

# Red Hat Summit 2018

## Hands on Red Hat Management Lab

Red Hat Team supporting you in this lab:

**Will Nix, Amaya Gil Pippino, Victor Estival López, Chris Henderson, Camry Fedei**

### README.1ST

The goal of these lab exercises are to introduce you to a variety of Red Hat products that can help you with managing and automating infrastructure resources. You will be able to demonstrate the power and flexibility of Red Hat management using either one or a combination of Red Hat products, such as Red Hat Satellite, Ansible Tower by Red Hat, Red Hat Insights, and Red Hat CloudForms.

Upon completion of this lab material, you should have a better understanding of how to perform management tasks in your environments, how to more securely maintain hosts, and how to achieve greater efficiencies with automating tasks and managing systems within your infrastructure.

Most of the labs are exclusive of any other lab - i.e. If you want to only perform the CloudForms lab, then you can skip to the CloudForms lab. The timing of the lab is structured so an experienced system administrator should be able to complete the entire lab in allotted time following the documentation as written. If you would like to focus on an individual lab or specific portfolio product, please use the LAB INDEX to select the Lab you would like to begin on. **Lab 0 should be viewed before attempting any other labs. If there is a problem with a step in your lab please raise your hand and contact one of the lab instructors.**

This lab is geared towards systems administrators, cloud administrators and operators, architects, and others working on infrastructure operations interested in learning how to automate management across a heterogeneous infrastructure. The prerequisite for this lab include basic Linux skills gained from Red Hat Certified System Administrator (RHCSA) or equivalent system administration skills. Knowledge of virtualization and basic Linux scripting would also be helpful, but not required.

There are lab assistants and presenters roaming the room to help you with issues or answer questions - just raise your hand!

# LAB INDEX

## [LAB INDEX](#)

### [Lab 0: Setup & Getting Started](#)

### [Lab 1: Red Hat Satellite for Content Lifecycle Management](#)

[Goal of Lab](#)

[Notes about Satellite Environment and Configurations](#)

[Log into the environment](#)

[Verify Synced Content:](#)

[Create Lifecycle Environment](#)

[Create Content View:](#)

### [Lab 2: Satellite for Content Host Management](#)

[Create Activation Key:](#)

[Register Content Hosts:](#)

[Update Content Hosts:](#)

### [Lab 3: Proactive Security and Automated Risk Management with Red Hat Insights](#)

[Goal of Lab](#)

[Introduction](#)

[Fixing the payload injection security issue in your system using Red Hat Insights from the Satellite UI](#)

[Cleaning your environment](#)

[Adding your Insights hosts to Satellite](#)

[Installing the Insights client](#)

[Fixing the payload injection security issue](#)

[Automatically fixing the payload injection security issue via Ansible Playbook](#)

[Bonus! Automatically fix all the issues on systems ic8.example.com and ic9.example.com](#)

### [Lab 4: Automatic Remediation with Red Hat Insights and Ansible Tower](#)

[Goal of Lab](#)

[Introduction](#)

[Setting up an Insights Scan Project](#)

[Create Insights Credentials](#)

[Create insights credentials](#)

[Creating an inventory](#)

[Creating an Scan Project](#)

[Creating a Job Template](#)

[Back to Table of Contents](#)

[Viewing Insights data into Tower](#)

[BONUS!](#)

[Automatically remediate Insights Inventory](#)

[Creating a Remediation Project](#)

[Creating a Remediation Job Template](#)

[Lab 5: Introduction to Ansible Tower](#)

[Why Ansible Tower?](#)

[Architecture and Installation](#)

[About this Lab](#)

[Ansible Tower Concepts](#)

[Dashboard](#)

[Projects](#)

[Inventories](#)

[Credentials](#)

[Templates](#)

[Jobs](#)

[Create an Inventory](#)

[Machine Credentials](#)

[Run Ad Hoc Commands](#)

[Using Variables](#)

[Add a new Project](#)

[Setup Git Repository](#)

[Create a Job Template and Run a Job](#)

[Ansible Tower Role Based Access Control](#)

[Ansible Tower Users](#)

[Ansible Tower Teams](#)

[Granting Permissions](#)

[Test Permissions](#)

[Lab 6: Build a Service Catalog with CloudForms](#)

[Value provided by a Service Catalog](#)

[Service Basics](#)

[Virtual Machine Provisioning example](#)

[Build a VM Provisioning Service Dialog](#)

[Build a VM Provisioning Service Catalog](#)

[Build a Virtual Machine Service Catalog Item](#)

[Order the Simple Virtual Machine Service Catalog Item](#)

[Verify the order](#)

[Lab 7: CloudForms Ansible Example](#)

[Back to Table of Contents](#)

[Introduction to Ansible](#)  
[Make sure embedded Ansible role is enabled and running](#)  
[Add a Git repository of Ansible Playbooks](#)  
[Store Virtual Machine Credentials](#)  
[Create an Ansible Service Catalog](#)  
[Create a Service Catalog Item](#)  
[Test the Service Catalog Item](#)

## Lab 0: Setup & Getting Started

Logging into all the Red Hat Products:

Let's log into the Red Hat Products that you will use in this lab so they are ready to use.

In this lab application based self-signed SSL certs are going to be used, please note that they are being used and should be accepted in order to complete the lab exercises.

Let's log into the Red Hat Products that you will use in this lab so they are ready to use.

1. Open Firefox to The Red Hat Summit [GUID Grabber](#) application in order to obtain your lab GUID.
2. For Lab Code select **L1044 - Red Hat Integrated Management Technologies Lab**
3. Enter the activation key provided by the lab instructor then click Next.
  - a. **for this lab the key is "manage"**
4. In your Red Hat Summit Lab Information webpage, take note of your assigned GUID. You will use this GUID to access your lab's systems. Click on the link at the bottom of this page to access your lab environment.

1. From the lab environment information page, copy the hostname of the Workstation system (it should be workstation-GUID.rhpds.opentlc.com where GUID matches your environment's guid).
2. Open a terminal window on your desktop environment and make sure you can SSH into the workstation host as you see below:
3. `[lab-user@localhost ~]$ ssh workstation-GUID.rhpds.opentlc.com`

If you need to troubleshoot or power on/off/reboot a system, you can use the environment's power control and consoles by clicking the link on your GUID page. The password for any consoles will be with username 'root' and password "**r3dh4t1!**". From this page, you will be able to access all of the Red Hat Products that you will use in this lab. Press the start button at the top right to turn on all the Red Hat Product VMs. Then, click on https for all the Red Hat Products to access the UI. For applications, You may also log into the UI of all the Red Hat Products with 'admin' as the Username and "**r3dh4t1!**" (without the quotes) as the Password.

The following labs take place within the fictional EXAMPLE.COM company.

# Lab 1: Red Hat Satellite for Content Lifecycle Management

## Goal of Lab

In this lab, you will be provided the necessary steps and background information on how to use Red Hat Satellite for content management with managed hosts. Some of the values may be pre-populated for you, and the pre-populated values may be required for subsequent labs, so ***do not remove the pre-populated values.***

## Notes about Satellite Environment and Configurations

1. Each checkpoint will have a pre-built/configured object (Activation Key, Content View, Lifecycle Environment, etc.).
  - a. These pre-built objects can be used as examples as you create your own.
  - b. You'll need to name your objects differently, however configurations will need to match, so we have these here for your reference.
2. **Lab Exercise Setup:** Based on the structure/mobility of the lab, it's possible to run into some issues with certain services therein. If you run into anything, you can run the following to fully restart the satellite services:

SSH from your jumpbox as outlined in Lab 0 to root@sat.example.com

```
[lab-user@workstation $] sudo -i
```

```
[root@workstation #] ssh sat.example.com
```

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

## Log into the environment

1. **Access to the Satellite server from your browser:** Although several browsers are supported, we recommend the use of Firefox. Remember, as previously mentioned, if you see an SSL warning, accept this for the lab as its a self-signed certificate from the application.

Point your web browser to `https://sat-<GUID>.rhpd.opentlc.com` or click on your Lab GUID page that is open in your browser and open the link for your Sat

Where <GUID> is your unique identifier as described in Lab 0.

When logging into these systems you could potentially receive the following due to a self signed certificate:

Click “Advanced” to open the advanced menu, and then “Proceed to ....” at the bottom of the dialogue. The warning is due to a self signed certificate authority potentially not being available on your local workstation and if you’re using the Chrome browser.

Now that we’ve navigated to the login, login as user “admin” with the password “**r3dh4t1!**” as mentioned previously in this guide.

Let’s begin.



## Verify Synced Content:

In this lab we will have a few items populated for us already, and we will also have to create some items. The scenario is that you're the proud new beneficiary of another team's Satellite server. Let's look at some of the ways EXAMPLE.COM is using Satellite for content management. Content management

### 1. Navigate to, and confirm content Sync Status

- a. **\*\*\* NOTE:** To save time, in your lab all content is up to date and ready to Continue through to Step 2. **\*\*\***
  - **This is for information purposes only. No action is required from you for this step.**
- b. To Navigate to the Sync Status page, select **"Content"** from your toolbar, and select **"Sync Status"** from the dropdown.
- c. In the top right corner, you can click **"Expand All"**, or you can simply expand each item in the tree manually to confirm which of the repositories is available and recently synced.

### Background:

> In a live environment we want to ensure the content is available, and up to date. With that, we can ensure that when creating our Content View we're publishing the latest available content so we can promote it through our Lifecycles to be used by our Content Hosts.

> In this screenshot, you can see the content was not up to date, so a sync would be needed in order to get it up to date. To sync you would simply select the

desired repository, and select “Synchronize” in the bottom right corner. **(There is no need to perform this command. It is for informational purposes only and will only hinder your progress in the lab.)**

## Create Lifecycle Environment

### 1. Navigate to Lifecycle Environments

- a. To Navigate to the Lifecycle Environment page, select “**Content**” from your toolbar, and select “**Lifecycle Environments**” from the dropdown.
- b. Since this is the first time we’re logging into our virtual environment, the page will take a moment to populate with the ‘Summit\_DEV’ and ‘Summit\_PROD’ Lifecycle Environments in your Environment Path. Once loaded the “**Add New Environment**” button will appear for you to create the new Lifecycle

Environment in your Environment Path.

### 2. Create New Environment

- a. Click the “**Add New Environment**” button discovered in the previous step, and continue with naming the Lifecycle Environment whatever you’d like.
  - i. **Note:** This is NOT the blue “Create Environment Path” button in the top right corner. In the previous step, this may take a moment to load.
- b. Click “**Save**” to save the new environment.

**Background:**

The most common environment use-case would be stage-releasing content through multiple environments in a lifecycle, before finally pushing out into production. The simplest illustration would be to first promote content to a Development environment. Once verified there, that same content can then be promoted out to a Live or QA Testing environment, and then finally out to Production.

Creating a Lifecycle Environment allows for the creation of this simple and efficient stage-releasing structure through which your Content Hosts can receive this content in each staging cycle.

NOTE - Even without creating a new Lifecycle Environment, each Satellite server is pre-configured with a Library Environment which is effectively ALL synced content available on the Satellite. This environment is unchangeable, and you cannot add new environments to this path. That said, Content Views will promote to this Library Environment by default, and Content Hosts can be configured to pull directly from Library if desired.

## Create Content View:

- 1. Navigate to Content Views**

- a. To Navigate to the Content Views page, select “Content” from your toolbar, and select “Content Views” from the dropdown.**

## 2. Create the new Content View:

- a. Click the **“Create New View”** button discovered in the previous step, and continue with naming the Content View whatever you’d like.
- b. Click **“Save”** to save the new view.

## 3. Add Content to New View

- a. From the Content Views page, select your newly created Content View from the list.
  - **Note:** If continuing from the last step, you should be directed to this page automatically.
- b. Select **“Yum Content”** from the Content View toolbar, and then select **“Repositories”** from the dropdown list. (Dropdown located under “Yum Content” option)
- c. Select **“Add”** from the new section under Repository Selection.
- d. Click the checkbox next to each of the following repositories and then click **“Add Repositories”** in the top right corner of the Repository Selection section.
  - Red Hat CloudForms Management Engine 5.8 RPMs x86\_64
  - Red Hat Enterprise Linux 7 Server - Extras RPMs x86\_64
  - Red Hat Enterprise Linux 7 Server Kickstart x86\_64 7.4

- Red Hat Enterprise Linux 7 Server - RH Common RPMs x86\_64 7Server
- Red Hat Enterprise Linux 7 Server RPMs x86\_64 7Server
- Red Hat Enterprise Linux 7 Server - Supplementary RPMs x86\_64 7Server
- Red Hat Satellite Tools 6.3 for RHEL 7 Server RPMs x86\_64
- Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server x86\_64 7Server

#### **4. Confirm required repositories are present in Content View**

- a. Once added, you can confirm (and/or remove) the added repositories by clicking “List/Remove” in the Repository Selection section.

## 5. Publish New Version of Content View

- a. From the Content Views page, select your newly created Content View from the list (Content -> Content Views -> Whatever you named your Content View)
- b. Select “**Versions**” from the Content View toolbar.
- c. Click the blue “**Publish New Version**” button in the top right corner of the page to publish this new Content View.
- d. Feel free to add a Description, and then click “**Save**” to lock the content in at this current state
- e. \*\*\* **NOTE:** Publishing this Content View will take ~5 minutes to complete. Feel free to read ahead over the next steps, and/or over the Background section while waiting. \*\*\*

## 6. Promote Content View through Lifecycle Environment

- a. From the Content Views page, select your newly created Content View from the list.
- b. Select “**Versions**” from the Content View toolbar.
- c. Click “Promote” under the Actions column of the desired Version
  - Note: The Starred Lifecycle Environment is representative of the “Next Environment” based on your designed Lifecycle Path, which creates an easy way to identify where you are currently posted in your Lifecycle, and what Environment is next in line.
- d. Select your new Lifecycle Environment created in the last step.

- e. **\*\*\* NOTE:** Promoting Content View should take ~3 minutes to complete **\*\*\* As the final step of this exercise, you may move on to Background and to the next lab if you made it this far.**

**\*\*\* NOTE:** If you receive an error about “Required lock is already taken by other running tasks” then your previous request (Content view publish) has not yet completed.

**Background:**

As mentioned earlier, the most common environment use-case would be stage-releasing content through a number of environments in a lifecycle before pushing into production. Publishing a Content View allows you to lock in content at a certain point in time, allowing for intentional and accurate content control when ultimately promoting through your lifecycle stages.

Beyond just locking content, further filtering can be applied to include/exclude specific packages/errata/puppet modules/etc., depending on your exact use-case and business needs.

Should any additional changes be made to a Content View, in order for them to take effect from the Content Hosts perspective, a new version must be published and promoted each time. (**Note:** Older versions of each publish will remain to allow for multi-staging and/or backups of known, stable versions.)

## Lab 2: Satellite for Content Host Management

The preconfigured “rhel7” activation key is required for subsequent lab functions. Do not remove this activation key.

### Create Activation Key:

#### 1. Navigate to Activation Keys

- a. To Navigate to the Activation Keys page, select “**Content**” from your toolbar, and select “**Activation Keys**” from the dropdown.

#### 2. Create Activation Key

- a. From the Activation Keys page, click the blue “**Create Activation Key**” button in the top right corner
- b. Name Activation Key whatever you’d like, and associate with your new Lifecycle Environment and Content View from the previous lab.
  - Note: We have pre-built a Lifecycle Environment and Content View in case you did not do Lab 1. If that’s the case, feel free to use these pre-built objects going forward.
- c. Click “**Save**” to save your Activation Key.



### 3. Attach Subscriptions to Activation Key

- a. From the Activation Keys page, select your newly created activation key.
- b. Select **“Subscriptions”** from the Activation Key toolbar
  - Note: If continuing from the previous step, this may direct you to this page automatically.
- c. Select **“Add”** from the new section available.
- d. Click the checkbox next to the **“Employee SKU”** and then click **“Add Selected”** in the top right corner of the this new section.

### 4. Configure “Enabled Repository” Default

- a. From the Activation Keys page, select your newly created Activation Key

- b. Select **“Repository Sets”** from the Activation Key toolbar
  - Note: If continuing from the last step, this may direct you to this page automatically.
  - Note: the repositories may take a moment to load due to using the Employee SKU in this example.
- c. Click the checkbox next to each of the following repositories
  - Red Hat CloudForms Management Engine 5.8 (RPMs)
  - Red Hat Enterprise Linux 7 Server - Extras (RPMs)
  - Red Hat Enterprise Linux 7 Server (Kickstart)
  - Red Hat Enterprise Linux 7 Server - RH Common (RPMs)
  - Red Hat Enterprise Linux 7 Server (RPMs)
  - Red Hat Enterprise Linux 7 Server - Supplementary (RPMs)
  - Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server
  - Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs)
- d. Click **“Select Action”** followed by **“Override to Enabled”** in the top right corner of the Repository Selection section.

**Background:**

By creating an Activation Key, and associating the created Lifecycle Environments, Content Views, Subscriptions, and Repositories, we’ve enable the ability to activate simple, precise, and efficient deployment of new systems.

Activation Keys can be used to easily automate provisioning, as well as allow for consistent registration and initialization of Content Hosts without requiring users to obtain admin credentials.

## Register Content Hosts:

### 1. Navigate to Content Hosts

- a. To Navigate to the Content Hosts page, select **“Hosts”** from your toolbar, and select **“Content Hosts”** from the dropdown.

### 2. Register Content Host

- a. Click **“Register Content Host”** button in the top right corner
- b. Follow steps outlined in the Satellite WebUI, on each Content Host (See **“Background”** section on next page for client access steps).
  - i. Select Content Source (this will be your desired Satellite or Capsule server). Based on your selection here, it will adjust the following rpm location accordingly.
  - ii. Install pre-built bootstrap RPM. This adjusts configurations on the host to point to your satellite/capsule for registration, subscription, and content delivery.
  - iii. From the client console, run subscription-manager using the Activation Key created in Step 1.
  - iv. Ensure Satellite Tools repository is enabled (If Activate Key was pre-configured to enable this repo by default, this should be set in the last step). This provides access to Katello Agent in the next step.
  - v. Install Katello Agent, which provides the ability to run remote execution(like content patching), as well as displays the errata status (applicable bugs, security, etc) for each Content Host.
- c. Repeat for client systems ic[2-4].example.com (See **“Background”** section on next page for client access steps).

**Background:**

In order to accomplish this, you'll need to login to your client VM, first SSH into your workstation node at `workstation-<GUID>.rhpds.opentlc.com` as `lab-user`, then `sudo` to root. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use "**r3dh4t1!**" as your password.

1. `[lab-user@localhost ~]$ ssh workstation-GUID.rhpds.opentlc.com`
2. `[lab-user@workstation-GUID ~]$ sudo -i`  
`[root@workstation ~]#`

From there, this is your "jumpbox" that will allow you to access each of the client machines (`ic1.example.com` through `ic4.example.com`), also using ssh (ssh passwordless has already been configured for your convenience) and repeat the following commands in `ic2-ic4` machines (`ic1.example.com` has been configured for you):

**IMPORTANT:** Only Do this for all `ic1-ic4` machines, `ic5-ic9` are available, but will be used for a subsequent lab.

In this scenario we are taking over management of these systems from another team, that may have previously had them registered to a different, or older Satellite server.

```
[root@workstation ~]# ssh ic1.example.com
Last login: Mon Apr  9 07:01:12 2018 from tower.example.com
[root@ic1 ~]# yum clean all ; rm -rf /var/cache/yum/*
[root@ic1 ~]# rpm -Uvh http://sat.example.com/pub/katello-ca-consumer-latest.noarch.rpm
[root@ic1 ~]# subscription-manager register --org="Default_Organization"
--activationkey="rhel7"
[root@ic1 ~]# subscription-manager repos --enable=rhel-*-satellite-tools*-rpms
[root@ic1 ~]# yum -y install katello-agent
[root@ic1 ~]# systemctl restart goferd
```

Tip: Here are all commands on individual lines:

```
yum clean all ; rm -rf /var/cache/yum/*
```

```
rpm -Uvh http://sat.example.com/pub/katello-ca-consumer-latest.noarch.rpm
```

```
subscription-manager register --org="Default_Organization" --activationkey="rhel7"
```

```
subscription-manager repos --enable=rhel-*-satellite-tools*-rpms
```

```
yum -y install katello-agent
```

```
systemctl restart goferd
```

Remember to do the above commands on hosts ic1 through ic4.

```
[root@workstation ~]# ssh ic2.example.com
yum clean all ; rm -rf /var/cache/yum/*
rpm -Uvh http://sat.example.com/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --org="Default_Organization" --activationkey="rhel7"
subscription-manager repos --enable=rhel-*satellite-tools*-rpms
yum -y install katello-agent
systemctl restart goferd
```

```
[root@workstation ~]# ssh ic3.example.com
yum clean all ; rm -rf /var/cache/yum/*
rpm -Uvh http://sat.example.com/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --org="Default_Organization" --activationkey="rhel7"
subscription-manager repos --enable=rhel-*satellite-tools*-rpms
yum -y install katello-agent
systemctl restart goferd
```

```
[root@workstation ~]# ssh ic4.example.com
yum clean all ; rm -rf /var/cache/yum/*
rpm -Uvh http://sat.example.com/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --org="Default_Organization" --activationkey="rhel7"
subscription-manager repos --enable=rhel-*satellite-tools*-rpms
yum -y install katello-agent
systemctl restart goferd
```

## Update Content Hosts:

### 1. Navigate to Content Host

- a. To Navigate to the Content Hosts page, select “**Hosts**” from your toolbar, and select “**Content Hosts**” from the dropdown.
- b. Notice after registering with the previously configured activation key, we now have the newly registered Content Host reporting it’s subscription status, Lifecycle Environment and Content View associations, and Installable Errata.
- c. Some of the hosts may have the **.localdomain** domain instead of **.example.com** domain. This is a limitation of the lab environment’s networking, but all actions should complete successfully against these hosts in the environments.

### 2. Navigate to Content Host’s Installable Errata

- a. From the Content Hosts page, select a newly registered Content Host, for example **ic1.example.com**
  - Note: If continuing from the last step, this should direct you to this page automatically.
- b. This will bring up the Details page by default. From here, select “**Errata**” from the Content Hosts toolbar.

### 3. Update Content Host

- a. Looking at the Errata for this system, it's clear the system is severely out of date. Given this, we have a couple of options to update the clients (Specifically Push/Pull).
  - **Pull:** Probably the most widely recognized method, and sometimes faster than selecting hundreds of errata from Satellite UI, from the client system, we can simply run “yum update” from the client console after registering and subscribing:
    - **SSH from your jumpbox to the affected host, in this case,**  
ic1.example.com: # ssh root@ic1.example.com  
[root@ic1 ~]# yum -y update
  - **Push:** We can push specific errata from the satellite, to the client by selecting the specific desired errata from this list, and clicking “**Apply Selected**”.
    - **\*\* NOTE:** For a number of outdated systems, the following packages may need to be updated and goferd needs to be restarted in order to successfully update via push from the Satellite 6.3 host. These packages have already been updated in your environment, but the goferd service will likely still need to be restarted. **\*\***
      - **SSH from your jumpbox to the affected host, in this case,**  
ic1.example.com: # ssh root@ic1.example.com  
[root@ic1 ~]# yum -y upgrade katello-agent \  
katello-host-tools katello-host-tools-fact-plugin \  
pulp-rpm-handlers qpidd-proton-c  
[root@ic1 ~]# systemctl restart goferd



- **\*\* NOTE:** If above issue is encountered, it will be evident by either the error “No handler for: {‘type’: u’erratum’}”, or a communication timeout. **\*\***
  - To Install All Errata you can select to view up to 100 available errata at a time at the bottom of the screen, and select all from the master checkbox in the top left corner of the view, then Apply Selected.
    - Repeat this step until all errata have been applied.
    - **For systems registered to a prior Satellite, like this scenario:** there may be some old remnants of the previous system registration. In this case, the remote push errata update may fail. You can login to the system and as root run the commands:  
**# yum clean all ; rm -rf /var/cache/yum/\***
      - This clears out the old yum repo data from the previous satellite registration, and prepares the system for updates from your newly configured Satellite server.
- b. Once the errata has been applied you can verify there’s no pending erratas on the UI.

# Lab 3: Proactive Security and Automated Risk Management with Red Hat Insights

## Goal of Lab

The goal of this lab is to introduce you to the proactive security capabilities of Red Hat Insights. This lab assumes the following:

- You started all your VMs, per the instructions in Lab 0.
- Did not delete any of the key items in the pre-populated Satellite 6 configuration, which are necessary for Insights to work properly.

## Introduction

Red Hat Insights was designed to proactively evaluate the security, performance, and stability of your Red Hat platforms by providing prescriptive analytics of your systems. Insights helps move you from reactive to proactive systems management, delivers actionable intelligence, and increases visibility of infrastructure risks and the latest security threats. Operational analytics from Insights empowers you to prevent downtime and avoid firefighting, while also responding faster to new risks.

**In this lab, we will focus only on the specific security features of Red Hat Insights.**

Red Hat Insights recommendations are tailored for the individual system where risk is detected. This allows you to be certain that actions identified by Insights are validated and have a verified resolution for each detected risk, reducing false positives you may experience from critical risks identified by third-party security scanners. Insights provides predictive analysis of security risk in your infrastructure based on a constantly evolving threat feed from Red Hat.

Through analysis of Insights metadata and curated knowledge based on over fifteen years of enterprise customer support, Red Hat is able to identify critical security vulnerabilities, statistically frequented risks, and known bad configurations. We scale this knowledge to our customers with Insights reporting and alerts, allowing prediction of what will happen on a monitored system, why it will happen, and how to fix a problem before it can occur.

Red Hat Insights functionality is integrated into Red Hat's Customer Portal, Satellite, CloudForms, and Ansible Tower by Red Hat. Recommendations from Insights are human-readable and in most cases can simply be copy and pasted into the terminal to resolve the issue. You may also automate remediation of hosts in your infrastructure with Insights generated Ansible playbooks or Ansible Tower integration.

# Fixing the payload injection security issue in your system using Red Hat Insights from the Satellite UI

## Cleaning your environment

As you may have discerned from the previous lab exercises, you've been put in charge of some new-to-you servers. These servers were previously registered to an old Satellite server. You'll need to re-register them to your company's new Satellite, using the provided bootstrap script to quickly and easily begin managing them, and fix some of the critical vulnerabilities with these systems. In a real-world-scenario, Insights can be installed automatically by Satellite upon provisioning or registration of a new system to the Satellite, which means you do not have to manually login and register each host. These steps can be easily automated with Ansible or a shell script, but for sake of completion we will perform them manually in these labs with only a few systems.

To login to your client VM, first SSH into your workstation node at workstation-*<GUID>*.rhpds.opentlc.com as lab-user. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use "r3dh4t1!" as your password.

- a. **[lab-user@localhost ~]\$ ssh workstation-GUID.rhpds.opentlc.com**
- b. **[lab-user@workstation-GUID ~]\$ sudo -i**  
**[root@workstation ~]#**

From it, jump into every single one of the insights client machines (from ic5.example.com to ic9.example.com), also using ssh (ssh passwordless has already been configured for your convenience):

```
[root@workstation ~]# ssh ic5.example.com
Last login: Mon Apr  9 07:01:12 2018 from tower.summit.example.com
```

Make sure old satellite info is removed:

```
[root@ic5 ~]# yum clean all ; rm -rf /var/cache/yum/*
[root@ic5 ~]# subscription-manager clean
[root@ic5 ~]# rm -fv /etc/rhsm/ca/katello*
[root@ic5 ~]# rm -fv /etc/rhsm/facts/katello*
```

```
[root@ic5 ~]# rm -rfv /var/lib/puppet
[root@ic5 ~]# yum remove -y katello-ca-consumer*
```

## Adding your Insights hosts to Satellite

Pull down new bootstrap script from new Satellite 6 server:

```
[root@ic5 ~]# curl https://sat.example.com/pub/bootstrap.py > bootstrap.py --insecure
[root@ic5 ~]# chmod +x bootstrap.py
```

Run the bootstrap.py script using the Satellite manifest that uses certificate based authorization in order to register your machines with Satellite as follows (provide the password when required):

```
# ./bootstrap.py -l admin -s sat.example.com -o 'EXAMPLE.COM' -L 'Default Location' -g rhel7 -a rhel7
```

### Foreman Bootstrap Script

This script is designed to register new systems or to migrate an existing system to a Foreman server with Katello  
admin's password:

[...]

Loaded plugins: product-id, subscription-manager

```
[SUCCESS], [2018-04-10 12:44:33], [/usr/bin/yum -y remove rhn-setup rhn-client-tools
yum-rhn-plugin rhnsd rhn-check rhnlib spacewalk-abrt spacewalk-oscaps osad
'rh-*-*rhui-client' 'candlepin-cert-consumer-*'], completed successfully.
```

IMPORTANT! Do not forget to repeat the steps for:

On you workstation \$ sudo -i then

```
# ssh ic6.example.com
yum clean all ; rm -rf /var/cache/yum/*
subscription-manager clean
rm -fv /etc/rhsm/ca/katello*
rm -fv /etc/rhsm/facts/katello*
rm -rfv /var/lib/puppet
yum remove -y katello-ca-consumer*
curl https://sat.example.com/pub/bootstrap.py > bootstrap.py --insecure
chmod +x bootstrap.py
./bootstrap.py -l admin -s sat.example.com -o 'EXAMPLE.COM' -L 'Default Location' -g rhel7
-a rhel7
```

```
# ssh ic7.example.com
```

```
yum clean all ; rm -rf /var/cache/yum/*
subscription-manager clean
rm -fv /etc/rhsm/ca/katello*
rm -fv /etc/rhsm/facts/katello*
rm -rfv /var/lib/puppet
yum remove -y katello-ca-consumer*
curl https://sat.example.com/pub/bootstrap.py > bootstrap.py --insecure
chmod +x bootstrap.py
./bootstrap.py -l admin -s sat.example.com -o 'EXAMPLE.COM' -L 'Default Location' -g rhel7
-a rhel7
```

```
# ssh ic8.example.com
yum clean all ; rm -rf /var/cache/yum/*
subscription-manager clean
rm -fv /etc/rhsm/ca/katello*
rm -fv /etc/rhsm/facts/katello*
rm -rfv /var/lib/puppet
yum remove -y katello-ca-consumer*
curl https://sat.example.com/pub/bootstrap.py > bootstrap.py --insecure
chmod +x bootstrap.py
./bootstrap.py -l admin -s sat.example.com -o 'EXAMPLE.COM' -L 'Default Location' -g rhel7
-a rhel7
```

```
# ssh ic9.example.com
yum clean all ; rm -rf /var/cache/yum/*
subscription-manager clean
rm -fv /etc/rhsm/ca/katello*
rm -fv /etc/rhsm/facts/katello*
rm -rfv /var/lib/puppet
yum remove -y katello-ca-consumer*
curl https://sat.example.com/pub/bootstrap.py > bootstrap.py --insecure
chmod +x bootstrap.py
./bootstrap.py -l admin -s sat.example.com -o 'EXAMPLE.COM' -L 'Default Location' -g rhel7
-a rhel7
```

And repeat the steps. If no errors, proceed to the next section, If there was an error see below:

**NOTE: If bootstrap.py fails with this error:**

**[RUNNING], [2018-04-19 06:28:43], [Calling Foreman API to create a host entry associated with the group & org]**

**An error occurred: HTTP Error 422: Unprocessable Entity**

**url: https://sat.example.com:443/api/v2/hosts/**

code: 422

[...]

"full\_messages": [

"Name has already been taken"

[...]

*Output truncated*

then it means machines have already been registered using subscription manager, in that case, you just have to delete the hosts from the Satellite UI and re-run bootstrap. See the below graphics. If your hosts were successful, continue past these screenshots.

In your Firefox web browser, click on the tab you have opened to your Red Hat Satellite 6.3 UI. Log back in with **admin** as the username and **r3dh4t!** as your password.

Go to **Hosts** → **All Hosts** and you will see them listed as follows:



Select the **ic[1-4].example.com** hosts and delete them by going to **Select Action** → **Delete Hosts**, as in the following screen.



Then, run the **bootstrap.py** script again.

When logging into the Satellite UI you should see your systems registered. Go to **Hosts** → **Content Hosts** and you will see them listed as follows:

## Installing the Insights client

Now it's the time to install the Insights RPM and register your system to Red Hat Insights.

bootstrap.py should have taken care of this for you, so navigate to

NOTE: On RHEL 7.5 client RPM has been renamed to insights-client, but this laboratory machines are using RHEL 7.0 and 7.3 for demonstration purposes, so the package name is still the old one.

To install Insights RPM in each of your systems issue the following command:

```
[root@ic5 ~]# yum -y install redhat-access-insights
```

And then, simply register each machine with Red Hat Insights as follows:

```
[root@ic5 ~]# redhat-access-insights --register
```

**Automatic daily scheduling for Insights has been enabled.**

**Starting to collect Insights data**

**Uploading Insights data, this may take a few minutes**

**Upload completed successfully!**

## Fixing the payload injection security issue

Now, going back to the Satellite UI, click on **Red Hat Insights → Overview**, where you should see all your registered systems, actions summary (highlighted by priority) as well as latest updates from Red Hat.

In this lab, we will fix the specific “**Kernel vulnerable to man-in-the-middle via payload injection (CVE-2016-5696)**” on your client VMs without causing downtime.

**STEPS:**

2. From your Satellite 6.3 UI, click on **Red Hat Insights → Inventory**.
3. Click on your client VM, which is **ic6.example.com**. You will see the list of issues affecting it when clicking on the system name.



4. Notice that your system shows up with multiple security vulnerabilities.

*Note: Our objective is to fix the payload injection problem without causing downtime, and see that it no longer appears as a vulnerability in Insights. Specifically, this payload injection problem causes the kernel to be vulnerable to man-in-the-middle via payload injection. A flaw was found in the implementation of the Linux kernel's handling of networking challenge ack ([RFC 5961](#)) where an attacker is able to determine the shared counter. This flaw allows an attacker located on different subnet to inject or take over a TCP connection between a server and client without needing to use a traditional man-in-the-middle (MITM) attack.*

5. Use your browser's search function to search for “**payload injection**”.

*Note: Reading the description for the vulnerability shows that the **sysctl** variable is set to a level that allows being exploited. We want to do the active mitigation by changing the **sysctl** variable and making it permanent on reboot. In this case, we do not want to update the kernel or reboot since we don't want downtime.*

6. If not already there, login to your client VM, first SSH into your workstation node at workstation- <GUID>.rhpds.opentlc.com as lab-user. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use “r3dh4t1!” as your password.

```
[lab-user@localhost ~]$ ssh workstation-GUID.rhpds.opentlc.com
```

```
[lab-user@workstation-GUID ~]$ sudo -i
```

```
[root@workstation ~]#
```

7. Now that you are in the workstation node, SSH into your RHEL7 client/host.

```
# ssh ic6
```

8. Now, as **root**, perform the recommended active mitigation. Edit the **/etc/sysctl.conf** file to add the mitigation configuration, and reload the kernel configuration:

```
# echo "net.ipv4.tcp_challenge_ack_limit = 2147483647" >> /etc/sysctl.conf  
# sysctl -p
```

```
net.ipv4.tcp_challenge_ack_limit = 100  
vm.legacy_va_layout = 0  
net.ipv4.tcp_challenge_ack_limit = 2147483647
```

9. After applying the active mitigation, we want to have the system report any changes, run the following command as root on ic6.example.com:

```
# redhat-access-insights  
Starting to collect Insights data  
Uploading Insights data, this may take a few minutes  
Upload completed successfully!
```

Wait until this step completes before moving to the next step.

10. From your Satellite 6.3 UI, click on **Red Hat Insights → Inventory**.

11. Click on your client VM, which is **ic6.example.com**. You will notice that the number of actions has decreased (from 18 to 17).

12. Use your browser's search function to search for "**payload injection**". You will notice that this payload injection issue is no longer listed due to fixing the vulnerability.

Congratulations, you're no longer vulnerable to the payload injection vulnerability!

## Automatically fixing the payload injection security issue via Ansible Playbook

It is also possible to automate some of the issues with an Ansible Playbook that Insights provides us. You can see that in the top left corner of every single issue with the Ansible logo in blue if a playbook is available, or in grey if it's not.

In the particular case of the payload injection security issue, an Ansible Playbook is available for us.

Now we need to create a plan in which the issues that are found will be solved using an Ansible Playbook. In order to do so, from your Satellite 6.3 UI, click on **Red Hat Insights → Planner**.

And once there, click on **Create a plan**.

Fill in the boxes as in the example, and do not forget to select only the payload injection security issue and select **ic7.example.com** as the system in which this solution is to be applied. Then click “save”.

As seen in the previous part of this laboratory, there are two ways to solve this issue, one is by updating the kernel, and the other one is apply the needed changes to the **/etc/sysctl.conf** file to add the mitigation configuration, and reload the kernel configuration.

Insights gives us the opportunity to choose the resolution that we want. Please make sure to select **“Set sysctl ip4 challenge ack limit”** as your preferred choice and then click on the Save button.

Once the plan is saved, the planner screen is shown where you can see the newly created plan, as well as the issues it resolves and the systems affected.



You should now download the playbook, however, it's been already downloaded for your convenience to the tower machine.

If not already there, login to your client VM, first SSH into your workstation node at workstation- <GUID>.rhpbs.opentlc.com as the lab-user. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use "r3dh4t1!" as your password.

```
[lab-user@localhost ~]$ ssh workstation-GUID.rhpbs.opentlc.com
```

```
[lab-user@workstation-GUID ~]$ sudo -i
[root@workstation ~]#
```

Now that you are in the workstation node, SSH into the tower machine in order perform the recommended active mitigation with the Ansible Playbook.

```
[root@workstation ~]# ssh tower
```

Inspect the Ansible Playbook that Insights has created automatically for you:

```
[root@tower ~]# less payload-injection.yml
```

```
---
```

```
# Red Hat Insights has recommended one or more actions for you, a system administrator,
to review and if you
# deem appropriate, deploy on your systems running Red Hat software. Based on the
analysis, we have automatically
# generated an Ansible Playbook for you. Please review and test the recommended actions
and the Playbook as
# they may contain configuration changes, updates, reboots and/or other changes to your
systems. Red Hat is not
# responsible for any adverse outcomes related to these recommendations or Playbooks.
#
# Addresses maintenance plan 34918 (payload)
# https://access.redhat.com/insights/planner/34918
# Generated by Red Hat Insights on Wed, 11 Apr 2018 09:54:52 GMT

# Kernel vulnerable to man-in-the-middle via payload injection (CVE-2016-5696)
# Identifier: (CVE_2016_5696_kernel|KERNEL_CVE_2016_5696_URGENT,105,mitigate)
# Version: c988b9061f0c3720900ae391d72a59a89bf57294
- name: Set sysctl ipv4 challenge ack limit
```

```
hosts: "ic7.example.com"
become: true
tasks:
  - name: "set the sysctl net.ipv4.tcp_challenge_ack_limit = 2147483647"
    sysctl:
      name: net.ipv4.tcp_challenge_ack_limit
      value: 2147483647
      sysctl_set: true
```

```
- name: run insights
  hosts: ic7.example.com
  become: True
  gather_facts: False
  tasks:
    - name: run insights
      command: redhat-access-insights
      changed_when: false
```

Now, simply proceed to remediate the payload injection security issue by executing the Ansible Playbook as follows:

```
[root@tower ~]# ansible-playbook payload-injection.yml
```

```
PLAY [Set sysctl ipv4 challenge ack limit]
```

```
*****
```

```
TASK [Gathering Facts]
```

```
*****
```

```
*****
```

```
ok: [ic7.example.com]
```

```
TASK [set the sysctl net.ipv4.tcp_challenge_ack_limit = 2147483647]
```

```
*****
```

```
changed: [ic7.example.com]
```

```
PLAY [run insights]
```

```
*****
```

```
*****
```

```
TASK [run insights]
```

```
*****
```

```
*****
```

ok: [ic7.example.com]

## PLAY RECAP

```
*****  
*****
```

```
ic7.example.com      : ok=3  changed=1    unreachable=0    failed=0
```

Please note that when the execution is completed, the Insights agent is also run, so the latest state of the system is reporting into Insights automatically.

Now from the Satellite UI, click on **Red Hat Insights → Inventory** you will notice that system **ic7.example.com** has one less issue, just like **ic6.example.com** (both with 17).

Bonus! Automatically fix all the issues on systems **ic8.example.com** and **ic9.example.com** that have playbooks

From the Satellite UI, click on **Red Hat Insights → Inventory** so we can focus on systems **ic8.example.com** and **ic9.example.com**, please notice these two show 18 actions each to be solved.

In the inventory screen, select both systems and click on Actions, on the top left corner, and then select Create a new Plan / Playbook.

This way, we are going to create an Ansible Playbook-based plan to solve issues on those two specific systems (systems can also be grouped as per our convenience, from that very same menu).

The Plan / Playbook Builder screens appears. Please make sure to fill the boxes as follows:

Plan name: ic8-ic9-all

Actions: all (do this by clicking on the box by the Action label at the top).

Your screen should look like:

Then click on the Save button in the bottom right corner.

As before, you are given the option to choose between different ways to solve your issues. In this lab, we've chosen to go for the ones that do not require a reboot, in order to save some time.

The plan description screen appears.

You should see all the issues this plan is going to solve as well as the affected systems.

Scrolling down the screen, you should be able to download the playbook. Per your convenience, this has already been downloaded to the tower machine.

Like in the previous exercise, we need to log into the tower machine in order to run the Ansible Playbook.

If not already there, login to your client VM, first SSH into your workstation node at workstation- <GUID>.rhpds.opentlc.com as the lab-user. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use “r3dh4t1!” as your password.

```
[lab-user@localhost ~]$ ssh workstation-GUID.rhpds.opentlc.com
```

```
[lab-user@workstation-GUID ~]$ sudo -i
```

```
[root@workstation ~]#
```

Now that you are in the workstation node, SSH into the tower machine in order perform the recommended active mitigation with the Ansible Playbook.

```
[root@workstation ~]# ssh tower
```

Inspect the Ansible Playbook that Insights has created automatically for you:

```
[root@tower ~]# vi ic8-ic9-all.yml
```

```
---
```

```
# Red Hat Insights has recommended one or more actions for you, a system administrator,
to review and if you
```

```
# deem appropriate, deploy on your systems running Red Hat software. Based on the
analysis, we have automatically
```

```
# generated an Ansible Playbook for you. Please review and test the recommended actions
and the Playbook as
```

```
# they may contain configuration changes, updates, reboots and/or other changes to your
systems. Red Hat is not
```

```
# responsible for any adverse outcomes related to these recommendations or Playbooks.
#
```

```
# Addresses maintenance plan 34921 (ic8-ic9-all)
```

```
# https://access.redhat.com/insights/planner/34921
```

```
# Generated by Red Hat Insights on Wed, 11 Apr 2018 13:59:14 GMT
```

```
# Warning: Some of the rules in the plan do not have Ansible support and this playbook does
not address them!
```

```
- name: run insights to obtain latest report info
```

```
hosts: ic8.example.com,ic9.example.com
```

```
become: True
```

```
tasks:
```

```
  - name: determine insights version
```

```
    shell: 'redhat-access-insights --version'
```

```
    changed_when: false
```

```
    register: insights_version
```

```
  - when: insights_version.stdout[0:2] != '1.'
```

**Block:**

[...]

*Output truncated*

Now, simply proceed to remediate the issues by executing the Ansible Playbook as follows:

```
[root@tower ~]# ansible-playbook ic8-ic9-all.yml
```

[...]

*Output truncated*

Please note that when the execution is completed (this may take a while), the Insights agent is also run, so the latest state of the system is reporting into Insights automatically.

Note that some of the issues won't be able to be solved by the ansible playbook and you'd need to do some extra steps.

Now from the Satellite UI, click on **Red Hat Insights → Inventory** you will notice that your systems have few issues.



# Lab 4: Automatic Remediation with Red Hat Insights and Ansible Tower

## Goal of Lab

The goal of this lab is to introduce you to the proactive security capabilities of Red Hat Insights and automatic remediation with Ansible Tower. This lab assumes that you started all your VMs, as instructed in Lab 0, which is necessary for Red Hat Insights to work properly. This will also be your first use of Tower in this lab if you are doing the labs sequentially. You should require no prior Tower knowledge to follow all instructions.

## Introduction

Red Hat Insights was designed to proactively evaluate the security, performance, and stability of your Red Hat platforms by providing prescriptive analytics of your systems. Red Hat Insights helps move you from reactive to proactive systems management, delivers actionable intelligence, and increases visibility of infrastructure risks and the latest security threats. Operational analytics from Red Hat Insights empowers you to prevent downtime and avoid firefighting while responding faster to new risks.

## Setting up an Insights Scan Project

Tower supports integration with Red Hat Insights. Once a host is registered with Insights, it will be continually scanned for vulnerabilities and known configuration conflicts. Each of the found problems may have an associated fix in the form of an Ansible playbook. Insights users create a maintenance plan to group the fixes and, ultimately, create a playbook to mitigate the problems. Tower tracks the maintenance plan playbooks via an Insights project in Tower. Authentication to Insights via Basic Auth, from Tower, is backed by a special Insights Credential, which must first be established in Tower. To ultimately run an Insights Maintenance Plan in Tower, you need an Insights project, an inventory, and a Scan Job template.

## Create Insights Credentials

If not already there, login to your client VM, first SSH into your workstation node at workstation- <GUID>.rhpds.opentlc.com as the lab-user. An ssh key is already in the home directory of your laptop, which should allow you to login without a password. Should a password be required, use “r3dh4t1!” as your password.

```
[lab-user@localhost ~]$ ssh workstation-GUID.rhpds.opentlc.com
```

```
[lab-user@workstation-GUID ~]$ sudo -i  
[root@workstation ~]#
```

In your Firefox web browser, click on the tab you have opened to your Red Hat Ansible Tower UI. Log back in with **admin** as the username and **r3dh4t1!** as your password.

Credentials have already been provided for your convenience.

## Creating an inventory


The Insights playbook contains a **hosts:** line where the value is the hostname that Insights itself knows about, which may be different than the hostname that Tower knows about. Therefore, make sure that the hostnames in the Tower inventory match up with the system in the Red Hat Insights Portal.

For your convenience, the Insights inventory has been already created for you.


Please note that typically, your inventory already contains Insights hosts. Tower just doesn't know about them yet. The Insights credential allows Tower to get information from Insights about an Insights host. Tower identifying a host as an Insights host can occur without an Insights credential with the help of scan facts.yml file.

In order for Tower to utilize Insights Maintenance Plans, it must have visibility to them. Create and run a scan job against the inventory using a stock manual scan playbook (this is provided by Red Hat to its customers).

## Creating a Scan Project

1. Click the **Projects** main link to access the Projects page.
2. Click the  button, which launches the **New Project** window.
3. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:
  - **Name:** Insights Scan Summit
  - **Organization:** Red Hat's Management BU Example.com
  - **SCM Type:** Select **Git**.
  - Upon selecting the SCM type, the **Source Details** field expands.
4. In the **SCM URL** field, enter **https://github.com/ansible/awx-facts-playbooks**. This is the location where the scan job template is stored.
5. Click **Save** when done.


Your screen should look as the following:

All SCM/Project syncs occur automatically the first time you save a new project. However, if you want them to be updated to what is current in Insights, manually update the SCM-based project by clicking the  button under the project's available Actions.


Syncing imports into Tower any Maintenance Plans in your Insights account that has a playbook solution. It will use the default Plan resolution. Notice that the status dot beside the name of the project updates once the sync has run.

Now it's the time to put all the pieces together, by using a job template that uses the fact scan playbook.

## Creating a Job Template

1. Click the **Templates** main link to access the Templates page.
2. Click the  button and select **Job Template**, which launches the New Job Template window.
3. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:
  - **Name:** Insights Scan Summit
  - **Job Type:** Choose **Run** from the drop-down menu list.
  - **Inventory:** Example.com Satellite Inventory
  - **Project:** Enter the name of the Scan project you previously created, Insights Scan Summit.
  - **Playbook:** Select scan\_facts.yml from the drop-down menu list. This is the playbook associated with the Scan project you previously set up.
  - **Credential:** Example.com SSH Password. The credential does not have to be an Insights credential, but machine (also created for your convenience).
  - Click to select **Use Fact Cache** from the Options field.
4. Click **Save** when done.

Your screen should look as follows:

Click the  icon to launch the scan job template, the output you should see is something like this, if it all went as expected:

With this, we know the state our machines are in and issues they may have that should be remediated. This is what we are going to do in the next part of the lab.

Now, we can see Insights data from the Ansible Tower UI.

### Viewing Insights data into Tower

1. Click the **Inventories** main link to access the Inventories page.
2. In the list of inventories, click to open the details of your Insights inventory.
3. Click the **Hosts** tab to access the Insights hosts that have been loaded from the scan process.

4. Click to open one of the hosts that was loaded from Insights (ic6.example.com, for instance).

Notice the Insights tab is now shown on Hosts page. This indicates that Insights and Tower have reconciled the inventories and is now set up for one-click Insights playbook runs. Click on it.

You now can see a list of issues Insights has identified and whether or not the issues can be resolved with a playbook is also shown.



## BONUS!

### Automatically remediate Insights Inventory

#### Creating a Remediation Project

Remediation of an Insights inventory allows Tower to run Insights playbooks with a single click, instead of using the CLI as we did in the previous Lab.

First thing is to create a remediation project, similar to the scan project previously created.

1. Click the **Projects** main link to access the Projects page.
2. Click the  button, which launches the **New Project** window.
3. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:
  - **Name:** Insights Remediation Summit
  - **Organization:** Red Hat's Management BU Example.com, or click the  button and select it from the pop-up window.
  - **SCM Type:** Select **Manual**.
  - **Playbook:** Select insights remediation summit
  - **Project Base Path:** /var/lib/awx/projects (automatically populated).
4. Click on Save


Your screen should look as follows:

NOTE: In the real world, we would be accessing directly to our Red Hat account, however for the limitations of the configuration in this lab, this is not possible. In the real world, we'd need to choose Red Hat Insights as the SCM type and our Customer Portal Credentials (later in this lab we'd need to use them, and they have already been created for your convenience).

In the real world, you should be seeing something like the following screen:

## Creating a Remediation Job Template

Similar to the one created before, we are to create a remediation job template that can use the remediation project we have just created.

4. Click the **Templates** main link to access the Projects page.
5. Click the  button and select **Job Template**, which launches the New Job Template window.
6. Enter the appropriate details into the required fields, at minimum. Note the following fields requiring specific Insights-related entries:
  - **Name:** Insights Remediation Summit.
  - **Job Type:** Choose **Run** from the drop-down menu list.
  - **Inventory:** Example.com Satellite Inventory
  - **Project:** Enter the name of the project you previously created, Insights Remediation Summit.
  - **Playbook:** Select summit.yml from the drop-down menu list. This is the playbook associated with the Remediation project you previously set up.
  - **Credential:** Example.com SSH Password. The credential does not have to be an Insights credential, but machine (also created for your convenience).
7. Click **Save** when done.

Your screen should look as follows:



Now, simply execute it by clicking on the rocket next to the name of the template in the templates list down the screen.

The playbook execution screen appears and shows you the result.

# Lab 5: Introduction to Ansible Tower

## Why Ansible Tower?

Ansible Tower is a web-based UI that provides an enterprise solution for IT automation. It

- has a user-friendly dashboard
- complements Ansible, adding automation, visual management, and monitoring capabilities.
- provides user access control to administrators.
- graphically manages or synchronizes inventories with a wide variety of sources.
- a RESTful API
- And much more...

## Architecture and Installation

Ansible Tower is a Django web application that requires that Ansible be installed. It also relies on a database back-end that uses PostgreSQL. The default installation installs all components (web application, REST API and database) required by Ansible Tower on a single machine.

## About this Lab

This lab is about giving an overview and providing hands-on experience with Ansible Tower. It's assumed that attendees already have some basic Ansible knowledge (concepts, Playbook writing etc) as the labs are covering Tower-specific topics.

## Ansible Tower Concepts

To start using Ansible Tower, some concepts and naming convention should be known.

### Dashboard

When logged in to Ansible Tower using the web UI, the administrator can view a graph that shows

- recent job activity
- the number of managed hosts
- quick pointers to lists of hosts with problems.

The dashboard also displays real time data about the execution of tasks completed in playbooks.

### Projects

Projects are logical collections of Ansible playbooks in Ansible Tower. These playbooks either reside on the Ansible Tower instance, or in a source code version control system supported by Tower.

## **Inventories**

An Inventory is a collection of hosts against which jobs may be launched, the same as an Ansible inventory file. Inventories are divided into groups and these groups contain the actual hosts. Groups may be populated manually, by entering host names into Tower, or from one of Ansible Tower's supported cloud providers.

## **Credentials**

Credentials are utilized by Tower for authentication when launching Jobs against machines, synchronizing with inventory sources, and importing project content from a version control system. Credential configuration can be found in the Settings.

Tower credentials are imported and stored encrypted in Tower, and are not retrievable in plain text on the command line by any user. You can grant users and teams the ability to use these credentials, without actually exposing the credential to the user.

## **Templates**

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams. To execute a job, Tower requires that you first create a job template.

## **Jobs**

A job is basically an instance of Tower launching an Ansible playbook against an inventory of hosts.

## Create an Inventory

The first thing we need is an inventory of your managed hosts. This is the equivalent of an inventory file in Ansible Engine.

For your convenience, all the inventories we will use today are already created for you, but you can add a dynamic inventory source from many different sources, such as Satellite 6 as we have already added in this lab. This creates a powerful way to manage systems with Ansible Tower that you're also managing with Satellite.

## Machine Credentials

One of the great features of Ansible Tower is to make credentials usable to users without making them visible. To allow Tower to execute jobs on remote hosts, you must configure connection credentials.

As mentioned, for your convenience we have already created these for you.

## Run Ad Hoc Commands

As you've probably done with Ansible before, you can run ad hoc commands from Tower as well.

- In the web UI go to **INVENTORIES → Example.com Clients**
- Click the **HOSTS** button to change into the hosts view and select the two hosts by ticking the boxes to the left of the host entries.
- Click **RUN COMMANDS**. In the next screen you have to specify the command:
  - As **MODULE** choose **Ping**
  - For **MACHINE CREDENTIAL** click the magnifying glass icon and select **Example.com SSH Password**.
  - Click **LAUNCH**, sit back and enjoy the show... when completed you should see **SUCCESS** with a pong response like below.

Tip: The simple **Ping** module doesn't need options. For the command module you need to supply the command to run as an argument.

Try other modules in ad hoc commands, as well:

- Find the userid of the executing user using an ad hoc command.
  - **MODULE:** command
  - **ARGUMENTS:** id
- Print out */etc/shadow*.
  - **MODULE:** command
  - **ARGUMENTS:** cat */etc/shadow*
- Re-run the last ad hoc command but this time tick the **ENABLE PRIVILEGE ESCALATION** box.

Tip: For tasks that have to run as root you need to escalate the privileges. This is the same as the **become: yes** you've probably used often in your Ansible Playbooks.

## Using Variables

You might have seen you can add variables for a host in the inventory.

- Go to **INVENTORIES → Example.com Clients**, switch to the **HOSTS** view and edit *ic1.example.com* by clicking the pen icon.
- Add a variable named "file" by putting **file: /etc/passwd** in the **VARIABLES** field under the YAML start (the three dashes) if it isn't already there
- Click **SAVE**

- Now run an ad hoc command (use steps from last exercise, with the command module and arguments instead of the ping module) on *ic1.example.com*
  - **MODULE:** command
  - **ARGUMENTS:** cat {{ file }}
  - **MACHINE CREDENTIAL:** Example.com SSH Password

Tip: There has to be a blank between the **file:** and the content string.

- The output should now show the content of the file.

## Add a new Project

A Tower **Project** is a logical collection of Ansible playbooks. You can manage playbooks by either placing them manually on your Tower server, or by placing your playbooks into a source code management (SCM) system supported by Tower, including Git, Subversion, and Mercurial. You should definitely keep your Playbooks under version control. In this lab we'll use Playbooks kept in a Git repository.

## Setup Git Repository

For this lab you will use a pre-configured Git repository on tower.example.com that can be accessed via SSH. A Playbook to set the Message Of The Day(motd) has already been committed to the repository:

```
$ cat motd.yml
```

```
---
```

```
- hosts: all
```

```
  tasks:
```

```
    - template:
```

```
      src: ./motd.j2
```

```
      dest: /etc/motd
```

```
$ cat motd.j2
```

```
{{ motd }}
```

#### Create the Project

- In the **PROJECTS** view click **+ADD**
- **NAME:** Easy MOTD
- **ORGANIZATION:** Red Hat's Management BU
- **SCM TYPE:** Git
- Point to the Git repo on github
- **SCM URL:** https://github.com/IPvSean/easy\_motd
- **SCM CREDENTIAL:** <leave blank>
- **SCM UPDATE OPTIONS:** Tick all three boxes to always get a fresh copy of the repository and to update the repository when launching a job.
- Click **SAVE**



Sync the Project again with the Git repository by going to the **PROJECTS** view and clicking the cloudy **Start an SCM Update** icon to the right of the Project.

- After starting the sync job, go to the **JOBS** view, find your job and have a look at the details. You should see successful output where there was a change on the host, that is the playbook being downloaded to Tower.

## Create a Job Template and Run a Job

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. So before running an Ansible **Job** from Tower you must create a **Job Template** that pulls together:

- **Inventory:** On what hosts should the job run?
- **Credentials** for the hosts
- **Project:** Where is the Playbook?
- **What Playbook** to use?

Okay, let's just do that:

- Go to the **TEMPLATES** view and click **+ADD → JOB TEMPLATE**
  - **NAME:** MOTD
  - **JOB TYPE:** Run
  - **INVENTORY:** Example.com Clients
  - **PROJECT:** Easy MOTD
  - **PLAYBOOK:** motd.yml
  - **CREDENTIAL:** Example.com SSH Password
  - We need to run the tasks as root so check **Enable privilege escalation**
  - **This is the fun part, where we get to define the MOTD variable.**
    - If the **LIMIT** field is blank this will change the MOTD on all clients in the Example.com Clients inventory. To change only one or a few systems you can specify them in the **LIMIT** field like in the screenshot.
    - In the **VARIABLES** box for the template provide the value  
motd: This is my MOTD from the Management Lab, Summit 2018!

- Click **SAVE**

Start a Job using this Job Template by going to the **TEMPLATES** view and clicking the rocket icon. Have a good look at the information the view provides.

Tip: This might take some time because you configured the Project to update the SCM on launch.

After the Job has finished go to the **JOBS** view:

- All jobs are listed here, you should see directly before the Playbook run an SCM update was started.
- This is the Git update we configured for the **PROJECT** on launch!

## Ansible Tower Role Based Access Control

You have already learned how Tower separates credentials from users. Another advantage of Ansible Tower is the user and group rights management.

### Ansible Tower Users

There are three types of Tower Users:

- **Normal User:** Have read and write access limited to the inventory and projects for which that user has been granted the appropriate roles and privileges.
- **System Auditor:** Auditors implicitly inherit the read-only capability for all objects within the Tower environment.
- **System Administrator:** Has admin, read, and write privileges over the entire Tower installation.

Let's create a user:

- Go to **Settings** by clicking the "gear"-icon and choose **USERS**
- Click **+ADD**
- Fill in the values for the new user:
  - **FIRST NAME:** Werner
  - **LAST NAME:** Web
  - **ORGANIZATION:** Red Hat's Management BU Example.com
  - **EMAIL:** wweb@example.com
  - **USERNAME:** wweb
  - **USER TYPE:** Normal User
  - **PASSWORD:** P@ssword1!
  - Confirm password
- Click **SAVE**

## Ansible Tower Teams

A Team is a subdivision of an organization with associated users, projects, credentials, and permissions. Teams provide a means to implement role-based access control schemes and delegate responsibilities across organizations. For instance, permissions may be granted to a whole Team rather than each user on the Team.

Create a Team:

- Go to **Settings** and choose **TEAMS**.
- Click **+ADD** and create a team named Web Content.
- **Select ORGANIZATION:** Red Hat's Management BU Example.com
- Click **SAVE**

Now you can add a user to the Team:

- Switch to the Users view of the Web Content Team by clicking the **USERS** button.
- Click **+ADD** and select the wweb user, you may need to click to Page 2 in the user list.
- The dialog now asks for a role to assign, the following permission settings are available:
  - Admin: This User should have privileges to manage all aspects of the team
  - Member: This User should be a member of the team
- Assign the **Member** role.
- Click **SAVE**
- You may receive a 404 error due to system load, if this happens, ignore the error, simply refresh the page, and click Users to verify wweb is listed in the Web Content group as a Member.

Now click the **PERMISSIONS** button in the **TEAMS** view, you will be greeted with "No Permissions Have Been Granted".

Permissions allow to read, modify, and administer projects, inventories, and other Tower elements. Permissions can be set for different resources.

## Granting Permissions

To allow users or teams to actually do something, you have to set permissions. The user **wwweb** should only be allowed to modify content of the assigned webserver.

- In the Permissions view of the Team Web Content click the **+ ADD PERMISSIONS** button.
- A new window opens. You can choose to set permissions for a number of resources.
  - Select the resource type **JOB TEMPLATES**
  - Choose the Insights Scan template by ticking the box next to it. You may need to click to the a different page in the selector until you get the Insights Scan template.
- The second part of the window opens, here you assign roles to the selected resource.
  - Choose **EXECUTE**
- Click **SAVE**

## Test Permissions

Now log out of Tower's web UI and in again as the **wwweb** user.

- Go to the **TEMPLATES** view, you should notice for Werner only the Insights Scan template is listed. He is allowed to view and launch, but not to edit the Template.
- Launch the Job Template. Click the rocket icon to the right of the job template in the Templates list (at bottom of screen, or from main Templates view)
- In the following **JOBS** view have a good look around, note in the play recap that the scan should have completed successfully, with each host reporting ok from the playbook output.

Check the result in the output window that appears:

Just recall what you have just done: You enabled a restricted user to run an Ansible Playbook

- Without having access to the credentials
- Without being able to change the Playbook itself
- But with the ability to change variables you predefined!
- This user can do NOTHING but launch the playbook we specified. Cannot see projects or inventories (unless you allow them to), you have granular control over access to these users.

Tip This capability is one of the main points of Ansible Tower!

## Lab 6: Build a Service Catalog with CloudForms

This lab will guide you through the process of creating a service catalog in CloudForms.

### Access the lab environment

To access the Red Hat CloudForms Management Engine use the URL and credentials below:

URL: <https://cf46-<GUID>.rhpds.opentlc.com>

User: admin

Password: r3dh4t1!

The ID <GUID> is unique to your lab environment and was presented to you on the browser start page! Replace the <GUID> with your lab specific value. For example, if your <GUID> would be "123a" the URL would become:

<https://cf46-123a.rhpds.opentlc.com>

**Note:** Your browser might give you a warning message about the used SSL Certificates. These warning messages can be accepted and are due to the fact that each lab deployed with new certificates on request.

### Value provided by a Service Catalog

One of the features a Cloud Management Platform provides, is a self service user interface. Here users can order, manage and retire services. Services are categorized in catalogs, where they can be organized and easily consumed.

By providing a service catalog, users can deploy the services they need quickly and simply. This will improve agility, reduce provisioning time and free up resources in internal IT.

### Service Basics

But first some basics. Four items are required to make a service available to users from the CloudForms self service portal:

1. A Provisioning Dialog which presents the basic configuration options for a virtual machine or instance.
2. A Service Dialog where you allow users to configure virtual machine or instance options.
3. A Service Catalog which is used to group Services Catalog Items together.
4. A Service Catalog Item (the actual Service) which combines a Service Dialog, a Provisioning Dialog and some additional meta data in the Service Catalog.

We can also use Role Based Access Control to make certain Service Catalog Items available to specific groups of users.

## Virtual Machine Provisioning example

The first example will guide you through the process of offering a Service Catalog Item to provision a simple virtual machine. This will include:

- Design a Service Dialog: a form which will ask the user for the necessary input data
- Create a Service Catalog: this will allow to organize services in a structured way
- Publish a Service Catalog Item: puts everything together and build the item which users can order

The previous chapter mentions a fourth object, the Provisioning Dialog. We do not have to create one, since there are examples shipped with the product, which does everything we need.

The following chapters will guide you through the process step-by-step.

## Build a VM Provisioning Service Dialog

For this example we will create a Service Dialog which will ask the user for two parameters:

- the name of the new virtual machine
- how much memory should be allocated to the new virtual machine

Follow these steps to design the service dialog:



1. Navigate to *Automation* -> *Automate* -> *Customization*

2. Navigate to *Service Dialogs* in the accordion on the left.

3. Click on *Configuration* -> *Add a new Dialog*
4. Chose a label and description:
5. *Dialog's name*: Simple VM
6. *Dialog's Description*: Simple VM provisioning dialog

*Note*: Do not try to save the changes right now! The dialog is not finished and you will receive an error message ("Validation failed: Dialog Simple VM must have at least one Tab")

7. Add a new text box by using drag and drop of the "Text Box" symbol

8. Click the pen icon to edit the text box

9. The first element will allow the user to specify a VM name. Modify the following fields:

10. The Label is the name of the element as it will be shown in the UI:

*Label:* VM Name

11. The name will be used for the internal variable of the provisioning workflow:

*Name:* option\_0\_vm\_name

12. The help is some text which will be shown if the mouse hovers over the little question mark icon next to the element. It can be used to provide additional information to the user to fill out this field:

*Help:* Specify the name of the new virtual machine

13. CloudForms allows us to design Service Dialogs comprised of many different types of Elements:

- Check box: allows to user to check or uncheck the element, often used to ask for additional optional data
- Date Control: allows the user to select a date from a calendar widget. Often used for retirement or other date related options
- Date/Time Control: same as Date Control, but also allows to specify a time, for example used to specify an automated shutdown or if a change should be scheduled for later
- Drop Down List: allows the user to select one or multiple options from a list, for example to choose from a list of available networks, applications, cost centers and many more
- Radio Button: Similar to the check box, but only one of the options can be selected, for example the base OS version (RHEL 6 or RHEL 7, but never more than one)
- Tag Control: a special element which allows the user to chose from available tags. More about tagging later in this lab
- Text Area Box: allows the user to enter relatively large amounts of text (multiple lines), could be used for example to provide description information

- Text Box: allows the user for short amounts of text (one line), in this example we use this element to ask the user for a name of the virtual machine

14. The remaining options can be ignored for now.

15. Click Save to save the Field Details

16. Add a new Dropdown field by using drag and drop of the "Dropdown" symbol

17. Click the pen icon to edit the dropdown field

18. Modify the following fields:
19. *Label*: Memory size
20. *Name*: option\_0\_vm\_memory
21. *Help*: Select how much memory the virtual machine should have
22. Switch to the *Options* tab of the dialog and add the following entries.
23. A element of type "Drop Down List" allows the user to select one of the predefined values. To create the list of selectable values, scroll down to the table "Entries" and add the following lines:

*Key*: 2048

*Value*: 2 GB

*Key*: 4096

*Value*: 4 GB

*Key*: 8192

*Value*: 8 GB

*Required*: Yes

*Value Type*: Integer

*Note*: To be able to add a line to the table, click on the little plus icon on the bottom of the list

24. Click **Save** to save the Field Details

*Note:* The memory values in the dialog editor will not reflect your changes immediately and will still show the original options One, Two, Three.

25. We are finally done designing the dialog. Click on **Save** to save the dialog.

26. *Note:* If you're having trouble creating the Service Dialog, you can download it from [GitHub](#) and import it. Follow the instructions on how to [import a service dialog](#) ONLY if you were unable to create the dialog.

## Build a VM Provisioning Service Catalog

The following steps will create a Service Catalog.

1. The next step is to create a Service Catalog. First we have to navigate to *Services -> Catalogs*.

2. Click on *Catalogs* in the accordion on the left



3. Click on *Configuration* and *Add a New Catalog*
4. Fill out name and description:

*Name:* Virtual Machines

*Description:* Deploy Virtual Machines from the Catalog

5. Click *Add* to save the Service Catalog

## **Build a Virtual Machine Service Catalog Item**

To tie everything together, the last step is to define a service catalog item.

1. Navigate to *Services* -> *Catalogs*

2. Click on *Catalog items* in the accordion on the left.
3. You should already see two Service Catalogs:
4. *Unassigned*: Catalog items which are not published yet, will be listed here
5. *Virtual Machines*: the Service Catalog we just created in the previous step

6. In the *Configuration* Menu, click on *Add a New Catalog Item*
7. Catalog Bundles are used for multi tier applications and consist of many Catalog Items. Since we do not have any existing Catalog Items, we can not create a Bundle.
8. Chose the Catalog Item Type. For this example we want to use the Red Hat Virtualization Provider, so click on *Red Hat Virtualization*

9. *Note:* It can take a few seconds for the next screen to load.

10. The next dialog will ask for the details of the new Service Catalog Item:
11. The name of the Service Catalog Item shown in the UI:

*Name:* Simple VM

12. A more descriptive text about the Service Catalog Item:

*Description:* A simple Linux Virtual Machine

13. Check the "Display in Catalog" box. If not selected, the Service Catalog Item will not be visible to users. This can be used for Items which are either still in draft mode, or should only be ordered as a part of a bundle:
14. *Display in Catalog:* check this box
15. Select the previously created Service Catalog:
16. *Catalog:* Virtual Machines
17. Select the previously created Service Dialog:
18. *Dialog:* Simple VM
19. All other fields on this tab can remain unchanged.

20. Entry Points are the hooks into CloudForms' powerful Automation Framework. It allows administrators to define provisioning, reconfiguration and retirement workflows which are different from the out of the box behavior. For example we could add integration into an IP Address Management Tool, a ticketing system or a CMDB Service. For this lab, we want to stick with the out of the box experience and leave those fields unchanged.

21. Click on the *Details* tab. You can provide some more descriptive explanation about the service here. We can even use basic HTML formatting in this box.

<h1>Simple VM</h1>

<p>When ordering this item, the user will be provided some simple questions to specify the hostname and memory size of the requested virtual machine.</p>

<p>The VM will be deployed with <a href=http://www.redhat.com>Red Hat Enterprise Linux 7</a>.

22. The *Request Info* tab of the dialog allows us to provide all the settings we want to use when provisioning a virtual machine from this Service Catalog Item.

23. Select the template used for provisioning:
24. *Selected VM*: RHEL7
25. For automatic naming chose "changeme"
26. *VM Name*: changeme
27. If no name is specified, this will cause CloudForms to automatically assign a name based on "cfme" as a prefix. The name will be expanded with a unique ID starting with 001.
28. Also make sure the *Provision Type* is "Native Clone" and do not change this value.
29. Click on the sub tab *Environment*
30. Although it sounds the most convenient option, we can not use "Choose Automatically". This will require the definition of a provisioning scope, which we haven't done yet. Instead we we set the appropriate values manually.
31. The datacenter of our Red Hat Virtualization Provider:
32. *Datacenter*: Default
33. The cluster in the Red Hat Virtualization Datacenter:
34. *Cluster*: Default
35. The host which will perform the actual tasks and where the VM will initially run on:
36. *Host*: rhvh1.example.com
37. The storage domain to store the VM
38. *Datastore*: data
39. Click on the next sub tab *Hardware*
40. For the purpose of the lab, the provided defaults are fine.
41. Click on the next sub tab *Network*
42. The lab environment is very simple, there is only one VLAN available:
43. *Network*: ovirtmgmt
44. Click on the next sub tab *Customize*
45. This allows to reconfigure certain settings inside the virtual machine. For the purpose of this lab, we keep them all empty
46. Click on the last sub tab *Schedule*
47. This allows us to delay provisioning to a later time, for example during the night or off hours. We can also set a retirement date. After notifying the user and allowing him or her to extend the lifespan of the virtual machine, retirement will shutdown and, by default, delete the virtual machine.
48. For the purpose of the lab, we keep these settings unchanged.
49. Finally click on *Add* to save the Catalog Item

## Order the Simple Virtual Machine Service Catalog Item

For sure you want to test the Service Catalog Item you just created!

1. Navigate to *Services* -> *Catalogs* and then click on *Service Catalogs* in the accordion on the left.

2. You should see the Service Catalog Item we just created:



3. Click on the Item to see more details.

4. Note that the Link for Red Hat Enterprise Linux in fact opens the Red Hat Homepage.
5. Click on *Order*
6. The Service Dialog we created earlier will be presented and ask for the name of the virtual machine and the memory size. As you can see, the name is a free text field, and the memory size is a drop down list.
7. *VM Name*: test
8. *Memory Size*: 2 GB
9. Set the default value of the dropdown to 2GB

10. Click *Submit* to order the Virtual Machine
11. You will be redirected to the request queue where you can see CloudForms working on your request.

12. *Note:* Since we are using nested virtualization to run these labs, performance will be slow and it can take several minutes to complete the request (20-30 minutes).

## Verify the order

In the requests queue you can click on *Reload* to see how CloudForms processes the order. If you click the button a few times, you should see the status is progressing.

While the VM is cloned from template, it does not show up in the CloudForms inventory. Watch the request queue and after the VM was cloned, check the result.

1. Navigate to *Compute -> Infrastructure -> Virtual Machines*

2. You should see a VM with the name "test" in the overview
3. *Note:* If you don't see the VM yet, it is probably still in creation. Check *Services -> Requests* to see the current status and reload the VM page periodically.

4. Click on the new VM "test" to see the VM details

5. This concludes this part of the lab.

## Lab 7: CloudForms Ansible Example

This lab will guide you through the process of creating a Service Catalog Item based on an Ansible Playbook.

### Introduction to Ansible

Today, every business is a digital business. Technology is your innovation engine, and delivering your applications faster helps you win. Historically, that required a lot of

manual effort and complicated coordination. But today, there is Ansible - the simple, yet powerful IT automation engine that thousands of companies are using to drive complexity out of their environments and accelerate DevOps initiatives.

Red Hat CloudForms can integrate with IaaS, PaaS, public and private cloud and configuration management providers. Since version 4.2 of CloudForms, it can also integrate with Ansible Tower by Red Hat. The latest version which is 4.6, which has an improved "embedded Ansible" role which allows it to run Playbooks, manage credentials and retrieve Playbooks from a source control management like git.

This integration allows customers to build service catalogs from Ansible Playbooks to allow end users to easily browse, order and manage resources from Ansible. Ansible Playbooks can be used in Control Policies which can not only detect problems, but also automatically fix them. The user interface of CloudForms can be extended seamlessly with additional menus and buttons, which utilize Ansible Playbooks to perform user initiated tasks.

## **Make sure embedded Ansible role is enabled and running**

Before we start, we want to make sure the embedded Ansible role is enabled and running.

1. Log into your CloudForms Appliance
2. Click on your user name on the top right and click on *Configuration*
3. Make sure the "Embedded Ansible" and the "Git Repositories Owner" Roles are enabled

4. Click on *Diagnostics* in the accordion on the left and click on the *Workers* tab
5. Make sure you can see a line indicating the "Embedded Ansible Worker" is in state "started"
6. *Note:* The git role is not represented by a specific worker process.

## Add a Git repository of Ansible Playbooks

To be able to run Ansible Playbooks, they have to become available in CloudForms. Custom git repositories can be used as well as GitHub, GitLab or others. Other Source Control Management Systems like Subversion or Mercurial are planned for later versions.

1. Navigate to Automation, Ansible, Repositories.

2. Click on *Configuration, Add New Repository*



3. *Note:* If the menu item "Add New Repository" is disabled, the Git Repository Role is not active.
4. Fill in the form.
5. *Name:* Github
6. *Description:* Example Playbooks
7. *URL:* <https://github.com/cbolz/summit-fy19.git>
8. *SCM Update Options:* check "Update on Launch"
9. Update on Launch causes CloudForms to check for new Playbooks are updated Playbooks before a Playbook is executed.

10. Click on *Add* to save the settings

*Note:* It takes a few seconds for the action to complete. A pop up notification will inform you after the task was completed.

## Store Virtual Machine Credentials

Ansible is using SSH by default to perform actions on the target machine. To be able to login, it has to know the login credentials.

1. Navigate to *Automation -> Ansible -> Credentials*

2. Click on *Configuration -> Add a new Credential*

3. Use the following settings:
4. *Name*: Virtual Machine credentials
5. *Credential type*: Machine
6. *Username*: root
7. *Password*: r3dh4t!

8. Click \*Add to save the credentials
9. Once more this is an action which is performed in the background and it can take a few seconds until you can see the new credentials in the Web UI.

## Create an Ansible Service Catalog

To offer a Service Catalog Item to users, they have to be organized in Service Catalogs. Create one by following these steps:

1. The next step is to create a service catalog. First we have to navigate to *Services* -> *Catalogs*.

2. On this screen click on *Catalogs* on the left
3. You should already see one Service Catalogs:
4. *Virtual Machines*: the Service Catalog we created in the Virtual Machine part of the lab

5. *Note:* You might already have some catalogs from previous labs.
6. Click on *Configuration* and *Add a New Catalog*
7. Fill out name and description:
8. *Name:* Ansible
9. *Description:* Order Ansible Playbooks from a Service Catalog

10. Click on *Add* to save the new Catalog

## Create a Service Catalog Item

In the following step we create a Service Catalog Item which will execute an Ansible Playbook.

1. Navigate to *Services -> Catalogs*

2. Navigate to *Catalog Items* in the accordion on the left

3. Click on *Configuration* -> *Add a New Catalog Item*

4. Select *Ansible Playbook* as Catalog Item Type



5. Use the following parameters when defining the Service Catalog Item:
6. *Name*: Install Package
7. *Description*: Install Package via Ansible Playbook
8. *Display in Catalog*: Yes
9. *Catalog*: Ansible
10. *Repository*: Github
11. *Playbook*: playbooks/InstallPackage.yml
12. *Machine Credentials*: Virtual Machine credentials
13. *Variables & Default Values*: add one new entry with:
14. *Variable*: package\_name
15. *Default Value*: httpd
16. Click the little plus ("+") icon to save the row.
17. *Dialog*: Create New
18. Use "InstallPackage" as the name of the Dialog.

19. Click *Add* to save all changes

## Test the Service Catalog Item

We want to make sure the resulting Service Catalog Item actually works.

1. Navigate to *Services -> Catalogs*

2. Click on *Service Catalogs* in the accordion on the left, if not already selected

3. Select the "Install Package" Service Catalog Item

4. Click *Order*
5. Select the following options:
6. *Machine Credentials*: Virtual Machine Credentials
7. *Hosts*: localhost (should already be the default)
8. *package\_name*: httpd (should already be the default)

9. Click on *Submit*
10. After submitting your order, you will be redirected to the Requests Queue. You should also see pop up notifications on the top right informing you about the progress of your order.
11. OPTIONAL: Click on *Refresh* to monitor the progress of your order
12. Navigate to *Services* -> *My Services*

13. Every time a user places an order a object under "My Services" gets created. You should see one tile labeled "Install Package"

14. Click on the tile icon to get more details

15. Click on the tab *Provisioning* to see details of the Ansible Playbook run

*Note:* In this example the Playbook completed successfully. In your case it might be still running and not be complete. Click the little reload icon on the page to reload the information while the Playbook is executed in the background.

16. This concludes this part of the lab.

THE END

We appreciate you giving the latest management portfolio labs a spin here at Summit 2018.  
Please rate your session presenters and the session content in your RH Summit Events App.