Railway Deployment Documentation

ExpressJS + MySQL Database

1. Introduction

Railway is a cloud deployment platform that allows you to deploy backend applications and databases directly from GitHub with minimal configuration.

This documentation explains how to deploy an **ExpressJS backend** with a **MySQL database** on Railway.

2. Tech Stack

- Backend: ExpressJS (Node.js)
- Database: MySQL
- Deployment Platform: Railway
- Version Control: GitHub

3. Project Structure

```
backend-express/
├── index.js
├── package.json
├── .env
└── .gitignore
```

4. package.json

```
{
  "name": "express-mysql-railway",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "mysql2": "^3.6.0"
  }
}
```

5. ExpressJS + MySQL Code (index.js)

```
const express = require("express");
const mysql = require("mysql2");
const cors = require("cors");
require("dotenv").config();

const app = express();
app.use(cors());
app.use(express.json());

// MySQL Connection
const db = mysql.createConnection({
  host: process.env.MYSQL_HOST,
  user: process.env.MYSQL_USER,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE,
  port: process.env.MYSQL_PORT
});

db.connect(err => {
  if (err) console.log("Database Error:", err);
  else console.log("MySQL Connected Successfully");
});

// Test API
app.get("/", (req, res) => {
  res.send("Express + MySQL running on Railway");
});

// Fetch users
app.get("/users", (req, res) => {
  db.query("SELECT * FROM users", (err, result) => {
    if (err) return res.status(500).json(err);
    res.json(result);
  });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

6. Create MySQL Database on Railway

1. Login to Railway
2. Click **New Project**
3. Select **Provision MySQL**

4. Railway automatically creates:
    - Database
    - Username
    - Password
    - Host
    - Port

Open the **MySQL plugin** → **Variables tab** to view credentials.

---

7. Environment Variables (Railway)

Add the following variables in **Railway** → **Variables**:

```
MYSQL_HOST=containers-xxxx.railway.app
MYSQL_USER=root
MYSQL_PASSWORD=your_password
MYSQL_DATABASE=railway
MYSQL_PORT=3306
```

Note: Never hardcode database credentials in code.

---

8. Create Table in Railway MySQL

Use the Railway MySQL Console and execute:

```sql
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);

INSERT INTO users (name, email)
VALUES ('Yogesh', 'yogesh@gmail.com');
```

---

9. Deploy Express App on Railway

1. Push your project to GitHub
2. Open Railway Dashboard
3. Click **New Project**
4. Choose **Deploy from GitHub Repo**
5. Select your repository

Railway will automatically:

- Install dependencies

- Run `npm start`
- Generate a public URL

---

10. Test APIs

Base URL:

```
https://your-project-name.up.railway.app
```

Endpoints:

```
GET /
GET /users
```

Test using browser or Postman.

---

11. Common Issues & Fixes

App crashes on start

- Ensure PORT is read from `process.env.PORT`

Database connection error

- Check MySQL plugin is running
- Verify environment variables

Works locally but not on Railway

- Do not hardcode port numbers
- Use Railway environment variables

---

12. Best Practices

- Use mysql2 connection pooling
- Handle errors properly
- Keep `.env` file in `.gitignore`
- Use Railway Variables in production

---

13. Use Cases

- Student projects
- Internship demos

- Full-stack applications
- React + Express + MySQL projects

---

14. Conclusion

Railway provides a simple and powerful way to deploy ExpressJS applications with MySQL databases. With minimal setup, you can move from local development to production-ready deployment quickly.