



Red Hat Satellite 6.2

Architecture Guide

Planning Satellite 6 Deployment

Red Hat Satellite 6.2 Architecture Guide

Planning Satellite 6 Deployment

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document explains architecture concepts of Red Hat Satellite 6 and provides recommendations for your deployment planning.

Table of Contents

PART I. SATELLITE 6 ARCHITECTURE	3
CHAPTER 1. INTRODUCTION TO RED HAT SATELLITE 6	4
1.1. SYSTEM ARCHITECTURE	4
1.2. SYSTEM COMPONENTS	8
1.3. SUPPORTED USAGE	8
1.4. SUPPORTED CLIENT ARCHITECTURES	9
1.4.1. Content Management	9
1.4.2. Host Provisioning	9
1.4.3. Configuration Management	10
CHAPTER 2. CAPSULE SERVER OVERVIEW	11
2.1. CAPSULE FEATURES	11
2.2. CAPSULE TYPES	12
2.3. CAPSULE NETWORKING	12
CHAPTER 3. HOST GROUPING CONCEPTS	15
3.1. HOST GROUP STRUCTURES	15
CHAPTER 4. CONTENT DELIVERY NETWORK (CDN) STRUCTURE	17
PART II. SATELLITE 6 DEPLOYMENT PLANNING	19
CHAPTER 5. DEPLOYMENT CONSIDERATIONS	20
5.1. SATELLITE SERVER CONFIGURATION	20
5.2. LOCATIONS AND TOPOLOGY	21
5.3. CONTENT SOURCES	22
5.4. CONTENT LIFE CYCLE	22
5.5. PROVISIONING	23
5.6. ROLE BASED AUTHENTICATION	24
5.7. ADDITIONAL TASKS	24
CHAPTER 6. COMMON DEPLOYMENT SCENARIOS	26
6.1. SINGLE LOCATION	26
6.2. SINGLE LOCATION WITH SEGREGATED SUBNETS	26
6.3. MULTIPLE LOCATIONS	26
6.4. DISCONNECTED SATELLITE	26
6.5. CAPSULE WITH EXTERNAL SERVICES	27
APPENDIX A. TECHNICAL USERS PROVIDED AND REQUIRED BY SATELLITE	28
APPENDIX B. GLOSSARY OF TERMS	29

PART I. SATELLITE 6 ARCHITECTURE

CHAPTER 1. INTRODUCTION TO RED HAT SATELLITE 6

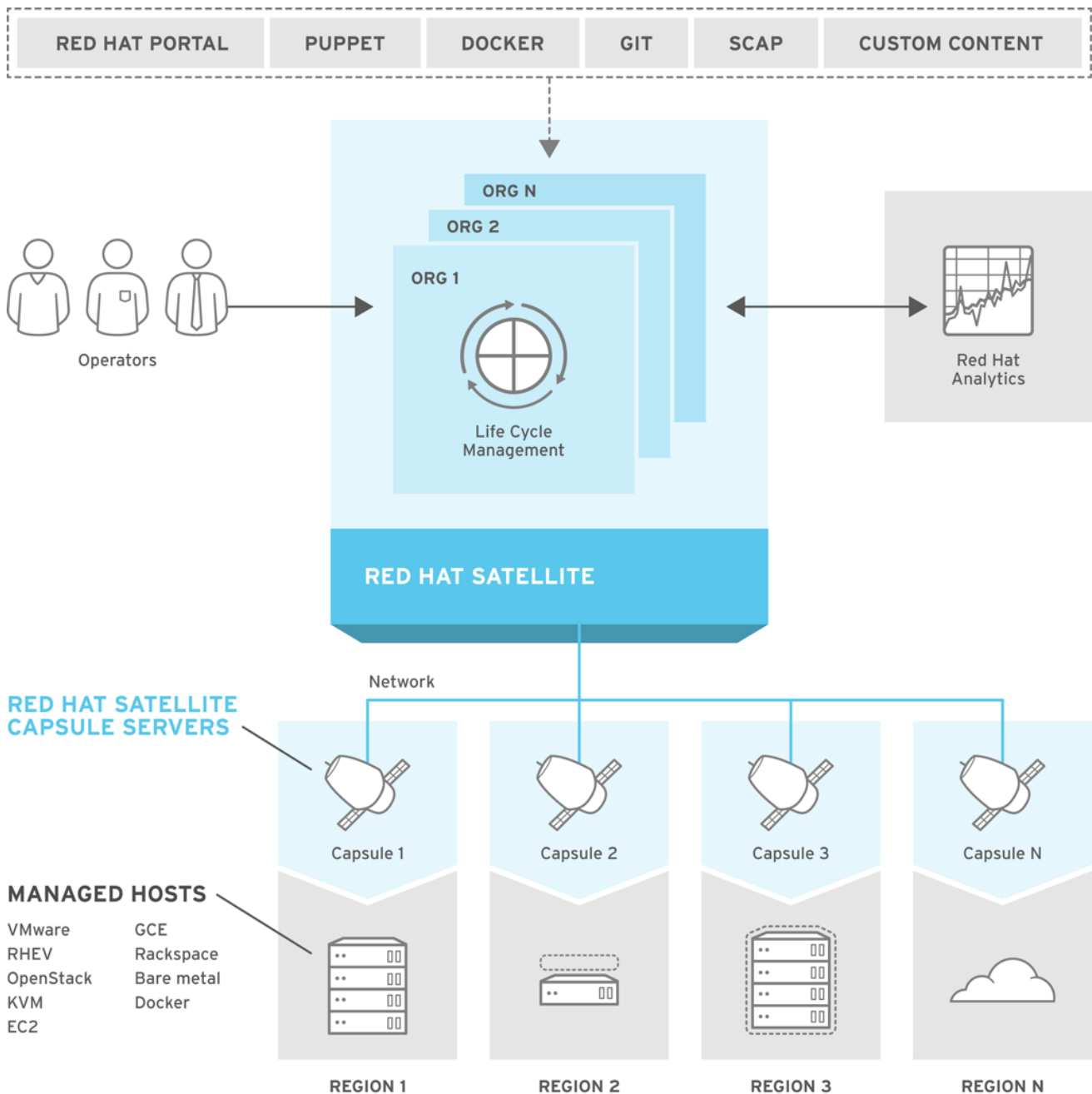
Red Hat Satellite is a system management solution that enables you to deploy, configure, and maintain your systems across physical, virtual, and cloud environments. Satellite provides provisioning, remote management and monitoring of multiple Red Hat Enterprise Linux deployments with a single, centralized tool. *Red Hat Satellite Server* synchronizes the content from Red Hat Customer Portal and other sources, and provides functionality including fine-grained life cycle management, user and group role-based access control, integrated subscription management, as well as advanced GUI, CLI, or API access.

Red Hat Satellite Capsule Server mirrors content from Red Hat Satellite Server to facilitate content federation across various geographical locations. Host systems can pull content and configuration from the Capsule Server in their location and not from the central Satellite Server. The Capsule Server also provides localized services such as Puppet Master, DHCP, DNS, or TFTP. Capsule Servers assist you in scaling Red Hat Satellite as the number of managed systems increases in your environment.

1.1. SYSTEM ARCHITECTURE

The following diagram represents the high-level architecture of Red Hat Satellite 6.

Figure 1.1. Red Hat Satellite 6 System Architecture



SATELLITE6_352601_0715

There are four stages through which content flows in this architecture:

External Content Sources

The *Red Hat Satellite Server* can consume diverse types of content from various sources. The required connection is the one with Red Hat Customer Portal, which is the primary source of software packages, errata, Puppet modules, and container images. In addition, you can use other supported content sources (Git repositories, Docker Hub, Puppet Forge, SCAP repositories) as well as your organization's internal data store.

Red Hat Satellite Server

The Red Hat Satellite Server enables you to plan and manage the content life cycle and the configuration of Capsule Servers and hosts through GUI, CLI, or API.

The Satellite Server organizes the life cycle management by using organizations as principal division units. Organizations isolate content for groups of hosts with specific requirements and administration tasks. For example, the OS build team can use a different organization than the web development

team.

The Satellite Server also contains a fine-grained authentication system to provide Satellite operators with permissions to access precisely the parts of the infrastructure that lie in their area of responsibility.

Capsule Servers

Capsule Servers mirror content from the Satellite Server to establish content sources in various geographical locations. This enables host systems to pull content and configuration from the Capsule Servers in their location and not from the central Satellite Server. The recommended minimum number of Capsule Servers is therefore given by the number of geographic regions where the organization that uses Satellite operates.

Using Content Views, you can specify the exact subset of content that the Capsule Server makes available to hosts. See [Figure 1.2, “Content Life Cycle in Red Hat Satellite 6”](#) for a closer look at life cycle management with the use of Content Views.

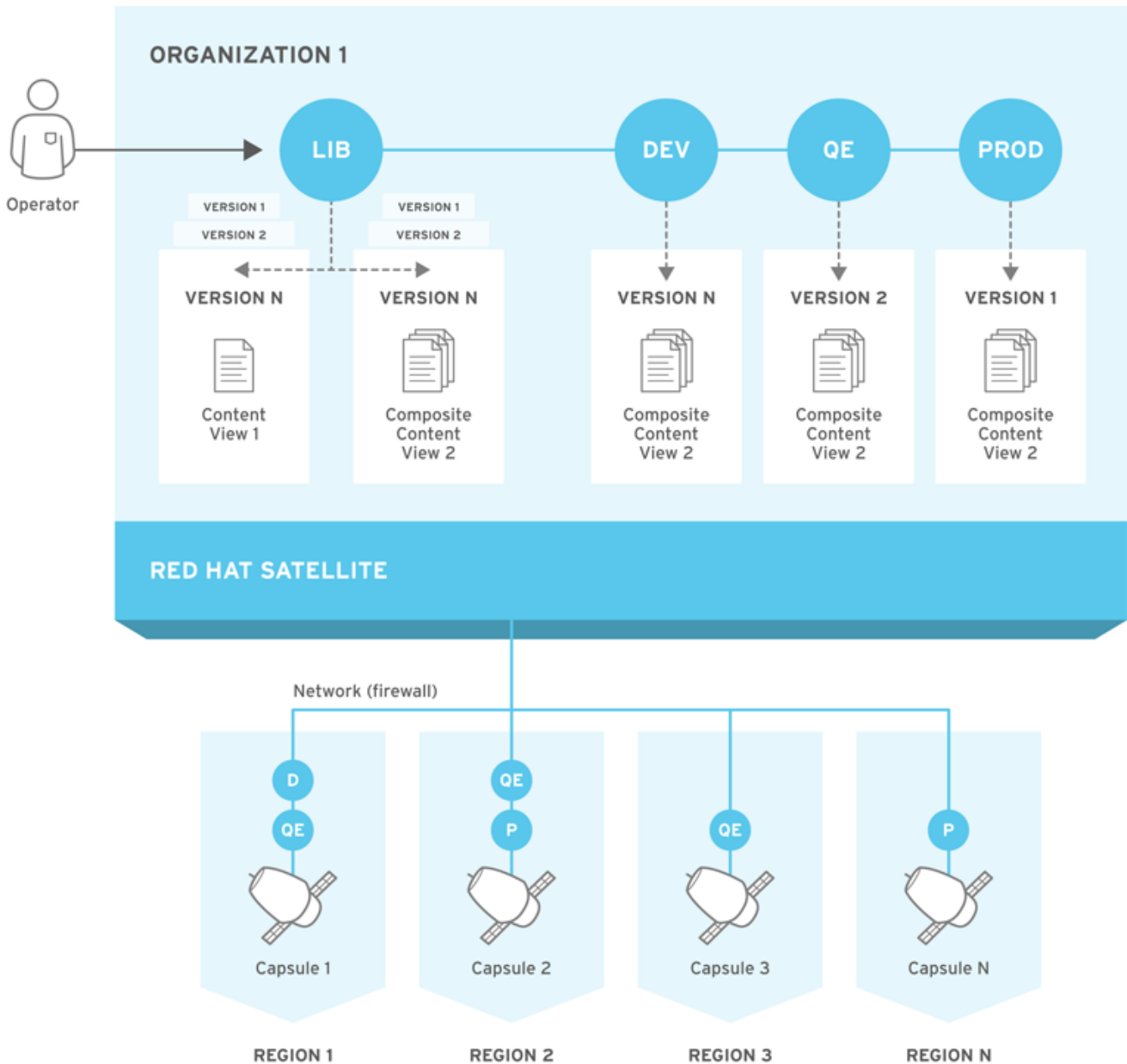
The communication between managed hosts and the Satellite Server is routed through the Capsule Server that can also manage multiple services on behalf of hosts. Many of these services use dedicated network ports, but the Capsule Server ensures that a single source IP address is used for all communications from the host to the Satellite Server, which simplifies firewall administration. For more information on Capsule Servers see [Chapter 2, Capsule Server Overview](#).

Managed Hosts

Hosts are the recipients of content from Capsule Servers. Hosts can be either physical or virtual (deployed on KVM, VMware vSphere, OpenStack, Amazon EC2, Rackspace Cloud Services, Google Compute Engine, or in a Docker container). The Satellite Server can have directly managed hosts. The base system running a Capsule Server is also a managed host of the Satellite Server.

The following diagram provides a closer look at the distribution of content from the Satellite Server to Capsules.

Figure 1.2. Content Life Cycle in Red Hat Satellite 6



SATELLITE6_352601_0615

By default, each organization has a Library of content from external sources. Content Views are subsets of content from the Library created by intelligent filtering. You can publish and promote Content Views into life cycle environments (typically Dev, QA, and Production). When creating a Capsule Server, you can choose which life cycle environments will be copied to that Capsule and made available to managed hosts.

Content Views can be combined to create Composite Content Views. It can be beneficial to have a separate Content View for a repository of packages required by an operating system and a separate one for a repository of packages required by an application. One advantage is that any updates to packages in one repository only requires republishing the relevant Content View. You can then use Composite Content Views to combine published Content Views for ease of management.

Which Content Views should be promoted to which Capsule Server depends on the Capsule's intended functionality. Any Capsule Server can run DNS, DHCP, and TFTP as infrastructure services that can be supplemented, for example, with content or configuration services.

You can update the Capsule Server by creating a new version of a Content View using synchronized content from the Library. The new Content View version is then promoted through life cycle

environments. You can also create in-place updates of Content Views. This means creating a minor version of the Content View in its current life cycle environment without promoting it from the Library. For example, if you need to apply a security erratum to a Content View used in Production, you can update the Content View directly without promoting to other life cycles. For more information on content management see the [Red Hat Satellite Content Management Guide](#).

1.2. SYSTEM COMPONENTS

Red Hat Satellite 6 consists of several open source projects which are integrated, verified, delivered and supported as Satellite 6. This information is maintained and regularly updated on the Red Hat Customer Portal, see [Satellite 6 Component Versions](#).

Red Hat Satellite 6 consists of the following open source projects:

Foreman

Foreman is an open source application used for provisioning and life cycle management of physical and virtual systems. Foreman automatically configures these systems using various methods, including kickstart and Puppet modules. Foreman also provides historical data for reporting, auditing, and troubleshooting.

Katello

Katello is a Foreman plug-in for subscription and repository management. It provides a means to subscribe to Red Hat repositories and download content. You can create and manage different versions of this content and apply them to specific systems within user-defined stages of the application life cycle.

Candlepin

Candlepin is a service within Katello that handles subscription management.

Pulp

Pulp is a service within Katello that handles repository and content management. Pulp ensures efficient storage space by not duplicating RPM packages even when requested by Content Views in different organizations.

Hammer

Hammer is a CLI tool that provides command line and shell equivalents of most Web UI functions.

REST API

Red Hat Satellite 6 includes a RESTful API service that allows system administrators and developers to write custom scripts and third-party applications that interface with Red Hat Satellite.

The terminology used in Red Hat Satellite and its upstream components is extensive, for explanation of frequent terms see [Appendix B, Glossary of Terms](#).

1.3. SUPPORTED USAGE

Each Red Hat Satellite subscription includes one supported instance of Red Hat Enterprise Linux Server. This instance should be reserved solely for the purpose of running Red Hat Satellite. Using the operating system included with Satellite to run other daemons, applications, or services within your environment is not supported.

Support for Red Hat Satellite components is described below.

Puppet

Red Hat Satellite 6 includes supported Puppet packages. The installation program allows users to install and configure Puppet Masters as a part of Red Hat Satellite Capsule Servers. A Puppet module, running on a Puppet Master on the Red Hat Satellite Server or Satellite Capsule Server, is also supported by Red Hat. For information on what versions of Puppet are supported, see the Red Hat Knowledgebase article [Satellite 6 Component Versions](#).

Red Hat supports many different scripting and other frameworks, including Puppet modules. Support for these frameworks is based on the article [How does Red Hat support scripting frameworks?](#)

Pulp

Pulp usage is only supported via the Satellite Server web UI, CLI, and API. Direct modification or interaction with Pulp's local API or database is not supported, as this can cause irreparable damage to the Red Hat Satellite 6 databases.

Foreman

Foreman can be extended using plug-ins, but only plug-ins packaged with Red Hat Satellite are supported. Red Hat does not support plug-ins in the Red Hat Satellite Optional repository. Red Hat Satellite also includes components, configuration and functionality to provision and configure operating systems other than Red Hat Enterprise Linux. While these features are included and can be employed, Red Hat supports their usage for Red Hat Enterprise Linux.

Candlepin

The only supported methods of using Candlepin are through the Red Hat Satellite 6 web UI, CLI, and API. Red Hat does not support direct interaction with Candlepin, its local API or database, as this can cause irreparable damage to the Red Hat Satellite 6 databases.

Embedded Tomcat Application Server

The only supported methods of using the embedded Tomcat application server are through the Red Hat Satellite 6 web UI, API, and database. Red Hat does not support direct interaction with the embedded Tomcat application server's local API or database.

1.4. SUPPORTED CLIENT ARCHITECTURES

1.4.1. Content Management

Supported combinations of major versions of Red Hat Enterprise Linux and hardware architectures for registering and managing hosts with Satellite 6.3. This includes the Satellite Tools Repositories.

Table 1.1. Content Management Support

Platform	Architectures
Red Hat Enterprise Linux 7	x86_64, ppc64 (BE), ppc64le, aarch64, s390x
Red Hat Enterprise Linux 6	x86_64, i386, s390x, ppc64 (BE)

1.4.2. Host Provisioning

Supported combinations of major versions of Red Hat Enterprise Linux and hardware architectures for host provisioning with Satellite 6.3.

Table 1.2. Host Provisioning Support

Platform	Architectures
Red Hat Enterprise Linux 7	x86_64
Red Hat Enterprise Linux 6	x86_64, i386

1.4.3. Configuration Management

Supported combinations of major versions of Red Hat Enterprise Linux and hardware architectures for configuration management with Satellite 6.3.

Table 1.3. Puppet 3 Support

Platform	Architectures
Red Hat Enterprise Linux 7	x86_64, ppc64 (BE), ppc64le, aarch64, s390x
Red Hat Enterprise Linux 6	x86_64, i386, s390x, ppc64 (BE)



NOTE

Usage of all Red Hat Satellite components is supported within the context of Red Hat Satellite only. Third-party usage of any components falls beyond supported usage.

CHAPTER 2. CAPSULE SERVER OVERVIEW

Capsule Servers provide **content federation** and run **localized services** to discover, provision, control, and configure hosts. You can use Capsules to extend the Satellite deployment to various geographical locations. This section contains an overview of features that can be enabled on Capsules as well as their simple classification.

For details on Capsule requirements, installation process, scalability considerations and more, see the [Red Hat Satellite Installation Guide](#).

2.1. CAPSULE FEATURES

There are two sets of features provided by Capsule Servers. You can configure the Capsule to mirror content from the Satellite Server. You can also use the Capsule to run services required for host management.

Content related features are:

- **Repository synchronization** – the content from the Satellite Server (more precisely from selected life cycle environments) is pulled to the Capsule Server for content delivery (enabled by Pulp).
- **Content delivery** – hosts configured to use the Capsule Server download content from that Capsule rather than from the central Satellite Server (enabled by Pulp).
- **Host action delivery** – Capsule Server executes scheduled actions on hosts, for example package updates (provided by the Katello Agent on the host and the Qpid Dispatch Router on the Capsule).
- **Red Hat Subscription Management (RHSM) proxy** – hosts are registered to their associated Capsule Servers rather than to the central Satellite Server or the Red Hat Customer Portal (provided by Candlepin).

Infrastructure and host management services are:

- **DHCP** – Capsule can act as a DHCP server or it can integrate with an existing solution, including ISC DHCP servers, Active Directory, and Libvirt instances.
- **DNS** – Capsule can act as a DNS server or it can integrate with an existing solution, including ISC DNS, Active Directory, or BIND.
- **TFTP** – Capsule can act as a TFTP server or integrate with any UNIX-based TFTP server.
- **Realm** – Capsule can manage Kerberos realms or domains so that hosts can join them automatically during provisioning. Capsule can integrate with an existing infrastructure including IdM, FreeIPA, and Active Directory.
- **Puppet Master** – Capsule can act as a configuration management server by running Puppet Master.
- **Puppet Certificate Authority** – Capsule can act as a Puppet CA to provide certificates to hosts.
- **Baseboard Management Controller (BMC)** – Capsule can provide power management for hosts.
- **Provisioning template proxy** – Capsule can serve provisioning templates to hosts.

- **OpenSCAP** – Capsule can perform security compliance scans on hosts.

2.2. CAPSULE TYPES

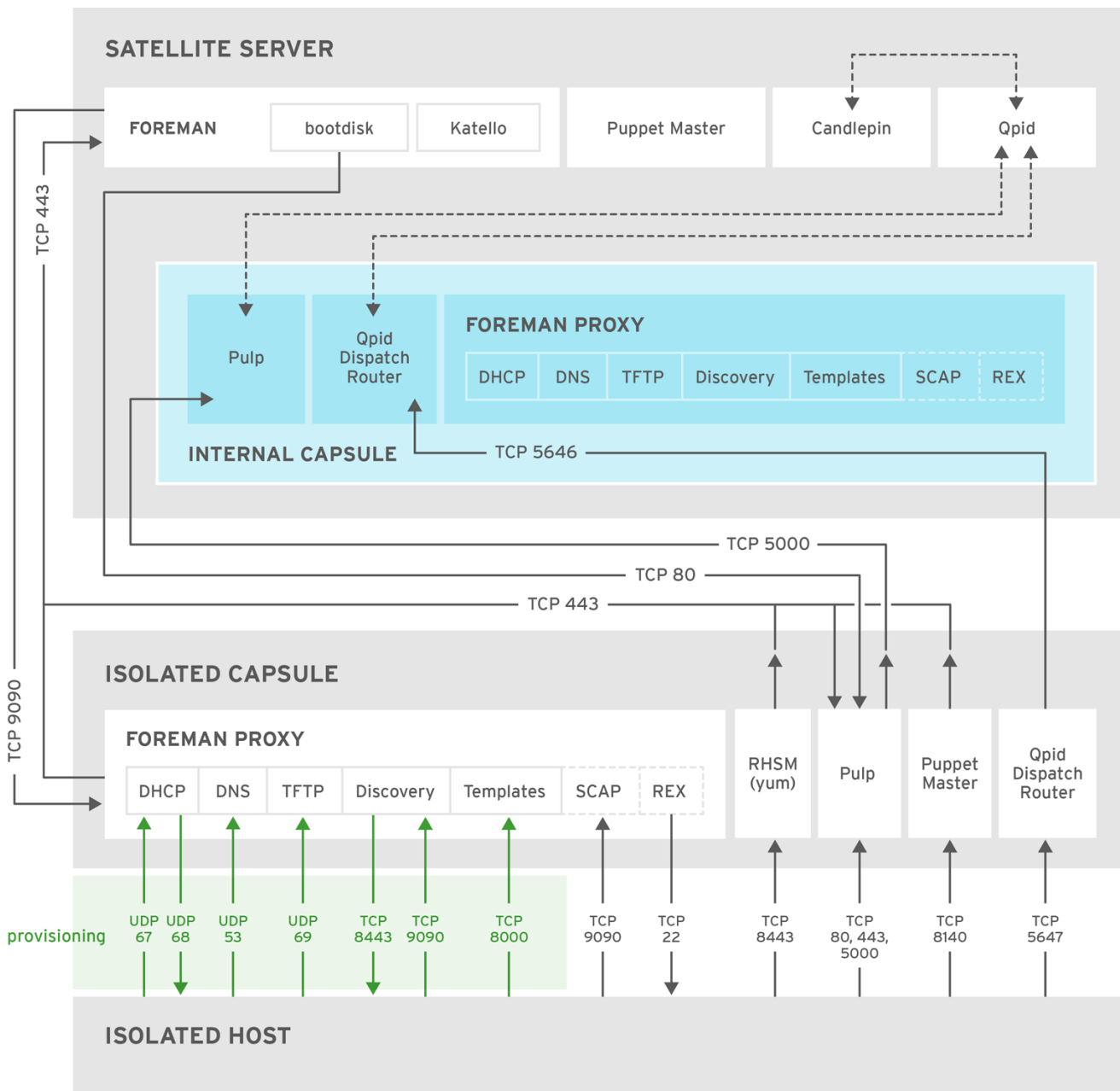
Not all Capsule features have to be enabled at once. You can configure a Capsule Server for a specific limited purpose. Some common configurations include:

- **Infrastructure Capsules** [DNS + DHCP + TFTP] – provide infrastructure services for hosts. With provisioning template proxy enabled, infrastructure Capsule has all necessary services for provisioning new hosts.
- **Content Capsules** [Pulp] – provide content synchronized from the Satellite Server to hosts.
- **Configuration Capsules** [Pulp + Puppet + PuppetCA] – provide content and run configuration services for hosts.
- **All-in-one Capsules** [DNS + DHCP + TFTP + Pulp + Puppet + PuppetCA] – provide a full set of Capsule features. All-in-one Capsules enable host isolation by providing a single point of connection for managed hosts.

2.3. CAPSULE NETWORKING

The goal of Capsule isolation is to provide a single endpoint for all of the host's network communications, so that in remote network segments, you need only open firewall ports to the Capsule itself. The following diagram shows how the Satellite components interact in the scenario with hosts connecting to an isolated Capsule.

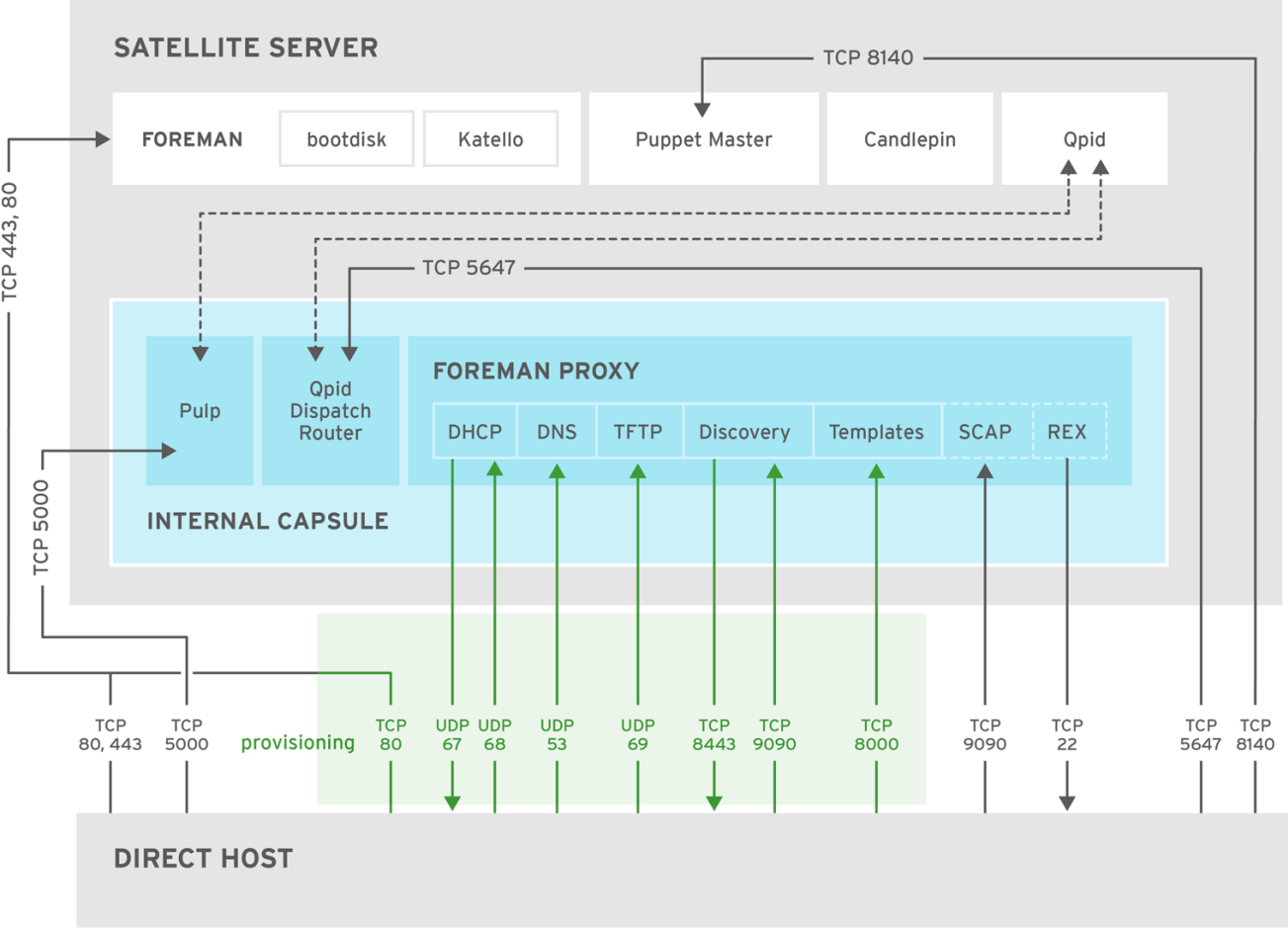
Figure 2.1. Satellite Topology with Isolated Capsule



SATELLITE_439131_0317

The following diagram shows how the Satellite components interact when hosts connect directly to the Satellite Server. Note that as the base system of an external Capsule is a Client of the Satellite, this diagram is relevant even if you do not intend to have directly connected hosts.

Figure 2.2. Satellite Topology with Internal Capsule



SATELLITE_439131_0317

The [Ports and Firewalls Requirements](#) section of the [Red Hat Satellite Installation Guide](#) contains complete instructions for configuring the host-based firewall to open the ports required. A matrix table of ports is also available in the following Knowledgebase solution [Red Hat Satellite 6.2 List of Network Ports](#).

CHAPTER 3. HOST GROUPING CONCEPTS

Apart from the physical topology of Capsule Servers, Red Hat Satellite provides several logical units for grouping hosts. Hosts that are members of those groups inherit the group configuration. For example, the simple parameters that define the provisioning environment can be applied at the following levels (for more information on the use of parameters, see [Parameters](#) in the *Red Hat Satellite Host Configuration Guide*):

Global > Organization > Location > Domain > Host group > Host

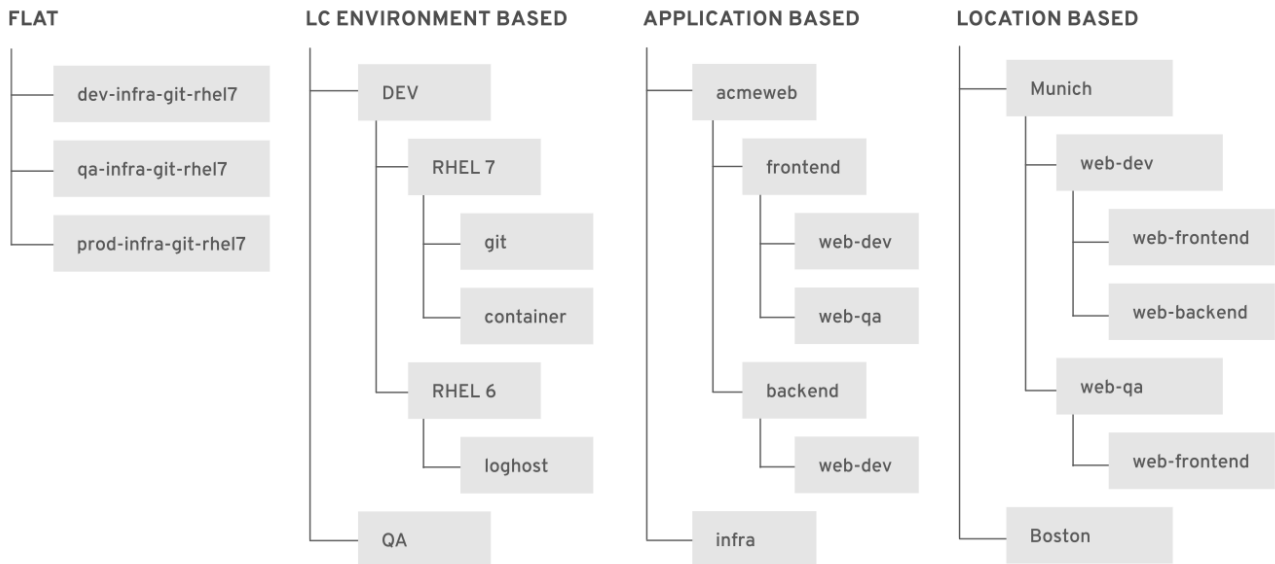
The main logical groups in Red Hat Satellite are:

- **Organizations** – the highest level logical groups for hosts. Organizations provide a strong separation of content and configuration. Each organization requires a separate subscription manifest, and can be thought of as a separate virtual instance of a Satellite Server. Avoid the use of organizations if a lower level host grouping is applicable.
- **Locations** – a grouping of hosts that should match the physical location. Locations can be used to map the network infrastructure to prevent incorrect host placement or configuration. For example, you cannot assign a subnet, domain, or compute resources directly to a Capsule Server, only to a location.
- **Host groups** – the main carriers of host definitions including assigned Puppet classes, Content View, or operating system. Find the complete list of host group parameters in the [Configuring the Provisioning Environment Chapter](#) in the *Red Hat Satellite Host Configuration Guide*. It is recommended to configure the majority of settings at the host group level instead of defining hosts directly. Configuring a new host then largely becomes a matter of adding it to the right host group. As host groups can be nested, you can create a structure that best fits your requirements (see [Section 3.1, “Host Group Structures”](#)).
- **Host collections** – a host registered to the Satellite Server for the purpose of subscription and content management is called **content host**. Content hosts can be organized into host collections, which enables performing bulk actions such as package management or errata installation.

Locations and host groups can be nested, organizations and host collections are flat.

3.1. HOST GROUP STRUCTURES

The fact that host groups can be nested to inherit parameters from each other allows for designing host group hierarchies that fit particular workflows. A well planned host group structure can help to simplify the maintenance of host settings. This section outlines four approaches to organizing host groups.

Figure 3.1. Host Group Structuring Examples

Flat Structure

The advantage of a flat structure is limited complexity, as inheritance is avoided. In a deployment with few host types, this scenario is the best option. However, without inheritance there is a risk of high duplication of settings between host groups.

Life Cycle Environment Based Structure

In this hierarchy, the first host group level is reserved for parameters specific to a life cycle environment. The second level contains operating system related definitions, and the third level contains application specific settings. Such structure is useful in scenarios where responsibilities are divided among life cycle environments (for example, a dedicated owner for the **Development**, **QA**, and **Production** life cycle stages).

Application Based Structure

This hierarchy is based on roles of hosts in a specific application. For example, it enables defining network settings for groups of back-end and front-end servers. The selected characteristics of hosts are segregated, which supports Puppet-focused management of complex configurations. However, the content views can only be assigned to host groups at the bottom level of this hierarchy.

Location Based Structure

In this hierarchy, the distribution of locations is aligned with the host group structure. In a scenario where the location (Capsule Server) topology determines many other attributes, this approach is the best option. On the other hand, this structure complicates sharing parameters across locations, therefore in complex environments with a large number of applications, the number of host group changes required for each configuration change increases significantly.

CHAPTER 4. CONTENT DELIVERY NETWORK (CDN) STRUCTURE

The Red Hat Content Delivery Network, nominally accessed via `cdn.redhat.com` is a geographically distributed series of static web servers, which contain content and errata that is designed to be consumed by systems. This content can be consumed directly (such as via a system registered via Red Hat Subscription Management) or mirrored via on premise solution, such as Red Hat Satellite 6. The Red Hat Content Delivery network is protected by x.509 certificate authentication, to ensure that only valid users can access it.

In the case of a system registered to Red Hat Subscription Management, the attached subscriptions govern which subset of the CDN the system can access. In the case of Satellite 6, the subscriptions that are attached to the subscription manifest govern which subset of the CDN the system can access.

Directory Structure of the CDN.

A CDN mirror directory structure looks like:

```
$ tree -d -L 11
├── content
│   ├── beta
│   │   └── rhel
│   │       └── server
│   │           └── 7
│   │               └── x86_64
│   │                   └── sat-tools
│   │                       └── 6
│   └── dist
│       └── rhel
│           └── server
│               └── 7
│                   ├── 7.2
│                   │   ├── x86_64
│                   │       └── kickstart
│                   └── 7Server
│                       ├── x86_64
│                       └── os
```

This directory structure is important and has the following meaning

- Top-level directory (always named content)
- Second Level Directory (What is the lifecycle of this content? Common directories include beta (for Beta code), dist (for Production Bits) and eus (For Extended Update Support bits))
- Third Level Directory (which product. Usually rhel for Red Hat Enterprise Linux)
- Fourth Level Directory (Which Variant of the product. For Red Hat Enterprise Linux this includes server, workstation, and computenode)
- Fifth Level Directory (Major version, such as 5,6, or 7)
- Sixth Level Directory (Release version such as 7.0, 7.1, and 7Server)
- Seventh Level Directory (Base architecture, such as i386 or x86_64)

- Eighth Level Directory (repository name such as kickstart, optional, rhscl, etc). Some components have additional subdirectories, and those may vary.

This directory structure is also used in the subscription manifest. We can look at a subscription manifest to determine which directories of the CDN each subscription has access to.

PART II. SATELLITE 6 DEPLOYMENT PLANNING

CHAPTER 5. DEPLOYMENT CONSIDERATIONS

This section provides an overview of general topics to be considered when planning a Red Hat Satellite 6 deployment together with recommendations and references to more specific documentation. For an example implementation based on a sample customer scenario (specific to Satellite 6.1), see the article [10 Steps to Build an SOE: How Red Hat Satellite 6 Supports Setting up a Standard Operating Environment](#).

5.1. SATELLITE SERVER CONFIGURATION

The first step to a working Satellite infrastructure is installing an instance of Red Hat Satellite Server on a dedicated Red Hat Enterprise Linux 7 Server as described the [Red Hat Satellite Installation Guide](#). Consider the [installation prerequisites](#) and [considerations for large deployments](#) outlined in the same guide.

Adding Satellite Subscription Manifests to Satellite Server

A subscription manifest is a set of encrypted files that contains your subscription information. Satellite Server uses this information to access the CDN and find what repositories are available for the associated subscription. For instructions on how to create and import a subscription manifest see [Managing Subscriptions](#) in the [Red Hat Satellite Content Management Guide](#).

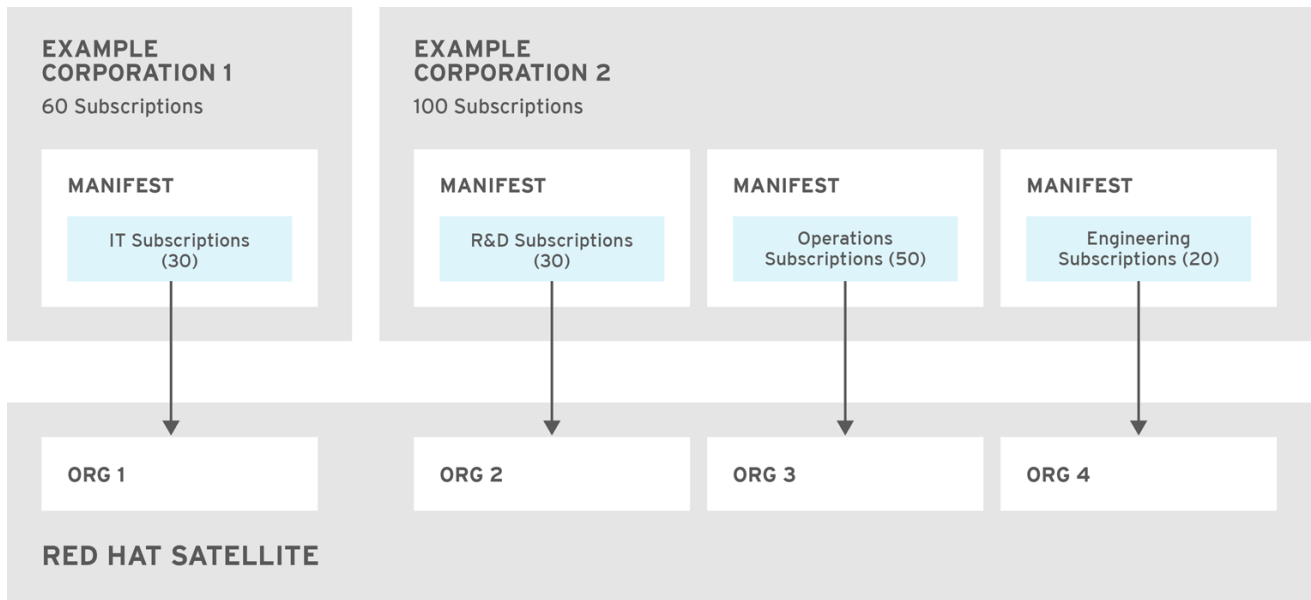
Red Hat Satellite 6 requires a single manifest for each organization configured on the Satellite. If you plan to use the Organization feature of Satellite 6 to manage separate units of your infrastructure under one Red Hat Network account, then assign subscriptions from the one account to per-organization manifests as required.

If you plan to have more than one Red Hat Network account, or if you want to manage systems belonging to another entity that is also a Red Hat Network account holder, then you and the other account holder can assign subscriptions, as required, to manifests. A customer that does not have a Satellite subscription can create a Subscription Asset Manager manifest, which can be used with Satellite, if they have other valid subscriptions. You can then use the multiple manifests in one Satellite Server to manage multiple organizations.

If you need to manage systems but do not have access to the subscriptions, then make use of the Smart Management subscriptions which entitles you to manage systems without have the entitlement to subscribe to software repositories.

The following diagram shows two Red Hat Network account holders, who want their systems to be managed by the same Satellite 6 installation. In this scenario, Example Corporation 1 can allocate any subset of their 60 subscriptions, in this example they have allocated 30, to a manifest. This can be imported into the Satellite as a distinct Organization. This allows system administrators the ability to manage Example Corporation 1's systems using Satellite 6 completely independently of Example Corporation 2's organizations (R&D, Operations, and Engineering).

Satellite Server with Multiple Manifests



SATELLITE_441823_0317

When creating a subscription manifest:

- Add the subscription for Satellite Server to the manifest if planning a disconnected or self-registered Satellite Server. This is not necessary for a connected Satellite Server that is subscribed using the Red Hat Subscription Manager utility on the base system.
- Add subscriptions for all Capsule Servers you want to create.
- Add subscriptions for all Red Hat Products you want to manage with Satellite.
- Create one manifest per organization. You can use multiple manifests and they can be from different Red Hat subscriptions.

Note that the subscription manifest can be modified and reloaded to the Satellite Server in case of any changes in your infrastructure, or when adding more subscriptions. Manifests should not be deleted. If you delete the manifest from the Red Hat Customer Portal or in the Satellite Web UI it will unregister all of your content hosts.

5.2. LOCATIONS AND TOPOLOGY

This section outlines general considerations that should help you to specify your Satellite 6 deployment scenario. The most common deployment scenarios are listed in [Chapter 6, Common Deployment Scenarios](#). The defining questions are:

- **How many Capsule Servers do I need?** – The number of geographic locations where your organization operates should translate to the number of Capsule Servers. By assigning a Capsule to each location, you decrease the load on Satellite Server, increase redundancy, and reduce bandwidth usage. Satellite Server itself can act as a Capsule (it contains an integrated Capsule by default). This can be used in single location deployments and to provision the base system's of Capsule Servers. Using the integrated Capsule to communicate with hosts in remote locations is not recommended as it can lead to suboptimal network utilization.
- **What services will be provided by Capsule Servers?** – After establishing the number of Capsules, decide what services will be enabled on each Capsule. Even though the whole stack of content and configuration management capabilities is available, some infrastructure services

(DNS, DHCP, TFTP) can be outside of a Satellite administrator's control. In such case, Capsules have to integrate with those external services (see [Section 6.5, "Capsule with External Services"](#)).

- **Is my Satellite Server required to be disconnected from the Internet?** – Disconnected Satellite is a common deployment scenario (see [Section 6.4, "Disconnected Satellite"](#)). If you require frequent updates of Red Hat content on a disconnected Satellite, plan an additional Satellite instance for inter-Satellite synchronization.
- **What compute resources do I need for my hosts?** – Apart from provisioning bare metal hosts, you can use various compute resources supported by Satellite 6. To learn about provisioning on different compute resources see the [Red Hat Satellite Provisioning Guide](#).

5.3. CONTENT SOURCES

The subscription manifest determines what Red Hat repositories are accessible from your Satellite Server. Once you enable a Red Hat repository, an associated Satellite Product is created automatically. For distributing content from custom sources you need to create products and repositories manually. Red Hat repositories are signed with GPG keys by default, and it is recommended to create GPG keys also for your custom repositories. The configuration of custom repositories depends on the type of content they hold (RPM packages, Puppet modules, Docker images, or OSTree snapshots).

Repositories configured as **yum** repositories, that contain only RPM packages, can make use of the new download policy setting to save on synchronization time and storage space. This setting enables selecting from **Immediate**, **On demand**, and **Background**. The **On demand** setting saves space and time by only downloading packages when requested by clients. The **Background** setting saves time by completing the download after the initial synchronization. For detailed instructions on setting up content sources see the [Importing Red Hat Content](#) section of the [Red Hat Satellite Content Management Guide](#).

A custom repository within the Satellite Server is in most cases populated with content from an external staging server. Such servers lie outside of the Satellite infrastructure, however, it is recommended to use a revision control system (such as Git) on these servers to have better control over the custom content.

5.4. CONTENT LIFE CYCLE

Satellite 6 provides features for precise management of the content life cycle. A **life cycle environment** represents a stage in the content life cycle, a **Content View** is a filtered set of content, and can be considered as a defined subset of content. By associating Content Views with life cycle environments, you make content available to hosts in a defined way (see [Figure 1.2, "Content Life Cycle in Red Hat Satellite 6"](#) for visualization of the process). For a detailed overview of the content management process see the [Red Hat 6 Content Management Guide](#). The following section provides general scenarios for deploying content views as well as life cycle environments.

The default life cycle environment called **Library** gathers content from all connected sources. It is not recommended to associate hosts directly with the Library as it prevents any testing of content before making it available to hosts. Instead, create a life cycle environment path that suits your content workflow. The following scenarios are common:

- **A single life cycle environment** – content from Library is promoted directly to the production stage. This approach limits the complexity but still allows for testing the content within the Library before making it available to hosts.



- **A single life cycle environment path** – both operating system and applications content is promoted through the same path. The path can consist of several stages (for example **Development, QA, Production**), which enables thorough testing but requires additional effort.



- **Application specific life cycle environment paths** – each application has a separate path, which allows for individual application release cycles. You can associate specific compute resources with application life cycle stages to facilitate testing. On the other hand, this scenario increases the maintenance complexity.



The following content view scenarios are common:

- **All in one content view** – a content view that contains all necessary content for the majority of your hosts. Reducing the number of content views is an advantage in deployments with constrained resources (time, storage space) or with uniform host types. However, this scenario limits the content view capabilities such as time based snapshots or intelligent filtering. Any change in content sources affects a proportion of hosts.
- **Host specific content view** – a dedicated content view for each host type. This approach can be useful in deployments with a small number of host types (up to 30). However, it prevents sharing content across host types as well as separation based on criteria other than the host type (for example between operating system and applications). With critical updates every content view has to be updated, which increases maintenance efforts.
- **Host specific composite content view** – a dedicated combination of content views for each host type. This approach enables separating host specific and shared content, for example you can have a dedicated content view for Puppet configuration. By including this content view into composite content views for several host types, you can update Puppet configuration with higher frequency than other host content.
- **Component based content view** – a dedicated content view for a specific application. For example a database content view can be included into several composite content views. This approach allows for greater standardization but it leads to an increased number of content views.

The optimal solution depends on the nature of your host environment. Avoid creating a large number of content views, but keep in mind that the size of a content view affects the speed of related operations (publishing, promoting). Also make sure that when creating a subset of packages for the content view, all dependencies are included as well. Note that kickstart repositories should not be added to content views, as they are used for host provisioning only.

5.5. PROVISIONING

Satellite 6 provides several features to help you automate the host provisioning, including provisioning templates, configuration management with Puppet, and host groups for standardized provisioning of

host roles. For a description of the provisioning workflow see [Understanding the Provisioning Workflow in the Red Hat Satellite 6 Provisioning Guide](#). The same guide contains instructions for provisioning on various compute resources.

5.6. ROLE BASED AUTHENTICATION

Assigning a role to a user enables controlling access to Satellite 6 components based on a set of permissions. You can think of role based authentication as a way of hiding unnecessary objects from users who are not supposed to interact with them.

There are various criteria for distinguishing among different roles within an organization. Apart from the administrator role, the following types are common:

- **Roles related to applications or parts of infrastructure** – for example, roles for owners of Red Hat Enterprise Linux as the operating system versus owners of application servers and database servers.
- **Roles related to a particular stage of the software life cycle** – for example, roles divided among the development, testing, and production phases, where each phase has one or more owners.
- **Roles related to specific tasks** – such as security manager, license manager, or Access Insights administrator.

When defining a custom role, consider the following recommendations:

- **Define the expected tasks and responsibilities** – define the subset of the Satellite infrastructure that will be accessible to the role as well as actions permitted on this subset. Think of the responsibilities of the role and how it would differ from other roles.
- **Use predefined roles whenever possible** – Satellite 6 provides a number of sample roles that can be used alone or as part of a role combination. Copying and editing an existing role can be a good start for creating a custom role.
- **Consider all affected entities** – for example, a content view promotion automatically creates new Puppet Environments for the particular life cycle environment and content view combination. Therefore, if a role is expected to promote content views, it also needs permissions to create and edit Puppet Environments.
- **Consider areas of interest** – even though a role has a limited area of responsibility, there might be a wider area of interest. Therefore, you can grant the role a read only access to parts of Satellite infrastructure that influence its area of responsibility. This allows users to get earlier access to information about potential upcoming changes.
- **Add permissions step by step** – test your custom role to make sure it works as intended. A good approach in case of problems is to start with a limited set of permissions, add permissions step by step, and test continuously.

Find instructions on defining roles and assigning them to users in the [Red Hat Satellite Server Administration Guide](#). The same guide contains information on configuring external authentication sources.

5.7. ADDITIONAL TASKS

This section provides a short overview of selected Satellite capabilities that can be used for automating certain tasks or extending the core usage of Satellite 6:

- **Importing existing hosts** – if you have existing hosts that have not been managed by Satellite 6 in the past, you can import those hosts to the Satellite Server. This procedure is usually a step in transitioning from Red Hat Satellite 5, see the [Red Hat Satellite Transition Guide](#) for detailed documentation. A high level overview of the transition process is available in Red Hat Knowledgebase article [Transitioning from Red Hat Satellite 5 to Satellite 6](#).
- **Discovering bare metal hosts** – the Satellite 6 Discovery plug-in enables automatic bare-metal discovery of unknown hosts on the provisioning network. These new hosts register themselves to the Satellite Server and the Puppet agent on the client uploads system facts collected by Facter, such as serial ID, network interface, memory, and disk information. After registration you can initialize provisioning of those discovered hosts. For details, see [Discovering Bare-metal Hosts on Satellite](#) in the *Red Hat Satellite Host Configuration Guide*.
- **Backup management** – procedures for backup and disaster recovery of Satellite Server are available (see [Backup and Disaster Recovery](#) in the *Red Hat Satellite Server Administration Guide*). Using remote execution, you can also configure recurring backup tasks on managed hosts. For more information on remote execution see [Running Jobs on Satellite Hosts](#) in the *Red Hat Satellite Host Configuration Guide*.
- **Security management** – Satellite 6 supports security management in various ways, including update and errata management, OpenSCAP integration for system verification, update and security compliance reporting, and fine grained role based authentication. Find more information on errata management and OpenSCAP concepts in the [Red Hat Satellite Host Configuration Guide](#).
- **Incident management** – Satellite 6 supports the incident management process by providing a centralized overview of all systems including reporting and email notifications. Detailed information on each host is accessible from the Satellite Server, including the event history of recent changes. Satellite 6 is also integrated with the [Red Hat Insights](#).
- **Scripting with Hammer and API** – Satellite 6 provides a command line tool called Hammer that provides a CLI equivalent to the majority of web UI procedures. In addition, you can use the access to the Satellite API to write automation scripts in a selected programming language. For more information see the [Red Hat Hammer CLI Guide](#) and [Red Hat API Guide](#).

CHAPTER 6. COMMON DEPLOYMENT SCENARIOS

This section provides a brief overview of common deployment scenarios for Red Hat Satellite. Note that many variations and combinations of the following layouts are possible.

6.1. SINGLE LOCATION

An *integrated Capsule* is a virtual Capsule Server that is created by default in Satellite Server during the installation process. This means Satellite Server can be used to provision directly connected hosts for Satellite deployment in a single geographical location, therefore only one physical server is needed. The base systems of isolated Capsules can be directly managed by Satellite Server, however it is not recommended to use this layout to manage other hosts in remote locations.

6.2. SINGLE LOCATION WITH SEGREGATED SUBNETS

Your infrastructure might require multiple isolated subnets even if Red Hat Satellite is deployed in a single geographic location. This can be achieved for example by deploying multiple Capsule Servers with DHCP and DNS services, but the recommended way is to create segregated subnets using a single Capsule. This Capsule is then used to manage hosts and compute resources in those segregated networks to ensure they only have to access the Capsule for provisioning, configuration, errata, and general management. For more information on configuring subnets see the [Red Hat Satellite Host Configuration Guide](#).

6.3. MULTIPLE LOCATIONS

It is recommended to create at least one Capsule Server per geographic location. This practice can save bandwidth since hosts obtain content from a local Capsule Server. Synchronization of content from remote repositories is done only by the Capsule, not by each host in a location. In addition, this layout makes the provisioning infrastructure more reliable and easier to configure. See [Figure 1.1, “Red Hat Satellite 6 System Architecture”](#) for an illustration of this approach.

6.4. DISCONNECTED SATELLITE

In high security environments where hosts are required to function in a closed network disconnected from the Internet, Red Hat Satellite can provision systems with the latest security updates, errata, packages and other content. In such case, the Satellite Server does not have direct access to the Internet, but the layout of other infrastructure components is not affected. For information on how to install or upgrade a disconnected Satellite see the [Red Hat Satellite Installation Guide](#).

There are two options for importing content to a disconnected Satellite Server:

- **Disconnected Satellite with Content ISO** – in this setup, you download ISO images with content from the Red Hat Customer Portal and extract them to the Satellite Server or a local web server. The content on Satellite Server is then synchronized locally. This allows for complete network isolation of the Satellite Server, however, the release frequency of content ISO images is around six weeks and not all product content is included (today only Red Hat Enterprise Linux and layered products such as RHEL-OSP7, RHDS, and Red Hat Enterprise Linux for Real Time). For instructions on how to import content ISOs to a disconnected Satellite, see [Importing Content ISOs into a Disconnected Satellite](#) in the *Red Hat Satellite Content Management Guide*.
- **Disconnected Satellite with Inter-Satellite Synchronization** – in this setup, you install a connected Satellite Server and export content from it to populate a disconnected Satellite using some storage device. This allows for exporting both Red Hat provided and custom content at the

frequency you choose, but requires deploying an additional server with a separate subscription. For instructions on how to configure Inter-Satellite synchronization, see [Synchronizing Content Between Satellite Servers](#) in the *Red Hat Satellite Content Management Guide*.

The above methods for importing content to a disconnected Satellite Server can also be used to speed up the initial population of a connected Satellite.

6.5. CAPSULE WITH EXTERNAL SERVICES

You can configure a Capsule Server (integrated or standalone) to use external DNS, DHCP, or TFTP service. If you already have a server that provides these services in your environment, you can integrate it with your Satellite deployment. For information on how to configure a Capsule with external services, see [Configuring External Services](#) in the *Red Hat Satellite Installation Guide*.

APPENDIX A. TECHNICAL USERS PROVIDED AND REQUIRED BY SATELLITE

During the installation of Satellite, system accounts are created. They are used to manage files and process ownership of the components integrated into Satellite. Some of these accounts have fixed UIDs while others take the next available UID on the system instead. In order to control the UIDs assigned to the various accounts, it is possible to fix the UID by predefining those accounts. Because some of the accounts have hard-coded UIDs, it is not possible to do this with all accounts created during Satellite installation.

The following table provides an overview of all the accounts created by Satellite during installation. Accounts marked with **flex UID** are allowed to be pre-defined with a custom UID before the installation of Satellite.

Red Hat does not recommend changing any parameter or value of a given account other than the UID, because fields such as home or shell are requirements for Satellite to work correctly.

Table A.1. Technical Users Provided and Required by Satellite

Username	UID	Flex UID	Home	Shell
qpidd	N/A	yes	/var/lib/qpidd	/sbin/nologin
foreman	N/A	yes	/usr/share/foreman	/sbin/nologin
unbound	N/A	yes	/etc/unbound	/sbin/nologin
foreman-proxy	N/A	yes	/usr/share/foreman-proxy	/sbin/nologin
puppet	52	no	/var/lib/puppet	/sbin/nologin
postgres	26	no	/var/lib/pgsql	/bin/bash
mongodb	184	no	/var/lib/mongodb	/sbin/nologin
apache	48	no	/usr/share/httpd	/sbin/nologin
hsqldb	96	no	/var/lib/hsqldb	/sbin/nologin
tomcat	91	no	/usr/share/tomcat	/bin/nologin
qdrouterd	N/A	yes	N/A	/sbin/nologin
saslauth	76	yes	N/A	/sbin/nologin

APPENDIX B. GLOSSARY OF TERMS

This glossary documents various terms used in relation to Red Hat Satellite 6.

Activation Key

A token for host registration and subscription attachment. Activation keys define subscriptions, products, content views, and other parameters to be associated with a newly created host.

Answer File

A configuration file that defines settings for an installation scenario. Answer files are defined in the YAML format and stored in the `/etc/foreman-installer/scenarios.d/` directory.

ARF Report

The result of an OpenSCAP audit. Summarizes the security compliance of hosts managed by Red Hat Satellite.

Audits

Provide a report on changes made by a specific user. Audits can be viewed in the Satellite web UI under **Monitor > Audits**

Baseboard Management Controller (BMC)

Enables remote power management of bare-metal hosts. In Satellite 6, you can create a BMC interface to manage selected hosts.

Boot Disk

An ISO image used for PXE-less provisioning. This ISO enables the host to connect to the Satellite Server, boot the installation media, and install the operating system. There are several kinds of boot disks: **host image**, **full host image**, **generic image**, and **subnet image**.

Capsule (Capsule Server)

An additional server that can be used in a Red Hat Satellite 6 deployment to facilitate content federation and distribution (act as a Pulp node), and to run other localized services (Puppet Master, **DHCP**, **DNS**, **TFTP**, and more). Capsules are useful for Satellite deployment across various geographical locations. In upstream Foreman terminology, Capsule is referred to as Smart Proxy.

Catalog

A document that describes the desired system state for one specific host managed by Puppet. It lists all of the resources that need to be managed, as well as any dependencies between those resources. Catalogs are compiled by a Puppet Master from Puppet manifests and data from Puppet Agents.

Candlepin

A service within Katello responsible for subscription management.

Compliance Policy

Refers to a scheduled task executed on the Satellite Server that checks the specified hosts for compliance against SCAP content.

Compute Profile

Specifies default attributes for new virtual machines on a compute resource.

Compute Resource

A virtual or cloud infrastructure, which Red Hat Satellite 6 uses for deployment of hosts and systems. Examples include Red Hat Enterprise Virtualization, OpenStack, EC2, and VMWare.

Container (Docker Container)

An isolated application sandbox that contains all runtime dependencies required by an application. Satellite 6 supports container provisioning on a dedicated compute resource.

Container Image

A static snapshot of the container's configuration. Satellite 6 supports various methods of importing container images as well as distributing images to hosts through content views.

Content

A general term for everything Satellite distributes to hosts. Includes software packages (RPM files), Puppet Modules, Docker images, or OSTree snapshots. Content is synchronized into the Library and then promoted into life cycle environments using content views so that they can be consumed by hosts.

Content Delivery Network (CDN)

The mechanism used to deliver Red Hat content to the Satellite Server.

Content Host

The part of a host that manages tasks related to content and subscriptions.

Content View

A subset of Library content created by intelligent filtering. Once a content view is published, it can be promoted through the life cycle environment path, or modified using incremental upgrades. Content views are a refinement of the combination of channels and cloning from Red Hat Satellite 5.

Discovered Host

A bare-metal host detected on the provisioning network by the Discovery plug-in.

Discovery Image

Refers to the minimal operating system based on Red Hat Enterprise Linux that is PXE-booted on hosts to acquire initial hardware information and to communicate with the Satellite Server before starting the provisioning process.

Discovery Plug-in

Enables automatic bare-metal discovery of unknown hosts on the provisioning network. The plug-in consists of three components: services running on the Satellite Server and the Capsule Server, and the Discovery image running on host.

Discovery Rule

A set of predefined provisioning rules which assigns a host group to discovered hosts and triggers provisioning automatically.

Docker Tag

A mark used to differentiate container images, typically by the version of the application stored in the image. In the Satellite 6 web UI, you can filter images by tag under **Content > Docker Tags**.

ERB

Embedded Ruby (ERB) is a template syntax used in provisioning and job templates.

Errata

Updated RPM packages containing security fixes, bug fixes, and enhancements. In relationship to a host, erratum is **applicable** if it updates a package installed on the host and **installable** if it is present in the host's content view (which means it is accessible for installation on the host).

External Node Classifier

A Puppet construct that provides additional data for a Puppet Master to use when configuring hosts. Red Hat Satellite 6 acts as an External Node Classifier to Puppet Masters in a Satellite deployment.

Facter

A program that provides information (facts) about the system on which it is run; for example, Facter can report total memory, operating system version, architecture, and more. Puppet modules enable specific configurations based on host data gathered by Facter.

Facts

Host parameters such as total memory, operating system version, or architecture. Facts are reported by Facter and used by Puppet.

Foreman

The Red Hat Satellite 6 component mainly responsible for provisioning and content life cycle management. Foreman is the main upstream counterpart of Red Hat Satellite 6.

Foreman Hook

An executable that is automatically triggered when an orchestration event occurs, such as when a host is created or when provisioning of a host has completed.

Full Host Image

A boot disk used for PXE-less provisioning of a specific host. The full host image contains an embedded Linux kernel and init RAM disk of the associated operating system installer.

Generic Image

A boot disk for PXE-less provisioning that is not tied to a specific host. The generic image sends the host's MAC address to the Satellite Server, which matches it against the host entry.

Hammer

A command line tool for managing Red Hat Satellite 6. You can execute Hammer commands from the command line or utilize them in scripts. Hammer also provides an interactive shell.

Host

Refers to any system, either physical or virtual, that Red Hat Satellite 6 manages.

Host Collection

A user defined group of one or more Hosts used for bulk actions such as errata installation. Equivalent to a Satellite 5 System Group.

Host Group

A template for building a host. Host groups hold shared parameters, such as subnet or life cycle environment, that are inherited by host group members. Host groups can be nested to create a hierarchical structure.

Host Image

A boot disk used for PXE-less provisioning of a specific host. The host image only contains the boot files necessary to access the installation media on the Satellite Server.

Incremental Upgrade (of a Content View)

The act of creating a new (minor) content view version in a life cycle environment. Incremental upgrades provide a way to make in-place modification of an already published content view. Useful for rapid updates, for example when applying security errata.

Job

A command executed remotely on a host from the Satellite Server. Every job is defined in a job template. Similar to remote command in Satellite 5.

Job Template

Defines properties of a job.

Katello

A Foreman plug-in responsible for subscription and repository management.

Lazy Sync

The ability to change a **yum** repository's default download policy of **Immediate** to **On Demand** or **Background**. The **On Demand** setting saves storage space and synchronization time by only downloading the packages when requested by a client, and the **Background** setting saves synchronization time by downloading packages after synchronizing the repository's metadata.

Location

A collection of default settings that represent a physical place.

Library

A container for content from all synchronized repositories on the Satellite Server. The primary life cycle environment existing by default for each organization, the root of every life cycle environment path and the source of content for every content view.

Life Cycle Environment

A container for content view versions consumed by the content hosts. A Life Cycle Environment represents a step in the life cycle environment path. Content moves through life cycle environments by publishing and promoting content views.

Life Cycle Environment Path

A sequence of life cycle environments through which the content views are promoted. You can promote a content view through a typical promotion path; for example, from development to test to production. Channel cloning implements this concept in Red Hat Satellite 5.

Manifest (Subscription Manifest)

A mechanism for transferring subscriptions from Red Hat Customer Portal to Red Hat Satellite 6. This is similar in function to certificates used with Red Hat Satellite 5.

OpenSCAP

A project implementing security compliance auditing according to the Security Content Automation Protocol (SCAP). OpenSCAP is integrated in Satellite 6 to provide compliance auditing for managed hosts.

Organization

An isolated collection of systems, content, and other functionality within a Satellite 6 deployment.

OSTree

A tool for managing bootable, immutable, versioned file system trees. Satellite 6 supports mirroring OSTree snapshots as well as distributing them in content views.

Parameter

Defines the behavior of Red Hat Satellite components during provisioning. Depending on the parameter scope, we distinguish between global, domain, host group, and host parameters. Depending on the parameter complexity, we distinguish between simple parameters (key-value pair) and smart parameters (conditional arguments, validation, overrides).

Parametrized Class (Smart Class Parameter)

A parameter created by importing a class from Puppet Master.

Permission

Defines an action related to a selected part of Satellite infrastructure (resource type). Each resource type is associated with a set of permissions, for example the **Architecture** resource type has the following permissions: **view_architectures**, **create_architectures**, **edit_architectures**, and **destroy_architectures**. You can group permissions into roles and associate them with users or user groups.

Product

A collection of content repositories. Products are either provided by Red Hat CDN or created by the Satellite administrator to group custom repositories.

Promote (a Content View)

The act of moving a content view from one life cycle environment to another.

Provisioning Template

Defines host provisioning settings. Provisioning templates can be associated with host groups, life cycle environments, or operating systems. In Satellite 6 they provide similar functionality to Kickstart Profiles and Cobbler Snippets in Red Hat Satellite 5.

Publish (a Content View)

The act of making a content view version available in a life cycle environment and usable by hosts.

Pulp

A service within Katello responsible for repository and content management.

Pulp Node

A Capsule Server component that mirrors content. This is similar to the Red Hat Satellite 5 Proxy. The main difference is that content can be staged on the Pulp Node before it is used by a host.

Puppet

The configuration management component of Satellite 6.

Puppet Agent

A service running on a host that applies configuration changes to that host.

Puppet Environment

An isolated set of Puppet Agent nodes that can be associated with a specific set of Puppet Modules.

Puppet Manifest

Refers to Puppet scripts with file names using the **.pp** extension.

Puppet Master

A Capsule Server component that provides Puppet manifests to hosts for execution by the Puppet Agent.

Puppet Module

A self-contained bundle of code (Puppet manifests) and data (facts) that you can use to manage resources such as users, files, and services.

Recurring Logic

A job executed automatically according to a schedule. In the Satellite 6 web UI, you can view those jobs under **Monitor > Recurring logics**.

Registry

An archive of container images. Satellite 6 supports importing images from local and external registries. Satellite itself can act as an image registry for hosts. However, hosts cannot push changes back to the registry.

Red Hat Access Insights

A module providing access to selected Red Hat Customer Portal services from the Satellite web UI.

Repository

Provides storage for a collection of content.

Resource Type

Refers to a part of Satellite infrastructure, for example host, capsule, or architecture. Used in permission filtering.

Role

Specifies a collection of permissions that are applied to a set of resources, such as hosts. Roles can be assigned to users and user groups. Satellite provides a number of predefined roles.

SCAP content

A file containing the configuration and security baseline against which hosts are checked. Used in compliance policies.

Scenario

A set of predefined settings for the Satellite CLI installer. Scenario defines the type of installation, for example to install the Capsule Server execute **satellite-installer --scenario capsule**. Every scenario has its own answer file to store the scenario settings.

Smart Proxy

A Capsule Server component that can integrate with external services, such as **DNS** or **DHCP**. In upstream Foreman terminology, Smart Proxy is a synonym of Capsule.

Smart Variable

A configuration value used by classes in Puppet modules.

Standard Operating Environment (SOE)

A controlled version of the operating system on which applications are deployed.

Subnet Image

A type of generic image for PXE-less provisioning that communicates through the Capsule Server.

Subscription

An entitlement for receiving content and service from Red Hat.

Synchronization

Refers to mirroring content from external resources into the Red Hat Satellite 6 Library.

Synchronization Plan

Provides scheduled execution of content synchronization.

Task

A background process executed on the Satellite or Capsule Server, such as repository synchronization or content view publishing. You can monitor the task status in the Satellite web UI under **Monitor > Tasks**.

Trend

A means of tracking changes in specific parts of Satellite 6 infrastructure. Configure trends in Satellite web UI under **Monitor > Trends**.

User Group

A collection of roles which can be assigned to a collection of users. This is similar to a Role in Red Hat Satellite 5.

User

Anyone registered to use Red Hat Satellite. Authentication and authorization is possible through built-in logic, through external resources (LDAP, Identity Management, or Active Directory), or with Kerberos.

virt-who

An agent for retrieving IDs of virtual machines from the hypervisor. When used with Satellite 6, virt-who reports those IDs to the Satellite Server so that it can provide subscriptions for hosts provisioned on virtual machines.