Mini Replit Clone (Updated) — Docker + Nix + Express

🎯 Goal (Updated)

We are building a mini Replit-style system where:

- Each project runs in an isolated Docker container
- Environment is reproducible using Nix
- Express.js is used as control API
- When creating a project, user sends dependencies (express, mongoose, etc.)
- Dependencies are automatically installed inside container

🏗️ High-Level Architecture

User → Express API → Workspace Folder → Docker Container → Nix → npm install → node index.js

---

📁 Folder Structure

mini-replit/ | ├── api/ | └── server.js | ├── docker/ | ├── Dockerfile | └── workspaces/

1️⃣ Nix Environment

File: docker/flake.nix

```
{
  description = "Node.js Dev Environment";

  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-23.11";
  };

  outputs = { self, nixpkgs }:
    let
      system = "x86_64-linux";
      pkgs = import nixpkgs { inherit system; };
    in
```

```
    {
      devShells.${system}.default = pkgs.mkShell {
        buildInputs = [
          pkgs.nodejs_18
          pkgs.nodePackages.npm
        ];
      };
    };
}
```

This guarantees:

- Node 18
- npm
- Reproducible environment

2️⃣ Dockerfile

File: docker/Dockerfile

```
FROM nixos/nix

WORKDIR /workspace

RUN mkdir -p /etc/nix &&
    echo "experimental-features = nix-command flakes" >> /etc/nix/nix.conf

COPY flake.nix /workspace/

RUN nix develop --command true

EXPOSE 3000

CMD ["nix", "develop"]
```

Build image:

```
docker build -t mini-replit-node ./docker
```

3️⃣ Express API (UPDATED — Dynamic Dependencies)

File: api/server.js

```
import express from "express";
import { exec } from "child_process";
import { v4 as uuid } from "uuid";
import fs from "fs";
import path from "path";

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

const WORKSPACE_DIR = path.join(process.cwd(), "../workspaces");

// CREATE PROJECT
app.post("/create", async (req, res) => {
  try {
    const { dependencies = {} } = req.body;

    const projectId = uuid();
    const projectPath = path.join(WORKSPACE_DIR, projectId);

    fs.mkdirSync(projectPath, { recursive: true });

    // Create package.json dynamically
    const packageJson = {
      name: projectId,
      version: "1.0.0",
      type: "module",
      main: "index.js",
      dependencies
    };

    fs.writeFileSync(
      path.join(projectPath, "package.json"),
      JSON.stringify(packageJson, null, 2)
    );

    // Create starter index.js
    fs.writeFileSync(
      path.join(projectPath, "index.js"),
      `
import express from "express";

const app = express();
```

```
app.get("/", (req, res) => {
  res.json({ message: "Project ${projectId} running" });
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
});
`
    );

    res.json({ projectId });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// RUN PROJECT
app.post("/run/:id", async (req, res) => {
  try {
    const projectId = req.params.id;
    const projectPath = path.join(WORKSPACE_DIR, projectId);

    if (!fs.existsSync(projectPath)) {
      return res.status(404).json({ error: "Project not found" });
    }

    const containerName = `repl-${projectId}`;

    const command = `
    docker run -d
      --name ${containerName}
      -v ${projectPath}:/workspace
      -p 0:3000
      --memory=256m
      --cpus=0.5
      --pids-limit=100
      mini-replit-node
      nix develop --command sh -c "if [ ! -d node_modules ]; then npm
install; fi && node index.js"
    `;

    exec(command, (err, stdout, stderr) => {
      if (err) return res.status(500).send(stderr);
      res.json({ message: "Project running", containerName });
    });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.listen(4000, () => {
```

```
  console.log("Mini Replit API running on port 4000");
});
```

📦 How To Create Project (API Example)

POST /create

```
{
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^7.6.0"
  }
}
```

This automatically creates:

package.json with express + mongoose

When /run is called:

1. Docker container starts
2. Workspace mounted
3. Nix loads Node 18
4. If node_modules missing → npm install
5. express + mongoose installed
6. node index.js runs

🔥 What Just Happened?

We built:

- Dynamic dependency injection per project
- Automatic npm install inside container
- Cached node_modules (won't reinstall every run)
- Resource-limited sandbox

🚀 Production Improvements (Next Level)

- Replace exec() with dockerode
- Add container cleanup
- Add persistent Docker volumes

- Add WebSocket terminal
- Add port detection
- Add project stop endpoint
- Add auth + multi-tenant org support

✅ Final Result

You now have a dynamic mini Replit system where:

- User sends dependencies as JSON
- System generates package.json
- Docker + Nix loads environment
- npm installs dependencies
- App runs in isolated container

This is real sandbox architecture used in cloud IDE systems.