BASE DE DATOS APLICADA

Instalación y Configuración de SGBD: MySQL

Claves para el despliegue y administración de bases de datos

¿Qué es un SGBD?

- Software que permite a los usuarios **crear, mantener y controlar el acceso** a las bases de datos.
- Actúa como una interfaz entre la base de datos y las aplicaciones/usuarios.
- **Funciones clave:** Almacenamiento, recuperación, seguridad, integridad, concurrencia, respaldo.

Ejemplo: Un SGBD es el sistema que organiza los datos, permite recuperarlos, asegura concurrencia y ayuda a encontrarlos rápidamente.

¿Cómo elegir el SGBD correcto?

- **Tipo de Base de Datos:** Relacional (SQL), NoSQL (documentos, grafos, clave-valor).
    - *En esta materia trabajamos con BD Relacionales y MySQL.*
- **Escalabilidad:** ¿Podrá manejar el crecimiento futuro de datos y usuarios?
- **Rendimiento:** ¿Qué tan rápido puede procesar las consultas?
- **Costos:** Licencias, hardware, mantenimiento, soporte.
- **Comunidad y Soporte:** Disponibilidad de documentación, foros, ayuda.
- **Integración:** Compatibilidad con otras tecnologías y sistemas.

Requisitos del Sistema (para MySQL)

- **Antes de instalar, verifica:**
  - **Sistema Operativo (SO):** Windows, Linux (distribuciones como Ubuntu, CentOS), macOS. MySQL es multiplataforma.
  - **Hardware:**
    - **CPU:** Procesador con suficientes núcleos para la carga esperada.
    - **RAM:** Suficiente memoria (ej. 4GB mínimo para desarrollo, 8GB+ para producción) para caché y operaciones.
    - **Almacenamiento:** Espacio en disco para la base de datos y logs. Preferiblemente SSD para rendimiento.
  - **Red:** Conectividad si es un servidor remoto.

Consideraciones: Los requisitos varían mucho según el tamaño de la base de datos y la concurrencia de usuarios. Un servidor de desarrollo es muy diferente a uno de producción.

Instalación y Configuración Inicial de MySQL

- **Windows**
  - **Descargar MySQL Installer:** Desde el sitio oficial de MySQL (dev.mysql.com/downloads/installer/).
  - **Elegir tipo de instalación:** Desarrollador, Servidor, Cliente. Seleccionar componentes.
  - **Configuración del servidor:**
    - **Tipo de Servidor:** Desarrollo, Servidor dedicado, Servidor de desarrollo.
    - **Puerto TCP/IP:** Por defecto, **3306**.
    - **Contraseña del usuario 'root':** Seguridad
    - **Nombre del servicio de Windows:** Cómo aparecerá en los servicios.
  - **Iniciar el servicio de MySQL.**

- Linux (Ubuntu)
  - sudo apt update
  - sudo apt install mysql-server
  - sudo mysql_secure_installation

Seguridad en MySQL

- **Seguridad**
- **Contraseña de root fuerte**
- **Creación de usuarios con privilegios mínimos:** No se debe usar 'root' para todo. Crea usuarios específicos para aplicaciones.
    - CREATE USER 'mi_app_user'@'localhost' IDENTIFIED BY 'password_segura';
    - GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_de_datos.* TO 'mi_app_user'@'localhost';
    - FLUSH PRIVILEGES;
- **Acceso remoto:** Restringir el acceso. Si es necesario, configurar firewall (puerto 3306).
    - GRANT ALL PRIVILEGES ON mi_base_de_datos.* TO 'otro_user'@'192.168.1.100' IDENTIFIED BY 'otra_password';
- **Encriptación:** Usar SSL/TLS para la conexión entre cliente y servidor.
- **Auditoría:** Registrar quién hace qué en la base de datos.

Bases de Datos de Usuario

- Las bases de datos que **contienen la información de tu aplicación o sistema.**
- **Creación:**
  - CREATE DATABASE nombre_de_mi_app_db;
  - Luego, seleccionar para usar:

    USE nombre_de_mi_app_db;

- **Tablas:** Son las estructuras donde se guardan los datos.
- **Tareas en la base de Datos:**
  - ingresar datos (INSERT)
  - actualizar datos (UPDATE)
  - borrar datos (DELETE)
  - consultar datos (SELECT)

- **Herramientas:** MySQL Workbench (GUI), línea de comandos (MySQL Shell/Client).

- Ejemplo:
  - **CREATE DATABASE nombre_de_mi_app_db;**
  - **USE nombre_de_mi_app_db;**
  - **CREATE TABLE Usuarios (**

    id INT PRIMARY KEY AUTO_INCREMENT,

    nombre VARCHAR(50),

    email VARCHAR(100) UNIQUE);
  - **INSERT INTO Usuarios (nombre, email) VALUES ('Juan Pérez', 'juan.perez@example.com');**
  - **SELECT * from Usuarios;**

Bases de Datos del Sistema

- Bases de datos internas de MySQL que gestionan el propio SGBD.
- **Principales bases de datos del sistema:**
  - **mysql:** Almacena información sobre usuarios, permisos, tablas del sistema. **No modificar manualmente**
  - **information_schema:** Contiene metadatos (información sobre la estructura de las bases de datos, tablas, columnas, etc.).
  - **performance_schema:** Recopila datos de rendimiento de MySQL.
  - **sys:** Proporciona un conjunto de objetos para ayudar a los administradores de MySQL.
- **Importancia:** Son vitales para el funcionamiento y la administración de MySQL. Se gestionan a través de comandos SQL estándar y herramientas de administración.

Monitorización y Mantenimiento

- **Monitorización:** Observar el rendimiento y estado de MySQL.
  - **Herramientas:** MySQL Workbench (dashboard de rendimiento), SHOW STATUS;, SHOW PROCESSLIST;, mytop, Nagios/Zabbix.
  - **Métricas clave:** Uso de CPU/RAM, conexiones activas, consultas lentas, uso de disco.
- **Mantenimiento:** Tareas para mantener la base de datos sana.
  - **Backups (Copias de seguridad):** ¡Esenciales! Regularmente.
    - mysqldump -u root -p nombre_db > backup_db.sql
  - **Optimización de consultas:** Usar EXPLAIN para analizar y mejorar el rendimiento de SQL.
  - **Optimización de tablas:** OPTIMIZE TABLE nombre_tabla;
  - **Limpieza de logs:** Gestionar los archivos de registro.

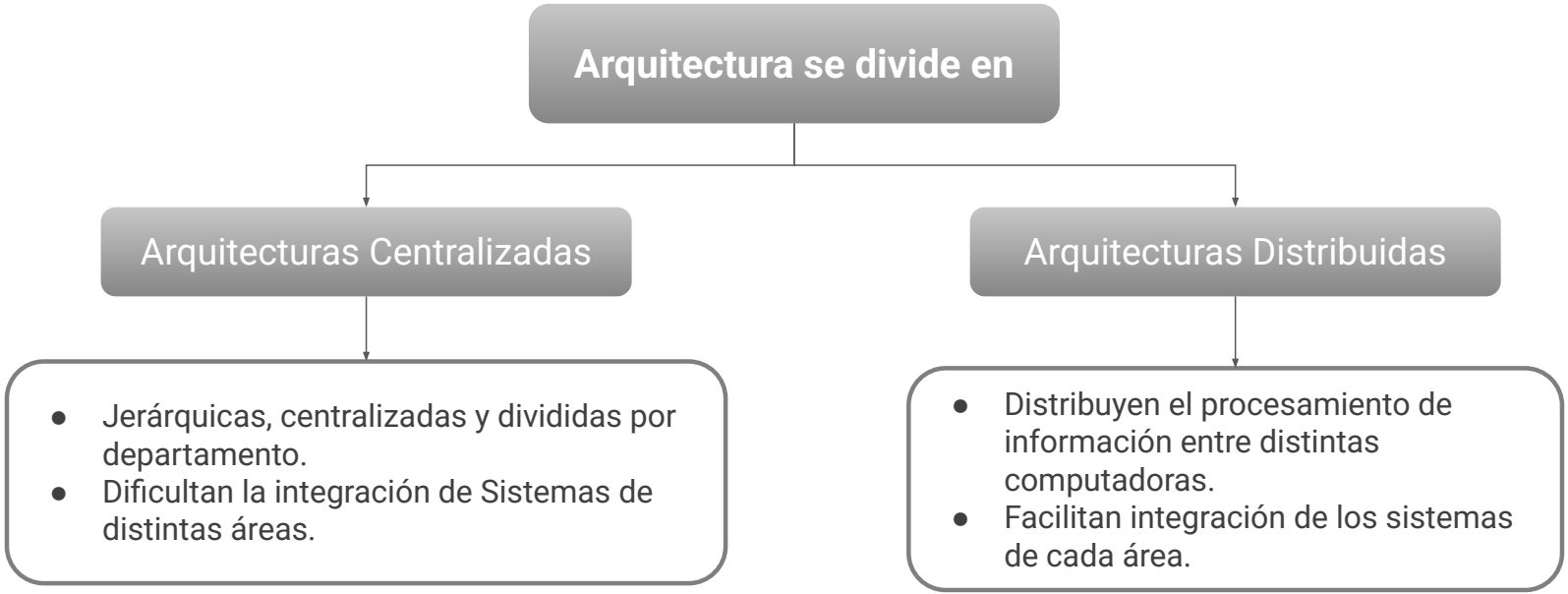
Pruebas y Ajustes (Tuning)

- **Pruebas:** Asegurar que la base de datos y las aplicaciones funcionen correctamente bajo carga.
  - **Pruebas de carga:** Simular muchos usuarios y transacciones simultáneas. (Ej: Apache JMeter).
  - **Pruebas de estrés:** Llevar el sistema al límite para ver cómo se comporta.
  - **Validación de datos:** Asegurar la integridad y precisión de la información.
- **Ajustes (Tuning):** Modificar la configuración de MySQL para mejorar el rendimiento.
  - **Archivo my.cnf (Linux) o my.ini (Windows):** Donde se configuran parámetros como innodb_buffer_pool_size, query_cache_size, etc.
  - **Optimización de índices:** Crear índices adecuados para acelerar las consultas.
    - CREATE INDEX idx_email ON Usuarios (email);

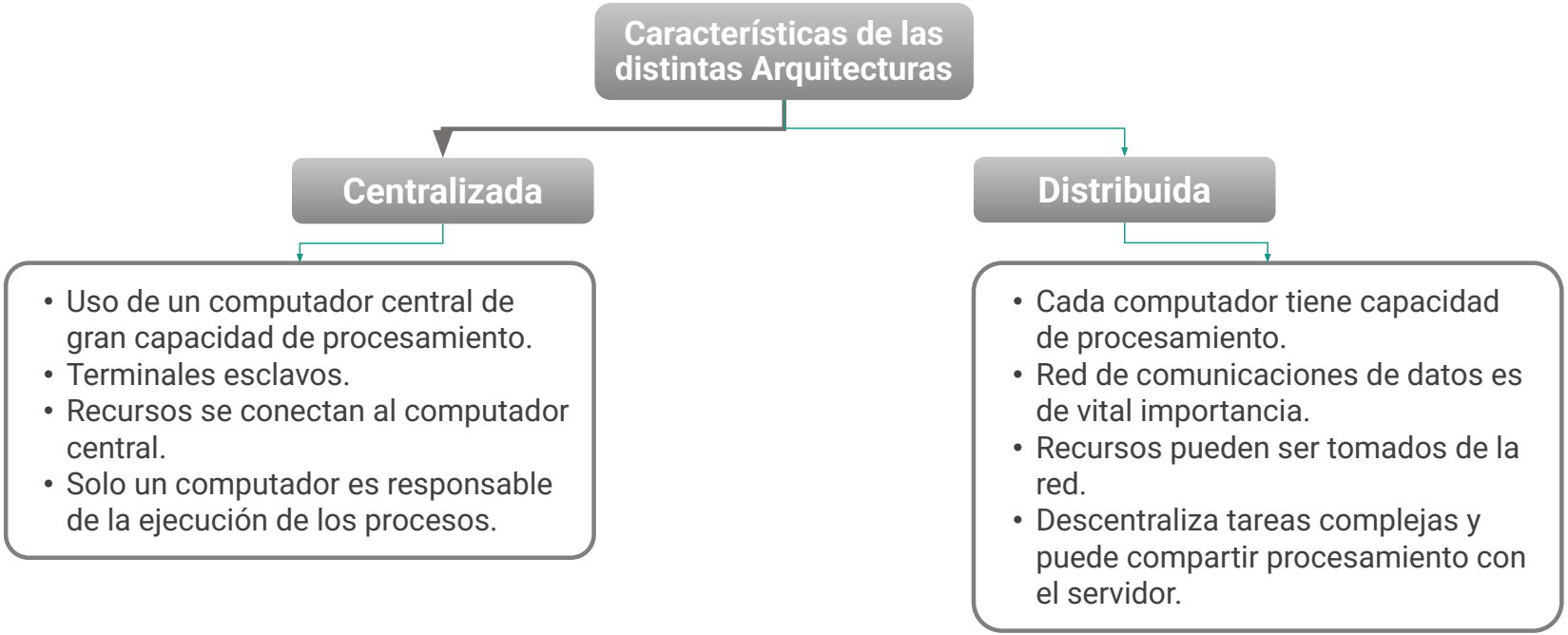
Arquitectura Cliente Servidor

- **Concepto:** Modelo de comunicación donde un **cliente** (una aplicación, un navegador) solicita recursos o servicios a un **servidor**.
- **En bases de datos:**
  - El **servidor de base de datos** (ej. MySQL) almacena y gestiona los datos.
  - El **cliente** (ej. una aplicación web, una aplicación de escritorio, MySQL Workbench) envía solicitudes (consultas SQL) al servidor y recibe los resultados.
- **Ventajas:** Centralización de datos, seguridad, concurrencia, escalabilidad.
- **Imagen:** Un diagrama con "Cliente" y "Servidor" conectados por una flecha, con un icono de base de datos en el servidor.

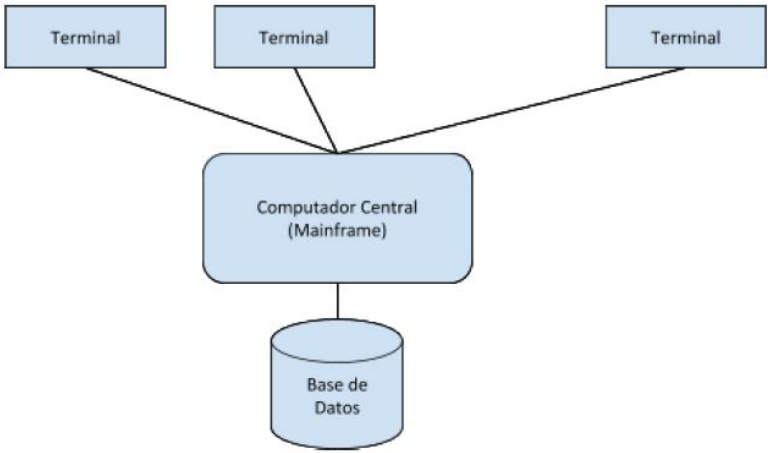
Arquitectura Cliente Servidor



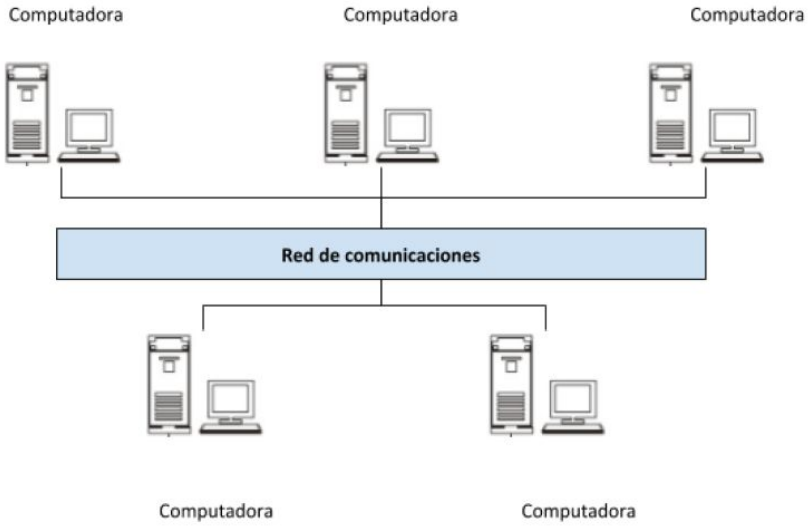
Arquitectura Cliente Servidor



Arquitectura Cliente Servidor



Esquema Arquitectura Centralizada



Esquema Arquitectura Distribuida.

Arquitectura Cliente Servidor

Arquitectura Centralizada

Ventajas	Desventajas
Implementa un único computador central con alta capacidad de procesamiento.	Computador central de hardware muy potente.
Alto nivel de seguridad.	Caída del servidor paraliza el sistema.
Compartición de Recursos.	Altamente dependiente de las comunicaciones.
Administración de un solo computador central.	

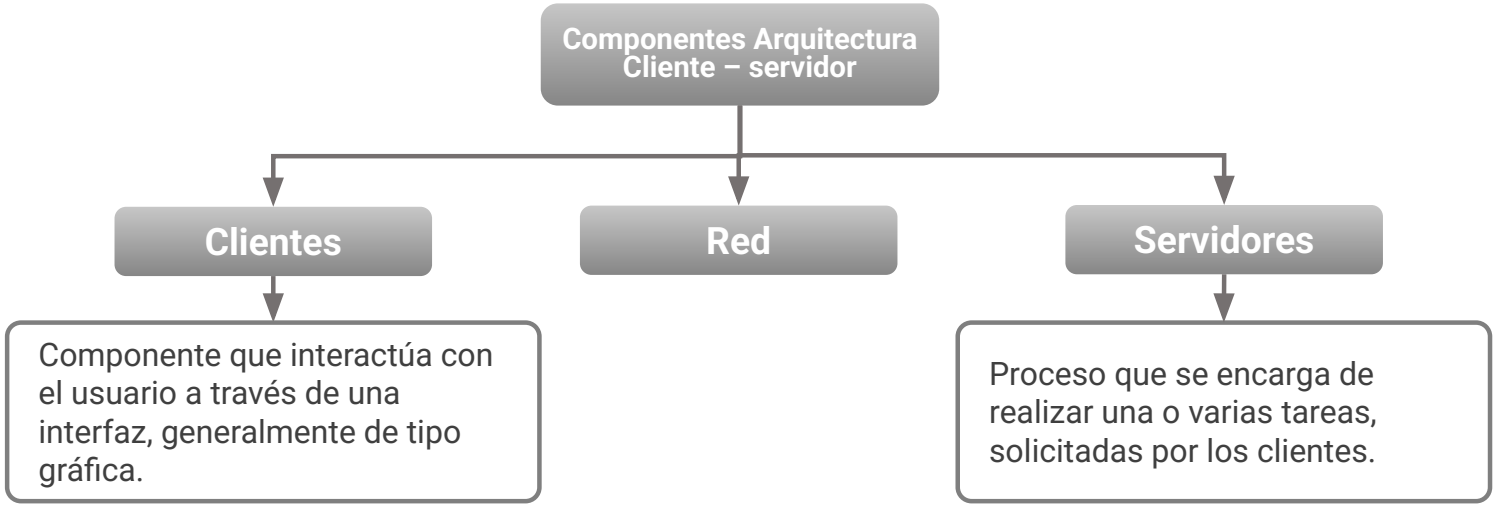
Arquitectura Distribuida.

Ventajas	Desventajas
Cada computadora tiene capacidad de procesamiento autónomo.	Altamente dependiente de la red de comunicaciones de datos.
Mayor independencia en los terminales al no tener todos los datos en un único computador.	No se puede descentralizar todo. Muchas tareas se delegan al servidor.
Facilita la integración a los sistemas.	Es de mayor complejidad.
Mejores tiempos de respuesta.	

Arquitectura Cliente Servidor

Arquitectura Cliente - Servidor

La Arquitectura Cliente - Servidor, permite modelar los sistema de información como un conjunto de servicios proporcionados por equipos servidores, los cuales son utilizados por los equipos clientes.



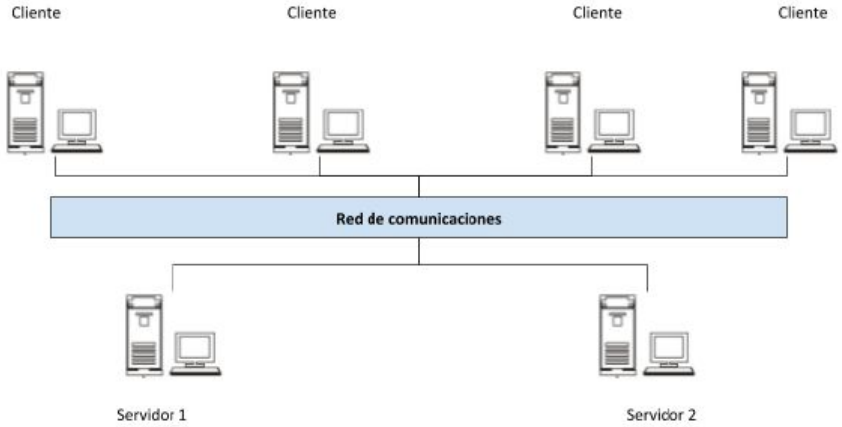
Arquitectura Cliente Servidor

Arquitectura Cliente Servidor

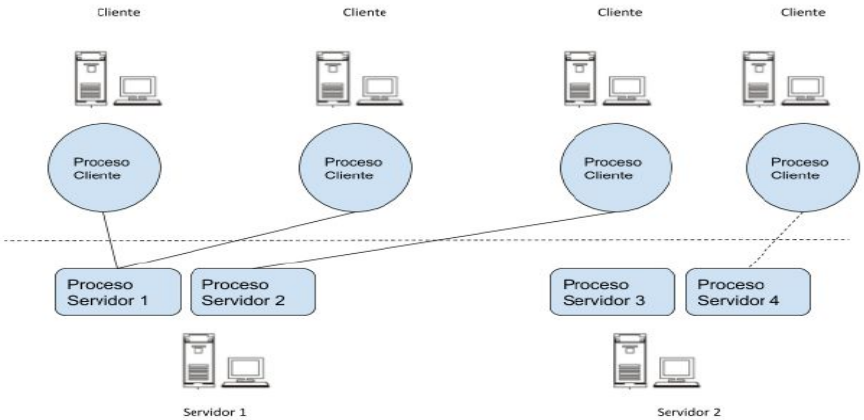
Cliente - Servidor

- Los clientes necesitan saber que servidores están disponibles, no así si hay clientes disponibles.
- Los clientes y servidores son procesos distintos.
- Varios procesos servidores pueden ejecutarse en un mismo servidor físico.

Esquema Físico



Esquema lógico



Conexión al Sistema de Gestión de Base de Datos

- **¿Cómo se conectan los clientes al servidor MySQL?**
  - **Protocolo:** Generalmente TCP/IP sobre el puerto 3306 (por defecto).
  - **Credenciales:** Se requiere un usuario y una contraseña válidos.
  - **Dirección:** La IP o el nombre de host del servidor MySQL.
- **Herramientas GUI:** MySQL Workbench, DBeaver, HeidiSQL.

- **Ejemplo (línea de comandos):**
  - `mysql -h localhost -u root -p` (Conexión local como root)
  - `mysql -h 192.168.1.50 -u mi_app_user -p mi_base_de_datos` (Conexión remota a una DB)

Conexión de Aplicaciones a Bases de Datos Locales y en la Nube

- **¿Cómo se conectan los clientes al servidor MySQL?**
  - **Protocolo:** Generalmente TCP/IP sobre el puerto 3306 (por defecto).
  - **Credenciales:** Se requiere un usuario y una contraseña válidos.
  - **Dirección:** La IP o el nombre de host del servidor MySQL.
- **Herramientas GUI:** MySQL Workbench, DBeaver, HeidiSQL.

- **Ejemplo (línea de comandos):**
  - `mysql -h localhost -u root -p` (Conexión local como root)
  - `mysql -h 192.168.1.50 -u mi_app_user -p mi_base_de_datos` (Conexión remota a una DB)

Conexión de Aplicaciones a Bases de Datos Locales y en la Nube

- **Aplicaciones locales:**
  - La aplicación se conecta al servidor MySQL instalado en la misma máquina o en otro servidor dentro de la red local.
  - La cadena de conexión especifica localhost o la IP interna (192.168.x.x).


```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost", # O la IP interna del servidor
  user="mi_app_user",
  password="password_segura",
  database="nombre_de_mi_app_db"
)
print("Conectado a la BD local!")
mydb.close()
```

Conexión de Aplicaciones a Bases de Datos Locales y en la Nube

- **Aplicaciones en la nube (ej. AWS RDS, Google Cloud SQL, Azure Database for MySQL):**
  - La base de datos es un servicio gestionado por un proveedor de la nube.
  - La cadena de conexión utiliza un **endpoint (URL)** proporcionado por el servicio en la nube.


```
import mysql.connector
mydb = mysql.connector.connect(
  host="tu-endpoint-rds.amazonaws.com", #
O el endpoint del servicio cloud
  user="mi_app_user_cloud",
  password="password_segura_cloud",
  database="nombre_de_mi_app_db_cloud"
)
print("Conectado a la BD en la nube!")
mydb.close()
```

Conclusión

- **IMPORTANTE:**
- Correcta instalación
- Configuración
- Mantenimiento de un SGBD como MySQL.

Determinan:

- Rendimiento
- Seguridad
- Fiabilidad de cualquier aplicación que dependa de datos.

- Seguridad y respaldo son tan importantes como la instalación inicial.