Managing the SMW as a git Branch

Douglas M. Jacobsen, NERSC
Randy Kleinman, Cray
Harold Longley, Cray

Cray User Group 2018

Motivation

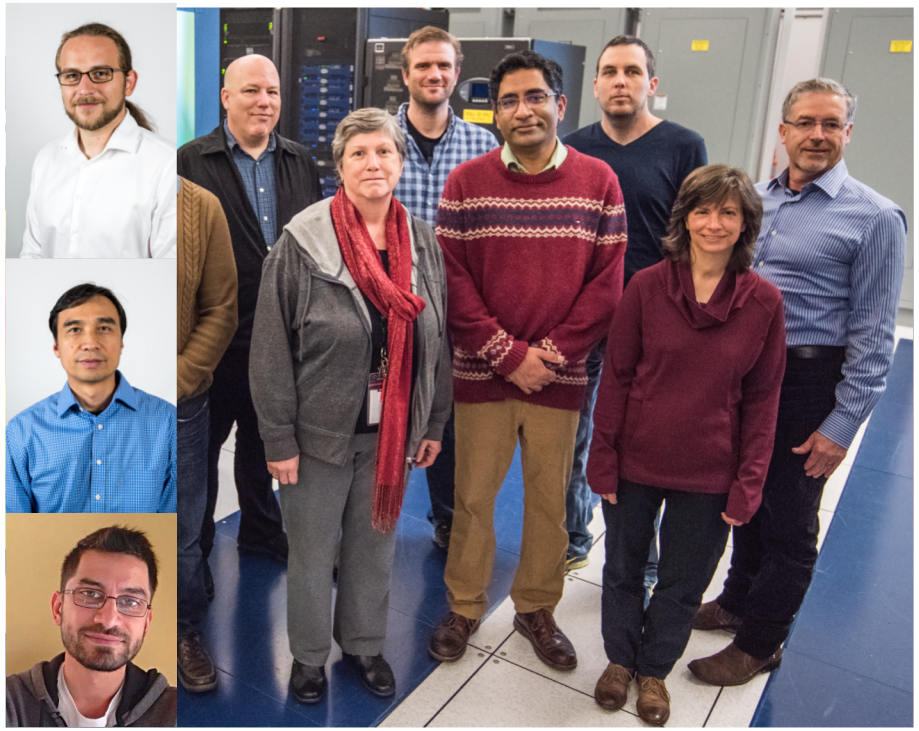
Use modern software engineering approaches and tools to increase effectiveness and efficiency of systems management by:

- Providing a mechanism for NERSC and Cray engineers to collaborate
- Centralize and unify the configuration of multiple systems
- Develop all configurations on test systems and reliably deploy to production
- Use test systems for speculative development and production modeling
- Recover SMW efficiently

This enables a DevOps view and workflow for Cray Supercomputers.

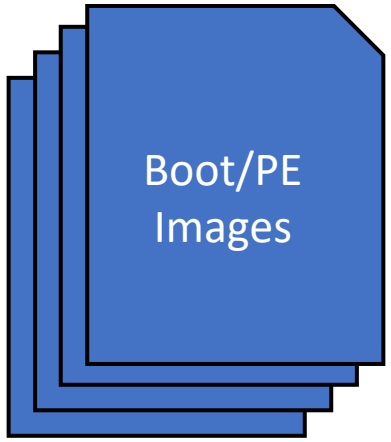
Collaboration on Systems Development

- Operating and Configuring systems is a collaborative effort between
  - NERSC Computational Systems Group
  - Cray On-site support team
  - Other NERSC-staff contributors (Operations, Data Services, Storage, Networking, Security)



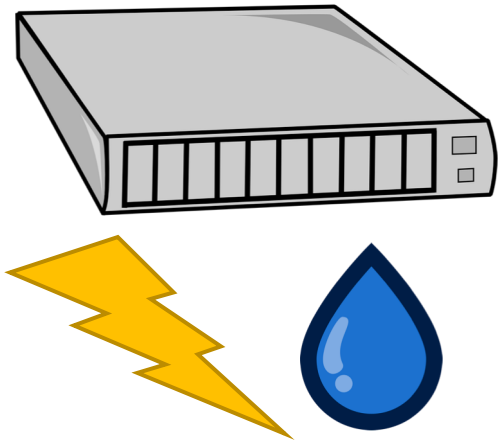
Many contributors!

Simplified CLE Workflow



+

+



=

Booted System!

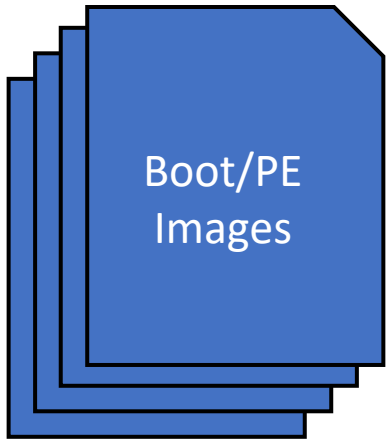
- Image Recipes
- Package Collections
- RPM Repos

- Worksheets
- Custom Ansible Plays/Roles
- Custom Ansible Variables
- Files positioned by SimpleSync

- HSS Configurations
- SMW Configurations
- SMW Software
- Functional Cray XC Hardware
- Power/Water

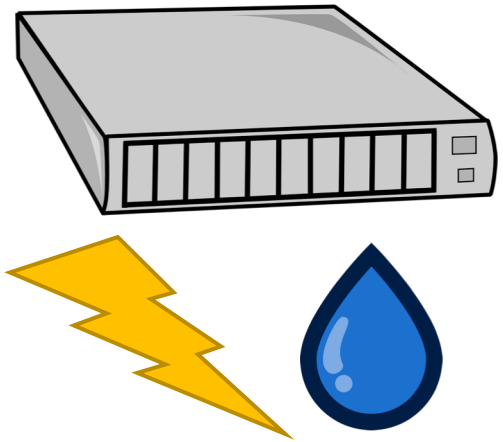
Iterate on software inputs incrementally over life of the system.

Managed Configurations



+

+



=

Booted System!

- Image Recipes
- Package Collections
- RPM Repos

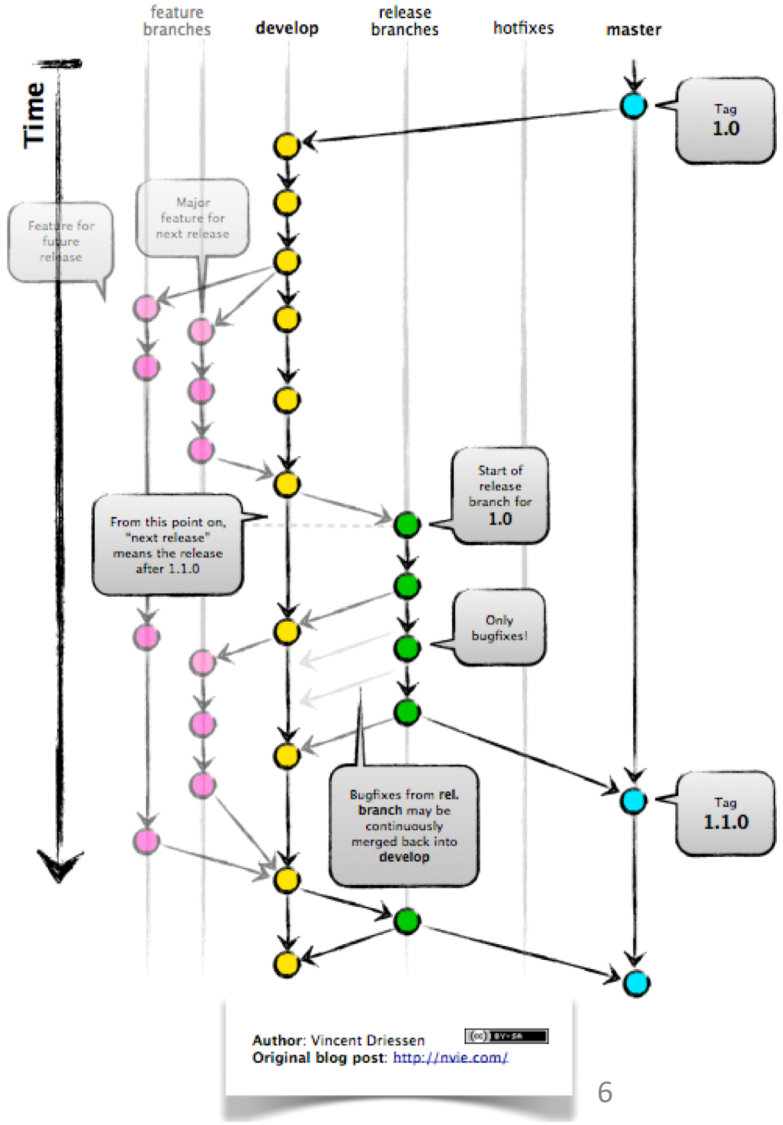
- Worksheets
- Custom Ansible Plays/Roles
- Custom Ansible Variables
- Files positioned by SimpleSync

- HSS Configurations
- SMW Configurations
- SMW Software
- Functional Cray XC Hardware
- Power/Water

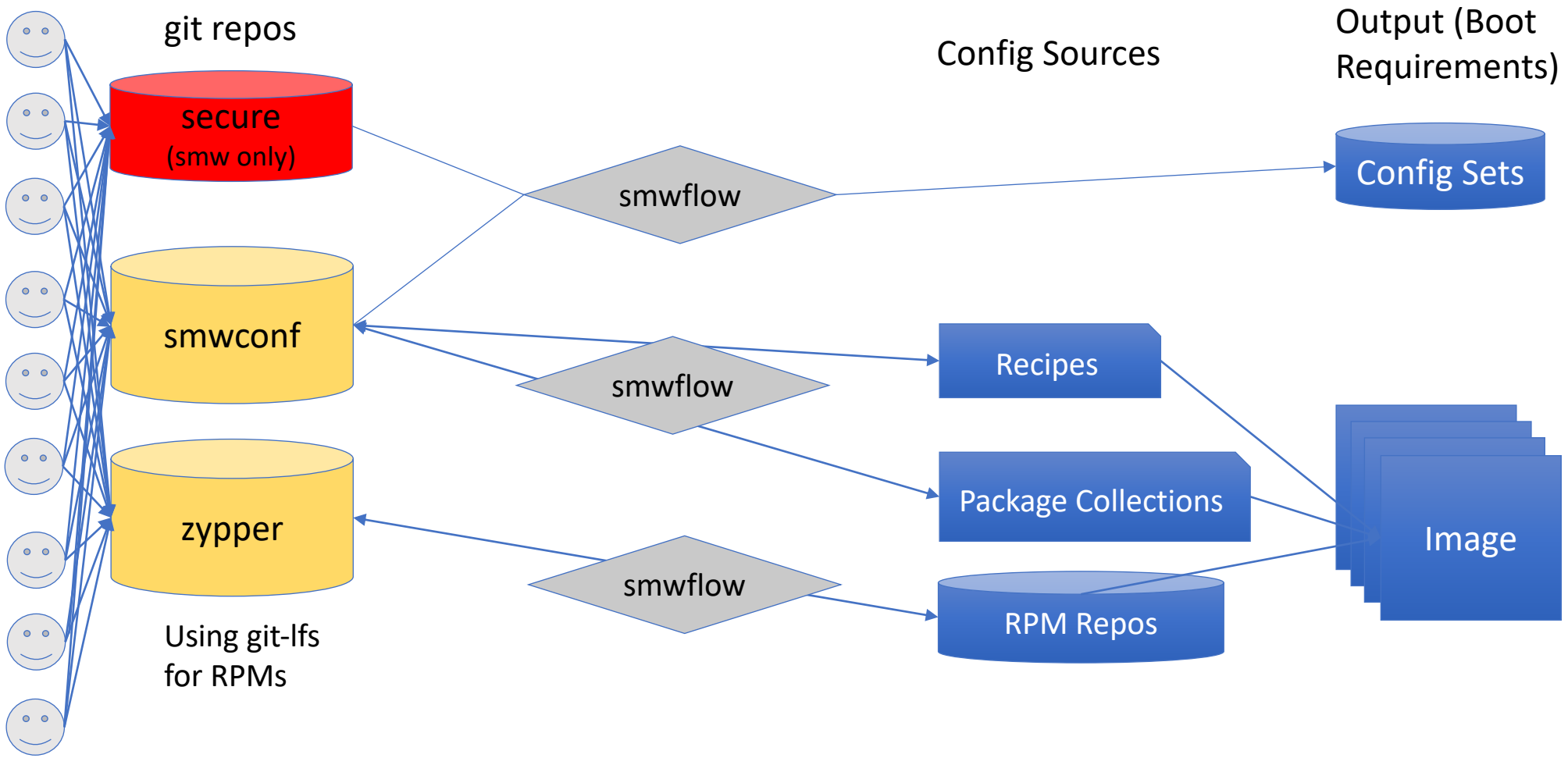
Iterate on software inputs incrementally over life of the system.

git as a System Management Solution

- In CLE6 most site-customizable are just files
  - Can be stored in SCM
- git branching is ideal for feature development
- Gitflow development style provides natural means to promote development branches to production
- On-site Atlassian Bitbucket/Stash provides change control capabilities
  - Branch level authorizations
  - Read-only keyed access (for smw root accounts)
- Distributed nature of git provides loss and failure resilience



Putting SMW Config in git



smwflow: Software to Control SMW Config

- Subdivides SMW configurations into different object classes:
  - imps: image recipes, package collections, IMPS configurations
    - Also config sets, with sub-classes of “global” and “cle”, and sub-sub-classes for all the various components of the config sets
  - zypper: RPM repositories stored in specialized git repository using git-lfs (Large File Support)
  - hss: Hardware Supervisory System configuration files
  - smw: Base configurations of the SMW itself
- The data (and metadata) of the configuration objects are stored in git
- Within each object class an evaluation hierarchy is supported to allow multiple systems or platforms to share configurations
- Templating of configurations is supported to allow the same base configuration to support multiple systems

smwflow: Operations and Plugins

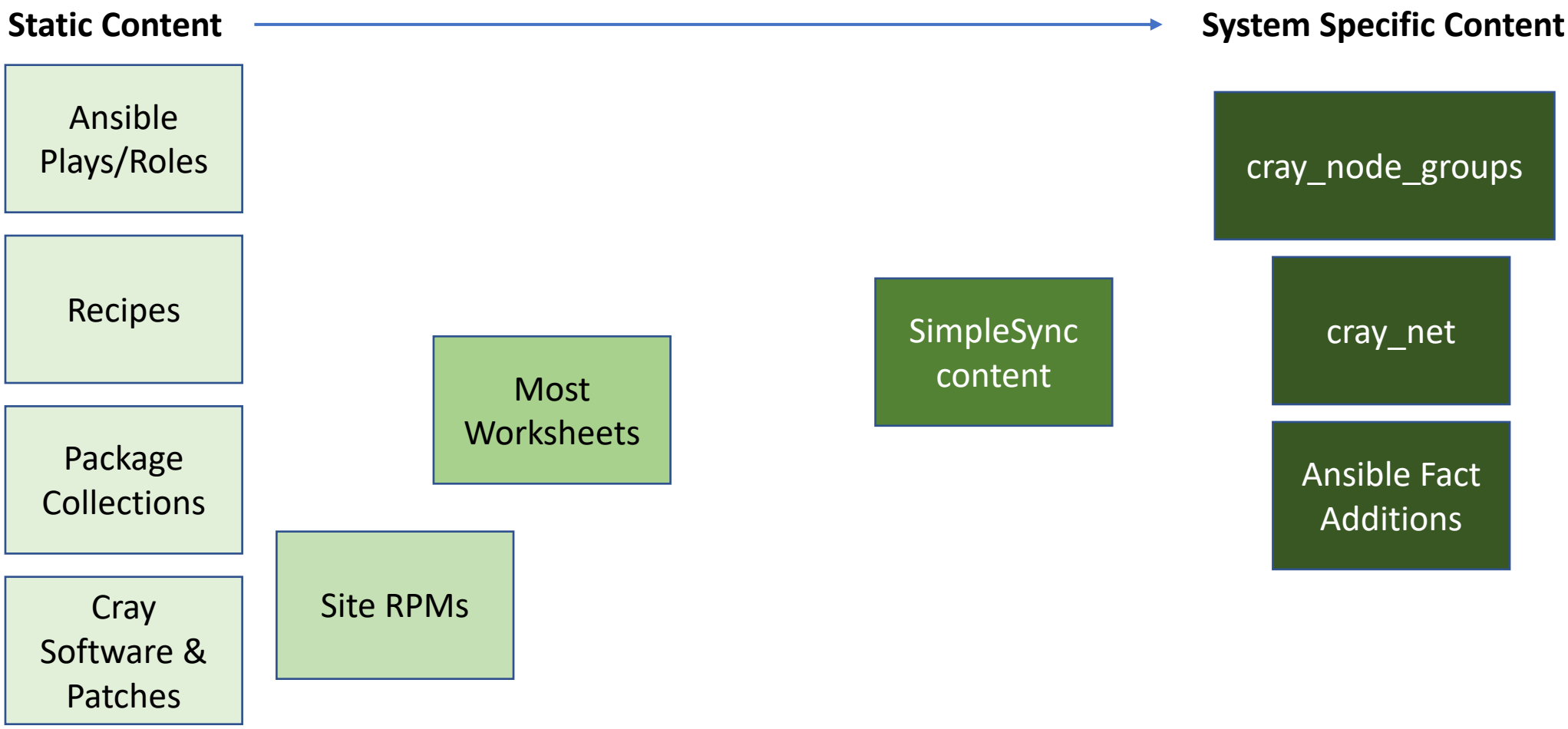
smwflow Operations

import: copy content from SMW to git (fairly stupid)
create: create new config set or map
verify: check that SMW matches git
update: replace SMW configurations from git
objdiff: generate diff of two different git-based files to support merging and templating

smwflow Plugins

- Allow arbitrary code to participate in building custom configurations.
- Enables multi-system configurations (e.g., hostbased ssh auth across systems), or pre-generation of configuration data in SimpleSync

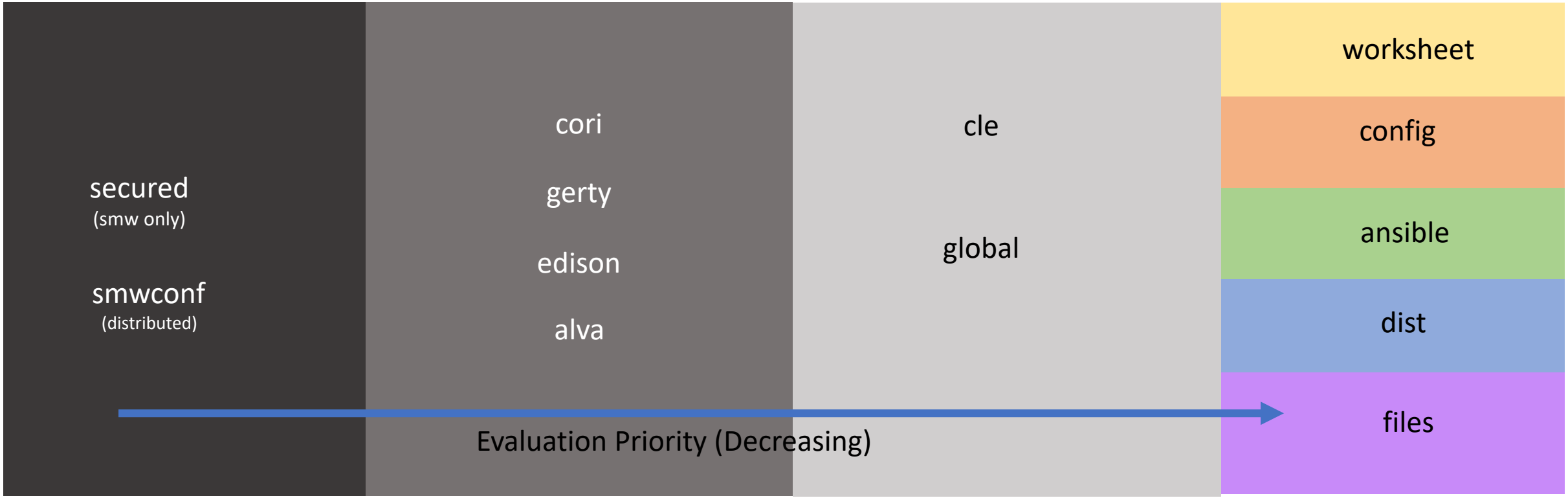
Git for Multi-system Management



Config Set in git: Multi-system

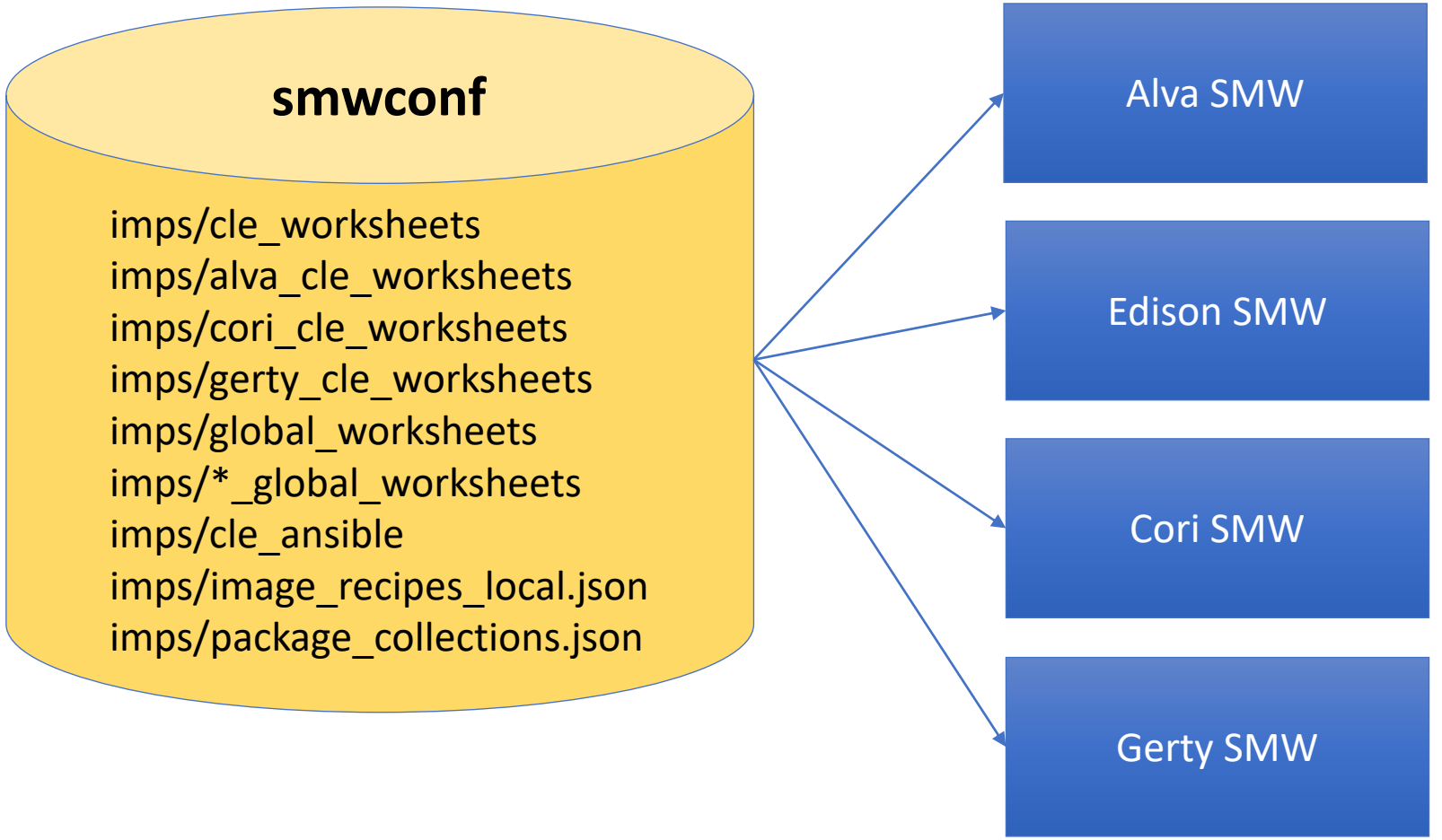
- Hierarchical Assembly from multiple sources

```
<git repo>/imps/(<system>_)?<cfgsetType>_<objectType>
```



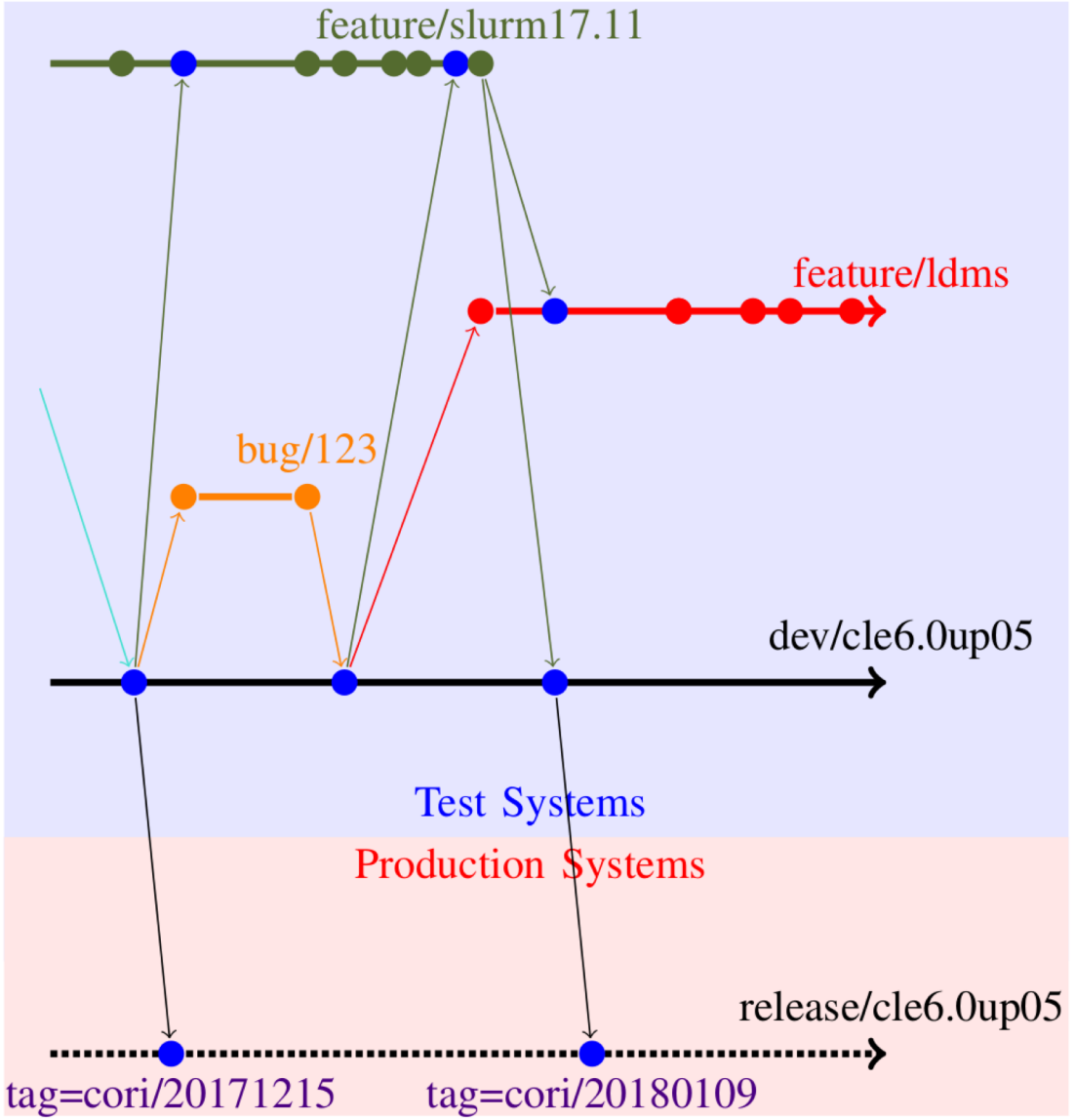
git for Multi-system Management

Most worksheets are shared
All ansible is shared, system specific variables
Recipes and package collections identical across systems



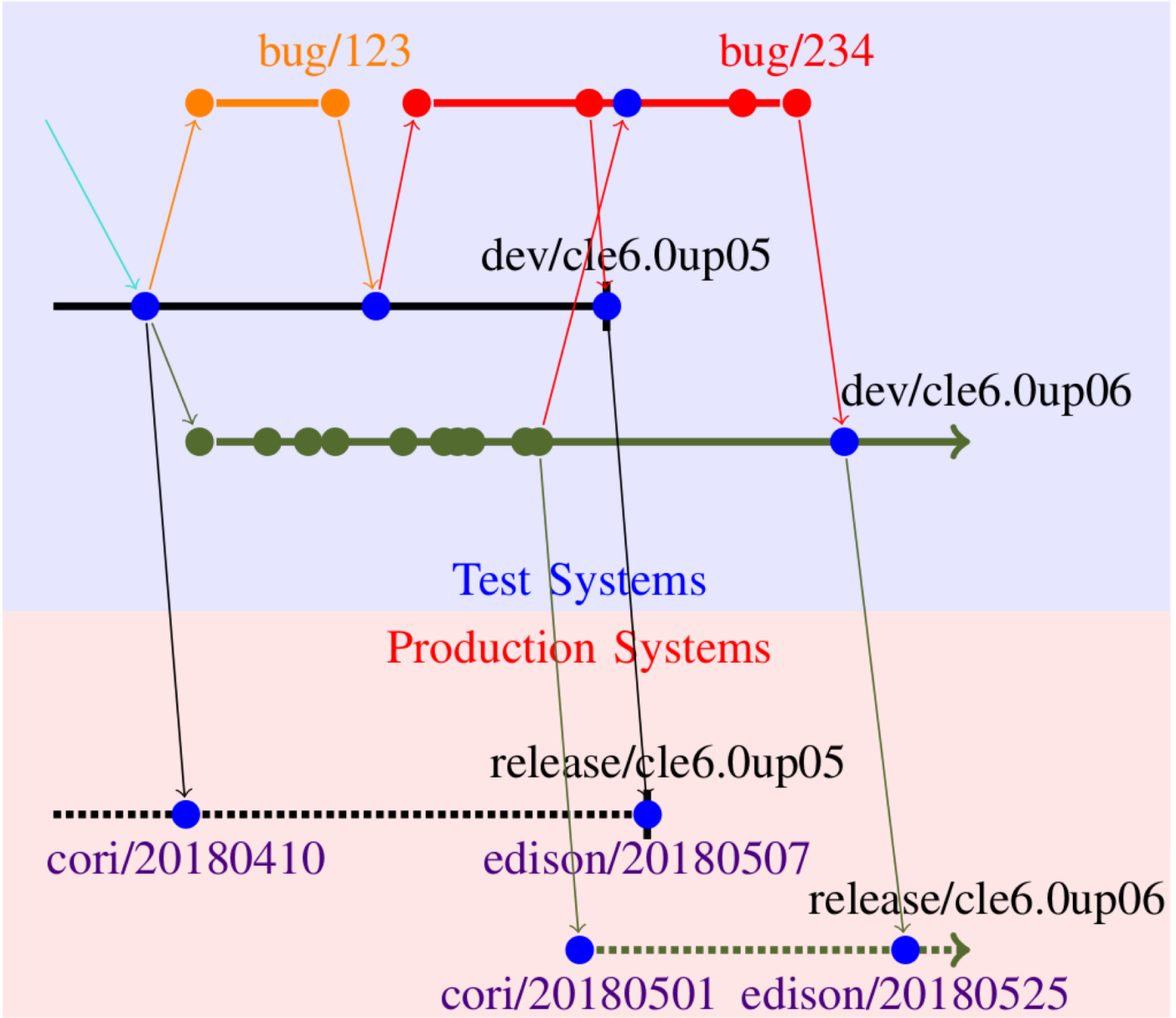
Development to Production

- Did you know?
  - Your TDS is the most exciting system on the floor
  - Your production system is a totally boring endpoint
- Recipes static, must test recipe build through operations on TDS
- TDS iteration → regression testing → iterate on TDS → production



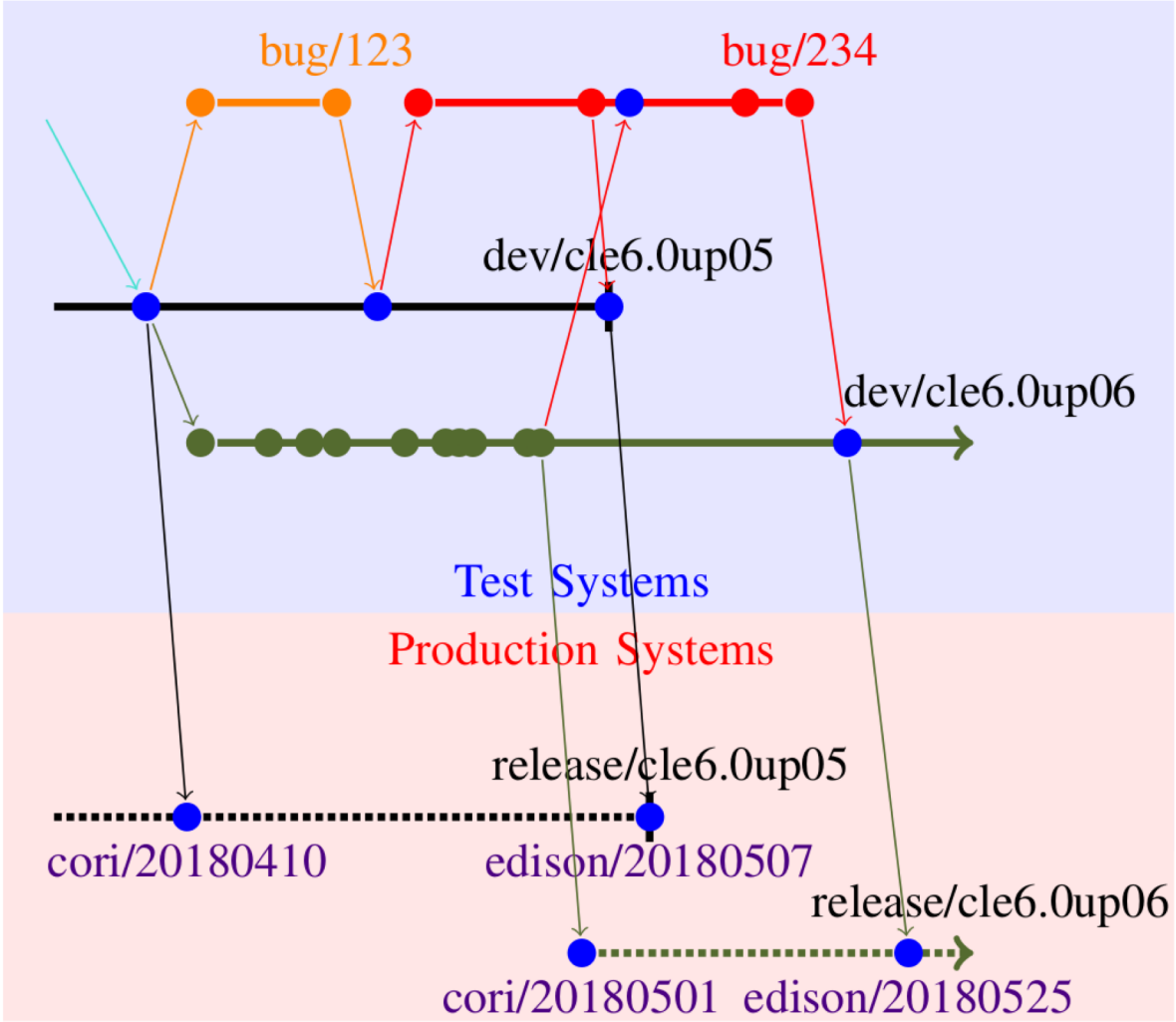
CLE Update Branching

- Track multiple primary development branches
  - One per CLE update level
- Allows version skew between systems
- Allows forward propagation of legacy configurations
- Enables coherent merging of bug/feature branches into multiple devel/release branches through time



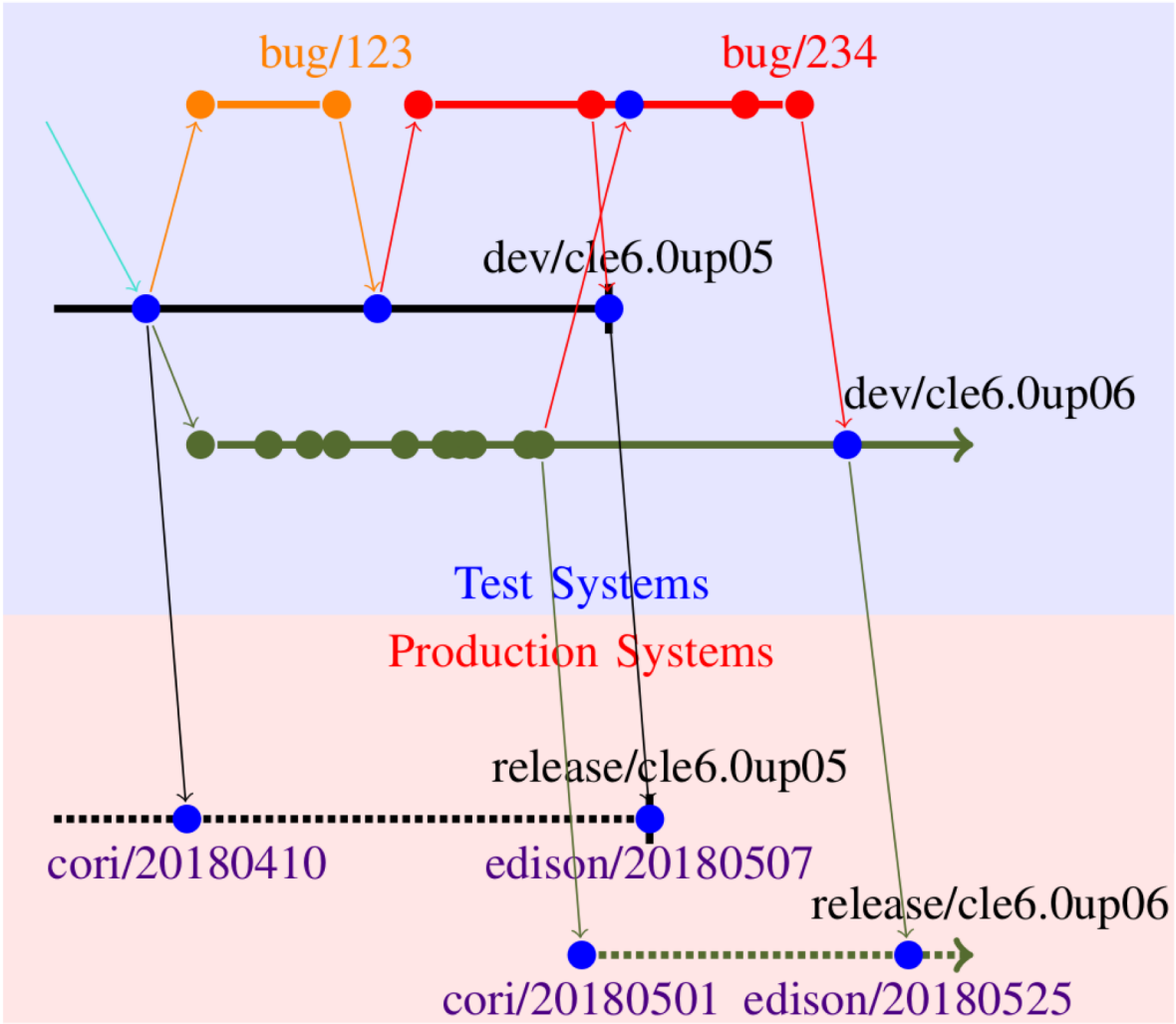
Results: Patch/Minor Reconfig Maintenance

- Average patch maintenance + reconfiguration: **4 hours**
  - 30 minutes shutdown
  - 30 minutes reconfiguration (patches preinstalled and images prebuilt)
  - 8 minutes: automated config set construction
  - 30 minutes bouncing
  - 1.5 hours booting



Results: Fresh Install of CLE6.0up05

- 6.0UP05 Fresh Install: **12 hours** (pipelined)
  - Preinstall SLES 12sp3, prebuild PE and stage
  - 1 hour bootraid bootstrapping
  - 2 hours SMWInstall
  - 1 hour patching (overlap xtdiscover)
  - 6 hours xtdiscover/xtzap/xtbounce
  - 1 hour imagebuild/config set construction (overlap xtzap/xtbounce)
  - 2.5 hours booting



Future Work

- Adding more configurations
- System Management Automation and Continuous Integration
- Secure Commits with GPG
- Automated Configuration Pre-Computation
- (NERSC) Enabling some Rolling Updates
- Boot Performance Tracking / Overall git bisection for any given metric
- (Cray) Improving Cray SMW tools to better integrate with git

Conclusions

- Git branching provides a robust method for:
  - Tracking and managing feature and bug development processes
  - Managing multiple versions of CLE through time
  - Social coding and peer review, collaboration

Conclusion

Cray CMF	+	SMW Configuration Flexibility	+	smwflow (process & software)	+	git branching

Enables:

- Process flexibility in systems management
- Software **process** in systems management
- Multi-system configuration (not required)
- Higher engineer productivity
- Increased system availability and resilience

Getting Started:

https://github.com/nersc/smwflow

PS: still awaiting U.C. and DOE approval for open source
PPS: it'll be available soon.

Questions?

Come to XC System Management Usability BOF
(Event-Large)