Big Data - Concepts

20 December 2023 09:02

- Data Curation Services in Action,
- Open Exit: Reaching the End of The Data Life Cycle,
- The Current State of Meta-Repositories for Data
- Curation of Scientific Data at Risk of Loss: Data Rescue And Dissemination

- Data Curation Services in Action
  - Data curation services play a very critical role --> managing, organizing and enhancing the quality of data.
    - Data Discovery --> locating the data for the curation process
    - Metadata creation --> develop metadata (datasets origin, format, structure and other relevant details)
    - Data cleaning and standardization ---> clean and standardize data to remove errors, inconsistencies and redundancies. Hence we ensure accuracy and reliability, prevent misinterpretations and improve the overall quality for analysis
    - Version control --> track changes and updates to the datasets over time
    - Data integration
    - Data security any privacy measures
    - Data preservation
    - User access and training
    - Quality assurance and implement feedback mechanisms

- Reaching the End of The Data Life Cycle --> the conclusion or final stage of the data life cycle, it's often known as data archiving or data disposal.
  - Data archiving
    - Involves moving data that is no longer active / used - to a separate storage locations for long-term retention
    - These data are stored in secure and cost-effective manner in such a way that it can be retrieved if needed (compliance, historical analysis, or other purposes)
  - Data disposal
    - EOL of data which is no longer needed and this has to disposed to prevent unauthorized access
    - Organization data disposal practices ---> shredding of physical documents or permanently deleting electronic files.
  - Documentations and metadata preservations --> detailed documentation about the archived data, including metadata, ensures that future users can comprehend the context, structure and significance of the stored information
  - Compliance with regulations --> legal requirements for data retention periods and disposal methods.
  - Closure of Data-Related Projects
  - Data life cycle assessment --> Organizations periodically assess and review their data life cycle management practices.

Data management practices can vary depending on factors such as industry, regulatory environment, and organizational policies.
A well-managed data life cycle ensures that data is handled responsibly throughout its existence, from creation to the final stages of archiving or disposal.

- The Current State of Meta-Repositories for Data
  - Meta-repositories --> context of data typically refers to repositories that contain metadata bout various datasets, rather than the datasets themselves . They server the centralised hubs where users can discover, access and understand the characteristics of different data sets.
    - General state of meta-repositories for data
      - Data.gov - The United States government's open data website. It provides access to datasets published by agencies across the federal government. Data.gov is intended to provide access to government open data to the public, achieve agency missions, drive innovation, fuel economic activity, and uphold the ideals of an open and transparent government.
    - European Data portal
    - Google dataset search
    - Data catalog or research institution
    - Commercial data platforms
    - And so on ..
  - These will help people involved in data science and data management domains.

- Curation of Scientific Data at Risk of Loss : Data Rescue And Dissemination
  - Curation of Scientific Data at Risk of Loss this is often referred to as Data Rescue And Dissemination - which is a critical aspect of preserving valuable scientific information.
  - Scientific data may be at risk of loss due to various factors such as outdated storage formats, deterioration of physical media, lack of funding, or institutional changes.
  - The process of rescuing and disseminating data involves several key steps
    - Data identification
    - Assessment of data significance
    - Metadata extraction and enhancement
    - Data rescue techniques
    - Quality control and validations
    - Data standardizations
    - Creation of data repositories
    - Data dissemination strategies
    - Documentation of data reuse processes
    - Long-Term preservation

- **Comparison of Hadoop with Other Systems**

Hadoop --> mapreduce, batch processing , disk base storeage (HDFS), well suited for batch processing jobs but has higher latenct for certain use case
Apache Spark --> In-momory processing, multiple processing options (batch, interactive, streaming, machine learning), resilient Distributed datasets (RDDs), it is more versative and faster for iterative alogrithms and interactive data analysis
Apache Flink --> desinged for stram proceessing can handle both batch as well as realtime data, low-latency procesing
Apache Hive:
hadoop -> query language whcih provies a SQL live langugage for data anaylysis (HiveQL)
Apache Spark --> Spark SQL (performance advantages due to in-memory processing)
HBase : NoSQL database built on top of HDFS, designed for random read and wirte access to large data sets
Apache Cassandra -- Distributed NoSQL Database known for High availabity and scalability
Hbase is often associate d with the Hadoop while casandra can be used independently
HDFS Vs.HBase

Hadoop is a distributed storage and processing framework designed to handle large-scale data processing across a cluster of computers. While Hadoop is a powerful and widely used tool, there are other systems and frameworks that serve similar purposes or address specific aspects of big data processing. Here's a comparison of Hadoop with a few other systems:

1. Apache Spark:
   Hadoop:
- MapReduce Paradigm: Hadoop primarily uses the MapReduce programming model for distributed processing.
- Batch Processing: Suited for batch processing of large datasets.
- Disk-Based Storage: Hadoop Distributed File System (HDFS) is often used for distributed storage.

   Apache Spark:
- In-Memory Processing: Spark processes data in-memory, which can lead to faster processing compared to Hadoop's disk-based approach.
- Multiple Processing Models: Supports batch processing, interactive queries, streaming, and machine learning.
- Resilient Distributed Datasets (RDDs): Spark introduces the concept of RDDs, which can be cached in memory for iterative and interactive computations.

   Comparison:
- Spark is often considered more versatile and faster for iterative algorithms and interactive data analysis.
- Hadoop MapReduce is well-suited for batch processing jobs but may have higher latency for certain use cases.

2. Apache Flink:
   Hadoop:
- Batch Processing: Primarily designed for batch processing.
- Latency: Higher latency for real-time data processing.

   Apache Flink:
- Stream Processing: Flink is designed for stream processing and can handle both batch and real-time data.

- Low Latency: Flink provides low-latency processing, making it suitable for real-time analytics.
  Comparison:
- Flink is preferred for scenarios that require low-latency processing and support for stream processing.
- Hadoop is typically used for traditional batch processing workloads.

3. Apache Hive:
   Hadoop:
- Query Language: Hadoop ecosystem includes Hive, which provides a SQL-like query language (HiveQL) for data analysis.
- Schema on Read: Hive supports schema-on-read, allowing users to query data without a predefined schema.
  Apache Spark (with Spark SQL):
- In-Memory Processing: Spark SQL provides a SQL interface for querying structured data in Spark.
- Schema on Read: Spark also supports schema-on-read, similar to Hive.
  Comparison:
- Both Hive and Spark SQL provide SQL-like interfaces, but Spark SQL may have performance advantages due to Spark's in-memory processing capabilities.

4. HBase:
   Hadoop:
- HDFS vs. HBase: Hadoop includes HDFS for distributed storage, while HBase is a NoSQL database built on top of HDFS.
- Random Access: HBase is designed for random read and write access to large datasets.
  Apache Cassandra:
- Distributed NoSQL Database: Cassandra is a distributed NoSQL database known for its high availability and scalability.
- Partitioning: Cassandra uses partitioning for horizontal scalability.
  Comparison:
- HBase and Cassandra both provide distributed, scalable, and fault-tolerant NoSQL databases.
- HBase is often associated with the Hadoop ecosystem, while Cassandra can be used independently.

It's important to note that the choice between these systems depends on specific use cases, requirements, and the nature of data processing tasks. Many organizations use a combination of these tools within their big data ecosystems to address different aspects of data storage, processing, and analysis.

Hadoop Operation Modes

There are three supported by Hadoop clusters to operate

1. Local/Standalone Mode: After downloading Hadoop and installing it , by default is it configured in a standalone mode and can be used to run as single java process
2. Pseudo distributed mode: It is a distributed simulation on a single machine. Each Hadoop daemon such as hdfs, yarn, MapReduce etc.., will run as a separate java process. Very used for development purpose.
3. Fully distributed mode: Consists of minimum two or more machines as a cluster. (note: preferably try to Maintain similar configurations on all the nodes present in the cluster)

Installing Hadoop in Pseudo Distributed Mode

1. Setting up Hadoop
   $ hadoop version
   Edit the ~/.bashrc file and configure the variables
   $ su - root
   $ cd $HADOOP_HOME/etc/hadoop ---> all the Hadoop configuration files are present in this locations.
   Note: for making changes to these file you can also login as a privileged user
   # nano /etc/bash.bashrc
   <add the following lines to the end of the file >

```
export JAVA_HOME=/usr/local/jdk1.7.0_80
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_INSTALL=$HADOOP_HOME
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

   <save and exit>

   # source /etc/bash.bashrc [changes done in this file will be applicable to all the users]
   Note: if you have updated only in local users bash then you need to use $ source ~/.bashrc [changes done in this file will be applicable to only particular user login only]
   # env

   # cd $HADOOP_HOME/etc/hadoop
   # nano hadoop-env.sh
   Replace the following value
   export JAVA_HOME=/usr/local/jdk1.7.0_80
   <save and exit>

   # nano core-site.xml

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

   <save and exit>

   # nano hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
  </property>
</configuration>
```

   <save and exit>

   # nano yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

   <save and exit>

   # cp mapred-site.xml.template mapred-site.xml

```
# nano mapred-site.xml
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>

# cd /usr/local
# chown -R hadoop.hadoop  hadoop

Login as user hadoop and perform the following activity
$ cd ~
$ hdfs namenode -format
```

```
hadoop@mainserver1:~$ hdfs namenode -format
23/12/20 09:01:51 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = mainserver1/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.4.1
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/jets3t-
0.9.0.jar:/usr/local/hadoop/share/hadoop/common/lib/zookeeper-3.4.5.jar:/usr/local/hadoop/share/hadoop/co
mmon/lib/commons-beanutils-core-1.8.0.jar:/usr/local/hadoop/share/hadoop/common/lib/jsch-0.1.42.jar:/usr/
local/hadoop/share/hadoop/common/lib/xmlenc-0.52.jar:/usr/local/hadoop/share/hadoop/common/lib/activation
-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/slf4j-api-1.7.5.jar:/usr/local/hadoop/share/hadoop/com
mon/lib/jettison-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/guava-11.0.2.jar:/usr/local/hadoop/sha
re/hadoop/common/lib/jackson-xc-1.8.8.jar:/usr/local/hadoop/share/hadoop/common/lib/jsr305-1.3.9.jar:/usr
/local/hadoop/share/hadoop/common/lib/servlet-api-2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commo
ns-collections-3.2.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-core-1.9.jar:/usr/local/hadoop/
share/hadoop/common/lib/commons-io-2.4.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-6.1.26.jar:/us
r/local/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.8.8.jar:/usr/local/hadoop/share/hadoop/common/lib/
httpcore-4.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/usr/local/hadoop/shar
e/hadoop/common/lib/jackson-mapper-asl-1.8.8.jar:/usr/local/hadoop/share/hadoop/common/lib/asm-3.2.jar:/u
sr/local/hadoop/share/hadoop/common/lib/commons-compress-1.4.1.jar:/usr/local/hadoop/share/hadoop/common/
lib/java-xmlbuilder-0.4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-digester-1.8.jar:/usr/local
/hadoop/share/hadoop/common/lib/commons-lang-2.6.jar:/usr/local/hadoop/share/hadoop/common/lib/jsp-api-2.
1.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-util-6.1.26.jar:/usr/local/hadoop/share/hadoop/comm
on/lib/mockito-all-1.8.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-math3-3.1.1.jar:/usr/local
/hadoop/share/hadoop/common/lib/stax-api-1.0-2.jar:/usr/local/hadoop/share/hadoop/common/lib/httpclient-4
.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-logging-1.1.3.jar:/usr/local/hadoop/share/hado
op/common/lib/protobuf-java-2.5.0.jar:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar:/
usr/local/hadoop/share/hadoop/common/lib/netty-3.6.2.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/
```

```
STARTUP_MSG:   build = http://svn.apache.org/repos/asf/hadoop/common -r 1604318; compiled by 'jenkins' on
 2014-06-21T05:43Z
STARTUP_MSG:   java = 1.7.0_80
************************************************************/
23/12/20 09:01:51 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]
23/12/20 09:01:51 INFO namenode.NameNode: createNameNode [-format]
23/12/20 09:01:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
Formatting using clusterid: CID-c2f8ca6a-6d08-42c7-8b4c-dbd46e58e7f9
23/12/20 09:01:55 INFO namenode.FSNamesystem: fsLock is fair:true
23/12/20 09:01:55 INFO namenode.HostFileManager: read includes:
HostSet(
)
23/12/20 09:01:55 INFO namenode.HostFileManager: read excludes:
HostSet(
)
23/12/20 09:01:55 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000
23/12/20 09:01:55 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-ch
eck=true
23/12/20 09:01:55 INFO util.GSet: Computing capacity for map BlocksMap
23/12/20 09:01:55 INFO util.GSet: VM type       = 64-bit
23/12/20 09:01:55 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB
23/12/20 09:01:55 INFO util.GSet: capacity      = 2^21 = 2097152 entries
23/12/20 09:01:55 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false
23/12/20 09:01:55 INFO blockmanagement.BlockManager: defaultReplication         = 1
23/12/20 09:01:55 INFO blockmanagement.BlockManager: maxReplication             = 512
23/12/20 09:01:55 INFO blockmanagement.BlockManager: minReplication             = 1
23/12/20 09:01:55 INFO blockmanagement.BlockManager: maxReplicationStreams      = 2
23/12/20 09:01:55 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks  = false
```

```
23/12/20 09:01:55 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000
23/12/20 09:01:55 INFO blockmanagement.BlockManager: encryptDataTransfer        = false
23/12/20 09:01:55 INFO blockmanagement.BlockManager: maxNumBlocksToLog          = 1000
23/12/20 09:01:55 INFO namenode.FSNamesystem: fsOwner             = hadoop (auth:SIMPLE)
23/12/20 09:01:55 INFO namenode.FSNamesystem: supergroup          = supergroup
23/12/20 09:01:55 INFO namenode.FSNamesystem: isPermissionEnabled = true
23/12/20 09:01:55 INFO namenode.FSNamesystem: HA Enabled: false
23/12/20 09:01:55 INFO namenode.FSNamesystem: Append Enabled: true
23/12/20 09:01:56 INFO util.GSet: Computing capacity for map INodeMap
23/12/20 09:01:56 INFO util.GSet: VM type       = 64-bit
23/12/20 09:01:56 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB
23/12/20 09:01:56 INFO util.GSet: capacity      = 2^20 = 1048576 entries
23/12/20 09:01:56 INFO namenode.NameNode: Caching file names occuring more than 10 times
23/12/20 09:01:56 INFO util.GSet: Computing capacity for map cachedBlocks
23/12/20 09:01:56 INFO util.GSet: VM type       = 64-bit
23/12/20 09:01:56 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB
23/12/20 09:01:56 INFO util.GSet: capacity      = 2^18 = 262144 entries
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
23/12/20 09:01:56 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
23/12/20 09:01:56 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache ent
ry expiry time is 600000 millis
23/12/20 09:01:56 INFO util.GSet: Computing capacity for map NameNodeRetryCache
23/12/20 09:01:56 INFO util.GSet: VM type       = 64-bit
23/12/20 09:01:56 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
23/12/20 09:01:56 INFO util.GSet: capacity      = 2^15 = 32768 entries
23/12/20 09:01:56 INFO namenode.AclConfigFlag: ACLs enabled? false
23/12/20 09:01:57 INFO namenode.FSImage: Allocated new BlockPoolId: BP-737919163-127.0.1.1-1703062916845
```

```
23/12/20 09:01:56 INFO util.GSet: capacity      = 2^18 = 262144 entries
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
23/12/20 09:01:56 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
23/12/20 09:01:56 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
23/12/20 09:01:56 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache ent
ry expiry time is 600000 millis
23/12/20 09:01:56 INFO util.GSet: Computing capacity for map NameNodeRetryCache
23/12/20 09:01:56 INFO util.GSet: VM type       = 64-bit
23/12/20 09:01:56 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
23/12/20 09:01:56 INFO util.GSet: capacity      = 2^15 = 32768 entries
23/12/20 09:01:56 INFO namenode.AclConfigFlag: ACLs enabled? false
23/12/20 09:01:57 INFO namenode.FSImage: Allocated new BlockPoolId: BP-737919163-127.0.1.1-1703062916845
23/12/20 09:01:57 INFO common.Storage: Storage directory /home/hadoop/hadoopinfra/hdfs/namenode has been
successfully formatted.
23/12/20 09:01:58 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
23/12/20 09:01:58 INFO util.ExitUtil: Exiting with status 0
23/12/20 09:01:58 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at mainserver1/127.0.1.1
************************************************************/
hadoop@mainserver1:~$ ss -ant
State      Recv-Q      Send-Q           Local Address:Port           Peer Address:Port      Process
LISTEN     0           4096             127.0.0.53%lo:53                 0.0.0.0:*
LISTEN     0           128                    0.0.0.0:22                 0.0.0.0:*
ESTAB      0           64              192.168.56.100:22           192.168.56.1:17861
ESTAB      0           0               192.168.56.100:22           192.168.56.1:41537
LISTEN     0           128                       [::]:22                    [::]:*
```

$ start-dfs.sh

```
hadoop@mainserver1:~$ start-dfs.sh
23/12/20 09:09:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
Starting namenodes on [localhost]
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:AojXpepX/dNKoHN/t6wzgvHeXgCYE2SOJ8mqo09II1s.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
localhost: Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-mainserver1.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-mainserver1.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is SHA256:AojXpepX/dNKoHN/t6wzgvHeXgCYE2SOJ8mqo09II1s.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondarynamenode-ma
inserver1.out
23/12/20 09:10:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
```

```
hadoop@mainserver1:~$ ss -ant
State          Recv-Q     Send-Q           Local Address:Port           Peer Address:Port      Process
LISTEN         0          128                    0.0.0.0:50020               0.0.0.0:*
LISTEN         0          128                127.0.0.1:9000                0.0.0.0:*
LISTEN         0          128                    0.0.0.0:50090               0.0.0.0:*
LISTEN         0          4096           127.0.0.53%lo:53                    0.0.0.0:*
LISTEN         0          128                    0.0.0.0:50070               0.0.0.0:*
LISTEN         0          128                    0.0.0.0:22                  0.0.0.0:*
LISTEN         0          50                     0.0.0.0:50010               0.0.0.0:*
LISTEN         0          128                    0.0.0.0:50075               0.0.0.0:*
ESTAB          0          64             192.168.56.100:22           192.168.56.1:17861
ESTAB          0          0                127.0.0.1:37078             127.0.0.1:9000
ESTAB          0          0              192.168.56.100:22           192.168.56.1:41537
TIME-WAIT      0          0                127.0.0.1:59378             127.0.0.1:9000
ESTAB          0          0                127.0.0.1:9000              127.0.0.1:37078
LISTEN         0          128                   [::]:22                     [::]:*
hadoop@mainserver1:~$
```

```
hadoop@mainserver1:~$ tree
.
└── hadoopinfra
    └── hdfs
        ├── datanode
        │   ├── current
        │   │   ├── BP-737919163-127.0.1.1-1703062916845
        │   │   │   ├── current
        │   │   │   │   ├── finalized
        │   │   │   │   ├── rbw
        │   │   │   │   └── VERSION
        │   │   │   ├── dncp_block_verification.log.curr
        │   │   │   └── tmp
        │   │   └── VERSION
        │   └── in_use.lock
        └── namenode
            ├── current
            │   ├── edits_0000000000000000001-0000000000000000002
            │   ├── edits_inprogress_0000000000000000003
            │   ├── fsimage_0000000000000000000
            │   ├── fsimage_0000000000000000000.md5
            │   ├── fsimage_0000000000000000002
            │   ├── fsimage_0000000000000000002.md5
            │   ├── seen_txid
            │   └── VERSION
            └── in_use.lock

11 directories, 13 files
hadoop@mainserver1:~$
```

$ start-yarn.sh

```
hadoop@mainserver1:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-mainserver1.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-mainserver1.ou
t
```

```
hadoop@mainserver1:~$ ss -ant
State       Recv-Q      Send-Q            Local Address:Port             Peer Address:Port     Process
LISTEN      0           128                   0.0.0.0:50020                 0.0.0.0:*
LISTEN      0           128               127.0.0.1:9000                    0.0.0.0:*
LISTEN      0           128                   0.0.0.0:50090                 0.0.0.0:*
LISTEN      0           4096           127.0.0.53%lo:53                     0.0.0.0:*
LISTEN      0           128                   0.0.0.0:50070                 0.0.0.0:*
LISTEN      0           128                   0.0.0.0:22                    0.0.0.0:*
LISTEN      0           50                    0.0.0.0:50010                 0.0.0.0:*
LISTEN      0           128                   0.0.0.0:50075                 0.0.0.0:*
ESTAB       0           64              192.168.56.100:22              192.168.56.1:17861
ESTAB       0           0                 127.0.0.1:37078                 127.0.0.1:9000
ESTAB       0           0               192.168.56.100:22              192.168.56.1:41537
TIME-WAIT   0           0                 127.0.0.1:41310                 127.0.0.1:9000
TIME-WAIT   0           0                 127.0.0.1:39896                 127.0.0.1:22
ESTAB       0           0                 127.0.0.1:9000                  127.0.0.1:37078
LISTEN      0           128                       *:8033                        *:*
LISTEN      0           128                       *:35715                       *:*
LISTEN      0           128                       *:8040                        *:*
LISTEN      0           128                       *:8042                        *:*
LISTEN      0           128                    [::]:22                       [::]:*
LISTEN      0           128                       *:8088                        *:*
LISTEN      0           50                        *:13562                       *:*
LISTEN      0           128                       *:8030                        *:*
LISTEN      0           128                       *:8031                        *:*
LISTEN      0           128                       *:8032                        *:*
ESTAB       0           0          [::ffff:127.0.0.1]:32786      [::ffff:127.0.1.1]:8031
ESTAB       0           0          [::ffff:127.0.1.1]:8031       [::ffff:127.0.0.1]:32786
hadoop@mainserver1:~$
```

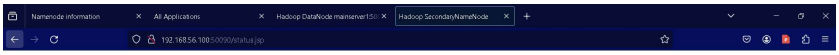
It looks like you are making an HTTP request to a Hadoop IPC port. This is not the correct port for the web interface on this daemon.

DataNode 'mainserver1:50010' (active)

Version:	2.4.1, 1604318
Compiled:	2014-06-21T05:43Z by jenkins from branch-2.4.1
Cluster ID:	CID-c2f8ca6a-6d08-42c7-8b4c-dbd46e58e7f9

DataNode Logs
View/Set Log Level
Metrics
Configuration
Block Scanner Report

Hadoop, 2023.



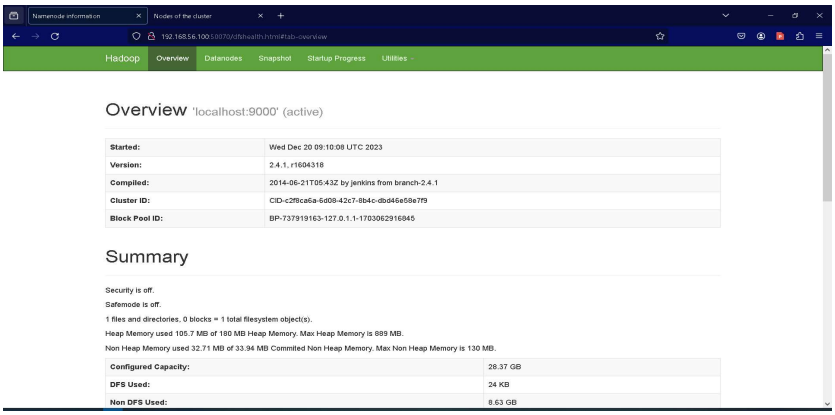
SecondaryNameNode

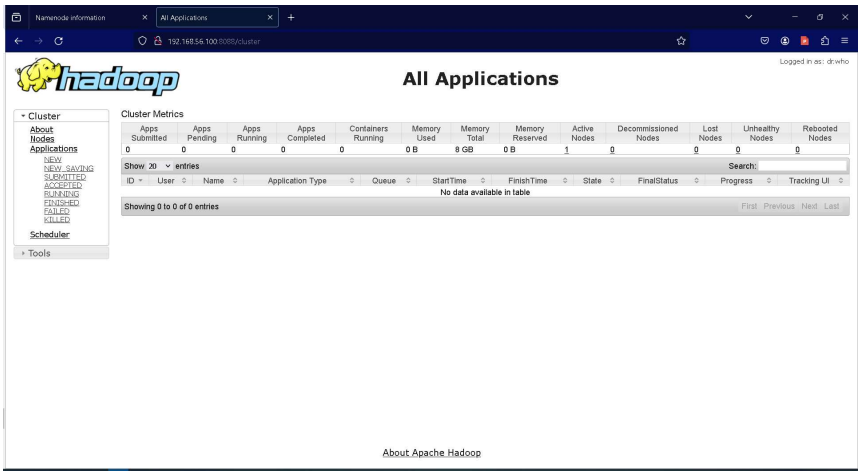
Version:	2.4.1, 1604318
Compiled:	2014-06-21T05:43Z by jenkins from branch-2.4.1

```
SecondaryNameNode Status
Name Node Address    : localhost/127.0.0.1:9000
Start Time           : Wed Dec 20 09:10:26 UTC 2023
Last Checkpoint Time : Wed Dec 20 09:11:32 UTC 2023
Checkpoint Period    : 3600 seconds
Checkpoint Size      : 976.56 KB (= 1000000 bytes)
Checkpoint Dirs      : [file:///tmp/hadoop-hadoop/dfs/namesecondary]
Checkpoint Edits Dirs: [file:///tmp/hadoop-hadoop/dfs/namesecondary]
```

Logs

Hadoop, 2023.





NOTE: after rebooting the system you need to start the following daemons $ start-dfs.sh and $start-yarn.sh use $ ss -ant to very for the open ports

******** ------ Installation of Hadoop Complete ------ ********