**Baze de date distribuite in PostgreSQL**

1. **Descrierea tehnologiilor alese**
   1. **PostgreSQL**

“PostgreSQL este un sistem de gestiune de baze de date relaționale avansat, de tip enterprise și open source. PostgreSQL acceptă atât interogări SQL (relaționale), cât și JSON (non-relaționale). Este extrem de stabil si este utilizat pentru multe aplicații web, precum și aplicații mobile și analitice.”[8]

* 1. **Replicarea**

“Definiția generică a replicării este actul de copiere sau reproducere a unei entități astfel încât să existe mai multe copii. Conceptul de replicare în baze de date este strâns legat de definiția standard. Replicarea bazei de date este procesul de partajare a informațiilor între două sau mai multe baze de date.”[9]

“Replicarea ar putea avea loc între bazele de date găzduite pe aceeași mașină fizică sau în întreaga rețea. Baza de date care acționează ca nod activ pe care apar modificările inițiale este cunoscută ca nod principal / master / primar, în timp ce celelalte baze de date care primesc modificările ulterior (efectiv, copiază datele), sunt cunoscute ca nod standby / slave / secundar.”[9]

* 1. **Avantajele replicării**

“Există trei motive principale pentru care bazele de date sunt reproduse, și anume - disponibilitatea ridicată, echilibrarea traficului de date și recuperarea în caz de dezastru. Disponibilitatea ridicată se referă la posibilitatea de a avea în permanență o copie actualizată a bazei de date. Aceasta înseamnă că, în cazul unei defecțiuni a bazei de date principale, copia de standby poate fi promovată la master și se poate începe primirea traficului.”[9]

“Echilibrarea traficului de date se referă la practica de distribuire a cererilor primite către aplicație într-un mod echilibrat, astfel încât nicio bază de date anume să nu experimenteze o sarcină de lucru inegală. Cu replicarea, acest lucru este posibil, deoarece există mai multe copii ale datelor în orice moment de timp.”[9]

“Motivul final al replicării este necesitatea unei recuperări eficiente în caz de dezastru în cazul unei defecțiuni de sistem.”[9]

* 1. **Modele de replicare**

“O altă clasificare a sistemelor de replicare a bazelor de date este replicarea unui *singur master* versus replicarea *multi-master*. Într-o singură configurare principală, există un singur nod activ care poate primi tranzacții de scriere și citire. Toate celelalte noduri pot efectua numai operații de citire. Nodul principal este singurul nod pe care se pot efectua operațiuni de scriere, apoi propagă tranzacția către alte noduri.”[9]

“Pe de altă parte, într-o replicare multi-master, mai multe noduri pot primi tranzacții de scriere / citire. Chiar dacă acest lucru înseamnă că scrierile pot fi realizate mai repede, introduce un nivel de complexitate în configurare. Conflictele între diferite nodurile bazei de date trebuie tratate cu grijă, deoarece scrierile pot avea loc oriunde și, prin urmare, datele trebuie sincronizate.”[9]

“Replicarea multi-master adaugă la următoarele:” [10]

* “disponibilitatea scrierii și scalabilitatea” [10]
* “Abilitatea de a utiliza o rețea extinsă (WAN) de baze de date master care pot fi apropiate geografic de grupuri de clienți, dar păstrează coerența datelor în întreaga rețea.”[10]
  1. **Clase de replicare**

“Replicarea single-master este, de asemenea, numită *unidirecțională*, deoarece datele de replicare circulă într-o singură direcție, de la master la replică.” [10]

“Datele de replicare multi-master, pe de altă parte, circulă în ambele direcții și, prin urmare, se numește replicare *bidirecțională*.” [10]

* 1. **Moduri de replicare**

“În *replicarea sincronă*, atunci când nodul principal primește o tranzacție de scriere, acesta propagă această modificare la toate nodurile de așteptare și primește o confirmare înainte ca tranzacția să fie permanentă sau marcată ca finalizată. În acest fel, datele sunt identice în toate nodurile și se spune că sunt consistente. În timp ce, în *replicarea asincronă*, atunci când nodul principal primește o tranzacție, continuă să finalizeze tranzacția înainte de a propaga această modificare în nodurile de așteptare. O face cu așteptarea ca modificările propagate către nodurile de așteptare să fie comise, dar nu așteaptă o confirmare.”[9]

“Aceasta înseamnă că va exista întotdeauna un decalaj între nodul principal și standby. De asemenea, este important de reținut că există o scurtă fereastră în care pierderea datelor ar putea apărea dacă nodul principal eșuează înainte ca acesta să poată propaga modificările către standby-uri, cu excepția cazului în care este implementat un mecanism special de eșec pentru a garanta coerența eventuală, dar însăși natura replicării asincrone înseamnă că posibilitatea pierderii datelor într-un astfel de scenariu nu poate fi exclusă complet.”[9]

“Teoretic, replicarea sincronă are avantajul de a fi consecventă, dar în practică, este de obicei costisitoare de implementat și, în multe cazuri, de-a dreptul imposibil. Din acest motiv, majoritatea proceselor de replicare a bazelor de date adoptă abordarea asincronă.”[9]

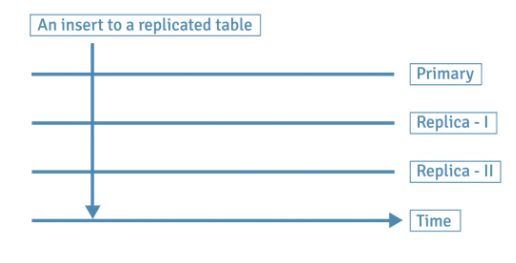


Figura 1.3: Replicarea sincronă[10]

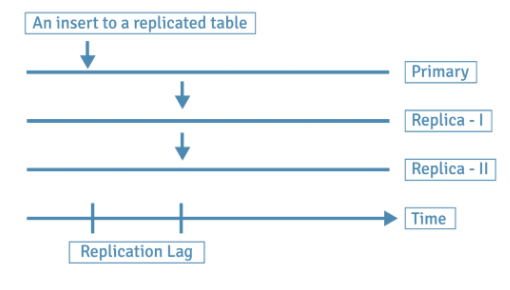


Figura 1.2: Replicarea asincronă[10]

* 1. **Tipuri de replicare în cazul PostgreSQL**

Pe baza diagramei de mai jos se vor prezenta principalele 3 tipuri de replicări din PostgreSQL: replicarea logică, replicarea fizică și replicarea streaming sau în flux pentru care se va face și o implementare ce urmează să fie detaliată în capitolul 2.



Figura 1.3: Tipuri de replicare în PostgreSQL[7]

* “Replicarea logică, pe de altă parte, se ocupă de baze de date, tabele și operațiuni DML(SELECT, INSERT, UPDATE, DELETE etc). Prin urmare, în replicarea logică este posibil să se reproducă doar un anumit set de tabele. Replicarea logică se face la nivelul clusterului bazei de date.”[10] “Replicarea logică în PostgreSQL se realizează prin construirea unui flux de modificări de date logice din fișierul WAL. Nu necesită o anumită bază de date pentru a fi desemnată ca bază de date principală, dar acceptă fluxul de date multidirecțional. Replicarea logică în PostgreSQL poate fi utilizată pentru a copia tabele individuale în loc de o bază de date întreagă. Trebuie reținut totuși că, deoarece acceptă fluxul de date în mai multe direcții, ar putea apărea conflicte și ar trebui să existe o procedură pentru a gestiona aceste evenimente.”[9]
* “Replicarea fizică se ocupă de fișiere și directoare. Nu știe ce reprezintă în sine aceste fișiere și directoare. Replicarea fizică se face la nivel de sistem de fișiere sau la nivel de disc.”[10]
* “În PostgreSQL, replicarea în flux permite bazelor de date de așteptare să fie cât mai aproape de baza de date principală, deoarece folosește înregistrările de jurnal Write-Ahead (WAL) care transformă fluxul către bazele standby în format binar, deoarece este generat în loc să aștepte fișierul WAL să fie completat. Replicarea în flux este deosebit de utilă în configurarea unui sistem cu disponibilitate ridicată, deoarece failover-ul poate fi realizat cu ușurință deoarece clusterele de baze de date sunt identice.”[9]
  1. **Write-Ahead Logging (WAL)**

“În PostgreSQL, toate modificările efectuate de o tranzacție sunt salvate mai întâi într-un fișier jurnal(log), iar apoi rezultatul tranzacției este trimis clientului inițiator. Fișierele de date în sine nu sunt modificate la fiecare tranzacție. Acesta este un mecanism standard pentru a preveni pierderea datelor în cazul unor situații precum blocarea sistemului de operare, eșecul hardware sau blocarea PostgreSQL Acest mecanism se numește Write Ahead Logging (WAL), iar fișierul jurnal se numește Write Ahead Log.”[10]

“Fiecare modificare efectuată de tranzacție (INSERT, UPDATE, DELETE, COMMIT) este scrisă în jurnal ca o înregistrare WAL. Înregistrările WAL sunt scrise mai întâi într-un buffer WAL în memorie. Când tranzacția este efectuată, înregistrările sunt scrise într-un fișier segment WAL de pe disc.”[10]

“Numărul de secvență de jurnal (LSN) al unei înregistrări WAL reprezintă locația / poziția în care este salvată înregistrarea în fișierul jurnal. LSN este utilizat ca un id unic al înregistrării WAL. În mod logic, un jurnal de tranzacții este un fișier a cărui dimensiune este de 2 ^ 64 octeți. Prin urmare, LSN este un număr pe 64 de biți reprezentat ca două numere hexazecimale pe 32 de biți separate printr-o /.”[10]

“Dacă există un blocaj de sistem, baza de date poate recupera tranzacțiile efectuate de la WAL. Recuperarea începe de la ultimul punct REDO sau punct de control. Un punct de control este un punct din jurnalul de tranzacții la care toate fișierele de date au fost actualizate pentru a se potrivi cu informațiile din jurnal. Procesul de salvare a înregistrărilor WAL din fișierul jurnal în fișierele de date reale se numește check-pointing.”[10]

1. **Descrierea implementării practice**
   1. **Instalare**

* Se începe prin a descărca și instala VirtualBox-ul de la Oracle folosind link-ul furnizat in referința [1]
* Se continuă cu descărcarea și instalarea distribuției Kubuntu folosind referințele [2], [4], precum și distribuției Lubuntu folosind referințele [3], [5] (aceste două distribuții au fost alese datorită faptului că deși ele sunt bazate pe Ubuntu, consumă mult mai puține resurse decât acesta).
* Urmează instalarea sistemului de gestiune de baze de date PostgreSQL în ambele distribuții conform referinței [6]

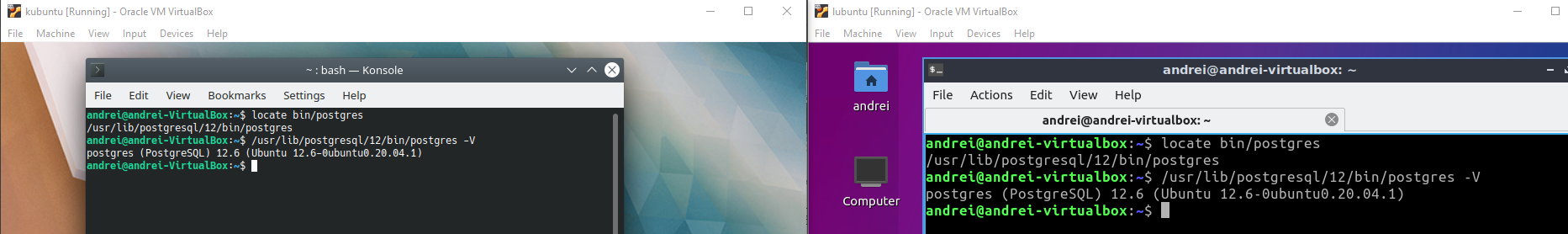


Figura 2.1: Instalarea cu succes a sistemului de gestiune de baze de date PostgreSQL pe ambele distribuții

* 1. **Configurare**
* Se începe prin a verifica adresa IP a fiecărei mașini virtuale ca în figură:

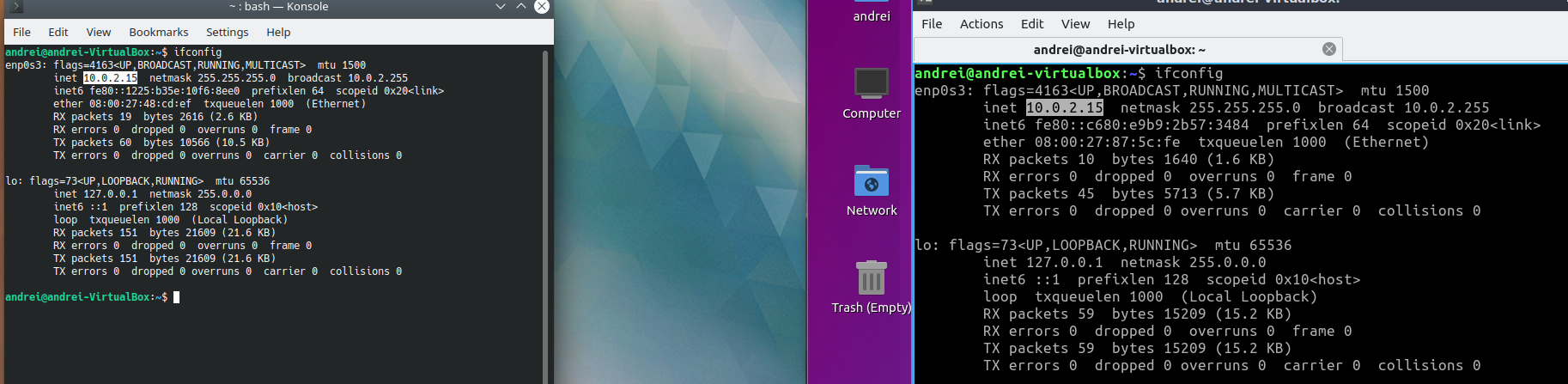
****

Figura 2.2: Adresa IP implicită 10.0.2.15 comună ambelor mașini virtuale

* Se observă în figura de mai sus adresa IP 10.0.2.15 care este pusă inițial de VirtualBox(asta se datorează faptului că în setările de la VirtualBox la secțiunea Network pentru fiecare mașină virtuală tipul de adaptor de rețea este NAT, care pune fiecare mașină virtuală în propria rețea, izolată de celelalte). Practic ca cele 2 mașini virtuale să comunice între ele este nevoie să fie schimbate setările în așa fel încât să se afle în aceeași rețea.
* Se începe prin a merge din meniul de la VirtualBox astfel: File-Preferences-Network-se dă click pe semnul + din dreapta pentru a crea o nouă rețea ca în figură

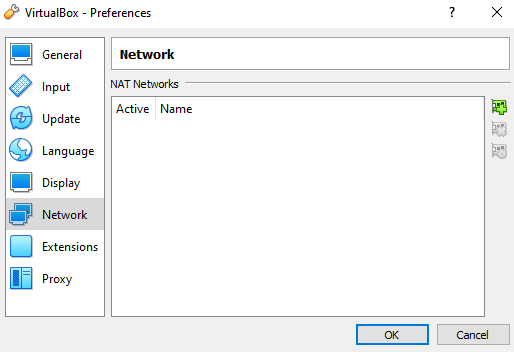
****

Figura 2.3: Opțiunea de unde se poate crea o nouă rețea în VirtualBox

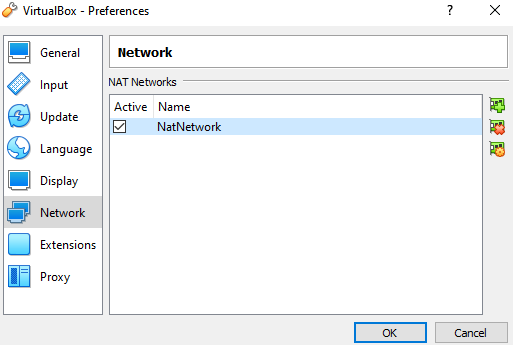
****

Figura 2.4: Crearea unei noi rețele în VirtualBox

* Se dă click dreapta-Edit Nat Network și se schimbă numele în BdasNetwork, se pune adresa IP 192.168.100.0 cu masca /24 și se lasă bifată opțiunea Supports DHCP pentru ca mașinile virtuale să poată primi ulterior adrese IP.

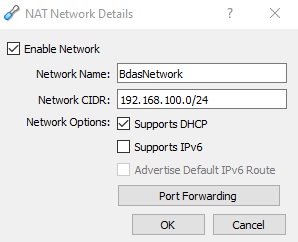


Figura 2.5: Editarea noii rețele în VirtualBox

* Urmează plasarea în aceeași rețea a ambelor mașini virtuale mergând pentru fiecare distribuție în parte la Settings-Network ca în figurile de mai jos:

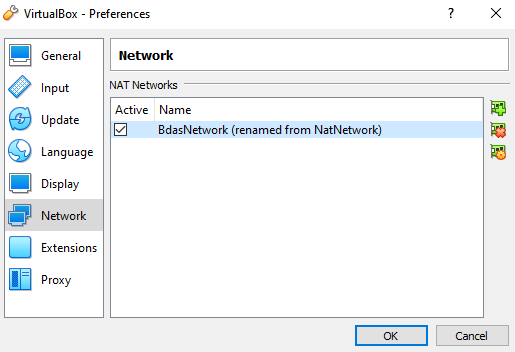


Figura 2.6: Rețeaua care va fi folosită pentru a conecta cele 2 mașini virtuale

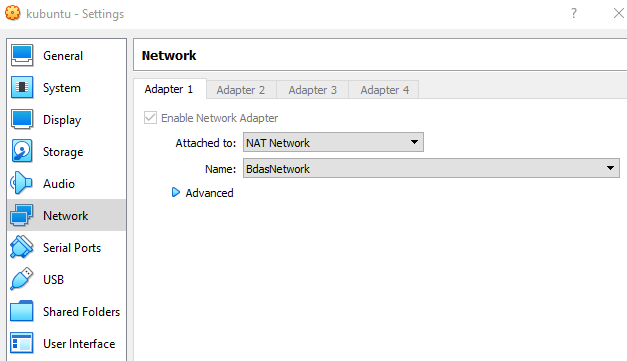


Figura 2.7: Plasarea mașinii virtuale kubuntu în BdasNetwork

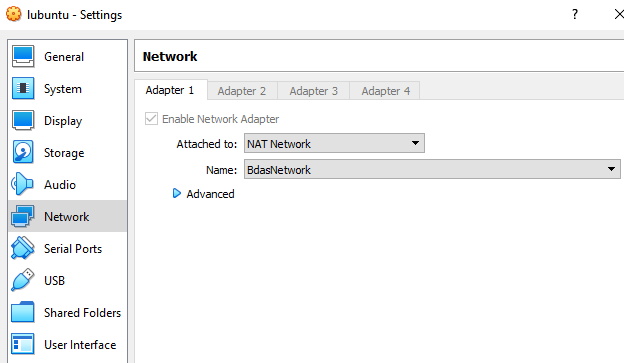


Figura 2.8: Plasarea mașinii virtuale lubuntu în BdasNetwork

* Se verifică folosind comanda *ifconfig* adresele IP ale fiecărei mașini virtuale și se observă că au primit adresele 192.168.100.4 pentru kubuntu și 192.168.100.5 pentru lubuntu ca în figurile următoare:

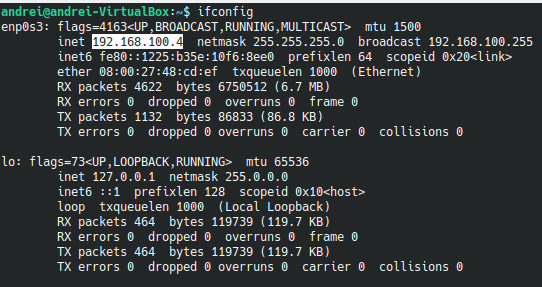


Figura 2.9: Adresa IP a mașinii virtuale kubuntu

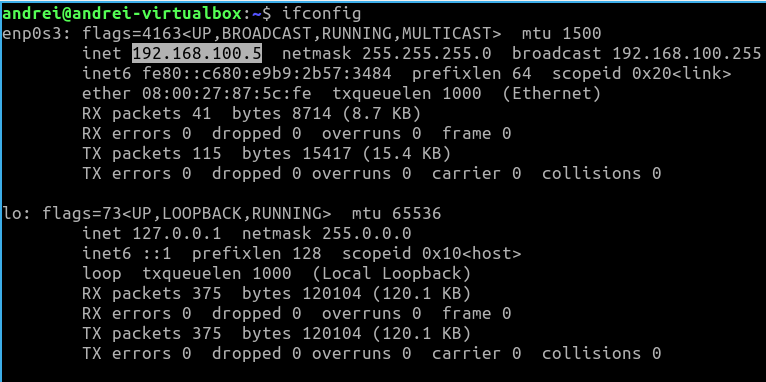


Figura 2.10: Adresa IP a mașinii virtuale lubuntu

* Se verifică folosind comanda *ping* împreună cu adresa IP a mașinii virtuale dorite faptul că cele 2 mașini pot comunica între ele și se confirmă acest lucru conform figurilor următoare:

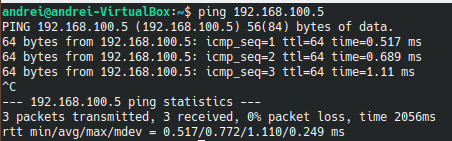


Figura 2.11: Transmiterea cu succes pachetelor de la kubuntu la lubuntu

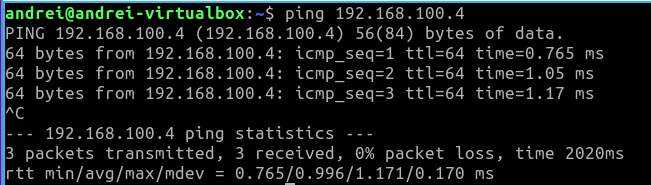


Figura 2.12: Transmiterea cu succes pachetelor de la lubuntu la kubuntu

Legătura dintre cele două mașini virtuale s-a făcut folosind referința [11].

* 1. **Implementare replicare streaming**

Se va folosi mașina virtuală kubuntu ca nod master și mașina virtuală lubuntu ca nod slave.

* Se începe prin a verifica existența cluster-ului PostgreSQL pe ambele mașini virtuale folosind comanda *sudo pg\_lsclusters* [12] [13]

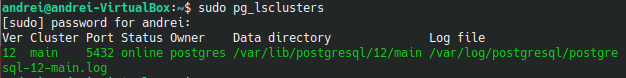


Figura 2.13: Verificarea cluster-ului pe mașina virtuală kubuntu

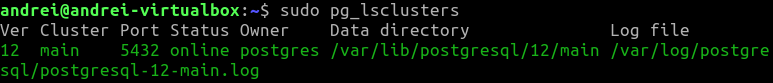


Figura 2.14: Verificarea cluster-ului pe mașina virtuală lubuntu

* Urmează crearea de directoare pentru arhivare a fișierelor WAL, folosind comanda *sudo -H -u postgres mkdir /var/lib/postgresql/pg\_log\_archive/main -p* care vor fi generate datorită replicării. Se dorește arhivarea lor deoarece ele vor fi produse în mod constant, lucru care în timp duce la umplerea memoriei.[12] [13]



Figura 2.15: Crearea directorului de arhivare pe mașina virtuală kubuntu



Figura 2.16: Crearea directorului de arhivare pe mașina virtuală lubuntu

* Se face copierea fișierului postgresql.conf pentru motive de siguranță [14][15]

Text

Description automatically generated

Figura 2.17: Copierea fișierului postgresql.conf pe mașina virtuală kubuntu

Text

Description automatically generated

Figura 2.18: Copierea fișierului postgresql.conf pe mașina virtuală lubuntu

* Se continuă cu editarea fișierului de configurare pe master(Kubuntu) folosind comanda *sudo nano /etc/postgresql/12/main/postgresql.conf* [12] [13]

1. Se decomentează listen\_addresses eliminând # și se pune în loc de localhost \* pentru a se aștepta conexiuni de la toți clienții [14] [15]

Graphical user interface, text, chat or text message

Description automatically generated

Figura 2.19: Decomentarea liniei listen\_addresses

Text

Description automatically generated

Figura 2.20: Ascultarea tuturor clienților

1. Setarea parametrului wal\_level pe replica care scrie suficiente date pentru a permite arhivarea și replicarea[16]

Graphical user interface, text, application

Description automatically generated

Figura 2.21: Setarea parametrului wal\_level pe replica

1. Activarea modului de arhivare + comanda de arhivare

Text

Description automatically generated

Figura 2.22: Activarea modului de arhivare + comanda de arhivare

1. Setarea numărului maxim de conexiuni simultane de la servere de așteptare sau clienți de rezervă de bază de streaming[16] + setarea numărului de segmente WAL care să fie păstrate de master în cazul unei deconectări temporare între master-slave astfel ca în momentul în care se reconectează replicarea să se poată relua de unde a rămas[12]

Text

Description automatically generated

Figura 2.23: Setarea numărului maxim de conexiuni simultane + setarea numărului de segmente WAL

1. Se creează utilizatorul de replicare cu *sudo -H -u postgres psql -c " CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD 'admin'; "*



Figura 2.24: Crearea utilizatorului de replicare

Text

Description automatically generated

Figura 2.25: Verificarea utilizatorului de replicare

1. Se editează fișier de acces bazat pe gazdă pg\_hba.conf folosind *sudo nano /etc/postgresql/12/main/pg\_hba.conf*  pentru a înregistra utilizatorul de replicare unde se va pune și adresa IP a slave-ului(Lubuntu)

Graphical user interface, text

Description automatically generated

Figura 2.26: Înregistrarea utilizatorului de replicare

1. Se restartează master-ul pentru a putea fi folosite configurările anterioare folosind *sudo systemctl restart postgresql@12-main*

Graphical user interface, text

Description automatically generated

Figura 2.26: Restartarea master-ului

* Se oprește slave-ul folosind *sudo systemctl stop postgresql@12-main*

Text

Description automatically generated

Figura 2.27: Oprirea slave-ului

* Se face o copie a directorului main din slave

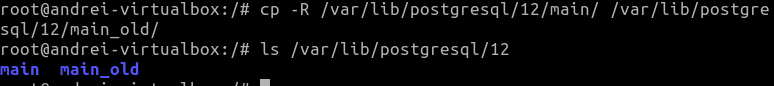


Figura 2.28: Copierea main-ului din slave

* Se face editarea fișierului de configurare postgresql.conf pe slave(Lubuntu) la fel ca la master
* Se face editarea fișierului de acces bazat pe gazdă pg\_hba.conf pe slave la fel ca la master
* Se șterge conținutul folder-ului main de pe slave cu utilizatorul postgres folosind *rm -rf /var/lib/postgresql/12/main/*



Figura 2.29: Ștergerea folder-ului main de pe slave

* Se va face sincronizarea slave-ului cu master-ul folosind *pg\_basebackup -h 192.168.100.4 -D /var/lib/postgresql/12/main/ -U replicator -P -v -R -X stream -C -S slaveslot1* și se furnizează parola de la utilizatorul de replicare de pe master

Text

Description automatically generated

Figura 2.30: Sincronizarea slave-ului cu master-ul

* Se verifică sincronizarea cu master-ul(copierea main-ului din master în slave) prin prezența fișierului standby.signal din folderul main din slave( se merge pe ideea că un slave de replicare va rula în modul „Hot Standby” dacă parametrul hot\_standby este activat (valoarea implicită) în postgresql.conf și există un fișier standby.signal prezent în directorul de date) [15]

Text

Description automatically generated

Figura 2.31: Verificare sincronizării cu master-ul

* Se pornește slave-ul

Text

Description automatically generated

Figura 2.32: Pornirea slave-ului

* Se verifică copierea bazelor de date din master în slave

Text

Description automatically generated

Figura 2.32: Bazele de date din master

Text

Description automatically generated

Figura 2.33: Bazele de date din slave

* Se observă copierea cu succes din master în slave
* Se încearcă crearea unei baze de date în slave, lucru care trebuie să eșueze deoarece slave-ul este un server read-only [15]



Figura 2.34: Confirmarea slave-ului ca fiind server read-only

* Se creează o bază de date *stream* pe master care trebuie să fie reflectată și pe slave [15]



Figura 2.35: Crearea bazei de date stream pe master

Text

Description automatically generated

Figura 2.36: Reflectarea bazei de date stream pe slave

* Se verifică pe master prezența lui slaveslot1 care a fost creat cu ajutorul slave-ului folosind *SELECT \* FROM pg\_replication\_slots;* [15]

**A picture containing text

Description automatically generated**

Figura 2.37: Slaveslot1 pe master

* Se verifică faptul că replicarea are loc pe slave folosind *SELECT \* FROM pg\_stat\_wal\_receiver;* [15]

Text

Description automatically generated

Figura 2.38: Realizarea replicării pe slave

* Se verifică tipul replicării pe master, care în acest caz este asincronă(care este de fapt și tipul implicit de replicare pentru replicarea de tip streaming) folosind *SELECT \* FROM pg\_stat\_replication;* [15]

Text

Description automatically generated

Figura 2.39: Replicare asincronă pe master

* Se creează tabelul posts în master și se verifică și în slave [13]

Text

Description automatically generated

Figura 2.40: Crearea tabelului posts pe master

**Text

Description automatically generated**

Figura 2.41: Vizualizarea tabelei posts în slave

**Text

Description automatically generated**

Figura 2.42: Inserare date în posts pe master

Text

Description automatically generated

Figura 2.43: Vizualizare date în slave



Figura 2.44: Update în master

**Text

Description automatically generated**

Figura 2.45: Vizualizarea update-ului în slave



Figura 2.46: Delete pe master

Text

Description automatically generated

Figura 2.47: Vizualizarea delete-ului în slave

1. **Bibliografie**

[1] *Download VirtualBox*, https://www.virtualbox.org/wiki/Downloads, accesat la data: 14.04.2021

[2] *Download Kubuntu*, <https://kubuntu.org/getkubuntu/>, accesat la data: 14.04.2021

[3] *Lubuntu*, <https://lubuntu.me/downloads/>, accesat la data: 14.04.2021

[4] *How to install Kubuntu on virtual box*, <https://www.youtube.com/watch?v=ZLlvtJ0eEYg&t=534s>, accesat la data: 14.04.2021

[5] *How to install Lubuntu on Virtualbox*, <https://www.youtube.com/watch?v=tOfmkv_Cqww>, accesat la data: 14.04.2021

[6*] How to Install PostgreSQL on Ubuntu Linux*, <https://www.youtube.com/watch?v=VNy2nhho9Pg>, accesat la data: 14.04.2021

[7] Streaming replication with PostgreSQL 9.6, 10 & 11 - PostgreSQL Standby Databases, https://www.sqlpac.com/en/documents/postgresql-9.6-10-11-configuration-setup-streaming-replication-standby-ubuntu.html, accesat la data: 14.04.2021

[8] *What is PostgreSQL?*, <https://www.postgresqltutorial.com/what-is-postgresql/>, accesat la data: 14.04.2021

[9] *Postgres Replication Types: Comprehensive Analysis*, <https://hevodata.com/learn/postgres-replication-types/#psql>, accesat la data: 14.04.2021

[10] PostgreSQL Replication and Automatic Failover Tutorial , <https://www.enterprisedb.com/postgres-tutorials/postgresql-replication-and-automatic-failover-tutorial#replication>, accesat la data: 14.04.2021

[11] *How to Make Virtual Machines Talk to Each Other in VirtualBox*, <https://www.youtube.com/watch?v=vReAkOq-59I>, accesat la data: 15.04.2021

[12] *PostgreSQL Streaming Replication*, <https://www.youtube.com/watch?v=NaPnYQBBdyU>, accesat la data: 01.05.2021

[13] *PostgreSQL Streaming Replication*, <https://www.scalingpostgres.com/tutorials/postgresql-streaming-replication/>, accesat la data: 01.05.2021

[14] *49. PostgreSQL DBA: How to setup streaming replication in PostgreSQL step by step on Ubuntu*, <https://www.youtube.com/watch?v=LhPAg583pKc>, accesat la data: 01.05.2021

[15] *How to setup streaming replication in PostgreSQL step by step on Ubuntu*, <https://www.postgresql.r2schools.com/how-to-setup-streaming-replication-in-postgresql-step-by-step-on-ubuntu/>, accesat la data: 01.05.2021

[16] *POSTGRESQLCO.NF*, https://postgresqlco.nf/doc/en/param/wal\_level/, accesat la data: 01.05.2021