Apache Maven for Spring Boot

Declaring properties will override the inherited one in pom.
Declaring dependency version inside properties -> makes it a curated dependency.
We can override these dependencies in our own pom
(Provided in bom)...

Spring Boot project has parent pom spring boot starter pom (declares some properties and overrides plugin behaviour) and now this pom has parent spring dependencies pom (it declares bom and their versions in properties which we can override), and now this has parent spring boot build (it contains descriptions about modules), and finally we have super pom.

The parent.pom can't contain any java code, only Maven specifics e.g. See:

Spring boot creates a fat jar. All the resources and dependencies will be included in the jar...
All the dependencies are put in lib folder under jar.

Main-class in manifest file is starting point and it invokes the Start-class mentioned in manifest file.
We can deploy the jar file on docker container.

Combination of spring-boot-starter-parent and maven-plugin dependency help in building the jar.

jar.original -> normal jar that would have been built.
The jar -> will contain dependencies also.

Spring boot plugin has following goals:
Build-help, help, repackage, run, start, stop

Start, stop are to be used inside the build context.
Run kicks of the build phase.
Mvn spring-boot:run

Spring Boot Integration tests:
By default spring boot does not kick in failsafe plugin.
We can configure it.

```
<execution>
   <id>build-info-goal</id>
   <goals>
      <goal>build-info</goal>
```

```
    </goals>
</execution>
```

Hooks into compile resources phase of default build cycle. It actually enables it.

```
<execution>
    <id>build-info-goal</id>
    <goals>
        <goal>build-info</goal>
    </goals>
    <configuration>
        <additionalProperties>
            <java.version>${java.version}</java.version>
            <some.custom.prop>${some.custom.prop}</some.custom.prop>
        </additionalProperties>
    </configuration>
</execution>
```

Adding the additional properties also reflects inside the build-info.

Build-info is useful for sharing the project.

```
<plugin>
    <groupId>pl.project13.maven</groupId>
    <artifactId>git-commit-id-plugin</artifactId>
</plugin>
```

Adding this plugin helps in uploading the source info about the host name, email, git last commit, etc to the git. git.properties forms in classes folder inside target folder.
management.info.git.mode = full in application.properties gives full git info at spring boot actuator endpoint.

Putting
<spring-boot.repackage.skip>false</spring-boot.repackage.skip>
Helps in making normal jar for the module instead of making fat jar.

Let's say we created a project and made it parent and removed everything from it. Now we created 2 modules as its children. Now we moved every stuff removed stuff into one of the module and left the other module. Let's say in pom which is empty for parent we added the spring boot plugin in build section. What we will try that the modules inheriting from it should be converted into fat jar and every fat jar should have main class. But we do not want one of them to be fat jar , so we can put the above line in its pom. So what we are doing is moving the plugins to parent and telling its child to inherit from there.
<build>
<plugins>

```
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
```