

**Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки**

ЛАБОРАТОРНА РОБОТА № 2

з дисципліни «Побудова Cloud-систем»
на тему «Архітектури розподілених систем — кластер бази даних»

Виконав:
студент I курсу ФІОТ
групи ІМ-21мп
Андрейченко Кирило

Перевірів:
Таран В. І.

ЗМІСТ

Мета роботи:	3
Завдання:	3
Виконання лабораторної роботи:	3
Встановлення програмного забезпечення на вузли	4
Налаштування кластеру	5
Налаштування PostgreSQL для кластеризації	7
Налаштування ресурсів кластеру	9
Висновок:	11

Мета роботи:

Ознайомитися із багаторівневими архітектурами на прикладі кластеру бази даних.

Завдання:

1. Масштабувати горизонтально окремі компоненти системи.
2. Реалізувати відмовостійкість системи.

Виконання лабораторної роботи:

Для розгортання кластера необхідно підготувати інфраструктуру, що складається з щонайменше трьох вузлів. Один вузол я вже маю – db-server, потрібно встановити ще два, тобто, ще дві віртуальні операційні системи. Встановлення описувати не буду, це все вже описано в першій лабораторній роботі.

Вузли можуть мати довільні назви. Три вузли зватимуться db-server(з першої лабораторної роботи), node2 та node3.

Підключення по SSH до кожного серверу:

1. db-server: `ssh kyrylo@192.168.0.188`. Pass: **admin**.
2. node2: `ssh kyrylo@192.168.0.114`. Pass: **admin**.
3. node3: `ssh kyrylo@192.168.0.133`. Pass: **admin**.

Для розгортання відмовостійкого кластера PostgreSQL використаю наступне ПЗ:

- **PostgreSQL** – СУБД;
- **Pacemaker** – це програмне забезпечення, яке забезпечує кластеризацію вузлів високої доступності, тобто можливість автоматичного відновлення роботи системи у випадку збою в одному з вузлів. Він контролює ресурси кластера та реагує на зміни стану вузлів, запускає і зупиняє необхідні сервіси, які забезпечують роботу кластера.;
- **Corosync** – це програмний засіб для обміну повідомленнями та кластеризації в комп'ютерних системах з високою доступністю. Він є ключовою складовою у багатьох відкритих кластерних рішеннях, таких як Pacemaker, Red Hat Cluster Suite та інших. За допомогою Corosync, комп'ютери в кластері можуть взаємодіяти один з одним, обмінюючись повідомленнями про свій стан та стан інших компонентів в системі. Це дозволяє комп'ютерам у кластері працювати разом як єдине ціле та забезпечувати високу доступність сервісів та даних. Крім того, Corosync забезпечує механізми для

виявлення та відновлення випадків відмов компонентів в кластері, що допомагає забезпечити безперебійну роботу системи;

- **Fence-agents** – використовуються для забезпечення безпеки і стабільності кластерів у разі виникнення неполадок або несправностей на різних вузлах кластера. Якщо вузол не відповідає або не працює належним чином, fence agent може відключити його з електропостачання, щоб запобігти його подальшому використанню та зберегти цілісність кластера;
- **Resource-agents** – використовуються для автоматизації процесів розподілу та балансування навантаження на кластері, що дозволяє забезпечити високу доступність та надійність роботи кластера. Resource-agents можуть бути використані для керування різноманітними ресурсами, такими як бази даних, файлові системи, віртуальні машини та інші.

Встановлення програмного забезпечення на вузли

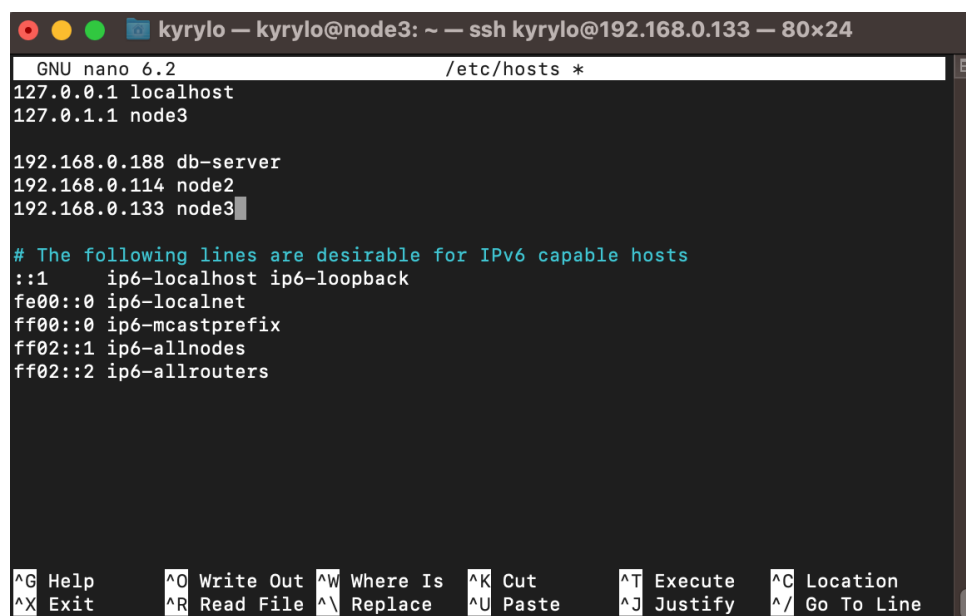
На всі підготовлені вузли кластера встановлюю необхідне ПЗ за допомогою команди: **sudo apt install corosync pcs pacemaker fence-agents postgresql**.

На всіх вузлах додаю IP-адреси всіх вузлів у файл /etc/hosts. Команда **sudo nano /etc/hosts**. Додаю:

192.168.0.188 db-server

192.168.0.114 node2

192.168.0.133 node3



```

GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 node3

192.168.0.188 db-server
192.168.0.114 node2
192.168.0.133 node3

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
  
```

Рисунок 1 – Вигляд файлу /etc/hosts після додавання IP адрес.

Налаштування кластеру

Щоб налаштувати кластер, виконую такі дії:

1. Змінюю пароль користувача `hacluster`, який було створено автоматично на всіх вузлах кластера при встановленні пакетів: **`sudo passwd hacluster`**. Пароль для всіх вузлів буде: **admin**.
2. Виконую аутентифікацію на одному з вузлів кластера, для того щоб забезпечити безпеку та захист конфіденційної інформації в системі. Команда: **`sudo pcs host auth db-server node2 node3`**.

```
kyrylo@db-server:~$ sudo pcs host auth db-server node2 node3
Username: hacluster
Password:
node2: Authorized
node3: Authorized
db-server: Authorized
```

Рисунок 2 – Виконання команди `sudo pcs host auth db-server node2 node3`.

3. Видаляю всі існуючі файли конфігурації кластера за допомогою команди: **`sudo pcs cluster destroy`**.

```
kyrylo@db-server:~$ sudo pcs cluster destroy
Shutting down pacemaker/corosync services...
Killing any remaining services...
Removing all cluster configuration files...
```

Рисунок 3 – Видалено всі існуючі файли конфігурації кластера.

4. Створюю кластер під назвою `pgcluster`, команда: **`sudo pcs cluster setup --force pgcluster db-server addr=192.168.0.188 node2 addr=192.168.0.114 node3 addr=192.168.0.133`**.

```
kyrylo@db-server:~$ sudo pcs cluster setup --force pgcluster db-server addr=192.168.0.188 node2 addr=192.168.0.114 node3 addr=192.168.0.133
Warning: node3: The host seems to be in a cluster already as the following services are found to be running: 'corosync', 'pacemaker'. If the host is not part of a cluster, stop the services and retry
Warning: node3: The host seems to be in a cluster already as cluster configuration files have been found on the host. If the host is not part of a cluster, run 'pcs cluster destroy' on host 'node3' to remove those configuration files
Warning: node2: The host seems to be in a cluster already as the following services are found to be running: 'corosync', 'pacemaker'. If the host is not part of a cluster, stop the services and retry
Warning: node2: The host seems to be in a cluster already as cluster configuration files have been found on the host. If the host is not part of a cluster, run 'pcs cluster destroy' on host 'node2' to remove those configuration files
Destroying cluster on hosts: 'db-server', 'node2', 'node3'...
db-server: Successfully destroyed cluster
node3: Successfully destroyed cluster
node2: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'db-server', 'node2', 'node3'
db-server: successful removal of the file 'pcsd settings'
node3: successful removal of the file 'pcsd settings'
node2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'db-server', 'node2', 'node3'
db-server: successful distribution of the file 'corosync authkey'
db-server: successful distribution of the file 'pacemaker authkey'
node3: successful distribution of the file 'corosync authkey'
node3: successful distribution of the file 'pacemaker authkey'
node2: successful distribution of the file 'corosync authkey'
node2: successful distribution of the file 'pacemaker authkey'
Sending 'corosync.conf' to 'db-server', 'node2', 'node3'
db-server: successful distribution of the file 'corosync.conf'
node3: successful distribution of the file 'corosync.conf'
node2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
```

Рисунок 4 – Створення кластеру.

5. Запускаю кластер та налаштовую автоматичний запуск кластера під час запуску системи:

```
sudo systemctl enable corosync.service  
sudo systemctl enable pacemaker.service  
sudo pcs cluster enable --all
```

```
kyrylo@db-server:~$ sudo systemctl enable corosync.service  
[sudo] password for kyrylo:  
Sorry, try again.  
[sudo] password for kyrylo:  
Synchronizing state of corosync.service with SysV service script with /lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable corosync  
Created symlink /etc/systemd/system/multi-user.target.wants/corosync.service → /lib/systemd/system/corosync.service.  
kyrylo@db-server:~$ sudo systemctl enable pacemaker.service  
Synchronizing state of pacemaker.service with SysV service script with /lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable pacemaker  
Created symlink /etc/systemd/system/multi-user.target.wants/pacemaker.service → /lib/systemd/system/pacemaker.service.  
kyrylo@db-server:~$ sudo pcs cluster enable --all  
db-server: Cluster Enabled  
node2: Cluster Enabled  
node3: Cluster Enabled  
kyrylo@db-server:~$
```

Рисунок 5 – Запуск кластеру та налаштування.

6. Щоб дізнатися про статус кластера, використовую команду: **sudo pcs cluster status**.

```
kyrylo@db-server:~$ sudo pcs cluster start --all  
node3: Starting Cluster...  
node2: Starting Cluster...  
db-server: Starting Cluster...  
kyrylo@db-server:~$ sudo pcs cluster status  
Cluster Status:  
Cluster Summary:  
* Stack: unknown  
* Current DC: NONE  
* Last updated: Wed May 17 19:25:46 2023  
* Last change: Wed May 17 19:25:41 2023 by hacluster via crmd on db-server  
* 3 nodes configured  
* 0 resource instances configured  
Node List:  
* Node db-server: UNCLEAN (offline)  
* Node node2: UNCLEAN (offline)  
* Node node3: UNCLEAN (offline)  
  
PCSD Status:  
db-server: Online  
node2: Online  
node3: Online
```

Рисунок 6 – Старт кластеру та перевірка його статусу.

7. Щоб перевірити синхронізацію вузлів кластера, використовую команду:

```
[kyrylo@db-server:~$ sudo corosync-cmapctl | grep members
runtime.members.1.config_version (u64) = 0
runtime.members.1.ip (str) = r(0) ip(192.168.0.188)
runtime.members.1.join_count (u32) = 1
runtime.members.1.status (str) = joined
runtime.members.2.config_version (u64) = 0
runtime.members.2.ip (str) = r(0) ip(192.168.0.114)
runtime.members.2.join_count (u32) = 1
runtime.members.2.status (str) = joined
runtime.members.3.config_version (u64) = 0
runtime.members.3.ip (str) = r(0) ip(192.168.0.133)
runtime.members.3.join_count (u32) = 1
runtime.members.3.status (str) = joined
```

Рисунок 7 – Перевірка синхронізації вузлів кластеру.

8. Для відстеження стану кластера використовую команду: **sudo crm_mon -Afr**.

```
Cluster Summary:
* Stack: corosync
* Current DC: db-server (version 2.1.2-ada5c3b36e2) - partition with quorum
* Last updated: Wed May 17 19:40:21 2023
* Last change: Wed May 17 19:26:02 2023 by hacluster via crmd on db-server
* 3 nodes configured
* 0 resource instances configured

Node List:
* Online: [ db-server node2 node3 ]

Full List of Resources:
* No resources
```

Рисунок 8 – Відстеження стану кластера.

Налаштування PostgreSQL для кластеризації

Для налаштування PostgreSQL виконую такі кроки:

1. Вимикаю запуск postgresql.service під час завантаження системи на кожному вузлі кластера. Вмикати та вимикати сервіс за потреби тепер буде Pacemaker: **sudo systemctl disable postgresql.service**.

```
kyrylo@db-server:~$ sudo systemctl disable postgresql.service
Synchronizing state of postgresql.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable postgresql
Removed /etc/systemd/system/multi-user.target.wants/postgresql.service.
```

Рисунок 9 – Приклад вимикання запуску postgresql.service.

2. На вузлі db-server, який спочатку буде мастером, ініціалізую нову базу даних:
/usr/lib/postgresql/14/bin/initdb -D /var/lib/postgresql/14/main

```

kyrylo@db-server:~$ /usr/lib/postgresql/14/bin/initdb -D /var/lib/postgresql/14/main
The files belonging to this database system will be owned by user "kyrylo".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/14/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main -l logfile start

```

Рисунок 10 – Ініціалізація бази даних.

3. Виявив помилку, що на порту «5432», який використовується для підключення до PostgreSQL, вже зайнятий іншим процесом. Виправлення показано на рисунку номер 11.

```

kyrylo@db-server:~$ sudo lsof -i :5432
COMMAND  PID  USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
postgres 4023 postgres 5u  IPv4  31195      0t0  TCP *:postgresql (LISTEN)
postgres 4023 postgres 6u  IPv6  31196      0t0  TCP *:postgresql (LISTEN)
kyrylo@db-server:~$ kill 4023
-bash: kill: (4023) - Operation not permitted
kyrylo@db-server:~$ sudo kill 4023
kyrylo@db-server:~$ sudo lsof -i :5432

```

Рисунок 11 – Виправлення помилки з портом.

4. Запускаю базу командою:

`/usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main start.`

```

postgres@db-server:~$ /usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main start
waiting for server to start...2023-05-17 21:29:03.112 UTC [23165] LOG:  starting PostgreSQL 14.7 (Ubuntu 14.7-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.3.0-1ubuntu1-22.04) 11.3.0, 64-bit
2023-05-17 21:29:03.113 UTC [23165] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2023-05-17 21:29:03.117 UTC [23165] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2023-05-17 21:29:03.126 UTC [23166] LOG:  database system was shut down at 2023-05-17 21:24:23 UTC
2023-05-17 21:29:03.133 UTC [23165] LOG:  database system is ready to accept connections
done
server started

```

Рисунок 11 – Запуск бази даних.

5. Створюю користувача для реплікації:

`createuser --replication --createdb --username=postgres --no-createrole --no-superuser repl.`

6. Змінюю файл `/var/lib/postgresql/14/main/pg_hba.conf` та додаю до нього необхідні дозволи наступним чином:

`host replication all 192.168.0.0/24 trust`


```
host all all 192.168.0.0/24 trust
```

7. Змінюю файл `/var/lib/postgresql/14/main/postgresql.conf` та додаю до нього наступні рядки:

```
listen_addresses = '*'
wal_level = replica
logging_collector = on
hot_standby = on lc_messages = 'C'
wal_keep_size = 10
```

8. Перезапускаю СУБД PostgreSQL:

```
/usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main stop
/usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main start
```

```
postgres@db-server:~$ /usr/lib/postgresql/14/bin/pg_ctl -D /var/lib/postgresql/14/main start
waiting for server to start....2023-05-17 21:52:21.536 UTC [23273] LOG:  redirecting log output to logging co
llector process
2023-05-17 21:52:21.536 UTC [23273] HINT:  Future log output will appear in directory "log".
done
server started
```

Рисунок 11 – Сервер перезапущено.

9. На інших двох вузлах (node2 та node3) виконайте такі дії:

1. Зупиняю процес postgresql: **sudo systemctl stop postgresql**
2. Чищу каталог `/var/lib/postgresql/14/main`: **sudo rm -rf /var/lib/postgresql/14/main/***
3. Скопіюйте базу даних із майстра (node1) за допомогою команди:
pg_basebackup -D /var/lib/postgresql/14/main -h 192.168.0.188 -X stream

Налаштування ресурсів кластера

Після налаштування СУБД PostgreSQL на всіх вузлах кластера слідє створити ресурси кластера. Для цього виконую такі дії:

1. Створюю ресурс з ім'ям `virtual_ip` типу `IPAddr2`, який використовуватиметься для підключення до бази даних PostgreSQL:
sudo pcs resource create virtual_ip IPAddr2 ip="192.168.0.204" cidr_netmask="24" meta migration-threshold="0" op monitor timeout="60s" interval="10s" on-fail="restart" op stop timeout="60s" interval="0s" on-fail="ignore" op start timeout="60s" interval="0s" on-fail="stop"
2. Створюю ресурс з ім'ям `my-pgsql` типу `pgsql` для керування конфігурацією PostgreSQL:
**sudo pcs resource create my-pgsql pgsql
pgctl="/usr/lib/postgresql/14/bin/pg_ctl"
psql="/usr/lib/postgresql/14/bin/psql"
pgdata="/var/lib/postgresql/14/main" rep_mode="sync"
node_list="db-server node2 node3" master_ip="192.168.0.204"
restart_on_promote="false" check_wal_receiver="true"**

pgport="5432"
repuser="repl"

primary_conninfo_opt="password=12345"

3. Для створеного ресурсу my-pgsql вказую, що він може мати один із кількох станів та змінювати їх залежно від типу вузла (master та slave):

**sudo pcs resource promotable my-pgsql promoted-max=1
promotednode-max=1 clone-max=3 clone-node-max=1 notify=true**

4. Зв'язую два створених ресурси, щоб вони запускалися разом на одному вузлі, і встановлюю черговість запуску таким чином, щоб ресурс virtual_ip запускався лише після успішного запуску ресурсу my-pgsql. Для цього створюю групу ресурсів master-group і додаю до неї ресурси:

sudo pcs resource group add master-group virtual_ip

**sudo pcs constraint colocation add master-group with Master
mypgsql-clone**

**sudo pcs constraint order promote my-pgsql-clone then start master-
group symmetrical=false score=INFINITY**

**sudo pcs constraint order demote my-pgsql-clone then stop master-
group symmetrical=false score=0**

Для захисту ресурсів, що розділяються, і ізоляції вузла кластера при його несправності використовую механізм фенсингу.

Щоб вивести список доступних fence-agents, використовую команду:

sudo pcs stonith list

Щоб вивести необхідні налаштування для агента fence_virsh використовую команду:

sudo pcs stonith describe fence_virsh

Висновок:

У цій лабораторній роботі я ознайомився з багаторівневими архітектурами на прикладі кластеру бази даних і реалізував масштабування горизонтально окремих компонентів системи та відмовостійкість системи.

Масштабування горизонтально дозволяє розподілити навантаження між багатьма серверами, що дає можливість збільшити продуктивність системи і підвищити її масштабованість. Реалізація відмовостійкості системи дозволяє системі продовжувати свою роботу, коли виникають непередбачувані помилки, такі як збої в обладнанні або програмному забезпеченні.

Отже, я успішно навчився використовувати багаторівневі архітектури та забезпечувати відмовостійкість та масштабованість системи, що є важливими функціональними вимогами до багатьох сучасних програмних систем.