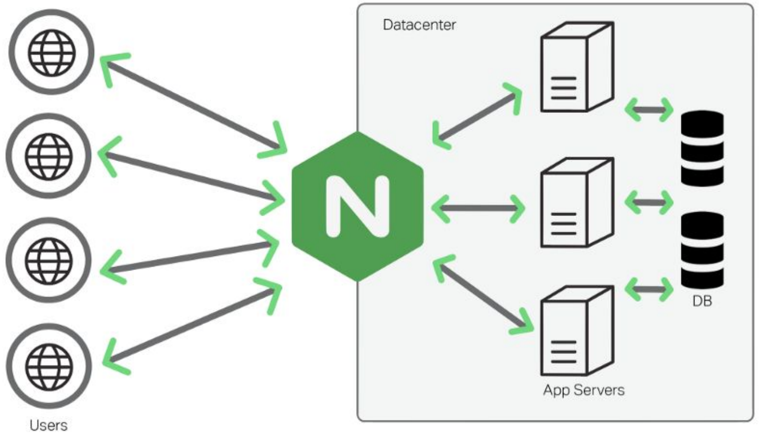


Nginx for Load Balancing

[Waheed Iqbal]()

DevOps (Fall 2023)
Department of Data Science, FCIT, University of the Punjab
Lahore, Pakistan

Introduction

- Nginx (pronounced "engine-x") is a high-performance, open-source web server and reverse proxy server software.
- It is well-known for its speed, reliability, and efficiency in handling concurrent connections.
- Originally created by Igor Sysoev in 2004, Nginx has gained widespread popularity and is used by many large-scale websites and applications.

Nginx Key Features

Web Server:

- Nginx can serve static content like HTML, CSS, and images directly to clients.
- Its efficient handling of concurrent connections makes it suitable for high-traffic websites.

Reverse Proxy:

- Nginx can act as a reverse proxy, forwarding requests to other servers (e.g., application servers) and then returning the responses to clients.

Load Balancer:

- Nginx can distribute incoming traffic across multiple servers, balancing the load and ensuring that no single server is overwhelmed.
- Provides improved scalability and fault tolerance.

HTTP Server and Cache:

- Nginx supports various HTTP features, including SSL/TLS termination, virtual hosting, and URL redirection.
- It can also cache static content, reducing the load on backend servers and improving response times.

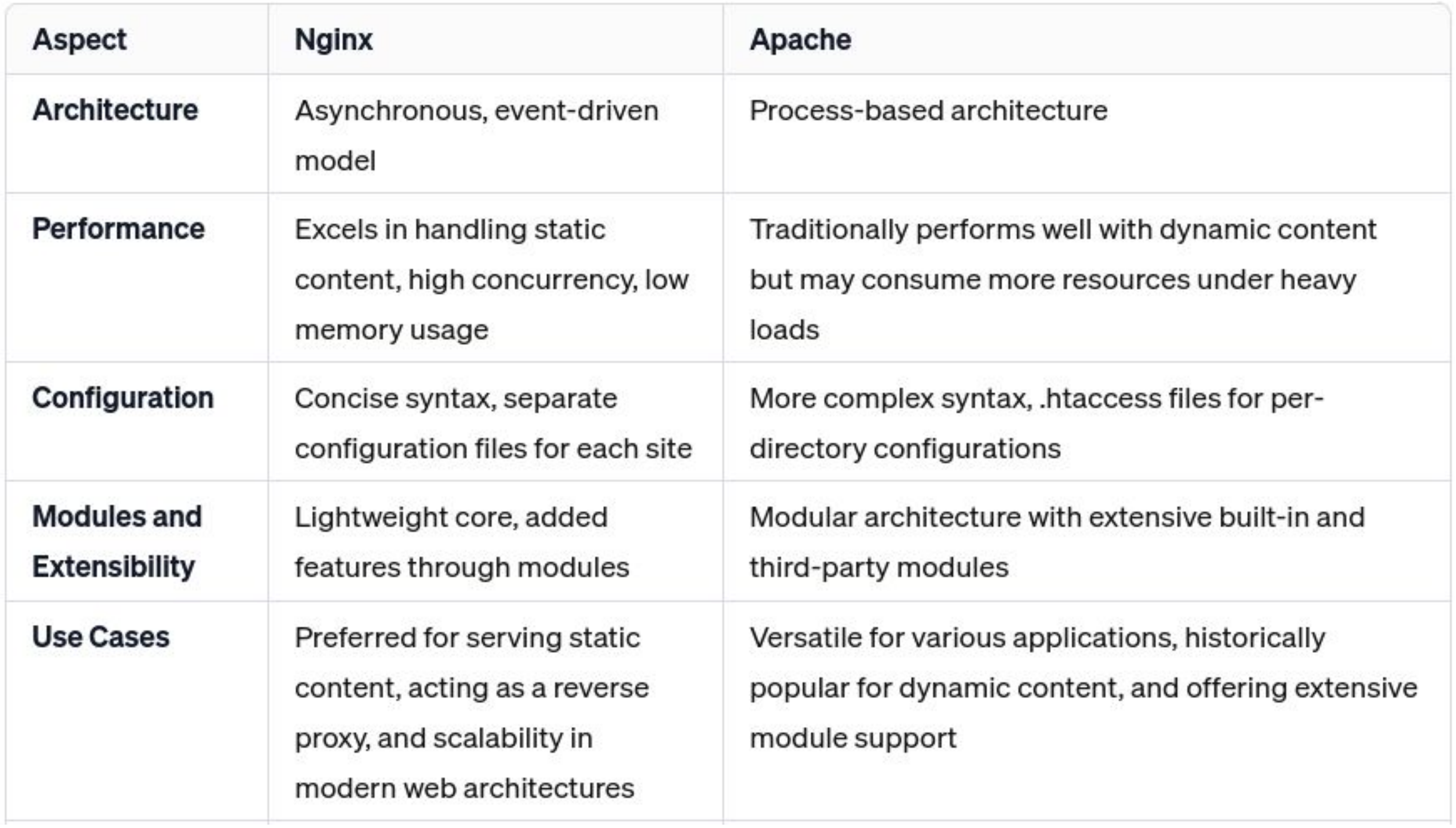
Security:

- Nginx includes features for securing web applications, such as access control, SSL/TLS support, and protection against common web attacks like DDoS and SQL injection.

Open Source and Community Support:

- Nginx is open-source software, and its source code is freely available.
- It has a large and active community, providing support, documentation, and third-party modules.

Nginx vs Apache



Aspect	Nginx	Apache
Architecture	Asynchronous, event-driven model	Process-based architecture
Performance	Excels in handling static content, high concurrency, low memory usage	Traditionally performs well with dynamic content but may consume more resources under heavy loads
Configuration	Concise syntax, separate configuration files for each site	More complex syntax, .htaccess files for per-directory configurations
Modules and Extensibility	Lightweight core, added features through modules	Modular architecture with extensive built-in and third-party modules
Use Cases	Preferred for serving static content, acting as a reverse proxy, and scalability in modern web architectures	Versatile for various applications, historically popular for dynamic content, and offering extensive module support

Key Terms

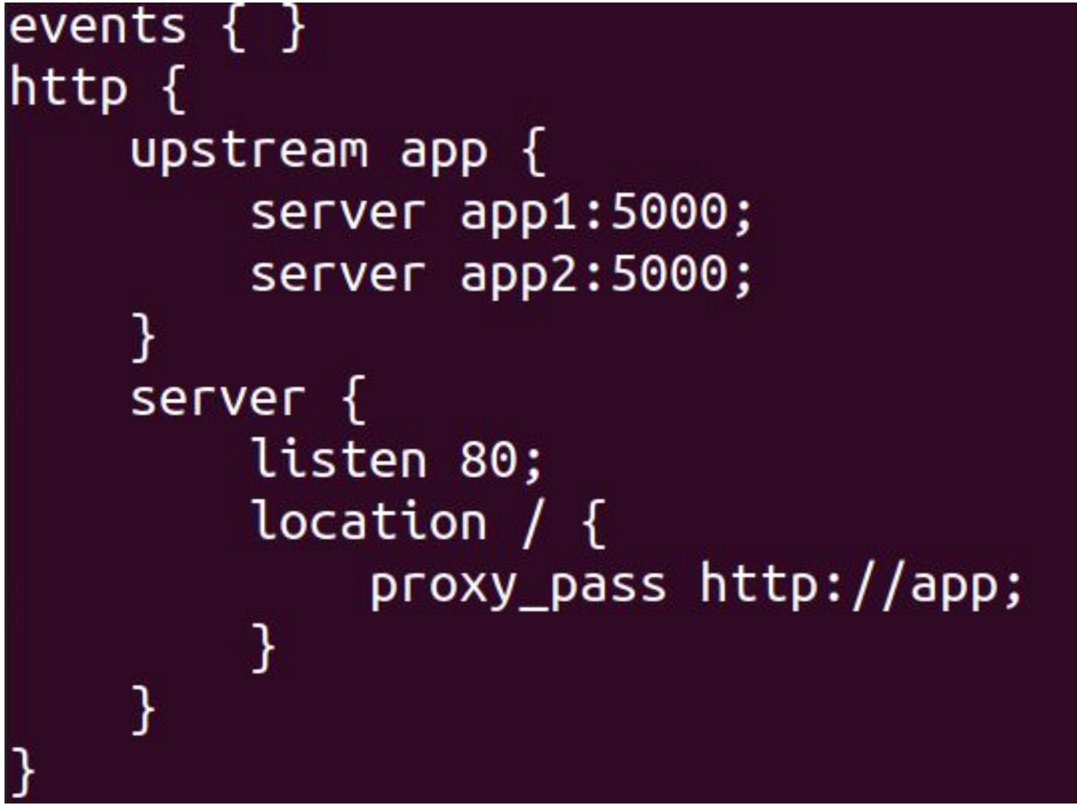
- **Reverse Proxy:** Handles requests from clients on behalf of backend servers. Clients request resources from the reverse proxy, which then forwards the requests to the appropriate backend server.

- **Forward Proxy:** Handles requests from clients to the internet. Clients send requests to the forward proxy, which forwards those requests to the internet on behalf of the clients.

- **Load Balancing:** Primarily focuses on distributing incoming traffic across multiple servers to optimize performance and ensure high availability.

Nginx as Reverse Proxy and Load Balancer

We need to modify **nginx.conf** file to configure it as reverse proxy and load balancing



```
events { }
http {
    upstream app {
        server app1:5000;
        server app2:5000;
    }
    server {
        listen 80;
        location / {
            proxy_pass http://app;
        }
    }
}
```

Task

- Lets create a simple load balancing example for a web application through
- Nginx. We can automate this whole process through **docker-compose**!