OpenShift 4.19 を OpenStack 17.1 にインストールする

改訂：2026 年 4 月 8 日

新機能および変更された機能に関する情報

次の表は、この最新リリースまでの主な変更点の概要を示したものです。ただし、今リリースまでの変更点や新機能の一部は表に記載されていません。

Cisco ACI CNI プラグインのリリース バージョン	機能
6.1(1)	Cisco Application Centric Infrastructure（ACI）は、Red Hat OpenStack Platform（OSP）にネストされた Red Hat OpenShift 4.19 をサポートします。

OpenStack 上の OpenShift 4.19

Cisco Application Centric Infrastructure（ACI）は、Red Hat OpenStack Platform（OSP）17.1 にネストされた Red Hat OpenShift 4.19 をサポートします。このサポートを有効にするために、Cisco ACI は、アップストリームの OpenShift インストーラを補完するカスタマイズされた Ansible モジュールを提供します。このドキュメントでは、次のドキュメント類に記載されている OpenStack User-Provisioned Infrastructure（UPI）での OpenShift の推奨インストール プロセスに従う手順とガイダンスを提供します。

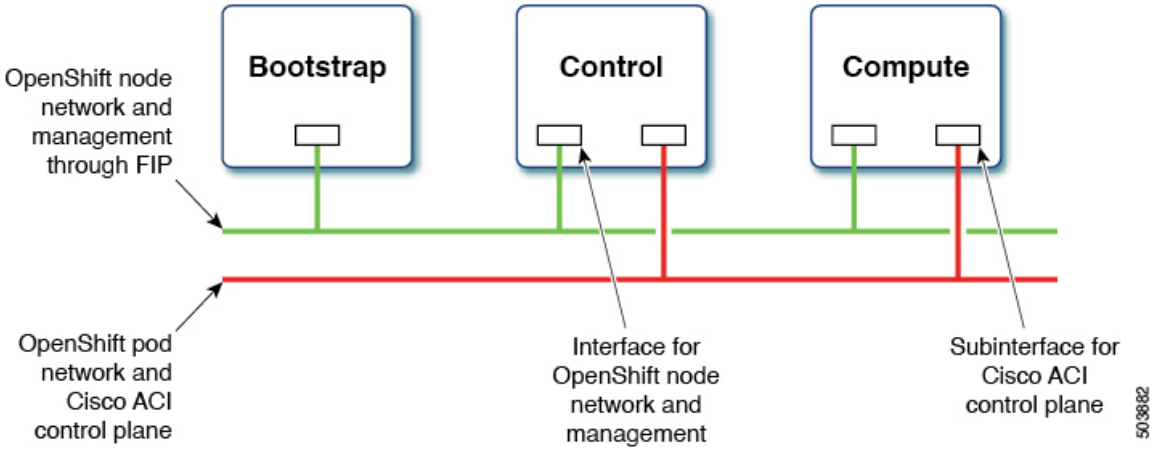
- OpenShift 4.19 に合わせてカスタマイズされた *OpenStack* を クラスタにインストールする（Red Hat OpenShift Web サイト）
- GitHub の *OpenStack* ユーザー プロビジョニング インフラストラクチャへの *OpenShift* のインストール

（注） Cisco ACI CNI を使用して既存の OpenShift 4.18 クラスタがインストールされている場合、 OCP 4.19 にアップグレードできます。最初に ACI CNI をアップグレードしてから（「Cisco ACI CNI プラグインのアップグレード」ガイドを参照）、Red Hat のドキュメントに従って OpenShift 4.18 を 4.19 にアップグレードします。

ネットワーク設計と Cisco ACI CNI プラグイン

このセクションでは、Cisco ACI Container Network Interface（CNI）プラグインを利用するネットワーク設計について説明します。

この設計では、OpenShift ノード トラフィックを異なる Neutron ネットワーク上のポッド トラフィックから分離します。分離により、次の図に示すように、ブートストラップ、制御、およびコンピューティング仮想マシン（VM）に 2 つのネットワーク インターフェイスが割り当てられます。



1 つのインターフェイスはノード ネットワーク用で、2 つ目はポッド ネットワーク用です。2 番目のインターフェイスも、Cisco ACI コントロール プレーン トラフィックを伝送します。VLAN タグ付きサブインターフェイスは、ポッド トラフィックと Cisco ACI コントロール プレーン トラフィックを伝送するように 2 番目のインターフェイスで設定されます。

このネットワーク設計では、Red Hat OpenShift インストーラ UPI Ansible モジュールにいくつかの変更を加える必要があります。これらの変更は、シスコが提供する OpenShift インストーラ UPI Ansible モジュールに実装されており、OpenShift インストーラ tar ファイル（openshift_installer-6.1.1.<z> .src.tar.gz）にパッケージ化されています。これは、他の Cisco ACI CNI 6.1.(1) リリース アーティファクトとともに利用できます。具体的には、次の点が変更されます。

- 別のプレイブックに 2 番目の Neutron ネットワークを作成します。
- コントロールを起動し、仮想マシン（VM）を計算する既存のプレイブックを次のように変更します。
  - 2 番目の Neutron ネットワークに 2 番目のポートを作成し、2 番目のインターフェイスとして VM 設定に追加します。
  - Neutron フローティング IP アドレスに追加の属性「nat_destination」を追加します。
- 最初の Neutron ネットワークを作成するプレイブックを次のように更新します。
  1. 定義済みの Cisco ACI 仮想ルーティングおよび転送（VRF）コンテキストにマッピングする Neutron アドレス スコープを作成します。
  2. 前の手順のアドレス範囲に、Neutron サブネットプールを作成します。
  3. サブネットの作成を変更して、前の手順のサブネットプールからサブネットを選択します。
  4. neutron ネットワークの最大伝送ユニット（MTU）を設定します（後述の設定ファイルから取得）。
- 2番目のネットワーク インターフェイス（およびそのインターフェイス上のサブインターフェイス）の作成に加えて、「openshift-install create initiator-configs」ステップで作成されたストック イグニッション ファイルを更新する必要があります。これは、提供されている追加のプレイブックによって実行されます。

（注） このセクションのカスタマイズの一部を実行するために必要な構成は、インベントリ ファイルの新しいパラメータを使用して行います。

OpenShift 4.19 をインストールするための前提条件

OpenStack 17.1 に OpenShift Container Platform（OCP）4.19 を正しくインストールするには、次の要件を満たす必要があります。

Cisco ACI

1. 独立した Cisco ACI VRF および「共通の」Cisco ACI テナントで Cisco ACI レイヤ 3 外部接続（L3Out）を設定して、エンドポイントが次のことを実行できるようにします。
   - パッケージとイメージを取得するため、外部にアクセスします。
   - Cisco Application Policy Infrastructure Controller（APIC）に到達します。
2. エンドポイントが以下を実行できるように、OpenShift クラスタ（acc-provision 入力ファイルで構成）で使用される独立 VRF で別の L3Out を構成します。
   - OpenShift クラスタ外部の API エンドポイントに到達します。
   - OpenStack API サーバーに到達します。

   OpenShift ポッド ネットワークはこの L3Out を使用します。
3. Cisco ACI インフラ VLAN を識別します。
4. OpenShift クラスタ サービス トラフィックに使用できる別の未使用の VLAN を指定します。

   サービスは、OpenShift クラスタの `acc_provision` 入力ファイルの `service_vlan` フィールドで設定されます。

OpenStack [英語]

1. Red Hat OpenStack Platform（OSP）17.1 を、Cisco ACI Neutron プラグイン（リリース 5.2(3)）を使用してネスト モードでインストールします。次のパラメータを、Cisco ACI `.yaml` モジュラ レイヤ 2（ML2）設定ファイルで設定します。
   - `ACIOpflexInterfaceType: ovs`
   - `ACIOpflexInterfaceMTU: 8000`

   既存のインストールを更新するには（上記の 2 つのパラメータが構成されていない場合）、Cisco.com の『*OpenStack Platform 17.1 Director* を使用した *Red Hat OpenStack* の *Cisco ACI Installation Guide*』を参照してください。
2. OpenStack プロジェクトと、OpenShift クラスタをホストするために必要なクォータを作成し、その他の必要な構成を実行します。

Red Hat OpenStack Web サイトにある OpenStack 4.19 の『独自のインフラストラクチャ上の *OpenStack* にクラスタをインストールする』の手順に従います。

3. 関連する Cisco ACI 拡張機能を使用し、OpenStack L3Out にマッピングして、以下を含む OpenStack Neutron 外部ネットワークを作成します。

   - セキュア ネットワーク アドレス変換（SNAT）用に設定されたサブネット。
   - フローティング IP アドレス用に設定されたサブネット。

   Cisco.com の、『*OpenStack Platform 17.1 Director* を使用した *Red Hat OpenStack* の *Cisco ACI Installation Guide*』の「OpenStack 外部ネットワークの追加」の章を参照してください。

   （注） すべての OpenStack プロジェクトは、OpenStack L3Out および Neutron 外部ネットワークを共有できます。

4. Cisco ACI ファブリックで管理されていないエンドポイントから OpenShift ノードネットワークへの直接アクセスが必要な場合（つまり、Neutron フローティング IP を使用しない場合）、この直接アクセスが予想されるすべての IP サブネットを特定します。これらの IP サブネットは、後でインストール プロセス中に Neutron サブネット プールを作成するために使用されます。

5. 『*OpenStack* ユーザー プロビジョニング インフラストラクチャへの *OpenShift* のインストール』の「Red Hat Enterprise Linux CoreOS (RHCOS)」セクションの手順に従って、RHCOS を取得し、OpenStack イメージを作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file
rhcos-4.19.0-x86_64-openstack.x86_64.qcow2 rhcos-4.19
```

OpenShift

インストール中にすべてのポッドからのトラフィックを送信元で NAT 処理するために、Cisco ACI コンテナ ネットワーク インターフェイス（CNI）によって使用される SNAT IP アドレスを特定します。`Inventory.yaml` ファイルの `aci_cni` セクションの `cluster_snat_policy_ip` 設定で SNAT IP アドレスを使用します。

インストーラ ホスト

ノード ネットワークと OpenStack Director API にアクセスしてインストール スクリプトを実行するには、Linux ホストにアクセスする必要があります。次のものがインストールされている必要があります。

- Ansible 6.7 以降をインストールします。

  Ansible の Web サイトの『*Ansible* のインストール』を参照してください。

- Python 3
- jq：JSON linting
- yq：YAML linting：**sudo pip install yq**
- python-openstackclient 5.4 以降：**sudo pip install python-openstackclient==6.5.0**
- openstacksdk 1.0 以降：**sudo pip install openstacksdk==3.0.0**

- python-swiftclient 3.9.0：**sudo pip install python-swiftclient==4.5.0**
- Ansible の Kubernetes モジュール：**sudo pip install openshift==0.13.2**

（注） Cisco では、Ansible バージョン 6.7.0 および Python 3.8.10 を使用して上記のバージョンを検証しています。ただし、後続のマイナー リリースも機能するものと予想されています。

このドキュメントでは、OpenShift クラスタに `openupi` という名前とディレクトリ構造（`~/openupi/openshift-env/upi`）を使用します。

```
$ cd ~/
$ mkdir -p openupi/openshift-env/upi
$ cd openupi/
$ tar xfz <path>/openshift_installer-6.1.1.<z>.src.tar.gz
$ cp openshift_installer/upi/openstack/* openshift-env/upi/
```

OpenShift 4.19 を OpenStack 17.1 にインストールする

事前に準備したインストーラ ホストからインストールを開始します。

始める前に

「前提条件」の項に記載されているタスクを完了します。

手順

ステップ 1 `oc` クライアントと `openshift-install` バイナリ ファイルをダウンロードして解凍します。

```
$ cd ~/openupi/openshift-env/
$ wget
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest-4.19/openshift-client-linux.tar.gz
$ tar xfz  openshift-client-linux.tar.gz
$ mv oc /usr/local/bin/
$ wget
https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest-4.19/openshift-install-linux.tar.gz
$ tar xfz openshift-install-linux.tar.gz
```

（注）
上記のテキストのリンクは、Cisco が検証した OpenShift 4.19.2 リリースを参照しています。ただし、後続のマイナー リリースも機能するものと予想されています。

ステップ 2 Cisco ACI Container Network Interface（CNI）6.1(1)リリース アーティファクトに存在する `acc-provision` パッケージをインストールします。

（注）
現在、RHEL8 でのみ満たされる Python 3 の依存関係が存在するため、acc-provision ツールは RHEL8 オペレーティング システムでのみ実行できます。

ステップ 3 acc-provision ツールを実行して、OpenShift クラスタ用の Cisco APIC を設定します。これにより、Cisco ACI CNI プラグインをインストールするためのマニフェストも生成されます。

例：

```
$ cd ~/openupi
$ acc-provision -a -c acc-provision-input.yaml -u <user> -p <password> -o aci_deployment.yaml -f
 openshift-4.19-openstack
```

この手順では、`aci_deployment.yaml` ファイルが生成されます。`aci_deployment.yaml.tar.gz` という名前の、Cisco ACI CNI マニフェストを含む tar ファイルも生成されます。`aci_deployment.yaml.tar.gz` ファイルの場所をメモします。後で `install-config.yaml` ファイルで指定する必要があるからです。

次に、acc-provision 入力ファイルの例を示します（ここで使用されている acc-provision フレーバーは `openshift-4.19-openstack` です）。

```
#
# Configuration for ACI Fabric
#
aci_config:
  system_id: <cluster-name>             # Every opflex cluster on the same fabric must have a
distinct ID
  tenant:
    name: <openstack-tenant-name>
  apic_hosts:                           # List of APIC hosts to connect to for APIC API access
    - <apic-ip>
  apic_login:
    username: <username>
    password: <password>
  vmm_domain:                           # Kubernetes VMM domain configuration
    encap_type: vxlan                   # Encap mode: vxlan or vlan
    mcast_range:                        # Every vxlan VMM on the same fabric must use a distinct
 range
        start: 225.125.1.1
        end: 225.125.255.255
  # The following resources must already exist on the APIC,
  # this is a reference to use them
  aep: sauto-fab3-aep         # The attachment profile for ports/VPCs connected to this cluster
  vrf:                        # VRF used to create all subnets used by this Kubernetes cluster
    name: l3out_2_vrf         # This should exist, the provisioning tool does not create it
    tenant: common            # This can be tenant for this cluster (system-id) or common
  l3out:                      # L3out to use for this kubernetes cluster (in the VRF above)
    name: l3out-2             # This is used to provision external service IPs/LB
    external_networks:
        - l3out_2_net         # This should also exist, the provisioning tool does not create it
#
# Networks used by Kubernetes
#
net_config:
  node_subnet: 10.11.0.1/27         # Subnet to use for nodes
  pod_subnet: 10.128.0.1/16         # Subnet to use for Kubernetes Pods
  extern_dynamic: 150.3.1.1/24      # Subnet to use for dynamically allocated ext svcs
  extern_static: 150.4.1.1/21       # Optional: Subnet for statically allocated external services

  node_svc_subnet: 10.5.168.1/21    # Subnet to use for service graph
  service_vlan: 1022                # The VLAN used for external LoadBalancer services
  infra_vlan: 4093
  interface_mtu: 1400
```

上記の acc-provision 入力ファイルで使用する *system_id* が Cisco ACI Object Naming and Numbering: Best Practices に準拠していることを確認します。これは、OpenStack プロジェクトの作成時に選択したテナント名にも当てはまります（上記の入力ファイルで指定します）。

ステップ 4 **install**、**create**、**wait-for** の OpenShift インストーラ コマンドは、`openshift-env` ディレクトリから実行します。

`clouds.yaml` ファイルが現在の作業ディレクトリまたは `~/.config/openstack/clouds.yaml` に存在し、環境変数 OS_CLOUD が正しいクラウド名に設定されていることを確認します。

OpenStack Web サイトの python-openstackclient3.12.3.dev2 の「構成」を参照してください。

ステップ 5 先ほど acc-provision ツールが生成した `aci_deployment.yaml.tar.gz` ファイルを展開します。

```
$ cd ~/openupi
$ tar xfz aci_deployment.yaml.tar.gz
```

ステップ 6 GitHub にある、リリース 4.19 に対応する「*OpenStack* ユーザー プロビジョニングインフラストラクチャへの *OpenShift* のインストール」の「構成のインストール」セクションの説明に従って、`install-config.yaml` を作成します。

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create install-config --dir=upi --log-level=debug
```

次に、Cisco ACI コンテナ ネットワーク インターフェイス（CNI）を `networkType` として設定する `install-config.yaml` ファイルの例を示します。

```
apiVersion: v1
baseDomain: noiro.local
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: openupi
networking:
  clusterNetwork:
  - cidr: 15.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 15.11.0.0/27
  networkType: CiscoACI
  serviceNetwork:
  - 172.30.0.0/16
platform:
  openstack:
    cloud: openstack
    computeFlavor: aci_rhel_huge
    externalDNS: ["<ip>"]
    externalNetwork: sauto_l3out-2
lbFloatingIP: 60.60.60.199
    octaviaSupport: "0"
    region: ""
    trunkSupport: "1"
    clusterOSImage: rhcos-4.19
publish: External
proxy:
```

```
  httpsProxy: <proxy-ip>
  httpProxy: <proxy-ip>
  noProxy: "localhost,127.0.0.1,<add-more-as-relevant>,172.30.0.1,172.30.0.10,oauth-
      openshift.apps.openupi.noiro.local,console-openshift-
      console.apps.openupi.noiro.local,downloads-openshift-
      console.apps.openupi.noiro.local,downloads-openshift-
      console.apps.openupi.noiro.local,alertmanager-main-openshift-
      monitoring.apps.openupi.noiro.local"
pullSecret:
sshKey:
```

ステップ 7 前の手順で生成されたファイルを環境に合わせて編集します。

例に記載されているように、編集には、GitHub にあるリリース 4.19 対応の「*OpenStack* ユーザー プロビジョニング インフラストラクチャへの *OpenShift* のインストール」の「ノード サブネットの修正」および「からのコンピューティング プール」セクションで説明されている `networkType` の変更を含める必要があります。

ステップ 8 次の例に示すように、`install-config.yaml` ファイルと `acc-provision-input.yaml` ファイルの関連フィールドと一致するように `inventory.yaml` ファイルを編集します。

```
all:
  hosts:
    localhost:
      aci_cni:
        acc_provision_tar: <path>/aci_deployment.yaml.tar.gz
        kubeconfig: <path>/kubeconfig
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '15.11.0.0/27'
      os_flavor_master: 'aci_rhel_huge'
      os_flavor_worker: 'aci_rhel_huge'
      os_image_rhcos: 'rhcos-4.19.'
      os_external_network: 'l3out-2'
      # OpenShift API floating IP address
      os_api_fip: '60.60.60.6'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '60.60.60.8'
      # Subnet pool prefixes
      cluster_network_cidrs: '15.128.0.0/14'

      # Name of the SDN.
      os_networking_type: 'CiscoACI'

      # Number of provisioned Control Plane nodes
      # 3 is the minimum number for a fully-functional cluster.
      os_cp_nodes_number: 3
      # Number of provisioned Compute nodes.
      # 3 is the minimum number for a fully-functional cluster.
      os_compute_nodes_number:0
      os_apiVIP: '{{ os_subnet_range | next_nth_usable(5) }}'
      os_ingressVIP: '{{ os_subnet_range | next_nth_usable(7)
      }}'
```

（注）

- `inventory.yaml` ファイルは、この手順の後半で `update_ign.py` スクリプトを実行した後に更新されます。同じクラスタを再度インストールするために再利用できるように、この段階で `inventory.yaml` ファイルのコピーを作成することをお勧めします。

- Cisco ACI CNI 固有の設定が `inventory.yaml` ファイルの `aci_cni` セクションに追加されます。この手順の例では必須のフィールドを取り上げていますが、さらに多くのオプション設定も使用することができます。オプションのリストについては、このガイドの「オプション設定」のセクションを参照してください。

手順 12 の説明に従って `update_ign.py` を実行すると、一部のデフォルト値と派生値がインベントリ ファイルに追加されることに注意してください。たとえば、入力されているすべてのオプション値と派生値を含む設定を確認するには、GitHub の `openshift_installer/upi/openstack/inventory.yaml` を参照してください。

ステップ 9 OpenShift マニフェストを生成し、Cisco ACI CNI マニフェストにコピーします。

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create manifests  --log-level debug --dir=upi
# Copy the ACI CNI manifests obtained earlier in Step 5
$ cp ../cluster-network-* upi/manifests/
# Update control-plane machines and machine-set manifests to manage control-plane machines through
 ControlPlaneMachineSet resource.
$ cd upi
$ python update_master_manifests.py
```

ステップ 10 Cisco ACI ネットワークタイプの OpenStack Octavia ロードバランサの作成を無効にします。

```
$ cd ~/openupi/openshift-env/upi
$ ansible-playbook -i inventory.yaml disable-octavia.yaml
```

ステップ 11 コントロール プレーンノードをスケジュール不可にします。

GitHub のリリース 4.19対応「*OpenStack* ユーザー プロビジョニング インフラストラクチャへの *OpenShift* のインストール」の「コントロールプレーン ノードをスケジュール不可にする」セクションの手順に従います。

ステップ 12 イグニッション ファイルを更新します。

```
$ cd ~/openupi/openshift-env
$ ./openshift-install create ignition-configs --log-level debug --dir=upi
$ cd upi
$ export INFRA_ID=$(jq -r .infraID metadata.json)
$ echo "{\"os_net_id\": \"$INFRA_ID\"}" | tee netid.json
$ source ~/openupi/overcloudrc
# Run the update_ign.py from the Cisco OpenShift installer package
$ python update_ign.py # This assumes that the inventory file is already configured

$ swift upload bootstrap bootstrap.ign
(To be executed in undercloud after copying the ignition file or host having connectivity to
openstack controller with overcloudrc)

$ swift post bootstrap --read-acl ".r:*,.rlistings"

(To be executed in undercloud after copying the ignition file host having connectivity to openstack
 controller with overcloudrc)
```

このステップのコマンドでは、イグニッション ファイルを作成し、Cisco ACI CNI に従って更新し、`bootstrap.ign` ファイルを swift ストレージにアップロードします。また、GitHub のリリース 4.19 対応「*OpenStack* ユーザー プロビジョニング インフラストラクチャへの *OpenShift* のインストール」の「Bootstrap Ignition Shim」セクションの説明に従って、`bootstrap-ignition-shim` を生成します。

ステップ 13 Cisco OpenShift インストーラ パッケージから取得した Ansible プレイブックを実行して、次のタスクを実行します。

a) セキュリティ グループとネットワークの作成。

```
ansible-playbook -i inventory.yaml security-groups.yaml
ansible-playbook -i inventory.yaml network.yaml
ansible-playbook -i inventory.yaml update-network-resources.yaml
ansible-playbook -i inventory.yaml 021_network.yaml
```

b) Cisco ACI ファブリックによって管理されていないエンドポイントから OpenShift ノードネットワークに直接アクセスするには、次の例に示すように、この直接アクセスが予想されるすべての IP サブネットに対して Neutron サブネットプールを作成します。

```
$ neutron subnetpool-create --pool-prefix <direct_access_src_subnet> --address-scope
node_network_address_scope <subnetpool_name>
```

前の例で、`node_network_address_scope` は、`network.yaml` ファイルによって作成された Neutron アドレス範囲の名前です。

c) コントロール プレーンをインストールします。

```
ansible-playbook -i inventory.yaml bootstrap.yaml
ansible-playbook -i inventory.yaml control-plane.yaml
```

d) ブートストラップ/コントロール プレーンのインストールが完了していることを確認します。

```
./openshift-install wait-for bootstrap-complete --dir=upi --log-level=debug
```

e) コントロール プレーンがインストールされたら、ブートストラップ ノードを削除します。

```
ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

f) （オプション）コントロールプレーンが起動したら、クラスタの送信元 IP ネットワーク アドレス変換（SNAT）ポリシーを設定します。

```
ansible-playbook -i inventory.yaml cluster_snat_policy.yaml
```

g) 以下で説明するように、ワーカー マシンセットをスケーリングしてコンピューティング ノードを起動します。

```
$ oc get machineset -A
NAMESPACE NAME DESIRED CURRENT READY AVAILABLE AGE
openshift-machine-api openupi-vkkn6-worker 0 0  5h10m
$ oc scale machineset -n openshift-machine-api openupi-vkkn6-worker --replicas=1
```

（注）

`control-plane.yaml` プレイブックは、実行中、マシンセット設定を自動的に更新して、複数のネットワーク インターフェイスをサポートできるようにします。これにより、インベントリ ファイルに非 0 の**os_compute_nodes_number** が記述されていれば、レプリカのスケーリングが可能になります。

ステップ 14 Ansible プレイブックを使用してコンピューティング ノードを作成した場合は、保留中の証明書署名要求を承認してください。

```
oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm
certificate approve
```

ステップ 15 LoadBalancerService を使用するように、デフォルトの IngressController の公開戦略を更新します。

```
ansible-playbook -i inventory.yaml post-install.yaml
```

ステップ 16 インストールのステータスを確認します。

```
./openshift-install wait-for install-complete --dir=upi --log-level=debug
```

ステップ 17 クラスタを破棄します。

```
ansible-playbook -i inventory.yaml down-compute-nodes.yaml
ansible-playbook -i inventory.yaml down-control-plane.yaml
ansible-playbook -i inventory.yaml down-network.yaml
ansible-playbook -i inventory.yaml down-security-groups.yaml
```

この手順でプレイブックを実行すると、ノードネットワークに対応する Cisco ACI BridgeDomain も削除されます。クラスタを再インストールするには、このドキュメントで前述したように、`-a` を指定して `acc-provision` を再度実行します。

オプション設定

ここでは、いくつかのオプション構成の方法について説明します。

ACI CNI を使用して OpenShift 4.x クラスタで Multus CNI プラグインを有効化する

新しいクラスタ、またはすでにインストールされているクラスタで、Multusを有効にできます。

新しいクラスタ インストールでの Multus の有効化

acc-provision を実行する場合は、*disable-multus* 引数を *False* に設定します。

```
$ acc-provision -a -c acc_provision_input.yaml -f openshift-4.19-openstack -u <username> -p <password> -o
aci_deployment.yaml --disable-multus false
```

次の手順は、すでにインストールされているクラスタで Multus を有効にするためのものです。

手順

ステップ 1 新しい ACI CNI 展開構成を生成します。

```
$ acc-provision -c acc_provision_input.yaml -f openshift-4.19-openstack -u <username> -p <password>
 -o aci_deployment.yaml --disable-multus false
```

（注）
上記のコマンドでは、`-a` フラグを使用しないでください。

ステップ 2 acicontainersoperator CR を削除します。

```
$ oc delete acicontainersoperator acicnioperator -n aci-containers-system
```

ステップ 3 新しい `aci_deployment.yaml` ファイルを適用します。

```
$ oc apply -f aci_deployment.yaml
```

ステップ 4 `cluster-network-03-config.yaml` を編集して、現在の OpenShift ネットワークオブジェクトから「disableMultiNetwork: true」を削除します。

```
$ oc edit -f cluster-network-03-config.yaml
```

CPMS によるコントロール プレーン ノードの管理を有効にする

Control Plane Machine Set（CPMS）リソースを活用することで、コントロール プレーン ノードを自動管理できます。

（注） これらの手順は、OpenShift クラスタをバージョン 4.18 から 4.19 にアップグレードする際、かつ 4.18 で CPMS 構成されていない場合に役立ちます。

CPMS を有効にするには、以下の手順に従ってください。

手順

ステップ 1 既存のコントロール プレーン ノードに対してマシン オブジェクトを作成します。

これは、コントロール プレーン ノードのマシン構成例です。

```
apiVersion: machine.openshift.io/v1beta1 kind: Machine metadata: labels:
machine.openshift.io/cluster-api-cluster: openupi-tvqjc machine.openshift.io/cluster-api-machine-role:
 master machine.openshift.io/cluster-api-machine-type: master name: openupi-tvqjc-master-0 namespace:
 openshift-machine-api annotations: machine.openshift.io/instance-id:
6c578563-f295-487a-a1c1-e81a942bfd28 spec: lifecycleHooks: {} metadata: {} providerSpec: value:
apiVersion: openstackproviderconfig.openshift.io/v1alpha1 cloudName: openstack cloudsSecret: name:
 openstack-cloud-credentials namespace: openshift-machine-api flavor: aci_rhel_medium image: rhcos-4.19
 kind: OpenstackProviderSpec metadata: creationTimestamp: null networks: - filter: {} subnets: -
filter: name: openupi-tvqjc-nodes uuid: - ec2413e7-2b02-46ef-9a2f-bf308eeb5a0c uuid:
c5f49c1d-3e78-47ff-af4e-3c89d47e4ae5 - filter: {} subnets: - filter: name:
openupi-tvqjc-acicontainers-nodes uuid: - d577cf9d-0920-4b60-bf2c-ae76b24e42e6 uuid:
a29eb31a-c590-47f1-a592-4a35e265c2b3 securityGroups: - filter: {} name: openupi-tvqjc-master
serverGroupName: openupi-tvqjc-master serverMetadata: Name: openupi-tvqjc-master openshiftClusterID:
 openupi-tvqjc tags: - openshiftClusterID=openupi-tvqjc trunk: true userDataSecret: name:
master-user-data providerID: openstack:///6c578563-f295-487a-a1c1-e81a942bfd28
```

マニフェストを使用して、既存のマスター ノードごとにマシン オブジェクトを作成します。テンプレートから以下のフィールドを更新してください。

- `openupi-tvqjc` をクラスタの OpenShift クラスタ ID に置換します。
- `metadata.name`： このフィールドにマスター ノードの名前を入力してください。
- `metadata.annotations.machine.openshift.io/instance-id`：このマスター ノードに対応する OpenStack サーバ ID をこのフィールドに設定してください。
- `spec.template.spec.providerSpec.value.flavor`：マスター ノードの作成に使用する OpenStack のフレーバー名をこのフィールドに設定してください。
- `spec.template.spec.providerSpec.value.image`：このフィールドには、マスター ノードの作成に使用される RHCOS イメージに対応する OpenStack イメージ名を設定してください。

- `spec.template.spec.providerSpec.value.networks`：この構成セクションを、マスター ノードの作成に使用する OpenStack のネットワークおよびサブネットの詳細に合わせて更新してください。
- `spec.providerID`：このフィールドの UUID を、このマスター ノードに対応する OpenStack サーバ ID に更新してください。

各マスター ノードにこれらのマニフェストを適用した後、次のコマンドを実行して、マシンが作成され、「実行中」と表示されていることを確認してください：

```
$ oc get machines -n openshift-machine-api
```

ステップ 2 コントロール プレーン マシン セット（CPMS）リソースを作成します。

以下は CPMS リソースの例です：

```
apiVersion: machine.openshift.io/v1 kind: ControlPlaneMachineSet metadata: name: cluster namespace:
 openshift-machine-api spec: replicas: 3 selector: matchLabels:
machine.openshift.io/cluster-api-cluster: openupi-tvqjc machine.openshift.io/cluster-api-machine-role:
 master machine.openshift.io/cluster-api-machine-type: master state: Active strategy: type:
RollingUpdate template: machineType: machines_v1beta1_machine_openshift_io
machines_v1beta1_machine_openshift_io: metadata: labels: machine.openshift.io/cluster-api-cluster:
 openupi-tvqjc machine.openshift.io/cluster-api-machine-role: master
machine.openshift.io/cluster-api-machine-type: master spec: lifecycleHooks: {} metadata: {}
providerSpec: value: apiVersion: openstackproviderconfig.openshift.io/v1alpha1 cloudName: openstack
 cloudsSecret: name: openstack-cloud-credentials namespace: openshift-machine-api flavor:
aci_rhel_medium image: rhcos-4.19 kind: OpenstackProviderSpec metadata: creationTimestamp: null
networks: - filter: {} subnets: - filter: name: openupi-tvqjc-nodes - filter: {} subnets: - filter:
 name: openupi-tvqjc-acicontainers-nodes securityGroups: - filter: {} name: openupi-tvqjc-master
serverGroupName: openupi-tvqjc-master serverMetadata: Name: openupi-tvqjc-master openshiftClusterID:
 openupi-tvqjc tags: - openshiftClusterID=openupi-tvqjc trunk: true userDataSecret: name:
master-user-data
```

構成を適用する前に、以下のフィールドを更新してください：

- `openupi-tvqjc` をクラスタの OpenShift クラスタ ID に置換します。
- `spec.template.spec.providerSpec.value.flavor`：マスター ノードの作成に使用する OpenStack のフレーバー名をこのフィールドに設定してください。
- `spec.template.spec.providerSpec.value.image`：このフィールドには、マスター ノードの作成に使用される RHCOS イメージに対応する OpenStack イメージ名を設定してください。
- `spec.template.spec.providerSpec.value.networks`：この構成セクションを、マスター ノードの作成に使用する OpenStack のネットワークおよびサブネットの詳細に合わせて更新してください。

  （注）
  ControlPlaneMachineSet を稼働させるには、`spec.state` を `active` に設定する必要があります。

  `network/subnet` UUID は、ControlPlaneMachineSet でがサポートされていません。

ステップ 3 コントロール プレーン マシン セット（CPMS） リソースを作成すると、既存のマスター ノードは新しいものに置き換えられます。以下のコマンドを使用して、その進捗状況を確認できます：

```
$ oc get controlplanemachineset -n openshift-machine-api # To check CPMS status

$ oc get machines -n openshift-machine-api # To check machines status

$ oc get nodes -l node-role.kubernetes.io/master # To check nodes status
```

ワーカーノードで IP 転送を有効にする

ノードポート構成を使用してサービスをテストするには、ワーカーノードで IP 転送を有効にする必要があります。

この手順を使用して、ワーカーノードで IP 転送を有効にします。

手順

ステップ 1 ワーカー VM にフローティング IP を割り当てます。

ステップ 2 ワーカーノードに関連付けられたセキュリティグループで SSH アクセスを許可するセキュリティルールを追加して、SSH トラフィックを許可します。

ステップ 3 各ワーカーノードに SSH で接続し、次のコマンドを実行して、各ワーカーノードで IP 転送を有効にします。

```
sudo sysctl -w net.ipv4.ip_forward=1
```

オプションのインベントリ構成

「*OpenShift 4.19* を *OpenStack* にインストールする」のセクションのステップ 8 で、inventory.yaml ファイルの aci_cni セクションにある、Cisco ACI コンテナ ネットワーク インターフェイス（CNI）構成に必要なフィールドを書き留めました。ここでは、オプションの構成とデフォルト値について説明します。

オプション	説明とデフォルト値
`cluster_snat_policy_ip`	デフォルトでは、この値は設定されていません。 送信元 IP ネットワーク アドレス変換（SNAT）の IP アドレスは、クラスタ全体に適用される Cisco ACI-CNI SNAT ポリシーを作成するために使用されます。この SNAT ポリシーは、このガイドの「*OpenShift 4.19* を *OpenStack* にインストールする」のセクションで説明されているように、`cluster_snat_policy.yaml` Ansible プレイブックを実行して作成します。（この値が設定されていない場合は、このプレイブックを実行しないでください）。

オプション			説明とデフォルト値
dns_ip			デフォルトでは、この値は設定されていません。 『*Installing OpenShift on OpenStack User-Provisioned Infrastructure on GitHub*』の「Subnet DNS (optional)」セクションで説明されている手順に従わない場合は、このフィールドを設定します。この手順では、Nova サーバーが使用するデフォルトのリゾルバーを制御します。 値を使用して、*-primaryClusterNetwork ネットワークに関連付けられたサブネットの dns_nameservers フィールドを設定します。1 つ以上の DNS サーバー IP を指定できます。
network_interfaces	ノード	name	RHCOS イメージによって設定されたノードネットワークインターフェイスの名前。 デフォルト値は「enp3s0」です。
		mtu	*-primaryClusterNetwork Neutron ネットワークに設定された MTU。 デフォルト値は 1500 です
	opflex	name	RHCOS イメージによって設定されたノードネットワークインターフェイスの名前。 デフォルト値は「enp4s0」です。
		mtu	*-secondaryClusterAciNetwork Neutron ネットワークに設定された MTU。 デフォルト値は 1500 です
		subnet	デフォルト値は 192.168.208.0/20 です。 これは、*-secondaryClusterAciNetwork Neutron ネットワークに関連付けられているサブネットで使用される CIDR です。このサブネットのサイズは、少なくとも *-primaryClusterNetwork Neutron ネットワークで使用されるサブネットのサイズと同じである必要があります。また、OpenShift プロジェクトのアドレス範囲内の他の CIDR と重複しないようにする必要があります。

⚠

注意 OSP 17.1 の新規インストールでは、ノードのネットワーク インターフェイス名が以前のバージョンの OSP で観察されたものと異なる場合があります。インターフェイスが ens* ではなく enp*s と表示される場合があります。この命名のバリエーションは、コンピューティング ノードでの Nova の構成の hw_machine_type パラメータによって制御されます。インストールの問題を防ぐために、このセクションで説明されているように、正しいインターフェイス名が一覧ファイルで更新されていることを確認してください。

【注意】シスコ製品をご使用になる前に、安全上の注意（**www.cisco.com/jp/go/safety_warning/**）をご確認ください。本書は、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

この資料の記載内容は2008年10月現在のものです。
この資料に記載された仕様は予告なく変更する場合があります。

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255（フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。