Centralized Fabric Management Using Puppet, Git & GLPI

Jason A. Smith
Brookhaven National Lab

Motivation

- Several years ago most admin work was either done manually or using home-made scripts
  - ssh in a loop from a management gateway
  - time-consuming, reactive, fire-fighting work
  - little sharing of work & backup expertise
- Standardize and unify our sysadmin work
- Self documenting build and config system
- Audit trail for complete change management
- Separate dev/test/prod env (little extra work)

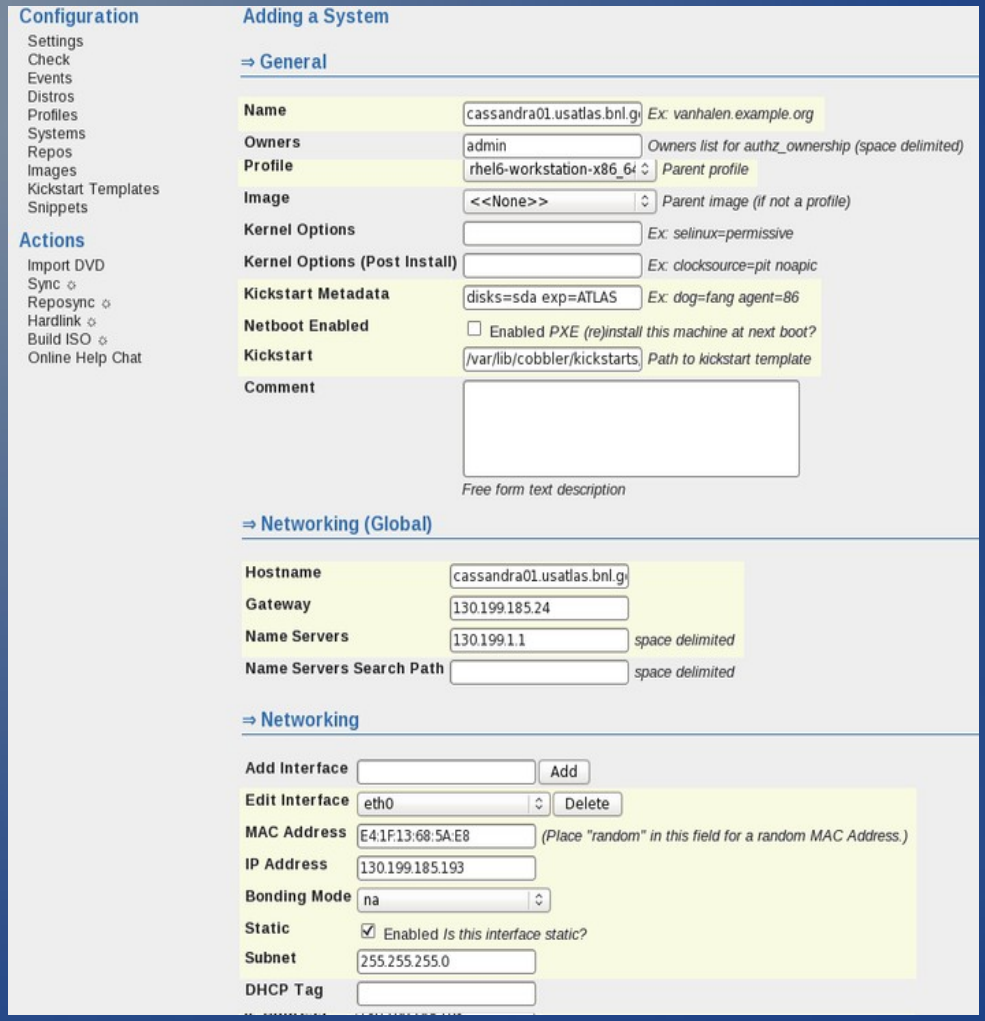
Components

- Cobbler/RHEV – New system provisioning
- Puppet – Centralized config management
  - Complete service config after provisioning
  - Dashboard monitoring & change auditing
- Git – Puppet catalog repository
  - Distributed development & historical record
- GLPI – Asset mgmt. & node classification
  - Fusioninventory-agent: auto asset inventory
  - ENC uses GLPI, custom DB & dashboard

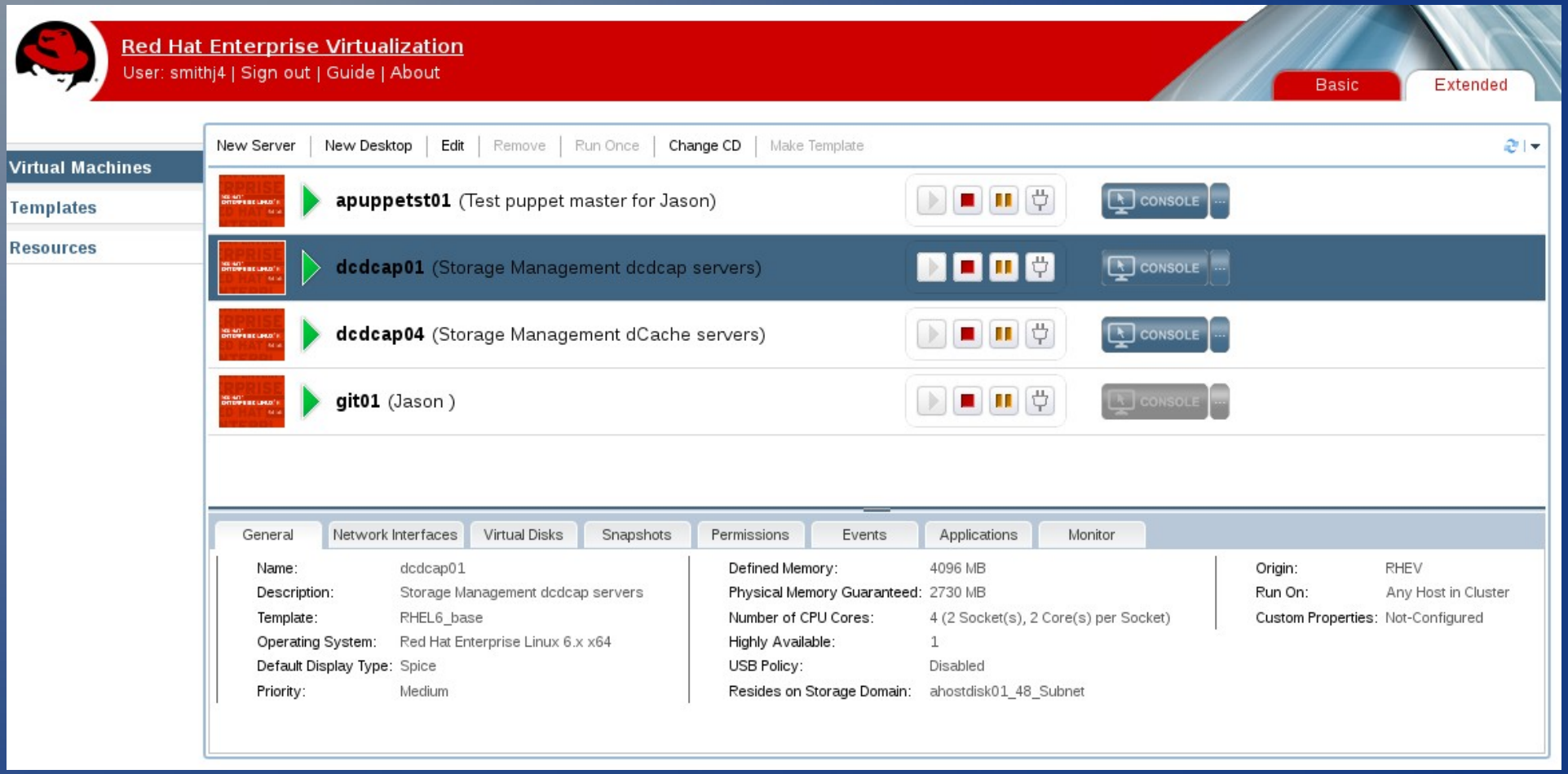
Provisioning

- Cobbler for hardware installs (poster #539):
  - Powerful Cheetah templating language and config/code reuse with “Snippets”
  - Single ks template used for most systems
  - Specify OS version & arch, network (MAC, IP, etc) & template metadata to install base OS, including fusioninventory-agent & puppet
- RHEV 3.0 for virtual machines:
  - Single template image used for new systems
  - 10 node cluster with 4TB of shared fiber storage

Cobbler Screenshot



RHEV Screenshot



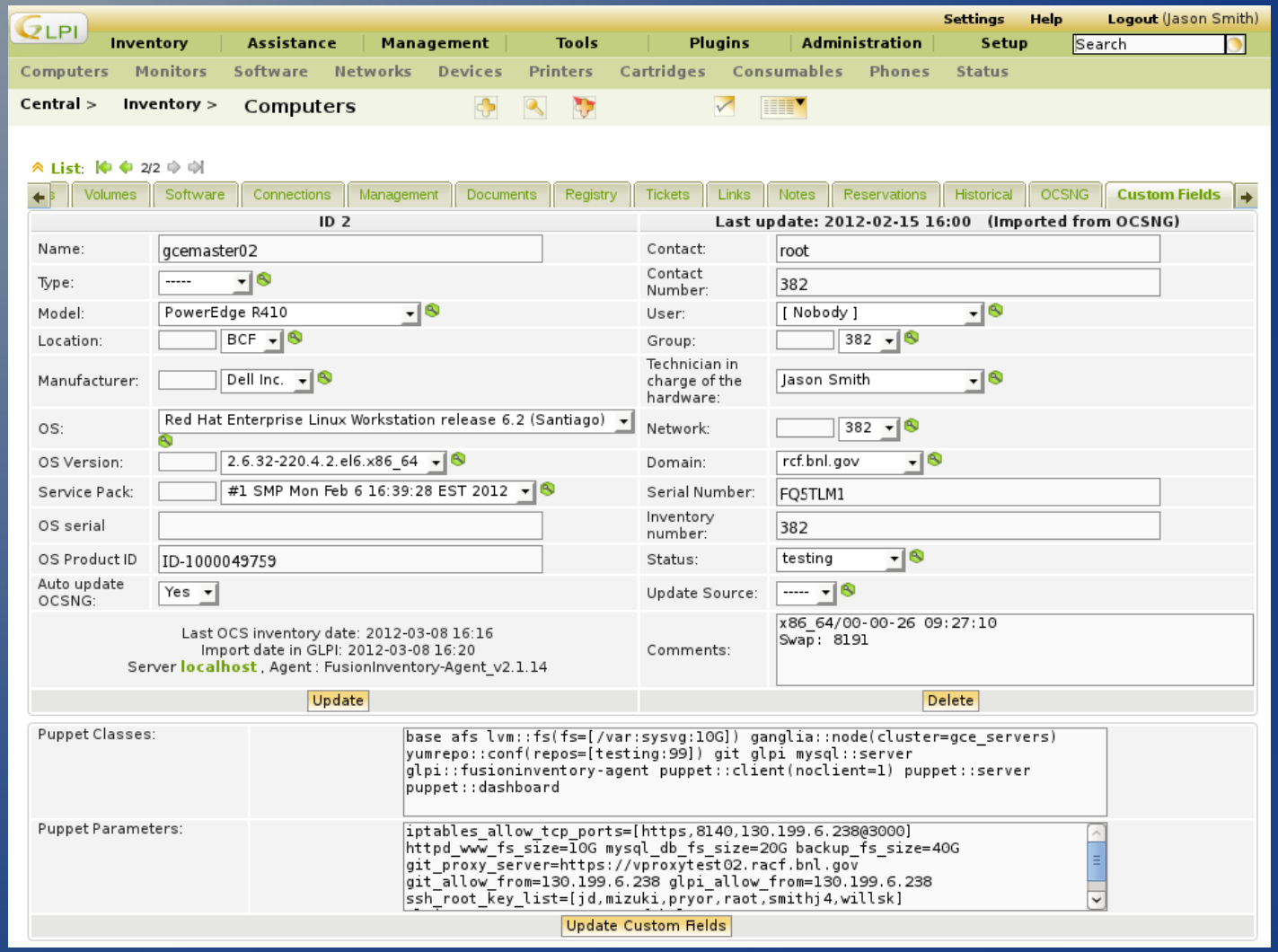
Why Git?

- Distributed version control system
- Faster, completely localized project copies
  - Commits and other work can be done offline
  - Local copy contains complete history
- Reduced single point of repository failure
  - Git can merge changes between many “servers”
- Simple, fast & clean branching (and merging)
  - Branches easily merged with other branches
  - All changes can be treated as branches

Why Puppet?

- Cfengine, puppet, chef, etch, bcfg2, AutomateIt
- Puppet was selected for several reasons:
  - Simple yet powerful DSL (Domain-Specific Lang) & RAL (Resource Abstraction Layer)
  - Explicitly declared dependency graphing model
    - Provides better deterministic state convergence
  - Central config catalog & dependency resolution
    - Better security, conflict resolution & logic analysis
  - Web dashboard, GraphViz config visualization
  - Long history, stable codebase, large user base

GLPI Node Classification



Puppet Environments

- 3 puppet environments linked to git branches:
  - Development: extensive module changes
  - Testing: small changes and wider testing
    - Changes staged for more manual tests by wider audience before merging into production
  - Production: main server management
    - Changes must be approved before they are merged into the production branch/environment
- Git branches are automatically sync'ed to puppet environments by push hooks.
  - Also verifies puppet syntax and other checks

Production Approval

Git/Puppet updates to production that are pending approval.

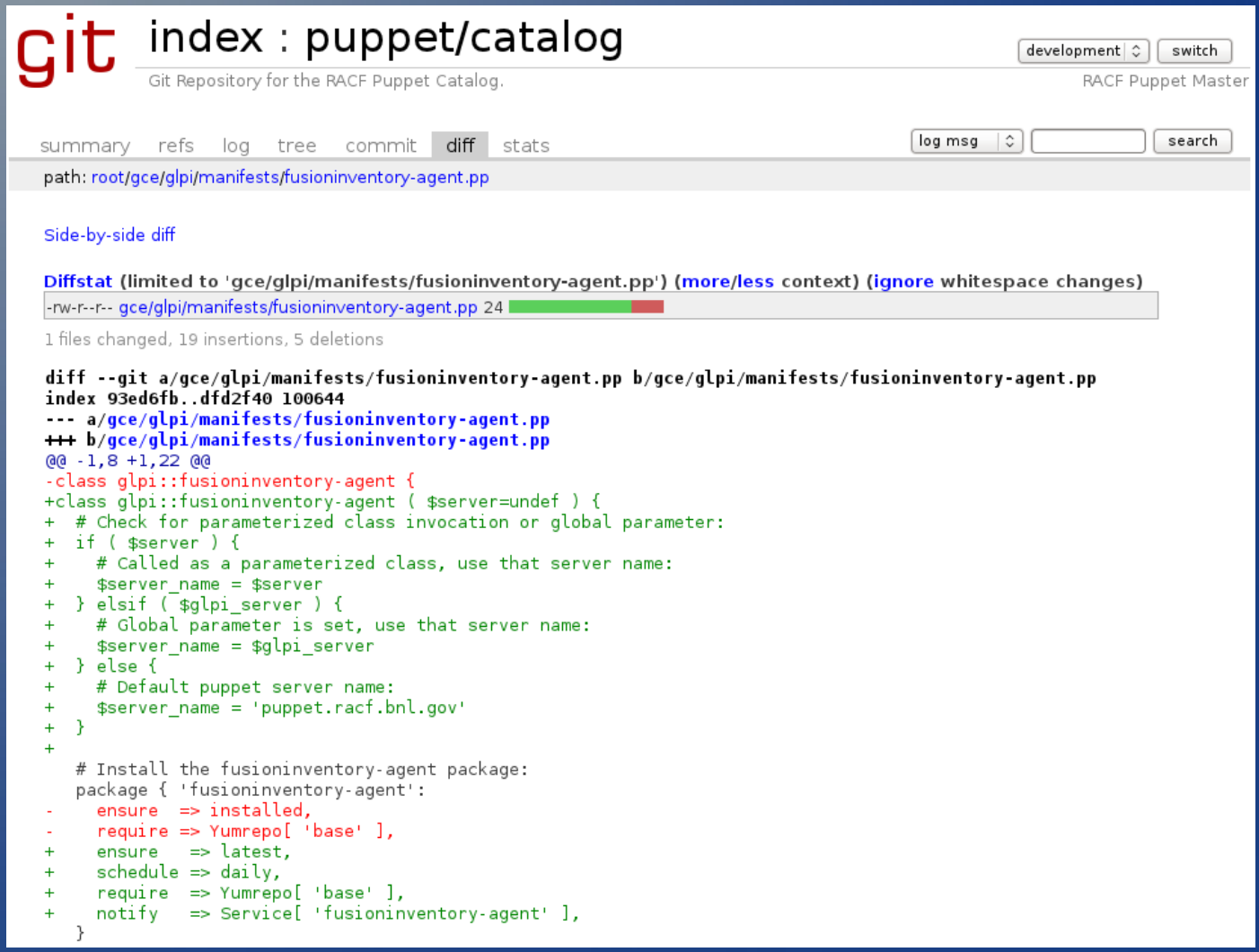
Hello Jason A. Smith, there are currently 2 changes waiting for approval:

Date	Age	User	Changes	Changelog	Approve	Reject
Fri Mar 9 15:43:12 2012	2 days	Zhenping Liu	diff	pending-zhliu-cb36590-20120309T204312UTC	merge	delete
Mon Mar 12 10:18:27 2012	1 minute	Jason A. Smith	diff	pending-smithj4-cb36590-20120312T141827UTC	merge	delete

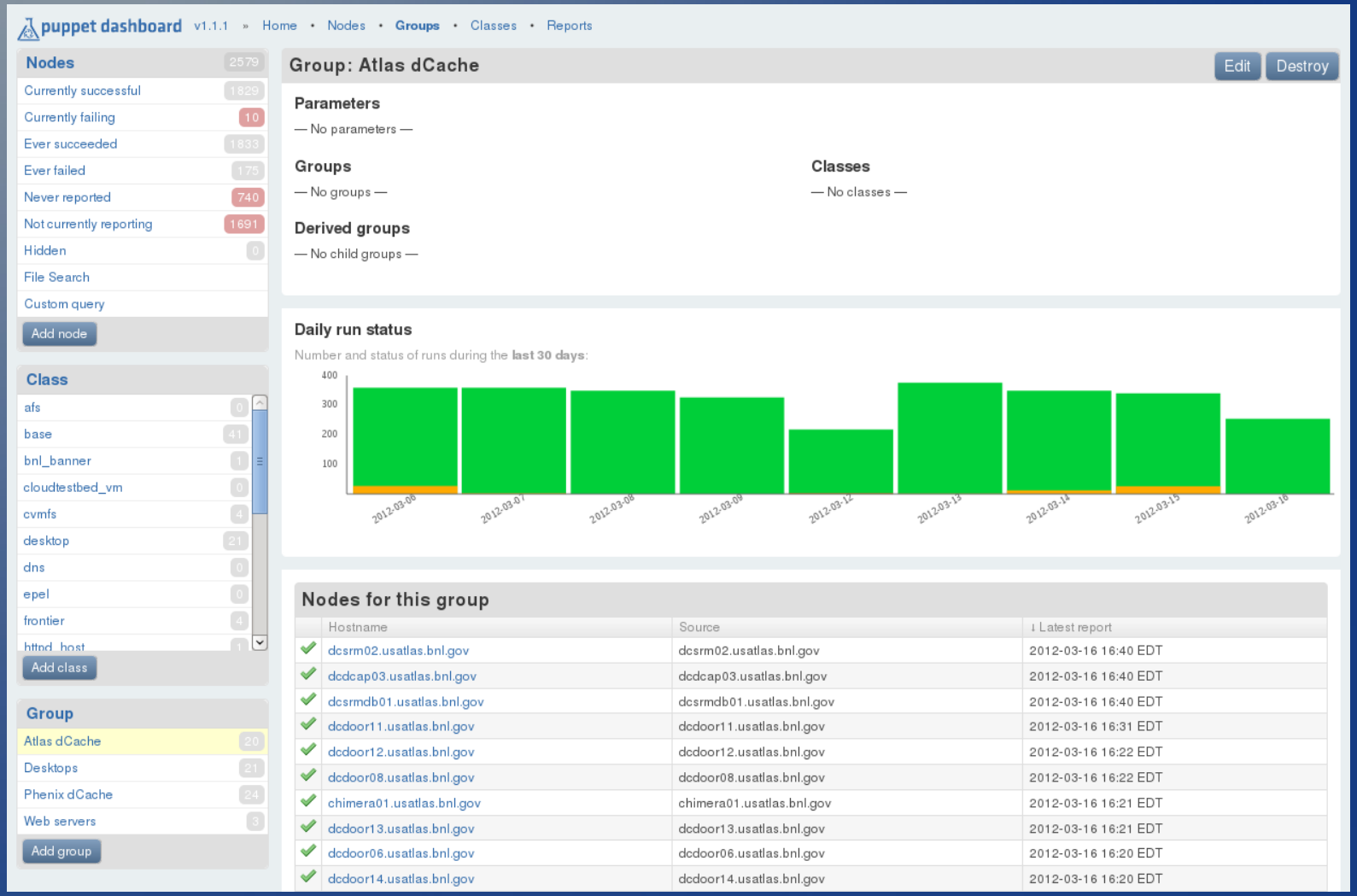
Instructions:

- The table above lists all changes to puppet's production environment that are currently pending approval.
- The diff link in the Changes column uses the cgit interface to display the detailed changes to all files contained in that pending update.
- The branch link in the Changelog column uses the cgit interface to display the commit history of that branch since it diverged from production.
- Use the merge link in the Approve column to accept the changes and merge them into production.
- Use the delete link in the Reject column to delete the branch if you do not want it merged into production.
- Email notifications are sent after confirmation of the chosen action.
- A side effect of this approval process is that you might see a lot of these old temporary pending branches accumulate in your locally cloned repo. You can clean these up by using the "git remote prune origin" command.

Cgit Diff View



Puppet Dashboard



Puppet Config & Scalability

- Still using 2.6.16 on RHEL5 with ruby 1.8.5
  - testing upgrade to 2.7 on RHEL6 with ruby 1.8.7
- Apache with Phusion Passenger (mod_rails)
- Queue daemon with activemq for fast DB updates of storeconfigs
- Over 2k agents currently using puppet
- Noticed MySQL errors with inventory service enabled at a rate >= about 1 client/second
- Tomcat/JRuby in future for improved scalability

Future Plans

- Change Management
  - Policy & procedures used to control changes made to production systems (ITIL, DevOps).
  - Changes made only during official windows.
  - Absolutely no unauthorized changes, no “cowboy” type behavior tolerated.
  - Use testbed environment to test changes before putting them into production.
    - Create replica of prod using VMs for auto-tests
  - Tools like Puppet, Git & GLPI can help make changes and keep a historical change record.

Automated Validation

- Add a new “validation” git branch & puppet environment
  - Contents: production with all changes currently pending approval automatically merged in
- Replica of all critical production services using RHEV VMs
- Automated testing of production and proposed changes using puppet agent runs and nagios monitoring of all Vms to validate that all production systems still work as expected

Why do it?

- Uncontrolled change can work sometimes, but often cause self inflicted problems and future firefighting episodes & upgrade nightmares.
- Stop duplicating work and effort, standardize.
- Stop making time consuming manual changes.
- Without it, servers become like snowflakes: they may all start out identical, but over time, config drift eventually makes each one unique.

Benefits

- Shift staff time from perpetual reactive firefighting mode, that often only addresses the symptoms, to more proactive work, that addresses the root causes of problems (fire prevention).
- Repeatable and standard build & config process means it is often faster and easier to rebuild problematic servers, rather than waste hours or days troubleshooting problems.

References

- Cobbler: http://cobbler.github.com
- RHEV: http://www.redhat.com/products/virtualization
- FusionInventory: http://fusioninventory.org
- GLPI: http://www.glpi-project.org
- Git: http://git-scm.com
- Puppet: http://puppetlabs.com
- Email: smithj4@bnl.gov