# A First Video Streaming Microservice

To acquire the train training material:

git clone https://github.com/RichardNaoufal/a\_first\_video\_streaming\_microservice.git

## Step 1: Install Necessary Tools

### Install Node.js

Node.js is the runtime environment for JavaScript-based microservices.

Check if Node.js is installed:

node --version

Check if npm is installed:

npm –version

### 1.2 Install VS Code

A recommended IDE for editing and running the microservice code.

Download from: <https://code.visualstudio.com> (already installed)

## Step 2: Create the Project Directory

Manually create a folder for your microservice and navigate into it:

mkdir video-streaming-microservice  
cd video-streaming-microservice

## Step 3: Setup the Node.js Project

Initialize a new Node.js project inside the folder:

npm init -y

## Step 4: Install Dependencies

### 4.1 Install Express

Express is used to create the HTTP server.

npm install --save express@5.0.0-beta.1

**NOTE**:

* Express v5 beta supports asynchronous route handlers
* (except Express v4 and earlier will handle asynchronous errors a little differently).

## Step 5: Create the Microservice Code

Inside the `video-streaming-microservice` folder, create a new folder named `src`:

mkdir src

Inside `src`, create a new file named `index.js` and add the following code:

const express = require("express");  
const fs = require("fs");  
  
//  
// Throws an error if the PORT environment variable is missing.  
//  
if (!process.env.PORT) {  
 throw new Error("Please specify the port number for the HTTP server with the environment variable PORT.");  
}  
  
//  
// Extracts the PORT environment variable.  
//  
const PORT = process.env.PORT;  
// Creates an instance of an Express “app”  
const app = express();  
  
//  
// Registers a HTTP GET route for video streaming.  
//  
app.get("/video", async (req, res) => {  
 const videoPath = "./videos/SampleVideo\_1280x720\_1mb.mp4";  
 const stats = await fs.promises.stat(videoPath);  
  
 res.writeHead(200, {  
 "Content-Length": stats.size,  
 "Content-Type": "video/mp4",  
 });  
 fs.createReadStream(videoPath).pipe(res);  
});  
  
//  
// Starts the HTTP server.  
//  
app.listen(PORT, () => {  
 console.log(`Microservice listening on port ${PORT}, point your browser at http://localhost:${PORT}/video`);  
});

## Step 6: Run the Microservice

### 6.1 Set the Required Environment Variable

The microservice requires a `PORT` environment variable to be set before running.

Linux/macOS:

export PORT=3000

Windows:

set PORT=3000

### 6.2 Run the Microservice

Run the microservice:

node src/index.js

Open a browser and visit: <http://localhost:3000/video>

You should see a streamed video being played.

“Ctrl+C” will stop the server

### Live Reloading for Fast Iteration

For a live reload in development mode, we use a package called nodemon: Watches for code changes and restarts our microservice at every change of code

npm install --save-dev nodemon

### 6.4 Add the npm ‘start:dev’ script to ‘package.json’

This is to be able to run the project with “npm start.”

"scripts": {

"start:dev": "nodemon src/index.js"

},

=> the following command starts the microservice in development mode:

npm run start:dev

## Step 7: Setup for Production

When preparing for production, install dependencies without development tools:

npm install --omit=dev

### 7.1 Add the npm ‘start’ script to ‘package.json’

This is to be able to run the project with “npm start.”

"scripts": {

"start": "node src/index.js"

},

=> the following command starts the microservice in production mode:

npm run start