Novell Open Enterprise Server 11: Best Practices Lab

OES05

Novell Training Services

www.novell.com

ATT LIVE 2012 LAS VEGAS

Novell®

Legal Notices

Novell, Inc.

404 Wyman Street, Suite 500

Waltham, MA 02451

U.S.A.

www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

Novell Trademarks

Third-Party Materials

Contents

SECTION 1 OES2 Troubleshooting and Support Tips Labs

The labs in this troubleshooting and support tips section deal with common Tech Support issues we have seen at Novell. In addition, some performance enhancement and diagnostics labs have also been included.

The sections in lab are:

Exercise 1-1 Configuring Bonding With Yast2

In this exercise you will set up bonding using yast2. Then you will look at the resulting config file. Finally make the suggested change to that file.

1. Configure NICs for bonding
   a. From the desktop of the VM open a terminal window by right clicking and selecting **Open Terminal**
   b. From the command prompt enter **yast2 lan**
   c. Highlight the first NIC and select **Edit**
   d. Select **No IP_address (for bonding)** the select **Next**
   e. Select the **General** tab
   f. From the Activate device drop down select **Never**
   g. Select the other NIC and select **Edit**
   h. Select **No IP_address (for bonding)** the select **Next**
   i. Select the **General** tab
2. Add bond for networking
   a. From the Network setting window Select **Add**
   b. From the **Device Type** drop down Select **Bond** the select **Next**
   c. Select Statically assigned IP Address and enter
      - IP Address 172.17.0.11
      - Subnet Mask 255.255.255.0
      - Hostname OES11.da
   d. Select the **Bond Slaves** tab
   e. Check the two boxes next to the NICs and select Next
   f. Select **OK**
   g. At the command prompt enter **init 6**
   h. After the VM reboots log in as **root** password **novell**
   i. From the desktop of the VM open a terminal window by right clicking and selecting **Open Terminal**
   j. At the command line enter **ping 172.17.0.1**
   k. Let is go for a bit the enter **Crtl+c** to stop the ping
   l. At the command line enter **ifconfig**

      the output should look something like this

```
bond0     Link encap:Ethernet  HWaddr 00:0C:29:6D:2D:47
          inet addr:172.17.0.11  Bcast:172.17.0.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe6d:2d47/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2740 (2.6 Kb)  TX bytes:3724 (3.6 Kb)

eth0      Link encap:Ethernet  HWaddr 00:0C:29:6D:2D:47
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:30 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2680 (2.6 Kb)  TX bytes:3724 (3.6 Kb)

eth1      Link encap:Ethernet  HWaddr 00:0C:29:6D:2D:47
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60 (60.0 b)  TX bytes:0 (0.0 b)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1142 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1142 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:212579 (207.5 Kb)  TX bytes:212579 (207.5 Kb)
```

Look at the packets sent and received on bond0 notice that they are the same of almost the same as the packets sent and received on ehth0. Almost nothing has been sent or received via eth1, this is because The bond is in Active-backup mode. This allows for fault tolerance.

3. Set the Bond to use ARP ping to determine if the NIC is alive

   a. At the command prompt enter **gedit /etc/sysconfig/network.ifcfg-bond0**

   b. Edit the BONDING_MODULE_OPT line so you replace miimon=100 with arp_interval= 1000 arp_ip_target= 172.17.0.1 the complete line should look like this (all one line)

      BONDING_MODULE_OPTS='mode=active-backup apr_interval=1000 arp_ip_target=172.17.0.1'

      if this was you sever you would need to restart the server or the networking for the change to take effect.

(End of Exercise)

Exercise 1-2 Troubleshooting Startup by Pausing init

One of the best methods for troubleshooting a booting Linux system is stepping through the initialization of services. This can be accomplished by editing the /etc/sysconfig/boot file and changing a couple of entries.

1. When booted, log in as **root** with password of **novell**.
2. Open a terminal window and enter **gedit /etc/sysconfig/boot**.
3. Change PROMPT_FOR_CONFIRM="**no**" to "**yes**".
4. Change CONFIRM_PROMPT_TIMEOUT="**5**" to "**30**".
5. Save and exit the file.
6. In order to see the boot process easier, change the splashscreen and grub entries:
7. At the terminal prompt enter **gedit /boot/grub/menu.lst**.
8. Find the line that says "title SUSE Linux Enterprise Server 11 SP1". It should be line 9. Just below that, on line 11 is the kernel load line.
9. Change "vga=0x317" to "**vga=normal**".
10. Change "splash=silent" to "**splash=0**" (That is a zero).
11. Save and exit the file.
12. Now edit the splashscreen call by entering **gedit /etc/sysconfig/bootsplash**.
13. Change SPLASH="yes" to "**no**". Save and exit that file.
14. At the oes2linux prompt, enter **reboot**.
15. Upon reboot, boot to the default entry in grub, and watch what happens. You will notice that as soon as it starts, the splash screen will blank and you will see boot text.
16. At the prompt to go into interactive mode, enter **y**.
17. Step through the init scripts by clicking entering **y** at each prompt. Do that for a few, then click **c** to continue unassisted the rest of the way.

(End of Exercise)

Exercise 1-3 Tracing and Monitoring Processes

In this lab you'll determine what files and libraries are being utilized and are required by programs when they run using tracing tools.

Using strace:

1. Revert OES11 VM to snapshot labled "**StartOfLab**" and power it on.
2. Log in as **root** with password of **novell**.
3. Open a terminal window.
4. Run the top program in a trace environment, looking for what files it reads when it runs with the command:

   strace -e trace=read -o strace_top1 top
5. This will run top interactively, sending all it's trace information for the session to the outfile strace_top1
6. Run the command, then quit top with the **q** key and then view the trace file by entering **less stace_top1**. (Use **q** to quit the less pager)
7. Notice that all the lines shown are read lines, this may not be what you need to figure out what files are being used, try the above command again with the open option added as shown below:

   strace -e trace=read,open -o strace_top2 top
8. View the file you just created and see if the addition of the open option shows the files you are trying to discover.
9. You will see a lot of files that are referenced by full paths, such as the /etc/toprc and /proc/stat etc.
10. Try the following command to see another view of the possible information:

    strace -e trace=file -o strace_top3 top
11. View the resulting file by entering **less strace_top3**.
12. The file option shows anything that used a file as a parameter, such as an open call that specified any library open.

Using ltrace:

1. Run the ltrace program on another program such as top, with the restriction of showing only System Calls, not library traces:

   ltrace -SL --output=ltrace_top1 top
2. Less the output file and see what the system calls output will tell you about a program. In troubleshooting, what you're most likely to look for is files that are being requested and not found.
3. The ltrace program also allows you to attach to an existing PID, such as top running in another session, so open another terminal window and run top in that console, then switch back to this console.

4. On this console, run the ps program, looking for the PID of the top program on the other console:

   ps aux | grep top

5. For example, if the top program is running with the PID of 13543, you can attach to that running process (it could be a misbehaving custom-written client server application) and get trace information with the command:

   ltrace -p 13543

6. Be prepared for a major amount of output, particularly from a program that refreshes the screen every few seconds like top does.

7. Press **Ctrl-C** to break out of ltrace.

8. You may want to send this output to a file, then tail the file as it's being created, such as the following:

   ltrace -p 13543 --output=ltrace_log1 &

   tail -f ltace_log1

9. This will allow you to both log the activity for later use, and keep an eye on it as it's happening.

(End of Exercise)

Exercise 1-4 Using sar to Capture System Activity

Getting real time metrics is not the only important thing; the historical trend is equally as important.

Furthermore, consider this situation: how many times has someone reported a performance problem, but when you dive in to investigate, everything is back to normal? Performance issues that have occurred in the past are difficult to diagnose without any specific data as of that time. Using the sar command, you can examine the performance data over the past few days to decide on some settings or to make adjustments.

The sar utility accomplishes that goal. sar stands for System Activity Recorder, which records the metrics of the key components of the Linux system—CPU, Memory, Disks, Network, etc.—in a special place: the directory /var/log/sa. The data is recorded for each day in a file named sa<nn> where <nn> is the two digit day of the month. For instance the file sa27 holds the data for the date 27th of that month. This data can be queried later on and even graphed for historical or forensic purposes.

1. If not already launch the OES2Linux VM and log in as root.
2. Open a terminal window.
3. Install the sysstat package by entering the following:

   yast2 -i sysstat

   NOTE: If you are prompted to insert a CD, make sure the SLES-10-SP2-DVD-i386-GM-DVD1.iso is attached to the VM.

   While you're at it, install the gnuplot apps, which will be needed later in this lab:

   yast2 -i gnuplot

4. Change directory to /var/log/sa and notice that there are no file present. This is because the sadc daemon is not collecting any data yet.
5. Enter the following to turn on the sar daemon:

   /etc/init.d/boot.sysstat start

6. Now look in /var/log/sa. Notice that there is a file by the name saXX. This file is used to hold the binary sar data.
7. Have a look at /etc/sysstat/sysstat.cron. How often will a new report be generated? Actually there are 2 reports: one every 10 minutes, and one every 6 hours.
8. Run the sar command. It should show that the service is started, but there are no data yet (it hasn't been 10 minutes).
9. One performance issue could be caused by an overloaded interrupt request line.
10. Enter the following to view the current interrupts in use and determine which interrupt the ethernet card is using:

    cat /proc/interrupts

11. If the ethernet interrupt is 67 (vmxnet ether), then enter the following at the command line:

    sar -I 67 2 5

    This will generate a sar report every 2 seconds 5 times for interrupt 67. You will notice that there isn't much activity.

12. Put some load on the internet card by flood pinging it from the host:

13. Minimize VMWare and open a terminal window.

14. Enter the following:

    ping -f 172.17.0.21

15. Now maximize VMWare and run the sar command again.

16. This time you should see some interrupt activity.

17. Cancel the ping on the host by pressing **Ctrl-C**.

18. At the OES2Linux prompt, enter the following:

    sar -n ALL

    This will display all network statistics.

Using isag to graph the results

1. Install sysstat-isag it by entering the following:

   yast2 -i sysstat-isag

2. After 10 minutes, you should have at least 1 sar report generated. Once reports are generated, you can run the system activity graphing utility (isag) to view historical sar trends.

3. At the command line enter the following to filter for ALL files that star with sa:

   isag -m sa*

4. Select '-' (the minus button) to select a data source and choose the first date file.

5. Select options, Select whole day (so that it is not red). This should be the default. Since we don't have an entire day's data yet, we don't need to show the whole day graph.

6. Select **chart->CPU utilization** to set the filter for CPU.

(End of Exercise)

Exercise 1-5 Forcing a seg Fault on an Application to Retrieve Core Dump Info

For Unix systems, the term "core dump" generally refers to a dump of the state of an individual process. The term "core dump" is also used in the context of NetWare, but in that context it refers to a dump of the state of a complete system. The equivalent of a NetWare core dump for Unix systems is typically called a "system crash dump" or "kernel crash dump".

In this lab, you will configure the system for, and then force a core dump of a specific application.

1. At the terminal prompt on oes2linux, enter the following to list the current ulimit configuration:

   ulimit -a

   On Unix systems, limits can be set for various resources available to processes. These limits can be displayed through "ulimit -a". One limitable value is the size of core dumps. To ensure a complete core dump can be written, run "ulimit -c unlimited" prior to starting the application of which the core dump needs to be captured. Changing the limit does not affect running processes and only applies to processes started the current shell.

2. Enter the following to enable core dumps:

   ulimit -c unlimited

3. To change limits permanently, install the ulimit package, configure the limits in the /etc/sysconfig/ulimit configuration file and reboot the system for the changed limits to take effect.

4. To install the ulimit package, ether the following:

   yast2 -i ulimit

5. Make the ulimit permanent by editing /etc/sysconfig/ulimit:

   gedit /etc/sysconfig/ulimit

   Change the SOFTCORELIMIT=”0” to “**unlimited**”

6. By default, the kernel writes core dump files in the current working directory of the crashing process (if file system permissions and ACLs allow). This can make it difficult to locate core dump files.

7. To store core dumps in a fixed directory, first create a suitable directory, say /var/local/dumps:

   install -m 1777 -d /var/local/dumps

   1777 corresponds to world writable, deletable only by owner. Next instruct the kernel to store dumps there:

   echo "/var/local/dumps/core.%e.%p"> /proc/sys/kernel/core_pattern

   The kernel will expand %e to the name of the crashing process and %p to the PID.

8. Make this permanent by editing **/etc/sysctl.conf** and putting the following into it:

kernel.core_pattern=/var/local/dumps/core.%e.%p

9. AppArmor application security is based on learning the normal/good behaviour of an application and then preventing the application from performing operations that do not fit the behaviour learnt. As writing a core dump is typically not part of an application's normal/good behaviour, AppArmor is likely to prevent a core file from being written.

10. Refer to the AppArmor documentation for details on how to disable AppArmor selectively. For this exercise, manually disable it by entering:

    rcapparmor stop

11. Unix systems provide a facility, the setuid and setgid bits, to execute processes under a different user id or group id than that of the user/group starting the process. Such a process may be dealing with data which should not be directly accessible to the invoking user/group and if a core dump were written for such a process, that data could be leaked. For this reason, the kernel does not write core dumps for setuid/setgid processes by default. Similarly, by default the kernel will not write core dumps for processes running as the root user when doing so could leak information. This default behaviour can be overridden through the kernel.suid_dumpable sysctl:

    sysctl -w kernel.suid_dumpable=2

12. To make this configuration change persistent over reboots, add the line:

    kernel.suid_dumpable=2

    to the /etc/sysctl.conf configuration file.

13. Now launch firefox by entering **firefox &** at the command prompt.

14. Manullay trigger a core dump of firefox by entering the following:

    kill -ABRT $(pidof firefox-bin)

15. Now check that the core dump file was written in /var/local/dumps/. It will be called core.firefox-bin.<PID>

16. Now that the core dump has been written, it can be sent into Novell Technical Support for debugging.

(End of Exercise)