RED HAT SUMMIT

BOSTON, MA
JUNE 23-26, 2015

10 steps to build a Standard Operating Environment

Dirk Herrmann
Principal Software Engineer

Benjamin Kruell
Senior Domain Architect

redhat.

Subject of this presentation

Red Hat Reference Architecture Series

10 steps to build a
Standard Operating Environment

Dirk Herrmann
Principal Software Engineer

Benjamin Kruell
Senior Domain Architect

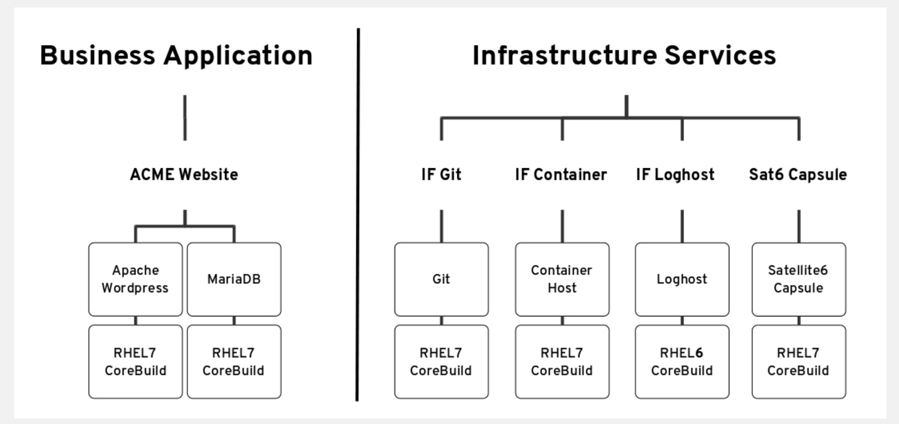
- Comprehensive Doc

  (~ 300 pages)
- Validated in our lab
- Target Publishing Date:

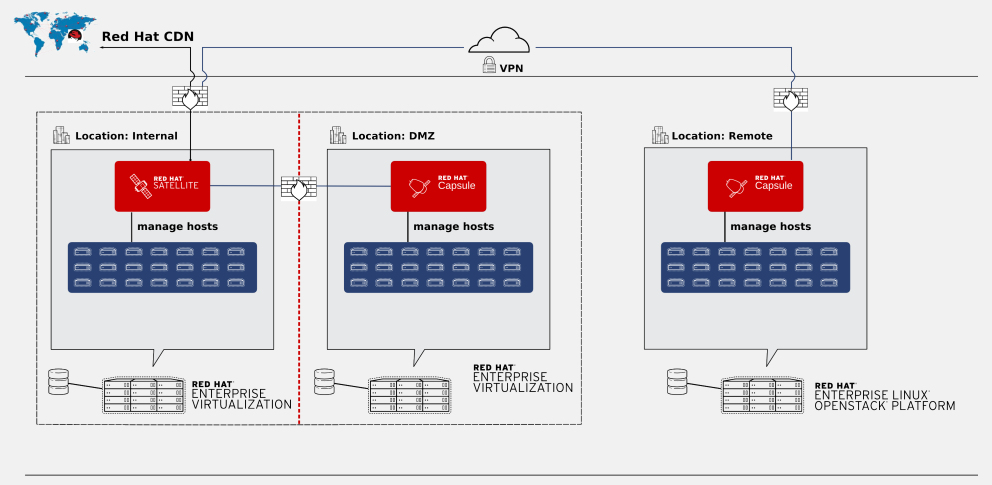
  Satellite 6.1 GA

Sample Customer Scenario

ACME Corp.

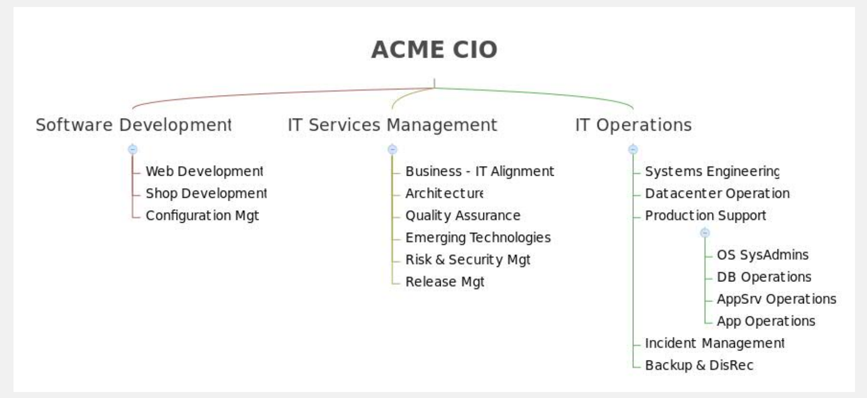


Sample Application Architecture



Red Hat CDN
VPN
Location: Internal
Location: DMZ
Location: Remote
manage hosts
manage hosts
manage hosts
RED HAT ENTERPRISE VIRTUALIZATION
RED HAT ENTERPRISE VIRTUALIZATION
RED HAT ENTERPRISE LINUX OPENSTACK PLATFORM

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.

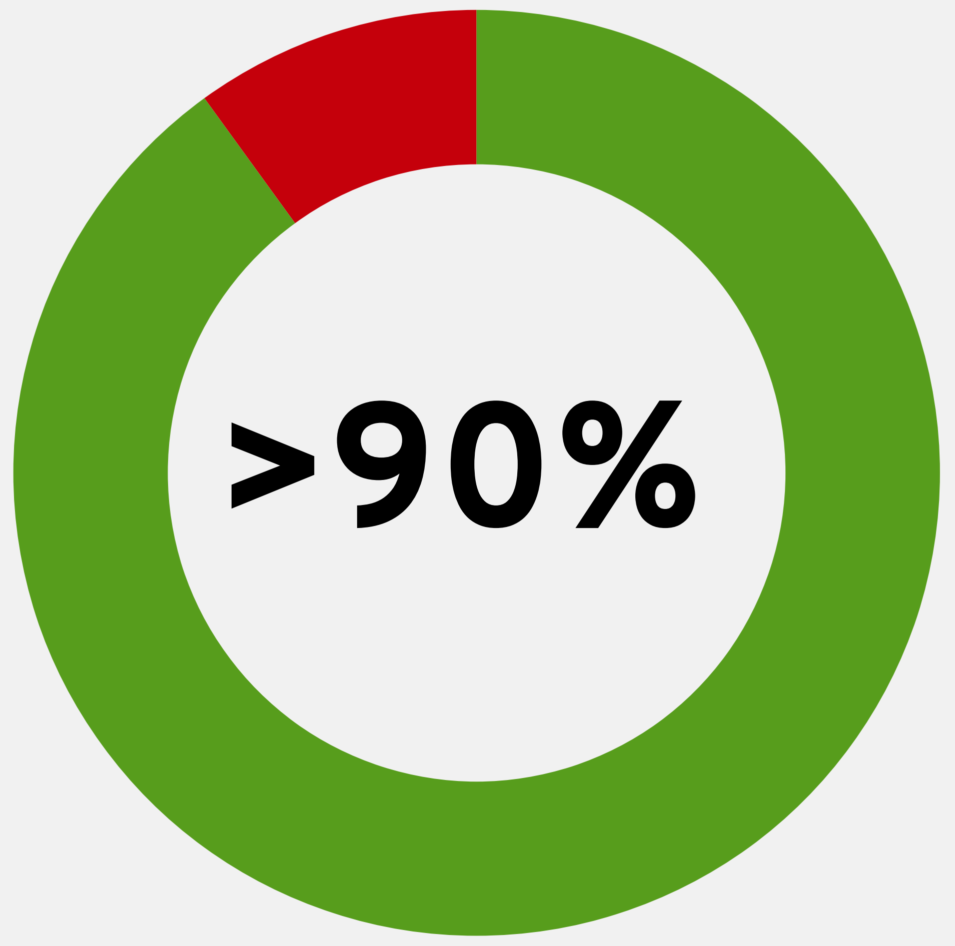
Objective

Enabling our customers and partners to setup a similar scenario in less than

one
week

Hammer CLI Coverage / Scripted Setup

1. Setup your System Management Infrastructure	N/A
2. Map your Location and Datacenter Structure	95 %
3. Define your Definitive Media Library Content	**100 %**
4. Define your Content Lifecycle	**100 %**
5. Define your Core Build	**100 %**
6. Define your Application Content	**100 %**
7. Automate your Provisioning	95 %
8. Map your IT Organization & Roles	**100 %**
9. Continuous Lifecycle Management	50 %
10. Automate and extend your setup	N/A



Setup your System Management Infrastructure

1

STEP

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

2

Map your Location and DC Topology

Step 2 Topic Coverage

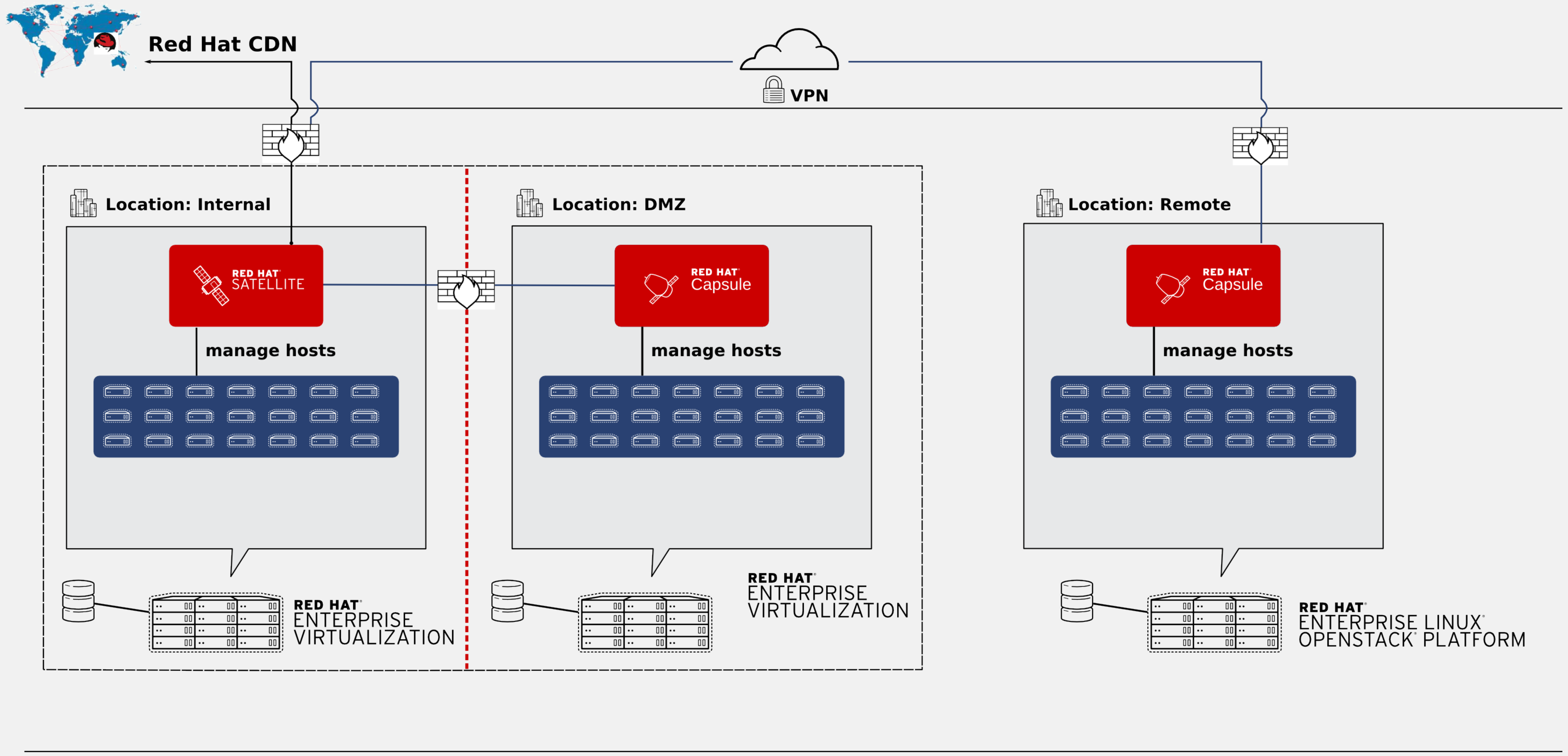
- **Red Hat Satellite 6 Entities**
  - Red Hat Satellite 6 Capsules
  - Capsule Features
  - Compute Resources (RHEV + RHELOSP)
  - RHELOSP Specific Adaptions
  - Red Hat Satellite 6 Locations
  - Red Hat Satellite 6 Domains
  - Red Hat Satellite 6 Subnets

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

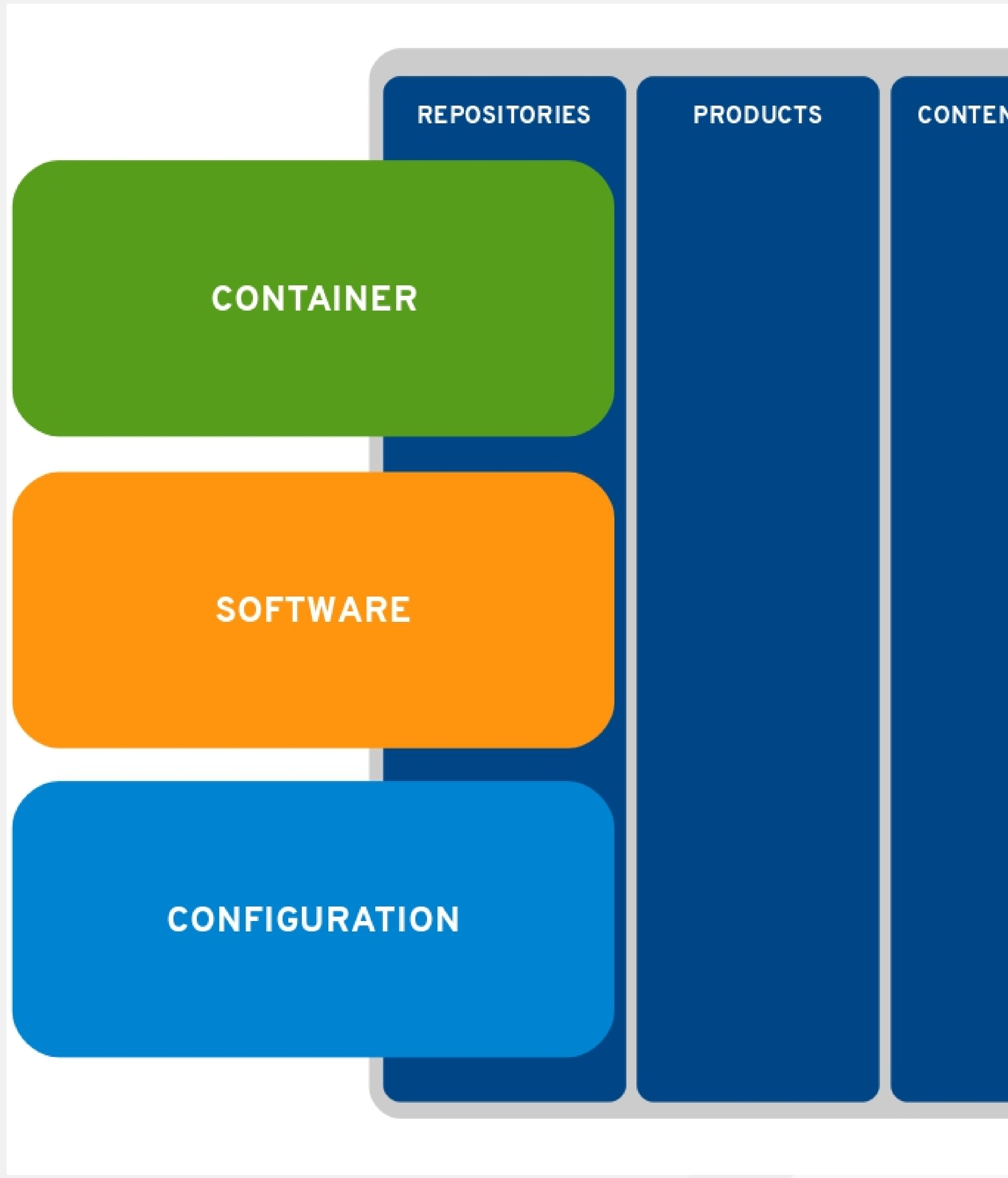


3

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



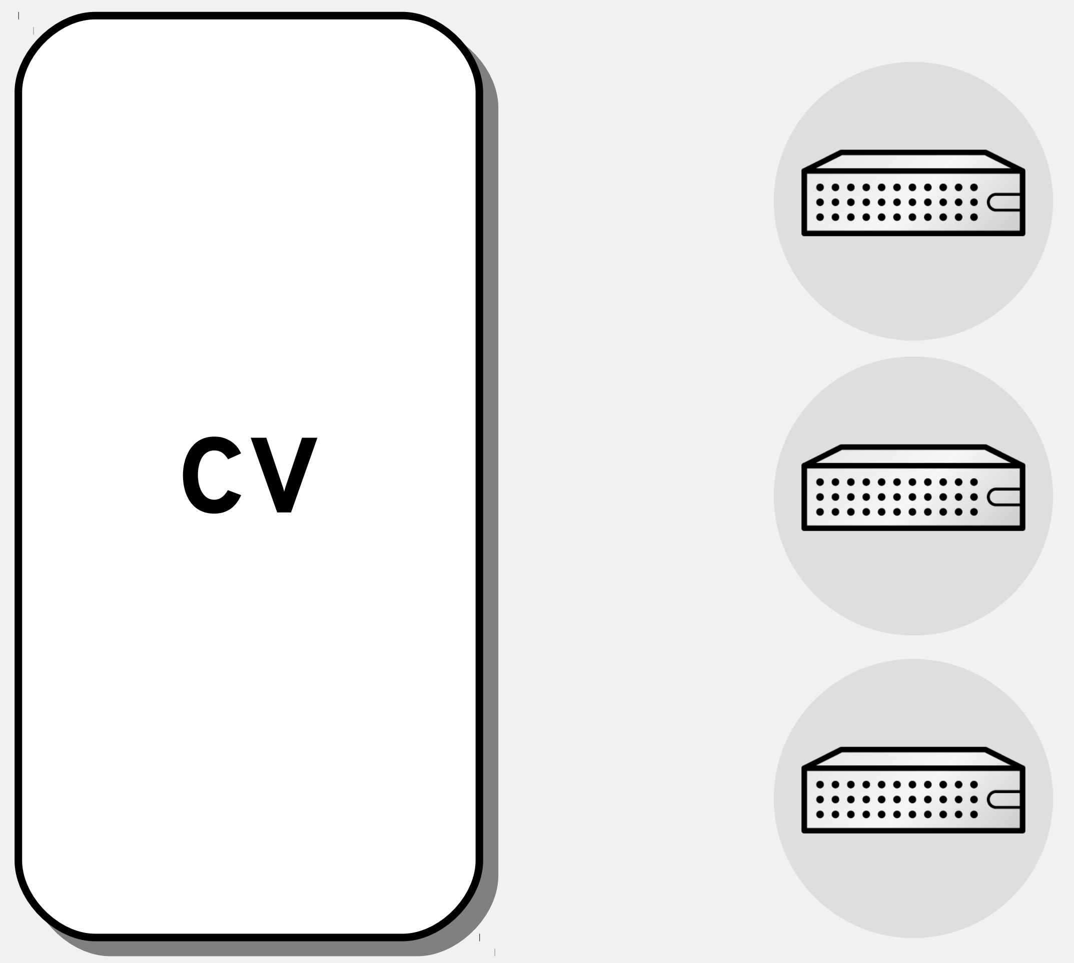
4

Define your Content Lifecycle

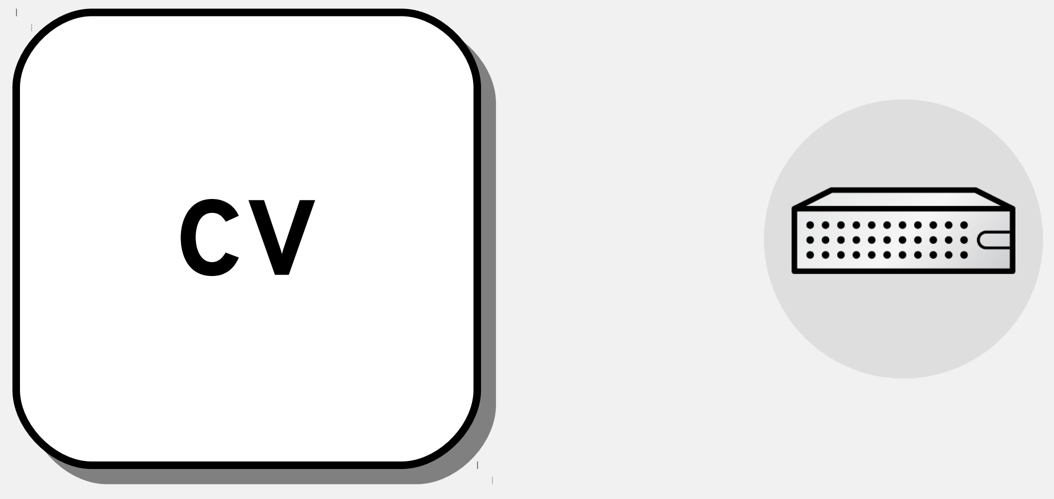
Step 4 Topic Coverage

- **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

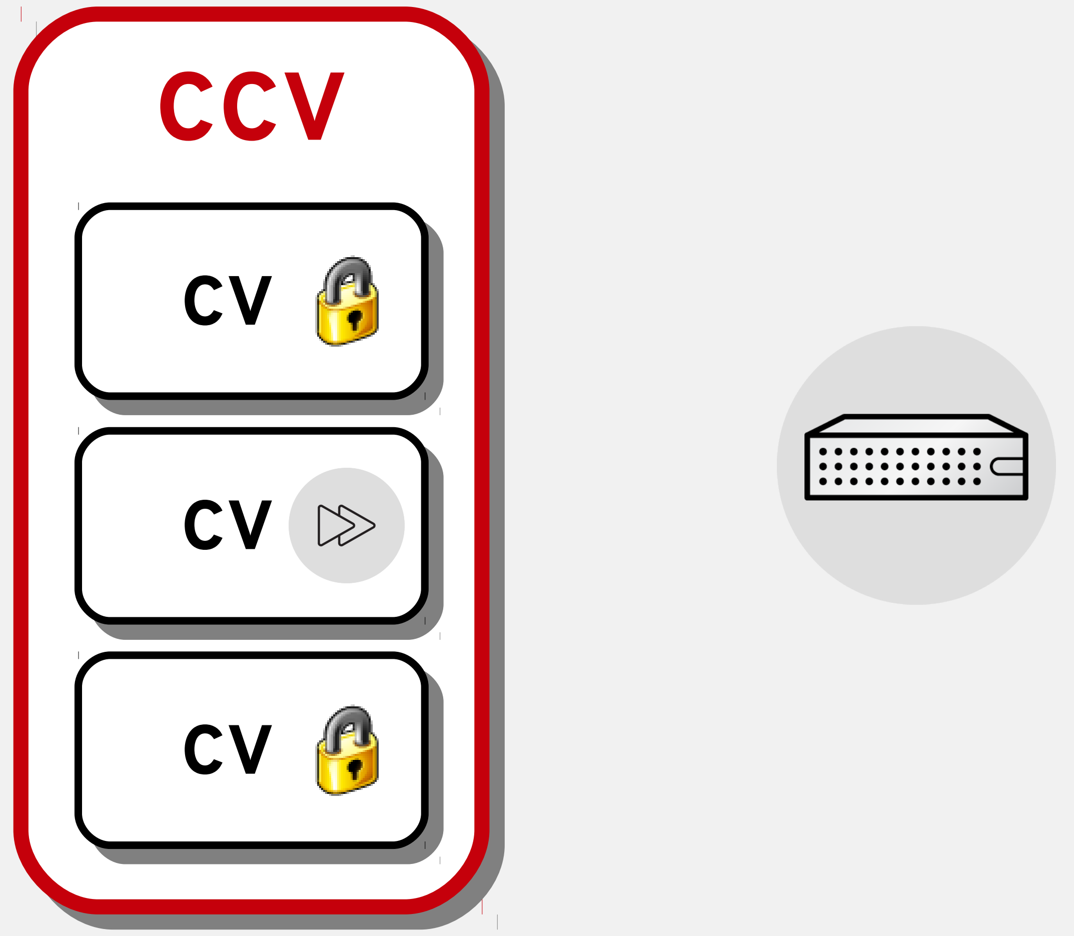
Content View Scenarios



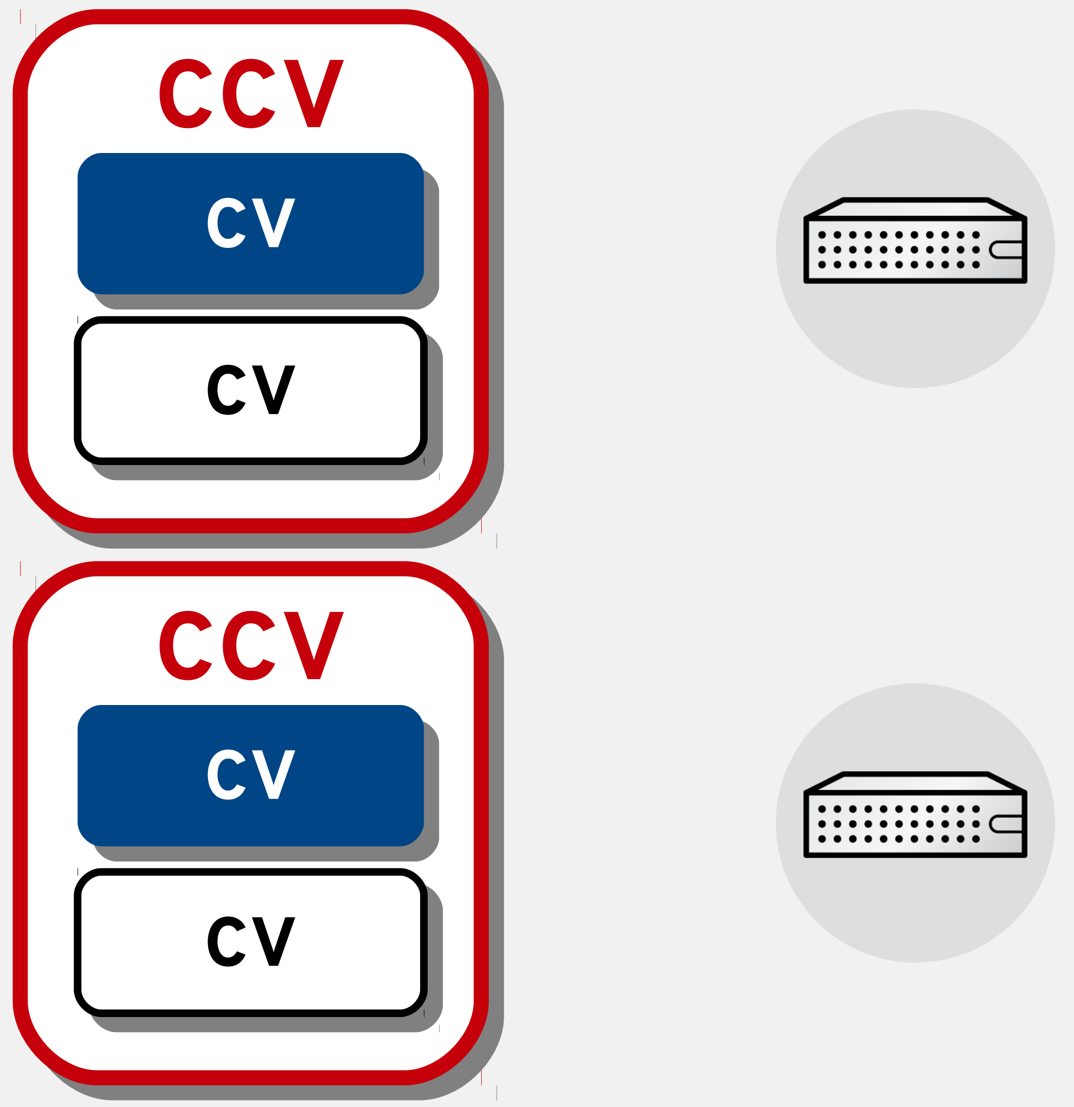
- **One large “all-in-one”** content view including all content (Red Hat and 3rd party) for all / many systems / server types
- Dynamic repository enablement using activation keys to avoid subscription overconsumption



- **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)



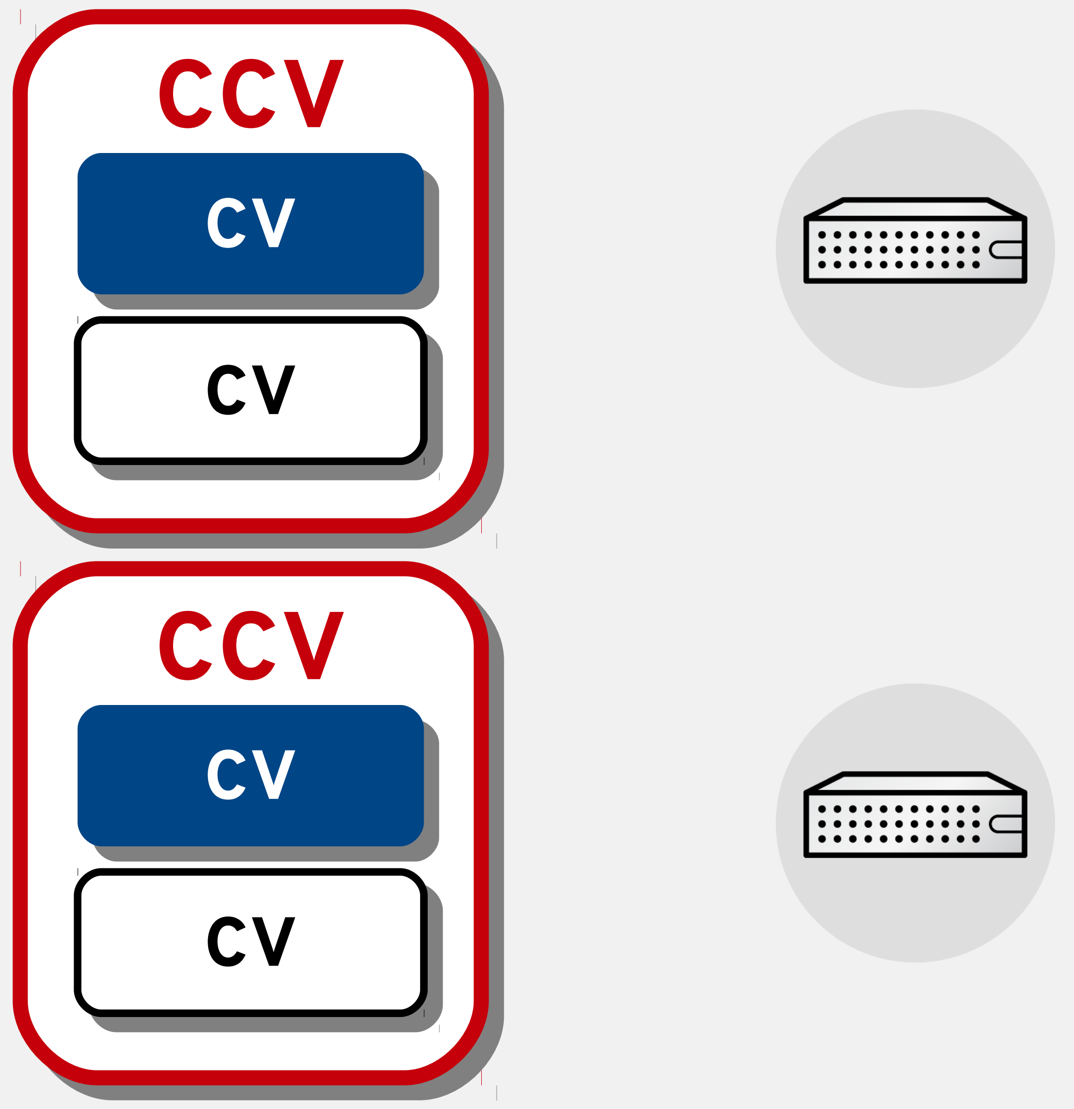
- **Host / server type specific** **composite** CVs for all types
- CCVs allow individual content updates for a particular subset (e.g. puppet config in a dedicated CV while leaving RHEL Base CV unchanged)



- **Host / server type specific composite** CVs for all types based on combining **re-usable 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

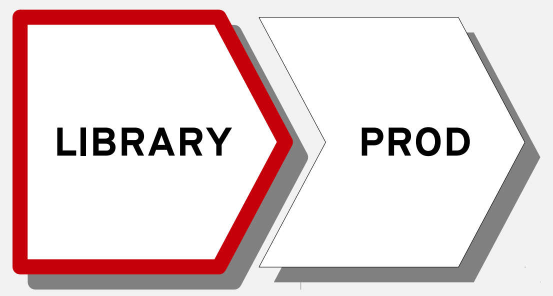


- **Host / server type specific composite** CVs for all types based on combining **re-usable application components**
- CCVs (**profiles**) flowing through lifecycle environments while inherent CVs (**profiles**) don't

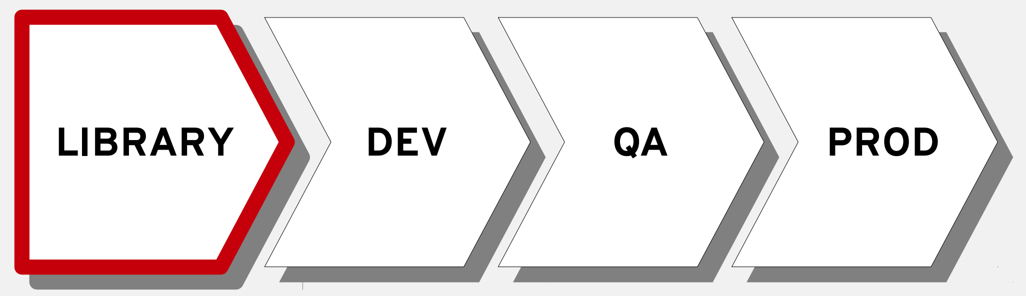
Content View Recommendations

- **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

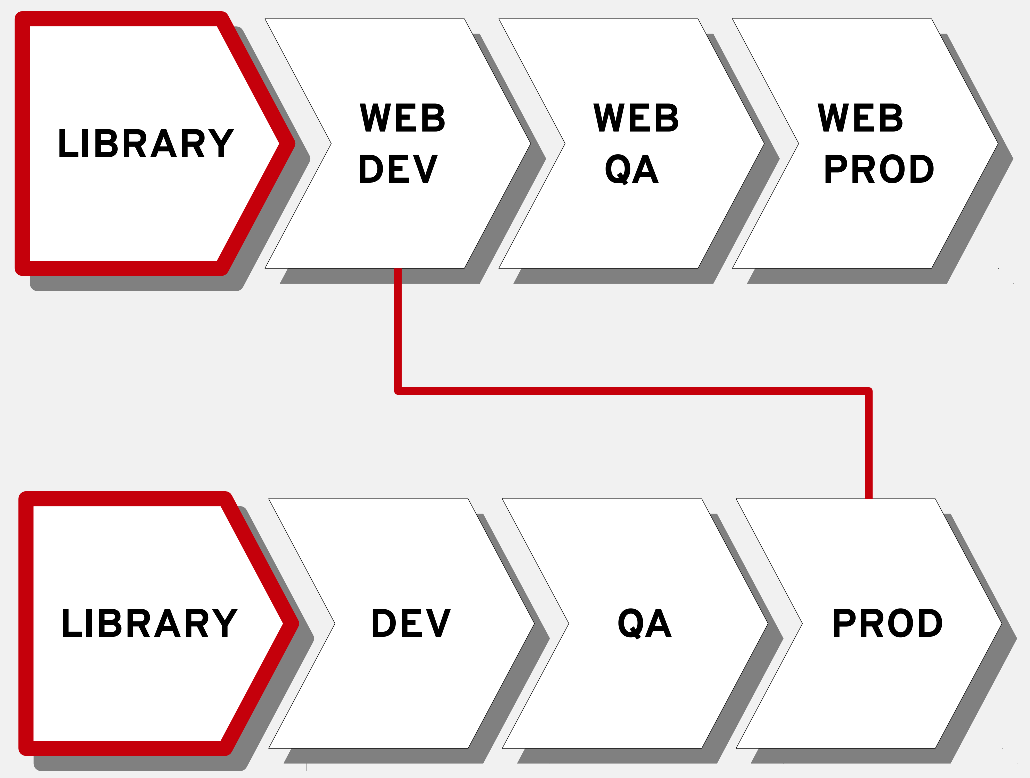
Lifecycle Environment Scenarios



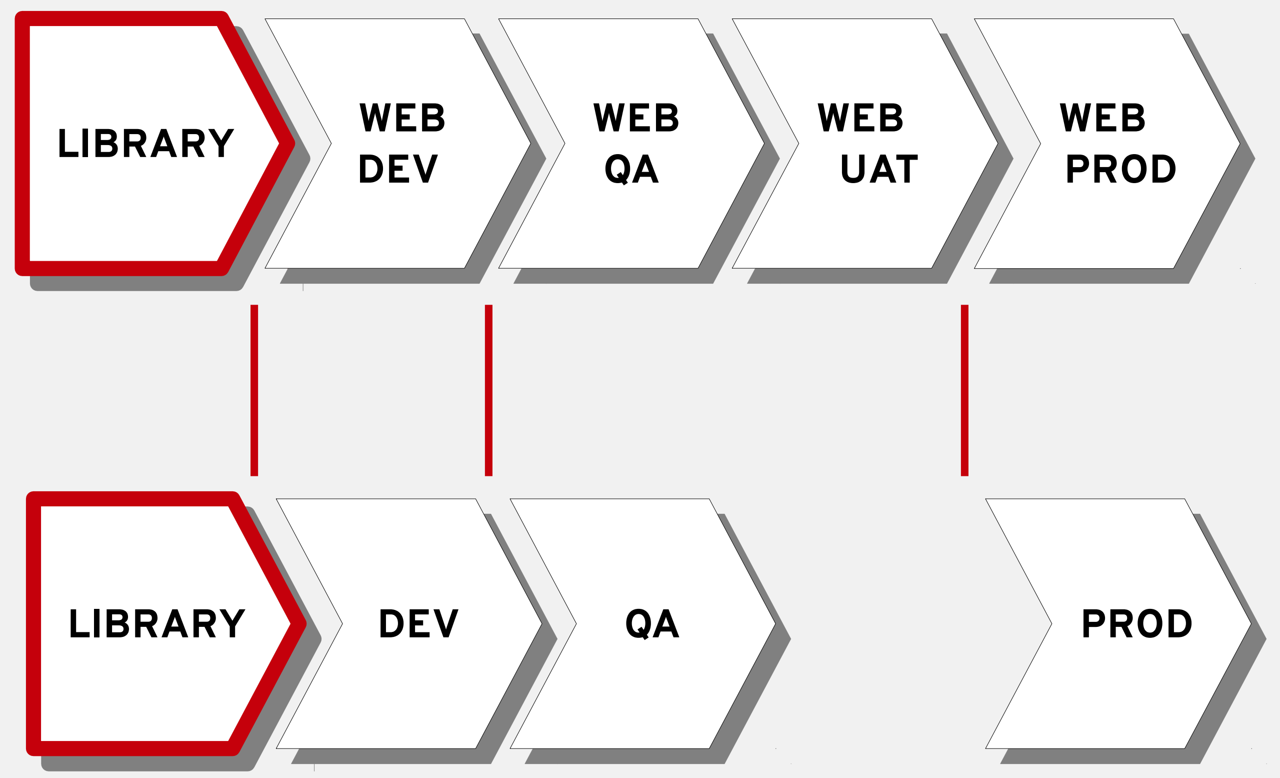
- Simplest options: one lifecycle stage for all applications and operating systems (no lifecycle management at all)
- Even if the single prod stage is optional we strongly recommend to have at least one stage if you're using sync plans



- 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



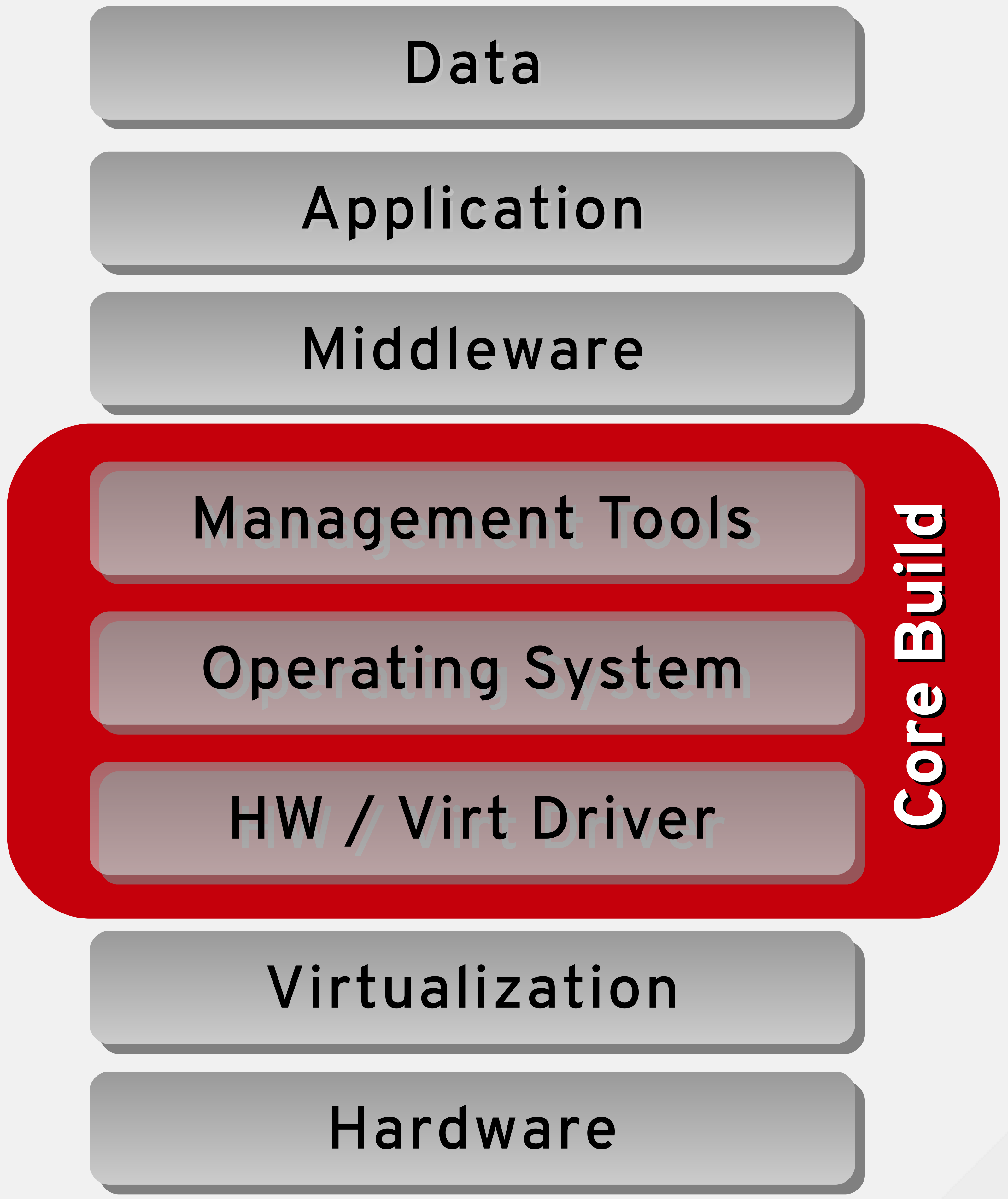
- Deviant lifecycle environments paths for particular applications require an enhanced staging
- Typically for applications require additional QA steps (UAT) to better align to a release pipeline
- Requires an overall mapping of these stages (process)

5

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

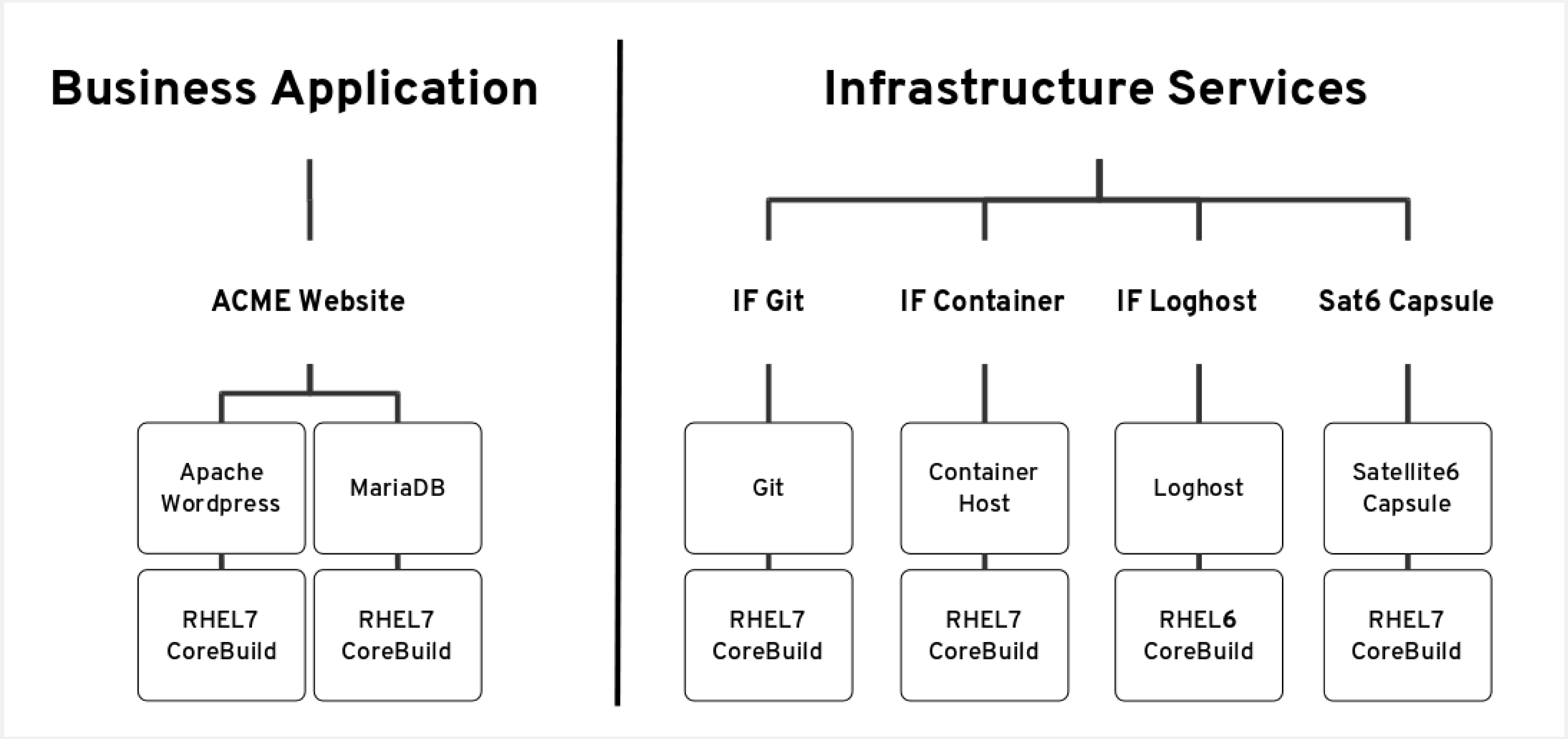
6

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



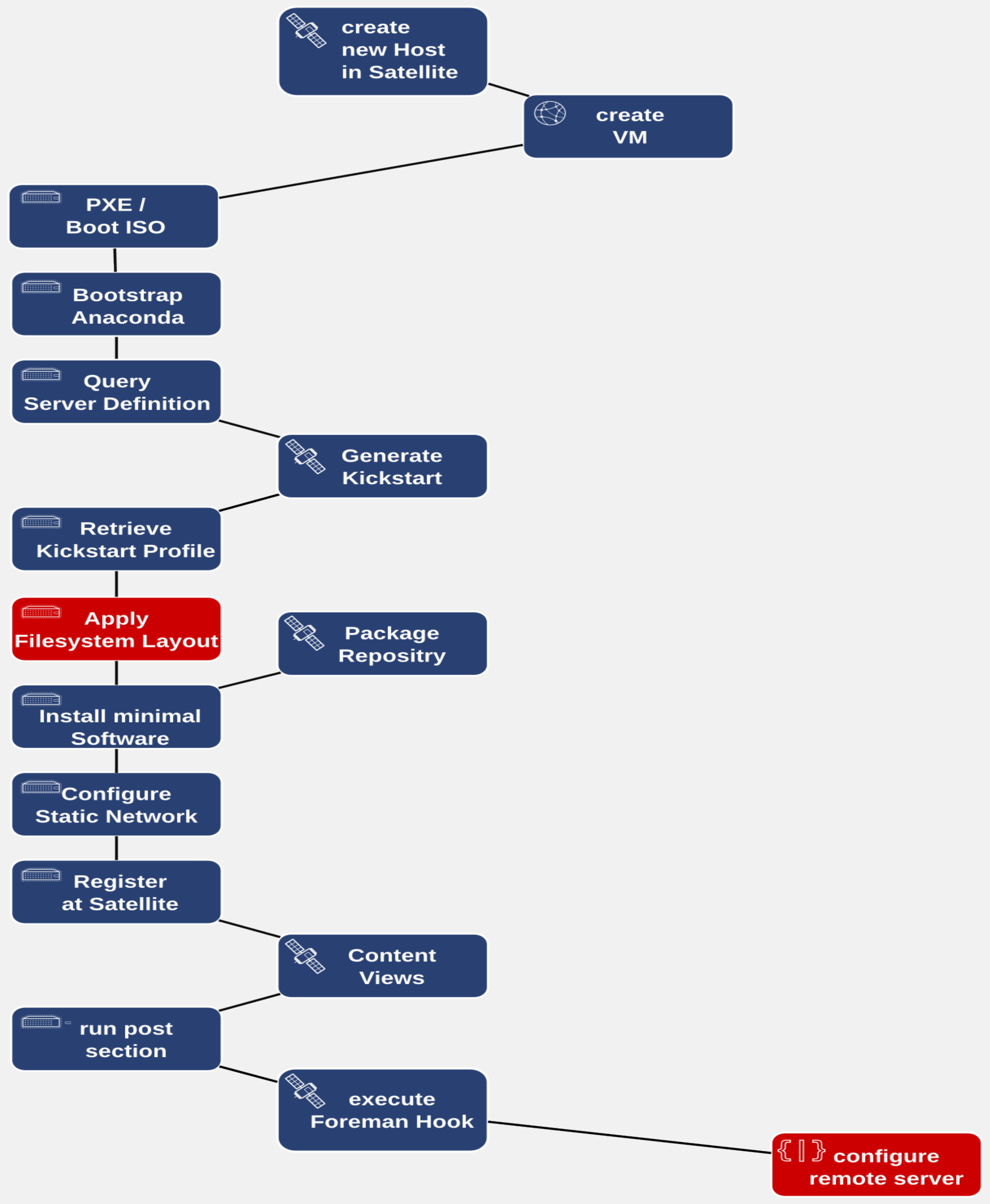
7

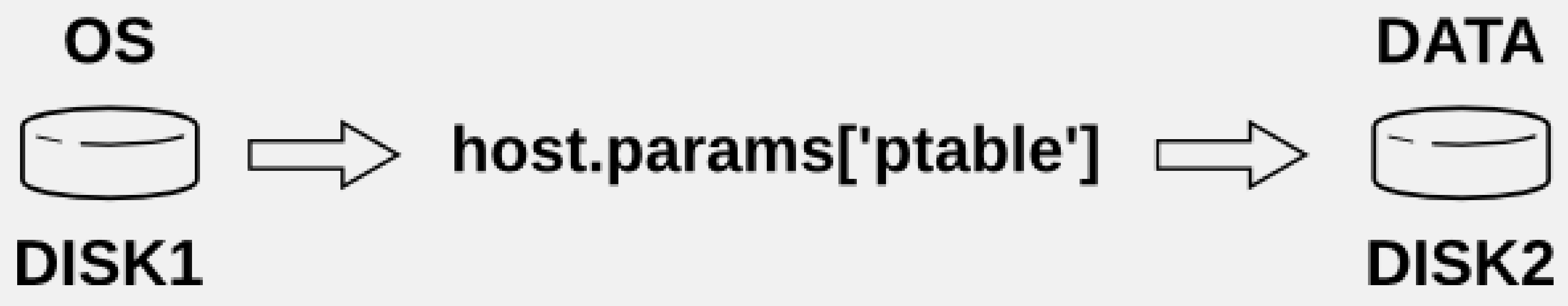
Automate your Provisioning

Step 7 Topic Coverage

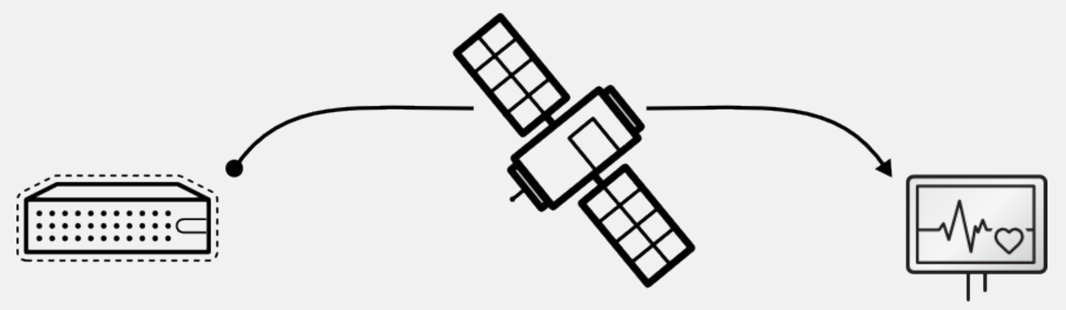
- **Red Hat Satellite 6 Entities**
  - PXE & Boot ISO
  - Provisioning Templates
  - Host Groups & Activation Keys
  - Parameters & Smart Class Parameters
- **Provisioning Examples**
  - Flexible Provisioning
  - Restore capable provisioning

Advanced examples: Dynamic Part Tables & Hooks



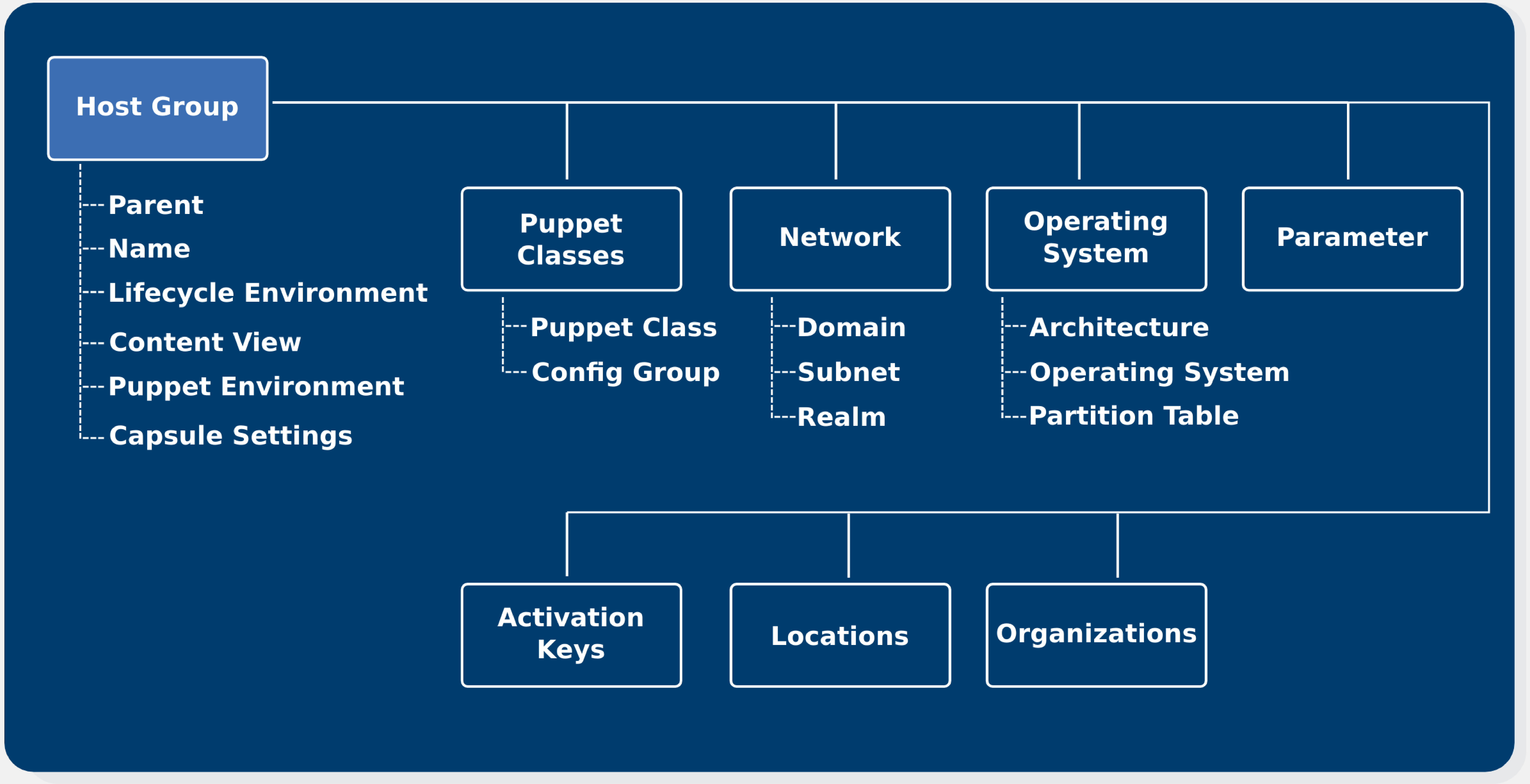


- Param controlled nested partition tables example
- Supports resiliency approach without data harming (fast re-provisioning)

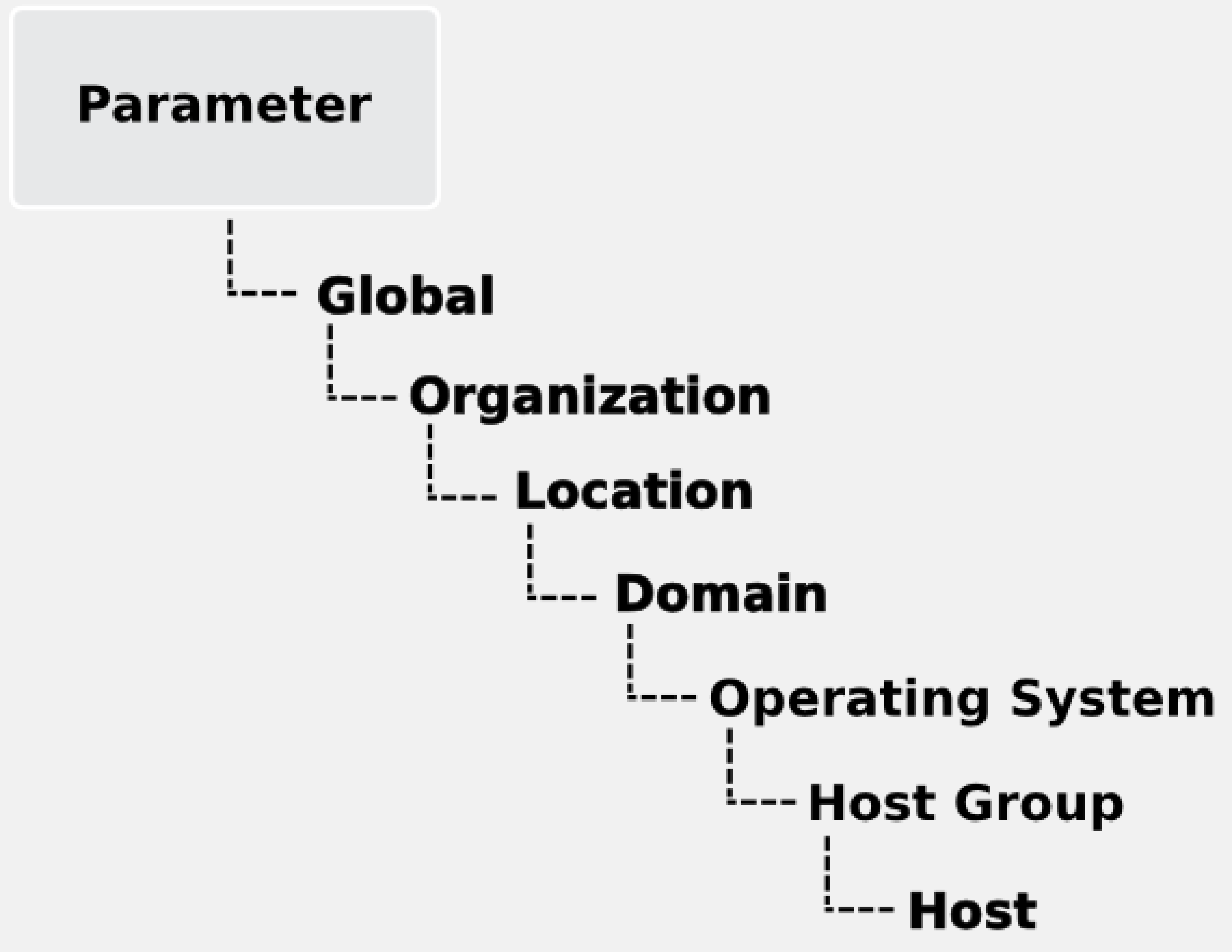


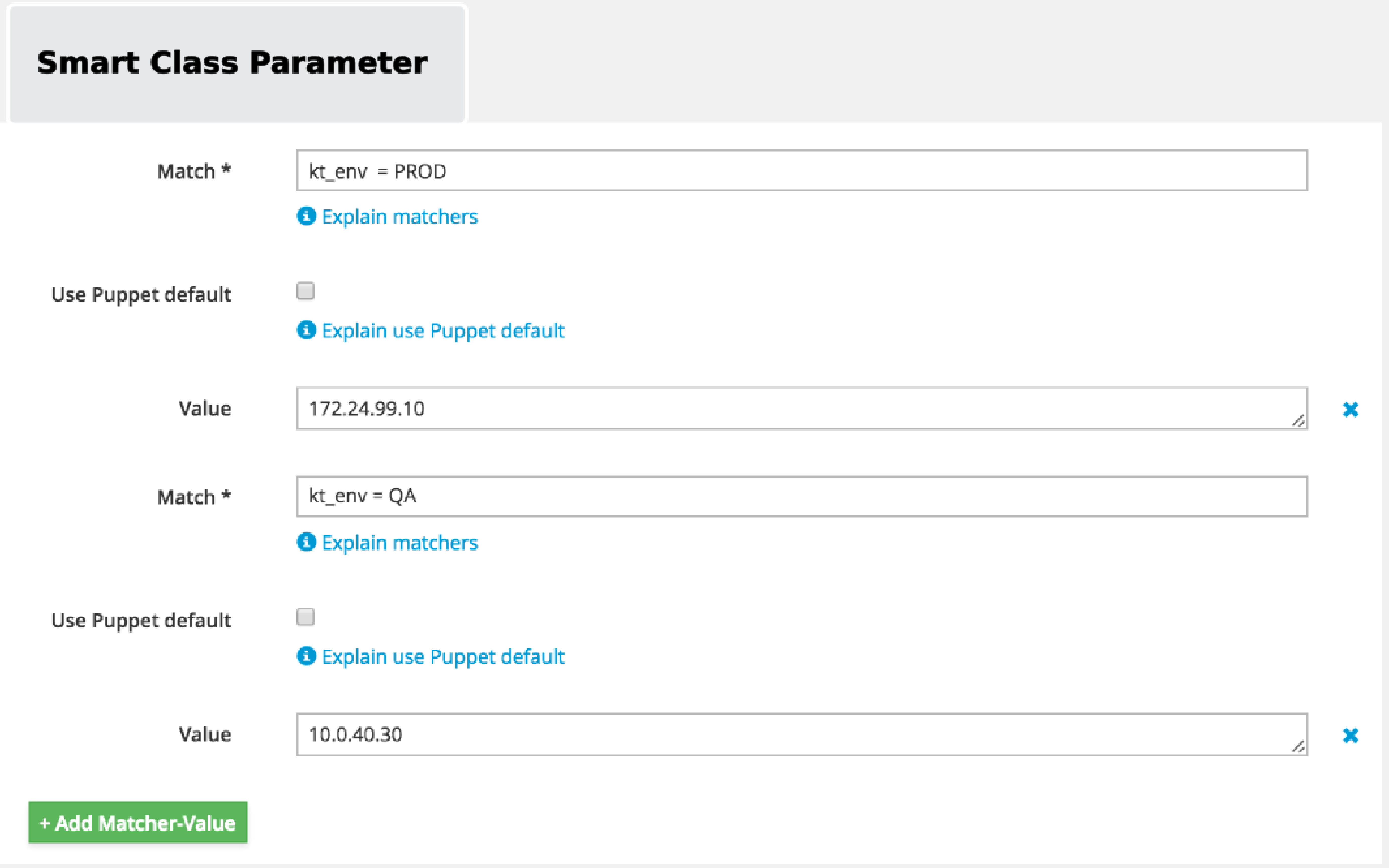
- Foreman Hooks used to
  - integrate into external systems (Zabbix)
  - execute actions on Satellite (adding container host as compute resource if HG matches)

Host Groups


Host Group
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

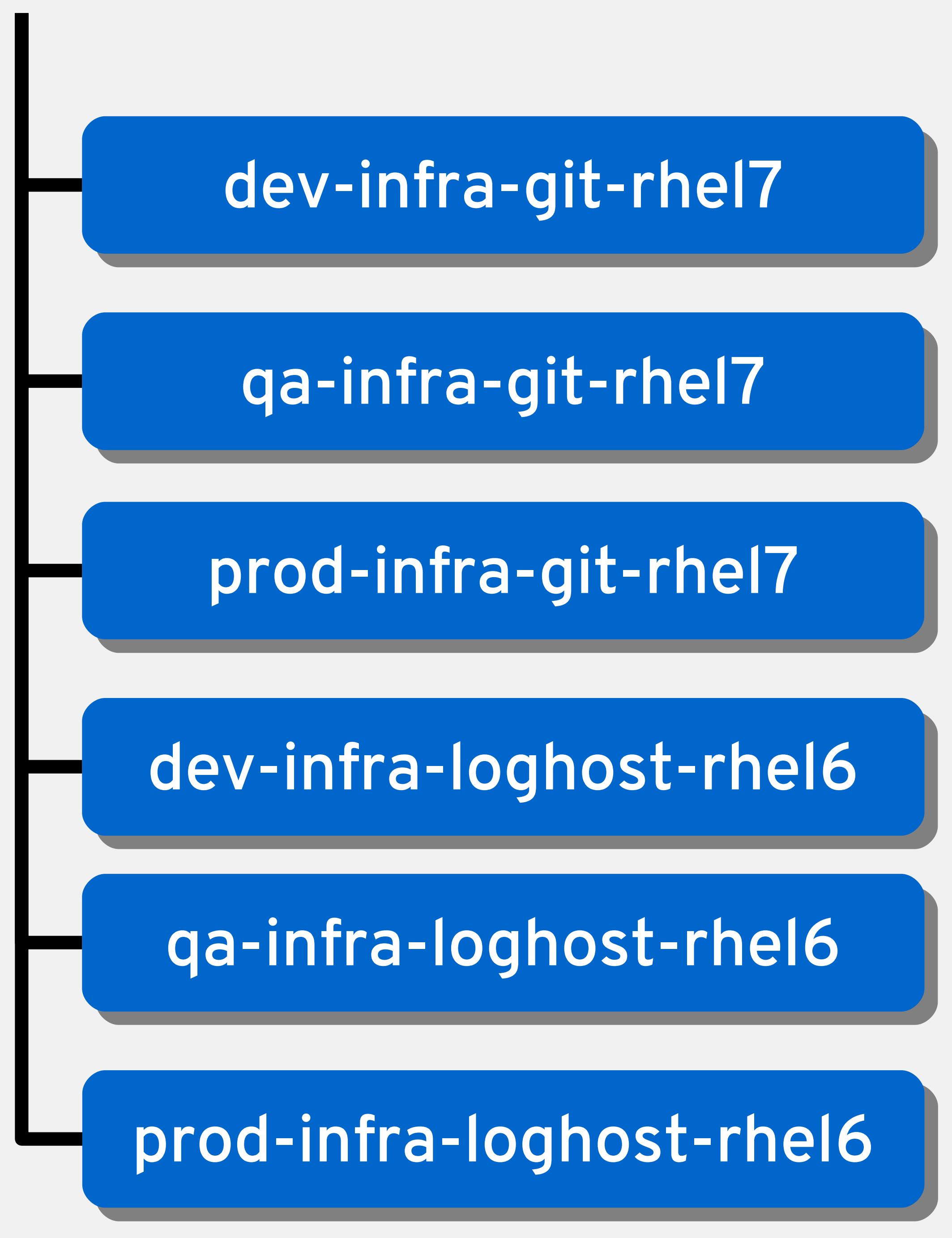
Satellite 6 Parameter & Smart Class Parameter

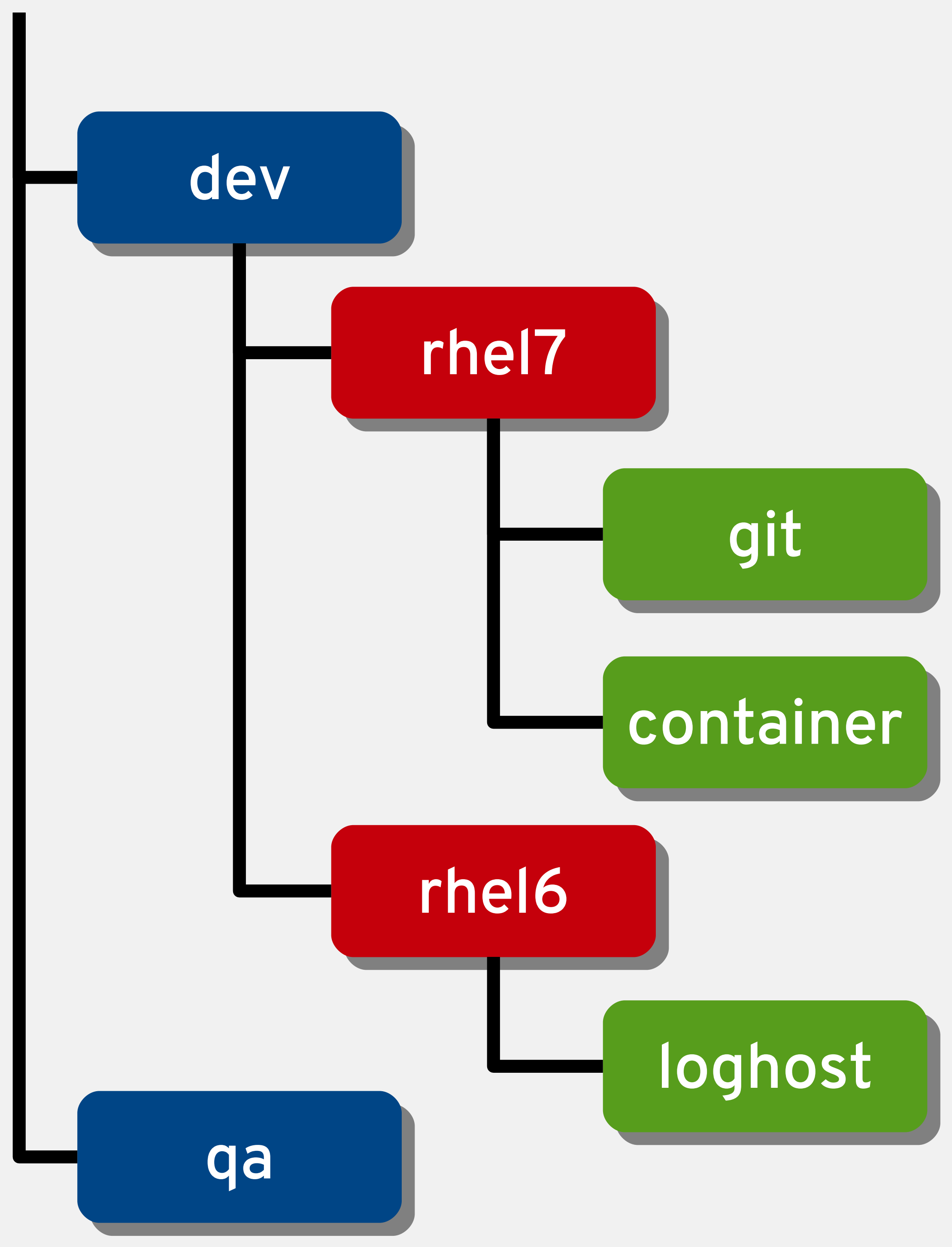


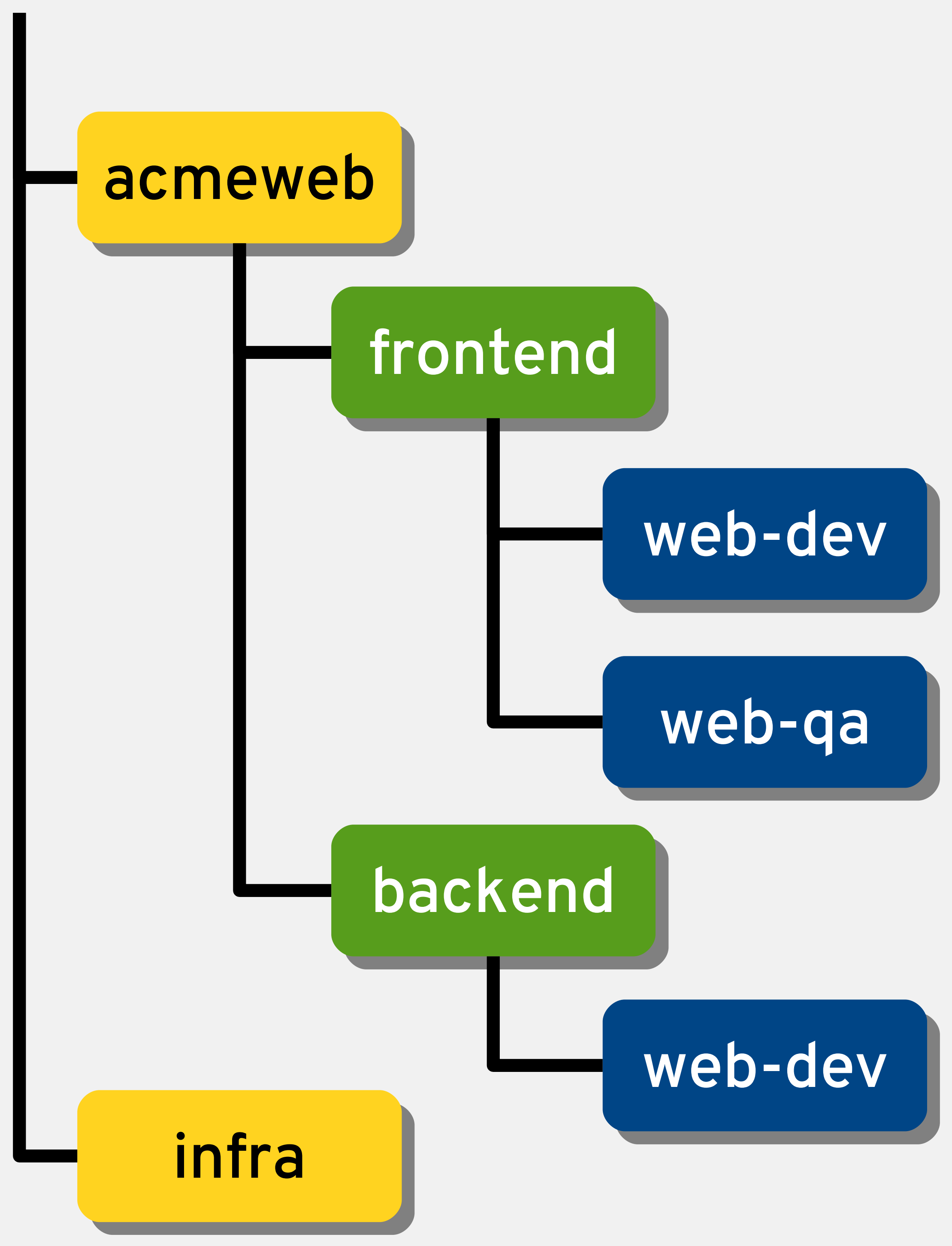

Smart Class Parameter

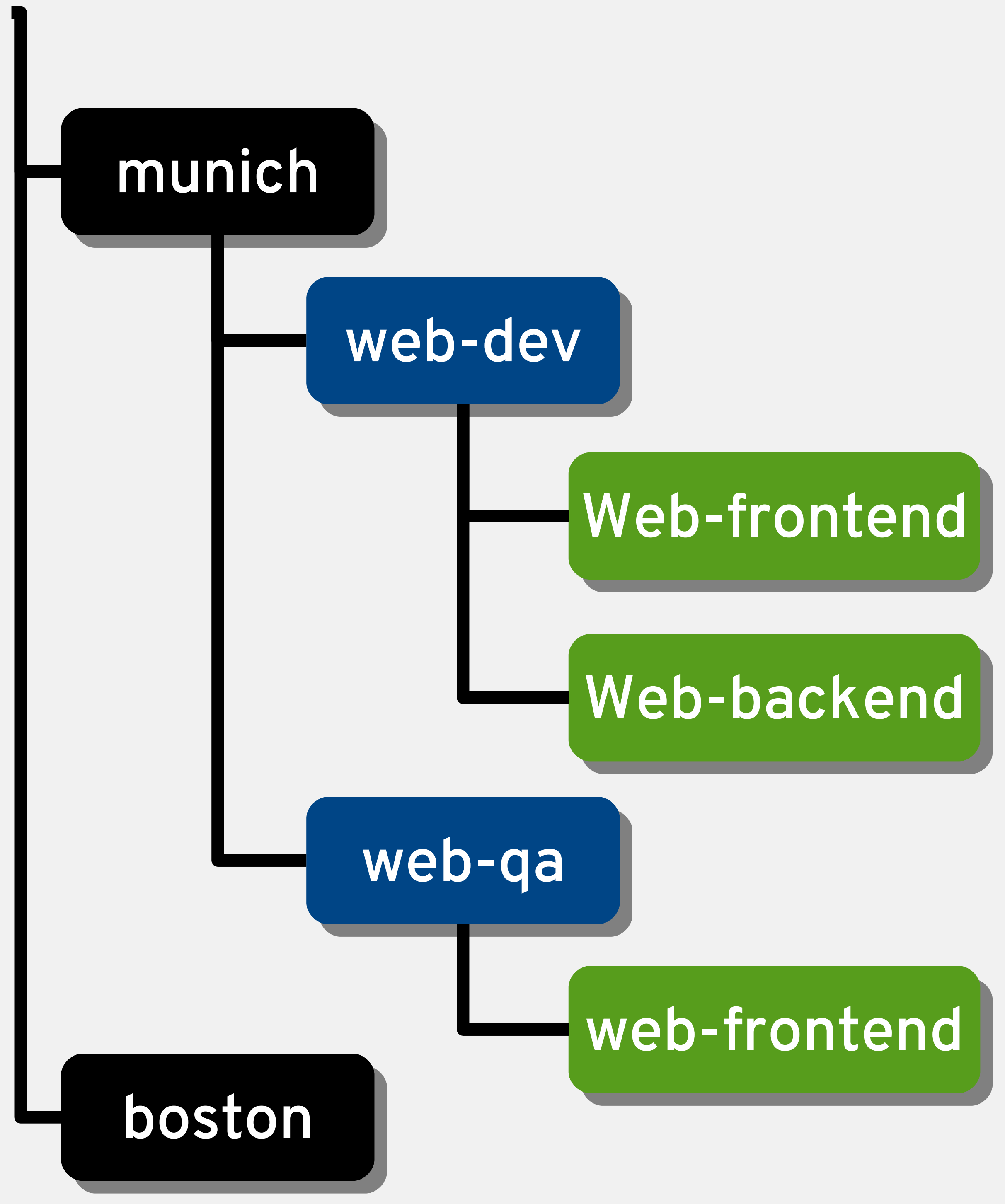
Match *	kt_env = PROD
Use Puppet default	☐
Value	172.24.99.10
Match *	kt_env = QA
Use Puppet default	☐
Value	10.0.40.30

Host Group Scenarios









Flat Structure

LC ENV Focus

App Focus

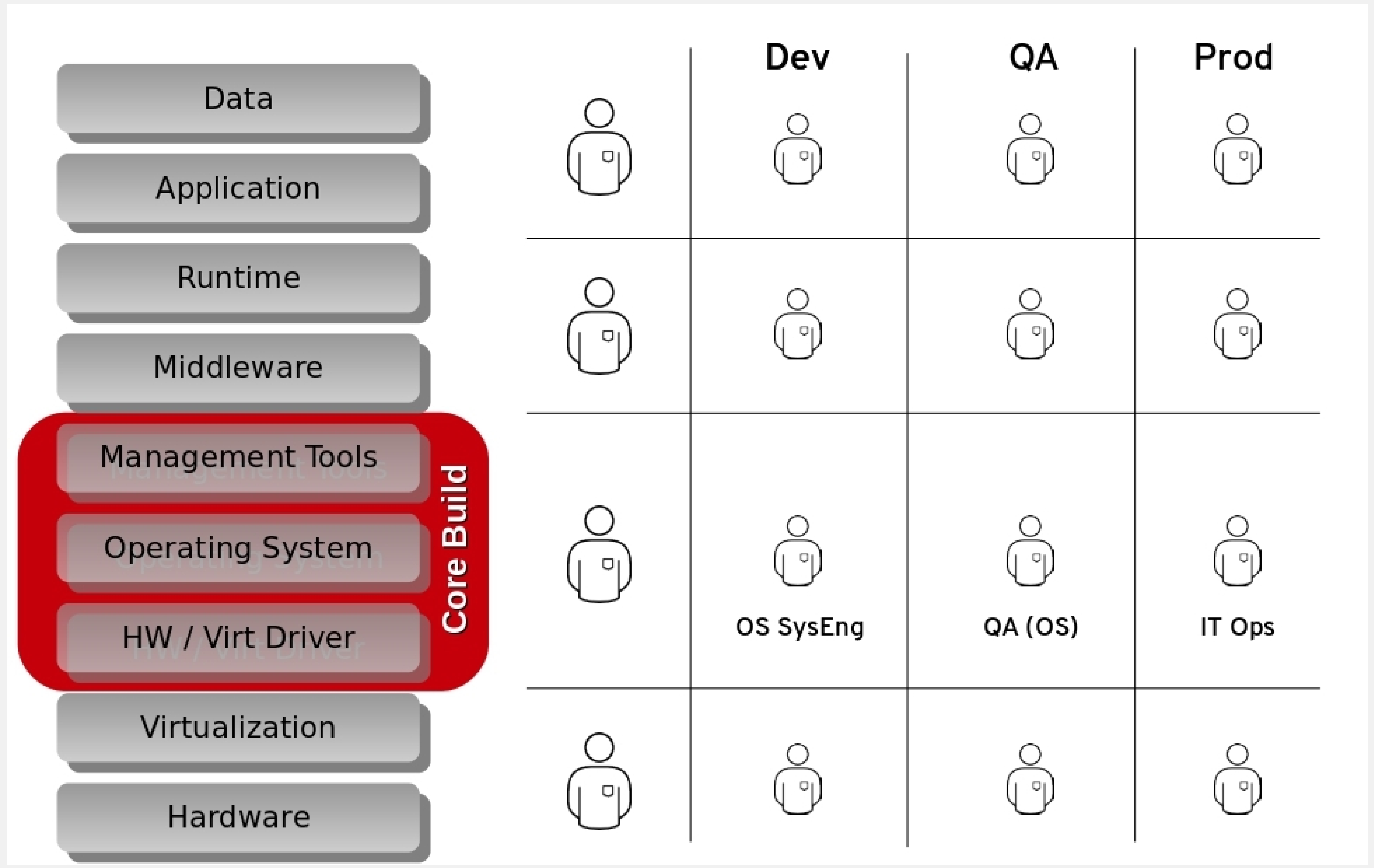
Location View

Map your IT Org & Roles

8

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

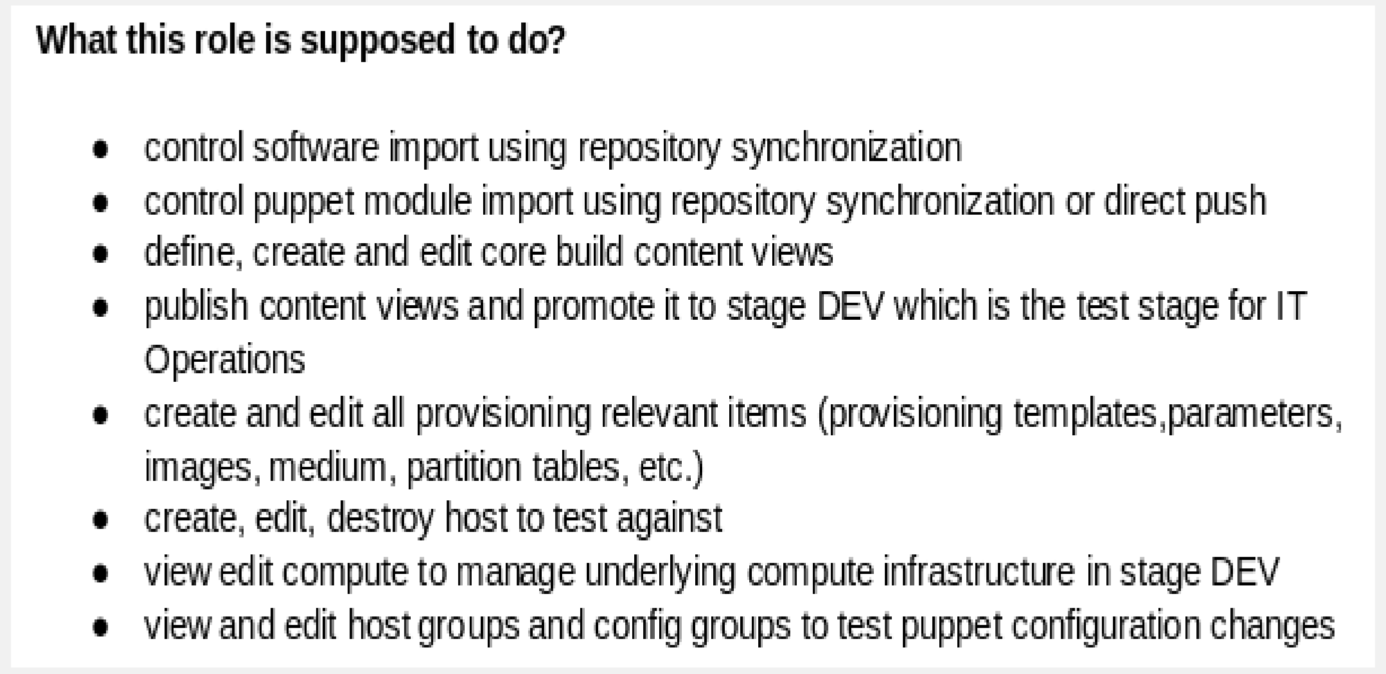
Sample Role – OS / Core Build SysEng

- Expected tasks of this role

- RBAC configuration of this role

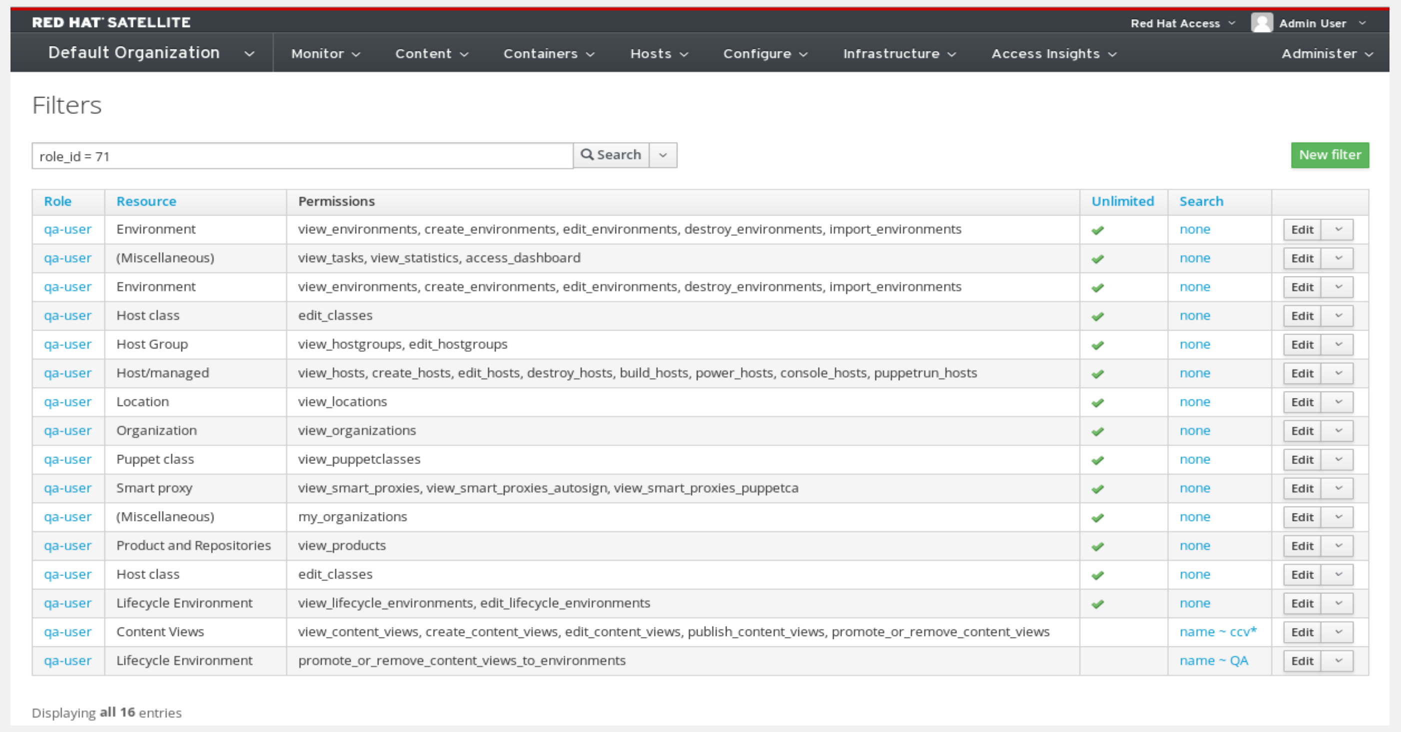
  - Predefined Manager role
  - Permissions
  - Filter

- Role creation using hammer CLI



What this role is supposed to do?

- control software import using repository synchronization
- control puppet module import using repository synchronization or direct push
- define, create and edit core build content views
- publish content views and promote it to stage DEV which is the test stage for IT Operations
- create and edit all provisioning relevant items (provisioning templates,parameters, images, medium, partition tables, etc.)
- create, edit, destroy host to test against
- view edit compute to manage underlying compute infrastructure in stage DEV
- view and edit host groups and config groups to test puppet configuration changes

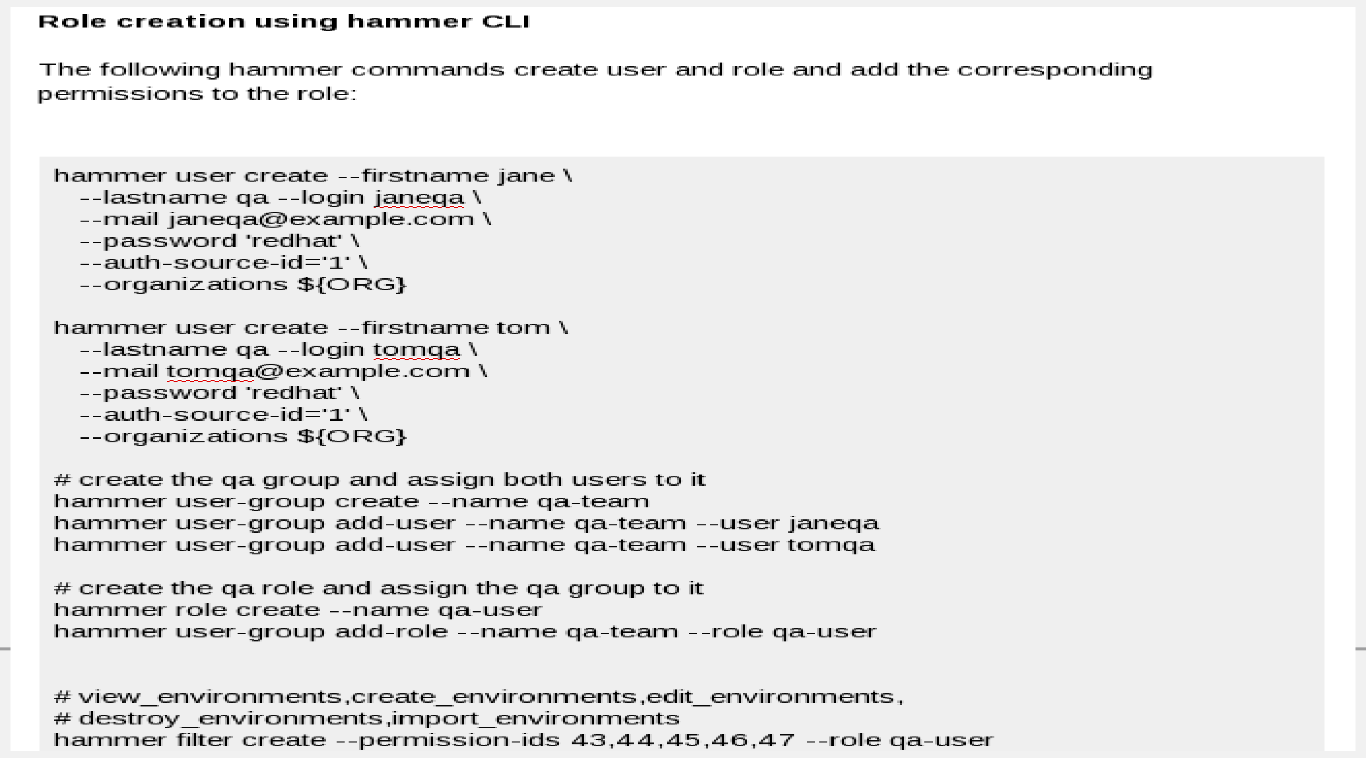


Filters

role_id = 71

Role	Resource	Permissions	Unlimited	Search	
qa-user	Environment	view_environments, create_environments, edit_environments, destroy_environments, import_environments	✓	none	Edit
qa-user	(Miscellaneous)	view_tasks, view_statistics, access_dashboard	✓	none	Edit
qa-user	Environment	view_environments, create_environments, edit_environments, destroy_environments, import_environments	✓	none	Edit
qa-user	Host class	edit_classes	✓	none	Edit
qa-user	Host Group	view_hostgroups, edit_hostgroups	✓	none	Edit
qa-user	Host/managed	view_hosts, create_hosts, edit_hosts, destroy_hosts, build_hosts, power_hosts, console_hosts, puppetrun_hosts	✓	none	Edit
qa-user	Location	view_locations	✓	none	Edit
qa-user	Organization	view_organizations	✓	none	Edit
qa-user	Puppet class	view_puppetclasses	✓	none	Edit
qa-user	Smart proxy	view_smart_proxies, view_smart_proxies_autosign, view_smart_proxies_puppetca	✓	none	Edit
qa-user	(Miscellaneous)	my_organizations	✓	none	Edit
qa-user	Product and Repositories	view_products	✓	none	Edit
qa-user	Host class	edit_classes	✓	none	Edit
qa-user	Lifecycle Environment	view_lifecycle_environments, edit_lifecycle_environments	✓	none	Edit
qa-user	Content Views	view_content_views, create_content_views, edit_content_views, publish_content_views, promote_or_remove_content_views		name ~ ccv*	Edit
qa-user	Lifecycle Environment	promote_or_remove_content_views_to_environments		name ~ QA	Edit

Displaying **all 16** entries



Role creation using hammer CLI

The following hammer commands create user and role and add the corresponding permissions to the role:

```
hammer user create --firstname jane \
  --lastname qa --login janeqa \
  --mail janeqa@example.com \
  --password 'redhat' \
  --auth-source-id='1' \
  --organizations ${ORG}

hammer user create --firstname tom \
  --lastname qa --login tomqa \
  --mail tomqa@example.com \
  --password 'redhat' \
  --auth-source-id='1' \
  --organizations ${ORG}

# create the qa group and assign both users to it
hammer user-group create --name qa-team
hammer user-group add-user --name qa-team --user janeqa
hammer user-group add-user --name qa-team --user tomqa

# create the qa role and assign the qa group to it
hammer role create --name qa-user
hammer user-group add-role --name qa-team --role qa-user


# view_environments,create_environments,edit_environments,
# destroy_environments,import_environments
hammer filter create --permission-ids 43,44,45,46,47 --role qa-user
```

9

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

10

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

10:40am – 11:40am

Security Compliance Made Easy(er): Entering SCAP Renaissance

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!

RED HAT SUMMIT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.