Red Hat Satellite 6.14

Satellite Overview, Concepts, and Deployment Considerations

Explore the Satellite architecture and plan Satellite deployment

Last Updated: 2024-04-11

Red Hat Satellite 6.14 Satellite Overview, Concepts, and Deployment Considerations

Explore the Satellite architecture and plan Satellite deployment

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

Legal Notice

Abstract

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

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better.

You can submit feedback by filing a ticket in Bugzilla:

1. Navigate to the Bugzilla website.
2. In the **Component** field, use **Documentation**.
3. In the **Description** field, enter your suggestion for improvement. Include a link to the relevant parts of the documentation.
4. Click **Submit Bug**.

PART I. SATELLITE ARCHITECTURE

CHAPTER 1. INTRODUCTION TO RED HAT SATELLITE

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. *Satellite Server* synchronizes the content from Red Hat Customer Portal and other sources, and provides functionality including fine-grained lifecycle management, user and group role-based access control, integrated subscription management, as well as advanced GUI, CLI, or API access.

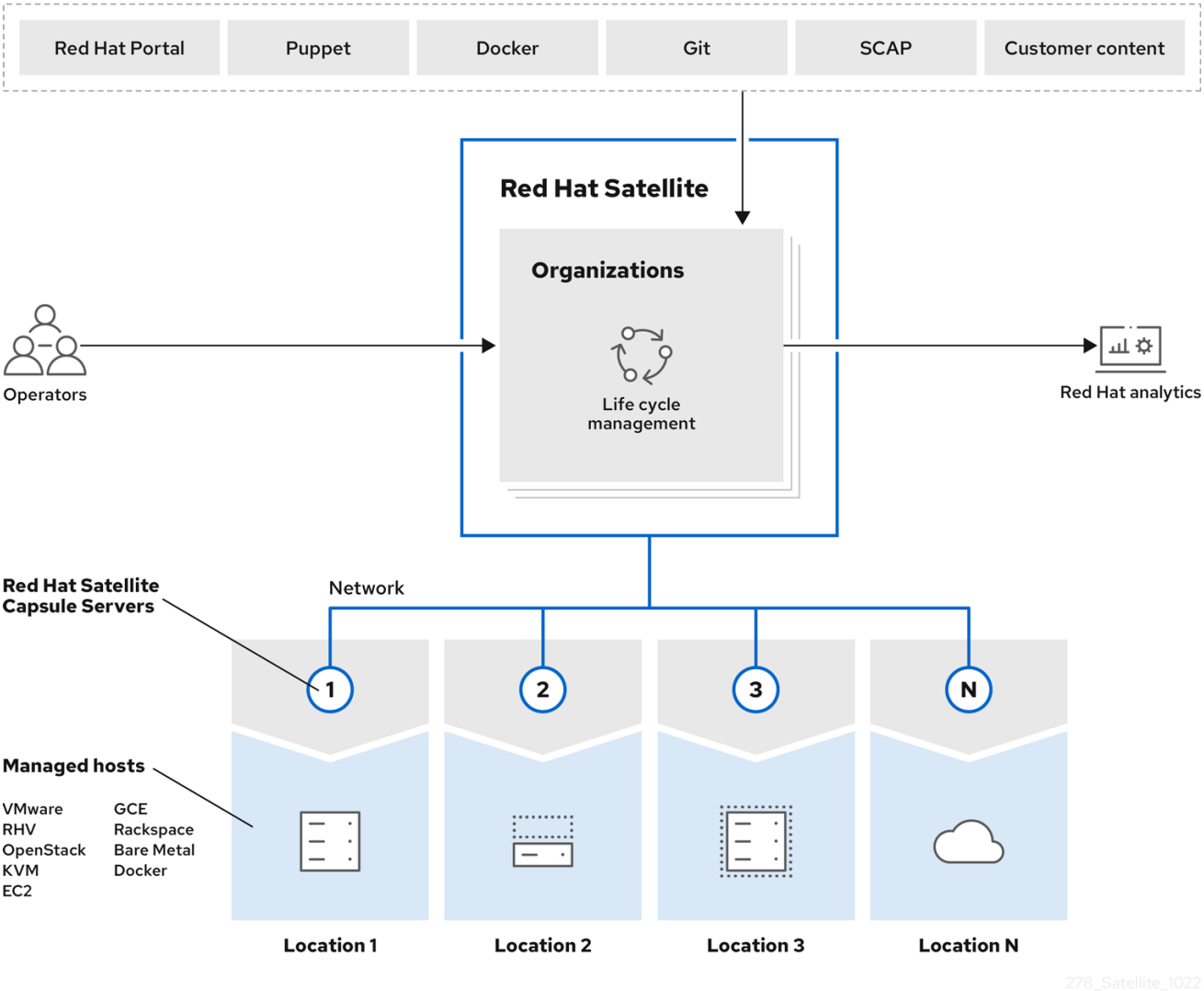
Capsule Server mirrors content from Satellite Server to facilitate content federation across various geographical locations. Host systems can pull content and configuration from Capsule Server in their location and not from the central Satellite Server. Capsule Server also provides localized services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

Capsule Servers decrease the load on the central server, increase redundancy, and reduce bandwidth usage. For more information, see Chapter 2, *Capsule Server Overview*.

1.1. SYSTEM ARCHITECTURE

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

Figure 1.1. Red Hat Satellite System Architecture



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

External Content Sources

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

Satellite Server

The Satellite Server enables you to plan and manage the content lifecycle and the configuration of Capsule Servers and hosts through GUI, CLI, or API.
Satellite Server organizes the lifecycle 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.

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 Satellite Server to establish content sources in various geographical locations. This enables host systems to pull content and configuration from 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 Capsule Server makes available to hosts. See Figure 1.2, "Content Lifecycle in Red Hat Satellite" for a closer look at lifecycle management with the use of Content Views.

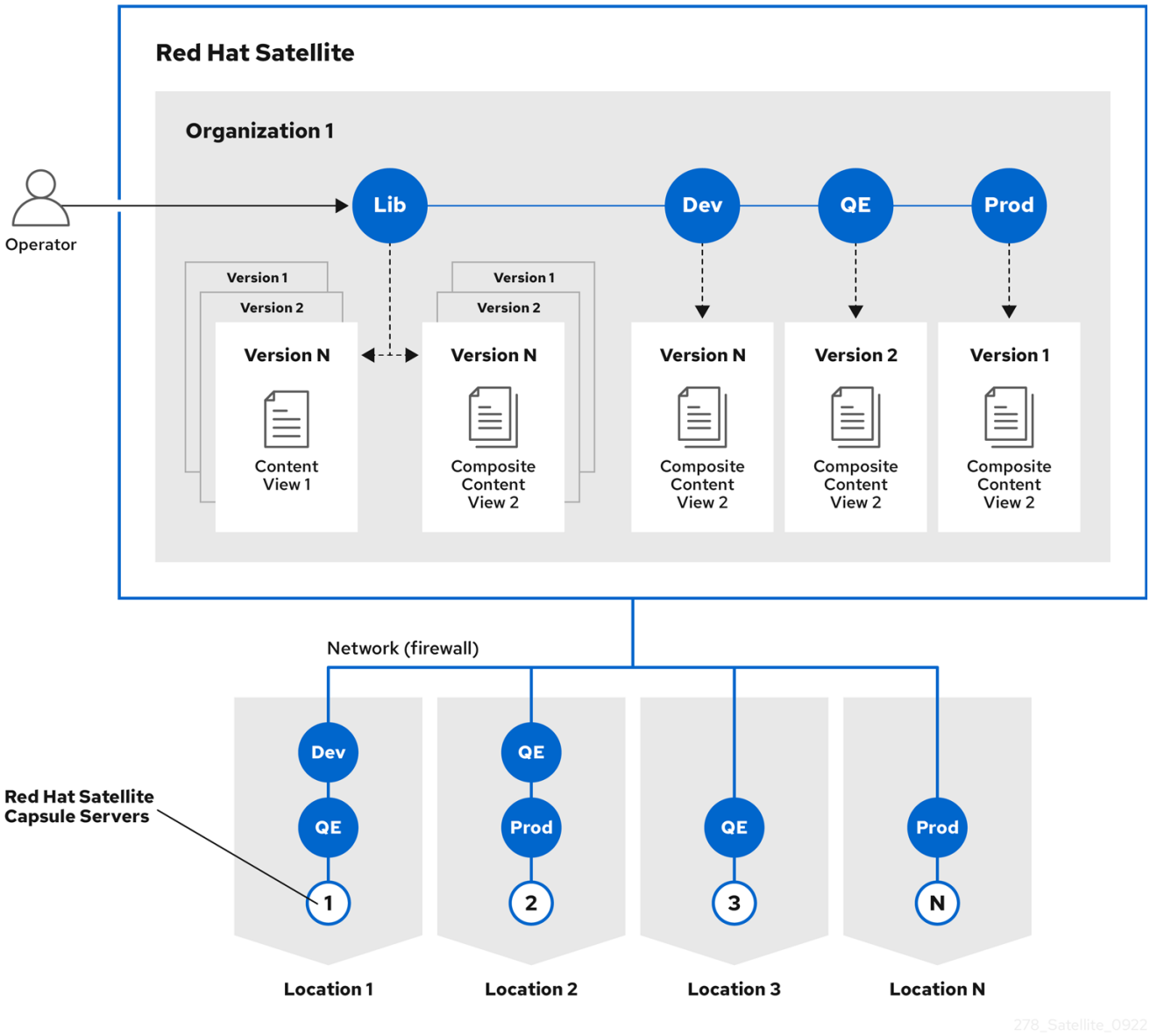
The communication between managed hosts and Satellite Server is routed through Capsule Server that can also manage multiple services on behalf of hosts. Many of these services use dedicated network ports, but Capsule Server ensures that a single source IP address is used for all communications from the host to 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. Satellite Server can have directly managed hosts. The base system running a Capsule Server is also a managed host of Satellite Server.

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

Figure 1.2. Content Lifecycle in Red Hat Satellite



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 lifecycle environments (typically Dev, QA, and Production). When creating a Capsule Server, you can choose which lifecycle 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 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 lifecycle environments. You can also create in-place updates of Content Views. This means creating a minor version of the Content

View in its current lifecycle 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 lifecycles. For more information on content management, see *Managing Content*.

1.2. SYSTEM COMPONENTS

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

Red Hat Satellite consists of the following open source projects:

Foreman

Foreman is an open source application used for provisioning and lifecycle 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 lifecycle.

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 Satellite web UI functions.

REST API

Red Hat Satellite 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 components is extensive. For explanations of frequently used 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.

SELinux must be either in enforcing or permissive mode, installation with disabled SELinux is not supported.

Puppet

Red Hat Satellite includes supported Puppet packages. The installation program allows users to install and configure Puppet servers as a part of Capsule Servers. A Puppet module, running on a Puppet server on the 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 Red Hat Knowledgebase article How does Red Hat support scripting frameworks.

Pulp

Pulp usage is only supported via Satellite 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 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 Satellite 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 databases.

Embedded Tomcat Application Server

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

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.

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. This includes the Satellite Client 6 repositories.

Table 1.1. Content Management Support

Platform	Architectures
Red Hat Enterprise Linux 9	x86_64, ppc64le, s390x, aarch64
Red Hat Enterprise Linux 8	x86_64, ppc64le, s390x
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.

Table 1.2. Host Provisioning Support

Platform	Architectures
Red Hat Enterprise Linux 9	x86_64
Red Hat Enterprise Linux 8	x86_64
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.

Table 1.3. Puppet Agent Support

Platform	Architectures
Red Hat Enterprise Linux 9	x86_64
Red Hat Enterprise Linux 8	x86_64, aarch64
Red Hat Enterprise Linux 7	x86_64
Red Hat Enterprise Linux 6	x86_64, i386

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 more information about Capsule requirements, installation process, and scalability considerations, see Installing Capsule Server.

2.1. CAPSULE FEATURES

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

Infrastructure and host management services:

- **DHCP** – Capsule can manage a DHCP server, including integration with an existing solution such as ISC DHCP servers, Active Directory, and Libvirt instances.
- **DNS** – Capsule can manage a DNS server, including integration with an existing solution such as ISC BIND and Active Directory.
- **TFTP** – Capsule can 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 Red Hat Identity Management and Active Directory.
- **Puppet server** – Capsule can act as a configuration management server by running Puppet server.
- **Puppet Certificate Authority** – Capsule can integrate with Puppet's CA to provide certificates to hosts.
- **Baseboard Management Controller (BMC)** – Capsule can provide power management for hosts using IPMI or Redfish.
- **Provisioning template proxy** – Capsule can serve provisioning templates to hosts.
- **OpenSCAP** – Capsule can perform security compliance scans on hosts.
- **Remote Execution (REX)** – Capsule can run remote job execution on hosts.

Content related features:

- **Repository synchronization** – the content from Satellite Server (more precisely from selected lifecycle environments) is pulled to Capsule Server for content delivery (enabled by Pulp).
- **Content delivery** – hosts configured to use 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.

- **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).

2.2. CAPSULE TYPES

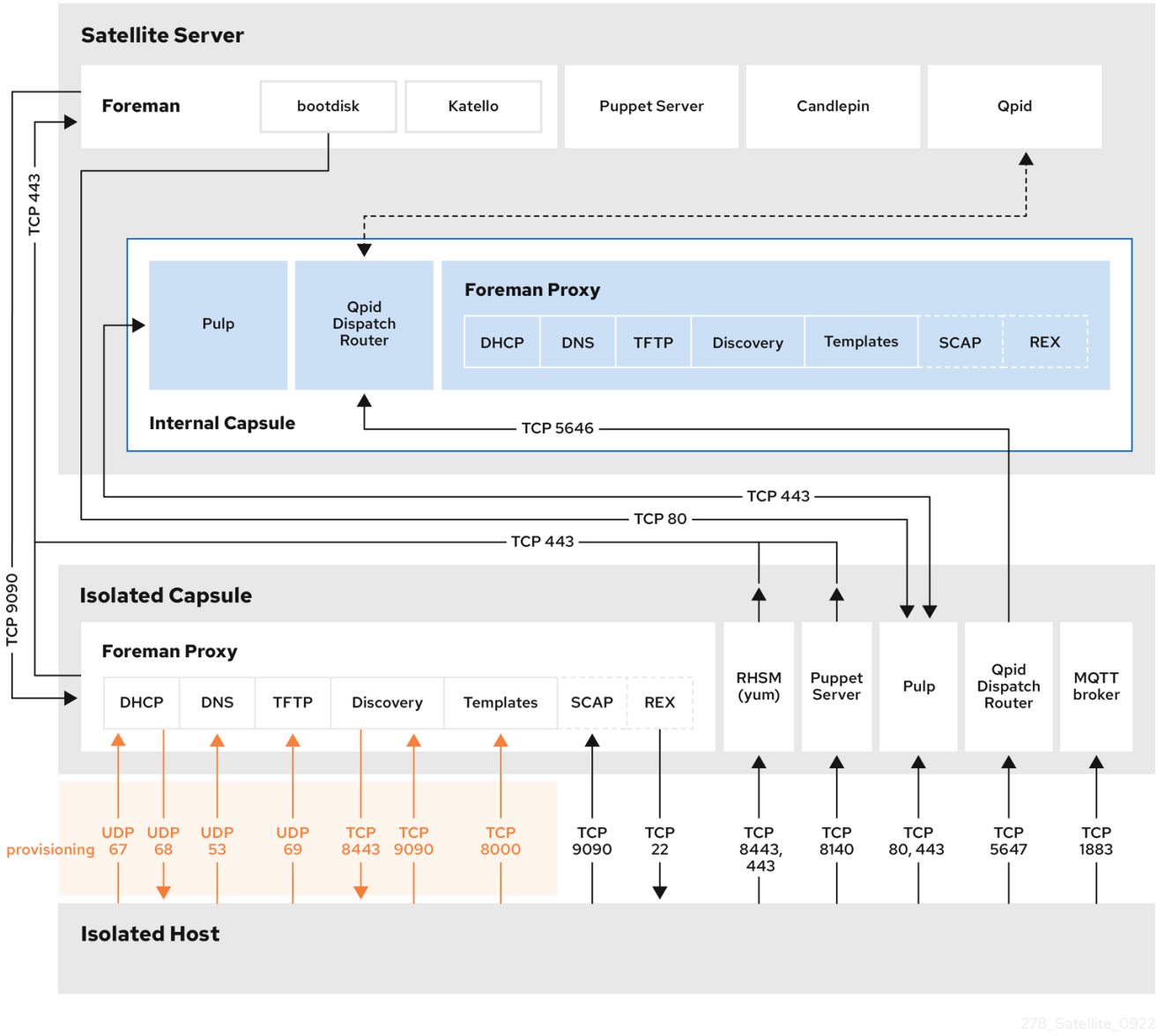
Not all Capsule features have to be enabled at once. You can configure a Capsule Server for a specific limited purpose. Some common configurations include:

- **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 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

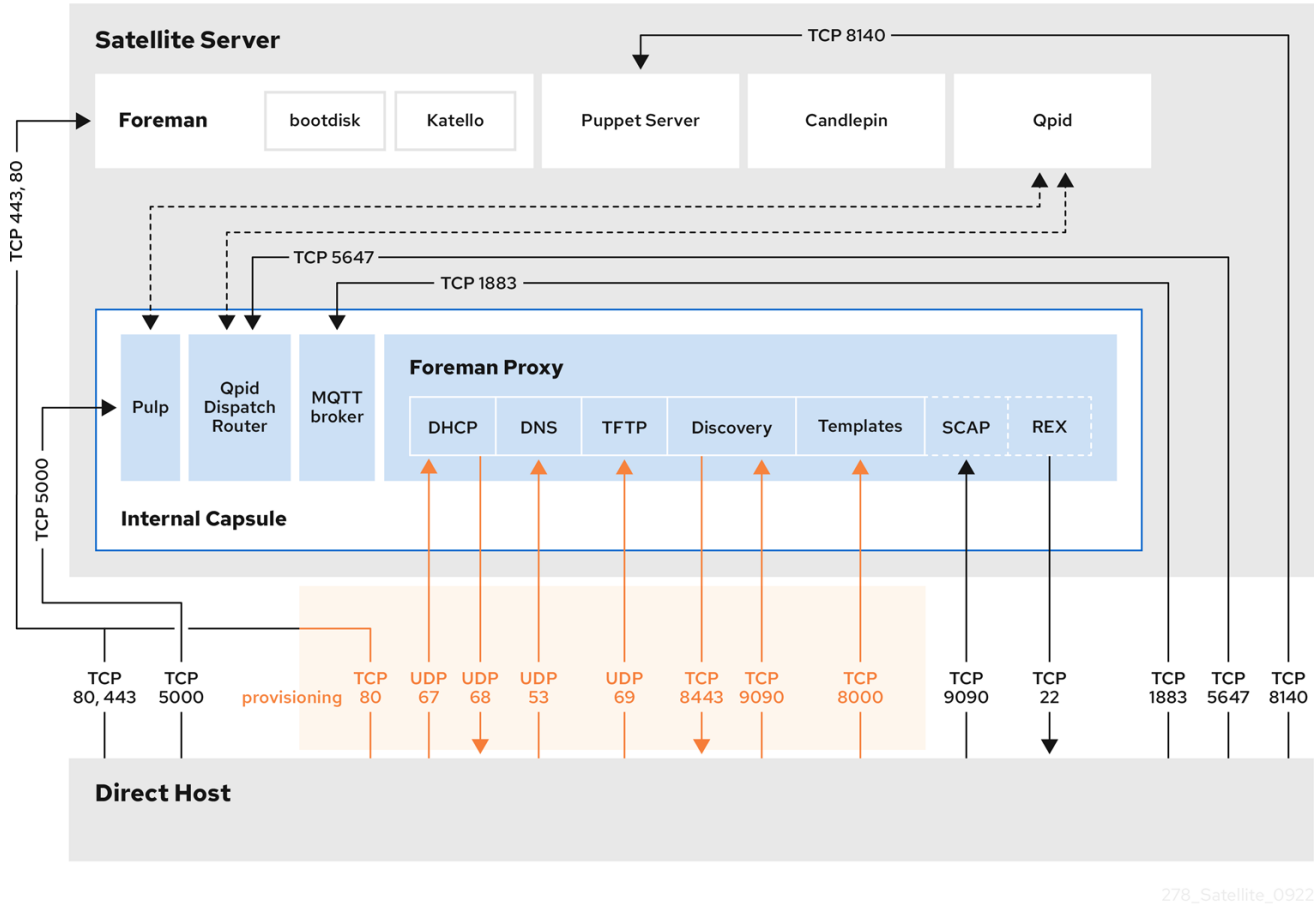
The goal of Capsule isolation is to provide a single endpoint for all of the host's network communications so that in remote network segments, you need only open firewall ports to the Capsule itself. The following diagram shows how the Satellite components interact in the scenario with hosts connecting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

Figure 2.1. Satellite Topology with Isolated Capsule



The following diagram shows how the Satellite components interact when hosts connect directly to 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



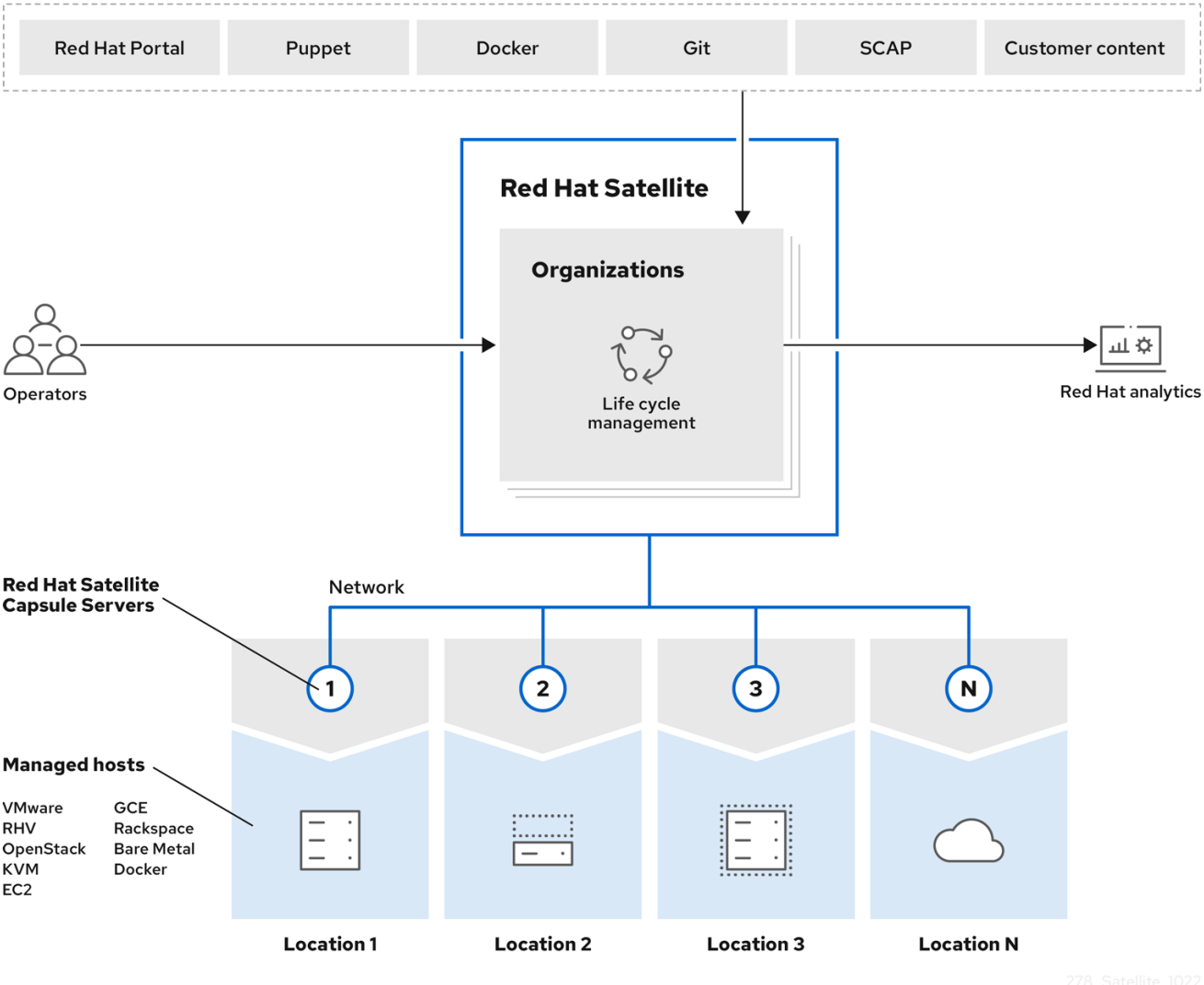
You can find complete instructions for configuring the host-based firewall to open the required ports in the following documents:

- Ports and Firewalls Requirements in *Installing Satellite Server in a Connected Network Environment*
- Ports and Firewalls Requirements in *Installing Satellite Server in a Disconnected Network Environment*
- Ports and Firewalls Requirements in *Installing Capsule Server*

CHAPTER 3. ORGANIZATIONS, LOCATIONS, AND LIFECYCLE ENVIRONMENTS

Red Hat Satellite takes a consolidated approach to Organization and Location management. System administrators define multiple Organizations and multiple Locations in a single Satellite Server. For example, a company might have three Organizations (Finance, Marketing, and Sales) across three countries (United States, United Kingdom, and Japan). In this example, Satellite Server manages all Organizations across all geographical Locations, creating nine distinct contexts for managing systems. In addition, users can define specific locations and nest them to create a hierarchy. For example, Satellite administrators might divide the United States into specific cities, such as Boston, Phoenix, or San Francisco.

Figure 3.1. Example Topology for Red Hat Satellite



Satellite Server defines all locations and organizations. Each respective Satellite Capsule Server synchronizes content and handles configuration of systems in a different location.

The main Satellite Server retains the management function, while the content and configuration is synchronized between the main Satellite Server and a Satellite Capsule Server assigned to certain locations.

3.1. ORGANIZATIONS

Organizations divide Red Hat Satellite resources into logical groups based on ownership, purpose, content, security level, or other divisions. You can create and manage multiple organizations through Red Hat Satellite, then divide and assign your subscriptions to each individual organization. This provides a method of managing the content of several individual organizations under one management system.

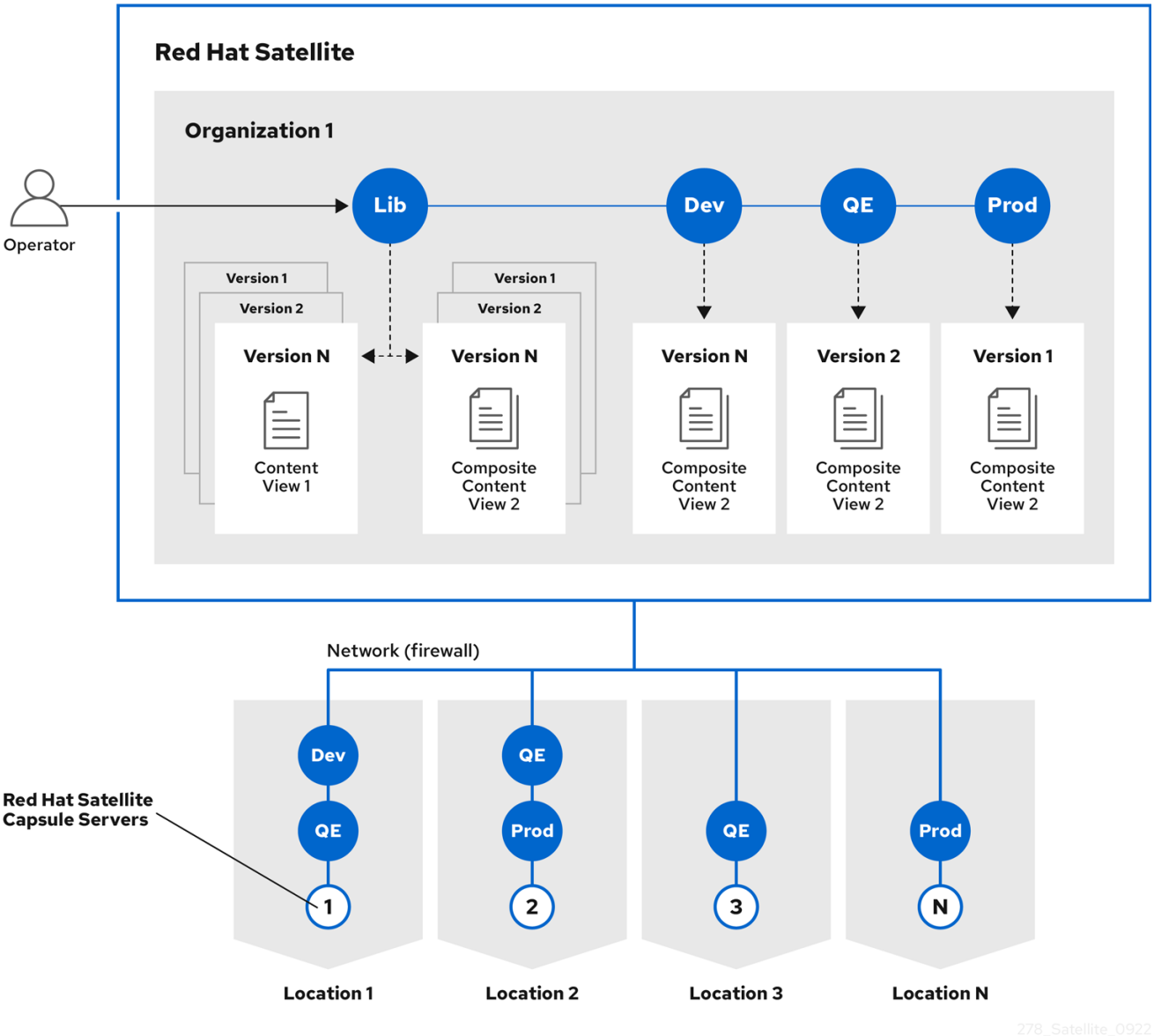
3.2. LOCATIONS

Locations divide organizations into logical groups based on geographical location. Each location is created and used by a single account, although each account can manage multiple locations and organizations.

3.3. LIFECYCLE ENVIRONMENTS

Application lifecycles are divided into **lifecycle environments** which represent each stage of the application lifecycle. Life cycle environments are linked to form an **environment path**. You can promote content along the environment path to the next lifecycle environment when required. For example, if development ends on a particular version of an application, you can promote this version to the testing environment and start development on the next version.

Figure 3.2. An Environment Path Containing Four Environments



CHAPTER 4. 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:

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

The main logical groups in Red Hat Satellite are:

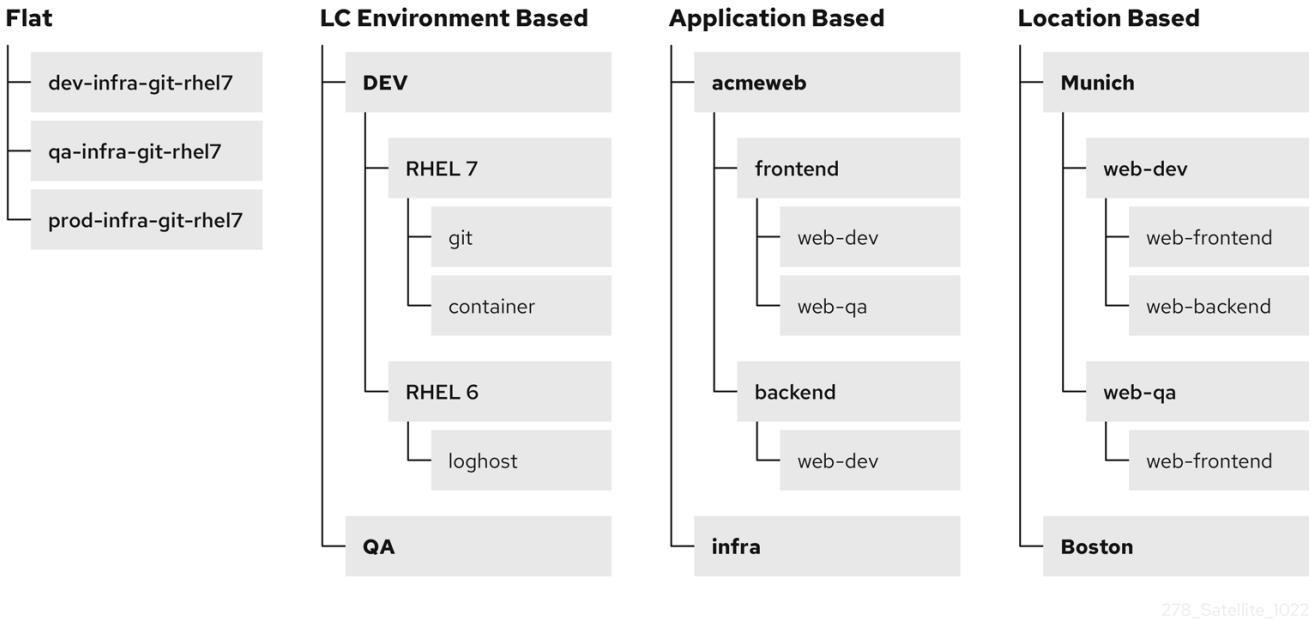
- **Organizations** – the highest level logical groups for hosts. Organizations provide a strong separation of content and configuration. Each organization requires a separate Red Hat Subscription Manifest, and can be thought of as a separate virtual instance of a Satellite Server. Avoid the use of organizations 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. 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 4.1, “Host Group Structures”).
- **Host collections** – a host registered to 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.

4.1. 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 4.1. 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.

Lifecycle Environment Based Structure

In this hierarchy, the first host group level is reserved for parameters specific to a lifecycle 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 lifecycle environments (for example, a dedicated owner for the **Development**, **QA**, and **Production** lifecycle 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 5. PROVISIONING CONCEPTS

An important feature of Red Hat Satellite is unattended provisioning of hosts. To achieve this, Red Hat Satellite uses DNS and DHCP infrastructures, PXE booting, TFTP, and Kickstart. Use this chapter to understand the working principle of these concepts.

5.1. PXE BOOTING

Preboot execution environment (PXE) provides the ability to boot a system over a network. Instead of using local hard drives or a CD-ROM, PXE uses DHCP to provide host with standard information about the network, to discover a TFTP server, and to download a boot image. For more information about setting up a PXE server see the Red Hat Knowledgebase solution How to set-up/configure a PXE Server.

5.1.1. PXE Sequence

1. The host boots the PXE image if no other bootable image is found.
2. A NIC of the host sends a broadcast request to the DHCP server.
3. The DHCP server receives the request and sends standard information about the network: IP address, subnet mask, gateway, DNS, the location of a TFTP server, and a boot image.
4. The host obtains the boot loader **image/pxelinux.0** and the configuration file **pxelinux.cfg/00:MA:CA:AD:D** from the TFTP server.
5. The host configuration specifies the location of a kernel image, **initrd** and Kickstart.
6. The host downloads the files and installs the image.

For an example of using PXE Booting by Satellite Server, see Provisioning Workflow in *Provisioning Hosts*.

5.1.2. PXE Booting Requirements

To provision machines using PXE booting, ensure that you meet the following requirements:

Network requirements

- Optional: If the host and the DHCP server are separated by a router, configure the DHCP relay agent and point to the DHCP server.

Client requirements

- Ensure that all the network-based firewalls are configured to allow clients on the subnet to access the Capsule. For more information, see Figure 2.1, “Satellite Topology with Isolated Capsule”.
- Ensure that your client has access to the DHCP and TFTP servers.

Satellite requirements

- Ensure that both Satellite Server and Capsule have DNS configured and are able to resolve provisioned host names.

- Ensure that the UDP ports 67 and 68 are accessible by the client to enable the client to receive a DHCP offer with the boot options.
- Ensure that the UDP port 69 is accessible by the client so that the client can access the TFTP server on the Capsule.
- Ensure that the TCP port 80 is accessible by the client to allow the client to download files and Kickstart templates from the Capsule.
- Ensure that the host provisioning interface subnet has a DHCP Capsule set.
- Ensure that the host provisioning interface subnet has a TFTP Capsule set.
- Ensure that the host provisioning interface subnet has a Templates Capsule set.
- Ensure that DHCP with the correct subnet is enabled using the Satellite installer.
- Enable TFTP using the Satellite installer.

5.2. HTTP BOOTING

You can use HTTP booting to boot systems over a network using HTTP.

5.2.1. HTTP Booting Requirements with managed DHCP

To provision machines through HTTP booting ensure that you meet the following requirements:

Client requirements

For HTTP booting to work, ensure that your environment has the following client-side configurations:

- All the network-based firewalls are configured to allow clients on the subnet to access the Capsule. For more information, see Figure 2.1, “Satellite Topology with Isolated Capsule” .
- Your client has access to the DHCP and DNS servers.
- Your client has access to the HTTP UEFI Boot Capsule.

Network requirements

- Optional: If the host and the DHCP server are separated by a router, configure the DHCP relay agent and point to the DHCP server.

Satellite requirements

Although TFTP protocol is not used for HTTP UEFI Booting, Satellite uses TFTP Capsule API to deploy bootloader configuration.

For HTTP booting to work, ensure that Satellite has the following configurations:

- Both Satellite Server and Capsule have DNS configured and are able to resolve provisioned host names.
- The UDP ports 67 and 68 are accessible by the client so that the client can send and receive a DHCP request and offer.

- Ensure that the TCP port 8000 is open for the client to download the bootloader and Kickstart templates from the Capsule.
- The TCP port 9090 is open for the client to download the bootloader from the Capsule using the HTTPS protocol.
- The subnet that functions as the host's provisioning interface has a DHCP Capsule, a HTTP Boot Capsule, a TFTP Capsule, and a Templates Capsule
- The **grub2-efi** package is updated to the latest version. To update the **grub2-efi** package to the latest version and execute the installer to copy the recent bootloader from **/boot** into **/var/lib/tftpboot** directory, enter the following commands:

```
# satellite-maintain packages update grub2-efi
# satellite-installer
```

5.2.2. HTTP Booting Requirements with unmanaged DHCP

To provision machines through HTTP booting without managed DHCP ensure that you meet the following requirements:

Client requirements

- HTTP UEFI Boot URL must be set to one of:
  - **http://capsule.example.com:8000**
  - **https://capsule.example.com:9090**
- Ensure that your client has access to the DHCP and DNS servers.
- Ensure that your client has access to the HTTP UEFI Boot Capsule.
- Ensure that all the network-based firewalls are configured to allow clients on the subnet to access the Capsule. For more information, see Figure 2.1, "Satellite Topology with Isolated Capsule".

Network requirements

- An unmanaged DHCP server available for clients.
- An unmanaged DNS server available for clients. In case DNS is not available, use IP address to configure clients.

Satellite requirements

Although TFTP protocol is not used for HTTP UEFI Booting, Satellite use TFTP Capsule API to deploy bootloader configuration.

- Ensure that both Satellite Server and Capsule have DNS configured and are able to resolve provisioned host names.
- Ensure that the UDP ports 67 and 68 are accessible by the client so that the client can send and receive a DHCP request and offer.

- Ensure that the TCP port 8000 is open for the client to download bootloader and Kickstart templates from the Capsule.
- Ensure that the TCP port 9090 is open for the client to download the bootloader from the Capsule via HTTPS protocol.
- Ensure that the host provisioning interface subnet has a HTTP Boot Capsule set.
- Ensure that the host provisioning interface subnet has a TFTP Capsule set.
- Ensure that the host provisioning interface subnet has a Templates Capsule set.
- Update the **grub2-efi** package to the latest version and execute the installer to copy the recent bootloader from the **/boot** directory into the **/var/lib/tftpboot** directory:

```
# satellite-maintain packages update grub2-efi
# satellite-installer
```

5.3. KICKSTART

You can use Kickstart to automate the installation process of a Red Hat Satellite or Capsule Server by creating a Kickstart file that contains all the information that is required for the installation. For more information about Kickstart, see Performing an automated installation using Kickstart in *Performing an advanced RHEL 8 installation*.

5.3.1. Workflow

When you run a Red Hat Satellite Kickstart script, the following workflow occurs:

1. It specifies the installation location of a Satellite Server or a Capsule Server.
2. It installs the predefined packages.
3. It installs Subscription Manager.
4. It uses Activation Keys to subscribe the hosts to Red Hat Satellite.
5. It installs Puppet, and configures a **puppet.conf** file to indicate the Red Hat Satellite or Capsule instance.
6. It enables Puppet to run and request a certificate.
7. It runs user defined snippets.

CHAPTER 6. CONTENT DELIVERY NETWORK STRUCTURE

Red Hat Content Delivery Network (CDN), located at **cdn.redhat.com**, is a geographically distributed series of static webservers which include content and errata designed to be used by systems. This content can be accessed directly through a system registered using Subscription Manager or through the Satellite web UI. The accessible subset of the CDN is configured through subscriptions attached to a system using Red Hat Subscription Management or using Satellite Server.

Red Hat Content Delivery network is protected by X.509 certificate authentication to ensure that only valid users can access it.

Directory Structure of the CDN

```
$ tree -d -L 11
└── content (1)
    ├── beta (2)
    │   └── rhel (3)
    │       └── server (4)
    │           └── 7 (5)
    │               └── x86_64 (6)
    │                   └── sat-tools (7)
    └── dist
        └── rhel
            └── server
                └── 7
                ├── 7.2
                │   └── x86_64
                │       └── kickstart
                └── 7Server
                    └── x86_64
                        └── os
```

1. The **content** directory.
2. Directory responsible for the lifecycle of the content. Common directories include **beta** (for Beta code), **dist** (for Production) and **eus** (For Extended Update Support) directories.
3. Directory responsible for the Product name. Usually **rhel** for Red Hat Enterprise Linux.
4. Directory responsible for the type of the Product. For Red Hat Enterprise Linux this might include **server**, **workstation**, and **computenode** directories.
5. Directory responsible for the release version, such as **7**, **7.2** or **7Server**.
6. Directory responsible for the base architecture, such as **i386** or **x86_64**.
7. Directory responsible for the repository name, such as **sat-tools**, **kickstart**, **rhscl**. Some components have additional subdirectories which might vary.

This directory structure is also used in the Red Hat Subscription Manifest.

PART II. SATELLITE DEPLOYMENT PLANNING

CHAPTER 7. 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.

7.1. 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.

7.2. 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 *Managing Hosts*.

7.3. 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 System Architecture” for an illustration of this approach.

7.4. 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, Satellite Server does not have direct access to the Internet, but the layout of other infrastructure components is not affected. For information about installing Satellite Server from a disconnected network, see Installing Satellite Server in a Disconnected Network Environment. For information about upgrading a disconnected Satellite, see Upgrading a Disconnected Satellite Server in *Upgrading Red Hat Satellite to 6.14*.

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 Satellite Server or a local web server. The content on Satellite Server is then synchronized locally. This allows for complete network isolation of Satellite Server, however, the release frequency of content ISO images is around six weeks and not all product content is included. To see the products in your subscription for which content ISO images are available, log in to the Red Hat Customer Portal at https://access.redhat.com, navigate to **Downloads** > **Red Hat Satellite**, and click **Content ISOs**. For instructions on how to import content ISOs to a disconnected Satellite, see Configuring Satellite to Synchronize Content with a Local CDN Server in *Managing Content*. Note that Content ISOs previously hosted at redhat.com for import into Satellite Server have been deprecated and will be removed in the next Satellite version.

- **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 in Satellite, see Synchronizing Content Between Satellite Servers in *Managing Content*.

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.

7.5. 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 about how to configure a Capsule with external services, see Configuring Capsule Server with External Services in *Installing Capsule Server*.

CHAPTER 8. DEPLOYMENT CONSIDERATIONS

This section provides an overview of general topics to be considered when planning a Red Hat Satellite deployment together with recommendations and references to more specific documentation.

8.1. SATELLITE SERVER CONFIGURATION

The first step to a working Satellite infrastructure is installing an instance of Satellite Server on a dedicated Red Hat Enterprise Linux 8 server.

- For more information about installing Satellite Server in a connected network, see *Installing Satellite Server in a Connected Network Environment*.
  On large Satellite deployments, you can improve performance by configuring your Satellite with predefined tuning profiles. For more information, see Tuning Satellite Server with Predefined Profiles in *Installing Satellite Server in a Connected Network Environment* .

- For more information about installing Satellite Server in a disconnected network, see *Installing Satellite Server in a Disconnected Network Environment*.
  On large Satellite deployments, you can improve performance by configuring your Satellite with predefined tuning profiles. For more information, see Tuning Satellite Server with Predefined Profiles in *Installing Satellite Server in a Disconnected Network Environment* .

Adding Red Hat Subscription Manifests to Satellite Server

A Red Hat 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 Red Hat Subscription Manifest see Managing Red Hat Subscriptions in *Managing Content*.

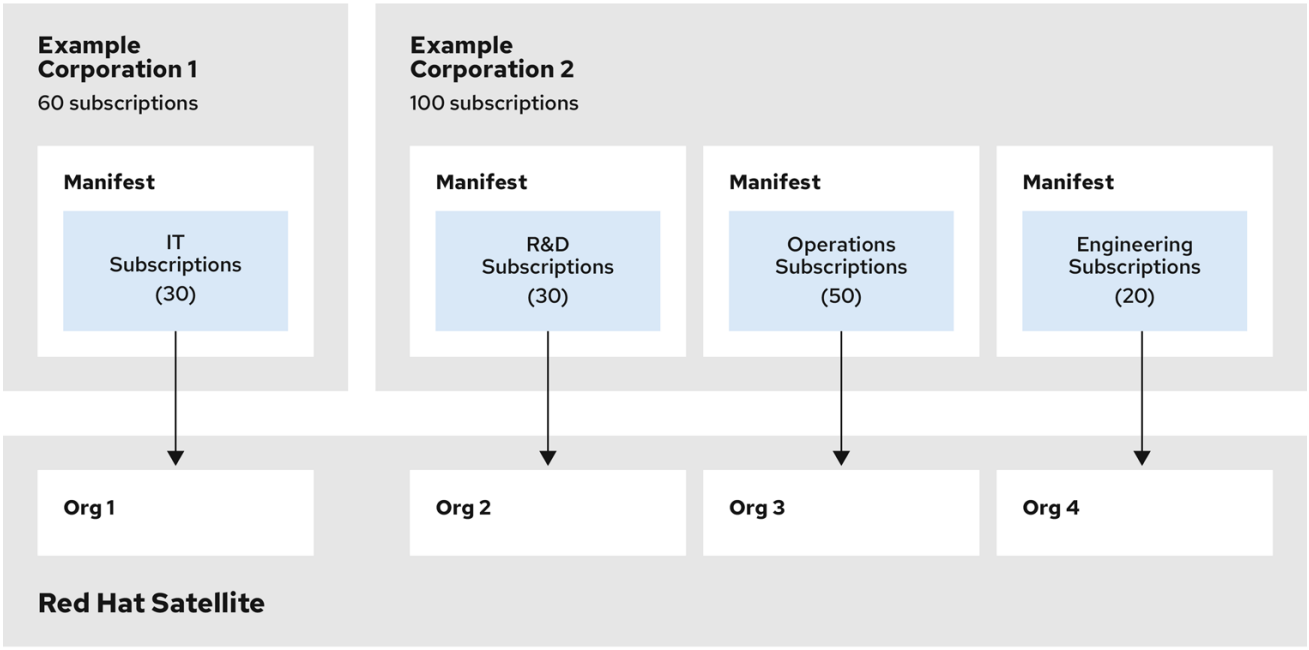
Red Hat Satellite requires a single manifest for each organization configured on the Satellite. If you plan to use the Organization feature of Satellite to manage separate units of your infrastructure under one Red Hat Network account, then assign subscriptions from the one account to per-organization 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 must manage systems but do not have access to the subscriptions for the RPMs, you must use Red Hat Enterprise Linux Satellite Add-On. For more information, see Satellite Add-On.

The following diagram shows two Red Hat Network account holders, who want their systems to be managed by the same Satellite 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 completely independently of Example Corporation 2's organizations (R&D, Operations, and Engineering).

Figure 8.1. Satellite Server with Multiple Manifests



When creating a Red Hat 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 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.
- Note the date when the subscriptions are due to expire and plan for their renewal before the expiry date.
- Create one manifest per organization. You can use multiple manifests and they can be from different Red Hat subscriptions.

Red Hat Satellite allows the use of future-dated subscriptions in the manifest. This enables uninterrupted access to repositories when future-dated subscriptions are added to a manifest before the expiry date of existing subscriptions.

Note that the Red Hat Subscription Manifest can be modified and reloaded to 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.

8.2. SATELLITE SERVER WITH EXTERNAL DATABASE

When you install Satellite, the **satellite-installer** command creates databases on the same server that you install Satellite. Depending on your requirements, moving to external databases can provide increased working memory for Satellite, which can improve response times for database operating requests. Moving to external databases distributes the workload and can increase the capacity for performance tuning.

Consider using external databases if you plan to use your Satellite deployment for the following scenarios:

- Frequent remote execution tasks. This creates a high volume of records in PostgreSQL and generates heavy database workloads.
- High disk I/O workloads from frequent repository synchronization or Content View publishing. This causes Satellite to create a record in PostgreSQL for each job.
- High volume of hosts.
- High volume of synced content.

For more information about using an external database, see Using External Databases with Satellite in *Installing Satellite Server in a Connected Network Environment* .

8.3. LOCATIONS AND TOPOLOGY

This section outlines general considerations that should help you to specify your Satellite deployment scenario. The most common deployment scenarios are listed in Chapter 7, *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 7.5, "Capsule with External Services").
- **Is my Satellite Server required to be disconnected from the Internet?**– Disconnected Satellite is a common deployment scenario (see Section 7.4, "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. To learn about provisioning on different compute resources see *Provisioning Hosts*.

8.4. CONTENT SOURCES

The Red Hat 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, or Docker images).

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** and **On demand**. The **On demand** setting saves space and time by only downloading packages when requested by clients. For detailed instructions on setting up content sources see Importing Content in *Managing Content*.

A custom repository within 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.

8.5. CONTENT LIFECYCLE

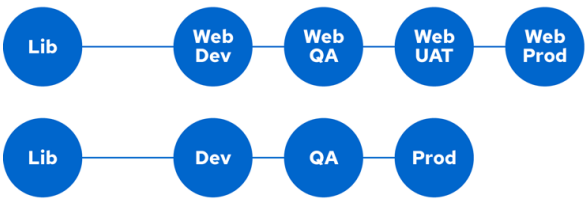
Satellite provides features for precise management of the content lifecycle. A **lifecycle environment** represents a stage in the content lifecycle, a **Content View** is a filtered set of content, and can be considered as a defined subset of content. By associating Content Views with lifecycle environments, you make content available to hosts in a defined way. See Figure 1.2, "Content Lifecycle in Red Hat Satellite" for visualization of the process. For a detailed overview of the content management process see Importing Custom Content in *Managing Content*. The following section provides general scenarios for deploying content views as well as lifecycle environments.

The default lifecycle 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 lifecycle environment path that suits your content workflow. The following scenarios are common:

- **A single lifecycle 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 lifecycle 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 lifecycle environment paths** – each application has a separate path, which allows for individual application release cycles. You can associate specific compute resources with application lifecycle 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 dedicated content views for the operating system and application content. By using a composite, you can manage your operating system and applications separately and at different frequencies.

- **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.

8.6. CONTENT DEPLOYMENT

Content deployment is the management of errata and packages on content hosts. There are two methods for content deployment on Satellite. The default method is **remote execution**, and the second method is the deprecated **Katello agent**.

- **Remote execution** – Remote execution allows installation, update, or removal of packages, the bootstrap of configuration management agents, and the trigger of Puppet runs. You can configure Satellite to perform remote execution either over SSH (push based) or over MQTT/HTTPS (pull based). While remote execution is enabled on Satellite Server by default, it is disabled on Capsule Servers and content hosts. You must enable it manually.
  This is the preferred method for content deployment.

- **Katello agent** – Katello agent installs and updates packages. It uses the **goferd** service, which communicates to and from the Satellite Server. It is enabled and started automatically on content hosts after a successful installation of the **katello-agent** package.
  Note that the Katello agent is deprecated and will be removed in a future Satellite release.

8.7. PROVISIONING

Satellite 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 Provisioning Workflow in *Provisioning Hosts*. The same guide contains instructions for provisioning on various compute resources.

8.8. ROLE BASED AUTHENTICATION

Assigning a role to a user enables controlling access to Satellite 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 lifecycle**– 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 or license manager.

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 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 lifecycle 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.

For instructions on defining roles and assigning them to users, see Managing Users and Roles in *Administering Red Hat Satellite*. The same guide contains information on configuring external authentication sources.

8.9. 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:

- **Discovering bare metal hosts**– the Satellite Discovery plug-in enables automatic bare-metal discovery of unknown hosts on the provisioning network. These new hosts register themselves to 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 more information, see Creating Hosts from Discovered Hosts in *Provisioning Hosts*.

- **Backup management** – backup and disaster recovery instructions, see Backing Up Satellite Server and Capsule Server in *Administering Red Hat Satellite*. Using remote execution, you can also configure recurring backup tasks on managed hosts. For more information on remote execution see Configuring and Setting up Remote Jobs in *Managing Hosts*.

- **Security management** – Satellite 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 *Managing Hosts*.

- **Incident management** – Satellite 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 Satellite Server, including the event history of recent changes. Satellite is also integrated with Red Hat Insights.

- **Scripting with Hammer and API**– Satellite 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 Hammer CLI Guide and API 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 and GIDs, while others take the next available UID and GID on the system instead. To control the UIDs and GIDs assigned to accounts, you can define accounts before installing Satellite. Because some of the accounts have hard-coded UIDs and GIDs, it is not possible to do this with all accounts created during Satellite installation.

The following table lists all the accounts created by Satellite during installation. You can predefine accounts that have **Yes** in the **Flexible UID and GID** column with custom UID and GID before installing Satellite.

Do not change the home and shell directories of system accounts because they are requirements for Satellite to work correctly.

Because of potential conflicts with local users that Satellite creates, you cannot use external identity providers for the system users of the Satellite base operating system.

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

User name	UID	Group name	GID	Flexible UID and GID	Home	Shell
foreman	N/A	foreman	N/A	yes	/usr/share/foreman	/sbin/nologin
foreman-proxy	N/A	foreman-proxy	N/A	yes	/usr/share/foreman-proxy	/sbin/nologin
apache	48	apache	48	no	/usr/share/httpd	/sbin/nologin
postgres	26	postgres	26	no	/var/lib/pgsql	/bin/bash
pulp	N/A	pulp	N/A	no	N/A	/sbin/nologin
puppet	52	puppet	52	no	/opt/puppetlabs/server/data/puppetserver	/sbin/nologin
qdrouterd	N/A	qdrouterd	N/A	yes	/var/lib/qdrouterd	/sbin/nologin
qpidd	N/A	qpidd	N/A	yes	/var/lib/qpidd	/sbin/nologin

User name	UID	Group name	GID	Flexible UID and GID	Home	Shell
saslauth	N/A	saslauth	76	no	/run/saslauthd	/sbin/nologin
tomcat	53	tomcat	53	no	/usr/share/tomcat	/bin/nologin
unbound	N/A	unbound	N/A	yes	/etc/unbound	/sbin/nologin

APPENDIX B. GLOSSARY OF TERMS

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

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, 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 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 deployment to facilitate content federation and distribution (act as a Pulp mirror), and to run other localized services (Puppet server, **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 server 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 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 uses for deployment of hosts and systems. Examples include Red Hat Virtualization, Red Hat OpenStack Platform, EC2, and VMWare.

Container (Docker Container)

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

Container Image

A static snapshot of the container's configuration. Satellite 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), or Docker images. Content is synchronized into the Library and then promoted into lifecycle 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 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 lifecycle environment path, or modified using incremental upgrades.

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 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 Satellite Server and 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 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 construct that provides additional data for a server to use when configuring hosts. Red Hat Satellite acts as an External Node Classifier to Puppet servers in a Satellite deployment.
Note that the External Node Classifier will be removed in the next Satellite version.

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 component mainly responsible for provisioning and content lifecycle management. Foreman is the main upstream counterpart of Red Hat Satellite.

Satellite services

A set of services that Satellite Server and Capsule Servers use for operation. You can use the **satellite-maintain** tool to manage these services. To see the full list of services, enter the **satellite-maintain service list** command on the machine where Satellite or Capsule Server is installed.

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.
Note that Foreman Hook functionality is deprecated and will be removed in the next Satellite version.

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 Satellite Server, which matches it against the host entry.

Hammer

A command line tool for managing Red Hat Satellite. 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 manages.

Host Collection

A user defined group of one or more Hosts used for bulk actions such as errata installation.

Host Group

A template for building a host. Host groups hold shared parameters, such as subnet or lifecycle 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 Satellite Server.

Incremental Upgrade (of a Content View)

The act of creating a new (minor) content view version in a lifecycle 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 Satellite Server. Every job is defined in a job template.

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**. The **On Demand** setting saves storage space and synchronization time by only downloading the packages when requested by a client.

Location

A collection of default settings that represent a physical place.

Library

A container for content from all synchronized repositories on Satellite Server. Libraries exist by default for each organization as the root of every lifecycle environment path and the source of content for every content view.

Lifecycle Environment

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

Lifecycle Environment Path

A sequence of lifecycle 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.

Manifest (Red Hat Subscription Manifest)

A mechanism for transferring subscriptions from the Red Hat Customer Portal to Red Hat Satellite. Do not confuse with Puppet Manifest.

Migrating Satellite

The process of moving an existing Satellite installation to a new instance.

OpenSCAP

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

Organization

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

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 server.

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 lifecycle environment to another.

Provisioning Template

Defines host provisioning settings. Provisioning templates can be associated with host groups, lifecycle environments, or operating systems.

Publish (a Content View)

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

Pulp

A service within Katello responsible for repository and content management.

Pulp Mirror

A Capsule Server component that mirrors content.

Puppet

The configuration management component of Satellite.

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, which are files with the **.pp** extension. The files contain code to define a set of necessary resources, such as packages, services, files, users and groups, and so on, using a set of key-value pairs for their attributes.
Do not confuse with Manifest (Red Hat Subscription Manifest) .

Puppet Server

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 web UI, you can view those jobs under **Monitor** > **Recurring logics**.

Registry

An archive of container images. Satellite 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.

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 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.

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 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 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 infrastructure. Configure trends in Satellite web UI under **Monitor** > **Trends**.

Updating Satellite

The process of advancing your Satellite Server and Capsule Server installations from a z-stream release to the next, for example Satellite 6.14.0 to Satellite 6.14.1.

Upgrading Satellite

The process of advancing your Satellite Server and Capsule Server installations from a y-stream release to the next, for example Satellite 6.13 to Satellite 6.14.

User Group

A collection of roles which can be assigned to a collection of users.

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, virt-who reports those IDs to Satellite Server so that it can provide subscriptions for hosts provisioned on virtual machines.