1. 准备资源

- 3台Linux裸机服务器，配置4G，2核
- JDK 8运行环境
- 下载Pulsar最新版本安装包

2. 集群组成说明

1. 搭建Pulsar集群至少需要3个组件：ZooKeeper 集群、BookKeeper集群和broker集群（Broker 是 Pulsar 的自身实例)。这三个集群组件如下：

- ZooKeeper 集群（3 个 ZooKeeper 节点组成）
- bookie 集群（也称为 BookKeeper 集群，3 个 BookKeeper 节点组成）
- broker 集群（3 个 Pulsar 节点组成）

2. Pulsar 的安装包已包含了搭建集群所需的各个组件库。无需单独下载 ZooKeeper 安装包和 BookKeeper 安装包。
3. 3 台 Linux 服务器 IP 分别为 10.0.100.60，10.0.100.70 和 10.0.100.80。 如果是在内网测试环境搭建集群，为了避免防火墙造成端口开启繁琐，可以关闭服务器防火墙。

3. 安装JDK

在 3 台 Linux 裸机服务器上安装 JDK（要求版本不低于 JDK 8）。

1. 下载 JDK.
2. 把安装包上传至一台 Linux 服务器中。例如：/home/admin/jdk-8u201-linux-x64.tar.gz.
3. 用如下命令新建 java 文件夹，把 JDK 安装包（jdk-8u201-linux-x64.tar.gz）解压到该文件夹中.

```
cd /usr
sudo -u root mkdir java
cp /home/admin/jdk-8u201-linux-x64.tar.gz  /usr/java/
cd java/
tar -zxvf jdk-8u201-linux-x64.tar.gz
```

4. 编辑配置文件，配置环境变量。

```
# 编辑配置文件
vim /etc/profile

# 配置文件中添加以下配置信息JAVA_HOME=/usr/java/jdk1.8.0_201
CLASSPATH=$JAVA_HOME/lib/
PATH=$PATH:$JAVA_HOME/binexport PATH JAVA_HOME CLASSPATH
```

5. 让配置文件生效。

```
source /etc/profile
```

6. 检查 JDK 是否安装成功。若显示 JDK 版本，则安装成功。

```
java -version
```

> 按照以上步骤，在另外两台 Linux 服务器上安装 JDK。

4. 创建集群环境

1. 在 Linux 服务器上创建三个文件夹：zookeepers、brokers、bookies。

```
# 在工作目录下创建三个文件夹：zookeepers、brokers、bookies
cd /home/admin
mkdir zookeepers
mkdir brokers
mkdir bookies
```

2. 把下载的 Pulsar 安装包上传到 Linux 服务器，解压安装包。

```
# 解压安装包
tar -zxvf apache-pulsar-2.x.x-bin.tar.gz
```

3. 把解压后的文件分别复制到 3 个文件夹。

```
# 把解压后的文件分别复制到3个文件夹
cp -ir  /home/admin/apache-pulsar-2.x.x/*    /home/admin/zookeepers/
cp -ir  /home/admin/apache-pulsar-2.x.x/*     /home/admin/brokers/
cp -ir  /home/admin/apache-pulsar-2.x.x/*     /home/admin/bookies/
```

> 按照以上步骤，在另外两台 Linux 服务器上创建集群环境。

5. 配置部署 ZooKeeper 集群

1. 新建文件夹，并写入配置内容。

```
# 新建文件目录
mkdir -p data/zookeeper

# 新建文件 myid，写入 1
echo 1 > data/zookeeper/myid
```

> 在/home/admin目录执行上面的名利。

注意
另外两台服务器的 myid 文件内容分别写入 2 和 3。

```
# 服务器 2
mkdir -p data/zookeeperecho 2 > data/zookeeper/myid
# 服务器 3
mkdir -p data/zookeeperecho 3 > data/zookeeper/myid
```

2. 配置 zookeeper.conf 文件。

```
# 指定 dataDir 目录dataDir=/home/admin/data/zookeeper

# zookeeper 节点地址
server.1=10.0.100.60:2888:3888
server.2=10.0.100.70:2888:3888
server.3=10.0.100.80:2888:3888
```

注意
在另外两台服务器上，对 zookeeper.conf 文件进行完全相同的配置。

3. 在 zookeepers 目录中，执行启动命令。

```
# 进入 zookeepers 目录
cd /home/admin/zookeepers
# 执行后台运行命令
bin/pulsar-daemon start zookeeper
```

4. 验证 ZooKeeper 节点是否启动成功。

```
# 进入 zookeepers 目录
cd /home/admin/zookeepers

# 执行 zookeeper 客户端连接命令
bin/pulsar zookeeper-shell
```

注意
Enter 键进入命令行界面后，可完全使用 ZooKeeper 的各种命令，如 ls、get 等命令。使用 quit 命令退出命令行界面。

5. 按照以上步骤，在另外两台服务器上部署 ZooKeeper 节点。
6. 在任一个 ZooKeeper 节点，初始化集群元数据。

```
# 进入 zookeepers 目录
cd /home/admin/zookeepers

# 执行命令初始化集群元数据
bin/pulsar initialize-cluster-metadata \
--cluster pulsar-cluster \
--zookeeper 10.0.100.60:2181 \
--configuration-store 10.0.100.60:2181 \
--web-service-url http://pulsar.cluster.com:8080 \
--web-service-url-tls https://pulsar.cluster.com:8443 \
--broker-service-url pulsar://pulsar.cluster.com:6650 \
--broker-service-url-tls pulsar+ssl://pulsar.cluster.com:6651
```

7. 查看集群元数据是否初始化成功。

```
# 进入 zookeepers 目录
cd /home/admin/zookeepers

# 执行 ZooKeeper 客户端连接命令
bin/pulsar zookeeper-shell

# Enter键，使用 ZooKeeper 命令查看
ls /
```

6. 配置部署 BookKeeper 集群

1. 修改配置文件 bookkeeper.conf。

```
# 进入bookie 配置文件目录
cd /home/admin/bookies/conf

# 编辑 bookkeeper.conf 文件
vim bookkeeper.conf

# advertisedAddress 修改为服务器对应的ip,在另外两台服务器也做对应的修改
advertisedAddress=10.0.100.60

# 修改以下两个文件目录地址
journalDirectories=/home/admin/bookies/tmp/journal
ledgerDirectories=/home/admin/bookies/tmp/ledger

# 修改zk地址和端口信息
zkServers=10.0.100.60:2181,10.0.100.70:2181,10.0.100.80:2181
```

2. 初始化元数据，并启动 bookie 集群。

```
# 先执行初始化元数据命令；再执行启动命令

# 进入 bookies 目录
cd  /home/admin/bookies

# 执行初始化元数据命令；若出现提示，输入 Y，继续（只需在一个bookie节点执行一次）
bin/bookkeeper shell metaformat

# 以后台进程启动bookie
bin/pulsar-daemon start bookie
```

3. 按照以上步骤，启动另外两个 bookie 节点。
4. 验证 bookie 是否启动成功。

```
# 进入 bookies 目录
cd  /home/admin/bookies

# 验证是否启动成功
bin/bookkeeper shell bookiesanity

# 出现如下显示，表示启动成功Bookie
sanity test succeeded.
```

7. 部署配置 Broker 集群

1. 修改配置文件 broker.conf。

```
# 进入配置文件目录
cd /home/admin/brokers/conf

# 编辑 broker.conf 文件

# 修改集群名，和 ZooKeeper 里初始化元数据时指定的集群名(--cluster pulsar-cluster)相同
clusterName=pulsar-cluster

# 修改如下两个配置，指定的都是 ZooKeeper 集群地址和端口号
zookeeperServers=10.0.100.60:2181,10.0.100.70:2181,10.0.100.80:2181
configurationStoreServers=10.0.100.60:2181,10.0.100.70:2181,10.0.100.80:2181

# 修改如下参数为本服务器ip地址，另外两个 broker 节点配置文件也做对应修改
advertisedAddress=10.0.100.60
```

2. 启动 broker 节点。

```
# 进入 brokers 目录
cd /home/admin/brokers

# 以后台进程启动 broker
bin/pulsar-daemon start broker
```

3. 按照以上步骤，对另外两个 broker 节点做对应配置，并启动 broker 节点。
4. 检查 JDK 是否安装成功。若显示 JDK 版本，则安装成功。
5. 查看集群中 brokers 节点信息，验证 broker 是否都启动成功。

```
# 进入任一个 broker 目录
cd /home/admin/brokers

# 查看集群 brokers 节点情况
bin/pulsar-admin brokers list pulsar-cluster
```

> **至此，集群 ZooKeeper，Broker，Bookie 节点启动完毕，集群部署成功！接下来可以进行 HelloWorld 测试！**

8. 测试 Pulsar

1. 依次创建集群、租户、命名空间、分区 topic，并为命名空间指定集群名。

```
# 进入 brokers 目录，选取任一个 broker 节点执行命令即可
cd /home/admin/brokers

# 创建集群(集群名:pulsar-cluster)
bin/pulsar-admin clusters create --url http://pulsar.cluster.com:8080  pulsar-cluster

# 创建租户(租户名:my-tenant)
bin/pulsar-admin tenants create my-tenant

# 创建命名空间(命名空间名：my-tenant/my-namespace，它指定了租户 my-tenant)
bin/pulsar-admin namespaces create my-tenant/my-namespace

# 创建持久性分区topic(topic全名:persistent://my-tenant/my-namespace/my-topic；分区数为 3)
bin/pulsar-admin topics create-partitioned-topic persistent://my-tenant/my-namespace/my-topic -p 3

# 更新命名空间为其指定集群名
bin/pulsar-admin namespaces set-clusters my-tenant/my-namespace --clusters pulsar-cluster
```

2. 设置 maven 依赖。

```
<dependency>
    <groupId>org.apache.pulsar</groupId>
    <artifactId>pulsar-client</artifactId>
    <version>2.3.1</version>
</dependency>
```

3. 创建生产者。

```
public class PulsarProducerDemo {

    private static String localClusterUrl = "pulsar://10.0.100.60:6650";

    public static void main(String[] args) {
        try {
            Producer<byte[]> producer = getProducer();
            String msg = "hello world pulsar";

            Long start = System.currentTimeMillis();
            MessageId msgId = producer.send(msg.getBytes());
            System.out.println("spend=" + (System.currentTimeMillis() - start)
                    + ";send a message msgId = " + msgId.toString());
        } catch (Exception e) {
            System.err.println(e);
        }
    }

    public static Producer<byte[]> getProducer() throws Exception {
        PulsarClient client;
        client = PulsarClient.builder().serviceUrl(localClusterUrl).build();
        Producer<byte[]> producer = client.newProducer()
                .topic("persistent://my-tenant/my-namespace/my-topic")
                .producerName("producerName")
                .create();
        return producer;
    }

}
```

4. 创建消费者。

```
public class PulsarConsumerDemo {

    private static String localClusterUrl = "pulsar://10.0.100.60:6650";

    public static void main(String[] args) {
        try {
            Consumer<byte[]> consumer = getClient().newConsumer()
                    .topic("persistent://my-tenant/my-namespace/my-topic")
                    .subscriptionName("my-subscription")
                    .subscribe();
            while (true) {
                Message msg = consumer.receive();
                System.out.printf("consumer-Message received: %s.\n", new String(msg.getData()));
                consumer.acknowledge(msg);
            }
        } catch (Exception e) {
            System.err.println(e);
        }
    }

    public static PulsarClient getClient() throws Exception {
        PulsarClient client;
        client = PulsarClient.builder().serviceUrl(localClusterUrl).build();
        return client;
    }

}
```