

Realm Linux Training: Configuration Management Overview

Jack Neely
linux@help.ncsu.edu

Campus Linux Services
Office of Information Technology
North Carolina State University

Thursday, June 30, 2011

Copyright © 2011 NC State University

Git Presentation Source:

Copyright © 2008 Karel Zak

Copyright © 2008 Tomas Janousek

Copyright © 2008 Florian Festi

Copyright © 2008 Bart Trojanowski

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Git Presentation Original Source: <http://kzak.fedorapeople.org/>

Source Code: <git://git.linux.ncsu.edu/bcfg2-training.git>

Please download this presentation and follow along.

<http://go.ncsu.edu/bcfg2-overview>



Table of Contents

- 1 Why Configuration Management?
- 2 Overview of Our Bcfg2 Architecture
- 3 Conclusions

Section Outline

1 Why Configuration Management?

- What is Realm Linux?
- Tools of the Realm Linux Past and Present
- The Realm Linux Future

A Distributed Environment

- Realm Linux – NCSU’s Linux “Kit”
- Realm Linux is Designed for a Distributed Environment
 - Distributed System Administration
 - Local System Administrator (SA) Groups Retain the Most Control
 - Building Blocks
- The Realm Linux Project Provides Extra Packages, Tools and Minimal Configuration¹
- Local SAs Use These Tools to Build Their Infrastructure
- Colleges Can Customize Realm Linux and Pass Customizations to Departments

¹http://www.linux.ncsu.edu/realm_linux/

A Configured Environment

- Realm Linux Includes Basic Configuration
 - /etc/krb5.conf
 - /etc/ldap.conf
- Local Configuration Deployed By Web-Kickstart²
- Web-Kickstart Gives Us the Following Benefits
 - Trusted SAs
 - Hooks into Root Password Management System
 - Custom Configuration and Scripts Provided by the SAs

²<http://web-kickstart.linux.ncsu.edu>

A Changing Environment

- Change is Unavoidable
- Change Managed by In-House Scripts and Applications
 - Apply Configuration Globally
 - Poor or Absent Concepts of Groups and Functions
 - Cases Where SAs Must Restore Original Configuration
 - Barrier to Entry for Realm Linux Development
 - Poor Configuration Enforcement
 - No Verification
 - Commonly Reinvented by Each SA Group
- Change in Realm Linux is a Kludge at Best

A Better Environment

- A Realm Linux Powered by a Configuration Management (CM) Tool
- A Default and Basic Configuration Specification for Realm Linux
- Change That is Version Controlled, Authenticated, Logged
- Handle Functions and Groups That Result in Different Configuration
- Support Distributed Systems Administration
 - Partitioned SA Groups Like Web-Kickstart
 - Not a Centralized System
 - Authorization and Protection of Sensitive Data
- Lower Barriers to Developing Realm Linux
- A Realm Linux for Other Distributions

Section Outline

- 1 Why Configuration Management?
 - What is Realm Linux?
 - **Tools of the Realm Linux Past and Present**
 - The Realm Linux Future

RealmConfig

- Rewritten 4 Times
 - 1999: Matthew Wilson
 - 2000: Nalin Dahyabhai
 - 2001: Jeremy Katz
 - 2003: Jack Neely
- Few People Understand its Workings
- Only Makes Global Changes on RPM Upgrade
- A Bear to Maintain
- Special Case of an RPM Config Package (Next Slide)

RPM Configuration Packages

- Run Through the Yum Update Mechanism with RPM Versioning
- Scripts, Config Files, Package Requirements
- Maintenance of RPM SPEC Files
- Automated One Off Configuration Virtually Not Possible
- Only Works Well When Targeted at a Specific Function
- Still Based on In-House Scripts and Tools

Web-Kickstart Hands Off Installs

- Very Successful at Managing Installation and Deploying Initial Configuration
 - Via RPM Configuration Packages
 - Via Custom Scripting
 - Or Via Proper CM Tools
- ACLs and Partitioning
- Protects Sensitive Data
- Local SAs Fully Control the Resulting Install
- A VCS Can Be Used But Not Required
- Cannot Manage Change After Installation

Realm Linux Management Tools

- Known as RLMTools or as the “Liquid Dragon Project”
- Implements a Hierarchy of Key/Value Attributes for Each Realm Linux Machine
- Sorts Machines Into Departments or Logical Groupings
- Collects Statistics and Information From Machines
- Creates `/etc/update.conf` If It Does Not Exist

Section Outline

1 Why Configuration Management?

- What is Realm Linux?
- Tools of the Realm Linux Past and Present
- The Realm Linux Future

Tools for Change

- Realm Linux Has All the Infrastructure to Support a CM Tool
 - Auto Installs
 - Auto Errata and Updates
 - Database of Additional Attributes
 - RFC 4122 Compliant UUIDs
- Need a CM Tool to Manage and Push Change During Deployment
 - Cfengine
 - Puppet
 - LCFG
 - Bcfg2 (Bee-Config-Two)
- Replace the RealmConfig Tool

Problems with Existing CM Tools

- Assume a Centralized Environment
 - One Master Configuration Specification Repository
 - One Trusted Group of SAs
 - No Concept of Partitioning
- We Have a Distributed Administration Environment
- Unable to Keep Sensitive Data Private From Other SA Groups
- How Would Student Contributions Work in a Centralized Setup?
- Development and Porting Efforts?

Section Outline

2 Overview of Our Bcfg2 Architecture

- Bcfg2 Goals
- Mixing in Git
- Python
- Liquid Dragon

Configuration Management Goals

- Generic Tool to Verify and Enforce Configuration
- Store, Version Control, Log, and Serve Files Containing Sensitive Data
- Clients Must be Authenticated and Encrypted
- Multiple Configuration Repositories for Distributed Administration
- Merge Configuration Updates from Master Repository into Child Repositories
- Utilize Department Strings for Domains of Control
- Utilize Client Data From RLMTools

Bcfg2

- <http://bcfg2.org>
- Documentation: <http://docs.bcfg2.org>
- Python Based
- Integrates Easily With Realm Linux 5 and Future Versions
- Supports Fedora, RHEL, Ubuntu, SuSE, Solaris, OSX, et al.
- No Custom Specification Language
- Simple XML, Genshi³ Templates, Flat Files
- Plugins Allow Easy Access to RLMTools and Other Data
- Client to Server Communication Authenticated and Encrypted Using XMLRPC

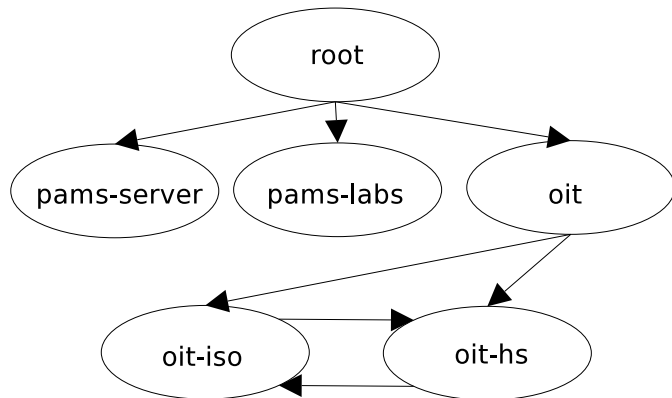
³<http://genshi.edgewall.org/>

Multiple Bcfg2 Repository Design

- A Tree Structure of Repositories Support Partitioning
- Basic Realm Linux Configuration Lives in a root Repository
- Realm Linux Machines Use root By Default
- Bcfg2 Repositories Live in AFS
 - Backups
 - Authorization
- SA Groups Can Create As Many Repositories as Needed
 - By Departments
 - Machines Search Tree for Closest Bcfg2 Repository
 - Any Level of Customization for Realm Linux
- Changes in the root Repository Are Pushed
- Any Repository Can Share Changes With Any Other

Repository Branches

Figure: An Example of Multiple Repositories Branched From the root.



Section Outline

2 Overview of Our Bcfg2 Architecture

- Bcfg2 Goals
- **Mixing in Git**
- Python
- Liquid Dragon

Managing Related Repositories

- Version Control Systems (VCS)
 - All CM Specification Repositories Live in a VCS
 - A VCS is the Tool Providing Change Logs and History
 - Core Technology in the Concepts of CM
 - Assumption of a Centralized VCS
- VCS Technology Has Advanced Significantly
 - 2005: BitKeeper and the Linux Kernel⁴
 - Emergence of Git and Other DVCSs
 - Scale, Fast, Many Branches, Low Cost to Merge Branches
- Distributed Version Control Systems (DVCS)
- Each IT Group or Department Gets Its Own Bcfg2 Repository in Git
- Issues with Subversion⁵

⁴<http://lwn.net/Articles/130746/>

⁵<https://secure.linux.ncsu.edu/moin/Bcfg2/WhyGit>

Git DVCS

- <http://git-scm.com/>
- Very Popular and Featureful⁶
- The Swiss Army Knife of DVCSes
- Merging Strategies and Rebasing
- Support for Non-Linear Branching and Merging
- Git Repositories Are File System Agnostic
- Ability to Create Off Site Repositories

⁶<http://whygitisbetterthanx.com>

Section Outline

2 Overview of Our Bcfg2 Architecture

- Bcfg2 Goals
- Mixing in Git
- Python
- Liquid Dragon

About Using Python

Knowledge of the Python Programming Language is not required to use Bcfg2. However, it is definitely a plus.

- <http://python.org>
- <http://diveintopython.org/>
- Genshi Templates Use a Python Like Syntax and Can Include Python Code
- Python Code Snippets Can Reference RLMTools Attributes

Section Outline

2 Overview of Our Bcfg2 Architecture

- Bcfg2 Goals
- Mixing in Git
- Python
- Liquid Dragon

Client Metadata

- RLMTTools
 - Client Authentication via Public Key Certificates
 - Assigns Universally Unique ID
 - Knowledge of Departments
- Added Key/Value Attributes
 - Clients
 - Departments
- Hierarchy of Departments and Clients
 - Authorization
 - Clients/Departments Inherit Attributes from Parents
- Imported Web-Kickstart as Attributes
- A Linux AD?

Putting It All Together

- CM Server Pool (2 Servers)
 - Each Server Runs a Bcfg2 Server Instance For Each Repository
 - Each Server Uses Git to Pull Latest Repository Changes to Local Storage
 - Bcfg2 Server Instances Use RLMTTools to Identify via UUID and Store Data About Each Client
- Realm Linux Bcfg2 Clients
 - Clients Find Bootstrap Information from RLMTTools Per Department
 - Repository Password Not Sent in Clear Text to Client
 - Initial Bcfg2 Client Run Configures Machine To Use The Same Repository For Future Runs
- Welcome To A Distributed Configuration Management System

Section Outline

3 Conclusions

Getting Help

- Support Tickets: `linux@help.ncsu.edu`
- Jabber: `cls@conference.jabber.eos.ncsu.edu`
- Mailing Lists
 - `realmlinux-users@lists.ncsu.edu`
 - `realmlinux-dev@lists.ncsu.edu`
- CLS Wiki: <https://secure.linux.ncsu.edu/moin/>
- See the Following References

Git References



CLS Wiki Bcfg2 Documentation

<https://secure.linux.ncsu.edu/moin/Bcfg2>



Dive Into Python

<http://diveintopython.org/>

Feedback and Questions