Install and Configure Cisco Catalyst 8000V on STACKIT Cloud

Contents

Introduction

This document describes the steps needed to install and bring up Cisco Catalyst 8000V on STACKIT cloud belonging to Schwarz Digits.

Background

Cisco Catalyst 8000V is a virtual router that runs Cisco IOS XE. On STACKIT, the C8000V image is uploaded as a custom image and used to create the boot volume for the virtual server.

During server creation, STACKIT places the supplied user data on a config drive. C8000V reads the bootstrap during the initial boot. The bootstrap content depends on the intended operating mode:

- Autonomous mode uses iosxe_config.txt.
- Controller mode uses ciscosdwan_cloud_init.cfg generated by Cisco SD-WAN Manager.
- SD-Routing mode uses ciscosdwan_cloud_init.cfg generated by Cisco SD-WAN Manager for SD-Routing.

The same image, volume, interface, and server-create sequence is used for all three modes. The bootstrap content and post-deployment verification differ by mode.

Prerequisites

Requirements

- An existing STACKIT project and an authenticated operator with permission to manage images, volumes, network interfaces, and servers.
- A Linux or macOS workstation with a Bash-compatible shell. The command examples in this guide are intended for Linux and macOS.
- Approved network prefixes, security controls, management access, and an optional public IP that follow the STACKIT design.
- An authorized C8000V QCOW2 image for the selected Cisco IOS XE release.
- A STACKIT machine type that meets the published C8000V CPU and memory requirements for the selected release and feature set.
- A boot-volume size and firmware setting appropriate for the selected C8000V image.
- QEMU installed, including the *qemu-img* utility used to inspect the source image.
- *stackit*, *jq*, and *base64* available in the command environment.
- A Day-0 bootstrap prepared for Autonomous, Controller, or SD-Routing mode.

The bootstrap can contain passwords, certificates, or device identity information. Store it in a protected location. Base64 encoding does not encrypt the file.

Validated Software Releases and VM Types

The deployment workflow in this document was validated with the following software releases and STACKIT VM flavors. This tested baseline documents the lab environment; it does not replace the current Cisco Catalyst 8000V release notes, platform requirements, or feature-specific sizing guidance.

Software releases

- Cisco IOS XE release: 17.18.5
- Cisco SD-WAN Controllers release: 20.18.5, used for Controller (Cisco SD-WAN) mode

STACKIT VM flavors

The following STACKIT VM flavors were included in validation:

Instance size	vCPUs	Memory (GiB)
c3i.4	4	8
c3i.8	8	16
c3i.16	16	32

Select a flavor that meets the CPU, memory, throughput, and feature requirements for the intended C8000V deployment.

Guest-reported hypervisor details

The validated C8000V guest reported the STACKIT virtualization environment as follows:

<#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#
```

Configure

1. Install the STACKIT CLI

This guide uses Homebrew to install the STACKIT CLI. Other package managers and installation methods are available. For more information, see the STACKIT CLI installation guide.

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

2. Set the environment

Configure the STACKIT CLI environment before running the deployment commands.

2.1 Log in and authenticate

Log in to your STACKIT account.

```
stackit auth login
```

Complete the authentication in your browser. After authentication succeeds, the CLI confirms that you are logged in.

2.2 Set the project

Set the default project for the remaining commands.

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

If you do not know the project ID, list the available projects.

```
stackit project list
```

3. Prepare the deployment inputs

Collect the values required by the deployment. The example creates two networks and two NICs: the first for management or transport and the second for service or data traffic.

```
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>"
```

Use non-overlapping prefixes from the approved network design. Specify *TRUSTED_SOURCE_CIDR* in CIDR notation, including the prefix length. Use */32* to permit one IPv4 address. The management rule examples below allow SSH only from *TRUSTED_SOURCE_CIDR*.

4. Upload the C8000V image

4.1 Inspect the source image

Verify that the file is the authorized image for the intended release. Record the filename and checksum in the deployment record.

```
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 Create the STACKIT custom image

Set *IMAGE_UEFI* to *true* for an EFI image or *false* for a BIOS image. Select the setting specified for the downloaded C8000V image. Set the minimum disk size according to the selected image and current product documentation.

```
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 Poll until the image is available

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

Continue when the image status is *AVAILABLE*.

5. Create the boot volume

Create a new boot volume from the custom image. The volume size must meet the disk requirement of the selected image.

```
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"
```

Poll the volume until its status is *AVAILABLE*.

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

6. Create networks and security

Use existing networks and security groups when they already meet the approved design. Otherwise, create the resources as shown below.

6.1 Create or identify the networks

List the existing networks before creating new ones.

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

If suitable networks do not exist, create the management or transport network and the service or data network.

```
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"
```

If approved networks already exist, set *MGMT_NETWORK_ID* and *DATA_NETWORK_ID* to their IDs instead.

6.2 Create the management security group

Create a stateful security group for the management NIC.

```
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"
```

Allow SSH from an approved source CIDR.

```
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"
```

Add HTTPS, NETCONF, ICMP, or Cisco SD-WAN control and data-plane rules only when required by the deployment. Limit each ingress rule to the narrowest practical source.

7. Create ordered NICs

Create the NICs before creating the server. Attach the management security group to NIC 1 and submit the resulting NIC IDs in the intended C8000V interface order.

7.1 Create NIC 1 and NIC 2

```
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"
```

The server-create example submits *MGMT_NIC_ID* first for *GigabitEthernet1* and *DATA_NIC_ID* second for *GigabitEthernet2*. Additional NIC IDs map to subsequent IOS XE interfaces in the order submitted. Verify the resulting mapping from IOS XE after the server boots.

Record the NIC ID, network ID, MAC address, assigned IP address, and intended IOS XE interface for each NIC.

7.2 Associate an optional public IP

A public IP is associated with a specific NIC. When external management is required, associate it with NIC 1.

Omit this step when management access is provided through a private path.

```
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. Prepare the Day-0 configuration

8.1 Select the bootstrap file

For Autonomous mode, create *iosxe_config.txt* with one IOS XE configuration command per line.

For Controller mode, generate a Controller-mode Cloud-Init bootstrap in Cisco SD-WAN Manager and retain the filename *ciscosdwan_cloud_init.cfg*.

For SD-Routing mode, generate an SD-Routing-mode Cloud-Init bootstrap in Cisco SD-WAN Manager and retain the filename *ciscosdwan_cloud_init.cfg*.

Do not manually edit a bootstrap generated by Cisco SD-WAN Manager. A generated file contains device identity and onboarding information and must be protected as a credential-bearing artifact.

8.2 Autonomous-mode example

```
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
```

Set the path to the selected bootstrap and restrict access to the file.

```
export BOOTSTRAP_FILE="/path/to/<BOOTSTRAP_FILE>"
```

```
chmod 600 "$BOOTSTRAP_FILE"
```

8.3 Encode the bootstrap

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

9. Create the C8000V instance

9.1 Build the server-create request

The C8000V-specific settings are configDrive: true, the base64-encoded bootstrap in userData, the image-based boot volume, and the ordered NIC IDs.

```
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
```

Review the request without displaying the bootstrap:

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

9.2 Create the server

Use the current STACKIT IaaS v2 endpoint documented for the target environment. Replace STACKIT API URL as needed.

```
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}'
```

Example response:

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

Poll until the server reaches the expected running state.

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

10. Verify the deployment

10.1 Verify the STACKIT resources

Confirm that:

- the server is running in the intended project, region, and availability zone;
- the intended boot volume is attached;

- the expected NICs are attached;
- the optional public IP is associated with the intended NIC; and
- the applied access controls match the approved design.

10.2 Verify Cisco IOS XE

Connect through the approved management path and run the checks appropriate to the selected release and mode.

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

Confirm that the intended mode is active, the Day-0 configuration was applied, and the IOS XE interfaces match the submitted NIC order.

Additional commands for Controller mode:

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

Additional commands for SD-Routing mode:

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

After verification, rotate temporary provisioning credentials and remove the local request file and encoded variable.

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

11. Remove the deployment

The cleanup commands in this section permanently delete resources. Confirm that the deployment is no longer required and that each resource ID belongs only to this C8000V deployment. Do not delete existing or shared networks, security groups, images, or other resources.

The cleanup examples retain interactive confirmation prompts. Review the resource shown in each STACKIT CLI confirmation prompt before approving the operation.

For Controller or SD-Routing mode, complete any required controller-side decommissioning before deleting the STACKIT resources.

11.1 Prepare for removal

Review the recorded resource IDs before continuing.

```
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"
```

If the deployment has a public IP, detach it before deleting the server.

```
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 Delete the deployment resources

Delete the server first. Continue after the server no longer appears in the server list.

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

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

Delete the optional public IP, the two ordered NICs, and the boot volume.

```
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"
```

If the management security group and networks were created only for this deployment, delete them after their dependent resources are gone.

```
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"
```

Keep the custom image when it will be used for another C8000V deployment. Otherwise, delete it after all dependent boot volumes are removed.

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

11.3 Verify the cleanup

List the remaining resources and confirm that the deleted IDs are no longer present. Shared resources that were intentionally retained should still appear.

```
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"
```

Confirm that any local bootstrap artifacts retained after deployment have been handled according to the organization's credential-retention policy.

Appendix

Useful STACKIT commands

```
# 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"
```

Useful C8000V commands

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