Spring Boot 3.4 Release Notes

Andy Wilkinson edited this page · 20 revisions

Upgrading from Spring Boot 3.3

RestClient and RestTemplate

Support for auto-configuring `RestClient` and `RestTemplate` to use Reactor Netty's `HttpClient` or the JDK's `HttpClient` has been added. In order of precedence, the supported clients are now:

- Apache HTTP Components (`HttpComponentsClientHttpRequestFactory`)
- Jetty Client (`JettyClientHttpRequestFactory`)
- Reactor Netty `HttpClient` (`ReactorClientHttpRequestFactory`)
- JDK `HttpClient` (`JdkClientHttpRequestFactory`)
- Simple JDK `HttpURLConnection` (`SimpleClientHttpRequestFactory`)

Notably, if you don't have an HTTP client library on the classpath, this will likely result in the use of `JdkClientHttpRequestFactory` where `SimpleClientHttpRequestFactory` would have been used previously. A particular client can be selected by setting `spring.http.client.factory`. Supported values are `http-components`, `jetty`, `reactor`, `jdk`, and `simple`.

All five clients will follow redirects by default. To disable this behavior, set `spring.http.client.redirects` to `dont-follow`.

Apache HTTP Components and Envoy

Apache HTTP Components have changed defaults in the `HttpClient` relating to HTTP/1.1 TLS upgrades. Most proxy servers handle upgrades without issue, however, you may encounter issues with Envoy or Istio.

If you need to restore previous behaviour, you can use the new `ClientHttpRequestFactoryBuilder`. Define a `HttpComponentsClientHttpRequestFactoryBuilder` and apply the following customization:

```
@Bean
public HttpComponentsClientHttpRequestFactoryBuilder
httpComponentsClientHttpRequestFactoryBuilder() {
        return ClientHttpRequestFactoryBuilder.httpComponents()
                        .withDefaultRequestConfigCustomizer((builder) ->
builder.setProtocolUpgradeEnabled(false));
```

Bean Validation of Configuration Properties

Previously, when a `@ConfigurationProperties` class annotated with `@Validated` was being validated using a Bean Validation implementation such as Hibernate Validator, validation of nested properties would be performed as they were bound, irrespective of the use of `@Valid`. In Spring Boot 3.4, validation now follows the behavior of the Bean Validation specification. Validation is performed from the `@ConfigurationProperties`-annoated class and only cascades down to nested properties where the corresponding field is annotated with `@Valid`.

When upgrading, inspect your `@ConfigurationProperties` classes with Bean Validation constraints. Add `@Valid` as necessary where you want the validation to cascade down to nested properties.

Bean-based Conditions

The behavior of `@ConditionalOnBean` and `@ConditionalOnMissingBean` has changed when used on a `@Bean` method and the `annotation` attribute is set. As before, both conditions will use the return type of the `@Bean` method as a default for the type to match. Previously, this default was not used if `name`, `type`, or `value` had been set. As of Spring Boot 3.4, this default will also not be used if `annotation` has been set. To restore the previous behavior, specify both a `value` that is the return type of the `@Bean` method and `annotation`.

Graceful Shutdown

Graceful shutdown of the embedded web server (Jetty, Rector Netty, Tomcat, or Undertow) is now enabled by default. If you need to restore the previous behavior, set `server.shutdown` to `immediate`.

Paketo tiny Builder for Building OCI Images

The default Cloud Native Buildpacks builder used when building OCI images for JVM applications using the Maven `spring-boot:build-image` goal or Gradle `bootBuildImage` task has changed from `paketobuildpacks/builder-jammy-base` to `paketobuildpacks/builder-jammy-java-tiny`. This should result in smaller images. The `tiny` builder does not include a shell, so it might not work for applications that require a start script to run the application. It also includes a reduced set of system libraries which, depending on your application, may not meet its needs. See the Maven or Gradle documentation for information on customizing the builder.

Dynamic Properties with Testcontainers

Support for defining dynamic properties by injecting a `DynamicPropertyRegistry` into a `@Bean` method has been deprecated and attempting to do so will now fail by default. Instead of injecting `DynamicPropertyRegistry` in a `@Bean` method, implement a separate `@Bean` method that returns a `DynamicPropertyRegistrar`. This separate bean method should inject the container from which the properties' values will be sourced. This addresses some container lifecycle issues and ensures that the container from which a property’s value has been sourced will have been started before the property is used.

If you wish to continue injecting `DynamicPropertyRegistry` (at the risk of encountering the lifecycle issue described above), set `spring.testcontainers.dynamic-property-registry-injection` to either `warn` or `allow`. The former will log a warning while allowing the use of an injected `DynamicPropertyRegistry`. The latter will silently allow the use of an injected `DynamicPropertyRegistry`, fully restoring Spring Boot 3.3's behavior.

@AutoConfigureTestDatabase with Containers

The `@AutoConfigureTestDatabase` annotation now attempts to detect if a database has been sourced from a container. This should remove the need to add `replace=Replace.NONE` if you want to use the annotation with container databases.

If you need to revert to the old behavior, set `replace=Replace.AUTO_CONFIGURED` on the annotation.

Controlling Access to Actuator Endpoints

Support for enabling and disabling endpoints has been reworked, replacing the on/off support that it provided with a finer-grained access model. The new model supports only allowing `read-only` access to endpoint operations, in addition to disabling an endpoint (access of `none`) and fully enabling it (access of `unrestricted`).

The following properties have been deprecated:

- `management.endpoints.enabled-by-default`
- `management.endpoint.<id>.enabled`

Their replacements are:

- `management.endpoints.access.default`
- `management.endpoint.<id>.access`

Similarly, the `enableByDefault` attribute on `@Endpoint` has been deprecated with a new `defaultAccess` attribute replacing it.

As part of these changes, `enabled-by-default` is now applied consistently and irrespective of the use of `@ConditionalOnEnabledEndpoint`. If you lose access to an endpoint when upgrading, set `management.endpoint.<id>.access` to `read-only` or `unrestricted` or set `management.endpoint.<id>.enabled` to `true` to make the endpoint accessible again.

Additionally, a new property has been introduced that allows an operator to control the permitted level of access to Actuator endpoints:

- `management.endpoints.access.max-permitted`

This property caps any access that may have been configured for an endpoint. For example, if `management.endpoints.access.max-permitted` is set to `read-only` and `management.endpoint.loggers.access` is set to `unrestricted`, only read-only access to the loggers endpoint will be allowed.

Cloud Foundry ConditionalOnAvailableEndpoint Exposure

The `EndpointExposure.CLOUD_FOUNDRY` enum value used with the `@ConditionalOnAvailableEndpoint` has been deprecated in favor of `EndpointExposure.WEB`. Typical Spring Boot application will probably not be affected by this change, however, if you have custom Cloud Foundry specific actuator endpoint beans you should update your conditions to use `EndpointExposure.WEB`.

HtmlUnit 4.5

HtmlUnit has been upgraded to 4.5. With this upgrade comes a change in dependency coordinates from `net.sourceforge.htmlunit:htmlunit` to `org.htmlunit:htmlunit` and a change in package names from `com.gargoylesoftware.htmlunit.` to `org.htmlunit.`. When upgrading, update your build configuration and imports accordingly.

Selenium HtmlUnit 4.25

Selenium HtmlUnit has been updated to 4.25. With this upgrade comes a change in dependency coordinates from `org.seleniumhq.selenium:htmlunit-driver` to `org.seleniumhq.selenium:htmlunit3-driver`. When upgrading, update your build configuration accordingly.

WebJars Locator Integration

For faster startup times and efficient WebJars asset resolution, you will need to update your pom.xml/build.gradle to depend on `org.webjars:webjars-locator-lite` instead of `org.webjars:webjars-locator-core` (both dependencies are managed by Spring Boot). Note that `org.webjars:webjars-locator-core` support in Spring is now deprecated and will be removed in a future version. See the reference documentation section on this feature.

OkHttp Dependency Management Removed

Spring Boot no longer depends on OkHttp so it no longer manages its version. If your application has OkHttp dependencies, update its build to use an OkHttp version that meets its needs.

Netty in Native Image

Note This is not needed when upgrading directly to Spring Boot 3.4.1 or later.

Spring Boot 3.4.0 uses a version of Netty which isn't supported yet by the GraalVM reachability metadata included in the Native Build Tools. To get Netty working in a native image, you'll need to upgrade the GraalVM reachability metadata version manually.

For Maven:

```
<plugin>
        <groupId>org.graalvm.buildtools</groupId>
        <artifactId>native-maven-plugin</artifactId>
        <configuration>
                <metadataRepository>
                        <version>0.3.14</version>
                </metadataRepository>
        </configuration>
```

```
</plugin>
```

For Gradle:

```
graalvmNative {
        metadataRepository {
                version = '0.3.14'
        }
}
```

Deprecation of @MockBean and @SpyBean

`@MockBean` and `@SpyBean` have been deprecated in favor of `@MockitoBean` and `@MockitoSpyBean` in Spring Framework. The functionality provided by the Spring Framework annotations is not exactly the same as that offered by Spring Boot. For example, `@MockitoBean` is not supported on `@Configuration` classes and you may need to migrate to annotating fields on a test class instead.

Deprecations from Spring Boot 3.2

Classes, methods, and properties that were deprecated in Spring Boot 3.2 and marked for removal in 3.4 have been removed in this release. Please ensure that you aren't calling deprecated methods before upgrading.

Minimum Requirements Changes

Gradle

Gradle 7.5, 8.0, 8.1, 8.2 and 8.3 are no longer supported. Gradle 7.x (7.6.4 or later) or Gradle 8.x (8.4 or later) is now required.

New and Noteworthy

Tip Check the configuration changelog for a complete overview of the changes in configuration.

Structured Logging

Support for structured logging has been introduced with built-in support for Elastic Common Schema (`ecs`), Graylog Extended Log Format (`gelf`) and Logstash (`logstash`). To enable structured file logging set `logging.structured.format.file` to `ecs`, `gelf` or `logstash`. Similarly, to enable structured console logging set `logging.structured.format.console`.

To learn more about Spring Boot's support for structured logging, including how to define a custom format, see the reference documentation.

`@Fallback` Beans

`@ConditionalOnSingleCandidate` now supports `@Fallback` beans. The condition will match if there's a single primary bean or, if there are no primary beans, if there's a single non-fallback bean.

Defining Additional Beans

When type matching, bean-based conditions will now ignore any beans that are not default candidates. By declaring that a bean is not a default candidate (using `@Bean(defaultCandidate=false)`), a bean of an auto-configured type can now be defined without causing the auto-configure bean of the same type to back off. This reduces the configuration required to, for example, use two `DataSource` beans or two `EntityManagerFactory` beans in the same application.

ClientHttpRequestFactory Builders

A new `ClientHttpRequestFactoryBuilder` interface has been added which allows you to build `ClientHttpRequestFactory` instances for specific technologies. Builders allow for fine-grained customization of the underlying components, as well as a consistent way to apply common settings.

The following builders can be created for specific libraries using static factory methods from the interface:

- Apache HTTP Components (`ClientHttpRequestFactoryBuilder.httpComponents()`)
- Jetty Client (`ClientHttpRequestFactoryBuilder.jetty()`)
- Reactor Netty `HttpClient` (`ClientHttpRequestFactoryBuilder.reactor()`)
- JDK `HttpClient` (`ClientHttpRequestFactoryBuilder.jdk()`)
- Simple JDK `HttpURLConnection` (`ClientHttpRequestFactoryBuilder.simple()`)

See the updated reference docs for more details, including how to apply common settings using configuration properties.

Observability Improvements

Application Groups

The new `spring.application.group` property can be used to group applications together, for example if they all belong to some business unit or one bigger application arrangement. When this property is set, it’s also included in the log messages. This behavior can be controlled with the property `logging.include-application.group`. The application group is also automatically added to the OpenTelemetry `Resource`.

OTLP

It’s now possible to send OTLP spans over the gRPC transport. For this, set the new configuration property `management.otlp.tracing.transport` to `grpc`. This property defaults to `http`. Service connection support for this has been added, too.

The new properties under `management.otlp.logs` can be used to auto-configure OpenTelemetry's `OtlpHttpLogRecordExporter` and `SdkLoggerProvider`.

Other Observability Updates

The `ProcessInfoContributor` now also shows memory info about heap and non-heap usage.

New `management.otlp.tracing.export.enabled`, `management.wavefront.tracing.export.enabled` and `management.zipkin.tracing.export.enabled` properties can now be used to enable or disable trace exporting more finely grained.

AssertJ Support for MockMvc

Auto-configuration for `MockMvcTester` is provided when AssertJ is on the classpath. `MockMvcTester` lets you define the requests and the assertions using a fluent API. It can be injected anywhere `MockMvc` is.

For more details, see the dedicated section of the Spring Framework reference documentation.

Spring Pulsar

Configuration properties are now provided for configuring a default tenant and namespace. The defaults apply when consuming or producing messages with a topic URL that is not fully qualified. Configure them using the `spring.pulsar.defaults.topic.tenant` and `spring.pulsar.defaults.topic.namespace` configuration properties or define your own `PulsarTopicBuilder` bean. Set `spring.pulsar.defaults.topic.enabled=false` to disable the defaults.

A new `PulsarContainerFactoryCustomizer` interface has been added to support customization of the auto-configured `PulsarContainerFactory`.

The `spring.pulsar.consumer.subscription.name` configuration property now applies to the auto-configured Pulsar listener container.

Two new configuration properties for configuring the Pulsar client's concurrency have been introduced:

- `spring.pulsar.client.threads.io` controls the number of threads to be used for handling connections to brokers.
- `spring.pulsar.client.threads.listener` controls the number of threads to be used for message listeners.

Lastly, the new `spring.pulsar.listener.concurrency` property can be used to control the concurrency of the auto-configured Pulsar message listener container.

Couchbase Authentication

Client certificates can now be used to authenticate with a Couchbase cluster, as an alternative to basic username and password authentication. See the reference documentation for more details.

FreeMarker

FreeMarker variables that are used by the auto-configured FreeMarker's `Configuration` object can now be customized. To do so, define one or more beans of type `FreeMarkerVariablesCustomizer`. These are invoked according to their defined order (if any).

Embedded Broker support with ActiveMQ Classic

Now that ActiveMQ Classic supports an embedded broker again, the auto-configuration has been updated to support it.

Note that contrary to Spring Boot 2.7.x, the ActiveMQ starter is client only. To use the embedded broker, `org.apache.activemq:activemq-broker` should be added to your application.

Configuration Metadata

The default value of an `Enum` is now detected by the annotation processor. If you have added manual metadata to provide the value for a custom property, make sure to remove it.

Deprecating and Replacing Auto-configuration Classes

To make it easier to evolve auto-configuration, support for deprecating and replacing auto-configuration classes has been introduced. Replacements can be declared in a new `META-INF/spring/org.springframework.boot.autoconfigure.AutoConfiguration.replacements` file. To learn more, please refer to the reference documentation.

Base64 Resource Support and Automatic ProtocolResolver registration

Any property resolved to a `Resource` can now make use of the `Base64ProtocolResolver` without needing to explicitly register it. For example, you can now use it to specify the certificate location of a SAML2 relying party signing credential:

```
spring:
  security:
    saml2:
      relyingparty:
        registration:
          keycloak:
            entity-id: "saml-test"
            signing:
              credentials:
                - private-key-location: classpath:local.key
                  certificate-location: base64:LS...
```

You can also easily add your own protocol resolvers by adding them to a `META-INF/spring.factories` file under the `org.springframework.core.io.ProtocolResolver` key.

Virtual Threads

If virtual threads are enabled, the following components will now use them:

- `OtlpMeterRegistry`

- Undertow web server

Image Building Improvements

Spring Boot now uses the `paketobuildpacks/builder-jammy-java-tiny` by default. This builder supports ARM and x64 platforms out of the box.

A `trustBuilder` option has been added to the Maven and Gradle plugins for building OCI images. This option controls how the CNB lifecycle is invoked, providing improved security when using builders from untrusted sources. By default, builders from the Paketo project, Heroku, and Google are trusted. See the Maven or Gradle documentation for information.

An `imagePlatform` option has been added to the Maven and Gradle plugins for building OCI images. This option can be used to specify the operating system and architecture of any CNB builder, run, and buildpack images that are pulled in order to run CNB buildpacks. This can be used to build an image for an operating system and architecture that is different from the OS/architecture of the host platform, when the host platform supports emulation of the other OS/architecture (for example, when using Rosetta on a Mac with Apple silicon to emulate the AMD archicture on an ARM host). See the Maven or Gradle documentation for more information.

Docker Compose Improvements

Docker Compose now supports multiple Docker Compose configuration files.

Command Line Arguments

The new properties `spring.docker.compose.start.arguments` and `spring.docker.compose.stop.arguments` can be used to specify additional command line arguments that are passed to the Docker Compose subcommands when starting and stopping services. A new `spring.docker.compose.arguments` property has been added to pass arguments to Docker Compose.

Updated Support

- Postgres `POSTGRES_HOST_AUTH_METHOD=trust` environment variable is now supported.
- Support for Redis Stack and Redis Stack Server has been added using the `redis/redis-stack` and `redis/redis-stack-server` container images respectively.
- Support for Grafana LGTM has been added using the `grafana/otel-lgtm` container image.
- Support has been added for Hazelcast (using `HazelcastConnectionDetails`).
- Support has been added for OTLP logging.

Testcontainers Improvements

- Support has been added for `org.testcontainers.kafka.KafkaContainer`.
- Support for Redis Stack and Redis Stack Server has been added using the `redis/redis-stack` and `redis/redis-stack-server` container images respectively.

- Support has been added for `org.testcontainers.grafana.LgtmStackContainer`.
- Support has been added for Hazelcast (using `HazelcastConnectionDetails`).
- Support has been added for OTLP logging.
- Support for `RedisContainer` has been added

Actuator

Pluggable Actuator Exposers

It is now possible to extend Spring Boot to expose actuator endpoints in a pluggable way. The new `EndpointExposureOutcomeContributor` interface can be implemented to influence `@ConditionalOnAvailableEndpoint` conditions.

This extension should make it easier to offer additional platform integrations similar to our existing Cloud Foundry support.

SSL information and health check

If you're using SSL bundles, there's now a new endpoint showing SSL information (validity dates, issuer, subject, etc.) available under `/actuator/info`. This endpoint also shows soon to be expired certificates to alert you that they need to be rotated soon. There's a new configuration property named `management.health.ssl.certificate-validity-warning-threshold` to configure the threshold.

A new health check monitoring the SSL certificates has been added, too. If a certificate is invalid, it sets the status to `OUT_OF_SERVICE`.

Additional info in `/actuator/scheduledtasks` endpoints

The `/scheduledtasks` Actuator endpoint now exposes additional metadata about scheduled tasks, such as "next scheduled execution time" and "last execution time, status and exception".

Dependency Upgrades

Spring Boot 3.4 moves to new versions of several Spring projects:

- Spring AMQP 3.2
- Spring Authorization Server 1.4
- Spring Batch 5.2
- Spring Data 2024.1
- Spring Framework 6.2
- Spring HATEOAS 2.4
- Spring Integration 6.4
- Spring Kafka 3.3

- Spring Pulsar 1.2
- Spring Security 6.4
- Spring Session 3.4

Numerous third-party dependencies have also been updated, some of the more noteworthy of which are the following:

- Apache Http Client 5.4
- AssertJ 3.26
- Artemis 2.37
- Elasticsearch Client 8.15
- Flyway 10.20
- Gson 2.11
- Hibernate 6.6
- HtmlUnit 4.5
- JUnit Jupiter 5.11
- Jackson 2.18.0
- Jedis 5.2
- Kafka 3.8
- Lettuce 6.4
- Liquibase 4.29
- Log4j 2.24
- MariaDB 3.4
- Micrometer 1.14
- Micrometer Tracing 1.4
- Mockito 5.13
- MongoDB 5.2.0
- MySQL 9.1
- OpenTelemetry 1.41
- Oracle Database 23.4
- R2DBC MySQL 1.3
- Rabbit AMQP Client 5.22
- Rabbit Stream Client 0.18.0
- Reactor 2024.0

- Selenium 4.25
- Testcontainers 1.20.3
- XMLUnit 2.10

Miscellaneous

Apart from the changes listed above, there have also been lots of minor tweaks and improvements including:

- You can now use a `Customizer<Liquibase>` bean to customize Liquibase before it is being used
- The properties used to create a JCache `CacheManager` can now be customized by defining a `JCachePropertiesCustomizer` bean.
- The `RequestToViewNameTranslator` used by Spring MVC can now be customized by defining a bean named `viewNameTranslator`.
- Lettuce's `ClientOptions` can now be customized using a `LettuceClientOptionsBuilderCustomizer` bean. For broader configuration of the whole `LettuceClientConfiguration`, continue to use `LettuceClientConfigurationBuilderCustomizer`.
- The new customizer `ProxyConnectionFactoryCustomizer` can be used to customize a R2DBC `ProxyConnectionFactory`.
- An audit event is now published if a Spring Security logout happens.
- TLS on `JavaMailSender` can now be configured with SSL bundles using the new properties `spring.mail.ssl.*`
- GSON's strictness can be configured using the new `spring.gson.strictness` property.
- `@Name` can now be used on the field of a JavaBean-style configuration property to customize its name.
- When derived from another `DataSource`, `DataSourceBuilder` can now determine the Driver class name using the source DataSource's URL if it does not expose the Driver class name.
- Liveness and Readiness health probes are now automatically enabled on Cloud Foundry Platforms.
- The new property `spring.application.version` can be used to read and set the application version. The default value for the property is taken from the `Implementation-Version` of the manifest.
- The auto-configured `EntityManagerFactoryBuilder` defines the native (e.g. Hibernate) properties as well.

- Spring Integration’s `TaskScheduler` is now virtual thread aware, even if `@EnableScheduling` hasn’t been used.
- `@ConditionalOnAvailableEndpoint` now has a `value` alais for `endpoint`.
- A new configuration property, `spring.data.web.pageable.serialization-mode`, for configuring Spring Data Web’s serialization mode has been added.
- When using the `SpringApplication.from(…)` syntax, it’s now possible to specify the additional profiles to activate.
- The Spring Boot plugin no longer sets `BP_NATIVE_IMAGE: true` in the buildpack environment.
- Registered `@ConfigurationProperties` beans now respect `@DependsOn`, `@Description`, `@Fallback`, `@Lazy`, `@Primary`, `@Scope` and `@Role` annotations.
- Log4j2’s `MultiFormatStringBuilderFormattable` is now supported in structured logging
- A new configuration property, `spring.jms.listener.max-messages-per-task`, for configuring the maximum number of messages a listener processes in one task has been added.
- The default security configuration now exposes health groups mapped to additional paths. In addition, both `EndpointRequest` classes now offer `toAdditionalPaths(…)` methods.
- The partitioned attribute of session cookies can now be set through properties.
- A new `server.jetty.max-form-keys` property has been added to customize Jetty’s max form keys.
- New properties `management.otlp.logging.connect-timeout` and `management.otlp.tracing.connect-timeout` have been added to configure the connect timeout to the OTLP collector.
- Support for gRPC transport when shipping logs over OTLP has been added.
- When binding a directory in the container which is used by buildpacks in the build process, a warning is now shown.
- When building a native image with `--enable-sbom=sbom`, this SBOM is now auto-detected.
- The `DatabaseDriver` enum now support the ClickHouse JDBC driver.
- The new properties `management.logging.export.enabled` and `management.otlp.logging.export.enabled` can be used to disable log exporting.
- The `TaskExecutor` used by Spring Batch can be customized by defining a `TaskExecutor` bean annotated with `@BatchTaskExectuor`.

- Spring Session auto-configuration now supports the `indexed` repository type in reactive web applications.
- A warning from `HikariCheckpointRestoreLifecycle` will be logged if pool suspension isn’t configured and a checkpoint is created.

Deprecations in Spring Boot 3.4

- `spring.gson.lenient` in favor of `spring.gson.strictness`.
- `@MockBean` and `@SpyBean` in favor of Spring Framework’s `@MockitoBean` and `MockitoSpyBean` respectively.
- `org.springframework.boot.ResourceBanner#getApplicationVersion(Class<?>)` in favor of `spring.application.version` property.
- `org.springframework.boot.SpringApplication#logStartupInfo(boolean)` in favor of `org.springframework.boot.SpringApplication#logStartupInfo(ConfigurationApplicationContext)`.
- `org.springframework.boot.logging.logback.ApplicationNameConverter` in favor of `org.springframework.boot.logging.logback.EnclosedInSquareBracketsConverter`.
- `org.springframework.boot.actuate.autoconfigure.endpoint.expose.EndpointExposure#CLOUD_FOUNDRY` in favor of `org.springframework.boot.actuate.autoconfigure.endpoint.expose.EndpointExposure#WEB`.
- `org.springframework.boot.actuate.autoconfigure.tracing.otlp.OtlpTracingConnectionDetails#getUrl()` in favor of `getUrl(Transport)`
- `org.springframework.boot.actuate.autoconfigure.tracing.OpenTelemetryAutoConfiguration` in favor of `org.springframework.boot.actuate.autoconfigure.tracing.OpenTelemetryTracingAutoConfiguration`
- `OtlpAutoConfiguration` has been in favor of `OtlpTracingAutoConfiguration`
- `management.endpoints.enabled-by-default` and `management.endpoint.<id>.enabled` in favor of `management.endpoints.access.default` and `management.endpoint.<id>.access` respectively
- `enableByDefault` on `@Endpoint` in favor of `defaultAccess`