1. 以Docker Compose安装kafka:
2. docker compose简介：

Compose是一个用于定义和运行多容器Docker应用程序的工具。使用Compose，可以使用YAML文件来配置应用程序的服务。然后，使用单个命令，您可以从配置中创建并启动所有服务。使用docker compose基本为以下三个步骤：

1. 使用Dockerfile定义应用程序的环境，以便可以在任何地方进行复制。
2. 在docker-compose.yml中定义构成应用程序的服务，以便它们可以在隔离环境中一起运行
3. 运行docker-compose up和Compose启动并运行整个应用程序。
4. Docker compose安装（linux）：
5. 下载docker compose:

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

(要安装不同版本的Compose，只需将1.23.2替换为要使用的Compose版本。)

1. 对二进制文件应用可执行权限：

sudo chmod +x /usr/local/bin/docker-compose

1. 查看当前安装docker compose版本：

$ docker-compose --version

1. Zookeeper:
2. 简介：

ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，它是集群的管理者，监视着集群中各个节点的状态根据节点提交的反馈进行下一步合理操作。最终，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

1. Zookeeper通知机制：

客户端注册监听它关心的目录节点，当目录节点发生变化（数据改变、被删除、子目录节点增加删除）时，zookeeper会通知客户端。

1. Zookeeper的配置管理：

程序总是需要配置的，如果程序分散部署在多台机器上，要逐个改变配置就变得困难。现在把这些配置全部放到zookeeper上去，保存在 Zookeeper 的某个目录节点中，然后所有相关应用程序对这个目录节点进行监听，一旦配置信息发生变化，每个应用程序就会收到 Zookeeper 的通知，然后从 Zookeeper 获取新的配置信息应用到系统中就好。

1. Zookeeper集群管理：

所谓集群管理无在乎两点：是否有机器退出和加入、选举master。 对于第一点，所有机器约定在父目录GroupMembers下创建临时目录节点，然后监听父目录节点的子节点变化消息。一旦有机器挂掉，该机器与 zookeeper的连接断开，其所创建的临时目录节点被删除，所有其他机器都收到通知：某个兄弟目录被删除。新机器加入也是类似，所有机器收到通知：新兄弟目录加入。

1. Zookeeper分布式锁：

有了zookeeper的一致性文件系统，锁的问题变得容易。锁服务可以分为两类，一个是保持独占，另一个是控制时序。对于第一类，我们将zookeeper上的一个znode看作是一把锁，通过createznode的方式来实现。所有客户端都去创建 /distribute\_lock 节点，最终成功创建的那个客户端也即拥有了这把锁。用完删除掉自己创建的distribute\_lock 节点就释放出锁。 对于第二类， /distribute\_lock 已经预先存在，所有客户端在它下面创建临时顺序编号目录节点，和选master一样，编号最小的获得锁，用完删除，依次方便。

1. Zookeeper队列管理：
   1. 同步队列，当一个队列的成员都聚齐时，这个队列才可用，否则一直等待所有成员到达。
   2. 队列按照 FIFO 方式进行入队和出队操作。

第一类，在约定目录下创建临时目录节点，监听节点数目是否是我们要求的数目。

第二类，和分布式锁服务中的控制时序场景基本原理一致，入列有编号，出列按编号。

1. zookeeper在kafka中的作用:

Kafka将元数据信息保存在Zookeeper中，但是发送给Topic本身的数据是不会发到Zk上的，否则Zk就疯了。kafka使用zookeeper来实现动态的集群扩展，不需要更改客户端（producer和consumer）的配置。broker会在zookeeper注册并保持相关的元数据（topic，partition信息等）更新。而客户端会在zookeeper上注册相关的watcher。一旦zookeeper发生变化，客户端能及时感知并作出相应调整。这样就保证了添加或去除broker时，各broker间仍能自动实现负载均衡。这里的客户端指的是Kafka的消息生产端(Producer)和消息消费端(Consumer)Producer端使用zookeeper用来"发现"broker列表,以及和Topic下每个partition的leader建立socket连接并发送消息。也就是说每个Topic的partition是由Lead角色的Broker端使用zookeeper来注册broker信息,以及监测partition leader存活性.Consumer端使用zookeeper用来注册consumer信息,其中包括consumer消费的partition列表等,同时也用来发现broker列表,并和partition leader建立socket连接,并获取消息.

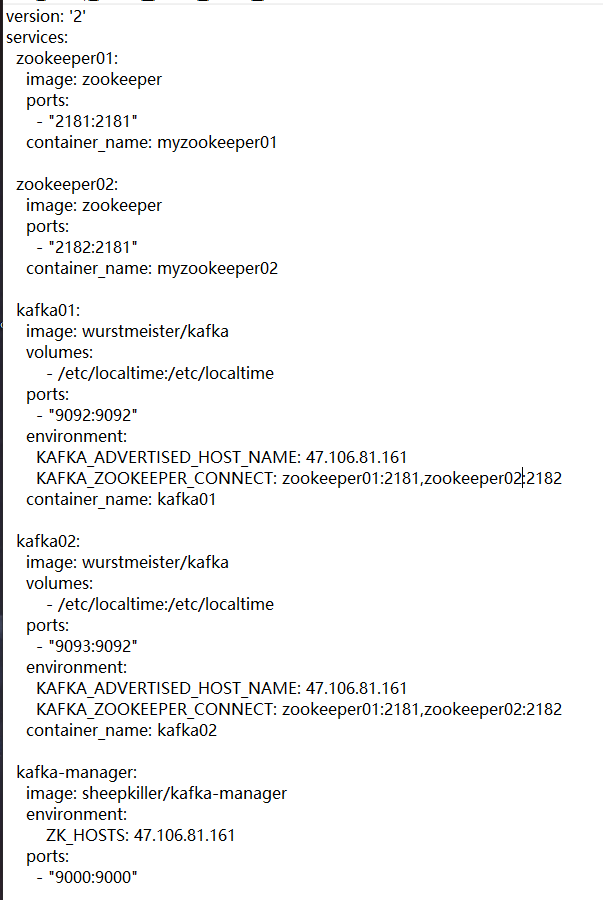
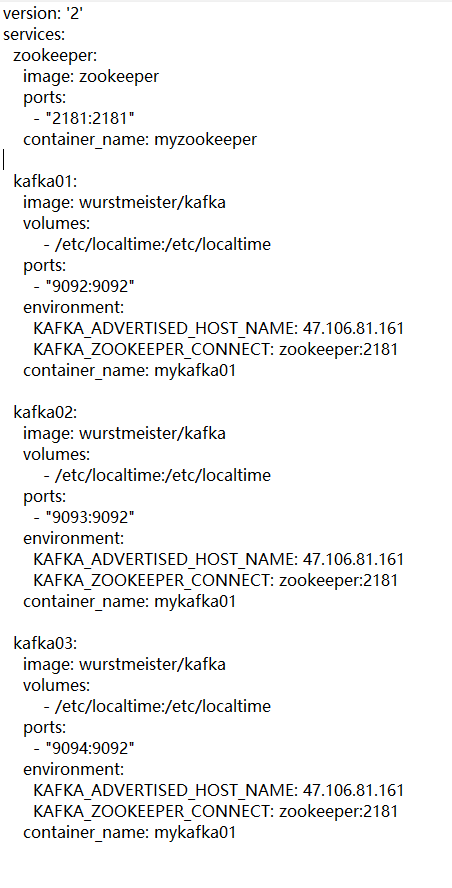
1. Kafka、Zookeeper、Kafka-manager安装：
2. 创建并编辑docker-compose.yml文件：



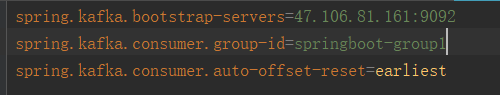
1. 运行
2. Kafka与zookeeper集群部署（docker-compose.yml）：

可能遇到的坑：使用docker 启动kafka时出现秒退

原因：kafka需要大量内存，磁盘内存不足，kafka无法启动，默认启动为1G

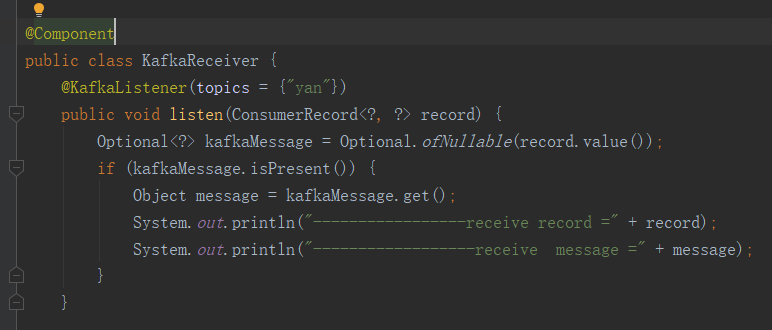
1. Springboot整合kafka示例：
2. 配置：



1. 发送：



1. 接收：



参考：docker compose:<https://docs.docker.com/compose>

zookeeper: <http://www.cnblogs.com/felixzh/p/5869212.html>