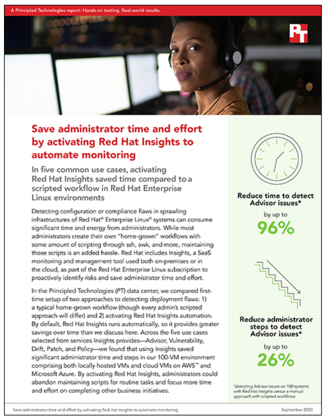

A Principled Technologies report: Hands-on testing. Real-world results.

Save administrator time and effort by activating Red Hat Insights to automate monitoring

In five common use cases, activating Red Hat Insights saved time compared to a scripted workflow in Red Hat Enterprise Linux environments

Detecting configuration or compliance flaws in sprawling infrastructures of Red Hat® Enterprise Linux® systems can consume significant time and energy from administrators. While most administrators create their own "home-grown" workflows with some amount of scripting through ssh, awk, and more, maintaining those scripts is an added hassle. Red Hat includes Insights, a SaaS monitoring and management tool used both on-premises or in the cloud, as part of the Red Hat Enterprise Linux subscription to proactively identify risks and save administrator time and effort.

In the Principled Technologies (PT) data center, we compared first-time setup of two approaches to detecting deployment flaws: 1) a typical home-grown workflow (though every admin's scripted approach will differ) and 2) activating Red Hat Insights automation. By default, Red Hat Insights runs automatically, so it provides greater savings over time than we discuss here. Across the five use cases selected from services Insights provides—Advisor, Vulnerability, Drift, Patch, and Policy—we found that using Insights saved significant administrator time and steps in our 100-VM environment comprising both locally hosted VMs and cloud VMs on AWS™ and Microsoft Azure. By activating Red Hat Insights, administrators could abandon maintaining scripts for routine tasks and focus more time and effort on completing other business initiatives.

Reduce time to detect Advisor issues* by up to 96%

Reduce administrator steps to detect Advisor issues* by up to 26%

*detecting Advisor issues on 100 systems with Red Hat Insights versus a manual approach with scripted workflows

Save administrator time and effort by activating Red Hat Insights to automate monitoring September 2020

The science behind the report:

Save administrator time and effort by activating Red Hat Insights to automate monitoring

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Save administrator time and effort by activating Red Hat Insights to automate monitoring.

We concluded our hands-on testing on July 31, 2020. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on July 9, 2020 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

Table 1: Advisor results

Method	Time (mm:ss)	Steps
Manual - 1 system	15:37	15
Manual - 50 systems	31:19	23
Manual - 100 systems	33:18	26
Insights - 100 systems	01:19	19
Time and steps saved - 100 systems	**31:59**	**7**

Table 2: Vulnerability results

Method	Time (mm:ss)	Steps
Manual - 1 system	03:34	9
Manual - 50 systems	09:49	17
Manual - 100 systems	15:45	23
Insights - 100 systems	01:24	10
Time and steps saved - 100 systems	**14:21**	**13**

Table 3: Drift results

Method	Time (mm:ss)	Steps
Manual - 1 system	25:37	49
Manual - 80 systems	32:14	31
Insights - 80 systems	03:21	19
Time and steps saved - 80 systems	**28:53**	**12**

Table 4: Patch results

Method	Time (mm:ss)	Steps
Manual - 1 system	02:45	7
Manual - 50 systems	10:23	15
Manual - 100 systems	19:47	20
Insights - 100 systems	02:11	17
Time and steps saved - 100 systems	**17:36**	**3**

Table 5: Policy results

Method	Time (mm:ss)	Steps
Manual - 1 system	00:44	4
Manual - 50 systems	08:54	12
Manual - 100 systems	12:00	15
Insights - 100 systems	03:37	23
Time and steps saved - 100 systems	**08:23**	**-8**

System configuration information

Table 6: Detailed information on the systems that hosted the local VMs for testing. For information about cloud VM configuration, see the How we tested section.

Server configuration information	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10
BIOS name and version	HPE U30 v2.02	HPE U30 v2.02	HPE U30 v2.34
Non-default BIOS settings	None	None	None
Operating system name and version/build number	VMware ESXi™ 7.0.0, 15843807	VMware ESXi 7.0.0, 15843807	VMware ESXi 7.0.0, 15843807
Date of last OS updates/patches applied	6/8/20	6/8/20	6/8/20
Power management policy	High Performance	High Performance	High Performance
Processor			
Number of processors	2	2	2
Vendor and model	Intel® Xeon® Platinum 8164	Intel Xeon Platinum 8164	Intel Xeon Platinum 8164
Core count (per processor)	26	26	26
Core frequency (GHz)	2.00	2.00	2.00
Stepping	H0	H0	H0
Memory module(s)			
Total memory in system (GB)	192	192	192
Number of memory modules	12	12	6
Vendor and model	Hynix Semiconductor HMA82GR7AFR8N-VK	Hynix Semiconductor HMA82GR7AFR8N-VK	Samsung® M393A4K40BB1-CRC
Size (GB)	16	16	32
Type	DDR4	DDR4	DDR4
Speed (MHz)	2,666	2,666	2,400
Speed running in the server (MHz)	2,666	2,666	2,400
Storage controller			
Vendor and model	HPE Smart Array P408i-a SR Gen10	HPE Smart Array P408i-a SR Gen10	HPE Smart Array P408i-a SR Gen10
Cache size (GB)	2	2	2
Firmware version	1.04	1.04	1.04
Driver version	2.0.50-1vmw.700.1.0.15843807	2.0.50-1vmw.700.1.0.15843807	2.0.50-1vmw.700.1.0.15843807
Local storage (type A)			
Number of drives	1	1	1
Drive vendor and model	Seagate® ST600MM0006	Seagate ST9600205SS	Seagate ST600MM0006
Drive size (GB)	600	600	600
Drive information (speed, interface, type)	10K, 6Gb SAS, HDD	10K, 6Gb SAS, HDD	10K, 6Gb SAS, HDD

Server configuration information	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10
Network adapter			
Vendor and model	HPE Ethernet 1Gb 4-port 331i Adapter - NIC	HPE Ethernet 1Gb 4-port 331i Adapter - NIC	HPE Ethernet 1Gb 4-port 331i Adapter - NIC
Number and type of ports	4 x 1GbE	4 x 1GbE	4 x 1GbE
Driver version	4.1.4.1-1vmw.700.1.0.15843807	4.1.4.1-1vmw.700.1.0.15843807	4.1.4.1-1vmw.700.1.0.15843807
Cooling fans			
Vendor and model	HPE 875075-001	HPE 875075-001	HPE 875075-001
Number of cooling fans	6	6	6
Power supplies			
Vendor and model	HPE 865408-B21	HPE 865408-B21	HPE 865408-B21
Number of power supplies	2	2	2
Wattage of each (W)	500	500	500

How we tested

*All ssh commands in this document are meant to be run by root. If EC2 and Azure instances do not permit root login, insert a sudo before all commands.

```
This format indicates text that should be executed via ssh at the command line or should be entered
to create a file in a vi editor.
```

Setting up the VM templates

1. In our local vSphere, set up a system named BaseVMTemplate with 1CPU, 2500MB of memory, 1 16GB HDD, and 1 network adapter connected. Perform a minimal install of RHEL 7.6. Set up a User account during setup.
2. Use ssh to log into the BaseVMTemplate system, and shut it down:

```
shutdown -h now
```

3. Create a clone of the BaseVMTemplate called AdvisorBaseVMTempate.
4. Power on AdvisorBaseVMTemplate.
5. Use ssh to log into the AdvisorBaseVMTemplate.
6. Install Microsoft SQL Server in the AdvisorBaseVMTemplate.
    a. Edit install_sql.sh to match this modified script provided by Red Hat:

```
#!/bin/bash -e

# Use the following variables to control your install:
# Password for the SA user (required)
MSSQL_SA_PASSWORD=Password1!

# Product ID of the version of SQL server you're installing.
# Must be evaluation, developer, express, web, standard, enterprise, or your 25 digit product key
MSSQL_PID='evaluation'

# Adding Microsoft repositories...
curl -o /etc/yum.repos.d/mssql-server.repo https://packages.microsoft.com/config/rhel/7/mssql-
server-2019.repo

#Update system cache
yum makecache

#Download and install SQL Server 2019...
yum install -y mssql-server

#Running mssql-confg setup...
MSSQL_SA_PASSWORD=$MSSQL_SA_PASSWORD \
MSSQL_PID=$MSSQL_PID \
/opt/mssql/bin/mssql-conf -n setup accept-eula

#Installing client tools
echo Installing mssql-tools and unixODBC developer...
curl -o /etc/yum.repos.d/msprod.repo https://packages.microsoft.com/config/rhel/7/prod.repo
ACCEPT_EULA=Y yum -y install mssql-tools unixODBC-devel

echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc

ln -s /usr/lib64/libssl.so.10 /opt/mssql/lib/libssl.so
ln -s /usr/lib64/libcrypto.so.10 /opt/mssql/lib/libcrypto.so

# Configure firewall to allow TCP port 1433:
echo Configuring firewall-rules to allow traffic on port 1433...
systemctl start firewalld
firewall-cmd --zone=public --add-port=1433/tcp --permanent
firewall-cmd --reload
```

```
# Restart SQL Server after installing:
echo Restarting SQL Server...
systemctl restart mssql-server
systemctl status mssql-server --no-pager

echo Done!
```

   b. Make the file executable:

```
chmod 0744 install_sql.sh
```

   c. Register system:

```
subscription-manager register --auto-attach
```

   d. Run the script to install Microsoft SQL:

```
./install_sql.sh
```

7. Create a clone of the BaseVMTemplate called ComplianceBaseVMTempate.
8. Power on ComplianceBaseVMTemplate.
9. Use ssh to log into the ComplianceBaseVMTemplate.
10. Register system:

```
subscription-manager register --auto-attach
```

11. Install open scap on ComplianceBaseVMTemplate:

```
yum install -y openscap-scanner scap-security-guide
```

Setting up an admin VM

This VM can be used to automate setup of Insights on all systems, store reports for each use case for aggregation, and in theory could be used to run any Insights-provided remediation playbooks. We will not be using Ansible to automate any manual test cases; this is just to properly set up Insights and take advantage of automation when setting up our test environment.

1. Create a clone of the ComplianceBaseVMTemplate called AdminVM.
2. Use ssh to log into system.
3. Install EPEL:

```
yum install -y wget
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -i epel-release-latest-7.noarch.rpm
```

4. Install oscap-ssh:

```
yum install openscap-utils
```

5. Update package list:

```
yum update
```

6. Install Ansible:

```
yum install -y ansible
```

7. Set up passwordless ssh in your environment:
   a. Generate an ssh key (when prompted, press enter to select the default destination for the key, and press enter two more times to have an empty passphrase):

```
ssh-keygen
```

   b. Create a Servers file with the IP addresses of all the local systems, and a Servers_Azure file with the IP addresses of all the Azure systems.
   c. Copy the key to all non-cloud test systems (when prompted, enter yes or the root password.):

```
username=root
readarray -t SERVERS < ~/Servers
for HOST in ${SERVERS[@]}; do ssh-copy-id -i ~/.ssh/id_rsa.pub ${username}@${HOST}; done
```

d. Validate ssh for the cloud systems:
   i. Create an AWS key pair at the EC2 console and save the private key to ~/AWSKeyPair.pem.
   ii. Validate the pem file setup works:

```
ssh-agent bash
ssh-add ~/AWSKeyPair.pem
```

   iii. Create an Azure key pair at the Azure console and save the private key to ~/AzureKey.pem.
   iv. Validate the pem file setup works:

```
chmod 0400 AzureKey.pem
ssh-agent bash
ssh-add ~/AzureKey.pem
```

8. Create an inventory.yml file as follows,using the correct IPs:

```
---
all:
children:
SUTs:
hosts:
RedHat001:
ansible_host: "10.211.2.23"
ansible_user: root
label: RedHat001
RedHat002:
ansible_host: "10.211.2.24"
ansible_user: root
label: RedHat002
RedHat003:
ansible_host: "10.211.2.25"
ansible_user: root
label: RedHat003
RedHat004:
ansible_host: "10.211.2.26"
ansible_user: root
label: RedHat004
RedHat005:
ansible_host: "10.211.2.28"
ansible_user: root
label: RedHat005
RedHat006:
ansible_host: "10.211.2.27"
ansible_user: root
label: RedHat006
RedHat007:
ansible_host: "10.211.2.29"
ansible_user: root
label: RedHat007
RedHat008:
ansible_host: "10.211.2.30"
ansible_user: root
label: RedHat008
RedHat009:
ansible_host: "10.211.2.31"
ansible_user: root
label: RedHat009
RedHat010:
ansible_host: "10.211.2.33"
ansible_user: root
label: RedHat010
RedHat011:
ansible_host: "10.211.2.112"
ansible_user: root
label: RedHat011
RedHat012:
```

```
ansible_host: "10.211.2.32"
ansible_user: root
label: RedHat012
RedHat013:
ansible_host: "10.211.2.35"
ansible_user: root
label: RedHat013
RedHat014:
ansible_host: "10.211.2.34"
ansible_user: root
label: RedHat014
RedHat015:
ansible_host: "10.211.2.36"
ansible_user: root
label: RedHat015
RedHat016:
ansible_host: "10.211.2.38"
ansible_user: root
label: RedHat016
RedHat017:
ansible_host: "10.211.2.37"
ansible_user: root
label: RedHat017
RedHat018:
ansible_host: "10.211.2.39"
ansible_user: root
label: RedHat018
RedHat019:
ansible_host: "10.211.2.40"
ansible_user: root
label: RedHat019
RedHat020:
ansible_host: "10.211.2.41"
ansible_user: root
label: RedHat020
RedHat021:
ansible_host: "10.211.2.42"
ansible_user: root
label: RedHat021
RedHat022:
ansible_host: "10.211.2.43"
ansible_user: root
label: RedHat022
RedHat023:
ansible_host: "10.211.2.44"
ansible_user: root
label: RedHat023
RedHat024:
ansible_host: "10.211.2.45"
ansible_user: root
label: RedHat024
RedHat025:
ansible_host: "10.211.2.49"
ansible_user: root
label: RedHat025
RedHat026:
ansible_host: "10.211.2.50"
ansible_user: root
label: RedHat026
RedHat027:
ansible_host: "10.211.2.113"
ansible_user: root
label: RedHat027
RedHat028:
ansible_host: "10.211.2.114"
```

```
ansible_user: root
label: RedHat028
RedHat029:
ansible_host: “10.211.2.52”
ansible_user: root
label: RedHat029
RedHat030:
ansible_host: “10.211.2.54”
ansible_user: root
label: RedHat030
RedHat031:
ansible_host: “10.211.2.55”
ansible_user: root
label: RedHat031
RedHat032:
ansible_host: “10.211.2.56”
ansible_user: root
label: RedHat032
RedHat033:
ansible_host: “10.211.2.57”
ansible_user: root
label: RedHat033
RedHat034:
ansible_host: “10.211.2.60”
ansible_user: root
label: RedHat034
RedHat035:
ansible_host: “10.211.2.58”
ansible_user: root
label: RedHat035
RedHat036:
ansible_host: “10.211.2.61”
ansible_user: root
label: RedHat036
RedHat037:
ansible_host: “10.211.2.62”
ansible_user: root
label: RedHat037
RedHat038:
ansible_host: “10.211.2.63”
ansible_user: root
label: RedHat038
RedHat039:
ansible_host: “10.211.2.115”
ansible_user: root
label: RedHat039
RedHat040:
ansible_host: “10.211.2.116”
ansible_user: root
label: RedHat040
RedHat041:
ansible_host: “10.211.2.65”
ansible_user: root
label: RedHat041
RedHat042:
ansible_host: “10.211.2.66”
ansible_user: root
label: RedHat042
RedHat043:
ansible_host: “10.211.2.68”
ansible_user: root
label: RedHat043
RedHat044:
ansible_host: “10.211.2.71”
ansible_user: root
```

```
label: RedHat044
RedHat045:
ansible_host: “10.211.2.72”
ansible_user: root
label: RedHat045
RedHat046:
ansible_host: “10.211.2.73”
ansible_user: root
label: RedHat046
RedHat047:
ansible_host: “10.211.2.74”
ansible_user: root
label: RedHat047
RedHat048:
ansible_host: “10.211.2.75”
ansible_user: root
label: RedHat048
RedHat049:
ansible_host: “10.211.2.76”
ansible_user: root
label: RedHat049
RedHat050:
ansible_host: “10.211.2.77”
ansible_user: root
label: RedHat050
RedHat051:
ansible_host: “10.211.2.117”
ansible_user: root
label: RedHat051
RedHat052:
ansible_host: “10.211.2.118”
ansible_user: root
label: RedHat052
RedHat053:
ansible_host: “10.211.2.78”
ansible_user: root
label: RedHat053
RedHat054:
ansible_host: “10.211.2.79”
ansible_user: root
label: RedHat054
RedHat055:
ansible_host: “10.211.2.80”
ansible_user: root
label: RedHat055
RedHat056:
ansible_host: “10.211.2.81”
ansible_user: root
label: RedHat056
RedHat057:
ansible_host: “10.211.2.82”
ansible_user: root
label: RedHat057
RedHat058:
ansible_host: “10.211.2.84”
ansible_user: root
label: RedHat058
RedHat059:
ansible_host: “10.211.2.83”
ansible_user: root
label: RedHat059
RedHat060:
ansible_host: “10.211.2.86”
ansible_user: root
label: RedHat060
```

```
RedHat061:
ansible_host: “10.211.2.121”
ansible_user: root
label: RedHat061
RedHat062:
ansible_host: “10.211.2.85”
ansible_user: root
label: RedHat062
RedHat063:
ansible_host: “10.211.2.87”
ansible_user: root
label: RedHat063
RedHat064:
ansible_host: “10.211.2.88”
ansible_user: root
label: RedHat064
RedHat065:
ansible_host: “10.211.2.89”
ansible_user: root
label: RedHat065
RedHat066:
ansible_host: “10.211.2.90”
ansible_user: root
label: RedHat066
RedHat067:
ansible_host: “10.211.2.91”
ansible_user: root
label: RedHat067
RedHat068:
ansible_host: “10.211.2.92”
ansible_user: root
label: RedHat068
RedHat069:
ansible_host: “10.211.2.120”
ansible_user: root
label: RedHat069
RedHat070:
ansible_host: “10.211.2.122”
ansible_user: root
label: RedHat070
RedHat071:
ansible_host: “10.211.2.93”
ansible_user: root
label: RedHat071
RedHat072:
ansible_host: “10.211.2.95”
ansible_user: root
label: RedHat072
RedHat073:
ansible_host: “10.211.2.96”
ansible_user: root
label: RedHat073
RedHat074:
ansible_host: “10.211.2.97”
ansible_user: root
label: RedHat074
RedHat075:
ansible_host: “10.211.2.98”
ansible_user: root
label: RedHat075
RedHat076:
ansible_host: “10.211.2.99”
ansible_user: root
label: RedHat076
RedHat077:
```

```
ansible_host: “10.211.2.124”
ansible_user: root
label: RedHat077
RedHat078:
ansible_host: “10.211.2.100”
ansible_user: root
label: RedHat078
RedHat079:
ansible_host: “10.211.2.101”
ansible_user: root
label: RedHat079
RedHat080:
ansible_host: “10.211.2.102”
ansible_user: root
label: RedHat080
RedHat081:
ansible_host: “10.211.2.103”
ansible_user: root
label: RedHat081
RedHat082:
ansible_host: “10.211.2.104”
ansible_user: root
label: RedHat082
RedHat083:
ansible_host: “10.211.2.105”
ansible_user: root
label: RedHat083
RedHat084:
ansible_host: “10.211.2.106”
ansible_user: root
label: RedHat084
RedHat085:
ansible_host: “10.211.2.123”
ansible_user: root
label: RedHat085
RedHat086:
ansible_host: “10.211.2.107”
ansible_user: root
label: RedHat086
RedHat087:
ansible_host: “10.211.2.109”
ansible_user: root
label: RedHat087
RedHat088:
ansible_host: “10.211.2.110”
ansible_user: root
label: RedHat088
RedHat089:
ansible_host: “10.211.2.125”
ansible_user: root
label: RedHat089
RedHat090:
ansible_host: “10.211.2.111”
ansible_user: root
label: RedHat090
RedHat091:
ansible_become_user: root
ansible_host: “0.0.0.0”
ansible_user: root
label: RedHat091
RedHat092:
ansible_become_user: root
ansible_host: “0.0.0.0”
ansible_user: root
label: RedHat092
```

```
RedHat093:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat093
RedHat094:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat094
RedHat095:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat095
RedHat096:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat096
RedHat097:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat097
RedHat098:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat098
RedHat099:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat099
RedHat100:
ansible_become_user: root
ansible_host: "0.0.0.0"
ansible_user: root
label: RedHat100

vars:
ansible_ssh_private_key_file: ~/.ssh/id_rsa
```

Cloning and/or creating the VMs for each test

1. Before doing each use case, delete all VMs that are not templates or revert them back to snapshots from when they were freshly installed.
2. Use the BaseVMTemplate or an alternate use-case specific image if one was created for the current use case, to create 90 clones of the VM, with hostnames RedHat001 through RedHat090, and distribute the VMs evenly between three VMware hosts (30 each).
3. Create five Red Hat Enterprise 7.6 VMs with hostnames RedHat091 through RedHat095 using the EC2 console to search the Public Images for RHEL-7.6 with Red Hat owner ID 309956199498. For the Advisory use case, use the t2.medium instance type. For all others, use the t2.micro instance type.
4. Create five Azure 7-LVM VMs with hostnames RedHat096 through RedHat100 on B2s instances for the MSSQL workload. For all others, use the Standard B1ls. (https://docs.microsoft.com/en-us/azure/virtual-machines/workloads/redhat/redhat-imagelist)
5. If alternate use-case-specific images are needed for the current use case, repeat the steps used to create the alternate templates on each cloud VM.
6. Take snapshots if desired before making use-case specific changes to the systems to simplify reverting VMs to the last snapshots for subsequent. This is useful for multiple use cases that use base install systems, but not the Advisory use case.

Automatically setting up all test systems for Insights

For the app-specific tunings use case:

The base VM for these use cases had to be registered with subscription manager to download and install the appropriate packages to get MSSQL or openscap installed. Then, we cloned these systems in VMware and gave them different hostnames, but did not clear out and rerun the subscription service. Consequently, these systems behave as if they already had a subscription even though they do not. You must redo the subscription-management before using the systems in Insights. To fix the subscription errors on these systems, complete the following steps:

1. SSH to the admin VM.
2. Log in.
3. Edit redosubs as follows:

```
subscription-manager remove --all
subscription-manager unregister
subscription-manager clean
subscription-manager register --username=<username> --password=<password>
subscription-manager refresh
subscription-manager attach –auto
```

4. Edit redosudosubs as follows:

```
sudo subscription-manager remove --all
sudo subscription-manager unregister
sudo subscription-manager clean
sudo subscription-manager register --username=<username> --password=<password>
sudo subscription-manager refresh
sudo subscription-manager attach –auto
```

5. Unsubscribe and resubscribe all the systems:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 } ‘ | head -90 > ~/Servers
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 } ‘ | head -95 | tail -n 5 > ~/Servers_EC2
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 } ‘ | tail -n 5 > ~/Servers_Azure
readarray -t SERVERS < ~/Servers
username=root
for HOST in ${SERVERS[@]}; do cat redosubs | ssh ${username}@${HOST}; done
readarray -t SERVERS < ~/Servers_EC2
username=ec2-user
for HOST in ${SERVERS[@]}; do cat redosudosubs | ssh ${username}@${HOST}; done
readarray -t SERVERS < ~/Servers_Azure
username=AzureUser
for HOST in ${SERVERS[@]}; do cat redosudosubs | ssh ${username}@${HOST}; done
```

For all use cases:

1. SSH to the admin VM.
2. Log in.
3. Install the ansible role:

```
ansible-galaxy install RedHatInsights.insights-client
```

4. Edit the `/root/.ansible/roles/RedHatInsights.insights-client/examples/example-insights-client-playbook.yml` file as follows:

```
---
hosts: all
roles:
{ role: RedHatInsights.insights-client, redhat_portal_username: ‘<your username>’ redhat_portal_
password: ‘<yourpassword>’ }
```

5. Run the ansible-playbook to install Insights:

```
ssh-agent bash
ssh-add ~/AzureKey.pem
ssh-add ~/AWSKeyPair.pem
ansible-playbook -vvv -i inventory.yml /root/.ansible/roles/RedHatInsights.insights-client/examples/
example-insights-client-playbook.yml
```

6. When the Ansible playbook doesn't execute correctly for all systems, fix the subscriptions again using steps 3-5 from previous section, and execute step 5 again.

Testing the use cases

Testing Advisor: App-specific tunings (SQL on RHEL)

Set up the machines to fail some Microsoft SQL checks

Create machines using the AdvisorBaseVMTemplate. Some default settings in a base minimal install are already not optimal, such as:

- Disk readaheads by default are 8192
- Auto numa balancing set to 0 by default on single CPU vm.
- Some sysctl disk settings
- The default sysctl virtual address settings are already not optimal.

Create a script to make these changes to some of the machines. These will change only the running parameters and will not be persistent, but should be enough to cause an error to be detected. Work with fresh VMs and do not conduct a reboot between VM setup and testing.

1. Change any optimal sysctl disk settings on RedHat001 through RedHat040.
   a. Edit Advisoryprep1.sh as follows:

```
sysctl -w kernel.sched_min_granularity_ns=50000000 \
kernel.sched_wakeup_granularity_ns=20000000 \
vm.dirty_background_ratio=20
```

   b. Execute it on 40 systems:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -40 > AdvisoryEdit1
username=root
readarray -t SERVERS < ~/AdvisoryEdit1
for HOST in ${SERVERS[@]}; do cat Advisoryprep1.sh | ssh ${username}@${HOST}; done
```

2. Disable the swap file on RedHat041 through RedHat080.
   a. Edit Advisoryprep2.sh as follows:

```
swapoff -a
```

   b. Execute it on 40 systems:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -80 | tail -n 40 > AdvisoryEdit2
readarray -t SERVERS < ~/AdvisoryEdit2
for HOST in ${SERVERS[@]}; do cat Advisoryprep2.sh | ssh ${username}@${HOST}; done
```

There will still be 10 local machines with default configuration parameters and 10 cloud VMs with the default cloud configurations.

Manual workflow – 1 system

1. Read through the best practices document provided by Microsoft: https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-performance-best-practices?view=sql-server-ver15
2. Connect to admin system with ssh.
3. Log in.
4. Check your disk readaheads, and set to 4096:

```
lsblk --list | grep "sd[a-z] " | awk '{print "/dev/"$1}' > /tmp/devices
readarray -t devices < /tmp/devices
for device in ${devices[@]}; do
blockdev --getra ${device} | grep -v "4096"
done
```

5. Check your sysctl disk settings:

```
diff <(sysctl kernel.sched_min_granularity_ns kernel.sched_wakeup_granularity_ns vm.dirty_ratio
vm.dirty_background_ratio vm.swappiness) <(echo -e "kernel.sched_min_granularity_ns = 10000000\
nkernel.sched_wakeup_granularity_ns = 15000000\nvm.dirty_ratio = 40\nvm.dirty_background_ratio = 10\
nvm.swappiness = 10")
```

6. Check your sysctl numa balancing setting:

```
diff <(sysctl kernel.numa_balancing) <(echo kernel.numa_balancing = 0)
```

7. Check your sysctl virtual address space setting:

```
diff <(sysctl vm.max_map_count) <(echo vm.max_map_count = 262144)
```

8. Check your swap file utilization:

```
free -m | grep Swap | awk '{ print ($2) ? $0 ", ratio:  " $3/$2 : "Swap is off" }'
```

9. Prepare the fixes:
   a. Save a script for later that sets disk readaheads to 4096.
      Use a runtime fix:

```
lsblk --list | grep "sd[a-z] " | awk '{print "/dev/"$1}' > /tmp/devices
readarray -t devices < /tmp/devices
for device in ${devices[@]}; do
blockdev --setra 4096 ${device}
done
```

   b. Prepare a tuned.conf to set your sysctl disk settings and a script to use later to enable it:

```
vi ~/tuned.conf
```

   c. Edit the file as follows:

```
#
# A tuned configuration for SQL Server on Linux
#

[main]
summary=Optimize for Microsoft SQL Server
include=throughput-performance

[cpu]
force_latency=5

[sysctl]
vm.swappiness = 1
vm.dirty_background_ratio = 3
vm.dirty_ratio = 80
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
vm.transparent_hugepages=always
# For , use
# vm.transparent_hugepages=madvice
vm.max_map_count=1600000
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
kernel.numa_balancing=0
kernel.sched_latency_ns = 60000000
kernel.sched_migration_cost_ns = 500000
kernel.sched_min_granularity_ns = 15000000
kernel.sched_wakeup_granularity_ns = 2000000
```

d. Create a script to run later that will install the tuned-profiles-mssql package and enable tuned.conf:

```
subscription-manager repos –enable=rhel-7-server-optional-rpms
yum install -y tuned-profiles-mssql
cp ~/tuned.conf /usr/lib/tuned/mssql/tuned.conf
chmod +x /usr/lib/tuned/mssql/tuned.conf
tuned-adm profile mssql
```

e. Prepare a script for later to extend your swap file utilization:

```
swapoff -a
lvextend -L +2G /dev/mapper/rhel_<hostname>-swap
mkswap /dev/mapper/rhel_<hostname>-swap
swapon -a
```

Manual workflow - Scaling up to multiple systems

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for advisory reports:

```
mkdir ~/Advisory
```

4. Edit the ~/Advisory_servers file to contain all local servers you want to check the MSSQL tuning status for:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | head -90 > ~/Advisory_servers
```

5. Edit the ~/Advisory_servers_EC2 file to contain all AWS systems you want to check the MSSQL tuning status for:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | head -95 | tail -n 5 > ~/Advisory_
servers_EC2
```

6. Edit the ~/Advisory_servers_Azure file to contain all Azure systems you want to check the MSSQL tuning status for:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | tail -n 5 > ~/Advisory_servers_Azure
```

7. Edit the Advisory.sh file as follows:

```
Advisory_filename=Advisory_”$(hostname)”_$(date +”%Y-%m-%d”)
touch “/tmp/${Advisory_filename}”
lsblk --list | grep “sd[a-z] “ | awk ‘{print “/dev/”$1}’ > /tmp/devices
readarray -t devices < /tmp/devices
for device in ${devices[@]}; do
blockdev --getra ${device} | grep -v “4096” >> “/tmp/${Advisory_filename}”
done
diff <(sysctl kernel.sched_min_granularity_ns kernel.sched_wakeup_granularity_ns vm.dirty_ratio
vm.dirty_background_ratio vm.swappiness) <(echo -e “kernel.sched_min_granularity_ns = 10000000\
nkernel.sched_wakeup_granularity_ns = 15000000\nvm.dirty_ratio = 40\nvm.dirty_background_ratio = 10\
nvm.swappiness = 10”) >> “/tmp/${Advisory_filename}”
diff <(sysctl kernel.numa_balancing) <(echo kernel.numa_balancing = 0) >> “/tmp/${Advisory_filename}”
diff <(sysctl vm.max_map_count) <(echo vm.max_map_count = 262144) >> “/tmp/${Advisory_filename}”
free -m | grep Swap | awk ‘{ print ($2) ? $0 “, ratio:  “ $3/$2 : “Swap is off” }’ >> “/
tmp/${Advisory_filename}”
```

8. Edit the ALL-Advisory.sh to look like this:

```
username=root
readarray -t SERVERS < ~/Advisory_servers
for HOST in ${SERVERS[@]}; do
cat Advisory.sh | ssh ${username}@${HOST}
mydate=`date +”%Y-%m-%d”`
scp ${username}@${HOST}:/tmp/Advisory_*_$mydate_* ~/Advisory/Advisory_${HOST}_${mydate}
touch ~/Advisory/Advisory_summary.txt
cat ~/Advisory/Advisory_${HOST}_${mydate} | awk -v h=$HOST ‘{print h”: “$0} ‘ >> ~/Advisory/Advisory_
summary.txt
done
username=ec2-user
readarray -t SERVERS < ~/Advisory_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Advisory.sh | ssh ${username}@${HOST}
mydate=`date +”%Y-%m-%d”`
```

```
scp ${username}@${HOST}:/tmp/Advisory_*_$mydate_* ~/Advisory/Advisory_${HOST}_${mydate}
touch ~/Advisory/Advisory_summary.txt
cat ~/Advisory/Advisory_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Advisory/Advisory_
summary.txt
done
username=AzureUser
readarray -t SERVERS < ~/Advisory_servers_Azure
for HOST in ${SERVERS[@]}; do
cat Advisory.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/Advisory_*_$mydate_* ~/Advisory/Advisory_${HOST}_${mydate}
touch ~/Advisory/Advisory_summary.txt
cat ~/Advisory/Advisory_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Advisory/Advisory_
summary.txt
done
```

9. Make ALL-Advisory.sh executable:

```
chmod 0744 ALL-Advisory.sh
```

10. Execute the looping script:

```
ssh-agent bash
ssh-add ~/AzureKey.pem
ssh-add ~/AWSKeyPair.pem
./ALL-Advisory.sh
```

11. View the status of all the systems:

```
More Advisory/Advisory_summary.txt
```

Insights workflow

1. The Insights schedule had already gathered information from the system after Microsoft SQL was installed, so it it not necessary to run Insights again. (insights-client)
2. Go to cloud.redhat.com.
3. Log in.
4. On the Red Hat Insights panel, click Advisor.
5. On the left side of the screen, under Advisor, click Topics.
6. Click Microsoft SQL Server.
7. View a list of Microsoft SQL Server Recommendations and how many systems each applies to.
8. Save the ansible script to remediate each issue:
    a. Click on the issue.
    b. Underneath Affected systems, to the left of the down arrow, click the box to select all systems.
    c. Click the blue Remediate button.
    d. To the left Create new Playbook, click the radio button.
    e. To the right of Create new Playbook, enter MSSQLremediation.
    f. At the bottom left, click Next.
    g. At the bottom left, click Create. Your remediation playbook is saved for later.

Testing Vulnerability (CVE)

For testing purposes, we chose one important and recent CVE for investigation: CVE-2020-8616 from 18 May 2020.

Create machines using the BaseVMTemplate. Precede the hostnames with V, i.e., VRedHat001

Manual workflow– 1 system

1. Connect to system with ssh.
2. Log in.
3. Check the system for available updates, and notice that there is a RHSA-2020:2344 for CVE-2020-8616 and CVE-2020-8617.

```
yum updateinfo info cve | grep "CVEs"
```

4. Get more information:
   a. Get more information from the command prompt (which also confirms that this CVE applies to this system):

```
yum updateinfo --cve CVE-2020-8616 --cve CVE-2020-8617
```

   b. Visit https://access.redhat.com/security/security-updates/#/cve
   c. To the left of the Go button, enter CVE-2020-8616.
   d. Click Go.
   e. Click on the CVE.
   f. Read through the Description and Mitigation.
   g. Look to see which package needs updating (bind).
   h. Save the update and instructions for later.

Manual workflow – Scaling up to multiple systems

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Vulnerability reports:

```
mkdir ~/Vulnerability
```

4. Edit the ~/Vulernability_servers file to contain all local servers you want to check the vulnerability on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 > ~/Vulnerability_servers
```

5. Edit the ~/Vulnerability_servers_EC2 file to contain all AWS systems you want to check the vulnerability on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -95 | tail -n 5 > ~/Vulnerability_
servers_EC2
```

6. Edit the ~/Vulnerability_servers_Azure file to contain all Azure systems you want to check the vulnerability on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | tail -n 5 > ~/Vulnerability_servers_Azure
```

7. Edit the Vulnerability.sh file as follows:

```
Vulnerability_filename=SysVulnerability_"$(hostname)"_$(date +"%Y-%m-%d")
touch "/tmp/${Vulnerability_filename}"
if [ "$(whoami)" != "root" ]; then
sudo yum updateinfo --cve CVE-2020-8616 --cve CVE-2020-8617 | grep "Important Security notice" >> "/
tmp/${Vulnerability_filename}"
else
yum updateinfo --cve CVE-2020-8616 --cve CVE-2020-8617 | grep "Important Security notice" >> "/
tmp/${Vulnerability_filename}"
fi
```

8. Edit the ALL-Vulnerability.sh as follows:

```
username=root
readarray -t SERVERS < ~/Vulnerability_servers
for HOST in ${SERVERS[@]}; do
cat Vulnerability.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/SysVulnerability_*_$mydate_* ~/Vulnerability/
Vulnerability_${HOST}_${mydate}
cat ~/Vulnerability/Vulnerability_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/
Vulnerability/Vulnerability_summary.txt
```

```
done
username=ec2-user
readarray -t SERVERS < ~/Vulnerability_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Vulnerability.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/SysVulnerability_*_$mydate_* ~/Vulnerability/
Vulnerability_${HOST}_${mydate}
cat ~/Vulnerability/Vulnerability_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/
Vulnerability/Vulnerability_summary.txt
done
username=AzureUser
readarray -t SERVERS < ~/Vulnerability_servers_Azure
for HOST in ${SERVERS[@]}; do
cat Vulnerability.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/SysVulnerability_*_$mydate_* ~/Vulnerability/
Vulnerability_${HOST}_${mydate}
cat ~/Vulnerability/Vulnerability_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/
Vulnerability/Vulnerability_summary.txt
done
```

9. Make ALL-Vulnerability.sh executable:

   ```
   chmod 0744 ALL-Vulnerability.sh
   ```

10. Execute the looping script:

    ```
    ssh-agent bash
    ssh-add ~/AzureKey.pem
    ssh-add ~/AWSKeyPair.pem
    ./ALL-Vulnerability.sh
    ```

11. View the status of all the systems:

    ```
    more Vulnerability/Vulnerability_summary.txt
    ls Vulnerability
    ```

Insights workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Vulnerability.
4. Click the Systems tab.
5. Click on the first system.
6. Notice that there is a CVE-2020-8616.
7. Gather information on the CVE.
   a. Click on the CVE.
   b. Read through the text.
   c. Scroll down to see a list of exposed systems.
   d. Underneath Exposed systems, to the left of the down arrow, click the box to select all systems.
   e. Click the blue Remediate button.
   f. To the left of Create new Playbook, click the radio button.
   g. To the right of Create new Playbook, enter CVE-2020-8616remediation.
   h. At the bottom left, click Next.
   i. At the bottom left, click Create. Your remediation playbook is saved for later.

Testing Drift (baseline and change)

Use one system as the baseline VM to compare the other VMs to. To do the most comparisons while still have the comparisons make sense, we recommend using RedHat001, a local VM, for the baseline, and then making changes to each of the 90 local VMs. These changes should be detected as drift.

Create machines using the BaseVMTemplate. Precede the hostnames with V, i.e., VRedHat001.

Manual workflow – Creating a Baseline

1. Connect to RedHat001 system with ssh.
2. Log in.
3. Check the cloud_provider:

```
baselinefilename=baseline_"$(hostname)"_$(date +"%Y-%m-%d")
bios=`dmidecode -s bios-version`
if [[ $bios = *mazon* ]]; then echo cloud_provider=Amazon >> "/tmp/${baselinefilename}"
elif [[ $bios = *zure*  ]]; then echo cloud_provider=Azure >> "/tmp/${baselinefilename}"
else echo cloud_provider=N/A >> "/tmp/${baselinefilename}"; fi
```

4. Check the architecture:

```
echo arch=`uname -m` >> "/tmp/${baselinefilename}"
```

5. Check the bios_release_date:

```
echo bios_release_date=`dmidecode -s bios-release-date` >> "/tmp/${baselinefilename}"
```

6. Check the bios_vendor:

```
echo bios_vendor=`dmidecode -s system-manufacturer` >> "/tmp/${baselinefilename}"
```

7. Check the bios_version:

```
echo bios_version=`dmidecode -s bios-version` >> "/tmp/${baselinefilename}"
```

8. Check the cpu_information:

```
echo cpu_info=`lscpu` >> "/tmp/${baselinefilename}"
```

9. Check the enabled_services:

```
echo enabled_services=`systemctl list-unit-files` >> "/tmp/${baselinefilename}"
```

10. Check the infrastructure_type:

```
echo infrastructure_type=`dmidecode -s system-product-name` >> "/tmp/${baselinefilename}"
```

11. Check the infrastructure_vendor:

```
echo infrastructure_vendor=`dmidecode -s system-manufacturer` >> "/tmp/${baselinefilename}"
```

12. Check the installed_packages:

```
echo installed_packages=`rpm -qa --last` >> "/tmp/${baselinefilename}"
```

13. Check the kernel_modules:

```
echo kernel_models=`lsmod` >> "/tmp/${baselinefilename}"
```

14. Check the os_kernel_version:

```
echo os_kernel_version=`uname -r` >> "/tmp/${baselinefilename}"
```

15. Check the os_release:

```
echo os_release=`cat /etc/redhat-release` >> "/tmp/${baselinefilename}"
```

16. Check the running_processes:

```
echo running_processes=`ps aux | awk '{print $11}'` >> "/tmp/${baselinefilename}"
```

17. Check the Subscriptions:

```
echo subscriptions=`subscription-manager status` >> "/tmp/${baselinefilename}"
```

18. Check the system_memory:

```
echo system_memory=`free -h | grep Mem | awk '{print $2}'` >> "/tmp/${baselinefilename}"
```

19. Check the yum_repos:

```
echo yum_repos=`yum repolist` >> "/tmp/${baselinefilename}"
```

20. Copy the file to a central location (Admin server):

```
ssh root@<admin server> 'if [ ! -f "~/Drift" ]; then mkdir ~/Drift; fi'
scp "/tmp/${baselinefilename}" "root@<admin system ip>:~/Drift/${baselinefilename}"
```

Making some changes to the systems

1. Run updates on RedHat001 through RedHat030.
   a. ssh to the admin system.
   b. Log in.
   c. Edit Patchprep1.sh as follows:

```
sudo yum update -y
```

   d. Run the updates:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -30 > PatchEdit1
username=root
readarray -t SERVERS < ~/PatchEdit1
for HOST in ${SERVERS[@]}; do cat Patchprep1.sh | ssh ${username}@${HOST}; done
```

2. Install extra packages on RedHat031 through RedHat060.
   a. Edit Patchprep2.sh as follows:

```
sudo yum install python3 -y
```

   b. Run the updates:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -60 | tail -n 30 > PatchEdit2
username=root
readarray -t SERVERS < ~/PatchEdit2
for HOST in ${SERVERS[@]}; do cat Patchprep2.sh | ssh ${username}@${HOST}; done
```

3. Load a new kernel module on RedHat061 through RedHat090.
   a. Edit Patchprep3.sh to look like this:

```
modprobe wacom
```

   b. Run the updates:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 | tail -n 30 > PatchEdit3
username=root
readarray -t SERVERS < ~/PatchEdit3
for HOST in ${SERVERS[@]}; do cat Patchprep3.sh | ssh ${username}@${HOST}; done
```

Manually comparing the current configuration against a baseline – 1 system

1. Connect to RedHat001 system with ssh.
2. Log in.
3. Run the same commands from creating a baseline, output it to a new file and copy the file to the Admin system:

```
ssh root@<admin server> 'if [ ! -f "~/Drift" ]; then mkdir ~/Drift; fi'
scp "/tmp/${currentfilename}" "root@<admin system ip>:~/Drift/${currentfilename}"
```

4. Connect to Admin system with ssh.
5. Log in.
6. Copy the new file to the Admin system.
7. Compare the two files:

```
baselinefilename=<baseline file name>
currentfilename=<new file name>
diff "Drift/${baselinefilename}" "Drift/${currentfilename}"
```

Manually comparing the current configuration - Scaling up to multiple systems

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Drift reports, if necessary:

```
if [ ! -f "~/Drift" ]; then mkdir ~/Drift; fi
```

4. Edit the ~/Drift_servers file to contain all local servers you want to check the drift on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 > ~/Drift_servers
```

5. Edit the Drift.sh file as follows:

```
myfilename=driftcompare_"$(hostname)"_$(date +"%Y-%m-%d")
bios=`sudo dmidecode -s bios-version`;
if [[ $bios = *mazon* ]]; then echo cloud_provider=Amazon >> "/tmp/${myfilename}"
elif [[ $bios = *zure*  ]]; then echo cloud_provider=Azure >> "/tmp/${myfilename}"
else echo cloud_provider=N/A >> "/tmp/${myfilename}"; fi
echo arch=`uname -m` >> "/tmp/${myfilename}"
echo bios_release_date=`dmidecode -s bios-release-date` >> "/tmp/${myfilename}"
echo bios_vendor=`dmidecode -s system-manufacturer` >> "/tmp/${myfilename}"
echo bios_version=`dmidecode -s bios-version` >> "/tmp/${myfilename}"
echo cpu_info=`lscpu` >> "/tmp/${myfilename}"
echo enabled_services=`systemctl list-unit-files` >> "/tmp/${myfilename}"
echo infrastructure_type=`dmidecode -s system-product-name` >> "/tmp/${myfilename}"
echo infrastructure_vendor=`dmidecode -s system-manufacturer` >> "/tmp/${myfilename}"
echo installed_packages=`rpm -qa --last` >> "/tmp/${myfilename}"
echo kernel_models=`lsmod` >> "/tmp/${myfilename}"
echo os_kernel_version=`uname -r` >> "/tmp/${myfilename}"
echo os_release=`cat /etc/redhat-release` >> "/tmp/${myfilename}"
echo running_processes=`ps aux | awk '{print $11}'` >> "/tmp/${myfilename}"
echo subscriptions=`subscription-manager status` >> "/tmp/${myfilename}"
echo system_memory=`free -h | grep Mem | awk '{print $2}'` >> "/tmp/${myfilename}"
echo yum_repos=`yum repolist` >> "/tmp/${myfilename}"
```

6. Edit the ALL-Drift.sh to look like this so it will compare all systems to the baseline we created earlier:

```
username=root
readarray -t SERVERS < ~/Drift_servers
for HOST in ${SERVERS[@]}; do
cat Drift.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/driftcompare* ~/Drift/driftcompare_${HOST}_${mydate}
touch ~/Drift/Drift_summary.txt
diff Drift/driftcompare_${HOST}_${mydate} Drift/baseline* | awk -v h=$HOST '{print h": "$0} ' >> ~/
Drift/Drift_summary.txt
done
```

7. Make ALL-Drift.sh executable:

```
chmod 0744 ALL-Drift.sh
```

8. Execute the looping script:

```
./ALL-Drift.sh
```

9. View the status of all the systems, saving the information for action later:

```
cat Drift/Drift_summary.txt
```

Insights workflow – Creating a Baseline

Before performing this test, delete the VMs from the manual test case and recreate them according to the procedures from the Clone and/or create the VMs for each test section of this document.

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Drift.
4. On the left side of the screen, under Drift, click Baselines.
5. Click Create baseline.
6. To the left of Copy an existing system, click the radio button.
7. Underneath the Baseline name text, type the hostname and date in the textbox.
8. To the left of the name of the system desired, click the checkbox to select it as the baseline system.
9. Click Create baseline.
10. To the right of last_boot_time, click the column of three dots, and click Delete fact.
11. To the right of fqdn, click the column of three dots, and click Delete fact.
12. Click Delete facts.
13. To the left of network interfaces, click the column of three dots, and click Delete category.

Insights workflow – Making some changes to the systems

The same changes should be made to the same systems as we did in the manual use case.

Insights workflow – Comparing the current configurations against a baseline

1. Manually start a scan on all the systems:
    a. Connect to the Admin system with ssh.
    b. Log in.
    c. Edit the ~/Drift_Scan_servers file to contain all local servers you want to check the Drift on:

    ```
    grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 > ~/Drift_Scan_servers
    ```

    d. Edit the Drift_Scan.sh file as follows:

    ```
    insights-client
    ```

    e. Edit the ALL_Driftscan.sh as follows:

    ```
    username=root
    readarray -t SERVERS < ~/Drift_Scan_servers
    for HOST in ${SERVERS[@]}; do
    cat Drift_Scan.sh | ssh ${username}@${HOST}
    done
    ```

    f. Make ALL-DriftScan.sh executable:

    ```
    chmod 0744 ALL-DriftScan.sh
    ```

    g. Execute the looping script to trigger an update in Insights for all the systems:

    ```
    ssh-agent bash
    ./ALL-DriftScan.sh
    ```

2. Go to cloud.redhat.com.
3. Log in.
4. On the Red Hat Insights panel, click Drift.
5. Click Add to comparison.
6. To the left of the Name column header, click the box to select all systems.
7. Click the Baselines tab.
8. To the left of the name of the baselines desired, click the checkbox to select it as the baseline to compare to.
9. Click Submit

Testing Patch (patches)

For testing purposes, we chose one recent bug fix that needed investigation: RHBA-2020:2355 from 02 June 2020.

Create machines using the BaseVMTemplate. Precede the hostnames with V, i.e., VRedHat001.

Manual workflow – 1 system

1. Connect to system with ssh.
2. Log in.
3. Check the system for available kernel updates, and notice that there is a RHBA-2020:2355:

```
yum updateinfo bugfix | grep kernel
```

4. Get more information:
   a. Get more information from the command prompt:

```
yum updateinfo info --advisory RHBA-2020:2355
```

   b. Visit https://access.redhat.com/errata/#/
   c. Under Advisory Type, click Bug Fix.
   d. To the left of the Go button, enter `2020-2355`
   e. Click Go.
   f. Click the RHBA.
   g. Read through the advisory.
   h. Save the update and instructions for later.

Manual workflow – Scaling up to multiple systems

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Patch reports:

```
mkdir ~/Patch
```

4. Edit the ~/Patch_servers file to contain all local servers you want to check the patches on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 > ~/Patch_servers
```

5. Edit the ~/Patch_servers_EC2 file to contain all AWS systems you want to check the patches on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -95 | tail -n 5 > ~/Patch_servers_
EC2
```

6. Edit the ~/Patch_servers_Azure file to contain all Azure systems you want to check the patches on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | tail -n 5 > ~/Patch_servers_Azure
```

7. Edit the Patch.sh file as follows:

```
myfilename=patchesneeded_"$(hostname)"_$(date +"%Y-%m-%d")
if [ "$(whoami)" != "root" ]; then
sudo yum updateinfo list --advisory RHBA-2020:2355 | grep -v "Loaded plugins" | grep -v "updateinfo
list done" > "/tmp/${myfilename}"
else
yum updateinfo list --advisory RHBA-2020:2355 | grep -v "Loaded plugins" | grep -v "updateinfo list
done" | tail -n 4 > "/tmp/${myfilename}"
fi
```

8. Edit the ALL-Patch.sh as follows:

```
username=root
readarray -t SERVERS < ~/Patch_servers
for HOST in ${SERVERS[@]}; do
cat Patch.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/patchesneeded_*_$mydate ~/Patch/patchesneeded_${HOST}_${mydate}
cat ~/Patch/patchesneeded_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Patch/Patch_
summary.txt
done
username=ec2-user
```

```
readarray -t SERVERS < ~/Patch_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Patch.sh | ssh ${username}@${HOST}
mydate=`date +”%Y-%m-%d”`
scp ${username}@${HOST}:/tmp/patchesneeded_*_$mydate ~/Patch/patchesneeded_${HOST}_${mydate}
cat ~/Patch/patchesneeded_${HOST}_${mydate} | awk -v h=$HOST ‘{print h”: “$0} ‘ >> ~/Patch/Patch_
summary.txt
done
username=AzureUser
readarray -t SERVERS < ~/Patch_servers_Azure
for HOST in ${SERVERS[@]}; do
cat Patch.sh | ssh ${username}@${HOST}
mydate=`date +”%Y-%m-%d”`
scp ${username}@${HOST}:/tmp/patchesneeded_*_$mydate ~/Patch/patchesneeded_${HOST}_${mydate}
cat ~/Patch/patchesneeded_${HOST}_${mydate} | awk -v h=$HOST ‘{print h”: “$0} ‘ >> ~/Patch/Patch_
summary.txt
done
```

9. Make ALL-Patch.sh executable:

```
chmod 0744 ALL-Patch.sh
```

10. Execute the looping script:

```
ssh-agent bash
ssh-add ~/AzureKey.pem
ssh-add ~/AWSKeyPair.pem
./ALL-Patch.sh
```

11. View the status of all the systems:

```
more Patch/Patch_summary.txt
```

Insights workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Patch.
4. Notice that there is a RHBA-2020:2355.
5. Gather information on the bugfix:
    a. Click on the bugfix.
    b. Read through the text.
    c. Scroll down to see a list of Affected systems.
    d. Save the update and instructions for later.

Testing Policy (firewall)

Create machines using the BaseVMTemplate. Precede the hostnames with V, i.e., VRedHat001.

Setting up some machines to fail checks

1. Disable firewall on VRedHat001 through VRedhat040
   a. Edit Policyprep1.sh as follows:

```
systemctl stop firewalld.service
systemctl disable firewalld
```

   b. Run the updates:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -40 > PolicyEdit1
username=root
readarray -t SERVERS < ~/PolicyEdit1
for HOST in ${SERVERS[@]}; do cat Policyprep1.sh | ssh ${username}@${HOST}; done
```

2. Uninstall openscap on VRedHat041 through VRedHat080.
   a. Edit Policyprep2.sh as follows:

```
yum remove openscap-scanner
```

   b. Run the updates:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -80 | tail -n 40 > PolicyEdit2
username=root
readarray -t SERVERS < ~/PolicyEdit2
for HOST in ${SERVERS[@]}; do cat Policyprep2.sh | ssh ${username}@${HOST}; done
```

   c. Leave the AWS VMs and Azure VMs as is, they are already heterogeneous.

Manual workflow – 1 system

1. Use ssh to log into system.
2. Check firewall:

```
systemctl status firewalld | grep Active
firewall-cmd --state
```

3. Check for openscap package:

```
yum list installed | grep openscap-scanner
```

4. Prepare the fixes.
   a. Save a script for later that enables your firewall:

```
systemctl start firewalld
systemctl enable firewalld
```

   b. Save a script for later that installs openscap:

```
yum install openscap-scanner
```

Manual workflow – Scaling up to multiple systems

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Policy reports:

```
mkdir ~/Policy
```

4. Edit the ~/Policy_servers file to contain all local servers you want to check the firewall and openscap status on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -90 > ~/Policy_servers
```

5. Edit the ~/Policy_servers_EC2 file to contain all AWS systems you want check the firewall and openscap status on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | head -95 | tail -n 5 > ~/Policy_servers_
EC2
```

6. Edit the ~/Policy_servers_Azure file to contain all Azure systems you want to check the firewall and openscap status on:

```
grep ansible_host inventory.yml | awk -F\" '{ print $2 }' | tail -n 5 > ~/Policy_servers_Azure
```

7. Edit the Policy.sh file as follows:

```
policyfilename=policy_"$(hostname)"_$(date +"%Y-%m-%d")
touch "/tmp/${policyfilename}"
systemctl status firewalld | grep Active >> "/tmp/${policyfilename}"
if [ "$(whoami)" != "root" ]; then
sudo firewall-cmd --state >> "/tmp/${policyfilename}"
sudo openscapinstalled=`yum list installed | grep openscap-scanner`
else
firewall-cmd --state >> "/tmp/${policyfilename}"
openscapinstalled=`yum list installed | grep openscap-scanner`
fi
if [ -z "$openscapinstalled" ]; then openscapinstalled="Openscap not installed"; fi
echo $openscapinstalled >> "/tmp/${policyfilename}"
```

8. Edit the ALL-Policy.sh as follows:

```
username=root
readarray -t SERVERS < ~/Policy_servers
for HOST in ${SERVERS[@]}; do
cat Policy.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/policy_*_$mydate ~/Policy/policy_${HOST}_${mydate}
cat ~/Policy/policy_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Policy/Policy_summary.
txt
done
username=ec2-user
readarray -t SERVERS < ~/Policy_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Policy.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/policy_*_$mydate ~/Policy/policy_${HOST}_${mydate}
cat ~/Policy/policy_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Policy/Policy_summary.
txt
done
username=AzureUser
readarray -t SERVERS < ~/Policy_servers_Azure
for HOST in ${SERVERS[@]}; do
cat Policy.sh | ssh ${username}@${HOST}
mydate=`date +"%Y-%m-%d"`
scp ${username}@${HOST}:/tmp/policy_*_$mydate ~/Policy/policy_${HOST}_${mydate}
cat ~/Policy/policy_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Policy/Policy_summary.
txt
done
```

9. Make ALL-Policy.sh executable:

```
chmod 0744 ALL-Policy.sh
```

10. Execute the looping script:

```
ssh-agent bash
ssh-add ~/AzureKey.pem
ssh-add ~/AWSKeyPair.pem
./ALL-Policy.sh
```

11. View the status of all the systems:

```
more Policy/Policy_summary.txt
ls ~/Policy
```

Insights workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Policies.
4. From the Policies screen, click Create policy.
5. Give the policy the name `firewalld not running` and description `firewalld is not running`
6. Click Next.
7. Enter the condition text `not (facts.enabled_services contains [‘firewalld’] and facts.running_process contains [‘firewalld’])`
8. Click Validate Condition.
9. Click Next.
10. Click the Add trigger actions drop-down menu.
11. Click to select the Email option.
12. In the new tab that opened, under Enable email alerts, click Open email preferences.
13. To the right of policies, next to Instant notification, click the box.
14. Click Submit.
15. Close the tab.
16. Click Next.
17. Next to the Policy is disabled text, click the slider to enable this policy.
18. Click Finish.
19. From the Policies screen, click Create policy.
20. Click the From scratch radio button.
21. Give the policy the name `openscap-scanner not installed` and description `openscap-scanner not installed`
22. Click Next.
23. Enter the condition text `not (facts.installed_packages contains [‘openscap-scanner’])`
24. Click Validate Condition.
25. Click Next.
26. Click the Add trigger actions drop-down menu.
27. Click to select the Email option.
28. Click Next.
29. Next to the Policy is disabled text, click the slider to enable this policy.
30. Click Finish.
31. Connect to the Admin system using ssh.
32. Log in.
33. Trigger a report on each of the systems:
    a. Edit the ~/Insights_servers file to contain all local servers you want to gather a report from:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | head -90 > ~/Insights_servers
```

    b. Edit the ~/Insights_servers_EC2 file to contain all AWS systems you want to gather a report from:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | head -95 | tail -n 5 > ~/Insights_
servers_EC2
```

    c. Edit the ~/Insights_servers_Azure file to contain all Azure systems you want to gather a report from:

```
grep ansible_host inventory.yml | awk -F\” ‘{ print $2 }’ | tail -n 5 > ~/Insights_servers_Azure
```

    d. Edit the insights.sh file as follows:

```
sudo insights-client
```

    e. Edit the ALL-Insights.sh as follows:

```
username=root
readarray -t SERVERS < ~/Insights_Servers
for HOST in ${SERVERS[@]}; do
cat insights.sh | ssh ${username}@${HOST}
done
username=ec2-user
readarray -t SERVERS < ~/Insights_Servers_EC2
for HOST in ${SERVERS[@]}; do
cat insights.sh | ssh ${username}@${HOST}
```

```
done
username=AzureUser
readarray -t SERVERS < ~/Insights_Servers_Azure
for HOST in ${SERVERS[@]}; do
cat insights.sh | ssh ${username}@${HOST}
done
```

34. Make ALL-Policy.sh executable:

```
chmod 0744 ALL-Insights.sh
```

35. Execute the looping script, which will put all the policy alerts in your inbox:

```
ssh-agent bash
ssh-add ~/AzureKey.pem
ssh-add ~/AWSKeyPair.pem
./ALL-Insights.sh
```

36. Prepare the fixes.
    a. Save a script for later that enables your firewall:

```
systemctl start firewalld
systemctl enable firewalld
```

    b. Save a script for later that installs openscap:

```
yum install openscap-scanner
```

Running the tests

Run four tests for each use case, for a total of 20 runs.

1. 1 system - Use the instructions in the "Manual workflow – 1 system" section of each use case. This test represents the initial investment of time to research and figure out how to check one system.
2. 50 systems - Use the instructions in the "Manual workflow – Scaling up to multiple systems" section of each use case. Portions of scripts for EC2 and Azure instances can be omitted in this test run.
3. 100 systems – Use the instructions in the "Manual workflow –Scaling up to multiple systems" section of each use case. Portions of scripts for EC2 and Azure instances should be included in this test run.
4. 100 systems – Use the instructions in the "Insights workflow" section of each use case.

Recording time and steps

1. List the necessary commands and methodology for the manual and Insights use cases before beginning.
2. Start a video screen capture and then run through the pre-determined steps at a steady pace, stopping and saving the video screen capture after completion. (When reading is necessary, allocate 1 minute for each screen scroll.)
3. Note the video timestamp when the use case started and stopped and subtract to find the total time taken to complete the test.
4. Use the number of steps and substeps listed in the methodology to determine the number of steps taken to complete the test.

Read the report at http://facts.pt/SaFHCpZ ▶

This project was commissioned by Red Hat.