Now let's add a script to set up language-specific development environments:

```javascript name=create-devenv.js

#!/usr/bin/env node

const fs = require('fs');

const path = require('path');

const { execSync } = require('child\_process');

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

const askQuestion = (query) => new Promise((resolve) => rl.question(query, resolve));

const templates = {

python: {

dockerfile: `FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["python", "app.py"]`,

files: {

'requirements.txt': 'flask\npandas\nnumpy\nmatplotlib\nscikit-learn',

'app.py': `from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from DevPod!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8000, debug=True)`,

'.devcontainer/devcontainer.json': `{

"name": "Python Development",

"dockerFile": "../Dockerfile",

"extensions": [

"ms-python.python",

"ms-python.vscode-pylance"

],

"settings": {

"python.linting.enabled": true

}

}`

}

},

node: {

dockerfile: `FROM node:18-slim

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

CMD ["node", "index.js"]`,

files: {

'package.json': `{

"name": "devpod-node-app",

"version": "1.0.0",

"description": "Node.js app on DevPod",

"main": "index.js",

"scripts": {

"start": "node index.js",

"dev": "nodemon index.js"

},

"dependencies": {

"express": "^4.18.1"

},

"devDependencies": {

"nodemon": "^2.0.20"

}

}`,

'index.js': `const express = require('express');

const app = express();

const port = process.env.PORT || 3000;

app.get('/', (req, res) => {

res.send('Hello from DevPod Node.js!');

});

app.listen(port, () => {

console.log(\`App listening at http://localhost:\${port}\`);

});`,

'.devcontainer/devcontainer.json': `{

"name": "Node.js Development",

"dockerFile": "../Dockerfile",

"extensions": [

"dbaeumer.vscode-eslint",

"esbenp.prettier-vscode"

],

"settings": {

"editor.formatOnSave": true

}

}`

}

},

java: {

dockerfile: `FROM openjdk:17-slim

WORKDIR /app

COPY . .

RUN javac App.java

CMD ["java", "App"]`,

files: {

'App.java': `public class App {

public static void main(String[] args) {

System.out.println("Hello from DevPod Java!");

}

}`,

'.devcontainer/devcontainer.json': `{

"name": "Java Development",

"dockerFile": "../Dockerfile",

"extensions": [

"vscjava.vscode-java-pack",

"redhat.java"

]

}`

}

}

};

async function createDevEnvironment() {

console.log('\x1b[36m%s\x1b[0m', '╔══════════════════════════════════════════════╗');

console.log('\x1b[36m%s\x1b[0m', '║ DevPod Environment Creator ║');

console.log('\x1b[36m%s\x1b[0m', '╚══════════════════════════════════════════════╝');

try {

const projectName = await askQuestion('Project name: ');

if (!projectName) {

console.error('Project name is required');

return;

}

console.log('\nAvailable templates:');

Object.keys(templates).forEach((key, i) => {

console.log(`${i + 1}. ${key}`);

});

const templateIndex = parseInt(await askQuestion('\nSelect template number: '));

const templateKey = Object.keys(templates)[templateIndex - 1];

if (!templateKey || !templates[templateKey]) {

console.error('Invalid template selection');

return;

}

const projectDir = path.join(process.cwd(), 'workspace', projectName);

// Create project directory

if (!fs.existsSync(projectDir)) {

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

}

if (!fs.existsSync(path.join(projectDir, '.devcontainer'))) {

fs.mkdirSync(path.join(projectDir, '.devcontainer'), { recursive: true });

}

// Write Dockerfile

fs.writeFileSync(

path.join(projectDir, 'Dockerfile'),

templates[templateKey].dockerfile

);

// Write template files

for (const [filename, content] of Object.entries(templates[templateKey].files)) {

const filePath = path.join(projectDir, filename);

const dirPath = path.dirname(filePath);

if (!fs.existsSync(dirPath)) {

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

}

fs.writeFileSync(filePath, content);

}

console.log(`\n\x1b[32m✓ Created ${templateKey} project in ${projectDir}\x1b[0m`);

console.log('\nTo open this project in DevPod:');

console.log(`1. Navigate to http://localhost:8080`);

console.log(`2. Open folder: /home/coder/project/${projectName}`);

// Add database connection example

if (templateKey === 'python') {

const dbExample = `

# Example database connection

import psycopg2

def get\_db\_connection():

conn = psycopg2.connect(

host='postgres',

database='devpod',

user='devpod',

password='your\_secure\_password'

)

return conn

`;

fs.appendFileSync(path.join(projectDir, 'db\_example.py'), dbExample);

fs.appendFileSync(path.join(projectDir, 'requirements.txt'), '\npsycopg2-binary');

console.log('Added database connection example in db\_example.py');

}

const deployYaml = `version: '3'

services:

${projectName}:

build:

context: .

ports:

- "8000:8000"

restart: unless-stopped

labels:

- "traefik.enable=true"

- "traefik.http.routers.${projectName}.rule=Host(\`${projectName}.localhost\`)"

`;

fs.writeFileSync(path.join(projectDir, 'docker-compose.deploy.yml'), deployYaml);

console.log('Added deployment configuration in docker-compose.deploy.yml');

} catch (error) {

console.error('Error creating development environment:', error);

} finally {

rl.close();

}

}

createDevEnvironment();