这是一份在三台 Dell R450 服务器（128G 内存，1TB SSD + 4TB SATA，16 核 CPU，双千兆网卡，Ubuntu 22.04）上安装配置 Hadoop、Hive、Spark 和 Kafka 集群的详细指南。

重要提示:

1. **版本兼容性:** Hadoop 生态系统组件之间有严格的版本兼容性要求。在选择具体版本时，请务必查阅各组件官方文档，确保 Hadoop, Hive, Spark, Kafka 以及依赖的 Java, Zookeeper 版本相互兼容。本指南以较新且常用的版本为例，但请根据实际情况调整。
2. **数据盘使用:** 您有 4TB 的 SATA 数据盘。强烈建议将其格式化并挂载（例如挂载到 `/data` 目录），并将 HDFS 数据、Kafka 日志、Zookeeper 数据等存储在该盘上，以获得更好的性能和容量。
3. **网络:** 确保服务器之间网络互通，并且防火墙已关闭或正确配置了所需端口。使用静态 IP 地址是最佳实践。
4. **用户:** 建议创建一个专门的用户（例如 `hadoop`）来运行这些服务，避免使用 root 用户。
5. **一致性:** 所有节点上的配置文件、软件版本、Java 版本、用户和路径应保持一致。

指南结构:

假设:

- 服务器主机名和 IP 地址（请替换为您的实际配置）：
  - `server1`：`192.168.1.101` (作为 Master 节点，运行 NameNode, ResourceManager, Zookeeper, Kafka Broker, Hive Metastore/Server2, Spark Master/History Server)
  - `server2`：`192.168.1.102` (作为 Worker 节点，运行 DataNode, NodeManager, Zookeeper, Kafka Broker, Spark Worker)
  - `server3`：`192.168.1.103` (作为 Worker 节点，运行 DataNode, NodeManager, Zookeeper, Kafka Broker, Spark Worker)
- 安装目录: `/opt`
- 数据目录: `/data` (假设 4TB 硬盘已挂载到此目录)
- 运行用户: `hadoop`

---

1. 准备阶段 (所有节点)

在 `server1`, `server2`, `server3` 上执行以下步骤：

1. **更新系统:**

   ```
   sudo apt update
   sudo apt upgrade -y
   ```

2. **安装必要工具:**

   ```
   sudo apt install -y wget tar openssh-server pdsh # pdsh 用于并行执行命令
   ```

3. **配置静态 IP (如果尚未配置):** 编辑 `/etc/netplan/` 下的 yaml 文件 (例如 `00-installer-config.yaml`) 配置静态 IP、网关、DNS。然后运行 `sudo netplan apply`。

4. **配置 Hostname 和 Hosts 文件:**

   - 设置每个服务器的主机名:

     ```
     # 在 server1 上执行
     sudo hostnamectl set-hostname server1
     # 在 server2 上执行
     sudo hostnamectl set-hostname server2
     # 在 server3 上执行
     sudo hostnamectl set-hostname server3
     # 重新登录或重启使之生效
     ```

   - 编辑 `/etc/hosts` 文件，添加所有节点的 IP 和主机名映射：

     ```
     sudo nano /etc/hosts
     # 添加以下内容 (替换为你的实际 IP)
     127.0.0.1       localhost
     192.168.1.101  server1
     192.168.1.102  server2
     192.168.1.103  server3
     ```

   - 验证所有节点之间可以通过主机名 `ping` 通。

5. **创建运行用户:**

5. **创建运行用户**:

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hadoop
# 设置 sudo 权限 (可选，但方便管理)
sudo usermod -aG sudo hadoop
# 为 hadoop 用户设置密码
sudo passwd hadoop
# 切换到 hadoop 用户进行后续操作
su - hadoop
```

注意：后续大部分命令都应以 `hadoop` 用户身份执行。

6. **配置 SSH 免密登录**:
   - 在 `server1` 上，以 `hadoop` 用户身份执行：

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa # 一路回车
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

   - 将公钥复制到 `server2` 和 `server3`：

```
ssh-copy-id hadoop@server2
ssh-copy-id hadoop@server3
```

   - 验证免密登录：

```
ssh server1 date
ssh server2 date
ssh server3 date
# 应该无需密码即可执行命令
```

7. **安装 Java (OpenJDK)**: 选择一个被您要安装的 Hadoop/Spark/Kafka/Hive 版本支持的 JDK 版本（例如 OpenJDK 11 或 8）。这里以 OpenJDK 11 为例：

```
sudo apt update
sudo apt install -y openjdk-11-jdk
# 验证安装
java -version
javac -version
```

   - 配置 `JAVA_HOME` 环境变量。编辑 `~/.bashrc` 文件：

```
nano ~/.bashrc
```

   - 在文件末尾添加（路径可能因系统或安装方式略有不同，请确认）：

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

   - 使配置生效：

```
source ~/.bashrc
# 验证
echo $JAVA_HOME
```

   - **重要**: 在所有节点上重复 Java 安装和环境变量配置步骤。

8. **关闭防火墙 (简化配置，生产环境请配置规则)**:

```
sudo systemctl stop ufw
sudo systemctl disable ufw
sudo systemctl status ufw # 确认 inactive
```

9. **格式化并挂载数据盘 (如果需要)**:
   - 查看磁盘标识符 (假设是 `/dev/sdb`): `lsblk`
   - 格式化 (例如使用 ext4): `sudo mkfs.ext4 /dev/sdb`
   - 创建挂载点: `sudo mkdir /data`
   - 设置开机自动挂载，编辑 `/etc/fstab`: `sudo nano /etc/fstab`
   - 添加类似行 (获取 UUID: `sudo blkid /dev/sdb`):

```
UUID=<your_disk_uuid> /data ext4 defaults 0 0
```

- 挂载: `sudo mount -a`
- 更改 `/data` 目录的所有者为 `hadoop` 用户:

```
sudo chown -R hadoop:hadoop /data
```

- **重要:** 在所有需要存储数据的节点上重复此步骤。

---

2. 安装 Zookeeper (所有节点)

Kafka 依赖 Zookeeper。Hadoop HA 模式也需要 Zookeeper (本指南暂不配置 HA)。

1. **下载 Zookeeper:** (选择一个稳定版本，例如 3.8.x 或 3.7.x) 在 `server1` 上执行：

```
cd /opt
# 访问 Apache Zookeeper 官网查找最新稳定版下载链接
wget https://dlcdn.apache.org/zookeeper/zookeeper-3.8.4/apache-zookeeper-3.8.4-bin.tar.gz # 替换为最新版链接
tar -zxvf apache-zookeeper-3.8.4-bin.tar.gz
ln -s apache-zookeeper-3.8.4-bin zookeeper # 创建软链接方便升级
sudo chown -R hadoop:hadoop /opt/apache-zookeeper-3.8.4-bin /opt/zookeeper
```

2. **配置 Zookeeper:**
   - 创建数据和日志目录:

```
mkdir -p /data/zookeeper/data
mkdir -p /data/zookeeper/logs
```

   - 创建配置文件 `zoo.cfg`:

```
cd /opt/zookeeper/conf
cp zoo_sample.cfg zoo.cfg
nano zoo.cfg
```

   - 修改 `zoo.cfg` 内容如下：

```
tickTime=2000
initLimit=10
syncLimit=5
# 修改数据目录
dataDir=/data/zookeeper/data
# 可选: 修改日志目录，需设置 dataLogDir=/data/zookeeper/logs
clientPort=2181
# 设置 ZK Server 列表 (server.节点ID=主机名:心跳端口:选举端口)
server.1=server1:2888:3888
server.2=server2:2888:3888
server.3=server3:2888:3888
```

   - 创建 `myid` 文件，内容为节点的 ID：

```
# 在 server1 上
echo "1" > /data/zookeeper/data/myid
# 在 server2 上
echo "2" > /data/zookeeper/data/myid
# 在 server3 上
echo "3" > /data/zookeeper/data/myid
```

3. **分发 Zookeeper 到其他节点:** 在 `server1` 上执行：

```
cd /opt
scp -r apache-zookeeper-3.8.4-bin server2:/opt/
scp -r apache-zookeeper-3.8.4-bin server3:/opt/
# 在 server2 和 server3 上创建软链接和修改权限
ssh server2 "sudo ln -s /opt/apache-zookeeper-3.8.4-bin /opt/zookeeper && sudo chown -R hadoop:hadoop /opt/apache-zookeeper-3.
ssh server3 "sudo ln -s /opt/apache-zookeeper-3.8.4-bin /opt/zookeeper && sudo chown -R hadoop:hadoop /opt/apache-zookeeper-3.
# 分发配置文件和 myid (确保 myid 内容正确)
scp /opt/zookeeper/conf/zoo.cfg server2:/opt/zookeeper/conf/
scp /opt/zookeeper/conf/zoo.cfg server3:/opt/zookeeper/conf/
scp /data/zookeeper/data/myid server1:/data/zookeeper/data/ # 确认 server1 的 myid
scp /data/zookeeper/data/myid server2:/data/zookeeper/data/ # 发送给 server2
scp /data/zookeeper/data/myid server3:/data/zookeeper/data/ # 发送给 server3
# **重要**: 登录 server2 和 server3，修改 /data/zookeeper/data/myid 的内容为 2 和 3。
ssh server2 'echo "2" > /data/zookeeper/data/myid'
ssh server3 'echo "3" > /data/zookeeper/data/myid'
```

4. **启动 Zookeeper 集群**: 在 **所有** 节点上执行：

```
cd /opt/zookeeper
bin/zkServer.sh start
```

5. **验证 Zookeeper 状态**: 在 **每个** 节点上执行：

```
bin/zkServer.sh status
# 输出应该显示 Mode: follower 或 Mode: leader
```

或者使用 Zookeeper 客户端连接：

```
bin/zkCli.sh -server server1:2181
# 进入客户端后输入 ls / 查看根节点
```

---

3. 安装 Hadoop (HDFS + YARN)

选择一个稳定版本，例如 Hadoop 3.3.x。

1. **下载 Hadoop**: 在 `server1` 上执行：

```
cd /opt
# 访问 Apache Hadoop 官网查找最新稳定版下载链接
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz # 替换为最新版链接
tar -zxvf hadoop-3.3.6.tar.gz
ln -s hadoop-3.3.6 hadoop # 创建软链接
sudo chown -R hadoop:hadoop /opt/hadoop-3.3.6 /opt/hadoop
```

2. **配置 Hadoop 环境变量**: 在 **所有** 节点上编辑 `~/.bashrc`：

```
nano ~/.bashrc
```

添加以下内容：

```
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
# 如果使用 pdsh
export PDSH_RCMD_TYPE=ssh
```

使配置生效：

```
source ~/.bashrc
# 验证
echo $HADOOP_HOME
hadoop version
```

3. **配置 Hadoop**: 进入 Hadoop 配置目录 `cd /opt/hadoop/etc/hadoop`

- **hadoop-env.sh**：

```
nano hadoop-env.sh
# 设置 JAVA_HOME (如果未自动检测到)
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64 # 确认路径正确
# 设置 HDFS 用户
export HDFS_NAMENODE_USER=hadoop
export HDFS_DATANODE_USER=hadoop
export HDFS_SECONDARYNAMENODE_USER=hadoop
export YARN_RESOURCEMANAGER_USER=hadoop
export YARN_NODEMANAGER_USER=hadoop
```

- **core-site.xml**：

```
nano core-site.xml
```

添加以下配置：

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://server1:9000</value> </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/data/hadoop/tmp</value> </property>
    <property>
         <name>ha.zookeeper.quorum</name>
         <value>server1:2181,server2:2181,server3:2181</value>
    </property>
</configuration>
```

- **hdfs-site.xml**：

```
nano hdfs-site.xml
```

添加以下配置：

```
<configuration>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///data/hadoop/hdfs/name</value> </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///data/hadoop/hdfs/data</value> </property>
    <property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>server2:9868</value> </property>
    <property>
        <name>dfs.namenode.checkpoint.dir</name>
        <value>file:///data/hadoop/hdfs/namesecondary</value> </property>
     <property>
         <name>dfs.namenode.checkpoint.edits.dir</name>
         <value>${dfs.namenode.checkpoint.dir}</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
    <property>
         <name>dfs.permissions.enabled</name>
         <value>false</value>
    </property>
    <property>
        <name>dfs.namenode.http-address</name>
        <value>server1:9870</value>
    </property>
    <property>
       <name>dfs.datanode.http.address</name>
       <value>0.0.0.0:9864</value>
    </property>
</configuration>
```

- **yarn-site.xml**：

```
nano yarn-site.xml
```

添加以下配置：

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>server1</value> </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>102400</value> </property>
    <property>
        <name>yarn.nodemanager.resource.cpu-vcores</name>
        <value>14</value>
    </property>
    <property>
         <name>yarn.log-aggregation-enable</name>
         <value>true</value>
    </property>
    <property>
         <name>yarn.nodemanager.remote-app-log-dir</name>
         <value>/user/logs/apps</value> </property>
    <property>
         <name>yarn.resourcemanager.webapp.address</name>
         <value>server1:8088</value>
    </property>
</configuration>
```

- **`mapred-site.xml`**：(可能需要从 `mapred-site.xml.template` 复制)

```
 cp mapred-site.xml.template mapred-site.xml
nano mapred-site.xml
```

添加以下配置：

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
         <name>mapreduce.jobhistory.address</name>
         <value>server1:10020</value>
    </property>
    <property>
         <name>mapreduce.jobhistory.webapp.address</name>
         <value>server1:19888</value>
    </property>
    <property>
      <name>yarn.app.mapreduce.am.env</name>
      <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
    </property>
    <property>
      <name>mapreduce.map.env</name>
      <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
    </property>
    <property>
      <name>mapreduce.reduce.env</name>
      <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
    </property>
</configuration>
```

- **`workers`** **(Hadoop 3.x 之前是 `slaves`)**：

```
nano workers
```

列出所有 DataNode 和 NodeManager 的主机名：

```
server1 # 通常 Master 也作为 Worker
server2
server3
```

4. **创建所需目录 (在对应节点上):**
   - 在 `server1` 上创建 NameNode 目录和临时目录：

     ```
     mkdir -p /data/hadoop/tmp
     mkdir -p /data/hadoop/hdfs/name
     sudo chown -R hadoop:hadoop /data/hadoop
     ```

   - 在 `server2` 上创建 SecondaryNameNode 目录和临时目录：

     ```
     ssh server2 "mkdir -p /data/hadoop/tmp /data/hadoop/hdfs/namesecondary && sudo chown -R hadoop:hadoop /data/hadoop"
     ```

   - 在 **所有** Worker 节点 (`server1`, `server2`, `server3`) 上创建 DataNode 目录和临时目录：

     ```
     # server1 上已创建 /data/hadoop/tmp
     mkdir -p /data/hadoop/hdfs/data
     sudo chown -R hadoop:hadoop /data/hadoop/hdfs/data # 确保权限

     ssh server2 "mkdir -p /data/hadoop/hdfs/data && sudo chown -R hadoop:hadoop /data/hadoop/hdfs/data"
     ssh server3 "mkdir -p /data/hadoop/tmp /data/hadoop/hdfs/data && sudo chown -R hadoop:hadoop /data/hadoop"
     ```

5. **分发 Hadoop 到其他节点:** 在 `server1` 上执行：

   ```
   cd /opt
   scp -r hadoop-3.3.6 server2:/opt/
   scp -r hadoop-3.3.6 server3:/opt/
   # 在 server2 和 server3 上创建软链接和修改权限
   ssh server2 "sudo ln -s /opt/hadoop-3.3.6 /opt/hadoop && sudo chown -R hadoop:hadoop /opt/hadoop-3.3.6 /opt/hadoop"
   ssh server3 "sudo ln -s /opt/hadoop-3.3.6 /opt/hadoop && sudo chown -R hadoop:hadoop /opt/hadoop-3.3.6 /opt/hadoop"
   # 分发配置文件
   scp /opt/hadoop/etc/hadoop/* server2:/opt/hadoop/etc/hadoop/
   scp /opt/hadoop/etc/hadoop/* server3:/opt/hadoop/etc/hadoop/
   # **重要**: 确保所有节点的 .bashrc 都已配置并 source
   ssh server2 'source ~/.bashrc'
   ssh server3 'source ~/.bashrc'
   ```

6. **格式化 HDFS NameNode: 仅在 `server1` (NameNode 所在节点) 上执行一次！**

   ```
   cd /opt/hadoop
   bin/hdfs namenode -format
   # 看到 "Storage directory ... has been successfully formatted" 表示成功。
   # 如果提示 Re-format? 输入 Y。
   ```

7. **启动 Hadoop 集群:**
   - **启动 HDFS:** 在 `server1` 上执行：

     ```
     sbin/start-dfs.sh
     ```

     使用 `jps` 命令查看进程：
     - `server1` : NameNode, DataNode
     - `server2` : SecondaryNameNode, DataNode
     - `server3` : DataNode
   - **启动 YARN:** 在 `server1` 上执行：

     ```
     sbin/start-yarn.sh
     ```

     使用 `jps` 命令查看进程：
     - `server1` : ResourceManager, NodeManager
     - `server2` : NodeManager
     - `server3` : NodeManager
   - **(可选) 启动 MapReduce Job History Server:** 在 `server1` 上执行：

     ```
     sbin/mr-jobhistory-daemon.sh start historyserver
     # jps 查看 MapReduce Job History Server 进程
     ```

8. **验证 Hadoop 集群:**

- **Web UI**:
  - NameNode: `http://server1:9870`
  - ResourceManager: `http://server1:8088`
  - Job History: `http://server1:19888`
  - 在 Web UI 上检查 DataNode 和 NodeManager 是否都已注册且状态正常。
- **命令行**:
  - 查看 HDFS 报告: `hdfs dfsadmin -report`
  - 创建 HDFS 目录: `hdfs dfs -mkdir /test`
  - 上传文件: `echo "hello hadoop" > test.txt && hdfs dfs -put test.txt /test/`
  - 查看文件: `hdfs dfs -cat /test/test.txt`
  - 运行 MapReduce 示例 (PI): `yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi 10 100` (查看 YARN UI 任务状态)
  - 创建 YARN 日志聚合目录：

```
hdfs dfs -mkdir -p /user/logs/apps
hdfs dfs -chown yarn:hadoop /user/logs/apps
hdfs dfs -chmod 1777 /user/logs/apps
# 可能还需要创建 hadoop 用户目录
hdfs dfs -mkdir -p /user/hadoop
hdfs dfs -chown hadoop:hadoop /user/hadoop
```

---

4. 安装 Kafka

选择一个稳定版本，例如 Kafka 3.x.x。

1. **下载 Kafka**: 在 `server1` 上执行：

```
cd /opt
# 访问 Apache Kafka 官网查找最新二进制下载链接 (Scala 2.13 或 2.12)
wget https://dlcdn.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz # 替换为最新版链接
tar -xzf kafka_2.13-3.7.0.tgz
ln -s kafka_2.13-3.7.0 kafka
sudo chown -R hadoop:hadoop /opt/kafka_2.13-3.7.0 /opt/kafka
```

2. **配置 Kafka**:
   - 创建 Kafka 日志目录 (数据存储)：

```
mkdir -p /data/kafka-logs
sudo chown -R hadoop:hadoop /data/kafka-logs
```

   - 编辑 Kafka 配置文件 `server.properties`：

```
cd /opt/kafka/config
nano server.properties
```

   - **为每个节点修改以下配置**:
     - **`broker.id`**: 每个节点必须唯一。`server1` 设为 1, `server2` 设为 2, `server3` 设为 3。

```
# 在 server1 的配置文件中:
broker.id=1
# 在 server2 的配置文件中:
broker.id=2
# 在 server3 的配置文件中:
broker.id=3
```

     - **`listeners` 和 `advertised.listeners`**: 监听地址和广播给 Zookeeper 的地址。

```
# 如果所有通信都在内网，可以用 PLAINTEXT://主机名:9092
# 如果有跨网段或 Docker 访问，需要配置 advertised.listeners 为外部可访问的 IP/域名
# 示例 (内网):
listeners=PLAINTEXT://:9092 # 监听所有接口
# 在 server1 的配置文件中 (如果 server1 的主机名能被其他节点解析):
advertised.listeners=PLAINTEXT://server1:9092
# 在 server2 的配置文件中:
advertised.listeners=PLAINTEXT://server2:9092
# 在 server3 的配置文件中:
advertised.listeners=PLAINTEXT://server3:9092
```

     - **`log.dirs`**: Kafka 数据存储目录。

```
log.dirs=/data/kafka-logs # 使用数据盘
```

- **`zookeeper.connect`**: Zookeeper 集群地址。

```
zookeeper.connect=server1:2181,server2:2181,server3:2181
```

- **(可选) 其他重要配置**: `num.partitions`, `default.replication.factor`, `offsets.topic.replication.factor`, `transaction.state.log.replication.factor` 等，根据需要调整。对于 3 节点集群，副本因子通常设为 3。

3. **分发 Kafka 到其他节点**: 在 `server1` 上执行：

```
cd /opt
scp -r kafka_2.13-3.7.0 server2:/opt/
scp -r kafka_2.13-3.7.0 server3:/opt/
# 在 server2 和 server3 上创建软链接和修改权限
ssh server2 "sudo ln -s /opt/kafka_2.13-3.7.0 /opt/kafka && sudo chown -R hadoop:hadoop /opt/kafka_2.13-3.7.0 /opt/kafka && mk
ssh server3 "sudo ln -s /opt/kafka_2.13-3.7.0 /opt/kafka && sudo chown -R hadoop:hadoop /opt/kafka_2.13-3.7.0 /opt/kafka && mk
# 分发配置文件 (注意：分发后需要登录 server2 和 server3 修改 broker.id 和 advertised.listeners)
scp /opt/kafka/config/server.properties server2:/opt/kafka/config/
scp /opt/kafka/config/server.properties server3:/opt/kafka/config/
# **重要**: 登录 server2 和 server3，修改 /opt/kafka/config/server.properties 中的 broker.id 和 advertised.listeners
ssh server2 "sed -i 's/broker.id=1/broker.id=2/' /opt/kafka/config/server.properties && sed -i 's/advertised.listeners=PLAINTE
ssh server3 "sed -i 's/broker.id=1/broker.id=3/' /opt/kafka/config/server.properties && sed -i 's/advertised.listeners=PLAINTE
```

4. **启动 Kafka Brokers**: 在 **所有** 节点上以后台模式启动 Kafka Server：

```
cd /opt/kafka
bin/kafka-server-start.sh -daemon config/server.properties
```

使用 `jps` 查看 `Kafka` 进程。

5. **验证 Kafka 集群**:

- **创建 Topic**: (在任意节点执行)

```
cd /opt/kafka
bin/kafka-topics.sh --create --topic my-test-topic --bootstrap-server server1:9092,server2:9092,server3:9092 --replicati
# 查看 Topic 列表
bin/kafka-topics.sh --list --bootstrap-server server1:9092
# 查看 Topic 详情
bin/kafka-topics.sh --describe --topic my-test-topic --bootstrap-server server1:9092
```

- **启动生产者发送消息**: (在一个终端执行)

```
bin/kafka-console-producer.sh --topic my-test-topic --bootstrap-server server1:9092,server2:9092,server3:9092
> Hello Kafka!
> This is a test message.
> (Ctrl+C 退出)
```

- **启动消费者接收消息**: (在另一个终端执行)

```
bin/kafka-console-consumer.sh --topic my-test-topic --from-beginning --bootstrap-server server1:9092,server2:9092,server
# 应该能看到之前发送的消息
# (Ctrl+C 退出)
```

---

5. 安装 Hive

选择与您的 Hadoop 版本兼容的 Hive 版本，例如 Hive 3.1.x。Hive 需要一个 Metastore 来存储元数据，通常使用关系型数据库 (如 MySQL 或 PostgreSQL)。这里以 PostgreSQL 为例。

1. **安装和配置 PostgreSQL (在 server1 上)**:

- 安装 PostgreSQL:

```
sudo apt update
sudo apt install -y postgresql postgresql-contrib
```

- 启动并设置开机自启:

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

- 创建 Hive Metastore 数据库和用户:

```
# 切换到 postgres 用户
sudo -u postgres psql
# 在 psql 命令行中执行:
CREATE DATABASE metastore;
CREATE USER hiveuser WITH PASSWORD 'hivepassword'; -- 设置强密码
GRANT ALL PRIVILEGES ON DATABASE metastore TO hiveuser;
\q
```

- 配置 PostgreSQL 允许远程连接 (如果 HiveServer2 和 Metastore 不在同一台机器，或者需要其他机器访问) 和密码认证。编辑 `pg_hba.conf` 和 `postgresql.conf`。
  - 找到配置文件路径: `sudo -u postgres psql -c 'SHOW config_file;'` 和 `sudo -u postgres psql -c 'SHOW hba_file;'` (通常在 `/etc/postgresql/<version>/main/`)
  - 编辑 pg_hba.conf (例如 `sudo nano /etc/postgresql/14/main/pg_hba.conf`)，添加或修改允许 `hiveuser` 从 HiveServer2 节点连接的规则 (用 md5 密码认证)：

    ```
    # Example: Allow connection from server1 IP for hiveuser to metastore db
    host    metastore   hiveuser    192.168.1.101/32      md5
    # Or allow from any IP on the subnet (adjust as needed)
    # host    metastore   hiveuser    192.168.1.0/24        md5
    ```

  - 编辑 `postgresql.conf` (例如 `sudo nano /etc/postgresql/14/main/postgresql.conf`)，修改监听地址：

    ```
    listen_addresses = '*' # 或者指定具体的 IP 地址，例如 'localhost,192.168.1.101'
    ```

  - 重启 PostgreSQL 服务: `sudo systemctl restart postgresql`

2. **下载并安装 Hive (在 server1 上):**
   - 下载 Hive:

   ```
   cd /opt
   # 访问 Apache Hive 官网查找与 Hadoop 兼容的二进制版本
   wget https://dlcdn.apache.org/hive/hive-3.1.3/apache-hive-3.1.3-bin.tar.gz # 替换为兼容版本链接
   tar -zxvf apache-hive-3.1.3-bin.tar.gz
   ln -s apache-hive-3.1.3-bin hive
   sudo chown -R hadoop:hadoop /opt/apache-hive-3.1.3-bin /opt/hive
   ```

   - 下载 PostgreSQL JDBC 驱动：

   ```
   cd /opt/hive/lib
   # 从 PostgreSQL JDBC Driver 官网下载 (选择与 PG 版本兼容的)
   wget https://jdbc.postgresql.org/download/postgresql-42.7.3.jar # 替换为最新版链接
   ```

3. **配置 Hive (在 server1 上):**
   - 配置 Hive 环境变量 (可选，方便命令行): 编辑 `~/.bashrc`

   ```
   nano ~/.bashrc
   export HIVE_HOME=/opt/hive
   export PATH=$PATH:$HIVE_HOME/bin
   source ~/.bashrc
   echo $HIVE_HOME
   ```

   - 创建 Hive 配置文件 `hive-site.xml`：

   ```
   cd /opt/hive/conf
   cp hive-default.xml.template hive-site.xml
   nano hive-site.xml
   ```

   - 在 `<configuration>` 标签内添加或修改以下核心配置：

```
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:postgresql://server1:5432/metastore</value> <description>JDBC connect string for a JDBC metastore</d
    </property>

    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>org.postgresql.Driver</value>
        <description>Driver class name for a JDBC metastore</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>hiveuser</value> <description>Username to use against metastore database</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hivepassword</value> <description>Password to use against metastore database</description>
    </property>

    <property>
        <name>hive.exec.scratchdir</name>
        <value>/tmp/hive</value> <description>HDFS scratch directory for Hive jobs</description>
    </property>

    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value> <description>Location of default database for the warehouse</description>
    </property>

    <property>
        <name>hive.execution.engine</name>
        <value>mr</value> <description>Expects one of [mr, tez, spark]</description>
    </property>

    <property>
      <name>hive.server2.thrift.port</name>
      <value>10000</value>
    </property>
    <property>
      <name>hive.server2.thrift.bind.host</name>
      <value>server1</value> </property>
     <property>
       <name>hive.server2.enable.doAs</name>
       <value>false</value> </property>

     <property>
       <name>hive.hadoop.classpath</name>
       <value>${env:HADOOP_HOME}/share/hadoop/common/*:${env:HADOOP_HOME}/share/hadoop/common/lib/*:${env:HADOOP_HOME}/s
     </property>

     <property>
       <name>hive.server2.materializedviews.registry.impl</name>
       <value>NONE</value>
     </property>
     <property>
       <name>hive.aux.jars.path</name>
       <value></value>
     </property>


</configuration>
```

- **(重要) 处理 Guava 版本冲突**: Hive 和 Hadoop 可能依赖不同版本的 Guava。检查 `/opt/hive/lib` 和 `/opt/hadoop/share/hadoop/common/lib` 下的 `guava-*.jar`。通常需要将 Hive lib 目录下的 Guava jar 替换为 Hadoop 使用的版本，或者删除其中一个（通常是 Hive 的）。
  - 查看 Hadoop Guava 版本: `ls /opt/hadoop/share/hadoop/common/lib/guava*`
  - 查看 Hive Guava 版本: `ls /opt/hive/lib/guava*`
  - 如果版本不同，删除或重命名 Hive 的 Guava jar: `mv /opt/hive/lib/guava-<version>.jar /opt/hive/lib/guava-<version>.jar.bak`

4. **创建 HDFS 目录并设置权限 (在 server1 上)**:

```
hdfs dfs -mkdir -p /tmp/hive
hdfs dfs -mkdir -p /user/hive/warehouse
hdfs dfs -chmod g+w /tmp # Hadoop 临时目录权限
hdfs dfs -chmod g+w /tmp/hive
hdfs dfs -chmod g+w /user/hive/warehouse
# 确保运行 Hive 的用户 (hadoop) 对这些目录有写权限
hdfs dfs -chown hadoop:hadoop /tmp/hive
hdfs dfs -chown hadoop:hadoop /user/hive/warehouse
```

5. **初始化 Hive Metastore Schema (在 server1 上)**:

```
cd /opt/hive
# 使用 schematool 初始化数据库
bin/schematool -dbType postgres -initSchema
# 看到 Schema initialization finished successfully 表示成功
```

6. **启动 Hive Metastore 和 HiveServer2 (在 server1 上)**:
   - **启动 Metastore (后台运行)**:

     ```
     cd /opt/hive
     nohup bin/hive --service metastore > /opt/hive/logs/metastore.log 2>&1 &
     # 检查 metastore.log 和 jps 确认 Metastore 进程启动
     ```

   - **启动 HiveServer2 (后台运行)**:

     ```
     cd /opt/hive
     nohup bin/hive --service hiveserver2 > /opt/hive/logs/hiveserver2.log 2>&1 &
     # 检查 hiveserver2.log 和 jps 确认 HiveServer2 进程启动
     ```

7. **验证 Hive**:
   - 使用 Beeline 客户端连接 HiveServer2 (可以在 server1 或其他安装了 Hive 客户端的节点)：

     ```
     cd /opt/hive
     bin/beeline -u jdbc:hive2://server1:10000 -n hadoop # -n 指定用户名，如果启用了认证则需要密码
     ```

   - 在 Beeline 中执行 HQL 命令：

     ```
     show databases;
     create table test_hive (id int, name string) row format delimited fields terminated by ',';
     show tables;
     insert into test_hive values (1, 'hive');
     select * from test_hive;
     quit;
     ```

   - 检查 HDFS 上是否生成了表对应的目录: `hdfs dfs -ls /user/hive/warehouse/test_hive`

---

6. 安装 Spark

选择与您的 Hadoop 版本兼容的 Spark 版本 (例如 Spark 3.x.x)。这里配置 Spark on YARN 模式。

1. **下载 Spark**: 在 `server1` 上执行：

```
cd /opt
# 访问 Apache Spark 官网，选择 Pre-built for Apache Hadoop <version> 的二进制包
wget https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz # 替换为兼容版本链接
tar -zxvf spark-3.5.1-bin-hadoop3.tgz
ln -s spark-3.5.1-bin-hadoop3 spark
sudo chown -R hadoop:hadoop /opt/spark-3.5.1-bin-hadoop3 /opt/spark
```

2. **配置 Spark**:
   - 配置 Spark 环境变量 (在 **所有** 节点上编辑 `~/.bashrc`)：

     ```
     nano ~/.bashrc
     export SPARK_HOME=/opt/spark
     export PATH=$PATH:$SPARK_HOME/bin
     source ~/.bashrc
     echo $SPARK_HOME
     ```

- 配置 `spark-env.sh` (在 `server1` 上配置，然后分发)：

  ```
  cd /opt/spark/conf
  cp spark-env.sh.template spark-env.sh
  nano spark-env.sh
  ```

  添加以下内容：

  ```
  #!/usr/bin/env bash
  export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64 # 确认 Java 路径
  export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop      # 指向 Hadoop 配置目录
  export YARN_CONF_DIR=/opt/hadoop/etc/hadoop        # 指向 YARN 配置目录
  # export SPARK_MASTER_HOST=server1 # 仅在 Standalone 模式下需要指定 Master 地址
  # export SPARK_WORKER_MEMORY=100g # Standalone 模式 Worker 内存
  # export SPARK_WORKER_CORES=14    # Standalone 模式 Worker 核数
  # Spark History Server 配置 (可选)
  export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.fs.logDirectory=hdfs://server1:9000/user/spark/
  ```

- 配置 `spark-defaults.conf` (可选，设置默认属性，在 `server1` 配置后分发)：

  ```
  cp spark-defaults.conf.template spark-defaults.conf
  nano spark-defaults.conf
  ```

  添加常用配置：

  ```
  # 使用 YARN 作为 Master
  spark.master                     yarn
  # 事件日志记录 (用于 History Server)
  spark.eventLog.enabled           true
  spark.eventLog.dir               hdfs://server1:9000/user/spark/applicationHistory # 需要在 HDFS 创建
  spark.yarn.historyServer.address server1:18080
  # 序列化器
  spark.serializer                 org.apache.spark.serializer.KryoSerializer
  # Driver/Executor 内存和核数 (根据需要调整)
  # spark.driver.memory              4g
  # spark.executor.memory            8g
  # spark.executor.cores             2
  ```

3. **创建 Spark History Server 所需的 HDFS 目录 (在 server1 上):**

   ```
   hdfs dfs -mkdir -p /user/spark/applicationHistory
   hdfs dfs -chown -R hadoop:hadoop /user/spark
   hdfs dfs -chmod 1777 /user/spark/applicationHistory # 设置 sticky bit
   ```

4. **分发 Spark 到其他节点:** 在 `server1` 上执行：

   ```
   cd /opt
   scp -r spark-3.5.1-bin-hadoop3 server2:/opt/
   scp -r spark-3.5.1-bin-hadoop3 server3:/opt/
   # 在 server2 和 server3 上创建软链接和修改权限
   ssh server2 "sudo ln -s /opt/spark-3.5.1-bin-hadoop3 /opt/spark && sudo chown -R hadoop:hadoop /opt/spark-3.5.1-bin-hadoop3 /o
   ssh server3 "sudo ln -s /opt/spark-3.5.1-bin-hadoop3 /opt/spark && sudo chown -R hadoop:hadoop /opt/spark-3.5.1-bin-hadoop3 /o
   # 分发配置文件
   scp /opt/spark/conf/spark-env.sh server2:/opt/spark/conf/
   scp /opt/spark/conf/spark-env.sh server3:/opt/spark/conf/
   scp /opt/spark/conf/spark-defaults.conf server2:/opt/spark/conf/
   scp /opt/spark/conf/spark-defaults.conf server3:/opt/spark/conf/
   # **重要**: 确保所有节点的 .bashrc 都已配置并 source
   ssh server2 'source ~/.bashrc'
   ssh server3 'source ~/.bashrc'
   ```

5. **启动 Spark History Server (在 server1 上):**

   ```
   cd /opt/spark
   sbin/start-history-server.sh
   # 使用 jps 查看 HistoryServer 进程
   # 访问 Web UI: http://server1:18080
   ```

6. **验证 Spark on YARN:**

   - **运行 Spark Shell (在任意节点):**

```
spark-shell --master yarn --deploy-mode client # 或者 --deploy-mode cluster
```

启动时会向 YARN 申请资源。进入 Scala REPL 后，可以执行简单的 Spark 代码：

```
val data = Array(1, 2, 3, 4, 5)
val distData = sc.parallelize(data)
distData.count()
// 读取 HDFS 文件
val textFile = sc.textFile("hdfs://server1:9000/test/test.txt")
textFile.first()
// 退出
:quit
```

在 YARN UI (`http://server1:8088`) 上可以看到运行的 Spark Application。运行结束后，可以在 Spark History Server UI (`http://server1:18080`) 查看历史记录。

- **运行 Spark Pi 示例 (在任意节点)**:

```
cd /opt/spark
bin/spark-submit --class org.apache.spark.examples.SparkPi \
    --master yarn \
    --deploy-mode cluster \
    --driver-memory 1g \
    --executor-memory 1g \
    --executor-cores 1 \
    examples/jars/spark-examples*.jar 10
```

查看 YARN UI 和 History Server UI。

---

7. 验证与后续步骤

1. **全面检查**:
   - 再次检查所有服务的 Web UI (HDFS, YARN, Spark History Server) 是否正常。
   - 使用 `jps` 在每个节点上检查所有预期的 Java 进程是否都在运行。
   - 检查各服务的日志文件 (通常在各自安装目录的 `logs` 子目录下或配置的日志目录中) 是否有错误。
2. **停止服务**:
   - 停止 Spark History Server: `/opt/spark/sbin/stop-history-server.sh` (在 server1)
   - 停止 Hive: `kill <pid_of_metastore>` 和 `kill <pid_of_hiveserver2>` (在 server1, 使用 `jps` 查找 pid)
   - 停止 Kafka: 在 **所有** Kafka 节点执行 `/opt/kafka/bin/kafka-server-stop.sh`
   - 停止 Hadoop: `/opt/hadoop/sbin/stop-yarn.sh` 然后 `/opt/hadoop/sbin/stop-dfs.sh` (在 server1)
   - 停止 Zookeeper: 在 **所有** Zookeeper 节点执行 `/opt/zookeeper/bin/zkServer.sh stop`
3. **启动服务 (按依赖顺序)**:
   - 启动 Zookeeper: 在 **所有** Zookeeper 节点执行 `/opt/zookeeper/bin/zkServer.sh start`
   - 启动 Hadoop: `/opt/hadoop/sbin/start-dfs.sh` 然后 `/opt/hadoop/sbin/start-yarn.sh` (在 server1) (可选启动 `/opt/hadoop/sbin/mr-jobhistory-daemon.sh start historyserver`)
   - 启动 Kafka: 在 **所有** Kafka 节点执行 `/opt/kafka/bin/kafka-server-start.sh -daemon config/server.properties`
   - 启动 Hive Metastore 和 HiveServer2 (在 server1):

```
cd /opt/hive
nohup bin/hive --service metastore > /opt/hive/logs/metastore.log 2>&1 &
nohup bin/hive --service hiveserver2 > /opt/hive/logs/hiveserver2.log 2>&1 &
```

   - 启动 Spark History Server: `/opt/spark/sbin/start-history-server.sh` (在 server1)
4. **后续配置与优化**:
   - **资源调优**: 根据您的硬件和工作负载，仔细调整 YARN (`yarn-site.xml`) 和 Spark (`spark-defaults.conf` , 提交任务时的参数) 的内存、CPU 配置。
   - **高可用 (HA)**: 为 NameNode 和 ResourceManager 配置 HA，需要 Zookeeper 支持。
   - **安全性**: 配置 Kerberos 认证以提高集群安全性。
   - **监控**: 部署监控系统，如 Ganglia、Prometheus + Grafana 或 Ambari (如果选择发行版)。
   - **日志管理**: 配置更集中的日志管理方案。
   - **备份**: 为 HDFS NameNode 元数据和 Hive Metastore 数据库配置定期备份。
   - **Tez/Spark on Hive**: 配置 Hive 使用 Tez 或 Spark 作为执行引擎以获得更好的性能。

这份指南提供了详细的步骤，但实际操作中可能遇到各种问题 (版本兼容性、配置错误、网络问题等)。请务必仔细阅读官方文档，并根据日志信息进行排错。祝您配置顺利！