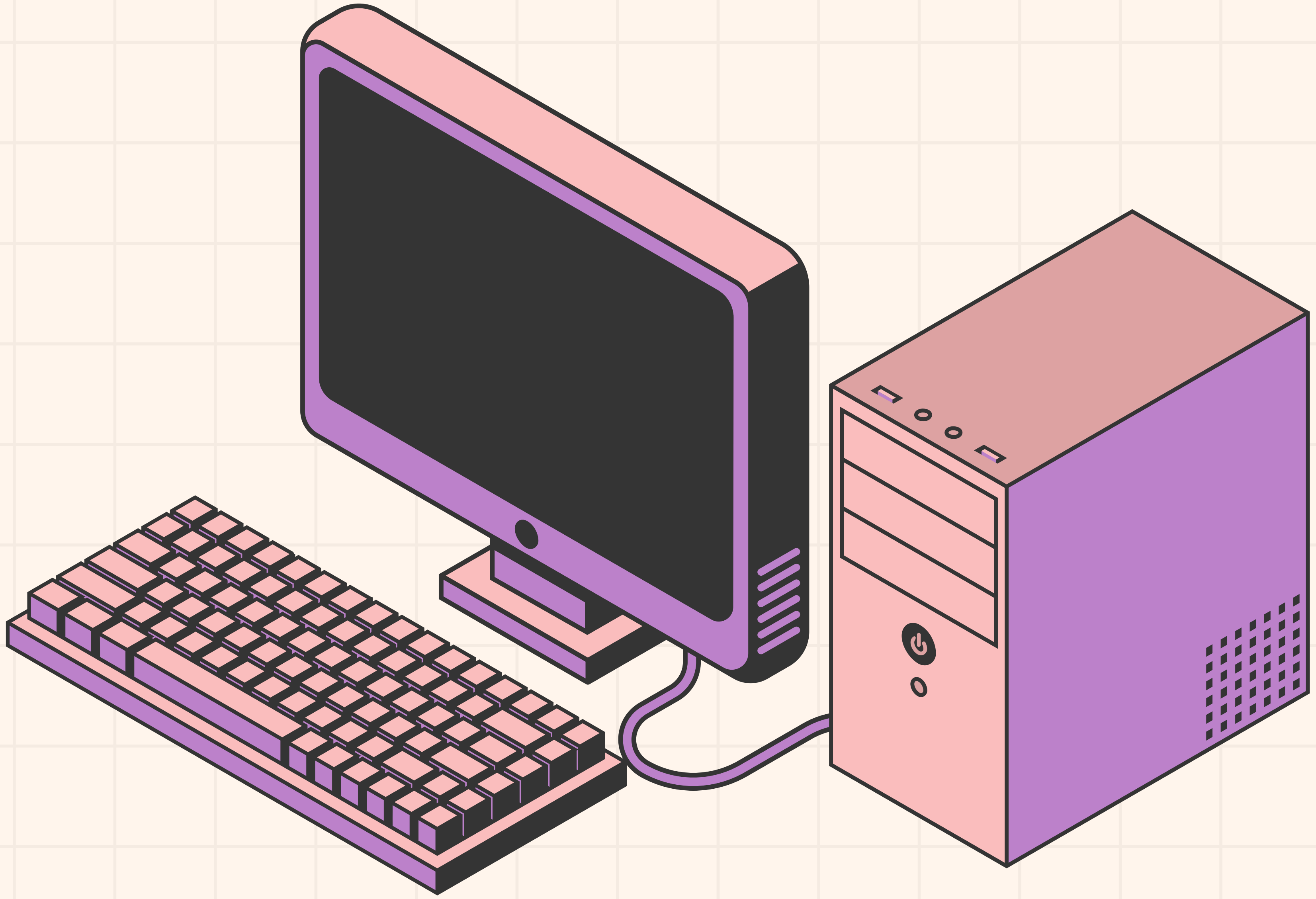
HIGH AVAILABILITY CLUSTER

USING POSTGRESQL SERVER ON AZURE

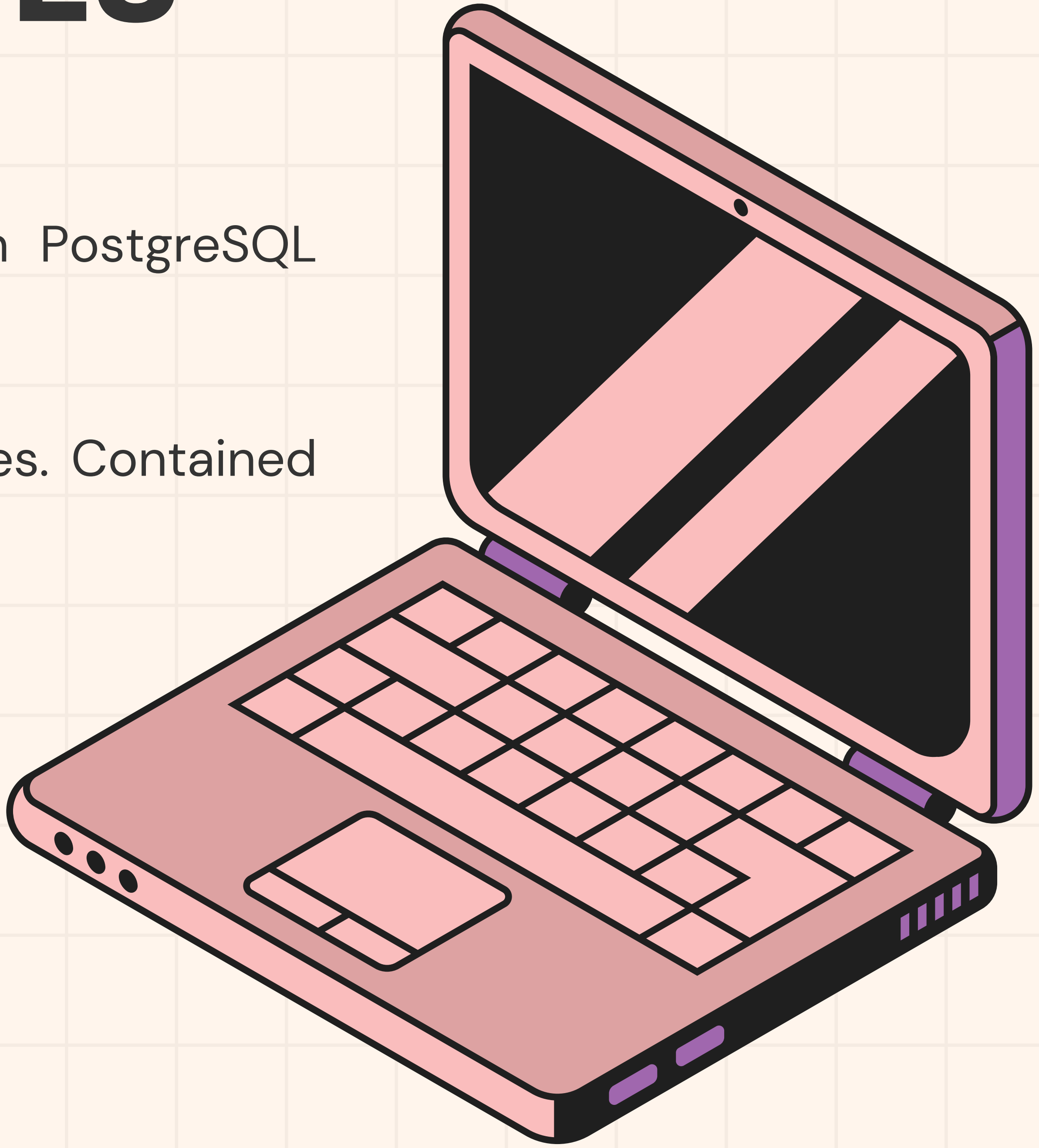


INTRODUCTION & OBJECTIVES

We aimed to explore the implementation of high availability through PostgreSQL database, servers hosted on Microsoft Azure.

Sample db: Imaginary application that keeps track of personal expenses. Contained tables; Expenses, Expenses Categories, Subcategories, Income, Users

- **Deploy a PostgreSQL high availability cluster across multiple Azure VMs**
- **Implement replication of primary with secondary nodes**
- **Test failover scenarios and validate cluster behavior under node failures**
- **Automate regular backups of the database using cron**
- **Demonstrate full restoration of the system from backup**



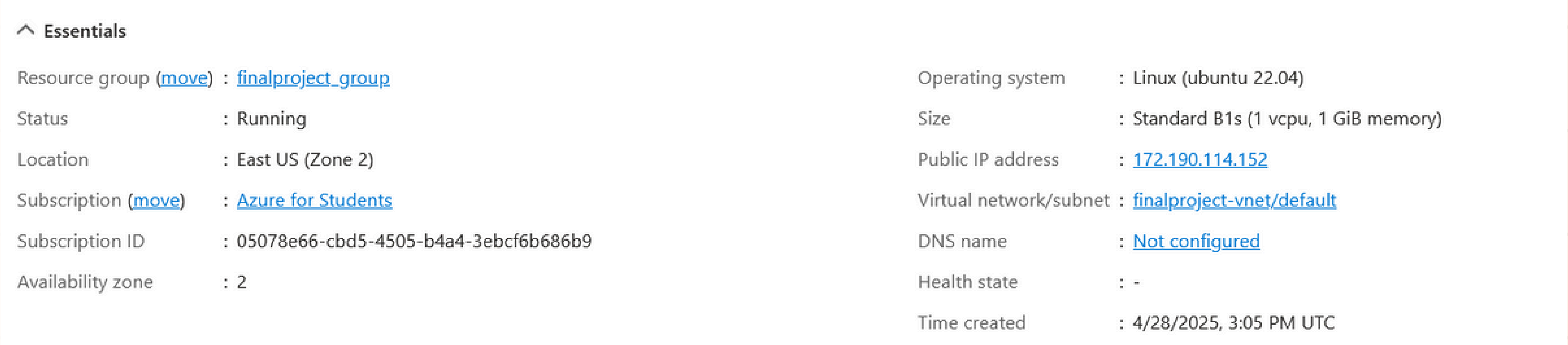
ENVIRONMENT SETUP

VM CREATION

We created 4 virtual machines (VMs) on Microsoft Azure:

- VM1: On Tejas' Azure account (Named: node1)
- VM2 and VM3: On Vaishnavi's Azure Account (Named: secondary-1 and secondary-2)
- A Monitor node (VM4): On Vaishnavi's machine for consensus

All VMs - on Ubuntu 22.04



^ Essentials

Resource group (move) : finalproject_group
Status : Running
Location : East US (Zone 2)
Subscription (move) : Azure for Students
Subscription ID : 05078e66-cbd5-4505-b4a4-3ebcf6b686b9
Availability zone : 2

Operating system : Linux (ubuntu 22.04)
Size : Standard B1s (1 vcpu, 1 GiB memory)
Public IP address : 172.190.114.152
Virtual network/subnet : finalproject-vnet/default
DNS name : Not configured
Health state : -
Time created : 4/28/2025, 3:05 PM UTC

POSTGRESQL INSTALLATION

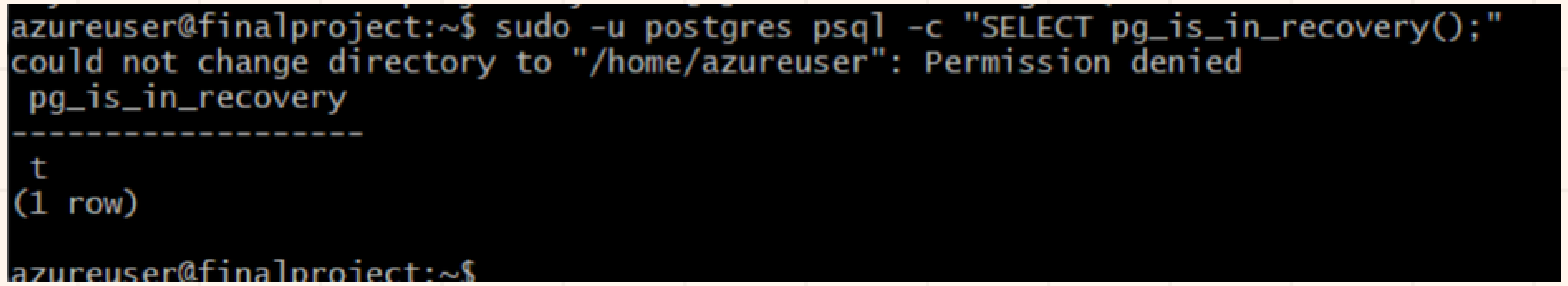
PostgreSQL 14 was installed and configured manually (except the monitoring node).

Configuration involved:

- adjusting postgresql.conf to allow replication and remote access
- pg_hba.conf to accept connections from the other nodes

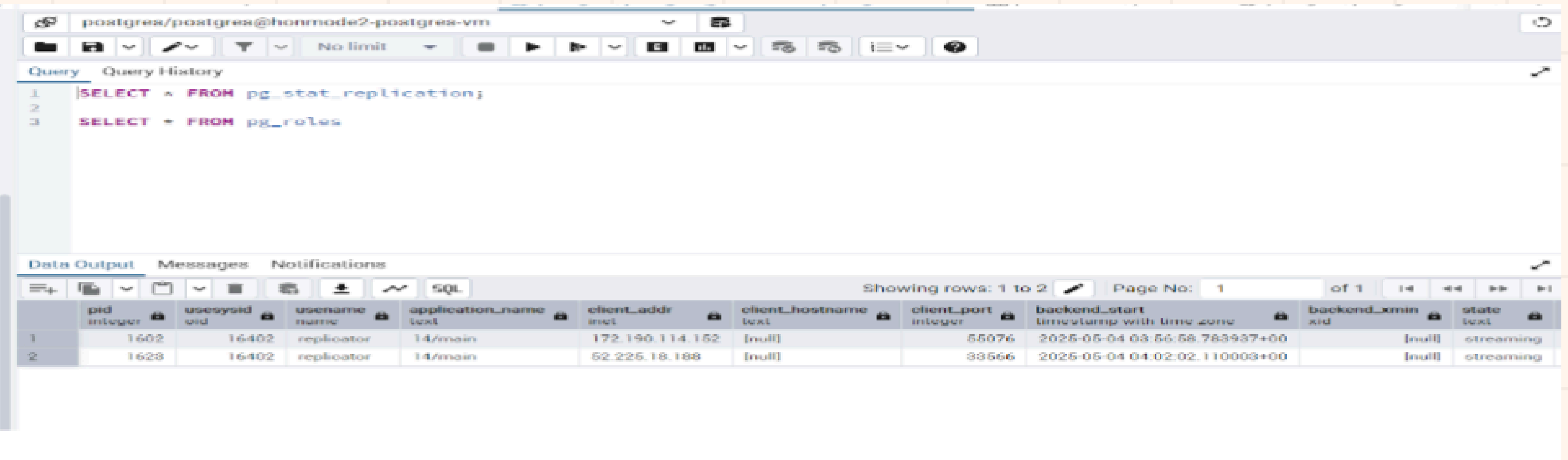
STREAMING REPLICATION SETUP

- node1 was treated as the primary.
- A replication user (replicator) was created with replication privileges.
- On node1:
    - Enabled wal_level = replica, max_wal_senders = 10, archive_mode = on.
- Used pg_basebackup to clone the data directory from node1 to both replicas.
- Placed a standby.signal file on secondary nodes and configured primary_conninfo in postgresql.conf.
- Ensured correct entries in pg_hba.conf to allow replication traffic.
- Verified real-time replication using pg_stat_replication on the primary.



```
azureuser@finalproject:~$ sudo -u postgres psql -c "SELECT pg_is_in_recovery();"
could not change directory to "/home/azureuser": Permission denied
 pg_is_in_recovery
-------------------
 t
(1 row)

azureuser@finalproject:~$
```



postgres/postgres@honnode2-postgres-vm

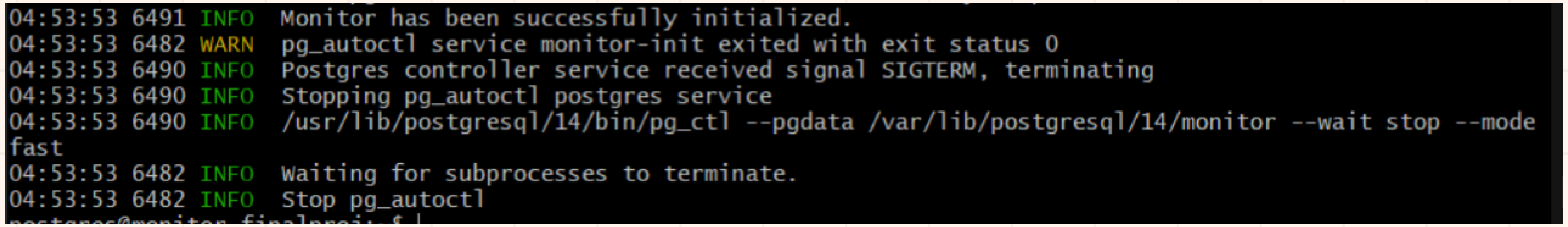
```
SELECT * FROM pg_stat_replication;

SELECT * FROM pg_roles
```

	pid integer	usesysid oid	usename name	application_name text	client_addr inet	client_hostname text	client_port integer	backend_start timestamp with time zone	backend_xmin xid	state text
1	1602	16402	replicator	14/main	172.190.114.152	[null]	55076	2025-05-04 03:56:58.783937+00	[null]	streaming
2	1623	16402	replicator	14/main	52.225.18.188	[null]	33566	2025-05-04 04:02:02.110003+00	[null]	streaming

FAILOVER WITH
PG_AUTO_FAILOVER

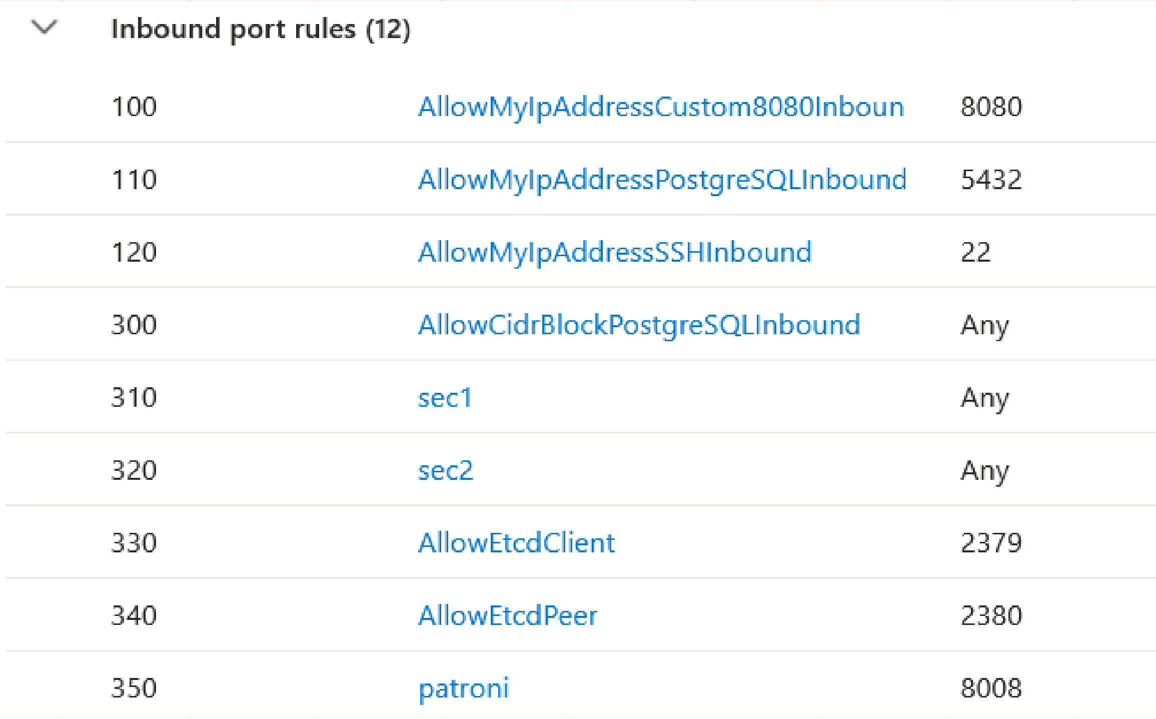
- Installed pg_autoctl on all three database VMs and a 4th monitor node.
- Registered each node with the monitor: node1 as primary, secondary-1 & secondary-2 as secondaries.
- However, we encountered repeated errors:
    - Connection refused to the monitor
    - SSL requirement & no pg_hba.conf entries
    - Retry deadline exceeded and stuck node states
- Despite multiple attempts (rebuilding nodes, re-registering, checking NSGs), we couldn't stabilize the setup.
- Switched to Patroni, which offered more transparent logs and control.



```
04:53:53 6491 INFO  Monitor has been successfully initialized.
04:53:53 6482 WARN  pg_autoctl service monitor-init exited with exit status 0
04:53:53 6490 INFO  Postgres controller service received signal SIGTERM, terminating
04:53:53 6490 INFO  Stopping pg_autoctl postgres service
04:53:53 6490 INFO  /usr/lib/postgresql/14/bin/pg_ctl --pgdata /var/lib/postgresql/14/monitor --wait stop --mode
fast
04:53:53 6482 INFO  Waiting for subprocesses to terminate.
04:53:53 6482 INFO  Stop pg_autoctl
postgres@monitor-finalproj:~$
```

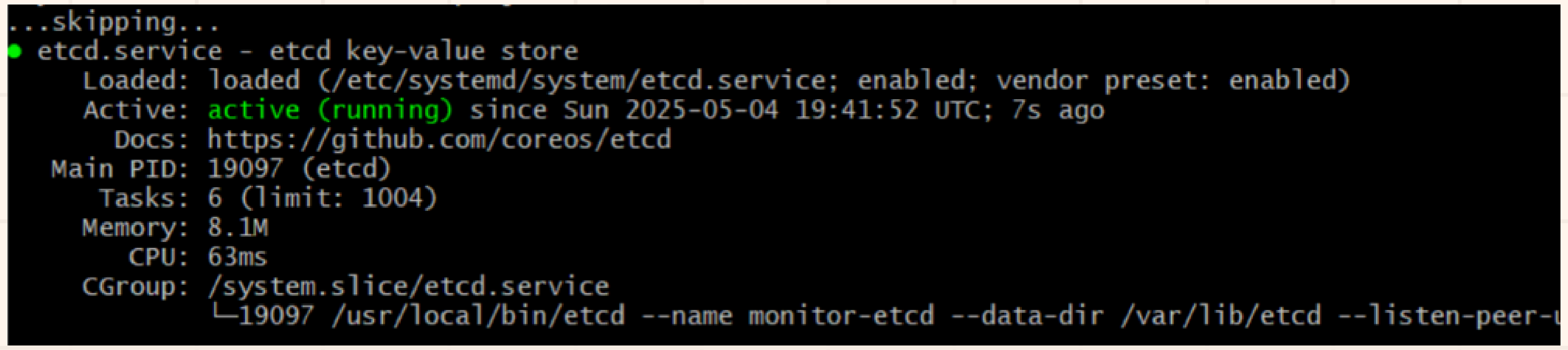
SWITCHING TO PATRONI

- Patroni was installed along with Python, psycopg2, and etcd.
- etcd was configured on a separate monitor VM for consensus and leader election.
- Each PostgreSQL VM had a dedicated patroni.yml config file with:
  - PostgreSQL data directory path
  - Node-specific restapi.listen (port 8008) and postgresql.listen (port 5432)
  - etcd URL for DCS coordination
  - Replication settings (user, password, slots)
- Network Security Groups (NSGs) were configured to allow intra-cluster traffic.
- Patroni was started via systemctl and verified using REST endpoints.



Inbound port rules (12)

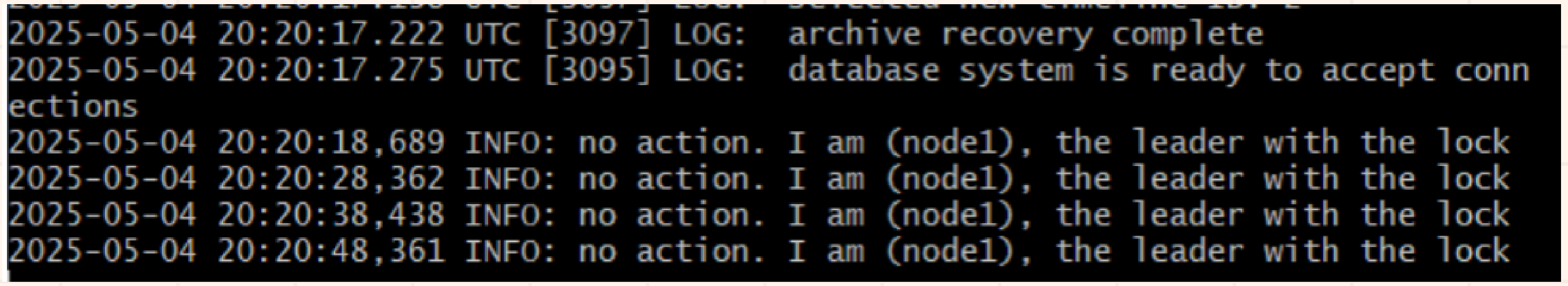
100	AllowMyIpAddressCustom8080Inboun	8080
110	AllowMyIpAddressPostgreSQLInbound	5432
120	AllowMyIpAddressSSHInbound	22
300	AllowCidrBlockPostgreSQLInbound	Any
310	sec1	Any
320	sec2	Any
330	AllowEtcdClient	2379
340	AllowEtcdPeer	2380
350	patroni	8008



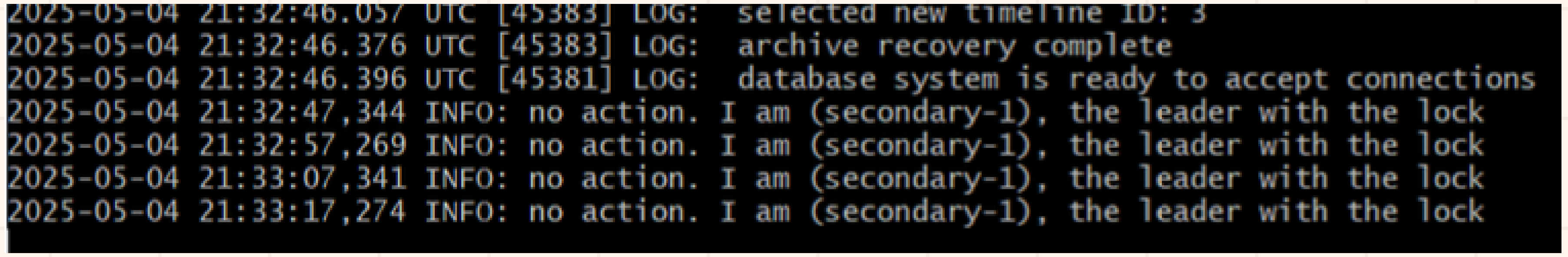
```
...skipping...
● etcd.service - etcd key-value store
     Loaded: loaded (/etc/systemd/system/etcd.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-05-04 19:41:52 UTC; 7s ago
       Docs: https://github.com/coreos/etcd
   Main PID: 19097 (etcd)
      Tasks: 6 (limit: 1004)
     Memory: 8.1M
        CPU: 63ms
     CGroup: /system.slice/etcd.service
             └─19097 /usr/local/bin/etcd --name monitor-etcd --data-dir /var/lib/etcd --listen-peer-
```

FAILOVER DEMONSTRATION

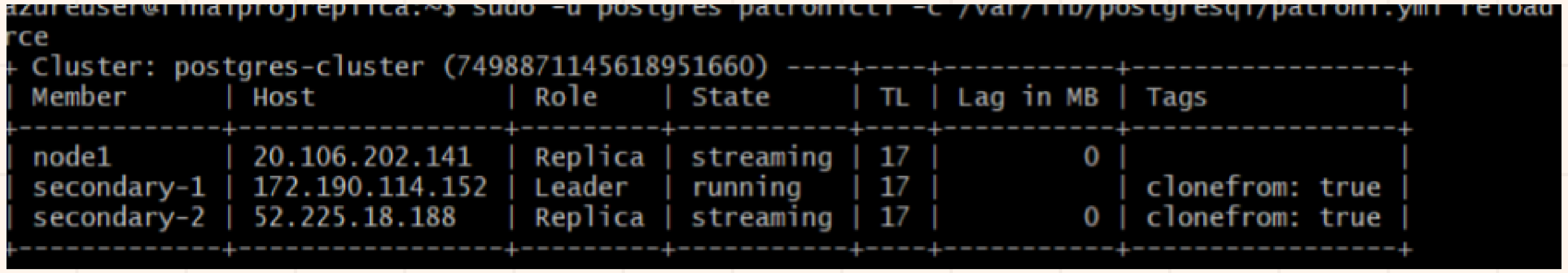
- Patroni automatically elected node1 as the initial leader.
- We killed PostgreSQL on node1 → secondary-1 took over as the new leader.
- Confirmed this via Patroni REST API and psql.
- When secondary-1 was killed, secondary-2 was automatically promoted.
- Once stopped node restarted, it instantly joined as follower.
- We validated **replication integrity**:
  - Inserted/deleted data on primary
  - Checked propagation to both replicas
  - **Tested Access:** Verified that writes on replicas were correctly rejected (read-only state)



```
2025-05-04 20:20:17.198 UTC [3097] LOG:  selected new timeline ID: 2
2025-05-04 20:20:17.222 UTC [3097] LOG:  archive recovery complete
2025-05-04 20:20:17.275 UTC [3095] LOG:  database system is ready to accept conn
ections
2025-05-04 20:20:18,689 INFO: no action. I am (node1), the leader with the lock
2025-05-04 20:20:28,362 INFO: no action. I am (node1), the leader with the lock
2025-05-04 20:20:38,438 INFO: no action. I am (node1), the leader with the lock
2025-05-04 20:20:48,361 INFO: no action. I am (node1), the leader with the lock
```



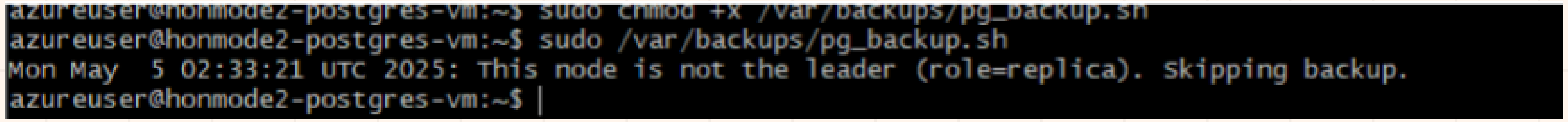
```
2025-05-04 21:32:46.057 UTC [45383] LOG:  selected new timeline ID: 3
2025-05-04 21:32:46.376 UTC [45383] LOG:  archive recovery complete
2025-05-04 21:32:46.396 UTC [45381] LOG:  database system is ready to accept connections
2025-05-04 21:32:47,344 INFO: no action. I am (secondary-1), the leader with the lock
2025-05-04 21:32:57,269 INFO: no action. I am (secondary-1), the leader with the lock
2025-05-04 21:33:07,341 INFO: no action. I am (secondary-1), the leader with the lock
2025-05-04 21:33:17,274 INFO: no action. I am (secondary-1), the leader with the lock
```



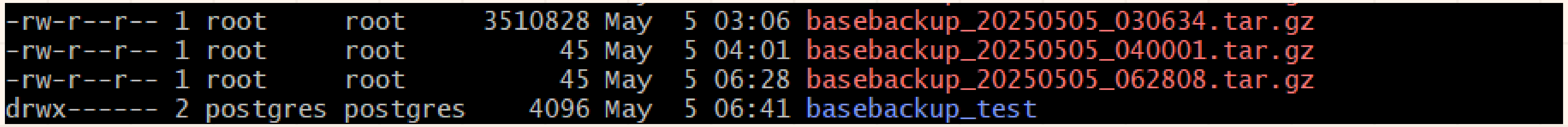
```
azureuser@finalprojreplica:~$ sudo -u postgres patronictl -c /var/lib/postgresql/patroni.yml reload
rce
+ Cluster: postgres-cluster (7498871145618951660) ----+----+-----------+-----------------+
| Member      | Host            | Role    | State     | TL | Lag in MB | Tags            |
+-------------+-----------------+---------+-----------+----+-----------+-----------------+
| node1       | 20.106.202.141  | Replica | streaming | 17 |         0 |                 |
| secondary-1 | 172.190.114.152 | Leader  | running   | 17 |           | clonefrom: true |
| secondary-2 | 52.225.18.188   | Replica | streaming | 17 |         0 | clonefrom: true |
+-------------+-----------------+---------+-----------+----+-----------+-----------------+
```

BACKUP

- A bash script (pg_backup.sh) was created to:
  - Check if node is primary (via Patroni API)
  - Run pg_basebackup and store the dump as .tar.gz in /var/backups/postgres/
  - Maintain logs & keeps only 3 latest files
- A cron job was configured to trigger this script every 30 minutes:
  - */30 * * * * /var/backups/pg_backup.sh
- Verified backups by listing and inspecting the tarballs.
- Ensured only the leader performs the backup to maintain consistency.



```
azureuser@honmode2-postgres-vm:~$ sudo chmod +x /var/backups/pg_backup.sh
azureuser@honmode2-postgres-vm:~$ sudo /var/backups/pg_backup.sh
Mon May  5 02:33:21 UTC 2025: This node is not the leader (role=replica). Skipping backup.
azureuser@honmode2-postgres-vm:~$ |
```

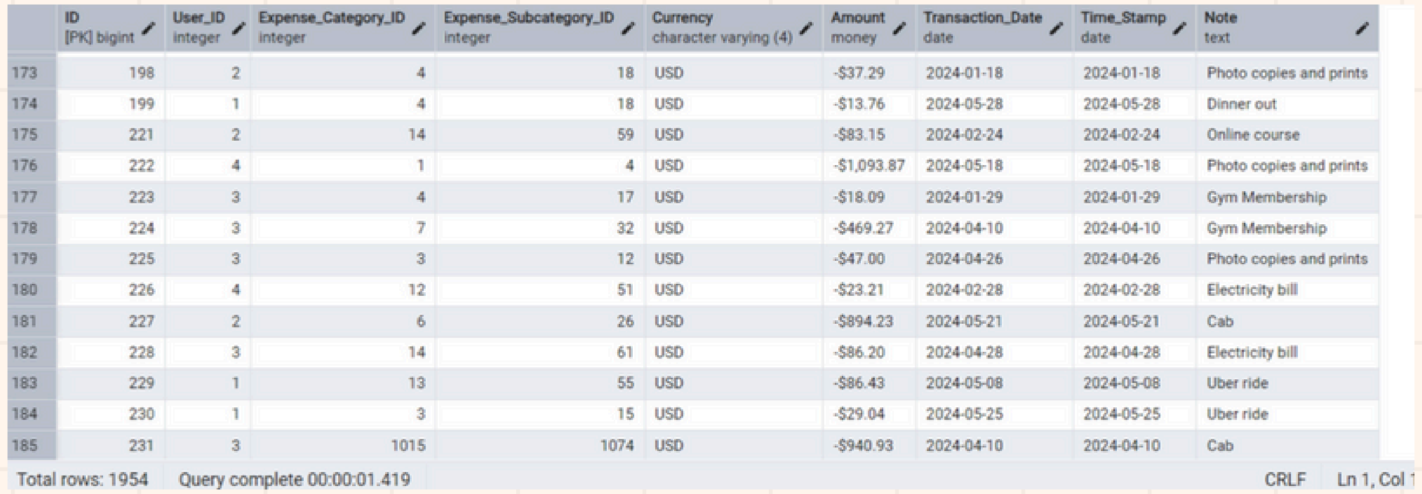


```
-rw-r--r-- 1 root     root     3510828 May  5 03:06 basebackup_20250505_030634.tar.gz
-rw-r--r-- 1 root     root          45 May  5 04:01 basebackup_20250505_040001.tar.gz
-rw-r--r-- 1 root     root          45 May  5 06:28 basebackup_20250505_062808.tar.gz
drwx------ 2 postgres postgres    4096 May  5 06:41 basebackup_test
```

RESTORE

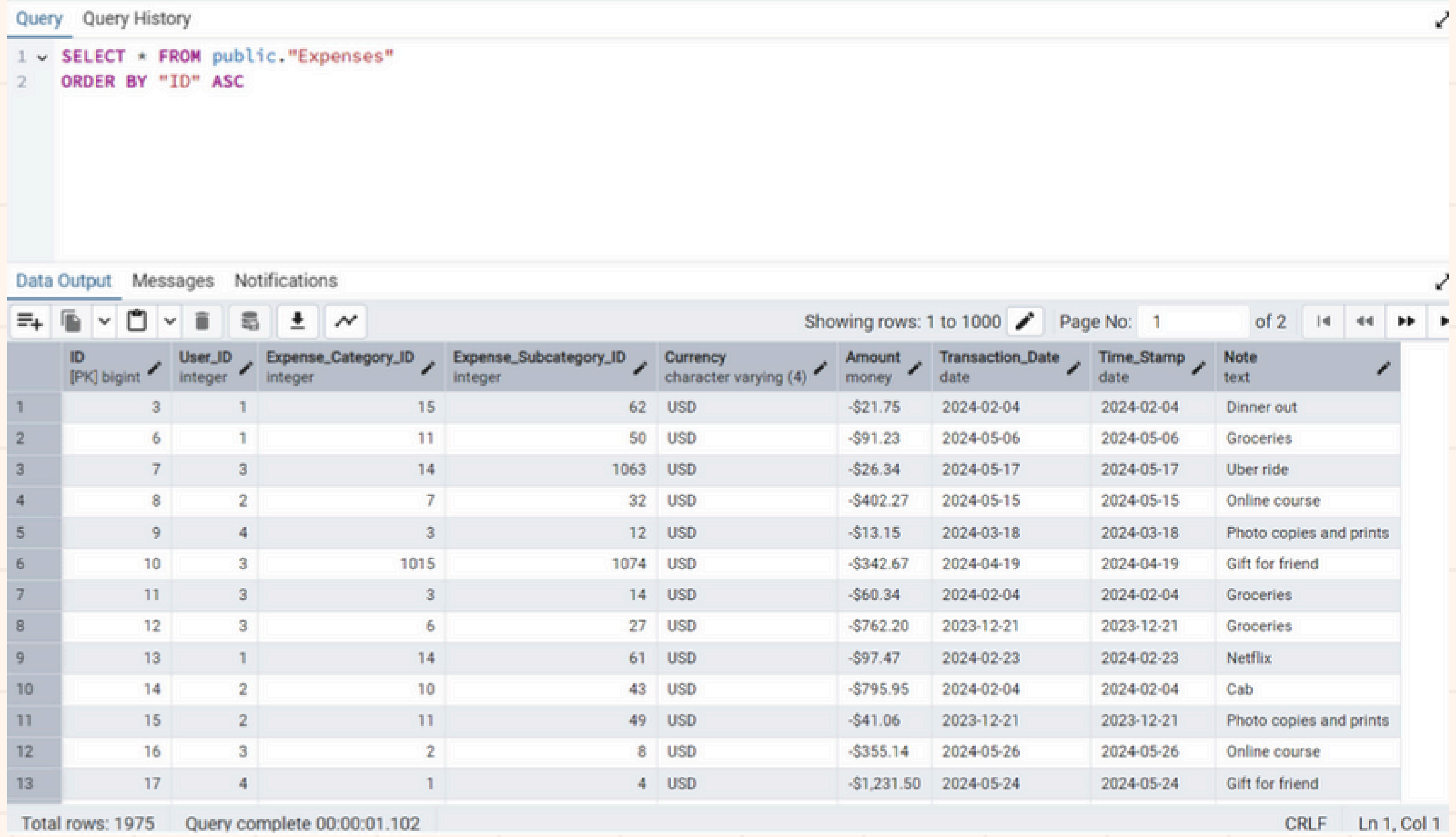
- To simulate recovery:
  - All nodes were stopped
  - On the node where the backup was taken, the data directory /var/lib/postgresql/14/main was cleared
  - A specific backup archive was extracted into this location
  - postgresql.conf and pg_hba.conf were manually restored
  - Permissions were reset (chown postgres:postgres, chmod 700)
  - Did not bootstrap as leader
- PostgreSQL was launched in standalone mode (not via Patroni):
  - sudo -u postgres /usr/lib/postgresql/14/bin/postgres -D /var/lib/postgresql/14/main
- pgAdmin confirmed that the restored data reflected the snapshot taken at backup time

Rows at time of backup: 1975, after backup – deleted 21 rows, after restore → back to 1975



	ID [PK] bigint	User_ID integer	Expense_Category_ID integer	Expense_Subcategory_ID integer	Currency character varying (4)	Amount money	Transaction_Date date	Time_Stamp date	Note text
173	198	2	4	18	USD	-$37.29	2024-01-18	2024-01-18	Photo copies and prints
174	199	1	4	18	USD	-$13.76	2024-05-28	2024-05-28	Dinner out
175	221	2	14	59	USD	-$83.15	2024-02-24	2024-02-24	Online course
176	222	4	1	4	USD	-$1,093.87	2024-05-18	2024-05-18	Photo copies and prints
177	223	3	4	17	USD	-$18.09	2024-01-29	2024-01-29	Gym Membership
178	224	3	7	32	USD	-$469.27	2024-04-10	2024-04-10	Gym Membership
179	225	3	3	12	USD	-$47.00	2024-04-26	2024-04-26	Photo copies and prints
180	226	4	12	51	USD	-$23.21	2024-02-28	2024-02-28	Electricity bill
181	227	2	6	26	USD	-$894.23	2024-05-21	2024-05-21	Cab
182	228	3	14	61	USD	-$86.20	2024-04-28	2024-04-28	Electricity bill
183	229	1	13	55	USD	-$86.43	2024-05-08	2024-05-08	Uber ride
184	230	1	3	15	USD	-$29.04	2024-05-25	2024-05-25	Uber ride
185	231	3	1015	1074	USD	-$940.93	2024-04-10	2024-04-10	Cab

Total rows: 1954 Query complete 00:00:01.419 CRLF Ln 1, Col



```
SELECT * FROM public."Expenses"
ORDER BY "ID" ASC
```

Showing rows: 1 to 1000 Page No: 1 of 2

	ID [PK] bigint	User_ID integer	Expense_Category_ID integer	Expense_Subcategory_ID integer	Currency character varying (4)	Amount money	Transaction_Date date	Time_Stamp date	Note text
1	3	1	15	62	USD	-$21.75	2024-02-04	2024-02-04	Dinner out
2	6	1	11	50	USD	-$91.23	2024-05-06	2024-05-06	Groceries
3	7	3	14	1063	USD	-$26.34	2024-05-17	2024-05-17	Uber ride
4	8	2	7	32	USD	-$402.27	2024-05-15	2024-05-15	Online course
5	9	4	3	12	USD	-$13.15	2024-03-18	2024-03-18	Photo copies and prints
6	10	3	1015	1074	USD	-$342.67	2024-04-19	2024-04-19	Gift for friend
7	11	3	3	14	USD	-$60.34	2024-02-04	2024-02-04	Groceries
8	12	3	6	27	USD	-$762.20	2023-12-21	2023-12-21	Groceries
9	13	1	14	61	USD	-$97.47	2024-02-23	2024-02-23	Netflix
10	14	2	10	43	USD	-$795.95	2024-02-04	2024-02-04	Cab
11	15	2	11	49	USD	-$41.06	2023-12-21	2023-12-21	Photo copies and prints
12	16	3	2	8	USD	-$355.14	2024-05-26	2024-05-26	Online course
13	17	4	1	4	USD	-$1,231.50	2024-05-24	2024-05-24	Gift for friend

Total rows: 1975 Query complete 00:00:01.102 CRLF Ln 1, Col 1

LEARNINGS

INFRASTRUCTURE MANAGEMENT

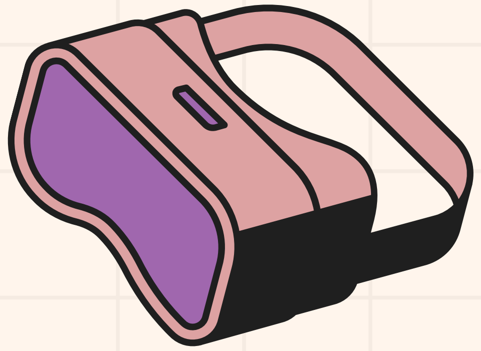
- Provisioned and networked 4 Ubuntu VMs on Azure
- Configured internal IPs, NSG rules, and SSH access
- Enabled secure communication across cluster nodes

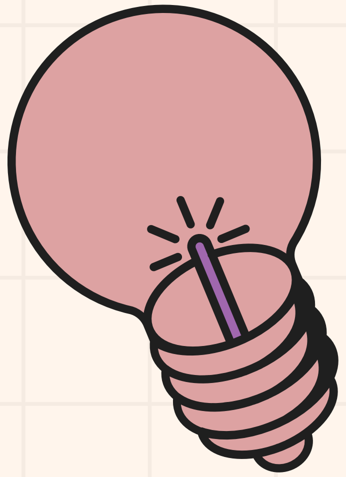
HIGH AVAILABILITY & FAILOVER

- Set up streaming replication between primary and replicas
- Tested pg_auto_failover and resolved registration issues
- Successfully implemented automatic failover using Patroni

BACKUP & RESTORE STRATEGIES

- Automated compressed base backups using cron and pg_basebackup
- Simulated full recovery from tar.gz backup files
- Handled missing config files and recovered database manually





THANK YOU

QUESTIONS?