**How to Run Test Cases using Selenium Grid**

Selenium Grid is a tool that distributes the tests across multiple physical or virtual machines so that we can execute scripts in parallel (simultaneously). It dramatically accelerates the testing process across browsers and across platforms by giving us quick and accurate feedback.

Selenium Grid allows us to execute multiple instances of WebDriver or Selenium Remote Control tests in parallel which uses the same code base, hence the code need NOT be present on the system they execute. The selenium-server-standalone package includes Hub, WebDriver, and Selenium RC to execute the scripts in grid.

Selenium Grid has a Hub and a Node.

* Hub − The hub can also be understood as a server which acts as the central point where the tests would be triggered. A Selenium Grid has only one Hub and it is launched on a single machine once.
* Node − Nodes are the Selenium instances that are attached to the Hub which execute the tests. There can be one or more nodes in a grid which can be of any OS and can contain any of the Selenium supported browsers.

Architecture

**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).

**Pre-requisites**

1) We Should have drivers & browsers on Hub & Node machines.

2) Make local system as HUB server , then register all the VM's(nodes) with Hub

Download Selenium Standalone Server jar from  **https://goo.gl/4g538W**

\* we should have same jar on all the Nodes

Step1: Create Hub in your local Windows system.

  java -jar selenium-server-standalone-3.9.1.jar -role hub

Step2: Register your remote node with Hub (Windows/Chrome)

  java -Dwebdriver.chrome.driver="C:\Drivers\chromedriver\_win32\chromedriver.exe" -jar selenium-server-standalone-3.9.1.jar -role node -hub <http://192.168.13.1:4444/grid/register/>

Check Hub is running with One Node(i.e Windows) on browser

<http://localhost:4444/grid/console>

Step3: Register your remote node with Hub (Linux/Firefox)

  java -Dwebdriver.gecko.driver="/home/pavan/Drivers/geckodriver" -jar selenium-server-standalone-3.9.1.jar -role node -hub <http://192.168.13.1:4444/grid/register/>

Check Hub is running with Two Nodes(i.e Windows and Linux)

<http://localhost:4444/grid/console>

**Test Case 1:  Hub and Node are same machines.**

**Environment: Chrome on Windows**

import java.net.MalformedURLException;

import java.net.URL;

import org.openqa.selenium.By;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.testng.annotations.Test;

public class LogintestOnWindowsusingChrome {

static WebDriver driver;

@Test(priority = 1)

void setup() throws MalformedURLException {

String nodeURL = "http://192.168.13.1:5555/wd/hub"; // The URL will be

//  IP Address of Hub Machine + Hub Port + /wd/hub

// "http://192.168.13.1:4444/wd/hub"

// Here Hub and Node are same machines

DesiredCapabilities cap = DesiredCapabilities.chrome();

cap.setBrowserName("chrome");

cap.setPlatform(Platform.WIN10);

driver = new RemoteWebDriver(new URL(nodeURL), cap);

}

@Test(priority = 2)

void login() {

driver.get("http://practice.automationtesting.in/my-account/");

driver.findElement(By.id("username")).sendKeys("pavanoltraining");

driver.findElement(By.id("password")).sendKeys("Test@selenium123");

driver.findElement(By.name("login")).click();

String captext = driver.findElement(By.xpath("//\*[@id='page-36']/div/div[1]/div/p[1]")).getText();

if (captext.contains("pavanoltraining")) {

System.out.println("Test passed");

} else {

System.out.println("Test failed");

}

driver.close();

}

}

**Test Case 2:  Hub and Node are different machines.**

**Environment: Firefox on Linux**

import java.net.MalformedURLException;

import java.net.URL;

import org.openqa.selenium.By;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.testng.annotations.Test;

public class LogintestOnLinuxusingFirefox {

static WebDriver driver;

@Test(priority = 1)

void setup() throws MalformedURLException {

String nodeURL = "http://192.168.13.1:4444/wd/hub"; // The URL will be

// IP Address of Hub

// Machine + Hub

// Port + /wd/hub

// "http://192.168.13.1:4444/wd/hub"

DesiredCapabilities cap = DesiredCapabilities.firefox();

cap.setBrowserName("firefox");

cap.setPlatform(Platform.LINUX);

driver = new RemoteWebDriver(new URL(nodeURL), cap);

}

@Test(priority = 2)

void login() {

driver.get("http://practice.automationtesting.in/my-account/");

driver.findElement(By.id("username")).sendKeys("pavanoltraining");

driver.findElement(By.id("password")).sendKeys("Test@selenium123");

driver.findElement(By.name("login")).click();

String captext = driver.findElement(By.xpath("//\*[@id='page-36']/div/div[1]/div/p[1]")).getText();

if (captext.contains("pavanoltraining")) {

System.out.println("Test passed");

} else {

System.out.println("Test failed");

}

driver.close();

}

}