Question 1: What's relationship between Apache Zookeeper and paxos protocol?

Zookeeper用的是类似paxos的ZAB算法，Paxos算法用于构建一个分布式一致性状态机系统，ZAB算法用于构建一个高可用的分布式数据主备系统。Paxos算法分为读阶段（收集上一个主进程提出的提案，并提交）和写阶段（当前主进程Leader开始提出自己的提案），而ZAB分为发现（等同于读阶段）、同步阶段、广播阶段（等同于写阶段），新增的同步阶段中，新Leader会确保存在过半的Follower已经提交之前Leader周期中的所有事务Proposal。

Question 2: Many databases or file systems also provide distributed replication support. However, many real-world applications choose Zookeeper for configurations management. Why? What feature of Zookeeper makes it an excellent choice for configuration management?

首先configuration management需要什么?

- 同步性，即配置文件更新后，保证能被同步到微服务中
- 可靠性，配置文件更新后，就一直被保留下来，直到被再次更改
- 一致性，必须保证同步到每个为服务中，不能只同步一部分
- 方便快捷，管理配置肯定越快越好

而基于ZAB算法的ZooKeeper就能很好的保证可靠性、一致性与同步性（注意ZooKeeper仅仅保证在一定的时间段内，客户端最终一定能够从服务端上读取到最新的数据状态）。zookeeper可以看作一个精简的文件系统，用于管理小文件，相对数据库和文件系统来说很轻量级，很有优势。而且zookeeper的设计采用的是观察者的设计模式，zookeeper主要是负责存储和管理大家关心的数据，然后接受观察者的注册，一旦这些数据的状态发生变化，Zookeeper 就将负责通知已经在 Zookeeper 上注册的那些观察者做出相应的反应，从而实现集群中类似 Master/Slave 管理模式，因此十分适合做configuration management。

Question 3: Please implement the naming service by yourself. You need to submit your source codes and all extended docker files in the final answer package and briefly describe your implementation in the document.

先起一个zookeeper集群，然后用java写一个watcher程序（输入是zookeeper的ip：port、监听的节点、自己的名字以及要写入的文件，他会一直监听指定节点的改变，然后把最新改变写入到指定的要写入文件，本作业中我给的参数是172.20.0.3,172.20.0.4,172.20.0.5(zookeeper集群的ip), /jy, $NAME（每个app的名字）, /etc/hosts ），然后就是原来的每个app都后台跑一个watcher，在刚起来的时候读取自己的/etc/hosts找到自己的ip，和自己的名字更新到zookeeper中，zookeeper的更新又触发每一个已注册的watcher的回调函数：将最新的数据写入到/etc/hosts（即参数要写入的文件），于是这个naming service就工作起来了。

Question 4: Please partition the database according to the previous description and briefly introduct your solution.

还是像之前那样，制作mongo镜像的时候加一个后台运行的watcher，然后docker run的时候用这个镜像docker run三次，传入的环境变量分别为：NAME=carts-db、NAME=orders-db、NAME=user-db就可以了。