CDH离线环境搭建

目录

[一、 简介（CDH） 2](#_Toc46595937)

[1. CDH 是什么 2](#_Toc46595938)

[2. CDH 优点是什么 2](#_Toc46595939)

[3. CDH组件有哪些 2](#_Toc46595940)

[二、 版本环境（CDH） 2](#_Toc46595941)

[1. TLS版本支持 2](#_Toc46595942)

[2. 软件依赖 9](#_Toc46595943)

[三、 环境准备（CDH） 11](#_Toc46595944)

[1. 安装前准备 11](#_Toc46595945)

[2. 安装顺序 12](#_Toc46595946)

[四、 安装服务（CDH） 12](#_Toc46595947)

[1. 关闭seLinux （所有机器） 12](#_Toc46595948)

[2. 关闭防火墙 （所有机器） 12](#_Toc46595949)

[3. 设置hostname （所有机器） 12](#_Toc46595950)

[4. 修改hosts （所有机器） 12](#_Toc46595951)

[5. 配置免密登录 （所有机器） 13](#_Toc46595952)

[6. 配置文件yum源 （一台主节点） 13](#_Toc46595953)

[7. 安装httpd 13](#_Toc46595954)

[8. 重新配置yum源 14](#_Toc46595955)

[9. 安装gcc 、python、ntp （所有机器） 14](#_Toc46595956)

[10. 安装jdk（oracle 版本rpm安装，所有机器） 16](#_Toc46595957)

[11. 安装mysql （任意一台节点） 16](#_Toc46595958)

[12. 通过yum 安装cloudera-scm-server 18](#_Toc46595959)

[13. 登录页面配置 21](#_Toc46595960)

[14. 系统文件存储结构 24](#_Toc46595961)

[15. 硬件选择 26](#_Toc46595962)

[16. 角色及节点分配 30](#_Toc46595963)

[五、 Data-web安装 38](#_Toc46595964)

[1. 安装前准备 38](#_Toc46595965)

[2. 安装服务 38](#_Toc46595966)

[3. 测试是否安装成功 39](#_Toc46595967)

[六、 CDH与datax-web使用和注意事项 40](#_Toc46595968)

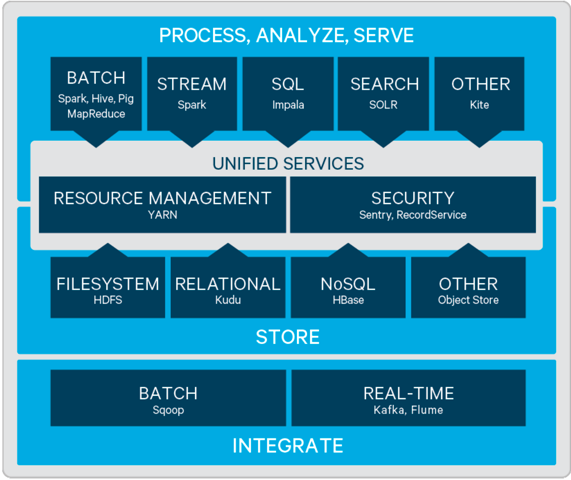
# 简介（CDH）

## CDH 是什么

CDH是Apache Hadoop和相关项目最完整、最受测试和最受欢迎的发行版。CDH提供了Hadoop的核心元素—可扩展的存储和分布式计算—以及基于Web的用户界面和重要的企业功能。CDH是Apache许可的开放源码，是唯一提供统一批处理、交互式SQL和交互式搜索以及基于角色的访问控制的Hadoop解决方案。

## CDH 优点是什么

* 灵活性：可以存储任何类型的数据，并使用各种不同的计算框架对其进行操作，包括批处理、交互式SQL、自由文本搜索、机器学习和统计计算。
* 整合：集成在一个完整的Hadoop平台上快速启动和运行，该平台与广泛的硬件和软件解决方案一起工作。
* 安全：过程控制敏感数据。
* 可扩展性：使应用程序和扩展范围更广，并扩展它们以满足您的需求。
* 高可用性：信任并执行关键业务任务。
* 兼容性：可以利用现有的IT基础架构和资源。



## CDH组件有哪些

Apache Avro、Apache Crunch、Apache Flume、Apache Hadoop、Apache HBase、Apache Hive、Hue、Kite、Apache Impala、Apache Oozie、Apache Parquet、Apache Pig、Apache Sentry、Apache Solr、Apache Spark、Apache Sqoop、Apache ZooKeeper等。

# 版本环境（CDH）

## TLS版本支持

* CDH和Cloudera Manager支持的传输层安全版本

指定的传输层安全性（TLS）版本支持以下组件：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 组件 | 角色 | 名称 | 端口 | 版本号 |
| Cloudera Manager | Cloudera Manager Server |  | 7182 | TLS 1.2 |
| Cloudera Manager | Cloudera Manager Server |  | 7183 | TLS 1.2 |
| Flume |  |  | 9099 | TLS 1.2 |
| Flume |  | Avro Source/Sink |  | TLS 1.2 |
| Flume |  | Flume HTTP Source/Sink |  | TLS 1.2 |
| HBase | Master | HBase Master Web UI Port | 60010 | TLS 1.2 |
| HDFS | NameNode | Secure NameNode Web UI Port | 50470 | TLS 1.2 |
| HDFS | Secondary NameNode | Secure Secondary NameNode Web UI Port | 50495 | TLS 1.2 |
| HDFS | HttpFS | REST Port | 14000 | TLS 1.1, TLS 1.2 |
| Hive | HiveServer2 | HiveServer2 Port | 10000 | TLS 1.2 |
| Hue | Hue Server | Hue HTTP Port | 8888 | TLS 1.2 |
| Impala | Impala Daemon | Impala Daemon Beeswax Port | 21000 | TLS 1.2 |
| Impala | Impala Daemon | Impala Daemon HiveServer2 Port | 21050 | TLS 1.2 |
| Impala | Impala Daemon | Impala Daemon Backend Port | 22000 | TLS 1.2 |
| Impala | Impala StateStore | StateStore Service Port | 24000 | TLS 1.2 |
| Impala | Impala Daemon | Impala Daemon HTTP Server Port | 25000 | TLS 1.2 |
| Impala | Impala StateStore | StateStore HTTP Server Port | 25010 | TLS 1.2 |
| Impala | Impala Catalog Server | Catalog Server HTTP Server Port | 25020 | TLS 1.2 |
| Impala | Impala Catalog Server | Catalog Server Service Port | 26000 | TLS 1.2 |
| Oozie | Oozie Server | Oozie HTTPS Port | 11443 | TLS 1.1, TLS 1.2 |
| Solr | Solr Server | Solr HTTP Port | 8983 | TLS 1.1, TLS 1.2 |
| Solr | Solr Server | Solr HTTPS Port | 8985 | TLS 1.1, TLS 1.2 |
| Spark | History Server |  | 18080 | TLS 1.2 |
| YARN | ResourceManager | ResourceManager Web Application HTTP Port | 8090 | TLS 1.2 |
| YARN | JobHistory Server | MRv1 JobHistory Web Application HTTP Port | 19890 | TLS 1.2 |

硬件支持

* **Cloudera Manager Server 存储需求**

|  |  |  |
| --- | --- | --- |
| 位置 | 存储大小 | 备注 |
| /usr | 1 GB |  |
| /var | 5 GB 至 1 TB | 根据管理的节点数进行缩放 |
| /opt | 大于15 GB | 使用率随着下载的包裹数量的增加而增加 |
| Cloudera Manager Database Server | 5 GB | 如果Cloudera Manager数据库与服务监视器和主机监视器共享，则需要更多的存储空间来满足这些组件的要求。 |

* **基于主机的Cloudera Manager服务器要求**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 群集主机数 | 数据库主机配置 | 堆大小 | 逻辑处理器个数 | Cloudera Manager Server /var 目录大小 |
| 非常小集群 (≤10) | 共享 | 2 GB | 4 | 5 GB |
| 小集群 (≤20) | 共享 | 4 GB | 6 | >=20 GB |
| 中等集群 (≤200) | 专用 | 8 GB | 6 | >=200 GB |
| 大型集群 (≤500) | 专用 | 10 GB | 8 | >=500 GB |
| 超大集群 (>500) | 专用 | 16 GB | 16 | >=1 TB |

* **带有 HDFS, YARN, or Impala 的集群**

|  |  |  |  |
| --- | --- | --- | --- |
| 监控的实体对象个数 | 主机数量 | Java Heap Size | 推荐 Non-Java Heap Size |
| 0-2,000 | 0-100 | 1 GB | 6 GB |
| 2,000-4,000 | 100-200 | 1.5 GB | 6 GB |
| 4,000-8,000 | 200-400 | 1.5 GB | 12 GB |
| 8,000-16,000 | 400-800 | 2.5 GB | 12 GB |
| 16,000-20,000 | 800-1,000 | 3.5 GB | 12 GB |

* **Flume**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | 磁盘 |
| Flume | 最小 1 GB最大 4 GB  Java堆大小应该大于最大信道容量。  使用代理的Java堆大小设置该值，以字节为单位的Flume配置属性。 | 使用以下公式计算芯数：（源数量+Flume数量）/2 | 建议将多个磁盘用于文件通道，既可以是JBOD设置，也可以是RAID10（由于可靠性提高，因此首选）。 |

* **HDFS**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Memory | CPU | 磁盘数 |
| JournalNode | 1 GB (默认)使用JavaLoad的Java堆大小设置该值，以字节为HDFS配置属性。 | 最小1 core | 1 个专用 磁盘 |
| NameNode | 最小： 1 GB (用于概念验证部署)  为每增加1000000个块添加一个额外的1 GB  快照和加密可以增加所需的堆内存。  使用名为HDFS配置属性的NAMENODE的Java堆大小设置此值。 | 至少4个专用核心；对于较大的集群，可能需要更多 | 至少2个元数据专用磁盘  1个日志文件专用磁盘（此磁盘可以与操作系统共享）  最大4个磁盘 |
| DataNode | 最少: 4 GB  增加内存以获得更高的副本计数或每个数据节点的块数。当增加内存时，Cloudera建议每100万个复制副本（在数据节点上超过400万个）增加1 GB内存。例如，500万个副本需要5 GB的内存。  使用DATION的Java堆大小设置此值，以字节为HDFS配置属性。 | 最小: 4 核. 为高度活跃的集群添加更多的核心 | 最小: 4  最大: 24  最大可接受大小将根据平均块大小的大小而变化。DN的可伸缩性限制主要是每个DN的副本数量的函数，而不是存储的总字节数。也就是说，如果机器或机架发生故障，拥有超密集的DNS将影响恢复时间。Cloudera不支持每个数据节点超过100 TB。您可以使用12 x 8 TB主轴或24 x 4TB主轴。Cloudera不支持大于8 TB的驱动器。 |

* **Hbase**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | 磁盘 |
| Master | 100-10,000个 regions需要 4 GB  拥有200个或更多Region Servers的10000个或更多Region: 8 GB  拥有300个或更多Region Servers的10000个或更多Region: 12 GB  使用HBASE主控器的Java堆大小设置此值，以字节为基础配置属性。 | 至少4个专用核心。在使用复制或批量加载时，可以为较大的群集添加更多的核心。 | 1个本地日志磁盘，可与操作系统和/或其他Hadoop日志共享 |
| Region Server | 最少: 8 GB  中型: 16 GB  大于16GB的堆需要特殊的垃圾收集优化  使用HbaseRegionServer的Java堆大小设置此值，以字节为基础配置属性。 | 至少4个专用核心 | 每个HDFS DataNode 至少有4个或者更多  1个用于本地日志的磁盘（此磁盘可以与操作系统和/或其他Hadoop日志共享 |

* **Hive**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | Java Heap | | CPU | 磁盘 |
| HiveServer 2 | 单连接 | 4 GB | 至少4个专用核 | 至少1个  以下操作需要此磁盘：  HiveServer2 日志文件  stdout 和 stderr 输出文件  配置文件  存储在operation\_logs\_dir目录中的操作日志，可配置  可能由/tmp目录下的本地映射任务创建的任何临时文件 |
| 2-10个连接数 | 4-6 GB |
| 11-20 个连接数 | 6-12 GB |
| 21-40 个连接数 | 12-16 GB |
| 41 to 80 个连接数 | 16-24 GB |
| Cloudera建议将HiveServer 2拆分为多个实例，并在开始向HiveServer 2分配超过16GB的空间后对其进行负载平衡。目标是调整大小以减少Java垃圾收集对服务的主动处理的影响。 | |
| 使用HiveServer2的Java堆大小设置这个值，以字节为单位的HIVE配置属性。 | |
| Hive Metastore | 单连接 | 4 GB | 至少4个专用核 | 至少1个  此磁盘是必需的，因此配置单元元存储可以存储以下项目：  日志  配置文件  如果数据库服务器也托管在同一节点上，则用于存储元数据的后端数据库 |
| 2-10个连接数 | 4-10 GB |
| 11-20 个连接数 | 10-12 GB |
| 21-40 个连接数 | 12-16 GB |
| 41 to 80 个连接数 | 16-24 GB |
| 使用Hive Metastore Server的Java堆大小设置该值，以字节为单位的HIVE配置属性。 | |
| Beeline CLI | 至少: 2 GB | | N/A | N/A |

* **Hive on Spark 执行节点**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Memory | CPU | 磁盘 |
| Hive-on-Spark | 至少: 16 GB  推荐: 32 GB 用于更大的数据  单个执行器堆不应大于16GB，因此具有更多RAM的计算机可以使用多个执行器。 | 至少 4 核  建议：8个核心用于更大的数据大小 | 磁盘空间要求由Spark spill的空间要求驱动。 |

* **Spark**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | Disk |
| Spark History Server | 至少: 512 MB  使用Spark History Server的Java堆大小设置此值，以字节配置Spice配置属性。 | 1个  Cloudera建议您根据特定的集群使用模式调整Spark History服务器的CPU和内存数量。 | 至少1个磁盘用于存放日志 |

* **YARN**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | 其他建议 |
| Job History Server | 至少: 1 GB  对于内存中保存的每100000个任务，将内存增加1.6 GB。例如：  5个作业@100000个  mappers+20000个  reducers =600000个总任务需要9.6GB的堆。  使用Job History Server服务器的Java堆大小设置此值，以字节配置纱线配置属性。 | 至少: 1 核 | 将mapreduce.jobhistory.jhist.format属性设置为binary（使用此设置，历史文件的加载速度将加快2-3倍）  将mapreduce.jobhistory.loadedtasks.cache.size属性设置为总加载任务计数。使用Java堆栈列中的示例，在650000个总任务中，可以将其设置为700000，以允许某些安全余量。这还应防止在垃圾收集期间挂起JobHistoryServer，因为作业计数限制没有任务限制。 |
| NodeManager | 至少: 1 GB.  为以下条件配置其他堆内存：  大量容器  大的 shiffle  在 Spark  或者 MapReduce | 至少8-16 核  推荐: 32-64 核 | 磁盘:  至少: 8 个磁盘  推荐: 12 或者更多  网络:  至少: 双1Gbps或更快  推荐: 单/双10 Gbps的速度。 |
| ResourceManager | 至少: 6 GB  为以下条件配置其他堆内存：  更多 jobs  更大的 cluster  保留的已完成应用程序数（使用yarn.resourcemanager.max-completed-applications属性配置）。  调度程序配置  使用ResourceManager的Java堆大小设置该值，以字节为单位的纱线配置属性。 | 至少: 1 核 |  |
| 其他设置 | 将ApplicationMaster Memory YARN配置属性设置为512 MB  将Container Memory Minimum YARN配置属性设置为1 GB。 | N/A | N/A |

* **ZooKeeper**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | 磁盘 |
| ZooKeeper Server | 至少: 1 GB  当观察10000-100000个短暂的znode并且使用1000个或更多的客户机时，增加堆大小。  使用ZooKeeper Server的Java堆大小设置这个值，以字节为单位的ZooKeeper配置属性。 | 至少: 4 核 | ZooKeeper并不是为低延迟服务而设计的，也不能从使用SSD驱动器中获益。ZooKeeper访问模式（仅附加写入和顺序读取）的设计考虑了旋转磁盘。因此，Cloudera建议使用HDD驱动器。 |

* **Oozie**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Java Heap | CPU | 磁盘 |
| Oozie | 至少: 1 GB (这是由Cloudera Manager设置的默认值). 这对于不到10个同时工作流程是足够的，而不需要分叉。  如果发现垃圾收集过多或内存不足错误，请将堆大小增加到4 GB（对于中型生产群集）或8 GB（对于大型生产群集）。  使用Oozie server的Java堆大小设置该值，以字节为Oozie配置属性。 | 不需要资源 | 无需资源 |
| 额外调整 | 对于使用复杂工作流运行的多个协调器的工作负载（已达到最大并发性！）日志中出现警告，oozie admin-queuedump命令显示一个大队列）：  1.将oozie.service.callablequeueservice.callable.concurrency属性的值增加到50。  2.将oozie.service.callablequeueservice.threads属性的值增加到200。  3.不要将Derby数据库用作Oozie的后端数据库。 |  |  |

* **Impala**

根据使用IMPALA的工作量大小和类型，IMPALA的尺寸要求可能会有很大的不同。

对于多机架集群的网络拓扑结构，建议使用 [Leaf-Spine](https://blog.westmonroepartners.com/a-beginners-guide-to-understanding-the-leaf-spine-network-topology/)以获得最佳性能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 本机存储 | JVM Heap | CPU | 磁盘 |
| Impala Daemon | 设置 Impala Daemon 内存属性.  至少: 32 GB  推荐: 128 GB | 设置Java Heap Size of Impala Daemon in Bytes 守护进程属性.  至少: 4 GB  推荐: 8 GB | 至少: 4  推荐: 16 或者更多  CPU指令设置为: AVX2 | 至少: 1 个  推荐: 8 或者更多 |
| Catalog Server | 设置 Java Heap Size of Catalog Server in Bytes 配置属性.  至少: 4 GB  推荐: 8 GB | | 至少: 4  推荐: 16 或者更多  CPU 指令设置成: AVX2 | 至少并对讲: 1 个 |

* **Kafka**

kafka需要相当少量的资源，特别是在一些配置调整中。默认情况下，Kafka可以在1核和1GB内存上运行，存储容量根据数据保留的要求而定。

CPU很少成为瓶颈，因为Kafka的I/O很重，但是具有足够线程的中等大小的CPU对于处理并发连接和后台任务仍然很重要。

Kafka代理往往具有与HDFS数据节点类似的硬件配置文件。如何构建它们取决于什么对您的Kafka用例很重要。使用以下准则：

|  |  |
| --- | --- |
| 要影响这些功能的性能： | 调整这些参数： |
| 消息保留 | 磁盘大小 |
| 客户吞吐量（生产者和消费者） | 网络容量 |
| 生产者吞吐量 | 磁盘 I/O |
| 消费者吞吐量 | 内存 |

Kafka节点的常见选择如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 内存/Java Heap | CPU | 磁盘 |
| Broker | RAM: 64 GB推荐Java heap: 4 GB使用Broker Kafka配置属性的Java堆大小设置此值。 | 12- 24 核 | 操作系统用1个硬盘   Zookeeper dataLogDir 用一个硬盘  存储kafka 用10- HDDs, 使用 Raid 10 |
| MirrorMaker | 1 GB heap使用MirrorMaker Kafka配置属性的Java堆大小设置此值。 | 1核3～4 streams | MirrorMaker实例上不需要磁盘空间。目标代理应该有足够的磁盘空间来存储要复制的主题。 |

## 软件依赖

* **iproute 版本**

Python：Cloudera enterprise 6（色调除外）在操作系统中默认包含的python版本以及更高版本上受支持，但与python 3.0或更高版本不兼容。例如，Cloudera Enterprise 6要求在与RHEL 6兼容的操作系统上安装python 2.6或更高版本，但在与RHEL 7兼容的操作系统上安装python 2.7或更高版本。CDH 6中的色调在所有操作系统上都需要python 2.7或更高版本。对于运行hue的RHEL6兼容操作系统，必须手动安装python 2.7。不支持python 3。

Perl: Cloudera Manager需要perl。

python-psycopg2：Cloudera manager 6依赖于包python-psycopg2。CDH 6中的色调需要的psycopg2版本高于Cloudera Manager依赖项所需的版本。

iproute package:Cloudera Enterprise 6依赖于iproute包。运行Cloudera Manager代理的任何主机都需要该包。所需版本因操作系统而异:

|  |  |
| --- | --- |
| 操作系统 | iproute 版本 |
| RHEL 7 Compatible | iproute-3.10 |
| RHEL 6 Compatible | iproute-2.6 |
| SLES 12 SP2, SP3 | iproute2-3.12 |

* **Cloudera Enterprise 6.1.x 支持的操作系统**

|  |  |
| --- | --- |
| 操作系统 | 版本号 |
| RHEL-compatible | |
| RHEL/CentOS/OL with RHCK kernel | **7.6**, 7.5, 7.4, 7.3, 7.2  6.10, 6.9 , 6.8 |
| Oracle Linux (OL) | 7.4, 7.3, 7.2 (UEK default)  **6.10** (UEK default) |
| SUSE Linux Enterprise Server | |
| SLES | 12 SP3, 12 SP2 |
| Ubuntu | |
| Ubuntu | 16.04 LTS (Xenial) |

* **数据库对应版本**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cloudera Enterprise | MySQL | MariaDB | PostgreSQL | Oracle |
| 6.x | 5.7 | 5.5、10.0 |  | 12.2 |
| 6.0 |  |  | 8.4、9.2、9.4 |  |
| 6.1 |  |  | 8.4、9.2、9.4、10.0 |  |

* **JAVA JDK版本**

Java优化称为压缩OOP（普通对象指针），使64位JVM能够使用4字节指针来处理堆大小达到大约32 GB。对于较大的堆大小，需要8字节指针。这意味着稍微小于32GB的堆可以容纳比稍微大于32GB的堆大小更多的对象。

如果不需要超过32GB的堆，请将堆大小设置为31GB或更低以避免此问题。如果您需要32 GB或更大的内存，请将堆大小设置为48 GB或更大，以考虑更大的指针。通常，对于大于32GB的堆大小，将需要的堆数量乘以1.5。

仅支持64位JDK。Cloudera Manager 6和CDH 6不支持JDK 7。尽管所有版本的cdh 5都支持JDK7，但由ClouderaManager6.x管理的cdh 5.x集群必须在所有集群主机上使用JDK8。Cloudera Manager 6和CDH 6支持Oracle JDK 8。CDH 5.3及更高版本也支持JDK 8。

OpenJDK 8支持Cloudera Enterprise 6.1.0及更高版本，以及Cloudera Enterprise 5.16.1及更高版本。

CDH 6不支持用JDK 7编译的应用程序。升级到CDH 6之前，必须使用JDK 8重新编译应用程序。

任何Cloudera Manager或CDH版本都不支持Oracle JDK 9。

不支持在不同的JDK版本上运行同一集群中的CDH节点。所有集群主机必须使用相同的JDK更新级别。

Cloudera Manager 6和CDH 6需要JDK 8

由于JDK问题影响到CDH功能，因此不支持JDK 8u40, 8u45,和 8u60。

**Oracle JDK 8**

|  |  |
| --- | --- |
| Oracle JDK版本 | 备注 |
| 1.8u181 | 推荐 / 最新版被测试 |
| 1.8u162 | 推荐 |
| 1.8u141 | 推荐 |
| 1.8u131 | 推荐 |
| 1.8u121 | 推荐 |
| 1.8u111 | 推荐 |
| 1.8u102 | 推荐 |
| 1.8u91 | 推荐 |
| 1.8u74 | 推荐 |
| 1.8u31 | 最低要求 |

# 环境准备（CDH）

## 安装前准备

* JDK1.8（Oracle版本RPM安装）
* Cloudera Manager 6.2.0
* Linux 系统三台以上（centos 7X 以上 16G以上两台、32G以上一台）
* Linux 镜像（必须与系统版本对应）
* MYSQL 5.7
* Python 2.7
* CDH 6.2.0

<https://archive.cloudera.com/cdh6/6.2.0/parcels/>

[CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel](https://archive.cloudera.com/cdh6/6.2.0/parcels/CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel)

[CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel.sha1](https://archive.cloudera.com/cdh6/6.2.0/parcels/CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel.sha1)

[CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel.sha256](https://archive.cloudera.com/cdh6/6.2.0/parcels/CDH-6.2.0-1.cdh6.2.0.p0.967373-el7.parcel.sha256)

[manifest.json](https://archive.cloudera.com/cdh6/6.2.0/parcels/manifest.json)

个人镜像站下载地址：（cdh6.3.2）

<http://decs.pcl.ac.cn:3731/mirrors/software/BigData/cdh/CDH6.3.2.zip>

* Yum源
* Ntp时间同步工具
* DataX-web datax

## 安装顺序

* 关闭seLinux （所有机器）
* 关闭防火墙 （所有机器）
* 设置hostname （所有机器）
* 设置hosts （所有机器）
* 配置免密登录 （所有机器）
* 配置文件yum源 （一台主节点）
* 通过yum源安装httpd 、ntp 、gcc 、python （httpd 主节点，其他全部）
* 通过httpd 重新配置yum 源仓库
* 安装jdk （Oracle版本RPM安装）（所有机器）
* 安装MYSQL 并配置 （一台主节点）
* 通过yum 源方式安装 cloudera-scm-server
* 通过cm web端安装cloudera-scm-agent
* 配置具体服务角色

# 安装服务（CDH）

## 关闭seLinux （所有机器）

* vi /etc/selinux/config
* SELINUX=enforcing 改为 SELINUX=disabled
* 重启生效

## 关闭防火墙 （所有机器）

* Systemctl stop firewalld 关闭防火墙
* Systemctl disable firewalld 禁用防火墙
* Systemctl enable firewalld 开机自启防火墙
* Systemctl start firewalld 开启防火墙
* Systemctl status firewalld 查看防火墙

## 设置hostname （所有机器）

* 查看主机名称 hostname
* 编辑vi /etc/hostname
* 修改并保存
* 重启生效

## 修改hosts （所有机器）

* vim /etc/hosts
* 编辑一下内容，其实就是地址对应，按照各自设计的网络名称进行对应，例如：

192.168.179.129 work-1

192.168.179.130 work-2

192.168.179.131 work-3

* 执行命令：vim /etc/sysconfig/network
* 修改如下：

NETWORKING=yes

HOSTNAME=work-1

* 执行命令：service network restart重启网络服务生效。

## 配置免密登录 （所有机器）

* 生成ssh秘钥：ssh-keygen -t rsa
* Copy秘钥到其他机器：ssh-copy-id cdh01
* 测试是否免密跳转：ssh cdh01

## 配置文件yum源 （一台主节点）

* 拷贝centos系统镜像至主节点
* 镜像授权：chmod -R 777 CentOS-7-x86\_64-DVD-1810.iso
* 备份原yum配置：mv /etc/yum.repos.d/ /etc/yum.repos.d.back
* 根目录新建iso目录 ： mkdir /iso
* 挂载镜像至iso文件：mount -o loop CentOS-7-x86\_64-DVD-1810.iso /iso/
* 新建文件目录 mkdir /etc/yum.repos.d
* 进入yum源配置目录：cd /etc/yum.repos.d
* 新建centos.repo文件并编辑

[localrepo]

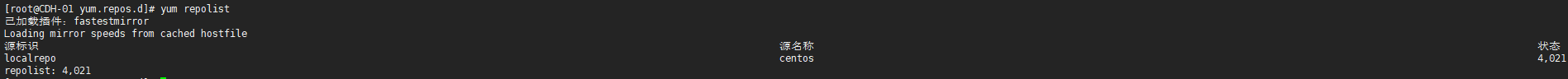
name=centos

baseurl=file:///iso

gpgcheck=false

enabled=true

* 保存并退出
* 执行：yum clean all 和 yum makecache
* 检查是否安装成功 yum repolist

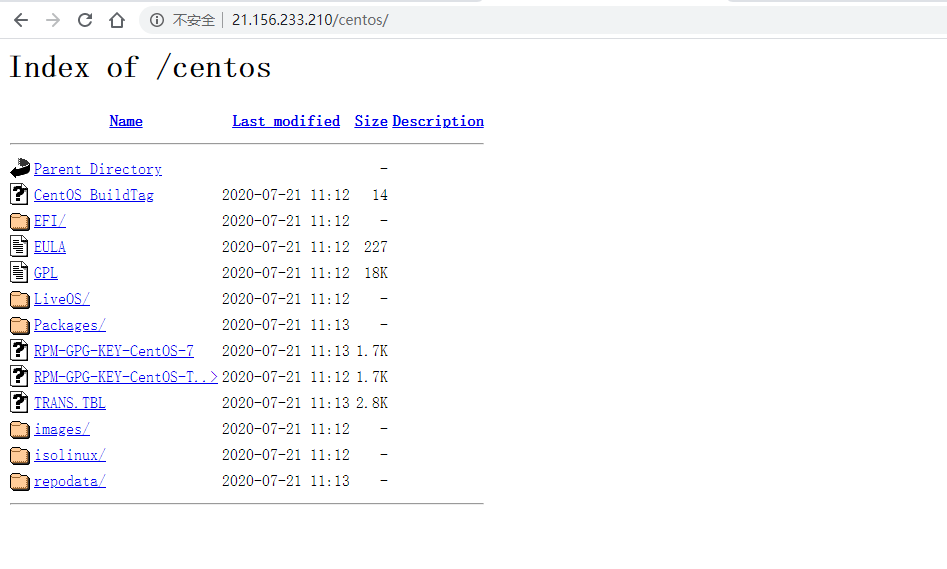


## 安装httpd

* 执行命令：yum -y install httpd
* 启动httpd ：systemctl start httpd
* 设置开机自启：systemctl enable httpd
* 进入httpd文件目录: cd /var/www/html
* 新建文件夹 centos 、cm 、cdh
* Copy 根目录iso文件夹下所有文件至/var/www/html/centos :

cp -r /iso/\* /var/www/html/centos/

* 通过页面访问 ip/centos



## 重新配置yum源

* 修改vi /etc/yum.repos.d/centos.repo

[localrepo]

name=centos

baseurl=http://21.156.233.210/centos

gpgcheck=false

enabled=true

* 保存并退出
* 执行：yum clean all 和 yum makecache
* 检查是否安装成功 yum repolist
* 移除其他机器 /etc/yum.repos.d 下所有文件
* 拷贝/etc/yum.repos.d/centos.repo至其他机器/etc/yum.repos.d
* 其他机器执行yum clean all 和 yum makecache

## 安装gcc 、python、ntp （所有机器）

* 安装gcc ：yum -y install gcc\*
* 安装gcc ：yum -y install python
* 安装ntp ：yum -y install ntp
* 修改server配置文件

vim /etc/ntp.conf

# For more information about this file, see the man pages

# ntp.conf(5), ntp\_acc(5), ntp\_auth(5), ntp\_clock(5), ntp\_misc(5), ntp\_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not

# permit the source to query or modify the service on this system.

restrict default nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could

# be tightened as well, but to do so would effect some of

# the administrative functions.

restrict 127.0.0.1

restrict ::1

# Hosts on local network are less restricted.

restrict 192.168.0.0 mask 255.255.0.0 nomodify notrap ##允许内网其他机器同步时间

# Use public servers from the pool.ntp.org project.

# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

#外部服务不可用，使用本地时间作为服务

server 127.127.1.0 #本地时间就写这个ip

fudge 127.127.1.0 stratum 10

#broadcast 192.168.1.255 autokey # broadcast server

#broadcastclient # broadcast client

#broadcast 224.0.1.1 autokey # multicast server

#multicastclient 224.0.1.1 # multicast client

#manycastserver 239.255.254.254 # manycast server

#manycastclient 239.255.254.254 autokey # manycast client

# Enable public key cryptography.

#crypto

includefile /etc/ntp/crypto/pw

# Key file containing the keys and key identifiers used when operating

# with symmetric key cryptography.

keys /etc/ntp/keys

# Specify the key identifiers which are trusted.

#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.

#requestkey 8

# Specify the key identifier to use with the ntpq utility.

#controlkey 8

# Enable writing of statistics records.

#statistics clockstats cryptostats loopstats peerstats

# Disable the monitoring facility to prevent amplification attacks using ntpdc

# monlist command when default restrict does not include the noquery flag. See

# CVE-2013-5211 for more details.

# Note: Monitoring will not be disabled with the limited restriction flag.

disable monitor

* 修改client端配置

vim /etc/ntp.conf

# For more information about this file, see the man pages

# ntp.conf(5), ntp\_acc(5), ntp\_auth(5), ntp\_clock(5), ntp\_misc(5), ntp\_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not

# permit the source to query or modify the service on this system.

restrict default nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could

# be tightened as well, but to do so would effect some of

# the administrative functions.

restrict 127.0.0.1

restrict ::1

# Hosts on local network are less restricted.

#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

##将原来的server注释掉，添加自己的ntp-server-ip

#server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

server 192.168.100.105 iburst

...

* 启动服务端并设置开机自启
* 启动客户端并设置开机自启
* Systemctl start ntpd ， systemctl enable ntpd
* 启动客户端自动同步

## 安装jdk（oracle 版本rpm安装，所有机器）

* 首先需要卸载centos自带openjdk,选择安装官方的jdk，执行命令：

rpm -qa | grep jdk

* 对查询到的java程序执行卸载命令

rpm -e --nodeps java相关名称

或

yum -y remove jdk相关

或

yum remove java\* //若有，予以删除

* CDH不会使用系统默认JAVA\_HOME环境变量，而是使用Bigtop进行管理，故我们需要安装Bigtop的规则在指定的位置安装jdk，比较麻烦，这里我们选择rpm安装即可，命令如下：

rpm -ivh jdk-8u181-linux-x64.rpm

cloudera不支持混合环境，所以集群中的所有节点都必须运行相同的主要JDK版本，Cloudera仅支持Oracle提供的JDK

对于Cloudera Manager和CDH 6.0.0及更高版本，仅支持Oracle JDK 8

## 安装mysql （任意一台节点）

* 下载Linux版mysql .tar.gz压缩包至服务器；
* 解压mysql 至指定目录：tar -zxvf xxx.mysql.tar.gz -C /usr/local/
* cd /usr/local ,修改mysql目录名称为mysql：mv ./xxxx ./mysql
* 新建用户mysql：useradd mysql
* 新建系统组 mysql：groupadd mysql
* 修改当前目录拥有者为mysql用户 chown -R mysql:mysql mysql
* 在mysql 目录下新建data目录：mkdir data
* 修改mysql配置：vi /etc/my.cnf

[mysqld]

[mysqld]

port=3306

basedir=/usr/local/mysql

datadir=/usr/local/mysql/data

socket=/usr/local/mysql/tmp/mysql.sock

symbolic-links=0

wait\_timeout = 31536000

interactive\_timeout = 31536000

character\_set\_server=utf8

init\_connect='SET NAMES utf8'

lower\_case\_table\_names = 1

[mysqld\_safe]

log-error=/usr/local/mysql/log/mysqld.log

pid-file=/usr/local/mysql/data/mysqld.pid

[client]

port=3306

socket=/usr/local/mysql/tmp/mysql.sock

* 进入mysql用户：su mysql
* 进入mysql目录并创建目录以及文件

Cd /usr/local/mysql

Mkdir data

Mkdir tmp

Mkdir log

Touch ./log/mysqld.log

* 给mysql目录授权

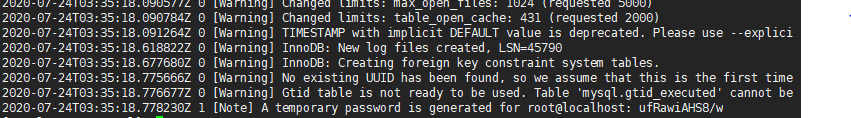
Chmod -R 777 /usr/local/mysql

* 初始化数据库

Cd /usr/local/mysql

bin/mysqld --initialize --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data

初始化密码在/usr/local/mysql/log/mysqld.log中



* 测试启动 ./support-files/mysql.server start
* 添加开机自启动

切换root用户

cp /usr/local/mysql/support-files/mysql.server  /etc/init.d/mysqld

修改   vim /etc/init.d/mysqld

添加路径 在46行

basedir=/usr/local/mysql

datadir=/usr/local/mysql/data

切换至mysql用户启动mysql   service mysqld start

加入开机起动    chkconfig --add mysqld  (root用户)

ln -s /usr/local/mysql/bin/mysql /usr/bin (root用户)

ln -s /usr/local/mysql/tmp/mysql.sock /tmp/mysql.sock  (root用户 不加登录的时候回报 ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock')

* 修改密码，并更改权限

切换到mysql 用户   登录修改密码 mysql -uroot -p 上面初始化时的密码

alter user 'root'@'localhost' identified by 'root';

退出重新登录

Use mysql；

Update user set host = ‘%’ where user =‘root’；

flush privileges;    #刷新权限

* 创建hive用户

create database scm default character set utf8 default collate utf8\_general\_ci;

grant all on scm.\* to 'scm'@'%' identified by 'scm';

create database amon default character set utf8 default collate utf8\_general\_ci;

grant all on amon.\* to 'amon'@'%' identified by 'amon';

create database rman default character set utf8 default collate utf8\_general\_ci;

grant all on rman.\* to 'rman'@'%' identified by 'rman';

create database hue default character set utf8 default collate utf8\_general\_ci;

grant all on hue.\* to 'hue'@'%' identified by 'hue';

create database hive default character set utf8 default collate utf8\_general\_ci;

grant all on hive.\* to 'hive'@'%' identified by 'hive';

create database sentry default character set utf8 default collate utf8\_general\_ci;

grant all on sentry.\* to 'sentry'@'%' identified by 'sentry';

create database oozie default character set utf8 default collate utf8\_general\_ci;

grant all on oozie.\* to 'oozie'@'%' identified by 'oozie';

create database cmf default character set utf8 default collate utf8\_general\_ci;

grant all on cmf.\* to 'cmf'@'%' identified by 'cmf';

create database nav default character set utf8 default collate utf8\_general\_ci;

grant all on nav.\* to 'cmf'@'%' identified by 'nav';

create database cm default character set utf8 default collate utf8\_general\_ci;

grant all on cm.\* to 'cm'@'%' identified by 'cm';

#这是授权root用户在主节点拥有所有数据库的访问权限

grant all privileges on \*.\* to 'root'@'master' identified by 'root' with grant option;

flush privileges;

需要把mysql-connector-java-5.1.46.jar 重命名为mysql-connector-java.jar 放到/usr/share/java  所有的集群服务器都要放

chmod 777 /usr/share/java/mysql-connector-java.jar  (赋权，不然后面会报找不到jdbc包)

## 通过yum 安装cloudera-scm-server

* 拷贝cloudera-repos-6.2.0下所有文件至/var/www/html/cm

cp -r cloudera-repos-6.2.0/\* /var/www/html/cm/

* 拷贝parcel-6.2.0下所有文件至/var/www/html/cdh

cp -r parcel-6.2.0/\* /var/www/html/cdh/

* 执行yum -y install createrepo
* 进入cm和cdh目录执行createrepo .
* 在/etc/yum.repo.d/下添加cm.repo和cdh.repo

vi cm.repo

[cmrepo]

name=cm

baseurl=http://21.156.233.210/cm

gpgcheck=false

enabled=true

vi cdh.repo

[cdhrepo]

name=cdh

baseurl=http://21.156.233.210/cdh

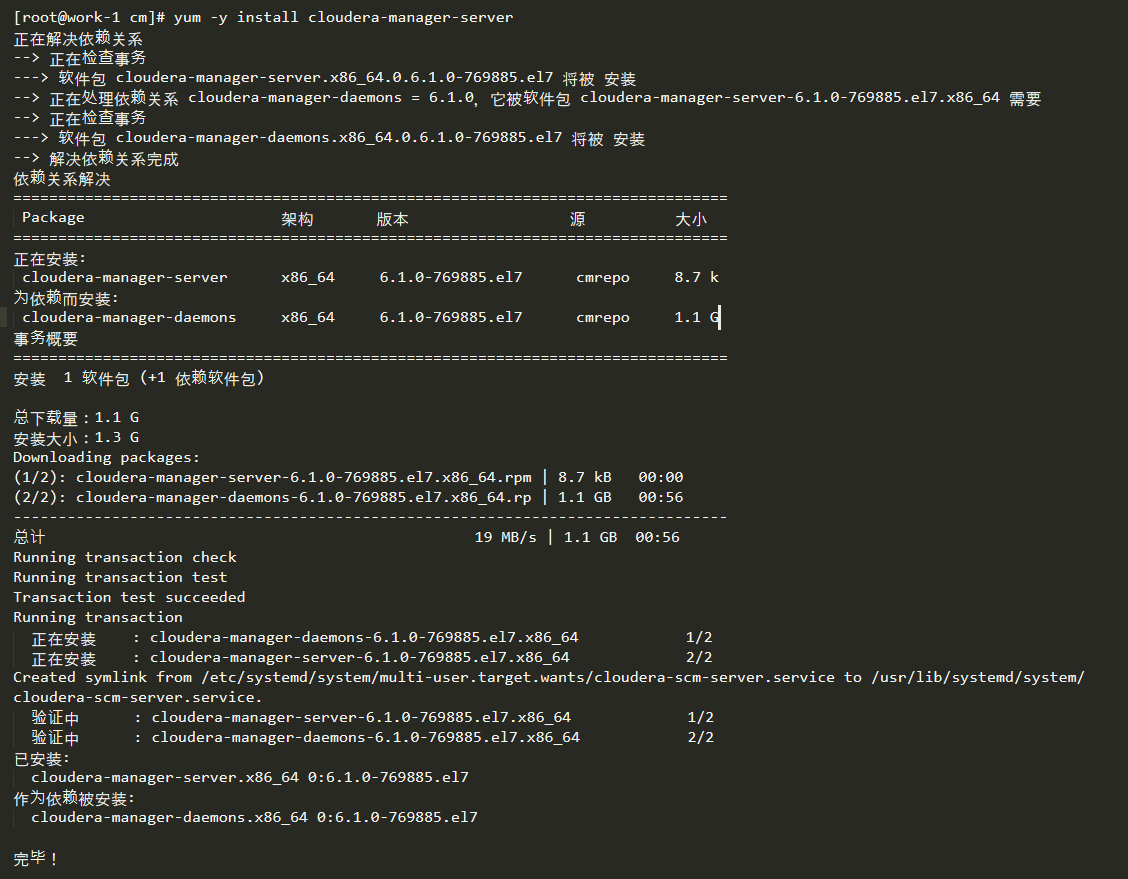
gpgcheck=false

enabled=true

* 拷贝新加文件至其他服务器

scp cm.repo cdh.repo root@cdh-03:$PWD

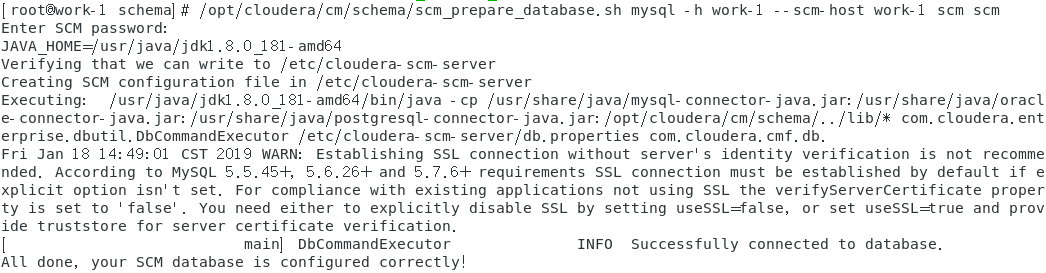
* 所有机器执行yum clran all 和 yum makecache
* 主节点执行yum -y install cloudera-manager-server



vim /etc/cloudera-scm-agent/config.ini

初始化cm的数据库，执行下面命令

/opt/cloudera/cm/schema/scm\_prepare\_database.sh mysql -h work-1 --scm-host work-1 scm scm

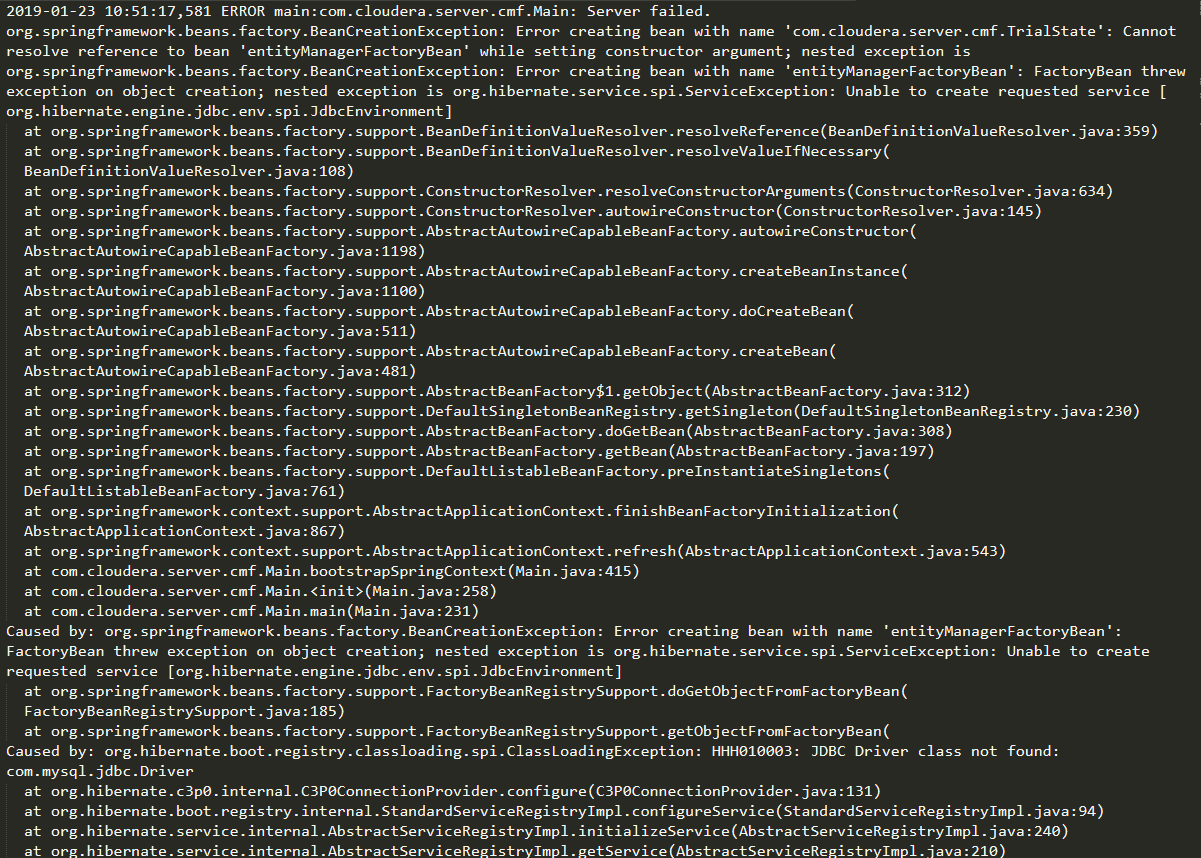


启动cloudera-scm-server

systemctl start cloudera-scm-server

查看7180 端口占用情况  netstat -lnp | grep 7180

查看启动日志 tail -f /var/log/cloudera-scm-server/cloudera-scm-server.log



发现是jdbc没有找到的问题

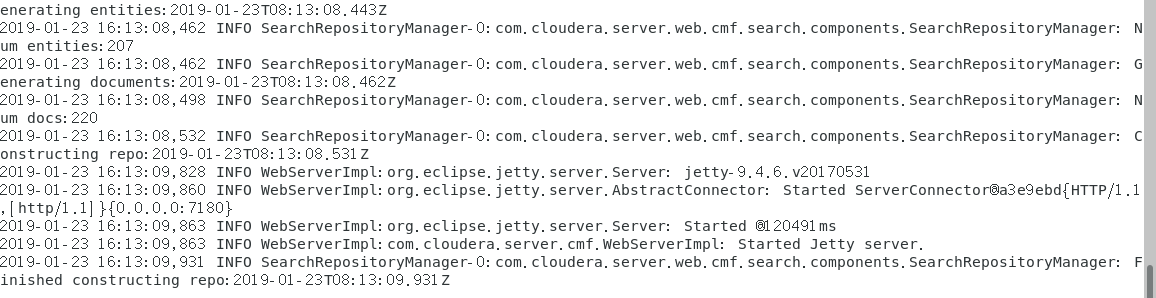
我的jdbc 的 jar包放在  /usr/share/java/mysql-connector-java.jar  但是还是没有找到

1.需要检查 /usr/share/java/mysql-connector-java.jar 是否赋权。

2.export CLASSPATH=./:/usr/share/java/mysql-connector-java.jar  添加环境变量

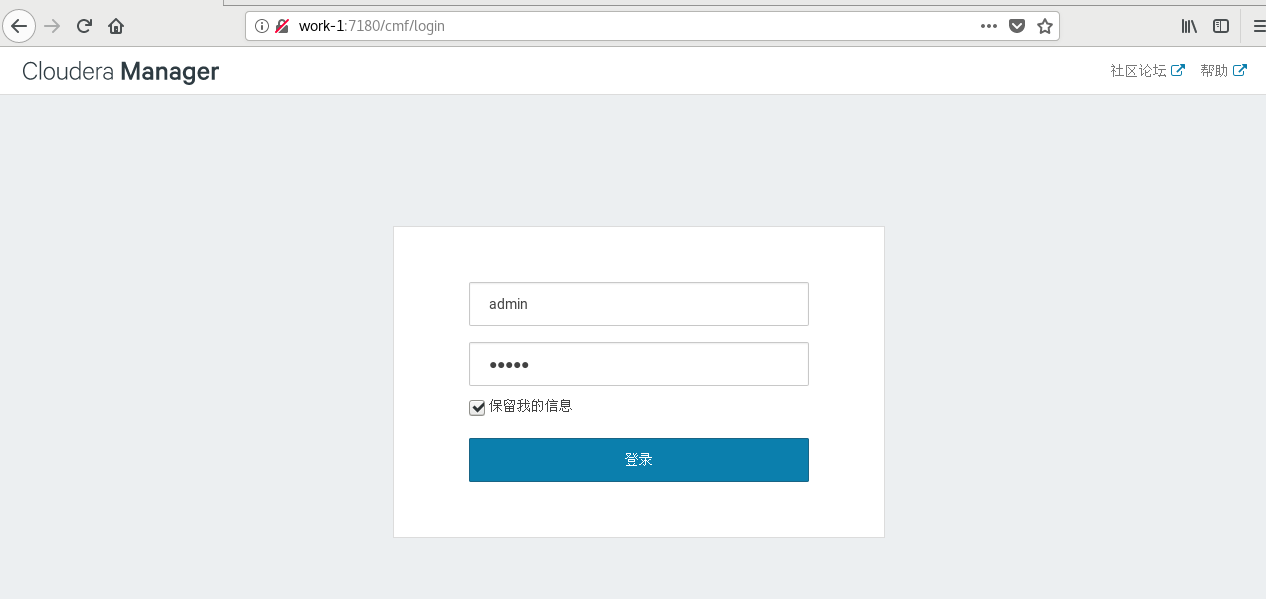
在 vim /etc/cloudera-scm-server/db.properties 下可以查看登录的mysql用户名，密码信息等。

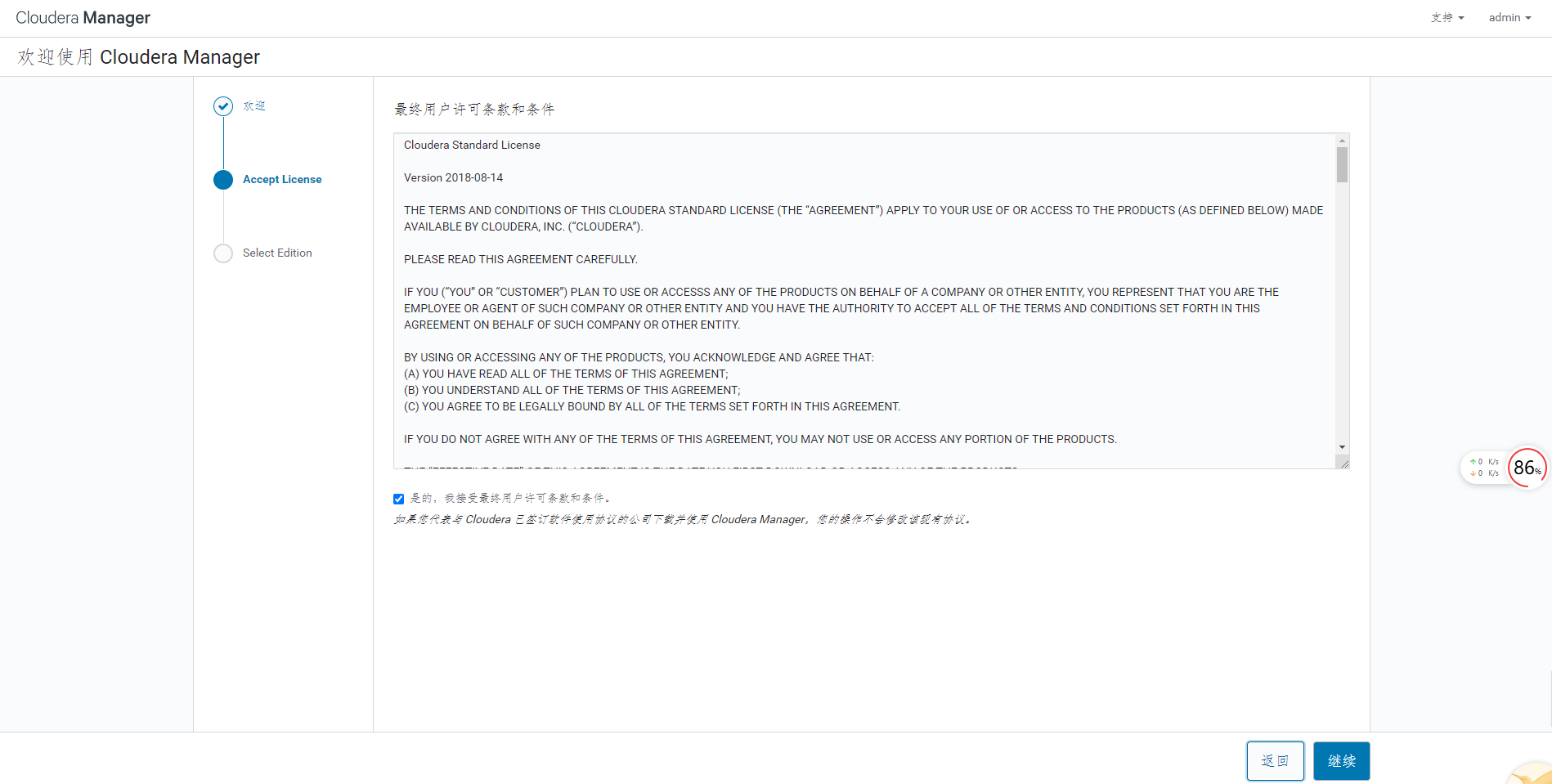
cloudera-scm-server 启动成功之后会有下面信息

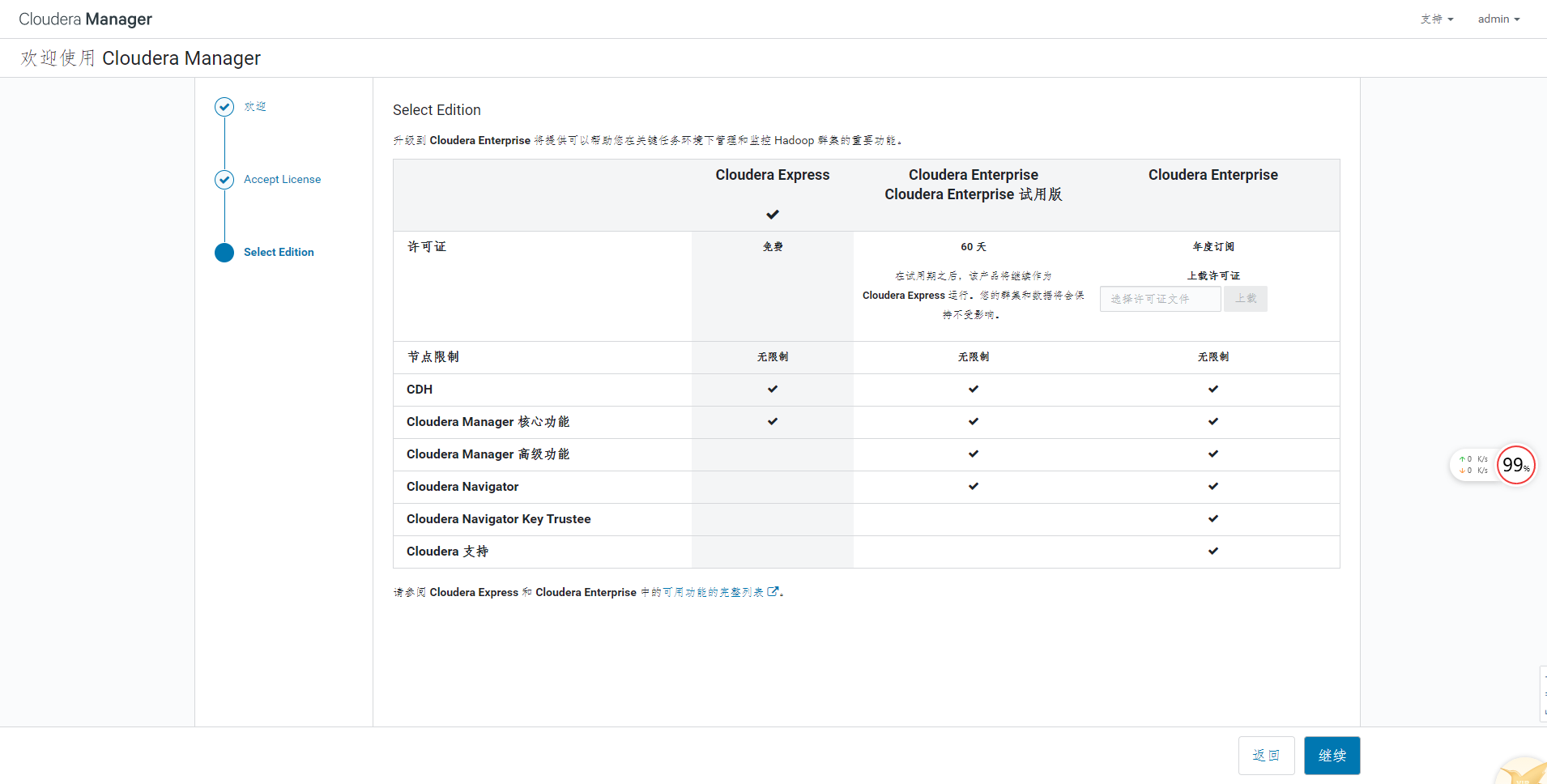
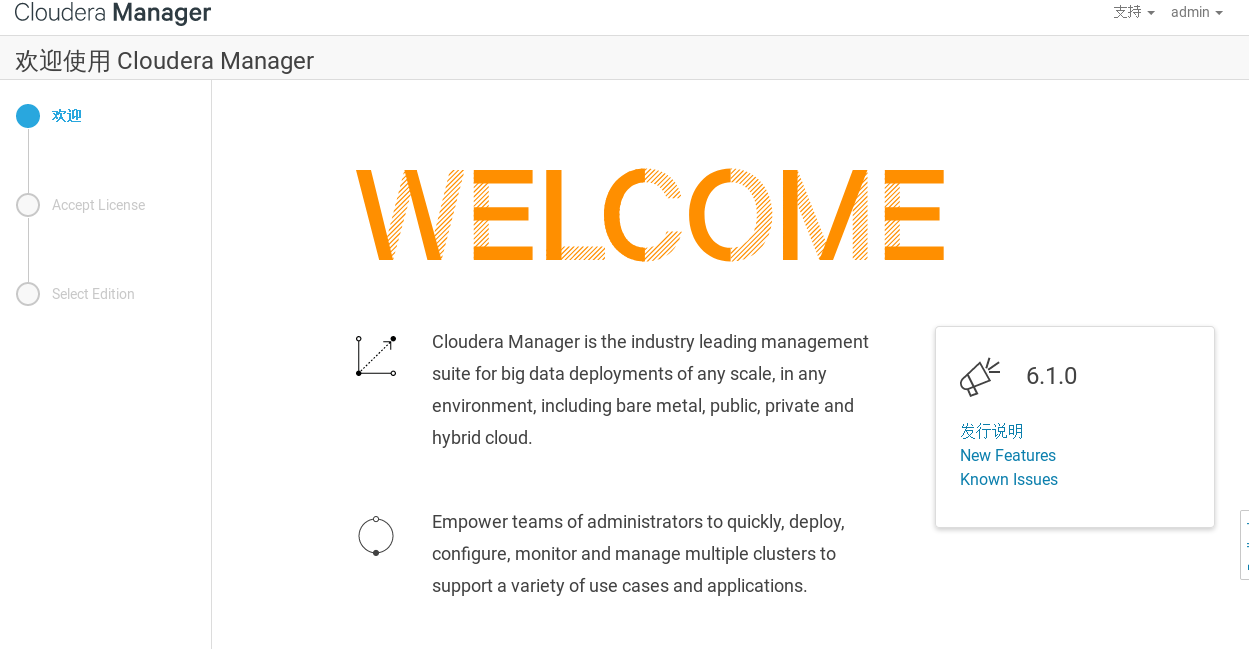


登录 <http://IP:7180/cmf/login>  账号 admin  密码 admin

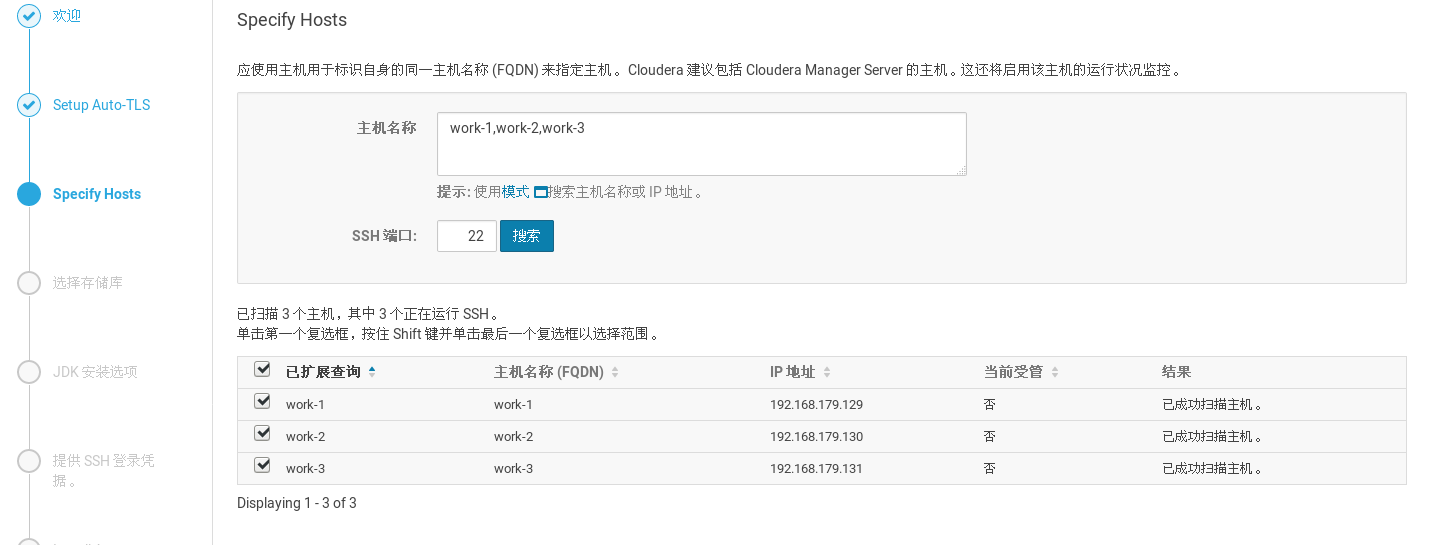
## 登录页面配置



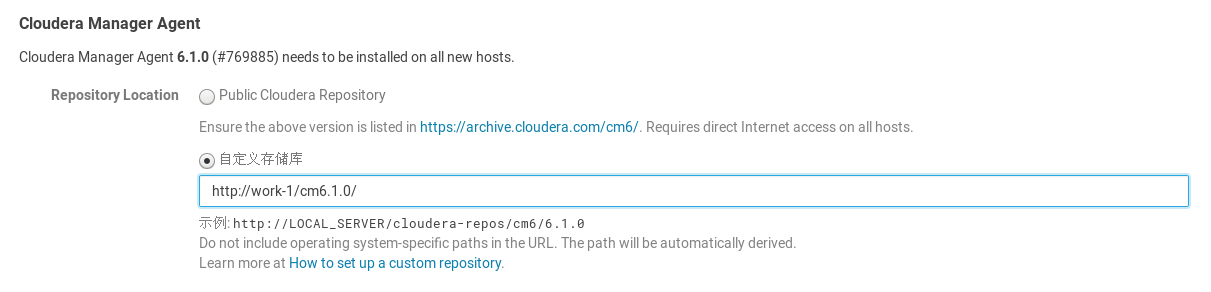




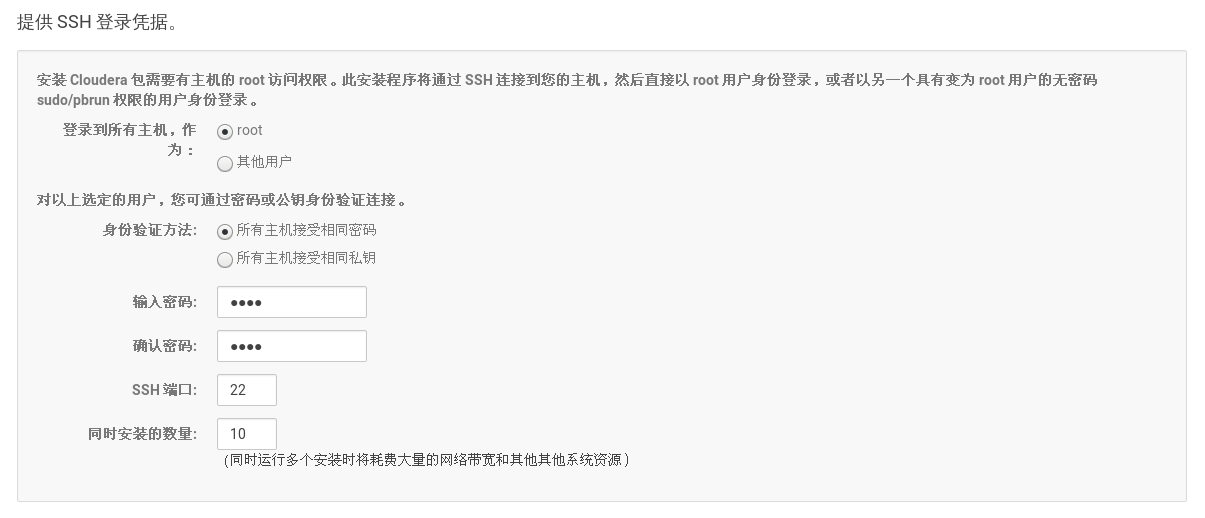
* 点击 继续  继续  继续



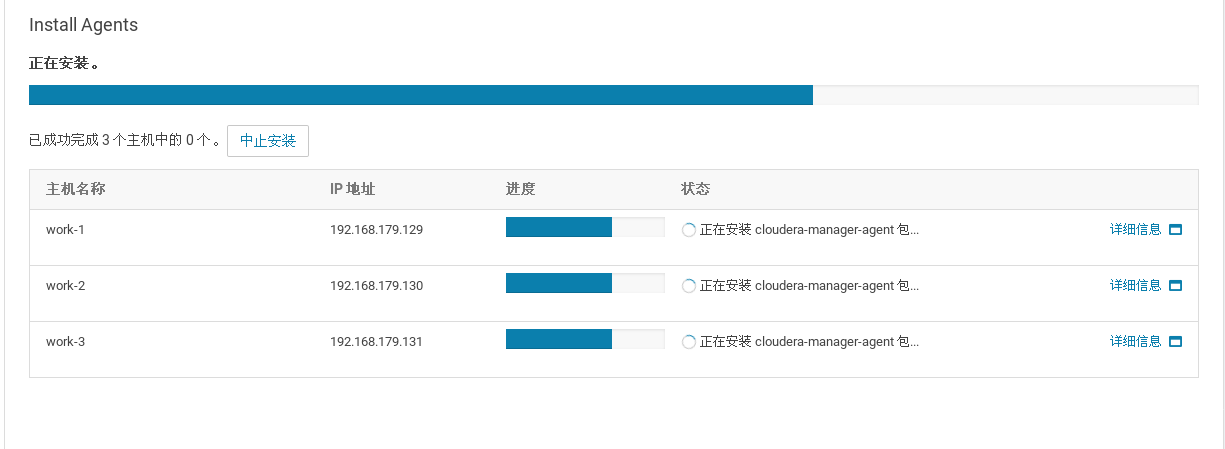
* 需要填写主机网络名，点击搜索，选中需要管理的主机，点击继续



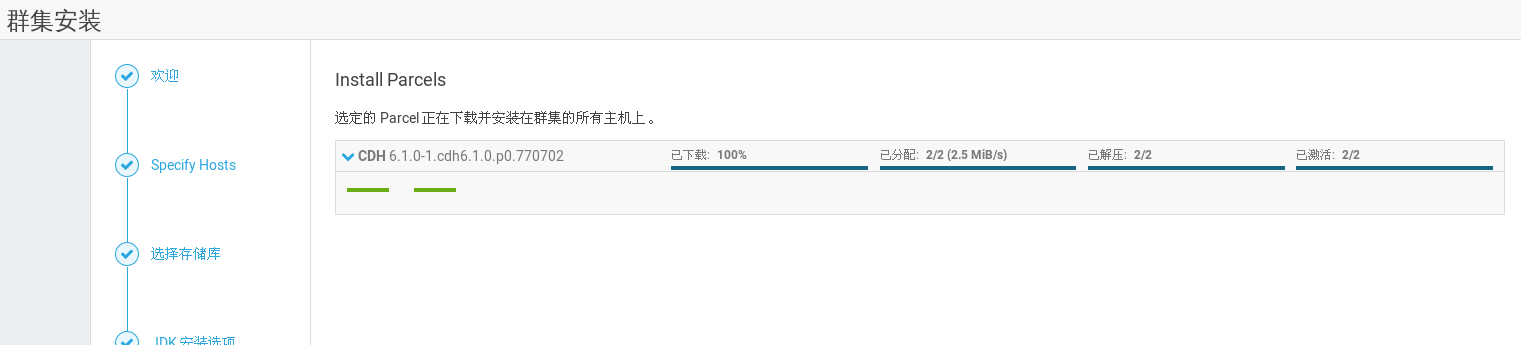
* 选择我们配置的本地的httpd 服务的地址，其他选项默认



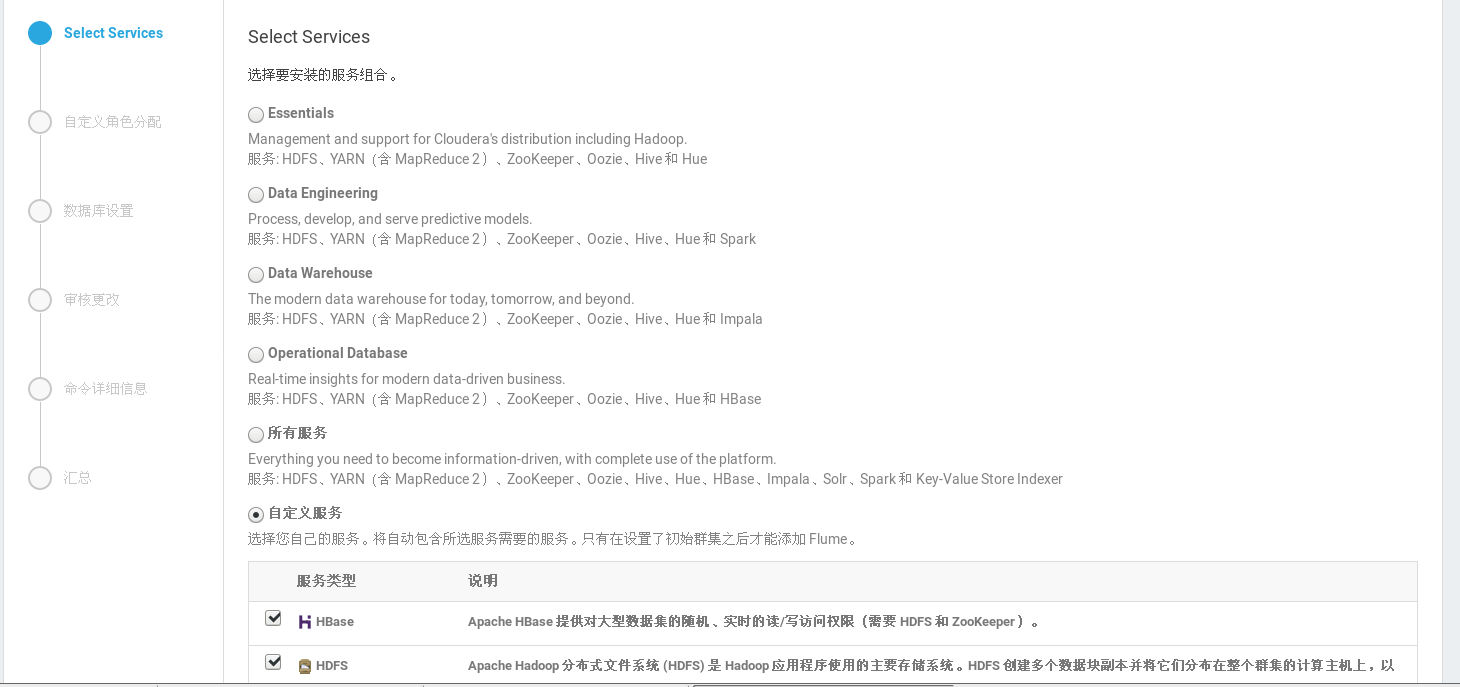
* 以root用户安装，不然权限控制很麻烦



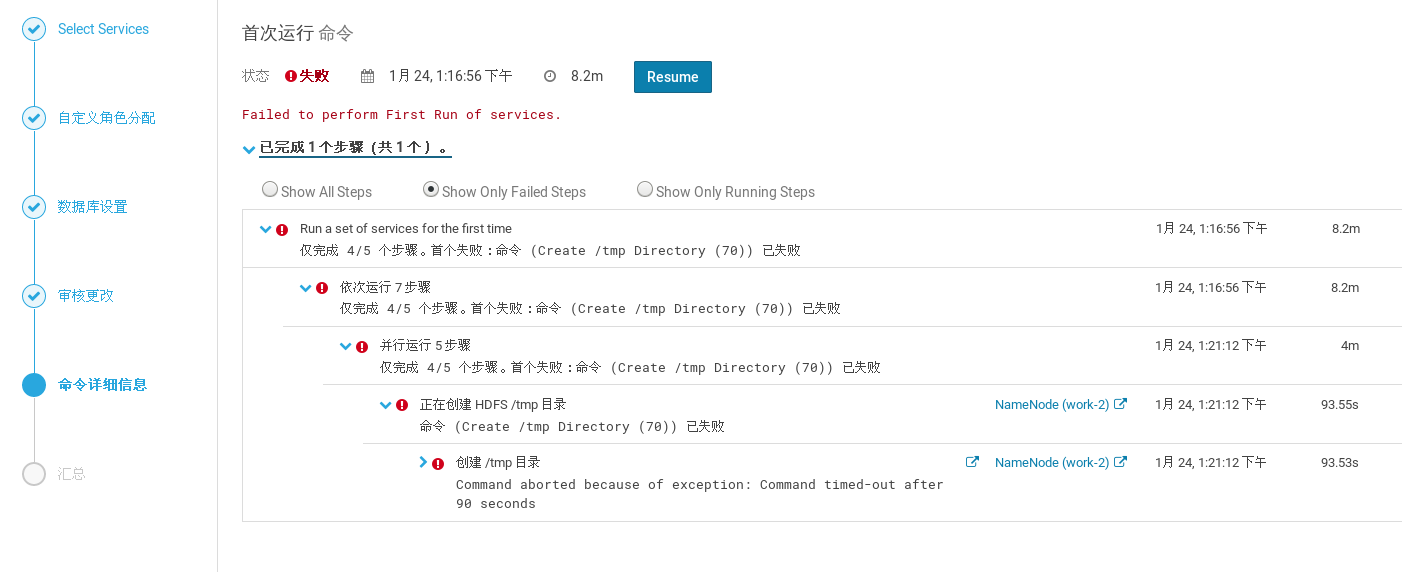




* 点击下一步会出现安装界面，点击详情可以看到安装进度，及安装报错信息。



* 选择安装的版本即可



* 安装过程中出现了错误

Command aborted because of exception: Command timed-out after 90 seconds

在这里遇到的错误是内存不够用导致的，需要多试几次，如果是虚拟机，可以考虑降低角色分配开减少资源开支



* 安装完成

## 系统文件存储结构

###### 与服务相关联目录

/var/log/cloudera-scm-installer : 安装日志目录。

/var/log/\* : 相关日志文件（相关服务的及CM的）。

/usr/share/cmf/ : 程序安装目录。

/usr/lib64/cmf/ : Agent程序代码。

/var/lib/cloudera-scm-server-db/data : 内嵌数据库目录。

/usr/bin/postgres : 内嵌数据库程序。

/etc/cloudera-scm-agent/ : agent的配置目录。

/etc/cloudera-scm-server/ : server的配置目录。

/opt/cloudera/parcels/ : Hadoop相关服务安装目录。

/opt/cloudera/parcel-repo/ : 下载的服务软件包数据，数据格式为parcels。

/opt/cloudera/parcel-cache/ : 下载的服务软件包缓存数据。

/etc/hadoop/\* : 客户端配置文件目录。

###### Hdfs相关配置

配置文件放在目录：/var/run/cloudera-scm-agent/process/之下。

如：/var/run/cloudera-scm-agent/process/193-hdfs-NAMENODE/core-site.xml。这些配置文件是通过Cloudera Manager启动相应服务（如HDFS）时生成的，内容从数据库中获得（即通过[界面](https://www.baidu.com/s?wd=%E7%95%8C%E9%9D%A2&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)配置的参数）。

在CM界面上更改配置是不会立即反映到配置文件中，这些信息会存储于数据库中，等下次重启服务时才会生成配置文件。且每次启动时都会产生新的配置文件。

CM Server主要数据库为scm基中放置配置的数据表为configs。里面包含了服务的配置信息，每一次配置的更改会把当前页面的所有配置内容添加到数据库中，以此保存配置修改历史。

SCM数据库被配置成只能从localhost访问，如果需要从外部连接此数据库，修改vim /var/lib/cloudera-scm-server-db/data/pg\_hba.conf文件,之后重启数据库。运行数据库的用户为cloudera-scm。

###### 集群相关配置信息（JSON格式）

直接查询scm数据库的configs数据表的内容。

访问REST API： http://hostname:7180/api/v4/cm/deployment，返回JSON格式部署配置信息

###### 配置生成方式

CM为每个服务进程生成独立的配置目录（文件）。所有配置统一在服务端查询数据库生成（因为scm数据库只能在localhost下访问）生成配置文件，再由agent通过网络下载包含配置文件的zip包到本地解压到指定的目录。

###### 配置修改

CM对于需要修改的配置预先定义，对于没有预先定义的配置,则通过在高级配置项中使用xml配置片段的方式进行配置。而对于/etc/hadoop/下的配置文件是客户端的配置，可以在CM通过部署客户端生成客户端配置。

      这就需要部署客户端配置（CM界面有），它会把配置下载到/etc/hadoop/conf/下（实际是通过update-alternatives命令实现），如果当前机器有HDFS服务，则会下载HDFS相关配置文件到/etc/hadoop/conf/（如core-site.xml和hdfs-site.xml），如果是mapred服务则会下载mapred-site.xml等相关文件。如果希望某台[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)不提供hadoop服务也可以通过hadoop命令管理集群，则需要为它添加"GateWay"角色，然后下发客户端配置。

###### 数据库

Cloudera manager主要的数据库为scm,存储Cloudera manager运行所需要的信息：配置，主机，用户等。

###### CM结构

* CM分为Server与Agent两部分及数据库（自带更改过的嵌入Postgresql）。它主要做三件事件：
* 管理监控集群主机。
* 统一管理配置。
* 管理维护Hadoop平台系统。
* 实现采用C/S结构，Agent为客户端负责执行服务端发来的命令，执行方式一般为使用python调用相应的服务shell脚本。Server端为Java REST服务，提供REST API，Web管理端通过REST API调用Server端功能，Web界面使用富客户端技术（Knockout）。
* Server端主体使用Java实现。
* Agent端主体使用Python, 服务的启动通过调用相应的shell脚本进行启动，如果启动失败会重复4次调用启动脚本。
* Agent与Server保持心跳，使用Thrift RPC框架。

###### 升级

* 在CM中可以通过界面向导升级相关服务。升级过程为三步：
* 下载服务软件包。
* 把所下载的服务软件包分发到集群中受管的机器上。
* 安装服务软件包，使用软链接的方式把服务程序目录链接到新安装的软件包目上

###### 卸载

* 在cm安装的服务器执行下面命令：
* sudo /usr/share/cmf/uninstall-cloudera-manager.sh
* 然后删除/var/lib/cloudera-scm-server-db/目录，不然下次安装可能不成功。

###### 开启数据库远程访问

CM内嵌数据库被配置成只能从localhost访问，如果需要从外部查看数据，数据修改vim /var/lib/cloudera-scm-server-db/data/pg\_hba.conf文件,之后重启数据库。运行数据库的用户为cloudera-scm。

## 硬件选择

搭建大数据环境的首要任务就是选择硬件，但是硬件的选择也不是一拍脑袋想出来的，需要结合数据处理特点，数据量，负载，性能，价格等多方面因素进行平衡，并且还需要进行测试验证其有效性，稳定性。

选择硬件之前我们先要了解以下内容：

* 分析业务数据处理的负载，预判处理可能遇到的硬件瓶颈，针对性加大相对应的硬件投资。
* 了解团队现有资源，比如服务器类型中需要了解是云服务器还是物理机。需要了解可以提供的最大资源。
* 在技术选型时，需要确定使用那些大数据组件，并且了解每个组件所需要的资源情况，综合计算所使用的资源总量。

###### 工作负载

大数据处理问题的瓶颈大致分为两种，从磁盘或者网络读取数据时遇到瓶颈（称为IO受限的应用），处理数据时遇到瓶颈（CPU受限）。

IO受限的例子如：

* 索引
* 分组
* 数据导入导出
* 数据移动和转换

CPU受限的工作负载的例子：

* 聚类/分类
* 复杂文本挖掘
* 自然语言处理
* 特征提取

在选择硬件之前，需要冷静分析所需要任务，找到硬件选择的最优解，在实际使用中，还会出现一些预料之外的，有些任务原本是IO的瓶颈，结果由于选择的算法等不同，处理逻辑不同，引起cpu受限，Cloudera Manager有仪表盘，可以比较直观的观察到每台机器的运行情况。

###### CDH硬件推荐

CDH的大多数节点是需要两个角色的任务，就是数据存储和数据计算。

主流节点安装推荐：

* 需要挂载单块硬盘为1T以上为宜
* 物理CPU为2 路8核，2路10核，2路12核，2路14核，2路16核的CPU, 主频至少2-2.5GHz
* 64-512GB内存
* 万兆网卡

###### CPU、内存、磁盘IO、YARN监控

1.CPU

CPU是计算机计算核心，在大数据处理中，有些高并发处理，科学计算，图像分析等都是比较消耗CPU的。大数据中文件的拷贝主要是直接存储器存取（DMA）来完成，CPU参与其中很少一部分工作。当我们大数据程序运行时，我们监控CPU和针对CPU进行调优是很有必要的。

CPU的管理也是内核的进程调度，来管理和分配CPU资源，决定那个进程的队列，那个进程执行，那个进程等待。在linux中，资源分配有不同的优先级：硬件中断，内核进程（或线程），用户进程（或线程）由高到低来分级，优先处理的肯定是高优先级高的。每一个cpu运维这个一个线程队列，线程有6种状态：初始、运行、阻塞、等待、超时等待、终止，其中原理这里不做详细说明，我们可以通过JVM自带工具及linux提供的CPU监控工具来监控分析。

CPU使用到什么情况才算合理呢，这里提供一个参考：

* 如果CPU利用率为100%，则应该达到这样一个平衡：用户时间（65%-70%），系统时间（30%-35%），中断时间（0%-5%）。如果CPU利用率达到平衡，有大量上下文切换是合理的。注意CPU长期处于100%是不健康的状态。
* 每个CPU的逻辑核其实相当于线程数，根据逻辑核个数判断自己可以调配的线程数，避免出现资源争夺的情况。

这里列举vmstat和mpstat这两个工具来分析CPU使用情况

vmstat用来显示Linux系统虚拟内存状态，也可以报告关于进程、内存、I/O等系统整体运行状态。

Procs（进程）

|  |  |
| --- | --- |
| r: | 运行队列中进程数量，这个值也可以判断是否需要增加CPU。（长期大于1） |
| b | 等待IO的进程数量。 |

Memory（内存）

|  |  |
| --- | --- |
| swpd | 使用虚拟内存大小，如果swpd的值不为0，但是SI，SO的值长期为0，这种情况不会影响系统性能。 |
| free | 空闲物理内存大小。 |
| buff | 用作缓冲的内存大小。 |
| cache | 用作缓存的内存大小，如果cache的值大的时候，说明cache处的文件数多，如果频繁访问到的文件都能被cache处，那么磁盘的读IO bi会非常小。 |

Swap

|  |  |
| --- | --- |
| si | 每秒从交换区写到内存的大小，由磁盘调入内存。 |
| so | 每秒写入交换区的内存大小，由内存调入磁盘。 |

IO

|  |  |
| --- | --- |
| bi | 每秒读取的块数 |
| bo | 每秒写入的块数 |

注意：随机磁盘读写的时候，这2个值越大（如超出1024k)，能看到CPU在IO等待的值也会越大。

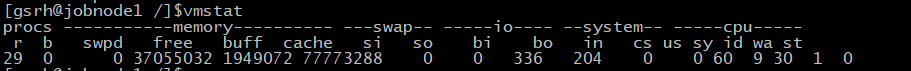
system（系统）

|  |  |
| --- | --- |
| in | 每秒中断数，包括时钟中断。 |
| cs | 每秒上下文切换数。 |

注意：上面2个值越大，会看到由内核消耗的CPU时间会越大。

CPU（以百分比表示）

|  |  |
| --- | --- |
| us | 用户进程执行时间百分比(user time) us的值比较高时，说明用户进程消耗的CPU时间多，但是如果长期超50%的使用，那么我们就该考虑优化程序算法或者进行加速。 |
| sy: | 内核系统进程执行时间百分比(system time) sy的值高时，说明系统内核消耗的CPU资源多，这并不是良性表现，我们应该检查原因。 |
| wa | IO等待时间百分比 wa的值高时，说明IO等待比较严重，这可能由于磁盘大量作随机访问造成，也有可能磁盘出现瓶颈（块操作）。 |
| id | 空闲时间百分比 |



我们执行了一个命令，可以个根据上面结果进行分析，用户时间占比60%，说明CPU使用率过高，我们可以考虑调整算法或增加计算资源。在大数据环境，我们不推荐使用虚拟内存，所以虚拟内存被干掉了，此处显示为0。

mpstat 可以显示多个CPU的使用情况。

mpstat -P ALL 所有CPU（逻辑核）的使用情况显示。

CPU : 处理器编号。关键字all表示统计信息计算为所有处理器之间的平均值。

     ％usr：显示在用户级（应用程序）执行时发生的CPU利用率百分比。

     ％nice：  显示以优先级较高的用户级别执行时发生的CPU利用率百分比。

     ％sys： 显示在系统级（内核）执行时发生的CPU利用率百分比。请注意，这不包括维护硬件和软件的时间中断。

     ％Iowait ： 显示系统具有未完成磁盘I / O请求的CPU或CPU空闲的时间百分比。

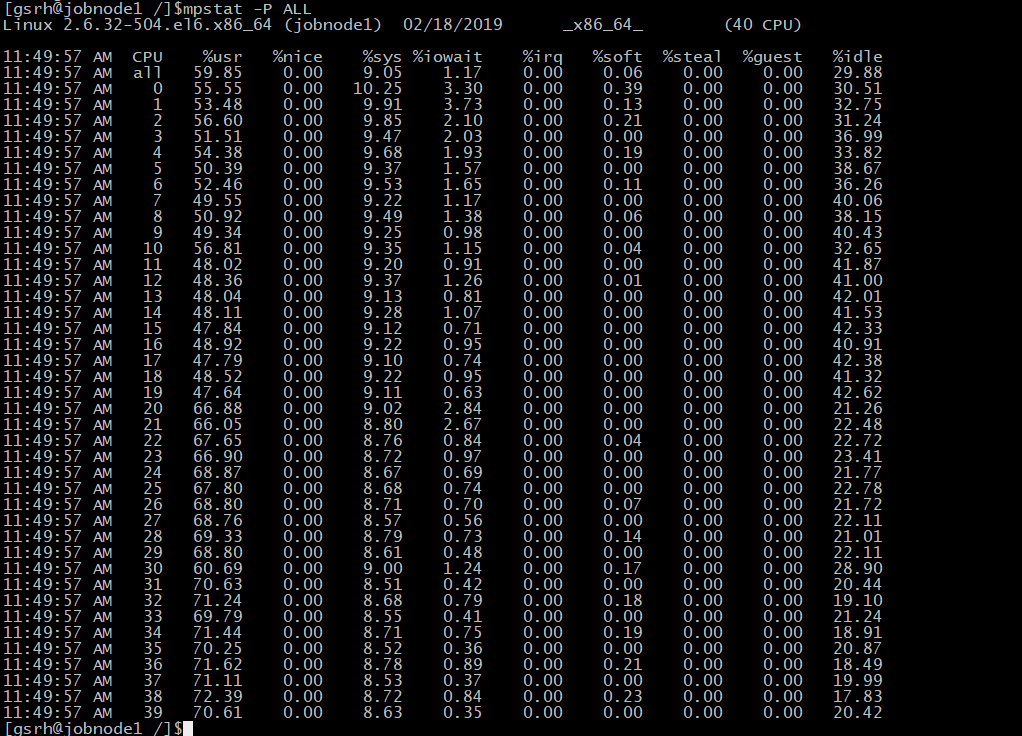
     ％irq ： 显示CPU或CPU用于服务硬件中断的时间百分比。

     %soft ： 显示CPU或CPU用于服务软件中断的时间百分比。

     %steal：  显示在虚拟机管理程序服务另一个虚拟处理器时虚拟CPU或CPU无意中等待的时间百分比。

     %guest ：显示CPU或CPU运行虚拟处理器所花费的时间百分比。

     %idle： 显示CPU或CPU空闲且系统没有未完成的磁盘I / O请求的时间百分比。



1. 内存

内存只要物理内存和虚拟内存，物理内存我们都明白，虚拟内存就是把内存扩展到磁盘上，以助于可以运行更大的软件，但是缺点也很明显，磁盘的读写速度要比内存慢了很多个数量级，并且RAM 和 SWAP 之间的交换增加了系统的负担，在大数据处理中会出现处理时间不一致导致系统延时，更甚者导致整个大数据环境崩溃，所以大数据框架搭建后，建议干掉虚拟内存，提升处理的速度和稳定性。

JVM本身就需要给其分配内存的10%及以上来管理虚拟机。通常大数据的集群任务可能会各自在自己的JVM中运行，一个JVM中的线程是共享其空间的，但是单个线程还有自己的独立空间，所以在大数据任务中线程个数分配和空间分配显得格外重要。大数据任务内存通常通过YARN来分配，YARN可支配资源就是大数据任务使用的资源。

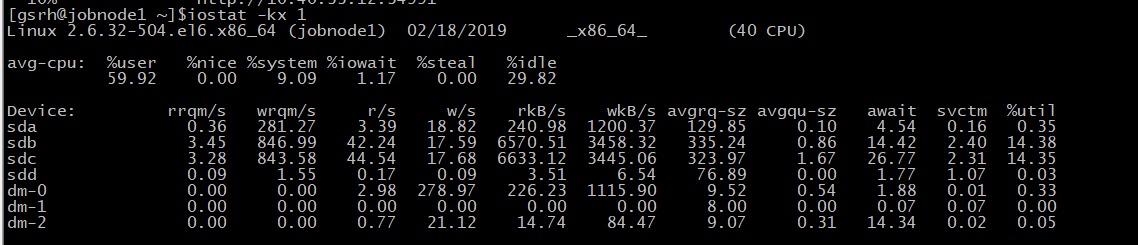
Linux查看内存使用情况

1. cat /proc/meminfo 这个是内存使用最简单的查看方法。
2. 用vmstat 也可以查看内存。
3. free -h

其实还有许多工具可以查看内存使用情况，在大数据任务中可以监控到，如果内存不够可能会报内存溢出等错误。选择内存时要根据自己要处理的数据量去判断，建议内存总量大于需要处理的数据量这种搭配模式。

1. IO

磁盘现在是计算机中最慢的子系统，并且是很容易出现瓶颈的地方，因为磁盘离CPU距离最远而且CPU访问磁盘要涉及到机械操作，比如转轴、寻轨等。IO 可分为顺序IO和随机IO两种，性能监测前需要弄清楚系统偏向顺序 IO 的应用还是随机IO应用。顺序IO是指同时顺序请求大量数据，比如数据库执行大量的查询、流媒体服务等，顺序IO可以同时很快的移动大量数据。可以这样来评估 IOPS 的性能，用每秒读写IO字节数除以每秒读写IOPS数，rkB/s 除以 r/s，wkB/s 除以 w/s，顺序IO更应该重视每次IO的吞吐能力。随机IO是指随机请求数据，其IO速度不依赖于数据的大小和排列，依赖于磁盘的每秒能IO的次数，比如Web服务、Mail 服务等每次请求的数据都很小，随机IO每秒同时会有更多的请求数产生，所以磁盘的每秒能IO多少次是关键。



## 角色及节点分配

CDH 软件体系结构包括以下模块：系统部署和管理，数据存储，资源管理， 处理引擎，安全、数据管理，工具库以及访问接口。一些关键组件的角色信息如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **模块** | **组件** | **管理角色** | **工作角色** |
| 系统部署和管理 | Cloudera Manager | Cloudera Manager Server | Cloudera Manager Agent |
| Host Monitor |  |
| Service Monitor |  |
| Reports Manager |  |
| Alert Publisher |  |
| Event Server |  |
| Cloudera Director |  |  |
|  | HDFS | NameNode | DataNode |
| Secondary NameNode |  |
| JournalNode |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 数据存储 |  | FailoberController |  |
| HBase | HBase Master | RegionServer |
| 资源管理 | YARN | ResourceManager | NodeManager |
| Job HistoryServer |  |

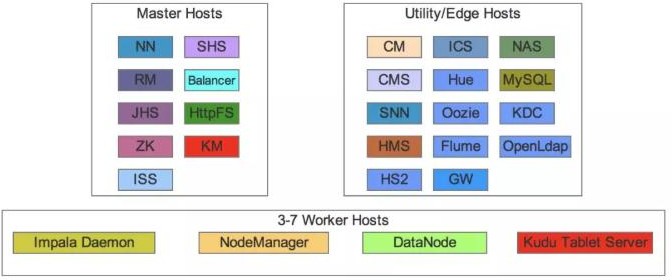
|  |  |  |  |
| --- | --- | --- | --- |
| 处理引擎 | Spark | History Server |  |
| Impala | Impala Catalog Server | Impala Daemon |
| Impala StateStore |  |
| Search |  | Solr Server |
| 安全、数据管理 | Sentry | Sentry Server |  |
| Cloudera Navigator | Navigator KeyTrustee |  |
| Navigator Metadata  Server |  |
| Navigator Audit Server |  |
| 工具库 | Hive | Hive Metastore |  |
| Hive Server2 |  |

本节主要介绍由 Cloudera Manager 管理的 CDH 集群的角色划分。实际部署可能还需要考虑工作负载的类型和数量，真实要部署的哪些服务，硬件资源，配置，以及其他因素。当使用 Cloudera Manager 的安装向导来安装 CDH 时，CM 会根据主机的可用资源，自动的分配角色到各台主机，边缘节点除外。可以在向导中使用”自定义角色分配 - Customize Role Assignments”来更改这些默认划分，当然也可以以后再使用 Cloudera Manager 来增加或修改角色分配。

在介绍角色划分时，我们首先来看看有哪几种主要的角色：

* + - 1. 管理节点（Master Hosts）：主要用于运行 Hadoop 的管理进程，比如HDFS 的 NameNode，YARN 的 ResourceManager。
      2. 工具节点（Utility Hosts）:主要用于运行非管理进程的其他进程，比如Cloudera Manager 和 Hive Metastore.
      3. 边缘节点（Edge Hosts）：用于集群中启动作业的客户端机器，边缘节点的数量取决于工作负载的类型和数量。
      4. 工作节点（Worker Hosts）：主要用于运行 DataNode 以及其他分布式进程，比如 ImpalaD。

###### 测试/开发集群（小于 10 台）

搭建测试/开发集群，建议至少 5 台机器，没有高可用。一个管理节点主要用于安装 NameNode 和 ResourceManager，工具节点和边缘节点复用一个，主要用于安装 Cloudera Manager 等，剩余 3-7 台工作节点。

###### 小规模集群（10-20 台）

搭建小规模集群一般是为了支撑专有业务，受限于集群的存储和处理能力， 不太适合用于多业务的环境。这可以部署成一个 HBase 的集群；也可以是一个分析集群，包含 YARN，Impala。在小规模集群中，为了最大化利用集群的存储和处理能力，节点的复用程度往往也比较高。下图是一个典型的小规模集群部署方式：

小规模的生产集群，高可用性仍然是必须的配置，集群中两台主节点配置高可用的组件角色：

|  |
| --- |
| HDFS NameNode |
| HDFS  FailoverController |
| HDFS JournalNode |
| Zookeeper |
| YARN  ResourceManager |
| HBase Master |
| Impala StateStore |

对于那些需要两个以上节点来支持 HA 功能的，集群中分配有一个工具节点可以承载这些角色，并同时可以部署一些其他工具角色，这些工具角色本身消耗不了多少资源：

|  |
| --- |
| HDFS JournalNode |
| Zookeeper |
| HBase Master |
| Cloudera Manager Server |
| Cloudera Management Service |
| History Server |
| Job History Server |
| Hive Metastore |
| HiveServer2 |
| Impala Catalog Server |
| Hue Server |
| Oozie Server |
| Gateway |

其余节点可以部署为纯工作节点，包含：

|  |
| --- |
| HDFS DataNode |
| YARN NodeManager |
| Impala Daemon |
| HBaseRegionServer |

###### 中等规模集群（20-200 台）

一个中等规模的集群，集群的节点数一般在 20 到 200 左右，通常的数据存储可以规划到几百 TB，适用于一个中型企业的数据平台，或者大型企业的业务部门数据平台。节点的复用程度可以降低，可以按照管理节点、主节点、工具节点和工作节点来划分。



管理节点上就安装 ClouderaManager、Cloudera Management Service。主节点上安装有个 CDH 服务的管理节点以及 HA 的组件，可以如下方式部署：

|  |  |  |  |
| --- | --- | --- | --- |
| 服务 | 主节点 1 | 主节点 2 | 主节点 3 |
| HDFS | NameNode | NameNode |  |
| FailoverController | FailoverController |  |
| JournalNode | JournalNode | JournalNode |
| YARN | Resource Manager | Resource Manager |  |
|  |  | Job History Server |
| Zookeeper | Zookeeper Server | Zookeeper Server | Zookeeper Server |
| HBase | HBase Master | HBase Master | HBase Master |
| Impala | Impala StateStore | Impala Category |  |
|  | Server |  |
| HDFS | NameNode | NameNode |  |
| FailoverController | FailoverController |  |
| JournalNode | JournalNode | JournalNode |
| YARN | Resource Manager | Resource Manager |  |
| Zookeeper |  |  | Job History Server |
| Zookeeper Server | Zookeeper Server | Zookeeper Server |

|  |  |  |  |
| --- | --- | --- | --- |
| HBase | HBase Master | HBase Master | HBase Master |
| Impala | Impala StateStore | Impala Category |  |
|  | Server |  |
| Hive |  |  | Hive Metastore |
| Spark |  |  | History Server |

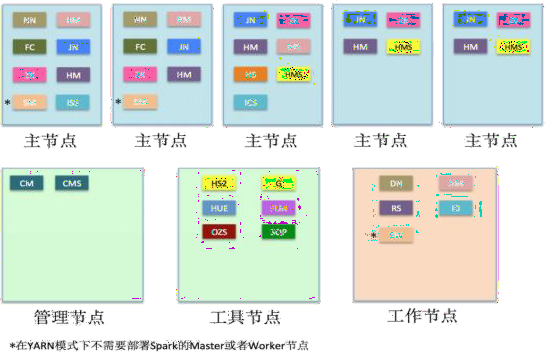
工具节点可以部署以下一些角色：

|  |
| --- |
| HiveServer2 |
| Hue Server |
| Oozie Server |
| Flume Agent |
| Sqoop Client |
| Gateway |

工作节点的部署和小规模类似：

|  |
| --- |
| HDFS DataNode |
| YARN NodeManager |
| Impala Daemon |
| HBaseRegionServer |

###### 大规模集群（200 台以上）

大规模集群的数量一般会在 200 以上，存储容量可以是大几百的 TB 甚至是 PB 级别，适用于大型企业搭建全公司的数据平台。和中等规模的集群相比， 部署的方案相差不大，主要是一些主节点可用性的增强。

HDFS JournalNode 由 3 个增加到 5 个，Zookeeper Server 和 HBase Master 也由 3 个增加到 5 个，Hive Metastore 的数量有 1 个增加到 3 个。

###### 节点分配

本节介绍 Hadoop 集群典型的节点分配，如下表所示：

|  |  |  |
| --- | --- | --- |
| 角色 | 描述 | 节点数目 |
| HDFS  NameNode | 分布式文件系统用以存储文件系统以及数据块的元数据 | 1 个独立节点 |
| HDFS Secondary NameNode | NameNode 影子节点 | 小规模集群可以和NameNode 共享节点，大规模集群用独立节点 |
| HDFS DataNode | Hdfs 数据存储 | 多个独立节点 |
| MapReduce JobTracker | MapReduce 调度程序 | 小规模集群可以和NameNode 共享节点，大规模集群用独立节点 |
| MapReduce TaskTracker | MapReduce 实际计算节点 | 与 DataNode 运行在相同的节点之  上 |
| Hive | Hive 元数据以及驱动程序 | 独立配置的话可以与NameNode 共  享节点，或者将元数据存放在客户端 |
| Zookeeper | 用以提供集群高可用性的锁服务 | 3 个或 3 个以上的奇数的独立节点  （小规模集群可以和其他角色共享节点） |
| HBase HMaster | HBase 用以调度RegionServer  的主模块 | 与其他角色共享节点的多个节点 |
| HBase RegionServer | HBase 中用以管理数据的模块 | 一般与 DataNode 运行在相同的节点之上 |
| Management  Node | 可能的集群监控节点 | 一般为一个独立的节点，如果小规  模集群的话可以与其他角色共享。 |

# Datax-web安装

## 安装前准备

* 安装机器必须有python2.7的环境
* 准备datax压缩包

http://datax-opensource.oss-cn-hangzhou.aliyuncs.com/datax.tar.gz

* 准备datax-web压缩包

<https://github.com/WeiYe-Jing/datax-web/blob/master/doc/datax-web/datax-web-deploy.md>

* Jdk环境
* Mysql环境

## 安装服务

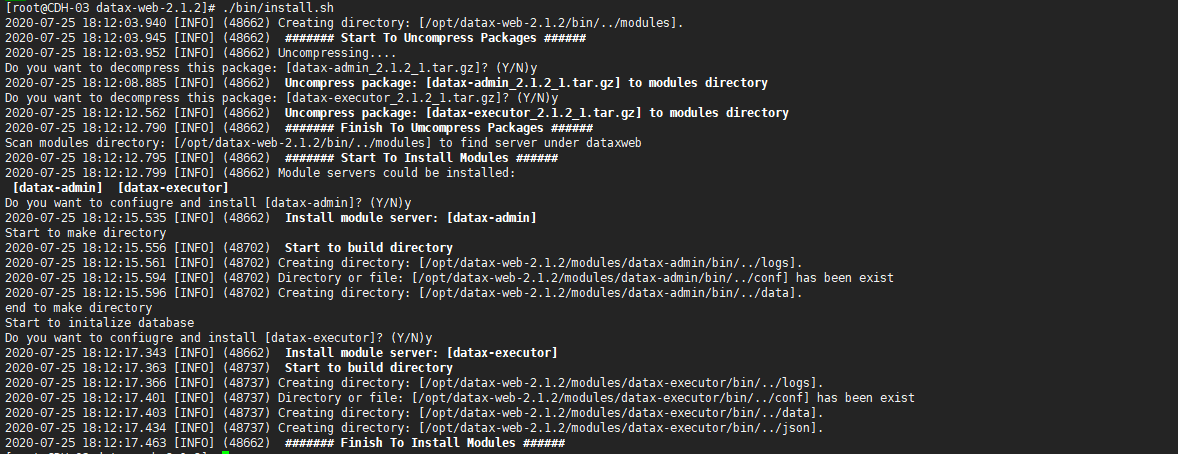
* 拷贝datax和datax-web到指定机器
* 解压两个压缩包到指定路径

tar -zxvf datax-web-2.1.2.tar.gz -C /opt/

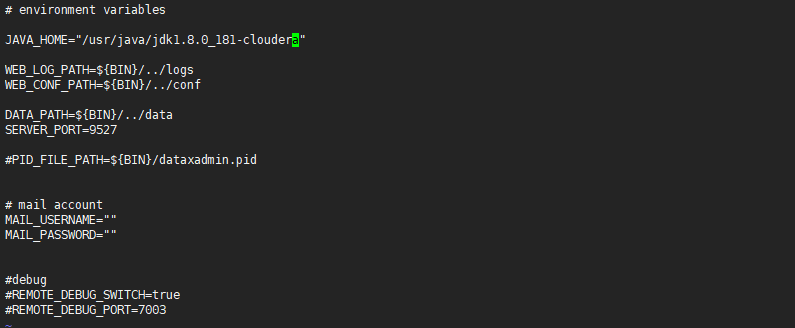
tar -zxvf datax.tar.gz -C /opt/

* 在mysql新建数据库datax-web库并导入data-web中bin\db中的sql文件
* 进入datax-web目录执行：

./bin/install.sh



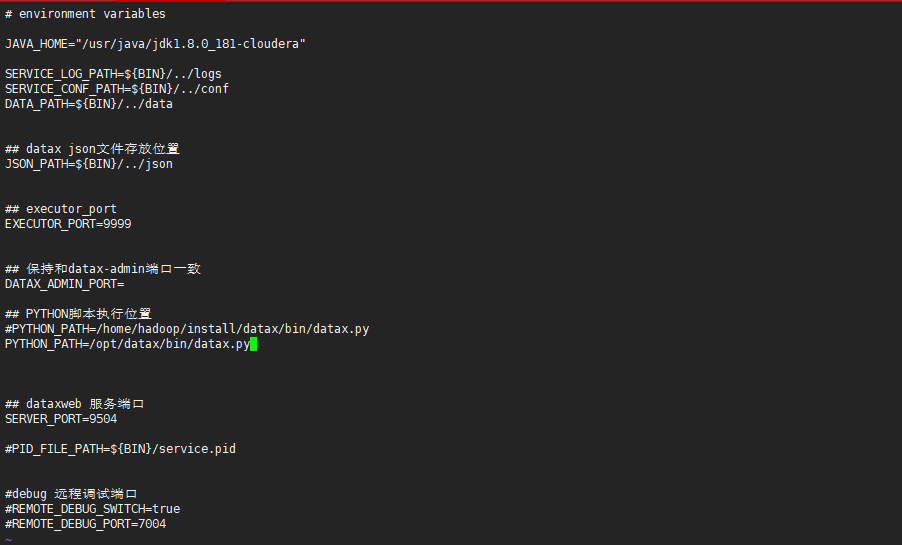
* 进入./modules/datax-admin/bin 修改env.properties 文件中的java\_home



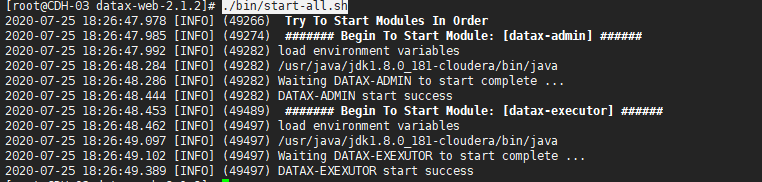
* 进入./modules/datax-admin/conf 修改bootstrap.properties 文件中的数据库信息



* 进入./modules/datax-executor/bin 修改env.properties 文件中的java\_home和PYTHON\_PATH

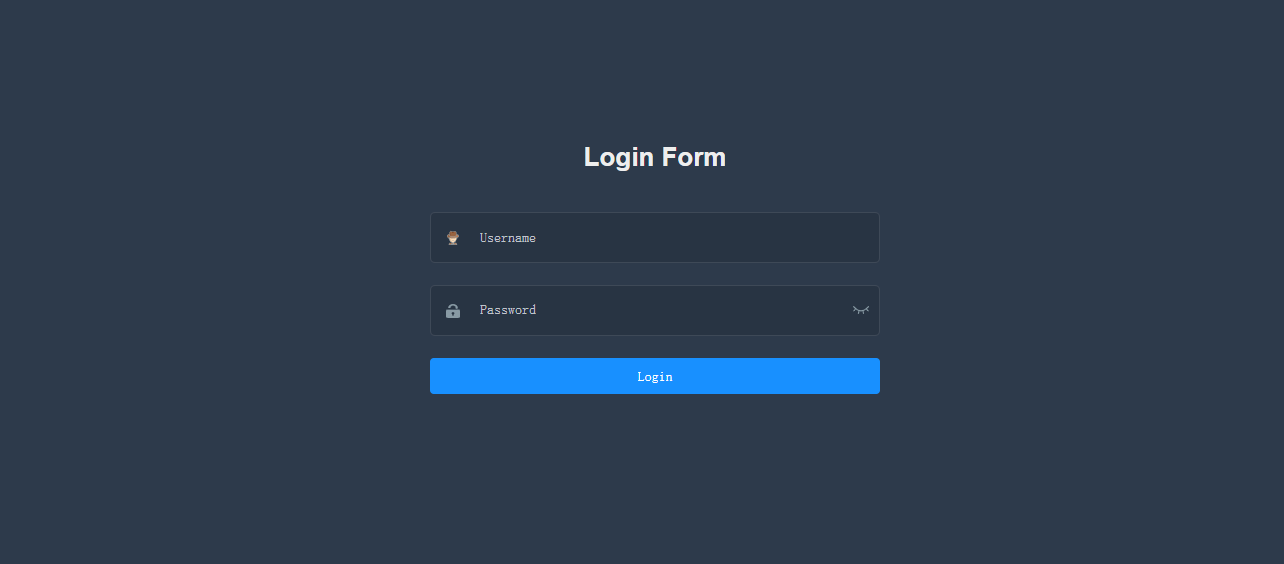


* 返回根目录执行./bin/start-all.sh



## 测试是否安装成功

* 访问浏览器：IP:9527/index.html



* 登录默认用户admin密码123456



* 点击执行器管理，可以看到datax执行器，安装成功

# CDH与datax-web使用和注意事项