仮想ネットワーク機能のスケーリング

• ETSI API を使用した仮想ネットワーク機能のスケーリング （1 ページ）

ETSI API を使用した仮想ネットワーク機能のスケーリング

ESC の主な利点の 1 つは、サービスを柔軟に拡張できることです。これにより、VNF 内で特定のロールまたはアスペクトを実行する VNFC が、要求を処理し、高い需要を満たすためにスケールアウトしたり、使用率が低い場合にスケールインしたりできます。このアスペクトは、複数の VNFC に広がる場合があります。

スケーリング要求は手動でも自動でもかまいません。スケーリングを実現するためのさまざまなアプローチについて、以下で詳しく説明します。

これらの概念と仕様の詳細については、*ETSI GS NFV-SOL 003* の Annex B を参照してください。

REST および NETCONF API を使用した VNF のスケーリングについては、『*Cisco Elastic Services Controller User Guide*』を参照してください。

拡張性

VNF のスケーリング要求は、VnfInstance リソースをクエリするときに instantiatedVnfInfo の一部として見つかる属性である *scaleStatus* を使用します。この属性は、VNF の各アスペクトの現在のスケールレベルを示します。次に例を示します。

```
"scaleInfo": [
  {
    "aspectId": "webserver", "scaleLevel": "4"
  },
  {
    "aspectId": "processing", "scaleLevel": "2"
  }
]
```

これは VNF のスケーリング要求の開始点を形成します。これにより、1 つのアスペクトを、VNF のその寸法において、現在の *scaleLevel* に対して水平方向にスケーリング（VNFC を追加

または削除）できます。アスペクトのスケーリング操作は、そのアスペクトをサポートする各 VNFC に適用されます。

（注） 現在の仕様では、垂直スケーリング（既存の VNFC インスタンスへのリソースの追加/削除）はサポートされていません。

要求ペイロード（ETSI データ構造：ScaleVNFRequest）

```
{
    "type": "SCALE_OUT",
    "aspectId": "processing",
    "numberOfSteps": 1,
    "additionalParams": {}
}
```

上記のペイロードにより、上記の *scaleStatus* の例は更新され、scaleLevel 3 にスケールアウトするために必要なこの手順において、VNFC の数が追加されます。

```
"scaleInfo": [
  {
    "aspectId": "webserver", "scaleLevel": "4"
  },
  {
    "aspectId": "processing", "scaleLevel": "3"
  }
]
```

スケーリング手順およびスケーリングをサポートするその他の関連ポリシーについては、「スケーリングの VNFD ポリシー」を参照してください。

レベルへのスケーリング

Scale VNF が提供する相対的なスケーリングではなく、VNF をレベルにスケーリングする要求は、求められる絶対的なスケーリング結果を指定します。その結果、一部のアスペクトはスケールアウトされ、その他のアスペクトはスケールインされます。このオプションは、スケーリングで必要な 2 つのアプローチのうちの 1 つを使用します。

- インスタンス化レベル
- スケールレベル

これらは相互に排他的であり、1 つの要求で複数のアスペクトをスケーリングできます。

インスタンス化レベル

インスタンス化レベルは各アスペクトに事前に定義されたサイズで、各レベルには各アスペクトに関連付けられたスケールレベルがあります。これ以上の細分性は提供されないため、VNF 全体（すなわちすべてのアスペクト）が、要求されるインスタンス化レベルに従ってスケーリングされます。

例：

要求ペイロード（ETSI データ構造：ScaleVNFToLevelRequest）

```
{
    "instantiationLevelId": "premium"
}
```

インスタンス化レベルの定義については、VNFD ポリシーを参照してください。

スケールレベル

スケールレベルもまた各アスペクトに事前定義されたサイズで、各アスペクトにはターゲット VNFC、定義された step_deltas（各スケーリングステップは均一ではない可能性があるため）、最大スケールレベルがあります。このオプションを定義するポリシーでは、ターゲットごとに異なるスケーリング結果を使用できます。

（注） スケールレベルは VM の数を表すものではありません。たとえば、scaleLevel=0 はターゲット VNFC 上のそのアスペクトのインスタンスの初期数（初期デルタ）を意味し、scaleLevel=1 は初期デルタに、そのアスペクトと VNFC タプルで定義した最初のスケーリングステップを加えたものです。

要求ペイロード（ETSI データ構造：ScaleVNFToLevelRequest）

```
{
    "scaleInfo": [
      {
            "aspectId": "processing",
            "scaleLevel": "2"
      },
      {
            "aspectId": "webserver",
            "scaleLevel": "3"
      }
    ]
}
```

スケールレベルの定義については、「スケーリングの VNFD ポリシー」を参照してください。

スケーリングの VNFD ポリシー

VNF の全体的なスケーリング動作を作るポリシーは多数あります。これらのポリシーは、上記のさまざまなスケーリングアプローチをサポートします。最初のポリシーは、スケーリングされる（またはスケーリングされない）アスペクトを定義します。

```
policies:
    - scaling_aspects:
        type: tosca.policies.nfv.ScalingAspects
        properties:
          aspects:
            webserver:
              name: 'webserver'
              description: 'The webserver cluster.'
              max_scale_level: 5
              step_deltas:
```

```
        - delta_1
    processing:
      name: 'processing'
      description: 'An example processing function'
      max_scale_level: 3
      step_deltas:
        - delta_1
        - delta_2
        - delta_1
    database:
      name: 'database'
      description: 'A test database'
      max_scale_level: 0
```

この例では、データベースアスペクトの max_scale_level が 0 であることがわかります。これはスケールアウトできないことを意味し、そのアスペクトのインスタンスが 0 であることを意味するわけではありません。理由については、以下のアルゴリズムを参照してください。Web サーバのアスペクトには 1 つの step_delta しかありません。つまり、すべてのスケーリングステップが均一であるのに対し、処理アスペクトにはスケーリングステップごとに異なる step_delta が指定されます。これは不均一スケーリングと呼ばれます。これはこの VNF のアスペクトの宣言にすぎず、これはスケーリング要求を受信したときに検証を実行するために使用されるポリシーの 1 つです。

次に、動作を制御するためにこれらを VNFC に適用する必要があります。

```
- db_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu1 ]

    - ws_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu2, vdu4 ]

    - pc_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu3 ]

    - ws_scaling_aspect_deltas:
        type: tosca.policies.nfv.VduScalingAspectDeltas
        properties:
          aspect: webserver
          deltas:
            delta_1:
              number_of_instances: 1
        targets: [ vdu2, vdu4 ]

    - pc_scaling_aspect_deltas:
        type: tosca.policies.nfv.VduScalingAspectDeltas
        properties:
          aspect: processing
          deltas:
            delta_1:
              number_of_instances: 1
```

```
            delta_2:
              number_of_instances: 2
        targets: [ vdu2, vdu4 ]
```

上記の例では、VNFC がターゲットとして識別されています。アスペクトは VNFCS ごとに異なる動作の場合がありますが、ここでは示しません。スケーリング要求の検証と生成に使用される step_deltas の定義もここに示します（これらの手順は要求されるスケールレベルによって推測されます）。VNFC のインスタンスの最小数は常に 0 と仮定され、最大数は次のアルゴリズムによって計算されます。

initial_delta に max_scale_level までの逓増する各インスタンス数を足したもの。

これらのポリシーは、スケールレベルベースのスケーリングと見なされます。インスタンス化レベルに基づくスケーリングには、同様の構成が使用されます。

```
- instantiation_levels:
        type: tosca.policies.nfv.InstantiationLevels
        properties:
          levels:
            default:
              description: 'Default instantiation level'
              scale_info:
                database:
                  scale_level: 0
                webserver:
                  scale_level: 0
                processing:
                  scale_level: 0
            premium:
              description: 'Premium instantiation level'
              scale_info:
                database:
                  scale_level: 0
                webserver:
                  scale_level: 2
                processing:
                  scale_level: 3
          default_level: default
```

スケーリングアスペクトと同様に、インスタンス化レベルの定義の最初の部分は単なる宣言です。ここでは、各アスペクトはすでに宣言されている必要があり、その後、各アスペクトの scale_level はインスタンス化レベルで宣言されます。デフォルトのインスタンス化レベルは、他に何も指定されていない場合にも指定されます。各 VNFC の各 scale_level の意味は、VduInstantiationLevels ポリシーでさらに詳しく説明されています。次に例を示します。

```
- ws_instantiation_levels:
        type: tosca.policies.nfv.VduInstantiationLevels
        properties:
          levels:
            default:
              number_of_instances: 1
        targets: [ vdu2, vdu4 ]
```

したがって、これらのポリシーは、デフォルトのインスタンス化レベルが「default」であり、その結果 Web サーバのアスペクトが、scale_level 0（1 VNFC インスタンス）でインスタンス化されることを示します。

複数の IP アドレスへの依存

スタティック IP アドレス

VNFC に静的 IP アドレスが設定された接続ポイントがある場合、新たにスピンアップされた VNFC インスタンスの接続ポイントに割り当てる IP アドレスがないため、VNFC を拡張できません。代わりに、インスタンス化要求で静的 IP アドレスのプールを指定するか、リストとして応答（extVirtualLinks 要素内）を付与できます。

- 1 つの cpProtocolDatain の fixedAddresses
- 複数の cpProtocolData の個別の fixedAddresses

（注） 1 つの cpProtocolData の ipAddresses のリストは、すべての IP アドレスを 1 つの VNFC インスタンスの 1 つのポートに割り当てます。

または、ipAddresses エントリで連続した範囲を addressRange として指定することもできます。特定の IP アドレスを指定する必要がない場合は、仮想ネットワーク機能のインスタンス化の例に従って、subnetId を使用できます。

次の例では、1 つの cpProtocolData の fixedAddresses で IPアドレスをリストとして指定することで、4 つの IP アドレスを持つ静的 IP プールを作成する方法を説明します。

```
  {
…
"extVirtualLinks": [
  {
    "id": "extVL-dbf477ad-199a-47ff-939a-cb0101c92585",
    "resourceId": "ext-net",
    "extCps": [
      {
        "cpdId": "ecp_1_vdu_node_1",
        "cpConfig": [
          {
            "cpProtocolData": [
              {
                "layerProtocol": "IP_OVER_ETHERNET",
                "ipOverEthernet": {
                  "ipAddresses": [
                    {
                      "type": "IPV4",
                      "fixedAddresses": [
                        "172.16.0.10",
                        "172.16.0.11",
                        "172.16.0.12",
                        "172.16.0.13"
                      ]
                    }
                  ]
                }
              }
            ]
          }
        ]
      }
```

```
      ]
    }
  ]
  …
}
```

IP アドレスの静的プールは、複数の cpProtocolData で個別の fixedAddresses として指定することによっても作成できます。

```
{
  …
  "extVirtualLinks": [
    {
      "id": "extVL-dbf477ad-199a-47ff-939a-cb0101c92585",
      "resourceId": "ext-net",
      "extCps": [
        {
          "cpdId": "ecp_1_vdu_node_1",
          "cpConfig": [
            {
              "cpProtocolData": [
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.10"
                        ]
                      }
                    ]
                  }
                },
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.11"
                        ]
                      }
                    ]
                  }
                },
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.12"
                        ]
                      }
                    ]
                  }
                },
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
```

```
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.13"
                        ]
                      }
                    ]
                  }
                }
              ]
            }
          ]
        }
      ]
    }
  ]
  …
}
```

addressRange を使用して作成された IP アドレスの静的プール：

```
{
  …
  "extVirtualLinks": [
    {
      "id": "extVL-dbf477ad-199a-47ff-939a-cb0101c92585",
      "resourceId": "ext-net",
      "extCps": [
        {
          "cpdId": "ecp_1_vdu_node_1",
          "cpConfig": [
            {
              "cpProtocolData": [
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "addressRange": {
                          "minAddress": "172.16.0.10",
                          "maxAddress": "172.16.0.13"
                        }
                      }
                    ]
                  }
                }
              ]
            }
          ]
        }
      ]
    }
  ]
  …
}
```

これらの IP アドレスプールの実装は、*ETSI NFV MANO SOL003* 仕様書の *4.4.1.10* 章に準拠します。

スタティック MAC アドレス

VNFC に静的 MAC アドレスが設定された接続ポイントがある場合、新たにスピンアップされた VNFC インスタンスの接続ポイントに割り当てる MAC アドレスがないため、VNFC を拡張できません。代わりに、追加の静的 MAC アドレスのプールをインスタンス化要求で指定するか、応答を付与できます。

静的 MAC アドレスプールは、複数の cpProtocolData で macAddress を指定することによって、インスタンス化要求の extVirtualLinks 要素で作成するか、応答を付与できます。

次の例では、複数の cpProtocolData で MAC アドレスを指定することで、4 つの MAC アドレスを持つ静的 MAC プールを作成する方法を説明します。

```
{
  …
  "extVirtualLinks": [
    {
      "id": "extVL-dbf477ad-199a-47ff-939a-cb0101c92585",
      "resourceId": "ext-net",
      "extCps": [
        {
          "cpdId": "ecp_1_vdu_node_1",
          "cpConfig": [
            {
              "cpProtocolData": [
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "macAddress": "fa:16:3e:0b:10:10",
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.10"
                        ]
                      }
                    ]
                  }
                },
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "macAddress": "fa:16:3e:0b:10:11",
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.11"
                        ]
                      }
                    ]
                  }
                },
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "macAddress": "fa:16:3e:0b:10:12",
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "fixedAddresses": [
                          "172.16.0.12"
                        ]
```

```
                        }
                      ]
                    }
                  },
                  {
                    "layerProtocol": "IP_OVER_ETHERNET",
                    "ipOverEthernet": {
                      "macAddress": "fa:16:3e:0b:10:13",
                      "ipAddresses": [
                        {
                          "type": "IPV4",
                          "fixedAddresses": [
                            "172.16.0.13"
                          ]
                        }
                      ]
                    }
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
    …
}
```

デイゼロ設定

VNF を展開後、展開サービスの VNFC インスタンスに day 0 の変数が設定されます。多くの場合、day 0 の設定値は一定です。それ以外の場合、day 0 のパラメータに指定される値のリソースプールがあり、新しい VNFC インスタンスに新しい値を割り当てられます。

VNFD の vendor_section 内の Day 0 の設定：

```
    vdu3:
      type: cisco.nodes.nfv.Vdu.Compute
      properties:
        name: 'Processing1'
        description: 'Processing VNFC'
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 5
        vendor_section:
          cisco_esc:
            config_data:
              '/tmp/OSRESTTestETSIDay0_Inline_data.cfg':
                data: |
                  NODE_NAME $NODE_NAME
                  NUM_OF_CPU $NUM_OF_CPU
                  MEM_SIZE $MEM_SIZE
                  PROXY_ADDRS $PROXY_ADDRS
                  SPECIAL_CHARS $SPECIAL_CHARS
                variables:
                  NODE_NAME: vdu_node_1
                  NUM_OF_CPU: 1
                  MEM_SIZE: 1GB
                  PROXY_ADDRS: ["1.1.1.1", "1.1.2.1", "1.1.3.1", "1.1.4.1", "1.1.5.1",
"1.1.6.1", "1.1.7.1"]
                  SPECIAL_CHARS: '`~!@#$%^&*()-_=+[{]}|;:<.>/?'
```

上記の例では、day 0 の設定はインラインで指定されており、速度変数はターゲット設定で定義されています。これらの各変数は、1 つ以上の値を持つ変数によってサポートされます。$PROXY_ADDRS 変数の複数の値をサポートするため、値のリストが提供されます。これらの値は、VNFC の新しいインスタンスの変数を後続で使用する際に事前入力するために使用されます。

展開モデルの day 0 の設定の詳細については、『*Cisco Elastic Services Controller User Guide*』の「Day Zero Configuration」を参照してください。

VNF の自動スケーリング

VNFD で定義される KPI、ルール、およびアクションによって、スケーリングを考慮する必要がある条件が決まります。詳細については、「仮想ネットワーク機能のモニタリング」を参照してください。スケーリングポリシーは、許可されるスケーリング境界を制御するいくつかのポリシータイプを使用して、VNFD でも定義されます。次に、これらのポリシー項目について説明します。

展開後、ESC は各 VNFC をモニタするために、KPI を使用してモニタリングエージェント（これは集中管理型インスタンスまたは分散型インスタンスの場合があります）を設定します。KPI がしきい値に達すると、スケーリングワークフローが開始されます。定義されたアクションに基づいて、ESC はスケールインまたはスケールアウトを実行し、適切な通知とイベントログを生成します。これは、ログやオンボードスクリプトなど、指定できる一部の組み込み関数に従います。

ESC は、サブスクライブされたコンシューマに適切な通知を送信します。この時点で、ESC は *isAutoscaleEnabled* フラグについて、VNF インスタンスリソースに問い合わせます（これは最初に VNFD の値によって設定されますが、作成後に変更できます）。このフラグが true に設定されている場合、ESC はスケーリングワークフローを呼び出します（*ScaleVnfToLevelRequest* を使用して問い合わせ、1 つの要求で複数のアスペクトのスケーリングを要求します）。*isAutoscaleEnabled* が false に設定されている場合、上記の要求を使用して、制御は目的のアクションをトリガーするために、NFVO や EM などの外部システムを使用します。

（注） 自動スケーリングまたは自動修復要求の作成中は、新しい外部要求はブロックされます。ブロックされた要求の対応する応答と問題の詳細がユーザに通知されます。