

Spring Boot Actuator

What is Spring Boot Actuator?

Spring Boot Actuator is a built-in Spring Boot module that provides **production-ready features** to help us to **monitor and manage your application**.

It exposes **REST endpoints** that give insights into the internal state of our application — such as health, metrics, environment properties, etc.

Key Use Cases / Benefits

- Monitor **application health**
- Gather **metrics** (CPU, memory, heap, GC, requests, etc.)
- View **application environment** properties (profiles, beans, configs)
- Expose **custom monitoring endpoints**
- Integrate with monitoring tools like **Prometheus, Grafana, ELK stack, Datadog**

Common Actuator Endpoints (/actuator)

| Endpoint | Description |
|--------------------|--------------------------------------------------------|
| /actuator/health | Application health status (disk space, DB, etc.) |
| /actuator/info | Custom info about the application |
| /actuator/metrics | Application metrics (CPU, memory, HTTP requests, etc.) |
| /actuator/env | Environment variables and properties |
| /actuator/beans | Lists all Spring beans |
| /actuator/mappings | Lists all HTTP request mappings |
| /actuator/loggers | Manage logging levels at runtime |

[/actuator/threads](#) Current state of application threads

Step-by-Step Guide to Integrate Spring Boot Actuator

Step 1: Add Actuator Dependency

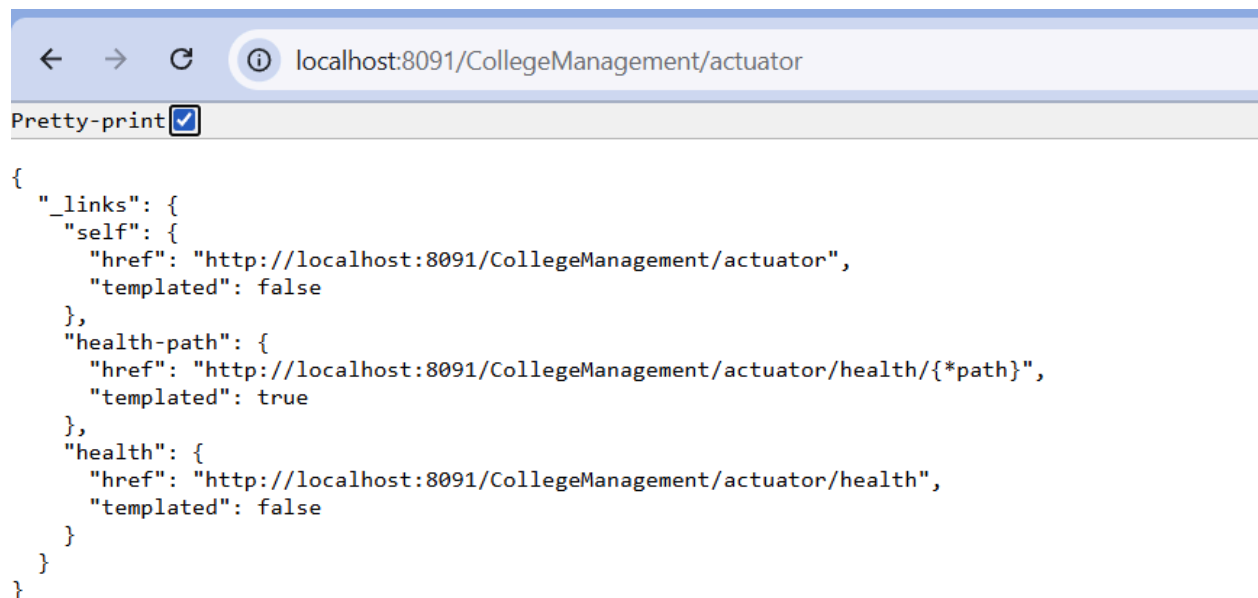
add the following to your `pom.xml`:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Step 2: Enable Actuator Endpoints in `application.properties` or `application.yml`

By default, only a few endpoints like [/actuator/health](#) and [/actuator/info](#) are enabled.

```
Configuration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be p
OptionalLiveReloadServer           : LiveReload server is running on port 35729
.web.EndpointLinksResolver         : Exposing 1 endpoint(s) beneath base path '/actuator'
EmbeddedTomcatTomcatWebServer      : Tomcat started on port(s): 8091 (http) with context path '/CollegeManagement'
CollegeManagementApplication       : Started CollegeManagementApplication in 8.228 seconds (process running for 8.818)
```



To enable all endpoints:

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

```

22
23 # actuator
24 # To enable all endpoints:
25 management.endpoints.web.exposure.include=*
26

```

```

. web.EndpointLinksResolver      : Exposing 14 endpoint(s) beneath base path '/actuator'
mbedded.tomcat.TomcatWebServer   : Tomcat started on port(s): 8091 (http) with context path '/CollegeManagement'
legeManagementApplication        : Started CollegeManagementApplication in 1.142 seconds (process running for 384.448)
nEvaluationDeltaLoggingListener  : Condition evaluation unchanged

```

← → ↺ ① localhost:8091/CollegeManagement/actuator

pretty-print ☒

```

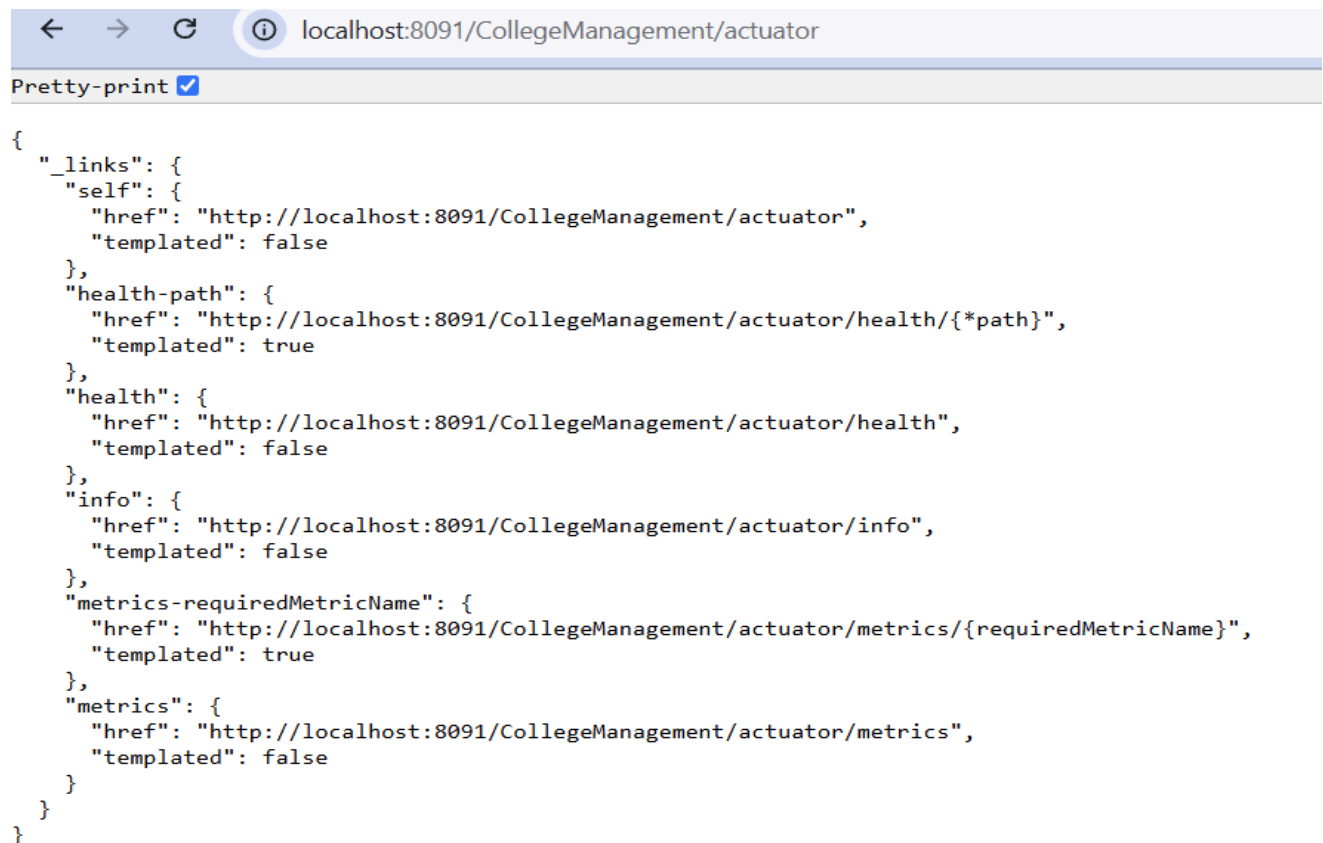
{
  "_links": {
    "self": {
      "href": "http://localhost:8091/CollegeManagement/actuator",
      "templated": false
    },
    "beans": {
      "href": "http://localhost:8091/CollegeManagement/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href": "http://localhost:8091/CollegeManagement/actuator/caches/{cache}",
      "templated": true
    },
    "caches": {
      "href": "http://localhost:8091/CollegeManagement/actuator/caches",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8091/CollegeManagement/actuator/health/{*path}",
      "templated": true
    },
    "health": {
      "href": "http://localhost:8091/CollegeManagement/actuator/health",
      "templated": false
    },
    "info": {
      "href": "http://localhost:8091/CollegeManagement/actuator/info",
      "templated": false
    },
    "conditions": {
      "href": "http://localhost:8091/CollegeManagement/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href": "http://localhost:8091/CollegeManagement/actuator/configprops",
      "templated": false
    },
    "configprops-prefix": {
      "href": "http://localhost:8091/CollegeManagement/actuator/configprops/{prefix}",
      "templated": true
    },
    "env": {
      "href": "http://localhost:8091/CollegeManagement/actuator/env",
      "templated": false
    }
  }
}

```

For production, don't expose all endpoints—expose only what is needed (like `health`, `metrics`, etc.)

Edit your `application.properties`:

```
28 # management.server.port=8081
29
30 # Expose only selected actuator endpoints
31 management.endpoints.web.exposure.include=health,info,metrics
32 |
33
```



```
{
  "_links": {
    "self": {
      "href": "http://localhost:8091/CollegeManagement/actuator",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8091/CollegeManagement/actuator/health/{*path}",
      "templated": true
    },
    "health": {
      "href": "http://localhost:8091/CollegeManagement/actuator/health",
      "templated": false
    },
    "info": {
      "href": "http://localhost:8091/CollegeManagement/actuator/info",
      "templated": false
    },
    "metrics-requiredMetricName": {
      "href": "http://localhost:8091/CollegeManagement/actuator/metrics/{requiredMetricName}",
      "templated": true
    },
    "metrics": {
      "href": "http://localhost:8091/CollegeManagement/actuator/metrics",
      "templated": false
    }
  }
}
```

Step 3: Optional — Customize Server Port for Actuator

If you want actuator endpoints to be served on a different port:

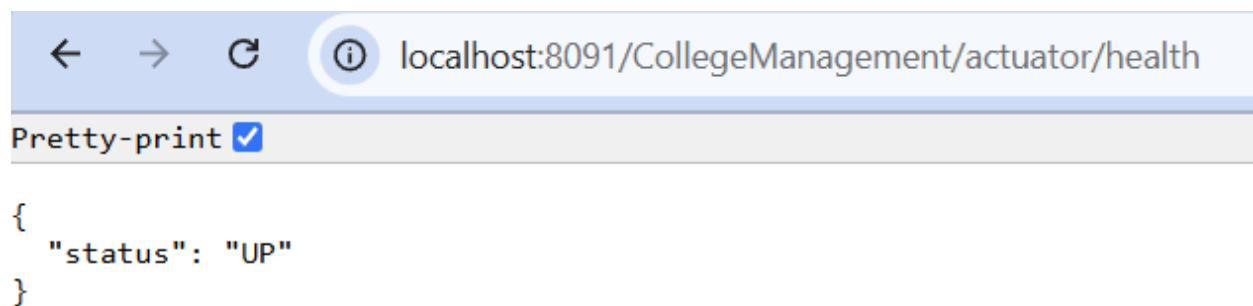
`management.server.port=8081`



```
{
  "_links": {
    "self": {
      "href": "http://localhost:8081/actuator",
      "templated": false
    },
    "beans": {
      "href": "http://localhost:8081/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href": "http://localhost:8081/actuator/caches/{cache}",
      "templated": true
    },
    "caches": {
      "href": "http://localhost:8081/actuator/caches",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8081/actuator/health/{*path}",
      "templated": true
    },
    "health": {
      "href": "http://localhost:8081/actuator/health",
      "templated": false
    },
    "info": {
      "href": "http://localhost:8081/actuator/info",
      "templated": false
    },
    "conditions": {
      "href": "http://localhost:8081/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href": "http://localhost:8081/actuator/configprops",
      "templated": false
    },
    "configprops-prefix": {
      "href": "http://localhost:8081/actuator/configprops/{prefix}",
      "templated": true
    },
    "env": {
      "href": "http://localhost:8081/actuator/env",
      "templated": false
    }
  }
}
```

✓ Health Status Types (`Health.status()` values)

| Status | Meaning |
|----------------|-----------------------------------------------------------------------|
| UP | ✓ Everything is working fine |
| DOWN | ✗ At least one component is not healthy |
| OUT_OF_SERVICE | ⊘ Component is intentionally taken out of service (e.g., disabled DB) |
| UNKNOWN | ? Health check could not determine the status |



```
{
  "status": "UP"
}
```

Examples

`./actuator/beans`

What it shows:

- All Spring Beans registered in the `ApplicationContext`
- Bean class, scope, and dependencies (injected beans)

Internally uses:

- `BeansEndpoint` class
- Scans all beans in `ApplicationContext`

```
localhost:8091/CollegeManagement/actuator/beans
ty-print ✓
{
  "type": "org.springframework.boot.autoconfigure.web.servlet.MultipartProperties",
  "dependencies": []
},
  "studentController": {
    "aliases": [],
    "scope": "singleton",
    "type": "com.spring.project.controller.StudentController",
    "resource": "file [C:\\Users\\rames\\Documents\\spring boot
ects\\college_management\\target\\classes\\com\\spring\\project\\controller\\StudentController.class]",
    "dependencies": [
      "studentService"
    ]
  },
  "sslPropertiesSslBundleRegistrar": {
    "aliases": [],
    "scope": "singleton",
    "type": "org.springframework.boot.autoconfigure.ssl.SslPropertiesBundleRegistrar",
    "resource": "class path resource [org/springframework/boot/autoconfigure/ssl/SslAutoConfiguration.class]",
    "dependencies": [
      "org.springframework.boot.autoconfigure.ssl.SslAutoConfiguration",
      "spring.ssl-org.springframework.boot.autoconfigure.ssl.SslProperties"
    ]
  },
  "org.springframework.doc.webmvc.core.configuration.SpringDocWebMvcConfiguration$SpringDocWebMvcRouterConfiguration": {
    "aliases": [],
    "scope": "singleton",
    "type": "org.springframework.doc.webmvc.core.configuration.SpringDocWebMvcConfiguration$SpringDocWebMvcRouterConfiguration",
    "dependencies": []
  }
}
```

2. `/actuator/mappings`

What it shows:

- All HTTP request mappings in the app
- Which controller methods handle which URLs

Internally uses:

- `RequestMappingHandlerMapping` and `RequestMappingEndpoint`

```
localhost:8091/CollegeManagement/actuator/mappings
Pretty-print ☒
{
  "handler": "com.spring.project.controller.StudentController#findByName(String)",
  "predicate": "{GET [/api/students/search/{name}]}",
  "details": {
    "handlerMethod": {
      "className": "com.spring.project.controller.StudentController",
      "name": "findByName",
      "descriptor": "(Ljava/lang/String;)Ljava/util/List;"
    },
    "requestMappingConditions": {
      "consumes": [],
      "headers": [],
      "methods": [
        "GET"
      ],
      "params": [],
      "patterns": [
        "/api/students/search/{name}"
      ],
      "produces": []
    }
  }
},
{
  "handler": "com.spring.project.controller.StudentController#create(Student)",
  "predicate": "{POST [/api/students]}",
  "details": {
    "handlerMethod": {
      "className": "com.spring.project.controller.StudentController",
      "name": "create",
      "descriptor": "(Lcom/spring/project/entity/Student;)Lcom/spring/project/entity/Student;"
    },
    "requestMappingConditions": {
      "consumes": [],
      "headers": [],
      "methods": [
        "POST"
      ],
      "params": [],
      "patterns": [
        "/api/students"
      ]
    }
  }
}
```

3. /actuator/health



What it shows:

- Status: **UP**, **DOWN**, etc.
- Health of DB, disk, custom services (if configured)



Internally uses:

- All beans that implement **HealthIndicator**
- **CompositeHealthContributor**

Pretty-print ☒

```
{  
  "status": "UP"  
}
```

4. `/actuator/info`


 **What it shows:**

- Metadata about your app from `application.properties`

 **Output with configuration:**

```
info.app.name=My Spring App  
info.version=1.0.0  
info.author=J Ramesh
```

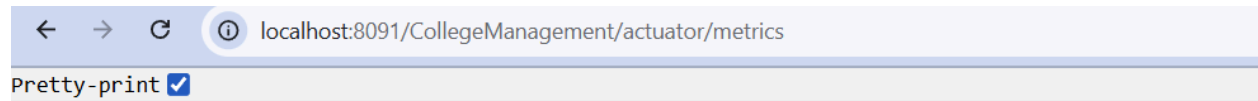
5. `/actuator/metrics`

 **What it shows:**

- JVM memory, CPU, GC, thread, HTTP metrics
- Custom metrics (if added)

 **Internally uses:**

- **Micrometer** library and registered meters



```
{
  "names": [
    "application.ready.time",
    "application.started.time",
    "disk.free",
    "disk.total",
    "executor.active",
    "executor.completed",
    "executor.pool.core",
    "executor.pool.max",
    "executor.pool.size",
    "executor.queue.remaining",
    "executor.queued",
    "hikaricp.connections",
    "hikaricp.connections.acquire",
    "hikaricp.connections.active",
    "hikaricp.connections.creation",
    "hikaricp.connections.idle",
    "hikaricp.connections.max",
    "hikaricp.connections.min",
    "hikaricp.connections.pending",
    "hikaricp.connections.timeout",
    "hikaricp.connections.usage",
    "http.server.requests",
    "http.server.requests.active",
    "jdbc.connections.active",
    "jdbc.connections.idle",
    "jdbc.connections.max",
    "jdbc.connections.min",
    "jvm.buffer.count",
    "jvm.buffer.memory.used",
    "jvm.buffer.total.capacity",
    "jvm.classes.loaded",
    "jvm.classes.unloaded",
    "jvm.compilation.time",
    "jvm.gc.live.data.size",
    "jvm.gc.max.data.size",
    "jvm.gc.memory.allocated",
    "jvm.gc.memory.promoted",
    "jvm.gc.overhead",
    "jvm.gc.pause"
  ]
}
```

6. `/actuator/env`

 **What it shows:**

- Complete environment variables loaded into the Spring `Environment`

```
localhost:8091/CollegeManagement/actuator/env
Pretty-print ☒

{
  "activeProfiles": [],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": "*****"
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "java.specification.version": {
          "value": "*****"
        },
        "sun.cpu.isalist": {
          "value": "*****"
        },
        "sun.inp.encoding": {

```

7. /actuator/threaddump

What it shows:

- All running threads in the JVM

```
localhost:8091/CollegeManagement/actuator/threaddump
Pretty-print ☒

{
  "threads": [
    {
      "threadName": "Reference Handler",
      "threadId": 2,
      "blockedTime": -1,
      "blockedCount": 12,
      "waitedTime": -1,
      "waitedCount": 0,
      "lockOwnerId": -1,
      "daemon": true,
      "inNative": false,
      "suspended": false,
      "threadState": "RUNNABLE",
      "priority": 10,
      "stackTrace": [
        {
          "moduleName": "java.base",
          "moduleVersion": "17.0.5",
          "methodName": "waitForReferencePendingList",
          "fileName": "Reference.java",
          "lineNumber": -2,
          "nativeMethod": true,
          "className": "java.lang.ref.Reference"
        },
        {
          "moduleName": "java.base",
          "moduleVersion": "17.0.5",
          "methodName": "processPendingReferences",
          "fileName": "Reference.java",
          "lineNumber": 253,
          "nativeMethod": false.

```

8. /actuator/loggers

What it shows:

- All loggers in the app
- Current logging levels
- Can dynamically change log level at runtime!



```
{
  "levels": [
    "OFF",
    "ERROR",
    "WARN",
    "INFO",
    "DEBUG",
    "TRACE"
  ],
  "loggers": {
    "ROOT": {
      "configuredLevel": "INFO",
      "effectiveLevel": "INFO"
    },
    "_org": {
      "effectiveLevel": "INFO"
    },
    "_org.springframework": {
      "effectiveLevel": "INFO"
    },
    "_org.springframework.web": {
      "effectiveLevel": "INFO"
    },
    "_org.springframework.web.servlet": {
      "effectiveLevel": "INFO"
    },
    "_org.springframework.web.servlet.HandlerMapping": {
      "effectiveLevel": "INFO"
    },
    "_org.springframework.web.servlet.HandlerMapping.Mappings": {
      "effectiveLevel": "INFO"
    }
  }
}
```