Spring Boot Actuator is a production-ready module that provides features for monitoring, managing, and interacting with your Spring Boot application, offering built-in endpoints accessible via HTTP or JMX for tasks like health checks, metrics, and auditing.

Here's a more detailed explanation:

What it is:

* **Production-Ready Features:**

Actuator provides features that are essential for managing and monitoring applications in a production environment.

* **Monitoring and Management:**

It allows you to monitor the health, performance, and behavior of your application.

* **Endpoints:**

Actuator exposes various endpoints (e.g., /actuator/health, /actuator/metrics) that you can interact with to gather information about your application.

* **Access Methods:**

You can access these endpoints via HTTP (using tools like curl or a web browser) or JMX (Java Management Extensions).

* **Dependency:**

To use Actuator, you need to add the spring-boot-starter-actuator dependency to your project.

* **Auditing:**

Actuator provides features for auditing application events.

Key Features and Benefits:

* **Health Checks:** Monitor the health of your application and its dependencies.
* **Metrics:** Collect and expose application metrics for performance analysis.
* **Info:** Access application information, such as build details.
* **Environment:** View the application's environment variables.
* **Loggers:** Manage log levels and configurations.
* **Custom Endpoints:** You can create your own custom endpoints to expose specific application data.
* **Security:** Actuator endpoints can be secured using authentication and authorization.
* **Auditing:** A flexible audit framework that publishes events to an AuditEventRepository.

How to use it:

1. 1. **Add the Dependency:**

Include the spring-boot-starter-actuator dependency in your project's pom.xml (Maven) or build.gradle (Gradle) file.

1. 2. **Access Endpoints:**

Use HTTP or JMX to access the exposed endpoints.

1. 3. **Configure:**

You can configure Actuator's behavior, including enabling or disabling specific endpoints, setting security configurations, and configuring metrics exporters.

1. 4. **Example:**
   * To check the application's health, you can access http://localhost:8080/actuator/health.
   * To view metrics, you can access http://localhost:8080/actuator/metrics.

Security Considerations:

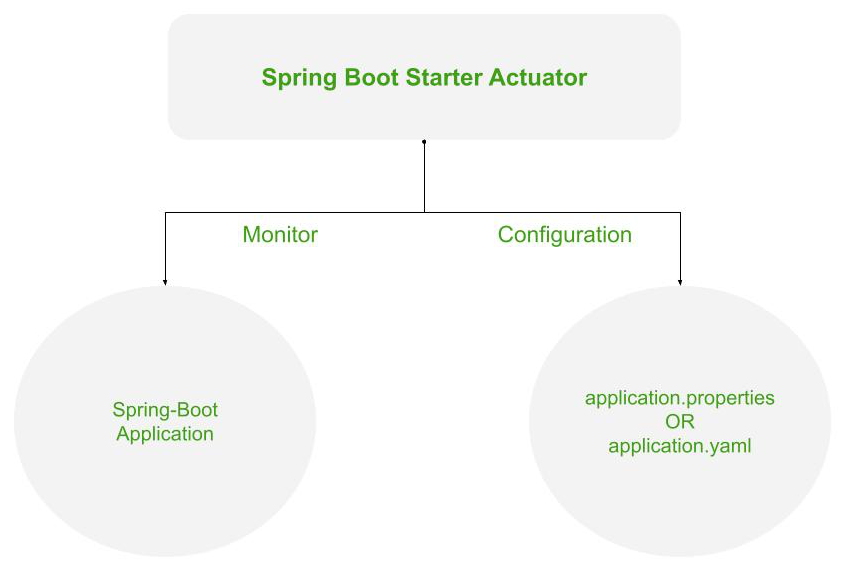
* **Expose endpoints with caution:** Actuator endpoints can expose sensitive information, so it's crucial to secure them properly, especially in production environments.
* **Authentication and Authorization:** Configure authentication and authorization to restrict access to Actuator endpoints.
* **Disable unnecessary endpoints:** Disable endpoints that you don't need to reduce the attack surface.
* **Regularly update Spring Boot:** Keep your Spring Boot version up to date to benefit from security patches.

**Spring Boot Actuator**

Developing and Managing an application are the two most important aspects of the application’s life cycle. It is very crucial to know what’s going on beneath the application. Also when we push the application into production, managing it gradually becomes critically important. Therefore, it is always recommended to monitor the application both while at the development phase and at the production phase.

For the same use case, Spring Boot provides an actuator dependency that can be used to monitor and manage your Spring Boot application, By /actuator and /actuator/health endpoints you can achieve the purpose of monitoring.

* With the help of Spring Boot, we can achieve the above objectives.
* Spring Boot’s ‘Actuator’ dependency is used to monitor and manage the Spring web application.
* We can use it to monitor and manage the application with the help of HTTP endpoints or with the JMX.



*Working of the Spring’s Actuator*

**Advantages of Actuator the Application**

1. It increases customer satisfaction.
2. It reduces downtime.
3. It boosts productivity.
4. It improves Cybersecurity Management.
5. It increases the conversion rate.

**1. Configuration for Actuator**

In order to use hibernate validators, these configurations are necessary in your Spring Boot project.

**1.1 Dependency for Actuator**

To use the ‘Actuator’ add the following dependency in your application’s project settings file.

Dependency configuration for both Maven and Gradle build system.

***Maven* -> pom.xml**  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
</dependencies>

***Gradle***

**-> build.gradle**  
  
dependencies {  
 implementation 'org.springframework.boot:spring-boot-starter-actuator'  
}

**1.2 Application Properties configuration for Actuator**

There more configurations available for Actuator, few of them are listed:

* You can also change the default endpoint by adding the following in the application.properties file.

management.endpoints.web.base-path=/details

* Including IDs/Endpoints

By default, all IDs are set to false except for ‘health’. To include an ID, use the following property in the application.properties file.

management.endpoint.<id>.enabled  
  
**Example ->** management.endpoint.metrics.enabled=true

* List down all IDs that you want to include which are separated by a comma.

management.endpoints.web.exposure.include=metrics,info

* Include only metrics and info IDs and will exclude all others (‘health’ too).

To add/include all ID information about your application, you can do it in the application.properties file by simply adding the following –

management.endpoints.web.exposure.include=\*

* Excluding IDs/Endpoints

To exclude an ID or endpoint, use the following property and list out the respective IDs separated by a comma in the application.properties file.

management.endpoints.web.exposure.exclude  
  
**Example ->** management.endpoints.web.exposure.exclude=info