**Day 7 :**

**Selenium Grid : Grid Architecture and Demo**

Selenium Grid is a tool in the Selenium suite that facilitates **parallel test execution** by allowing tests to run on multiple machines or browsers concurrently, providing a scalable and efficient solution for distributed test automation.

Selenium suite comprises four components:

1. Selenium Grid

2. Selenium IDE

3. Selenium RC

4. Selenium Webdriver

**Selenium Grid**

○ Selenium Grid is a feature in Selenium that allows you to run test cases in different machines across different platforms.

○ The control of triggering the test cases is on the local machine, and when the test cases are triggered, they are automatically executed by the remote machine. Suppose you have 5 test cases.

○ Your local machine is running multiple applications, so you want to run your test cases in a remote machine. You need to configure the remote server so that the test cases can be executed there.

## 

## 

## 

## 

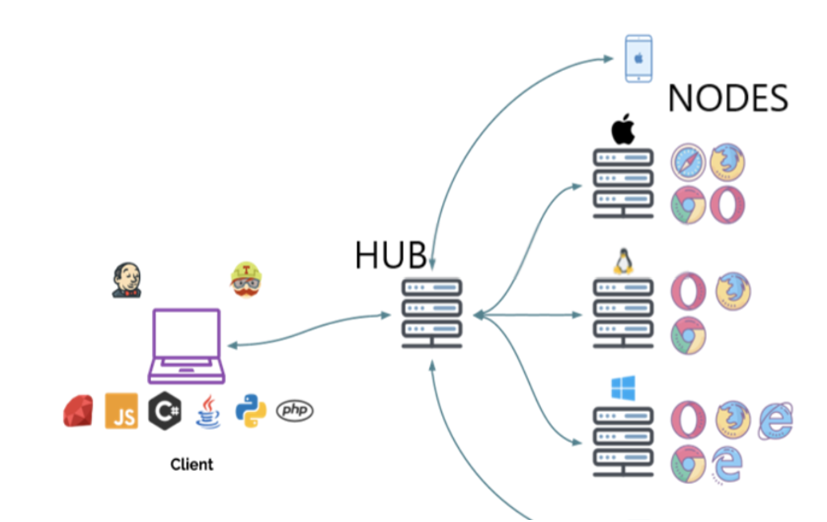
## 

## 

## 

## 

## **Architecture of Selenium Grid**



### 

### **Hub**

○ A Hub is a central point or a local machine that receives all the test requests and distributes them to the right nodes. The machine which actually triggers the test case known as Hub.

○ There can be only one hub in a selenium grid.

○ The machine which is containing the hub triggers the test case, but you will see the browser being automated on other machines.

### 

### 

### 

### 

### 

### **Node**

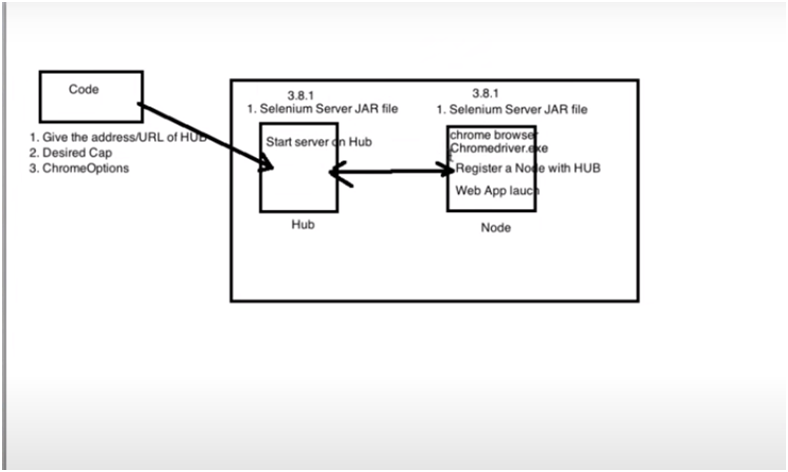
Nodes are the selenium instances which **will execute the test cases that you loaded on the hub.**

Nodes can be launched on multiple machines with different platforms and browsers.

Simple Selenium Grid architecture to setup both Hub and node in the same localhost.

Name of the Jar file is –**Selenium standalone server.jar**

With the help of this Jar file we can setup both Hub and Nodes.



**Brief Working :**

**Before writing the code Process :**

**1)** **Download Seleniumstandloneserver.**

**2)** **We have to install Hub server in our Local.**

**3)** **We have to install Node server in our Local.**

**In the code/ Eclipse :**

1. In code, we should give the address/URL of Hub.

2. We should specify the DesiredCapabilities of the platform. Browser, version for Node.

3. We should use ChromeOptions in our code.

**For the Hub :**

1.For Hub , we need a Selenium standalone server jar file to be specified.

2. We need to start the Hub server on Hub.

**For Node to execute in chrome browser :**

1. For Hub , we need a Selenium standalone server jar file to be specified.

2. Chrome browser should be installed.

3. Chromedriver.exe should be present in node.

4. We need to Register a node with Hub.(Connecting Hub with node is called Registering.)

**Implementation :**

**Step 1** : Download Selenium standalone server jar file and keep it in a folder.

**Step 2** : Open command prompt at the location of your Selenium server jar file/ change path after opening cmd.

**Step 3 : Hub Configuration**

type the command to install Selenium Hub :

Java -jar selenium-server-standalone-3.141.59.jar -role hub (include .jar in the command)

**The following message will be displayed in your cmd.**

20:12:29.036 INFO [Hub.start] - Selenium Grid hub is up and running

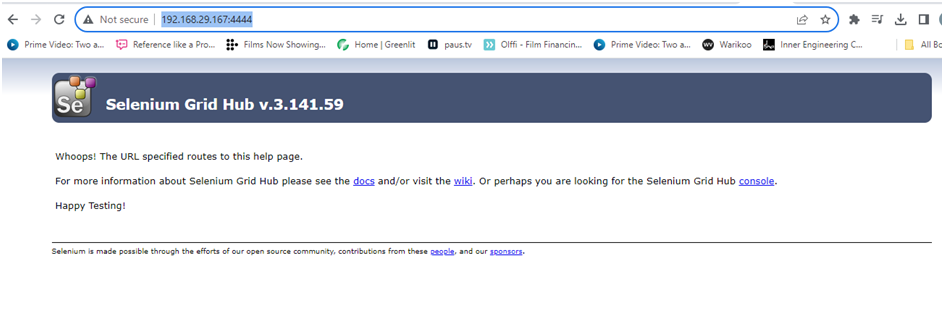
17:21:26.537 INFO [Hub.start] - Nodes should register to http://192.168.29.167:4444/grid/register/

17:21:26.538 INFO [Hub.start] - Clients should connect to<http://192.168.29.167:4444/wd/hub>

**Important note : Don't ever close your cmd after installing Hub, chances are there the hub config might be lost.**

The marked address is the address of your hub in your localhost.

Go to your browser and type the same address the following screen should be visible.



This confirms that your **Hub is up and running**

**Step 4** : **Node Configuration** .

Open another instance of cmd(Don't close the Hub installed cmd instance)

Type the following command :

java -Dwebdriver.chrome.driver="C:\Users\Dell\Documents\seleniumgrid\chromedriver.exe" -jar selenium-server-standalone-3.141.59.jar -role node -hub http://192.168.56.1:4444/grid/register

After entering the command , we will get the following msg in cmd

20:17:45.784 INFO [GridLauncherV3.lambda$buildLaunchers$7] - Selenium Grid node is up and ready to register to the hub

20:17:45.881 INFO [SelfRegisteringRemote$1.run] - Starting auto registration thread. Will try to register every 5000 ms.

20:17:46.284 INFO [SelfRegisteringRemote.registerToHub] - Registering the node to the hub: http://192.168.29.167:4444/grid/register

20:17:46.386 INFO [SelfRegisteringRemote.registerToHub] - The node is registered to the hub and ready to use

And also in other cmd the following msg will be displayed :

20:17:46.385 INFO [DefaultGridRegistry.add] - Registered a node http://192.168.29.167:22472

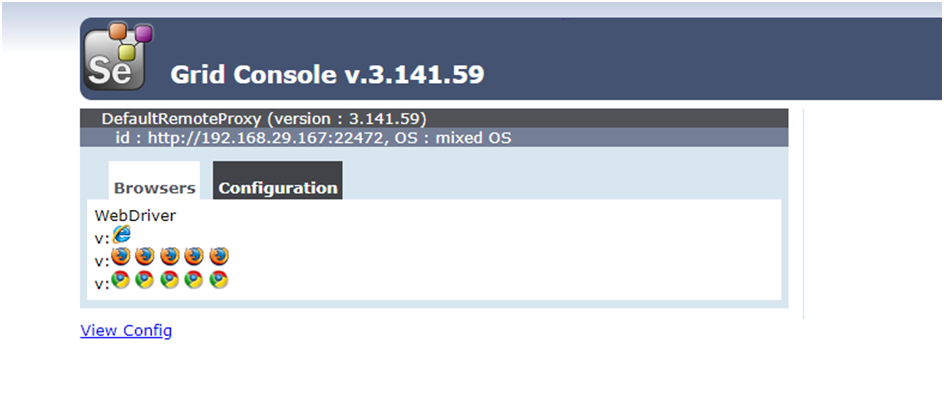
The node is registered in the 22472 port mentioned above , and hence :

**Hub address** : 192.168.29.167:4444

**Nodes address** : http://192.168.29.167:22472

And the following page should be displayed after clicking on Console option :

This is the **Selenium Grid Console.**



**Step 5**

Now, It’s time to write the code in Eclipse

Create a Maven project and add selenium and testNg dependency (Avoid this step if already present)

**Important note : In your Maven Project, use the same dependency and version of the selenium which you have used for the standalone server jar file.**

**Step 6 :** In java class, add Desired Capabilities, set Capability browser and OS.

// Define Desired Capabilities

DesiredCapabilities des = new DesiredCapabilities();

des.setBrowserName("chrome");

des.setPlatform(Platform.WIN10);

**Step 7** : ChromeOptions Definition

ChromeOptions options = new ChromeOptions();

**Step 8 :**  Merging the ChromeOptions with the Desired capabilities, that the node is supposed to execute.

options.merge(des);

**Step 9** : We execute the Test with RemoteWebDriver providing the HubURL

String HubUrl = "http://192.168.29.167:4444/wd/hub";

WebDriver driver = new RemoteWebDriver(new URL(HubUrl),options);

driver.get("<https://www.selenium.dev/>");

Java class complete code :

public class GridTest {

public static void main(String[] args) throws MalformedURLException {

// Define Desired Capabilities

DesiredCapabilities des = new DesiredCapabilities();

des.setBrowserName("chrome");

des.setPlatform(Platform.WINDOWS);

//Chrome Options Definition.

ChromeOptions options = new ChromeOptions();

// here we are merging the ChromeOptions with the Desired capabilities, that the node is supposed to execute

options.merge(des);

// We execute the Test with RemoteWebDriver providing the HubURL and

String HubUrl = "http://192.168.29.167:4444/wd/hub";

WebDriver driver = new RemoteWebDriver(new URL(HubUrl),options);

driver.get("https://www.selenium.dev/");

System.out.println(driver.getTitle());

}

}

**Introduction to BDD-Cucumber**

**BDD – Behavioral Driven Development.**

**Approach to Develop,Test, maintain, Deliver the Software.**

**BDD is not a framework**

# **What is BDD?**

BDD is a way for software teams to work that closes the gap between **business people** and **technical people.**

BDD – 3 Amigos.

3 Amigos are – **Developers , Testers , Business Analyst/ Product Owner.**

Between these 3 Amigos , there is a constant Communication being conducted on daily basis.

Example –

**Redbus**

Encouraging collaboration across roles to build shared understanding of the problem to be solved

● Working in **rapid, small iterations** to increase **feedback** and the flow of value

● Producing system **documentation** that is automatically checked against the **system’s behaviour**

We do this by focusing collaborative work around concrete, real-world examples that illustrate how we want the system to behave. We use those examples to guide us from concept through to implementation, in a process of **continuous collaboration.**

# **Programs :**

package griddemo;

import java.net.MalformedURLException;

import java.net.URL;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

public class GridTest {

public static void main(String[] args) throws MalformedURLException {

// Step 1- Create object of Desired Capabilities

DesiredCapabilities des = new DesiredCapabilities();

//Step -2 Through Desired Capabilities we set the browser name and platform name.

des.setBrowserName("chrome");

des.setPlatform(Platform.WINDOWS);

// step 3 - I have to create Object of ChromeOptions class

ChromeOptions op = new ChromeOptions();

//Step 4 - Merging ChromeOptions with the Desired Capabilities

op.merge(des);

// We are making use of ChromeOptions class to merge the desired capabilities with

// the chromeoptions reference and mentioning the same in RemoteWebDriver

// Step 5 - Store the HubURL in a string

String HubURL="http://192.168.56.1:4444/wd/hub";

try {

// Step 6 - Invoke the browser with RemoteWebDriver

//and pass the HubuRL and chromeoptions instance using URL object.

// In the First parameter of RemoteWebDriver we are passing the details of Hub.

// IN second parameter we are passing the details of the node.

WebDriver driver = new RemoteWebDriver(new URL(HubURL),op);

driver.get("https://www.amazon.in/");

System.out.println(driver.getTitle());

}

catch (Exception e) {

// TODO: handle exception

}

}

}