# **Provisioning VMWare using userdata via Satellite 6.3-6.6**

October 9, 2018[Will McDonald](https://www.redhat.com/en/authors/will-mcdonald)

SHARE

* [Back to all posts](https://www.redhat.com/en/blog)

Tags:[*Services*](https://www.redhat.com/en/blog?f%5B0%5D=taxonomy_blog_post_category_tid:70051)

#### ***This article was originally published on the Red Hat Customer Portal. The information may no longer be current. For up-to-date documentation on using the VMware vSphere Cloud-init and Userdata Templates for Provisioning,*** [***click here***](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.10/html/provisioning_guide/provisioning_virtual_machines_in_vmware_vsphere#Provisioning_Virtual_Machines_in_VMware_vSphere-Provisioning_with_cloudinit_and_userdata_templates)***.***

## **Background**

There are a number of different Red Hat technologies that can all be used to provision and configure VMware virtual machines.

* Cloudforms
* Ansible
* Satellite

Each has its own strengths, and scenarios where one may be favoured over another, framed by technical requirements, environmental constraints and local preferences.

During a recent engagement, we wanted to use Satellite to bulk-provision relatively large numbers of VMs to VMware Vcenter/ESX. Due to environmental constraints, we could not use Ansible, REX or SSH finish templates for post-provisioning configuration.

As a result we ended up with a solution using the upstream [Foreman Userdata plug-in](https://github.com/theforeman/foreman_userdata) to further customise provisioned hosts via userdata injection for VMware customisation specification and cloud-init to ‘phone-home’. This plugin is currently being merged into Foreman core, it will likely appear in future Satellite 6 releases. It’s not currently a supported Red Hat component so be aware of this if you plan to use it.

## **Prerequisites**

* Red Hat Satellite 6.3 (we used 6.3.3)
* VMware vCenter Server Appliance 6.7
* VMware ESXi 6.7

It is possible to reproduce this in a lab-like environment using nested virtualisation but this requires:

* 8 CPU cores
* 32GB
* 200GB of disk, thin provisioned (> 670 GB if thick provisioned)
* Vmware Workstation 14 or
* A nested ESX host with enough memory to run the VCSA in addition to VMs

*Note:* for those familiar with earlier releases of VCSA, the memory footprint for even a [Tiny VCSA](http://vcdx56.com/2018/04/vmware-vcenter-server-6-7-resource-requirements/) is substantial.

## **Setup Sequence of Events**

In order to set up the initial environment, there are a few moving parts to set up. At a high-level we need to:

* Deploy a Satellite
* Deploy VCSA
* Deploy ESXi
* Satellite initial configuration
  + Add manifest
  + Configure and sync repositories
  + Create a Lifecycle Environment
  + Create a Content View and add a Repository to it
  + Publish and promote the Content View
  + Create Activation Keys and add Subscription
  + Create a Subnet
  + Create a Compute Resource
  + Create a Compute Profile
  + Create a Hostgroup
  + Configure virt-who (optional)
  + Install the Foreman Userdata Plug-in and restart Satellite
  + Prepare provisioning templates (userdata and cloud-init)
  + Configure the Operating System to use the userdata and cloud-init templates
* Prepare a VM template
  + Create an empty VM
  + Upload an ISO
  + Install RHEL7
  + Tweak networking (if required)
  + Connect to Satellite temporarily
  + Configure a public key (optional)
  + Install open-vm-tools
  + Install Perl
  + Yum update
  + Create copy (optional but recommended)
  + Install cloud-init
  + Configure cloud-init
  + Create and run clean-up script
  + Bake template
* Create Image in Satellite
* Test provisioning

## **Provisioning Sequence of Events**

Once the environment is prepared, the provisioning workflow at a high-level is:

* User provisions VM(s) via the UI, API or hammer
* Satellite calls Vcenter to clone VM template
  + Satellite userdata provisioning template injects customisation specification identity information
  + Satellite cloud-init provisioning template instructs the VM to callback to Satellite when cloud-init runs post-provision
* Vcenter clones template to VM
* Vcenter applies customisation specification (VM identity including hostname, IP and DNS)
* VM builds, cloud-init is invoked and calls-back to Satellite on port 80 which then redirects to 443

*Note:* Even if registering the VM to a Capsule, the VM’s cloud-init phone-home always calls the Satellite, not its corresponding capsule. Make sure you take this into consideration in terms of firewall rules.

## **Network Configuration**

The hosts were configured with the following FQDNs, IPs, operating systems and services. Forward and reverse DNS were correctly configured in DNS for all hosts.

| **Hostname** | **IP** | **OS** | **Notes** |
| --- | --- | --- | --- |
| dns.example.com | 192.168.0.1 | RHEL7 | External DNS service |
| satellite.example.com | 192.168.0.3 | RHEL7 | Yum Content, Provisioning, Subscription Management |
| vcenter.example.com | 192.168.0.4 | Vmware Photon | vCenter |
| esxi1.example.com | 192.168.0.5 | ESXi | ESX Hypervisor |

## **Management Endpoints**

As this is just a demonstration lab, all SSL certificates are self-signed.

| **URL** | **Purpose** |
| --- | --- |
| [https://satellite.example.com](https://satellite.example.com/) | Satellite management UI |
| <https://vcenter.example.com:4580/> | VCSA management UI |
| <https://vcenter.example.com/> | Vcenter management UI |
| <https://esxi1.example.com/> | *Note:* *the ESX host's Self-signed cert must be trusted in order to upload ISOs*. |

## **VMware Installation**

Installing vCenter and an ESX host or two is not covered in-depth but isn't terribly onerous.

Vmware Workstation 14 (or a suitably sized ESX host capable of running the VCSA) is recommended in order to correctly install the Vmware 6.7 VCSA. In theory it should be able to be run nested on other platforms but the installer and the OVA have extensive customisation which would need to be worked through.

## **Satellite Preparation**

### **Satellite Installation**

This walkthrough assumes you have a basic Satellite deployed with an Organization and Location configured, a manifest installed but no other objects created (no products or repositories, content views, activation keys etc.).

See the following documents for further information on planning for, and installing, Red Hat Satellite:

* [Red Hat Satellite 6.3 - Architecture Guide](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/html/architecture_guide/)
* [Red Hat Satellite 6.3 - Installation Guide](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/html/installation_guide/)

Red Hat professional services will obviously *always* be more than happy to help here, too. :D

This demonstration used Satellite 6.3.3 but this should work for earlier releases too.

## **Satellite Initial Preparation**

### **Hammer Setup**

Hammer has already been [configured with sensible defaults](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/html/hammer_cli_guide/chap-cli_guide-introduction_to_hammer#sect-CLI_Guide-Setting_a_Default_Organization) for organization and location.

### **Repositores**

Configure upstream Red Hat Repositories, in the Satellite UI under Content and Red Hat Repositories, enable the following

* RPMS / Red Hat Enterprise Linux Server / Red Hat Enterprise Linux 7 Server (RPMs) / Red Hat Enterprise Linux 7 Server RPMs x86\_64 7Server

Sync the upstream content in the UI, under Content and Sync Status, Expand All, Select All, and Synchronize Now.

### **Lifecycle Environments**

Create a lifecycle environment (LE):

[root@satellite ~]# hammer lifecycle-environment create --description le-eng-rhel7-std-server --prior Library --name le-eng-rhel7-std-server

### **Content View**

Create a content view (CV):

[root@satellite ~]# hammer content-view create --description cv-rhel7-std-server --name cv-rhel7-std-server

### **List Repository IDs**

List the repository IDs of the synchronised content for the subsequent step:

[root@satellite ~]# hammer repository list --order id | awk -F '|' '{ print $1, $2 }'

### **Add Repositories to Content View**

Add the required repositories from the previous step into the Content View:

[root@satellite ~]# hammer content-view update --repository-ids 1,2 --name "cv-rhel7-std-server"

### **Publish Content to Library**

Publish a new version of the Content View into the initial Library:

[root@satellite ~]# hammer content-view publish --name "cv-rhel7-std-server" --async

### **Promote Content into Content View**

Now promote the newly published Content View from the Library into the first Lifecycle Environment:

[root@satellite ~]# hammer content-view version promote --content-view "cv-rhel7-std-server" --to-lifecycle-environment "le-eng-rhel7-std-server" --async

### **Create Activation Key**

Create an Activation Key (AK) to be used subsequently to register newly provisioned hosts:

[root@satellite ~]# hammer activation-key create --content-view "cv-rhel7-std-server" --lifecycle-environment "le-eng-rhel7-std-server" --name "ak-eng-rhel7-std-server"

### **List Available Subscriptions**

List the available subscriptions (those loaded into the Satellite through its manifest):

[root@satellite ~]# hammer subscription list | awk -F'|' '{ print $1, $3, $9, $10 }'

--- ------------------------------------ ---------- ---------

ID NAME QUANTITY CONSUMED

--- ------------------------------------ ---------- ---------

1 Red Hat Satellite (Self-Supported) 1 0

2 Red Hat Enterprise Linux 1 0

--- ------------------------------------ ---------- ---------

### **Add Subscriptions to the Activation Key**

Add the desired subscriptions to the Activation Key:

[root@satellite ~]# hammer activation-key add-subscription --name "ak-eng-rhel7-std-server" --subscription-id 2

### **Configure the Domain**

A domain will already have been created during Satellite installation. Add this domain to the default location and organization through the UI.

### **Create a subnet**

Create a new subnet for the network to be provisioned to:

[root@satellite ~]# hammer subnet create --name "sn-default-servers" --network "192.168.0.0" --mask "255.255.255.0" --gateway "192.168.0.1" --dns-primary "192.168.0.8" --domains example.com --locations loc-example --organizations org-example

### **Create a Compute Resource**

Either username@domain and domain\username both work from a compute resource creation perspective.

[root@satellite ~]# hammer compute-resource create --caching-enabled 1 --datacenter "dc-example" --name "cr-vcenter" --password "password" --provider "Vmware" --server "vcenter.example.com" --user 'vsphere.local\administrator' --locations "loc-example" --organizations "org-example"



[root@satellite ~]# hammer compute-resource create --caching-enabled 1 --datacenter "dc-example" --name "cr-vcenter" --password "password" --provider "Vmware" --server "vcenter.example.com" --user 'administrator@vsphere.local' --locations "loc-example" --organizations "org-example"

### **Create a Compute profile**

*Note:* This can’t currently be done via hammer, see the following articles for more information:

* [Satellite 6: How to create compute-profile through hammer cli?](https://access.redhat.com/solutions/2868241)
* [Bug 1278917 - RFE Commands for read operations around compute profiles and attributes](https://bugzilla.redhat.com/show_bug.cgi?id=1278917)
* [Commands for read operations around compute profiles and attributes](https://projects.theforeman.org/issues/6344)

In the UI select Infrastructure > Compute profile > Create Compute Profile

Populate the indicated fields following values, select Compute Resource (cr-vcenter):

| **Field** | **Value** |
| --- | --- |
| Name | vmware-small |
| Cluster | esxi1.example.com |
| Resource pool | auto populated ‘Resources’ |
| Folder | vm |
| Guest OS | Red Hat Enterprise Linux 7 (64-bit) |
| Image | img-rhel7-std-server |
| NIC type | VMXNET3 |
| Network | VM Network |
| Storage | See below... |
| Data store | datastore1 |
| Thin provision | uncheck |

### **Create a Hostgroup**

*Note:* This can be accomplished in hammer but for a one-off it's quicker to create in the UI to save having to enumerate each value to illustrate the process.

A hostgroup was created in the UI with the following values:

[root@satellite ~]# hammer hostgroup info --id 1

Id: 1

Name: hg-rhel7-std-server

Title: hg-rhel7-std-server

Operating System: RedHat 7.5

Subnet: sn-default-servers

Domain: example.com

Architecture: x86\_64

Puppet CA Proxy Id:

Puppet Master Proxy Id:

ComputeProfile: vmware-small

Puppetclasses:

Parameters:

kt\_activation\_keys => ak-eng-rhel7-std-server

Locations:

loc-example

Organisations:

org-example

Parent Id:

OpenSCAP Proxy:

Content View:

ID: 2

Name: cv-rhel7-std-server

Lifecycle Environment:

ID: 2

Name: le-eng-rhel7-std-server

Content Source:

ID: 1

Name: satellite.example.com

Kickstart Repository:

ID:

Name:

### **virt-who Configuration**

Given the target compute in this walkthrough is Vmware, it’s likely you will need to configure virt-who for VDC subscription.

In this example, we are installing virt-who on the Satellite and this approach is generally fine for small to medium virtual estates. For very large estates, please refer to Red Hat documentation, support or consulting.

#### **Install virt-who**

Install the virt-who package and its dependencies

[root@satellite ~]# yum install -y virt-who

#### **Create virt-who configuration**

Create a virt-who configuration for each vCenter to be targeted

[root@satellite ~]# hammer virt-who-config create --name vcenter.example.com --organization org-example --interval 60 --filtering-mode none --hypervisor-id hostname --hypervisor-type esx --hypervisor-server vcenter.example.com --hypervisor-username 'administrator@vsphere.local' --hypervisor-password 'password' --satellite-url satellite.example.com

#### **Deploy the virt-who configuration**

Deploy the virt-who configuration

[root@satellite ~]# hammer virt-who-config deploy --id 1

#### **Debugging virt-who**

In the event of problems registering the vCenter, virt-who can be debugged by stopping the service and running in debug mode in the foreground.

[root@satellite ~]# systemctl stop virt-who.service

[root@satellite ~]# virt-who -d -o

### **Foreman Userdata Plugin Configuration**

Next we install the upstream Foreman Userdata plugin appropriate for the target release.

*Note:* this plugin is not currently part of core Foreman, nor is it shipped by Red Hat. As a result this is NOT currently a supported solution. The plugin is on the roadmap for future inclusion.

#### **Check Foreman Version**

Check the version of Foreman on which the installed Satellite release is based:

[root@satellite ~]# rpm -q foreman

foreman-1.15.6.48-1.el7sat.noarch

*Note:* foreman-\*1.15\*.6.48-1. The *1.15* value is important for the subsequent step.

#### **Enabled Foreman Plugin Repository**

Create a Yum repository configuration stub. For Satellite 6.3:

[root@satellite ~]# cat > /etc/yum.repos.d/foreman-plugins.repo <<EOF

[foreman-plugins]

name=Foreman plugins

baseurl=http://yum.theforeman.org/plugins/1.15/el7/x86\_64/

enabled=1

gpgcheck=0

EOF

For Satellite 6.4:

[root@satellite ~]# cat > /etc/yum.repos.d/foreman-plugins.repo <<EOF

[foreman-plugins]

name=Foreman plugins

baseurl=http://yum.theforeman.org/plugins/1.18/el7/x86\_64/

enabled=1

gpgcheck=0

EOF

For Satellite 6.5:

[root@satellite ~]# cat > /etc/yum.repos.d/foreman-plugins.repo <<EOF

[foreman-plugins]

name=Foreman plugins

baseurl=http://yum.theforeman.org/plugins/1.20/el7/x86\_64/

enabled=1

gpgcheck=0

EOF

For Satellite 6.6:

[root@satellite ~]# cat > /etc/yum.repos.d/foreman-plugins.repo <<EOF

[foreman-plugins]

name=Foreman plugins

baseurl=http://yum.theforeman.org/plugins/1.22/el7/x86\_64/

enabled=1

gpgcheck=0

EOF

The plugin code has been merged into core functionality in Foreman 1.23, therefore future Satellite updates will contain the feature out of box and this step can be skipped completely.

#### **Install the Foreman Userdata plugin**

Now we can 'yum install' the foreman\_userdata plugin from the upstream repository:

[root@satellite ~]# yum -y install tfm-rubygem-foreman\_userdata

#### **Restart Satellite**

Restart the Satellite services to pick up the newly installed plugin.

[root@satellite ~]# katello-service restart

### **Configure Provisioning Templates**

Now we need to create and load some additional provisioning templates to be used by the userdata injection and cloud-init client.

#### **Create provisioning template files**

##### **vmware-cloud-init-template**

Create a cloud-init template file:

[root@satellite ~]# cat > ~/vmware-cloud-init-template.erb <<EOF

#cloud-config

hostname: <%= @host.name %>

fqdn: <%= @host %>

manage\_etc\_hosts: true

users: {}

runcmd:

- touch ~/cloud-init

phone\_home:

url: <%= foreman\_url('built') %>

post: []

tries: 10

EOF

*Note*: the ‘#cloud-config’ line is required in the cloud-init template, similar to a magic number of shebang.  
*Note:* the runcmd illustrated here is merely a simple test to ensure cloud-init is functioning. More complex logic can be used in the template for registration, subscription management or other bespoke callback or finish tasks.

##### **vmware-userdata-template**

Create a userdata template file:

[root@satellite ~]# cat > ~/vmware-userdata-template.erb <<EOF

# Template for VMWare customization via open-vm-tools

identity:

LinuxPrep:

domain: <%= @host.domain %>

hostName: <%= @host.shortname %>

hwClockUTC: true

timeZone: <%= @host.params['time-zone'] || 'UTC' %>

globalIPSettings:

dnsSuffixList: [<%= @host.domain %>]

<%- @host.interfaces.each do |interface| -%>

<%- next unless interface.subnet -%>

dnsServerList: [<%= interface.subnet.dns\_primary %>, <%= interface.subnet.dns\_secondary %>]

<%- end -%>

nicSettingMap:

<%- @host.interfaces.each do |interface| -%>

<%- next unless interface.subnet -%>

- adapter:

dnsDomain: <%= interface.domain %>

dnsServerList: [<%= interface.subnet.dns\_primary %>, <%= interface.subnet.dns\_secondary %>]

gateway: [<%= interface.subnet.gateway %>]

ip: <%= interface.ip %>

subnetMask: <%= interface.subnet.mask %>

<%- end -%>

EOF

#### **Install provisioning templates**

Now install the newly created templates:

[root@satellite ~]# hammer template create --name vmware-cloud-init --file ~/vmware-cloud-init-template.erb --locations loc-example --organizations org-example --operatingsystem-ids 1 --type cloud-init

[root@satellite ~]# hammer template create --name vmware-userdata --file ~/vmware-userdata-template.erb --locations loc-example --organizations org-example --operatingsystem-ids 1 --type user\_data

#### **Configure Templates in the Operating System**

Now the templates are created and loaded into the Satellite, we can update the Operating System definition to use the desired templates during provisioning requests.

In the UI, switch the Red Hat 7.5 Operating System to use these templates:

Go to Hosts > Operating systems > Red Hat 7.5 > Templates  
- User data template: vmware-userdata  
- Cloud-init: vmware-cloud-init

## **Template Preparation**

### **Create Empty Virtual Machine**

In vCenter, create an empty RHEL7 VM container with the following properties:  
- 1 vCPU  
- 2GB vMem  
- 16GB VMDK

### **Upload a Red Hat Enterprise Linux ISO**

Download rhel-server-7.5-x86\_64-dvd.iso from the [Red Hat Customer Portal | <https://access.redhat.com/downloads> ], then upload into vCenter to a datastore ‘isos’ folder

*Note:* log in to the ESXi host directly in a browser to accept its self-signed certificate or ISO upload won’t work.

### **Install Red Hat Enterprise Linux**

Install RHEL7 minimal from the ISO, consult Red Hat Enterprise Linux 7 documentation for more details.

### **Virtual Machine Network Configuration**

The virtual machine will require some temporary network identity settings in order to prepare the template. Tweak interface connection name and DNS as appropriate for the environment:

[root@localhost ~]# nmcli con modify 'System ens192' connection.id ens192

[root@localhost ~]# nmcli con mod ens192 ipv4.method auto

[root@localhost ~]# nmcli con edit ens192

nmcli connection edit ens192

nmcli> remove ipv4.dns

nmcli> set ipv4.ignore-auto-dns yes

nmcli> set ipv4.dns 192.168.0.2

nmcli> save

nmcli> quit

nmcli device reapply ens192

[root@localhost ~]# hostnamectl set-hostname "localhost.localdomain"

### **Subscribe to Satellite**

Now we need to temporarily subscribe to the Satellite for some template prerequisite packages and a few other bits and pieces:

[root@localhost ~]# rpm -ivh http://satellite.example.com/pub/katello-ca-consumer-latest.noarch.rpm

[root@localhost ~]# subscription-manager register --org=org-example --activationkey=ak-eng-rhel7-std-server

### **Drop in SSH Keys**

This is an opportune moment to drop in any SSH public keys if desired:

[root@localhost ~]# mkdir ~/.ssh

[root@localhost ~]# chmod 700 ~/.ssh

[root@localhost ~]# cat > ~/.ssh/authorized\_keys <<EOF

ssh-rsa AAAABlongkeyfingerprint will@example.com

EOF

[root@localhost ~]# chmod 600 ~/.ssh/authorized\_keys

*Note:* keys can also be injected into the template during provisioning time via cloud-init.

### **Install open-vm-tools**

Install open-vm-tools to ensure the userdata customisation specification is properly handled:

[root@localhost ~]# yum -y install open-vm-tools.x86\_64

### **Install Perl**

Perl is also required for open-vm-tools to correctly parse and handle the customisation specification injection at provisioning time:

[root@localhost ~]# yum -y install perl

### **Yum Update Virtual Machine**

While the VM template is temporarily registered to the Satellite, it will reduce future updates if we bring it up to the latest patch level:

[root@localhost ~]# yum -y update

Note: At this stage, it can be wise to power off and create a copy of the template without cloud-init. cloud-init can be quite invasive, if the template preparation requires multiple iterations this is a good place to restart from.

### **Install cloud-init**

Power the virtual machine back on and install cloud-init:

[root@localhost ~]# yum -y install cloud-init

### **Configure cloud-init to Skip Networking**

Drop in a configuration stub to instruct cloud-init to skip network configuration:

[root@localhost ~]# cat << EOF > /etc/cloud/cloud.cfg.d/01\_network.cfg

network:

config: disabled

EOF

*Note:* Satellite will do this via the Vmware customisation specification).

### **Configure cloud-init to Callback to Satellite Post-build**

Drop in a configuration stub to instruct cloud-init to call back to Satellite post-provision. This informs Satellite of the build's completion, and allows additional post-provisioning behaviour to be injected via cloud-init configuration (in the form of the cloud-init provisioning template).

[root@localhost ~]# cat << EOF > /etc/cloud/cloud.cfg.d/10\_foreman.cfg

datasource\_list: [NoCloud]

datasource:

NoCloud:

seedfrom: http://satellite.example.com/userdata/

EOF

### **Replace Default cloud-init Configuration**

The default cloud-init configuration may be considered quite invasive by the standards of some environments (YMMV). Back-up the default cloud-init configuration and replace with a more minimal set of configuration that does just what we require:

[root@localhost ~]# cp /etc/cloud/cloud.cfg ~/cloud.cfg.`date -I`

[root@localhost ~]# cat << EOF > /etc/cloud/cloud.cfg

cloud\_init\_modules:

- bootcmd

cloud\_config\_modules:

- runcmd

cloud\_final\_modules:

- scripts-per-once

- scripts-per-boot

- scripts-per-instance

- scripts-user

- phone-home

system\_info:

distro: rhel

paths:

cloud\_dir: /var/lib/cloud

templates\_dir: /etc/cloud/templates

ssh\_svcname: sshd

# vim:syntax=yaml

EOF

### **Unregister Virtual Machine from Satellite**

Now the template is fully prepared bar the *clean/sysprep* phase, we can unregister it from the Satellite:

[root@localhost ~]# subscription-manager unregister

Unregistering from: satellite.example.com:443/rhsm

System has been unregistered.

[root@localhost ~]# subscription-manager clean

All local data removed

### **Create Clean Script**

Now we drop in a simple *clean/sysprep* script to remove the virtual machine identity prior to converting back to a template:

[root@localhost ~]# cat > ~/clean.sh <<EOF

#!/bin/bash

# stop logging services

/usr/bin/systemctl stop rsyslog

/usr/bin/systemctl stop auditd

# remove old kernels

# /bin/package-cleanup -oldkernels -count=1

#clean yum cache

/usr/bin/yum clean all

#force logrotate to shrink logspace and remove old logs as well as truncate logs

/usr/sbin/logrotate -f /etc/logrotate.conf

/bin/rm -f /var/log/\*-???????? /var/log/\*.gz

/bin/rm -f /var/log/dmesg.old

/bin/rm -rf /var/log/anaconda

/bin/cat /dev/null > /var/log/audit/audit.log

/bin/cat /dev/null > /var/log/wtmp

/bin/cat /dev/null > /var/log/lastlog

/bin/cat /dev/null > /var/log/grubby

#remove udev hardware rules

/bin/rm -f /etc/udev/rules.d/70\*

#remove uuid from ifcfg scripts

/bin/cat > /etc/sysconfig/network-scripts/ifcfg-ens192 <<EOM

DEVICE=ens192

ONBOOT=yes

EOM

#remove SSH host keys

/bin/rm -f /etc/ssh/\*key\*

#remove root users shell history

/bin/rm -f ~root/.bash\_history

unset HISTFILE

#remove root users SSH history

/bin/rm -rf ~root/.ssh/known\_hosts

EOF

### **Run Clean Script**

Clean the system:

[root@localhost ~]# sh ~/clean.sh

### **Power Off Virtual Machine**

Power off the VM:

[root@localhost ~]# systemctl poweroff

### **Create Virtual Machine Template**

Now convert the VM to a template in vCenter ‘template-rhel7-cloudinit’

## **Finalising Satellite Preparation**

Now we have our Satellite and virtual machine template prepared, there's just one final step before moving to testing the provisioning flow.

### **List Operating System, Architecture and Compute Resource**

List the Operating System, Architecture and Compute Resource configuration in Satellite to identify the corresponding object IDs:

[root@satellite ~]# hammer os list; hammer architecture list; hammer compute-resource list

---|------------|--------------|-------

ID | TITLE | RELEASE NAME | FAMILY

---|------------|--------------|-------

1 | RedHat 7.5 | | Redhat

---|------------|--------------|-------

---|-------

ID | NAME

---|-------

1 | x86\_64

2 | i386

---|-------

---|------------|---------

ID | NAME | PROVIDER

---|------------|---------

1 | cr-vcenter | VMware

---|------------|---------

### **Create Image in Satellite**

Now we create an Image in Satellite, linking the template in vCenter with the required resources in Satellite:

[root@satellite ~]# hammer compute-resource image create --operatingsystem-id 1 --architecture-id 1 --compute-resource-id 1 --user-data true --uuid template-rhel7-cloudinit --username root --name img-rhel7-std-server

## **Test Guest Provisioning**

We should now be in a position to test provisioning of guest VMs via the Satellite UI or hammer.

For example:

[root@satellite ~]# hammer host create --name test-host --organization org-example --location loc-example --hostgroup hg-rhel7-std-server --compute-resource cr-vcenter --provision-method image --image img-rhel7-std-server --enabled true --managed true --compute-profile vmware-small --operatingsystem-id 1 --compute-attributes="start=1" --interface "managed=true,compute\_type=VirtualVmxnet3,type=interface,domain\_id=1,identifier=ens192,ip=192.168.0.50,subnet\_id=1,primary=true,compute\_network='VM Network'"

## **Next Steps**

Once provisioning works, guests are being cloned, allotted the correct hostname, domain and IP, and the ~/cloud-init file is being created, the cloud-init provisioning template can be extended to perform any other post/finish tasks as required. For example:

* Installing the katello-consumer-ca from the appropriate content host
* Registering with a specific activation key
* Installing the katello-agent

Once host provisioning works for a single host, it’s relatively trivial to scale this up to 10s or 100s of hosts.

Even in a small, nested environment, the end-to-end process takes just a couple of minutes per-VM at most. However, care should be taken around concurrent host registration so it may be wise to batch bulk provisioning into groups of 10s or 20s of VMs and wait for completion of each batch.

## **References**

### **Red Hat Documentation**

* [https://access.redhat.com/documentation/en-us/red\_hat\_satellite/6.2/htm…](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.2/html-single/hammer_cli_guide/index#sect-CLI_Guide-Authentication)
* [https://access.redhat.com/documentation/en-us/red\_hat\_satellite/6.3/htm…](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/html/installation_guide/installing_satellite_server#performing_initial_configuration_sat_server_parent)
* <https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/>
* [https://access.redhat.com/documentation/en-us/red\_hat\_satellite/6.3/htm…](https://access.redhat.com/documentation/en-us/red_hat_satellite/6.3/html/virtual_instances_guide/)

### **Useful Links**

* [https://translate.google.co.uk/translate?hl=en&sl=fr&u=https://www.mche…](https://translate.google.co.uk/translate?hl=en&sl=fr&u=https://www.mchelgham.com/installer-vcsa-6-7-dans-vmware-workstation-fusion/&prev=search)
* [https://www.virten.net/2017/02/homelab-downsizing-vcenter-server-applia…](https://www.virten.net/2017/02/homelab-downsizing-vcenter-server-appliance-6-5/)
* [https://www.starwindsoftware.com/blog/vmware-vcenter-server-appliance-v…](https://www.starwindsoftware.com/blog/vmware-vcenter-server-appliance-vcsa-and-after-install-tricks)
* [http://everything-virtual.com/2016/05/06/creating-a-centos-7-2-vmware-g…](http://everything-virtual.com/2016/05/06/creating-a-centos-7-2-vmware-gold-template/)

### **ABOUT THE AUTHOR**

****

### **[Will McDonald](https://www.redhat.com/en/authors/will-mcdonald)**

[Services Delivery Manager](https://www.redhat.com/en/authors/will-mcdonald)

After a stint as a Technical Account Manager with Red Hat in 2008-2009, Will McDonald rejoined in 2015 as a Consulting Architect, moving to delivery management in 2019. McDonald has worked in support, systems administration, systems engineering, architecture and consulting over a 20-plus-year IT career.

[**Read full bio**](https://www.redhat.com/en/authors/will-mcdonald)

Enter