# Selenium Grid

In the current scenario, technology is going through rapid changes. We see a host of new software and lot of existing software being updated frequently. Likewise browsers which we use are also upgrading day by day. Chrome browser reached to its version 57 as of today March’17.

For better user experience, it becomes necessary to validate the application with the different combinations of browsers and also different versions of same browser. Sure, we can have the browsers installed on our box, but then we are limited because we can have only one version of a browser flavor at a given time. Also, it is not advisable to manage different versions of the same browser on a single machine.

***The Problem***

If you’re looking to run your tests on different browsers and operating system combinations but you’re unable to justify using a third-party solution like ***Sauce Labs*** or ***Browser Stack*** then what do you do?

***The Solution***

With ***Selenium Grid*** you can create a network of connected test machines (also called nodes). This network of test machines is controlled by a Hub, using which you can run your tests on different connected nodes. Each node is basically a computer (even a virtual machine) with a combination of Operating system and Browsers. This enables us to create a network of test machines with varying combinations of Operating system and browsers. Using Selenium Grid you can run tests on a variety of Operating System and Browser combinations. Lets understand Selenium Grid in more details below.

## 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 specialise 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 nodes.***

### What is a Hub?

In Selenium Grid, the hub is a computer which is the central point where we can load our tests into. Hub also acts as a server because of which it acts as a central point to control the network of Test machines. The Selenium Grid has only one hub and it is the master of the network. When a test with given DesiredCapabilities is given to Hub, the Hub searches for the node witch matches the given configuration. For example, you can say that you want to run the test on Windows 10 and on Chrome browser with verision XXX. Hub will try to find a machine in the Grid which matches the criterion and will run the test on that Machine. If there is no match, then hub returns an error. There should be only one hub in a Grid.

### What is a Node?

In Selenium Grid, a ***node***is referred to a Test Machine which opts to connect with the Hub. This test machine will be used by Hub to run tests on. A Grid network can have multiple nodes. A node is supposed to have different platforms i.e. different operating system and browsers. The node does not need the same platform for running as that of hub.

### ***How it works?***

First you need to create a hub. Then you can connect (or “register”) nodes to that hub. Nodes are where your tests will run, and the hub is responsible for making sure your tests end up on the right one (e.g., the machine with the operating system and browser you specified in your test).

### ***What is the difference between Grid 1 and Grid 2?***

The Selenium Grid has two versions – the older Grid 1 and the newer Grid 2.

|  |  |
| --- | --- |
| ***Grid 1*** | ***Grid 2*** |
| It has its own remote control which is different from the Selenium RC server. | It is enclosed with Selenium Server jar file. |
| It can only support Selenium RC commands. | It can support both Selenium RC and WebDriver scripts. |
| We need to install and configure Apache Ant first before we can use Grid 1. | In this we do not need to install Apache Ant in Grid 2. |
| We can only automate one browser per remote control. | In this one remote control can automate up to 5 browsers. |

Why the Selenium Grid is used?

With ***Selenium Grid*** you can create a simple infrastructure of various browsers on different operating systems to not only distribute test load, but also give you a diversity of browsers to work with.

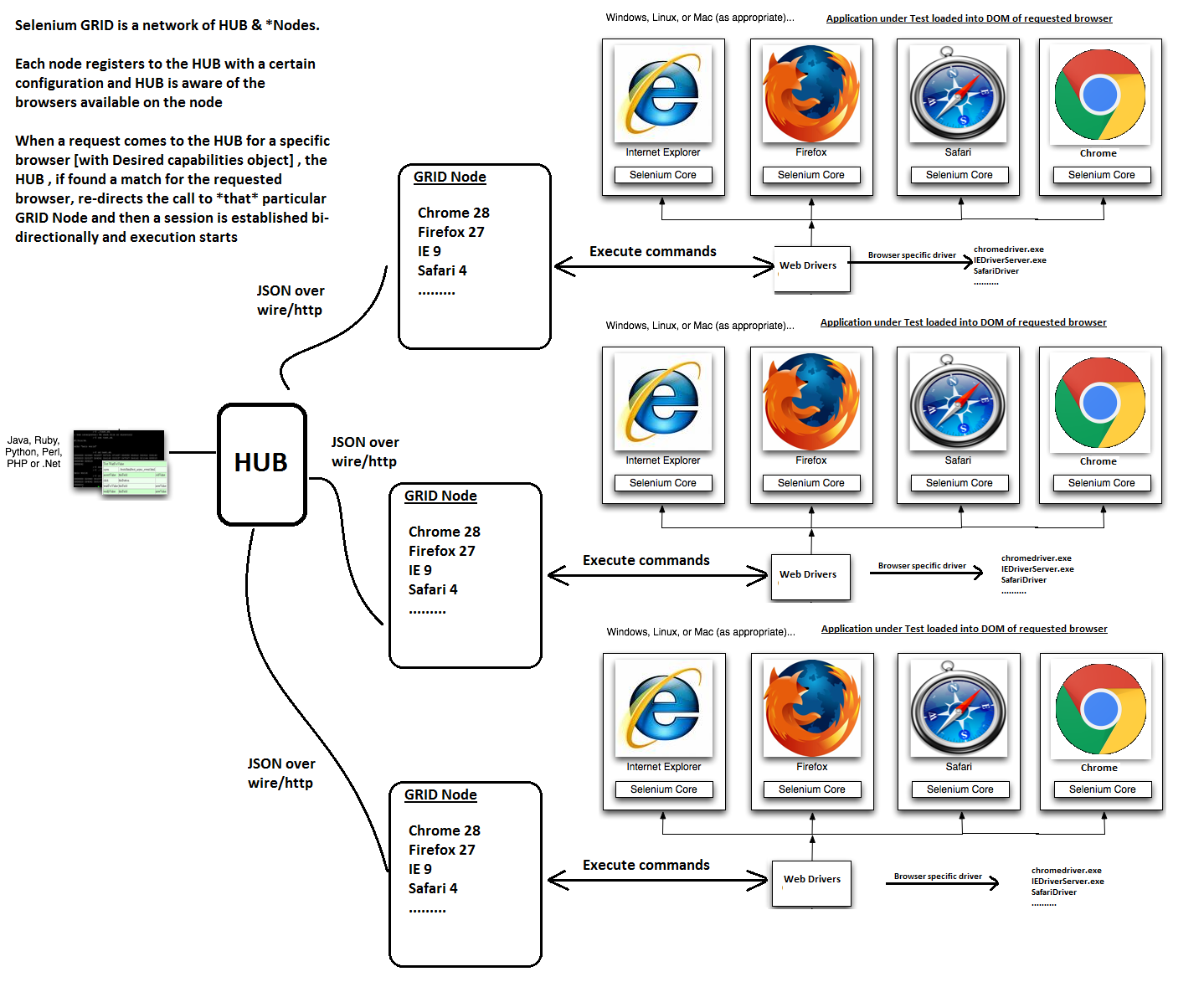
The Selenium Grid is used because of many reasons. Here are a few

* *When we want to run our tests against multiple browsers, the multiple versions of browsers and the browsers running on different operating system.*
* *It is also used to reduce the time taken by the test suite to complete a test pass by running tests in parallel.*

Architecture and RemoteWebDriver WorkFlow

You can use RemoteWebDriver the same way you would use WebDriver locally. The primary difference is that RemoteWebDriver needs to be configured so that it can run your tests on a separate machine. The RemoteWebDriver is composed of two pieces: a client and a server. The client is your WebDriver test and the server is simply a Java servlet, which can be hosted in any modern JEE app server.

* RemoteWebDriveris an implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the RemoteWebDriver server on a remote machine.
* There are two parts to RemoteWebDriver: a server(hub) and a client(node)
  + The RemoteWebDriverserver is a component that listens on a port for various requests from a RemoteWebDriver Once it receives the requests, it forwards them to any of the following: Firefox Driver, IE Driver, or Chrome Driver, whichever is asked.
  + The language-binding client libraries that serve as a RemoteWebDriver The client, as it used to when executing tests locally, translates your test script requests to JSON payload and sends them across to the RemoteWebDriverserver using the JSON wire protocol.
* When you execute your tests locally, the WebDriver client libraries talk to your Firefox Driver, IE Driver, or Chrome Driver directly. Now, when you try to execute your tests remotely, the WebDriver client libraries talk to the RemoteWebDriverserver and the server talks to either the Firefox Driver, IE Driver, or Chrome Driver, whichever the WebDriver client asks for.
* ***Desired Capabilities****: Object in automation code that has requesting browser configuration information*
* ***RemoteWebDriver****: Object in automation code that knows how to communicate remotely with WebDriver / Selenium GRID*



* ***Desired Capabilities****: Object in automation code that has requesting browser configuration information*
* ***RemoteWebDriver****: Object in automation code that knows how to communicate remotely with WebDriver / Selenium GRID*