(슬라이드 1) 안녕하십니까 오늘 NGINX에 대해 발표하게 된 소프트웨어학과 22학번 이이슬입니다.

(슬라이드 2) NGINX설명에 앞서 웹 동작 원리에 대해 보도록 하겠습니다.   
이 페이지를 보시면 웹 클라이언트, DNS, 웹서버, WAS, 데이터베이스가 있는데요.  
여기에 대해 간략히 설명 드리면 클라이언트는 서비스를 요청하는 요청자, DNS는 도메인 주소를 IP로 변환하여 제공하는 시스템,   
웹서버는 HTML, CSS, 자바스크립트, 이미지 파일과 같은 정적 파일을 제공하는 소프트웨어이고, 오늘 주제인 NGINX도 대표적인 웹 서버 중 하나입니다.  
그리고, WAS는 동적인 처리를 담당하고 제공하며, 데이터베이스는 정보를 저장, 관리, 검색할 수 있는 시스템입니다.

이제 동작 순서를 보자면 사용자가 검색창에 특정 사이트 주소를 입력하면,   
클라이언트가 DNS에게 사이트 주소를 요청하게 되고, DNS는 IP 주소를 제공합니다.   
그리고 받은 IP주소를 이용해 클라이언트가 서버에게 html 문서를 요청합니다.   
하지만 웹 서버는 바로 웹페이지를 공급하지 못하고, 이와 같이 웹 애플리케이션 서버에 동적인 작업을 요청하고, 웹 애플리케이션 서버는 데이터베이스에 저장된 데이터를 받아 작업처리결과를 웹 서버로 보냅니다.   
그 뒤 웹 서버가 클라이언트에게 정적 문서들을 제공하고, 클라이언트가 화면에 웹페이지를 출력하여 사용자, 즉 저희가 웹 페이지를 볼 수 있게 됩니다!

(슬라이드 3) 다음으로 웹 서버가 발전해온 과정을 통해 NGINX 배경 및 구조에 대해 설명 드리겠습니다!  
최초로 개발된 웹 서버들 중 하나에 버그가 많아 이를 개선하고 발전시켜 개발된 것이 아파치로 1995년에 등장하였습니다.  
아파치의 구조를 보자면 클라이언트에게서 요청이 들어오면 커넥션을 형성하기 위해 프로세스를 생성합니다.  
여기 커넥션을 형성하는 과정에서 시간이 오래 걸려 PREFORK를 사용하였는데요. 이와 같이 미리 프로세스를 형성한 후, 요청이 들어오면 만들어진 프로세스를 사용하였습니다. 하지만 만들어진 프로세스가 전부 할당되었다면 새로운 프로세스를 만들어 진행되었습니다.  
이러한 구조는 개발하기 쉽다는 장점이 있어, 개발자는 다양한 모듈을 만들어 서버에 빠르게 기능을 추가할 수 있었습니다.

(슬라이드 4) 하지만 1999년 컴퓨터가 많이 보급됨에 따라 클라이언트의 요청이 많아져 서버에 동시에 연결된 커넥션도 많아 더 이상 새로운 커넥션을 생성하지 못하게 되었는데 이를 C10K 문제라고 합니다.  
앞에서 구조를 보았듯이 커넥션이 연결될 때마다 프로세스를 생성하는데 프로세스란 컴퓨터 안에서 계속 실행 중인 프로그램이기 때문에 메모리를 계속 차지하게 되어 메모리 부족 문제가 있고, 프로세스간 작업을 진행하기 위해 CPU에서 context switching이 계속해 발생되어 CPU에 부담이 많아지는 문제가 있었습니다.

(슬라이드 5) 이러한 아파치 서버의 문제를 보안하기 위해 NGINX가 나왔는데요. 당시 NGINX가 아파치 서버를 대체하기 위한 것이 아닌, 보완하기 위한 목적으로 나와 이와 같이 아파치 서버가 감당해야 했던 동시 커넥션을 감당하면서 정적 파일에 대한 요청은 스스로 처리하고 클라이언트로부터 동적 파일 요청을 받았을 때만 아파치 서버와 커넥션을 형성하여 아파치 서버의 부하를 줄여줍니다.   
그렇다면 NGINX가 어떤 구조를 가지고 있기에 동시 커넥션을 감당할 수 있는지 한번 알아보겠습니다.

(슬라이드 6) NGINX 구조는 2가지 프로세스를 가지고 있는데 일단 처음으로 Master Process가 있습니다. 이 프로세스는 config파일을 읽어 worker process를 생성하고 업데이트를 합니다.  
생성된 worker process는 실제로 일을 하는 프로세스로, 만들어질 때 지정된 listen(리슨)소켓을 배정받습니다. 이와 같이 소켓에 새로운 클라이언트의 요청이 들어오면 connection을 형성하고 처리합니다.  
그리고 connection은 정해진 접속 대기 시간만큼 유지되나, 형성된 커넥션으로부터 아무런 요청이 없다면 새로운 커넥션을 형성하거나 이미 만들어진 다른 커넥션으로부터 들어온 요청을 처리하는 등 워커 프로세스가 커넥션 하나만 담당하지는 않습니다.  
NGINX에서는 이러한 커넥션 형성과 제거, 그리고 새로운 요청을 처리하는 것을 이벤트라고 합니다.  
그리고 이 이벤트들을 OS커널이 큐형식으로 워커프로세스에게 전달하고, 이벤트들은 큐에서 비동기 상태로 대기하게 됩니다.  
이와 같이 이벤트를 기반으로 동작하며, 비동기 상태를 이용한 방식을 Event-Driven(이벤트 드리븐)방식이라고 하고 NGINX는 event-driven방식으로 동작한다고 할 수 있습니다.

(슬라이드 7) 마지막으로 오늘 내용 요약 정리하겠습니다. 웹서버란 클라이언트의 요청에 따라 정적 파일을 응답하여 제공하는 소프트웨어이고 대표적인 웹서버로는 아파치, 엔진엑스가 있는데 아파치는 요청당 프로세스가 처리하여 안전성, 확장성, 호환성에서 우세하나 많은 접속자들 대응 문제가 있었고 이를 보완하고자 엔진엑스가 등장하였고 event-driven 방식으로 동작하여 많은 접속자 대응이 가능했습니다. 그리고 제가 지금 어떤 웹 서버의 점유율이 높을지 궁금하여 조사해 보았는데 2024년 3월 기준 아파치가 조금 높았습니다. 저는 엔진엑스가 아파치의 문제점을 보완하고자 나와 엔진엑스의 점유율이 높을 줄 알았는데 예상과 달라 한번 넣어봤습니다. 이상으로 발표를 마치도록 하겠습니다. 감사합니다.