Hewlett Packard
Enterprise

HPE Cray Operating System Installation Guide: CSM on HPE Cray EX Systems(2.3.101) (S-8025)

Part Number: S-8025
Published: July 2022

HPE Cray Operating System Installation Guide: CSM on HPE Cray EX Systems

Contents

1 Copyright and Version

COS: 2.3.101-53

Doc git hash: 994ac7dc96c64495399412a2504f39378873d6b3

Generated: Wed Jul 13 2022

2 Overview

2.1 Installation Overview

This document describes how to install, upgrade, configure and boot the HPE Cray Operating System (COS) on a system managed by Cray System Management (CSM) software.

Installing or upgrading COS software is primarily accomplished by executing an `install.sh` script included in the product release distribution file. This process does the following:

- Uploads RPMs to Nexus repositories,
- Uploads Docker images to the Docker repository,
- Uploads Helm charts to the Helm chart repository,
- Registers COS product information with the CSM product catalog, and
- Installs or upgrades the Content Projection Service (CPS) and Node Memory Dump (NMD) microservices. When a new version of a microservice is provided, the microservice will automatically upgrade to the new version in a rolling fashion. Administrators and users should not notice a loss in capability while the microservice is updated.
- Does not set up GPU support. To set up GPU nodes, see the "Enable GPU Support" section in ***HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems***.
- The installation process does not automatically affect the state of any other existing COS software on the system, for example COS software like the Data Virtualization Service (DVS) running on non-compute node (NCN) worker nodes or any COS software that runs on compute nodes. Additional operational tasks must be performed after the install.sh script has completed to use those components.

Multiple releases of COS can be installed on a system at the same time. The administrator can decide which release to use on compute nodes (or use multiple releases). Only one release of COS, typically the most recent version installed, can run on NCNs.

The same instructions are followed whether the administrator is installing COS for the first time or upgrading COS on a previously installed system.

2.2 COS Version Information

- COS 2.3 is an asynchronous release.
- New COS configuration content will be uploaded to the CSM Version Control Service (VCS) with a specific version number (2.3.XX) to distinguish it from any previously released COS configuration content.
- A new COS image recipe will be uploaded to the CSM Image Management Service (IMS) with a new version number (2.3.XX) to distinguish it from any previously released COS image recipe.
- The new version of COS (2.3.XX) and its artifacts will be displayed in the CSM product catalog alongside any previously released version of COS and its artifacts.
- COS RPMs will be stored in new Nexus raw repositories cos-2.3.XX-sle-15sp3-compute and cos-2.3.XX-net-sle-15sp3-compute, as well as cos-2.3.XX-sle-15sp3 and cos-2.3.XX-net-sle-15sp3. Nexus group repositories cos-2.3-sle-15sp3-compute, cos-2.3-net-sle-15sp3-compute, cos-2.3-sle-15sp3, and cos-2.3-net-sle-15sp3 will be configured to reference the newly installed Nexus raw repositories.
  - When a COS image is built from the COS image recipe, it references the cos-2.3-sle-15sp3-compute Nexus group repository to determine which RPM content to include in the image.
- COS NCN content references the cos-2.3-sle-15sp3 and cos-2.3-net-sle-15sp3 Nexus group repository.

2.3 Dependencies

COS depends on the following products:

- Cray System Management (CSM): General installation and system management capabilities. The COS compute image recipe references a CSM repository stored in Nexus.

- System Admin Toolkit (SAT): SAT commands are referenced in the install and admin guides.
- SUSE Linux Enterprise Operating System (SLE OS): The COS compute image recipe references SLE OS Nexus repos.
- Slingshot Host Software (SHS): CFS configuration sessions created for compute and UAN images based on COS must include a SHS layer before any COS layers in order to satisfy COS RPM dependencies. The NCN customization configuration layer has the same requirement.

For recipe-specific version compatibility information, refer to ***HPE Cray EX System Software Getting Started Guide S-8000 22.03***

2.4 Differences from the Previous Release

Significant changes from the previous release of COS are described in the following subsections.

2.4.1 New Features

- General security improvements and RPM signing
- DVS now has support for the `kkfilnd` Lustre Network Driver (LND) for running on the following hardware:
  - HPE Slingshot switch and HPE Slingshot SA210S Ethernet 200GB 1-port PCIe NIC (air-cooled)
  - HPE Slingshot switch and HPE Slingshot SA220M Ethernet 200GB NIC mezzanine card (NMC) (liquid-cooled)
- GPU support for Nvidia SDK 22.3 has been added.
- GPU support for CUDA driver 510.47.03 has been added.
- GPU support for AMD ROCm 5.0.2 has been added.
- GPU support for AMD driver 21.50.2 has been added.
- GPU support for `gpu-nexus-tool` has been added which makes managing GPU vendor support RPMs easier for an admin. This tool is available on all worker nodes. The admin guide still details how to do this manually without the tool.
- GPU support instructions have been added to Install or Upgrade COS. The changes result in fewer steps for the admin to add GPU support and utilizes the `gpu-nexus-tool` and `sat bootprep`
- GPU support provided by the COS configuration layer has also changed so be sure to follow the new instructions. A standard installation no longer requires editing the Ansible and managing additional Git branches.

2.4.2 Deprecated Features

- None

2.4.3 Removed Features

- GPU support for Nvidia SDK 21.9 has been removed.
- GPU support for CUDA driver 470.103.01 has been removed.
- GPU support for AMD ROCm 4.5.2 has been removed.
- GPU support for AMD driver 21.40.2 has been removed.

2.4.4 Other Changes

- The procedure for configuring routes to Arista switches for Lustre mounts on various node types has changed significantly. If you are using Arista switches in this manner, please see the Arista switch route configuration step in the ***Mount a Lustre File System on NCNs*** and ***Mount a Lustre File System on CNs and UANs*** sections of the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems for details.
- CPS now supports s3fs. This improves how files managed by CPS are stored on NCN worker nodes, minimizing the amount of disk space used. Refer to the ***Content Projection Service*** section in the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems for details.

- This document has been updated to include instructions on how to use the `sat bootprep` command to build and configure COS compute images and to create BOS session templates for booting COS compute images. Using `sat bootprep` greatly streamlines these processes.
- A new section for booting HPE Apollo 6500 Gen10 Plus blades with HPE XL675d Nvidia A100 Modular accelerator Trays has been added. The section documents additional steps required to properly boot this hardware type. See Boot COS on HPE Apollo 6500 Gen10 Plus XL675d with Nvidia A100 GPU Tray for more information.

2.5 Prerequisites

The following prerequisites apply to this release:

- If an upgrade is being performed, the system is required to be at the COS 2.2 release or newer.
- If an upgrade is being performed, a CSM 1.2 pre-installation script provided by COS 2.3 must be executed prior to the upgrade to CSM 1.2 on NCN worker nodes. See the COS 2.3 Upgrade Preparation section of this document for details.

The following prerequisites apply to both install and upgrade:

- SUSE Linux Enterprise Operating System for HPE Cray EX product must be installed.
  - This can be verified by seeing the correct repos are in Nexus.
- The System Admin Toolkit must be installed and configured.
  - This can be verified by the command “sat –version”
- CSM has been installed and all services are healthy. The CSM system management health checks can be run to identify issues, if necessary.
- CPS s3fs support requires CSM version 1.2 or newer. CPS s3fs support is configurable to ensure that CPS will work with releases prior to CSM 1.2. Refer to the “Content Projection Service” section in the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems for details.
- The Cray command line interface (CLI) tool is initialized and configured on the system. See “Configure the Cray Command Line Interface (CLI)” in the CSM documentation for more information.
- SHS has been installed. See “Install Slingshot Host Software” section in ***Slingshot Operations Guide (Customer)*** for more information.

3 COS 2.3 Upgrade Preparation

3.1 Install COS NCN SP3 rpms

This section is only applicable when performing an upgrade to COS 2.3. If you are performing a fresh install, ignore this section and refer to the ***HPE Cray EX System Software Getting Started Guide S-8000*** for details on when to perform the COS installation.

If the system is being upgraded to COS 2.3 in conjunction with an upgrade of CSM to version 1.2, the COS pre-install procedure documented in this section must be executed **before upgrading CSM.** This ensures that COS will continue to function properly during the CSM 1.2 upgrade. This is the only section of this document relevant to the COS pre-install process; all other sections describe the normal COS install and upgrade procedures. The ***HPE Cray EX System Software Getting Started Guide S-8000*** describes when to perform general pre-installation instructions as well as when to run the COS installation and upgrade procedures relative to any other HPE Cray EX software products being installed or upgraded.

3.2 Run install.sh

See the “COS Version Information” section of this document for details on how COS version numbers are referenced in the commands and output below.

1. Start a typescript to capture the commands and output from this installation.

   ```
   ncn-m001# script -af pre-install-cos.$(date +%Y-%m-%d).txt
   ncn-m001# export PS1='\u@\H \D{%Y-%m-%d} \t \w # '
   ```

2. Copy the preinstall distribution compressed tar file to ncn-m001. Save the release version for later usage.

   ```
   ncn-m001# export COS_RELEASE=2.3.XX
   ```

3. Unzip and extract the pre-installation distribution.

   ```
   ncn-m001# tar xzvf cos-${COS_RELEASE}-pre-req-for-csm-1.2.tar.gz
   ```

4. Change to the extracted pre-installation distribution directory.

   ```
   ncn-m001# cd cos-${COS_RELEASE}-pre-req-for-csm-1.2/
   ```

5. COS 2.2 is the expected installed COS product, if the COS version being upgraded is one of the earlier COS product versions such as COS 2.1 or COS 2.0, an edit is required for the nexus-repositories.yaml file that is part of the installation artifacts. Otherwise, skip this step and continue with the next one.

   Modify the following to match the COS product version currently installed on the system if not COS 2.2, in this example COS 2.1 is the version installed:

   ```
   ---
   cleanup: null
   format: raw
   name: cos-2.3.55-sle-15sp3-preinstall
   online: true
   storage:
     blobStoreName: cos
     strictContentTypeValidation: false
     writePolicy: ALLOW_ONCE
   type: hosted
   ---
   format: raw
   group:
     memberNames:
     - cos-2.3.55-sle-15sp3-preinstall
   name: cos-2.2-sle-15sp2      <<<--- CHANGE cos-2.2 TO cos-2.1
   online: true
   storage:
     blobStoreName: cos
     strictContentTypeValidation: false
   type: group
   ```

   The 'cos-2.3.55-sle-15sp3-install' raw repository name is autogenerated when the pre-install tarball is created and will change based on when the artifact is released and is only an example. The nexus group name 'cos-2.X-sle-15sp2' is the key item that must exist on the system and is the item the pre-install script will manipulate to get the proper rpms loading when the NCN worker node begins its upgrade to the SLE 15 SP3 OS.

6. Run the `install.sh` script to begin installing COS NCN SP3 rpms.

   ```
   ncn-m001# ./install.sh
   ```

7. Perform the following steps to update the cos-config-management repository with changes necessary for the upgrade process.

   1. Clone the COS configuration management repository using the proper credentials. Checkout the branch currently used to hold COS configuration.

      The output of the first command provides the password required for the clone. The following example assumes the current COS configuration branch is "cos-integration".

      ```
      ncn-m001# kubectl get secret -n services vcs-user-credentials \
      --template={{.data.vcs_password}} | base64 --decode
      ncn-m001# git clone https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git
      Username for 'https://api-gw-service-nmn.local': crayvcs
      Password for 'https://crayvcs@api-gw-service-nmn.local':
      ncn-m001# cd cos-config-management/
      ncn-m001# git checkout cos-integration
      ```

   2. If you are currently using DVS over the NMN, you will need to change the interface name `vlan002` to `bond0.nmn0` in your COS configuration wherever it occurs, as this interface name has changed in CSM 1.2. Skip this step if you are using DVS over the HSN.

Using a text editor such as `vi`, manually replace each occurrence of `vlan002` with `bond0.nmn0`. You can use the following command to identify occurrences (by file and line number) of `vlan002`:

```
ncn-m001# git grep -n vlan002
```

3. Using a text editor such as `vi`, manually edit or create the following files. The text here is a unified diff suitable for use with `patch` and indicates lines that should be removed (-) or added (+). Those markers should not be put in the final files.

```
diff --git a/playbooks/cos-services-restart.yml b/playbooks/cos-services-restart.yml
new file mode 100644
index 0000000..3c80781
--- /dev/null
+++ b/playbooks/cos-services-restart.yml
@@ -0,0 +1,10 @@
+# Copyright 2022 Hewlett Packard Enterprise Development LP
+#
+# This file is part of HPE Cray Operating System (COS)
+#
+---
+- hosts: Management_Worker
+  any_errors_fatal: true
+  remote_user: root
+  roles:
+    - { role: cos-services-restart }

diff --git a/ncn.yml b/ncn.yml
index 1f87f61..d072f87 100644
--- a/ncn.yml
+++ b/ncn.yml
@@ -7,6 +7,7 @@
   remote_user: root
   roles:
     - cos-services-install
+    - cos-services-restart
     - cray_lnet_install
     - cray_dvs_install
     - cray_lnet_load

diff --git a/roles/cos-services-restart/tasks/main.yml b/roles/cos-services-restart/tasks/main.yml
new file mode 100644
index 0000000..56315eb
--- /dev/null
+++ b/roles/cos-services-restart/tasks/main.yml
@@ -0,0 +1,18 @@
+# Copyright 2022 Hewlett Packard Enterprise Development LP
+#
+# This file is part of the HPE Cray Operating System (COS)
+#
+---
+# tasks file for cos-services-restart
+
+- name: Restart cray-ifscan service
+  ansible.builtin.systemd:
+    name: cray-ifscan
+    state: restarted
+
+- name: Wait for cray-ifscan to finish
+  shell: systemctl is-active cray-ifscan || true
+  register: result
```

```
+  until: result.stdout == "inactive"
+  retries: 60
+  delay: 30
```

4. Commit and push the changes for the branch to the repository using the proper credentials. The output of the first command provides the password required for the push.

   ```
   ncn-m001# kubectl get secret -n services vcs-user-credentials \
   --template={{.data.vcs_password}} | base64 --decode
   ncn-m001# git commit -am 'Added COMMIT_COMMENT'
   ncn-m001# git push --set-upstream origin cos-integration
   Username for 'https://api-gw-service-nmn.local': crayvcs
   Password for 'https://crayvcs@api-gw-service-nmn.local':
   ```

5. Identify the commit hash for this branch, and save the commit hash for later use when updating the CFS COS configuration layer.

   ```
   ncn-m001# git rev-parse --verify HEAD
   <== commit hash output ==>
   ncn-m001# export COS_CONFIG_COMMIT_HASH=<commit hash output>
   ```

6. Disable NCN personalization for all worker nodes before modifying the CFS configuration for workers in the next step.

   Rationale: We want this change to take place only after the worker is upgraded to CSM 1.2 and rebooted, as the interface name change is only valid after the CSM upgrade. The reboot at that time will automatically re-enable personalization.

   For each worker, run the following command:

   ```
   ncn-m001# cray cfs components update <NCN_XNAME> --enabled false
   ```

7. Update the COS config commit hash in the NCN personalization CFS configuration.

   1. Download the current ncn-personalization configuration.

      ```
      ncn-m001# cray cfs configurations describe ncn-personalization \
      --format json > ncn-personalization.json
      ncn-m001# cat ncn-personalization.json
      {
        "lastUpdated": "2021-04-02T15:10:36Z",
        "layers": [
          {
            "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
            "commit": "120a5d2f4acb42203c74ee8920f00c24253247b7",
            "name": "cos-integration",
            "playbook": "ncn.yml"
          },
          ...
        ],
        "name": "ncn-personalization"
      }
      ```

   2. Edit and update the JSON file.

      1. Remove the lastUpdated key.
      2. Remove the second name key that contains the value “ncn-personalization”.
      3. Remove the comma after the layers array.
      4. Update the git commit hash for the COS layer with the COS_CONFIG_COMMIT_HASH saved in Step 5, and leave other product stream layers untouched. The JSON file should now resemble the following:

         ```
         ncn-m001# cat ncn-personalization.json
         {
         "layers": [
             {
               "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
         ```

```
            "commit": "<COS_CONFIG_COMMIT_HASH>",
            "name": "cos-integration",
            "playbook": "ncn.yml"
          },
          ...
        ]
      }
```

3. Upload the new configuration into CFS.

```
ncn-m001# cray cfs configurations update ncn-personalization \
--file ncn-personalization.json --format json
{
  "lastUpdated": "2021-05-19T21:30:09Z",
  "layers": [
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": "3d5114dd96201ca13463db5b69e192d28d404efc",
      "name": "cos-integration",
      "playbook": "ncn.yml"
    }
    ...
],
  "name": "ncn-personalization"
}
```

8. Finish the typescript file started at the beginning of this procedure.

```
ncn-m001# exit
```

At this point, COS 2.3 NCN SP3 rpms have been pre-installed. The NCN workers are now ready to upgrade to CSM 1.2.

Refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine what step to perform next.

4 Install or Upgrade COS

If the system is being upgraded to COS 2.3 in conjunction with an upgrade of CSM to version 1.2, the COS pre-install procedure documented in section Install COS NCN SP3 rpms must be executed **before upgrading CSM.** This ensures that COS will continue to function properly during the CSM 1.2 upgrade.

The ***HPE Cray EX System Software Getting Started Guide S-8000*** describes when to perform general pre-installation instructions as well as when to run the COS installation and upgrade procedures relative to any other HPE Cray EX software products being installed or upgraded.

4.1 Run install.sh

See the "COS Version Information" section of this document for details on how COS version numbers are referenced in the commands and output below.

1. Start a typescript to capture the commands and output from this installation.

```
ncn-m001# script -af product-cos.$(date +%Y-%m-%d).txt
ncn-m001# export PS1='\u@\H \D{%Y-%m-%d} \t \w # '
```

2. Copy the release distribution compressed tar file to ncn-m001. Save the release version for later usage.

```
ncn-m001# export COS_RELEASE=2.3.XX
```

3. Unzip and extract the release distribution.

```
ncn-m001# tar xzvf cos-${COS_RELEASE}.tar.gz
```

4. Change to the extracted release distribution directory.

```
ncn-m001# cd cos-${COS_RELEASE}
```

5. Run the `install.sh` script to begin installing COS.

   ```
   ncn-m001# ./install.sh
   ```

6. Run `./post-install-validation/validate` to verify the COS install. Use the `--help` and `--verbose` options for additional help and debugging information, if needed. The validation tasks performed are summarized in the output below. Note that the output includes the value of `IMS_RECIPE_ID` which can be used in later steps when creating a COS image from recipe.

   ```
   ncn-m001# ./post-install-validation/validate

   [... Fetching required Python packages ...]

   SUMMARY of COS post-install validation

   Post installation check passed.
   Passed:
     All COS Repos were present
     All COS jobs were found
     All COS pods were found
     All kubernetes jobs succeeded
     COS IMS recipe obtained
     COS IMS recipe verified
     goss -g /opt/cray/tests/install/cos/cps/goss-cps.yaml validate
     goss -g /opt/cray/tests/install/cos/nmdv2/goss-NMD.yaml validate
     goss -g /opt/cray/tests/install/cos/nmdv2/goss-prereqs.yaml validate
   EXIT 0

   IMS_RECIPE_ID is required for later steps, PLEASE DO:

           export IMS_RECIPE_ID=f5e0ea59-5687-476c-95c5-85d41c93749c
   ```

At this point, COS 2.3 software has been installed:

- COS configuration content for this release has been uploaded to VCS.
- COS image recipe for this release has been uploaded to IMS.
- COS information for this release has been uploaded to the CSM product catalog.
- COS content for this release has been uploaded to Nexus repositories.
- CPS and NMD microservices have been upgraded if new versions were available.

The remainder of this document describes how to:

- Configure COS host OS software on NCNs via NCN Personalization
- Build a compute image with COS content
- Customize a compute image with COS content
- Boot a compute image with COS content

If other HPE Cray EX software products are being installed in conjunction with COS, refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine what step to perform next.

If other HPE Cray EX software products are not being installed at this time, continue to the next section of this document to configure and boot COS.

5 Configure COS - VCS Content

5.1 VCS Configuration

1. Create a branch using the imported branch from the installation to customize COS.

   The imported branch will be reported in the cray-product-catalog and can be used as a base branch. The imported branch from the installation should not be modified. It is recommended that a branch is created from the imported branch to customize COS configuration content as necessary. The following steps create an integration branch to accomplish this.

   a. Obtain `import_branch` from the cray-product-catalog.

```
ncn-m001# kubectl get cm cray-product-catalog -n services -o yaml \
| yq r - 'data.cos' | yq r - \"${COS_RELEASE}\"
2.3.XX:
  configuration:
    clone_url: https://vcs.DOMAIN_NAME.dev.cray.com/vcs/cray/cos-config-management.git
    commit: 215eab2c316fb75662ace6aaade8b8c2ab2d08ee
    import_branch: cray/cos/2.3.XX
    import_date: 2021-02-21 23:01:16.100251
    ssh_url: git@vcs.DOMAIN_NAME.dev.cray.com:cray/cos-config-management.git
  images: {}
  recipes:
    cray-shasta-compute-sles15sp3.x86_64-1.4.64:
      id: 5149788d-4e5d-493d-b259-f56156a58b0d
```

b. Store the import branch for later use.

```
ncn-m001# export IMPORT_BRANCH=cray/cos/${COS_RELEASE}
```

c. Obtain the credentials for the crayvcs user from a Kubernetes secret. The credentials are required for cloning a branch from VCS. Use git to clone the COS configuration content.

```
ncn-m001# kubectl get secret -n services vcs-user-credentials --template={{.data.vcs_password}} | \
base64 --decode && echo
<==password output==>

ncn-m001# git clone https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git
Username for 'https://api-gw-service-nmn.local': crayvcs
Password for 'https://crayvcs@api-gw-service-nmn.local': <password from preceding output>

ncn-m001# cd cos-config-management/
ncn-m001# git checkout -b $IMPORT_BRANCH origin/$IMPORT_BRANCH
Branch 'cray/cos/<X>.<Y>.<Z>' set up to track remote branch 'cray/cos/<X>.<Y>.<Z>'
from 'origin'.
Switched to a new branch 'cray/cos/<X>.<Y>.<Z>'
```

d. If the integration branch exists, run the following command:

```
ncn-m001# git checkout -b integration origin/integration
Branch 'integration' set up to track remote branch 'integration' from 'origin'.
Switched to a new branch 'integration'
```

e. If the integration branch does not exist, run the following command to create new branch:

```
ncn-m001# git checkout -b integration
Switched to a new branch 'integration'
```

f. Merge import branch into integration branch:

```
ncn-m001# git merge $IMPORT_BRANCH
```

g. Fix any conflicts created by the merge

In particular, there could be merge conflicts in this release if the LNet and DVS configuration had been previously changed. There will be messages similar to:

```
CONFLICT (modify/delete): roles/cray_lnet_load/files/lnet.conf deleted
in origin/cray/cos/X.Y.Z and modified in HEAD. Version HEAD of
roles/cray_lnet_load/files/lnet.conf left in tree.
CONFLICT (modify/delete): roles/cray_lnet_load/files/lnet-worker.conf deleted
in origin/cray/cos/X.Y.Z and modified in HEAD. Version HEAD of
roles/cray_lnet_load/files/lnet-worker.conf left in tree.
CONFLICT (modify/delete): roles/cray_lnet_load/files/lnet-router.conf deleted
in origin/cray/cos/X.Y.Z and modified in HEAD. Version HEAD of
roles/cray_lnet_load/files/lnet-router.conf left in tree.
```

```
CONFLICT (modify/delete): roles/cray_dvs_load/files/dvs_node_map.conf deleted
in origin/cray/cos/X.Y.Z and modified in HEAD. Version HEAD of
roles/cray_dvs_load/files/dvs_node_map.conf left in tree.
CONFLICT (modify/delete): roles/cray_dvs_load/files/dvs.conf deleted
in origin/cray/cos/X.Y.Z and modified in HEAD. Version HEAD of
roles/cray_dvs_load/files/dvs.conf left in tree.
```

Instructions on how to migrate the data from the deleted files are in the "On upgrade" step below. When you've finished moving the data for a file, you can resolve conflict by using "git rm" on the file. For example:

```
ncn-m001# git rm roles/cray_lnet_load/files/lnet.conf
```

2. **Optional**: For an upgrade only, set the number of NCN workers that are upgraded in parallel in the Ansible configuration.

   NOTE: This step can be skipped if you wish to upgrade each worker node serially, one at a time.

   Before the new COS software for DVS and LNet can run on the workers, the old COS software for DVS and LNet needs to be unloaded. The unload of the old software and the reload of the new software is performed on one worker at a time by default, so that the CNs and UANs can keep running by failing over to other workers that are still running and not being reloaded.

   For a COS upgrade on the worker nodes, a rolling upgrade is performed by setting the `serial` keyword in the `ncn-upgrade.yml` playbook. The default value is set to `1` which results in upgrading each worker node one at a time. All Ansible tasks in the `ncn-upgrade.yml` playbook are completed successfully on each worker before the next worker is upgraded.

   To change the number of workers that are upgraded at a time, change the value for the `serial` keyword in `ncn-upgrade.yml` to the number of workers to be upgraded in parallel. The value for the `serial` keyword can also be a percentage of the total number of workers.

   For example, to change the number of NCN workers to be upgraded in parallel to 2:

   ```
   ncn-m001# cat ncn-upgrade.yml
   ...
   - hosts: Management_Worker
     serial: 1
   ...
   ncn-m001# vi ncn-upgrade.yml
   ...
   ncn-m001# cat ncn-upgrade.yml
   ...
   - hosts: Management_Worker
     serial: 2
   ...
   ```

   For example, to change the number of NCN workers to be upgraded in parallel to 25%:

   ```
   ncn-m001# cat ncn-upgrade.yml
   ...
   - hosts: Management_Worker
     serial: 1
   ...
   ncn-m001# vi ncn-upgrade.yml
   ...
   ncn-m001# cat ncn-upgrade.yml
   ...
   - hosts: Management_Worker
     serial: "25%"
   ...
   ```

3. Apply any customizations or modifications to the Ansible configuration, if required.

   If this is an initial install, DVS and LNet are initially configured to run on the High-Speed Network (HSN) with the ko2iblnd Lustre Network Driver and Lustre RPMs for Industry Standard (non-HPE Slingshot) NICs. If that is not what your system requires, the VCS configuration needs to be changed.

In the same way, if this is an upgrade and the DVS transport should change as part of the upgrade, VCS changes need to be made. It is **IMPORTANT** to note that changing the DVS transport in the context of a COS upgrade will cause system downtime with respect to DVS projected content (e.g. PE; analytics) until DVS client transports are made to match those of the servers. In particular, if your system was previously using the legacy default DVS transport of ksocklnd over the NMN and you have not already performed the COS 2.2 upgrade, your VCS will now reflect the new default transport: ko2iblnd over the HSN. If that is your situation and you do not wish to incur downtime with respect to DVS projected content, you should restore the previous transport by changing your config (though we strongly recommend upgrading the DVS transport to ko2iblnd at a later time for enhanced performance). On the other hand, you may ***want*** to alter your DVS transport if you are upgrading to COS 2.3 and HPE Slingshot NICs at the same time (though the same warnings about downtime apply).

For instructions on how to change your config, find the appropriate recipe in the ***Make the appropriate changes to the*** *group_vars* ***directory...*** step of the ***Procedure To Change DVS Network Transport*** chapter of ***HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems***. To be clear, you only need to implement the VCS configuration changes from the appropriate recipe (as opposed to executing any of the additional steps outlined in that procedure).

Apply those changes to the VCS configuration now.

4. If there are any modifications, commit the changes.

   Do a “git status” to find any untracked files that need to be committed. Then use “git add” to add them. Then “git commit -a” will commit any outstanding changes.

   ```
   ncn-m001# git status
   ...
   ncn-m001# git add <FILES_TO_BE_COMMITTED>
   ncn-m001# git commit -a
   ```

5. Use the same crayvcs credentials (from previous steps) to push changes to the integration branch of the COS repository in VCS.

   If it is a newly created integration branch, perform the following:

   ```
   ncn-m001# git push --set-upstream origin integration
   Username for 'https://api-gw-service-nmn.local': crayvcs
   Password for 'https://crayvcs@api-gw-service-nmn.local': <password from preceding output>
   ```

   Otherwise, perform the following:

   ```
   ncn-m001# git push origin integration
   ```

6. Identify the commit hash for this branch. This will be used later when creating the CFS configuration layer.

   ```
   ncn-m001# git rev-parse --verify HEAD
   <== commit hash output ==>
   ```

7. Store the commit hash for later use.

   ```
   ncn-m001# export COS_CONFIG_COMMIT_HASH=<commit hash output>
   ```

8. Return to your working directory.

   ```
   ncn-m001# cd ..
   ```

COS configuration data is now properly defined in the integration branch of the `cos-config-management` repo in VCS. It will be used when performing the operations described in Configure COS - NCN Content and Configure COS - Compute Content sections.

6 Configure COS - NCN Content

The following steps describe how to use the NCN Personalization infrastructure to install and update DVS and related COS software on the NCN Workers. These steps are necessary to get the Workers ready to DVS project root file systems and other content to the Compute Nodes(CNs) and the User Access Nodes(UANs).

Ensure that the preceding Configure COS-VCS Content section has been completed so that the proper configuration data exists to configure COS on NCNs.

6.1 Perform NCN Personalization

1. For an upgrade only, create and run a new CFS configuration for the COS upgrade on all workers. Skip to step 2 if you are doing a fresh install.

   a. Create a new CFS configuration for upgrading COS with the latest COS config commit hash.

   The CFS configuration includes the `ncn-upgrade.yml` playbook which defines the tasks to be executed to upgrade COS.

   1. Determine if the CFS configuration for upgrading COS already exists.

      ```
      ncn-m001# cray cfs configurations describe cos-upgrade-2.3.XX \
      --format json | jq ". | {layers}" > cos-upgrade-2.3.XX.json
      ```

      If the configuration exists, the cos-upgrade-2.3.XX.json file will be created and populated with a previously defined configuration layer. If it does not exist, the file will be created empty and the command will respond with an error. This error can be ignored.

   2. Edit and update the cos-upgrade-2.3.XX.json file so that it contains the COS upgrade configuration layer information.

      Insert a Slingshot Host Software layer if it does not exist. A Slingshot Host Software layer must be defined before any COS layer. See the "Operational Activities" section in the ***Slingshot Operations Guide (Customer)***.

      The `COS_CONFIG_COMMIT_HASH` can be obtained as shown in steps 6 and 7 in the VCS Configuration section.

      The JSON file should now resemble the following:

      ```
      ncn-m001# cat cos-upgrade-2.3.XX.json
      {
        "layers": [
          {
            "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
            slingshot-host-software-config-management.git",
            "commit": "<SHS-COMMIT-HASH>",
            "name": "<SHS-INTEGRATION-XX>",
            "playbook": "<SHS-DEFINED-PLAYBOOK>"
          },
          {
            "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
            "commit":"<COS_CONFIG_COMMIT_HASH>",
            "name": "<COS-INTERGRATION-2.3.XX>",
            "playbook":"ncn-upgrade.yml"
          }
        ]
      }
      ```

   3. Upload the new configuration into CFS.

      ```
      ncn-m001# cray cfs configurations update cos-upgrade-2.3.XX \
      --file cos-upgrade-2.3.XX.json --format json
      {
        "lastUpdated": "2021-12-05T18:51:01Z",
        "layers":[
          {
            "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
            slingshot-host-software-config-management.git",
            "commit": "<SHS-COMMIT-HASH>",
            "name": "<SHS-INTEGRATION-XX>",
            "playbook": "<SHS-DEFINED-PLAYBOOK>"
          },
          {
            "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
            "commit":"<COS_CONFIG_COMMIT_HASH>",
            "name":"<COS-INTEGRATION-2.3.XX>",
      ```

```
        "playbook":"ncn-upgrade.yml"
      }
        ..
    ]
   ],
   "name": "cos-upgrade-2.3.XX"
 }
```

b. Create a CFS session to run the CFS configuration for upgrading COS.

```
ncn-m001# cray cfs sessions create --name cos-upgrade-2.3.XX \
--configuration-name cos-upgrade-2.3.XX \
--ansible-limit Management_Worker
```

This command creates a single CFS session that will upgrade COS on all hosts defined using the `hosts` keyword in the `ncn-upgrade.yml` playbook. By default, the value is set to `Management_Worker` which will result in the CFS session upgrading COS on all of the worker NCNs on the system. The –ansible-limit option which was set to `Management_Worker` ensures that the Slingshot Host Software (SHS) layer also runs only on the worker NCNs.

The CFS session name specified using –name must be unique. If the name was already used for a previous session, use a different unique name or delete the existing name using:

```
ncn-m001# cray cfs sessions delete cos-upgrade-2.3.XX
```

c. Monitor the progress of the CFS session for the COS layer.

```
ncn-m001# kubectl logs -c ansible-1 --tail 100 -f -n services \
--selector=cfsession=cos-upgrade-2.3.XX
```

Ansible plays, which are run by the CFS session, will upgrade COS on all of the worker NCNs on the system. During this procedure, the DVS and LNet services will be unloaded, upgraded, and loaded on each worker node using a rolling upgrade procedure.

d. If you encountered a problem with one of the worker nodes during the upgrade, and after you have resolved the problem, you can rerun the upgrade on that worker node by specifying the worker node's xname to the –ansible-limit option, as follows:

```
ncn-m001# cray cfs sessions create --name cos-upgrade-2.3.XX \
--configuration-name cos-upgrade-2.3.XX \
--ansible-limit <NCN_XNAME>
```

2. For both upgrade and fresh installs, update the COS config commit hash in the NCN personalization CFS configuration.

The NCN personalization CFS configuration includes the `ncn.yml` and `ncn-final.yml` playbooks which define the tasks to be executed for both upgrade and fresh installs of COS.

The layer that includes the `ncn-final.yml` playbook should be added as the last layer at the end of the NCN personalization CFS configuration.

a. Download the current NCN personalization configuration.

```
ncn-m001# cray cfs configurations describe ncn-personalization \
--format json | jq ". | {layers}" > ncn-personalization.json
  {
    "layers": [
      {
        "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
        "commit": "6609d19394425a4c19907f673e87acf4789c2f5d",
        "name": "cos-integration-2.0.44",
        "playbook": "ncn.yml"
      },
      ..
```

b. Edit and update the ncn-personalization.json file so that it contains the COS configuration layer information.

For a fresh install, add two new layers to the ncn-personalization.json file at the end of the list of existing layers. The first layer is for COS and the second layer is for tasks that must run after PE and Analytics are installed.

Insert a Slingshot Host Software layer if it does not exist. A Slingshot Host Software layer must be defined before any COS layer. See the "Operational Activities" section in the ***Slingshot Operations Guide (Customer)***.

Update the value COS-INTEGRATION-2.3.XX.

The `COS_CONFIG_COMMIT_HASH` can be obtained as shown in steps 6 and 7 in the VCS Configuration section.

The JSON file should now resemble the following:

```
ncn-m001# cat ncn-personalization.json
{
  "layers":[
    ...
    {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
       slingshot-host-software-config-management.git",
       "commit": "<SHS-COMMIT-HASH>",
       "name": "<SHS-INTEGRATION-XX>",
       "playbook": "<SHS-DEFINED-PLAYBOOK>"
    },
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"<COS-INTEGRATION-2.3.XX>",
       "playbook":"ncn.yml"
    },
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"ncn-final",
       "playbook":"ncn-final.yml"
    }
  ]
}
```

For an upgrade, modify the existing COS layer and update the `name` and `commit` values. Modify or add the second layer for tasks that must run after PE and Analytics are upgraded. Update the `commit` value with the same `commit` value used for the COS layer.

Insert a Slingshot Host Software layer if it does not exist. A Slingshot Host Software layer must be defined before any COS layer. See the "Operational Activities" section in the ***Slingshot Operations Guide (Customer)***.

The `COS_CONFIG_COMMIT_HASH` can be obtained as shown in steps 6 and 7 in the VCS Configuration section.

The JSON file should now resemble the following:

```
ncn-m001# cat ncn-personalization.json
{
  "layers":[
    ...
    {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
       slingshot-host-software-config-management.git",
       "commit": "<SHS-COMMIT-HASH>",
       "name": "<SHS-INTEGRATION-XX>",
       "playbook": "<SHS-DEFINED-PLAYBOOK>"
    },
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"<COS-INTEGRATION-2.3.XX>",
       "playbook":"ncn.yml"
```

```
    },
    ...
    {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cpe-config-management.git",
       "commit": "dd614c46794d9c5d917fdae13098484807c2fca7",
       "name": "cpe-21.10-integration",
       "playbook": "pe_deploy.yml"
    },
    {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/analytics-config-management.git",
       "commit": "621b95e050e9f47fd0e9fa3f166a18e7cdadefd8",
       "name": "analytics-integration-1.1.24",
       "playbook": "site.yml"
    },
    ...
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"ncn-final",
       "playbook":"ncn-final.yml"
    }
  ]
}
```

c. Upload the new configuration into CFS.

```
ncn-m001# cray cfs configurations update ncn-personalization \
--file ncn-personalization.json --format json
{
  "lastUpdated": "2021-12-05T18:51:01Z",
  "layers":[
    ...
    {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
       slingshot-host-software-config-management.git",
       "commit": "<SHS-COMMIT-HASH>",
       "name": "<SHS-INTEGRATION-XX>",
       "playbook": "<SHS-DEFINED-PLAYBOOK>"
    },
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"<COS-INTEGRATION-2.3.XX>",
       "playbook":"ncn.yml"
    },
    ...
    {
       "cloneUrl":"https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit":"<COS_CONFIG_COMMIT_HASH>",
       "name":"ncn-final",
       "playbook":"ncn-final.yml"
    }
  ],
  "name":"ncn-personalization"
}
```

3. Set the desired configuration on the worker NCNs.

   a. Update the desired configuration for all worker NCNs.

```
ncn-m001# export NCN_WORKERS=$(cray hsm state components list --role Management \
--subrole Worker --format json | jq -r .Components[].ID)
ncn-m001# for xname in $NCN_WORKERS
do
    cray cfs components update --desired-config ncn-personalization --enabled true \
    --state '[]' --error-count 0 --format json $xname
done
```

After this command is issued, the CFS Batcher service will dispatch CFS sessions to configure the NCNs. For a fresh install, the COS software will be installed and configured on all worker nodes in parallel. For an upgrade, the installation and configuration has already been completed in step 1 of NCN personalization so this task should complete quickly. Each NCN is now associated with the desired CFS configuration, and that configuration will be applied every time the NCN boots.

b. Query the status of the NCN Personalization process. The status will be `pending` while the node is being configured by CFS, and will change to `configured` when the configuration has completed.

```
ncn-m001# export NCN_WORKERS=$(cray hsm state components list --role Management \
--subrole Worker --format json | jq -r .Components[].ID)
ncn-m001# for xname in $NCN_WORKERS
do
    cray cfs components describe $xname --format json \
    | jq -r ' .id+" status="+.configurationStatus'
done
x3000c0s25b0n0 status=pending
x3000c0s11b0n0 status=configured
x3000c0s9b0n0 status=configured
...
```

The NCN personalization step is complete and the NCNs are configured as specified in the ncn-personalization configuration layers when each node's status is `configured`.

4. Verify that COS was successfully configured on the worker NCNs.

   a. Verify that the worker is running new DVS and LNet software.

   ```
   ncn-m001# ssh ncn-w001
   ncn-w001# lsmod | grep -P 'lnet|dvs'
   ```

   b. Verify that new Lustre and DVS RPMs are installed. Ensure the RPMs that are currently installed on the worker NCNs match the RPMs that are part of the current COS release.

   Run the following commands to verify that DVS was upgraded successfully for the worker node.

   ```
   ncn-w001# rpm -qa | grep -P 'lustre|dvs|craytrace|orca' | sort
   ```

   Use the nq.sh helper script to list the DVS and Lustre RPMs that were included in the new COS release and compare with the installed RPMs. The nq.sh script queries Nexus and lists the assets for the repository provided as a command line argument. The nq.sh script can be found in the ~/cos-configuration-management/roles/cray_dvs_install/files directory of the ncn-m001 master NCN. The repository name is provided in the "Query Nexus and verify that repositories were created for COS" step of the "Run install.sh" section. You will want to use the "group" repository name. The group repository has a more generic name, cos-2.3-sle-15sp3 versus cos-2.3.66-sle-15sp3 for example.

   ```
   ncn-m001# ~/cos-configuration-management/roles/cray_dvs_install/files/nq.sh \
           cos-2.3-sle-15sp3 | grep -P 'lustre|dvs|craytrace|orca'
   ```

   If the modules are not listed for each worker node, and you have done the Section Perform NCN Personalization above, refer to "Managing Configuration with CFS" in the CSM documentation for NCN Personalization details.

At this point, the COS content has been installed/upgraded and configured on NCNs. If other HPE Cray EX software products are being installed in conjunction with COS, refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine what step to perform next. If other HPE Cray EX software products are not being installed at this time, continue to the next section of this document to configure COS compute content.

7 Use SAT to Perform Compute Node Operations

This document provides detailed instructions on how to build, configure and boot compute node images in the following sections:

- Configure COS - Compute Content
- Configure COS - Customize a COS Compute Image
- Boot COS

However, if System Admin Toolkit (SAT) version 2.2.16 or later is installed, the `sat bootprep` command can be used to perform these tasks more quickly and with fewer operations. Use `sat showrev` to determine which version of SAT is installed on the system.

It is highly recommended that `sat bootprep` be used to perform these tasks. If `sat bootprep` is used, the content in the aforementioned sections is mainly informational and does not have to be followed to perform these operations as `sat bootprep` performs them. `sat bootprep` does not initiate the actual compute node boot process; see Reboot Compute Nodes with COS for details on that procedure.

7.1 SAT Bootprep Usage

The "SAT Bootprep" section of the ***HPE Cray EX System Admin Toolkit (SAT) Guide*** provides information on how to use `sat bootprep` to create CFS configurations, build images with IMS, and create BOS session templates. To include COS software and configuration data in these operations, ensure that the `sat bootprep` input file includes content similar to that described in the following subsections.

NOTE: The `sat bootprep` input file will contain content for additional HPE Cray EX software products and not only COS. The following examples focus on COS entries only.

NOTE: Before using SAT Bootprep, refer to Set Root Password For Compute Nodes and verify that HashiCorp COS secret password is properly configured.

7.1.1 COS Configuration Content

The `sat bootprep` input file should contain sections similar to the following to ensure COS configuration data is used when configuring the compute image prior to boot and when personalizing compute nodes after boot. Replace `2.2.87` with the version of COS desired. The version of COS installed resides in the CSM product catalog and can be displayed with the `sat showrev` command.

The first COS layer is cos-compute.yml and is required. The second COS layer is cos-compute-last.yml and is optional. Its purpose is to enhance the runtime operation of the Low Noise Mode systemd service. This playbook must be the last playbook specified in the CFS configuration to ensure CPU Affinity masks are applied correctly to running system processes.

```
configurations:
- name: cos-config
  layers:
  - name: cos-compute-integration-2.2.87
    playbook: cos-compute.yml
    product:
      name: cos
      version: 2.2.87
      branch: integration

  ...

  - name: cos-compute-last-integration-2.2.87
    playbook: cos-compute-last.yml
    product:
      name: cos
      version: 2.2.87
      branch: integration
```

7.1.2 COS Compute Image Recipe Content

The `sat bootprep` input file should contain a section similar to the following to ensure the compute node image is built from a COS image recipe. Replace `cray-shasta-compute-sles15sp3.x86_64-2.2.35` with the name of the COS image recipe for the version of COS desired. The image recipe for a release of COS resides in the CSM product catalog and can be displayed with the `sat showrev` command.

```
images:
- name: cray-shasta-compute-sles15sp3.x86_64-2.2.35
  ims:
    is_recipe: true
    name: cray-shasta-compute-sles15sp3.x86_64-2.2.35
  configuration: cos-config
  configuration_group_names:
  - Compute
```

7.1.3 COS BOS Session Template Content

The `sat bootprep` input file should contain a section similar to the following to ensure the BOS session template for compute nodes is built using the COS image, configuration data, and kernel parameters. Replace `cray-shasta-compute-sles15sp3.x86_64-2.2.35` with the name of the COS image recipe for the version of COS desired. The image recipe for a release of COS resides in the CSM product catalog and can be displayed with the `sat showrev` command.

```
session_templates:
- name: cray-shasta-compute-sles15sp3.x86_64-2.2.35
  image: cray-shasta-compute-sles15sp3.x86_64-2.2.35
  configuration: cos-config
  bos_parameters:
    boot_sets:
      compute:
        kernel_parameters: ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN}
        node_roles_groups:
        - Compute
```

NOTE: `sat bootprep` does not initiate the actual compute node boot process; see Reboot Compute Nodes with COS for details on that procedure.

`sat bootprep` will ensure that COS compute images are built from the COS image recipe, are properly configured, and will be booted with the information provided in the BOS session template.

In order to support GPUs you also need to complete the following sections Upload GPU Vendor Supplied Content to Nexus and GPU SAT Bootprep Configuration. It is recommended you do this before running `sat bootprep` but it can be done separately as well.

If other HPE Cray EX software products are being installed in conjunction with COS, refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine what step to perform next. If other HPE Cray EX software products are not being installed at this time, continue to the next sections of this document if you want to perform these operations without using `sat bootprep`.

7.1.4 Upload GPU Vendor Supplied Content to Nexus

In order to support GPUs you will need to upload GPU content to Nexus and add additional sections to the `sat bootprep` input file detailed in the next section. In order for the appropriate configuration to take place during `sat bootprep` the GPU content in Nexus needs to be available before running the command. On the worker node there is a tool `gpu-nexus-tool` to help upload the correct content to Nexus for the vendor(s) you need to support. Below are the steps for a standard setup using the gpu-nexus-tool.

For more details on the gpu-nexus-tool please see the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems and the section GPU Support. Additionally the GPU Support section details manual steps that do not involve use of the `gpu-nexus-tool` or `sat bootprep`, how to do health checks once booted, and additional optional configuration.

1. If you are not on an air-gapped system you should run one or both of following commands. For more information about options run `gpu-nexus-tool -h`. This also works for subcommmands, for example, `gpu-nexus-tool upload -h`. If you would like more feedback you can utilize the `-S` flag to show underlying commands as they are being ran and `-V` to print all the output from the underlying commands as well. If you are on an air-gapped system you should refer to the “Enable GPU Support” section in the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems guide which details how to obtain the vendor RPMs supported by this COS release.

   It is only necessary to run the commands for the vendor you support. You can do both vendors simultaneously.

   For Nvidia:

   ```
   ncn-w001# gpu-nexus-tool repo upload -v Nvidia
   ```

For AMD:

```
ncn-w001# gpu-nexus-tool repo upload -v AMD
```

2. After upload completes you should double check all content is uploaded and available. It can take a few minutes for Nexus to make the content fully available after upload completes.

   ```
   ncn-w001# gpu-nexus-tool repo check -v <vendor>
   ```

3. If after a few minutes you are still missing a few packages use the `-m` option to save time and reattempt the upload for just the missing packages. You may need to use a longer timeout with the `-t` option for larger files. A shorter timeout is nice if you are experiencing frequent timeouts and are not timing out because the upload is too large to complete on time.

   ```
   ncn-w001# gpu-nexus-tool repo upload -m -t 1000 -v <vendor>
   ```

4. After you are done and the check has completed successfully you will want clear out the the RPMs that were downloaded to the worker node. If you utilized the flags `--download-dir` or provided the RPMs some other way and utilized `--upload-dir` you do not need to complete this step.

   ```
   ncn-w001# gpu-nexus-tool clean
   ```

7.1.5 GPU SAT Bootprep Configuration

The `sat bootprep` input file should append sections similar to the following in order to support GPU. These sections will add a configuration used for creating the CPS bind mount image based on the COS image. If running separately you can also create a separate bootprep file for just this step.

In order for the bootprep creation of gpu-image to be correct all vendor content that needs to be supported should be uploaded to Nexus prior to running `sat bootprep`.

In place of `<cos-image-name>` you will want to use the name defined for the COS compute image in a previous section of the bootprep file.

```
configurations:
- name: gpu-cps-bind-image-create
  layers:
  - name: cos-config-management-for-gpu
    playbook: gpu_customize_playbook.yml
    git:
      url: "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git"
      branch: integration
images:
  - name: gpu-image
    description: >
      The Nvidia CPS image.
    ims:
      name: <cos-image-name>
      is_recipe: false
    configuration: gpu-cps-bind-image-create
    configuration_group_names: [Compute]
```

While configuring a node the vendor GPU hardware will be detected and then the content will be mounted onto the compute node from the image named `gpu-image`. There is no longer any need for a separate BOS sessiontemplate as previously needed for GPU support prior to COS 2.3.

8 Configure COS - Compute Content

NOTE: These instructions are not necessary if `sat bootprep` is being used to build and configure COS compute images as described in the Use SAT to Perform Compute Node Operations section.

8.1 CFS Configuration

If you are performing a fresh install, you will be creating a new configuration for COS 2.3. If you are upgrading, a configuration already exists, you will be modifying an existing configuration - perform Section Retrieve And Modify An Existing Configuration For COS.

8.1.1 Create An Initial Configuration For COS

This procedure applies only to fresh installation of COS. Skip this section if you are upgrading an existing COS installation.

1. Create a new Configuration Framework Service (CFS) session configuration for COS.

   Create a JSON file as input to the CFS configurations CLI command.

   With COS 2.3, the Slingshot Host Software layer must exist before the COS layer. In addition, there are two COS layers. The first layer runs during image customization and the second layer runs only during node personalization.

   Do not attempt to cut and paste the following example. This will lead to an incorrectly formatted JSON syntax.

```
ncn-m001# cat cos-config-${COS_RELEASE}.json
   {
     "layers": [
       {
         "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
         slingshot-host-software-config-management.git",
         "commit": "<SHS-COMMIT-HASH>",
         "name": "<SHS-INTEGRATION-XX>",
         "playbook": "<SHS-DEFINED-PLAYBOOK>"
       },
       {
         "name": "cos-integration-2.3.XX",
         "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
         "playbook": "cos-compute.yml",
         "commit": "<COS_CONFIG_COMMIT_HASH>"
       },
  .. PLACE ALL OTHER PRODUCT LAYERS PRIOR TO FINAL COS LAYER ..
       {
         "name": "cos-integration-2.3.XX-last",
         "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
         "playbook": "cos-compute-last.yml",
         "commit": "<COS_CONFIG_COMMIT_HASH>"
       }
     ]
   }
```

   If other products have been installed on the system, add additional configuration layers to be applied during the CFS session. This configuration can be used for pre-boot image customization as well as post-boot node personalization. The clone URL, commit, and top-level play will be run for all configuration layers.

   You may add on other layers e.g. WLM, PE, Analytics etc. at this time. However, first ensure that COS installation is successful.

   NOTE If it is desired to place the compute node default route on the HSN, adding a UAN layer such as the following will perform that task. The example CFS layer listed would be inserted into the expanded CFS layer used for compute node image customization.

   Also note that the BICAN System Default Route must be set to 'CHN' in SLS for this UAN layer to function:

```
   {
       "name": "uan-integration-2.4.XX",
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/uan-config-management.git",
       "playbook": "site.yml",
       "commit": "<UAN_CONFIG_COMMIT_HASH>"
   }
```

   Configuration of 'CHN' on the compute nodes should result in a second IP address on the hsn0 interface and the default route being set on the 'CHN' gateway IP.

8.1.2 Retrieve And Modify An Existing Configuration For COS

This procedure applies to COS upgrades where an existing configuration exists. The existing configuration will likely include other Cray EX product entries. Update the COS entry as described in this section. The ***HPE Cray EX System Software Getting Started Guide*** provides guidance on how and when to update the entries for the other products.

1. Retrieve the existing configuration file.

```
ncn-m001# cray cfs configurations describe cos-config-2.0.XX --format json | jq ". | {layers}"
> cos-config-2.3.XX.json
    {
      "layers": [
        {
          "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
          "commit": "6609d19394425a4c19907f673e87acf4789c2f5d",
          "name": "cos-integration-2.0.XX",
          "playbook": "site.yml"
        },
        {
          "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/slurm-config-management.git",
          "commit": "4e934b9db15fa92bc8dcaf2d823b907b5143b55d",
          "name": "slurm-integration-0.1.0",
          "playbook": "site.yml"
        },
    ..
```

2. Modify existing configuration in cos-config-2.3.XX.json for COS 2.3 upgrade.

   Update the “cos-integration-2.0.XX” layer to reflect “cos-integration-2.3.XX”

   - Insert Slingshot Host Software layer, which must be defined before any COS layer. See the “Operational Activities” section in the ***Slingshot Operations Guide (Customer)***.
   - Update key “commit” with COS_CONFIG_COMMIT_HASH above for first COS layer
   - Update key “name” with new name that represent this COS 2.3 upgrade for first COS layer
   - Update key “playbook” with new name “cos-compute.yml” for first COS layer
   - Replicate first COS layer and insert as the final layer with the key “playbook” set to “cos-compute-last.yml”
   - The second COS layer with the key “playbook” set to “cos-compute-last.yml” is optional and must be the last playbook specified in the CFS configuration.

   The updated JSON file looks like:

```
    {
      "layers": [
        {
          "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
          slingshot-host-software-config-management.git",
          "commit": "<SHS-COMMIT-HASH>",
          "name": "<SHS-INTEGRATION-XX>",
          "playbook": "<SHS-DEFINED-PLAYBOOK>"
        },
        {
          "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
          "commit": "<COS_CONFIG_COMMIT_HASH>",
          "name": "<COS-INTEGRATION-2.3.XX>",
          "playbook": "cos-compute.yml"
        },
        {
          "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/slurm-config-management.git",
          "commit": "4e934b9db15fa92bc8dcaf2d823b907b5143b55d",
          "name": "slurm-integration-0.1.0",
          "playbook": "site.yml"
        },
```

```
..
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": "<COS_CONFIG_COMMIT_HASH>",
      "name": "<COS-INTEGRATION-2.3.XX-LAST>",
      "playbook": "cos-compute-last.yml"
    }
   ],
   "name": ..
```

NOTE If it is desired to place the compute node default route on the HSN, adding a UAN layer such as the following will perform that task. The example CFS layer listed would be inserted into the expanded CFS layer used for compute node image customization.

Also note that the BICAN System Default Route must be set to 'CHN' in SLS for this UAN layer to function:

```
{
    "name": "uan-integration-2.4.XX",
    "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/uan-config-management.git",
    "playbook": "site.yml",
    "commit": "<UAN_CONFIG_COMMIT_HASH>"
}
```

Configuration of 'CHN' on the compute nodes should result in a second IP address on the hsn0 interface and the default route being set on the 'CHN' gateway IP.

8.1.3 Update Configuration Framework Service (CFS) Session With New COS Configuration

1. Update the configuration using the new configuration JSON file.

   This is an example output of just a minimal COS layer with a required SHS layer:

```
ncn-m001# cray cfs configurations update \
cos-config-${COS_RELEASE} --file ./cos-config-$
{COS_RELEASE}.json \
--format json
{
  "lastUpdated": "2020-11-05T17:05:58Z",
  "layers": [
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/
      slingshot-host-software-config-management.git",
      "commit": "<SHS-COMMIT-HASH>",
      "name": "<SHS-INTEGRATION-XX>",
      "playbook": "<SHS-DEFINED-PLAYBOOK>"
    },
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": "5cd59022d5e4d953a4124cff73064a810cb7ed4f",
      "name": "cos-integration-2.3.XX",
      "playbook": "cos-compute.yml"
    },
    {
      "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
      "commit": "5cd59022d5e4d953a4124cff73064a810cb7ed4f",
      "name": "cos-integration-2.3.XX-last",
      "playbook": "cos-compute-last.yml"
    }
  ],
  "name": "cos-config-2.3.XX"
}
```

At this point, COS configuration content has been updated in CFS. If other HPE Cray EX software products are being installed in conjunction with COS, refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine what step to perform next. If other HPE Cray EX software products are not being installed at this time, continue to the next section of this document to build a COS compute image.

8.2 Build a COS Compute Image

The following steps describe how to build a COS compute image from the COS image recipe. If there is a need to modify the COS image recipe, refer to the "Upload and Register an Image Recipe" section of the CSM documentation for details.

1. Ensure that the `slingshot-host-software` product is installed. This can be verified by using the `sat showrev` command or by examining the CSM product catalog. `slingshot-host-software` provides Slingshot kernel modules and related software compiled to match the COS kernel environment. The version of `slingshot-host-software` installed should correspond to the version of COS being installed. For example, if you are installing `cos-2.3.51`, the `slingshot-host-software` product should be named similar to `slingshot-host-software-1.2.1-18-cos-2.3.51`. The `slingshot-host-software` version may be different from 1.2.1-18, but the "cos-2.3.51" string should be present.

2. Identify the IMS public key.

   If the system already has an IMS public key, use that for the build. Check by running `cray ims public-keys list`. If one does not exist, create a new one. Export SSH Key for IMS.

   ```
   ncn-m001# cray ims public-keys create --name "my public key" --public-key \
   ~/.ssh/id_rsa.pub

   [[results]]
   created = "2020-11-18T17:53:28.163021+00:00"
   id = "4a629135-9ab6-4164-9fa2-86bd018a4c61"
   name = "my public key"
   ...

   ncn-m001# export IMS_PUBLIC_KEY_ID=4a629135-9ab6-4164-9fa2-86bd018a4c61
   ```

3. Verify `IMS_RECIPE_ID` has the expected value set previously in the "Run install.sh" section of this document.

   ```
   ncn-m001# echo $IMS_RECIPE_ID
   ```

4. Create an IMS image from the recipe.

   The results of this command should have a status value of `creating`. Store the job `id` and `kubernetes_job` values from the output for later use. The argument to –image-root-archive-name is just a plain text string and should be easy to identify later. `sles15sp3-recipe-example-image` is just our example name plain text string.

   ```
   ncn-m001# cray ims jobs create --job-type create \
   --image-root-archive-name sles15sp3-recipe-example-image \
   --artifact-id $IMS_RECIPE_ID \
   --public-key-id $IMS_PUBLIC_KEY_ID \
   --enable-debug False
   ```

   ```
   ...
   - initrd_file_name = "initrd"
     status = "creating"
     artifact_id = "a89a1013-f1dd-4ef9-b0f8-ed2b1311c98a"
     enable_debug = false
     kubernetes_configmap = "cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-configmap"
     kubernetes_service = "cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-service"
     kubernetes_job = "cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-create"
     job_type = "create"
     id = "52e2d6f6-977f-4b91-b0fb-174dc7e195e2"
     created = "2020-11-06T20:40:51.823126+00:00"
     kubernetes_namespace = "ims"
     public_key_id = "4a629135-9ab6-4164-9fa2-86bd018a4c61"
     kernel_file_name = "vmlinuz"
   ```

```
            build_env_size = 10
            image_root_archive_name = "skern-sles15sp3-image"
          ...
          kubernetes_job = "cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-create"
          job_type = "create"
          id = "52e2d6f6-977f-4b91-b0fb-174dc7e195e2"
          ...
```

```
ncn-m001# export IMS_JOB_ID=52e2d6f6-977f-4b91-b0fb-174dc7e195e2
ncn-m001# export IMS_KUBERNETES_JOB=cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-create
```

Use kubectl to describe the image create job. We want to identify the pod doing the IMS image customization build.

```
ncn-m001# kubectl -n ims describe job $IMS_KUBERNETES_JOB
          ...
          Events:
            Type    Reason           Age    From            Message
            ----    ------           ----   ----            -------
            Normal SuccessfulCreate 9m41s  job-controller Created pod:
            cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-create-tnk92
```

```
ncn-m001# export POD=cray-ims-52e2d6f6-977f-4b91-b0fb-174dc7e195e2-create-tnk92
```

Check the Kubernetes job containers for progress. Each Kubernetes log check is a tail that will not end until the container has completed. Wait on the completion of each container in the order listed below. Once they have all completed, the admin will be able to obtain the customized image ID to be used for CFS image customization.

```
ncn-m001# kubectl -n ims logs -f $POD -c fetch-recipe
ncn-m001# kubectl -n ims logs -f $POD -c wait-for-repos
ncn-m001# kubectl -n ims logs -f $POD -c build-ca-rpm
ncn-m001# kubectl -n ims logs -f $POD -c build-image
ncn-m001# kubectl -n ims logs -f $POD -c buildenv-sidecar
```

Find and store the resultant ID listed in the IMS buildenv-sidecar container's log using `$IMS_JOB_ID` as input when describing the IMS job.

```
ncn-m001# cray ims jobs describe 52e2d6f6-977f-4b91-b0fb-174dc7e195e2
...
resultant_image_id = "e3ba09d7-e3c2-4b80-9d86-0ee2c48c2214"
...
```

```
ncn-m001# export RESULTANT_IMAGE_ID=e3ba09d7-e3c2-4b80-9d86-0ee2c48c2214
```

At this point, a COS compute image has been built but has not been configured. Continue to the next section of this document to customize the COS compute image.

9 Configure COS - Customize a COS Compute Image

NOTE: These instructions are not necessary if `sat bootprep` is being used to build and configure COS compute images as described in the Use SAT to Perform Compute Node Operations section.

Compute node customization most often applies configuration data from multiple HPE Cray EX products and not just COS. If additional HPE Cray EX products need to be configured before customizing the COS compute image, refer to the ***HPE Cray EX System Software Getting Started Guide*** to determine how to do that before returning to this section.

9.1 Set Root Password For Compute Nodes

1. Verify the existence of the HashiCorp COS secret password.

    a. Obtain the HashiCorp Vault root token.

```
ncn-m001# kubectl get secrets -n vault cray-vault-unseal-keys \
-o jsonpath='{.data.vault-root}' | base64 -d; echo
```

b. Log into the HashiCorp Vault pod.

```
ncn-m001# kubectl exec -itn vault cray-vault-0 -- sh
```

c. Once attached to the pod's shell, log into vault and read the secret/cos key by executing the following commands. If the secret is empty, "No value found at secret/cos" will be displayed.

```
pod# export VAULT_ADDR=http://cray-vault:8200
pod# vault login
pod# vault read secret/cos
```

d. If no value is found for this key, complete the following steps in another shell on the NCN management node.

1. Generate the password HASH for the root user. Replace 'PASSWORD' with a chosen root password.

```
ncn-m001# openssl passwd -6 -salt $(< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c4) PASSWORD
```

2. Take the HASH value returned from the previous command and enter the following in the vault pod's shell. Instead of HASH, use the value returned from the previous step. You must escape the HASH value with the single quote to preserve any special characters that are part of the HASH value. If you previously have exited the pod, repeat steps a-c above; there is no need to perform the vault read since the content is empty.

```
pod# vault write secret/cos root_password='HASH'
```

3. Verify the new hash value is stored.

```
pod# vault read secret/cos
...

pod# exit
```

9.2 Customize a COS Compute Image

1. Run a CFS image customization session.

```
ncn-m001# cray cfs sessions create --name cos-config-${COS_RELEASE} \
--configuration-name cos-config-${COS_RELEASE} --target-definition image \
--target-group Compute ${RESULTANT_IMAGE_ID} --format json
```

2. Poll the CFS sessions job for completion.

```
ncn-m001# cray cfs sessions describe cos-config-${COS_RELEASE} --format json \
| jq -r .status.session.status

ncn-m001# cray cfs sessions describe cos-config-${COS_RELEASE} --format json \
| jq -r .status.session.succeeded
```

3. Obtain image `result_id`.

```
ncn-m001# cray cfs sessions describe cos-config-${COS_RELEASE} --format json \
| jq -r .status.artifacts[].result_id
5f3fb8a7-4189-4d04-b0c3-eec98fdc6440
```

4. Store the CFS result_id.

```
ncn-m001# export IMAGE_ID=5f3fb8a7-4189-4d04-b0c3-eec98fdc6440
```

5. Query the manifest.json artifact created by CFS; the ETag value is needed for the Boot Orchestration Service (BOS) session template that will be created later in this procedure.

```
ncn-m001# cray artifacts describe boot-images ${IMAGE_ID}/manifest.json --format json
{
   "artifact":{
      "AcceptRanges":"bytes",
      "LastModified":"2021-03-30T01:20:33+00:00",
```

```
            "ContentLength":1147,
            "ETag":"\"151cd5effbda573d8d05b3429c150e62\"",
            "ContentType":"binary/octet-stream",
            "Metadata":{
               "md5sum":"151cd5effbda573d8d05b3429c150e62"
            }
         }
      }
```


6. Retrieve and save the Etag value.

   ```
   ncn-m001# export ETAG_VALUE=151cd5effbda573d8d05b3429c150e62
   ```

At this point, the COS compute image has been customized with configuration data from all applicable HPE Cray EX products. Continue to the next section of this document to boot the COS compute image.

10 Boot COS

This section details how to boot COS images on compute nodes, and provides CPS details that may be relevant to the upgrade process.

NOTE: These instructions are not necessary if `sat bootprep` is being used to create a BOS session template as described in the Use SAT to Perform Compute Node Operations section.

10.1 Remove CPS images that are no longer used

If you are performing an upgrade of COS, it may be beneficial to remove any images no longer being managed by CPS. Please refer to the "Clean up CPS Content" section in HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems for details.

10.2 CPS s3fs support

In COS 2.3, CPS has been updated to use s3fs on NCN worker nodes to cache images instead of creating a copy of them in /var/lib/cps-local/. If this is the first upgrade from pre-COS 2.3 to COS 2.3 or beyond, you need to follow the instructions in this section once (and only once). If the upgrade has already been done or if you are performing a fresh install, you can skip to the next step. As described in the previous step, removing unused CPS images before upgrading to CPS with s3fs support is strongly recommended.

When CPS with s3fs support is installed for the first time via an upgrade, it creates NCN-local copies of existing images in order to support references to existing images during the upgrade process. After all compute nodes and UANs have rebooted as part of the upgrade process, CPS should be configured to s3fs-only mode to stop creating NCN-local copies of images.

To update CPS to s3fs-only mode, the `kubectl edit ds/cray-cps-cm-pm -n services` command is used and environment variables in CPS containers are set. Refer to the "CPS s3fs Migration" section in the HPE Cray Operating System Administration Guide: CSM on HPE Cray EX Systems for additional details on how to perform this one-time upgrade operation.

10.3 Monitor Compute Node Console Logging Or Establishing Serial Connection

Refer to the "ConMan" section of the CSM documentation for details on how to access compute node logs and consoles in case problems are encountered when booting COS.

10.4 Create a Boot Session Template

The current list of boot parameters can be viewed using:

`cray artifacts get boot-images <IMAGE_ID>/boot_parameters <MY-BOOT-PARAMETERS>`

You can add new parameters and parameters to override parameters in the boot_parameters file by adding them in the list of kernel_parameters in the sessiontemplate file. Kernel parameters in the sessiontemplate are appended to the list from the boot_parameter file for booting.

Refer to the "CSM Operational Activities" manual's "Boot Orchestration" section for details on kernel boot parameters.

1. Construct a boot session template using the xnames of the compute nodes, the customized image ID, and the CFS session configuration name.

- For a fresh install, create a new boot session template using the example below.

  a. Insert ETAG_VALUE, IMAGE_ID, and the CONFIGURATION-NAME into the file.

```
{
  "boot_sets": {
    "compute": {
      "boot_ordinal": 2,
      "kernel_parameters": "ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN}",
      "network": "nmn",
      "node_roles_groups": [
        "Compute"
      ],
      "path": "s3://boot-images/<IMAGE_ID>/manifest.json",
      "rootfs_provider": "cpss3",
      "rootfs_provider_passthrough": "dvs:api-gw-service-nmn.local:300:hsn0,nmn0:0",
      "etag": "<ETAG_VALUE>",
      "type": "s3"
    }
  },
  "cfs": {
    "configuration": "<CONFIGURATION-NAME>"
  },
  "enable_cfs": true
}
```

- For an upgrade, obtain the running template and modify it

  a. Run the command

```
cray bos sessiontemplate describe cos-sessiontemplate-2.0.XX --format json >
cos-sessiontemplate-${COS_RELEASE}.json
```

  b. replace “etag”: value with the new ETAG_VALUE value

  c. replace “path”: “s3://boot-images/IMAGE ID/manifest.json” with the new IMAGE_ID value

  d. remove the session template name, which is the line with field “name” on it

  e. edit the previous line with “enable_cfs” to remove commas from the end

  f. The only required kernel parameters in the sessiontemplate are:

```
"kernel_parameters": "ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN}"
```

  Add any site specific kernel parameters if needed.

- Register the session template file with BOS.

  Any name or string may be used for the sessiontemplate `--name` in the following example. For more information about BOS templates, refer to BOS sections of the CSM documentation.

```
ncn-m001# cray bos sessiontemplate create --file cos-sessiontemplate-${COS_RELEASE}.json \
--name cos-sessiontemplate-${COS_RELEASE}
/sessionTemplate/cos-sessiontemplate-${COS_RELEASE}
```

10.5 Reboot Compute Nodes with COS

1. Create a BOS Session to reboot the compute nodes.

```
ncn-m001# cray bos session create --template-uuid \
cos-sessiontemplate-${COS_RELEASE} --operation reboot
```

   If the attempt to reboot the compute nodes fails, use the compute node console log to determine the state of the node. For example, the following (`WARN: readyForMount=false type=dvs ready=0 total=2`) shows that the DVS server nodes are not ready to provide content to the compute node:

```
2021-03-19 01:32:41 dracut-initqueue[420]: DVS: node map generated.
2021-03-19 01:32:41 katlas: init_module: katlas loaded, currently disabled
2021-03-19 01:32:41
2021-03-19 01:32:41 DVS: Revision: kbuild Built: Mar 17 2021 @ 15:14:05 against LNet 2.12.4
2021-03-19 01:32:41 DVS debugfs: Revision: kbuild Built: Mar 17 2021 @ 15:14:05 against LNet 2.12.4
2021-03-19 01:32:41 dracut-initqueue[420]: DVS: loadDVS: successfully added 10 new nodes into map.
2021-03-19 01:32:41 ed dvsproc module.
2021-03-19 01:32:41 DVS: message size checks complete.
2021-03-19 01:32:41 dracut-initqueuedvs_thread_generator: Watching pool DVS-IPC_msg (id 0)
2021-03-19 01:32:41 [420]: DVS: loaded dvs module.
2021-03-19 01:32:41 dracut-initqueue[420]: mount is:
/opt/cray/cps-utils/bin/cpsmount.sh -a api-gw-service-nmn.local -t dvs -T 300 -i
nmn0 -e 3116cf653e84d265cf8da94956f34d9e-181 s3://boot-images/
763213c7-3d5f-4f2f-9d8a-ac6086583f43/rootfs /tmp/cps
2021-03-19 01:32:41 dracut-initqueue[420]: 2021/03/19 01:31:01 cpsmount_helper Version: 1.0.0
2021-03-19 01:32:47 dracut-initqueue[420]: 2021/03/19 01:31:07 Adding content: s3://boot-images/
763213c7-3d5f-4f2f-9d8a-ac6086583f43/rootfs 3116cf653e84d265cf8da94956f34d9e-181 dvs
2021-03-19 01:33:02 dracut-initqueue[420]: 2021/03/19 01:31:22
WARN: readyForMount=false type=dvs ready=0 total=2
2021-03-19 01:33:18 dracut-initqueue[420]: 2021/03/19 01:31:38
WARN: readyForMount=false type=dvs ready=0 total=2
2021-03-19 01:33:28 dracut-initqueue[420]: 2021/03/19 01:31:48 2 dvs servers [10.252.1.7 10.252.1.8]
```

If this occurs, address any issues with the DVS servers and then repeat the BOS command.

For more information regarding the compute boot process, refer to the BOS and compute node boot sections of the CSM documentation.

2. SSH to a newly booted compute node using the HashiCorp COS secret password from the Section Set Root Password For Compute Nodes above.

   ```
   ncn-m001# ssh nid001000-nmn
   ..
   Password:
   ..
   Last login: Wed Mar 17 19:10:12 2021 from 10.252.1.12
   nid001000#
   ```

3. Log out of the compute node.

   ```
   nid000001# exit
   ```

4. Finish the typescript file started at the beginning of this procedure.

   ```
   ncn-m001# exit
   ```

10.6 Boot COS on HPE Apollo 6500 Gen10 Plus XL675d with Nvidia A100 GPU Tray

Due to the large number of devices on a fully populated XL675d Modular Accelerator Tray with Nvidia A100 GPUs and the complexities of determining appropriate PCI device address ranges across the various bridge devices, these nodes can experience PCI device startup failures that can prevent completion of early compute node boot. This is caused by the Linux kernel attempting to reallocate PCI bridge resources after BIOS completion. The reallocation can prevent correct initialization of devices on the accelerator tray.

To enable the correct boot of nodes with the XL675d GPU Tray, the kernel PCI bridge reallocation can be disabled. Disable reallocation by adding the following kernel parameter to the `kernel_parameters` variable of the COS compute node BOS session template:

```
pci=realloc=off
```

For example:

```
"kernel_parameters": "ip=dhcp quiet spire_join_token=${SPIRE_JOIN_TOKEN} pci=realloc=off"
```

See Create a Boot Session Template for information on creating, modifying, and changing parameters of session templates.

11 Documentation Conventions

Several conventions have been used in the preparation of this documentation.

- Markdown Format
- File Formats
- Typographic Conventions
- Annotations for how we identify sections of the documentation that do not apply to all systems
- Command Prompt Conventions which describe the context for user, host, directory, chroot environment, or container environment

11.1 Markdown Format

This documentation is in Markdown format. Although much of it can be viewed with any text editor, a richer experience will come from using a tool which can render the Markdown to show different font sizes, the use of bold and italics formatting, inclusion of diagrams and screen shots as image files, and to follow navigational links within a topic file and to other files.

There are many tools which can render the Markdown format to get these advantages. Any Internet search for Markdown tools will provide a long list of these tools. Some of the tools are better than others at displaying the images and allowing you to follow the navigational links.

11.2 File Formats

Some of the installation instructions require updating files in JSON, YAML, or TOML format. These files should be updated with care since some file formats do not accept tab characters for indentation of lines. Only space characters are supported. Refer to online documentation to learn more about the syntax of JSON, YAML, and TOML files.

11.3 Typographic Conventions

`This style` indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, and other software constructs.

(backslash) At the end of a command line, indicates the Linux shell line continuation character (lines joined by a backslash are parsed as a single line).

11.4 Annotations

This repository may change annotations, for now, under the MarkDown governance these are the available annotations.

You must use these to denote the right steps to the right audience.

These are context clues for steps, if they contain these, and you are not in that context you ought to skip them.

```
EXTERNAL USE
```

This tag should be used to highlight anything that an HPE Cray internal user should ignore or skip.

```
INTERNAL USE
```

This tag is used before any block of instruction or text that is only usable or recommended for internal HPE Cray systems.

External (GitHub or customer) should disregard these annotated blocks - they maybe contain useful information as an example but are not intended for their use.

11.5 Command Prompt Conventions

11.5.0.1 Host name and account in command prompts

The host name in a command prompt indicates where the command must be run. The account that must run the command is also indicated in the prompt. - The root or super-user account always has the # character at the end of the prompt - Any non-root account is indicated with account@hostname>. A non-privileged account is referred to as user.

11.5.0.2 Node abbreviations

The following list contains abbreviations for nodes used below

- CN - compute Node
- NCN - Non Compute Node
- AN - Application Node (special type of NCN)
- UAN - User Access Node (special type of AN)
- PIT - Pre-Install Toolkit (initial node used as the inception node during software installation booted from the LiveCD)

Prompt	Description
ncn#	Run the command as root on any NCN, except an NCN which is functioning as an Application Node (AN), such as a UAN.
ncn-m#	Run the command as root on any NCN-M (NCN which is a Kubernetes master node).
ncn-m002#	Run the command as root on the specific NCN-M (NCN which is a Kubernetes master node) which has this hostname (ncn-m002).
ncn-w#	Run the command as root on any NCN-W (NCN which is a Kubernetes worker node).
ncn-w001#	Run the command as root on the specific NCN-W (NCN which is a Kubernetes master node) which has this hostname (ncn-w001).
ncn-s#	Run the command as root on any NCN-S (NCN which is a Utility Storage node).
ncn-s003#	Run the command as root on the specific NCN-S (NCN which is a Utility Storage node) which has this hostname (ncn-s003).
pit#	Run the command as root on the PIT node.
linux#	Run the command as root on a linux host.
uan#	Run the command as root on any UAN.
uan01#	Run the command as root on hostname uan01.
user@uan>	Run the command as any non-root user on any UAN.
cn#	Run the command as root on any CN. Note that a CN will have a hostname of the form nid124356, that is “nid” and a six digit, zero padded number.
hostname#	Run the command as root on the specified hostname.
user@hostname>	Run the command as any non-root user son the specified hostname.

11.5.0.3 Command prompt inside chroot

If the chroot command is used, the prompt changes to indicate that it is inside a chroot environment on the system.

```
hostname# chroot /path/to/chroot
chroot-hostname#
```

11.5.0.4 Command prompt inside Kubernetes pod

If executing a shell inside a container of a Kubernetes pod where the pod name is $podName, the prompt changes to indicate that it is inside the pod. Not all shells are available within every pod, this is an example using a commonly available shell.

```
ncn# kubectl exec -it $podName /bin/sh
pod#
```

11.5.0.5 Command prompt inside image customization session

If using ssh during an image customization session, the prompt changes to indicate that it is inside the image customization environment (pod). This example uses $PORT and $HOST as environment variables with specific settings. When using chroot in this context the prompt will be different than the above chroot example.

```
hostname# ssh -p $PORT root@$HOST
root@POD# chroot /mnt/image/image-root
:/#
```

11.5.0.6 Directory path in command prompt

Example prompts do not include the directory path, because long paths can reduce the clarity of examples. Most of the time, the command can be executed from any directory. When it matters which directory the command is invoked within, the cd command is used to change into the directory, and the directory is referenced with a period (.) to indicate the current directory

Examples of prompts as they appear on the system:

```
hostname# cd /etc
hostname:/etc# cd /var/tmp
hostname:/var/tmp# ls ./file
hostname:/var/tmp# su - user
user@hostname:~> cd /usr/bin
user hostname:/usr/bin> ./command
```

Examples of prompts as they appear in this publication:

```
hostname # cd /etc
hostname # cd /var/tmp
hostname # ls ./file
hostname # su - user
user@hostname > cd /usr/bin
user@hostname > ./command
```

11.5.0.7 Command prompts for network switch configuration

The prompts when doing network switch configuration can vary widely depending on which vendor switch is being configured and the context of the item being configured on that switch. There may be two levels of user privilege which have different commands available and a special command to enter configuration mode.

Example of prompts as they appear in this publication:

Enter “setup” mode for the switch make and model, for example:

```
remote# ssh admin@sw-leaf-001
sw-leaf-001>  enable
sw-leaf-001#  configure terminal
sw-leaf-001(conf)#
```

Refer to the switch vendor OEM documentation for more information about configuring a specific switch.