# Anatomy of a Write Operation

This lesson explains how HDFS handles a write operation.

**We'll cover the following**



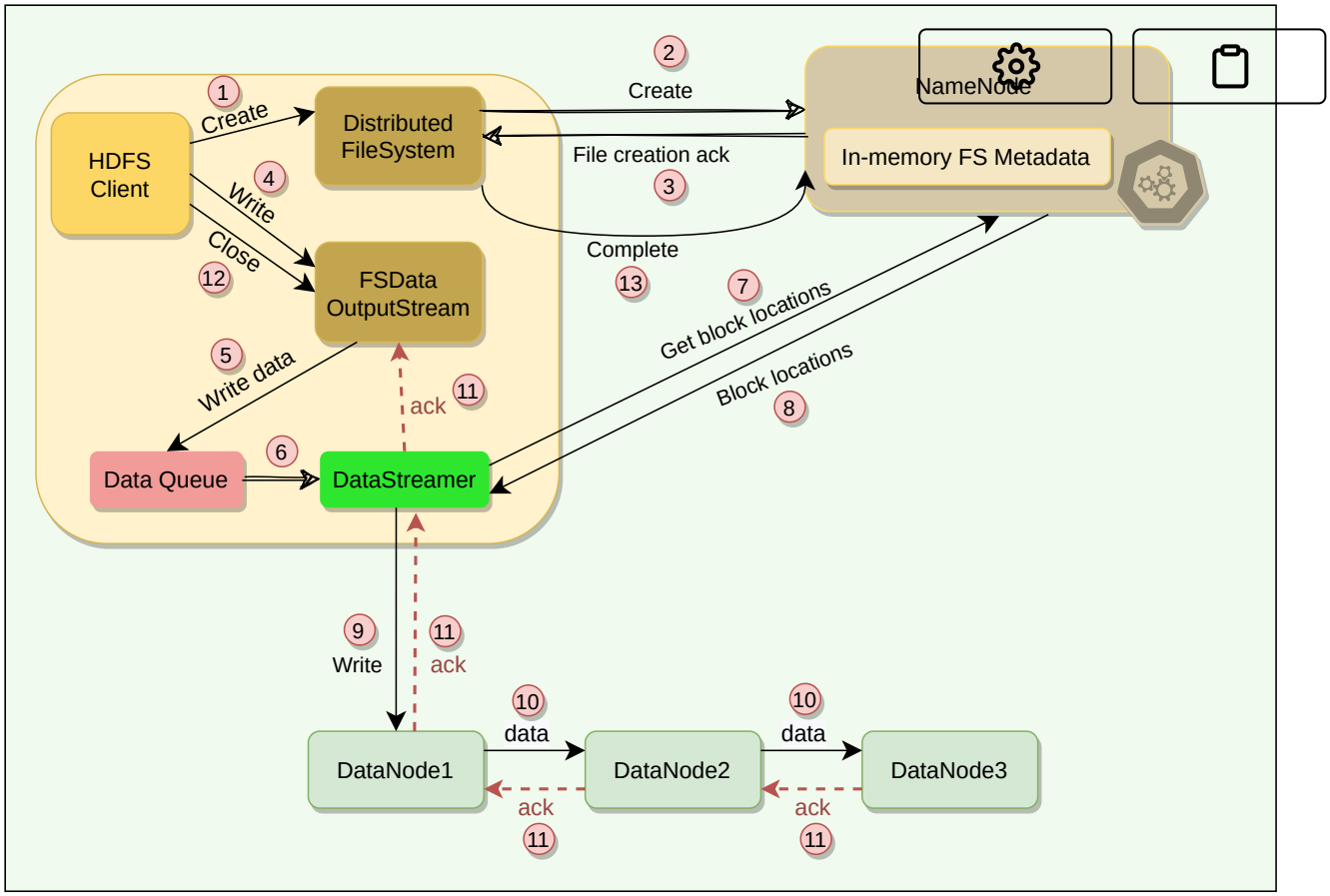
# HDFS write process#

HDFS write process can be outlined as follows:

1. HDFS client initiates a write request by calling the `create()` method of the `Distributed FileSystem` object.
2. The `Distributed FileSystem` object sends a file creation request to the NameNode.
3. The NameNode verifies that the file does not already exist and that the client has permission to create the file. If both these conditions are verified, the NameNode creates a new file record and sends an acknowledgment.
4. The client then proceeds to write the file using `FSData OutputStream`
5. The `FSData OutputStream` writes data to a local queue called 'Data Queue.' The data is kept in the queue until a complete block of data is accumulated.
6. Once the queue has a complete block, another component called `DataStreamer` is notified to manage data transfer to the DataNode.
7. `DataStreamer` first asks the NameNode to allocate a new block on DataNodes, thereby picking desirable DataNodes to be used for

replication.

8. The NameNode provides a list of blocks and the locations of each block replica.
9. Upon receiving the block locations from the NameNode, the `DataStreamer` starts transferring the blocks from the internal queue to the nearest DataNode.
10. Each block is written to the first DataNode, which then pipelines the block to other DataNodes in order to write replicas of the block. This way, the blocks are replicated during the file write itself. It is important to note that HDFS does not acknowledge a write to the client until all the replicas for that block have been written by the DataNodes.
11. Once the `DataStreamer` finishes writing all blocks, it waits for acknowledgments from all the DataNodes.
12. Once all acknowledgments are received, the client calls the `close()` method of the `OutputStream`.
13. Finally, the `Distributed FileSystem` contacts the NameNode to notify that the file write operation is complete. At this point, the NameNode commits the file creation operation, which makes the file available to be read. If the NameNode dies before this step, the file is lost.




← Back

Anatomy of a Read Operation

Next →

Data Integrity & Caching

Mark as Completed

Report an Issue