Chapter 13 – do it practically

**Chapter 11 - Load Balancers, Clustering in Node.js, or Authentication + Rate Limiting? – To be discussed Later**

**Chapter 10 –**

Concurrency in nodejs

process object details i8n node js

**File Descriptors (fds) and Socket Descriptors**

**File Descriptors (FDs)** are integral to Unix-like operating systems, including Linux and macOS. They are used by the operating system to manage open files, sockets, and other I/O resources.

**Socket descriptors** are a special type of file descriptor used to manage network connections. They are essential for network programming, allowing processes to communicate over a network.

**Event Emitters**

**Event Emitters** are a core concept in Node.js, used to handle asynchronous events. They allow objects to emit named events that can be listened to by other parts of the application. The Eve ntEmitter class is provided by the Node.js events module. Here's a brief overview:

* **Creating an EventEmitter**: You create an instance of EventEmitter and use the on method to register event listeners.
* **Emitting Events**: Use the emit method to trigger events and pass data to listeners.
* **Handling Events**: Listeners (functions) handle the emitted events and perform actions based on the event data.

**Streams**

**Streams** in Node.js are objects that facilitate reading from or writing to a data source in a continuous fashion. Streams are particularly useful for handling large amounts of data efficiently.

**Buffers**

**Buffers** are used to handle binary data in Node.js. They provide a way to work with raw memory allocations and are useful for operations involving binary data, such as reading files or network communications.

**Pipes in Node.js**

**Pipes** in Node.js are a powerful feature for managing the flow of data between streams. They simplify the process of reading from a readable stream and writing to a writable stream, facilitating efficient and seamless data processing.

Node.js provides both **synchronous (blocking)** and **asynchronous (non-blocking)** file operations through the built-in fs module.

* Synchronous APIs: Ends with Sync (e.g., readFileSync)
* Asynchronous APIs: Uses callbacks or Promises (e.g., readFile, promises.readFile)