Using xCAT and GIT to manage software consistency and DevOps

Ross Mickens
Penn State University

Institute for Computational and Data Sciences

Agenda

- Introductions
- Design principles
- Possible Solutions
- Solution Overview
- Implementation
- Final Thoughts
- Resources

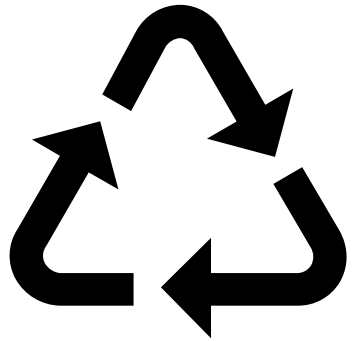
- Ross Mickens – PSU ICDS Engineering Lead

- ICDS
  - "Build capacity to solve problems of scientific and societal importance through interdisciplinary, cyber-enabled research."
  - Est 2012
  - Roar Collab – new cluster environment that will allow for more frequent updates to keep pace with research needs, 2020

- xCAT – **Ex**treme **C**loud **A**dministration **T**oolkit
  - 1999 by IBM
  - Python, Perl
  - Automates deployment, scaling, and management of bare metal and VMs

Roar Collab Design - Principles

- Use of centralized services
- Infrastructure as code
- Stateless/diskless deployment
- Minimize config drift
- Reduce time to deployment
- Extendable and scalable
- Team member accessibility



DevOps Lifecycle:
continuous software development, integration, testing, deployment, and monitoring

Possible Solutions

- Ansible / Foreman
- Chef / Puppet
- Warewulf
- Bright computing
- xCAT

Key Features of xCAT

1. HW management (ipmi)
2. Infrastructure Discovery
3. Remote power control
4. OS Provisioning
5. OS image life cycle management
   Stateless and Stateful (local disk)

- xCAT Objects (Nodes, Groups)
- xCAT Database - data for objects stored in csv tables
  - osimage
  - package list
  - sync list
  - Scripts (PostScript/PostbootScript)

Solution Overview: Image Hierarchy

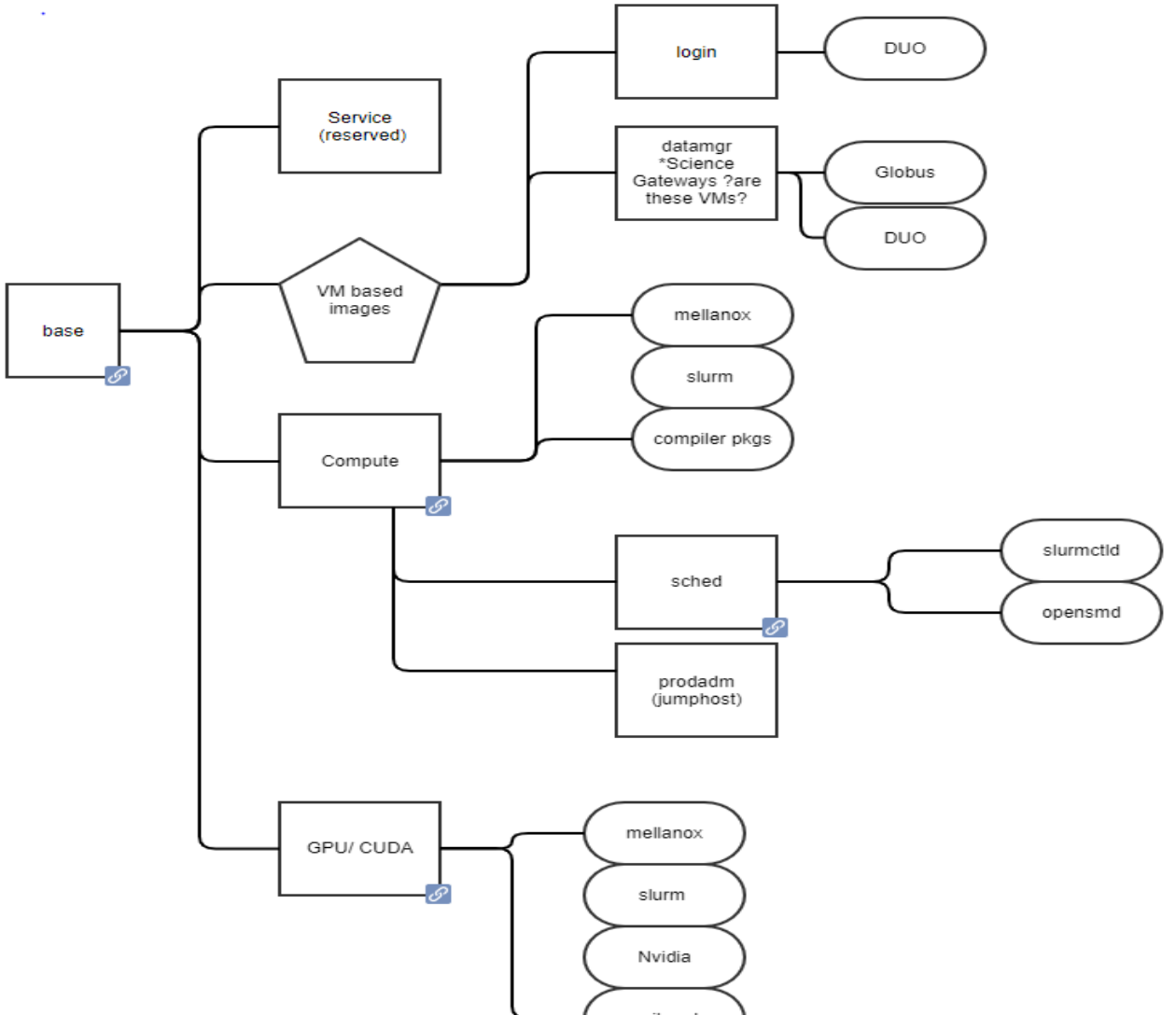
Node images are created using a hierarchy of profiles.

Each profile inherits from the base profile.

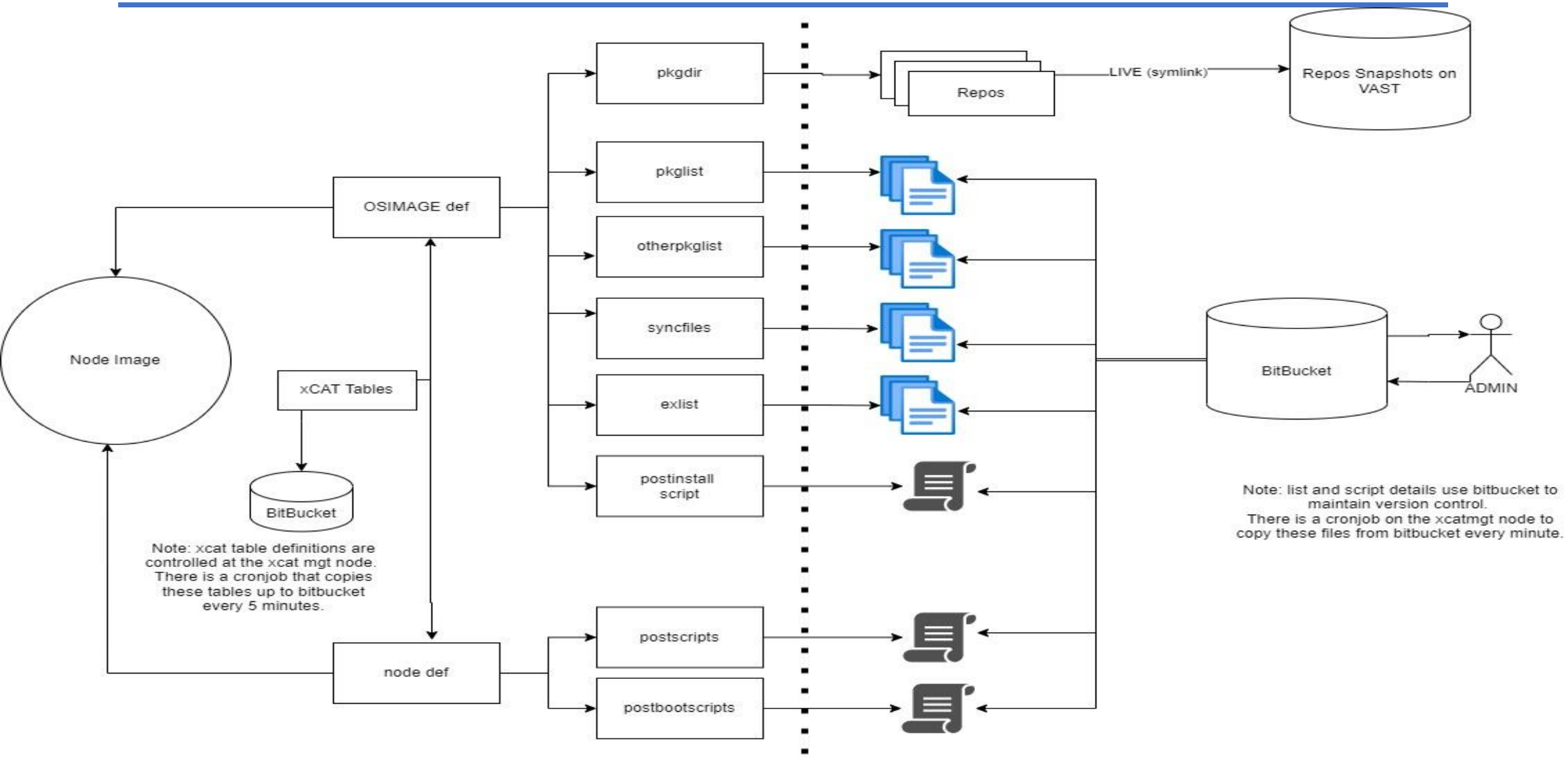
Each profile has an entry in the osimage table.

Image customizations are managed via package list, sync lists, and postscripts.

Software components are managed using a modular package list and postscripts to ensure consistent and compartmentalized management.



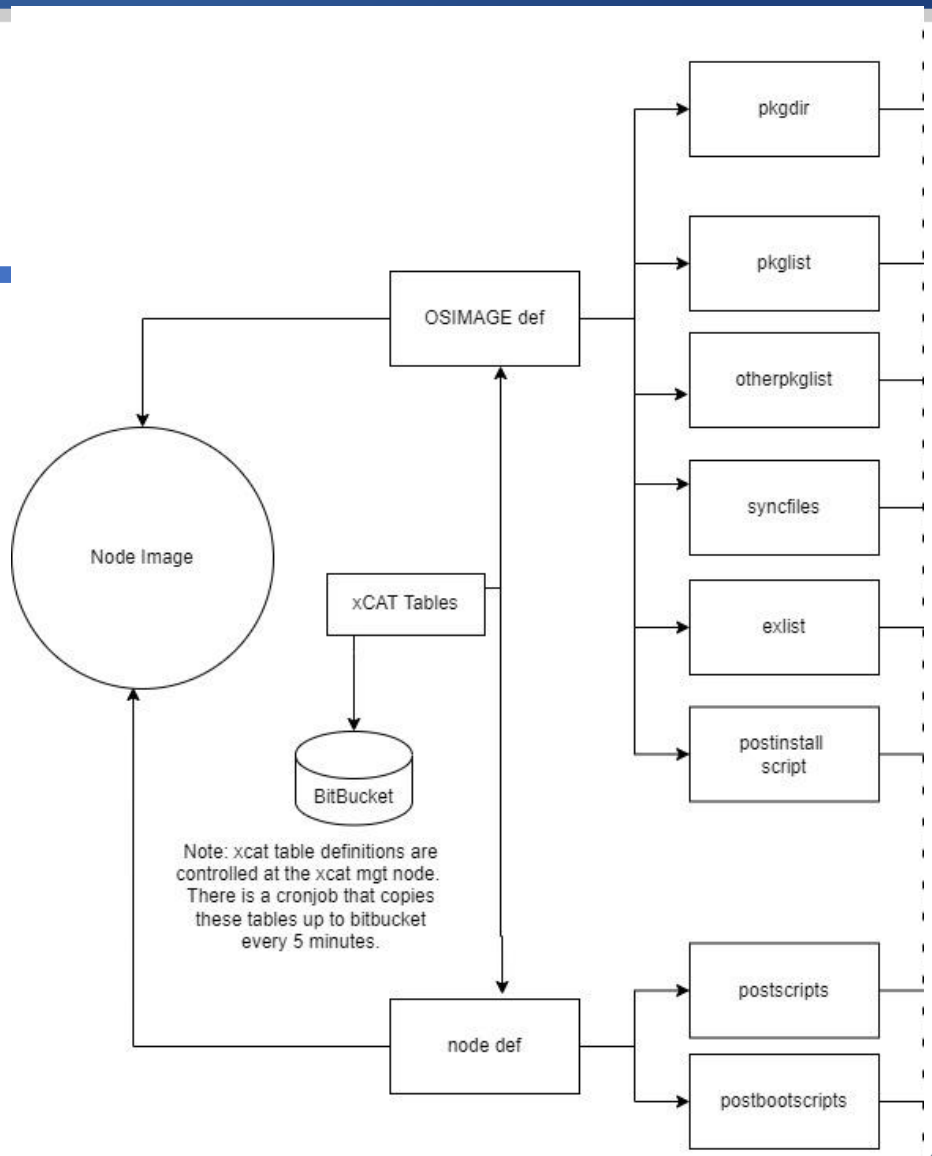
Solution Overview



Implementation : xCAT parameters

- xCAT Objects: Nodes (servers), groups

- xCAT Database – stores data and config details for objects in csv tables
  - osimage
  - package list
  - sync list
  - exlist
  - scripts (postscripts,postbootscripts)



Implementation : xCAT osimage definition

[root ~]# lsdef -t osimage rhels83-compute

Object name: rhels83-compute

exlist=/install/custom/netboot/e2-RHEL-8.3.0-RC1/base.exlist

osdistroname=rhels83

otherpkglist=/install/custom/netboot/e2-RHEL-8.3.0-RC1/compute.pkglist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/mlx.pkglist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/swst-build.pkglist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/ood.pkglist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/gpfs.pkglist

pkgdir=http://10.6.80.11:80/repos/iDRAC/,http://10.6.80.11:80/repos/MLNX/,http://10.6.80.11:80/repos/rhel83-baseos/,http://10.6.80.11:80/repos/rhel83-appstream/,http://10.6.80.11:80/repos/codeready-builder-rhel-8/,http://10.6.80.11:80/repos/icds_stuff/,http://10.6.80.11:80/repos/epel/,http://10.6.80.11:80/repos/cuda/,http://10.6.80.11:80/repos/gpfs_rpms/

pkglist=/install/custom/netboot/e2-RHEL-8.3.0-RC1/base.pkglist

postinstall=/install/custom/netboot/e2-RHEL-8.3.0-RC1/base.postinstall,/install/custom/netboot/e2-RHEL-8.3.0-RC1/compute.postinstall

profile=compute

provmethod=netboot

rootimgdir=/install/netboot/rhels8.3.0/x86_64/compute

synclists=/install/custom/netboot/e2-RHEL-8.3.0-RC1/base.synclist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/compute.synclist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/gpfs.synclist,/install/custom/netboot/e2-RHEL-8.3.0-RC1/ood.synclist

Implementation : xCAT list and script modularity

- Use symlinks to build from modules to reuse functions
- Add numbers to name to control execution order

- Certain actions do not complete correctly before packimage on stateless images
- This limitation can be over come with the use of postscripts

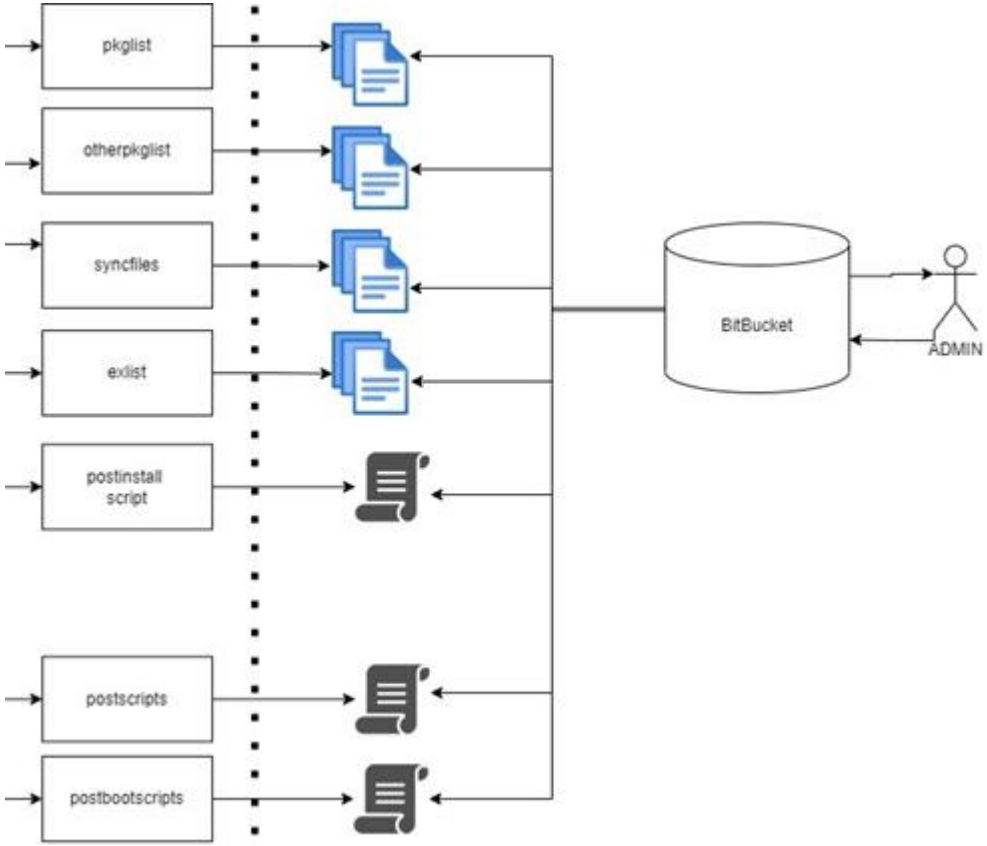
```
[root@          compute]# cat ./available/set_NY_timezone
#!/bin/bash
## post script to set time zone for every one.

rm -f /etc/localtime
ln -s /usr/share/zoneinfo/America/New_York /etc/localtime
```

```
[root@          postscripts]# tree
.
├── aux
├── available
│   ├── collectl_startup
│   ├── config_compute_sshd
│   ├── config_dualhome_routes
│   ├── config_duo_sshd
│   ├── config_globus_setup
│   ├── config_ib_fstab
│   ├── config_ib_mountpoints
│   ├── config_nessus
│   ├── config_opensm
│   ├── config_scheduler
│   └── turn_off_smt
├── base
│   ├── config_systype -> ../available/config_systype
│   ├── rpm_set_filecaps.sh -> ../available/rpm_set_filecaps.sh
│   ├── run_config_network -> ../available/run_config_network
│   └── set_NY_timezone -> ../available/set_NY_timezone
└── compute
    ├── 00_partition_local_drive -> ../available/partition_local_drive
    ├── 01_config_slurm -> ../available/config_slurm
    ├── 02_config_ib_mountpoints -> ../available/config_ib_mountpoints
    ├── 98_run_sdr_restore_ib -> ../available/run_sdr_restore_ib
    ├── 98_set_perms_ood -> ../available/set_perms_ood
    └── 99_config_compute_sshd -> ../available/config_compute_sshd
```

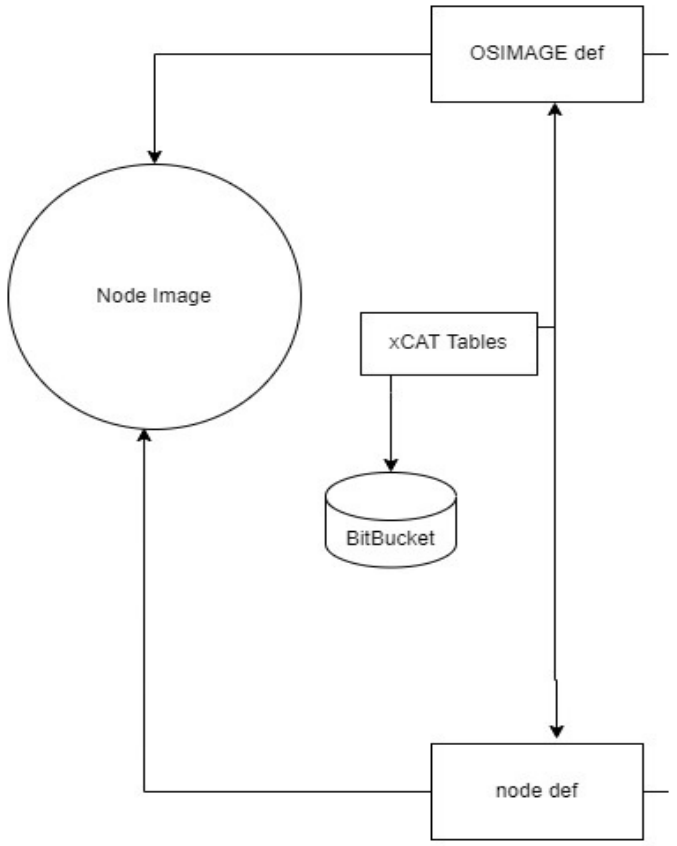
Implementation : Git Manager

- Admins make changes to list and scripts definitions via git commit
- Every minute, the Git repo syncs over the files on the xCAT management node.
- An admin could make changes on the management node, but those changes are quickly reverted (lost).
- Commit approvals ensure changes are not enacted without understanding impact to users.



Implementation : Backup and Recovery

- When using stateless compute images, xCAT Management node can be down if compute nodes are not restarted
- To avoid lost of configuration, we backup the complete xCAT database to BitBucket using dumpxCATdb on a 5-minute cron.
- This also allows us to rebuild the xCAT management node and restore the database.
- Recovery time from dead management node is 2 hours.
- If no compute is rebooted in that time, there is no client outage.
- Note: this does require a 2nd DNS source



Other Considerations

- Currently, most Implementation and Operations Admins have been trained on the process to enable changes
- Developers can work asynchronously using custom definitions
- Dev/Test is node agnostic – node boot definitions can be changed as needed
- Previously this required developers to work on dedicated nodes, then request changes be propagated out to all nodes by an Operations Admin.

- Maintaining a test cluster vs production cluster will lead to some "hard-coded" aspects that don't allow for seamlessly promotion.
- Monitoring: not provided for free, requires more configuration
  - i.e. PRTG checks for kernel files to alert admins of failed image builds
  - Slurm Node Health Check, Nagios

- Resources: xcat.org, gitlab.pnnl.gov, icds.psu.edu