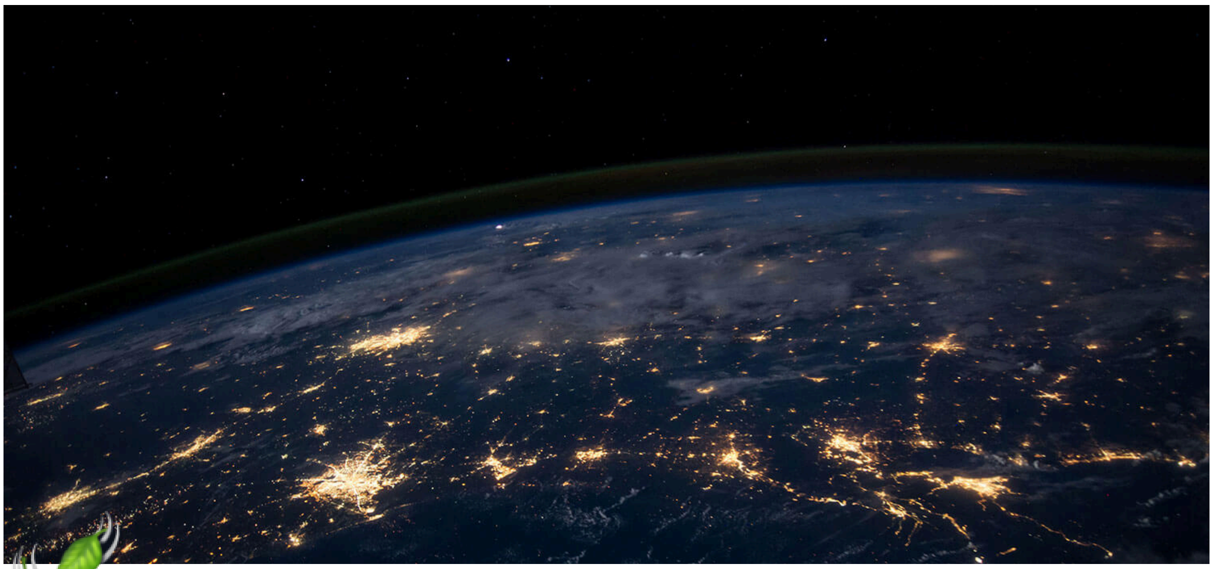

Section 2- What is Maven and how it is used?

🍃 Setting Up Dependencies in a Spring or Spring Boot Application

When building a **Spring** or **Spring Boot** application, we need various dependencies to ensure our application runs smoothly. These dependencies must be added to the **class-path** so that our application can access them when needed.

🍃 Dependency Management with Maven

To manage dependencies efficiently, we use **Maven**, a powerful build automation and dependency management tool. Instead of manually adding dependencies, we define them in a single configuration file called `pom.xml` (Project Object Model).

Maven ensures that all dependencies listed in `pom.xml` are automatically downloaded and added to the classpath, making them available for our application.

> 🍃 Maven is our go-to tool for managing dependencies in Java-based projects.

🍃 Lets see how the maven works:

Maven follows the parent child structure. It means that the parent will have something that are already pre defined and then the child will always inherit from the parent.

In very simple terms, the group Id, artifact id and version which here is 3.4.2 what happens here is the parent tag and everything that is defined within it is the default setting. Now what we say here is that any dependency that we want to add in our class path should be of the version matching to the version mentioned in the parent tag. The dependency should be the part of springboot.framework.boot and it should also get all the default settings from the starter-parent.

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.4.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
```

Now when we declare the dependency we will just tell the name and we will already get the inherited default dependencies.

💡 imagine that you go to the factory which makes 100 types of different products. Every product is manufactured in multiple dates and in multiple colors and in multiple price ranges.
💠Now what you go and say is that I want your products which were manufactured with the below details:

🔥in month of jan 2025
🔥color combination as grey and black
🔥 price of the product ranging from 100 to 200 rupees.

💠This is your default settings.

Now you will start naming the product and the worker there will automatically will bring that respective product which matches this details mentioned above.

This is specifications are getting inherited from the parent tag. Every time you are declaring the child dependency (product that you want) the worker (in this case is maven) is referring to the parent and getting the product matching this details.

💠Now lets say that there is a single product which doesn't match any one of the filter that you have set in the parent tag, at that time you will exclusively define that specific details and ask that worker for this specific product it is okay if the color is maroon and olive green.

This is how the dependency (child part) and the parent part in maven work.

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

Here the dependency is saying same thing that get me everything that comes in the starter-web package that is the part of springboot framework (defined in the parent tag) and let the version be the same as defined in the parent tag. Hence here in the dependency we have not defined its version.