直播间特惠

活动团购基础上在减$300

仅限今日直播间下单

送 LintCode VIP 1年 热

+送 价值$199【递归九讲】 热

再送 FB, Google, Amazon, Microsoft等一线公司
1次**内推机会**
【仅限今日直播间下单前10位！！！】

扫码立即购课
优惠码
【FCDBD5】

早鸟优惠码【FCDBD5】

立减$1300

早鸟优惠

北京时间2021年8月1日失效!

扫码立即使用：↓

扫码立即使用：

加豆豆（JZbanbanya）咨询更多福利

扫码进群，领课件、回放等更多福利

老师介绍

九章算法班任课老师

前 A 厂全栈工程师

系统设计班任课老师

前算法竞赛国家集训队员

前 F 厂基础架构工程师

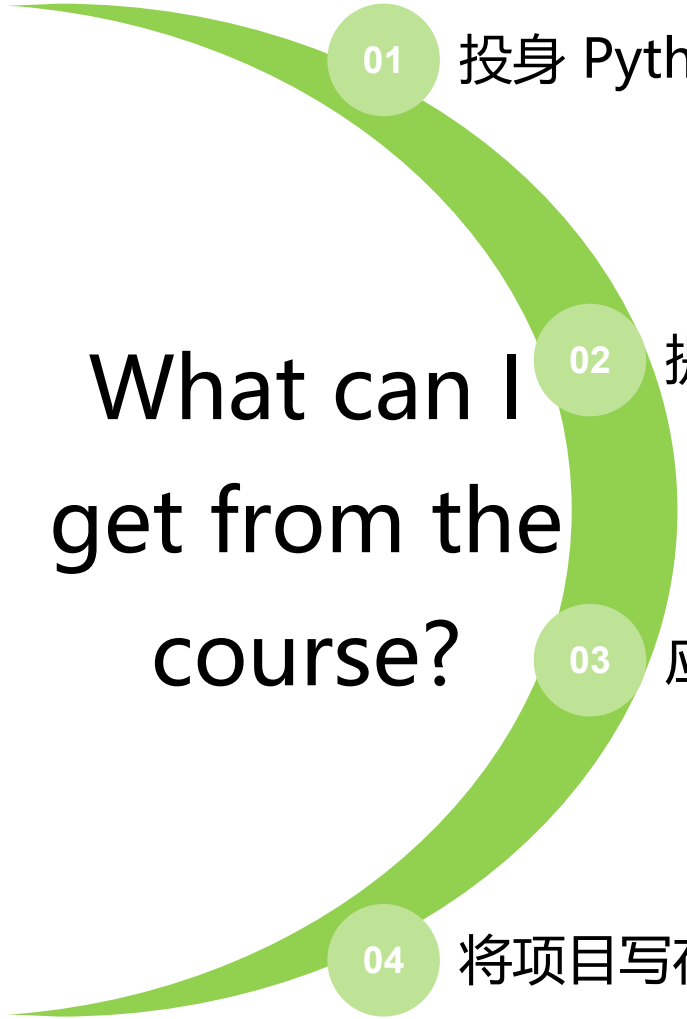
今日课程刚要

- 课程介绍（30m）
  - 学了这个课可以干嘛？
  - 在简历中如何展示这个项目？
  - 项目优势与规划
  - 独特的 Git-based 授课形式介绍
- 技术要点预览（1h）
  - 什么是 Push 写扩散（Fanout）模型
  - 什么是消息队列（Message Queue）
  - 什么是冗余存储技术（Denormalization）
- 后续流程 & QA（30m）
  - 有哪些先修知识需要学
  - 每周建议投入多少时间
  - 往期学员真实评价

第一部分 课程介绍

学了这个课可以干嘛？
在简历中如何展示这个项目？
项目优势与规划
独特的 Git-based 授课形式介绍

学了这个课程可以干嘛?



01 投身 Python Web 后端开发

02 提升自身水平到阿里 P8，Facebook E5，Google L5 的水平

03 应对在系统设计类面试中与 Web Backend 相关的各类问题

04 将项目写在简历中提升简历通过率

简历模板

Twitter 后端全栈项目

- 技术栈：Python, Django, MySQL, HBase, Redis, Memcached, RabbitMQ, Amazon S3
- 使用推 (Push) 模型实现写扩散机制的新鲜事 (news feeds)
- 使用 Memcached 和 Redis 缓存数据，优化读多写少的数据库表单
- 使用 Key-value Store - HBase 存储写多读少的数据库表单
- 使用冗余存储技术（Denormalization）记录评论数和点赞数，减少数据库查询次数
- 使用 Message Queue 传递异步任务，降低响应时间
- 涉及的代码更改 10000+ 行，耗时 3 个月

Backend Fullstack Project - Twitter

- Tech Stack: Python, Django, MySQL, HBase, Redis, Memcached, RabbitMQ, Amazon S3.
- Used push model to fanout news feeds.
- Used Redis and Memcached to reduce db queries for tables which has lot reads and less writes.
- Used Key-value store HBase to split db queries for tables which has less reads and lot writes.
- Used denormalization to store the number of comments & likes in order to reduce db queries.
- Used Message Queue to deliver asynchronized tasks to reduce response time.
- The whole project resulted in 10000 lines of code changes, cost over 3 months.

这是一个什么样的课程

从 0 -> 1 -> N 搭建起一个工业级标准的信息流系统后端

1. Python、Django、Celery
2. Redis、Memcached、Message Queue
3. MySQL、HBase
4. AWS / Aliyun Cloud Service (S3、Relational DB、Hbase、Load Balancer)
5. Rest API、Serialization、Denormalization

本课程 vs 市面上的其他网课

本课程

通过每次 100 - 200 行代码的 Git Commit 让你分步骤的了解一个可以达到工业化标准的项目是如何搭建起来的

硅谷顶尖大厂高级工程师，拥有丰富的亿级用户开发经验

通过一个项目从 P5/E3 的难度级别做到 P8/E5 的难度级别，逐步深入，真正能够实现一个工业级要求的项目

这个是 Hello World 你会了吧？来，接下来我们造坦克

通常是找不到好的工作的人去培训机构当老师

很多只是做一个 Demo 玩具，达不到工业水准，项目多但是都很浅

其他课程

课程规划

为期三个月

每周一节直播课

讲**理论和原理**

每周三节录播课

讲**代码实现**

1. 信息流系统的理论基础与先修知识
   - Pull
   - Push
   - HTTP
   - MySQL
   - Git
   - System Design
   - ORM
2. Django Web Backend 开发从 0 到 1
   - Web Framework
   - Rest API
   - Unit Tests
   - Serialization
   - Model
   - N + 1 Queries
3. 核心模块 Tweets / Frendships / NewsFeeds 的实现
   - ManyToMany
   - AliyunOSS
   - Thirdparty Package
   - AWS S3
   - Pagination
   - Cache
   - Message Queue
   - RateLimiter
   - Denormailzation
4. 其他模块 Likes / Comments / Image / Notification 的实现
5. 构建完整的工业化后端系统架构
6. 实现云端分布式的后端系统系统
   - AWS
   - Aliyun
   - Load Balancer
   - NoSQL

第二部分 技术要点预览

什么是 Push 写扩散（Fanout）模型
什么是消息队列（Message Queue）
什么是冗余存储技术（Denormalization）

什么是信息流系统

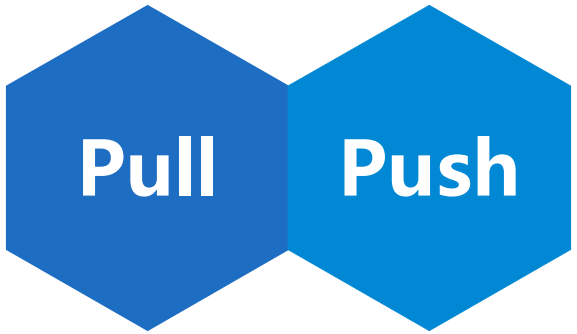
在我们的生活中，或多或少都会接触到一些信息流系统，但是你知道它是如何实现的吗？这也是系统设计面试中的高频题！

技术难点

接下来我将针对信息流系统设计到的几个技术难点进行理论分析和项目代码展示，让大家了解到我们将在这个课程中学到哪些有意思的技术知识

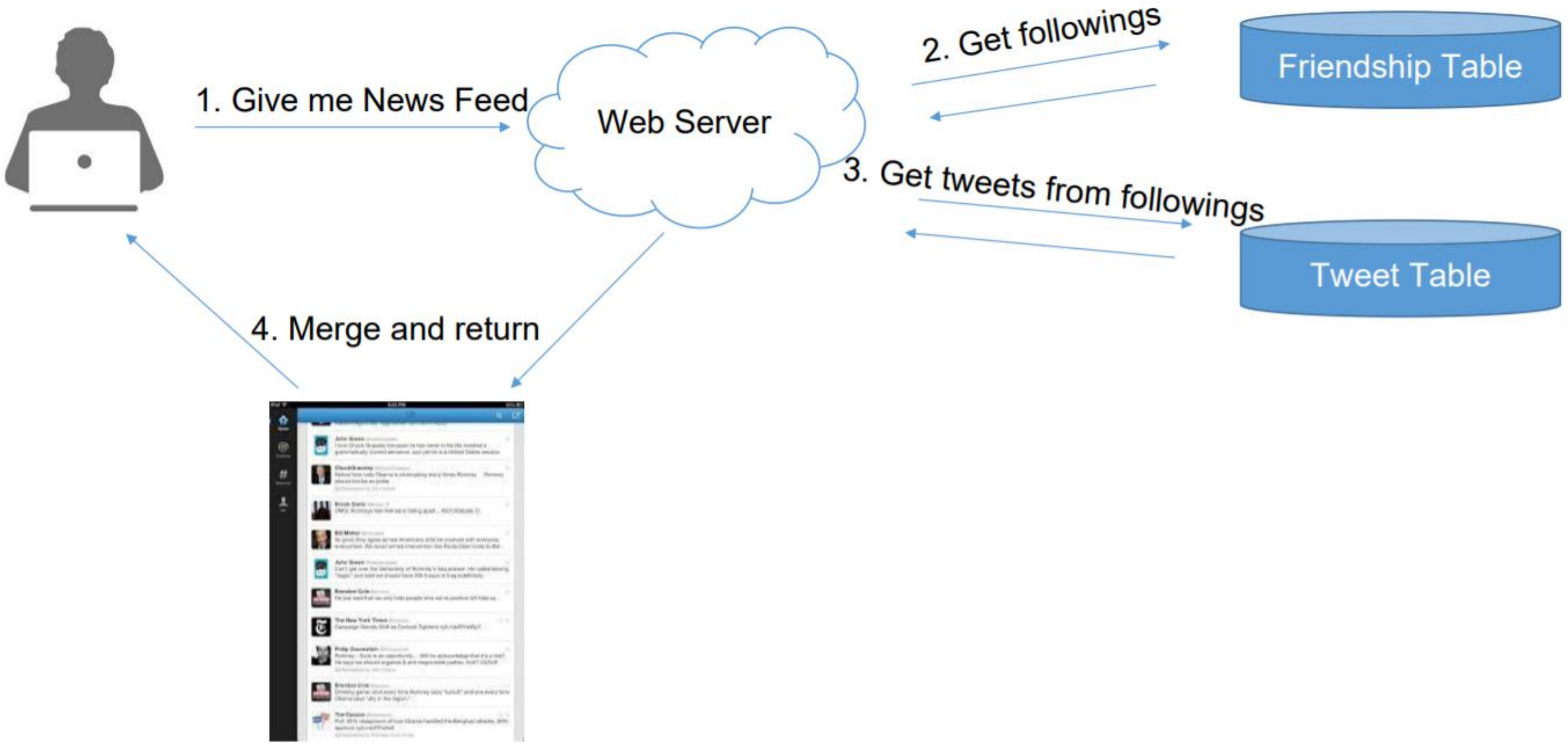
Pull & Push

Pull = 主动撩 Ta

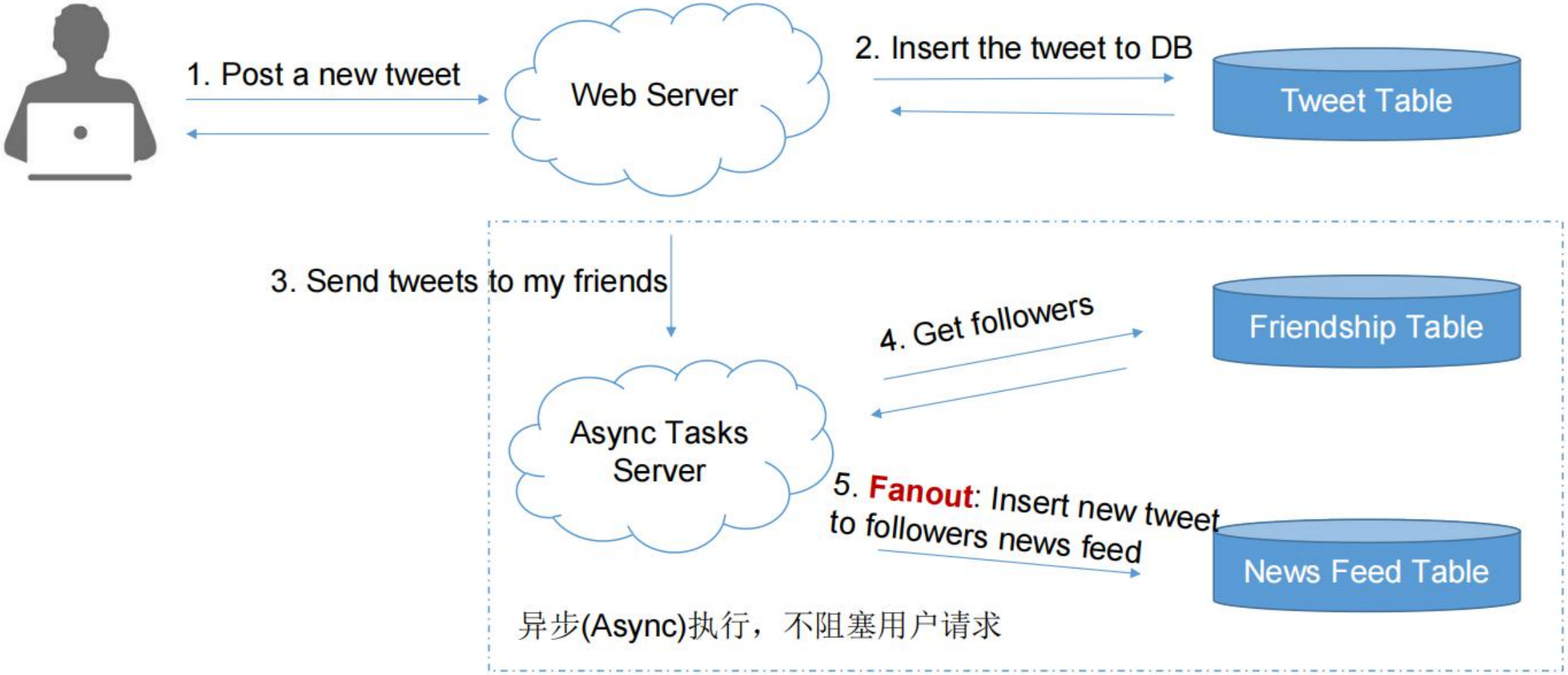


Push = 坐等被撩

Pull

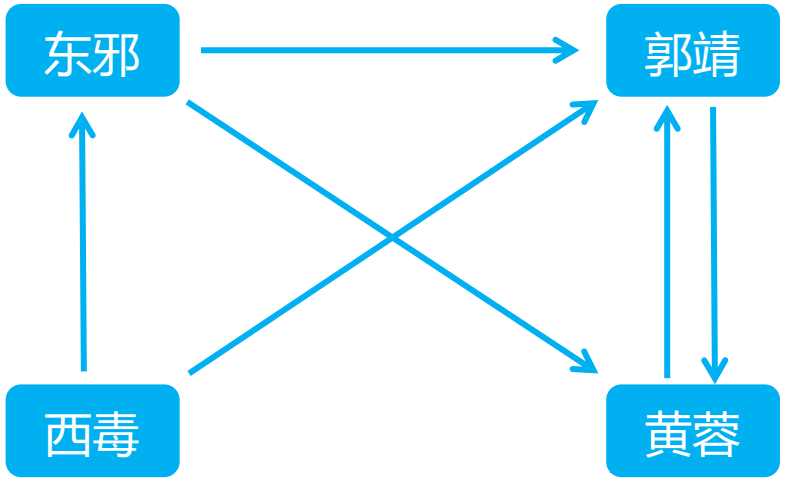


Push



什么是 News Feed Table

东邪西毒，郭靖黄蓉的好友关系如下：



东邪发了一条帖子：“好想念超风”
西毒发了一条帖子：“好想念嫂子”
郭靖发了一条帖子：“不知道华筝怎么样了”
黄蓉发了一条帖子：“男人都不是好东西”

News Feed Table			
id	owner_id	tweet_id	created_at
1	东邪	东邪：好想念超风	2021/03/29 16:30:00
2	西毒	东邪：好想念超风	2021/03/29 16:30:00
3	西毒	西毒：好想念嫂子	2021/03/29 16:35:00
4	郭靖	郭靖：不知道华筝怎么样了	2021/03/29 17:00:00
5	东邪	郭靖：不知道华筝怎么样了	2021/03/29 17:00:00
6	西毒	郭靖：不知道华筝怎么样了	2021/03/29 17:00:00
7	黄蓉	郭靖：不知道华筝怎么样了	2021/03/29 17:00:00
8	黄蓉	黄蓉：男人都不是好东西	2021/03/29 18:00:00
9	郭靖	黄蓉：男人都不是好东西	2021/03/29 18:00:00
10	东邪	黄蓉：男人都不是好东西	2021/03/29 18:00:00

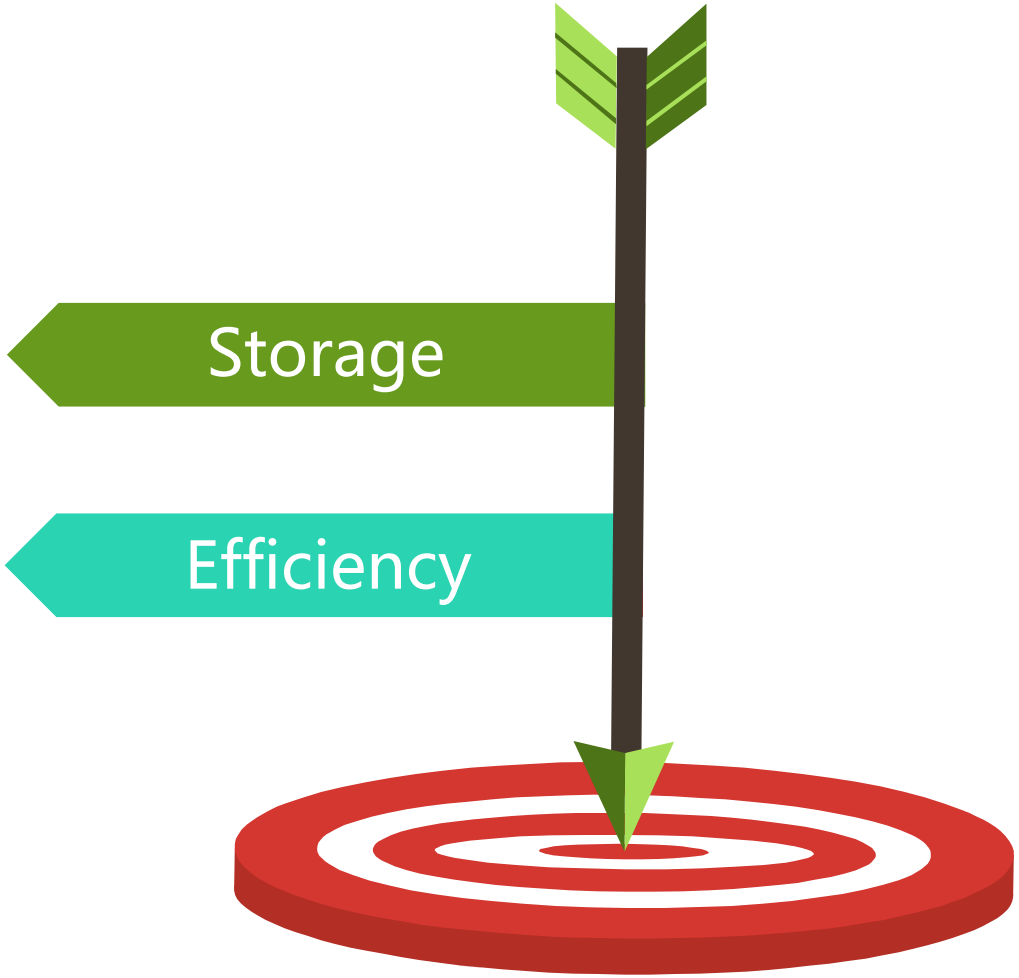
黄蓉登陆“射雕APP”之后可以看到的所有帖子通过一句 SQL 查询可以拿到：
select * from news_feed_table where owner_id=黄蓉 order_by created_at desc limit 20;

存储

- Pull 模式下一个人发帖只在 Tweet 表单中产生一条新记录
- Push 模式下一个人发帖除了在 Tweet 表单中产生一条新纪录，还会产生 N 条 NewsFeed 记录，N 是粉丝的个数

效率

- Pull is heavy on read, light on write
- Push is heavy on write, light on read



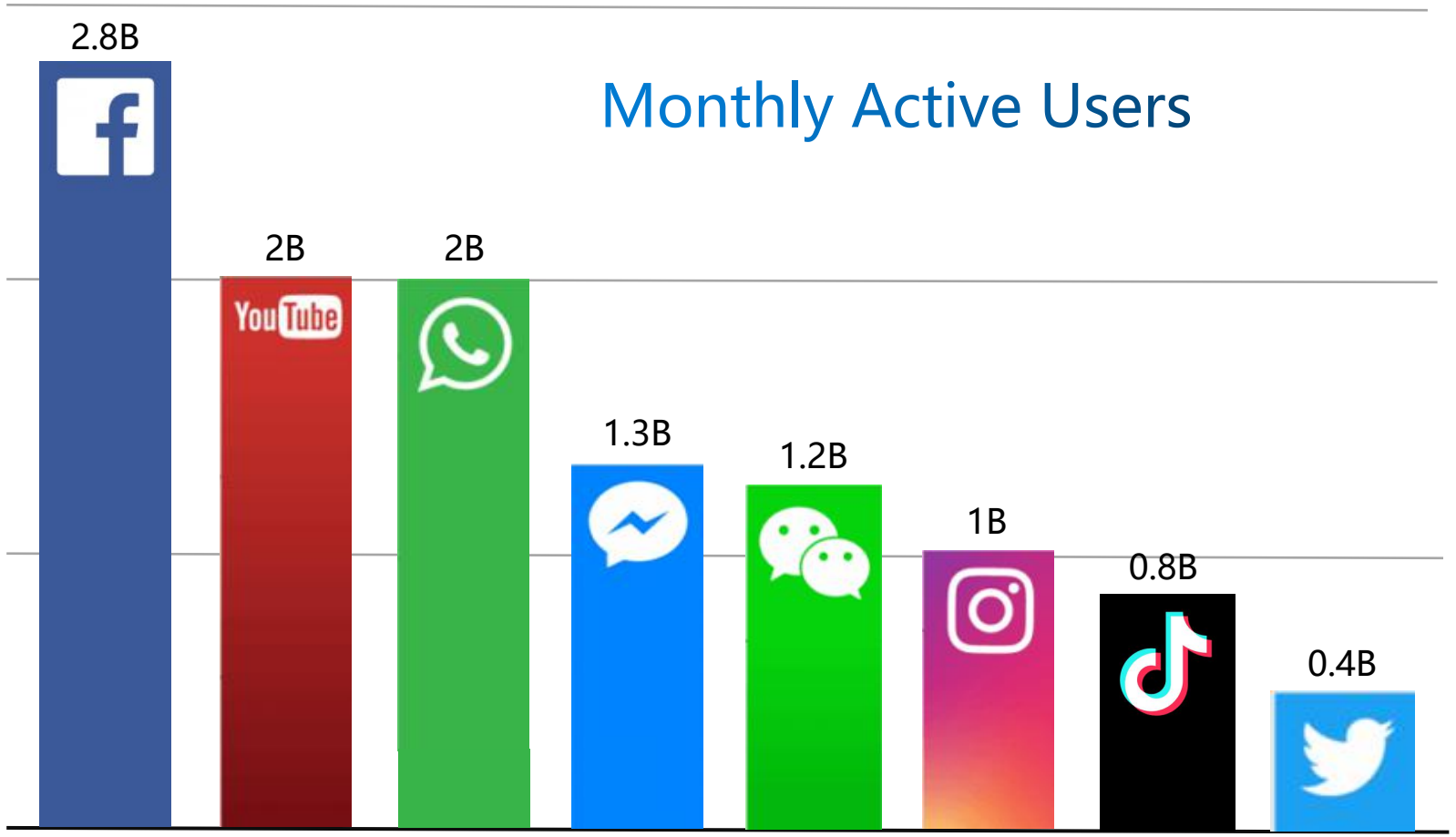
热门 Social App 的模型

- Facebook - Pull
- Instagram - Push + Pull
- Twitter - Pull
- 朋友圈 - ?

误区

- 不坚定想法，摇摆不定
- 不能表现出 Tradeoff 的能力
- 无法解决特定的问题

猜猜微信朋友圈用的是什么模型?



Push or Pull ?

- 大厂的选择：几乎所有的大厂都用了 Pull 的模式来实现信息流系统。但也都经历过先实现 Push 的方式再慢慢迭代优化为 Pull 的过程。
- 本课程将带着大家实现 Push 模式下的信息流系统。

Push 的代码实现

Project Code

```python
class NewsFeedViewSet(viewsets.GenericViewSet):
    permission_classes = [IsAuthenticated]
    pagination_class = EndlessPagination

    def get_queryset(self):...

    @method_decorator(ratelimit(key='user', rate='5/s', method='GET', block=True))
    def list(self, request):
        newsfeeds, queryset = NewsFeedService.get_cached_newsfeeds_with_queryset(request.user.id)
        paginated_newsfeeds = self.paginator.paginate_cached_list_or_queryset(
            newsfeeds,
            queryset,
            request,
        )
        serializer = NewsFeedSerializer(paginated_newsfeeds, many=True)
        return self.paginator.get_paginated_response(serializer.data)
```

Push 的代码实现

Project Code

```python
class NewsFeedService(object):

    @classmethod
    def fanout_to_followers(cls, tweet):
        ...
        fanout_newsfeeds_main_task.delay(tweet.id, tweet.user_id)

    @classmethod
    def get_cached_newsfeeds_with_queryset(cls, user_id):
        if isinstance(user_id, User):
            # 避免不小心传入了 user object 作为参数导致 key 的构造出错
            user_id = user_id.id
        queryset = NewsFeed.objects.filter(user_id=user_id).order_by('-created_at')
        key = USER_NEWSFEEDS_PATTERN.format(user_id=user_id)
        return RedisHelper.load_objects(key, queryset), queryset

    @classmethod
    def push_newsfeed_to_cache(cls, newsfeed):
        key = USER_NEWSFEEDS_PATTERN.format(user_id=newsfeed.user_id)
        RedisHelper.push_object(key, newsfeed)
```

直播间特惠

活动团购基础上在减$300

仅限今日直播间下单

送 LintCode VIP 1年 热

+送 价值$199【递归九讲】 热

再送 FB, Google, Amazon, Microsoft等一线公司
1次**内推机会**
【仅限今日直播间下单前10位！！！】

扫码立即购课
优惠码
【FCDBD5】

早鸟优惠码【FCDBD5】

立减$1300

早鸟优惠

北京时间2021年8月1日失效!

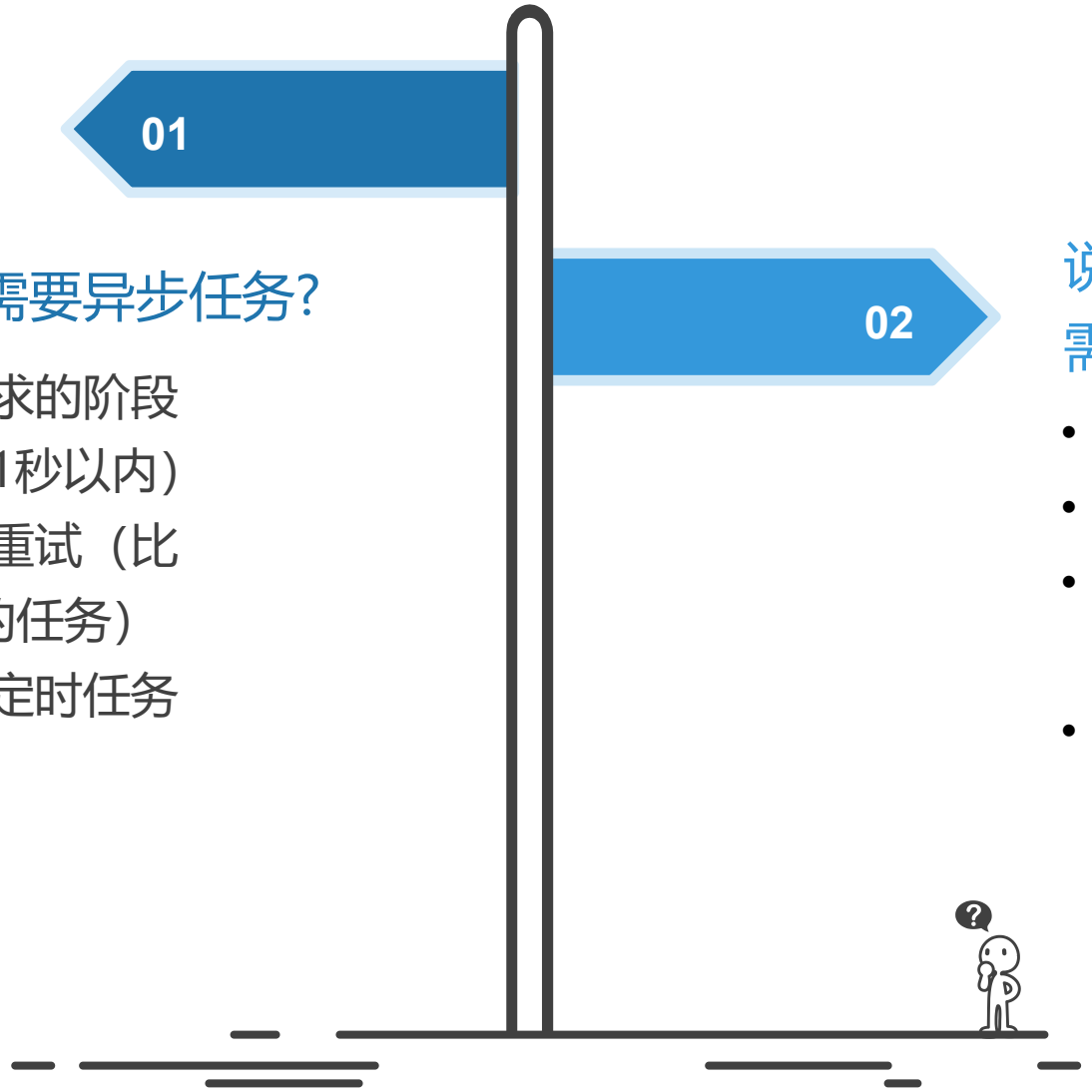
扫码立即使用：↓

扫码立即使用：

加豆豆（JZbanbanya）咨询更多福利

扫码进群，领课件、回放等更多福利

什么是异步任务



一个 Web 系统中为什么需要异步任务?

- 有一些任务无法在用户请求的阶段完成（一般用户的耐心是1秒以内）
- 有一些任务需要进行失败重试（比如和第三方API进行沟通的任务）
- 有一些需要周期性执行的定时任务

说说看我们平时遇到的哪些需求是需要通过异步实现的?

- 发邮件
- 注册账号
- 在 LintCode 上提交代码评测发送提醒邮件/短信
- 红包超时未领取之后自动撤回

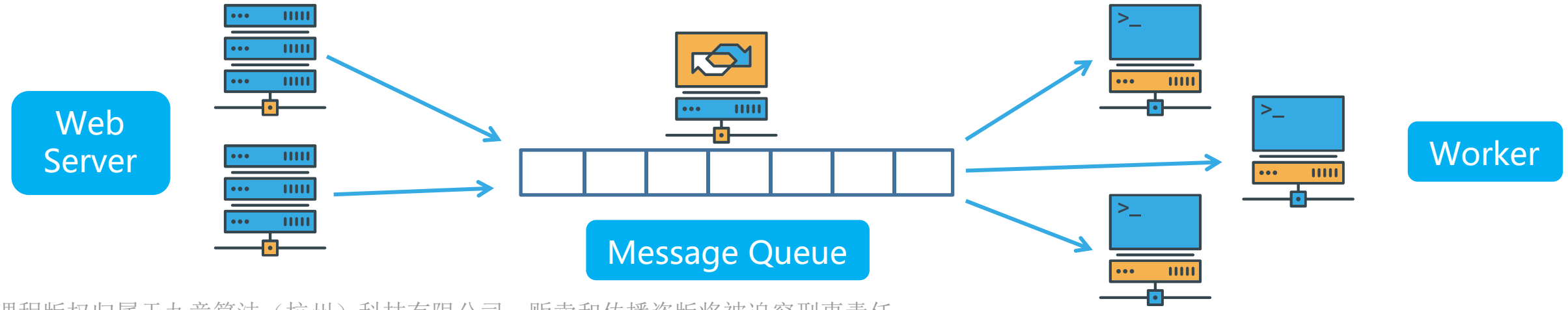
什么是消息队列

一个异步任务的执行离不开消息队列，因为异步任务的信息会被作为数据放在消息队列中，等着执行者（worker）来取走进行执行。

01 消息队列是一种中间件（Middleware）啥是中间件？大概可以理解为系统中汇总传递信息的组件。

02 消息队列可以控制任务的优先级，可以分配不同的队列让不同的执行者机器执行不同的任务。

03 消息队列可以确保一个任务只被一个执行者拿到避免重复执行。



什么是消息队列

常见的消息队列有:

- RabbitMQ
- Redis
- AntMQ
- ZeroMQ

...

在本项目课中，我们将使用 **Redis** 作为我们的消息队列中间件，使用 **Celery** 作为异步任务执行框架的第三方库。

Views & Likes 有什么区别?

- views = 被阅读过多少次
- likes = 被多少人点赞过

Views & Likes 在存储上有什么区别?

- Views 只是单纯有一个计数器+1
- Likes 会记录下到底哪些人点赞了

Normalization

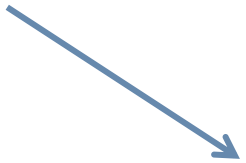
Tweet Table	
id	integer
user_id	Foreign Key
content	text
created_at	timetamp

Like Table *	
id	integer
user_id	Foreign Key
tweet_id	Foreign Key
created_at	timetamp

请写出 SQL 语句，查询 tweet_id = 1 的点赞人数有多少?

Normalization - 查询点赞人数

Tweet Table	
id	integer
user_id	Foreign Key
content	text
created_at	timetamp



Like Table *	
id	integer
user_id	Foreign Key
tweet_id	Foreign Key
created_at	timetamp

```
SELECT COUNT(*) FROM like_table WHERE tweet_id=1;
```

优点： 标准化，最准确。

缺点： 炒鸡慢，会增加 O(N) 个 SQL Queries（对于某一页的 Tweets，每个都得来这么一句查询）

什么是 Denormalization ?

Tweet Table	
id	integer
user_id	Foreign Key
content	text
created_at	timetamp
likes_count *	integer
comments_count *	integer
retweets_count *	integer

Like Table *	
id	integer
user_id	Foreign Key
tweet_id	Foreign Key
created_at	timetamp

冗余存储

Denormalization - 查询和更新点赞人数

Tweet Table	
id	integer
user_id	Foreign Key
content	text
created_at	timetamp
likes_count *	integer
comments_count *	integer
retweets_count *	integer

Like Table *	
id	integer
user_id	Foreign Key
tweet_id	Foreign Key
created_at	timetamp

冗余存储

查询点赞人数：

```
SELECT likes_count FROM tweet_table WHERE id=1
```

当有人点赞时：

```
UPDATE tweet_table SET likes_count = likes_count + 1 WHERE id = 1;
```

点赞功能代码展示

Project Code

```python
class Tweet(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    photos = models.ManyToManyField(Photo, blank=True)
    ...
    likes_count = models.BigIntegerField(default=0, null=True)
    comments_count = models.BigIntegerField(default=0, null=True)
```

Project Code

```python
class Like(models.Model):
    # https://docs.djangoproject.com/en/3.1/ref/contrib/contenttypes/#generic-relations
    object_id = models.PositiveIntegerField()
    content_type = models.ForeignKey(
        ContentType,
        on_delete=models.SET_NULL,
        null=True,
    )
    # user liked content_object at created_at
    content_object = GenericForeignKey('content_type', 'object_id')
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    created_at = models.DateTimeField(auto_now_add=True)
```

模型

点赞功能代码展示

Project Code

```python
class LikeViewSet(viewsets.GenericViewSet):
    queryset = Like.objects.all()
    permission_classes = [IsAuthenticated]
    serializer_class = LikeSerializer

    @required_params(method='POST', params=['content_type', 'object_id'])
    @method_decorator(ratelimit(key='user', rate='10/s', method='POST', block=True))
    def create(self, request, *args, **kwargs):
        serializer = LikeSerializerForCreate(
            data=request.data,
            context={'request': request},
        )
        if not serializer.is_valid():
            return Response({
                'message': 'Please check input',
                'errors': serializer.errors,
            }, status=status.HTTP_400_BAD_REQUEST)
        instance, created = serializer.get_or_create()
        if created:
            NotificationService.send_like_notification(instance)
        return Response(
            LikeSerializer(instance).data,
            status=status.HTTP_201_CREATED,
        )
```

视图

点赞功能代码展示

Project Code

```python
def incr_likes_count(sender, instance, created, **kwargs):
    from tweets.models import Tweet
    from django.db.models import F
    from utils.redis_helper import RedisHelper

    if not created:
        return

    model_class = instance.content_type.model_class()
    if model_class != Tweet:
        # TODO 给 Comment 使用类似的 Dernomalization 的方法进行 likes_count 的统计
        return

    # handle tweet likes
    tweet = instance.content_object
    Tweet.objects.update(likes_count=F('likes_count') + 1)
    RedisHelper.incr_count(tweet, 'likes_count')
```

方法

Project Code

```python
def decr_likes_count(sender, instance, **kwargs):
    from tweets.models import Tweet
    from django.db.models import F
    from utils.redis_helper import RedisHelper

    model_class = instance.content_type.model_class()
    if model_class != Tweet:
        # TODO 给 Comment 使用类似的 Dernomalization 的方法进行 likes_count 的统计
        return

    # handle tweet likes cancel
    tweet = instance.content_object
    Tweet.objects.update(likes_count=F('likes_count') - 1)
    RedisHelper.decr_count(tweet, 'likes_count')
```

第三部分 后续流程 & QA

有哪些先修知识需要学？
每周建议投入多少时间

课程先修知识

Python 基础（3课时）、SQL 基础（1课时）

Web 基础（1课时）、Linux & Git 基础（1课时）

全都已经包含在了第一周的课程中（其中4个课时免费）

相关知识涉猎的编程练习题在阶梯训练前两个阶梯：
https://www.lintcode.com/collection/194/

务必在第二周课程前**听完、刷完**

每周建议投入时间

直播课 2 小时（回放有效期一周，一定要来，要不然很容易懈怠）

录播互动课 3 - 6 小时（回放有效期一个月，可以倍速播放，可以回看暂停）

代码实现 6 - 10 小时（600 - 1000行代码）

平均每天 2-3 小时

加班班微信进班级群