Red Hat

Satellite 6

Managed Design Program Phase 2

A general "How To" of what's expected, interacting with Red Hat, and time lines.

REVISIONS

Rev	Date	Author	Change
0.1	17-Sept 2013	David Caplan	Began with MDP1 final copy
0.2	22-Oct 2013	David Caplan	First round of revisions from QE and Dev
0.3	24-Oct 2013	David Caplan	Updated workflow based on first end-2-end testing by QE
1.0	11-Nov-2013	David Caplan and Mike McCune	Final Edits and additions using QE approved release
1.1	12-Nov-2013	David Caplan	Corrected SKU counts

WELCOME!

Thank you for your interest in participating in the Red Hat Satellite 6 Managed Design Program with the Red Hat Satellite product team. We have crafted the program (MDP) as a means to interact with customers who are crafting solutions related to systems management with Satellite 5, Puppet, with or without Foreman, and have a vested interest in the functionality of Satellite 6. The MDP program will provide early access to Satellite 6, allowing participants to utilize the evolving user interface, APIs, and CLI. In return, we are requesting candid feedback and suggestions for new functionality, insights into workflow improvements, and better integration into existing IT ecosystems.

The current roadmap for Satellite 6 calls for 3 MDP code drops. The first drop occurred the last week of June 2013, the second is planned for November 12, 2013, the third later in 2014. Each release represents a snapshot of Satellite 6 as it progresses toward its General Availability milestone in June of 2014. Each MDP drop is associated with a major theme. The theme for MDP1 was "The Satellite 6 Experience". The theme for MDP2 is "Satellite 6 Content and Configuration Management".

READ THIS FIRST

The following are known issues with the current MDP2 Drop.

- **Installation and General Support**
  - MDP2 is designed to be installed on up to 6 machines: One Satellite 6 head-end and up to 5 remote federated nodes.
  - Upgrading from MDP1 to MDP2 will not be supported.
  - Upgrading from MDP2 to MDP3 will not be supported
  - MDP2 is composed of Foreman V1.3 and Katello V1.4. You are free to exercise any of the included functionality and provide us feedback. However, we have have only exhaustively tested the set of use cases defined below for this release.
  - MDP2 designed for localization, however MDP2 is English only
  - The installer will create many lines of output on a small terminal. This will be corrected in for the next MDP release.
  - The machine which you install on must have a fully qualified domain name
  - Installing from the ISO requires that the Red Hat production and Beta GPG keys are imported before running install_packages.sh.
- **System Requirements for Satellite 6**
  - Minimum RAM requirement is 8 GB for Satellite 6 Core and 4GB for distributed Satellite Nodes (Federated Content and Provisioning Server)
  - Minimum for both Satellite 6 Server and Nodes is 100GB of storage to sync RHEL5 and RHEL6 10GB for the basic installation of Satellite 6 MDP and 90GB for syncing RHEL5 and RHEL6 repositories. This applies to remote nodes with large environments
  - Provisioning RHEL 5 machines
    - RHEL5 repositories contain yum metadata in the distribution and until the

following Bugzilla is resolved installing packages on RHEL5 systems may not function properly: https://bugzilla.redhat.com/show_bug.cgi?id=976575

- There are known performance issues fetching errata for system groups while other actions are taking place (i.e. syncs, promotions)

- **System Requirements for Remote Satellite node**
  - Minimum RAM requirement is 8 GB
  - Minimum for both 100GB of storage to sync RHEL5 and RHEL6 10GB for the basic installation of Satellite 6 MDP and 90GB for syncing RHEL5 and RHEL6 repositories
  - MDP2 has only officially tested x86_64 and i386 RHEL5/6 client but ppc, s390 and s390x packages are available for use at your own risk.
- **User administration**
  - For MDP2 only the Admin User is supported
  - LDAP authentication has been tested but only the default admin user can perform the majority of available tasks with some exceptions
    - No user and role creation
    - https://bugzilla.redhat.com/show_bug.cgi?id=966263
    - https://bugzilla.redhat.com/show_bug.cgi?id=966921
  - Data Entry Validation
  - Some Foreman fields whose data is not fully validated/parsed (usually whitespace) can lead to confusion and messy data.
- **Katello Foreman Interworking**
  - For MDP2, the integration between Foreman and Katello (Provisioning and Content) has had limited testing. Users should manage organization and environment data only from the Katello side.
  - Configuring provisioning requires knowledge of how to configure Foreman
- **Provisioning RHEL 5 machines**
  - RHEL5 repositories contain yum metadata in the distribution and until the following Bugzilla is resolved installing packages on RHEL5 systems may not function properly: https://bugzilla.redhat.com/show_bug.cgi?id=976575
  - There are known performance issues fetching errata for system groups while other actions are taking place (i.e. syncs, promotions)
- **User Interface**
  - There are several references to the Foreman website for support. This will updated to reflect Red Hat support locations in a future release.

MDP OVERVIEW

What you will receive from us

For MDP2, your account will be provided with a 90 day evaluation subscription. The subscription will provide you access to the Satellite 6 Server packages (1 Satellite 6 server and 4 remote nodes), install ISO, and 10 Red Hat Enterprise Server subscriptions to support your testing. In addition to this

document, you will be provided with installation instructions, providing steps for accessing the content and installing the software.

The subscriptions you will receive are:

- On RHN Hosted MCT1550 maps to the RHN Satellite 6 Beta ISO
- On RH CDN SER0440 maps to the all relevant Satellite MDP client channels

Meetings

Due to the scope and size of MDP2 we are waiving the MDP1 requirement for an initial kickoff meeting. Instead we will auto-enroll you into the Satellite 6 MDP customer portal group where you can find our Knowledge Base and participate in forums to maximize the benefits of participation in the Managed Design Program. We would still welcome any opportunity to engage in conference calls or on-site visits (when practical for all concerned) and would expect that you will work with your Red Hat Sales Executive, SA, or TAM to schedule meetings with representatives from internal MDP stakeholders. If there is a problem with your enrollment in the Satellite 6 MDP customer portal please contact David Caplan dcaplan@redhat.com, Alice McClure, amcclure@redhat.com, Xixi xdmoon@redhat.com, or Brian Hamrick bhamrick@redhat.com and we will add you to the membership list.

After installation of Satellite 6 MDP2 we would prefer to weekly or biweekly email updates during the evaluation period.

The Support Process

The Red Hat MDP team, as well as GSS, will be providing support for the MDP program. The Red Hat MDP team is composed of Product Managers, Red Hat Support, Development Leads, and your SA and TAM.

We have created the following mechanisms for communication and collaboration during the MDP cycles:

- Satellite6-mdp2@redhat.com email list for questions, feedback, accolades, and general communications. The email list is unidirectional such that all external user communication is delivered to internal Red Hat Satellite 6 MDP stakeholders.
- Freenode IRC: You can join #satellite6-mdp on freenode to ask questions in an interactive fashion with Red Hat Satellite 6 Stakeholders
- Red Hat Customer Portal Group: Please send us your rhn login if you have not been auto-enrolled in the portal group and we will enroll you into the group.

Regardless of how you choose to communicate, we ask that you follow the process below:

1. Ask question or send commentary (a bug report, of sorts) to the mailing list
2. Your question / bug-report will be triaged by PM, Engineering and Red Hat Support
3. Red Hat will manage formalized bug-filing on the issue (if an issue) and respond back to you directly (or via your TAM or SA)

In addition to ad hoc communication and collaborated by the aforementioned channels, we would like to schedule informal conference calls to discuss your experiences and ideas for improving the User Experience. Please work with your TAM/SA/SR to schedule calls with

members of the Red Hat MDP2 team (listed below).

How to provide us with feedback

We encourage you to provide:

- Regular email correspondence to Satellite6-mdp2@redhat.com. We will do our best to respond to all emails within 48 hours (depending on time zones and resources on any given day)
- Commentary, bugs, questionable behaviors, accolades, suggested alternatives
- Suggested reporting template (feel free to improvise):
  - Tested item (e.g. Manifest Import, Content View Definition Creation, etc)
  - Expected behavior (e.g. Attempted to publish a content View)
  - Anticipated results vs. actual results
  - Performance and responsiveness issues
  - Format: Any format is acceptable

Direct Contact with the MDP Team

If you would prefer direct contact with members of the MDP team, you can use:

- David Caplan, Product Manager, dcaplan@redhat.com
- Bryan Kearney, Engineering Manager, bkearney@redhat.com
- Todd Sanders, Engineering Director, tsanders@redhat.com
- Ohad Levy, Engineering Manager, olevy@redhat.com
- Mike McCune, Engineering Manager, mmccune@redhat.com
- Og Maciel, Quality Engineering Manager, omaciel@redhat.com
- Brian Hamrick, Cloud BU GSS Manager, bhamrick@redhat.com
- Xixi, Systems Management GSS Tech Lead, xdmoon@redhat.com

What to Test?

Red Hat Satellite 6 has been designed to deliver enterprise-class systems management tailored to the realities of the modern IT ecosystem. Satellite 6 represents a completely fresh approach to provisioning and managing systems, with new capabilities that span subscription services, content, provisioning, configuration, automation, integration, and lifecycle operations. The packaging of Satellite 6 released in this program is represents a work-in-progress and is being offered to expose you to the essential underpinnings of Satellite 6 along a trajectory toward general availability. Therefore, the success of the Managed Design Program hinges on important 2 axes:

1) **Our ability to release Satellite 6 with known constraints from a feature/function/operational perspective (a natural outcrop of early phase development)**
2) **Your willingness to work within the constraints associated with each MDP drop and your agreement to concentrate your initial testing efforts accordingly. You are free to extend and expand your testing and experimentation, but our ability to support you in this program is predicated on focusing your initial efforts executing the prescribed tasks delineated below.**

THE WORKFLOW FOR MDP2

The theme for MDP2 is "Satellite 6 Infrastructure for Content Federation and Puppet Configuration". The workflow in Satellite 6 for MDP2 begins with Installation of Satellite 6 and, at least, one federated node for delivering content and provisioning orchestration to one or more RHEL clients. After installation of Satellite 6 infrastructure the focus of the evaluation shifts to subscription and content management, then provisioning, and configuration, and finally subsequent registration for errata and updates. Our primary focus for MDP2 includes:

- Packaging and Installation of Satellite 6, Remote Nodes that federate content and provisioning orchestration, and supply a Puppet Master
- Single Sign-on (as admin) across content and provisioning cores
- Creation of Organizations and Environments across Katello and Foreman
- Integration with Red Hat CDN for synchronizing Red Hat Content
- Integration with Puppet Forge and GIT (indirectly for modules in *.tar.gz format) for synchronizing Puppet Manifests
- Establishment of remote Satellite 6 Content and Provisioning Nodes to federate the deployment
- Creation, publication, and promotion of Content Views and Composite Content Views (that contain RPMs and Puppet Modules) into Environments
- Creation of Activation Keys
- Creation of host groups to define standard builds of application stacks on RHEL with RPMs and Puppet Modules bounded by Content Views
- Consumption of Activation Keys, Environments, Content Views, Puppet Modules and other provisioning artifacts to build, configure, and register RHEL guests on a given VM or physical machine
- Directing provisioning and content to originate from a federated Satellite node
- Insuring that provisioned systems properly register with Satellite 6 core or federated node for updates and errata
- Configuration and drift management
- Subscription Management and reporting

The suggested storyboard provided below is designed to guide your activities and interaction with the MDP 2 version of Satellite 6. Many details are intentionally removed to encourage experimentation (trial and error) and enhance the diversity of feedback from all MDP participants.

After you complete the constrained workflow and have reported to us your experience, you are welcome to expand testing in any direction that suits your interest, resources, and timetable. We will be happy to accept your feedback in these "out-of-scope-for-MDP2" activities.

MDP2 Storyboard for Testing

This section provides the basic storyboard for testing and hopefully some context around the purpose of each step

1) INITIAL SETUP

- NOTE: If you are re-using a system that had MDP1 installed we require that all previous versions of Satellite 6 be removed or the OS freshly installed.
- After downloading the ISO from *access.redhat.com/download* you can install the Satellite 6 Application on a suitable machine. You may also install via *subscription-manager and yum*. This is accomplished by running the Satellite 6 installer, which sets up Katello, Foreman, and Signo the single sign-on application on the host machine. After the installation is complete, log onto Katello as *admin* and create an initial Organization and Environment
- Installation instructions can be found at, please follow the instructions there for your initial installation steps: *https://access.redhat.com/site/documentation/Red_Hat_Satellite/*
- MDP2 only supports the admin role on Katello
- Add an Environment called "Satellite Infrastructure" for use by Satellite 6 to envelope its own infrastructure components
- Download the manifest from the customer portal here:, and synchronize the content. The manifest will include the satellite subscriptions.
- Please evaluate and comment on the installation process for Satellite 6 core and remote nodes

2) Create multiple Federated Satellite Nodes for distributed Content & Smart Proxy actions

For Satellite 6 there are three supported services bundles for Federated Nodes:

- **Provisioning and configuration services**: This includes a combined Smart Proxy for provisioning services: DNS, DHCP, TFTP and a Puppet Master which register to the Satellite 6 core server
- **Content Only Node plus Puppet Master**: This only includes content services but adds the puppet master to automate content delivery from the Satellite 6 core
- **Combined Provisioning, Configuration, and Content**. This configuration combines all services on a single node.

Satellite 6 offers these combinations to reflect the varying best practices used by Red Hat Satellite Customers. Some customers use in-house methods for provisioning and therefore will not require Satellite 6 orchestration services.

For the purposes of Satellite 6 MDP2 we would encourage you to use bundle #3 as it exercises the full capabilities of Satellite 6 for content, provisioning, and configuration.

- To establish a Satellite node you must provision a RHEL 6 machine via the ISO provided from RHN. Then you need to create a certificate and activation key on the Satellite 6 core host using the node-certs-generate cli.
- After the remote node is running you will register it the Satellite server using subscription manager and activation key from above.

```
# rpm -Uvh http://sat6host.example.redhat.com/pub/candlepin-cert-consumer-latest.noarch.rpm
# subscription-manager register --org Katello_Infrastructure --activationkey node --force
```

- After the node is registered you need to run the node-certs-generate-cli. See the INSTALLATION NOTES below for more details.

3) Create Environments

After one or more remote Satellite Nodes are created go ahead and create additional application life-cycle environments (e.g. Dev -> Test -> Prod)

- After you have created the environments on the Satellite 6 core server use the CLI call: `katello node add_environment --org=<org> --environment=<env>` to specify which environments you want to synchronize to the remote node(s). You can also issue the Katello node sync at anytime to sync content from the Satellite server to the node.

4) Define Puppet Application Content

You must now create a puppet provider, and pull in puppet content from a remote puppet repository or a local disk. You can use the Puppet Forge or your own GIT repo for the source. NOTE: Puppet content must be in the form of a Puppet Module, information about authoring Puppet in Module format can be found here:
http://docs.puppetlabs.com/puppet/2.7/reference/modules_fundamentals.html#writing-modules

5) Define RPM Application Content

User creates a custom RPM provider and pulls in custom rpm content

6) Creating Content Views

- Next you will create Content View Definitions by choosing products from synced repos found in the library. Please be sure to use filters, which are either white (include) or black (exclude) lists in combination with date and version ranges for packages and/or errata. It is highly recommended that you filter Puppet Module content by author to manage module name conflicts
- The Content View Definitions need to be published, after which they are known simply as "Content Views." Changes made to existing Content View Definitions will show incremented version numbers when refreshed or republished. If you wish to modularize your Content Views to enable reuse of common products or repos, you can combine multiple Content Views into a single Composite Content View.

7) Promote Content

By promoting Content Views into desired environments you may trigger several remote actions depending on your setup. Content in environments found on remote nodes will be automatically synchronized. Puppet Masters may be also be re-synchronized when new content is added to environments that they manage.

- In MDP2 the requirement to promote Content Views into environments for subsequent consumption has been relaxed. It is now possible to consume Content Views (via activation keys) directly from the Library context. However, if you choose to use the promotion path for exporting Content Views you must perform this action by creating a "change set" and then specifying the desired Content Views for inclusion. Change sets can be used to progress content along your customized promotion cycle.

8) Create Activation Keys

- Create Activation Keys that entitle your system to the desired repositories. When you create the key, you will have the opportunity to associate it one of your Content Views. Activation keys can also be associated to System Groups for greater efficiency when provisioning numerous new systems.

9) Create an Application Stack

- Satellite 6 uses a "host group" concept to fully define the system you intend to build. You will need to create a host group or reuse one that has already been added to the system. Typically host groups begin with a base group that serves as a foundation to layered groups that in combination define the complete application stack. The base group contains the OS and serves as a parent to layered child host groups that contain specific application components (apache, jboss, etc). By stacking host groups you will be able to define any application stack served by Puppet Classes included in your Satellite 6 library.
- First check to insure that you are in the correct organizational context and use the Environment and Content View selectors to associate the application stack with the content views you prepared in step 6.
- You will need to further select the desired Puppet Classes (from the appropriate tab) to tailor the precise stack from the available list (sourced from the previously selected Content View).
- You are encouraged to explore all the tabs associated with Content Views as well as the "More" menu and insure that there are values provided from your earlier work in organizing your media for building new systems.

10) Provision a machine

- You will need to verify the Smart Proxies you added in step 2 to automate the operation of DHCP, TFTP, and DNS for provisioning, and in doing so, specify one or more subnets and domain names for new systems.
- Select the desired host group or host group stack
- Now you are ready to create a new host on either a physical machine or Compute Resource (Compute Resource is an association to a virtualization provider, e.g libvirt, ovirt, VMware)
- The new host will register back to Satellite 6 for Errata.

What's Included in Red Hat Satellite 6 MDP2?

Red Hat Satellite 6 has been completely redesigned to meet the evolving needs of Enterprise Systems Management. Satellite 6 is built from a collection of highly innovative upstream community projects, including Puppet, Foreman, Katello, Pulp, Candlepin, QPID, and FreeIPA, among others.

Key Features of MDP2

- **Integration with Red Hat Customer Portal**
  - *Feature*: Accepts Manifests created in Portal
  - *Description*: After authentication on the Portal, a Satellite distributor is created and associated with subscriptions. The manifest is then generated and downloaded to the

user's local file system. Thereafter the user can import the manifest into Satellite 6, enable the desired repositories and synchronize the content

- *Feature*: Within a given organization, the user can refresh the number of subscriptions locally on Satellite 6.
- *Description:* This feature obviates the need to generate and export and import a new manifest. The user can synchronize the number of entitlements on Satellite 6 with the number added (or removed) on the customer portal without the need to log on to the portal.

- **New Content Repository Model**
  - *Feature:* Satellite 6 leverages the full capabilities of Pulp to efficiently mirror and distribute RPM and Puppet Module content
  - *Description:* Pulp is designed to replicate software repositories from a variety of supported sources (http/https, file system, ISO) to the Satellite 6 library. Satellite 6 content includes Red Hat content, customer RPM repositories, and Puppet Modules sources from the Puppet Forge or from local GIT repos.
- **New Subscription Management System**
  - *Feature:* Systems managed by Satellite 6 are activated with subscription certificates, which reside in a pool.
  - *Description:* Certificates can be added or removed at anytime and are instrumented for fine-grained reporting.
- **Structuring Content for System Builds**
  - *Feature:* Satellite 6 introduces Content Views as a means to combine repositories, and products with white and black lists, and errata parameterization.
  - *Description*: Once a Content View is defined, it can be published and directly consumed from the library or promoted into an externally accessible environment (URL) and consumed by systems under management. Content Views can be refreshed to reevaluate filters and rules which, in effect, creates a new version of the previous Content View. Content Views can contain either RPM or Puppet Module content types. Content Views can be grouped together into Composite Content Views to fully specify content for both OS, Packages, and Configuration recipes (via Puppet Classes). The Composite Content View can be promoted into environments or accessed directly from the library.
- **Promotion Paths and Change Sets**
  - *Feature:* Content Views are published in the Satellite 6 Library where they can be promoted into externally accessible containers called Environments.
  - *Description:* Environments are represented as a serial progression or path that enables administrators to manage content lifecycle. These paths can be customized and used to direct content from development to production in a structured and persistent manner. Content Views can be refreshed (with new rule evaluation) within an existing Environment which increments its version number. Each promotion activity requires the creation of an explicit "change set" that serves to document content alterations (additions, subtractions, promotions).
- **Federated Nodes for Content and Provisioning**
  - *Feature:* Satellite supports the establishment and management of remote nodes for mirroring content environments and performing local orchestration of provisioning

infrastructure

- *Description*: Federated nodes are managed from the headend Satellite. Environments defined on Satellite can be pro-actively distributed to remote nodes. Content Views promoted into Environments are automatically synchronized across all federated points of presence. Additionally Smart Proxies for managing the provisioning process and optionally supplying DHCP, DNS, and TFTP services are also installed on remote Satellite Nodes scale provisioning and improve performance with affinity to bare metal and VM infrastructure.

- **Activation Keys**
  - *Feature*: Activation Keys are generated on Satellite 6 and placed on managed systems to enable access to Red Hat entitled content as well as custom content. Activation Keys point to Content Views, which have been placed in Environments.
  - *Description*: Activation Keys contain credentials and URLs required by a recipient system to locate and access content managed by Satellite 6. The Activation Key installation is added to the Kickstart process. Activation Keys can also be associated with System Groups.
- **System Definition with Host Groups**
  - *Feature:* Satellite 6 provides an efficient mechanism to combine essential elements of a new system (OS, Disk Partition, NIC configuration, IP address, and Puppet Classes) and Content Views with a dynamic Kickstart file delivered by a Satellite 6 managed PXE process which seamlessly segues the system build to Puppet.
  - *Description*: Satellite 6 Host Groups simplify the task of fully describing a system build. In MDP2 many requisite Host Group resources are automatically populated based on the establishment of Organizations, Content Views, and Environments in the content portion of Satellite 6. Host Groups direct references to Content Views, Activation Keys, disk partition tables, networking and domain information, as well as collections of puppet classes (constrained by the selection of Environment + Content View). Host Groups are used to render a Kickstart File for the build process and define the finished application stack through Puppet automation. Host Groups can be applied to bare metal, virtualization and cloud systems as a means to standardize system builds.
- **Dashboards for Configuration and Subscription Management**
  - *Feature*: Satellite 6 provides dashboards for monitoring system compliance and configuration state.
  - *Description*: For MDP2 system compliance is an important area of concentration and is represented under the Content UI context. Additionally configuration state including success/failure of Puppet Runs, drift management, and Fact trending is rolled up in a click-through dashboard.

TIMELINES AND FEATURE SET COVERED

Feature and functions delineated below are subject to change so please utilize this list as a general guide of directionality across the Managed Design Program.

- **MDP1 June 2013**
  - New Content Management with Content Views

- New Subscription Management with Red Hat CDN
- New Provisioning based on PXE and Kickstart integrated with Content Management

- **MDP2: November 12, 2013**
  - Puppet Integration with Satellite 6, Puppet Installation of Master and Client Components
  - Puppet Master pulls content from Satellite 6 Content Management System
  - Content Distribution through federated points of presence
  - Provisioning and Kickstart distribution through federated points of presence
  - Greater automation between content creation and consumption
  - Improvements to User Experience
  - MDP1 Enhancements and bug Fixes
  - Disconnected Operation whereby the Satellite 6 library can be synchronized by a local file system (e.g. USB thumb drive)
- **MDP3: Q2 2014**
  - Beta Candidate for Satellite 6 General Availability
  - Scalability Improvements
  - IDM and Active directories integration
  - Versioned System Definition
  - MDP2 Enhancements and bug fixes
- **Satellite 6 GA Mid 2014**
  - Operational Hardening
  - Backup & Recovery
  - Highly Available deployment
  - Multiple Language Support
  - Improved Doc Coverage
  - MDP3 Enhancements and bug fixes

INSTALLATION NOTES, KNOWN BUGS AND WORKAROUNDS

INSTALLATION NOTES

The following sections provide changes to or more detailed steps around the installation and configuration document which can be found at

https://access.redhat.com/site/documentation/Red_Hat_Satellite/

Running Satellite 6 as a VM

You can have your Satellite 6 setup in a VM, but still provision standard guests on the host on which you are running your Satellite 6 VM. This will eliminate the need to nest guested with libvirt, which has performance issues.

Satellite 6 Content

To setup your Satellite 6 server for provisioning you need to synchronize and publish content. The

following notes were used by our team during installation. Please feel free to use these, or to reach out to the team as listed above.

- **Import Manifest and Synchronize Content**
  - Login to Satellite 6's WebUI and click on "Content" on the right navigation
  - Create new organization called "Satellite Infrastructure" with single environment to start with (eg: dev)
  - Import a valid RH subscription manifest
  - Enable Red Hat Enterprise Linux 6.4 Server RPMs x86_64 6Server
  - Enable Red Hat Enterprise Linux 6.4 Kickstart RPMs x86_64 6Server
    - **NOTE:** This is **CRITICAL** as it contains the media and kickstart tree used for provisioning
  - Enable Red Hat Satellite Tools 6.0 MDP 2 for RHEL 6 Server RPMs x86_64 6.4
  - Synchronize the above repositories
  - Create new activation key -> node
  - Associate 'node' activation key to 'Library' and 'Default Organization View'
  - Associate "90 Day Self-Supported Red Hat Satellite 6 MDP-2" subscription to the node activation key
- **libvirtd:**
  - We want to be able to provision systems using Satellite 6 MDP2 we want to use libvirt to provision guests. The following steps will guide you on setting up your Satellite 6 server to also host virtual guests.
  - For the sake of simplicity we will configure libvirt without any authentication. This lacks security, but for our testing purposes it will simplify installation and configuration. On the libvirt server make the following changes:
    - **# puppet module install domcleal/katellovirt**
    - **# puppet apply -v -e 'include katellovirt'**
    - (source code at https://github.com/domcleal/katellovirt)
    - This module will configure:
      - libvirt with a NAT network on 192.168.100.0/24, storage pool under /var/lib/libvirt/images
      - iptables permitting HTTP, HTTPS, TFTP, DHCP, DNS, Puppet inbound
      - iptables kernel modules for TFTP conntrack
      - sysctl for IP forwarding
      - To test the connection later on the Foreman guest do the following command (it should not ask for username or password):
      - **# virsh -c 'qemu+tcp://localhost:16509/system' list**
      - Once you have libvirt setup we can use the node-installer to setup the Smart Proxy, DNS and TFTP
- **Node Installer:**
  - Now that you have libvirt functional we need to setup your Satellite Node & Smart Proxy. You need a separate system from your main Satellite 6 server with a clean Red

Hat Enterprise Linux 6 OS installed and operational.

- Once that system is functional you can initiate the node-installer from a root terminal on the host. Note that the OAUTH_SECRET must be obtained from the Satellite 6 server's oauth_token_file

```
# SATELLITE_6=[hostname of satellite 6 server]
# OAUTH_SECRET=$(cat /etc/katello/oauth_token-file)
# FORWARDERS=$(for i in $(cat /etc/resolv.conf |grep nameserver|awk
'{print $2}'); do echo  --dns-forwarders $i; done)

# node-install -v \
 --parent-fqdn "$SATELLITE_6" \
 --dns true $FORWARDERS \
 --dns-interface virbr1 \
 --dns-zone example.org \
 --dhcp true \
 --dhcp-interface virbr1 \
 --tftp true \
 --tftp-servername $(hostname) \
 --puppet true \
 --puppetca true \
 --register-in-foreman true \
 --foreman-oauth-secret "$OAUTH_SECRET" \
 --pulp true \
 --pulp-oauth-secret "$OAUTH_SECRET"
```

Satellite 6 Provisioning

Using Satellite 6 provisioning requires knowledge of how to set up Foreman. This can be varied based on your environment. Our team used the following notes during installation. Please feel free to use these, or to reach out to the team as listed above.

- **Validate that promoted tree has been pushed into Satellite 6 Provisioning**
- **Setup Base Parameters:**
  - More -> Configuration -> Smart Proxies
  - Name: my_smart_proxy
  - Url: http://localhost:9090
  - Click on "Import Subnets"; give discovered subnet (e.g. 192.168.100.0/24) a name: "my_subnet"; domain: "example.org" and submit.
- **Organization: Select the appropriate Organization**
  - Create Subnet
    - More -> Provisioning -> Subnets -> Edit my_subnet
    - DHCP Proxy: my_smart_proxy
    - TFTP Proxy: my_smart_proxy
    - More -> Provisioning -> Provisioning Templates
    - Click the Build PXE Default button
    - More -> Provisioning -> Domain -> New
    - Name: example.org
    - Full Name: example.org

- DNS Proxy: None
- Submit

- Compute Resource
  - More -> Provisioning -> Compute Resources -> New
  - Name: my_libvirt
  - Provider: LibVirt
  - Description:
  - URL: qemu+tcp://localhost:16509/system
  - Test Connection will only provide feedback if an error condition exists.
  - Organizations: Select the appropriate Organization
  - Submit
- Create HostGroup
  - More -> Configuration -> Host Groups -> New
  - Name: my_group
  - Environment ID: Select the published content view from Katello (dev -> my_RHEL6_2)
  - Puppet Master: my_smartp
  - Network
  - Domain: example.org
  - Subnet: my_subnet (192.168.100.0/24)
- Operating System
  - Architecture: x86_64
  - Operating System: Red Hat 6.2
  - Media: RHEL-6.2
  - Partition Table: RedHat default
  - Root pass: if left blank, it will use default 123123. To change the default password then click
  - More -> Settings -> Provisioning -> root_pass
- Parameters
  - Should automatically have the selected content view
- Organizations
  - Organizations: Should be automatically selected to match content view's organization
- Activation Keys
  - Activation Keys: type the name of the activation key created in Satellite 6: Content: "my_key"
  - Submit
- Edit Provisioning Templates
  - Open the More -> Provisioning -> Provisioning Templates
  - Open the "*Katello Kickstart Default for RHEL*"
  - Under the <%= snippets "subscription_manager_registration" %> section add: `subscription-manager repos --enable=rhel-server-6-sat-tools-6-mdp-2-rpms`

- Associate HostGroup with Template
  - More -> Provisioning -> Provisioning Templates
  - Edit Katello Kickstart Default
  - Associate with my_host_group
  - Submit
  - Create Host
  - Hosts -> New Host
  - Name: my_guest
  - Deploy On: my_libvirt
  - HostGroup: my_hostgroup
  - Virtualization -> Interface: virbr1
  - Submit
- VNC Console
  - If host VNC console is not working -> http://projects.theforeman.org/projects/1/wiki/NoVNC