Advanced Configuration Options （詳細設定オプション）

NetApp Solutions

NetApp
September 26, 2024

目次

Advanced Configuration Options （詳細設定オプション）

ロードバランサオプションの確認

ロードバランサのオプションの確認：ネットアップを使用した **Red Hat OpenShift**

ほとんどの場合、 Red Hat OpenShift は、ルートを介してアプリケーションを外部で利用できるようにします。サービスは、外部からアクセス可能なホスト名を付与することで公開されます。定義されたルートおよびサービスによって識別されるエンドポイントは、 OpenShift ルータによって使用され、外部クライアントにこの名前付き接続を提供できます。

ただし、アプリケーションでは、適切なサービスを公開するために、カスタマイズしたロードバランサの導入と設定が必要になる場合があります。その一例が、ネットアップアストラコントロールセンターです。このニーズを満たすために、いくつかのカスタムロードバランサオプションを評価しました。このセクションでは、これらのインストールと設定について説明します。

以下のページでは、解決策追加情報を搭載した Red Hat OpenShift で検証済みのロードバランサオプションについて説明します。

- "MetalLB"
- "F5 BIG-IP"

MetalLB ロードバランサのインストール：ネットアップでの **Red Hat OpenShift**

このページでは、 MetalLB ロードバランサのインストールおよび設定手順を示します。

MetalLB は、 OpenShift クラスタにインストールされた自己ホスト型ネットワークロードバランサであり、クラウドプロバイダで実行されないクラスタでタイプロードバランサの OpenShift サービスを作成できます。LoadBalancer サービスをサポートするために連携する MetalLB の 2 つの主な機能は、アドレス割り当てと外部アナウンスメントです。

MetalLB 設定オプション

MetalLB が OpenShift クラスタの外部でロードバランササービスに割り当てられた IP アドレスをどのようにアナウンスするかに基づいて、次の 2 つのモードで動作します。

- * レイヤ 2 モード。 * このモードでは、 OpenShift クラスタ内の 1 つのノードがサービスの所有権を取得し、その IP の ARP 要求に応答して、 OpenShift クラスタ外で到達可能にします。IP をアドバタイズするのはノードだけなので、帯域幅のボトルネックと低速フェールオーバーの制限があります。詳細については、のドキュメントを参照してください "こちらをご覧ください"。
- * このモードでは、 OpenShift クラスタ内のすべてのノードがルータとの BGP ピアリングセッションを確立し、トラフィックをサービス IP に転送するためにルートをアドバタイズします。このための前提条件は、 MetalLB をそのネットワーク内のルータと統合することです。BGP のハッシュメカニズムにより、サービスの IP-to-Node マッピングが変更されることがあります。詳細については、のドキュメントを参照してください "こちらをご覧ください"。

このマニュアルでは、レイヤ 2 モードで MetalLB を設定します。

MetalLB ロードバランサをインストールします

1. MetalLB リソースをダウンロードします。

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. ファイル「 metallb.yaml 」を編集し、「 pec.template.spec.securityContext` 」をコントローラ展開とスピーカー DemonSet から削除します。

   ◦ 削除する行数： *

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. 「 metallb-system' 」ネームスペースを作成します。

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. MetalLB CR を作成します。

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. MetalLB スピーカを設定する前に、スピーカ DemonSet の昇格特権を与えて、ロードバランサを動作させるために必要なネットワーク設定を実行できるようにします。

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. 「 metalLB - システム」ネームスペースに「 ConfigMap 」を作成して、 MetalLB を設定します。

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. これで、ロードバランササービスが作成されると、 MetalLB は外部 IP をサービスに割り当て、 ARP 要求に応答して IP アドレスをアドバタイズします。

BGP モードで MetalLB を設定する場合は、上記の手順 6 を省略し、 MetalLB マニュアルの手順に従います "こちらをご覧ください"。

F5 BIG-IP ロードバランサのインストール

F5 BIG-IP は、 L4-L7 ロードバランシング、 SSL/TLS オフロード、 DNS 、ファイアウォールなど、高度な運用レベルのトラフィック管理およびセキュリティサービスを幅広く提供する Application Delivery Controller （ ADC; アプリケーションデリバリコントローラ）です。これらのサービスにより、アプリケーションの可用性、セキュリティ、パフォーマンスが大幅に向上します。

F5 BIG-IP は、専用ハードウェア、クラウド、またはオンプレミスの仮想アプライアンスに、さまざまな方法で導入、使用できます。要件に応じて F5 BIG-IP を調査し、導入するには、ここで説明しているドキュメントを参照してください。

F5 BIG-IP サービスを Red Hat OpenShift と効率的に統合するために、 F5 は BIG-IP Container Ingress Service （ CIS ）を提供します。CI は、特定のカスタムリソース定義（ CRD ）の OpenShift API を監視し、 F5 BIG-IP システム構成を管理するコントローラポッドとしてインストールされます。F5 BIG-IP CIS は、 OpenShift でサービスタイプ Loadancers とルートを制御するように構成できます。

さらに、タイプ LoadBalancer にサービスを提供するための自動 IP アドレス割り当てには、 F5 IPAM コントローラを使用できます。F5 IPAM コントローラは、 LoadBalancer サービスの OpenShift API を ipamLabel 注釈で監視し、事前構成済みプールから IP アドレスを割り当てるコントローラポッドとしてインストールされます。

このページには、 F5 BIG-IP CIS および IPAM コントローラのインストールおよび設定手順がリストされてい

ます。前提条件として、 F5 BIG-IP システムを導入し、ライセンスを取得しておく必要があります。また、デフォルトでは BIG-IP VE 基本ライセンスに含まれている SDN サービスのライセンスも必要です。

F5 BIG-IP は、スタンドアロンモードまたはクラスタモードで導入できます。この検証の目的上、 F5 BIG-IP はスタンドアロンモードで導入されましたが、本番環境では、単一点障害を避けるために、大量の IP で構成されたクラスタを使用することを推奨します。

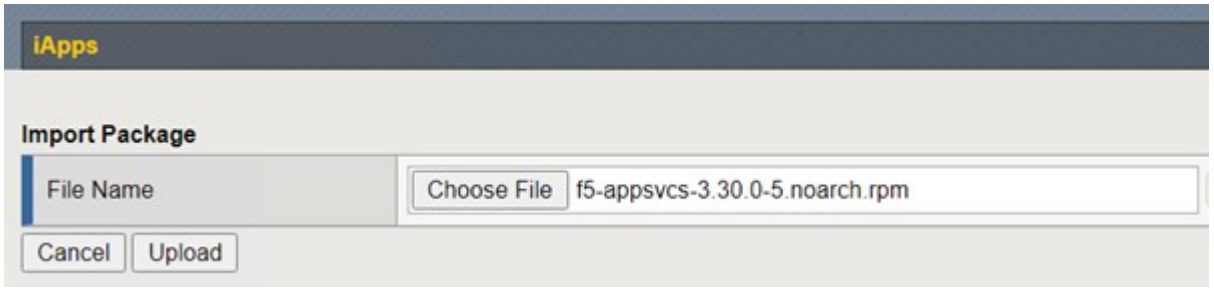
F5 BIG-IP システムは、専用のハードウェア、クラウド、またはオンプレミスの仮想アプライアンスとして、バージョンが 12.x よりも大きいオンプレミスに導入でき、 F5 CIS と統合できます。このドキュメントでは、 BIG-IP VE エディションなどを使用して、 F5 BIG-IP システムを仮想アプライアンスとして検証しました。

検証済みのリリース

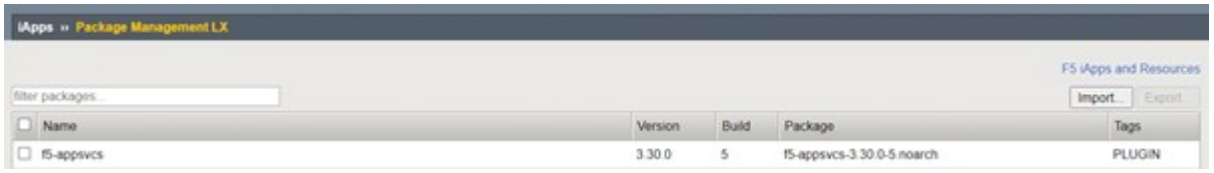
テクノロジ	ソフトウェアのバージョン
Red Hat OpenShift のサービスです	4.6 EUS 、 4.7
F5 BIG-IP VE エディション	16.1.0
F5 Container Ingress Service の略	2.5.1
F5 IPAM コントローラ	0.1.4
F5 AS3	3.30.0

インストール

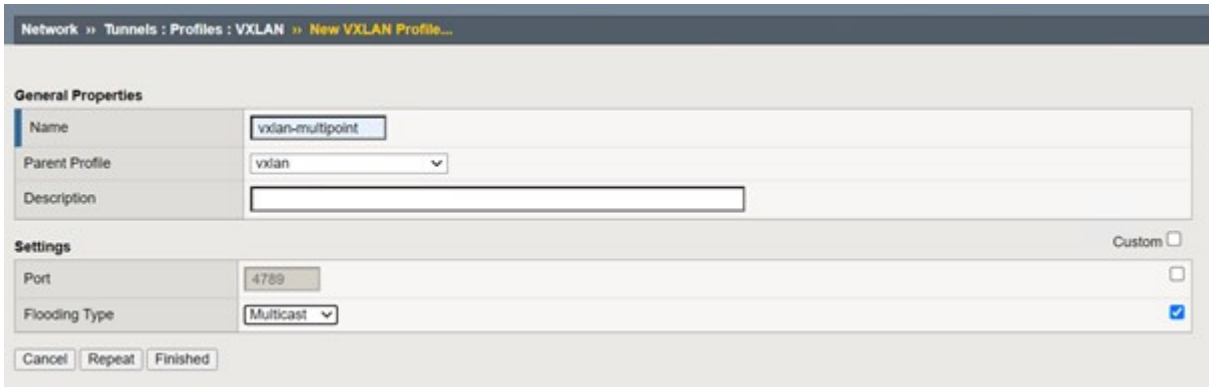
1. F5 Application Services 3 拡張機能をインストールして、 big-IP システムが命令コマンドではなく JSON で構成を受け入れるようにします。に進みます "F5 AS3 GitHub リポジトリ"をクリックし、最新の RPM ファイルをダウンロードします。
2. F5 BIG-IP システムにログインし、 iApps > Package Management LX に移動して、 Import （インポート）をクリックします。
3. [ファイルの選択] をクリックして、ダウンロードした AS3 RPM ファイルを選択し、 [OK] をクリックして、 [アップロード] をクリックします。



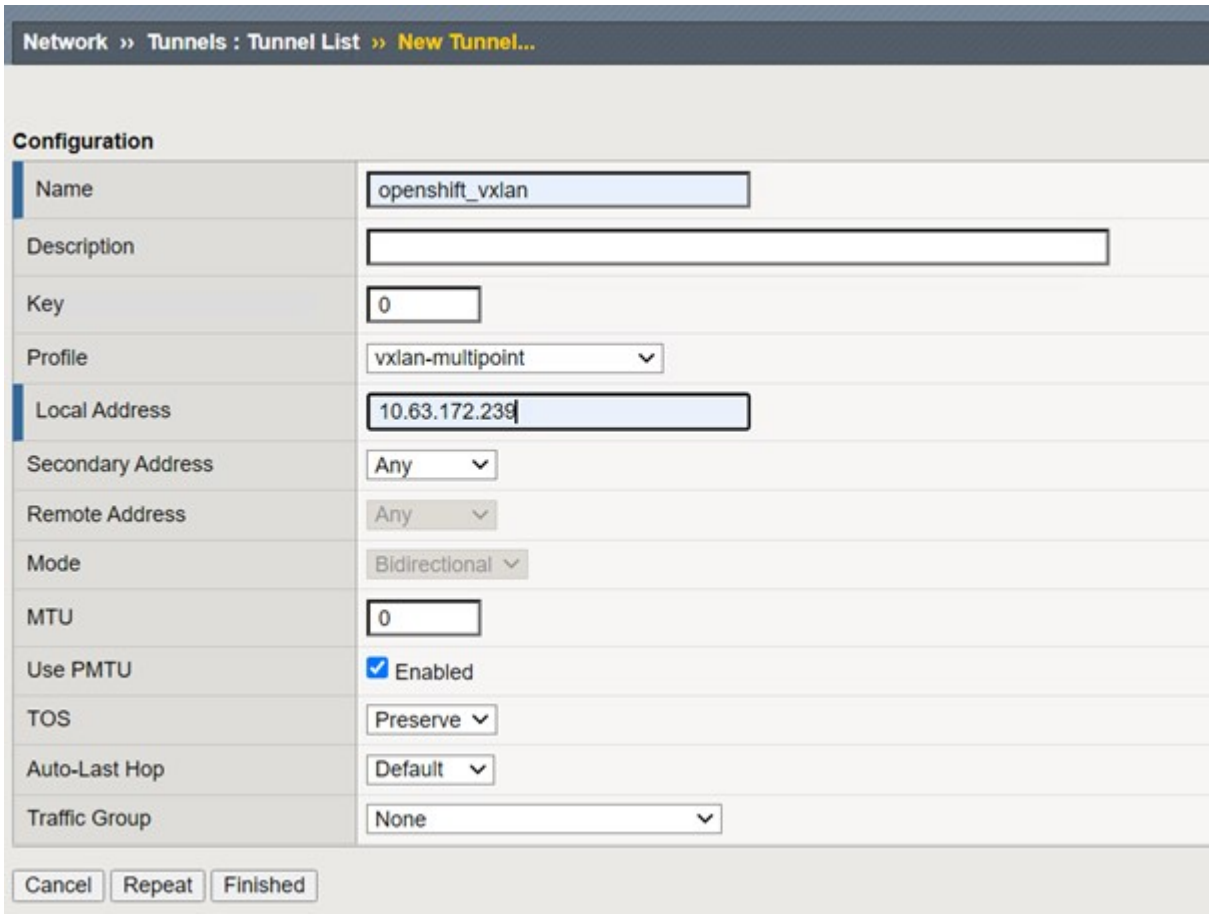
4. AS3 拡張機能が正常にインストールされたことを確認します。



5. 次に、 OpenShift システムと BIG-IP システム間の通信に必要なリソースを構成します。まず、 OpenShift SDN のための BIG-IP システムに VXLAN トンネルインターフェイスを作成し、 OpenShift と BIG-IP サーバ間にトンネルを作成します。Network > Tunnels > Profiles と進み、 Create をクリックして Parent Profile を VXLAN に設定し、フラッディング Type を Multicast に設定します。プロファイルの名前を入力し、 [完了] をクリックします。



6. Network > Tunnels > Tunnel List と進み、 Create をクリックして、トンネルの名前とローカル IP アドレスを入力します。前の手順で作成したトンネルプロファイルを選択し、 [完了] をクリックします。



7. クラスタ管理者権限で Red Hat OpenShift クラスタにログインします。
8. F5 BIG-IP サーバの OpenShift にホストサブネットを作成します。このサブネットは、 OpenShift クラスタから F5 BIG-IP サーバに拡張します。ホストサブネット YAML 定義をダウンロードします。

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. ホストサブネットファイルを編集し、 OpenShift SDN の BIG-IP VTEP （ VXLAN トンネル） IP を追加します。

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

ご使用の環境に応じて、 hostIP などの詳細情報を変更します。

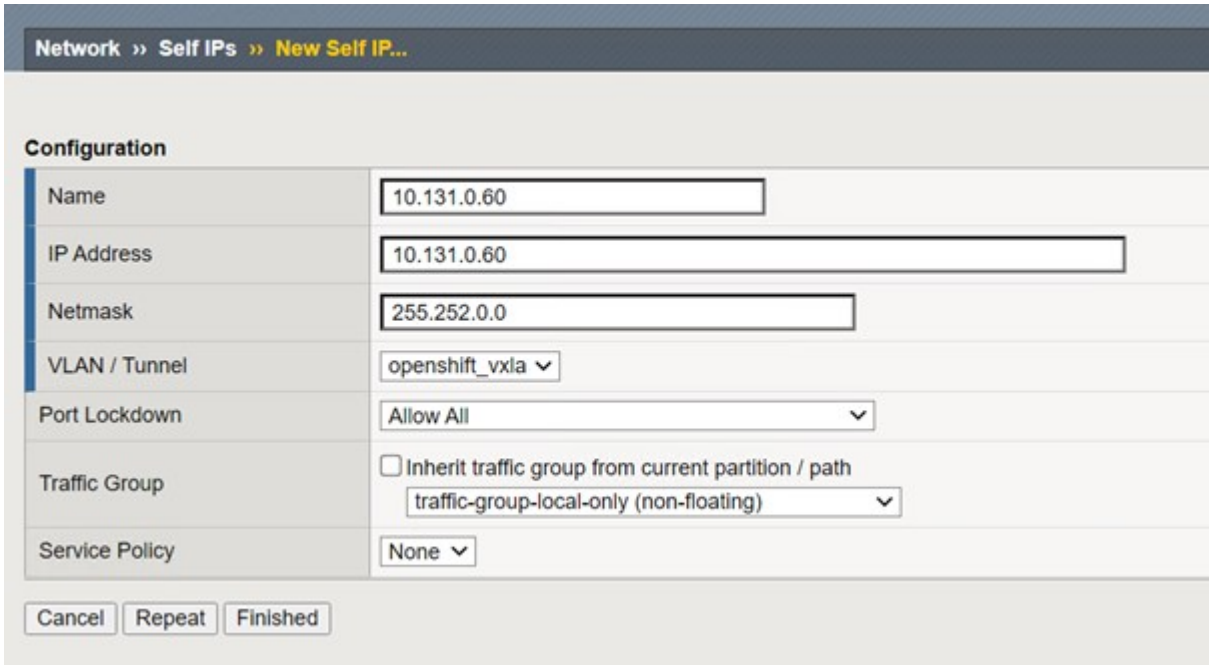
10. HostSubnet リソースを作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

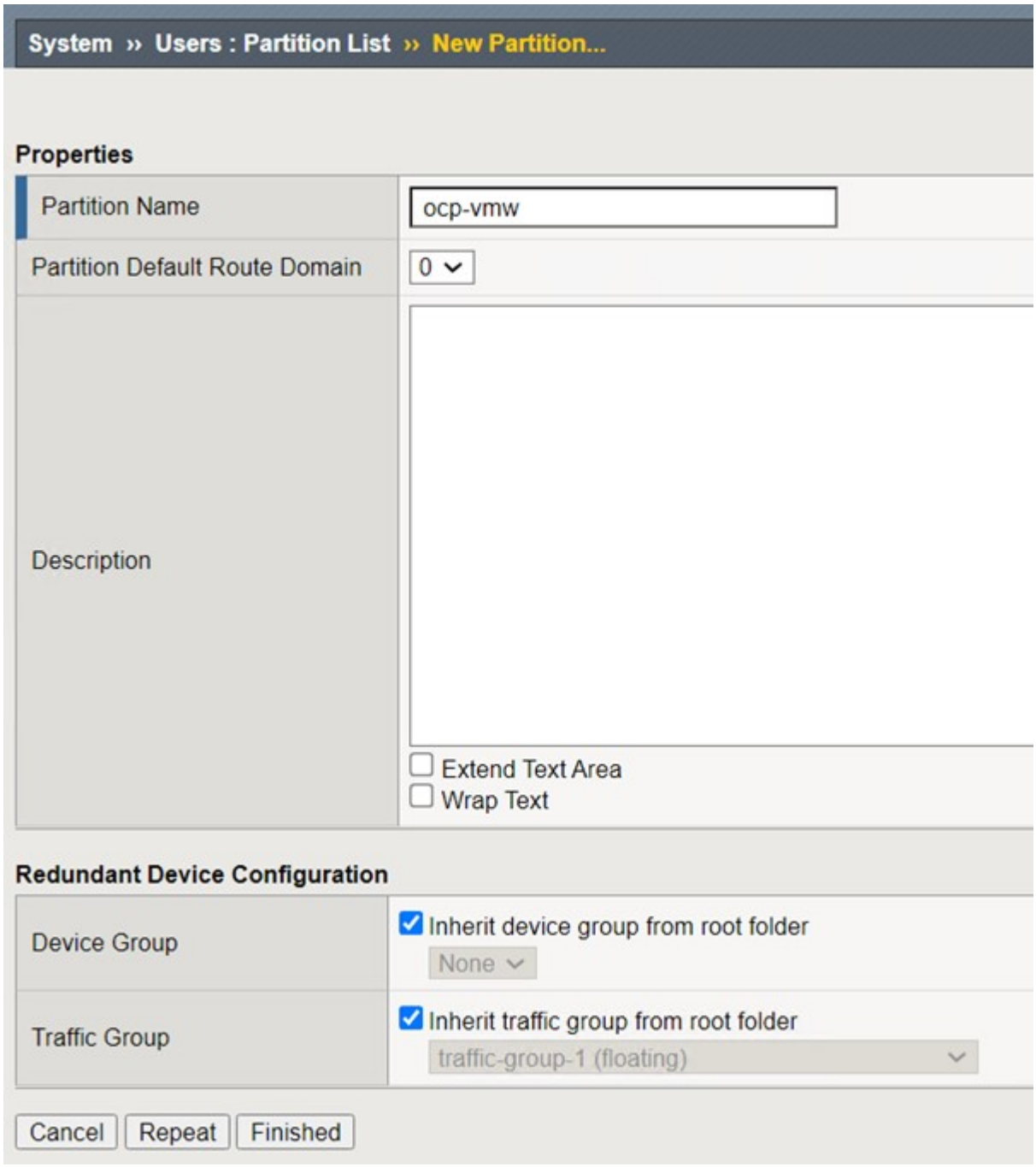
11. F5 BIG-IP サーバ用に作成されたホストサブネットのクラスタ IP サブネット範囲を取得します。

```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                           HOST                           HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                      f5-server                      10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0         ocp-vmw-nszws-master-0         10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1         ocp-vmw-nszws-master-1         10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2         ocp-vmw-nszws-master-2         10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4     ocp-vmw-nszws-worker-r8fh4     10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46     ocp-vmw-nszws-worker-tvr46     10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg     ocp-vmw-nszws-worker-wdxhg     10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4     ocp-vmw-nszws-worker-wg8r4     10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw     ocp-vmw-nszws-worker-wtgfw     10.63.172.17
10.128.4.0/23
```

12. F5 BIG-IP サーバに対応する OpenShift のホストサブネット範囲の IP を使用して、 VXLAN OpenShift 上に自己 IP を作成します。F5 BIG-IP システムにログインし、 [ネットワーク]>[自己 IP] の順に選択し、 [作成] をクリックします。F5 BIG-IP ホストサブネット用に作成されたクラスタ IP サブネットから IP を入力し、 VXLAN トンネルを選択して、その他の詳細を入力します。[完了] をクリックします。



13. CIS で設定および使用する F5 BIG-IP システムにパーティションを作成します。[システム]>[ユーザ]>[パーティションリスト] の順に選択し、 [作成] をクリックして詳細を入力します。[完了] をクリックします。



CIS で管理されるパーティションでは手動で設定しないことをお勧めします。

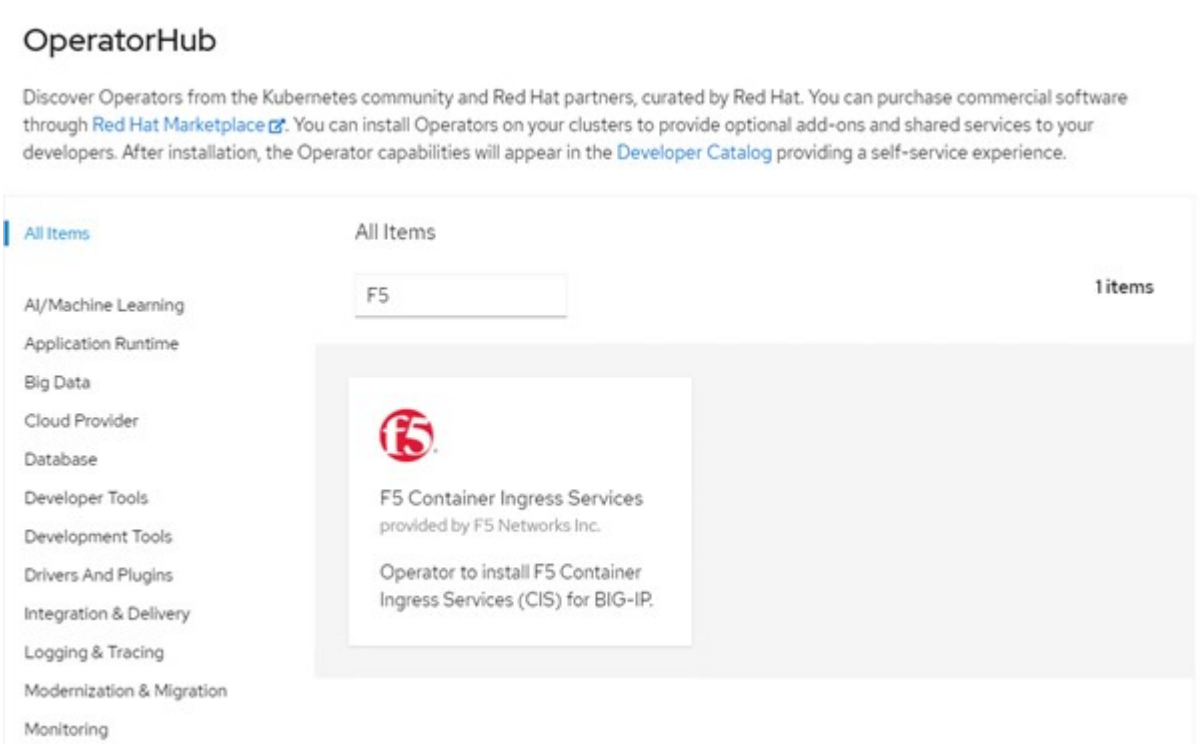
14. OperatorHub のオペレータを使用して F5 BIG-IP CIS をインストールします。cluster-admin 権限を持つ Red Hat OpenShift クラスタにログインし、 F5 BIG-IP システムログインクレデンシャルを使用してシークレットを作成します。これはオペレータの前提条件です。

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

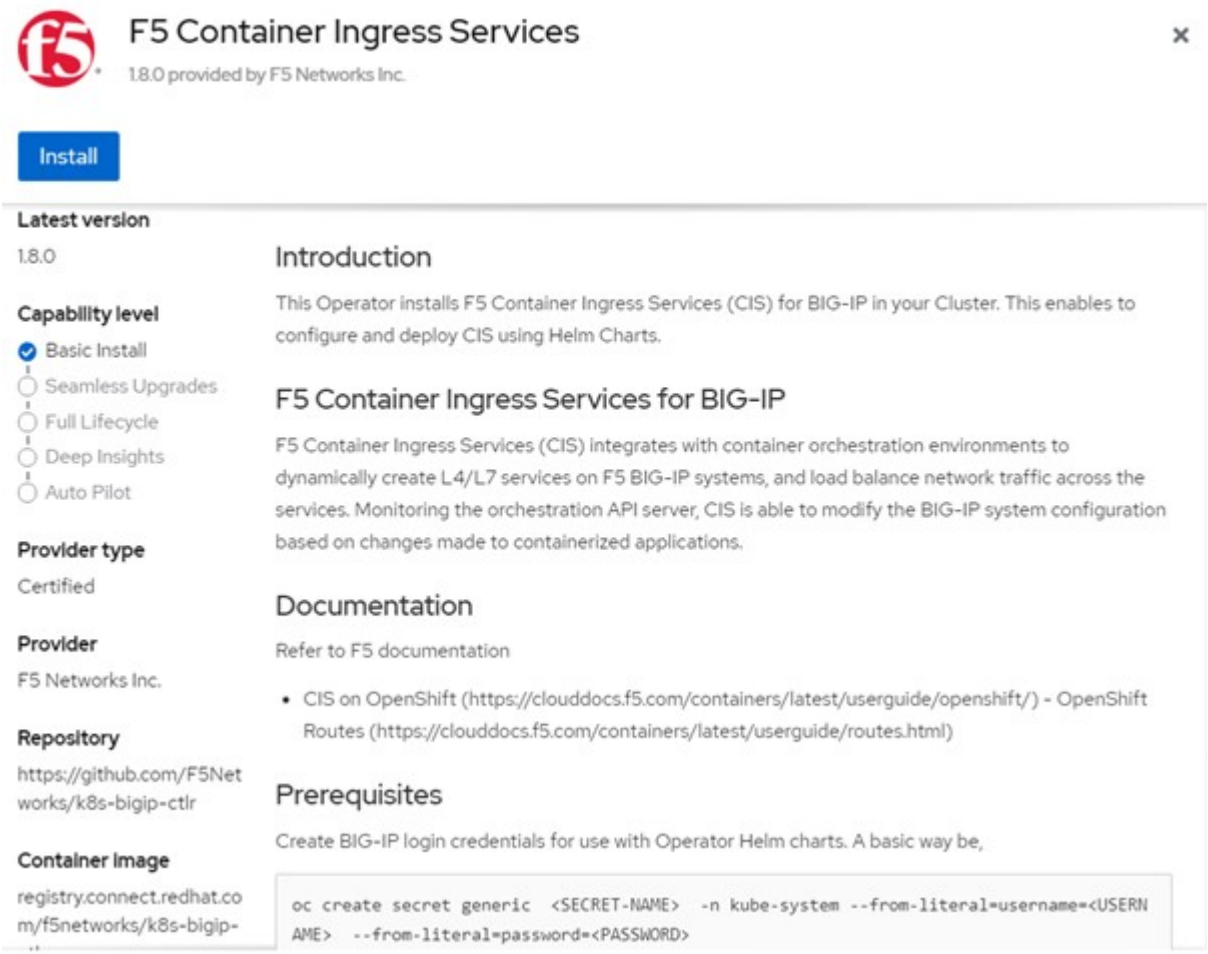
15. F5 CIS CRD をインストールします。

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

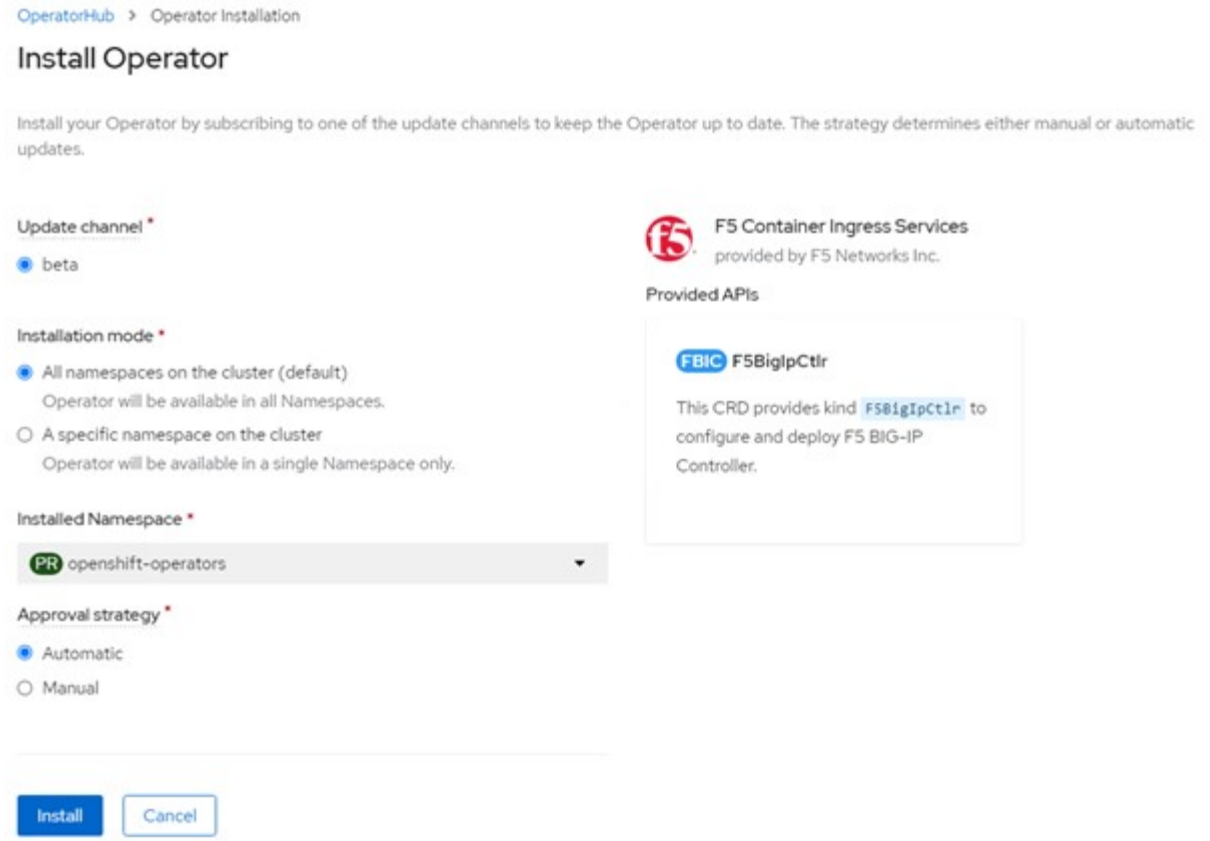
16. [演算子]>[演算子ハブ] に移動し、キーワード F5 を検索して、 F5 Container Ingress Service タイルをクリックします。



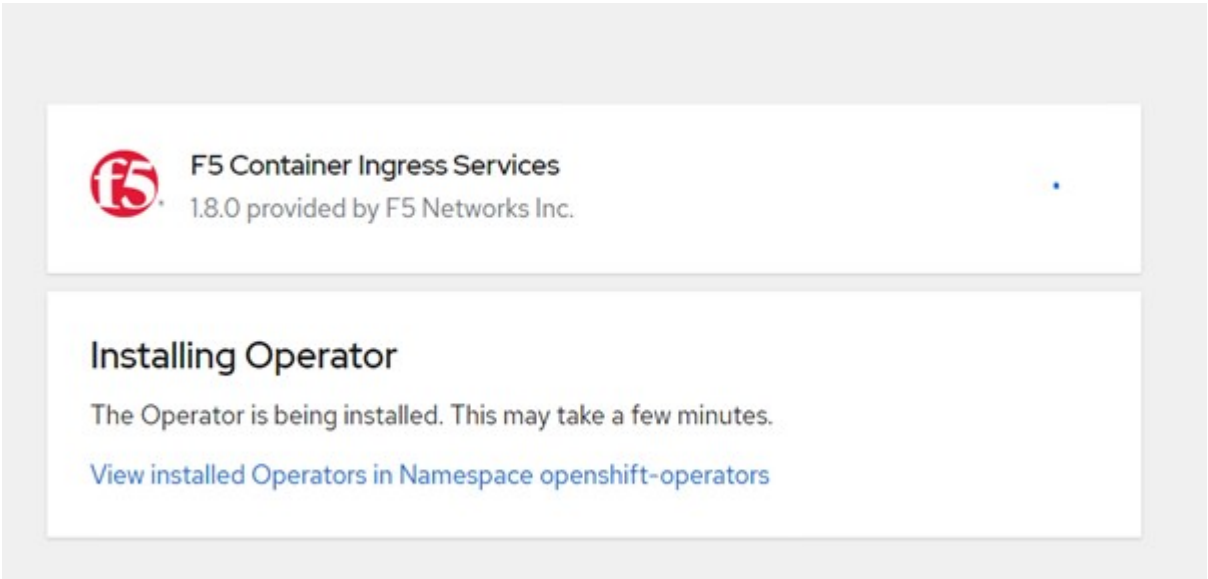
17. オペレータ情報を読み、 [インストール] をクリックします。



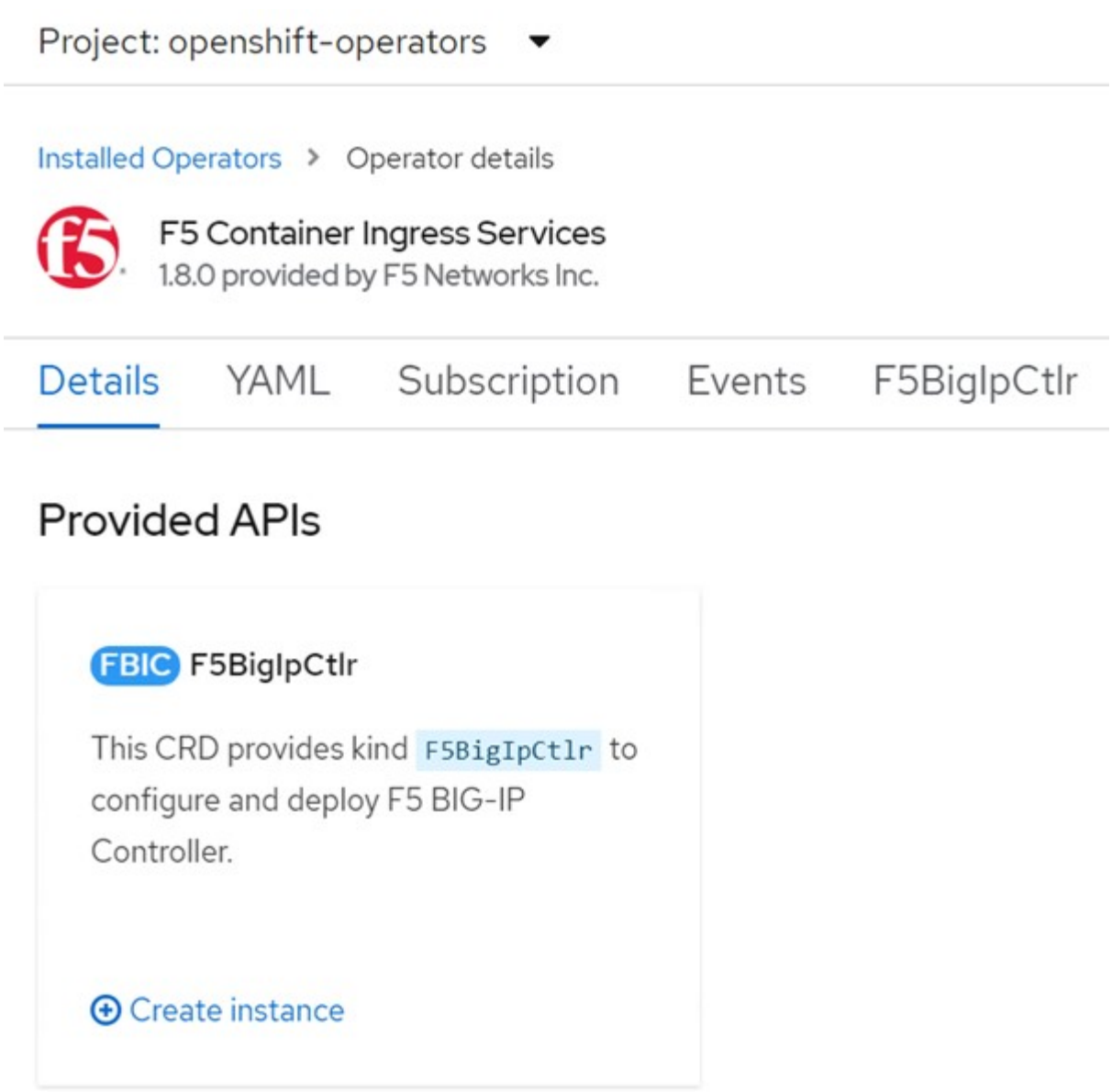
18. Install Operator （オペレータのインストール）画面で、デフォルトのパラメータをすべてそのままにして、 Install （インストール）をクリックします。



19. オペレータのインストールには時間がかかります。



20. オペレータがインストールされると、「 Installation Successful 」というメッセージが表示されます。
21. [演算子]>[インストールされている演算子] に移動し、 [F5BigIpCtlr] タイルの下にある [F5 Container Ingress Service] をクリックして、 [インスタンスの作成] をクリックします。

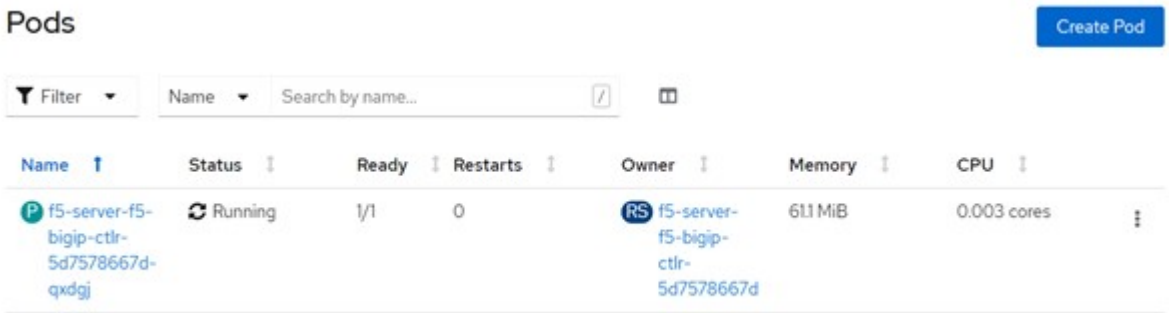


22. YAML View をクリックし、必要なパラメータを更新した後で次の内容を貼り付けます。

以下のパラメータ「 bigip_dpartition 」、「 OpenShift 」 SDN_NAME 」、「 bigip_url 」、「 bigip_login_secret 」を更新して、内容をコピーする前にセットアップの値を反映させます。

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. このコンテンツを貼り付けたら、 [作成] をクリックします。これにより、 CIS ポッドが kube-system 名前空間にインストールされます。



Red Hat OpenShift は、デフォルトで、 L7 ロードバランシングのルートを介してサービスを公開する方法を提供します。組み込みの OpenShift ルータは、これらのルートのトラフィックのアドバタイズと処理を行います。ただし、外部 F5 BIG-IP システムを介してルートをサポートするように F5 CIS を構成することもできます。このシステムは、補助ルータとして実行することも、自己ホスト型 OpenShift ルータに代わるものでもあります。CIS は、 OpenShift ルートのルータとして機能する BIG-IP システムに仮想サーバを作成し、BIG-IP はアドバタイズメントとトラフィックルーティングを処理します。この機能を有効にするためのパラメータについては、次のドキュメントを参照してください。これらのパラメータは、 APPS/v1 API の OpenShift Deployment リソースに対して定義されています。したがって、 F5BigIpCtlr リソース cis.f5.com/v1 API でこれらを使用する場合は、パラメータ名にハイフン (-) をアンダースコア (_) に置き換えます。

24. CIS リソースの作成に渡される引数には 'IPAM:true' と 'custom_resource_mode:true' がありますこれらのパラメータは 'IPAM コントローラとの CIS 統合を有効にするために必要ですF5 IPAM リソースを作成して 'CIS で IPAM 統合が有効になっていることを確認します

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE   NAME                                AGE
kube-system   ipam.10.61.181.19.ocp-vmw       43s
```

25. F5 IPAM コントローラに必要なサービスアカウント、ロール、およびロールバインドを作成します。YAML ファイルを作成し、次の内容を貼り付けます。

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. リソースを作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. YAML ファイルを作成し、下記の F5 IPAM 展開定義を貼り付けます。

以下の spec.template.spec.containers [0] の ip-range パラメータを更新して、設定に対応する ipamLabel と IP アドレス範囲を反映させます。

IPAM コントローラが定義された範囲から IP アドレスを検出して割り当てるには 'ipamLabels[`range1`' および `range2` を以下の例に示します] が 'LoadBalancer 型のサービスに注釈を付ける必要があります

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. F5 IPAM コントローラ配置を作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. F5 IPAM コントローラポッドが実行されていることを確認します。

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. F5 IPAM スキーマを作成します。

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

検証

1. LoadBalancer タイプのサービスを作成します

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. IPAM コントローラが外部 IP を割り当てるかどうかを確認します。

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. 導入環境を作成し、作成した LoadBalancer サービスを使用します。

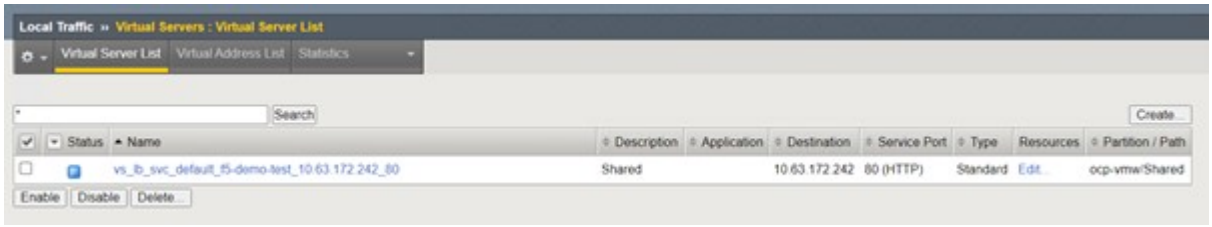
```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. ポッドが実行されているかどうかを確認します。

```
[admin@rhel-7 ~]$ oc get pods

NAME                            READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. 対応する仮想サーバが、 OpenShift の LoadBalancer タイプのサービス用に BIG-IP システムに作成されているかどうかを確認します。Local Traffic > Virtual Servers > Virtual Server List の順に選択します。



プライベートイメージレジストリを作成しています

Red Hat OpenShift の導入では、のようなパブリックレジストリを使用します "キー・IO" または "DockerHub" お客様のほとんどのニーズに対応ただし、お客様が独自のプライベートイメージまたはカスタマイズされたイメージをホストしたい場合があります。

この手順ドキュメントでは、 Astra Trident と NetApp ONTAP が提供する永続的ボリュームを使用して作成された、プライベートイメージレジストリを作成しています。

> Astra Control Center では、 Astra コンテナに必要なイメージをホストするためにレジストリが必要です。次のセクションでは、 Red Hat OpenShift クラスタにプライベートレジストリをセットアップし、 Astra Control Center のインストールをサポートするために必要なイメージをプッシュする手順について説明します。

プライベートイメージレジストリを作成しています

1. 現在のデフォルトストレージクラスからデフォルトのアノテーションを削除し、 OpenShift クラスタの Trident バック対象ストレージクラスをデフォルトとしてアノテートします。

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. 「 PEC 」セクションに以下の保管パラメータを入力して、 imagegeistry のオペレータを編集します。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. カスタムホスト名を使用して OpenShift ルートを作成するには、「 PEC 」セクションに次のパラメータを入力します。保存して終了します。

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

上記のルート設定は、ルートのカスタムホスト名が必要な場合に使用されます。OpenShift でデフォルトのホスト名を持つルートを作成するには、「 PEC 」セクションに「 defaultRoute ： true 」というパラメータを追加します。

カスタム **TLS** 証明書

ルートにカスタムホスト名を使用している場合、デフォルトでは、 OpenShift 入力オペレータのデフォルトの TLS 設定が使用されます。ただし、カスタム TLS 設定をルートに追加することはできます。これには、次の手順を実行します。

a. ルートの TLS 証明書とキーを使用して秘密を作成します。

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. imageregistry 演算子を編集して 'PEC' セクションに次のパラメータを追加します

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. このような場合は、すべての管理者をもう一度編集し、管理状態を「管理状態」に変更してください。保存して終了します。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. すべての前提条件を満たしている場合は、プライベートイメージレジストリに PVC 、ポッド、およびサービスが作成されます。数分後にレジストリが起動します。

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                  READY   STATUS
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                     0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                     0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                     0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                    1/1     Running
0          76m
pod/node-ca-bwb5r                                     1/1     Running
0          90d
pod/node-ca-f8w54                                     1/1     Running
0          90d
pod/node-ca-gjx7h                                     1/1     Running
0          90d
pod/node-ca-lcx4k                                     1/1     Running
0          33d
pod/node-ca-v7zmx                                     1/1     Running
0          7d21h
pod/node-ca-xpppp                                     1/1     Running
0          89d

NAME                               TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry             ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator    ClusterIP   None             <none>
60000/TCP   90d

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca    6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                           READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h
```

```
NAME                                                                DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                 COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                           SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                       HOST/PORT
PATH   SERVICES         PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. 入力オペレータ OpenShift レジストリルートにデフォルトの TLS 証明書を使用している場合は、次のコマンドを使用して TLS 証明書を取得できます。

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. OpenShift ノードがレジストリにアクセスしてイメージをプルできるようにするには、 OpenShift ノード上の Docker クライアントに証明書を追加します。TLS 証明書を使用して「 OpenShift -config 」ネームスペースに ConfigMap を作成し、証明書を信頼できるようにクラスタイメージ設定にパッチします。

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. OpenShift の内部レジストリは認証によって制御されます。OpenShift ユーザはすべて OpenShift レジストリにアクセスできますが、ログインユーザが実行できる操作はユーザ権限によって異なります。
   a. ユーザーまたはユーザーのグループがレジストリから画像をプルできるようにするには、ユーザーにレジストリビューアの役割が割り当てられている必要があります。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

   b. ユーザーまたはユーザーグループにイメージの書き込みまたはプッシュを許可するには、ユーザーにレジストリエディタの役割が割り当てられている必要があります。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

9. OpenShift ノードがレジストリにアクセスし、イメージをプッシュまたはプルするには、プルシークレットを設定する必要があります。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

10. このプルシークレットは、サービスアカウントにパッチを適用するか、対応するポッド定義で参照できます。
    a. サービスアカウントにパッチを適用するには、次のコマンドを実行します。

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. ポッド定義でプルシークレットを参照するには、「 PEC 」セクションに次のパラメータを追加します。

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. OpenShift ノードとは別にワークステーションからイメージをプッシュまたはプルするには、次の手順を実行します。

a. TLS 証明書を Docker クライアントに追加します。

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

b. OC ログインコマンドを使用して OpenShift にログインします。

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

c. podman/docker コマンドで OpenShift ユーザクレデンシャルを使用してレジストリにログインします。

ポッドマン

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+ 注 : 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、パスワードの代わりにトークンを使用します。

Docker です

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+ 注 : 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、パスワードの代わりにトークンを使用します。

d. 画像を押したり引いたりします。

ポッドマン

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

Docker です

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

著作権に関する情報

商標に関する情報