WebDriverManager: the Swiss Army Knife for Selenium WebDriver

Selenium Conference 2022
July 29, 2022

Boni García

 boni.garcia.c@saucelabs.com  https://bonigarcia.dev

 @boni_gg  https://github.com/bonigarcia

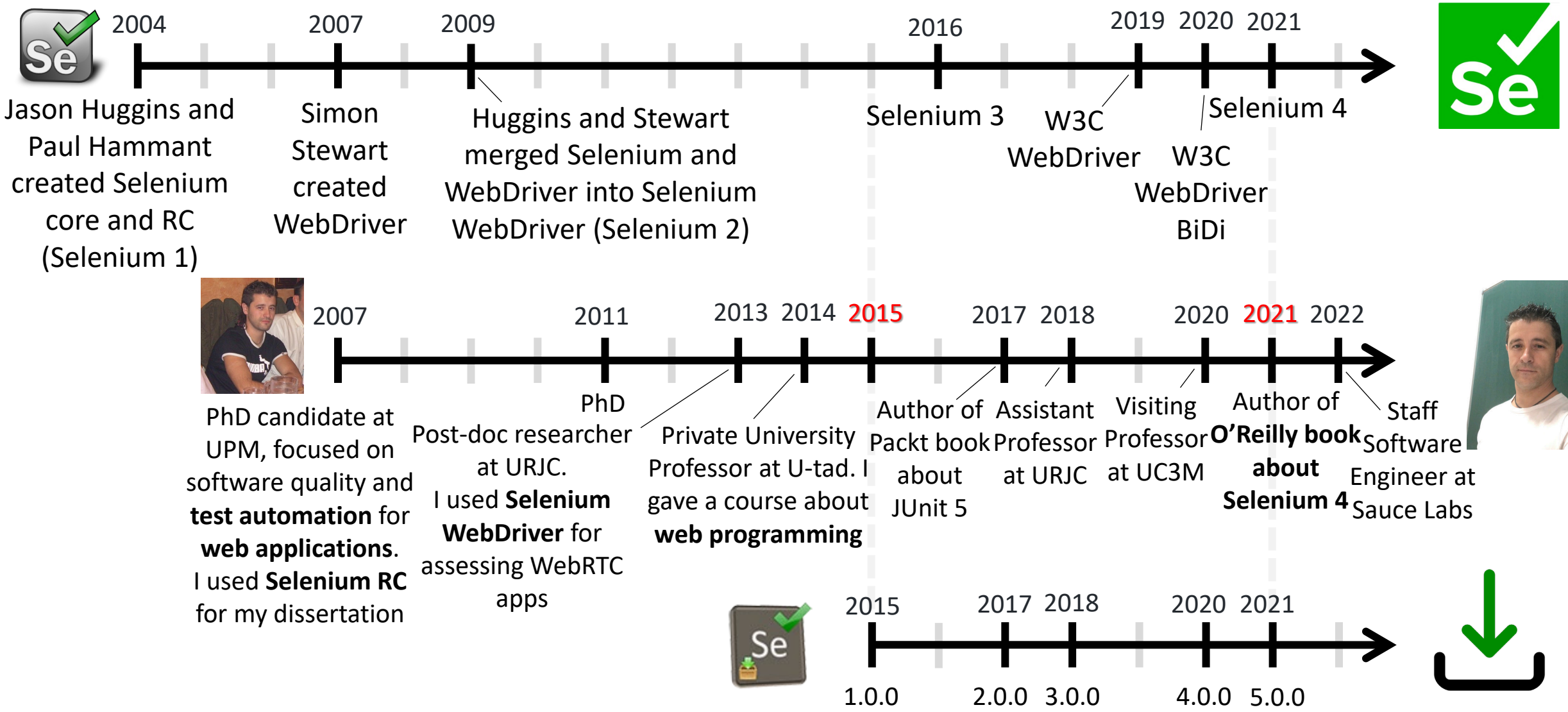
Presentation

WebDriverManager 

“Automated driver management and other helper features for Selenium WebDriver in Java

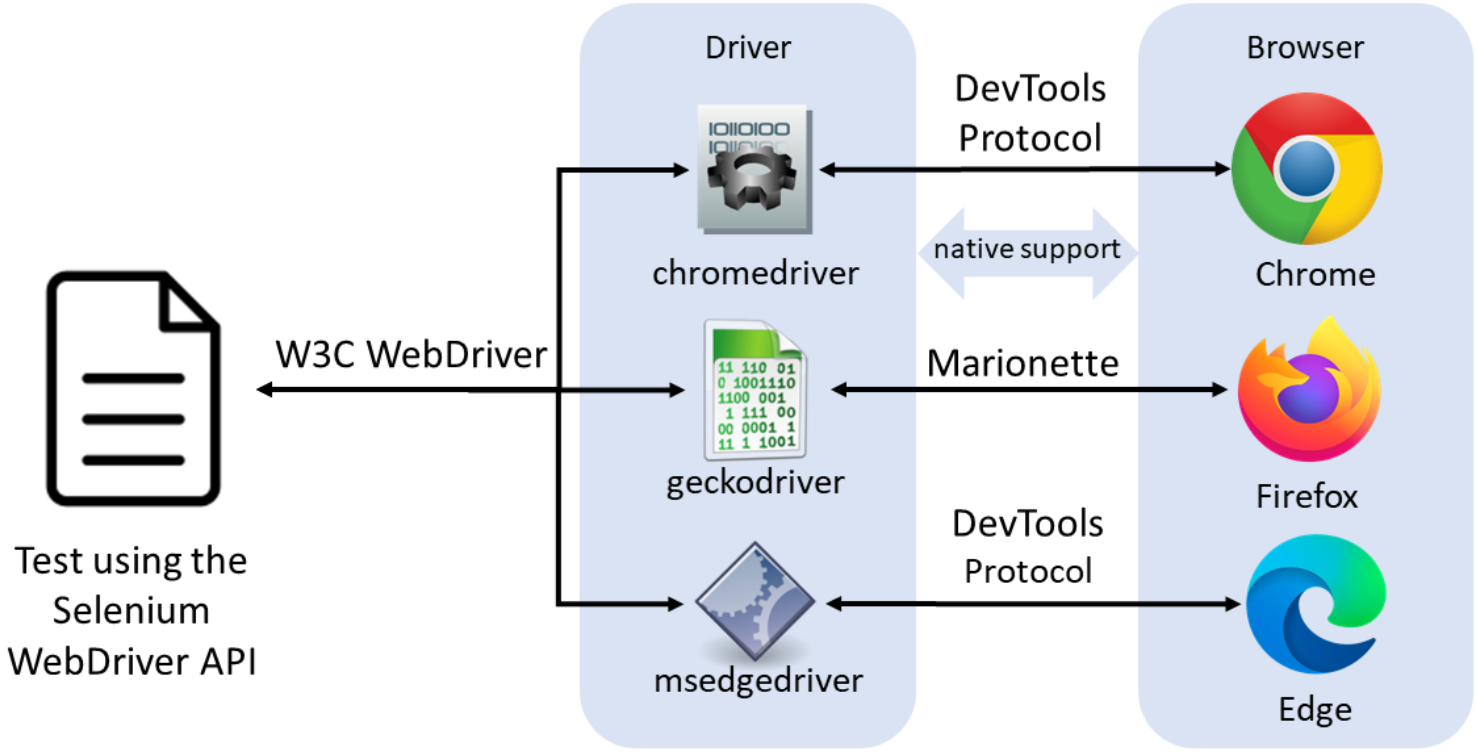
Source code: https://github.com/bonigarcia/webdrivermanager
Doc: https://bonigarcia.dev/webdrivermanager/

1. The little history of WebDriverManager



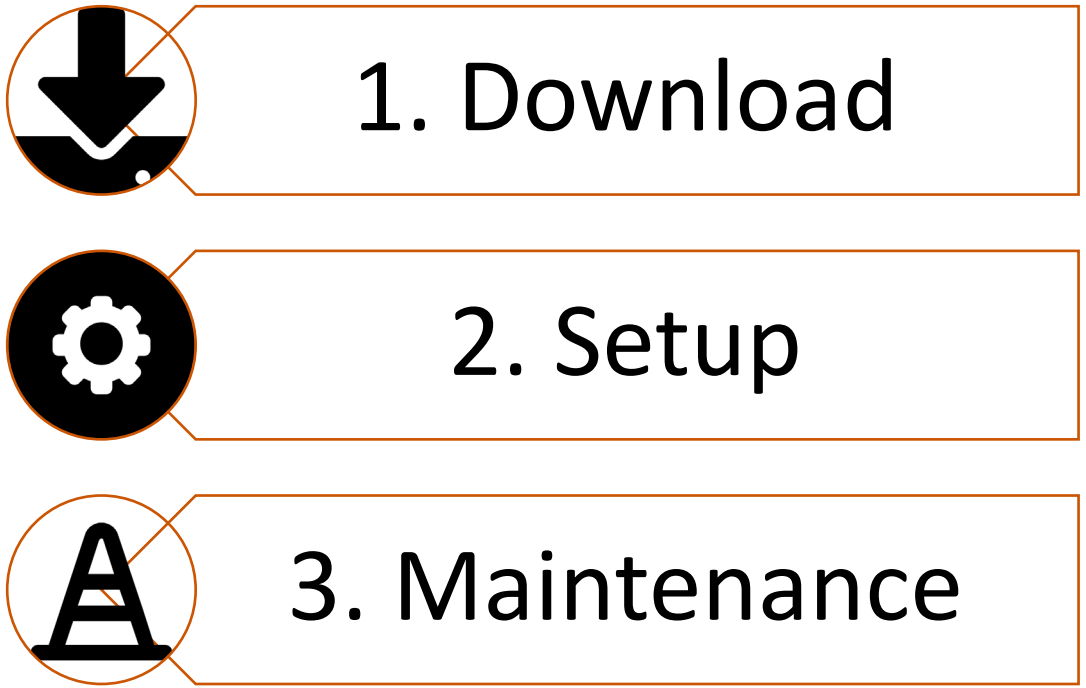
2. The original motivation: automated driver management

- Selenium WebDriver (i.e. Selenium 2+) uses the native capabilities of each browser to support the automation



2. The original motivation: automated driver management

- For a given browser (e.g. Chrome), I called **driver management** to the process of resolving its proper driver (e.g., chromedriver), and it is composed of three steps:



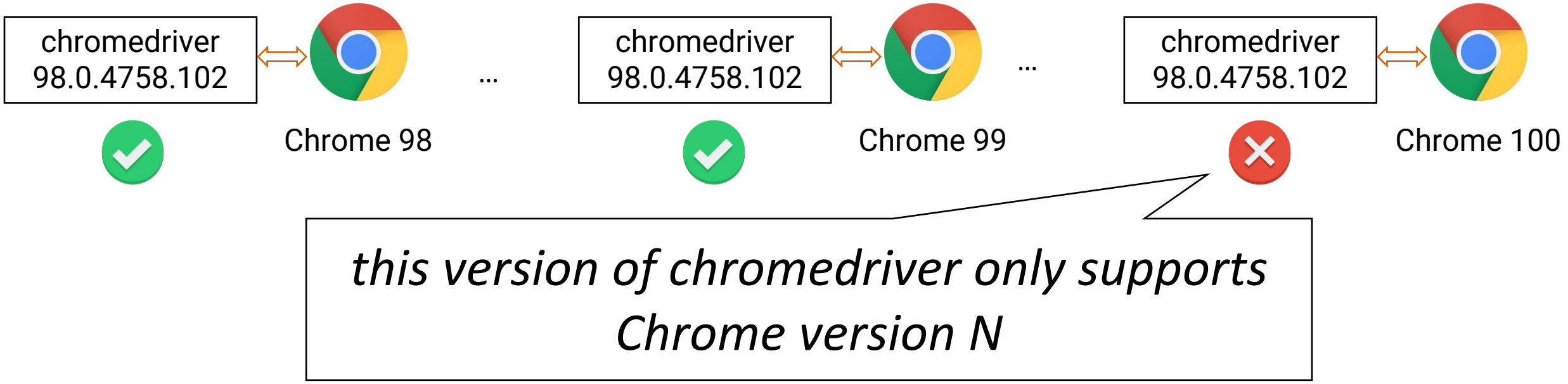
2. The original motivation: automated driver management

- Regarding the **setup**:
  - Once the driver is downloaded, to be used by Selenium WebDriver, it needs to be available in the PATH environment variable
  - Alternatively, the driver path needs to be exported using a given property before creating a `WebDriver` object

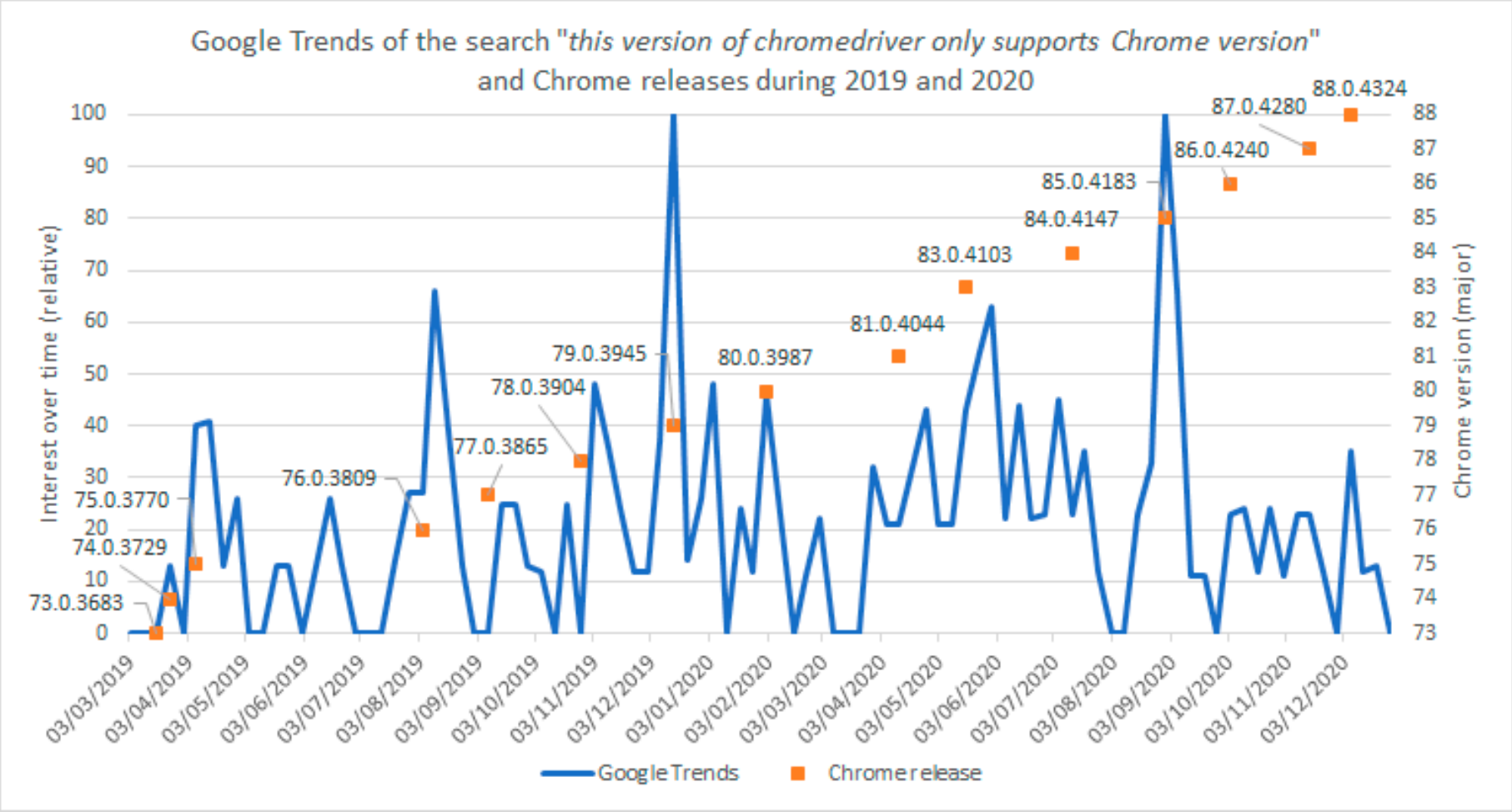
```
System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
System.setProperty("webdriver.gecko.driver", "/path/to/geckodriver");
System.setProperty("webdriver.edge.driver", "/path/to/msedgedriver");
System.setProperty("webdriver.opera.driver", "/path/to/operadriver");
System.setProperty("webdriver.ie.driver", "C:/path/to/IEDriverServer.exe");
```

2. The original motivation: automated driver management

- Regarding the **maintenance**:
  - Modern web browsers are *evergreen* (i.e., they automatically and silently upgrade to the next stable version)
  - Due to this upgrade, the driver-browser compatibility is not satisfied in the long run



2. The original motivation: automated driver management



2. The original motivation: automated driver management

- **WebDriverManager** is an open-source Java library that carries out the management (i.e., download, setup, and maintenance) of the drivers required by Selenium WebDriver (e.g., chromedriver, geckodriver, msedgedriver, etc.) in a fully automated manner.

```xml
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>5.2.2</version>
    <scope>test</scope>
</dependency>
```

Maven™

```
dependencies {
    testImplementation("io.github.bonigarcia:webdrivermanager:5.2.2")
}
```

```
WebDriverManager.chromedriver().setup();
WebDriverManager.firefoxdriver().setup();
WebDriverManager.edgedriver().setup();
WebDriverManager.chromiumdriver().setup();
WebDriverManager.operadriver().setup();
WebDriverManager.iedriver().setup();
```

2. The original motivation: automated driver management

```java
class ChromeTest {

    WebDriver driver;

    @BeforeAll
    static void setupClass() {
        WebDriverManager.chromedriver().setup();
    }

    @BeforeEach
    void setup() {
        driver = new ChromeDriver();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    public void test() {
        // Your test logic here
    }

}
```


```java
public class FirefoxTest {

    private WebDriver driver;

    @BeforeClass
    public static void setupClass() {
        WebDriverManager.firefoxdriver().setup();
    }

    @BeforeMethod
    public void setup() {
        driver = new FirefoxDriver();
    }

    @AfterMethod
    public void teardown() {
        driver.quit();
    }

    @Test
    public void test() {
        // Your test logic here
    }

}
```

2. The original motivation: automated driver management

Selenium-Jupiter

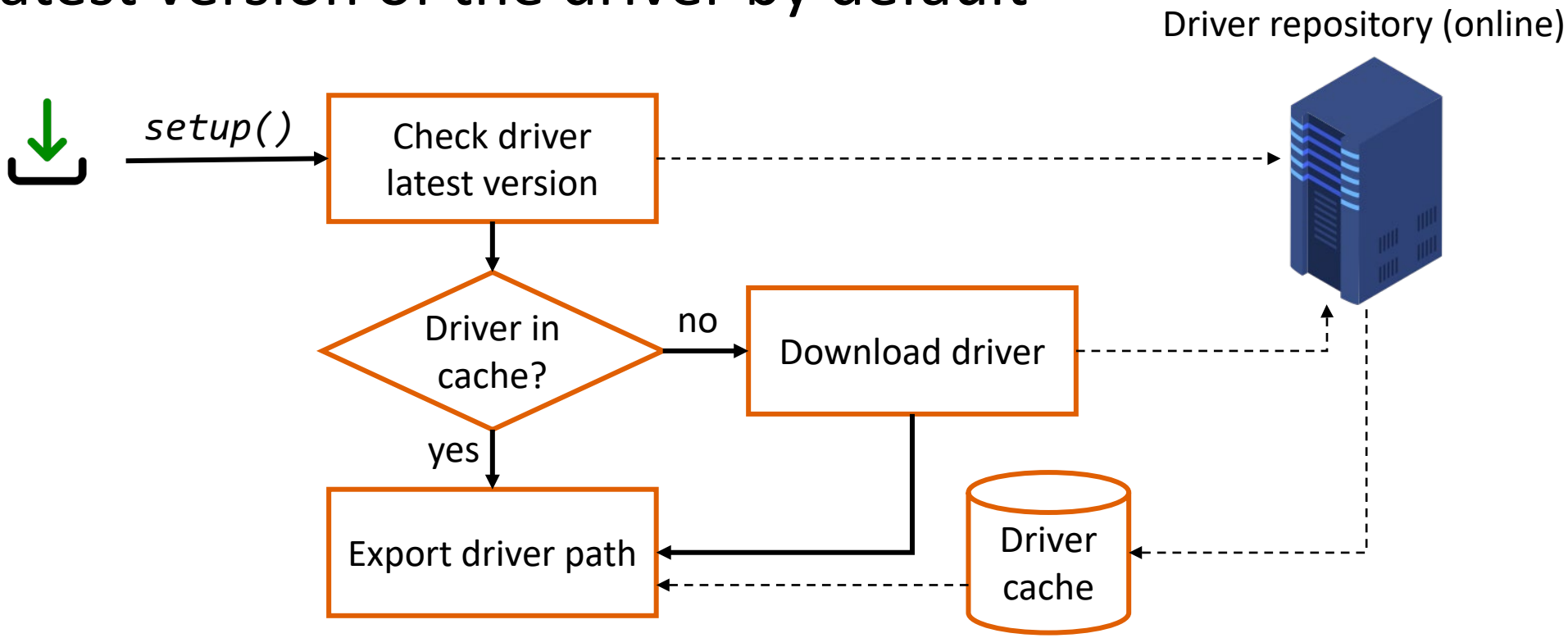
> *JUnit 5 extension for Selenium WebDriver*


```java
@ExtendWith(SeleniumJupiter.class)
class HelloWorldChromeSelJupTest {

    @Test
    void test(ChromeDriver driver) {
        // Your test logic here
    }

}
```

Source code: https://github.com/bonigarcia/selenium-jupiter
Doc: https://bonigarcia.dev/selenium-jupiter/

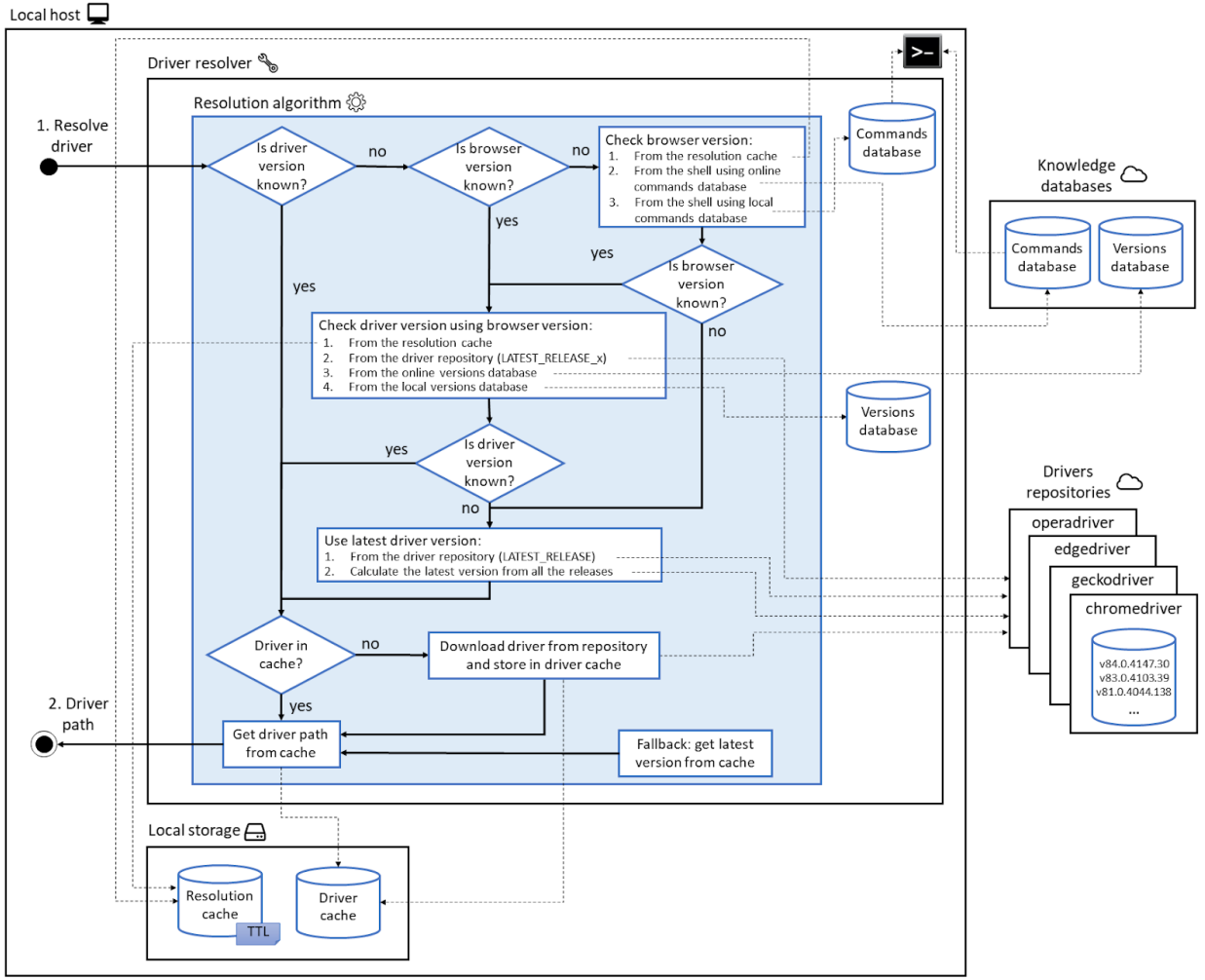
2. The original motivation: automated driver management

- WebDriverManager was first released on 21st March 2015
- In its earlier versions, WebDriverManager downloaded the latest version of the driver by default



2. The original motivation: automated driver management

- Currently, WebDriverManager *resolution algorithm* is much richer



García, Boni, et al. "Automated driver management for Selenium WebDriver." *Empirical Software Engineering* (2021)

2. The original motivation: automated driver management

- WebDriverManager exposes a rich fluent API, e.g.:

```
WebDriverManager.chromedriver().browserVersion("99").setup();
```
Force a chromedriver version compatible with Chrome 99

```
WebDriverManager.firefoxdriver().arch32().setup();
```
Force 32-bit architecture for geckodriver

```
WebDriverManager.operadriver().forceDownload().setup();
```
Force the download of operadriver (even if it is in the cached)

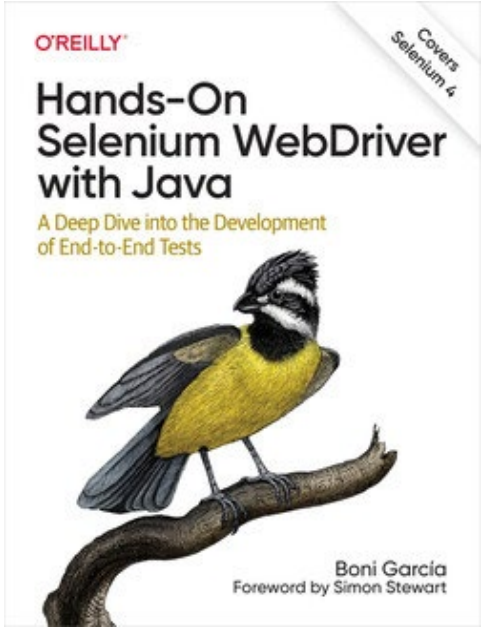
```
WebDriverManager.edgedriver().proxy("server:port").setup();
```
Set proxy setup when managing msedgedriver

- WebDriverManager is highly configurable with:
  1. Java properties, e.g.: `mvn test -Dwdm.cachePath=~/.selenium`
  2. Environment variables, e.g.: `export WDM_CACHEPATH=~/.selenium`

https://bonigarcia.dev/webdrivermanager/

3. The evolution: WebDriverManager 5

- WebDriverManager 5 was released on August 30, 2021
  - As of this version, WebDriverManager provides other features than automated driver management for Selenium WebDriver



O'Reilly will give three e-books as a giveaway to the participants in the following survey about the challenges and solutions of Selenium WebDriver: https://forms.gle/QzsSpJD5R2bGx2ZZ7

Source code: https://github.com/bonigarcia/selenium-webdriver-java
Practice site: https://bonigarcia.dev/selenium-webdriver-java/

3. The evolution: WebDriverManager 5

1. Browser finder

WebDriverManager provides the method `getBrowserPath()` to find out if the browser is installed or not

```
class SafariTest {

    WebDriver driver;

    @BeforeAll
    static void setupClass() {
        Optional<Path> browserPath = WebDriverManager.safaridriver()
                .getBrowserPath();
        assumeThat(browserPath).isPresent();
    }

    @BeforeEach
    void setupTest() {
        driver = new SafariDriver();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

3. The evolution: WebDriverManager 5

2. WebDriver builder

WebDriverManager provides the method `create()` to instantiate `WebDriver` objects (e.g., `ChromeDriver`, `FirefoxDriver`, etc.)

```
class ChromeCreateTest {

    WebDriver driver;

    @BeforeEach
    void setupTest() {
        driver = WebDriverManager.chromedriver().create();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

3. The evolution: WebDriverManager 5

3. Browsers in Docker

WebDriverManager provides the method `browserInDocker()` to control browsers in Docker containers

https://aerokube.com/images/latest/

```
class DockerChromeTest {

    WebDriver driver;

    WebDriverManager wdm = WebDriverManager.chromedriver().browserInDocker();

    @BeforeEach
    void setupTest() {
        driver = wdm.create();
    }

    @AfterEach
    void teardown() {
        wdm.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

3. The evolution: WebDriverManager 5

3. Browsers in Docker

The method `enableVnc()` allows to connect to the remote desktop using VNC and noVNC

The method `enableRecording()` allows recoding the remote session using FFmpeg


```java
class DockerChromeVncTest {

    WebDriver driver;

    WebDriverManager wdm = WebDriverManager.chromedriver().browserInDocker()
            .enableVnc();

    @BeforeEach
    void setupTest() {
        driver = wdm.create();
    }

    @AfterEach
    void teardown() {
        wdm.quit();
    }

    @Test
    void test() throws Exception {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

3. The evolution: WebDriverManager 5

3. Browsers in Docker

The method `browserVersion()` allows to change the version of the *dockerized* browser. This version can be fixed (e.g., `"100"`), or the following wildcards: `"latest"` or `"latest-N"`. Also: `"beta"` and `"dev"` (for Chrome and Firefox)

WebDriverManager provides the static method *`isDockerAvailable()`* to check if there is a Docker engine installed on the local machine

```
class DockerChromeBetaTest {

    WebDriver driver;

    WebDriverManager wdm = WebDriverManager.chromedriver().browserInDocker()
            .browserVersion("beta");

    @BeforeEach
    void setupTest() {
        assumeThat(isDockerAvailable()).isTrue();
        driver = wdm.create();
    }

    @AfterEach
    void teardown() {
        wdm.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

3. The evolution: WebDriverManager 5

4. Monitoring
   - WebDriverManager 5.2.0 provides seamless integration with BrowserWatcher

> *Browser extension for console monitoring, tab recording, Content Security Policy (CSP) disabling, and JavaScript/CSS injection*

Source code: https://github.com/bonigarcia/browserwatcher
Doc: https://bonigarcia.dev/browserwatcher/

3. The evolution: WebDriverManager 5

4. Monitoring

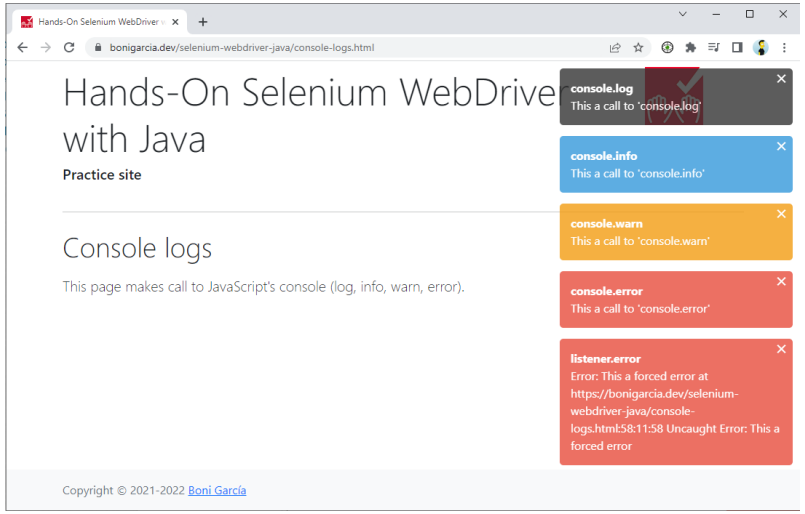
The method `watch()` allows to install BrowserWatcher in the browser controlled with Selenium WebDriver. Then, it allows to gather the browser logs (i.e., its console) invoking the method `getLogs()`

```
class GatherLogsFirefoxTest {

    static final Logger log = getLogger(lookup().lookupClass());

    WebDriverManager wdm = WebDriverManager.firefoxdriver().watch();
    WebDriver driver;

    @BeforeEach
    void setup() {
        driver = wdm.create();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/console-logs.html");
        List<Map<String, Object>> logMessages = wdm.getLogs();

        for (Map<String, Object> map : logMessages) {
            log.debug("[{}] [{}] {}", map.get("datetime"),
                    String.format("%1$-14s",
                            map.get("source").toString().toUpperCase() + "."
                                    + map.get("type").toString().toUpperCase()),
                    map.get("message"));
        }        assertThat(logMessages).hasSize(5);
    }

}
```

3. The evolution: WebDriverManager 5

4. Monitoring

The method `watchAndDisplay()` allows displaying the console logs as dialog notifications on the page




```
class DisplayLogsChromeTest {

    static final Logger log = getLogger(lookup().lookupClass());

    WebDriverManager wdm = WebDriverManager.chromedriver().watchAndDisplay();
    WebDriver driver;

    @BeforeEach
    void setup() {
        driver = wdm.create();
    }

    @AfterEach
    void teardown() throws InterruptedException {
        // pause for manual browser inspection
        Thread.sleep(Duration.ofSeconds(3).toMillis());

        driver.quit();
    }

    @Test
    void test() {
         // test logic
    }

}
```

3. The evolution: WebDriverManager 5

4. Monitoring

The methods `startRecording()` and `stopRecording()` allows record the browser viewport

```
class RecordEdgeTest {

    WebDriver driver;
    WebDriverManager wdm = WebDriverManager.edgedriver().watch();

    @BeforeEach
    void setup() {
        driver = wdm.create();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() throws InterruptedException {
        driver.get(
                "https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html");

        wdm.startRecording(REC_FILENAME);

        // test logic

        wdm.stopRecording();
    }

}
```

4. Beyond Java

- WebDriverManager can also be used as a:

1. CLI (command line interface) tool:
   - Using the WebDriverManager fat-JAR:

```
java -jar webdrivermanager-5.2.2-fat.jar resolveDriverFor chrome
```

```
java -jar webdrivermanager-5.2.2-fat.jar runInDocker chrome
```

The option `resolveDriverFor` allows to resolve drivers from the shell

The option `runInDocker` allows to execute browsers in Docker containers and interact with them using noVNC

   - Using its source code and Maven:

```
mvn exec:java -Dexec.args="resolveDriverFor chrome"
```

```
mvn exec:java -Dexec.args="runInDocker chrome"
```

   - Using its Docker image:

```
docker run --rm -v ${PWD}:/wdm -e ARGS="resolveDriverFor chrome" bonigarcia/webdrivermanager:5.2.2
```

```
docker run --rm -it -v /var/run/docker.sock:/var/run/docker.sock -e ARGS="runInDocker chrome"
bonigarcia/webdrivermanager:5.2.2
```

4. Beyond Java

- WebDriverManager can also be used as a:

2. Server:
   - Using the WebDriverManager fat-JAR:
   - Using its source code and Maven:
   - Using its Docker image:

```
java -jar webdrivermanager-5.2.2-fat.jar server
```

```
mvn exec:java -Dexec.args="server"
```

```
docker run --rm -p 4444:4444 -v
/var/run/docker.sock:/var/run/docker.sock
bonigarcia/webdrivermanager:5.2.2
```

This server provides two features:

i. To resolve drivers using a REST-like API:

http://localhost:4444/chromedriver http://localhost:4444/chromedriver?chromeVersion=100

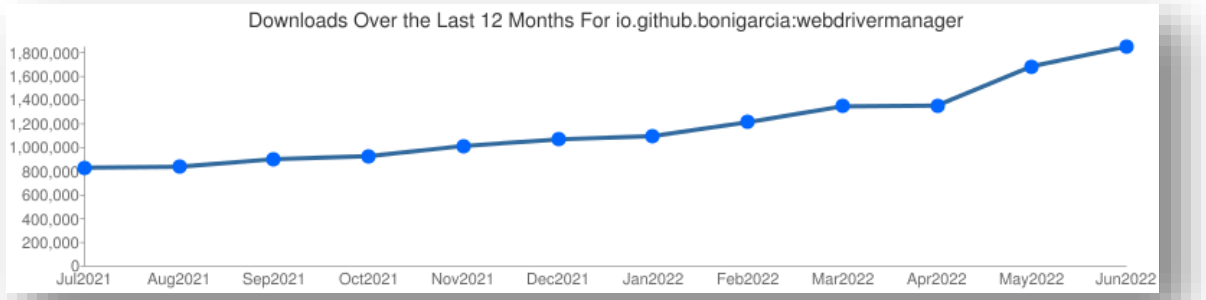
ii. As a Selenium Server (that uses Docker to manage the browser infrastructure) http://localhost:4444/

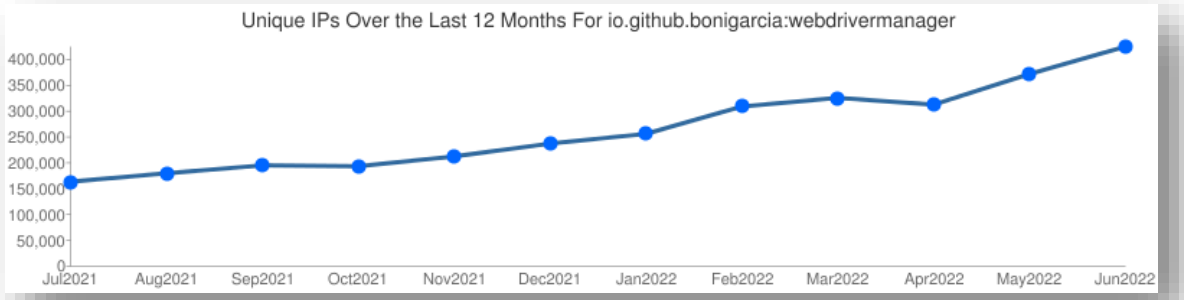
5. Limitations of WebDriverManager

- WebDriverManager is mainly used from Java
  - Although it has other usages (CLI, server), its main use is together the Selenium WebDriver Java language binding
- Mobile Chrome (Android) in Docker requires hardware virtualization
- Docker browsers are not yet available on macOS ARM
- Monitoring capabilities through BrowserWatcher are not available on headless browsers
  - For log gathering, a fallback based on `LoggingPreferences` for Chrome is implemented (WebDriver BiDi capabilities is pending to be implemented)
- Tab recording through BrowserWatcher only works in Chrome/Edge, since this feature is based on the *chrome.tabCapture* API

6. Summary and outlook

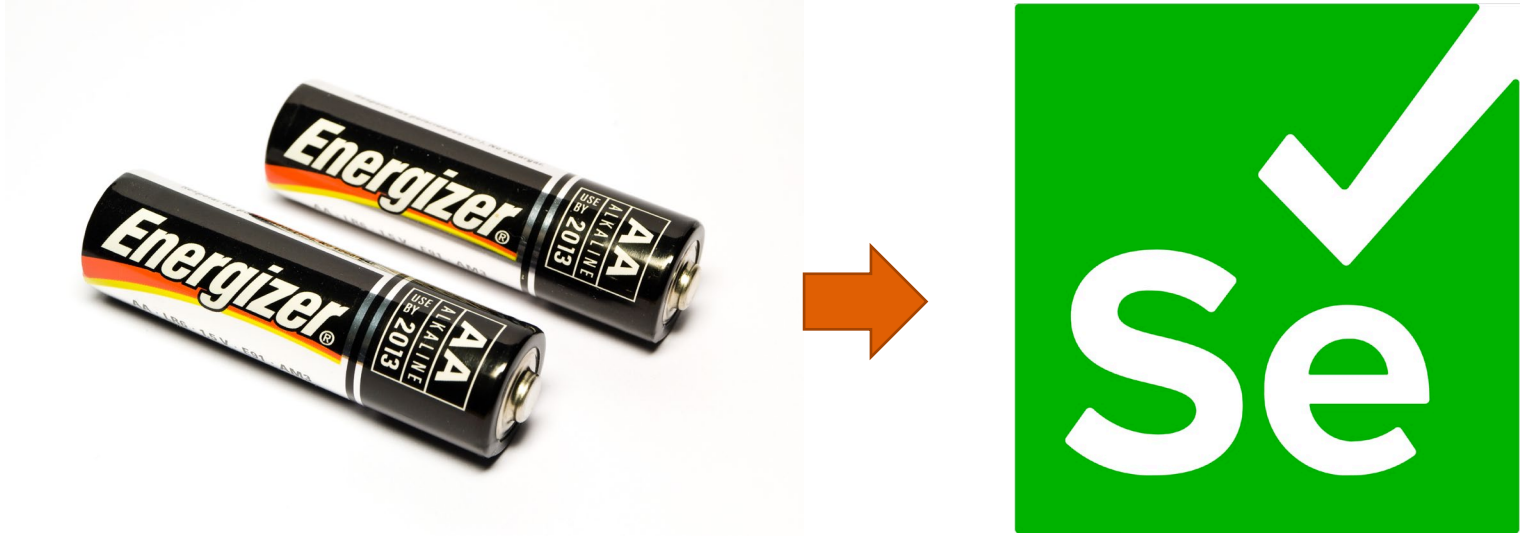
- WebDriverManager is a helper tool for Selenium in Java
  - It was born to carry out automated management (i.e., driver, setup, and maintenance) of the drivers required by Selenium WebDriver
  - As of version 5, it provides other features: browser finder, `WebDriver` builder, browsers in Docker, and monitoring through BrowserWatcher





6. Summary and outlook

- Currently, I am working in the proposal for the implementation something similar to WebDriverManager, but as an official component of the Selenium project: **Selenium Manager** (a.k.a. batteries included)



WebDriverManager: the Swiss Army Knife for Selenium WebDriver

Thank you very much!
Q&A

Boni García

 boni.garcia.c@saucelabs.com  https://bonigarcia.dev

 @boni_gg  https://github.com/bonigarcia