**IISNode + Express Deployment Guide (Known Good Configuration)**

**1️⃣ Folder Structure**

C:\inetpub\wwwroot\JBG\_SITES\node\_app\

│

├── server\_fs.js <-- Main Node.js/Express app

├── package.json

├── public\ <-- Optional frontend assets

├── web.config <-- IISNode + rewrite rules

└── iisnode\ <-- Auto-created log folder

**2️⃣ server\_fs.js (Minimal Express Example)**

const express = require("express");

const path = require("path");

const app = express();

// Serve static files

app.use("/node\_app", express.static(path.join(\_\_dirname, "public")));

// Simple test route

app.get("/node\_app/api", (req, res) => {

console.log("✅ API hit!");

res.json({ message: "Hello from Node.js via IISNode!" });

});

// Example FeatureServer route

app.get("/node\_app/FeatureServer/0", (req, res) => {

res.json({ type: "FeatureLayer", features: [] });

});

// IISNode listens on a named pipe

app.listen(process.env.PORT, () => {

console.log("🚀 Node.js app listening via IISNode");

});

**3️⃣ web.config (Proven Working)**

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.webServer>

<!-- IISNode handler -->

<handlers>

<add name="iisnode" path="server\_fs.js" verb="\*" modules="iisnode" />

</handlers>

<rewrite>

<rules>

<!-- Skip frontend path -->

<rule name="SkipFrontend" stopProcessing="true">

<match url="^frontend(/.\*)?$" ignoreCase="true" />

<action type="None" />

</rule>

<!-- Skip static assets -->

<rule name="StaticFiles" stopProcessing="true">

<match url=".\*\.(html|htm|css|js|png|jpg|jpeg|gif|ico|svg|json)$" ignoreCase="true" />

<action type="None" />

</rule>

<!-- Catch-all to Node -->

<rule name="NodeApp" stopProcessing="true">

<match url=".\*" ignoreCase="true" />

<action type="Rewrite" url="server\_fs.js" appendQueryString="true" />

</rule>

</rules>

</rewrite>

<security>

<requestFiltering>

<hiddenSegments>

<add segment="node\_modules" />

<add segment="iisnode" />

</hiddenSegments>

</requestFiltering>

</security>

<!-- IISNode settings -->

<iisnode

nodeProcessCommandLine="C:\Program Files\nodejs\node.exe"

loggingEnabled="true"

devErrorsEnabled="true"

logDirectory="iisnode"

/>

</system.webServer>

</configuration>

**4️⃣ Testing**

1. Copy files to: C:\inetpub\wwwroot\JBG\_SITES\node\_app\
2. Run iisreset
3. In browser:
   * https://www.jbgap4.com/node\_app/api → { "message": "Hello from Node.js via IISNode!" }
   * https://www.jbgap4.com/node\_app/FeatureServer/0?f=json → { type: "FeatureLayer", features: [] }

**5️⃣ Tips (Ya Baba Wisdom)**

* Always prefix routes with /node\_app/... to work under IIS rewrite.
* Use .\* catch-all unless you need strict regex rules.
* For static files, change HTML paths to /node\_app/...
* Logs are in: C:\inetpub\wwwroot\JBG\_SITES\node\_app\iisnode

**✅ This configuration is tested and works on older IIS versions as well.**

Yes — you can (and should) load credentials.json from a folder outside the IIS web root to keep it secure and inaccessible to the public.

**Steps:**

1️⃣ **Move credentials.json**  
Place it outside C:\inetpub\wwwroot, for example:

D:\secure\_keys\credentials.json

2️⃣ **Update your Node.js code**

const auth = new google.auth.GoogleAuth({

keyFile: 'D:/secure\_keys/credentials.json',

scopes: ['https://www.googleapis.com/auth/spreadsheets']

});

3️⃣ **Set Permissions**  
Grant read permissions to the IIS AppPool user (e.g., IIS\_IUSRS) for the folder containing the file.

4️⃣ **Why this matters**  
Anything in C:\inetpub\wwwroot can be served by IIS if not explicitly blocked. Storing credentials outside the web root prevents accidental exposure.

✅ **Rule of Thumb:** Always keep secrets, keys, and tokens in a secure folder outside the public web root.