STACKIT CloudでのCisco Catalyst 8000Vのインストールと設定

内容

はじめに

このドキュメントでは、Schwarz Digitsに属するSTACKITクラウドにCisco Catalyst 8000Vをインストールし、起動するために必要な手順について説明します。

背景

Cisco Catalyst 8000Vは、Cisco IOS XEを実行する仮想ルータです。STACKITでは、C8000Vイメージがカスタムイメージとしてアップロードされ、仮想サーバのブートボリュームの作成に使用

されます。

サーバの作成中、STACKITは構成ドライブに提供されたユーザデータを配置します。C8000Vは最初のブート時にブートストラップを読み取ります。ブートストラップの内容は、目的の動作モードによって異なります。

- 自律モードではiosxe_config.txtを使用します。
- コントローラモードでは、Cisco SD-WAN Managerによって生成されたciscosdwan_cloud_init.cfgが使用されます。
- SDルーティングモードでは、Cisco SD-WAN Managerによって生成されたciscosdwan_cloud_init.cfgがSDルーティングに使用されます。

3つのモードすべてで、同じイメージ、ボリューム、インターフェイス、およびサーバ作成シーケンスが使用されます。ブートストラップの内容と導入後の検証は、モードによって異なります。

前提条件

要件

- 既存のSTACKITプロジェクトと、イメージ、ボリューム、ネットワークインターフェイス、およびサーバを管理する権限を持つ認証済みのオペレータ。
- Bash互換シェルを持つLinuxまたはmacOSワークステーション。このガイドのコマンド例は、LinuxおよびmacOSを対象としています。
- STACKIT設計に従った、承認済みのネットワークプレフィックス、セキュリティ制御、管理アクセス、およびオプションのパブリックIP。
- 選択したCisco IOS XEリリース用の認定C8000V QCOW2イメージ。
- 選択したリリースおよび機能セットに関して公開されているC8000VのCPUおよびメモリ要件を満たすSTACKITマシンタイプ。
- 選択したC8000Vイメージに適したブートボリュームサイズとファームウェア設定。
- QEMUがインストールされていること。これには、ソースイメージの検査に使用されるqemu-imgユーティリティが含まれます。
- コマンド環境でstackit、jq、およびbase64を使用できる。
- 自律、コントローラ、またはSDルーティングモード用に準備されたDay-0ブートストラップ。

ブートストラップには、パスワード、証明書、またはデバイスID情報を含めることができます。保護された場所に保存します。Base64エンコーディングでは、ファイルは暗号化されません。

検証済みソフトウェアリリースとVMタイプ

このドキュメントの導入ワークフローは、次のソフトウェアリリースとSTACKIT VMの種類で検証されています。このテスト済みのベースラインはラボ環境を文書化したもので、現在のCisco Catalyst 8000Vのリリースノート、プラットフォーム要件、または機能固有のサイジングガイダンスに置き換わるものではありません。

ソフトウェア リリース

- Cisco IOS XEリリース：17.18.5
- Cisco SD-WANコントローラリリース：20.18.5、コントローラ(Cisco SD-WAN)モードで使用

STACKIT VMの種類

検証には、次の種類のSTACKIT VMが含まれています。

インスタンスサイズ	vCPU	メモリ(GiB)
c3i.4	4	8
c3i.8	8	16
c3i.16	16	32

C8000Vの導入に必要なCPU、メモリ、スループット、および機能の要件を満たすタイプを選択します。

ゲストレポートによるハイパーバイザの詳細

検証済みのC8000Vゲストから、STACKIT仮想化環境は次のように報告されました。

<#root>

```
Router#

show platform software system hypervisor


Hypervisor:
```

KVM

Manufacturer:

STACKIT Cloud

Product Name:

OpenStack Nova

Serial Number: <INSTANCE_UUID>
UUID: <INSTANCE_UUID>
Image Variant: None

Router#

設定

1. STACKIT CLIのインストール

このガイドでは、Homebrewを使用してSTACKIT CLIをインストールします。他のパッケージマネージャとインストール方法も使用できます。詳細については、STACKIT CLIインストールガイドを参照してください。

```
brew tap stackitcloud/tap
brew install --cask stackit
```

2. 環境の設定

導入コマンドを実行する前に、STACKIT CLI環境を設定します。

2.1ログインと認証

STACKITアカウントにログインします。

```
stackit auth login
```

ブラウザで認証を完了します。認証に成功すると、CLIによってログインしていることが確認されます。

2.2プロジェクトの設定

残りのコマンドの既定のプロジェクトを設定します。

```
stackit config set --project-id xxxxxxxx-yyyy-zzzz-aaaa-bbbbbbbbbbbb
```

プロジェクトIDが不明な場合は、使用可能なプロジェクトを一覧表示します。

```
stackit project list
```

3. 導入に関する入力の準備

展開に必要な値を収集します。この例では、2つのネットワークと2つのNICを作成します。1つ目は管理またはトランスポート用で、2つ目はサービスまたはデータトラフィック用です。

```
export PROJECT_ID="<PROJECT_ID>"
export REGION="<REGION>"
export AVAILABILITY_ZONE="<AVAILABILITY_ZONE>"
export MACHINE_TYPE="<MACHINE_TYPE>"

export SERVER_NAME="<C8000V_NAME>"
export IMAGE_NAME="<C8000V_IMAGE_NAME>"
export IMAGE_FILE="/path/to/<C8000V_IMAGE>.qcow2"
export IMAGE_MIN_DISK_SIZE="<MINIMUM_DISK_GB>"
export BOOT_VOLUME_SIZE="<BOOT_VOLUME_GB>"
export IMAGE_UEFI="<true_or_false>"

export MGMT_NETWORK_NAME="${SERVER_NAME}-mgmt"
export DATA_NETWORK_NAME="${SERVER_NAME}-data"
export MGMT_IPV4_PREFIX="<MANAGEMENT_IPV4_PREFIX>"
export DATA_IPV4_PREFIX="<DATA_IPV4_PREFIX>"
export TRUSTED_SOURCE_CIDR="<TRUSTED_SOURCE_IP_OR_NETWORK>/<PREFIX_LENGTH>"
```

承認されたネットワーク設計の重複しないプレフィックスを使用します。プレフィックス長を含

むCIDR表記でTRUSTED_SOURCE_CIDRを指定します。1つのIPv4アドレスを許可するには/32を使用します。次の管理規則の例では、TRUSTED_SOURCE_CIDRからのSSHのみが許可されています。

4. C8000Vイメージのアップロード

4.1ソースイメージの検査

ファイルが目的のリリースの承認済みイメージであることを確認します。ファイル名とチェックサムを導入レコードに記録します。

```
qemu-img info "$IMAGE_FILE"

if command -v shasum >/dev/null 2>&1; then
  shasum -a 256 "$IMAGE_FILE"
else
  sha256sum "$IMAGE_FILE"
fi
```

4.2 STACKITカスタムイメージの作成

EFIイメージの場合はIMAGE_UEFIをtrueに設定し、BIOSイメージの場合はfalseに設定します。ダウンロードしたC8000Vイメージに指定されている設定を選択します。選択したイメージと最新の製品マニュアルに従って、最小ディスクサイズを設定します。

```
IMAGE_RESPONSE="$(
  stackit image create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --name "$IMAGE_NAME" \
    --disk-format qcow2 \
    --uefi="$IMAGE_UEFI" \
    --min-disk-size "$IMAGE_MIN_DISK_SIZE" \
    --local-file-path "$IMAGE_FILE" \
    --output-format json
)"

export IMAGE_ID="$(printf '%s' "$IMAGE_RESPONSE" | jq -r '.id')"
printf 'IMAGE_ID=%s\n' "$IMAGE_ID"
```

4.3イメージが使用可能になるまでポーリングする

```
stackit image describe "$IMAGE_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION" \
  --output-format json |
  jq '{id, name, status, diskFormat}'
```

イメージのステータスがAVAILABLEになったら、作業を続けます。

5. ブートボリュームを作成します

カスタムイメージから新しいブートボリュームを作成します。ボリュームサイズは、選択したイメージのディスク要件を満たしている必要があります。

```
BOOT_VOLUME_RESPONSE="$(
  stackit volume create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "${SERVER_NAME}-boot" \
    --source-id "$IMAGE_ID" \
    --source-type image \
    --size "$BOOT_VOLUME_SIZE" \
    --output-format json
)"

export BOOT_VOLUME_ID="$(printf '%s' "$BOOT_VOLUME_RESPONSE" | jq -r '.id')"
printf 'BOOT_VOLUME_ID=%s\n' "$BOOT_VOLUME_ID"
```

ボリュームのステータスがAVAILABLEになるまでボリュームをポーリングします。

```
stackit volume describe "$BOOT_VOLUME_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION" \
  --output-format json |
  jq '{id, name, status, size}'
```

6. ネットワークとセキュリティの作成

既存のネットワークとセキュリティグループがすでに承認済みの設計に適合している場合は、それらを使用します。それ以外の場合は、次に示すようにリソースを作成します。

6.1ネットワークの作成または特定

新しいネットワークを作成する前に、既存のネットワークをリストします。

```
stackit network list \
  --project-id "$PROJECT_ID" \
  --region "$REGION"
```

適切なネットワークが存在しない場合は、管理ネットワークまたはトランスポートネットワークと、サービスまたはデータネットワークを作成します。

```
MGMT_NETWORK_RESPONSE="$(
  stackit network create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --name "$MGMT_NETWORK_NAME" \
    --ipv4-prefix "$MGMT_IPV4_PREFIX" \
    --output-format json
)"

DATA_NETWORK_RESPONSE="$(
  stackit network create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --name "$DATA_NETWORK_NAME" \
    --ipv4-prefix "$DATA_IPV4_PREFIX" \
    --output-format json
)"

export MGMT_NETWORK_ID="$(printf '%s' "$MGMT_NETWORK_RESPONSE" | jq -r '.id')"
export DATA_NETWORK_ID="$(printf '%s' "$DATA_NETWORK_RESPONSE" | jq -r '.id')"
printf 'MGMT_NETWORK_ID=%s\nDATA_NETWORK_ID=%s\n' "$MGMT_NETWORK_ID" "$DATA_NETWORK_ID"
```

承認されたネットワークがすでに存在する場合は、代わりにMGMT_NETWORK_IDとDATA_NETWORK_IDをそれらのIDに設定します。

6.2管理セキュリティグループの作成

管理NICのステートフルセキュリティグループを作成します。

```
SECURITY_GROUP_RESPONSE="$(
  stackit security-group create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --name "${SERVER_NAME}-mgmt" \
```

```
    --description "Management access for ${SERVER_NAME}" \
    --stateful \
    --output-format json
)"

export MGMT_SECURITY_GROUP_ID="$(printf '%s' "$SECURITY_GROUP_RESPONSE" | jq -r '.id')"
printf 'MGMT_SECURITY_GROUP_ID=%s\n' "$MGMT_SECURITY_GROUP_ID"
```

承認されたソースCIDRからのSSHを許可します。

```
stackit security-group rule create \
  --project-id "$PROJECT_ID" \
  --region "$REGION" \
  --security-group-id "$MGMT_SECURITY_GROUP_ID" \
  --direction ingress \
  --ether-type IPv4 \
  --protocol-name tcp \
  --port-range-min 22 \
  --port-range-max 22 \
  --ip-range "$TRUSTED_SOURCE_CIDR" \
  --description "SSH from approved source"
```

HTTPS、NETCONF、ICMP、またはCisco SD-WANのコントロールルールとデータプレーンルールは、導入で必要な場合にのみ追加します。各入力ルールを実用的な最も狭い送信元に制限します。

7. 順序付けられたNICの作成

サーバを作成する前に、NICを作成します。管理セキュリティグループをNIC 1に接続し、目的のC8000Vインターフェイスの順序で結果のNIC IDを送信します。

7.1 NIC 1とNIC 2の作成

```
MGMT_NIC_RESPONSE="$(
  stackit network-interface create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --network-id "$MGMT_NETWORK_ID" \
    --name "${SERVER_NAME}-mgmt" \
    --nic-security \
    --security-groups "$MGMT_SECURITY_GROUP_ID" \
    --output-format json
)"

DATA_NIC_RESPONSE="$(
  stackit network-interface create \
    --project-id "$PROJECT_ID" \
```

```
    --region "$REGION" \
    --network-id "$DATA_NETWORK_ID" \
    --name "${SERVER_NAME}-data" \
    --nic-security \
    --output-format json
)"

export MGMT_NIC_ID="$(printf '%s' "$MGMT_NIC_RESPONSE" | jq -r '.id')"
export DATA_NIC_ID="$(printf '%s' "$DATA_NIC_RESPONSE" | jq -r '.id')"
printf 'MGMT_NIC_ID=%s\nDATA_NIC_ID=%s\n' "$MGMT_NIC_ID" "$DATA_NIC_ID"
```

server-createの例では、最初にGigabitEthernet1に対してMGMT_NIC_IDを送信し、2番目にGigabitEthernet2に対してDATA_NIC_IDを送信しています。追加のNIC IDは、送信された順序で後続のIOS XEインターフェイスにマッピングされます。サーバのブート後にIOS XEから得られたマッピングを確認します。

各NICのNIC ID、ネットワークID、MACアドレス、割り当てられたIPアドレス、および目的のIOS XEインターフェイスを記録します。

7.2オプションのパブリックIPの関連付け

パブリックIPは特定のNICに関連付けられます。外部管理が必要な場合は、NIC 1に関連付けます。

プライベートパスを介して管理アクセスが提供される場合は、この手順を省略します。

```
PUBLIC_IP_RESPONSE="$(
  stackit public-ip create \
    --project-id "$PROJECT_ID" \
    --region "$REGION" \
    --associated-resource-id "$MGMT_NIC_ID" \
    --output-format json
)"

export PUBLIC_IP_ID="$(printf '%s' "$PUBLIC_IP_RESPONSE" | jq -r '.id')"
printf '%s\n' "$PUBLIC_IP_RESPONSE" | jq '{id, ip, associatedResourceId}'
```

8. ゼロデイ設定の準備

8.1ブートストラップファイルの選択

自律モードの場合、1行に1つのIOS XEコンフィギュレーションコマンドでiosxe_config.txtを作成します。

コントローラモードの場合は、Cisco SD-WAN ManagerでコントローラモードCloud-Initブートストラップを生成し、ファイル名ciscosdwan_cloud_init.cfgを保持します。

SDルーティングモードの場合、SDルーティングモードCloud-Initブートストラップを Cisco SD-WAN Managerで生成し、ファイル名ciscosdwan_cloud_init.cfgを保持します。

Cisco SD-WAN Managerで生成されたブートストラップを手動で編集しないでください。生成されたファイルには、デバイスIDとオンボーディング情報が含まれており、クレデンシャルを持つアーティファクトとして保護する必要があります。

8.2自律モードの例

```
hostname <ROUTER_HOSTNAME>
!
ip domain name <DOMAIN_NAME>
!
username <ADMIN_USER> privilege 15 secret <ADMIN_SECRET>
enable secret <ENABLE_SECRET>
!
interface GigabitEthernet1
 description Management
 ip address dhcp
 no shutdown
!
interface GigabitEthernet2
 description Service
 no ip address
 no shutdown
!
crypto key generate rsa modulus 2048
ip ssh version 2
!
line vty 0 4
 login local
 transport input ssh
!
end
```

選択したブートストラップへのパスを設定し、ファイルへのアクセスを制限します。

```
export BOOTSTRAP_FILE="/path/to/<BOOTSTRAP_FILE>"
chmod 600 "$BOOTSTRAP_FILE"
```

8.3ブートストラップのエンコード

```
export USER_DATA_B64="$(base64 < "$BOOTSTRAP_FILE" | tr -d '\r\n')"
```

9. C8000Vインスタンスの作成

9.1 server-createリクエストを作成する

C8000V固有の設定は、configDrive: true、userData内のbase64エンコードされたブートストラップ、イメージベースのブートボリューム、および順序付けされたNIC IDです。

```
jq -n \
  --arg name "$SERVER_NAME" \
  --arg machineType "$MACHINE_TYPE" \
  --arg availabilityZone "$AVAILABILITY_ZONE" \
  --arg bootVolumeId "$BOOT_VOLUME_ID" \
  --arg mgmtNicId "$MGMT_NIC_ID" \
  --arg dataNicId "$DATA_NIC_ID" \
  --arg userData "$USER_DATA_B64" \
  '{
    name: $name,
    machineType: $machineType,
    availabilityZone: $availabilityZone,
    configDrive: true,
    userData: $userData,
    bootVolume: {
      source: {
        type: "volume",
        id: $bootVolumeId
      }
    },
    networking: {
      nicIds: [$mgmtNicId, $dataNicId]
    },
    labels: {
      product: "cisco-c8000v"
    }
  }' > c8000v-server.json
```

ブートストラップを表示せずに要求を確認します。

```
jq 'del(.userData)' c8000v-server.json
```

9.2サーバの作成

対象の環境について文書化されている現行のSTACKIT IaaS v2エンドポイントを使用します。必要に応じて、STACKIT API URLを置き換えます。

```
SERVER_RESPONSE="$(
  stackit curl \
    -X POST \
    "https://<STACKIT API URL>/v2/projects/${PROJECT_ID}/regions/${REGION}/servers" \
    -H "Content-Type: application/json" \
    --data @c8000v-server.json \
    --fail
)"

export SERVER_ID="$(printf '%s' "$SERVER_RESPONSE" | jq -r '.id')"
printf '%s\n' "$SERVER_RESPONSE" | jq '{id, name, status, configDrive}'
```

回答例：

```
{
  "id": "<SERVER_ID>",
  "name": "<C8000V_NAME>",
  "status": "CREATING",
  "configDrive": true
}
```

サーバが予期される実行状態に達するまでポーリングします。

```
stackit server describe "$SERVER_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION" \
  --output-format json |
  jq '{id, name, status, powerStatus}'
```

10. 導入の確認

10.1 STACKITリソースの確認

次の項目を確認します。

- サーバは、目的のプロジェクト、リージョン、およびアベイラビリティーゾーンで実行されています。
- 目的のブートボリュームが接続されます。
- 予想されるNICが接続されます。
- オプションのパブリックIPアドレスは、目的のNICに関連付けられます。
- 適用されたアクセス制御は、承認された設計と一致します。

10.2 Cisco IOS XEの確認

承認された管理パスを介して接続し、選択したリリースとモードに適したチェックを実行します。

```
show version
show platform software vnic-if interface-mapping
show ip interface brief
show ip route
```

目的のモードがアクティブで、Day-0設定が適用され、IOS XEインターフェイスが送信されたNICの順序と一致することを確認します。

コントローラモード用の追加コマンド：

```
show sdwan control connections
show sdwan control local-properties
show sdwan system status
```

SDルーティングモード用の追加コマンド：

```
show sd-routing control connections summary
show sd-routing control local-properties summary
show sd-routing system status
```

確認後、一時的なプロビジョニングクレデンシャルをローテーションし、ローカル要求ファイルとエンコードされた変数を削除します。

```
rm -f c8000v-server.json
unset USER_DATA_B64
```

11. 配置を削除します

このセクションのクリーンアップコマンドは、リソースを完全に削除します。展開が不要になり、各リソースIDがこのC8000V展開にのみ属していることを確認します。既存または共有のネットワーク、セキュリティグループ、イメージ、またはその他のリソースを削除しないでください。

クリーンアップの例では、インタラクティブな確認プロンプトが維持されます。操作を承認する前に、各STACKIT CLI確認プロンプトに示されているリソースを確認します。

コントローラまたはSDルーティングモードの場合は、STACKITリソースを削除する前に、必要なコントローラ側の使用停止を完了します。

11.1取り外しの準備

続行する前に、記録されたリソースIDを確認してください。

```
printf 'SERVER_ID=%s\n' "$SERVER_ID"
printf 'PUBLIC_IP_ID=%s\n' "${PUBLIC_IP_ID:-not-created}"
printf 'MGMT_NIC_ID=%s\nDATA_NIC_ID=%s\n' "$MGMT_NIC_ID" "$DATA_NIC_ID"
printf 'BOOT_VOLUME_ID=%s\n' "$BOOT_VOLUME_ID"
printf 'MGMT_SECURITY_GROUP_ID=%s\n' "$MGMT_SECURITY_GROUP_ID"
printf 'MGMT_NETWORK_ID=%s\nDATA_NETWORK_ID=%s\n' "$MGMT_NETWORK_ID" "$DATA_NETWORK_ID"
printf 'IMAGE_ID=%s\n' "$IMAGE_ID"
```

導入にパブリックIPがある場合は、サーバを削除する前にIPを切断します。

```
if [ -n "${PUBLIC_IP_ID:-}" ]; then
  stackit server public-ip detach "$PUBLIC_IP_ID" \
    --server-id "$SERVER_ID" \
    --project-id "$PROJECT_ID" \
    --region "$REGION"
fi
```

11.2導入リソースの削除

最初にサーバを削除します。サーバがサーバリストに表示されなくなったら、続行します。

```
stackit server delete "$SERVER_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"

stackit server list \
  --project-id "$PROJECT_ID" \
  --region "$REGION"
```

オプションのパブリックIP、順序付けされた2つのNIC、およびブートボリュームを削除します。

```
if [ -n "${PUBLIC_IP_ID:-}" ]; then
  stackit public-ip delete "$PUBLIC_IP_ID" \
    --project-id "$PROJECT_ID" \
    --region "$REGION"
fi

stackit network-interface delete "$MGMT_NIC_ID" \
  --network-id "$MGMT_NETWORK_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"

stackit network-interface delete "$DATA_NIC_ID" \
  --network-id "$DATA_NETWORK_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"

stackit volume delete "$BOOT_VOLUME_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"
```

管理セキュリティグループとネットワークがこの展開専用に作成されている場合は、依存リソースが削除された後で削除してください。

```
stackit security-group delete "$MGMT_SECURITY_GROUP_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"

stackit network delete "$MGMT_NETWORK_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"

stackit network delete "$DATA_NETWORK_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"
```

カスタムイメージは、別のC8000Vの導入に使用する際に保持してください。それ以外の場合は、依存するすべてのブートボリュームが削除された後に削除します。

```
stackit image delete "$IMAGE_ID" \
  --project-id "$PROJECT_ID" \
  --region "$REGION"
```

11.3クリーンアップの確認

残りのリソースをリストし、削除したIDが存在しないことを確認します。意図的に保持された共有リソースも引き続き表示されます。

```
stackit server list --project-id "$PROJECT_ID" --region "$REGION"
stackit public-ip list --project-id "$PROJECT_ID" --region "$REGION"
stackit volume list --project-id "$PROJECT_ID" --region "$REGION"
stackit security-group list --project-id "$PROJECT_ID" --region "$REGION"
stackit network list --project-id "$PROJECT_ID" --region "$REGION"
stackit image list --project-id "$PROJECT_ID" --region "$REGION"
```

導入後に保持されたローカルブートストラップのアーティファクトが、組織のクレデンシャル保持ポリシーに従って処理されていることを確認します。

付録

便利なSTACKITコマンド

```
# List servers
stackit server list --project-id "$PROJECT_ID" --region "$REGION"

# List volumes
stackit volume list --project-id "$PROJECT_ID" --region "$REGION"

# List custom images
stackit image list --project-id "$PROJECT_ID" --region "$REGION"

# List network interfaces
stackit network-interface list --project-id "$PROJECT_ID" --region "$REGION"
```

便利なC8000Vコマンド

```
show version
show platform software vnic-if interface-mapping
show ip interface brief
show ip route
```

```
show logging
```

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。