

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
INGENIERÍA INFORMÁTICA
SISTEMAS DISTRIBUIDOS
CARLOS LANDAETA, C.I: 27.249.061

PRÁCTICA 3

1. ¿Cuál es la diferencia entre las dos salidas anteriores?

La salida del primer comando muestra la fecha y hora local configurada en el equipo, junto con el desfase que este tiene con respecto al horario UTC. La salida del segundo comando muestra la fecha y hora UTC.

2. Recuerdan en primaria que cada día los primero que escribían de su cuaderno antes de iniciar la clase era la fecha en un formato específico. ¿Cómo sería el comando para obtener la fecha en ese formato?

El comando para escribir la fecha en el formato es: `date +"Caracas, %A %d de %B de %Y"`

3. ¿Cómo sería el comando para colocar la fecha en el primer día de este año?

El comando para colocar la fecha del equipo el primer día del año (primero de enero de 2023) es el siguiente:

```
manager@debian:~$ sudo date -s "01/01/2023"
Sun 01 Jan 2023 12:00:00 AM -04
```

4. Tome captura de la ejecución de estos comandos, mostrando el desfase y la corrección de la hora del reloj local.

La ejecución de los comandos mostrando el desfase, luego la sincronización, y finalmente mostrando que sí se modificó la fecha y hora es el siguiente:

```
manager@debian:~$ sudo ntpdate -q 200.2.15.250
server 200.2.15.250, stratum 2, offset +12327594.259743, delay 0.02786
1 Jan 00:03:49 ntpdate[855]: step time server 200.2.15.250 offset +12327594
.259743 sec
manager@debian:~$ sudo ntpdate 200.2.15.250
-bash: sudo: command not found
manager@debian:~$ sudo ntpdate 200.2.15.250
23 May 16:24:42 ntpdate[860]: step time server 200.2.15.250 offset +12327594.260711 sec
manager@debian:~$ date
Tue 23 May 2023 04:24:49 PM -04
```

5. ¿Qué se supone que el host en 200.2.15.250?

Es un servidor de tiempo, el cual utiliza el protocolo NTP(Network Time Protocol) para mantener la hora local actualizada y distribuirla a las computadoras y dispositivos conectados a la red de la universidad. Este servidor garantiza que todos los dispositivos conectados a la red de la UCAB mantengan una hora precisa y uniforme.

6. Investigue y describa paso a paso la instalación, configuración y activación de alguno de estos servicios para que funcione en la red del laboratorio en el servidor NTP de la universidad.

Se investigó cómo instalar, configurar y activar *ntpd* en Ubuntu:

- a. **Instalación:** Para instalar *ntpd*, se utiliza el siguiente comando: *sudo apt-get install ntp*
- b. **Configuración:** El archivo de configuración de *ntpd* usualmente se encuentra en */etc/ntp.conf*. El archivo de configuración predeterminado incluye algunos servidores NTP públicos. En este archivo se debe agregar el servidor NTP de la UCAB. Se debe agregar la siguiente línea al archivo: *server 200.2.15.250*
- c. **Activación:** Luego de configurar el archivo de configuración, se debe reiniciar el servicio *ntpd* para que los cambios funcionen. Se debe ejecutar el siguiente comando: *sudo systemctl restart ntp*.

Para verificar que el servicio está ejecutándose correctamente, se puede ejecutar el siguiente comando: *ntpq -p*. El comando anterior muestra la lista de servidores a los que está conectado ntpd y su estado actual.

7. Muestre la serie de comandos que ejecuto para:

- a. crear la estructura que se muestra en la siguiente imagen.
- b. Dar el valor "Esto es un dato cualquiera 1" a el znode */app_01/sala_01*.
- c. Dar el valor "Esto es un dato cualquiera 2" a el znode */app_01/sala_02*.
- d. Mostrar los valores de los znodes */app_01/sala_01* */app_01/sala_02*.

Los comandos utilizados para la creación de la aplicación en Zookeeper son los siguientes:

```
manager@debian:~$ docker run --rm -d --name zookeeper-server --network zoo-net --network-alias zookeeper -p 2181:2181 zookeeper:3.8.1
3e4e47e68be70c0802dcd250c28e818a291d1287fd1a68eaedb8aac5af8d8488
manager@debian:~$ docker run --rm -it --name zookeeper-client --network zoo-net zookeeper:3.8.1 zkCli.sh -server zookeeper
Connecting to zookeeper
```

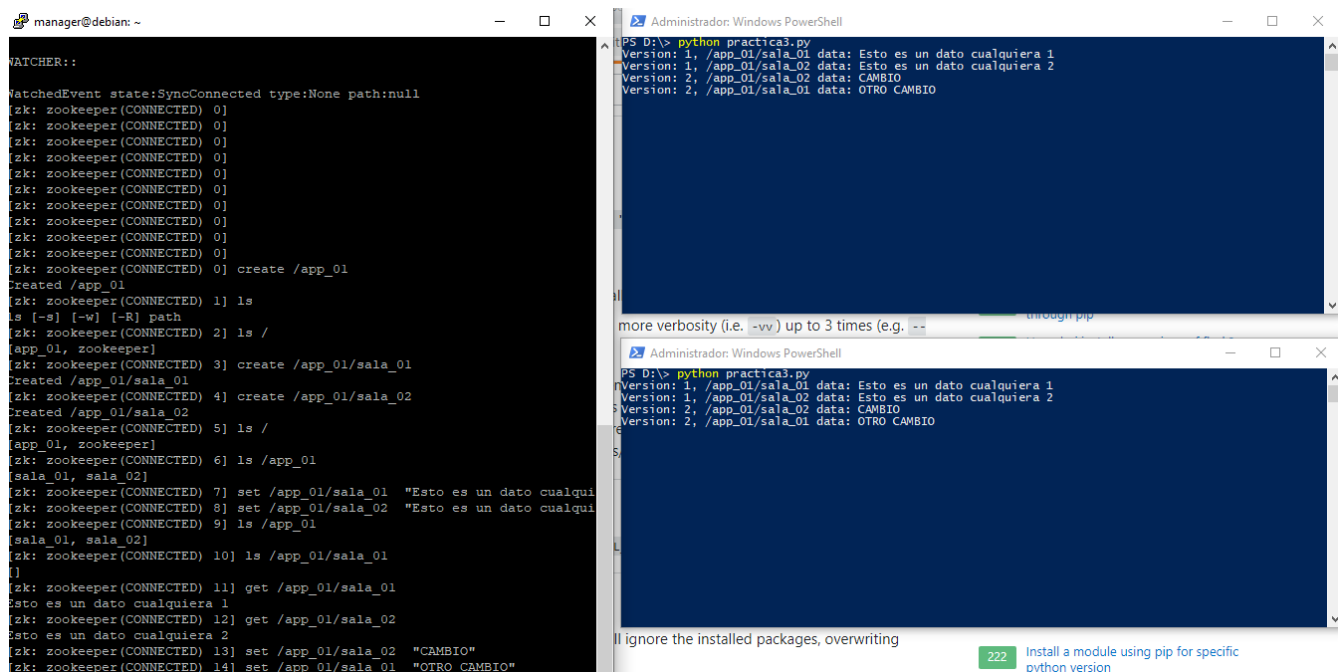
```
[zk: zookeeper(CONNECTED) 0] create /app_01
Created /app_01
[zk: zookeeper(CONNECTED) 1] ls
ls [-s] [-w] [-R] path
[zk: zookeeper(CONNECTED) 2] ls /
[app_01, zookeeper]
[zk: zookeeper(CONNECTED) 3] create /app_01/sala_01
Created /app_01/sala_01
[zk: zookeeper(CONNECTED) 4] create /app_01/sala_02
Created /app_01/sala_02
[zk: zookeeper(CONNECTED) 5] ls /
[app_01, zookeeper]
[zk: zookeeper(CONNECTED) 6] ls /app_01
[sala_01, sala_02]
[zk: zookeeper(CONNECTED) 7] set /app_01/sala_01 "Esto es un dato cualquiera 1"
[zk: zookeeper(CONNECTED) 8] set /app_01/sala_02 "Esto es un dato cualquiera 2"
```

```
[zk: zookeeper(CONNECTED) 11] get /app_01/sala_01
Esto es un dato cualquiera 1
[zk: zookeeper(CONNECTED) 12] get /app_01/sala_02
Esto es un dato cualquiera 2
```

8. Explique que es lo que hace el código y que es lo que se refleja en las distintas instancias.

Lo primero que se realiza en el código es crear una instancia de *KazooClient* con la dirección IP y el puerto del servidor. Luego, el método *start()* se conecta al servidor. Posteriormente, el método *ensure_path()* se asegura que los nodos llamados */app_01/sala_01* y */app_01/sala_02* existan, si no existen, los crea. A continuación, las dos funciones con el decorador *DataWatch* monitorean los nodos */app_01/sala_01* y */app_01/sala_02*. Estas funciones se ejecutan automáticamente cuando los datos en los nodos cambian. Cada función imprime la versión actual de los datos y los datos mismos. Finalmente, se entra en un loop infinito que espera un segundo antes de repetir el ciclo.

En las diferentes instancias del cliente, se puede observar que si se cambia un dato de un nodo, Zookeeper le notifica a todas las instancias del cambio realizado.



The image shows two terminal windows side-by-side. The left window is a terminal on a Debian system with the prompt 'manager@debian: ~'. It shows a Zookeeper command-line interface where a user connects to a Zookeeper server, creates a path '/app_01', and then creates two sub-paths '/app_01/sala_01' and '/app_01/sala_02'. The right window is a Windows PowerShell window titled 'Administrador: Windows PowerShell'. It shows the execution of a Python script 'practica3.py'. The script outputs the current data and version for '/app_01/sala_01' and '/app_01/sala_02'. It shows that the data for '/app_01/sala_01' is 'Esto es un dato cualquiera 1' and for '/app_01/sala_02' is 'Esto es un dato cualquiera 2'. Then, it shows that the data for '/app_01/sala_02' has changed to 'CAMBIO' and for '/app_01/sala_01' to 'OTRO CAMBIO'.

```
manager@debian: ~  
WATCHER:~  
WatchedEvent state:SyncConnected type:None path:null  
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0]   
zk: zookeeper(CONNECTED) 0] create /app_01  
Created /app_01  
zk: zookeeper(CONNECTED) 1] ls  
ls [-s] [-w] [-R] path  
zk: zookeeper(CONNECTED) 2] ls /  
/app_01, zookeeper  
zk: zookeeper(CONNECTED) 3] create /app_01/sala_01  
Created /app_01/sala_01  
zk: zookeeper(CONNECTED) 4] create /app_01/sala_02  
Created /app_01/sala_02  
zk: zookeeper(CONNECTED) 5] ls /  
/app_01, zookeeper  
zk: zookeeper(CONNECTED) 6] ls /app_01  
/sala_01, sala_02  
zk: Zookeeper(CONNECTED) 7] set /app_01/sala_01 "Esto es un dato cualqui  
zk: zookeeper(CONNECTED) 8] set /app_01/sala_02 "Esto es un dato cualqui  
zk: zookeeper(CONNECTED) 9] ls /app_01  
/sala_01, sala_02  
zk: Zookeeper(CONNECTED) 10] ls /app_01/sala_01  
/  
zk: zookeeper(CONNECTED) 11] get /app_01/sala_01  
Esto es un dato cualquiera 1  
zk: zookeeper(CONNECTED) 12] get /app_01/sala_02  
Esto es un dato cualquiera 2  
zk: zookeeper(CONNECTED) 13] set /app_01/sala_02 "CAMBIO"  
zk: zookeeper(CONNECTED) 14] set /app_01/sala_01 "OTRO CAMBIO"
```

```
PS D:\> python practica3.py  
Version: 1, /app_01/sala_01 data: Esto es un dato cualquiera 1  
Version: 1, /app_01/sala_02 data: Esto es un dato cualquiera 2  
Version: 2, /app_01/sala_02 data: CAMBIO  
Version: 2, /app_01/sala_01 data: OTRO CAMBIO
```

9. Explique que es lo que hace el código y que es lo que se refleja en las distintas instancias.

Primero, crea una instancia de *KazooClient* con la dirección IP y el puerto del servidor. Luego, se utiliza el método *start()* para conectarse al servidor. A continuación, se asegura que el nodo */app_01/group* exista utilizando el método *ensure_path()*, si no existe, lo crea. A continuación, se crea un nodo efímero y secuencial con el método *create()*. El valor del nodo se establece en el primer argumento de la línea de comandos y es codificado. Luego, se utiliza una función con el decorador *ChildrenWatch* para monitorear los nodos hijos del nodo */app_01/group*. Esta función se ejecuta automáticamente cuando se produce un cambio en los nodos hijos, e imprime los datos de cada nodo hijo. Finalmente, se entra en un bucle infinito que espera un segundo antes de repetir el ciclo.

En las diferentes instancias se observa que al crear o eliminar un nodo hijo, se le notifica a todas las instancias de el cambio realizado.

```
Administrador: Windows PowerShell

#####
nodo_01
#####

#####
nodo_02
nodo_01
a#####
a#####
a#####
a#####
a#####
a#####
a#####
#####

Administrador: Windows PowerShell

PS D:\> python practica3-2.py nodo_02
#####
nodo_02
nodo_01
#####

#####
nodo_02
nodo_03
nodo_01
#####

Windows PowerShell

PS D:\> python practica3-2.py nodo_03
#####
nodo_02
nodo_03
nodo_01
#####
```

Administrador: Windows PowerShell

```
2 nodo_02
nodo_01
#####

#####
nodo_02
nodo_03
nodo_01
3 #####

#####
6 nodo_03
nodo_01
#####
```

Administrador: Windows PowerShell

```
nodo_01
#####

#####
nodo_02
nodo_03
nodo_01
#####

Traceback (most recent call last):
  File "D:\practica3-2.py", line 21, in <module>
    time.sleep(1)
KeyboardInterrupt
PS D:\>
```

Windows PowerShell

```
PS D:\> python practica3-2.py nodo_03
#####
nodo_02
nodo_03
nodo_01
#####

#####
nodo_03
nodo_01
#####
```