"
        raid_type:                  raid4

      - hostname:                   "{{ peer_cluster_name }}"
        disk_count:                 24
        name:                       n01_aggr1
        nodes:                      "{{ peer_cluster_name }}-01"
        raid_type:                  raid4

      ontap_license:
      - hostname:                   "{{ cluster_name }}"
        license_codes:
        - XXXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXXAAAAAAAAAAAAAA
```

```
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA

    - hostname:                   "{{ peer_cluster_name }}"
      license_codes:
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
```

```
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
    - XXXXXXXXXXXXXXAAAAAAAAAAAAAA

ontap_motd:
- hostname:                     "{{ cluster_name }}"
  vserver:                      "{{ cluster_name }}"
  message:                      "New MOTD"

- hostname:                     "{{ peer_cluster_name }}"
  vserver:                      "{{ peer_cluster_name }}"
  message:                      "New MOTD"

ontap_interface:
- hostname:                     "{{ cluster_name }}"
  vserver:                      "{{ cluster_name }}"
  interface_name:               ic01
  role:                         intercluster
  address:                      10.0.0.101
  netmask:                      255.255.255.0
  home_node:                    "{{ cluster_name }}-01"
  home_port:                    e0c
  ipspace:                      Default
  use_rest:                     never

- hostname:                     "{{ cluster_name }}"
  vserver:                      "{{ cluster_name }}"
  interface_name:               ic02
  role:                         intercluster
  address:                      10.0.0.101
  netmask:                      255.255.255.0
  home_node:                    "{{ cluster_name }}-01"
  home_port:                    e0c
```

```
    ipspace:                    Default
    use_rest:                   never

  - hostname:                   "{{ peer_cluster_name }}"
    vserver:                    "{{ peer_cluster_name }}"
    interface_name:             ic01
    role:                       intercluster
    address:                    10.0.0.101
    netmask:                    255.255.255.0
    home_node:                  "{{ peer_cluster_name }}-01"
    home_port:                  e0c
    ipspace:                    Default
    use_rest:                   never

  - hostname:                   "{{ peer_cluster_name }}"
    vserver:                    "{{ peer_cluster_name }}"
    interface_name:             ic02
    role:                       intercluster
    address:                    10.0.0.101
    netmask:                    255.255.255.0
    home_node:                  "{{ peer_cluster_name }}-01"
    home_port:                  e0c
    ipspace:                    Default
    use_rest:                   never

  ontap_cluster_peer:
  - hostname:                   "{{ cluster_name }}"
    dest_cluster_name:          "{{ peer_cluster_name }}"
    dest_intercluster_lifs:     "{{ peer_lifs }}"
    source_cluster_name:        "{{ cluster_name }}"
    source_intercluster_lifs:   "{{ cluster_lifs }}"
    peer_options:
      hostname:                 "{{ peer_cluster_name }}"
```

- * 選項 2* ：使用忍者範本定義要求：

  您也可以使用下列 Jinja 範本格式來取得 `raw_service_request`值。

```
raw_service_request: "{{ cluster_initial }}"
```

4. 為第一個叢集執行初始叢集組態：

```
ansible-playbook -i inventory/hosts site.yml -e
cluster_name=<Cluster_01>
```

繼續之前、請確認沒有錯誤。

5. 對第二個叢集重複執行命令：

```
ansible-playbook -i inventory/hosts site.yml -e
cluster_name=<Cluster_02>
```

確認第二個叢集沒有錯誤。

往上捲動至 Ansible 輸出的開頭時、您應該會看到傳送至架構的要求、如下列範例所示：

顯示範例

```
TASK [Show the raw_service_request]
********************************************************************
****************************************
ok: [localhost] => {
    "raw_service_request": {
        "operation": "create",
        "req_details": {
            "ontap_aggr": [
                {
                    "disk_count": 24,
                    "hostname": "Cluster_01",
                    "name": "n01_aggr1",
                    "nodes": "Cluster_01-01",
                    "raid_type": "raid4"
                }
            ],
            "ontap_license": [
                {
                    "hostname": "Cluster_01",
                    "license_codes": [
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
```

```
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAAA"
                    ]
                }
            ],
            "ontap_motd": [
                {
                    "hostname": "Cluster_01",
                    "message": "New MOTD",
                    "vserver": "Cluster_01"
                }
            ]
        },
        "service": "cluster_initial",
        "std_name": "none"
    }
}
```

6. 登入每個 ONTAP 執行個體、並驗證要求是否成功。

步驟 **2** ：設定叢集間的生命週期

您現在可以將 LIF 定義新增至要求並定義微服務、 `ontap_interface`以設定叢集間的生命 `cluster_initial`體。

服務定義和要求會共同運作、以決定行動：

- 如果您提供的微服務服務請求不在服務定義中、則不會執行該要求。
- 如果您在服務定義中定義了一或多個微服務、但在要求中省略、則不會執行該要求。

教戰手冊會 `execution.yml`依所列順序掃描微服務清單、以評估服務定義：

- 如果要求中有一個項目的字典金鑰與微服務定義中包含的項目相符 `args`、則會執行該要求。
- 如果服務要求中沒有相符的項目、則會跳過該要求、而不會發生錯誤。

步驟

1. 瀏覽至 `cluster_initial.yml`您先前建立的檔案、並在要求定義中新增下列行以修改要求：

```
ontap_interface:
- hostname:                    "{{ cluster_name }}"
  vserver:                     "{{ cluster_name }}"
  interface_name:              ic01
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ cluster_name }}"
  vserver:                     "{{ cluster_name }}"
  interface_name:              ic02
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ peer_cluster_name }}"
  vserver:                     "{{ peer_cluster_name }}"
  interface_name:              ic01
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ peer_cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ peer_cluster_name }}"
  vserver:                     "{{ peer_cluster_name }}"
  interface_name:              ic02
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ peer_cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never
```

2. 執行命令：

```
ansible-playbook -i inventory/hosts  site.yml -e
cluster_name=<Cluster_01> -e peer_cluster_name=<Cluster_02>
```

3. 登入每個執行個體、檢查是否已將生命期新增至叢集：

   顯示範例

```
Cluster_01::> net int show
  (network interface show)
            Logical    Status     Network            Current
Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node
Port    Home
----------- ---------- ---------- ------------------ -------------
------- ----
Cluster_01
            Cluster_01-01_mgmt up/up 10.0.0.101/24   Cluster_01-01
e0c     true
            Cluster_01-01_mgmt_auto up/up 10.101.101.101/24
Cluster_01-01 e0c true
            cluster_mgmt up/up    10.0.0.110/24      Cluster_01-01
e0c     true
5 entries were displayed.
```

   輸出顯示已添加 * 非 * 的生命。這是因為 `ontap_interface`仍需要在檔案中定義微服務 `services.yml`。

4. 確認已將生命項新增至 `raw_service_request`變數。

顯示範例

以下範例顯示已將生命提升新增至要求：

```
            "ontap_interface": [
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_01-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_01",
                     "interface_name": "ic01",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_01"
                 },
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_01-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_01",
                     "interface_name": "ic02",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_01"
                 },
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_02-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_02",
                     "interface_name": "ic01",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_02"
                 },
                 {
                     "address": "10.0.0.126",
                     "home_node": "Cluster_02-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_02",
```

```
                    "interface_name": "ic02",
                    "ipspace": "Default",
                    "netmask": "255.255.255.0",
                    "role": "intercluster",
                    "use_rest": "never",
                    "vserver": "Cluster_02"
                }
            ],
```

5. 在檔案中 `services.yml`的下定義 `ontap_interface`微服務 `cluster_initial`。

   將下列各行複製到檔案中以定義微服務：

```
        - name: ontap_interface
          args: ontap_interface
          role: na/ontap_interface
```

6. 現在已 `ontap_interface`在要求和檔案中定義微服務 `services.yml`、請再次執行要求：

```
ansible-playbook -i inventory/hosts  site.yml -e
cluster_name=<Cluster_01> -e peer_cluster_name=<Cluster_02>
```

7. 登入每個 ONTAP 執行個體、並確認已新增生命。

步驟 **3** ：選擇性地設定多個叢集

如果需要、您可以在同一個要求中設定多個叢集。定義要求時、您必須為每個叢集提供變數名稱。

步驟

1. 在檔案中新增第二個叢集的項目 `cluster_initial.yml`、以便在同一個要求中設定兩個叢集。

   下列範例顯示 `ontap_aggr`新增第二個項目之後的欄位。

```
ontap_aggr:
 - hostname:                    "{{ cluster_name }}"
   disk_count:                  24
   name:                        n01_aggr1
   nodes:                       "{{ cluster_name }}-01"
   raid_type:                   raid4

 - hostname:                    "{{ peer_cluster_name }}"
   disk_count:                  24
   name:                        n01_aggr1
   nodes:                       "{{ peer_cluster_name }}-01"
   raid_type:                   raid4
```

2. 對下的所有其他項目套用變更 `cluster_initial`。
3. 將下列各行複製到檔案中、將叢集對等關係新增到要求：

```
ontap_cluster_peer:
- hostname:                     "{{ cluster_name }}"
  dest_cluster_name:            "{{ cluster_peer }}"
  dest_intercluster_lifs:       "{{ peer_lifs }}"
  source_cluster_name:          "{{ cluster_name }}"
  source_intercluster_lifs:     "{{ cluster_lifs }}"
  peer_options:
    hostname:                   "{{ cluster_peer }}"
```

4. 執行 Ansible 要求：

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01>
site.yml -e peer_cluster_name=<Cluster_02> -e
cluster_lifs=<cluster_lif_1_IP_address,cluster_lif_2_IP_address>
-e peer_lifs=<peer_lif_1_IP_address,peer_lif_2_IP_address>
```

步驟 **4** ：初始 **SVM** 組態

在本程序的這個階段、您可以在叢集中設定 SVM 。

步驟

1. 更新 `svm_initial`**檔案中的要求** `tutorial-requests.yml`、以設定 SVM 和 SVM 對等關係。

   您必須設定下列項目：

   - SVM

- SVM 對等關係
- 每個 SVM 的 SVM 介面

2. 更新要求定義中的變數定義 `svm_initial`。您必須修改下列變數定義：
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   若要更新定義、請在 `svm_initial`定義之後移除 * 「 {} 」 * 、然後 `req_details`新增正確的定義。

3. 在資料夾中建立服務要求的檔案 `logic-tasks`。例如，建立名為的檔案 `svm_initial.yml`。

   將下列各行複製到檔案：

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial SVM configuration
  set_fact:
    raw_service_request:
```

4. 定義 `raw_service_request`變數。

   您可以使用下列其中一個選項、在資料夾中 `logic-tasks`定義 `raw_service_request`的變數 `svm_initial`：

   - * 選項 1* ：手動定義 `raw_service_request`變數。

     使用編輯器開啟 `tutorial-requests.yml`檔案、並將內容從第 179 行複製到第 222 行。將內容貼到新檔案中的變數 `svm_initial.yml`下方 `raw service request`、如下列範例所示：


service: svm_initial
std_name: none
req_details:
ontap_vserver:
- hostname: "{{ cluster_name }}"
name: "{{ vserver_name }}"
root_volume_aggregate: n01_aggr1
- hostname: "{{ peer_cluster_name }}"
name: "{{ peer_vserver }}"
root_volume_aggregate: n01_aggr1

顯示範例

範例 `svm_initial.yml`檔案：

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial SVM configuration
  set_fact:
    raw_service_request:
     service:          svm_initial
     operation:        create
     std_name:         none
     req_details:

      ontap_vserver:
      - hostname:                   "{{ cluster_name }}"
        name:                       "{{ vserver_name }}"
        root_volume_aggregate:      n01_aggr1

      - hostname:                   "{{ peer_cluster_name }}"
       name:                       "{{ peer_vserver }}"
       root_volume_aggregate:      n01_aggr1

      ontap_vserver_peer:
      - hostname:                   "{{ cluster_name }}"
        vserver:                    "{{ vserver_name }}"
        peer_vserver:               "{{ peer_vserver }}"
        applications:               snapmirror
        peer_options:
          hostname:                 "{{ peer_cluster_name }}"

      ontap_interface:
```

```
    - hostname:                    "{{ cluster_name }}"
      vserver:                     "{{ vserver_name }}"
      interface_name:              data01
      role:                        data
      address:                     10.0.0.200
      netmask:                     255.255.255.0
      home_node:                   "{{ cluster_name }}-01"
      home_port:                   e0c
      ipspace:                     Default
      use_rest:                    never

    - hostname:                    "{{ peer_cluster_name }}"
      vserver:                     "{{ peer_vserver }}"
      interface_name:              data01
      role:                        data
      address:                     10.0.0.201
      netmask:                     255.255.255.0
      home_node:                   "{{ peer_cluster_name }}-01"
      home_port:                   e0c
      ipspace:                     Default
      use_rest:                    never
```

。* 選項 2* ：使用忍者範本定義要求：

您也可以使用下列 Jinja 範本格式來取得 `raw_service_request`值。

```
raw_service_request: "{{ svm_initial }}"
```

5. 執行要求：

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01> -e
peer_cluster_name=<Cluster_02> -e peer_vserver=<SVM_02>  -e
vserver_name=<SVM_01> site.yml
```

6. 登入每個 ONTAP 執行個體並驗證組態。
7. 新增 SVM 介面。

在檔案中的 `services.yml`下定義 `ontap_interface`服務 `svm_initial`、然後再次執行要求：

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01> -e
peer_cluster_name=<Cluster_02> -e peer_vserver=<SVM_02>  -e
vserver_name=<SVM_01> site.yml
```

8. 登入每個 ONTAP 執行個體、並確認 SVM 介面已設定完成。

步驟 **5** ：選擇性地動態定義服務要求

在先前的步驟中、 `raw_service_request`變數是硬式編碼的。這對學習、開發和測試都很有用。您也可以動態產生服務要求。

如果您不想將其與較高層級的系統整合、則下節提供動態產生所需的選項 `raw_service_request`。

- 如果未在命令中定義變數、 `logic.yml`則 `logic_operation`檔案不會從資料夾匯入任何檔案 `logic-tasks`。這表示`raw_service_request`必須在 Ansible 之外定義、並提供給執行架構。
- 資料夾中的工作檔案名稱必須符合不含 .yml 副檔名 `logic-tasks`的變數值 `logic_operation`。
- 資料夾中的工作檔案 `logic-tasks`會動態定義 `raw_service_request`。唯一的需求是將有效`raw_service_request`定義為相關檔案中的最後一項工作。

如何動態定義服務要求

有多種方法可以套用邏輯工作來動態定義服務要求。以下列出其中一些選項：

- 使用資料夾中的 Ansible 工作檔案 `logic-tasks`
- 啟動可傳回適合轉換為 varaible 之資料的自訂角色 `raw_service_request`。
- 在 Ansible 環境以外調用其他工具以提供所需的資料。例如、 REST API 呼叫 Active IQ Unified Manager 。

下列命令範例會使用檔案動態定義每個叢集的服務要求 `tutorial-requests.yml`：

```
ansible-playbook -i inventory/hosts -e cluster2provision=Cluster_01
-e logic_operation=tutorial-requests site.yml
```

```
ansible-playbook -i inventory/hosts -e cluster2provision=Cluster_02
-e logic_operation=tutorial-requests site.yml
```

步驟 **6** ：部署 **ONTAP Day 0/1** 解決方案

在此階段、您應該已經完成下列工作：

- 根據您的需求檢閱及修改中的所有檔案 `playbooks/inventory/group_vars/all`。每個檔案中都有詳細的註解、可協助您進行變更。
- 已將任何必要的工作檔案新增至 `logic-tasks`目錄。
- 已將任何必要的資料檔案新增至 `playbook/vars`目錄。

使用下列命令部署 ONTAP Day 0/1 解決方案、並驗證部署的健全狀況：

在此階段、您應該已經解密並修改 `vault.yml`檔案、而且必須使用新密碼來加密。

- 執行 ONTAP Day 0 服務：

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_day_0 -e service=cluster_day_0 -vvvv --ask-vault
-pass <your_vault_password>
```

- 執行 ONTAP Day 1 服務：

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_day_1 -e service=cluster_day_0 -vvvv --ask-vault
-pass <your_vault_password>
```

- 套用叢集整體設定：

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_wide_settings -e service=cluster_wide_settings
-vvvv --ask-vault-pass <your_vault_password>
```

- 執行健全狀況檢查：

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=health_checks -e service=health_checks -e
enable_health_reports=true -vvvv --ask-vault-pass <your_vault_password>
```

自訂 **ONTAP Day 0/1** 解決方案

若要根據您的需求自訂 ONTAP Day 0/1 解決方案、您可以新增或變更 Ansible 角色。

角色代表 Ansible 架構內的微服務。每項微服務都會執行一項作業。例如、 ONTAP Day 0 是包含多個微服務的服務。

新增 **Ansible** 角色

您可以新增 Ansible 角色、針對您的環境自訂解決方案。必要角色是由 Ansible 架構內的服務定義所定義。

角色必須符合下列需求、才能用作微服務：

- 接受變數中的引數清單 `args`。
- 使用 Ansible 「區塊、救援、永遠」結構、並針對每個區塊提供特定需求。
- 使用單一 Ansible 模組、在區塊內定義單一工作。
- 根據本節所詳述的要求實作每個可用的模組參數。

必要的微服務架構

每個角色都必須支援下列變數：

- `mode`：如果模式設置爲角色，則 `test`會嘗試導入，以顯示角色在未實際執行的情況下執行的 `test.yml`操作。

  

  由於某些相依性、因此不一定能實作此項目。

- `status`：教戰手冊執行的整體狀態。如果值未設定為角色、則 `success`不會執行。
- `args`：具有與角色參數名稱匹配的關鍵字的角色特定字典列表。
- `global_log_messages`：在執行教戰手冊期間收集記錄訊息。每次執行角色時都會產生一個項目。
- `log_name`：用於引用條目的角色的名稱 `global_log_messages`。
- `task_descr`：職務內容的簡短說明。
- `service_start_time`：用於追蹤每個角色執行時間的時間戳記。
- `playbook_status`： Ansible 教戰手冊的狀態。
- `role_result`：包含角色輸出的變數、會包含在項目中的每則訊息 `global_log_messages`中。

角色結構範例

以下範例提供實作微服務之角色的基本結構。您必須針對組態變更本範例中的變數。

顯示範例

基本角色結構：

```
- name:  Set some role attributes
  set_fact:
    log_name:     "<LOG_NAME>"
    task_descr:   "<TASK_DESCRIPTION>"

-  name: "{{ log_name }}"
   block:
      -  set_fact:
            service_start_time: "{{ lookup('pipe', 'date
+%Y%m%d%H%M%S') }}"

      -  name: "Provision the new user"
         <MODULE_NAME>:

#-------------------------------------------------------------
            # COMMON ATTRIBUTES

#-------------------------------------------------------------
            hostname:            "{{
clusters[loop_arg['hostname']]['mgmt_ip'] }}"
            username:            "{{
clusters[loop_arg['hostname']]['username'] }}"
            password:            "{{
clusters[loop_arg['hostname']]['password'] }}"

            cert_filepath:       "{{ loop_arg['cert_filepath']
| default(omit) }}"
            feature_flags:       "{{ loop_arg['feature_flags']
| default(omit) }}"
            http_port:           "{{ loop_arg['http_port']
| default(omit) }}"
            https:               "{{ loop_arg['https']
| default('true') }}"
            ontapi:              "{{ loop_arg['ontapi']
| default(omit) }}"
            key_filepath:        "{{ loop_arg['key_filepath']
| default(omit) }}"
            use_rest:            "{{ loop_arg['use_rest']
| default(omit) }}"
            validate_certs:      "{{ loop_arg['validate_certs']
| default('false') }}"
```

```
            <MODULE_SPECIFIC_PARAMETERS>

#-------------------------------------------------------------
            # REQUIRED ATTRIBUTES

#-------------------------------------------------------------
            required_parameter:     "{{ loop_arg['required_parameter']
}}"

#-------------------------------------------------------------
            # ATTRIBUTES w/ DEFAULTS

#-------------------------------------------------------------
            defaulted_parameter:    "{{ loop_arg['defaulted_parameter']
| default('default_value') }}"

#-------------------------------------------------------------
            # OPTIONAL ATTRIBUTES

#-------------------------------------------------------------
            optional_parameter:     "{{ loop_arg['optional_parameter']
| default(omit) }}"
         loop:    "{{ args }}"
         loop_control:
            loop_var:   loop_arg
         register:   role_result

   rescue:
      -  name: Set role status to FAIL
         set_fact:
            playbook_status:   "failed"

   always:
      -  name: add log msg
         vars:
            role_log:
               role: "{{ log_name }}"
               timestamp:
                  start_time: "{{service_start_time}}"
                  end_time: "{{ lookup('pipe', 'date +%Y-%m-
%d@%H:%M:%S') }}"
               service_status: "{{ playbook_status }}"
               result: "{{role_result}}"
         set_fact:
            global_log_msgs:   "{{ global_log_msgs + [ role_log ] }}"
```

範例角色中使用的變數：

- <NAME>：必須為每個微服務提供可更換的值。
- <LOG_NAME>：用於記錄的角色的簡短名稱。例如 ONTAP_VOLUME：。
- <TASK_DESCRIPTION>：微服務的功能簡介。
- <MODULE_NAME>：任務的 Ansible 模塊名稱。

> 最上層的 `execute.yml`教戰手冊會指定 `netapp.ontap`集合。如果模組是集合的一部分、則 `netapp.ontap`不需要完整指定模組名稱。

- <MODULE_SPECIFIC_PARAMETERS>：特定於用於實施微服務的模塊的可接受模塊參數。下列清單說明參數類型及其分組方式。
  - 必要參數：指定所有必要參數時不使用預設值。
  - 具有微服務特定預設值的參數（與模組文件所指定的預設值不同）。
  - 所有剩餘參數都會用 `default(omit)`作預設值。

使用多層字典做為模組參數

某些 NetApp 提供的 Ansible 模組會使用多層級字典來處理模組參數（例如、固定和調適性 QoS 原則群組）。

使用這些字典時、單獨使用 `default(omit)`並不適用、尤其是當有多個字典且彼此互斥時。

如果您需要使用多層字典做為模組參數、則應將功能分割成多個微服務（角色）、以保證每個字典都能為相關字典提供至少一個二層字典值。

下列範例顯示固定和自適應 QoS 原則群組、可在兩個微服務之間分割。

第一個微服務包含固定的 QoS 原則群組值：

```
fixed_qos_options:
  capacity_shared:           "{{
loop_arg['fixed_qos_options']['capacity_shared']         | default(omit)
}}"
  max_throughput_iops:       "{{
loop_arg['fixed_qos_options']['max_throughput_iops']     | default(omit)
}}"
  min_throughput_iops:       "{{
loop_arg['fixed_qos_options']['min_throughput_iops']     | default(omit)
}}"
  max_throughput_mbps:       "{{
loop_arg['fixed_qos_options']['max_throughput_mbps']     | default(omit)
}}"
  min_throughput_mbps:       "{{
loop_arg['fixed_qos_options']['min_throughput_mbps']     | default(omit)
}}"
```

第二個微服務包含調適性 QoS 原則群組值：

```
adaptive_qos_options:
  absolute_min_iops:          "{{
loop_arg['adaptive_qos_options']['absolute_min_iops'] | default(omit) }}"
  expected_iops:              "{{
loop_arg['adaptive_qos_options']['expected_iops']     | default(omit) }}"
  peak_iops:                  "{{
loop_arg['adaptive_qos_options']['peak_iops']         | default(omit) }}"
```