HOW TO INSTALL MARIADB

1. Update packages: sudo apt-get update

2. Install it from the Ubuntu repository:
   sudo apt-get install mariadb-server

3. Once installed, enable the service to automatically run it at the boot:
   a. *sudo systemctl start mariadb*
   b. *sudo systemctl enable mariadb*

4. Run the mysql_secure_installation to set a password for ‘root’ (suggested: ‘Robotics.123456’)
   a. *sudo mysql_secure_installation*
   b. *unix_socket authentication: ‘n’.*
   c. *Change root password: ‘n’.*
   d. *Remove anonymous user: ‘y’.*
   e. *Disallow root login remotely: ‘y’.*
   f. *Remove test database: ‘y’.*
   g. *Reload privilage tables: ‘y’.*

5. Verify MariaDB is installed and configure an user:
   a. *sudo mariadb*
   b. (you should see: ‘Server version: 10.6.18-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04’)
   c. MariaDB [(none)]> *CREATE USER 'user'@'localhost' IDENTIFIED BY 'Robotics.123456';*
   d. MariaDB [(none)]> *GRANT ALL PRIVILEGES ON *.* TO 'user'@'localhost';*
   e. MariaDB [(none)]> *FLUSH PRIVILEGES;*
   f. MariaDB [(none)]> *exit;*

6. Install DBeaver for the User Interface, installing from:

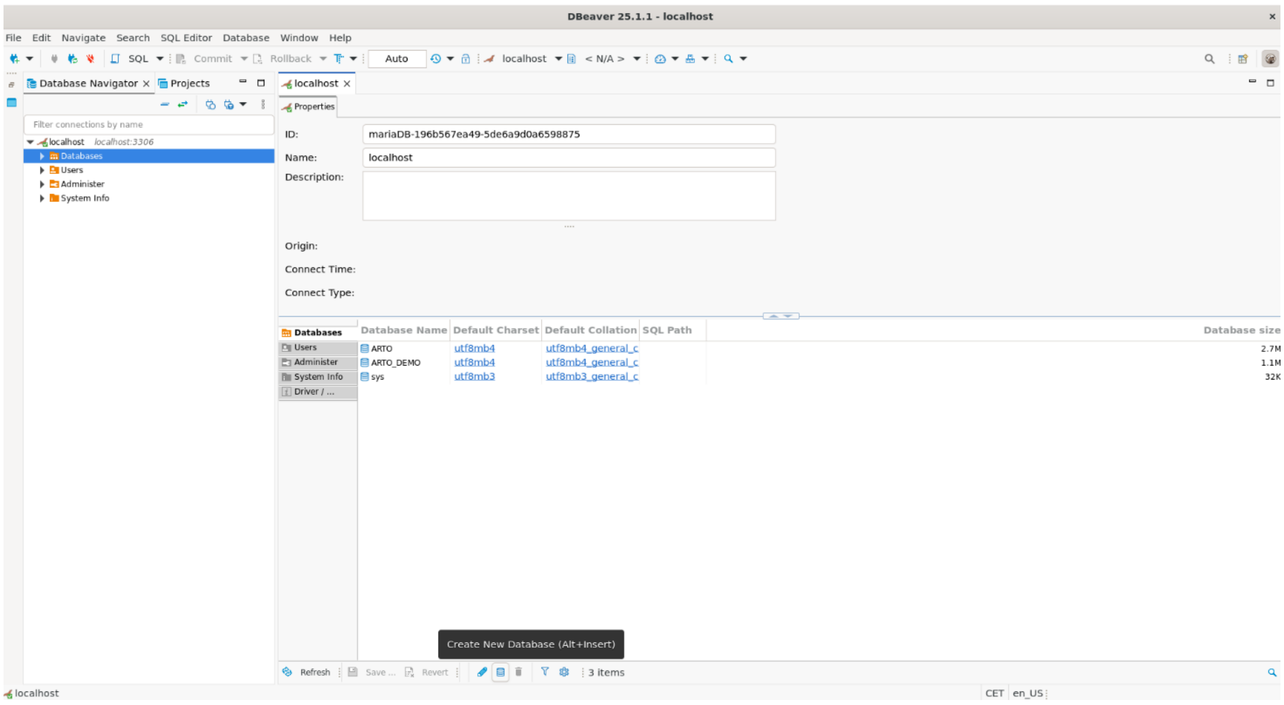
   How To Install DBeaver on Ubuntu 24.04|22.04|20.04 | ComputingForGeeks

   SKIP THE POINT 3 !!!

   At point 4 of the guide, use the credentials created above.

HOW TO IMPORT A DATABASE IN DBeaver

1. Open DBeaver, open the drop-down menu of ‘localhost’ server. Make a double click on Databases.
2. At the bottom of this window, select the databases icon that says 'Create new Database'



ID: mariaDB-196b567ea49-5de6a9d0a6598875

Name: localhost

Description:

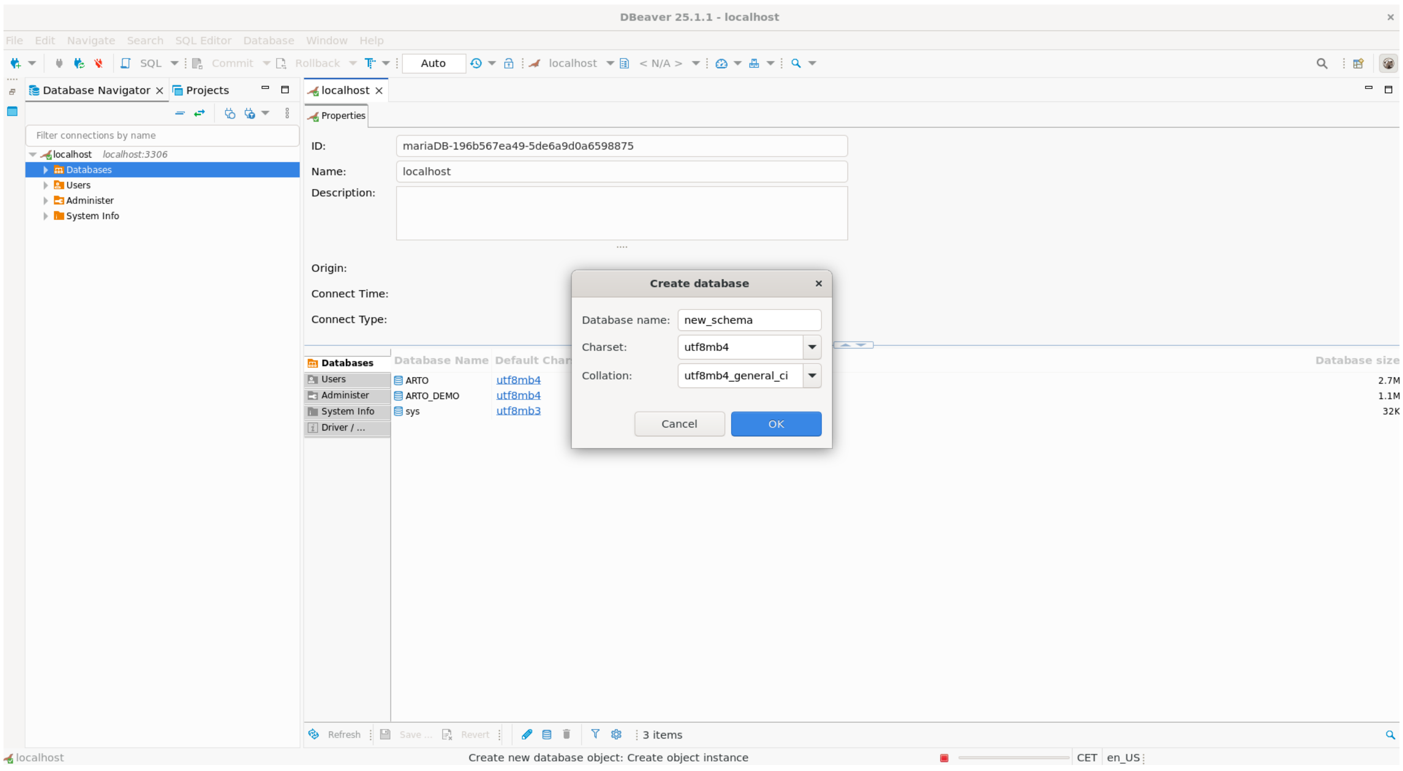
Origin:

Connect Time:

Connect Type:

Database Name	Default Charset	Default Collation	SQL Path	Database size
ARTO	utf8mb4	utf8mb4_general_c		2.7M
ARTO_DEMO	utf8mb4	utf8mb4_general_c		1.1M
sys	utf8mb3	utf8mb3_general_c		32K

3. Choose a name to the new database and select OK



ID: mariaDB-196b567ea49-5de6a9d0a6598875

Name: localhost

Description:

Origin:

Connect Time:

Connect Type:

Database name: new_schema

Charset: utf8mb4

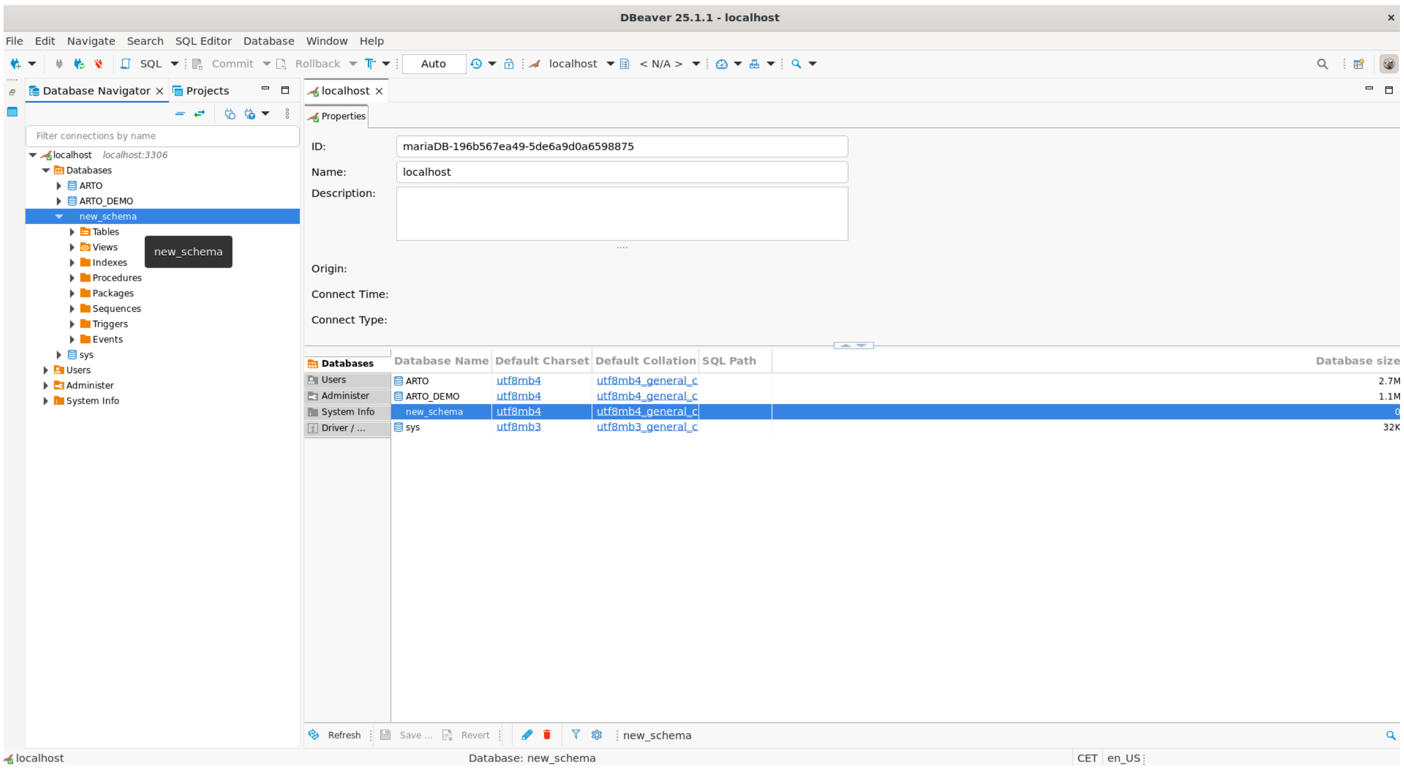
Collation: utf8mb4_general_ci

Database Name	Default Char	Database size
ARTO	utf8mb4	2.7M
ARTO_DEMO	utf8mb4	1.1M
sys	utf8mb3	32K

Create new database object: Create object instance

4. Open the drop-down menu Databases and check if the new database schema has appeared.

5. Right click on the new schema, then click ‘Tools’ and finally 'Restore Database’.



ID: mariaDB-196b567ea49-5de6a9d0a6598875

Name: localhost

Description:

Origin:

Connect Time:

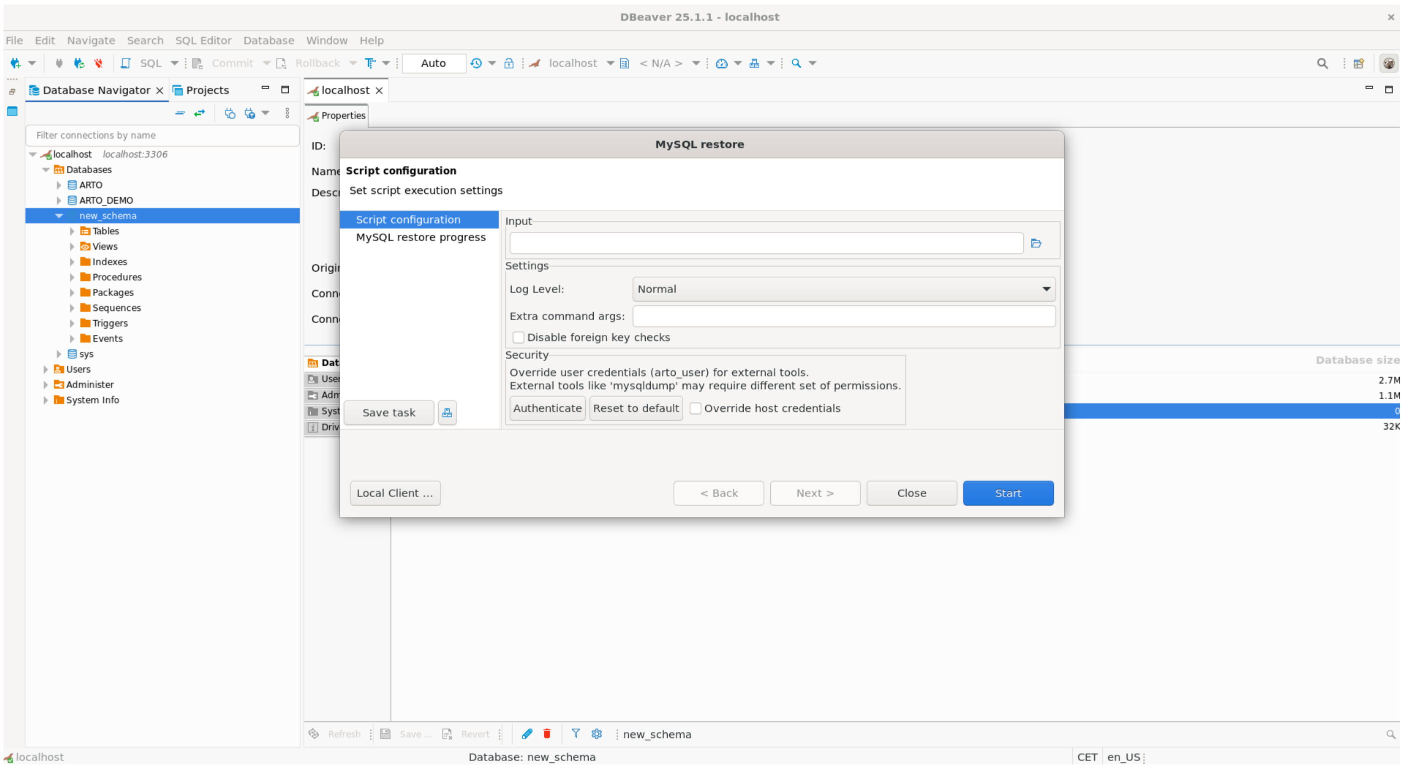
Connect Type:

Databases	Database Name	Default Charset	Default Collation	SQL Path	Database size
Users	ARTO	utf8mb4	utf8mb4_general_c		2.7M
Administer	ARTO_DEMO	utf8mb4	utf8mb4_general_c		1.1M
System Info	new_schema	utf8mb4	utf8mb4_general_c		0
Driver / ...	sys	utf8mb3	utf8mb3_general_c		32K

new_schema

Database: new_schema

6. In the input section select the .sql file of your database schema backup and then Start.



MySQL restore

Script configuration

Set script execution settings

Script configuration
MySQL restore progress

Input

Settings

Log Level: Normal

Extra command args:

☐ Disable foreign key checks

Security

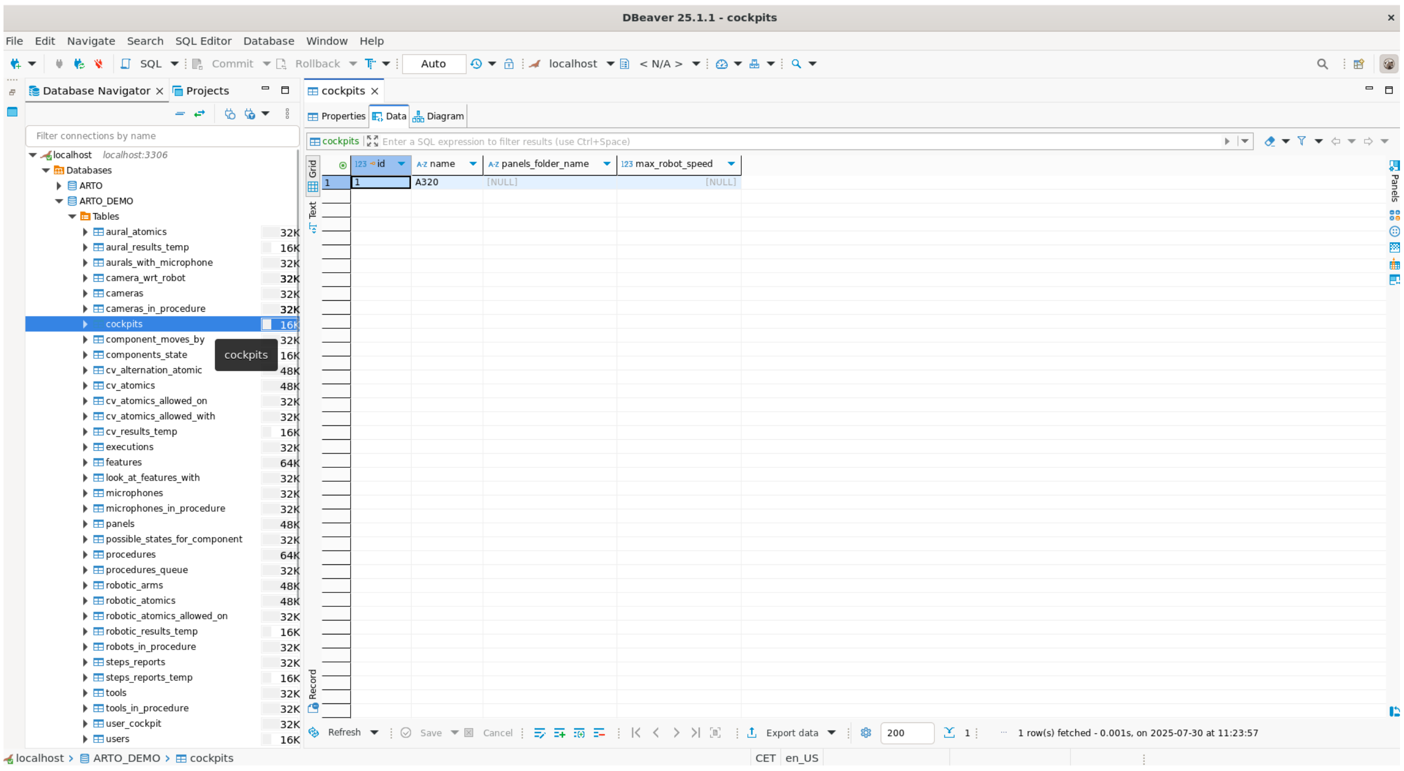
Override user credentials (arto_user) for external tools.
External tools like 'mysqldump' may require different set of permissions.

☐ Override host credentials

new_schema

Database: new_schema

7. Finally, the database will be searchable under its drop-down menu and under the Tables drop-down menu by double-clicking on the table of interest.


DBeaver 25.1.1 - cockpits

localhost localhost:3306
- Databases
  - ARTO
  - ARTO_DEMO
    - Tables

aural_atomics	32K
aural_results_temp	16K
aurals_with_microphone	32K
camera_wrt_robot	32K
cameras	32K
cameras_in_procedure	32K
cockpits	16K
component_moves_by	32K
components_state	16K
cv_alternation_atomic	48K
cv_atomics	48K
cv_atomics_allowed_on	32K
cv_atomics_allowed_with	32K
cv_results_temp	16K
executions	32K
features	64K
look_at_features_with	32K
microphones	32K
microphones_in_procedure	32K
panels	48K
possible_states_for_component	32K
procedures	64K
procedures_queue	32K
robotic_arms	48K
robotic_atomics	48K
robotic_atomics_allowed_on	32K
robotic_results_temp	16K
robots_in_procedure	32K
steps_reports	32K
steps_reports_temp	16K
tools	32K
tools_in_procedure	32K
user_cockpit	32K
users	16K

	id	name	panels_folder_name	max_robot_speed
1	1	A320	[NULL]	[NULL]

1 row(s) fetched - 0.001s, on 2025-07-30 at 11:23:57

localhost > ARTO_DEMO > cockpits

USE MARIADB WITH PYTHON

To use mariadb in python, execute this command:

pip install mariadb
sudo apt install libmysqlclient-dev default-libmysqlclient-dev

Here an example of code .ipynb

```
import mariadb

# === CONFIG ===
DB_CONFIG = {
    'host': '127.0.0.1',
    'port': 3306,
    'user': 'user',
    'password': 'Robotics.123456',
    'database': 'ARTO'
}

# === Funzione per connettersi al DB e leggere il valore offset_pose e task_frame ===
def get_atomic_by_name(atomic_name):

    # Connessione database
    conn = mariadb.connect(**DB_CONFIG)
    cursor = conn.cursor()

    # Esecuzione query
    query = """ SELECT offset_pose, task_frame FROM robotic_atomics WHERE name = %s """
    cursor.execute(query, (atomic_name,))
    result = cursor.fetchone()

    # Chiusura connessione database
    cursor.close()
    conn.close()
    # Analisi risultati query
    if not result:
        raise ValueError(f"No RoboticAtomic found with name: {atomic_name}")

    offset_pose = result[0]

    # Se la stringa è un dizionario o lista rappresentato come stringa, la correggiamo
    try:
        # Rimuovere eventuali parentesi graffe o caratteri indesiderati
        offset_pose = offset_pose.strip('{}')
        # Dividere la stringa per ottenere i singoli valori
        offset_values = [float(x) for x in offset_pose.split(',')]

        if len(offset_values) != 7:
            raise ValueError(f"Invalid offset_pose size ({len(offset_values)}) for RoboticAtomic: {atomic_name}")

    except ValueError as e:
        raise ValueError(f"Error parsing offset_pose: {e}")

    task_frame_id = result[1]

    return offset_values, task_frame_id

# === ESECUZIONE ===
atomic_name_to_load = "MRfcuappr"  # Cambia con il nome corretto nel DB
try:
    offset_values, task_frame_id = get_atomic_by_name(atomic_name_to_load)
    print(f"Offset values for {atomic_name_to_load}: {offset_values}\nTask frame id: {task_frame_id}")
except Exception as e:
    print(f"Error: {e}")
```