Here’s a structured learning path for Spring Boot projects, starting from beginner to advanced levels. These projects will help you build your skills progressively.

**1. Beginner Level Projects:**

These projects focus on basic Spring Boot concepts like REST APIs, dependency injection, and connecting to a database.

**1.1. Simple REST API for Managing To-Do Tasks**

* **Description**: Build a REST API to manage a list of tasks (To-Do application). Each task should have a title, description, status (pending/done), and priority.
* **Skills**:
  + Basic Spring Boot setup.
  + Creating RESTful services using **@RestController**.
  + CRUD operations (Create, Read, Update, Delete) using **Spring Data JPA**.
  + Connect to an in-memory database like **H2** or an external database like **MySQL**.

**Features**:

* + GET: Retrieve a list of tasks.
  + POST: Create a new task.
  + PUT: Update a task.
  + DELETE: Remove a task.

**1.2. Student Management System**

* **Description**: Create a simple system to manage student records. The system will allow adding, updating, and retrieving student data such as name, age, and class.
* **Skills**:
  + Use **Thymeleaf** for server-side rendering.
  + CRUD operations with **Spring MVC**.
  + Form validation using **Spring Validation**.

**Features**:

* + Add new students.
  + List all students.
  + Edit and delete student records.

**2. Intermediate Level Projects:**

These projects introduce concepts like security, pagination, file uploads, and external APIs.

**2.1. User Authentication System**

* **Description**: Build a system that allows users to sign up, log in, and access protected resources.
* **Skills**:
  + Implement **Spring Security** for authentication and authorization.
  + Use **JWT (JSON Web Token)** for stateless authentication.
  + Password hashing with **BCrypt**.
  + Role-based access control.

**Features**:

* + User registration.
  + Secure login with JWT.
  + Role-based access (admin/user).

**2.2. Blog Application with Pagination and Search**

* **Description**: Create a blog system where users can write, update, and delete posts. Add pagination, sorting, and search functionality to the list of blog posts.
* **Skills**:
  + Pagination and sorting using **Spring Data JPA**.
  + Searching by title or content.
  + File uploads (for blog post images) using **MultipartFile**.
  + API documentation with **Swagger**.

**Features**:

* + Write, edit, and delete blog posts.
  + Paginated list of posts with sorting.
  + Search functionality.

**2.3. Weather Application (Using External API)**

* **Description**: Create a Spring Boot application that fetches weather data from an external API (e.g., OpenWeatherMap).
* **Skills**:
  + **RestTemplate** or **WebClient** for making HTTP requests to external APIs.
  + Parsing JSON responses.
  + Displaying the data using **Thymeleaf** or exposing it via REST.

**Features**:

* + Fetch current weather data by city.
  + Display weather details such as temperature, humidity, and conditions.

**3. Advanced Level Projects:**

These projects dive into microservices, asynchronous processing, and deploying Spring Boot applications to the cloud.

**3.1. E-Commerce System with Microservices Architecture**

* **Description**: Build an e-commerce platform using a microservices architecture. The system should have services for product management, user management, and order processing.
* **Skills**:
  + **Spring Cloud** for building microservices.
  + **Eureka** for service discovery.
  + **Spring Cloud Gateway** for API Gateway.
  + **Feign** for communication between services.
  + **Docker** for containerization.
  + **MySQL/PostgreSQL** for database management.
  + **RabbitMQ/Kafka** for message-driven communication (event-driven architecture).

**Features**:

* + Product catalog (CRUD for products).
  + User service (registration, login, etc.).
  + Order service (placing and managing orders).
  + Service discovery and load balancing.

**3.2. Real-Time Chat Application with WebSockets**

* **Description**: Build a chat application where users can send messages to each other in real time.
* **Skills**:
  + **WebSockets** in Spring Boot for real-time communication.
  + **STOMP** protocol for sending messages.
  + **Spring Security** for securing chat rooms.
  + **Redis** for session storage or message caching.

**Features**:

* + Real-time messaging between users.
  + Group chat rooms.
  + Authentication for users.

**3.3. Online Payment Processing System (Stripe Integration)**

* **Description**: Create a payment processing system using **Stripe** API. Users can make payments, and the system should securely handle credit card information.
* **Skills**:
  + Integrating with external payment gateways like Stripe or PayPal.
  + Handling secure payments.
  + Asynchronous processing using **Spring Events** or **@Async** for payment notifications.

**Features**:

* + Payment gateway integration.
  + Secure credit card handling.
  + Payment confirmation and notifications.

**4. Expert Level Projects:**

Projects focusing on cloud, serverless architecture, and enterprise-level systems.

**4.1. Deploying Spring Boot Applications on AWS with CI/CD**

* **Description**: Create a Spring Boot application and deploy it to **AWS** using **Elastic Beanstalk** or **EC2**. Set up CI/CD pipelines using **Jenkins**, **GitHub Actions**, or **GitLab CI**.
* **Skills**:
  + **AWS EC2/Elastic Beanstalk** for cloud deployment.
  + **Docker** for containerizing the application.
  + **CI/CD pipeline** configuration for automatic deployments.

**Features**:

* + Deploy a Spring Boot application to AWS.
  + Set up CI/CD pipelines for automatic testing and deployment.

**4.2. Serverless Spring Boot Application on AWS Lambda**

* **Description**: Convert a Spring Boot application into a serverless function using **AWS Lambda** and **API Gateway**.
* **Skills**:
  + Writing serverless functions with Spring Boot.
  + Integrating with **AWS Lambda** and **API Gateway**.
  + **AWS S3** for file storage.
  + **AWS DynamoDB** for database storage.

**Features**:

* + Deploy a Spring Boot function as a serverless service.
  + Manage serverless functions via API Gateway.

**Resources to Learn Along the Way:**

* **Official Spring Documentation**: [spring.io/guides](https://spring.io/guides)
* **Spring Boot Projects**: GitHub repositories of open-source Spring Boot projects.
* **YouTube Tutorials**: Follow along with practical videos.
* **Udemy or Coursera**: Structured courses on Spring Boot and microservices.

Each project builds on skills from the previous one, helping you progress from understanding the basics of Spring Boot to mastering advanced features like microservices, real-time applications, and cloud deployment. Let me know if you need further guidance on any of these projects!