Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
Bases de Datos Avanzadas PF-3861

Practica aplicada #2
Bases de datos distribuidas

Estudiante:
Gabriel Umaña Frías C09913

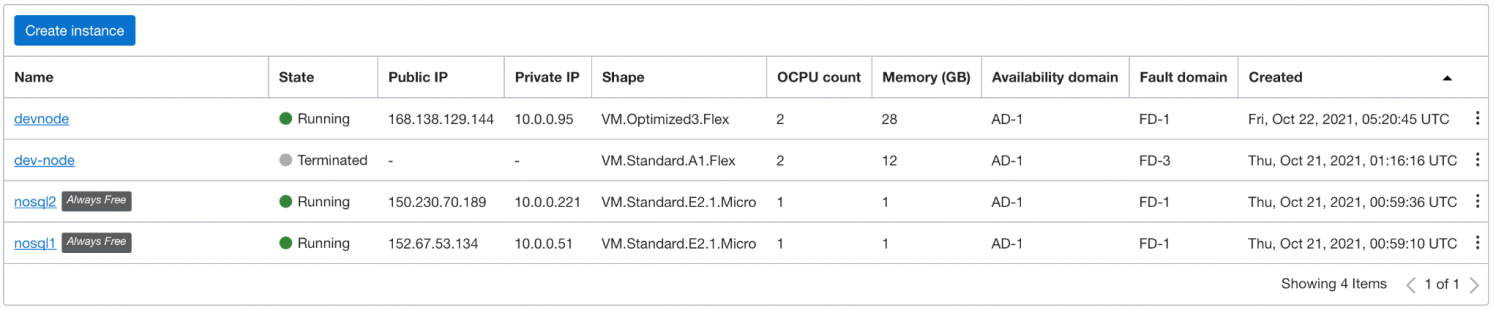
Profesor:
Msc. Richard Delgado

II Ciclo 2020

Resultados:

Primera Parte - Configuración del Ambiente Base

1. Se utiliza la cuenta de Oracle Cloud Academy para crear tres máquinas virtuales con la imagen de Oracle Linux. Dos de ellas denominadas “nosql” donde se corre el software de Oracle NoSQL y una de ellas donde se desarrolla mediante el IDE llamado Intellij IDEA:



Create instance

Name	State	Public IP	Private IP	Shape	OCPU count	Memory (GB)	Availability domain	Fault domain	Created
devnode	Running	168.138.129.144	10.0.0.95	VM.Optimized3.Flex	2	28	AD-1	FD-1	Fri, Oct 22, 2021, 05:20:45 UTC
dev-node	Terminated	-	-	VM.Standard.A1.Flex	2	12	AD-1	FD-3	Thu, Oct 21, 2021, 01:16:16 UTC
nosql2 Always Free	Running	150.230.70.189	10.0.0.221	VM.Standard.E2.1.Micro	1	1	AD-1	FD-1	Thu, Oct 21, 2021, 00:59:36 UTC
nosql1 Always Free	Running	152.67.53.134	10.0.0.51	VM.Standard.E2.1.Micro	1	1	AD-1	FD-1	Thu, Oct 21, 2021, 00:59:10 UTC

Showing 4 Items 1 of 1

2. Como se aprecia en la imagen anterior, las IPs privadas de los servidores nosql son 10.0.0.221 y 10.0.0.51. A estas se les realiza la instalación de NoSQL mediante el script “*cluster_setup.sh*”, el cual de vuelve la siguiente información:

```
Cluster successfully created. Use the following parameters when
connecting to the service via java client:
   helperHosts=10.0.0.221:5000,10.0.0.51:5000,
   storeName=myStore

Start/stop scripts for NoSQL kvstore are located on host(s) at:
   /home/opc/nosql/kvstore/scripts/start_kvstore.sh
   /home/opc/nosql/kvstore/scripts/stop_kvstore.sh

Extended test script and TestClient.java example program are
located at:
   /home/opc/nosql/kvstore/examples/TestClient

httpproxy(s) successfully started. Use the following parameters when
connecting to the proxy(s) using various language drivers:
   endpoint=http://10.0.0.221:8080
   endpoint=http://10.0.0.51:8080

Start/stop scripts for NoSQL httpproxy are located on host(s) at:
   /home/opc/nosql/proxy/scripts/start_proxy.sh
   /home/opc/nosql/proxy/scripts/stop_proxy.sh
```

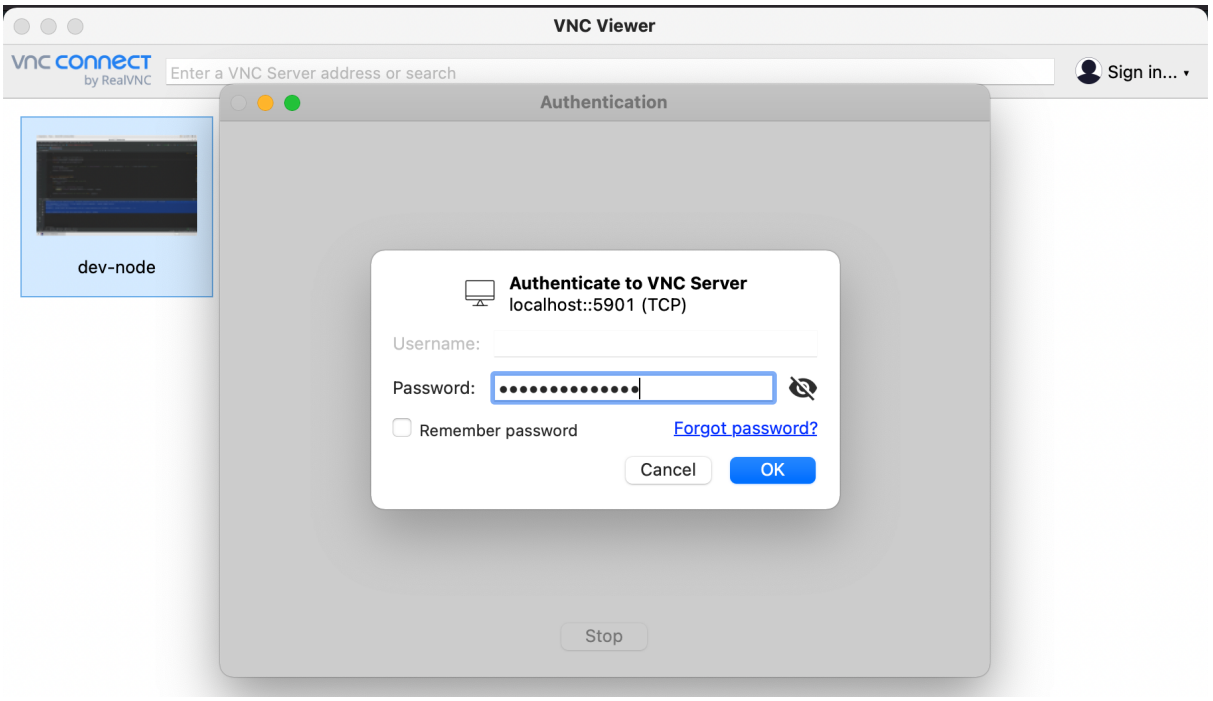
3. Los tres servidores cuentan con el Java Development Kit para desarrollar y correr el código. A continuación se muestra el resultado de correr "*java -version*" en los tres servidores:

```
[opc@devnode ~]$ java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)

[opc@nosql2 ~]$ java -version
ejava version "1.8.0_301"
Java(TM) SE Runtime Environment (build 1.8.0_301-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.301-b09, mixed mode)
[opc@nosql1 ~]$ java -version
java version "1.8.0_301"
Java(TM) SE Runtime Environment (build 1.8.0_301-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.301-b09, mixed mode)
```

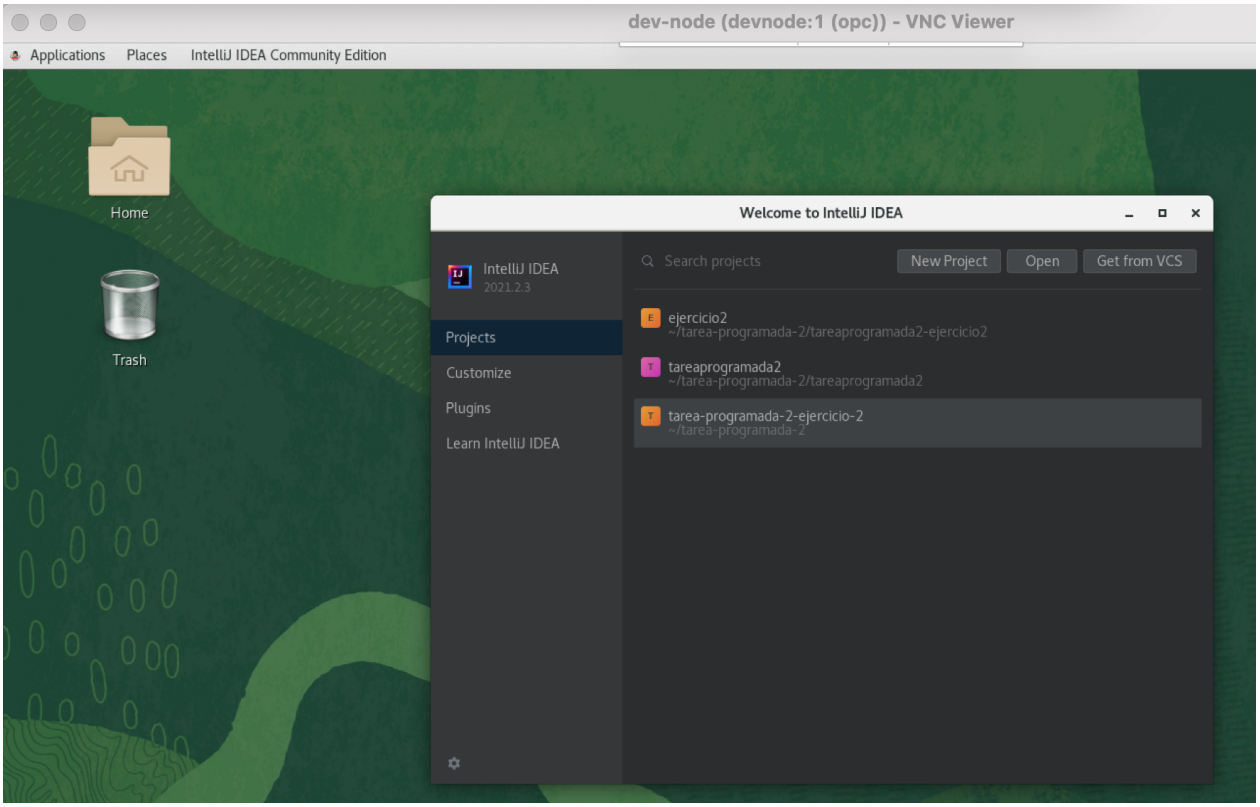
4. Podemos probar que el servidor de desarrollo tiene conectividad a los servidores de base de datos al ejecutar el comando "*java -jar kv-20.3.18/lib/kvstore.jar ping -host x.x.x.x -port 5000*":

```
[opc@devnode tarea-programada-2]$ java -jar kv-20.3.18/lib/kvstore.jar ping -host
10.0.0.221 -port 5000
Pinging components of store myStore based upon topology sequence #269
256 partitions and 4 storage nodes
Time: 2021-10-25 03:40:23 UTC   Version: 20.3.18
Shard Status: healthy:2 writable-degraded:0 read-only:0 offline:0 total:2
Admin Status: healthy

[opc@devnode tarea-programada-2]$ java -jar kv-20.3.18/lib/kvstore.jar ping -host
10.0.0.51 -port 5000
Pinging components of store myStore based upon topology sequence #269
256 partitions and 4 storage nodes
Time: 2021-10-25 03:41:39 UTC   Version: 20.3.18
Shard Status: healthy:2 writable-degraded:0 read-only:0 offline:0 total:2
Admin Status: healthy
```

5. Para conectarnos al servidor de desarrollo, utilizamos el comando "*ssh -L 5901:localhost:5901 opc@168.138.129.144 -N*". Una vez establecida la comunicación, procedemos a abrir el programa “VNC Viewer” y realizar una conexión:



6. Dentro de la máquina virtual, abrimos el IDE llamado Intellij IDEA y abrimos el proyecto en el que queremos trabajar:



Resultados:

Segunda Parte - Ejercicio 1, información de contactos

En esta segunda parte se deben de cumplir los siguientes requerimientos:

- Los contactos se consultan y se actualizan por su apellido y su nombre.
- Se requiere contar una manera eficiente para obtener a los contactos de un género
  (hombre, mujer)
- La información del teléfono y del correo electrónico típicamente se requiere al mismo tiempo.

Key model:

- Para consultar y actualizar los usuarios, se utiliza un key model donde la llave contiene como “major components” **/people/lastName/firstName/id/**. Los “minor components” utilizados son las diferentes propiedades que cada usuario tiene, como id, nombre, apellido, correo electrónico, teléfono y género. Al tener toda la información de un usuario bajo este modelo, obtener el teléfono y correo electrónico es fácil si se conoce el nombre y apellido de la persona.
- Para poder obtener los contactos, filtrados por género, de manera eficiente, se hace uso de otro key model donde se almacenan los usuarios hombres bajo el modelo **/M/lastName/firstName** y las mujeres como **/F/lastName/firstName.**

El código fuente puede ser visto en el siguiente repositorio:
https://github.com/gabriel2297/PF-3861

Por último, mediante un video se ilustra la funcionalidad del código, junto con una demostración de cómo se ve el key model para ambos ejercicios.

Segunda Parte - Ejercicio 2, Operaciones en Bulk y Rendimiento

En el ejercicio dos se crea un programa en Java, el cual inserta un registro 25000 veces de manera serial y paralela. La funcionalidad de este ejercicio puede verse en el video mencionado anteriormente así como se puede ver el código fuente en el repositorio https://github.com/gabriel2297/PF-3861

Los resultados obtenidos son:

1. Se ingresa al programa y se digitan los datos requeridos:

```
--------- MENU ---------
1. Load data concurrently (will first delete any data)
2. Load data in parallel (will first delete any data)
3. Delete current data
4. Show results
5. Exit program
```

2. Inserción de datos de manera concurrente y tiempo que toma:

```
Deleting data from DB
Deleted 0 records from DB
Inserting data to DB
[Concurrent run] - Inserted 25000 records in 67731ms (67 seconds)
```

3. Se insertan los datos de manera paralela. El tiempo que este toma es mucho menor al anterior

```
Deleting data from DB
Deleted 25000 records from DB
Inserting data to DB in parallel
Thread-0: 0~12500 completed, loaded: 12500
Thread-1: 12500~25000 completed, loaded: 12500
[Parallel run] - Inserted 25000 records in 2761ms (2 seconds)
```

4. Comparativa del tiempo tomado entre una corrida y otra:

```
[Concurrent run] - Inserted 25000 records in 67731ms (67 seconds)
[Parallel run] - Inserted 25000 records in 2761ms (2 seconds)
```

5. Se procede a borrar todos los datos para salir del programa.

```
Deleting data from DB
Deleted 24999 records from DB
```