**NOHUP COMMAND IN LINUX**

Every command in Linux starts a process at the time of its execution, which automatically gets terminated upon exiting the terminal. Suppose, you are executing programs over SSH and if the connection drops, the session will be terminated, all the executed processes will stop, and you may face a huge accidental crisis. In such cases, running commands in the background can be very helpful to the user and this is where **nohup command**comes into the picture. **nohup (No Hang Up)** is a command in Linux systems that runs the process even after logging out from the shell/terminal.

Usually, every process in Linux systems is sent a **SIGHUP (Signal Hang UP)** which is responsible for terminating the process after closing/exiting the terminal. Nohup command prevents the process from receiving this signal upon closing or exiting the terminal/shell. Once a job is started or executed using the nohup command, **stdin** will not be available to the user and **nohup.out** file is used as the default file for **stdout** and**stderr**. If the output of the nohup command is redirected to some other file, **nohup.out** file is not generated.

## **Nohup Command Syntax**

The syntax for using the Nohup command is straightforward:

**nohup command [options] &**

* **command** : Specifies the command or script that you want to execute.
* **options** : Optional arguments or flags that modify the behavior of the command.
* **&** :Placing an ampersand (&) at the end of the command instructs the shell to run the command in the background.

## **Checking the version of Nohup**

## 

## **nohup –version**

## **Starting a Process Using Nohup**

## **nohup java -jar /opt/myapp/app.jar > output.log 2>&1 &**

* **output.log →** saves standard output
* **2>&1** → redirects errors to the same file
* **&** → runs the process in the background

After running this, you'll see something like:

**[1] 23456**

23456 is the **Process ID (PID)**

**nohup bash geekfile.sh**

**To redirect the output to the output.txt file**

**nohup bash geekfile.sh > output.txt**

## **Starting a Process in the Background Using Nohup**

## **nohup bash geekfile.sh &**

## **fg**

## **To run multiple commands in the background**

## **nohup bash -c 'commands'**

**Example:**

**nohup bash -c 'cal && ls'**

the output will be by default stored in **nohup.out**. To redirect it, type

**cat nohup.out**

## **What is a PID?**

**PID** = **Process ID**, a unique number assigned to every running process in Linux.

You need the PID when you want to:

* Monitor a process
* Stop (kill) a process

## **Check Running Processes with ps**

Use ps to view all running processes.

### **Show All Java Processes**

### **ps aux | grep java**

This might return something like:

**kali 23456 0.5 3.2 123456 7890 ? Ssl 11:00 0:02 java -jar /opt/myapp/app.jar**

* 23456 is the PID
* You now know the app is running and what command launched it

## **Kill the Process Using PID**

To stop your Spring Boot app:

**kill 23456**

Or force it if it doesn’t stop:

**kill -9 23456**

## **Stop Any JAR App by Name**

## **pkill -f app.jar**

This kills all Java processes with "app.jar" in the command

## **View Output of the Running App**

Since you used nohup with an output file:

**tail -f output.log**

This shows live log output just like

**tail -f spring.log.**