10 steps to build a Standard Operating Environment

Dirk Herrmann
Principal Software Engineer

Benjamin Kruell
Senior Domain Architect
Subject of this presentation

- Comprehensive Doc (~ 300 pages)
- Validated in our lab
- Target Publishing Date: Satellite 6.1 GA
Sample Customer Scenario

ACME Corp.

Sample Application Architecture

Sample Datacenter Topology

Sample IT Organization
Chapter Contents

- Introduction into related Satellite 6 entities
- Demonstration of possibilities -> 4 scenarios
- Background, concepts & recommendations
- Step-by-step Implementation using UI
- Implementation using hammer CLI
10 Steps to build a Standard Operating Environment

1. Setup your System Management Infrastructure
2. Map your Location and Datacenter Structure
3. Define your Definitive Media Library Content
4. Define your Content Lifecycle
5. Define your Core Build
6. Define your Application Content
7. Automate your Provisioning
8. Map your IT Organization & Roles
9. Continuous Lifecycle Management
10. Automate and extend your setup

Starting with an empty Satellite 6, creating step by step all required Satellite entities up to an up and running infrastructure and its ongoing maintenance.
Objective

Enabling our customers and partners to setup a similar scenario in less than **one** week.
1. Setup your System Management Infrastructure  
2. Map your Location and Datacenter Structure  
3. Define your Definitive Media Library Content  
4. Define your Content Lifecycle  
5. Define your Core Build  
6. Define your Application Content  
7. Automate your Provisioning  
8. Map your IT Organization & Roles  
9. Continuous Lifecycle Management  
10. Automate and extend your setup

<table>
<thead>
<tr>
<th>Task</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Setup your System Management Infrastructure</td>
<td>N/A</td>
</tr>
<tr>
<td>2. Map your Location and Datacenter Structure</td>
<td>95 %</td>
</tr>
<tr>
<td>3. Define your Definitive Media Library Content</td>
<td>100 %</td>
</tr>
<tr>
<td>4. Define your Content Lifecycle</td>
<td>100 %</td>
</tr>
<tr>
<td>5. Define your Core Build</td>
<td>100 %</td>
</tr>
<tr>
<td>6. Define your Application Content</td>
<td>100 %</td>
</tr>
<tr>
<td>7. Automate your Provisioning</td>
<td>95 %</td>
</tr>
<tr>
<td>8. Map your IT Organization &amp; Roles</td>
<td>100 %</td>
</tr>
<tr>
<td>9. Continuous Lifecycle Management</td>
<td>50 %</td>
</tr>
<tr>
<td>10. Automate and extend your setup</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Setup your System Management Infrastructure
Step 1 Topic Coverage

- Red Hat Satellite 6 Configuration
  - Red Hat Satellite 6 Configuration
  - Embedded Capsule Infrastructure Services
  - Red Hat Satellite 6 Organization
  - Red Hat Subscription Manifest

- Support Systems Configuration
  - Monitoring Server
  - Revision Control Server
  - Hammer CLI
Map your Location and DC Topology
Step 2 Topic Coverage

- Red Hat Satellite 6 Entities
  - Red Hat Satellite 6 Capsules
  - Capsule Features
  - Compute Resources (RHEV + RHELOSP)
  - RHELOSP Specific Adaptions
  - Red Hat Satellite 6 Locations
  - Red Hat Satellite 6 Domains
  - Red Hat Satellite 6 Subnets
Satellite 6.1 Capsule Improvements

- Provisioning
  - DNS
  - DHCP
  - TFTP
  - BMC
  - Realm Management

- Federated
  - Content Synchronization
  - Templates Synchronization
  - Reverse Proxy
  - Puppet Master
  - Puppet CA
ACME Sample Datacenter Topology
Define your Definitive Media Library Content
Step 3 Topic Coverage

- **Software Entry Points & Formats**
  - Red Hat Satellite 6 Content Types
  - Red Hat Satellite 6 Product & Repositories

- **Red Hat Satellite 6 Content Import**
  - GPG Keys
  - Red Hat & 3rd party Software Repositories
  - Custom and 3rd party Puppet Modules
  - Container Images
  - Synchronization Plans
Define your Content Lifecycle
Step 4 Topic Coverage

- **Red Hat Satellite 6 Content Views**
  - Content Views & Composite Content Views
  - Content View Scenarios
  - Content Views Recommendations

- **Red Hat Satellite 6 Lifecycle Environments**
  - Typical Lifecycle Environment Paths
  - Content View Lifecycle Management Scenarios
  - ACME Lifecycle Environments
Content View Scenarios

- One large “all-in-one” content view including all content (Red Hat and 3rd party) for all / many systems / server types
- Dynamic repository enablement using activation keys to avoid subscription overconsumption

- Host / server type specific content views for all types
- Automation for CV creation and updates using filters to ensure consistency of content (e.g. updating RHEL base chan at the same time for all affected CVs)

- Host / server type specific composite CVs for all types
- CCVs allow individual content updates for a particular subset (e.g. puppet config in a dedicated CV while leaving RHEL Base CV unchanged)

- Host / server type specific composite CVs for all types based on combining re-usurable application components
- CCVs (profiles) flowing through lifecycle environments while inherent CVs (profiles) don't
Content View Scenarios

**Advantages of this scenario**

- Highest degree of standardization
- Highest degree of re-usable components
- Puppet modules can ensure cross RHEL release CVs
- Easier handling of segregation of duty on a CV basis
- Overall owner use Composite CVs (immutable CVs)
- Easier handling of independent release cycles

**Disadvantages of this scenario**

- Additional maintenance of Composite CVs
Content View Recommendations

● Content View Filters
  - Use filters with caution (especially include filters)
  - Filters do not resolve dependencies
  - Always select affected repositories

● Composite Content Views
  - Usage of a repo / module more than once not possible
  - CVs could be selected independent of LC ENV
  - Consider a separated CV for puppet configuration
Lifecycle Environment Scenarios

- Simplest options: one lifecycle stage for all applications and operating systems (no lifecycle management at all)
- Even if the single prod stage is optional we strongly recommend to have at least one stage if you're using sync plans

- Dedicated lifecycle environments which reflect software / content lifecycle stages used by all applications and OS
- (physical and virtual) resources are mapped to these lifecycle environments (could be persistent or non-persistent)

- Individual lifecycle env path’s for particular applications
- Supports segregation of duty in combination with independent release cycles and independent compute resources
- Note: special role of Core Build and app env’s for IT Ops

- Deviant lifecycle environments paths for particular applications require an enhanced staging
- Typically for applications require additional QA steps (UAT) to better align to a release pipeline
- Requires an overall mapping of these stages (process)
Define your Core Build Definition
Core Build

- **Core Build Characteristics**
  - Smallest common denominator for OS
  - Based on minimal install ( > kickstart definition)
  - Includes OS + typical management tools
  - Includes basic hardening
  - RHEL ABI/API Commitment

- **Core Build Content View Creation**
  - Software Repositories (Red Hat & 3rd party)
  - Example OS Configuration Puppet Modules
Core Build Recommendations

- Be the smallest common denominator of all Red Hat Enterprise Linux servers
- Be infrastructure (hardware and virtualization) agnostic
- Provides an application or platform-independent OS configuration
- Be a universal size that allows scaling up to all the sizes used
- Be based on a minimal installation
- Contains a partitioning schema and default filesystem layout
- Contains all Red Hat, third-party and custom software required on all systems
- Contains all configuration settings required on all systems
- Typically include basic hardening
Define your Application Content
Step 6 Topic Coverage

- **Application Layer Content Views (Profile)**
  - Puppet Modules
  - Config Groups
  - Software Repositories
  - Content View Publish

- **Server Type Composite Content Views (Role)**
  - Content View Assembly
  - Composite CV Publish & Promote
ACME Application Architecture

**Business Application**
- ACME Website
  - Apache Wordpress
  - MariaDB
  - RHEL7 CoreBuild
  - RHEL7 CoreBuild

**Infrastructure Services**
- IF Git
  - Git
  - RHEL7 CoreBuild
- IF Container
  - Container Host
  - RHEL7 CoreBuild
- IF Loghost
  - Loghost
  - RHEL6 CoreBuild
- Sat6 Capsule
  - Satellite6 Capsule
  - RHEL7 CoreBuild
Automate your Provisioning
Step 7 Topic Coverage

• Red Hat Satellite 6 Entities
  – PXE & Boot ISO
  – Provisioning Templates
  – Host Groups & Activation Keys
  – Parameters & Smart Class Parameters

• Provisioning Examples
  – Flexible Provisioning
  – Restore capable provisioning
Advanced examples: Dynamic Part Tables & Hooks

- Param controlled nested partition tables example
- Supports resiliency approach without data harming (fast re-provisioning)

- Foreman Hooks used to
  - integrate into external systems (Zabbix)
  - execute actions on Satellite (adding container host as compute resource if HG matches)
Host Groups

- Parent
- Name
- Lifecycle Environment
- Content View
- Puppet Environment
- Capsule Settings

- Puppet Classes
  - Puppet Class
  - Config Group

- Network
  - Domain
  - Subnet
  - Realm

- Operating System
  - Architecture
  - Operating System
  - Partition Table

- Parameter

- Activation Keys

- Locations

- Organizations
Satellite 6 Parameter & Smart Class Parameter

- Global
- Organization
- Location
- Domain
- Operating System
- Host Group
- Host

Smart Class Parameter

- Match: kt_env = PROD
- Use Puppet default
- Value: 172.24.99.10

- Match: kt_env = QA
- Use Puppet default
- Value: 10.0.40.30

+ Add Matcher-Value
Map your IT Org & Roles
Step 8 Topic Coverage

- **Sample Roles / Separation of Responsibilities**
  - Admin Role(s)
  - IT Ops Mgr (read-only)
  - License / Subscription Manager
  - OS / Core Build SysEng
  - QA Team

- **Satellite 6 Entities**
  - Satellite 6 Users & User Groups
  - Satellite 6 Roles & RBAC
Sample Role – OS / Core Build SysEng

- Expected tasks of this role

- RBAC configuration of this role
  - Predefined Manager role
  - Permissions
  - Filter

- Role creation using hammer CLI
Continuous Lifecycle Management
Step 9 Topic Coverage

• Red Hat Satellite 6 Lifecycle Management
  - Errata Overview & Search
  - Applicable vs. Installable Errata
  - Emergency Errata Management

• Content and Composite Content View Lifecycle
  - Core Build Updates
  - Application Updates + Combined Updates
  - Incremental Updates (Emergency Errata)
  - Puppet Module Updates
Automate & Extend
Step 10 Topic Coverage

• Intention of Step 10
  – Provide an outlook to further enhancements
  – Further ITSM process relationships
  – Short overview on items not covered in detail
  – Outlook to upcoming doc’s
What’s next?

Stay tuned :-)  

Read it

Provide feedback
Satellite-Related Sessions

**Wednesday**
1:20pm – 2:20pm
Satellite 6 Roadmap

2:30pm – 3:30pm
IKEA vs Shellshock: 1-0

3:40pm – 4:40pm
Real-World Perspectives: Managing Infrastructures with Satellite (Panel)

4:50pm – 5:50pm
Transitioning From Satellite 5 to 6

**Thursday (continued)**
1:20pm – 2:20pm
Shellshock, Heartbleed -- What's The Next Headache for Compliance

1:20pm – 2:20pm
CloudForms, Satellite 6 and Puppet for Automating JBoss EAP 6

3:40pm – 4:40pm
10 Steps To Build A Standard Operating Environment

4:50pm – 5:50pm
Puppet Enterprise and Satellite 6

**Friday**
9:45am – 10:45am
Satellite 6 Power User Tips and Tricks
Satellite Labs, Training and More

**Labs**

**Thursday**
3:30pm-5:30pm
Security Compliance Made Easy With OpenSCAP

**Friday**
9am-11am
Migrate From Red Hat Satellite 5 To Satellite 6

11:30am-1:30pm
Hands-On With Satellite 6.1

**Taste Of Training**

**Wednesday**
3:40pm – 4:40pm
Managing Software & Errata Deployment With Satellite 6

**Come See Us!**

- Visit the Satellite team in the Infrastructure Booth (306)!
- Visit the Foreman team in the Community Booth!

#redhat #rhsummit
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.