Click to prove you're human



```
Do you need to know hammer commands for satellite exam
For Satellite 6.2 Hammer is a command-line tool provided with Red Hat Satellite 6. You can use Hammer to configure and manage a Red Hat Satellite Server through either CLI commands. See the Hammer CLI Guide for more
 information on Hammer. General information --help Display hammer commands and options, append after a subcommand to get more information with: hammer defaults add --param-name \ organization id --param-value org ID (Satellite 6.2 only) loc
The setting is location-specific, append --location loc name, or set default loction with: hammer defaults add --param-name location id \ --param-value loc ID (Satellite 6.2 only) Organization create \ --name organizati
organizationshammer organization list location See the options for organization subscription manifesthammer subscription upload \ --file path repository-set enable \ --product "prod name" \ --basearch "base arch" \ --releasever "rel v" \ --name "repository-set organization subscription manifesthammer subscription upload \ --file path repository-set organization subscription su
Synchronize a repository hammer repository synchronize \ --product "prod name" \ --name "repo name" \ --product "prod name" \ --product \ 
product "prod name" \ --id "repo id" --path "path to dir" Content life cycle environment \ create --name env name \ --description "env desc" \ --prior prior env name List life cycle environment \ list content-
 vieworg Create a content viewhammer content-view create \ --name cv n \ --repository-id repo ID Add Puppet modules to a content viewhammer content-view \ puppet-module add \ --content-view cv n \ --repository-id repo ID Add Puppet modules to a content viewhammer content-view \ puppet-module add \ --content-view cv n \ --
name module name Publishing a content-view version \ incremental update of a content-view version \ incremental-update \ --content-view version-id \ cv ID --packages
pkg n1,...\--environment-ids env ID1,...\--location-ids org ID1,...\--location-ids org ID1,...\--domain-ids dom ID1,...\--domain-ids dom ID1,...\--boot-mode
 boot mode \ --network network address \ --mask netmask --ipam ipam compute-resourceorgloc Create a compute resource accompute resource route resource router provider name medium Add an installation medium medium reate \ --name medium reate \ --name med name \ --path
path to medium partition-table Add a partition table hammer partition-table create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create --name to reate \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name\ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name \ --path path to file \ --os-family os family template os Add an operating systemhammer os create \ --name tab name \ --path path to file \ --os-family os family tab name \ --path path to file \ --os-family os family tab name \ --path path to file \ --os-family os family tab name \ --path path to file \ --os-family os family tab name \ --path path to file \ --os-family os family \ --os-family \ --os-fa
 Subcommand Description and Tasks activation-keyorg Create an activation-key reate \ --name ak name \ --content-view cv n \ --lifecycle-environment lc name Add a subscription id sub ID Users and Permissions Subcommand Description
and Tasks userorg Create a userhammer user create --login user name \ --mail user mail --auth-source-id 1 \ --organization-ids org ID1,org ID2... Add a role to a userhammer user grouphammer user grouphammer user grouphammer user grouphammer user grouphammer user.
id ug id \ --role role name role Create a rolehammer role create a rolehammer role create a filter create a fi
erratumhammer erratum info --id err ID host List errata applicable to a hosthammer host errata list \ --host host name \ --errata-ids err ID1,err ID2... Docker containers Subcommand Description and Tasks docker Create a container hammer docker container create \ --name
container name \ --compute-resource-id cr ID \ --repository-name repo name \ --tag tag --command Command Comma
 "arch name" \ --domain domain name \ --subnet subnet name \ --puppet-proxy proxy name \ --puppet-proxy proxy name \ --puppet-ca-proxy name \ --pup
hostgroup "hg_name" \ --name "kt_activation_keys" \ --value key_name hostorgloc Create a host (inheriting parameters from a host group)hammer host create \ --name "host group)hammer host create \ --name "kt_activation_keys" \ --value key_name hostorgloc Create a host (inheriting parameters from a host group)hammer host create \ --name "host group)hammer host group)hammer host group | hos
yes job-template Add a job template name" \ --job-template name \ --job-temp
 jobhammer job-invocation output \ --id job id --host host name Tasks Subcommand Description and Tasks task List all taskshammer task list Monitor progress of a running taskhammer task progress --id task ID For Satellite 6.3 Hammer is a command-line tool provided with Red Hat Satellite 6. You can use Hammer to configure and manage a Red Hat
 Satellite Server through either CLI commands or automation in shell scripts. The following cheat sheet provides a condensed overview of essential Hammer CLI GuideFormatMulti-pageSingle-pageView full doc as PDFAbstract This document describes how to use the Hammer CLI tool to
 configure and manage Red Hat Satellite. Hammer is a powerful command-line tool provided with Red Hat Satellite Server either through CLI commands or automation in shell scripts. Hammer also provides an interactive shell. Hammer compared to Satellite web UI Compared to
navigating the web UI, using Hammer can result in much faster interaction with the Satellite Server, as common shell features such as environment variables and aliases are at your disposal. You can also incorporate Hammer commands into reusable scripts for automating tasks of various complexity. Output from Hammer commands can be
redirected to other tools, which allows for integration with your existing environment. You can issue Hammer commands directly on the base operating system is required to issue Hammer commands, which can limit the number of potential users compared to the web UI.
Although the parity between Hammer and the web UI is almost complete, the web UI has development priority and can be ahead especially for newly introduced features. Hammer and Satellite API are equally applicable. Hammer can be used as a human friendly interface to Satellite API, for
example to test responses to API calls before applying them in a script (use the -d option to inspect API calls issued by Hammer, while scripts using the API directly have to be updated manually. In the background, each Hammer command first
establishes a binding to the API, then sends a request. This can have performance implications when executing a large number of Hammer commands in seguence. In contrast, a script communicating directly with the API establishes the binding only once. See the Red Hat Satellite API Guide for more information. View the full list of hammer options
and subcommands by executing: $ hammer --help Use --help to inspect any subcommand, for example: $ hammer prompts for your Satellite credentials each time you issue a command. You can specify your
credentials when executing a command as follows: $ hammer -u -p As an alternative, follow these steps to use saved credentials: Create the file: :foreman: :host: ':username: 'username': password: 'password' Replace the example values with your own details. Do not use tabs in the
file, always use indentation by spaces. To protect your password, make sure the file is readable only by the current user: $ chmod 600 ~/.hammer/cli config.yml file. Use only spaces for indentation in Hammer configuration files. Do not use
tabs for indentation in Hammer configuration files. Examples in this guide assume saved credentials. It is possible to install hammer individually on a server where there is no Satellite installed, and use it to connect the server to a remote Satellite. The rhel-X-server-satellite installed, and use it to connect the server to a remote Satellite.
 workstation to connect should install the repository manually, see the Red Hat Satellite Installation Guide. Install hammer. # yum install tfm-rubygem-hammer cli katello Edit /etc/hammer/cli.modules.d/foreman.yml to point to the desired Satellite. Many hammer commands are organization specific. You can set a default organization, as well as
location, for hammer commands so that you do not have to specify it every time using the --organization id --param-name organization id --param-name organization
name location id --param-value To view the currently specified default settings, issue the following command: $ hammer defaults list Specifying a default organization, you still have to use a command-line
option to specify it. Examples in this guide do not assume a saved default organization, instead they use the shell variable approach described in Note. The default location for global hammer configuration files. You can set user
specific directives for hammer (in ~/.hammer/cli_config.yml) as well as for CLI modules (in respective .yml files under ~/.hammer/cli_modules, issue: $ hammer -d --version Loading configuration for many CLI modules can slow down the execution of
 ~/.hammer/cli config.yml. Note that these directives affect only the current user and are not applied globally. :log level: 'warning' :log size: 5 #in MB Similarly, you can set the number of lines displayed at once in the hammer output (equivalent of the --per-page option): :per-page: 30 You can set hammer to log debugging information for various
Satellite components. You can set debug or normal configuration options for all Satellite components, use the following command: # hammer admin logging --all --level-debug # katello-service restart To set debug level for all components. You must restart To set debug or normal configuration options for all Satellite services. # katello-service restart To set debug level for all components.
production level logging, use the following command: # hammer admin logging --all --level-production # katello-service restart To list the currently recognized components, that you can set logging --help Usage: hammer admin logging --level-production # katello-service restart To list the currently recognized components, that you can set logging --help Usage: hammer admin logging --help Usage: hammer admin logging --list To list all the available options of this tool: # hammer admin logging --help Usage: hammer admin logging --help Usag
 You can issue hammer commands through the interactive shell. To invoke the shell, issue the following command: $ hammer shell In the shell, you can enter subcommands directly without typing "hammer", which can be useful for testing commands before using them in a script. To exit the shell, type exit or press [Ctrl + D]. You can modify the
 default formatting of the output of hammer commands to simplify the processing of this output by other command line tools and applications. For example, to list organizations in a CSV format with a custom separator (in this case a semicolon), issue the following command: $ hammer --csv --csv-separator ";" organization list Output in CSV format is
table — generates output in the form of a human readable table (default). base — generates output in the Comma Separater output in the YAML format. csv — generates output in the YAML format. csv — generates output in the Comma Separater output in the YAML format.
JavaScript Object Notation format. silent — suppresses the output. You can use the hammer ping command to check the status of core Satellite issues. If all services are running as expected, the output looks as follows: $ hammer ping
 locations, and repositories. For web UI equivalents of the following procedures see Configuring Organizations, Locations and Life Cycle Environments in the Red Hat Satellite Server Administration Guide. Organization in Red Hat Satellite Server Administration Guide.
 shows how to create and modify organizations using hammer. Use the following command to create an organization: $ hammer organization using hammer organization using hammer. Use the following command to create an organization using hammer organization using hammer. Use the following command to create an organization using hammer organization using hammer. Use the following command to create an organization using hammer organization using hammer.
cannot contain white space and you cannot change them later. If not specified, label is generated automatically from the organization. This parameter is not required, but it can help you to manage a large number of organizations. You can fully configure an
organization while creating it (issue hammer organization create an organization update command. Example 2.1. Creating and Updating ACME organization using the hammer organization update command. Example 2.1. Creating and Updating ACME organization using the hammer organization update command.
organization create \ --name $ORG \ --compute resource to the organization: $ hammer organization update \ --name $ORG \ --compute resource to the organization update \ --name $ORG \ --compute resource to the organization update \ --name $ORG \ --compute resource to the organization update \ --name $ORG \ --compute resource to the organization update \ --name $ORG \ --compute resource ids 1 Many tasks you can perform in the Satellite Server are specific to an organization.
organization: by using the organization, organization ist If your organization list If your orga
Team" $ hammer product list --organization $ORG This approach is used in example: $ hammer defaults add --param-value 1 With the above setting, organization specific commands will assume --organization-id 1 is
 specified, so you no longer have to type it. A Subscription Manifest transfers subscriptions from the Red Hat Customer Portal to Satellite Content Management Guide. Then upload the manifest to the organization as follows: $ hammer
 subscription upload \ --organization-label \ --file Example 2.2. Uploading a Manifest to ACME Organization The following example shows how to upload --organization upload --organization $ORG --file /tmp/manifest.zip To view
the subscriptions imported with the manifest, issue: $ hammer subscription list --organization $ORG Location in Red Hat Satellite is collection of default settings that represent a physical place. This section shows how to create a location in Red Hat Satellite is collection of default settings that represent a physical place.
 Example 2.3. Creating Multiple Locations Using a Script The following Bash script creates three locations (london, munich, boston), and assigns them to the ACME organization. ORG="ACME" LOCATIONS="london munich boston" for LOC in ${LOCATIONS} do hammer location create --name "${LOC}" hammer location add-organization --name
 "${LOC}" --organization "${ORG}" done Run hammer location --help to view all possible location related operations. Repository provides storage for a collection of content. This section shows how to enable and synchronize repository provides storage for a collection of content. This section shows how to enable and synchronize repository provides storage for a collection of content.
 provides, the base architecture, and the release version. Use the following command to enable a repository-set enable \ --product "" \ --name ""Example 2.4. Enabling a Red Hat Enterprise Linux Repository The following command enables the Red Hat Enterprise Linux 7 Server
 repository for the organization: $ hammer repository-set enable \ --organization $ORG \ --product "Red Hat Enterprise Linux 7 Server" \ --basearch "x86_64" \ --releasever "7 
 By synchronizing a repository you pull its content from Red Hat Customer Portal to the Satellite Server. To synchronize \ --product "" \ --name "" \ --organization-label \ --async Note that if you have created Content Views, multiple repositories with the
 same name can exist within a single organization. In such a case, use the --id option to identify the repository you want to synchronize (issue hammer repository The following command performs a single synchronization of the Red Hat Enterprise Linux 7
 Server repository in the organization: $ hammer repository synchronize \ --product "Red Hat Enterprise Linux 7 Server" \ --name "Red Hat Enterprise Linux 7 Server (RPMs)" \ --organization $0RG \ --async The task ID is displayed after executing the above command: Repository is being synchronized in task 640bb71f-0ce5-40a3-a675-425a4acacceb To
view the progress of the task, issue: $ hammer task progress --id 640bb71f-0ce5-40a3-a675-425a4acacceb After finishing the first synchronizetion, the repository is added to the list of repository is added to the list of repository is added to the list of repository is added to the list.
 repositories within a product as follows: $ hammer product synchronize \ --organization-label \ --name "" \ --async With the --async option, the repositories in parallel. Example 2.6. Synchronizing All Repositories in ACME Organization
The following Bash script synchronizes all repositories within the ACME organization. ORG="ACME" for i in $(hammer --csv repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $ORG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi '^ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi ''ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi ''ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization $0RG | grep -vi ''ID' | awk -F, {'print $1'}) do hammer repository synchronize --id ${i} --organization ${i} --organization ${i} --organization ${i} --organization ${i} --organization ${i} --organization ${i} --or
 smallest unit of the synchronization process. You can create a synchronization plan to automatically update repositories of a selected product in a given time interval. To define a synchronization plan, issue the following command: $ hammer sync-plan create \ --name "" \ --enabled=true \ --interval \ --organization-label \ --sync-date "" Replace with
hourly, daily, or weekly. Replace with the date and time of the initial synchronization in the form of "YYYY-MM-DD HH:MM:SS". Example 2.7. Creating a Synchronization, that runs at 3 a.m., starting from 15 January 2016: $ hammer sync-plan create \ --
 name "daily sync at 3 a.m." \ --enabled=true \ --interval daily \ --organization $ORG \ --sync-date "2016-01-15 03:00:00" To associate the synchronization plan with a product, issue the following command: $ hammer product set-sync-plan \ --organization plan to Multiple Products.
The following Bash script selects the products in the ACME organization that have been synchronized at least once and contain at least one repository and assigns them a synchronization plan named "daily sync at 3 a.m.". ORG="ACME" SYNC_PLAN="daily sync at 3 a.m." for i in $(hammer --csv product list --organization $ORG --per-page 999 | grep
vi '^ID' | grep -vi not synced | awk -F, {'{ if ($5!=0) print $1}'}) do hammer product set-sync-plan $SYNC PLAN --organization $ORG --id $i done After executing the script, issue the following command to see which products have been assigned the synchronization plan: $ hammer product list --organization $ORG --sync-plan "daily sync at
3 a.m." To view synchronization plans available for a selected organization, issue the following command: $ hammer sync-plan --help and hammer product --help. After enabling a Red Hat repository, the corresponding
product is created automatically. To enable a repository with custom packages, you first need to manually create a product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product: $ hammer product create --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates a new repository under the custom product creates --name "" --organization-label The following command creates --name "" --organization-label The f
 repository create \ --name "" \ --organization-label \ --product "" \ --content-type --publish-via-http true \ --url "" Replace the example values with your own details, in particular: specifies the URL on which the repository will be available, valid only if \cdot --publish-via-http
is enabled. To upload packages to a custom repository, issue the following command: $ hammer repository upload-content \ --product "" \ --path Replace with the path to the directory with content (RPM packages, Puppet modules, or Docker images) to be added to the custom repository. This section shows how to use
 hammer to create Content Views and to promote them through life cycle environments. Life cycle environments represent stages of the content life cycle. This section shows how to view and create life cycle environments with hammer. By default, the Library environment is present for each organization. Use the following syntax to create a new life
cycle environment: $ hammer lifecycle-environment triangle a Life Cycle Environment triangle a Life Cycle Environment This example shows how to create a new environment triangle a Life Cycle Environment triangle a Life Cycle Environment This example shows how to create a new environment triangle a Life Cycle En
environment create \ --name Development \ --description "Initial testing" \ --organization $ORG \ --prior Library You can create another life cycle environment based on Development using the --prior option. To view existing life cycle environment list --organization $ORG \ --prior option.
                                                                                          the Library created by intelligent filtering. You can publish and promote Content View, issue the following command: $ hammer content-view create \ --name \ --repository-ids ,, \ --description "" \ --organization-label
The --repository-ids option adds the selected repositories to the Content View, use the hammer repository list command to find the IDs. It is also possible to omit this option adds the selected repository subcommands. Example 3.2. Creating a Content View The following example creates
a Content View under the ACME organization and assigns it three repositories: $ hammer content-view create \ --name cv-rhel7-server \ --repository-ids 1,2,3 \ --description "Initial CV for RHEL 7" \ --organization $ Composite Content View A Composite Content View A Composite Content View Content View Content View Content View A Composite Conten
 shows how to create a Composite Content View from two existing Content Views: $ hammer content-view create \ --component-ids 2,6 Find the IDs for the --component-ids option by executing hammer content-view list. There are
three content types you can add to the Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View: $ hammer content-view update \ --repositories to an existing Content View update \ --repositories to an existing
Note that it will overwrite any existing repository \ --organization-label \ --name \ --repository from the Content View. Use hammer content-view info to
 inspect repositories in a Content View. Example 3.4. Filtering Packages for a Content View - Excluding a Package Filters allow you to select a subset of packages from a repository (either by including or excluding to create customized Content Views. This example shows how to create a filter to exclude the emacs package from the cv-rhel7-server
 Content View. First create a filter for the Content View in the organization: $ hammer content-view filter create \ --type rpm \ --name exclude-emacs \ 1,2,3 \ --content-view cv-rhel7-server Find the repository IDs by executing hammer repository list
 Create a rule to exclude packages with name starting with "emacs" and add it to the filter as follows: $ hammer content-view filter rule create \ --content-view filter exclude-emacs As a result, hosts using the cv-rhel7-server Content View will not have access to the emacs package
 You can add multiple rules to a filter, see the output hammer content-view rule create --help for the full list of filtering parameters. To inspect rules present in a filter, see the output hammer content-view some $\fontarrow$. Filtering parameters. To inspect rules present in a filter, issue the following command: $\fontarrow$ hammer content-view filter rule list \\ --content-view filter rule lis
 Packages for a Content View - Limiting Errata by Date This example shows how to create a filter to exclude errata released before a specific date from the cv-rhel7-server Content View as follows: $ hammer content-view filter create \
type erratum \ --name limit-errata-by-date \ --description "Excluding errata by date" \ --inclusion false \ --organization $ORG \ --repository-ids 1,2,3 \ --content-view cv-rhel7-server Create a rule to exclude errata by date" \ --end-date \ --organization
 $ORG \ --content-view cv-rhel7-server \ --content-view first upload this module to a Puppet repository within a custom product. Use the commands from Section 2.3.4, "Creating a Custom Repository" to create a product with a repository, and to
 upload a Puppet module to the repository. To add a Puppet module to a Content View, issue the following command: $ hammer content-view puppet-module add \ --content view and a Puppet module from an external source to the cv-rhel7-server Content
 View. Download the concat module (that constructs files from multiple text fragments) from Puppet Forge: $ wget -O /tmp/puppetlabs-concat-1.2.3.tar.gz (reate a Puppet product and upload the module to this repository (the example assumes repository ID is 6): $ hammer
product create \ --name "ACME-puppet" \ --organization $ORG\ hammer repository upload-content \ --organization $ORG\ --product ACME-puppet \ --id 6 \ --path /tmp/puppetlabs-concat-1.2.5.tar.gz Add the
 module to the Content View using the id, name, or author parameters. To find the exact values, enter: $ hammer puppet-module list --organization $ORG ---|------- I | concat | puppetlabs | 1.2.3 ---|------- To add the module to the Content View, issue: $ hammer
 content-view puppet-module add \ --name concat \ --content-view cv-rhel7-server \ --organization $ORG You can upload Docker images directly to the dedicated repository
 as follows: $ hammer repository upload-content --path --id Replace with a path to the archive containing the Docker image. Use to identify the repository of docker content View you make it visible and usable by hosts. Use the following command to publish a selected
Content View: $ hammer content-view publish \ --id \ --organization-label \ --async Find the of the Content Views become available in the Library environment. To verify the Content View status, issue the following command: $ hammer content-view info --
id Promoting is the act of moving a Content View from one life cycle environment to another. To do so, issue the following command. $ hammer content-view \--content-view to another. To do so, issue the following command. $ hammer content-view \--content-view \--content-
 Life Cycle Environment Path The following Bash script promotes the selected Content View from Library through all life cycle environments in the ACME organization $ORG | grep -vi '^ID' | awk -F, {'print $1'} | sort -n) do hammer content-view version promote -
organization $ORG --to-lifecycle-environment-id $i --id $CV_ID done To verify if the Content View has been promoted correctly, issue the following command: $ hammer content-view version info --id 1 Incremental updates enable modifying a published Content View without the need to promote a new Content View version through the life cycle
environment. As a result of the incremental update, a new minor Content View version is created. Incremental updates are useful for fast emergency updates, you can use them to add errata, packages, or Puppet modules. To create an incremental update adding new packages to a Content View, issue: $ hammer content-view version incremental updates.
 update \ --content-view-version-id \ --packages , \ --lifecycle-environment-ids , ,... Find the Content View version ID in the output of hammer Content View version list. Instead of supplying packages with the --packages option, you can add Puppet modules with --puppet-modules, or errata with --errata-ids (see Example 3.8, "Adding Errata to a Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version ID in the output of hammer Content View version 
 View using an Incremental Update"). For more information on working with incremental updates issue hammer content-view version incremental update --help. Example 3.8. Adding Errata to a Content View using an Incremental update --help. Example 3.8. Adding Errata to a Content View using an incremental update --help.
 update of its Content View: $ hammer content-view version incremental-update \ --content-view-version-id 4 \ --errata-ids 8c3801f6-12a7-4a62-83f4-addbb1f34ce6 \ --lifecycle-environments Infrastructure To find the required information for the above command, perform the following steps: Find the Content-View your host is registered to as well as its
life cycle environment by executing: $ hammer content-host info --name auth01.example.com --organization $ORG Then find the current version of the Content View name RHEL7_Infra" --organization $ORG Find the IDs of errata you want to apply in the list of applicable
errata in Library: $ hammer erratum list --content-view "RHEL7 Infra" --organization $ORG $ hammer host errata list --host auth01.example.com Activation keys define the subscription properties of a host. Using an activation keys define the subscription properties of a host. Using an activation keys define the subscription properties of a host. Using an activation keys define the subscription properties of a host.
Host Configuration Guide. There are three possible use cases for activation keys are only used when hosts are registered with the amended activation key, it is only applicable to hosts that are registered with the amended activation key.
 issue the following command: $ hammer activation-key create --name \ --organization-label \ --content-view \ --lifecycle-environment Note that the Content View has to be published. To see the full list of operations related to activation key, use the hammer activation-key --help command. To add a subscription to the activation key, issue the following
 command: $ hammer activation-key add-subscription \ --id \ --subscription list. Example 4.1. Creating an Empty Activation Key This example shows how to create an activation key that directs the associated hosts to automatically attach
a best fitting subscription: $ hammer activation-key create \ --name "automatically attach key" \ --organization $ORG \ --content-view cv-rhel7-server Content View. In case
of conflicting settings, the values from the last specified key take precedence. You can specify the order of precedence by setting a host group parameter \--name kt activation keys \--value, ,... \--hostgroup This section shows how to configure various stages of your provisioning environment using
 associate them with parameters and Puppet variables. To create a new domain, issue the following command: $ hammer domain create --name You can associate the newly created domain to organizations and locations using the hammer domain create --name You can associate the newly created domain commands. To view the status of a domain, issue the
following command: $ hammer domain info --name Subnets in Red Hat Satellite define networks specified for groups of systems. Subnets use standard IP-addresses to systems within the subnet. The following command contains the
 minimal set of options required for subnet create \ --name \ --roganization-ids ,... \ --location-ids ,... \ --boot-mode \ --network \ --name \ --roganization-ids ,... \ --boot-mode \ --network \ --name \ --roganization-ids ,... \ --boot-mode \ --name \ --ipam Here, is one of DHCP, is 
 therefore it is rarely needed to create them manually (even though hammer provides this option). Compute resources to provision virtual machines and containers. Run the following command to create a compute resource: $ hammer compute-
resource create \ --name \ --organization-ids ,... \ --location-ids ,... \ --provider Here, is one of: RHEV, RHEL OpenStack Platform, Libvirt, Docker, Rackspace, Google, EC2, or VMware. Depending on the provider type, other options such as --url, or --user may be required. See the output of the hammer compute-resource create --help command for
details. Installation media (ISO images) provide content for kickstart trees and new host installations in Red Hat Satellite. To list the medium create --name --path You can make the medium available to organizations and
locations directly when adding it (see the output of the hammer medium create --help command), or later by using the hammer location add-medium or hammer location add-medium create and file systems. Red Hat Satellite provides default partitions and file systems.
tables associated with operating system families, to view them, issue the following command: $ hammer partition-table create \ --name \ --file \ --os-family See the output of the hammer partition-table --help command for other subcommands. Provisioning
templates provide the systematic means to run unattended installations. To view the provisioning template --help command: $ hammer template create --name --file See the output of the hammer template --help command for other
subcommands. Operating systems define combinations of installation methods and media and are grouped within families. As a default, Red Hat Satellite uses a Red Hat family. Families allow Satellite to change certain behaviors when provisioning hosts. To list operating systems, issue the following command: $ hammer os list To create a new
operating system, issue the following command: $ hammer os create --name --major Then you can add architectures, partition tables, installation media, and configuration templates to the operating Systems The following Bash script
assigns each operating system a partition table (Kickstart default), configuration template (Kickstart default PXELinux), and provisioning template (Satellite Kickstart default). PARTID=$(hammer --csv partition-table list --per-page=1000 | grep "Kickstart default PXELinux")
cut -d, -f1) SATID=$(hammer --csv template list --per-page=1000 | grep "provision" | grep "Satellite Kickstart Default" | cut -d, -f1) for i in $(hammer --csv os list | grep -vi '^ID' | awk -F, {'print $1'}) do hammer partition-table add-operatingsystem --id="${PXEID}" -
-operatingsystem-id="${i}" -config-template --id="${SATID}" done You can add grep commands to the for statement to further specify the affected
operating systems. To verify if the assignment was performed correctly, use the hammer os info command. Parameters define the behavior of Red Hat Satellite Host Configuration Guide for details. Use the following example to set a global parameter: $ hammer
--- firewall | --disabled ------|---
                                                                                                 --- Similarly, you can use hammer to set other parameter types: To set domain parameters, use: $ hammer hostgroup set-parameter \ --name \ --value \ --hostgroup To set host parameters, use:
$ hammer host set-parameter \ --name \ --value \ not refers to any physical or virtual system Red Hat Satellite manages. This section shows how to create and configure hosts and host groups using hammer. For web UI equivalents of the following procedures
see the Red Hat Satellite Host Configuration Guide. A host group is a collection of hosts or host groups. It is recommended to create host groups in a
hierarchical manner. The following command demonstrates a basic set of options for creating a host group: $ hammer hostgroup create \ --name "" \ --environment "" \ 
hostgroup create --help for the full list of configurable options. There are two settings that cannot be configured during host group set-parameter \ --hostgroup "" \ --name "kt activation keys has to be added afterwards using: $ hammer hostgroup set-parameter \ --hostgroup set-par
Chapter 4, Managing Activation Keys for details on activation keys). The root password has to be specified when adding a host to the host group for each life cycle environment. MAJOR="7" OS=$(hammer --output csv os list | awk -F ","
"/RedHat ${MAJOR}/ {print \$2;exit}") ARCH="x86_64" ORG="ACME" LOCATIONS="london,munich" PTABLE_NAME="ptable-acme-os-rhel-server" DOMAIN="example.com" hammer lifecycle-environment list --organization "${ORG}" | awk -F "|" '/[[:digit:]]/ {print $2}' | sed s'/ //" | while read LC_ENV do if [[ ${LC_ENV}] == "Library" ]]; then
continue fi LC ENV LOWER=$(echo ${LC ENV} | tr '[[:upper:]' '[[:lower:]]') ParentID=$(hammer --output csy hostgroup list --per-page 999 | awk -F"." "(\$3 ~ /^${LC ENV LOWER}$/) {print \$1}") hammer hostgroup create --name "rhel-${MAJOR} server-${ARCH}" \ --medium
"${ORG}/Library/Red_Hat_Server/Red_Hat_Enterprise_Linux_${MAJOR}_Server_Kickstart_${ARCH}_${ARCH}_$ --operating system "${OS}" \ --partition-table "${PTABLE NAME}" \ --subnet "${DOMAIN}" \ --operating system "${ORG}" \ --locations
"${LOCATIONS}"\--content-view "cv-os-rhel-${MAJOR}Server"\--environment-id $(hammer --output csv environment-id $(hammer --output csv hostgroup list --per-page 999 | awk -F"," "(\$3 ~ /^${LC ENV LOWER}\/rhel-${MAJOR}Server-
$\{ARCH\}\"\) \{\print \\$1\"\}\\ hammer hostgroup set-parameter \ --hostgroup-id "$\{HgID\}\"\ --name "kt activation keys\\ \--value "act-\$\{LC ENV LOWER\}-os-rhel-\$\{MAJOR\}\ server-\$\{ARCH\}\"\ done It is recommended to set general parameters in a host group to reduce the number of required options when creating a host. The following command creates
a basic host associated to a host group: $ hammer host create \ --name "" \ --interface="primary=true, \ provision=true, \ mac=, \ ip=" \ --organization-id \ --ask-root-password yes After executing the above command you will be prompted to specify the root password. It is required to specify the host's IP and MAC address,
other properties of the primary network interface can be inherited from the host group or set using the --interface option, which accepts a list of key-value pairs. For the list of available interface settings, see Table 6.1, "Available Keys for the --interface Option". If you decide
to create a host without host group membership, specify additional options described in Section 6.1, "Creating a Host Group". There is a wide range of available host parameters, for details see the output of hammer host create --help. The value of certain parameters depends on the type of compute resource the host is provisioned on, see Table 6.2,
 "Host Options Specific to Provider" for reference. Table 6.1. Available Keys for the --interface OptionKeysDescription type Defines the interface. mac, ip, domain (or domain id), subnet (or subnet id) Network settings, domain and subnet identification can
be inherited from the host group, primary, provision, managed, virtual interfaces to which the virtual interface belongs, for example
eth1. Specific to bonded interfaces mode Bonding mode, one of balance-rr, active-backup, balance-alb. Specific to BMC interfaces provider BMC provider, set to IPMI. username, password BMC access credentials. Specific to hosts provisioned on Libvirt compute type Interface type, one of bridge,
network. compute network or compute bridge Specifies interface name, pick one depending on the interface name, pic
RHEV. Specific to hosts provisioned on VMware compute type Type of the network adapter, depends on your version of vSphere. compute type Type of the network Network ID form VMware. Table 6.2. Host Options Specific to ProviderProviderKeys Keys for the --compute attributes option EC2 flavor id, image id, availability zone, security group ids, managed ip
GCE machine type, image id, network, external ip Libvirt cpus, memory, start VMware cpus, corespersocket, memory mb, cluster, template, cores, memory, start VMware cpus, corespersocket, memory mb, cluster, path, guest id, scsi controller type, hardware version, start Keys for the --volume option Libvirt
poll name, capacity, format type RHEV size gb, storage domain, bootable VMware datastore, name, size gb, thin, eager zero Example 6.2. Creating a Host with a Bonded Interface Pair The following example shows how to create a host with a Bonded Interface Pair The following example shows how to create a host with a Bonded Interface Pair The following example 6.2. Creating a Host with a Bonded Interface Pair The following example 6.2.
Networking Guide. $ hammer host create --name bondtest \ --interface="identifier=eth1, \ mac=52:54:00:14:92:2a \ --subnet-id=1 \ \ --interface="identifier=eth1, \ mac=52:54:00:d3:87:8f, \ ac=52:54:00:d3:87:8f, \ ac=52:54:
managed=true, \type=Nic::Managed, \domain id=1, \subnet id=1"\--interface="identifier=bond0, \ip=172.25.18.123, \type=Nic::Bond, \mode=active-backup, \attached devices=[eth1,eth2], \attached devices=[eth2,eth2], \attached devices=[eth2,eth2], \attached devices=[eth2,eth2], \attached devices=[eth2,eth2], \attached d
group of hosts. The following command shows the minimal set of options required to create a host collection create \ --organization-label \ --name To add hosts to a host collection, issue the following command: $ hammer host-collection create \ --organization-label \ --incommand shows the minimal set of options required to create a host collection.
with an activation key (see Chapter 4, Managing Activation Keys for details on activation key add-host-collection \--id \--host-collection Hosts grouped in the host collection how inherit the configuration from the activation key. The remote execution feature enables defining arbitrary commands on the Satellite Server and
executing them on remote hosts. Commands are defined in job templates that are similar to provisioning templates are included by default, you can use them or define a custom template for example to manage software packages or start a Puppet process on remote hosts. For more information about remote execution on
Satellite, see the Running Jobs on Satellite Hosts section of the Host Configuration Guide. To use this feature in Hammer, install the remote execution To list job templates available, issue: $ hammer job-template list To create a
job template using a template-definition file, use a command as follows: $ hammer job-template create \ --file "" \ --name "" \ --provider-type SSH \ --job-category "" Replace with the path to the file containing the template definition. Specify a custom or select one of the existing categories (Commands, Katello, Packages, Power, Puppet, or Services). See
the output of hammer job-template create --help for information on other available parameters. To invoke a job with custom parameters, issue: $ hammer job-invocation create \ --job-template create --help for information on other available parameters, issue: $ hammer job-invocation create \ --job-template create \ --j
pairs. Run hammer job-template info to see what parameters are required by your template. Replace with the filter expression defining which hosts will be affected (for example "name ~ rex01"). Example 6.3. Starting the httpd Service on Selected Hosts This example shows how to execute a remote job based on the default Service Action - SSH
Default template, that will start the httpd service on hosts that have a name that contains "target". $ hammer job-invocation create \ --job-template "Service Action - SSH Default" \ --inputs service = "httpd", action = "start" \ --search-query "name ~ target". $ hammer job-invocation output \ --id \ --host Find the in the
output of hammer job-invocation list. For more information on executing remote commands with hammer, issue hammer job-invocation --help or hammer job-invocation --help. For the administrator, Red Hat Satellite provides the ability to create, modify, and remove users. Also, it is possible to configure access permissions through assigning roles to
users. This section shows how to perform these tasks using hammer. For web UI equivalents of the following procedures see Users and Roles in the Red Hat Satellite Server Administration Guide. User in Red Hat Satellite defines a set of details for individuals using the system. To configure a user in Red Hat Satellite, hammer provides the user create
and user update commands. Create a new user with the following command: $ hammer user create \ --login \ --auth-source-id 1 setting means that the user is authenticated internally, you can specify an external authentication source as an alternative. Add the --admin option to grant
administrator privileges to the user. Specifying organization IDs is not required, you can modify the user details later using the update subcommands see the output of hammer user --help. You can manage permissions of several users at once by organizing them into user groups. User groups
themselves can be further grouped to create a hierarchy of permissions. Use the following command to create a new user group; $ hammer user-group add-user --user group issue the following command: $ hammer user-group add-user group.
add user groups using the add-user-group subcommand. For more information on operations related to user groups see the output of hammer user-group --help. Roles in Red Hat Satellite define a set of permissions and access levels. Satellite provides a number of predefined roles, to view them, enter the following command: $$ hammer role list To
view permissions associated with a role, issue the following command: $ hammer role filters --id Here, is the ID of the role from the output of hammer role issue the following command: $ hammer rol
```

permission-ids,... Find the permissions to be added to the role by using hammer filter available-permissions to selected instances

of a resource type. Use the --search option to limit permission filters, for example: \$ hammer filter create \ --permission-ids 91 \ --search "name ~ ccv*" \ --role qa-user The above command adds to the qa-user The above c Permission Filtering in the Satellite Server Administration Guide for more information. To assign a role to a user group: \$ hammer user-group add-role --id --role Software packages in Red Hat products are subject to updates, referred to as errata, that are released at regular intervals as well as asynchronously. This section shows how to inspect and apply errata using hammer. For web UI equivalents of the following procedures see the Red Hat Satellite Host Configuration Guide. To view errata that are available for all organizations, issue the following command: \$ hammer erratum listExample 8.1. Filtering Errata The hammer erratum list command provides numerous options for filtering and ordering the output list. For example, to find an erratum that contains a specific security fix, issue: \$ hammer erratum list --cve CVE-2014-0453 The following command displays applicable errata for the selected product that contains the specified bugs ordered so that the security errata are displayed on top: \$ hammer erratum list \ --product-id 7 \ --search "bug = 1213000 or bug = 1207972" \ --errata-restrict-applicable 1 \ --order "type" desc" For more information on syntax used in the --search option, refer to the Red Hat Satellite Host Configuration Guide. For more information on filtering options implemented in hammer, see the output of hammer erratum list --help. To view details of a specific erratum, issue the following command: \$ hammer erratum list command. You can identify errata also by name and repository name, see the output of hammer erratum info --help for details. To list errata available for a host, issue the following command: \$ hammer host errata apply \ --host \ --errata-ids ,... Example 8.2. Applying All Available Errata to a Host The following Bash script applies all errata available to a host (auth01.example.com): HOST="auth01.example.com" for i in \$(hammer --csv host errata apply --host \$HOST --errata-ids \$i doneExample 8.3. Applying a Security Advisory This example shows how to apply a security fix to hosts using hammer: Find the erratum that contains a fix for a selected issue (CVE-2015-3238): \$ hammer erratum list --cve CVE-2015-3238 -------- ID | ERRATA ID | TYPE | TITLE ------|---- f30e66 | RHSA-2015:1640 | security | Moderate: pam security update ------|--- Verify if the security erratum (RHSA-2015:1640) is applicable for your host (auth01.example.com): \$ hammer host errata list \ --host auth01.example.com \ --search "RHSA-2015:1640" You can use the following Bash script to apply a security erratum (for example RHSA-2015:1640) to all hosts where it is applicable: ORG="ACME" RHSA="RHSA-2015:1640" for i in \$(hammer --csv host list --organization \$ORG | grep -vi '^ID' | awk -F, {'print \$2'}) do hammer host errata apply --host \$i --errata-ids \$RHSA done To see if errata were applied successfully, find the corresponding task in the output of the following command: \$ hammer task progress --id To apply selected errata to a Host Collection, enter a command as follows: \$ hammer task progress --id To apply selected errata to a Host Collection, enter a command: \$ hammer task progress --id To apply selected errata to a Host Collection, enter a command: \$ hammer task progress --id To apply selected errata to a Host Collection erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name "my-collection" --organization erratum install \ --errata ",,..." \ --name \ --organization erratum install \ --errata ",,..." \ --name \ --organization erratum install \ --errata \ \$ORG This command is available in Red Hat Satellite 6.2.8 or later. A Docker container is a sandbox for isolating applications. The container image stores the configuration for the container image stores the Red Hat Satellite Host Configuration Guide. In Red Hat Satellite, you can deploy containers only on a compute resource of the Docker provider type. See the Satellite Host Configuration Guide for instructions on how to prepare a container host. To register this host as a compute resource, issue the following command: \$\\$\text{hammer compute-resource create --name} \--\] organization-ids ,... \ --location-ids ,... \ --location-ids ,... \ --compute resource ID in the output of hammer compute-resource list. Replace with the name of the synchronized repository that contains your docker images. This can be a custom Repository pointing to Docker Hub or your internal registry (see Section 2.3.4, "Creating a Custom Repository"), or the official Red Hat image repository. If you provision from a Content View, replace with the name of the Content View. See Section 3.2.3, "Adding Docker Images to a Content View" for details on adding images to a Content View. By starting a container related options, see the output of the hammer docker container --help command. FormatMulti-pageSingle-pageView full doc as PDFBack to top 0 ratings0% found this document useful, undefined