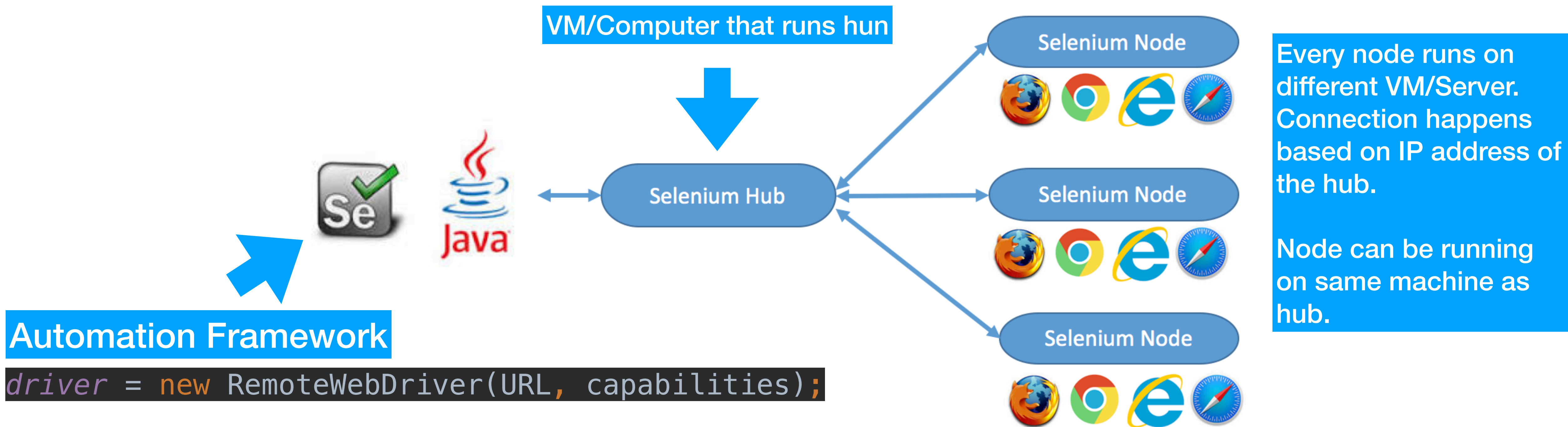


Selenium Grid

What is Selenium Grid?

The **Selenium Grid** is a testing tool which allows us to run our tests on different machines against different browsers. It is a part of the Selenium Suite which specialize in running multiple tests across different browsers, operating system and machines. You can connect to it with Selenium Remote by specifying the browser, browser version, and operating system you want. You specify these values through Selenium Remote's Capabilities. There are two main elements to **Selenium Grid** – a **hub** and **node**.



The HUB

- The hub is the central point where you load your tests into.
- There should only be one hub in a grid.
- The hub is launched only on a single machine, say, a computer whose O.S is Windows 10 and whose browser is Chrome.
- The machine containing the hub is where the tests will be run, but you will see the browser being automated on the node.
- The hub can be parametrized with json file.

The NODE

- Nodes are the Selenium instances that will execute the tests that you loaded on the hub.
- There can be one or more nodes in a grid.
- Nodes can be launched on multiple machines with different platforms and browsers.
- Nodes can be located on different machines.
- Nodes can be parametrized with json file.

How Selenium-Grid Works–With a Hub and Nodes

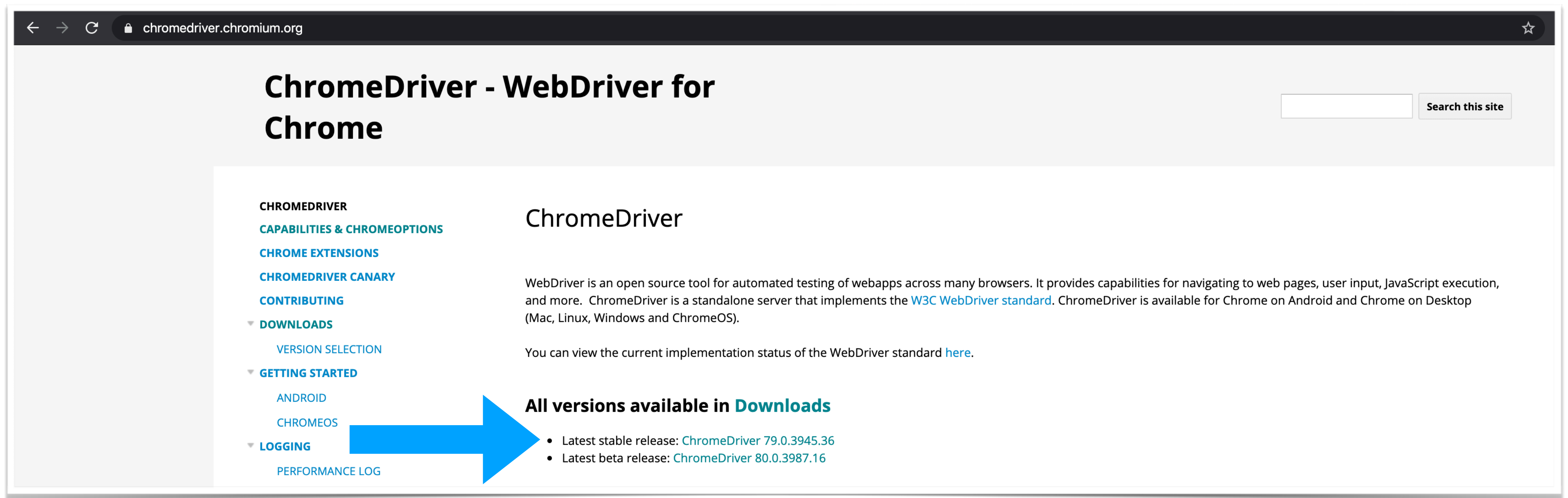
A grid consists of a single hub, and one or more nodes. Both are started using the selenium-server.jar executable.

The hub receives a test to be executed along with information on which browser and ‘platform’ (i.e. WINDOWS, LINUX, etc) where the test should be run. It ‘knows’ the configuration of each node that has been ‘registered’ to the hub. Using this information it selects an available node that has the requested browser-platform combination.

Once a node has been selected, Selenium commands initiated by the test are sent to the hub, which passes them to the node assigned to that test. The node runs the browser, and executes the Selenium commands within that browser against the application under test.

How to Set Up Selenium Grid

- Step 1: Go to <https://chromedriver.chromium.org/>
- Step 2: download chrome driver (stable



The screenshot shows the ChromeDriver website interface. The browser's address bar displays 'chromedriver.chromium.org'. The main heading is 'ChromeDriver - WebDriver for Chrome'. A search bar is located in the top right corner. On the left side, there is a navigation menu with links: CHROMEDRIVER, CAPABILITIES & CHROMEOPTIONS, CHROME EXTENSIONS, CHROMEDRIVER CANARY, CONTRIBUTING, DOWNLOADS (highlighted with a blue arrow), GETTING STARTED, and LOGGING. Below the 'DOWNLOADS' link, there are sub-links: VERSION SELECTION, ANDROID, CHROMEOS, and PERFORMANCE LOG. The main content area is titled 'ChromeDriver' and contains a paragraph explaining that WebDriver is an open source tool for automated testing of webapps across many browsers. It also mentions that ChromeDriver is a standalone server that implements the W3C WebDriver standard. Below this, there is a link to view the current implementation status of the WebDriver standard. At the bottom of the main content area, there is a section titled 'All versions available in Downloads' which lists the latest stable release (ChromeDriver 79.0.3945.36) and the latest beta release (ChromeDriver 80.0.3987.16).






How to Set Up Selenium Grid

- Step 3: Download version based on your OS

Linux/Ubuntu

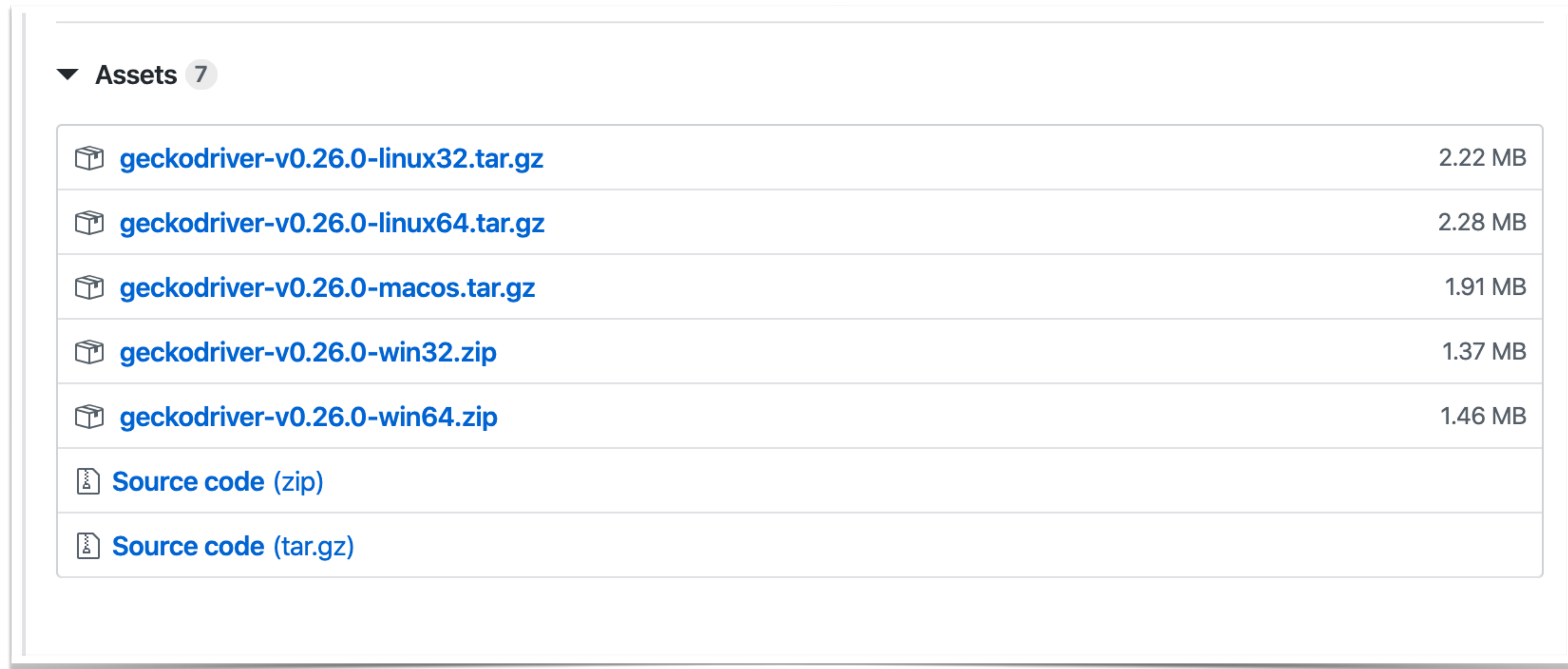
Mac

Windows







Index of /79.0.3945.36/				
	<u>Name</u>	Last modified	Size	ETag
	Parent Directory	-	-	-
	chromedriver linux64.zip	2019-11-18 18:20:03	4.65MB	77e6b631478c63c2df5809822a0af916
	chromedriver mac64.zip	2019-11-18 18:20:05	6.59MB	57d2a9629298aa6dc2d759fe09da5d13
	chromedriver win32.zip	2019-11-18 18:20:06	4.07MB	9665be96d739035efdf91684f406fdcf
	notes.txt	2019-11-18 18:20:10	0.00MB	c4ebd5d56bbe3948e7fbbf96cfe8a75b

How to Set Up Selenium Grid

- Step 4: Go to <https://github.com/mozilla/geckodriver/releases/tag/v0.26.0>
- Step 5: download gecko driver



▼ Assets 7

 geckodriver-v0.26.0-linux32.tar.gz	2.22 MB
 geckodriver-v0.26.0-linux64.tar.gz	2.28 MB
 geckodriver-v0.26.0-macos.tar.gz	1.91 MB
 geckodriver-v0.26.0-win32.zip	1.37 MB
 geckodriver-v0.26.0-win64.zip	1.46 MB
 Source code (zip)	
 Source code (tar.gz)	

Step 5: Run the Hub

Open a command prompt and navigate to the directory where you copied the selenium-server-standalone file.

Execute the following command:

```
java -jar selenium-server-standalone-<version>.jar -role hub
```

The hub will automatically start-up using port 4444 by default. To change the default port, you can add the optional parameter -port when you run the command. You can view the status of the hub by opening a browser window and navigating to: <http://localhost:4444/grid/console>

Step 6: Run the node

Open a command prompt and navigate to the directory where you copied the selenium-server-standalone file. Type the following command (for chrome):

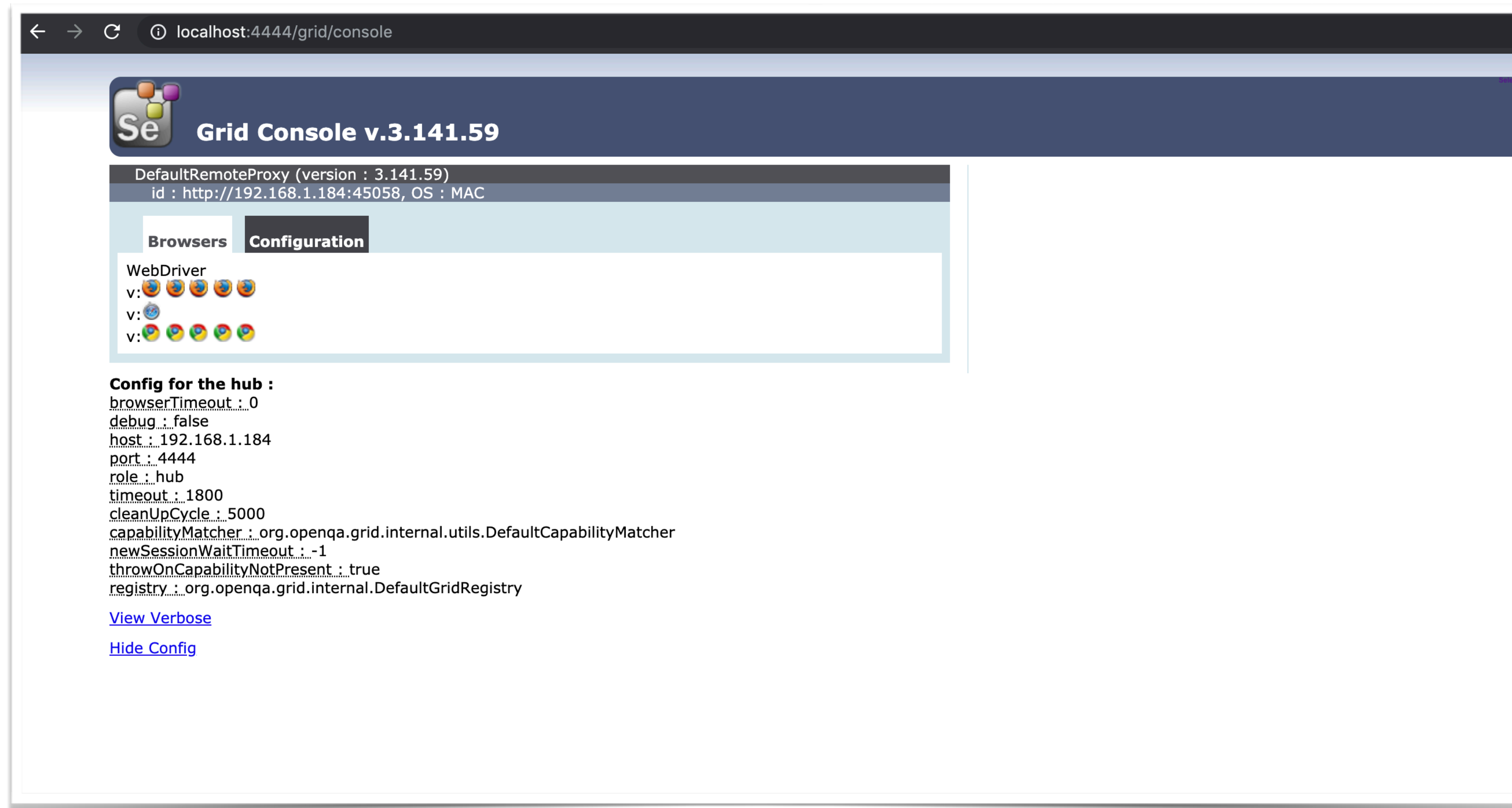
```
java -Dwebdriver.chrome.driver="./chromedriver" -Dwebdriver.gecko.driver="./gecodriver" jar selenium-server-standalone-<version>.jar -role node -hub http://localhost:4444/grid/register
```

Note: The port defaults to 5555 if not specified whenever the "-role" option is provided and is not hub.

For backwards compatibility "wd" and "rc" roles are still a valid subset of the "node" role. But those roles limit the types of remote connections to their corresponding API, while "node" allows both RC and WebDriver remote connections.

Step 7: Check Grid console

Go to localhost:4444/grid/console in order to view your selenium grid console



The screenshot shows the Selenium Grid Console v.3.141.59 interface. The browser address bar displays 'localhost:4444/grid/console'. The page header includes the Selenium logo and the title 'Grid Console v.3.141.59'. Below the header, the 'DefaultRemoteProxy (version : 3.141.59)' is identified with its ID 'http://192.168.1.184:45058, OS : MAC'. The interface has two tabs: 'Browsers' and 'Configuration', with 'Configuration' currently selected. Under the 'Browsers' tab, there are three entries for 'WebDriver' with version icons. The 'Configuration' tab displays a list of settings for the hub, including 'browserTimeout : 0', 'debug : false', 'host : 192.168.1.184', 'port : 4444', 'role : hub', 'timeout : 1800', 'cleanUpCycle : 5000', 'capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher', 'newSessionWaitTimeout : -1', 'throwOnCapabilityNotPresent : true', and 'registry : org.openqa.grid.internal.DefaultGridRegistry'. At the bottom of the configuration list, there are links for 'View Verbose' and 'Hide Config'.

Grid Console v.3.141.59

DefaultRemoteProxy (version : 3.141.59)
id : http://192.168.1.184:45058, OS : MAC

Browsers **Configuration**

WebDriver
v: [Icons]
v: [Icons]
v: [Icons]

Config for the hub :
browserTimeout : 0
debug : false
host : 192.168.1.184
port : 4444
role : hub
timeout : 1800
cleanUpCycle : 5000
capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout : -1
throwOnCapabilityNotPresent : true
registry : org.openqa.grid.internal.DefaultGridRegistry

[View Verbose](#)
[Hide Config](#)

Step 8: Configure Driver class

Create RemoteWebDriver instance. Provide hub URL address and DesiredCapabilities in order to specify node preferences.

```
case "remote_chrome":
    try {
        WebDriverManager.chromedriver().setup();
        ChromeOptions chromeOptions = new ChromeOptions();
        chromeOptions.setCapability(capabilityName: "platform", Platform.ANY);
        driverPool.set(new RemoteWebDriver(new URL(spec: "http:localhost:4444/wd/hub"), chromeOptions));
    } catch (Exception e) {
        e.printStackTrace();
    }
    break;
case "remote_firefox":
    try {
        WebDriverManager.firefoxdriver().setup();
        FirefoxOptions firefoxOptions = new FirefoxOptions();
        firefoxOptions.setCapability(capabilityName: "platform", Platform.ANY);
        driverPool.set(new RemoteWebDriver(new URL(spec: "http:localhost:4444/wd/hub"), firefoxOptions));
    } catch (Exception e) {
        e.printStackTrace();
    }
    break;
```

Configure the nodes by command line

By default, starting the node allows for concurrent use of 11 browsers... : 5 Firefox, 5 Chrome, 1 Internet Explorer. The maximum number of concurrent tests is set to 5 by default.

To change this and other browser settings, you can pass in parameters to each -browser switch (each switch represents a node based on your parameters). If you use the -browser parameter, the default browsers will be ignored and only what you specify command line will be used.

`-browser browserName=firefox,version=3.6,maxInstances=5,platform=LINUX`

Configure the nodes by JSON

```
java -Dwebdriver.chrome.driver="chromedriver" jar selenium-  
server-standalone-<version>.jar -role node -hub http://  
localhost:4444/grid/register -nodeConfig nodeconfig.json
```

Terminologies used in Grid Configuration

role = When launching a node, it will forward the parameter to server on the node like, -role node.

host = Though it not needed and determined automatically. For some network configuration, network with VPN, specifying the host might be necessary.

port = the port the remote/hub will listen on. Default to 4444.

throwIn Capability Not Present = Default value is true; if no proxy is currently registered hub will reject the test request. On contrast, the request queued until a node supporting the capability which is added to the grid.

new Session Wait_Timeout = Default to no timeout (-1)ms after which a new test waiting for a node to become available will timeout else test will throw an exception before starting a browser.

hubConfig = It defines the hub properties in JSON format.

nodeConfig = It defines the node properties in JSON format.

cleanupCycle = It will check the timeout thread in ms.

node Timeout = The timeout in seconds prior to hub automatically ends the test so that browser will be released for another test to run.

browser Timeout= The browser gets hang after defined timeout in ms.

hub = It used to post the registration request using url – <http://localhost:4444/grid/register>

proxy =This will be used to represent the node. By default org.openqa.grid.selenium.proxy.DefaultRemoteProxy.

maxSession = Maximum number of sessions to run multiple tests at same time independently in different browsers.

register Cycle = It defines how often registered node will try to register itself again in ms,however, without restarting the node command file.

nodePolling = How often the hub checks if the node is still alive in ms.