Debug a problem with a Cloud Deployment and Fix it

Introduction

You're an IT administrator in a small-sized startup that runs a web server named `ws01` in the cloud. One day, when you try to access the website served by `ws01`, you get an HTTP Error 500. Since your deployment is still in an early stage, you suspect a developer might have used this server to do some testing or run some experiments. Now you need to troubleshoot `ws01`, find out what's going on, and get the service back to a healthy state.

Error detection

Open the website served by ws01 by typing the external IP address of `ws01` in a web browser. The external IP address of `ws01` can be found in the Connection Details Panel on the left-hand side.

You should now find **500 Internal Server Error!** on the webpage. Later, during this lab, you'll troubleshoot this issue.

What you'll do

- Understand what `http` status code means
- Learn how to check port status with the `netstat` command
- Learn how to manage services with the `systemctl` command
- Know how to monitor system resources and identify the root cause of an issue

You'll have 90 minutes to complete this lab.

Debug the issue

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped into five classes:

- Informational responses (100–199)
- Successful responses (200–299)
- Redirects (300–399)
- Client errors (400–499)
- Server errors (500–599)

The HyperText Transfer Protocol (HTTP) **500 Internal Server Error** response code indicates that the server encountered an unexpected condition that prevented it from fulfilling the request. Before troubleshooting the error, you'll need to understand more about `systemctl`.

`systemctl` is a utility for controlling the `systemd` system and service manager. It comes with a long list of options for different functionality, including starting, stopping, restarting, or reloading a daemon.

Let's now troubleshoot the issue. Since the webpage returns an HTTP error status code, let's check the status of the web server i.e apache2.

```
sudo systemctl status apache2
```

The command outputs the status of the service.

Output:

```
gcpstaging100395_student@ws01:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: failed (Result: exit-code) since Thu 2019-12-26 14:29:21 UTC; 4min 58s ago

Dec 26 14:29:21 ws01 systemd[1]: Starting The Apache HTTP Server...
Dec 26 14:29:21 ws01 apachectl[2955]: (98)Address already in use: AH00072: make_sock: could not bind to address [::
Dec 26 14:29:21 ws01 apachectl[2955]: (98)Address already in use: AH00072: make_sock: could not bind to address 0.0
Dec 26 14:29:21 ws01 apachectl[2955]: no listening sockets available, shutting down
Dec 26 14:29:21 ws01 apachectl[2955]: AH00015: Unable to open logs
Dec 26 14:29:21 ws01 apachectl[2955]: Action 'start' failed.
Dec 26 14:29:21 ws01 apachectl[2955]: The Apache error log may have more information.
Dec 26 14:29:21 ws01 systemd[1]: apache2.service: Control process exited, code=exited status=1
Dec 26 14:29:21 ws01 systemd[1]: apache2.service: Failed with result 'exit-code'.
Dec 26 14:29:21 ws01 systemd[1]: Failed to start The Apache HTTP Server.
```

The outputs say "Failed to start The Apache HTTP Server." This might be the reason for the HTTP error status code displayed on the webpage. Let's try to restart the service using the following command:

```
sudo systemctl restart apache2
```

Output:

```
gcpstaging100395_student@ws01:~$ sudo systemctl restart apache2
Job for apache2.service failed because the control process exited with error code.
See "systemctl status apache2.service" and "journalctl -xe" for details.
```

Hmm this command also fails. Let's check the status of the service again and try to find the root cause of the issue.

```
sudo systemctl status apache2
```

Output:

```
gcpstaging100395_student@ws01:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: failed (Result: exit-code) since Thu 2019-12-26 14:36:53 UTC; 1min 11s ago
  Process: 4181 ExecStart=/usr/sbin/apachectl start (code=exited, status=1/FAILURE)

Dec 26 14:36:53 ws01 systemd[1]: Starting The Apache HTTP Server...
Dec 26 14:36:53 ws01 apachectl[4181]: (98)Address already in use: AH00072: make_sock: could not bind to address [::]:80
Dec 26 14:36:53 ws01 apachectl[4181]: (98)Address already in use: AH00072: make_sock: could not bind to address 0.0.0.0:80
Dec 26 14:36:53 ws01 apachectl[4181]: no listening sockets available, shutting down
Dec 26 14:36:53 ws01 apachectl[4181]: AH00015: Unable to open logs
Dec 26 14:36:53 ws01 apachectl[4181]: Action 'start' failed.
Dec 26 14:36:53 ws01 apachectl[4181]: The Apache error log may have more information.
Dec 26 14:36:53 ws01 systemd[1]: apache2.service: Control process exited, code=exited status=1
Dec 26 14:36:53 ws01 systemd[1]: apache2.service: Failed with result 'exit-code'.
Dec 26 14:36:53 ws01 systemd[1]: Failed to start The Apache HTTP Server.
```

Take a close look at the output. There's a line stating "Address already in use: AH00072: make_sock: could not bind to address [::]:80." The Apache webserver listens for incoming connection and binds on port 80. But according to the message displayed, port 80 is being used by the other process, so the Apache web server isn't able to bind to port 80.

To find which processes are listening on which ports, we'll be using the `netstat` command, which returns network-related information. Here, we'll be using a combination of flags along with the `netstat` command to check which process is using a particular port:

```
sudo netstat -nlp
```

Output:

```
gcpstaging100395_student@ws01:~$ sudo netstat -nlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      2283/python3
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      822/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1626/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      1626/sshd
udp        0      0 127.0.0.53:53           0.0.0.0:*                           822/systemd-resolve
udp        0      0 10.128.0.2:68           0.0.0.0:*                           782/systemd-network
udp        0      0 127.0.0.1:323           0.0.0.0:*                           2091/chronyd
udp6       0      0 ::1:323                 :::*                                2091/chronyd
raw6       0      0 :::58                   :::*                    7           782/systemd-network
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     SEQPACKET  LISTENING     12217    1/init               /run/udev/control
unix  2      [ ACC ]     STREAM     LISTENING     30871    3604/systemd         /run/user/1898256998/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     30875    3604/systemd         /run/user/1898256998/gnupg/S.gpg-agent.browser
unix  2      [ ACC ]     STREAM     LISTENING     30876    3604/systemd         /run/user/1898256998/snapd-session-agent.socket
unix  2      [ ACC ]     STREAM     LISTENING     30877    3604/systemd         /run/user/1898256998/gnupg/S.gpg-agent.extra
```

You can see a process ID (PID) and an associated program name that's using port 80. A python3 program is using the port.

Note: Jot down the PID of the python3 program in your local text editor, which will be used later in the lab. Let's find out which python3 program this is by using the following command:

```
ps -ax | grep python3
```

Output:

```
gcpstaging100395_student@ws01:~$ ps -ax | grep python3
 1120 ?        Ssl    0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
 1289 ?        Ssl    0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
 1384 ?        Ss     0:00 /usr/bin/python3 /usr/bin/google_network_daemon
 1413 ?        Ss     0:00 /usr/bin/python3 /usr/bin/google_accounts_daemon
 1414 ?        Ss     0:00 /usr/bin/python3 /usr/bin/google_clock_skew_daemon
 2283 ?        Ss     0:00 python3 /usr/local/bin/jimmytest.py
 4706 pts/0    S+     0:00 grep --color=auto python3
```

There is a list of python3 processes displayed here. Now, look out for the PID of the process we're looking for and match it with the one that's using port 80 (output from `netstat` command).

You can now obtain the script `/usr/local/bin/jimmytest.py` by its PID, which is actually using port 80.

Have a look at the code using the following command:

```
cat /usr/local/bin/jimmytest.py
```

This is indeed a test written by developers, and shouldn't be taking the default port.

Let's kill the process created by `/usr/local/bin/jimmytest.py` by using the following command:

```
sudo kill [process-id]
```

Replace `[process-id]` with the PID of the python3 program that you jotted down earlier in the lab.

List the processes again to find out if the process we just killed was actually terminated.

```
ps -ax | grep python3
```

This time you'll notice that similar process running again with a new PID.

This kind of behavior should be caused by service. Since this is a python script created by Jimmy, let's check for the availability of any service with the keywords "`python`" or "`jimmy`".

```
sudo systemctl --type=service | grep jimmy
```

Output:

```
gcpstaging100395_student@ws01:~$ sudo systemctl --type=service | grep jimmy
● jimmytest.service                  loaded failed failed  Jimmy python test service
```

There is a service available named `jimmytest.service`. We should now stop and disable this service using the following command:

```
sudo systemctl stop jimmytest && sudo systemctl disable jimmytest
```

Output:

```
gcpstaging100395_student@ws01:~$ sudo systemctl stop jimmytest && sudo systemctl disable jimm
ytest
Removed /etc/systemd/system/default.target.wants/jimmytest.service.
```

The service is now removed.

To confirm that no processes are listening on 80, using the following command:

```
sudo netstat -nlp
```

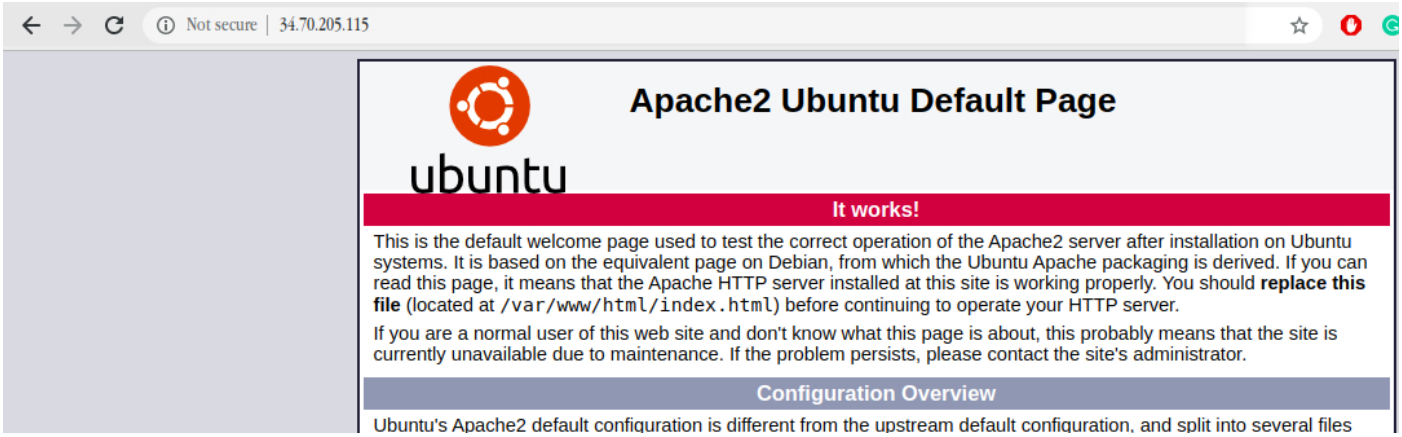
Output:

```
student-04-918b156dde6d@ws01:~$ sudo netstat -nlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      786/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1533/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      1533/sshd
udp        0      0 127.0.0.53:53           0.0.0.0:*                           786/systemd-resolve
udp        0      0 10.128.0.2:68           0.0.0.0:*                           760/systemd-network
udp        0      0 127.0.0.1:323           0.0.0.0:*                           1952/chronyd
udp6       0      0 ::1:323                 :::*                                1952/chronyd
raw6       0      0 :::58                   :::*                    7           760/systemd-network
```

Since there are no processes listening on port 80, we can now start `apache2` again.

```
sudo systemctl start apache2
```

Refresh the browser tab that showed **500 Internal Server Error!** Or you can open the webpage by typing the external IP address of `ws01` in a new tab of the web browser. The external IP address of `ws01` can be found in the Connection Details Panel on the left-hand side.



You should now be able to see the **Apache2 Ubuntu Default Page**.