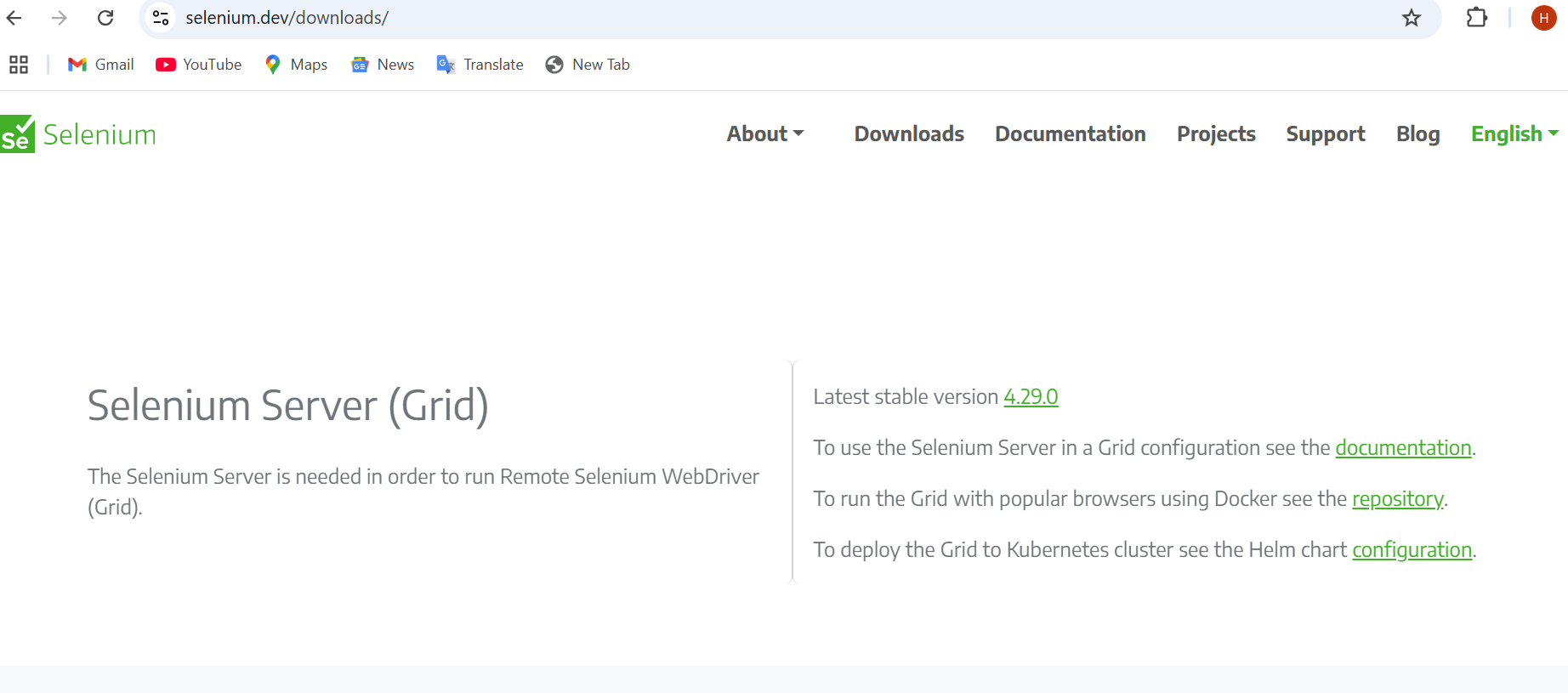
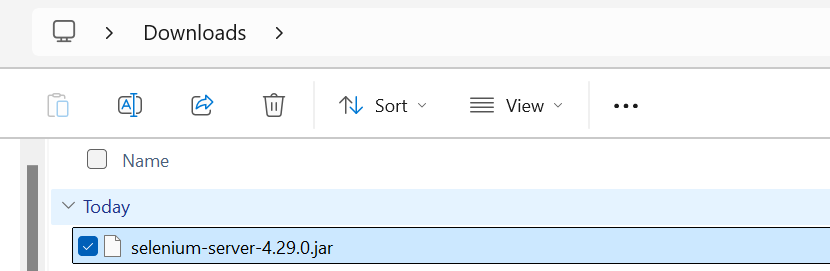
**Selenium Grid Setup**

**Step 1: Download Selenium Grid**

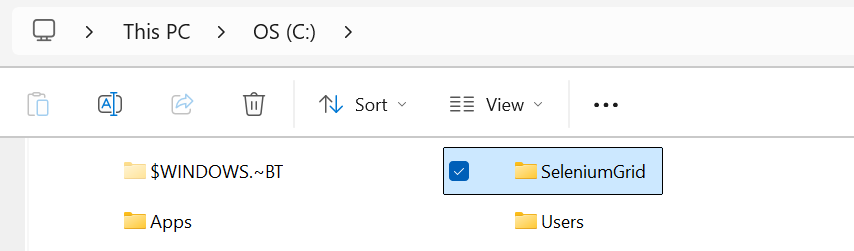
1. **Go to Selenium Grid official website**: <https://www.selenium.dev/downloads/>

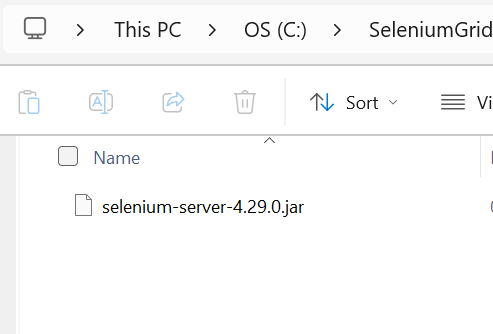


1. **Download Selenium Server JAR file** (usually named selenium-server-<version>.jar).



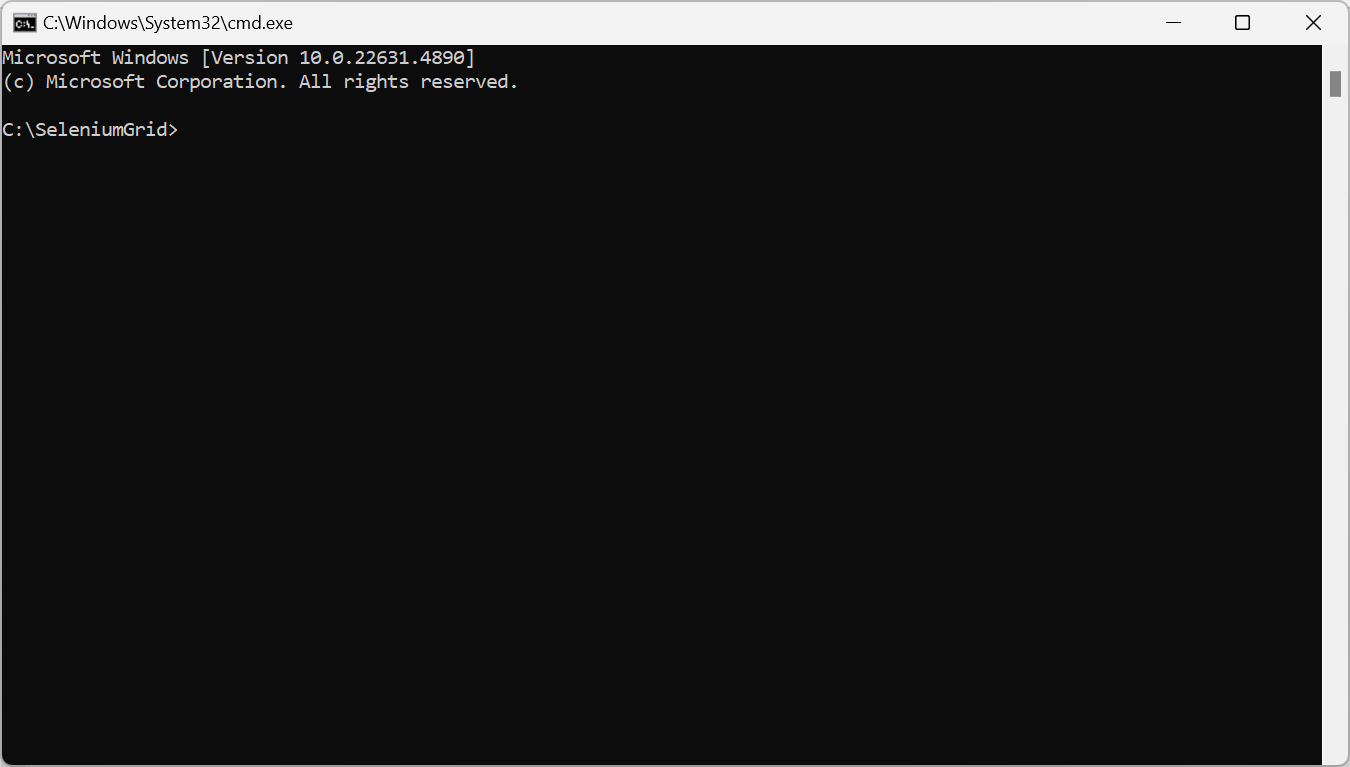
1. **Place it in your local system** (preferably C:/SeleniumGrid/).



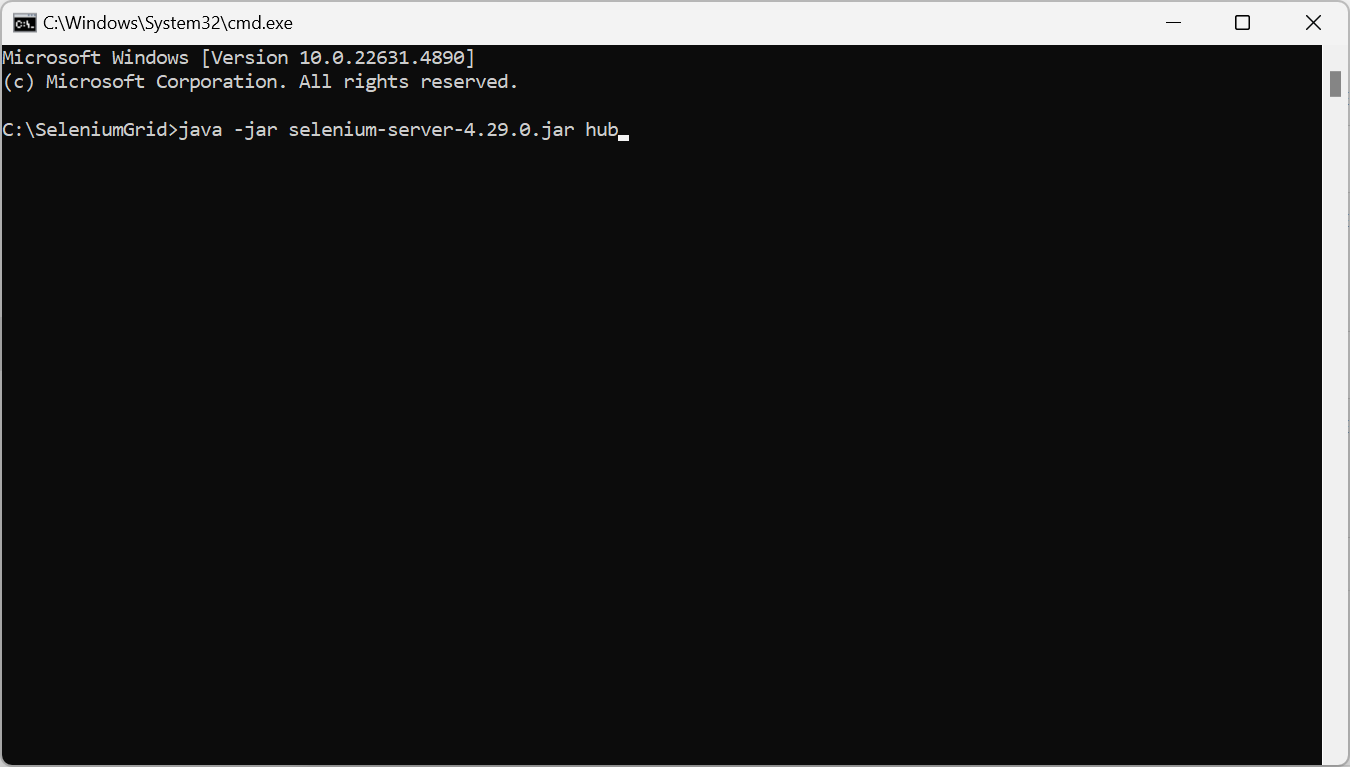


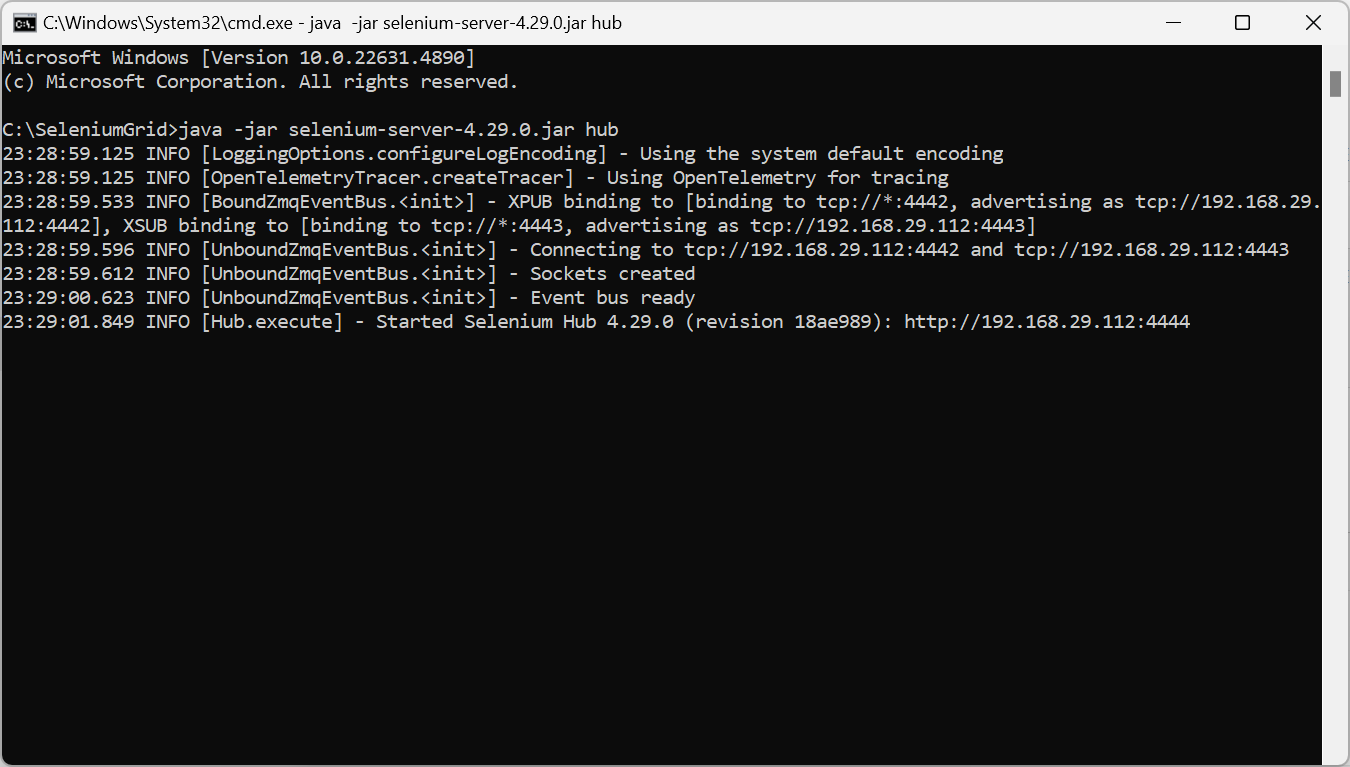
**Step 2: Start the Hub (Central Controller)**

Open **Command Prompt (CMD)** and navigate to your Selenium Grid folder:

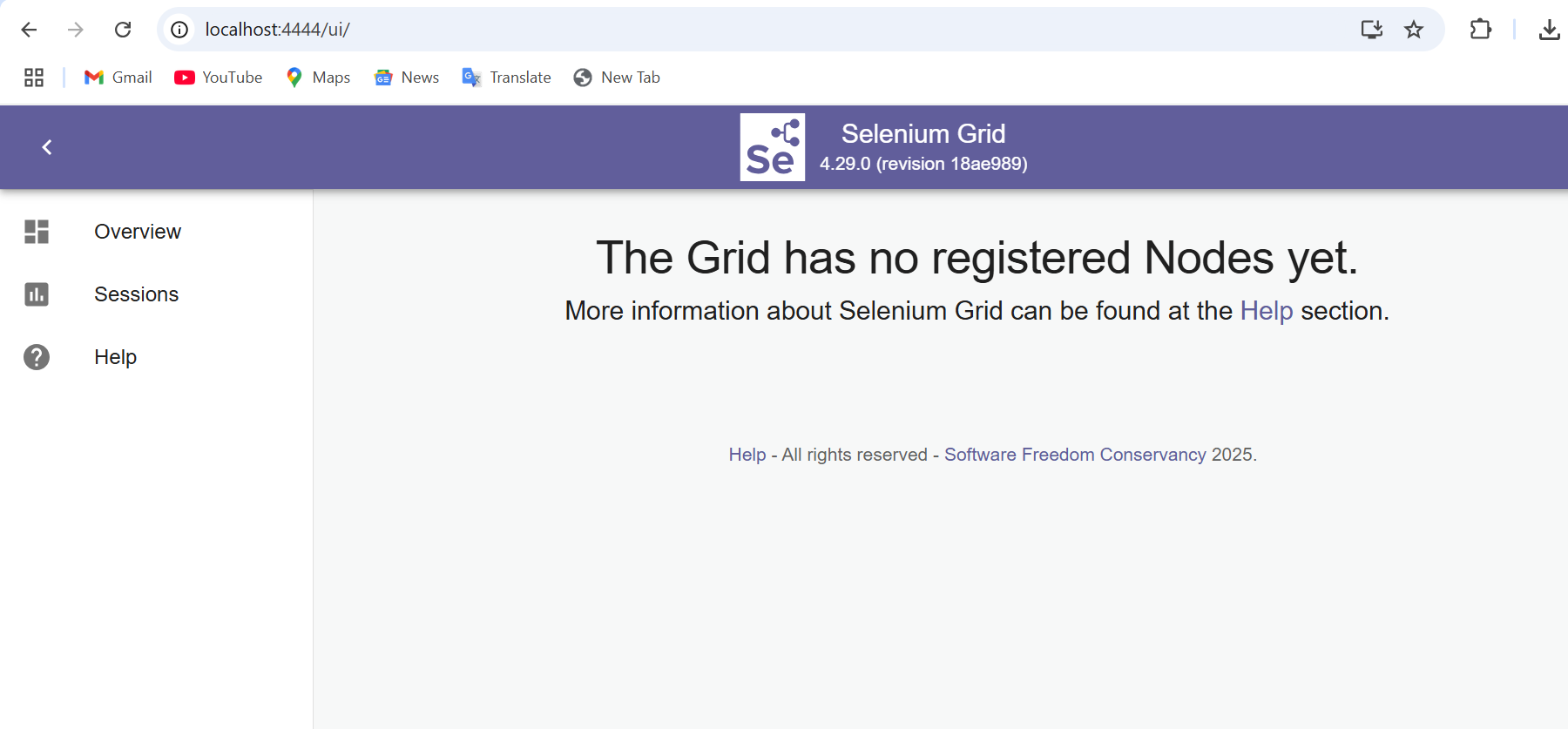


**Command** 🡪 java -jar selenium-server-4.29.0.jar hub





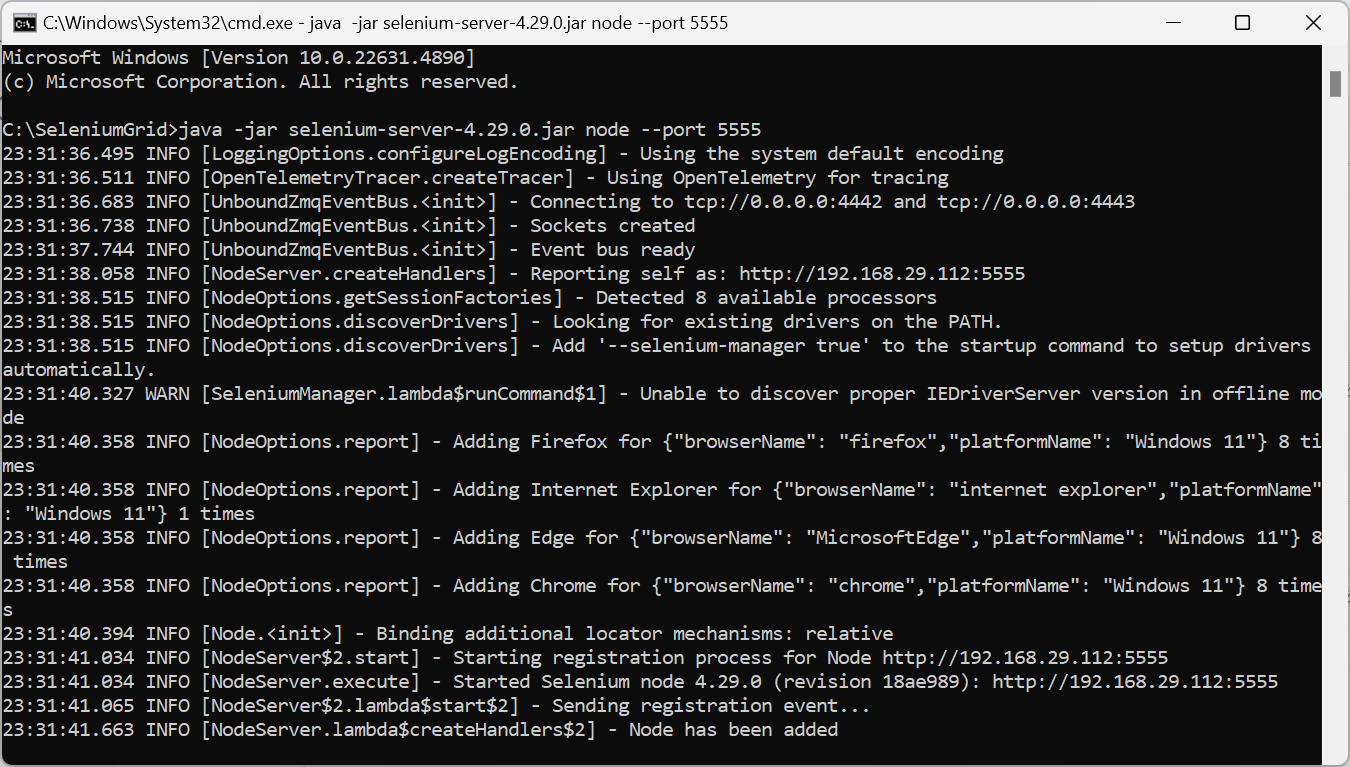
**Hub URL:** <http://localhost:4444>



**Step 3: Start the Node (Test Machine)**

Open **another CMD** window and run the following command:

Command 🡪 java -jar selenium-server-4.29.0.jar node --port 5555

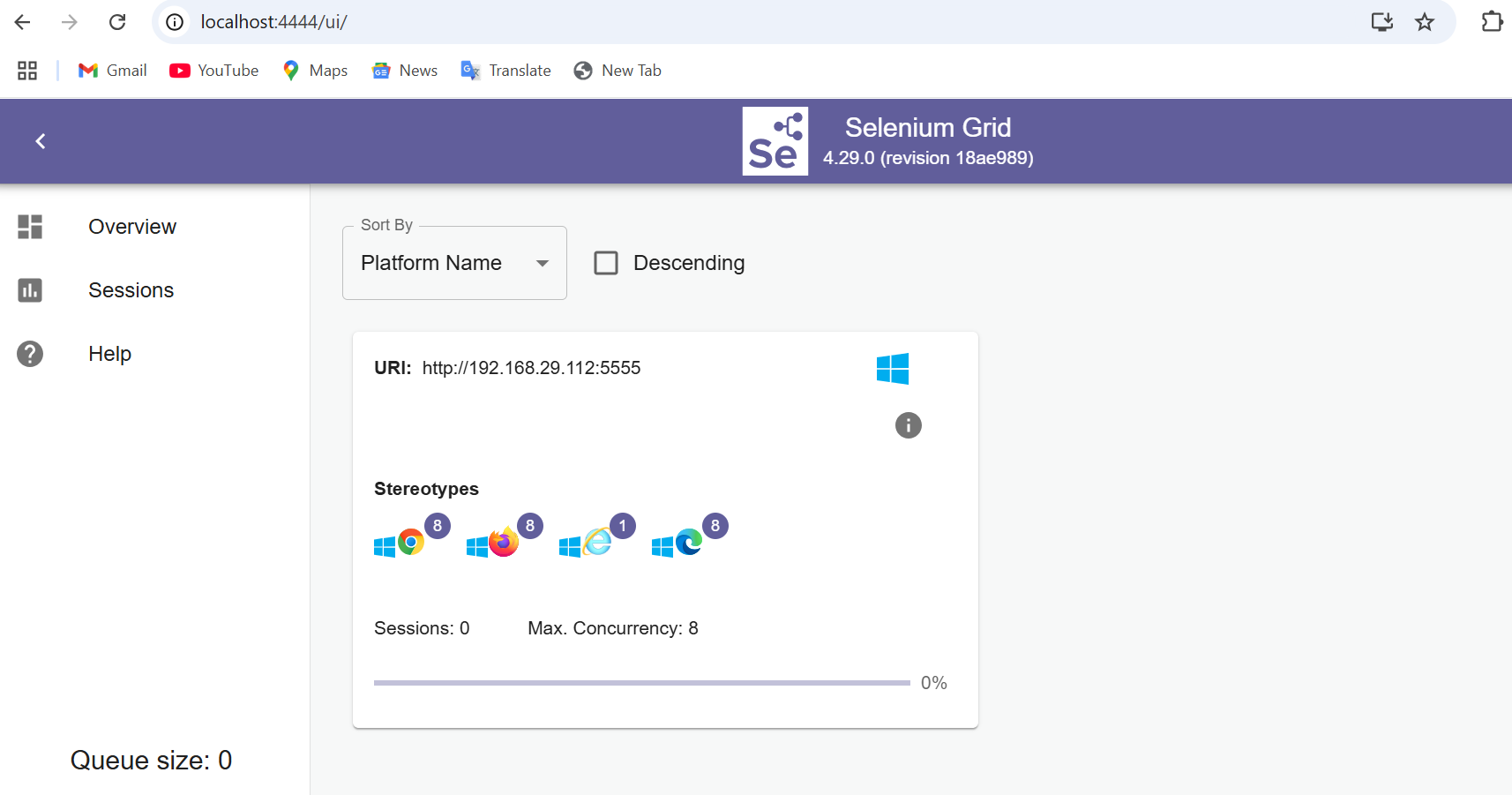


**Step 4: Verify Node Registration**

Open your browser and go to: <http://localhost:4444>

You will see your Node is registered and ready for execution.

Node is registered to the Hub at <http://localhost:4444/grid/register>



**Step 5: Modify Your Framework Code**

Now modify your **BaseClass** in the framework to use **RemoteWebDriver** instead of **WebDriver**.

**BaseClass.java**

**src/main/java/com/yourprojectname/base/BaseClass.java**

1. **package** com.orangehrm.base;
3. **import** java.io.FileInputStream;
4. **import** java.io.IOException;
5. **import** java.net.MalformedURLException;
6. **import** java.net.URL;
7. **import** java.time.Duration;
8. **import** java.util.Properties;
9. **import** java.util.concurrent.TimeUnit;
10. **import** java.util.concurrent.locks.LockSupport;
12. **import** org.apache.logging.log4j.Logger;
13. **import** org.openqa.selenium.WebDriver;
14. **import** org.openqa.selenium.chrome.ChromeDriver;
15. **import** org.openqa.selenium.chrome.ChromeOptions;
16. **import** org.openqa.selenium.edge.EdgeDriver;
17. **import** org.openqa.selenium.edge.EdgeOptions;
18. **import** org.openqa.selenium.firefox.FirefoxDriver;
19. **import** org.openqa.selenium.firefox.FirefoxOptions;
20. **import** org.openqa.selenium.remote.DesiredCapabilities;
21. **import** org.openqa.selenium.remote.RemoteWebDriver;
22. **import** org.testng.annotations.AfterMethod;
23. **import** org.testng.annotations.BeforeMethod;
24. **import** org.testng.annotations.BeforeSuite;
25. **import** org.testng.annotations.Parameters;
26. **import** org.testng.asserts.SoftAssert;
28. **import** com.orangehrm.actiondriver.ActionDriver;
29. **import** com.orangehrm.utilities.ExtentManager;
30. **import** com.orangehrm.utilities.LoggerManager;
32. **public** **class** BaseClass {
34. **protected** **static** Properties prop;
35. // protected static WebDriver driver;
36. // private static ActionDriver actionDriver;
38. **private** **static** ThreadLocal<WebDriver> driver = **new** ThreadLocal<>();
39. **private** **static** ThreadLocal<ActionDriver> actionDriver = **new** ThreadLocal<>();
40. **public** **static** **final** Logger logger = LoggerManager.getLogger(BaseClass.**class**);
42. **protected** ThreadLocal<SoftAssert> softAssert = ThreadLocal.withInitial(SoftAssert::**new**);
44. // Getter method for soft assert
45. **public** SoftAssert getSoftAssert() {
46. **return** softAssert.get();
47. }
49. @BeforeSuite
50. **public** **void** loadConfig() **throws** IOException {
51. // Load the configuration file
52. prop = **new** Properties();
53. FileInputStream fis = **new** FileInputStream(
54. System.getProperty("user.dir") + "/src/main/resources/config.properties");
55. prop.load(fis);
56. logger.info("config.properties file loaded");
58. // Start the Extent Report
59. // ExtentManager.getReporter(); --This has been implemented in TestListener
60. }
62. @BeforeMethod
63. @Parameters("browser")
64. **public** **synchronized** **void** setup(String browser) **throws** IOException {
65. System.out.println("Setting up WebDriver for:" + **this**.getClass().getSimpleName());
66. launchBrowser(browser);
67. configureBrowser();
68. staticWait(2);
69. // Sample logger message
70. logger.info("WebDriver Initialized and Browser Maximized");
71. logger.trace("This is a Trace message");
72. logger.error("This is a error message");
73. logger.debug("This is a debug message");
74. logger.fatal("This is a fatal message");
75. logger.warn("This is a warm message");
77. /\*
78. \* // Initialize the actionDriver only once if (actionDriver == null) {
79. \* actionDriver = new ActionDriver(driver);
80. \* logger.info("ActionDriver instance is created. "+Thread.currentThread().getId
81. \* ()); }
82. \*/
84. // Initialize ActionDriver for the current Thread
85. actionDriver.set(**new** ActionDriver(getDriver()));
86. logger.info("ActionDriver initlialized for thread: " + Thread.currentThread().getId());
88. }
90. /\*
91. \* Initialize the WebDriver based on browser defined in config.properties file
92. \*/
93. **private** **synchronized** **void** launchBrowser(String browser) {
95. //String browser = prop.getProperty("browser");
97. **boolean** seleniumGrid = Boolean.parseBoolean(prop.getProperty("seleniumGrid"));
98. String gridURL = prop.getProperty("gridURL");
100. **if** (seleniumGrid) {
101. **try** {
102. **if** (browser.equalsIgnoreCase("chrome")) {
103. ChromeOptions options = **new** ChromeOptions();
104. options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1080");
105. driver.set(**new** RemoteWebDriver(**new** URL(gridURL), options));
106. } **else** **if** (browser.equalsIgnoreCase("firefox")) {
107. FirefoxOptions options = **new** FirefoxOptions();
108. options.addArguments("-headless");
109. driver.set(**new** RemoteWebDriver(**new** URL(gridURL), options));
110. } **else** **if** (browser.equalsIgnoreCase("edge")) {
111. EdgeOptions options = **new** EdgeOptions();
112. options.addArguments("--headless=new", "--disable-gpu","--no-sandbox","--disable-dev-shm-usage");
113. driver.set(**new** RemoteWebDriver(**new** URL(gridURL), options));
114. } **else** {
115. **throw** **new** IllegalArgumentException("Browser Not Supported: " + browser);
116. }
117. logger.info("RemoteWebDriver instance created for Grid in headless mode");
118. } **catch** (MalformedURLException e) {
119. **throw** **new** RuntimeException("Invalid Grid URL", e);
120. }
121. } **else** {
123. **if** (browser.equalsIgnoreCase("chrome")) {
125. // Create ChromeOptions
126. ChromeOptions options = **new** ChromeOptions();
127. options.addArguments("--headless"); // Run Chrome in headless mode
128. options.addArguments("--disable-gpu"); // Disable GPU for headless mode
129. //options.addArguments("--window-size=1920,1080"); // Set window size
130. options.addArguments("--disable-notifications"); // Disable browser notifications
131. options.addArguments("--no-sandbox"); // Required for some CI environments like Jenkins
132. options.addArguments("--disable-dev-shm-usage"); // Resolve issues in resource-limited environments
134. // driver = new ChromeDriver();
135. driver.set(**new** ChromeDriver(options)); // New Changes as per Thread
136. ExtentManager.registerDriver(getDriver());
137. logger.info("ChromeDriver Instance is created.");
138. } **else** **if** (browser.equalsIgnoreCase("firefox")) {
140. // Create FirefoxOptions
141. FirefoxOptions options = **new** FirefoxOptions();
142. options.addArguments("--headless"); // Run Firefox in headless mode
143. options.addArguments("--disable-gpu"); // Disable GPU rendering (useful for headless mode)
144. options.addArguments("--width=1920"); // Set browser width
145. options.addArguments("--height=1080"); // Set browser height
146. options.addArguments("--disable-notifications"); // Disable browser notifications
147. options.addArguments("--no-sandbox"); // Needed for CI/CD environments
148. options.addArguments("--disable-dev-shm-usage"); // Prevent crashes in low-resource environments
150. // driver = new FirefoxDriver();
151. driver.set(**new** FirefoxDriver(options)); // New Changes as per Thread
152. ExtentManager.registerDriver(getDriver());
153. logger.info("FirefoxDriver Instance is created.");
154. } **else** **if** (browser.equalsIgnoreCase("edge")) {
156. EdgeOptions options = **new** EdgeOptions();
157. options.addArguments("--headless"); // Run Edge in headless mode
158. options.addArguments("--disable-gpu"); // Disable GPU acceleration
159. options.addArguments("--window-size=1920,1080"); // Set window size
160. options.addArguments("--disable-notifications"); // Disable pop-up notifications
161. options.addArguments("--no-sandbox"); // Needed for CI/CD
162. options.addArguments("--disable-dev-shm-usage"); // Prevent resource-limited crashes
164. // driver = new EdgeDriver();
165. driver.set(**new** EdgeDriver(options)); // New Changes as per Thread
166. ExtentManager.registerDriver(getDriver());
167. logger.info("EdgeDriver Instance is created.");
168. } **else** {
169. **throw** **new** IllegalArgumentException("Browser Not Supported:" + browser);
170. }
171. }
172. }
174. /\*
175. \* Configure browser settings such as implicit wait, maximize the browser and
176. \* navigate to the URL
177. \*/
179. **private** **void** configureBrowser() {
180. // Implicit Wait
181. **int** implicitWait = Integer.parseInt(prop.getProperty("implicitWait"));
182. getDriver().manage().timeouts().implicitlyWait(Duration.ofSeconds(implicitWait));
184. // maximize the browser
185. getDriver().manage().window().maximize();
187. // Navigate to URL
188. **try** {
189. getDriver().get(prop.getProperty("url"));
190. } **catch** (Exception e) {
191. System.out.println("Failed to Navigate to the URL:" + e.getMessage());
192. }
193. }
195. @AfterMethod
196. **public** **synchronized** **void** tearDown() {
197. **if** (getDriver() != **null**) {
198. **try** {
199. getDriver().quit();
200. } **catch** (Exception e) {
201. System.out.println("unable to quit the driver:" + e.getMessage());
202. }
203. }
204. logger.info("WebDriver instance is closed.");
205. driver.remove();
206. actionDriver.remove();
207. // driver = null;
208. // actionDriver = null;
209. // ExtentManager.endTest(); --This has been implemented in TestListener
210. }
212. /\*
213. \*
214. \*
215. \* //Driver getter method public WebDriver getDriver() { return driver; }
216. \*/
218. // Getter Method for WebDriver
219. **public** **static** WebDriver getDriver() {
221. **if** (driver.get() == **null**) {
222. System.out.println("WebDriver is not initialized");
223. **throw** **new** IllegalStateException("WebDriver is not initialized");
224. }
225. **return** driver.get();
227. }
229. // Getter Method for ActionDriver
230. **public** **static** ActionDriver getActionDriver() {
232. **if** (actionDriver.get() == **null**) {
233. System.out.println("ActionDriver is not initialized");
234. **throw** **new** IllegalStateException("ActionDriver is not initialized");
235. }
236. **return** actionDriver.get();
238. }
240. // Getter method for prop
241. **public** **static** Properties getProp() {
242. **return** prop;
243. }
245. // Driver setter method
246. **public** **void** setDriver(ThreadLocal<WebDriver> driver) {
247. **this**.driver = driver;
248. }
250. // Static wait for pause
251. **public** **void** staticWait(**int** seconds) {
252. LockSupport.parkNanos(TimeUnit.SECONDS.toNanos(seconds));
253. }
255. }

**Config.proerties file**

1. seleniumGrid=**false**
2. gridURL=http://localhost:4444/wd/hub
4. #Application URL
5. url\_demo\_site = https://opensource-demo.orangehrmlive.com/web/index.php/auth/login
6. url = http://localhost/orangehrm/web/index.php/auth/login
8. #Browser Configuration
9. browser = chrome
11. #Credentials
12. username = orangehrm\_hverma
13. password = Passw0rd@7654321
15. #ImplicitWait (in seconds)
16. implicitWait =30
18. #ExplicitWait
19. explicitWait = 30

**testng.xml**

1. <!--?xml version="1.0" encoding="UTF-8"?-->
3. <suite name="OrangeHRMSuite" parallel="tests" thread-count="7">
4. <listeners>
5. <listener **class**-name="com.orangehrm.listeners.TestListener"></listener>
6. </listeners>
7. <test name="OrangeHRMTest\_chrome">
8. <parameter name="browser" value="chrome">
9. <classes>
10. <**class** name="com.orangehrm.test.DBVerificationTest">
11. <**class** name="com.orangehrm.test.ApiTest">
12. </**class**></**class**></classes>
13. </parameter></test>
14. <test name="OrangeHRMTest\_edge">
15. <parameter name="browser" value="edge">
16. <classes>
17. <**class** name="com.orangehrm.test.DummyClass">
18. <**class** name="com.orangehrm.test.DummyClass2">
19. </**class**></**class**></classes>
20. </parameter></test>
21. <test name="OrangeHRMTest\_firefox">
22. <parameter name="browser" value="firefox">
23. <classes>
24. <**class** name="com.orangehrm.test.LoginPageTest">
25. <**class** name="com.orangehrm.test.HomePageTest">
26. </**class**></**class**></classes>
27. </parameter></test>
28. </suite>

**Step 6: Run TestNG Suite in Parallel**

You will now see:

* + **One test running in Chrome**
  + **One test running in Edge**
  + **One test running in Firefox**

**All tests running parallel** via Selenium Grid!

**Step 7: View Results in Selenium Grid Dashboard**

Open your browser and go to:  
👉 http://localhost:4444

You will see:

* Active sessions running.
* Completed test cases.

**Run Tests on Another Machine**

If you want to run tests on a **different machine (Node)**:

1. Install Java and Selenium JAR on the second machine.
2. Start Node like this:

java -jar selenium-server-<version>.jar node --port 5555 --browser "chrome" --hub http://<HUB\_MACHINE\_IP>:4444

1. Replace the localhost URL in your **BaseClass.java** with the Node IP.