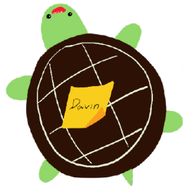
와플스튜디오 Backend Seminar[4]

변다빈 @davin111



2020.10.10.(토) 13:00 - Zoom

Assignment 2 and 3

- 과제 내용과 제출 방식을 **잘 읽자** (대충 읽지 말자. 특히 과제 2는 README에 꼭 해야할 것이 있었음)
- 제출 방식을 잘 지키고(please 🙏), 과제 내용 및 제출 방식의 **취지를 생각하자**
- 중요한 것은 이미 문서에 적어두긴 하지만, 강조할 만한 것이 단톡방에서 언급되니까 **잘 확인하자**
- 과제 기간의 길이가 길다고 벼락치기 할 생각을 가급적 하지 말자 (기간의 길이가 내용에 고려됨)
- 과제 2와 3의 경우, **시작하면 미리 해당 Pull Request를 생성하자** (진행자들이 과제 진척도와 이해도를 확인 가능)
- **git을 공부하자** (과제를 떠나 개발을 할 것이면 너무너무너무너무너무 중요하고, 인생의 효율이 달라짐)
- Issues에 좋은 내용과 질문이 꾸준히 공유되고 있음!
- 과제 2 및 3의 명세와 맞는 **React 프로젝트 제공** (https://github.com/Hank-Choi/wba2_client)

- 약 **1시간 30분의 종합 피드백 및 질의응답 시간 진행**, 현재 과제 2의 **개별 피드백 진행 중**
- 50명 → (과제 0) → 48명 → (과제 1) → 43명 → (과제 2) → 32명 → …
- 과제 2 완료한 분들의 평균 grace day 잔여량 == 4.3일 (과제 3 진행 과정에서 추가 grace day가 필요할까?)

Contents

오늘 이야기할 주제들

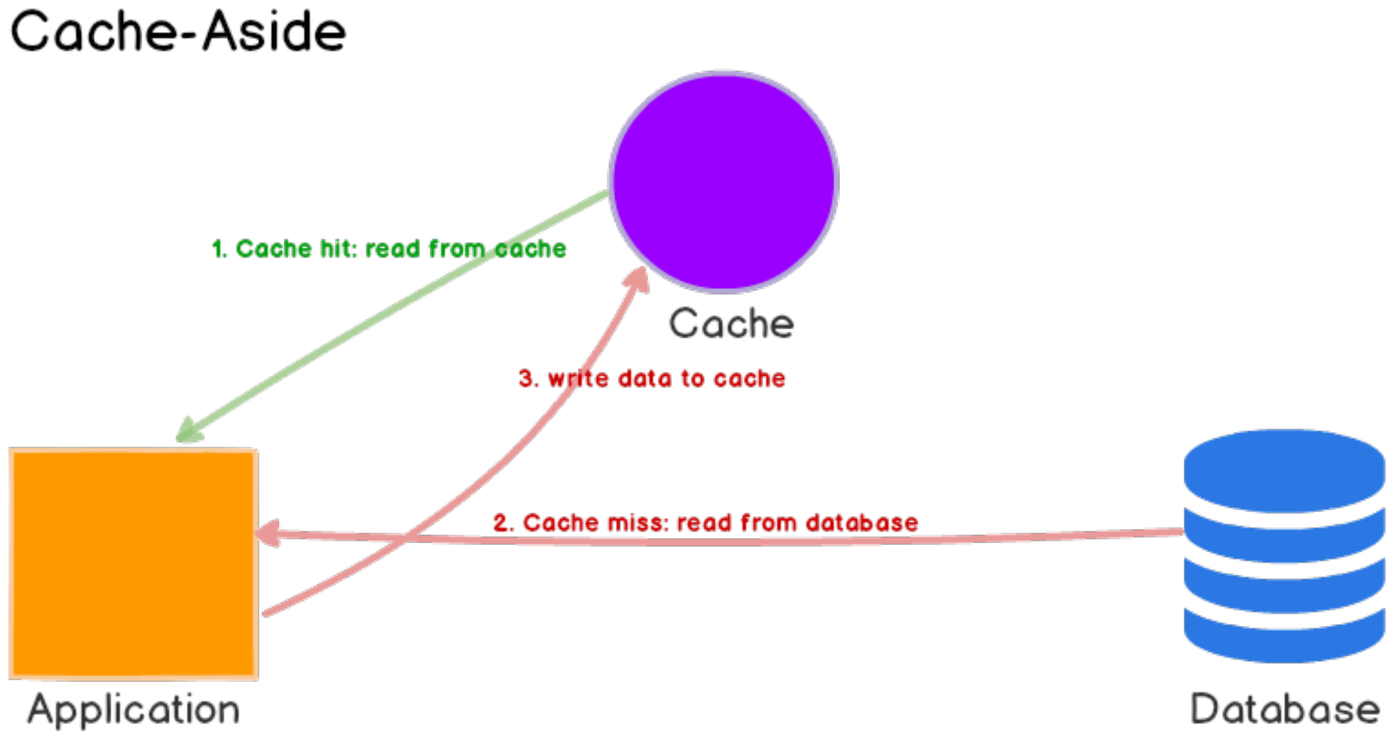
- 좀 더 현실적으로 생각하기
  - Caching (QuerySet에서, 그리고 Django CACHES의 의미에서)
  - DB의 ACID와 Transaction

- Deployment
  - 웹 서버와 웹 어플리케이션 서버
  - Nginx 설정하기와 log 보기
  - 배포 중인 서버 동작 살펴보기
  - frontend와의 결합

https://docs.djangoproject.com/en/3.1/topics/cache/

Caching
어디서든 유용한 개념

- 실제 서비스에서 안 쓰일 수가 없다
- 컴퓨터의 거의 모든 층위에서 사용되는 개념 중 하나 아닐까…



https://medium.com/@bdv111/cache-만료-시점에-생기는-부하를-mintcache를-통해-해소하기-9946bd249b4c
(~~Medium 계정을 만들고 follow와 clap하는 것을 권장~~)

Caching
어디서든 유용한 개념

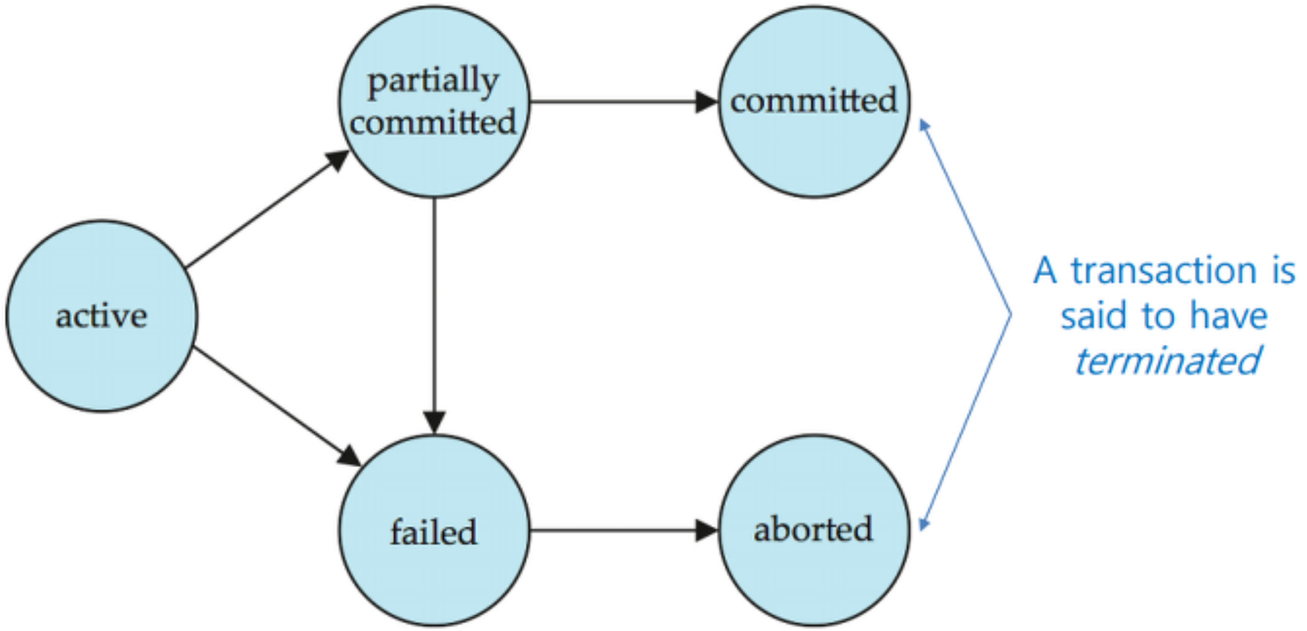
- Django에서 Local-memory, Memcached, Redis 등 다양한 방식의 caching을 지원
- 어떨 때 쓰면 좋을까?
  - 무거운 query가 자주 발생할 때
  - 잦은 요청이 발생할 때
  - 여러 사용자가 공통으로 공유하는 데이터일 때
  - client에게 주어야 할 데이터가 꼭 최신이거나 정확할 필요가 없을 때
- 실제 동작 살펴보기

https://coding-factory.tistory.com/226
https://github.com/wafflestudio/rookies/issues/114

Transaction

슬슬 존재감을 느껴야할, 사실 계속 사용하던 개념

- DB에 대한 작업의 단위
- 결과적으로 commit되거나 rollback됨



https://victorydntmd.tistory.com/129

ACID

DB transaction의 원칙

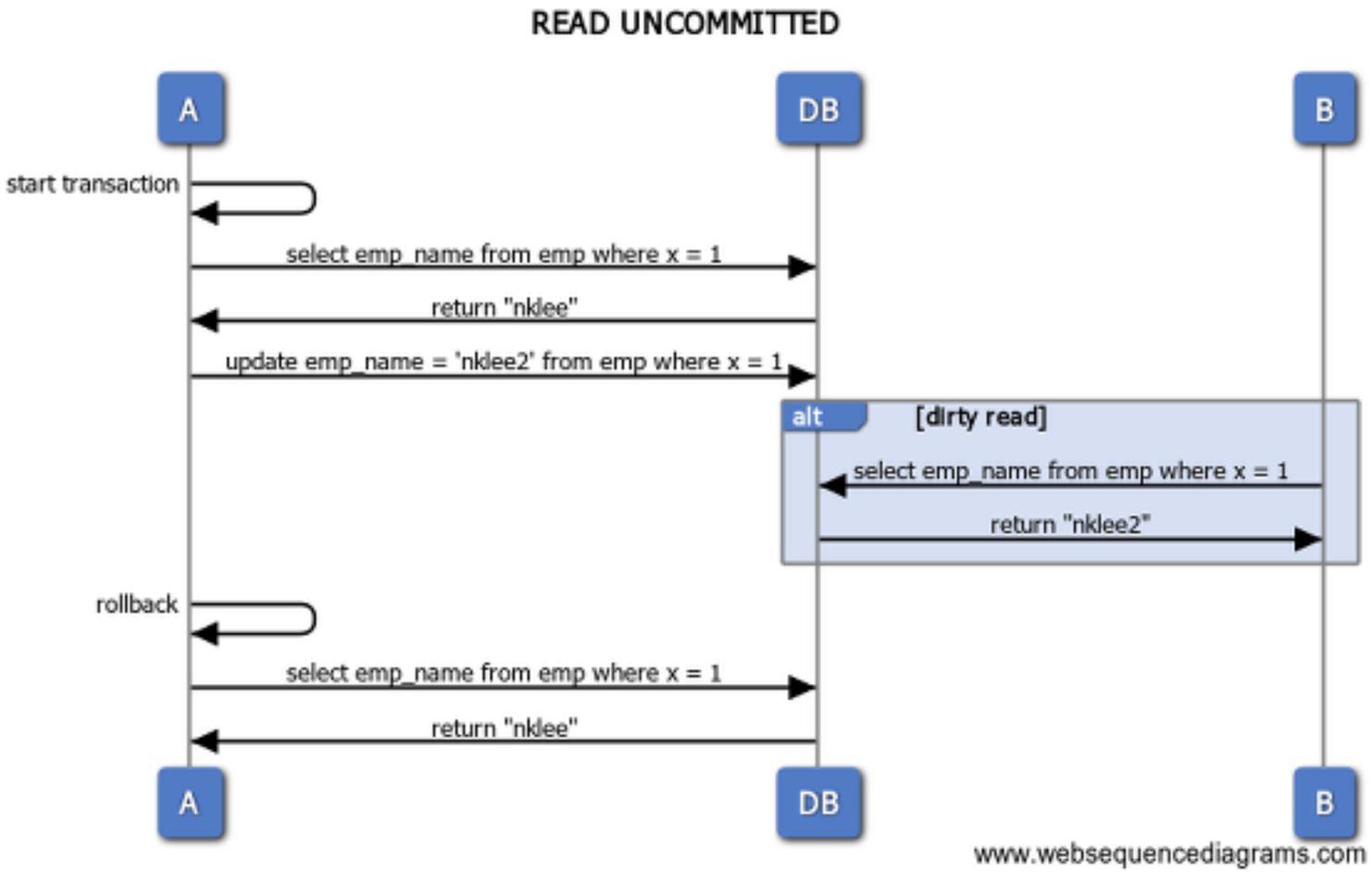
- Atomicity (원자성)
  - 다 되거나 다 안 되거나, 둘 중에 하나여야 함
- Consistency (일관성)
  - 미리 정해둔 규칙에 맞게 데이터를 저장, 변경해야 함
- Isolation (고립성)
  - 복수 개의 transaction이 실행될 때 서로 연산이 끼어들면 안 되며, 중간 과정이 이용되어도 안 됨
- Durability (영구성)
  - 일단 commit했으면 어떠한 문제가 발생해도 영구적으로 반영된 상태여야 함

https://suhwan.dev/2019/06/09/transaction-isolation-level-and-lock/
https://docs.djangoproject.com/en/3.1/ref/databases/#mysql-isolation-level
https://lng1982.tistory.com/287

Isolation Level

동시성과의 trade-off

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZBLE

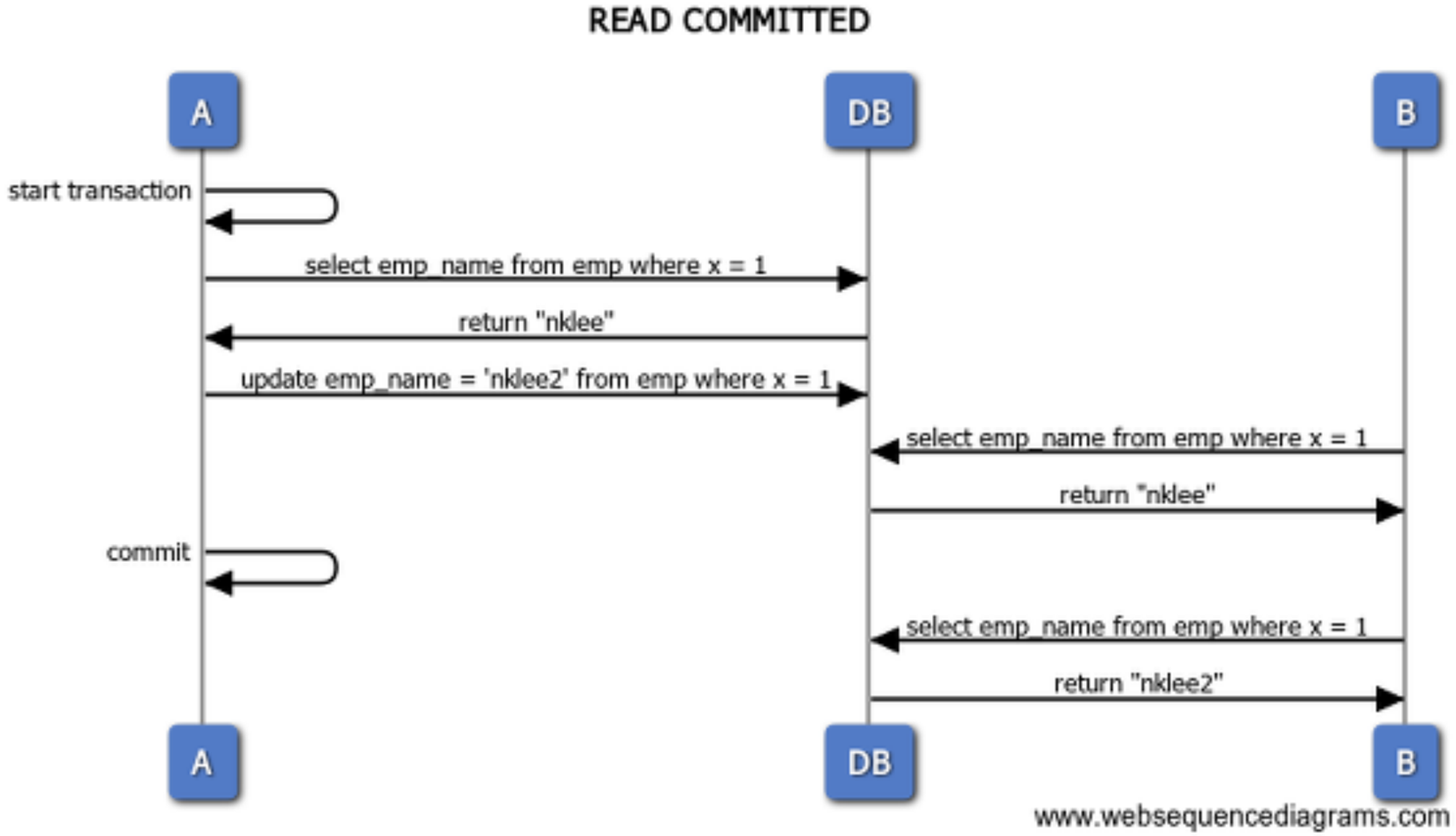


동시성이 높지만 dirty read 문제 발생 가능

Isolation Level

동시성과의 trade-off

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZBLE



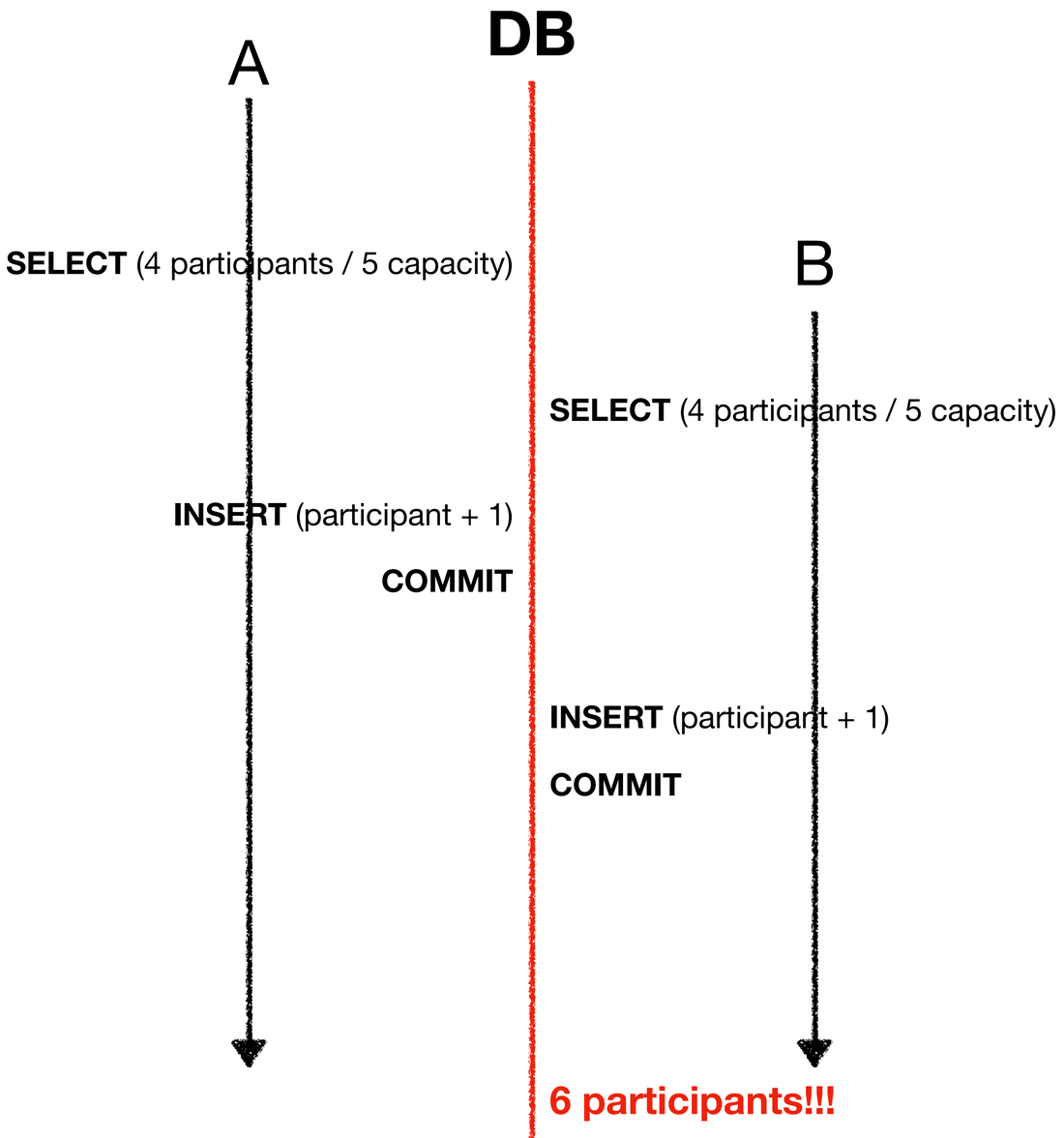
Django default isolation level (PostgreSQL)

https://docs.djangoproject.com/en/3.1/topics/db/transactions/#controlling-transactions-explicitly
https://github.com/wafflestudio/rookies/issues/223

동시성을 고려해 transaction.atomic 사용하기

Isolation Level과 설명상에서 비슷한 측면

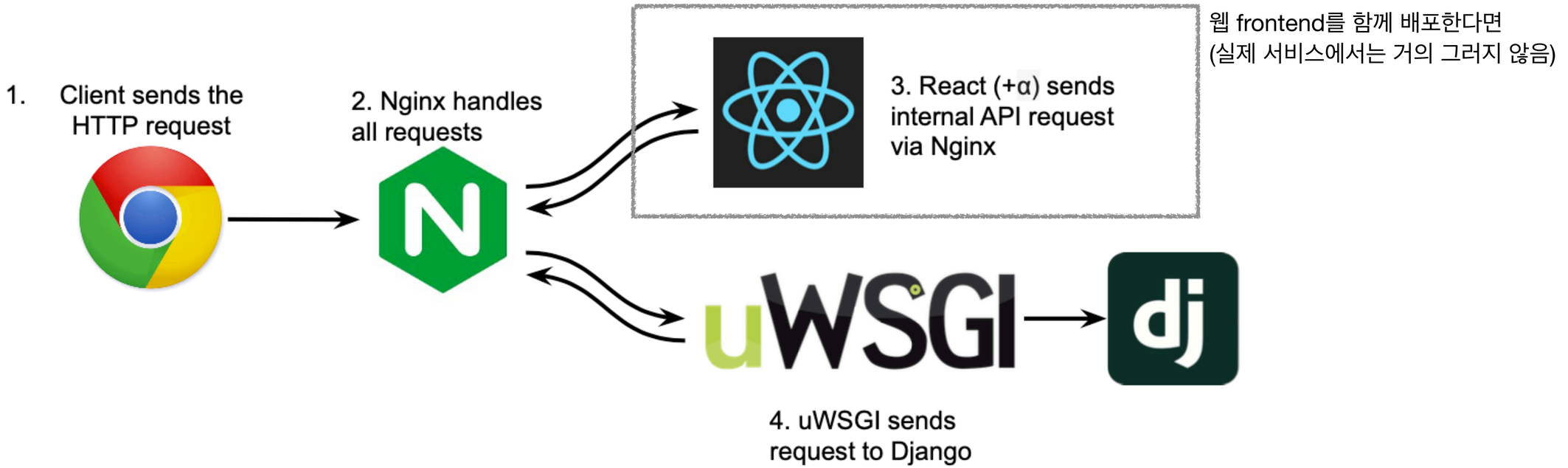
- get과 create가 별개의 transaction에 속하는 query이기에 발생할 수 있는 문제
- 문제 해결을 위해서 추가적인 고민들이 더 필요할 수 있음



https://medium.com/@bdv111/aws-ec2에서-nginx로-react-앱-직접-배포하기-c1e09639171e
(~~Medium 계정을 만들고 follow와 clap하는 것을 매우 권장~~)

Deployment

이걸 안 하면, 지금까지 해왔던 모든 것이 의미 없음



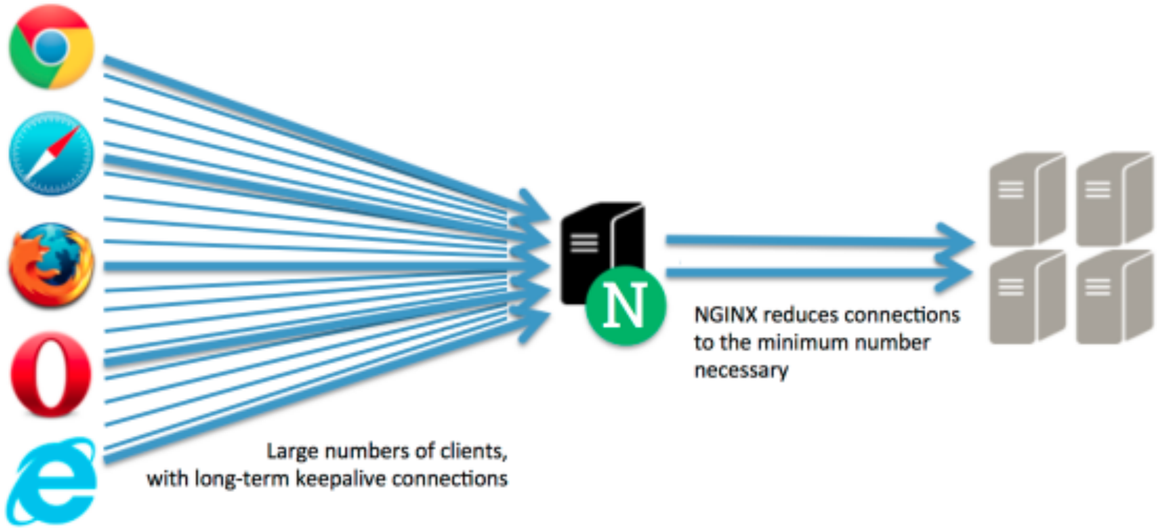
웹 frontend를 함께 배포한다면
(실제 서비스에서는 거의 그러지 않음)

하나의 컴퓨터에 backend, 웹 frontend를 모두 배포하는 상황을 가정한 그림

https://whatisthenext.tistory.com/123
https://wayhome25.github.io/django/2018/03/03/django-deploy-02-nginx-wsgi/
https://en.wikipedia.org/wiki/Unix_domain_socket
https://github.com/wafflestudio/rookies/issues/125

Nginx
웹 서버의 일종

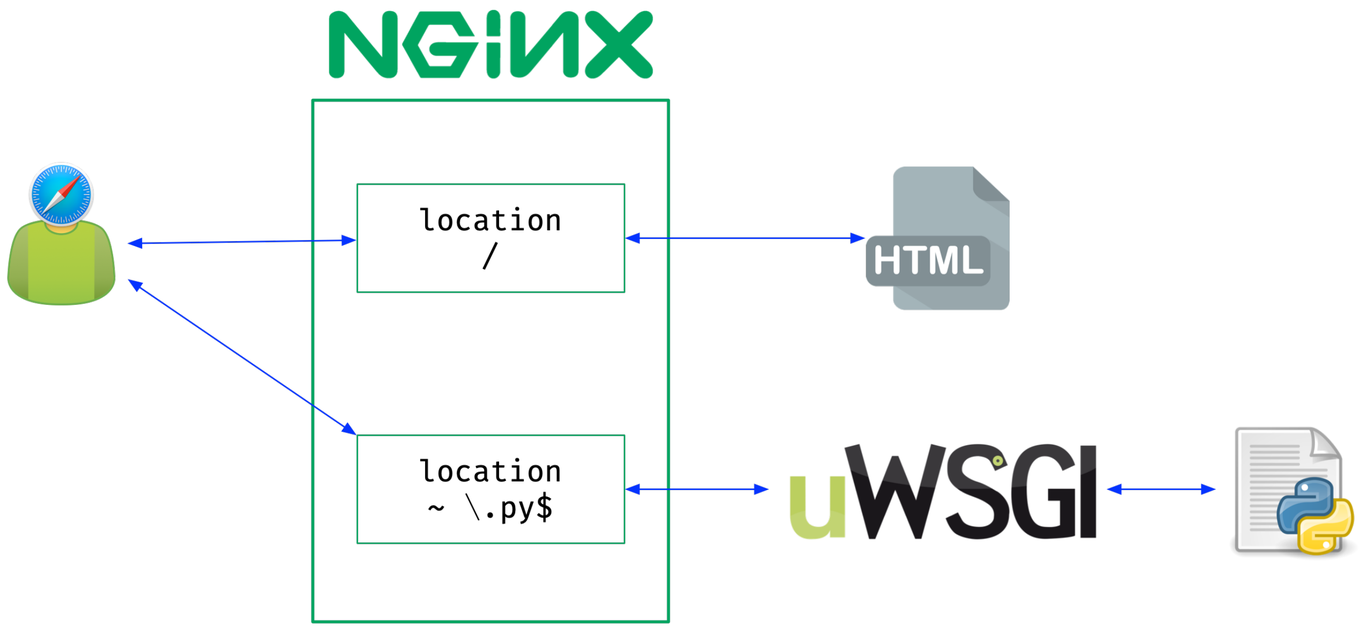
- Nginx
  - 클라이언트로부터 HTTP 요청을 받아 정적인 페이지 및 파일로 응답. 동적인 것은 uWSGI까지 이용
  - 웹 어플리케이션 서버에 앞서 요청을 관리하고 배분하여 효율적인 동작을 가능하게 함
  - 하나의 instance 내에서도 가상 호스트(서버) 개념을 사용해, 단일 IP 내에서 여러 웹 배포 가능
- unix socket: web server(Nginx)와 web application server(uWSGI)를 잇는다



https://uwsgi-docs.readthedocs.io/en/latest/
https://brownbears.tistory.com/350
https://m.blog.naver.com/wlsdml1103/221329528740

uWSGI

웹 어플리케이션 서버



- WAS (Web Application Server)
  - 웹 서버가 동적으로 기능한다면 WAS

- uWSGI
  - web application server의 일종
  - web server(Nginx)와 web application(Django) 사이에 위치
  - 통역가와 비슷한 역할이며, HTTP 요청을 Python으로, 그 반대로 통역해줌
  - WSGI(Web Server Gateway Interface)는 Python의 WAS라고 할 수 있음
  - uWSGI는 WSGI의 구현체

uWSGI

https://ko.wikipedia.org/wiki/데브옵스
https://medium.com/@bdv111/도메인을-ip-주소에-연결하는-방법과-nslookup-9e70a32eec57
(~~Medium 계정을 만들고 follow와 clap하는 것을 강력히 권장~~)

Good Deployment

배포에 대해서도 방금의 내용들은 새발의 피 정도

- Domain Name System
- HTTPS
- 배포의 자동화
- Docker
- Auto Scaling, Load Balancing 등
- AWS, Kubernetes, Azure 등의 서비스를 이용하는 방법
- Logging과 Monitoring
- …

DevOps가 관심 있다면 이쪽으로 더더더 파고들자

Any Questions?

또는 세미나장이 하고 싶은 말, 못다한 말