Managing SMW/eLogin and ARM XC nodes

CUG 2018 Tutorial

Harold Longley, Eric Cozzi, Jeff Keopp
and Mark Ahlstrom, Cray Inc.

Agenda

- Managing eLogin nodes from SMW
- Migrating from CMC/eLogin to SMW/eLogin
- Managing ARM XC nodes
- Q&A

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

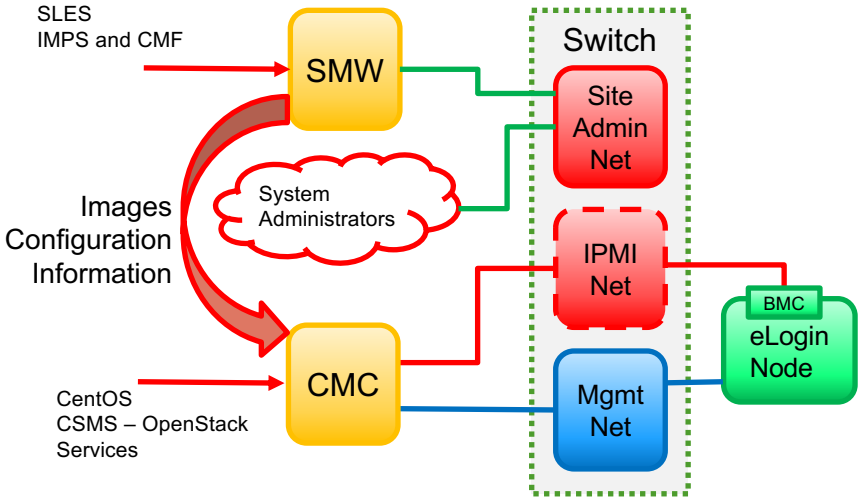
What is eLogin?

- **External login nodes (eLogin) enable user access to Cray XC system**
  - Log in
  - Access their home directories that are external to the XC system
  - Do application development using the Cray Programming Environment (PE)
  - Access the parallel file system, such as Lustre or Spectrum Scale (GPFS)
  - Submit jobs to the workload manager (WLM)
- **Available to users independently of Cray XC availability**
  - Local storage for operating system and persistent storage

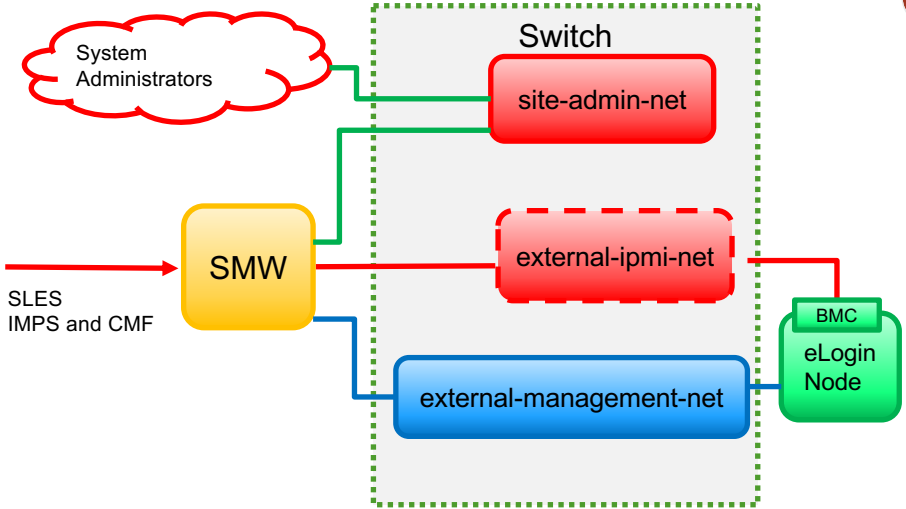
Copyright 2018 Cray Inc.

eLogin Management Topology

Before UP06



UP06



Requires 2 additional Ethernet ports on the SMW

- Single SMW will need an additional quad port Ethernet card
- HA SMW already has enough ports

Copyright 2018 Cray Inc.

SMW/eLogin Improved Usability

- **Reduced complexity**
  - Remove CMC server / OpenStack / CSMS component from the systems management infrastructure used with CLE 6.0 prior to UP06
  - Removes the CIMS server / Bright Cluster Manager used with CLE 5.2 and earlier
- **Consistency**
  - Use IMPS to build recipes into image roots and boot images
  - Use CMF to provide config sets and Ansible plays for node customization
  - Software is part of SMW and CLE media in SMW 8.0.UP06/CLE 6.0.UP06
    - No special eLogin media is needed for installation of the software
- **Scalability**
  - SMW management of external nodes is not limited to eLogin
  - Allows support of any external node type with images created from IMPS recipes
    - Purge servers
    - Data movers
    - Visualization servers

Copyright 2018 Cray Inc.

SMW/eLogin Increased Security

- **Network security**
  - Software firewall (iptables) on the SMW provides a layer of access protection from the eLogin nodes
- **Data Security**
  - Config set is sanitized before being pushed from the SMW to the elogin node
- **Console security**
  - Plain text passwords are not exposed as they were with some OpenStack-based operations
  - BMC (iDRAC) of eLogin should always be on a private network between the SMW and node to protect IPMI traffic

Copyright 2018 Cray Inc.

SMW/eLogin Reduced Deployment Time

- **Significantly faster deployment times compared to CMC-managed eLogin nodes**
  - Only one pass through POST/BIOS not two
    - saves 7-10 minutes/node
  - Very small initramfs used for PXE boot
  - SquashFS formatted image root for operating system and PE
- **60% faster boot time, 75% faster PE transfer in early testing!**

Copyright 2018 Cray Inc.

SMW/eLogin Improved Reliability/Availability

- **Diskfull eLogin nodes are not dependent on SMW**
  - Can be rebooted from disk in the absence of SMW
- **eLogin nodes managed by active SMW in an SMW HA pair**
  - No HA was available with CMC/eLogin
- **Ongoing image and configuration management without requiring an immediate reboot**
  1. Deliver new (sanitized) config set data to node and run cray-ansible (without booting)
  2. Stage new eLogin image, PE image, config set, and kernel parameters to node while node is booted, then boot when ready

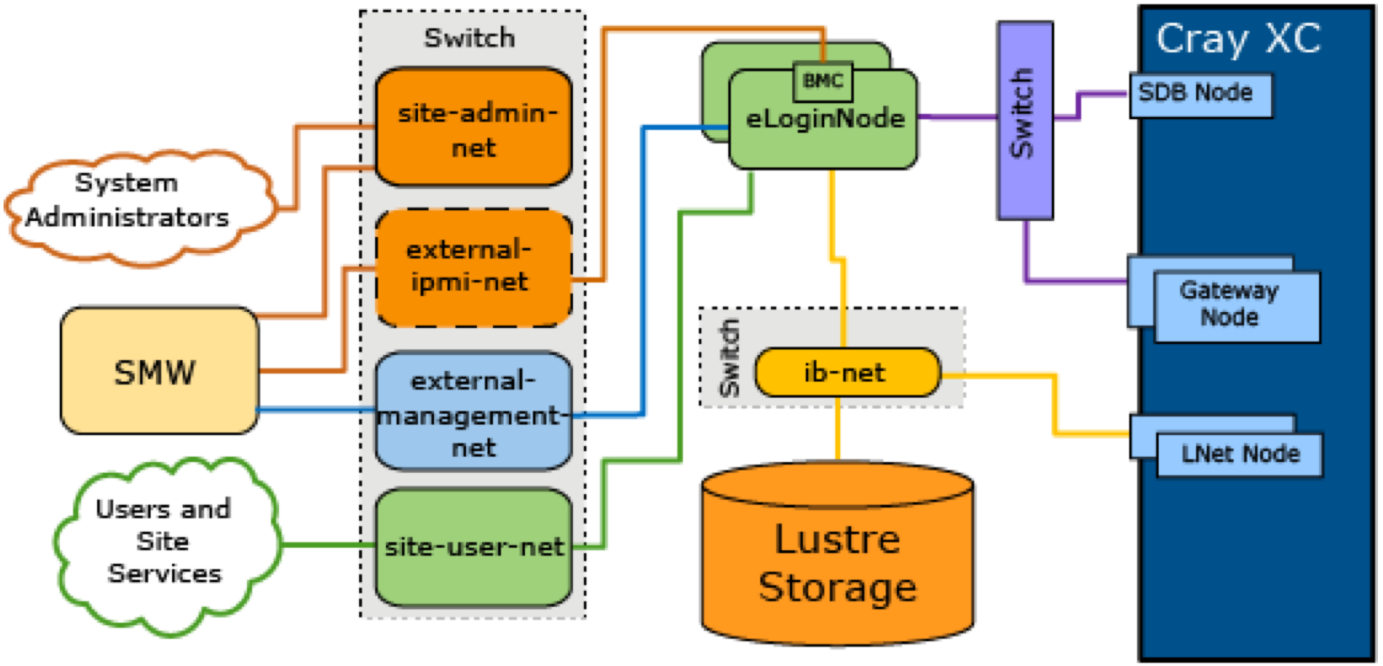
Copyright 2018 Cray Inc.

eLogin Networking to the Cray XC

- **4 supported topologies**
  - Differ in connection to the Gateway and SDB nodes of the Cray XC Series system
  - Gateway Node connection options
    - Direct connection over private network
    - Connect over the Site User network
  - SDB node connection options
    - Routed via Gateway Node
    - Direct connection over private network

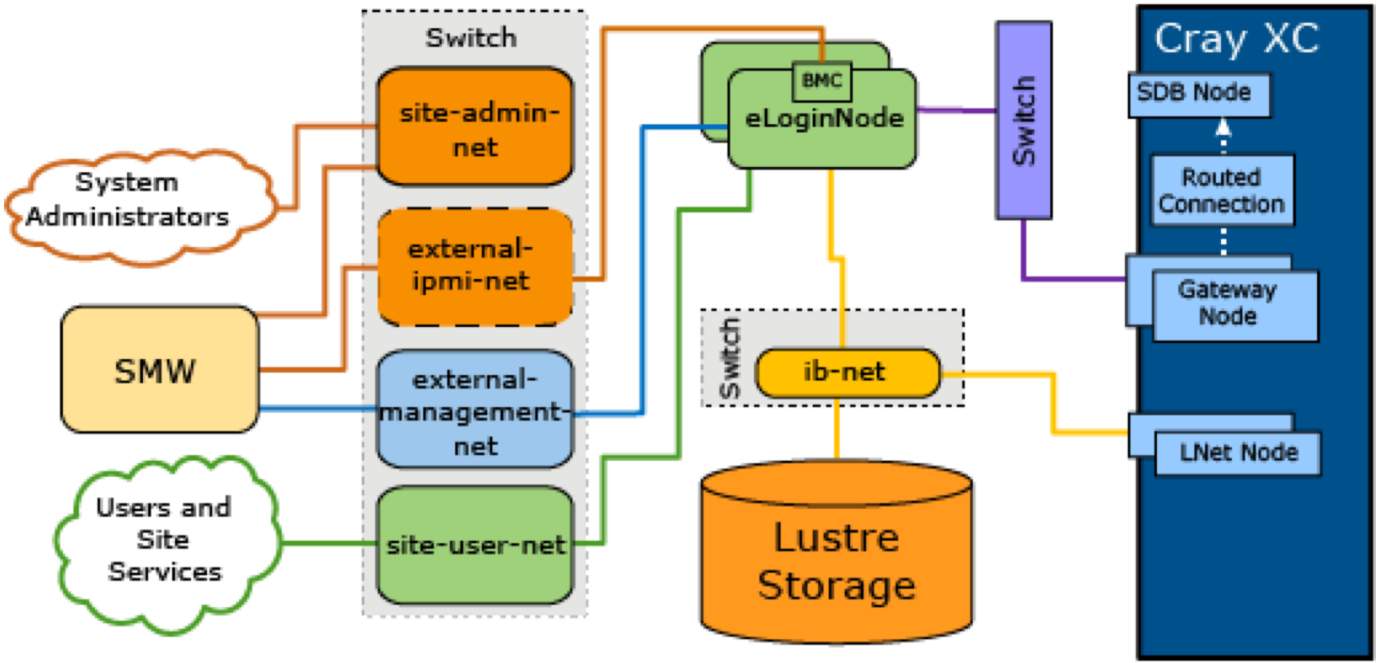
Copyright 2018 Cray Inc.

eLogin Nodes Connected to SDB Node and to Gateway Node via Switch



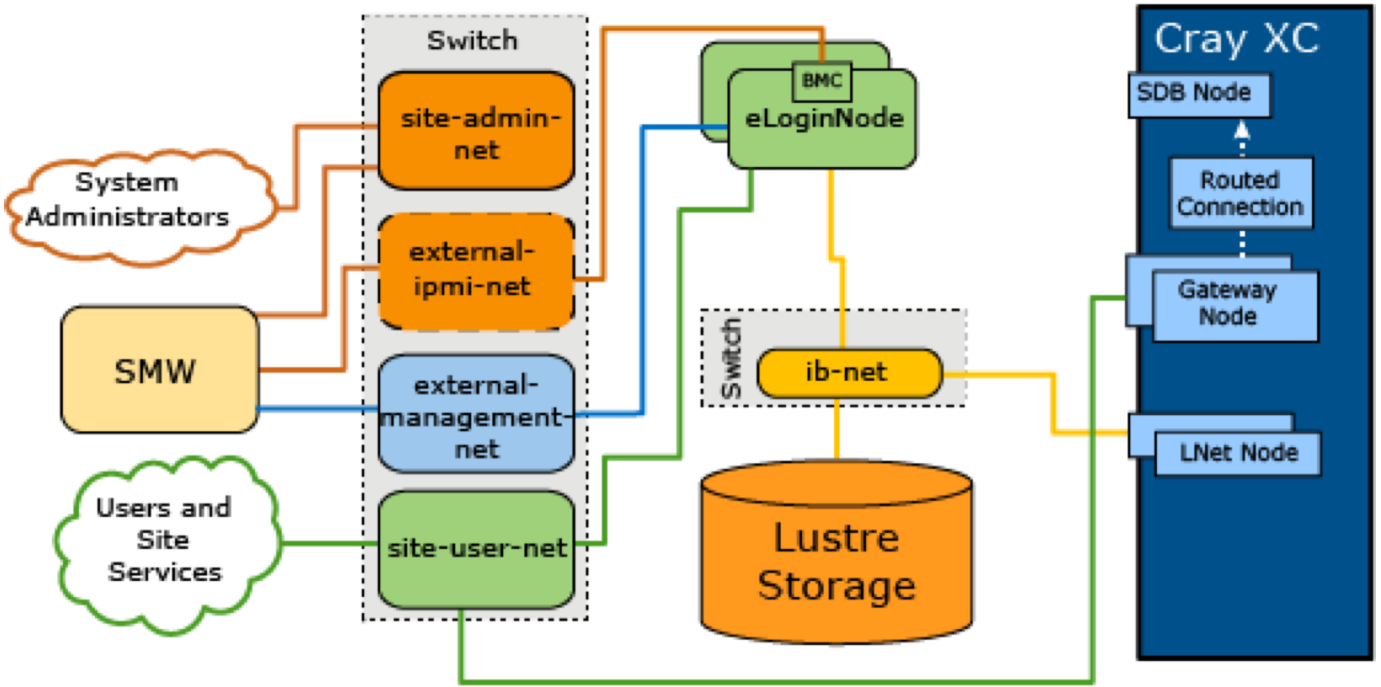
Copyright 2018 Cray Inc.

eLogin Nodes Connected to SDB Node via Routed Connection from Gateway Node and to Gateway Node via Switch



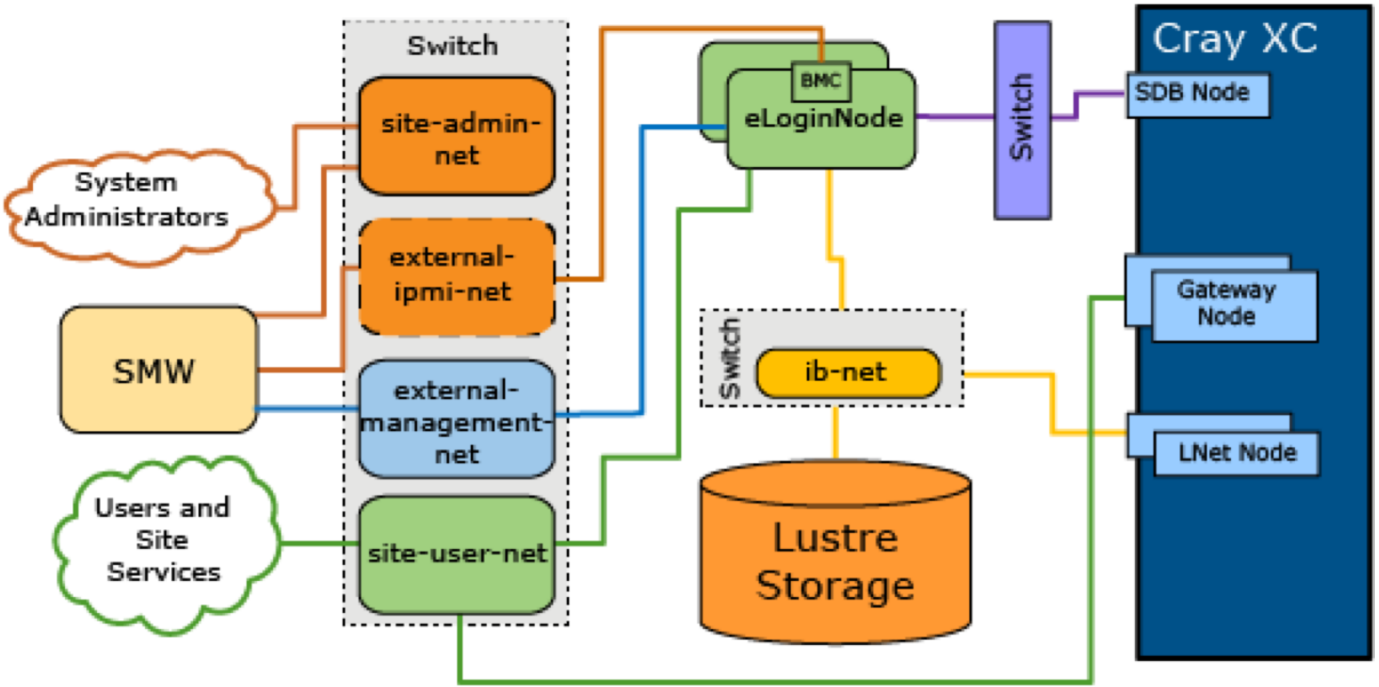
Copyright 2018 Cray Inc.

eLogin Nodes Connected to SDB Node via Routed Connection from Gateway Node and to Gateway Node via Site User Net



Copyright 2018 Cray Inc.

eLogin Nodes Connected to SDB Node via Switch and to Gateway Node via Site User Net



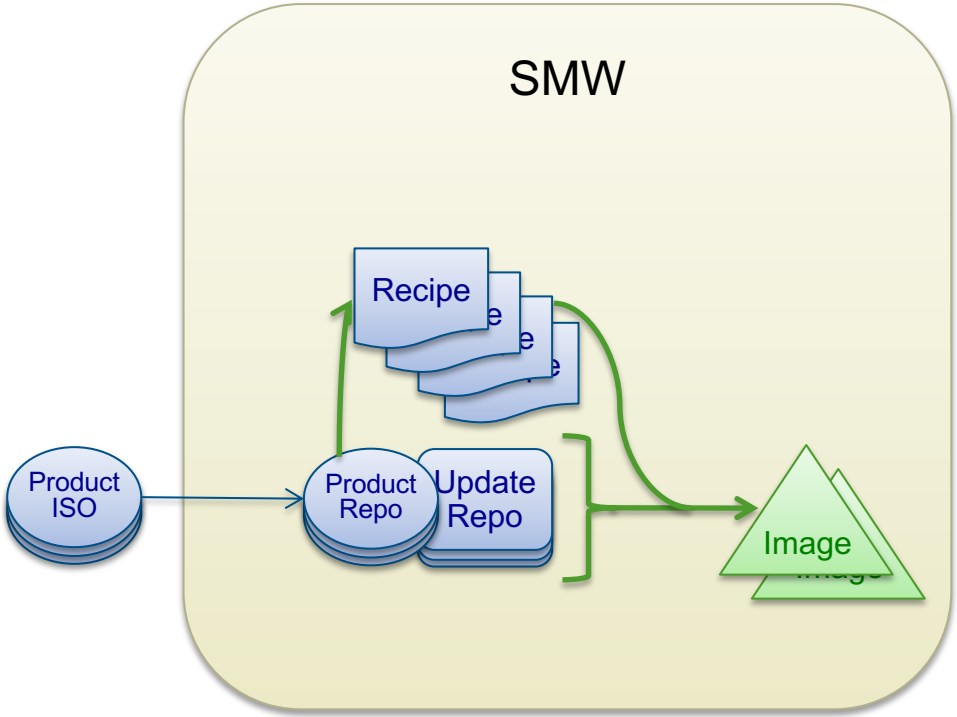
Copyright 2018 Cray Inc.

SMW/eLogin Initial Deployment Process

- **Admin tasks on SMW**
  - Register eLogin node information to control PXE boot process
  - Create eLogin image using IMPS recipes and tools
  - Modify global and CLE config set data for eLogin
  - Define disk layout for internal storage on eLogin node
  - Associate image, config set, kernel parameters, storage profile for each eLogin node
  - Configure eLogin node hardware to PXE boot
  - Power on eLogin node
  - Monitor the state of the node as it boots
  - Connect to the node console via interactive terminal
  - View logs or initiate a dump of the node

Copyright 2018 Cray Inc.

eLogin Image Management: Image Creation



- **Images are built on the SMW**
  - Package repositories are under: /var/opt/cray/repos
  - All rpm dependencies are resolved from these repositories
- **Prescribed by Image Recipes**
  - Images are under: /var/opt/cray/imps/image_roots
  - eLogin recipes based on CLE recipes plus eLogin specific packages
  - New site recipes can include Cray sub-recipes and site content
  - Images must be exported to the proper squashfs format

Copyright 2018 Cray Inc.

eLogin Configuration Sets

- **Same as Cray CLE images – Cray Management Framework (CMF)**
  - Configuration data is separate from the image
  - eLogin configuration data is in the same configuration set as Cray CLE
- **Configuration sets are created and maintained on the SMW**
  - Must be pushed to the eLogin node any time they are changed

`smw:/var/opt/cray/imps/config/sets/<config_set_name>`

- **More than one configuration set can exist to support alternative configurations**
  - Only one configuration set is active on a given eLogin node at a time

Copyright 2018 Cray Inc.

eLogin Configuration Sets

- **Config set on an eLogin node is at:**

  `elogin:/etc/opt/cray/config/current`

  `elogin:/etc/opt/cray/config/global`

- **/etc/opt/cray/config/current contains the following subdirectories**
  - ansible, config, dist, files
    - config subdirectory contains the configuration YAML files

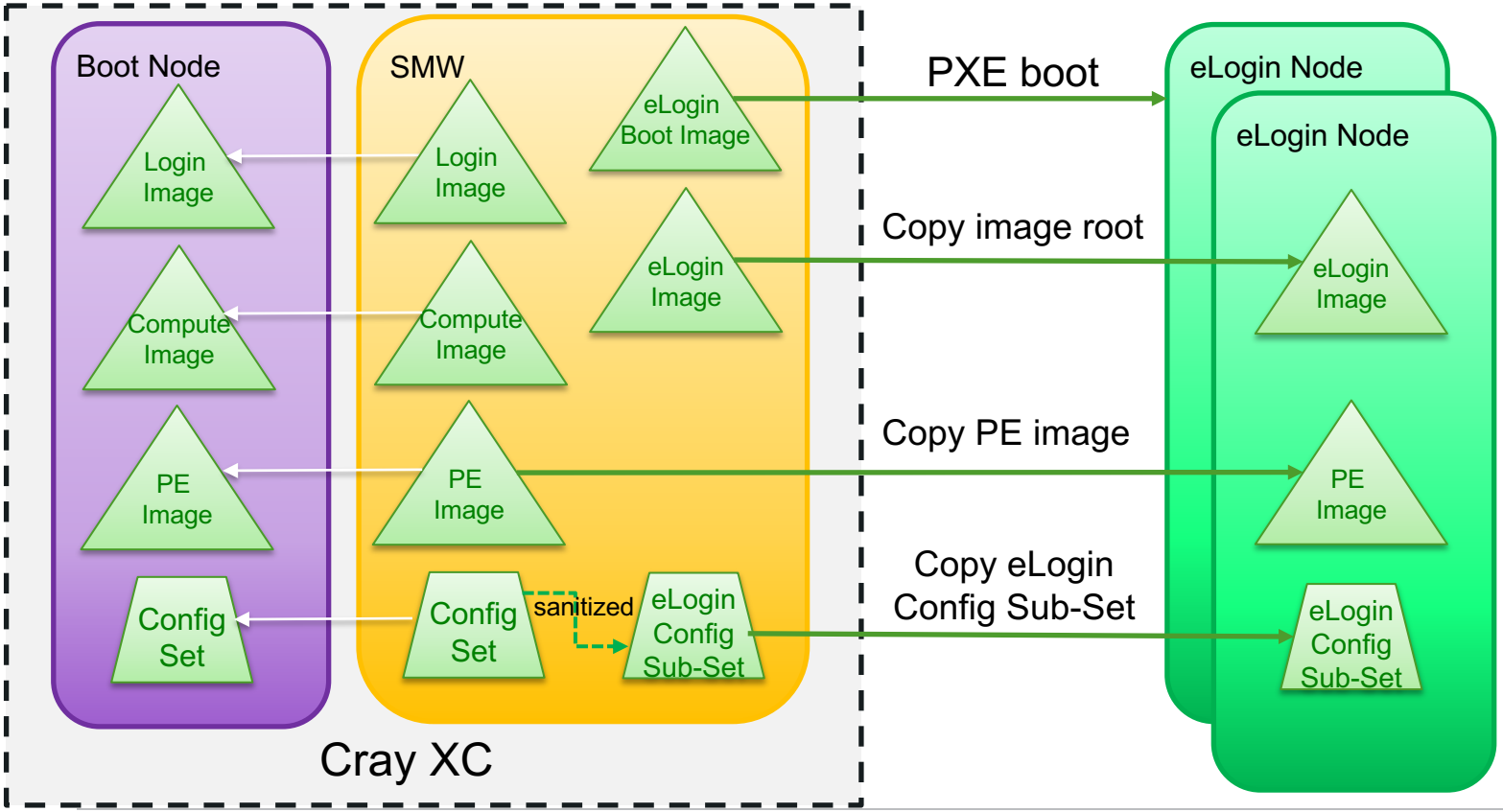
Copyright 2018 Cray Inc.

Managing eLogin Nodes: Extending Config

- **Cray's Configuration Management Framework (CMF) allows additional configuration tasks**
  - Add site-specific tasks in concert with Cray-provided Ansible boot-time execution
- **As with CLE, site Ansible plays may perform the following tasks on eLogin**
  - Start/stop services
  - Change crontab entries
  - Modify files
  - Copy files
  - Etc.

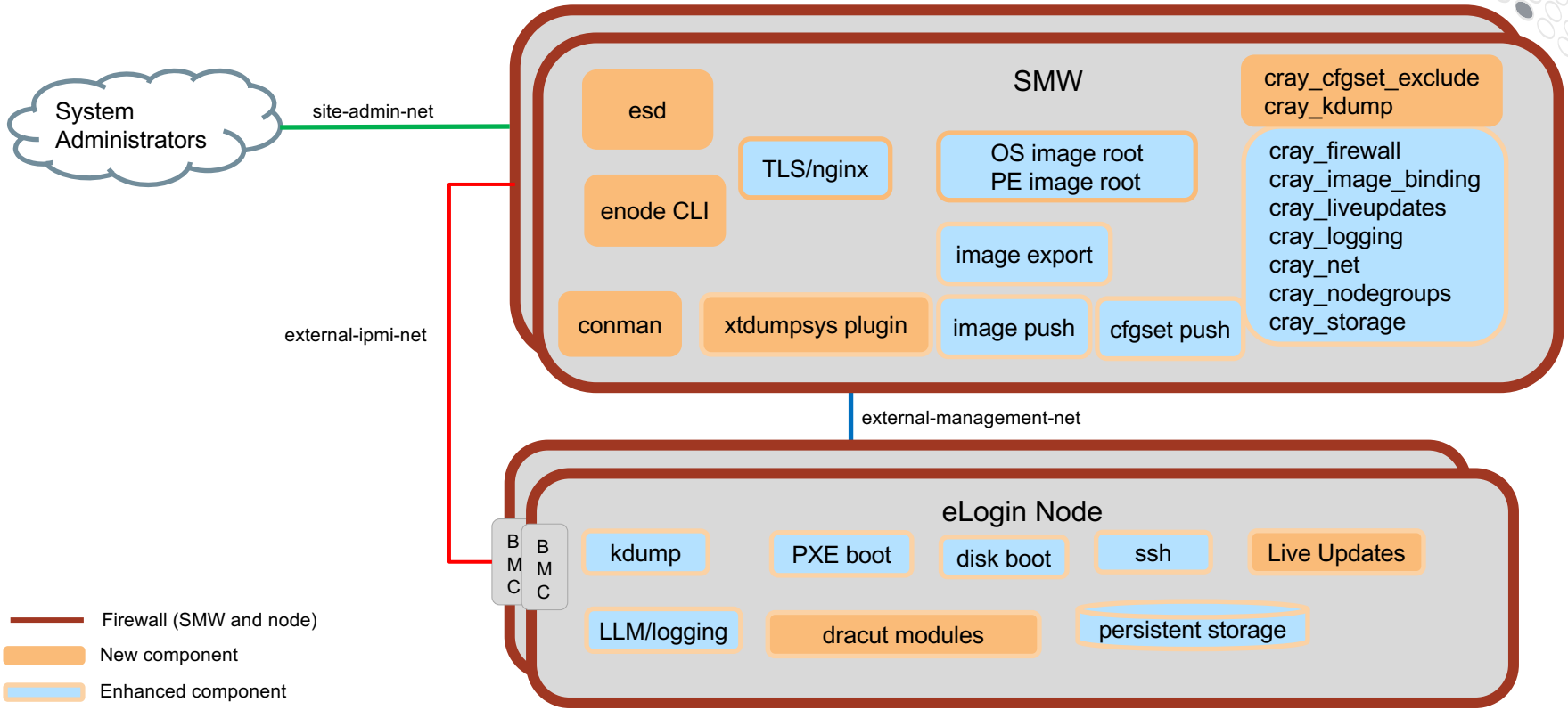
Copyright 2018 Cray Inc.

Image and Config Set Flow: SMW to eLogin



Copyright 2018 Cray Inc.

SMW/eLogin Management Topology



Copyright 2018 Cray Inc.

External State Daemon (esd)

- **Maintains external node registry**
- **Maintains external node state**
- **Maintains the following configuration for external nodes**
  - Console logging configuration
  - DHCP configuration
  - TFTP configuration
    - /opt/tftpboot/external/…
  - SMW /etc/hosts entries
- **Performs all node life cycle tasks**
  - boot, reboot, shutdown, status check
- **enode is the command line interface to esd**

Copyright 2018 Cray Inc.

enode command

- **Command line interface for external node registry**
  - list – list node information
  - create – adds a new node in the registry
  - enroll – adds new nodes in the registry from a csv file format
  - delete – deletes nodes from the registry
  - update – update registered node data
- **Command line interface for external node life-cycle**
  - boot – boots nodes
  - reboot – reboots nodes
  - status – display node status
  - shutdown – shuts down nodes
  - stage – stages changes in config sets and images to booted node

Copyright 2018 Cray Inc.

SMW Node Management commands

Task	Internal CLE Node	eLogin Node
Discover nodes	automatic with xtdiscover	manual
Enroll/register nodes	xtdiscover (registers nodes with HSS database)	enode create, enode enroll
Map image, config set, kernel parameters to nodes	cnode update	enode create, enode update
List node attributes	cnode list	enode list
Delete nodes	N/A	enode delete
Boot nodes	xtbootsys or xtcli boot	enode boot, enode reboot
Shut down nodes	xtbootsys or xtcli shutdown	enode shutdown
Check node status	xtcli status	enode status
Dump nodes	cdump and xtdumpsys	kdump and new plugin for xtdumpsys
Interactive console session	xtcon	conman

Copyright 2018 Cray Inc.

eLogin Node Booting Processes

- **BIOS boot**
  - Enables interaction with console for BIOS/System Setup
  - Pauses the boot on the BIOS/System Setup screen
  - Can interact with internal RAID controller to modify virtual disks
- **PXE boot**
  - Node boots over the connection to external-management-net using PXE (DCHCP/TFTP), to get the kernel, and initrd (boot image) from SMW
  - Then dracut scripts in initrd and cray-ansible assist with transfer
    - X.509 certificate for node
    - SSH host keys for node
    - Storage profile used to partition internal disks
    - Full image root (squashfs format)
    - global and CLE config sets
    - PE image root (squashfs format)
  - Must be PXE booted for initial deployment

Copyright 2018 Cray Inc.

eLogin Node Booting Processes

- **Disk boot**
  - Node boots from local storage prepared via earlier PXE boot
- **Staged boot**
  - Transfer these to node while node is booted
    - image root , PE image, config sets, kernel parameters
  - Choose when to reboot node to activate transferred items
  - Minimizes node down time because updates no longer include network file transfer time

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- **Enhancements to IMPS**
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

Enhancements to IMPS

- recipes
- image create
- image export
- imgbuilder
- image push using node_groups

Copyright 2018 Cray Inc.

SMW/eLogin Image Recipes

- **New recipes that support SMW control of the eLogin nodes**
  - elogin-smw-large_cle_6.0up06_sles_12sp3_x86-64_ari
  - elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari
- **Show subrecipes**
  - elogin-smw-large includes elogin-smw

```
smw# recipe show --fields recipes \
elogin-smw-large_cle_6.0up06_sles_12sp3_x86-64_ari \
elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari

elogin-smw-large_cle_6.0up06_sles_12sp3_x86-64_ari:
  recipes:    elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari
elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari:
  recipes:    seed_common_6.0up06_sles_12sp3_x86-64
```


Copyright 2018 Cray Inc.

SMW/eLogin Package Collection

- **New package collection**

elogin-cray-base_cle_6.0up06_sles_12sp3

smw# **recipe show --fields package_collections **
elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari
elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari:
package_collections:
elogin-cray-base_cle_6.0up06_sles_12sp3
intel-firmware_cle_6.0up06_sles_12sp3
login-small-external_cle_6.0up06_sles_12sp3

Copyright 2018 Cray Inc.

SMW/eLogin rpms

- **New rpms provide dracut support on the eLogin node**

smw# **pkgcoll show elogin-cray-base_cle_6.0up06_sles_12sp3**

elogin-cray-base_cle_6.0up06_sles_12sp3:

name: elogin-cray-base_cle_6.0up06_sles_12sp3

packages:

cray-dracut-external

cray-esd-node-client

Copyright 2018 Cray Inc.

SMW/eLogin Image Root

- **Create a new image root from SMW/eLogin recipe**

smw# **image create -r elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari **
elogin-smw-integration_cle_6.0.UP06-build6.0.6038_sles_12sp3-created20171103

Copyright 2018 Cray Inc.

image export Command

- **New format option to produce squashfs image**

  smw# **image export --format squashfs $IMAGE_NAME**
- **Creates a squashfs image in a directory at /var/opt/cray/imps/boot_images/$IMAGE_NAME**
  - Creates a squashfs file and an .imps_Image_metadata file in that directory
- **The exported image can be pushed to an elogin node**
  - You can use an existing squashfs with --prefer-existing

  smw# **image push --prefer-existing $IMAGE_NAME --format \ squashfs -d elogin1**
  - Or ignore any existing squashfs, generate a new squashfs before pushing it to the node

  smw# **image push $IMAGE_NAME --format squashfs -d elogin1**

Copyright 2018 Cray Inc.

imgbuilder Command

- **Imgbuilder can automatically build eLogin images at the same time as other images needed for CLE nodes**
  - cray_image_groups.yaml has new export_format
  - Any acceptable value for the format to “image export --format” can be used as export_format
    - Only cpio and squashfs will be needed for CLE and SMW/eLogin
  - Specifying cpio export format using .cpio extension on the “dest” line is deprecated, but still supported

Copyright 2018 Cray Inc.

cray_image_groups.yaml with export_format

```
cray_image_groups:
  default:
      - recipe: "admin_cle_6.0up06_sles_12sp3_x86-64_ari"
        dest: "admin{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp3-created{date}"
        export_format: "cpio"
        nims_group: "admin"

      - recipe: "elogin-smw_cle_6.0up06_sles_12sp3_x86-64_ari"
        dest: "elogin-smw{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp3-created{date}"
        export_format: "squashfs"
```


Copyright 2018 Cray Inc.

`image push/sqpush` Enhanced to Support Node Groups

- **New `--node-group/-g` option to `image push` (or `image sqpush`)**
  - Specify one or more node groups to which the config set should be pushed
  - Can be specified along with `--dest/-d` option to specify node groups and individual hosts
- **New `--ref-config-set/-r` option specifies config set to use when looking up node groups**
  - Required for node groups lookup when the `-g` option is used
- **`--prefer-existing` will use an existing exported squashfs image**
- **`--persist-conversion` will save the exported squashfs image in `/var/opt/cray/imps/boot_images/` for later pushing**

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- **Enhancements to CMF**
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

Enhancements to CMF

- Config set
- cfgset push using node_groups
- cray_cfgset_exclude
- cfgset push with exclude profiles
- Storage profile
- PE image binding
- Scalable services external tier1
- Simple Sync

Copyright 2018 Cray Inc.

Configuration Sets: Services Shared with Cray CLE

- **Config data affecting both CLE and eLogin nodes:**
  - Bold are new or changed for SMW/eLogin
  - cray_auth – user authentication settings – LDAP, NIS, etc.
  - **cray_firewall – firewall enablement**
  - **cray_liveupdates –** enables zypper update on node from repositories on SMW **to nodes in external_tier1_nodes**
  - cray_local_users – local users configuration
  - **cray_logging –** centralized logging settings **applied to nodes in external_tier1_nodes**
  - cray_ssh – SSH settings
  - cray_sysenv – adjust sysctl settings, limits, systemd settings, etc.
  - cray_time – settings for various aspects of time including ntp and timezone

Copyright 2018 Cray Inc.

Configuration Sets: Services Shared with Cray CLE

- **Config data needing additions for eLogin nodes:**
  - Bold are new or changed for SMW/eLogin
  - **cray_image_binding –** Bind mount PE or other projected images **to elogin_nodes**
  - cray_lustre_client – Lustre client settings
  - **cray_net** – site and Lnet network, **external-ipmi-net, external-management-net** settings
  - **cray_node_groups –** defines groups of nodes used in other config services including elogin_nodes and **external_tier1_nodes**
  - cray_simple_sync – a generic method of distributing files to targeted locations by hostname or node group membership
  - **cray_storage –** how to use boot RAID for CLE nodes **and eLogin local storage**

Copyright 2018 Cray Inc.

Configuration Sets: eLogin Specific Services

- **Config data only used by eLogin nodes:**
  - Bold are new or changed for SMW/eLogin
  - **cray_cfgset_exclude – profiles of CLE config set content to be excluded when pushed to eLogin**
  - cray_elogin_lnet – Luster Network (Lnet) settings for eLogin
  - cray_elogin_motd – Message Of The Day for eLogin
  - cray_elogin_networking – eLogin postfix configuration
  - cray_eproxy – settings for executing certain Cray XC and WLM commands
  - **cray_kdump – settings for kernel dump tool on eLogin**

Copyright 2018 Cray Inc.

`cfgset push` Enhanced for Node Groups

- **New `--node-group/-g` option to `cfgset push`**
  - Specify one or more node groups to which the config set should be pushed
  - Can be specified along with `--dest/-d` option to specify node groups and individual hosts
- **New `--ref-config-set/-r` option specifies config set to use when looking up node groups**
  - Useful when pushing config sets that do not have node groups definitions (e.g. the global config set)
- **If no `-r` option, the config set being pushed will be used to lookup node groups**

Copyright 2018 Cray Inc.

Configuration Service: cray_cfgset_exclude

- New configuration service in CLE config set
- Replaces `add_configset` script on the CMC
- Allows for users to specify config set content to exclude from config sets when they are pushed (rsync'd) to other nodes for:
  - Security
  - Performance
- Works in conjunction with `cfgset push`
- Node groups aware, exclude list supports rsync exclude wildcards
- Pre-populated with elogin exclude data used in pre-UP06

Copyright 2018 Cray Inc.

Configuration Service: cray_cfgset_exclude

```
cray_cfgset_exclude:
    settings:
        profiles:
            data:
            -   key: elogin_security
                exclude_content:
                - config/cray_sdb_config.yaml
                - config/cray_drc_config.yaml
                - config/cray_lmt_config.yaml
                - files/roles/common/etc/ssh
                - files/roles/common/root
                - files/roles/munge
                - files/roles/common/etc/opt/cray/xtremoted-agent
                - files/roles/merge_account_files
                - files/simple_sync/common/files/etc/ssh
                - files/simple_sync/common/files/root/.ssh
                - worksheets
                groups:
                - elogin_nodes
```

Profile name

Exclude list

Node group(s)

Copyright 2018 Cray Inc.

`cfgset push` with Exclude Profiles

- **`cfgset push` on a cle config set excludes files for each host as defined in the `cray_cfgset_exclude` service in the config set**
  - The config set specified via `--ref-config-set` is used to look up exclude profiles and node groups
  - If no `--ref-config-set` is specified, exclude profiles are looked up in the config set being pushed
  - `--no-exclude-errors` option allows for pushing of old config sets without exclude profiles defined
- **cfgset push on a global config set does not exclude any files**

Copyright 2018 Cray Inc.

Configuration Service Modifications: `cray_storage`

- eLogin nodes need to configure their storage
- Re-using the existing `cray_storage` service
  - Before CLE 6.0.UP06 only sets the active CLE storage set from the `cray_bootraid` configuration
- Introducing "storage profiles"
  - List of devices/partitions to configure on each device
  - Very similar to the existing cray_bootraid storage schema
  - Required eLogin partitions provided as pre-populated data

Copyright 2018 Cray Inc.

eLogin Node Storage Provisioning

- **Need to provision/prepare the storage on the internal eLogin disks when booting**
- **eLogin nodes require specifically-labeled partitions**
  - Non-persistent: BOOT, TMP, SWAP, WRITELAYER, GRUB
  - Persistent: CRASH, PERSISTENT
- **Ansible play will be run during dracut pre-mount phase**
  - Setup disks/partitions as per the storage profile in the assigned config set

Copyright 2018 Cray Inc.

Configuration Service Modifications: cray_storage

```
storage_profiles:
        data:
        -   key: elogin_default
            enabled: true
            layouts:
            -   key: /dev/sda              # Add devices here
                partition_type: gpt
                persist_on_boot: false     # Re-create the partitions on boot
                partitions:                # Add partitions here
                -   key: BOOT              # key is partition label
                    description: Partition storing the grub boot data.
                    type: ext4             # ext3, ext4, btrfs, xfs, swap
                    size: 2GiB             # MiB, GiB, TiB or 'ALL'
                    partition_flags: []
                -   key: SWAP
                [...]
```


Copyright 2018 Cray Inc.

Configuration Service Modifications: `cray_storage`

Device	Label	Type	Size	Mountpoint	Description
/dev/sda	GRUB	ext3	1MiB		Required partition for grub2 boot loader
/dev/sda	BOOT	ext4	2GiB	/boot	Partition storing the grub2 boot data
/dev/sda	WRITELAYER	ext3	20GiB	/tmp/writable	Partition for the writable overlay
/dev/sda	TMP	xfs	256GiB	/tmp	Underlying partition for the temporary file system
/dev/sda	SWAP	swap	128GiB		Standard swap partition
/dev/sdb	CRASH	ext4	10GiB	/var/crash	Location for kdumps
/dev/sdb	PERSISTENT	xfs	ALL	/var/opt/cray/persistent	Stores data that should persist between image deployments

Copyright 2018 Cray Inc.

Storage Provisioning/Play Walkthrough

- **state = storage_send**
  - Pull storage config YAML from SMW to node via TFTP
- **state = provisioning (Ansible play runs)**
  - Runs for both PXE and GRUB/disk boot types
  1. Import storage config and get assigned storage profile
  2. Verify devices in profile exist on the node
  3. Create partition type (gpt) on devices where `!persist_on_boot`
  4. Create partitions on devices where `!persist_on_boot` or `persist_on_boot`, but partition does not exist yet (if possible)
  5. Run mkfs on newly created partitions
  6. Verify that /dev/disk/by-label/{LABEL} exists for all partitions in config

Copyright 2018 Cray Inc.

Persistence behavior

- **`persist_on_boot` – determines if the disk will be completely wiped during boot**
- **For disks where `persist_on_boot: true`,**
  - The Ansible play will attempt to create partitions that have been added to the config (will fail unless the new partition “fits”)
  - Resizing/reordering/removing partitions not supported
  - Manual intervention required to make these kinds of changes

Copyright 2018 Cray Inc.

Configuration Service Modifications: `cray_storage`

- **Validation rules for storage profiles run by**

  smw# **cfgset validate p0**

  1. **type='swap' partitions have mount_point='swap'**
  2. **Partition labels are unique per storage profile**
  3. **size='ALL' partitions are specified last in the list, only one per device/layout**

Copyright 2018 Cray Inc.

Configuration Service Modifications: `cray_storage`

- eLogin nodes are assigned a storage profile at creation (or upon update)
- Profile does not need to exist at create/update time
- Validation occurs on enode boot/reboot
  - Config set existence
  - Storage profile existence in assigned config set

Copyright 2018 Cray Inc.

PE Image Binding on eLogin

- **New behavior:**
  - eLogin nodes find all matching images from `cray_image_binding` service in config set based on membership in `client_groups`
  - esd pushes matching images to node as squashfs during `image_binding_sync` state of boot
  - Image binding images reside in `/var/opt/cray/imps/image_roots` on the eLogin node
- **Benefits of new behavior**
  - Internal CLE nodes and eLogin nodes now share image binding implementation
  - Increased flexibility for image binding on eLogin nodes
    - Can use image binding with images other than just PE image

Copyright 2018 Cray Inc.

Configure PE Image Binding on eLogin

- **Ensure elogin nodes in elogin_nodes node group**
  - To view

```
smw# cfgset get cray_node_groups.settings.groups.data.elogin_nodes.members p0
elogin1
elogin2
```

  - To add elogin3 to elogin_nodes node group

```
smw# cfgset modify -a elogin3 \
cray_node_groups.settings.groups.data.elogin_nodes.members p0
```

  - To view again

```
smw# cfgset get cray_node_groups.settings.groups.data.elogin_nodes.members p0
elogin1
elogin2
elogin3
```


Copyright 2018 Cray Inc.

Configure PE Image Binding on eLogin

- **Ensure elogin_nodes specified for client_groups of PE image binding profile**
  - To view

```
smw# cfgset get cray_image_binding.settings.profiles.data.PE.client_groups
login_nodes
compute_nodes
```

  - To add elogin_nodes node group to the setting

```
smw# cfgset modify -a elogin_nodes \
cray_image_binding.settings.profiles.data.PE.client_groups p0
```

  - To view again

```
smw# cfgset get cray_image_binding.settings.profiles.data.PE.client_groups
login_nodes
compute_nodes
elogin_nodes
```


Copyright 2018 Cray Inc.

Configure PE Image Binding on eLogin

- **Ensure PE image binding profile is enabled**
  - To view

```
smw# cfgset get cray_image_binding.settings.profiles.data.PE.enabled p0
false
```

  - To change setting from false to true

```
smw# cfgset modify -s true \
cray_image_binding.settings.profiles.data.PE.enabled p0
```

  - To view again

```
smw# cfgset get cray_image_binding.settings.profiles.data.PE.enabled p0
true
```


Copyright 2018 Cray Inc.

Configure PE Image Binding on eLogin

- **Ensure PE image is exported and name matches PE image binding profile**
  - To find the name of the PE image

```
smw# ls -d /var/opt/cray/imps/boot_images/pe*
/var/opt/cray/imps/boot_images/pe_compute_cle_6.0up06_sles_12sp3
smw# ls /var/opt/cray/imps/boot_images/pe_compute_cle_6.0up06_sles_12sp3
.imps_Image_metadata  squashfs
```

  - To export it if not already exported

```
smw# image export --format squashfs $PE_IMAGE_NAME
```

  - To view the currently configured PE image name in image binding

```
smw# cfgset get cray_image_binding.settings.profiles.data.PE.image p0
pe_compute_cle_6.0up06_sles_12sp3
```

  - To update currently configured PE image name in image binding

```
smw# cfgset modify -s $PE_IMAGE_NAME \
cray_image_binding.settings.profiles.data.PE.image p0
```

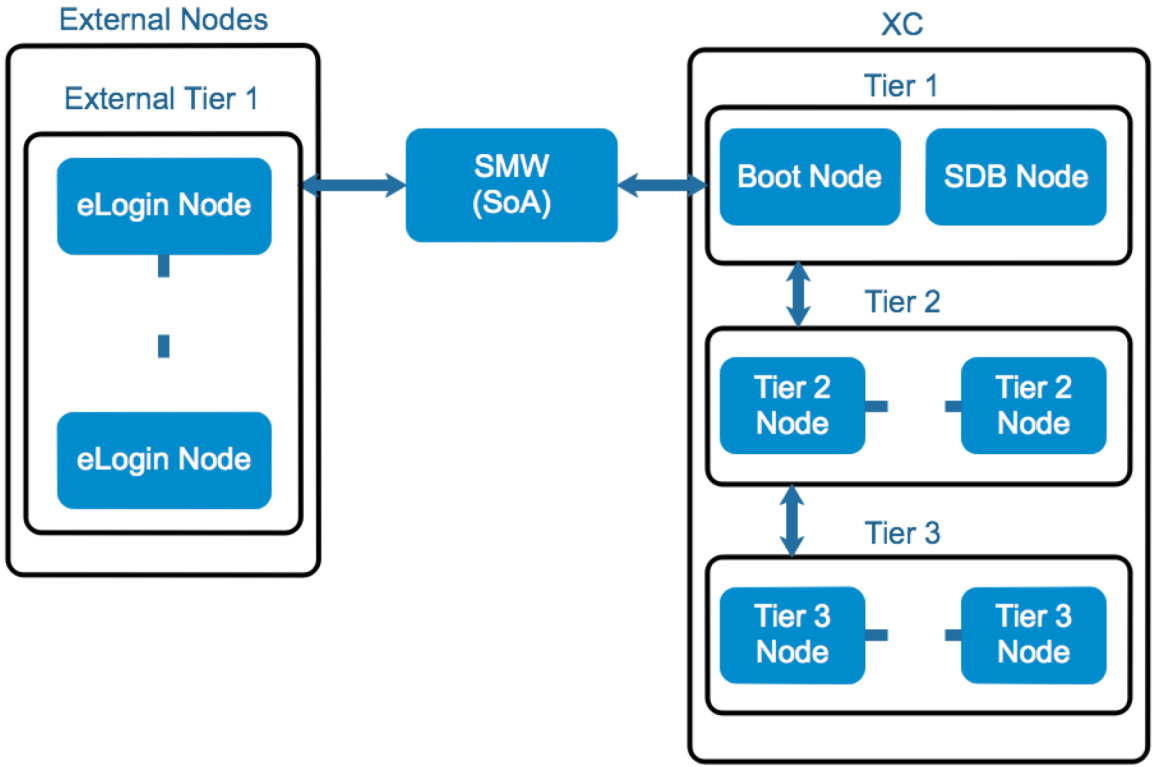

Copyright 2018 Cray Inc.

Scalable Services External Tier1 Nodes

- **Created a new tier of nodes called `external_tier1_nodes`**
- **External tier1 nodes**
  - must include all eLogin nodes
  - must have a direct connection to the SMW
- **External tier1 nodes communicate directly with the SMW for the following services**
  - Live Updates
  - Lightweight Log Manager (LLM)

Copyright 2018 Cray Inc.

Scalable Services External Tier1 Nodes



Copyright 2018 Cray Inc.

Scalable Services External Tier1 Nodes

- New `external_tier1_groups` setting will default to the list containing just the `elogin_nodes` node group.
- No additional configuration necessary if using `elogin_nodes` as node group name.

```
smw# cfgset get \
cray_scalable_services.settings.scalable_service.data.external_tier1_groups p0
elogin_nodes
```


Copyright 2018 Cray Inc.

Managing eLogin Nodes: Simple Sync

- **Method for installing files to eLogin nodes without using a site Ansible play**
  - Simple rsync of files from a root in the config set container to "/" on the eLogin node
  - Files can be installed by hostname or by node_group membership
- To install a file to a specific eLogin node, place it under the following directory on the SMW

  `/var/opt/cray/imps/config/sets/<config_set>/files/simple_sync/hostnames/<host_name>/files/<path_to_file_on_elogin>`
- To install a file to a group of eLogin nodes, place it under the following directory on the SMW
  - The <node_group> must be created in the config set prior to being able to use it

  `/var/opt/cray/imps/config/sets/<config_set>/files/simple_sync/nodegroups/<node_group>/files/<path_to_file_on_elogin>`

Copyright 2018 Cray Inc.

eLogin SSH Host Keys

- **SSH host keys**
  - Shared by all nodes or unique for each node
  - Generated automatically or supplied by the site
- **ssh_host_keys node registry field**
  - simple_sync (default)
    - esd looks for SSH host key files under Simple Sync in config set
      - hostname, then node group, then common
  - generate
    - Create SSH host keys when PXE booting node
      - Boot will fail if there are SSH host keys in Simple Sync
  - /some/directory/path
    - Copy site-provided SSH host keys when PXE booting node
      - Boot will fail if there are SSH host keys in Simple Sync

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

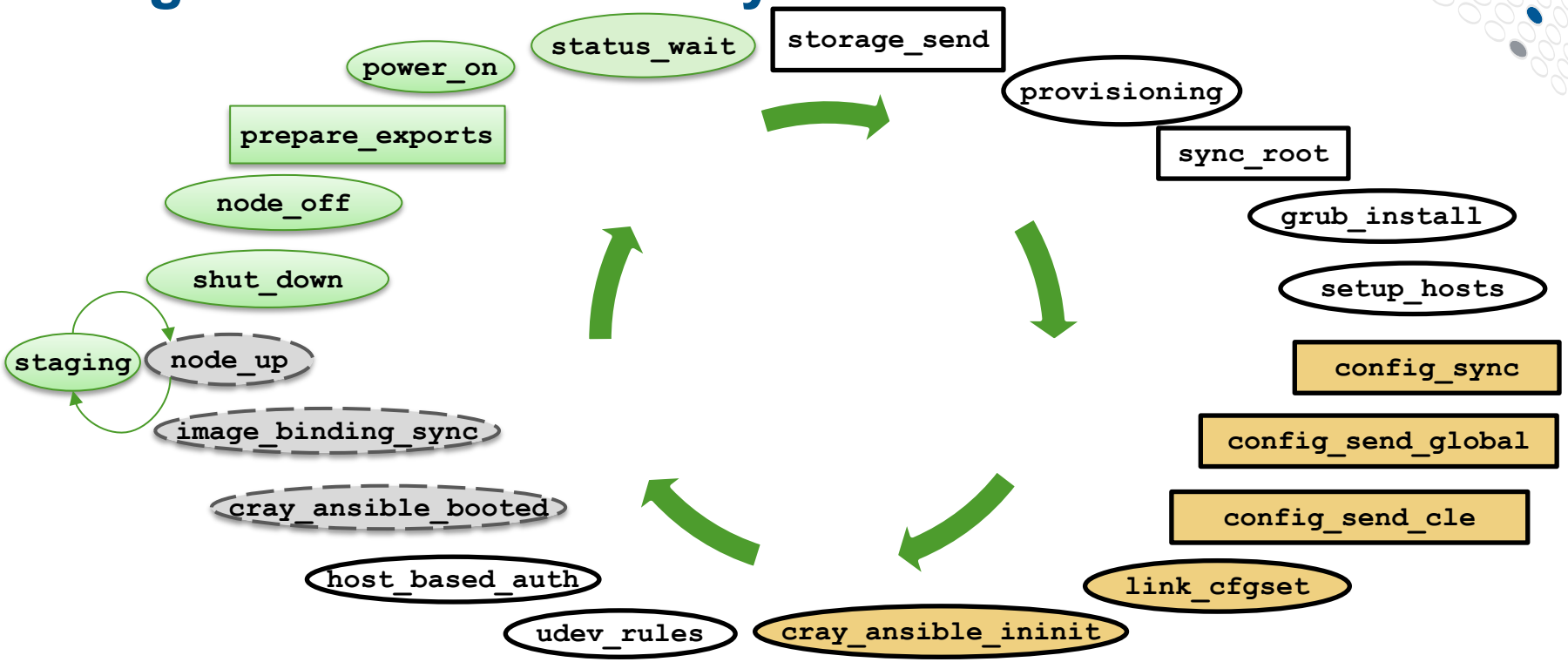
Copyright 2018 Cray Inc.

States and State Transitions

- esd (external state daemon) provides a service to manage Cray external nodes
- Tracking state of node
  - Allows sysadmin to know whether nodes are powered on/off, booting, or ready for users
  - Enables enforcement of security profile during PXE boot
    - Open access during proper state in boot
      - x.509 certificate (so SMW will trust identify of the external node)
      - Node's SSH host keys
      - SMW's public root SSH key (so node will trust root@smw for SSH)
      - OS image
    - Close access at proper state in boot

Copyright 2018 Cray Inc.

eLogin Node State Life Cycle



Copyright 2018 Cray Inc.

States and State Transitions by esd (SMW-side Actions)

State	Action
prepare_exports	Only for a PXE boot, the security profile opens up for transfer of information to the node
power_on	enode boot and enode reboot commands will trigger a transition to the power_on state. esd will issue the IPMI power control command to the node to turn on the power
status_wait	esd transitions to the status_wait state after the power_on state to see what the node does next. Some time elapses between turning on the power to the node and when the node finishes initialization (POST). This state indicates that ESD is waiting for communication from the node. During this state, certificates are retrieved from the SMW for a PXE boot. Only once secure communication can be established between the node and esd do we advance states. If a node is booting from disk and the network connection via the management network is missing or misconfigured, the node will continue to boot while esd will remain in status_wait state
shut_down	enode shutdown command will cause the node to begin the shutdown process, and esd moves the node into the shut_down state. The shut_down state begins with issuing a soft power off to the node, then waits for a timeout before issuing a hard power off. If the hard poweroff fails, the node will enter an error state. The "shut_down" state indicates that the node is in the process of shutting down
node_off	The node_off state is the result of a successful node shutdown

Copyright 2018 Cray Inc.

States and State Transitions by esd (SMW-side Actions)

State	Action
staging	New information is being staged (transferred to a booted node) for OS image, PE image, global config set, CLE config set, kernel parameters, and kernel and initrd for grub config. Once staging is complete the node can be rebooted.
UNKNOWN	The UNKNOWN state typically occurs when esd is started on the SMW. If the node is physically powered off, esd can check for that, but otherwise, esd does not know the state of the eLogin node
ERROR	The esd daemon will not intentionally put a node in the ERROR state, but any of the other states can transition to this state. If that happens, ``esd`` does some cleanup, such as closing any security access

Copyright 2018 Cray Inc.

Node States When esd Starts

- **If node is powered off**
  - node_off
- **If node is powered on (UP06)**
  - UNKNOWN
- **If node is powered on (UP07)**
  - esd will check via ssh to node
    - If SSH fails
      - UNKNOWN
    - If SSH succeeds
      - Set state to last state sent to esd stored in /var/run/cray/esd_state
        - This may set node to ERROR state

Copyright 2018 Cray Inc.

States and State Transitions by External Node

- **State updates from node**
  - Sent by calling /bin/cray/dracut_dispatch_state with state payload
- **Called by all dracut scripts needing to communicate state**

  /etc/opt/cray/dracut/elogin-pre-mount.d:
  - 10SetupDisk.sh
  - 20SyncRoot.sh
  - 25MountRoot.sh

  /etc/opt/cray/dracut/elogin-pre-pivot.d:
  - 10GrubInstall.py
  - 12Hosts.sh
  - 20ConfigSync.sh
  - 30CrayAnsible.sh
  - 32ConfigNetworkUdevRules.sh
  - 33HostbasedAuthClientSetup.sh
- **Called by cray-ansible in the booted phase for some states**
  - Beginning cray-ansible
  - Transferring PE image
  - Done with cray-ansible

Copyright 2018 Cray Inc.

States and State Transitions by External Node

State	PXE boot	Disk boot	Actions
storage_send	X		Node requests access to storage configuration from esd and then transfers the storage.yaml via TFTP
provisioning	X	X	Node applies storage configuration to format the storage for the node
sync_root	X		Node NFS mounts SMW /var/opt/cray/imps/boot_images/$IMAGE readonly and copies the squashfs formatted image root with the operating system to persistent storage
mount_root	X	X	Node prepares overlayfs with writable layer and mounts the squashfs image root readonly
grub_install	X	X	Node installs grub2 on BOOT device, which enables future disk boots
setup_hosts	X	X	Node sets up host file to ensure it has enough to continue to the next boot phase

Copyright 2018 Cray Inc.

States and State Transitions by External Node

State	PXE boot	Disk boot	Actions
config_sync	X		Node prepares overlayfs with config set directory and starts sshd so esd can push config sets
config_send_global	X		Node requests that esd send the global config set
config_send_cle	X		Node requests that esd send the CLE config set and then stops sshd
link_cfgset	X	X	Node links the persistent storage where the config sets are placed to the running system
cray_ansible_ininit	X	X	Node runs cray-ansible in the init phase to run Ansible plays
udev_rules	X	X	Node runs the udev rules script to properly order the network interfaces
hostbased_auth	X	X	Node prepares host-based authentication
cray_ansible_booted	X	X	Node begins running cray-ansible in the booted phase
image_binding_sync	X	X	Node requests transfer of the squashfs formatted PE image root (and other image_binding images) to persistent storage
node_up	X	X	Node finishes in cray-ansible in the booted phase, everything is ready for users to log in

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- **Security**
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

Security

- BMC username/password
- Console access
- Firewall
- TLS/x.509 certificates/ssh host keys

Copyright 2018 Cray Inc.

BMC Username & Password

smw# **enode create --bmc_username <user> --bmc_password <filename>**

- bmc_username defaults to root
- No default bmc_password,
  - will prompt interactively if no –bmc_password filename is provided

smw# **enode update --set-bmc_username <user>**

smw# **enode update --unset-bmc_username**

- Switches to default (root)

smw# **enode update --set-bmc_password <filename or empty string>**

- Empty string to prompt

- **Password stored in node registry, but not displayable**

smw# **enode list --field name,bmc_username**

smw# **enode list --filter bmc_username=root**

- **Can't use bmc_password in field or filter**

Copyright 2018 Cray Inc.

Console Access – conman

- **conman/conmand has duties similar to xtcon and console logging of HSS events**
  - Used for esMS/esLogin (first generation external login nodes)
  - Used for CIMS/CDL (second generation external login nodes)
  - Used for CMC/eLogin (third generation external login nodes)
  - Used for SMW/eLogin (fourth generation external login nodes)
- **SMW/eLogin**
  - conmand daemon
    - manages consoles defined by its configuration file (/etc/conman.conf)
    - listens for connections from clients
  - conman command
    - connects to remote consoles being managed by conmand

Copyright 2018 Cray Inc.

conman

- eLogin node documentation
  - Prepares BIOS/system setup to use serial-over-LAN (SOL) for remote console access to the node via IPMI protocol to the BMC or iDRAC device
    - IPMI - Intelligent Platform Management Interface
    - BMC – Baseboard Management Controller for out-of-band management
    - iDRAC – integrated Dell Remote Access Controller

Copyright 2018 Cray Inc.

conman

- Add new node to the node registry and configure `conmand`

  smw# **enode create elogin2**
- `conman` supports three modes of console access:
  - monitor (read-only)
  - interactive (read-write)
  - broadcast (write-only)
  - If neither the '-m' (monitor) nor '-b' (broadcast) options are specified, the console session is opened in interactive mode.

Copyright 2018 Cray Inc.

conman

- **What consoles (for nodes) are defined?**

smw# **conman -q**
elogin1
elogin2

- **Console logs**

/var/opt/cray/log/external/conman/console-elogin2

- **Interactive console terminal access with console-sharing**

smw# **conman -j elogin1**

Once a connection is established, **enter "&." to close the session**, or "&?" to see a list of currently available escape sequences

Copyright 2018 Cray Inc.

Firewall Configuration

- **Firewall implemented using SuSEfirewall2**
  - Configuration settings stored in /etc/sysconfig/SuSEfirewall2 to create iptables rules.
  - Sites should not make changes directly to that configuration file because it is changed whenever Cray Ansible plays are run, and site changes (e.g., custom zoning) could be overwritten.
- **SMW firewall configuration**
  - Enable cray_firewall in global config set
  - Firewall Zones
    - INT (for SMW eth1-eth5 interfaces)
    - EXT (for SMW eth0 to site-admin-net and SMW eth6 to external-ipmi-net)
    - MGMT (for SMW eth7 to external-management-net)
- **eLogin node firewall configuration**
  - Enable cray_firewall in CLE config set
  - Firewall Zones
    - EXT (for eLogin to site-user-net)
    - MGMT (for eLogin to external-management-net)

Copyright 2018 Cray Inc.

Firewall SMW Ports

- **MGMT zone**
  - (table to the right)
- **EXT zone**
  - Port 22 open (for SSH access)

Service	Port	Protocol
esd	8449	TCP
TFTP	69	UDP
DHCP	67	UDP
NFS	48451	UDP
NFS	49478	TCP
NFS	41276	UDP
NFS	35938	TCP
NFS	mountd (20048)	UDP
NFS	mountd (20048)	TCP
NFS	NFS (2049)	UDP
NFS	NFS (2049)	TCP
NFS	SUNRPC (111)	UDP
NFS	SUNRPC (111)	TCP
LiveUpdates	2526	TCP

Copyright 2018 Cray Inc.

Firewall eLogin Node Ports

- **EXT zone**
  - Port 22 open (for SSH access)
- **MGMT zone**

Service	Port	Protocol
SSH	22	TCP
NTP	123	UDP

Copyright 2018 Cray Inc.

TLS Background
CAPMC – x.509 Certificate Management

- **CAPMC implemented this already, we're using it**
- **xtmake_ca for certificates**

  /var/opt/cray/certificate_authority
  - CA certificate: certificate_authority.crt
  - SMW server certificate: host/host.crt, host/host.key
  - Default client certificate: client/client.crt, client/client.key
- **nginx configuration on SMW**
  - For xtremoted

    /etc/nginx/conf.d/xtremoted.conf
  - For esd

    /etc/nginx/conf.d/cray/esd.conf

Copyright 2018 Cray Inc.

TLS Configuration

- **esd configuration -- /etc/opt/cray/esd/esd.ini**

```
[esd]
allowed_clients=crayadm
```

  - A comma-separated list of CN values
  - An empty list means to allow all clients except for the elogin nodes, where the CN of an elogin node certificate starts with "elogin-"
- **enode configuration -- /etc/opt/cray/esd/esd.ini**

```
[enode]
cacert =
/var/opt/cray/certificate_authority/certificate_authority.crt
cert = /var/opt/cray/certificate_authority/client/client.crt
key = /var/opt/cray/certificate_authority/client/client.key
```

  - Override with OS_CACERT, OS_CERT, OS_KEY (same as CAPMC client)
- **esd server (nginx) enforces https**
- **enode uses https to the esd server (nginx)**

Copyright 2018 Cray Inc.

TLS Node Certificate

- **On enode create, certificate is created**

```
smw# enode create \
 --bmc_ip $BMC_IP --mgmt_ip $MGMT_IP \
 --mgmt_mac $MGMT_MAC \
 --image $IMAGE \
 --configset p0 --storage_profile elogin_default \
 $NODE
smw# ls -l /var/opt/cray/esd/nodes
smw# ls -l /var/opt/cray/esd/nodes/$NODE
```

- **On boot, certificate files are made available to client through nginx**

```
smw# enode boot --pxe $NODE
smw# cat /var/opt/cray/esd/nodes/$NODE.conf
```

  Limited to the node's IP address

- **Once boot gets far enough, access is revoked**
- **State transition notifications require the client certificate**
  - esd rejects state transition if client certificate CN incorrect or IP address

Copyright 2018 Cray Inc.

TLS esd Client Access Enforcement

- **State transition notifications require the client certificate**
  - esd rejects state transition if
    - client cert invalid (not signed by CA)
    - client cert CN incorrect
    - source IP address doesn't match node mgmt IP address

Copyright 2018 Cray Inc.

TLS eLogin Node Boot

- **CA certificate is included in the image**
  - /var/lib/esd/certificate_authority.crt
- **SSH host key files**
  - Generated by esd when the node is created and added to the system known_hosts file
  - Deleted by esd when the node is deleted from node registry
- **Early in Cray dracut scripts (01GetKeys.sh) the node client x.509 certificate files are downloaded using https**
  - /var/opt/cray/esd/certs/client.crt , client.key
- **Uses the CA cert and its node certificate when sending notifications**
  - Validates the esd server's host certificate using CA cert

Copyright 2018 Cray Inc.

SSH Host Keys Enable Config Set Transfer

- Early in Cray dracut scripts (01GetKeys.sh) the node's SSH host keys are downloaded using https
- sshd daemon started in dracut (20ConfigSync.sh) to allow transfer of config sets
- Config sets transferred to node using call to the API for "cfgset push"
  - CLE config set is sanitized during transfer using cray_cfgset_exclude
- Four states in the boot sequence (20ConfigSync.sh) related to config sets
  - config_sync
  - config_send_global
  - config_send_cle
  - link_config
- In Cray dracut script the node sends a state transition notification to esd, causing server action to push config set from SMW to node
- sshd daemon stopped in dracut after receiving config sets

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- **enode**
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

enode

- enode list
- enode clear_error (introduced in UP07)
- enode create
- enode delete
- enode enroll
- enode status
- enode shutdown
- enode update
- enode boot
- enode reboot
- enode stage

Copyright 2018 Cray Inc.

enode Command Syntax

```
smw# enode -h
usage: enode [-h] [-V] [-v] [-q] [--node_type {elogin}]
             {list,enroll,create,update,delete,boot,reboot,shutdown,status,stage}
             ...

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         print version header
  -v, --verbose         increase the verbosity of the command
  -q, --quiet           suppresses unnecessary output
  --node_type {elogin}  specify node type

subcommands:
  {list,enroll,create,update,delete,boot,reboot,shutdown,status,stage}
```


Copyright 2018 Cray Inc.

enode list Command Output

```
smw# enode list
NAME     CONFIGSET STORAGE_PROFILE ESD_GROUP
IMAGE             BMC_IP    MGMT_IP  MGMT_MAC           PARAMETERS  STATE
elogin1 -         elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:33 -
          UNKNOWN
elogin2 -         elogin_default  elogin    elogin-smw-large 10.6.0.6 10.7.0.6 00:11:22:33:44:22 -
          UNKNOWN
elogin3 -         elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:55 -
          UNKNOWN
elogin4 -         elogin_default  elogin    elogin-smw-large 10.6.0.5 10.7.0.5 00:11:22:33:44:88 -
          UNKNOWN
elogin5 p0        elogin_default  elogin    elogin-smw       10.6.0.7 10.7.0.7 90:B1:1C:3A:0A:1B -
          UNKNOWN
smw# enode list --format csv
NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,MGMT_IP,MGMT_MAC,PARAMETERS,STATE
elogin1,,elogin_default,elogin,elogin-smw-large,10.6.0.4,10.7.0.4,00:11:22:33:44:33,,UNKNOWN
elogin2,,elogin_default,elogin,elogin-smw-large,10.6.0.6,10.7.0.6,00:11:22:33:44:22,,UNKNOWN
elogin3,,elogin_default,elogin,elogin-smw-large,10.6.0.4,10.7.0.4,00:11:22:33:44:55,,UNKNOWN
elogin4,,elogin_default,elogin,elogin-smw-large,10.6.0.5,10.7.0.5,00:11:22:33:44:88,,UNKNOWN
elogin5,p0,elogin_default,elogin,elogin-smw,10.6.0.7,10.7.0.7,90:B1:1C:3A:0A:1B,,UNKNOWN
```


Copyright 2018 Cray Inc.

enode list Command Output

```
smw# enode list --format csv --output /tmp/node-list
smw# cat /tmp/node-list
NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,MGMT_IP,MGMT_MAC,PARAMETERS,STATE
elogin1,,elogin_default,elogin,elogin-smw-
large,10.6.0.4,10.7.0.4,00:11:22:33:44:33,,UNKNOWN
elogin2,,elogin_default,elogin,elogin-smw-
large,10.6.0.6,10.7.0.6,00:11:22:33:44:22,,UNKNOWN
elogin3,,elogin_default,elogin,elogin-smw-
large,10.6.0.4,10.7.0.4,00:11:22:33:44:55,,UNKNOWN
elogin4,,elogin_default,elogin,elogin-smw-
large,10.6.0.5,10.7.0.5,00:11:22:33:44:88,,UNKNOWN
elogin5,p0,elogin_default,elogin,elogin-smw,10.6.0.7,10.7.0.7,90:B1:1C:3A:0A:1B,,UNKNOWN
```


Copyright 2018 Cray Inc.

enode list Command Output (Other Fields)

```
smw# enode list –fields \
name,rootdev,bootif,pci,remcon,bmc_username,kdump_enable,kdump_high,kdump_low,ssh_hos
t_keys
NAME    ROOTDEV  BOOTIF PCI       REMCON           BMC_USERNAME  KDUMP_ENABLE
KDUMP_HIGH  KDUMP_LOW  SSH_HOST_KEYS
elogin1 /dev/sda eth2  pci=bfsort console=ttyS1,115200n8 root    False         -
-   simple_sync
elogin2 /dev/sda eth2  pci=bfsort console=ttyS1,115200n8 root    False         -
-   simple_sync
elogin3 /dev/sda eth0  pci=bfsort console=ttyS1,115200n8 root    False         -
-   simple_sync
elogin4 /dev/sda eth0  pci=bfsort console=ttyS1,115200n8 root    False         -
-   simple_sync
elogin5 /dev/sda eth2  pci=bfsort console=ttyS1,115200n8 root    False         -
-   simple_sync
smw# enode list --fields all
NAME            CONFIGSET                      STORAGE_PROFILE  ESD_GROUP  IMAGE
BMC_IP    MGMT_IP   MGMT_MAC           PARAMETERS  STATE     ROOTDEV   BOOTIF  PCI
REMCON          BMC_USERNAME  KDUMP_ENABLE  KDUMP_HIGH  KDUMP_LOW  SSH_HOST_KEYS
```


Copyright 2018 Cray Inc.

enode create Command Output

```
smw# enode create --group elogin-test --bmc_ip 10.6.0.8 --mgmt_ip 10.7.0.8 --mgmt_mac 12:34:56:78:90:AB elogin6
Please enter the BMC password for the root user on node elogin6:
Validate password for the BMC password for the root user on node elogin6:
Creating the following node:
elogin6
Successfully created ['elogin6'].
smw# enode list
NAME    CONFIGSET STORAGE_PROFILE ESD_GROUP IMAGE            BMC_IP    MGMT_IP  MGMT_MAC           PARAMETERS
STATE
elogin1 -         elogin_default  elogin    elogin-smw-large 10.6.0.4  10.7.0.4 00:11:22:33:44:33  -
          UNKNOWN
elogin2 -         elogin_default  elogin    elogin-smw-large 10.6.0.6  10.7.0.6 00:11:22:33:44:22  -
          UNKNOWN
elogin3 -         elogin_default  elogin    elogin-smw-large 10.6.0.4  10.7.0.4 00:11:22:33:44:55  -
          UNKNOWN
elogin4 -         elogin_default  elogin    elogin-smw-large 10.6.0.5  10.7.0.5 00:11:22:33:44:88  -
          UNKNOWN
elogin5 p0        elogin_default  elogin    elogin-smw       10.6.0.7  10.7.0.7 90:B1:1C:3A:0A:1B  -
          UNKNOWN
elogin6 -         -               elogin-test -             10.6.0.8  10.7.0.8 90:B1:1C:3A:0A:AB  -
          UNKNOWN
```


Copyright 2018 Cray Inc.

enode delete Command Output

```
smw# enode delete elogin6
Deleting the following nodes:
elogin6
Successfully deleted ['elogin6']

smw# enode list
NAME    CONFIGSET STORAGE_PROFILE ESD_GROUP
IMAGE             BMC_IP    MGMT_IP  MGMT_MAC          PARAMETERS  STATE
elogin1 -         elogin_default  elogin    elogin-smw-large
10.6.0.4 10.7.0.4 00:11:22:33:44:33 -          UNKNOWN
elogin2 -         elogin_default  elogin    elogin-smw-large 10.6.0.6 10.7.0.6
00:11:22:33:44:22 -           UNKNOWN
elogin3 -         elogin_default  elogin    elogin-smw-
large 10.6.0.4 10.7.0.4 00:11:22:33:44:55 -           UNKNOWN
elogin4 -         elogin_default  elogin    elogin-smw-large 10.6.0.5 10.7.0.5
00:11:22:33:44:88 -           UNKNOWN
elogin5 p0        elogin_default  elogin    elogin-smw       10.6.0.7
10.7.0.7 90:B1:1C:3A:0A:1B -          UNKNOWN
```


Copyright 2018 Cray Inc.

enode enroll Command (UP06)

- **Enroll nodes from an inventory.csv file**
  - This option is used for migration from CMC/eLogin and CIMS/esLogin to SMW/eLogin management
  - Nodes must be subsequently updated with the enode update command

```
smw# cat /root/inventory.csv
NODE_NAME, BMC_IP, MAC_ADDR, N_CPUs, ARCH, RAM_MB, DISK_GB, NODE_DESC
flubber-elogin1,10.148.0.8,90:b1:1c:26:93:62,32,x86_64,131072,550,flubber-
elogin1
flubber-skl,10.148.0.9,18:66:DA:94:4D:32,32,x86_64,131072,550,flubber-skl
smw# enode enroll /root/inventory.csv
Creating the following nodes:
flubber-elogin1
flubber-skl
Successfully created ['flubber-elogin1', 'flubber-skl'].
```


Copyright 2018 Cray Inc.

enode enroll Command (UP06)

```
smw# enode list
NAME    CONFIGSET STORAGE_PROFILE ESD_GROUP IMAGE             BMC_IP    MGMT_IP   MGMT_MAC
     PARAMETERS  STATE
elogin1 -          elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:33
     -           UNKNOWN
elogin2 -          elogin_default  elogin    elogin-smw-large 10.6.0.6 10.7.0.6 00:11:22:33:44:22
     -           UNKNOWN
elogin3 -          elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:55
     -           UNKNOWN
elogin4 -          elogin_default  elogin    elogin-smw-large 10.6.0.5 10.7.0.5 00:11:22:33:44:88
     -           UNKNOWN
elogin5 p0         elogin_default  elogin    elogin-smw       10.6.0.7 10.7.0.7 90:B1:1C:3A:0A:1B
     -           UNKNOWN
flubber-elogin1 - -                 None      -                10.148.0.8  0.0.0.0
90:b1:1c:26:93:62  -        UNKNOWN
flubber-skl     - -                 None      -                10.148.0.9  0.0.0.0
18:66:DA:94:4D:32  -        UNKNOWN
```

Note: BMC_IP is not on 10.6.0.0 network so this will have to be changed with “enode update”

Copyright 2018 Cray Inc.

enode enroll Command (UP07)

- **Enroll nodes from two CSV formats**
  - Migration from CMC/eLogin and CIMS/esLogin in inventory.csv format

    ```
    NODE_NAME, BMC_IP, MAC_ADDR, N_CPUs, ARCH, RAM_MB,
    DISK_GB, NODE_DESC
    ```

  - Output from enode list in csv format

    ```
    NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,
    MGMT_IP,MGMT_MAC,PARAMETERS,STATE,ROOTDEV,BOOTIF,PCI,R
    EMCON,BMC_USERNAME,KDUMP_ENABLE,KDUMP_HIGH,KDUMP_LOW,S
    SH_HOST_KEYS
    ```


Copyright 2018 Cray Inc.

enode enroll Command Output (UP07)

- Save current node registry to a csv file

```
smw# enode list --format csv --fields all --output file.csv
smw# cat file.csv
NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,MGMT_IP,MGMT_MAC,PARAMETERS,STATE,ROOTDEV,BOOTIF,PCI,REMCON,BMC_USERNAME,KDUMP_ENABLE,KDUMP_HIGH,KDUMP_LOW,SSH_HOST_KEYS
flubber-elogin1,p0,elogin_default,elogin,elogin-smw-large_cle_6.0.UP07-build6.0.7084_sles_12sp3-created20180514,10.6.0.1,10.7.0.1,80:18:44:EB:BA:74,,node_up,/dev/sda,eth0,pci=bfsort,"ttyS1,115200n8",root,False,,,simple_sync
```

- Modify csv file somehow, then import it

```
smw# vi file.csv
smw# enode enroll file.csv
```


Copyright 2018 Cray Inc.

enode status Command Output

```
smw# enode status elogin1 elogin2 elogin5
NODE      PING   POWER                    STATE
elogin1   Up     Chassis Power is on      node_up
elogin2   Down   Chassis Power is off     UNKNOWN
elogin5   Down   Chassis Power is on      UNKNOWN
```

- Notice that elogin5 has a PING status of DOWN and is powered on
  - This is because the node is booted to the BIOS setup and is thus not pingable

Copyright 2018 Cray Inc.

enode Error State Reporting

- **Additional error state reporting with the enode status**
  - Added with -l or --long option
- **Sample output:**

```
smw# enode status elogin1 -l
NODE    PING  POWER                STATE
elogin1 Up    Chassis Power is on  Error
                                   State prior to error: node_up
                                   ERROR 1: Failed to transition. Self state:
node_up desired state: First_error; Error: exception: Failed to transition
to state: First_error
                                   ERROR 2: Failed to transition. Self state:
Error desired state: Second_error; Error: exception: Failed to transition
to state: Second_error
```


Copyright 2018 Cray Inc.

enode Error State Reporting

- Error message is identical to message in the esd log file
  - /var/opt/cray/log/external/esd.log

2017-12-14 17:06:18,473 - state[22845] - transition - ERROR: Node(elogin1) exception: Failed to transition to state: First_error

2017-12-14 17:06:18,474 - node[22845] - transition_state - ERROR: Node(elogin1) Failed to transition. Self state: node_up desired state: First_error; Error: exception: Failed to transition to state: First_error

2017-12-14 17:06:18,474 - enode_elogin_node[22845] - set_error - ERROR: Node elogin1 encountered an error:

2017-12-14 17:06:34,514 - state[22845] - transition - ERROR: Node(elogin1) exception: Failed to transition to state: Second_error

2017-12-14 17:06:34,514 - node[22845] - transition_state - ERROR: Node(elogin1) Failed to transition. Self state: Error desired state: Second_error; Error: exception: Failed to transition to state: Second_error

2017-12-14 17:06:34,515 - enode_elogin_node[22845] - set_error - ERROR: Node elogin1 encountered an error:

- Anything placing node in error state gets tracked as a new error

Copyright 2018 Cray Inc.

enode boot --bios Command Output

```
smw# enode boot --bios elogin1
Booting the following nodes: mode: bios
elogin1
['elogin1']: All node(s) started boot process.

smw# enode status elogin1
NODE      PING  POWER                    STATE
elogin1   DOWN  Chassis Power is on      status_wait
```


Copyright 2018 Cray Inc.

enode shutdown

- **Attempts a graceful shutdown**
  - Will perform a hard power off if the graceful shutdown fails after 3 minutes
- **Successful shutdown results in the node being powered off and in a *node_off* state**
- **May be used on any node that is powered on**
  - If the node is in *shut_down* state, it will exit stating it is in *shut_down* state
  - Nodes in unknown states which are powered on can be recovered to a known state (powered off in *node_off* state)
- **If a node remains in *shut_down* state, something has gone wrong**
  - */var/opt/cray/log/external/esd.log* should be consulted for details
- **Node state should be checked with the "enode status" command**

Copyright 2018 Cray Inc.

enode update

- **Updates configuration settings for existing nodes**
  - Multiple nodes may be updated by a single command with exception of the following fields which are unique to each node:
    - bmc_ip
    - mgmt_ip
    - mgmt_mac
  - Limitations
    - Nodes cannot be selected by wildcarding node names
    - Nodes cannot be selected by matching field values (filters)

Copyright 2018 Cray Inc.

enode update

- **Create three nodes**

```
smw# enode create --bmc_ip 10.6.0.1 --mgmt_ip 10.7.0.1 --mgmt_mac 00:11:22:33:44:55 elogin1
Creating the following node:
elogin1
Successfully created ['elogin1'].
smw# enode create --bmc_ip 10.6.0.2 --mgmt_ip 10.7.0.2 --mgmt_mac 00:11:22:33:44:56 elogin2
Creating the following node:
elogin2
Successfully created ['elogin2'].
smw# enode create --bmc_ip 10.6.0.3 --mgmt_ip 10.7.0.3 --mgmt_mac 00:11:22:33:44:57 elogin3
Creating the following node:
elogin3
Successfully created ['elogin3'].
smw# enode list
NAME     CONFIGSET  STORAGE_PROFILE  ESD_GROUP  IMAGE  BMC_IP    MGMT_IP   MGMT_MAC           PARAMETERS  STATE
elogin1  -          -                -          -      10.6.0.1  10.7.0.1  00:11:22:33:44:55  -           UNKNOWN
elogin2  -          -                -          -      10.6.0.2  10.7.0.2  00:11:22:33:44:56  -           UNKNOWN
elogin3  -          -                -          -      10.6.0.3  10.7.0.3  00:11:22:33:44:57  -           UNKNOWN
```


Copyright 2018 Cray Inc.

enode update

- **Add configset p0 to all 3 nodes**

```
smw# enode update --set-configset p0 elogin1 elogin2 elogin3
Updating the following node(s):
elogin3
elogin2
elogin1
Successfully updated ['elogin3', 'elogin2', 'elogin1']

smw# enode list --fields name,configset
NAME     CONFIGSET
elogin1  p0
elogin2  p0
elogin3  p0
```


Copyright 2018 Cray Inc.

enode update

- Change elogin2 to configset p1

```
smw# enode update --set-configset p1 elogin2
Updating the following node(s):
elogin2
Successfully updated ['elogin2']

smw# enode list --fields name,configset
NAME     CONFIGSET
elogin1  p0
elogin2  p1
elogin3  p0
```


Copyright 2018 Cray Inc.

enode update

- **Remove configset from elogin3**

```
smw# enode update --unset-configset elogin3
Updating the following node(s):
elogin3
Successfully updated ['elogin3']

smw# enode list --fields name,configset
NAME     CONFIGSET
elogin1  p0
elogin2  p1
elogin3  -
```


Copyright 2018 Cray Inc.

enode update

- **Set image and esd_group for all nodes**

```
smw# enode update --set-image elogin-image \
--set-group elogin-group elogin1 elogin2 elogin3
Updating the following node(s):
elogin3
elogin2
elogin1
Successfully updated ['elogin3', 'elogin2', 'elogin1']

smw# enode list --fields name,image,esd_group
NAME     IMAGE          ESD_GROUP
elogin1  elogin-image   elogin-group
elogin2  elogin-image   elogin-group
elogin3  elogin-image   elogin-group
```


Copyright 2018 Cray Inc.

enode update

- **Add elogin-test esd_group to elogin1**

```
smw# enode update --add-group elogin-test elogin1
Updating the following node(s):
elogin1
Successfully updated ['elogin1']

smw# enode list --fields name,image,esd_group
NAME     IMAGE          ESD_GROUP
elogin1  elogin-image   elogin-group,elogin-test
elogin2  elogin-image   elogin-group
elogin3  elogin-image   elogin-group
```


Copyright 2018 Cray Inc.

enode update

- **Add elogin-tds esd_group to elogin2 and elogin3**

```
smw# enode update --add-group elogin-tds elogin2 elogin3
Updating the following node(s):
elogin3
elogin2
Successfully updated ['elogin3', 'elogin2']

smw# enode list --fields name,image,esd_group
NAME     IMAGE          ESD_GROUP
elogin1  elogin-image   elogin-group,elogin-test
elogin2  elogin-image   elogin-group,elogin-tds
elogin3  elogin-image   elogin-group,elogin-tds
```


Copyright 2018 Cray Inc.

enode update

- **Remove elogin-group esd_group from elogin2**

```
smw# enode update --remove-group elogin-group elogin2
Updating the following node(s):
elogin2
Successfully updated ['elogin2']

smw# enode list --fields name,image,esd_group
NAME     IMAGE          ESD_GROUP
elogin1  elogin-image   elogin-group,elogin-test
elogin2  elogin-image   elogin-tds
elogin3  elogin-image   elogin-group,elogin-tds
```


Copyright 2018 Cray Inc.

enode update

- **Check remcon setting on all nodes**

```
smw# enode list --fields name,remcon
NAME     REMCON
elogin1  ttyS1,115200n8
elogin2  ttyS1,115200n8
elogin3  ttyS1,115200n8
```

- **Change remcon on elogin1**

```
smw# enode update --set-remcon ttyS1,57600n8 elogin1
Updating the following node(s):
elogin1
Successfully updated ['elogin1']

smw# enode list --fields name,remcon
NAME     REMCON
elogin1  ttyS1,57600n8
elogin2  ttyS1,115200n8
elogin3  ttyS1,115200n8
```


Copyright 2018 Cray Inc.

enode update

- Set remcon back to default on elogin1

```
smw# enode update --unset-remcon elogin1
Updating the following node(s):
elogin1
Successfully updated ['elogin1']

smw# enode list --fields name,remcon
NAME     REMCON
elogin1  ttyS1,115200n8
elogin2  ttyS1,115200n8
elogin3  ttyS1,115200n8
```


Copyright 2018 Cray Inc.

enode update

- **Set mgmt_ip to 10.7.0.20 on elogin3**

```
smw# enode update --set-mgmt_ip 10.7.0.20 elogin3
Updating the following node(s):
elogin3
Successfully updated ['elogin3']

smw# enode list --fields name,mgmt_ip
NAME     MGMT_IP
elogin1  10.7.0.1
elogin2  10.7.0.2
elogin3  10.7.0.20
```

- **Set mgmt_ip to 10.7.0.21 on elogin1 and elogin2**

```
smw# enode update --set-mgmt_ip 10.7.0.21 elogin1 elogin2
ERROR: only one node may be targeted with the bmc_ip, mgmt_mac, or
mgmt_ip options.
```


Copyright 2018 Cray Inc.

PXE Boot

- **Boot a node using PXE boot to prepare storage, transfer x.509 certificate, SSH host keys, global and CLE config sets, OS image, and PE image.**

```
smw# enode boot --pxe elogin1
Booting the following nodes: mode: pxe
elogin1
['elogin1']: All node(s) started boot process.
```

- **Check status during the boot**

```
smw# enode status elogin1
NODE     PING  POWER                  STATE
elogin1  DOWN  Chassis Power is on    status_wait
```

- **Watch state transitions for the node during the boot**

```
smw# tail -f /var/opt/cray/log/external/esd.log | grep Transition
```


Copyright 2018 Cray Inc.

Disk Boot with States

- **Boot a node from disk using existing x.509 certificate, SSH host keys, global and CLE config sets, OS image, and PE image on the disk.**

```
smw# enode boot --disk elogin1
Booting the following nodes: mode: disk
elogin1
['elogin1']: All node(s) started boot process.
```

- **Watch the status during the boot**

```
smw# enode status elogin1
```

- **Watch state transitions during the boot**

```
smw# tail -f /var/opt/cray/log/external/esd.log | grep Transition
```


Copyright 2018 Cray Inc.

enode reboot

- The reboot command will reboot *booted* nodes or *off* nodes
- First, it will attempt to shut down the nodes
  - This will be a graceful power off.
  - If the graceful power off fails, then after three minutes of waiting, it will do a hard power-off of the node
- Once the nodes are off, then they will be booted.

smw# **enode reboot --pxe elogin1**

smw# **enode reboot --disk elogin2**

smw# **enode reboot --bios elogin3**

Copyright 2018 Cray Inc.

Staged Boot

- **Ability to prep a node for new deployment while the node is up and in use**
- **Multiple nodes can be staged at the same time, with the ability to reboot all of them at some point in the near future, either together or as a phased deployment**
  - Stage now, reboot later

smw# **enode stage $NODE**
smw# **enode reboot $NODE**

  - Stage now, reboot immediately after staging is complete

smw# **enode reboot --staged $NODE**

Copyright 2018 Cray Inc.

Staging State

- Node state 'staging'
  - Can only get to ‘staging’ from 'node_up'
  - When staging is finished, node is returned to 'node_up'
  - Failure to stage results in the node being in ERROR state

Copyright 2018 Cray Inc.

Clearing ERROR State

- **How to get a node out of ERROR State**
  - In UP06, save node registry data and recreate node

    smw# **enode list -fields all $NODE**

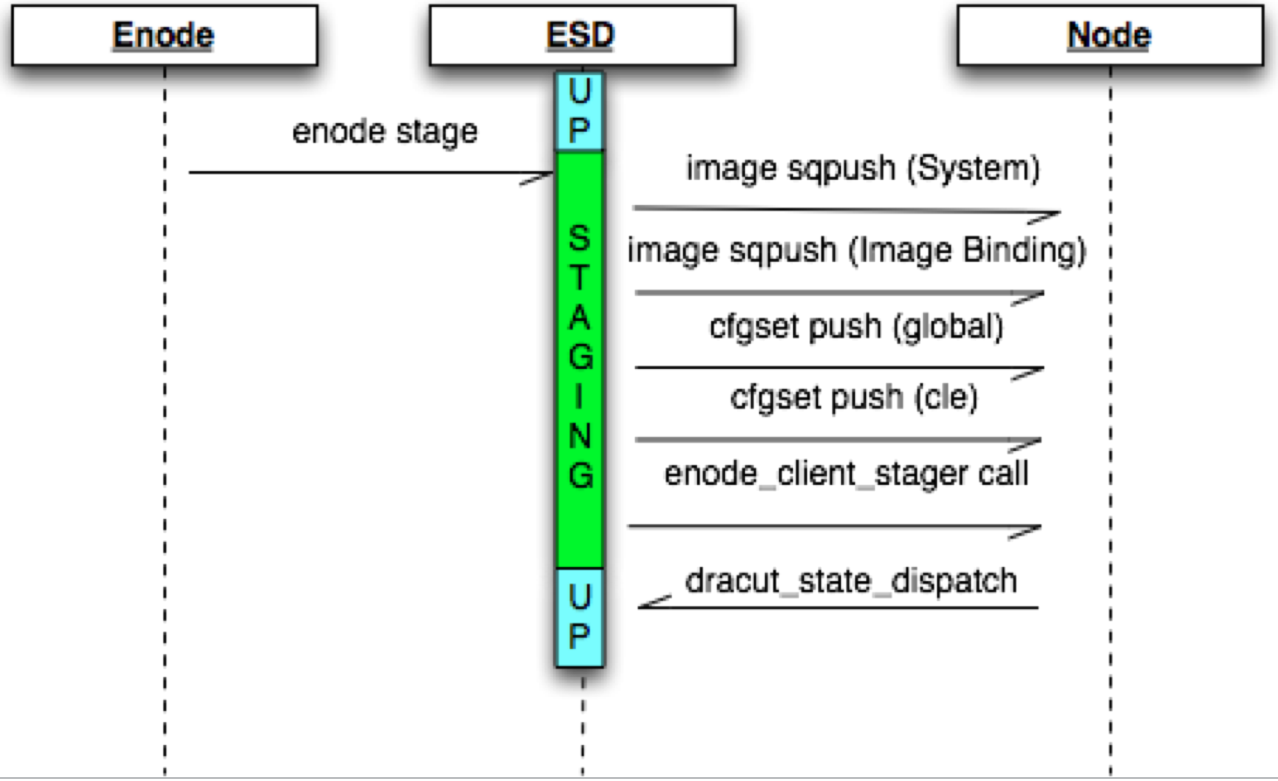
    smw# **enode delete $NODE**

    smw# **enode create (providing data from all fields above) $NODE**
  - In UP07, clear the error state

    smw# **enode clear_error $NODE**
    - The node will end up in
      - The node_up state, if it is powered on
      - The node_off state if powered off.
    - WARNING: Depending upon the error, the node may not be in a usable state until successfully rebooted!

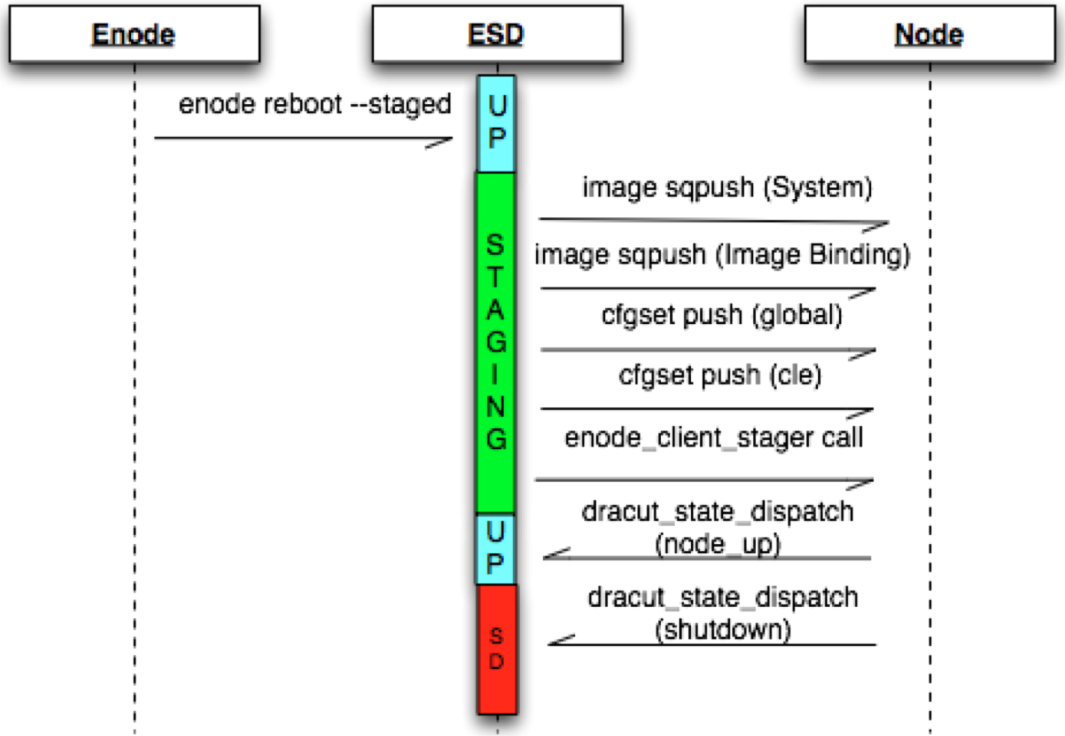
Copyright 2018 Cray Inc.

enode stage



Copyright 2018 Cray Inc.

enode reboot --staged



Copyright 2018 Cray Inc.

Staging Caveats

- Images must be exported as squashfs before issuing commands
- Whenever a staging operation occurs, the next disk boot will be whatever was staged last
- Staged initrd + kernels in the boot partition will aggregate and need to be cleaned out, if many staging operations occur
  - We cannot assume what to keep or remove automatically
  - Default size of boot volume is 2 gigabytes; will hold roughly 40 staging operations
- cfgset operations are not currently revertible
- Every staging operation creates a new GRUB menu entry; staging the same images/cfgsets/PE is faster after initial transfer (no-op/rsync).
- Mixed deployments (non-homogeneous) will complete at different times
- Multiple nodes will reboot when ready when using" enode reboot –staged" and are not synchronized.
  - To synchronize, use "enode stage" to prepare the nodes, and then "enode boot --disk" when ready to reboot them.

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- **DEBUG shell**
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

DEBUG Shell, Kernel Parameter and Breakpoints

- **During a boot, if there is a problem in the dracut scripts, then the node may enter the DEBUG shell on the console for troubleshooting**

  DEBUG#
- **During a boot, if the kernel parameter for DEBUG has been set, predefined breakpoints in the the dracut scripts will cause the node to enter the DEBUG shell on the console for troubleshooting**
  - The breakpoints will be at the beginning and end of each dracut script

    DEBUG SHELL: DEBUG=true kernel parameter set! Entering DEBUG shell
    DEBUG SHELL: 30CrayAnsible.sh ; starting cray-ansible
    DEBUG SHELL: 30CrayAnsible.sh ; exit will resume
    DEBUG(30CrayAnsible.sh) elogin3#
  - Enable the DEBUG flag kernel parameter, boot node, interact with console

    smw# **enode update --set-parameter DEBUG=true elogin3**
    smw# **enode boot –pxe elogin3**
    smw# **conman -j elogin3**
  - Disable the DEBUG flag kernel parameter

    smw# **enode update --remove-parameter DEBUG elogin3**

Copyright 2018 Cray Inc.

DEBUG Shell, Kernel Parameter and Breakpoints

- **Dracut scripts inside image root**
  - /etc/opt/cray/dracut
    - elogin-pre-mount.d/
      - 01GetKeys.sh
      - 10SetupDisk.sh
      - 20SyncRoot.sh
      - 25MountRoot.sh
      - 40CopyInitFiles.sh
    - elogin-pre-pivot.d/
      - 10GrubInstall.py
      - 12Hosts.sh
      - 15fstab_entries.sh
      - 20ConfigSync.sh
      - 30CrayAnsible.sh
      - 32ConfigNetworkUdevRules.sh
      - 33HostbasedAuthClientSetup.sh

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

Logging and Dumping

- Log locations for troubleshooting
- kdump
- xtdumpsys plugin for logs/dumps

Copyright 2018 Cray Inc.

Log Locations on SMW

- **DHCP** /var/opt/cray/log/smwmessages-YYYYMMDD
- **TFTP** /var/log/atftpd/atftp.log
- **conmand** /var/opt/cray/log/external/conman.log
- **node console log** /var/opt/cray/log/external/conman/console.$NODE
- **enode** /var/opt/cray/log/external/enode.log
- **esd** /var/opt/cray/log/external/esd.log
- **esd-uwsgi** /var/opt/cray/log/external/esd-uwsgi.log
- **nginx** /var/log/nginx
- **Ansible** /var/opt/cray/log/ansible

Copyright 2018 Cray Inc.

Log Locations on eLogin Nodes

- **Cray dracut logs** /root/.boot.log
- **State message log** /var/log/dracut_stat.log
- **syslog** /var/log/messages
- **Ansible** /var/opt/cray/log/ansible
- **dmesg output**
- **journalctl output**

Copyright 2018 Cray Inc.

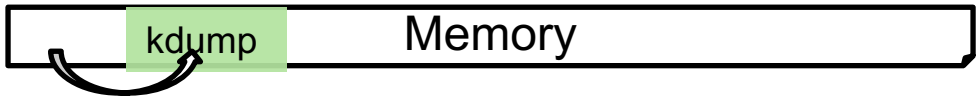
kdump - What is kdump?

- Production kernel; get access to all of memory

- Boot production kernel with *crashkernel* kernel command line parameters defined; get access to memory except for crashkernel reserved memory
- Production kernel does not access the kdump memory

- Kdump memory is not initialized at this point
- *systemctl start kdump* loads the crash kernel and crash initrd in the reserved kdump memory

- When a crash occurs *kexec* transfers control to the kernel in the kdump area



Copyright 2018 Cray Inc.

kdump Configuration

- Configuration
  - Configure memory to be reserved for the kdump capture kernel and the kdump initrd
  - Update the eLogin node definition to set kernel command line parameters that define low and high memory requirements
  - Test the kdump configuration by triggering a kdump
  - Dumps will be saved on /var/crash

Copyright 2018 Cray Inc.

kdump Configuration

- Reserve memory for the kdump kernel and initrd
  - Reserved memory is off limits to the production kernel
  - Memory will have to be reserved in both High Memory and in the 32 bit addressable Low Memory
  - Calculate the amount of High Memory required – memory located above 4GB
  - Calculate the amount of Low Memory required – memory below 4GB – memory in the DMA32Zone
  - Update the eLogin node definition to
    - Enable kdump
    - Set the amount of High Memory to be reserved
    - Set the amount of Low Memory to be reserved

Copyright 2018 Cray Inc.

kdump Configuration

- **Configure memory to be reserved for kdump**
  - **On the eLogin node execute the *kdumptool calibrate* command**

```
elogin# kdumptool calibrate
Total: 65490
Low: 72
High: 116
MinLow: 72
MaxLow: 3281
MinHigh: 0
MaxHigh: 62208
elogin#
```

  - **All values are in MB**
  - **Low 72**
  - **High 116**

Copyright 2018 Cray Inc.

kdump Configuration

- Determine amount of High Memory to reserve
  - Example: Determine the amount of memory to reserve above the 4GB memory limit on a system with 3TB of memory and 6 LUNs

    SIZE_HIGH = (RECOMMENDATION * RAM_IN_TB) + (LUNs / 2)

    SIZE_HIGH = (116 * 3) + (6 / 2)

    SIZE_HIGH = 351MB

  - LUNs - The maximum number of LUN kernel paths that you expect to ever create on the computer. Exclude multipath devices from this number, as these are ignored.

Copyright 2018 Cray Inc.

kdump Configuration

- Determine amount of Low Memory to reserve
  - Example: Determine the amount of memory to reserve below the 4GB memory limit on a system with 3TB of memory and 6 LUNs

    SIZE_LOW = (RECOMMENDATION * RAM_IN_TB) + CUSTOM_DRIVER-RESERVATION_ADJUSTMENT

    SIZE_LOW = (72 * 3) + (Custom_Drive_Adjustment)

    SIZE_LOW = 256MB

  - If the drivers for your device make many reservations in the DMA32 zone, the Low value also needs to be adjusted. However, there is no simple formula to calculate these. Finding the right size can therefore be a process of trial and error. To start, use the Low value recommended by *kdumptool calibrate*.

Copyright 2018 Cray Inc.

kdump Configuration

- **Determine amount of Low Memory to reserve**
  - **Using the calculated values for SIZE_HIGH and SIZE_LOW, pass those values to the eLogin node definition using the *enode update* command**

```
smw# enode update --set-kdump_high=512M --set-kdump_low=256M --set-kdump_enable elogin2
smw# enode list --fields name,state,kdump_enable,kdump_low,kdump_high elogin2
NAME       STATE      KDUMP_ENABLE  KDUMP_LOW  KDUMP_HIGH
elogin2     node_up   True                256M          512M
```

  - **When booting the eLogin node, the kernel command line parameter *crashkernel* will now include**

    crashkernel=512M,high crashkernel=256M,low

Copyright 2018 Cray Inc.

kdump Configuration

- **Determine amount of Low Memory to reserve**
  - **”Finding the right size of low memory to reserve can be a process of trial and error.”**

  2017-11-29 11:03:38 [8.893214] ---[end Kernel panic - not syncing: Can not allocate SWIOTLB buffer earlier and can't now provide you with the DMA bounce buffer
  - **Always trigger a test kdump in order to verify that you have reserved the correct amount of memory**

Copyright 2018 Cray Inc.

kdump Configuration

- **Trigger a kdump on a node manually**
  - **Enable the kdump service**
    elogin# **systemctl enable kdump**
  - **Start the kdump service**
    elogin# **systemctl start kdump**
  - **Initiate a test kdump crash**
    elogin# **echo c > /proc/sysrq-trigger**
- **Trigger kdump on a node via kernel parameter manually**
  - **Enable kdump in node registry**
    smw# **enode update --set-kdump_high=512M **
    --set-kdump_low=256M --set-kdump_enable elogin2
  - **Stage and reboot new kernel parameters**
    smw# **enode reboot --staged elogin2**
  - **Initiate a test kdump crash**
    elogin# **echo c > /proc/sysrq-trigger**

Copyright 2018 Cray Inc.

kdump Configuration

- **Reference**

  https://www.suse.com/documentation/sles-12/book_sle_tuning/data/sec_tuning_kexec_crashkernel.html

Copyright 2018 Cray Inc.

xtdumpsys Plugin for Logs/Dumps

- Dump and log collection for support
- A collection of xtdumpsys plugins and a scenario file (edumpsys.conf)
- New cray-edumpsys rpm
- Collects file globs and executes commands defined by elogin-data-capture.ini
  - SMW
  - Targeted eLogin nodes
- Collects kdumps and provides an option to trigger kdumps
- User interface is pre-defined via xtdumpsys

Copyright 2018 Cray Inc.

xtdumpsys Usage

- **Run xtdumpsys with edumpsys scenario file (edumpsys.conf)**
- **Must specify eLogin targets to dump with --add**
  - Example:

```
crayadm@smw> xtdumpsys -r 'reason' --config-file \
/etc/opt/cray/edumpsys/config/edumpsys.conf --add \
elogin2 elogin4
```


Copyright 2018 Cray Inc.

Usage – Triggering kdump

- **Triggering will attempt to kdump each targeted node (via --add)**
- **Add --conf trigger_kdump=1**
  - Example:

```
crayadm@smw> xtdumpsys -r 'reason' --conf-file\
/etc/opt/cray/edumpsys/config/edumpsys.conf --add \
elogin2 elogin4 --conf trigger_kdump=1
```


Copyright 2018 Cray Inc.

Configuration: Data Capture File for SMW and eLogin Nodes

smw# **vi /etc/opt/cray/edumpsys/conf/elogin-data-capture.ini**

```
[External Node]
files =
    /.imps_Image_metadata
    /root/.boot.log
    /var/log/dracut_stat.log
    /var/opt/cray/log/ansible/*
    /var/log/messages*
commands =
    cat /proc/cmdline
    cat /proc/cpuinfo
    cat /proc/meminfo
    cat /proc/filesystems
    dmidecode
    systemctl status kdump
    #iptables commands are disabled by default for security reasons
    #iptables -L input_MGMT
    #iptables -L forward_MGMT
    dmesg
    journalctl --no-pager
```


Copyright 2018 Cray Inc.

Trigger kdump Use Case

- 3 runs of xtdumpsys
- Run 1 – normal usage tells you that nodes do not have any kdumps
- Run 2 – add the trigger command and run again to start dumps
- Run 3 – after nodes have rebooted, run once more time to collect all the kdumps

Copyright 2018 Cray Inc.

Run 1 Output – No kdumps Found

```
crayadm@smw> xtdumpsys -r 'reason' --conf-file /etc/opt/cray/edumpsys/config/edumpsys.conf --add elogin4
...
INFO: eLogin Kdump: starting thread (timeout: 1800s)
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=5 -tt -x
-l root elogin4 'echo "###start";find /var/crash -mindepth 1 -maxdepth 1 -type d ;echo "###end"'’
INFO: eLogin Kdump: RC=0
WARNING: eLogin Kdump: No kdumps found on elogin4.
WARNING: eLogin Kdump:
WARNING: eLogin Kdump: The following nodes do not have kdumps found in /var/crash that fit within the log window:
WARNING: eLogin Kdump:    elogin4 (kdump not enabled)
WARNING: eLogin Kdump:
WARNING: eLogin Kdump: You can trigger kdumps on enabled nodes by adding the following command line argument to xtdumpsys:
WARNING: eLogin Kdump:    --config trigger_kdump=1
WARNING: eLogin Kdump:
WARNING: eLogin Kdump: NOTE: kdump must be enabled on a node before triggering. Refer to the XC Series SMW-managed eLogin
Installation Guide (S-3020) for information on how to enable kdumps before triggering.
WARNING: eLogin Kdump:
WARNING: eLogin Kdump: NOTE: All nodes targeted via '--add' will then trigger a kdump (if enabled) and then reboot. Once
the nodes are booted, you can then re-run xtdumpsys without '--config trigger_kdump=1' to collect the kdumps.
WARNING: eLogin Kdump:
INFO: eLogin Kdump: thread finished
INFO: eLogin Kdump: Finished in 852 ms
INFO: ################################################################################
INFO: # Your dump is available in /var/opt/cray/dump/p0-20171221t080455-1712211611 #
INFO: ################################################################################
```


Copyright 2018 Cray Inc.

Run 2 Output – Trigger kdump

```
crayadm@smw> xtdumpsys -r "development" --config-file /etc/opt/cray/edumpsys/config/edumpsys.conf \
--add elogin4  --config trigger_kdump=1
...
INFO: eLogin Kdump: starting thread (timeout: 1800s)
INFO: eLogin Kdump: kdump_trigger option detected
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o
ConnectTimeout=5 -tt -x -l root elogin4 'echo "###start";find /var/crash -mindepth 1 -maxdepth 1 -type d ;echo
"###end"’’
INFO: eLogin Kdump: RC=0
WARNING: eLogin Kdump: No kdumps found on elogin4.
INFO: eLogin Kdump: Triggering kdump on elogin4...
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o
ConnectTimeout=5 -tt -x -l root elogin4 'echo c > /proc/sysrq-trigger''INFO: eLogin Kdump:
INFO: eLogin Kdump: kdumps have been triggered on the following nodes: elogin4
INFO: eLogin Kdump:
INFO: eLogin Kdump: The nodes will now dump and then reboot. Once the nodes are rebooted, you can then re-run
xtdumpsys without '--config trigger_kdump=1' to collect the kdumps. Use the enode command to determine when the
nodes have rebooted.
INFO: eLogin Kdump: thread finished
INFO: eLogin Kdump: Finished in 11729 ms
INFO: ################################################################################
INFO: # Your dump is available in /var/opt/cray/dump/p0-20171221t080455-1712211850 #
INFO: ################################################################################
```


Copyright 2018 Cray Inc.

Run 3 Output – Collect kdump

```
crayadm@smw> xtdumpsys -r "development" --config-file /etc/opt/cray/edumpsys/config/edumpsys.conf --add elogin4
...
INFO: eLogin Kdump: starting thread (timeout: 1800s)
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=5 -tt -x
-l root elogin4 'echo "###start";find /var/crash -mindepth 1 -maxdepth 1 -type d ;echo "###end"'’
INFO: eLogin Kdump: RC=0
INFO: eLogin Kdump: Attempting to retrieve the following kdumps from elogin4: /var/crash/2017-12-21-18:18
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=5 -tt -x
-l root elogin4 'tar -zcvf /var/crash/2017-12-21-18:18.tar.gz /var/crash/2017-12-21-18:18’’
INFO: eLogin Kdump: RC=0
INFO: eLogin Kdump: Running 'scp -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=5 -r
'root@elogin4:/var/crash/2017-12-21-18:18.tar.gz' /var/opt/cray/dump/p0-20171221t080455-
1712211853/edumpsys/elogin4/kdumps’
INFO: eLogin Kdump: RC=0
INFO: eLogin Kdump: Running 'ssh -v -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=5 -tt -x
-l root elogin4 'rm -r /var/crash/2017-12-21-18:18.tar.gz’’
INFO: eLogin Kdump: RC=0
INFO: eLogin Kdump: thread finished
INFO: eLogin Kdump: Finished in 74822 ms
INFO: ################################################################################
INFO: # Your dump is available in /var/opt/cray/dump/p0-20171221t080455-1712211853 #
INFO: ################################################################################
```


Copyright 2018 Cray Inc.

Dump Location

- Dumps go to `/var/opt/cray/dump/p0-<session-id>-<dump-time>/`
  - This is predetermined by xtdumpsys and cannot be changed
- xtdumpsys will give you this directory in the output
- edumpsys data is contained in the `edumpsys/` directory inside the main dump directory
- Each node gets its own directory within edumpsys

Copyright 2018 Cray Inc.

edumpsys Output Structure

```
crayadm@smw> tree -ah /var/opt/cray/dump/p0-
20171221t080455-1712211853/edumpsys/
/var/opt/cray/dump/p0-20171221t080455-
1712211853/edumpsys/
├── [ 56] elogin2
│   ├── [582K] elogin_cmds.out
│   ├── [4.0K] files
│   │   ├── [228K] ansible-booted
│   │   ├── [263K] ansible-booted.1
│   │   ├── [244K] ansible-init
│   │   ├── [ 0] ansible-init.1
│   │   ├── [ 28K] .boot.log
│   │   ├── [1.7K] dracut_stat.log
│   │   ├── [ 19K] file-changelog-booted
│   │   ├── [ 0] file-changelog-booted.1
│   │   ├── [ 34K] file-changelog-booted.yaml
│   │   ├── [ 0] file-changelog-booted.yaml.1
│   │   ├── [ 19K] file-changelog-init
│   │   ├── [ 36K] file-changelog-init.yaml
│   │   ├── [1.2K] .imps_Image_metadata
│   │   └── [454K] messages
│   └── [ 37] kdumps
│       └── [1.7G] 2017-12-21-18:18.tar.gz
├── [7.9K] smw_cmds.out
└── [4.0K] smw_files
    ├── [5.4M] access.log
    ├── [ 12K] atftp.log
    ├── [4.0K] conman
    │   ├── [ 25M] console.elogin1
    │   ├── [ 17M] console.elogin2
    │   ├── [6.6M] console.elogin3
    │   ├── [9.3M] console.elogin4
    │   └── [9.0M] console.elogin5
    ├── [4.3K] conman.log
    ├── [ 94M] enode.log
    ├── [191K] error.log
    ├── [ 77M] esd.log
    ├── [8.6M] esd-uwsgi.log
    └── [3.9M] smwmessages-20171221

4 directories, 29 files
```


Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- **SMW HA**

Copyright 2018 Cray Inc.

SMW HA Configuration

- **Prepare eth6/eth7 on both SMWs**
- **Configuring SMW HA to manage SMW/eLogin**
  - Check cluster status
  - Run SMWHAconfig to add cluster resources
  - Check cluster status
- **Configuring SMW HA to not manage SMW/eLogin**
  - Check cluster status
  - Run SMWHAconfig to remove cluster resources
  - Check cluster status

Copyright 2018 Cray Inc.

Prepare eth6/eth7 for SMW HA

smw# **yast2 lan**

- **Standalone SMW IP address**
  - eth6 10.6.1.1 on external-ipmi-net
  - eth7 10.7.1.1 on external-management-net
- **SMW HA**
  - SMW1
    - eth6 10.6.1.2 on external-ipmi-net
    - eth7 10.7.1.2 on external-management-net
  - SMW2
    - eth6 10.6.1.3 on external-ipmi-net
    - eth7 10.7.1.3 on external-management-net

Copyright 2018 Cray Inc.

Configuring SMW HA to Manage SMW/eLogin

- Check status of cluster

smw1# **crm_mon -r1**
Stack: unknown
Current DC: smw1 (version unknown) - partition with quorum
Last updated: Thu Nov 30 08:31:59 2017
Last change: Tue Nov 28 16:26:10 2017 by hacluster via crmd on smw1

2 nodes configured
33 resources configured

Online: [smw1 smw2]

Full list of resources:
ClusterIP (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP1 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP2 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP3 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP4 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP5 (ocf::heartbeat:IPaddr2): Started smw1
ClusterMonitor (ocf::smw:ClusterMonitor): Started smw1
ClusterTimeSync (ocf::smw:ClusterTimeSync): Started smw1
HSSDaemonMonitor (ocf::smw:HSSDaemonMonitor): Started smw1
Notification (ocf::heartbeat:MailTo): Started smw1
ResourceInit (ocf::smw:ResourceInit): Started smw1
cray-cfgset-cache (systemd:cray-cfgset-cache): Started smw1
dhcpd (systemd:dhcpd.service): Started smw1

Copyright 2018 Cray Inc.

Configuring SMW HA to Manage SMW/eLogin

```
fsync  (ocf::smw:fsync):       Started smw1
hss-daemons    (lsb:rsms):     Started smw1
stonith-1      (stonith:external/ipmi):        Started smw1
stonith-2      (stonith:external/ipmi):        Started smw2
Resource Group: HSSGroup
    mysqld     (ocf::heartbeat:mysql): Started smw1
Resource Group: IMPSGroup
    cray-ids-service   (systemd:cray-ids-service):     Started smw1
    cray-ansible       (systemd:cray-ansible): Started smw1
    IMPSFilesystemConfig       (ocf::smw:FileSystemConfig):    Started
smw1
Resource Group: LogGroup
    cray-syslog        (systemd:llmrd.service):        Started smw1
    LogFilesystemConfig        (ocf::smw:FileSystemConfig):    Started
smw1
Resource Group: SharedFilesystemGroup
    homedir    (ocf::heartbeat:Filesystem):    Started smw1
    md-fs      (ocf::heartbeat:Filesystem):    Started smw1
    imps-fs    (ocf::heartbeat:Filesystem):    Started smw1
    ml-fs      (ocf::heartbeat:Filesystem):    Started smw1
    repos-fs   (ocf::heartbeat:Filesystem):    Started smw1
Resource Group: SystemGroup
    NFSServer  (systemd:nfsserver):    Started smw1
    EnableRsyslog      (ocf::smw:EnableRsyslog):       Started smw1
    syslog.socket      (systemd:syslog.socket):        Started smw1
Clone Set: clo_PostgreSQL [PostgreSQL]
    Started: [smw1 smw2 ]
```

- If all of the cluster resources are not running as expected, refer to the SMW HA Admin Guide for troubleshooting

Copyright 2018 Cray Inc.

Configuring SMW HA to Manage SMW/eLogin

- Log into the hostname of the Active SMW

  node# **ssh smw**

- Configure the cluster resources for eLogin
  - SMWHAconfig puts the cluster into maintenance mode

  smw# **cd /opt/cray/ha-smw/default/hainst**

  smw# **./SMWHAconfig --update --add_elogin**

- Exit maintenance mode and wait for cluster to stabilize

  smw# **maintenance_mode_configure disable**

  smw# **sleep 240**

Copyright 2018 Cray Inc.

Configuring SMW HA to Manage SMW/eLogin

- Verify cluster has started all resources

smw1# **crm_mon -r1**

Stack: unknown

Current DC: smw1 (version unknown) - partition with quorum

Last updated: Thu Nov 30 08:31:59 2017

Last change: Tue Nov 28 16:26:10 2017 by hacluster via crmd on smw1

2 nodes configured

36 resources configured

Online: [smw1 smw2]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started smw1

ClusterIP1 (ocf::heartbeat:IPaddr2): Started smw1

ClusterIP2 (ocf::heartbeat:IPaddr2): Started smw1

ClusterIP3 (ocf::heartbeat:IPaddr2): Started smw1

ClusterIP4 (ocf::heartbeat:IPaddr2): Started smw1

ClusterIP5 (ocf::heartbeat:IPaddr2): Started smw1

ClusterMonitor (ocf::smw:ClusterMonitor): Started smw1

ClusterTimeSync (ocf::smw:ClusterTimeSync): Started smw1

HSSDaemonMonitor (ocf::smw:HSSDaemonMonitor): Started smw1

Notification (ocf::heartbeat:MailTo): Started smw1

ResourceInit (ocf::smw:ResourceInit): Started smw1

Copyright 2018 Cray Inc.

Configuring SMW HA to Manage SMW/eLogin

fsync (ocf::smw:fsync): Started smw1
hss-daemons (lsb:rsms): Started smw1
stonith-1 (stonith:external/ipmi): Started smw1
stonith-2 (stonith:external/ipmi): Started smw2
Resource Group: HSSGroup
mysqld (ocf::heartbeat:mysql): Started smw1
Resource Group: IMPSGroup
cray-ids-service (systemd:cray-ids-service): Started smw1
cray-ansible (systemd:cray-ansible): Started smw1
IMPSFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: LogGroup
cray-syslog (systemd:llmrd.service): Started smw1
LogFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: SharedFilesystemGroup
homedir (ocf::heartbeat:Filesystem): Started smw1
md-fs (ocf::heartbeat:Filesystem): Started smw1
imps-fs (ocf::heartbeat:Filesystem): Started smw1
ml-fs (ocf::heartbeat:Filesystem): Started smw1
repos-fs (ocf::heartbeat:Filesystem): Started smw1
Resource Group: SystemGroup
NFSServer (systemd:nfsserver): Started smw1
EnableRsyslog (ocf::smw:EnableRsyslog): Started smw1
syslog.socket (systemd:syslog.socket): Started smw1
Clone Set: clo_PostgreSQL [PostgreSQL]
Started: [smw1 smw2]
ClusterIP6 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP7 (ocf::heartbeat:IPaddr2): Started smw1
Resource Group: eloginGroup
esd (systemd:esd.service): Started smw1

- If all of the cluster resources are not running as expected, refer to the SMW HA Admin Guide for troubleshooting

Copyright 2018 Cray Inc.

Configuring SMW HA to Not Manage SMW/eLogin

- ## Check status of cluster

smw1# **crm_mon -r1**
Stack: unknown
Current DC: smw1 (version unknown) - partition with quorum
Last updated: Thu Nov 30 08:31:59 2017
Last change: Tue Nov 28 16:26:10 2017 by hacluster via crmd on smw1

2 nodes configured
36 resources configured

Online: [smw1 smw2]

Full list of resources:
ClusterIP (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP1 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP2 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP3 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP4 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP5 (ocf::heartbeat:IPaddr2): Started smw1
ClusterMonitor (ocf::smw:ClusterMonitor): Started smw1
ClusterTimeSync (ocf::smw:ClusterTimeSync): Started smw1
HSSDaemonMonitor (ocf::smw:HSSDaemonMonitor): Started smw1
Notification (ocf::heartbeat:MailTo): Started smw1
ResourceInit (ocf::smw:ResourceInit): Started smw1
cray-cfgset-cache (systemd:cray-cfgset-cache): Started smw1
dhcpd (systemd:dhcpd.service): Started smw1

Copyright 2018 Cray Inc.

Configuring SMW HA to Not Manage SMW/eLogin

fsync (ocf::smw:fsync): Started smw1
hss-daemons (lsb:rsms): Started smw1
stonith-1 (stonith:external/ipmi): Started smw1
stonith-2 (stonith:external/ipmi): Started smw2
Resource Group: HSSGroup
mysqld (ocf::heartbeat:mysql): Started smw1
Resource Group: IMPSGroup
cray-ids-service (systemd:cray-ids-service): Started smw1
cray-ansible (systemd:cray-ansible): Started smw1
IMPSFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: LogGroup
cray-syslog (systemd:llmrd.service): Started smw1
LogFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: SharedFilesystemGroup
homedir (ocf::heartbeat:Filesystem): Started smw1
md-fs (ocf::heartbeat:Filesystem): Started smw1
imps-fs (ocf::heartbeat:Filesystem): Started smw1
ml-fs (ocf::heartbeat:Filesystem): Started smw1
repos-fs (ocf::heartbeat:Filesystem): Started smw1
Resource Group: SystemGroup
NFSServer (systemd:nfsserver): Started smw1
EnableRsyslog (ocf::smw:EnableRsyslog): Started smw1
syslog.socket (systemd:syslog.socket): Started smw1
Clone Set: clo_PostgreSQL [PostgreSQL]
Started: [smw1 smw2]
ClusterIP6 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP7 (ocf::heartbeat:IPaddr2): Started smw1
Resource Group: eloginGroup
esd (systemd:esd.service): Started smw1

- If all of the cluster resources are not running as expected, refer to the SMW HA Admin Guide for troubleshooting

Copyright 2018 Cray Inc.

Configuring SMW HA to Not Manage SMW/eLogin

- Log into the hostname of the Active SMW

  node# **ssh smw**

- Configure the cluster resources for eLogin
  - SMWHAconfig puts the cluster into maintenance mode

  smw# **cd /opt/cray/ha-smw/default/hainst**

  smw# **./SMWHAconfig --update --remove_elogin**

- Exit maintenance mode and wait for cluster to stabilize

  smw# **maintenance_mode_configure disable**

  smw# **sleep 240**

Copyright 2018 Cray Inc.

Configuring SMW HA to Not Manage SMW/eLogin

- Verify cluster has started all resources

smw1# **crm_mon -r1**
Stack: unknown
Current DC: smw1 (version unknown) - partition with quorum
Last updated: Thu Nov 30 08:31:59 2017
Last change: Tue Nov 28 16:26:10 2017 by hacluster via crmd on smw1

2 nodes configured
33 resources configured

Online: [smw1 smw2]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP1 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP2 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP3 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP4 (ocf::heartbeat:IPaddr2): Started smw1
ClusterIP5 (ocf::heartbeat:IPaddr2): Started smw1
ClusterMonitor (ocf::smw:ClusterMonitor): Started smw1
ClusterTimeSync (ocf::smw:ClusterTimeSync): Started smw1
HSSDaemonMonitor (ocf::smw:HSSDaemonMonitor): Started smw1
Notification (ocf::heartbeat:MailTo): Started smw1
ResourceInit (ocf::smw:ResourceInit): Started smw1

Copyright 2018 Cray Inc.

Configuring SMW HA to Not Manage SMW/eLogin

cray-cfgset-cache (systemd:cray-cfgset-cache): Started smw1
dhcpd (systemd:dhcpd.service): Started smw1
fsync (ocf::smw:fsync): Started smw1
hss-daemons (lsb:rsms): Started smw1
stonith-1 (stonith:external/ipmi): Started smw1
stonith-2 (stonith:external/ipmi): Started smw2
Resource Group: HSSGroup
mysqld (ocf::heartbeat:mysql): Started smw1
Resource Group: IMPSGroup
cray-ids-service (systemd:cray-ids-service): Started smw1
cray-ansible (systemd:cray-ansible): Started smw1
IMPSFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: LogGroup
cray-syslog (systemd:llmrd.service): Started smw1
LogFilesystemConfig (ocf::smw:FileSystemConfig): Started smw1
Resource Group: SharedFilesystemGroup
homedir (ocf::heartbeat:Filesystem): Started smw1
md-fs (ocf::heartbeat:Filesystem): Started smw1
imps-fs (ocf::heartbeat:Filesystem): Started smw1
ml-fs (ocf::heartbeat:Filesystem): Started smw1
repos-fs (ocf::heartbeat:Filesystem): Started smw1
Resource Group: SystemGroup
NFSServer (systemd:nfsserver): Started smw1
EnableRsyslog (ocf::smw:EnableRsyslog): Started smw1
syslog.socket (systemd:syslog.socket): Started smw1
Clone Set: clo_PostgreSQL [PostgreSQL]
Started: [smw1 smw2]

- If all of the cluster resources are not running as expected, refer to the SMW HA Admin Guide for troubleshooting

Copyright 2018 Cray Inc.

Managing eLogin Nodes from SMW

- Overview of SMW/eLogin
- Enhancements to IMPS
- Enhancements to CMF
- esd/node states and state transitions
- Security
- enode
- DEBUG shell
- Logging and dumping
- SMW HA

Copyright 2018 Cray Inc.

Documentation

- *XC™ Series SMW-managed eLogin Installation Guide (CLE 6.0.UP06 S-3020 Rev C or later)*
- *XC™ Series SMW-managed eLogin Administration Guide (CLE 6.0.UP06 S-3021)*
- **Existing documentation**
  - *XC™ Series Ansible Play Writing Guide (CLE 6.0.UP06 S-2582)*
  - *XC™ Series SMW HA Installation Guide (SLEHA12.SP3.UP06 S-0044)*
  - *XC™ Series Software Installation and Configuration Guide (CLE 6.0.UP06 S-2559)*
  - *XC™ Series System Administration Guide (CLE 6.0.UP06 S-2393)*
  - *XC™ Series System Configurator User Guide (CLE 6.0.UP06 S-2560)*

Copyright 2018 Cray Inc.

Agenda

- Managing eLogin nodes from SMW
- **Migrating from CMC/eLogin to SMW/eLogin**
- Managing ARM XC nodes
- Q&A

Copyright 2018 Cray Inc.

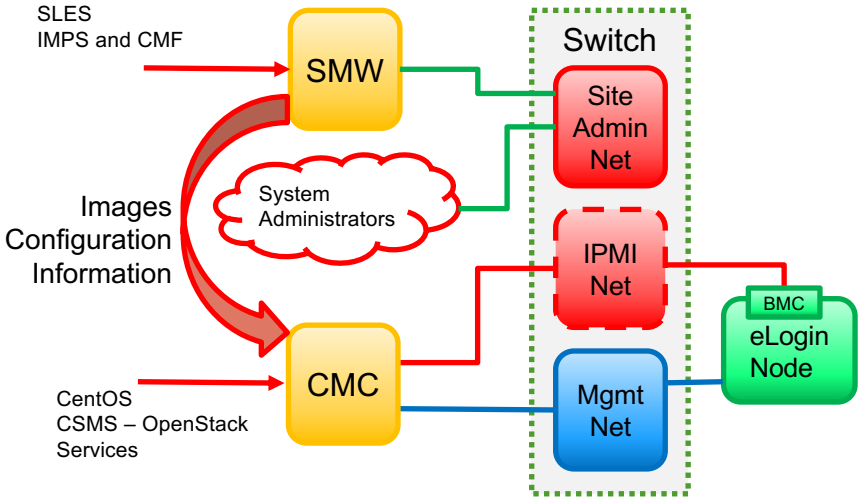
Migration from CIMS/CDL or CMC/eLogin to SMW/eLogin

- Open an SFDC case with Cray Customer Service
  - Download smw_enode_migration tool
    - PDF of instructions for its use, *MIGRATION: ESLOGIN, CDL, ELOGIN TO CLE-6.0.UP06*
- Use the Cray smw_enode_migration tool to gather configuration data from CIMS/CDL or CMC/eLogin
  - This tool generates an inventory.csv file, which will be used to register eLogin nodes with the SMW
- Follow the installation and configuration process paying attention to the marked “migration” sections
  - *XC Series SMW-managed eLogin Installation Guide (CLE 6.0.UP06) S-3020 Rev C (or later)*

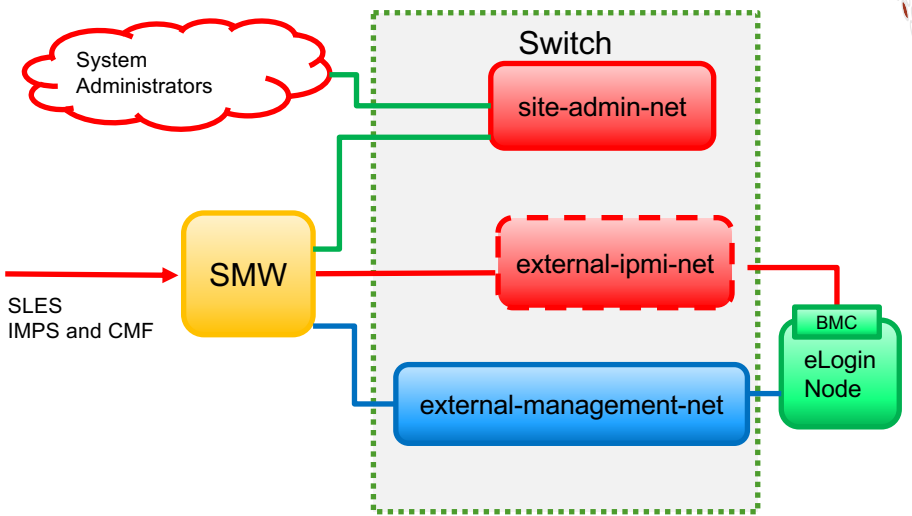
Copyright 2018 Cray Inc.

Migration from CMC/eLogin to SMW/eLogin

Before UP06



UP06



Requires 2 additional Ethernet ports on the SMW

- Single SMW will need an additional quad port Ethernet card
- HA SMW already has enough ports

Copyright 2018 Cray Inc.

Migration from CMC/eLogin to SMW/eLogin

- **Config data needed from CMC**
  - node_name (hostname of eLogin node)
  - BMC_IP (iDRAC IP address on elogin-ipmi-net)
  - MAC address (eth0 on elogin-management-net)
  - Migration tool will capture this data in a format usable by SMW/eLogin management
- **Ensure any site firewall needs are migrated to the cray_firewall configuration service**
- **Config data needed from the SMW**
  - config sets used by eLogins
  - Any image recipe modifications made by the site
  - Any package collections and/or packages added by the site for eLogin images

Copyright 2018 Cray Inc.

enode enroll Command (UP06)

- **Enroll nodes from an inventory.csv file**
  - Used for migration from CMC/eLogin and CIMS/esLogin to SMW/eLogin management
  - Nodes must be subsequently updated with the enode update command for missing information

```
smw# cat /root/inventory.csv
NODE_NAME, BMC_IP, MAC_ADDR, N_CPUs, ARCH, RAM_MB, DISK_GB, NODE_DESC
flubber-elogin1,10.148.0.8,90:b1:1c:26:93:62,32,x86_64,131072,550,flubber-
elogin1
flubber-skl,10.148.0.9,18:66:DA:94:4D:32,32,x86_64,131072,550,flubber-skl
smw# enode enroll /root/inventory.csv
Creating the following nodes:
flubber-elogin1
flubber-skl
Successfully created ['flubber-elogin1', 'flubber-skl'].
```


Copyright 2018 Cray Inc.

enode enroll Command Output (UP06)

```
smw# enode list
NAME    CONFIGSET STORAGE_PROFILE ESD_GROUP IMAGE             BMC_IP    MGMT_IP   MGMT_MAC
     PARAMETERS  STATE
elogin1 -          elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:33
     -           UNKNOWN
elogin2 -          elogin_default  elogin    elogin-smw-large 10.6.0.6 10.7.0.6 00:11:22:33:44:22
     -           UNKNOWN
elogin3 -          elogin_default  elogin    elogin-smw-large 10.6.0.4 10.7.0.4 00:11:22:33:44:55
     -           UNKNOWN
elogin4 -          elogin_default  elogin    elogin-smw-large 10.6.0.5 10.7.0.5 00:11:22:33:44:88
     -           UNKNOWN
elogin5 p0         elogin_default  elogin    elogin-smw       10.6.0.7 10.7.0.7 90:B1:1C:3A:0A:1B
     -           UNKNOWN
flubber-elogin1 - -                 None      -                10.148.0.8  0.0.0.0
90:b1:1c:26:93:62  -        UNKNOWN
flubber-skl     - -                 None      -                10.148.0.9  0.0.0.0
18:66:DA:94:4D:32  -        UNKNOWN
```

Note: BMC_IP is not on 10.6.0.0 network so this will have to be changed with “enode update”

Copyright 2018 Cray Inc.

enode enroll Command (UP07)

- **Enroll nodes from two CSV formats**
  - Migration from CMC/eLogin and CIMS/esLogin in inventory.csv format

    ```
    NODE_NAME, BMC_IP, MAC_ADDR, N_CPUs, ARCH, RAM_MB,
    DISK_GB, NODE_DESC
    ```

  - Output from enode list in csv format

    ```
    NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,
    MGMT_IP,MGMT_MAC,PARAMETERS,STATE,ROOTDEV,BOOTIF,PCI,R
    EMCON,BMC_USERNAME,KDUMP_ENABLE,KDUMP_HIGH,KDUMP_LOW,S
    SH_HOST_KEYS
    ```


Copyright 2018 Cray Inc.

enode enroll Command Output (UP07)

- Save current node registry to a csv file

```
smw# enode list --format csv --fields all --output file.csv
smw# cat file.csv
NAME,CONFIGSET,STORAGE_PROFILE,ESD_GROUP,IMAGE,BMC_IP,MGMT_IP,MGMT_MAC,PARAMETERS,STATE,ROOTDEV,BOOTIF,PCI,REMCON,BMC_USERNAME,KDUMP_ENABLE,KDUMP_HIGH,KDUMP_LOW,SSH_HOST_KEYS
flubber-elogin1,p0,elogin_default,elogin,elogin-smw-large_cle_6.0.UP07-build6.0.7084_sles_12sp3-created20180514,10.6.0.1,10.7.0.1,80:18:44:EB:BA:74,,node_up,/dev/sda,eth0,pci=bfsort,"ttyS1,115200n8",root,False,,,simple_sync
```

- Modify csv file somehow, then import it

```
smw# vi file.csv
smw# enode enroll file.csv
```


Copyright 2018 Cray Inc.

Configuration Service: cray_cfgset_exclude

- **New configuration service in CLE config set**
  - Replaces `add_configset` script on the CMC
- **Allows users to specify config set content to exclude when they are pushed (rsync'd) to other nodes**
  - Used for security and performance
- **Node Groups aware**
- **Supports rsync exclude wildcards**
- **Pre-populated with elogin exclude data used in pre-UP06**
  - Be sure to migrate any site changes from the CMC /etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist to the SMW's CLE config set service cray_cfgset_exclude

Copyright 2018 Cray Inc.

Configuration Service: cray_cfgset_exclude

```
cray_cfgset_exclude:
    settings:
        profiles:
            data:
            -   key: elogin_security
                exclude_content:
                - config/cray_sdb_config.yaml
                - config/cray_drc_config.yaml
                - config/cray_lmt_config.yaml
                - files/roles/common/etc/ssh
                - files/roles/common/root
                - files/roles/munge
                - files/roles/common/etc/opt/cray/xtremoted-agent
                - files/roles/merge_account_files
                - files/simple_sync/common/files/etc/ssh
                - files/simple_sync/common/files/root/.ssh
                - worksheets
                groups:
                - elogin_nodes
```

Profile name

Exclude list

Node group(s)

Copyright 2018 Cray Inc.

`cfgset push` with Exclude Profiles

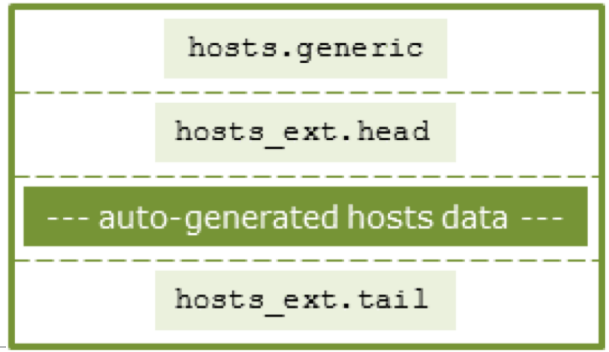
- **`cfgset push` on a cle config set**
  - Excludes files for each host as defined in the config set **`cray_cfgset_exclude`** service
  - The config set specified via **`--ref-config-set`** is used to look up exclude profiles and node groups
  - If no **`--ref-config-set`** is specified, exclude profiles are looked up in the config set being pushed
  - **`--no-exclude-errors`** option allows for pushing of old config sets without exclude profiles defined
- **`cfgset push` on a global config set does not exclude any files**

Copyright 2018 Cray Inc.

Migrate /etc/hosts Entries

- **If any site host entries were directly added to /etc/hosts for CMC/eLogin**
  - Move that content to either hosts_ext.head or hosts_ext.tail
    - /var/opt/cray/imps/config/sets/<CONFIG_SET>/files/roles/common/etc/
  - Content will be assembled into hosts.external for eLogin nodes

[eLogin node]:/etc/hosts



Copyright 2018 Cray Inc.

eLogin Node Storage Configuration

- **Reconfigure eLogin node internal RAID as two virtual disks**
  - sda for volatile storage
  - sdb for persistent storage
  - Guidelines in documentation
- **Plan storage profile for node**
  - Details in cray_storage configuration service
  - Guidelines in documentation
- *XC™ Series SMW-managed eLogin Installation Guide (CLE 6.0.UP06 S-3020 Rev C or later)*

Copyright 2018 Cray Inc.

Agenda

- Managing eLogin nodes from SMW
- Migrating from CMC/eLogin to SMW/eLogin
- **Managing ARM XC nodes**
- Q&A

Copyright 2018 Cray Inc.

ARM Based XC50 Supercomputer

In 2017, Cray announced that it is creating the world's first production-ready, Arm®-based supercomputer with the addition of Cavium ThunderX2™ processors, based on 64-bit Armv8-A architecture, to the Cray® XC50™ supercomputer.

The system features a full software environment, including the Cray Linux Environment, the Cray Programming Environment, and Arm-optimized compilers, libraries, and tools for running today's supercomputing workloads.

Copyright 2018 Cray Inc.

Agenda

- Review Changes that were introduced in CLE6.0UP06 in support of building and booting aarch64 based images on an ARM based Cray XC50 supercomputer

NOTE: These changes affect any system that adopts the CLE6.0UP06 release, not just those that have ARM nodes.

Copyright 2018 Cray Inc.

Agenda – Managing ARM XC Nodes

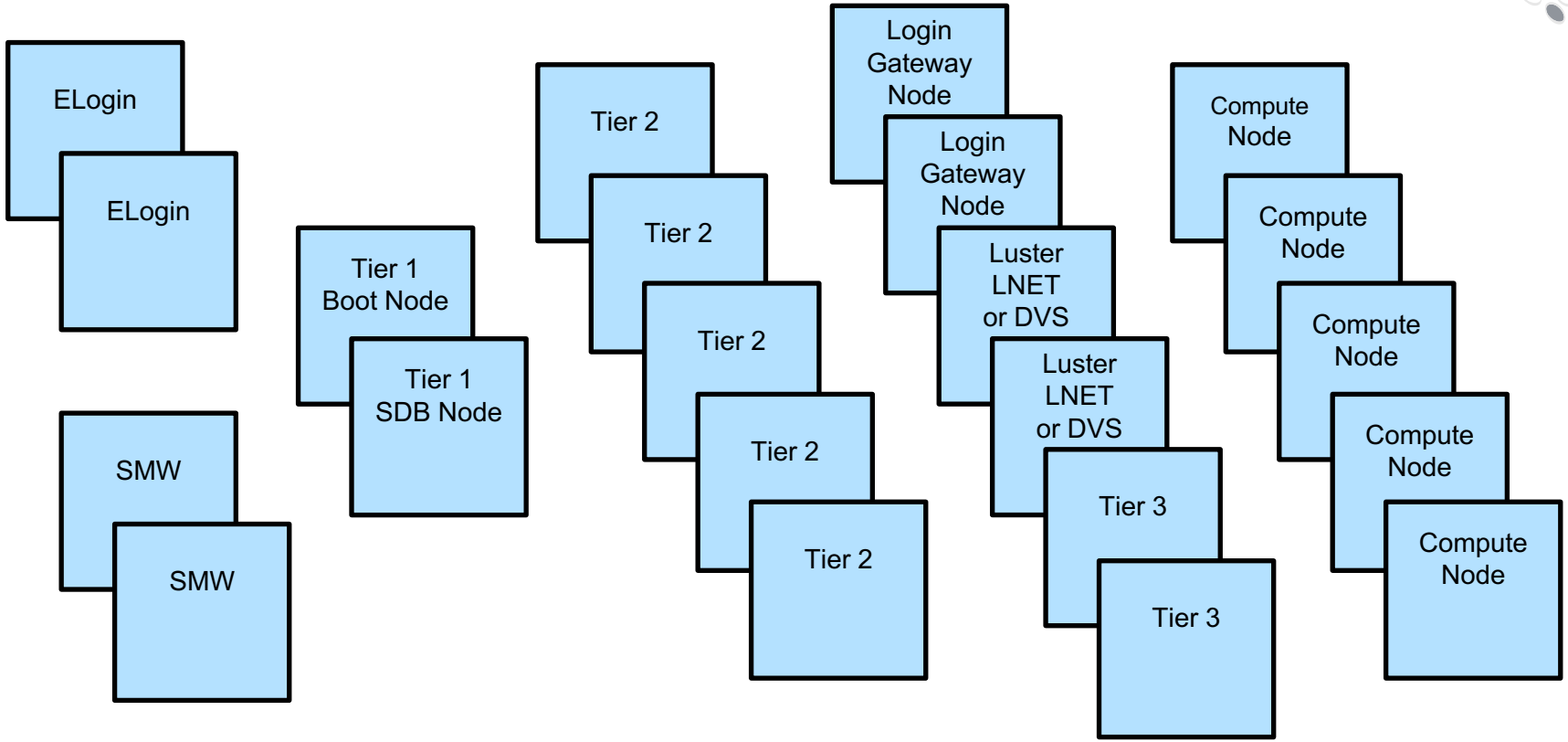
- **Review Changes that were introduced in CLE6.0UP06 in support of building and booting aarch64 based images on an ARM based Cray XC50 supercomputer**
  - Building aarch64 images on an x86_64 SMW
  - Changes to the Image Management and Provisioning System (IMPS)
  - Demo
  - Booting ARM Images

  NOTE: These changes affect any system that adopts the CLE6.0UP06 release, not just those that have ARM nodes

Copyright 2018 Cray Inc.

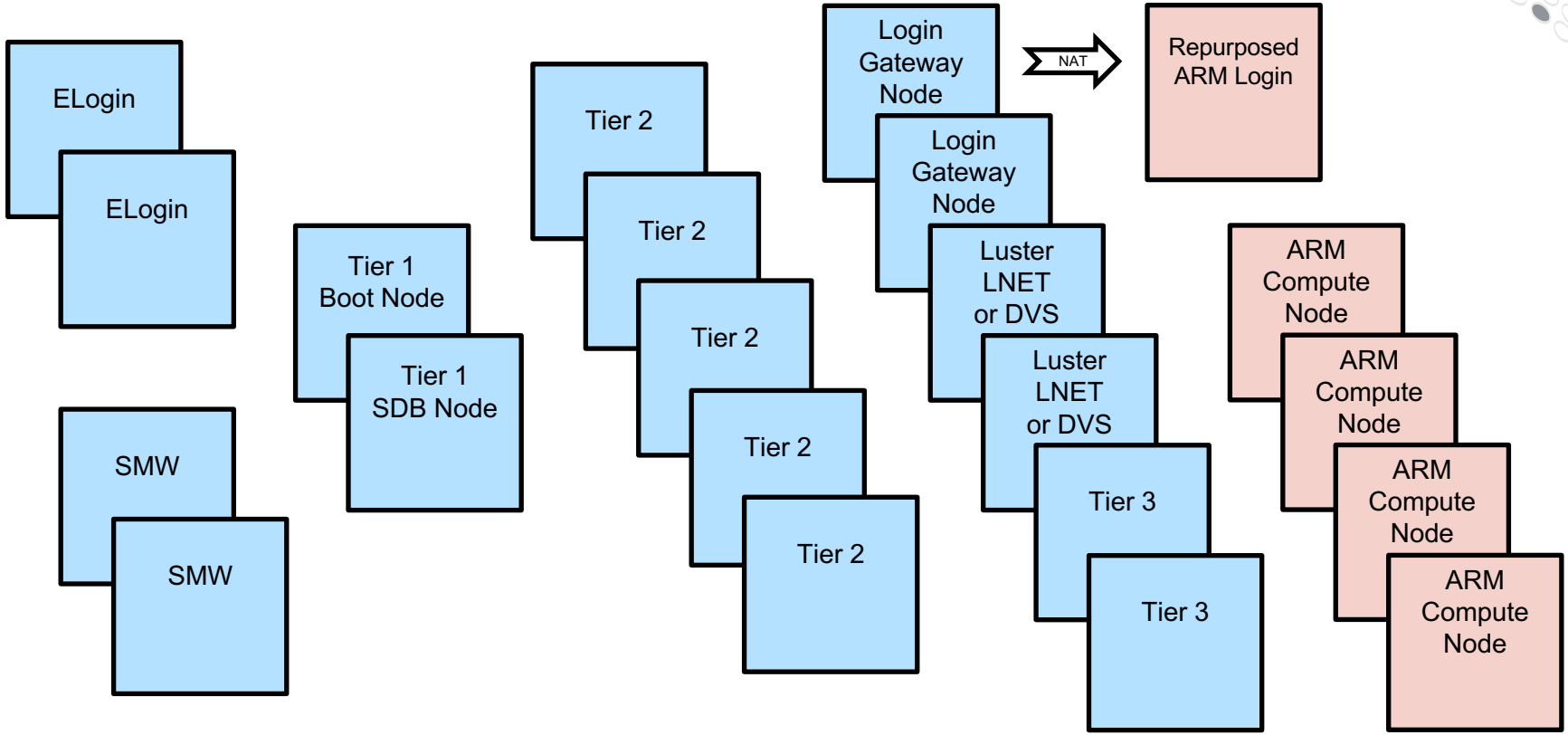
Building aarch64 images on an x86_64 SMW

Traditional Homogeneous Architecture



Copyright 2018 Cray Inc.

New Heterogeneous Architecture



Copyright 2018 Cray Inc.

System Management Architecture

Prior to CLE6.0UP06, Cray tools did not have support for building or booting images of disparate architectures.

Copyright 2018 Cray Inc.

Zypper and RPM

x86_64 SMW == x86_64 Image Root

Copyright 2018 Cray Inc.

So the Question Became…

How do we build images for a different processor architecture?

Copyright 2018 Cray Inc.

Requirements

- **Need way to build non-x86_64 image roots**
- **Minimize the workflow differences between building images for x86_64 and "other" processors.**
  - We wanted the the solution to be "familiar" and work with the Cray Image Management and Provisioning System (IMPS) tools.

Copyright 2018 Cray Inc.

Options

1. Dedicated ARM Hardware

2. Full Virtualization (Linux KVM, XEN, VirtualBox, etc)

3. Lightweight Virtualization (QEMU User Mode Emulation + JeOS environment)

Copyright 2018 Cray Inc.

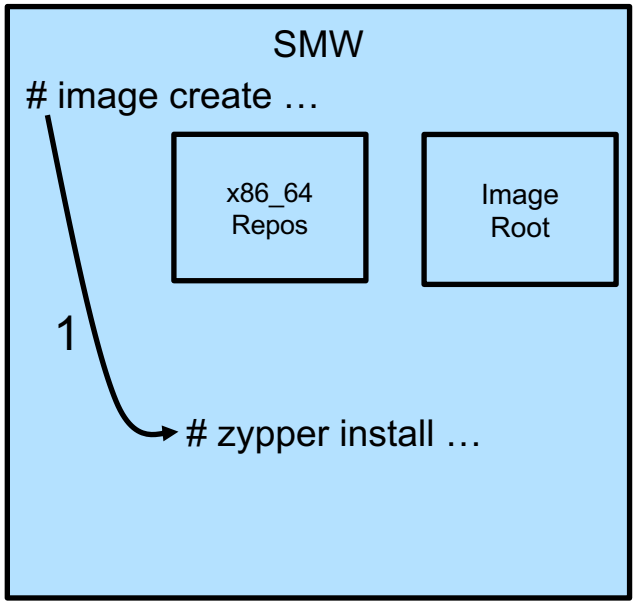
QEMU User Mode Emulation

- The user mode emulation functionality of QEMU is essentially a translation layer
- Allows compiled executables that do not match the host processor to be executed on the host.

Copyright 2018 Cray Inc.

Building an x86_64 Based Image

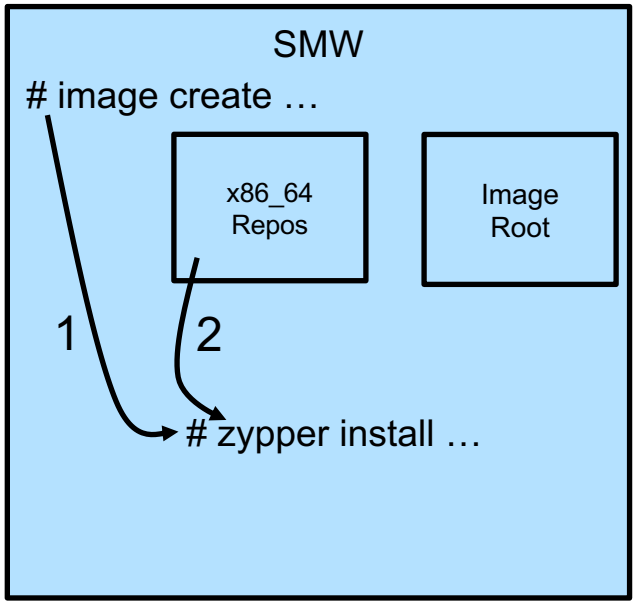
1. The IMPS Image create command calls zypper with the the repos to use, the destination image root and list of packages to install



Copyright 2018 Cray Inc.

Building an x86_64 Based Image

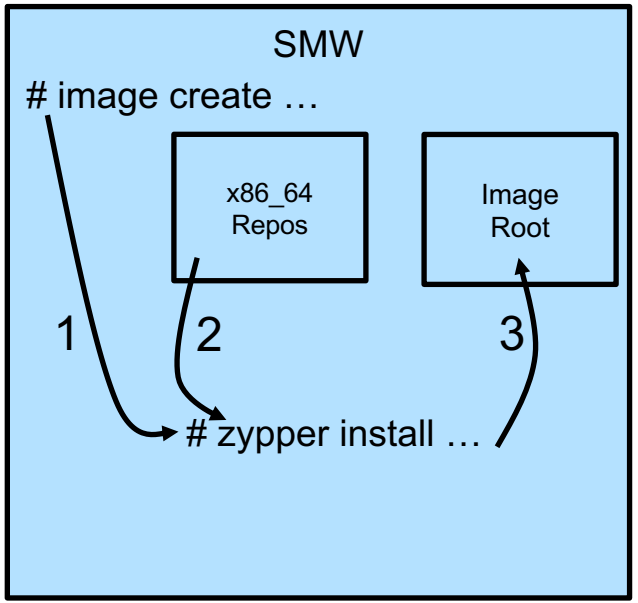
1. The IMPS Image create command calls zypper with the the repos to use, the destination image root and list of packages to install
2. Zypper resolves any RPM dependencies



Copyright 2018 Cray Inc.

Building an x86_64 Based Image

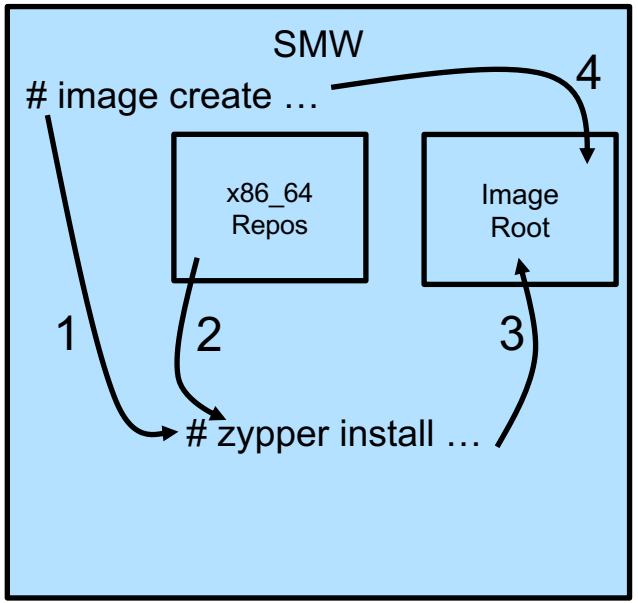
1. The IMPS Image create command calls zypper with the the repos to use, the destination image root and list of packages to install
2. Zypper resolves any RPM dependencies
3. RPM Content is unpacked into the destination image root



Copyright 2018 Cray Inc.

Building an x86_64 Based Image

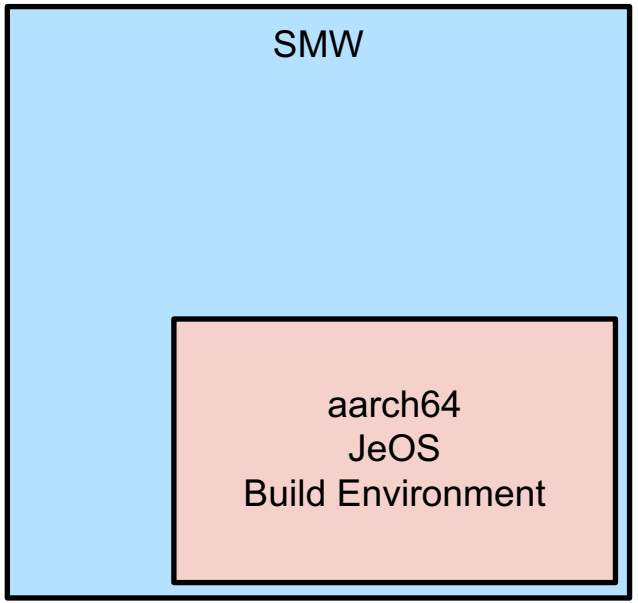
1. The IMPS Image create command calls zypper with the the repos to use, the destination image root and list of packages to install
2. Zypper resolves any RPM dependencies
3. RPM Content is unpacked into the destination image root
4. Image create command runs any postbuild steps



Copyright 2018 Cray Inc.

Creating an aarch64 Build Environment

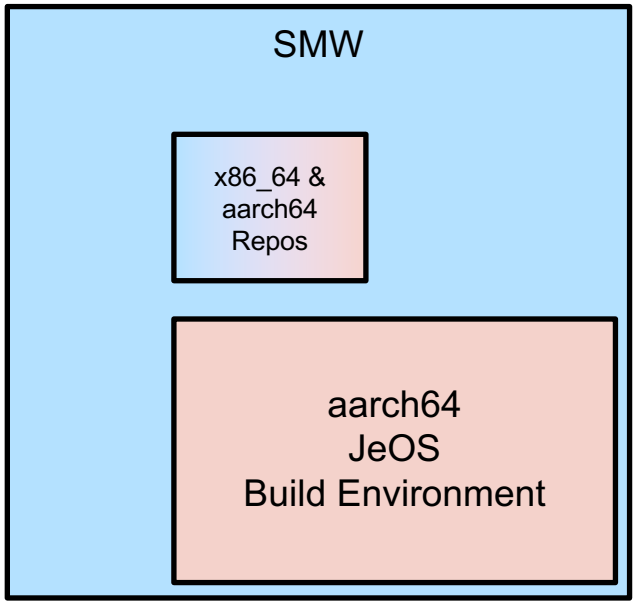
Just Enough OS (JeOS) chroot environment with a minimal ARM based SLES installation



Copyright 2018 Cray Inc.

Adding aarch64 RPM Content

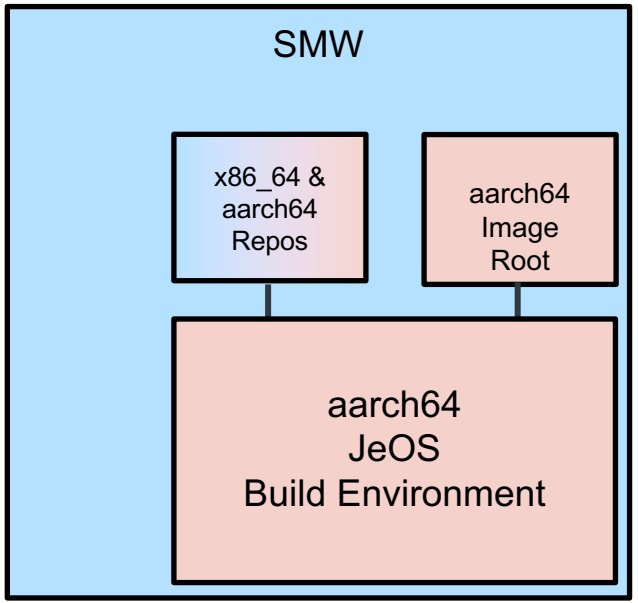
- Renamed Cray provided repositories
- Organized rpms by architecture
- Added aarch64 RPMs required to build image roots
- Removed ability to tag a repository for a given architecture.



Copyright 2018 Cray Inc.

Mounting In Required Directories

- /var/opt/cray/imps
- /var/opt/cray/repos
- /proc
- /sys
- /dev
- /dev/pts

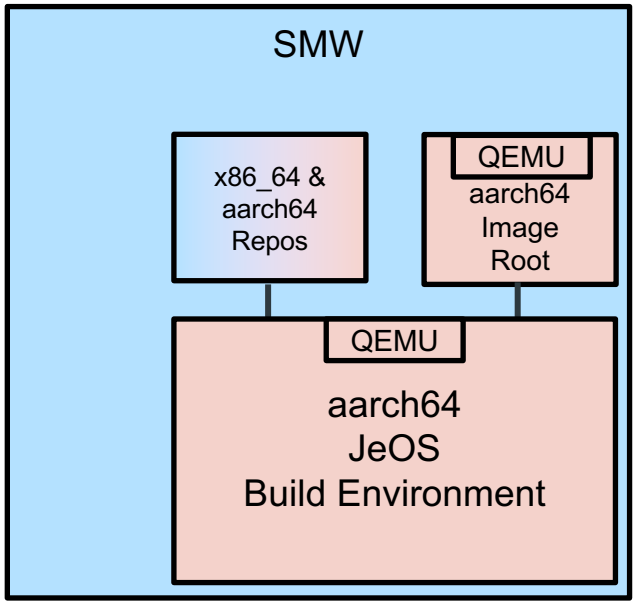


Copyright 2018 Cray Inc.

Add QEMU Binaries to aarch64 Environments

Copy QEMU into the JeOS build environment and target image root.

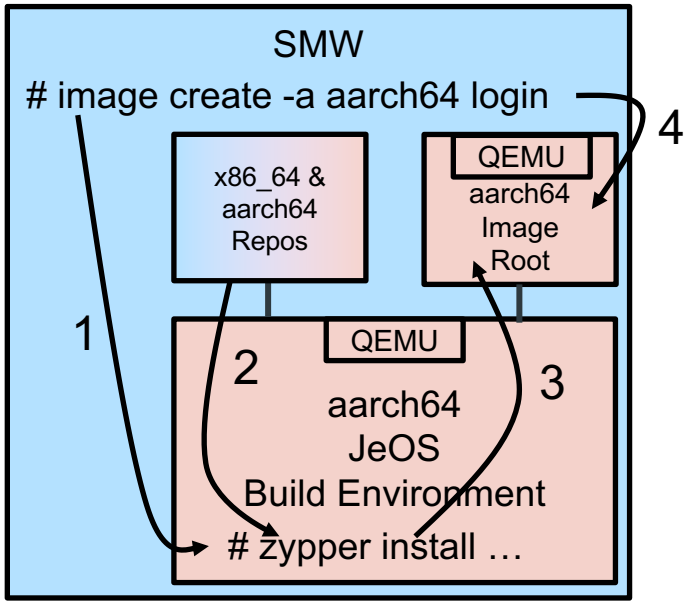
- The Kernel expects the QEMU binaries in the directory `/usr/bin/qemu-aarch64*`
- When we chroot, this location falls outside of our new root



Copyright 2018 Cray Inc.

Building the aarch64 Image Root

Call zypper within the JeOS build environment instead of on the host system



Copyright 2018 Cray Inc.

It Worked! But…

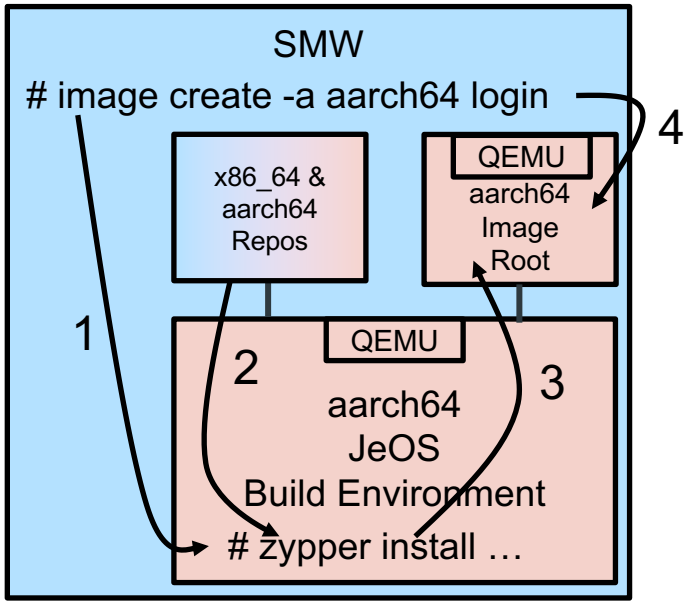
Target	Compute	Initrd-compute	Login	Initrd-compute-large
x86_64	120.726s	44.850s	194.748s	44.247s
aarch64	480.750s	286.256s	986.743s	290.378s
Increase	4x	6.5x	5x	6.5x

Copyright 2018 Cray Inc.

Analyzing the Problem

Two places that we were using QEMU:

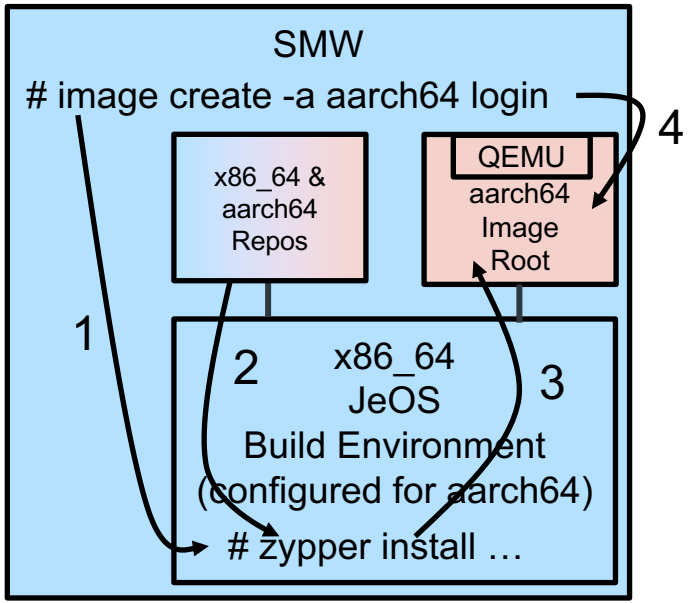
1. During the zypper install stage
2. When running post-build scripts



Copyright 2018 Cray Inc.

The Solution

- Use an x86_64 build environment instead of an aarch64 build environment
- Configure the x86_64 build environment to prefer aarch64 RPMs.
  - /etc/zypp/zypp.conf
  - /etc/rpm/platform



Copyright 2018 Cray Inc.

Success!

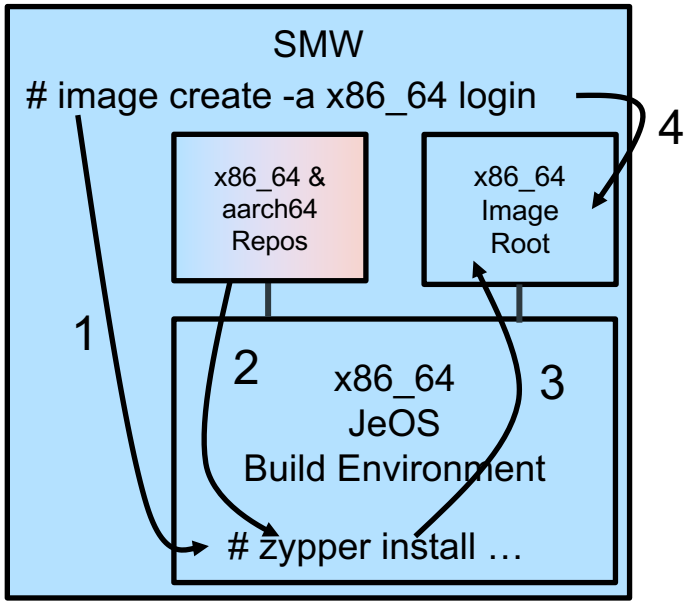
Target	Compute	Initrd-compute	Login	Initrd-compute-large
x86_64	120.726s	44.850s	194.748s	44.247s
aarch64 (no QEMU)	205.368s	150.697s	557.582s	151.090s
aarch64 (QEMU)	480.750s	286.256s	986.743s	290.378s
Increase	1.7x / 4x	3.4x / 6.5x	2.9x / 5x	3.4x / 6.5x

Copyright 2018 Cray Inc.

Looking to the Future

We may look at building x86_64 image roots using an x86_64 JeOS build environment

- Better separation of and support for concurrent image builds
- Additional system security and protections



Copyright 2018 Cray Inc.

Changes to the Image Management and Provisioning System (IMPS)

Changing the Rest of IMPS

- **Add multi-architecture support to the rest of the IMPS commands (image, recipe, pkgcoll and repo)**
  - Recipes, Package Collections and Repositories updated to support different architectures
  - Update IMPS CLIs to support new multi-architecture command line options
  - Implement new `image chroot` command
  - Enable new methods to organize rpms when using repo update
  - Update image builder to support multiple architectures

Copyright 2018 Cray Inc.

Requirements

- **A single recipe should support building an image for disparate processor targets**
  - There should be a single source of truth for what constitutes a given image type
  - Different architectures may require different RPMs
- **Not all recipes need to be built for more than one architecture (admin, etc)**
  - Add protections to prevent recipes being built targeting unsupported architectures and configurations

Copyright 2018 Cray Inc.

Recipe and Package Collection Data Structure Changes

The file format for both recipes and package collections were changed fairly significantly:

- Changed serialization schema
- Added support for Jinja2 templating in recipes and package collections
- Added several new tags

Copyright 2018 Cray Inc.

Recipe Schema Changes

```
"recipe_name": {
    "description": "",
    "dist": "SUSE",
    "default-arch": "x86_64",
    "valid-arch": [ "x86_64", "aarch64" ],
    "packages": [
        {'name': 'common_package_1', 'rationale': ' ... '},
        {'name': 'common_package_2', 'rationale': ' ... '},
        "{% if arch == 'aarch64' %}{'name': 'aarch64_package', 'rationale': ' ... '}{% endif %}",
        "{% if arch == 'x86_64' %}{'name': 'x86_64_packag', 'rationale': ' ... '}{% endif %}",
        {'name': 'common_package_3', 'rationale': ' ... '},
        {'name': 'common_package_4', 'rationale': ' ... '}
    ],
    "package_collections": [
      ... Similar to packages ...
    ],
    "recipes": [
      ... Similar to packages ...
    ],
    "repositories": [
      ... Similar to packages ...
    ],
```

A recipe can now target more than one architecture. However, a recipe can only target a single Linux distribution.

Packages, Package Collections, Recipes and Repos are all now a list instead of a dict of dicts..

Packages, Package Collections, Recipes and Repos are all now a list instead of a dict of dicts. The items in the lists can either be a dictionary or a jinja2 marked string of a valid json dict.

Copyright 2018 Cray Inc.

Recipe Schema Changes

```
"postbuild_chroot": [
      "common_command_1",
      "common_command_2",
      "{% if arch == 'aarch64' %}aarch64_command_1{% endif %}",
      "{% if arch == 'x86_64' %}x86_64_command_1{% endif %}",
      "common_command_3",
      "common_command_4"
      ...
   ],
   "postbuild_copyfiles": [
      "/path/to/file/commom_copyfile_1",
      "/path/to/file/commom_copyfile_2",
      "{% if arch == 'aarch64' %}/path/to/file/aarch64_copyfile_1{% endif %}",
      "{% if arch == 'x86_64' %}/path/to/file/aarch64_copyfile_2{% endif %}",
      "/path/to/file/commom_copyfile_3"
      "/path/to/file/commom_copyfile_4"
      ...
   ],
   "version": "2.0.0"
}
```

Postbuild_chroot and postbuild_copy commands can now be marked up with jinja2 syntax.

A new version field. This is for internal use of the recipe and pkgcoll commands only.

Copyright 2018 Cray Inc.

PkgColl Schema Changes

```
"package_collection_name": {
  "description": "",
  "dist": "SUSE",
  "packages": [
    {'name': 'common_package_1', 'rationale': ' ... '},
    {'name': 'common_package_2', 'rationale': ' ... '},
    "{% if arch == 'aarch64' %}{'name': 'aarch64_package', 'rationale': ' ... '}{% endif %}",
    "{% if arch == 'x86_64' %}{'name': 'x86_64_packag', 'rationale': ' ... '}{% endif %}",
    {'name': 'common_package', 'rationale': ' ... '},
    {'name': 'common_package', 'rationale': ' ... '},
    ...
  ],
  "package_collections": [
    {'name': 'common_package_collection_1', 'rationale': ' ... '},
    {'name': 'common_package_collection_2', 'rationale': ' ... '},
    "{% if arch == 'aarch64' %}{'name': 'aarch64_package_collection', 'rationale': ' ... '}{% endif %}",
    "{% if arch == 'x86_64' %}{'name': 'x86_64_package_collection', 'rationale': ' ... '}{% endif %}",
    {'name': 'common_package_collection_3', 'rationale': ' ... '},
    {'name': 'common_package_collection_4', 'rationale': ' ... '},
    ...
  ],
  "version": "2.0.0"
}
```

A package collection now specifies the Linux distribution that it supports. NOTE: you cannot specify either valid-arch or default-arch for a pkgcoll.

Packages, Package Collections, Recipes and Repos are all now a list instead of a dict of dicts..

The items in the lists can either be a dictionary or a jinja2 marked string of a valid json dict.

A new version field. This is for internal use of the recipe and pkgcoll commands only.

Copyright 2018 Cray Inc.

Linux Distribution

When building or validating a recipe, the IMPS tools will ensure that that the top level recipe, sub-recipes, package collections and repositories all have a consistent distribution

- This is to help uncover mistakes where a recipe accidentally includes objects for a different Linux distribution

Copyright 2018 Cray Inc.

Default vs. Valid Architectures

- Default Architecture: The architecture that will be used when the IMPS tools build or evaluate a recipe if the user did not otherwise specify a target architecture on the command line

- Valid Architectures: A list of architectures for which a recipe can be built or evaluated – Must include the default architecture

Copyright 2018 Cray Inc.

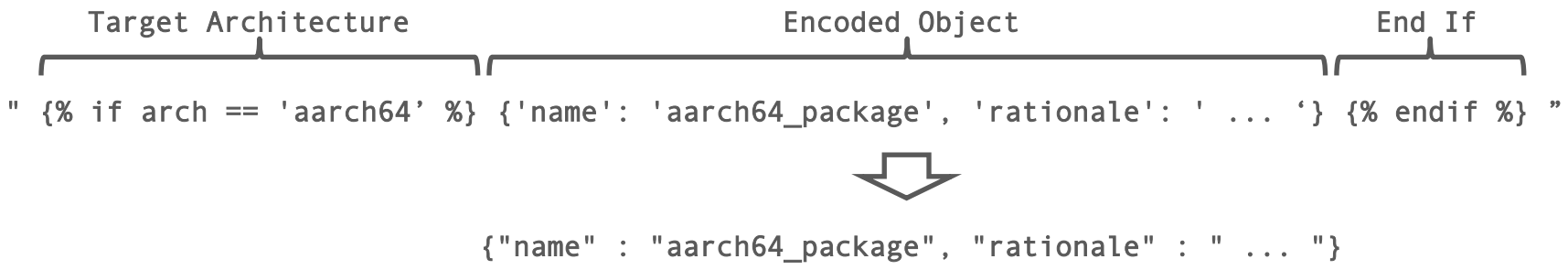
Architecture Specific Objects

- **Different architectures may require different RPMs and post-install steps**
  - need a way to be able to “specialize” IMPS recipes and package collections
- **Objects (Recipes, package collections and repositories) and postbuild (copyfiles and chroot) actions can be conditionally added using Jinja2 encoded strings**
  - As the recipe or package collection is processed, each conditional string is rendered individually to determine if the architecture specific object or post build action should be included or not

Copyright 2018 Cray Inc.

Rendering of Architecture Specific Objects

Recipes, Package Collections and Repository Objects

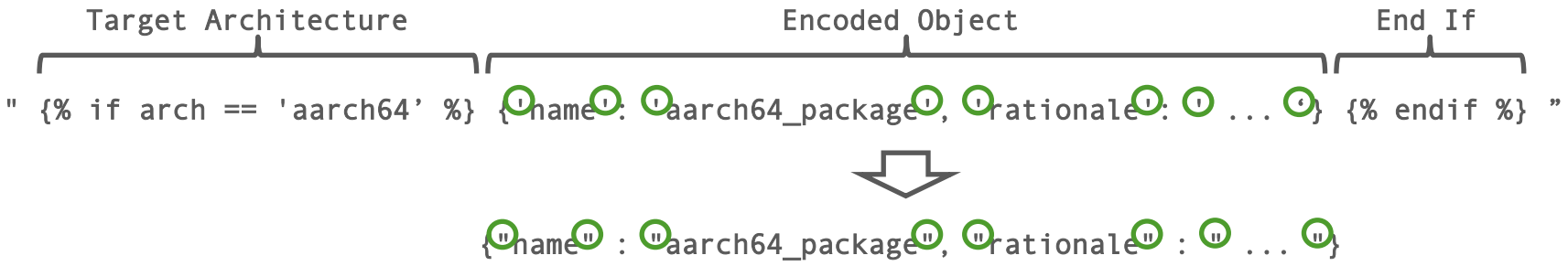


To increase readability of the encoded JSON object, the IMPS tools will translate single quotes to double quotes automatically

Copyright 2018 Cray Inc.

Rendering of Architecture Specific Objects

Recipes, Package Collections and Repository Objects

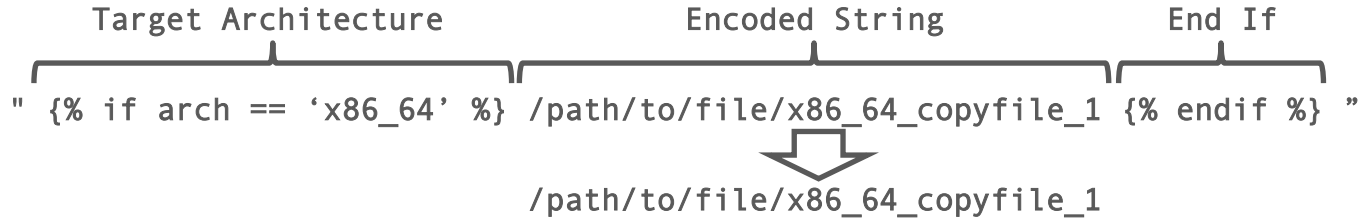


To increase readability of the encoded JSON object, the IMPS tools will translate single quotes to double quotes automatically

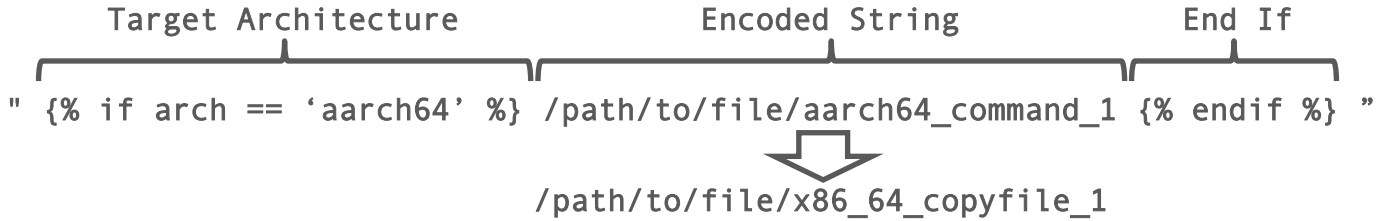
Copyright 2018 Cray Inc.

Rendering of Architecture Specific Post Build

Postbuild Copy Files



Postbuild Chroot



Copyright 2018 Cray Inc.

recipe and pkgcoll Command Changes

New CLI options were added to:

- Add and remove architecture specific objects
- Set the default architecture and valid architectures for recipes.
- Set the distribution for recipes and package collections
- Changes to recipe validate to support different architectures.

Copyright 2018 Cray Inc.

Setting a Recipe's dist, default-arch and valid-arch

When creating or editing a recipe by the command line, you can set the dist, default-arch and valid-arch values:

```
smw # recipe create [options] recipe_name
smw # recipe update [options] recipe_name

-t DIST, --dist DIST
                        Set the distribution of the recipe.
-e ARCH, --default-arch ARCH
                        Set the default architecture.
-l ARCH, --add-valid-arch ARCH
                        Update the list of valid-architectures.


-L ARCH, --remove-valid-arch ARCH
                        Remove ARCH from the list of valid architectures.
```

Recipe Create

Recipe Update

Copyright 2018 Cray Inc.

Setting a Package Collection's Distribution

When creating or editing a package collection using the command line, you can now set the Linux distribution

```
smw # pkgcoll create [-t DIST, --dist DIST] pkgcoll_name
smw # pkgcoll update [-t DIST, --dist DIST] pkgcoll_name

-t DIST, --dist DIST    Set the distribution of the package_collection
```


Copyright 2018 Cray Inc.

Adding/Removing Architecture Specific Objects

Use the new [–A ARCH, --arch ARCH] command line option when adding / removing architecture specific objects

- If not specified, objects will be added such that they will be used when building both x86_64 and aarch64 architectures
- In order to remove an architecture specific object, you must specify the [-A ARCH, --arch ARCH] flag

Copyright 2018 Cray Inc.

Adding/Removing Architecture Specific Objects

```
smw # recipe update -A ARCH [options] recipe_name
smw # pkgcoll update -A ARCH [options] recipe_name

-A ARCH, --arch ARCH  Specify the architecture for add/remove operations.
                      Specifically this applies to packages, package
                      collections, sub-recipes, and repos.

-p PACKAGE, --add-pkg PACKAGE
                      add/update RPMs specified by PACKAGE to the recipe
-P PACKAGE, --remove-pkg PACKAGE
                      remove RPMs specified by PACKAGE from the recipe
-c COLLECTION, --add-coll COLLECTION
                      add/update COLLECTION to the recipe
-C COLLECTION, --remove-coll COLLECTION
                      remove COLLECTION from the recipe
...
```


Copyright 2018 Cray Inc.

Adding Architecture Specific Objects

```
smw # recipe update -r my_repo my_recipe

    ”repositories": [
        {'name': ’my_repo', 'rationale': ' ... '},
    ]

smw # pkgcoll update -A aarch64 -p my_aarch64_package my_pkgcoll

    ”packages": [
        "{% if arch == ’aarch64’ %}
         { 'name': 'my_aarch64_package', 'rationale': ’...’ }
         {% endif %}",
    ]
```


Copyright 2018 Cray Inc.

Removing Architecture Specific Objects

```
smw # recipe update -R my_repo my_recipe

    ”repositories": [
        {'name': ’my_repo', 'rationale': ' ... '},
    ]

smw # pkgcoll update -A aarch64 –P my_aarch64_package my_pkgcoll

    ”packages": [
        "{% if arch == ’aarch64’ %}
         { 'name': 'my_aarch64_package', 'rationale': ’...’ }
         {% endif %}",
    ]
```


Copyright 2018 Cray Inc.

Recipe validate, diff and show

- recipe validate, diff and show default to evaluating recipes using the recipe's 'default-arch'
- This can be overridden using the [–A ARCH, --arch ARCH] option

```
recipe show [-A ARCH,--arch ARCH] recipe_name
recipe diff [-A ARCH,--arch ARCH] recipe_name
recipe validate [-A ARCH,--arch ARCH] recipe_name

-A ARCH, --arch ARCH  Evaluate the recipe for the given ARCH
                      Defaults to the default-arch value
                      specified in the recipe.
```


Copyright 2018 Cray Inc.

Repository Changes

- Cray provided repositories now include x86_64, aarch64 and noarch rpms
- Cray-provided repositories no longer contain ‘x86_64’ in their name.

Copyright 2018 Cray Inc.

Repo Create / Repo Update

- **Repositories are no longer tagged to a particular architecture**
  - repo command no longer accepts the --arch parameter.

- **The option to set the repo distribution was changed:**

[-t DIST, --type DIST]

to

[-t DIST, --dist DIST]

Copyright 2018 Cray Inc.

Repo Create / Repo Update

- **Repositories are no longer tagged to a particular architecture**
  - repo command no longer accepts the --arch parameter.

- **The option to set the repo distribution was changed:**

[-t DIST, --type DIST]

to

[-t DIST, --dist DIST]

Copyright 2018 Cray Inc.

Repo Update

- Using the IMPS repo update command, RPMs will now be organized by architecture

- This can be overridden using the new --organize-by and --custom-dir options

```
-o METHOD, --organize-by METHOD
                      how should RPMs being added to the repo be organized
                      (Default: AUTO_ORGANIZE_BY_ARCH)
-c SUBDIR, --custom-dir SUBDIR
                      sub-directory to add rpms to when using
                      ADD_TO_CUSTOM_SUBDIR
```


Copyright 2018 Cray Inc.

Organize By Options

-o METHOD, --organize-by METHOD

Establishes the method by which rpms and files being added via the –add parameter will be organized in the repository directory. The following methods are allowed.

* AUTO_ORGANIZE_BY_ARCH - The repo command will attempt to determine the architecture of the rpm being added, and place the rpm file into a subdirectory matching the architecture. Files being copied into the repo that are not rpms have no inherent architecture and will be added to the root of the repository.
* ADD_TO_REPOSITORY_ROOT - All files copied into the repository will be placed in the repository root regardless of architecture or other consideration.
* MAINTAIN_EXISTING_DIRS - If the path pointed to by --add PATH is a directory, all files and subdirectories below the root PATH will be maintained and duplicated under the repository root.
* ADD_TO_CUSTOM_SUBDIR - All files copied into the repository will be placed in the sub-directory passed by the --custom-dir SUBDIR parameter.

Copyright 2018 Cray Inc.

Repo Update Examples

```
/tmp/rpm/
    1/my_first_rpm-1.0-0.noarch.rpm
    2/my_second_rpm-1.0-0.x86_64.rpm
    3/my_third_rpm-1.0-0.aarch64.rpm
```

```
repo update -a "/tmp/rpm/" -o AUTO_ORGANIZE_BY_ARCH my_repo
```

```
/var/opt/cray/repos/my_repo
    noarch/my_first_rpm-1.0-0.noarch.rpm
    x86_64/my_second_rpm-1.0-0.x86_64.rpm
    aarch64/my_third_rpm-1.0-0.aarch64.rpm
```

```
repo update -a "/tmp/rpm/" -o ADD_TO_REPOSITORY_ROOT my_repo
```

```
/var/opt/cray/repos/my_repo
    my_first_rpm-1.0-0.noarch.rpm
    my_second_rpm-1.0-0.x86_64.rpm
    my_third_rpm-1.0-0.aarch64.rpm
```

```
repo update -a "/tmp/rpm/" -o MAINTAIN_EXISTING_DIRS my_repo
```

```
/var/opt/cray/repos/my_repo
    1/my_first_rpm-1.0-0.noarch.rpm
    2/my_second_rpm-1.0-0.x86_64.rpm
    3/my_third_rpm-1.0-0.aarch64.rpm
```

```
repo update -a "/tmp/rpm/" -o ADD_TO_CUSTOM_SUBDIR -c foo my_repo
```

```
/var/opt/cray/repos/my_repo
    foo/my_first_rpm-1.0-0.noarch.rpm
    foo/my_second_rpm-1.0-0.x86_64.rpm
    foo/my_third_rpm-1.0-0.aarch64.rpm
```


Copyright 2018 Cray Inc.

Known Issues With the Repo Command

In CLE6.0UP06 the following are known issues:

- Repo validate will show failure for any repo that contains at least one aarch64 rpm
- Repo show will not show any aarch64 packages

These will be fixed in a future release

Copyright 2018 Cray Inc.

Image Create

- **Image create command now accepts the target architecture when building an image**

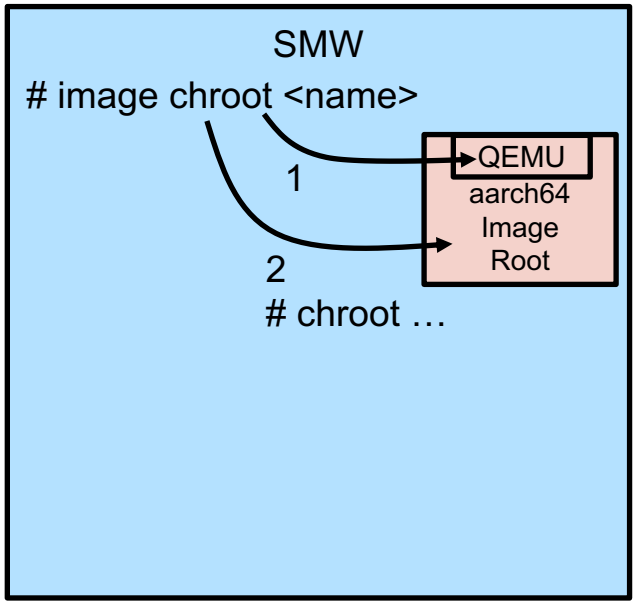
```
-a ARCH, --arch ARCH    Set the target architecture for the image to be ARCH.
                        Defaults to the default-arch value in the recipe.
                        NOTE: only valid if creating an image from a recipe.
```

- **If no target architecture is specified, the default-arch specified in the recipe is used**

Copyright 2018 Cray Inc.

image chroot

- Works for both x86_64 and aarch64 image roots
- Just need to know the name of the image root
- For aarch64 based images, the ‘image chroot’ command will automatically copy in the required QEMU binaries



Copyright 2018 Cray Inc.

image chroot Command

```
smw # image chroot -help
usage: image chroot [-h] [-V] [-v] [-q] IMAGE [COMMAND [ARG]...]

positional arguments:
  IMAGE                 name of image to create
  COMMAND               Optional command to run. If no command is given, run
                        '${SHELL} -i' (default: '/bin/sh -i').
  ARG...                Argument(s) to pass to optional command

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         print version header
  -v, --verbose         increase the verbosity of the command
  -q, --quiet           suppresses unnecessary output

examples:
  # image chroot login_cle_rhine_sles_12_x86-64_ari
```


Copyright 2018 Cray Inc.

imgbuilder Command Changes

- Image builder can now build images targeting both x86_64 and aarch64 architectures
- Recipes can be listed multiple times, each with a different arch value

```
cray_image_groups:
  default:
      ...
   - recipe: "compute_cle_6.0up06_sles_12sp3_ari"
     dest: "compute{note}_..._sles_12sp3-x86_-created{date}.cpio"
     arch: "x86_64"
     nims_group: "compute"
   - recipe: "compute_cle_6.0up06_sles_12sp3_ari"
     dest: "compute{note}_...-aarch64-created{date}.cpio"
     arch: "aarch64"
     nims_group: "compute_aarch64"
...
```


Copyright 2018 Cray Inc.

imgbuilder Command Changes

- Image builder can now build images targeting both x86_64 and aarch64 architectures
- Recipes can be listed multiple times, each with a different arch value

```
cray_image_groups:
  default:
      ...
    - recipe: "compute_cle_6.0up06_sles_12sp3_ari"
      dest: "compute{note}_..._sles_12sp3-x86_-created{date}.cpio"
      arch: "x86_64"
      nims_group: "compute"
    - recipe: "compute_cle_6.0up06_sles_12sp3_ari"
      dest: "compute{note}_...-aarch64-created{date}.cpio"
      arch: "aarch64"
      nims_group: "compute_aarch64"
...
```


Copyright 2018 Cray Inc.

IMPS Multi-Architecture Support Cheat Sheet

#	Command Line	Set Distribution Target -t DIST, --dist DIST	Set Default Arch -e ARCH, --default-arch ARCH	Set Valid Arch -I ARCH, --add-valid-arch ARCH	Remove Valid Arch -L ARCH, --remove-valid-arch ARCH	Evaluate for Arch -A ARCH, --arch ARCH	--fields
1	imgbuilder	Can now specify an arch within a recipe block in the cray_image_groups.yaml file. A recipe can be specified multiple times for different architectures.					
2	image create					x	
3	image show						dist, arch
4	image list						dist, arch
5	image chroot	New Command that can be used to chroot into image roots.					
6	image diff	Show differences in installed packages, which may include architecture differences					
7	image validate	Verify that QEMU binaries are not in the image root.					
8	image export	Architecture specific load address changes. Add image architecture to exported CPIO file					
9	recipe create	x	x	x			
10	recipe show					x	dist, default-arch, valid-arch
11	recipe update	x	x	x	x	x	
12	recipe validate					x	
13	recipe diff					x	dist, default-arch, valid-arch
14	pkgcoll create	x					
15	pkgcoll show					x	dist
16	pkgcoll update	x				x	
17	pkgcoll validate					x	
18	repo create	x	Removed --arch parameter. Changed --type to --dist for consistency.				
19	repo update	x	Removed --arch parameter. Changed --type to --dist for consistency. Added --organize-by and --custom-dir to add RPMs into sub-directories.				
20	repo list						Removed arch Changed type to dist
21	repo show						Removed arch Changed type to dist
22	cnode update	Add --no-arch-check flag to prevent architecture validation					arch
23	cnode list						arch

Copyright 2018 Cray Inc.

Demo

Putting This All Together…

For the purposes of demonstration, lets create a recipe that can be built for either x86_64 or aarch64

- Both architectures should include ‘curl’
- x86_64 should also include ‘emacs’
- aarch64 should also include ‘wget’

Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
INFO - Successfully created new empty Recipe 'example'

smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
INFO - Successfully created new empty Recipe 'example'

smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example

smw # recipe update -p curl –a “add curl” example
INFO - Successfully extended Recipe 'example' with 1 Package.

smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example

smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
INFO - Successfully extended Recipe 'example' with 1 Package.

smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example

smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
INFO - Successfully extended Recipe 'example' with 1 Package.

smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example

smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
INFO - Successfully extended Recipe 'example' with 1 Package.

smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example

smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
INFO - Successfully extended Recipe 'example' with 1 Package.

smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example

smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
INFO - Successfully extended Recipe 'example' with 1 Recipe.

smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example

smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
INFO - Successfully extended Recipe 'example' with 1 Recipe.

smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example

smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example
INFO - Successfully extended Recipe 'example' with 1 Repository.

smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
smw # recipe create -l aarch64 --description "my example recipe" example
smw # recipe update -p curl –a “add curl” example
smw # recipe update -A aarch64 -p wget –a “add wget for aarch64” example
smw # recipe update -A x86_64 -p emacs –a “add emacs for x86_64” example
smw # recipe update -i seed_common_6.0up06_sles_12sp3 –a “base recipe” example
smw # recipe update -r sle-server_12sp3_updates –a “updates repo” example

smw # recipe update -r sle-server_12sp3 –a “reqd for curl, wget, emacs” example
INFO - Successfully extended Recipe 'example' with 1 Repository.
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
"example": {
    "dist": "SLES12",
    "default_arch": "x86_64",
    "valid_arch": [ "aarch64", "x86_64" ],
    "description": "my example recipe",
    "package_collections": [],
    "packages": [
        { "name": "curl", "rationale": "add curl" },
        "{% if arch=='aarch64' %}{'name': 'wget', 'rationale': 'add wget for aarch64'}{% endif %}",
        "{% if arch=='x86_64' %}{'name': 'emacs', 'rationale': 'add emacs for x86_64'}{% endif %}"
    ],
    "postbuild_chroot": [],
    "postbuild_copy": [],
    "recipes": [
        { "name": "seed_common_6.0up06_sles_12sp3", "rationale": "base recipe" }
    ],
    "repositories": [
        { "name": "sle-server_12sp3_updates", "rationale": "updates repo" },
        { "name": "sle-server_12sp3", "rationale": "reqd for curl, wget, emacs" }
    ],
}
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
"example": {
    "dist": "SLES12",
    "default_arch": "x86_64",
    "valid_arch": [ "aarch64", "x86_64" ],
    "description": "my example recipe",
    "package_collections": [],
    "packages": [
        { "name": "curl", "rationale": "add curl" },
        "{% if arch=='aarch64' %}{'name': 'wget', 'rationale': 'add wget for aarch64'}{% endif %}",
        "{% if arch=='x86_64' %}{'name': 'emacs', 'rationale': 'add emacs for x86_64'}{% endif %}"
    ],
    "postbuild_chroot": [],
    "postbuild_copy": [],
    "recipes": [
        { "name": "seed_common_6.0up06_sles_12sp3", "rationale": "base recipe" }
    ],
    "repositories": [
        { "name": "sle-server_12sp3_updates", "rationale": "updates repo" },
        { "name": "sle-server_12sp3", "rationale": "reqd for curl, wget, emacs" }
    ],
}
```


Copyright 2018 Cray Inc.

Putting This All Together…

```
"example": {
    "dist": "SLES12",
    "default_arch": "x86_64",
    "valid_arch": [ "aarch64", "x86_64" ],
    "description": "my example recipe",
    "package_collections": [],
    "packages": [
        { "name": "curl", "rationale": "add curl" },
        "{% if arch=='aarch64' %}{'name': 'wget', 'rationale': 'add wget for aarch64'}{% endif %}",
        "{% if arch=='x86_64' %}{'name': 'emacs', 'rationale': 'add emacs for x86_64'}{% endif %}"
    ],
    "postbuild_chroot": [],
    "postbuild_copy": [],
    "recipes": [
        { "name": "seed_common_6.0up06_sles_12sp3", "rationale": "base recipe" }
    ],
    "repositories": [
        { "name": "sle-server_12sp3_updates", "rationale": "updates repo" },
        { "name": "sle-server_12sp3", "rationale": "reqd for curl, wget, emacs" }
    ],
}
```

Copyright 2018 Cray Inc.

Putting This All Together…

```
"example": {
    "dist": "SLES12",
    "default_arch": "x86_64",
    "valid_arch": [ "aarch64", "x86_64" ],
    "description": "my example recipe",
    "package_collections": [],
    "packages": [
        { "name": "curl", "rationale": "add curl" },
        "{% if arch=='aarch64' %}{'name': 'wget', 'rationale': 'add wget for aarch64'}{% endif %}",
        "{% if arch=='x86_64' %}{'name': 'emacs', 'rationale': 'add emacs for x86_64'}{% endif %}"
    ],
    "postbuild_chroot": [],
    "postbuild_copy": [],
    "recipes": [
        { "name": "seed_common_6.0up06_sles_12sp3", "rationale": "base recipe" }
    ],
    "repositories": [
        { "name": "sle-server_12sp3_updates", "rationale": "updates repo" },
        { "name": "sle-server_12sp3", "rationale": "reqd for curl, wget, emacs" }
    ],
}
```

Copyright 2018 Cray Inc.

Building for default (x86_64) arch

```
smw # image create -r example example-x86_64
INFO - Building Image 'example-x86_64'
INFO - Building out example > seed_common_6.0up06_sles_12sp3
INFO - Calling package manager to build Recipe 'seed_common_6.0up06_sles_12sp3'; this
can take a few minutes.
INFO - Build of Recipe 'seed_common_6.0up06_sles_12sp3' has completed successfully.
INFO - Building out example
INFO - Calling package manager to build Recipe 'example'; this can take a few
minutes.
INFO - Rebuilding Image 'example-x86_64' RPM database.
INFO - Image 'example-x86_64' RPM database successfully rebuilt.
```


Copyright 2018 Cray Inc.

Building for aarch64

```
smw # image create -A aarch64 -r example example-aarch64
INFO - Building Image 'example-aarch64'
INFO - Building out example > seed_common_6.0up06_sles_12sp3
INFO - Calling package manager to build Recipe 'seed_common_6.0up06_sles_12sp3'; this
can take a few minutes.
INFO - Build of Recipe 'seed_common_6.0up06_sles_12sp3' has completed successfully.
INFO - Building out example
INFO - Calling package manager to build Recipe 'example'; this can take a few
minutes.
INFO - Rebuilding Image 'example-aarch64' RPM database.
INFO - Image 'example-aarch64' RPM database successfully rebuilt.
```


Copyright 2018 Cray Inc.

Inspecting the x86_64 Recipe

```
smw # recipe show -A x86_64 example
example:
  name: example
  distribution: SLES12
  default_architecture: x86_64
  valid_architectures:
    aarch64
    x86_64
  rendered_repositories_with_details:
    sle-server_12sp3
    sle-server_12sp3_updates
  rendered_recipes_with_details:
    seed_common_6.0up06_sles_12sp3
  rendered_packages_with_details:
    curl
    emacs
```


Copyright 2018 Cray Inc.

Chrooting into the x86_64 Image Root

```
smw # image chroot example-x86_64
[example-x86_64] root@node:/ # rpm -qa | grep curl
libcurl4-7.37.0-37.8.1.x86_64
curl-7.37.0-37.8.1.x86_64

[example-x86_64] root@node:/ # rpm -qa | grep wget

[example-x86_64] root@node:/ # rpm -qa | grep emacs
emacs-24.3-25.3.1.x86_64
emacs-info-24.3-25.3.1.noarch
emacs-nox-24.3-25.3.1.x86_64

[example-x86_64] root@node:/ # exit
```


Copyright 2018 Cray Inc.

Inspecting the aarch64 Image Root

```
smw # recipe show -A aarch64 example
example:
  name: example
  distribution: SLES12
  default_architecture: x86_64
  valid_architectures:
    aarch64
    x86_64
  rendered_repositories_with_details:
    sle-server_12sp3
    sle-server_12sp3_updates
  rendered_recipes_with_details:
    seed_common_6.0up06_sles_12sp3
  rendered_packages_with_details:
    curl
    wget
```


Copyright 2018 Cray Inc.

Chrooting into the aarch64 Image Root

```
smw # image chroot example-aarch64
[example-aarch64] root@node:/ # rpm -qa | grep curl
libcurl4-7.37.0-37.8.1.aarch64
curl-7.37.0-37.8.1.aarch64

[example-aarch64] root@node:/ # rpm -qa | grep wget
wget-1.14-21.3.1.aarch64

[example-aarch64] root@node:/ # rpm -qa | grep emacs

[example-aarch64] root@node:/ # exit
logout
```


Copyright 2018 Cray Inc.

Showing postbuild Steps

```
smw # recipe show --fields postbuild_chroot seed_common_6.0up06_sles_12sp3
seed_common_6.0up06_sles_12sp3:
  postbuild_chroot_scripts:
    /usr/sbin/groupadd --gid 25 at
    /usr/sbin/groupadd --gid 26 postgres
    /usr/sbin/groupadd --gid 400 nginx
    /usr/sbin/groupadd --gid 481 gdm
    /usr/sbin/groupadd --gid 482 scard
    /usr/sbin/groupadd --gid 483 winbind
    /usr/sbin/groupadd --gid 484 svn
    /usr/sbin/groupadd --gid 485 oprofile
    ...
```


Copyright 2018 Cray Inc.

Booting Arm Images

Requirements

- **Add protections to prevent assigning nodes an image of a different architecture**
  - This helps to enable the system to “fail fast” when it is clear that a node won’t boot because the wrong image architecture is being assigned
- **Update Node Groups to target aarch64 nodes separately from x86_64 nodes**

Copyright 2018 Cray Inc.

NIMS & cnode

- The NIMS daemon will now verify that the architecture of an image and the physical node the image is being mapped to are the same

```
smw# cnode update -m eric-test \
      -i /var/opt/cray/imps/boot_images/initrd-compute-large-...-x86_64.cpio c0-0c0s11n3

ERROR - The following nodes have incompatible image architectures based on node type:
c0-0c0s11n3  aarch64  /.../imps/boot_images/initrd-compute-large-...-x86_64.cpio x86_64
```

- This can be overridden using the --no-arch-check flag

Copyright 2018 Cray Inc.

NIMS & cnode

- The NIMS daemon will now verify that the architecture of an image and the physical node the image is being mapped to are the same

```
smw# cnode update -m eric-test \
      -i /var/opt/cray/imps/boot_images/initrd-compute-large-...-x86_64.cpio c0-0c0s11n3

ERROR - The following nodes have incompatible image architectures based on node type:
c0-0c0s11n3   aarch64   /.../imps/boot_images/initrd-compute-large-...-x86_64.cpio x86_64
```

- This can be overridden using the --no-arch-check flag

Copyright 2018 Cray Inc.

NIMS & cnode

The cnode command can now show the architecture of nodes:

```
smw# cnode list --fields name,type,arch,group,image
NAME         TYPE     ARCH     GROUP                          IMAGE
c0-0c0s0n0   service  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s0n1   service  x86_64   admin                          /var/opt/cray/imps/...
c0-0c0s0n2   service  x86_64   login                          /var/opt/cray/imps/...
c0-0c0s0n3   service  x86_64   compute                        /var/opt/cray/imps/...
...
c0-0c0s10n3  compute  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s11n0  service  aarch64  service_aarch64                /var/opt/cray/imps/...
c0-0c0s11n1  compute  aarch64  compute_aarch64,arm_compute    /var/opt/cray/imps/...
c0-0c0s11n2  compute  aarch64  compute_aarch64,arm_compute    /var/opt/cray/imps/...
c0-0c0s11n3  compute  aarch64  compute_aarch64,arm_compute    /var/opt/cray/imps/...
c0-0c0s12n0  compute  x86_64   compute                        /var/opt/cray/imps/...
```


Copyright 2018 Cray Inc.

NIMS & cnode

The cnode command can now show the architecture of nodes:

```
smw# cnode list --fields name,type,arch,group,image
NAME         TYPE     ARCH     GROUP                          IMAGE
c0-0c0s0n0   service  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s0n1   service  x86_64   admin                          /var/opt/cray/imps/...
c0-0c0s0n2   service  x86_64   login                          /var/opt/cray/imps/...
c0-0c0s0n3   service  x86_64   compute                        /var/opt/cray/imps/...
...
c0-0c0s10n3  compute  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s11n0  service  aarch64  service_aarch64                /var/opt/cray/imps/...
c0-0c0s11n1  compute  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s11n2  compute  x86_64   compute                        /var/opt/cray/imps/...
c0-0c0s11n3  compute  aarch64  compute_aarch64,arm_compute    /var/opt/cray/imps/...
c0-0c0s12n0  compute  x86_64   compute                        /var/opt/cray/imps/...
```


Copyright 2018 Cray Inc.

NIMS & cnode

You can also filter nodes by architecture. This works for cnode list and cnode update

```
smw# cnode list --filter arch=aarch64
NAME         TYPE     GROUP                          IMAGE
c0-0c0s11n0  service  service_aarch64                /var/opt/cray/imps/boot_...
c0-0c0s11n3  compute  compute_aarch64,arm_compute    /var/opt/cray/imps/boot_...
```


Copyright 2018 Cray Inc.

NIMS & cnode

You can also filter nodes by architecture. This works for cnode list and cnode update

```
smw# cnode list --filter arch=aarch64
NAME         TYPE     GROUP                        IMAGE
c0-0c0s11n0  service  service_aarch64              /var/opt/cray/imps/boot_...
c0-0c0s11n3  compute  compute_aarch64,arm_compute  /var/opt/cray/imps/boot_...
```


Copyright 2018 Cray Inc.

Node Group Changes

The default set of Cray node groups has changed in CLE6.0UP06 in order to support architecture specific node group definitions

The new node group definitions are installed as part of a fresh install of CLE6.0UP06

Copyright 2018 Cray Inc.

Node Group Changes

When updating to CLE6.0UP06 you must decide whether or not to update your node groups using the documented procedure in S-2559 – "XC™ Series Software Installation and Configuration Guide (CLE 6.0.UP06)"

- **If you need to boot to previous versions of CLE older than CLE6.0UP06, these changes should NOT be made**
- **These changes are required if your system will contain ARM blades**

Copyright 2018 Cray Inc.

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.

Copyright 2018 Cray Inc.


Q&A
Harold Longley htg@cray.com
Eric Cozzi ecozzi@cray.com