Django 보조자료

Django 설치-1

1. 가상 환경 생성 (venv는 가상 환경 이름, 원하는 이름으로 변경 가능)

python -m venv venv

2. 가상 환경 활성화

Windows

.₩venv₩Scripts₩activate

Linux/macOS

source venv/bin/activate

Django 설치-2

pip를 최신 버전으로 업데이트하는 것도 좋습니다.
python -m pip install --upgrade pip

Django 설치
pip install Django
python -m django –version

Django –프로젝트, App 생성

<프로젝트명>은 원하는 이름으로 지정 (예: myproject)
django-admin startproject <프로젝트명>

예시:
django-admin startproject mysite

프로젝트 폴더로 이동
cd mysite

앱 생성 (<앱 이름>은 원하는 이름으로 지정, 예: main)
python manage.py startapp <앱 이름>

예시:
python manage.py startapp main

Django –프로젝트, App 생성

1.생성한 앱을 Django 프로젝트에 등록해야 합니다.

mysite/settings.py 파일을 엽니다.

INSTALLED_APPS 리스트에 생성한 앱 이름('main')을 추가합니다.

```
# mysite/settings.py

INSTALLED_APPS = [
    # ... (기존 설정)
    'main',  # 새로 생성한 앱 이름 추가
    # ...
]
```

Django –프로젝트, App 생성

2.View(뷰)작성(main/views.py)

mysite/views.py

간단한 “Hello, Django!” 메시지를 반환하는 함수를 작성합니다.

main/views.py

```
from django.http import HttpResponse

def index(request):
    # HTTP 요청을 받아 HTTP 응답을 반환합니다.
    return HttpResponse("<h1>Hello, Django! Welcome to my first page.</h1>")
```

Django –프로젝트, App 생성

3.URL 매핑(URLconf)

mysite/views.py

사용자의 URL요청과 작성한 View함수를 연결하는 작업입니다.

3-1 앱 RULconf 생성 및 설정(main/urls.py)

앱폴더(main/)내부에 urls.py파일을 생성하고 View 함수를 연결합니다.

main/urls.py (새로 생성)

```
from django.urls import path
from . import views

# URL 패턴 리스트
urlpatterns = [
    # 사용자가 'http://.../main/'으로 요청하면 views.index 함수가 실행됩니다.
    path(' ', views.index, name='index'),
]
```

Django –프로젝트, App 생성

3.URL 매핑(URLconf)

3-2 프로젝트 URLconf 설정(mysite/urls.py)

프로젝트 레벨의 urls.py에 앱의 URLconf를 연결

```
# mysite/urls.py

from django.contrib import admin
from django.urls import path, include  # 'include'를 import 함
urlpatterns = [
    path('admin/', admin.site.urls),
    # 'http://.../main/'로 시작하는 모든 요청을 main 앱의 urls.py로 보냄
    path('main/', include('main.urls')),
]
```

Django –프로젝트, App 생성

4.서버 실행

프로젝트 폴더(mysite/)에서 개발 서버를 실행3-2 프로젝트 URLconf 설정(mysite/urls.py)

python manage.py runserver

서버가 성공적으로 실행되면 터미널에 다음과 같은 메시지 출력

Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C.

5.동작확인

웹브라우저 열고 다음 주소 접속

Http://127.0.0.1:8000/main/

Django –프로젝트, App 생성

6.요약 명령어

파일 / 명령어	역할
pip install django	Django 프레임워크 설치
django-admin startproject	Django 프로젝트 초기 폴더 생성
python manage.py startapp	프로젝트 내부에 기능 모듈(앱) 생성
settings.py	프로젝트의 전반적인 환경 설정 (앱 등록)
views.py	사용자의 요청을 처리하는 함수 작성 (핵심 로직)
urls.py	URL 경로와 View 함수 연결 (라우팅)
python manage.py runserver	개발용 웹 서버 실행

Django 실습 추가 (CRUD)

- 사용자가 볼 수 있는 index.html 추가
  - 앱 main/views.py 에서 index.html 추가
    - from django.shortcuts import render

    - # index.html로 연결해주는 index 함수
    - def index(request):
    - return render(request, 'main/index.html’)

    - ……

NGINX

- NGINX는 **이벤트 기반(Event-Driven)**, **비동기(Asynchronous)**, **비차단(Non-blocking)** 모델을 기반으로 하는 **마스터-워커(Master-Worker) 아키텍처**로 구성되어 있어, 적은 자원으로도 **높은 동시성(Concurrency)**과 **고성능**을 제공하는 데 강점

구성 요소	역할	상세 설명
마스터 프로세스 (Master Process)	**관리자 역할**	설정 파일 읽기 및 유효성 검사, 소켓 생성/바인딩/종료, 워커 프로세스 관리(시작, 종료, 유지), 무중단 설정 재로딩 및 바이너리 업그레이드 제어 등을 담당
워커 프로세스 (Worker Process)	**실제 요청 처리**	클라이언트의 실제 요청을 처리하는 자식 프로세스입니다. 일반적으로 **CPU 코어 수**만큼 설정하는 것이 효율적이며, 각 워커는 **단일 스레드** 방식으로 동작
이벤트 모듈 (Event Module)	**비동기 이벤트 처리 엔진**	워커 프로세스 내부에서 동작하며, 비동기 이벤트 처리 엔진(예: epoll, kqueue)을 사용하여 **Non-blocking I/O** 방식으로 수많은 동시 연결을 효율적으로 처리

NGINX

- **웹 서버 (Web Server):**

  HTML, CSS, JavaScript, 이미지 등 정적 파일을 효율적으로 클라이언트에게 제공합니다.

- **리버스 프록시 (Reverse Proxy):**

  클라이언트의 요청을 받아 백엔드 서버(예: 톰캣, Node.js)로 전달하고, 그 결과를 다시 클라이언트에게 반환 ,이를 통해 백엔드 서버의 정보를 숨기고 보안을 강화

- **로드 밸런서 (Load Balancer):**

  여러 백엔드 서버에 트래픽을 분산하여 서버 부하를 줄이고 안정성 증대

- **HTTP 캐시 (HTTP Cache):**

  자주 요청되는 콘텐츠를 캐시하여 응답 속도를 향상하고 백엔드 서버의 부하를 감소

- **SSL/TLS 터미네이션:**

  클라이언트와의 암호화된 연결(HTTPS)을 처리하여 백엔드 서버의 부담 경감

NGINX

•Nginx 설정은 **디렉티브(Directive)**와 **컨텍스트(Context)**로 구성됩니다. 디렉티브는 세미콜론(;)으로 끝나며, 블록 디렉티브는 중괄호({})를 사용하여 **컨텍스트 생성**

```
# main 컨텍스트 (전역 설정)
user  nginx;
worker_processes  auto; # CPU 코어 수에 맞게 설정 권장

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    # events 컨텍스트 (연결 처리 방식 설정)
    worker_connections  1024; # 워커 프로세스당 최대 연결 수
}

http {
    # http 컨텍스트 (HTTP 트래픽 관련 전역 설정)
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    # ... 다른 HTTP 설정 (로그, 압축 등)

    server {
        # server 컨텍스트 (가상 호스트, 특정 도메인 또는 포트 설정)
        listen 80;
        server_name example.com www.example.com;

        location / {
            # location 컨텍스트 (특정 URL 경로 처리 방식 설정)
            root   /usr/share/nginx/html;
            index  index.html;
        }

        # ... 다른 location 블록
    }

    # ... 다른 server 블록
}
```

NGINX

- 기본적인 웹 서버 설정(정적 파일 제공)
  - 가장 기본적인 웹 서버 설정은 특정 도메인으로 들어온 요청에 대해 **정적 파일**을 제공

```
server {
    listen 80; # 80번 포트로 수신
    server_name mywebsite.com; # 도메인 이름 지정

    location / {
        root   /var/www/mywebsite; # 웹 파일의 실제 경로 지정
        index  index.html index.htm; # 기본 파일 지정

        # 파일이 없을 경우 404 에러 반환
        try_files $uri $uri/ =404;
    }

    # 에러 로그와 액세스 로그 설정
    error_log /var/log/nginx/mywebsite_error.log;
    access_log /var/log/nginx/mywebsite_access.log;
}
```

NGINX

- 리버스 프록시 및 로드 밸런싱
- 리버스 프록시 설정(Reverse Proxy)
  - Nginx를 사용하여 클라이언트의 요청을 **백엔드 애플리케이션 서버**

```
server {
    listen 80;
    server_name api.mywebsite.com;

    location / {
        # 요청을 8080 포트에서 실행 중인 백엔드 서버로 전달
        proxy_pass http://127.0.0.1:8080;

        # 백엔드 서버로 원본 요청 정보를 전달하기 위한 헤더 설정
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

NGINX

- 로드 밸러신 설정(Load Balancing)
  - upstream 블록을 사용하여 여러 백엔드 서버 그룹을 정의하고, server 블록에서 이 그룹으로 요청을 분산

`/etc/nginx/nginx.conf`의 `http` 블록 또는 별도의 설정 파일에 다음 내용을 추가합니다.

Nginx

```
http {
    # ... 기존 설정

    # 백엔드 서버 그룹 정의 (backend_servers라는 이름 사용)
    upstream backend_servers {
        # 라운드 로빈 (기본): 서버에 요청을 순서대로 분배
        server 192.168.1.10:8080;
        server 192.168.1.11:8080;

        # 다른 로드 밸런싱 알고리즘 (주석 해제하여 사용 가능)
        # least_conn; # 가장 적은 활성 연결을 가진 서버에 요청
        # ip_hash;    # 클라이언트 IP 해시를 기반으로 서버 할당 (세션 유지에 유용)
    }

    # ... server 블록 시작
}
```

NGINX

- 로드 밸러신 설정(Load Balancing)
  - Server블럭에서 Upstream 사용

```
server {
    listen 80;
    server_name loadbalancer.mywebsite.com;

    location / {
        # 요청을 위에서 정의한 upstream 그룹으로 전달
        proxy_pass http://backend_servers;

        # 프록시 헤더 설정 (필수)
        proxy_set_header Host $host;
        # ... 기타 proxy_set_header 설정
    }
}
```

NGINX

•설정 적용 및 관리 명령어

명령어	기능	설명
sudo systemctl start nginx	Nginx 시작	Nginx 서비스를 시작합니다.
sudo systemctl stop nginx	Nginx 중지	Nginx 서비스를 중지합니다.
sudo systemctl restart nginx	Nginx 재시작	Nginx를 완전히 중지 후 다시 시작합니다.
sudo systemctl reload nginx	설정 다시 로드	**현재 요청을 유지한 채** 새로운 설정만 적용합니다. **운영 중**에 가장 권장되는 방법입니다.
sudo nginx -t	설정 파일 테스트	설정 파일에 문법 오류가 없는지 확인합니다. 재시작/리로드 전에 **반드시** 실행해야 합니다.

NGINX

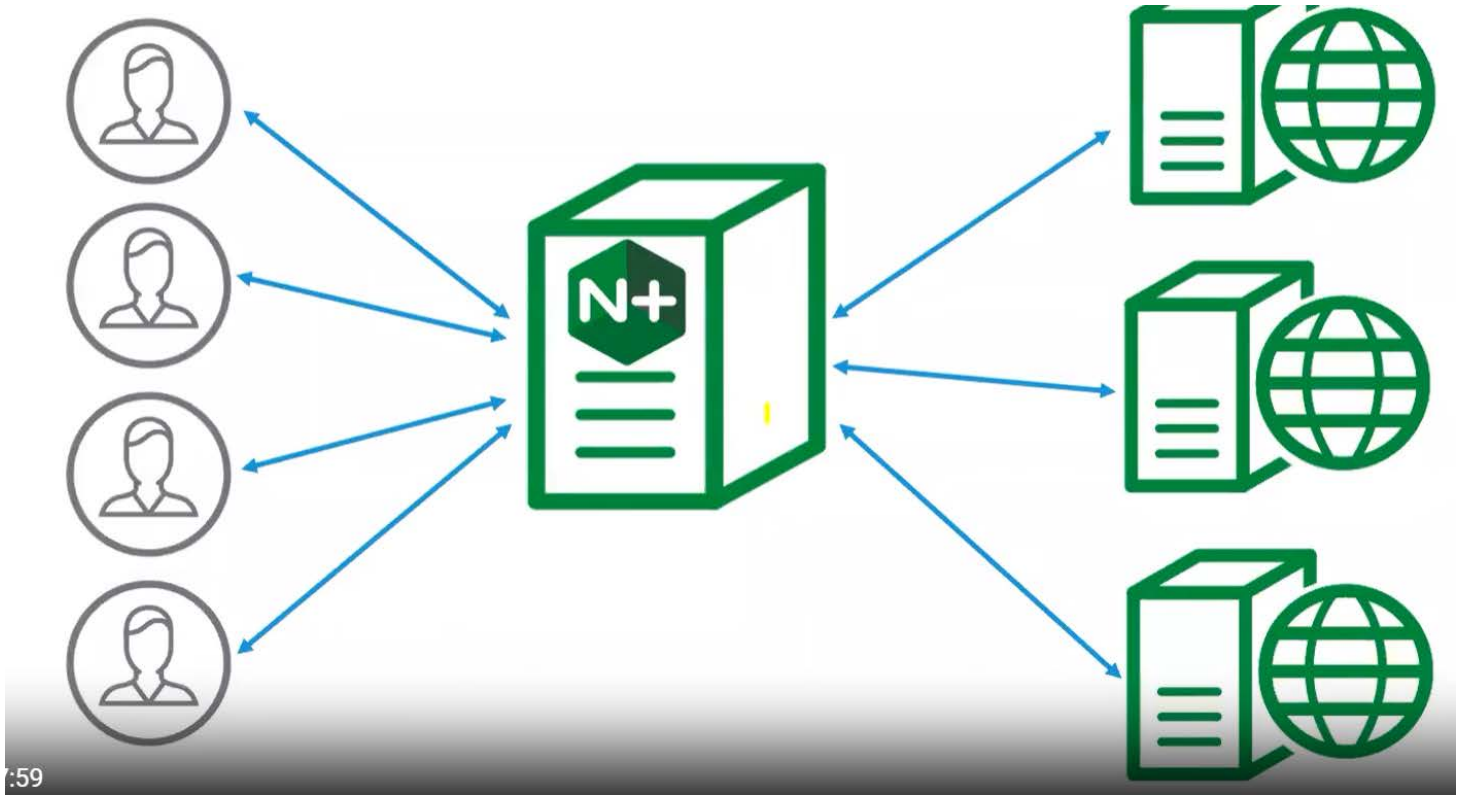
- Open Source

Features
Load Balancer (HTTP/TCP/UDP & Layer 7 request routing)
Content Cache (Static/Dynamic)
Web Server/Reverse Proxy
Security Controls/Rate Limiting
Monitoring
Web Application Firewall

https://www.nginx.com/products/nginx/#compare-versions

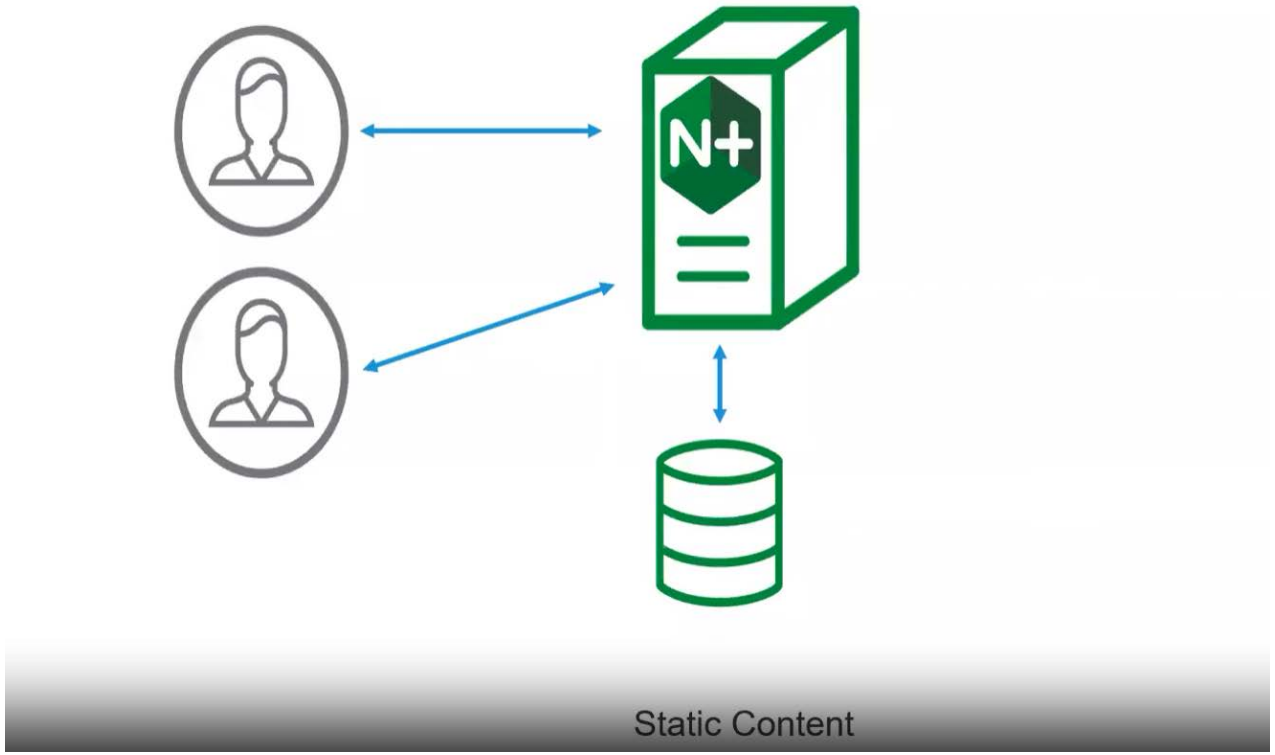
NGINX

- Load balancer



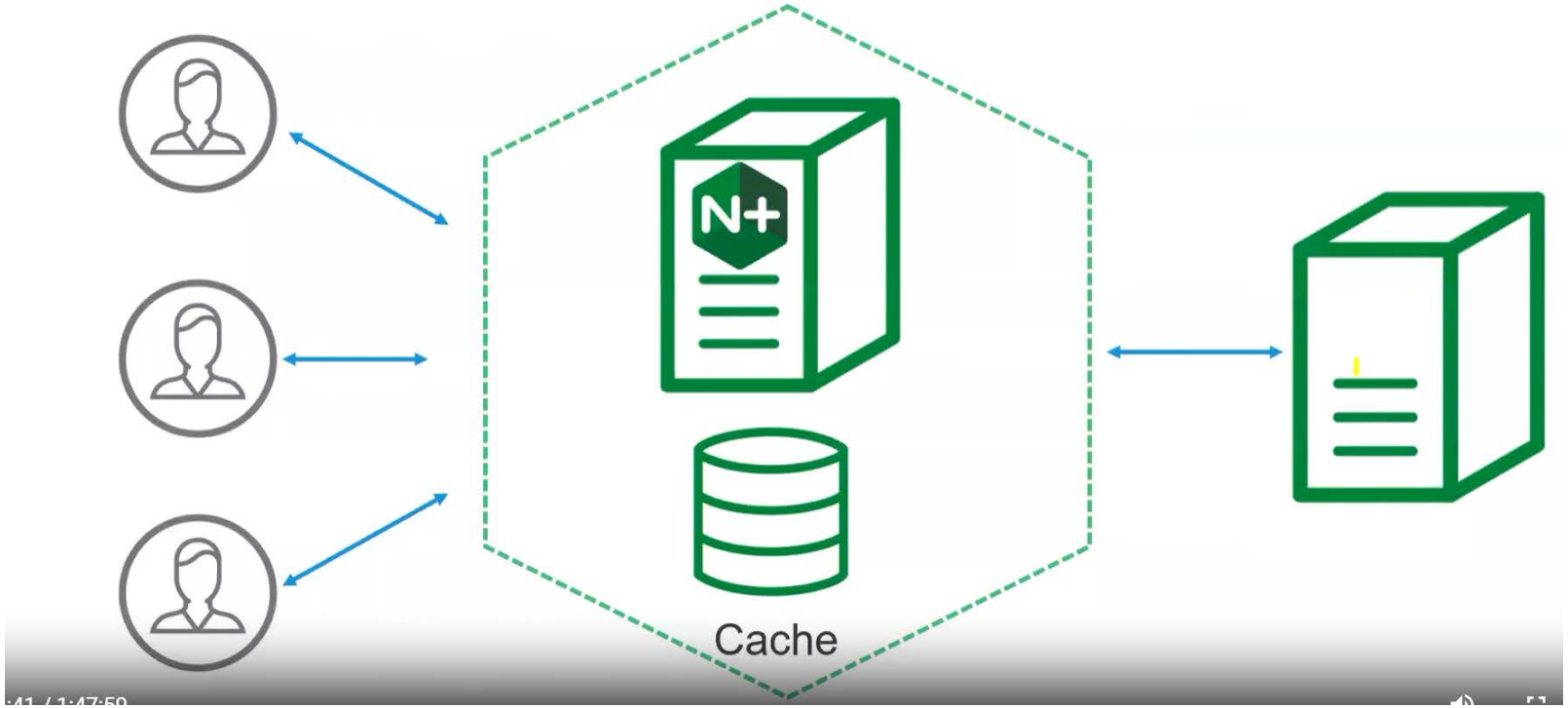
NGINX

- Nginx as WebServer



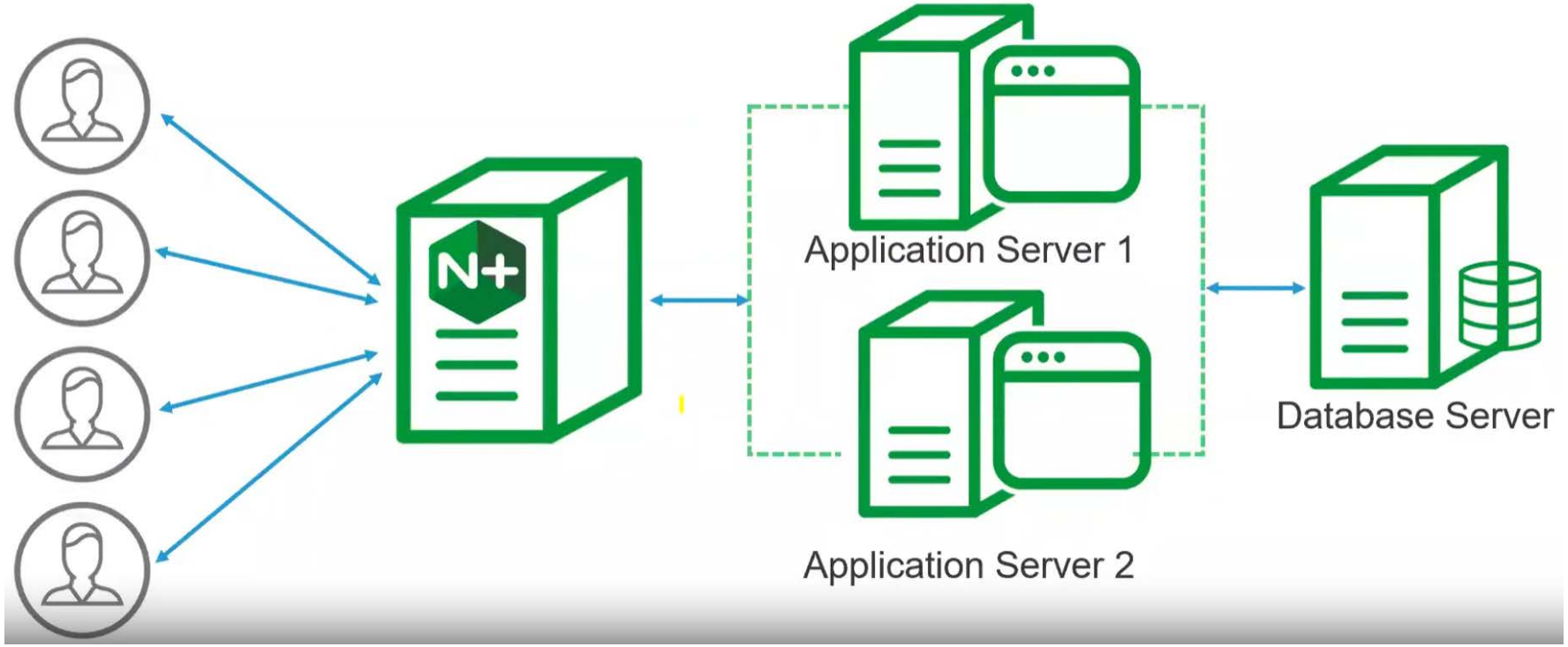
NGINX

- Nginx as a Contents Cache



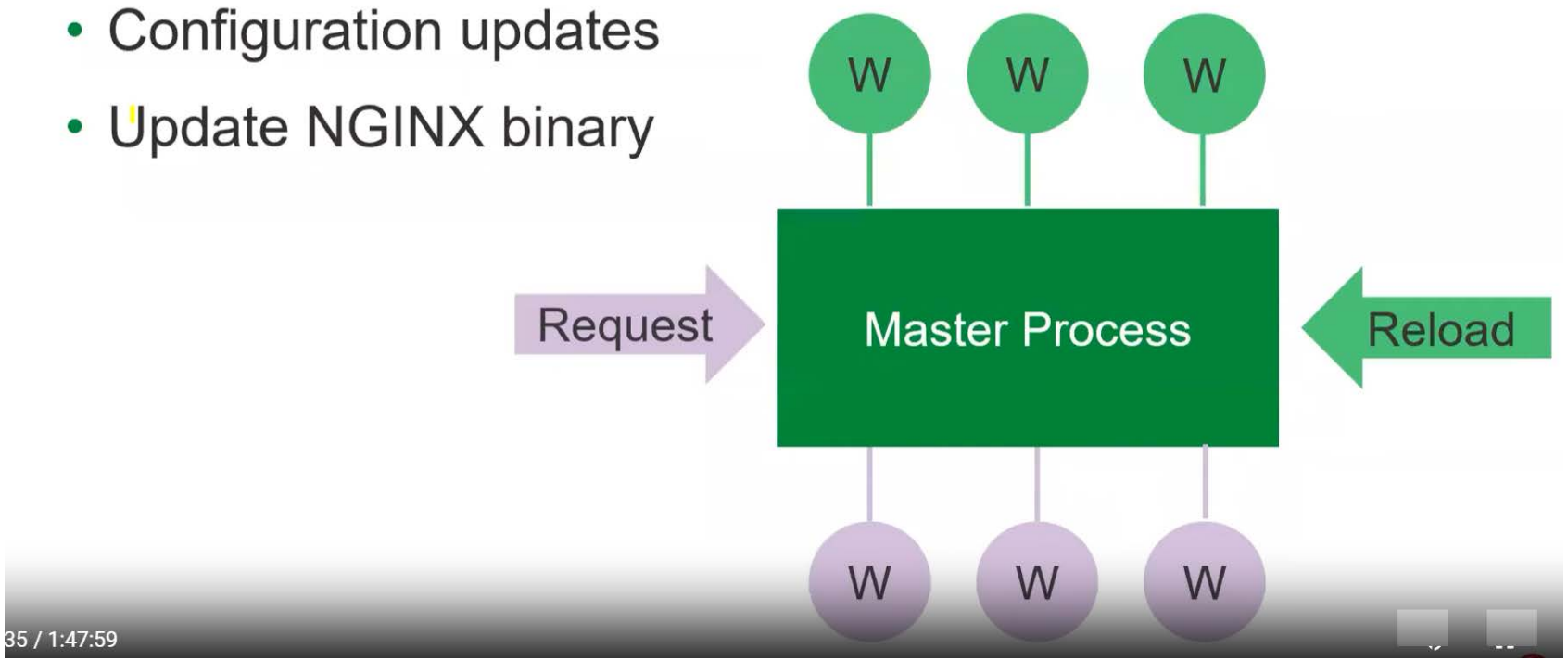
NGINX

- Nginx as a Reverse Proxy Server



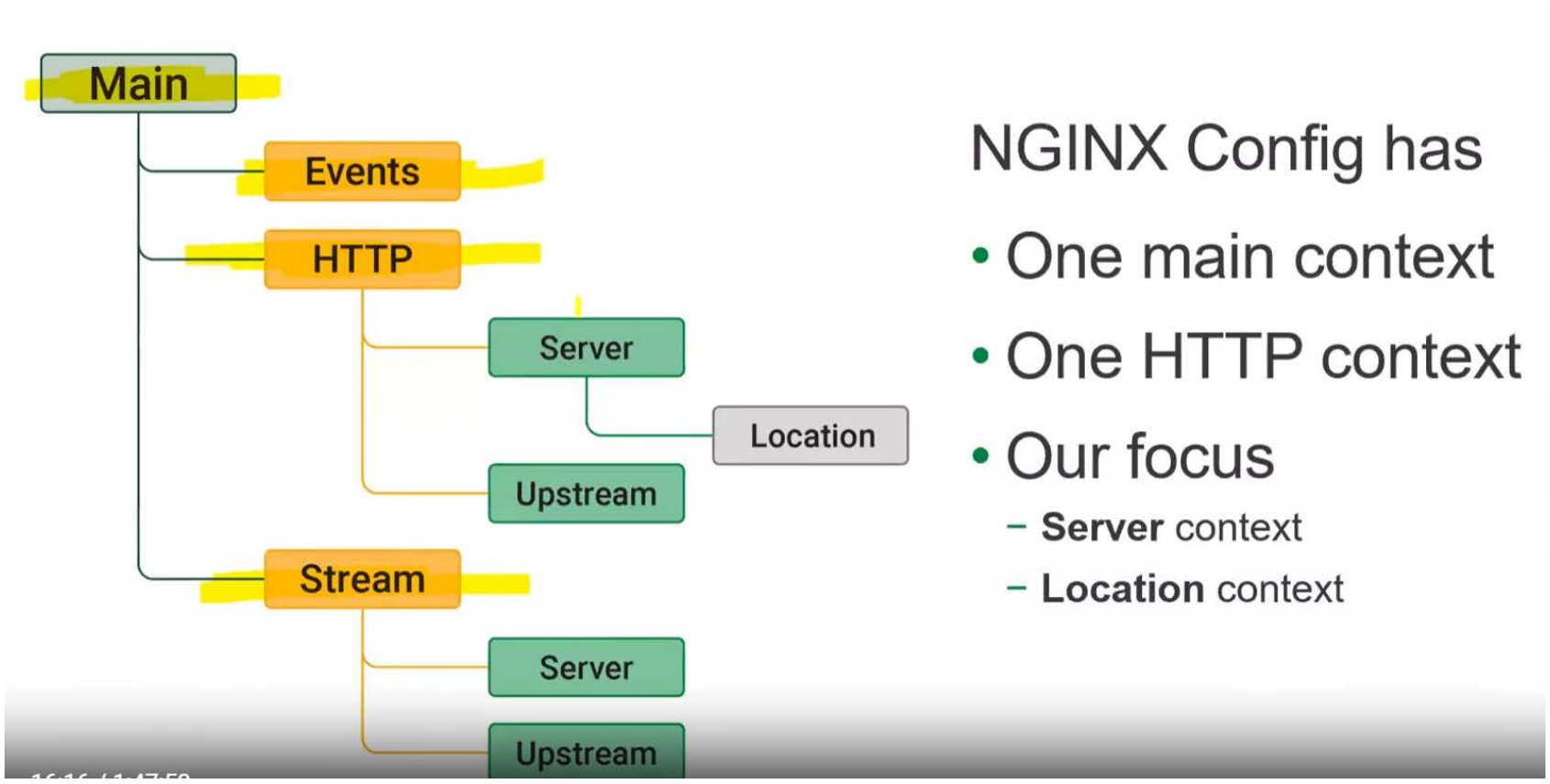
NGINX

- No Downtime



NGINX

- Configuration



NGINX

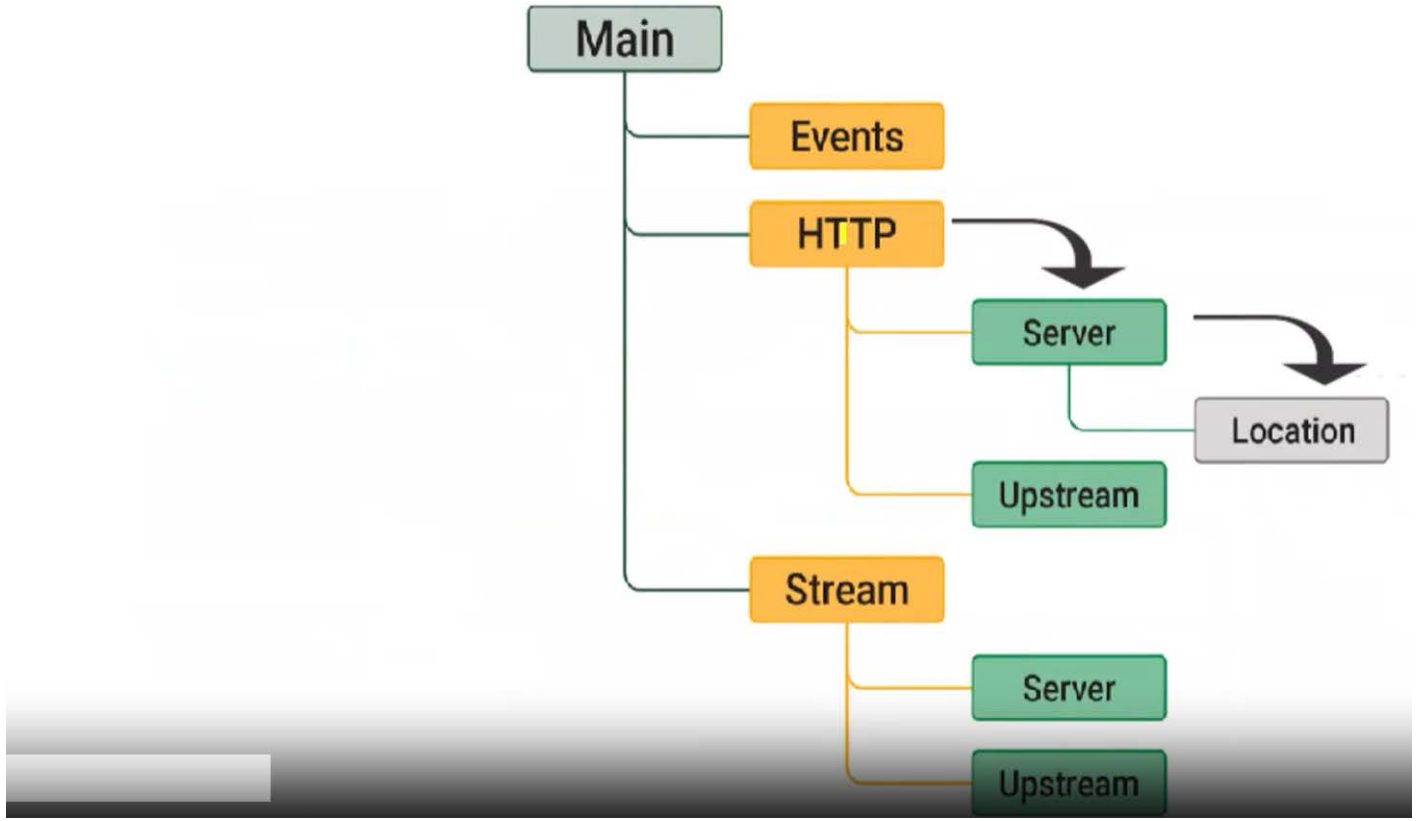
- Command
- Reload the configuration
  - nginx –s reload
- Graceful shutdown
  - nginx –s quit
- Fast shutdown
  - nginx –s stop
- Check Nginx version
  - nginx –v
- Check config file syntax
  - nginx –t
- Display current configuration(s) and check file syntax
  - nginx -T

NGINX

- Command
- Reload the configuration
  - nginx –s reload
- Graceful shutdown
  - nginx –s quit
- Fast shutdown
  - nginx –s stop
- Check Nginx version
  - nginx –v
- Check config file syntax
  - nginx –t
- Display current configuration(s) and check file syntax
  - nginx -T

NGINX

- Inheritance



NGINX

- Install

sudo apt install nginx

Nginx 시작하기

sudo systemctl start nginx

Nginx 상태 확인

sudo systemctl status nginx

부팅 시 Nginx 자동 실행

sudo systemctl enable nginx

변경사항 적용

sudo systemctl restart nginx

NGINX

- 방화벽 설정

# 방화벽 활성화
sudo ufw enable

# 80포트와 443포트를 전부 허용
sudo ufw allow 'Nginx Full'
# 80포트 허용
sudo ufw allow 'Nginx HTTP'
# 442포트 허용
sudo ufw allow 'Nginx HTTPS'

# 방화벽 변경사항 적용(서비스를 재시작해도 됨)
sudo ufw reload

# 방화벽 상태 확인
sudo ufw status