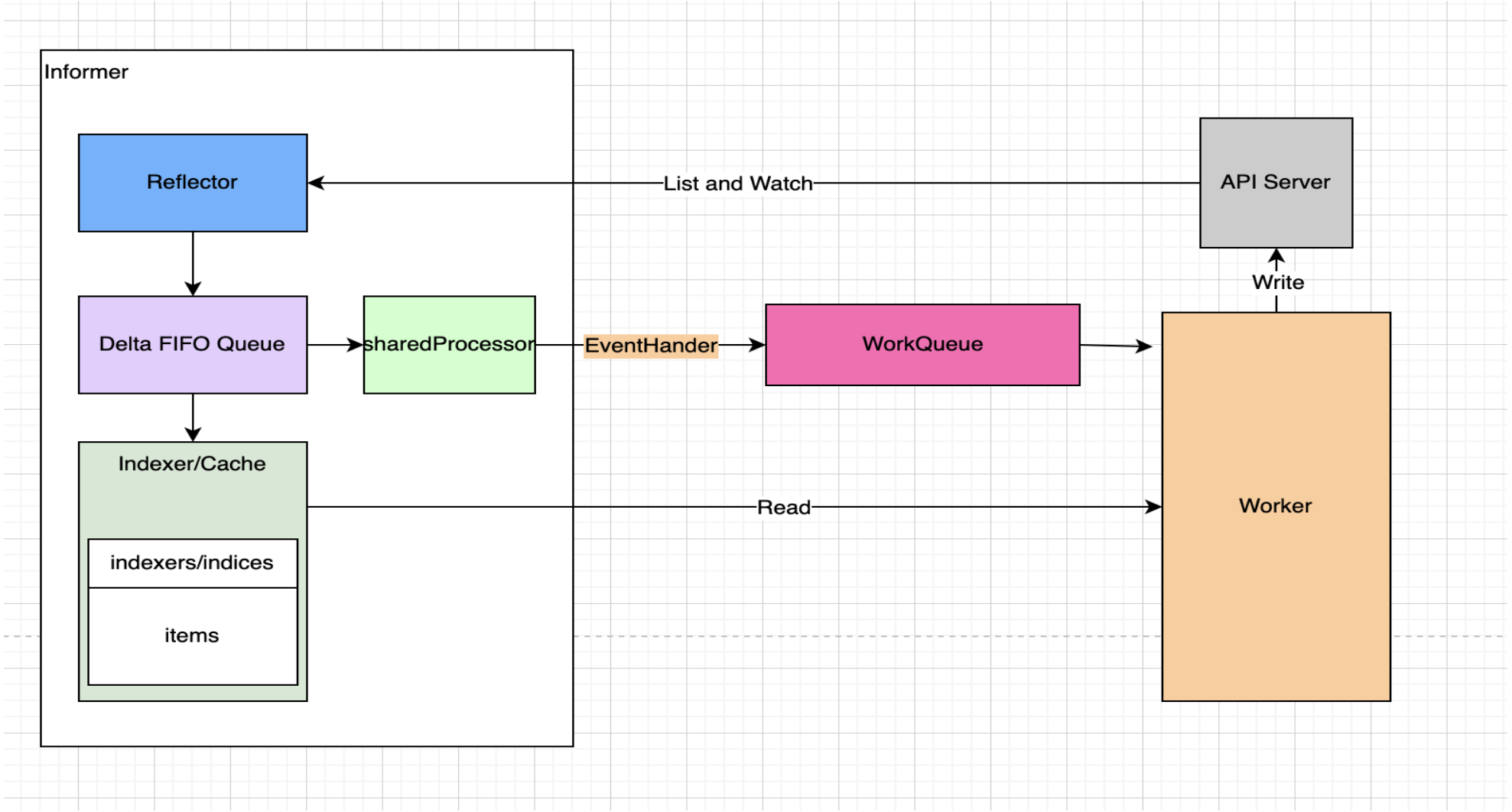
DeltaFIFO原理


Informer
Reflector
List and Watch
API Server
Write
Delta FIFO Queue
sharedProcessor
EventHander
WorkQueue
Indexer/Cache
Read
Worker
indexers/indices
items

Store的类型

- **cache**:实现Store，利用threadSafeMap存放数据(下节课介绍)
- **UndeltaStore**:实现Store，利用cache存放数据，数据变更时通过PushFunc发送当前完整状态
- **FIFO**:实现Queue（包含Store），利用自己内部的items数据结构存放数据
- **DeltaFIFO**:本节课内容，稍后讲解
- **Heap**:实现Store，利用data数据结构存放数据，实现堆数据结构，用于优先级队列
- **ExpirationCache**:实现Store，利用threadSafeMap存放数据

DeltaFIFO的应用场景

DeltaFIFO主要用在以下场景中：

- 你希望处理每个对象的变化最多一次
- 当你处理一个对象时，希望知道这个对象与你上次处理时，发生了哪些变化
- 你希望一个对象删除时，你仍然能够处理它
- 能够周期性的重新处理所有的对象

DeltaFIFO定义

```
type DeltaFIFO struct {
  //...
  //存放Delta
  //与queue中存放的key是同样的key
        items map[string]Deltas
  //可以确保顺序性
        queue []string
  //...
  //默认使用MetaNamespaceKeyFunc，默认使用<namespace>/<name>的格式，不指定namespace时用<name>
  //那么我们从队列的key里面也可以获取到重要的信息了
        keyFunc KeyFunc
  //其实就是Indexer
        knownObjects KeyListerGetter
  //...
}
```

事件的生产和消费

生产

- Reflector的List
- Reflector的Watch
- Reflector的Resync

消费

- 事件派发到work queue（第九节WorkQueue原理时讲解）
- 刷新本地缓存(下节课介绍)

谢谢