PRACTICAL-4

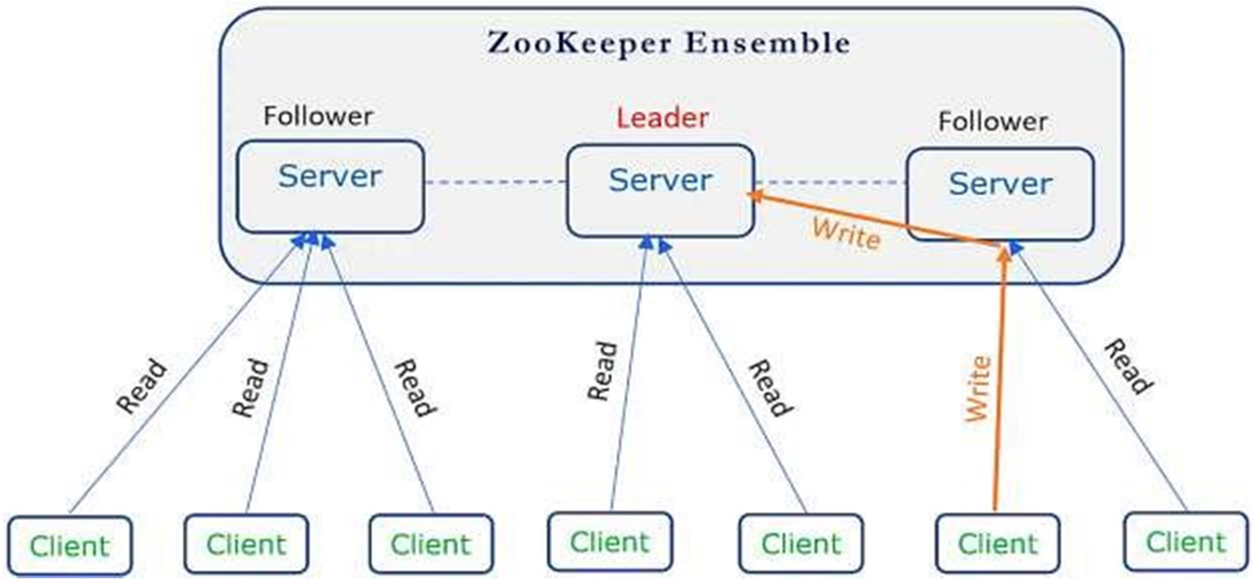
AIM :- Perform the Zookeeper commands in CloudxLab and Windows

ZOOKEEPER

- Zookeeper in Hadoop can be considered a centralized repository where distributed applications can put data into and retrieve data from.
- It makes a distributed system work together as a whole using its synchronization, serialization, and coordination goals.
- Zookeeper can be thought of as a file system where we have nodes that store data instead of files or directories that store data.

Why do we need Zookeeper in the Hadoop?

Distributed applications are difficult to coordinate and work with, as they are much more error prone due to huge number of machines attached to network. As many machines are involved, race condition and deadlocks are common problems when implementing distributed applications. Race condition occurs when a machine tries to perform two or more operations at a time and this can be taken care by serialization property of ZooKeeper. Deadlocks are when two or more machines try to access same-shared resource at the same time. More precisely, they try to access each other's resources, which leads to lock of system as none of the system is releasing the resource but waiting for other system to release it. Synchronization in Zookeeper helps to solve the deadlock. Another major issue with distributed application can be partial failure of process, which can lead to inconsistency of data. Zookeeper handles this through atomicity, which means either whole of the process will finish or nothing will persist after failure. Thus Zookeeper is an important part of Hadoop that take care of these small but important issues so that developer can focus more on functionality of the application. Following figure shows the Zookeeper architecture



Writes in Zookeeper

All the writes in Zookeeper go through the Master node, thus it is guaranteed that all writes will be sequential. On performing write operation to the Zookeeper, each server attached to that client persists the data along with master. Thus, this makes all the servers updated about the data. However, this also means that concurrent writes cannot be made. Linear writes guarantee can be problematic if Zookeeper is used for write dominant workload. Zookeeper in Hadoop is ideally used for coordinating message exchanges between clients, which involves less writes and more reads. Zookeeper is helpful till the time the data is shared but if application has concurrent data writing then Zookeeper can come in way of the application and impose strict ordering of operations.

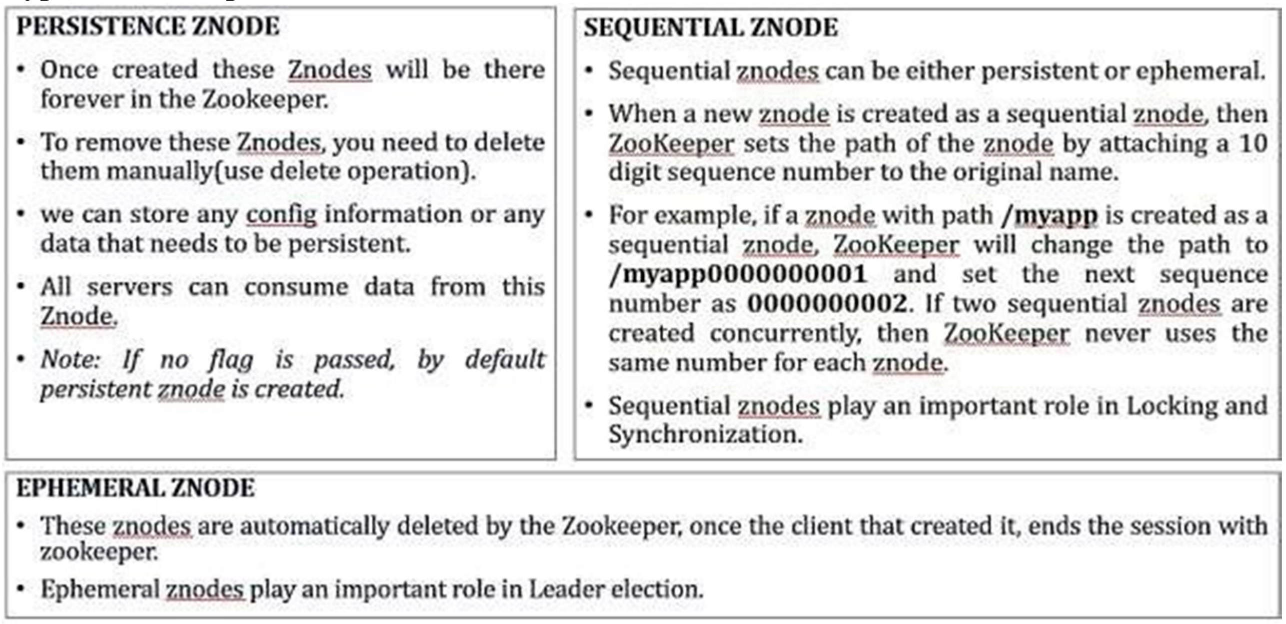
Reads in Zookeeper

Zookeeper is best at reads as reads can be concurrent. Concurrent reads are done as each client is attached to different server and all clients can read from the servers simultaneously, although having concurrent reads leads to eventual consistency as master is not involved. There can be cases where client may have an outdated view, which gets updated with a little delay.

What is ZNode?

ZNode can be thought of as directories in the file system. The only difference is that Znodes have the capability of storing data as well. ZNode's task is to maintain the statistics of the structure and version numbers for data changes. Data 'read' and 'write' happen in its entirety, as Zookeeper does not allow partial 'read' and 'write'.

Types of Zookeeper Node



PERSISTENCE ZNODE

- Once created these Znodes will be there forever in the Zookeeper.
- To remove these Znodes, you need to delete them manually(use delete operation).
- we can store any config information or any data that needs to be persistent.
- All servers can consume data from this Znode.
- *Note: If no flag is passed, by default persistent znode is created.*

SEQUENTIAL ZNODE

- Sequential znodes can be either persistent or ephemeral.
- When a new znode is created as a sequential znode, then ZooKeeper sets the path of the znode by attaching a 10 digit sequence number to the original name.
- For example, if a znode with path **/myapp** is created as a sequential znode, ZooKeeper will change the path to **/myapp0000000001** and set the next sequence number as **0000000002**. If two sequential znodes are created concurrently, then ZooKeeper never uses the same number for each znode.
- Sequential znodes play an important role in Locking and Synchronization.

EPHEMERAL ZNODE

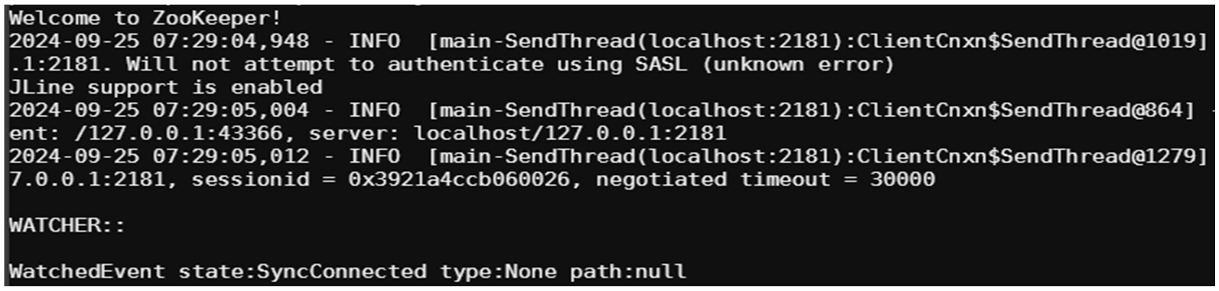
- These znodes are automatically deleted by the Zookeeper, once the client that created it, ends the session with zookeeper.
- Ephemeral znodes play an important role in Leader election.

ZOOKEEPER Command Line Interface – CLOUDXLAB

ZooKeeper Command Line Interface (CLI) is used to interact with the ZooKeeper ensemble, which lets you perform simple, file-like operations. It is useful for debugging purposes. To perform ZooKeeper CLI operations, first start your ZooKeeper server and Client.

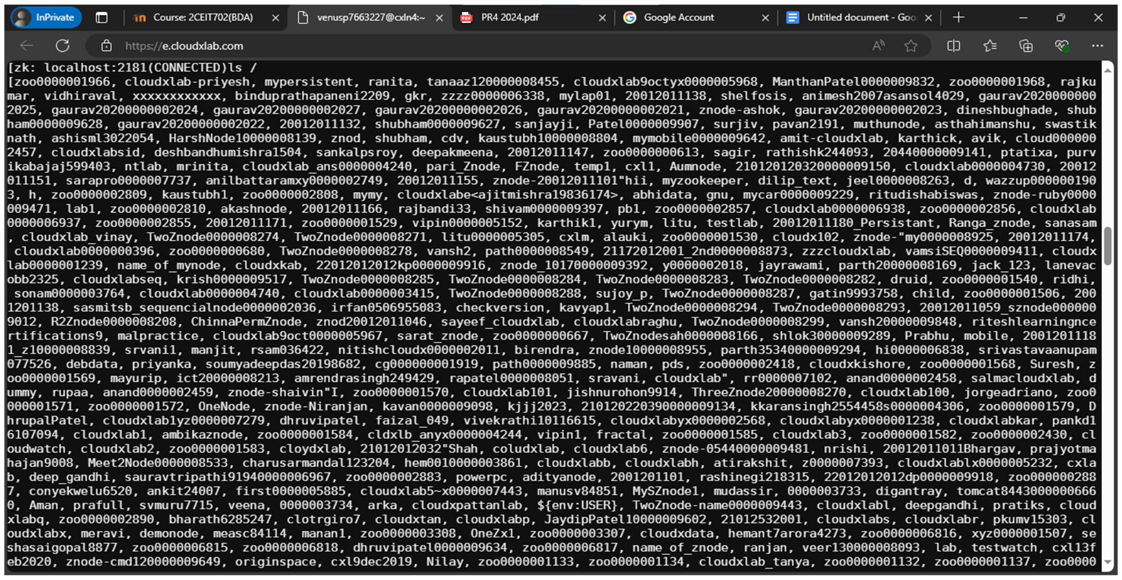
STEP-1: Open webconsole (in cloudxlab)

STEP-2: Type the command "zookeeper-client" and enter



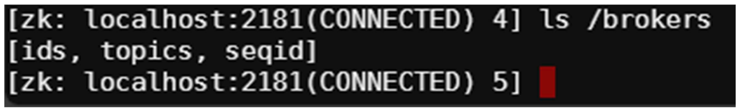
```
Welcome to ZooKeeper!
2024-09-25 07:29:04,948 - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1019]
.1:2181. Will not attempt to authenticate using SASL (unknown error)
JLine support is enabled
2024-09-25 07:29:05,004 - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@864]
ent: /127.0.0.1:43366, server: localhost/127.0.0.1:2181
2024-09-25 07:29:05,012 - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1279]
7.0.0.1:2181, sessionid = 0x3921a4ccb060026, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
```

1. **ls / :-** See the list of ZNodes at the top level



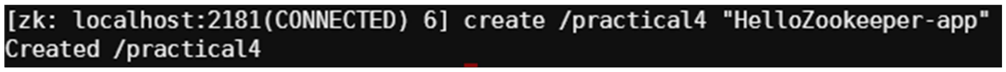
```
[zk: localhost:2181(CONNECTED)ls /
[zoo0000001966, cloudxlab-priyesh, mypersistent, ranita, tanaaz120000008455, cloudxlab9octyx0000005968, ManthanPatel0000009832, zoo0000001968, rajku
mar, vidhiraval, xxxxxxxxxxxx, binduprathapaneni2209, gkr, zzzz0000006338, mylap01, 20012011138, shelfosis, animesh2007asansol4029, gaurav2020000000
2025, gaurav20200000002024, gaurav20200000002027, gaurav20200000002026, gaurav20200000002021, znode-ashok, gaurav20200000002023, dineshbughade, shub
ham0000009628, gaurav20200000002022, 20012011132, shubham0000009627, sanjayji, Patel0000009907, surjiv, pavan2191, muthunode, asthahimanshu, swastik
nath, ashisml3022054, HarshNode10000008139, znod, shubham, cdv, kaustubh10000008804, mymobile0000009642, amit-cloudxlab, karthick, avik, cloud000000
2457, cloudxlabsid, deshbandhumishra1504, sankalpsroy, deepakmeena, 20012011147, zoo0000000613, sagir, rathishk244093, 20440000009141, ptatixa, purv
ikabajaj599403, ntlab, mrinita, cloudxlab_ans0000004240, pari_Znode, FZnode, temp1, cxl1, Aumnode, 210120120320000009150, cloudxlab0000004730, 20012
011151, sarapro0000007737, anilbattaramxy0000002749, 20012011155, znode-20012011101"hii, myzookeeper, dilip_text, jeel0000008263, d, wazzup000000190
3, h, zoo0000002809, kaustubh1, zoo0000002808, mymy, cloudxlabe<ajitmishra19836174>, abhidata, gnu, mycar0000009229, ritudishabiswas, znode-ruby0000
009471, lab1, zoo0000002810, akashnode, 20012011166, rajbandi33, shivam0000009397, pb1, zoo0000002857, cloudxlab0000006938, zoo0000002856, cloudxlab
0000006937, zoo0000002855, 20012011171, zoo0000001529, vipin0000005152, karthik1, yurym, litu, testlab, 20012011180_Persistant, Ranga_znode, sanasam
, cloudxlab_vinay, TwoZnode0000008274, TwoZnode0000008271, litu0000005305, cxlm, alauki, zoo0000001530, cloudx102, znode-"my0000008925, 20012011174,
 cloudxlab0000000396, zoo0000000680, TwoZnode0000008278, vansh2, path0000008549, 21172012001_2nd0000008873, zzzcloudxlab, vamsiSEQ0000009411, cloudx
lab0000001239, name_of_mynode, cloudxkab, 22012012012kp0000009916, znode_10170000009392, y0000002018, jayrawami, parth20000008169, jack_123, lanevac
obb2325, cloudxlabseq, krish0000009517, TwoZnode0000008285, TwoZnode0000008284, TwoZnode0000008283, TwoZnode0000008282, druid, zoo0000001540, ridhi,
 sonam0000003764, cloudxlab0000004740, cloudxlab0000003415, TwoZnode0000008288, sujoy_p, TwoZnode0000008287, gatin9993758, child, zoo0000001506, 200
1201138, sasmitsb_sequencialnode0000002036, irfan0506955083, checkversion, kavyap1, TwoZnode0000008294, TwoZnode0000008293, 20012011059_sznode000000
9012, R2Znode0000008208, ChinnaPermZnode, znod20012011046, sayeef_cloudxlab, cloudxlabraghu, TwoZnode0000008299, vansh20000009848, riteshlearningnce
rtifications9, malpractice, cloudxlab9oct0000005967, sarat_znode, zoo0000000667, TwoZnodesah0000008166, shlok30000009289, Prabhu, mobile, 2001201118
1_z10000008839, srvani1, manjit, rsam036422, nitishcloudx0000002011, birendra, znode10000008955, parth35340000009294, hi0000006838, srivastavaanupam
077526, debdata, priyanka, soumyadeepdas20198682, cg000000001919, path0000009885, naman, pds, zoo0000002418, cloudxkishore, zoo0000001568, Suresh, z
oo0000001569, mayurip, ict20000008213, amrendrasingh249429, rapatel0000008051, sravani, cloudxlab", rr0000007102, anand0000002458, salmacloudxlab, d
ummy, rupaa, anand0000002459, znode-shaivin"I, zoo0000001570, cloudxlab101, jishnurohon9914, ThreeZnode20000008270, cloudxlab100, jorgeadriano, zoo0
000001571, zoo0000001572, OneNode, znode-Niranjan, kavan0000009098, kjjj2023, 210120220390000009134, kkaransingh2554458s0000004306, zoo0000001579, D
hrupalPatel, cloudxlab1yz0000007279, dhruvipatel, faizal_049, vivekrathi10116615, cloudxlabyx0000002568, cloudxlabyx0000001238, cloudxlabkar, pankd1
6107094, cloudxlab1, ambikaznode, zoo0000001584, cldxlb_anyx0000004244, vipin1, fractal, zoo0000001585, cloudxlab3, zoo0000001582, zoo0000002430, cl
oudwatch, cloudxlab2, zoo0000001583, cloydxlab, 21012012032"Shah, coludxlab, cloudxlab6, znode-05440000009481, nrishi, 20012011011Bhargav, prajyotma
hajan9008, Meet2Node0000008533, charusarmandal123204, hem0010000003861, cloudxlabb, cloudxlabh, atirakshit, z0000007393, cloudxlablx0000005232, cxla
b, deep_gandhi, sauravtripathi91940000006967, zoo0000002883, powerpc, adityanode, 2001201101, rashinegi218315, 22012012012dp0000009918, zoo000000288
7, conyekwelu6520, ankit24007, first0000005885, cloudxlab5~x0000007443, manusv84851, MySZnode1, mudassir, 0000003733, digantray, tomcat844300000666
0, Aman, prafull, svmuru7715, veena, 0000003734, arka, cloudxpattanlab, ${env:USER}, TwoZnode-name0000009443, cloudxlabl, deepgandhi, pratiks, cloud
xlabq, zoo0000002890, bharath6285247, clotrgiro7, cloudxtan, cloudxlabp, JaydipPatel10000009602, 21012532001, cloudxlabs, cloudxlabr, pkumv15303, cl
oudxlabx, meravi, demonode, measc84114, manan1, zoo0000003308, OneZx1, zoo0000003307, cloudxdata, hemant7arora4273, zoo0000006816, xyz0000001507, se
shasaigopal8877, zoo0000006815, zoo0000006818, dhruvipatel0000009634, zoo0000006817, name_of_znode, ranjan, veer130000008093, lab, testwatch, cxl13f
eb2020, znode-cmd120000009649, originspace, cxl9dec2019, Nilay, zoo0000001133, zoo0000001134, cloudxlab_tanya, zoo0000001132, zoo0000001137, zoo0000
```

2. **ls /brokers :-** See the children of znodes under a parent node



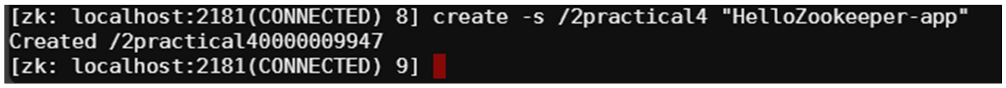
```
[zk: localhost:2181(CONNECTED) 4] ls /brokers
[ids, topics, seqid]
[zk: localhost:2181(CONNECTED) 5]
```

3. **create /path "data" :-** Create Znodes



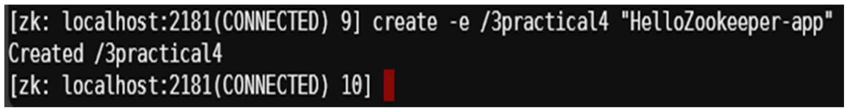
```
[zk: localhost:2181(CONNECTED) 6] create /practical4 "HelloZookeeper-app"
Created /practical4
```

4. **create –s /path "data" :-** Create Sequential znode



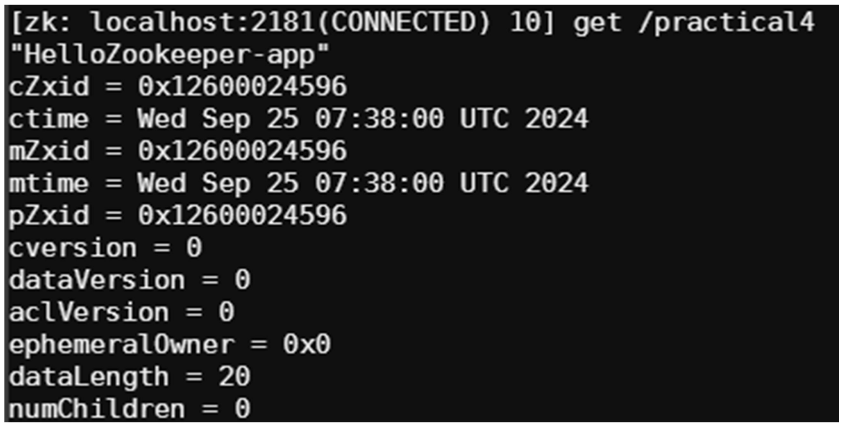
```
[zk: localhost:2181(CONNECTED) 8] create -s /2practical4 "HelloZookeeper-app"
Created /2practical40000009947
[zk: localhost:2181(CONNECTED) 9]
```

5. **create -e /path "data" :-** Create an Ephemeral Znode



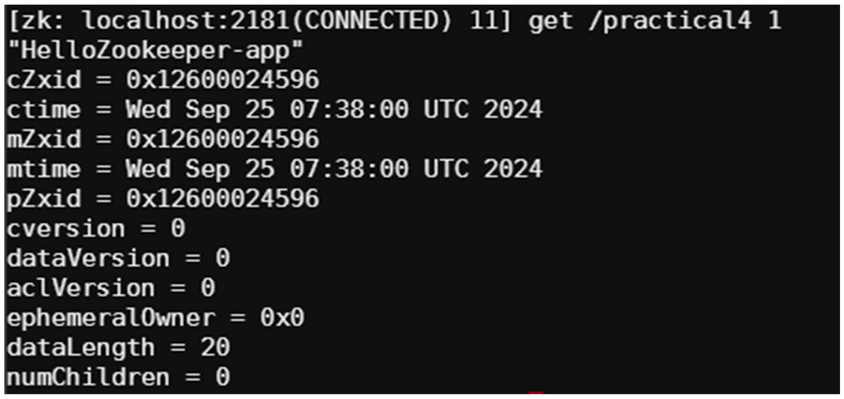
```
[zk: localhost:2181(CONNECTED) 9] create -e /3practical4 "HelloZookeeper-app"
Created /3practical4
[zk: localhost:2181(CONNECTED) 10]
```

6. **get /path :-** Get Data



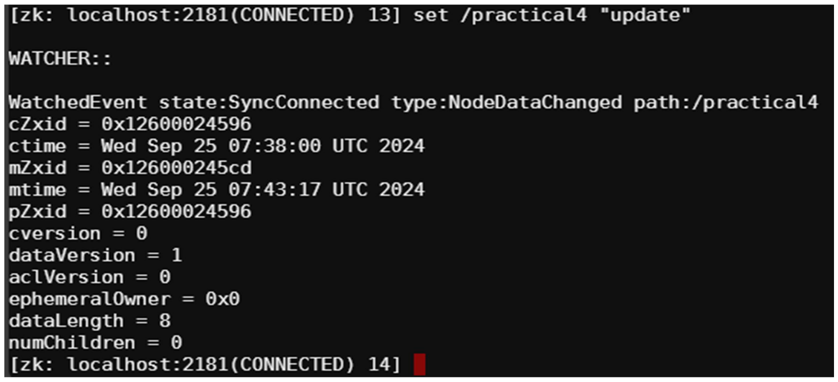
```
[zk: localhost:2181(CONNECTED) 10] get /practical4
"HelloZookeeper-app"
cZxid = 0x12600024596
ctime = Wed Sep 25 07:38:00 UTC 2024
mZxid = 0x12600024596
mtime = Wed Sep 25 07:38:00 UTC 2024
pZxid = 0x12600024596
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 20
numChildren = 0
```

7. **get /path [watch] 1 :-** Create Watch on znode



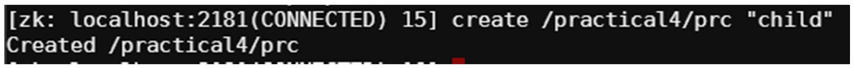
```
[zk: localhost:2181(CONNECTED) 11] get /practical4 1
"HelloZookeeper-app"
cZxid = 0x12600024596
ctime = Wed Sep 25 07:38:00 UTC 2024
mZxid = 0x12600024596
mtime = Wed Sep 25 07:38:00 UTC 2024
pZxid = 0x12600024596
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 20
numChildren = 0
```

8. **set /path “data” :-** set /FirstZnode “Data-updated”



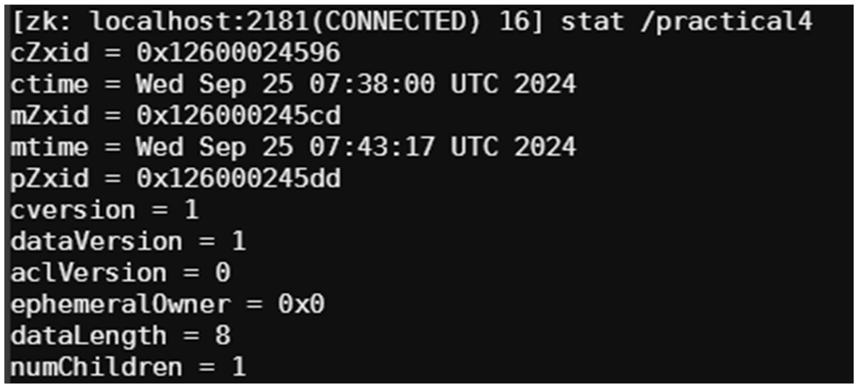
```
[zk: localhost:2181(CONNECTED) 13] set /practical4 "update"

WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/practical4
cZxid = 0x12600024596
ctime = Wed Sep 25 07:38:00 UTC 2024
mZxid = 0x126000245cd
mtime = Wed Sep 25 07:43:17 UTC 2024
pZxid = 0x12600024596
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 8
numChildren = 0
[zk: localhost:2181(CONNECTED) 14]
```

9. **create /parent/path “data” :-** Create Children / Sub-znode



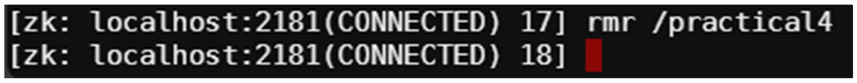
```
[zk: localhost:2181(CONNECTED) 15] create /practical4/prc "child"
Created /practical4/prc
```

10. **stat /path :-** Check Status



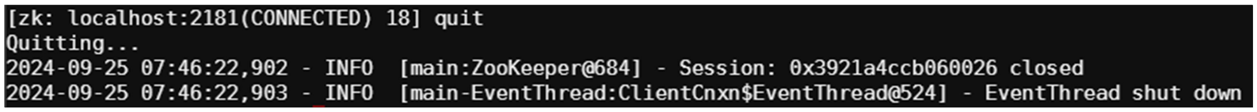
```
[zk: localhost:2181(CONNECTED) 16] stat /practical4
cZxid = 0x12600024596
ctime = Wed Sep 25 07:38:00 UTC 2024
mZxid = 0x126000245cd
mtime = Wed Sep 25 07:43:17 UTC 2024
pZxid = 0x126000245dd
cversion = 1
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 8
numChildren = 1
```

11. **rmr /path :-** Remove a Znode



```
[zk: localhost:2181(CONNECTED) 17] rmr /practical4
[zk: localhost:2181(CONNECTED) 18]
```

12. **quit :-**



```
[zk: localhost:2181(CONNECTED) 18] quit
Quitting...
2024-09-25 07:46:22,902 - INFO  [main:ZooKeeper@684] - Session: 0x3921a4ccb060026 closed
2024-09-25 07:46:22,903 - INFO  [main-EventThread:ClientCnxn$EventThread@524] - EventThread shut down
```