Assignment 4 - Introduction to Apache Zookeeper

In this week's assignment, 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

The following screenshots are to be submitted in a PDF file:

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

Name the file **<SECTION>_<SRN>_<NAME>_E4** (eg: *E_PES1UG19CS999_Rumali_E4.pdf*").

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 = /opt/apache-zookeeper-3.8.0-bin/data
clientPort = 2181
initLimit = 5
syncLimit = 2
```

6. Start the Zookeeper Server
```
bin/zkServer.sh start
```

   The output should resemble -
```
laruim@pop-os:/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
laruim@pop-os:/opt/apache-zookeeper-3.8.0-bin$
```
   (Take a screenshot, 1a):

7. Start the Zookeeper CLI
```
bin/zkCli.sh
```

   The output should resemble -
```
laruim@pop-os:/opt/apache-zookeeper-3.8.0-bin$ bin/zkCli.sh
Connecting to localhost:2181
2023-03-17 00:31:16,991 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
ironment:zookeeper.version=3.8.0-5a02a05eddb59aee6ac762f7ea82e92a68eb9c0f, built
 on 2022-02-25 08:49 UTC
2023-03-17 00:31:16,993 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
ironment:host.name=pop-os.localdomain
2023-03-17 00:31:16,993 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
ironment:java.version=17.0.1
2023-03-17 00:31:16,993 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
ironment:java.vendor=Oracle Corporation
2023-03-17 00:31:16,993 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
ironment:java.home=/usr/lib/jvm/java-17-oracle
2023-03-17 00:31:16,993 [myid:] - INFO  [main:o.a.z.Environment@98] - Client env
```

```
2023-03-17 00:31:17,016 [myid:localhost:2181] - INFO  [main-SendThread(localhost
:2181):o.a.z.ClientCnxn$SendThread@1005] - Socket connection established, initia
ting session, client: /[0:0:0:0:0:0:0:1]:46054, server: localhost/[0:0:0:0:0:0:0
:1]:2181
JLine support is enabled
2023-03-17 00:31:17,030 [myid:localhost:2181] - INFO  [main-SendThread(localhost
:2181):o.a.z.ClientCnxn$SendThread@1444] - Session establishment complete on ser
ver localhost/[0:0:0:0:0:0:0:1]:2181, session id = 0x10001119c280000, negotiated
 timeout = 30000

WATCHER::

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

Type `quit` to exit from the ZK CLI.

8. Stop the Zookeeper Server
   `bin/zkServer.sh stop`

   The output should resemble -

```
laruim@pop-os:/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 as described in **Task A.**

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 with 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, 2a)**
   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, 2b):**

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.

   `get -s /your_srn_seq<some_sequence>`

   (Take a screenshot, 2c)

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

3. **Create Children/ Sub-Znodes**

   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 "childnode"
   ```

   Let the **parent** be *your_srn* and **children** be *your_name*
   Create **two children,** one with your **first name** and one with your **last name.**

   List all children using `ls /parent` **(Take a screenshot, 2d)**

4. **Delete Znode**

   Usage: `delete /path`
   OR
   Usage: `rmr /path`

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

Leader election doesn't need to happen only via Paxos or Raft. Even Zookeeper can be used for it! 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.
This is **not** a great or foolproof leader election algorithm,

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

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

1. Download the start_terminals.sh file and navigate to the place you downloaded it. Now, do the below from whichever directory you downloaded the file in:
   `mv start_terminals.sh /opt/apache-zookeeper-3.8.0-bin/start_terminals.sh`
   Now do:
   `cd /opt/apache-zookeeper-3.8.0-bin`
   Type `sudo chmod +x ./start_terminals.sh` to grant permissions.

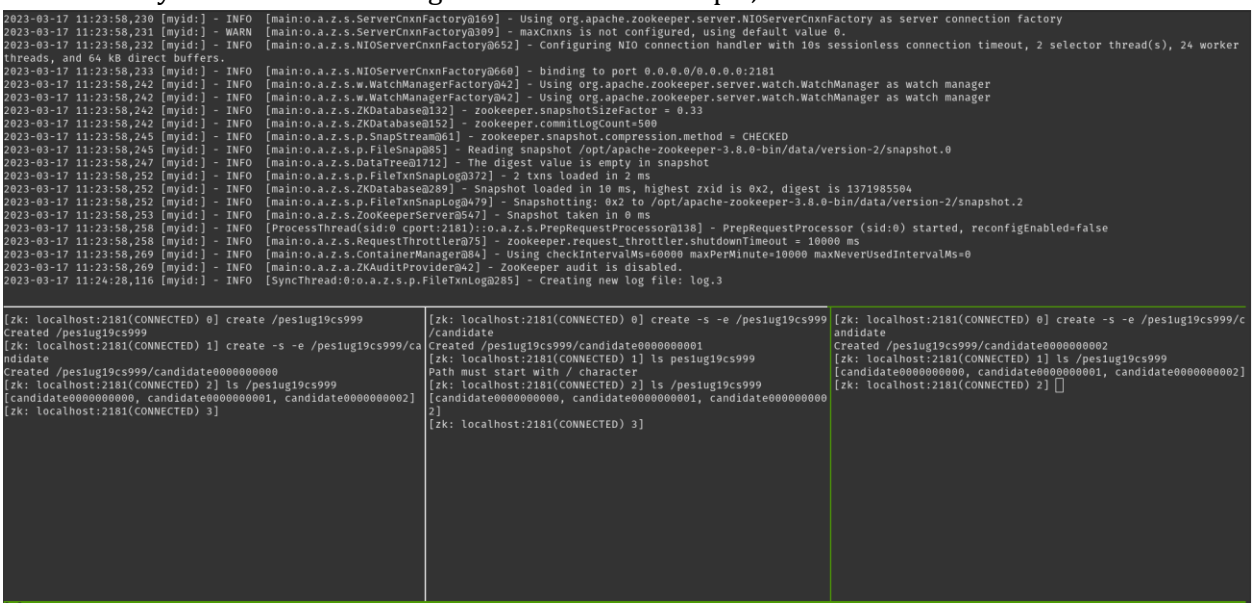
2. Run `./start_terminals.sh` file provided to you to open four panes; One of these will run the zookeeper server while the other three will function as zookeeper clients. Refer to Experiment 4's manual for instructions on how to switch between panes.
   a. Start up the zookeeper server on top (biggest) pane. Run it in the foreground as so: `bin/zkServer.sh start-foreground`
   b. Start up the zookeeper cli on the other terminals. These will function as clients. `bin/zkCli.sh`

3. 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; This is to make it **persistent**, as we want our ephemeral sequential znodes to continue in sequence later even for new clients.

4. On each of the clients, create a **sequential, ephemera**l znode under the root znode created earlier; i.e., under **/your_srn**. You may name these znodes 'candidate'. Refer to the previous commands guide to learn how to create this.

5. 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:

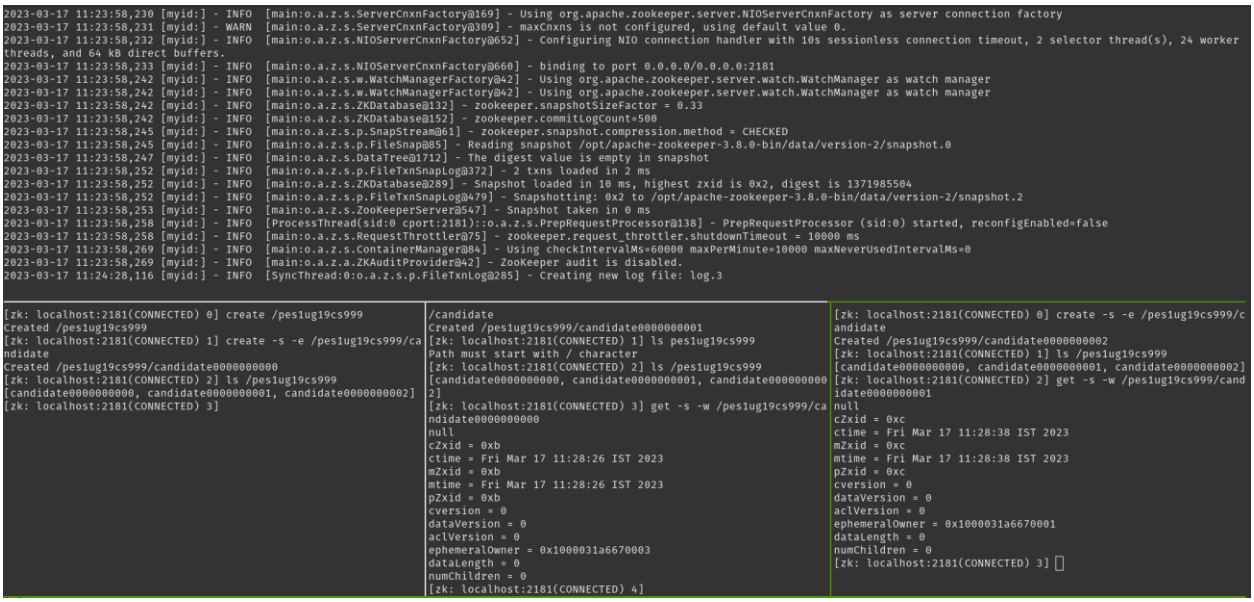


(Take a screenshot, 3a)

b) **<u>Implement Leader Election</u>**

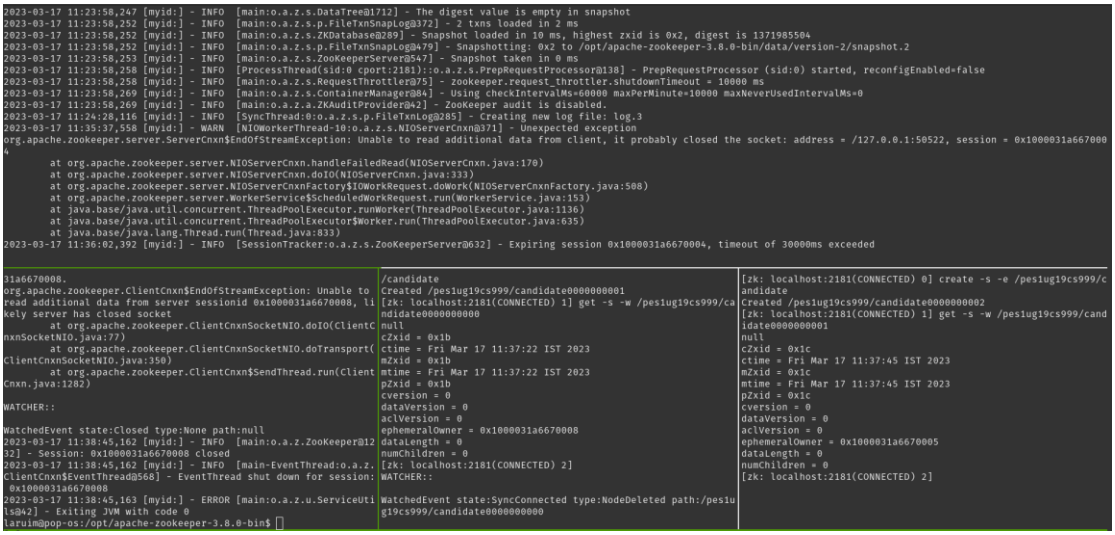
Leader Election is implemented in a manner such that every client has a corresponding **ephemeral sequential znode** that 'watches' the znode that is sequentially before it. If the znode that is the leader fails, the znode watching the leader receives a watch event and elects itself as leader. **There is no concept of a majority** here.

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

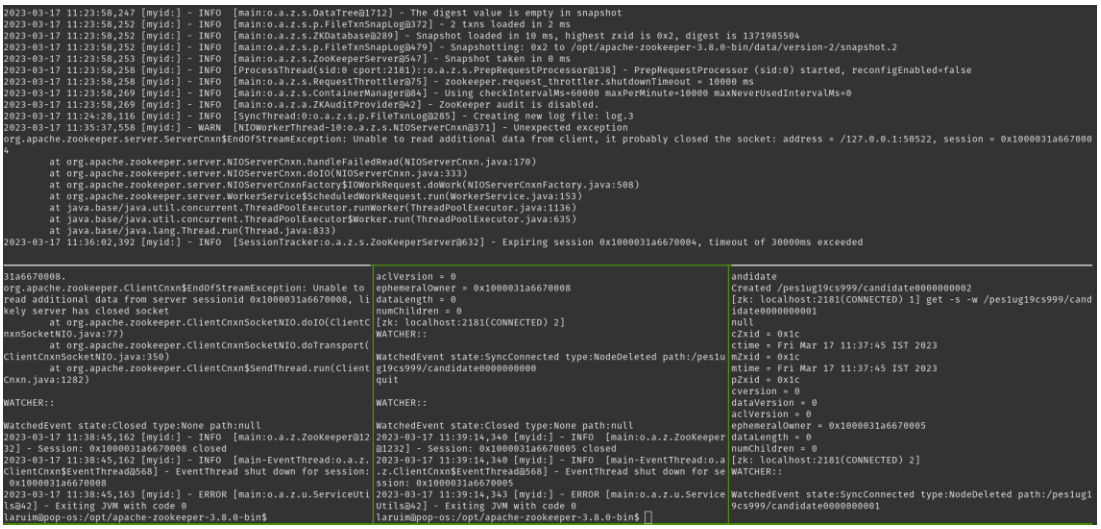


If possible, clear each pane with ctrl+l; this might not be possible to do on Mac.

2. Now, quit from the first, and then the second client. Since the znodes created were ephemeral, the znodes will die as well.



On quitting the first client, the second client registers a NodeDeleted event, because the first client's ephemeral znode was deleted. If we implemented full leader election, **this event** would prompt the second client to register itself as leader.



On quitting the second client, the third client registers a NodeDeleted event, because the second client's ephemeral znode was deleted. If we implemented full leader election, **this event** would prompt the third client to register itself as leader.

(Take a screenshot, 3b)

3. Now, in the first client terminal pane (the one where the first client was), restart the client and repeat the process of creating an ephemeral sequential node (**not the root node)**; you'll see that the sequence number isn't 0, but **3**! This is because the sequential nodes are created as children of the root persistent node! This is to ensure that if the old leader fails and comes back, it needs to wait its turn instead of hijacking and potentially losing data in the process.

   (Take a screenshot, 3c)

The CLIs themselves are substitutes for the **Zookeeper API**. Most applications would use this along with its main purpose, for leader election. However, what you've done is directly use the CLI to make your life just a tad bit easier.

[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.