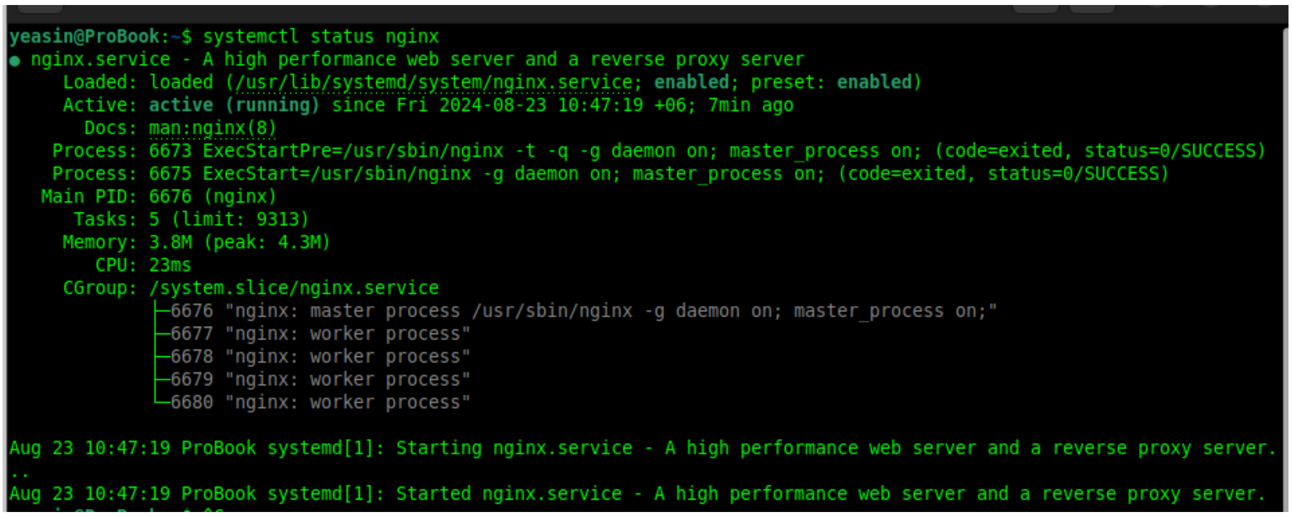
**Nginx:** Nginx is a popular web server which makes sure the availability of the busiest websites on the internet. It can work as load balancer, reverse proxy and rate limiting for websites. Nginx also can work as cache server.
I am giving basic setup of Nginx Load Balancing configurations.
Installation of Nginx in Debian:

1. sudo apt install nginx (Installing nginx)
2. ps aux | grep nginx (Check if nginx is running. If its running then you will see, a master process and some worker processes)
3. systemctl status nginx (nginx service can also be checked by the following command where it will be shown master and worker processes)



```
yeasin@ProBook:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-08-23 10:47:19 +06; 7min ago
       Docs: man:nginx(8)
    Process: 6673 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 6675 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 6676 (nginx)
      Tasks: 5 (limit: 9313)
     Memory: 3.8M (peak: 4.3M)
        CPU: 23ms
     CGroup: /system.slice/nginx.service
             ├─6676 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─6677 "nginx: worker process"
             ├─6678 "nginx: worker process"
             ├─6679 "nginx: worker process"
             └─6680 "nginx: worker process"

Aug 23 10:47:19 ProBook systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server.
..
Aug 23 10:47:19 ProBook systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```

4. Some important commands:
   I. sudo systemctl start nginx (to start nginx, it won't start in next boot)
   II. sudo systemctl enable nginx (it will make nginx as a service and restart at every reboot)
   III. sudo systemctl stop nginx (to stop nginx)

## Nginx Load Balancing

For example, we have 3 servers, the first one (server1.yeasin.com) is very powerful, second one (server2.yeasin.com) is default and third one (190.4.0.97) is for backup.
The content of /etc/nginx/nginx.conf will be,

```
http {
        upstream backend_servers_list{
                        server server1.yeasin.com weight 10;
                        server server2.yeasin.com;
                        server 190.4.0.97 backup;
        }
}
```

The powerful first server will handle requests 10 times more than the second server. If the first 2 became unavailable, then backup server will handle the requests.

Passing requests to the server list.
For example, I have a website named yeasincse19.com. Whenever my clients make some requests to my website, I want to load balance to my servers listed above.

```
server {
        listen 80;
        server_name www.yeasincse19.com;
        location / {
                proxy_pass http://backend_server_list;
        }
}
```

This is the request sequence:

1. Someone visits www.yeasincse19.com
2. Nginx checks its directive and sees that the request is for www.yeasincse19.com. So, it will use the configuration of the server block.
3. It will check the location (which is /, for all location) and pass the request to the backend_server_list which has 3 servers.

## Load Balancing Methods

**1. Default Method:** Round Robin, in Operating System course, I learned this algorithm for Process Scheduling, also modified it with Shortest Job First (SJF) algorithm to make a new better one. It’s a simple algorithm which assigns jobs to CPUs one by one and in a circular method. There is a time called time quantum, every task/job will have the CPU for quantum of time to process. The queue will be first come first serve.
As its default load balancing method, we don’t need extra configuration.

**2. Least Connections:** As the name suggests, the request will be assigned to the server which has least active connection.

```
        upstream backend_servers_list{
                least_conn;
                server server1.yeasin.com weight 10;
                server server2.yeasin.com;
                server 190.4.0.97 backup;
        }
```

**3. IP-Hash:** This method will determine which server will get the the next request. There is a value ip_hash, its determined from the first 3 octec of ipv4 address or the entire ipv6 address.

```
        upstream backend_servers_list{
                ip_hash;
                server server1.yeasin.com weight 10;
                server server2.yeasin.com;
                server 190.4.0.97 backup;
        }
```

**4. Generic Hash:** This method similar to IP-has but the hash will be user given which might form from different things like ip address, port number, url.

```
        upstream backend_servers_list{
                hash $user_given_string_hash;
                server server1.yeasin.com weight 10;
                server server2.yeasin.com;
                server 190.4.0.97 backup;
        }
```