# Extending CLE6 to a multi-supercomputer OS

Douglas Jacobsen

NERSC

Cray User Group Meeting 2017

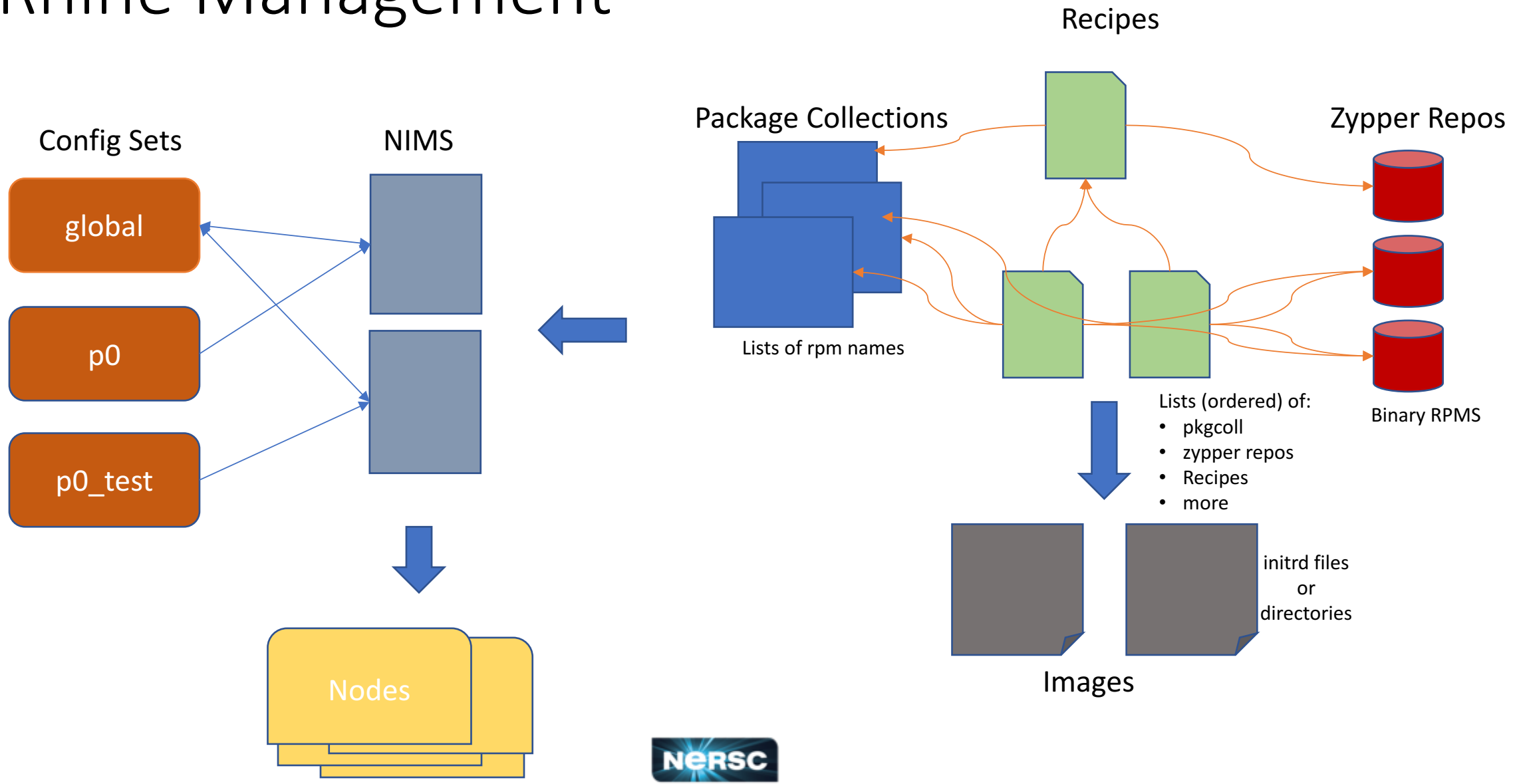
# Motivation

- NERSC operates four Cray XC machines
  - Two well known production machines: cori (XC40) and edison (XC30)
  - Two less well known TDS machines: gerty (XC40-AC) and alva (XC30-AC)
- We have many people contributing to the software configuration of the system
  - Seven Site Administrators in the Computational Systems Group
  - Three Cray-Ons providing Software Support
  - Five or more from other teams directly manipulating parts of the system configuration

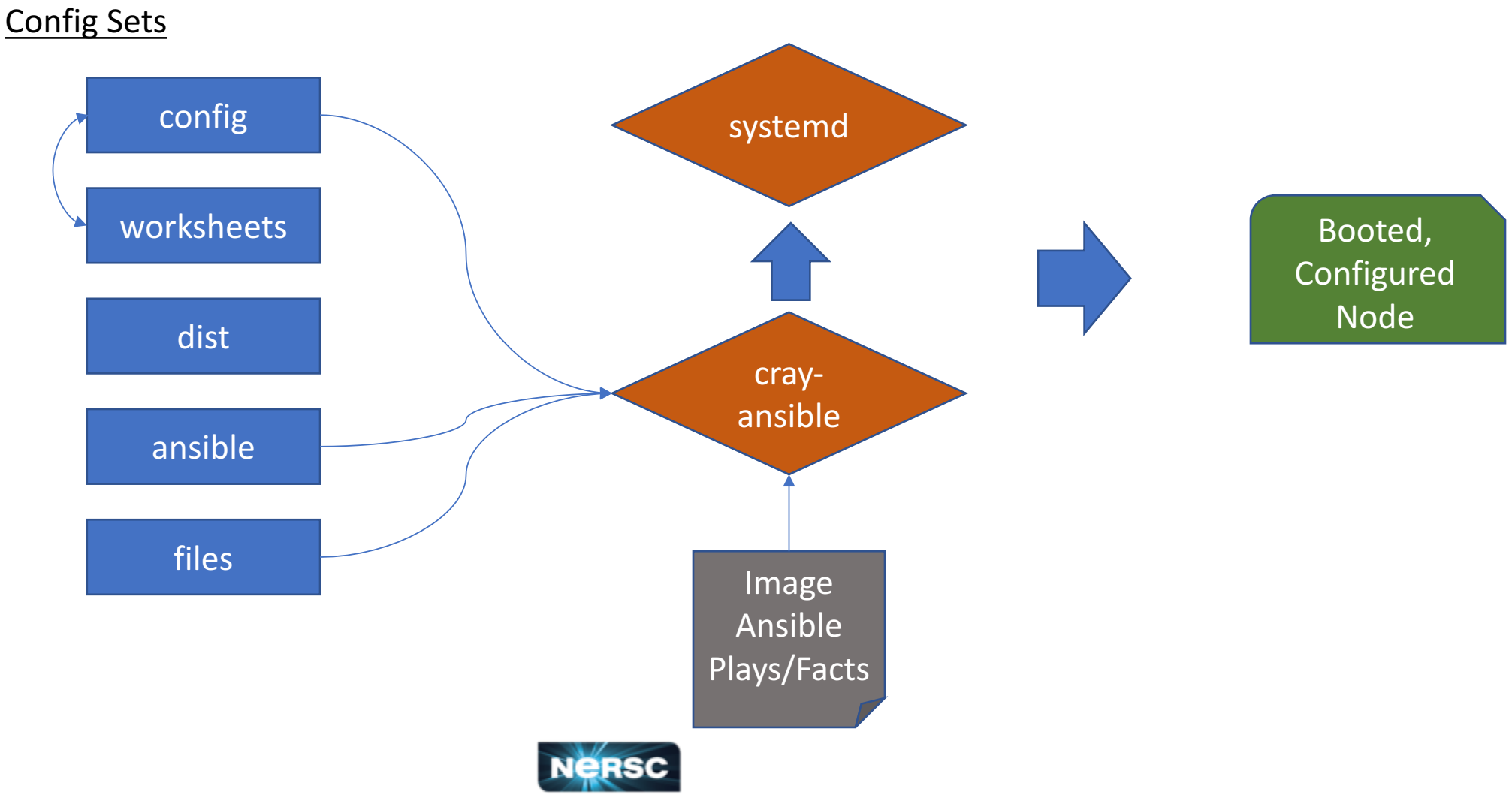
Goals:

- TDS should model as perfectly as possible the production machine for development and reproducing system problems, validating hardware
- TDS should be experimental and possible to wildly change configurations ahead of the production machine to validate new work
- Expense/effort invested in configurations should be directly transferrable to all systems
- All changes should be trackable, traceable, revert-able, reviewable

# Rhine Management



# CLE6 Management



# Co-Managing Multiple Systems

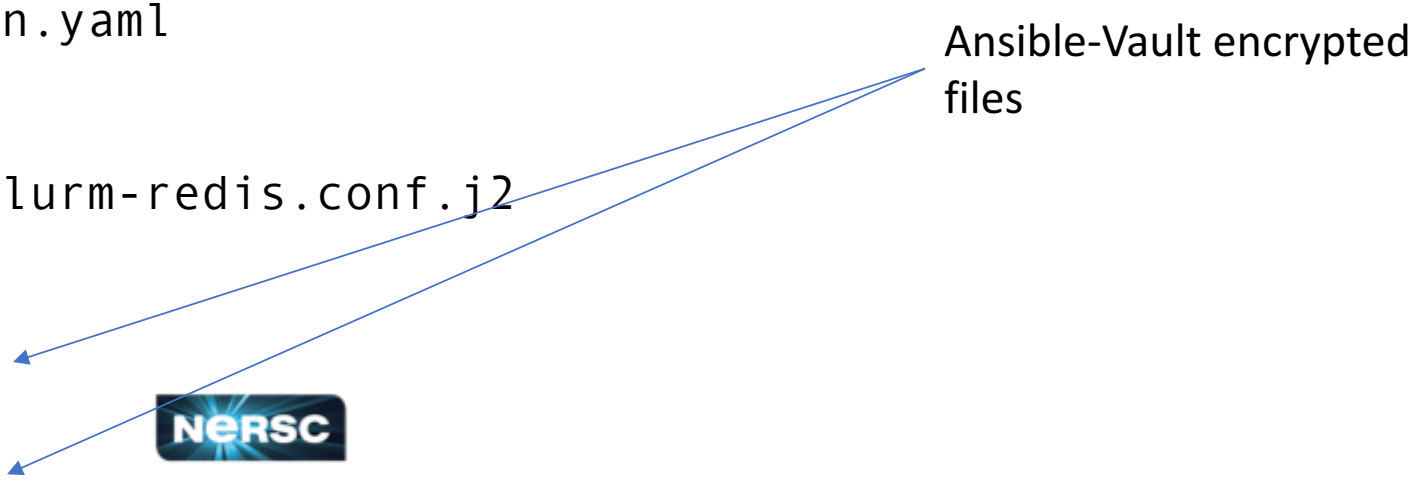
- All platforms run exactly the same site-provided ansible plays
  - Plays operate on platform-specialized variable/fact trees
- All platforms use identical Package Collections
- All platforms use identical Recipes
- All platforms use *mostly* the same RPMs in same-named zypper repos
- Co-manage worksheets across all platforms (not identical in some cases)
- Avoid simple-sync except for credential (munge/certificate) distribution
- Do **everything possible** the Rhine/CLE6/SLES way
  - Means that we prefer to add process and a few simple tools to achieve all this
  - Prefer RPM installation / LiveUpdates to network fs installation for system software

# Ansible

- All plays need to work on all machines
- Achieved by handling machine-specific settings with machine-specific variables, and ensuring we have correct-type defaults in role-specific defaults/main.yaml

Example Role:

```
dmj@corismw1:~/git/corenerscansible/roles/slurm-redis> find .
.
./defaults
./defaults/main.yaml
./handlers./handlers/main.yaml
./tasks
./tasks/main.yaml
./templates./templates/slurm-redis.conf.j2
./vars
./vars/alva.yaml
./vars/alva_secrets.yaml
./vars/cori.yaml
./vars/cori_secrets.yaml
```



# Ansible

- All plays need to work on all machines
- Achieved by handling machine-specific settings with machine-specific variables, and ensuring we have correct-type defaults in role-specific defaults/main.yaml
- Make heavy use of templating

Example role tasks/main.yaml:

```
---
- include_vars: "{{nersc.machineName}}.yaml"
- include_vars: "{{nersc.machineName}}_secrets.yaml”
  no_log: true
- file: path=/etc/redis state=directory mode=0755 owner=root
- template: src=slurm-redis.conf.j2 dest=/etc/redis/slurm-redis.conf
            owner=redis group=redis mode=0600
- file: path={{slurmRedisSaveStateDir}} state=directory
        owner=redis group=redis mode=0700
- file: path=/var/run/redis state=directory
        owner=redis group=redis mode=0700
- service: name=redis@slurm-redis enabled=yes
```

# Ansible Fact Tree Customization

- Inject facts into configset-based facts tree

```
corismw1:/var/opt/cray/imps/config/sets/global/config # cat nersc_vars.yaml
---
nersc:
  machineName: cori
  machineMailingList: <retracted>@nersc.gov
```

- These, like all files in cfgset config/*yaml files, gets linked to /etc/ansible/host_vars/localhost/<integer>.yaml
- Accessible in ansible plays using dictionary naming, i.e., nersc.machineName
- The configurator will ignore so long as missing _config in filename

# Ansible Fact Tree Customization

- Add python script that outputs dynamically discoverable variables in /etc/ansible/facts.d
- Variables accessible in `ansible_local.<filename_no_ext>`

```
boot-cori:~ # /etc/ansible/facts.d/nersc.fact
{"node_groups": ["boot_nodes", "bootnodes"], "machineName": "cori"}
boot-cori:~ #
```

- We used this capability to make node_groups work on XC and eLogin in UP01. The NERSC-provided node_groups variable is less useful in UP03 since UP03 provides similar capabilities (and more)
- Having own fact tree did allow us to migrate our plays extremely quickly since our interface remained unchanged
- Installed in all images using a common PkgColl installing a machine-specific RPM (nersc-ansible-sec)

# Ansible Vault

- Encrypts sensitive information with AES-256 encryption
- Allows us to securely check-in “lesser” passwords (encrypted) into our on-site BitBucket git (more on that later)
- Relies on a key embedded in all images
  - postbuild_copy/postbuild_chroot to copy key from SMW during image construction
- Relies on a modification to /etc/ansible/ansible.cfg (before ansible runs)
  - Post-install scriptlet in nersc-ansible-sec-<machineName>.rpm adds: vault_password_file=/path/we/use/to/ansible.hash
- Separate keys (and secrets) for all platforms, prevents accidental crosstalk between systems
- Vault files only editable on SMW

# Recipes

- Recipes (and package collections) are stored in json files under /etc/opt/cray/imps
- We store these files in git (later)
- We copy the json files and maintain exactly the same recipes, pkgcoll on all systems
- “diff_imps.py” (custom tool) is used to detect changes

```
dmj@corismw1:~/git/nersc-cle6/imps> recipe show nersc-compute-production…
nersc-compute-production-cle_6.0up03_sles_12_x86-64_ari:
  name: nersc-compute-production-cle_6.0up03_sles_12_x86-64_ari
  created: 2016-06-08T08:55:45
  repositories:
    common_cle_6.0up03_sles_12_x86-64_ari_updates
    sle-sdk_12_x86-64
    …
    sle-module_legacy_12_x86-64
    sles_12_x86-64_updates
    nersc-slurm
    common_cle_6.0up03_sles_12_x86-64_ari
  recipes:
    nersc-seed_common_6.0up03_sles_12_x86-64
    compute-large_cle_6.0up03_sles_12_x86-64_ari
    …
    …
  package_collections:
    nersc-slurm-build-deps
    slurm-build_cle_6.0up03_sles_12
    cray-bootlog
    nersc-initrd-addons
    nersc-base-compute
    nersc-blcr
    nersc-slurm-compute
    nersc-ansible
    nersc-vtune-kmod
    nersc-base
  path: /etc/opt/cray/imps/image_recipes.d/image_recipes.local.json
```

# Detecting changes to Recipes/Pkgcoll

```
dmj@gertsmw:~/git/nersc-cle6/imps> git checkout abcdef01 -- image_recipes.local.json
dmj@gertsmw:~/git/nersc-cle6/imps> ./diff_imps.py
sys recipe:nersc-service-production-cle_6.0up03_sles_12_x86-
64_ari:postbuild_chroot:sed -i
's/groups|(network="hsn")/groups|grps2hosts(network="hsn")/g'
/etc/ansible/roles/ntp/tasks/ntp.yaml
sys recipe:nersc-admin-production-cle_6.0up03_sles_12_x86-64_ari:postbuild_chroot
sys recipe:nersc-epurge-production-cle_6.0up03_sles_12_x86-
64_ari:postbuild_chroot:echo -e '#!/bin/bash\nexit 0' > /etc/opt/cray/pre-
pivot.d/32ConfigNetworkUdevRules.sh
```

Computes diffs and displays either “sys” or “git” to tell us what is changed in recipes (like diff “<“ or “>”)

Helper scripts keep git and system in-sync

# Worksheet/Config files in cfgsets

- Worksheets are stored in git (not config yaml) files
- 36 worksheets are identical for all platforms
- 11 are customized for specific platforms (cori_cray_net_worksheet.yaml, etc)
- Update worksheets either singly or en masse:
  - `cfgset update -w nersc-cle6/imps/n8_worksheets/gerty_cray_network_worksheet.yaml --no-scripts p0`

**or**

```
dmj@gertsmw:~/git/nersc-cle6/imps> ./update_worksheets ./n8_worksheets p0
skipping  alva_cray_auth_worksheet.yaml
…
skipping  cori_cray_net_worksheet.yaml
run:  cfgset update -w "/tmp/tmpcsSZbL/*yaml" --no-scripts p0
dmj@gertsmw:~/git/nersc-cle6/imps>
```

**finally**

```
cfgset update p0
```

# Detecting Changes to worksheets

- During patchset installations / etc, values in cfgsets may be changed. After running cfgset update, or installing patches, check for diffs:

```
dmj@gertsmw:~/git/nersc-cle6/imps> ./diff_worksheet.py n8_worksheets p0
skipping alva_cray_auth_worksheet.yaml
…
worksheets in n8_worksheets but not in cfgset p0:  set([])
worksheets in cfgset p0 but not in n8_worksheets:  set([])
…
n8_worksheets/gerty_cray_dws_worksheet.yaml /var/opt/cray/imps/config/sets/p0/worksheets/cray_dws_worksheet.yaml
values differ between git and cfgset for keys:  ['cray_dws.settings.dwmd.data.dwmd_conf']
git: cray_dws.settings.dwmd.data.dwmd_conf,  ['iscsi_initiator_cred_path: /etc/opt/cray/dws/iscsi_target_secret',
       'iscsi_target_cred_path: /etc/opt/cray/dws/iscsi_initiator_secret', \
       'capmc_os_cacert: /etc/pki/trust/anchors/certificate_authority.pem']
cfg: cray_dws.settings.dwmd.data.dwmd_conf,  ['iscsi_initiator_cred_path: /etc/opt/cray/dws/iscsi_target_secret',
       'iscsi_target_cred_path: /etc/opt/cray/dws/iscsi_initiator_secret',
       'capmc_os_cacert: /etc/pki/trust/anchors/certificate_authority.pem', \
       'xfs_mnt_opt: \\"allocsize=1m,nodiscard\\"']
```

# Git Process

## Git Repos:

**CoreNerscAnsible** – has all ansible plays, roles, variables
**nersc-cle6** – has pkgcol, recipes, worksheets for all systems
**nersc-slurm** – slurm configurations and deployments

All content created, committed, and pushed by “regular” uids. root has read-only pull permissions on repos

## Branching:

cle6.0up01
cle6.0up02
cle6.0up03
MyDevelopmentBranchForAwesomeFeature
release/cle6.0up01
release/cle6.0up02
release/cle6.0up03

”ansible” directory in cfgsets is a direct clone of CoreNerscAnsible

IMPs components are updated in-place on the system

# Future Directions

- We still manage zypper repos with rsync and manual calls to ”repo update <reponame”
- Plan to implement remote metarepo with git-supported lists of RPMs that are to be included in each
- This work has not been initiated

- We are currently updating alva to cle6.0up03
- Edison will move (with these techniques) to cle6.0up04 in short order

# Conclusions

- These techniques allowed us to upgrade TDS (gerty) from up01 to up03 in one week.
  - We concurrently updated cori cfgset *on* the gerty smw
- Stored, identical values for ansible, images, worksheets, allowed us to perform upgrade of cori in a single day
  - (if btrfs on the SMW hadn't melted we would have been early!)
- Group is still learning and refining git skills and techniques
  - Proven successful across multiple contributors though
- We can update systems with confidence by correctly performing exactly the same operations, assisted by SCM

# The End

Questions?