Set up a test environment for the API that mimics the production environment database setup.

1. Environment Setup in OpenShift

- **Namespace**: Use the provided namespace in OpenShift for the test environment.
- **Database Deployment**: The test environment's database will be deployed using the same Crunchy Postgres Operator version 5.4.3 and Postgres version 15 as the production environment. This ensures consistency between the test and production environments.

2. Database Schema and Data Migration

- **Schema Duplication**: Replicate the production database schema in the test environment. This can be achieved by exporting the schema from the production database and applying it to the test database.
- **Data Migration**:
  - **Data Sampling**: Given the large number of rows in the production tables, consider copying a representative subset of data to the test environment. For example:
    - **housing**: 150,000 rows (10% of production data).
    - **housing_type**: All 10 rows.
    - **ownership**: 175,000 rows (10% of production data).
    - **owners**: 125,000 rows (10% of production data).
  - **Data Import**: Use database dump tools (pg_dump and pg_restore) or custom scripts to export and import the data. Ensure that foreign key relationships and constraints are maintained.

3. Synchronization with Production

- **Data Sync Tools**: Use tools like pg_dump for schema and data export from the production environment and pg_restore for importing into the test environment.
- **Automated Sync Process**: If regular updates are needed, consider setting up a cron job or an OpenShift CronJob to periodically sync data from production to the test environment.

4. Resource Management in OpenShift

- **Database Resources**: Allocate sufficient CPU, memory, and storage resources in OpenShift to handle the test data. This should be scaled down compared to production, but adequate to perform meaningful tests.
- **Persistent Volume Claims (PVCs)**: Set up PVCs in OpenShift to ensure data persistence for the Postgres database.

5. Validation and Testing

- **Data Integrity**: After the database setup, validate the data integrity by running sanity checks to ensure that the data in the test environment closely mimics the production environment.
- **Application Testing**: Deploy the API in the same test namespace and conduct functional and performance tests to ensure the environment is a suitable replica of production.

Tools and Commands

- **pg_dump**: To export schema and data from the production database.
- **pg_restore**: To import schema and data into the test environment.
- **OpenShift CLI (oc)**: To manage resources in OpenShift, including deploying the Postgres database and configuring namespaces.