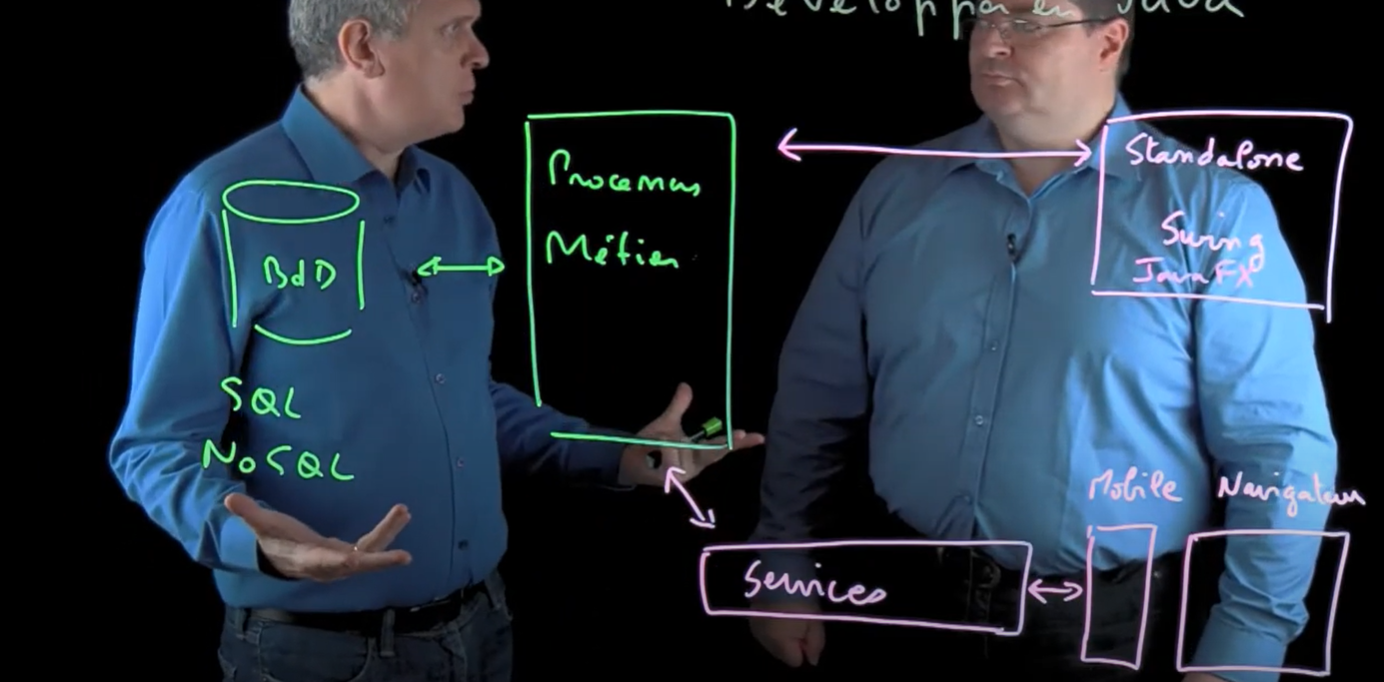
1. **Introduction JAVA**

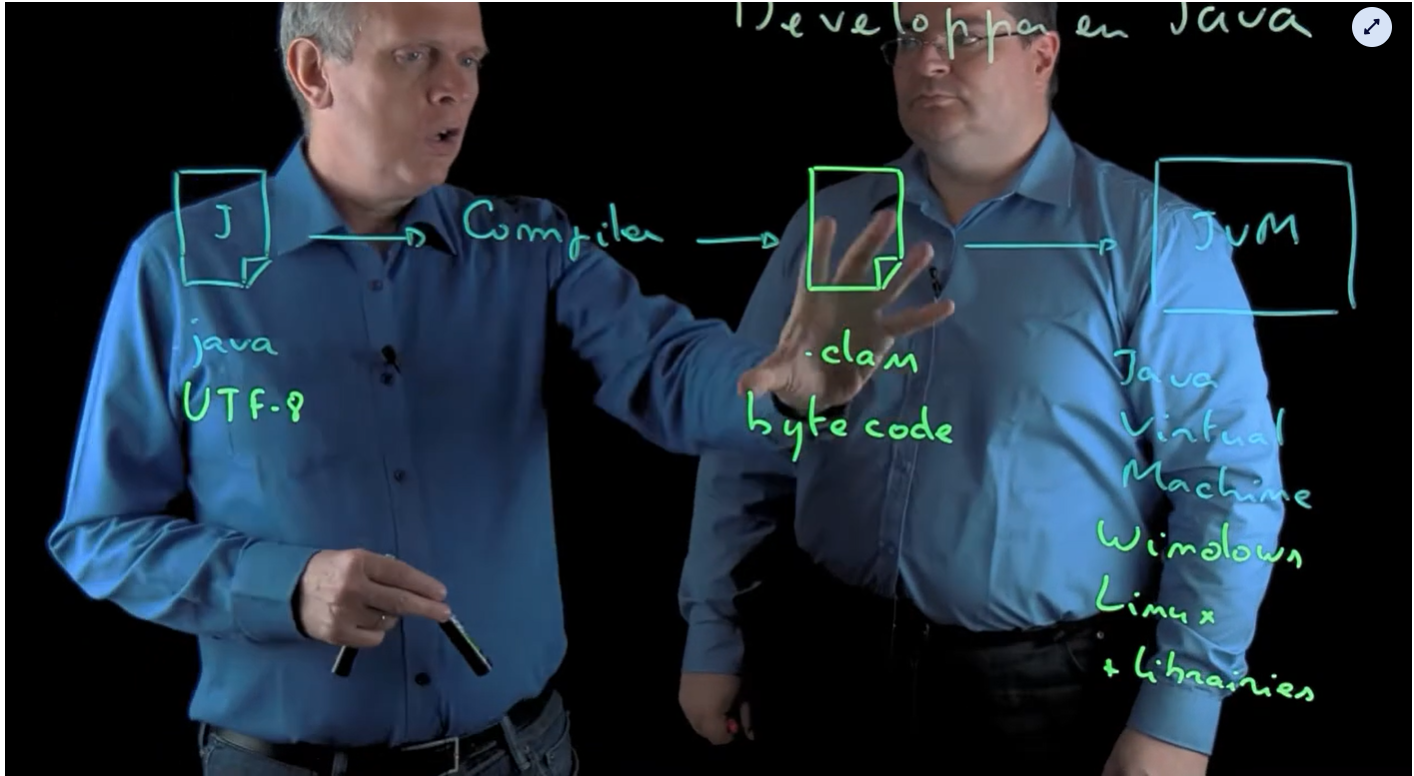
BDD 🡨----JDBC/Android Natif ----🡪 processus métier 🡨🡪 Services (SoAP/Rest) 🡨🡪 IHM stand alone sur le même ES (Java Swing) / Apps web services Web (HTML5/CSS3/JS) / Android (langage Java mais pas déploiement)



* JDK : Fournir un ensemble d’outils pour dévélopper les apps Java (écrire un programme Java)
* JRE : Outils pour exécuter les apps Java (RunTime)
* Java SE : API de base pour construire les app Java - client
* Java EE : App entreprise (s’appuie sur SE + API d’haut niveau ) - Serveur (SoAP)

Syntaxe proche du c et moteur Java (Write once run everywhere) : on va la compiler et le résultat de cette exécution on va le compiler sur différents plateformes (le binaire java est indépendant de la machine et l’OS)

-Déploiement code Java : UTF-8, langage java, après la compilation ça crée un fichier binaire .class qui contient du Byte Code pour exécuter le code sur des différents OS,   
ensuite on a la JVM, java virtual machine, et elle va s’appuyer sur des API et librairies



Eléments fournis par JDK : compilateur

Eléments fournis par JDE : JVM

Main.java 🡨🡪 Main.class

Pour compiler $ javac Main.class

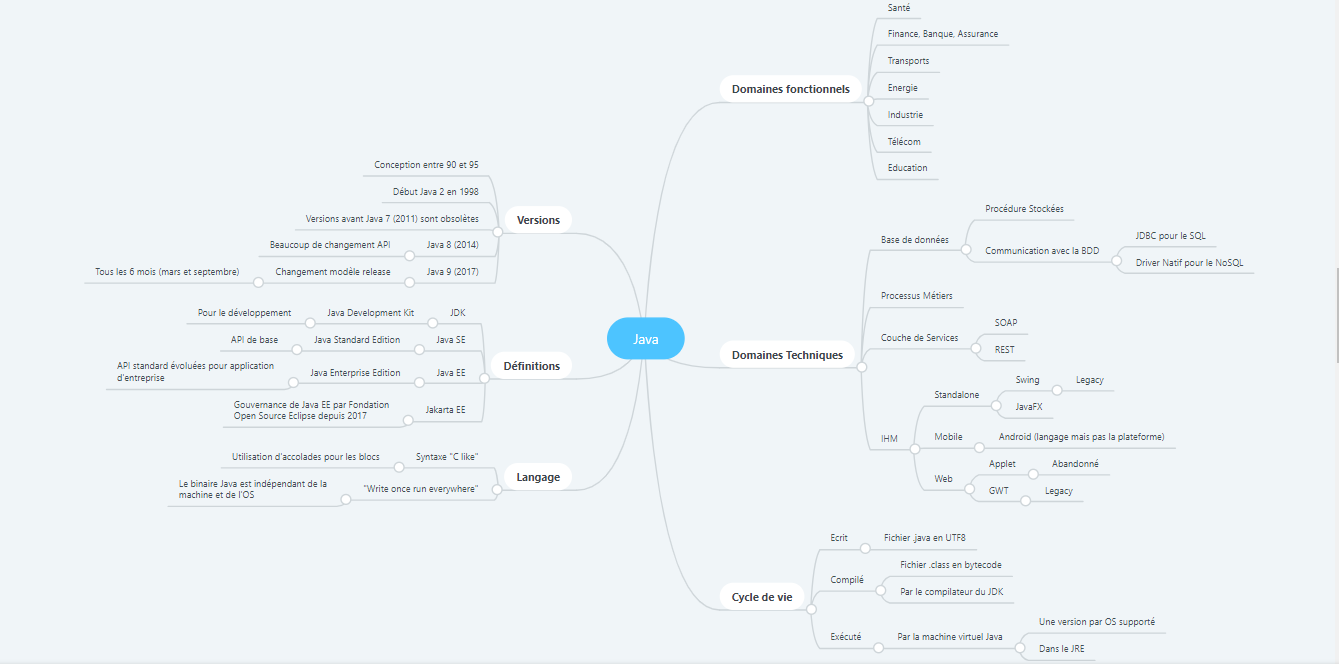
Pour executer $ java Main

$JAVA\_HOME/bim -> PATH

Vérifier la version de Java $ java –version

Systemln -> sortie + entrée

**RESUME**



* 1. **Classes abstraites (extends)**
* une classe abstraite peut étendre une autre classe abstraite ;
* une classe abstraite peut étendre une classe concrète ;
* une classe concrète qui étend une ou plusieurs classes abstraites (indirectement), doit obligatoirement fournir une implémentation pour toutes les méthodes abstraites existantes.
* Une classe abstraite ne peut pas être instanciée
* Une classe abstraite peut implémenter autant d'interfaces que l'on veut. Elle n'est pas tenue d'implémenter les méthodes des interfaces qu'elle utilise
  1. **Interfaces (implements)**

La notion d'interface est utilisée pour représenter des propriétés transverses de classes

* Une interface peut en étendre une autre, et même plusieurs.
* Elle ne peut pas étendre de classe, abstraite ou concrète.
* Une classe n'étend pas une interface, elle l'implémente
  1. **Différences**
* les classes abstraites servent à factoriser du code, tandis que les interfaces servent à définir des contrats de service. Et pourquoi ne pas utiliser des classes abstraites (dans lesquelles aucune méthode ne serait implémentée) en lieu et place des interfaces ? La réponse est simple : dans la plupart des langages actuels (c’est notamment le cas de Java, C#, PHP), il n’est possible pour une classe d’hériter que d’une seule classe parente (abstraite ou non), mais d’implémenter plusieurs interfaces.
  1. **API Stream**

L’API est construite sur un pattern de traitement de données : map / filter /reduce   
Exemple : utilisateur (map )🡪 Age (filter) 🡪 Moins de 20 ans (reduce) 🡪 moyenne

* Stream est un objet vide, il ne porte pas les données qu’il traite
* Map et Filter sont des méthodes intermédiaires parce qu’ils retournent un autre Stream
* Reduce est une méthode terminale car elle retourne une autre chose qu’un Stream

Déclaration d’opération :

* + 1. **Mapping ( fonction < T , R >)**

Chang le type

Conserve le nombre d‘objets

Conserve l’ordre des objets

* + 1. **Filter (Predicate <T>)**

Change pas le type

Ne conserve pas le nombre d’objets

Conserve l’ordre

* + 1. **Reduce (Bifonction )**

C’est comme une agrégation en SQL

* + 1. **Exemple :**

**1.4.4.1. Code standar :**

List<users>

Int count = 0 ;

Int num = 0 ;

For (user : users){

If(user.getAge() > 20 ){

count ++ ; 🡪 1 stockage Un seul Int

num += user.getAge() ; 🡪 2 stockage Un seul Int

}

}

Moyenne = num / count ; 🡪 3 stockage Un seul Int

Cette solution utlise moins de mémoire quand il y a beaucoup d’objets dans la liste

**1.4.4.2. Code pattern :**

ages = user.map( u -> u.getAge()) ; 🡪1 stockage tous les objets

agesFiltres = ages.filter(a -> a > 20) ; 🡪 2 stockage un pourcentage d’objets

moyenne = agesFiltres.average() ; 🡪 3 stockage un seul INT

Cette méthode utilise beaucoup de mémoire

**1.4.4.3. Stream :**

Users.stream() 🡪 stream <user>

.map(u –> u.getAge()) 🡪stream <Integer>

.filter(age. -> age > 20) 🡪stream <2eme Integer>

.reducer(…) 🡪Integer

Stream est un objet vide, il porte pas les données qu’il traite

**1.4.4.4. Traitement (méthode reduce) :**

.reducer(…) 🡪Integer

* + 1. **Construction de streams**

Stream.empthy()

Stream.of(« one », « two » , « three »)

Arrays.stream(tableauu)

intStream ints = intsRandom.ints()

longStream longs = intsRandom.longs()

* + - 1. **Stream d’objets**

Stream<t>

* + - * 1. **Stream de nombres**
* Faire un stream sur des primitifs est utilisé pour une raison de performance
* mapToObj : prendre les types primitifs et retourne un stream d’objet
* mapToInt, mapToLong, mapToDouble : prendre les objets et retourne un stream de primitif
* Il y a des méthodes de réduction dédiées à ces streams de nombres
* Stream.range(0,10) : Stream avec des valeurs de 0 à 9

List<String> liste = Stream.range(0,10) .mapToObj( index -> « bonjour »).collect(Collectors.toList()) 🡪 liste avec 10 fois « bonjour »

* + - 1. **Découper une chaine de caractères**
* Split(« ») ; 🡪 c’est une expression régulière
* Split va découper la chaine de caractères, faire à mesure qu’elle est consommée
  + 1. **Pattern**

Pattern pattern = Pattern.comple(« ») 🡪 découper une chaine de caractère et bien le placer surtout en occupation de mémoire

Stream<String> st2 = Pattern.splitAnStream(s) ;

* + 1. **n.char()🡪 (ASCII)**
    2. **Streams sur des fichiers texte**

Path path = Paths.set(« temp/file.txt »)

Stream<String> file = Files.lines(path,…(type d’encodage))

1. **SQL (Structured Query Language - langage de requête structurée)**

Le SGBD (Système de gestion de base de données) est le logiciel qui va vous permettre de manipuler les données d’une base

**Lancer mysql :** Sélectionnez MySQL Command Line Client. Cela va lancer un programme en ligne de commande. Tapez votre mot de passe et vous devriez avoir ceci

* 1. **CREATION D4UNE TABLE**

1. CREATE DATABASE nametable
2. USE nomdelabase;
3. CREATE TABLE utilisateur (

id INTEGER NOT NULL AUTO\_INCREMENT PRIMARY KEY,

nom VARCHAR(100),

prenom VARCHAR(100),

email VARCHAR(255) NOT NULL UNIQUE);

**Commande :**

CREATE DATABASE nomdelabase ; (créer basse de données)

USE nomdelabase ; (utiliser une base de données)

SHOW DATABASES ; (montrer toute les BDDs)

**Créer table avec paramètres directement :**

CREATE TABLE aliment (id INTEGER NOT NULL AUTO\_INCREMENT PRIMARY KEY,  
nom VARCHAR(100) NOT NULL,  
marque VARCHAR(100),  
sucre FLOAT,  
calories INTEGER NOT NULL,  
graisses FLOAT,  
proteines FLOAT,  
bio BOOLEAN DEFAULT false  
);

**Changer les paramètres :**

ALTER TABLE `SCHEMANAME`.`TABLE1`   
ADD COLUMN `FK\_COLUMN` BIGINT(20) NOT NULL,   
ADD CONSTRAINT `FK\_TABLE2\_COLUMN` FOREIGN KEY (`FK\_COLUMN`)   
REFERENCES `SCHEMANAME`.`TABLE2`(`PK\_COLUMN`);

**Vérifiez l'intégrité de votre table avec SHOW tables et SHOW columns**

SHOW tables;

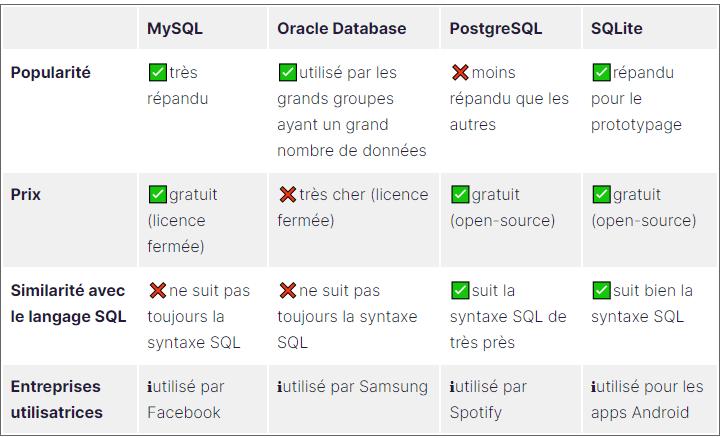
**Pour lire le schéma de vos tables**

SHOW COLUMNS FROM lenomdematable;

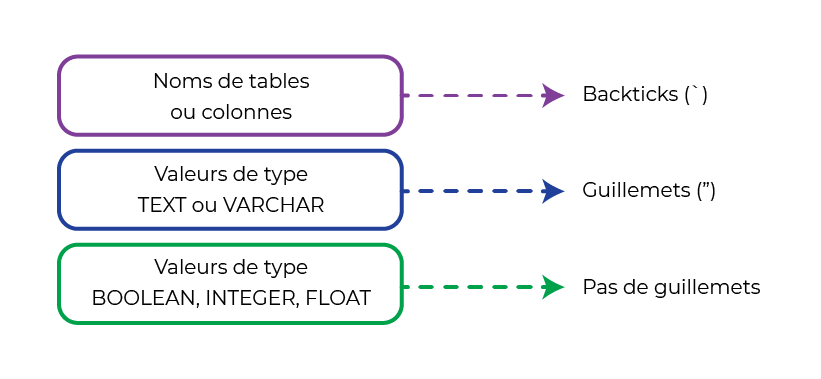
**Connecter =VBeaver à mysql :**

https://www.youtube.com/watch?v=FmPkZnOBCuo

**Differents SGBD**



Le langage SQL sert à communiquer entre votre application et votre base de données. Mais c’est le SGBD qui récupère la commande en SQL pour en sortir de la donnée depuis la base.



yta0F8PzOpRPp-uVxUrN-Q

**GLOASSAIRE**

**Processus métier** : un processus d'affaires, également appelé processus métier ou processus d'entreprise ou processus opérationnel, désigne un ensemble d'activités corrélées ou en interaction qui contribue aux finalités des affaires d'une organisation

**SQL :** Structured Query Language - langage de requête structurée

**SGBD** : Système de gestion de base de données

**JDK** : Java développement Kit

**JRE :** Java RunTime Enivrement

**Java SE :** Standar edition

**Java EE :** Enterprise edition

**JVM** : java virtual machine

**PROBLEMES ET SOLUTIONS**

MySQL Error 2003 (HY000): Can't connect to MySQL server on 'localhost:3306' (10061)  
1.1. Solution :  
1.1.1. Ensure the MySQL Server is running on your system :  
The first step is to verify that the server is running on your system. On your Windows terminal, use the command:

*$*mysqlshow

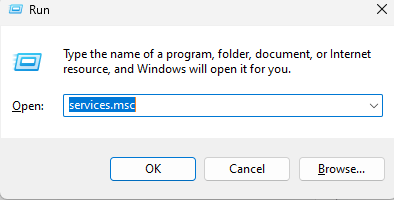
If you get the error shown below:

mysqlshow: Can't connect to MySQL server on 'localhost:3306' (10061)

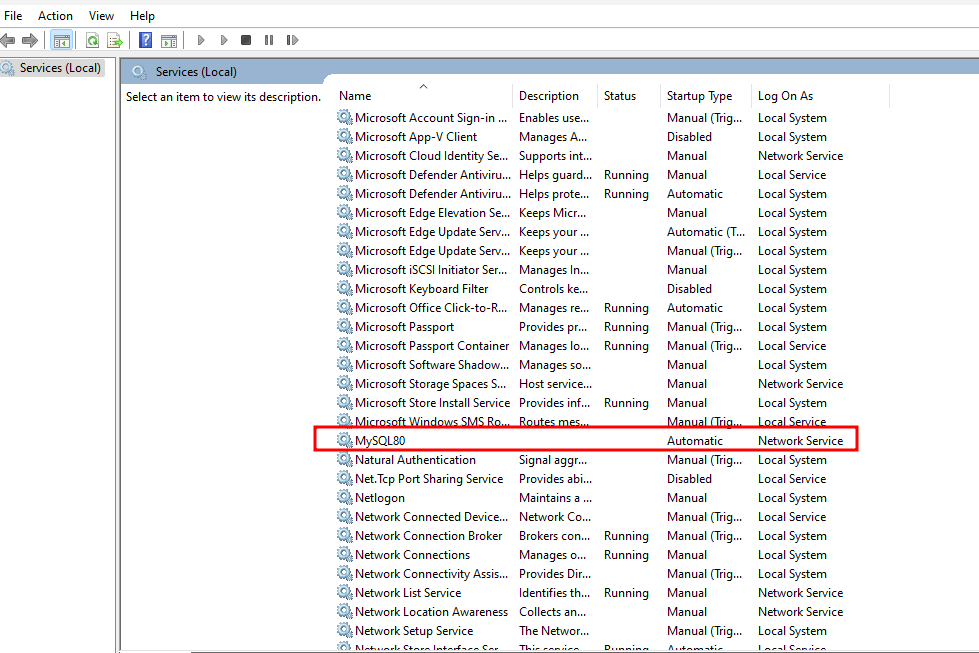
Then the Server is not running on your system.

To fix it, open the services manager on your Windows system. Press the Windows Key + R to launch the Run utility.

Next, enter “services.msc” and press enter.

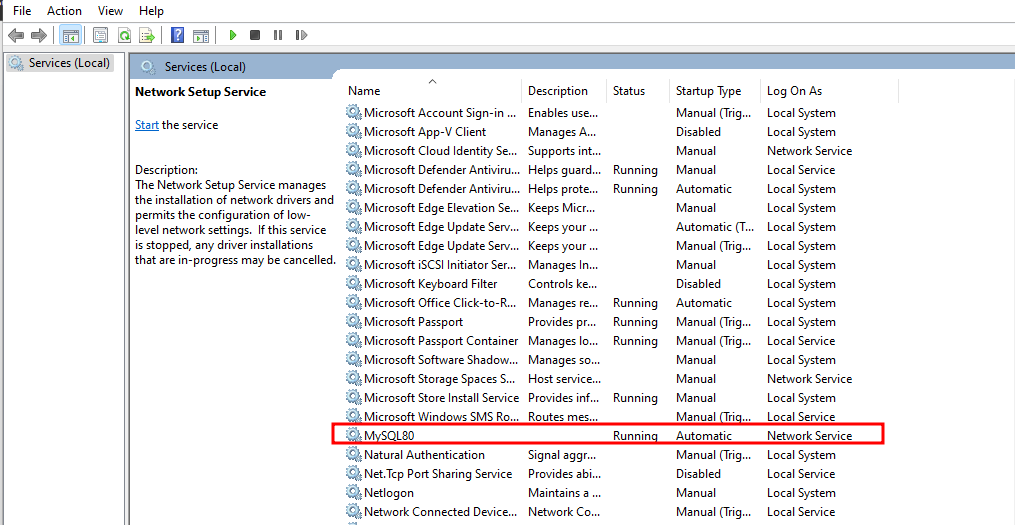


In the Services Manager, scroll down until you locate the service for your MySQL server. For example, if you have MySQL version 5.0, the service will be under ‘MySQL50’, and MySQL version 8.0 will be under ‘MySQL80’, etc.



We have MySQL version 8 installed in the picture above. However, the service is not running as the Status tab is empty.

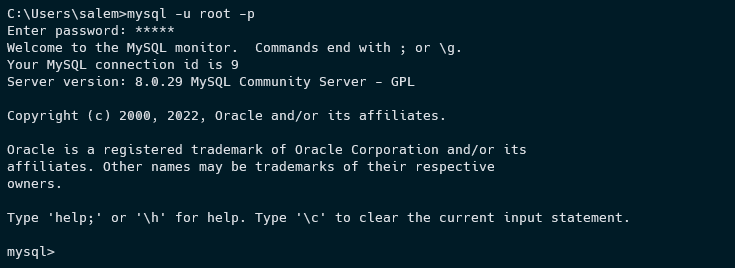
To fix this, right-click on the service name and press start. This should start the service and ensure the MySQL Server is running.



As you can see, the service status changes to ‘Running.’

Now we can connect to MySQL as:

*$*mysql -u root -p



In some cases, you may not find the MySQL service in the Services manager. In such a case, you will need to install the MySQL service.

Open the command prompt as an administrator and run the command:

*$*mysqld --install

https://linuxhint.com/wp-content/uploads/2022/06/Connect-to-MySQL-Server-6.png

The command above should install the MySQL service, allowing Windows to start and stop the service on startup.

* + 1. Ensure you specify the target port if the Server is running on a different port :

**Table 'tbl\_name' doesn't exist :**