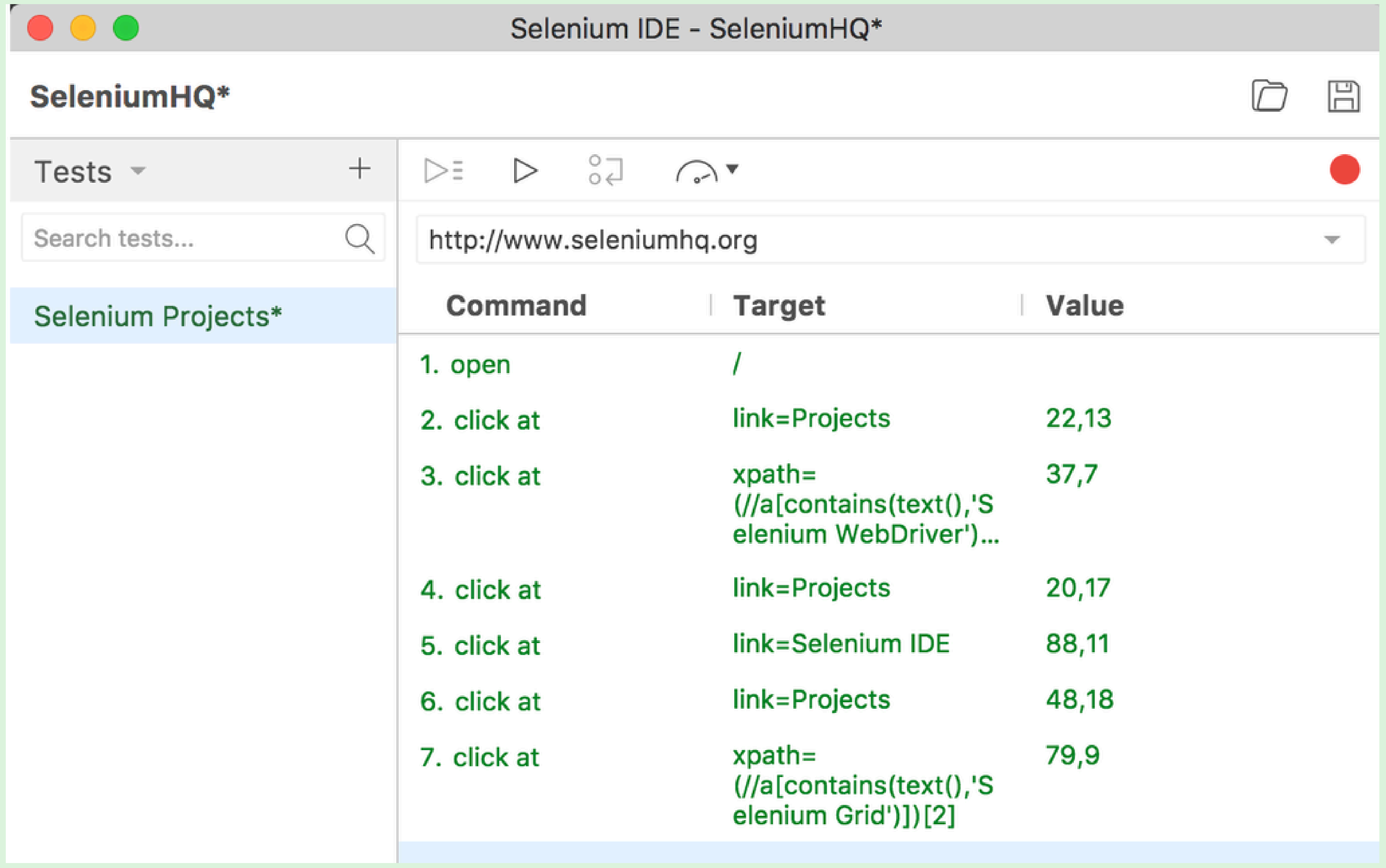
Selenium Automation Testing with Java

HISTORY OF SELENIUM

Selenium is an open-source tool initially developed by Jason Huggins in 2004. It started as a basic tool to automate web applications and evolved over time into a powerful suite of tools supporting multiple browsers and programming languages. Selenium gained popularity due to its simplicity and flexibility, making it a go-to choice for web automation testing.

Selenium Tools Suite

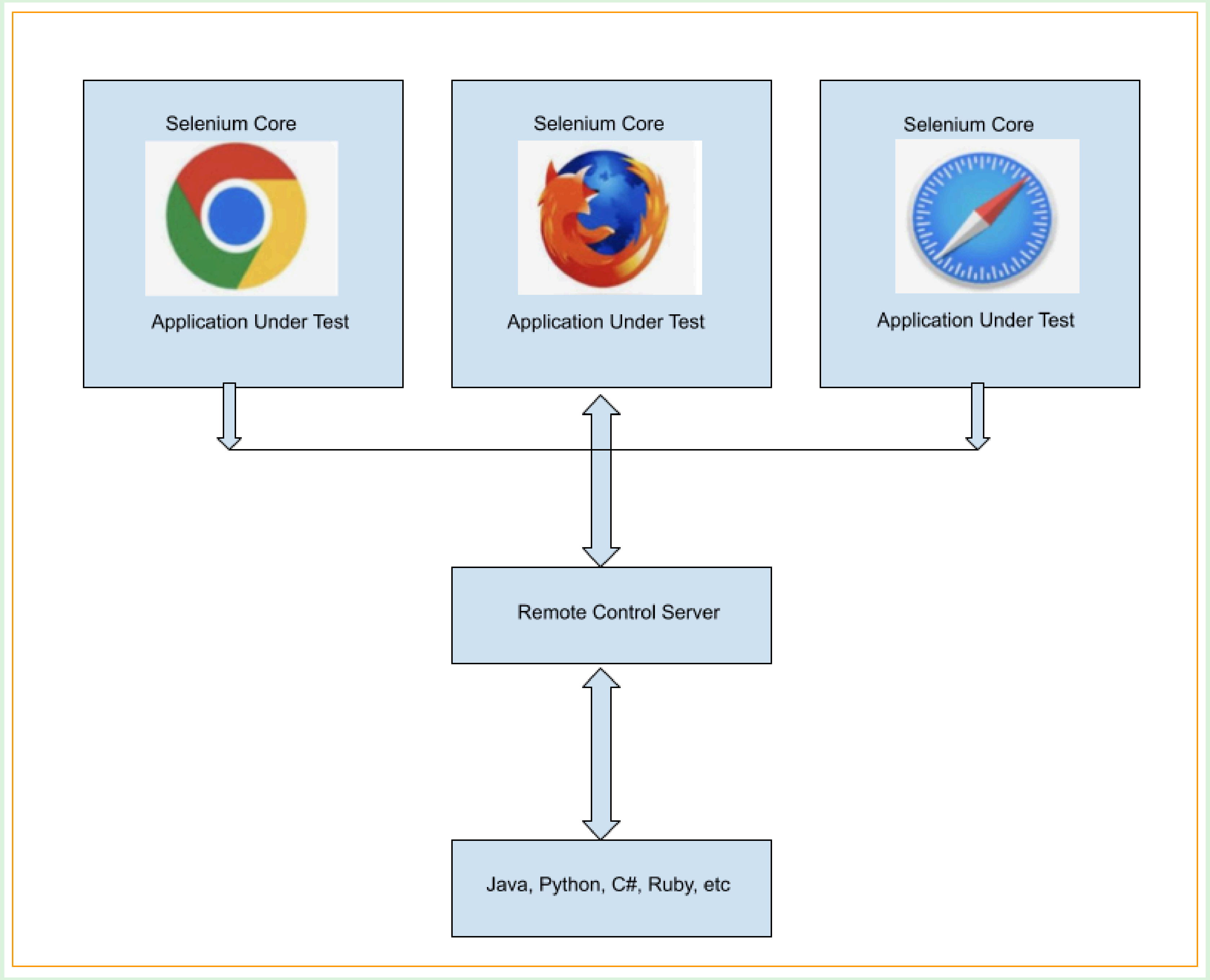
Selenium IDE



Command	Target	Value
1. open	/	
2. click at	link=Projects	22,13
3. click at	xpath=(//a[contains(text(),'Selenium WebDriver')...	37,7
4. click at	link=Projects	20,17
5. click at	link=Selenium IDE	88,11
6. click at	link=Projects	48,18
7. click at	xpath=(//a[contains(text(),'Selenium Grid')])[2]	79,9

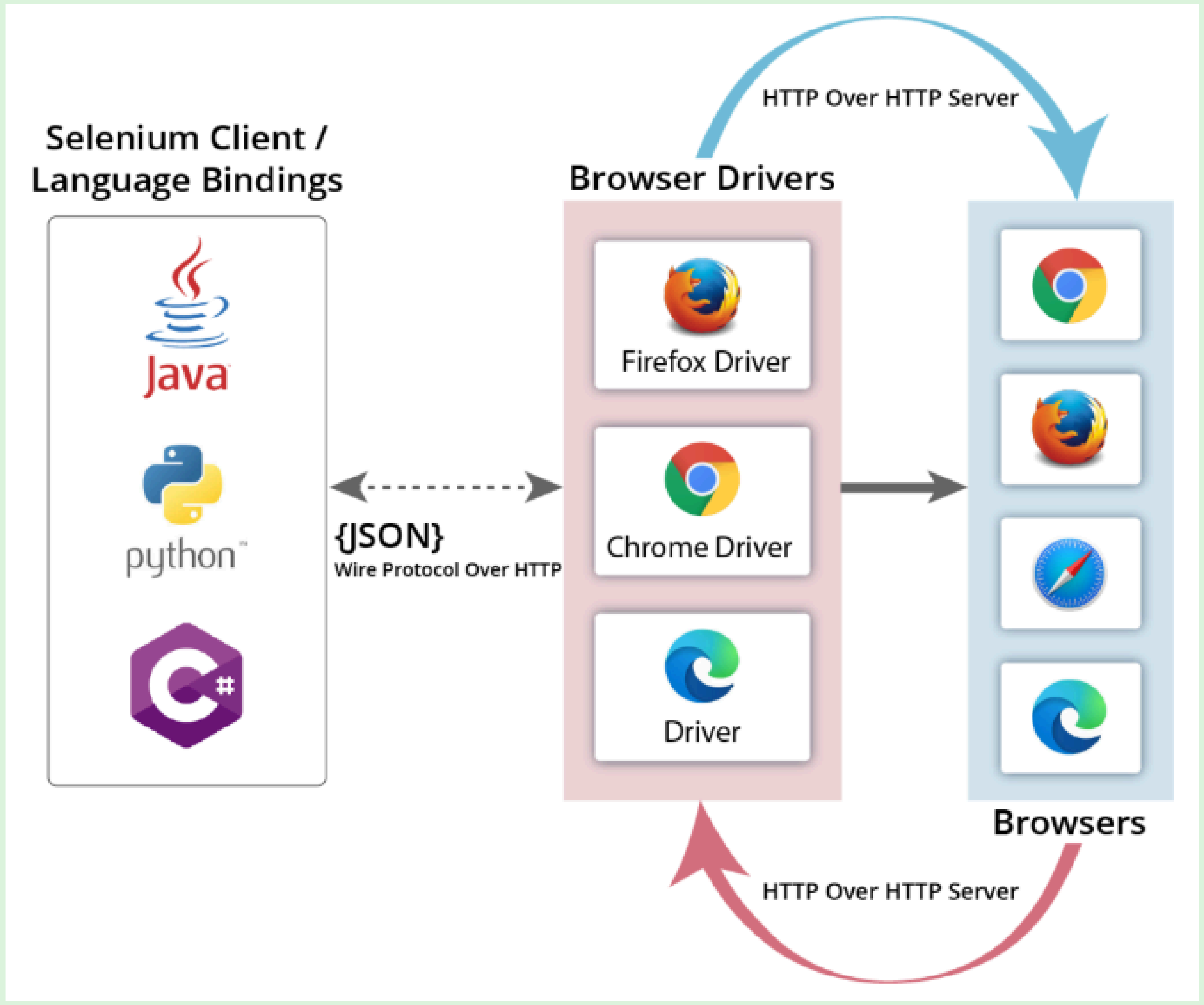
A record-and-playback tool for creating quick test cases. It is a browser extension that allows testers to record actions and generate test scripts without writing code. Selenium IDE is ideal for beginners or quick test case development.

Selenium RC (Remote Control)



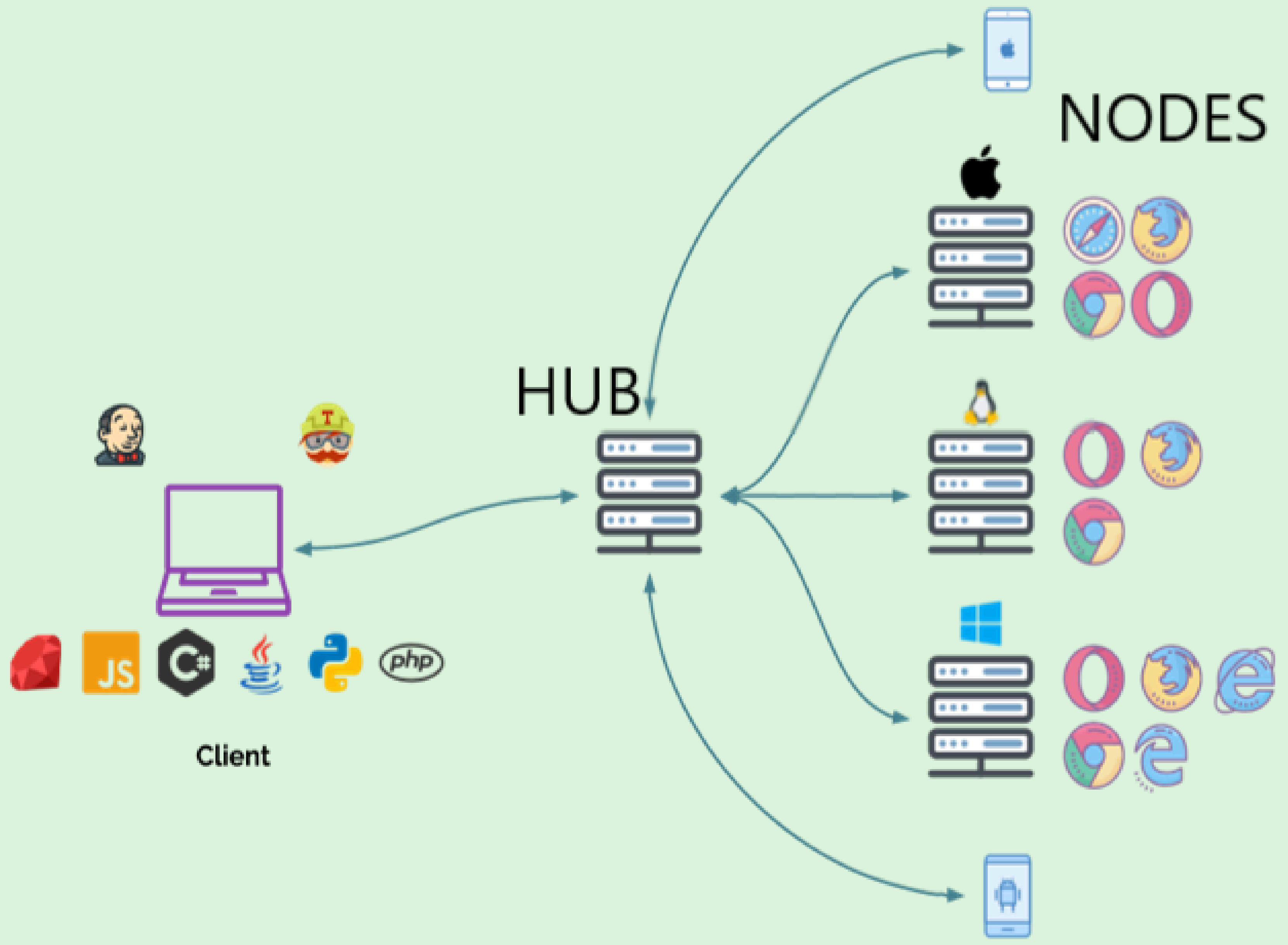
A server-based tool that allowed developers to write tests in multiple programming languages. It served as the main component before WebDriver and required starting a server to interact with browsers. While powerful for its time, it was slower and more complex compared to modern tools.

Selenium WebDriver



A programming interface that interacts directly with browsers, enabling faster and more reliable test execution. It supports multiple browsers like Chrome, Firefox, and Edge, and is the most widely used component of Selenium today.

Selenium Grid



A tool designed for running tests in parallel across multiple machines and browsers. It is especially useful for testing at scale, as it significantly reduces execution time by distributing the workload.

Advantages of Selenium

01. Open Source

Selenium is free to use, making it accessible to organizations of all sizes. Its open-source nature has fostered a large and active community that continually improves the tool, shares solutions, and provides support.

02. Cross-Browser Compatibility

Selenium supports testing on various browsers like Chrome, Firefox, Safari, and Edge. This ensures consistent application behavior across different browser environments.

03. Language Support

Selenium provides bindings for multiple programming languages, including Java, Python, C#, and Ruby, allowing developers and testers to use a language they are most comfortable with.

04. Parallel Testing

Selenium Grid enables the execution of tests in parallel across multiple browsers and machines, significantly reducing the time required for test execution.

05. Integration with Tools

Selenium integrates seamlessly with popular tools like Jenkins (for CI/CD pipelines), TestNG/JUnit (for test management), and Maven/Gradle (for dependency management). This flexibility makes it suitable for various automation and development workflows.

Disadvantages of Selenium

01. No Built-In Reporting

- Requires third-party tools for detailed reporting.

02. Limited Mobile Testing Support

- Not designed for mobile applications

03. Steep Learning Curve

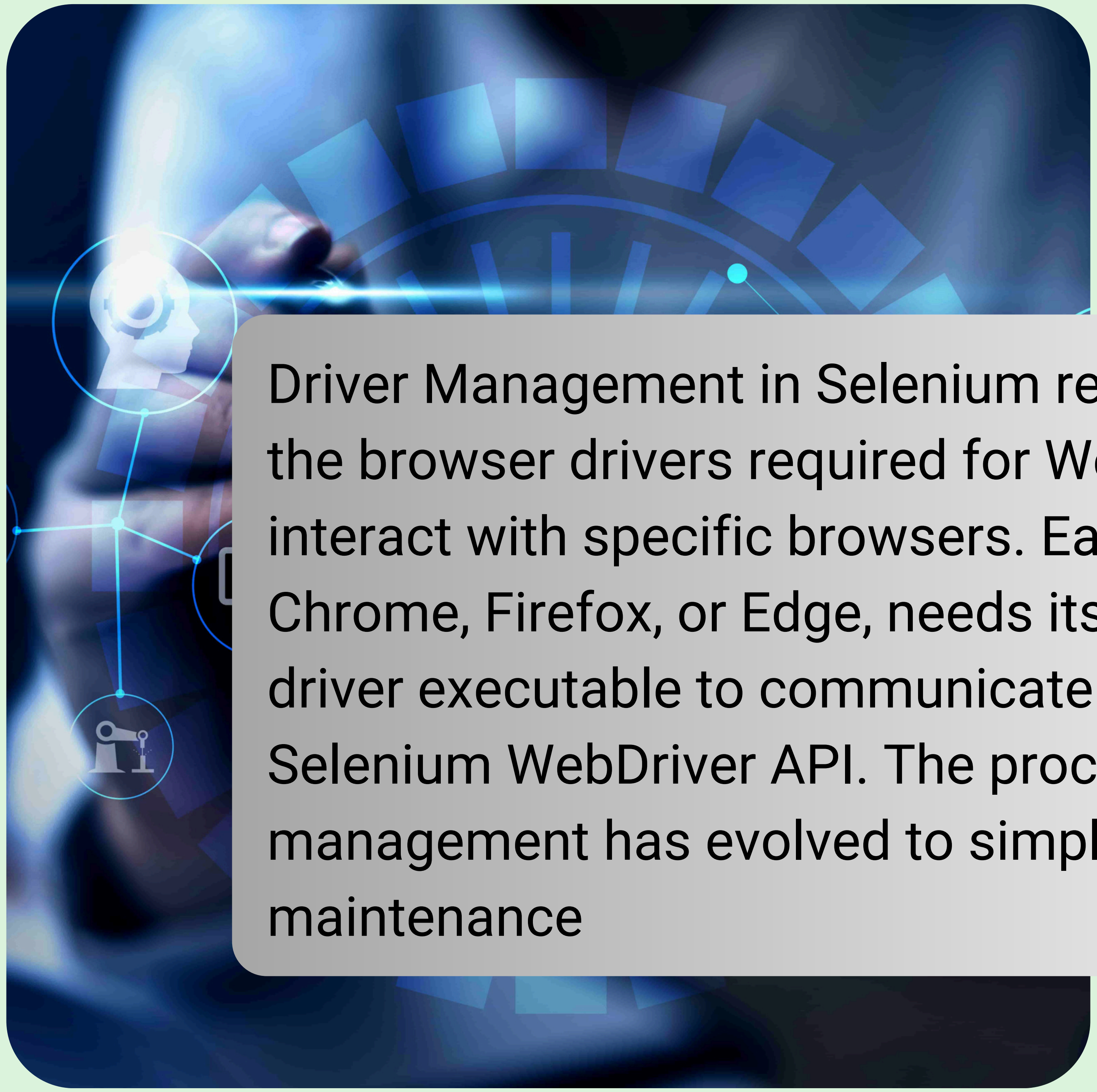
- Demands knowledge of programming and testing concepts.

03. No Support for Image-Based Testing

- Selenium cannot validate images or visual elements on a web page directly without integrating additional tools.: Relies on community support for issue resolution.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec quis erat et.

Evolution of Driver Management in Selenium



Driver Management in Selenium refers to handling the browser drivers required for WebDriver to interact with specific browsers. Each browser, like Chrome, Firefox, or Edge, needs its corresponding driver executable to communicate with the Selenium WebDriver API. The process of driver management has evolved to simplify setup and maintenance

01

System.setProperty()

02

WebDriverManager

03

Selenium Manager

System.setProperty()

- Manually set the path for browser drivers.

```
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
WebDriver driver = new ChromeDriver();
```

WebDriverManager

- Automatically downloads and manages drivers.

```
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
```

Selenium Manager

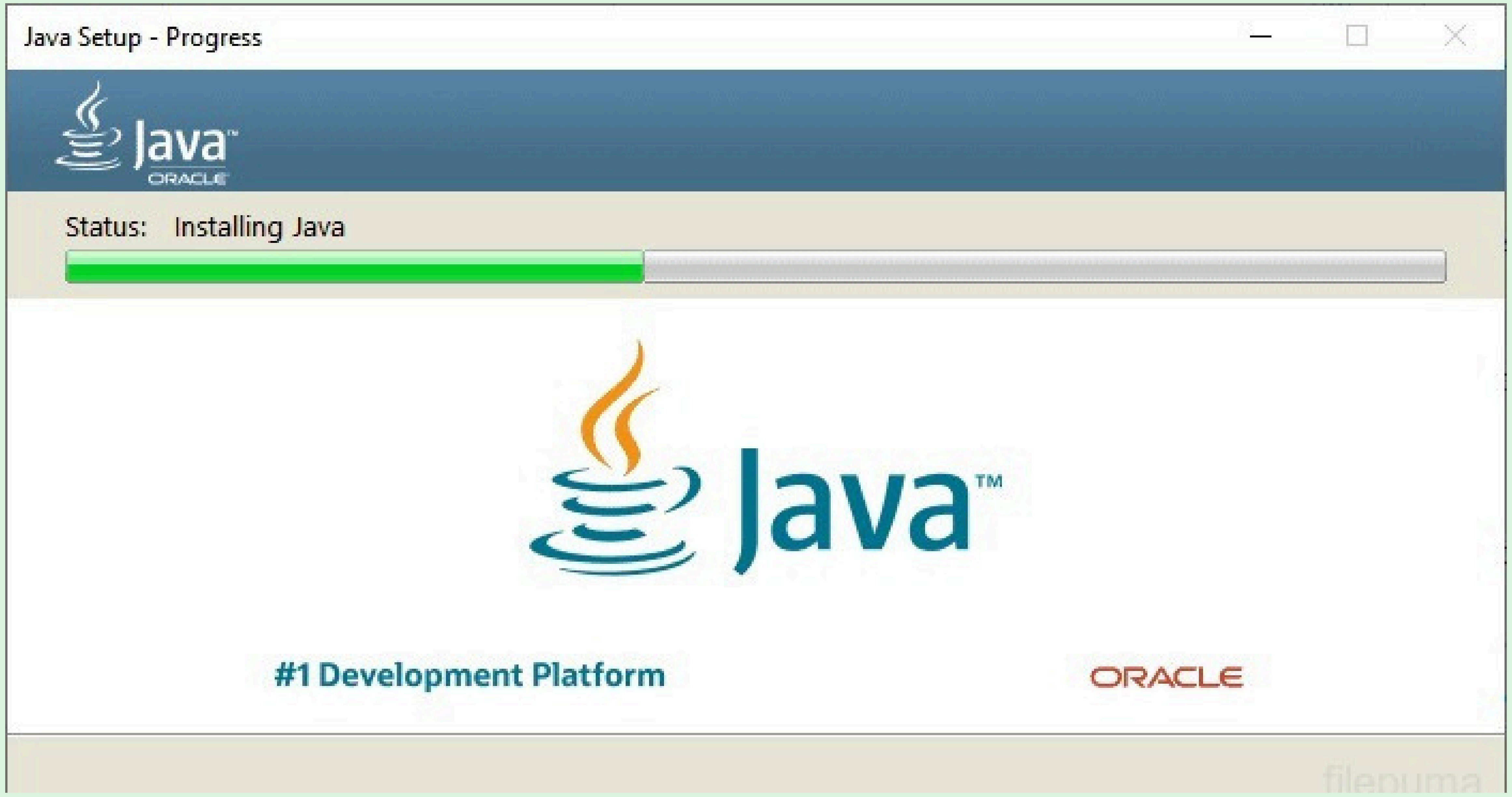
- A built-in tool in Selenium 4.10+ that simplifies driver management without additional libraries.

```
WebDriver driver = new ChromeDriver();
```

Setting Up Automation Testing with Selenium and Java

Step 1: Set Up Java

Download and install the Java Development Kit (JDK) from Oracle's official website or OpenJDK.


Java Setup - Progress

Status: Installing Java

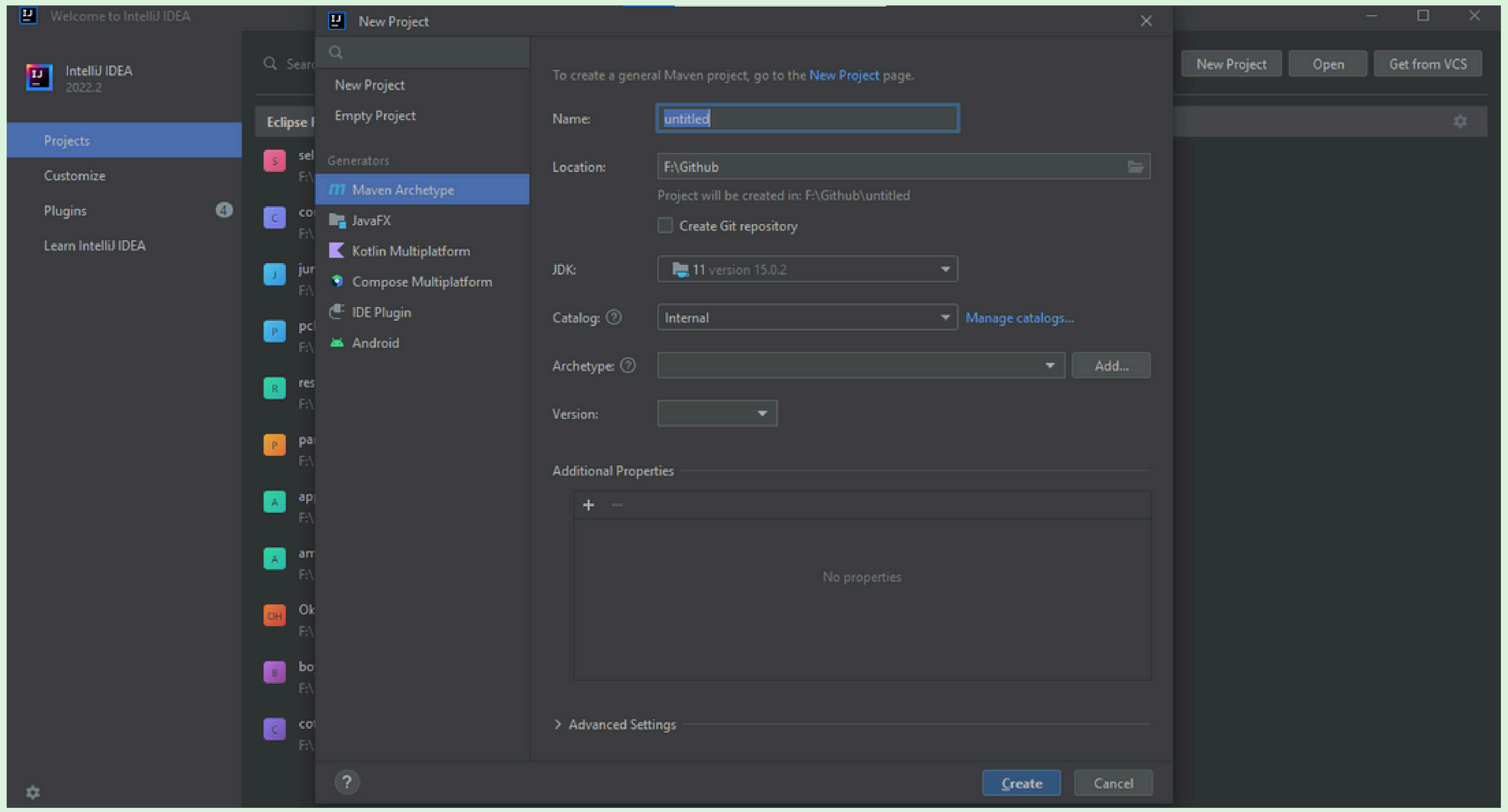
Step 2: Set Up an IDE

You can use IntelliJ IDEA or Eclipse to write and manage your Selenium scripts.

Step 3: Create a Maven Project

A Maven project is a software project management tool that simplifies the build process and dependency management. Maven uses a central configuration file called pom.xml to define project dependencies, build configurations, and other project-related information.

- There are main 3 key dependences that define the external libraries required for your project.



New Project

To create a general Maven project, go to the New Project page.

Name: untitled

Location: F:\Github

Project will be created in: F:\Github\untitled

Create Git repository

JDK: 11 version 15.0.2

Catalog: Internal   Manage catalogs...

Archetype: Add...

Version:

01. Selenium Java - Provides the core functionalities of Selenium for automating web browsers. This dependency includes the WebDriver API and other necessary components for browser interactions.

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.27.0</version>
</dependency>
```

When specifying Selenium dependencies, ensure that the version is compatible with the browser’s CDP version. This compatibility is crucial for the WebDriver to function correctly with the browser, especially for features like debugging and automation through the Chrome DevTools Protocol. Mismatched versions may result in unexpected behavior or failures during test execution.

02.TestNG - A testing framework that helps manage and execute test cases effectively. It provides annotations, grouping, and parallel execution capabilities to streamline test workflows.

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.10.2</version>
    <scope>test</scope>
</dependency>
```

03. Log4j - A logging library used to capture and manage logs during test execution. It enables effective debugging and tracking by recording execution details and errors in a structured format.

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <version>2.24.2</version>
    <scope>test</scope>
```

www.linkedin.com/in/sandunikanawullage

END