paloalto TECHDOCS

# CNシリーズ ファイアウォール デプロイメント モード

docs.paloaltonetworks.com

### Contact Information

Corporate Headquarters:
Palo Alto Networks
3000 Tannery Way
Santa Clara, CA 95054
www.paloaltonetworks.com/company/contact-support

### About the Documentation

- For the most recent version of this guide or for access to related documentation, visit the Technical Documentation portal docs.paloaltonetworks.com.
- To search for a specific topic, go to our search page docs.paloaltonetworks.com/search.html.
- Have feedback or questions for us? Leave a comment on any page in the portal, or write to us at documentation@paloaltonetworks.com.

### Copyright



### Last Revised

December 13, 2021

# Table of Contents



# クイック スタート - CNシリーズ ファイアウォールのデプロイメント

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

以下の手順を使用して、CN-Seriesのデプロイメントを開始します。

1. CSPアカウントにログインし、クレジットをアクティベートします。
2. デプロイメントプロファイルの作成。
3. CN シリーズ ファイアウォールにデバイス証明書をインストールします。
4. Kubernetes プラグインをインストールし、CN-Series 用 Panorama をセットアップする。
5. Palo Alto Networks GitHubリポジトリから CNシリーズのデプロイメント ファイルをダウンロードします。オンプレミスまたはクラウド デプロイメントでのネイティブKubernetesで使用するファイルをNative-k8sフォルダーから取得します。
6. HELMチャートリポジトリの有無にかかわらず、CN-Seriesをデプロイします。

   ***HELM***チャートを使用して*CN*シリーズ ファイアウォールを展開することをお勧めします。
7. Panorama を設定して Kubernetes デプロイメントをセキュリティで保護する

CN シリーズ ファイアウォールは、以下のデプロイメント モードでデプロイすることを選択できます。

- CNシリーズ ファイアウォールをKubernetesサービスとしてデプロイする(推奨されたデプロイメント モード)- CN シリーズ ファイアウォールは、クラスタ デプロイメント モデルでデプロイされます。このデプロイメントのモードでは、自動スケーリング機能を使用し、Kubernetesベースのネイティブ デプロイメント モデルを使用して、使用率の向上、コストの削減、拡張性の向上を実現します。
- CN-Series ファイアウォールを DaemonSet としてデプロイする-CNシリーズファイアウォールは分散型デプロイメント モデルでデプロイされます。このデプロイメント モードは、環境ごとに確保するノード数が少ない場合に適しています。

- CN-Series ファイアウォール を Kubernetes CNF としてデプロイする-このデプロイメントモードは、コンテナと非コンテナの両方のワークロードを保護します。スタンドアロンのレイヤー3デプロイメントとしてデプロイできます。

# CNシリーズ ファイアウォールのデプロイメント モード

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

CNシリーズ コアビルディング ブロックとCNシリーズ ファイアウォールによるKubernetesの安全なワークロード内のワークフローの概要を確認したら、同じクラスタ内のコンテナ間やコンテナと他のワークロード タイプ(仮想マシンやベアメタル サーバーなど)間のトラフィックを保護するためのCNシリーズ ファイアウォールのデプロイから始めることができます。

OpenShift 環境で作業している場合は、を参照してください。5G トラフィックの保護については、CN-Series ファイアウォールを使用して 5G をセキュリティで保護するを参照してください。

*Kubernetes* クラスタ、アプリケーション、およびファイアウォール サービスをデプロイして管理するためには、*kubectl* や *Helm* などの標準の *Kubernetes* ツールが必要です。*Panorama* は、*Kubernetes* クラスタのデプロイメントと管理用のオーケストレーターになるようには設計されていません。クラスタ管理用のテンプレートがマネージド *Kubernetes* プロバイダから提供されています。*Palo Alto Networks* は、Helm および Terraform で *CN-Series* をデプロイするためのコミュニティサポートのテンプレートを提供しています。

- CNシリーズ ファイアウォールをKubernetesサービスとしてデプロイする(推奨されたデプロイメント モード)
- CN-Series ファイアウォールを DaemonSet としてデプロイする
- CN-Series ファイアウォール を Kubernetes CNF としてデプロイする
- Kubernetes CNF L3をスタンドアロンモードでデプロイする

*CN-Series* を *DaemonSet* としての *CN-Series* からサービスとしての *CN-Series*、またはその逆に移行する前に、`plugin-serviceaccount.yaml` を削除して再適用する必要があります。

- *DaemonSet* としての *CN-Series* をデプロイする場合、`pan-plugin-cluster-mode-secret` が存在してはなりません。
- *CN-Series* を *Kubernetes* サービスとしてデプロイする場合は、`pan-plugin-cluster-mode-secret` が存在する必要があります。

# CNシリーズ ファイアウォールをKubernetesサービスとしてデプロイする(推奨されたデプロイメント モード)

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

以下の手順を実行して、CN-Series ファイアウォールを Kubernetes サービスとしてデプロイします。

開始する前に、CN-Series YAML ファイルのバージョンが PAN-OS バージョンと互換性があることを確認します。

- PAN-OS10.1.2 以降には YAML 2.0.2 が必要です
- PAN-OS10.1.0 および 10.1.1 には YAML 2.0.0 または 2.0.1 が必要です

**STEP 1 |** Kubernetes クラスタをセットアップします。

1. クラスタのリソースが適切であることを確認します。クラスタにファイアウォールをサポートするためのCNシリーズの前提条件のリソースが含まれていることを確認します。

   **kubectl get nodes**

   **kubectl describe node <node-name>**

   コマンド出力の[容量]見出しの下にある情報を表示して、指定したノードで使用可能な CPU とメモリーを確認します。

   CPU、メモリ、およびディスク ストレージの割り当てはニーズによって異なります。CNシリーズのパフォーマンスとスケーリングを参照してください。

   以下の情報があることを確認してください。

   - Panorama 上で API サーバーをセットアップするためのエンドポイント IP アドレスを収集します。Panorama は、この IP アドレスを使用して、Kubernetes クラスタに接続します。
   - テンプレート スタック名、デバイス グループ名、Panorama IP アドレス、およびオプションで Panorama からログ コレクタ グループ名を収集します。
   - VM 認証キーと自動登録の PIN ID と値を収集します。
   - イメージをダウンロードしたコンテナ イメージ リポジトリの場所。

**STEP 2 |** （任意）Panorama の Kubernetes プラグインでカスタム証明書を設定した場合は、次のコマンドを実行して証明書シークレットを作成する必要があります。ファイル名を ca.crt から変更しないでください。pan-cn-mgmt.yaml および pan-cn-ngfw.yaml のカスタム証明書のボリュームはオプションです。

```
kubectl -n kube-system create secret generic custom-ca --from-file =
ca.crt
```

**STEP 3 |** YAML ファイルを編集して、CN-Series ファイアウォールをデプロイするために必要な詳細を記入します。

プライベート レジストリへのパスを含み、必要なパラメータを提供するように YAML ファイル内のイメージ パスを置き換える必要があります。詳細は CN-Series デプロイメント YAML ファイル内の編集可能なパラメータ を参照してください。

**STEP 4 |** （AWS Outpost 上の EKS の CN-Series のみ）ストレージクラスを更新します。AWS Outpost にデプロイされた CN-Series をサポートするには、ストレージドライバーaws-ebs-csi-driver を使用する必要があります。これにより、動的永続ボリューム（PV）の作成中に Outpost がOutpost からボリュームをプルします。

1. 以下の yaml を適用します。

   ```
   kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/
   deploy/kubernetes/overlays/stable/?ref=release-0.10"
   ```

2. ebs-sc コントローラーが実行されていることを確認します。

   ```
   kubectl -n kube-system get pods
   ```

3. 以下の例に一致するように pan-cn-storage-class.yaml を更新します。

   ```
   apiVersion: v1 kind:StorageClass apiVersion: storage.k8s.io/
   v1 metadata: name: ebs-sc provisioner: ebs.csi.aws.com
    volumeBindingMode:WaitForFirstConsumer parameters: type: gp2
   ```

4. 以下に示す場所で、**storageClassName**：**ebs-sc** を pan-cn-mgmt.yaml に追加します。

   ```
   volumeClaimTemplates: - metadata: name: panlogs spec:
    #storageClassName: pan-cn-storage-class //For better disk
    iops performance for logging accessModes: [ "ReadWriteOnce" ]
    storageClassName: ebs-sc // resources: requests: storage:20Gi
    # ディスク IOPS を向上させるためにstorageClassName を使用
   しているときにこれを 200Gi に変更します - metadata: name:
    varlogpan spec: #storageClassName：pan-cn-storage-
   class // dp ログのディスク IOPS パフォーマンスを向上させるため
    accessModes：["ReadWriteOnce"] storageClassName： ebs-
   sc resources: requests: storage:20Gi # ディスク IOPS 向
   上のために storageClassName を使用している間、これを 200Gi に
   変更します - metadata: name: varcores spec: accessModes:
    [ "ReadWriteOnce" ] storageClassName: ebs-sc resources:
    requests: storage:2Gi - metadata: name: panplugincfg spec:
    accessModes: [ "ReadWriteOnce" ] storageClassName: ebs-sc
    resources: requests: storage:1Gi - metadata: name: panconfig
    spec: accessModes: [ "ReadWriteOnce" ] storageClassName:
   ```

```
ebs-sc resources: requests: storage:8Gi - metadata:
name: panplugins spec: accessModes: [ "ReadWriteOnce" ]
storageClassName: ebs-sc resources: requests: storage:200Mi
```

**STEP 5 |** Kubernetes 環境で自動スケーリングを使用している場合は、続行する前に水平ポッドの自動スケーリングを参照してください。

**STEP 6 |** CN-NGFW サービスをデプロイします。

1. pan-cni-serviceaccount.yaml を使用してサービス アカウントが作成されたことを確認します。

   クラスタ認証用にサービス アカウントを作成するを参照してください。

2. Kubectl を使用して pan-cni-configmap.yaml を実行します。

   **kubectl apply -f pan-cni-configmap.yaml**

3. kubectl を使用して pan-cn-ngfw-svc.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-svc.yaml**

   この *yaml* は *pan-cni.yaml* の前にデプロイする必要があります。

4. Kubectl を使用して pan-cni.yaml を実行します。

   **kubectl apply -f pan-cni.yaml**

5. pan-cni-configmap YAML ファイルと pan-cni YAML ファイルが変更されたことを確認します。
6. 以下のコマンドを実行して、出力が以下の例のようになっていることを確認します。

   **kubectl get pods -n kube-system | grep pan-cni**

```
@cloudshell:~/Kubernetes-master/pan-cn-k8s-service/gke (v  eries-mktplace)$ kubectl get pods -n
pan-cni-nmqkf                                               Running   0          2m11s
pan-cni-wjrkq                                               Running   0          2m11s
pan-cni-xrc2z                                               Running   0          2m12s
@cloudshell:~/Kubernetes-master/pan-cn-k8s-service/gke (v  eries-mktplace)$
```

**STEP 7 |** CN-MGMT StatefulSet をデプロイします。

デフォルトで、管理プレーンは耐障害性を提供する StatefulSet としてデプロイされます。1 つの CN-MGMT StatefulSet に最大 30 個のファイアウォール CN-NGFW ポッドを接続できます。

1. (静的にプロビジョニングされた PV のみに必要) CN-MGMT StatefulSet 用の永続ボリューム (PV) をデプロイします。

   1. pan-cn-pv-local.yaml で定義されたローカル ボリューム名と一致するディレクトリを作成します。

      少なくとも 2 つのワーカー ノード上に 6 つのディレクトリが必要です。CN-MGMT StatefulSet をデプロイする各ワーカー ノードにログインして、ディレクトリを作成

します。たとえば、/mnt/pan-local1 から /mnt/pan-local6 という名前のディレクトリを作成するには、次のコマンドを使用します。

```
mkdir -p /mnt/pan-local1 /mnt/pan-local2 /mnt/pan-local3 /
mnt/pan-local4 /mnt/pan-local5 /mnt/pan-local6
```

2. pan-cn-pv-local.yaml を変更します。

   nodeaffinity の下でホスト名を一致させ、上記で spec.local.path に作成したディレクトリが変更されたことを確認してから、そのファイルをデプロイして、新しいストレージ クラス pan-local-storage とローカル PV を作成します。

2. pan-cn-mgmt-configmap YAML ファイルと pan-cn-mgmt YAML ファイルが変更されたことを確認します。

   EKS から pan-cn-mgmt-configmap をサンプリングします。

```
 apiVersion: v1 kind:ConfigMap metadata: name: pan-mgmt-
config namespace: kube-system data:PAN_SERVICE_NAME: pan-
mgmt-svc PAN_MGMT_SECRET: pan-mgmt-secret # Panorama 設
定 PAN_PANORAMA_IP: "<panorama-IP>" PAN_DEVICE_GROUP:
 "<panorama-device-group>" PAN_TEMPLATE_STACK: "<panorama-
template-stack>" PAN_CGNAME: "<panorama-collector-
group>" # ctnr mode: "k8s-service", "k8s-ilbservice"
 PAN_CTNR_MODE_TYPE: "k8s-service" #必須でないパラメータ #
 Panorama Kubernetes プラグインで提供されるクラスタ名と同じ名前
を持つことを推奨 - 同じ Panorama で複数のクラスタを管理する場合、
ポッドの識別が容易になります #CLUSTER_NAME: "<Cluster name>"
 #PAN_PANORAMA_IP2: "" # CERT を使用する場合はコメントアウト
します。それ以外の場合は、pan-mgmt と pan-ngfw 間の IPsec 用に
 PSK を使用します #IPSEC_CERT_BYPASS: "" # 値は不要 # jumbo-
frame モードの自動検出をオーバーライドし、システム全体を強制的に有
効にします #PAN_JUMBO_FRAME_ENABLED: "true" # GTP を有効にし
て MGMT を起動します。完全な機能を実現するには、Panorama でも GTP
 # を有効にする必要があります。#PAN_GTP_ENABLED:「true」# 高い
機能容量を有効にします。これらは MGMT ポッドには高いメモリを必要と
し、NGFW ポッドには以下の指定以上のメモリが必要です。#PAN_NGFW_MEMORY =
 "6Gi」#PAN_NGFW_MEMORY = "40Gi」# より高速なデータパス-AF_XDP を有
効にするには、デフォルトは AF_PACKETV2 です。これにはカーネルのサポート
が必要です。#PAN_DATA_MODE: "次世代" #HPA params #PAN_CLOUD:"EKS"
 #PAN_NAMESPACE_EKS:"EKSNamespace" #PUSH_INTERVAL:"15" #AWS
 cloudwatch にメトリクスを発行する間隔
```

pan-cn-mgmt.yaml のサンプル

```
initContainers: - name: pan-mgmt-init image: <your-private-
registry-image-path>
```

```
 containers: - name: pan-mgmt image: <your-private-registry-
image-path> terminationMessagePolicy:FallbackToLogsOnError
```

3. Kubectl を使用して yaml ファイルを実行します。

   **kubectl apply -f pan-cn-mgmt-configmap.yaml**

   **kubectl apply -f pan-cn-mgmt-slot-crd.yaml**

   **kubectl apply -f pan-cn-mgmt-slot-cr.yaml**

   **kubectl apply -f pan-cn-mgmt-secret.yaml**

   **kubectl apply -f pan-cn-mgmt.yaml**

   pan-mgmt-serviceaccount.yamlは、クラスター認証用のサービスアカウントの作成を以前に完了していない場合にのみ実行する必要があります。

4. CN-MGMT ポッドが起動していることを確認します。

   これには、5 ～ 6 分かかります。

   **kubectl get pods -l app=pan-mgmt -n kube-system** を使用します。

**STEP 8 |** CN-NGFW ポッドをデプロイします。

1. PAN-CN-NGFW-CONFIGMAP と PAN-CN-NGFW に詳述されているように YAML ファイルが変更されたことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

2. Kubectl apply を使用して pan-cn-ngfw-configmap.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap.yaml**

3. Kubectl apply を使用して pan-cn-ngfw.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw.yaml**

4. CN-NGFW ポッドがデプロイされたことを確認します。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

**STEP 9 |** 「CN-Series で水平ポッド自動スケーリングを有効にする」を行います。

**STEP 10 |** Kubernetes クラスタ上の CN-MGMT、CN-NGFW、および PAN-CNI が表示されていることを確認します。

**kubectl -n kube-system get pods**

**STEP 11 |** 新しいポッドからのトラフィックがファイアウォールにリダイレクトされるようにアプリケーション yaml または名前空間に注釈を付けます。

検査のためにトラフィックを CN-NGFW にリダイレクトするには、以下のアノテーションを追加する必要があります：

```
annotations：paloaltonetworks.com/firewall：pan-fw
```

たとえば、「default」名前空間のすべての新しいポッドの場合：

```
kubectl annotate namespace default paloaltonetworks.com/
firewall=pan-fw
```

一部のプラットフォームでは、*CNI* プラグイン チェーン内で *pan-cni* がアクティブになっていない状態でアプリケーション ポッドが開始する可能性があります。このようなシナリオを回避するには、アプリケーションポッド *YAML* にここに示すようにボリュームを指定する必要があります。

```
volumes: - name: pan-cni-ready hostPath: path: /var/log/
pan-appinfo/pan-cni-ready type:ディレクトリ
```

**STEP 12 |** (オプション)*PortInfo* カスタム リソースに基づいて、特定のトラフィックがファイアウォールをバイパスできます。

1. PortInfo カスタムリソース定義 (YAML) を適用

   ```
   kubectl apply-f pan-cn-ngfw-port-crd.yaml
   ```

2. pan-cn-ngfw-port-cr.yaml を例として使用して、バイパスしたいプロトコルとポートを含む PortInfo カスタムリソースを作成します。アプリポッドからはアウトバウンド方向のみで、TCP と UDP をサポートし、最大 10 個の個別ポート (ポート範囲なし) をサポートします。

   ```
   apiVersion: "paloaltonetworks.com/v1" kind:PortInfo metadata:
   name: "bypassfirewall" namespace: kube-system spec:
   portinfo:"TCP:8080,TCP:8081"
   ```

3. PortInfo カスタム リソース YAML を適用してください。

   ```
   kubectl apply -f pan-cn-ngfw-port-cr.yaml
   ```

4. pan-fw 注釈に加えて、アプリポッドに注釈を付けます。注釈は、アプリポッドの起動時に表示されるはずです。

   ```
   annotations: paloaltonetworks.com/firewall: pan-fw
   paloaltonetworks.com/bypassfirewall: kube-system/
  bypassfirewall
   ```

**STEP 13 |** クラスタでアプリケーションをデプロイします。

# CN-Series で水平ポッド自動スケーリングを有効にする

| どこで使用できますか? | 何が必要ですか? |
|---|---|
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

水平ポッド自動スケーラー(HPA)は、すべてのクラウド環境で利用可能な Kubernetes リソースで、監視対象のメトリクスに基づいてデプロイメント内の CN-MGMT ポッドと CN-NGFW ポッドの数を自動的にスケーリングします。HPA では、すべてのクラウド環境で、CPU とメモリの使用率という 2 つの標準メトリクスと、各クラウド環境に固有のカスタム メトリクスを使用します。そのため、各クラウドでは、AKS、EKS、および GKE で HPA を有効にするために特定の yaml ファイルが必要です。

HPA は、クラウド固有のメトリクスアダプターを使用して、クラウド環境のモニタリングアダプター (EKS の CloudWatch など) からメトリクスデータを取得し、定義したしきい値に基づいてスケールアップまたはスケールダウンするタイミングを決定します。必要な yaml ファイルを変更して、レプリカの最小数と最大数、各メトリックのしきい値、およびファイアウォールの自動スケールに使用するメトリックを設定する必要があります。

*PAN OS 10.1*では、*CN-MGMT*ポッドの*HPA*スケーリングを使用すると、*DP*ポッドが接続されていない*CN-MGMT*ポッドを多数スケーリングする場合があります。不要なスケーリングを防ぐために、*CN-MGMT*ポッドのレプリカを最大数作成することをお勧めします。

| クラウド環境 | メトリック | | 平均値 |
|---|---|---|---|
| AKS、EKS、および GKE | CN-MGMT | panloggingrate | ログ数 |
| | | pandataplaneslots | データプレーン スロット数 |
| | CN-NFGW | dataplanecpuutilizationpct | CN-NGFW CPU 使用率の割合 |
| | | dataplanepacketbufferutilization | CN-NGFW パケット バッファ使用率のパーセント |

| クラウド環境 | メトリック | | 平均値 |
|---|---|---|---|
| | | pansessionactive | CN-NGFW でアクティブなセッション数 |
| | | pansessionutilization | セッション使用率の割合 |
| | | pansessionsslproxyutilization | セッション SSL プロキシの使用率 |
| | | panthroughput | kbps 単位のスループット |
| | | panpacketrate | 1 秒あたりのパケット数のパケット レート (pps) |
| | | panconnectionspersecond | Connections per Second（接続数/秒） |

以下の例は、EKS 用の pan-cn-hpa-dp.yaml ファイルです。この例では、データプレーンの CPU 使用率の割合を使用して CN-NGFW ポッドを自動スケールします。25% で、クラスターはスケール アップします。CPU 使用率が 50% に達すると、クラスターはポッドを 1 つ追加でデプロイします。CPU 使用率が 75% に達すると、クラスターは 2 つの追加ポッドをデプロイします。これは、メトリックのしきい値で合計メトリックを割り、メトリックをクラスター内のすべての CN-NGFW ポッドで設定しきい値まで下げるのに十分なポッドをデプロイすることで決定されます。ただし、クラスターは、maxReplicas より多くの CN-NGFW ポッドをデプロイしません。複数のメトリックが同時にしきい値を超えた場合、クラスターは、より高いメトリックに対処するために必要な数のポッドをデプロイします。

デフォルトでは、HPA アダプタは 15 秒ごとにメトリック アダプタをポーリングします。指定したメトリックが設定されたしきい値を 60 秒間超えた場合、クラスターは追加の CN-NGFW ポッドをデプロイします。その後、クラスターは 300 秒 (5 分) 待機してから、追加の CN-NGFW ポッドが必要かどうかを判断します。デフォルトでは、一度に 1 つのポッドがデプロイされます。クラスターは、300 秒後にメトリック (この場合は CPU 使用率) をチェックします。使用率がポッドが不要になったレベルまで下がった場合、クラスターはポッドを削除します。その後、クラスターはさらに 60 秒待機してから、別のポッドを削除できるかどうかを判断します。

以下に示すすべての値と任意のメトリックの値は、デプロイメントに最適となるように変更できます。

```
kind:HorizontalPodAutoscaler apiVersion: autoscaling/v2beta2
 metadata: name: hpa-dp-eks namespace: kube-system spec:
 scaleTargetRef: apiVersion: apps/v1beta1 kind:デプロイメン
ト名: pan-ngfw-dep minReplicas:1 maxReplicas:10 behavior:
 scaleDown: stabilizationWindowSeconds:300 policies: - type:ポッ
ド値:1 periodSeconds:60 - type:パーセント値:1 periodSeconds:60
 selectPolicy:Max scaleUp: stabilizationWindowSeconds:60 policies:
 - type:ポッド値:1 periodSeconds:300 # dp の準備時間を 5 分と想定 -
```

```
type:パーセント値:1 periodSeconds:300 # dp の準備時間を 5 分と想定
selectPolicy:最大メトリック: - type:External external: metric: name:
dataplaneCpuUtilizationPct target: type:Value value:25
```

## AKS

**STEP 1 |** クラスター内に Azure Application Insights インスタンスをデプロイします。K8s シークレットとして、必要な Azure Application Insights インストルメンテーション キーと Azure Application Insights APP ID API キーを指定する必要があります。

**STEP 2 |** Palo Alto Networks GitHub リポジトリから AKS 固有の HPA yaml ファイルをダウンロードします。

**STEP 3 |** CN-MGMT がカスタム名前空間にデプロイされている場合は、カスタム名前空間を使用して pan-cn-adapater.yaml を更新します。デフォルトの名前空間は **kube-system** です。

**STEP 4 |** まだ更新していない場合は、AKS 固有の **pan-cn-mgmt-configmap.yaml**の HPA パラメータを更新します。

```
 #PAN_CLOUD:"AKS" #HPA_NAME: "<name>" #名前空間またはテナントごとに
HPA リソースを識別する固有名 #PAN_INSTRUMENTATION_KEY: "<>" #Azure
APP Insight インストルメンテーション キー #PUSH_INTERVAL:"15" # Azure
Application Insights にメトリックを公開する時間間隔
```

**STEP 5 |** **pan-cn-hpa-secret.yaml** を編集します。

```
appinsights-appid: "<Azure App Insight Application ID obtained
 from API Access>" appinsights-key: "<Azure App Insight API Key
 created under API Access>" azure-client-id: "<Azure SP APP ID
 associated with corresponding resource group with monitoring
 reader access>" azure-client-secret: "<Azure SP Password
 associated with corresponding resource group with monitoring
 reader access>" azure-tenant-id: "<Azure SP tenant ID associated
 with corresponding resource group with monitoring reader access>"
```

**STEP 6 |** 上記で作成した HPA 名を、**pan-cn-custommetrics.yaml** の適切な場所に追加します。

**STEP 7 |** **pan-cn-hpa-dp.yaml** と **pan-cn-hpa-mp.yaml** を変更します。

1. レプリカの最小数と最大数を入力します。
2. (任意) スケールダウンを変更し、デプロイメントに合わせて頻度値をスケールアップします。これらの値を変更しない場合は、デフォルト値が使用されます。
3. スケーリングに使用する各メトリックについて、以下のセクションをコピーします。

```
- type:Pods pods: metric: name: pansessionactive target:
 type:AverageValue averageValue:30
```

4. 使用するメトリックの名前を変更し、**averageValue** を上記の表で説明したしきい値に設定します。これらの値を変更しない場合は、デフォルト値が使用されます。
5. 変更を保存します。

**STEP 8 |** HPA yaml ファイルをデプロイします。ファイルは、以下に説明する順序でデプロイする必要があります。

1. Kubectl を使用して pan-cn-hpa-secret.yaml を実行します

   **kubectl apply -f pan-cn-hpa-secret.yaml**
2. Kubectl を使用して pan-cn-adapter.yaml を実行します

   **kubectl apply -f pan-cn-adapter.yaml**
3. Kubectl を使用して pan-cn-custommetrics.yaml を実行します

   **kubectl apply -f pan-cn-custommetrics.yaml**
4. Kubectl を使用して pan-cn-hpa-dp.yaml を実行します

   **kubectl apply -f pan-cn-hpa-dp.yaml**
5. Kubectl を使用して pan-cn-hpa-mp.yaml を実行します

   **kubectl apply -f pan-cn-hpa-mp.yaml**

**STEP 9 |** デプロイメントを確認します。

- kubectl を使用して、カスタム メトリックス名前空間内のカスタム メトリック アダプター ポッドを確認します。

  **kubectl get pods -n custom-metrics**
- kubectl を使用して、HPA リソースを確認します。

  **kubectl get hpa -n kube-system**

  **kubectl describe hpa <hpa-name> -n kube-system**

## EKS

**STEP 1 |** CN-Series で、Kubernetes 用 Amazon CloudWatch メトリクス アダプターをサービスクラスターとしてデプロイします。CloudWatch に、Kubernetes ポッドとクラスターに関連付けられた両方の IAM ロールへの完全なアクセスを許可する必要があります。カスタムメトリクスを CloudWatch に公開するには、HPA がそれらを取得できるように、ワーカーノードのロールに AWS 管理ポリシー **CloudWatchAgentServerPolicy** が必要です。

**STEP 2 |** Palo Alto Networks GitHub リポジトリから EKS 固有の HPA yaml ファイルをダウンロードします。

**STEP 3 |** CN-MGMT がカスタム名前空間にデプロイされている場合は、カスタム名前空間を使用して pan-cn-adapater.yaml を更新します。デフォルトの名前空間は **kube-system** です。

**STEP 4 |** **pan-cn-hpa-dp.yaml** と **pan-cn-hpa-mp.yaml** を変更します。

1. レプリカの最小数と最大数を入力します。
2. (任意) スケールダウンを変更し、デプロイメントに合わせて頻度値をスケールアップします。これらの値を変更しない場合は、デフォルト値が使用されます。
3. スケーリングに使用する各メトリックについて、以下のセクションをコピーします。

```
- type:Pods pods: metric: name: pansessionactive target:
 type:AverageValue averageValue:30
```

4. 使用するメトリックの名前を変更し、**averageValue** を上記の表で説明したしきい値に設定します。これらの値を変更しない場合は、デフォルト値が使用されます。
5. 変更を保存します。

**STEP 5 |** HPA yaml ファイルをデプロイします。ファイルは、以下に説明する順序でデプロイする必要があります。

1. Kubectl を使用して pan-cn-adapter.yaml を実行します

   **kubectl apply -f pan-cn-adapter.yaml**
2. Kubectl を使用して pan-cn-externalmetrics.yaml を実行します

   **kubectl apply -f pan-cn-externalmetrics.yaml**
3. Kubectl を使用して pan-cn-hpa-dp.yaml を実行します

   **kubectl apply -f pan-cn-hpa-dp.yaml**
4. Kubectl を使用して pan-cn-hpa-mp.yaml を実行します

   **kubectl apply -f pan-cn-hpa-mp.yaml**

**STEP 6 |** デプロイメントを確認します。

- kubectl を使用して、カスタム メトリックス名前空間内のカスタム メトリック アダプター ポッドを確認します。

  **kubectl get pods -n custom-metrics**
- kubectl を使用して、HPA リソースを確認します。

  **kubectl get hpa -n kube-system**

  **kubectl describe hpa <hpa-name> -n kube-system**

# CN-Series ファイアウォールを DaemonSet としてデプロイする

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

CN-Series ファイアウォールを Daemonset としてデプロイするには、次の手順を実行します。

開始する前に、CN-Series YAML ファイルのバージョンが PAN-OS バージョンと互換性があることを確認します。

- PAN-OS10.1.2 以降には YAML 2.0.2 が必要です
- PAN-OS10.1.0 および 10.1.1 には YAML 2.0.0 または 2.0.1 が必要です

**STEP 1 |** Kubernetes クラスタをセットアップします。

1. クラスタのリソースが適切であることを確認します。クラスタにファイアウォールをサポートするためのCN シリーズの前提条件のリソースが含まれていることを確認します。

   **kubectl get nodes**

   **kubectl describe node <node-name>**

   コマンド出力の[容量]見出しの下にある情報を表示して、指定したノードで使用可能な CPU とメモリーを確認します。

   CPU、メモリ、およびディスク ストレージの割り当てはニーズによって異なります。CNシリーズのパフォーマンスとスケーリングを参照してください。

   以下の情報があることを確認してください。

   - Panorama 上で API サーバーをセットアップするためのエンドポイント IP アドレスを収集します。Panorama は、この IP アドレスを使用して、Kubernetes クラスタに接続します。
   - テンプレート スタック名、デバイス グループ名、Panorama IP アドレス、およびオプションで Panorama からログ コレクタ グループ名を収集します。
   - 認証コードと自動登録の PIN ID と値を収集します。
   - イメージをダウンロードしたコンテナ イメージ リポジトリの場所。

**STEP 2 |** （任意）Panorama の Kubernetes プラグインでカスタム証明書を設定した場合は、次のコマンドを実行して証明書シークレットを作成する必要があります。ファイル名を ca.crt から変更しないでください。pan-cn-mgmt.yaml および pan-cn-ngfw.yaml のカスタム証明書のボリュームはオプションです。

```
kubectl -n kube-system create secret generic custom-ca --from-file =
ca.crt
```

**STEP 3 |** YAML ファイルを編集して、CN-Series ファイアウォールをデプロイするために必要な詳細を記入します。

プライベート レジストリへのパスを含み、必要なパラメータを提供するように YAML ファイル内のイメージ パスを置き換える必要があります。詳細は CN-Series デプロイメント YAML ファイル内の編集可能なパラメータ を参照してください。

**STEP 4 |** CNI DaemonSet をデプロイします。

CNI コンテナは、DaemonSet (ノードあたり 1 つのポッド) としてデプロイされ、ノード上にデプロイされたアプリケーションごとに 2 つずつのインターフェースを CN-NGFW ポッド上に作成します。kubectl コマンドを使用して pan-cni YAML ファイルを実行すると、それが各ノード上のサービス チェーンに組み込まれます。

1. pan-cni-serviceaccount.yaml を使用してサービス アカウントが作成されたことを確認します。

   クラスタ認証用にサービス アカウントを作成するを参照してください。

2. Kubectl を使用して pan-cni-configmap.yaml を実行します。

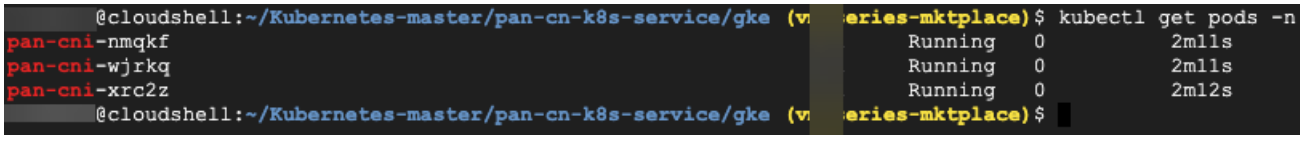
   ```
   kubectl apply -f pan-cni-configmap.yaml
   ```

3. Kubectl を使用して pan-cni.yaml を実行します。

   ```
   kubectl apply -f pan-cni.yaml
   ```

4. pan-cni-configmap YAML ファイルと pan-cni YAML ファイルが変更されたことを確認します。
5. 以下のコマンドを実行して、出力が以下の例のようになっていることを確認します。

   ```
   kubectl get pods -n kube-system | grep pan-cni
   ```




**STEP 5 |** （AWS Outpost 上の EKS の CN-Series のみ）ストレージクラスを更新します。AWS Outpost にデプロイされた CN-Series をサポートするには、ストレージドライバーaws-

ebs-csi-driver を使用する必要があります。これにより、動的永続ボリューム（PV）の作成中に Outpost がOutpost からボリュームをプルします。

1. 以下の yaml を適用します。

   **kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-0.10"**

2. ebs-sc コントローラーが実行されていることを確認します。

   **kubectl -n kube-system get pods**

3. 以下の例に一致するように pan-cn-storage-class.yaml を更新します。

```
apiVersion: v1 kind:StorageClass apiVersion: storage.k8s.io/
v1 metadata: name: ebs-sc provisioner: ebs.csi.aws.com
 volumeBindingMode:WaitForFirstConsumer parameters: type: gp2
```

4. 以下に示す場所で、**storageClassName**：**ebs-sc** を pan-cn-mgmt.yaml に追加します。

```
volumeClaimTemplates: - metadata: name: panlogs spec:
 #storageClassName: pan-cn-storage-class //For better disk
 iops performance for logging accessModes: [ "ReadWriteOnce" ]
 storageClassName: ebs-sc // resources: requests: storage:20Gi
 # ディスク IOPS を向上させるためにstorageClassName を使用
しているときにこれを 200Gi に変更します - metadata: name:
 varlogpan spec: #storageClassName：pan-cn-storage-
class // dp ログのディスク IOPS パフォーマンスを向上させるため
 accessModes：["ReadWriteOnce"] storageClassName： ebs-
sc resources: requests: storage:20Gi # ディスク IOPS 向
上のために storageClassName を使用している間、これを 200Gi に
変更します - metadata: name: varcores spec: accessModes:
 [ "ReadWriteOnce" ] storageClassName: ebs-sc resources:
 requests: storage:2Gi - metadata: name: panplugincfg spec:
 accessModes: [ "ReadWriteOnce" ] storageClassName: ebs-sc
 resources: requests: storage:1Gi - metadata: name: panconfig
 spec: accessModes: [ "ReadWriteOnce" ] storageClassName:
 ebs-sc resources: requests: storage:8Gi - metadata:
 name: panplugins spec: accessModes: [ "ReadWriteOnce" ]
 storageClassName: ebs-sc resources: requests: storage:200Mi
```

**STEP 6 |** CN-MGMT StatefulSet をデプロイします。

デフォルトで、管理プレーンは耐障害性を提供する StatefulSet としてデプロイされます。1 つの CN-MGMT StatefulSet に最大 30 個のファイアウォール CN-NGFW ポッドを接続できます。

1. (静的にプロビジョニングされた PV のみに必要) CN-MGMT StatefulSet 用の永続ボリューム (PV) をデプロイします。

   1. pan-cn-pv-local.yaml で定義されたローカル ボリューム名と一致するディレクトリを作成します。

      少なくとも 2 つのワーカー ノード上に 6 つのディレクトリが必要です。CN-MGMT StatefulSet をデプロイする各ワーカー ノードにログインして、ディレクトリを作成

します。たとえば、/mnt/pan-local1 から /mnt/pan-local6 という名前のディレクトリを作成するには、次のコマンドを使用します。

```
mkdir -p /mnt/pan-local1 /mnt/pan-local2 /mnt/pan-local3 /
mnt/pan-local4 /mnt/pan-local5 /mnt/pan-local6
```

2. pan-cn-pv-local.yaml を変更します。

   nodeaffinity の下でホスト名を一致させ、上記で spec.local.path に作成したディレクトリが変更されたことを確認してから、そのファイルをデプロイして、新しいストレージ クラス pan-local-storage とローカル PV を作成します。

2. pan-cn-mgmt-configmap YAML ファイルと pan-cn-mgmt YAML ファイルが変更されたことを確認します。

   EKS から pan-cn-mgmt-configmap をサンプリングします。

```
 復元されたセッション コンテンツ apiVersion: v1 kind:ConfigMap
 metadata: name: pan-mgmt-config namespace: kube-system
 data:PAN_SERVICE_NAME: pan-mgmt-svc PAN_MGMT_SECRET: pan-
mgmt-secret # Panorama 設定 PAN_PANORAMA_IP: "x.y.z.a"
 PAN_DEVICE_GROUP: "dg-1" PAN_TEMPLATE_STACK: "temp-stack-1"
 PAN_CGNAME:"CG-EKS" # 意図されたライセンス バンドル タイプ - "CN-
X-BASIC", "CN-X-BND1", "CN-X-BND2" # Panorama K8S プラグインに適
用された認証コードに基づく" PAN_BUNDLE_TYPE:"CN-X-BND2"# 必須ではな
いパラメーター # Panorama Kubernetesプラグインで提供されるクラスター名
と同じ名前にすることをお勧めします-同じ Panorama #CLUSTER_NAME で複数
のクラスターを管理する場合、ポッドを簡単に識別できます。"Cluster-name"
 #PAN_PANORAMA_IP2: "passive-secondary-ip" # CERTを使用する場合
はコメントアウトします。それ以外の場合は、pan-mgmt の etcd への暗号化さ
れた接続をバイパスします。# EKS バグのために etcd に CERT を使用しない
 ETCD_CERT_BYPASS: "" # 値は必要ありません # CERT を使用するように
コメントアウトしてください。それ以外の場合は、pan-mgmtとpan-ngfw 間の
 IPSec に PSK を使用します。 # IPSEC_CERT_BYPASS: "" # 値は必要あり
ません
```

pan-cn-mgmt.yaml のサンプル

```
initContainers: - name: pan-mgmt-init image: <your-private-
registry-image-path>
```

```
 containers: - name: pan-mgmt image: <your-private-registry-
image-path> terminationMessagePolicy:FallbackToLogsOnError
```

3. Kubectl を使用して yaml ファイルを実行します。

   **kubectl apply -f pan-cn-mgmt-configmap.yaml**

   **kubectl apply -f pan-cn-mgmt-slot-crd.yaml**

   **kubectl apply -f pan-cn-mgmt-slot-cr.yaml**

   **kubectl apply -f pan-cn-mgmt-secret.yaml**

   **kubectl apply -f pan-cn-mgmt.yaml**

   pan-mgmt-serviceaccount.yaml は、クラスター認証用のサービスアカウントの作成を以前に完了していない場合にのみ実行する必要があります。

4. CN-MGMT ポッドが起動していることを確認します。

   これには、5 ～ 6 分かかります。

   **kubectl get pods -l app=pan-mgmt -n kube-system** を使用します。

   ```
   NAME READY STATUS RESTARTS AGEpan-mgmt-sts-0 1/1 Running 0
   27hpan-mgmt-sts-1 1/1 Running 0 27h
   ```

**STEP 7 |** CN-NGFW ポッドをデプロイします。

デフォルトで、ファイアウォール データプレーン CN-NGFW ポッドは DaemonSet としてデプロイされます。CN-NFGW ポッドのインスタンスは、1 つのノード上で最大 30 個のアプリケーション ポッドのトラフィックを保護することができます。

1. PAN-CN-NGFW-CONFIGMAP と PAN-CN-NGFW に詳述されているように YAML ファイルが変更されたことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

2. Kubectl apply を使用して pan-cn-ngfw-configmap.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap.yaml**

3. Kubectl apply を使用して pan-cn-ngfw.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw.yaml**

4. すべての CN-NGFW ポッド (クラスタ内のノードあたり 1 つずつ) が実行していることを確認します。

   これは、4 ノード オンプレミス クラスタからの出力例です。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

   ```
   NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

   pan-ngfw-ds-8g5xb 1/1 Running 0 27h 10.233.71.113 rk-k8-node-1 <none> <none>

   pan-ngfw-ds-qsrm6 1/1 Running 0 27h 10.233.115.189 rk-k8-vm-worker-1 <none> <none>

   pan-ngfw-ds-vqk7z 1/1 Running 0 27h 10.233.118.208 rk-k8-vm-worker-3 <none> <none>

   pan-ngfw-ds-zncqg 1/1 Running 0 27h 10.233.91.210 rk-k8-vm-worker-2 <none> <none>
   ```

**STEP 8 |** Kubernetes クラスタ上の CN-MGMT、CN-NGFW、および PAN-CNI が表示されていることを確認します。

**kubectl -n kube-system get pods**

```
pan-cni-5fhbg 1/1 Running 0 27hpan-cni-9j4rs 1/1 Running 0 27hpan-cni-ddwb4 1/1 Running 0 27hpan-cni-fwfrk 1/1 Running 0 27hpan-cni-h57lm 1/1 Running 0 27hpan-cni-j62rk 1/1 Running 0 27hpan-cni-lmxdz 1/1 Running 0 27hpan-mgmt-sts-0 1/1 Running 0 27hpan-mgmt-sts-1 1/1 Running 0 27hpan-ngfw-ds-8g5xb 1/1 Running 0 27hpan-ngfw-ds-qsrm6 1/1 Running 0 27hpan-ngfw-ds-vqk7z 1/1 Running 0 27hpan-ngfw-ds-zncqg 1/1 Running 0 27h
```

**STEP 9 |** 新しいポッドからのトラフィックがファイアウォールにリダイレクトされるようにアプリケーション yaml または名前空間に注釈を付けます。

検査のためにトラフィックを CN-NGFW にリダイレクトするには、以下のアノテーションを追加する必要があります：

```
annotations：paloaltonetworks.com/firewall：pan-fw
```

たとえば、「default」名前空間のすべての新しいポッドの場合：

```
kubectl annotate namespace default paloaltonetworks.com/
firewall=pan-fw
```

一部のプラットフォームでは、*CNI* プラグイン チェーン内で *pan-cni* がアクティブになっていない状態でアプリケーション ポッドが開始する可能性があります。このようなシナリオを回避するには、アプリケーションポッド *YAML* にここに示すようにボリュームを指定する必要があります。

```
volumes: - name: pan-cni-ready hostPath: path: /var/log/
pan-appinfo/pan-cni-ready type:ディレクトリ
```

**STEP 10 |** クラスタでアプリケーションをデプロイします。

# CN-Series ファイアウォール を Kubernetes CNF としてデプロイする

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

CN-Series をコンテナネットワーク機能 （CNF） としてKubernetes 環境にデプロイできるようになりました。

CN-Series-as-a-daemonsetおよびCN-Series-as-a-kubernetes-serviceデプロイメントモードは、自動化されたセキュリティデプロイメントを提供し、Kubernetesのオートスケーリング機能を活用します。ただし、これらのデプロイメントモードには差し込みオプションが制限されており、I/O アクセラレーションをサポートしていません。さらに、検査が必要で複数のネットワークインターフェイスを使用するアプリケーションポッドでは達成可能なスループットが制限されます。

CN-series-as-a-kubernetes-CNFをデプロイすると、クラウドプロバイダーのネイティブルーティング、vRouters、Top of Rack（TOR)スイッチなどの外部エンティティを介して Service Function Chaining（SFC)を使用するトラフィックでこれらの課題が解決されます。CN-series-as-a-kubernetes-CNFのデプロイメントモードは、アプリケーションポッドに影響を与えません。

以下の手順を実行して、CN-Series-as-a-kubernetes-CNFをデプロイします。

開始する前に、CN-Series YAML ファイルのバージョンが PAN-OS バージョンと互換性があることを確認します。

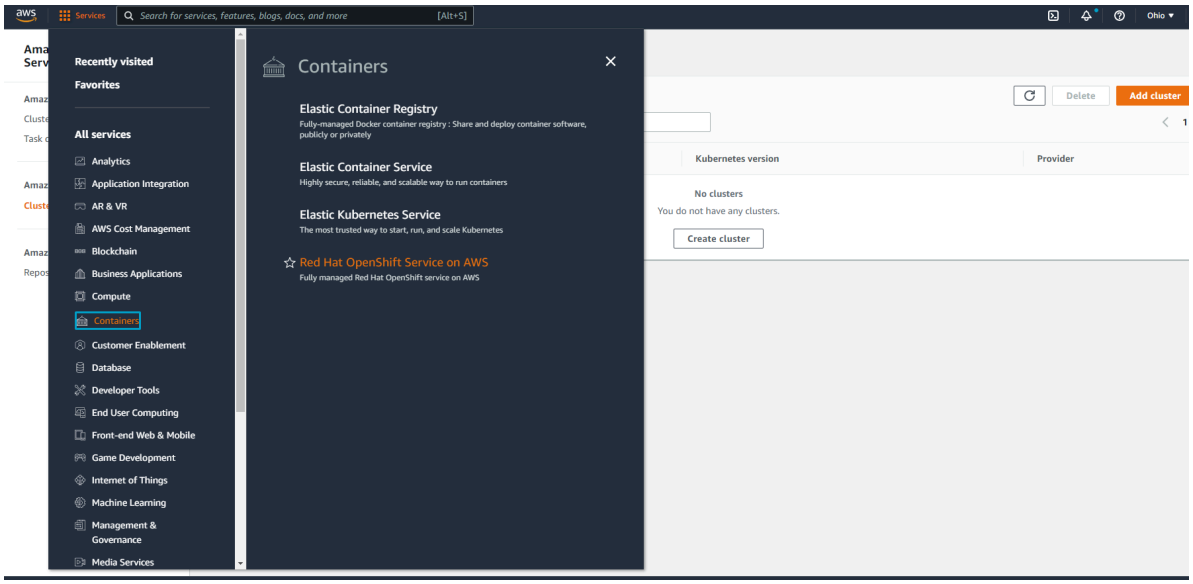
PAN-OS 10.2.0 もしくはそれ以降ではYAML 3.0.0 が必要です

**STEP 1 |** Kubernetes クラスタをセットアップします。詳細については、Amazon EKS クラスタの作成およびポッド用の複数のネットワークインターフェースを参照してください。

AWS EKS でクラスタを作成するには、次の手順を実行します。

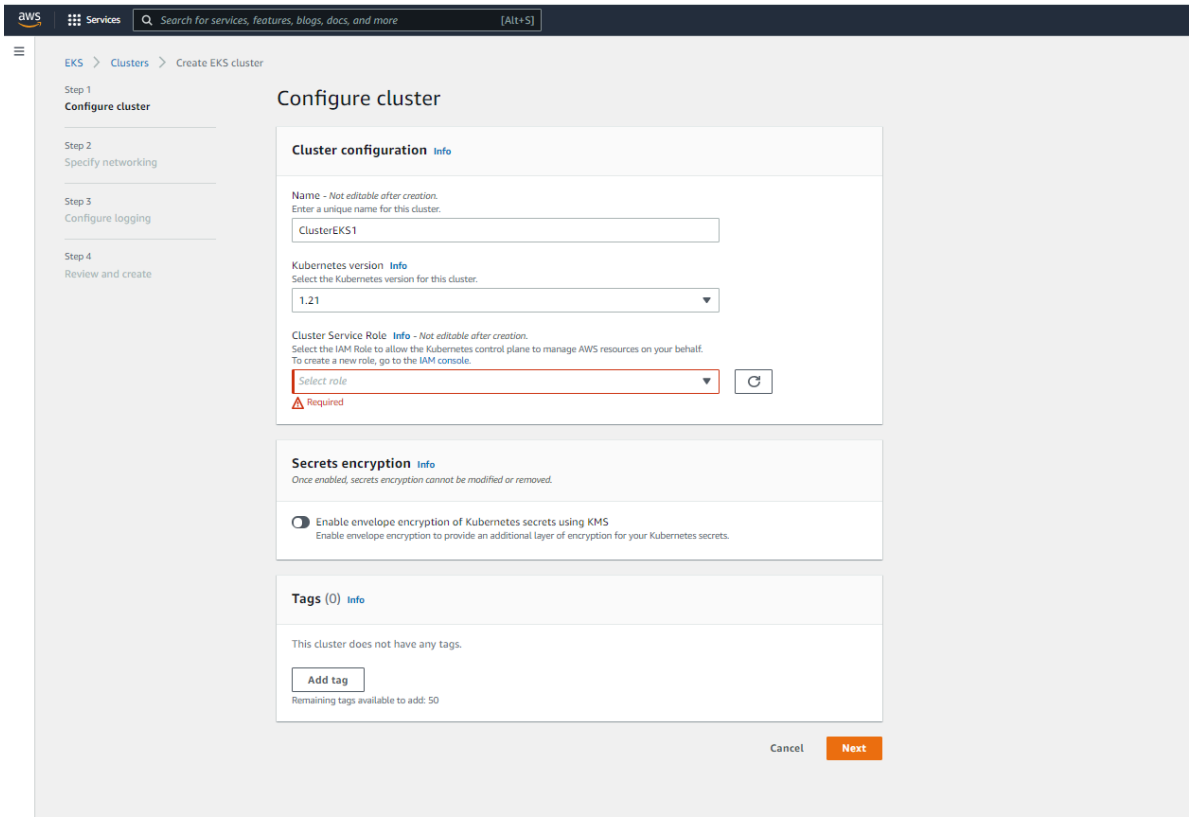
1. サービスナビゲーションメニューをクリックし、コンテナ->**Elastic Kubernetes Service**に移動します。



2. [クラスタの作成]をクリックします。
3. 必要な詳細を入力し、［作成］をクリックします。



1. クラスタのリソースが適切であることを確認します。クラスタにファイアウォールをサポートするためのCNシリーズの前提条件のリソースが含まれていることを確認します。

   **kubectl get nodes**

   **kubectl describe node <node-name>**

   コマンド出力の[容量]見出しの下にある情報を表示して、指定したノードで使用可能なCPU とメモリーを確認します。

   CPU、メモリ、およびディスク ストレージの割り当てはニーズによって異なります。CNシリーズのパフォーマンスとスケーリングを参照してください。

   以下の情報があることを確認してください。

   - Panorama 上で API サーバーをセットアップするためのエンドポイント IP アドレスを収集します。Panorama は、この IP アドレスを使用して、Kubernetes クラスタに接続します。
   - テンプレート スタック名、デバイス グループ名、Panorama IP アドレス、およびオプションで Panorama からログ コレクタ グループ名を収集します。
   - 認証コードと自動登録の PIN ID と値を収集します。
   - イメージをダウンロードしたコンテナ イメージ リポジトリの場所。

**STEP 2 |** （任意）Panorama の Kubernetes プラグインでカスタム証明書を設定した場合は、次のコマンドを実行して証明書シークレットを作成する必要があります。ファイル名を ca.crt から変更しないでください。pan-cn-mgmt-0.yaml、pan-cn-mgmt-1.yaml、pan-cn-ngfw-0.yaml、およびpan-cn-ngfw.yaml-1 のカスタム証明書のボリュームはオプションです。

```
kubectl -n kube-system create secret generic custom-ca --from-file =
ca.crt
```

**STEP 3 |** YAML ファイルを編集して、CN-Series ファイアウォールをデプロイするために必要な詳細を記入します。

プライベート レジストリへのパスを含め、必要なパラメータを指定するように YAML ファイル内のイメージパスを置き換える必要があります。詳細は CN-Series デプロイメント YAML ファイル内の編集可能なパラメータ を参照してください。

HA の CN-Series-as-a-kubernetes-CNF は、セッションと構成の同期を備えたアクティブ/パッシブ HA のみをサポートします。

CN-Series-as-a-kubernetes-CNF をHAにデプロイすると、アクティブノードとパッシブノードにそれぞれ 2 つの PAN-CN-MGMT-CONFIGMAP、PAN-CN-MGMT、およびPAN-CN-NGFWYAMLファイルが次のように作成されます。

- pan-cn-mgmt-0.yaml
- pan-cn-mgmt-1.yaml
- pan-cn-mgmt-configmap-0.yaml
- pan-cn-mgmt-configmap-1.yaml
- pan-cn-ngfw-configmap-0.yaml
- pan-cn-ngfw-configmap-1.yaml

次のデフォルト値は、pan-cn-mgmt-configmap-0.yaml ファイルと pan-cn-mgmt-configmap-1.yaml ファイルで定義されています。

pan-cn-mgmt-configmap-0.yaml：

```
metadata:
```

```
name: pan-mgmt-config
```

```
namespace: kube-system
```

```
data:
```

```
PAN_SERVICE_NAME: pan-mgmt-svc-0
```

```
PAN_MGMT_SECRET: pan-mgmt-secret
```

pan-cn-mgmt-configmap-1.yaml：

```
metadata:
```

```
name: pan-mgmt-config
```

```
namespace: kube-system
```

```
data:
```

```
PAN_SERVICE_NAME: pan-mgmt-svc-1
```

```
PAN_MGMT_SECRET: pan-mgmt-secret
```

CPU ピニング用のnumaオプションを追加できます。pan-cn-ngfw-configmap-0.yaml ファイルと pan-cn-ngfw-configmap-1.yaml ファイルに PAN_NUMA_ENABLED パラメータの単一の numa ノード番号を追加します。

CN-Series-as-a-kubernetes-CNFをHAにレイヤー3 をサポートありで正常にデプロイするために：

- HAでは、各 Kubernetes ノードに少なくとも 3つのインターフェースが必要です。管理（デフォルト）、HA2、およびデータインターフェイスです。
- L3モードのCN-Series ファイアウォールの場合、少なくとも 2 つのインターフェイスが必要です。管理 (デフォルト) とデータインターフェイス。
- 新しいネットワーク接続定義のYAML ファイルに次の変更を加えます。
  - ワーカーノードで、次のコマンドを実行して、ハイパーバイザーインターフェイスから **pciBusID** 値を取得します。

    ```
    lspci | grep -i ether
    ```

以下に例を示します。

```
00:05.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:06.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:07.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:08.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

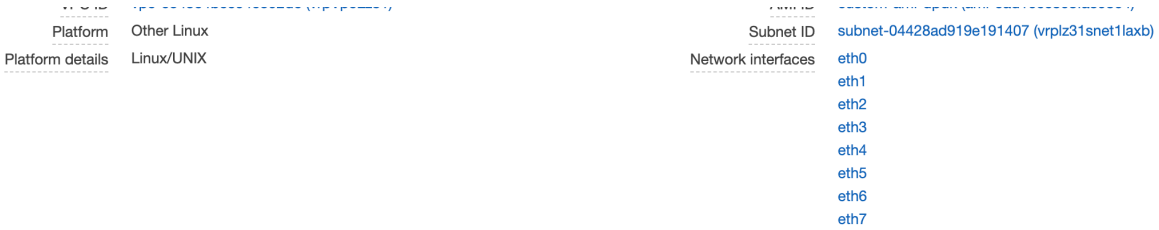
```
00:09.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0a.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0b.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0c.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

PCIの順序は、AWS EC2 UIに表示される eth インターフェイスの順序と同じです。




上記で取得した **pciBusID** 値を次のネットワーク定義ファイルに追加します。

```
net-attach-def-1.yaml
```

```
net-attach-def-2.yaml
```

```
net-attach-def-3.yaml
```

```
net-attach-def-ha2-0.yaml
```

```
net-attach-def-ha2-1.yaml
```

- AWS コンソール上の対応するノードインスタンスから HA2 インターフェースの静的 IP アドレスを取得し、`net-attach-def-ha2-0.yaml` および `net-attach-def-ha2-1.yaml` ファイルの アドレス パラメータに追加します。

**STEP 4 |** CN-MGMT StatefulSet をデプロイします。

デフォルトで、管理プレーンは耐障害性を提供する StatefulSet としてデプロイされます。CN-MGMT StatefulSet 単体に接続できるファイアウォール CN-NGFW ポッドは 1 つだけです。

1. (静的にプロビジョニングされた PV のみに必要) CN-MGMT StatefulSet 用の永続ボリューム (PV) をデプロイします。
   1. pan-cn-pv-local.yaml で定義されたローカル ボリューム名と一致するディレクトリを作成します。

      少なくとも 2 つのワーカー ノード上に 6 つのディレクトリが必要です。CN-MGMT StatefulSet をデプロイする各ワーカー ノードにログインして、ディレクトリを作成します。たとえば、/mnt/pan-local1 から /mnt/pan-local6 という名前のディレクトリを作成するには、次のコマンドを使用します。

      ```
      mkdir -p /mnt/pan-local1 /mnt/pan-local2 /mnt/pan-local3 /
      mnt/pan-local4 /mnt/pan-local5 /mnt/pan-local6
      ```

   2. pan-cn-pv-local.yaml を変更します。

      `nodeaffinity`の下でホスト名を一致させ、`spec.local.path`で上で作成したディレクトリを変更したことを確認してから、そのファイルをデプロイして、新しいストレージ クラスpan-local-storageとローカル PV を作成します。
2. pan-cn-mgmt-configmap YAML ファイルと pan-cn-mgmt YAML ファイルが変更されたことを確認します。
3. Kubectl を使用して yaml ファイルを実行します。

   **`kubectl apply -f pan-cn-mgmt-configmap-0.yaml`**

   **`kubectl apply -f pan-cn-mgmt-configmap-1.yaml`**

   **`kubectl apply -f pan-cn-mgmt-secret.yaml`**

   **`kubectl apply -f pan-cn-mgmt-0.yaml`**

   **`kubectl apply -f pan-cn-mgmt-1.yaml`**

   pan-mgmt-serviceaccount.yamlは、クラスター認証用のサービスアカウントの作成を以前に完了していない場合にのみ実行する必要があります。
4. CN-MGMT ポッドが起動していることを確認します。

   これには、5 ～ 6 分かかります。

   **`kubectl get pods -l app=pan-mgmt -n kube-system`** を使用します。

   ```
   NAME READY STATUS RESTARTS AGEpan-mgmt-sts-0 1/1 Running 0
   27hpan-mgmt-sts-1 1/1 Running 0 27h
   ```

**STEP 5 |** CN-NGFW を k8s-CNF モードでデプロイします。

1. 手順3で説明したように YAML ファイルを変更したことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

*multus* デーモンセットがインストールされ、ネットワーク接続定義ファイルが作成されていることを確認する必要があります。*pan-cn-ngfw-configmap-0.yaml* および *pan-cn-ngfw-configmap-1.yaml* ファイルの*PAN_SERVICE_NAME* のパラメータ値は、*pan-cn-mgmt-0.yaml* および *pan-cn-mgmt-1.yaml* のサービス名パラメータ値と一致する必要があります。

*HA* をサポートするには、*DP* ポッドを別のワーカーノードにデプロイすることをお勧めします。これは、***yaml nodeSelector*** フィールドから、または*Pod*のアンチアフィニティをオンにすることで確認できます。

*HA* サポートを有効にするには、次の*YAML*ファイルで *PAN_HA_SUPPORT*パラメータ値が ***true*** であることを確認する必要があります。

```
pan-cn-mgmt-configmap-0.yaml
```

```
pan-cn-mgmt-configmap-1.yaml
```

*DP* ポッドのデータインターフェイスの場合、必要に応じて *CNI* とインターフェイスリソースを *DP YAML* ファイルに追加する必要があります。以下に例を示します。

```
k8s.v1.cni.cncf.io/networks: net-attach-1,net-attach-2,net-attach-3
```

*DPDK*サポートを有効にするには、*pan-cn-ngfw-configmap-0.yaml* および *pan-cn-ngfw-configmap-1.yaml* ファイルの*PAN_DATA_MODE* パラメータ値が***dpdk***であることを確認する必要があります。

また、*HUGEPAGE_MEMORY_REQUEST* パラメータ値は、*pan-cn-ngfw-0.yaml* および *pan-cn-ngfw-1.yaml* ファイルの *hugepage* メモリ要求と一致する必要があります。

詳細については、CN-Series ファイアウォール上で DPDK を設定するを参照してください。

2. Kubectl applyを使用して、pan-cn-ngfw-configmap-0.yaml と pan-cn-ngfw-configmap-1.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap-0.yaml**

   **kubectl apply -f pan-cn-ngfw-configmap-1.yaml**

3. Kubectl applyを使用して、pan-cn-ngfw-0.yamlとpan-cn-ngfw-1.yamlを実行します。

   **kubectl apply -f pan-cn-ngfw-0.yaml**

   **kubectl apply -f pan-cn-ngfw-1.yaml**

4. CN-NGFW ポッドがデプロイされたことを確認します。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

**STEP 6 |** CN-NGFW ポッドをデプロイします。以下を実行してください。

1. PAN-CN-NGFW-CONFIGMAP-0、PAN-CN-NGFW-CONFIGMAP-1、PAN-CN-NGFW-0、およびPAN-CN-NGFW-1 で説明されているように YAML ファイルを変更したことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

2. Kubectl apply を使用して pan-cn-ngfw-configmap.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap.yaml**

3. Kubectl apply を使用して pan-cn-ngfw.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw.yaml**

4. CN-NGFW ポッドがデプロイされたことを確認します。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

**STEP 7 |** Kubernetes クラスタで CN-MGMT と CN-NGFW が表示されることを確認します。以下のコマンドを実行します：

```
kubectl -n kube-system get pods
```

# Kubernetes CNF L3をスタンドアロンモードでデプロイする

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

CN-Series ファイアウォールは、Kubernetes環境でL3スタンドアロンモードのコンテナネットワーク機能（CNF）としてデプロイできます。

CN-Series は、vRouterを介したトラフィックをサポートするようになりました。静的ルートは、トラフィックをファイアウォールのデータプレーンインターフェイスにリダイレクトするように構成されています。逆方向の場合、トラフィックは、IPv4 IP アドレスを使用した L3 ポリシーベースルーティング（PBR） を使用して同じファイアウォールにリダイレクトされます。K8s 環境のインターフェースの IP アドレスは、通常、DHCP を使用して CNI を介してプログラムされます。

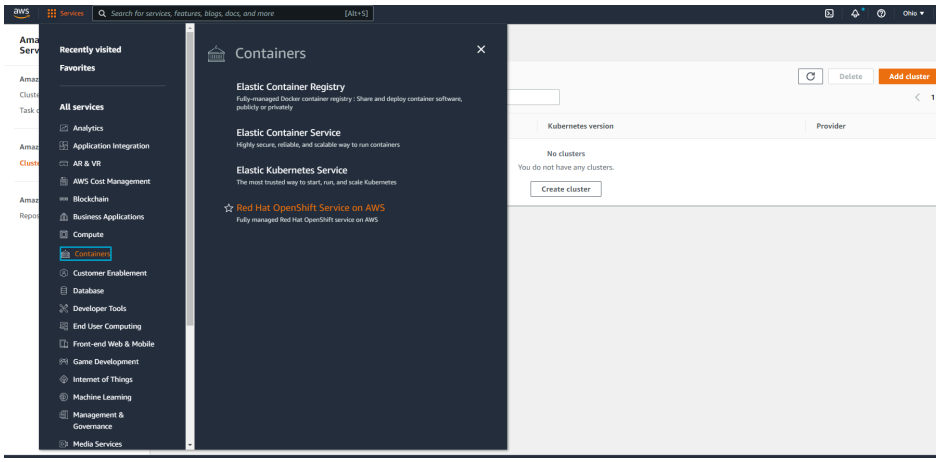
Kubernetes CNF を L3 スタンドアロンモードでデプロイするには：

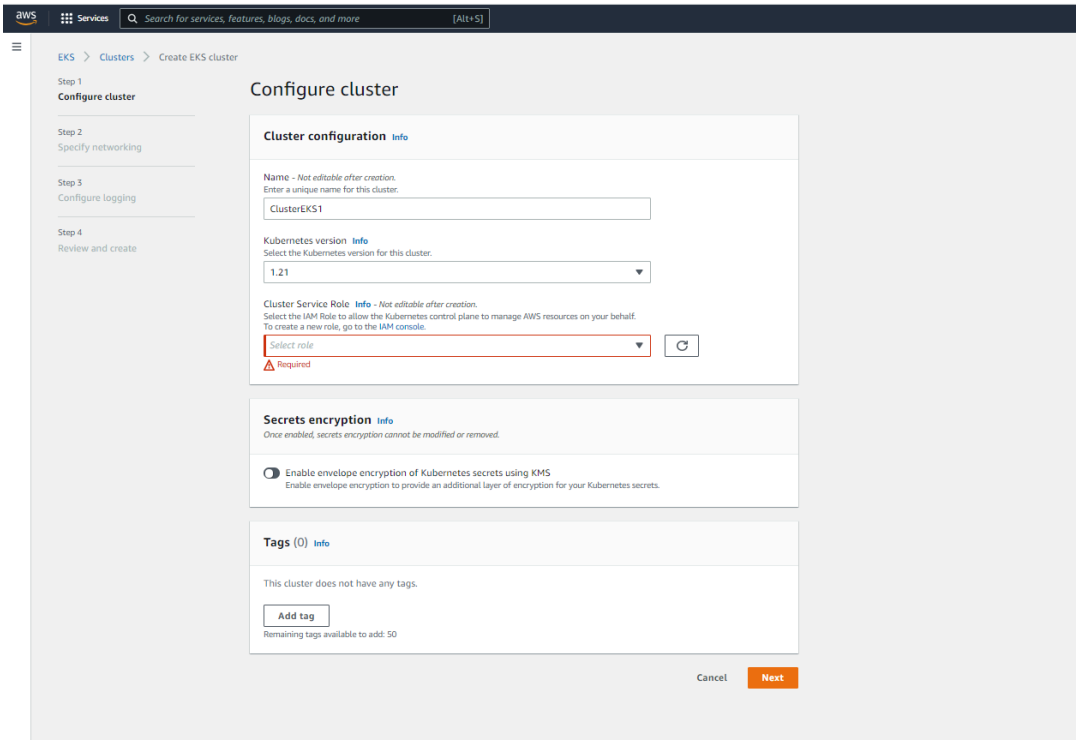
**STEP 1 |** Kubernetes クラスタをセットアップします。

AWS EKS でクラスターを作成するには、次の手順を実行します。

1. サービスナビゲーションメニューをクリックし、コンテナ->**Elastic Kubernetes Service**に移動します。




2. [クラスタの作成]をクリックします。
3. 必要な詳細を入力し、［作成］をクリックします。




1. クラスタのリソースが適切であることを確認します。クラスタにファイアウォールをサポートするためのCNシリーズの前提条件のリソースが含まれていることを確認します。

```
kubectl get nodes
kubectl describe node <node-name>
```

コマンド出力の[容量]見出しの下にある情報を表示して、指定したノードで使用可能な CPU とメモリーを確認します。

CPU、メモリ、およびディスク ストレージの割り当てはニーズによって異なります。CNシリーズのパフォーマンスとスケーリングを参照してください

以下の情報があることを確認してください。

- Panorama 上で API サーバーをセットアップするためのエンドポイント IP アドレスを収集します。Panorama は、この IP アドレスを使用して、Kubernetes クラスタに接続します。
- テンプレート スタック名、デバイス グループ名、Panorama IP アドレス、およびオプションで Panorama からログ コレクタ グループ名を収集します。
- 認証コードと自動登録の PIN ID と値を収集します。
- イメージをダウンロードしたコンテナ イメージ リポジトリの場所。

**STEP 2 |** 証明書シークレットを作成します。（オプション）Panorama の Kubernetes プラグインでカスタム証明書を設定した場合は、次のコマンドを実行して証明書シークレットを作成する必要があります。ファイル名を ca.crt から変更しないでください。pan-cn-mgmt-0.yaml および pan-cn-ngfw-0.yaml のカスタム証明書のボリュームはオプションです。

```
kubectl -n kube-system create secret generic custom-ca --from-file =
ca.crt
```

**STEP 3 |** YAML ファイルを編集して、CN-Series ファイアウォールをデプロイするために必要な詳細を記入します。

- pan-cn-mgmt-0.yaml
- pan-cn-mgmt-configmap-0.yaml
- pan-cn-ngfw-configmap-0.yaml

プライベート レジストリへのパスを含め、必要なパラメータを提供するように YAML ファイル内のイメージ パスを置き換える必要があります。詳細は CN-Series デプロイメント YAML ファイル内の編集可能なパラメータ を参照してください。

以下のデフォルト値が、pan-cn-mgmt-configmap-0.yaml ファイルで定義されています。

pan-cn-mgmt-configmap-0.yaml:

```
metadata:
```

```
name: pan-mgmt-config
```

```
namespace: kube-system
```

```
data:
```

```
PAN_SERVICE_NAME: pan-mgmt-svc-0
```

```
PAN_MGMT_SECRET: pan-mgmt-secret
```

CPU ピニング用の numa オプションを追加できます。pan-cn-ngfw-configmap-0.yamlファイルに PAN_NUMA_ENABLED パラメータの単一の numa ノード番号を追加します。

レイヤー 3 をサポートする CN-Series-as-a-kubernetes-CNF を正常にデプロイするには、次の手順に従います。

- 各 Kubernetes ノードには、少なくとも3つのインターフェースが必要です。管理（デフォルト）、HA2 リンク、およびデータインターフェイス。
- L3モードの CN-Series ファイアウォールの場合、少なくとも2つのインターフェイスが必要です。管理（デフォルト）、およびデータインターフェイス。
- 新しいネットワーク接続定義 YAML ファイルに次の変更を加えます。
  - ワーカーノードで、次のコマンドを実行して、ハイパーバイザーインターフェイスから **pciBusID** 値を取得します。

    ```
    lspci | grep -i ether
    ```

以下に例を示します。

```
00:05.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:06.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:07.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:08.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

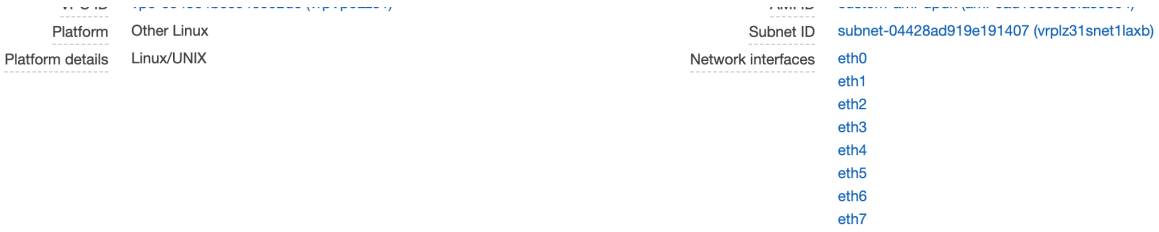
```
00:09.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0a.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0b.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

```
00:0c.0 Ethernet controller:Amazon.com, Inc.Elastic Network
 Adapter (ENA)
```

PCIの順序は、AWS EC2 UIに表示される eth インターフェイスの順序と同じです。




上記で取得した **pciBusID** 値を次のネットワーク定義ファイルに追加します。

```
net-attach-def-1.yaml
```

```
net-attach-def-2.yaml
```

```
net-attach-def-3.yaml
```

**STEP 4 |** CN-MGMT StatefulSet をデプロイします。

デフォルトで、管理プレーンは耐障害性を提供する StatefulSet としてデプロイされます。CN-MGMT StatefulSet 単体に接続できるファイアウォール CN-NGFW ポッドは 1 つだけです。

1. (静的にプロビジョニングされた PV のみに必要) CN-MGMT StatefulSet 用の永続ボリューム (PV) をデプロイします。
   1. pan-cn-pv-local.yaml で定義されたローカル ボリューム名と一致するディレクトリを作成します。

      少なくとも 2 つのワーカー ノード上に 6 つのディレクトリが必要です。CN-MGMT StatefulSet をデプロイする各ワーカー ノードにログインして、ディレクトリを作成します。たとえば、/mnt/pan-local1 から /mnt/pan-local6 という名前のディレクトリを作成するには、次のコマンドを使用します。

      ```
      mkdir -p /mnt/pan-local1 /mnt/pan-local2 /mnt/pan-local3 /
      mnt/pan-local4 /mnt/pan-local5 /mnt/pan-local6
      ```

   2. pan-cn-pv-local.yaml を変更します。

      `nodeaffinity` の下でホスト名を一致させ、上記で `spec.local.path` に作成したディレクトリが変更されたことを確認してから、そのファイルをデプロイして、新しいストレージ クラス pan-local-storage とローカル PV を作成します。
2. pan-cn-mgmt-configmap YAML ファイルと pan-cn-mgmt YAML ファイルが変更されたことを確認します。
3. Kubectl を使用して yaml ファイルを実行します。

   **`kubectl apply -f pan-cn-mgmt-secret.yaml`**

   **`kubectl apply -f pan-cn-mgmt-configmap-0.yaml`**

   **`kubectl apply -f $ dir / pan-cn-mgmt-0.yaml`**

   **`kubectl apply -f $ dir / net-attach-def-1.yaml`**

   **`kubectl apply -f $ dir / net-attach-def-2.yaml`**

   **`kubectl apply -f $ dir / pan-cn-mgmt-0.yaml`**

   **`kubectl apply -f $ dir / pan-cn-ngfw-configmap-0.yaml`**

   **`kubectl apply -f $ dir / pan-cn-ngfw-0.yaml`**

   pan-mgmt-serviceaccount.yamlは、クラスター認証用のサービスアカウントの作成を以前に完了していない場合にのみ実行する必要があります。
4. CN-MGMT ポッドが起動していることを確認します。

   これには、5 ～ 6 分かかります。

   **`kubectl get pods -l app=pan-mgmt -n kube-system`** を使用します。

   ```
   NAME READY STATUS RESTARTS AGEpan-mgmt-sts-0 1/1 Running 0
   27hpan-mgmt-sts-1 1/1 Running 0 27h
   ```

**STEP 5 |** CN-NGFW を k8s-CNF モードでデプロイします。

1. 手順3で説明したように YAML ファイルを変更したことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

   *multus* デーモンセットがインストールされ、ネットワーク接続定義ファイルが作成されていることを確認する必要があります。*pan-cn-ngfw-configmap-0.yaml*ファイルの *PAN_SERVICE_NAME* のパラメータ値は、*pan-cn-mgmt-0.yaml*ファイルの サービス名 パラメータ値と一致する必要があります。

   *CN-NFGW*ポッドのデータインターフェイスの場合、必要に応じて *CNI* とインターフェイスリソースを *CN-NFGW YAML* ファイルに追加する必要があります。以下に例を示します。

   *k8s.v1.cni.cncf.io/networks: <interface-cni1>@eth1,<interface-cni2>@eth2*

   *DPDK*サポートを有効にするには、*pan-cn-ngfw-configmap-0.yaml*ファイルの*PAN_DATA_MODE* パラメータ値が ***dpdk*** であることを確認する必要があります。

   また、*HUGEPAGE_MEMORY_REQUEST* パラメータ値は、*pan-cn-ngfw-0.yaml*ファイルの *hugepage* メモリ要求と一致する必要があります。

   詳細については、CN-Series ファイアウォール上で DPDK を設定するを参照してください。

2. Kubectl apply を使用して pan-cn-ngfw-configmap-0.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap-0.yaml**

3. Kubectl applyを使用して、pan-cn-ngfw-0.yamlとpan-cn-ngfw-1.yamlを実行します。

   **kubectl apply -f pan-cn-ngfw-0.yaml**

4. CN-NGFW ポッドがデプロイされたことを確認します。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

**STEP 6 |** CN-NGFW ポッドをデプロイします。以下を実行してください。

1. PAN-CN-NGFW-CONFIGMAP-0 と PAN-CN-NGFW-0 に詳述されているように YAML ファイルが変更されたことを確認します。

   **containers: - name: pan-ngfw-container image: <your-private-registry-image-path>**

2. Kubectl apply を使用して pan-cn-ngfw-configmap.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw-configmap.yaml**

3. Kubectl apply を使用して pan-cn-ngfw.yaml を実行します。

   **kubectl apply -f pan-cn-ngfw.yaml**

4. CN-NGFW ポッドがデプロイされたことを確認します。

   **kubectl get pods -n kube-system -l app=pan-ngfw -o wide**

**STEP 7 |** Kubernetes クラスタで CN-MGMT と CN-NGFW が表示されることを確認します。以下のコマンドを実行します：

```
kubectl -n kube-system get pods
```

```
root@master-1:~/CNv3-cnf/native# kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-694b4c9455-bxqbf   1/1     Running   4          246d
calico-node-fvr2c                          1/1     Running   23         246d
calico-node-js7v9                          1/1     Running   3          246d
calico-node-ssp9t                          1/1     Running   3          246d
coredns-dff8fc7d-87bnh                     1/1     Running   2          246d
coredns-dff8fc7d-l67mk                     1/1     Running   3          212d
dns-autoscaler-66498f5c5f-8kr4p            1/1     Running   2          246d
kube-apiserver-master-1                    1/1     Running   2          246d
kube-controller-manager-master-1           1/1     Running   2          246d
kube-multus-ds-5drrn                       1/1     Running   3          205d
kube-multus-ds-6vv4z                       1/1     Running   4          205d
kube-multus-ds-f6bhf                       1/1     Running   19         205d
kube-proxy-c4tth                           1/1     Running   2          246d
kube-proxy-fhtz9                           1/1     Running   2          246d
kube-proxy-gd5lj                           1/1     Running   21         246d
kube-scheduler-master-1                    1/1     Running   2          246d
kubernetes-dashboard-667c4c65f8-8wgtx      1/1     Running   4          246d
kubernetes-metrics-scraper-54fbb4d595-pp6qk   1/1     Running   2          246d
nginx-proxy-worker-1                       1/1     Running   27         246d
nginx-proxy-worker-2                       1/1     Running   2          246d
nodelocaldns-6nc4x                         1/1     Running   3          246d
nodelocaldns-d5s6g                         1/1     Running   4          246d
nodelocaldns-jcfst                         1/1     Running   29         246d
pan-mgmt-sts-0-0                           1/1     Running   0          16m
pan-ngfw-dep-0-5ff468684f-2fnv6            1/1     Running   0          4m6s
root@master-1:~/CNv3-cnf/native# kubectl exec -it pan-mgmt-sts-0-0 -n kube-system -- bash
[root@pan-mgmt-sts-0-0 /]# ipsec status
Security Associations (1 up, 0 connecting):
       to-mp[2]: ESTABLISHED 3 minutes ago, 10.233.73.23[CN=pan-mgmt-svc-0.kube-system.svc]...10.233.73.24[CN=pan-fw.kube-system.svc]
       to-mp{1}:  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: 20a5f62c_i abec4c31_o
       to-mp{1}:   0.0.0.0/0 === 169.254.202.2/32
[root@pan-mgmt-sts-0-0 /]# su admin

Warning: Your device is still configured with the default admin account credentials. Please change your password prior to deployment.
admin@pan-mgmt-sts-0-0> show jobs all

Enqueued            Dequeued            ID  PositionInQ                          Type                        Status Result Completed
------------------------------------------------------------------------------------------------------------------------------------
2022/02/25 10:41:22  10:41:30           5                                        Commit                       FIN    OK 10:42:16
2022/02/25 10:40:56  10:40:56           4                                      AutoCom                        FIN    OK 10:41:24
2022/02/25 10:32:47  10:32:47           3                                   CommitAll                         FIN    OK 10:33:24
2022/02/25 10:30:52  10:30:52           2                                      AutoCom                        FIN    OK 10:31:30

admin@pan-mgmt-sts-0-0> show panorama-status

Panorama Server 1 : 10.3.252.196
    Connected     : yes
    HA state      : Unknown
```

```
admin@pan-mgmt-sts-0-0> request plugins vm_series list-dp-pods

DP pods                          Licensed               License Type
-------                          ---------      ------------------------------------------------------------
pan-ngfw-dep-0-5ff468684f-2fnv6         yes             Threat Prevention, URL Filtering, Wildfire, DNS


admin@pan-mgmt-sts-0-0> debug show internal interface all

total configured hardware interfaces: 2

name                    id    speed/duplex/state            mac address
-------------------------------------------------------------------------------
ethernet1/1             16    10000/full/up                 00:0c:29:e7:ec:13
ethernet1/2             17    10000/full/up                 00:0c:29:e7:ec:3b

aggregation groups: 0


total configured logical interfaces: 2

name                id    vsys zone            forwarding              tag    address
------------------ ----- ---- --------------- ----------------------- ------ ------------------
ethernet1/1         16    1    trust           vr:vr1                  0      192.168.10.10/24
ethernet1/2         17    1    untrust         vr:vr1                  0      192.168.20.10/24
```

# CN-Series ファイアウォールのデプロイ

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

CNシリーズ ファイアウォールは、Kubernetesオーケストレーションを使用して簡単にデプロイでき、継続的な統合/継続的開発（CI/CD）プロセスへのネットワークセキュリティの統合を簡素化します。CNシリーズ ファイアウォールの継続的な管理は、Panorama™ネットワークセキュリティ管理で一元化されます。Palo Alto Networksのすべてのファイアウォールと同じ管理コンソールにより、ネットワークセキュリティチームは、組織全体のネットワークセキュリティ態勢を一元管理できます。

この章では、以下のセクションについて説明します。

# CN-Series デプロイメントチェックリスト

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

CN-Series ファイアウォールをデプロイするには、以下のタスクを完了する必要があります。

- まだ行っていない場合は、CNシリーズ ファイアウォールのライセンスを取得します。– CNシリーズ ファイアウォールをデプロイする準備ができたら認証コードを生成し、手元に置いておきます。
- CNシリーズの前提条件を確認–デプロイメントを開始する前に、CNシリーズ ファイアウォールのデプロイに必要なシステム要件を理解していることを確認してください。
- コンポーネントを準備します。
  - Panorama で VM 認証キーを生成します。
  - (任意)CNシリーズ ファイアウォールへのデバイス証明書のインストール
  - クラスタ認証用にサービス アカウントを作成する
  - Panorama のデプロイ-CN-Series ファイアウォールのデプロイメントを設定、デプロイ、および管理するには、Panorama を使用する必要があります。Panorama アプライアンスのデプロイとセットアップの詳細については、Panorama の設定を参照してください。
  - CNシリーズ用のKubernetesプラグインをインストールします。
  - CN-Series デプロイメント用にイメージとファイルを取得する-Palo Alto Networksリポジトリにアクセスして Dockerファイルをダウンロードし、GitHubにアクセスしてCNシリーズ ファイアウォールをKubernetes環境にデプロイするために必要なyamlファイルを取得します。
- CN-Series ファイアウォールをデプロイします。
  - デプロイに合わせて HELM チャートを編集する – または、CN-Series ファイアウォールをデプロイする前に、yaml ファイルを編集して、CN-Series デプロイメント YAML ファイル内の編集可能なパラメータをレビューすることもできます。CN-Series ファイアウォールを正常にデプロイするには、yaml ファイルに設定されているパラメーターの多くを変更する必要があります。
  - 「CNシリーズ ファイアウォールをKubernetesサービスとしてデプロイする(推奨されたデプロイメント モード)」を行います。
  - 「CN-Series ファイアウォールを DaemonSet としてデプロイする」を行います。

- （オプション）CN-Series ファイアウォールを Kubernetes サービスとしてデプロイする場合は、CN-Series で水平ポッド自動スケーリングを有効にする ができます。水平ポッド自動スケーリング（HPA）を使用すると、CN-Series ファイアウォールのデプロイメントを Kubernetes 環境に合わせて動的に自動スケーリングできます。
- CNシリーズをOpenShift環境に展開する場合は、OpenshiftでCNシリーズ ファイアウォールをデプロイするを参照してください。
- CN-Series ファイアウォールで 5G トラフィックを保護している場合は、CN-Series ファイアウォールを使用して 5G をセキュリティで保護する を参照してください。

❑ Panorama を設定して Kubernetes デプロイメントをセキュリティで保護する- CN-Series ファイアウォールをデプロイした後、Panorama を使用して、トラフィックの強制を有効にするセキュリティポリシーを設定し、それらのポリシーをファイアウォールにプッシュします。

# Helmチャートを使用した場合(推奨)と使用しない場合のCNシリーズ ファイアウォールをデプロイします

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

Helm リポジトリには、Kubernetes 用 Helm パケットマネージャーを使用して Palo Alto Networks CN-Series コンテナー化ファイアウォールをデプロイするためのチャートとテンプレートが含まれています。

CNシリーズHelmチャートはGitHubからダウンロードできます。

- Helm チャートとテンプレートを使用する準備
- HELMチャートを使用してCNシリーズ ファイアウォールをデプロイする(推奨)
- YAMLファイルによるCNシリーズファイアウォールをデプロイする

## Helm チャートとテンプレートを使用する準備

必要なソフトウェアをインストールします。これらの手順には最小バージョンが記載されていますが、上限が指定されていない限り、同じファミリにそれ以降のバージョンをインストールできます。

**STEP 1 |** CNシリーズ ファイアウォール10.1.x、10.2.x、11.0.x、または 11.1.xコンテナイメージをデプロイします。

**STEP 2 |** 1.16～1.25のKubernetesバージョンをインストールし、Kubernetesクラスタを作成します。環境でサポートされているKubernetesのバージョンの詳細については、CNシリーズ デプロイメント サポート環境を参照してください。

**STEP 3 |** Panorama は、Kubernetes クラスタおよびクラスタのセキュリティ保護に使用する CN-Series ファイアウォールからアクセスできる場所にデプロイします。

1. Panorama PAN-OS のバージョンが 10.x.x 以降であることを確認します。
2. Panorama バージョン 1.0.x または 2.0.x 用の Kubernetes プラグインをインストールします。

**STEP 4 |** Helm クライアントバージョン 3.6.0 以降をインストールします。

HELMチャートを使用してCNシリーズ ファイアウォールをデプロイする(推奨)または YAMLファイルによるCNシリーズファイアウォールをデプロイするに進みます。

# HELMチャートを使用してCNシリーズ ファイアウォールをデプロイする(推奨)

この手順を使用して、リポジトリのクローンを作成し、ローカル環境からデプロイします。

**STEP 1 |** Panorama で VM 認証キーを生成します。

**STEP 2 |** GitHub からリポジトリをクローン作成します。

```
$ git clone https://github.com/PaloAltoNetworks/cn-series-helm.git
```

**STEP 3 |** クローン作成されたリポジトリのローカルディレクトリに変更します。以下に例を示します。

```
$ cd cn-series-helm
```

**STEP 4 |** デプロイメント用サブディレクトリに変更します。

- ディレクトリhelm_cnv1を使用して、CNシリーズをデーモン セットとしてデプロイします
- ディレクトリhelm_cnv2を使用して、CNシリーズをサービスとしてデプロイします。
- ディレクトリhelm_cnv3 を使用して、CNシリーズをcnfとしてデプロイします。

**STEP 5 |** plugin-serviceaccount.yamlのサービスアカウントYAMLをダウンロードし、yamlを適用します。サービス アカウントは、Panorama が Kubernetes ラベルとリソース情報を取得するためにクラスターに対して認証するために必要な権限を有効にします。このサービス アカウントには、デフォルトでpan-plugin-userという名前が付けられています。以下のコマンドを実行して、plugin-serviceaccount.yamlファイルをデプロイします。

```
kubectl apply -f plugin-serviceaccount.yaml
```

```
kubectl -n kube-system get secrets | grep pan-plugin-user
```

このサービス アカウントに関連付けられたシークレットを表示するには、以下の手順を実行します。

```
kubectl -n kube-system get secrets <secrets-from-above-command> -o
json >> cred.json
```

シークレットを含む認証情報ファイル(この例ではcred.jsonという名前が付けられている)を作成し、保存します。このファイルをPanoramaにアップロードして、 CNシリーズ ファ

イアウォールのためのKubernetesプラグインをインストールするでクラスターを監視するためのKubernetesプラグインをセットアップする必要があります。

> ***Openshift***では、*Helm*チャートをデプロイする前に、各*Openshift*ネームスペースファイルの*pan-cni-net-attach-def.yaml*を手動でデプロイする必要があります。

**STEP 6 |** values.yaml ファイルを編集して、設定情報を入力します。以下の値はhelm_cnv1サブディレクトリの値です。

```
# The K8s environment # 有効なdeployToタグ: [gke|eks|aks||native]
 # 有効なmultusタグ: [enable|disable] オープンシフトやネイティブ デプロイメントのためにmultusは有効にしておきます。クラスター: deployTo: eks multus:
 disable
```

```
# Panorama タグ panorama: ip: "<Panorama-IP>" ip2: authKey:
 "<Panorama-auth-key>" deviceGroup: "<Panorama-device-group>"
 template: "<panorama-template-stack>" cgName: "<panorama-
collector-group>"
```

```
# MPコンテナタグ mp: initImage: gcr.io/pan-cn-series/pan_cn_mgmt_init
 initVersion: 最新の画像: gcr.io/pan-cn-series/panos_cn_mgmt
 version:10.2.3 cpuLimit:4 # DPコンテナタグ dp: image: gcr.io/pan-cn-
series/panos_cn_ngfw version:10.2.3 cpuLimit:2 # CNIコンテナタグcni:
 image: gcr.io/pan-cn-series/pan_cni version: latest
```

**STEP 7 |** レンダリングされた YAML ファイルを表示します。

```
helm install --debug --generate-name helm_cnv1/ --dry-run
```

**STEP 8 |** ヘルムチャートでリント チェックを行います。

```
helm lint helm_cnv1/
```

**STEP 9 |** HELM チャートをデプロイします。

```
helm install <deployment-name> helm_cnv1
```

> *HELM*チャートをアンインストールしても、永続ボリューム クレームは削除されません。*HELM*のインストールが機能するためには、これらのクレームを事前にクリアする必要があります。

HELMの詳細については、HELM Classicを参照してください:Kubernetesパッケージ マネージャー。

# YAMLファイルによるCNシリーズファイアウォールをデプロイする

リポジトリをクローン作成せずにデプロイするには、リポジトリを Helm クライアントに追加します。

**STEP 1 |** Panorama で VM 認証キーを生成します。

**STEP 2 |** `plugin-serviceaccount.yaml`のサービスアカウントYAMLをダウンロードし、yamlを適用します。サービス アカウントは、Panorama が Kubernetes ラベルとリソース情報を取得するためにクラスターに対して認証するために必要な権限を有効にします。このサービス アカウントには、デフォルトで`pan-plugin-user`という名前が付けられています。次のコマンドを実行して、`plugin-serviceaccount.yaml`ファイルをデプロイします。

```
kubectl apply -f plugin-serviceaccount.yaml
```

```
kubectl -n kube-system get secrets | grep pan-plugin-user
```

このサービス アカウントに関連付けられたシークレットを表示するには、以下の手順を実行します。

```
kubectl -n kube-system get secrets <secrets-from-above-command> -o
json >> cred.json
```

シークレットを含む認証情報ファイル(この例では`cred.json`という名前が付けられている)を作成し、保存します。このファイルをPanoramaにアップロードして、 CNシリーズ ファイアウォールのためのKubernetesプラグインをインストールするでクラスターを監視するためのKubernetesプラグインを設定する必要があります。

> ***Openshift***では、*Helm*チャートをデプロイする前に、各*Openshift*ネームスペースファイルの*`pan-cni-net-attach-def.yaml`*を手動でデプロイする必要があります。

**STEP 3 |** CN-Series のリポジトリをローカルの Helm クライアントに追加します。

次のコマンドを 1 行で入力します。

```
$ helm repo add my-project https: / /paloaltonetworks.github.io/cn-
series-helm
```

```
「cn-series」がリポジトリに追加されました
```

**STEP 4 |** リポジトリが Helm クライアントに追加されたことを確認します。

```
$ helm search repo cn-series
```

**STEP 5 |** Kubernetes クラスタを選択します。

```
$ kubectl config set-cluster NAME
```

**STEP 6 |** Helm チャートリポジトリを使用してデプロイします。以下のコマンドを編集して、設定情報を含めます。

```
$ helm install cn-series/cn-series --name="deployment name"
--set cluster.deployTo="gke|eks|aks|openshift"
- set panorama.ip="panorama hostname or ip"
- set panorama.ip2="panorama2 hostname or ip"
- set-string panorama.authKey="vm auth key"
--set panorama.deviceGroup="device group"
--set panorama.template="template stack"
--set panorama.cgName="collector group"
--set cni.image="container repo"
--set cni.version="container version"
--set mp.initImage="container repo"
--set mp.initVersion="container version"
--set mp.image="container repo"
--set mp.version="container version"
--set mp.cpuLimit="cpu max"
--set dp.image="container repo"
--set dp.version="container version"
--set dp.cpuLimit="cpu max"
```

*HELM*チャートをアンインストールしても、永続ボリューム クレームは削除されません。*HELM*のインストールが機能するためには、これらのクレームを事前にクリアする必要があります。

# Terraform テンプレートを使用した CN-Series ファイアウォールのデプロイ

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Terraform 0.13.0以上 |

CN-Series のデプロイメント リポジトリには、GKE、EKS、またはAKSクラスタをデプロイするための Terraform プランが含まれています。これらのプランにより、クラスターノードのサイズ設定とコンテナーネットワークインターフェース（CNI）がクラスター内での CN-Series ファイアウォールのデプロイメントを確実にサポートします。リポジトリには、CN-Series ファイアウォールのデプロイメントプランと、ファイアウォールで保護できるサンプルのPHPゲストブックアプリケーションも用意されています。

このオプションの手順には、以下のオプションのワークフローがあります。

- Helm チャートとテンプレートを使用する準備
- サンプルアプリケーションのデプロイ
- Terraform を使用した CN-Series ファイアウォールのデプロイ
- Panorama 用の Kubernetes プラグインの設定

## サンプルアプリケーションのデプロイ

Palo Alto Networks の GitHub リポジトリには、`guestbook.yml` という名前の Kubernetes マニフェストファイルを持つ、コミュニティがサポートするサンプルアプリケーションが含まれています。

このファイルは、Redis バックエンドを利用する単純な PHP ゲストブックウェブアプリケーションをデプロイします。

**STEP 1 |** Palo Alto Networks の GitHub リポジトリの cn-series-deploy ディレクトリで、sample-application ディレクトリに変更します。

```
$ cd sample-application
```

**STEP 2 |** ゲストブック アプリケーションをデプロイします。

```
$ kubectl apply -f guestbook.yml
```

**STEP 3 |** アプリケーション ポッドがデプロイされ、[実行中] から [準備完了] 状態になったことを確認します。

```
$ kubectl get pods -n sample-app
```

```
NAME READY STATUS RESTARTS AGE frontend-69859f6796-96bs7
 1/1 Running 0 111m frontend-69859f6796-k2k4z 1/1 Running
 0 53m frontend-69859f6796-zwwbg 1/1 Running 0 111m redis-
master-596696dd4-5l5qv 1/1 Running 0 53m redis-slave-6bb9896d48-
dwhw2 1/1 Running 0 53m redis-slave-6bb9896d48-nhqzh 1/1 Running 0
 111m
```

**STEP 4 |** サービスを一覧表示して、Web フロントエンドのパブリック IP アドレスを決定します。

```
$ kubectl get services -n sample-app
```

Panorama でダイナミック アドレス グループとセキュリティルールを設定して、ゲストブックアプリケーションを保護できるようになりました。

引き続きTerraformを使用してCNシリーズ ファイアウォールをデプロイします。

## Terraform を使用した CN-Series ファイアウォールのデプロイ

Terraform を使用して CN-Series のファイアウォールをデプロイします。

**STEP 1 |** ローカルの `cn-series\tfvars` を使用して `terraform.tfvars` という名前のファイルを作成し、以下の変数とそれに関連する値を追加します。

```
k8s_environment = ""           # Kubernetes 環境
                                # (gke|eks|aks|openshift|
native) panorama_ip = ""              # Panorama IP アドレス
 panorama_auth_key = ""         # Panorama 認証キー VM シリーズ登
録  panorama_device_group = ""    # Panorama デバイス グループ
 panorama_template_stack = ""  # Panorama テンプレート スタック
 panorama_collector_group = "" # Panorama ログ コレクタ グループ
 k8s_dp_cpu = ""               # DP コンテナ CPU 制限
```

**STEP 2 |** Terraform プランを検証します。

```
$ terraform init
```

**STEP 3 |** Terraform プランを検証します。

```
$ terraform plan
```

**STEP 4 |** Terraform プランを適用します。

```
$ terraform apply
```

**STEP 5 |** ポッドがデプロイされて [準備完了] となり、ステータスが [実行中] になっていることを確認します。

```
$ kubectl get pods -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE ... kube-system pan-
cni-6kkxw 1/1 Running 0 26m kube-system pan-cni-tvx2b 1/1 Running
 0 26m kube-system pan-mgmt-sts-0 1/1 Running 0 26m kube-system
 pan-mgmt-sts-1 1/1 Running 0 26m kube-system pan-ngfw-ds-nrtrn 1/1
 Running 0 26m kube-system pan-ngfw-ds-rcmmj 1/1 Running 0 26m
```

パノラマ用のKubernetesプラグインを設定する準備ができました。

# Panorama 用の Kubernetes プラグインの設定

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している |

Panorama用のKubernetesプラグインを使用して、Panorama デバイス グループにラベルを伝播します。

Kubernetes プラグインを使用して、Panorama と Kubernetes API の統合を完了できます。プラグインは新しいラベルを学習し、それらを Panorama デバイス グループに伝播します。これらのラベルには、Kubernetesラベル、サービス、名前空間、およびDynamic Address Group(ダイナミック アドレス グループ)の一致条件を定義できるその他のメタデータを含めることができます。

クラスタの資格情報ファイルのサイズが*32KB*を超えると、*Panorama Kubernetes*プラグイン上で資格情報ファイルをインポートする際にエラー メッセージが表示されます。エラー メッセージには、エラーの原因としてファイルのサイズが表示されます。

クラスタが*ca.crt*バンドルに多くの*CA*証明書を保有している

いる場合、*Kubernetes*プラグインは最上位の*CA*証明書のみを必要とします。最上位の*CA*証明書だけを保持し、他のすべての*CA*証明書と*service.crt*をクレデンシャル ファイルから削除する必要があります。その後、この更新された認証情報ファイルを使用できます。

この手順は、Helm チャートとテンプレートを使用する準備にリストされているサポートソフトウェアがインストールされていることを前提としています。

**STEP 1 |** Kubernetes マスターから pan-plugin-user サービスアカウントのクレデンシャルを取得します。

各コマンドを1行で入力します。

```
$ MY_TOKEN=`kubectl get serviceaccounts pan-plugin-user -n kube-system
  -o jsonpath='{.secrets[0].name}'`
$ kubectl get secret $MY_TOKEN -n kube-system -o json >
  ~/Downloads/pan-plugin-user.json
```

**STEP 2 |** Panorama Kubernetes プラグインでクラスター定義を作成します。

Terraform 出力に表示される Kubernetes マスターアドレスと`~/Downloads/pan-plugin-user.json` にある JSON クレデンシャルファイルを使用します。

Kubernetes API からインポートするラベルを定義します。

**STEP 3 |** Panorama Kubernetes プラグインで通知グループ定義を作成します。

この定義は、Kubernetes API から学習したラベルを Panorama デバイス グループに伝播するために使用されます。

以下の手順を実行し、Panorama Kubernetesプラグインで通知グループを作成します。

1. **Panorama**>プラグイン>**Kubernetes**>セットアップ>通知グループおよび追加を選択します。



2. 通知グループの名前を最大 31 文字で入力します。
3. クラスタ用に作成された外部タグ(デフォルト)に加えて内部タグを共有する場合は、**Enable sharing internal tags with Device Groups(**デバイス グループとの内部タグの共有を可能にする**)**を選択します。
4. タグを登録するデバイス グループを選択します。

5. **OK**をクリックします。

**STEP 4 |** Panorama プラグインでモニタリング定義を作成します。

前の手順で作成したクラスターと通知グループの定義を使用します。

**STEP 5 |** Panorama にコミットします。

**STEP 6 |** API 接続と MP コンテナの登録を確認するには、モニタリング定義に移動し、[詳細ステータス]と[クラスタ MP]をクリックします。

これで、アプリケーションをデプロイし、CN-Series ファイアウォールで保護する準備が整いました。

# Rancher オーケストレーションを使用した CN-Series ファイアウォールのデプロイ

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している |

これで、Rancher オーケストレーションと PAN OS 10.1 を使用して、CN-Series ファイアウォールをKubernetesサービスとしてデプロイできます。Rancher は、CN-Series ファイアウォールのデプロイに使用できるオープンソースのコンテナオーケストレーションプラットフォームです。

Rancher クラスターをサポートする CN-Series ファイアウォールのデプロイメントでは、Panorama インスタンスに 16 個の vCPU、32G メモリおよび追加の 2TB ディスク が必要です。Panorama は、CN-Series ファイアウォール デプロイメントからのログの収集を容易にするモードでデプロイされます。

オンプレミスの Rancher Kubernetes クラスター内に CN-Series ファイアウォールをデプロイする場合は、次の手順を実行します。

- CNシリーズ ファイアウォールを使用してKubernetesクラスタをセキュリティで保護するために必要なコンポーネントが利用可能であることを確認します。
- Kubernetes クラスターが最小システム要件を満たしていることを確認します。詳細については、CNシリーズ システム要件を参照してください。
- Rancher オーケストレーションを使用した CN-Series ファイアウォールのデプロイを実行します。
- 
- Rancher クラスターオプションの YAML ファイルを変更する
- CNシリーズ ファイアウォール用のKubernetesプラグインをインストールします。
- CNシリーズ ファイアウォールのライセンス取得。
- Rancher上のCNシリーズ ファイアウォールをKubernetesサービスとしてデプロイする(推奨されたデプロイメント モード)

## Rancher クラスターのデプロイ

次の2つの手順で Rancher をデプロイできます。

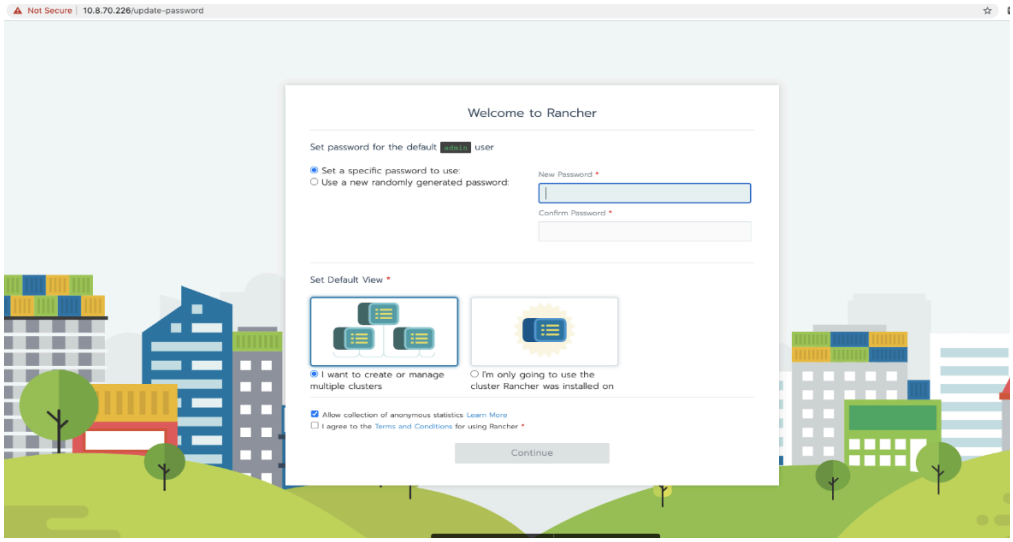
1. サポートされている Linux ディストリビューションと4GBのメモリを備えた Linux ホストを準備します。サポートされているバージョンの Docker をホストにインストールします。

2. サーバーを起動します。

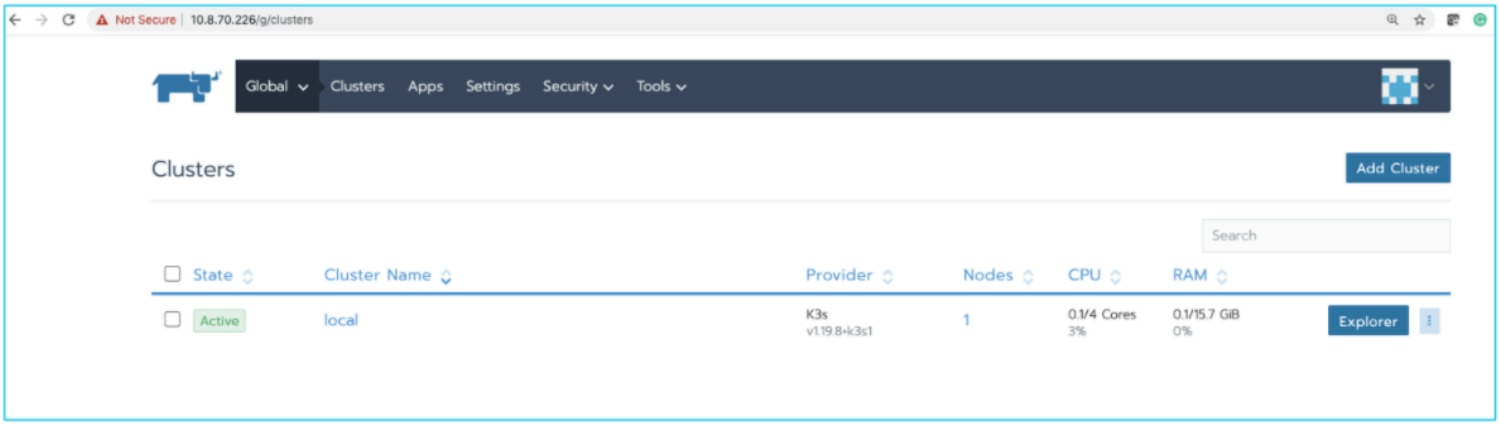
   Rancher をインストールして実行するには、ホストで以下のDocker コマンドを実行します。

```
$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80
 -p 443:443 rancher/rancher
```

デプロイメントが成功すると、Rancher サーバーの UI にアクセスして、管理者ユーザーのパスワードを設定できます。Rancher サーバーの UI にアクセスするには、ブラウザを開き、コンテナーがインストールされたホスト名またはアドレスに移動します。最初のクラスターのセットアップについて説明があります。



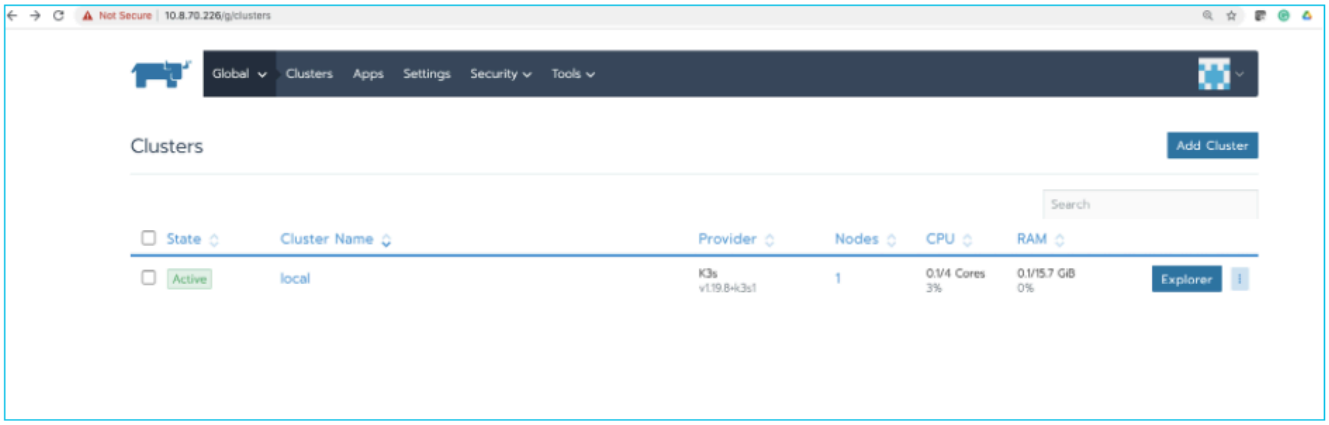

管理者ユーザーの作成に続いて、以下に示すようにローカルクラスターが作成されます。



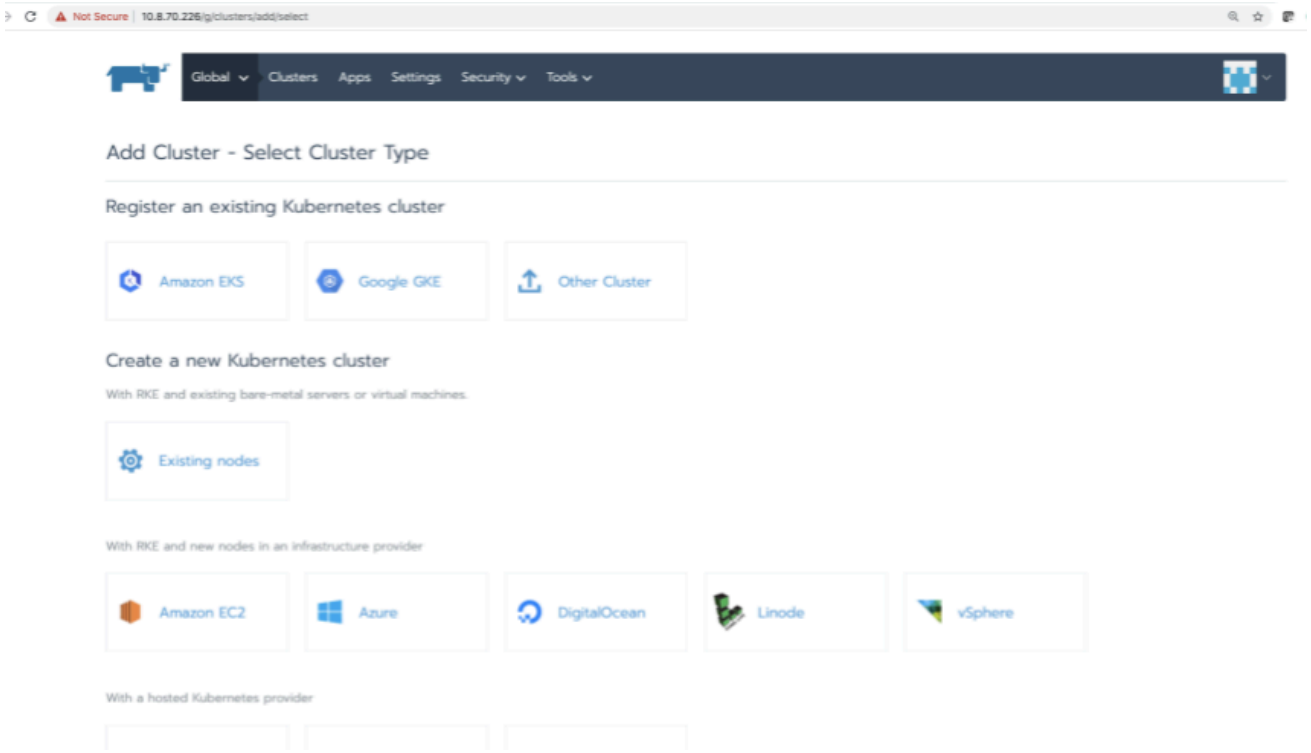

## Rancher クラスターにマスターノードとワーカーノードをセットアップする

Rancher UI でローカルクラスターを作成した後、マスターノードとワーカーノードをセットアップし、以下の手順を実行します。

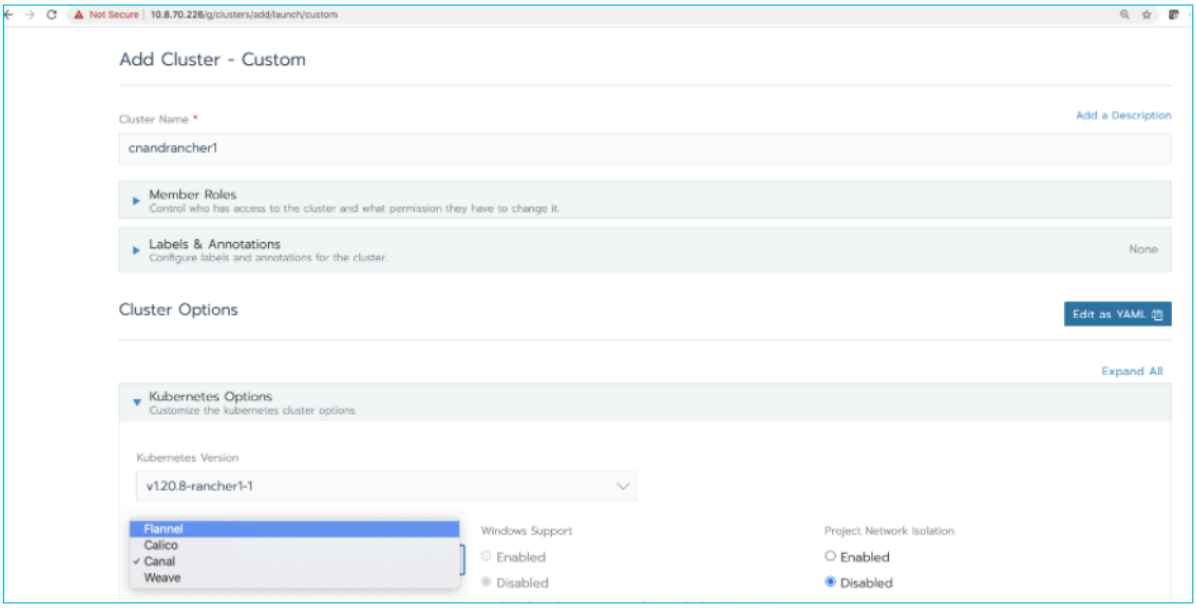
1. Rancher UI に移動し、[クラスターの追加]をクリックします。



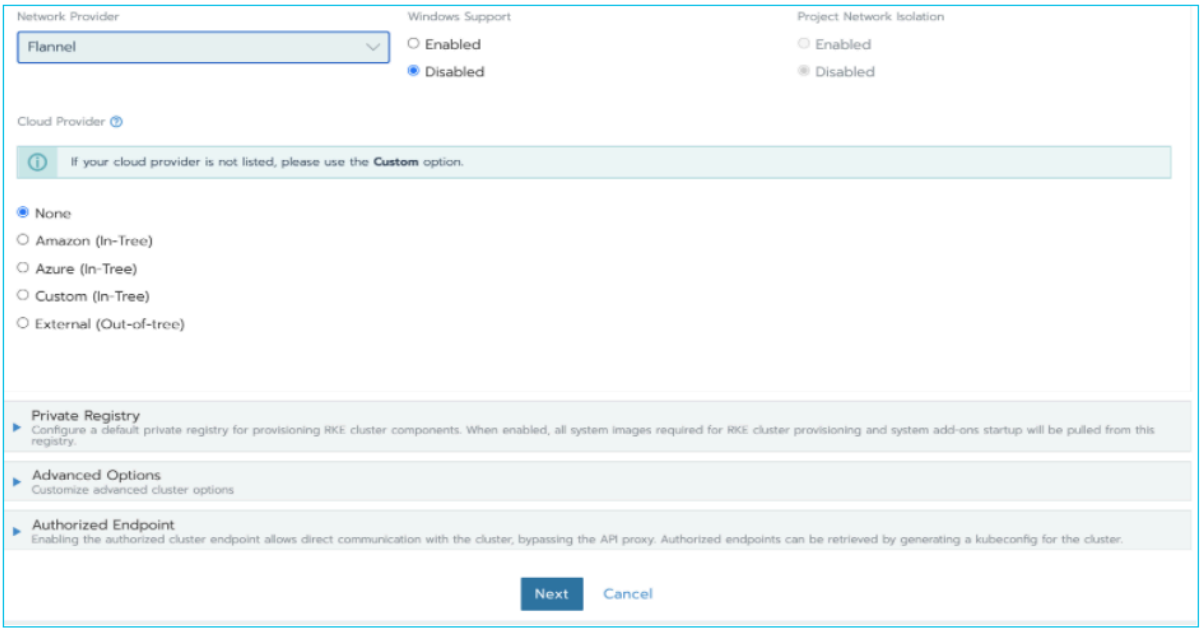

2. [既存のノード]をクリックします。



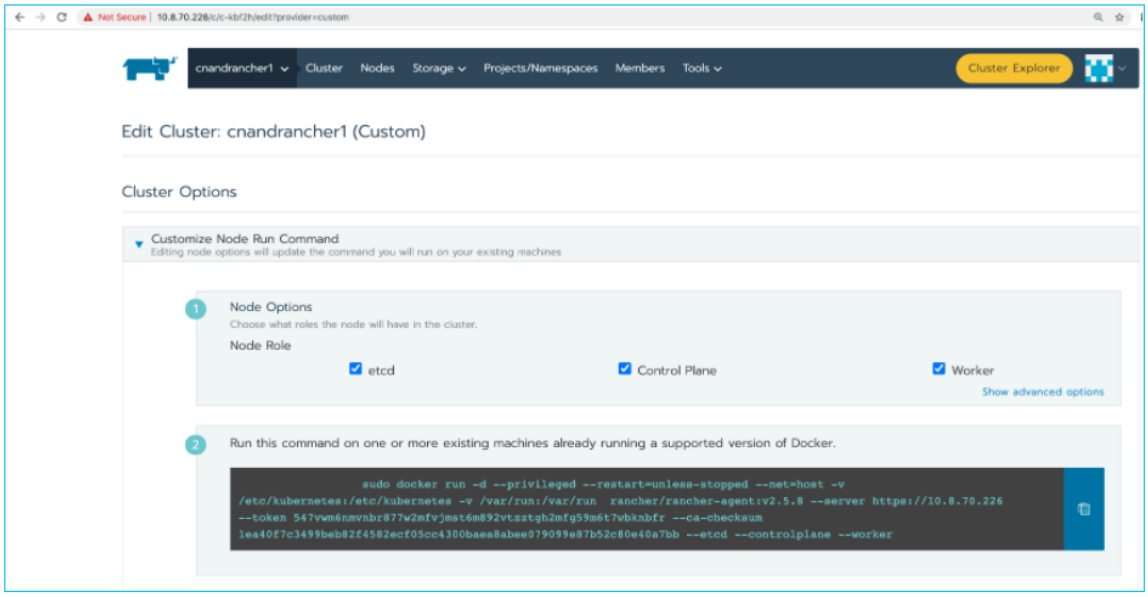
3. クラスター名を入力し、[ネットワークプロバイダー]ドロップダウンから[フランネル]を選択します。



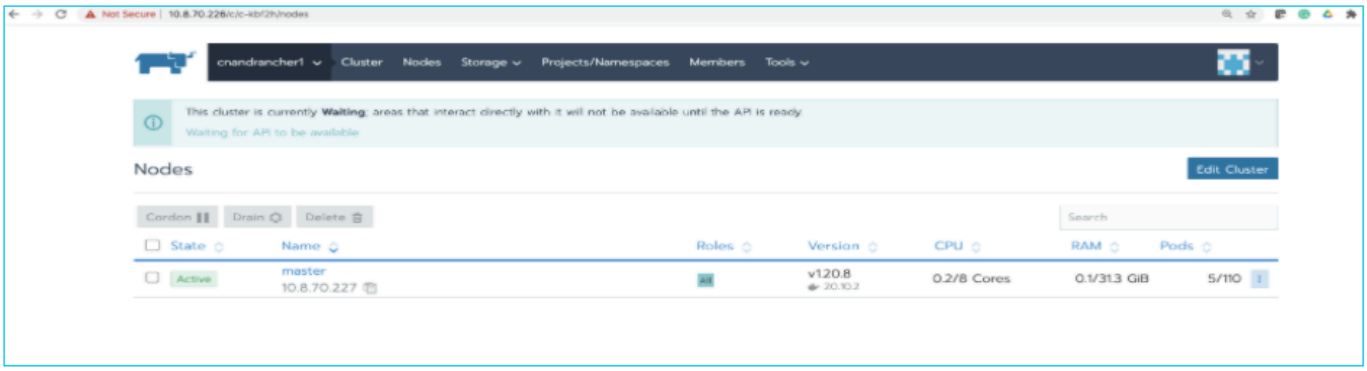

4. 他のすべてのフィールドのデフォルト値を保持してから、「次へ」をクリックします。



5. [ノード]オプションで、3つのノードロールオプションをすべて選択し、SSH を使用してマスターノードで指定されたコマンドを実行します。



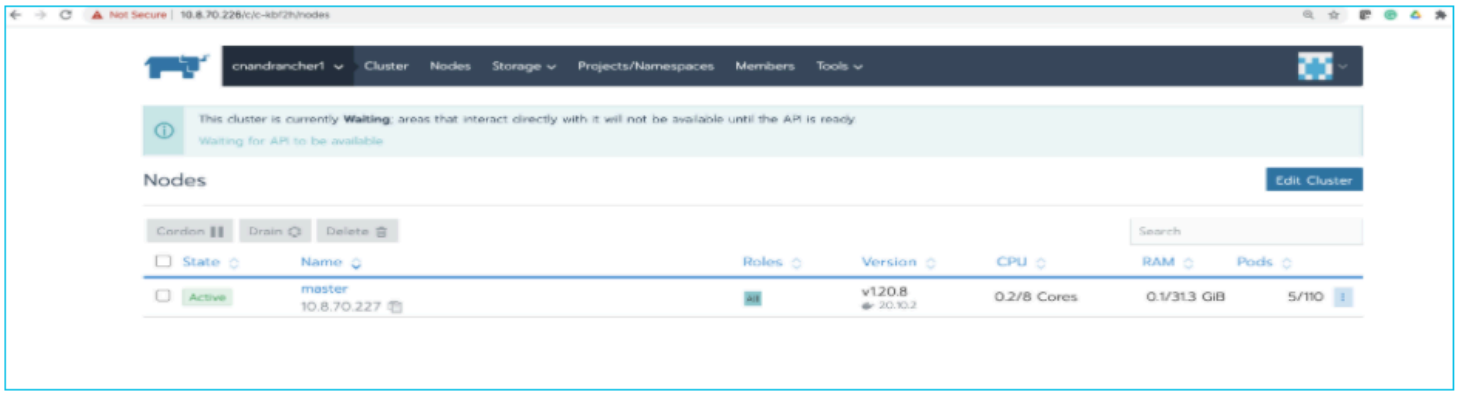

6. マスターノードが正常に追加されたことを確認します。




7. 各ワーカーノードに SSH で接続し、以下のコマンドを実行します。:

```
sudo docker run -d --privileged --restart=unless-stopped --
net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run
 rancher/rancher-agent:v2.5.8 --server https://10.8.70.226 --token
 547vwm6nmvnbr877w2mfvjmst6m892vtzztgh2mfg59m6t7wbknbfr --ca-
checksum
 1ea40f7c3499beb82f4582ecf05cc4300baea8abee079099e87b52c80e40a7bb
 --worker
```

1つのマスターノードと2つのワーカーノードでコマンドを正常に実行すると、以下に示すように、Rancher クラスターの準備ができていることがわかります。




# Rancher クラスターオプションの YAML ファイルを変更する

CN-Series ファイアウォールをデプロイする前に、以下に示すようにクラスタ オプション YAML ファイルを変更する必要があります。

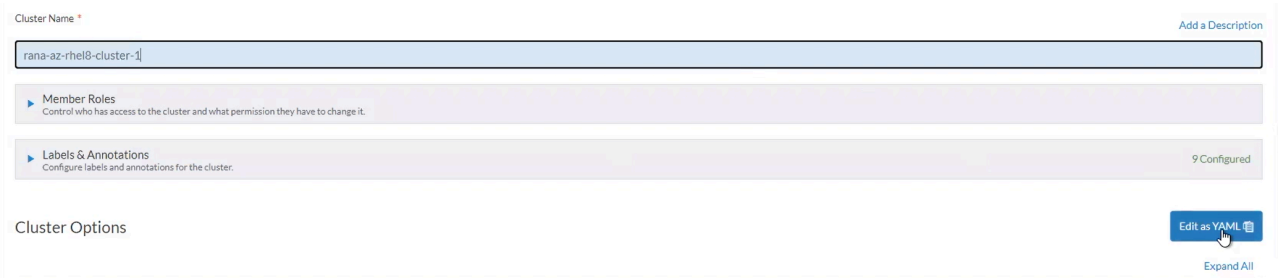
*Rancherを使用した CN-Series のファイアウォールは、k8s 1.20.5で Rancher 2.5 以降をサポートします。*

**STEP 1 |** 以前に作成した管理者の資格情報を使用して、Rancher ポータルにログインします。

**STEP 2 |** ナビゲーションメニューをクリックし、クラスタ管理]を選択します。

**STEP 3 |** 変更するクラスタを見つけ、縦方向の省略記号メニューをクリックして、**Edit Config (**設定の編集**)** を選択します。

**STEP 4 |** **Edit as YAML (YAML**として編集**)** をクリックします。




*Rancher* の各バージョンについては、Rancherドキュメントを参照してください。

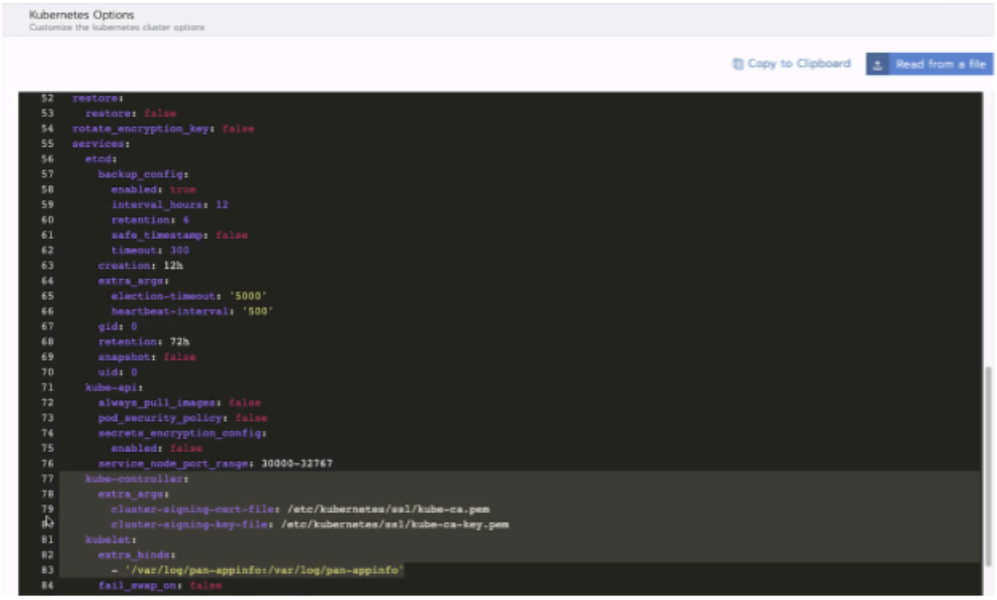
 

**STEP 5 |** 既存の YAML ファイルの **Services (**サービス**)** セクションに以下の行を追加します。

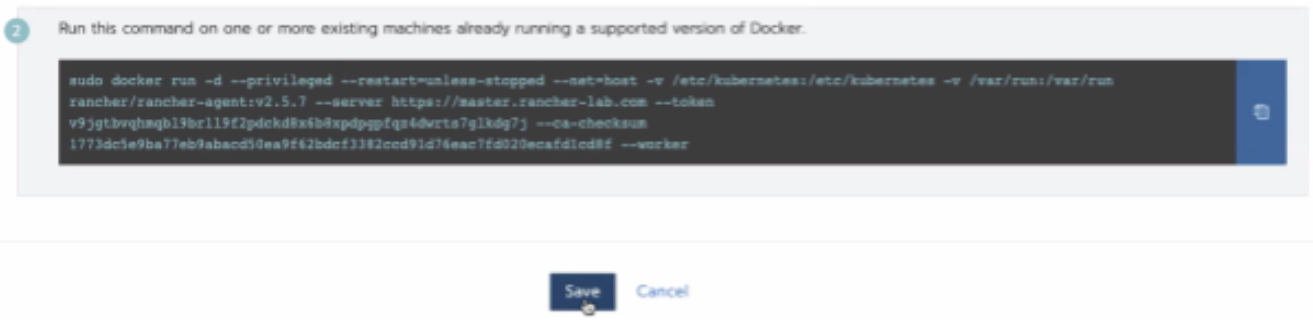
```
kube-controller: extra_args: cluster-signing-cert-file: "/etc/
kubernetes/ssl/kube-ca.pem" cluster-signing-key-file: "/etc/
kubernetes/ssl/kube-ca-key.pem"
```

```
kubelet: extra_binds: - '/mnt:/mnt:rshared' - '/var/log/pan-
appinfo:/var/log/pan-appinfo'
```

'*/mnt*'以外のストレージパスを使用している場合は、*extra_binds*の下のストレージパスを必ず変更する必要があります。




**STEP 6 |** 保存をクリックし、クラスタのアップグレードがアクティブになるまで待ってから CN-Series ファイアウォールをデプロイします。







# CN-Series デプロイメント YAML ファイル内の編集可能なパラメータ

YAML ファイルには、いくつかの編集可能なパラメータが含まれています。CN-Series ファイアウォールのデプロイを成功させるために変更する必要のあるパラメータの表を以下に示します。

- PAN-CN-MGMT-CONFIGMAP
- PAN-CN-MGMT-SECRET
- PAN-CN-MGMT
- PAN-CN-NGFW-CONFIGMAP
- PAN-CN-NGFW
- PAN-CNI-CONFIGMAP
- PAN-CNI
- PAN-CNI-MULTUS

**PAN-CN-MGMT-CONFIGMAP**

| PAN-CN-MGMT-CONFIGMAP | |
|---|---|
| 詳細ルーティング (Kubernetes 3.0.0 のデプロイメントに必要)<br>`PAN_ADVANCED_ROUTING:"true"` | Kubernetes 3.0.0 プラグインで高度なルーティングを使用している場合は、最初にPAN-OS で有効にしてから、テンプレート スタックで手動で構成する必要があります。有効にした後、設定をコミットしてプッシュします。詳細については、[詳細ルーティング]を参照してください。 |
| Panorama IP アドレス<br>`PAN_PANORAMA_IP:` | CN-MGMT ポッドが接続する Panorama IP アドレスが設定されます。Panorama 管理サーバーを高可用性 (HA) 設定にしてある場合は、プライマリ アクティブな Panorama の IP アドレスを指定します。<br>Panorama IP アドレスは、ダッシュボード > 一般情報で確認できます。 |
| デバイス グループ名<br>`PAN_DEVICE_GROUP:` | CN-NGFW ポッドを割り当てるデバイスグループ名を指定します。Panorama から、CN-MGMT ポッドのペアによって管理される (または PAN-SERVICE-NAME に属している) すべての CN-NGFW ポッドに同じポリシーをプッシュします。 |

| PAN-CN-MGMT-CONFIGMAP | |
|---|---|
| | デバイス グループ名は、**Panorama** > デバイスグループで確認できます。 |
| テンプレート スタック名<br>`PAN_TEMPLATE_STACK:` | ネットワーク上でファイアウォール (CN-NGFW ポッド) を動作させるための設定を行うことができます。<br>テンプレートスタック名は、**Panorama** > テンプレートで確認できます。 |
| ログ コレクタ グループ名<br>`PAN_PANORAMA_CGNAME:` | CN-NGFW ファイアウォール上で生成されたログ用のログ ストレージを有効にします。コレクタ グループがない場合は、ファイアウォール ログが保存されません。<br>コレクタ グループ名は、**Panorama** > コレクタグループで確認できます。 |
| (任意)<br>`#CLUSTER_NAME:` | クラスタ名を指定します。CN-MGMT ポッドのホスト名に、`PAN-CN-MGMT.yaml` で定義された StatefulSet 名とこのオプションの CLUSTER_NAME の組み合わせが使用されます。このホスト名を使用すれば、同じ Panorama アプライアンス上で複数のクラスタを管理している場合に、異なるクラスタに関連付けられたポッドを識別することができます。ベスト プラクティスとして、ここで指定するものと同じ名前を Kubernetes plugin on Panorama でも使用してください。 |
| (任意) Panorama HA ピアの IP アドレス<br>`#PAN_PANORAMA_IP2:` | 高可用性設定の Panorama ピア (パッシブ セカンダリ) の IP アドレス。PAN_PANORAMA_IP がプライマリ アクティブな Panorama のものであることを確認してください。<br>Panorama HA ピア IP アドレスは、**Panorama** > 高可用性 > セットアップで確認できます。 |
| (GTP のために必要) GTP セキュリティ<br>`#PAN_GTP_ENABLED: "true"` | CN-Series ファイアウォール上の GTP セキュリティに対してこのパラメータを有効にします。GTP を有効にすると、Panorama を使用して、GTP セキュリティを設定し、ファイアウォール上の GTP トラフィックを監視できます。 |
| (プライマリ CNI でジャンボ フレームが使用されていない場合に、ジャンボ フレーム | CN-MGMT ポッドは、起動中に、eth0 MTU を使用して、ジャンボ フレーム |

| PAN-CN-MGMT-CONFIGMAP | |
|---|---|
| サポートのために必要) ジャンボ フレーム モード<br>`#PAN_JUMBO_FRAME_ENABLED: "true"` | モードを有効にするかどうかを自動検出します。そのため、セカンダリ CNI でジャンボ フレームを使用しているが、プライマリ CNI では使用していない場合、`PAN_JUMBO_FRAME_ENABLED:"True"` を定義して、CN-Series ファイアウォール上でジャンボ フレーム モードを有効にする必要があります。<br>この変更は、CN-MGMT StatefulSet のデプロイ前に行う必要があります。 |
| (柔軟なシステム リソース割り当てのために必要)<br>• DaemonSet としての CN-Series<br>`#PAN_NGFW_MEMORY:"42Gi"`<br>• K8s サービスとしての CN-Series<br>`#PAN_NGFW_MEMORY:"6.5Gi"`<br>`#PAN_NGFW_MEMORY:"42Gi"`<br>*5G ネイティブ セキュリティのために、48Gi が推奨されます。* | デプロイメント ニーズを満たすために、スループットを向上する必要があり、より多くのメモリを設定したい場合は、このパラメータを使用してメモリ値を定義します。<br>• DaemonSet としての CN-Series<br>スモールの容量は 42Gi 以下、ラージの容量は 42Gi 以上です。<br>• K8s サービスとしての CN-Series<br>スモールの容量は 6.5Gi 未満、ミディアムの容量は 6.5Gi～42Gi、ラージの容量は 42Gi 以上です。<br>この変更では、*pan-cn-ngfw.yaml* で定義したメモリ以上の割り当てが必要です。 |
| （任意）AF-XDP<br>`#PAN_DATA_MODE: “next-gen”` | このパラメータは、アドレスファミリエクスプレスデータパス（AF-XDP）を有効にするために必要です。<br>AF-XDP は、クラウドネイティブサービスに適した高性能パケット処理に最適化された eBPF ベースのソケットで、効果的なスループットを増加させます。これには、カーネルバージョン 5.4 以降が必要です。また、ジャンボモードはサポートされていません。ジャンボモードはデフォルトで有効になっているため、EKS はこのパラメータを使用できません。<br>さらに、PAN-CN-NGFW では特権モードが必要です。 |

| PAN-CN-MGMT-CONFIGMAP | |
|---|---|
| (HPA を有効にするために必要)<br>(AKS と GKE) #HPA_NAME<br>(EKS のみ) #PAN_NAMESPACE_EKS<br>(AKS のみ) #PAN_INSTRUMENTATION_KEY | サービスとしての CN-Series ファイアウォールで水平ポッド自動スケーリング (HPA) を有効にするには、いくつかのパラメーターが必要です。<br>• それぞれの環境について、名前空間またはテナントごとに HPA リソースを識別する一意の名前を指定する必要があります。<br>• AKS デプロイの場合は、Azure Application Insight インストルメンテーション キーを指定する必要があります。 |

以下のデフォルト値が、*pan-cn-mgmt-configmap.yaml* ファイルで定義されています。

```
metadata: name: pan-mgmt-config namespace: kube-system
 data:PAN_SERVICE_NAME: pan-mgmt-svc PAN_MGMT_SECRET: pan-mgmt-
secret
```

これらのデフォルト値を使用すれば、これらのファイルを迅速な概念実証に使用することができます。たとえば、最大 30 個の PAN-NGFW ポッドを管理する PAN-MGMT ポッドの耐障害性のあるペアを複数デプロイするように値を変更する場合は、別のサービス名を使用するように pan-mgmt-svc を変更する必要があります。これらの値を変更したら、このファイルで定義された値と一致するように他の YAML ファイル内の対応する参照を更新する必要があります。

### PAN-CN-MGMT-SECRET

| PAN-CN-MGMT-SECRET | |
|---|---|
| VM 認証キー<br>PAN_PANORAMA_AUTH_KEY: | Panorama でファイアウォールを認証して、各ファイアウォールを管理対象デバイスとして追加できるようにします。VM 認証キーは、デプロイメントの有効期限が切れるまで必要になります。接続リクエストに有効なキーがない場合、CN-Series ファイアウォールを Panorama に登録できません。<br>CNシリーズ ファイアウォール用のKubernetesプラグインのインストールを参照してください。 |
| CN-Series のデバイス証明書<br>CN-SERIES-AUTO-REGISTRATION-PIN-ID | サイト ライセンス資格を取得して、Palo Alto クラウド配信型サービスに安全にアクセスするためには、ファイアウォールにデバイス証 |

| PAN-CN-MGMT-SECRET | |
|---|---|
| CN-SERIES-AUTO-REGISTRATION-PIN-VALUE | 明書が必要です。Palo Alto Networks CSP 上で PIN ID と PIN 値を生成し、期限切れ前の PIN を使用します。以下に例を示します。<br>CN-SERIES-AUTO-REGISTRATION-PIN-ID:<br>"01cc5-0431-4d72-bb84-something”<br>CN-SERIES-AUTO-REGISTRATION-PIN-VALUE:<br>"12……..13e"<br>*CN-SERIES-AUTO-REGISTRATION-API-CSP* の以下の追加のフィールドは、コメントアウトされており、必要ありません。*"certificate.paloaltonetworks.com"*<br>CNシリーズ ファイアウォールへのデバイス証明書のインストールを参照してください。 |

**PAN-CN-MGMT**

| PAN-CN-MGMT | |
|---|---|
| CN-MGMT ファイアウォールの初期コンテナイメージのイメージ パス<br>`initContainers: - name: pan-mgmt-init image: <your-private-registry-image-path>` | 初期コンテナが、CN-MGMT ポッドのインスタンス間と、CN-MGMT ポッドと CN-NGFW ポッド間の通信を保護するために使用される証明書を生成します。<br>CN-MGMT コンテナの Docker イメージをアップロードした場所を指すようにイメージ パスを編集します。 |
| CN-MGMT イメージ コンテナのイメージ パス:<br>`initContainers: - name: pan-mgmt image: <your-private-registry-image-path>` | CN-MGMT コンテナの Docker イメージをアップロードした場所を指すようにイメージ パスを編集します。 |
| CN-MGMT ファイアウォールのホスト名<br>`kind:StatefulSet metadata: name: pan-mgmt-sts` | CN-MGMT ファイアウォールのホスト名は、StatefulSet 名と、任意で `pan-cn-mgmt-configmap.yaml` で定義したクラス |

| PAN-CN-MGMT | |
|---|---|
| | タ名を組み合わせることによって導出されます。<br><br>CN-MGMT ポッドのデフォルトのホスト名は、pan-mgmt-sts-0 と pan-mgmt-sts-1 です。これは、StatefulSet 名が pan-mgmt-sts で、クラスタ名が定義されていないためです。<br><br>ホスト名が *30* 文字を超えている場合は、*30* 文字で切り捨てられます。 |
| (柔軟なシステム リソース割り当てのためのメモリを定義した場合に必要) | 40Gi 以上のメモリ値を `#PAN_NGFW_MEMORY:` の「`40Gi`」では、`pan-cn-mgmt-configmap.yaml`要求に同じ値があることを確認し、CPU とメモリを制限して、<br><br>`containers: resources: requests:` でより高いキャパシティ使用率を達成します。`#` 希望するロギングに基づいて設定可能, `capacities cpu:"4" memory:"16.0Gi" limits: cpu:"4" memory:"16.0Gi"`<br><br>5G ネイティブ セキュリティの場合の推奨値は、cpu=4 と memory=16Gi です。 |
| (オンプレミスまたは自己管理ネイティブ Kubernetes デプロイメントの場合のみ)<br><br>`storageClassName: local` | 自己管理デプロイメントの場合のデフォルト設定は、"storageClassName: local" です。<br><br>クラスタで永続ボリューム (PV) が動的にプロビジョニングされている場合は、その storageClass と一致するように "storageClassName: local" を変更するか、DefaultStorageClass が使用されている場合はこれらの行を削除する必要があります。<br><br>クラスタで PV が動的にプロビジョニングされていない場合は、クラスタ管理者が、少数の PV が 2 セット (PAN-CN-MGMT statefulSet ポッドごとに 1 つずつ) 含まれている指定された `pan_cn_pv_local.yaml` を使用して静的 PV を作成できます。セットアップ内のボリュームと一致するように `pan_cn_pv_local.yaml` を変更して、それ |

| PAN-CN-MGMT | |
|---|---|
| | を PAN-CN-MGMT.yaml のデプロイ前にデプロイすることができます。 |

### PAN-CN-NGFW-CONFIGMAP

以下を変更する必要がない場合は、PAN 値を変更する必要がありません。

- PAN_SERVICE_NAME: pan-mgmt-svc

  サービス名は、PAN-CN-MGMT-CONFIGMAPで定義した名前と一致する必要があります。

- FAILOVER_MODE: failopen

  これを failclose に変更できます。CN-NGFW がライセンスの取得に失敗した場合にのみ有効になります。

  - fail-open モードでは、ファイアウォールがパケットを受信して、それを検査せずに送信します。fail-open モードに移行すると、内部で再起動が引き起こされ、しばらくトラフィックが中断します。
  - fail-close モードでは、ファイアウォールが受信したすべてのパケットをドロップします。fail-close モードでは、CN-NFGW もダウンし、他のライセンス供与された CN-NFGW が使用するために割り当てられたスロットが解放されます。

- CPU ピニング - pan-cn-ngfw-configmap.yaml では、CPU ピニングとハイパースレッディングが無効になっています。Palo Alto Networks サポートから指示されない限り、この設定を切り替えて、ハイパースレッディングを使用した論理コアではなく、専用の物理コアに対して CPU ピニングを有効にしないでください。

  ```
  PAN_CPU_PINNING_ENABLED:"True"/"False"
  PAN_HYPERTHREADING_ENABLE:"True"/"False"
  ```

### PAN-CN-NGFW

| PAN-CN-NGFW | |
|---|---|
| CN-NGFW コンテナ イメージのイメージパス<br><br>image<br><br>`containers: - name: pan-ngfw-container image: <your-private-registry-image-path>` | CN-NGFW コンテナの Docker イメージをアップロードした場所を指すようにイメージパスを編集します。 |
| (柔軟なシステム リソース割り当てのためのメモリを定義した場合に必要) | 40Gi 以上のメモリ値を #PAN_NGFW_MEMORY:の「40Gi」 で |

| PAN-CN-NGFW | |
|---|---|
| | は、pan-cn-mgmt-configmap.yaml 要求に同じ値があり、CPU とメモリの制限が<br>`containers: resources: requests:` の下で保証された QoS を達成するようにしてください。#望ましいスループットに基づき設定可能、実行中のポッド数 `cpu:"1" memory:"40.0Gi" limits: cpu:"1" memory:"40.0Gi"`<br>5G ネイティブ セキュリティの場合の推奨値は、cpu=12 と memory=48Gi です。 |
| 注:<br>• 以下の注釈は、PAN-NGFW daemonset を識別します。<br>`paloaltonetworks.com/app: pan-ngfw-ds`<br>この値は変更しないでください。<br>• 以下の注釈は、ファイアウォール名 ("pan-fw") を識別します。<br>`paloaltonetworks.com/firewall: pan-fw`<br>`pan-cni-configmap.yaml` では、このファイアウォール名が `cni_network_config: “firewall”` 内の名前と正確に一致する必要があります。<br>また、この注釈は、各アプリケーション ポッドをデプロイするために使用するアプリケーション yaml 内の名前と正確に一致する必要があります。 | 各ノード上の CN-NGFW ポッドが、次の注釈が含まれているアプリケーション ポッドと名前空間を保護します。<br>`paloaltonetworks.com/firewall: pan-fw`<br>この注釈はそのまま維持してください。 |
| （任意）AF-XDP<br>`ImagePullPolicy:Always securityContext: capabilities: #add: ["NET_ADMIN","NET_RAW","NET_BROADCAST","NET_BIND_SERVICE"] add: ["ALL"] privileged: true resources:` | 左側のセクションに `privileged: true` を追加する必要があります。このパラメータは、アドレスファミリエクスプレスデータパス（AF-XDP）を有効にするために必要です。<br>また、PAN-CN-MGMT-CONFIGMAPで AF-XDP を有効化する必要があります。 |

### PAN-CNI-CONFIGMAP

これらのパラメータは任意です。

| PAN-CNI-CONFIGMAP | |
|---|---|
| アプリケーション ポッドが属している可能性のあるファイアウォール名のリスト:<br>`"firewall": [ "pan-fw" ]` | 変更する必要はありませんが、`pan-cn-ngfw.yaml` 内の注釈 `paloaltonetworks.com/firewall: pan-fw` を変更する場合は、一致するように `"firewall": [ "pan-fw" ]` の値を置き換える必要があります。 |
| `"exclude_namespaces": []` | 変更する必要はありませんが、特定の名前空間を除外する場合は、**“exclude_namespaces"** を追加して、その名前空間内のアプリケーション ポッドの注釈が無視され、トラフィックが検査用に CN-NGFW ポッドにリダイレクトされないようにします。 |
| `"security_namespaces": [ "kube-system" ]` | security_namespaces で CN-NGFW daemonset をデプロイした名前空間を追加します。デフォルトの名前空間は kube-system です。 |
| “interfaces” | トラフィックを検査用に CN-NGFW ポッドにリダイレクトするアプリケーション ポッドにインターフェースを追加します。デフォルトで、eth0 トラフィックだけが検査され、文字列のコンマ区切りリストとして新しいインターフェースを追加できます (例: [“eth0”, “net1”, “net 2”])。<br>`cni_network_config:`<br>`{ "cniVersion":"0.3.0", "name": "pan-cni", "type": "pan-cni", "log_level": "debug", "appinfo_dir": "/var/log/pan-appinfo", "mode": "daemonset", "firewall": [ "pan-fw" ], "interfaces": ["eth0", "net1", "net2", "net3"],`<br>`}` |

| PAN-CNI-CONFIGMAP | |
|---|---|
| | これに加えて、アプリケーション ポッド内の *k8s.v1.cni.cncf.io/networks* 注釈に *pan-cni* を付加する必要もあります。<br>以下に例を示します。<br>`metadata: name: testpod annotations: paloaltonetworks.com/firewall: pan-fw k8s.v1.cni.cncf.io/networks: sriov-net1, sriov-net2, macvlan-conf, pan-cni`<br>*CN-Series* は、現時点で、*DPDK* をサポートしておらず、アプリケーション ポッドによる *DPDK* の使用を許可していません。アプリケーションが自動的に非 *DPDK* モードに調整されない場合は、アプリケーション ポッドを変更する必要がある場合があります。 |
| (Kubernetes サービスとしての CN-Series のみ)<br>“dpservicename”<br>“dpservicenamespace” | CN-Series をサービスとしてデプロイする場合は、dpservicename と dpservicenamespace が必要です。デフォルトでは、dpservicenameは “pan-ngfw-svc”、dpservicenamespaceは “kube-system” です。 |

### PAN-CNI

| PAN-CNI | |
|---|---|
| CNI バイナリと各ノード上の CNI ネットワーク設定ファイルを含む PAN-CNI コンテナ イメージのイメージ パス。<br>`containers: name: install-pan-cni image: <your-private-registry-image-path>` | PAN-CNI コンテナの Docker イメージをアップロードした場所を指すようにイメージ パスを編集します。 |

### PAN-CNI-MULTUS

VMware TKG+ などの Kubernetes の自己管理実装またはネイティブ実装で Multus CNI を使用している場合は、`pan-cni.yaml` ではなく、`pan-cni-multus.yaml` を使用します。

# CN-Series ファイアウォールを使用して 5G をセキュリティで保護する

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

Kubernetes 上のモバイル オペレーター ネットワーク内のプライベート エンタープライズおよび 5G モバイル パケット コア デプロイメントの 5G トラフィックの可視性と制御に関して、以降のセクションで、サポートされている環境と CN-Series ファイアウォール上の GTP セキュリティと 5G ネイティブ セキュリティを有効にするように YAML ファイルを変更する方法を確認します。CN-Series ファイアウォールをデプロイするときにこれらの機能を有効にすることに加えて、GTP セキュリティおよび/または SCTP セキュリティを Panorama で有効にする必要もあります。

| | |
| --- | --- |
| コンテナ ランタイム | Docker<br>CRI-O<br>Containerd |
| Kubernetes のバージョン | 1.17～1.27 |
| クラウド プロバイダが管理する Kubernetes | • AWS EKS(DaemonSetとしてのCNシリーズおよびデプロイメントのサービス モードとしてのCNシリーズの場合は1.17～1.27)<br>• AWS EKS (CNF デプロイメント モードとしてのCNシリーズの場合は 1.17～1.22)<br>• AWS EKS (CN-Clusterデプロイメントとしてのシリーズの場合は1.22～1.27)<br>• AWS Outpost（1.17～1.25）のEKS<br>*AWS Outpost の EKS 用 CN-Series は SR-IOV または Multus をサポートしていません。* |

| | |
|---|---|
| | • Azure AKS（1.17～1.27）<br><br>*Azure AKS*では、*Kubernetes 1.25*以降をサポートするために最低限必要なバージョンは*PAN-OS 11.0.2*です。<br>• GCP GKE（1.17～1.27）<br><br>*GKE* データプレーン *V2* が含まれています。<br>• OCI OKE (1.23) |
| 顧客が管理する Kubernetes | パブリック クラウドまたはオンプレミス データ センター。<br><br>Kubernetes のバージョン、CNI のタイプ、および ホスト VM OS のバージョンがこの表のとおりであることを確認してください。<br><br>VMware TKG+ バージョン 1.1.2<br>• インフラストラクチャ プラットフォーム - vSphere 7.0<br>• Kubernetes ホスト VM OS - Photon OS |
| Kubernetes ホスト VM | オペレーティングシステム：<br>• Ubuntu 16.04<br>• Ubuntu 18.04<br>• Ubuntu-22.04<br>• RHEL/Centos 7.3 以降<br>• CoreOS 21XX、22XX<br>• Container-Optimized OS |
| | Linux カーネルバージョン：<br>• 4.18以降（K8s サービスモードのみ）<br>• AF_XDP モードを有効にするには 5.4 以降が必要です。詳細については、CN-Series デプロイメント YAML ファイルの編集可能なパラメーターを参照してください。 |
| | Linux カーネル Netfilter：Iptables |
| CNI プラグイン | CNI Spec 0.3 以降:<br>• AWS-VPC |

| | |
|---|---|
| | • Azure<br>• Calico<br>• Flannel<br>• Weave<br>• Openshift の場合、OpenshiftSDN<br>• 以下は、CN-Series ファイアウォールで DaemonSet としてサポートされています。<br>  • Multus<br>  • Bridge<br>  • SR-IOV<br>  • Macvlan |
| OpenShift | • バージョン 4.2、4.4、4.5、4.6、4.7、4.8、4.9、4.10、4.11、4.12 および4.13。<br><br>*OpenShift 4.7* は、*CN-Series* で *DaemonSet* としてのみ認定されています。<br><br>*PAN-OS 11.0.2*は、*4.12*以上をサポートするために最低限必要なバージョンです。<br>• AWS での OpenShift |

| コンテナ ランタイム | バージョン |
|---|---|
| CN-Series ファイアウォール | PAN-OS 10.0.3 以降 |
| Kubernetes プラグイン | 1.0.1 以降 |
| Panorama | 10.0.0 以降 |
| CN-Series ファイアウォールをデプロイするために使用する YAML ファイル内のすべての編集可能なパラメータのリストを以下に示します。詳細については、CN シリーズ デプロイメントyamlファイル内の編集可能なパラメーターおよびCN シリーズ コア ビルディング ブロックを参照してください。 | |
| GTP を有効化にする | pan-cn-mgmt-configmap.yaml で、`PAN_GTP_ENABLED :"True"` を設定してから、CN-MGMT StatefulSet をデプロイします。 |

| コンテナ ランタイム | バージョン |
| --- | --- |
| ジャンボ フレーム モードを有効にする | pan-cn-mgmt-configmap.yaml で、`PAN_JUMBO_FRAME_ENABLED:"True"` を設定してから、CN-MGMT StatefulSet をデプロイします。<br><br>CN-MGMT ポッドは、起動中に "eth0" MTU を使用して、ジャンボ フレーム モードを有効にするかどうかを自動検出します。そのため、セカンダリ CNI でジャンボ フレームを使用しているが、プライマリ CNI では使用していない場合、`PAN_JUMBO_FRAME_ENABLED:"True"` を定義して、CN-Series ファイアウォール上でジャンボ フレーム モードを有効にする必要があります。<br><br>*CN-Series* は、現時点で、*DPDK* をサポートしておらず、アプリケーション ポッドによる *DPDK* の使用を許可していません。アプリケーションが自動的に非 *DPDK* モードに調整されない場合は、アプリケーション ポッドを変更する必要がある場合があります。 |
| システム リソースの柔軟性を有効にする | デプロイメント ニーズを満たすために、より高いスループットと多くのメモリが必要な場合は、pan-cn-mgmt-configmap.yaml で次のように設定します。PAN_NGFW_MEMORY="48Gi"<br><br>テンプレート化 *(Helm)* では、*CN-NGFW* ポッドに割り当てられたものと同じ変数を使用できます。より大きなメモリ フットプリントを有効にした場合は、*CN-MGMT StatefulSet* が *1* つの *CN-NGFW* ポッドしかサポートしません。 |
| 5G 用の vCPU とメモリを設定する | CN-MGMT ポッド (pan-cn-mgmt.yaml 内) と NGFW ポッド (pan-cn-ngfw.yaml 内) の推奨設定は、CPU とメモリの "request" と "limit" |

| コンテナ ランタイム | バージョン |
| --- | --- |
| | に同じ値を設定して、保証された QoS を実現することです。<br><br>CN-MGMT ポッドの場合の推奨値は、cpu=4 と memory=16Gi です。CN-NGFW ポッドがデプロイされているノードと同じノードまたは異なるノードなど、CN-MGMT ポッドの配置を制御するには、k8s の node-selector 機能を使用します。 |
| | CN-NGFW ポッドの場合の推奨値は、cpu=12 と memory=48Gi です。CN-NGFW ポッドがデプロイされているノードと同じノードまたは異なるノードなど、CN-NGFW ポッドの配置を制御するには、k8s の node-selector 機能を使用します。 |
| CNI yaml ファイルを選択する | Multus CNI は、他の CNI プラグインを呼び出すメタプラグインとして機能します。OpenShift 環境では、Multus がデフォルトで有効になるため、`pan-cni.yaml` を使用できます。Multus はサポートされているが、任意である他の環境 (自己管理 (ネイティブ) 環境など) では、`pan-cni.yaml` ではなく、`pan-cni-multus.yaml` を使用します。 |

CN シリーズ ファイアウォールのデプロイメントを続行する前に、CN シリーズ ファイアウォールのシステム要件も確認してください。

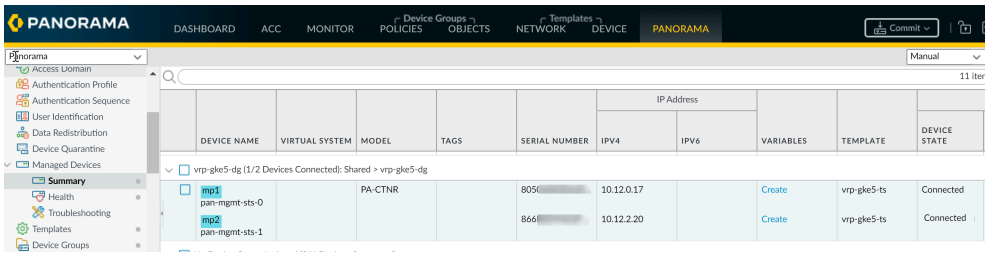
# Panorama を設定して Kubernetes デプロイメントをセキュリティで保護する

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmチャートを使用したCNシリーズのデプロイメント用 |

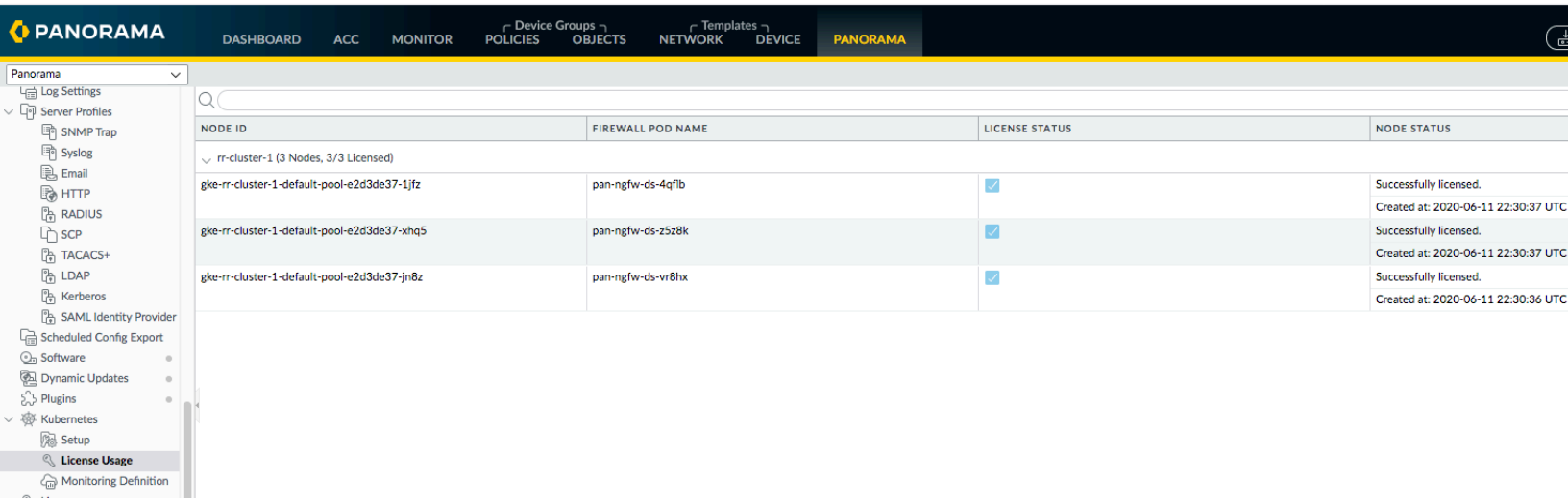
CNシリーズのKubernetesプラグインのインストールとCNシリーズ ファイアウォールのデプロイが終わったら、Kubernetesクラスタを監視し、トラフィック適用を可能にするセキュリティポリシーを設定するに、以下のタスクを実行する必要があります

**STEP 1 |** CN-MGMT ポッドが Panorama 上で登録され、CN-NGFW ポッドがライセンスされていることを確認します。

1. **Panorama** > **Managed Devices**（マネージド デバイス） > **Summary**を選択します。




2. **[Panorama]** > **[プラグイン]** > **[Kubernetes]** > **[ライセンス使用状況]** を選択して、クラスタ内の各ノードにライセンス トークンが割り当てられていることを確認します。




**STEP 2 |** ログを Panorama に転送するためのログ転送プロファイルを作成します。

このプロファイルでは、ファイアウォール上で生成されるさまざまなログの宛先が定義されます。

1. k8s デプロイメント用に作成したデバイス グループを **[デバイス グループ]** ドロップダウンから選択します。
2. **[オブジェクト]** > **[ログ転送]** を選択し、**[追加]** をクリックします。
3. プロファイルを識別する **Name**（名前）を入力します。プロファイルを自動的に新しいセキュリティ ルールとゾーンに割り当てるには、「**`default`**」と入力します。デフォルト プロファイルが必要ないか、既存のデフォルト プロファイルをオーバーライドする場合は、プロファイルをセキュリティ ルールに割り当てるときに識別しやすい名前を入力します。
4. 転送するログ タイプを追加します。
5. **OK** をクリックします。

**STEP 3 |** タグを指定されたデバイス グループにプッシュするように Kubernetes プラグインを設定します。

Panorama が事前に定義されたラベルと通知グループ (任意) を取得する Kubernetes クラスタの名前を含むモニタリング定義を追加する必要があります。

*CN-Series* が *kube-system* 以外の名前空間にデプロイされている場合は、通知グループが必要です。

通知グループは、タグ更新を受け取るデバイス グループのリストです。Kubernetes プラグインでは、通知グループにクラスタ外部のファイアウォール (つまり、属性を収集している Kubernetes クラスタと同じデバイス グループに属していないファイアウォール) を含める必要があります。

CN-Series ファイアウォールのデプロイに使用される YAML ファイルでデバイス グループ名が指定されているため、Kubernetes プラグインは自動的に、クラスタ内部のすべてのデバイス グループを認識し、デフォルトで、事前に定義されたすべてのタグをそれらのデバイス グループにプッシュします。

Kubernetes プラグインは、Kubernetes シークレットを使用して、各クラスタ内のデバイス グループを自動的に認識します。CN-MGMT StatefulSet がデプロイされるたびに、シークレットが Kubernetes API サーバーに発行され、Panorama が次のモニタリング間隔でそれを認識します。

1. クラスタを監視するためのKubernetesプラグインをセットアップします。
2. 通知グループを追加します。通知グループを追加して、Kubernetesクラスタに関連するタグを受け取るデバイス グループを選択します。
   1. **Panorama** > プラグイン > **Kubernetes** > セットアップ > 通知グループ追加を選択します。
   2. 通知グループの名前を最大 31 文字で入力します。
   3. クラスタ用に作成された外部タグ (デフォルト) に加えて内部タグを共有する場合は、[**Enable sharing internal tags with Device Groups (**デバイス グループとの内部タグの共有を可能にする**)**] を選択します。
   4. タグを登録するデバイス グループを選択します。

      選択した通知グループに対して、Panorama は外部タグだけをプッシュします。

      外部タグとは、外部サービス IP アドレスとクラスタ IP アドレスのポート、すべてのノードとノード ポートの外部 IP アドレス、および外部ロード バランサーの IP ア

ドレスとポートまたはノード ポート用に生成されたタグなどのクラスタ外部から到達可能なタグのことです。

内部タグには、内部クラスタ IP アドレス、ポッド IP アドレス、ノード、およびノード ポートに関する詳細が含まれています。

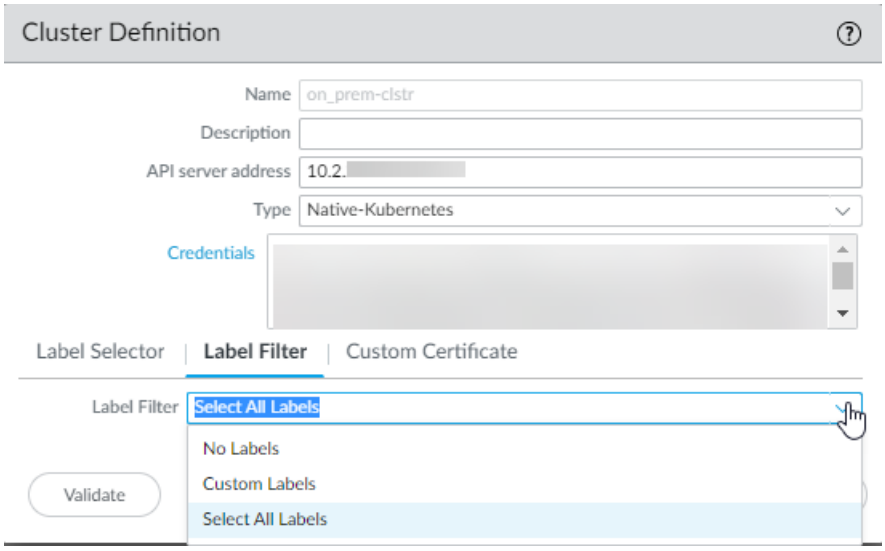
デフォルトで、Panorama は、CN-MGMT ポッドのデプロイに使用された YAML ファイルで定義されているように、検出したすべてのタグ (選択されたラベル フィルタに基づく) をクラスタに関連付けられたデバイス グループにプッシュします。

3. クラスタごとに 1 つモニタリング定義を追加します。
   1. **[PanoramaPlugins]** > **[Kubernetes]** > **[Monitoring Definition (**モニタリング定義**)]** を選択して、[追加] を選択します。
   2. モニタリング定義の名前を入力します。
   3. 監視するクラスタを選択します。
   4. (任意) IP アドレスとタグのマッピング情報を送信する [通知グループ] を選択します。

      デフォルトで、タグは、クラスタ内のすべての CN-NFGW ポッドと共有されます。
   5. **OK** をクリックして変更内容を保存します。
4. Panorama にコミットします。

**STEP 4 |** (任意) アプリケーション YAML ファイルからユーザー定義のラベルを受け取るように Kubernetes プラグインをセットアップします。

1. **[PanoramaPlugins]** > **[Kubernetes]** > [セットアップ] > [クラスタ] を選択し、リストからクラスタ定義を選択します。
2. 以下のオプションからラベル フィルタを選択します。




1. **No Labels** (ラベルなし) - Kubernetes ラベル用のタグを作成しません。
2. **Custom Labels** (カスタム ラベル) - 対象のラベルにのみタグを作成します。

   カスタム ラベルを使用するには、まず、Kubernetes のデプロイメントで YAML ファイルに注釈を付けてから、以下の組み合わせのいずれかを使用して、対応する IP アドレス用のカスタム タグを生成します。

   名前空間、キー、および値を指定します。すべてを指定する場合は * を使用します。3 つすべての入力が有効であれば、プラグインによってタグが作成されます。

   名前空間内のすべての一致するキーのタグを作成する場合は、名前空間とキーを指定します。

   名前空間内のすべてのラベルのタグを作成する場合は、名前空間のみを指定します。
3. **Select All Labels** (すべてのラベルを選択) - カスタム ラベルを含むすべての Kubernetes ラベルのタグを作成します。

3. ラベル セレクタ式を追加します。

   ラベル セレクタは、Kubernetes クラスタ内の指定されたラベルを照合し、そのラベルに関連付けられた IP アドレスを 1 つのタグにマッピングします。サポートされている

プレフィックス一覧については、「Kubernetes 属性の IP アドレスからタグへのマッピング」を参照してください。

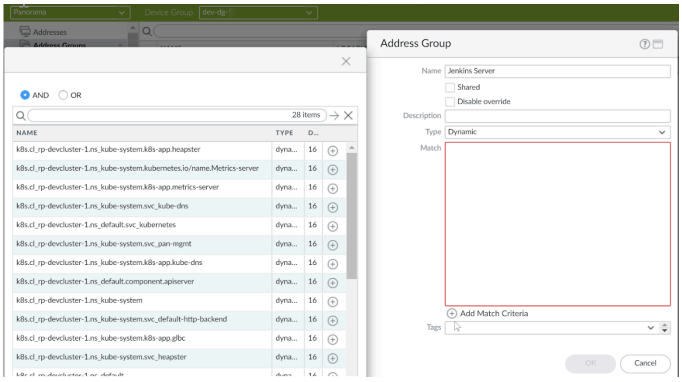
ラベル セレクタごとに、Panorama がダイナミック アドレス グループ内の一致条件として使用可能になるタグを 1 つ生成し、セキュリティ ポリシーを適用できるようにします。

1. **Tag Prefix (**タグ プレフィックス**)** - タグの識別を容易にするために各タグの最後に付ける語句。たとえば、ラベル セレクタk8s.cl_<clustername>.<selector-name> は、そのセレクタと一致するすべてのクラスタ IP と、そのセレクタと一致するすべてのポッド上で一致します。これらは、設定に応じて、すべての名前空間に含めることも、特定の名前空間だけ含めることもできます。
2. 名前空間 - * と入力してすべての名前空間を指定するか、特定の名前空間の値を入力します。
3. **Label Selector Filter(**ラベル セレクタ フィルタ**)** - Kubernetesプラグインは、ラベル キーとラベル値の設定ベースと等式ベースのセレクタをサポートしています。key = value; key ==value; key != valueという等値ベースのセレクタがサポートされています(例:app = redis)。app == web, tier != backend のように、1 つの式で複数のセレクタをコンマ区切りリストとして指定することもできます。key in (value1,value2)、key notin (value1,value2)、key 、 ! key などのセットベースのセレクタがサポートされています(例:frontend、backend)。
4. **Apply On (**適用先**)** - 適用するリソースのタイプ。サービス、ポッド、およびすべてです。

**STEP 5 |** ダイナミック アドレス グループをセットアップします。

1. CN-NGFW ポッドを管理するためのデバイス グループを選択します。
2. **Objects**（オブジェクト） > **Address Groups**（アドレス グループ）を選択します。
3. **Add**（追加）をクリックし、**Name**（名前）にアドレス グループの名前を、**Description**（内容）にアドレス グループの内容を入力します。
4. **Type**（タイプ）で **Dynamic**（ダイナミック）を選択します。

**STEP 6 |** **[Add Match Criteria (**一致条件を追加**)]** をクリックして、**AND** または **OR** 演算子を選択し、フィルタリングまたは照合する属性を選択します。



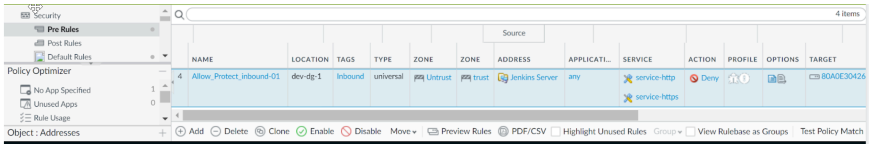

 

**STEP 7 |** **[OK]** と **[Commit on Panorama (Panorama** でコミット**)]** をクリックします。

[詳細...] リンクを使用して、オブジェクト (この例ではクラスタ内の Jenkins サーバー) に関連付けられた IP アドレスを表示します。

**STEP 8 |** トラフィック適用に関するセキュリティ ポリシー ルールを作成します。

1. **Policies**（ポリシー） > **Security**（セキュリティ）の順に選択します。
2. [追加] をクリックし、ポリシーの [名前] および [説明] を入力します。
3. **Source Zone (**送信元ゾーン**)** を追加して、トラフィックの送信元となるゾーンを指定します。
4. トラフィックが終端する **Destination Zone (**宛先ゾーン**)** を追加します。
5. **Destination Address (**宛先アドレス**)** については、先ほど作成したダイナミック アドレス グループを選択します。
6. トラフィックに対するアクション（[拒否]）を指定し、必要に応じて、デフォルト セキュリティ プロファイルをルールに関連付けます。
7. **Actions**（アクション）を選択し、作成した **Log Forwarding**（ログ転送）プロファイルを選択します。
8. **Commit**（コミット）をクリックします。




名前空間内の east-west トラフィックにセキュリティ ポリシーを適用することもできます。たとえば、staging cluster という名前のクラスタ内に stage-ns と db-ns という名前の 2 つの名前空間があるとします。このクラスタでは、投票アプリケーション用のフロントエンド ポッドが stage-NS でデプロイされ、Redis バックエンド ポッドが DB-NS 名前空間で動作しています。このクラスタを監視のために Kubernetes plugin on Panorama に追加すると、タグを作成するためのラベル メタデータが取得されます。これらのタグを使用して、セキュリティ ポリシー ルールを適用できます。これを行うには

- が必要です。フロントエンド アプリケーションとバックエンド アプリケーションのデプロイに使用する名前空間または YAML ファイルに paloaltonetworks.com/firewall: pan-fw の注釈が付けられていることを確認します。
- フロントエンドポッドとバックエンドポッドのダイナミック アドレス グループを作成します。

  クラスターに関連付けられたデバイス グループでダイナミック アドレス グループの設定をし、最初にフロントエンドサーバーのタグを選択する必要があります。次に、このプロセスを繰り返して、バックエンド サーバー用の別のダイナミック アドレス グループを作成します。

- フロントエンドポッドからバックエンドポッドへの Redis アプリケーションのトラフィックを許可するセキュリティ ポリシー ルールを追加します。

  送信元はフロントエンドサーバーのダイナミック アドレス グループであり、宛先はバックエンドサーバーのダイナミック アドレス グループであり、アクションは許可されます。

# Kubernetes 属性の IP アドレスからタグへのマッピング

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

Panorama上のKubernetes Plugin が、Kubernetes クラスタ内の事前に定義されたタグ、ポッドとサービス用のユーザー定義のラベル、およびサービス オブジェクト用のタグを作成します。

このプラグインは、次の Kubernetes オブジェクト用のタグを作成します。

- ポッド クラス:ReplicaSets、DaemonSets、StatefulSets
- サービス タイプ:ClusterIP、NodePort、LoadBalancer
- サービス オブジェクト：port、targetPort、nodePort、およびポッドインターフェース

デフォルトで、Panorama上のKubernetes Plugin は、Panorama 上で監視しているすべての Kubernetes クラスタから次の事前に定義されたタグを取得し、以下の形式でタグを作成します。その後で、これらのタグをダイナミック アドレス グループ内の一致条件として使用し、それぞれのタグに関連付けられた基礎となる IP アドレスに関するセキュリティ ポリシーを適用します。

各タグの最大長は *127* 文字です。タグが最大文字数を超えると切り捨てられます。*2* つの切り捨てられたタグが同じ場合、タグに一意のハッシュが追加され、タグが互いに区別されます。

Kubernetes プラグインを使用すると、Kubernetes クラスタ内にデプロイされたポッド、ノード、名前空間、およびサービスに関する IP アドレスとタグのマッピングを、そのクラスタに CN-Series ファイアウォールがデプロイされていなくても、物理または VM-Series ファイアウォールに配布することができます。

| 事前に定義されたタグ | Panorama 上のタグの形式 | 収集される IP アドレス |
|---|---|---|
| DaemonSet | k8s.cl_<cluster-name>.ns_<namespace>.ds_<pod-name> | ポッド IP アドレス |
| ReplicaSet | k8s.cl_<cluster-name>.ns_<namespace>.rs_<pod-name> | ポッド IP アドレス |
| StatefulSet | k8s.cl_<cluster-name>.ns_<namespace>.ss_<pod-name> | ポッド IP アドレス |
| サービス | k8s.cl_<cluster-name>.ns_<namespace>.svc_<svc-name> | クラスタ IP アドレス<br>ポッド IP アドレス |
| 外部サービス | k8s.cl_<cluster-name>.ns_<namespace>.exsvc_<svc-name> | 外部サービス IP アドレス<br>LoadBalancer IP アドレス |
| ノード | k8s.cl_<cluster-name>.nodes | すべてのノードのプライベート IP アドレス |
| 外部ノード | k8s.cl_<cluster-name>.ex_nodes | すべてのノードのパブリック IP アドレス |
| 名前空間 | k8s.cl_<cluster-name>.ns_<namespace> | 名前空間内のすべてのクラスタ IP アドレス<br>名前空間内のすべてのポッド IP アドレス |
| インターフェイス | • k8s.cl_<cluster-name>.ns_<namespace>.ds_<daemonset-name>.if_<interface><br>• k8s.cl_<cluster-name>.ns_<namespace>.rs_<replicaset-name>.if_<interface><br>• k8s.cl_<cluster-name>.ns_<namespace>.ss_<statefulset-name>.if_<interface> | デプロイメント内の各ポッド上のすべてのインターフェースのすべての IP アドレス。 |

Kuberbetes クラスタ内のポッドとサービスを整理するためにラベルを使用している場合は、Panorama 上 の Kubernetes Plugin がこれらのラベルを問い合わせ、ユーザーの代わりにタグを作成することができます。以下のユーザー定義のラベルがサポートされています。

| ユーザー定義のタグ | Panorama 上のタグの形式 | 収集される IP アドレス |
|---|---|---|
| ラベル | k8s.cl_<cluster-name>.ns_<namespace>.<label-key>.<label-value> | 指定されたラベルと一致する名前空間内のすべてのクラスタ IP アドレス。<br>指定されたラベルと一致する名前空間内のすべてのポッド IP アドレス。 |
| ラベル セレクタ | k8s.cl_<cluster-name>.<selector-name> | 指定されたセレクタと一致するすべてのクラスタ IP アドレス。<br>指定されたセレクタと一致するすべてのポッド IP アドレス。 |

ラベル セレクタは、Kubernetes クラスタ内のポットとサービスに対して指定されたラベルを照合し、そのラベルに関連付けられた IP アドレスを 1 つのタグにマッピングします。Kubernetes プラグインは、ラベル キーとラベル値の集合ベースと等価ベースのセレクタをサポートしています。

以下の等価ベースのセレクタがサポートされています。

- `key = value; key ==`
- `value; key != value (例: app = redis)`

1 つの式で複数のセレクタをコンマ区切りリストとして指定することもできます。以下に例を示します。

```
app == web, tier != backend
```

以下の集合ベースのセレクタがサポートされています。

- key in (value1, value2)
- key notin (value1, value2) (例: tier notin (frontend, backend))
- key
- !key

監視対象のサービス オブジェクトでは、プラグインが、以下の命名体系を使用して、port、targetPort、および nodePort サービス オブジェクト用のポートを生成します。

```
<namespace>-<svc_name>-<type>-<port_value>-<hash>
```

k8s クラスタ全体で名前空間とサービス名が重複している場合でも、サービス オブジェクトは一意であることがハッシュによって保証されます。

# タグ付けされた VLAN トラフィックの検査を有効化する

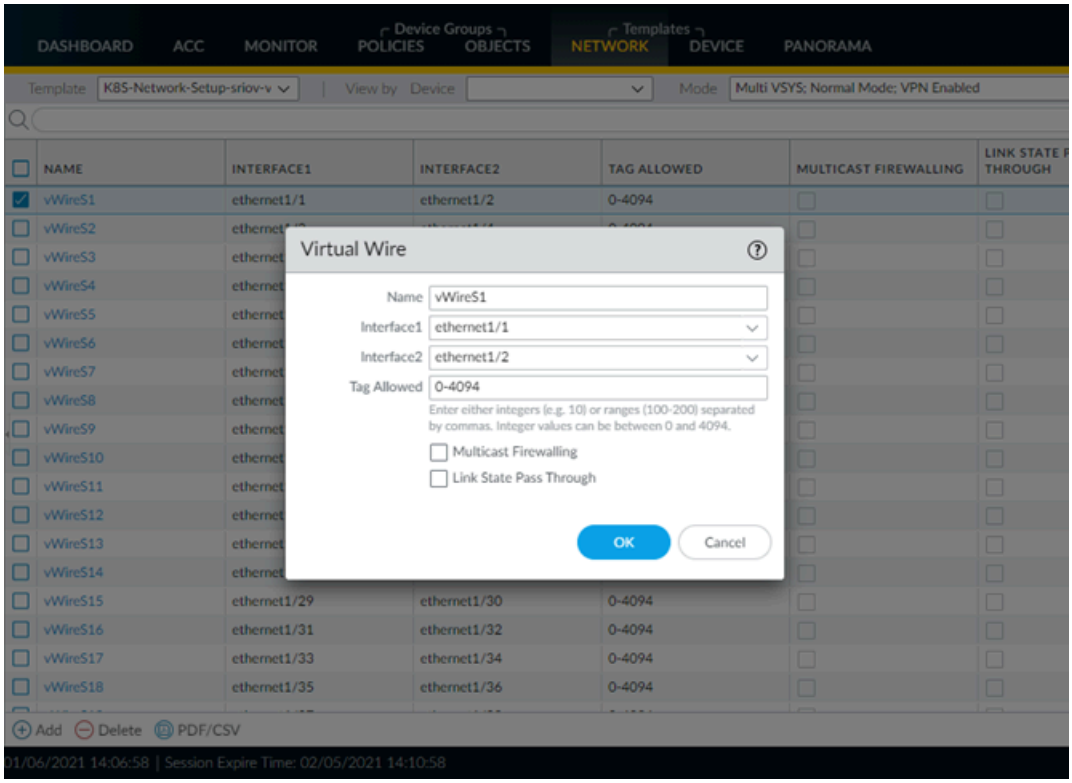
| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

以下の手順を実行して、CN-Series ファイアウォールでタグ付き VLAN トラフィックを検査できるようにします。VLAN タグ付きトラフィックを検査するには、すべての VLAN タグを許可するように Panorama 上のすべてのバーチャル ワイヤの設定を更新する必要があります。その後で、アプリケーション ポッド インターフェースに VLAN タグを割り当てるように、アプリケーション ポッド YAML ファイルに注釈を付ける必要があります。この注釈は、ファイアウォール経由で送信されるパケットに適用されるタグを CN-NGFW に指示します。

二重 *VLAN* タグ付けはサポートされていません。

**STEP 1 |** CN-NGFW のすべてのインターフェース上ですべての VLAN を有効にします。

1. Panorama にログインします。
2. [ネットワーク] > [バーチャル ワイヤ] を選択します。
3. [テンプレート] ドロップダウンから **[K8S-Network-Setup]** テンプレートを選択します。
4. 1 つ目のバーチャル ワイヤを選択します。
5. [タグを許可] を 0-4094 に設定します。
6. 各バーチャル ワイヤに対してこの手順を繰り返します。
7. 変更を **Commit (コミット)** します。




**STEP 2 |** アプリケーション ポッド YAML ファイルに以下の注釈を付加して、インターフェースごとに静的 VLAN ID を適用します。

インターフェースごとにサポートされる *VLAN* タグは *1* つだけです。

paloaltonetworks.com/interfaces: '[ {"name" : "eth0"}, {"name" : "net1", "vlan" : <VLAN-ID> } {"name" : "net2", "vlan" : <VLAN-ID> }]'

例:

```
 annotations: k8s.v1.cni.cncf.io/networks: bridge-conf-1,bridge-
conf-2,bridge-conf-0,pan-cni paloaltonetworks.com/firewall: pan-fw
 paloaltonetworks.com/interfaces: '[ {"name" : "eth0"}, {"name" :
```

```
"net1", "vlan" :101 }, {"name" : "net2", "vlan" :102 }, {"name" :
"net3", "vlan" :103 } ]'
```

# IPVLAN を有効にする

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.1.x or above Container Images<br>• PanoramaPAN-OS 10.1.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

IPVLAN は、ホストネットワークにアクセスするためにコンテナ化された環境で使用できる仮想ネットワーキングデバイス用のドライバです。L2 モードでは、IPVLAN は、ホストネットワーク内で作成された IPVLAN デバイスの数に関係なく、単一の MAC アドレスを外部ネットワークに公開します。すべての論理 IP インターフェースは同じ MAC アドレスを使用します。これにより、親 NIC で無差別モードを使用することを回避でき、NIC またはスイッチで MAC 制限される可能性を防止できます。

CN-Series ファイアウォールで IPVLAN を使用できるようになりました。ただし、以下の制限があります。

- PAN-OS 10.1.2 以降が必要
- IPv4 のみ
- L2 モードのみ
- インターフェースごとに 1 つの IP アドレス
- Multus を使用している場合は、**pan-cni.yaml** ではなく **pan-cni-multus.yaml** を展開します。さらに、pan-cni-net-attach-def.yaml を、Multus アプリケーションポッドがデプロイされているすべての名前空間にデプロイする必要があります。

同じホスト（同じ親インターフェースを共有）で *IPVLAN* 子インターフェース通信は機能しません。

IPVLAN を有効にするには、アプリケーションポッドの yaml ファイルに注釈を付ける必要があります。IPVLAN を有効にするために CN-Series の yaml ファイルに変更を加える必要はありません。次に、IPVLAN のネットワークアタッチメント定義の例を示します。モードは「**l2**」に設定されていることに注意してください。CN-Series ファイアウォールは L2 モードだけをサポートします。

```
cat ipvlan-nw-10.yaml apiVersion: "k8s.cni.cncf.io/v1"
 kind:NetworkAttachmentDefinition metadata: name: ipvlan-conf-10
 spec: config: '{ "cniVersion":"0.3.0", "name": "ipvlan-conf-10",
 "type": "ipvlan", "master": "eth1", "mode": "l2", "ipam": { "type":
 "static", "addresses": [ { "address":"10.154.102.89/24" } ] } }'
```

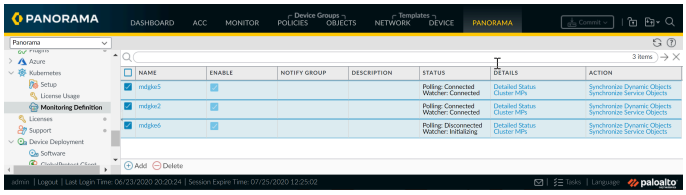
# Kubernetes Plugin on Panorama をアンインストールする

以下のワークフローを使用して、Kubernetes plugin on Panorama をアンインストールし、すべてのトークンを正常に Palo Alto Networks ライセンス サーバーに返せるようにしてから、認証コードをクリアします。このワークフローを使用すれば、トークンを他の Panorama で使用できるようにすることができます。Panorama 管理サーバーを高可用性設定でデプロイしている場合は、アクティブ-プライマリ Panorama 上でこの手順を実行してから、パッシブ-プライマリ Panorama ピアに移動する必要があります。

**STEP 1 |** HA 設定でデプロイしている場合は、アクティブ-プライマリ Panorama ピアにログインします。

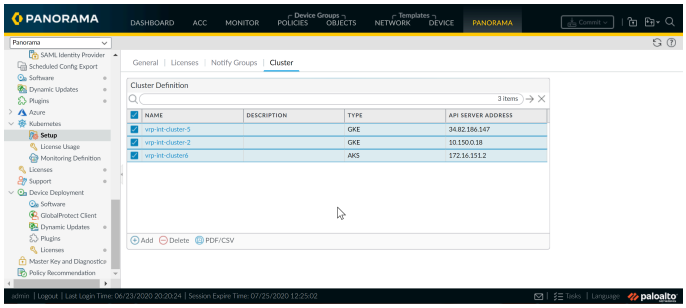
1. プラグインからすべてのクラスタ設定を削除します。
   1. モニタリング定義を削除します。

      [プラグイン] > **[Kubernetes]** > **[Monitoring Definition (**モニタリング定義**)]** を選択し、モニタリング定義と [削除] を選択します。

   2. Kubernetes クラスタ定義を削除します。

      [プラグイン] > **[Kubernetes]** > [セットアップ] > [クラスタ] を選択し、クラスタ定義と [削除] を選択します。

2. Panorama 上の変更内容をコミットします。

   [コミット] > **[Panorama** へのコミット**]** を選択します。
3. 使用済みトークンのカウントが 0 であることを確認します。

   すべてのトークンがライセンス サーバーに返されることを確認するには、以下の手順を実行します。
4. 認証コードのクリアを実行し、ライセンス列の認証コードが [なし] になっていることを確認します。
5. 設定を削除して、変更内容をコミットします。
   1. [プラグイン] を選択して、インストールした Kubernetes プラグイン バージョンを探し、**[Remove Config (**設定の削除**)]** を選択します。
   2. [コミット] > **[Panorama** へのコミット**]** を選択します。
6. Kubernetes プラグインをアンインストールします。
7. アクティブな Panorama ピアをサスペンド状態にします。

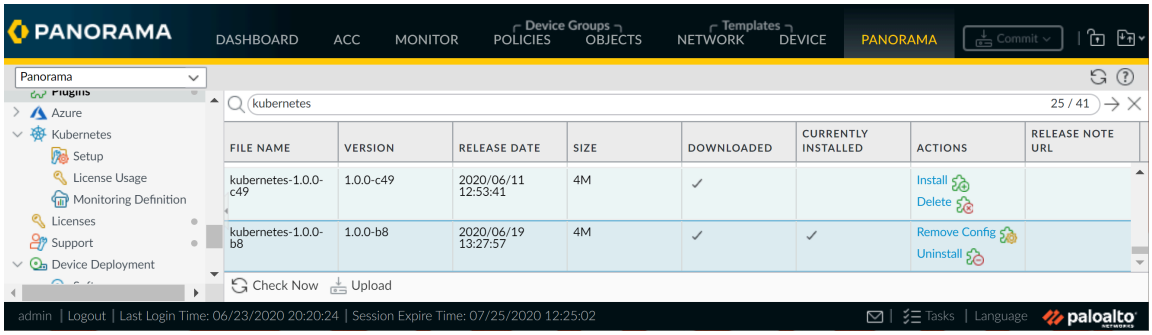
   **[Panorama]** > [高可用性] を選択してから、[操作コマンド] セクションで **[Suspend local Panorama (**ローカル **Panorama** をサスペンド**)]** リンクをクリックします。

**STEP 2 |** 他の Panorama ピアにログインします。

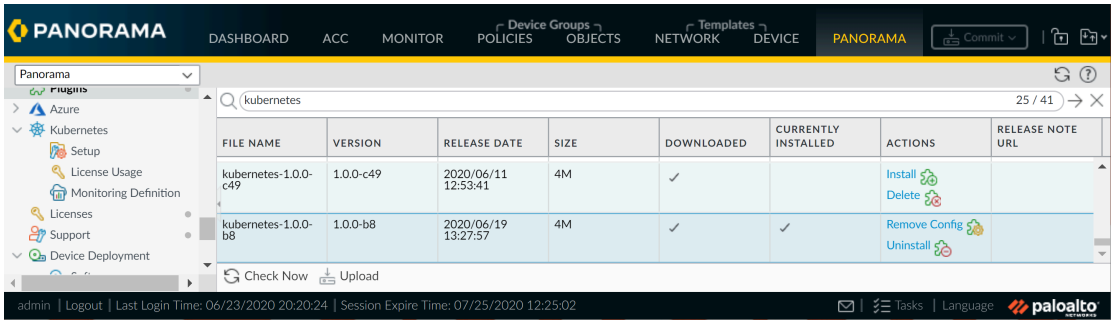
現時点では、このピアがアクティブ-セカンダリ ピアです。

1. [プラグイン] を選択して、インストールした Kubernetes プラグイン バージョンを探し、**[Remove Config (設定の削除)]** を選択します。




2. プラグインをアンインストールします。

   1. [プラグイン] を選択して、インストールした Kubernetes プラグイン バージョンを探し、**[Uninstall (アンインストール)]** を選択します。




   2. アンインストールが成功したことを確認します。

## Panorama 上で CN-Series ファイアウォールの認証コードをクリアする

以下の回避策は、プラグライン設定を削除し、変更内容をコミットしてから、認証コードをクリアした場合にのみ使用してください。この回避策を使用すれば、トークンをライセンス サーバーに解放して戻し、他の Panorama アプライアンスで使用できるようにすることができます。

**STEP 1 |** 1.新しいプラグイン ユーザーを追加して、変更内容をコミットします。

1. **Panorama** > **Administrators** [管理者]
2. **__kubernetes** という名前の新しいユーザーを追加します。
3. [コミット] > **[Panorama** へのコミット] を選択します。

**STEP 2 |** Panorama 上で認証コードをクリアします。

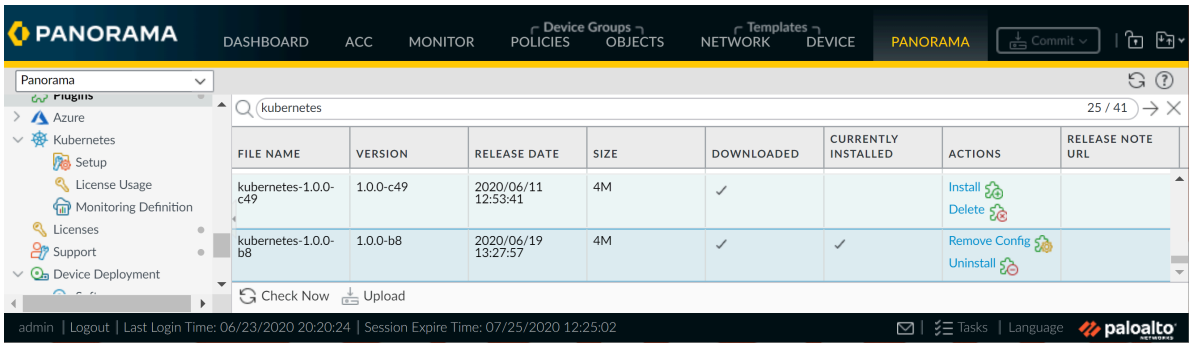
1. **[Panorama]** > [プラグイン] > **[Kubernetes]** > [セットアップ] > [ライセンス] を選択します。
2. **[Activate/update using authorization code (認証コードを使用してアクティベート/更新)]** と **[Clear Auth Code (認証コードをクリア)]** を選択します。
3. ライセンス列に認証コード [なし] と表示されていることを確認します。

**STEP 3 |** ステップ 1 で作成したプラグイン ユーザー **__kubernetes** を削除します。

**STEP 4 |** 変更をコミットします。

**STEP 5 |** プラグインをアンインストールします。

1. [プラグイン] を選択して、インストールした Kubernetes プラグイン バージョンを探し、**[Uninstall (アンインストール)]** を選択します。




2. アンインストールが成功したことを確認します。

# CN-Series でサポートされていない機能

PAN-OSでサポートされている次の機能は、以下で特に明記されていない限り、CN-Series では使用できません。

| 機能 | DaemonSet | K8s サービス | CNF モード | HSF モード |
|---|---|---|---|---|
| 認証 | いいえ | いいえ | いいえ | いいえ |
| Cortex Data Lake へのログ | いいえ | いいえ | いいえ | いいえ |
| Enterprise DLP | いいえ | いいえ | いいえ | いいえ |
| Non-vWire インターフェース | いいえ | なし | あり | あり。 |
| IoTセキュリティ | いいえ | いいえ | いいえ | いいえ |
| IPv6 | あり。 | なし | あり | いいえ |
| NAT | いいえ | なし | あり | いいえ |
| ポリシー ベース フォワーディング | いいえ | なし | あり | いいえ |
| QoS | いいえ | いいえ | いいえ | いいえ |
| SD-WAN | いいえ | いいえ | いいえ | いいえ |
| User-ID | いいえ | なし | あり | いいえ |
| WildFire インライン ML | いいえ | いいえ | いいえ | いいえ |
| SaaS インライン | いいえ | いいえ | いいえ | いいえ |
| IPSec | いいえ | いいえ | いいえ | いいえ |
| Tunnel Content Inspection（トンネル コンテンツ検査） | いいえ | いいえ | いいえ | なし |

# CNシリーズ ファイアウォールの高可用性とDPDKサポート

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

高可用性（HA）とは、ネットワーク上の単一障害点を回避するために、2 つのファイアウォールを 1 つのグループに配置して、この 2 つの設定を同期させる設定です。ファイアウォール ピア間のハートビート接続では、ピアがダウンした場合シームレスにフェイルオーバーを実行できます。2 つのデバイス クラスターでファイアウォールを設定すると冗長性が得られるため、ビジネス継続性を確保できます。

この章では、以下のセクションについて説明します。

- Kubernetes CNF としての CN-Series ファイアウォールの高可用性サポート
- AWS EKS におけるCN-Series ファイアウォールの高可用性
- CN-Series ファイアウォール上で DPDK を設定する

# Kubernetes CNF としての CN-Series ファイアウォールの高可用性サポート

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

高可用性（HA）とは、ネットワーク上の単一障害点を回避するために、2 つのファイアウォールを 1 つのグループに配置して、この 2 つの設定を同期させる設定です。ファイアウォール ピア間のハートビート接続では、ピアがダウンした場合シームレスにフェイルオーバーを実行できます。2 つのデバイス クラスターでファイアウォールを設定すると冗長性が得られるため、ビジネス継続性を確保できます。

CN-series-as-a-kubernetes-CNFを HA にデプロイできるようになりました。このデプロイメントモードでは、セッションと構成の同期を伴うアクティブ/パッシブ HA のみがサポートされます。

CN-Series-as-a-Kubernetes CNF を HA にデプロイすると、アクティブ ノードとパッシブ ノード用にそれぞれ 2 つの PAN-CN-MGMT-CONFIGMAP、PAN-CN-MGMT、および PAN-CN-NGFW YAML ファイルができます。

レイヤ 3 をサポートする HA でCN-Series ファイアウォールを Kubernetes CNF として正常にデプロイするには:

- HA では、各 Kubernetesノードに少なくとも 3 つのインターフェイスが必要です。管理 (デフォルト)、HA2 インターフェース、およびデータインターフェース。
- L3モードの CN-Series ファイアウォールの場合、少なくとも2つのインターフェイスが必要です。管理 (デフォルト) とデータインターフェイス。

3 items

| INTERFACE | TEMPLATE | INTERFACE TYPE | MANAGEMENT PROFILE | IP ADDRESS | VIRTUAL ROUTER | TAG | VLAN / VIRTUAL-WIRE | VIRTUAL SYSTEM | SECURITY ZONE | SD-WAN INTERFACE PROFILE | UPSTREAM NAT | FEATURES | COMMENT |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Slot 1 | | | | | | | | | | | | | |
| ethernet1/1 | K8S-Network-Setup-V3 | HA | | none | none | Untagged | none | none | none | | Disabled | | ha |
| ethernet1/2 | K8S-Network-Setup-V3 | Layer3 | ping | Dynamic-DHCP Client | vr1 | Untagged | none | vsys1 | trust | | Disabled | | |
| ethernet1/3 | K8S-Network-Setup-V3 | Layer3 | ping | Dynamic-DHCP Client | vr1 | Untagged | none | vsys1 | untrust | | Disabled | | |

- 新しいネットワーク接続定義 YAML ファイルに次の変更を加えます。
  - 以下の YAML ファイルで PAN_HA_SUPPORT パラメータ値が**true** である ことを確認してください。

    ```
    pan-cn-mgmt-configmap-0.yaml
    ```

    ```
    pan-cn-mgmt-configmap-1.yaml
    ```

  - 次のコマンドを実行して、ハイパーバイザー・インターフェースから **pciBusID** 値を取得します。

    ```
    ethtool -i interface name
    ```

    上記で取得した **pciBusID** 値を次のネットワーク定義ファイルに追加します。

    ```
    net-attach-def-1.yaml
    ```

    ```
    net-attach-def-2.yaml
    ```

    ```
    net-attach-def-3.yaml
    ```

    ```
    net-attach-def-ha2-0.yaml
    ```

    ```
    net-attach-def-ha2-1.yaml
    ```

  - AWS コンソール上の対応するノードインスタンスから HA2 インターフェイスの静的 IP アドレスを取得し、それを `net-attach-def-ha2-0.yaml` および `net-attach-def-ha2-1.yaml`ファイルのアドレス パラメータに追加します 。

高度なルーティング を使用している場合は、CNF モードでデプロイされた CN シリーズのファイアウォールは EKS とオンプレミス環境でのみサポートされていることを考慮してください。Kubernetes 3.0.0 プラグインで高度なルーティングを使用している場合は、テンプレートスタックで手動で設定する必要があります。`pan-cn-mgmt-console.yaml` ファイルで、`PAN_ADVANCED_ROUTING:”true”.`というフラグを設定してください。

# AWS EKS におけるCN-Series ファイアウォールの高可用性

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

これで、CN-Series-as-a-Kubernetes-CNF を HA にデプロイできるようになりました。このデプロイメントモードでは、セッションと構成の同期を伴うアクティブ/パッシブ HA のみがサポートされます。

*IPV6によるHAでのCN-Series-as-a-Kubernetes CNFデプロイメントはAWS環境ではサポートされていません。*

冗長性を確保するため、アクティブ/パッシブの高可用性（HA）設定でCN-Series ファイアウォールをAWS にデプロイできます。アクティブ ピアは、まったく同一に構成されているパッシブ ピアと絶え間なく設定とセッション情報を同期させています。2 台のデバイス間のハートビート接続により、アクティブなデバイスがダウンした場合のフェイルオーバーが可能です。CN-Series ファイアウォールは、セカンダリIP移動を通じて HA のAWS EKSにデプロイできます。

インターネットに接続するアプリケーションへのすべてのトラフィックがこのファイアウォールを通過するようにするために、AWS ingress ルーティングを設定できます。AWS 入口ルーティング機能により、ルート テーブルを AWS インターネット ゲートウェイと関連付け、アプリケーション トラフィックを CN-Series ファイアウォール経由でリダイレクトするルーティング ルールを追加することができます。このリダイレクトにより、アプリケーションのエンドポイントを再設定しないでも、すべてのインターネット トラフィックがファイアウォールを通過します。

セカンダリ移動

アクティブピアがダウンした場合、パッシブピアがその障害を検出してアクティブに変化します。また、AWSインフラにAPI呼び出しを行い、設定されているセカンダリIPアドレスを、障害が発生したピアのデータプレーンインターフェイスから自己へと移動します。さらに、トラフィックがアクティブなファイアウォールインスタンスに送信されるように、AWSがルートテーブルを更新します。これらの2つの処理により、インバウンド トラフィックとアウトバウンド トラフィックセッションがフェイルオーバー後に復元されます。このオプションを使用すると、DPDK を利用して CN-Series ファイアウォールインスタンスのパフォーマンスを向上させることができます。

# HA 用の IAM ロール

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

AWSでは、AWS自身が発行する認証情報を使用して、すべてのAPI要求が暗号化署名されている必要があります。HA ペアとしてデプロイされる CN-Series ファイアウォールの API 権限を有効化するには、ポリシーを作成し、そのポリシーをAWS Identity and Access Management (IAM) サービス内にロールとして添付する必要があります。このロールは、起動時に CN-Series ファイアウォールにアタッチする必要があります。このポリシーにより、フェイルオーバーが発生した場合に、アクティブなピアからパッシブピアにインターフェイスまたはセカンダリIPアドレスを移動するために必要な、APIアクションを開始するためのアクセス許可がIAMロールに与えられます。

ポリシー作成の詳細な手順については、AWS ドキュメントのカスタマー管理ポリシーの作成を参照してください。IAM ロールの作成、ロールを割り当てるアカウントやAWSサービスの定義、ロールを割り当てたアプリケーションが使用してよいAPIアクションやリソースの定義の詳細な手順については、AWSドキュメントの IAM Roles for Amazon EC2（Amazon EC2用のIAMロール）を参照してください。

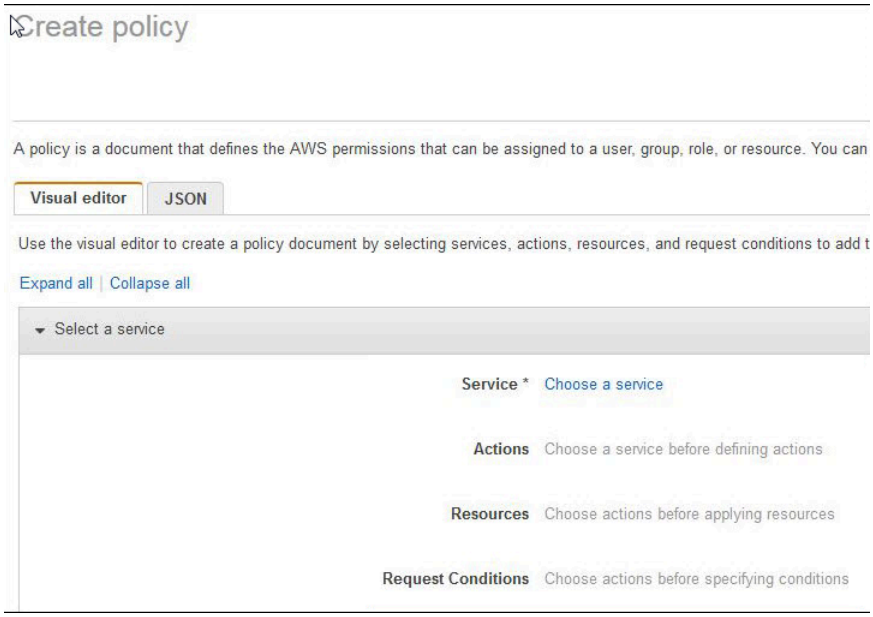
AWS コンソールで設定される IAM ポリシーは、以下のアクションおよびリソースを利用する際は、最低でも許可の取得を必要とします。

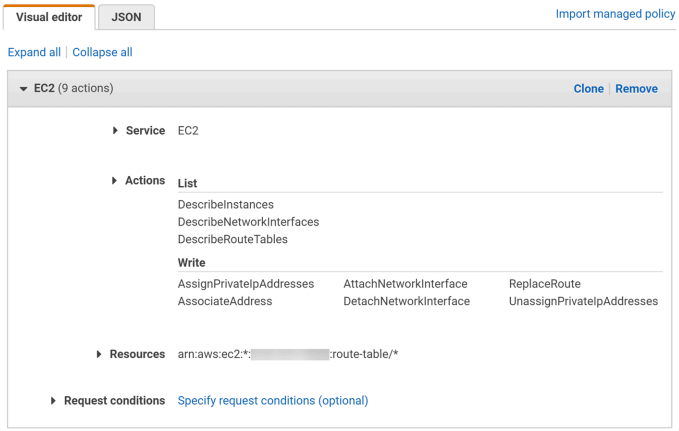
HAを有効にするには、次のIAMアクション、アクセス許可、およびリソースが必要になります。

| IAMアクション、アクセス許可、またはリソース | の意味 | Secondary IP Move（セカンダリIP移動） |
| --- | --- | --- |
| AttachNetworkInterface | ENIをインスタンスに取り付ける際のアクセス許可。 | ✓ |
| DescribeNetworkInterfaces | インスタンスにインターフェイスを取り付けるためにENIパラメータを取得する許可。 | ✓ |
| DetachNetworkInterface | ENIをEC2インスタンスから取り外す許可。 | ✓ |
| DescribeInstances | VPC内のEC2インスタンスについて情報を取得するための許可。 | ✓ |

| IAMアクション、アクセス許可、またはリソース | の意味 | Secondary IP Move（セカンダリIP移動） |
|---|---|---|
| AssociateAddress | プライマリIPアドレスに関連するパブリックIPアドレスを、パッシブからアクティブインターフェイスに移動するための許可。 | ✓ |
| AssignPrivateIpAddresses | セカンダリIPアドレスと関連するパブリックIPアドレスを、パッシブピア上のインターフェイスに割り当てるための許可。 | ✓ |
| DescribeRouteTables | CN-Series ファイアウォールインスタンスに関連するすべてのルートテーブルを取得するためのアクセス許可。 | ✓ |
| ReplaceRoute | AWSルートテーブルエントリを更新するための許可。 | ✓ |
| GetPolicyVersion | AWSポリシーバージョン情報を取得するための許可。 | ✓ |
| GetPolicy | AWSポリシー情報を取得するための許可。 | ✓ |
| ListAttachedRolePolicies | 指定したIAMロールにアタッチされている、すべての管理対象ポリシーのリストを取得するための許可。 | ✓ |
| ListRolePolicies | 指定したIAMロールに埋め込まれている、インラインポリシー名のリストを取得するための許可。 | ✓ |
| GetRolePolicy | 指定したIAMロールに埋め込まれている、特定のインラインポリシーを取得するための許可。 | ✓ |
| policy | IAMポリシーAmazonリソースネーム（ARN）にアクセスするための許可。 | ✓ |
| role | IAMロールARNにアクセスするための許可。 | ✓ |
| route-table | ルートテーブルAmazonリソースネーム（ARN）にアクセスして、フェイルオーバー時にそれを更新するための許可。 | ✓ |
| ワイルドカード（*） | ARNフィールドで、*をワイルドカードとして使用します。 | ✓ |

以下のスクリーンショットは、セカンダリIP HA用に、上記のIAMロールのアクセス管理設定を表しています。




セカンダリIP移動HA用に最低限必要なアクセス許可: {"Version":"2012-10-17","Statement": [{"Sid":"VisualEditor0","Effect":"Allow","Action": ["ec2:AttachNetworkInterface","ec2:DetachNetworkInterface","ec2:DescribeInstances","ec2:DescribeNetwork “ec2:AssignPrivateIpAddresses”,“ec2:AssociateAddress”,“ec2:DescribeRouteTables”],"Resource": "*"}{"Sid":"VisualEditor1","Effect":"Allow","Action": "ec2:ReplaceRoute", "Resource": "arn:aws:ec2:*:*:route-table/*"}]}

# HA リンク

| どこで使用できますか? | 何が必要ですか? |
|---|---|
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している |

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| | • Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

HA ペアの各デバイスは、HA リンクを使用してデータを同期し、状態情報を管理します。AWS では、CN-Series ファイアウォールは次のポートを使用します。

- **Control Link**（コントロールリンク） - HA1 リンクは、Hello、ハートビート、HA の状態情報、ルーティング用管理プレーン同期などの情報交換に使用されます。またこのリンクを使用して、アクティブ デバイスまたはパッシブ デバイスの設定変更をピア デバイスと同期します。

  管理ポートはHA1が使用します。クリアテキスト通信には TCP ポート 28769 と 28260、暗号化通信（SSH over TCP）にはポート 28 を使用します。

- **Data Link**（データリンク） - HA2リンクを使用して、セッションの同期、テーブルの転送、IPSec SA、およびARPテーブルをHAペアのデバイス間で同期します。HA2 リンクのデータ フローは（HA2 キープアライブを除き）常に単向性なので、データはアクティブ デバイスからパッシブ デバイスに流れます。

  ethernet1/1 は HA2 リンクとして割り当てる必要があります。これは、AWS 上の CN-Series ファイアウォールを HA で展開するために必要です。HA データリンクは、IP（プロトコル番号 99）または UDP（ポート 29281）のいずれかを転送ポートとして使用するように設定できます。

AWS上 の CN-Series ファイアウォールは、HA1 あるいは HA2 のバックアップリンクをサポートしていません。

## ハートビートポーリングおよび Hello メッセージ

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

ファイアウォールでは、Hello メッセージおよびハートビートを使用して、ピアの応答状態と動作状態を確認します。設定した *Hello* 間隔で Hello メッセージがピアの一方からもう一方へ送信され、デバイスの状態を検証します。ハートビートは、コントロール リンクを介した HA ピアに対する ICMP ping の一種で、ピアがこの ping に応答することで、デバイスの接続および応答状態を証明します。フェイルオーバーをトリガーする HA タイマーの詳細は、HA タイマーを参照してください。（CN-Series ファイアウォール用の HA タイマーは、PA-5200 Series ファイアウォールと同じものです）。

# デバイス優先度およびプリエンプション

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

HA ペアのデバイスにデバイス優先度の値を割り当てることにより、フェイルオーバー発生時にどちらのデバイスにアクティブな役割を持たせて優先的にトラフィックを管理させるかを示すことができます。HA ペアの特定のデバイスを使用してアクティブにトラフィックを保護する必要がある場合、両方のファイアウォールでプリエンプティブ機能を有効にし、各デバイスに優先度の値を割り当てる必要があります。数値の小さい方のデバイス、つまり優先度の高いデバイスがアクティブ デバイスに指定され、ネットワーク上のすべてのトラフィックを管理します。もう一方のデバイスはパッシブ状態となり、アクティブ デバイスと設定情報や状態の情報を同期し、障害が発生した場合のアクティブ状態への移行に備えます。

小さい方の数値は、最初のデプロイメント時にアクティブになります。大きい方の数値が最初にデプロイされ、プリエンプションが無効になっている場合、大きい方の数値がアクティブになります。

*AWS* 上の *CN-Series* ファイアウォール内の *HA* には、プリエンプションは推奨されません。

デフォルトでは、ファイアウォールではプリエンプションは無効になっています。プリエンプティブの動作を有効にすると、優先度の高い（数値の低い方の）ファイアウォールが障害から回復した後に、そのファイアウォールをアクティブ ファイアウォールとして再開させることができます。プリエンプションが発生すると、そのイベントがシステム ログに記録されます。

優先順位を追加するには、`pan-cn-mgmt-configmap-0.yaml` および `pan-cn-mgmt-configmap-1.yaml` ファイルで、パラメータ値 `PAN_HA_PRIORITY` が数値に設定されていることを確認する必要があります。

以下に例を示します。

```
PAN_HA_PRIORITY:“10”
```

# HA タイマー

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images |

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| | • PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version clientHelmを使用したCNシリーズのデプロイメント用 |

高可用性（HA）タイマーは、ファイアウォール障害の検出およびフェイルオーバーのトリガーに使用します。HAタイマーの設定時に煩雑さを軽減するため、以下3種類のプロファイルから選択できます。**Recommended** [推奨]、**Aggressive** [アグレッシブ] および **Advanced** [高度]これらのプロファイルでは、特定のファイアウォール プラットフォームに最適な HA タイマー値が自動入力され、HA の導入速度を高めることができます。

通常のフェイルオーバー タイマー設定には **Recommended**（推奨）プロファイルを使用し、高速なフェイルオーバー タイマー設定には **Aggressive**（アグレッシブ）プロファイルを使用します。**Advanced**（詳細）プロファイルでは、ネットワーク要件に合わせてタイマー値をカスタマイズできます。

| AWS 上のCN-Series の HA タイマー | 推奨/アグレッシブプロファイルの初期設定値 |
| --- | --- |
| プロモーションホールドタイム | 2000/500 ms |
| Hello間隔 | 8000/8000 ms |
| ハートビート間隔 | 2000/1000 ms |
| 最大フラップ数 | 3/3 |
| プリエンプションホールドタイム | 1/1 分 |
| モニター障害時ホールドアップタイム | 0/0 ms |
| 追加のマスターホールドアップタイム | 500/500 ms |

# セカンダリ IP を使用して AWS EKS 上でアクティブ/パッシブ HA を設定する

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images |

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| | • PanoramaPAN-OS 10.2.x以降のバージョンを実行している |

新しい CN-Series ファイアウォールを、セカンダリIPアドレスを持つ HA ペアとしてデプロイするには、以下の作業を行ってください。

**STEP 1 |** HA ペア向けに CN-Series ファイアウォールをデプロイする前に、以下の事項を確認してください。

- 両方のHAピアを同じAWSアベイラビリティゾーンの中にデプロイしている。HA 用の IAM ロールを参照してください。
- インスタンスをデプロイする際に、IAM ロールを作成し、CN-Series ファイアウォールを実行しているワーカーノードにロールを割り当てます。
- アクティブファイアウォールとパッシブファイアウォールには、それぞれ少なくとも3つのインターフェイス（管理インターフェイス、HA2 インターフェイス、およびデータインターフェイス）が必要です。

  デフォルトでは、管理インターフェースが HA1インターフェースとして使用されます。
- クラスタと同じアベイラビリティゾーンで AWS にネットワークインターフェースを作成します。`eni` にタグを追加して、AWSによって管理されず、multusで使用できるようにします。

  `node.k8s.amazonaws.com/no_manage:真`
- ネットワーク コンポーネントとセキュリティ コンポーネントが適切に定義されていることを確認します。
  - インターネットとの通信を有効化します。デフォルトの VPC にはインターネット ゲートウェイが用意されていますので、デフォルトのサブネットで CN-Series ファイアウォールをインストールすれば、インターネットにはアクセスできる状態になっています。
  - サブネットを作成します。サブネットは、EC2 インスタンスを起動できる VPC に割り当てられた IP アドレス範囲のセグメントです。CN-Series ファイアウォールがインターネットにアクセスできるように設定するには、パブリックサブネットに属している必要があります。
  - ファイアウォールデータインターフェイスを含む、データセキュリティグループを作成します。また、すべてのトラフィックを許可するようにセキュリティを設定し、ファイアウォールによってセキュリティを適用するようにします。フェイルオーバー時に既存のセッションを維持するために、この作業が必要になります。

- プライベート サブネットのルート テーブルにルートを追加し、該当する VPC 内のセブネットおよびセキュリティグループ全体に確実にトラフィックをルーティングできるようにします。

*CN-Series* ファイアウォールを*EKS*にデプロイする場合、*`http-put-response-hop-limit`* 値がデフォルト値の*1*に設定されていると、*IMDSv2* トークンの取得は失敗します。*IMDSv2* が有効になっている場合は、ホップ制限値が *3* 以上に設定されていることを確認する必要があります。

以下に例を示します。

以下のコマンドを実行します：

```
aws ec2 modify-instance-metadata-options --instance-id
<your-instance-id> --http-tokens required --http-endpoint
enabled --http-put-response-hop-limit 3
```

**STEP 2 |** CNシリーズ ファイアウォールをEKSにデプロイする。

1. 各HAピアのHA2インターフェイスとして、ethernet 1/1を設定します。
   1. Amazon EC2コンソールを開きます。
   2. Network Interface（ネットワークインターフェイス）を選択して、次にネットワークインターフェイスを選択します。
   3. **Actions(アクション) > Manage IP Addresses(IPアドレスの管理)**を選択します。
   4. AWS が動的に IP アドレスを割り当てるか、CN-Series ファイアウォールのサブネット範囲内の IP アドレスを入力するには、フィールドを空白のままにします。これにより、セカンダリIPがHA2 インターフェイスに割り当てられます。
   5. **Yes**（はい）をクリックして、**Update**（更新）を選択します。
   6. アクション > ソース/宛先の変更を選択します。をオンにして 無効化を選択します。
   7. この作業をセカンダリ（パッシブ）HAピアで繰り返します。
2. 最初の（アクティブ）ピア上のデータプレーンインターフェイスに、セカンダリIPアドレスを追加します。
   1. **Network Interface**（ネットワークインターフェイス）を選択して、次にネットワークインターフェイスを選択します。
   2. **Actions(アクション) > Manage IP Addresses(IPアドレスの管理) > IPv4 Addresses(IPv4アドレス) > Assign new IP(新規IPの割り当て)**を選択します。
   3. AWS が動的に IP アドレスを割り当てるか、CN-Series ファイアウォールのサブネット範囲内の IP アドレスを入力するには、フィールドを空白のままにします。
   4. **Yes**（はい）をクリックして、**Update**（更新）を選択します。
3. セカンダリエラスティック（パブリック）IPアドレスを、アクティブピアのUntrustインターフェイスと関連付けます。
   1. **Elastic IPs**（エラスティックIP）を選択し、次に関連付けるエラスティックIPアドレスを選択します。
   2. **Actions(アクション) > Associate Elastic IP(エラスティックIPの関連付け)**を選択します。
   3. **Resource Type**（リソースタイプ）で、**Network Interface**（ネットワークインターフェイス）を選択します。
   4. Elastic IP アドレスを関連付けるネットワークインターフェイスを選択します。
   5. **Associate**（関連付け）をクリックします。
4. アウトバウンドトラフィック検査の場合、next-hop をファイアウォールのトラストインターフェイスとして設定するエントリをサブネットルートテーブルに追加します。
   1. **VPC > Route Tables(ルートテーブル)**を選択します。
   2. サブネットルートテーブルを選択します。
   3. アクション > ルートの編集 > ルートの追加を選択します。
   4. **Destination**（宛先） CIDRブロックまたはIPアドレスを入力します。

5. **Target**（ターゲット）に、ファイアウォールTrustインターフェイスのネットワークインターフェイスを入力します。
6. **Save routes**（ルートの保存）をクリックします。

5. AWSイングレスルーティングを使用するには、ルートテーブルを作成して、それにインターネットゲートウェイを関連付けます。次に、アクティブファイアウォールのアントラストインターフェイスとして設定されせているnext-hopを持つエントリを追加します。
   1. **Route Tables(ルート テーブル) > Create Route Table(ルート テーブルを作成)**を選択します。
   2. （任意）ルートテーブル用の、分かりやすい**Name tag**（名前タグ）を入力します。
   3. 作成をクリックします。
   4. ルートテーブルをクリックして、**Actions(アクション) > Edit edge associations(エッジ関連付けの編集)**を選択します。
   5. **Internet gateways**（インターネットゲートウェイ）を選択して、VPCインターネットゲートウェイを選択します。
   6. **Save**（保存）をクリックします。
   7. ルートテーブルをクリックして、**Actions(アクション) > Edit routes(ルートの編集)**を選択します。
   8. **Target**（ターゲット）で、**Network Interface**（ネットワークインターフェイス）を選択して、アクティブファイアウォールのUntrustインターフェイスを選択します。
   9. **Save routes**（ルートの保存）をクリックします。

**STEP 3 |** HA を有効にします。

HAサポートを有効にするには、次の YAML ファイルで PAN_HA_SUPPORT パラメータ値が true であることを確認する必要があります。

- `pan-cn-mgmt-configmap-0.yaml`
- `pan-cn-mgmt-configmap-1.yaml`

ピアHA1のIP アドレスは自動構成されます。

**STEP 4 |** AWS コンソール上の対応するノードインスタンスから HA2 インターフェースの静的 IP アドレスを取得し、`net-attachdef-ha2-0.yaml` および `net-attach-def-ha2-1.yaml` ファイルのアドレスパラメータに追加します。

（任意）**HA2 Keep-alive**（**HA2** キープアライブ）パケットの **Threshold**（しきい値）を変更します。初期設定では、ピア間のHA2データリンクをモニタリングするため**HA2 Keep-alive**（**HA2** キープアライブ）が有効化されています。障害が発生し、このしきい値（初期設定は 10000ms です）を超えた場合は、定義されたアクションが実行されます。HA2 キー

プアライブに障害が発生すると、「critical」レベルのシステムログ メッセージが生成されます。

***HA2 keep-alive***（***HA2*** キープアライブ）オプションは、*HA* ペアの両方のデバイス、または一方のデバイスに設定できます。一方のデバイスでこのオプションが有効化されている場合、キープアライブメッセージはそのデバイス単体から送信されます。

**STEP 5 |** ファイアウォールがアクティブ/パッシブ HA でペアになっていることを確認します。

1. 両方のファイアウォールで **Dashboard** にアクセスして、High Availability （高可用性）ウィジットを表示します。
2. アクティブな HA ピアで、**Sync to peer**（ピアに同期）をクリックします。
3. ファイアウォールがペアになっていて同期されていることを確認します。
   - パッシブ ファイアウォール: ローカル ファイアウォールの状態が **Passive**、**Running Config**（実行コンフィグ） が Synchronized と表示されます。
   - アクティブ ファイアウォール: ローカル ファイアウォールの状態が **Active**、**Running Config**（実行コンフィグ） が Synchronized と表示されます。
4. ファイアウォールのコマンドラインインターフェイスから、次のコマンドを実行します:
   - フェイルオーバーの準備状況を確認するには:

     ```
     show plugins vmw_series aws ha state
     ```
   - セカンダリIPマッピングを表示するには:

     ```
     show plugins vm_series aws ha ips
     ```

# CN-Series ファイアウォール上で DPDK を設定する

| どこで使用できますか? | 何が必要ですか? |
| --- | --- |
| • CN-Seriesデプロイメント | • CN-Series 10.2.x or above Container Images<br>• PanoramaPAN-OS 10.2.x以降のバージョンを実行している<br>• Helm 3.6 or above version client |

データプレーン開発キット（DPDK）は、データプレーンアプリケーションでの高速パケット処理のためのシンプルなフレームワークを提供します。

*DPDK*モードは、*CN-Series* ファイアウォールで *Kubernetes Container Network Function*（*CNF*）としてのみサポートされます。

*DHCP IPAM*は、*DPDK* モードではサポートされていません。

システム要件

DPDK アプリケーションを実行するには、ターゲットマシンで次のカスタマイズを行う必要があります。

- カーネル設定-ホスト OS カーネルで HUGETLBFS オプションを有効にします。
- **KNI** および **UIO/VFIO**：ホスト OS カーネルに KNI および UIO/VFIO を挿入します。
- **Hugepages**
  1. **hugepage** を予約する
     - ポッドが起動する前に、実行時に hugepage を予約します。特定のページサイズ（KB 単位）に対応する `/ sys /kernel/` ディレクトリの `nr_hugepages` ファイルに、必要な hugepage 数を追加します。例えば、シングルノードシステムで、2M ページの 1024 が必要な場合は、次のコマンドを使用します。

       ```
       echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/
       nr_hugepages
       ```

     - 起動時に hugepage を予約します。例えば、メモリ 4G の hugepage を 4つの 1G ページとして予約するには、以下のオプションをカーネルに渡す必要があります。

       ```
       default_hugepagesz=1G hugepagesz=1G hugepages=4
       ```

2. **DPDK**で **hugepages** を使用する– PanOS 10.2 は DPDK セカンダリプロセスを使用するため、hugepages のマウントポイントを作成します。

   以下は、DPDK で使用するサイズ 1GB のhugepage を作成するためのサンプルコマンドです。

   ```
   mkdir /mnt/huge mount -t hugetlbfs pagesize=1GB /mnt/huge
   ```

3. 次のコマンドを使用して hugepages を有効にした後、ホストで kubelet サービスを再起動します。

   ```
   sudo systemctl restart kubelet
   ```

4. `/sys/fs/cgroup/hugetlb/kubepods.slice/hugetlb.2MB.limit_in_bytes` をチェックして、サイズが hugepage のサイズと一致していることを確認します。サイズが hugepage サイズと一致しない場合は、次のコマンドを使用してサイズを更新します。

```
echo 2147483648 > /sys/fs/cgroup/hugetlb/kubepods.slice/
hugetlb.2MB.limit_in_bytes
```

ポッドでは、アプリケーションが複数のサイズの事前に割り当てられた *hugepage* を割り当てて消費できます。アプリケーションは、リソース名 *hugepages-<size>*を使用した、コンテナレベルのリソース要件によって *hugepages* を消費します。例えば、*hugepages-2Mi* または *hugepages-1Gi*です。

*CPU*やメモリとは異なり、*hugepages* はオーバーコミットをサポートしていません。

ホストデバイススペースにアクセスするために特権モードが有効になっています。ネットワークデバイスを一覧表示してコンテナにバインドするには、*/sys* をコンテナにマウントして、*DPDK* がディレクトリ内のファイルにアクセスできるようにします。

以下は、*DPDK* で *hugepages* を有効にするためのコードの抜粋です。

```
requests: cpu:"1" memory:"4Gi" hugepages-2Mi:4Gi limits:
 cpu:"1" memory:"4Gi" hugepages-2Mi:4Gi volumeMounts:
 - mountPath: /sys name: sys - mountPath: /dev name:
 dev - mountPath: /dev/shm name: dshm - mountPath: /
run/tmp name: hosttmp - mountPath: /etc/pan-fw-sw
 name: sw-secret envFrom: - configMapRef: name: pan-
ngfw-config-0 env: - name:CPU_REQUEST valueFrom:
 resourceFieldRef: containerName: pan-ngfw-container
 resource: requests.cpu - name:CPU_LIMIT valueFrom:
 resourceFieldRef: containerName: pan-ngfw-container
 resource: limits.cpu - name:MEMORY_REQUEST valueFrom:
 resourceFieldRef: containerName: pan-ngfw-container
 resource: requests.memory - name:MEMORY_LIMIT
 valueFrom: resourceFieldRef: containerName: pan-ngfw-
container resource: limits.memory - name:MY_POD_UUID
 valueFrom: fieldRef: fieldPath: metadata.uid -
 name:MY_NODE_NAME valueFrom: fieldRef: fieldPath:
 spec.nodeName - name:MY_POD_NAME valueFrom: fieldRef:
 fieldPath: metadata.name - name:MY_POD_NAMESPACE
 valueFrom: fieldRef: fieldPath: metadata.namespace
 - name:MY_POD_SERVICE_ACCOUNT valueFrom: fieldRef:
 fieldPath: spec.serviceAccountName - name:MY_POD_IP
 valueFrom: fieldRef: fieldPath: status.podIP volumes:
 - name: sys hostPath: path: /sys - name: dev hostPath:
 path: /dev - name: hosttmp hostPath: path: /tmp/pan -
 name: dshm emptyDir: medium:Memory - name: sw-secret
 secret: secretName: pan-fw-sw
```

- **NUMA** と **CPU**ピニングー 複数の DPDK プロセスを同じコアで実行することはできません。これは、とりわけメモリプールキャッシュの破損を引き起こすためです。二次プロセスは別のコアに固定されています。*configmap* でCPU ピニングオプションを使用して、セカンダリプロセスを制御します。
- 設定とポッドの変更
  - `pan-cn-ngfw-configmap-0.yaml` および `pan-cn-ngfw-configmap-1.yaml`で `PAN_DATA_MODE："dpdk"` を有効にします。

    *DPDK* は、*CN-Series-as-a-kubernetes-CNF*のデフォルトモードではありません。
  - `#HUGEPAGE_MEMORY_REQUEST` パラメータを `pan-cn-ngfw-configmap-0.yaml` および `pan-cn-ngfw-configmap-1.yaml`のhugepage メモリリクエストと一致させます。

    *hugepage* メモリが使用できない場合、デフォルトで *MMAP* になります。

詳細については、「DPDK システム要件」を参照してください。

オンプレミスワーカーノードとAWS EKS クラスタに DPDK を設定できます

- オンプレミスのワーカーノード に DPDK をセットアップする
- AWS EKSにDPDK を設定する

## オンプレミスのワーカーノード に DPDK をセットアップする

**STEP 1 |** 以下の依存関係をインストールします。

DPDKをセットアップするワーカーノードですべてのコマンドを実行します。

- CentOSの場合：

```
yum groupinstall 'Development Tools' -y yum install net-tools
 pciutils -y yum install git gcc make -y yum install numactl-
devel -y yum install which -y yum install -y sudo libhugetlbfs-
utils libpcap-devel kernel kernel-devel kernel-headers yum
 update -y yum install epel-release -y yum install python36 -y
```

- Ubuntu OSの場合：

```
sudo apt install build-essential sudo apt-get install libnuma-
dev
```

**STEP 2 |** 依存関係のインストール後は以下を行ないます。

- https://fast.dpdk.org/rel/からDPDKのtarファイルをダウンロードします。コンパイル手順については、DPDKのドキュメントを参照してください。

```
wget https: / /fast.dpdk.org/rel/dpdk-19.11.9.tar.xz
```

- ファイルをuntarします。

```
tar -xvf dpdk-19.11.9.tar.xz cd dpdk-stable-19.11.9
```

- ファイルをコンパイルします。コンパイルされたファイルは`x86_64-native-linuxapp-gcc`サブフォルダにあります

```
make install T=x86_64-native-linuxapp-gcc
```

**STEP 3 |** コンパイルされたカーネルモジュールを実行時に統計的または動的に挿入する(modprobe/insmod)。詳細は、カーネルモジュールを参照してください。

```
cd x86_64-native-linuxapp-gcc/kmod insmod igb_uio.ko insmod
 rte_kni.ko
```

*Ubuntuで`insmod`のエラーを確認した場合:エラー: `igb_uio.ko`モジュールを挿入できませんでした。先に`uio`モジュールを挿入してください。*

```
modprobe uio
```

**STEP 4 |** 起動時にモジュールを挿入するためにディストリビューション固有の方法を使用します。あるいは、システムの起動のたびにmodprobe/insmodコマンドを実行するサービスを作成することもできます。

```
cp <service-file> to /etc/systemd/system sudo systemctl daemon-
reload
```

**STEP 5 |** 2048Kの容量の2M Hugepagesをアクティベートしてマウントします。

ステップ4のサービス スクリプトを使用して、Hugepagesをアクティベートすることもできます。

```
echo 2048 > /sys/devices/system/node/node0/hugepages/
hugepages-2048/nr_hugepages echo 4292967296 > /sys/fs/cgroup/
hugetlb/kubepods.slice/hugetlb.2MB.limit_in_bytes mkdir /mnt/huge
 mount -t hugetlbfs nodev /mnt/huge
```

**STEP 6 |** 今後の使用のためにVMのスナップショットを作成します。

# AWS EKSにDPDK を設定する

AWS EKS では、各ポッドにAmazon VPC CNI プラグインによって割り当てられた 1つのネットワークインターフェースがあります。Multus を使用すると、複数のインターフェイスを備えたポッドを作成できます。

**STEP 1 |** AWS アカウントを作成します (まだお持ちでない場合)。

**STEP 2 |** カスタム AMI を使用して EKS クラスターを作成します。詳細については、Amazon EKS クラスターの作成を参照してください。

**STEP 3 |** VPCとノードの設定を変更します。詳細については、AWS EKS のドキュメントを参照してください。

**STEP 4 |** （Multus）複数のENI をEKS ノードに追加し、KNI および UIO ドライバーをロードします。

- 次のタグを使用して、EKS ノードに複数のENI を追加します。

  ```
  'Key': 'node.k8s.amazonaws.com/no_manage', 'Value': 'true'
  ```

  タグが検出されると、Multus CNI はそのインターフェイスを使用できるようになります。詳細は、Azure ドキュメントを参照してください。

- AWS CLI で次のコマンドを実行します。

  ```
  aws ec2 create-network-interface --subnet-id <>
   --description "test" --groups <> --region=us-
  west-1 --tag-specifications 'ResourceType=network-
  interface,Tags=[{Key='node.k8s.amazonaws.com/
  no_manage',Value='true'}]' aws ec2 attach-network-interface --
  network-interface-id <> --instance-id <> --device-index 2
  ```

- （カスタム AMI を使用していない場合） ワーカーノードで hugepages を有効にします。

  ```
  echo 1024 > /sys/devices/system/node/node0/hugepages/
  hugepages-2048kB/nr_hugepages mkdir -p /mnt/huge mount -t
   hugetlbfs nodev /mnt/huge service kubelet restart
  ```