Data Store
Setup, Querying, and View Population

- Downloads: https://downloads.mysql.com/archives/workbench/
- Version 8.0.15

Prereq: Environment Variables for Secrets

- Our server-side app runs in two different contexts: **development** and **production**
    - Industrial apps can have additional contexts, such as **test** and **staging**
- Ideally, variables specific to a context are "passed in" rather than hard-coded
    - What port to listen on?
    - What username and password to connect to the database with?
- The easiest way to pass a variable to a program is via **environment variables**
- There are many ways to manage environment variables, we'll look at a common, modern way

Using a .env File

- We'll store our project's context-specific variables in a .env file
- This file *should not* be added to our project's repo because it stores secrets
    - So, let's add a '.env' line to the .gitignore file to tell git not to include it
- Then, let's create a file named '.env' and add some variables to it:
    - NODE_ENV=development
    - PORT=1234
- Finally, we need to add a library and initialize it in our project to read these vars:
    - npm package: https://www.npmjs.com/package/dotenv
    - $ npm install --save dotenv
    - Add code at the top of our server's initialization:

      ```
      // Read environment variables
      import * as dotenv from "dotenv";
      dotenv.config();
      ```
- Then we need to go replace our hard-coded 1234 with: process.env.PORT and test.
- We'll also need to add a .env file to our server's project folder before deploying.

Installing a Relational Database Management System (RDBMS)

- We'll choose MySQL, a popular open source database
- Install using aptitude, Ubuntu's package manager (app store):
    - $ sudo apt install mysql-server
- Configure using the included installer:
    - $ sudo mysql_secure_installation
    - Options (we're using these to simplify our process, you should use safer options in a real production server):
        - Secure password checker: **No**
        - Root password: **Choose a password you know** (perhaps your computer's password or ONYEN)
        - Remove anonymous users: **Yes**
        - Disable remote root login: **No**
        - Remove test database: **Yes**
        - Reload table privileges: **Yes**

Setting up a new Database

- $ sudo mysql
    - Begins an interactive MySQL prompt.
- mysql> SHOW databases;
    - A SQL command to list the databases managed by MySQL
- mysql> CREATE DATABASE blog;
    - Creates a new database named blog

-
  ```
  mysql> GRANT ALL PRIVILEGES ON blog.*
                TO 'blog_app'@'%'
                IDENTIFIED BY 'choose_a_password';
  ```
    - Create a user, also named 'blog', who can access the database, and connect from any IP '@%'
- MySQL is another server daemon running on your cloud machine listening on port 3306.
- We need to open the firewall to access.

Accepting Outside Connections

- First, MySQL server needs to be configured to listen on all IP addresses (currently only on listening on 127.0.0.1/localhost)
    - $ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
    - Look for the line: bind-address = 127.0.0.1
    - Change to: bind-address = *
- Open the AWS EC2 console: https://console.aws.amazon.com/ec2/
- Select your running instance
- In its detail description, you'll see "Security Groups" and yours was setup via the Launch Wizard, click on "launch-wizard-N"
- We need to add an inbound rule to permit inbound connections for MySQL:
    - Add Rule: MySQL/Aurora
    - For source: 0.0.0.0/0
- Save

Test the Connection

- Let's try connecting from MySQL Workbench!
- Create a new connection
    - Host: your server's IP
    - Username: blog_dev
    - Schema/Database: blog_dev
    - Password: what you chose
- Save the connection once it works
- (And let's go add and test a production database and user, as well.)

Creating a Table

- Let's create a **table** in the **blog_dev** database named **todos**
- We'll setup this table with a few columns:
    - id: INT primary key, auto increment
    - item: Text
    - url: Text
- Let's add a few rows, as well

Shortcuts We're Taking that You Shouldn't

- For the purposes of simplifying our development process this week, we're taking a few shortcuts you generally should not.

1. You should not open up access to your database for *any* IP to connect to.

2. You should run a development database separate from a production database (ideally on your development machine in a container/ VM)

3. You should *probably* use a database as a service (DBaaS) such as Amazon RDS rather than self-installing and managing a database.

   Database Administrator (DBA) can be a full-time job and career path.

Adding Database Settings to .env

- Let's now add additional variables to our .env files. We'll need to do this both locally in our machine and on the server.
- Local:

```
MYSQL_HOST=<your-server-ip>
MYSQL_USER=blog
MYSQL_DB=blog
MYSQL_PASSWORD=<your-password>
```

- Server:

```
MYSQL_HOST=localhost
MYSQL_USER=blog
MYSQL_DB=blog
MYSQL_PASSWORD=<your-password>
```

Connecting to a Data Store from Code

- Now that your database is *running*
- Database vendors (or the open source community) will provide libraries in most popular programming languages for interacting with their database
- We'll use the open source MySQL2
    - NPM Package: https://www.npmjs.com/package/mysql2
    - npm install --save mysql2
    - npm install --save-dev types/mysql2

Add TypeScript File for DB Connection

Querying