Lab 2 : HDFS File System Administration

Objectives

- Start and validate a mini HDFS + YARN cluster under Docker.
- Practice HDFS administration tasks: users, permissions, ACLs, snapshots, quotas, replication, and integrity.
- Discover basic monitoring through the NameNode and ResourceManager web interfaces.

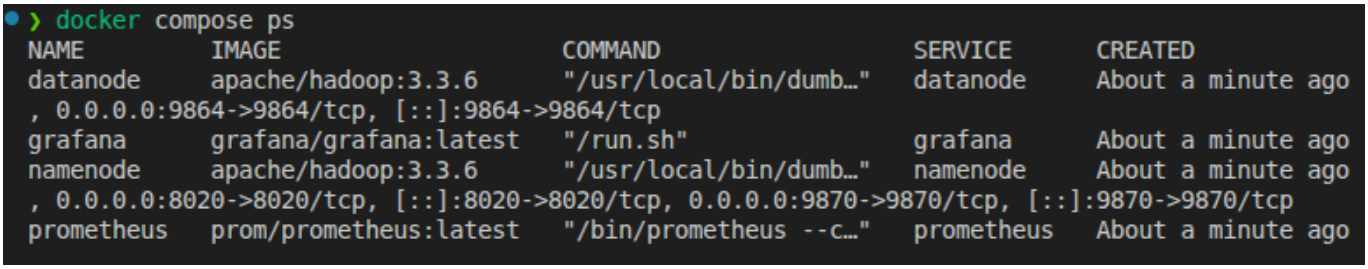
1) Launching the Cluster

From `lab2-hdfs/`:

```
docker compose up -d
docker compose ps
```

Verify that the following services are running: `namenode`, `datanode`, `resourcemanager`, and `nodemanager`.

Snapshots

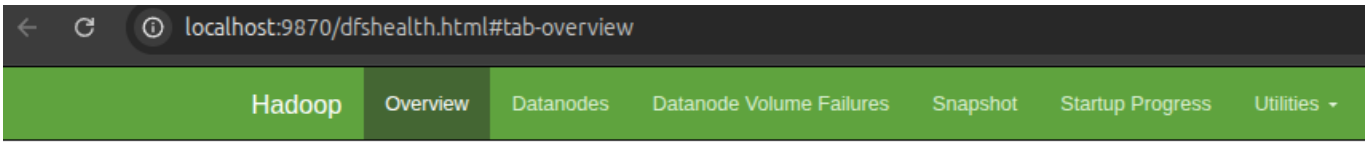


2) Quick Checks via Web UI

- NameNode UI (HDFS): http://localhost:9870
  → Check: number of live DataNodes, capacity, and block status.
- YARN ResourceManager UI: http://localhost:8088
  → Check: NodeManager is “HEALTHY.”

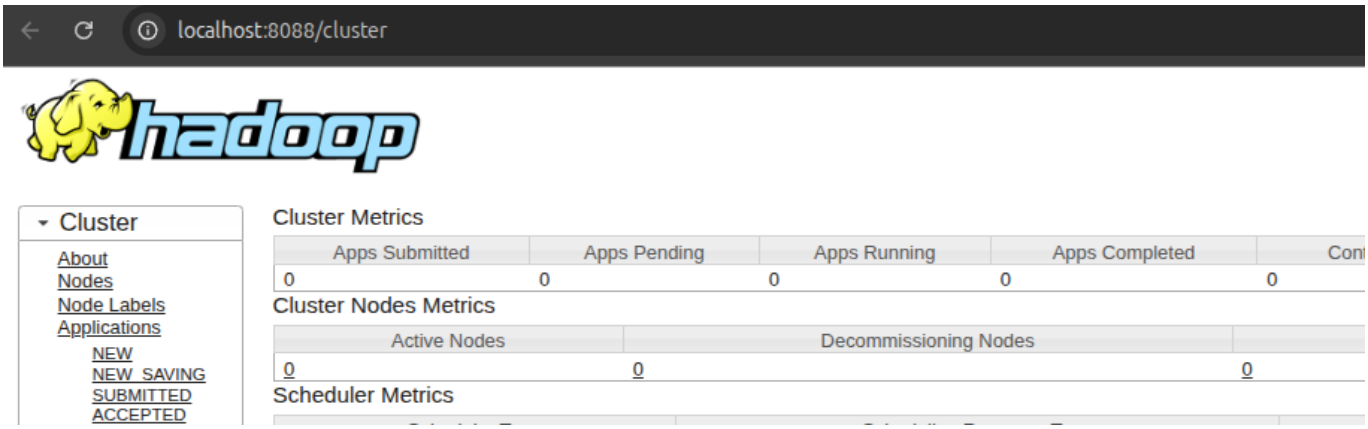
Take screenshots as evidence.

Snapshots



Overview 'namenode:8020' (active)

Started:	Wed Dec 31 16:58:38 +0100 2025
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 09:22:00 +0100 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-1e369f75-9488-4f24-a973-7b0d115deccc
Block Pool ID:	BP-10522430-172.22.0.3-1767196710391



3) Admin Shell in the NameNode

All the following HDFS commands are executed from inside the NameNode container:

```
docker exec -it namenode bash
# Then inside the container:
hdfs dfs -ls /
hdfs dfsadmin -report
```

Snapshots

```
❯ docker exec -it namenode bash
bash-4.2$ hdfs dfs -ls /
bash-4.2$ hdfs dfsadmin -report
Configured Capacity: 322725670912 (300.56 GB)
Present Capacity: 21477736448 (20.00 GB)
DFS Remaining: 21477711872 (20.00 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Replicated Blocks:
        Under replicated blocks: 0
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
```

4) User Tree and Permissions

Create a “cours” directory and simulate two logical users (without OS accounts):

```
hdfs dfs -mkdir -p /cours/tp1
hdfs dfs -mkdir -p /user/student1 /user/student2
hdfs dfs -chmod 755 /user /cours
hdfs dfs -chmod 700 /user/student1 /user/student2
```

Create a small sample dataset:

```
cat > /tmp/sales.csv << 'EOF'
id_client,date,total
101,2025-09-01,120.50
102,2025-09-01,45.00
101,2025-09-02,33.20
103,2025-09-02,250.00
EOF

hdfs dfs -put -f /tmp/sales.csv /cours/tp1/
hdfs dfs -ls -h /cours/tp1
```

Snapshots

```
bash-4.2$ hdfs dfs -mkdir -p /cours/tp1
bash-4.2$ hdfs dfs -mkdir -p /user/student1 /user/student2
bash-4.2$ hdfs dfs -chmod 755 /user /cours
bash-4.2$ hdfs dfs -chmod 700 /user/student1 /user/student2
bash-4.2$
```

```
bash-4.2$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2025-12-31 16:13 /cours
drwxr-xr-x   - hadoop supergroup          0 2025-12-31 16:13 /user
```

```
bash-4.2$ cat > /tmp/sales.csv << 'EOF'
> id_client,date,total
> 101,2025-09-01,120.50
> 102,2025-09-01,45.00
> 101,2025-09-02,33.20
> 103,2025-09-02,250.00
> EOF
bash-4.2$
bash-4.2$ hdfs dfs -put -f /tmp/sales.csv /cours/tp1/
h /cours/tp1bash-4.2$ hdfs dfs -ls -h /cours/tp1
Found 1 items
-rw-r--r--   3 hadoop supergroup        107 2025-12-31 16:17 /cours/tp1/sales.csv
bash-4.2$
```

5) ACLs (Fine-Grained Access Control) on HDFS

Objective: give `student2` read-only access to `/cours/tp1` without changing standard permissions.

```
# Display existing ACLs
hdfs dfs -getfacl /cours/tp1

# Add read (r-x) ACL for “student2”
hdfs dfs -setfacl -m user:student2:r-x /cours/tp1

# Verify
hdfs dfs -getfacl /cours/tp1
```

If “student2” doesn’t exist on the host system, HDFS still stores it symbolically.
For demonstration, focus on how ACLs work and the resulting access behavior.

Snapshots

```
bash-4.2$ hdfs dfs -getfacl /cours/tp1
# file: /cours/tp1
# owner: hadoop
# group: supergroup
user::rwx
group::r-x
other::r-x

bash-4.2$ hdfs dfs -setfacl -m user:student2:r-x /cours/tp1
bash-4.2$ hdfs dfs -setfacl -m user:student2:r-x /cours/tp1
bash-4.2$ hdfs dfs -getfacl /cours/tp1
# file: /cours/tp1
# owner: hadoop
# group: supergroup
user::rwx
user:student2:r-x
group::r-x
mask::r-x
other::r-x

bash-4.2$
```

6) Snapshots (Logical Directory Backup)

Enable snapshots on `/cours` and create one before modification:

```
hdfs dfsadmin -allowSnapshot /cours
SNAP="pre_modif_$(date +%Y%m%d_%H%M%S)"
hdfs dfs -createSnapshot /cours "$SNAP"
hdfs dfs -ls /cours/.snapshot
```

Test accidental deletion and restoration:

```
hdfs dfs -rm /cours/tp1/sales.csv
hdfs dfs -ls /cours/tp1

# Restore from snapshot
hdfs dfs -cp /cours/.snapshot/$SNAP/tp1/sales.csv /cours/tp1/
hdfs dfs -ls /cours/tp1
```

Snapshots

```
bash-4.2$ hdfs dfsadmin -allowSnapshot /cours
f_$(date +%Y%m%d_%H%M%S)"
hdfs dfs -createSnapshot /cours "$SNAP"
hdfs dfs -ls /cours/.snapshotAllowing snapshot on /cours succeeded
bash-4.2$ SNAP="pre_modif_$(date +%Y%m%d_%H%M%S)"
bash-4.2$ hdfs dfs -createSnapshot /cours "$SNAP"
Created snapshot /cours/.snapshot/pre_modif_20251231_165247
bash-4.2$ hdfs dfs -ls /cours/.snapshot
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2025-12-31 16:52 /cours/.snapshot/pre_modif_20251231_165247
bash-4.2$ hdfs dfs -rm /cours/tp1/sales.csv
urs/tp1Deleted /cours/tp1/sales.csv
bash-4.2$ hdfs dfs -ls /cours/tp1
bash-4.2$ hdfs dfs -cp /cours/.snapshot/$SNAP/tp1/sales.csv /cours/tp1/
bash-4.2$ hdfs dfs -ls /cours/tp1
Found 1 items
-rw-r--r--   3 hadoop supergroup        107 2025-12-31 16:54 /cours/tp1/sales.csv
bash-4.2$
```

7) Quotas (Space & File Limits)

Set a quota on `/user/student1`:

```
hdfs dfsadmin -setSpaceQuota 1m /user/student1
hdfs dfsadmin -setQuota 100 /user/student1
hdfs dfs -count -q -h /user/student1
```

Test by copying a slightly large file:

```
dd if=/dev/zero of=/tmp/bigfile.bin bs=64K count=40 # ~2.5 MB
hdfs dfs -put /tmp/bigfile.bin /user/student1/ || echo "Expected failure
(quota exceeded)"
```

Reset if needed:

```
hdfs dfsadmin -clrSpaceQuota /user/student1
hdfs dfsadmin -clrQuota /user/student1
```

Snapshots

```
bash-4.2$ hdfs dfsadmin -setSpaceQuota 1m /user/student1
bash-4.2$ hdfs dfsadmin -setQuota 100 /user/student1
bash-4.2$ hdfs dfs -count -q -h /user/student1
        100             99            1 M            1 M            1            0                0 /user/student1
bash-4.2$ dd if=/dev/zero of=/tmp/bigfile.bin bs=64K count=40
40+0 records in
40+0 records out
2621440 bytes (2.6 MB) copied, 0.00279483 s, 938 MB/s
bash-4.2$ hdfs dfs -put /tmp/bigfile.bin /user/student1/ || echo "Expected failure (quota exceeded)"
put: The DiskSpace quota of /user/student1 is exceeded: quota = 1048576 B = 1 MB but diskspace consumed = 402653184 B = 384 MB
Expected failure (quota exceeded)
bash-4.2$ hdfs dfsadmin -clrSpaceQuota /user/student1
bash-4.2$ hdfs dfsadmin -clrQuota /user/student1
bash-4.2$ hdfs dfs -put /tmp/bigfile.bin /user/student1/
```

8) Replication Factor & Data Integrity

Check replication settings:

```
# Default replication factor
hdfs getconf -confKey dfs.replication || true

# For a specific file
hdfs dfs -stat %r /cours/tp1/sales.csv
```

Change the replication factor (e.g., from 1 → 2) and optionally run the balancer:

```
hdfs dfs -setrep 2 /cours/tp1/sales.csv
hdfs dfs -stat %r /cours/tp1/sales.csv
hdfs balancer -threshold 10
```

Check integrity:

```
hdfs fsck / -files -blocks -locations | head -n 50
```

Snapshots

```
bash-4.2$ hdfs getconf -confKey dfs.replication || true
3
bash-4.2$ hdfs dfs -stat %r /cours/tp1/sales.csv
3
bash-4.2$ hdfs dfs -setrep 2 /cours/tp1/sales.csv
Replication 2 set: /cours/tp1/sales.csv
bash-4.2$ hdfs dfs -stat %r /cours/tp1/sales.csv
2
```

```
bash-4.2$ hdfs balancer -threshold 10
2025-12-31 17:00:19 INFO  Balancer:938 - Using a threshold of 10.0
2025-12-31 17:00:19 INFO  Balancer:738 - namenodes  = [hdfs://namenode:8020]
2025-12-31 17:00:19 INFO  Balancer:739 - parameters = Balancer.BalancerParameters [BalancingPolicy.Node, thre
ed nodes = 0, #included nodes = 0, #source nodes = 0, #blockpools = 0, run during upgrade = false]
2025-12-31 17:00:19 INFO  Balancer:740 - included nodes = []
2025-12-31 17:00:19 INFO  Balancer:741 - excluded nodes = []
2025-12-31 17:00:19 INFO  Balancer:742 - source nodes = []
Time Stamp               Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Moved  NameNode
2025-12-31 17:00:19 INFO  NameNodeConnector:185 - getBlocks calls for hdfs://namenode:8020 will be rate-limit
2025-12-31 17:00:22 INFO  Balancer:264 - dfs.namenode.get-blocks.max-qps = 20 (default=20)
2025-12-31 17:00:22 INFO  Balancer:246 - dfs.balancer.movedWinWidth = 5400000 (default=5400000)
2025-12-31 17:00:22 INFO  Balancer:264 - dfs.balancer.moverThreads = 1000 (default=1000)
2025-12-31 17:00:22 INFO  Balancer:264 - dfs.balancer.dispatcherThreads = 200 (default=200)
2025-12-31 17:00:22 INFO  Balancer:255 - dfs.balancer.getBlocks.size = 2147483648 (default=2147483648)
2025-12-31 17:00:22 INFO  Balancer:255 - dfs.balancer.getBlocks.min-block-size = 10485760 (default=10485760)
2025-12-31 17:00:22 INFO  Balancer:264 - dfs.datanode.balance.max.concurrent.moves = 100 (default=100)
2025-12-31 17:00:22 INFO  Balancer:255 - dfs.datanode.balance.bandwidthPerSec = 104857600 (default=104857600)
2025-12-31 17:00:22 INFO  Balancer:255 - dfs.balancer.max-size-to-move = 10737418240 (default=10737418240)
2025-12-31 17:00:22 INFO  Balancer:255 - dfs.blocksize = 134217728 (default=134217728)
2025-12-31 17:00:22 INFO  NetworkTopology:156 - Adding a new node: /default-rack/172.22.0.5:9866
2025-12-31 17:00:22 INFO  Balancer:469 - 0 over-utilized: []
2025-12-31 17:00:22 INFO  Balancer:469 - 0 underutilized: []
Dec 31, 2025 5:00:22 PM          0                 0 B                 0 B                 0 B
The cluster is balanced. Exiting...
Dec 31, 2025 5:00:22 PM  Balancing took 3.707 seconds
```

```
bash-4.2$ hdfs fsck / -files -blocks -locations | head -n 50
Connecting to namenode via http://namenode:9870/fsck?ugi=hadoop&files=1&blocks=1&locations=1&path=%2F
FSCK started by hadoop (auth:SIMPLE) from /172.22.0.3 for path / at Wed Dec 31 17:00:40 UTC 2025

/ <dir>
/cours <dir>
/cours/tp1 <dir>
/cours/tp1/sales.csv 107 bytes, replicated: replication=2, 1 block(s):  Under replicated BP-10522430-172
Replicas is 2 but found 1 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
0. BP-10522430-172.22.0.3-1767196710391:blk_1073741826_1002 len=107 Live_repl=1  [DatanodeInfoWithStorag
5038310,DISK]]
```

9) Safemode and Admin Operations

Show the “safemode” (read-only NameNode state):

```
hdfs dfsadmin -safemode get
```

Snapshots

```
bash-4.2$ hdfs dfsadmin -safemode get
Safe mode is OFF
bash-4.2$
```

10) Cluster Monitoring with Prometheus and Grafana

a monitoring system for Hadoop components (NameNode & DataNodes) using Prometheus and Grafana, to visualize HDFS metrics: capacity, blocks, files, and JVM health.

10.1 Monitoring Architecture

- JMX Exporter on each Hadoop service (NameNode, DataNode) exposes JVM and HDFS metrics.
- Prometheus collects the metrics.
- Grafana visualizes them as dashboards and charts.

```
NameNode (JMX) ─┐
                ├──► Prometheus ──► Grafana
DataNode (JMX) ─┘
```

10.2 JMX Configuration for HDFS

(File: `monitoring/jmx/hadoop.yml`)
Contains metric mapping rules for HDFS and JVM.

Snapshots

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
  - pattern: 'Hadoop<service=NameNode, name=FSNamesystem><>CapacityTotal'
    name: hadoop_namenode_fs_namesystem_capacitytotal
    type: GAUGE
  - pattern: 'Hadoop<service=NameNode, name=FSNamesystem><>CapacityUsed'
    name: hadoop_namenode_fs_namesystem_capacityused
    type: GAUGE
  - pattern: 'Hadoop<service=NameNode, name=FSNamesystem><>CapacityRemaining'
    name: hadoop_namenode_fs_namesystem_capacityremaining
    type: GAUGE
  - pattern: 'Hadoop<service=NameNode, name=FSNamesystem><>BlocksTotal'
    name: hadoop_namenode_fs_namesystem_blockstotal
    type: GAUGE
  - pattern: 'Hadoop<service=NameNode, name=FSNamesystem><>FilesTotal'
    name: hadoop_namenode_fs_namesystem_filestotal
    type: GAUGE
  - pattern: 'java.lang<type=Threading><>(.+)'
    name: jvm_threads_$1
    type: GAUGE
  - pattern: 'java.lang<type=Memory><HeapMemoryUsage>(\w+)'
    name: jvm_memory_bytes_$1
    type: GAUGE
  - pattern: '.*'
    name: jmx_misc
    type: GAUGE
```

10.3 Prometheus & Grafana Integration

Launch monitoring:

```
docker compose up -d
```

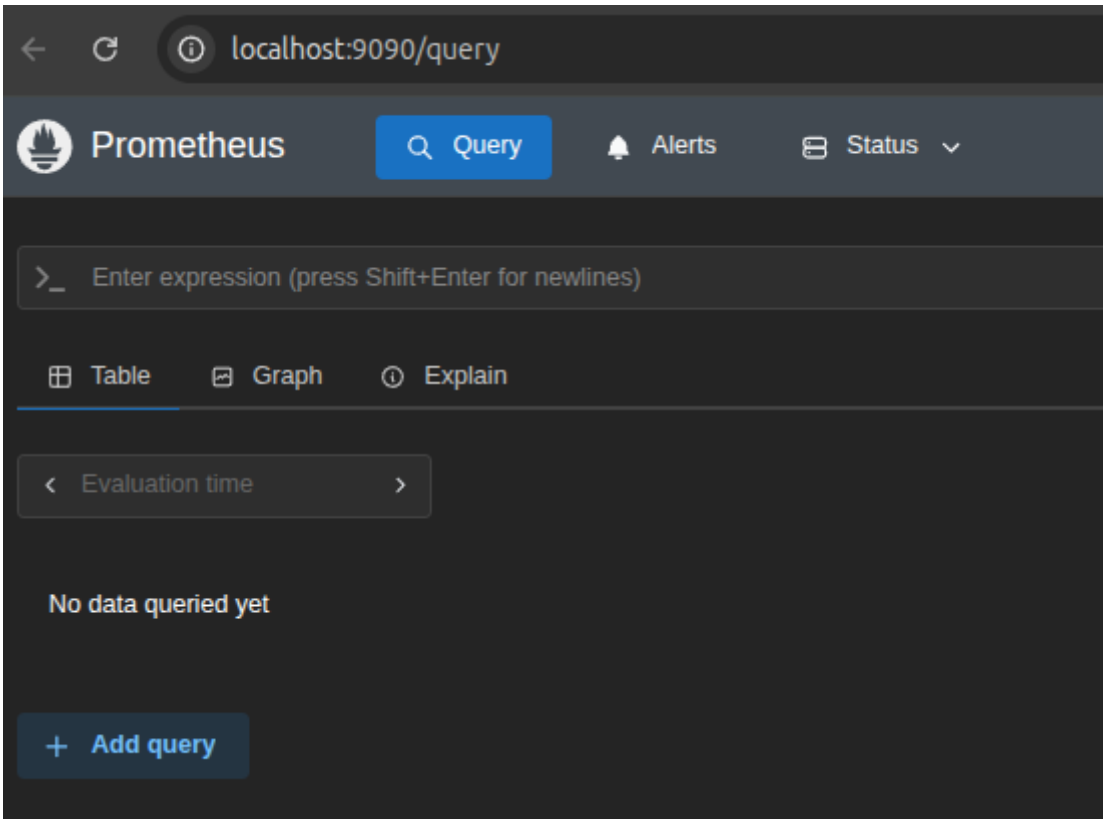
```
❯ docker compose ps
NAME              IMAGE                      COMMAND                  SERVICE
datanode          apache/hadoop:3.3.6        "/usr/local/bin/dumb…"   datanode
7001->7001/tcp, 0.0.0.0:9864->9864/tcp, [::]:9864->9864/tcp
grafana           grafana/grafana:latest     "/run.sh"                grafana
3000->3000/tcp
namenode          apache/hadoop:3.3.6        "/usr/local/bin/dumb…"   namenode
7000->7000/tcp, 0.0.0.0:8020->8020/tcp, [::]:8020->8020/tcp, 0.0.0.0:9870->9870/tcp,
nodemanager       apache/hadoop:3.3.6        "/usr/local/bin/dumb…"   nodemanager
8042->8042/tcp
prometheus        prom/prometheus:latest     "/bin/prometheus --c…"   prometheus
9090->9090/tcp
resourcemanager   apache/hadoop:3.3.6        "/usr/local/bin/dumb…"   resourcemanager
8032->8032/tcp, 0.0.0.0:8088->8088/tcp, [::]:8088->8088/tcp
```

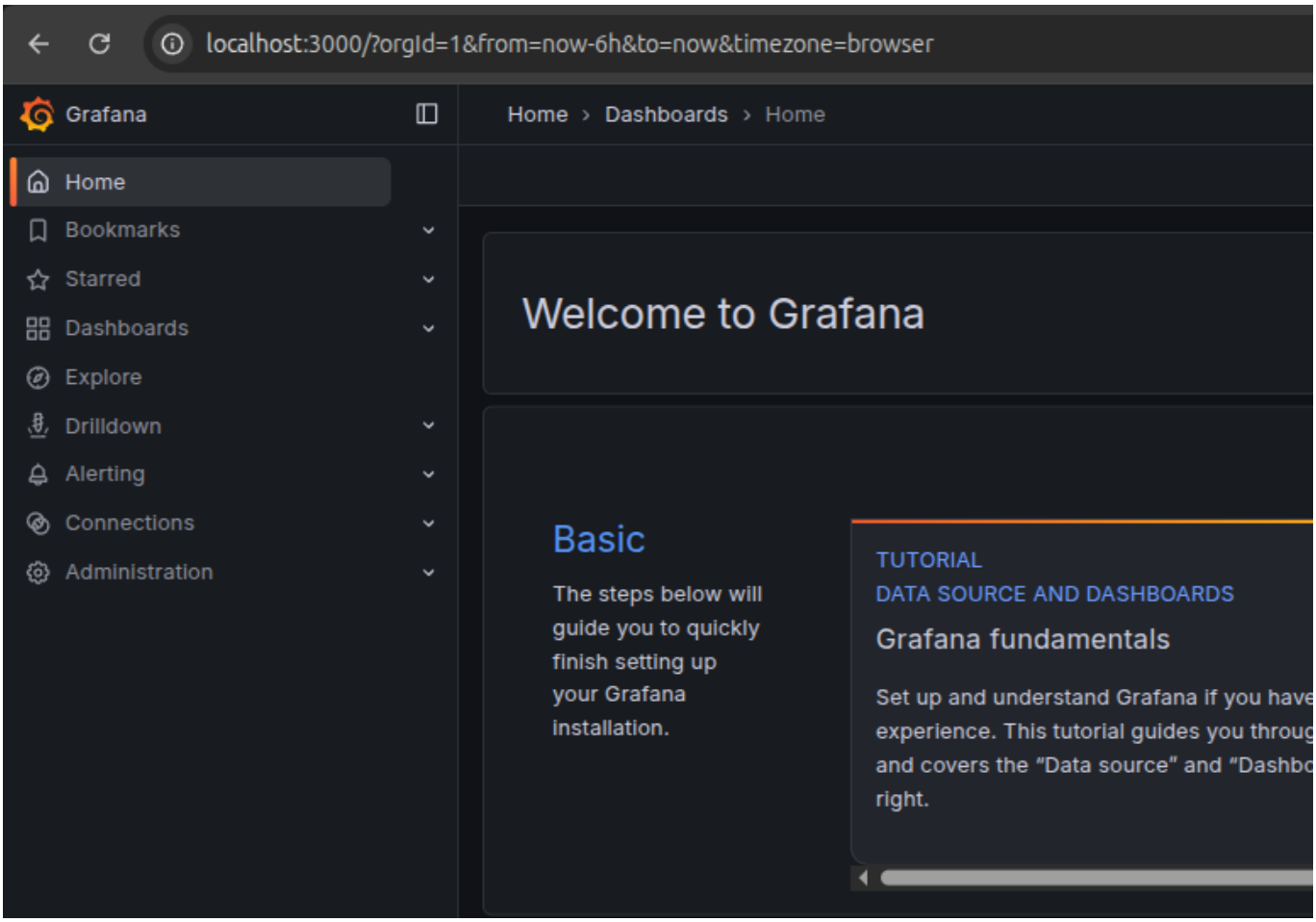
Access:

- Prometheus → http://localhost:9090

- Grafana → http://localhost:3000 (`admin`/`admin`)

Snapshots



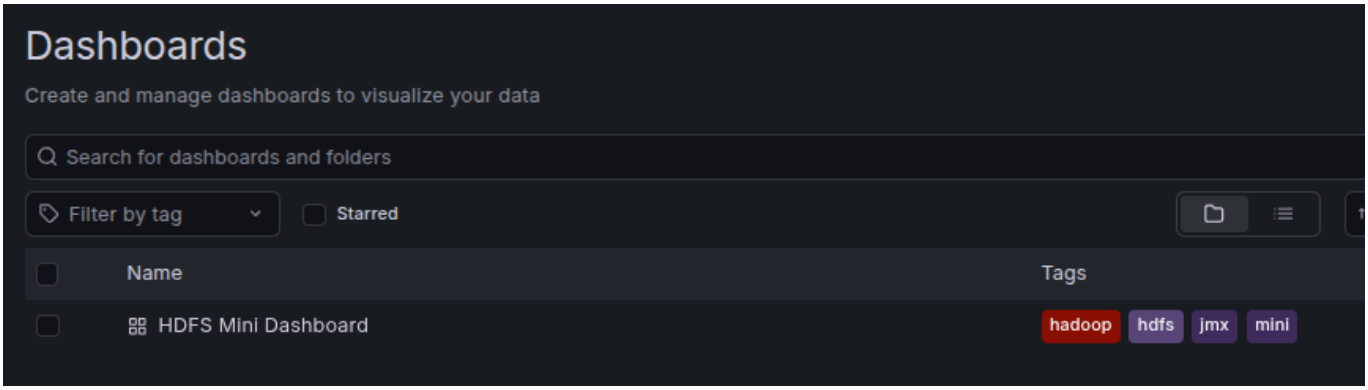


10.4 Grafana Dashboard

Import the dashboard JSON file `monitoring/jmx/hdfs_dashboard.json` (Grafana → Dashboards → Import) and visualize:

- HDFS Capacity Used
- HDFS Capacity Remaining
- HDFS Blocks Total
- HDFS Files Total
- JVM Heap Used
- JVM Threads Current

Snapshots

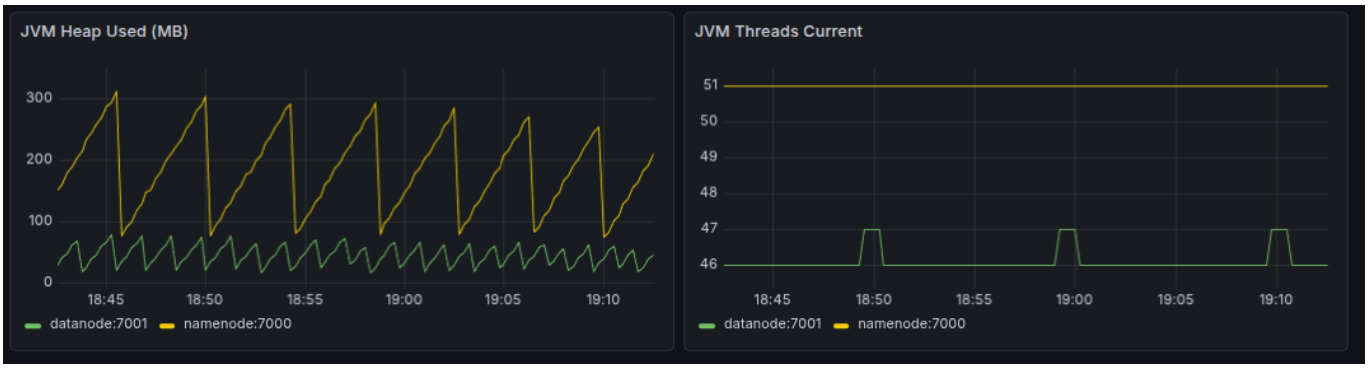


10.5 Experimentation

```
hdfs dfs -mkdir /test
hdfs dfs -put /etc/passwd /test/
hdfs dfs -put /etc/hosts /test/
```

Then refresh Grafana to see metrics evolve.

Snapshots



11) Prometheus Alert System Integration

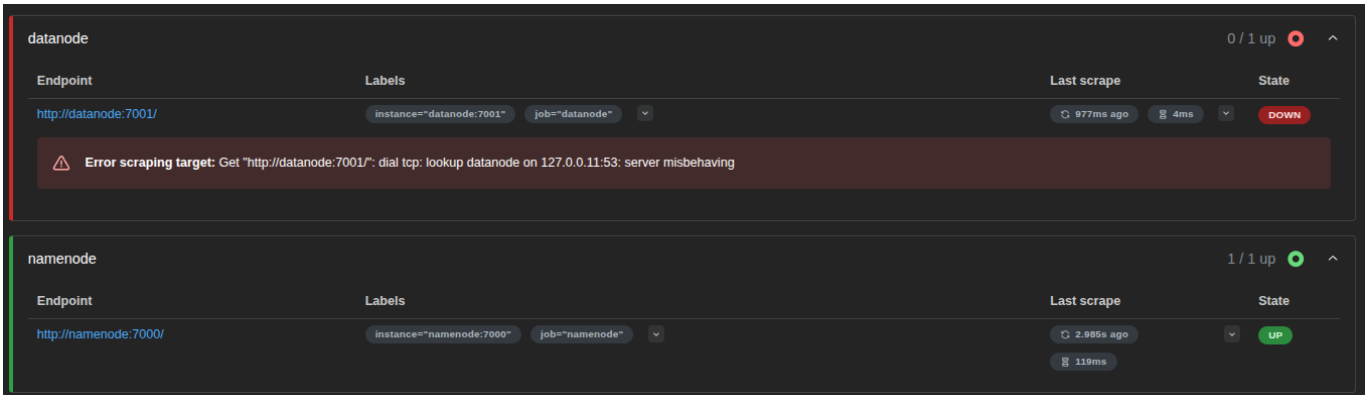
automatic alerts for real-time cluster issues: DataNode down, NameNode down, or low disk space.

Where to see alerts:

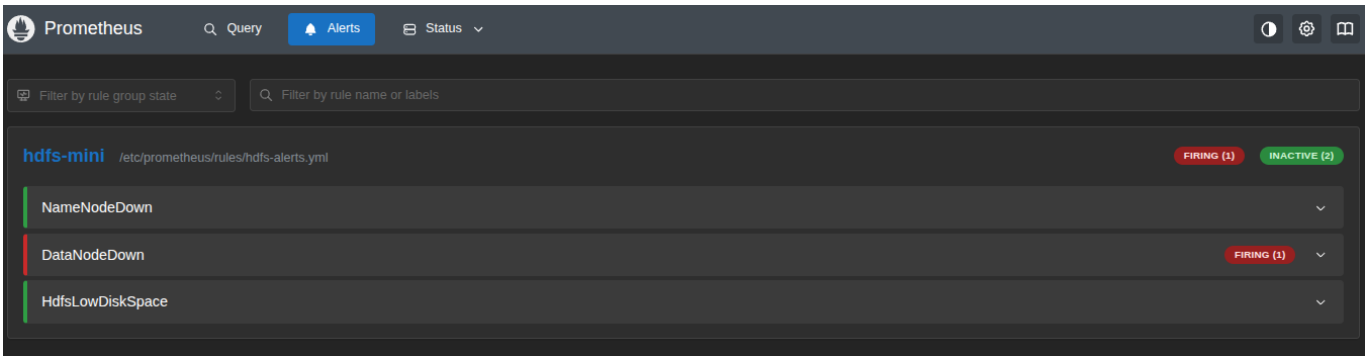
- Prometheus alerts page → http://localhost:9090/alerts
- Alertmanager UI → http://localhost:9093

Simulate a failure:

```
docker stop datanode
```



→ After 30 seconds, the `DataNodeDown` alert appears.



Restart:

```
docker start datanode
```

→ The alert automatically resolves.