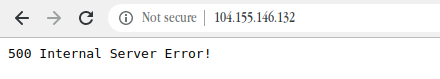
Debug a problem with a Cloud Deployment and Fix it

**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](https://developer.mozilla.org/en-US/docs/Web/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.

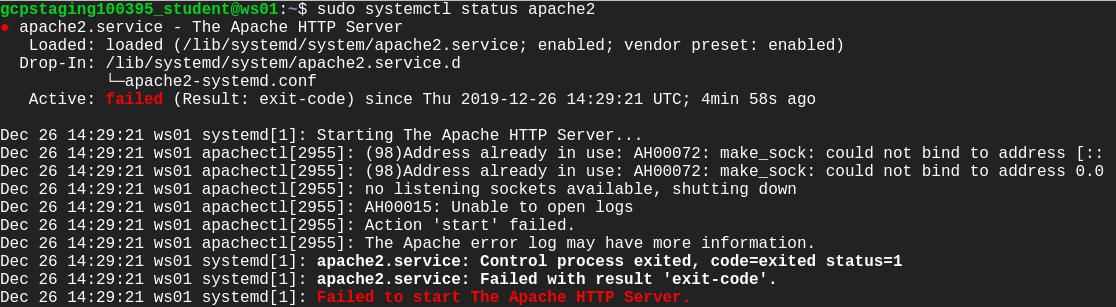
systemctl is a utility for controlling the systemd system and service manager. It comes with a long list of options for different functionality, including starting, stopping, restarting, or reloading a daemon.

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:



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

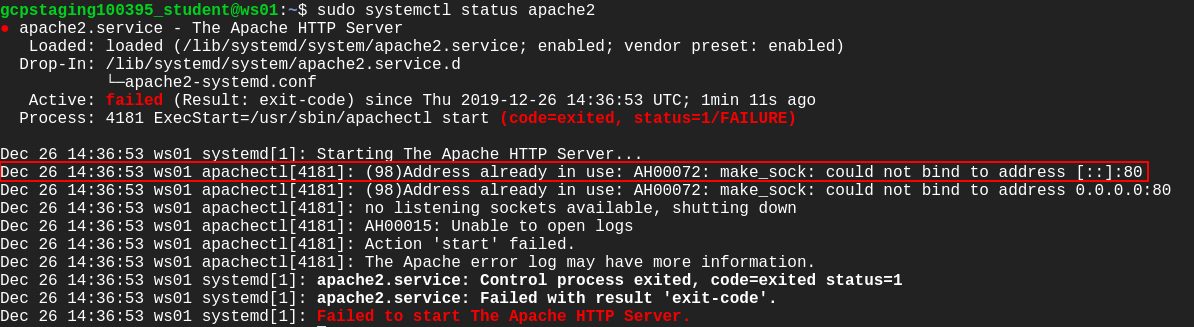
Output:



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:

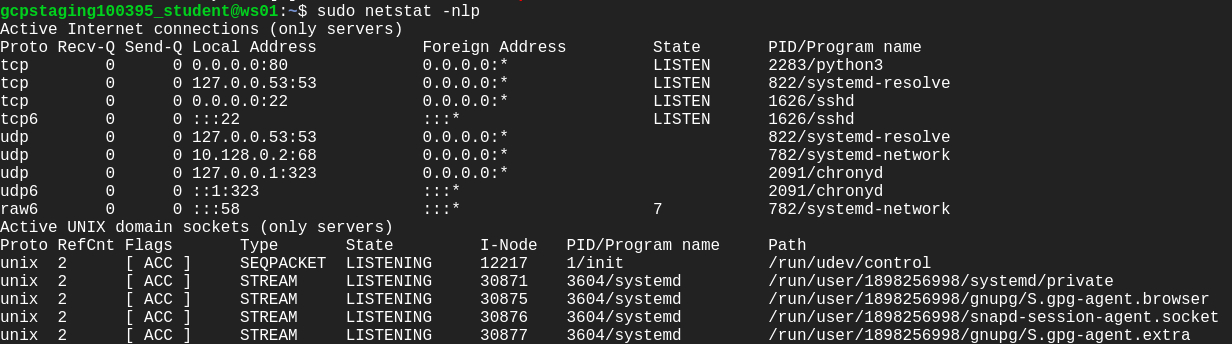


Take a close look at the output. There's a line stating "Address already in use: AH00072: make\_sock: could not bind to address [::]:80." The Apache webserver listens for incoming connection and binds on port 80. But according to the message displayed, port 80 is being used by the other process, so the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



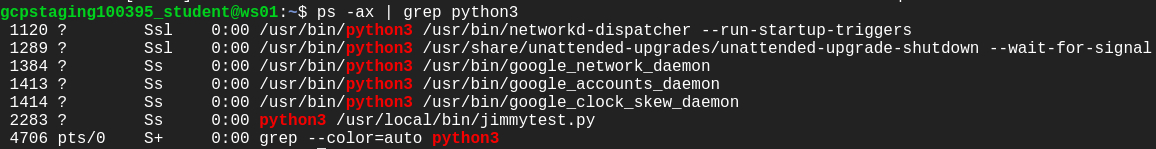
You can see a process ID (PID) and an associated program name that's using port 80. A python3 program is using the port.

**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:



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:



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:

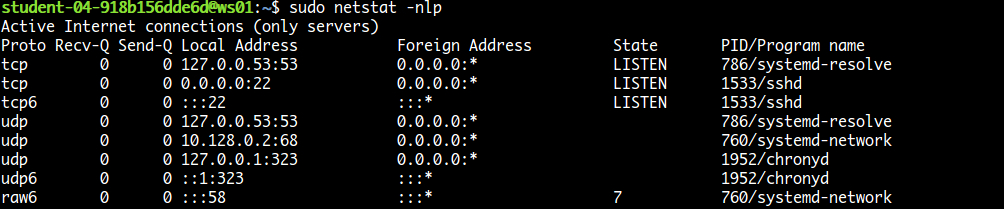


The service is now removed.

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

sudo netstat -nlp

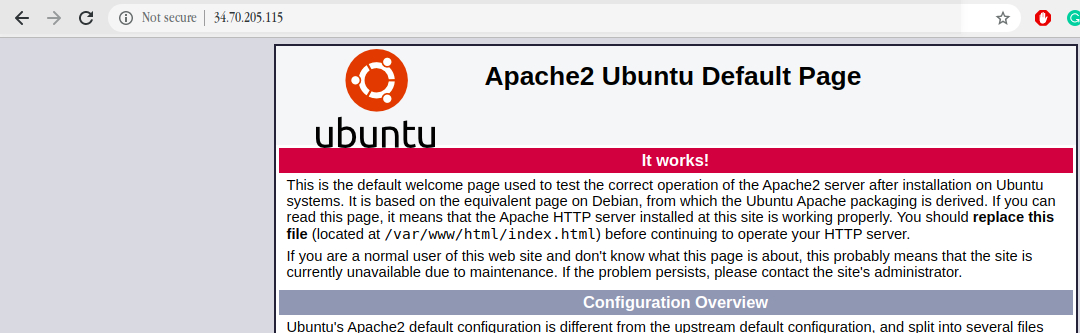
Output:



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**.

**Congratulations!**

You've successfully fixed the website served by the ws01 and brought the service back to a healthy state! Nice work.