Assignment 4 - Introduction to Apache Zookeeper

In this lab, you will learn how to:

- Install and Configure Apache Zookeeper
- Use the Zookeeper CLI to interact with the Zookeeper Ensemble

What is Zookeeper?

Zookeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. The implementation of distributed applications is heavy on fixing bugs, race conditions, handling updates, change management, etc. The Zookeeper service helps manage all of this and more with the various APIs and services it implements.

More on Apache Zookeeper

Deliverables:

1a.jpg	Screenshot of Zookeeper Server having started.
2a.jpg	Screenshot of Listing of the created ZNodes
2b.jpg	Screenshot of Listing of the deleted ephemeral node
2c.jpg	Screenshot of Data and Metadata associated with the created sequential znode.
2d.jpg	Screenshot of Listing of all the children nodes
3a.jpg	Screenshot of each of the clients listing all the znodes
3b.jpg	Screenshot of the Watched Event and the Zookeeper Server logs.

Prerequisites:

1. Linux OS
2. Java (JDK 6 or greater)
   To verify installation/ install, refer this document.

TASK A: Installing Zookeeper

1. Download the Zookeeper framework onto your system from here.

2. Extract the tar file using the following command -
```
tar -zxf apache-zookeeper-3.8.0-bin.tar.gz
```

3. Move the extracted file to /opt directory.
```
sudo mv /path/to/download/apache-zookeeper-3.8.0-bin /opt
cd /opt/apache-zookeeper-3.8.0-bin
```

4. Create a Data Directory

```
mkdir data
```

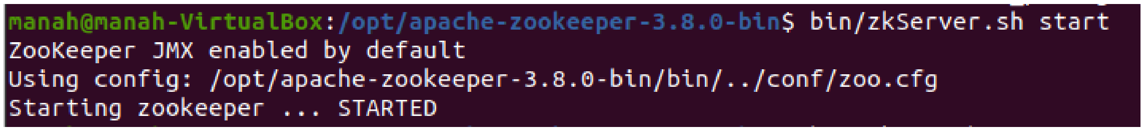
5. Create a Configuration file (conf/zoo.cfg) for Zookeeper with the following parameters -

```
tickTime = 2000
dataDir = /path/to/zookeeper/data
clientPort = 2181
initLimit = 5
syncLimit = 2
```

6. Start the Zookeeper Server

```
bin/zkServer.sh start
```

The output should resemble -

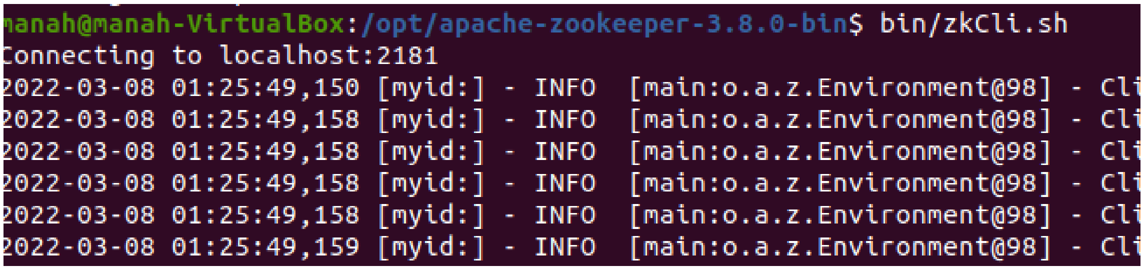


```
manah@manah-VirtualBox:/opt/apache-zookeeper-3.8.0-bin$ bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /opt/apache-zookeeper-3.8.0-bin/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

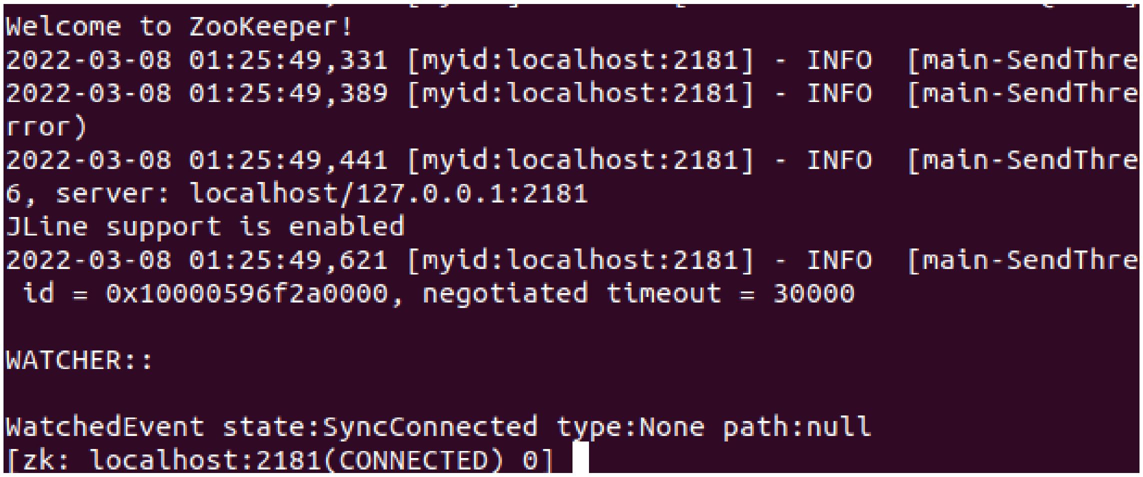
7. Start the Zookeeper CLI

```
bin/zkCli.sh
```

The output should resemble -



```
anah@manah-VirtualBox:/opt/apache-zookeeper-3.8.0-bin$ bin/zkCli.sh
Connecting to localhost:2181
2022-03-08 01:25:49,150 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
2022-03-08 01:25:49,158 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
2022-03-08 01:25:49,158 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
2022-03-08 01:25:49,158 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
2022-03-08 01:25:49,158 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
2022-03-08 01:25:49,159 [myid:] - INFO  [main:o.a.z.Environment@98] - Cli
```



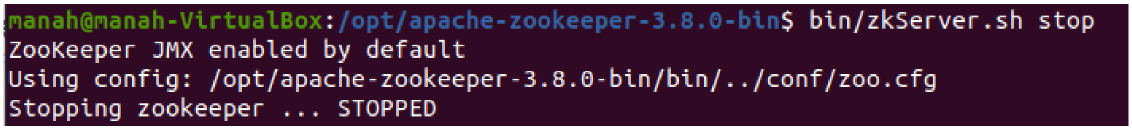
```
Welcome to ZooKeeper!
2022-03-08 01:25:49,331 [myid:localhost:2181] - INFO  [main-SendThre
2022-03-08 01:25:49,389 [myid:localhost:2181] - INFO  [main-SendThre
rror)
2022-03-08 01:25:49,441 [myid:localhost:2181] - INFO  [main-SendThre
6, server: localhost/127.0.0.1:2181
JLine support is enabled
2022-03-08 01:25:49,621 [myid:localhost:2181] - INFO  [main-SendThre
 id = 0x10000596f2a0000, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0]
```

Use `quit` command to exit from the ZK CLI.

8. Stop the Zookeeper Server

```
bin/zkServer.sh stop
```

The output should resemble -



```
manah@manah-VirtualBox:/opt/apache-zookeeper-3.8.0-bin$ bin/zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /opt/apache-zookeeper-3.8.0-bin/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

TASK B: Getting Familiar with Zookeeper CLI

Start the Zookeeper Server and the Zookeeper CLI.

1. Create Znodes

   Usage: `create [FLAGS] /path /data`

   The flags are used to specify the type of znode.

   - Ephemeral znodes (flag e) will be automatically deleted when a session expires, or a client disconnects.
   - Sequential znodes (flag s) are those nodes that guarantee that the path will be unique. Each sequential znode that gets created is suffixed a sequential counter with 10 digit padding to do so. These nodes are persistent.

   a. Create a sequential node and name it with your_srn_seq

      `create -s /your_srn_seq “a sequential node”`

   b. Create an ephemeral node and name it with your_srn_eph

      `create -e /your_srn_eph “an ephemeral node”`

   c. List the nodes

      `ls /`

      Notice the sequence appended to the sequential node. Take a screenshot.

   d. Now, `quit` the Zookeeper Client and start it up again after an approximate of 30s.

      List all the nodes.

      You will observe that the ephemeral node does not exist anymore while the sequential node does.

      Take a screenshot.

2. Set and Get Data

   Usage: `set /path /data`

   Affix the string “updated” to the data of the sequential node created.

   Get the data and metadata associated with the specified znode.

   Usage: `get -s /path`

   Get the information related to the sequential node, and take a screenshot.

   `get -s /your_srn_seq<some_sequence>`

   One may use `get -s -w /path` to add watch, which you will do in the next section.

3. Create Children/ Sub-Znode

   Creating children is similar to creating new znodes. The only difference is that the path of the child znode will have the parent path as well.

   Usage: `create /parent/path/subnode/path /data`

   ```
   create /parent
   create /parent/child "child node"
   ```

   Let the parent be your_srn and child be your_name

   Create two children, one with your first name and one with your last name.

   List all children. Take a screenshot.

4. Delete Znode

   Usage: `delete /path`

   OR

   Usage: `rmr /path`

TASK B: Naive Leader Election with Zookeeper CLI - Understanding the Algorithm

Leader Election Algorithm in Zookeeper:

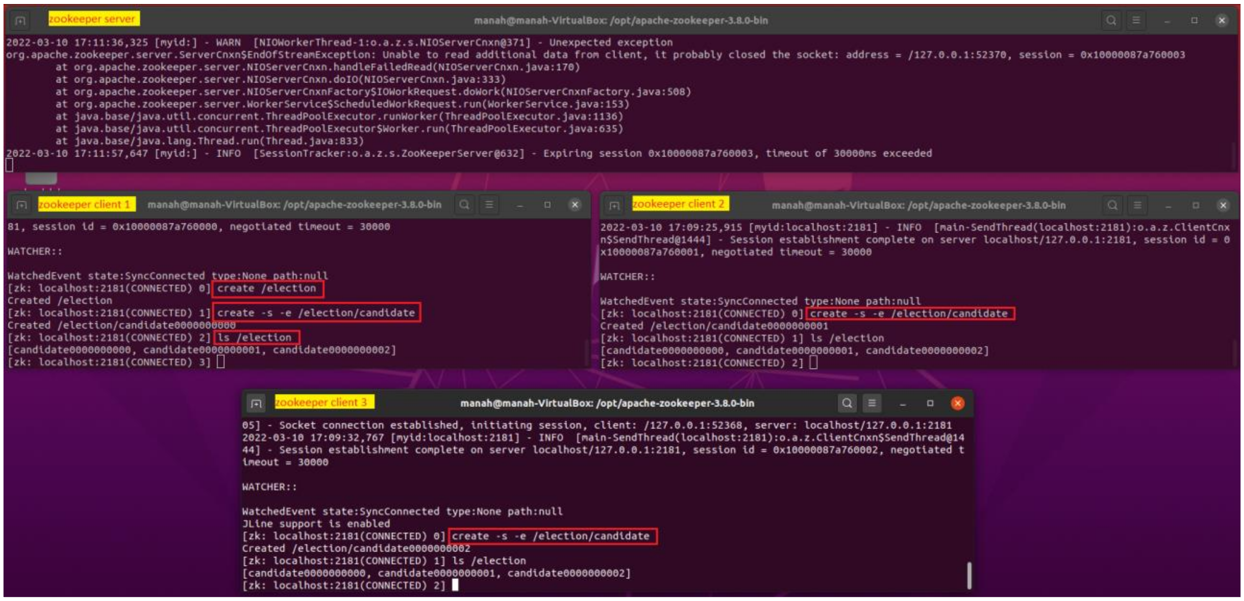
Every sequential ephemeral znode is associated with a sequence as observed earlier. The znode with the smallest sequence is the leader, and all other znodes are followers. Each follower sets a 'watch' on the znode sequentially before it. If the leader goes down, the follower that is sequentially the next gets notified of the event because of the watch. It checks if there is a znode with a sequentially smaller number, and if not, elects itself as leader.

In the following task, you will explore how 'watches' work.

a) Set-Up the Server and clients with znodes.

1. Spin up 4 terminals. One of these will run the zookeeper server while the other three will function as zookeeper clients.
   a. Start up the zookeeper server on one of the terminals. Run it in the foreground.
      `bin/zkServer.sh start-foreground`
   b. Start up the zookeeper cli on the other terminals. These will function as clients.
      `bin/zkCli.sh`
2. On any one of the clients, under the root, create a znode and name it with `your_srn`. This will function as the root znode for this task. Do not specify any type.
3. On each of the clients, create a sequential, ephemeral znode under the root znode created earlier. I.e., under /your_srn. You may name these znodes 'candidate'.
4. List all the nodes under the root znode (/your_srn) on each of the clients. Take a screenshot.

Every client is now running a znode. For an example, refer the screenshot below:



```
2022-03-10 17:11:36,325 [myid:] - WARN  [NIOWorkerThread-1:o.a.z.s.NIOServerCnxn@371] - Unexpected exception
org.apache.zookeeper.server.ServerCnxn$EndOfStreamException: Unable to read additional data from client, it probably closed the socket: address = /127.0.0.1:52370, session = 0x10000087a760003
        at org.apache.zookeeper.server.NIOServerCnxn.handleFailedRead(NIOServerCnxn.java:170)
        at org.apache.zookeeper.server.NIOServerCnxn.doIO(NIOServerCnxn.java:333)
        at org.apache.zookeeper.server.NIOServerCnxnFactory$IOWorkRequest.doWork(NIOServerCnxnFactory.java:508)
        at org.apache.zookeeper.server.WorkerService$ScheduledWorkRequest.run(WorkerService.java:153)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
        at java.base/java.lang.Thread.run(Thread.java:833)
2022-03-10 17:11:57,647 [myid:] - INFO  [SessionTracker:o.a.z.s.ZooKeeperServer@632] - Expiring session 0x10000087a760003, timeout of 30000ms exceeded
```

```
81, session id = 0x10000087a760000, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0] create /election
Created /election
[zk: localhost:2181(CONNECTED) 1] create -s -e /election/candidate
Created /election/candidate0000000000
[zk: localhost:2181(CONNECTED) 2] ls /election
[candidate0000000000, candidate0000000001, candidate0000000002]
[zk: localhost:2181(CONNECTED) 3]
```

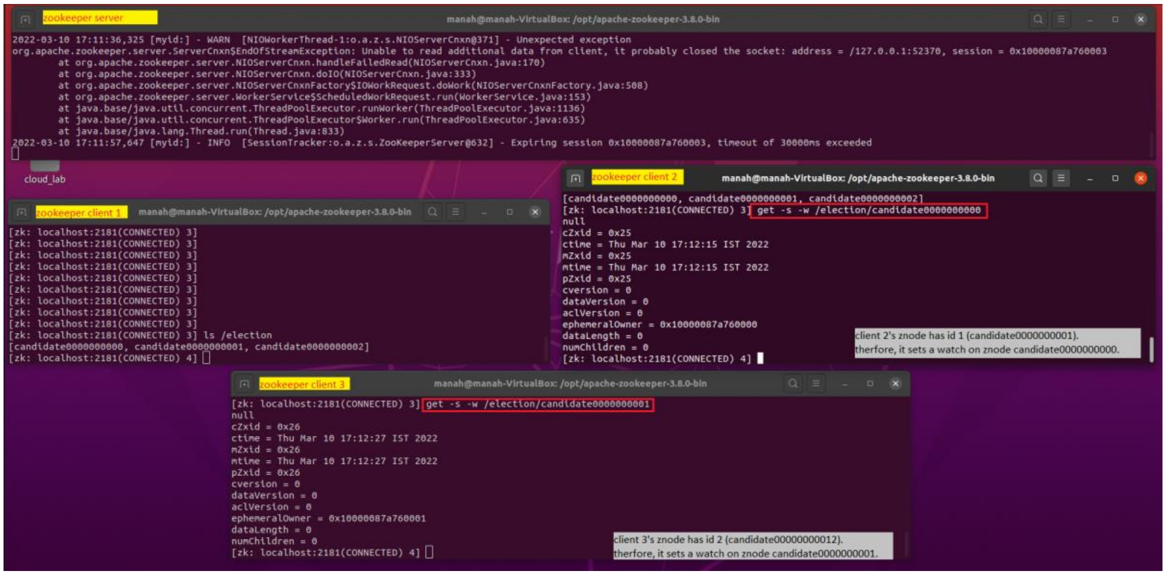
```
2022-03-10 17:09:25,915 [myid:localhost:2181] - INFO  [main-SendThread(localhost:2181):o.a.z.ClientCnx
n$SendThread@1444] - Session establishment complete on server localhost/127.0.0.1:2181, session id = 0
x10000087a760001, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0] create -s -e /election/candidate
Created /election/candidate0000000001
[zk: localhost:2181(CONNECTED) 1] ls /election
[candidate0000000000, candidate0000000001, candidate0000000002]
[zk: localhost:2181(CONNECTED) 2]
```

```
05] - Socket connection established, initiating session, client: /127.0.0.1:52368, server: localhost/127.0.0.1:2181
2022-03-10 17:09:32,767 [myid:localhost:2181] - INFO  [main-SendThread(localhost:2181):o.a.z.ClientCnxn$SendThread@14
44] - Session establishment complete on server localhost/127.0.0.1:2181, session id = 0x10000087a760002, negotiated t
imeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
JLine support is enabled
[zk: localhost:2181(CONNECTED) 0] create -s -e /election/candidate
Created /election/candidate0000000002
[zk: localhost:2181(CONNECTED) 1] ls /election
[candidate0000000000, candidate0000000001, candidate0000000002]
[zk: localhost:2181(CONNECTED) 2]
```

b) Implement Leader Election

Leader Election is implemented in a manner such that every znode 'watches' the znode that is sequentially before it. If the znode that is the leader fails, the znode watching the leader receives a watch event and elects itself as leader.

1. Setup watches such that every znode 'watches' the znode that's sequentially before it.



```
2022-03-10 17:11:36,325 [myid:] - WARN  [NIOWorkerThread-1:o.a.z.s.NIOServerCnxn@371] - Unexpected exception
org.apache.zookeeper.server.ServerCnxn$EndOfStreamException: Unable to read additional data from client, it probably closed the socket: address = /127.0.0.1:52370, session = 0x10000087a760003
        at org.apache.zookeeper.server.NIOServerCnxn.handleFailedRead(NIOServerCnxn.java:170)
        at org.apache.zookeeper.server.NIOServerCnxn.doIO(NIOServerCnxn.java:333)
        at org.apache.zookeeper.server.NIOServerCnxnFactory$IOWorkRequest.doWork(NIOServerCnxnFactory.java:508)
        at org.apache.zookeeper.server.WorkerService$ScheduledWorkRequest.run(WorkerService.java:153)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
        at java.base/java.lang.Thread.run(Thread.java:833)
2022-03-10 17:11:57,647 [myid:] - INFO  [SessionTracker:o.a.z.s.ZooKeeperServer@632] - Expiring session 0x10000087a760003, timeout of 30000ms exceeded
```

```
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3]
[zk: localhost:2181(CONNECTED) 3] ls /election
[candidate0000000000, candidate0000000001, candidate0000000002]
[zk: localhost:2181(CONNECTED) 4]
```

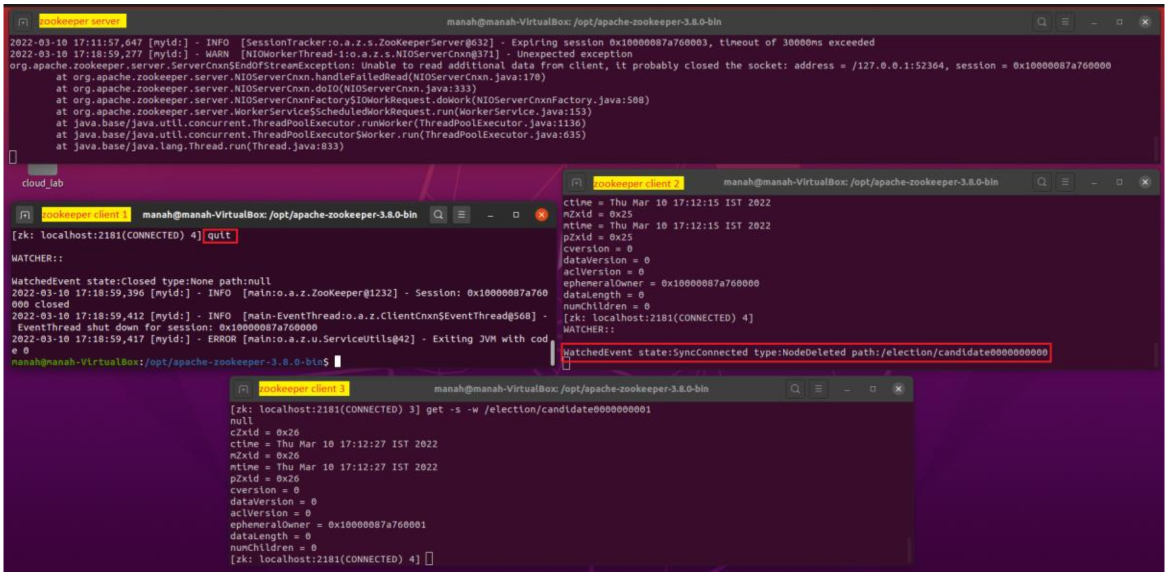
```
[candidate0000000000, candidate0000000001, candidate0000000002]
[zk: localhost:2181(CONNECTED) 3] get -s -w /election/candidate0000000000
null
cZxid = 0x25
ctime = Thu Mar 10 17:12:15 IST 2022
mZxid = 0x25
mtime = Thu Mar 10 17:12:15 IST 2022
pZxid = 0x25
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x10000087a760000
dataLength = 0
numChildren = 0
[zk: localhost:2181(CONNECTED) 4]
```

client 2's znode has id 1 (candidate0000000001).
therfore, it sets a watch on znode candidate0000000000.

```
[zk: localhost:2181(CONNECTED) 3] get -s -w /election/candidate0000000001
null
cZxid = 0x26
ctime = Thu Mar 10 17:12:27 IST 2022
mZxid = 0x26
mtime = Thu Mar 10 17:12:27 IST 2022
pZxid = 0x26
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x10000087a760001
dataLength = 0
numChildren = 0
[zk: localhost:2181(CONNECTED) 4]
```

client 3's znode has id 2 (candidate00000000012).
therfore, it sets a watch on znode candidate0000000001.

2. Now, quit any of the clients (except the client which had the sequentially greatest znode/ the one with no watches set on it). Since the znodes created were ephemeral, the znodes will die as well.
3. Watch the logs on the zookeeper server terminal as well as the zookeeper client terminal that set a watch on the client whose znode was killed.
   Observe the WatchedEvent logs and take a screenshot that shows both the zookeeper logs and the WatchedEvent logs.
   Observe the details presented in the log, see how it specifies a NodeDeleted event.



```
2022-03-10 17:11:57,647 [myid:] - INFO  [SessionTracker:o.a.z.s.ZooKeeperServer@632] - Expiring session 0x10000087a760003, timeout of 30000ms exceeded
2022-03-10 17:18:59,277 [myid:] - WARN  [NIOWorkerThread-1:o.a.z.s.NIOServerCnxn@371] - Unexpected exception
org.apache.zookeeper.server.ServerCnxn$EndOfStreamException: Unable to read additional data from client, it probably closed the socket: address = /127.0.0.1:52364, session = 0x10000087a760000
        at org.apache.zookeeper.server.NIOServerCnxn.handleFailedRead(NIOServerCnxn.java:170)
        at org.apache.zookeeper.server.NIOServerCnxn.doIO(NIOServerCnxn.java:333)
        at org.apache.zookeeper.server.NIOServerCnxnFactory$IOWorkRequest.doWork(NIOServerCnxnFactory.java:508)
        at org.apache.zookeeper.server.WorkerService$ScheduledWorkRequest.run(WorkerService.java:153)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
        at java.base/java.lang.Thread.run(Thread.java:833)
```

```
[zk: localhost:2181(CONNECTED) 4] quit

WATCHER::

WatchedEvent state:Closed type:None path:null
2022-03-10 17:18:59,396 [myid:] - INFO  [main:o.a.z.ZooKeeper@1232] - Session: 0x10000087a760
000 closed
2022-03-10 17:18:59,412 [myid:] - INFO  [main-EventThread:o.a.z.ClientCnxn$EventThread@568] -
 EventThread shut down for session: 0x10000087a760000
2022-03-10 17:18:59,417 [myid:] - ERROR [main:o.a.z.u.ServiceUtils@42] - Exiting JVM with cod
e 0
manah@manah-VirtualBox:/opt/apache-zookeeper-3.8.0-bin$
```

```
ctime = Thu Mar 10 17:12:15 IST 2022
mZxid = 0x25
mtime = Thu Mar 10 17:12:15 IST 2022
pZxid = 0x25
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x10000087a760000
dataLength = 0
numChildren = 0
[zk: localhost:2181(CONNECTED) 4]
WATCHER::

WatchedEvent state:SyncConnected type:NodeDeleted path:/election/candidate0000000000
```

```
[zk: localhost:2181(CONNECTED) 3] get -s -w /election/candidate0000000001
null
cZxid = 0x26
ctime = Thu Mar 10 17:12:27 IST 2022
mZxid = 0x26
mtime = Thu Mar 10 17:12:27 IST 2022
pZxid = 0x26
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x10000087a760001
dataLength = 0
numChildren = 0
[zk: localhost:2181(CONNECTED) 4]
```

[OPTIONAL]

Apache also provides a Zookeeper API, using which an application can connect, interact, manipulate data, coordinate, and finally disconnect from a Zookeeper ensemble. This API may be used to implement the leader election logic and more. You may explore the same here.