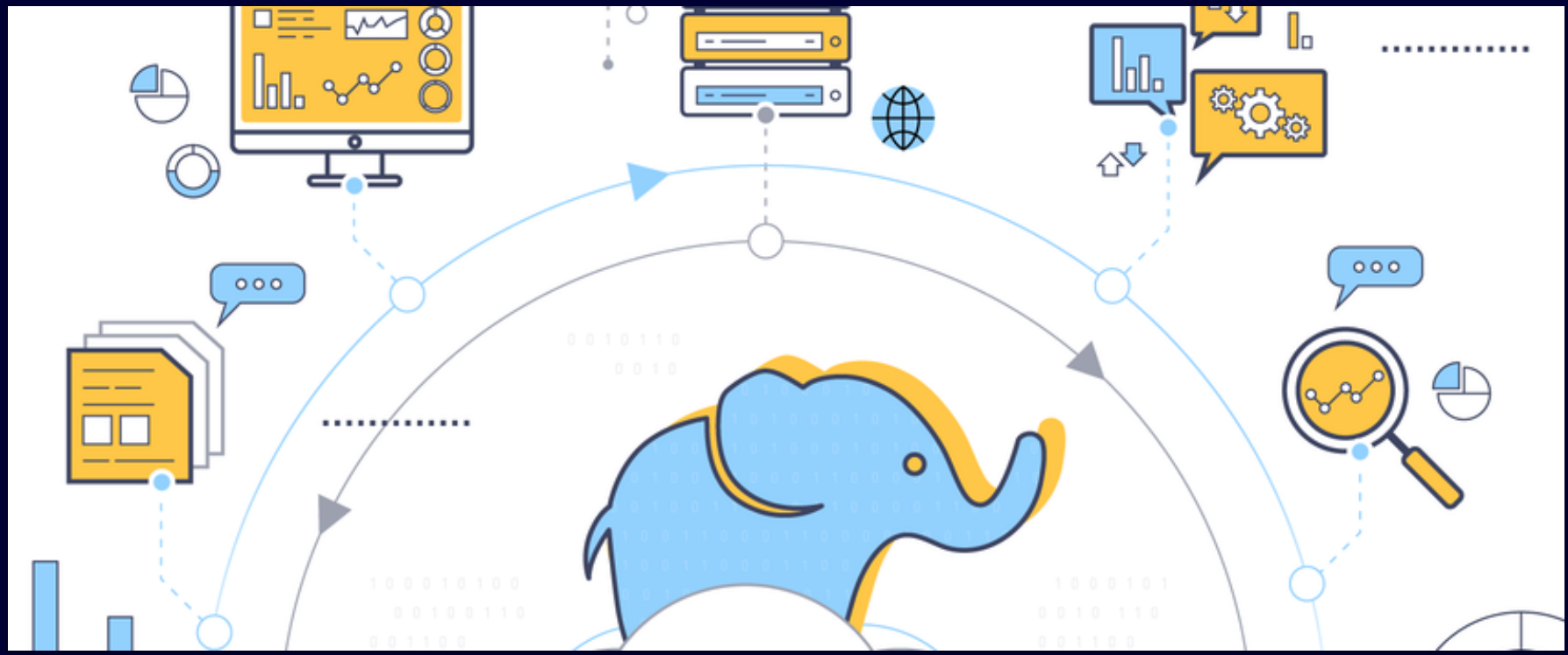


Hadoop Part 2

BY AARON JOHNS

Best practices Hadoop deployment

- **Start small:** Like other software projects, an implementation Hadoop also involves risks and cost. It's always better to set up a small Hadoop cluster of four nodes. This small cluster can be set up as proof of concept (POC). Before using any Hadoop component, it can be added to the existing Hadoop POC cluster as proof of technology (POT). It allows the infrastructure and development team to understand big data project requirements. After successful completion of POC and POT, additional nodes can be added to the existing cluster.

- **Hadoop cluster monitoring:** Proper monitoring of the NameNode and all DataNodes is required to understand the health of the cluster. It helps to take corrective actions in the event of node problems. If a service goes down, timely action can help avoid big problems in the future. Setting up Gangalia and Nagios are popular choices to configure alerts and monitoring. In the case of the Hortonworks cluster, Ambari monitoring, and the Cloudera cluster, Cloudera (CDH) manager monitoring can be an easy setup.

- **Automated deployment:** Use of tools like Puppet or Chef is essential for Hadoop deployment. It becomes super easy and productive to deploy the Hadoop cluster with automated tools instead of manual deployment. Give importance to data analysis and data processing using available tools/components. Give preference to using Hive or Pig scripts for problem solving rather than writing heavy, custom MapReduce code. The goal should be to develop less and analyze more.

- **Implementation of HA:** While deciding about HA infrastructure and architecture, careful consideration should be given to any increase in demand and data growth. In the event of any failure or crash, the system should be able to recover itself or failover to another data center/site.
- **Security:** Data needs to be protected by creating users and groups, and mapping users to the groups. Setting appropriate permissions and enforcing strong passwords should lock each user group down.
- **Data protection:** The identification of sensitive data is critical before moving it to the Hadoop cluster. It's very important to understand privacy policies and government regulations for the better identification and mitigation of compliance exposure risks.

Hadoop file formats

- **Text/CSV file**

  Text and CSV files are very common in Hadoop data processing algorithms. Each line in the file is treated as a new record.

- **JSON**

  The JSON format is typically used in data exchange applications and it is treated as an object, record, struct, or an array. These files are text files and support schema evolutions. It's very easy to add or delete attributes from a JSON file.

- **Sequence file**

  A sequence file is a flat file consisting of binary key/value pairs. They are extensively used in MapReduce as input/output formats. They are mostly used for intermediate data storage within a sequence of MapReduce jobs.

- **Avro**

  Avro is a widely used file type within the Hadoop community. It is popular because it helps schema evolution. It contains serialized data with a binary format. An Avro file is splittable and supports block compression. It contains data and metadata. It uses a separate JSON file to define the schema format. When Avro data is stored in a file, its schema is stored with it so that files may be processed later by any program.

- **Parquet**

  Parquet stores nested data structures in a flat columnar format. Parquet is more efficient in terms of storage and performance than any row-level file formats. Parquet stores binary data in a column-oriented way. In the Parquet format, new columns are added at the end of the structure.

- **ORC**

  ORC files are optimized record columnar file format and are the extended version of RC files. These are great for compression and are best suited for Hive SQL performance when Hive is reading, writing, and processing data to reduce access time and the storage space.

Batch processing versus real-time processing

Batch processing

- Very efficient in processing a high volume of data.
- All data processing steps (that is, data collection, data ingestion, data processing, and results presentation) are done as one single batch job.
- Throughput carries more importance than latency. Latency is always more than a single minute.
- Throughput directly depends on the size of the data and available computational system resources.
- Available tools include Apache Sqoop, MapReduce jobs, Spark jobs, Hadoop DistCp utility, and so on.

Real-time processing

- Latency is extremely important, for example, less than one second
- Computation is relatively simple
- Data is processed as an independent unit
- Available tools include Apache Storm, Spark Streaming, Apache Fink, Apache Kafka, and so on

Installing Hadoop on Ubuntu Server

First Install JDK11

```
sudo apt install openjdk-11-jdk -y
```

Once the installation process is complete, verify the current Java version:

```
aaronjohns@aaron-hadoop:~$ java -version; javac -version
openjdk version "11.0.10" 2021-01-19
OpenJDK Runtime Environment (build 11.0.10+9-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.10+9-Ubuntu-0ubuntu1.20.04, mixed mode, sha
ring)
javac 11.0.10
```

Install OpenSSH

```
sudo apt install openssh-server openssh-client -y
```

Create a user named hadoop and set the password for this user

```
sudo adduser hadoop
```

Then switch to that user

```
su - hadoop
```

Generate an SSH key pair for this user

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
chmod 0600 ~/.ssh/authorized_keys
```

Download hadoop

```
wget https://mirrors.estointernet.in/apache/hadoop/common/
hadoop-3.2.2/hadoop-3.2.2.tar.gz
```

Extract the files

```
tar xzf hadoop-3.2.2.tar.gz
```

Since we are deploying Hadoop on a single node, we do the following settings

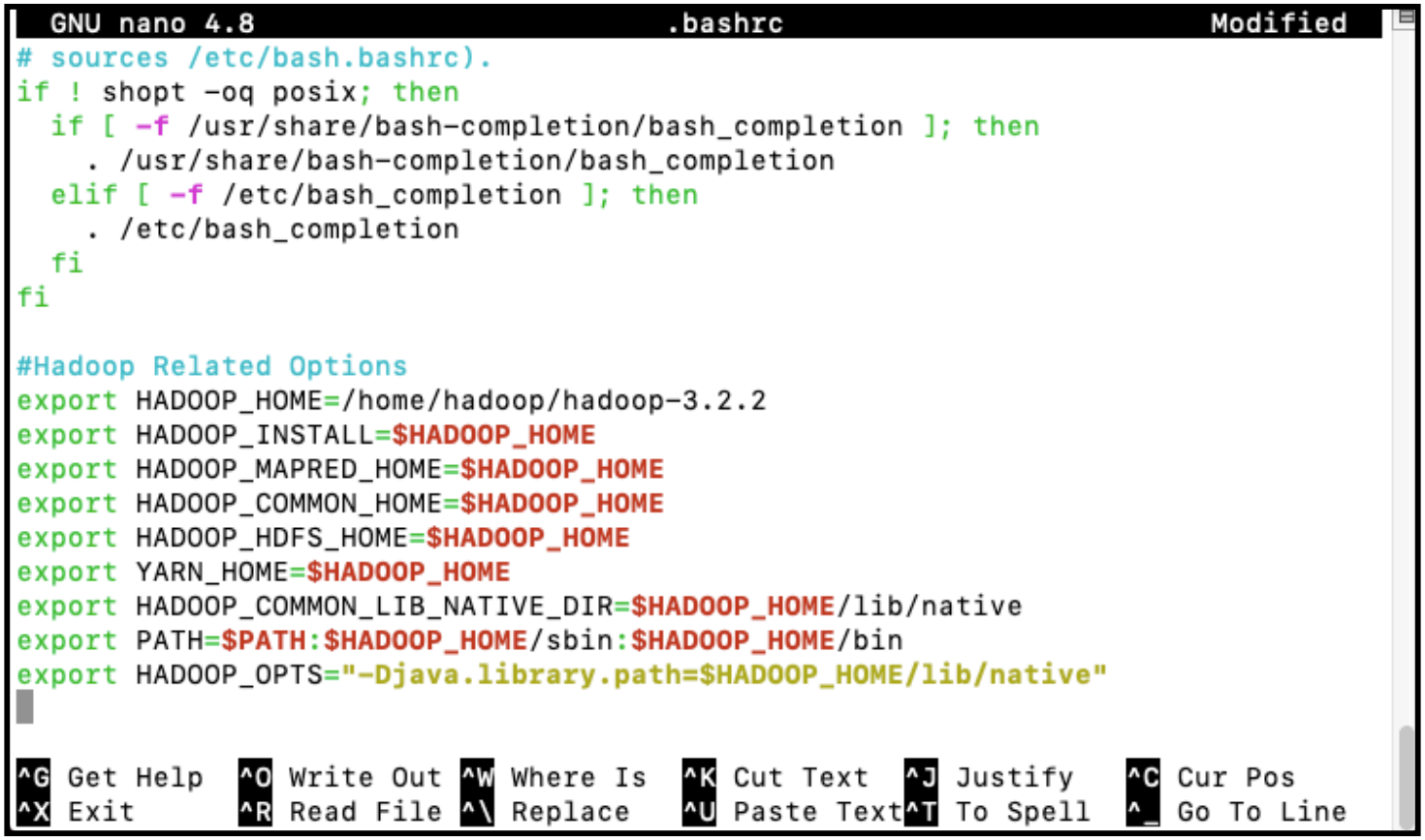
Open the .bashrc file and append the following to the bottom of the file. Then exit the file after you make the changes.

```
nano .bashrc
```

```
#Hadoop Related Options
export HADOOP_HOME=/home/hadoop/hadoop-3.2.2
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$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 PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Apply the changes to the current running environment

```
source ~/.bashrc
```

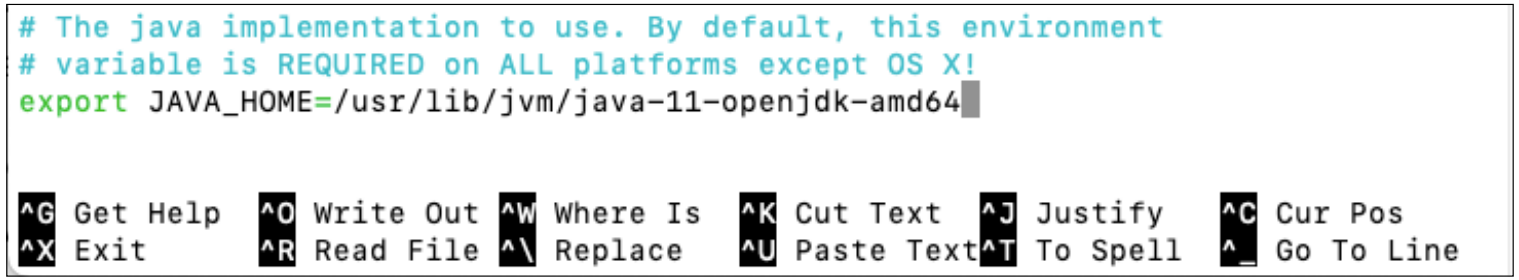

GNU nano 4.8 .bashrc Modified
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
if [-f /usr/share/bash-completion/bash_completion]; then
. /usr/share/bash-completion/bash_completion
elif [-f /etc/bash_completion]; then
. /etc/bash_completion
fi
fi
#Hadoop Related Options
export HADOOP_HOME=/home/hadoop/hadoop-3.2.2
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$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 PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

Find the path to your jdk and note it down. The area to note down is highlighted with a red box

```
aaronjohns@aaron-hadoop:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-11-openjdk-amd64/bin/javac
aaronjohns@aaron-hadoop:~$
```

Edit the hadoop-env.sh file. Uncomment the $JAVA_HOME variable (i.e., remove the # sign) and add the full path to the OpenJDK installation on your system.

```
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

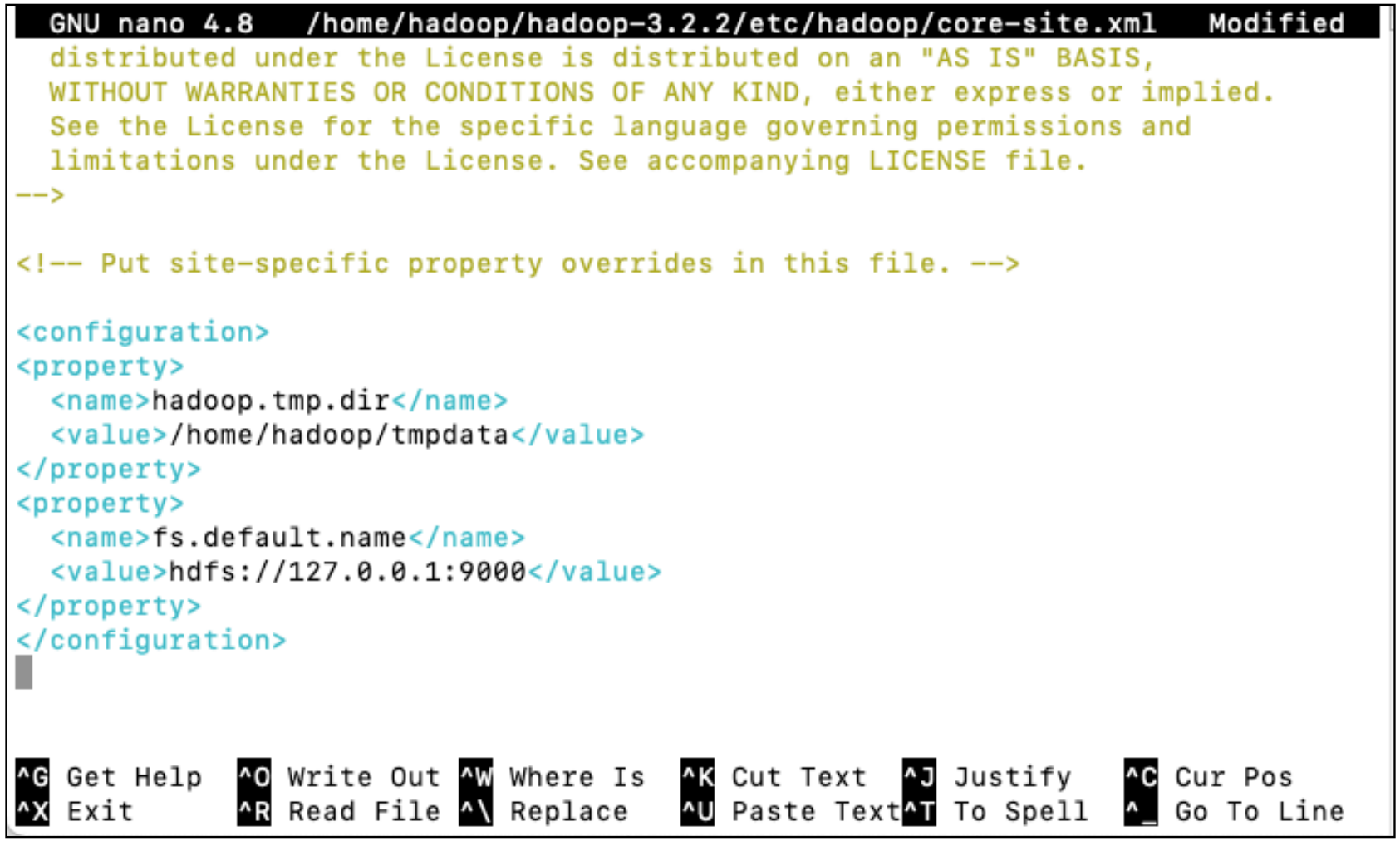


Edit the core-site.xml file.

```
nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hadoop/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

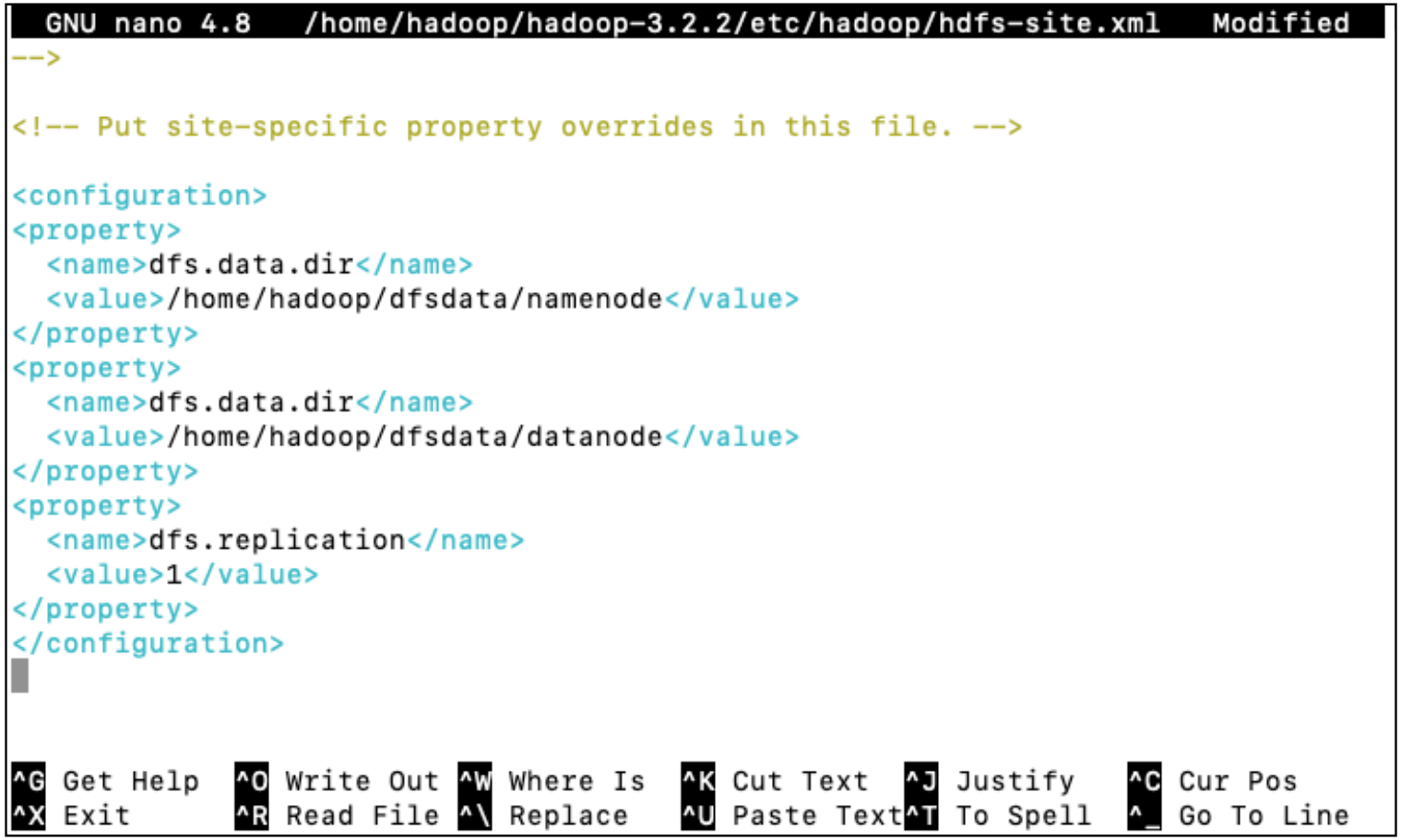

GNU nano 4.8 /home/hadoop/hadoop-3.2.2/etc/hadoop/core-site.xml Modified
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hadoop/tmpdata</value>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

Edit the hdfs-site.xml File

```
nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Add the following configuration to the file

```
<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hadoop/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hadoop/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>
```


GNU nano 4.8 /home/hadoop/hadoop-3.2.2/etc/hadoop/hdfs-site.xml Modified
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>dfs.data.dir</name>
<value>/home/hadoop/dfsdata/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/home/hadoop/dfsdata/datanode</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

Edit the mapred-site.xml File

```
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

Add the following configuration to the file

```
<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```


GNU nano 4.8 /home/hadoop/hadoop-3.2.2/etc/hadoop/mapred-site.xml Modified
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

Edit the yarn-site.xml File

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

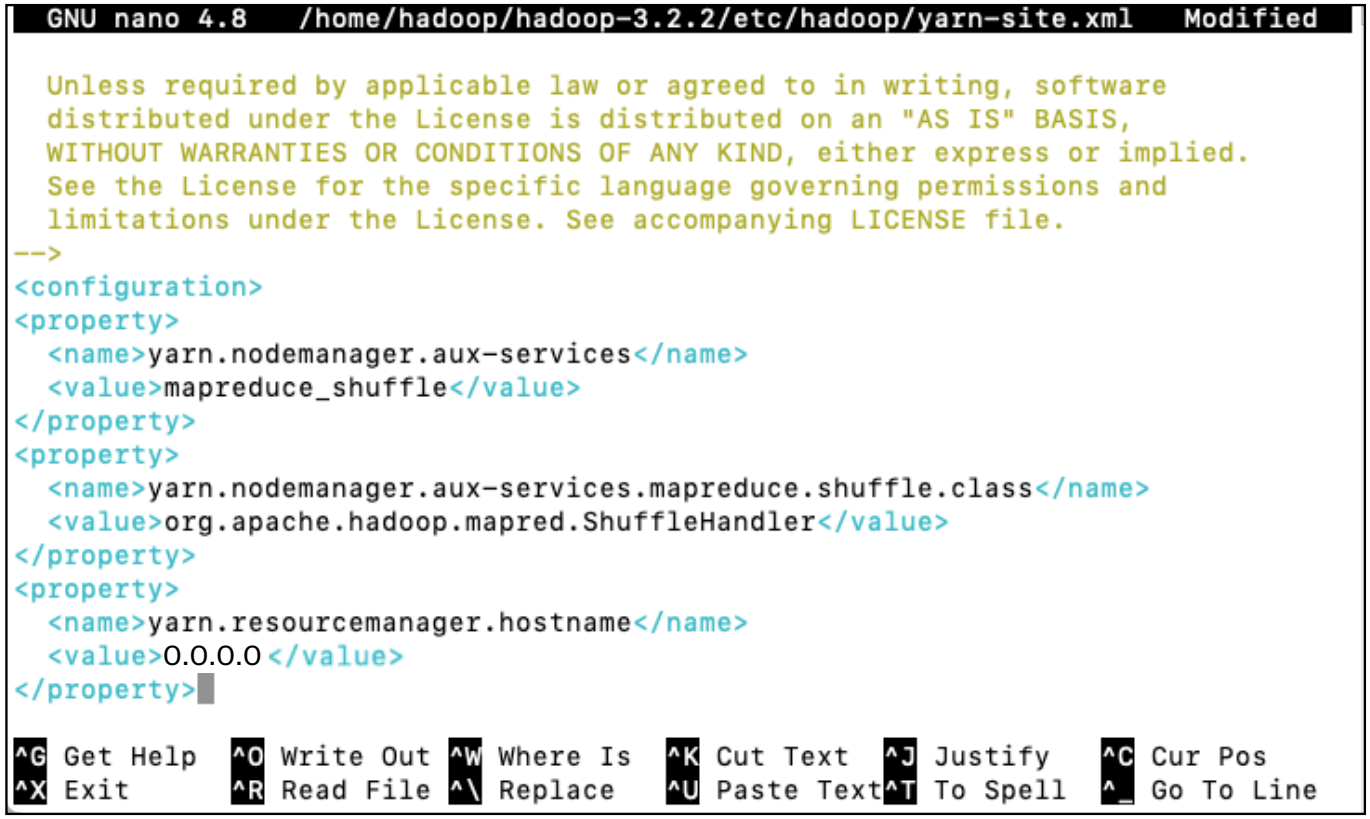
Add the following configuration to the file

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

<value>org.apache.hadoop.mapred.Shuff
leHandler</value>
</property>
<property>

<name>yarn.resourcemanager.hostname</
name>
  <value>0.0.0.0</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-
whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,H
ADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASS
PATH_PERPEND_DISTCACHE,HADOOP_YARN_HO
ME,HADOOP_MAPRED_HOME</value>
</property>
</configuration>
```


GNU nano 4.8 /home/hadoop/hadoop-3.2.2/etc/hadoop/yarn-site.xml Modified
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>0.0.0.0 </value>
</property>
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

Format the NameNode before starting Hadoop services

```
hdfs namenode -format
```

The shutdown notification signifies the end of the NameNode format process.

```
2021-04-05 10:04:59,941 INFO util.GSet: Computing capacity for map NameNodeRetry
Cache
2021-04-05 10:04:59,941 INFO util.GSet: VM type       = 64-bit
2021-04-05 10:04:59,941 INFO util.GSet: 0.029999999329447746% max memory 984 MB
= 302.3 KB
2021-04-05 10:04:59,941 INFO util.GSet: capacity      = 2^15 = 32768 entries
2021-04-05 10:04:59,963 INFO namenode.FSImage: Allocated new BlockPoolId: BP-149
4305226-127.0.1.1-1617617099956
2021-04-05 10:04:59,981 INFO common.Storage: Storage directory /home/hadoop/tmpd
ata/dfs/name has been successfully formatted.
2021-04-05 10:05:00,021 INFO namenode.FSImageFormatProtobuf: Saving image file /
home/hadoop/tmpdata/dfs/name/current/fsimage.ckpt_0000000000000000000 using no c
ompression
2021-04-05 10:05:00,095 INFO namenode.FSImageFormatProtobuf: Image file /home/ha
doop/tmpdata/dfs/name/current/fsimage.ckpt_0000000000000000000 of size 401 bytes
 saved in 0 seconds .
2021-04-05 10:05:00,113 INFO namenode.NNStorageRetentionManager: Going to retain
 1 images with txid >= 0
2021-04-05 10:05:00,119 INFO namenode.FSImage: FSImageSaver clean checkpoint: tx
id=0 when meet shutdown.
2021-04-05 10:05:00,120 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at aaron-hadoop/127.0.1.1
************************************************************/
```

Start the namenode and datanode.

```
cd ~/hadoop-3.2.2/sbin
```

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

```
hadoop@aaron-hadoop:~/hadoop-3.2.2/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [aaron-hadoop]
aaron-hadoop: Warning: Permanently added 'aaron-hadoop' (ECDSA) to the list of k
nown hosts.
```

Start the YARN resource and nodemanagers

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

```
hadoop@aaron-hadoop:~/hadoop-3.2.2/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

Check if all the daemons are active and running as Java processes

`jps`

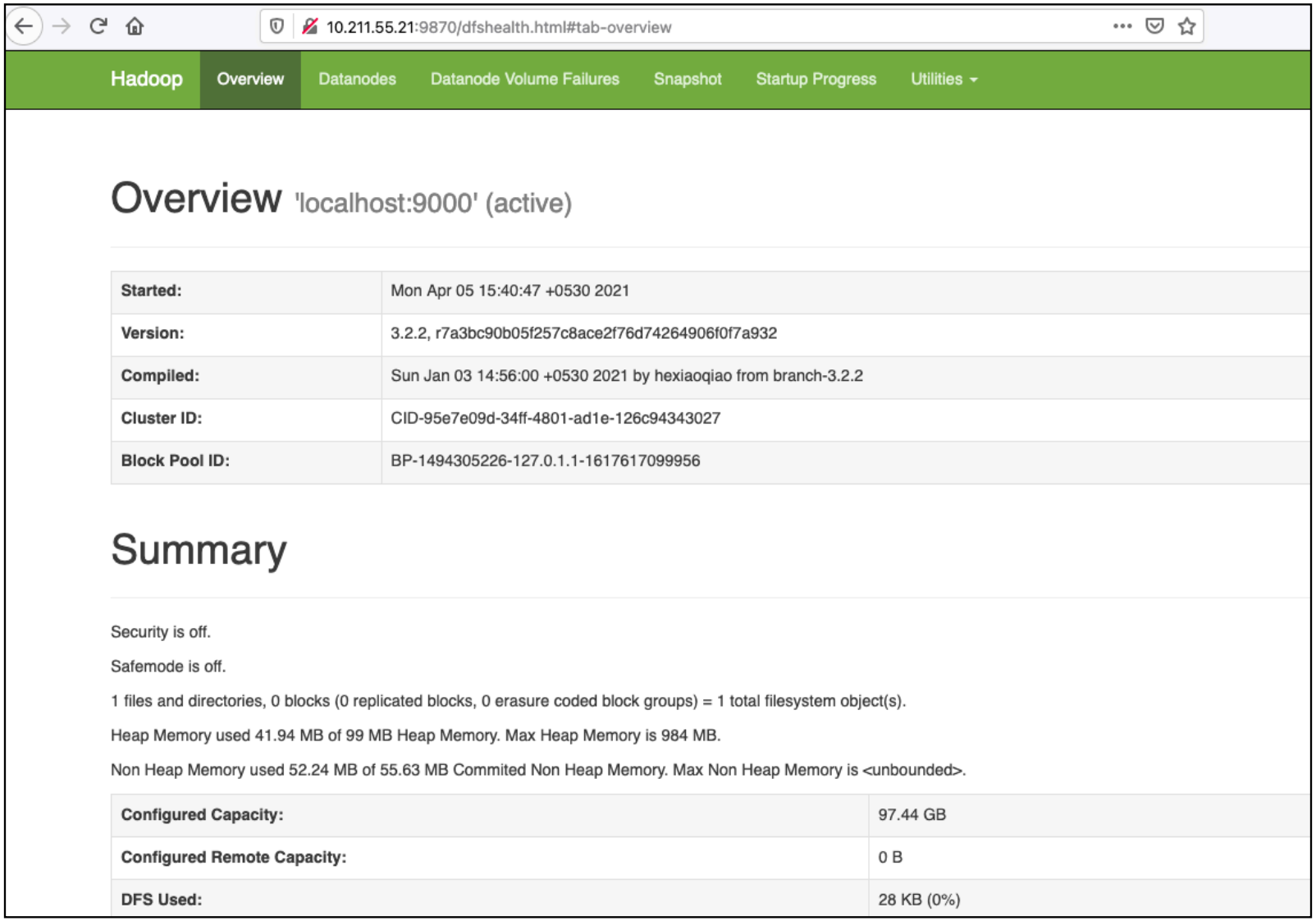
```
hadoop@aaron-hadoop:~/hadoop-3.2.2/sbin$ jps
4611 Jps
3878 SecondaryNameNode
3451 NameNode
3645 DataNode
4094 ResourceManager
4271 NodeManager
```

Note your ip address down

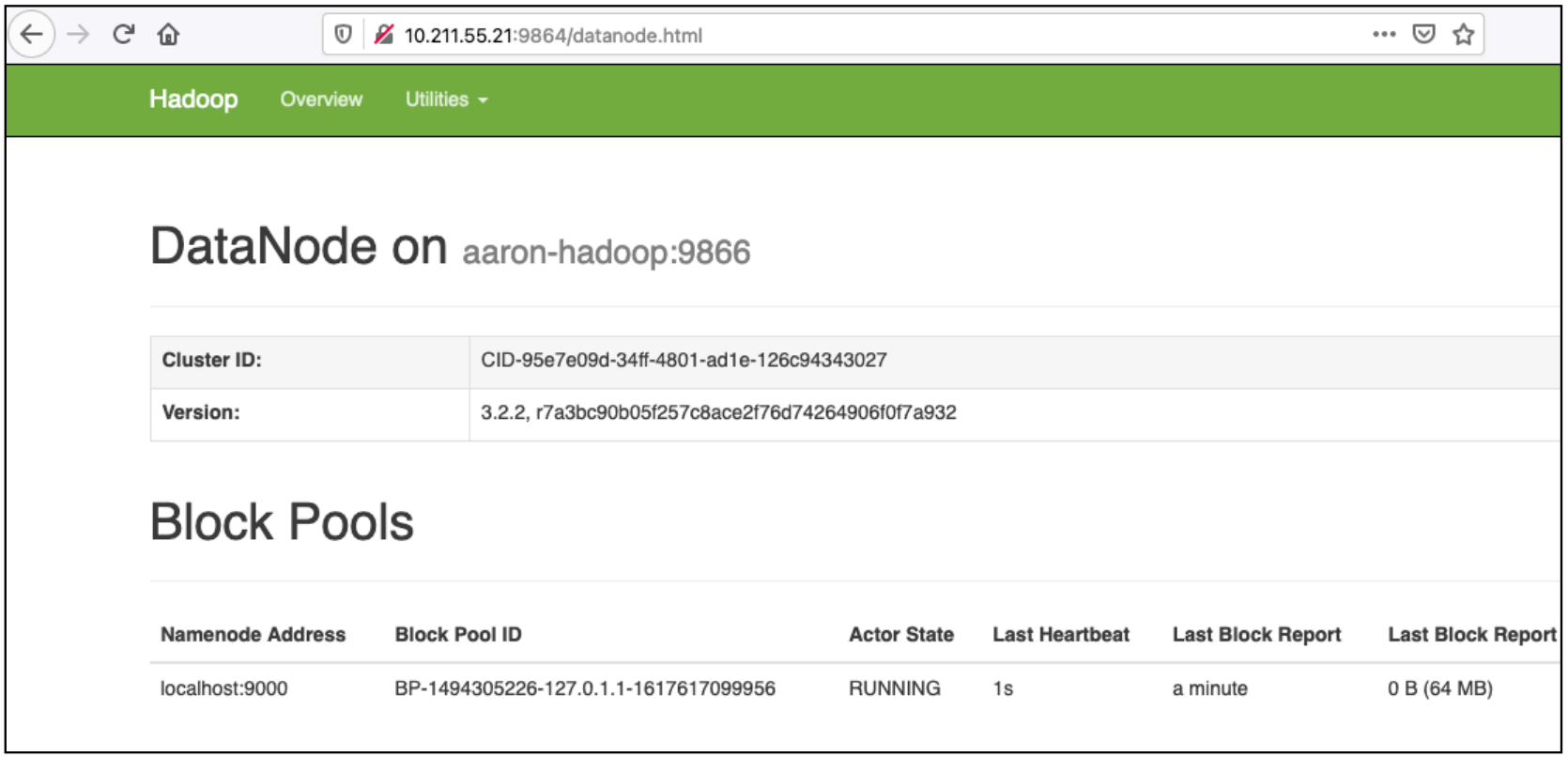
`ip a`

```
hadoop@aaron-hadoop:~/hadoop-3.2.2/sbin$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 00:1c:42:a4:9e:6c brd ff:ff:ff:ff:ff:ff
    inet 10.211.55.21/24 brd 10.211.55.255 scope global dynamic enp0s5
       valid_lft 1080sec preferred_lft 1080sec
    inet6 fdb2:2c26:f4e4:0:21c:42ff:fea4:9e6c/64 scope global dynamic mngtmpaddr
 noprefixroute
       valid_lft 2591946sec preferred_lft 604746sec
    inet6 fe80::21c:42ff:fea4:9e6c/64 scope link
       valid_lft forever preferred_lft forever
```

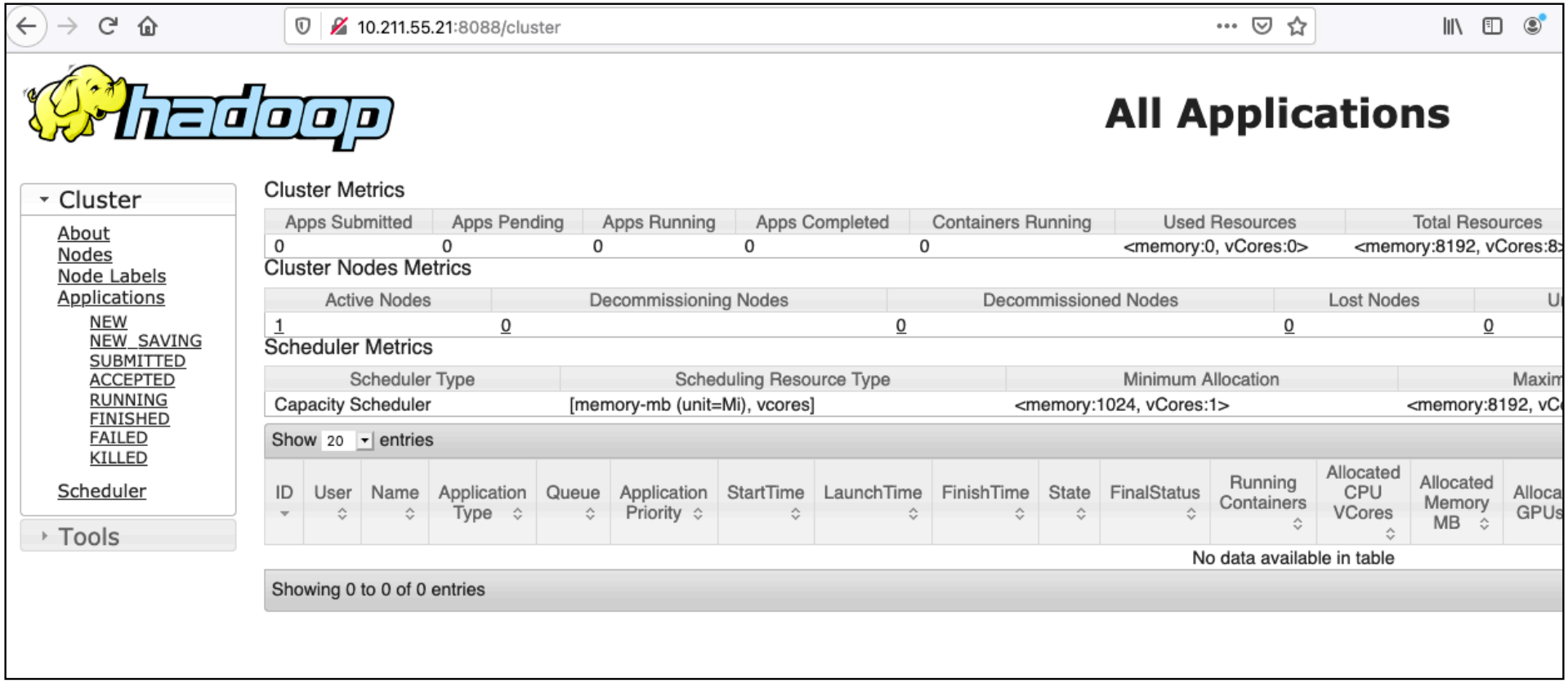
The default port number 9870 gives you access to the Hadoop NameNode UI.



The default port 9864 is used to access individual DataNodes directly from your browser



The YARN Resource Manager is accessible on port 8088



Testing a program on the standalone node

Create a file named WordCount.java in a directory named programs

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new
StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable>
values,
                       Context context
                       ) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Then compile the program

```
hadoop com.sun.tools.javac.Main WordCount.java

jar cf wc.jar WordCount*.class
```

Create two files named file1 and file2. These files contain some text.

```
hadoop@aaron-hadoop:~/programs$ echo "Hello World Bye World" > file1
hadoop@aaron-hadoop:~/programs$ echo "Hello Hadoop Goodbye Hadoop" > file2
```

Then check for the root directory in the hdfs file system

```
hadoop@aaron-hadoop:~/programs$ hdfs dfs -ls /
Found 1 items
drwx------   - hadoop supergroup_          0 2021-04-05 12:07 /tmp
```

Create a directory in the HDFS file system

```
hdfs dfs -mkdir /tmp/input
```

Now copy the two files you created into HDFS

```
hadoop@aaron-hadoop:~/programs$ hdfs dfs -copyFromLocal file1 /tmp/input
hadoop@aaron-hadoop:~/programs$ hdfs dfs -copyFromLocal file2 /tmp/input
```

Now run the application

```
hadoop jar wc.jar WordCount /tmp/input /tmp/output
```

```
hadoop@aaron-hadoop:~/programs$ hadoop jar wc.jar WordCount /tmp/input /tmp/output
2021-04-05 12:34:44,217 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-04-05 12:34:44,520 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface an
d execute your application with ToolRunner to remedy this.
2021-04-05 12:34:44,539 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1
617625112512_0001
2021-04-05 12:34:44,771 INFO input.FileInputFormat: Total input files to process : 2
2021-04-05 12:34:44,842 INFO mapreduce.JobSubmitter: number of splits:2
2021-04-05 12:34:45,080 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1617625112512_0001
2021-04-05 12:34:45,081 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-04-05 12:34:45,255 INFO conf.Configuration: resource-types.xml not found
2021-04-05 12:34:45,255 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-04-05 12:34:45,481 INFO impl.YarnClientImpl: Submitted application application_1617625112512_0001
2021-04-05 12:34:45,539 INFO mapreduce.Job: The url to track the job: http://aaron-hadoop:8088/proxy/application_1617625112512_0001/
2021-04-05 12:34:45,539 INFO mapreduce.Job: Running job: job_1617625112512_0001
2021-04-05 12:34:52,690 INFO mapreduce.Job: Job job_1617625112512_0001 running in uber mode : false
2021-04-05 12:34:52,691 INFO mapreduce.Job:  map 0% reduce 0%
2021-04-05 12:34:58,794 INFO mapreduce.Job:  map 100% reduce 0%
2021-04-05 12:35:03,827 INFO mapreduce.Job:  map 100% reduce 100%
2021-04-05 12:35:03,838 INFO mapreduce.Job: Job job_1617625112512_0001 completed successfully
2021-04-05 12:35:03,911 INFO mapreduce.Job: Counters: 54
```

...

```
        Physical memory (bytes) snapshot=685424640
        Virtual memory (bytes) snapshot=8191078400
        Total committed heap usage (bytes)=488636416
        Peak Map Physical memory (bytes)=261640192
        Peak Map Virtual memory (bytes)=2729811968
        Peak Reduce Physical memory (bytes)=166596608
        Peak Reduce Virtual memory (bytes)=2735349760
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=50
File Output Format Counters
        Bytes Written=41
```

Let us see the output of the program

```
hadoop@aaron-hadoop:~/programs$ hdfs dfs -ls /tmp/output
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2021-04-05 12:35 /tmp/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup         41 2021-04-05 12:35 /tmp/output/part-r-00000
hadoop@aaron-hadoop:~/programs$ hdfs dfs -cat /tmp/output/part-r-00000
Bye     1
Goodbye 1
Hadoop  2
Hello   2
World   2
hadoop@aaron-hadoop:~/programs$
```

THE END