This Selenium Grid tutorial offers a detailed overview of Selenium Grid, when to use it, and a step-by-step tutorial for set up, installation, and utilization in cross browser testing.

What is Selenium?

Selenium is a framework of automation testing tools, based on the JavaScript framework. Automated Selenium testing is greatly favoured by QAs for replicating end-user actions on websites to monitor their behaviour. It drives the interactions that occur on the target web page and could run them automatically, without requiring manual inputs.

Selenium suite comprises 4 components:

- Selenium Grid
- Selenium IDE
- Selenium RC
- Selenium Webdriver

Let's learn about Selenium Grid in detail.

Featured Articles

Selenium 4: Understanding Key Features

Shift from VMs to Selenium Grid on Cloud for Cross Browser Testing

Desired Capabilities in Appium

Automation Testing | Selenium | Selenium Webdriver

What is Selenium Grid?

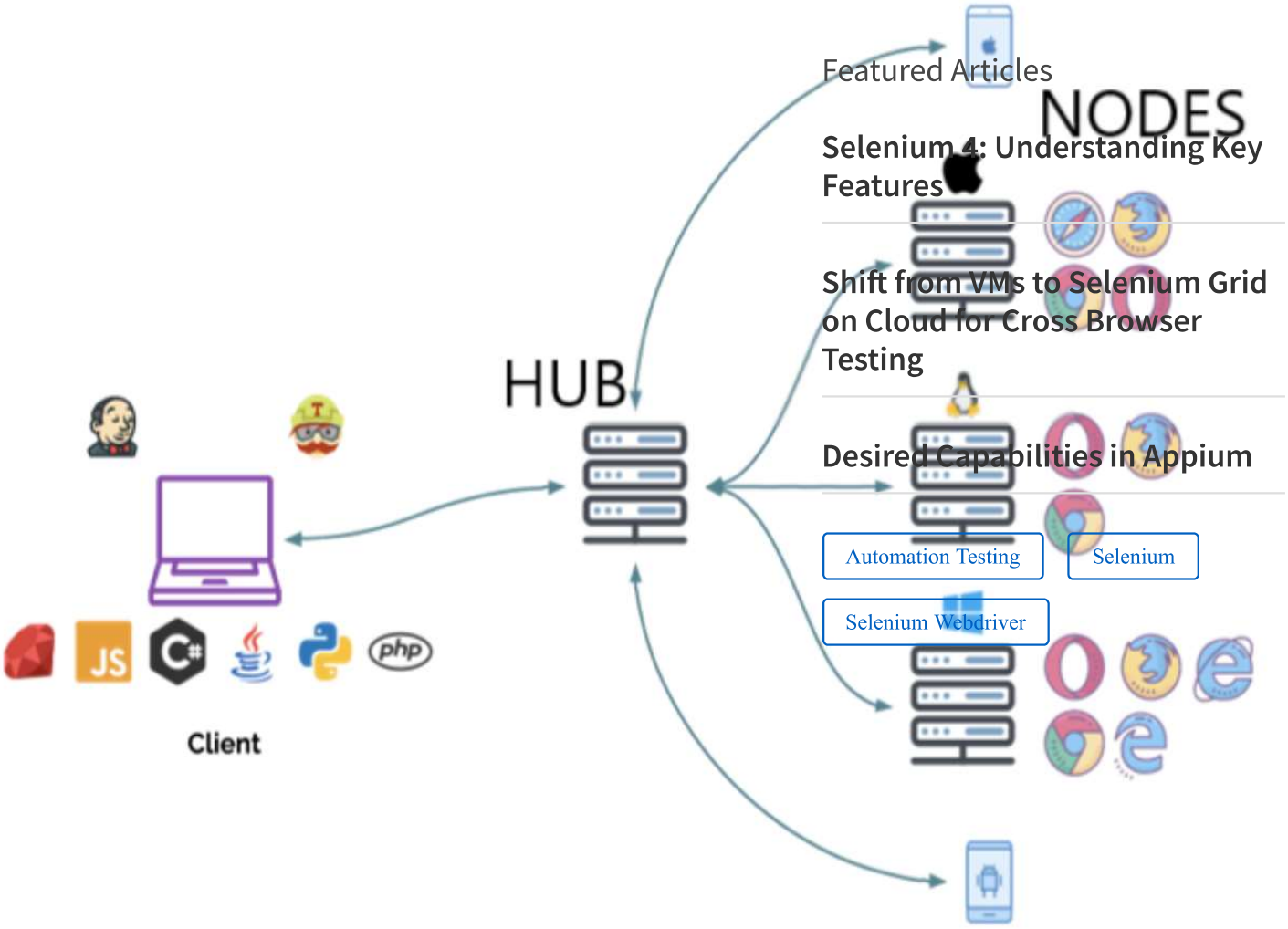
Selenium Grid is a smart proxy server that makes it easy to run tests in parallel on multiple machines. This is done by routing commands to remote web browser instances, where one server acts as the hub. This hub routes test commands that are in JSON format to multiple registered Grid nodes.

Hub enables simultaneous execution of tests on multiple machines, managing different browsers centrally, instead of conducting different tests for each of them. Selenium Grid makes cross browser testing easy as a single test can be carried on multiple machines and browsers, all together, making it easy to analyze and compare the results.

The two major components of the Selenium Grid architecture are:

- **Hub** is a server that accepts the access requests from the WebDriver client, routing the JSON test commands to the remote drives on nodes. It takes instructions from the client and executes them remotely on the various nodes in parallel

- **Node** is a remote device that consists of a native OS and a remote WebDriver. It receives requests from the hub in the form of JSON test commands and executes them using WebDriver



When should testers use Selenium Grid?

Testers should use Selenium Grid in the following circumstances:

- To run tests on multiple browsers and their versions, different devices, and operating systems
- To reduce the time that a test suite takes to complete execution

Selenium Grid improves the turnaround time of the test results. It is especially useful when the test suite is large and takes more time to run. It offers flexibility and expands test coverage within a limited time. Since

the virtual infrastructure is in use, maintenance becomes easy.

Getting started with Selenium Grid browser testing

Here are the steps to follow for Selenium Grid configuration:

Step 1: Installation

Before getting started, download the Selenium Server Standalone package. This package is a jar file, which includes the Hub, WebDriver, and legacy RC that is needed to run the Grid. To get started with Selenium Grid, it is essential to have Java already installed, and set up the environment variables.

Step 2: Start Hub

Hub is the central point in the Selenium Grid that routes the JSON test commands to the nodes. It receives test requests from the client and routes them to required nodes. To set up the Selenium Hub, open command prompt, and navigate to the directory where the Selenium Server Standalone jar file is stored (downloaded in Step 1).

Type the following command

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

This will start the Hub automatically using port 4444 by default. Testers can change the default port by adding an optional parameter port, using

```
-host <IP | hostname>
```

while running the command. Testers need not specify the hostname as it can be automatically determined unless someone is using an exotic network configuration or networking with VPN. In that case, specifying the host becomes necessary.

To view the status of the hub, open a browser window and navigate to https://localhost:4444/grid/console

Step 3: Start Nodes

Whether testers are looking to running a grid with new WebDriver functionality or with the Selenium 1 RC functionality or running both of them simultaneously, testers have to use the same Selenium Server Standalone jar file, to start the nodes. To start nodes open the command prompt and navigate to the directory, where the Selenium Server Standalone jar file is stored.

Type the following command

```
java -jar selenium-server-standalone-<version>.jar -role node -hub https://localhost:4444/g
```

When *-role* option that is provided is not specified, and it is not the hub, the default port is 5555. So, it is important to define the -role to be a node in this case.

Step 4: Configure Nodes

When testers start the nodes, by default, it allows 11 browsers, i.e., 5 Firefox, 5 Chrome, and 1 Internet Explorer for concurrent use. It also allows testers to conduct a maximum of 5 concurrent tests by default.

Testers can change this and other browser settings, by configuring nodes. This can be done by passing parameters to each of the *-browser* switches that represent a node, based on the parameters.

As soon as the *-browser* parameter is used, the default browser settings shall be ignored and only the parameters that are specified in the command line shall be used.

Let us understand this with an example to set 4 Firefox version 4 nodes on a Windows machine.

```
-browser browserName=firefox,version=4,maxInstances=4,platform=WINDOWS
```

In a case where the machine has multiple versions of Firefox, map the location of each binary to the compatible version on the same machine.

Let us understand this by the following example where there are two versions of Firefox, namely 3.6 and 4 on the same Windows machine that have to be used at 5 and 4 instances respectively.

```
-browser browserName=firefox,version=3.6,firefox_binary=/home/myhomedir/firefox36/firefox,m
```

This way, testers can configure the nodes as per their cross browser testing requirements, using desired combination of browsers, their versions, and operating systems.

Featured Articles

Selenium 4: Understanding Key Features

Shift from VMs to Selenium Grid on Cloud for Cross Browser Testing

Desired Capabilities in Appium

Automation Testing Selenium Selenium Webdriver

Step 5: Using Selenium Grid to run tests

Once the Selenium Grid setup is done by following the above 4 steps, testers can access the grid to run tests. If Selenium 1 RC nodes are being used, testers can use *DefaultSelenium* object and pass the same in the hub formation using the following command.

```
Selenium selenium = new DefaultSelenium("localhost", 4444, "*firefox", "https://www.browser
```

If testers are using Remote WebDriver nodes, they must the *RemoteWebDriver* and *DesiredCapabilities* objects to define the browser, version, and platform. For this, create the target browser capabilities to run the test on:

```
DesiredCapabilities capability = DesiredCapabilities.firefox();
```

Once created, pass this set of browser capabilities into the *RemoteWebDriver* object:

```
WebDriver driver = new RemoteWebDriver(new URL("https://localhost:4444/wd/hub"), capability
```

Once this is done, the hub would assign the test to a matching node, if all the requested capabilities meet. To request any specific capabilities on the grid, specify them before passing it to the WebDriver object in the following pattern:

```
capability.setBrowserName();
capability.setPlatform();
capability.setVersion()
capability.setCapability(,);
```

If these capabilities do not exist on the Grid, the code returns no match and thus the test fails to run.

Let us understand this using an example, considering a node is registered with the setting:

```
-browser browserName=firefox,version=4,maxInstances=4,platform=WINDOWS
```

Then, it is a match with the following set of capabilities defined for the test:

```
capability.setBrowserName("firefox" );
capability.setPlatform("WINDOWS");
capability.setVersion("4");
```

It would also match with the following set of capabilities defined for the test"

```
capability.setBrowserName("firefox" );
capability.setVersion("4");
```

Note that the capabilities which are not specified for the test would be ignored, such as in the above example where the platform parameter is not specified and it gets a match.

Using these steps, testers can easily set up, configure, and perform tests on Selenium Grid for concurrent execution of test suites.

Conclusion