**Step by Step Tutorial for Creating a Poetry Project with FastAPI**

**Introduction:**

Poetry is a powerful dependency management tool for Python projects, offering several advantages over traditional methods. Before we dive into the tutorial, let's explore three key advantages of using Poetry:

1. **Simplified Dependency Management**: Poetry simplifies the process of managing project dependencies by providing a single tool for dependency resolution, installation, and version control.
2. **Virtual Environments**: With Poetry, each project gets its own isolated virtual environment by default. This ensures that project dependencies are kept separate from other projects and system-wide Python installations, reducing conflicts and ensuring consistency.
3. **Deterministic Builds**: Poetry generates a **poetry.lock** file, which locks the dependencies to specific versions. This guarantees that all developers and environments using the project will have the exact same set of dependencies, leading to more predictable and reliable builds.

Now, let's proceed with the tutorial:

1. **Install pipx**:
   * **pipx** is a tool used to install and run Python applications in isolated environments.
   * Open your terminal and run:

pip install pipx

* + After installation, verify the version to ensure it's installed correctly:

pipx --version

1. **Create a Python Environment**:
   * This step ensures that your Python environment is properly set up.
   * Run the following command to set up the Python environment:

pipx ensurepath

1. **Install Poetry**:
   * Poetry simplifies dependency management for Python projects.
   * Install Poetry using **pipx**:

pipx install poetry

* + Verify the installation:

bashCopy code

1. **Initialize a Poetry Project**:
   * Create a new directory for your project and initialize a Poetry project within it:

poetry new my\_first\_api

* + Replace **my\_first\_api** with your preferred project name.
  + This command creates a new directory with the project structure and necessary files.

1. **Understanding Project Structure**:
   * Navigate to the project directory (**my\_first\_api/**).
   * **my\_first\_api/pyproject.toml**: Contains project metadata and dependencies managed by Poetry.
2. **Add Dependencies**:
   * Navigate to your project directory (**my\_first\_api**) and add FastAPI and Uvicorn as dependencies:

poetry add fastapi uvicorn

* + This command updates the **pyproject.toml** file with the added dependencies.

1. **Understanding main.py**:
   * Create a new Python file named **main.py** within the **my\_first\_api** directory.
   * Populate **main.py** with the provided code.
   * Comments within the code explain each line's purpose and functionality.
2. **Run the Project**:
   * Execute the following command to run your FastAPI application:

poetry run uvicorn my\_first\_api.main:app --reload

* + This command starts the Uvicorn server with your FastAPI application and enables automatic reloading of the server upon code changes.

1. **Testing Your Application**:
   * Open your web browser and navigate to **https://localhost:8000/** to view the "Hello World" message.
   * Append **/city** or **/country** to the URL to view corresponding data.
2. **Accessing Documentation**:
   * FastAPI generates interactive documentation for your API automatically.
   * Access it by appending **/docs** to the URL: **https://localhost:8000/docs**.

**Conclusion:**

Congratulations! You have successfully created your first FastAPI project with two routes: **/city** and **/country**. By following this tutorial, you've gained insight into the benefits of using Poetry for Python dependency management and learned how to create a basic FastAPI application. Keep exploring and building amazing projects with Python and Poetry!

postgresql://hamidneondb\_owner:\*\*\*\*\*\*\*\*\*\*\*\*@ep-weathered-shadow-a18hgbjw.ap-southeast-1.aws.neon.tech/hamidneondb?sslmode=require

Project name : neon\_project01

Database : hamidneondb

main

