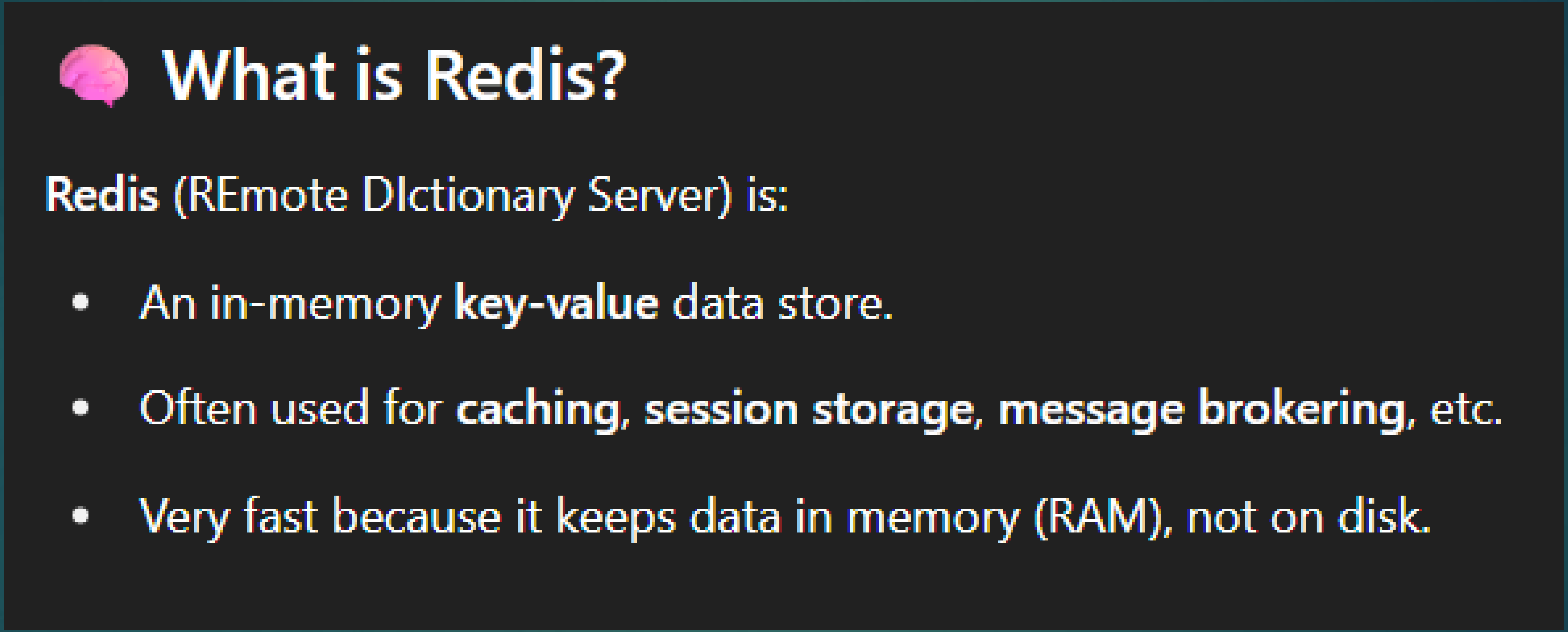
spring

EraaSoft



🧠 What is Redis?

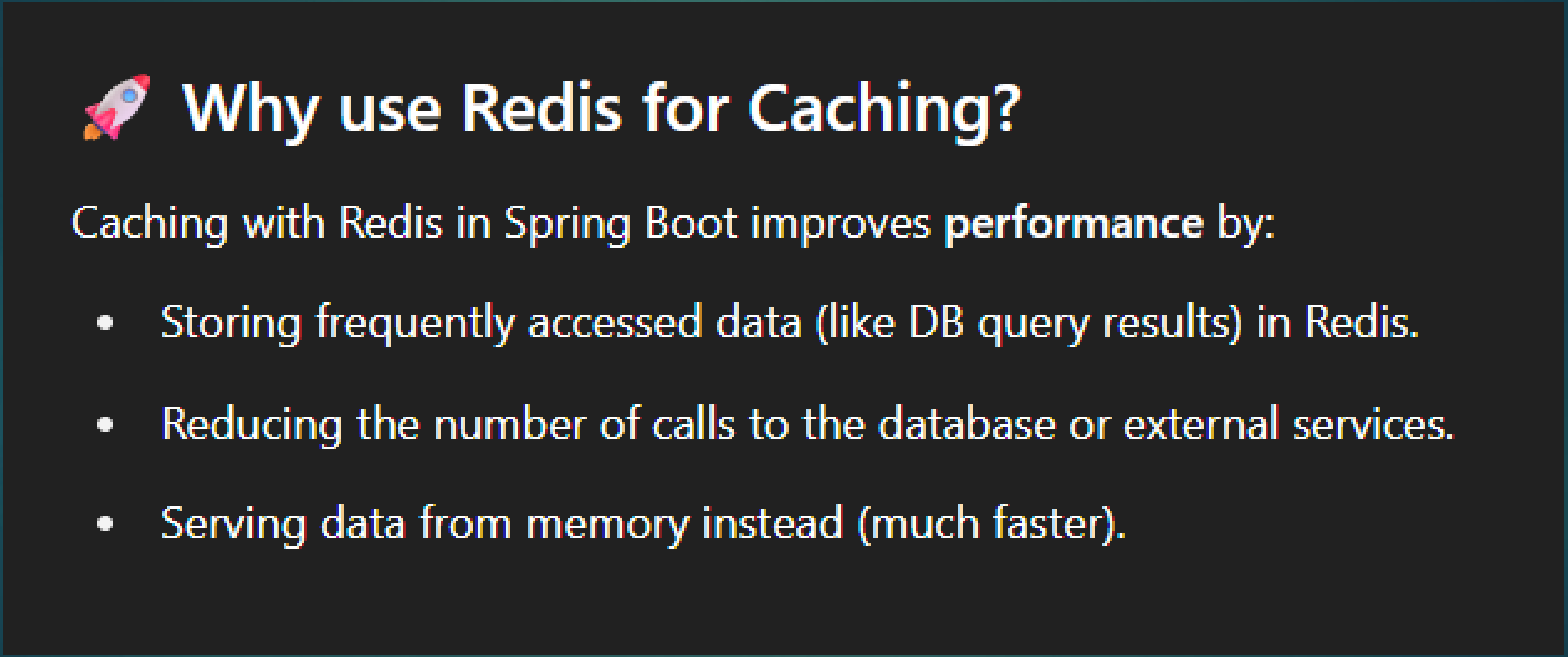
Redis (REmote DIctionary Server) is:

- An in-memory **key-value** data store.
- Often used for **caching**, **session storage**, **message brokering**, etc.
- Very fast because it keeps data in memory (RAM), not on disk.

By Eslam Khder

spring

EraaSoft

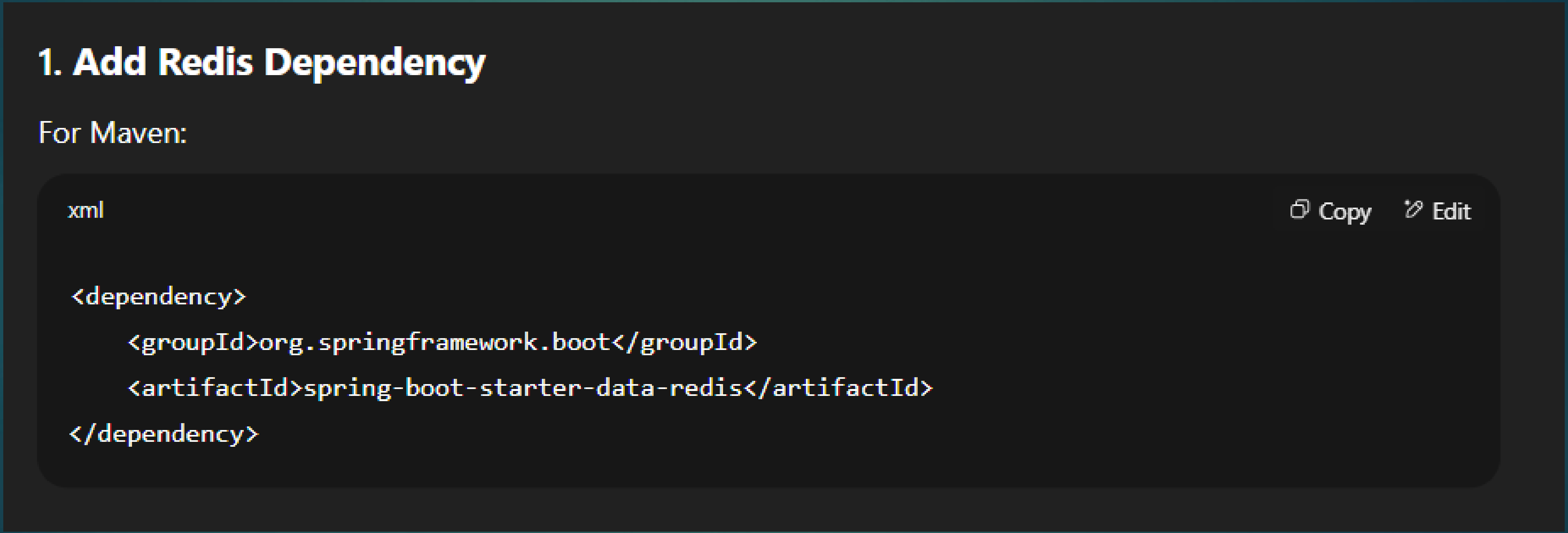


🚀 Why use Redis for Caching?

Caching with Redis in Spring Boot improves **performance** by:

- Storing frequently accessed data (like DB query results) in Redis.
- Reducing the number of calls to the database or external services.
- Serving data from memory instead (much faster).

By Eslam Khder

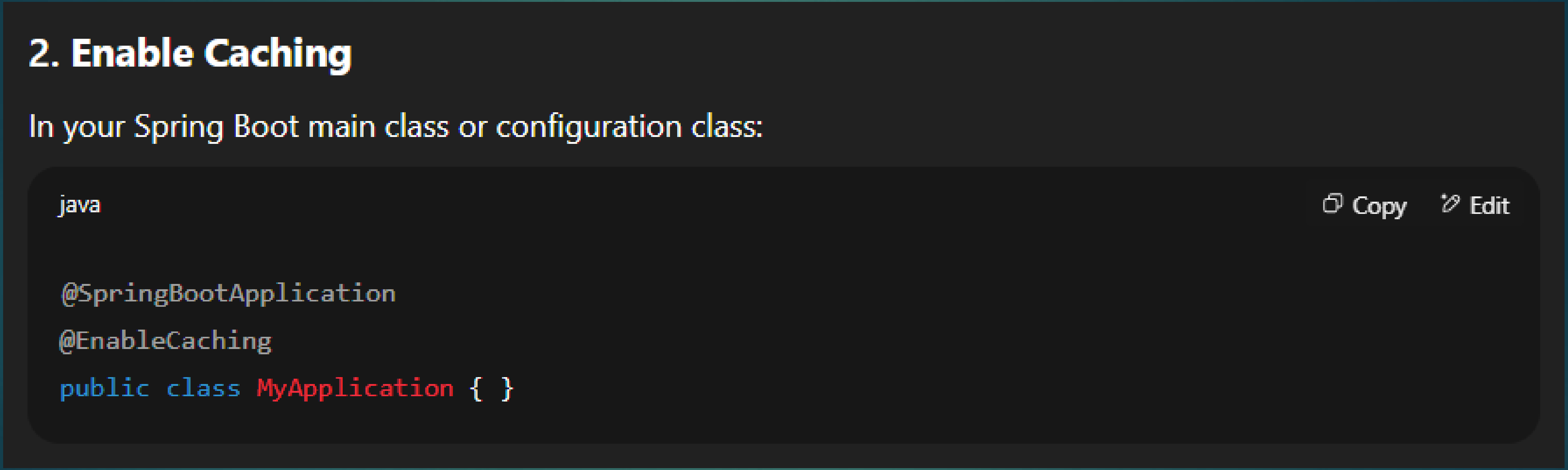


1. Add Redis Dependency

For Maven:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

By Eslam Khder

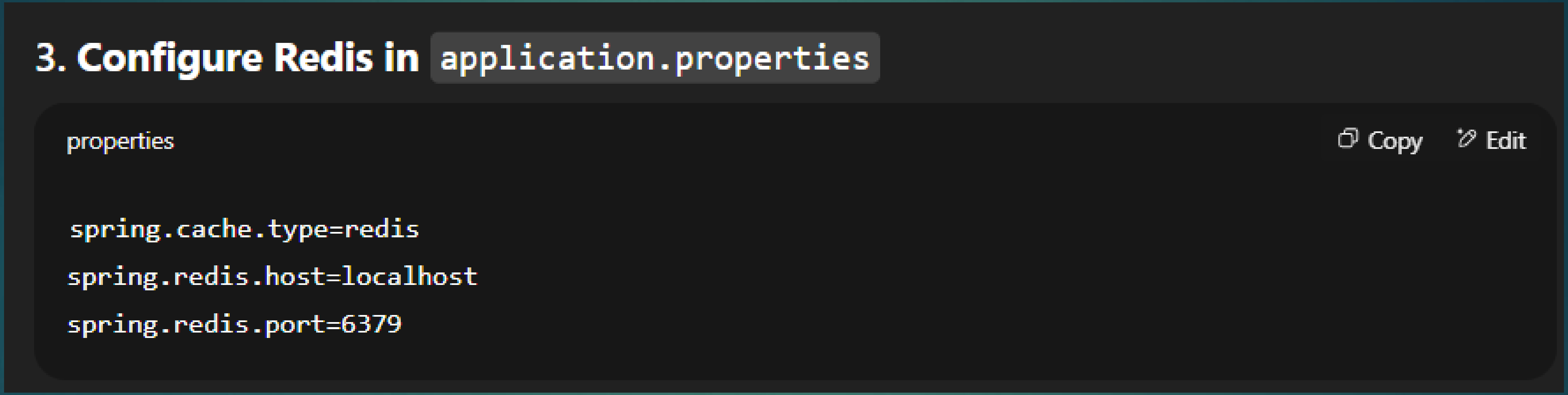


2. Enable Caching

In your Spring Boot main class or configuration class:

```java
@SpringBootApplication
@EnableCaching
public class MyApplication { }
```

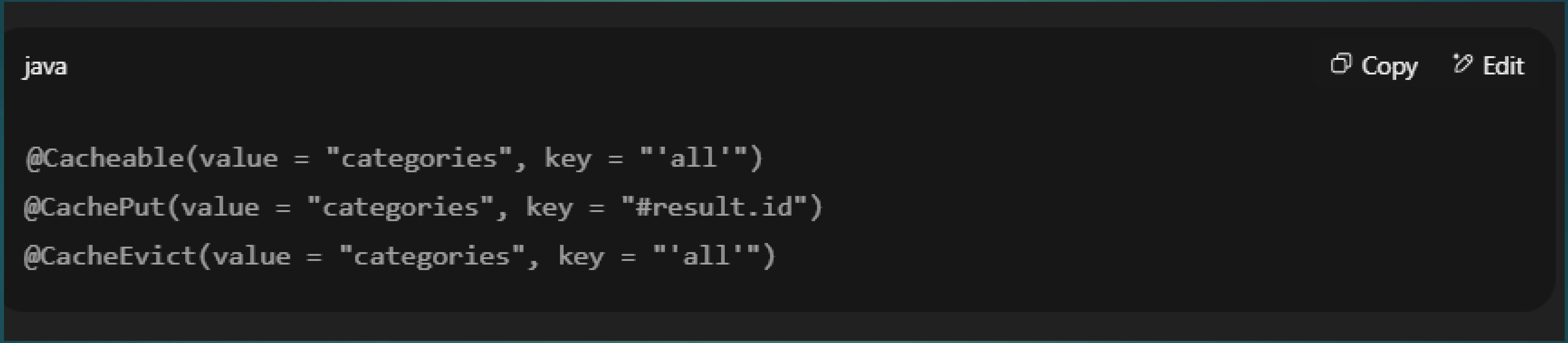
By Eslam Khder



3. Configure Redis in `application.properties`

```properties
spring.cache.type=redis
spring.redis.host=localhost
spring.redis.port=6379
```

By Eslam Khder



java

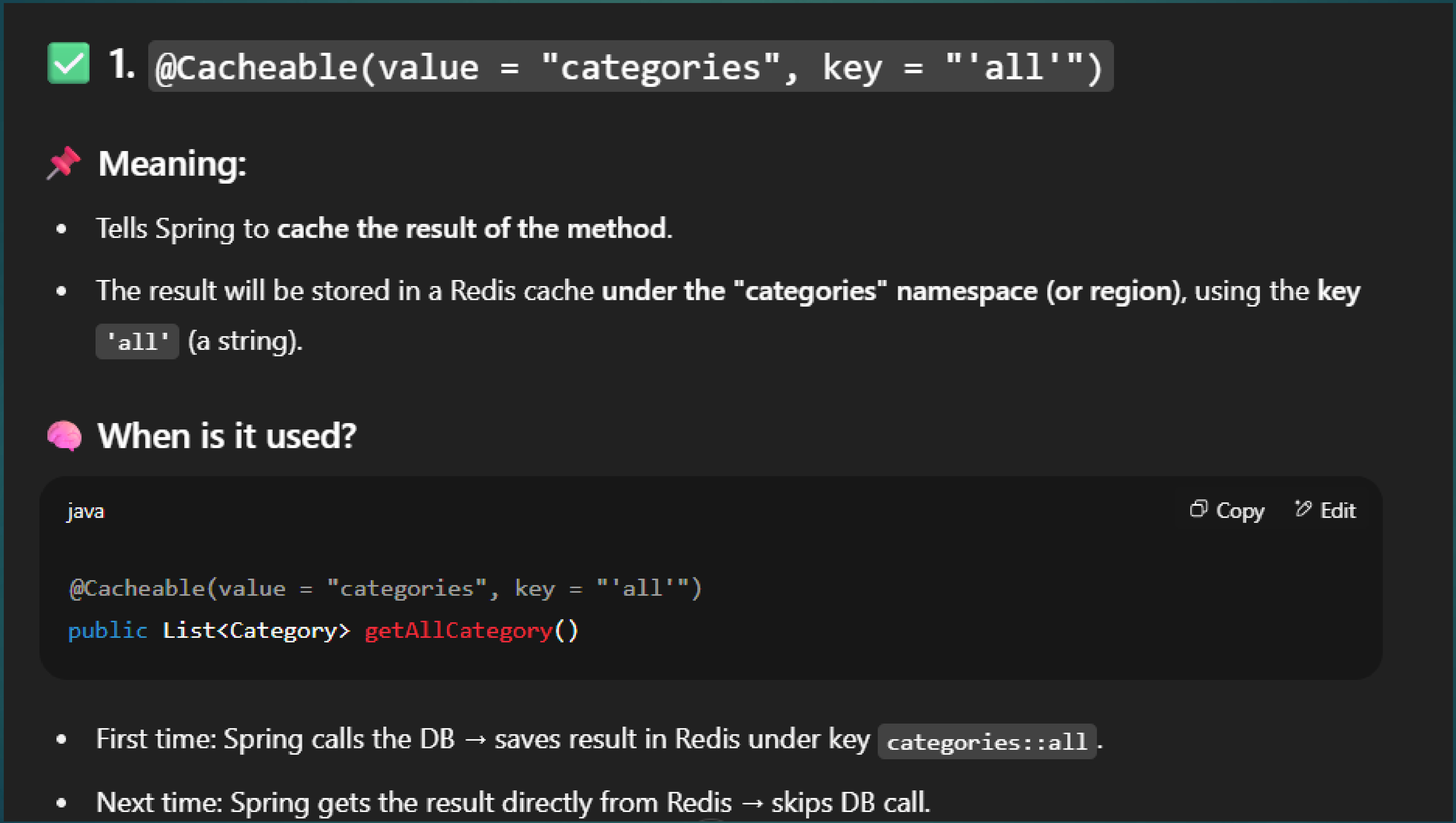
Copy Edit

```java
@Cacheable(value = "categories", key = "'all'")
@CachePut(value = "categories", key = "#result.id")
@CacheEvict(value = "categories", key = "'all'")
```

By Eslam Khder

spring

EraaSoft



✅ 1. `@Cacheable(value = "categories", key = "'all'")`

📌 Meaning:

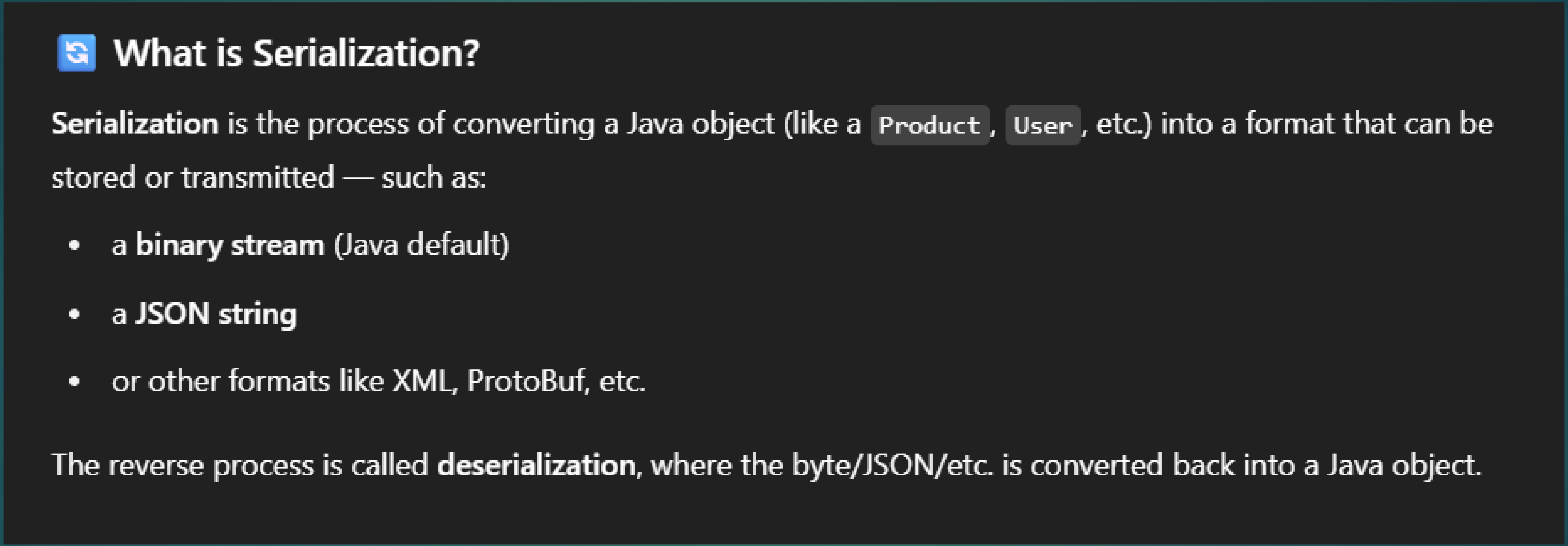
- Tells Spring to **cache the result of the method.**
- The result will be stored in a Redis cache **under the "categories" namespace (or region)**, using the **key** `'all'` (a string).

🧠 When is it used?

```java
@Cacheable(value = "categories", key = "'all'")
public List<Category> getAllCategory()
```

- First time: Spring calls the DB → saves result in Redis under key `categories::all`.
- Next time: Spring gets the result directly from Redis → skips DB call.

By Eslam Khder



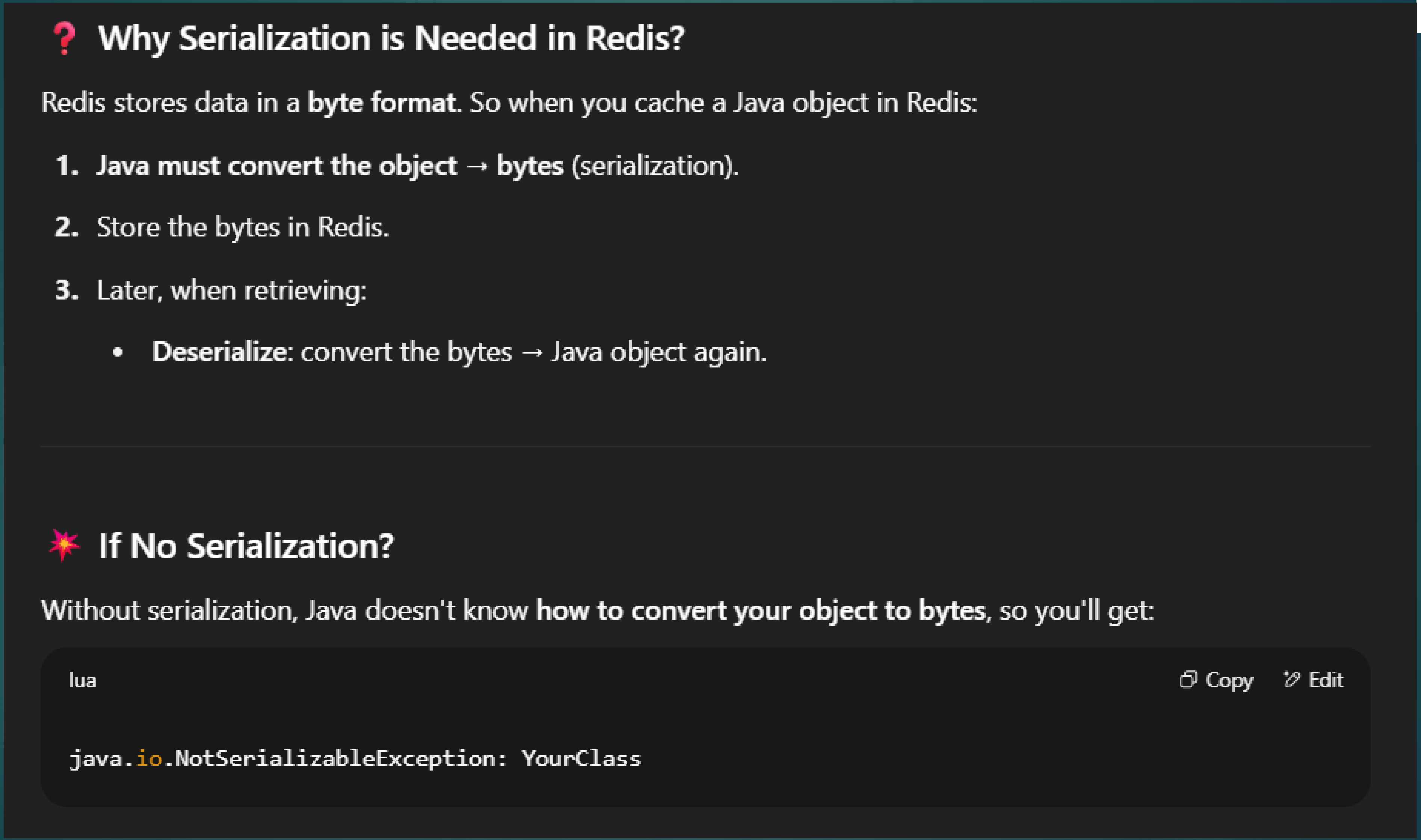
🔄 What is Serialization?

Serialization is the process of converting a Java object (like a `Product`, `User`, etc.) into a format that can be stored or transmitted — such as:

- a **binary stream** (Java default)
- a **JSON string**
- or other formats like XML, ProtoBuf, etc.

The reverse process is called **deserialization**, where the byte/JSON/etc. is converted back into a Java object.

By Eslam Khder



❓ Why Serialization is Needed in Redis?

Redis stores data in a **byte format**. So when you cache a Java object in Redis:

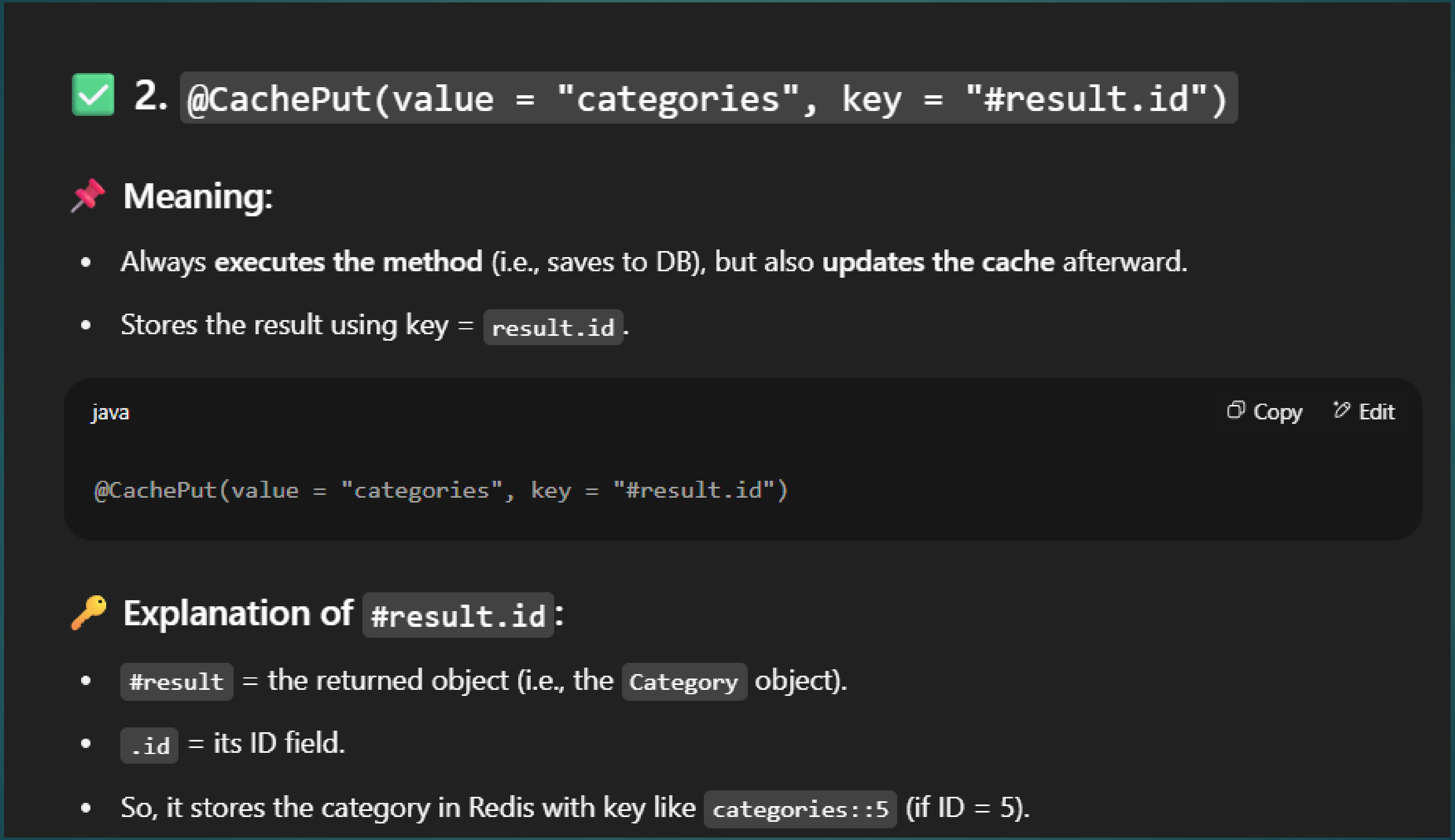
1. **Java must convert the object → bytes** (serialization).
2. Store the bytes in Redis.
3. Later, when retrieving:
   - **Deserialize**: convert the bytes → Java object again.

💥 If No Serialization?

Without serialization, Java doesn't know **how to convert your object to bytes**, so you'll get:

```lua
java.io.NotSerializableException: YourClass
```

By Eslam Khder



✅ 2. `@CachePut(value = "categories", key = "#result.id")`

📌 Meaning:

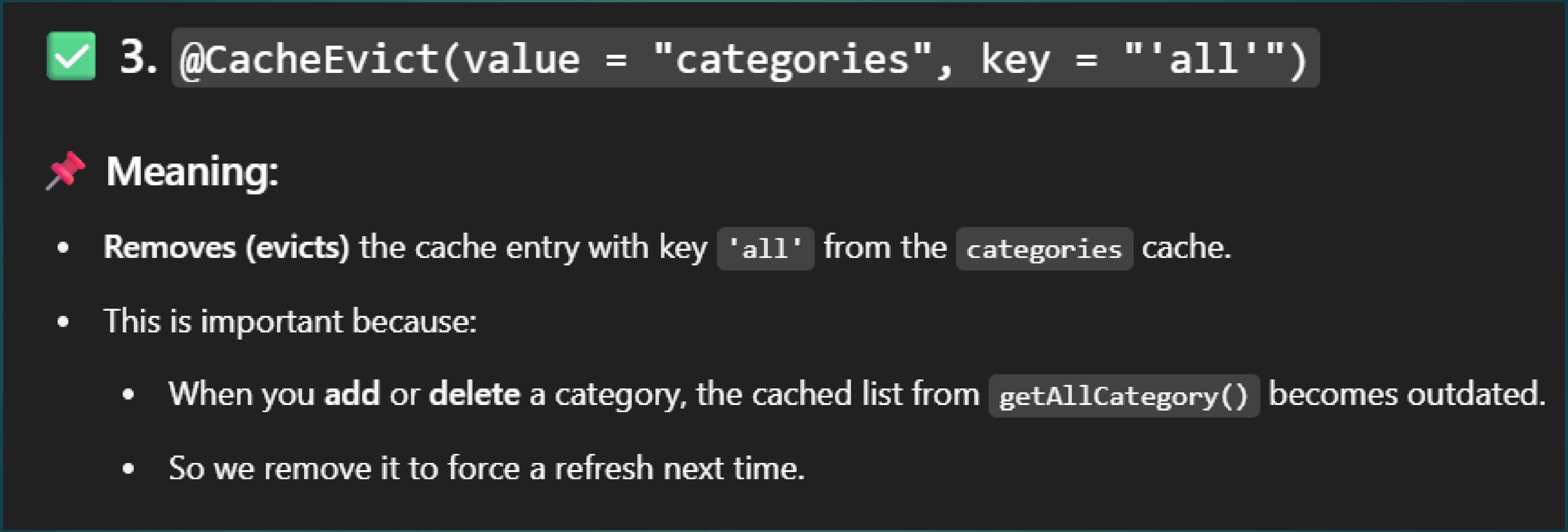
- Always **executes the method** (i.e., saves to DB), but also **updates the cache** afterward.
- Stores the result using key = `result.id`.

```java
@CachePut(value = "categories", key = "#result.id")
```

🔑 Explanation of `#result.id`:

- `#result` = the returned object (i.e., the `Category` object).
- `.id` = its ID field.
- So, it stores the category in Redis with key like `categories::5` (if ID = 5).

By Eslam Khder

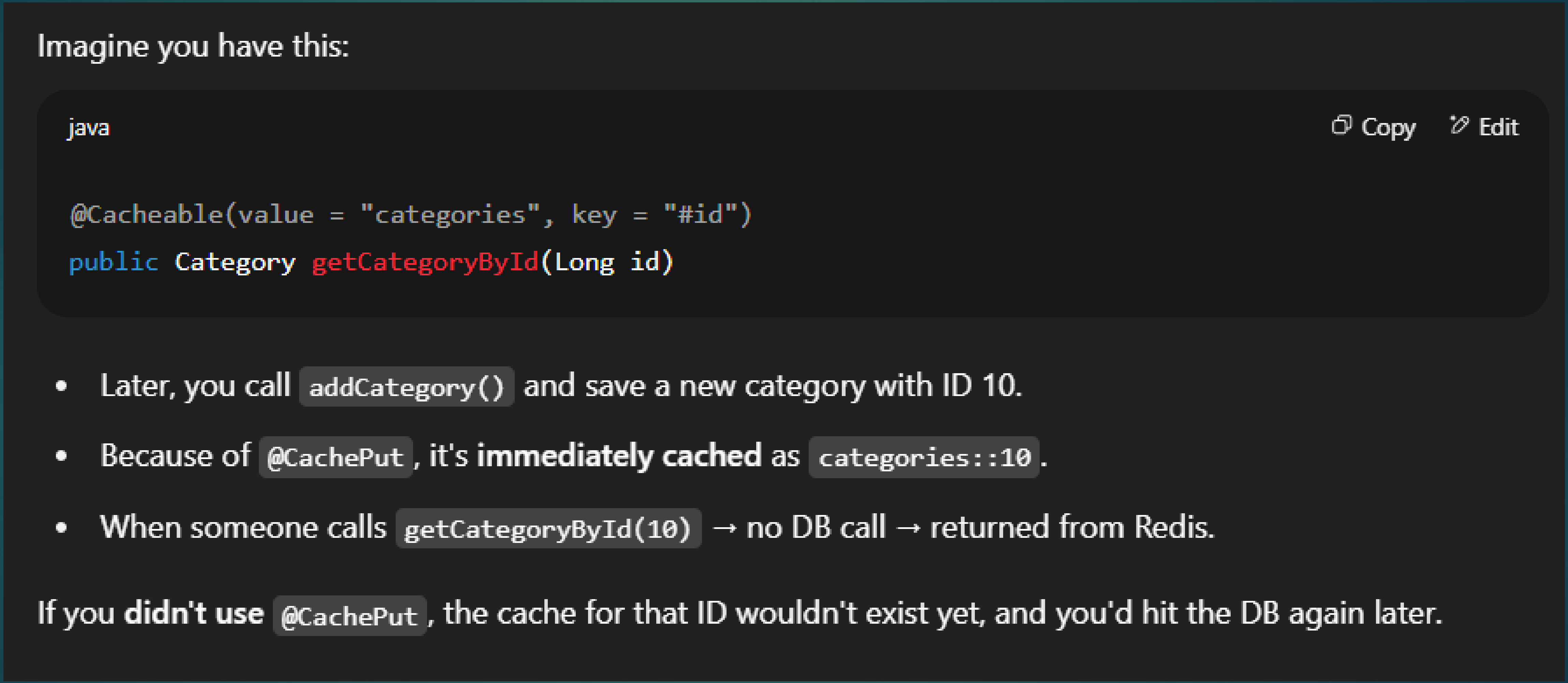


✅ 3. `@CacheEvict(value = "categories", key = "'all'")`

📌 Meaning:

- **Removes (evicts)** the cache entry with key `'all'` from the `categories` cache.
- This is important because:
  - When you **add** or **delete** a category, the cached list from `getAllCategory()` becomes outdated.
  - So we remove it to force a refresh next time.

By Eslam Khder



Imagine you have this:

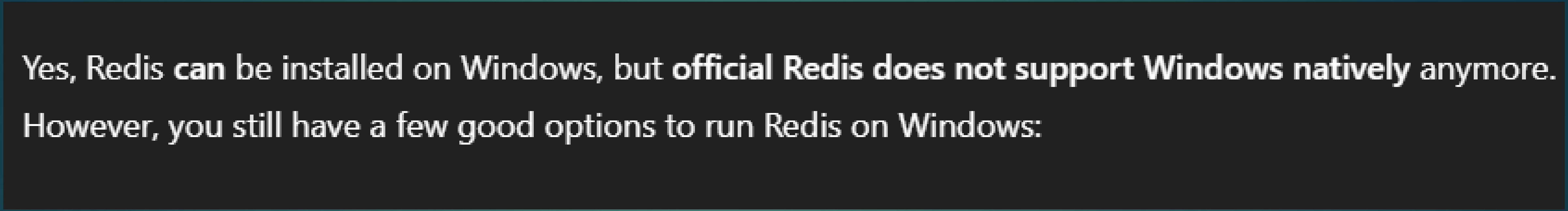
java Copy Edit

```java
@Cacheable(value = "categories", key = "#id")
public Category getCategoryById(Long id)
```

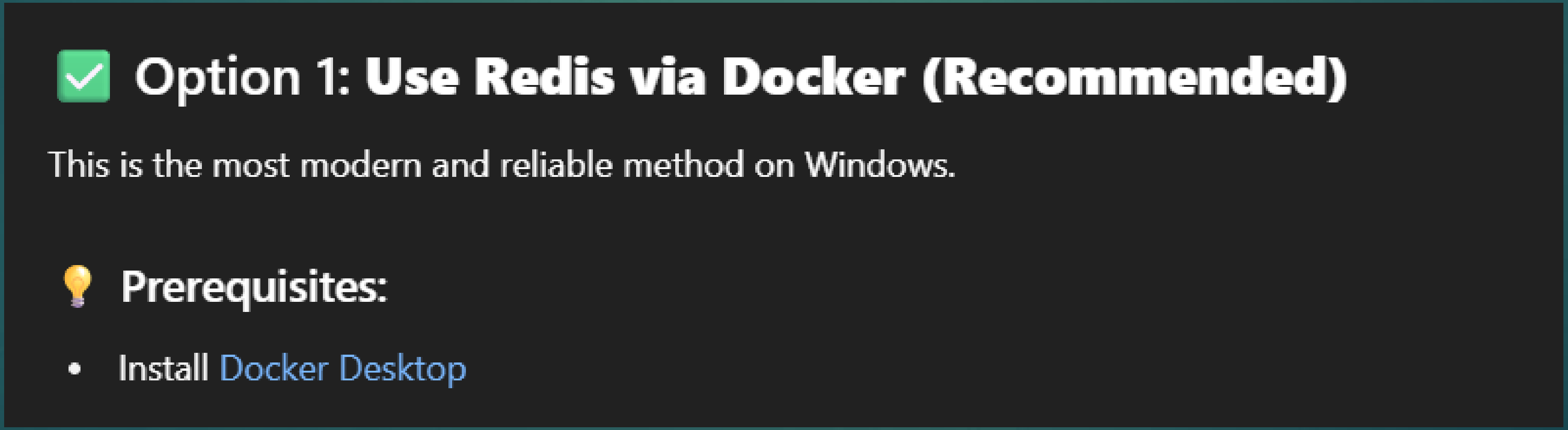
- Later, you call `addCategory()` and save a new category with ID 10.
- Because of `@CachePut`, it's **immediately cached** as `categories::10`.
- When someone calls `getCategoryById(10)` → no DB call → returned from Redis.

If you **didn't use** `@CachePut`, the cache for that ID wouldn't exist yet, and you'd hit the DB again later.

By Eslam Khder



Yes, Redis **can** be installed on Windows, but **official Redis does not support Windows natively** anymore. However, you still have a few good options to run Redis on Windows:



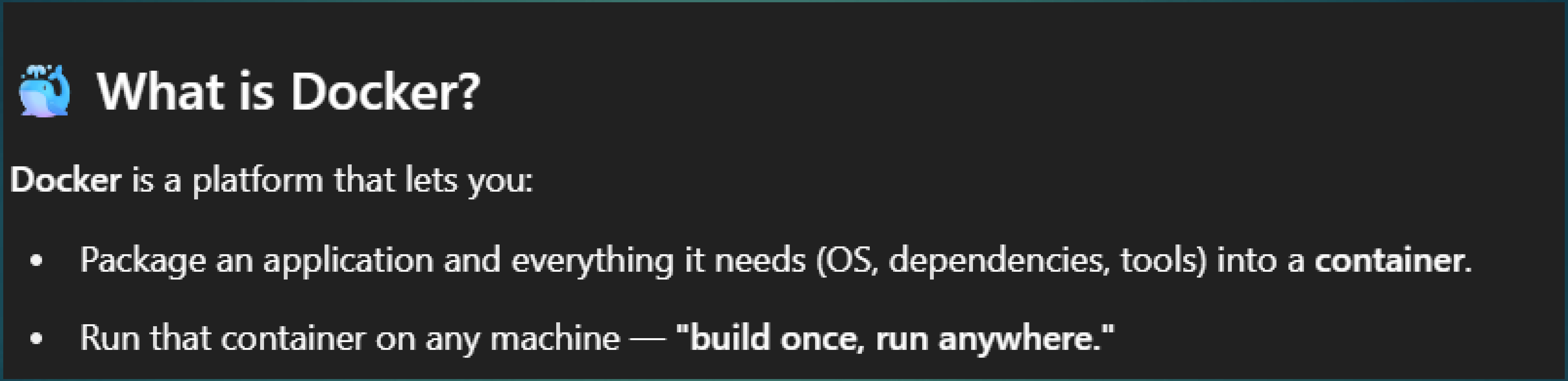
✅ Option 1: **Use Redis via Docker (Recommended)**

This is the most modern and reliable method on Windows.

💡 **Prerequisites:**

- Install Docker Desktop

By Eslam Khder

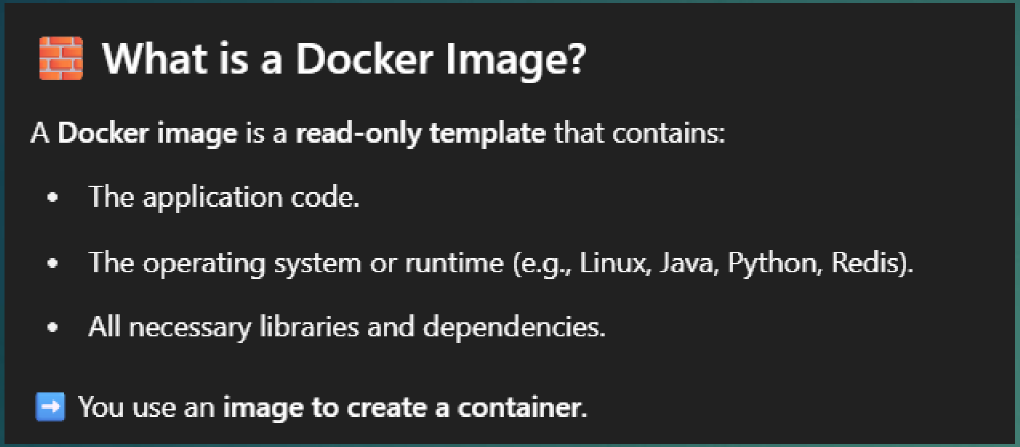


🐳 What is Docker?

Docker is a platform that lets you:

- Package an application and everything it needs (OS, dependencies, tools) into a **container**.
- Run that container on any machine — **"build once, run anywhere."**

By Eslam Khder

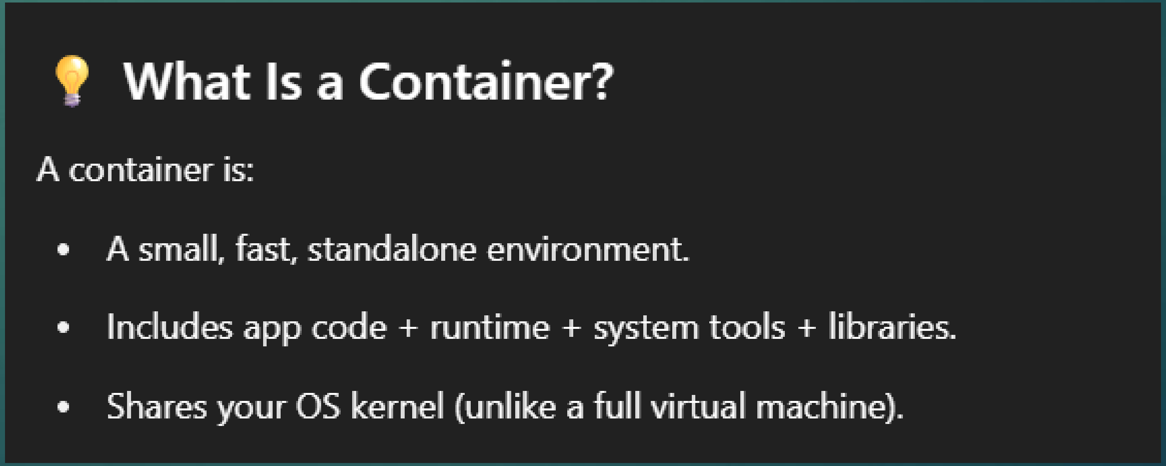


🧱 What is a Docker Image?

A **Docker image** is a **read-only template** that contains:

- The application code.
- The operating system or runtime (e.g., Linux, Java, Python, Redis).
- All necessary libraries and dependencies.

➡️ You use an **image to create a container.**



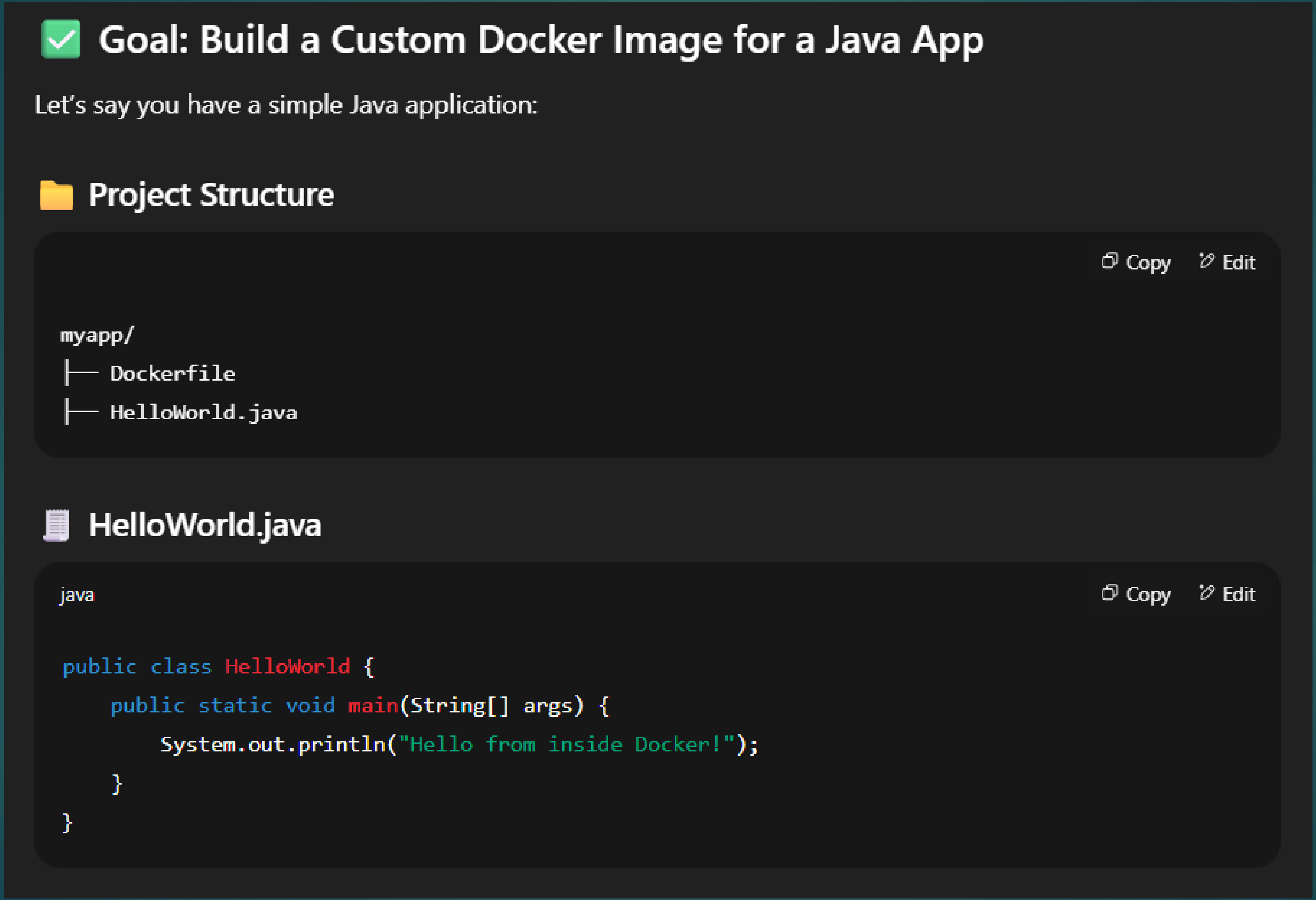
💡 What Is a Container?

A container is:

- A small, fast, standalone environment.
- Includes app code + runtime + system tools + libraries.
- Shares your OS kernel (unlike a full virtual machine).

spring

EraaSoft



✅ Goal: Build a Custom Docker Image for a Java App

Let's say you have a simple Java application:

📁 Project Structure

Copy Edit

```
myapp/
├── Dockerfile
├── HelloWorld.java
```

📃 HelloWorld.java

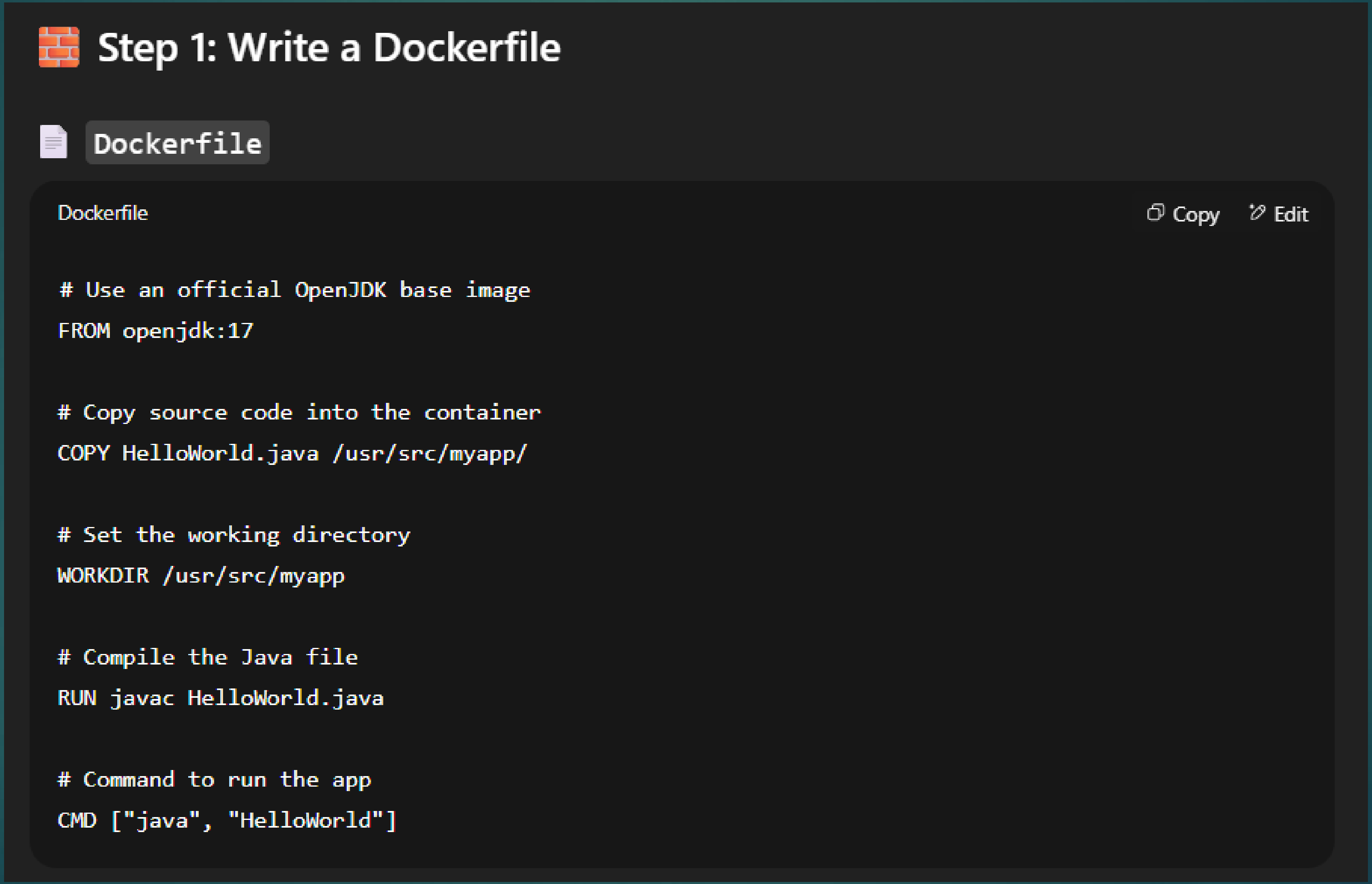
java Copy Edit

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello from inside Docker!");
    }
}
```

By Eslam Khder

spring®

EraaSoft



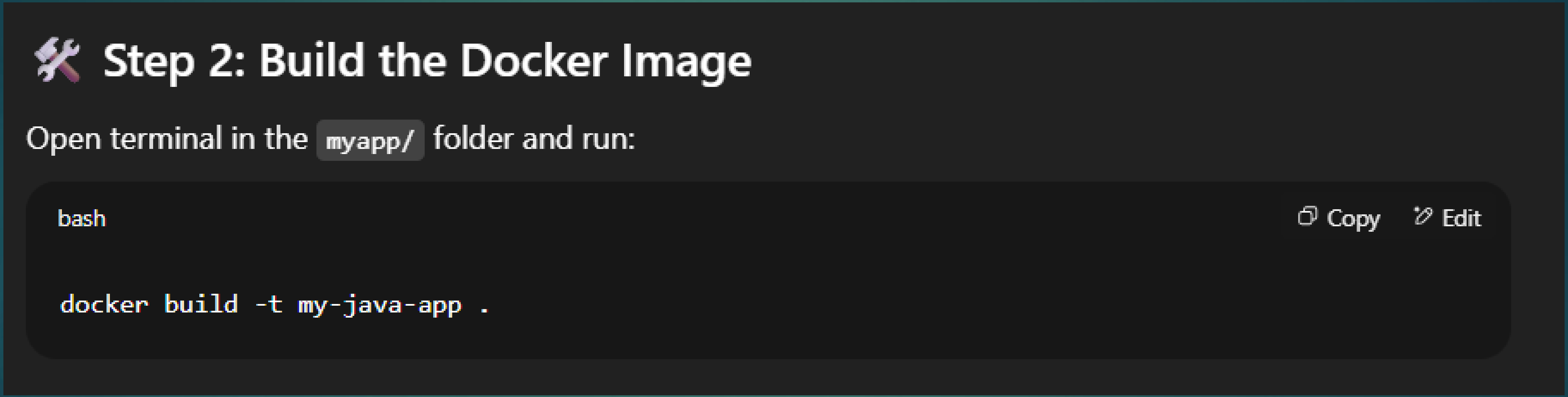
🧱 Step 1: Write a Dockerfile

📄 `Dockerfile`

Dockerfile

```
# Use an official OpenJDK base image
FROM openjdk:17

# Copy source code into the container
COPY HelloWorld.java /usr/src/myapp/

# Set the working directory
WORKDIR /usr/src/myapp

# Compile the Java file
RUN javac HelloWorld.java

# Command to run the app
CMD ["java", "HelloWorld"]
```

By Eslam Khder

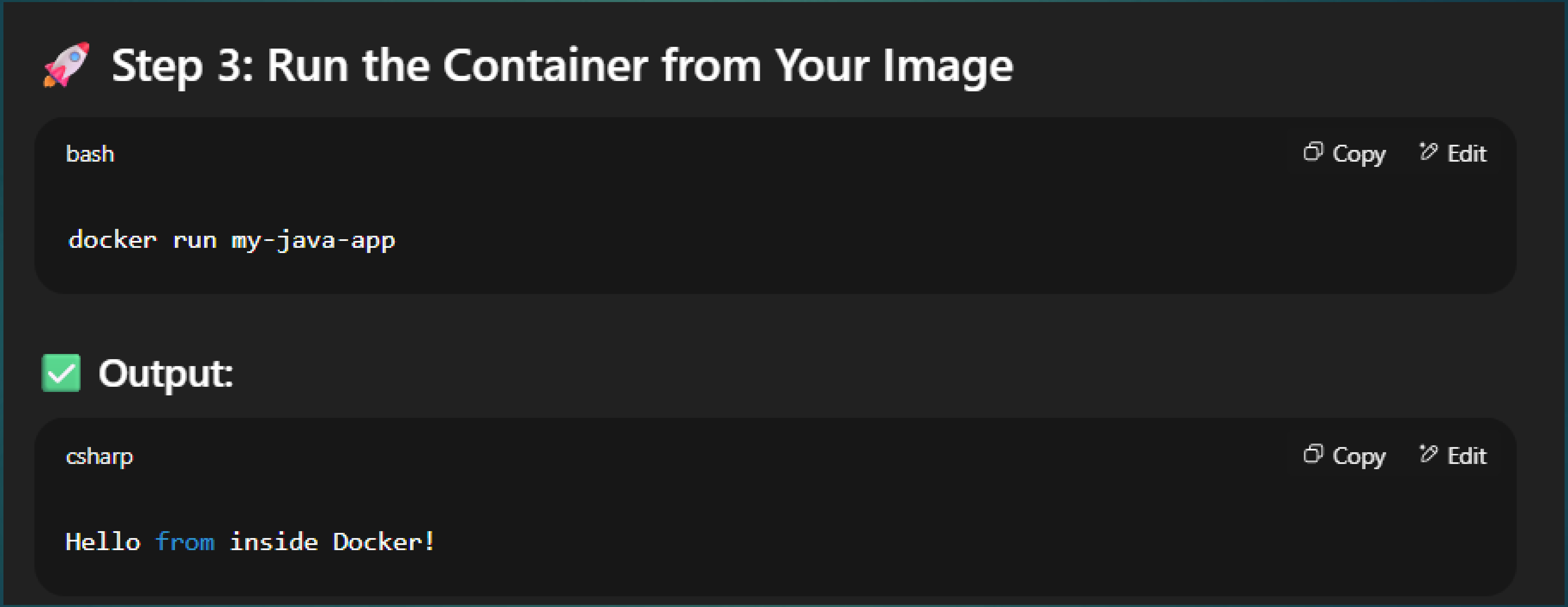


🛠️ Step 2: Build the Docker Image

Open terminal in the `myapp/` folder and run:

```bash
docker build -t my-java-app .
```

By Eslam Khder



🚀 Step 3: Run the Container from Your Image

bash

```bash
docker run my-java-app
```

✅ Output:

csharp

```csharp
Hello from inside Docker!
```

By Eslam Khder