EX342

Red Hat Enterprise Linux

Diagnostics and Troubleshooting

EXAMAIDES

PASS YOUR EXAM AT FIRST TRY

Volume: 18 Chapter Labs + One Final Lab (include 20 task Labs)

Lab1: Using the troubleshooting tools available within the RHEL environment
Lab2: Monitor systems for vital characteristics
Lab3: Configuring Remote Logging on Linux
Lab4: Troubleshoot service errors on start
Lab5: Manage kernel modules and their parameters
Lab6: Recover a corrupted filesystem
Lab7: Recover Mis-Configured or Broken LVM Configurations
Lab8: Recover data from encrypted file systems
Lab9: Identify and fix iSCSI issues
Lab10: Troubleshooting file system issues
Lab11: Troubleshooting RPM Issues
Lab12: Troubleshooting network issues
Lab13: Troubleshooting application issues
Lab14: Troubleshooting SELinux issues
Lab15: Identify and fix pluggable authentication (PAM) issues
Lab16: Identify and fix LDAP and Kerberos identity management issues
Lab17: Troubleshooting authentication issues
Lab18: Troubleshooting kernel issues
Final Lab: 20 task labs

Lab1: Using the troubleshooting tools available within the RHEL environment

Introduction

In this activity, you will practice using the troubleshooting tools available with RHEL 7.

Users are complaining of general slowness on this particular host ever since a junior employee was using it. They are also complaining about a lack of disk space available. You will need to log in and assess the situation and resolve any issues contributing to these complaints, if necessary.

Solution

Start by logging in to the lab server using the credentials provided on the hands-on lab page:
ssh cloud_user@PUBLIC_IP_ADDRESS

Become the root user:
sudo su –

Collect system information

You should begin by logging in to the host and obtaining the following information to verify the user

complaints:

- Current system load:
  uptime
- Note the load average shown and compare it to how many processors this server has:
  nproc
- Current disk usage:
  df -h
- Swap usage:
  free -m

Performance Co-Pilot is installed. You should use it to help assess the problems.

- Enable pmcd
  systemctl enable pmcd
- Start pmcd
  systemctl start pmcd

Resolve disk space issues

1. Start by using du to find large directories in /:
du -h --max-depth=1 /

2. We have narrowed down which directory is consuming the largest amount of space. Let's try and narrow it down a little more:
du -h --max-depth=1 /home/

3. Narrow it down a little further:
du -h --max-depth=1 /home/cloud_user/

4. The space is being consumed by a file in this directory. Let's list the files in this directory:
ll /home/cloud_user/

5. We can see a rather large file called old_personal_files.copy in this directory. Let's delete this file:
rm -rf /home/cloud_user/old_personal_files.copy

6. Verify the disk space issue is now resolved:
df -h

Resolve CPU load issues

1. Since we have pmcd running, let's use one of it's tools to see what is causing our CPU load issue:
pcp atop

2. The list of processes are sorted by CPU usage. This makes it easy to see that the cpuhog.sh process is using the largest amount of CPU. Copy the process ID (PID) for the cpuhog.sh process and close pcp atop.

3. Now we can kill that process. Replace PID in the following command with the PID of the process:
kill -9 PID

4. Verify that the CPU issue is resolved by running pcp atop again. After a moment, we should see the CPU usage return to a normal value:
pcp atop

Resolve disk I/O issues

1. With pcp atop still running, sort the list of processes by disk I/O by pressing **SHIFT+D**.

2. We can see an iohog.sh command is causing the issue. Same as before, copy the PID for that process and close pcp atop.

3. We can now kill that process. Replace PID in the following command with the PID of the process:
kill -9 PID

4. Verify that the disk I/O issue is resolved by running pcp atop again. After a moment, we should see disk I/O return to normal.
pcp atop

Lab2: Monitor systems for vital characteristics

Introduction

In this exercise, you will need to configure monitoring on a system with Performance Co-Pilot.

You've been asked to configure a system to provide live and historical metrics of its CPU load, disk I/O, and network traffic.

Solution

Start by logging in to the lab server using the credentials provided on the hands-on lab page:
ssh cloud_user@PUBLIC_IP_ADDRESS

Become the root user:
sudo su -

Install Performance Co-Pilot

Install pcp and pcp-system-tools:
yum -y install pcp pcp-system-tools

Enable and start the pmcd and pmlogger services.
systemctl enable pmcd pmlogger && systemctl start pmcd pmlogger

Take a baseline of CPU load

Take a baseline of the kernel.all.load metric for 10 seconds and put this into the file /home/cloud_user/kernel.all.load.txt.

You can do this using the pmval or pmrep command:
pmval -T 10s kernel.all.load > /home/cloud_user/kernel.all.load.txt
Or:
pmrep -T 10s kernel.all.load > /home/cloud_user/kernel.all.load.txt

View the contents of this file to ensure our command ran as intended:
cat /home/cloud_user/kernel.all.load.txt

Take a baseline of disk I/O

Take a baseline of the disk.partitions.total_rawactive metric for 10 seconds and put this into the file /home/cloud_user/disk.partitions.total_rawactive.txt.

You can do this using the pmval or pmrep command:
pmval -T 10s disk.partitions.total_rawactive > /home/cloud_user/disk.partitions.total_rawactive.txt
Or:
pmrep -T 10s disk.partitions.total_rawactive > /home/cloud_user/disk.partitions.total_rawactive.txt

Generate some disk I/O and CPU load

By now, pmlogger has been running for a few minutes. Generate some load so that we can look at it in the archive.

Before and after each of the commands that generate load, make a note of the system time. You can do so using the command:
Date

Generate some CPU load

Run the following command to generate some CPU load for 1 minute:
date && timeout -sHUP 1m openssl speed

Generate some disk I/O

Run the following command to generate some disk I/O:

```
date && fallocate -l 1G /home/cloud_user/bigfile && shred -zvu -n 1 /home/cloud_user/bigfile
```

Make a note of the start and end times from the commands above. We'll need them to know when to look for the increases in resource usages.

Verify the CPU and disk load in the pcp archive file

Get the pcp archive file:

```
pcp | grep logger
```

Look in the archive log directory and make note of the archive files:

```
ls -lh /var/log/pcp/pmlogger/ip-10-0-1-10.ec2.internal/
```

Depending on how long you've taken to do these tasks, the archive log may have rolled over to a new file. The format of the filename is YYYYMMDD.HH.MM. Using your notes of when you ran the CPU and disk I/O commands, determine which file to use.

Display the kernel.load.all values from the selected archive log in 1 minute increments:

Note: You can use pmval or pmrep here, with these particular metrics, I find pmrep to be easier to read.

```
pmrep -t 1m -a /var/log/pcp/pmlogger/ip-10-0-1-10.ec2.internal/&lt;FILE&gt; kernel.all.load
```

Display the kernel.load.all values from the selected archive log in 1 minute increments:

```
pmrep -t 1m -a /var/log/pcp/pmlogger/ip-10-0-1-10.ec2.internal/&lt;FILE&gt; kernel.all.load
```

Lab3: Configuring Remote Logging on Linux

Introduction

In this hands-on lab, you will configure remote logging from one server to another. The goal of this activity is to gain experience with being able to set up logging between servers.

In this activity, you need to configure Server1 as the log host for Server2.

Solution

Start by logging in to the lab servers using the credentials provided on the hands-on lab page:

```
ssh cloud_user@PUBLIC_IP_ADDRESS
```

Become the root user:

```
sudo su -
```

Be sure to log in to both Server1 and Server2 in separate tabs or windows.

Configure Server1 to receive logs

1. Server1 will need to be configured to receive logs via TCP.
vim /etc/rsyslog.conf

2. Uncomment the following two lines within the # Provides TCP syslog reception section:
$ModLoad imtcp
$InputTCPServerRun 514

3. Then, under the line starting with local7.* at the bottom of the file, add the following:
$template DynFile,"/var/log/hosts/system-%HOSTNAME%.log"
. -?DynFile

4. Save and close the file:
:wq

5. Restart the rsyslog service.
systemctl restart rsyslog

6. Verify the host is listening on port 514.
ss -lntp

7. Open the firewall to permanently permit incoming traffic on TCP port 514 and reload it.
firewall-cmd --permanent --add-port=514/tcp && firewall-cmd –reload

Configure Server2 to send logs to Server1

1. Verify Server2 can connect to Server1 over TCP port 514.

2. On Server2, modify the /etc/rsyslog.conf file.
vim /etc/rsyslog.conf

3. Uncomment the following lines in the ### begin forwarding rule ### section:
$ActionQueueFileName fwdRule1 # unique name prefix for spool files
$ActionQueueMaxDiskSpace 1g # 1gb space limit (use as much as possible)
$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
$ActionQueueType LinkedList # run asynchronously
$ActionResumeRetryCount -1 # infinite retries if host is down

4. Uncomment the following line and edit as follows:
. @@10.0.1.10:514

5. Restart the rsyslog service.

systemctl restart rsyslog

Verify logs are being sent to Server1

1. On Server1 verify the /var/log/hosts directory was created and is being populated.
ll /var/log/hosts

2. Use tail on the /var/log/hosts/system-ip-10-0-1-11.log file to see entries from Server2.
tail -f /var/log/hosts/system-ip-10-0-1-11.log

3. You can use the logger command to add entries to the log. On Server2, enter the following command 3 times:
logger "THIS IS A TEST"

4. Verify these entries are showing up in the log file on Server1.

Lab4: Troubleshoot service errors on start

Introduction

In this exercise, you will troubleshoot and resolve service errors upon start of the pmcd service.

Another administrator has escalated an issue regarding startup of the pmcd service. They are unable to successfully start the service.

Solution

Start by logging in to the lab servers using the credentials provided on the hands-on lab page:
ssh cloud_user@PUBLIC_IP_ADDRESS

Become the root user:
sudo su -

Resolve the startup issue with pmcd

1. Attempt to start the pmcd service:
systemctl start pmcd

2. If the service fails to start, take a look at the details:
systemctl status -l pmcd

Note the "Drop-In" file /etc/systemd/system/pmcd.service.d/dependency.conf.

3. View the contents of this file and note any directives referencing other services:
cat /etc/systemd/system/pmcd.service.d/dependency.conf

4. Attempt to start the pmlogger service:
systemctl start pmlogger

If the service won't start, perform the same exercise as above with pmcd.

5. Resolve the cyclical dependency by modifying the pmcd drop-in file from After=pmlogger.service to Before=pmlogger.service.
vim /etc/systemd/system/pmcd.service.d/dependency.conf

6. Start the pmcd service:
systemctl start pmcd

7. Verify pmlogger is running:
systemctl status pmlogger

Configure pmcd to start by default

Enable pmcd to startup upon reboot:
systemctl enable pmcd

Lab5: Manage kernel modules and their parameters

Introduction

In this exercise, you need to modify a kernel module to provide more information.

You have been asked to enable connection tracking timeflow stamping in the Linux kernel of a particular host. This change should take effect immediately, as well as persist upon reboot.

The kernel module to change is nf_conntrack, you need to modify the parameter that enables time stamping.

Solution

Start by logging in to the lab servers using the credentials provided on the hands-on lab page:
ssh cloud_user@PUBLIC_IP_ADDRESS

Become the root user:
sudo su -

Enable nf_conntrack time stamping in the running kernel

1. Check to see if the module is loaded:
`lsmod | grep nf_conntrack`

2. View the parameters available for the `nf_conntrack` module:
`modinfo nf_conntrack`

3. Verify the current setting of the timestamp parameter:
`cat /sys/module/nf_conntrack/parameters/tstamp`

4. Stop the firewall and unload the module:
`systemctl stop firewalld`
`modprobe -r nf_conntrack`

5. Load the module with timestamping enabled:
`modprobe nf_conntrack tstamp=1`

6. Verify the current setting of the timestamp parameter:
`cat /sys/module/nf_conntrack/parameters/tstamp`

Make the change persist through a restart

Make the change persist through a reboot:
`echo "options nf_conntrack tstamp=1" > /etc/modprobe.d/nf_conntrack.conf`

Lab6: Recover a corrupted filesystem

Introduction

In this exercise, you will repair and mount two corrupted file systems.

A junior administrator has asked for your assistance in troubleshooting a couple of mounts in `fstab`. They are unable to mount the drives, but insist the mounts have worked in the past.

Solution

Start by logging in to the lab servers using the credentials provided on the hands-on lab page:
`ssh cloud_user@PUBLIC_IP_ADDRESS`

Become the root user:
`sudo su -`

Resolve the first mount in `/etc/fstab`