Ubuntu安装Mysql

ubuntu安装Mysql

下面的步骤将帮助你在 Ubuntu 上安装并初始配置 MySQL 数据库。演示环境以 Ubuntu 20.04/22.04 为例，其他版本步骤也相似。

一、更新软件包索引

首先更新本地软件包索引，保证安装的是最新的软件：

```
bash

  sudo apt update
```

二、安装 MySQL

Ubuntu 官方仓库一般包含了 MySQL 社区版，可以直接使用 `apt` 命令安装：

```
bash

  sudo apt install mysql-server
```

安装完成后，MySQL 服务会自动启动并在系统引导时自动启动。你可以通过以下命令来检查 MySQL 服务状态：

```
bash

  sudo systemctl status mysql
```

如果你看到状态为 `active (running)`，表示 MySQL 已经正常运行。

三、执行安全加固配置

MySQL 安装完成后，建议执行一个安全脚本来提升数据库的安全性，该脚本会引导你设置 MySQL 的 root 密码并执行其他安全项设置：

```bash
sudo mysql_secure_installation
```

该脚本将依次引导你进行以下操作：

1. **设置或修改 root 用户密码**
2. **是否删除匿名用户**
3. **是否禁止 root 用户远程登录**
4. **是否删除测试数据库**
5. **是否重新加载权限表**

根据实际需求进行选择，一般推荐选择“是（Y）”来提高安全性。

四、登录并验证安装

执行完安全配置后，就可以以 root 身份登录 MySQL：

```bash
sudo mysql -u root -p
```

输入你在上述步骤中设置的 root 密码后，即可进入 MySQL 命令行：

```sql
mysql> SELECT VERSION();
+-----------+
| VERSION() |
+-----------+
| 8.0.xx    |
+-----------+
1 row in set (0.00 sec)
```

出现版本号表示 MySQL 已成功安装并运行。

五、（可选）创建普通用户并授予权限

建议不要直接使用 root 账户操作数据库，可以根据实际需求创建一个新的用户并指定权限。例如创建一个名为 `myuser` 的用户并授予其对某个数据库或所有数据库的访问权限：

1. **创建用户**

```sql
CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';
```

2. **创建数据库（如有需要）**

```sql
CREATE DATABASE mydb;
```

3. **授予权限**

```sql
GRANT ALL PRIVILEGES ON mydb.* TO 'myuser'@'localhost';
```

或者授予所有数据库权限：

```sql
GRANT ALL PRIVILEGES ON *.* TO 'myuser'@'localhost' WITH GRANT OPTION;
```

4. **刷新权限**

```sql
FLUSH PRIVILEGES;
```

六、常用管理命令

1. **启动 MySQL 服务**

```bash
sudo systemctl start mysql
```

2. **停止 MySQL 服务**

```bash
sudo systemctl stop mysql
```

3. **重启 MySQL 服务**

```bash
sudo systemctl restart mysql
```

4. **查看 MySQL 状态**

```bash
sudo systemctl status mysql
```

5. **开机自启动/禁用自启动**

```bash
# 启用开机启动
sudo systemctl enable mysql
# 禁用开机启动
sudo systemctl disable mysql
```

完成以上步骤后，你就成功在 Ubuntu 上安装并配置了一个可用的 MySQL 数据库服务。根据自己的项目需求，你还可以进一步优化 MySQL 配置文件（/etc/mysql/mysql.conf.d/mysqld.cnf），以及调整数据存储路径、最大连接数等高级参数。

祝你使用顺利！

root@ubuntu2404:~# sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No:

`**VALIDATE PASSWORD**` 组件用于在你设置密码时对密码强度进行检查，并只允许符合安全要求的密码（如长度、字符组合等）。启用后，每当你创建或修改用户密码时，MySQL 都会进行严格的密码复杂度验证，从而提升数据库安全性。但也会带来一些不便，比如在测试环境中，可能需要频繁设置较弱的临时密码。

当你在执行 `**mysql_secure_installation**` 时，遇到下述提示时：

```vbnet

  VALIDATE PASSWORD COMPONENT can be used to test passwords
  and improve security. It checks the strength of password
  and allows the users to set only those passwords which are
  secure enough. Would you like to setup VALIDATE PASSWORD component?

  Press y|Y for Yes, any other key for No:
```

你可以根据实际需求来选择：

1. **按 `y` 或 `Y`（推荐在生产环境中使用）**
   - 接下来会提示你选择验证等级（LOW/MEDIUM/STRONG），数字 0/1/2 对应三个等级。
   - 当验证等级越高，对密码长度、字符种类、复杂度的要求就越严格。
   - 如果你是在正式生产环境中，为了尽可能保证安全，推荐至少选择 `MEDIUM` 或者更高等级。
   - 需要注意的是，较高的密码验证等级可能要求非常复杂的密码，不满足条件会导致修改密码失败。
2. **按其它任意键（不启用）**
   - `VALIDATE PASSWORD` 组件将不会启用，你可以使用任意强度的密码。
   - 在测试环境或对密码安全性要求不高的场景下，可能比较方便，但安全性相对较弱。

选择示例

- 如果你按 `y` 选择启用验证功能，MySQL 会提示如下信息（示例）：

  ```sql

    There are three levels of password validation policy:

    LOW    Length >= 8
    MEDIUM Length >= 8, numeric, mixed case, and special characters
    STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

    Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG:
  ```

  根据需要输入对应数字即可。随后会提示你设置 root 密码，并会对你输入的密码进行强度校验。

- 如果你按其它任意键跳过，则直接进入下一步让你设置或修改 MySQL root 用户的密码（不会校验强度）。

后续操作

启用或跳过 `VALIDATE PASSWORD` 后，`mysql_secure_installation` 还会提示你进行以下操作：

- 是否删除匿名用户（Yes/No）
- 是否禁止 root 用户远程登录（Yes/No）
- 是否删除 test 测试数据库（Yes/No）
- 是否重新加载权限表（Yes/No）

一般来说，处于安全考虑都建议选择 “Yes”，但具体可根据你的实际场景决定。

总结：
如果你处于生产环境并且对数据库安全性要求较高，推荐启用 `VALIDATE PASSWORD` 并选择合适的验证等级；若只是测试或个人开发环境，不想被复杂密码策略限制，也可以选择跳过。

我这是测试环境

如果仅是测试环境，不想被复杂的密码策略所限制，可以在出现下述提示时 **选择跳过**（即按任意其他键而不是 `y` 或 `Y`）：

```
vbnet


  VALIDATE PASSWORD COMPONENT can be used to test passwords
  and improve security. It checks the strength of password
  and allows the users to set only those passwords which are
  secure enough. Would you like to setup VALIDATE PASSWORD component?

  Press y|Y for Yes, any other key for No:
```

1. **直接按回车或者按 N**，这样就不会启用 `VALIDATE PASSWORD` 验证功能。
2. 随后，你依然需要设置或修改 MySQL `root` 用户的密码，但这时不会对密码强度进行强制校验。
3. 接着的步骤（删除匿名用户、禁止 root 远程登录、删除测试数据库、刷新权限表等）可以根据实际需要选择，通常测试环境也建议进行简单的安全配置，比如删除匿名用户、删除测试数据库，免得将来忘记清理。

这样，你在测试环境中就不会受到严格密码策略的干扰，可以更自由地设置较简单的密码，方便开发与调试。

Skipping password set for root as authentication with auth_socket is used by default.
If you would like to use password authentication instead, this can be done with the "ALTER_USER" command.
See https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-password-management for more information.

By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) :

在 MySQL 8.0+ 的 Ubuntu/Debian 系统环境中，MySQL 默认使用 **auth_socket** 插件来认证 `root` 用户，所以 `mysql_secure_installation` 脚本会提示 **Skipping password set for root**，即跳过设置 `root` 密码。这代表你可以直接使用如下方式登录 MySQL 而无需输入密码：

```bash
sudo mysql
```

如果你以后希望使用传统的 **密码认证** 来登录 `root`，需要执行以下命令进行修改：

```sql
-- 以 root 身份登录（无需密码）
sudo mysql

-- 将 auth_socket 改为 mysql_native_password，并设置密码
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'NewPassword';

-- 刷新权限
FLUSH PRIVILEGES;
```

这样就会变为常见的密码登录方式（`mysql -u root -p`），并使用你设置的密码进行认证。

Remove anonymous users? (Press y|Y for Yes, any other key for No)

接下来脚本提示你是否 **移除匿名用户**。MySQL 在安装时默认创建了匿名用户，以便于一些简单测试和快速上手。如果只是临时的测试环境，是否删除匿名用户可以自行决定：

- **如果是测试环境，且想要快速操作、不在意潜在的安全风险**，你可以选择 **No**（即按任意其他键）。这样匿名用户依然存在，可以在测试中随时登录、无需密码。
- **如果是面向生产环境，或者尽管在测试环境也想保持清洁和安全**，推荐输入 **y/Y** 删除匿名用户。这样可以减少被未授权访问的潜在风险。

  建议：即使是测试环境，通常也建议养成良好的安全习惯，**删除匿名用户** 以避免日后环境混乱或安全问题。如果真的需要无密码登录做测试，也可以自行创建一个专门的测试账号，赋予指定权限并在用完后再删除。

小结

1. **auth_socket 插件**
   - Ubuntu 默认使用 `auth_socket` 插件来管理 `root` 登录，不需要输入密码。
   - 如需改为传统密码登录，使用 `ALTER USER … IDENTIFIED WITH mysql_native_password …`。
2. **匿名用户去留**
   - **y/Y**：推荐删除，尤其在生产环境中更安全。
   - **其他键**：保留匿名用户，适合非常轻量、临时的测试场景，但不安全。

根据你的实际需求做选择即可。如果你确认只是在私有环境中做一些简单测试，保留匿名用户也没大碍；不过从长远角度和良好习惯来说，还是建议删除匿名用户，免得以后忘记导致潜在安全风险。

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) :

`Disallow root login remotely? (Press y|Y for Yes, any other key for No) :`

这一步是让你选择**是否禁止（Disallow）`root` 用户的远程登录**。从安全角度来说，**不允许 `root` 用户远程登录**是更好的做法，这能有效减少暴力破解和其他网络攻击的风险。下面是常见场景下的选择建议：

1. **正式/生产环境**
   - **推荐选择 `y`/`Y` 来禁止 `root` 用户远程登录。**
   - 可以在需要远程管理时，用其他具有管理员权限的普通用户登录，再执行管理操作。
2. **测试环境**
   - 如果你只在本地使用 MySQL 或者没有强安全要求，选择 **禁止** 远程登录也没有坏处。
   - 如果你确实需要用 `root` 账号在远程管理测试服务器上的 MySQL，可以选择 **No**，即按其他任意键跳过禁止操作—but 需要确保你的服务器处于安全的测试环境中，且有其他安全策略（如防火墙、VPN）进行保护。

最佳实践：

- 尽量**不要**在生产环境中用 `root` 用户进行远程登录管理，安全风险高。
- 可以创建一个权限受控的普通管理员用户来执行远程管理，例如：

```sql
CREATE USER 'adminuser'@'%' IDENTIFIED BY 'StrongPassword';
GRANT ALL PRIVILEGES ON *.* TO 'adminuser'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

这样，你既能远程管理数据库，也不会暴露 `root` 账号给外网。

... skipping.
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) :

这一步是让你选择是否**删除默认的测试数据库 `test` 及其访问权限**。该数据库通常被用于一些简单的测试或学习场景。如果你不再需要它，或者你的环境准备上线到生产环境，删除它能够减少数据库中的不必要部分，并降低潜在的安全风险。

选择建议

1. **如果是生产或长期使用环境**
   - 推荐输入 `y` 或 `Y` 选择 **Yes**，来删除 `test` 数据库以及对它的所有访问权限。
   - 这样做能让数据库环境更简洁、安全。
2. **如果是简单的测试或学习环境**
   - 如果你仍会用到 `test` 数据库进行测试，选择 **No**（按下除 `y` 以外的任意键）也无妨。
   - 需要注意，`test` 数据库默认所有用户都能访问，这存在安全隐患。若测试结束后不再需要，最好还是删除。

小结：

- 在大多数情况下，尤其是准备投入生产或长期维护的环境，**删除 `test` 数据库**是更安全和干净的选择。
- 如果你是纯粹用来练手或一次性测试，不会对安全有严格要求，就可以保留。

... skipping.
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) :

在 MySQL 中，用户账户及其权限是存储在系统的“权限表”（privilege tables）中，当你对用户或权限进行修改后，这些更改通常需要通过 `FLUSH PRIVILEGES`（或者在该脚本中称作 **“Reload privilege tables”**）操作才能立即生效。

选择建议

- **按 `y` / `Y`：**
  - 会刷新权限表，让之前对用户、密码、远程登录、数据库等方面的修改马上生效。
  - 一般来说，**强烈推荐**在这里选择 **Yes**，以确保你刚才的所有更改立刻在当前会话中生效。
- **按其他任意键（No）：**
  - 不会刷新权限表，权限更改将不会立刻在当前会话中生效，直到下次 MySQL 重启或你手动执行 `FLUSH PRIVILEGES;` 命令。

- 如果只是测试环境，短期内不需要马上应用更改，也可以暂时不刷新，但大多数情况下还是会选 Yes 以避免混淆。

总结：
选择 **Yes** 可以让所有的安全配置和权限修改立即生效，这也是最常见、最安全的做法。

如何设置mysql的root用户密码为root，让远程的任意服务器通过账号和密码都能连接mysql的root账户

以下步骤可让 **root** 用户使用密码 **root** 在 **任意远程主机** 连接到 MySQL。但是需要 **强烈提醒**：此操作极不安全，**仅用于测试或特殊需求场景**。在生产环境切勿使用如此弱的密码和放开的权限！

---

1. 修改 MySQL 配置，允许监听所有 IP

1. 打开 MySQL 的主配置文件，一般位于

```
bash

  /etc/mysql/mysql.conf.d/mysqld.cnf
```

或

```
bash

  /etc/mysql/my.cnf
```

2. 找到（或搜索）以下配置项（可能在 `[mysqld]` 段）：

```
css

  bind-address            = 127.0.0.1
```

将其改为：

```
css

  bind-address            = 0.0.0.0
```

这样 MySQL 将监听所有网卡地址，而不只监听 `127.0.0.1`（本地环回地址）。

3. 保存文件后，重启 MySQL：

```bash
sudo systemctl restart mysql
```

或者：

```bash
sudo service mysql restart
```

---

2. 将 root 用户从 auth_socket 插件改为密码认证

在大多数 Ubuntu / Debian 系统中，MySQL 默认使用 `auth_socket` 插件来让 `root` 通过系统用户直接登录（无需密码）。要想让 `root` 使用密码并允许远程访问，需执行以下操作：

1. 以 **系统 root** 或使用 `sudo` 登录 MySQL（不需密码）：

   ```bash
   sudo mysql
   ```

   进入 MySQL 交互命令行后，会显示类似：

   ```shell
   mysql>
   ```

2. 将 `root` 用户的认证插件改为 `mysql_native_password`，并设置密码为 `root`：

   ```sql
   ALTER USER 'root'@'localhost'
   IDENTIFIED WITH mysql_native_password BY 'root';
   ```

   （如果之前已经存在 `'root'@'%'` 用户，则需要改为对 `'root'@'%'` 执行以上命令，或者先删除旧的用户，再重新创建。）

3. 创建/修改 `root` 用户使其能从任意 IP 连接（即 `@'%'`），并设置密码：

   ```sql
   CREATE USER IF NOT EXISTS 'root'@'%'
       IDENTIFIED WITH mysql_native_password BY 'root';
   ```

或者，如果 `'root'@'%'` 已存在但你只想修改密码：

```
sql


  ALTER USER 'root'@'%'
  IDENTIFIED WITH mysql_native_password BY 'root';
```

4. 为 `root`@`%` 授予最大权限：

```
sql


  GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
```

5. 刷新权限：

```
sql


  FLUSH PRIVILEGES;
```

---

3. 配置防火墙，开放 3306 端口

若服务器开启了防火墙（如 `ufw` 或 iptables），还需要**允许**外部访问 MySQL 默认端口 **3306**：

- **使用 ufw（Ubuntu 常用）**：

```
bash


  sudo ufw allow 3306
```

- **使用 iptables**：

```
bash


  sudo iptables -A INPUT -p tcp --dport 3306 -j ACCEPT
```

（根据实际规则链、策略进行调整）

如果你的服务器在云平台上（例如 AWS、阿里云、腾讯云等），还需要在云控制台的安全组中开放 **3306** 端口。

---

4. 测试远程连接

1. 在远程主机上安装好 MySQL 客户端（或其他数据库管理工具）。
2. 执行类似下面的命令测试连接：

```bash
mysql -h <服务器公网IP或域名> -u root -p
```

3. 当提示输入密码时，输入 `root`（即上面设置的密码）。若连接成功，即表明一切配置正确。

---

5. 安全提示（非常重要）

- **弱密码**：设置 `root` 的密码为 `root` 非常危险，极易被暴力破解。只要数据库暴露在公网上，攻击者可轻松登录并获取数据或控制数据库。
- **开放 root 远程**：让 `root` 在任意地址可登录，更是给攻击者可乘之机；最安全的做法是使用普通账号做远程管理，`root` 仅限于在本地（`localhost`）或者 VPN/内部网络使用。
- **生产环境**：请务必使用强密码并限制访问来源（在防火墙、MySQL 用户、网络分段等层面做好安全防护）。最好在生产环境里只允许特定 IP 或 VPN 登录数据库，禁用 root 远程登录。

总结：

1. 修改 `mysqld.cnf` 中的 `bind-address` 为 `0.0.0.0`；
2. 将 `root` 用户认证方式改为 `mysql_native_password`，设置密码；
3. 给 `root`@`%` 授权并 **FLUSH PRIVILEGES**；
4. 开放 3306 端口；
5. 使用 `mysql -h <IP> -u root -p` 测试连接。

但请谨记，这样做在生产环境中是不推荐也不安全的，如果只是测试场景可以临时使用，务必做好防火墙限制或尽早改回更安全的做法。